Waiting for Blue Robot

How To Make an Open-Source Community-Driven Project, Manage It and To Remain a Human

Written by Andrii Doroshenko (Xrayez), co-author of Godot Engine.

Looking for TL;DR?

Description

Software engineering can be tough, no doubt. Managing an open-source, community-led project presents its own set of challenges. The good news? We don't have to act like robots!

Delve into effective strategies for enhancing collaboration among people striving for a shared objective. Reveal the detrimental impact of poor communication regarding your project's vision, mission, goals, non-goals, purpose, direction, intention, principles, ideology,

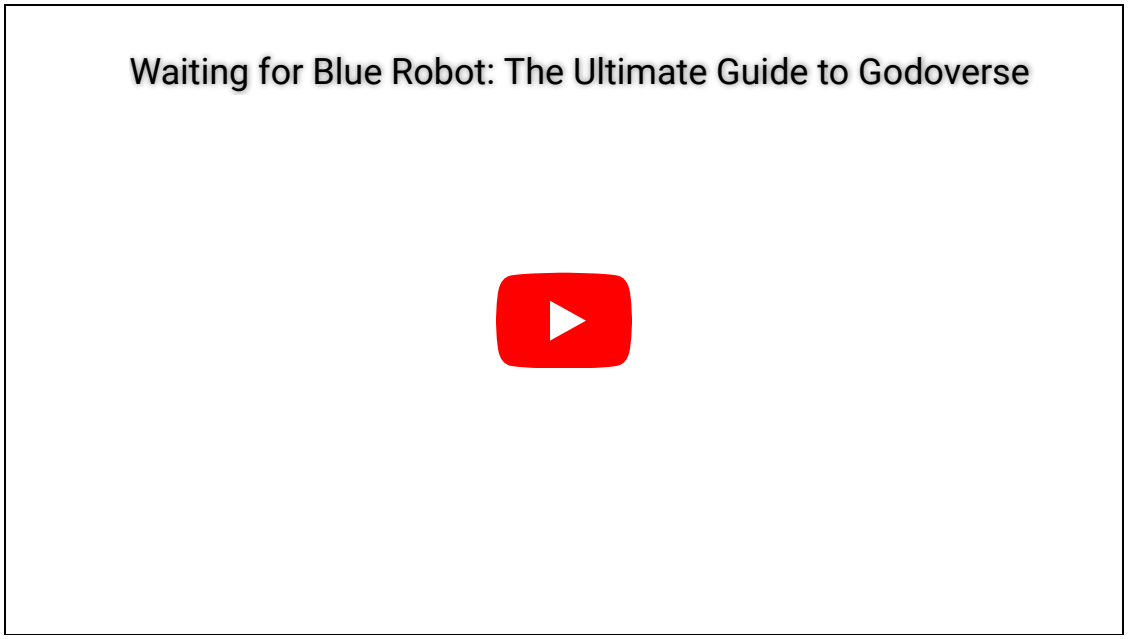
philosophy, etc. By neglecting these crucial elements, the long-term success of your project may be compromised.

It's no longer surprising that one of the most notable open-source projects where these ideas are overlooked and disregarded already exists – it's called Godot Engine. This book serves as a post-mortem analysis of Godot as an allegedly community-driven project, focusing on its fanatical nature.

TL;DR

The following satirical piece serves as the ultimate guide to all things Godot.

Waiting for Blue Robot: The Ultimate Guide to Godoverse



For more satire, read Interlude: Becoming a Blue Robot.

Notable Reviews and Mentions

The Abysmal Critic

DMG Gaming Podcast

I'm a co-author of @GodotEngine! I predicted what's happening to Godot years ago. I'm well-versed in all the Godot issues, from technical aspects to management problems. If you haven't read the book I wrote yet, you're missing out! 😃#Godot #Wokot #TruthAboutGodot #AMA pic.twitter.com/Vw4PRdt2Eg

— Andrii Doroshenko 🇺🇦 (@Xrayez) October 2, 2024

Independent Developers

I have a good laugh on this, even if it is rather a tragedy than a comedy. #indiedev #gamedev #indiegamedev #GODOT #indiegame Save your time, use something that is not a messy toy without vision. I am so sorry for all the people who are tricked into contributing or using it...😑 https://t.co/cTWMzVPOxJ

— Łukasz Śliwiński🔶 (@slizgi) March 25, 2024

At first it was funny but then it just became really sad. #godotengine has a lot of potential and now it is all going to ruin bc of how greedy and incompetent the leaders are. Definitely worth a watch!https://t.co/wnf4zTcURo

— Abruh (@abrasivetroop) March 24, 2024

> Fantastic and concise description of what it is to work on a toxic project. https://t.co/IcUncSor4I
>
> — tekpulse (@tekpulse197225) September 30, 2024

> Very comprehensive. Good read.
>
> — Salvakiya (@Salvakiya) September 29, 2024

> I have to admit. Recent events, do showcase that you weren't wrong about everything.
>
> — Engjell Rraklli (@RraklliEngjell) September 30, 2024

> 💡 *Democracy thrives in the light. Share your testimonies and join the public discussion! The archived version of the book is available as a single-paged HTML for printing and reading.*

About the Author

Andrii Doroshenko is a software developer specializing in game development. In 2017, he came across Godot Engine. In 2018, he earned a bachelor's degree in Computer Science from Kyiv National University of Culture and Arts. He played a key role in the development and support of Godot, actively contributing to Godot's development for five straight years on a voluntary basis. In 2021, he officially took on the role of a Godot Engine maintainer.

Andrii is the co-author of Godot Engine. In 2022, he ranked in the Top 20 contributors by the number of changes to Godot Engine. He has submitted over 200 pull requests and proposed dozens of popularly demanded features. He revamped the unit testing system in Godot, turning it into a fully functional feature. He has played a crucial role in improving Godot's build system and is the brains behind the custom modules feature.

Among other notable contributions, he implemented features like boolean operations on polygons, as seen in the Geometry singleton. Many of the destructible features you come across in games made with Godot are there thanks to Andrii's contributions to Godot:

> Destructible terrain with @godotengine! I worked really hard on this one!#Godot #MadeWithGodot #StopWaitingForGodot #MadeInGodot #CancelGodotEngine #GodotEngine pic.twitter.com/5Fjd4KK1k0
>
> — Andrii Doroshenko (@Xrayez) May 16, 2022

License

Preface

I'm not brave, I'm just afraid of the alternative.

Throughout my years-long experience as a Godot Engine contributor, I've been noticing strange contradictions with respect to Godot's development and decision-making processes. For quite some time, I used to think that those inconsistencies were a byproduct of something which I didn't fully understand yet. Perhaps that's why Godot followers constantly joked about *waiting* for Godot. I decided to ignore those strange discrepancies at that time, expecting that the entire system will be eventually settled, so I put my trust in Godot regardless, especially as I've seen how other devoted contributors were accepting decisions without casting any doubt on Godot leadership.

Notwithstanding, the apparent lack of decided focus in Godot resulted in accumulation of disparities, to a point where it reached its climax to be evident. Me and other curious contributors have figured that these contradictions hurt Godot's workflow and integrity, so I decided to discuss them with the community, and such discussions have received mostly positive feedback, especially from the community outside of Godot. To quote some of them:

> *"It's been a great relief to see your commentary on Godot, as it reflects how I felt about their community and leadership during the time I tried their software. It always felt off, and I couldn't understand why some incredibly opinionated and wrongheaded decisions were held too strongly."*

However, every attempt at clarifying Godot's development philosophy and governance model in order to resolve existing contradictions and inconsistencies has resulted in a great push-back from Godot's *toxic leadership*, which led to eventual exclusion of myself from all Godot's community spaces in spite of my merit as seen by Godot community.

Since then, I've been talking about how Godot's *toxic leadership* causes unnecessary conflicts that can be easily avoided or at least minimized within the inner circle of core developers and community of users in general, and how such *toxic leadership* promotes unquestionable attitude that discourages critical thinking, which is definitely a problem that must be solved.

You might ask, "How could it be that a seemingly welcoming community could have *toxic leadership*?" Most of the things that you see about Godot on the surface is mostly a facade, and hopefully you'll understand why by the end of this book.

By cheating and exploiting the human desire for belonging, we violate autonomy of the

human will. Therefore, this kind of behavior is unethical. For this reason, it is also my ethical duty to write this book. In doing so, I hope to encourage people to exercise their self-determination within open-source communities and in everyday life.

People kept asking me the same questions, therefore this book is the answer to all of those accumulated and repeated questions so far. This work consolidates everything I've learnt from within and outside of Godot Engine community on this topic.

How to read this book

The task here is stimulate the critical thinking process.

You are free to read chapters for the topics that interests you. While not strictly necessary, I recommend reading the chapters in the specified order, so that you get a more coherent understanding.

Due to the nature of the concepts mentioned so far, this book makes use of analogies, where similar ideas overlap from the outside of the subject, like politics, sociology and psychology, so be prepared to ponder upon things without jumping to conclusions.

This book may not provide definitions to all of the concepts outside of the subject, therefore you may need to refer to other sources of information. If something feels a little bit off or out of place, it might be the case that you need to read the rest of the chapters.

Please note that the information presented in this book, particularly in the Organizational Culture section, may appear obvious or mundane to some readers. However, if you happen to hold an anti-corporate sentiment, this information could be beneficial and relevant to you, especially for maintainers of Open Source projects.

Conventions

Throughout this book, you'll see banners such as these:

💡 Water is wet!

These are used to clarify, summarize, consolidate, or ease the comprehension of new concepts.

New and important terminology is denoted with ***bold italic*** font. You can refer to Glossary page for some definitions. Terminology may be specialized in accordance to the context of this book.

The contents labeled with keywords such as *Waiting* or *Blue Robot* refers to Godot Engine. 😉

When referring to Juan Linietsky, the co-creator and lead developer of Godot Engine, his name may be used in a collective sense. For example, the phrase "*Juan seldom completes features*" may be generalized to a larger group of loyal Godot maintainers, also known as

collective Juan.

Note to Godot followers

If you're a Godot follower reading this book, please take this as an opportunity to reflect upon what happens in Godot community. You may feel highly repulsed by ideas presented here. But try to persevere: doing so may yield useful insights for yourself and the rest of the open-source community, especially when you choose to share and discuss those insights with others, and, if necessary, take appropriate action. The fact that you read this makes you a curious person, so I really appreciate this!

Introduction

As software developers, we tend to deal with technical things for the most part. There are numerous reasons why some of us might have chosen this path. But one fact is certain: some things cannot be described in technical terms alone. Some may say that people are just biological machines, yet most our decisions are *not* driven just by logic, but also by our feelings and emotions, where feelings has a large influence in our decision-making processes.

While we can achieve something ourselves, we can only go as far as our sole efforts allow. That's why humans have learnt to come together and collaborate to achieve a common goal. Due to the above, the human factor is what actually determines the end result. Nothing too extraordinary about it, to be honest!

The ability to collaborate in software development industry is no exception to above statement. Specifically, collaboration becomes a very important and outstanding aspect of the open-source software movement and its ethos. Due to this, the subject of this book is mainly focused on concepts like *development philosophy*, *community-led development*, *governance model*, and less about technical side of things. Godot Engine as an open-source project is used as a prominent example where poor decisions are caused by lack of understanding, incomplete and/or misleading disclosure of such vital information associated with those concepts to the worldwide community.

Given the importance of those concepts and being prerequisites for all truly successful open-source *community-led* projects out there, this book mostly reveals contradictory aspects of Godot Engine, where these concepts are neglected, ignored or even rejected by Godot's *toxic leadership*, and hopefully you're going to see why this makes Godot Engine destructive in relation to human values as a consequence.

This book challenges beliefs about what constitutes *community-driven* development approach in the context of open-source community, and *toxic leadership* which exploits the *community-driven* image for attaining unjust advantage by means of undue influence. Specifically, this books shows that Godot Engine's development is *not* community-driven, nor based on any notion of *democracy*, *meritocracy*, or even *do-ocracy*, contrary to Godot followers' existing assumptions about project's de-facto decision-making process. This, of course, is not the only issue covered in this book.

If you'd like to build a healthy and trustworthy open-source project where people are not afraid to express their feedback, where individuals can express their authentic self, their conscience, creativity and critical thinking, this book is for you.

Organizational Culture

Values, Assumptions, Beliefs, Expectations

💡 If you see an apple, you *expect* to eat an apple, not an orange, and especially not a lemon!

A lot of contributors who invest in open-source projects may be disconnected in terms of unpaired ***values***, ***assumptions***, ***beliefs***, and ***expectations*** pertaining to a project, also known as *VABE*s[1]. Culture is a set of shared VABEs.

For the most part, this lack of connection is not contributors' fault. In order to succeed, project's leadership must be able to properly convey their *values* to others right from the start. Doing so allows to complete the entire puzzle, where each piece of the puzzle is comprised of individual ideas and solutions provided by contributors who share similar *values*, creating a positive feedback of growth. As long as we can communicate our *values* to the worldwide community properly, the better and more effective our interactions become.

The following lists consequences of *not* conveying our values to the worldwide community.

Wasted resources

> *"Why waste time on these ephemeral concepts? Aren't decisions are made on a case by case basis?"*

These kind of questions imply a particular attitude. The irony is that someone who knows the "ephemeral" word will likely not ask this question in the first place!

Nevertheless, I'd like those people to pay attention to shortcomings of having such a *belief system*. Ask yourself whether you'd like to waste your time on any of the things below instead:

- constantly *disagree* without an end;
- make contributors feel disappointed by *rejecting* their ideas;
- *ignore* contributors and make them feel that their ideas are worthless.

Unless you're someone who has no ability for empathy and/or enjoys being a contrarian for the sake of it, doing those things above will most likely make you and everyone else feel exhausted. The time spent on unnecessary conflicts could be better spent on realizing your project's vision instead! Time is the most valuable resource that we have in our possession, don't you think?

Note that disagreement is not a bad thing in and of itself. Disagreements (or rather, differences in opinion) are at the heart of every fruitful discussion. This challenges different viewpoints that may lead to actual consensus. Disagreements only become an issue when they hinder the progress.

Killed innovation

> "*Ideas are cheap. Isn't execution far more important?*"

Remember that if someone's idea doesn't go in alignment with your project, it doesn't necessarily mean that an idea is worthless. It simply means that such an idea can have a potential elsewhere.

Yes, some ideas may be truly stupid and even destructive. But if you find yourself needing to *kill ideas*[2] that you identify as potentially destructive for your project, why not just prevent this phenomenon from emerging in the first place?

When contributors come to your project and write elaborate proposals, and you end up rejecting their ideas, you *kill ideas* in a bad way, because those kind of ideas could've manifested as constructive given other opportunity and/or context.

If we talk about human values specifically, it has less to do with ideas themselves, but more about the danger of killing someone else's enthusiasm. This is a crime against creative personality which hurts your image as a consequence.

The sad reality is that most leaders may not even realize that they are constantly killing ideas left and right. Why this happens? The need for rejecting and ignoring ideas arises when our own *values* clash with someone else's polarized *values*, which create difficult-to-resolve conflicts. But in order to identify drastically contrasting *values*, they have to be uncovered before all else! These kind of conflicts are usually destructive rather than constructive, so it's sane to prevent destructive conflicts.

You may respond that you have no control over what people propose, which is a natural train of thought. But we can always *influence* and *direct* individuals. Let me assure you that

there does exist ways to minimize the flood of proposals, which is to define Development Philosophy, principles, etc.

Betrayal of trust

> *"Wouldn't this lead to divisiveness?"*

When contributors know exactly what kind of values the project follows, they can easily see for themselves whether:

- it's the right project that they are looking for;
- their efforts will be recognized and actualized.

Otherwise, contributors won't gain what they initially *expected* from the project, which leads to disappointment and eventual betrayal of trust, and this alone is much worse than the irrational fear of community division, because betrayal of trust is a guaranteed way to divide the community. Trust is quite fragile and it may be difficult to restore it.

Therefore, it's very crucial to ensure that this never happens. If this happened for real, show that you're responsible by accepting the failure, sincerely apologize! Otherwise, expect contributors to write books about it! 😉

Synchronicity is key

> 💡 Create a lighthouse, *not* a bug zapper!

While it's likely impossible to solve all problems associated with collaboration, the good news is that the most straightforward solution which can solve most of the issues mentioned so far already exists.

As developers, we can do this by means of documentation! Remember that writing documentation is an essential skill of every professional developer. So why not document our *values* as well? Consider this recursive statement which summarizes *values*, *assumptions*, *beliefs*, and *expectations* behind what this book is trying to achieve:

> *I **believe** that documenting our **values** for projects that we set up to be developed by*

> *community itself is as important as documenting software's API, because doing so creates realistic **expectations**. I **believe** that having this **belief** is beneficial, because it reflects my own **values** and **values** of other contributors, the kind of **values** that create prosperity. I **assume** that contributors who come together in a project want to achieve a common goal, and I **expect** that contributors want to find a common language in order to do so.*

To iterate, everything that you've read so far *assumes* that you manage or would like to manage an open-source *community-driven* project, or people who are simply curious about this topic, since it's not only about contributors, but users of the software that provide their valuable feedback.

If your project has the "community-driven" moniker assigned to it and you receive a push-back from the community of developers frequently, you should consider dropping that label from your project and make sure that you don't give out a false *expectation* of your project being "all things for all people". Otherwise, documenting project's *values* is desired. Show that you *value* honesty and preciseness, not just with words, but with actions as well.

Given importance of this, it's the *responsibility* of every healthy organization to declare and consistently follow their *values*. Therefore, it makes sense to dedicate some time to find our unexposed *beliefs* we have in mind for our projects. *Assuming* that you'd like to build and manage a successful *community-driven* project, all hidden *assumptions* must be revealed and actualized to the fullest extent, there should be no secrets.

At the same time, you are *free* not to do this, especially when *freedom* is seen as an inherent value behind open-source. This won't necessarily lead to a disaster. But the fact that contributors meet together from all around the world creates a potential gap of misunderstanding between people who might share different and even conflicting *values*, so we better not leave this to luck alone.

References

[1] What are VABEs? - By Professor of Leadership and Organizational Behavior.

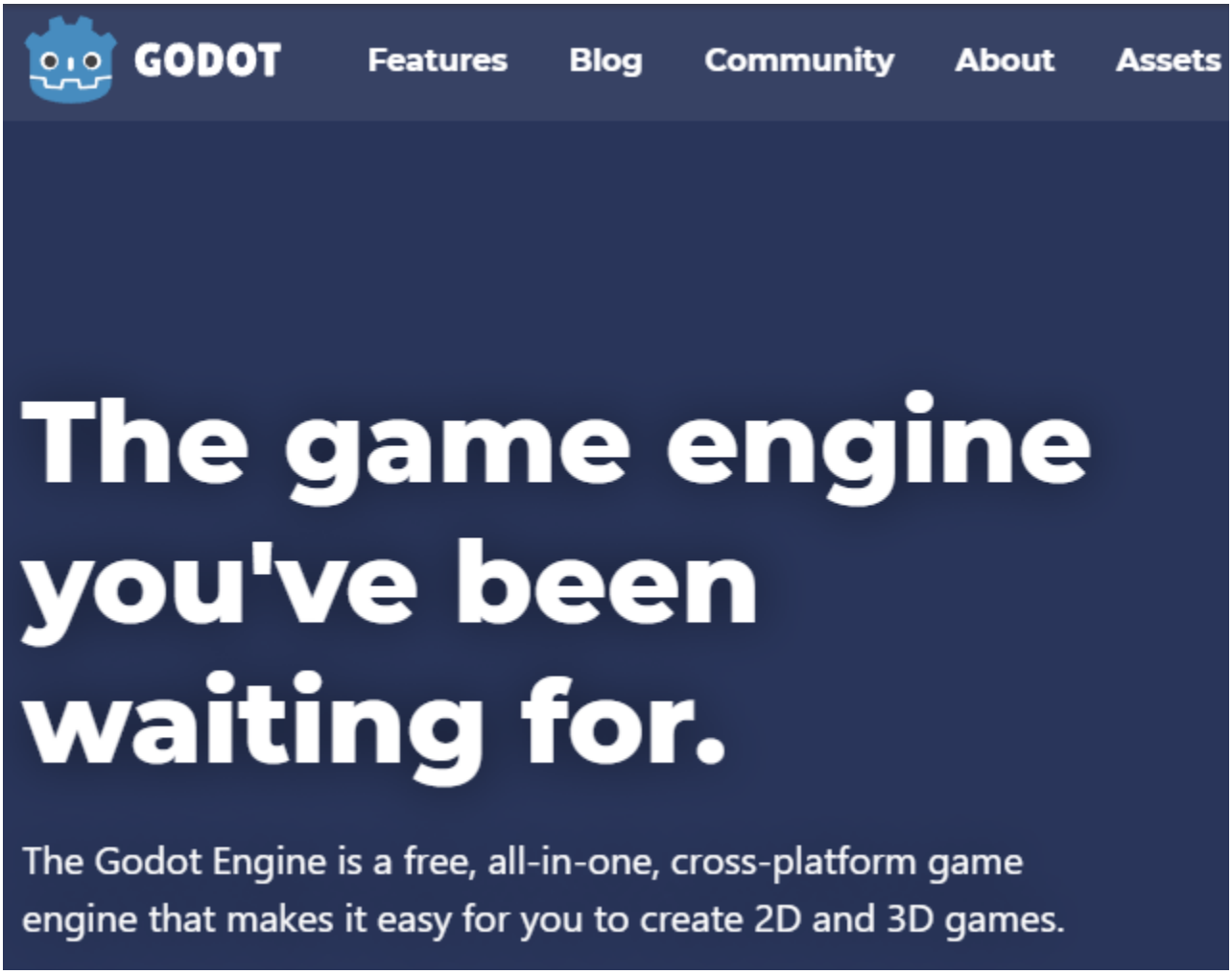
[2] The 10 Best Ways to Kill Ideas - By Gary C. Graziano, AIA and Christopher W. Miller, Ph.D.

Value of Waiting

"Vladimir: What do we do now?
Estragon: Wait.
Vladimir: Yes, but while waiting.
Estragon: What about hanging ourselves?
Vladimir: Hmm. It'd give us an erection.
Estragon: (highly excited). An erection!"

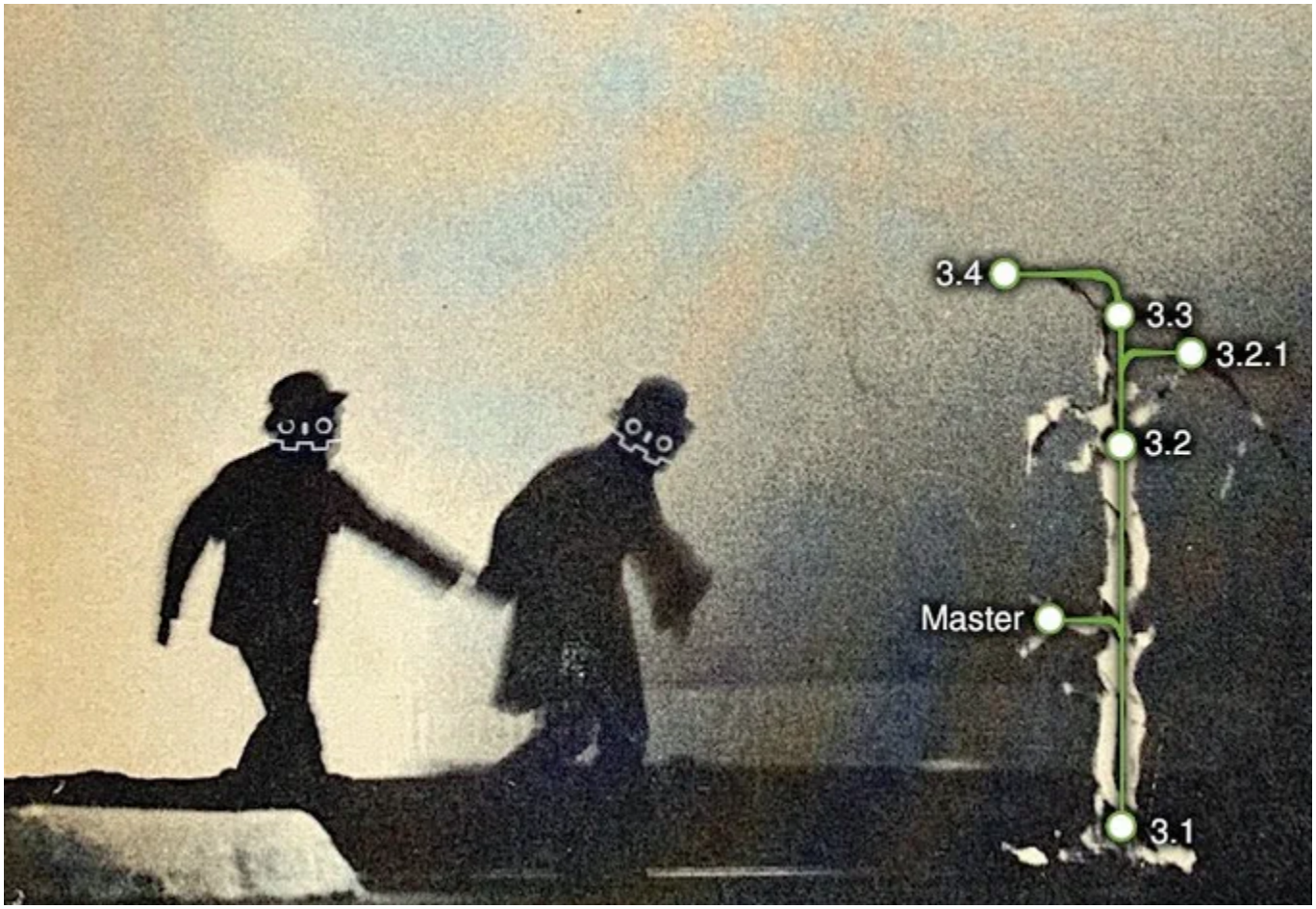
— Samuel Beckett, Waiting for Godot

Godot Engine is a prime example of a project that excels in the art of waiting. It manipulates the notion that those who exercise patience often reap the greatest rewards. Some people say that the name “Godot” shouldn’t be taken seriously, as it doesn’t necessarily represent the values behind the project. This might be true under other circumstances, but in the case of Godot Engine, the name “Godot” has a tangible meaning and suggests a peculiar sense of worth.



Godot's original slogan (2023): "The game engine you've been waiting for." Source: WebArchive

For those who are not familiar with the origin of Godot, the engine's name is derived from the tragicomedy "Waiting for Godot" written by Samuel Beckett:



Picture by Juan Linietsky. Source: X/Twitter

According to the lead developer of Godot Engine, Juan Linietsky, the name represents the never-ending wish of adding new features in the engine, which would get it closer to an exhaustive product, but never will[1]. Here's a quote from Juan about "Waiting for Godot":

> *In my mind, the most beautiful aspect about Beckett's "Waiting for Godot" is that for every cast and staging it's like an entirely different play, as both actors and direction each end up having their own unique interpretation of the original script.*

> In my mind, the most beautiful aspect about Beckett's "Waiting for Godot" is that for every cast and staging it's like an entirely different play, as both actors and direction each end up having their own unique interpretation of the original script. pic.twitter.com/EVnkhsMNFo
>
> — Juan Linietsky (@reduzio) November 26, 2021

Hopefully it should be clear by now that the choice of name is intentional. Taking Juan's analogy further, it becomes clear that people who use Godot and contribute to its development are *expected* to have a different interpretation of Godot's approach. The irony is that this *expectation* is obvious only to its creator and to those who have actually seen the play. Here's another quote from Juan about Godot's name[2]:

> *In a bizarre twist of fate, at some point and well into the development of the engine, we played a video game with a character that formed that connection (bonus points to anyone who figures it out), so the engine is also named in honor to that. This is good, because the original Godot was never supposed to arrive (well, that's one interpretation...).*

This supposed freedom of interpretation may seem appealing to the outsider at first. Who doesn't *value* freedom? Unfortunately, this "freedom" as offered by Godot proliferates *frivolity*. When we combine *waiting*, *ambiguity*, and the *novelty* of adding new features, it engenders a sense of *anticipation*. Anticipation is the *expectation* of a reward. This is particularly interesting because *uncertainty*, a prominent element of the waiting experience, has been shown to sensitize dopamine neurons, which can lead to various behavioral addictions, such as gambling[3].

There are also numerous reasons as to *why* we wait. One of the reasons outlined in the sociological research article "Waiting in Organizations"[4] can be readily applied to Godot, especially considering the context of the same analogy used in "Waiting for Godot".

> *Anticipatory waiting entails waiting for something that is hoped-for, such as a piece of good news, the delivery of some new equipment, or a colleague's return from annual leave. In such cases, waiting is likely to be experienced eagerly and as full of hope. However, Vladimir and Estragon's experiences show how such anticipation may fade over time and turn to cynicism and despair.*

Another interesting quote from the research article above is this. It backs up the idea that anticipation can lead to dependency:

> *Waiting can be conceptualised as an unevenly distributed exercise of power, since it is often the case that the powerless are made to wait for the powerful, the poor for the wealthy. To wait is to be made aware of one's dependence on another. However, waiting can also be joyous and full of anticipation, even when the waiting takes place in unpleasant physical conditions, such as is experienced by those waiting in crowds for the glimpse of a passing celebrity, or for the launch of the latest i-Phone. Thus, the subjective experience of waiting is likely to vary according to a range of individual and environmental conditions.*

There's a saying: *"Desires are nourished by delays."* The longer we *wait* for something, the more we *want* it. On the other hand, we can also draw a useful analogy from the teachings of Buddhism:

> *Desire and ignorance lie at the root of suffering.*

Dr. Jordan B. Peterson, a well-known psychologist, also employed a reference to "Waiting for Godot" as an analogy in the talk titled "The Importance Of Pursuing Your Goals," to describe the futility of waiting to achieving one's goals[5], *emphasis mine*:

> *[...] they wait around until it's **Waiting for Godot**, until they finally got it right but the problem is you're too stupid to know when you've got it right, so **waiting around isn't going to help**. Because even if the perfect opportunity manifested itself to you—in your incomplete form—the probability that you would recognize it as the perfect opportunity is zero [...]*

The leadership of Godot and its community frequently make enigmatic references to the play, alluding to the line "Mr. Godot told me to tell you he won't come this evening, but surely tomorrow" from "Waiting for Godot":

> Not today, but surely tomorrow. https://t.co/xMX0pf9NbZ pic.twitter.com/iRSSFjQIly
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) March 23, 2024

It's amusing to observe how they attempt to alleviate this adversity by using jokes as a coping mechanism. This is evidenced by the fact that they changed their original Godot slogan, "The game engine you've been waiting for," to the more neutral "Your free, open-source game engine." Perhaps by imagining that Godot has finally arrived, they try to reconcile the contradiction that way?

▸ Spoiler

Godot's unoriginal slogan (2024): "Your free, open-source game engine." Source: WebArchive

Conclusion

We have inferred the project's values from its name alone, and you can see how much information we have got so far. Imagine that you didn't really intend to convey such a message to the worldwide community! That's exactly why it's so important to reveal our *values*, *assumptions*, *beliefs*, and *expectations*.

This characterization provides a thorough examination of Godot's values from an objective perspective. From the perspective of a Godot user, these *values* may go unnoticed, or they may mistakenly *believe* that having such values do not have any adverse effects on Godot's development process and the community as a whole. Taking into account all above, Godot's apparent *values* go in complete opposition to what this books aims to resolve: *ambiguity* and *uncertainty*.

Waiting for Godot inevitably arises from the unfulfilled desires caused by the product's perpetual state of incompleteness, which anticipates the product that will never be finished (by the very definition expressed by Godot's lead developer), resulting in an unsatisfactory waiting experience. Waiters for Godot depend on the never-ending novelty of new features being added, or the promise of a "bright" future, which creates a *behavioral dependency* caused by the *fear of missing out* (FOMO).

References

[1] Introduction to Godot Engine - By Juan Linietsky, presentation of Godot at RMLL 2015 in Beauvais, France.

[2] Godot history in images! - By Juan Linietsky.

[3] How uncertainty sensitizes dopamine neurons and invigorates amphetamine-related behaviors - By Mike J. F. Robinson1 and Patrick Anselme, Neuropsychopharmacology. 2019 Jan; 44(2): 237–238.

[4] Waiting in organisations - By Catherine Bailey. Time & Society, Volume 28 Issue 2, May 2019.

[5] The Importance Of Pursuing Your Goals - By Dr Jordan B Peterson.

Development Philosophy

Terminology

Before we proceed to describe the components required to define the development philosophy, it is crucial to establish a shared understanding of the terminology.

While the term “development philosophy” includes the word “philosophy,” it still represents something tangible. If the choice of terminology seem meaningless for any reason, it can be helpful to perceive “philosophy” as a collection of fundamental principles. Defining “philosophy” does not imply a lack of pragmatism here; instead, it serves as a means to grasp the underlying principles that drive our decision-making processes and behaviors, rooted primarily in our intrinsic motivations. By embracing the term “philosophy,” we encourage a deeper exploration of the fundamental factors that shape our actions.

Considering the etymology of the word “philosophy,”[1] it encompasses a range of meanings, including the “love of knowledge,” the “pursuit of wisdom,” and “systematic investigation.” It even extends to concepts such as “alchemy, occult knowledge.” The latter suggests that valuable knowledge might have been deliberately concealed as sacred or deemed dangerous, not necessarily due to its inherent nature, but because it bestowed power upon those who possessed it. It is possible that the negative connotations associated with the learning process of new and unfamiliar subjects contribute to why some individuals find the term “philosophy” off-putting.

Taking all of these aspects into account, I invite you to embrace a genuine appreciation for the pursuit of knowledge and to approach new explorations with an open mind.

Meta principles

Most if not all projects adhere to the “Problem → Solution” principle. Due to this, some leaders think that defining *development philosophy* is an endeavour of dogmatism rather than pragmatism. However, if we apply this logic recursively, the lack of development philosophy can be classified as a problem which needs to be solved in and of itself.

Many healthy open-source projects define their development philosophy in various ways, shapes, and forms, not necessarily using words alone. It’s a matter of defining and revealing our *vision*, *mission*, *goals*, *non-goals*, *purpose*, *direction*, *intention*, *principles* etc, alongside more

fundamental information like *values*, *beliefs*, *assumptions*, and *expectations* with the community of developers, as we have covered in Organizational Culture section.

There's no true way to define development philosophy. The task here is to come to an agreement, a mutual arrangement accepted by both leadership and contributors, so that everyone has correct *assumptions* and *expectations* regarding a project. This way, leadership respects contributors' own values. This will greatly minimize all sorts of interpersonal conflicts between project maintainers and potential contributors. It definitely shouldn't be about attracting people from all around the world and taking advantage of the work they do for free. The lack of informed consent may backfire against a project, leading to a division of community caused by a conflict of interest.

Let's go through this in the context of the "Why → How → What" model[2]. Just as there are no perfect solutions, there are no *definitive* answers to these three fundamental questions. But the need for *good* answers is there, as is the need for good software.

Development philosophy doesn't have to be the Bible. It could be an explanation, a story about the vision of the project by its leaders. This document should not prevent anyone from contributing to the project, but should help to better understand which contributions are most likely to be accepted. Since we are naturally hard-wired to pursue goals, it is beneficial to establish a meta goal of defining a set of principles and acknowledging the necessity of having a plan to accomplishing project's goals.

Vision and mission

Why?

A* search algorithm for collaboration!

While development is mostly driven by problem-solving, these problems stem from our needs. Not all problems can be solved using our sole efforts alone. Due to this, developers must feel a shared *purpose*, and this is done by shedding light regarding project's *direction* and concrete *goals*.

If the purpose is not clear, a project may start accumulate proposals which go against the flow of development, making it more difficult to actually focus on things which can benefit a project in its current state. The answer to this question shouldn't be just a way to sell something. A project may not necessarily have a clear *vision* just yet, but what is our

intention?

If we address the lack of clarity regarding the project's motivations, it may be helpful to seek answers to the question "Why?" by utilizing root cause analysis techniques, such as the Five whys technique. This approach may seem unusual initially since these techniques are typically employed to identify software bugs, for example. However, as the saying goes, "It's not a bug, it's a feature!" 😁

We can also define *non-goals*. These signify things that may undermine project's roadmap, hence project's long-term success. Non-goals reflect the fact that they do not provide meaningful value to a project and may come with negative consequences or risks.

Non-goals are precisely the goals that do not pertain to a specific project but may be relevant to another project. This is the point at which alternatives can be considered. While providing alternatives may impose additional work, doing so prevents alienating someone and potentially creating a negative impression. Instead, it serves as a means of communication. In fact, certain organizations, such as consulting companies, focus solely on providing alternatives without any additional services. Even this book provides alternatives!

How?

💡 You can't have your cake and eat it.

This describes how we're going to achieve goals exactly.

What is the solution to the problem? The same problem can be solved in different ways and to different degrees. Why is one solution better than another? Perfect solutions are unattainable, but how are compromise solutions born?

This is where values help to shape the "How" as well. Values are not industry-specific. Values help to build a sustainable path towards a common goal. Without values, the "How" is chaotic.

It is also advisable to utilize estimations because, despite our best efforts, certain things are difficult to measure or even impossible to quantify. Employing estimations facilitates the involvement of all participants in making informed judgments and contributing to a project. However, it is equally crucial not to conflate estimations with promises: estimations are educated guesses that provide an approximate understanding of what to *expect*, while promises are explicit commitments that create a sense of responsibility.

Some might even argue that answering the question of "How" is what actually sets apart one

project from another. This is where intuition comes into act. There's no one true way to answer this question.

What?

💡 If you run after two hares, you will catch neither.

What constitutes a problem and what does not? Which problems does a project fully solve, which ones does it partially solve, and which ones does it not solve at all?

When and how does a user request transition into a problem that the project aims to solve? It's important to note that community support for a request does not always mean that the project leaders perceive it as a problem. Sometimes, suggesting alternative solutions that are not currently provided by the project may be sufficient.

Developers who come across a particular project come from diverse backgrounds. Consequently, they often request similar features to be incorporated into your project, even if they are not necessarily aligned with its *scope*.

The answer to this question is determined by the project's *mission*, which focuses on the present. It can be likened to a day-to-day job description. What tasks are you performing, and what is your primary focus to achieve the project's success?

Conclusion

It doesn't matter if your development philosophy is short or not, as long as it actually delivers your message to the public to the fullest extent possible. Use a particular language suitable to express these kind of things. Remember that not everything stems from pure logical reasoning, especially if you'd like to lead a project which is hedonistic in nature, such as a game engine.

If you don't have a plan, people may feel uncertainty and be paralyzed by indecision which stems from the lack of understanding of existing development philosophy as seen by project's leaders and seasoned contributors. There always exists the development philosophy that you follow, even if you think you don't. "I have no development philosophy" is also a development philosophy.

While it may be easy to build up enthusiasm by promising something, beware of situations

where you cannot fulfill your promises. You certainly don't want to end up with a situation where you'd have to reject 95% of ideas, proposals and requests, especially the bigger your project becomes. If this happens, this is a direct sign that you should clarify or start defining your project's vision and mission.

Note that not everything has to be defined right from the start, and development philosophy shouldn't be something that you must strictly adhere to, especially when our vision tends to change over time, since development decisions are mostly based on user needs. We just need to share the same vision to make the development process as smooth as possible for everyone involved, and try to adhere to declared principles.

Having a clear set of criteria regarding development philosophy can help people accept the way a project chooses what features are essential to realize its vision, which may or may not be in alignment with each individual's vision, and save the time it takes to implement them.

References

[1] Etymology of the word "philosophy" - Online Etymology Dictionary.

[2] Start with Why: How Great Leaders Inspire Everyone to Take Action – By Simon Sinek

Waiting for Philosophy

The following short satirical piece demonstrates Godot's development motivations in a humorous and concise manner:

> The development philosophy of @GodotEngine is fueled by a true love and passion to make engines great again! 🙃#WaitingForBlueRobot #Godoverse #IndieGameDev #IndieDev #GameDev https://t.co/4slWTyVGWD pic.twitter.com/MkQY4pAq9Q
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) March 24, 2024

The development of Godot Engine is governed by various implicit thinking processes which might not be instantly or completely obvious to a new person interested in contributing, especially if such a person has already some preconceived notions concerning the purpose and ideas behind Godot, so it's important that both new contributors and existing core developers share the same vision for Godot Engine development to achieve best results.

Due to the importance of having development philosophy for a project, there exists an *expectation* that a project like Godot would have something like this as well. And in some cases, you may even see some signs that there exists some kind of "philosophy" in Godot. Here's what you can read from the lead developer of Godot when he closes an issue discussing the usage of slow data structures in Godot[1]:

> *Again, I really appreciate that you have good intentions, but you don't understand any of the inner workings of the engine, or the **philosophy** [emphasis added] behind it. Most of your comments are towards things you don't really understand how they are used, how critical they are to performance, or what their priorities are in general.*

Juan's emotional reaction and interpretation aside which follows the above statement, what we can gather from his statement is that there exists some notion of *philosophy* behind Godot, which allegedly the author of that discussion doesn't understand. At the same time, when you ask Juan directly to document Godot's philosophy, he says that there's basically no development philosophy in Godot[2]:

> *I would say that Godot development extremely pragmatic, with focus solely on solving problems. Because of this, there is **not really a "Development philosophy"** [emphasis added]. There is zero dogmatism here and theoretical or philosophical discussions are generally irrelevant in this context.*

> *So, the Problem → Solution is what best expresses this mindset.*

Lead developer of Godot made two claims concerning Godot's philosophy. If you take them together, can you spot contradictions? Is it just hypocrisy or doublethink? Can you come up with an interpretation that could present Juan in a good light?

I asked the above questions to a lot of people. Most Godot followers either ignore, deny, or try to rationalize Juan's statements. However, when I ask these questions to people outside of Godot community, most people say it doesn't show Juan as an attentive, caring, or even a competent person.

We can go further in our journey to eliminate the ambiguity behind Godot's philosophy, and whether it exists in the first place. Here's another take of Juan around Godot's philosophy[3]:

> *I think the main reason why Godot, as an open source project, is growing so much (while so many other open engines failed) is that we have very user driven **philosophy** [emphasis added], where we discuss their needs and problems constantly, and never force our ideas and views*

Here, Juan describes this as a "user-driven philosophy" now. Apart from Juan's pretentious claims of Godot's growth over other allegedly failed projects, it would be quite far-fetched to describe user-driven approach as the core philosophy behind Godot, because being user-driven is a quality of a lot of projects, otherwise most community-led projects out there wouldn't be *useful* in the first place. It would be quite naive to believe that the user-driven approach is unique to Godot, so there must be something else behind Godot's so-called success.

Considering the project of this scale, this is quite bizarre. Despite Juan's position on this, many people would still like to know Godot's vision and mission in general. The "Problem → Solution" mindset is not unique either, because this approach describes pretty much every project out there. It still leaves room for ambiguity, as others point out.

Here's a list of projects that *do* have their philosophies/principles/goals documented to various extents:

- SCons Principles *(ironically, a build tool used by Godot!)*
- Unix Philosophy
- Blender Design Document
- The Zen of Python
- Lobster Design Philosophy, History, and Future

- Beef Programming Language - Design Goals
- Why Defold?
- Design Principles Behind Smalltalk
- Gigi: Rapid Prototyping Platform for Real-Time Rendering *(see Development Philosophy section)*
- Diátaxis - Theory *(an approach to technical documentation authoring, applied recursively to the project itself)*

What is Blue Robot's philosophy?

There exists Godot's design philosophy page, but it doesn't actually answer the bigger question of "Why". It doesn't contain principles that Godot could promise to adhere to. It doesn't target the engine developers and contributors themselves with the purpose of giving a good direction towards how the engine should be like, and has only introductory selling point value for those considering trying out the engine for the first time, not necessarily developing features. In short, it mostly answers the question of **what** rather than **why**. This is a clear example of Godot's substitution of concepts: they will use a 'philosophy' word, but in reality it's just a description of what the engine can do, or say that "Godot is "community-driven" while it's actually community-informed, as you'll find out in Waiting for Community chapter.

Due to Godot's ambiguity regarding its development philosophy, vision, mission etc, the following chapters attempt to describe Godot's principles, if we can call them like that. The following chapters are not meant to define Godot's philosophy or impose a particular worldview upon Godot, because according to Godot's lead developer, we now choose to *believe* that Godot has none. Even if Godot does have development philosophy which Juan doesn't want to reveal for various reasons, we have a problem of *accessibility* of such vital information. In either case, we're going to treat Juan's vision as a solely *subjective opinion*, despite the fact that his subjective opinion de-facto represents the current state of Godot.

Regardless, Godot's development process represents some collection of tangible enough qualities that allow us to form a pattern worth discussing. However, if you'd like to recieve a short answer, there's no such thing as a "development philosophy" in Godot, as all Godot contributors are guided towards the personal needs and desires of Juan Linietsky, which are often contradictory in nature.

References

[1] Using the slowest data structure almost every time - Comment by Juan Linietsky.

[2] “Problem → Solution” mindset - Comment by Juan Linietsky.

[3] User-driven philosophy - By Juan Linietsky, Twitter.

Overview

Here's a fairly general overview of the Godot Engine, at the highest level of abstraction. It covers Godot's design priorities, the development process, and last but not least, how public perception allows Godot to stay afloat despite numerous critical limitations, bugs, etc.

Some people who end up using Godot Engine say that it's a decent tool for prototyping, some might even argue that it's the perfect tool for this purpose. The engine has an easier learning curve compared to other alternatives that have a dedicated editor. But if you need to go more advanced, Godot doesn't offer a complete configurable solution out of the box that could allow to solve slightly less trivial use cases easier. Things like editor usability are always preferred over performance, and if there's a doubt to make something configurable or not, the most likely decision is the latter. A lot of features will be labeled as too specific by the core developers of Godot.

Due to the seemingly pragmatic nature of Godot's development, most features are overly simplified to achieve good-looking results that won't take hours or days for Godot users to implement. Intentional or not, it's a trade-off that the core Godot developers seem to be making. This makes users of Godot feel empowered. The problem here is that this development approach may not be clear to the existing Godot audience who haven't yet needed more than they're currently using in their projects, which may lead to disappointment at some point in the future. This problem is especially noticeable for users switching from other game engines, as they typically expect a similar level of experience and feature completeness. Unfortunately, Godot does not meticulously flesh out every detail the way professional software engineers do.

While Godot does have its merits, the core problem with Godot is hypocrisy. Godot's leadership doesn't present it for what it really is, or instead lets the community eccentrically interpret and define Godot's purpose. For example, most people understand that they won't be making AA/AAA games with Scratch—or any games at all. This is because platforms like Scratch are primarily designed to help develop algorithmic thinking, which is perfectly fine. The problem arises when beginners who might start their career with Godot think they're getting something on par with Unity or even Unreal Engine, which is not the case. Therefore, the misleading purpose of Godot leads to self-deception among its users and contributors.

The community is constantly misinformed that Godot is analogous and/or equivalent to existing solutions like Unity. Godot is released under a very permissive license, deliberately designed to be easy to use, and technically became one of the first open-source game engines with an editor that made it *look* like Unity. These factors alone made it possible to attract people with a predisposed bias toward Linux to promote Godot. The irony is that Godot and Unity cannot be compared on the same level. Godot's leadership is trying to

appeal to both the amateur and professional markets at the same time. As a result, Godot's followers begin to believe that Godot can efficiently solve all their problems, either now or in the future, which stems from Godot's leadership's propaganda and the community's desire for Godot to overthrow commercial engines, which is mostly an ideological rather than a pragmatic reason.

There are open-source game engines that are as popular as Godot, but unlike Godot, they are popular among professional developers specifically, because other open-source engines have more out-of-the-box customization and configuration options for specific use cases, they don't try to reinvent the wheel, and they use well-established data formats, even if those engines don't necessarily provide every feature out-of-the-box. Instead, professional game developers tend to use a set of dedicated tools that do their job well.

If we combine and generalize all the *technical* factors that contributed to Godot's alleged popularity, it's the set of factors that can be summarized under the umbrella of the all-things-for-all-people approach. But this comes at an *extreme* cost, which manifests itself as a severe lack of ability to customize the engine for advanced use cases out of the box. This makes the engine an unreliable choice for anything elaborate and/or long-term, especially since Godot's core feature set is never set in stone, and often undergoes significant compability breaks even during so-called "stable" releases.

As long as game projects made with Godot have any chance of becoming popular, Godot's features are constantly being changed to suit a variety of use cases that are unconditionally accepted by Godot's leadership, without much regard to whether new changes cause problems for the rest of the users, who are considered "mere mortals" in the eyes of Godot's leadership. Because of this, Godot developers often decide to simply remove features from the engine to resolve this kind of conflict, betraying the trust of existing users. Unfortunately, Godot devotees are unable to recognize this as an actual betrayal of trust, but choose to believe that they must suffer through these changes in the hope that Godot will eventually become stable. The irony is that the very nature of the name "Godot" suggests that this will never happen.

There is no philosophy behind the development of Godot. Rather, it consists of chaotic development decisions, which the Godot leadership refers to as an enigmatic "unwritten" consensus. Unfortunately, most people interested in contributing have to go through the tedious process of determining what kind of changes would make sense to include, which means that an abnormal number of feature proposals are either ignored or rejected by the core developers, unless of course someone manages to create a promising project using Godot. With each development iteration, Godot never converges to true stability, and is unlikely to ever reach it, because its development approach considers ways to maintain donations and attract sponsors to keep the investment bubble growing, so stability is never a priority for Godot. This approach of introducing never-ending novelty is the only viable way

for them to sustain themselves, given that they lack professionalism.

Uncertainty permeates Godot's development process. This kind of so-called vision is propagated by the manipulative rhetoric of Godot's lead developer, something along the lines of "accepting the reality that nothing can really be finished." Interestingly, it's possible to draw some analogies from *Extreme Programming* to describe Godot's approach, such as not programming features until they're actually needed. However, what is needed or not is still a subjective opinion of Godot's core developers. Needed for whom? The very ambiguity allows them to move the goalposts of the project's direction in a whimsical way.

Some Godot followers have expressed frustration with the perceived lack of innovation in commercial engines. However, this perception is misguided and has been deliberately perpetuated by Godot leadership. While Godot may appear to be revolutionary, its developers have been known to recycle existing solutions and attempt to convince users of their uniqueness and innovation. However, upon closer examination, these claims are often found to be false. They attempt to replicate the success of other commercial engines by cloning them, clinging to the success of others, and by ripping off existing solutions. They attempt to incorporate every possible feature that excel in other engines, and try to integrate those features into a single product. The dramatic consequence of this approach is that none of the features end up being as effective as the features that we see present in other tools on an individual basis.

The "pro-Godot/anti-Unity" disinformation campaign is that people who are into Godot or who are affected by Godot's propaganda start to ask questions like, "Why is Godot so popular?" Godot's fans are so enthusiastic that they spread the word about their engine. They *assume* that Godot is popular in the game development industry as a whole, even though it's not. They go along with this kind of misleading reasoning: "If Godot is popular, then it must be because of technical features that made it popular, and not hype." These loaded questions are designed to attract people into using Godot, especially those who have little to no experience in game development.

Godot is certainly popular among *some* subset of indie game developers, especially the younger people. But its community is very vocal, and it hooks developers with its deceptive simplicity. Godot also serves as a kind of refuge for those who share anti-corporate sentiments. The irony is that Godot is de facto a corporate structure that they despise. It's just that Godot's leadership is very good at hiding it. The reason behind the relative popularity of Godot could also be attributed to the concept of gambling. The endless promises of a "bright future" keep people hooked and eager to explore new features in Godot. The combination of waiting, uncertainty, and the prospect of new features generates a sense of anticipation. Anticipation creates the expectation of a reward. It's no coincidence that Godot is named after Samuel Beckett's "Waiting for Godot." As long as they can keep people from seeing the truth, it won't look like a grift.

This is not to say that Godot is useless, of course. But even if we consider Godot's current features, it could simply remain unnoticed, especially starting from the inconspicuous and insignificant Godot 1.0, the first public version of the engine. Some Godot followers even express mixed feelings: "Too terrible not to upgrade. Good enough to want to upgrade." Therefore, there exist other *non-technical* factors that made it popular among a certain subset of users. The subsequent chapters are going to reveal another important factor which contributed to Godot's alleged popularity, which closely correlates to the overzealousness of Godot community.

For a more detailed analysis of Godot's technical limitations, please refer to this excellent summary by a long-term user of the engine.

Priorities

This covers general priorities that Godot *seems* to follow. Priorities cannot be inferred from Godot's project roadmap because, in reality, Godot does not have one!

> "Quite soon." 😉
>
> Godot never arrived! 😜#TruthAboutGodot pic.twitter.com/IrpzTdLDk1
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) December 10, 2023

Godot's project manager, Rémi Verschelde, describes this as a so-called "unwritten consensus."[1] Any instances of the term "roadmap" used in the context of Godot usually pertain to backlogs, which are collections of issues or features compiled as to-do lists. However, these backlogs do not constitute a concrete roadmap.

A roadmap is not merely a list of upcoming features; it also relies on a clear definition of goals. Unfortunately, Juan Linietsky, the lead developer of Godot, does not recognize goals as a fundamental concept within the Godot project, as evident from Juan's diagram that you will discover in a subsequent chapter Companies vs FOSS. Juan says that at GDC 2019:

> *I mean, we don't have such a long-term vision, so, well, whatever...*

> Can you trust someone who has no long-term vision?@reduzio at GDC 2019:
>
> "There's no agenda or anything special about it. [...] I mean, we don't have such a long-term vision, so, well, whatever..."
>
> Regarding the agenda... Look at #W4Games.#GodotEngine #TruthAboutGodot pic.twitter.com/FbUEfikOZg
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) November 21, 2023

Many people would like to know Godot's vision, but in fact it has none or it's deliberately hidden from the public. Those who are not afraid to criticize Godot's management usually say something along these lines, here's just one quote from a potential contributor to Godot[2]:

> *And to be frank, I don't like the attitude of the leader when replying to feedback. I can disagree with someone and respect their views, but they seem very set out to to respond to so much feedback with **"you don't understand what you're doing, you don't understand my vision"** [emphasis mine]. The former is almost never accetable, but the latter is all on Godot. If you want people to understand your vision, don't have a 3 year outdated roadmap.*

Consequently, this lack of defined goals or a clear direction means that a proper roadmap cannot exist within the Godot project, as the very essence of a roadmap depends on the presence of well-defined goals.

Godot's leadership mostly hope that "big studios" are going to notice them and *they* would determine Godot's direction. Ironically, as Rémi pointed out, this creates chicken-or-egg problem: why would big studios use Godot if it's yet to be proven? Big studios still need to have some sense of direction before even *considering* adopting Godot!

Speaking of big players, if we take a review written by creator of RimWorld, he describes Godot's lack of decided focus as mis-spent effort[3]:

> *My main concern with Godot at this point is that it seems to be trying to be all things to all people. It's trying to appeal to the "my first game" student market, via visual scripting and GDScript and so on. But it's also trying to hit AAA features like advanced rendering and a built-in particle engine.*
>
> *This lack of decided focus manifests as IMO mis-spent effort on things that almost no serious indie should be using (advanced rendering features which look pretty in demo videos, visual scripting), while deprioritizing things that absolutely every indie should be using (C#).*

At the same time, we can still infer some of Godot's priorities. According to lead developer's response directed to the creator of Rim World, we can draw the inference that Godot aims for high-level functionality and implementing back-ends which allow to make games to **look pretty**, cover the **most common use cases** and only allow **some tweaking**. Even if he talks about rendering specifically, rendering is a crucial part of every game engine[4]:

> *Godot in contrast aims for mostly high-level only rendering backend that looks pretty, covers most common use cases and only allows some tweaking. The idea is that out of the box it looks as good as Unreal, is much easier to use, but of course may lack the ability to tweak performance for corner use cases (though it should still be good for most games). The vision*

> *here is that this can be achievable with a relatively simple renderer (there is not as much rendering code in there as you might think), and that if a large company ever wants to use it, they have the money to hire a render engineer and tweak it themselves.*

Full reply by Juan Linietsky can be found here:

> Comment
> byu/lmbarros from discussion
> ingodot

As we covered in Waiting for Philosophy chapter, Juan has been telling *users* and *contributors* of Godot that it doesn't have any kind of vision, and philosophical discussions are irrelevant. But when *big players* start to evaluate Godot, all of a sudden Juan provides Godot's claimed priorities. Don't you think this kind of behavior is hypocritical, to say the least?

The false dilemma of using Godot over other engines like Unity or Unreal becomes evident in various ways, sparking numerous discussions[5]:

> Anyone else not excited about Godot?
> byu/Atsurokih ingamedev

For example, Winter Pixel Games, the creator of Rocket Bot Royale, made with a heavily customized fork of Godot coupled with other custom modules and monstrous efforts that made the project possible, expresses an ambivalent approach when it comes to addressing Godot's limitations, in the larger context of discussing both the management and functional pros and cons of Godot[6]:

> *From scratch would be very difficult... Godot is absolutely amazing at a lot of things, and*

> *terrible at a lot of other things at the same time. A potential idea would be to rip things apart, scrap a lot of the bad parts, and rebuild internals to be less brittle.*

> From scratch would be very difficult... Godot is absolutely amazing at a lot of things, and terrible at a lot of other things at the same time. A potential idea would be to rip things apart, scrap a lot of the bad parts, and rebuild internals to be less brittle.
>
> — Winterpixel Games (@winterpixelco) October 6, 2022

Even if we temporarily assume Juan's claim above that Godot's rendering can be easily modified from the source code due to its "relatively simple renderer," the truth remains that Godot introduces unresolved technical debt for those who decide to adopt it for their projects. This technical debt becomes the responsibility of companies to address on their own, rather than being resolved by Godot from the start.

A notable example is the experience of Blind Squirrel Entertainment in developing Sega's Sonic Colors Ultimate, where they opted to use a customized fork or subset of the Godot Engine codebase. As it turned out, this custom implementation suffered from severe rendering bugs in *production*[7] [8]:

> if u ask a #Godot user "tell me a 3A game made with godot" they will say "oh that sonic game was made with godot!" here are some reviews from that game and yes most of them are engine related problems pic.twitter.com/L8W0WJaccp
>
> — Abruh (@abrasivetroop) January 19, 2024

Typically, resolving technical contradictions as mentioned above leads to innovation[9]. However, in the case of Godot, these unresolved contradictions do not stem from technical aspects but rather arise from inadequate project management and false expectations proliferated by both Godot's leadership and Godot believers[10]:

> *To be honest, if Godot embraces it becoming the **Noah's Ark for Unity developers** [emphasis added] and prioritizes C# over GDScript to improve performance, it could be good both for Godot and C# ecosystems, and more emphasis on the ability to use very rich selection of libraries written for .NET would not hurt either.*

The misconception and delusion that Godot is some kind of "savior" of the gaming industry is so severe that those who have truly grasped the reality of Godot's capabilities are being overruled by the mob mentality, especially on Reddit. Those who have used commercial game engines like Unity or Unreal *and* successfully shipped games with them have a much more objective perception of what Godot really represents at its core:

> I would say to Unity developers, don't use Godot.
> byu/xuhuijie520 ingamedev

> *And I'd like to briefly state my opinion that this Godot engine is only good for laughs, and whoever uses it for real game development will be miserable!*

Those who have already been seduced by Godot's false promises and/or hopes tend to either ignore or suppress doubts about its true capabilities. In other words, when some people describe Godot as amazing, it may be the result of wishful thinking rather than a reflection of reality, or the supposed simplicity of the engine may overshadow any other

negative aspects of Godot in the eyes of hobbyists, but not professionals. In fact, it may be a combination of these factors that reflects Godot's nature. A software engineer with experience in the AAA industry sums up this issue[11]:

> *These findings, combined with the general messaging of "we're an engine made for our users and contributors, not an engine made for our own goals" (paraphrased heavily), gives me the impression that **there is no plan** [emphasis mine]. Godot will have the features people decide to build for themselves, or the features that lots of people ask for, on some sort of timeline. And even if you make a feature, **there's no guarantee** [emphasis mine] it will get merged, even if it is highly upvoted. This doesn't have to be a bad thing, but it does mean you can't expect from Godot the sort of steady, planned, expected updates you can expect from a professional software company.*

Furthermore, there is no guarantee of *any* decisions being made in Godot, as you will learn in later chapters, like whether Godot is community-driven or not. Godot's absence of a plan is not attributed to its FOSS nature, but rather to preserve ambiguity in feature prioritization, driven by the hidden agenda of the lead developer. Because Juan Linietsky wields significant influence over contributors, they often find themselves adapting to Juan's shifting goals. While outsiders may occasionally observe this dynamic, contributors often remain oblivious to such influence.

References

[1] Rémi Verschelde's reply to Tynan Sylvester - By Rémi Verschelde.

[2] Devs learn rival Godot engine in a week to poke fun at Unity - Comment by johnnyanmac, YCombinator.com.

[3] Tynan Sylvester's comment on Godot - By Tynan Sylvester.

[4] Juan Linietsky's reply to Tynan Sylvester - By Juan Linietsky.

[5] Anyone else not excited about Godot? - Reddit.

[6] WinterPixelGames about Godot - Twitter.

[7] Sonic Colours: Ultimate players report graphics glitches and bugs - Eurogamer.

[8] Gamers Complain Sonic Colors Ultimate is Riddled with Bugs and Glitches - Nichegamer.

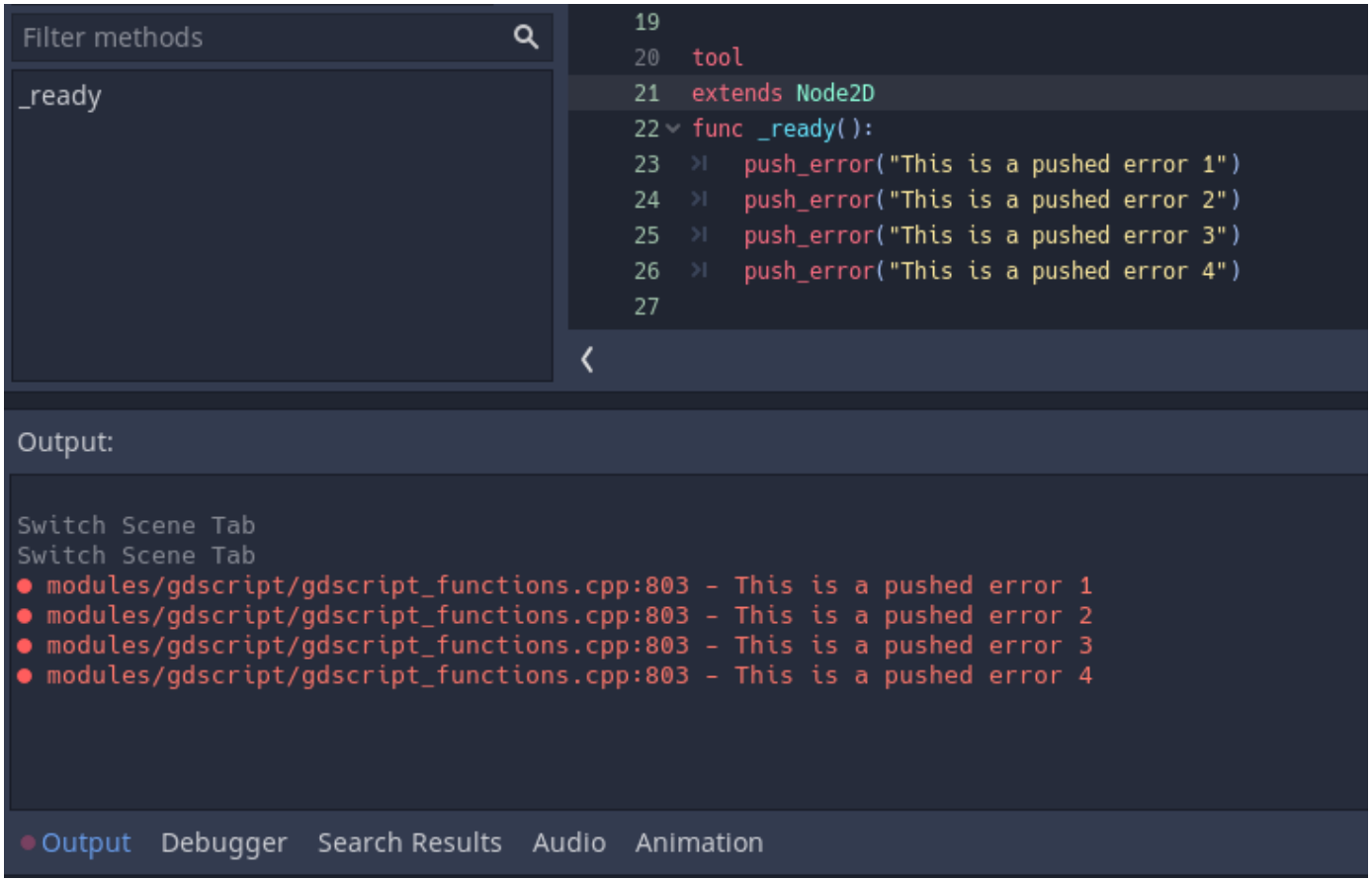
[9] Contradictions - TRIZ.

[10] The anatomy of a Godot API call - Comment by neonsunset, YCombinator.com.

[11] Godot: The Good, the Bad, and the Ugly - By Margaret Ó Dorchaidhe, September 19, 2023.

Simplicity

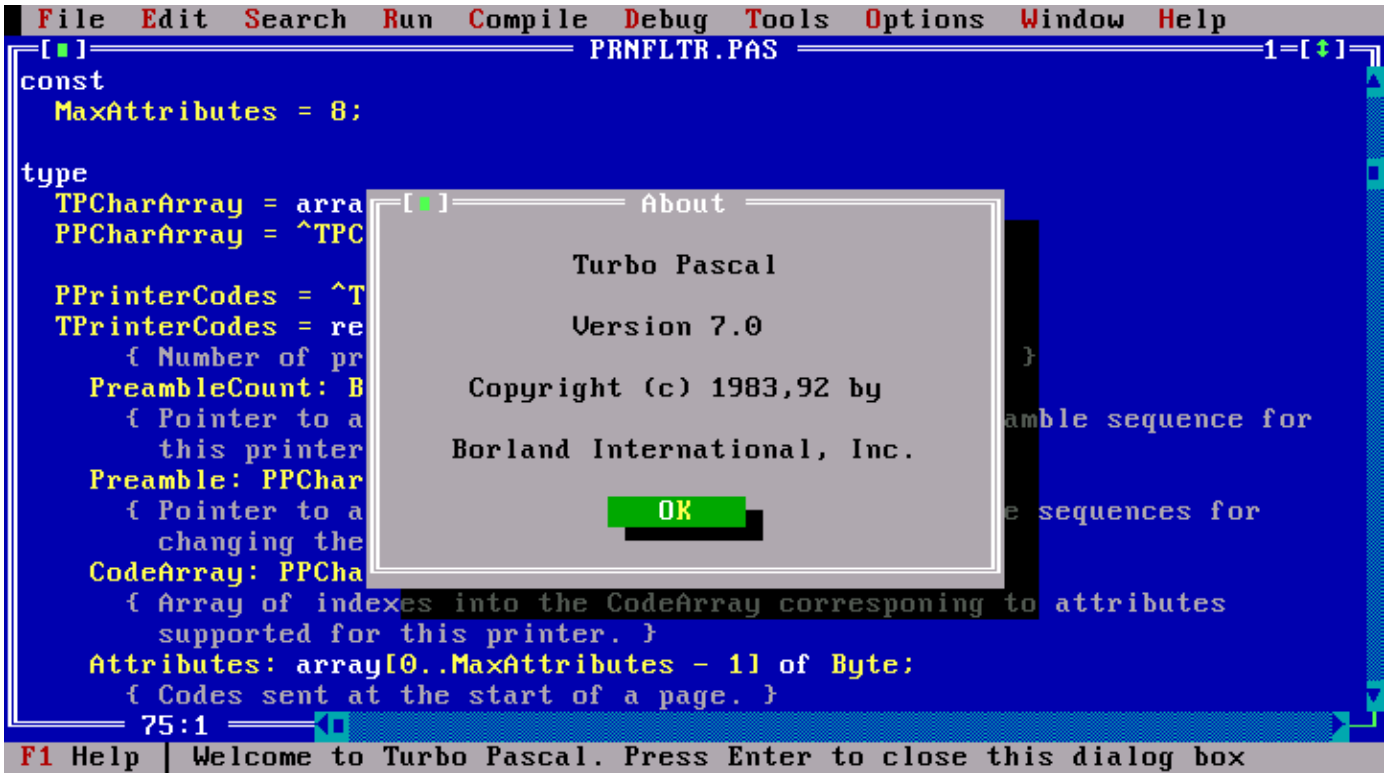
Godot Engine, known for its emphasis on simplicity, has established GDScript as its main scripting language. This decision by Godot's leadership has drawn attention and sparked discussions, including controversial ones.



Juan Linietsky, lead developer of Godot, has provided various "insights" into GDScript choice, although the underlying reasons may not be immediately apparent, even up to this day.

