INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Raul Dias Pereira

Light Escape

CAMPOS DO JORDÃO

2025

RESUMO

O projeto apresenta o desenvolvimento do jogo Light Escape, criado na linguagem C++ com o uso da biblioteca gráfica Raylib. O jogo consiste em um desafio no qual o jogador controla uma esfera de luz que percorre labirintos escuros em busca da saída. A mecânica central baseia-se na limitação de visibilidade, em que o cenário só é parcialmente iluminado ao redor do personagem, exigindo exploração constante do espaço. A estrutura do jogo conta com três fases fixas, representadas por labirintos distintos, e três níveis de dificuldade que se diferenciam exclusivamente pelo tempo disponível para completar todas as fases. A metodologia adotada envolveu o uso de programação orientada a objetos para organizar a lógica do jogo em classes específicas para o jogador, os níveis e o sistema de controle geral do jogo. Foram aplicadas práticas de encapsulamento, modularização e uso de estruturas de dados vetoriais. Conclui-se que o projeto atingiu seu objetivo principal de consolidar conceitos da programação orientada a objetos através de uma aplicação prática, visual e interativa. O sistema de iluminação localizado, combinado com o tempo como fator de dificuldade, compôs uma dinâmica eficiente de jogo.

Palavras-Chave: Programação orientada a objetos; C++; Raylib; jogo; labirinto.

ABSTRACT

This project details the development of the game Light Escape, an interactive application created in C++ using the Raylib graphics library. The game challenges the player to control a sphere of light navigating dark mazes in search of the exit. The core mechanic revolves around limited visibility, where the environment is only partially illuminated around the character, demanding constant exploration of the space. The game's structure comprises three fixed stages, each represented by a distinct maze, and offers three difficulty levels that differ exclusively by the time available to complete all stages. The methodology adopted for development focused on Object-Oriented Programming (OOP), organizing the game's logic into specific classes for the player, levels, and the overall game control system. Practices such as encapsulation, modularization, and the use of vector data structures were applied. In conclusion, the project achieved its primary objective of consolidating OOP concepts through a practical, visual, and interactive application. The localized lighting system, combined with time as a difficulty factor, proved to be an efficient and engaging game dynamic.

Keywords: Object-Oriented Programming; C++; Raylib; game; maze.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão introduzidos todos os assuntos abordados por este documento. Pretende-se apresentar a motivação, os objetivos e a organização do texto.

1.1 Objetivos

Este trabalho tem por objetivo principal consolidar e aplicar conceitos de Programação Orientada a Objetos (POO) de forma prática e interativa, por meio do desenvolvimento do jogo Light Escape.

Para a consecução deste objetivo foram estabelecidos os objetivos específicos:

- Desenvolver um sistema de jogo funcional que explorasse a mecânica de iluminação dinâmica em labirintos escuros.
- Implementar uma estrutura de fases e níveis de dificuldade que proporcionasse uma experiência de jogo diferente e desafiadora para cada tipo de pessoa.
- Utilizar a biblioteca Raylib para o desenvolvimento gráfico e a interação com o usuário, demonstrando seu potencial em projetos de jogos 2D.
- Estruturar o código de forma modular e escalável, aplicando princípios de encapsulamento e coesão.

1.2 Justificativa

A relevância deste trabalho se manifesta na oportunidade de aplicar e aprofundar conhecimentos na linguagem C++ e em Programação Orientada a Objetos criando um projeto desafiador, mas também divertido, como o desenvolvimento de um jogo. Jogos, por sua natureza na modelagem de entidades, gerenciamento de estados e interações, são um cenário ideal para a prática de conceitos como classes, objetos, herança e polimorfismo. Além disso, o Light Escape permite explorar soluções

criativas para a limitação de visibilidade, adicionando uma camada estratégica e de imersão.

