
Relatório de Desenvolvimento do Programa Releitura do jogo Pac-Man com SFML

Lista de figuras

Figura 1: Mapa do jogo	2
Figura 2: Bloco de funções	4
Figura 3: Bloco de funções	4

Sumário

1. Introdução	2
1.1. Ambiente de desenvolvimento	2
2. Descrição de programa	2
2.1. Funcionalidades implementadas	3
2.2. Fluxo de execução	3
2.3. Exemplo de uso	3
3. Funções auxiliares	3
3.1. canMove(int newX, int newY):	3
3.2. handleTeleport()	3
3.3. checkForCoin()	3
3.4. moveEnemy(Enemy& enemy, int pacManX, int pacManY)	3
3.5. checkCollisionWithPacMan(const Enemy& enemy)	3
4. Bibliotecas utilizadas.....	4
5. Conclusão	4
6. Anexos	4

1. Introdução

Este documento apresenta o desenvolvimento de um jogo de labirinto em C++ utilizando a biblioteca SFML. O jogo é uma releitura do clássico Pac-Man, onde o jogador controla um guerreiro em um labirinto, coletando moedas enquanto é perseguido por três tipos de inimigos: slime, fantasma e ciclope. Este projeto foi desenvolvido como parte do trabalho prático para a disciplina de Programação, com o objetivo de explorar conceitos de programação em C++ e manipulação gráfica com SFML.

1.1 Ambiente de desenvolvimento

Para a elaboração do código o autor deste trabalho optou pela utilização da IDE (Integrated Development Environment) Visual Studio Code, realizado e testado no sistema operacional Windows 10 Home Single Language versão 22H2.

2. Descrição do programa

O jogo segue a estrutura básica do Pac-Man, onde o jogador controla um guerreiro em um labirinto cheio de moedas e é perseguido por três inimigos. Abaixo, estão descritas as funcionalidades implementadas e o fluxo de execução do jogo.



Figura 1: Mapa do jogo

2.1 Funcionalidades implementadas

Movimentação do Jogador

- O jogador movimenta o guerreiro usando as setas direcionais. O movimento é restrito pelas paredes do labirinto e o guerreiro pode se teletransportar em áreas específicas do mapa.

Coleta de Moedas

- O jogador coleta moedas que aparecem no labirinto, aumentando sua pontuação.

Movimentação dos Inimigos

- Os inimigos se movem em direção ao guerreiro de acordo com algoritmos específicos para seguir o jogador.

Verificação de Colisões

- O jogo detecta colisões entre o guerreiro e os inimigos, encerrando o jogo se o guerreiro for pego.

2.2 Fluxo de execução

- **Inicialização:** O jogo carrega texturas e sons necessários, inicializa variáveis e posiciona o guerreiro e inimigos no labirinto.
- **Entrada do Usuário:** O jogo aguarda a entrada do usuário para movimentar o guerreiro.
- **Atualização do Jogo:** O estado do jogo é atualizado com base nas entradas do usuário e no movimento dos inimigos.
- **Renderização:** O jogo desenha o labirinto, o guerreiro, os inimigos e a pontuação na tela.
- **Finalização:** O jogo termina quando todas as moedas são coletadas ou o guerreiro é pego pelos inimigos.
-

2.3 Exemplo de uso

1. O jogador inicia o jogo e visualiza o labirinto com o guerreiro e os inimigos.
2. O jogador usa as setas direcionais para mover o guerreiro.
3. O guerreiro coleta moedas e evita os inimigos.
4. O jogo exibe a pontuação e o tempo decorrido.
5. O jogo termina quando todas as moedas são coletadas ou o guerreiro é capturado pelos inimigos.

3. Funções Auxiliares

3.1 canMove(int newX, int newY):

- Verifica se o movimento para uma nova posição é válido, ou seja, se não colide com paredes ou obstáculos.

3.2 handleTeleport()

- Gerencia o teletransporte do guerreiro em áreas específicas do labirinto.

