Jurnal Teknologi Sistem Informasi dan Aplikasi **ISSN: 2654-3788**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2654-4229**
Vol. 7, No. 2, April 2024 (892-896) **DOI: 10.32493/jtsi.v7i2.41654**

Comparing the Developmental Complexity of Different Game Engines by Creating the Same Game Using Two Different Engines

Normalisa[1], Pradana Atmadiputra[2], Jibran Wafi Prawiko[3]

Department of Computer Science, International University Liaison Indonesia, Associate 7th Intermark, BSD, 1530
e-mail: [1]normalisa@iuli.ac.id, [2]pradana.atmadiputra@iuli.ac.id, [3]jibran wafi prawiko@stud.iuli.ac.id

Submitted Date: April 16th, 2024 Reviewed Date: April 23rd, 2024
Revised Date: April 27th, 2024 Accepted Date: April 30th, 2024

Abstract

Game development is often considered to be a vague topic. With many beginner programmers interested in independent game development as an occupation, one must find out where should they start. Determining a first game engine could be a difficult choice for someone, and many beginner programmers hoped that their skills and early experiences could be utilized in the game development environment. Many comparisons do not detail what makes one game engine more difficult to learn than the other, and would only present vague terms such as because one engine can create a more complex game, yet it does not state how that would affect a game engine's learning curve. Research must be conducted to clear out this vagueness. Inside a game is basically a series of objects interacting with one another. Therefore, it should not be a problem when a developer switches between game engines, and yet these developers could have a faster development time when using a different engine. The result of this research is to determine how that difference is possible by comparing the developmental process of two different game engines (gamemaker and Godot) and determine which one is objectively better than the other in specific terms.

Keyword: Comparison; Game; Game engine; Gamemaker, Godot.

1 Introduction

Since the release of the Atari 2600, video games have become a mainstream media. It is a complex form of art that consists of every other type of entertainment and the development process that made them. Behind most game releases are designers, programmers, writers, artists, and especially the game engine itself (Goh, Al-Tabbaa, & Khan, 2023). Unreal engine is one of the earliest engines that became popular in 1998 due to the first game developed on it named being after that engine.

However, at the time of Unreal Engine's release, the video game industry was still dominated by large companies like Valve and Id Software. The idea of people outside these companies developing their own games was not prevalent until independent games became popular. Independent games are generally a form of video game release that was made by a small development team, often ranging from 10-20 people. Since most game engines are only available to the companies that made them such as the source engine that belonged to Valve, and the Frostbite engine that is owned by DICE, small independent teams must rely on proprietary software to develop their games, that was until game engines such as Unity and CryEngine became available to the public. Unity in particular, was one of the most accessible game engines during its early years, and was the main choice for many independent game developers in the 2010s. An engine's popularity can be observed by its marketing and the titles developed by it. Unity's great marketing itself can be seen when the engine was first announced at the Apple Worldwide Developers Conference as a game engine for the Mac.

The purpose of the following research is to compare the complexity between two different game engines based on the engine's popularity, the list of features and programming functions, complexity in managing objects and instances, typing, and the estimated development time. while also summarizing all the fundamentals of game

development at the same time. Therefore, The Author could recommend which game engine would be the most suitable for certain developers, as well as summarizing the similarities between the two engines in hopes of informing what kind of phrases and terminologies a developer should understand before proceeding to choose an engine and create their game (Sobota & Pietrikova, 2023).

The purpose of the following research is to compare the complexity between two different game engines based on its number of features, learning curve, programming knowledge, performance, and overall development time, while also summarizing all the fundamentals of game development at the same time. Therefore, the author could recommend which game engine would be the most suitable for certain developers, as well as summarizing the similarities between the four engines in hopes of informing what kind of phrases and terminologies a developer should understand before proceeding to choose an engine and create their game (Sobota & Pietrikova, 2023).

As time went on, Unity was no longer the only game engine choice in the market, as more and more different game engines kept being released. Today, opensource engines, such as Godot implied that anyone anywhere can develop their own game without spending any money, however difficult it might be. With competitions emerging left and right, people are starting to compare game engines with each other to see which engine might be the most suitable for them.