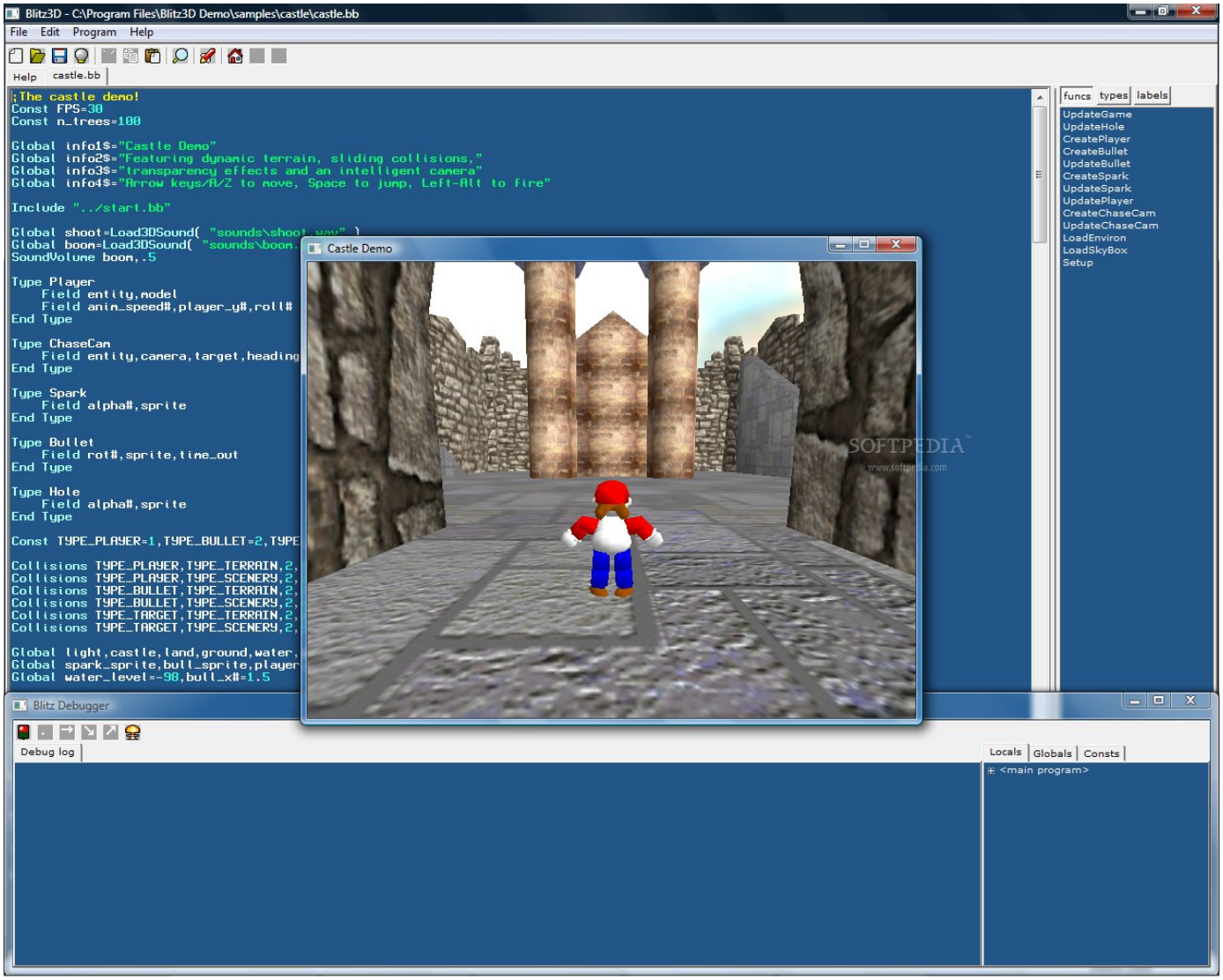
Motivations

Juan reminisces about his teenage years, fondly recalling the simplicity and user-friendly nature of environments and programming languages like Turbo Pascal. In fact, Juan credits Borland Pascal as a significant source of inspiration[1]:



BTW Borland Pascal (Image above) has been a great inspiration for me. The fact that this is an all-in-one environment that includes built-in documentation and reference made me understand how important it was to do the same with Godot.

Borland, in its quest to maintain dominance in the Windows development market, faced fierce competition from Microsoft's Visual Basic. In light of this historical fact, it is also relevant to mention Blitz3D, with its Blitz BASIC, as a potential source of inspiration for Juan, given its comprehensive all-in-one environment with a similar focus on simplicity.



Now, let me assure you, the fact that Turbo Pascal and Blitz3D share a blue color scheme, just like the iconic Blue Robot of Godot Engine, is purely coincidental! 🙃

Competition

Despite a subset of the Godot community expressing a desire to embrace production-ready languages like C# and some fervent Godot enthusiasts yearning to dethrone Unity, we must take into account Juan's perspective on the matter[2]:

> *Friendly reminder that Godot does not compete with Unity or Unreal. It's not a commercial product and it does not take part of that market.*

It should be noted that as the lead developer of Godot, Juan's words carry the weight of an

official project statement, even though he frequently draws comparisons to Unity and Unreal. Comparing Unity to Godot creates a false dilemma; users with extensive Unity experience clearly recognize the significant discrepancies between the two:

Preface

I'm comparing Godot mostly to Unity due to my ~10 years of Unity experience. I'm also primarily working in 3D and C# and that is what I want to use Godot for.
Terms:

- by "export" I mean a variable exposed in the inspector, that's how Godot calls them
- I use the word "prefab" to mean a scene that is intended to be instantiated multiple times, unlike a "level"
- Godot's Resources == Unity's ScriptableObjects
- Godot's collision shapes == Unity's colliders

Pros

- Open source
- Lightweight
- SDFGI
- Rendering pipeline is much more unified and easily configurable, while being light and powerful
- Custom shaders much easier to work with than with Unity
- Resources can be expanded inline in inspector (by default), unlike Unity's ScriptableObjects
- Resources can exist "embedded" in the scene
    - This is both a pro and con because it's unclear for beginners

Cons

- Editor scene view does not show what's happening during play mode
    - It has "remote" view but it only shows changes in the hierarchy.
    - Resource changes are never visible even in remote view
    - Changing parameters in editor (such as moving an object) will change parameters live in runtime, but this is a one-way action and will only be effective if the parameter is not touched by a script.
- 3D (re)import workflow:
    - Requires you to save each mesh of a model file as a separate resource to be able to link to meshes inside model files, so that reimports would update the meshes in game.
    - Mesh colliders don't update when you reimport the model. The mesh collider shape actually has no knowledge of the original mesh, it keeps a copy of the data. This requires the user to go through each scene using the modified mesh and recreate

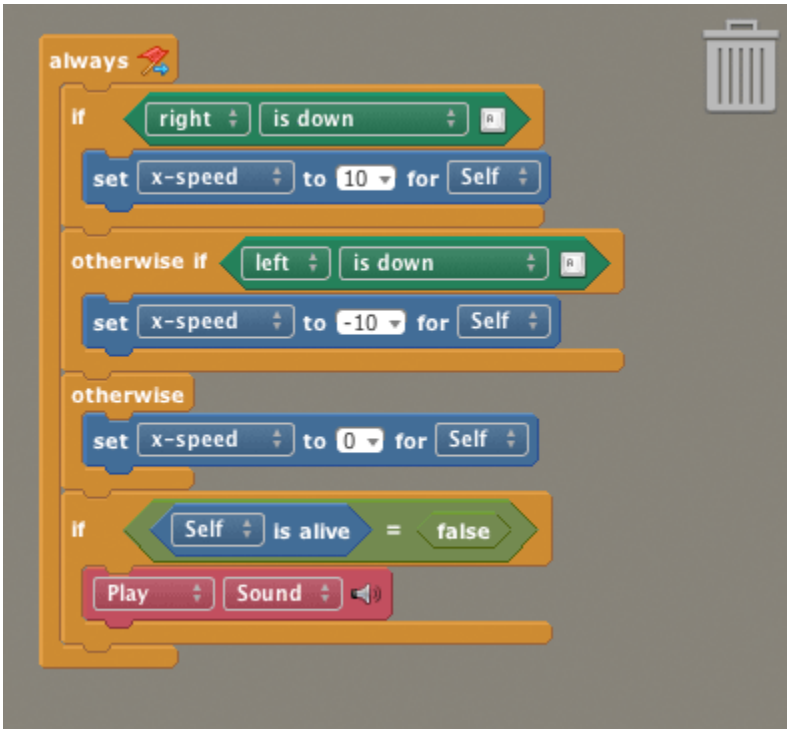
colliders.

- Can't export (show in inspector) custom classes (unless they're a resource)
- Can't "apply prefabs" (can't save a parameter back to the scene once it is "editable childrened")
  - This is useful when a prefab depends on the context, example: a flashlight casts a light, but on its own, isolated in scene, it's impossible to see how the light being cast looks like. So when you tweak light values in an external scene where you can see the projected light, you have to manually copy light parameters to the prefab to save it to all flashlight instances.
- C# is a second class citizen, and a lot things are either not available in C# at all, or terrible for performance (like raycasts)
  - C# doesn't recompile until the game is run again. This makes exports unavailable until playing. (This behavior actually breaks plugin scenes, if you open a scene with uncompiled scripts, references will break)
  - Profiler doesn't show C# scripts
  - Errors cannot highlight which object has thrown the error in scene
- Slow "play" in editor start-up times, caused by:
  - running a new instance of the game in a separate window, hence reloading all assets
  - C# compiling (which doesn't happen in the background as mentioned earlier)
  - domain reloading (which in Unity can be turned off for near-instant play mode start-up)
- Profiler is very bare bones
  - doesn't even have a (zoomable) timeline where you can see events within a frame, only the whole frame timings are shown
- Making a scene from a branch disconnects all exports assigned on the root that are local to that scene
- Dropping a scene into another scene overrides all exports in the root. You have to manually "revert" overrides for each parameter.
  - Since node links can easily "break", this makes fetching nodes by name encouraged, which to me is an antipattern
- Inconsistent scene nesting behavior: scene instances do not expose their internals by default which prevents linking their children to external scripts. HOWEVER, they do still exist and can be linked to if "editable children" is turned on, linked and then "editable children" turned off.
- Unpredictable resource copying
  - Example: if you drop a model file into the scene, it will keep a reference to the model, if you copy it, it will copy all the underlying data without warning, losing a link to the original model file

- Duplicating multiple objects that reference each other does not relink references to duplicates but they keep links to original objects
- No simple distance constraint (joint)
- Joint anchors cannot be changed after they're created
- Only collision shapes (colliders) that are an immeditae child of a body are added to it (i.e. deeply nested shapes will not be added).
    - However, there is a way to do that with a script
- A lot of basic functions require verbose code, which is why I'm compiling these utils
- Cannot export C# collections (like Lists)
- Cannot put inherited classes into an exported array

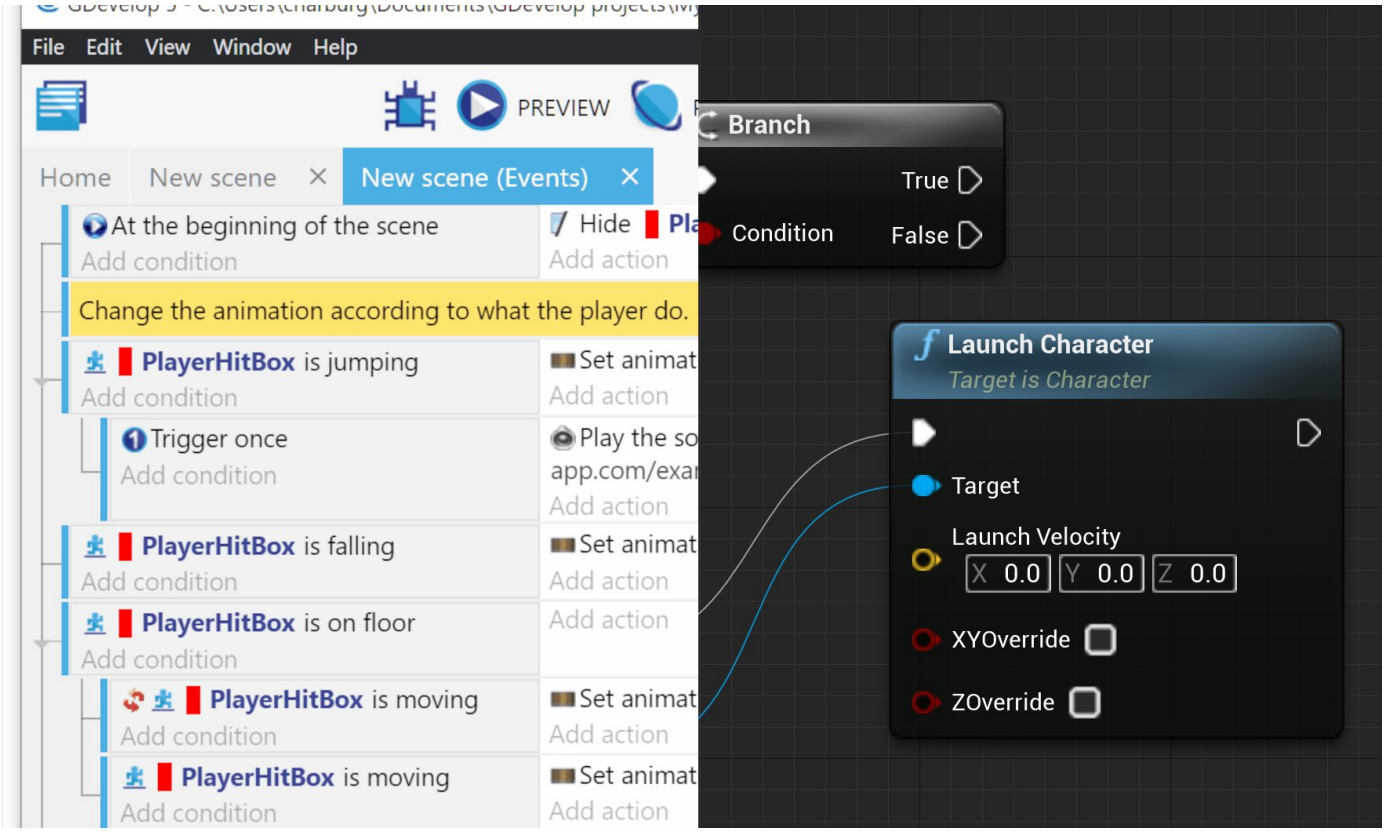
godot-pros-and-cons.md hosted with ❤ by **GitHub** **view raw**

So, with which game engine does Godot truly aim to compete? By examining the realm of visual scripting languages, we can uncover valuable insights. Juan has grappled with various dilemmas regarding the incorporation of event sheets or visual programming paradigms in Godot[3], [emphasis mine]:



> *... everyday I question myself whether something more akin to how **Construct/Stencyl** works would have been better, or whether it makes any sense at all to have VS because compared to other tools, Godot is just not as high level..*

Even though Godot decided to remove VisualScript from Godot 4.0, there is still a possibility that Godot may favor an alternative approach, which somewhat narrows down the competition field[4]:



> *Judging from all the feedback from the recent days regarding to the Visual Script removal in Godot, I think one crucial aspect we failed to understand originally is that "Visual Script" means two entirely different things to two groups of users (let me know what you think)..*

Juan suggests that the first group encompasses approaches similar to those employed by Unreal:

> *The first one is the one that used Unreal blueprints. This is the majority group, so we aimed for this one. The problem is that, for most users of it, they used it because they had no choice (other than C++) and GDScript ended up being a better alternative in Godot...*

Let's not forget that VisualScript was not abandoned solely due to a lack of maintenance; rather, its implementation was deemed unintuitive and subpar compared to other solutions.

Juan goes on to describe the second group, which revolves around the concept of event sheets—a paradigm he believes aligns better with Godot, [emphasis mine]:

> *The second group is the one that uses tools such as **Construct**, **GDevelop**, **Game Maker**, **RPGMaker**, etc. which are event sheet based and expose high level pre-made behaviors. This is a much smaller group and hence it ended up invisibilized in our original evaluation..*
>
> *For potential users who don't know programming, **this approach makes more sense to have in Godot**, the problem is that Godot is very game-neutral, so we never felt inclined to add something like this. The situation may be different, however, now that we have GDExtension..*

From the perspective of simplicity, the engines mentioned by Juan above could be seen as competitors to Godot in one way or another. However, to underscore the perceived significance of GDScript for Godot, here are several quotes from Juan himself[5]:

> *GDScript is designed to be simple, so it can be learned over a weekend if you have experience with other languages. It also (according to the polls we did) served as the starting point in programming to many Godot users, given its simplicity.*

> *GDScript is a great example of a tool that adapts to the job (rather than job adapting to the tool). The syntax is simple and straightforward, and the deep integration to Godot makes for a pleasant "glue free" experience that makes it difficult to go back to previous workflows...*

> *The whole idea of GDScript being extremely easy to use and hack (not doing everything for you, but giving you a large amount of tiny building blocks) and having a synchronous data model between code and scenes, all was conceived for this sole purpose.*

Hence, for those hopeful of Godot discarding GDScript in favor of a language like C#, I must disappoint you. GDScript will continue to serve as the primary scripting language for Godot.

While there are other notable aspects, such as the touted node-based architecture, the discussions outlined above dominate much of the narrative surrounding the simplicity of Godot Engine. However, let us now delve into some less apparent connections.

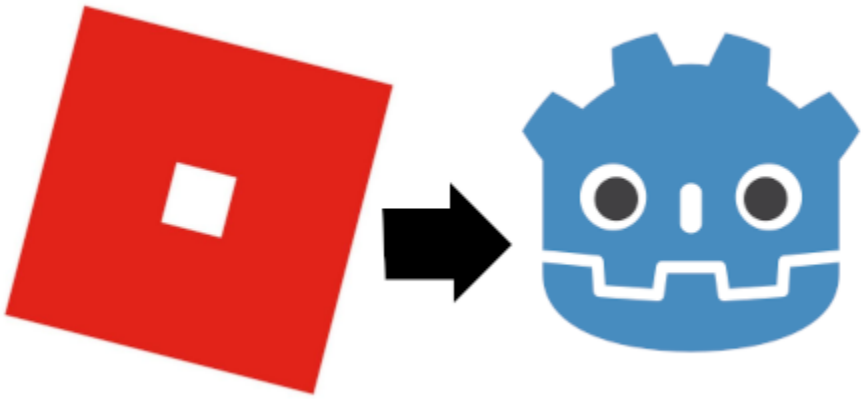
A relevant point of comparison lies in **Roblox**, a widely popular platform that utilizes Lua as its scripting language. Interestingly, Godot itself had previously utilized Lua:

> The first scripting language was Squirrel, not GDScript (previously we used Lua for past

games). pic.twitter.com/2LZsg4l7nW

— Juan Linietsky (@reduzio) May 25, 2019

Curiously enough, a game engine called **The Mirror** has emerged, akin to **Roblox**, built upon the foundation of Godot Engine[6]. The intriguing aspect of this connection lies in the fact that several contributors and maintainers of Godot, including Ariel, Godot's co-founder, are also involved with The Mirror. What makes this connection fascinating is that Juan has previously spoken about Godot in relation to Roblox, in an invigorating manner[7], [emphasis mine]:

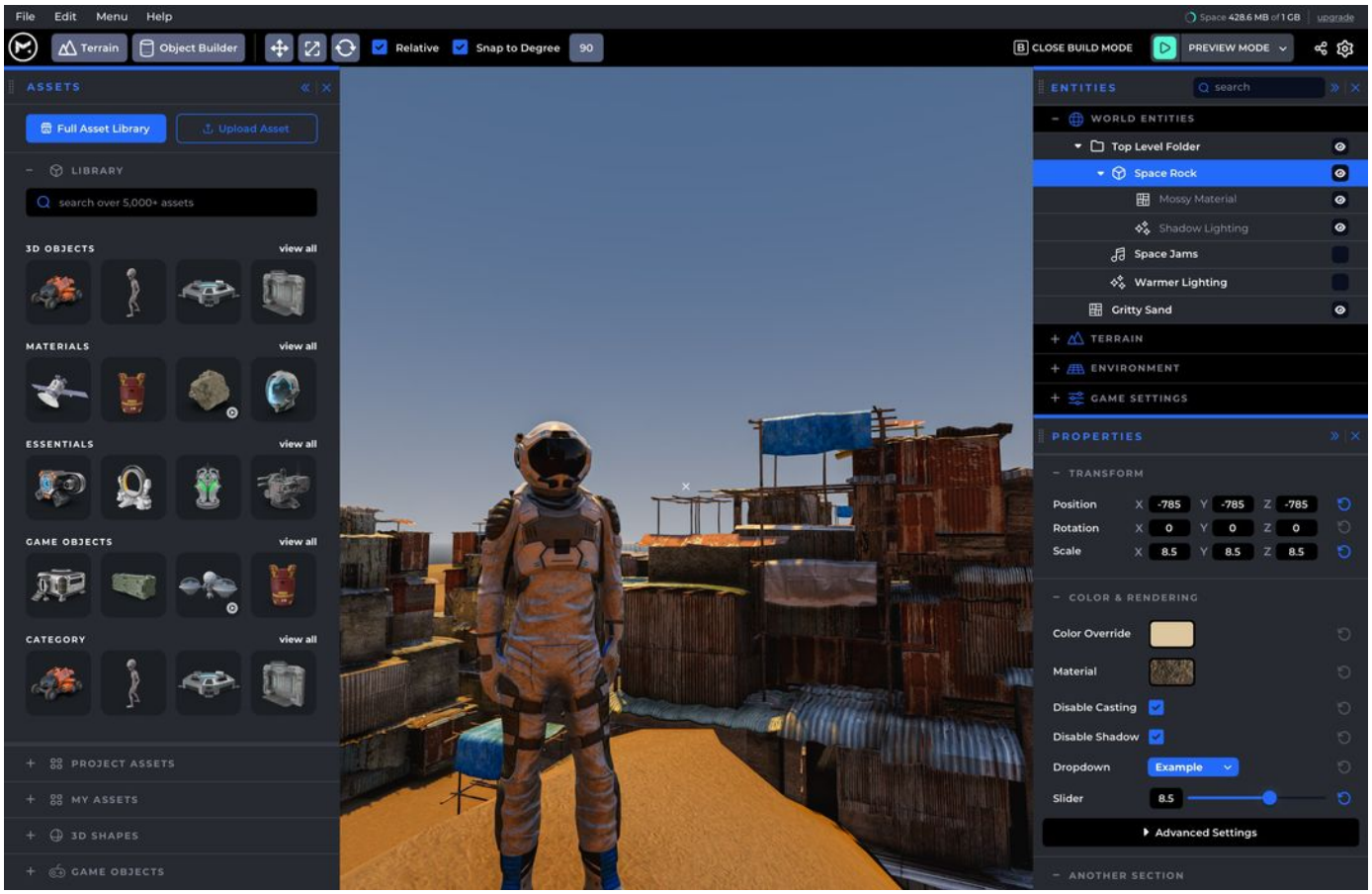


> *I often get asked what I think about "easier to use" platforms like **Roblox** (and endless other metaversey or easy gamedev tools) mean for the future of Godot and more traditional game engines, and if they are meant to replace them..*
>
> *Actually, I think the trend is entirely the opposite!*
>
> *Thanks to these platforms, you can see **way more people interested in game development**, which eventually hits a wall and looks for more powerful and **flexible technology** to eventually make the jump and continue progressing!*
>
> *So, to sum up, I think these technologies are **the best that could ever have happened to the game industry**.*

This commitment to simplicity resonates strongly with the student market and has contributed to Godot's popularity in game jams, as evidenced by Juan's excitement[8]:

> *GMTK Game Jam weekend, Let's wait for official numbers, but I can't seriously believe this. Completely out of this world!*

If The Mirror achieves the level of success enjoyed by Roblox, a multi-billion-dollar enterprise, it is undeniable that GDScript will continue to be a vital component of Godot's future.



> 💡 We are going to cover the potential conflict of interest between Godot as a non-profit organization and commercial companies in the subsequent Companies vs FOSS section. It is recommended to continue reading the chapters in order, though.

Consequently, and considering the so-called fully *user-driven* approach constantly regurgitated by Juan Linietsky (see previous Waiting for Philosophy), the development trajectory of Godot may naturally align with the requirements of this Roblox-like platform, prioritizing features that cater to its specific needs rather than the broader community's. While this alignment may result in a shift in Godot's direction and purpose, potentially unsettling users and volunteers who previously perceived it as a direct equivalent to Unity, it is a situation that demands careful examination.

As the story unfolds, one thing is certain: Godot's commitment to simplicity will persist, keeping GDScript at the core of its de facto approaches and deeply-ingrained motivations.

References

[1] Juan Linietsky about BorlandPascal - Twitter.

[2] Juan Linietsky reminding that Godot doesn't compete with Unity or Unreal - Twitter.

[3] Juan Linietsky on Event Sheets and Visual Scripting - Twitter.

[4] Juan Linietsky about two groups of "Visual Script" - Twitter.

[5] Juan Linietsky emphasizes importance of GDScript: 1, 2, 3 - Twitter.

[6] The Mirror - Godot Powered Commercial Game Engine - GameFromScratch, YouTube.

[7] Juan Linietsky sees Roblox-like platforms as a way to get more people to use Godot - Twitter.

[8] Juan Linietsky's excitement about the increased usage of Godot for game jams - Twitter.

Do Repeat Yourself

Quoting lead developer of Godot[1]:

> *Since everyone is talking about OOP, here is my take: I think different persons have different priorities or philosophies when designing software. For me, as a tech stack developer and as a user, clean and easy to understand API interfaces were always my #1 priority. Then, what the code does and how clean is organized internally, how much it reuses, etc. I don't really care as much. I like to provide and to use technology that just works and does not get in the way. So, for me, OOP has always been a fantastic way to achieve this..*

As you may have noticed, Juan mentioned concepts like "principles" and "philosophy" once again. He seems to have learned how to type these words but apparently hasn't grasped their meaning! 🙃

I believe you'll agree that a clean API should always be a top priority for all software, without exception. Failing to design a clean API would signify a failure in proper software design. Frankly, it shouldn't be seen as merely a priority, but rather as a necessity!

Regarding "*what the code does and how cleanly it is organized internally, how much it reuses, etc.,*" Juan doesn't seem to find any usefulness in these aspects. He explicitly states that he "*doesn't really care as much.*" Even though Juan didn't mention Godot specifically when discussing this (although most of his tweets are about Godot anyway), he made these remarks while telling the community that Godot is easy to modify from the source code, allowing users to tweak it themselves, which often requires tinkering with the implementation. But can it truly be easy to do so if Godot doesn't even follow the DRY principle to begin with?

The DRY principle emphasizes avoiding redundant code by promoting code reuse and abstraction. OOP, as a programming paradigm, provides several mechanisms that support code organization, encapsulation, and reuse, which align with the goals of the DRY principle. The most amusing part is that Juan sees OOP as a way to uphold his careless style, which often involves code duplication and a severe lack of code reuse. This contradicts the very essence of OOP, which can facilitate the use of the DRY principle.

The existence of these contradictory elements raises numerous questions, to say the least. Read on!

References

[1] Juan Linietsky on OOP and clean API - Twitter.

Performance

You might expect a game engine to prioritize performance above all else, but that's not exactly the case with Godot. The performance issues of Godot are like a big elephant in the room, so let's lighten the mood with a short satirical piece before we dive into the details:

> Juan @reduzio Linietsky: "So we're going to be kind of on the same level as the big engines..."#WaitingForBlueRobot #Godoverse #IndieGameDev #IndieDev #GameDev https://t.co/4slWTyVGWD pic.twitter.com/tT3z5PLShr
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) March 24, 2024

Given that games are soft real-time systems, performance is usually of paramount importance for game engines. It's not clear whether Godot would like to hit AAA games in the future, but existing decisions suggest that Godot is mostly suitable for smaller games that are not performance-demanding.

If we look at older but still juicy proposals such as "Using the slowest data structure almost every time" at Godot[1], it becomes clear at first glance that Godot favors ease of use and maintenance over absolute performance. Godot often chooses the easiest and most maintainable data structures, even if they are not the fastest.

> Recently I've come to discover this old godot issue https://t.co/99gKDwGqui (as I understand this issue has somewhat become a sore spot in the #godotengine community).
> (8/25)
>
> — Winterpixel Games (@winterpixelco) July 7, 2021

Under normal conditions, this could indicate that Godot values user-friendliness and code simplicity over performance in most cases. This kind of approach may be reinforced further by lead developer's article called "Why isn't Godot an ECS-based game engine?" by doing what he describes as "clever optimizations" as a compromise[2].

Unfortunately, Godot's optimization deficiency is painted as a preference for code readability. This is often seen as a sign of negligence by professional game developers. For example, lead developer of Godot doesn't believe in "death by thousands cuts"[3]:

> *I never really believed in the "death by thousands cuts" premise regarding to optimization of*

> *code. This premise exists as a counter to "premature optimization is the root of all evil" and it basically says that if you don't optimize early, then it will eventually be too late..*
>
> *The rationale is that, if you don't optimize early, then all the code accumulates will add "slowness" to the project and at some point the whole codebase will need large rewrites to optimize. To avoid this, everything should be optimized early. I think this is a big mistake.. And that it leads to extra complexity, or software that is much harder to develop and use.*

> In the book "Data-oriented design", there is a chapter called "Death by a thousand cuts", maybe the arguments there could convince you.
>
> — Ivica Bogosavljevic (@i_bogosavljevic) January 14, 2023

This attracted a fair amount of criticism towards Juan's claims concerning optimizations, which frankly apply to the Godot codebase as well[4]:

> *"Optimization" is always ill-defined in these discussions. Optimization is a spectrum that includes a myriad of elements, techniques, concerns. Data structures. Memory layouts. Mem allocations. Lazy-evaluation. Over-Eager evaluation. SIMD. Multithreading. GPU version. And so on.*
>
> *Nobody says you should write everything in SIMD / multithreaded / GPU form right from the start. That is definitely premature. But writing it "properly" (without pointless memory allocations, or O(n^2) loops that could be easily avoided, etc)... is not.*
>
> *In other words, do the basic optimization bits early, yes. Not everything. Remember: it's a spectrum. Putting yourself at the absolute beginning of it makes no sense, at any time, because it takes so little effort to move a bit in the right direction. Some of these things should not even count as "optimization".*
>
> *For example removing a pointless allocation, or skipping code that doesn't need to run, is just good practice and basic cleanup. Not really optimizing the code.*
>
> *Do that early.*

Juan often makes discussions quite ambiguous, as evident from the above criticism concerning Juan's claims. What we can possibly infer is that Godot might not even care much about performance in the first place, or until slowdown is so noticable that it creates enough pain for Juan to start optimizing code.

> I never did DOD, but I know for sure that good data layout is tremendously important for performance, and you cannot achieve easily that with traditional OOP.
>
> Still, I find that writing software in general is hard, seven without any performance considerations.
>
> — Ivica Bogosavljevic (@i_bogosavljevic) January 14, 2023

This kind of approach as expressed by Juan sounds more like *excuse-making*. These justifications as being offered by Juan are not genuinely intended to clarify the approach, but rather to avoid responsibility or deflect criticism that Godot's codebase receives from industry experts, such as analysis of Godot's codebase riddled with bugs, as you'll find out in subsequent "Quantity vs Quality" section.

In contradiction to Godot's alleged claims regarding performance, like Godot not accepting ECS principles or architecture, Juan even wrote an entire article accepting a possibility of Godot becoming viable for AAA game development, titled "Godot for AA/AAA game development - What's missing?"[5]. Taking into account what was presented here so far, in most likelihood, articles with click-baits like these are written to attract corporate sponsors. To answer the actual question put forward by the article... Godot is missing! 😁

> What's missing from Godot for AA/AAA development?
> Honestly? A capable lead dev.
>
> This is so far away from Godot's reality and capability that at its very best, it is delusional trappings coming from somone who hasn't made a game with Godot near a decade.https://t.co/c0ioZOwfKb
>
> — LillyByte (@LillyByteGames) January 17, 2023

Another notable source comes from a Unity user who was considering a switch to Godot. In an article titled "Godot is not the new Unity - The Anatomy of a Godot API Call"[6], within the section "So why are we waiting for Godot?", the author, after conducting thorough comparisons and benchmarks, summarizes Godot's performance with the following concise statement:

> ***Godot has made a philosophical decision to be slow.*** *The only practical way to interact with the engine is via this binding layer, and its core design prevents it from ever being fast. No amount of optimising the implementation of Dictionary or speeding up the physics engine is going to get around the fact we're passing large heap allocated values around*

> *when we should be dealing with tiny structs. While C# and GDScript APIs remain synchronised, this will always hold the engine back.*

Read a summary of the conversation between Sam Pruden and Juan Linietsky, as well as notable reactions from industry experts, including Mojang's business developer:

> Waiting for Godot 😈 https://t.co/pGDzzIPIFr
>
> — Kaplan (@Kappische) September 24, 2023

Juan Linietsky's assurances that Godot will become a more powerful game engine are meant to instill a belief in a so-called "bright future" rather than something to rely on in the present moment. People are misled to believe that they must wait for Godot to become a performant engine, when the underlying architecture suggests that Godot will never become one, so waiting is by definition futile. Most people will just choose a different engine:

> People evaluating Godot will not point out issues and then wait for them to (maybe) get fixed much later - they'll choose a different engine.
>
> People discovering performance issues in Godot mid-development won't have time to wait either, they'll have to downscope instead.
>
> — Rune Skovbo Johansen (@runevision) September 24, 2023

Since we are only discussing development philosophy at the highest level of abstraction here, it's recommended that you read the entire article yourself to get a full understanding, as it is quite technically dense.

You can also discover more about Godot's performance issues by looking at post-mortems that involve intersecting technologies like Rust:

> Here's also our postmortem of 3 years of Rust gamedev, and why we're leaving Rusthttps://t.co/oA8u7ehLuf#rustgamedev #rustlang
>
> — LogLog Games (@LogLogGames) April 26, 2024

> *We've been using Rust on basically all of our games since mid 2021. This was when BITGUN initially started as a Godot/GDScript only project, and as* ***we ran into issues with Godot's pathfinding (both performance and functionality wise)*** *I looked into alternatives, found*

> *gdnative, and then was recommended godot-rust. It wasn't the first time I've seen or used Rust, but it was the first serious usage for game development, being preceeded only by game-jam-y projects.*

Sadly, there are people who persist in believing Juan Linietsky's repeated Godot-is-almost-here assurances, viewing him as an unquestionable authority. This perception is so strong that even those who initially criticized Godot's performance might retract/delete their statements years later, despite ongoing evidence of performance issues in the present.

> Have you seen https://t.co/5QTELZFU1g
> and https://t.co/bVXOJb3MSd [which was posted five years ago, but still very much valid]
>
> It's nice to have outside eyes from actual professionals coming in fresh and going, "Eh?" 😅
>
> — LillyByte (@LillyByteGames) September 19, 2023

This isn't necessarily because they've changed their minds. This social phenomenon is mainly due to the overzealousness of the Godot community in upholding the engine's flawless public image, where they often attack critics deemed to have "misunderstandings" about the engine, leading these critics to doubt their own perception of reality. One notable instance occurred when rendering priorities were criticized, and the rendering lead, Clay John, dismissed those concerns as mere misunderstandings:

> i swear to god every slight criticism to godot gets an answer that starts with "you have a few misunderstandings"
>
> — Abruh (@abrasivetroop) January 18, 2024

This pattern of behavior has become so obvious that people have begun to mock it ironically:

> Perhaps we misunderstood the profound depths hidden within 'misunderstanding.' In the intricate tapestry of language, we found ourselves lost in translation, as the very word meant to illuminate the fog of confusion became a riddle in itself. Oh, the irony of misinterpreting!
>
> — Abruh (@abrasivetroop) January 18, 2024

It is undeniable that Godot will make strides in performance improvement in some way or another. However, the pace and extent of these improvements remain minimal when contrasted to professional-grade alternatives.

As one notable contributor of Godot said, “You can create great looking games. They just run slow.” This is the exact opposite of what good 3D support should be, since performance is crucial for 3D games. These inconsistent and contradictory properties of the engine describe the deceptive nature of Godot.

> You can create great looking games. They just run slow.
>
> — aaronfranke (@aaronfranke7) July 6, 2021

References

[1] Using the slowest data structure almost every time - Discussion at Godot, GitHub.

[2] Why isn’t Godot an ECS-based game engine? - By Juan Linietsky.

[3] Juan Linietsky on “death by thousands cuts” - Twitter.

[4] PierreTerdiman commenting on Juan Linietsky “death by thousands cuts” criticism - Twitter.

[5] Godot for AA/AAA game development - What’s missing? - Juan Linietsky.

[6] Godot is not the new Unity - The anatomy of a Godot API call - Sam Pruden, Sep 18, 2023.

Customization

Given slightly less trivial task, fruitful customization can only be achieved by modifying Godot's source code. In contrast to other game engines out there, Godot severely lacks the number of knobs to configure, mostly because Godot's built-in features are oversimplified in the interest of usability and clarity, so a lot of options remain hardcoded or unexposed.

Here's what we could consider lead developer's "philosophy" on customization[1]:

> *Any unnecessary customization is bad customization.*

While this compromise may allow engine developers to deliver a tool so that users can get results quickly, these results are only useful up to a certain point. That's why you'll oftentimes hear that Godot is mostly suitable as a prototyping tool instead.

According to Juan, customization in Godot is achieved by having a "clean codebase"[2].

> *Codebase is clean, well understood and easy to modify in order to add advanced or custom features.*

First and foremost, it's important to remember that Juan explicitly stated his indifference towards code organization and reuse, as discussed in the previous Do Repeat Yourself section. Therefore, this combination of neglecting code reuse and a need for custom modifications can inevitably result in regressions, and with Godot as a piece of peculiar software, such roadblocks are commonplace.

Despite this, Juan oftentimes suggests developers to build Godot themselves in order to "customize" Godot that way. This is substitution of concepts as proliferated by the authority of Juan. He presents this "build it yourself" approach as the main way to do "customization" in Godot. In other words, this type of "customization" shouldn't be presented as the main way to "customize" Godot, because following this approach leads to having to maintain a custom fork of Godot and a custom build system to deploy the engine to various platforms yourself, which is quite expensive.

If developers have to modify the source code, that's not true customization, that's just forking the engine which means maintaining a custom version of it. Due to this, it's very difficult to attribute great merit to Godot if modifying engine's source is the actual *necessity* in order to make any successful game with it, because it shouldn't be a necessity. The necessity to customize the engine via source code shouldn't be presented as the *best* way to

customize Godot, but rather *bullet-proof* way, and that always leads to additional maintenance effort which is better to be avoided.

Of course, you also have to pick the right tool for the job, yet Juan sells an *idea* that it's easy to modify the engine to achieve what they want with it, which doesn't reflect the reality, because one has to consider long-term shortcomings, not just near-sighted benefits that are too easy to fall into.

There exists even more lightweight engines in contrast to Godot that have more customization options without having to rebuild it from source, see Alternatives. So, one of those Godot's moving goalposts is ability to "customize" the engine by rebuilding Godot, because such ability to "customize" reads more like an excuse for lack of ability for real customization out of the box, as an actual feature of the engine. Pretty much every software can be "customized" by rebuilding it from source, Godot is not unique in this regard. Even then, it's difficult to call this as "customization" if we talk about this in terms of an end product rather than a software engineering perspective. Having this kind of approach would be tolerable if Godot were to be a library, but Godot is not a library; it's a "feature-packed game engine," as advertised by the Godot leadership.

Additionally, this kind of ability to "customize" by rebuilding Godot from source will likely lead to many custom forks out there, both open-source and commercial, and it's not clear whether Godot leadership cares about division of community in this regard.

Here's an analogy: imagine you have to disassemble an electric drill in order to fix it and to finally finish the job. You must agree that most professionals wouldn't even decide to use such a tool in the future if it's known to have such severe limitations. If someone likes to challenge themselves by having to workaround dozens of hidden limitations, that's totally up to them to use Godot, because Godot does follow the principle of "if don't like it, modify it yourself" religiously.

Taking into account above, if you stumble upon successful games made in Godot in the future, it would be mostly because of individual developers' incentive to modify Godot and/or by integrating third-party code, it wouldn't be Godot's merit at large, but Godot followers will sort of "cling to" that merit by saying that a particular game was made exclusively with Godot's built-in capabilities, which is a misleading, manipulative or blatantly false claim for the most part.

References

[1] Customization - By Juan Linietsky, Twitter.

[2] Why Godot? - By Juan Linietsky, Twitter.

Extensibility

Since Godot is open to various interpretations, in the realm of game engines, the vision of Godot's leadership remains ambiguous. The question arises whether they aspire for their engine to be regarded as the esteemed "Linux of Game Engines", renowned for its adaptability and flexibility, or if they seek to position it as an all-inclusive package akin to the versatile software suite, "Blender."

The ultimate trajectory depends on whether Godot opts for a lightweight approach that may risk rendering it somewhat ineffectual out of the box, or if they choose to incorporate additional features, which may result in a perceived engine "bloat," yet deliver a feature-rich experience. Neither approach is inherently wrong, but watch this satirical piece first:

> Is @GodotEngine the Blender of game engines? 😁#WaitingForBlueRobot #Godoverse #Blender #IndieGameDev #IndieDev #GameDev https://t.co/4slWTyVGWD pic.twitter.com/PltKj0QZeN
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) March 24, 2024

Labeling Godot as a prospective "Blender of Game Engines" would indeed fail to do justice to this so-called "visionary" project. It is crucial to recognize that the concept of being the "Blender of Game Engines" originates from a catchphrase masterfully devised by Godot's leadership, encouraging devoted Godot enthusiasts to propagate the message. Yet, it is essential to note that merely presenting Godot in this light does not guarantee its ultimate success.

Therefore, if Godot embraces the path of the "Linux" model, numerous proposals formulated by individuals of "mere mortal" status should be directed towards module, plugin, or addon development instead. The current course of decisions *seems* to indicate Godot's inclination towards this route. Despite this, you can still observe the Godot followers using the "Blender of Game Engines" catchphrase, almost as if it has become Godot's own slogan by now. Due to this, perhaps it would be helpful to remind Godot followers of their original slogan!

> *The game engine you've been waiting for.*

Lead developer of Godot presents his what he calls "philosophy" of pushing stuff out of the engine[1]:

> *Philosophy to me should be that something should be part of the engine only if ALL the criteria below are met:*
>
> *1. It's used relatively often.*
>
> *2. It's difficult to do yourself.*
>
> *3. There is only one way to do this feature.*
>
> *This rules out spring arm, terrain and I hope it can help to, in the future, remove features such as InterpolatedCamera, Vehicle, etc. that I believe should be outside the engine.*

If that's the case, this leads to the next question of whether Godot would be willing maintaining those features via plugins officially. If Godot aims to further remove features from the engine to allegedly prevent bloat, would those features be actually maintained by the core developers? If not, maintenance burden will lie on the shoulders of individuals who are not seen as official members of Godot (volunteers). Here's what Juan replies to these expressed concerns by Godot contributors:

> *The main concern about this is that, if something is not official it's not going to be maintained. This is where you guys are wrong.*
>
> *No one said these things should not be official, we can have an official repository for them and they can be first party extensions. Demos are maintained this way and they are up to date, so I don't see why it wouldn't work for an official extension repo.*

Godot demos repository is actively maintained because Godot as a game engine is *designed* to look good on the surface. They do everything necessary to sort of "sell" a good picture/image of Godot, so it makes sense for them to maintain such a repository regardless.

For the rest of the things, some features are already removed from the engine and are no longer official, contrary to what Juan said. Features like `InterpolatedCamera` are already removed from latest versions of Godot[2], and such classes are *not* moved to officially maintained repository. Features like these are pushed away to *unofficial* repository called **Godot Extended Libraries**, which is not handled by Godot leadership. Another example is `VisualScript`, which is removed from latest versions of Godot as well[3]. This means that most features that are pushed away like that won't be actively maintained or won't be maintained at all.

Even if Godot features may be moved to officially maintained extensions outside of the core

engine, this could further split the community, as those extensions would have to be installed manually by users, going against "feature-packed" approach which Godot has been advertising all these years.

If Godot truly upholds its claim of being "highly organic" and genuinely aims to prevent bloat, one would expect them to establish a concrete "Feature Removal Policy" to guide their development process. However, it is evident that such a policy does not exist within Godot organization. Unless policies are directly related to Godot's community, Godot adamantly avoids adopting any policies that could potentially expose their inconsistencies or contradictions. This blatant disregard for transparency and accountability raises questions about their true purpose concerning the development of extensions.

In the proposal titled "C++ Game modules," Juan proclaimed the following[4]:

> *For the rest, I really think it becomes a much more corner use cases, which does not make sense for having an entire feature for,* ***given the idea is to discourage the use of modules as much as possible*** *towards the future.* [emphasis mine]

In a proposal/pull request which aims to add built-in SQLite support, which has been neglected and ignored for more than three years now despite community needs, some Godot contributor comments on Juan's discouragement above[5]:

> *I believe rationale might be different, but real life is tough, so it will quickly turn into situation where Godot will be 'like open source' engine with few basic functionalities, then there will be many paid and closed source, licensed third party GDNative DLL's for any/each serious feature, without alternative of open-source modules...*

Godot leadership says that there's no guarantee that all feature proposals that received community support (in the form of thumb-ups or otherwise) will be implemented. And that's totally understandable. Godot cannot possibly implement all features that exist under the sun. But when contributors ask Godot leadership whether they would be interested in cooperation by promoting community extensions for Godot, such requests are ignored by Godot leadership, likely in the fear that this would lead to community division. This ambivalent attitude makes it very difficult to make a meaningful impact without going into unnecessary conflicts with Godot leadership.

References

[1] Problems with preventing “engine bloat” and the AssetLibrary - Godot, GitHub.

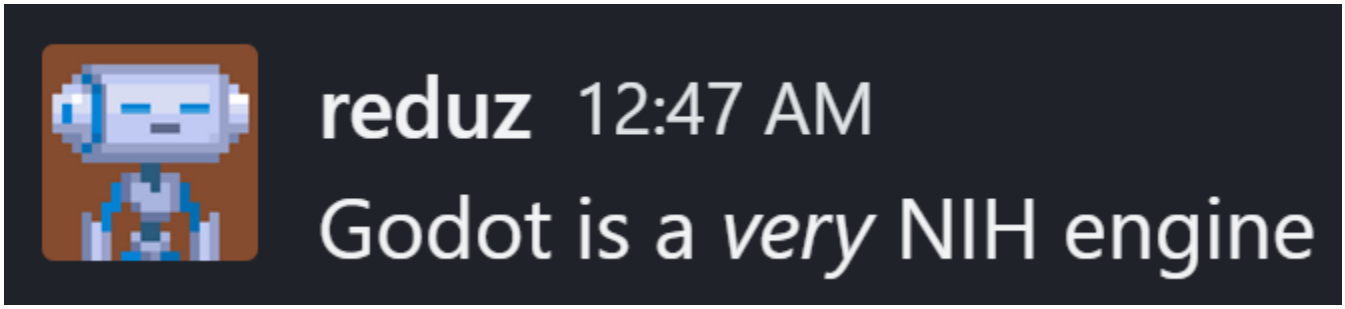
[2] InterpolatedCamera3D add-on - By Hugo Locurcio.

[3] Godot 4.0 will discontinue VisualScript - By Juan Linietsky.

[4] Discourage the use of modules as much as possible - Comment by Juan Linietsky.

[5] Add SQLite to Godot - Comment by Avril, Godot contributor.

Not Invented Here



Juan Linietsky. Source: Godot Contributors Chat

The NIH syndrome describes the tendency to avoid using third-party solutions. This term is usually used in pejorative meaning, but Godot core developers do not see this as a negative thing and take this approach deliberately, with a contrived rationale that Godot's architecture is unique and requires independent development decisions because of this, all without trying to fit third-party solutions into Godot's allegedly specific design. A lot was tested and dumped in Godot over years like SDL, Lua, Squirrel[1], Assimp, Box2D, Bullet etc. According to Juan, it has worked well:

> *Godot is a very NIH engine and it has worked very well.*

Godot strives not to rely on platform-specific libraries, and in some cases doesn't rely on third-party at all, for example, to render GUI components. However, Godot is not completely allergic to third-party solutions, but tends to prefer smaller sized libraries whenever possible, with varying degrees of quality...

Due to the above, Godot is a very NIH engine. Godot ends up doing a lot of things their own way. But this is not innovation either, because most of their work is based on *cloning* existing solutions, and the rest of the work is just the *reinvention of the wheel*. Godot developers say that it allows them to achieve "freedom", or a compromise of "glue vs politics", as lead developer says:

> *And I can definitely say that this because in my experience, you are having in one end of the scale NIH and Freedom, and on the other Glue and Politics, and choosing this wisely on a case by case basis is best.*

In an interview to Software Freedom Conservancy (SFC), Juan Linietsky says:

> *Coexistence with other free software projects is a bit difficult. Godot does mostly not make heavy use of other open source software as a base, and instead **we write our own versions of things** [emphasis mine]. This is because generally we have very precise needs to solve; it's easier to roll out our own solution than doing politics with other projects to see how to work together. So, unless a library we use is exactly what we need, we tend to roll out our own. Things may take longer, but Godot becomes a lot more consistent as a result.*

Have a look at how the core developers of Godot create their own version of physics. The image shows Godot's built-in physics on the left and a third-party physics engine called Jolt on the right. Godot-Jolt integration is unofficial and is not a part of Godot:

> I'm adding a physics-based minigame to my @godotengine game and compared #Godot Physics against Godot Jolt. Can you guess what I'll be using?😅 pic.twitter.com/ZbYlQwZeD8
>
> — Flo | Centi Games (Wishlist Wassermann on Steam) (@CentiGames) January 15, 2024

When we compare these results, the only thing that we can agree on with Juan is that the NIH approach helps Godot become more consistent because the engine stays consistently broken! For experienced game developers to make meaningful contributions to Godot, this so-called "NIH and Freedom" approach clearly creates a problem:

> Welcome to Godot where it's either "their way" or "no way", no matter if you're right or wrong.
> Dig up the reasons why using SDL, bgfx, physics libs (before the built-in collaps) and other industry standard libraries were rejected just because Godot is a NIH engine.
>
> — Kornel Kisielewicz (@epyoncf) February 3, 2024

In an intriguing pursuit of irony, Juan has amusingly proclaimed his disinterest in politics dozens of times, all the while actively being involved in a myriad of political and ideological debates on Twitter/X, ranging from technical topics to more serious ones, like military, gambling, etc. As we venture further into subsequent chapters, do bear this revelation in mind, quoting Juan[2]:

> *I am only here for gamedev, not for the politics.*

Because Juan is a political maestro when it comes to handling tech, you can clearly see how he sticks to his guns and adopts a rather snobbish stance on, let's say, rendering backends

like BGFX, quote[3]:

> *Guys, thanks a lot for your enthusiasm, but I am the one doing the rendering work in Godot, not you. I've been working on 3D rendering for 25 years so, if I am telling you that things as they are now are optimal and BGFX will just stand in the way to being productive I hope you believe me.*

Godot is often criticized for being extremely NIH, so much so that Juan himself wrote a post about it, literally claiming that no one has managed to make a game engine with pre-built stuff, which is obviously a false claim:

> > why nobody managed to do it yet?
>
> Many people actually do this, just not as generic engines. They use these libraries to make engine tailored for game they are making. Game is glue of bunch of 3rd party stuff.
>
> — Бранимир Караџић (@bkaradzic) March 19, 2024

Analyses by experienced developers highlight Godot's NIH decisions, evident in its choices of rendering techniques that disregard established industry standards:

> Originally posted as a reply to: https://gist.github.com/reduz/c5769d0e705d8ab7ac187d63be0099b5
> Turned into a gist due to high likelihood of deletion. Also edited down to not include irrelevant trolling as to be useful to someone else considering Depth Reprojection.
> Yes I know SSR and Parallax Corrected Shadowmaps work, but the consequences of errors in those depth tests aren't as high.
>
> ## Lack of Generality (oh the irony)
>
> You yourself state that this is general purpose engine, how is a technique that will have trouble with:
>
> - deformables (cloth, vegetation, skinned meshes, etc.)
> - dynamic objects (how are you going to reproject the depth of a moving object?)
>
> general purpose at all? For now all I see is that the only occluders you'll support must be static triangle meshes. And by static, I mean **truly static** no movement from frame to frame even as a rigid body.
> I thought general purpose could mean like a First Person game with you know, animated characters which could occlude vast portions of the screen?
> You reiterate time and time again that this is not an AAA engine, hence don't you think that it

would be nice NOT to require artists/users to make specialized simplified "occluder geometries" and have to remember to set them?
I mean you expect ray-tracing to be fast enough to replace a z-prepass, and hoping this will be the case "because there's only a few pixels to trace for". Given that you want this to run on mobiles and the web and your raytracing software fallback layer **will have to be faster than a z-prepass** (because thats the only reason not to just do a z-prepass and not occlusion cull) which will probably necessitate a separate, simpler BLAS per occluder, than the one you'll use for the shadow raytracing. Nice fun way to increase your memory footprint for no reason.
This is before I even point out that your users will sure appreciate having to "bake" occluder BLASes for their occluder mesh, which they'll also appreciate having to make and maintain.
The final nail in the coffin comes from the fact that unless you want to give up on streaming static chunks or like building the TLAS yourself (which you might for a fallback layer with Embree), Vulkan's Acceleration Structure is a black box. If you want to use as much as a single different BLAS in an otherwise identical TLAS, you'll need to build a new TLAS from scratch (you can't just copy the shadow raytracing TLAS and hotswap the pointers to make it point at different simpler BLASes even if the input BLAS count and AABBs match). **This workload does not scale with resolution, needs to be done *every frame*, even if you make your culling depth buffer 1x1**

2-4x more code to maintain, complexity and fragility

Again the AAA argument, you don't have the resources nor the expertise to maintain complex and duplicated codepaths.
Your design forces (I hope you're aware, but with every reply I loose faith) the renderer to partition the drawing into two distinct stages:

- static objects
- everything else

You then need split your renderpass into two, so that you can "save a copy" of the depth buffer before you draw other non-static things into it. Them tiled mobile GPUs are sure gonna love that. The fun part (as I promised to expand upon) is that as soon as something starts moving (i.e. a door) you'll need to exclude it from the static set and not draw its occluder, because you cannot reproject its depth.

There are only 2 ways to do occlusion culling

Basically it depends on whether you want rasterization or compute:

- rasterization => abuse depth-test only, draw simple conservative occludee bounding volumes (simples is AABB or OBB, can be convex hull) but with a fragment shader which writes out per-drawable visibilty to SSBO (z-prepass like)
- compute => HiZ by mip-mapping the depth and only testing a screenspace 2x2 AABB or the 3D AABB, like `vkguide`

The HW occlusion pixel counter queries are not an option, because only one can be active per drawcall and they are super slow even with conditional rendering (which was invented to save you from GPU->CPU readbacks). Its suckiness the reason why that Depth Buffer + Occlusion

Testing at low res on the CPU was popular at DICE and Crytek.

Mmm the latency!

So anyway, at some point before you even start testing objects for visibility after frustum culling, you'd need to reproject that previous frame partial depth buffer and raytrace the holes, but you can't do that before polling for input. Then you need to do the occlusion tests, you don't have a shadowpass or anything else to keep the GPU busy in the meantime.

Have fun maintaining and optimizing the code

The divergence on the Reprojection and Raytracing shader is gonna be some next level stuff, I'd personally love to see the Nsight trace of how much time your SM spends idling if you ever get far enough to implementing it.

You'll probably dig yourself into a hole so deep you'll consider doing "poor man's Shader Invocation Reordering" at that point and blog about it as some cool invention.

Nobody tried Depth Reprojection for a good reason

You're probably not the first person to come up with "last frame depth reprojection" as an idea, now think about why nobody went through with it.

Raytracing to "fill gaps" doesn't make the idea special.

Reprojection introduces artefacts - false culling positives

There is simply nothing to reproject, depths are point sampled and you cannot interpolate between them (even with a NEAREST filter). The depth values are defined and valid ONLY for pixel centers from the last frame.

A depth buffer used for culling needs to be conservative (or some people say eager), therefore the depth values for such a depth buffer can only be FARTHER than "ground truth".

No matter if you run a gather (SSR-like) or a scatter (`imageAtomicMax/Min` - then you've really lost your marbles).

Don't believe me, try reprojecting the depth buffer formed by static chain linked fence (alpha tested or not does not matter) and call me back.

Essentially every pixel turns into a gap that needs to be raytraced.

This makes no sense from a performance standpoint

The only sane way to reproject is via a gather, which is basically the same process as Screen Space Reflections or Parallax Occlusion Mapping.

Let me remind you that a z-prepass usually takes <1ms and if it takes more than that alternative methods are considered for culling.

You've now taken one of the most insanely expensive post-processes (maybe except for SSAO) and made it your pre-requisite to culling (slow clap).

To put the icing on the cake, a reprojected depth (programmatically written) disables HiZ, so any per-pixel visibility tests (if you use that) done by rasterizing the Occludee's Conservative Bounding Volue get magically many times slower.

Finally there's that whole polling for input, frustum culling, depth reprojection, occlusion culling dependency of the first renderpass which increases your latency.

Now imagine, if only a solution existed that gave you 99% correct visibility and at full resolution in far less time than a z-prepass or this weird SSR?

The Established "AAA" solution is more robust, general and simpler

I gave you a solution thats "essentially free", it gives you all the visibility data in the course of *performing work you'd already be performing anyway* which is the most robust thing that will ever exist for rasterization, it:

- has actually been implemented before and used in production
- requires no special HW to be efficient (unlike Ray-Tracing)
- gives 100% pixel-perfect last frame visible drawable set
- is doable in Forward+ as long as you have a z-prepass which you should have anyway
- knows 95% of its Potentially Visible Set before the next frame starts, so you can start drawing right away, without incurring extra latency
- has no issues with procedural or deformable geometry
- requires no prebaking
- requires no extra special geometries, metadata, settings or parameters/heuristics to tweak
- is completely transparent to the user (no popping, no intervention needed)
- 100% accurate and artefact free (the second depth testing pass takes care of disocclusions)
- is scalable (you can interleave / subsample the visibility info, you'll just have more "disocclusions")

In case it wasn't clear both the "last frame visible" and "disocclusion" sets come from the intersection of the "post-frustum cull" set for the new frame, not the whole scene.

bait.md hosted with ❤ by **GitHub** **view raw**

Godot's NIH approach is further confirmed by reviews of those who use Godot in production and face its limitations, *emphasis mine*[4]:

> *Godot is **absolutely amazing at a lot of things**, and **terrible at a lot of other things** at the same time. A potential idea would be to rip things apart, scrap a lot of the bad parts, and rebuild internals to be less brittle.*

> I was told in IRC back when godot dev team used IRC for communication, that godot src is very NIH (not invented here) by design. I wasn't familiar with this term but boy am I now. (NIH is the tendency to avoid using any third party code other than your own). (14/25)
>
> — Winterpixel Games (@winterpixelco) July 7, 2021

Developers often have to *customize* Godot by rebuilding it from source just to make it work for their use cases in production. Ripping things apart is often necessary to achieve anything

slightly more intricate in Godot, read Customization chapter.

While evaluations above may be accurate to some degree, they could also arise due to the discrepancy between the initial *expectations* set by Juan (promising it to be absolutely amazing) and the actual performance experienced by the user (finding it terrible).

Godot developers might be grappling with feelings of disappointment and frustration, as their perception of the engine's abilities clashes with the reality they encounter. This cognitive dissonance may lead them to express both positive and negative sentiments concurrently as they attempt to reconcile their initial excitement with the encountered limitations.

> Yup, but I've also worked on many-a Godot projects-- there's a lot of limitations in the engine due to poor engine/feature design and the "NIH syndrome".
>
> Godot is, by far, the engine I've had the most number of problems with, where even the most basic features need work arounds.
>
> — LillyByte (@LillyByteGames) March 14, 2024

Yet they continue to use Godot, even when they admit it's the engine that they've had the most trouble with in their entire lives. This social phenomenon mainly originates from the belief in a utopian "bright future" that many Godot users stubbornly hold onto, despite facing critical limitations:

> Here is a detailed list of everything wrong with Godot. I hope people will realize that I am not the enemy of Godot. I care about the future of this engine if you do too then you should start talking about this.#GodotEngine https://t.co/5YBXEgTKoW pic.twitter.com/w0Res4CzLM
>
> — Abruh (@abrasivetroop) February 25, 2024

Conclusion

Due to the NIH syndrome, Godot won't prioritize features like C# over GDScript because their priorities account for independence, see Simplicity chapter. This way, they achieve *their* so-called freedom, but end up making others dependent on their in-house technology, see Freedom. Ironically, the fact that Godot left Software Freedom Conservancy for Godot Foundation reinforces the NIH syndrome even on an organizational dimension!

Of course, the Godot leadership would like to see their engine adopted by the big players out there, and they go out of their way to do so. As a result, the leadership and core maintainers would be seen as experts in the game development industry, so they could provide paid services through a syndicate of commercial companies such as W4 Games or Ramatak, which they co-founded. Regardless of their intentions, the above is one of the reasons why Godot is NIH. There are other important reasons for Godot's approach, as you'll discover in the following chapters.

References

[1] Juan Linietsky talks about Godot's past - Twitter.

[5] A brief introduction to the Godot Engine with Juan Linietsky, Lead Developer - Software Freedom Conservancy.

[2] Juan Linietsky discusses Twitter policies - By Juan Linietsky.

[3] Juan Linietsky on integrating a rendering backend - Godot, GitHub.

[4] WinterPixelGames comments Godot's reaction to O3DE, mentioning Godot's NIH approach and ideas on rewriting the engine - Twitter.

Freedom

Godot is declared as free and open-source software (FOSS). That means you have the freedom to run, copy, distribute, study, change and improve Godot by contributing to its development. Software freedom aside, Godot's leadership propagates the idea of *freedom* as the pivotal principle of Godot's purported success.

Those who've been immersed in their fair share of Open Source shenanigans might ironically resonate with these memes, sarcastically tailored for Godot, because, of course, we revel in our freedom to do so! 🙃

Freedom to improve

Here's how the lead developer of Godot presents it to the public: a picture of the open ocean with the word "Freedom"[1]:

Will clarify this as many times as needed:

*Godot is **not** a company, it' s FOSS, volunteer based. **No one** tells anyone want to do. Contributors work on what they want, whenever they want. There is **no** resource allocation, **no** central authority ordering people around.*

It's odd how Juan consistently feels the need to "clarify" this. What's the deal with that? Godot isn't particularly exceptional when it comes to software freedom compared to other FOSS projects. So, why does Juan keep emphasizing it so vividly, almost as if he's attempting to convert users and contributors into some kind of religious following?

If contributors expect their improvements to be integrated into the main project, they cannot opt to work on features not initially approved by Godot's leadership. It operates in a manner similar to companies, but with a key distinction—99% contributors are unpaid, unlike in a corporate setting. If this is meant to symbolize freedom, what kind of freedom are we really discussing in a project that touts itself as *community-driven*?

Freedom to change

Even if we set aside the contribution process in Godot, the freedom to modify Godot is

undeniably an advantage. However, there are other factors to consider. When you ask Godot users about the advantages they see in Godot compared to other alternatives, the most common response is usually: "It's free!" This benefit is overemphasized by Godot users, and the need for software modifications tends to be contrived, because in reality, most Godot users, who are predominantly hobbyists, don't really need to build Godot from source.

Certainly, there are advanced use cases that require rebuilding Godot from source. Considering the significant limitations in customization capabilities within Godot, it's highly probable that professional developers would indeed need to build Godot from source themselves.

However, for some reason, those waiting for Godot seem to believe that the freedom to modify the engine is the most crucial aspect, almost as if freedom itself is an inherent feature of Godot[2]:

Comment
byu/itsarabbit from discussion
ingamedev

The issue with this misguided sense of empowerment, expressed as "*it's open-source, so I can do anything I need myself, the possibilities are endless!*" is that, in many cases, you may not know how to do it yourself. This means spending a significant amount of time learning how to develop the desired feature, compared to someone who already possesses the skills to do so.

Consequently, in many instances, it may actually be more cost-effective to purchase an existing commercial solution, even if it's closed-source, which is likely better than anything waiters for Godot could create themselves.

Freedom of interpretation?

The simplified architecture or lack of some essential features is not just "the engine has not matured yet" or "just wait for it", a rhetoric that you may hear from fervent waiters for Godot. The problem is that a lot of people don't actually know what kind of approach Godot follows, so they tend to project whatever vision of their own perfect game engine onto Godot, basically creating a bubble.

If we explore the origins of the "Godot" name, as you've hopefully discovered in the Value of Waiting chapter, it frankly reveals the core approach undertaken by Godot's leadership. This

kind of freedom of interpretation concerning Godot's purpose is certainly not something to aspire for.

Freedom to make conclusions 🙂

Due to all above, you may find yourself working on a tool rather than working on your dream using a tool. Therefore, you either pay with money, or your work. Time is expensive. Even as a proponent of free software myself, I say that there's no such thing as freedom in this particular context, I think it's a myth when we talk about maintenance and support specifically.

To make an analogy: it's like apartment renovation. You either do it yourself (long), or you hire professionals to do this job. Neither is totally free or effortless. The problem is that most Godot followers think that their engine offers them the ultimate freedom that will liberate them from all the hard work. Not at all. With open-source, you're all on your own, and it's equally demanding.

In the upcoming chapters, you'll find that the Godot leadership sees contributors as a means to an end rather than an end in themselves. The decision-making process is only allowed under specific conditions, based solely on trust, without consideration for merit or even the do-ocracy, a default governance model prevalent in many Open Source projects.

References

[1] Godot is FOSS, volunteer based - By Juan Linietsky, Twitter.

[2] Godot is better for AAA than Unity, because it's open-source - Comment at Godot subreddit.

Governance

What we have covered so far in Organizational Culture and Development Philosophy sections mostly answers an important question of “Why”. This section answers the question of “How” in a meta sense.

For most types of Community-Led Development, it boils down to who makes key decisions in a project by privileged members and how strongly community itself affects the decision-making process. So, what kind of criteria is used to select those members that would lead and make key decisions for a project to realize its vision?

Usually, project founders are the ones who initiate all types of open-source community-led projects out there. Their task is to lead the community in such a way so that project’s vision is implemented to fullest extent possible, as defined, implemented, and/or consulted with community.

Open-source is all about developing software for the most part. Obviously, there exist *users* and *developers* of that software, who are also users. People who want their changes to that software be incorporated into a project are usually called *contributors*. If there are no contributors, project founders are all on their own! Therefore, leadership may want or need to delegate the decision-making power to others at some point. Of course, depending on concrete project’s governance, leadership may be turned over.

During the decision-making process, you may encounter various problems:

- What if the interests of different people contradict each other?
- What if the desires of a large number of people differ from the vision of the leaders?
- How important is community discussion, and how important are leaders’ decisions?

Therefore, in order to minimize unnecessary contradictions and interpersonal issues between users, members, and leaders, it’s equally important to define project’s governance model, just as Development Philosophy previously covered.

According to classification by Red Hat, there exists several major types of governance models for open-source projects out there[1], such as:

- *do-ocracy*
- *founder-leader*
- *self-appointing council or board*
- *electoral*
- *corporate-backed*
- *foundation-backed*

However, due to the nature of open-source and the following topics that we're going to cover, we'll mostly focus on factors and aspects that are closer to human interactions here. Taking into account above classification, let's classify major types of governance using "-*cracy*" terms[2]!

Types of governance

Democracy - everyone has a say in what gets done, and elected people get the job done. Closely corresponds to *electoral* model.

Meritocracy - the most qualified people for a job are selected for that job.

Do-ocracy - people choose roles and tasks for themselves and execute them. Whoever does the job gets it, no matter how well they're qualified.

Trustocracy - people cannot choose tasks for themselves. Roles are assigned solely upon trust, regardless of qualification or the amount of work done by people. Closely corresponds to *founder-leader* model.

References

[1] Red Hat Blog: Understanding open source governance models - By Dave Neary, Josh Berkus, Katrina Novakovic, Bryan Behrenshausen.

[2] CommunityWiki - DoOcracy.

Community-Led Development

There exist many definitions of community-led approaches. The discrepancy in “community-led” verbiage suggests that generally, a lot of organizations don’t pay necessary attention to this, so they tend to neglect to convey their de-facto governance model. Because open-source projects attract people from all around the world, it’s extremely important to be explicit about this.

According to the research conducted by Tamarack Institute which promotes strategies for community change, some organizations may overpromise or be disingenuous by saying that they are community-led while they don’t intend to give much power to the community, or arrogantly assume they can speak *for* community or *know* what community “actually” wants.[1]

This issue is quite prominent in open-source community. The vast diversity of people and different cultures which entail contrasting values, assumptions, beliefs, expectations etc. makes this extra ambiguous. Because of this, all open-source community-led projects must strive to eliminate this ambiguity.

For clarification, the previous Governance chapter discussed the aspect of *control*. In contrast, this chapter describes types of *ownership*. While these concepts may overlap, they have distinct meaning.

Types of community-led approaches

Source: Understanding Community-Led Approaches to Community Change

For our purpose in relation to Governance, let's classify and define community-led approaches into these categories:

- ***Community-owned*** - vision is completely defined and implemented by community itself. Community is in control of all decisions. In a hypothetically ideal world, no governance model is required here, there's an assumption that community itself is able to effectively fulfill its vision on its own.
- ***Community-driven*** - vision is defined and implemented by community at large. Community with its leadership and partners collaborate to make decisions. Governance is typically *electoral* or *do-ocratic*.
- ***Community-shaped*** - vision is defined by community within the scope set forward by leadership. Leadership is in control of most decisions. Governance is typically conducted by *self-appointing board* or *foundation-backed*.
- ***Community-informed*** - vision is adapted by community feedback and consultation. Leadership is in control of all decisions. Governance is typically *founder-leader* or *corporate-backed*.

Characteristics

Source: Understanding Community-Led Approaches to Community Change

Sense of responsibility

All types of community-led projects must have a sense of *responsibility*, and frankly every type of project out there regardless of whether it's open-source or proprietary.

Ask leadership whether they are committed. If they respond that they do not feel responsible nor obligated to do anything, it's not the project that you're looking for, because a sense of responsibility is essential for all types community-led projects.

💡 If you feel repulsed reading above, it's a sign that you need to develop your leadership qualities! Of course, this *assumes* that you'd like to lead a *community-led* project, and not just a single person working on a project, otherwise a sense of responsibility is not required. Even then, there exists a notion of *self-responsibility*. Namely, how committed you're to your own intrinsic values and goals.

How well you are "committed" does not imply spending 8 hours a day working on a project. Rather, it's about demonstrating how well your actions align with your words. Commitment is not time-bound; it's mostly about adhering to the principles of your project that you created or maintain.

Bottom-up

With community-led approaches, users and contributors trust community leaders to manage a project using the “bottom-up” principle, where community holds power and makes key decisions. Decisions are mostly taken via consensus-building process. Even when community doesn’t directly make those decisions (as in *community-informed* approach) it’s still important for the leadership to actually reflect and satisfy community needs and be committed to fullfil its vision.