3.3 checkForCoin()

- Verifica se o guerreiro coleta uma moeda e atualiza a pontuação.

3.4 moveEnemy(Enemy& enemy, int pacManX, int pacManY)

- Move o inimigo em direção ao guerreiro, considerando a distância horizontal e vertical.

3.5 checkCollisionWithPacMan(const Enemy& enemy)

- Verifica se o inimigo colidiu com o guerreiro.

4. Bibliotecas utilizadas

O programa utiliza as seguintes bibliotecas da linguagem C++:

- **SFML**: Para manipulação gráfica e áudio.
- **iostream**: Para entrada e saída de dados.
- **cstdlib**: Para funções utilitárias.
- **cmath**: Para operações matemáticas.

5. Conclusão

O desenvolvimento deste jogo proporcionou uma oportunidade para aplicar conceitos de programação em C++ e utilizar a biblioteca SFML para gráficos e áudio. As funcionalidades implementadas demonstram a capacidade de criar um jogo básico com movimentação de personagens, coleta de itens e inteligência artificial para inimigos. A estrutura do jogo permite uma experiência interativa e foi projetada para ser extensível para futuras melhorias e adições.

6. Anexos

```
bool canMove(int newX, int newY) {
    if (newX < 0 || newX >= 17 || newY < 0 || newY >= 15) return false;
    char tile = mapa[newY][newX];
    return tile != '1' && tile != '2' && tile != '3' && tile != '4' && tile != '8' && tile != '9';
}

// Função para lidar com o teleporte
void handleTeleport() {
    if (posx == 0 && posy == 7) { posx = 15; dirX = 1; dirY = 0; }
    else if (posx == 15 && posy == 7) { posx = 0; dirX = -1; dirY = 0; }
    else if (posx == 7 && posy == 0) { posx = 7; posy = 14; dirX = 0; dirY = 1; }
    else if (posx == 8 && posy == 0) { posx = 8; posy = 14; dirX = 0; dirY = 1; }
    else if (posx == 7 && posy == 14) { posx = 7; posy = 0; dirX = 0; dirY = -1; }
    else if (posx == 8 && posy == 14) { posx = 8; posy = 0; dirX = 0; dirY = -1; }
}

// Função pontos e moedas
void checkForCoin() {
    if (mapa[posy][posx] == '0') {
        mapa[posy][posx] = '7';
        --moedas;
        pontos += 100;
    }
}
```

Figura 2: Bloco de funções

```

void moveEnemy(Energy& enemy, int pacManX, int pacManY) {
    int diffX = pacManX - enemy.posx;
    int diffY = pacManY - enemy.posy;

    // Tentar se mover na direção do Pac-Man
    if (std::abs(diffX) > std::abs(diffY)) {
        if (diffX > 0 && canMove(enemy.posx + 1, enemy.posy)) {
            enemy.posx += 1;
        } else if (diffX < 0 && canMove(enemy.posx - 1, enemy.posy)) {
            enemy.posx -= 1;
        } else if (diffY > 0 && canMove(enemy.posx, enemy.posy + 1)) {
            enemy.posy += 1;
        } else if (diffY < 0 && canMove(enemy.posx, enemy.posy - 1)) {
            enemy.posy -= 1;
        }
    } else {
        if (diffY > 0 && canMove(enemy.posx, enemy.posy + 1)) {
            enemy.posy += 1;
        } else if (diffY < 0 && canMove(enemy.posx, enemy.posy - 1)) {
            enemy.posy -= 1;
        } else if (diffX > 0 && canMove(enemy.posx + 1, enemy.posy)) {
            enemy.posx += 1;
        } else if (diffX < 0 && canMove(enemy.posx - 1, enemy.posy)) {
            enemy.posx -= 1;
        }
    }
}

// Função para verificar colisão com Pac-Man
bool checkCollisionWithPacMan(const Energy& enemy) {
    return enemy.posx == posX && enemy.posy == posy;
}

```

Figura 3: Bloco de funções