UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

MARCOS VINICIUS CALDEIRA PACHECO

Análise da Extensibilidade de Motores de Jogo para Construção de Interfaces de Desenvolvimento de Jogos.

Belo Horizonte
Dezembro 2023

UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

MARCOS VINICIUS CALDEIRA PACHECO

Análise da Extensibilidade de Motores de Jogo para Construção de Interfaces de Desenvolvimento de Jogos.

Projeto Orientado em Computação I apresentado ao Departamento de Ciências da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a Conclusão do Curso de Ciência da Computação.

Orientadora: Raquel Oliveira Prates.
Co Orientadora: Joana Gabriela Ribeiro de Souza.

Belo Horizonte
Dezembro 2023

Conteúdo

1 INTRODUÇÃO

O desenvolvimento de jogos eletrônicos é uma atividade complexa, que envolve múltiplas áreas de conhecimento, de lógica a artes visuais, e a utilização de ferramentas especializadas. Os desenvolvedores de jogos enfrentam o desafio não apenas de conceber jogos que se conectem com o público alvo, mas também de lidar com as complexas demandas técnicas inerentes à programação de jogos digitais [1].

Na esfera educacional, os jogos eletrônicos têm ganhado importância como ferramentas de ensino, servindo para envolver alunos de maneira mais imersiva. No entanto, a criação de jogos educacionais personalizados esbarra na complexidade dos motores de jogos (ou *engines* de jogos), que são softwares ou conjunto de ferramentas projetadas para auxiliar no desenvolvimento de jogos digitais. Geralmente essas *engines* são voltadas para profissionais da indústria de jogos e, portanto, inacessíveis para educadores e estudantes sem experiência em programação [2].

Neste cenário, surgem ferramentas de design de jogos simplificados, que buscam democratizar o processo de desenvolvimento, reduzindo a complexidade técnica e tornando a criação de jogos acessível para educadores que podem não possuir uma experiência técnica mais avançada. Esta pesquisa se concentra na necessidade de simplificar o processo de criação de jogos e visa explorar as inovações e tendências nas principais *engines* do mercado, com um foco no planejamento de um protótipo de criação de jogos que tenha uma interface simplificada e seja mais acessível a educadores do ensino fundamental brasileiro.

1.1 Objetivo geral

O objetivo deste trabalho é analisar motores de jogo (*game engines*) que possibilitam que sejam feitas modificações ou permitam a construção de *plugins* para o desenvolvimento de ferramentas que utilizem esses motores de jogo como base de funcionalidades. Neste trabalho analisaremos sistemas que estão em uso, observando o quanto elas permitem a adição de *plugins*, bibliotecas e a construção de interfaces que as utilizem como base para executar aplicações, com foco especial na desenvolvimento de interfaces de construção de jogos simplificadas.

A segunda etapa deste trabalho, o Projeto Orientado em Computação II (POC II), será conduzida com base nos resultados obtidos no POC I. O próximo passo será o desenvolvimento de um protótipo que implemente um modelo de ambiente que suporte o desenvolvimento de jogos educacionais por professores do ensino fundamental, que são considerados um público leigo para o desenvolvimento de jogos. A implementação deste

modelo pode facilitar a criação de jogos para esse público dado que não costumam ter conhecimentos de *game design* ou de programação.

1.2 Objetivos específicos

Projeto Orientado em Computação I (POC I):

1. Realizar uma busca por motores de jogos, sua popularidade perante a comunidade e possibilidade de implementar modificações e criar plugins.
2. Selecionar duas ferramentas que tenham documentação e comunidades *online* ativas e fazer uma avaliação mais detalhada considerando aspectos como nível de customização e tipo de plataforma (web, desktop).
3. Comparar os resultados da análise dos dois sistemas, avaliando as semelhanças e diferenças encontradas e definindo qual sistema é mais adequado para se realizar a segunda parte deste estudo.

Projeto Orientado em Computação II (POC II):

1. Implementar o modelo para o desenvolvimento de jogos por professores do ensino fundamental criando o protótipo baseado na ferramenta escolhida.
2. Realizar um teste com usuários do público alvo para identificar se o uso da ferramenta simplificada criada usando o modelo ajuda-os a criar jogos de forma facilitada.
3. Identificar possíveis melhorias nos sistemas a partir do feedback dos usuários, a fim de aperfeiçoar a comunicabilidade e a usabilidade dos mesmos.