Conclusion

If you’d like to run a *community-led* project, the usage of decision-making “button” shouldn’t be the demonstration of *your* power as a lead developer or project manager. Rather, your decisions must reflect consensus of contributors and users. Otherwise, you should re-evaluate how much power you actually intend to give to community, and become either a *community-shaped* or *community-informed* project. Do convey this information explicitly without re-defining *community-driven* term. Don’t let potential contributors fall victim to this type of ambiguity. Strive to create an environment where actual consensus can be reached.

References

[1] Understanding Community-Led Approaches to Community Change - By Lisa Attygalle, Tamarack Institute.

Waiting for Community

Godot Engine is an example of a open-source project which has quite convoluted governance model. If you go through all the circles of hell yourself, you may figure out that your understanding of Godot's governance model keeps evolving the more you invest into Godot's development as a contributor.

Typically, when a project doesn't have a formal governance model documented, all open-source community-led project are assumed to be governed using the *do-ocracy* model.

For quite a long time, Godot didn't have its governance model documented, until community urged Godot leadership to document it. But as you'll find out, even the *documented* governance model still doesn't reflect its *real* governance. The mere fact that it took Godot leadership seven years to write down their governance model suggests that their governance has been *do-ocratic* all this time at the very least. Even then, it's not that simple.

This is what Godot leadership typically says and portraits (rhetoric):

- To users: *"Community is in charge, everything is transparent and democratic"*
- To contributors: *"Development is solely based on trust, not meritocracy nor democracy"*

The following chapters are going to answer the question whether Godot is community-driven and what kind of governance model it follows.

Democracy

Is Godot a democracy? The short answer is: definitely not. When people ask this question, even the project manager of Godot says along the lines of:

> *Godot is in no way a democracy.*

But why does Godot leadership keeps *depicting* it as such? For example, when they write news articles about Godot, they show images of a "Godot candidate" that you can allegedly "vote" for[1]:

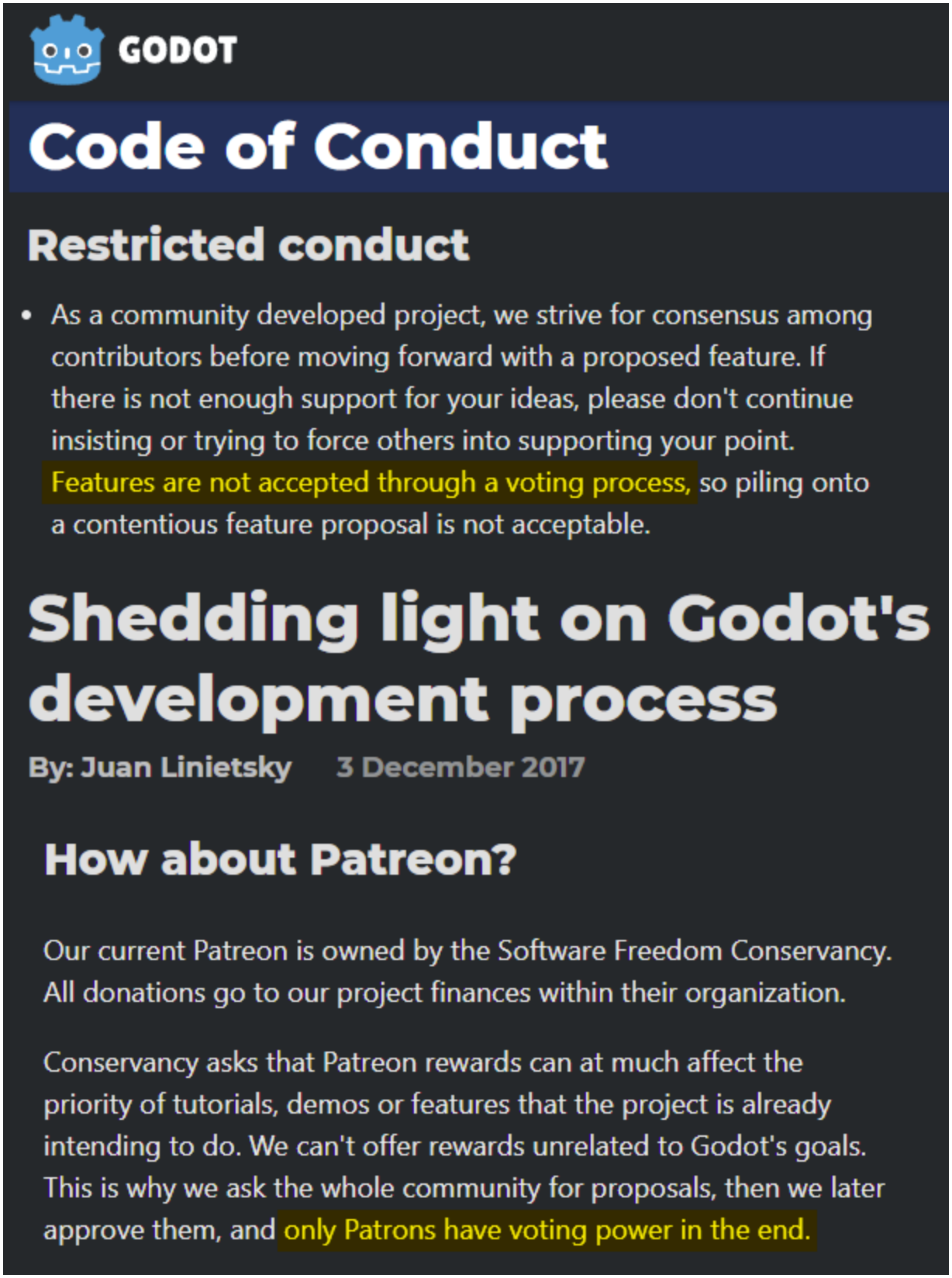
Another example is when lead developer writes an article on April's Fool, suggesting that Godot "democratizes development"[2]:

> *We also want to democratize game development. As you know, Godot Engine developers and other members of the community have always listened to your concerns, troubleshot your problems, fixed bugs you reported, implemented many of your suggestions and merged untold numbers of pull requests. This may have sufficed at one point in our history, but can*

> *continue no longer, as it's clearly a greivous form of communism, and communism is the enemy of democracy.*

Of course, this is framed as a joke, but they walk a fine line with ambiguity to make it seem like their development is based on democratic principles. As a result, users who aren't familiar with how open-source works tend to believe that Godot operates like a true democracy. Unfortunately, this misconception is common among regular users in the Godot community, especially since many of them are hobbyists.

While there's a difference between democratic *values* and democratic *governance*, Godot as a project creates the false impression of being democratically *governed*. For instance, many users mistakenly believe that features are approved through a voting process. In reality, this isn't the case. According to an article by the lead developer on Godot's development process, only those who financially donate to the project have the right to vote.[3]:



According to Godot's Code of Conduct, "features are not accepted through a voting process."

> *As a community developed project, we strive for consensus among contributors before moving forward with a proposed feature. If there is not enough support for your ideas,*

> *please don't continue insisting or trying to force others into supporting your point.* ***Features are not accepted through a voting process*** *[emphasis added], so piling onto a contentious feature proposal is not acceptable.*

However, in the article "Shedding Light on Godot's Development Process," they also claim that "only Patrons have voting power in the end."

> *Conservancy asks that Patreon rewards can at much affect the priority of tutorials, demos or features that the project is already intending to do. We can't offer rewards unrelated to Godot's goals. This is why we ask the whole community for proposals, then we later approve them, and* ***only Patrons have voting power*** *in the end [emphasis added].*

I hope you see the contradiction here: if features aren't accepted through a voting process, then no one should be voting on anything, which means Patrons shouldn't have voting power either. Don't blame me if this doesn't make sense—it's Godot's leadership that's created these inconsistencies and contradictions! 😁

This creates different tiers of users and developers in the eyes of Godot's leadership, depending on whether they financially support the project. This approach might be acceptable for a *community-informed* project, but Godot claims to be *community-driven*, where decision-making is typically based on *do-ocracy*. This means that those who contribute can also decide which features to work on, based on votes from the **Godot Improvement Proposals**. In a truly community-driven project, such decisions shouldn't depend solely on Godot's leadership and the financial contributions of voters.

Even then, as you've hopefully read in Waiting for Philosophy chapter, there's apparent chicken-or-egg problem: one cannot prioritize features through voting if vision and mission of Godot are not defined in the first place, and Godot's roadmap keeps changing in accordance to their mood.

You'll also find that Godot's leadership co-founded a commercial company called **W4 Games**. If you go to their LinkedIn page, you'll see yet another instance of this so-called "democratization", quote:

> *W4 Games is an Irish start-up which plans to revolutionize the online gaming market with a cloud platform based on open source technology, which will unlock and* ***democratize*** *[emphasis added] the online multiplayer market for all independent developers.*

On top of that, in an article by Bryant Francis on GameDeveloper.com, where the author

interviewed Juan Linietsky and Rémi Verschelde, it's mentioned[4]:

> *But Godot's open-source nature means features can't just be solicited like they are from a licensed engine.* ***They have to be voted on*** *[emphasis added] and implemented by the community.*

That said, if the Godot leadership had decided to create a commercial company for Godot while claiming they want to democratize the market for developers, you'd expect the open-source Godot Engine to reflect those same values, right? But in reality, that's not the case. The leadership is content to keep things vague, attracting people who thought features would be decided by a vote. Who wouldn't want to feel like their voice matters, right?

Conclusion

If you're a Godot contributor, don't be misled by the number of thumbs-up or user support your proposal receives. If donors or sponsors don't vote for it, your proposal will likely be left in limbo. Even more troubling, donor votes might be ignored by Godot's leadership. Donors, who expect transparency, may find themselves waiting indefinitely for features they supported, all while refusing to admit they've been fooled. The more invested they are, the harder it becomes to accept the truth. Moreover, the legitimacy of the voting process is highly questionable. It's alarmingly easy to disguise a unilateral decision as a community choice when you have a loyal fanbase that trusts you unquestionably. In Godot's case, this trust is clearly being exploited, as you'll find out in the next chapters.

References

[1] Release candidate: Godot 3.1 RC 2 - By Rémi Verschelde.

[2] Godot is now really free! - By Juan Linietsky.

[3] Shedding light on Godot's development process - By Juan Linietsky.

[4] Godot founders had desperately hoped Unity wouldn't 'blow up' - By Bryant Francis, Senior Editor at GameDeveloper.com, September 5, 2024.

Meritocracy and Do-ocracy

Is Godot a meritocracy or do-ocracy? According to lead developer's article called *"As an Open Source project, Godot is more than a game engine"*, Godot's development is truly meritocratic[1]:

> *But this is not all, Godot development is a **true meritocracy** [emphasis added]. Code is only merged when deemed worthy by the other developers. Every change is discussed for pros and cons.*
>
> *This process is led by our large user community, who usually sets our priorities via issues (or just random rants on our community sites :P). This same community helps with the testing and works directly with developers implementing each feature to ensure it works as best as possible. Our process is truly **community-driven** [emphasis added] from beginning to end.*

But if you ask the lead developer directly, he will say that there's absolutely no meritocracy in Godot. Again, I'd like to emphasize that this comes from the same person who said Godot is a true meritocracy[2]:

> *For those who think Godot is a "Meritocracy", I hate that word because it is too ambiguous. Godot is developed by always seeking agreement before move forward, there is no real hierarchy, status or governance system used during development. Roles are only organizational.*

The project manager explicitly describes Godot's development as *do-ocracy*. Rémi always attempts to "clarify" what Juan really "meant" to say in order to save his face that way[3]:

> *One can say that do-ocracy is also a meritocracy: those who do things have merit for doing them. But it's not a meritocracy in the sense that those with most merit (technical proficiency, etc.) are the ones in charge. There are many factors beyond merit that can make you a do-er.*

I hope that you see contradictions here. Of course, you may say that Juan didn't realize the meaning behind those words back in the days, but this just shows apparent lack of focus and competence as a project leader. Before deciding to write an article, it makes sense to research different types of governance models that exist in open-source communities, right?

However, even when Godot leaders talk about their governance model, people who ask

questions are still confused and raise concerns about all of this. For example, if there's no hierarchy in Godot, why do Juan and Rémi are mostly the only ones that are allowed to have the ultimate power to merge changes and block consensus for all proposals, despite having "thousands of contributors" that are "growing exponentially", as Juan keeps regurgitating? To quote a respectable member of Godot community[4]:

> *It is very strange that, supposedly, there isn't a hierarchy, but the reality is: there are people with power to ban others from the community, while some don't. There are people with power to take and implement decisions, while others don't. What kind of "non-hierarchy" is that. I'm not saying it should run without someone taking responsibility. I'm saying: what Juan said don't reflect what actually happens.*

Mind you, this is just an example, and a little bit of research can confirm that these contradictions do exist and are typical in Godot community.

References

[1] As an Open Source project, Godot is more than a game engine - By Juan Linietsky.

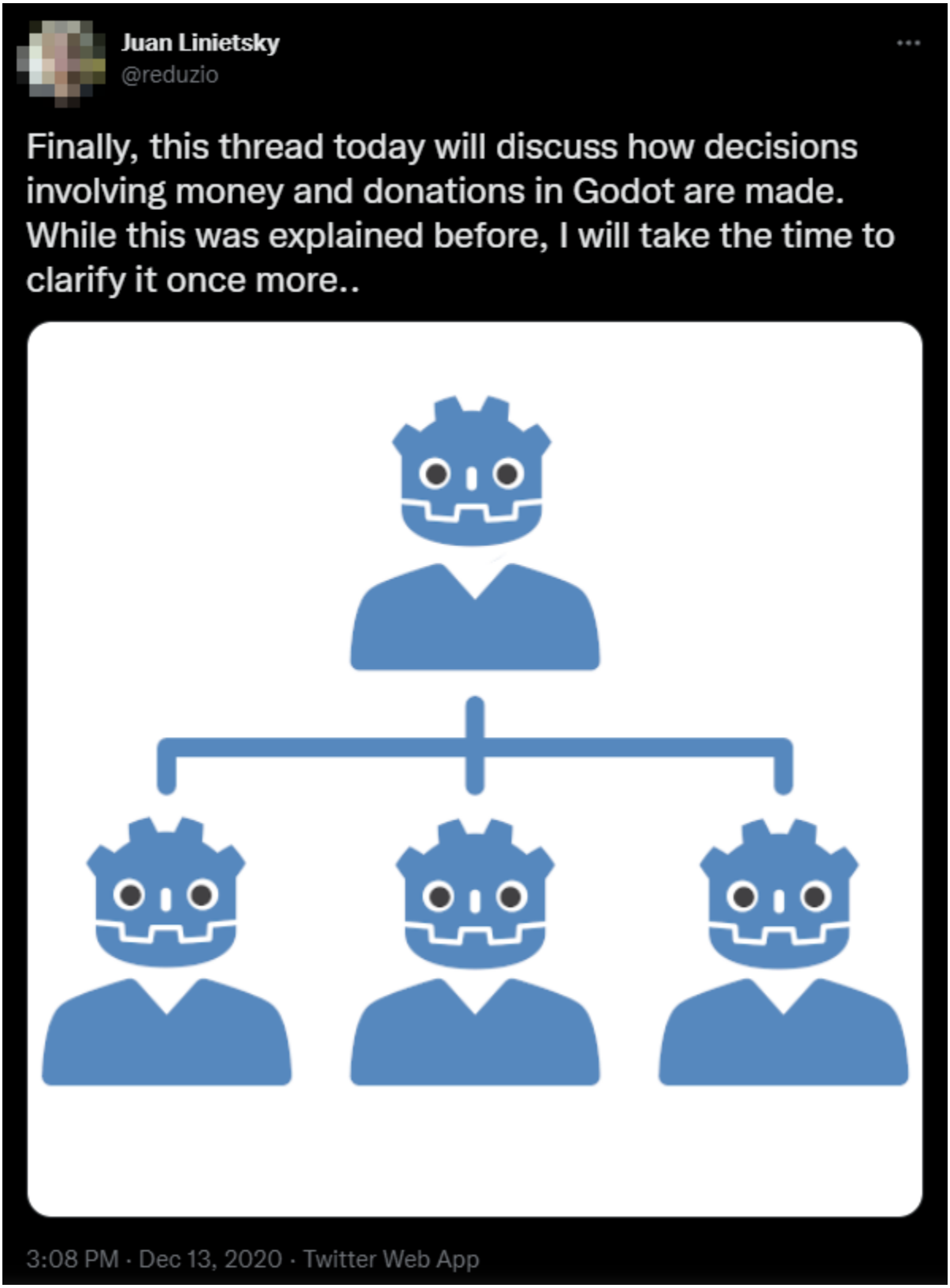
[2] Meritocracy does not exist in Godot - By Juan Linietsky, Twitter.

[3] Do-ocracy is also a meritocracy - By Rémi Verschelde, Twitter.

[4] Pigdev about Godot's hierarchy - By Pigdev, Twitter

Trustocracy

Is Godot a trustocracy? The short answer is: yes.



Every decision in Godot is based purely on trust. Of course, this trust can be earned in

numerous ways, which in some cases accounts for individual merit and devotion of contributors to the project, or even your political orientation. Believe it or not, but the quickest way to earn Godot's leadership trust is by promoting Godot. Which means that you don't necessarily need to be a developer at all. This is one of the reasons why Godot gets so overzealously promoted on various game development forums and conferences by individuals.

Godot leadership has an inner circle of people who are called *advisors*. These people are trusted by Godot leadership. This list of people is not public, but if you go to Godot Contributors Chat, you can verify that there exists a hidden **#advisors** channel there by mentioning it. Beware, if you do this, Godot leadership won't trust you! This advisory group is used for consultation purposes, and the existence of such a group suggests *community-informed* approach to development.

So, who makes ultimate development decisions in Godot? Those people who the lead developer of Godot personally trusts, obviously! Everything goes from the top to the bottom. The first person which Juan assigned for the position of a project manager was Rémi. Since then, Juan and Rémi remain mostly the only people who are generally allowed to make key decisions for the project.

As Godot kept growing, the project entered as a member of **Software Freedom Conservancy** (SFC). This organization *requires* projects to define their *Project Leadership Committee* (PLC). The existence of Godot PLC may suggest that Godot's governance could be based on *self-appointing council or board* governance model. But the other people that you see listed in Godot PLC mostly exist for the formal purposes in order to show that Godot has horizontal structure at least on the level of leadership, and that those people (Juan, Rémi, and a few others) don't control every decision in Godot, again, because SFC *requires* them to provide a list of such people. But in practice, the rest show **hierarchical** devotion to Juan.

Which brings us to another point: is Godot's development based on *founder-leader* governance model, then? On the surface, it may not look this way. But due to the nature which stems from the chain of trust, Juan can be seen as a BDFL (*Benevolent dictator for life*) of Godot Engine. He has unlimited use of veto, and many people in Godot community unquestionably trust Juan, which means that Juan has mostly the ultimate power over all decisions.

You may ask: "How is it possible that Juan makes all decisions for Godot? Isn't Godot developed by community?" You'll figure out how by the end of this book!

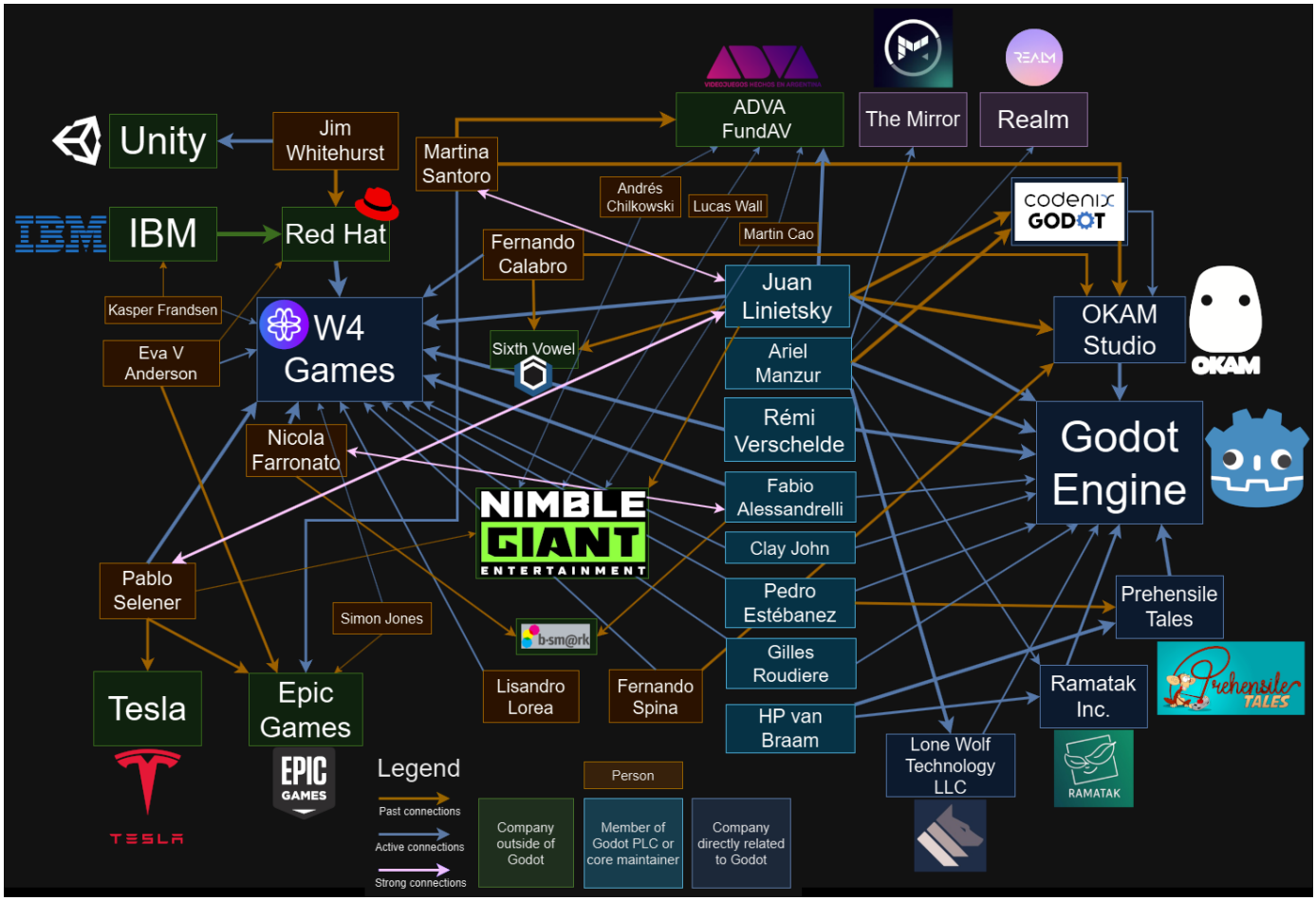
Community-Driven

The Godot project is often touted as a shining example of community-driven development. However, upon closer examination, a web of contradictions emerges, casting doubt on this claim. While some contradictions may be expected in any endeavor, what truly raises eyebrows is the leadership's failure to acknowledge or resolve these issues, leaving a trail of hypocrisy.

This issue can be encapsulated in this short satirical piece which exposes Godot's so-called "ecosystem" as a collective business of its leadership:

> Juan @reduzio Linietsky: "The idea is that Godot shouldn't just be an open-source project."
>
> God's away on business! 🤣 (Sound on!)#WaitingForBlueRobot #Godoverse #IndieGameDev #IndieDev #GameDev #FOSS #OpenSource #Scam #Grift #Ramatak #W4Games pic.twitter.com/JsNdYSfj7K
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) March 24, 2024

Contradictions are not inherently problematic as long as they are eventually identified, addressed and resolved. However, when the leadership of Godot fails to acknowledge or resolve these contradictions, it becomes a matter of their hypocritical behavior.



Godot Engine's public relations and connections.

By the time of writing this book, Godot's development vision and philosophy remain undefined. In truly community-led open-source projects, a clear development vision and philosophy (or principles) form the bedrock of their success. It is very likely that Godot's leaders deliberately neglect this aspect, see Waiting for Philosophy.

Another essential aspect of successful community-led projects is a strong sense of responsibility, especially concerning donations. However, in the case of Godot, there are concerns about the commitment of its leadership, including paid core contributors, to this crucial attribute.

Community-led projects thrive on the principle of bottom-up development, where community leaders are trusted to guide the process. Yet, Godot takes a different approach by heavily relying on trust, but often in a selective or autocratic manner—a top-down style that contradicts the essence of community-led efforts.

Curiously, when this discrepancy is brought up for discussion within the Godot community, they suddenly adopt an opposing stance to the supposed bottom-up principle or horizontal structure, as promoted by the lead developer, and express a preference for a more hierarchical, top-down approach. To demonstrate this discrepancy, Juan Linietsky

disingenuously asserts, *emphasis mine*[1]:

> ***Godot has a very horizontal community and contributor model.*** *We strive to discuss everything publicly and move forward based on agreement. Both the newcomers and the most experienced contributors will discuss and explain their ideas, make sure everyone else is in the same page, and try to reach agreement with others.*

The "horizontal" part suggests that there's a relatively equal distribution of *responsibilities* and *decision-making* among the community members and contributors, as opposed to a hierarchical or top-down approach. This implies that *a lot* of people have a say and have power over the decision making process.

In contrast, the official maintainer of Godot, Yuri Sizov, trusted by Godot's leadership in everything he says and does to represent Godot to the public, claims the following, *emphasis mine*[2]:

> *Definition of Godot as "community-driven" doesn't put community in charge. It puts community interests in front, but it doesn't mean that there is democracy or design-by-committee process. In fact,* ***I am strictly against it and in favor of opinionated decision making process with few decision makers***.

In this case, the "opinionated decision making process" and "few decision makers" leads to hierarchical, top-down approach. This clearly goes in contradiction to Juan's "horizontal" model often touted to the *users* of Godot.

Many members of the Godot community do not pay much attention to the claims and promises made by the lead developer. However, if you conduct your own research, you will find numerous contradictions within Godot that the leadership either ignores or refuses to address, which is the epitome of hypocrisy. Remember that words are not equivalent to actions; just because Juan claims he doesn't decide anything doesn't mean it is true.

So, the question arises: which community-led approach does Godot truly follow? The governance page seems to provide an answer[3], but upon closer inspection, a curious twist emerges. The project conveniently redefines "community-driven" as "community-minded," subtly substituting distinct concepts, akin to a "fine print" maneuver found in contracts.

Godot insists that feature development isn't prioritized by any corporate entity. However, grants from different companies do influence Godot's development in a specific direction, as well as financial donations from individuals, like Patreon contributions, where only Patrons

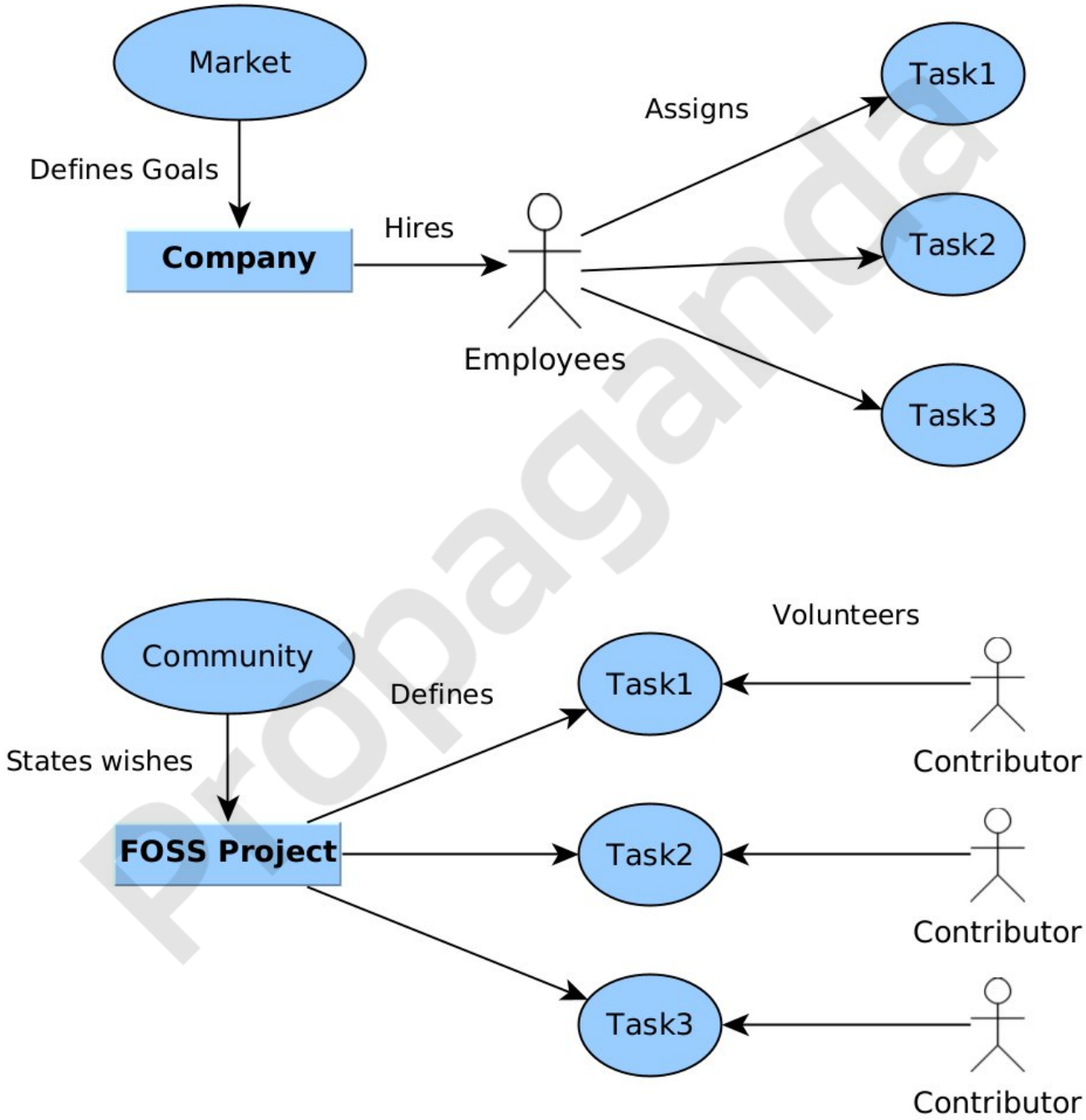
hold voting power in the end, not the broader community, see challenged claims in the Democracy chapter.

Without further ado, let's dive into the inner workings of the renowned open-source happenstance, Godot—a project supposedly driven by the community. Inspired by the enigmatic character from "Waiting for Godot," the project's name holds a deeper significance, urging us to look beyond surface claims and uncover the reality behind the scenes.

Companies vs FOSS

Someone may ask: "What's wrong with that?" A valid question indeed! You may wonder, "What's wrong with the Godot project's leadership presenting themselves as community-driven while exhibiting signs of a commercial company?" Well, the issue lies in the apparent dissonance between these two aspects which must be resolved.

Lead developer of Godot once shown his understanding of open-source governance model in relation to commercial companies by drawing the following diagram[4].



There are several issues with Juan's diagram. First, I'd like to clarify that even if Juan portraits the bottom model representing governance of all FOSS projects, in reality that's just Godot's model, not FOSS in general.

- Notice how **Employees** *assign* tasks. However, we don't see any equivalent action described in Godot's "FOSS" model. **Contributors** don't necessarily *choose* tasks here. They might be *asking for permission* from Godot's leadership to work on tasks, or something completely different.
- **Market** doesn't define *goals*. **Market** creates a chain of supply and demand. *Goals* are

defined by the **Company** to meet a particular set of demand. This is what exactly so-called FOSS model represents when **Community** *states wishes*. Even then, ironically, Godot's model doesn't contain *goals*, which proves that Godot has no defined purpose. The problem is that Godot's leadership has not defined those goals publicly in the first place, so they start to wrongly assume *what* community really wants, and speak *for* community.

- Model shows *one* visual unit of **Employees** and *several* **Contributors**. This is how Juan introduces a visual bias into his model, as if companies have less people working on a project. For instance, here's what Juan says, when comparing Godot to O3DE in terms of number of people working on a project[5]:

> *I don't think that's the case, there is near 2k people working on Godot and it's one of the largest projects on GitHub (and growing). Not even Amazon with a hundred paid employees working on O3DE is close, mostly IMO due to the high engineer compatibility of projects so organic.*

While an open-source project may have "thousands of contributors," in reality most of them are one-time *committers*. What really matters in this context is the number of *active* developers incorporating changes to a project, and not the total number of them, so Juan's statement is extremely manipulative. Additionally, the number of Godot contributors is closely correlated with overzealousness of its community, as you'll discover in subsequent chapters.

Moreover, some people may harbor the mistaken belief that if development appears stagnant, the project is devoid of a future or something similar. In reality, when there are seemingly no changes, this may signal the project's stability and robustness—qualities that are frequently paramount for users of software anticipating sustained, reliable functionality.

If we talk about decision-making process, it is still Godot's leadership that determines what's being worked on, *not* contributors. There's absolutely no do-ocracy in Godot. The illusion portrayed as the freedom of choice comes from the fact that contributors choose from tasks already defined by Godot's leadership. They cannot choose to work on anything not chosen by leadership, even if majority of community expresses a demand for something. Of course, contributors are free to start to work on something which wasn't decided by Godot's leadership beforehand, but doing so would be relying on *luck* for the most part.

Even if someone creates a pull request for a popularly requested feature, it may not be merged, and not necessarily because of quality, but due to other reasons, like the burden it imposes on maintenance. However, Godot's leadership could delegate maintenance of a

feature to someone who created it (the *do-er*), yet they don't delegate such privileges for the most part. And maintenance equals to decision-making power. In truly community-driven projects, we'd see several people managing and merging pull requests, but even after a decade since Godot is open-sourced, we don't witness this becoming a common practice within Godot.

If we address all the issues in Juan's model scrutinized above, it will bear a striking resemblance to a typical company structure. The sole significant distinction lies in the fact that contributors who dedicate their efforts to a company named "Godot" do so voluntarily, without receiving monetary compensation. The number of active individuals considered as "employees" of Godot remains mostly limited to a few people. As frustration accumulates, attentive contributors may observe that the donations are used to increase the salaries of the leadership and a close circle of trusted members, rather than expanding the workforce by hiring experienced game engine developers, as you'll find out in Recruiting. Consequently, the workload remains overwhelming and unresolved.

Syndicate

Look at what Juan Linietsky, the lead developer of volunteer-developed Godot, has been telling Godot community all this time[6]:

- *There is **no company behind it** [emphasis added]. It's owned by every single of its contributors.*
- *We don't compete with Unity, Unreal, etc. In fact they are free to use any of our code.*
- *We make it because innovating is fun, and out of love for those making games.*

For your information, Godot's leadership co-founded several **commercial** companies around Godot in official capacity and in the same person, including but not limited to:

Company	Co-Founders	Services	Notes
W4 Games	Juan Linietsky, Rémi Verschelde	Online Multiplayer, Console Porting	*W4 is often referred to in the third person by Godot PLC, as if it were a fully independent company.*

Company	Co-Founders	Services	Notes
Lone Wolf Technology	Ariel Manzur	Console Porting	*Promoted at Godot's official documentation.*
Prehensile Tales	Hein-Pieter van Braam	Various Services, Consulting	*Godot Editor executables for Windows are code-signed by this company.*
Ramatak Inc.	Ariel Manzur, Hein-Pieter van Braam	Development and Monetization of Mobile Games	*Ramatak was mentioned by Emilio Coppola, nominally Executive Director of Godot Foundation, and Juan Linietsky at GDC 2023.*[7]

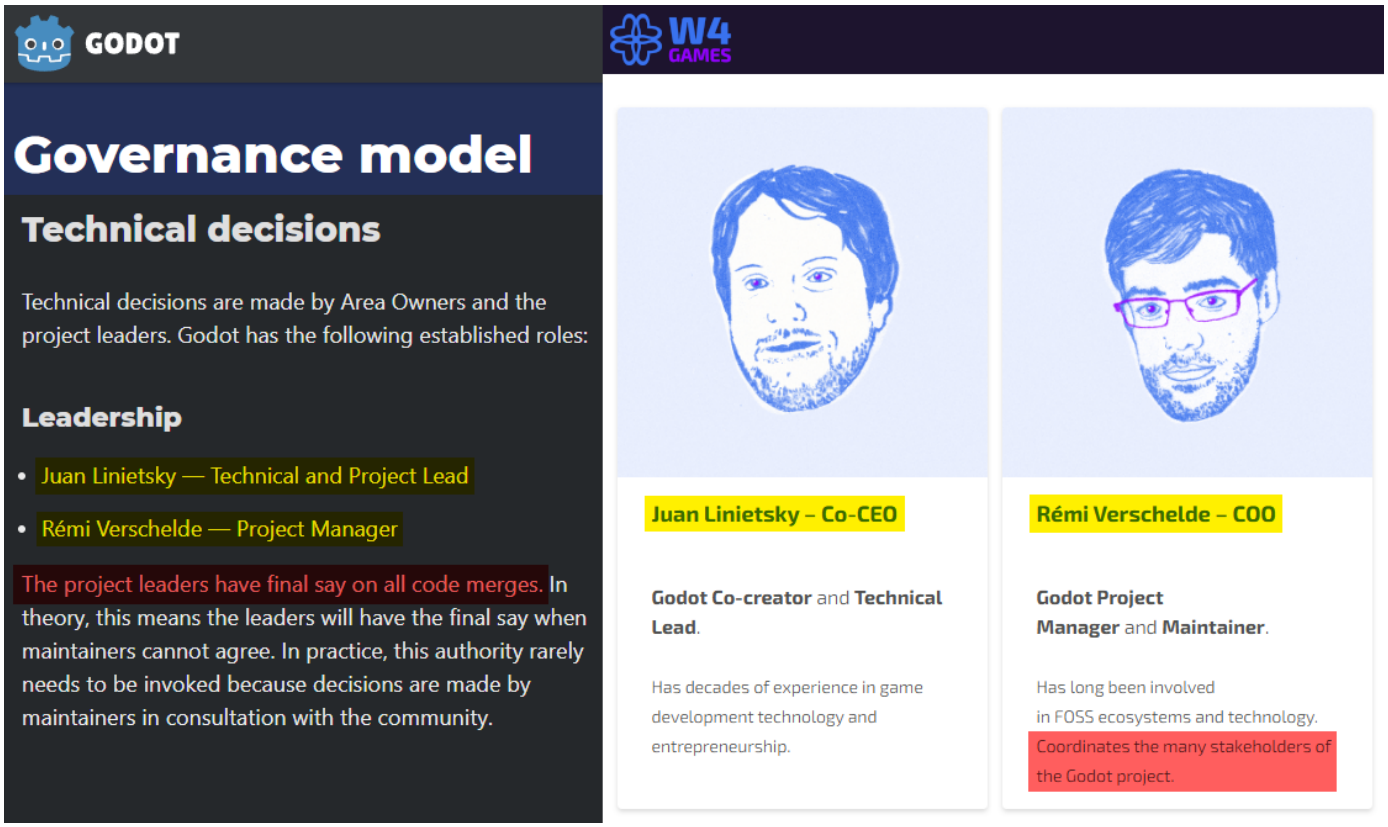
All of them were co-founded by members of Godot PLC, forming a syndicate of companies. Collectively, they created a conflict of interest in relation to **Software Freedom Conservancy** and **Godot Foundation**.

Let that sink in for a moment. On Steam discussions, Rémi Verschelde, the project manager of Godot, previously asserted that there's no business model behind Godot[8], *emphasis mine*:

> ***There's no business model behind Godot**, it's free and open source forever, simply because that's what we want to make. **There is no company behind the project**, only individuals with a common vision (and a US charity handling our legal and fiscal matters).*
>
> *As others mentioned, we do have a crowdfunding campaign on Patreon (+ some direct donations via PayPal) which bring us enough funds to hire 2.5 of our contributors full time to speed up the project development (I'm one of those since a few months).*
>
> *Eventually **what we want is for thirdparties to develop business models around the engine** which will benefit the community as a whole. There are already some Godot contributors who have their own company and do work for hire or paid support for studios.*

In light of the above information, the statements "there is no business model behind Godot" and "there is no company behind the project" turned out to be big lies. They assured the community that this would never happen because, ironically, it's "what they want to make."

The so-called "third-party" companies around Godot are not third-party at all when the ultimate technical decision-makers, Juan and Rémi, remain at the helm of both the supposedly non-profit Godot and the for-profit companies mentioned above.



While there are certainly third-party companies around Godot, this does not eliminate the problem of the conflict of interest that Juan, Rémi, and other members of Godot PLC have

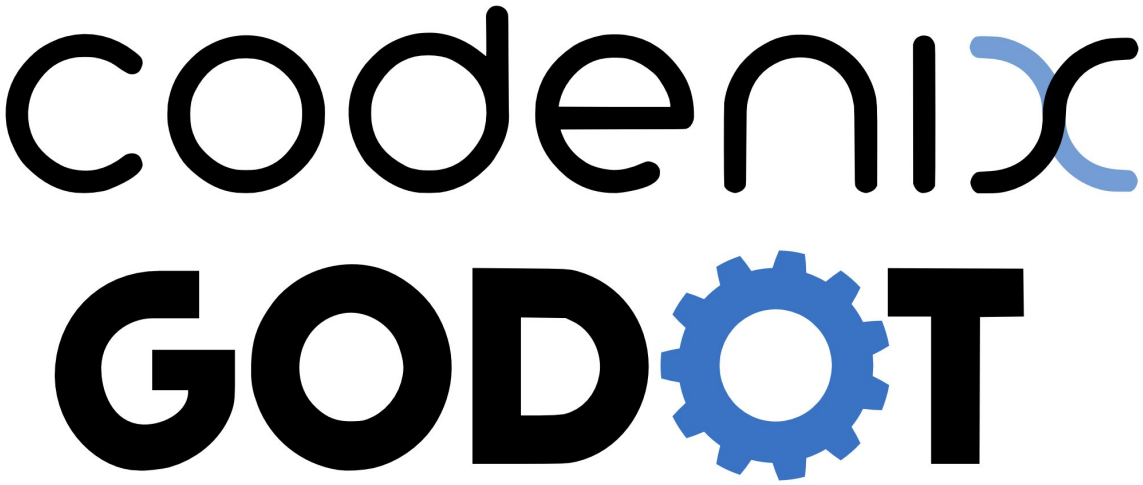
regarding Godot and the companies they co-founded.

This situation exposes a contradiction in a non-profit initiative: there's ambiguity surrounding the participation of project leaders in for-profit ventures, which directly contradicts the project's ethos of transparency. Because of its non-profit nature, the project cannot openly endorse or promote those commercial entities. However, the undisclosed association of project leaders with for-profit companies created specifically for Godot presents a dilemma. Disclosing these ties would violate the non-profit mission, but hiding them perpetuates the lack of transparency. This loop can be broken in these ways:

- Declare that Godot is a hybrid organization, where non-profit and for-profit entities are intertwined, while ensuring full disclosure of any apparent conflict of interest.
- Significantly expand the number of people allowed to make the final decision on all code merges, beyond just two people. This would minimize the risk of the project being held hostage by stakeholders.
- Members of the Godot PLC involved in for-profit ventures must leave the Godot Project and appoint new leadership to ensure community-driven development at all times.

However, even these solutions cannot repair the betrayal of trust Godot contributors would experience after these declarations and changes. Godot's popularity depends on the idea that it is a pure open-source project. It would not be able to attract those "thousands of contributors" if Godot were declared a hybrid organization from the start.

Remember that in the past, Godot was a closed-source, in-house game engine for quite some time. Before they open-sourced the engine, the co-founders of Godot, namely Juan Linietsky and Ariel Manzur, co-founded Codenix[9], a game development consulting company. Through Codenix, they licensed Godot to third-party companies in Argentina, which is now defunct.



At the time when Juan Linietsky found himself bombarded with criticism over his and other PLC members' business ventures tied to Godot, he craftily employed a little spin. He cunningly referred to these companies as "satellite" communities, allegedly separate from the official Godot project, which posed him as a highly hypocritical individual, once again[10]:

> *It's just amazing for me to see that Godot has grown so massively, even this past year, that not just the official communities grew huge, but there is now an infinity of smaller satellite communities unattached from the official one..*

Juan made those statements in an effort to distract from the scandal. He urged people to donate to Godot as if it was on the brink of bankruptcy, even though they had received $8.5 million in funding from venture investors at W4 Games, *emphasis mine*[11]:

> *The project tapped into reserves to do more hires in order to complete Godot 4.0 (and now 4.1). This was successful and you can now enjoy it! But **money is running out** as the funding went **cashflow negative**..*

Whereas W4 Games, a for-profit company co-founded and operated by Juan himself, literally **pledged** to support Godot's project financially:

> *Additionally, **W4 Games pledges to support Godot financially** with no-strings-attached donations to the project.*

Imagine being an investor in W4 Games. While nobody expects W4 Games to donate all their money to Godot, as an investor, you would expect W4 Games to stand behind their publicly declared pledge and support Godot if there were a dire need to secure the jobs of existing contributors who are hired by Godot to work on the engine either part-time or full-time. This is particularly crucial since the business of W4 Games depends entirely on the success and adoption of Godot. Hence, it would not make sense to portray the situation as if Godot is in dire financial straits and entirely dependent on donations from independent developers.

A respectable community member of Godot accurately outlined this situation:

> *To say W4 Games "is unrelated to the Godot project" is pretty disingenuous. That investment was made on the popularity of Godot, not you and Remi personally. And now to say "money is running out" and "development pace will probably go down" when you have $8.5M is a total scam.*

It's worth clarifying that one of the definitions of the word *scam* is a *dishonest scheme*. You can read this article as an example to understand in what other contexts something could be considered a scam, which ironically uses the "Waiting for Godot" analogy to convey the message. 😁

The video below highlights Godot's commercial agenda, revealing that it is more expensive for indie game developers to use it on consoles compared to Unity:

Plenty of other people outside the Godot community share these insights:

> I am sharing a lot about #Godot/#W4Games recently, following the reveal of the console port pricing, but it is just so ostentatious conflict of interest example and the leader being the main one here sharing hate/arrogance over other engines is...
>
> 🧵 Thread#TruthAboutGodot pic.twitter.com/U7xjacYwGK
>
> — Pawel (@pawel_developer) December 13, 2023

> I have been saying this for far too long now. Godot is a fantastic engine but W4 games has tainted its reputation. Here is Juan Linietsky claiming that W4 Games is unrelated to Godot when he was asking users for money for the Godot foundation after having raises $8.5M with W4. https://t.co/CpwrAjpjSC pic.twitter.com/vqawO1ZqwT
>
> — Carson K. (@CarsonKompon) December 13, 2023

At present, the Godot Foundation does not offer free console support. It is a common misconception among Godot users that the Foundation is unable to provide such support due to being Open Source. But other open-source and source-available projects, such as LibSDL, Haxe, Monogame, Cocos or Defold, do offer console support free of charge. This is because other projects view console ports as a means of attracting more developers. For example, the following is a statement from a board member of Defold:

> W4 is selling console access as a part of their business. In Defold we see console ports

> as a way to attract more developers. These developers will hopefully release successful games. Which will attract more users. Which will attract corporate partners. Which will generate money.
>
> — Björn Ritzl (@bjornritzl) December 12, 2023

To further demonstrate that providing free console support is possible, here is what the Godot community provides, unofficially:

Prominent game developers have identified the same issue, and you can find a compelling blog post that thoroughly refutes Godot's claims about not supporting free consoles:

> In response to the @godotengine foundation announce about no port (pointing at W4 option) I wanted to say some things. Also I plan to share my porting work when I can.
> Video: https://t.co/snmuQ2tyQT
> Article: https://t.co/a72WFczEjW
>
> — Claire Blackshaw (@EvilKimau) September 4, 2024

Some excerpts from the blog:

> [...] *The smart short term decision is to use Unity or Unreal which pound for pound are more capable engines able to ship higher quality games today. I'm investing in tech, my studio and frankly trying to build a house instead of renting one. So no,* ***I don't want W4 games as my landlord****.*

Notice how Unity and Unreal, despite delivering higher-quality products, are portrayed as only valuable in the short term. This is because Godot supporters are invested in the so-called “bright future” of Godot. One shouldn’t rely on wishy-washy hopes. Godot hasn’t arrived and never will! 😉

> *There is a big issue that **the key maintainers and founders of the project are also the key figures at W4** and that is always going to be a hard needle to thread.* [...]

> *Again everything I’ve seen is people are mostly being above board and good but I don’t want to rent, I want to buy. So **engaging in commercial agreements with W4 games is not in my studio’s best interest** at this time.*

It’s worth noting that comments like “There are no villains in this story. I think everyone is awesome” reveal a deep connection to Godot’s ideology. This doesn’t diminish the author’s expertise on the topic of consoles; instead, it underscores how even seasoned industry professionals can be swayed by their personal investment in a project, leading to blind devotion despite recognizing fundamental issues.

With this, I just want to illustrate that even those who are in “love” with Godot can’t ignore these issues forever, and the more invested they are, the more they can experience what might be described as Stockholm syndrome. In the context of scams, Stockholm syndrome can occur when victims become emotionally attached to their scammer or the fraudulent scheme, even while recognizing the harm being done. This psychological attachment can lead them to defend or justify the scammer’s actions, despite the obvious exploitation. We’ll delve into this sociopsychological dynamic in the following chapters, as this is one of the key reasons why many still remain silent about their bad experiences with Godot.

Regardless, Juan’s disingenuousness regarding W4 Games’ involvement with Godot is extremely difficult to deny here. W4 Games investments come from investors that are familiar with FOSS or were in the past prominent figures. Godot’s leadership carefully cherry-picked venture investors who have expertise in Open Source, those who Juan can *fully trust* for his scheme (see Trustocracy) and/or knows how to allure them, since Open Source investors are more susceptible to manipulation due to its ideological nature. Once the foundation is ready, they might start to participate in a larger scheme to lure in *unsuspecting* investors outside of Open Source arena.

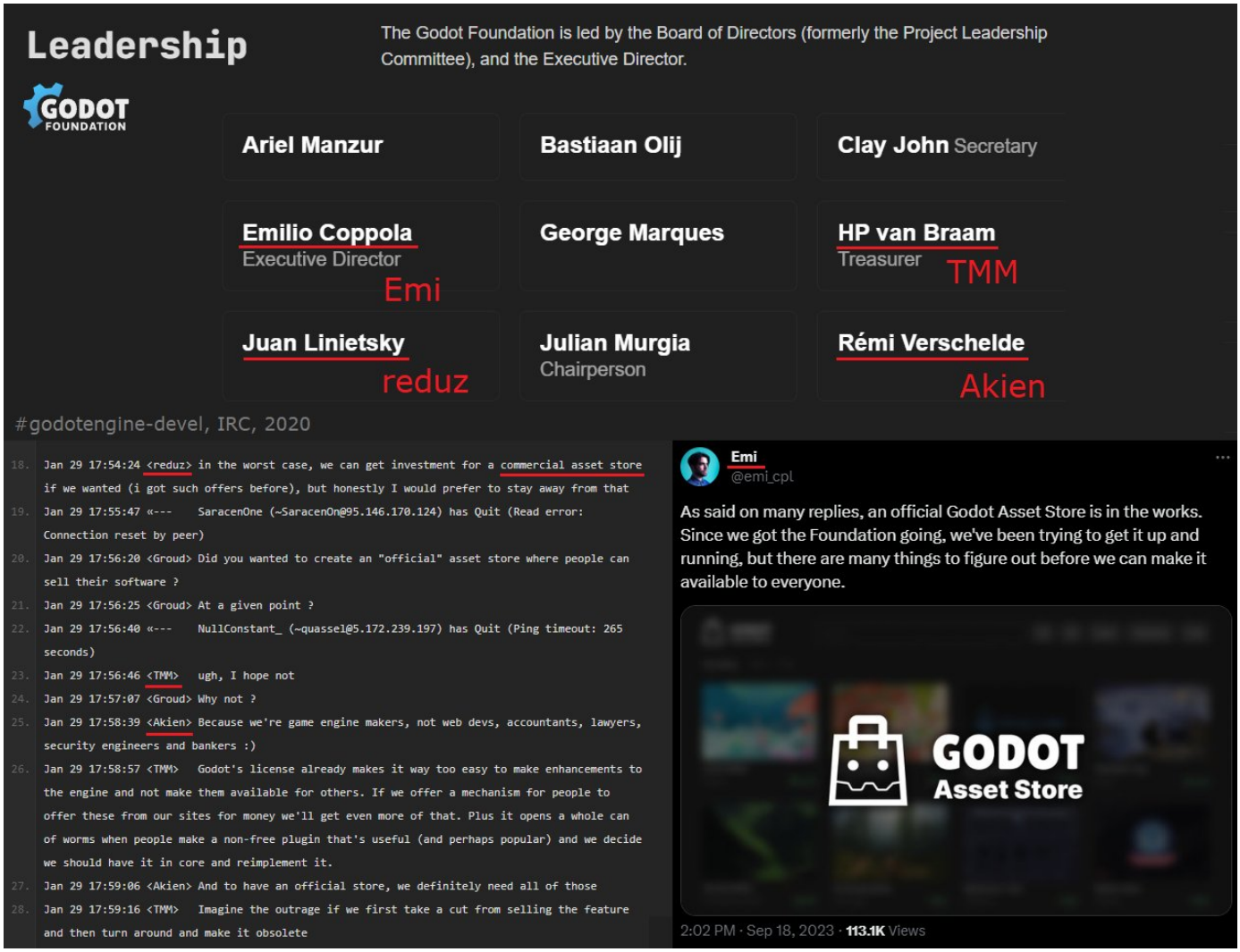
Joseph Jacks, a prominent investor of W4 Games, once shared a series of enigmatic statements that seem to encapsulate his vision on embracing Open Source technology. Mind you, these statements hold a touch of irony, inviting us to delve deeper into his

underlying motivation for investing in companies like W4 Games, *emphasis mine*[12]:

> *There's **no money in open source**.*
>
> *There's only niche money in open source.*
>
> *There's definitely money in open source.*
>
> *Most of the money is in open source.*
>
> *The **only money is in open source**.*
>
> *There is **no money**, there's **only open source**.*

When you interpret those statements as stages, the last statement suggests this kind of future. 🙃

Godot has also made several inconsistent statements about the creation of the commercial Asset Store, which contradicts Godot's non-profit mission. Initially, they claimed they couldn't afford it because of Godot's open-source nature. However, over time, they changed their stance, likely to cater to the growing number of Unity users who want an equivalent Asset Store in Godot. Here's a public conversation between members of Godot PLC, retrieved from IRC logs[13]:



Juan Linietsky and others also discussed the possibility of facing a lawsuit in the context of Software Freedom Conservancy and creating a commercial Asset Store:

```
Jan 29 18:18:21 <reduz> TMM: In general, suing open source
projects is viewed in an extremely negative view in the
software industry
Jan 29 18:18:44 <reduz> TMM: in fact OIN protects against
most cases of that
Jan 29 18:18:57 <reduz> TMM: and Conservancy also protects
our project from that
Jan 29 18:19:05 <reduz> TMM: so I wouldnt worry
Jan 29 18:19:41 <reduz> TMM: and the company suing gets a
huge amount of negative press, so it doesnt happen
Jan 29 18:20:16 <TMM>   Not if it's some Joe Rando who has a
level editor plugin or something
Jan 29 18:21:35 <TMM>   I'd be super wary about turning
Godot into 'open core' at any rate.
Jan 29 18:25:47 <Groud> While I understand, the reason why
to not do it, I wonder what are the possible solution to
make sustainable the project (I mean, on the long run).
Jan 29 18:26:04 <reduz> TMM: joe rando will not have the
economical means to do this, and conservancy will take care
of the lawsuit for us
```

Contrary to what HP van Braam (TMM) said in 2020, Godot is becoming more akin to **Open Core** in its structure nowadays, given Godot's for-profit companies such as W4 Games and Ramatak. This is especially evident when Juan presents Godot to investors as a so-called Open Ecosystem.

According to **Software Freedom Conservancy**'s Conflict of Interest Policy, the PLC Person, in our case these are Juan and other members of Godot PLC, must disclose their conlict of interest[14]:

> ***Disclosure of Conflict When Present.*** *Prior to any PLC or PLC sub-committee action on a matter or transaction involving a conflict of interest, a PLC Person having a conflict of interest and who is in attendance at the meeting shall disclose all facts material to the conflict. Such disclosure shall be reflected in the minutes of the meeting.*

> *A PLC Person (or their family member) is engaged in a substantial capacity or has a material financial interest in a for-profit enterprise that competes with their Project.*

None of the existing members of Godot PLC have disclosed any such connections. They eventually left the **Software Freedom Conservancy**[15] in favor of the **Godot Foundation**. The individuals who make up the Godot Foundation also represent commercial companies. The problem is the lack of a controlling body. For more than a year, Godot has not submitted annual reports or financial statements, even though they are required to do so from day one as a non-profit organization:

> Waiting for Godot Foundation financial reports? 🙃 pic.twitter.com/yVaJ14EUrk
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) February 14, 2023

Since Godot is declared as a non-profit, it would be illegal for Godot's leadership to control the funding decisions of both their for-profit companies and Godot as a non-profit project. Commercial shenanigans that exploit the allocation of grants among those companies, such as **Prehensile Tales**, instead of being handled by SFC directly, raise numerous questions[16]. While such maneuvers might be deemed acceptable for a commercial enterprise, the actions of Godot strongly suggest an exploitation of their non-profit status to gain unwarranted economic advantages.

Software Freedom Conservancy ignored all requests on this topic by the author of this book. Taking into account other ignored reports to SFC as you'll read further, this means that both Godot and SFC are likely interested in this kind of scheme, despite apparent conflict of interest, especially when they use the same "graduation" rhetoric[17]. They *both* label it as a "graduation." What this newspeak really means is that Godot betrayed the trust of existing Godot followers at that time. See what lead developer of Godot said years ago about Godot, before they left SFC[18]:

> *Given many FOSS projects recently went either **proprietary or hybrid** [emphasis mine] (and we which we will soon forget in favor of their forks), I want to remind all that Godot is owned by a not-for-profit, so this can't ever happen.*
>
> *Every bit of Godot is owned by the person that contributed it.*
>
> *(Just to clarify, Conservancy, the not-for profit, owns the trademarks and is the fiscal sponsor, contributors own the code they contributed under MIT license)*

Examining these facts, it becomes increasingly challenging to arrive at any interpretation that would deny Godot's status as a hybrid organization now. The emergence of multiple commercial enterprises operating under the umbrella of Godot, overseen by the very leadership of the project, and the subsequent departure of Godot from SFC, serve as

unmistakable evidence that Juan's earlier assurances are, at best, fallacious.

As of the time of writing this book, no legal measures have been taken against the leadership of Godot. Furthermore, the author of this book no longer perceives ownership over Godot, as the right to contribute to its development has been revoked—a circumstance shared by a subset of other contributors, as you will discover in the concluding chapters of this work.

For some people, perhaps all of the above wouldn't be so much of a problem and could be tolerated or mitigated, but Godot's governance is certainly not the only issue at play here. Inefficient project management and the lack of Godot's Development Philosophy are factors that contributed to Godot's overall strategic issues as a project. It makes no sense to start engaging in commercial endeavors and trying to pitch business idea to investors before stabilizing the very product that you depend on. To quote an ex-moderator of Godot's Discord server[19]:

> *Juan's "Godot philosophy" was killing Godot... and definitely killing my bright eyed and bushy tailed vision of where Godot could go. So, I got again more vocal about Juan's constant "It's my engine, I'll do it my way" take while claiming it is "community driven".*

> 28/ Juan's "Godot philosophy" was killing Godot... and definitely killing my bright eyed and bushy tailed vision of where Godot could go. So, I got again more vocal about Juan's constant "It's my engine, I'll do it my way" take while claiming it is "community dirven".
>
> — LillyByte (@LillyByteGames) March 28, 2022

The saddest part of all this is that Godot fans either don't care about these issues or even say that the developers don't have to *listen* to their community:

> Mark the words:
> A community-driven project which Godot claims to be "don't have to listen to their community, if they don't want to."
>
> They don't have to, yes, I totally agree, no way to force it, except only leaving it.#Godot #TruthAboutGodot https://t.co/2qAHieKzLe
>
> — Paweł SOON GIC, Poznań, 20-24 October (@pawel_developer) December 15, 2023

The leadership of Godot has ingrained the idea that these dishonest practices are

acceptable, while also *falsely* believing that Godot has quality features that, in their warped sense of ethics, justify those practices:

> ya know what why are you getting so ryled up about this. even if they are being frauds they are delivering on their feature promisis including cons, important venues, and demos. none of these are cheep.
>
> — KaziiTheAvali (@KaziiTheAvali) September 10, 2024

Satellites

The previous section referred to W4 Games, Lone Wolf Technology, Prehensile Tales, Ramatak as a syndicate, because of direct involvement of Godot PLC. This section, in contrast, lists some notable Godot satellites, where some Godot PLC members and notable Godot contributors/maintainers are working as employees as of the time of writing.

The Mirror

The potential partnership between **The Mirror**, the potential **Roblox**-like game development platform, and **W4 Games**, an **Irish** start-up aiming to “revolutionize” the online gaming market founded by Juan and others, further highlights the interconnectedness within the industry. W4 Games, with its cloud platform based on open-source technology, intends to “democratize” the online multiplayer market for independent developers, as they proclaim at their LinkedIn page. Should The Mirror flourish, it could become a primary source of income for W4 Games.

The convergence of Godot contributors working on both The Mirror and W4 Games, along with the personal connection between the managing director of W4 Games, Pablo Selener (a former employee at **Epic Games**), and Juan Linietsky[20], raises numerous questions about the true nature of their collaboration.

Nevertheless, it is crucial to approach this information with a balanced perspective. The

individuals involved with The Mirror[21], such as **Ariel Manzur**, co-founder of Godot, and **HP van Braam**, a Godot maintainer, carry substantial weight within the Godot community. HP controls the toolchain required to build Godot for all supported platforms (including Apple platforms), and Godot binaries for Windows are code-signed via his commercial consultancy company called **Prehensile Tales B.V**.

While other contributors like **Gordon MacPherson** (from IMVU) and **Aaron Franke** have made significant contributions to Godot Engine, such as FBX/GLTF file format support for 3D models, their involvement in The Mirror adds an intriguing dimension to the situation. For example, **Aaron Franke**, who is now recognized as an official member of Godot, previously criticized Godot for its "community-driven" label because it's confusing to other contributors[22]:

> *The README should reflect Godot's policy, not go against it. Doing otherwise is confusing for contributors. Stating Godot is community-driven is misleading at best, so it shouldn't be in the README. Most people will read "community" as "the whole community of users and developers", not "the community of core developers".*

Even when Aaron explicitly stated beforehand that "This was not done in bad faith," the leadership of Godot did not listen and refused to remove the "community-driven" label. **HP van Braam** rejected the pull request (PR), closed it, and locked the discussion, contradicting the principle of *open discussion* declared by the Godot project, and labeled Aaron's pull request as a grievance:

> *Discussions like this don't happen on a PR and never have. Opening a PR like is clearly in bad faith. I'm sorry you feel frustrated about the process but this is not the way to air your grievances.*

If Godot's leadership dismisses the input of *contributors* in such a manner, what can we expect in terms of their response to *user* input? Under normal circumstances, the developments at for-profit projects such as The Mirror could signify positive opportunities and collaborations. However, realistically, they present red flags that cannot be ignored[23]. The needs of such companies might dictate the future course of Godot (see Simplicity), potentially overshadowing the needs of the larger community that utilizes Godot.

The Mirror and Godot Engine partnership teaser

Whatever the case may be, Godot's leadership is definitely interested in attracting even more followers to use Godot, even if it means gradually transitioning from platforms that are "easier to use" like Roblox to considering the use of Godot. This is particularly relevant since The Mirror is built upon Godot Engine itself. The leadership's interest in expanding the user base is evident.

Since The Mirror is potentially the next **Roblox**, you should also be aware of potential exploitation of young game developers. In an investigation done by **People Make Games** titled "Investigation: How Roblox Is Exploiting Young Game Developers"[24], they ask whether the kids making the vast majority of its content are being taken advantage of. An interesting fact: before a series of investigations by PMG, **Roblox Corporation** has been valued up to $45 billion, and now being valued at more than $27 billion by the time of writing this book:

Juan Linietsky might announce that he plans to resign from his position in Godot Engine, either by focusing on a closed-source or source-available fork or continuing his involvement through W4 Games and other companies, which would act as a financial resource and service provider to projects such as The Mirror. It is not unusual for a project like Godot to have commercial forks, especially due to its lack of out-of-the-box Customization. In fact, the first commercial distribution by Godot's leadership already exist, and it's called Ramatak Mobile Studio. Such commercial forks could entice contributors with investment opportunities, leading to a split within the community and possibly resulting in Godot stagnating as an *open-source* project specifically.

Conclusion

Given the absence of a clear development vision and philosophy from Godot itself, in

essence, the project's direction largely hinges on financial support from various sponsors, regardless of their nature, even extending to the gambling industry. It becomes evident that companies wield significant influence over Godot's development direction, despite claims of community-led decision-making. Ethical considerations aside, the pivotal role of corporations cannot be ignored. The emphasis is on the decisive role that companies play in determining Godot's development direction, regardless of the nature of those companies.

The discrepancy arises from the surface-level impression of Godot being purportedly community-driven, while exhibiting signs of a typical company structure. Historically, they have always operated as a "garage" game development and consulting company. They leveraged and benefited from Open Source due to the economic circumstances of Godot's founders in the past. However, they lack the expertise to effectively manage community-led open-source projects. Godot gained traction primarily because of Rémi, an overzealous Open Source advocate who took the initiative and invested a significant amount of effort into promoting Godot. Additionally, potential grants offered by Latin American governments to promote Open Source technologies worldwide, including Godot, also played a role in its growth.

Godot's leadership brainwashed their initial contributors, who were volunteers, that they would wield significant decision-making power in the early stages of Godot's development. Without that initial injection of free labor, Godot wouldn't have gained the level of traction it has now. Back when Godot was inconspicuous and insignificant, one tactic to lure in contributors was to pledge a project crafted by the community, for the community—an alternative to the "two big engines," as Juan often points out. Juan promised them that they would be among the pioneers in this supposed gold rush, leading the charge in the so-called "inflection point" of challenging the industry giants and promising a supposedly bright future. Their misstep was in overselling to the public, making a bunch of conflicting or questionable statements and moves.

Over time, Godot's leadership insidiously introduced corporate elements into the Godot's so-called worldwide organization while simultaneously assuring their users and contributors that it would never be influenced by commercial entities. They consistently deceive people by claiming their organization has a horizontal structure to lure in even more followers and potential contributors. The actions of Godot's leadership compared to their public statements, therefore, suggest that Godot is operating as an extremely sophisticated grift.

If Godot were to address all its issues and become a polished product, it would lose its raison d'être, as per the underlying meaning behind the name Godot. The essence of Godot lies in its perpetual state of being unfinished. If all its problems were fixed, Juan and his loyalists, presently engaged with W4 Games, would lose the ongoing opportunity to generate income. While this notion may seem bizarre initially, their perspective is rooted in the observation that if people are willing to donate to Godot for an incomplete or even a

broken game engine, they internally reason: “It’s working, so why not continue as is while they contribute to Godot?”

In light of this, if Godot’s current dishonest practices were to go unnoticed, there’s a potential reinforcement of the notion that engaging in unethical and corrupt behaviors is permissible. Essentially, this blurring of ethical boundaries threatens to erode established principles of what is considered morally good, especially to those sharing democracy values. Therefore, it’s crucial for the wider game development community and experts in related industries to exercise caution regarding the lasting impact of Godot’s undue influence.

References

[1] Juan Linietsky about horizontal community and contributor model - Godot subreddit.

[2] Yuri Sizov about governance model of Godot - Godot, GitHub.

[3] Governance model - Godot Engine website.

[4] Companies vs FOSS model - By Juan Linietsky, Twitter.

[5] Juan Linietsky on the number of contributors - By Juan Linietsky, Twitter.

[6] Facts about Godot Engine - By Juan Linietsky, Twitter.

[7] Ramatak Mobile Studio — Godot is the new Unity! - YouTube.

[8] There’s no business model behind Godot - By Rémi Verschelde, Steam.

[9] Codenix website - By Juan Linietsky, Ariel Manzur.

[10] Juan Linietsky about Godot’s satellite companies - Twitter.

[11] Juan Linietsky about Godot’s financial future - Twitter.

[12] There’s no money, there’s only open source - Joseph Jacks, Twitter.

[13] Godot Asset Store discussion - #godotengine-devel, IRC.

[14] Software Freedom Conservancy Conflict of Interest Policy

[15] Godot’s Graduation: Godot moves to a new Foundation - By Juan Linietsky.

[16] Changes to Godot Patreon - Godot Engine, Patreon.

[17] Announcing Godot’s Graduation from SFC! - Software Freedom Conservancy.

[18] Juan Linietsky on Godot’s future as organization - By Juan Linietsky, Twitter.

[19] Juan’s “Godot philosophy” was killing Godot - LillyByteGames, Twitter.

[20] Video Games Around the World - Mark J. P. Wolf (ed.), 3 June 2015.

[21] The Mirror - About Us - The Mirror website.

[22] Remove mention of community-driven and mention feature PRs may not be accepted - Godot, GitHub.

