Extensibility Analysis of Game Engines for Building Simplified Game Development Interfaces

Joana Gabriela R. de Souza, Marcos Vinícius C. Pacheco, Raquel Oliveira Prates

[1]Computer Science Department – Federal University of Minas Gerais (UFMG)
Av. Antônio Carlos, 6627 – 31270-901 – Belo Horizonte – MG – Brazil

{joana.souza, rprates}@dcc.ufmg.br, vinipacheco08@gmail.com

Abstract. *Creating video games requires a multidisciplinary team and specialized tools for graphics and interaction mechanisms. While games are valuable educational tools, game engines are often too complex for non-programmers. Simplified design tools democratize game development for educators. This study aims to support Brazilian elementary school teachers by analyzing game engines for customization possibilities and developing a prototype authoring tool to create digital educational games quickly.*
Keywords *End-user Programming, Game engines, Human-Computer Interaction.*

1. Introduction

Creating video games is a complex process requiring a multidisciplinary team and specialized tools for content production. Developers must create engaging experiences while managing the technical demands of modern games, including sophisticated graphics, interaction mechanisms, and behavior definitions [Kanode e Haddad 2009, Gregory 2018]. They face the dual challenge of designing games that resonate with the target audience and handling the intricate technical aspects of digital game programming [Kanode e Haddad 2009].

In education, electronic games have become valuable teaching tools, but the complexity of game engines designed for professional developers hinders the creation of personalized educational games by educators [De Gloria et al. 2014]. Simplified game design tools aim to democratize this process, making it accessible to non-professionals [Gazis e Katsiri 2023, Barianos 2021, Chover et al. 2020], but it still is a challenge. This study, part of a project to aid Brazilian elementary school teachers, explores game engines to identify customization possibilities, aiming to develop a simplified authoring tool for teachers to create educational games. The analysis focuses on the flexibility and ease of extending game engines to support this goal.

2. Related Works

Several studies compare game engines, providing a basis for understanding their features and limitations, aiding in selection based on specific needs [Gazis e Katsiri 2023, Ullmann et al. 2022, Barczak e Woźniak 2019]. Game engines are tools that support game creation by simulating physical forces, lighting effects, and other necessary elements. They make game development easier, faster, and cheaper [Gazis e Katsiri 2023, Politowski et al. 2021]. Understanding these tools' characteristics helps identify the best options for various goals. This work aims not only to evaluate and select a game engine but also to simplify it so that non-professional developers can create their games. To this end, we highlighted works focusing on customization or extension of game engines.

Kupiainen's work [Kupiainen 2018] examines extending the Unity game engine through editor scripting, comparing Unity with Unreal Engine, and includes a practical project on creating a game with editor extensions. Barianos' research [Barianos 2021] proposes a framework for involving educators in content creation and customization of educational games, demonstrated through a pilot implementation. Chover et al. present a game engine for simplified 2D game development and tested with individuals having some programming knowledge [Chover et al. 2020].

Our study aims to create a simplified authoring tool for educational games by analyzing various game engines, focusing on implementing user-friendly constructs for teachers as game creators. Based on the literature, we did not find any work that compared game engines and game creation frameworks with a focus on their ability to serve as the basis for a simplified educational game authoring tool for educators.

3. Methodology

To identify the optimal platform for our project, we searched through game engines via internet searches and academic literature in October 2023. The goal was to evaluate engines that allow code or interface modifications or support plugin development. This evaluation aimed to inform the creation of a simplified authoring tool prototype for educators.

We filtered the engines based on language, cost, and storage requirements, favoring those with lower resource demands. As we aim to make the modified version available to elementary school teachers, we prioritized tools requiring less space and computer processing. At the end of this stage, we selected two tools to analyze in more detail. We then conducted an in-depth analysis of the selected game engines, understanding their specific aspects, how they work, and possible challenges that could arise when developing the prototype.

In previous works, we investigated how teachers interact with games and technology [Souza e Prates 2022] and conducted a workshop to observe their impressions using an authoring tool aimed at non-programmers creating games [Souza e Prates 2023]. As a result, these studies generated a set of constructs and requirements for an authoring tool for teachers to create educational games. Some of the constructs identified in those studies are: scenes, feedback elements, characters, barriers, questions, buttons, windows, and so forth. Based on these results, we defined a basic set of requirements necessary for the prototype.

Focusing on web games, we aimed to facilitate access for students in Brazilian computer labs and educators at home (to create games). The capacity to export to the web and other platforms was a key evaluation criterion. We conceived three development approaches: modifying an existing engine's interface, dividing the prototype into separate environments for development and gameplay, and creating a single application with dual internal environments. Evaluation criteria also included the engine's capacity to integrate with a database for login and game session creation via HTTP requests. Additionally, ease of use, including user interface creation, native interactive components, game previewing, and structural organization, were considered essential in selecting the game engine.

We then used this set as a guide to analyze each engine and its suitability to develop the intended prototype. Finally, we compared the final results of analyzing the two engines to assess their main differences and select one of them to continue our work.

