CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC (RES_020/2016 – 2024_2)	
( ) PRÉ-PROJETO (X) PROJETO	ANO/SEMESTRE: 2024/2

LOTUS, UM MOTOR DE JOGOS 2D FOCADO NO DESENVOLVIMENTO DE JOGOS DIGITAIS EM RUST

Alexandre Zeni

Prof. Dalton Solano dos Reis - Orientador

1 INTRODUÇÃO

O termo "motor de jogos" surgiu em meados de 1990, fazendo referência a um dos jogos em primeira pessoa mais populares até os dias de hoje, o Doom, lançado em 1993, desenvolvido pela Id Software. No projeto, fez-se claro o gerenciamento dos componentes utilizados para potencializar o ciclo de desenvolvimento, tendo por exemplo: um sistema separado para identificação de colisões, um sistema para renderização tridimensional e um sistema para gerenciamento de áudio (Gregory, 2018), tal organização é considerada uma espécie de revolução na forma como jogos são criados.

Nos últimos anos, o desenvolvimento de jogos usando linguagens não usuais, como Rust vem crescendo de forma exponencial, com o surgimento de bibliotecas e motores de jogos de código aberto à comunidade. Segundo Özgövde e Kurt (2023), Bevy é um motor de jogos lançado em 2020, com uma grande comunidade de contribuidores que realizam a criação de novas funcionalidades e a manutenção das que já existem. É focado no desenvolvimento tanto 2D quanto 3D, contudo não possuí uma interface gráfica ao usuário, como são encontrados nos motores de jogos mais famosos.

Embora sua primeira versão oficial tenha sido liberada no ano de 2015, Rust já se encontra entre as 15 linguagens de programação mais populares do mundo, de acordo com o *2024 Stack Overflow Developer Survey*, realizado pelo Stack Overflow e a sua comunidade de usuários. Sua popularidade se dá pelo fato de ser uma linguagem de programação com sintaxe moderna, extremamente veloz, sendo compilada diretamente em C e *memory safe*, permitindo ao desenvolvedor, juntamente com o auxílio do seu analisador de código nativo, o *borrow checker*, manualmente alocar e desalocar a memória de suas variáveis utilizadas ao longo do código fonte (Klabnik; Nichols, 2023). Com isso em vista, também há de se pontuar os benefícios aos desenvolvedores que criarão jogos fazendo uso do motor desenvolvido.

Segundo McMillian (2021, p.1, tradução nossa), "[...] um dos benefícios do desenvolvimento de jogos, é que há a oportunidade do desenvolvedor vestir múltiplos chapéus ao mesmo tempo [...]". Ou seja, ficar responsável por vários papéis durante o ciclo de vida do projeto, envolvendo arte, sons, design e o desenvolvimento propriamente dito do código fonte. Proporciona um desafio multidisciplinar que impulsiona além do pensamento lógico, a criatividade, o desenvolvimento *user first*, que em suma seria ter sempre em mente o usuário final do produto, entre outros.

Diante disso, este trabalho visa a pesquisa e o desenvolvimento de um motor de jogos 2D em Rust, que utiliza padrões de desenvolvimento e arquitetura modernos, como o Entity-Component-System (ECS). Visando facilitar o desenvolvimento de jogos bidimensionais e proporcionar o aprendizado da programação em uma tecnologia moderna e não usual. Também há de se realizar o estudo referente à comparação do projeto desenvolvido com outros motores já existentes no mercado, sendo eles de código aberto ou não, e em outras linguagens de programação.

1.1 OBJETIVOS

O objetivo deste trabalho é o desenvolvimento de um motor de jogos, com foco em jogos digitais bidimensionais fazendo uso da linguagem de programação Rust.

Os objetivos específicos são:

a) estudo sobre o desenvolvimento de motores de jogos;
b) disponibilizar para uso na criação de jogos digitais;
c) avaliar o uso da linguagem de programação Rust no desenvolvimento de jogos digitais.

2 TRABALHOS CORRELATOS

Nesta seção serão apresentados os trabalhos que se correlacionam com os objetivos deste projeto. Na subseção 2.1 é apresentada a revisão sistemática do processo de pesquisa para encontrar os trabalhos correlatos que foram escolhidos (Quadro 1), explanando o assunto do trabalho, as palavras chaves que foram utilizadas no protocolo de busca e as fontes bibliográficas relacionadas. Por fim é exposto uma justificativa para a escolha dos três trabalhos finais, para realmente serem utilizados como os correlatos.