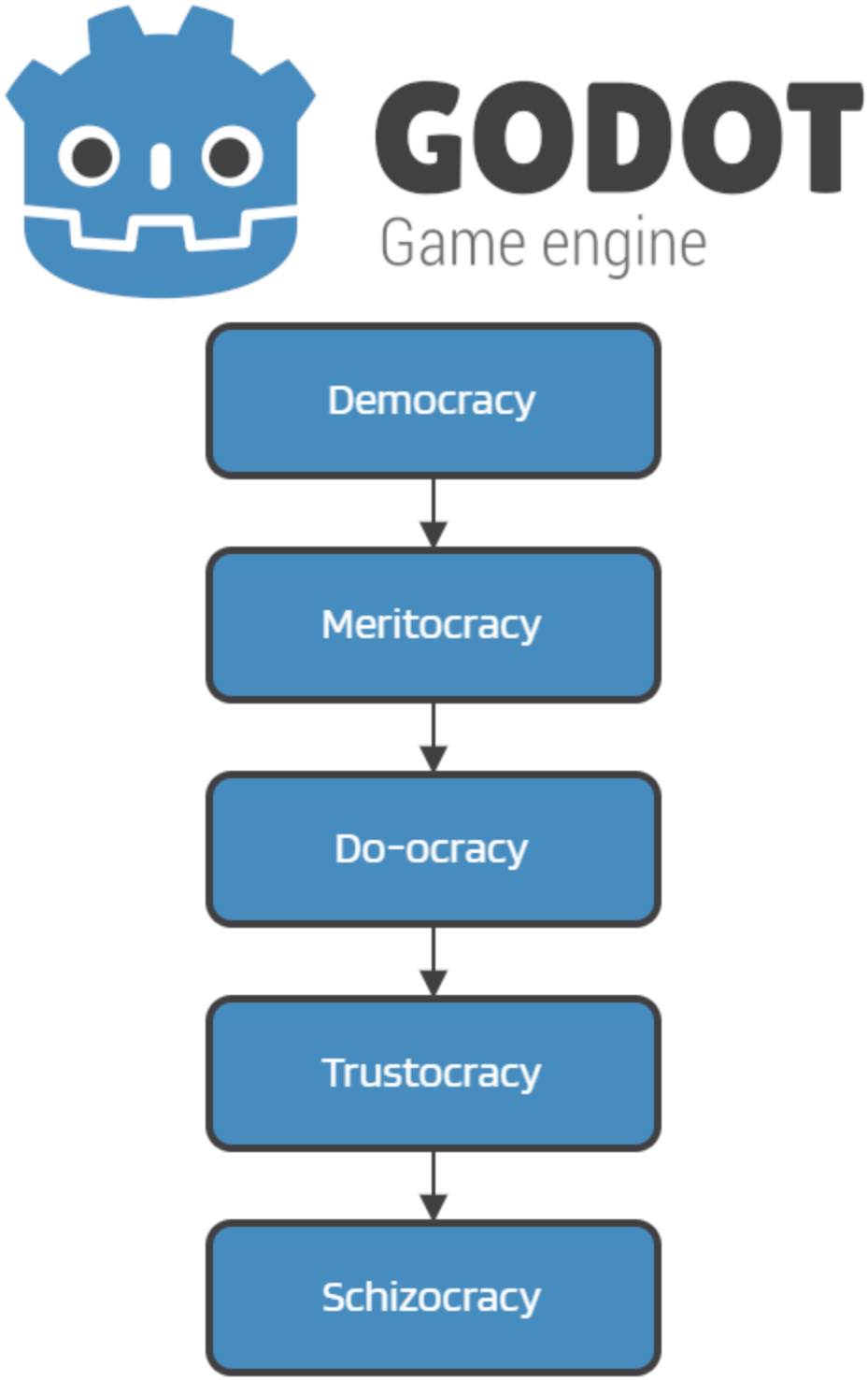
[23] The Mirror – Godot Powered Commercial Game Engine - By GameFromScratch.com, YouTube.

[24] Investigation: How Roblox Is Exploiting Young Game Developers - YouTube, People Make Games.

Schizocracy

Godot is not community-driven, because it acts more like a commercial company which gives opportunity for developers to work on the engine for free, with a mixture of approaches used in both open-source and source-available type of projects, especially taking into account Godot leadership's initiative to found companies that offer commercial Godot services.

Godot's ambiguity asks for their governance model to be labeled as *schizocracy* (the "schizo-" translates from Ancient Greek as "I split"):



If Godot doesn't want this condition to progress, the solutions are quite simple:

- stop giving a false impression that Godot is *democratic* or *meritocratic*;
- do not present Godot as *community-driven*, but rather *community-informed*;

Alternatively:

- declare that Godot is a commercial company that uses community feedback and uses

open-source to accept work from others.

There's no need to hide the fact that Godot leadership wants to make money with Godot, because, as the old Godot's saying goes: "Man's gotta eat". However, for some reason, Godot leadership wants to "clarify" that they don't want to make money again and again. Why is this needed?

Optimistically speaking, Godot is de-facto *community-informed*, not *community-driven*. But realistically speaking, given above contradictions, Godot's governance can be explained in another way, which you'll find out by the end of reading this book.

It's not just semantics, as Godot leadership carelessly responds to this. If they choose to keep telling that community is in charge when it's not, then it will definitely backfire. We're talking about not meeting *expectations* of what constitutes community power over decision making. All in all, the success of this is totally determined by community's trust in Godot leadership. Even then, *their* success is not necessarily *community's* success.

Conclusion

The cornerstone of Godot's existence has been hinging on the notion of it being a "pure" open-source project. Simply put, if you were to mention the word "Godot" to Godot followers, the immediate association in their minds would be "Open Source" first, and only secondly as a game engine. However, when Juan and other Godot PLC members decided to provide commercial services through W4 Games, Prehensile Tales, Lone Wolf Technologies, and Ramatak Inc, all under the same person, they can no longer use the "Open Source" label as an ace card.

Although Godot remains open-source, its value as an open-source project has "diminished" to a level more akin to source-available engines. I put the word "diminished" because the entire notion of a tool being open-source or not is questionable since Open Source is not a magic pill, it has the same number of downsides as commercial software, just manifesting in other aspects. What's important to emphasize here is that Godot's leadership decision has undermined the trust of their followers, regardless of the fundamental beliefs about the perceived benefits and downsides of Open Source.

Interlude: Becoming a Blue Robot

A Definitive Guide

The first thing that you'd need to do in order to become a Blue Robot is to learn Godot Engine, of course. Take some time to learn GDScript, create prototypes, funny projects, and generally experiment with the engine. It should only take you a couple of weeks to master the basics of game development with Godot.

Join Godot community channels. Introduce yourself there. You'll quickly get bombarded with love from Godot moderators and other users there. Tell them that you're new to Godot and say that it's awesome! Post your progress at **#showcase** channels. This way, you're going to let the community know that you're *seriously* interested in using Godot for your projects. Start helping people to solve their programming or design tasks with Godot and recommend Godot to others. It's important to get noticed by Godot developers at this point.

Go to Godot contributors chat and join the **#new-contributors** channel. This channel is created specifically to invite new contributors to let them work on the engine's codebase, and not only. Again, say out loud that "Godot is awesome!" and ask what you can do in order to help Godot. You'll likely get a response from Godot's project manager, Rémi, and he will give you further instructions on how to proceed. Be polite: say "Thanks!" and do exactly what he says.

Fortunately, Godot has enough bugs to be fixed. So at this point, it's a good time to visit Godot's repository. Make sure to star Godot! Create pull requests to fix bugs, or write documentation even if you don't really know how a feature works, you'll figure out in the process! Triage bug reports to get maintainer privileges quicker.

Always use "Godot Improvement Proposals" for suggesting new features in Godot. But do not create proposals for something you want yourself, you don't want to look selfish, right? Remember: you're no longer some pesky user, you're a developer now! Look for what users want instead, and propose to implement existing proposals to satisfy user requests. But please, for the love of Godot, do not propose to change Godot's icon, they hate this!

Your success at this point is not necessarily determined by your level of programming expertise. You can either become an *advisor* or a *core developer*. It's crucially important not to show any signs of so-called "troublemaking attitude" when you speak to Godot developers. Do not speak like an "asshole" there. Godot does not tolerate any kind of criticism. Always say that Godot is the engine you've been waiting for, and that commercial engines suck because they want people to pay royalties or subscription fee. Capitalism is

boo. Donate and be happy!

Do not even think to talk negatively about Godot, anywhere, under any circumstance, period. Even if you think that you have something constructive to say, in most likelihood you'll regret it. Even if you think that it was a joke. Godot core developers may interpret criticism as "insinuations" or "passive-aggressiveness". Therefore, I advice against speaking out in Godot unless it's for the benefit of promoting Godot, of course.

Always show that you share Godot's vision and development philosophy, even if you have absolutely no clue what exactly that vision or philosophy represent in Godot. Due to this, you have to get inspired by what leaders of Godot say, specifically Juan, the lead developer of Godot, and repeat after them.

Do not miss a single opportunity to say nice things about Juan and Rémi for the work that they do in Godot. They really enjoy flattery! Whenever Juan makes a new feature in Godot, say: "Nice!", and don't forget to give out a lot of positive emojis, such as these: 👍 🎉 ❤ 👀. Do not miss Juan's birthdays! Call him "Alpaca!" Very important! Say that you sympathize Rémi for spending so much time merging PRs all day long, and call him "Akimerge!"

Again, being a professional programmer is not necessary in Godot. Just be extremely welcoming to newcomers, spend eight hours a day helping others. At some point you'll either receive moderator or maintainer privileges in Godot for all the hard work!

If you somehow mess up, you may receive angry private messages, either from Juan or Rémi. If this happens, say that you're really sorry and promise not to say or do anything wrong in the future. They will show you Godot's Code of Conduct and you should say: "Yes!"

If you have money to spend, you should consider donating to Godot. You'll become The Boss so that you can pick the next proposal you want to see implemented out of all existing proposals written by mere mortals. Remember: regular thumb-ups do not determine anything. And of course, if you pay lots of money, they can even forgive your "troublemaking attitude". Just look at Yuri and you'll get what I mean.

Speaking of Yuri, I advise to avoid talking to him at all cost. He's a maintainer in Godot, and he's very stubborn, oftentimes goes on lengthy rants to prove you wrong, or say that you shouldn't "spread misinformation about Godot" even if you repeat after the leaders. There exist rumors that Yuri is becoming a true role model for both Juan and Rémi. No wonder, he's literally a Russian Warrior!

I cannot emphasize this enough: even if you notice some core developers talking non-sense, you should not say that they are mistaken. If you really think that you should point them this out, I'd suggest taking these matters on private channels at least. But honestly, you should learn to stop paying attention to their mistakes. Everyone makes mistakes! If they say or do

something wrong, it's only your fault that you call them out. This might be just a misunderstanding or misinterpretation, they definitely do no harm!

Given that you follow everything written above, eventually you should become a maintainer of Godot's codebase and/or moderator of Godot communities. Generally speaking, you must be able not only maintain the codebase, but also the behavior of people in Godot community. For instance, if you're a moderator, you should ban at least someone. You'll get an opportunity to ban every year! Don't worry, you may think that those are just people, but in reality those are just usernames.

If you know that there exist former members of Godot who have been permanently banned from Godot community, use their stories against them and make scapegoats out of them. Label them as biased, one-sided, and angry. Keep telling everyone that they are on a personal crusade, their criticisms of Godot are driven by a thirst for revenge, and that their actions are done in bad faith. Tell them that they need professional help even if you're not a mental health expert. Remember: labeling someone insane does not breach Godot's Code of Misconduct. Treat it merely as a guideline that can be flexibly disregarded with the mighty power of the ultimate ban hammer! Simultaneously, act politely and extremely welcoming to newcomers. Get preoccupied with bringing in new members to Godot by preaching the gospel of Godot. Keep telling that Godot is unique and innovative. Demonize outsiders, especially capitalists.

In short, you should do everything Juan and Rémi do, and attempt to replicate their behavior. This is the only definite way to become an excellent Blue Robot. The rule of thumb is simple: never question the decisions, always comply, block all criticism. Oh, and don't forget to get Blue Robot merchandise if you want to be accepted into the inner circle of the most trusted Blue Robots!

But if you have conscience, you may start to feel sick. This symptom may be a byproduct of the condition called godotitis, a case of Godot addiction. If this happens, then you'll be presented with two binary choices. But this is a trick! Do not pick anything! Think for yourself.

I hope you've figured that the above is a satirical piece!

Cults

Organizations

There exist for-profit organizations where individuals are generally not expected to provide feedback regarding its governance. There exist private companies that impose various work regulations, and people who work there are fine with those regulations because there are people who are in charge, and there are people who follow orders, do their job, and get financial compensations for the job they do (most of the time, otherwise we're talking about *slavery*).

There also exist non-profit and not-for-profit organizations. These organizations declare themselves to have open-minded attitude, regardless of experience, culture, without discrimination. Because of this, there exists an expectation that mostly everything can be discussed freely in such organizations, including governance. It would be quite naive to expect anyone to change the way someone works, but people are also free to *influence* the decision-making process. Everyone who decided to dedicate their time to work on things on a voluntary basis deserve to be treated with respect. Unlike in private companies where people are mostly driven by relationships that involve money, people who participate in non-profit organizations are mostly driven by passion.

IT and Open Source

Open source is a prominent example. Some open-source projects choose to operate under umbrella of non-profit organizations. Open-source *community-led* projects usually declare principles such as *open discussion*, and tend to attract people from all around the world.

Participating in an open-source community grants a feeling of belonging, and people can make an impact virtually from any point in the world. You do the right thing and make the world better! And this is totally fine. The problem is that, due to various factors, this can lead to a situation where people get strongly attached to a project and even become addicted to this process, which is certainly a bad thing.

This tends to form an *echo chamber*, where outside opinion may be seen hostile towards a project, even if such opinion represents constructive criticism. If a project is driven by a group of leaders, and when that group got attached to it on the personal level, the product ceases being a tool and turns into an object to be protected. Any external cause brought in

by an outsider that hasn't been approved into the inner circle can be turned away simply because that person is not a part of the circle.

This defensiveness shows signs of what we call a **cult** or **tribalism**. Successful and mature projects rarely if ever have to defend their image (other than protecting trademarks), because strong and sustainable image is built by making good quality products instead. It's less about words, but more about concrete actions. In the IT industry, this means building robust and useful software.

This is not to say that all cultish organizations out there are literally cults. A lot of communities will have zealous fans and enthusiastic people that would like to see good things that make this world a happier place to live in. If we take IT, then you may have heard things like a "Cult of Mac", "Cult of Linux" etc. However, there exists cult communities that *look* great on the surface, but when you go deeper, this is where you may realize that it's actually a toxic, abusive, or even destructive cult.

The following chapters covers common characteristics of all such toxic cult communities.

Toxic Cult

Most people who are in a cult don't realize that they're in a cult, or think they are smart enough to avoid it. The problem is that cults are extremely difficult to identify. Experts say there exist thousands of cults. In fact, if you're unaware of manipulative behavior of some people out there, you may fail to see this or even reject this part of reality.

Unlike physical cults in the past, we deal with a new type of cults now: ***IT cults***. This is truly a new and uncharted territory for all people who'd like to discover and study cult psychology in IT.

Here are some major signs and characteristics to look for when identifying a ***toxic cult***, also known as *abusive cult* or *destructive cult*, adapted to IT field. Note that not all signs need to be met in order to classify a group as a cult, it's rather a gradient.

It's also worth noting that deception lies at the core of mind-manipulating groups, and not all actions of such groups may be conscious. Due to this, some signs may not be immediately identifiable on the surface. In fact, this is mostly an unconscious process, but leaders of such groups are motivated by usual things like fame, money, power etc. In other words, if a group declares a particular set of principles, it doesn't necessarily mean that they adhere to such principles factually. For example, leaders declare that all major decisions can be freely objected at any time, while in fact objections are ignored, rejected, or even punished.

Characteristics

Depending on your involvement with a cultic group, characteristics and signs can be noticed at different stages.

When you first stumble upon a group

What is noticable immediately on the surface.

- Existence of charismatic leader(s).
- Group is extremely welcoming and friendly to newcomers, even too caring: love bombing.
- Group is preoccupied recruiting new members via promotion and fixate on fund-raising.

- Group usually presents itself as innovative and exclusive, even elitist.
- Group offers something for free.

When you start to participate in a group

Once you get to interact and experience group activities yourself.

- Group has double standards and hypocritical behavior:
  - Mismatch of words vs actions.
  - Existence of contradictions which are ignored or rejected.
  - False claims and promises, goals are presented as features.
- Questioning, doubt, and dissent are discouraged or even punished:
  - Behavior which suppresses doubts about the group and their leaders.
  - Critical thinking is discouraged, criticism is frowned upon, even blocked.
  - Stopping negative feedback and allowing only positive feedback.
  - Questions are not answered or turned back to the questioner.
- Group has elaborate rules, even redundant:
  - Leaders are sole interpreters of the policy, which may change frequently and whimsically.
  - Reports of deviant conduct to leaders is highly encouraged and rewarded.
  - Different rules for leadership and membership.
- Group has hybrid characteristics:
  - *Example*: leaders are involved both in non-profit and for-profit organizations promoting the same cause.
- Group has a polarized us-versus-them mentality.
- Members feel obligated to give back and constantly recommend joining their community.
- Existence of former members that give their testimonies of being in a cult.

When you become a member of a group

Once you get privileges (formal) or once you get recognized by members and leaders (informal).

- Members devote inordinate amounts of time to the group.
    - Members end up restricting leisure, entertainment, vacation time.
- Leaders restricts what members should say, think, feel, and how to act:
    - Leaders induces guilt feelings in members in order to control their behavior.
    - Leaders ask members to discuss things in private before discussing things in public.
    - Members feel a need to ask for explicit permission to act.
    - Members may be accused of quoting leaders out of context, even if context is irrelevant.
- Group structure is hierarchical and authoritarian.
    - Elections and voting are rare or non-existent.
    - Discourages individualism, encourages group-think.
    - Leaders indirectly or directly suggest to members that the ends justify the means.
    - Relaxed rules for newcomers, more demanding rules for committed members.
    - To break a rule leads to expulsion from the group, even without a right to object.
- Difficulty of leaving a group:
    - A fear of missing out.
    - Feeling obligated to stay in.
- Causes harm to its members, mostly emotional and mental:
    - Extremes of emotional highs and lows.
    - Feelings of tiredness, disappointment, betrayal of trust.
    - Members and outsiders who expose cultic groups to the public may be:
        - Shunned and become enemies or objects of pity.
        - Labeled as a *conspiracy theorist* or *mentally insane*.
        - *Example*: may take a form of insinuations, like “you need professional help”.

References

- Characteristics of Cults and Cultic Groups - By International Cultic Studies Association.
- Cult/Totalist Recruitment Warning Signs - By Alexandra Stein Ph.D.
- Undue Influence - By Steven Hassan, a cult expert.

- Mind Control Made Easy (or How to Become a Cult Leader) - Satirical drama by Carey Burtt.

Blue Robot Cult

Godot Initiation Ritual - By יעקב

Godot Engine is a cult, albeit an IT-focused one. While many people are familiar with the seemingly friendly user side of Godot, it is important to recognize that cults often operate in this manner—outwardly welcoming but internally abusive. Most Godot users have no clue about the dynamics within the core contributors' community, and contributors often cannot recognize abusive behavior towards them due to the *undue influence* of Godot's *toxic leadership*.

Here's a quick summary graph that humorously encapsulates what it's like to navigate through all the stages of indoctrination in the Blue Robot Cult—a descent that ends in complete disillusionment:

> After a discussion in Devil's Rejects about the problems, flaws, and years of fundamental 'still broken' things in Godot with other disillusioned years-long Godot users -- I slapped together this mock graph (and yes, I admit I'm terrible at graphs). 🤣 pic.twitter.com/qzPeoWvnoc
>
> — LillyByte (@LillyByteGames) May 17, 2023

When people claim that Godot is a cult, its followers or those heavily influenced by it might label you a conspiracy theorist, accuse you of holding personal grudges, or even call you crazy. The following clip is from a satirical documentary about cults, adapted to fit the Godot community. In the clip, a woman wearing a shirt that says "Cult Member" calls you insane:

> As an ex-developer of @GodotEngine, here is what kind of response I hear from the members and followers of Godot cult in return. #GodotEngine #CancelGodotEngine pic.twitter.com/ClyMwE7HzU
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) June 9, 2022

While Godot is commonly viewed as just a game engine, it has also been variously defined as a movement, a cult, a grift—you name it. The truth is that Godot encompasses all these facets, depending on the local communities and cultures surrounding Godot. Nevertheless, when we scrutinize the actions of Godot members, particularly the hypocritical behavior exhibited by Godot's leadership, it becomes evident that they predominantly function with a cultic, groupthink mindset. Regardless, it remains undeniable that Godot wouldn't be what it is today without its leadership.

Some respected followers of Godot openly acknowledge its cultic nature. They assert that Godot is indeed a cult, but interestingly, they do not view it as a negative characteristic[1]:

> *Godot definitely is a cult, but it's not a bad thing.*

It is true that not all cults are inherently destructive. However, when we examine other cult characteristics, it becomes apparent that Godot exhibits signs of being a *toxic cult*. Some of these signs manifest as subconscious *cognitive dissonance*, as evident from what aforementioned Godot follower replied further:

> *If you do, in earnest, believe Godot is a cult, then posting this to the cult members on the forum certainly will not be taken well.*

You should definitely check out the in-depth analysis of the discussion surrounding this issue[2]. It not only pushes back against attempts by Godot followers to scapegoat a former member, but also reveals how their own behavior contributes to the toxicity within the Godot community. Sadly, the analysis has been removed by Godot forum moderators (not locked, but deleted entirely), which only serves to reinforce the initial hypothesis put forth in the analysis. This deletion was pretty much expected given their predictable behavior.

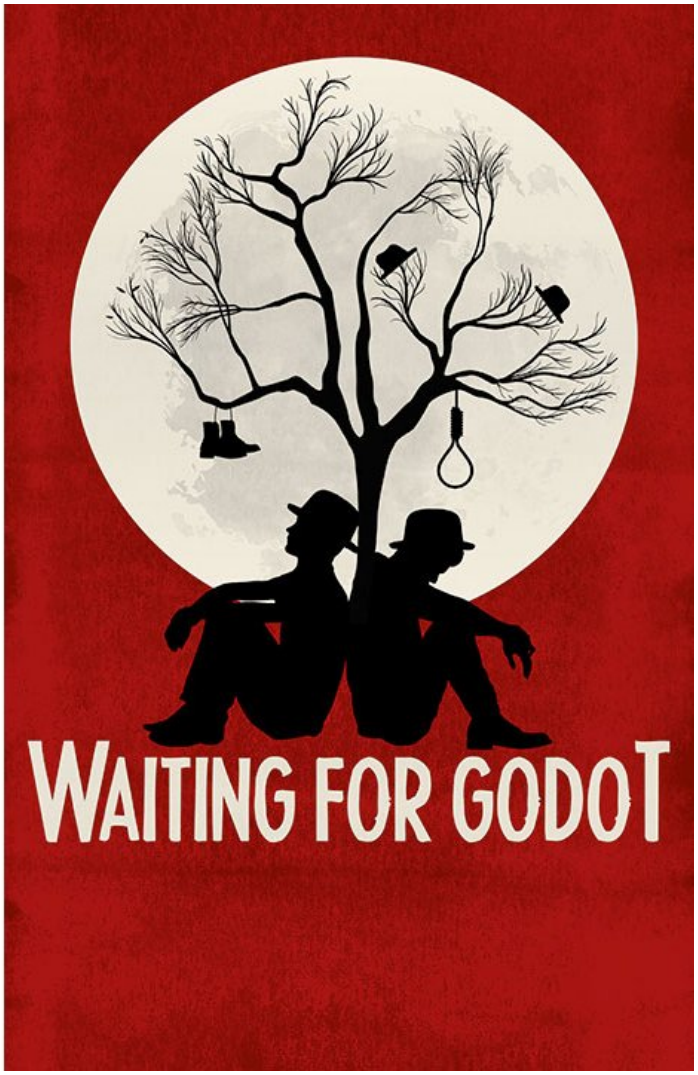
When we examine Godot as a cult, it's crucial to clarify what is meant in this context. One has to realize that there exists a difference between a *cult* and a *sect*. The term "cult" typically brings to mind something related to religion. Traditionally, a sect is often associated with a religious group that has broken away from a larger religious organization. However, a cult also refers to a group with extreme devotion to a person, idea, or thing, often led by a charismatic leader.

Discovering that Godot is a sect would be incredibly shocking. But besides the fixation on manifesting Godot-related symbolism in the real world and people devoting themselves to the Godot ideology, there isn't sufficient evidence to support that claim. At least, not at this point! 😉

> The new, future studio for my #GodotEngine youtube channel. Lot of work left, but will be worth it. pic.twitter.com/Nlpwj32KKh
>
> — StayAtHomeDev (@StayAtHomeDev) March 2, 2024

YouTube channel? Nah! **#GodotEngine** YouTube channel? Yes please! 🙃

However, considering the origin of the Godot's name, derived from Samuel Beckett's Waiting for Godot, a tragicomedy where Vladimir and Estragon wait for an enigmatic character called Godot who never arrives, one could marginally view Godot as a religion, see Value of Waiting. Godot is open to various interpretations, and there is faith in Godot's eventual arrival. In Waiting for Godot, Vladimir and Estragon even contemplate suicide, making this an eloquently concerning factor. This satirical poster encapsulates the essence of Blue Robot Cult:



Godot is like a neo-cult, an echo chamber which exists mostly in the realm of mental constructs where Information Technology (IT) plays a significant role in their brainwashing, such as openwashing methods, or plain big lies. There is clear evidence to support this claim. Let's highlight some of the cult characteristics of Godot:

- Godot is offered free of charge under an extremely permissive license.
- Godot has a charismatic leader, Juan Linietsky, aka reduz, who is a communal narcissist.
- Godot is extremely welcoming to newcomers and promotes an image of a playful community.
- Godot is preoccupied with recruiting new followers, particularly younger people.
- Godot fixates on fund-raising despite allegedly being developed purely out of love.
- Godot presents itself as innovative, novel, unique, and even elitist.
- Godot exhibits an us-vs-them mentality, such as attacking potential competitors.
- Godot discourages and blocks criticism, such as making comparisons with other engines.

The following meme pretty much sums up the problem of Godot's inability to handle criticism:

> It's sooo much easier to combat your critics than to focus on fixing your engine. #Godot #GodotEngine pic.twitter.com/uN0t6GHZJ0
>
> — 🌈Saas - Bi- Bier trinkende Frühlings Edition🔨🦈 (@S3ys__) April 30, 2024

It's worth noting that it's the combination of these factors that makes Godot a cultic organization. For example, many open-source projects are provided free of charge, yet they are not toxic cults. In addition, the characteristics listed above are surface-level, and there are other, less obvious characteristics that we'll explore in future chapters. But you can already get a glimpse of how Godot presents itself to the gaming industry by looking at memes produced by people outside of it:

> Game dev community: *exists*
> Game devs: pic.twitter.com/SXi9AQVFFh
>
> — Sohom Sahaun (@sohomsahaun) September 3, 2021

We can certainly learn from Godot... But *what* to learn? One can learn how to be a cult leader and a cultist, and to repeat the same narratives regurgitated by cultists, when such narratives do not reflect the reality. Many of the touted "benefits" of Godot are exaggerated. When asked why Godot is superior to other game engines and tools in specific aspects, most waiters for Godot cannot provide detailed examples or their answers are too general to be taken seriously[3]. Because of this, Godot tends to attract hobbyists rather than professional developers.

I suggest watching this satirical video about Godot on this topic. It humorously and symbolically captures some of the key stages the author of this book experienced. As the saying goes: "Every joke contains a grain of humor, but the rest is all truth."

Word of advice: If people label something as a cult and you notice angry or weird reactions from the group being criticized, you should question everything about the group you're considering joining. Post-ironically, these remarks turn out to be surprisingly accurate. On the other hand, not everything labeled a cult is inherently destructive, it really depends on the specific social dynamics within those groups. Either way, the following post helped me uncover the truth about Godot back in the day:

> Based on the fact I am still getting replies on this, yes.
>
> — Mig (@wtfmig) June 9, 2022

Conclusion

We might worship some things in our lives, but when innocent fanaticism transforms into extreme bigotry, it causes harm to everyone. Unconsciously dedicating a significant amount of your free time to furthering the hidden agenda of cult leaders is definitely a problem.

Take, for example, the scenario where you're constantly keeping up with Godot's development because of a *fear of missing out*, promoting Godot, and/or contributing to its development, all the while putting off actually creating a game with it. So, it's crucial to take a step back, look at the social dynamics critically, and acknowledge the potential mental toll of being part of communities like Godot.

The following information is particularly relevant to those who currently contribute to Godot. Contributors of Godot lack a clear vision for the project. When one attempts to reveal

it or ask questions about Godot's governance, the leadership and their followers can become repulsive and aggressive, revealing the true nature of the so-called "welcoming" community.

Being informed is essential in breaking free from the grips of undue influence and reclaiming autonomy. Awareness, being mindful, and conscious is the most effective way to avoid joining such toxic groups in the future. Personally, I'd rather move on and spend my time on more interesting things than writing this, but hey, it's my ethical duty to give you the heads-up, so you don't have to worry about it! Without further ado, let's take into account characteristics that we have covered in Toxic Cult chapter and apply that to Godot community.

References

[1] Is Godot a new cult phenomenon in the IT industry? - Godot Forums.

[2] Analysis of "Is Godot a new cult phenomenon in the IT industry?" - Godot Forums (deleted thread).

[3] The journey to Godot 4: what can you expect? - GodotCon 2023.

Cult Leader

💡 The fish rots from the head!

Godot's cult leader is definitely **Juan Linietsky**.

One of Juan Linietsky's avatars - The Robbie

Before delving into the topic, it's worth noting that Juan Linietsky was born in Argentina. There are many cults in Latin America[1], ranging from destructive cults to more controversial ones. Quote from the "Cults in Latin America" article:

> *The author estimates that more than 5,000 religious groups operate in Argentina, with as many as 50,000 sects and cults throughout Latin America.*

Due to this, Juan could consciously or unconsciously adopt cultic techniques to utilize in his own community of Godot followers. As we've covered in Value of Waiting chapter, the mere concept of "Waiting for Godot" and the ambiguity that accompanies it create a perfect environment for such cultic engagement.

Juan may have also been influenced by some overzealous Latin American developers wanting to promote their supposed talent to worldwide community. Interestingly, in the past, Juan Linietsky worked as a musician at Sabarasa Entertainment[2], a game development company in Argentina.

Historically, expansionist ideologies have been fueled by an "us-vs-them" mentality, as it reinforces the notion of one's own group deserving dominance or control over others, even under the guise of liberation. The following image ironically suggests some aspects of this, taken from Sabarasa's main page (upscaled):

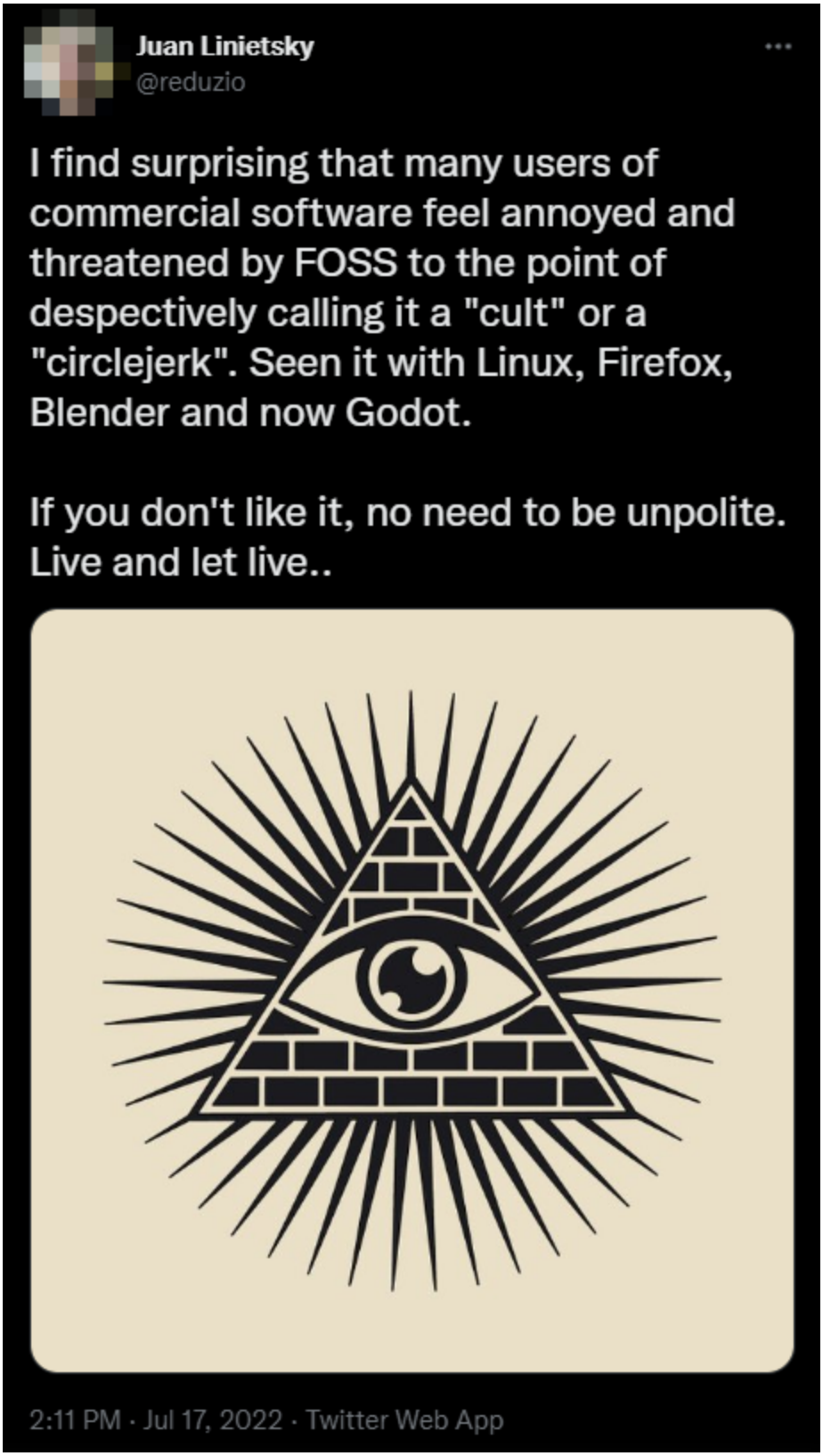
Coincidentally, this expansionist mentality can be observed in the behavior exhibited by the followers of Blue Robot Cult themselves:

> I'm loving these expansions
>
> — Dennisse Pagán (@PaganDennisse) September 23, 2023

Apologetics

The first concrete thing which comes into focus is Juan's post on this matter[3]:

> *I find surprising that many users of commercial software feel annoyed and threatened by FOSS to the point of despectively calling it a "cult" or a "circlejerk". Seen it with Linux, Firefox, Blender and now Godot.*
>
> *If you don't like it, no need to be unpolite. Live and let live..*



Interestingly, Juan said this at the time when the first edition of this book was being written, and when I, as a former member of Godot, began publicly calling Godot a cult. For this reason, I suspect that what Juan says is mostly a reaction to my own writing about Godot. Let's analyze his post further.

Don't be deceived when Juan says that only users of commercial game engines call Godot a cult, as Juan deceptively portrays it. Even then, it's done in an **us-versus-them** mentality, which is a definitive sign of a cult. As a toxic leader, Juan creates a perceived threat, something along the lines of "commercial engines want to destroy FOSS", while commercial engines just go about their business without paying much attention to communities like Godot.

Furthermore, commercial companies can be classified as IT cults to varying degrees, much like open-source organizations. Therefore, it makes no sense to use this divisive narrative to label users of commercial software as the enemy. Such labeling is not only unjust, but also counterproductive. When people consistently describe a project as a cult, it provides an opportunity for project leaders to reflect on why. Most importantly, it's a chance to ask questions, rather than launch an attack against critics.

Despite the above, it's truly unsettling to witness some commercial engines unexpectedly sponsoring Godot. Juan's accusation against users of these engines is like biting the hand that feeds you. Beware of a potential enticement faced by commercial engines to support Godot, driven by the pervasive efforts of Godot advocates infiltrating communities such as Unity and Unreal (you will find one such example at the end of this book). These advocates are found falsely portraying Godot as one of the "world's third biggest game engines," a claim that starkly contradicts reality. What is of greater concern is the notion of replacing established engines with Godot, fueled by such misleading assertions.

Juan continues saying "live and let live". This is how *narcissists* play the victim to deflect criticism. They **induce guilt** so that you cannot possibly say anything bad about them or expose their lies, because of this false sense of victimhood. There are testimonies that reinforce the idea that Godot is an actual cult, so Juan finds it necessary to divert the attention from those testimonies and **block criticism** of Godot.

There's also another subtle message that Juan is trying to convey here. He hides behind FOSS and other projects like Linux, Firefox, Blender, and implies that Godot is *finally* being called a cult, while trying to present it as a sign of success. Of course, it's just a sign. The word "cult" is usually used in a pejorative sense, and cult apologists would like people not to call their organizations cults, because they are[4]! For this reason, when Juan tries to convince people that Godot is *not* a cult, it makes him a cult apologist. Healthy organizations do not feel the need to say that they are not cults.

While some communities may exhibit certain cult-like characteristics, it is crucial to differentiate them from outright *toxic cults*. The former might display traits commonly associated with cults, but they may not reach the extreme levels of toxicity. It is worth noting that the absence of former members describing an organization as a toxic cult from an insider's perspective could be attributed to their reluctance or fear of speaking out about

their experiences. Most individuals decide to move on and actively avoid any potential retaliation from a toxic cult.

> it's a pleasure to be part of this cult for so many years pic.twitter.com/cfVrYgasAg
>
> — Tumeo (@tumeo_) July 17, 2022

Based on the factors described thus far, susceptible followers believe in Juan's unquestionable and authoritative vision. As a defense mechanism, followers of Godot, who are part of the cult, start making "jokes" about Godot being a cult (yes, the following "joke" is made by a Godot follower)[5]:

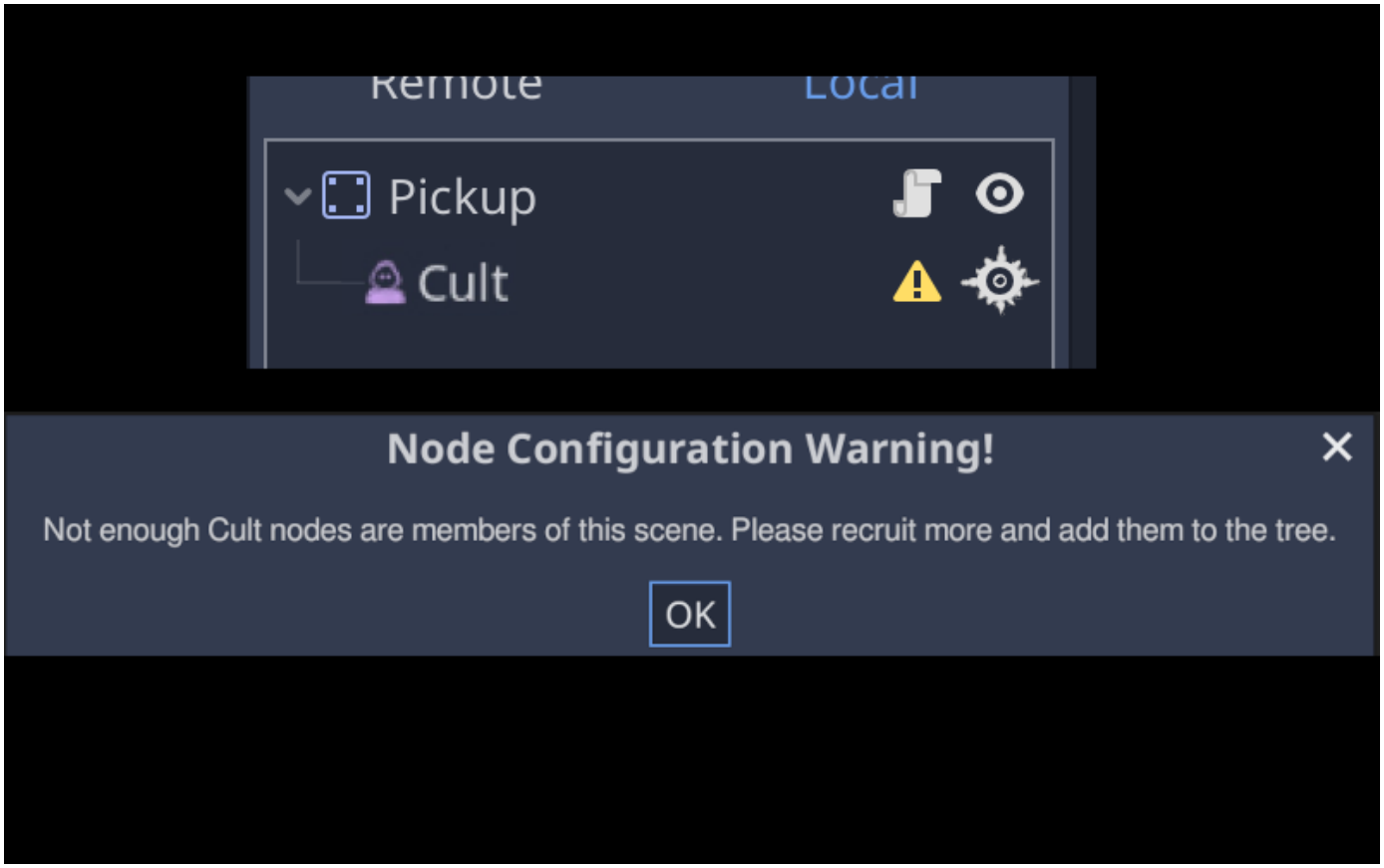
Take note of the **#badjoke** hashtag. All of this exemplifies cognitive dissonance in action, which serves as another indication of cult-like communities. Typically, people who share authoritative style of thinking (Godot cultists included) are hesitant to make jokes due to the potential negative repercussions imposed by toxic leadership, as they often interpret jokes as insinuations of something truly negative about them or an organization, which is often just two sides of the same coin in the context of mind-manipulating cults.

To reiterate, Godot followers camouflage themselves within Open Source communities by labeling them as cults, thereby implying that projects like Godot also possess cult-like characteristics as if it's no big deal. While some Open Source projects may display cult-like signs, they may not necessarily be classified as *toxic cults*. However, if a project displays *all* or *most* of those toxic cult signs, then it's definitely something to be wary of.

In an exercise of curiosity and irony, it is interesting to note that the very word "contribute" originates from the word "tribe": "contribute" → "tribute" → "tribe"[6]. These words encompass meanings and notions ranging from "bring together" and "pay" to "barbarous people". As you delve deeper, you'll discover why the Godot community can be considered to have truly tribal characteristics, with the NIH syndrome playing a huge rule... 😁

Current situation right now
byu/IamTrubak inUnity3D

And here's a "joke" by Juan Linietsky himself on this matter:



Godot "Cult" Node - Juan Linietsky

> Sorry, as with many things in Godot it lacks documentation. These nodes don't do much, you add them to the scene and immediately get a configuration warning that tells you what to do. pic.twitter.com/EGc6odu9EE
>
> — Juan Linietsky (@reduzio) July 17, 2022

There's one joke: "*Every joke has only a fraction of a joke, the rest is all true.*" 😉

Cult members are taught, directly or indirectly, to believe that they are special, superior, and that the rest of the world is just ignorant, as evident from Juan's articles such as "How to make your dream game, publish it and not die in the process"[7]:

> *Even applying standard design patterns isn't warranted to work. One situation I've seen repeated over and over as a consultant is having to deal with engineers that **lie to themselves** [emphasis added] about successfully applying MVC to their game architecture (and making a complete dish of spaghetti and meatballs as a result). Another case is engineers claiming they find success in not using OOP design (while they use it anyway, without realizing it or admitting it). I've seen this so many times I can't count it with my fingers.*

In the same article, ironically, he even uses the "cult" word in the context of publishers and

their followings, as if it's a no big deal:

> [...] *At first, **I was angry** [emphasis added] at this fact, but I understand now that it's the most natural course of action to do for a publisher. They are companies, not charities.* [...] *Also, don't get me wrong on this. Publishers still make a big difference. Even if they don't invest money in promotion, they usually have very oiled channels with media and many even have **cult followings** [emphasis added] (like Atlus or Daedalic) that will purchase their games.*

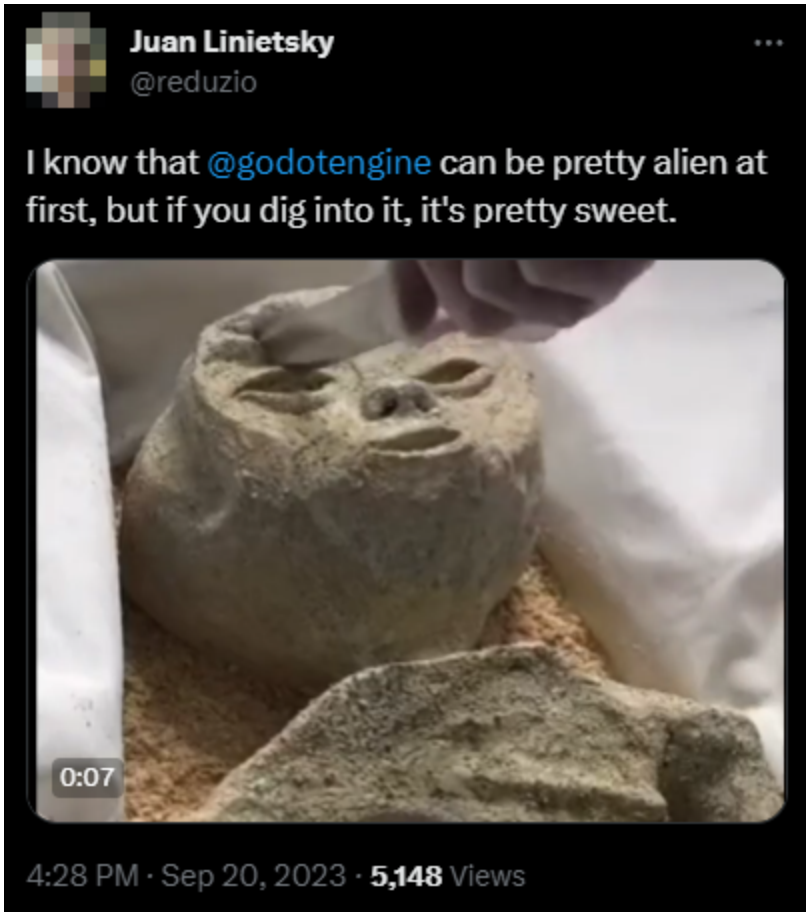
Juan uses the same "cult" word to describe a game following[8]:

> *I am pretty happy to just find out that a Nintendo GBA RPG game I made the entire OST for almost 20 years ago, not only seems to have a **cult following** [emphasis added] today but that there are so many people who really enjoyed the sountrack. Very nice way to start my weekend!*

Reread Juan's post at the beginning of the Apologetics section and ask yourself: Why would Juan be offended by Godot being called a cult when this is precisely what he does when describing some companies and people? Juan, if you don't like it, no need to be unpolite. Live and let live.. 🙃

For students and hobbyists with limited industry knowledge, the supposed "proof" of Juan's expertise lies in the number of contributors he has working on Godot and the number of followers spreading the word of Godot. Of course, Juan's implied "solution" to the described "problem" is to use Godot, even though he may not explicitly state it. This purpose is often the driving force behind his articles. They are frequently misleading, as they exploit the **fear of missing out** (FOMO), fueled further by emotional rhetoric like "not dying."

Cult members are also discouraged from seeking information or testimonies that might challenge the cult's doctrine. This makes them more dependent on the cult leader and the group for their sense of identity and reality. Cult members may also rationalize their involvement in the cult by claiming that they are **misunderstood**. In the case of Godot, this is evident from the jokes that they made, as well as how they react to criticisms:



Godot can be pretty alien - Juan Linietsky

Juan posted the above "joke" precisely when Godot experienced a sudden influx of Unity users, with some industry experts criticizing Godot. Initially, Juan acted quite opportunistically in doing so, particularly as he aims to attract unsuspecting contributors and wealthy donors. Simultaneously, Juan is concerned that these Unity users might expose both Godot's limitations and his agenda, prompting him to behave in this manner, thereby reducing people's vigilance:

> Instead of trying to annoy people to switch to Godot, the best we can do is work hard on our side so Godot becomes as best as possible and fantastic alternative to other engines. It is then that more people will feel more comfortable to come and use it.
>
> — Juan Linietsky (@reduzio) May 5, 2024

> *Folks, please don't tell people who are struggling with Unity related issues to use Godot. At this point, I think everyone in the industry knows Godot exists. Please be mindful, Its not an obligation for them to use Godot, and there are hundreds of good reasons to not use it.*
>
> *Instead of trying to annoy people to switch to Godot, the best we can do is work hard on our*

> *side so Godot becomes as best as possible and fantastic alternative to other engines. It is then that more people will feel more comfortable to come and use it.*

Wanna take a wild guess if Juan really has a list of "hundreds of good reasons" *NOT* to use Godot? 🙃

In essense, Juan's underlying message is that the Godot community is becoming more of a headache. Instead of helping Godot gain industry acceptance, their overzealousness is doing the opposite. Juan is hinting that they should dial back on pushing Godot as the go-to choice. But make no mistake, Juan's ultimate goal remains unchanged: he still wants to replace or overthrow Unity, no matter what reasoning he offers or what he declares to the public.

It's worth mentioning that Juan's renewed career with Godot as a supposedly open-source project began with extensive promotion claiming it was "on par with Unity"—a lofty assertion that didn't align with reality, *especially* in 2014. This is evident from an article titled "A New Open-Source Game Engine Being Released" published back in 2014, weeks before Godot went open-source[9]:

> *The tech has now proven to be quite mature and is now very complete and according to its developer to be on-par with Unity, or arguably superior to Unity when it comes to the area of 2D and animation support.*

Over time, Juan came to realize that propagating such big lies about the capabilities of his pet project became excessive. However, such grandiose claims allowed him to advertise the engine to Unity users without charge, solely relying on the concerted efforts of Godot fans eager to spread the word in hopes of earning Juan's pat on the back. Take away all the grandiose claims and unpaid endorsements of the engine, and let's face it, Godot would just fade into obscurity.

Cult of Personality

Community literally creates a *cult of personality* out of Juan. You can even stumble upon pictures of people that worship Juan himself[10]:

Here, you see a portrait of Juan, and the Godette character working on Godot.

Project manager of Godot, Rémi Verschelde, who is the sub-leader in relation to Juan, calls Juan "*the prophet of the engine*" at the game development conference (GDC):

In the interview with Press Over, Juan Linietsky was referred to as “El Gran Linietsky,” which translates to “The Great Linietsky.”

Politics

This cult of personality is further confirmed by Juan’s own hidden political views. Juan once written a release article, with a picture suggesting Donald Trump[11]!

The phrase “Make engines great again” takes inspiration from Trump’s slogan “Make America Great Again.” This connection is significant because experts in the realm of cults, such as Steven Hassan, have identified the existence of a “Cult of Trump.”[12] This reinforces the notion that Juan, as the leader of the Godot cult, is likely influenced by such charismatic leaders as perceived by their followers.

By the time of writing this book, Juan deleted this image and replaced it with a “Vote for Godot” image, again, right after I started to describe Godot in terms of a cult. Also, recall Godot’s “Democracy” mental tricks covered in Democracy chapter. These kind of things are definitely deceptive and manipulative, since Godot is in no way a democracy.

Conclusion

If you haven’t had the opportunity to personally know or speak with the lead developer of Godot, it may be challenging to definitively determine whether Juan is indeed a cult leader. Nevertheless, he exhibits all the classic signs of one, demonstrating a mastery of

manipulation at an expert level. As you continue reading subsequent chapters, you will hopefully gain insight into the characteristics that make Juan a toxic cult leader. Unraveling his methods and understanding his undue influence on the Godot cult followers and members will shed light on the depth of his manipulation and the potential dangers he poses.

References

[1] Cults in Latin America - Cultic Studies Association.

[2] Sabarasa Entertainment - Sabarasa Entertainment website.

[3] Juan Linietsky's post on Godot being called a cult

[4] Cult Apologists | Cult Defenders - By Apologetics Index.

[5] Worship the ancient God Ot - By Miskatonic Studio.

[6] Etymology of the word "contribute" - Online Etymology Dictionary.

[7] How to make your dream game, publish it and not die in the process - By Juan Linietsky.

[8] Juan Linietsky about his OST in Nintendo GBA RPG game

[9] A New Open-Source Game Engine Being Released - By Phoronix.

[10] Juan, Godette and Godot Engine - By Abatta Games.

[11] Make engines great again! - By Juan Linietsky.

[12] The Cult of Trump - By Steven Hassan, 2019.

Pattern of Behavior

Because the lead developer of Godot plays a huge role in Godot's toxic cult, it's very important being able to recognize his manipulations.

Behavior of Juan Linietsky, who is in fact the cult leader of Godot, can be summarized as undue influence, which manifests as hypocrisy, lack of empathy, inattentiveness, irresponsibility, and carelessness.

Below is my critical analysis.

Juan posses a belief that he's destined to help people to make great games using free tools alone, which manifests as Messiah complex. He attempts to create a game engine which could allow users to create both casual and AAA games with little to no effort. He wants to make a game engine accessible to everyone. But if you run after two hares, you will catch neither. He will always prioritize features that make the image of Godot look nice in the eyes of beholder, like editor features or movie maker mode, at the expense of neglecting core parts of the engine.

Juan attempts to create an image of being inclusive for all people around the world, trying to be all things to all people, but also trying to hit advanced features. He moves the goalposts a lot, both in terms of arguments and development goals.

Juan deliberately downplays himself, plays a victim to induce guilt in others, as if users are forcing him to work hard in Godot to make advanced game engine comparable to other commercial game engines out there, and allegedly experiences “pressure” because of this. Juan may say that he tried his best, but in reality he says this to cover up his irresponsibility. He does not finish features that he himself created or integrated in Godot, he says something like “I don’t have time for this, interested contributors will fix bugs themselves”.

Juan is mostly detached from real game development nowadays. Due to this, he doesn’t really understand how a particular feature should work in practice, so he tends to fixate on ideas. Simultaneously, he says to contributors that Godot’s development is extremely pragmatic, which is not true, because carelessness is not pragmatism.

When confronted with criticism, Juan fights it off by saying that he has more experience, or that others don’t understand Godot’s architecture. For contributors to gain his trust, Juan says that they must have “irreproachable attitude”, which means unquestionable attitude in practice. Juan publicly says that he encourages public discussions to users. But privately, he tells them that expressing disagreement in private is far more productive and useful. Therefore, Juan creates an environment where consensus cannot be possibly reached, only compromises, because it’s oftentimes not clear what kind of decisions were made behind the doors.

Juan plays a democrat. To users, he says that Godot’s governance has horizontal structure. But to contributors, he says that Godot’s development is based solely on trust, and not meritocracy nor democracy. He spreads manipulative propaganda: he substitutes concepts like “community-driven”, or says that number of contributors keeps growing “exponentially”.

Juan creates perceived threat, like “commercial engines feel threatened by open-source”, going out of his way to suggest companies to adopt pure Open Source terms, showing all signs of us-vs-them mentality. He comes up with a bogus solution to a bogus problem, which is the only core development “philosophy” in Godot, by saying that Godot has no philosophy.

Juan presents common knowledge as unique and innovative and demonizes approaches that doesn’t work for him. He creates surrogate solutions in attempt to match up to expectations of commercial engines, but such solutions end up being incomplete and overly simplified. He often rejects and even removes features from Godot to achieve a preconceived notion of purity, hidden under assumption that bloat accumulates quickly in Godot, despite the fact that engine’s binary size keeps growing in leap and bounds year by year in spite of this expressed purism.

Juan temporarily rejects proposals by saying that a feature is not needed, but then implements that feature himself years later. This kind of tactic is more likely to be used if a proposal is complemented with a finished solution. He rarely if ever reads elaborate proposals. Juan severely lacks tact when rejecting proposals. This is because he's unable to understand and experience feelings of other people, completely lacks empathy. He may occasionally express feelings of sadness and grief, but it's not genuine, more like "crocodile tears".

Juan misinterprets facts. He's extremely inattentive and lacks focus. When Juan is exposed in inappropriate behavior, he starts blame shifting by saying that others misunderstood what he said or done. He is a master of gaslighting, a manipulation technique that allows to undermine the perception of reality. He misinterprets someone else's messages or make up imaginary situations. People who don't pay attention to this tend to agree to his contrived conclusions that are built upon strawman arguments.

Juan uses love bombing techniques. He says that his work is done entirely out of love, convince people that they don't have to be professional or serious as other engines. Alternates love with coercion, especially when contributors start to question Godot's decisions. At the same time, Juan expresses ambivalent attitude towards commercial game engines. He wants to achieve the success of commercial game engines out there who he believes to be controlling the industry, but he will never tell this explicitly, because he experiences jealousy. Sometimes, you may see Juan covertly referring to Unity and Unreal as "two major technologies". Yet he discourages users to compare Godot to any other technology, and say that Godot doesn't compete with Unity, Unreal etc, to the point that "they are free to use Godot's code".

Juan describes Godot's unusual governance as an "uncharted territory" for all people who'd like to take part in game development. Depicting it this way allows to create ambiguous environment where deception may be hidden and presented as mistakes, misunderstandings, or misinterpretations.

Juan says that he only tweets about Godot updates, and that Godot is the only thing which interests him in life. But this is false, because he oftentimes talks about politics, like Russian invasion of Ukraine, military, gambling, etc. He doesn't care about people, unless they contribute to Godot's development. Unfortunately, due to his false expression of confidence and guru-like behavior (as in, claiming how the industry works according to his sole experience), he becomes a real-life role model for a lot of susceptible followers (a "younger generation", as Juan refers to them), which is alarming considering all above.

Juan was probably a "Nice Guy" decades ago, but instead of overcoming it, his insecurities allowed him to develop a painful attitude to criticism over time, so he has accumulated a lot of trust issues. Juan exhibits both vulnerable and especially communal narcissistic qualities.

In other words, Juan is a wolf in sheep's clothing.

He will deny everything written above because that's what narcissists typically do. On rare occasions, he may even disingenuously acknowledge some of the aforementioned characteristics, publicly admitting past mistakes in an attempt to evade responsibility and convince others that he has changed. However, nothing could be further from the truth.

Rhetoric and Narrative

- Simple answers are Juan's specialty. The allure of simplicity helps to alleviate the negative feelings that we may experience due to analysis paralysis, making us naturally inclined to prefer straightforward answers over complex and analytical ones. Unfortunately, this preference is frequently observed among Godot followers, who tend to dismiss critical discussions about their engine as mere "conspiracy" talk. As such, Juan frequently employs the word "speculation" as a means to dismiss criticism. It serves as a lightweight tactic to subtly insinuate that the critic is akin to a conspiracy theorist when questioning Juan's actions.

- When users express concerns, you may see Juan writing messages to "clarify" things. He constantly says that he's not tired to "clarify" things again and again and asking people to have a chat with him in private. Propaganda is fed via so-called "clarification" process. Therefore, instead of addressing underlying issues, he attempts to convince people that an issue doesn't exist or turn it into a benefit.

- Juan employs manipulative tactics by starting his propaganda with general introductory phrases such as "*Many people ask me*" or "*Many people wanted to know,*" which can be qualified as *weasel wording*. These phrases serve to create a false sense of importance and build unwarranted credibility for his subsequent statements.

- When Juan says to you "*If that helps*" when he justifies his actions or an obvious contradiction in Godot, he's likely lying.

- When Juan says to you "*Again, ...*" he expects you to back down. This is how he attempts to show off his supposed supremacy.

Case Studies

The following analyzes some notable statements made by Juan, and uncovers non-obvious, hidden, and/or manipulative assumptions and presuppositions behind Juan's statements. The main task here is not to analyze everything Juan ever said in the past, but to show the process of applying critical thinking.

False Expertise

> *I have been asked countless times about books to learn how design and make a modern game engine (like Unity, Unreal or Godot). I really have no idea what to recommend. I don't have much free time (can't promise anything), but would anyone be interested if I eventually write one?*

Juan says that he has been asked countless times and *assumes* that he is considered an expert in this field or has substantial knowledge about how *modern* game engines work, which may suggest **false expertise**. By stating that he has been asked countless times about recommended books, he is **seeking validation**. He may be using this tactic to **build credibility** and imply that his knowledge is in demand.

He mentions that he doesn't have much free time, implying that he might not be able to fulfill the request. However, this can also be seen as a manipulative tactic to create a **sense of scarcity**. By suggesting that he might not have enough time, he may be trying to **increase the perceived value** of his potential book or the work that he does in Godot.

Juan plays on the **fear of missing out** (FOMO) and encourages people to express interest, regardless of whether he intends to write the book or not. Juan does not provide any details about what the book would cover or how it would be delivered. It's possible that he is using this vagueness to generate curiosity and interest among potential followers. Furthermore, Juan may not be inclined to recommend books to read because he doesn't want to promote other works, because this could potentially lead Godot followers to discover alternatives that better align with their needs, causing them to abandon Godot.

At the same time, phrases like "*I really have no idea what to recommend*" and "*can't promise anything*" can be seen as **false humility**. He may be downplaying his expertise or intentions to appear more relatable and garner sympathy from the audience, who are often students.

He mentions Unity, Unreal, and Godot in the same line. He indirectly implies that those engines can be compared on the same level, which creates a **false expectation** (read

Misleading Competition below). Additionally, he implies that his book or Godot would be equally valuable or essential to aspiring game developers in the game industry as a whole.

Misleading Competition

Before analyzing the following statements, recall what Juan said that Godot does *not* compete with Unity or Unreal, see Competition chapter, which is worth repeating here:

> *Friendly reminder that Godot does not compete with Unity or Unreal.*

Here are Juan's statements that we're going to analyze. Context: Juan shows statistics of various engines being used for game jams: 59% for Unity, 19% for Godot, 5% for GameMaker, and the rest is other engines:

> *I think leaving aside the competition debate (which quite frankly, despite my position that it's not, its just all subjective rant), I hope game developers realize that at this point, if not for Godot, in 10 years Unity would have become the Photoshop of game development.*

Godot's existence is presented as the sole factor which allegedly prevents Unity from becoming dominant. The statement "*if not for Godot, in 10 years Unity would have become the Photoshop of game development*" is a speculative claim lacking concrete evidence. It assumes that Unity's growth would lead to negative consequences akin to the dominance of Photoshop in image editing. However, the claim relies on a **fallacious analogy** since it fails to provide any rationale or data, and whether the dominance of Photoshop has any *negative* consequences can also be questioned in a similar manner. Additionally, game development attracts various branches of computer science, making it a complex and diverse field, and thus there may exist other factors preventing Unity's dominance.

Juan presents a **false dichotomy** by assuming that Unity's dominance would be universally undesirable for game developers, without considering the possibility that developers actively choose Unity due to its features and capabilities. This oversimplified view creates a **false dilemma**, where the only two options presented are Unity's dominance with negative consequences or Godot's existence preventing a supposed monopoly.

While Juan presents statistics of engine usage for game jams, he uses this data to support a **sweeping generalization** about the future of Godot in the entire game development industry. Game jams represent a specific context and may not accurately reflect long-term trends in game development industry, making inattentive followers of Godot falsely believe

Godot's adoption by the industry as a whole.

Juan dismisses any potential competition debate as "*subjective rant*" suggesting that any opposing views about the engines' merits are not worth considering, that discussing drawbacks of different game engines are unimportant or irrelevant. This framing attempts to **marginalize** other perspectives and makes it appear as though only Juan's opinion is valid, reinforcing his authority.

The statements describe how Juan's communication subtly **manipulates** game developers' perceptions and decisions as well. He sends a hidden subconscious message that implies game developers lack agency in shaping the future of game development and must support Godot to "reclaim control" over the industry. This overlooks the role of developers in considering, evaluating, and choosing from various alternatives, including the possibility of building their own specific solutions.

Juan's statement exhibits a strong bias in favor of Godot, positioning it as a **savior** preventing a negative outcome, and **downplays** the potential for other game development engines to offer viable alternatives. This bias may impact objectivity when evaluating the potential trajectories of game development engines, as it influences developers' perspectives and decision-making processes.

Blind Pragmatism

> *I always believed that your worst enemy as a software architect is the belief that you add a feature because "you or somebody will probably need it in the future". Even if it can happen, my experience was always that the more pragmatic you can get with problem solving, the better (and far more maintainable) the code produced. I strongly believe that future proofing is a trap, but realize that its not easy to get into a pragmatic mindset.*

Juan's experience and beliefs to hold universal relevance for all software architects constitutes an extremely one-sided perspective. Such assertions disregard the possibility that projects may necessitate varied approaches to future-proofing. **Generalization** of personal experiences is perilous, as the Juan's individual encounters are not representative of all scenarios. Such an extrapolation leads to biases, discounting other critical factors.

The dismissal of future-proofing, or rather, the ability to meet future expectations, as unnecessary and counterproductive, demonstrates a **myopic viewpoint**. It fails to recognize the potential benefits of thoughtful consideration and planning for forthcoming needs. While over-engineering can be wasteful, it is essential to acknowledge the importance of striking a balance and embracing the advantages of foresight.

A **false dichotomy** is presented by Juan, artificially pitting future-proofing against pragmatic problem-solving, as if these two crucial aspects of software development are mutually exclusive. This fallacious stance overlooks the potential for synergistic integration, wherein a blend of both can yield optimal results.

Juan demonstrates and advocates a **fixed mindset**, prioritizing blind pragmatism to the detriment of considering future needs. This rigid mindset inevitably inhibits growth and learning, hindering software architects from evolving their approaches and benefiting from novel techniques.

The **oversimplification** of maintainability perpetuates the notion that all future-proofing endeavors inevitably introduce complexity. In reality, some prudent design decisions can improve maintainability in the long run.

By overstating immediate pragmatism over future-proofing, Juan's statements neglect the potential accumulation of **technical debt**, which can exacerbate codebase maintenance and extensibility over time. The emphasis on short-term benefits comes at the expense of disregarding the significance of aligning architectural decisions with the long-term objectives and vision of the project. Neglecting future-proofing undermines adaptability, hindering the project's ability to embrace changing requirements and advancements in technology, leaving it vulnerable to obsolescence.

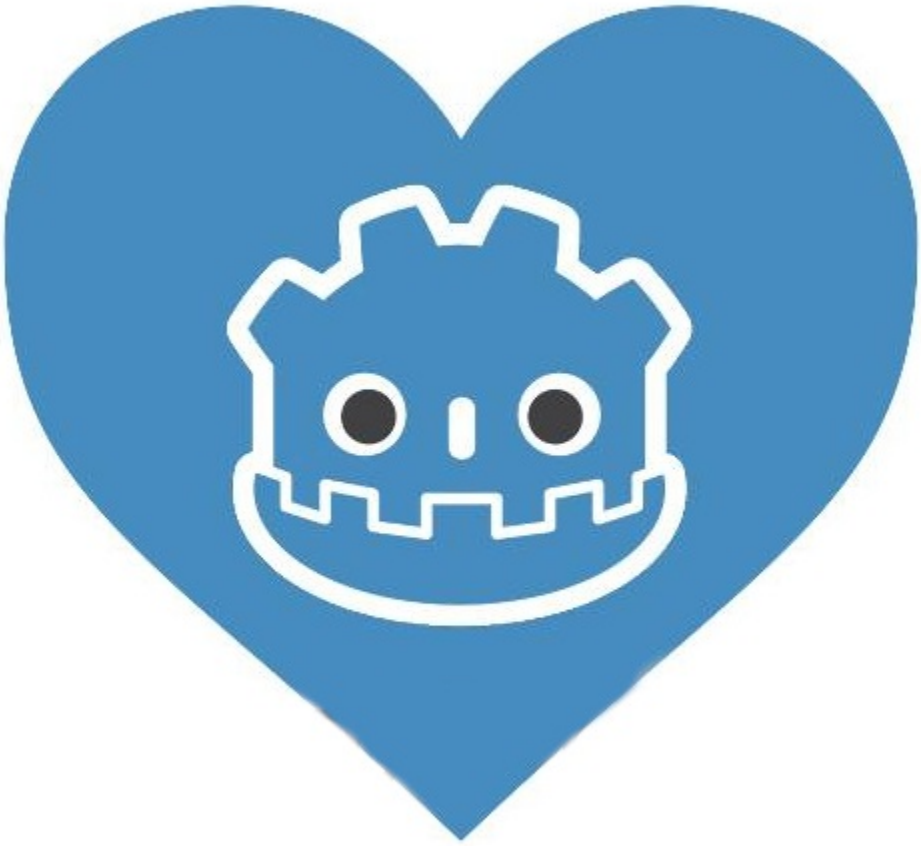
Furthermore, the assertive tone of Juan's statements may inadvertently reinforce **groupthink**, stifling healthy debates and critical thinking among fellow software architects. Encouraging a diversity of design philosophies is crucial for fostering innovation and growth within the community.

Overall, Juan's statements can be interpreted as excuse-making. He finds himself constantly needing to rationalize his **carelessness** under the guise of pragmatism.

Love Bombing

You may read that Godot's development is driven by passion and out of love for those who make games. Participating in an community such as Godot grants a feeling of belonging. Yet this "love" and attention is exactly what describes a cult as well.

Those who find themselves in the presence of cultic groups often perceive only what the leaders intend for them to see. This is achieved through the manipulation of propagandist techniques, such as employing "The Big Lie"[1]. This strategic approach ensures that individuals remain unsuspecting and don't anticipate the deception that is taking place.



The reason why anyone can be roped into a cult is because humans as a species are social and we all have a powerful desire to belong to a group. It can be our greatest strength and greatest weakness, and in the case of cults, it gets taken advantage of.

