**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO**

**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – CJOPROO (PROGRAMAÇÃO ORIENTADA A OBJETOS) – PROF. PAULO GIOVANI DE FARIA ZEFERINO**

**BRIAN GUSTAVO DE OLIVEIRA BRANDÃO**

# PROJETO FINAL: JOGO EM C++ (BRASIDLE)

**CAMPOS DO JORDÃO**

**2025**

# RESUMO

Este documento descreve o desenvolvimento e a implementação do Brasidle, um jogo do gênero *idle* com temática cultural brasileira, criado como projeto de aplicação prática de conceitos de programação orientada a objetos. O objetivo principal deste trabalho consistiu em projetar e construir uma aplicação de entretenimento digital funcional, utilizando a linguagem de programação C++ e a biblioteca gráfica Raylib, com foco na otimização de recursos e na facilidade de uso. Os resultados alcançados demonstram a viabilidade de construir jogos de pequena a média complexidade com a combinação das ditas ferramentas, oferecendo uma experiência de jogo fluida e engajadora. O projeto obteve sucesso na criação de um ambiente temático envolvente e na implementação correta das mecânicas *idle* progressivas, que equilibram a interação direta (*clicker*) com o acúmulo passivo de recursos. Conclui-se que a utilização de tais recursos oferece uma solução robusta e de baixo nível para o desenvolvimento de jogos que demandam controle de *hardware* e desempenho eficiente, cumprindo o objetivo de fornecer uma plataforma de aprendizado e desenvolvimento de software. A experiência adquirida valida o uso da biblioteca para futuros projetos de desenvolvimento de jogos 2D.

**Palavras-Chave**: Raylib; Brasidle; desenvolvimento de jogos.

# 1 INTRODUÇÃO

O presente trabalho descreve o processo de desenvolvimento do jogo eletrônico Brasidle, um *idle* temático que explora elementos da cultura brasileira. O projeto foi concebido como um estudo prático no campo do desenvolvimento de *software* e da programação orientada a objetos (POO).

## 1.1 Objetivos

Este trabalho tem por objetivo construir um produto de entretenimento digital funcional e estável, demonstrando a aplicação dos princípios de *game design* e das técnicas de codificação eficientes.

Para a consecução deste objetivo foram estabelecidos os objetivos específicos:

- Implementar mecânicas de jogo *idle* progressivas e equilibradas;
- Utilizar a linguagem C++ para garantir alto desempenho e controle de *hardware*;
- Explorar a biblioteca Raylib como ferramenta de desenvolvimento gráfico 2D simplificado.

## 1.2 Justificativa

O desenvolvimento do Brasidle se justifica pela relevância técnica e educacional de projetos que utilizam ferramentas de baixo nível (como C++) para desenvolvimento de jogos. Na área de *software*, a combinação de C++ com Raylib proporciona um ambiente de aprendizado valioso sobre otimização de memória, gestão de recursos e o ciclo de vida do *game loop*. Além disso, o gênero *idle* é uma tendência crescente no mercado de jogos, e este projeto oferece um estudo de caso sobre como aplicar essa mecânica em um contexto culturalmente rico e localizado no Brasil.

## 1.3 Aspectos Metodológicos

O presente estudo fez uso de uma abordagem metodológica mista, combinando pesquisa bibliográfica e pesquisa aplicada.

A abordagem da pesquisa bibliográfica foi fundamental para a construção do aporte teórico e para a base de conhecimento sobre a linguagem C++ e os princípios da programação orientada a objetos (POO). As informações foram coletadas a partir de materiais didáticos e pesquisas online.

Já a etapa da pesquisa aplicada corresponde à parte prática do trabalho, que é o desenvolvimento e a implementação do jogo. A metodologia de desenvolvimento utilizada foi de enfoque na prototipagem rápida e orgânica, permitindo a validação imediata da funcionalidade do código (cálculo de recursos e renderização gráfica) e a experimentação direta das mecânicas de jogo, garantindo que o projeto cumprisse seus objetivos de forma funcional.