Já na subseção 2.2, é realizada a síntese dos trabalhos correlatos escolhidos, explanando as suas referências, objetivos, principais funcionalidades, ferramentas de desenvolvimento e seus resultados e conclusões. Também são especificadas as principais características destes trabalhos, juntamente com um breve resumo sobre cada um.

2.1 REVISÃO SISTEMÁTICA

Para realização da busca pelos trabalhos correlatos, foi utilizada a plataforma Google Scholar. Os termos de pesquisa utilizados foram: Rust, 2D, 3D, motores de jogos, desenvolvimento de jogos, arquitetura e Entity-Component-System (ECS). A busca utilizando tais palavras-chaves retornou inúmeros registros, sendo limitado à artigos publicados a partir do ano de 2019.

Para realizar a escolha dos correlatos, foram analisados quais títulos e introduções faziam referência ao desenvolvimento de motores de jogos ou a utilização da linguagem de programação Rust dentro da área de desenvolvimento de jogos. As obras que envolviam o desenvolvimento de motores de jogos para construção de jogos tridimensionais ou de objetivo demasiadamente fora do proposto deste trabalho, não foram selecionados durante a busca.

Quadro 1 - Síntese dos trabalhos correlatos selecionados

Assunto	Filtro	Referência
Exploração de técnicas atuais de programação de jogos e geração procedural de ambientes através do desenvolvimento de um protótipo de jogo em Rust	Rust AND 2D AND Desenvolvimento de Jogos	Minini (2020)
Developing a Game Engine in C# Programming Language	Desenvolvimento de Jogos AND Motores de Jogos AND 2D	Sršen e Orehovački (2021)
Open-Source Game Engine & Framework for 2D Game Development	Desenvolvimento de Jogos AND Motores de Jogos AND 2D	Campos, Morales e Núñez (2022)
Coding a 2D Game Engine for ESP32: A Showcase of Design Patterns	Desenvolvimento de Jogos AND 2D AND Motores de Jogos AND Arquitetura	Gianadda (2024)
Game Engine Architecture, Third Edition	Desenvolvimento de Jogos AND Motores de Jogos AND Arquitetura AND 2D AND 3D	Gregory (2018)

Fonte: elaborado pelo autor.

2.2 SÍNTESE DOS TRABALHOS CORRELATOS

Nesta subseção são apresentados trabalhos que se correlacionam em objetivos e características com o estudo proposto. O primeiro trabalho (Quadro 2) é um protótipo de jogo digital bidimensional do gênero *roguelike* desenvolvido em Rust (Minini, 2020). O segundo trabalho (Quadro 3) é um motor de jogos denominado ProGameEn, para jogos 2D e 3D, desenvolvido em C# (Sršen; Orehovački, 2021) e o terceiro trabalho (Quadro 4) também é um motor de jogos, porém desenvolvido em C++ e com foco no desenvolvimento de jogos bidimensionais de forma eficiente e descomplicada (Campos; Morales; Núñez, 2022).

Quadro 2 - Exploração de técnicas atuais de programação de jogos e geração procedural de ambientes através do desenvolvimento de um protótipo de jogo em Rust

Referência	Minini (2020)
Objetivos	Estudo e comparação entre os algoritmos de geração procedural de ambientes através do desenvolvimento de um jogo bidimensional em Rust, do genêro *roguelike*. Juntamente com a exploração de técnicas modernas de desenvolvimento de jogos
Principais funcionalidades	*Pipelines* híbridos, Geração procedural de ambientes, design *data-driven* e utilização da programação Entity-Component-System (ECS)
Ferramentas de desenvolvimento	Rust, *specs*, *bracket-lib*, *Rusty Object Notation* (RON)
Resultados e conclusões	Para jogos do gênero *roguelike*, é mais proveitoso a combinação de mais de um algoritmo de geração procedural de ambientes, juntamente com o uso das mais novas técnicas de programação, design *data-driven* e arquitetura e*ntity-component-system*