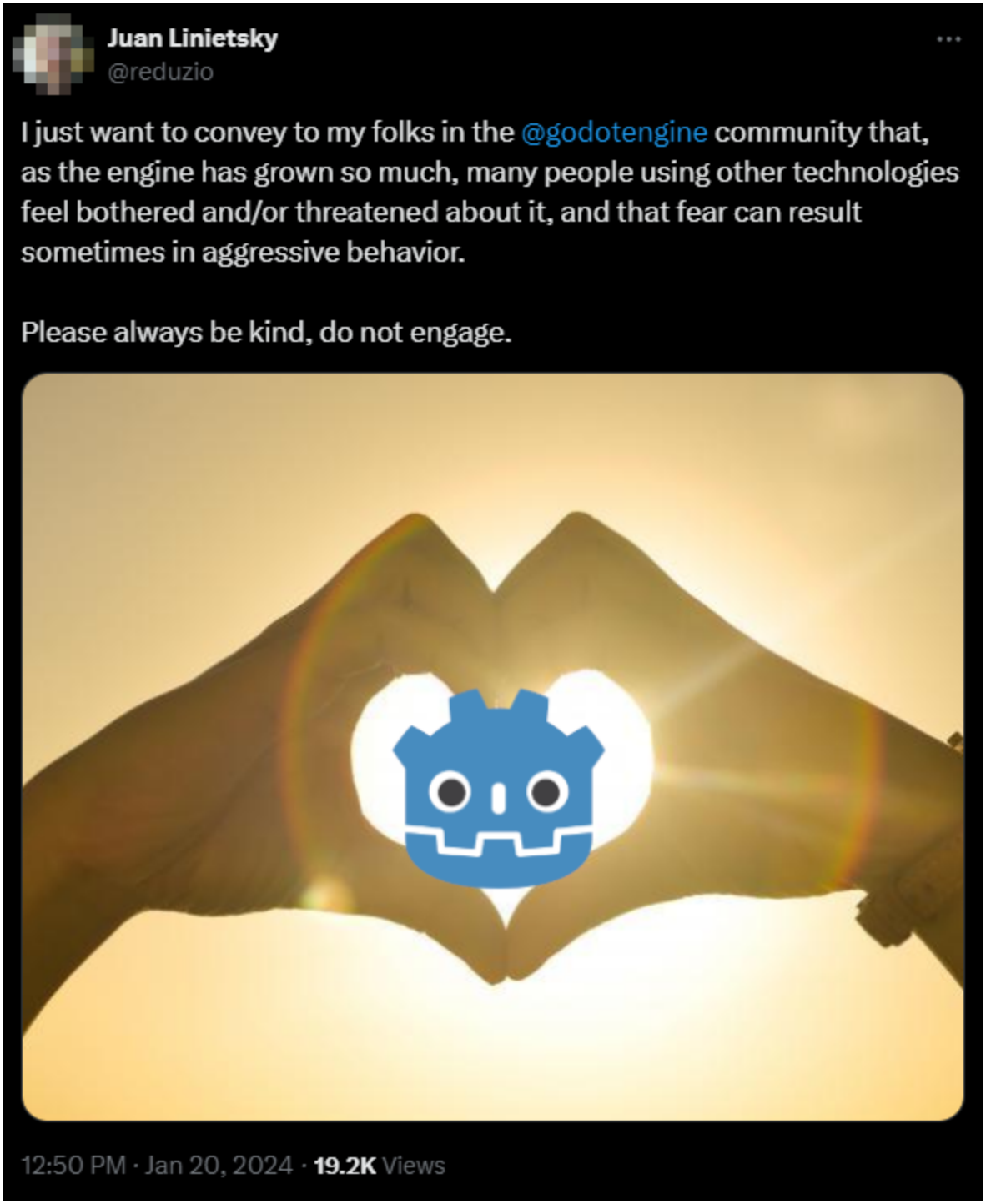
You'll read phrases like "preaching gospel of Godot" from those who donated to Godot financially[2]. Godot followers repeat lead developer's propaganda delivered via articles such as "As an Open Source project, Godot is more than a game engine"[3], so they feel obligated

to spread the word about Godot, which is one of the reasons that makes Godot ideology viral.

There are various nuances and subtleties that might not be immediately evident. Let's take a deeper dive into the so-called "love" displayed by Juan and the Godot community.

Kindness and Coercion

Let's take a closer look at the key messages from Godot's cult leader which eloquently demonstrates this love bombing technique, *emphasis mine*:



> *I just want to convey to **my folks** in the @godotengine community that, as **the engine has grown so much**, many people using other technologies feel bothered and/or **threatened** about it, and that **fear** can result sometimes in **aggressive** behavior.*
>
> *Please **always be kind**, **do not engage**.*

Analyzing Juan's message reveals the following:

- **In-Group Language**: The use and combination of “my folks” and “Godot community” creates a sense of *belonging* and unity among waiters for Godot. This reinforces an environment where people within Godot feel a special connection to Juan.
- **Big Lie of Growth and Power**: By stating that “the engine has grown so much,” Juan is implying that the community or ideology is becoming increasingly influential or *powerful*. This can instill a sense of *pride* in the followers and reinforce their commitment to Godot and keep waiting for its so-called “inevitable” success.
- **Creating an External Threat**: Juan introduces an unreal *threat* by suggesting that people using other technologies feel threatened by the purported success of Godot. This reinforces the *us-versus-them* mentality, making followers to be wary of outsiders. This framing sets up an external threat, making it appear as though Godot’s purported success is causing discomfort or fear in outsiders. This portrayal creates a sense of *vulnerability* among followers, reinforcing the idea that they need protection and solidarity within the community.
- **Fear Manipulation**: Juan suggests that this *fear* can lead to *aggressive* behavior. This is an attempt to control the narrative and discourage followers from questioning or criticizing Godot’s beliefs or practices. Waiters for Godot unconsciously link any criticism with aggression, believing they should steer clear of it to gain acceptance in Godot.
- **Directive for Followers**: The instruction to “always be kind” is a subtle *coercion* tactic. It positions Juan as a moral authority. The directive to “not engage” is an attempt to *isolate* the followers from external influences and differing perspectives. By following these directives, waiters for Godot come to believe that they can “shield” themselves from a threat that is, in fact, purely imaginary.

Cult members often become highly dependent on their leaders for emotional and psychological validation. In Godot, this dependency is largely developed through love bombing, which initially overwhelms members with affection and approval, creating a powerful bond. In the IT age, one way to do this is through the use of memes. This example shows how people **seek approval and validation**. A Godot user asks their leaders if they are proud of them:

Rémi Verschelde reposted

The Water Museum
@WaterMuseum_

Are you proud of me @godotengine

3:29 PM · May 20, 2024 · **16.4K** Views

4 6 189

Godot Engine · May 20, 2024
@godotengine · **Follow**
Replying to @WaterMuseum_

The Water Museum
@WaterMuseum_ · **Follow**

This is not the only example of such behavior. To outsiders of this cult, this may be interpreted as a joke, but such behavior has its psychological explanation. To some extent, they may also treat it as a semi-joke, but often the purpose of jokes is to communicate things that may not be socially acceptable, such as worship:

Godot “asks” people not to worship them, while at the same time claiming to have “supernatural” powers. What Godot does is an attempt to downplay any perceived authority and to hide Godot’s **hierarchical structure**. They always “joke” about these things. The added 💙 emoji further diminishes the seriousness of the issue and discourages its critical examination, such as this one.

Despite the downplaying of worship, they still cannot give up the idea that they possess “supernatural” qualities, which is inconsistent with the request not to worship them. They cannot deny that they have any **superiority**, since Godot’s existence depends on these grandiose claims.

The phrase “sharing the love” is again a classic example of love bombing. Encouraging mutual support and presenting the community as the *key* to Godot’s success is a typical cult strategy designed to cultivate **a sense of belonging**.

This builds loyalty and thus strengthens people’s devotion to Godot in exchange for the feeling of being part of something. So those waiting for Godot end up inadvertently serving the leaders’ personal agenda, and something like a pat on the back gets the job done.

Playfulness and Safety

Cults often make their environment look and feel safe, welcoming, and even childlike. Take Godot’s logo, for example. Many people describe it as playful or even childish, so you’ll often see numerous Godot proposals to change Godot’s icon[4], all of which are rejected by Godot’s leadership.

> Yes or no: Godot's current logo has GOT to go. pic.twitter.com/GE5iqQrDad
>
> — Thomas Brush (@thomasbrushdev) November 9, 2023

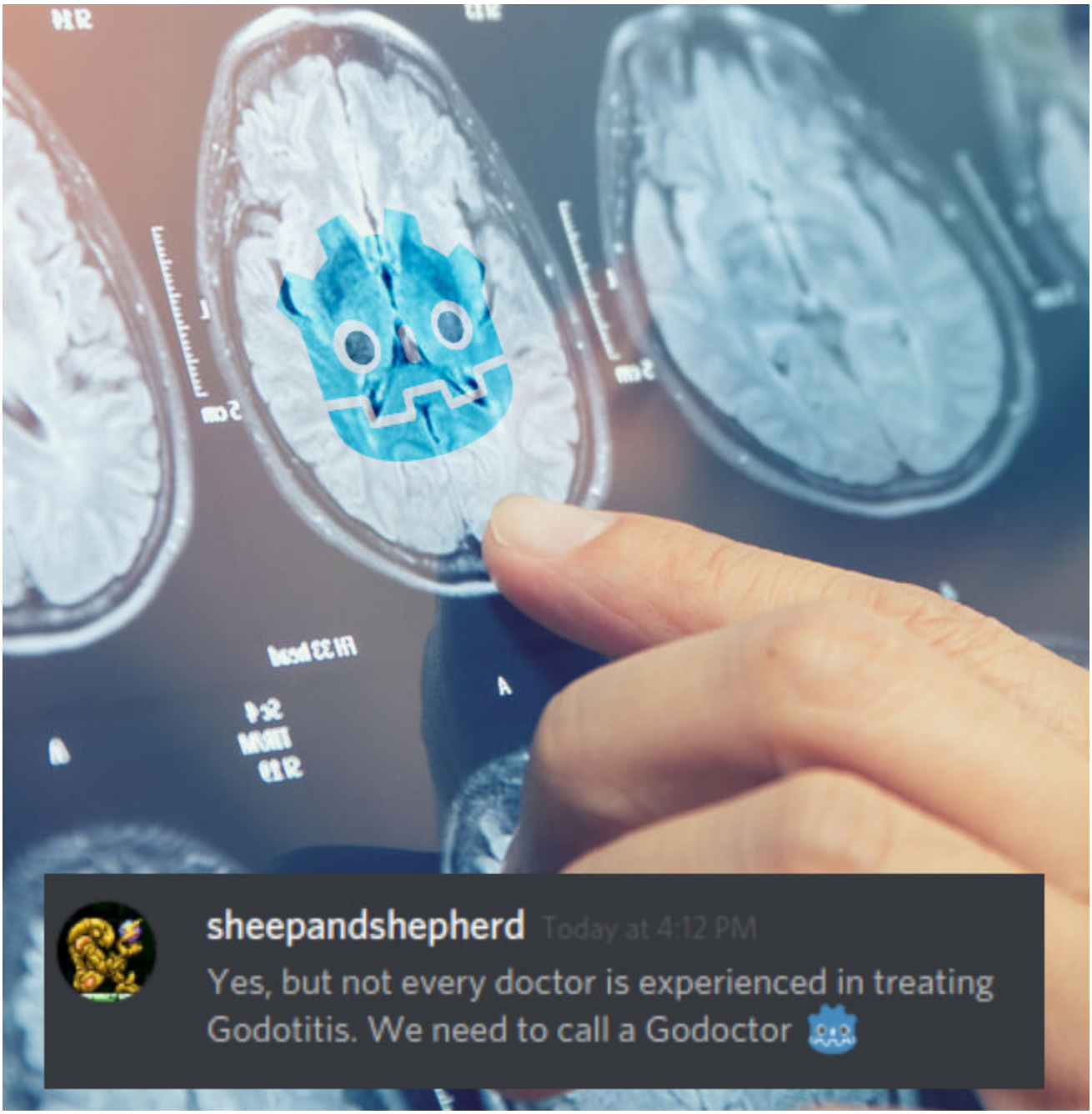
Of course, Godot founders are free to have whatever icon they want to represent Godot, and this child-like look is not a bad thing in and of itself. But let’s see what the leader of Blue Robot Cult has to say about it[5]:

> *In all seriousness, the current Godot logo was created to convey a friendly, welcoming and informal feeling, which is what the engine represents at heart..*
>
> *Our goal is to make the best engine you will ever use (having lots of fun along the way) and make sure it’s as accessible as possible for everyone. We are not here to sell a professional product or make money from you.*
>
> *So, I hope you can accept and enjoy Godot for what it is.*

Oftentimes, cults out there attempt to revert people back to childhood dependence and mindless obedience, so they even encourage this sort of playfulness. As expected, Juan appeals to emotions here, just like all cult leaders do.

Note that when Juan says it’s their goal to “*make the best engine you will ever use*” is not necessarily a lie. Why? Because you likely won’t experience other engines, stuck with Godot

and feel closely attached to it, as some users describe it[6], especially if you've never really used other game engines before. Ironically, this condition is jokingly called "Godotitis"[7]:



This condition may become so severe that Godot's project manager sacrifices vacation time. Other members may spend an inordinate amount of time contributing to Godot's development and maintenance[8]. But this is not all, of course! Because of the atmosphere created by the cult subleaders, the community has even created a character named Godette:

Godot's leadership attempted to mock the criticism that the engine has received, such as claims that it is not professional enough. To this end, they even created a Twitter account[9] and wrote an article for April Fool's Day[10].

> It is the dawn of a new era in game development. Godot has been rebranded to Godette Engine:https://t.co/jtcGBjmuYC
>
> — Godette Engine (@godetteengine) March 31, 2021

However, this attempt at "humor" was poorly received by some groups. Here is what Godot's leadership wrote in response:

> ***Edit***: *We have seen many comments on our April Fool's joke. We strive to make the Godot community, and game development in general, a welcoming place for everyone. By not understanding the various ways in which our joke could be interpreted we have failed in this goal.*
>
> *The Godette character was made years ago. It was not created for this joke. The joke itself was intended to poke fun at* ***a very vocal minority of users*** *[emphasis added] who think that our robot logo isn't "professional", thus the joke was us pretending to be switching to an even more playful character.*
>
> *Having said that: our intention does not matter if so many people interpreted the article in the way they did. We apologize for the oversight. We strive to do better in the future.*

Also note how they describe those people as "a very vocal minority of users," which is not accurate if you look at a poll created by the Godot community on this matter[11].

> Should Godot get a more professional logo?

byu/ElliotBakr ingodot

Rémi often refers to people as a “vocal minority” when they disagree with Godot’s decisions in order to mitigate the criticism that Godot receives, especially from people outside of the Godot community. If anything, it’s actually a case of psychological projection by Godot’s leadership. This is because Godot represents a vocal minority within the game development industry as a whole.

Love Equals Money

The cult leader of Godot is very outspoken when it comes to money as well:

By: Juan Linietsky

4 May 2018

As always, please remember Godot is made out of love, so please consider becoming our patron!

2 November 2019

There are still many pending features that will be worked on in the coming months, so stay tuned to all the new and shiny things! As always, remember that Godot is done entirely out of love for you and the game development community so, if you are not yet, please don't hesitate to become our patron, and help bring the world top quality, free and open source tools for making games.

29 February 2020

As always, all this work is is done out of love for everybody making games, and because we believe the world needs quality free and open source game technology. If you are not yet, please help us by becoming our patron.

28 March 2020

And as always, please remember than our work on Godot is done out of love for you and the game development community, we want to provide you with a top notch free and open source game engine, so you can own your work down to the last line of engine code. If you are not yet, please consider becoming our patron and help us realize this dream sooner.

1 May 2020

As always, please remember that Godot is made out of love for you and everybody else making games. We want to create the ultimate game engine and give it to you and the rest of the world for free, so you can own your games down to the last line of code. If you are not yet, please consider becoming our patron, and help us make this dream become reality.

28 November 2020

As always, keep in mind that we make Godot out of love for you and the whole game development community. We want to make the best and easiest to use game engine, and make it so free and open that you can feel that you made it yourself. If you are not, please consider becoming our patron to help us out!

2 February 2021

As we are approaching an Alpha release, Godot 4.0 is finally shaping up to be a very performant engine for 3D. And, as always, remember all this is done out of love for you and the community. If you are not yet, please consider becoming our patron!

29 March 2021

With the editor work done, I will now go back to working on rendering for the next month to finalize the missing bits and pieces pending in my TODO list. Afterwards, it will be time to start working towards our first Godot 4.0 alpha! And again, remember we do this out of love for you and the game development community so you can have the best possible engine we can make with the same freedom as if you made it yourself.

If you are not, please consider becoming our patron!

Why not just develop Godot Engine independently, then? Why receive donations? What, having passion is no longer enough for Godot to thrive? Sure, Godot cult leaders don't want to make money from you. 🙃

Of course, this is not to say that a project cannot receive funds. The core issue is Godot's hypocrisy: when they say that they can sustain themselves without depending on money at all by having "thousands of contributors," but at the same time, constantly ask you to donate *and* having for-profit companies co-founded specifically on the popularity of volunteer-developed Godot[12]:

> *So, if you care about the future of Godot, and believe in a future with true freedom of game development tools, please be attentive to it and lend us your support when the time comes!* ***Godot is 100% a product of love*** *[emphasis added], so let's make this mutual!*

> So... you spent the $8.5 million dollars already? https://t.co/6yDWxjjDv3
>
> — Andres Hernandez (@cybereality) July 12, 2023

This *love bombing* technique is typical to corporate cults[13], when "love" is immediately followed by "money", along with other factors that we're going to cover in subsequent chapters, where this so-called "love" alternates with intimidation when Godot receives criticism concerning its governance.

The following meme by the cult leader of Godot, Juan Linietsky, is a "TL;DR" of this section (*context: Godot planned exhibiting at GDC, close to Unity and Unreal booths*)[14]:

The heart is on the other side, Juan! 😁

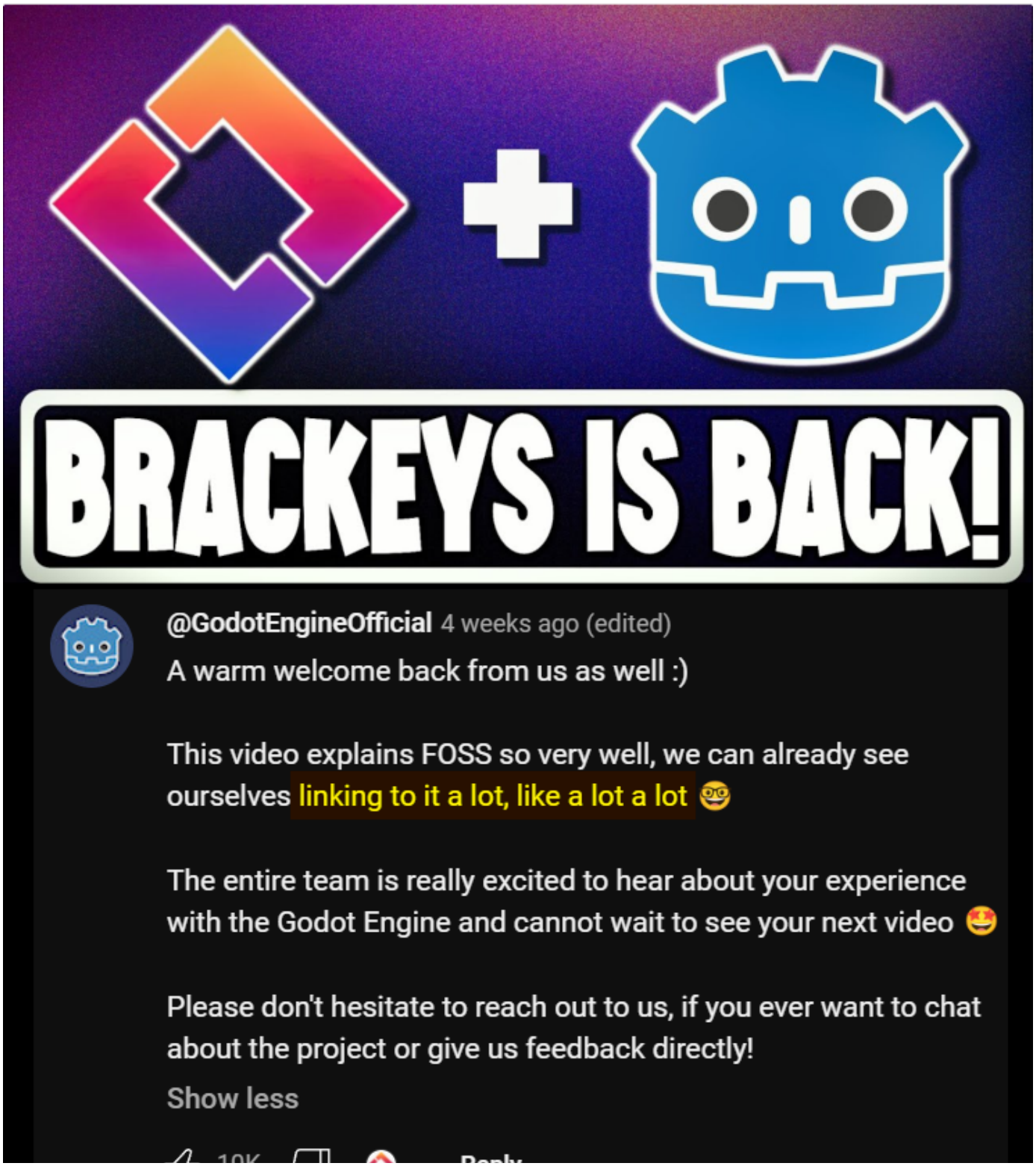
Celebrities

In true cult fashion, Godot leverages celebrity endorsements to bloat its supposed trustworthiness. A standout example is the recruitment of Brackeys, a popular Unity video tutorial creator, whose association with Godot has unfortunately bolstered its standing in the indie gamedev community.

ICSA Articles 1 Articles ICSA Home More

Characteristics of Cults and Cultic Groups

In most cases, leaders and followers in cultic groups believe fervently in the righteousness of their endeavors, and they strive to project this image to the public. Many of these kinds of groups attract intelligent, articulate, attractive people, including celebrities, and the group uses these followers as living advertisements for the group's purported legitimacy.



According to International Cultic Studies Association[15]:

> *In most cases, leaders and followers in cultic groups believe fervently in the righteousness of their endeavors, and they strive to project this image to the public. Many of these kinds of groups attract intelligent, articulate, attractive people,* ***including celebrities, and the group uses these followers as living advertisements for the group's purported legitimacy.*** *[emphasis added]*

In response to Brackeys' comeback after a long absence, Godot welcomed him "back" and said they'd link to his videos "a lot a lot"[16]:

> *A warm welcome back from us as well :)*
>
> *This video explains FOSS so very well,* ***we can already see ourselves linking to it a lot, like a lot a lot*** *[emphasis added]* 🤓
>
> *The entire team is really excited to hear about your experience with the Godot Engine and cannot wait to see your next video* 🤩
>
> *Please don't hesitate to reach out to us, if you ever want to chat about the project or give us feedback directly!*

It's evident that Godot is fixating on creating a buzz than on improving their product. They go to great lengths to attract influencers like Brackeys, using these endorsements to mask their shortcomings. While leveraging a celebrity's endorsement is a normal thing to do, Godot is obsessed with the idea of being noticed by famous figures rather than delivering a quality product.

In the topic titled "Brackeys is back... kind of" on the Unity forum, people have shared mixed feelings about Brackeys' new focus on Godot. Here's a subset of what they said[17]:

> *While a tiny part of me is sad he's making videos about Godot, I'm so glad to see him back. I guess I will learn Godot because I love watching his videos, even if I then never use it.*

> *I will never understand why people liked his videos.*

> *His content had a significant impact, attracting a stream of new users to Unity. He will most likely have a similar impact on the Godot community, offering a clearly defined pathway from Unity.*

> *He found a new engine to use and will start a video series on it. Kudos to him, ive tried Godot a few times and even though its open-source doesnt mean its good. Its not... (my opinion).*

> *He is not a good game dev. He is a good beginner tutorial factory. His tutorials and examples are riddled with inefficiencies, anti-patterns and problems will bite your *ss later, when he is gone and you're on your own with your problem.*

> *Such a kind and lovely chap; learnt so much from his Unity tutorials over the years! Can't wait to watch his next phase!*

Check out this satirical piece on the matter. It poses the ironic question of whether Brackeys is truly back, given that he worked with Unity for years:

Reportedly, it seems that a lot people enjoy watching Brackeys' videos about Unity purely for entertainment, if not for learning purposes. For more information in the context of Unity, read investigation article Waiting and Unionizing for Blue Robot: Are Godot advocates

infiltrating Unity?

The Godot fanbase certainly knows how to keep themselves busy—promoting Godot to the moon and back by associating it with celebrities, popular brands, and just about anything else you can think of. Of course, actually making games seems to be a bit too mainstream for them. A prime example of their creative efforts is the recent buzz around Keanu Reeves, who's set to star in a new Broadway production of Samuel Beckett's "Waiting for Godot." Predictably, fans couldn't resist making a playful connection between him and their beloved engine, despite the obvious lack of any real connection:

> I fixed the headline. :) pic.twitter.com/4Wpgq0kdfx
>
> — MX Cartoons (@MXCartoons) August 1, 2024

Singing

"They all start singing, running around, hugging each other, and praising the Lord," as one cult story tells[18]. Which is pretty typical to cults of all kinds, to be honest!

The following video is a testament that Godot is no exception. Consider this as an innocent extra bonus to all we've described so far, for good laughter!

Perhaps we will witness a Commune of Blue Robot at some point. 🤣

Notice how even this music video is really just an adaptation of the original one. In general, it

is interesting to observe how Godot followers constantly make connections with popular culture, in an attempt to promote Godot in some way, make stronger connections between community members, and most importantly, attract new followers.

Conclusion

Godot is highly focused on recruiting new members and raising funds through love bombing. It's no wonder, since Godot is offered for free, and the entire community operates under the undue influence of Godot's leadership. Members feel an abnormal sense of obligation to promote Godot. While Godot is technically free, the hidden cost lies in its nature as a cultic group.

It is highly likely that individuals who join Godot are seeking external validation, longing for the feeling of "love" or the less apparent sense of "coziness" that comes from returning to a family-like community after a challenging day at work. This community welcomes them unconditionally, as long as they never criticize leadership and their community, of course.

Frankly, Godot operates in a similar way scams do. When you are deep in a scam, you don't want to feel scammed, you don't want to admit that you got scammed, so you tend to get scammed further in the process of not admitting that you got scammed! Many followers simply don't realize that they are trading unpaid endorsement of Godot for a superficial sense of "love" and the need for it cannot be fulfilled this way.

If you are a follower of Godot reading this, it's important to understand that you don't truly "love" Godot. Instead, you are drawn to the community associated with Godot, or more accurately, the narcissistic facade of it. It appears as a welcoming community that you desire to be a part of. However, the truth is that we can already experience these feelings by looking within ourselves, as they are inherent to our human nature. There is no need to rely on the approval of a community to feel good about yourself.

References

[1] The Big Lie - Changing Minds.

[2] Godot: Why Open Source is Important - By "The Omniblog".

[3] As an Open Source project, Godot is more than a game engine - By Juan Linietsky.

[4] New Logo proposal for the Godot Engine 4 !!! - Godot proposals, GitHub.

[5] Juan Linietsky on Godot logo - Twitter.

[6] Why Godot doesn't let me move to other game engines? Wishful thinking... - Godot Q&A.

[7] Akien was supposed to be on vacations - Godot, GitHub.

[8] Calinou doesn't sleep - Godot, GitHub.

[9] Godette Engine - Twitter.

[10] Godot has been renamed to Godette Engine - By Juan Linietsky.

[11] Should Godot get a more professional logo? - Godot, Reddit.

[12] Juan Linietsky about Godot's financial future - Twitter.

[13] Am I Working for a Cult? 5 Signs You Are Working in a Corporate Cult and 7 Tips to Avoid Joining One - Damon Baker, Founder & CEO at Lean Focus.

[14] Juan Linietsky GDC vibes - Twitter.

[15] Characteristics of Cults and Cultic Groups - By International Cultic Studies Association.

[16] The Future of Game Development - Comment by Godot Engine official YouTube Channel.

[17] Brackeys is back... kind of - Unity forum.

[18] Wild Geese - Wild Geese, ICSA Today, Vol. 9, No. 1, 2018, 11-13.

Recruiting

Godot devotees actively recruit new users from all corners of the internet, tirelessly striving to expand the community and encourage more people to use Godot. Their tactics range from subtle questions like “Have you heard about Godot?” to more aggressive approaches, often employing memes to spread the word:

Made this quick poster for my horror game
byu/gandpop ingodot

Ex-moderator of Godot’s Discord server talks about her first interaction with Godot’s community[1]:

> *I was streaming regularly on Twitch then, about 6+ hours/day building my game. I hadn't yet joined the Godot community, and Godot's 3D was seemingly fine for the initial stages of my project. Everything was going grand...*
>
> *Then one day, as I was streaming on Twitch. The original admin of the Godot Discord found my Godot stream... and they did an at-everyone on the Godot server and my Twitch channel was flooded with hundreds of Godot users pouring in...*

> 5/ I was streaming regularly on Twitch then, about 6+ hours/day building my game. I hadn't yet joined the Godot community, and Godot's 3D was seemingy fine for the initial stages of my project. Everything was going grand...
>
> — LillyByte (@LillyByteGames) March 28, 2022

The above words perfectly describe how cults recruit new followers. Some even describe this behavior as that of the Jehovah's Witnesses of game engines, if you look at threads like "About Godot and haters"! [2] In addition, the title itself shows an *us-versus-them* mentality that manifests itself as fanboyism. Here's one notable reply from this thread:

> Comment
> byu/kaukamieli from discussion
> ingamedev

Godot's developer community also shows this kind of preoccupation. For example, you'll find dedicated public channels like #new-contributors in the Godot Contributors Chat, with other channels hidden from the public access unless you have a registered account. Godot's documentation is quite extensive when it comes to specifically contributing to Godot's development, as opposed to a development philosophy, which doesn't exist in Godot.

Even when Godot receives donations and funding, Godot's leadership is allergic to hiring independent professionals to work on Godot. Here's what Juan says[3]:

> *... the problem is that hiring someone who has not contributed to the project or has no experience on a given area is **always a risk** [emphasis added] because they may not be able to perform as expected and/or may need assistance from other contributors, taking away their time instead.*

In reality, even if Godot manages to attract the occasional professional developer, they must first show unwavering support for the project by never criticizing it for anything, and waste a

significant portion of their lives contributing to Godot's development as volunteers. Who knows, if they become passionate enough, Juan might not have to pay them anything for the hard work they do for free! 😉

Developers with solid experience rarely, if ever, need help from other contributors; what they really need is good documentation. Even if we follow Juan's perverse logic, new contributors also take time away from experienced contributors, who are volunteers. What Juan's words really mean is that Godot's leadership will not allow an outsider to work on Godot, because allowing outsiders with great experience to work on Godot **poses a risk** of exposing Godot's inner cult mechanisms, so the leadership must draw "red lines" to prevent that from ever happening.

In a video titled "Godot Documentary - Some Faces Behind the Godot Engine," in a chapter titled "A Different Way of Working" [4], Juan mentioned something similar to the previous quote.

However, let's shift our attention to address other complementary points, *[emphasis added]*:

> *... **many people in the community** say why don't you do bounties, you put a bounty to make a feature then you select the person that you think that's going to make it and then you pay that person. **But in reality it doesn't work like that**. The problem is that if you have somebody that you pay to work, that person may need help because if you want them to do things the way you believe are the best for the project, like write the code in a certain way, **doing it efficiently** using everything properly, you need to spend a lot of time helping the people do what they need to do instead of working on what you have to do as you can hear **this is a very different operation to what you might expect from a project of this size** there are other open source projects that are dealing with the challenges that Godot is*

> *[facing] but I don't think there are many at this scale yet so **there's still a lot to explore**.*

- The very phrase "*many people in the community*" implies an inconsistency between the actions of Godot leaders and the actual preferences of the Godot community regarding the allocation of funds, particularly the money that Juan has continuously begged the community to donate.
- The general phrase "*in reality it doesn't work like that*" makes Juan's position on this topic seem superior or knowledgeable, completely ignoring the experience of other open-source projects that happily utilize the bounty system, like Stride does.
- The phrase "*doing it efficiently*" implies prioritization of performance, but if you refer to Priorities, Godot doesn't really care about Performance. Phrases like "*if you want them to do things the way you believe are the best for the project*" imply a lack of trust in competence of developers who would be willing to work on a specific area in Godot. Frankly, such rationalizations by Juan imply an attitude of a control freak!
- The phrases "*this is a very different operation to what you might expect from a project of this size*" and "*there's still a lot to explore*" suggest that Godot is unique and requires different management approach. Indeed, cults do require a peculiar approach not typically seen in other healthy communities, and obviously, Juan will never reveal those unethical practices.

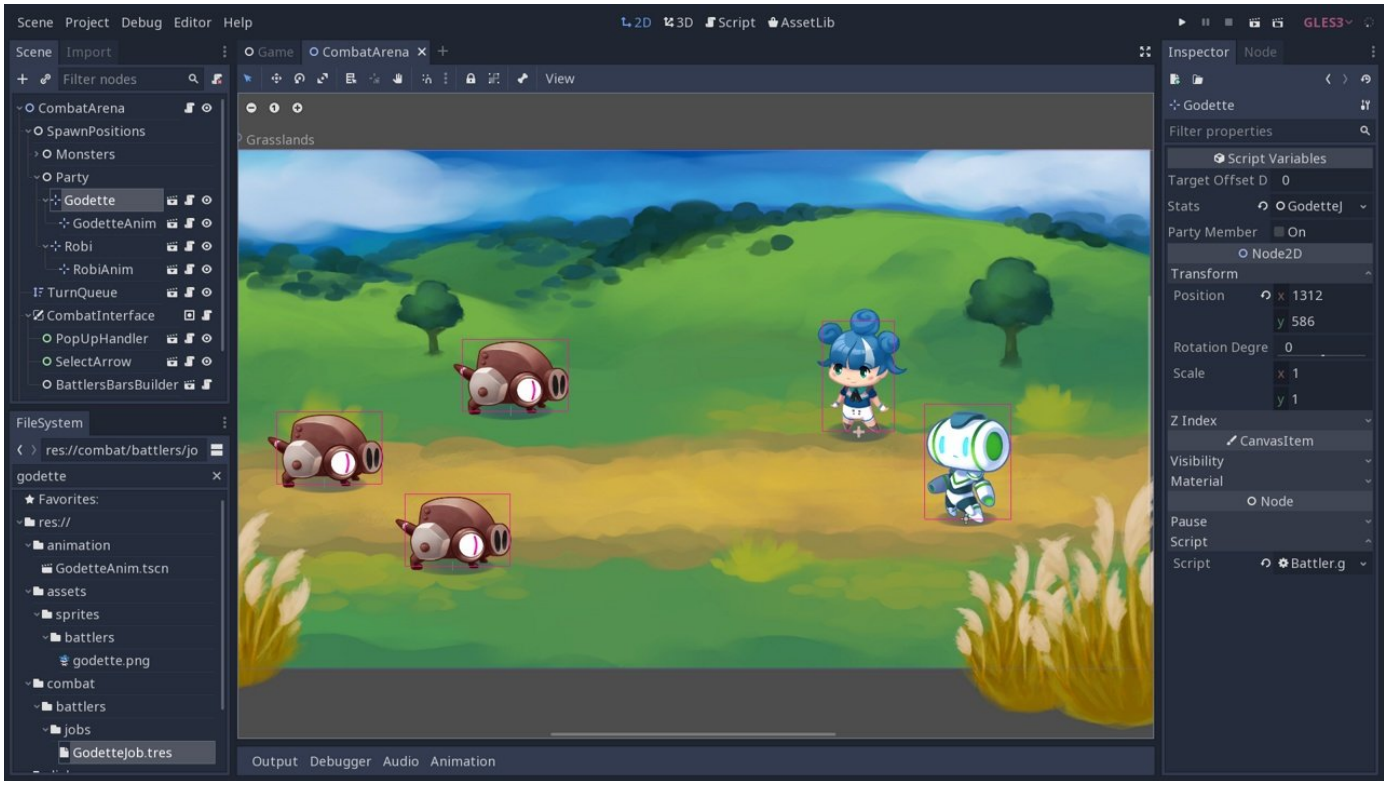
On top of everything mentioned earlier, Godot, the magnificent open-source wonder, enjoys the perks of scholarship programs like Google Summer of Code[5]. This program allows students to toil away on features and bug fixes (with Godot, the emphasis is on the features, of course). It's a marvelous blend that coincides with the bounty system, albeit indirectly, because, you see, it's good ol' Google who graciously provides the stipends.

These stipends are like a magical fixed sum of money handed out to those brilliant souls who successfully complete a coding project for free and open-source software during the summer. And guess what? Godot, being the cool kid on the block, attracts hoards of such students without Juan even having to dip his hand into his own pocket. How do you like that? 🙃

> GSOC means Google Summer of Code and IK means Inverse Kinematics. Google Summer of Code is what sponsored my ability to work on Godot, and Inverse Kinematics is a way to programmatically move a set of bones to a target location 🙂
>
> — TwistedTwigleg (@TwistedTwigleg) August 17, 2021

Dedicating time to assist students, who may often lack the necessary experience to work on features, is not perceived as a hindrance, whereas hiring professional developers would undoubtedly consume the valuable time of contributors, according to Juan. Do you notice a contradiction here? We can conclude that Juan surely aims to target the younger generation of developers instead[6]:

> *An interesting phenomena I see is that, among the younger generations (<25), most pick Godot because It's just far simpler to learn and understand than other software. I think this is quite curious..*



Notice how Juan uses Godette character that we covered in the previous Love Bombing section. Also, Juan hasn't provided any factual data to confirm his claim that most people under 25 years old pick Godot. He then continues to explain why, despite the industry's preference for using modern technology and techniques, Godot is one of the biggest FOSS projects in the entire ~~galaxy~~ world, [emphasis added]:

> *Godot uses **simple** C++ as it was used in early 2000s, standard OOP, **simple** data models, a very **traditional** scripting language, **traditional** concepts such as scene graph, retained mode UI with a late 90s design, a retro editor experience reminiscent of **Visual Basic**..*
>
> *And stuff such as ECS nowhere to be seen, preferring to use more **traditional** hand crafted data oriented code for optimization.*

> *As Grug would say, all things that nowadays big brained **developers consider dated and wrong**..*
>
> *Yet, here we are, **against all odds, fastest growing** piece of game technology and one of the 10 biggest FOSS projects in the world..*

To believe that preferring traditional, and even retro techniques would lead to unprecedented success would be naive. Therefore, this "secret sauce" of success must be something else... What could it be?! An interesting fact: the phrase "fastest growing" is commonly associated with religions or cults as well. 😉

Juan mentions Grug[7], referring to "The Grug Brained Developer" which Juan is likely fond of. The article could be seen as a collection of software design principles aimed to reduce *complexity*, with a particular humorous "Grug" style, "*grug brain developer not so smart*," as its written in the introduction. As you recall, Juan said Godot has no development philosophy. Yet, "against all odds," Juan manages to promote Godot for it to become popular enough among the younger generation, who view it as a sign of success and adoption in the game development industry, and they choose to believe the inflated authority of Juan, a "guru" of game development.

Studies on cults suggest that cult leaders target younger individuals, as they are more susceptible to cult indoctrination. In particular, the simplicity that Juan aims to offer provides a sense of relief from analysis paralysis. Quote from one article titled "The Appeal of Cults to Teens" confirms this[8], *[emphasis added]*:

> *Today's society can be described as a tyranny of options with multitudes of opportunities. Young people are faced with overwhelming possibilities for their future, which often results in **choice paralysis**.*
>
> *Throughout the ages, we can see a clear pattern that demonstrates how cults flourish at times of **existential questioning and uncertainty**. Our globalized, digitally connected culture offers numerous options for work, hobbies, sexuality, relationships, diet, aesthetics, and spirituality. Our society also places immense importance on the notion of self and creating a personal brand.*
>
> *Therefore, a charismatic cult leader can be an attractive proposition, as such a person narrows the playing field considerably and helps people make decisions. Dr. Adrian Furnham states that **humans crave clarity and find solace in the absolute answers a cult can offer**. The clear, unwavering messages of a **cult leader offer a simplicity** that is nearly impossible to find in everyday life.*

If you read the book carefully, Juan mentions concepts like *clarity*, *simplicity*, and *ease-of-use* a lot. Just think about it, Juan has an undue influence over a big portion of younger developers that experience the choice paralysis. They are full of potential, and Juan definitely exploits their enthusiasm (ignited by Juan's grandiose claims) to further promote Godot and most importantly, attract corporate sponsors that way. Speaking of ethics, Juan asks the public whether it makes sense to provide official certification of Godot contributors[9], *[emphasis added]*:

> *Question to you all, given there is an increase in companies wanting to do Godot hires (and many times they ask us and we don't really have a place where to officially post), what would you recommend us to do? Use existing sites,* ***create a job board****, etc?*
>
> *Additionally, companies may be looking for some sort of proof of experience. I think for a lot, linking to your projects on Steam/Play/itch/GitHub/etc is enough. So, would it make sense for us (or third parties) to also offer some sort of* ***"official" certification?***

Allowing this "idea" to create a job board and provide official certification of contributors to appear in the first place goes against the fundamental notion of Godot being developed by volunteers and being allegedly independent of the influence of corporations! Juan's question could also be seen as having a primarily propagandist purpose, as he poses a loaded question assuming something that may not be true, such as "*an increase in companies wanting to do Godot hires*" in this case.

Conclusion

We have highlighted several contradictions and questionable practices within the Godot community and its leadership. The first interaction described by the ex-moderator reflects a cultish behavior, with followers actively seeking new users. The focus on new contributors and the lack of emphasis on development philosophy raise concerns about the community's priorities.

The reluctance to hire professionals and the emphasis on relying on volunteers may hinder the growth and development of Godot. The reasoning provided by Juan, the lead developer, is questioned, as experienced developers often require good documentation rather than assistance from other contributors.

Dismissing alternative approaches like bounties adds to the perception of a management style that may not align with industry standards. Additionally, the suggestion of creating a job board and offering official certification of contributors is inconsistent to the claims of

Godot as an independent, volunteer-driven project. It raises concerns about potential exploitation of volunteers and the influence of corporations in the community.

Because of this cult, people tend to end up spending their lives promoting Godot and working on Godot for free, not necessarily working on game projects made with Godot. You may find yourself contributing to the engine rather than working on the actual game, and slowly abandon the development of your own project. It's fine to help others, but not at expense of your own goals, unless helping others is the ultimate goal in your life. However, when we talk about cults like Godot, the process of helping others becomes a slippery slope, and this help rather promotes the hidden agenda of cult leaders.

References

[1] LillyByte's first interaction with Godot community - Twitter.

[2] About Godot and haters. Don't believe everything they say - Godot post at Reddit.

[3] Juan Linietsky on hiring developers - Twitter.

[4] Godot Documentary - Some faces behind the Godot Engine - By Emilio Coppola, YouTube.

[5] Google Summer of Code 2022 ideas list (GSoC 2022) - Godot, GitHub.

[6] Juan Linietsky about younger generation picking Godot - Twitter.

[7] The Grug Brained Developer - The Grug.

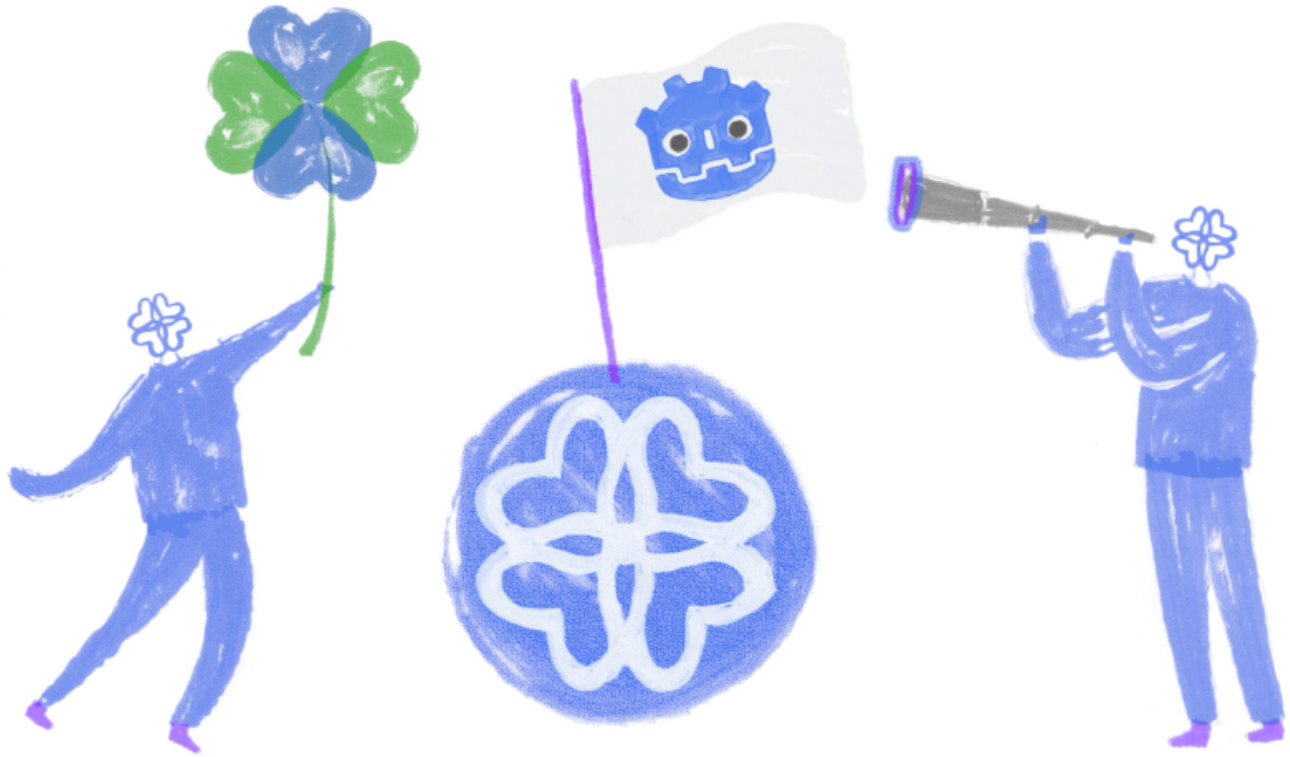
[8] The Appeal of Cults to Teens - Heather R. Hayes & Associates.

[9] Juan Linietsky on certificating Godot volunteers - Twitter.

Elitism

Cult leaders of Godot Engine show signs of toxic elitism. Cults want to show that they have the ultimate answer to life, so toxic leaders tend to present themselves as having the wisdom to make good decisions and have a great impact on society. Of course, the true purpose of cults is hidden behind the pretense of good intentions.

Special Mission



Images from official W4 Games website.

As we have covered in previous chapters, Godot's leadership doesn't really define its mission and vision, because those aspects manifest via other outlets that are closer to the real purpose of Godot, such as the commercial company **W4 Games** created by Godot's leadership, co-founded by Juan and Rémi. Here's how they define their mission[1], *emphasis added*:

> *We want to help the video game industry **reclaim their control** on the technology powering their games, and **reverse a dramatic trend** where they have to rely on proprietary*

> *solutions from an **ever-shrinking** number of vendors.*
>
> *Our aim is to **make Open Source the industry's first choice** by ensuring that both enterprise and independent developers can use the community-developed Godot platform with **peace of mind**.*

Cults claim to have a special mission, such as saving humanity, so often that mission is messianic or apocalyptic. The mission defined by Godot's leadership is quite analogous and pretty much reflects this, with appeals to emotions using words such as *dramatic*, *ever-shrinking*, *peace of mind*.

Perceived threat as a unifying cause:

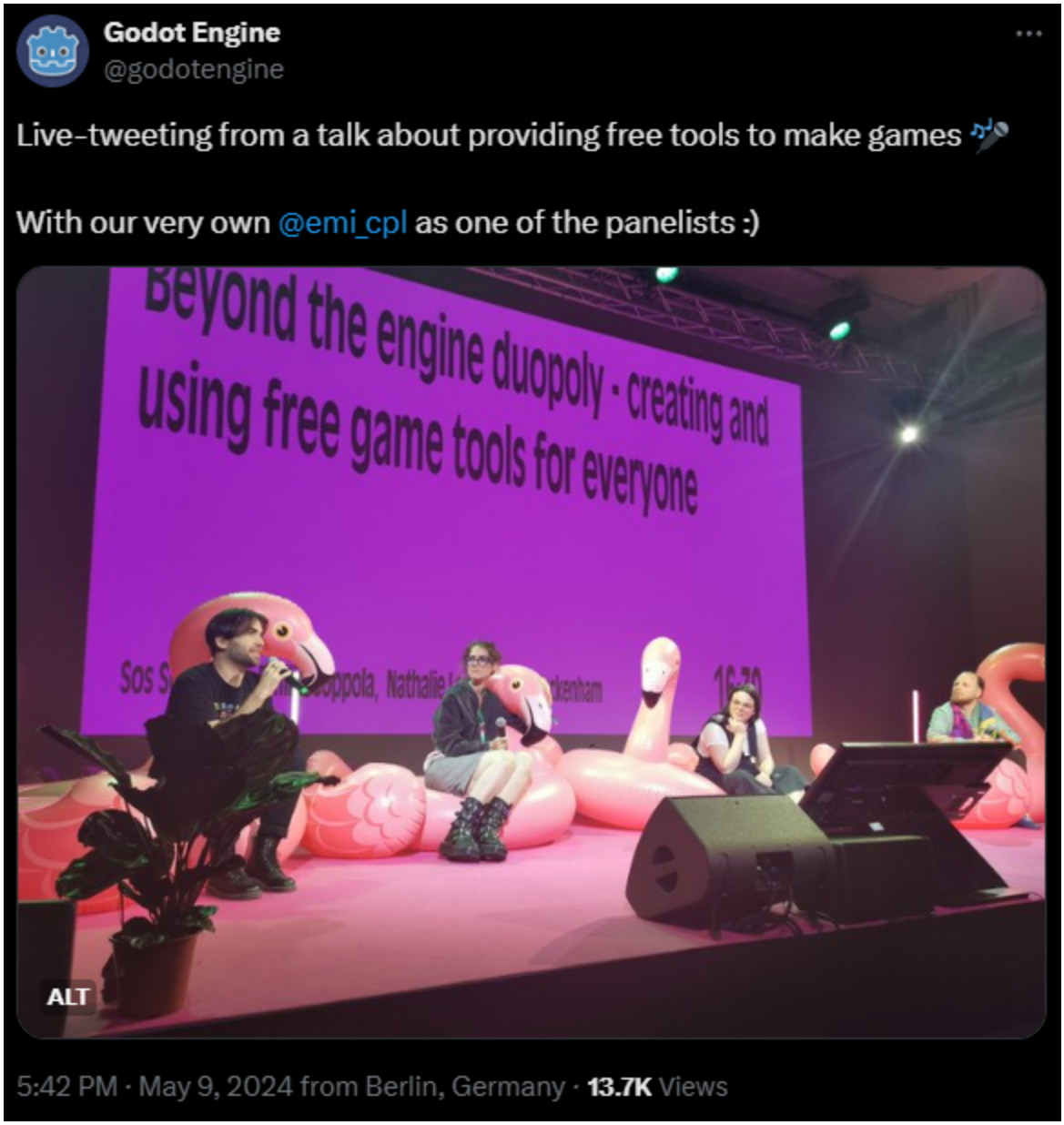
- There's an *assumption* that there was control in the first place. By whom and over what exactly?
- There's an *assumption* that somehow, video game industry is losing control over the technology, so it must be restored.
- Even if above assumptions are true, it's not clear who and how loses that control. The video game industry is not a monopoly.

Proposed solution to the _perceived threat_:

- There's an *assumption* that by making Open Source the first choice, developers can somehow reclaim this control. But *everything* requires maintenance regardless.
- Open-source and proprietary solutions can co-exist. Proprietary does not create a threat to Open Source.
- A private commercial company **W4 Games** will certainly help to reclaim control. 🙃

Just like with non-existent development philosophy in Godot, there's an *ambiguity* over what exactly constitutes control in the first place, so their mission is quite vague.

Emilio "Emi" Coppola, nominally Godot's executive director and what I humorously call Juan Linietsky's "left" hand, basically replaces Juan in public relations these days. His job is to attend various game development conferences and talks to spread the word about Godot:



@AMazeFest: "Beyond the engine duopoly - creating and using free game tools for everyone."

The Godot cult leads people in the game development community to adopt a toxic us-versus-them mentality. The so-called “engine duopoly” is pretty telling here. “Duo” stands for “two” so what exactly are those two engines? Why not mention them directly, right? Could they be referring to these two engines? 😉

Not to mention the flamingos. Whatever message Emilio is trying to convey here, run! 😂

The intention of “free tools for everyone” is fine. The problem is with wolves in sheep’s clothing. When talking about Godot, they aim to overthrow that so-called “duopoly” and replace it with their own potential oligopoly called the “open” Godot ecosystem via W4 Games, Ramatak, etc.

Godot is like a magnet for people who have very anti-corporate sentiments, and they exploit

those sentiments for their corporate benefit by shielding themselves behind the Open Source Initiative. This is especially evident from public talks of Juan Linietsky, who claims that Godot can spearhead the game industry as an "open" ecosystem, and let other Open Source game engines like Open 3D Engine or Bevy come later. 🤡

Godot as an Open Ecosystem – Juan Linietsky (GDC 2023)

Rest assured, if game developers end up inviting or attracting Godot representatives to their events, it could suggest that the community shares certain cult-like characteristics as well. Alternatively, there's a risk that they might unintentionally adopt such mentality, merely from prolonged interaction with cults like Godot, reflecting the traits of Toxic Cult.

Exalted Status

Here's an example when Juan talks about integrating a rendering backend into Godot[2]

> *Guys, thanks a lot for your enthusiasm, but I am the one doing the rendering work in Godot, not you. I've been working on 3D rendering for 25 years so, if I am telling you that things as they are now are optimal and BGFX will just stand in the way to being productive I hope you believe me.*
>
> *As always with Godot, you are free to make your own renderer with BGFX, show me that it's flawless, has better better performance, works better than mine (of course while supporting the full Godot feature set) and uses less resources and code to prove me wrong.. as well as commiting full time over the next years to maintain it.*
>
> *If you want a revolution, begin with it yourself. If you want to prove your ideas, invest in*

> *them because words are free.*
>
> *If not, keep pestering all you want and it will be ignored.*

Hopefully, you see why this kind of behavior is problematic. Rhetoric can be simplified down to this:

- *I'm the one in charge, not you*.
- *I've been working on this for 25 years, so I know better than you*.
- *I know this, so you must believe me*.
- *If you don't agree, I will ignore you*.

Juan said: "*if you want a revolution, begin with it yourself*". We follow Juan's recommendation! Juan pushes professional developers away from Godot because he is "*the one*" in Godot, a cult leader that must never be questioned if someone wants to get his trust.

Other members in that thread say that others weren't pestering Juan at all. However, when Juan feels like his ideas get attacked, he will go into denial and gaslighting mode. At the same time, Juan will suggest that saying "*do it yourself*" is a very rude behavior to *users* of Godot outside of developer community when he "explains" the governance model of Godot[3]:

> *While this is true, it is also true that answering people "do it yourself" is very rude. This is not how we encourage people to work on Godot.*

Which is yet another example of hypocritical behavior.

Pretension

Cult leaders of Godot, namely Juan and Rémi, also express resentment when new open-source or source-available communities appear, and the mere existence of promising projects create a threat to Godot in the eyes of Godot cult leaders, again, showing signs of **us-vs-them** mentality.

Public Relations

Here's just a couple of examples:

NeoAxis

> *Congrats to @neoaxis for opening up their engine source code today. As always, if your goal is to inspire confidence in others to help you and contribute to your project, I will again advise to use an OSI approved license and not your own.*[4]

Defold

> *As I stated previously, I think it's extremely important that game engines allow source access and this move is always welcome, but this is not "Open Source" by the standard OSI definition: opensource.org/osd. This is just making the source code available.*[5]

Open 3D Engine

Juan:

> *I like that this time it seems to be majorly FOSS so this is very welcome, although to be honest all these companies throwing huge amounts of of money at technology nobody wants to use is a bit disheartening.*[6]

Rémi:

> *Because now with the current name, one does wonder why well-established open source 3D tools are not even mentioned by the "Open 3D Foundation".*
>
> *No Godot, no Blender, no ThreeJS, no Stride...*
>
> *It's just a name but it's one that seems to negate everything that existed before it.*[7]

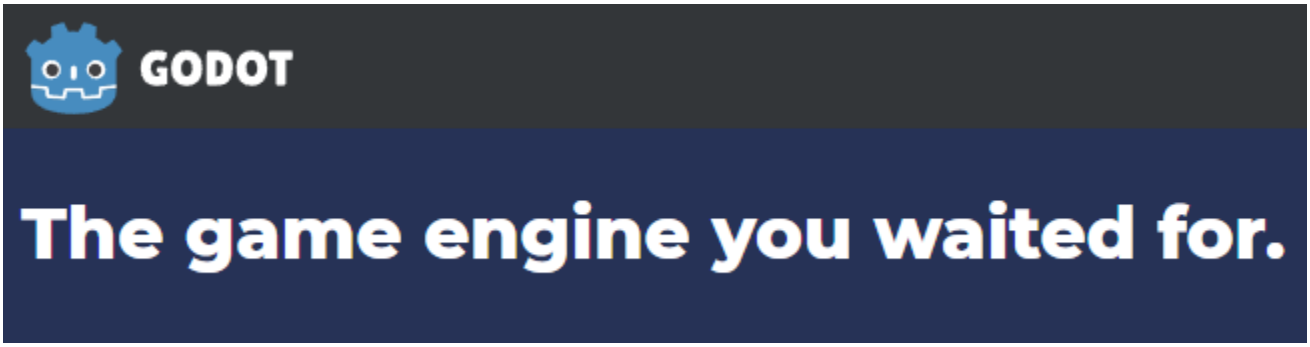
Projects are free to use whatever license they want, and Juan and Rémi are free to influence such decision, just like everybody else. However, what's mind-blowing is Godot cult leaders' obsession with projects not adopting pure Open Source™ terms.

For example, some users of Defold were particularly displeased by Godot's attacks towards the Defold project. Here's a quote shared by a Defold user[8]:

> *You probably noticed on Twitter, Remi (the unofficial vice-leader behind Godot) was the first and most vocal person harassing the Defold guys about the licensing nonsense, "it's sad to*

> *not be opensource", and bragging about his number of contributors, and Juan (the great cult leader) was bashing them from his personal account too. They are not good people. Ugh, they even got a $20k grant from the Mozilla foundation specifically to improve Godot's use of web technology. Juan said that he used it to pay for his living expenses and... a year or so later, Godot still didn't even have functioning web builds. How that wasn't considered fraud, I don't know.*

This toxic attitude becomes more apparent when we take Godot's frankly disrespectful reaction to O3DE, such as Juan's "*throwing huge amounts of money nobody wants to use*", or Rémi's insinuation that the name "Open 3D Foundation" negates everything which existed before it. As if Godot's "*The game engine you waiting for*" doesn't negate everything that existed before Godot. 😉



What can possibly explain such an ambivalent attitude? This is mostly due to projection and desire to dominate. They become envious when investors recognize other projects over Godot, so they use every opportunity to mention about themselves, even when they talk about others. This behavior can be summarized as: "*Please, notice and adopt us! We have the best engine you will ever use*!"

Community

Project manager of Godot, Rémi, once answered a question on how to act as a good open-source community citizen[9]:

> *Always assume positive intent.*
>
> *Know that not everything is written down in policies and that a lot of the development and decision making happens by **consensus among core contributors** [emphasis added], which is based on a shared philosophy and vision. It can take time to be acquainted with those.*

Notice how Rémi explicitly distinguished consensus among *core contributors*, not even *contributors*, but *core* contributors, and definitely not community at large. This suggests that community and contributors are treated differently by Godot leadership. There's nothing wrong with this whatsoever, but take this in mind and read on.

For years, due to the absence of development philosophy and vision for Godot (contrary to what Rémi says), leadership and core developers were expressing disapproval for various features to be integrated in Godot. Because of this, a separate community consensus formed which suggested that Godot needs an unofficial community organization for Godot-related plugins, modules, addons, and even ideas[10]. So, one member took action and created an organization for this purpose, initially called **godotengine-community** (which is called **godot-extended-libraries** now). Here's a reaction from the project manager, Rémi[11]:

> *Don't you think it worth discussing with us before creating a parallel "godot-community"? How is the official "godotengine" org not the Godot community?*
>
> *If you want to create an unofficial Godot platform for third-party work and discussions that would happen upstream of official platforms, that's fine, but then godot-community is a very misleading name.*
>
> *You're probably thinking in terms of typical company/users relationship, "they" (developers) and "we" (users). But in a project like Godot things don't work like this,* ***it's only "we"*** *[emphasis added]. Developers and users work together on the same platform, for the same goals.*

The concern raised by Rémi that "godot-community" name may be misleading is fine. What's perplexing is the attitude expressed as "*it's only we*". The thing is, in a project like Godot, which shows all signs of being a toxic cult, there's only "we" indeed, because everyone who diverges from the main community due to having a different vision is seen as "them", an enemy. Later, Rémi says that he doesn't agree with this consensus:

> *Well I was in holidays and was not given time to read nor answer this discussion, I'm not sure I agree with the consensus. I'll have to read when I have time.*

When you take everything together, do you see contradiction here? Feel free to re-read this section.

This also shows how consensus is treated in Godot. If Godot leadership doesn't agree with consensus, then nothing is going to happen, going against consensus.

To summarize, when leadership says something like "*get out, we don't need this*", community does exactly this, but when community creates a centralized place for modules, plugins, addons, leadership says along the lines of: "*there's only we, we want to prevent community division*". Therefore, we can also consider this as an **us-vs-them** mentality manifesting as ambivalence, but on the level of Godot community itself.

Copycats

Here's what Juan says to NeoAxis developers trying to convince them to change their license[12]:

> *A custom license also makes impossible to cross contribute with other projects like ours. You are free to take our code, **but we can't take yours** [emphasis added]. The key to Open Source is about credibility based on equal benefit for all parties involved, not just making your code available.*

I'd like to emphasize on the part when Juan hints that they cannot take NeoAxis code. "Why is this relevant?", you may ask. As has been mentioned already, Juan oftentimes says that they like to innovate in Godot, so they prefer *not* to copy code of others according to their words[13]:

> *If it helps, we at @godotengine are more focused at trying to innovate and solving our problems our own way than in copying something else.*

I'd like to clarify that using the work of others is totally fine. But if we take a concrete feature which Godot allegedly presents as unique, innovative or in-house, such as 2D physics, this is certainly not true. Godot developers take Box2D-Lite source code[14] and adapt it in Godot[15], which is equivalent in style, structure and algorithm. If you study Godot's source code further, you'll also notice a lot of code was actually taken from Box2D.

In a Godot proposal which suggested to integrate Box2D, Juan said that it doesn't make sense to integrate, and never mentions the fact that Godot's own 2D physics is copied from Box2D[16]:

> *As far as I know, I don't think Box2D or Chipmunk have everything required for what we're doing.*

This dishonesty serves as an example of how Godot developers plagiarize the work of others. At the time of writing this book, the code taken from the Box2D-Lite repository was not properly credited in Godot's source code, thereby violating the requirements of the license and essentially infringing upon the Box2D-Lite license terms. However, if Godot were to give credit to the Box2D author, it would undermine Godot's artificial image of being innovative.

Therefore, it is not surprising that Juan expresses resentment towards being unable to incorporate code from new game engines that have emerged. How many other features that we currently observe in Godot were shamelessly appropriated in a similar manner? This remains a subject for further investigation.

This "wannabe" attitude manifests in other ways, like using game trademarks to promote Godot in official capacity in news articles (reference to Half-Life game)[17]:

If this were to be created as a meme by the community, it would be acceptable. However, Godot leadership engages in these practices *officially*. They cling to others' success by associating Godot with well-known figures. No self-respecting organization would engage in any of these actions. It is a form of psychological manipulation inspired by marketing techniques that are considered unethical, to say the least.

Conclusion

Godot leadership's stuck-up attitude and hypocrisy reinforces an idea that Godot greatly suffers from the NIH syndrome, as we have covered in Not Invented Here section. So, this approach is less about pragmatism and innovation, but more about deeply-rooted resentment and desire for control, while at the same time making Godot look grandiose on the surface.

References

[1] W4 Games - Mission - W4 Games website.

[2] Juan Linietsky on integrating a rendering backend - Godot, GitHub.

[3] Saying "do it yourself" is very rude - By Juan Linietsky, Twitter.

[4] Juan Linietsky on NeoAxis - By Juan Linietsky, Twitter.

[5] Juan Linietsky on Defold - By Juan Linietsky, Twitter.

[6] Juan Linietsky on O3DE - By Juan Linietsky, Twitter.

[7] Rémi Verschelde on O3DE - By Rémi Verschelde, Twitter.

[8] Juan was bashing them from his personal account - Quote shared by Pawel, Twitter.

[9] Rémi Verschelde on being a good open-source citizen - Twitter.

[10] Allow to discuss and present ideas freely for core and/or plugin development - Godot, GitHub.

[11] Rémi Verschelde on Godot community - Twitter.

[12] Juan Linietsky's suggestion for NeoAxis's license - Twitter.

[13] Juan Linietsky on innovation in Godot - Twitter.

[14] Box2D-Lite physics code sample - Box2D-Lite repository.

[15] Godot 2D physics code sample - Godot repository.

[16] Box2D physics engine implementation ? - Godot proposal, GitHub.

[17] Versioning change for Godot 3.x - By Rémi Verschelde.

Groupthink

The rhetoric, narrative, and underlying mentality of the Godot community can be encapsulated in the following phrase: “You should not criticize Godot.” This satirical video illustrates this social dynamic:

> i am expecting 5 death threats, at least 2 "you are mentailly ill" and one "gobot is open source 😡😡"
>
> — Abruh (@abrasivetroop) February 14, 2024

In many cults, there is a strong emphasis on maintaining a unified belief system, and any challenge or criticism is often seen as an outside threat. Godot may *fake openness* by pretending to be receptive to criticism, as Godot’s community managers like Nat do in order to control the narrative:

> I love how she says "not to stop talking at all" and then proceeds to block you. Honestly, this whole thing is so sketch it wouldn't surprise me if the profile pic is fake and it's really Juan...
>
> — Andres Hernandez (@cybereality) December 19, 2023

Godot’s community managers may be trying to identify users and contributors within the Godot community who are more critical or outspoken, with the intention of keeping a closer eye on them and finding out who they can trust. Some who are more critical of Godot, but still remain Godot believers, share some insights about Nat. For example, the former moderator of Godot’s Discord server shares her experiences with the Godot team, including Nat and Yuri:

> So, before I get into the next bit of #GodotEngine tea spilled in my lap... I want to point out, I've said before I'm not unreasonable. While I don't post chat logs or other people's IMs, I will post the message I sent to TMM, the "head" of the Godot Foundation near a month ago. pic.twitter.com/dOam0qfFnP
>
> — LillyByte (@LillyByteGames) July 5, 2024

The vast majority of waiters for Godot won’t see this as a problem, insisting that they are *misunderstood*, *misinterpreted*, or even say that something is taken *out of context*, that it only seems weird because people don’t know enough about them, or information presented in a book such as this one mostly stems from disgruntled former members with *axes to grind*,

bones to pick, or *personal vendettas* in mind:

> The poking of fun is because many people already noticed a pattern - trolls usually just get laughed at, people that bring up actual Godot issues get blocked if they don't accept a handwavy explanation. This is mostly done as it buries the comment. Juan is now in an echo chamber.
>
> — Kornel Kisielewicz (@epyoncf) February 15, 2024

Ironically, it's often the critics of Godot who find themselves personally attacked for their critiques of the engine, much like the author of this book! The fanatical Godot community tends to ignore all facts and evidence presented to them:

> Yeah I've noticed as well. Every discussion with Godot cult members rapidly changes from being about Godot to being about u having some kind of personal vendetta against Godot or some bs like that
>
> — 🌈Saas - Bi- Bier trinkende Frühlings Edition🔨🦭 (@S3ys__) April 4, 2024

All these rationalizations are typical in cults and have been studied for decades[1]. The most amusing thing to observe is that cult members of Godot, whether they are users or developers, will instantly respond with "*there's something wrong with you*" and may even publicly label you as mentally ill if you go straight to the core of the issue with Godot's governance.

People report unsavory experiences with Godot users, but instead of delving deeper, they choose to blame themselves, believing that the toxic behavior of the community has nothing to do with their leadership or the software:

> If I'm being 100% honest, I'm slightly biased against Godot because I've had some rather unsavory experiences with Godot users.
> That says nothing about the software, leadership or even the community. It's just my personal experiences weren't great. I'll try to remain fair and...
>
> — Xor (@XorDev) September 15, 2023

Such conclusions arise when people who aren't part of the Godot community mistakenly believe they're the only ones facing negative experiences with it. Guilt-tripping often contributes to a reluctance to speak out fully in the Godot community. While dissatisfaction exists, many people withhold details of their negative experiences.

No one wants to repeat unfounded accusations against them, such as being labeled insane. Despite Code of Conduct violations, the leadership remains inert in addressing or curbing toxic behavior. It is therefore false to claim that the leadership bears no responsibility for the behavior of the community, when inaction may actually reinforce toxic behavior.

In the case of Godot, the blind fanaticism of its community negatively affects the quality of the software, because genuine criticism is often ignored during the development process, when even the lead developer himself uses a divisive "hater" label to discredit and suppress critics of Godot:

> *It must be very frustrating to be a **Godot hater** [emphasis added]. No evil company to blame, most stuff you like to complain about gets fixed after a while, so you need to go read social media to see what new trend to use for hating, etc.*

> Let's be clear @reduzio I am not against Godot.
> I am against your:
> - lies and megalomania,
> - decision-making,
> - programming approach (outdated),
> - gate kipping in project,
> - involvement in foundation and being in W4 at same time,
> - contempt for other engines,
> - blocking people.
>
> — Łukasz Śliwiński (@slizgi) February 1, 2024

Therefore, it becomes increasingly clear that these groupthink problems stem directly from Godot's toxic leadership, as the fish rots from the head.

Pitchforks and Torches

One such example is a blog post of a Godot follower who publicly insinuated that someone who criticizes Godot has mental health problems[2]:

> *I wasn't planing on writing about this guy for two main reasons: first, because I don't think that there's a lot to gain in talking about someone as ridiculously negative as this person appears to be (just who the hell hates game engines!?). And Second, because there's a*

> *genuine possibility of him having some sort of **mental health issue** [emphasis added].*
>
> *So, taking the second point into consideration, I'm going to do my best to avoid insulting this person or making assumptions about his personal character. I'm gonna stick to the reviews as much as possible, and focus on the hilariously ridiculous world he lives in.*

The very small percentage of Godot followers who posses critical thinking abilities say that the above is not a healthy behavior[3]:

Comment
byu/JarLowrey from discussion
ingodot

> *I don't agree with their idea of "high quality", their taste in games, or all the criteria they apply to evaluate games, but I also think way too many comments here in this thread give me the foul taste of a **raised torches and pitchforks towards an individual** [emphasis added] because this individual called our engine bad things and does not share our taste or opinion what games should be made or what makes a good game.*
>
> *If we know better, we should behave better imo.*

Comment
byu/JarLowrey from discussion
ingodot

> *Personally I think what's going on in those two threads with 500+ upvotes is a lot worse than what this guy is doing. That's one individual. **Here are hundreds of people rallying against a single person** [emphasis added] (who also might have a mental health issue).*
>
> *I read countless comments **full of derogatory insults and instigation towards this individual. These threads turned into a mob.** [emphasis added]*
>
> *u/reduz, u/vnen, u/akien-mga, u/Calinou, u/punto-, u/TMM2K12, u/groud0*
>
> *Is this not exactly what the code of conduct is meant to prevent?*

As you can see, even when such a person begins to criticize their community, the commenter still accepts the possibility that the person they're defending may have mental health issues. This is likely due to peer pressure; the answers above were heavily down-

voted by others.