## 1.4 Aporte Teórico

O aporte teórico principal sustenta-se em duas áreas. A primeira é a programação orientada a objetos (POO) e a linguagem C++, com a utilização de conceitos como encapsulamento, estruturas de dados e a gestão eficiente de memória. Estes fundamentos foram abordados tanto ao longo da disciplina quanto através de pesquisas e literatura especializada que definem as boas práticas de codificação de baixo nível.

A segunda área abrange o desenvolvimento de jogos 2D e as mecânicas *idle*. As referências para esta parte foram coletadas através de exemplos da biblioteca Raylib (para a renderização gráfica, *game loop* e *input*), bem como por pesquisas online sobre *game design*, que orientaram o equilíbrio e a progressão das mecânicas passivas e ativas do jogo.

## 2 RESULTADOS OBTIDOS

Como primeiro passo para começar a se desenvolver um jogo em Raylib, é necessário instalar a biblioteca em questão, além de um *template* que facilite o desenvolvimento para C++ através da inclusão prévia de *wrappers* próprios pra linguagem (já que a biblioteca originalmente é em C). Adicionalmente, foi usado também o complemento RayGUI para a interface de usuário, que precisa ser adquirido separadamente.

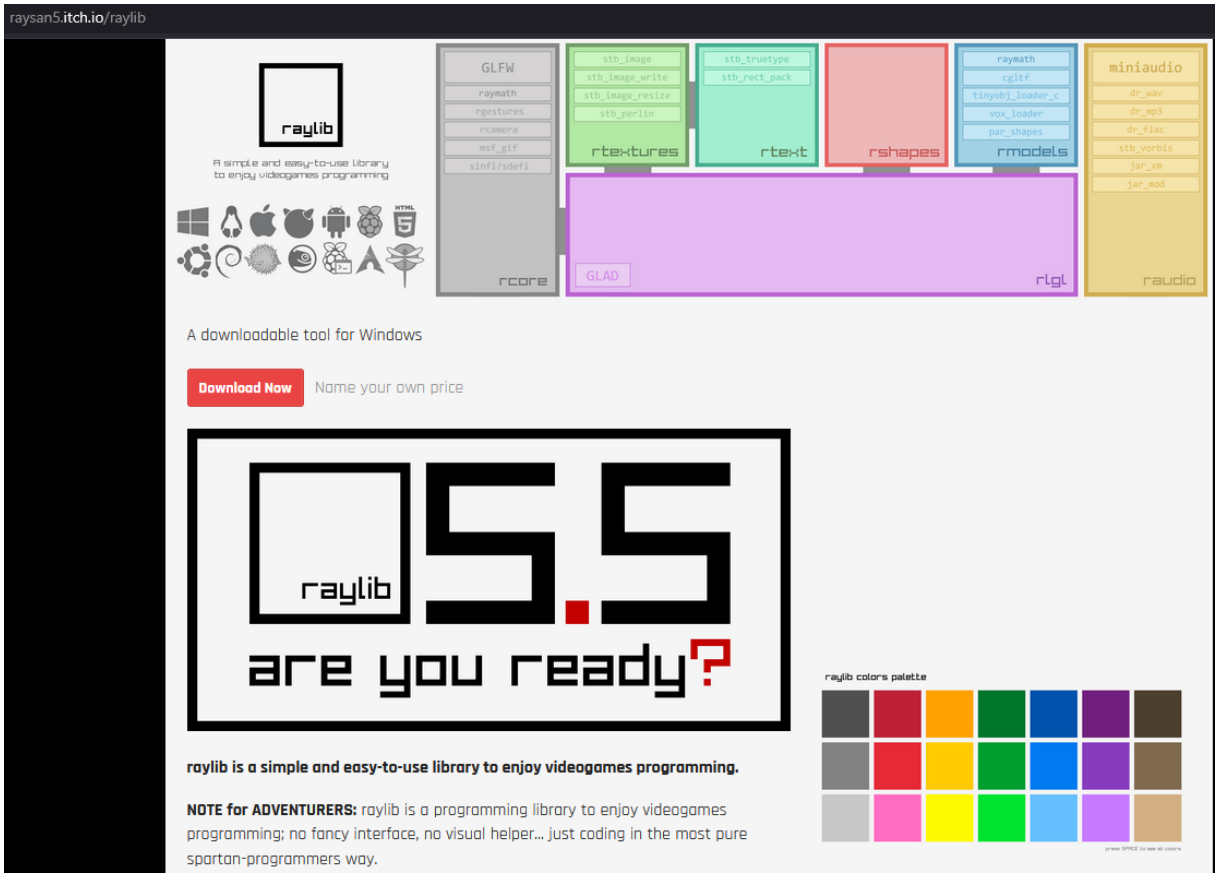


Figura 1: Instalador do Raylib na plataforma itch.io

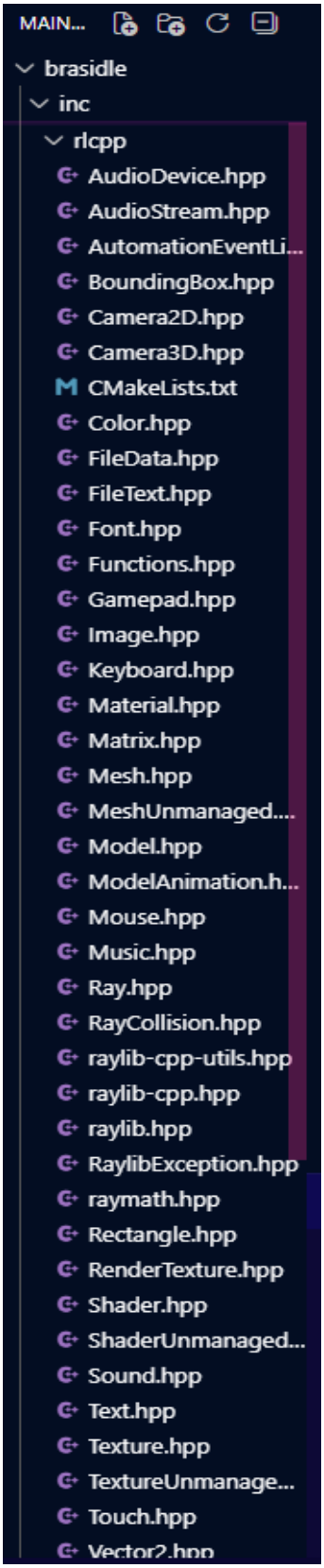


Figura 2: *Wrappers* para as funções do Raylib em C++

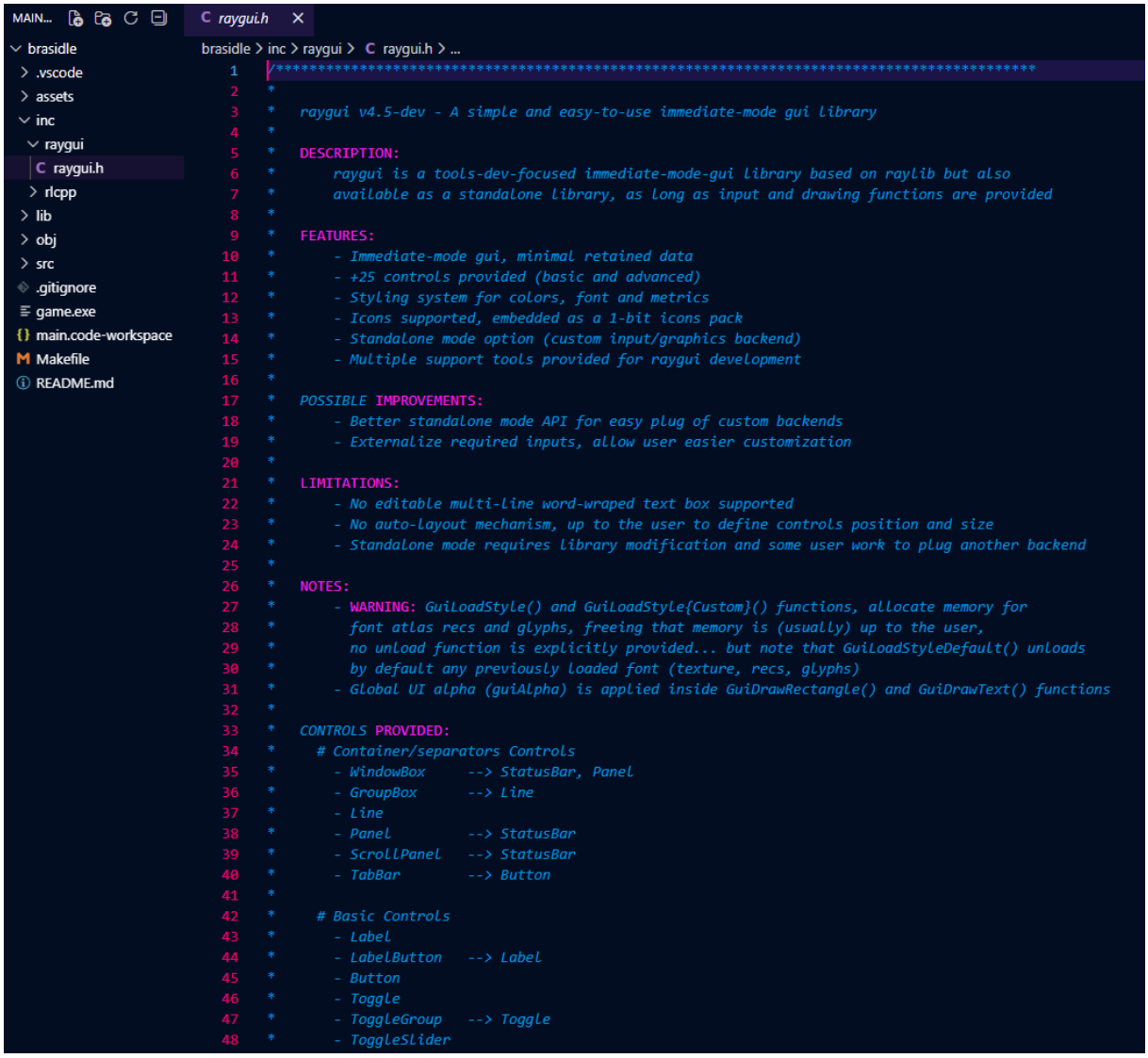


Figura 3: Arquivo complementar do RayGUI (*header-only library*)

Com a base estabelecida, foi possível começar a desenvolver a ideia. O conceito escolhido foi o do Brasidle, um jogo *idle* com temática brasileira e tom descontraído. A principal inspiração foi o Cookie Clicker, um dos exemplos mais famosos do gênero e conhecido pela sua simplicidade, sendo capaz até de funcionar em navegadores.


Figura 4: Imagem do Cookie Clicker

Para serem usadas como *assets* externos, foram escolhidas três imagens e uma música de fundo. A música é uma versão de “Boate Azul” (popularizada pela dupla sertaneja Bruno e Marrone) tocada no estilo “8-bit”, ou seja, remetendo à jogos antigos e suas respectivas trilhas sonoras. As imagens são dois ícones de áudio ligado e desligado (para a funcionalidade de desativar o som do jogo), além de uma moeda de 1 real brasileira, que serve como o equivalente do cookie no Cookie Clicker.