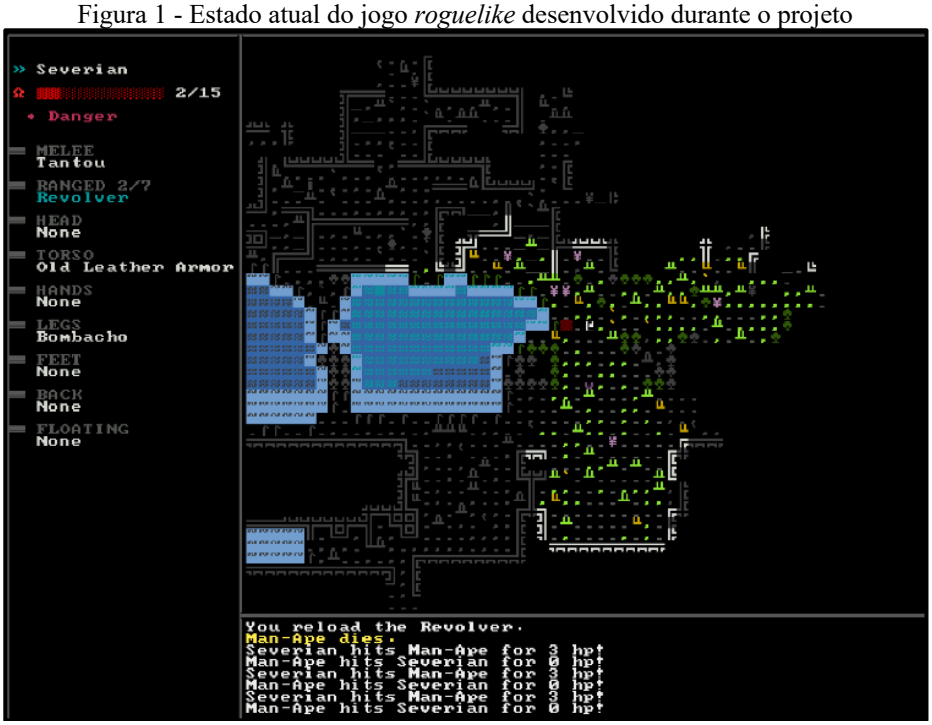
Fonte: elaborado pelo autor.

Minini (2020) descreve o seu interesse acerca da geração procedural de ambientes em jogos, juntamente com as técnicas mais modernas de desenvolvimento, o design *data-driven* e a arquitetura Entity-Component-System (ECS), em detrimento do paradigma convencional da programação orientada à objetos. Partindo deste princípio, decidiu realizar o desenvolvimento de um protótipo de jogo bidimensional em Rust, graficamente simples, do gênero *roguelike*, utilizando *pipelines* híbridos com diferentes algoritmos de geração procedural de *dungeons* (como são chamados os mapas a serem explorados pelos jogadores) para fins de comparação fim a fim, tendo como objeto de estudo principal o algoritmo WaveFunctionCollapse.

Como ferramentas durante o processo, fez uso da *specs*, uma biblioteca com foco no auxílio da implementação da arquitetura ECS, que se tratando de uma técnica relativamente nova, não há ainda um grande suporte nativo nas linguagens de programação. Também utilizou a biblioteca *bracket-lib*, que seria especificamente para o desenvolvimento de jogos com gráficos relativamente simples, facilitando o processo do desenvolvedor. Por fim, para realizar a implementação de um design *data-driven* em seu protótipo, que se aplica a ideia de arquivos *raw*, que seriam basicamente arquivos reais que armazenam dados que podem ser manualmente alterados e lidos durante a compilação do software, utilizou o tipo de arquivo *Rusty Object Notation* (RON), sendo um substituto para o famoso *JavaScript Object Notation* (JSON), para conseguir armazenar e posteriormente ler de forma facilitada os seus componentes escritos em Rust.

Durante o seu desenvolvimento, Minini (2020) utilizou uma gama de cinco algoritmos de geração procedural de terrenos, sendo eles o Passeio Aleatório, Autômatos Celulares, BSP Dungeon, Digger e o seu objeto principal de estudo, o WaveFunctionCollapse. Minini (2020) também traz em forma de *pipelines* híbridos, a comparação entre o uso de cada um destes. Tendo enfim a geração do jogo (Figura 1) que tem um ambiente dinâmico e altamente personalizado.

Figura 1 - Estado atual do jogo *roguelike* desenvolvido durante o projeto



Fonte: Minini (2020).

Em conclusão, Minini (2020) entende que o uso específico de apenas um algoritmo de geração procedural de ambientes pode não ser a melhor opção, sendo melhor realizar a junção deles, para um melhor resultado. Também pontua que o uso da arquitetura ECS se faz relativamente complexo e até desnecessariamente genérico em se tratando de poucas entidades concorrentes, sendo melhor realizar o uso do paradigma orientado à objetos nestes casos.

