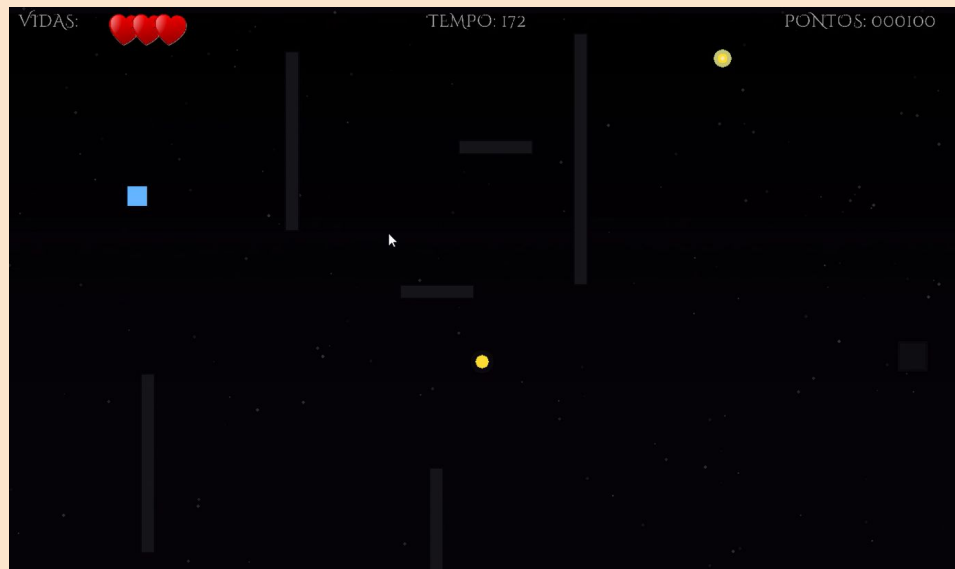
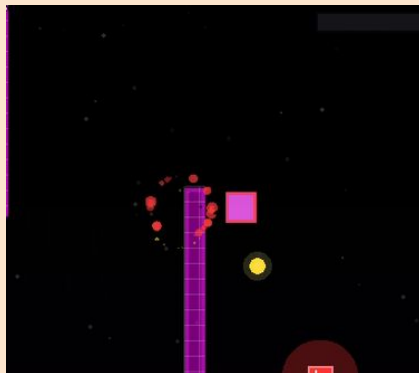


IFSP- CÂMPUS CAMPOS DO JORDÃO

LABIRINTO DAS SOMBRAS- JOGO
DESENVOLVIDO EM C++ UTILIZANDO
A BIBLIOTECA RAYLIB

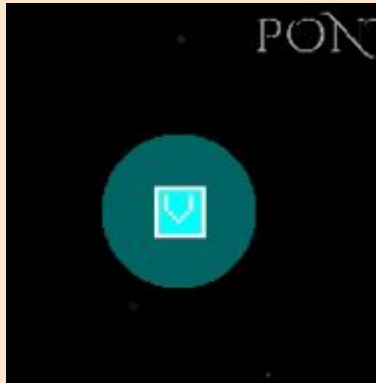
TIFANY LUIZA DE JESUS MOREIRA



O Jogo “O Labirinto das Sombras” surgiu com o intuito de melhorar as atribuições comuns de um jogo de labirinto padrão em que, por convenção, as paredes são estáticas, os caminhos definidos e sem muitos acréscimos, a certo ponto, predefinidos, sem adições de power-ups.

No game, foram prevalecidas as características padrões como: paredes, um início e um fim, e a necessidade de exercer o raciocínio lógico do player para achar o melhor caminho até o objetivo.

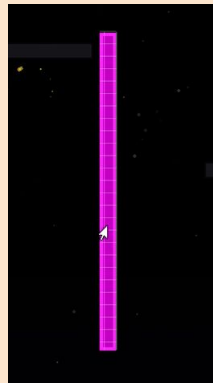
Ainda, além da adição e ênfase numa maior dinâmica durante as partidas, foram adicionados os conceitos de power-ups, tempo restante e vidas acumuladas, além de acréscimo de número de paredes e aumento de suas respectivas velocidades na medida que os níveis passa, a fim de tornar a experiência ainda mais divertida.