2 REFERENCIAL TEÓRICO

A partir do contexto apresentado, um estudo realizado por Chover et al. [3] propõe um novo motor de jogo 2D com o objetivo de reduzir essa complexidade dos processos de desenvolvimento de *videogames*. A principal característica desse motor é a simplificação das especificações dos jogos, o que diminui a complexidade da arquitetura do motor e introduz um ambiente de edição muito mais fácil de usar para a criação de jogos. Isso é feito por meio da redução do conjunto de condições e ações que definem o comportamento dos objetos, eliminando a necessidade de lidar com estruturas de dados complexas. Testes mostraram que usuários com pouca experiência em programação, foram capazes de desenvolver jogos arcade utilizando a ferramenta, o que corrobora o conceito e a hipótese de sua facilidade de uso e demonstram o potencial de motores de jogo simplificados.

Dentre outras soluções já existentes, destaca-se o Zetcil [4], que visa simplificar o processo de desenvolvimento de jogos usando a Unity Game Engine. Ele foi concebido como um *framework* de mecânica de jogo que visa abordar as dificuldades encontradas por estudantes que buscam criar jogos utilizando a linguagem C# no ambiente da Unity. Este *framework* transforma comandos de programação baseados em texto em propriedades visuais, tornando a *engine* mais acessível para os desenvolvedores. Resultados dessa pesquisa demonstraram um aumento na confiança dos estudantes ao criar rapidamente protótipos de jogos, destacando a eficácia dos *frameworks* no contexto de desenvolvimento de jogos.

Outra pesquisa relevante investigou o uso do u-Adventure, um ambiente de desenvolvimento de jogos criado especialmente para jogos de aventura gráfica com histórias interativas “aponte e clique”. A ideia principal era tornar a criação de jogos mais fácil, aproveitando as vantagens da Unity, que é uma ferramenta mais profissional, tornando-a menos complicada e sem exigir que as pessoas saibam programar. Além disso, o u-Adventure inclui recursos voltados para a educação, como avaliação e análise de aprendizado, e foi testado por pessoas com diferentes níveis de conhecimento técnico. Os resultados mostraram que foi muito mais simples criar histórias para quem nunca tinha usado a ferramenta antes, e também obteve resultados positivos com pessoas com mais conhecimento técnico que o utilizaram em projetos mais complicados [5].

No entanto, é importante ressaltar que as soluções mencionadas anteriormente não atendem integralmente à necessidade de desenvolver um ambiente especialmente direcionado para que educadores brasileiros possam criar jogos educativos. Apesar da existência de projetos voltados a jogos educacionais para ensinar um tema específico, como exemplificado pelo trabalho de Dairel [6], que visa proporcionar um ambiente de gerenciamento de fases de jogos para o ensino de química, criar uma interface de criação de jogos educacionais sem uma temática específica constitui um desafio único e requer uma abordagem mais personalizada, que é o foco principal desta pesquisa.

3 METODOLOGIA

Para a execução do projeto, foram realizados os seguintes passos:

1. Revisão da literatura e busca de ferramentas: Foi levantado informações sobre outros trabalhos similares, e listados diferentes motores de jogos considerando tanto a popularidade, quanto a viabilidade de extensões e criação de *plugins*.

2. Filtragem dos sistemas: Foram selecionadas duas ferramentas consideradas mais aptas para o desenvolvimento de interfaces simplificadas de criação de jogos, considerando os objetivos principais do projeto.
3. Análise dos sistemas selecionados: Foi realizada uma análise mais aprofundada dos motores selecionados, buscando levantar aspectos específicos de seu funcionamento e encontrar eventuais desafios que podem surgir no desenvolvimento do protótipo.
4. Planejamento do desenvolvimento: Foi levantado um escopo parcial para o desenvolvimento do protótipo levando em consideração o que foi observado na análise anterior, e foi analisado quais seriam os passos necessários para o desenvolvimento de alguns aspectos do sistema utilizando os dois motores.
5. Comparação dos resultados: Os resultados finais obtidos nas análises dos dois sistemas foram confrontados para avaliar as suas principais diferenças, e foi definido qual é o sistema mais adequado para dar continuidade durante o desenvolvimento do protótipo final.