O resultado final do jogo ficou da seguinte maneira:

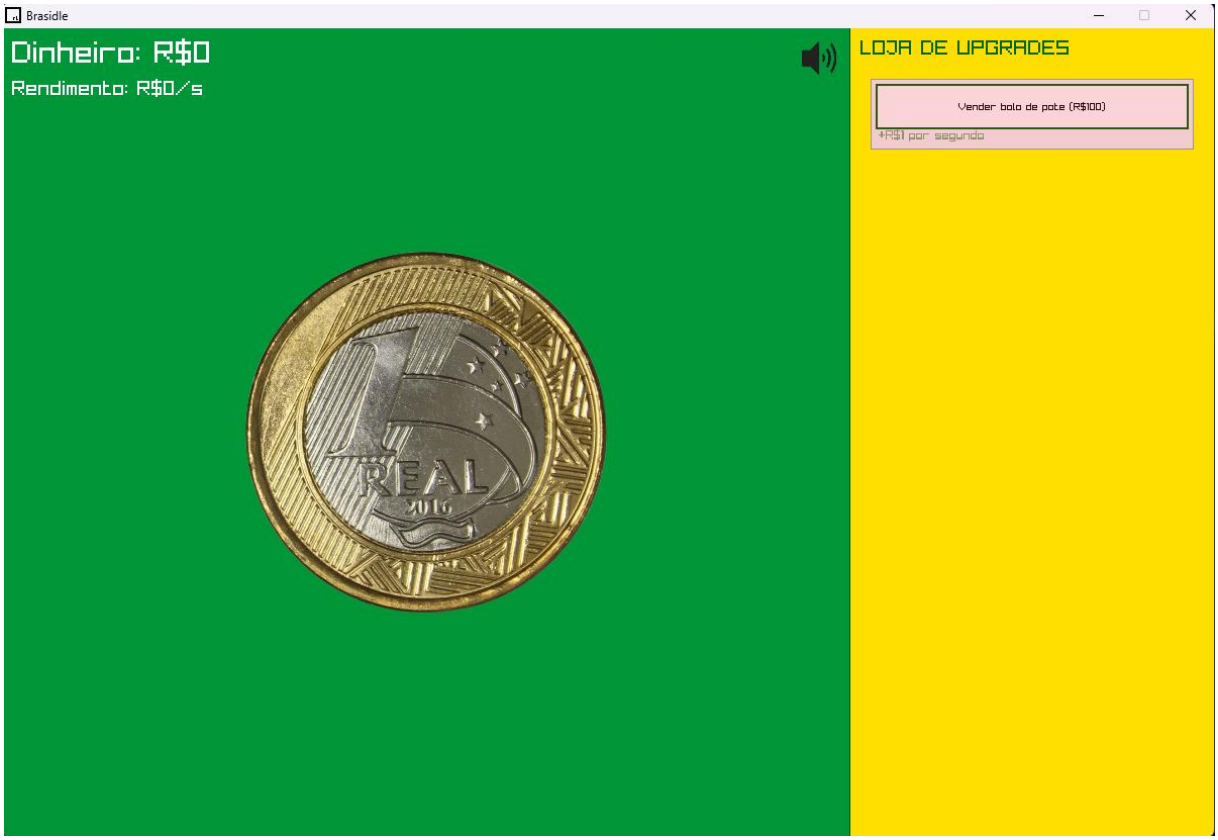


Figura 5: Captura de tela do Brasidle

Para progredir no jogo, é necessário clicar na moeda para receber reais, que por sua vez são usados para adquirir *upgrades*. Foram implementadas cinco destas melhorias (após a obtenção de todas, o jogo efetivamente se encerra), que podem aumentar o valor do clique de 1 real pra 5 ou adicionar mais dinheiro ao rendimento, que providencia uma certa quantia de reais automaticamente ao jogador por segundo.

Por questão de conveniência, se optou por não inserir a lógica de progresso offline conforme é encontrado em alguns jogos do gênero (tais como Adventure Capitalist), mas é possível para o jogador voltar de onde parou ao fechar e posteriormente reabrir seu jogo, pois um savegame.dat é automaticamente criado quando a janela é fechada – e, ao voltar, a existência desse salvamento (que armazena o progresso do usuário através de um JSON) será reconhecida para que o *player* tenha suas aquisições restauradas.