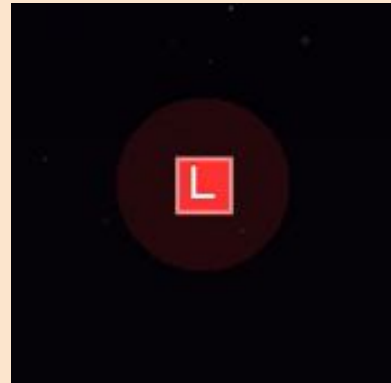
**Power-Up de
velocidade**



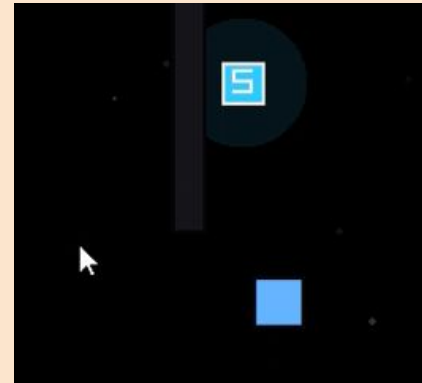
Coin



Wall



Life



**Power-Up de
proteção contra
paredes**



**Power-Up de
velocidade**

o **Power-up de Velocidade** aumenta a **velocidade base do jogador em 50%**. Ou seja, se a velocidade normal do jogador é de 250 unidades por segundo, ao coletar o power-up, ela sobe para 375 unidades por segundo. Esse bônus dura **4 segundos**, oferecendo um breve, mas crucial, período de agilidade.

A funcionalidade desse power-up é gerenciada na classe Player e na classe PowerUp.

- **Player::baseSpeed e Player::currentSpeed:** A classe Player possui duas variáveis para controlar a velocidade: baseSpeed (velocidade padrão) e currentSpeed (velocidade atual, que pode ser modificada por power-ups).
- **Player::speedBoostTimer:** Essa variável controla a duração do efeito. Quando o power-up é ativado, ela é definida para 4.0f segundos e decrece a cada atualização do jogo.
- **Aplicação do Efeito (PowerUp::TryCollect):** Quando o jogador colide com um power-up de velocidade e o coleta, o método TryCollect da classe PowerUp é chamado. Ele verifica o tipo do power-up e, se for PowerUpType::SPEED, a velocidade é acrescida ao player.



Coin

As **moedas (coins)** são itens coletáveis espalhados por cada fase, essenciais para a sua pontuação e para avançar no jogo. Cada moeda coletada adiciona **100 pontos** à sua pontuação total.

Além do acúmulo de pontos, o principal propósito das moedas é **habilitar o campo de objetivo (a saída do labirinto)**. Enquanto houver moedas ativas na fase, a saída permanecerá "desativada". Somente quando **todas as moedas de um nível são coletadas** o campo de objetivo se torna ativo, permitindo que você o alcance e complete a fase.

A lógica é simples: o **objetivo só é ativado quando todas as moedas da fase são coletadas**.

- A classe **Coin** gerencia cada moeda. Quando o jogador colide com uma, o método TryCollect() marca a moeda como inativa (active = false), concede pontos e toca um som como um feedback da coleta.
- A classe **Objetivo** representa a saída. Ela começa desabilitada (enabled = false).
- No loop principal do jogo, o método **Game::UpdatePlaying()** constantemente verifica se todas as moedas foram coletadas. Se sim, ele chama objetivo->Enable(), ativando a saída com um efeito visual e sonoro.

Essa mecânica garante que você explore o labirinto e colete tudo antes de avançar, adicionando estratégia ao seu caminho.