2 Analysis

Ludologists study games from different angles, such as their design, mechanics, storytelling, aesthetics, culture, and history. Game development could be considered as Ludology. They are also interested in different types of gamers, how they interact with games, and how games impact their behaviour, attitudes, and perception of the world (Barg, 2024).

3 Methodology

Shaun Spalding' theory is supported when observing the number of early games throughout history that were developed only by including what was mentioned (Shapiro, 2021). Some examples include score-based arcade games such as Pac-Man and Galaga, and also simple rogue-like games such as Vampire Survivors. These essential features became the criteria mentioned by The Author that needed to be satisfied before a comparison can be made in later chapters (Pierce, 2023).

Table 1 List Requirements

ID	Feature	Application Requirement
1	Player Movement	Utilize the vector from the game engine
		Create the horizontal speed and vertical speed variable for the player object
		Add and assign keyboard input the horizontal and vertical speed of the player object
2	Walls, Collisions, and Hitboxes	Add a wall object and place inside the room
		Stop player object movement when player object collides with a wall obejct
		Restrict vertical movement by decreasing the y axis with a gravity value
3	Enemies	Add enemies in the game with the same horizontal and vertical speed principals as the player
4	Player Attack	To edit the collision hitbox for objects
5	Health	The player object shall be able to detect collisions against an enemy object
		Make the player die after a collisions with the enemy has been detected
6	Player and Enemy Animations	Add animations for player and enemy movement
		Assign a sprite to an object
7	Camera	Create a camera that follows the player object
		To customize the camera's speed, size, and position in the room
8	Levels	To place and/or manage the objects in separate layers in the room
		Create multiple levels with an interconnected path
9	Menu Screen	Create a menu screen with options to start the game or quit the game

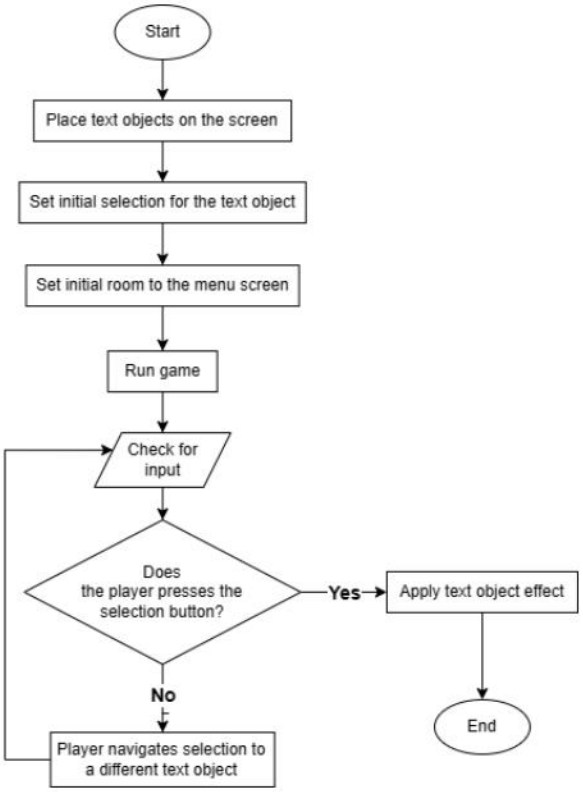

Figure 1. Menu screen flowchart

Menu screens act similarly to levels, since both are rooms in a game. Although this room should supposedly be what the player first witnesses when playing the game.

4 Result

The first category is the score of the titles made by the engine. This comparison is conducted by selecting five most popular 2D game titles made from each engine, and comparing its overall critical score and reception from metacritic, while also observing how many platforms the selected titles have been released in (Christopoulou & Xinogalos, 2021).

The data for this comparison is taken from the Steam and metacritic page of each title. Since Steam reception cannot be measured by numbers, it is only viable to measure the amount of people reviewing it rather than the game's critical score (Jonduke, 2020).