O foco do trabalho foi avaliar qual motor de jogo é o mais adequado visando sua utilização para o desenvolvimento de uma interface simplificada de desenvolvimento de jogos, considerando os objetivos finais do protótipo que será desenvolvido, e levantar informações que serão úteis para prosseguir a construção da interface durante o POC II.

4 DESENVOLVIMENTO

O principal objetivo deste trabalho consiste em realizar uma análise dos motores de jogo a serem empregados na POC II, visando iniciar o desenvolvimento de um protótipo de *game engine* simplificada direcionada a educadores. Com isso, o primeiro passo desse processo foi realizar um levantamento geral dos motores de jogos disponíveis, com o registro de suas características específicas, pontos positivos e negativos, para permitir a redução das opções a duas possibilidades, e facilitar assim a condução de uma análise mais aprofundada.

Após realizar um levantamento das *game engines* mais populares atualmente utilizadas, foram identificados os principais objetivos e focos dessas plataformas, a fim de realizar uma filtragem inicial. Foi observado que a maioria das *engines* criadas especificamente para educadores têm foco em auxiliar no ensino de lógica de programação para o público jovem. No entanto, dado que o propósito deste trabalho é proporcionar um ambiente que permita aos educadores desenvolver jogos educativos abordando temas escolares variados, além da lógica de programação, como história ou matemática, optou-se

por excluir *engines* direcionadas exclusivamente para educadores, como o GameMaker for Education[1].

Adicionalmente, *engines* voltadas para a construção de jogos bastante simples, como o Scratch[2], e aquelas com enfoque em gêneros específicos, como o Adventure Game Studio[3] e RPG Maker[4] (otimizados para jogos *point-and-click* e RPG, respectivamente), também foram excluídas. Esta decisão foi tomada por essas plataformas acabarem limitando ou dificultando a construção de jogos um pouco mais complexos ou de gêneros diferentes do seu objetivo principal.

Dentre as *engines* restantes, oito possíveis candidatas se destacaram. Elas estão listadas abaixo, e os principais critérios considerados para a seleção foram o custo financeiro do uso do software e o consumo de memória e demais recursos computacionais pelos arquivos da *engine*. Com base nesses parâmetros, elaborou-se a seguinte tabela:

Engine	**Linguagem**	**Custo**	**Site principal**	**Tamanho aproximado da *engine***
Unity	C#	Gratuito (versão de uso pessoal)	https://unity.com	~4 GB
Unreal	C++ Visual Scripting	Gratuito (até certo faturamento)	https://www.unrealengine.com	~20 GB
Godot	GDScript C#	Open source	https://godotengine.org	~60 MB
GDevelop	Visual Scripting	Open source	https://gdevelop-app.com	~100 MB
Game Maker	GML (Game Maker Language)	Pago	https://www.yoyogames.com/en/gamemaker	~200 MB
Construct	Visual Scripting Javascript	Pago	https://www.construct.net/	~100 MB
Cocos	C++ JavaScript	Open source	https://www.cocos.com/en/	~150 MB
Phaser	JavaScript	Open source	https://phaser.io/	~30 MB

[1] Acesso disponível em https://gamemaker.io/en/education
[2] Acesso disponível em https://scratch.mit.edu/
[3] Acesso disponível em https://www.adventuregamestudio.co.uk/
[4] Acesso disponível em https://www.rpgmakerweb.com/

Para viabilizar a construção do protótipo sem a necessidade de gastos financeiros, optou-se por descartar as *engines* pagas. Além disso, para garantir que o protótipo final fosse acessível a um amplo número de educadores, independentemente das capacidades de suas máquinas, optou-se por eliminar *engines* cujo tamanho ultrapassasse 100 MB. Apesar de o GDevelop ser um candidato que se encaixa nesses critérios, como seu principal método de desenvolvimento é através de programação visual, ele foi considerado desfavorável, uma vez que pode restringir a expressividade conforme o escopo desejado para a montagem do protótipo. A partir disso, duas *engines* foram selecionadas como as melhores candidatas a serem utilizadas para o desenvolvimento do protótipo: Godot e Phaser. As versões escolhidas para análise dessas *engines* foram Godot 4.1 e Phaser 3.60.0.