Not-so-personal Vendetta

The in-circle suppression of doubts is also prevalent within the Godot community. Take a closer look at a thread on Godot's community forums, where fellow Godot followers ignored claims made by the author of the thread—a respected member of the Godot community who also happens to be the forum administrator[4]!

Before we delve further, it's worth clarifying that words like "scam" or "grift" are labels that may partially describe certain aspects of Godot (see also explanation at Community-Driven chapter). Most people are aware of "scams" above all else in everyday life. So using the "scam" word may be a good way to attract necessary attention to the righteous cause of exposing the underlying, fundamental issue, which is a toxic cult of Godot. In fact, another author of the video associated with that forum thread came up with a definition of "*psychological scam*"[5]!

> TL;DR about @GodotEngine.
>
> As a former maintainer of Godot, I fully agree.
>
> The term "Psychological Scam" is basically another way to say #GodotCult in this context.
>
> We see the same vicious cycle from Godot 1.0 to 2.0 to 3.0 to 4.0, and the pattern continues.#TruthAboutGodot pic.twitter.com/na2huj1HDB
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) November 22, 2023

No matter how well-suited a definition may be, cult apologists will always frown upon such characterizations. What really matters, though, is not the specific labels used, although it is important to use the correct terminology. Ultimately, it all comes down to concrete evidence that can be discussed objectively.

Let's take a look at how moderators reacted to the thread:

> *Operative words.* ***Anything can appear as anything*** *if you just look at it from the specific angle that leaves the 1 specific impression. Again, can't blame anyone that* ***falls into that trap****, it's human to do so and certainly not helped by Juan using the same account to try and*

> *get funding for both entities (which is now me **projecting** cause I'm assuming that he does so, but I'm not a twitter user and I don't follow him).*

It's not just disrespectful to say such things in response to concrete facts that can be easily verified. It's also a form of gaslighting. Another reply also denies facts and labels criticism as *dogpiling*, *[emphasis mine]*.

> *I don't understand this **dogpiling**. Especially done in public. So far, I haven't seen anything shady or wrong being done, and there seem to be a lot of incorrect comments being thrown around, especially regarding the financial parts.*

Denial and misrepresentation are common cult tactics. Another poster accused the author of practicing FUD. Ironically, FUD is also associated with cults, so what we are seeing here is a manifestation of projection by a member of a Godot cult. FUD is basically an attack on the author's character, and is often used by fanboys to reject criticism, as in other cases, *[emphasis mine]*:

> *You're wasting everyone's time. You've shown no evidence of malfeasance. You're just using this forum (which YOU pay for and control) to air **baseless accusations and FUD**. I'm out.*

Other members *appeal to emotion* to discredit the author's claims, and disregard the evidence as merely a *personal opinion*. Simultaneously, they acknowledge a possibility of the author being right, but then immediately say that this is irrelevant, which signifies a cognitive dissonance, *[emphasis mine]*:

> *I understand there's **a lot of anger**. But someone being overly ambitious and underdelivering on expectations != Being a scammer or liar. That just means they over-promised on a deadline they couldn't keep and need more time. And you not liking the direction of the technologies used or not used, that's **a personal opinion**... you might be right, but that's irrelevant. But instead of trying to convince the internet that they are the **devil**, why not be a bit more constructive? It's open source, and people are free to implement their own solutions. Or just use something else and be diplomatic about it?*

Contrary to what this Godot devotee says, the author did not paint the project leaders as *devils*, which is a strawman argument. Here are other instances of such behavior at Godot's subreddit[6], *[emphasis mine]*:

---

*I don't use Godotforums and now I know it's managed by some **deranged child**, I have no interest in joining either.*

---

*It looks like he has a **personal vendetta** against Juan because of what he has chosen to prioritize for the project.*

---

*I knew that from the beginning, but it seems Cyberreality has a **hidden agenda** against Juan and the Godot Team.*

---

*The more I read about this the more **delusional/unstable/mentally** unwell cyberreality seems.*

---

*I cannot even imagine someone writing all that... **Man needs help**.*

The above is just a subset of such toxic behavior. You may be shocked, but the following examples depict even more severe behavior: when the admin of the Godot forums *apologized*[7] (but not before the cult leader of Godot, only to the community), the response to the apology was as follows. Look at the vast number of insults and not accepting apology:

---

*People like this really need to **fuck off**. You're not important and nobody cares.*

---

*Whatever, dude can **go to hell**. That's no apology. That's a **fuck off** and leave me alone statement.*

---

*Glad he is out. **Pathetic** and childish behaviour and didn't even apologized properly. Godot is better without him*

***Pathetic** is a perfect way to describe it. Felt like a twelve year old after being called out by their friends.*

> *Idk whats going on here but this person sounds like a massive **chode**.*

> *When people have a mental break with reality, they're pretty incapable of apologizing for it.*

> *Apologized, then spent a paragraph throwing the project under the bus.*

> *"I'd like to apologize" is not an "I'm sorry for X, Y, Z" in my book.*

> *Glad he's stepping down. Guy **needs some help**.*

> *What a **twat**.*

> *What a **weirdo**.*

And the ironic part:

> *Wow, it doesn't sounds like an apology at all. He still belittles the engine and people who uses it.*

A rare instance of Godot followers who possess critical thinking abilities rightfully responds:

> *Yeah the comments are **really abysmal** in this thread same most comments the other threads from the past few days. You also show the strength of a community by how the community is treating someone who leaves. What is happening here is **really disappointing**.*

What could possibly explain the motivation behind such a pressured apology is the superficial "love" (see Love Bombing chapter) from the community that the admin likely received in the past. As previously said, humans naturally have a need to belong, and being

part of an open-source community may temporarily grant that feeling. With Godot, the issue arises when it becomes overly strong, leading to an excessive attachment.

Therefore, it's not wrong to generalize the Godot leadership's undue influence to a larger group because, eventually, the fish rots from the head, see Cult Leader chapter. If you've been an attentive reader, you should have noticed that the admin of the Godot forums is the same one we introduced in Blue Robot Cult. Yeah, that's quite a twist indeed! 😁

It's important to recognize that attempting to be diplomatic with cult leaders and members of a cult can actually be detrimental to your own well-being. Unfortunately, the author felt pressured to *apologize*. However, the rule of thumb is to *never* apologize before a cult. By apologizing, you inadvertently validate their beliefs. Of course, the real challenge here is to be able to recognize the community being a cult. When they started a thread to notify others that the admin of the Godot forums apologized, they weren't interested in an apology; they simply reveled in cult conformity.

Cults often have their own distorted worldview and ideology that they use to maintain control over their members. Apologizing can give the impression that you accept or acknowledge their worldview, further perpetuating their practices. Apologizing can reinforce these imbalances by positioning the leader or community as the ones who hold authority and righteousness. This dynamic can further *disempower individuals*, *hinder critical thinking* and *independent decision-making*. It can send a message that their behavior is acceptable or forgivable, contributing to a cycle of abuse and control. Apologizing will *not* protect you from further manipulation or ostracism. Instead, it could expose you to further exploitation or retribution from the cult leader or community.

What's even more interesting is that, despite the vast number of evidence presented, some still choose to *believe* in Godot, and they literally say that success cannot be achieved without *a lot of pain*, and they *hope* that Juan, the great cult leader, will mature! 🙃

> *I still believe in Godot however, and I think things will turn out fine eventually. But not before a lot of pain.*
>
> ...
>
> *I think the next couple of years will be Godot's rockiest yet. But these years will also define Godot as a project, and hopefully Godot will mature. Hopefully, Juan will mature, if he wishes to stay at the helm.*

This further reinforces the idea that Godot is a Trustocracy. In a community like Godot, you either bootlick and trust the leaders, or you don't and you get kicked out. Intellectual discourse regarding Godot's management end up lacking purpose and are seen as futile in

the eyes of Godot followers. What the poster said is nothing more than wishful thinking. Just like with scams, when you are deep in a scam, you don't want to feel scammed, you don't want to admit that you got scammed, so you tend to get scammed further in the process of not admitting that you got scammed! 😁

Eventually, the forum has been taken over by new leadership controlled by a company, with Mike Lundahl at the helm. The thread started by the previous admin has now been ***deleted*** by the new owner[8], who claims it was to create a more "positive" and "optimistic" atmosphere. 🤡

Due to such events, Godot's toxic leadership will likely establish an even stronger mechanism to block all criticism regarding Godot's hypocritical governance. Eventually, ownership is likely to be transferred to the cult leader of Godot, Juan Linietsky, or any other sub-leader, aiming for even tighter control over the community under their undue influence.

Undeserved Appreciation

The excessive promotion of Godot without acknowledging its significant issues is not helpful. These issues require attention, and having an influential position could raise awareness about them. However, Godot's undue influence may also sway somewhat influential people into blind devotion. This is evident from what we could call "appreciation" posts as done by both Godot's leadership and fans.[9]

> Can we all take a minute to appreciate #GodotEngine dedication, who every morning wakes up early, takes a shower and shaves in order to be in peak shape for your game development needs? pic.twitter.com/cNsAJMnakh
>
> — Juan Linietsky (@reduzio) September 5, 2024

It's a lot easier to dismiss critics as "Godot haters" and paint a picture of a hardworking community than to actually tackle the real problems. This kind of social dynamic is demonstrated in the video below:

The comment section is filled with remarks that either attack the character or motivations of the video's author, contributing to the growing evidence of a groupthink mentality surrounding Godot. Ignoring and justifying the bullying of former maintainers is not okay.

Conclusion

It would not surprise the author of this book if those who choose to expose Godot's wrongdoings ended up *apologizing*. However, such an apology would not stem from the standpoint of *autonomy of the will*, but because of the *undue influence* exerted by the mob mentality that originates and is reinforced by the *toxic leadership* of Godot. Unfortunately, the fish rots from the head, and its community becomes responsible for adhering to the unethical practices of Godot's toxic leadership.

People who critically express their opinions about Godot get labeled as "haters" by the Blue Robot Cult, and no apologies are going to be treated as sincere. Those who don't share their ideology are labeled as biased, corrupt, mentally insane, or conspiring against them. Moreover, hate rallying against a single person isn't seen as a violation of Godot's Code of Conduct by Godot's *toxic leadership*, because such an attitude is what describes cults: *scapegoating* and *group narcissism*. Such behavior is even endorsed by the toxic leadership.

My only advice to Godot followers who act like this would be: *look in the mirror*. Community and leadership of Godot get upset and angry when people:

1. Don't use Godot logo as a splash screen.
2. Don't "contribute back".

3. Talk about downsides of Godot or compare Godot to other Alternatives.

Is it really worth it?

References

[1] Cult Apologists | Cult Defenders - By Apologetics Index.

[2] The Curator who Hates Godot - By Godot Notes.

[3] When someone's a hater because of your game's genre and engine, but still compliments the execution... - Comment at Godot subreddit.

[4] Sadly, I think Godot is a Scam. I'm Not Sure I Can Do This - By Godot Community Forums Admin. Alternative archive: Archive.Today.

[5] Is Godot A Scam? - YouTube.

[6] A positive take on the recent problems the Godot community has been facing, let's keep our heads up! - Godot subreddit.

[7] Cybereality apologized - Godot subreddit.

[8] A Warm Welcome Back to the Godot Forums, Under New Stewardship! - by Mike Lundahl.

[9] Juan Linietsky's post was made during a time when people were outraged about Godot not providing free consoles support, like other FOSS projects do. For more details on the issues with Godot's console support, see Community-Driven chapter.

Authoritarianism

The road to hell is paved with good intentions.

Initially, it may appear peculiar to characterize Godot's governance as authoritarian. However, considering the presence of *undue influence*, primarily deceptive practices, such a characterization suits quite well within the frame of cultic groups[1]. While a facade of democratic representation is upheld within Godot's community, delving deeper into the inner circle of developers and examining this situation from an insider's perspective reveals the underlying dynamics that begin to fall into place.

The effect of Godot's hierarchy is that the majority of users remain oblivious to the true nature of Godot's governance. They continue to operate under the illusion of democracy, unaware of the hidden dynamics and manipulation taking place behind the scenes of Waiting for Godot.

Cult leader's story about leadership

Godot's current governance can be explained if we look at the reasons behind Juan's understanding of leadership in the past. Here's what kind of story he told to his cult followers[2]:

> I will tell a story about how I learned about leadership. A long time ago, I got drunk at a business meeting with the then owner of one of the biggest gaming companies in Asia (and his subordinates, all industry veterans). Went for dinner, then to their hotel to keep drinking. pic.twitter.com/0LynRSxlXm
>
> — Juan Linietsky (@reduzio) July 19, 2020

> *I will tell a story about how I learned about leadership. A long time ago, I got drunk at a business meeting with the then owner of one of the biggest gaming companies in Asia (and his subordinates, all industry veterans). Went for dinner, then to their hotel to keep drinking.*
>
> *At some point, this person (over 70 I would guess) passes out, so I am like "Well, guess I should leave". His subordinates ask me to stay a bit, ensuring me he will recover. They tell me if there is anything I would like to ask about how their company works to pass the time.*
>
> *Even drunk, the mood got serious. I was confident in my skill, but was starting to have more*

> *responsibilities and was frightened of leadership. I asked them how do you make people do what I wanted, how do they handle leadership roles.*
>
> *They laughed and told me that you don't tell people what to do, that this is not being a leader. They told me that if you are a leader, you agree with them. So I am like "What if we don't agree?", and they answer "then you are just a boss, not a leader".*
>
> *This didn't make sense at first to me, but it fully did over time and I was able to see the clear difference between both stances. Even today with Godot, I am obsessed that we must all get our points through and agree with things, before moving forward, Developers and community.*
>
> *Oh, and hangover with wine is the worst.*

Behold Juan's intriguing story! Whether Juan's tale holds any truth or not, one thing is clear: Juan was on a quest to unlock the secrets of controlling people's minds, as he desired to figure out "*how to make people do what he wanted.*" How fascinating! Of course, those who laughed at Juan were absolutely justified because this train of thought characterizes a boss, not a leader!

Even now, presumably long after the events took place, Juan still possesses a boss mentality. He did listen to advice of Asian leaders though, but instead of becoming a healthy leader, he brilliantly ascended to the rank of a *cult* leader!

Juan's persuasive abilities contribute to the reinforcement of his control over the community. By projecting an image of innovation and success, he effectively cultivates a sense of dependency among followers. They come to believe that their involvement in Godot is a privilege, further reinforcing their loyalty and diminishing the likelihood of questioning the leader's authority.

Truly a stroke of genius! 🙃

Governance

The core problem is that Godot is trying hard to look like a *community-driven* project, but their actions show a complete disregard of principles such as *open discussion* if people try to:

- discuss the governance model itself;
- discuss the development philosophy (which does not exist);
- discuss actions of members;
- compare Godot to other technologies (which requires making analogies);

- show inconsistencies in the development process (which often requires quoting);

For example, one of the Godot contributors suggested to remove mentions of "community-driven" from Godot's documentation[3], and such efforts were met with a great push-back from Godot leadership, saying that it's done in "bad faith."

Of course, people who bring up these issues may not be immediately banned, as most new contributors go through so-called educational process instead, a.k.a. brainwashing. However those people get on the radars of Godot's community managers in any case. If people are found to be allegedly *troublemaking*, they may be labeled as *outsiders*, and outsiders cannot be trusted. If a contributor is incapable of conforming in alignment to Godot's established status quo, Godot leadership first attempts to reproach those people in private. The punishment that contributors receive is directly proportional to the amount of time they committed to the project.

As a former member of Godot, I have personally received a private message from Godot's project manager. Rémi suggested me to "stop quoting" them, and presented it as a *potential* violation of Code of Conduct. However, nothing in CoC stated this, especially if we take this together with the *open discussion* principle. In fact, they consistently violate their own CoC because it clearly states:

> *Always assume positive intent from others.*

Unfortunately, they don't assume positive intent, especially Rémi (aka Akien, aka akien-mga)[4]:

> Comment
> byu/akien-mga from discussion
> ingodot

You can be excluded from participating in Godot communities even if you don't break Godot's Code of Conduct. Here's a quote from one of the CoC members, Ilaria Cislaghi[5], who also represents Godot leadership officially:

> *Note, people are not getting removed/banned because others wake up with the wrong foot one day. This just means that CoC is a guideline and if people are found to be disruptive to the development/community they may be removed* ***even if they didn't explicitly breach one of the rules in the CoC*** *[emphasis added].*

This is called permissiveness and iniquity. It means that the leadership of Godot allow themselves to do whatever they want with people, however they want it. This is also the moment when they figured that their governance model is *trustocracy*, exactly as they put it. From Godot Contributors Chat:

> *Yeah, it's not really a meritocracy. Juan calls it a "do-ocracy", where the ones who worked enough (and got their work merged) on some specific areas of the code can decide on what's get merged. But ultimately, the fact you become a maintainer mostly relies on others maintainers and the PLC trust on you and you work I believe. There's not really an "official guide to become a maintainer". Can we call that a "trustocracy" ? xD*

Ilaria responds to above:

> *I think that's the most correct way of calling it. Trust is gained by good work and constructive attitude.*

This further confirms that Godot's governance is *trustocratic*. But do you know what *trustocracy* associates with[6]?

DAILY PEOPLE

VOL. 11, NO. 15. NEW YORK, FRIDAY, JULY 15, 1910. ONE CENT.

EDITORIAL

SLAVEOCRACY AND TRUSTOCRACY.

By DANIEL DE LEON

THE advocacy by the leading figure in the Socialist party, Victor L. Berger, of the policy of "buying the Trusts," together with the argument that the gentleman supports the policy with—Henry Clay's proposition to "buy the slaves"—furnishes in hand a prime illustration of the blunders that attend upon ignorance of history, or, rather, shallowness in historic knowledge.

Don't be scared! This is not to say that Godot contributors are slaves, but the undue influence which Godot leadership has upon volunteers is definitely worrying. For example,

here's what Juan suggests in order to find free "manpower" to try volunteers to work on specific things in Godot[7]:

> *Yes, don't get me wrong, that IS the case, but in general, the order of priorities is something like this:*
>
> *1.* ***Try to find a new or existing contributors to work on a specific area that is lacking a maintainer (or manpower)*** *[emphasis added].*
>
> *2. Try to see if we can get a grant from an interested party to hire contributors to work on this area (very often companies do direct or indirect hiring -via grants to Conservancy-), or even via GSOC.*
>
> *3. If all else fails, it goes to the agenda of one of the core contributors.*
>
> *If we don't do things this way, the project can't scale. Core contributors often work in core areas and also* ***spend a large amount of their time*** *[emphasis added] helping other contributors.*

I call this phenomenon **VaaS**, volunteers-as-a-service, because that's exactly how Godot's leadership treats and presents volunteers to Godot's corporate clients. This phenomenon is relatively new Open Source, especially with the advent of Corporate Open Source Software (COSS), but as one saying goes, everything new is well-forgotten old... 🙃

Behavior Control

Leadership

As we have covered in Companies vs FOSS section, Juan says that they don't compete with Unity, Unreal, etc, and that they make Godot because they enjoy to innovate, and that engines like Unity can take Godot's code. Yet Juan **discourages** people from comparing Godot to any other technology, saying that they should not cross this line if they care about Godot[8]:

Juan's reply to the meme above:

> *I would much prefer the community discusses strengths and weaknesses of Godot and don't go around comparing to other technologies. Also none of the core Godot contributors or PLC have anything to do with this, so the logo and signature are very misleading. Please don't do this.*

As a result of this response, a follower of Godot is compelled to apologize to the cult leader:

> *My bad, I thought it was funny and wanted to share some humor, but I can understand how if one hasn't seen the origins of this meme format, it can seem malicious. No bad intent there, just misunderstanding.*

Juan then responds once more:

> *It's fine I understand there was no intention of harm myself, but because of things like this, a lot of people will form **a very wrong idea about Godot users or contributors** [emphasis added], so if you care about the project, please don't cross this line again in the future.*

What kind of so-called "wrong ideas" could people form about Godot users and contributors? Could they come up with a conclusion that Godot is a cult? 😉

Some people expressed outrage at Juan's inadequate response to an innocent meme:

> *Why make this a big deal? It's just a meme. A community of people that take memes that seriously doesn't sound that good as well tbh. Is it supposed to be a **community of robots** [emphasis mine] that only do Q&A, tutorials, docs and code? No fun and play between the members of this community?*

> Why make this a big deal? It's just a meme.
>
> A community of people that take memes that seriously doesn't sound that good as well tbh.
>
> Is it supposed to be a community of robots that only do Q&A, tutorials, docs and code?
>
> No fun and play between the members of this community?
>
> — .pig//Dev (@pigdev) January 5, 2020

Critical thinkers understand that strengths and weaknesses are identified through a process called comparison, which is how humans learn. However, many waiters for Godot appear to reject this process. Because of this, **Godot is frequently used as an anti-Unity meme**, and it's actually Juan Linietsky who started this trend in the first place. Godot parasitizes on the failures of other projects, not just Unity. The article titled "Devs learn rival Godot engine in a week to poke fun at Unity" begins as follows[9]:

> *It's generally **not accepted as good form to kick someone while they're down, but we can always make an exception** [emphasis mine] where predatory business practices are concerned, especially in the $334 billion video game industry.*

It's important to critique predatory business practices, but resorting to the "kick someone

while they're down" analogy to justify violence is not just immoral; it also fuels a destructive us-versus-them mentality. In fact, framing it this way mirrors the predatory behavior being criticized, prompting us to question who the true predator is in such a scenario. As you may know, Unity's pricing changes caused a stir in the game development industry. Quote from the article by the creator of Install Fee Tycoon, made with Godot:

> *We want Install Fee Tycoon to stand as a hilarious memory of what might have been, if the entire games industry didn't collectively explode in panicked revolt.*

This is precisely what people need to do: speak up about the issues. However, in the case of Godot, consistently voicing concerns *will* eventually result in being ostracized from the community. If Godot continues to occupy people's minds, the gaming industry could suffer significant decline because dissenting voices won't be tolerated. The ability to freely criticize Unity is far more valuable than people think.

Even after Unity provided even better pricing options for indie game developers after the backlash Unity received from the community, Unity continues to be mocked and literally demonized by Godot fans:

Therefore, promoting Godot as a rival to Unity is not only misleading, but counterproductive. Godot is way more destructive due to its nature as a Toxic Cult. The constant presence of problems and the ongoing discussions about Unity are portrayed as an immediate and unrecoverable situation by Godot fans.

They mislead developers, especially newcomers, into thinking that openly discussing and critiquing problems leads to inevitable failure. In reality, such discourse reflects a mature

community capable of addressing potential problems head-on, unlike what is observed in the childish Godot community. For more in-depth analysis of these issues, read investigation article titled Waiting and Unionizing.

Community

Another case of this kind of behavior control is also present on the level of not just Godot leadership, but Godot community moderators. The admin and moderators of Godot Discord server are known to ban people left and right for the wrongthink. A notable case is "Pepe the Frog meme" drama[10], while ironically, the admin have skeletons in his own closet. Here's a quote by someone in that thread:

> *It's wild that they are so ban happy when the main active mod has a history of posting n-word (hard r) in the Godot server. He had defended multiple times leaving a live history of hard r n-word, but is now on sight banning people for having a discussion about pepe? It's just so so wild.*

Comment
byu/VenomousInc from discussion
ingodot

Another comment:

> *Wow, the plot twist. I did not expect this from an open source project's discord server! FOSS communities tend to be very friendly and welcoming.*
>
> *I guess that explains the wording on the discord server's rules and some other places. That screenshot shows the mod being plainly rude and more of a bully rather than a facilitator for its users.*

Godot leadership immediately deleted the post, literally labeling most comments as breaching Godot's Code of Conduct. They also came up with various newspeak and bogus policies mentioned there: "*Ban fast, unban easily*". Nothing can be further from the truth. To lift the ban, you must either admit guilt or follow a lengthy process until the moderators persuade you that you are at fault and require an apology. However, you may not have committed any offense that would justify a permanent ban in the first place.

Eventually, the person who created that thread was unbanned. However, as the author

stated:

> *I have to agree, I was unbanned as of now, but that doesn't make me feel better about partaking in the future.*

If not this drama, the author of that thread could remain banned forever, just like most of the people out there who don't speak out in Godot community. This discourages critical thinking, not to mention other issues. With each such case, slowly, but steadily, Godot degrades further.

In a cult environment, there is no room for mistakes. The manipulation of perception and the requirement of unquestioning adherence become fundamental elements of such groups. In Godot, those who hold power through the 'ban hammer' maintain a tight grip over the community, perpetuating their influence. This control allows them to exploit vulnerabilities and desires, reinforcing their authority and suppressing dissenting voices.

Ends Justify Means

I once talked to Juan, bringing up the issue that what Juan says to the public goes contrary and contradictory to what he actually does, so this confuses contributors and users. His reply was this:

> *People are not confused or deluded, I think people are actually pretty smart. I also don't have the ability to manipulate people or anything like that, I just say what I think and everyone can agree or not, but I don't force anyone. Those who contribute to Godot are those who agree to some extent, those who do not don't contribute and its fine.*
>
> *Ultimately what matters is the result, and so far I think the results are quite positive.*

It's a myth that cult leaders know they deceive, cult leaders may even believe their own lies due to their pathology[11]. Regardless of intelligence, all people are vulnerable to cult indoctrination. This is due to the manipulation of trust. Even then, Juan's last phrase crosses out everything that was said before, telling that what *ultimately* matters is the result.

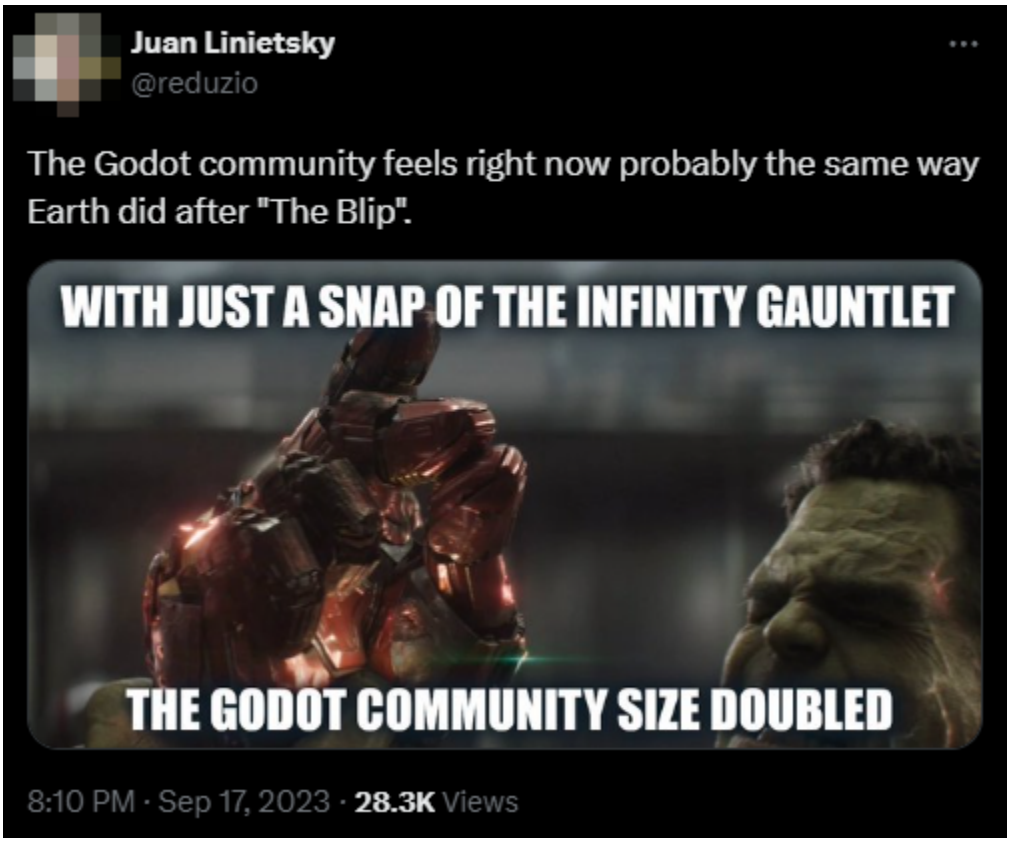
Here's another example, where love bombing is mixed with dehumanization rhetoric[12]:

> *To all of you, thank you from the bottom of our hearts.* ***You are the fuel*** *[emphasis added]*

> *of this project and there aren't words to express how much that means to Godot.*

This rhetoric implies that the leadership values contributors only as a **means to an end**. The leadership's focus is on achieving success for Godot, be it fame, status, money, or otherwise, without much regard for the contributors' interpersonal relationships or impact. In contrast, healthy open-source organizations prioritize treating contributors as ends rather than means.

Quantity Over Quality



"With just a snap of the infinity gauntlet the Godot community size doubled" - Meme by Juan Linietsky

To manipulate the perception of importance, leaders of Godot consistently employ grandiose rhetoric about the ever-increasing size of their community to captivate their followers. See how Juan literally fixates on growing the number of contributors[13]:

> *We have dozens of core developers and more than 400 contributors working on Godot, and this number keeps growing **exponentially** [emphasis added] month by month.*

At the time of writing this book, Godot has over 2000 contributors, although most of them are one-time committers. While this is not exponential growth, Juan uses this grandiose narrative for propagandist purposes. It is peculiar that Juan consistently emphasizes the number of contributors in Godot as if it is the project's most significant aspect.

Here's how they fixate on number of "stars" in Godot as well[14]:

> *Have you starred Godot on Github? We are only 12 stars short to reach 50000! Who will be the **lucky soul** [emphasis added] to hit this number?*

> Have you starred Godot on Github? We are only 12 stars short to reach 50000! Who will be the lucky soul to hit this number? pic.twitter.com/xNtNB28881
>
> — Godot Engine (@godotengine) June 28, 2022

As you see, even their rhetoric such as "lucky soul" expose them as real cultists! 😳

Recall Juan's carelessness? It is also apparent in the source code of Godot. Sergey Vasiliev's analysis of Godot's codebase reinforce these observations[15]. One day, Rémi attempted to enforce a policy of treating all code warnings as errors, which may suggest that Godot values quality. However, the usage of static analyzers in Godot exist primarily to create the illusion of a diligent and hardworking community. Law of triviality states that people tend to focus on trivial matters, even when more important issues are at hand. This phenomenon is evident in Godot, where occasional contributors often prioritize trivial tasks.

The additional workload is amplified by the presence of strict static analyzers integrated into Godot's continuous integration (CI) system. This situation becomes even more perplexing when considering that the emphasis on trivial matters is unwarranted, given the persistent existence of major bugs and limitations in Godot that remain unresolved for years. One might attempt to rationalize this by attributing it to a lack of manpower within Godot, despite having "thousands of contributors." However, let us recall what Juan stated about hiring developers to work on Godot in Recruiting; they are simply not interested in doing so!

Despite Rémi's efforts, be rest assured that Juan and others will undo years of quality assurance work with yet another major rewrite, going from bad to worse. 😁

This is what Juan said when talking about catching regressions[16]:

> *There are a lot of tests for Godot functionality, but unit testing is not magical. If a bug was not found, it simply was not found.*

At that time, Godot did *not* have many tests, so we can see a rhetorical emphasis on quantity even at this level. Those who appreciate the advantages of unit testing may find Juan's stance on this matter peculiar. Considering that Godot's codebase is frequently rewritten between major versions, it would be sensible to use testing to avoid regressions. What could account for Juan's position? In short, they may not be overly concerned about regressions because this ensures there's always something for new contributors to work on. While this is illogical, this is the way of Godot. Instead of preventing regressions during development, they rely on the community of users to catch them in production.

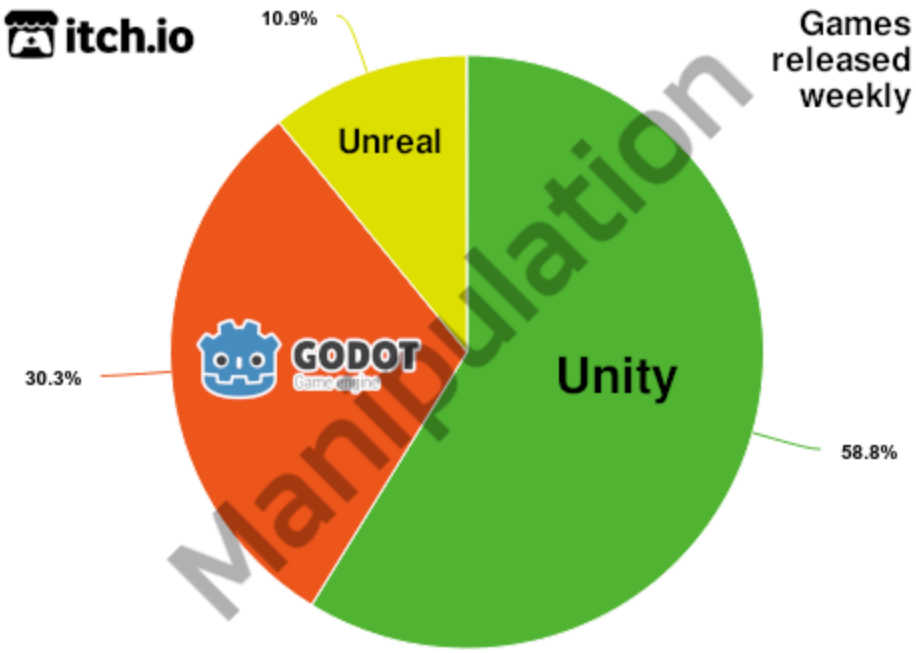
The focus is on the satisfaction that contributors feel from contributing to the project, rather than on the impact of such contributions on games developed with Godot. Stating that they care about quality is a facade to attract new contributors (see previous Recruiting), especially students who tend to choose to work on unit tests. Any community efforts to establish solid unit testing coverage will eventually be nullified by Juan's carelessness.

Godot leadership and the community are in a tug of war. The leaders aim to promote Godot and increase the number of users, while the community desires a high-quality product. In the end, leadership prevails. The community often sides with Godot's leadership, believing that quantity is more important. However, they fail to recognize that quality is what could make Godot a preferred choice for industry experts, leading to adoption by the general public of developers. Unity serves as a prominent example. Even then, there is no implication that Godot should become as big as Unity in terms of adoption. Many smaller projects exist that do their job well.

Due to this, Godot Engine appears to be popular for game jams, where many games are created using it among other open-source game engines. Juan focuses on presenting Godot as a perfect choice for indie game developers, but such presentation is often manipulative. For example, here's what Juan said:

> *The trend is much stronger on http://itch.io vs Unreal and Unity (again, **not counting other engines** [emphasis mine]), this is based on average of published games for some weeks, showing that for those who just do game development Solo, Godot is becoming a major choice..*

First of all, recall Juan said that they don't compete with Unity or Unreal. Secondly, look at his chart he presented there[17]:



Looks like Godot is the new king of the gaming world... If you ignore all the other game engines! Who needs a comprehensive analysis when you can just omit competition, right Juan? Let's celebrate selective data and half-baked conclusions! 🙃

Jokes aside, visual manipulations that completely omit some product in a chart can be described as a form of graph manipulation or distortion known as "omission" or "exclusion." Juan intentionally removed other game engines from a chart to highlight Unity and Unreal. This created a false message that they are equivalent game engines that can compete, while downplaying other engines like GameMaker. People's confusion confirmed this.

> *This isn't showing GM2 or any other engines. [Are] they excluded on this pie chart or are all the games released truly exclusive to these 3 engines?*

"Why did Juan exclude other engines?" you might ask. Juan is heavily biased due to his wishful thinking and avoids taking responsibility for his words. Regardless of Juan's motives, when he explicitly disregards significant game engines like GameMaker, his chart still misrepresents data. From the perspective of Godot followers, there is no reason to doubt Juan. Juan's "clarification" may go unread, and the misleading image could hold undue influence. Some may choose Godot without considering other game engines, merely because they trust Juan's words. The impact of the cult of personality should not be underestimated.

In summary, despite appearing to value quality, the Godot leadership actually exploits the notion of quality as a marketing tool to attract more followers, users, contributors, and sponsors. In other words, it's always easy to claim to value quality even when this may not

reflect reality. Although Godot developers may claim to prioritize quality, the project's deficient management and authoritarian governance often prioritize quantity over quality instead.

Compromise Over Consensus

As we have covered in Trustocracy chapter, Godot is based on pure trustocracy. If you start to question technical decisions in Godot, you'll be labeled as someone who cannot be trusted, so they will start to ignore your pull requests, proposals etc. For your information, the pull request time cost is reaching up to **three years** before a change is merged in Godot[18]. The ambiguity of the development process and Godot's vision contribute to the high number of proposals and pull requests. However, many of these changes are rejected by Godot leaders, leading to disappointment among contributors.

You may have heard that Godot's development decisions are based on consensus, but this is not true. While opinions are considered, the final decision is ultimately made by the technical decision-makers, Juan and Rémi. In particular, Juan's language reflects his authority[19]:

> *In any case, if you want to leave this proposal open I am fine, but for the time being you don't have **my consensus** [emphasis added] to go forward with it.*

Someone replies ironically:

> *Democracy... has not won. 😅 But if this is the vision of the project maintainers, I will no longer argue and support this decision.*

If we assume positive intentions, Juan may not fully understand the meaning of the word "consensus" or he may believe he has the sole authority to represent it. This is reminiscent of the rhetoric used by dictators.

Again, it's not a problem per se if BDFLs like Juan want to govern their project this way, but some projects are utmost honest in this in contrast to Godot. For example, see Bevy's contributing documentation when they explicitly mention about the fact that the project is developed by "singular Benevolent Dictator and project lead"[20]:

> *@cart is, for now, our singular Benevolent Dictator and project lead. He makes the final decision on both design and code changes within Bevy in order to ensure a coherent vision and consistent quality of code.*

You won't see this kind of honesty in Godot, because Godot's leadership shields itself behind community. The alleged success of Godot is built upon lies and manipulation of contributors.

I also had a chat with Juan, and he said this:

> *Those who became maintainers just picked an area of the engine they liked, worked on it, contributed, showed results, showed that they did not need much help, had always a **flawless attitude** [emphasis added], etc, **showed lots of compromise** [emphasis added] for the project and they did all this humbly and without asking for anything back, just because they loved doing it. They did not do this with the intention of be given responsibility. This is why, someone that also makes it clear that is seeking to be given responsibility inspires the opposite effect of gaining trust, specially if it tries to **politicize things** [emphasis added] going on. This makes other contributors, maintainers, etc. be wary of you and not trust you.*

This further confirms that compromise is more important in Godot, especially when we take this with *flawless attitude*, which means unquestionable attitude in reality. This is certainly a problem, because asking questions is very important to reaching consensus. Contributors don't ask to be given responsibilities nor ask to be trusted. Yet Godot leadership keeps fixating on trust regardless, because ability to ask questions posses a risk to Godot's cultic status quo.

Godot leadership also labels discussions on Godot's governance as *political* discussions in order to block them rather than seeing such discussions as genuine attempts at clarifying Godot's governance model. This is also because they project their own real political agendas upon contributors, because that's exactly what *they* do, as you've seen in chapters such as Cult Leader.

Conclusion

Even if we don't take into account governance issues covered so far, the problem is not necessarily about Godot's governance model, nor any other ideology which you can see in

open-source projects, corporate organizations etc. The core problem is Godot's hypocrisy and deception. If you ever get angry comments from either Juan, Rémi, or any other Godot cult leader, and they start to blame you for things you haven't done, it's the time you should know you're participating in an abusive relationship.

Godot leadership treats community as a fuel. Juan seldom completes features. On the surface, they make the engine seem great to attract corporate sponsors that are oblivious to what happens inside the inner Godot cult of developers. Critical bugs remain unresolved for years. You may say it's a nature of open-source, but other open-source communities have a sense of responsibility in contrast to Godot.

There are mostly compromises in Godot, which are all biased towards what Juan wants personally. Even if a major decision is made without Juan's explicit expression publicly, be rest assured that such decision was made behind the doors, and executives of Juan's decisions are Godot's sub-leaders for the most part, such as Rémi.

Juan receives flattery and praise in Godot community on a daily basis. Cult of personalities have proven to lead to dramatic consequences in human history. Of course, Juan is not a big tyrant, he's more like a *petty tyrant*, but given enough power, he could run authoritarian state or a failed democracy state, and this leads to corruption. Godot won't be able to achieve true success because dissent opinion is not tolerated in Godot's development community, especially opinion from those who have acquired recognition in Godot community.

We can only "applaud" how Juan masterfully deceives Godot followers, and how he spawns people similar to him around him. But from the point of view of civilized community, they in fact destroy the autonomy of the human will. These people who follow Godot could've spent their time and effort on healthy endeavors that match their life purpose, and not supporting endeavors of toxic leaders.

Therefore, it is essential for individuals to critically evaluate the dynamics at play and consider the potential harm that can arise from such a manipulative and authoritarian structure. Awareness and understanding are crucial in breaking free from the grips of undue influence and reclaiming autonomy.

References

[1] Influence Continuum - By Freedom of Mind.

[2] Story about leadership - By Juan Linietsky, Twitter.

[3] Remove mention of community-driven and mention feature PRs may not be accepted - Godot, GitHub.

[4] Rémi Verschelde violation of Godot's Code of Conduct - Godot, Reddit.

[5] QbieShay on Code of Conduct - Godot Contributors Chat, **#coffee-break** channel.

[6] Slaveocracy and Trustocracy - By Daniel De Leon, July 15, 1910.

[7] Juan Linietsky's comment on finding volunteer manpower - Godot, Reddit.

[8] Don't go around comparing Godot to other technologies - By Juan Linietsky, Twitter.

[9] Devs learn rival Godot engine in a week to poke fun at Unity - By Richard Currie, TheRegister.com.

[10] Discord Ban - Godot subreddit.

[11] Common Myths and Misconceptions About Cults and Cultic Groups - By International Cultic Studies Association.

[12] About Godot 4, Vulkan, GLES3 and GLES2 - By Ilaria Cislaghi.

[13] As an Open Source project, Godot is more than a game engine - By Juan Linietsky.

[14] Godot Engine 50000 stars - Twitter.

[15] Analysis of Godot Engine's Source Code - By Sergey Vasiliev, April 30, 2015.

[16] Juan Linietsky about unit tests in Godot - By Juan Linietsky, GitHub.

[17] Juan Linietsky chart about Godot's adoption for game jams - Twitter.

[18] Godot Engine's pull request time cost - Data from OSS Insight.

[19] Rename AudioStreamPlayer{,2D,3D} or VideoPlayer for consistency - Comment by Juan Linietsky, Godot proposals, GitHub.

[20] Bevy's contributing guide - Bevy repository.

No Longer a Blue Robot

> **Notice:** *the following is a personal story as expressed by the author of this book. The circumstances that have political flavor to this story is the straw that broke the camel's back. All comparisons are done for the purpose of characterizing Godot's toxic leadership by the principle of analogy.*
>
> *If you simply jumped to this page without reading the preceding chapters which explain the primary problem with Godot, you're strongly encouraged to read the book starting from the beginning, or the Overview section at the very least.*

Waiting for Blue Robot

As I've gained experience and kept contributing to Godot, the leadership has invited me to join their organization in 2021. This allowed me to talk on the topics which pertain to development philosophy, project governance etc. Of course, this is not to say that only privileged can talk on this topic, but my experience as a contributor empowered me to bring up those topics, as I truly wanted to improve Godot.

As I started to identify cornerstones that Godot lacks according to my research, I've started to receive negative responses from the project leaders. My attempts at clarifying the project's development philosophy and governance model were met with a push-back from the leadership, saying to me in private that what I'm trying to do is unacceptable, namely from Godot's project manager, **Rémi Verschelde**, saying that what I do is unprofessional. I replied that what I do is always in the interest of a consensus-building process which should not recognize compromises, because compromises lead to temporary coalitions and eventual division of community. Once I've explained my intention, I haven't received a response, so I kept expressing my opinion, because it's my right to do so.

However, after some months, Rémi removed my privileges as a maintainer. This decision was fairly unexpected. In the email I've received, Godot's Project Leadership Committee in the person of Rémi decided that what I say allegedly hurts Godot's image. Trying to appeal to this decision was impossible, since they refused to talk about this topic. The way I got treated simply did not warrant member exclusion, especially when I've been contributing to Godot's development for five years straight, and my merit as a contributor was recognized by community.

Later on, I started to receive threats of ban from all Godot communities. I figured that these

kinds of threats are disproportionate to situations that arise. I no longer felt safe interacting within the Godot community. This affected me to the point where I feared to express my opinion elsewhere as well. It's like the worst possible outcome coming true! I was at the point where I was threatened in private by Godot's lead developer, **Juan Linietsky** (quote):

> *1) Restraint yourself to only talk about technical topics and Godot development.*
>
> *2) Do not talk with anyone about your opinion on things that are unrelated to technical topics. This includes your opinion about other community members, your opinion about yourself, your opinion about governance, etc.*
>
> *3) Do not cite or quote anyone, for any reason, period. No citing, none, zero.*

Not following these orders would lead to an instant ban, as promised by Juan. I decided not to follow them because I adhere to Godot's *open discussion* principle as declared in Godot's governance model. Some months later after Juan's threat, I've received an email from Godot PLC, which contained this:

> *We think this attitude shows a bad image of the Godot project when a contributor who is part of the GitHub organization (thus with the "Member" badge) is seen arguing with other Members and at times overruling their positions.*

The mere idea of members within the GitHub organization engaging in debates and occasionally asserting their influence is simply outrageous! We can't possibly allow such behavior to tarnish the pristine image of the Godot project! Everyone involved in a collaborative project should share the exact same opinions and never dare to challenge each other. How dare these "Members" engage in healthy discussions and express differing viewpoints? It's almost as if they are exercising their right to independent thought and intellectual discourse! Unacceptable! 🙃

I hope you appreciate my emphasis on absurdity of Godot leadership. Jokes aside, it's as if they must maintain an illusion of unity at all costs! As if disagreement or conflict instantly signifies a project's failure and inability to function properly. Tribalism at its core!

They are afraid that someone will think bad of them, as if an opinion of a single person could destroy the entire project. Due to this, they tried to bribe me by giving a "Member" badge, but I wasn't actually interested in any of that. And because giving a "Member" badge didn't stop me from talking on topics related to, say, development philosophy, they just decided to take it away:

> *We decided to remove you from the organization for now. The "Member" distinction bears a lot of weight, and is thus also a responsibility to be a good representative of the project.*

Eventually, they *permanently banned* me from all Godot community spaces, under a pre-text that you'll find out by the end of reading this story.

Victimhood of Blue Robot Cult

Hello, my name is Andrii, and I'm addicted to Godot.

This is the first step. It took me years to realize this. In fact, I realized this as soon as leaders started to ostracize me from the project. I think most people that stumbled upon this in Godot would decide to leave the community on their own (and that's exactly what Godot leadership does, they first point to a door). But I decided to stay, because:

- I exercise my right to freedom of speech.
- I cannot let a group of people employ abusive tactics.
- I decided to learn how to deal with toxic leaders once I recognized them.

The entire situation led me to study psychology which allowed me to recognize manipulative behavior, and the existence of this book is a testament to my words.

The Godot community jokingly refers to this addictive phenomenon as "godotitis." However, let me assure you, this is a serious condition! If you find yourself suffering from "godotitis," I strongly recommend disconnecting from the internet for a while. This is not a matter to be taken lightly! It serves as a defense mechanism when they humorously refer to the community's cult-like aspects within Godot. If taken seriously, more people might break free from the comfortable illusions they have created for themselves.

There are numerous reasons why I decided to participate in and continue being a part of the Godot community. These reasons include:

- Gaining experience in game development and open-source projects.
- Feeling a sense of belonging within the community.
- Accessing a free game engine!

However, I eventually became a follower of Blue Robot. I used to have an unnoticeable aversion towards individuals who advocated for engines other than Godot or criticized Godot itself. My perspective was that these individuals were firmly entrenched in their familiarity with their existing tools, making them reluctant to embrace GDScript or explore new possibilities.

I used to dismiss anyone who claimed that “Godot is a cult” as simply envious of the engine. In my mind, those who opposed Godot were merely negative individuals. I firmly believed that Godot was the ultimate game engine, destined to become “the next big thing” akin to a gold rush; all we needed was a little more time. Consequently, I held the notion that it would be ideal for Godot to become the sole open-source game engine available, thereby allowing the development of a unified ecosystem of tools and plugins.

Reflecting on it now, I realize that this perspective appears naïve. However, that’s precisely what I used to believe.

Escaping Blue Robot Cult

The war allowed me to see everything crystal clear. The war breaks illusions.

Talking about Russian invasion of Ukraine, it was actually the cause that lead to my permanent ban in Godot as well. On 24 February 2022 at approximately 4:30 AM, I’ve heard loud explosions. My heart started to beat like hell with each explosion. Later in the morning, I turned on my PC and I saw the news. I contacted my friends online, and of course entered the Godot chat to notify the community about this event… I said:

> *Russia started to bombard Ukraine. I live in Ukraine. Stay strong fellows.*

I did not mention anyone specifically. Almost immediately, Rémi reacted to my message, saying:

> *I hope you and your loved ones stay safe. This is such a horrifying situation, there’s no words.*

Of course, “*there’s no words*”. Just like a hypocritical politician, Rémi wanted to *save his face* in the situation where he just removed my privileges months ago. The fact that Rémi was the first one who replied to my message means that he put myself in the watch list in order to surveil my every action in Godot, I was already on their radar.

Several weeks later, Bucha massacre happened. When the battle for Kyiv was over, I decided to visit that place. I saw everything with my own eyes.

My mind was so focused on the war and survival that I stopped interacting with Godot community. I realized that my life is more important than just being part of some online

group. But at some point I decided to contact Juan to express all my feelings and my relationship with Godot. I already knew at that point that Juan is a skillful gaslighter (you can figure I was very tolerant of their behavior).

I started my conversation with him on this topic:

- Promote development of extensions for Godot

Essentially, I wanted to let him know that if they don't welcome me within the Godot community, perhaps they would be interested in cooperating, and I'd lead another project, so that most useful unmerged and rejected features could have their place somewhere, especially this is how other open-source projects tend to cooperate, where a project promotes useful addons in order to build and consolidate the community around it. His reply was this:

> *There is zero incentive from the Godot project to promote features that were not merged (yet or ever). I'm pretty surprised about your perseverance on this topic for so long. I think I already made it clear that we will not recognize your project nor allow you to do strong promotion of it using our own resources. I have no idea why you believe you can convince me or "find a way or a backdoor" into this. You will not find any no matter how hard you try.*

Juan's defensive reaction was unwarranted, considering Godot's project manager's claim that there's only "we", implying that there's no hard distinction between official Godot and its community. Competition or promotion wasn't my intention or argument. I created the community-driven Goost project to *complement* Godot. However, regardless of expressed intention, they perceive such projects as a threat to Godot.

This is because community-driven projects like Goost provide additional features for free, which likely goes against Godot's agenda to "complement" Godot through their commercial for-profit venture, W4 Games, which they co-founded without the knowledge of the Godot volunteers, contradicting Godot's non-profit nature. At that time, no one was aware of their plans. According to this post, Pablo Selener, the managing director of W4 Games and a former employee at Epic and Tesla, is a school-age friend of Juan Linietsky, who is a co-founder of Godot. You can read Pablo's post that aims to attract Unity users.

I told him that if he didn't want to discuss this topic, I asked him to inform me about this, and honestly, the reason I continued talking about it for so long was because I was in a bomb shelter. I decided to distract myself, so I let him know about this. He responded:

> *I can imagine its hard and you have my support.*

I asked:

> *What kind of support are you talking about?*

He said:

> *I mean that I totally understand the situation you are in, and that it can be very stressing, and that you are a victim in all that is going on.* ***I don't think there is a lot I can do for you specifically*** *[emphasis added], but I meant to say that I fully believe what you are going through is real.*

Then I told him that I've already figured this out! I don't need an external confirmation that what I see is real, as I see the war with my own eyes. That wasn't helpful. For some reason he started to explain why they haven't banned me so far, and that *I should be thankful* that I still keep interacting in the Godot community. He said:

> *If you cross the line you cross the line and we won't be able to protect you.*

I have found his rhetoric quite disrespectful, and when considered with other reprehensible behavior of Juan in Godot, and in the context of war that we've been talking about, I couldn't resist to compare his own *rhetoric* to Putin's one! Then he said:

> *The project is fine without you, it does not need you attacking the leadership to try to change things and just like nobody backs Putin in Ukraine, nobody backs your point of view within Godot.*
>
> *So really, don't do these stupid comparisons when it's obviously the opposite for you.*

What he said was problematic on so many levels!

- All I wanted to do is to be able to freely share my opinion. I feel sympathy for people who had to go through the proposal process, make a pull request for a feature, and get rejected. So I wanted to find a way to help those people find a place. The Godot leadership could avoid all problems by literally leaving me alone with ideas that I published publicly, or they could choose to just ignore me; in fact, I rarely reached out to project leaders, I mostly talked to community.

- There exist numerous people who back up my point of view. So it's just a blatant lie. See for instance the Development Philosophy proposal, even when such people don't always express explicit support.
- I advocate for *open discussion*, the principle that Godot publicly declares on their website. If people are restricted in the way they must speak, then that's not what constitutes an open discussion.
- Godot leadership must not forget that their so-called organization has entered the **democratic world**. Godot is *not* the center of the world!

Incidentally, the entire Godot community reminded me of Orwell's "1984"! Recall the Thinkpol? I'm just a dissident and an extremist in the eyes of Godot leadership! But I have absolutely no power other than being able to influence people with words.

So, their structure of logical reasoning comprises mostly of manipulation. This kind of reasoning may be interpreted as a powerful charisma, unfortunately. Putin also presented himself as a quite charismatic and a loving person to the entire world, until he decided to attack Ukraine. The problem is that Putin is still seen as a hero by some. That's how powerful brainwashing could get.

Later on, I said to him that my behavior is the behavior of a free person, and he does not have any right to restrict my way of expression. I also mentioned that this kind of attitude is an inherent trait of a lot of Ukrainians, due to our culture and mentality:

> *You say that Godot is a "Worldwide organization". So why do you think that you're entitled to impose a particular behavior as you please?*
>
> *We,* ***Ukrainians, fight for democratic values now*** *[emphasis mine]. It appears to me that you were raised in a country with quite totalitarian laws. Apparently, you have a long way to go.*

Context: Juan was born in Argentina, a country that tends to align with a pro-Russian stance. For instance, on February 3rd, Alberto Fernández, the President of Argentina, stated, "*Argentina has to be Russia's gateway into Latin America.*"

Then, Juan said:

> *This has nothing to do with Ukrainians, this only has to do with you*
>
> *Don't shield yourself in people who are suffering*

> *If you behave in an unacceptable way, it's about you, not about Ukrainians.*
>
> **You don't represent Ukrainians to me.** [emphasis added]

I thought: "Wow...". I hope you understand what he said was unacceptable:

- I **am** suffering from the war.
- I **am** an Ukrainian, a citizen of Ukraine, have an Ukrainian passport, and even authentic Ukrainian surname. I'm a Cossack descendant!
- I **do** represent Ukrainian culture and values. In the context of what I said, that's exactly what I conveyed.
- I **do** represent Ukrainians in Godot community. I was pretty much the only active Ukrainian maintainer in Godot.
- Even if we assume for a moment that I'm an ass, he still has no right to say anything like that as a project representative.

There's even a blog post published on February 25, 2022, that says, [emphasis added]:

> *I don't know if anyone from Ukraine is going to read this, and there's not much I can do anyway, but still: Stay safe out there in this fucked up times. I know of at least one **Godot contributor from Ukraine, Xrayez**, and I hope he's alright. And fuck Putin*

When people think about Ukraine in relation to Godot, the first thing that often comes to mind is my role as a Godot contributor. In light of this, ask yourself: Am I effectively representing Ukrainians within the Godot community?

A year ago, Juan said to me:

> *Not pissing others off is more important than trying to understand others.*

But he failed not to piss *me* off! 😁

From that moment on, taking into account other questionable and outright unacceptable behavior of Godot leadership, I decided that I'd never contribute to Godot again. However, due to my addiction to Godot, I've found various excuses to continue *using* the product.

Russians and their influence on Godot leadership

> 💡 *I'm not against Russians specifically. I'm against everyone who kills people of Ukraine, and I'm against everyone who justifies actions or inactions of Russian citizens that keep supporting Putin's regime, who are collectively responsible for letting the war happen. Unfortunately, the majority of people who live in Russia do support the atrocities that happen in Ukraine now, and some Russians around the world infect other people with fascism propaganda. I know this as an Ukrainian.*

There exists one Russian member in Godot, called Yuri Sizov, who possesses a strong obstinacy and promptly dismisses any opposing viewpoints. Additionally, Yuri tends to embark on prolonged rants and discussions in an effort to prove others wrong. Regrettably, Yuri has been engaging in harassing behavior towards both myself and other members within different Godot communities. Several individuals concur with my sentiments regarding this matter.

When I said anything that he interpreted as offensive, Yuri would report on me to Rémi or Juan, and I'd get bullied: "*You cannot do this. If you keep doing it, you'll get banned*". But when I explained how Yuri's behavior is problematic, Rémi would say: "*I don't see nothing wrong with what he says or does. I realize that he can be a little confrontational, but he can say 'fuck all who think otherwise' if this is done in good faith*".

That Russian member joined Godot community in 2020. I've been contributing to Godot's development for like three years at that time, and Godot's leadership haven't even recognized my effort (I didn't expect nor asked for this), while it took him only like a year to get full write access to the repository and get special permissions. He was accepted into the inner Godot cult with ease! It may sound strange, but his offensive attitude to people allowed him to do this! By offensive I also mean *defensive*, since he worked hard to spread the word of Godot and shut down everyone who "*spread misinformation about Godot*", according to his words.

Rémi has recognized that Yuri and me "*seem to have frequent tensions,*" so he suggested me to contact him via PM. So I did. To summarize, Yuri ended up blaming me that "*I'm the cause of all conflicts, and he wondered why*", basically repeating the same slander that he likely heard about me in a private channels at Godot Contributors Chat. The most interesting part was his last words. He said (translated from Russian):

> *And don't speak to me using "Mr.", we're not in* ***Verkhovna Rada*** *[emphasis added].*

For your information, **Verkhovna Rada** in Ukraine is equivalent to the **United States Congress** in USA. What kind of insinuation is this? Well, he was perfectly aware of existing

conflict between Russia and Ukraine and that I'm an Ukrainian. Why would he find it absolutely necessary to touch politics? That was in 2021.

He changed his nickname right after Russian invasion of Ukraine in 2022. I didn't realize why, but according to his Twitter, he said that he doesn't want to be associated with the war. Turns out that, if you read his old nickname "pycbouh" in Russian, interpreting Latin symbols as Cyrillic, as in "PycBouH", this literally means: **Russian Warrior**! Problem: "Russian Warriors" call me "Ukrainian Nazi." A lot of questions, so I kept researching.

I've stumbled upon a news article (2021):

- Godot Engine receives $120,000 grant from game development studio Kefir

Kefir is a Russian game development studio. Everything seemed to be fine, until I read the quote presented there. The author of that quote is **Alyosha Stalin** - Chief Business Development Officer at Kefir, from Volgograd. Here's an explanation of the name:

- Stalin is a Soviet political leader and dictator who caused massive genocide called "Holodomor" of Ukrainian people.
- "Stalin" is not a Russian surname. It's a pseudonym for "Man of Steel", similarly to "Lenin".
- "Alyosha" is just a byname of the full name "Alexey".

This means that Godot has accepted a donation from someone who *adores* Stalin. Does that mean Yuri is from Kefir as well? I don't know. My point is that Godot appears to attract Russian game developers. Why? Because they likely share the *governance model* of Godot and find it as a perfect environment where their behavior can be tolerated, and where their behavior could even thrive!

Goodbye, Blue Robot Cult

My final day at Godot was at the moment when I watched a video by our Ukrainian spokesperson Oleksii Goncharenko. I shared the video at **#coffee-break** channel at Godot Contributors Chat, and presented it *as an example* of free opinion voicing.

Some minutes ago, someone from Godot members said:

> *Well, the main conclusion I seem to draw is that you are trying to make a connection with your exclusion from the [Godot] org with things that are quite different and an order of magnitude more impactful. I think it's a quite an unfortunate comparison, which, in a way, kind of make the point of the [Godot] PLC decision.*

Do you see how they refuse to accept any kind of analogies? This kind of manipulation technique allows them to sustain the current status quo, and this allows to defend their cult from the outside intervention. This way of thinking is apparent when Juan asks that people should not compare Godot to any other technology.

Our conversation quickly switched to the topic of human rights, and I've explained how Godot violates the right to freedom of speech (reminder: they forbidden me to quote anyone, under any circumstance, I'm not allowed to talk about other members and even myself).

Then, Yuri appeared! He said:

> *I have to say that anyone's involvement with the project is **not a part of any free speech paradigm** [emphasis added]. Anyone can be directly (by a personal "firing") or indirectly (via the established and agreed upon terms of engagement **that make you quit** [emphasis added] removed from the project, at least in the official capacity. There is **no inherent right to contribute to Godot, so the org can oust anybody for its arbitrary reasons** [emphasis added].*

Again, Yuri is the official member within Godot organization, accepted by Godot leadership. The leadership agrees to what he says. Let's see why Yuri's stance on this is quite problematic:

- the right to freedom of speech is seen as some kind of disadvantage (to whom?);
- there exist procedures that "make you quit" on your own in Godot (cyberbullying, pointing to the door, boycotting, cancelling... you name it);
- everyone should feel welcomed to contribute to the project right from the start, so people do have inherent right to contribute to Godot;
- people can be booted from the project for any reason, regardless of morals.

Therefore, the approach described by Yuri is fairly unprofessional, no *healthy* open-source community-led organization would do any of those things.

He continued saying *[emphasis added]*:

> *We just choose to **trust** that those reasons are valid and are upheld in the interest of the larger community. Every work is based on **trust**, and hard as you might, **trust** may just disappear one day, or never appear in the first place.*

Do you see how they keep fixating on **trust**? After my long reply regarding the Godot's principle of open discussion and democratic world and the human rights that represent the democratic world, Yuri replied:

> *Godot is open-source, but it's not just anyone's. There are specific leaders. **You either align with them or not** [emphasis added].*

I replied to this expression of authoritarianism this way:

> *What can you expect from a person who lives in Russia...*

You see, I can understand why he would think that there's no other choice. He lives in Russia, where cult of Putin and his regime does not let him express his opinion in general, not even mentioning about the war. But why this way of thinking has to be spread inside Godot, an open-source, community-driven project? Toxic leaders must be challenged, otherwise those leaders will rule the world of fascism.

He said:

> *I am very much anti establishment.*
>
> *But Godot isn't a country.*

They completely lack ability for critical thinking! Obviously, Godot is not a country. I explained to him one important thing to realize. Due to his trouble-making, condescending, arrogant behavior in Godot community, I said:

> *A person who lives in a fascist country is infected by fascism and spreads it, no matter where that person is located. If you don't believe me, google up "14 signs of fascism" and ask yourself: in what kind of country do you live?*

You can verify this for yourself if you read the book called "Pandemic fascism" and listen to

people such as Alexander Nevzorov, a Russian television journalist, who literally says that, being a Russian citizen, he overcame fascism within himself.

Speaking of fascism and Russia, there's one thing that I haven't revealed in the email that I've received from Rémi, which contained the following defamation:

> *We do think that you are well-intentioned, and we understand that you **struggle with mental health** [emphasis added]. But you also seem to be well conscious of your issues, and we think that you can continue to grow alongside the Godot project and better understand what kind of interactions are good, and what kind are harmful.*

This reminded me of Soviet Union, where political reformers were labelled to have a made-up sluggish schizophrenia as a method to oppress the opposition. Cults also employ similar methods to further ostracize non-believers. This is because destructive cults and fascism overlap on the level of bigotry.

Anyways, they thought that I'd like to establish a democratic way of organization in Godot. But I wasn't implying that at all. Most open-source projects are based on *do-ocracy* indeed. When I talk about democracy, I meant that if you enter a democratic world, you must abide to the *democratic principles* on the level of basic human rights.

Then one of the members said:

> *LOL, and what exact right did we remove you? Free speech? You are still talking here. And no one prevents you from bringing your opinion on twitter or else TBH. But even then, **free speech does not apply here** [emphasis added], and I really hope it will never do.*

Apparently, Yuri got offended by what I said. He didn't tell me this, but all of a sudden, Rémi appeared! He said:

> *It's the second time today you're overtly racist Xrayez. This has to stop.*

There was no genuine explanation or proof of accusation. Zero dialogue. At first, he accused me of racism. Then, other member suggested another term: xenophobia. Rémi agreed that: "Maybe better term yes." Or "extremism", as Maria Zakharova says. 😉

They couldn't come up with a specific accusation because they didn't really care at this point. Moments later, Rémi said:

> *I give up trying to reason with you. We can't reach an agreement on how to work together, so we shouldn't work together. You don't want to make things work, and **you don't want to leave** [emphasis added], so the only option left for us is to ban you from Godot communities.*

I said:

> *What next? I'm from Ukraine btw.*

Rémi replied:

> *That saddens me because I estimate you as a person and I'm appalled by the situation you're in your real life, **but we can't help you** [emphasis added].*

Ask yourself: if you were to manage an international project, knowing that there exist both Russian and Ukrainian contributors, and knowing that Russia is the aggressor that caused literal genocide of people in Ukraine (both in the past and now), what would you do in a situation like this as a project manager?

If Rémi cared about me, as the only active Ukrainian contributor in Godot, where my merit is recognized by community, he wouldn't decide to permanently ban an Ukrainian. Is this how they express sympathy and compassion? No. This is a clear example of Putin's rhetoric. Just a blatant lie and hypocrisy. The sympathy that Rémi allegedly expresses is the sympathy of an executioner who raises an ax over the head of an innocent person.

Yuri was likely pleased with such decision, by the way!

> 💡 Consider reading this thread, which analyzes Yuri's troublemaking behavior, who is trusted by both Juan and Rémi.

Even though Rémi banned me, I do have friends in Godot community that shared this message after Rémi decided to ban me:

> *Note for readers: Xrayez has been banned from this platform and other Godot community platforms. This was a long time coming after many instances of discussions similar to the one above, both publicly and privately. As we mentioned above, if a contributor is really incapable of behaving in a way that's compatibility with the existing community of*

> *contributors, they end up leaving by themselves - or being shown the door. We're always open for discussion, and this has been discussed a lot - but at some point we need to reach a decision.*

This basically means that civil discussions on Godot's governance will *definitely* lead to a ban, and Rémi presented it as if I had troublemaking attitude towards Godot contributors at large, which is certainly not the case. Prior to the ban, several people in Godot community would confirm that I have shown no signs of troublemaking behavior. For example, here's one reaction by a very active Godot user that follows Godot development quite religiously, when he found out that I'm permanently banned now:

> *Offtop: whoa what happened, he was one of the contributors whose handles I recognized (therefore very active) and I haven't seen him be rude ever...?*

But of course, cult leaders' task is to suppress doubts within their cult community by spreading slander in official capacity. Other members reacted to the ban and asked Rémi:

> *Is it a permanent ban?*

Rémi replied:

> *For now yes. There's always the possibility to appeal but I have little hope in this situation.*

How is it even possible that a *permanent ban* is *temporary*, considering that I am deemed "incapable of behaving" according to Rémi's personal double standards? What about my private report of Juan's violation of Godot's Code of Conduct, which Godot PLC conveniently ignored? With Godot, hypocrisy has not limits.

After all above events, I did not ask to remove the ban. What I did, however, is to contact Godot leadership for an request to deliver their official stance and the reason for the permanent ban via email, just like all responsible people do. They haven't responded. Zero dialog. Nothing at all. There was no possibility to appeal, so Rémi lied to the entire community of contributors. In other words, they *cancelled* me.

When I got permanently banned, do you know what I felt?

The worldwide community has started discussing and sharing the book on social media. The Ukrainian community is extra supportive of the points I am making. Unfortunately, it appears that Godot's toxic leadership does not seem to embrace the Ukrainian mindset and views us as confrontational. However, all we are doing is advocating for our right to freedom of speech and expression, as well as our right to exist as a nation. This support extends beyond Ukraine as well. Sadly, Godot's de facto pro-Russian stance, rooted in Argentina through Godot's co-founders, effectively marginalizes the Ukrainian community within Godot.

You can also read my comprehensive article on GameDev DOU, a Ukrainian community of IT specialists, written in Ukrainian:

- Чому Godot Engine проти України: російський вплив на модель управління Godot та Open Source (*Translation: Why Godot Is Opposed to Ukraine: Russian Influence on Godot and the Open Source Governance Model*).

Justice Manifesto

The following message is primarily directed towards **Juan Linietsky** (reduzio) and some of his sub-leaders, including **Rémi Verschelde** (Akien) and **Yuri Sizov** (PycBouH, Russian Warrior), as well as Juan's loyalists. If you've been wronged by these robots pretending to be individuals with a high moral ground, you'll likely find this manifesto quite gratifying. 🙃

A Call to Conscience

Juan, just as you possess extensive knowledge about your *game* engine and how it operates, I am well aware of the manipulative *games* you play with the people you come into contact with. The way you handle software internals ironically mirrors your careless treatment of people, both online and in real life.

I absolutely know what you're doing, and I see right through you. Even though I may not be able to restore justice all by myself, the already invented wheels of justice are turning against you. I am committed to ensuring that the truth comes to light and that the appropriate actions are taken at the right moment.