Most popular titles developed in Gamemaker according to the official site:

Figure 2. Gamemaker-made games ratings

Average metascore critic review: 86.3
Average metascore user review: 85.3
Total steam review: 240,742

Most popular titles developed in Godot according to the official site:

Jurnal Teknologi Sistem Informasi dan Aplikasi **ISSN: 2654-3788**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2654-4229**
Vol. 7, No. 2, April 2024 (892-896) **DOI: 10.32493/jtsi.v7i2.41654**

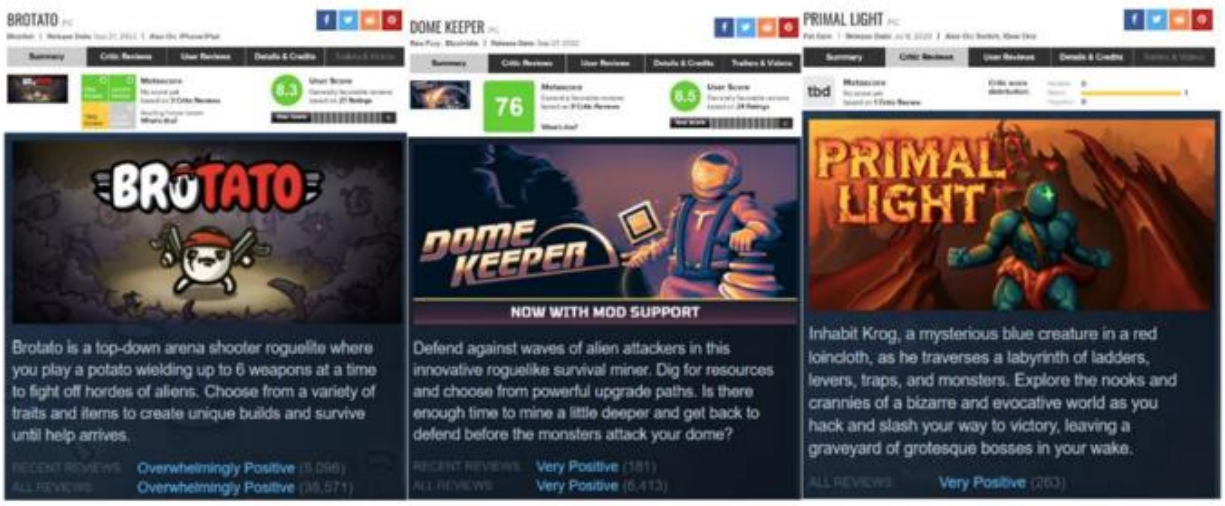

Figure 3. Godot-made games ratings

Average metascore critic review: 72.2
Average metascore user review: 84
Total steam review: 45,307

As seen from the image above, the titles developed by the Gamemaker engine don't just have a higher average user and critic score, but also have a higher total number of reviews. Which makes Gameamker a more popular engine than Godot in the gaming community. The importance of product reception by its user score can promote the makings of that product. This is observed by the amount of posts in forums that ask for which game engine is used to develop Undertale and Katana Zero, while there are no posts regarding the development of Brotato, Dome Keeper, or Primal Light.

Considering the result in the development timeline that presents Godot with a shorter learning time than Gamemaker. It shows that Godot's object oriented programming approach allows for more features to be added in the game engine which makes a faster development time, but not necessarily an easier one. Gamemaker's more straightforward approach that replicates visualbased sequences makes for a shorter learning time, but a longer development time. For example, sprites in Gamemaker need to be uploaded one by one, while Godot allows. to have sprites remain in a spritesheet. Both engines have dynamic programming languages. But Godot has a stronger programming typing, which is best used for large scale projects that focus on maintainability. The application made on this research is not considered to be a large project. It only covers the the basics and fundamentals of the game development process which technically will make development in Gamemaker faster than Godot. However, this statement only means that development in Godot will be faster if it was a larger game. However, there is a lack of focus in Godot because everything cannot be managed by code, which means it is a requirement for programmers to learn other skills such as animation, rendering, and advanced calculus in order to understand Godot as a whole. Whereas it is the opposite for Gamemaker. Every measurement speed size scale is all set inside the code. This is the reason why development in Gamemaker will take longer for larger projects.

