Comparative Analysis of Unity and Godot for 2D Game Development

Alikhan Alybaev *

Posted Date: 27 November 2025

doi: 10.20944/preprints202511.1981.v1

Keywords: unity; Godot; 2D game development; performance evaluation; usability analysis; game engines

 doi:10.20944/preprints202511.1981.v1

Article

Comparative Analysis of Unity and Godot for 2D Game Development

Alikhan Alybaev

Department of Computer Science, Ala-Too International University, Bishkek, Kyrgyzstan; alikhan.alybaev@alatoo.edu.kg

Abstract

Choosing an appropriate game engine—especially for independent developers—has gotten more important as the video game sector has exploded in size. In relation to 2D game production, this study offers a relative comparison of Unity and Godot. To assess three important elements—development process, performance, and usability—a basic 2D platformer prototype was made in both engines using the same assets and design. While usability was evaluated using qualitative aspects like simplicity of installation, project organizing, performance metrics covered frame rate (FPS), memory usage, and construct size. Learning curve, community support, interface clarity, documentation quality, and the results clarify the advantages and disadvantages of every engine, therefore offering useful advice for developers—especially novices—in selecting a 2D game development platform.

Keywords: unity; Godot; 2D game development; performance evaluation; usability analysis; game engines

---

1. Introduction

Independent creators are increasingly important in deciding original and creative ideas as the video game industry has expanded dramatically recently. Choosing the right game engine is a very important decision for developers as it immediately impacts how well they may work as well as the game runs and the enjoyment the players have. Unity and Godot among the most frequently used programs for developing 2D games. Unity has always been regarded as among the most frequently used engines for indie and expert game development. It has a large asset store, a solid ecosystem, and is cross-platform compatible. Though it can be resource-intensive, beginners can have a steeper learning curve. Godot, on the other hand, is an open-source engine getting much attention recently because it is simple to Particularly for creating 2D games, use, adaptable, and light. While some argue it falls short of the complex features and ecosystem found in Unity, its dedicated 2D engine and scripting language, GDScript, give developers a straightforward and efficient toolset. The goal of this research is to contrast Godot and Unity in the context of 2D game development. By building a basic prototype in both Evaluating performance and engines assist developers, especially newcomers, select the best tool for their requirements.

2. Methods

The study used an experimental method to evaluate 2D game building Unity and Godot game engines. The investigation concentrated on three main spheres: development process, usability, and performance. Both engines were made using the same design and assets in order to give a simple 2D platformer prototype for comparative study.

The 2D platformer project helps to focus on the basic features an indie developer would have to incorporate in a video game. Chosen was C, which was used to develop the prototype together with Unity's built-in 2D physics engine. GDScript and the engine's native 2D system were used in Godot to execute the same feature. When evaluating the process, these were considered: Understand the debugging process and use the results.

 doi:10.20944/preprints202511.1981.v1

Performance was assessed on three parameters: frame rate (FPS), memory, and build size. Two prototypes were examined on a Windows 10 computer having an Intel Core i5 CPU and 8GB of RAM. Each engine's integrated profiling features recorded the frame rate (FPS) and memory usage during gameplay. After three repetitions, the mean score of each test was calculated to ensure repeatable results.

During the qualitative study, installation simplicity, design of the project, user interface intuitive and clarity, and documentation quality were all noted. Other considerations were asset management, community support, and the learning curve. Developer input from both the beginner and intermediate levels was gathered to help ensure a just review.

The data collected was evaluated to ascertain the strengths and shortcomings of the engines. The research results were summarized utilizing tables and charts in order that they would be understandable and practical for computer engineers.

3. Results

The analysis showed that Unity and Godot differed significantly in terms of development workflow, usability, and performance. Although the 2D platformer prototype was successfully implemented by both engines, there were notable differences in their behavior and developer experiences.

3.1. Performance Comparison

Table 1 enumerates the quantitative findings from both engines. Tests showed that while Unity produced slightly higher frame rates in the majority of gameplay scenes, Godot typically used less memory and produced smaller build sizes.

Table 1. Performance Metrics Comparison Between Unity and Godot.

Metric	Unity	Godot
Average FPS	118	110
Memory Usage (MB)	420	310
Build Size (MB)	95	42

A marginally higher frame rate was achieved by Unity's built-in rendering and physics optimization, but this benefit came at the expense of increased memory usage and build size. Godot is lighter for smaller projects thanks to its specialized 2D engine, which effectively managed memory and assets.