1.3 Aspectos Metodológicos

O presente estudo fez uso de uma metodologia focada na programação orientada a objetos, com a criação de classes bem definidas para as entidades do jogo como Player e Level. O desenvolvimento seguiu um fluxo iterativo, permitindo revisões e aprimoramentos contínuos das funcionalidades. A implementação gráfica e a interação com o usuário foram realizadas utilizando a biblioteca Raylib.

1.4 Aporte Teórico

As bases teóricas que fundamentaram este trabalho estão centradas nos princípios da Programação Orientada a Objetos (POO), incluindo conceitos como encapsulamento, modularização e abstração. A implementação gráfica e a lógica de interação foram desenvolvidas com o suporte da biblioteca Raylib. Além disso, foram exploradas estruturas de dados para o gerenciamento dos elementos do jogo.

2 METODOLOGIA

Nesta seção, serão detalhadas as considerações iniciais sobre o projeto Light Escape, as ferramentas e tecnologias empregadas, e a descrição do desenvolvimento do jogo, com ênfase na aplicação da Programação Orientada a Objetos.

2.1 Considerações Iniciais do Projeto

O projeto Light Escape foi concebido com o objetivo de criar uma aplicação interativa que desafiasse o jogador em um ambiente de labirinto com visibilidade limitada. A ideia central era proporcionar uma experiência de exploração e estratégia, onde o tempo combinado com a falta de visibilidade seriam o fator de dificuldade. Desde o início, a intenção foi utilizar uma abordagem de desenvolvimento que reforçasse os conceitos de Programação Orientada a Objetos (POO), permitindo a organização, modularidade e reusabilidade do código.

2.2 Ferramentas Utilizadas

Para o desenvolvimento do jogo Light Escape, foram empregadas as seguintes ferramentas e tecnologias:

- Linguagem de Programação C++: Utilizada por sua capacidade de oferecer alta performance e controle sobre os recursos do sistema, sendo uma linguagem comumente aplicada no desenvolvimento de jogos. O C++ fornece um robusto suporte para a organização do código através da Programação Orientada a Objetos (POO).
- Biblioteca Gráfica Raylib: Uma biblioteca simples e fácil de usar para o desenvolvimento de jogos 2D e aplicações gráficas. A Raylib oferece funcionalidades essenciais para o desenho de elementos visuais, gerenciamento de entrada do usuário e outras operações gráficas, facilitando a construção da interface e da jogabilidade do projeto.
- Visual Studio Code: Para a edição, compilação e depuração do código, foi

utilizado o ambiente de desenvolvimento integrado (IDE) Visual Studio Code.

2.3 Descrição do Projeto Desenvolvido

O jogo Light Escape foi estruturado em um modelo orientado a objetos, com a definição de classes que encapsulam as características e comportamentos das principais entidades do jogo. Abaixo, detalha-se a implementação das classes e funcionalidades:

2.3.1 Estrutura de Dados e Constantes

As dimensões do labirinto e o tamanho dos tiles são definidos por constantes globais: MAZE_WIDTH, MAZE_HEIGHT e TILE_SIZE. A estrutura Level agrupa o labirinto (representado por um std::vector<std::vector<int>> onde 1 é parede e 0 é caminho) e a posição de saída (Vector2 exit).

2.3.2 Classe Player

A classe Player modela o personagem controlável pelo jogador. Seus principais atributos e métodos são:

Atributos:

- tileX, tileY: Coordenadas do jogador no grid do labirinto, essenciais para o sistema de colisão e movimento.
- color: Define a cor visual da esfera do jogador.
- moveDelay, moveTimer: Variáveis para controlar o tempo entre movimentos contínuos, proporcionando uma jogabilidade mais suave e menos suscetível a "teleportes" por pressionamento rápido de tecla.