4. Game Engine Analyses

Our initial search for available game engines that could be adapted for elementary school teachers to create educational games, took into consideration its language, cost, and storage space. From the initial eight candidates — Unity, Unreal, Godot, GDevelop, Game Maker, Construct, Cocos, and Phaser — we focused on free engines with a size under 100 MB to ensure accessibility and minimal resource consumption[1]. As a result, we selected Godot and Phaser[2] as the best candidates to develop the prototype. We analyzed Godot 4.1 and Phaser 3.60.0, considering their suitability for adaptation and ease of use for educators. The following subsections present our detailed analysis of these engines.

4.1. Godot

The Godot Engine is an open-source game development tool that supports the creation of 2D and 3D games. Its popularity among independent developers is attributed to its extensive documentation, active online community support, and MIT licensing, which allows unrestricted software use, modification, and distribution.

Godot is optimized for its native programming language, GDScript, which is object-oriented and features dynamic typing. Its signal system enables communication between objects within scenes, facilitating code compartmentalization. The integrated editor provides a comprehensive interface for game development, including scene editing, object manipulation, scene preview, and error detection in the code editor. Godot supports exporting games to various platforms, including Windows, macOS, Linux, Android, iOS, and HTML5. This versatility allows developers to target a wide range of devices and platforms[3].

4.2. Phaser

Phaser is an open-source framework for developing 2D web games. It simplifies game development by providing pre-built functionalities such as physics simulation, sprite manipulation, and user input handling. It is commonly used by developers who want to create interactive games for web applications. It can be integrated to coexist with other front-end development frameworks such as Vue[4] and Vuetify[5]. Phaser supports multiple rendering engines, including WebGL and Canvas, and utilizes HTML5 and JavaScript, with optional TypeScript support (Phaser documentation[6]).

Phaser lacks an Integrated Development Environment (IDE), necessitating an external development environment for code organization and typing, particularly when using TypeScript. Additionally, it relies on a local server to host game execution during

[1]See the summarization of information about tools in: https://bit.ly/45cU90s
[2]https://godotengine.org/ and https://phaser.io/ respectively
[3]https://docs.godotengine.org/en/4.1/
[4]https://vuejs.org/
[5]https://vuetifyjs.com/
[6]https://newdocs.phaser.io/docs/3.60.0

development. Phaser's ease of use, cross-platform support, and extensive documentation make it suitable for developing 2D web games. While it lacks an IDE, this can be addressed through the use of external development tools.

Despite being categorized as a framework, Phaser shares similarities with game engines, providing reusable components and software tools for game development [Politowski et al. 2021]. In this context, Phaser is considered a game engine for this research.

5. Results

Regarding our comparative analysis between the two tools to determine the best option for continuing the project, we identified essential elements for implementation in the final prototype and analyzed the implementation process for each engine. First, both engines supported connecting to a database via HTTP requests. Godot employs a specific node, 'HTTPRequest,' to handle HTTP requests, including downloading or sending files or web content. On the other hand, Phaser utilizes standard libraries such as Axios or Ajax for these requests.

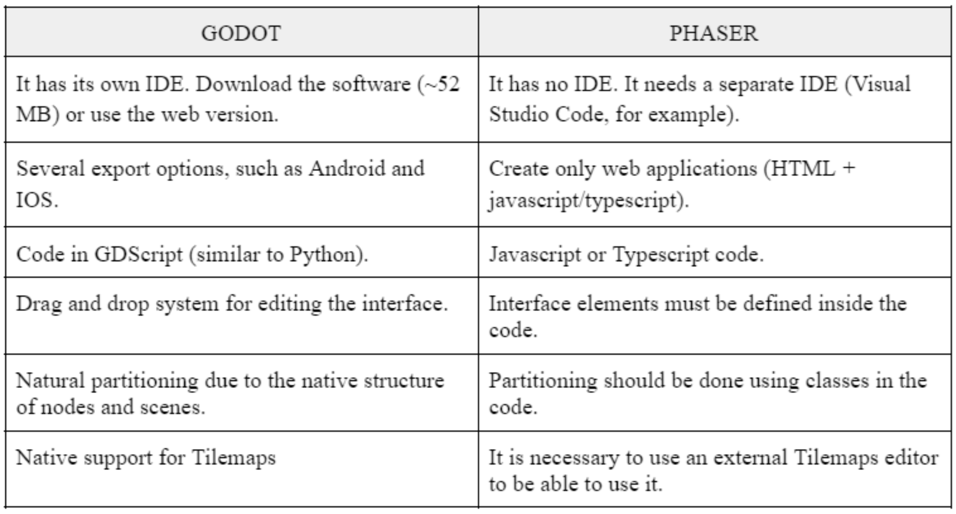
Regarding game export capabilities, both engines support web and mobile platforms. However, Godot offers an additional advantage by allowing game exports to Linux and Windows (executable). This feature is particularly beneficial given that these operating systems are prevalent in Brazilian public schools, enabling offline gameplay, which is crucial due to frequent internet unavailability in these schools [Souza et al. 2017].