Quadro 3 - Developing a Game Engine in C# Programming Language

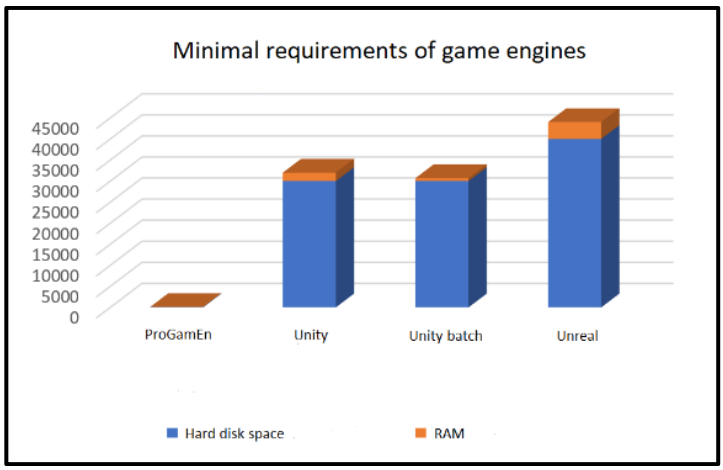
Referência	Sršen e Orehovački (2021)
Objetivos	Criar um motor de jogos para auxiliar no desenvolvimento de jogos digitais 2D e 3D, de forma descomplicada, usando a linguagem de programação C#. Realizar a comparação da performance do projeto realizado com outros motores de jogos disponíveis no mercado
Principais funcionalidades	Possibilidade de desenvolvimento de jogos 2D e 3D (renderização, cenas, animações e jogabilidade), acessibilidade facilitada, performance acima dos demais motores de jogos apresentados em comparação
Ferramentas de desenvolvimento	C#, *.NET*, *MonoGame*
Resultados e conclusões	Foi concluído que o motor de jogos proposto é mais performático e mais portável para computadores de baixo nível de hardware em comparação com outros motores de jogos disponíveis no mercado. Também foi concluído que o projeto proposto é uma ótima forma de introduzir o desenvolvimento de jogos para desenvolvedores interessados

Fonte: elaborado pelo autor.

Em seu trabalho, Sršen e Orehovački (2021) exploram a ideia de que os motores de jogos que estão hoje disponíveis no mercado são de difícil usabilidade e entendimento, por conta de serem pouco abertos em relação à sua arquitetura e componentes internos, deixando o desenvolvedor muitas vezes de certa forma, no escuro. Partindo

desse pressuposto apresentado, propõe a criação de um novo motor de jogos para desenvolvimento bidimensional e tridimensional, de código aberto e de fácil entendimento com relação ao seu código fonte e funcionamento geral, denominado ProGameEn, trazendo um protótipo de jogo em 2D e um em 3D, juntamente com comparações entre o projeto desenvolvido e os motores disponíveis no mercado, sendo possível observar a diferença de recursos utilizados com o motor de jogos desenvolvido (Figura 2).

Figura 2 - Comparação entre motores de jogos famosos no mercado em relação ao motor de jogos desenvolvido no trabalho correlacionado



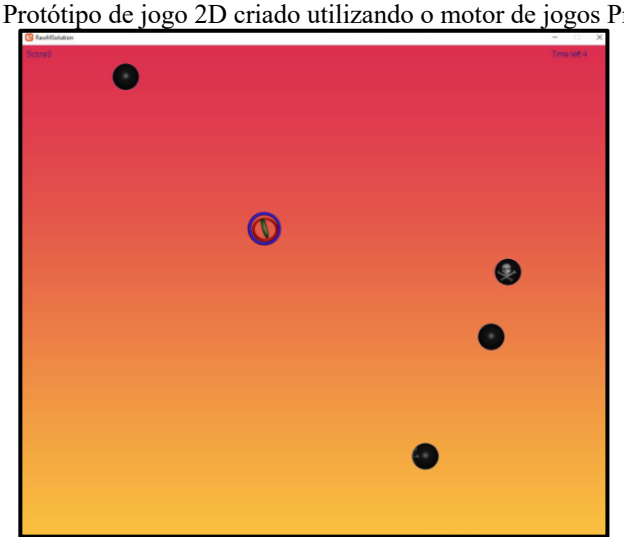
Fonte: Sršen e Orehovački (2021).

Segundo Sršen e Orehovački (2021), a linguagem de programação C# foi escolhida para este trabalho, por conta de sua fácil usabilidade em comparação com outras tecnologias comumente utilizadas, como o C++, que com alocação manual de memória, mesmo trazendo mais performance, acabava tendo códigos de complexidade mais elevada, enquanto o C# acaba sendo mais simples, porém menos performático por conta do uso do *garbage collector* nativamente. Como facilitador utilizou a biblioteca MonoGame, uma biblioteca .NET com o propósito de abstrair as partes mais complexas e verbosas do desenvolvimento de jogos, trazendo soluções para desenvolvimento 2D e 3D, com a habilidade de detectar mouse, teclado, joystick ou *touch screen*. Também possui funções otimizadas relacionadas à física e matemática, além de outras funcionalidades uteis no processo do desenvolvedor.