Métodos:

- Player(): Construtor que inicializa a posição do jogador.
- ResetPosition(): Método para retornar o jogador à posição inicial, utilizado ao iniciar um novo nível.
- Update(const std::vector<std::vector<int>>& maze): Responsável por processar a entrada do usuário (teclas de direção) e atualizar a posição do jogador, verificando colisões com as paredes do labirinto. A lógica de moveTimer garante que o jogador não se mova excessivamente rápido.
- GetPixelPosition() const: Retorna a posição do jogador em coordenadas de pixel, necessária para o desenho na tela.
- Draw(): Desenha o jogador como uma esfera e adiciona um elemento visual de "coroa" que representa a luz, reforçando a temática do jogo. Este método demonstra o uso de funções de desenho da Raylib.

2.3.3 Labirintos e Níveis

Embora não haja uma classe Maze explícita, a lógica dos labirintos é encapsulada na estrutura Level e gerenciada pela classe Game. Os três labirintos (maze1, maze2, maze3) são definidos como std::vector<std::vector<int>> globais, instanciando diferentes estados para cada nível do jogo. Esta abordagem modular permite adicionar novos labirintos facilmente.

2.3.4 Classe Game

A classe Game atua como o controlador principal do jogo, gerenciando o estado geral, a progressão dos níveis, a dificuldade e a interação entre o jogador e os elementos do ambiente.

Atributos:

- player: Instância da classe Player.
- timer: Controla o tempo restante para o jogador completar o labirinto, sendo um fator de dificuldade.
- gameOver, venceu: Flags booleanas que indicam o estado final do jogo (derrota ou vitória).

- difficulty: Armazena a dificuldade selecionada pelo jogador.
- currentLevel: Indica o nível (labirinto) atual em que o jogador se encontra.
- levels: Um std::vector de objetos Level, contendo todos os labirintos predefinidos e suas saídas.
- tempoFinal: Armazena o tempo restante ao vencer o jogo.

Métodos:

- Game(): Construtor que inicializa o estado do jogo e preenche o vetor levels com os labirintos e suas saídas.
- SetDifficulty(int difficultyChoice): Ajusta o timer inicial do jogo com base na dificuldade escolhida (1, 2 ou 3).
- NextLevel(): Avança o jogo para o próximo labirinto. Se todos os níveis forem completados, define o estado de vitória. Caso contrário, reseta a posição do jogador.
- Update(): Método principal de atualização da lógica do jogo. Ele chama o player.Update(), decrementa o timer e verifica as condições de game over (tempo esgotado) ou vitória no nível atual.
- CheckVictory(): Verifica se a posição do jogador coincide com a posição de saída do labirinto atual.
- Draw(): Responsável por renderizar todos os elementos do jogo na tela, incluindo o desenho dinâmico do labirinto (com o efeito de "luz" ao redor do jogador), a saída, o jogador e a interface de usuário (HUD) com o tempo restante. Também gerencia a exibição das telas de "Game Over" e "Vitória".
- DrawDifficultyMenu(): Exibe as opções de dificuldade para o jogador.

2.3.5 Fluxo do Jogo

O fluxo principal do jogo, gerenciado pela função main(), alterna entre diferentes estados:

- Tela Inicial: Exibe o título do jogo e a instrução para iniciar. Ao pressionar ENTER, transita para o menu de dificuldade.