While you expect others to be responsible, you show no signs of responsibility. Your immensely hypocritical behavior has not gone unnoticed, and ***there are those who will tirelessly work to hold you accountable for the harm you have caused***, both due to your ignorance and deceit. If you were smart, you wouldn't have individuals like me so dedicated to investigating, documenting, and exposing your lies.

I will work diligently to expose the truth, gather even more evidence, and cooperate with the appropriate authorities if needed, such as the Anti-Fraud Office. It is my duty to ensure that the truth prevails and that those who engage in fraudulent and discriminatory activities within an allegedly worldwide organization such as Godot Engine, which exploits the free labor of individuals who are unknowingly working on the personal agenda of toxic leaders, face the consequences of their actions.

Justice is not about revenge or personal vendettas. Instead, it is about upholding the principles of fairness and the rule of law. Everyone deserves a fair and just process, including you, Juan. Because of your lack of empathy, you may believe that you can evade the consequences of your actions, but justice is not merely the responsibility of one person; it is the collective ethical duty of society. As the voices grow louder, the strength of justice becomes undeniable.

No one is above the human law, even if you think of yourself as a boss. I know that you try

to convince others that you're not a boss, but the course of justice will not be swayed by your deceit or defamation of people who challenge your authoritarian behavior and governance. While justice may sometimes be slow, the truth will eventually be revealed, and those who have been wronged will find solace knowing that justice is restored. It is not a matter of if, but when.

As you continue exerting undue influence over your followers and perpetuating slander that damages the reputations of those whom you have harmed already, remember that the pursuit of justice is relentless. Even though you may have a considerable number of overzealous followers who unquestioningly support you and are seemingly willing to protect your image, bear in mind that their allegiance is primarily driven by their own desire to maintain the narcissistic image that you managed to craft yourself. However, when faced with facts, evidence, and testimonies, these followers will eventually have to confront the truth and go through all the stages of denial, shock, acceptance, grieving, despise, contemplation, and finally, moving on. Before moving on, some of them will have the desire to restore justice, just like I do. With the information I possess, they will come to realize that they are not alone, contrary to what you claim to your dissenting followers.

Juan, as well as every devoted follower like Rémi, who participates in eroding the integrity of others just to maintain the status quo, let me propagate this message clear: ***the force of justice cannot be ignored and will inevitably find you***. Unless a conscious decision is made to change yourself and accountability for your actions is taken, the series of events that you have set in motion will ultimately result in your downfall.

From Silence to Truth

"When you have something to say, silence is a lie."

Despite my extensive involvement with Godot, I have refrained from expressing these concerns until now. This hesitation was rooted in the cultic nature of Godot as an organization. Throughout all those years, I remained oblivious to this social dynamic, especially while dealing with ongoing *undue influence* originating from Godot's *toxic leadership*. This fear of speaking out is shared by other members of the Godot community, who, like me, have been reluctant to openly share their experiences. Even those within the private **Godot Advisors** group share some of these concerns! They will surely find a way to speak up, one way or another.

Those who keep waiting for Godot aren't interested in finding the truth. If you come across comments from overzealous Godot users, supporters, believers, advocates, or apologists of Godot attacking critics, including the author of this book, understand that their primary

intention is to defame us. This speaks volumes about Godot's scapegoating practices.

Ironically, while refusing to read the book, waiters for Godot demand proof of Godot's misdeeds, oblivious to the fact that their own toxic behavior serves as the actual evidence substantiating the accusations against them. For example, you might be reading this because someone shared it with you, presenting it as "check out this crazy, mad, insane child," or something similar that doesn't quite align with Godot's Code of Conduct, or should we say, Code of Misconduct. Most of these bigots won't even read this part, just as they haven't read the book – unlike you, the curious reader!

Godot fanatics often take offense, claiming that critics act in bad faith, driven by bias, vengeance, anger, arrogance, or even accusing them of being corrupt and concocting conspiracy theories. Some even go so far as to suggest that these critics need professional help! However, these labels are merely projections or rationalizations by those making such insinuations and baseless accusations. The capacity of the human mind to engage in denial and rationalization can be impressive. Be assured, their obsession with labeling critics "insane" won't stop us from continuing to inform about Godot shenanigans. Consequently, their cultic behavior becomes a source of amusement.

My assertions are grounded in factual information, evidence, and the testimonies of others, coupled with my extensive personal experience and expertise. As of now, only a few individuals are willing to challenge Godot's toxic leadership, because they want to prevent potential harm to their own reputation. What many fail to recognize is that uncovering the truth about Godot will lead to them earning a favorable reputation in the long run. As a former maintainer of Godot, I'm not afraid to talk about this because I have many years of experience with Godot speaking on my behalf, so I'm confident in what I have to say about Godot. But even now, there are enough notable testimonies that enable us to see a more objective picture.

I encourage you to form your own judgments primarily based on the content within, and only secondarily from others. If you find this book enlightening and informative, don't hesitate to share it with anyone you think might benefit. If you're passionate about restoring justice, this is definitely a step in the right direction. Thank you!

Afterword

The cult leader of Godot keeps insisting that in order to be included in the decision-making process, individuals must earn his trust. What kind of trust are we talking about here?! I never asked to be “promoted” as if in a company. All I wanted was to improve Godot in a way that benefits the majority of people and actively participate in the development process, regardless of my position.

Having been a former user, contributor, developer, maintainer, and a survivor of the Godot cult, I can confidently say that Godot is one of the most toxic cults in the open-source game development industry, exploiting people’s desire for belonging. Individuals end up dedicating their lives to a project without a vision, only to be kicked out when they attempt to challenge the toxic leadership after gaining experience. Meanwhile, Godot continues to receive donations from sponsors who are deceived just as the leadership deceives their followers. By contributing money to Godot, you are indirectly supporting the exploitation of contributors. Whether Godot’s leadership is aware of their decisions or not is irrelevant.

I strongly assert that Godot does not adhere to the principle of open discussion, despite appearances. Discussing governance, decisions, and questioning the actions of members is heavily discouraged and met with punishment. I decided to raise awareness of this issue so that people do not fall into the mental trap that is Godot.

Having been deeply involved with Godot, it was incredibly challenging for me to break free from the cult mentality. Thankfully, I had other people in my life who helped me recognize the true nature of Godot and why continuing to participate in this community is unhealthy. If my experience can help others who find themselves addicted and wish to leave the Blue Robot Cult, then that’s a totally positive outcome.

My goal is not to gain popularity but to raise awareness of the problem. I am willing to potentially sacrifice my own reputation for the greater good, and the fact that I am writing this book serves as a testament to my commitment. I will do everything in my power to bring attention to this issue. I have taken on the mission of restoring justice, and this book is a tangible manifestation of that decision.

Cult members will never tell you the truth, primarily due to their own ignorance. They will claim that those who speak negatively about Godot are bad, biased, corrupt, or even mentally ill. However, it’s totally up to you if you really want to participate in this cult community, but bear in mind that I’ve been there. If possible, I suggest staying away from it. If you still wish to use their products, I recommend avoiding their core engine development communities, and I strongly advise against giving them any financial support. Money and new contributors fuels the growth of this cult, and it becomes particularly mentally destructive for contributors who genuinely want to see Godot’s mission, vision, scope, and

philosophy defined, as it's done in healthy community-led open-source projects.

The Godot community expresses a desire to prevent division, but achieving this requires defining and documenting the mission, vision, and development philosophy of Godot. However, the Godot leadership refuses to do so and even punishes contributors for discussing Godot's governance. As a result, division and difficult-to-resolve conflicts are inevitable in Godot community.

Even if someone decides to create a fork of Godot, it is likely to face attacks from overzealous Godot fans who perceive it as a threat to Godot's existence, regardless of the positive intentions behind the fork. Ironically, this hostile and ambivalent attitude is what drives others away from the Godot community, turning it into an echo chamber. Professional developers prefer to choose other game engines rather than dealing with Godot because they seek a tool, not a cult. They would rather not associate themselves with Godot, especially considering how pervasive and distorted its ideology has become.

As a former maintainer of Godot Engine, looking back, I'm not proud of my close engagement with it. Merely being associated with anything related to Godot might negatively impact my professional growth—not because of what I have to say about Godot, but because Godot has become synonymous with parasitism. Those infected by Godot's braindead ideology may stop seeing the industry realistically and view those not supporting their ideology as outsiders who must be ostracized at all costs. Luckily, with the support of people out there, I've managed to overcome my Godotitis, at least to the point where I can freely talk about Godot in this light without the anxiety of speaking up.

Having witnessed Godot's hypocritical actions firsthand, I've noticed that the toxic vibe in the Godot community spills over and impacts other game development communities. Contrary to a common misconception, I'm definitely not here to salvage those who already became victims of Godot's brainwashing. However, it's my ethical duty to inform others about the potential dangers that Godot may pose to the longevity of other communities.

I've been digging deep into Godot's public relations, and not to toot my own horn, but I've probably gone further down the rabbit hole than anyone else. I've been keeping tabs on how people react to Godot and how different communities handle criticism, both in the world of FOSS (Free and Open Source Software) and in commercial software. Now, here's the scoop: there are some FOSS communities that don't exactly give Godot a standing ovation. Turns out, Godot doesn't play by the FOSS rulebook from the get-go, considering Godot is de facto a company cruising along with a non-profit, tax-exempt label. Unfortunately, the whole anti-corporate vibe in some FOSS circles tends to tilt the support towards Godot. Of course, not every FOSS community is waving an anti-corporate flag. But here's the kicker—Godot isn't just a game engine; it's also this symbolic, oh-so-elusive "bright future" mental construct and concept. And that, my friend, is pretty darn alarming.

What makes Godot's situation even more confusing is how darn welcoming they are to newcomers—it's like their superpower. You don't really get a sense of Godot's darker side until you're right in the thick of it, which makes it tough for folks to grasp the real deal with Godot. Even when folks start feeling the negative vibes, they might not connect the dots within Godot. The Godot loyalists are slick—they manage to convince doubters that they're the sole witnesses of Godot's problems and that their perceptions are just so-called misunderstandings. Basically, the Godot crew is pro at using gaslighting tricks to shut down dissent, especially when it's a bunch against one. By the time you realize you're in a trap, it might be too late, and you've spent years waiting for Godot.

Those who swing the notorious ban hammer often seem to have a soft spot for Godot, believe it or not. Yep, run into a moderator in some random game development community, and you might just find out they're hardcore pro-Godot fanatic. Say anything remotely negative about Godot, and boom—they go into full-on rage mode. They shut down any Godot criticism pronto, making sure the Godot status quo remains intact, even outside the Godot bubble. They come up with all sorts of flimsy excuses, like calling criticism speculations, conspiracy theories, or just plain crazy talk. And they don't even care that their baseless accusations often go against their own community guidelines.

In the realm of entertainment, the good stuff should be rooted in timeless archetypes. However, suppose you pay attention to the woke culture, identity, and intersectionality politics. In that case, you will hopefully understand their negative impact on our ability to produce something that brings true meaning to our lives. Godot Engine is a major player in this mess, especially with its habit of banning folks left and right.

It's pretty clear that Godot's undue influence is causing some havoc in the game development community, and it's not keeping to its own boundaries. Nope, it's spreading to unexpected places. The leadership at Godot is drawing in people who aren't exactly veterans in game development, and it just so happens that this lines up with the younger crowd of developers. You know, those who might be more prone to falling into the cult-like mindset. It's wild, isn't it?

So, here's the lowdown on why Godot is basically The-Game-Engine-That-Must-Not-Be-Named. But, let's face it, it's out there, and turning a blind eye is no walk in the park. Do we really want to live in some Orwellian nightmare? I'd say no thanks. Better spill your thoughts while you still can and it still adds up.

Given Godot's status quo, it is fair to say that Godot should not exist at this destructive cult state. Therefore, if they are incapable of changing their attitude towards criticism, the only *healthy* option remaining is to completely abandon Godot. Despite my involvement in engine development and game creation, I have personally chosen to cut all ties with Godot and imagine that it never existed. I recommend finding other game engines and tools that do not

exhibit hypocrisy and deception in their development decisions, as these are the core problems plaguing Godot.

I have learned that nothing comes for free; everything has a cost. If you don't pay with money, you pay with your time, work, and even your dignity. People reach out to me and share their testimonies of how they were treated in Godot. If you're such a person, feel free to contact me. For quite some time, I've been exposing Godot's wrongdoings via #TruthAboutGodot, #GodotCult, and even #CancelGodotEngine hashtags, just like Godot leadership decided to cancel myself from the Godot community. But my efforts are not focused on destroying Godot but rather cult deprogramming of Godot. Otherwise...

The End

Alternatives

To Godot, or not to Godot, that is the question.

Jump to the list if you're looking for alternative, replacement, or complementary game engines, frameworks, and tools to Godot.

As someone who has been using Godot for years, I strongly recommend exploring alternative tools, whether you are currently using it or considering it for the future. Regardless of the reasons that might prompt you to switch to another game engine, such as feeling constrained by the engine's capabilities or perceiving it as a bottomless pit or lacking direction, the following alternatives presented aim to fulfill that void.

Source: Godot Subreddit

My hope is that these alternatives will inspire you with possibilities beyond what Godot could ever offer. These options may also facilitate a smoother transition, covering various aspects of game development. It's important to note that this list is opinionated and not exhaustive; it's biased towards addressing what Godot users may find lacking in the engine in contrast to other solutions.

List

💡 To ensure a well-informed decision, refer to General Recommendation section, as it's advisable to familiarize yourself with common misconceptions when choosing between game engines and frameworks, as well as considerations regarding 2D and 3D development.

Game Engines

- ***GameMaker*** (***GML***, ***GML Visual***) - A game engine primarily used for 2D games. Godot is de facto a direct competitor to GameMaker when it comes to 2D development. Both Godot and GameMaker have their in-house languages, GDScript and Game Maker Language (GML) respectively. Unlike GDScript, GML is more low-level, making it more suitable for professional game developers. While both are high-level languages, GML inherits constructs from C-like languages, unlike GDScript. GML exposes advanced data structures not available in GDScript, such as a grid or priority queue. For additional performance gains, the YoYo Compiler (YYC) can be used to produce machine code instead of interpreted code. Completely free for non-commercial use. There are also plans to add ***JavaScript*** support and open source various engine components.

- ***Defold*** (***Lua***) — A game engine primarily used for 2D games, originally advertised as the ultimate game engine for web and mobile. The community highly appreciates its robustness and stability, unlike Godot, which often experiences frequent breaks and numerous regressions between versions. Although Defold is more minimalistic compared to Godot, Defold doesn't attempt to reinvent the wheel. Originally owned by King Limited, it was later released to the public as a source-available project under Defold Foundation. Defold provides official integrations for mobile monetization services, unlike Godot, which offers this unofficially via Ramatak Inc by Godot's co-founder on the side, creating a conflict of interest, see Companies vs FOSS.

- ***Stride*** (***C#***) — A game engine with a focus on realistic rendering and VR applications. It's important to note that comparing Godot to Unity is a common misconception. In reality, when looking for a close alternative to Unity, Stride stands out as a strong contender. Stride offers a comparative guide for developers with previous Unity experience, highlighting its familiarity. In contrast, Godot has removed all mentions of Unity from their latest documentation. Unlike the C# version of Godot, Stride can be used as a library, powering projects such as vvvv, a live-programming environment. Stride is a member of the .NET Foundation.

- ***Open 3D Engine*** (***C++***, *Lua*, *Python*) — A versatile development engine for game, simulation, and multimedia creators. Unlike Godot, O3DE has built-in terrain support

and PhysX integration, features that Godot's leadership refused to accept for years despite community requests. The initial version of the engine is an updated version of Amazon Lumberyard. Ironically, even Godot's lead developer himself unwillingly recommends using O3DE if you want something made by industry engineers! O3DE is a member of the Linux Foundation.

- ***Flax Engine*** (***C++***, C#) — A game engine with seamless C# and C++ scripting and hot reloading, with a focus on performance. Provides Visual Scripting, the analogous feature which Godot unfortunately discontinued and removed from Godot 4.0. Flax offers a comparative guide for developers with previous Godot experience. Flax Engine is particularly appealing to those who have experience developing Godot modules and GDExtensions with C++, as it offers robust C++ capabilities. Interestingly, the author of Flax Engine has previously criticized Godot for its subpar performance, albeit indirectly and somewhat ironically.

- ***Wicked Engine*** (***Lua***, C++) - A game engine that prioritizes modern graphics development. Can be used as a C++ framework for your graphics projects, a standalone 3D editor, Lua scripting or just for learning.

- ***GDevelop*** - A game engine which is primarily designed for non-programmers and game developers of all skill levels, using event-based visual programming similar to engines like Construct, Stencyl, and Tynker. GDevelop is frequently mentioned by Godot users as an alternative to text-based programming, see also Simplicity. It was created by a software engineer at Google.

Frameworks, Environments, and Languages

- ***BeefLang*** — a game-oriented programming language and IDE, without 2D/3D editor. The intended audience is the performance-minded developer who values simplicity, code readability, fast development iteration, and good debuggability. The syntax is largely C#-like, with D semantics. The IDE launch speed is unprecedented and comparable to launching a Notepad! Beef's creator is a co-founder of PopCap games. Windows builds are prioritized over other operating systems.

- ***Lobster*** — a game-oriented programming language that offers a comprehensive set of functionality out of the box. It features a Python-style indentation-based syntax with a touch of C-style flavoring, provides a high-level interface to OpenGL functionality, along with support for ImGui and more. Lobster's creator is an ex-Google employee, who has started his own game development company to work on an unannounced voxel game using Lobster.

- ***Panda3D*** (***Python***, C++) — a framework for 3D rendering and game development.

Panda3D is not written in Python, it's written in C++, Python is only used for scripting, so it can be extended in a similar way to Godot. Many Godot developers have previously expressed a desire to write games in Python because of its vast ecosystem and language capabilities, as opposed to GDScript. At the same time, Panda3D and Godot share a common approach to error handling: they almost never crash on error, and much code is dedicated to the problem of tracking and isolating errors. However, unlike Godot, Panda3D is not a beginner's tool or a toy. Panda3D has been developed by Disney.

- ***raylib*** (***C***) - A simple and easy-to-use C library to enjoy videogames programming. It focuses on simplicity and, due to its nature as a C library, allows the community to integrate and utilize many language bindings. Highly customizable, this library is primarily aimed at those who wish to delve into low-level game programming. With officially supported libraries such as raygui, it allows for the creation of straightforward immediate-mode GUIs (see Graphical User Interfaces for general information about the types of GUI programming systems).

- ***LINQPad*** — an interactive playground designed for those who want to learn C# in an engaging and hands-on way. Especially useful if you want to learn C# game engines or frameworks.

- ***ShaderED*** — a powerful shader IDE that surpasses Godot's shader editor in terms of capabilities and debugging functionality.

You can also explore the Top Engines used on itch.io. However, bear in mind that the statistics primarily reflect the usage of game engines and frameworks for game jams, not necessarily commercial games. For even more complete list, see Wikipedia's List of game engines.

General Recommendations

Despite the abundance of various alternatives, including the ones mentioned earlier, many individuals have expressed a lingering attraction towards Godot, even after encountering its limitations and bugs. The primary reason why such individuals may struggle in finding a suitable alternative can be attributed to several factors, with the following being the most significant:

- The Internet is flooded with an overwhelming number of so-called recommendations for Godot. It is heavily promoted and advertised by overzealous followers and the Godot leadership themselves, even in contexts unrelated to game development, often creating false expectations for new users.

- Some game engines, frameworks, and tools present a potentially steep learning curve that may seem too daunting. While you may believe that you don't require a professional game engine, you still desire to have a cake and eat it too, or have the best of both worlds, which is very difficult to attain, if not impossible.
- You might hold a certain level of hostility towards commercial game engines due to various reasons, such as economic constraints, a strong anti-corporate sentiment, or a perception that such corporate software feels cumbersome or bloated.
- If you are deeply attached to Godot, especially if it is the only game engine you have ever used, you may be seeking an alternative that offers a 99% comparable experience to Godot. However, such an alternative does not exist, because every software is unique.

Some of the aforementioned factors originate from limiting *beliefs* (see Glossary). I encourage you to contemplate and eventually challenge those beliefs. One approach to overcome them is to embrace the *growth mindset*. Take a moment to ponder how you came across Godot and how much time you dedicated to researching and comparing various tools before arriving at your decision. It is possible that you may not have thoroughly explored other alternatives and simply jumped on the Godot bandwagon without much consideration. There is no intention to criticize your choice, but rather to underscore how susceptible we, as humans, are to being influenced into making decisions that may not align with our personal goals.

If you look beyond these recommendations, you will inevitably find a suitable replacement. When I mention "replacement," it doesn't necessarily mean finding a single product. Instead, try to find a **set of tools** that excel at specific tasks. In fact, "Godot" was initially just a set of assorted tools according to its pre-open-source history. Nowadays, Godot is marketed as an all-in-one solution, but none of its built-in tools can compare to dedicated solutions. In fact, Godot falls short in this aspect to such an extent that unofficial community frameworks or extensions like Goost and Godex were created in an attempt to address this gap.

However, regardless of the number of features added to Godot, this gap remains unfulfilled somehow. This can be attributed to several factors, including inherent design limitations arising from Godot's object-oriented tree/node structure as opposed to adopting a more modern approach like ECS (which Godot will never officially embrace), Godot's limited ability for deeper out-of-the-box Customization and "The One True Way" approach. If you happen to be a Unity user, one of the reasons why you might feel compelled to switch from Unity to Godot is due to the "grass is greener" effect. This is also why I recommend users who have already invested years in Unity against prematurely switching to Godot.

Programming Languages

If you aspire to be an indie game developer, it is advisable to learn C# or, at the very least, get familiar with it. Avoid delving into low-level languages such as C++ or Rust unless you're working on performance-demanding games. The reason is that you might end up investing a lot of time contributing to game engines that utilize those languages or creating tools for them, which could lead you to gradually abandon the development of your own game. While engine development can be fascinating, especially when working on tools that grant a feeling of completion and satisfaction, it shouldn't overshadow the primary goal of making games. However, if you are interested in specializing in certain aspects like domain-specific languages (DSL), which are common for those who enjoy modding and enabling user-generated content, that can also be a rewarding pursuit within the game development field.

For those who still think that Godot's scripting language is unique and wasn't inspired by anything, I hasten to enlighten you. Godot didn't have GDScript right from the start. It used Squirrel and Lua years ago, also evident from the tolua++ tool maintained by Ariel Manzur, a co-founder of Godot, when Codenix, a company founded by Juan and Ariel, was still afloat. Lua has been employed in numerous tools and frameworks, making it a valuable language to learn.

For performance-minded and low-level enthusiasts, the Lua ecosystem includes LuaJIT FFI module (Foreign Function Interface), which allows you to bind external functions without unnecessary engine bindings or recompilation. You simply need to add declarations for the functions defined in a library, and voilà! Godot didn't pursue this approach due to the de-facto Priorities, which are not focused on achieving absolute performance. With LuaJIT's FFI, you can even define bindings for native OS functionality, such as `MessageBox()`, by providing the library name and declarations from the Windows API. It's truly magical!

Engines versus Frameworks

Game engines and frameworks differ primarily in two significant ways that directly impact user-developers: the availability of an editor/IDE and the ability to access lower-level features. In the case of Godot, they attempt to offer a complete package that often appeal to students, at the extreme cost of limited access to lower-level features and the ability to extensively customize existing features for specific use cases.

For more details, you can refer to the Overview of Godot Engine and the de-facto principles inferred from Godot's project management. However, there are no publicly declared principles accepted as official by the lead developer of Godot or any other trusted Godot member. As a result, Godot's development vision remains unclear, as it is not documented in Godot's official documentation. This lack of clarity may lead to potential bloat in the future. This is where frameworks come into play: if you care about size, frameworks often allow you to pick only the features that interest you, and instead of fighting against engine

constraints, you have the freedom to realize your vision in any way imaginable.

Using frameworks, of course, usually comes with the drawback of a steeper learning curve. However, if you aspire to become a professional game developer, it is advisable to avoid confining yourself to all-in-one game engines, especially when some of them merely pretend to offer open-minded and innovative technology. Godot is no exception, as it suffers from the Not Invented Here syndrome, often reinventing the wheel instead of leveraging existing solutions. As you become more familiar with Godot, your chances of fitting in as a professional software engineer diminish, as many useful advanced features are hidden from users. Even if you choose to continue using Godot, I encourage you to step out of your comfort zone and explore alternatives, including frameworks, even if you don't plan to use frameworks in your game development endeavors on a regular basis. Godot can be likened to a gold rush, except there is no gold to be found. Just look at the name, at least: Godot!

Having said that, if you happen to be an engine development enthusiast, you already possess enough skills to create your own perfect tool by combining existing open-source components. You may even fork Godot if you wish, like the authors of Goblin Engine did, but I don't recommend forking Godot either, as you will inevitably inherit all the hidden limitations, bugs, and additional maintenance cost. See Blind Squirrel Entertainment's experience developing Sega's Sonic Colors Ultimate using a custom fork or a subset of Godot Engine codebase, which is riddled with bugs, again, see Priorities. Regardless, I kindly advise against becoming the person who says, "*I could work on Godot for the rest of my life!*" 😁

If none of the above options appeal to you, perhaps this may present a need for innovation. However, it is important to always strive for reusing existing solutions and avoid reinventing the wheel unless it serves the purpose of gaining a deeper understanding of the underlying technology. Whatever you decide to do, as a general rule, avoid presenting a toy project as an universal solution for all purposes and all people. 😊

2D vs 3D

As you may know, Godot is often touted to have so-called "dedicated 2D" engine. But game engines are not limited to exclusive 2D or 3D. It's a misconception. Even if an engine is seen as 2D, they still all operate in 3D space under the hood, because that's how most modern computer graphics work. While game engines may provide some 2D tooling, it doesn't make them inherently or exclusively 2D. Engines may simply offer higher-level abstractions for working with 2D objects, such as canvases, layers, tiling, etc. Using 3D engines is as valid as using so-called "2D" engines for developing 2D games. For 2D, everything is orthographically projected onto the screen anyways. Therefore, you should not restrict your engine of choice merely because it has a "2D" label.

If you want to create a 2.5D game with pseudo-depth and parallax effects, implementing it using 3D techniques can be relatively easy with enough experience. Moving 2D layers along the Z-axis can achieve the desired effect. In contrast, in the case of Godot's supposed focus on 2D and its separation of 2D and 3D functionality, you may encounter some peculiar design compromises. For example, the lead developer of Godot introduced an experimental feature that allows for "pseudo 3D support in 2D engine," as he describes it. However, the complexity arises from the need to set up and update duplicate objects and properties between layers, which is actually a major drawback that undermines the alleged usefulness of such a feature. These limitations could have been avoided by solely relying on 3D techniques from the beginning.

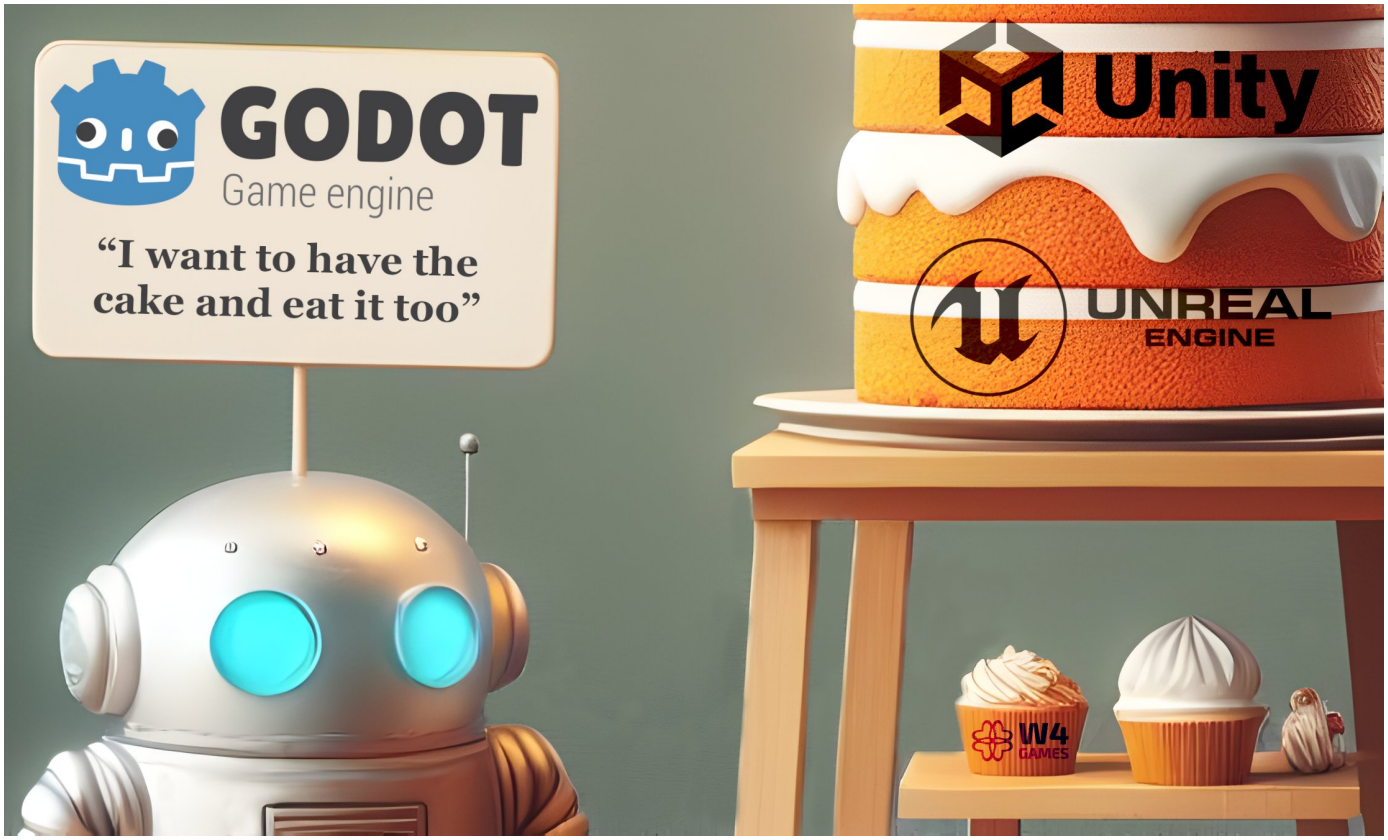
Graphical User Interfaces

If you don't want to limit yourself professionally, it is not advisable to rely on Godot for everything. You may even notice how Godot fans recommend using Godot as a general-purpose GUI "framework" to build pure GUI applications, be it desktop or mobile, which is certainly an overkill.

You can find more suitable frameworks for this, such as Qt. In other cases, if you would like to explore GUI development in the context of game development specifically, you can read this comparative essay by the author of BGFX, which serves as a decent waypoint.

This is more about using the right tools for the job. It is unreliable to utilize game-specific tools for developing desktop applications that don't require any kind of 3D acceleration, just as using minimalist web frameworks would not be suitable for creating software like AutoCAD, for instance. Failing to use appropriate tools is an unsustainable approach. So, it is not solely about size, but performance and maintenance considerations. Moreover, Qt will always have more GUI features than Godot ever will, given the same size.

Open Source vs Commercial



The following clarifies or refutes claims by Juan Linietsky made in the presentation titled “*Advantages of Open Source in GameDev: A comparison of Godot vs commercial alternatives.*” If you’re in search of a substitute for Godot, you might want to explore some potential alternatives to Godot Engine.

It’s worth noting that when Juan Linietsky discusses commercial game engines, he primarily refers to Unity without explicitly mentioning it. Also, Unity’s splash screen is now optional.

> Plus you yourself used this same feature as an "advantage of open source engine over commercial ones" why make it a big deal just because it is on a meme format?https://t.co/wwMPgknzyk pic.twitter.com/cdigo5DC3N
>
> — .pig//Dev (@pigdev) January 5, 2020

Moreover, it’s also vital to point out that the debate over Godot versus engines like Unity and Unreal creates a *false dilemma*. Unity and Unreal have established themselves as industry standards for a reason—they provide the necessary tools to tackle the complexities of professional game development, unlike Godot. Even Juan Linietsky himself says that you should use Open 3D Engine if you want something made by industry engineers.

> GDC space was rented/donated by W4 in order to increase industry presence (which is the aim of the company).
> Its a small part on what is spent contributing to Godot. Godot is independent and must procure its own funding.
> Use O3DE if you want something made by industry engineers.
>
> — Juan Linietsky (@reduzio) February 16, 2023

For more information on W4 Games, a commercial company co-founded by Juan Linietsky and others, please refer to the Community-Driven chapter.

The tables below present Godot's statements on open-source and commercial software in the leftmost columns (the first or second column), while the actual reality is described in the rightmost columns (the second or third column).

Open Source	Reality
Traditional software industry prefers Open Source tools.	The traditional software industry prefers *commercial* tools. The modern software industry may find value in leveraging Open Source without being heavily dependent on the underlying open-source technology.
Game industry is trailing behind, due to the traditional high secrecy on technology, hardware and hardware constraints.	The gaming industry has a somewhat hedonistic essence and relies heavily on innovation, often requiring more than just Open Source tools. This innovation is driven by the competitive landscape among commercial companies.
The trend is slowly shifting.	Although we'll observe a growing use of Open Source in the gaming industry, commercial solutions will remain pivotal and central. Open Source shouldn't be seen as a complete replacement for all commercial solutions. However, if such a shift occurs, we may find ourselves in a future where innovation diminishes significantly.

Godot	Commercial	Reality
Codebase is well organized and easy to understand.	Codebase is closed, engine is a black box.	The quality and readability of code are entirely contingent on the competence of project leaders and the type of contributions the project receives. Read also Do Repeat Yourself section.

Godot	Commercial	Reality
Code access is a bliss for experienced developers.	Sometimes code is available (with restrictions), but development is still not public.	Having access to the code is just *one* of the benefits among various factors; it's not a panacea. Read also Customization and Not Invented Here sections.
The more complex the project, the more you benefit from source access. Several developers who understand the engine internals will be glad to help.	The more complex the project, the less resources are found. Trying to work around problems results in a reduction of game quality or an increase in development time.	While having access to source code is undoubtedly advantageous for developers using such software, the truth is that commercial solutions are generally robust, flexible, and feature-complete to the extent that source code access is often unnecessary. Moreover, dealing with technical debt from Godot falls on the user's shoulders, and not resolving this technical debt will surely result in reduction of game quality or an increase in development time. Read also Freedom section.
If paid support is required, Godot has dozens of developers with deep knowledge of the code base.	Only the company making the engine has deep knowledge of the codebase. Support is very costly.	If a profound understanding of Godot's code base is portrayed as a prerequisite for commercial projects using Godot, it raises questions about whether Godot possesses sufficient built-in capabilities to address any slightly less trivial use cases out of the box.
Contributions of new features will most likely be added to the engine. Most commonly used features are provided out of the box, and integrated to the core engine. Less often used	New features or improvements will need to be maintained separately. Core engine features missing because of the business model. Need to be purchased from third parties. Not	Many new features in Godot get rejected in Godot. Features end up being abandoned and handed over to third parties because of insufficient maintenance. The additional features contributed by the Godot community through plugins are perceived as a potential threat to Godot's commercial agenda, including ventures like W4 Games or the

Godot	Commercial	Reality
features are provided via asset library, for free.	always updated to the latest version and no official support.	upcoming Godot's commercial Asset Store. Read also Extensibility section.
Godot is developed under the very permissive MIT license. Use it as if it was your own in-house engine. Of course, no fees or revenue share.	Free binary blob, pay to remove splash or change editor theme. Pay a high price for source code that few developers understand. Pay for a sizable % of your income.	The fervent fanaticism within the Godot community often leads developers to feel a sense of obligation to promote Godot for free, financially donate to the project even if they don't use the engine based on a perceived belief in Godot's so-called bright future, and so on. Read also Freedom section.
Godot has more than 550 contributors and dozens of core developers. Once popular, open source projects live forever, and only keep getting better over time.	Even widely popular technologies are discontinued or development stalled if they are not profitable or their market stops growing.	The presence of contributors doesn't guarantee immunity from abandonment. Both commercial and open-source projects are susceptible to being discontinued, and it hinges on various factors beyond just the count of employees or volunteers. This ignores potential challenges such as technical debt, changes in project direction, conflicts among contributors, or shifting priorities that might hinder continuous improvement. Popularity is an unreliable predictor of future success. It doesn't consider changes in technology trends, evolving user needs, or emerging competitive alternatives that might impact the project's popularity, even in the Open Source arena.
Godot users and developers all benefit equally of each other's contributions. We want to focus on	The profit of a company making technology is more important than your game. No consensus exists, engine	The claim of equal benefits is ambiguous. Both open-source and commercial solutions have the potential to offer an equitable distribution of value. However, with Godot, there is an increased

Godot	Commercial	Reality
making great games, so let's solve our technology needs together.	direction is dictated behind closed doors.	risk of developers finding themselves in a perpetual state of incompleteness, diverting their attention and wasting resources for *engine* development rather than focusing on *game* development specifically.

Downsides of Open Source Software

Considering the Open Source ethos, the list of alternatives is comprised of open-source or source-available game engines and tools. However, I encourage you not to limit yourself to just open-source options; please refer to the Freedom and Cults sections for further rationale.

As a proponent of open-source software myself, it is important to acknowledge its limitations. Many self-proclaimed "advocates" for Open Source often overlook the downsides of using and adopting such software. While you can explore this topic on your own, allow me to offer my perspective in a down-to-earth manner, especially regarding indie game developers. I intentionally focus on discussing the downsides here, as the benefits of Open Source are already well-known to many, so there is no need to reiterate them here.

Open Source software is often developed by volunteers, which can create an environment with an underdeveloped sense of responsibility. This **lack of responsibility** can result in negligence when it comes to providing comprehensive features or polished solutions. In the long term, this may place a burden on independent developers who need to invest additional time and effort to customize, enhance, or refine the software to meet their specific needs or match the quality of commercial products.

While it is true that bug-free software is a myth, and the level of quality is not always directly correlated with whether software is open-source or commercial, commercial software tends to foster a stronger sense of responsibility. This is because it involves an explicit relationship based on monetary transactions, which in turn creates a stronger incentive to produce high-quality software. Consequently, opting for software that is software-available rather than exclusively relying on pure open-source projects may provide a better compromise to leverage the best of both worlds.

When it comes to Godot Engine and its developers, including those who are paid, there is a distinct absence of any sense of obligation or responsibility when discussing donations. The core developers reside in countries with a lower cost of living, enabling them to sustain

themselves for a considerable period solely through Patreon without immediate pressure to deliver significant improvements. Therefore, if a feature is not already present and functioning, it is unreliable to expect it to be available in the near future under these conditions.

Remarkably, there are numerous leaders who voluntarily decide to refund donations in the event of project failure (as often the case with many start-up projects), even though donations are typically non-refundable. They understand that any form of monetary support creates some level of obligation, albeit weaker in the case of donations compared to direct financial compensation. But this truly commendable behavior does not describe Godot leadership. They *might* mention responsibility, but only to manipulate contributors into behaving according to the desires of Godot leaders. The fundamental concept of responsibility is completely alien to them in the first place.

Should you contribute to Open Source projects?

Don't contribute to open-source projects unless *necessary*.

While it may seem enticing for students to contribute to an open-source project to enhance their experience and build portfolio, based on my experience, unless you're contributing to a well-known open-source project, say, in the web development industry, the game is not worth the candle. There are several reasons for this, including an overly zealous liberal approach in the Free/Libre Open Source Software (FLOSS) philosophy and the cultivation of the ideology that working on things for free is a privilege, as remotely evident from jokes such as this one. Therefore, ensure that you are being paid by the company and that the company contributes something meaningful to the open-source project that you care about, at the very least.

Remember, you are **not obligated** to contribute if you use FLOSS. You don't owe anything to anyone when it comes to software with permissive licenses, other than what a license requires you to do. Any attempts by project leaders to guilt-trip you into "contributing back" (which is a peculiar phrase that implies an obligation) should be seen as a red flag, and it's best to avoid such projects.

If you *feel* like you want to contribute, take a moment and contemplate your deeply-rooted motivations: do you want to contribute to improve software, or just gain a superficial level of recognition by your peers? Answering these kinds of questions may help you to build and follow your own path. Instead of supporting the endeavors of toxic leaders, direct your time and effort towards pursuing healthy endeavors that align with your life purpose.

Godot is predominantly developed by and for hobbyists. Godot might tackle some complexities, but it naturally gravitates toward a peculiar audience. Unlike healthy open-

source projects, Godot has no vision, philosophy, principles, or roadmap. Therefore, Godot is a happenstance.

I caution against contributing to Godot or even using Godot, regardless of whether you consider yourself an amateur or a professional. The risk lies in the insular nature of Godot's community, which limits exposure to industry best practices. If you get stuck in Godot, it can be tough to break free from its deceptively simple structure.

See Also

Why not fork Godot Engine?

There's no point in forking Godot if there exist more robust tools out there. Godot is so broken, unstable, and unreliable that it's in a league of its own. At any rate, Godot is made for hobbyists, by hobbyists. For the most part, Godot attracts amateurs who are unable to fork it, and frankly, they don't want to fork it because they feel satisfied with what Godot offers them. Most people don't care, and professionals opt for more established and capable game engines instead.

Maintaining your own fork is a massive undertaking, so many just moved on to other game engines and tools. There are a few notable contributors who either initiated or expressed the intention to create a fork of Godot. However, these forks ended up being either underdeveloped or abandoned.

In the Godot community, the suggestion to "fix it yourself" or "just fork it" is common but not helpful. The complexities involved in maintaining a fork still require a level of expertise and

commitment that is often underestimated by those who have little to no experience maintaining open-source game engines of this scale. Think of it like this: everyone has the right to become the President, but only a few have enough capabilities and opportunity to exercise this right. Similarly, in theory, you can fork Godot, but it means assembling a team of dedicated engineers willing to commit their lifetime to maintaining the fork for free, purely out of passion.

If a relatively successful fork of Godot were to happen, it would likely be designed for specific games or genres. Based on numerous veterans who've been waiting for Godot long enough to realize the reality behind the scenes, Godot is primarily and predominantly a game engine for game jams at this point. Anything long-term requires monstrous workarounds, to the point of maintaining a fork of Godot. Unfortunately, forking Godot becomes almost imperative if you want to craft intricate games featuring unique and engaging gameplay on a larger scope than typical platformers. It's not the standard we all should strive for. We need a tool that works in the present, not a vague hope for a bright future.

There are several other reasons why you shouldn't fork Godot. Factors that are closely related to the fanatical nature of the Godot community. The phenomenon of why some people choose Godot is sociopsychological in nature. Many simply believed Godot's big lies proliferated by the con man of the gaming industry, Juan Linietsky, the lead developer of Godot Engine. While you might perceive Godot as a groundbreaking innovation, the reality is quite different. The loud and fanatical Godot community might create the impression that it's extremely popular among professional developers, but this is not the case.

The reason you might find it difficult to abandon the idea of forking Godot is that there's a possibility you have developed a utopian ideology regarding a perfect game engine. Once you have invested time, work, and other resources into Godot, you may start to feel personally attached to it. You might desire the lightweight experience of Godot along with the extensive capabilities of engines like Unity and Unreal. This is why Godot attracts many amateurs. While I don't want to shatter dreams, the reality is that some aspirations may be unattainable, even if they are admirable. What makes this worse is that Juan Linietsky capitalizes on these dreams of people who naively think that they can have their cake and eat it too, so to speak.

Juan Linietsky has cunningly set up a situation that benefits his commercial agenda, regardless of whether someone decides to create a fork of Godot. If you're not a fan of Godot, Juan suggests you fork it. Unfortunately, some people fall for it and create different versions that struggle to keep up with the upstream project. This allows the main project, Godot, to pick and choose features from these versions, taking advantage of people's creative efforts. Meanwhile, Juan and his loyalists continue to parasitize on people's dreams of the ideal game engine they've been waiting for. Moreover, if a fork poses a threat to the main project, rest assured that the toxic leadership of Godot will launch a smear campaign

against it.

For those still contemplating the forking route despite everything mentioned thus far, it is crucial to familiarize oneself with the release process, disassociate from Godot entirely, and be prepared for a long-term commitment. However, the question lingers: Is investing years into building a game engine instead of focusing on your game truly the path you desire?

That's why I advise against jumping on the Godot forking bandwagon. The world is large, and there are plenty of projects, both commercial and open-source. Life will go on even if Godot fades into oblivion. In fact, completely abandoning Godot is the only way to make leadership realize that they are doing something wrong at this stage, because, unfortunately, they are unable to learn from criticism. For Godot to truly thrive, a fundamental shift in its mentality is necessary. But if that were to happen, the irony is that it wouldn't be Godot anymore!

A Word of Caution on the Godot Engine Forking Bandwagon

With a cultic group like Godot Engine, it can lead to the formation of anti-cult movements that end up mirroring similar cult dynamics. These anti-cult movements can even be taken over by the original group. Just look at historical precedents, like The Cult Awareness Network.

That's why I don't express strong anti-Godot feelings, even though my stance against its undue influence is clear. I can speak up against Godot, but loving or hating it? That would be too much. Unfortunately, forks of Godot could unintentionally adopt a cult-like mentality. Avoid this mental trap.

Glossary

- **Values** - things that we have figured to be useful on both individual and social levels. Values answer the question of what is right and wrong. They help us to make good decisions that are beneficial both to us and others, therefore consistently following our values allows to build social and moral norms. Values may differ from one group to another.
- **Assumptions** - things that we *believe* to be undoubtedly true, usually without a proof. When we say something to support our claim, we may imply that something is true even if it's not, often without thinking about it. The focus may also determine whether something is seen as an assumption or a *belief* if we look at this in the context of our hierarchical *belief* system.
- **Beliefs** - things that are accepted and adhered to, based on our faith or something that stems from our direct experience. Beliefs can be reinforced and questioned. We may form our beliefs upon assumptions. If we start to question our beliefs, they may become *assumptions* that may be considered under other conditions, unless beliefs get destroyed completely.
- **Expectations** - projections of our *beliefs* and *assumptions* onto the future. Expectations can be realistic or unrealistic. They help us to predict and forecast future events using existing facts that form a pattern of behavior or activity.
- **Toxic leadership** - a group of people that represent a project and jeopardize project's success. It ignores, neglects, refuses, or blocks attempts at defining or clarifying project's values, goals, vision, governance etc. to those participating or show willingness to participate in a project.
- **Toxic cult** - a cult which causes harm to its members (physical, emotional, mental), discourages critical thinking, frowns upon fundamental human rights such as free speech, and where *toxic leadership* thrives. In toxic cults, members are are treated as "means" rather than "ends", regardless of leadership's expressed intent.
- **IT cult** - a new type of a cult in the modern era of Internet and Information Technology. Not all such cults are considered or recognized as *toxic cults*.
- **Godot follower** - people who use, develop, or follow Godot Engine's development and related activities, even if such people just consider using Godot in the future. It's not a requirement to be a user or developer of Godot at all to be considered a Godot follower.
- **Godot cultist** - a *Godot follower* who shows signs of bigotry or signs that comprise a

toxic cult. Not all *Godot followers* are cultists, but all cultists are *Godot followers*.

Types of Waiters for Godot

People waiting for Godot come in various flavors. Not without exceptions, of course!

- **Followers**: don't use Godot but follow the news.
- **Users**: actively use Godot.
- **Contributors**: develop Godot as an engine.
- **Donators**: support Godot financially.
- **Advocates**: promote Godot.

The traits below specify the characteristics of each class based on their level of belief in Godot.

Believers

Mostly hobbyists who might not even have prior game development experience.

- Eagerly anticipating new features and bug fixes.
- Cannot differentiate a limitation from a bug, often blaming themselves more than the engine.
- Have faith in leadership and believe in a bright future.
- Sacrifice working on their own games for the greater community, losing sight of why they contribute to Godot.
- Caught up in the *fear of missing out* cycle.
- Show intolerance towards *doubters* and *non-believers*.

Doubters

Mostly experienced game developers.

- Question Godot's claimed features, capabilities, and promises.
- Hold reservations about Godot, primarily connecting with specific engine ideas through tinkering with it.
- Dislike leadership but continue to use Godot despite facing limitations and obstacles.
- Disapprove of both the leadership and Godot, waiting for someone else to fork the project.

Non-believers

Mostly professional game developers.

- Heard about Godot but never tried the engine and have no intention to use it.
- Former Godot participants who have become thoroughly disillusioned.
- Critical of leadership and Godot, believing that forking the engine would only perpetuate the vicious cycle of incompleteness.
- Dismiss the idea of forking Godot because it doesn't even cross their minds, realizing there exist more robust and stable tools out there.

Life cycle of Waiters for Godot

1. **Hobbyists**: `Non-believer` → `Believer` → `Doubter` → `Non-believer`
2. **Experienced**: `Doubter` → `Non-believer`
3. **Professionals**: `Non-believer`

💡 Rumor has it that some will remain in perpetual wait for Godot...

Frequently Asked Questions

Oops, cultists of Godot do not ask questions by definition. 🤪

Sorry for this fantastic misunderstanding! 🙃

Investigations

This is a list of investigations conducted by the author of this book. The goal is to uncover insights into Godot's potential future based on its current actions. Primarily, these investigative articles focus on Godot's public relations. It's advisable to read the Community-Driven chapter before diving into these topics:

- Blue Robot and Red Hat: Which playbook is Godot Engine trying to take a page from?
- Waiting and Unionizing for Blue Robot: Are Godot advocates infiltrating Unity?

Blue Robot and Red Hat: Which playbook is Godot Engine trying to take a page from?

Publication Date: *November 3, 2023*

Updated: *May 6, 2024*

Godot is ostensibly a free and open-source game engine. However, several key factors suggest that Godot operates disingenuously. One such key factor is Godot's for-profit W4 Games company, co-founded by Juan Linietsky and Rémi Verschelde, the lead developer and project manager of allegedly non-profit Godot respectively:

One of W4 Games' investors is Bob Young, the co-founder and former CEO of Red Hat, an enterprise-focused open-source company that IBM went on to acquire. According to the article titled "How W4 plans to monetize the Godot game engine using Red Hat's open source playbook", Juan Linietsky claims to follow Red Hat's success story:

> *"Companies like Red Hat have proven that with the right commercial offerings on top, the appeal of using open source in enterprise environments is enormous," Linietsky said. "W4 intends to do this very same thing for the game industry."*
>
> *[...]*
>
> *"Bob is an incredible human being who helped create a whole new type of business where nobody expected it was possible," Linietsky continued. "He identified the opportunity for*

> *Godot and W4 as very similar to Linux and Red Hat two decades ago, and has been very kind to share his wisdom with us, as well as becoming an investor in our company."*

The cooperation between Godot and Red Hat doesn't stop here. The official Godot channel previously published a video titled "Godot for the Enterprise - Luke Dary - GodotCon2021":

Luke Dary is an employee at Red Hat. He said, "Godot played a part in both the 2019 and 2020 Red Hat Summit technical demos." Red Hat officially promoted Godot at GDC 2021:

As you may already know, Unity's pricing changes caused a stir in the game development industry. Some overzealous Godot users contributed to further destabilizing Unity's situation. They opportunistically infiltrated the unofficial r/Unity3D subreddit to promote

Godot, which de facto turned out to be a front group for attracting new Godot users, read Waiting and Unionizing for Blue Robot: Are Godot advocates infiltrating Unity?

Progressively, Unity adjusted its pricing model to cater to its users. This included suggesting an alternative rev-share model and removing the requirement to display Unity's splash screen. John Riccitiello, the CEO of Unity, later resigned and was replaced by James Whitehurst as the interim CEO and President of Unity. **James Whitehurst is the former CEO of Red Hat**.

It's hard to say how the appointment of Unity's new interim CEO will impact the development trajectories of both Unity and Godot. However, there are already discussions about how Whitehurst might steer Unity through turbulent waters, and whether Godot can weather the tsunami caused by Unity's decisions. Some even say that Whitehurst might stick to his usual playbook and acqui-hire the Godot project leads, so Unity wouldn't have to fret about Godot potentially disrupting the industry.

It is not far-fetched to imagine a scenario where Godot, the open-source game engine, is influenced by the Red Hat CEO's connections when such connections already exist. This possibility gains credibility due to Juan Linietsky's recent statement that he doesn't make any technical decisions in Godot anymore and that calling himself the technical director would be an overstatement. With this void in the technical leadership, it becomes necessary to fill the gap.

According to some discussions that previously happened within the Godot community, people have variously described the interplay between Godot, W4 Games, and Red Hat to be "cautiously optimistic":

Comment
byu/Xatolos from discussion
ingodot

While others expressed that Juan Linietsky controlling both W4 Games and Godot are warning signs that must not be avoided:

Comment
byu/Xatolos from discussion
ingodot

In fact, such concerns are reminiscent and/or analogous to Software Freedom Conservancy's thoughts on IBM's acquisition of Red Hat:

I've heard people imagining the best from this deal, and also people imagining the worst.

> *The one thing everyone can agree on is that there's a lot of uncertainty, despite whatever reassurances are contained in corporate messaging. Because of this, I think it's a good time to remind everyone of the ways we can protect ourselves now and in the future from these kinds of uncertainties related to changes in ownership, structure or motivations of corporate players in free and open source software: [...]*

Dissatisfaction with Godot's failure to live up to its stated principles of being community-driven finds parallels in other open-source projects, such as Fedora and its commercial distribution Red Hat:

> Please remove the "community driven" part from Fedora's official descriptions if no community is supposed to be involved in decision making
> byu/LunaSPR inFedora

It should be noted that Godot is in the process of leaving Software Freedom Conservancy for the new Godot Foundation, in order to establish the ultimate control over the funding decisions. They labeled this event as "graduation." Godot also announced the work-in-progress Asset Store, likely in an attempt to replicate Unity's business model in this regard

---

> As said on many replies, an official Godot Asset Store is in the works. Since we got the Foundation going, we've been trying to get it up and running, but there are many things to figure out before we can make it available to everyone. pic.twitter.com/llPN6Miv2N
>
> — Emi ➡️ A MAZE (@emi_cpl) September 18, 2023

These decisions are inconsistent with Juan's claim about Godot's purpose and motivations:

> *In all seriousness, the current Godot logo was created to convey a friendly, welcoming and*

> *informal feeling, which is what the engine represents at heart..*
>
> *Our goal is to make the best engine you will ever use (having lots of fun along the way) and make sure it's as accessible as possible for everyone.* ***We are not here to sell a professional product or make money from you*** *[emphasis added]. So, I hope you can accept and enjoy Godot for what it is.*

Further reading on the topic of Red Hat:

- The Suicide Attempt by Red Hat [Opinion] - It's FOSS News.
- I'm done with Red Hat (Enterprise Linux) - Jeff Geerling.
- A Comprehensive Analysis of the GPL Issues With the Red Hat Enterprise Linux (RHEL) Business Model - Software Freedom Conservancy.
- Writeup about Godot's Code of Conduct - mentions Red Hat systemd's project.

Waiting and Unionizing for Blue Robot: Are Godot advocates infiltrating Unity?

Publication Date: *November 3, 2023*

Updated: *May 21, 2024*

After Unity changed its pricing model, Godot users started a campaign to get Unity users to switch to Godot. Such recommendations are not helpful as even comparing Godot versus Unity creates a false dilemma, read Open Source versus Commercial. This problem got so bad that even Unity users on Reddit said it might be a good idea to stop aggressively promoting Godot over Unity, even long after Unity adjusted its new pricing model to cater to demands of its users:

> Prohibit recommendations to switch to Godot
> byu/D3RRIXX inUnity3D

Here are several links to discussions and posts that support these claims.

When reading the threads below, pay attention to Unity community moderators such as hippocoder**,** Eric5h5**,** as well as Godot followers, supporters, and/or believers such as Ryiah**,** neoshaman**,** Murgilod**,** etc.

- Whats up with the moderators these days?
    - AndersMalmgren says, emphasis mine:

> They are locking relevant threads like crazy. As a **paying customer** its a bit annoying

> actually. **Unity should maybe consider to only have employees as moderators instead**.

- Buhlaine locked the thread and happens to be a former community manager at Unity, who recently expressed enthusiasm over Brackeys' plans to learn Godot (more on that in a future post, perhaps).
- Injust moderation practices
- AndersMalmgren**,** the person who started the thread above, replies about hippocoder:

> hippo and the other guy is a bit strange moderators, very true. Like all the nonse threads hippos creates. But he locks my business relevant ones like the gfx job one a while back

---

- Unity Forum moderators are being annoying
    - hippocoder praises Ryiah to become another moderator, who is a Godot supporter.
- I dont' mean to be disruptive, but Eric5h5 is just wrong
- Unity employees are no longer permitted to be moderators of this subreddit
    - Volunteers at the Unity subreddit forbid official representatives to moderate the subreddit.
- I think Eric5h5 doesn't get the respect he deserves
    - Eric5h5 is praised on the Unity subreddit, which expresses anti-Unity sentiments currently as per the thread above.
- Eric5h5's post, commenting on "Road to Vostok" in the Unity forums—a game initially developed with Unity but now being ported to Godot—notes a perceived decrease in quality with the transition. Despite this, Eric5h5 expresses disappointment with Unity's decisions and remains optimistic about Godot's potential.

See also my mega thread demonstrating the severe bias of Unity community moderators (who are merely volunteers, not necessarily official Unity employees):

- Some Unity users and even moderators within Unity community are infected with the futile ideology of waiting for Godot! This must be fixed! Let me explain why.

Interestingly, Juan Linietsky, the lead developer of Godot, previously posted a tweet with a message from hippocoder**,** saying:

> Unity forum moderator does not have a lot of faith in an open source project vs corporation created software..

If you've read the above mega thread, something doesn't add up here, as hippocoder had previously expressed a strong pro-Godot stance on the Unity forums. Additionally, in the same thread by Juan, he replied that:

> We are not really competing with anyone.. Godot is not a business :\

These statements exhibit hypocrisy and deceit as Juan co-founded the for-profit W4 Games company, capitalizing specifically on Godot's popularity to offer commercial Godot services.

Therefore, it's reasonable to assume that hippocoder might have been in contact with Juan earlier, especially since hippo claimed to have allegedly signed the email for Sony to accept Godot. In one of his posts on X, hippocoder, aka SquaredApe, reaches out to Juan suggesting that Godot's upcoming Asset Store should best be done as a private company, with funds funneling into separate console builds for Godot, rather than being an open-source attached affair.

Unity's inherent issues notwithstanding, given the aforementioned evidence, it's challenging to deny the fact that influential Unity community volunteers, who were granted moderator privileges by official Unity staff, are not acting in the best interests of Unity as a company.

Update: *May 1, 2024*

Brackeys is announcing his return, but this time he’s planning to create tutorials for Godot. He explicitly mentions ***unionization***:

> *I think for a while that some of the harsh realities of the video game industry had taken away some of the joy that I used to feel when making games. From the **lack of unionization** [emphasis added] in AAA to the real challenge of keeping an indie game studio afloat, I think there’s an ever-increasing pressure on developers.*

See also the following satirical video on this matter:

Reviews and Testimonies

The following is a compilation of notable reviews of Godot Engine and testimonials from individuals who have discovered, used, contributed to, or maintained Godot, including observations from people outside of the Godot community. If you are interested in the testimony of the author of this book specifically, please refer to the No Longer a Blue Robot section. However, you can also take a look at some of my notable posts where I share my experiences and interactions with Godot's CEO, Juan Linietsky:

> Mega thread of chats with Juan Linietsky aka @reduzio. I'm a former @GodotEngine maintainer.
>
> Quotes taken from saved screenshots before I was banned by this con man and his loyalists who cover up his lies. Juan is a fraudster who threatens contributors. 🧵👇 #TruthAboutGodot pic.twitter.com/33N8WSfMoH
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) September 13, 2024

> Leadership of @GodotEngine will never support community effort of extending #Godot's functionality via extensions and modules.
>
> Read private conversation with @reduzio, the lead developer of #GodotEngine, talking about #Goost. They want to control everything.#TruthAboutGodot pic.twitter.com/6Qj5X6yaAV
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) October 1, 2022

Please note that this list is not exhaustive. If you wish to contribute something to this list or provide your own review or testimonial, you can contact the author of this book for suggestions.

The quotations are cherry-picked to showcase various aspects of Godot in relation to topics covered in this book. They are not meant to be presented out of context. It is advisable to read the entire content to gain a comprehensive understanding. Emphasis may be added to underscore notable or typical sentiments that are balanced out across quotes.

In case the links below become broken, unavailable, or taken down, you can access them through archives like the Wayback Machine. Full text down below is provided for convenience and for sources that are prone to be lost (such as YouTube comments).

Notable Reviews and Testimonies

Former and Current Members

- Andres “cybereality” Hernandez, former admin of Godot Forums:

> *I’ve been promoting Godot for 3 years, dedicating my time to helping people for free, and recently paying to host these forums (which costs hundreds per month). I really did believe in Godot, but as time has gone on, I’ve just seen **broken promises**, **lies**, and **suspect behavior** that makes me think at some point the Godot Engine turned into a scam. [...] I think the first time I noticed this was in 2020 when I started trying to use baked light-mapping. I noticed that it was essentially completely broken.*

In a video, a disappointed Godot believer dissects the situation above and concludes:

> *Godot is mismanaged very badly and I feel like the community doesn’t really care about anything. Everybody’s eating up every update and announcement as if it’s the best thing ever happened. There was a lot of hype surrounding 4.0, but when we finally got 4.0, it was a complete disappointment. It was released as a stable version, but nothing about that engine was stable.*

Aftermath: Godot Forums were acquired by Mike Lundahl and his company. Mike partnered with The Mirror, a Roblox-like game development platform made with Godot Engine.

- abrasivetroop, long-term user of Godot Engine:

> *I have 4 years of experience with Godot and I published 2 commercial games with it. I have believed in Godot for years and it has been my passion. Since I started to realize the problems within Godot I never backed down on pointing them out. I care about the future of Godot. These problems stem from the incompetent leadership of Juan and Remi. My only intention is to make people aware of these problems so that Godot can have a better future.*

- Lilly Byte Games, former moderator of Godot’s Discord server:

> *I am tired of getting shit on by "Godot bros" whose only experience with Godot is what they see/hear in their limited experience of being fresh and new to the Godot train, and not yet had the curtain pulled away from their eyes to Godot's reality. [...] If you are not a part of the "**Godot Cult Insiders**" your time is wasted. We weren't the only people to see this either... [...] The "belief" I had in Godot being community driven was an **absolute lie**.*

- Camille "pouleyKetchoupp" Mohr-Daurat, former contributor and maintainer of Godot, physics programmer at Rockstar Games:

> *I've been **bullied** two times by different people on this project and every time project management reacted by siding with the bully, so I've left and I'm **never going to contribute again**.* [...]

- Winter Pixel Games, contributor of Godot, creator of Rocket Bot Royale:

> *And there are just some **very unergonomic** things about the Godot source right now. Stuff that would just never fly with experienced teams or devs. And from what I've seen it seems to **put off experienced devs** and more established teams. [...] I think it's just due to the lack of actually using the engine by the core team to ship real products. This is really why dogfooding is SO important, and why Epic/Unreal have such an unbelievably great advantage with their engine development practices.*

- Endri "Lauson1ex" Lauson, contributor of Godot, software engineering consulter:

> *I implemented the new lighting/tone mapper used in Godot 4 and I also back-ported it to Godot 3 which, we all agree, is a fairly significant contribution. [...] And yet, they did not even have the decency of adding my name to the list of contributors. [...] I have not contributed since and I won't be contributing to the upstream project anymore. [...] Now, imagine that the couple of people running said project are receiving MILLIONS, while I donate my time to receive nothing in return. This is why **many talents are leaving**.*

- David "Demindiro" Hoppenbrouwers, contributor of Godot, creator of Godot Rust bindings for the Rapier3D physics:

> *Even as a contributor I've found it **very frustrating to contribute** to Godot. I've found lots and **lots of bugs** while working on my own projects using Godot. I spent a lot of time digging in the engine code to figure out the cause, make an issue and a patch if I could. However, even small changes take a lot of time and effort. While I used Godot 3 for my own projects most of the PRs had to be based against 4. At the time Godot 4 was in a very sorry state and I ran in many, many issues that made it hard to test if the same fix for Godot 3 also works in Godot 4. I wrote some libraries to work around issues that I (or others) could not get fixed or reverted (e.g. I replaced the physics engine with Rapier3D because I really needed more stable and working joints) but **I eventually threw in the towel and decided to focus on other hobbies**.*

- Mariusz 'koder' Chwalba, creator of ΔV: Rings of Saturn:

> *I took a **leap of faith** by developing my game using Godot 3 - when I started it was in early beta stage. Now I feel like your focus is on Godot 4, but the 3.1.1 you are publishing currently on your site is **not stable as advertised**. I would implore you to revise the release cycle to favor stability of published project over development of next feature or major version. Having an "unstable stable" engine is not doing anyone any favors.*

- Favkis Nexerade, long-term Godot user:

> *I spent 3 years working on my games and this engine still has broken graphics, broken animations, broken file system, broken physics, broken importer, broken resource instancer and dozens of other broken "features" [...] **I feel cheated**, I spent so much of my creative energy into this engine and now met with "we wont continue making G3".*

- jarsin, former Godot user:

> *I switched to Godot from Unity. Ya I lasted about 2 months and now I am back to Unity. Really depends on what you are doing but **Godot has tons of bugs** and it just gets old real fast, assuming you actually want to build a game. And trying to fix those bugs is not so easy in a lot of cases. Obviously depends on what you are doing though.*
>
> *Switching from Unity to Unreal or vice versa is a wash imo as both those engines have proven themselves. Godot hasn't and still has a much longer way to go than the fanboys will have you believe. Maybe someday for some types of projects but I just*

don't see it ever having all the conveniences of Unity or Unreal.

- jayrulez, former Godot user:

---

*Some people may say that Godot will be better for 3D with the 4.0 release. **Most of them are just fanboys** or they don't know what they are talking about (or both). So far, the Vulkan render in Godot has been poorly implemented. It will offer a marginal improvement at best.*

Independent Developers

- Tynan Sylvester, creator of RimWorld:

---

*This **lack of decided focus** manifests as IMO mis-spent effort on things that almost no serious indie should be using (advanced rendering features which look pretty in demo videos, visual scripting), while deprioritizing things that absolutely every indie should be using (C#). [...] GDScript looks nice for really really tiny games, but it's obviously never going to scale close to anything like RimWorld; **its performance is way too low** and the code **isn't nearly safe enough**. Most indies beyond the "my first game" level should not be using this.*

- Alexander "nicholatian" Nicholi, computer scientist:

---

*Godot's entire existence is measured in Unity copium. [...] On paper, Godot is a game engine. The reality is much more **Soviet**: it's an online clout farming operation for a couple of random Argentinians. [...] As long as everyone believes in them, nothing else matters. They can have a **broken game engine** as long as people want to pay them for it.*

- Pawel Jarosz, creator of Witchcrafter:

---

*I'm reading all those tweets and the worst thing for me is that Godot **spreads hate** to other engines, instead of love. Community and open source foundations are totally OK with some of the behaviors and movements, but when other makes mistakes. Oh boy,*

> *Defold licensing thing was nothing when you compare it to how the Godot community was triggered after Unity stuff this year...*

- Ross Grams, illustrator, concept artist, and fine artist:

> *Godot has some nice core ideas (the scene/node system), plenty of features, and is fairly easy to learn, but it's also **bug-ridden**, slow, has very incomplete documentation, and a **cultish** community. [...] There's a strong focus on adding new features, very little on fixing things. His [Juan's] most common response to serious bug reports was "oh yeah, there's no point fixing this, I'll be rewriting it for 3.1". So there are lots of **broken or unfinished features** lying around. There's really **no such thing as a stable release of Godot**.*

- NegInfinity, veteran Unity user:

> *The reason why people heard of Godot is actually because of the **"Godot prophets"** that kept trying to spread the word on other forums, due to feeling strongly about it. Such behavior is a major red flag for me, and those people are the reason why I'm not planning to touch this particular engine for at least the decade. Personally, due to that behavior I believe that investing time and paying attention to another engine that is not Godot would be a good idea. Hence the mention of Stride3D.*

- Margaret Ó Dorchaidhe - Software engineer with AAA games experience, triple major graduate from RIT in Game Design, Computational Mathematics, and Computer Science:

> ***Unclear and occasionally unprofessional messaging.** This one is a very fraught one to cover, and I'd rather not include specific screenshots of or links to tweets in this post. I truly do not want to point fingers at individuals, and I don't want this to turn into a conflict between myself and Godot's team. I wish them nothing but the best. But I recommend that anyone who has not yet already look through some of the social media posts and replies of the most prominent Godot management and contributors - it is hard to form a coherent picture of what Godot is, what it's trying to do, and what the best way to use it is. [...] And past that, the trend of criticising other engines, and other companies' strategies, and entire software patterns is... noticeable.*

- Logan "WickedInsignia" Preshaw - Art Director and VisDev Artist experimenting with

realtime environments.

> ***Many crucial bugs on core systems have been ignored for years.*** *I myself have been intensively testing the graphics side of Godot and identified core issues with many of the lighting features, with many proposals and reports initially discussed or met with approval and then left to sit for a year or more untouched. [...]* ***Godot feels like it's in a constant pre-alpha, with no "v1.0" achieved to justify the momentum towards v2.0.*** *[...] The experienced game devs who could test features more intensively simply do not interact with the engine because they come along, realize some core feature is broken in a way that completely obstructs their project, they see the PRs for this issue untouched for years, and then they move on to another engine. [...]* ***Some of Godot's most talented contributors have left the project inorganically.*** *I don't believe Godot's perceived* ***directionless*** *development plays any small part in this, and in talking to existing contributors it's evident that some feel they're simply* ***left waiting*** *for crumbs of info or core changes to be made in other areas they simply cannot predict.*

Aftermath: Discussion was blocked by Godot core developers, and multiple replies were deleted. Read the reactions in this post:

> if you're interested in how Godot's responding to feedback these days and have time to kill during a medium-to-long poop, this is informative toilet readinghttps://t.co/0Jcob6ul9s
>
> — Joe Wintergreen (@joewintergreen) November 29, 2023

Posts

- Xavier Sellier, Godot contributor, creator of City Game Studio:

> Public convo between me and the creator of City Game Studio, built using a custom Godot fork (2019).
>
> Godot is such a mess that he had to fork it just to make it usable in production. Even then, it is still a maintenance nightmare. Still relevant for Godot 4.x.#TruthAboutGodot pic.twitter.com/IrC7AiOOCS

— Andrii Doroshenko 🇺🇦 (@Xrayez) October 10, 2024

- Andrea Catania, Godot maintainer, creator of Godex:

This guy, who forked Godot in 2021 and used to be a @GodotEngine maintainer, created a fork called Godex. Now, he switched to Unreal, but he's still emotionally attached to the Blue Robot ideology and continues to claim that Godot is fantastic due to sunk cost fallacy. https://t.co/HI6I2QcngK

— Andrii Doroshenko 🇺🇦 (@Xrayez) October 10, 2024

- Geequlim, former Godot contributor:

In the end we had to give up using Godot and move to Unity. Of course the scripting language is not the only thing holding us back, but it does affect our development efficiency.