Em seu desenvolvimento, Sršen e Orehovački (2021) optaram por seguir o paradigma da Programação Orientada à Objetos (POO), seguindo o princípio básico de que, deve existir a consistência no design de componentes. Tendo como objeto base a classe abstrata `Component`, que define as funcionalidades básicas que os demais componentes devem herdar e aplicar. Como as funções `Enabled` e `Visible`, que definem se o componente está habilitado e visível em determinado momento. Também possuí a função `Update`, que fica encarregada de rodar funcionalidades lógicas que precisam acontecer várias vezes por segundo.

Como protótipo de jogo bidimensional, Sršen e Orehovački (2021) desenvolveram o jogo de nome “Collect All Balls” (Figura 3), no qual o jogador precisa coletar o máximo de pontos possíveis. O que ocorre através da colisão do objeto principal com os outros objetos em tela, sendo que a cada colisão, um som é emitido para evidenciar a pontuação. Pode ser jogado com mouse, teclado ou controle.

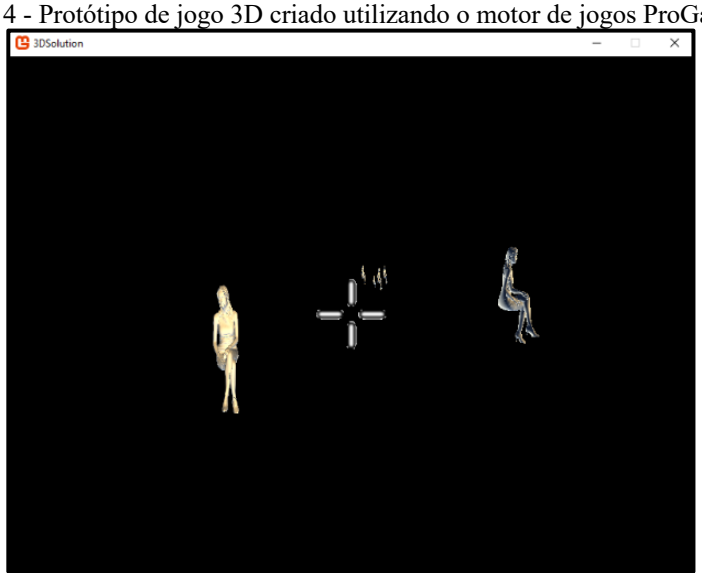
Figura 3 - Protótipo de jogo 2D criado utilizando o motor de jogos ProGameEn



Fonte: Sršen e Orehovački (2021).

Como protótipo de jogo tridimensional, foi desenvolvido o jogo denominado de “Guess the Queen” (Figura 4), no qual o jogador deve mirar e realizar disparos, utilizando o mouse, em objetos que aparecem aleatoriamente à sua frente, antes que o personagem denominado de rainha seja tocado. A cada nova fase, mais rapidamente esses objetos se movimentam e mais difícil se torna acertá-los.

Figura 4 - Protótipo de jogo 3D criado utilizando o motor de jogos ProGameEn



Fonte: Sršen e Orehovački (2021).

Em conclusão, Sršen e Orehovački (2021) afirmam que o projeto desenvolvido, ainda que inacabado, é uma ótima opção para desenvolvedores buscando aprender e aprimorar suas habilidades em desenvolvimento de jogos tanto 2D quanto 3D, sendo um software leve e de fácil utilização. Contudo é com certeza mais limitado se posto em comparação direta com os outros motores de jogos disponíveis no mercado, como o Unity ou o Unreal. Por mais que estes motores exijam mais recursos, oferecem interface gráfica integrada e diversas outras ferramentas para desenvolvimento de projetos complexos.

Quadro 4 - Open-Source Game Engine & Framework for 2D Game Development

Referência	Campos, Morales e Núñez (2022)
Objetivos	Desenvolvimento de um motor de jogos em C++, focado em 2D, utilizando arquitetura ECS e visando proporcionar uma experiencia facilitada ao desenvolvedor. Também procurando ser um projeto escalável, para 2.5D ou 3D.
Principais funcionalidades	Possibilidade de desenvolvimento de jogos 2D (renderização, cenas, animações e jogabilidade), interface gráfica para uso do usuário Graphical User Interface (GUI) e acessibilidade facilitada.
Ferramentas de desenvolvimento	C++, OpenGL, GLFW, Dear ImGUI
Resultados e conclusões	O projeto que visava facilitar o desenvolvimento de jogos em 2D, comparado aos motores de jogos pré-existentes, alcançou seu objetivo. O resultado foi explanado através de uma pesquisa entre os participantes que testaram o software.