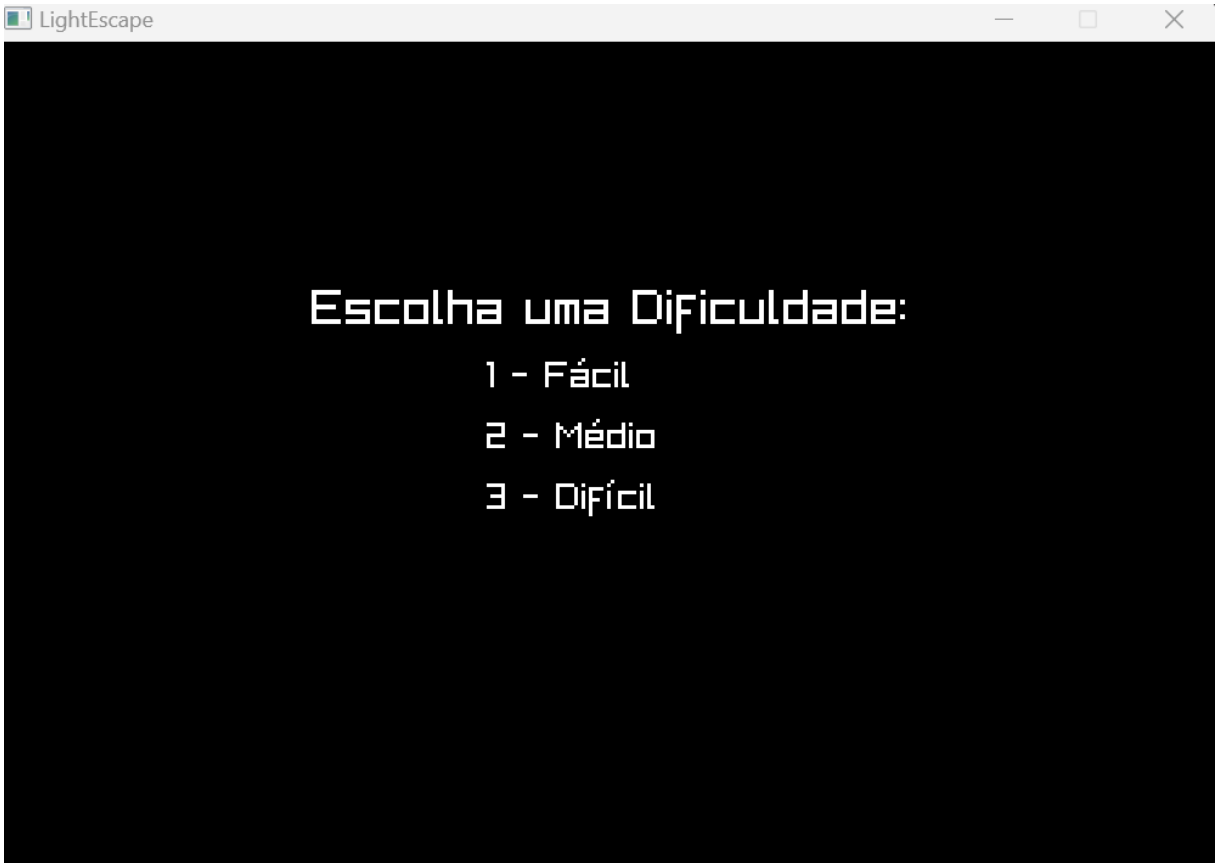
- Menu de Dificuldade: Apresenta as opções de dificuldade (Fácil, Médio, Difícil). A escolha define o tempo disponível e inicia o jogo.
- Jogo em Andamento: O loop principal do jogo executa os métodos Update() e Draw() da classe Game. O jogo continua até que o tempo se esgote (derrota) ou o jogador complete todos os labirintos (vitória).
- Telas Finais: Em caso de derrota ou vitória, as telas correspondentes são exibidas. Se o jogador vencer, é possível reiniciar o jogo pressionando ENTER para voltar à tela inicial.

Essa estrutura modular e a aplicação de POO permitiram uma clara separação de responsabilidades entre as classes, facilitando o desenvolvimento, a manutenção e a possível expansão do jogo com novas funcionalidades ou níveis.