3.2. Usability and Learning Curve

Developers noted that Godot's scripting language (GDScript) and interface were easier for novices to learn because of their Python-like syntax and ease of use. Despite its strength, Unity necessitated more setup steps and a deeper understanding of C# scripting and component-based architecture.

Both engines had good documentation, but Unity's larger user base allowed for more examples and coverage. Nonetheless, Godot's documentation received recognition for its community-driven enhancements and clarity.

3.3. Development Process and Workflow

More configuration was required for the prototype's creation and debugging in Unity, especially with regard to project settings and physics behavior. Godot's lightweight build times and real-time scene editing improved the development flow and allowed for a quicker iteration cycle.

Prototyping was made easier by Unity's Asset Store, which provided developers with immediate access to a vast array of resources for asset management. Although Godot's open-source nature allowed for flexible customization, it still required the manual integration of external assets.

 doi:10.20944/preprints202511.1981.v1

3.4. Summary of Findings

All things considered, Godot stood out for its usability, learning curve, and resource efficiency, while Unity dominated in terms of performance and ecosystem maturity. Unity is still useful for projects requiring sophisticated systems and third-party integrations or for professional studios. Godot offers a useful and effective substitute for independent or instructional projects that emphasize 2D gameplay and quick development.

4. Conclusions

This comparison brings out the unique strengths and shortcomings of Unity and Godot in 2D game development. Unity exhibited more stable performance and offered a robust ecosystem with rich community backing, industry-grade tools, and a mature asset store. These factors place Unity more aptly suited for large-scale projects or commercial projects that need sophisticated features and long-term scalability.

Conversely, Godot was simpler and less difficult to employ, boasting with its usability, learning curve, and efficient resource management. Being open-source and having a specialized 2D design, it is particularly suitable for indie developers, educators, and small studios seeking quick prototyping or 2D-focused projects.

Ultimately, the choice between Unity and Godot is dependent on project goals and what the developer requires. Developers in need of high-performance, feature-full development will want to use Unity, and developers in need of simplicity, flexibility, and open-source-based development will want to use Godot. Additional research might extend this comparison to 3D performance, plugin communities, and mobile deployment efficiency to create an even more comprehensive understanding of both engines.

References

1. G. Andrade, G. Ramalho, A. S. Gomes, and V. Corruble, "Dynamic Game Difficulty Adjustment: An Integrated Approach," in *Proc. AIIDE*, 2006.
2. C. Lewis, J. Whitehead, and N. Wardrip-Fruin, "What Went Wrong: A Taxonomy of Video Game Bugs," in *Proc. FDG*, 2010.
3. Unity Technologies, "Unity Performance Optimization Guidelines," Unity Documentation, 2023.
4. P. Dickson, "Evaluating the Unity Game Engine for Teaching Game Development," *Journal of Computing Sciences in Colleges*, 2015.
5. A. Ferreira and R. Silva, "Analysis of Performance Issues in Unity 2D Games," *IEEE Latin America Transactions*, 2020.
6. Godot Engine Documentation, "Godot Rendering and Performance Guide," Godot Docs, 2023.
7. J. Martinez, "Comparing GDScript and C# Performance in Godot Engine," *Procedia Computer Science*, 2022.
8. R. Paredes, "An Evaluation of Godot Engine for 2D Game-Based Learning Systems," in *Proc. IEEE EDUCON*, 2023.
9. D. Benavides and C. Ruiz, "Comparative Study of Unity and Godot for 2D Game Development," *International Journal of Game Development*, 2023.
10. M. Hasan, "Performance Comparison of Popular Game Engines for Real-Time Rendering," in *ACM SIGGRAPH*, 2021.
11. M. Rusu, "Lightweight vs Complex Game Engines: A Comparative Study," *Computers & Graphics*, 2019.
12. S. Rogers, *Level Design: Theory and Practice*, 2nd ed., 2014.
13. A. Bayliss, "Usability Evaluation of Game Engines for Novice Developers," *IEEE Transactions on Games*, 2022.
14. T. Smith, "Open-Source Tools in Game Development: A Systematic Review," *SoftwareX*, 2020.