Fonte: elaborado pelo autor.

De acordo com Campos, Morales e Núñez (2022), os motores de jogos disponíveis hoje no mercado, em sua maioria, têm enfoque principal no desenvolvimento de aplicações tridimensionais, deixando a cargo do desenvolvedor interessado, desenvolver jogos em 2D ou em 2.5D com um esforço maior e com menos suporte. Com isso em vista, os autores deste trabalho propuseram o estudo e desenvolvimento de um novo motor de jogos focado estritamente em aplicações bidimensionais, de código aberto e de alta escalabilidade, podendo receber novas funcionalidades desenvolvidas pelos usuários.

Foi escolhido o uso da linguagem de programação C++, para uma melhor performance, por conta de sua alocação manual de memória, entre outros benefícios. Utilizaram também o OpenGL, que tem como foco o desenvolvimento de aplicativos gráficos, juntamente com o GLFW, que é basicamente uma biblioteca para facilitar o uso dos contextos gráficos OpenGL. Por fim, usaram a Dear ImGui, uma biblioteca para desenvolvimento de interfaces gráficas para uso do usuário, as chamadas GUI (Figura 5).

Figura 5 - Interface gráfica criada para uso em tarefas no desenvolvimento de jogos digitais 2D, dentro do motor de jogos proposto

Fonte: Campos, Morales e Núñez (2022).

Em se tratando da estrutura de projeto, Campos, Morales e Núñez (2022), definem em quatro pontos, sendo *(i)* o *framework* de jogos, componente que fornece módulos abstratos base, contendo funcionalidades relacionadas à gráficos, matemática, física, áudio que podem ser implementadas conforme necessidade; *(ii)* o motor de jogos, componente que irá permitir o desenvolvimento de jogos bidimensionais, utilizando os módulos abstratos previamente fornecidos; *(iii)* o gerador de jogos, componente que irá compilar o código fonte com todos os *assets* utilizados durante desenvolvimento, para a plataforma esperada, podendo ser Windows, Linux ou MacOS; e *(iv)* a interface com o usuário, componente que irá permitir ao desenvolvedor que interaja com todas as funcionalidades internas do projeto.

Por fim, foi disponibilizado o motor desenvolvido para uso por doze desenvolvedores, sendo eles, dois experientes e dez não experientes. Eles tiveram que desenvolver um pequeno protótipo de jogo 2D, durante dois dias de uso. Foram verificados bons resultados ao fim do uso, de acordo com Campos, Morales e Núñez (2022), 85% dos candidatos repararam em um aumento significativo na velocidade do desenvolvimento e 75% dos mesmos afirmaram que usariam novamente o motor em um novo projeto.

Os autores concluíram que o projeto proposto e desenvolvido é uma solução ótima para desenvolvimento de jogos, possuindo limitações evidentes com relação à motores de jogos comerciais, porém sendo de código aberto, é altamente escalável a depender do crescimento de usuários e de sua comunidade.

3 PROPOSTA DO MOTOR DE JOGOS

Nesta seção é apresentada a proposta do motor de jogos referente a este estudo, contendo sua devida justificativa e metodologia. Na subseção 3.1 é apresentada a justificativa referente aos motivos que levaram a consideração de realizar tal projeto. Através da análise das características em comum entre os trabalhos correlatos escolhidos (Quadro 5), trazendo também as relações entre a metodologia escolhida e o sucesso deste estudo, juntamente de uma explanação sobre quais as principais contribuições tal objeto de estudo pode proporcionar por fim.

Já na subseção 3.2, encontra-se a explanação sobre a metodologia escolhida para este trabalho, trazendo detalhadamente, cada passo a ser seguido e os principais motivos que levaram à tal escolha. Métodos e ferramentas que serão utilizados também serão explanados em cada etapa da descrição da metodologia.

3.1 JUSTIFICATIVA

Os trabalhos correlatos escolhidos para este projeto se relacionam com 7 características em comum, descritas no Quadro 5. Primeiramente é trazido se o trabalho realizado utilizou uma linguagem de programação em seu desenvolvimento, que faz uso da alocação manual da memória, nos quais Minini (2020) e Campos, Morales e Núñez (2022), utilizam as tecnologias Rust e C++, respectivamente, enquanto Sršen e Orehovački (2021) fazem a utilização do C#. Também foi explanado quanto ao foco específico em desenvolvimento bidimensional, sendo que apenas em Sršen e Orehovački (2021) é relatado possuir uma abordagem híbrida.