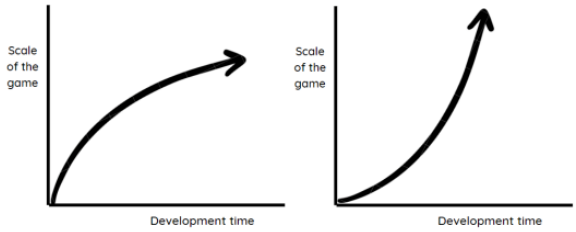

Figure 4. Gamemaker and Godot development time graph

This comparison can be summarized by the graphs above that represents the development time for Godot (represented by the right graph) and Gamemaker (represented by the left graph).

5 Conclusion

Based on the results received at the development process, here are the things that are the results concluded in this research. Recreating an already existing game is more difficult than creating an original game, There is a lack of focus in Godot because everything cannot be managed by code alone, which means it is a requirement for programmers to learn other skills such as animation, rendering, and advanced calculus in order to understand Godot as a whole. whereas Gamemaker is the opposite. Gamemaker is an engine that is dependent on code more than Godot, which resulted in a longer development time, because every attribute and measurements of an object needs to be set inside the script of an event. Mastering Godot will not decrease the amount of difficulty when a developer switches into Gamemaker for the first time. The same can be said for the latter. Since the properties of a node in a Godot engine can be edited by a variety of methods, game development using Godot is more preferrable when it is utilized by a large team that consists of animators, programmers, sound engineers, etc. who understood those methods. A Godot scene could only have one node with a script attach to it to determine its behaviour and mechanic, or it may only be utilized as a parent node that inherit other scenes, or it may consist of an animation-based node, a collision-based node, and a shader-based node at the same time. These situations indicates a need for larger teams that is divided to develop different mechanics of the game in Godot. On the other hand, Gamemaker is not a preferrable choice for a large team, being that the process of developing certain mechanics such as a healthbar, player movement, enemy behaviour, and wall collisions have the same principle of creating a new object, assigning a sprite to that object, adding code to the event of that object, and place that object in the room where it will be a part of the gameplay. This might made Gamemaker's lack of features to be its biggest strength, which makes early developers who are learning Gamemaker to only require to master fewer features of the engine compared to Godot that can be done by a smaller team, even if with the cost of a longer development time in Gamemaker.

References

Barg, M. (2024). *Ludo-Hermeneutics: The Interpretation of Digital Games Exemplified in the Puzzle Game Portal.* Stockholm: Stockholm University.

Christopoulou, E., & Xinogalos, S. (2021). Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *ResearchGate*, 21-36.

Goh, E., Al-Tabbaa, O., & Khan, Z. (2023). Unravelling the complexity of the Video Game Industry: An integrative framework and future research directions. *Elsevier, 12*(12), 1-18.

Jonduke. (2020). Analyzing Steam Reviews and Users Data. *Medium*, 1-8.

Pierce, M. (2023). The simple game that keeps on giving. *Medium*, 1-4.

Shapiro, L. (2021, June 25). Embodied Cognition . *Stanford Encyclopedia of Philosophy*, pp. 1-11.

Sobota, B., & Pietrikova, E. (2023). The Role of Game Engines in Game Development and Teaching. *intechopen*, 6-30.

Verbeke, W., Dejaeger, K., Martens, D., Hur, J., & Baesens, B. (2012). New Insights into Churn Prediction in the Telecommunication Sector: A Profit Driven Data Mining Approach. *European Journal of Operational Research, 218*(1), 211-229. doi:10.1016/j.ejor.2011.09.031

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Burlington: Morgan Kaufmann.

Yap, B. W., Rani, K. A., Rahman, H. A., Fong, S., Khairudin, Z., & Abdullah, N. N. (2014). An Application of Oversampling, Undersampling, Bagging and Boosting in Handling Imbalanced Datasets. *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013). 285*, pp. 13-22. Singapore: Springer. doi:10.1007/978-981-4585-18-7_2