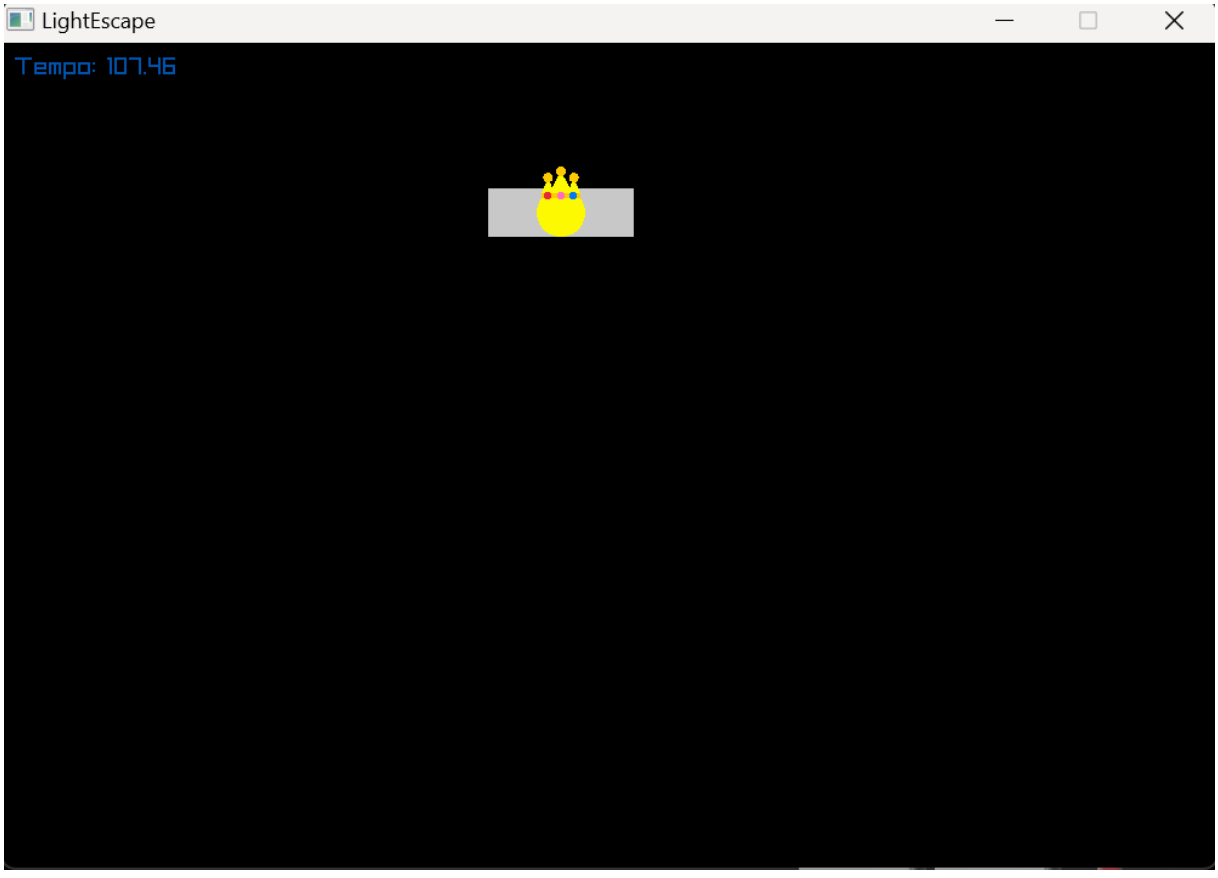
3 RESULTADOS OBTIDOS

3.1 Tela de login

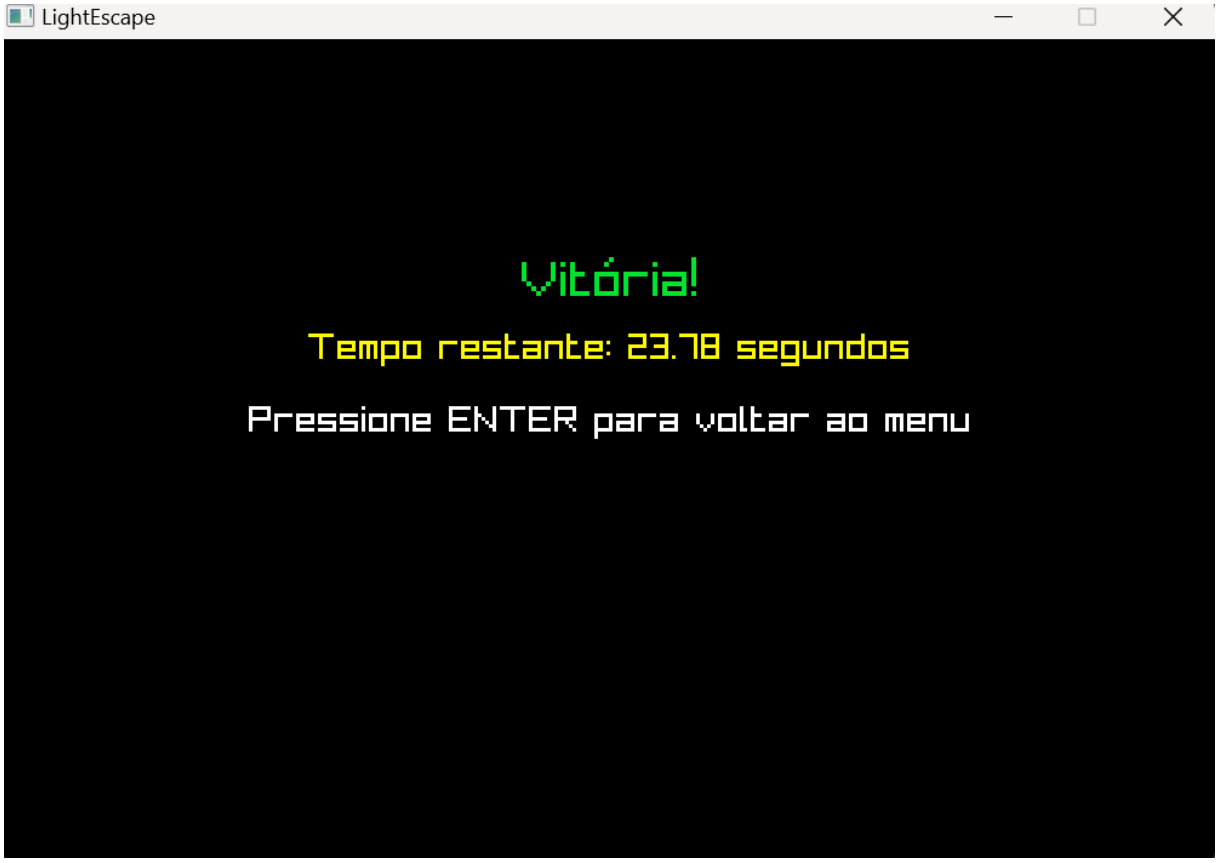
3.2 Tela de Escolha de dificuldade



3.3 Personagem em Jogo



3.4 Tela de Vitória



3.5 Tela de Derrota


LightEscape
Game Over!

4 CONCLUSÃO

O presente trabalho teve como objetivo principal consolidar e aplicar conceitos de Programação Orientada a Objetos (POO) de forma prática e interativa, por meio do desenvolvimento do jogo Light Escape. Ao longo do projeto, foi possível observar que os objetivos propostos foram satisfatoriamente alcançados, conforme detalhado a seguir.

O primeiro objetivo, de desenvolver um sistema de jogo funcional que explorasse a mecânica de iluminação dinâmica em ambientes escuros, foi plenamente atingido. O jogo Light Escape demonstra de forma eficaz a limitação de visibilidade através de uma área iluminada restrita ao redor do jogador, exigindo exploração cautelosa e estratégia. Essa mecânica, central para a experiência de jogo, foi implementada com sucesso, criando um ambiente imersivo e desafiador.