Todos estes projetos são de código aberto, podendo providenciar um grande aprendizado no desenvolvimento de jogos e na usabilidade de tais linguagens de programação, através da análise do código fonte já existente, juntamente com possíveis sugestões de mudanças e novas funcionalidades que podem ser

implementadas, para pessoas já experientes ou não. Por fim, todos podem possuir interfaces gráficas ao usuário, mas apenas Campos, Morales e Núñez (2022) se destacam com sua abordagem no desenvolvimento 2D.

Quadro 5 - Comparativo dos trabalhos correlatos

Características \ Trabalhos Correlatos	Minini (2020)	Sršen e Orehovački (2021)	Campos, Morales e Núñez (2022)
Faz uso de uma linguagem de programação com alocamento manual de memória	Sim	Não	Sim
Focado em desenvolvimento 2D	Sim	Não	Sim
É um motor de jogos	Não	Sim	Sim
Possui código aberto	Sim	Sim	Sim
Pode providenciar aprendizado no desenvolvimento de jogos	Sim	Sim	Sim
Desenvolvimento híbrido (2D e 3D)	Não	Sim	Não
Possuí interface gráfica (GUI)	Não	Não	Sim

Fonte: elaborado pelo autor.

Este trabalho traz contribuições para a área da computação, sendo estas de cunho teórico e tecnológico. Com a prototipação do motor de jogos em Rust, desenvolvedores poderão aprender uma nova linguagem de programação que está crescendo cada vez mais no mercado de tecnologia, elevando assim as possibilidades de criação de novos projetos usufruindo das funcionalidades da linguagem. Além deste ponto, há de se destacar a contribuição teórica com relação ao desenvolvimento de motores de jogos propriamente ditos. Os desenvolvedores também terão a oportunidade de adentrar na área de desenvolvimento de jogos digitais, que para muitos ainda pode ser uma novidade, pois é um mercado mais enxuto e menos acessível em comparação com outras áreas.

3.2 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) estudo sobre motores de jogos em Rust: realizar o estudo aprofundado do código fonte e funcionamento de outros motores de jogos já existentes, escritos em Rust;
b) escolha das bibliotecas: escolher quais bibliotecas serão utilizadas para o desenvolvimento das funções de renderização gráfica, matemática, entre outros;
c) especificação do motor de jogos: realizar a especificação quanto à arquitetura e paradigma a serem utilizados, juntamente com diagramas UML relacionados;
d) criação do motor de jogos: desenvolver o motor de jogos proposto com foco em jogos 2D;
e) criação dos testes do código fonte: desenvolver testes relacionados a funcionalidade do motor proposto, buscando trazer maior confiabilidade ao projeto;
f) disponibilização do projeto como uma biblioteca pública: disponibilizar o projeto como uma biblioteca a ser utilizada em projetos Rust;
g) implementação de dois protótipos de jogos bidimensionais: desenvolver dois protótipos de jogos 2D simples, para evidenciar o uso do projeto desenvolvido e trazer exemplos aos futuros usuários;
h) criação da documentação do projeto: criar a documentação completa do projeto, para facilitar o seu uso futuro;
i) testes com usuários: realizar testes com usuários que já possuem alguma experiencia prévia com programação e evidenciar se obtiveram resultados positivos utilizando o projeto desenvolvido.

4 REVISÃO BIBLIOGRÁFICA

Nesta seção são abordados dois temas e suas referências correlacionadas que foram utilizadas para formular o pensamento base do estudo a ser realizado: prototipação e desenvolvimento de um motor de jogos bidimensional em Rust. Na subseção 4.1 é abordado a prototipação, bidimensional ou tridimensional e de suas diferentes arquiteturas e possíveis especificações. Já a subseção 4.2 explana sobre o tema específico do desenvolvimento de jogos digitais tendo Rust como a ferramenta principal de escrita do código fonte. Demonstra os prós e os contras de usar essa linguagem, em detrimento das linguagens mais tradicionais, como C ou C++.

4.1 MOTORES DE JOGOS

De acordo com Gregory (2018), no passado, o desenvolvimento de jogos se dava especificamente para cada cenário específico. Cada jogo era criado de forma não controlada e sem um padrão específico. Foi com a chegada de jogos como Doom, lançado em 1993 e o Quake, lançado em 1996, que a criação e utilização de motores de jogos começou a ser algo comum neste meio.

Também explanado em seu livro, Gregory (2018) destaca que, motores de jogos podem ser considerados uma coletânea de vários sistemas a serem utilizados pelo desenvolvedor no ciclo de vida de um jogo, contendo sistemas relacionados à física, matemática, detecção e tratamento de colisões, renderização 2D e/ou 3D, incluindo renderização de *sprites*, entre outras funcionalidades.

