

HERO CITY



SOBRE O JOGO;

O jogo ocorre em uma biblioteca, com o usuário controlando um personagem, o "Nerd Man". O objetivo principal do jogo é coletar 5 itens (óculos, caderno, borracha, régua, lápis) e fugir de 4 meninas. Enquanto isso, o objetivo delas é beijar ele, mas como todo nerd, você precisa fugir delas.

No início do jogo, as 4 meninas estão correndo em movimentos aleatórios. Depois que o primeiro item é coletado, uma delas começa a seguir o nerd e assim sucessivamente, até as quatro seguirem ele e faltar somente um item para a vitória.



MOVIMENTAÇÃO DOS “FANTASMAS”;

*Persegução:

Utiliza o algoritmo de busca em largura (BFS) para calcular o caminho mais curto até o personagem principal. O fantasma segue esse caminho, contornando obstáculos para perseguir o jogador.

*Movimento aleatório:

1. Verifica direções possíveis:

O método checa se é possível andar para cima, direita, baixo ou esquerda (sem bater em parede) e armazena essas possibilidades no vetor direcoes_vilao.

2. Escolhe uma direção aleatória válida:

Sorteia uma direção (`temp = rand() % 4`). Repete o sorteio se a direção sorteada for oposta à última direção (para evitar ficar indo e voltando) ou se for inválida (parede ou bloqueio).

Se tentar 50 vezes e não conseguir, força o vilão a inverter a última direção.

3. Atualiza a última direção:

Salva a direção escolhida em `ultimo_aleatorios_viloes[vilao]` para usar na próxima rodada, evitando movimentos de vai-e-volta.

4. Verifica anticolisão:

Antes de mover, chama a função `pos_ocupada` para garantir que nenhum outro vilão está na posição de destino.

Só move se a posição estiver livre.

*`pos_ocupada()`:

Função auxiliar que verifica se a posição de destino já está ocupada por outro vilão, impedindo que dois fantasmas ocupem a mesma célula do mapa.

CARACTERÍSTICAS ADICIONAIS;

1. Telas de Interação

Imagens adaptadas para servir como interface entre usuário e programa.

Retângulos (`sf::IntRect`) definidos nas áreas dos botões para capturar cliques.

Detectção de clique:

`event.type == Event::MouseButtonPressed`

`event.mouseButton.button == Mouse::Left`

Posição do cursor obtida com `.getPosition()`.

Verificação se o clique ocorreu dentro do botão com `.contains()`.

Ações como `window.clear()` e `window.draw()` atualizam a tela.

*1.1 Menu Inicial

Exibe três opções ao jogador:

Jogar: inicia o jogo.

Créditos: mostra nomes, matrículas e retratos pixelados dos envolvidos.

Sair: fecha o jogo.

*1.2 Tela Game Over

Ativada automaticamente quando o personagem é capturado.

Duas opções:

Voltar ao menu: retorna ao menu inicial.

Jogar novamente: reinicia o jogo e aguarda novo clique para iniciar.

*1.3 Tela Win

Aparece quando o jogador coleta os 5 itens do mapa.

Indica a vitória com uma arte visual.

Botão de Retornar ao menu permite:

Jogar novamente.

Ver os créditos.

Sair do jogo.

CARACTERÍSTICAS ADICIONAIS;

2. Sprites

Personagens com sprites personalizados:

Jogador: menino nerd (calça, jaqueta e óculos).

Inimigas: quatro meninas com cabelos loiro, ruivo, moreno e castanho.

Ícone do jogo:

Sprite criado para representar o jogo na janela e na barra de tarefas.

Implementado com a função `.setIcon`.

Colisões e interações:

Utilização de `.getGlobalBounds()` para obter área do sprite.

Verificação de colisão com `.intersects()`.



3. Portais

Objetivo: Tornar a jogabilidade mais dinâmica.

Localização: Posicionados em cantos específicos do mapa (biblioteca).

Funcionamento:

Cada portal possui uma cor correspondente.

O jogador é teletransportado para o portal de mesma cor.

As meninas não utilizam portais.

Detecção de uso do portal:

Retângulo com as dimensões do portal verifica colisão.

Quando a sprite do personagem intersecta o retângulo:

A interação é confirmada.

A posição é atualizada para o portal correspondente.

Controle de tempo (cooldown):

Utilização de `.getElapsedTime()` para contar o tempo após teleporte.

Um intervalo de 0.4 segundos impede loop de teletransporte.

Atualização da direção após o teleporte:

Evita que o personagem colida com paredes ao sair do portal.

A direção é ajustada automaticamente para onde há caminho livre.

Garante movimentação fluída e dinâmica entre portais.

Lógica semelhante à dos botões interativos usados nas telas.

CARACTERÍSTICAS ADICIONAIS;

4. Coletáveis

Objetivo do jogo: Vencer ao coletar os 5 itens do mapa.

Itens coletáveis:

Lápis, borracha, livro, régua e óculos.

Geração dos itens:

Cada item aparece um por vez.

Posições fixas, mas ordem aleatória a cada partida.

Ordem definida usando random_shuffle() com índice de 0.

Detecção de coleta:

Baseada em interseção da sprite do personagem com a área do item (retângulo).

Controle feito com vetor (std::vector) de booleanos:

Cada item começa com false (não coletado).

Ao ser coletado, é marcado como true.

Se sorteado novamente, é ignorado.

Feedback visual (score):

Miniaturas dos itens aparecem no topo da tela ao serem coletados.

Mostra quais e quantos itens já foram adquiridos.

5. Música e Efeitos Sonoros

Implementação:

Utilizada a biblioteca de áudio da SFML.

Arquivos de som reagem às ações do jogador.

Eventos com som

Início do jogo:

Ao clicar em "Jogar", inicia uma música de fundo contínua.

Coleta de item:

Emite efeito sonoro que reforça a ação para o jogador.

Captura pelo inimigo (Game Over):

A música de fundo é interrompida.

Toca som de beijo.

Tela fica estática por 1.5 segundos durante o som.

Após isso, exibe a tela de derrota.

Vitória (Coleta dos 5 itens):

A música de fundo é interrompida.

Toca efeito sonoro de vitória.

Tela muda para a tela de Win.

PARTES DO TRABALHO E O RESPONSÁVEL;

O aluno Paulo Henrique foi o responsável por confeccionar a sprite do mapa e produzir a matriz de impacto. Neste processo, ele contou com a ajuda de inteligências artificiais para a criação de imagens semelhantes a uma biblioteca. Depois de um longo período (sem sucesso) de gerar a imagem via IA, ele decidiu pegar uma imagem pré-pronta e adaptá-la para as demandas de seu grupo. Essa adaptação foi feita via piskel.com que é um software responsável pela confecção de pixels arts. O mesmo também foi o autor dos slides.

O aluno Thales Neves foi o responsável pela criação dos sprites dos personagens e dos “fantasmas”, também foi o autor de todas as telas interativas (menu, créditos, tela game over, tela win) e atribuiu os efeitos sonoros para o jogo.

O aluno Luiz Filipe foi o responsável por implementar a movimentação dos “fantasmas” e por juntar os códigos que foram feitos separadamente por cada membro do grupo.



OBRIGADO
PELA
ATENÇÃO