Como o jogo é simples e não necessita de muitas linhas de código (não há nenhuma lógica de colisão ou outras situações que envolvam física, uso de 3D,

dentre outros quesitos que geralmente costumam causar complicações para desenvolvedores em projetos de pequena escala), todo o código foi mantido na classe principal main.cpp, apesar do Raylib possuir excelente suporte ao paradigma orientado à objetos (com arquivos .hpp e .cpp separados para definição de classes específicas a serem inseridas posteriormente em main através de suas funções).

# 3 CONCLUSÃO

Os resultados obtidos demonstram a viabilidade técnica de construir jogos 2D funcionais, mesmo com um escopo de desenvolvimento conciso. O projeto obteve sucesso na aplicação dos fundamentos da POO, organizando a lógica de jogo e a gestão de recursos de forma clara e eficiente, apesar da centralização do código em um único arquivo.

Os objetivos específicos propostos foram plenamente alcançados:

- Implementar mecânicas de jogo *idle* progressivas e equilibradas: As mecânicas foram implementadas e a progressão baseada em *upgrades* foi estabelecida com sucesso, permitindo o acúmulo passivo e ativo de recursos.
- Utilizar a linguagem C++ para garantir alto desempenho e controle de *hardware*: A escolha de C++ e Raylib garantiu um desempenho otimizado e um *game loop* fluido, cumprindo a meta de utilizar uma solução de baixo nível.
- Explorar a biblioteca Raylib como ferramenta de desenvolvimento gráfico 2D simplificado: O Raylib mostrou-se uma ferramenta eficaz para a renderização 2D e o gerenciamento de *input*, validando seu uso para prototipagem e desenvolvimento de jogos.

Em síntese, o Brasidle atingiu seu objetivo geral de construir um produto de entretenimento digital funcional e estável, oferecendo uma experiência de jogo coesa e validando o aprendizado prático das ferramentas e conceitos propostos pela disciplina.

Todas as atividades centrais previstas no cronograma (configuração do ambiente C++/Raylib, implementação do *game loop*, criação da lógica de *idle* e do sistema de *upgrades* base) foram integralmente cumpridas. Embora as funcionalidades essenciais estejam operacionais, a quantidade de conteúdo final (apenas cinco *upgrades*) foi limitada em relação ao potencial total do gênero *idle*. Esta

limitação não comprometeu a funcionalidade do produto final, mas representa uma restrição de escopo que pode ser atribuída à necessidade de cumprir o prazo da disciplina, priorizando a estabilidade e a correta aplicação dos conceitos de POO sobre a expansão massiva de conteúdo.

A experiência de desenvolvimento do Brasidle abre caminhos para futuras expansões e melhorias, que podem ser consideradas como hipóteses para trabalhos subsequentes. A principal hipótese é a criação de árvores de *upgrades* mais extensas e ramificadas, garantindo longevidade ao *gameplay* além dos minutos iniciais de jogo. Poderia também ser introduzida a mecânica de prestígio, onde o jogador pode resetar o progresso em troca de um bônus permanente, elevando a complexidade e a profundidade estratégica do jogo. Além disso, para fins de aprendizado, seria útil refatorar o código, migrando as funcionalidades do main.cpp para classes C++ dedicadas (*e.g.*, GerenciadorDeRecursos, InterfaceDeUsuario), para demonstrar uma aplicação mais completa dos pilares da POO (encapsulamento, herança).

O projeto, portanto, não é apenas um resultado final, mas um ponto de partida que valida a escolha das ferramentas e metodologia, oferecendo uma base sólida para a continuidade e a expansão.

## REFERÊNCIAS

Materiais didáticos do prof. Paulo Giovani de Faria Zeferino.

Documentação oficial e exemplos da biblioteca Raylib.

MANZANO, José Augusto. **C++ 23 para Windows: guia de introdução para iniciantes.** 1. ed. Clube de Autores, 2025