When analyzing the ease of use of the platforms, we observed that Godot's integrated development environment offers numerous features to facilitate development. These include a visual area for connecting nodes, a scene preview window, a drag-and-drop interface, and control of project files. Phaser lacks a native IDE, requiring external software like Visual Studio or the deprecated Intel XDK for type checking, class control, and game export settings. Thus, Godot provides simpler integration between code development and tool usage than Phaser.

We also analyzed the difficulty of implementing essential elements in the final prototype. Most elements, such as interface buttons, physics-affected characters, platforms with collision, dialog interfaces, and timers, have similar implementations in both Godot and Phaser. However, Godot offers pre-implemented nodes for some elements, like progress bars or color selection inputs, facilitating development.

The engines' ability to provide different approaches to implementing the prototype was a critical aspect of our decision. The engines can create separate environments for development and execution, but only Godot allows interface customization using plugins. Implementing an application with two internal environments is easier with Godot due to its user-friendly interface manipulation and extensive native User Interface components. Table 1 summarize our comparison between the game engines.

From our comparative analysis, we concluded that both engines could implement the prototype, but Godot's own IDE, ease of use, flexibility, and additional native resources make it the more appropriate choice. This flexibility ensures greater freedom for potential changes in the prototype's scope.



GODOT	PHASER
It has its own IDE. Download the software (~52 MB) or use the web version.	It has no IDE. It needs a separate IDE (Visual Studio Code, for example).
Several export options, such as Android and IOS.	Create only web applications (HTML + javascript/typescript).
Code in GDScript (similar to Python).	Javascript or Typescript code.
Drag and drop system for editing the interface.	Interface elements must be defined inside the code.
Natural partitioning due to the native structure of nodes and scenes.	Partitioning should be done using classes in the code.
Native support for Tilemaps	It is necessary to use an external Tilemaps editor to be able to use it.

Figura 1. Comparison Godot x Phaser.

6. Final remarks

This research aimed to analyze game engines and frameworks suitable for developing a prototype educational game authoring tool for elementary school teachers. We compared tools based on financial costs, interface flexibility, computational requirements, and support for implementing constructs that allow teachers to create games without game design or programming knowledge.

We conducted an in-depth analysis of Godot and Phaser, focusing on features facilitating implementing our project's functionalities. Godot was selected as the most suitable tool due to its flexibility and ease of use. A limitation of our study is that we should have evaluated all available tools, and the analysis was conducted by one author but mitigated with weekly discussions with the others. This article presents a partial result of our detailed analysis. Future work would include evaluating more tools and conducting further tests to refine our prototype using Godot and Phaser to determine which offers greater ease and adaptability.

7. Acknowledges

Joana de Souza would like to thank CAPES for the partial support received for her research.

Referências

Barczak, A. M. e Woźniak, H. (2019). Comparative study on game engines. *Studia Informatica. Systems and Information Technology. Systemy i Technologie Informacyjne*, (1-2).

Barianos, K.-A. (2021). Thimel-content: an inclusive content creation, game customization and gameplay personalization tool.

Chover, M., Marín, C., Rebollo, C., e Remolar, I. (2020). A game engine designed to simplify 2d video game development. *Multimedia Tools and Applications*, 79(17):12307–12328.

De Gloria, A., Bellotti, F., e Berta, R. (2014). Serious games for education and training. *International Journal of Serious Games*, 1(1).

Gazis, A. e Katsiri, E. (2023). Serious games in digital gaming: A comprehensive review of applications, game engines and advancements. *arXiv preprint arXiv:2311.03384*.

Gregory, J. (2018). *Game engine architecture*. AK Peters/CRC Press.

Kanode, C. M. e Haddad, H. M. (2009). Software engineering challenges in game development. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 260–265. IEEE.

Kupiainen, H. (2018). Extending unity game engine through editor scripting.

Politowski, C., Petrillo, F., Montandon, J. E., Valente, M. T., e Guéhéneuc, Y.-G. (2021). Are game engines software frameworks? a three-perspective study. *Journal of Systems and Software*, 171:110846.

Souza, E. A., Garcia, L. G., Silva, J. C. N., Garcia, L. G., e Moreira, P. L. (2017). A review of the use of information technology in brazilian schools from 2010 to 2014. *International Journal of Information and Education Technology*, 7(4):284.

Souza, J. G. R. e Prates, R. O. (2022). Professores do ensino fundamental brasileiro: contexto social em que estão inseridos e a relação com jogos educacionais. In *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 846–855. SBC.

Souza, J. G. R. d. e Prates, R. O. (2023). Desafios para a construçao de jogos digitais por professores do ensino fundamental-relato de uma oficina. In *Anais do XXXI Workshop sobre Educação em Computação*, pages 167–177. SBC.

Ullmann, G. C., Politowski, C., Guéhéneuc, Y.-G., e Petrillo, F. (2022). Game engine comparative anatomy. In *International Conference on Entertainment Computing*, pages 103–111. Springer.