Wall

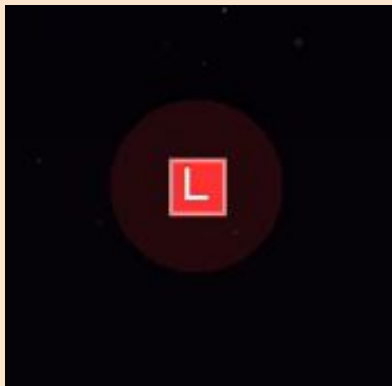
As paredes são os principais desafios durante uma partida (combinada com o tempo estipulado), exigindo agilidade para evitar a colisão contra as paredes. Existem dois tipos:

- **Paredes Comuns:** Blocos escuros e móveis. Bater nelas faz você perder uma vida, a não ser que tenha invencibilidade ou escudo.
- **Paredes Especiais:** Paredes roxas e pulsantes, mais perigosas por serem mais velozes. Elas também causam dano e podem ter movimentos mais complexos.

Ambos os tipos se movem pelo labirinto, e sua colisão resulta em perda de vida e efeitos visuais/sonoros, a menos que você esteja protegido por um Power-Up que te proteja de colisões.

A dinâmica das paredes no jogo é gerenciada por três classes principais:

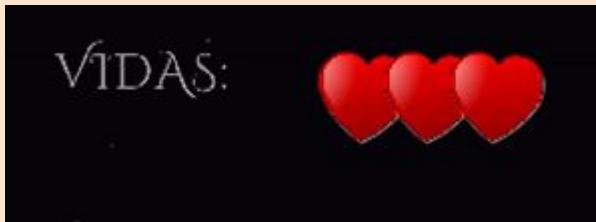
- **Wall:** É a base para todas as paredes. Define sua **posição, tamanho e velocidade** inicial. O método Update() movimenta a parede e inverte sua direção ao colidir com as bordas da tela (o efeito "quicar"). O Draw() renderiza a parede com cores básicas.
- **ParedeEspecial:** Herda de Wall, mantendo seu comportamento de movimento, mas sobrescreve o Draw() para criar um **efeito visual pulsante e translúcido**, usando a função sinf para animar a opacidade e cor, destacando sua periculosidade.
- **Game:** Orquestra as paredes. No LoadLevel(), ela **cria instâncias** de Wall e ParedeEspecial com características específicas para cada fase. Durante UpdatePlaying(), o jogo atualiza a posição de cada parede e, a cada frame, **verifica colisões** com o jogador. Se o jogador bater numa parede sem proteção (invencibilidade/escudo), ele perde uma vida, acionando efeitos sonoros, partículas e um tremor na tela para simular o impacto e dar um feedback mais visível para o player do evento.



Life

O sistema de **vidas** determina quantas chances o player tem de falhar antes do Game Over. A partida começa com **três vidas**, e cada colisão com uma parede ou obstáculo sem proteção resulta na perda de uma.

Para ajudar na sua jornada, existe o **power-up de vida**. Ao coletá-lo, será **recuperada uma vida**, até o máximo de três.



Como é estabelecido:

Player::lives: Essa variável na classe Player guarda o número atual de vidas do jogador, começando em 3 (Reset() a define).

Perda de Vidas (Player::TakeDamage()): Quando o jogador colide com uma parede, o método Player::TakeDamage() é chamado. Ele diminui lives em 1. Contudo, se o jogador estiver sob efeito de invincibilityTimer ou shieldTimer, ele não perde vida.

Coleta de Vidas (PowerUp::TryCollect()): No método PowerUp::TryCollect(), se o tipo do power-up for PowerUpType::LIFE e o jogador tiver menos de 3 vidas, player.lives é incrementado. Isso garante que não seja possível ter mais de 3 vidas.

Game Over (Game::UpdatePlaying()): No Game::UpdatePlaying(), o jogo constantemente checa player.lives. Se cair para 0 ou menos, o estado do jogo muda para GameState::GAMEOVER.



Power-Up de proteção contra paredes

O **power-up de escudo** concede **5 segundos de invulnerabilidade** contra paredes. Ao coletá-lo, você pode atravessar obstáculos sem perder vidas, ideal para momentos de perigo ou para se reposicionar no labirinto.



Player::shieldTimer: Uma variável na classe Player que inicia a contagem de 5 segundos de proteção.

Coleta (PowerUp::TryCollect): Quando o power-up de escudo é pego, player.shieldTimer é ativado.