Em relação ao segundo objetivo, de implementar uma estrutura de fases e níveis de dificuldade que proporcionasse uma experiência de jogo progressiva e desafiadora, o projeto também obteve êxito. Com três labirintos distintos e três níveis de dificuldade baseados no tempo disponível, o jogo oferece uma progressão clara e adaptável às habilidades do jogador, cumprindo a proposta de um desafio gradual.

O terceiro objetivo, de utilizar a biblioteca Raylib para o desenvolvimento gráfico e a interação com o usuário, demonstrando seu potencial em projetos de jogos 2D, foi amplamente cumprido. A Raylib se mostrou uma ferramenta robusta e eficiente, permitindo a criação de gráficos simples, mas funcionais, e a implementação de controles responsivos, validando sua aplicabilidade em projetos de jogos 2D.

Por fim, o quarto objetivo, de estruturar o código de forma modular e escalável, aplicando princípios de encapsulamento e coesão, foi um dos pilares do desenvolvimento. A utilização intensiva de Programação Orientada a Objetos, com classes bem definidas como Player, Level e Game, garantiu um código organizado, de fácil manutenção e expansão. A separação de responsabilidades entre as classes

e o uso de estruturas de dados vetoriais contribuíram significativamente para a modularidade do projeto.

Em síntese, o desenvolvimento do Light Escape comprovou a eficácia da aplicação dos conceitos de Programação Orientada a Objetos em um projeto prático e visual. O jogo demonstra a capacidade de criar uma experiência envolvente com mecânicas simples, mas bem elaboradas, como a iluminação localizada e o fator tempo como dificuldade.

4.1 Sugestões para Melhorias Futuras

Para futuras iterações e aprimoramento do projeto Light Escape, algumas sugestões podem ser consideradas:

- Geração Procedural de Labirintos: Implementar algoritmos para a criação de labirintos de forma procedural permitiria uma variedade infinita de fases, aumentando a rejogabilidade e o desafio, em vez de depender de labirintos fixos.
- Adição de Inimigos ou Obstáculos Dinâmicos: A introdução de elementos móveis, como inimigos que patrulham o labirinto ou armadilhas ativadas, adicionaria uma camada extra de complexidade e estratégia à jogabilidade.
- Sistema de Pontuação e Recordes: Desenvolver um sistema de pontuação baseado no tempo restante ou em itens coletados, com um placar de recordes, incentivaria a competição e a busca por melhores desempenhos.
- Power-ups ou Habilidades Especiais: Incluir itens que o jogador possa coletar para obter vantagens temporárias, como aumento da área de luz, invencibilidade ou aceleração, enriqueceria a dinâmica do jogo.
- Efeitos Sonoros e Trilha Sonora: A adição de áudios para interações (movimentos, colisões, vitória/derrota) e uma trilha sonora atmosférica melhoraria significativamente a imersão do jogador.
- Aprimoramento Visual: Explorar texturas para paredes e chão, além de efeitos de iluminação mais avançados (ex: gradiente de luz), poderia elevar a qualidade gráfica do jogo.

5 REFERÊNCIAS

MAZANO, Augusto. **C++23 para Windows**. Campos do Jordão: Editora Saraiva, 2025.

RAYLIB. **raylib: a simple and easy-to-use library to enjoy videogames programming**. Disponível em: https://www.raylib.com/. Acesso em: 27 maio 2025.

TERMINAL ROOT. **Crie jogos para Windows, Linux e Web com Raylib C/C++**. 2022. Disponível em: https://terminalroot.com.br/2022/11/crie-jogos-para-windows-linux-e-web-com-raylib-c-cpp.html. Acesso em: 27 maio 2025.

UNIVERSIDADE PONTIFÍCIA CATÓLICA DO RIO GRANDE DO SUL. **Vetores**. Disponível em: https://www.inf.pucrs.br/~pinho/LaproI/Vetores/Vetores.htm. Acesso em: 27 maio 2025.