Motores de jogos, principalmente para desenvolvimento bidimensional, podem ser altamente portáveis entre plataformas e até embarcados, como é o caso do trabalho apresentado por Giannada (2024), que propôs a prototipação de um motor de jogos 2D embarcado em um ESP32, em C++, fazendo uso das melhores práticas em desenvolvimento de software como um todo.

4.2 DESENVOLVIMENTO DE JOGOS DIGITAIS COM RUST

De acordo com Özgövde e Kurt (2023), em seu trabalho sobre o desenvolvimento de motores de jogos em Rust, Bevy é um motor de jogos desenvolvido em Rust, com abordagem bidimensional e tridimensional, de código aberto e de grande comunidade de contribuidores. Em seu desenvolvimento, foram usados os conceitos mais modernos em desenvolvimento de jogos, como a arquitetura Entity-Component-System (ECS) e o design *data-driven*, tendo um código altamente escalável e orientado à eventos. Este projeto é um dos motores de jogos mais populares e poderosos, dos que existem nesta mesma linguagem, sendo um ótimo exemplo da capacidade da tecnologia no meio do desenvolvimento de jogos.

Como apresentado em seu livro, Klabnik e Nichols (2023) destacam a segurança que a linguagem proporciona em relação ao gerenciamento da memória utilizada no processamento do software, que com a ajuda das próprias ferramentas nativas do Rust, é necessário manualmente especificar quais objetos serão mutáveis ou não e suas referências da memória se encontram dentro de cada escopo.

Com relação à performance da linguagem, de acordo com Bugden e Alahmar (2022), que usou como parâmetro de comparação a execução do algoritmo de ordenação Bubble Sort, Rust se destaca positivamente mesmo entre outras linguagens que fazem uso da alocação manual da memória no código fonte, como C e C++. Em comparação com linguagens que fazem uso de gerenciamento automático de memória, como o Java, Rust fica muito a frente em economia e velocidade.

REFERÊNCIAS

BUGDEN, William.; ALAHMAR, Ayman. Rust: The Programming Language for Safety and Performance. *In*: 2nd International Graduate Studies Congress (IGSCONG'22), 2., 8-11 jun. 2022, Turkey. ArXiv, jun. 2024.

CAMPOS, Sergio Alejandro Espinal.; MORALES, Bryan Antony Miramira.; NÚÑEZ, Ángel Augusto Velásquez. **Open Source Game Engine & Framework for 2D Game Development**. *In*: 2022 IEEE Engineering International Research Conference (EIRCON), 26-28 out. 2022, Lima. IEEE Xplore: IEEE, nov. 2022.

GIANADDA, David Jean Raymond. **Coding a 2D Game Engine for ESP32:** A Showcase of Design Patterns. 2024. Product type thesis (Bachelor of Business Information Technology) - Haaga-Helia University of Applied Sciences, Helsinki.

GREGORY, James. **Game Engine Architecture**, Third Edition. 4. ed. Nova Iorque. A K Peters/CRC Press, 2018. 1240p.

KLABNIK, Steve.; NICHOLS, Carol. **The Rust Programming Language**, 2nd Edition. 4. ed. San Francisco. No Starch Press, 2023. 560p.

MCMILLIAN, Taryn. **5 Excellent Reasons Why You Should Be Developing Games**, Canadá, 2021. Disponível em: https://tarynwritescode.hashnode.dev/5-excellent-reasons-why-you-should-be-developing-games. Acesso em: 5 set. 2024.

MININI, Pedro Probst. **Exploração de técnicas atuais de programação de jogos e geração procedural de ambientes através do desenvolvimento de um protótipo de jogo em Rust**. 2020. Trabalho de Conclusão de Curso (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Santa Maria (UFSM, RS), Santa Maria.

ÖZGÖVDE, Bahri Atay.; KURT, Fatih Mustafa. Edge Computing for Computer Games by Offloading Physics Computation. 2023. Research Article (Computer Engineering) - Department of Computer Engineering, Boğaziçi University, Istambul, 2023.

SRŠEN, Mirko.; OREHOVAČKI, Tihomir. Developing a Game Engine in C# Programming Language. *In*: 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), 44., 27 set. – 1 out. 2021, Opatija. IEEE Xplore: IEEE, nov. 2021.

2024 Stack Overflow Developer Survey. **Stack Overflow**, 2024. Disponível em: https://survey.stackoverflow.co/2024/. Acesso em: 5 set. 2024.