Proteção contra Dano (Player::TakeDamage): O método TakeDamage() do jogador verifica se o shieldTimer (ou o timer de invencibilidade pós-dano) está ativo. Se sim, o dano da colisão é ignorado, e nenhuma vida é perdida.



TEMPO: 175

TEMPORIZADOR

O **tempo** é um fator crucial durante cada partida, sendo este vital para a garantia da vitória. O ponto principal é conseguir completar todas as três fases dentro do tempo estipulado em que, caso não cumprido este quesito, o estado do Game é GAMEOVER. O jogador tem um **total de 3 minutos** para completar todos os níveis, lembrando que só é possível concluir um nível quando TODAS as coins forem coletadas. É válido ressaltar que esse tempo **não é renovado** em cada fase.



totalGameTime: Variável em Game que guarda o tempo restante. Começa em 180 segundos.

Game::UpdatePlaying(): A cada segundo, o totalGameTime diminui.

Game Over: Se o totalGameTime chega a zero, o jogo acaba.

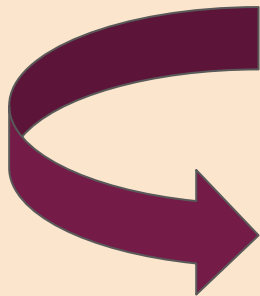
HUD: O tempo é exibido e muda de cor (laranja, depois vermelho piscando) conforme se esgota.

PONTOS: 000000

PONTUAÇÃO

No jogo, a **pontuação** mede o progresso do player progresso, em que:

- **Moedas:** Cada **coin** coletada vale **100 pontos**.
- **Fases Completadas:** Ao terminar um nível, é acrescido **bônus de 5.000 pontos**.



- **score:** Uma variável na classe Game guarda sua pontuação total.
- **Coleta de Moedas:** `score += 100`; adiciona pontos quando uma moeda é pega.
- **Completar Fase:** `score += 5000`; dá o bônus ao chegar no objetivo final do nível.

PLAYER



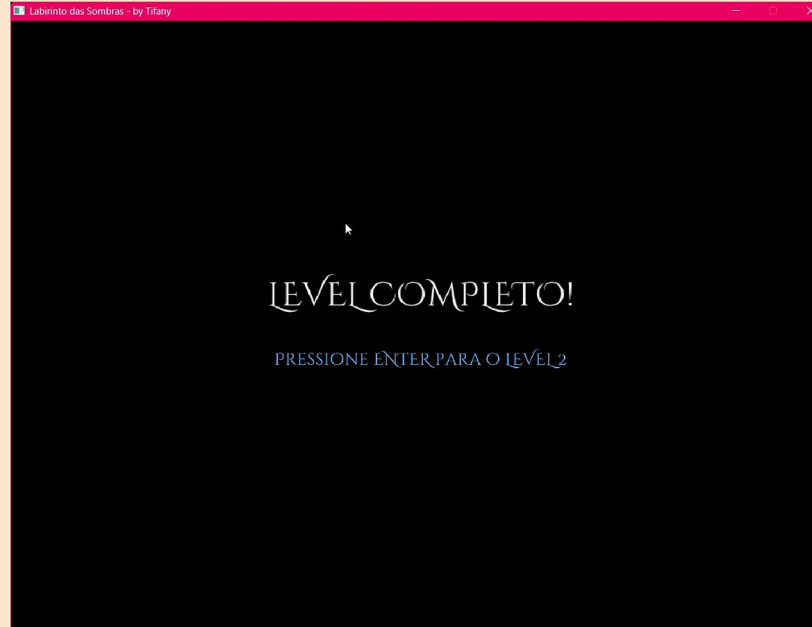
Representado por um quadrado liso azul, o player é o elemento conduzido pelo usuários pelas teclas UP, DOWN, RIGHT e LEFT, ou WASD. Quando o jogador colide contra a parede, A **invencibilidade** é um recurso de segurança. Após levar dano de uma parede, o jogador fica **invencível por 1.5 segundos**. Durante esse tempo, ele não pode sofrer mais danos, o que permite um respiro para se reposicionar e se localizar na partida, e evitar uma sequência rápida de perdas de vida e, por consequência, o estado de GAME OVER.