4.1 Análise do Godot

O Godot Engine é um motor de jogo de código aberto e gratuito, capaz de ser usado para desenvolvimento tanto de jogos 2D como 3D. Ela é bastante popular entre desenvolvedores independentes, e possui uma quantidade considerável de documentação e suporte em comunidades online. Essa *engine* foi desenvolvida por uma grande quantidade de desenvolvedores, e possui a licença MIT, o que permite usar seus arquivos sem restrição, incluindo a possibilidade de copiar, modificar e publicar esses arquivos, contanto que o aviso de copyright e uma cópia da licença permaneçam nas cópias do software.

O Godot possui suporte à linguagem C#, mas a engine é otimizada para funcionar com sua linguagem de programação nativa denominada GDScript. Ela é orientada a objetos, utiliza indentação para estruturação de código e adota tipagem dinâmica, e é bastante similar a outras linguagens populares e de fácil aprendizado, como Python. O GDScript também apresenta um sistema de sinais que possibilita a conexão e comunicação entre diferentes objetos dentro de uma cena, o que auxilia na compartimentalização de funcionalidades entre componentes.

Outro aspecto desse motor de jogo é o seu editor integrado, que proporciona uma interface dedicada para a edição de elementos do desenvolvimento de jogos. Isso inclui a organização de nós, movimentação de objetos no espaço 2D ou 3D, janela de pré-visualização das cenas, editor de código com detecção de erros, entre outras funcionalidades. Além disso, o Godot oferece suporte para exportação em diversas plataformas, como Windows, macOS, Linux, Android, iOS e HTML5. [7]

4.2 Análise do Phaser

O Phaser é um *framework* de código aberto e gratuito, especializado no desenvolvimento de jogos 2D para o ambiente web. Ele é comumente utilizado por desenvolvedores que desejam criar jogos interativos para aplicações web, e pode ser integrado para coexistir com outros *frameworks* de desenvolvimento front-end como Vue[5] e Vuetify[6]. O principal objetivo do Phaser é simplificar o desenvolvimento de jogos, fornecendo funcionalidades prontas, tais como simulação de física, manipulação de *sprites* e leitura de entrada do usuário. Além disso, o *framework* dispõe de uma variedade de tutoriais e documentação em seu site principal.

Outra característica do Phaser é o suporte a diversos mecanismos de renderização, como WebGL e Canvas, proporcionando abordagens variadas para a implementação de jogos em páginas web. Seu desenvolvimento envolve HTML5 e JavaScript, mas também possui suporte para tipagem com TypeScript. Além disso, também existe a possibilidade de utilizar ferramentas adicionais para poder exportar jogos criados no Phaser para plataformas de dispositivos móveis como iOS e Android.

O Phaser não possui uma interface de edição própria, sendo necessário recorrer a um ambiente de desenvolvimento externo para facilitar aspectos como organização de código e tipagem, especialmente no caso da utilização de TypeScript. Adicionalmente, depende de um serviço de execução de servidor local para hospedar a execução do jogo durante o processo de desenvolvimento. [8]

Como concluiu Politowski et al. [9] em sua pesquisa, apesar de motores de jogos terem diferenças em comparação com *frameworks* de código aberto, eles não exigem um tratamento especial. Por esse motivo, apesar de ser categorizado como um *framework* para a criação de jogos, neste trabalho, o Phaser será referenciado como uma *engine* ou um motor de jogo.

4.3 Escopo do protótipo

Para que a análise comparativa entre o Godot e o Phaser fosse feita, foi necessário estabelecer um escopo para o desenvolvimento do protótipo que será elaborado no POC II, a fim de que se tenha mais clareza sobre quais são os principais critérios de avaliação, permitindo uma melhor análise sobre qual é a opção mais promissora a dar continuidade ao projeto. É importante observar que, se necessário, os aspectos mencionados a seguir podem passar por leves ajustes durante o desenvolvimento do protótipo.

---

[5] Acesso disponível em https://vuejs.org/
[6] Acesso disponível em https://vuetifyjs.com/

Primeiramente decidiu-se que o foco do desenvolvimento da *engine* será em jogos web, com o objetivo de que alunos no sistema educacional brasileiro consigam utilizar os laboratórios de informática para poder jogar os jogos, e para que o educador consiga ter um acesso fácil e rápido à *engine*. A possibilidade de exportação dos jogos para outras plataformas, com arquivos executáveis ou aplicativos de dispositivos móveis, é desejada mas não é obrigatória. Com isso, um dos critérios a ser utilizado para avaliação dos motores de jogo é a sua capacidade de exportação para web e outras plataformas.