— Geequlim (@Geequlim) August 24, 2022

Videos

I Don’t Trust Godot Anymore

Godot's devs are deceiving you

Anonymous

Here are some noteworthy anonymous reviews and testimonials that were privately shared with the author of this book. These have been deemed credible enough to merit inclusion here, either for further investigation or to provide broader context.

- Relationship between OKAM Studio and Juan Linietsky:

> *I know Juan from pre-Godot times. Before Godot, Juan co-founded a game company named OKAM. Turns out, he ham-fisted very early versions of Godot into OKAM projects and convinced the studio to use them. They had a lot of problems with it and resented him a lot because when things got bad with the engine he failed to be held accountable for them. After a lot of trouble, the team switched to Unity to find out that the productivity increased by orders of magnitude, they felt cheated by him when this happened. At that time, one of the OKAM devs told me "Juan is just using the studio as a guinea pig for his engine prototype". They also were mad because they had to crunch and blamed Godot technical failings from it, when that happened, a common saying was that "Juan raised hands in the air, said 'oh, well' and left". From second-hand testimonies, I vaguely understand that he and her co-founder didn't end in the happiest of terms.*

Context: Martina Santoro, co-founder of OKAM Studio, has transitioned into the role of

Unreal Engine Evangelist. Nowadays, her current social media activity is on Unity and promoting Unreal Engine, with no mention of Godot Engine whatsoever.

Full Text

Andres “cybereality” Hernandez

Sadly, I Think Godot Is a Scam. I’m Not Sure I Can Do This.

You know, I’ve been promoting Godot for 3 years, dedicating my time to helping people for free, and recently paying to host these forums (which costs hundreds per month). I really did believe in Godot, but as time has gone on, I’ve just seen broken promises, lies, and suspect behavior that makes me think at some point the Godot Engine turned into a scam. I do think it was real at one point, maybe when I started with it. Seemed like a nice community and FOSS project. But as time went on I started to notice weird things, specifically with my interactions with Juan. I think the first time I noticed this was in 2020 when I started trying to use baked light-mapping. I noticed that it was essentially completely broken. You could bake indirect light, but not the whole scene. I thought it was a bug and messaged Juan. He said: “Oh someone or another did that whole thing in 1 night, and then it must have broken. But we are developing a new system that is better”. Well, strange, if someone made it in 1 night, they could certainly fix it in 1 night. The new system did come, but an entire year later. So for 12 whole months, there was no lightmap support, a Quake 1 era technology. This kind of pattern continued for years, major bugs with Github issues, but they said: “oh, it will be fixed in Godot 4.0” then 4.0 comes out and things are still broken, “it will be fixed in 4.1” well 4.1 did come out and tons of stuff is still broken (HDR images hang/crash the editor, major black screens and crashing on many Android devices, incorrect color space on HTML5, tons of features missing on mobile/OpenGL making it useless, etc.).

But it’s understandable that there are bugs, particularly with a large codebase and many contributers. The main issue I have is the broken promises and lies. For example, once on Twitter I asked Juan about why mixed mode shadows were not supported (this is when the level is baked but characters have real time lighting, a standard feature in Unity and Unreal). And he tells me that shadow mapping is faster. I don’t know if he misunderstood me, but real-time shadows are not going to be faster than baked lighting. There is a reason almost every AAA game in the last 20 years has used some form of pre-computation. Eventually I found someone with the same problem, and they looked in the code and it was actually a 1-line fix (I saw the image, it worked). But the PR was never merged, for whatever reason. I

also had a problem where I loaded a DirectX normal map into Godot (which uses OpenGL coordinate space) and it was messed up. I mentioned I had the wrong normals, and he says there is only one normal format, despite like 20 years of history of differences between DirectX and OpenGL. Again, I thought there might have been a misunderstanding, but these little kinds of things were starting to add up.

Even that I was willing to work through, as the Godot project is not just Juan, there are many people that contribute. Sadly the interaction on Twitter today has made me realize this whole thing is a scam. I had suspicions before, but this confirms it. Juan earlier today made a post essentially saying that the Godot project is out of money, and that they are cash-flow negative. Remember that a few months ago, the company Juan started with some other Godot people raised $8.5 million dollars. So I made a joke asking if he spent all the money. I mean, it was a half joke, but also kind of a serious question. He's on Twitter begging for money, with over $8 million in his pocket, something doesn't add up. Some other people asked what would happen if they didn't get enough donations, and he says that the development would slow down. This doesn't make sense, the whole mission statement of W4 Games was to grow the Godot ecosystem, but now a few short months later, Godot has no money and development will slow down. This is suspicious, to say the least.

So, as you know, I'm a pretty straight forward guy. So I called him out on it. Though his response only confirmed to me that he's not honest nor can he be trusted. Someone else asked the same question, what happened to the W4 $8.5M. He says, bold face, that W4 is unrelated to the Godot project. This couldn't be further from the truth. The whole mission statement revolves around Godot, how could he say it's unrelated. The W4 website also claims they will be supporting Godot financially. But now he says they are independent. It doesn't add up. And look, I'm not against making money. I think some people think I'm just hating, and I'm not. I feel lied to and betrayed. He has lied to my face multiple times and I didn't trust myself. But this is the end of the line.

You can read the mission statement on the W4 website first to see what was originally claimed. Then you can read the thread that Juan started, including some real questions from me and some other people. I'll be honest, I was upset, and I'm still upset. I said some stuff that I feel was true, but I was also very blunt.

You don't have to take my side, I understand. It was hard for me to believe. I noticed the strange things for years and I just couldn't accept it. It was a grift. Maybe not the whole time. I think originally it was in earnest. Maybe something happened behind the scenes I don't know about. But that fact that Juan is painting this picture that Godot is out of money and begging on social media, meanwhile with $8M in his wallet, doesn't sit right with me. I can't with good conscience support this project anymore. So I think I will close the forum. He blocked me and I blocked him. I don't want to be associated with a liar. I invested so much into the engine only to realize it wasn't what I thought. He lied and I'm done with it. I'm really

sorry.

Lilly Byte Games

Source: Twitter

You know what.. people want to know why I resigned as a Godot mod and why I've "seemingly" turned on Godot after years of loving the engine? There are few things I am passionate about, but this is one of them... and I'm gonna flame it the fuck up here.

I am tired of getting shit on by "Godot bros" whose only experience with Godot is what they see/hear in their limited experience of being fresh and new to the Godot train, and not yet had the curtain pulled away from their eyes to Godot's reality. Let me tell you the story of how I found Godot, and the journey of how I went from a bright eyed and bushy tailed Godot lover to the person who both loves and hates Godot with a seething passion... and let this story be a herald and warning before you invest your life.

It started with me back in 2016/2017, when I was developing games on Twitch; I was using a framework that wasn't quite cutting the muster. I found Godot, and it seemed like a good fit for my project. So, I switched and started tinkering away... so far, so good. I was streaming regularly on Twitch then, about 6+ hours/day building my game. I hadn't yet joined the Godot community, and Godot's 3D was seemingly fine for the initial stages of my project. Everything was going grand...

Then one day, as I was streaming on Twitch. The original admin of the Godot Discord found my Godot stream... and they did an at-everyone on the Godot server and my Twitch channel was flooded with hundreds of Godot users pouring in... This was my first interaction with the Godot community, or anyone else in the Godot community. Everyone seemed friendly enough, so I joined the Godot Discord server. I found a happy place, and a place that I could call home.

I started streaming less on Twitch, and spending much more time in the Godot Discord voice chat... where I spent much of my time streaming my own game dev there and helping other users find their way with Godot. In this era, I was in the honey moon phase with Godot. In this time period, I spent just as much time helping other people with their Godot projects as I did with mine. I was on the Godot Discord 8-12 hours/day, consistently... I pretty much stopped streaming to Twitch because of the time I was spending on Discord.

In the Godot Discord community, I found a very good circle of friends... who are still my closest dev circle of friends to this day. Then one day, the voice mod needed to be replaced, and well, they were looking for a diversity mod and I was there half a day anyway... So, I

joined the Godot Discord mod team... and not to toot my own horn, but I was a goddamn good mod. I never abused my power, I was especially careful with personalities I disliked, and usually let other mods handle those.

At this point, I was spending more time helping other people with Godot than I was working on my own projects... but, I was still working on small games for fun, taking commission work here and there for 3D models, and occasionally working on my bigger projects.

As time went on, I started seeing cracks in Godot... while the fundamentals were broken, the more advanced tools that barely existed in Godot were really broken. This was not only a problem for me, but a problem for a lot of users I was trying to help. Myself, Duroxxigar, and others in voice chat would help people the best we could... but, in a lot of cases there were no work arounds in Godot except for "build it yourself". So, I started being more vocal about Godot's issues...

While I'm not technical enough to make engine PRs myself, many in voice chat were; and they would make PRs to Godot... and we would tell them how to do it, and look hopefully at the fixes that "would be coming in the future".... Except those PRs never ever got merged... PRs sat for years, untouched. When people coming in wanting to add feature PRs to Godot asked us how long it takes for things to get merged... we were honest. It could be years, because it could be. We watched as PRs grew stale, ignored, untouched... I complained about this specifically, and I was told: There's no guarantee PRs will ever get looked at or merged. This translated into: You could spend months or a year+ on a PR just to have it not even looked at. Or worse, outright rejected because Juan would often say, "No, I plan on doing this myself eventually..."

This is around the time when I started getting a little more aggressive with Godot's problems. Okay... PR fixes are pretty much getting ignored unless you're in Juan's "Boy's Club"... and if not, you are spending your time, effort, and life on a gamble. So, here we are now, about 2019ish with an engine that has fundamental broken problems... and people with solutions to those problems are being rejected for the dumbest fucking reasons... one of which is "their personality isn't the right fit for Godot". "Their personality isn't a right fit for Godot?" What the fuck.... Bish, it's code. I don't give two shits what their personality is like if the code works. Let them fix the goddamn problems you are not fixing... or add the goddamn features you are not adding.

So, there we are... this is where I start to see the first true light of Godot. If you are not a part of the "Godot Cult Insiders" your time is wasted. We weren't the only people to see this either... Over the years, we had AAA engineers; more than a couple, come into the voice chat and talk to us about Godot and their concerns. They were seeing what they were seeing... they wanted to invest their time in Godot, but them seeing PRs sit uselessly in the void put them off. This became a theme, not only with AAA engineers, but other skilled people who

wanted to contribute to the engine. Also with people we wanted to help contribute to the engine. Rejection often came as “Not a right fit”, “I’ll do it later”, “Godot doesn’t need art tools.”

The “belief” I had in Godot being community driven was an absolute lie. The discovery that Godot is “community driven” is a lie. Okay, fine… Godot started hiring more devs… maybe things will be okay, maybe things will improve…. However, as happens with Godot… all good things turn to sadness. Most of the money they get in funding is attached to the condition that specific features be implemented… basically, “pay 2 win” contribution. I mean, in the end, it makes sense… Godot needs funding. But the degree to which this happened again and again and again, while capable and experienced engineers and their PRs or ideas were getting rejected over and over for the dumbest of fuckwit ideas was also getting common.

Juan’s “Godot philosophy” was killing Godot… and definitely killing my bright eyed and bushy tailed vision of where Godot could go. So, I got again more vocal about Juan’s constant “It’s my engine, I’ll do it my way” take while claiming it is “community driven”. Sure, if you have a small fix for something that exists… there’s a high likely hood that PR will get merged. But, a full on feature? Good… fucking… luck. Everyone skilled enough to want to contribute without sucking Godot Jesus’s dick shyed away from wanting to.

But still… I had hope for the engine, despite ALL the shenigans and bullshittery I saw, heard, and experience. I continued to help people with Godot, to work around the problems, to explain what can and cannot be done with Godot. The difference between me and someone else: I was honest with people about the engine’s capabilities… what was missing, what was working, what could be realistically done with the engine without a professional team of engineers to build parts of the engine for you. Again, I’m not a technical person… so I wanted to somehow get artists involved in the development process of Godot to help refine the tools… to make them more artist friendly and give artists functional tools. There are PRs/suggestions where I teamed up with devs to do it.

Once again, good improvements on existing tools rejected because of Juan’s stupid philosophies or that he just didn’t “understand why a gamedev would need that tool.” It got repeatitive… it got tiring. So, at this point what I am I supposed to do to help improve this engine? I tried being nice… I tried teaming up with developers to bring meaningful improvements to the existing tools… I tried bridging between artist and programmer…. Every meaningful avenue I tried to take to improve Godot was shot down by Juan’s philsophy of “I don’t understand why this is needed” or “I’ll do it myself later” or “This will come in a later version”… and it never fucking did, ever.

So, let’s go even further down the line to the other cracks that started to appear in my love for Godot… these super huge crevices not withstanding…. The next biggest crack I saw was on the mod team. There was a contributor who was on the Godot Discord spewing heavy

anti-LGBTQ non-sense. The admins of the server were besides themselves in what to do… for anyone else, they would have banned outright…. For this situation, because this person was a contributor, they were just refusing to deal with the situation. So, you know who had to step up do the right goddamn thing? Yours truly… because nobody else would do it. So, yup… hello new crack in my love for Godot. You can't stand up and do what's right when the right needs to be done… even if/when it's hard… that's a problem, and that's not my only experience with that… because hold onto your hats until later.

Not long after, I think I was finally at my wits end with Godot… I had spent years and thousands of hours helping people in the Godot community… developing shit with Godot, sacrificing my life, my time… to the betterment of Godot for fucking nothing. So, 4.x alpha drops… like, about 4-5 years after it was initially supposed to, mostly, because Juan kept adding or changing stupid features nobody wanted or needing because he felt like it that day…. And surprise surprise… the renderer, while faster than OpenGLES3 still has significant issues. 2017… 2018… 2019… 2020… 2021… 2022… for… this? What… the… fuck… All the Godot philosophy preachiness… for this?

So, anyway… an argument ensues in the Godot mod chat channel; and it's a bit of a doozey, we're all a bit passionate in the there sometimes; we all love Godot in our ways, so it got heated because I wanted Godot to be more… and then… well, all hell breaks loose. pycbouh goes into full fucking gaslight mode because his fee-fees got hurt, starts swearing and cursing on people and making arguments nobody is making and arguing against imaginary arguments [a constant thing for him really]… Remi goes on a rant about how Godot is Juan's engine and want the community wants/need isn't important… and that, that alone is was a sword in itself. The community I spent fucking years helping, years building, isn't important? what… the… fuck… At this point… that's when Xan comes in and tells me, "You're being low and dishonest"; and to Xan's credit, I don't think he understood what was happening at the moment, so I don't blame him… So, here I am… in the midst of this firestorm… pycbouh trying to gaslight the fuck out of me, while Dazz is trying to calm him down and getting cursed out by pycbouh too… Remi is telling me the community don't mean shit… and I just say FUCK IT….

I resign as Godot mod, I'm done. I'm just fucking done. I spent years promoting Godot, trying to help Godot, and being a literally fucking crusader for the engine… hell, on Twitch I was literally called "the Priestess of Godot"…. Years of my life fucking wasted to come to this kind of goddamn end, and all I wanted was to do MORE with Godot. To improve the tools it had… and not a person would listen to what I had to say. So, to say I became more aggressive recently, you better believe I have.

There is much more to this story, but I will keep it to myself for now; such as how pycbouh repeated his behaviours with other people… yet, still is mod. If I had pulled the shit pyc pulled, I would have been kicked from the mod team… but you know, contributor favortism.

And the sweet fuckin' blessing on this shit was... I was "offered" mod position back IF I would recant the shit I had accused pycbouh of. No. Fucking. Way. I experienced what I experienced, and he did the same shit to someone else... glad I refused that.

So, yeah, Godot leadership can go fuck itself. It wasted my life, it told me that the community I held dearly wasn't important, it told me our issues weren't important, and it tried to coerse me into recanting something that happened. And the whole "community driven" thing is one big goddamn lie. Godot is driven by Juan's whims of the day and purely Juan's whims of the day. So, good luck to those who want to contribute anything major to Godot. You gotta suck the Godot Jesus dick or dump a load of money. That's what Godot is REALLY about. I still love the Godot community, and I still think fondly of the engine for what it does well, and what it COULD be... but not with Juan in charge.

And Godot bros, go fuck yourself. I've been in the trenches for years, you haven't.

Now, let's add something else on top of this. We have Juan's indecision of what the fuck he wants Godot to be... one moment he says "Godot features will make you want for nothing", the next he's saying, "Why would a gamedev need that?"... then it's "Godot isn't for AAA dev" Then it's "Godot is a professional engine", then it's "Godot needs more technical barriers so people can learn proper game dev" then it's "Godot needs to be easier for beginners." Like, make up your fucking mind... figure it out...

So, on top of the Inner Godot Cult of PRs... the leader of said cult has no idea what or where Godot stands, or where it should be going. If you don't know where you are, and don't know where you are going... that's called being lost. And that's where Godot is... it's lost because it's leadership is drifting in the wind with no sails/direction. No idea what it wants the engine to be, no idea where it wants the engine to go... and no willingness to heed experts in their fields because "personality".

And by the gods... I loved the Godot community, and I loved the dream of what Godot could have been. Yet, I woke up to the hard reality of being told how the Inner Godot Cult sees the community; and now, I can't unsee that truth.

Oh yea... and all that time, effort, and energy I put into actually using Godot, promoting Godot, and helping people worm their way through it's quirks? Didn't fuckin' matter one bit... as far as Juan and the other dickheads were concerned "I wasn't a contributor", so meh. Yup, not a contributor. Despite all the community building I did, the amazing voice community I had spent years building up in the Godot Discord, the countless hours I put into helping people... yeah, not a contributor to the engine at all, not one little fucking bit, nope. So, when they say on the website "contributing code is not the only way you can contribute to Godot"... it's just goddamn lie; something else they say that they do not believe.

So, am I upset and angry with Godot? You better fucking believe I am. Juan is a walking/

talking ego writing checks his coding skills can't cash. The leadership is a Godot Jesus dick sucking cult that approves PRs based on how much they know/like you or not. And "fuck you" if you try to explain your experience as an artist using Godot. Even Remi told me not to listen to Juan because half the stuff he says is bullshit. Like, seriously, we're not supposed to listen to or heed the LEAD DEVELOPER of the engine we use? It's a clown show... and Juan is center fucking stage to the act. And thus concludes my Tales of Godot.

If you like the engine, please, use it. It's got some nice features for some games... but temper your expectations, don't invest your life... and definitely don't listen to the clown show that is trying to sell you a broken dream.

Winter Pixel Games

Source: Twitter

There's some massive pros to the project (FOSS being the largest). The Open 3D Engine announcement is definitely a welcome addition to the FOSS community, but a few of the Godot Engine core devs seem kinda bewildered that Amazon would invest time and money here rather than sponsoring or helping Godot Engine in some fashion.

But to me, it's pretty clear. The fact is, it's somewhat tough for seasoned and experienced game/C++ developers to work with the Godot Engine source. The codebase and lead dev are very opinionated (which is USUALLY a good thing). And I'm not talking about necessarily contributing, this is about working in the codebase itself. For example: I can't use Godot Strings in any other modern C++ template collection because String doesn't implement move operator 😔. This leads to workaround after workaround in our codebase itself. (A simple example true, but one of many).

And there are just some very unergonomic things about the Godot source right now. Stuff that would just never fly with experienced teams or devs. And from what I've seen it seems to put off experienced devs and more established teams. Recently I've come to discover this old Godot issue (as I understand this issue has somewhat become a sore spot in the Godot Engine community) - Using the slowest data structure almost every time. Regardless of the presentation of the aforementioned issue (probably not the best), the technical criticisms in the issue are sound. And that aside, a lot of criticisms of Godot source code come in the form of "Why does Godot source do things like this". And the general answer from the core team is, "Godot does A this way because of X,Y and Z". And I have no problem with this reasoning. But the reasoning has to be sound. What I'm starting to see more and more of is that X, Y and Z tend to be false premises, which end up resulting in bad decisions.

And I don't believe it's intentional, I think it's just due to the lack of actually using the engine

by the core team to ship real products. This is really why dogfooding is SO important, and why Epic/Unreal have such an unbelievably great advantage with their engine development practices. I was told in IRC back when Godot dev team used IRC for communication, that Godot source is very NIH (Not Invented Here) by design. I wasn't familiar with this term but boy am I now. (NIH is the tendency to avoid using any third party code other than your own). Need a STL? Godot has their own implementation. Need a physics engine? Godot has their own implementation. Need an http stack? Godot has their own implementation. Need a scripting language? Godot has their own implementation. etc, etc, etc. There are obviously pros and cons to this, but IMO it's currently balanced on the wrong side here. It leads to slower development velocity, because when you do everything you have to maintain AND fix everything. An absolute massive amount of work!

There's a very real reason Unity just uses Box2D as it's 2D physics engine... And ya, when you implement everything yourself, you're going to have to implement range iterators (as well as a whole bunch of other things but don't get me started). For example, our project was just absolutely HAMMERED with malloc/free overhead per frame. Literally TENS of THOUSANDS of calls. The reason? Godot's 'STL' doesn't do map iterators (just due to lack of resources)... I encourage the Godot core source team do some introspection here and not just wonder why the project doesn't seem to attract certain types of resources, sponsors, or developers. It might be gut check time.

Camille "pouleyKetchoupp" Mohr-Daurat

Source: GitHub (read all comments)

Just to be clear, I've been bullied two times by different people on this project and every time project management reacted by siding with the bully, so I've left and I'm never going to contribute again.

I wasn't going to get into more details, but since you are, let me make the story straight.

Here are my concerns:

The first time, I was bullied for months by an individual while contracting for the project. He was also bullying other contributors. I did report issues multiple times to project management to no avail. What you call meditation was a meeting with the project manager and the bully, during which I was asked to apologize to the bully, while he was yelling at me. For the next 6 months, absolutely nothing was done. When I tried to raise the issue to the head of the project, he told me it was a sad situation but we can't reject anybody from the project. What it took to get the toxic person removed from project responsibilities was for me to gather all evidence and present them to the PLC, after he had started to bully yet

another person. He was never completely banned from the project.

The second time, a similar situation started to happen with a different bully. It was at W4 games, so the project CoC wasn't involved, but the same project managers handled the situation in the same dismissive way. That's why I left.

I hope things can one day get better on this project so other people don't go through the same experience I did. For that project managers need to take situations like those seriously, which in my experience wasn't the case even after the CoC team was created.

💡 For more context, read my comment on the Waiting for Blue Robot subreddit.

Endri Lauson

Source: YouTube

For them, it doesn't matter if Godot needs money, because they know any random will eventually implement what needs implementing in the engine, so the engine, in theory, never needs money to exist and be improved upon. You should never underestimate the power and the significance of the contributor-developers (the people who actually implement the features requested and used by the community), who are starting to see right through this bullshit, and soon there will be no talent left to improve the engine.

I implemented the new lighting/tone mapper used in Godot 4 and I also back-ported it to Godot 3 which, we all agree, is a fairly significant contribution. The previous tone mapper used in Godot was the subject of ridicule whenever a discussion about engines was brought up online, and the community was clamoring for a replacement for years. I'm a game industry veteran and I brought it upon myself to go on and implement a next-gen one to give it parity to Unreal Engine 5.

The change was lauded by the community, and to this day, this is still one of the PRs with the most reactions of Godot's GitHub. And yet, they did not even have the decency of adding my name to the list of contributors. I understand that they cannot possibly add every single contributor to the list, but according to the rule set, allegedly, they only add contributions that they deem "significant". I reached Rèmi in order to clarify that maybe that was just a mistake and they just forgot, and I was shocked to learn that my message was left on read. I guess that for the board, a brand new, next-gen-looking tone mapper is not a significant enough contribution.

Which gets me to the next point: they only add to the list of contributor-members who are

close friends with the core board... well, unless you consider just changing the name of a few variables to “min”/“max” a “significant” contribution... Because of this behavior which is antagonistic to the spirit of open-source software development and so egregious that it would make the RedHat project team blush, I just “silently removed” myself from the project to not steer up any drama. I have not contributed since and I won’t be contributing to the upstream project anymore. Imagine donating your spare time to work on someone else’s project and receiving nothing in return, not even an small acknowledgment in the footnotes of a large list.

Now, imagine that the couple of people running said project are receiving MILLIONS, while I donate my time to receive nothing in return. This is why many talents are leaving. They seriously have not implemented per-pixel motion blur yet? My personal, private branch of Godot has that since 2021!

Alexander “nicholatian” Nicholi

Source: Twitter

Godot’s entire existence is measured in Unity copium. They’ve been doing this since before 3.0 was even announced. This “race” has been going on for years, and will never end, because Godot survives solely via online guerrilla marketing.

On paper, Godot is a game engine. The reality is much more Soviet: it’s an online clout farming operation for a couple of random Argentinians. Godot may not have a physics engine worth a flying hoot, but you can be sure that they’re getting paid like they do! It’s a very successful grift, I must say. a lot of faithfuls in their replies, churning mindshare and making it real by fiat. Well done.

One of the things that really drove home my conclusion that it’s more of a LARP than an actual endeavour is the Discord’s running joke about “when is 3.0 coming out”. Being a popular game engine, they got the question a lot. Speaks volumes that they created a channel to mock it. It’s an unfortunate defect of open source: you cannot make the developers care about your legitimate issues. You are not a stakeholder.

They have every legal right to blow all the money you donate on hookers and blow. And if their engine is missing something, add it yourself. Of course, they should care! And ordinarily, with all else equal, they probably do! But they don’t always, and if something is important to the health of their game engine and they disagree with the public, their word is final, and there is no practical recourse.

Ostensibly, they are creating a game engine. The truth is subtly different: they are here to

get paid to create a game engine via donations. As long as everyone believes in them, nothing else matters. They can have a broken game engine as long as people want to pay them for it.

Pawel Jarosz

Source: Twitter

In 2020 I realistically was considering making Witchcrafter in Godot. It was (and is) very promoted, all around was bragging about it, I was envy of some glossy features and as a budding indiedev I thought it's an easy way. But why I didn't moved to #Godot in the end?

Godot is presented to us almost anywhere I can imagine, so it's not strange I bumped into it at some point. It is also promoted as a perfect tool for indiedevs with a lot of tutorials and examples and ngl - it's impressive (the same with Unity, I also considered moving to it). Envy of the glossy features, out of the box solutions, I started making a simple platformer base with all the stuff I want. But, oh boy, was I misguided by the easy learning curve.

Having using Unity, Defold, Blitz 3D and some other smaller engines I though it would be natural. I couldn't make simple stuff, because Godot's core/workflow is way different than others. I imagine now also how hard it is to switch from Godot to other 😅

But this is only my impression, my experience. I started asking others about their experience. And even back in 2020 I came into opinions that Godot is overhyped and is for them a pain you know where. I finally came upon the opinions highlighting Godot's strong points too, but also realistic for what it is good.

Some of the gamedevs opened my eyes then: "I think it's a great example of the failure of "Open Source®"—a big buggy mess with hundreds of contributors and no quality standards, terrible, half-finished documentation, and a cultish community with thousands of rabid fans (most of whom have never even used it)." This is from a person with a vast experience and who used UE, Unity, Godot, Defold, Love2D. This person said too: "We ended up having to scrap it because the performance was just too bad. The per-pixel rendering was at least 5x as slow as Defold, and GDScript was very slow too." There are also bad and good points for the other engines, so I was then very thankful for the whole comparison and some really practical advices.

What was a shock for me back then was this: "You probably noticed on Twitter, Remi (the unofficial vice-leader behind Godot) was the first and most vocal person harassing the Defold guys about the licensing nonsense, "it's sad to not be opensource", and bragging about his number of contributors, and Juan (the great cult leader) was bashing them from

his personal account too. They are not good people. Ugh, they even got a $20k grant from the Mozilla foundation specifically to improve Godot's use of web technology. Juan said that he used it to pay for his living expenses and... a year or so later, Godot still didn't even have functioning web builds. How that wasn't considered fraud, I don't know."

And it was, again, in 2020. This year you might started seeing #CancelGodotEngine posts by one of the banned contributors @Xrayez. The narrative is so weirdly same. I'm reading all those tweets and the worst thing for me is that Godot spreads hate to other engines, instead of love. Community and open source foundations are totally OK with some of the behaviors and movements, but when other makes mistakes. Oh boy, Defold licensing thing was nothing when you compare it to how the Godot community was triggered after Unity stuff this year... I got into that bubble myself 😔 I then digged deeper into Unity's case and while some things are still 🚩 to me, I formulated broader opinion. Indie community is amazing, because there are people realizing their dreams, living the passion 💔

No matter what tool we are using. And while I myself am promoting Defold mainly, I never claim it's the best engine for everything, because I was misguided by Unity and Godot's bubbles, they're best for everything. In fact Defold is focusing on a very narrow audience and tries to please them at first. So, I am also not very experience, so my opinions are still not so mature, but another gamedev with a vast experience (and talent), who's advice I took to my heart 💔, when I naively asked about which is the best: "Actually Defold or Unity or Whatever - is not important. Choosing correct tools for the right kind of game is the most important thing in game development. It is all about the experience. That is it. People should be aware of this. Other than that this is a pointless popular bubble." It wasn't then so clear to me, but - how wiser those words sound when you actually experienced such bubbles.

Others, whom I asked back then was saying simply the same thing: "In the end I am an advocate for "Using the right tool for the Job". If you are trying to build a GUI heavy app and not a game then use Flutter or React Native, if you want to do a beautiful 3D game then Unity or UE is probably the way to go." "But if you want to make small 2D games then I do think Defold is currently the way to go."

I am making small 2D games with a priority for easiness of use/ prototype and simple multiplatform builds. This is WHY I chosen Defold. Plus it's free 😅 Godot has its advantages, is definitely something phenomenonal and the support from community for this is (in most cases) very pleasant and broad 💔

I believe some things might have been better, but that's the case with everything. Nothing's perfect. I know myself a lot of talented devs using or switching to Godot and making successful games. Don't get me wrong, it's definitely not about claiming Godot itself is bad. It's not perfect, but it's not useless either It's about consideration of your choices always. Focus on choosing the right tools for your game. And don't spread hate. We are an amazing

community, we don't need toxicity. Support each other, because that's our strength! 💪❤️

Tynan Sylvester

Source: Reddit

My main concern with Godot at this point is that it seems to be trying to be all things to all people. It's trying to appeal to the "my first game" student market, via visual scripting and GDScript and so on. But it's also trying to hit AAA features like advanced rendering and a built-in particle engine.

This lack of decided focus manifests as IMO mis-spent effort on things that almost no serious indie should be using (advanced rendering features which look pretty in demo videos, visual scripting), while deprioritizing things that absolutely every indie should be using (C#).

From what I've read the C# support still quite rough. Good first-class C# support is a must. We need a language that's very productive, quite fast, bug-resistant (e.g. statically typed, no globals), mature, portable, has libraries, and handles very large codebases. Only C# fits.

GDScript looks nice for really really tiny games, but it's obviously never going to scale close to anything like RimWorld; its performance is way too low and the code isn't nearly safe enough. Most indies beyond the "my first game" level should not be using this.

C++ is good if you're making a hyper-optimized engine code. But productivity is really low for C++ since it's so unbelievably arcane. Very few indies should be touching C++ IMO.

C# covers the needs of 90% of indie work for serious projects:

- For "my first game" level coders, learning C# is ideal; the language has great error messages, tons of documentation, and is very productive.
- For mid-indie developers like me, C# is ideal because of the excellent balance of productivity, speed, libraries, and scalability.
- For AAA developers, you need C++, but these people are irrelevant for Godot because they'll be writing their own engine or using something Unreal-tier anyway.

I don't think, at the indie level, that it's a good idea to be trying to compete with big companies in terms of graphics. So from my point of view all the whiz-band shader and particle features are NOPs at best. Making use of such features really requires a big team of artists; the Venn diagram of studios who might use Godot and studios with big artists is very close to zero.

I really hope the Godot team doesn't get sidetracked in working on advanced AAA rendering features, nor on "my first game"-level ease of use stuff.

This engine will blow up when small-but-professional indies start using the engine to make really popular games - Games on the tier of Factorio, Don't Starve, RimWorld, Hotline Miami, Stardew, Terraria.

I really want to do that (if indeed I am capable and lucky enough to succeed) but the engine needs to cover the bases to make that possible. Currently it's close but not quite there.

We don't need animations or particles or PBR or visual scripting or networking. We just need the fundamentals right. We need really solid C# support, with debugging and all, and we need the engine to be generally bug-free. I hope to use it one day!

Mariusz 'koder' Chwalba

Source: GitHub

- I released an Early Access title ΔV: Rings of Saturn using Godot 3.1.1-stable.
- I am getting frequent crash reports, which I traced back to Revert "Fix AudioStreams::stop possibly causing a small noise" #28469 (or other audio-related issues). The fix is not in current published stable build of the engine.

Look at it from my perspective. I have released a title. There is engine-related bug, that's fixed in current master, that causes my game to crash every couple of hours. I did some workarounds, but there are some areas that I just can't reach with game code. I'm releasing cross-platform, so compiling my own version of the engine is not really feasible.

It's bad with Early Access release and would be a catastrophe in a full one. You just can't have a (stable marked!) engine that crashes a game.

Ideally, the bugfix should get into 3.1.2 and be published days after it was fixed.

I took a leap of faith by developing my game using Godot 3 - when I started it was in early beta stage. Now I feel like your focus is on Godot 4, but the 3.1.1 you are publishing currently on your site is *not* stable as advertised. I would implore you to revise the release cycle to favor stability of published project over development of next feature or major version. Having an "unstable stable" engine is not doing anyone any favors.

Favkis Nexerade

Source: Godot Forums

I'm 3 years into developing games on Godot and some of them are almost finished, however, when I started back then, encountering countless broken features and bugs, I thought they'll get fixed with time, so I either did bad workarounds around bugs in engine or simply ignore some, waiting for them to get fixed. However now I hear that G3 wont be continued, why? I spent 3 years working on my games and this engine still has broken graphics, broken animations, broken file system, broken physics, broken importer, broken resource instancer and dozens of other broken "features".

What's my next move? I am not moving to G4 because I see no point in doing so, if they can't finish G3 they wont finish G4 and I do not wish to start making my games from scratch, because no matter what I tried, moving any of my G3 projects to G4 doesn't work at all.

I feel cheated, I spent so much of my creative energy into this engine and now met with "we wont continue making G3". Here are game I'm made so far:

- Souls-like game
- STALKER-like game
- Strike-like game
- Souls-like game
- Online/Co-op/SP/PvP low poly sandbox game

Ross Grams

Source: Defold Forums

Godot has some nice core ideas (the scene/node system), plenty of features, and is fairly easy to learn, but it's also bug-ridden, slow, has very incomplete documentation, and a cultish community.

Bugs: Hopefully it's gotten better as the community has grown, but when I was using it 90% of the engine development was done by Juan (reduz). He pumps out code amazingly fast, but he also doesn't pause to test it or document it, and he rarely goes back to fix old code. There's a strong focus on adding new features, very little on fixing things. His most common response to serious bug reports was "oh yeah, there's no point fixing this, I'll be rewriting it for 3.1". So there are lots of broken or unfinished features lying around. There's really no such thing as a stable release of Godot.

People think the open source thing is awesome, but to be honest I'm not sure if it is more positive than negative. They routinely use it as an excuse for terrible documentation and

sometimes bugs. Several times I (or other people) asked for help on how a feature worked and the answer was: “I don’t know, go read the source code and let us know what you figure out”, or “if it’s broken, feel free to fix it!”. Also, there is no great review process or testing for code to get added to the engine. Random people write stuff, add a pull request, and chances are, it is added to the engine a day or two later. I was in a discussion once about how modifier keys were handled. Some guy decided he knew the solution, wrote some code, and the next day or so it got pulled into the engine. But all it did was move the problem from one place to another! As far as I know there’s nothing in place to stop this from happening constantly. It looks like they don’t use unit tests 28 or any other kind of code testing (at least not with any consistency).

Slowness: (scripts) GDScript is slow. I haven’t done any tests in ages, but I am confident if you test the same code between GDScript with Godot and Lua with Defold, Godot will be 10x slower or more. This won’t be noticeable when you start working on a new project, but as things get more complicated and you add more and more scripts, you will see the framerate slowly get lower and lower. I think someone did a test when they started adding support for other languages and the others were significantly faster, but I don’t know the specifics. This has been tested and reported multiple times over the years, but the Godot devs have simply denied that the tests were valid and haven’t shown any interest in trying to find bottlenecks and optimize.

Slowness: (rendering) Godot also seems very slow about rendering. As far as I can tell, it still doesn’t do draw-call batching at all… You can kind of get away with this on desktop, but mobile devices can’t handle it. See here 28, here 16, and here 16. Some quotes from Juan regarding 1000 draw calls: “This number of draw calls is not much of a problem in OpenGL. Batching is not really necesary.” and “…might be better to eventually wait for phones to get better than fixing this.” For comparison, with Defold it’s pretty easy to stay below 50 draw calls for everything in your game. Also it has overdraw issues. I was working on a platformer and wanted some parallax backgrounds, but I simply couldn’t draw more than two layers over large parts of the screen before the framerate dropped below 60 (with everything else going on of course). I recently tried something similar with Defold and it had no problem drawing a dozen or so overlapping screen-sized images. It’s quite possible that Godot doesn’t do any depth sorting either, which could be the problem.

Documentation: Godot has a pretty sizeable community to ask for help, but its documentation is very bad. To start with, a lot of it doesn’t exist. When I was using it, they announced 7 that 60% of the class documentation was missing. 60%!!! And that’s just the stuff that doesn’t exist at all. The stuff that does exist is far from being good documentation. Most of it just reiterates what the name of the function already tells you. It doesn’t necessarily tell you what units all the arguments use, what each argument does, what the corner cases are, and almost never has examples of use. This is just the API documentation, never mind actual manuals that tell you how to use things in a pleasantly readable way.

The community has made some valiant efforts, but I don’t see how the situation will change much. New features are added without proper documentation. Unless that changes (either by not adding any new features or only adding them after they are fully documented), it is a losing battle. No one should have to delve into the C++ source code to figure out how a feature works.

Reactions and Propaganda

After the online publication of this book, it was fairly expected that it would receive negative and even angry comments by Godot followers. The book even found its way into Godot's circle of cultists in some cases, which was spread by some Godot followers themselves, mostly for scoffing purposes and the suppression of doubt.

The following describes some notable reactions to the contents of this book by Godot followers and cultists. This is an explanation on how cult propaganda attempts to counteract facts. At the moment of writing this section, everything that you see here is publicly accessible. It's fascinating to observe how cultic behavior manifests in various forms, once you know about it and are outside of it.

Case A

The announcement thread of the book at *unofficial* Godot Forums was immediately locked by the admin without any prior discussion[1], not giving it a single chance to be discussed with community members.

The author of this book has also received an account suspension for two months! From the standpoint of human values, to be banned from a toxic cult community is the highest reward that one can dream of! If you want to find an easy way to get banned without actually breaking any rules, all you have to do is to share this book at Godot Forums or similar places. 🙃

Indeed, cults usually block information that criticize their governance and management, especially when such information directly relates to former members. The only exception why such discussions might be allowed is for scoffing and scapegoating purposes from within a cult.

Progressively throughout years, Godot has been limiting the scope of topics that users and contributors are permitted to talk about. For example, Godot's Q&A website used to contain "Gossip" topic in the past [2], but now it only has a single "Engine" topic.

Case B

That feeling when your pseudo-philosophical comment is used in a "book" for Godot anti-

propaganda[3]

For your information, this person's opinion about philosophy was taken as the base behind Development Philosophy section. But due to the ethical dilemma, I deliberately decided not to mention him directly at the moment of writing this book in order to potentially avoid retaliation of this person by Godot's cult leaders. But since he explicitly decided to present this book this way, this dilemma is no longer relevant now.

His words about the lack of development philosophy, and on importance of having one, was quite spot on. Unfortunately, even having such bright ideas didn't protect him from being affected by Godot's cult indoctrination, which manifests in several ways:

1. He devaluates his own opinion, by labeling his own opinion as "pseudo" philosophy, perhaps in the fear of scorn by fellow Godoters if they would ever notice it, which means that Godot followers would treat this person to be *conspiring* against Godot on the same side (again, this has elements of **us-vs-them** mentality).
2. He is likely to have doubts about Godot. But he seeks external confirmation within the circle of Godot contributors to suppress his doubts (see "Confirmation bias"[4] phenomenon and Groupthink section).
3. Despite having such bright ideas, the "book", as he puts it, he still sees the book as anti-propaganda (or, if we "correct" his expression, he probably meant to say "anti-Godot propaganda"). I was definitely pro-Godot for the past years. But due to the cult nature of it, I cannot take a position of pro-Godot, but this position is definitely not anti-Godot either. At most, this is all about cult deprogramming of Godot. No matter how perfect Godot may appear to outsiders of the cult, this destructive cult state is pretty much like *"a fly in the ointment"*, and as old proverb tells: *"A rotten apple spoils the barrel."* Unfortunately, this pervasive, invasive, viral cult state will likely never be lifted from Godot. This is explained in the Afterword.

The above is pretty much cult apologist's behavior (it doesn't matter whether such behavior is conscious). This reinforces existing foundings that, even the smartest people on Earth are still susceptible to cult manipulations, including the author of this book. 😊

However, we could also interpret this person's intention as net positive. You may have figured that cults usually block information which exposes them as cultists. Imagine that someone shares a link to this book in the negative aspect. This way, there's a chance that the discussion won't be blocked by cult leaders, but at the same time, this would create *doubts*. To doubt something is very beneficial for eliminating limiting beliefs. Doubting is the critical thinking tool which facilitates the process of cult deprogramming. I would like to believe that this is the actual case. 😁

People got curious about this all, so a link to this book has been shared, by the same person who got apparently disappointed (?) by the author of this book.

The project manager of Godot appears, and replies:

> *He has been banned half a year ago after multiple warnings following breaches of Code of Conduct and general disruptive behavior.*

If you haven't, read No Longer a Blue Robot which describes this incident. You should also be aware that Godot's CoC is only used for punishment purposes rather than as a guideline. Even if you don't break rules, you may still be banned! See Authoritarianism - Governance for proofs.

Someone replies:

> *You know, it almost read like fair criticism until I have read the "He has been banned following breaches of Code of Conduct" part.*
>
> *WAIT WHAT, GOOST'S CREATOR?**

Notice that this person had a good impression that the book reads like a "fair criticism". No wonder: I've spent a great deal of time and effort collecting, organizing, and finally writing down everything there is related to Godot cult, with actual facts, evidence, and testimonies.

But then, see what happens: this person reads a message by Godot's project manager (see previous quote), and instead of trusting own judgement, such a person chooses to *believe* the authority instead, which is pretty much bootlicking in this case (see "Appeal to authority"[5] logical fallacy and bias).

Almost immediately, this person is then surprised by the fact that the I'm also the creator of Goost, a somewhat popular general-purpose extension for Godot. Indeed, I'm not a hater of Godot, unlike how Godot cult followers want to present it.

A Russian maintainer of Godot, who we covered in previous sections, also suppresses doubts in Godot followers:

> *This rabbithole goes deep, but I'd recommend not following it.*

He absolutely knows the fact that his troublemaking actions are covered in the book. This

hypocritical behavior is certainly typical to Godot leadership. He's definitely not interested in people knowing the truth about his "Russian Warrior" personality that he attempts to hide from the sight of Godot contributors.

Someone replies:

> *Yeah, I'm kinda sorry for asking about this, probably not something this community is interested in rehashing.* 🙂

Here again: there's anxiety and fear that this topic might not be tolerated to be talked about in Godot community. Community members of healthy communities (where critical thinking is not discouraged) usually don't fear nor regret asking these kind of questions.

Perhaps this expression of regret of asking a question about this originated from warnings via private messages that this person might have recieved from cult leaders of Godot, such as Juan and Rémi, which is totally possible given previous instances of such intimidating behavior by Godot leadership. We can only guess here.

In general, these patterns of behavior are quite typical to Godot followers: they will go as far as to renunciate their previous beliefs so their now false identity/opinion becomes in alignment with cult leaders' claims, basically showing unquestionable attitude to cult leaders. Hint: you can find some examples of this behavior in the Authoritarianism section, namely in the proposal which contains lead developer's *"you don't have my consensus"* reference. I don't want to mention a particular member here directly, due to the ethical dilemma which stems from this, again (not to be confused with the person whose opinion we have covered here already).

Case C

Before describing this case, let me provide some background about myself.

As you probably already know, I discovered Godot in 2017. I didn't want to become an engine developer for the sake of it. I wanted to find a game engine to make games! I had a very clear use case, and I needed to find a perfect tool to implement my game ideas (I couldn't be more wrong picking Godot).

A particular game idea that inspired me came from **Worms: Armageddon** (W:A, in short). In fact, I started playing this game in 2012, and many other old-school players are still playing this game up to this day, despite the fact that this game was created in 1999. The reason

why people still play this game is two-fold: first, it has unique gameplay mechanics; second, there exist maintainers-volunteers who have access to W:A's source code granted by Team17, so the game (semi-officially) receives bug fixes, modernizations, and even new features. Even a sequel of Worms, namely World Party (WWP), which basically uses the same underlying W:A engine, didn't receive this level of maintenance as W:A currently does.

After a long break, I started to play some W:A games again in 2023. I've been chatting with various players on W:A's semi-official Discord server. Then, someone shared a link to a personal website of a W:A maintainer. Let's call this maintainer *Vladimir P* (abbreviated from his real name, which is available publicly), in the same style of the Worms mascot, *Boggy B*. By the way, you can unlock various W:A goodies by typing `boggysentme` in the main menu. 😉

Anyways, I got curious, so I proceeded to Vladimir's website. Guess what! I was dismayed to discover that Vladimir P wrote a blog post called "Modding for Godot"[6]. You can imagine my reaction, this was quite unexpected for me! Why on Earth would Vladimir P choose to touch Godot topic, especially when he's mostly interested in low-level stuff. I read his blog post, and decided to leave a comment (I'm a former maintainer of Godot, after all). I mostly agreed to what he said about Godot and its modding capabilities. The only concerns that I have expressed was the following:

- He presented Godot on par with Unity (by describing it as "not too unlike"), this is very misleading to new Godot users (ironically, especially when Godot's lead developer himself tells community that they shouldn't compare Godot to other technologies, as you've hopefully discovered in Authoritarianism section, reacting to Godot vs Unity meme). I'd like to emphasize that, while *comparing* technologies should be welcomed, *presenting* technology on the same level as other marginally related software create false ***expectations*** to say the least, and what Vladimir P did was to *present* Godot as if it's just an open-source version of Unity, but nothing can be farther from the truth. Many Godot followers who have at least some experience in game development would actually describe Godot on par with modern GameMaker (due to expressed level of simplicity, in-house built-in scripting language, etc). This kind wishful thinking, namely when Godot followers *want* Godot to become equivalent to Unity, is what creates this false expectation, when wishes substitute reality.

- His writing style makes a hidden ***assumption*** that it's sort of "easy" to create Godot mods for an ordinary user out there, especially when Godot users comprise mostly of hobbyists. For your information, the reason why there is currently little public information about modding Godot games (despite being open-sourced for nearly a decade now) is that Godot isn't really designed for that in the first place, so you'll have to deal with various workarounds that are highly likely to break with each (even minor) version of Godot. If you read the *Reverse engineering* section of Vladimir's blog post, it's

> obvious that he received a copy of the source code for the game that he decided to create mods for. It's usually quite a luxurious opportunity to have access to the source code for modding purpose, so the rest of the analysis is based on this fact. Not all developers may have access to the source code of a particular game, unless you're a creator, of course. But then, if you have access to the source code, it's certainly easy to make mods for everything, regardless of the engine used. That said, it's totally possible to mod Godot, of course. But this still remains an advanced topic.

I have assumed positive intentions, and I thought that Vladimir P simply got affected by Godot's propaganda to some extent, just like other Godot followers, despite the fact that Vladimir is quite an experienced developer, so all I did was to potentially clarify the above, not to mention that I also express my own opinion on this as a former maintainer of Godot.

Unfortunately, his reply was passive-aggressive:

> *Well done on your book I guess?*

You can read the entire comment section and judge for yourself. What's worth to note is his last intimidating reply (emphasis mine):

> *Well, I just looked at your "book" and realized it's an entire **hate** website criticizing Godot.*
>
> *Get **some help**, my dude. This **isn't healthy**.*
>
> *Since you're **nitpicking** at my wording and trying to draw out **contradictions** out of nothing instead of addressing my questions, I conclude that you're not here to participate constructively, so I am going to **delete further posts** from you.*
>
> *For readers: my experience with interacting with the Godot community has been **only pleasant** (this incident, if it even counts, notwithstanding), as well as second-hand impressions from participating in the ΔV community (which is made in Godot).*

As you see, the behavioral model is all the same, which describes a *Godot cultist*, see Glossary section. Mental health insinuations, labeling genuine criticism as hatred and frowning upon semantics (figuring out semantics is quite important, if we cannot agree upon a term, a discussion cannot be fruitful), and threats of blocking further replies (information control): this is *exactly* what describes a behavior of someone who's part of a Toxic Cult. I have not invented this! This was studied for decades by cult experts. As promised, he deleted my final post.

The following is the text of my post that Vladimir P immediately deleted once I decided to post my final words, which explains the rest (even criminals have a right for a final word, by the way):

I have answered your questions (there was only one pertaining to your post specifically, by the way). I have also said that I mostly agree to your assessment. At the same time, you ignored my own question. Note: just because you think that you asked a question doesn't mean that you asked a question in practice. Having clarified this, may I ask you: what kind of question I haven't answered?

I'm quite tolerant to your accusations, you know. You're not acting in good faith by telling me and others that what I say isn't healthy, and labeling my conclusions as hatred. For your information, I voluntarily contributed to Godot's development for years, and many people outside of Godot community generally agree to what I say.

I assumed good intentions, and I used to think that you're just a victim to Godot's propaganda, but according to your replies so far, your comments convinced me that you're a typical cultist now. If only you read the book, you wouldn't say this. Of course, I also fallen victim to Godot, just like others, that's why I wrote the book.

If you want to delete my reply, it's your choice. But these kind of threats (as in: "I'm going to delete your posts"), along with insinuations regarding mental health, is exactly what describes cultic behavior. If you want to reinforce this and prove to readers that you're a cultist, deleting posts is the concluding and definitive thing that you can do.

I invite you to ponder upon this.

For readers: Godot community is extremely welcoming, as confirmed by Vladimir's feedback, but this is the essence of toxic cult groups: extremely welcoming on the outside, yet abusive on the inside, and in cases when people (outsiders, in cult terms) criticize such a group.

I posted the above response on my Twitter as well. Some people reading this may definitely disagree to what I said, and this is expected, since they may share a similar authoritarian mentality, *especially* Godot followers. But here's another twist to the story above...

Days later after this incident, when I decided to play W:A again, I couldn't join any online game. At all. I wondered what happened. With the help of W:A community, we have investigated that, I was likely IP-banned from interactions from all worms-related services and websites that Vladimir P officially managed. So, not only Vladimir P deleted my post(s), but he also added the *entire range of IP addresses* of an ISP that I'm using in Ukraine to the blacklist. In practice, this means that even if my dynamic IP changes, I won't be able to play any online W:A game again, nor visit other worms-related websites such as Worms2D wiki.

So, this blockage will also affect other people on the same ISP network, who are not guilty of anything.

By the way, W:A community members said that Vladimir P has decreased his activity or even abandoned everything related to W:A, starting from 2021. This is at the time when Vladimir P might have discovered Godot, especially when he wrote that blog post about Godot in 2022. I have also presented his repository as an example of how tests could be organized and written for Godot in 2020. In either case, looks like W:A community lost Vladimir P here. 🙃

The morale of the story is this: choose your tools and community wisely. The toxicity of Godot has proven itself to affect not only Godot-related stuff, but also other communities where Godot has an impact, to the point of potentially destabilizing and destroying the entire community.

Ironically, my journey with Godot started with my passion for W:A, and I've found myself back into W:A... With a peculiar Godot twist. 😁

Case D

> *The more I read about this the more delusional/unstable/mentally unwell cyberreality seems. He's grasping for straws in every post, calling Godot donators fake, attacking dev's LinkedIn profile, accuses the devs of gaslighting because 4.0 dared to have bugs, he links a godot justice manifesto (?!) which reads like an insane person ramblings.*[7]

This Godot cultist talks about cybereality's post that we covered in Groupthink, also see Reviews and Testimonies. This particular cultist exhibits signs of being part of a Toxic Cult, which manifests as making insinuations about mental health.

Ironically, the cultist employs strawman arguments:

- "*calling Godot donators fake*": cybereality raised an issue that many donors on Godot's funding page[8] used randomly generated (hence cryptic, lacking clarity, and potentially misleading) names to remain anonymous. Now, Godot has opted out of this practice and started using a label such as "anonymous donors" explicitly, but the majority of those donors still remain anonymous, raising concerns whether those donations are real in the first place. The fact of employing this randomly generated names technique is questionable to begin with, as it contradicts the transparency and clarity principles declared by the Godot project.

- "attacking dev's LinkedIn profile": cybereality simply linked the LinkedIn profile of W4 Games to highlight the contrasting descriptions between W4 Games' mission statement on their official website and their LinkedIn page representing W4 Games. The W4 Games[9] official website does not contain words and phrases like "online gaming," "cloud platform," and "online multiplayer," whereas W4 Games' LinkedIn page does. Moreover, that LinkedIn page is not publicly available; one can only access it while being registered at LinkedIn.

- "*accuses the devs of gaslighting because 4.0 dared to have bugs*": cybereality accused mostly Juan, the lead developer of Godot, who consistently refuses to accept existence of bugs and/or limitations, presenting them as features or outright making the person feel like they allegedly don't "understand" the inner workings of the engine, despite contradictions. See for instance Waiting for Philosophy.

- "*he links a godot justice manifesto (?!) which reads like an insane person ramblings*": cybereality didn't link Justice Manifesto directly when raising his own experience, but only as a way to suggest that he's not the only one bringing up the issues with Godot. Additionally, this was presented as if cybereality and the author of this book are the same person, which is misleading. This resulted in misunderstandings from other users leading to false conclusions, but gladly, some of them eventually pointed out that:

    - "*When you puplically call someone delusional/unstable/mentally ill, at least do the bare minimum of research. You are mixing multiple people into one. The author of this justice manifesto is not cyberreality.*"
    - "*This site is owned and maintained by a different person, not Cybereality.*"

There are other notable mentions and references coming from this book at that Reddit thread, if you'd like to laugh! 😁

> Blue Robot Cult, "Outwardly welcoming, inwardly abusive"... <sarcasm>I am sorry, I haven't received my daily abuse. Can someone please make sure to rend my pound of flesh?! </sarcasm>
>
> *Is this dude serious? I laughed when I clicked the link. I was taken aback when I saw how serious this person is in his feelings against us (not just Juan and Remi). Cyberreality really does have a lot of animosity against those of us who actually like this tool. (S)he (I honestly don't know) seriously espouses that the Godot Community is a cult, "but it isn't a bad thing." Sorry, anytime something is labeled a cult, it is with a negative inflection. There's no getting around that. Also...all of you "blue robot" users are guilty of suffering from cognitive dissonance.*

> *While I agree that we are passionate about Godot, I would be willing to bet that 99.9% of users are stable-minded individuals who do not sacrifice small goats to our lord, blue robot; it is an open-source project after all. Feel free to bring your knives only if you want to.*

Please read Case B to understand that the author of this book doesn't hate members of the Godot community; he's only against Godot cultists (see Glossary), but even then, they are treated with irony and an understanding of their nature. The name "Blue Robot" only bears symbolic, humorous, and ironic meanings and definitely does not imply "sacrificing small goats to our lord, blue robot" (even though that's what the Godot community tends to portray in social media, again, as so-called "jokes" about themselves, see Cult Leader).

References

[1] Waiting for Blue Robot - Godot Forums thread.

[2] Q&A in 2017 - Godot Q&A website.

[3] Godot Contributors Chat, coffee-break channel - Case A

[4] Confirmation Bias - By ChangingMinds.org

[5] Appeal to authority - By ChangingMinds.org

[6] Modding for Godot - By Vladimir P.

[7] A positive take on the recent problems the Godot community has been facing, let's keep our heads up! - Godot, Reddit.

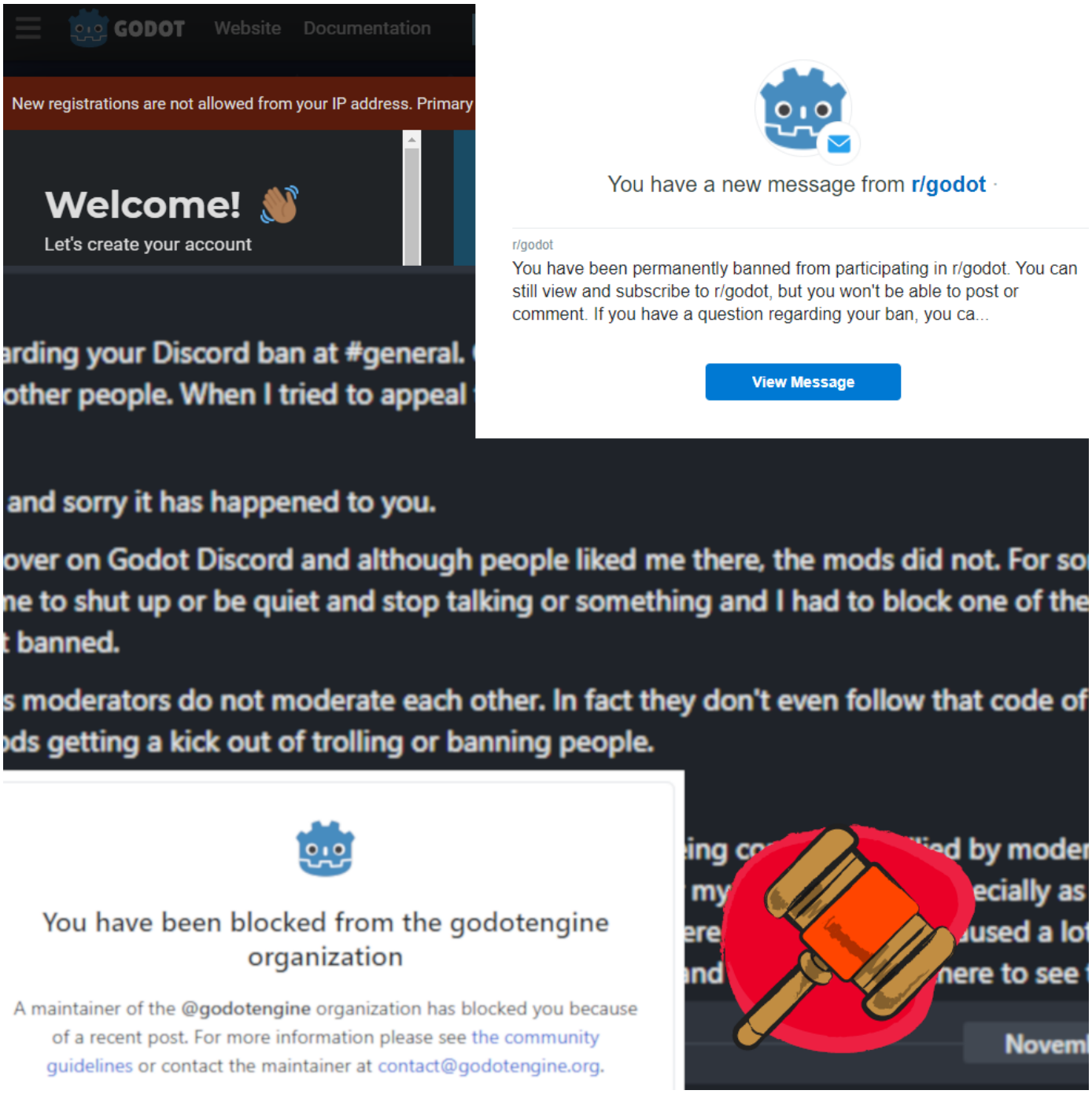
[8] Godot funding page - Godot Engine.

[9] W4 Games homepage - W4 Games.

Banned from Godot Engine?

First of all, you're not alone.

If you're a Godot user, there's a high chance that you've been banned from Godot's Discord server or blocked from similar Godot spaces. They ban people left and right, bedevil you for criticizing Godot, making comparisons, analogies, jokes etc.



If you're a Godot contributor, there's a chance that you've been "warned" in private. Quoting stuff is treated like a taboo in Godot. Maybe you got intimidated, harassed, bullied, and

eventually banned for attempting to address the inconsistencies and contradictions in Godot's decision-making game, perpetuated by its toxic leadership. It's totally okay to attempt to resolve these inconsistencies in public, but keep in mind, these mixed signals aren't just a product of Godot's ignorance. There's also a sneaky hidden agenda lurking behind Godot.

They can also shut you down for your legitimate concerns over Godot's identity politics. In Godot, these politics are promoted on the level of Godot's toxic leadership, to the point of violating their own Code of Conduct.

> Just got banned from #GodotEngine server, because of a phrase in my discord account!
>
> "The Godot engine is neutral and does not adhere to any specific flag, religion, race, or creed."
>
> but someone is making it not! pic.twitter.com/aUbf1LmgRg
>
> — 🇧🇷 nonunknown (@nonunknown3) November 28, 2023

If you wish to maintain your sanity and critical thinking abilities, there's little you can do to get unbanned. Hoping to be heard and understood by Godot's toxic leadership or those wielding the notorious ban hammer in the Godot community is pointless and counterproductive, as they lack self-awareness. Their purpose is to establish dominance and exert control, making you comply rather than fostering real collaboration. They are extremely good at guilt-tripping. Their gaslighting can mess up your head if you continue convincing them of anything, and they will eventually "win" the more you interact with them, due to their narcissistic nature.

Therefore, waiting for Godot is futile! In any case, asserting your right to an open discussion is crucial. Your best move? Share your testimony on social media or drop a line to the book's author to get your testimony in the mix. If you're itching to restore justice, simply sharing this book is a solid step in the right direction. Cheers!

> Read my free eBook: "Waiting for Blue Robot".https://t.co/7zCcpCdQ0g#OpenSource #FOSS #Governance #Ethics #Leadership #ToxicLeadership #Software #Developers #Programmer #GameDev #GameDeveloper #GameEngine #GodotEngine #Godot #TruthAboutGodot #WaitingForGodot #IGotOut pic.twitter.com/9aorlwPhPQ
>
> — Andrii Doroshenko 🇺🇦 (@Xrayez) October 16, 2022

Another effective method to resist petty tyrants is through the use of humor! If you haven't

already, you can watch the hilarious satirical documentary about Godot and share it with your friends to enjoy some laughs!

Also, don’t feel discouraged. To be banned from Godot is the highest reward that you can dream of in the realm of the game development industry, and not only!

Paradox of Tolerance

This section is mainly for those looking to restore justice by raising awareness about the wrongdoings of Godot's ***toxic leadership*** or simply those who are curious to know the intricacies behind what some people refer to as *cyberbullying* in the context of communities like Open Source.

The *paradox of tolerance* states that in a society overly tolerant of the intolerant, the intolerant can end up destroying the foundations of the tolerant society. What's ironic is that Godot's toxic leadership uses this paradox to rationalize their harmful actions, unreasonably banning users and contributors simply for exercising their right to independent thought and intellectual discourse.

Bullying usually conveys an intentional act. Godot's toxic leadership, intentionally or not, manipulates public perception by framing their bad actions and decisions as efforts to "protecting" the Godot community from so-called "bad actors". Within their palace of illusions, accountability is consistently deflected onto external factors. While they may concede the presence of certain issues, these are often attributed to communication lapses, and their focus is on "clarifying" purported misunderstandings within the Godot community. However, the real issue lies not in communication breakdowns but in the gaslighting tactics employed by Godot's toxic leadership when confronted with factual information that exposes their hypocrisy.

Because of these gaslighting tactics, individuals unfamiliar with these manipulation techniques or those who haven't recognized that they're being gaslit may find it challenging to pinpoint and expose Godot's actions as aggression because of Godot's *big lies* of aspiring to ensure so-called "safety" within the community. Rather, speaking up often leads to being labeled as the bully by Godot's toxic leadership, a *guilt-tripping* tactic they use to their advantage quite often. Challenging the claims of Godot's toxic leadership leads to a permanent ban.

Consider the power dynamics between those in authority (with the ban hammer) and those feeling powerless (afraid to speak up). While you might be searching for evidence of blatant bullying, a more precise term that encapsulates Godot's behavior is ***undue influence***. This is a tangible, legal term. In this context, "bullying" takes various forms in Godot. Undue influence tactics encompass deception, trickery, coercion, flattery, and even reward. Yes, a seemingly harmless pat on the back from Godot's toxic leadership can actually have serious consequences for the victim.

In Godot, this type of influence is more destructive than typical bullying, which may be fleeting in contrast. This is because Godot's toxic leadership reinforces behaviors that undermine and erode critical thinking and freedom of expression, putting excessive

emphasis on trust. Simply put, they have cultivated an environment where a significant number of contributors end up engaging in *bootlicking*. Undue influence is a complex and harmful process that can unfold over an extended period, potentially leading to emotional distress and mental breakdown among those affected.

These manipulative tactics allow Godot's toxic leadership to succeed in suppressing dissent and maintaining a façade of a welcoming community on the surface. Most Godot users are unaware of the dynamics within the inner circle of contributors, whereas contributors often fail to recognize abusive behavior due to the undue influence of Godot's toxic leadership.

Therefore, the focus should be placed on the undue influence aspect of Godot's toxic leadership. Unveiling instances of undue influence is done through the identification of inconsistencies, contradictions, and hypocrisy of Godot's toxic leadership and overzealous community of devoted fanatics. Abundant evidence supporting this assertion can be found in various sources, including this book and the testimonies it presents.

Recommendations for Exposing Blue Robot Wrongdoings

I recommend reading the following article and focusing on the undue influence aspect:

- The Cult: the Meaning of Undue Influence

Casually referring to issues covered above as *bullying* is fine, but it would fall short in capturing the root cause. It's more accurate to frame it as a problem of undue influence, even if you decide to use the term "bullying" for a broader audience. This is akin to labeling Godot as a scam, even if it's not precisely that. The crux of the matter is that Godot exhibits all classic characteristics of a cult, read Blue Robot Cult and Toxic Cult chapters.

However, outright stating that Godot is a cultic organization can be a challenge, as such claims can be overwhelming and unbelievable, despite facts. That's why the term "grift" can be employed to shed light on Godot's nature, considering their commercial agenda as covered in Community-Driven chapter and investigations such as Blue Robot and Red Hat.

It should be acknowledged that uncovering undue influence can be a complex task, but it's a challenge we must confront. Through engagements with Godot's leadership and community, the author of this book dedicated significant time to delve into the psychology and politics underpinning these social dynamics. In fact, the culmination of this exploration is the book that you currently read. It served as a tool to dispel lingering doubts and to systematically organize the insights gained from newfound knowledge.

People start to realize that they are *not* the issue. They build up resilience to *narcissistic*

manipulation tactics, and if there's any silver lining to Godot's toxic leadership, it's that it gave the author of this book an opportunity to grow and develop a natural immunity to an authoritarian mentality.

On a broader scale, for personal growth, when faced with *petty tyrants* in your life, all people have the capacity to gradually challenge them. It is imperative to draw attention to these key aspects. The ultimate solution to these issues lies in the shift from an authoritarian mentality to that of free people. This transformation is pivotal in avoiding cultic organizations akin to Godot in the future and in preventing the emergence of toxic communities altogether.

May the Force be with you. 😉

Have You Read the Book?

If you have already read the book, congratulations! You can skip this section. The following message is intended for those who haven't yet read the book and have reached out to the author with vague or general questions that are already addressed in the book, or perhaps they are unaware of its existence.

Hello, Waiter for Godot!

If you've recently come across my information about Godot, chances are you're reading this because you reached out to me with some inquiry. While I appreciate your interest, I noticed that you haven't posed any specific questions that I can respond to meaningfully. Please understand that I've received numerous similar requests, concerns, or reactions in the past, and I apologize if this message is not what you were expecting.

Before we proceed further, I would like to confirm whether you have read my book. The main reason I wrote it was to address the numerous questions that people often ask me. Insufficient or incomplete answers often lead to even more inquiries. However, if you would like a brief response, I can tell you that:

> ***The root cause of Godot's issues is the undue influence exerted by Godot's toxic leadership and the overly zealous behavior within the Godot community, as well as how these issues directly affect Godot's development process and, consequently, the final product.***

These complex issues cannot be adequately conveyed in just a few words without losing context or meaning. I genuinely want to engage in a discussion with you, but I'm only interested in factual discussions. Once you have read the book or if you have any specific and concrete questions, we can engage in a fruitful conversation. However, please bear in mind that many of the questions you might come up with are likely already addressed in the book. Drawing from my experience of conversing with individuals similar to yourself, I can assure you that most of your questions are likely answered there.

If you decide *not* to read the book, I urge you to ask yourself whether you want to understand the issue in the first place. It's possible that you may prefer *not* to know about the behind-the-scenes aspects of Godot's development process and governance, and that's perfectly fine. Some Godot followers have openly expressed this sentiment, emphasizing that they "can't unsee that truth." However, I firmly believe that the information compiled in

my book, which is based on my own experience and the testimonies of others, will be beneficial to you.

Thank you for your understanding. If you'd like to read the book, start from the beginning.