Dentro da classe **Player**, há a lógica responsável pela dinâmica deste elemento do jogo, como:

- **Vidas (lives):** Variável que guarda as vidas atuais, começando em 3 (player.Reset()) a define).
- **Velocidade (currentSpeed):** Controla o movimento; pode ser aumentada temporariamente por power-ups.
- **Update():** Move o jogador conforme a entrada do teclado (WASD ou setas) e o mantém dentro da tela. Também gerencia o tempo de duração dos power-ups, como o escudo e a velocidade.
- **Draw():** Desenha o jogador. Durante a invencibilidade, ele pulsa e muda de cor para indicar a proteção.
- **TakeDamage():** Reduz uma vida se o jogador não estiver invencível ou com escudo ativo. Inicia o invincibilityTimer.

→ **invincibilityTimer:** Variável que controla a duração da invencibilidade.

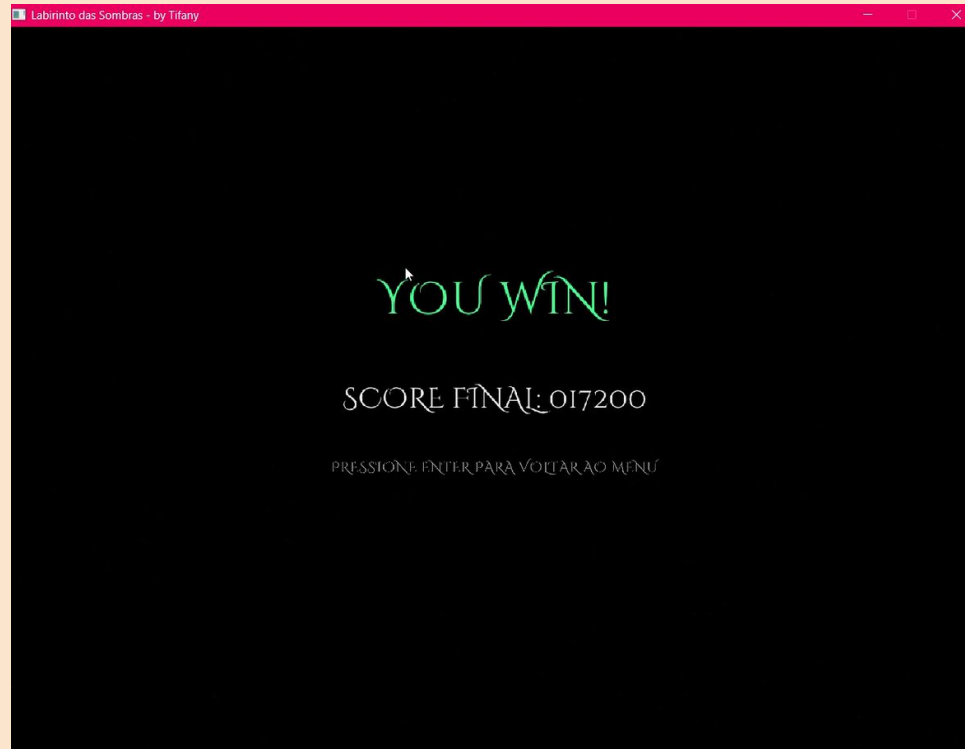
- **Ativação:** Definida para 1.5f segundos em Player::TakeDamage().
- **Contagem:** Decrementa em Player::Update() a cada frame.
- **Proteção:** Player::TakeDamage() só subtrai uma vida se invincibilityTimer e shieldTimer (do power-up de escudo) estiverem zerados.



Se o jogador conseguir todos os objetivos propostos dentro de uma fase, a tela de LEVEL COMPLETO é mostrada, indicando qual o próximo nível



Se o jogador não conseguir todos os objetivos propostos dentro de uma fase, a tela de GAME OVER é mostrada, indicando para retornar ao menu.



Ao completar as três fases propostas, a tela de YOU WIN é mostrada, juntamente com o score total e um indicador para retornar ao menú principal.