A partir disso, foram concebidas três abordagens distintas para o desenvolvimento do protótipo: efetuar modificações na própria interface de uma *engine* existente visando simplificá-la; dividir o protótipo em dois ambientes distintos, um destinado ao desenvolvimento do jogo e outro à execução dos jogos criados; ou utilizar a própria *engine* para criar uma aplicação única com dois ambientes internos, um para o desenvolvimento e outro para execução dos jogos. Um dos critérios fundamentais para a avaliação dos motores de jogos seria, portanto, a capacidade e facilidade de implementação dessas possíveis abordagens para o desenvolvimento do protótipo.

Outro aspecto considerado é a necessidade de integrar a *engine* com um banco de dados, para suportar uma estrutura de login para o educador, com a criação de sessões de jogos para liberação de acesso aos alunos. Nesse contexto, a capacidade da *engine* de realizar HTTP *requests* foi considerada como critério de avaliação, dado que é um dos métodos mais simples e comuns para acessar bancos de dados.

Por fim, considerou-se a facilidade de utilização da plataforma e de implementação dos componentes necessários para a criação de jogos educacionais dentro da *engine*. Alguns aspectos como a criação de interfaces do usuário, a quantidade de componentes interativos que existem nativamente, a facilidade de pré-visualização dos jogos sendo desenvolvidos, e organização estrutural nativa das *engines* foram todos levados em consideração para análise e comparação entre os motores de jogo.

5 RESULTADOS

Após a definição do escopo básico do protótipo e dos critérios de avaliação, realizou-se uma análise comparativa entre as duas *engines* para determinar qual seria a melhor opção para dar continuidade ao projeto. Além disso, foram levantados alguns elementos básicos que se planeja implementar no protótipo final, a fim de analisar como seria o processo de implementação utilizando cada um dos motores.

Primeiramente, em relação à conexão com um banco de dados por meio de HTTP *requests*, ambas *engines* se mostraram capazes de realizar chamadas e interpretar as respostas corretamente. No Godot, essa funcionalidade é suportada por um nó específico, o '*HTTPRequest'*, projetado para efetuar solicitações HTTP, como baixar ou enviar arquivos ou conteúdo da web via HTTP. No caso do Phaser, a utilização de bibliotecas padrão como Axios ou Ajax oferece opções simples e diretas para efetuar essas requisições.

Quanto à capacidade de exportação, ambas as *engines* oferecem suporte para exportação web e dispositivos móveis Android e iOS. No entanto, a capacidade adicional do Godot de exportar para Linux e executáveis Windows foi considerada uma vantagem em relação ao Phaser nesse aspecto.

Ao analisar a facilidade de utilização das plataformas, observou-se que a IDE integrada do Godot é notavelmente útil e possui inúmeros recursos para facilitar o processo de desenvolvimento. Isso inclui uma área visual para conexão de sinais entre diferentes nós, uma janela de pré-visualização da cena em desenvolvimento, uma interface de arrastar e soltar objetos nas cenas, além de controle de pastas e arquivos do projeto, entre outras funcionalidades. No caso do Phaser, por não possuir uma IDE nativa, é necessário recorrer a outros softwares, como o Visual Studio ou Intel XDK, que são opções recomendadas na própria documentação do Phaser, para oferecer ferramentas aos desenvolvedores, como conferência de tipagem, controle de classes e configurações de exportação dos jogos. Dessa forma, embora não seja um impedimento, o Godot proporciona uma integração nativa e consideravelmente mais simplificada entre o desenvolvimento do código e a utilização da ferramenta do que o Phaser.

Outro aspecto que foi objeto de análise foi a dificuldade de implementação de alguns elementos básicos que estão planejados para existir no protótipo final. A maioria dos elementos, como botões da interface, personagens afetados por física, plataformas com colisão, interface de diálogos e cronômetros para monitoramento de tempo, apresentam uma implementação bastante semelhante tanto no Godot quanto no Phaser, seja ela facilitada ou de dificuldade igualmente elevada. Apesar disso, alguns poucos elementos, como barras de progresso ou input de seleção de cores, já possuem nós com implementações prontas no Godot Engine, o que pode facilitar o desenvolvimento do protótipo.

Por fim, um dos aspectos mais importantes que teve grande peso na tomada de decisão, é a capacidade das *engines* de proporcionar diferentes abordagens para a implementação do protótipo. Embora ambos os motores ofereçam a possibilidade de criação de dois ambientes separados, um para desenvolvimento dos jogos e outro para execução,

apenas o Godot permite a edição da interface da *engine* por meio de *plugins* para customização e simplificação da IDE. Adicionalmente, a implementação de uma única aplicação com dois ambientes internos é consideravelmente mais facilitada pelo Godot, devido à facilidade de manipulação e visualização das interfaces, bem como à vasta quantidade de componentes de *User Interface* (UI) nativos que a *engine* oferece.

6 CONSIDERAÇÕES FINAIS

A partir da análise comparativa realizada entre os dois motores de jogo, concluiu-se que, embora ambas as opções sejam completamente válidas de serem usadas, cada uma com seus prós e contras, optar pelo desenvolvimento do protótipo utilizando o Godot Engine seria a escolha mais apropriada. Isso se deve não apenas à quantidade de recursos adicionais que ele oferece nativamente em comparação com o Phaser, mas também à flexibilidade proporcionada pelos diversos caminhos de implementação que o Godot permite. Essa característica garante um maior grau de liberdade no caso de mudanças no escopo final do protótipo.

A partir dessa conclusão, foi desenvolvida uma prova de conceito utilizando o Godot e o Firebase, uma plataforma de serviços do Google que inclui banco de dados e hospedagem de sites. O objetivo era validar a viabilidade da utilização da *engine* selecionada e testar uma estrutura de banco de dados não-relacional primitiva para sustentar a criação de sessões de jogos pelo educador e o acesso às sessões pelos alunos. Com isso, não apenas a escolha do Godot como *engine* foi validada, mas a prova de conceito demonstrou que a utilização do Firebase pode ser uma escolha vantajosa para o desenvolvimento do protótipo final, pois oferece serviços de fácil integração, como login com contas do Google, que serão úteis considerando a grande adoção pelas pessoas.

Pretende-se, portanto, utilizar as informações levantadas no POC I como base de referência durante o prosseguimento do trabalho no POC II. A expectativa é que essa análise seja fundamental para a inicialização da construção das interfaces do sistema, na estruturação do arquivo de detalhamento dos jogos e na implementação de uma integração mais robusta com um banco de dados. Ao final do POC II, espera-se ter um protótipo funcional de uma interface de desenvolvimento de jogos simplificada, usando o motor escolhido, acessível a um público sem conhecimentos aprofundados em linguagens de programação. Além disso, pretende-se levantar pontos de melhoria futuros e uma análise final de como se decorreu o processo de desenvolvimento do protótipo.

7 REFERÊNCIAS

[1] GREGORY, Jason. Game engine architecture. crc Press, 2018.

[2] DE GLORIA, Alessandro; BELLOTTI, Francesco; BERTA, Riccardo. Serious Games for education and training. International Journal of Serious Games, v. 1, n. 1, 2014.

[3] CHOVER, Miguel et al. A game engine designed to simplify 2D video game development. Multimedia Tools and Applications, v. 79, p. 12307-12328, 2020..

[4] ROEDAVAN, Rikman et al. Zetcil: Game Mechanic Framework for Unity Game Engine. IJAIT (International Journal of Applied Information Technology), p. 96-105, 2019.

[5] PÉREZ-COLADO, Víctor Manuel et al. UAdventure: Simplifying narrative serious games development. In: 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT). IEEE, 2019. p. 119-123.

[6] DAIREL, João Gabriel de Matos et al. Uma proposta para gerenciamento de fases de jogos educacionais desenvolvidos com Phaser.js para não desenvolvedores no contexto de Química. 2021.

[7] **Godot Docs – 4.1 branch.** Disponível em: <https://docs.godotengine.org/en/4.1/>. Acesso em: 20 nov. 2023.

[8] **Phaser 3 API Documentation (beta) - Index.** Disponível em: <https://newdocs.phaser.io/docs/3.60.0>. Acesso em: 20 nov. 2023.

[9] POLITOWSKI, Cristiano et al. Are game engines software frameworks? A three-perspective study. Journal of Systems and Software, v. 171, p. 110846, 2021.