RELATÓRIO DO JOGO PAC-MINE

Algorítmos e programação - CIC

Gabriel Borges Gambim - 00330012
Lucas Dall

Observações importantes:

Sharingan é o equivalente à esmeralda;
Marca da maldição é equivalente ao ouro;
Byakugan é o poder de enxergar tudo (poção).

ESTRUTURAS:

struct POSICAO:

Organiza uma coordenada X e uma coordenada Y e será usada para identificar a posição do jogador.

struct TOUPEIRA:

Organiza as coordenadas X e Y, valores referentes ao deslocamento no eixo X e Y, um contador de movimentos e um booleano para estado vivo/morto.
Será usada para manipular os inimigos durante o jogo.

struct PONTOS:

Organiza os dados de vida, esmeraldas e ouros coletados e inimigos mortos.

struct ESTADO:

Organiza uma POSICAO, o nível do jogo e PONTOS. Será usada para o salvamento do jogo.

FUNÇÕES

void SetupWindow():

Faz a inicialização da tela gráfica e do áudio.

int podeMoverJ (int posX, int posY, int desX, int desY, char mapa[][]):

Recebe a posição e o deslocamento do jogador nas coordenadas X e Y, além de uma matriz que possui o mapa do jogo.
Esta função percorre o mapa e, caso encontre algum caractere '#' ou 'S', verifica se o jogador ocupará a mesma posição se for realizado o deslocamento. Caso vá ocupar a mesma posição, retorna o número 0, que significa que a movimentação não pode ser realizada. Caso contrário, retorna 1.

int podeMoverT(int posX, int posY, int desX, int desY, char mapa[][]):

Recebe a posição e o deslocamento dos inimigos nas coordenadas X e Y, além de uma matriz que possui o mapa do jogo.
Esta função percorre o mapa e, caso encontre algum caractere '#', verifica se o inimigo ocupará a mesma posição se for realizado o deslocamento. Caso vá ocupar a mesma

posição, retorna o número 0, que significa que a movimentação não pode ser realizada. Caso contrário, retorna 1.

void move (int desX, int desY, int *posX, int *posY):
Recebe o deslocamento em X e Y, além de um ponteiro para as coordenadas em X e Y.
Esta função realiza a movimentação do jogador.

void Tmove (TOUPEIRA *Tponteiro):
Recebe um ponteiro para uma variável TOUPEIRA e altera as coordenadas X e Y acessando os valores de deslocamento e posição do inimigo.

int upload_mapa (char nome_arq_mapa[], char matriz[][], POSICAO *posicao_do_jogador, TOUPEIRA *toupeiras):
Recebe o nome de um arquivo texto, uma matriz, um ponteiro para a variável POSICAO do jogador e um ponteiro para uma variável TOUPEIRA.
Abre o arquivo e, enquanto o arquivo não chega ao final, preenche a matriz com os dados do arquivo. Caso encontre o caractere 'J', atribui a posição ao jogador. Caso encontre um caractere 'T', atribui a posição a uma variável TOUPEIRA.

int save(char nome_arq_mapa[], POSICAO posicao): salva o jogo em um arquivo. → Não funciona

int colisao (int JposX, int JposY, int posX_bloco, int posY_bloco):
Recebe dois pares de coordenadas X e Y (quatro inteiros) e avalia se as coordenadas X são iguais E as coordenadas Y são iguais. Caso sejam, a função retorna 1 para indicar que houve colisão. Caso contrário, retorna 0 para indicar que não houve.

void AmbientSound(Sound Ambiente):
Recebe uma variável do tipo Sound e toca o som se ele não estiver sendo tocado.

void abreMenu(): abre o menu do jogo. → Não funciona

void remove_do_mapa (char mapa[][], int posX, int posY):
Recebe um ponteiro para matriz e coordenadas X e Y. Realiza uma troca de valor dentro da matriz, trocando o caractere que tiver na posição mapa[posY][posX] por ' '.
Serve para controlar a quebra de blocos e a aquisição de esmeraldas e ouro.

<u>int main()</u>

Definições de variáveis:

```
218  //DEFINIÇÕES
219      int posX, posY, i = 0, t = 0, ultimo_move, ctrl_menu = 1, ctrl_nivel;
220      char mapa[NUM_BLOC_ALTURA_MAPA][NUM_BLOC_LARGURA_MAPA] = {0}, bloco, m1 = 0, m2 = 0;
221      char status[4][25] = {"Vida: ", "Esmeraldas: ", "Ouro: ", "Inimigos eliminados: "};
222      bool power_eye = false, pause = false, sair_jogo = 0;
223
224      FILE *ponteiro_mapa1, *ponteiro_mapa2, *ponteiro_mapa3;
225      TOUPEIRA toupeiras[TOUPEIRA_MAX];
226      PONTOS jogador = {jogador.vida = 3, jogador.contador_esmeralda = 0, jogador.contador_ouro = 0, jogador.inimigos_eliminados = 0};
227      PONTOS *Jponteiro = &jogador;
228      ESTADO jogador_estado;
229      POSICAO posicao_do_jogador;
230      Rectangle parede;
231      Rectangle parede2;
232      Rectangle toupeira_rec;
```

Inicializações: chama a função que inicia a tela gráfica e o áudio e faz o upload da música de fundo e das texturas utilizadas.

```
234     //INICIALIZAÇÕES
235         SetupWindow();
236
237     //CARREGAMENTO DA MÚSICA DE FUNDO
238         Music Ambiente;
239         Ambiente = LoadMusicStream("./Som/fundo.mp3");
240         PlayMusicStream(Ambiente);
241         SetMusicVolume(Ambiente, VOLUME);
242
243     //TEXTURAS
244         Texture2D Orochimaru = LoadTexture("./Texturas/Orochi.png");
245         Texture2D OrochimaruPower = LoadTexture("./Texturas/OrochimaruPower.png");
246         Texture2D Sharingan = LoadTexture("./Texturas/Sharingan.png");
247         Texture2D Byakugan = LoadTexture("./Texturas/Byakugan.png");
248         Texture2D Marca_da_maldicao = LoadTexture("./Texturas/Marca_da_maldicao.png");
249         Texture2D Bloco_fundo = LoadTexture("./Texturas/Bloco_fundo.png");
250         Texture2D Bloco_pedra = LoadTexture("./Texturas/Bloco_pedra.png");
```

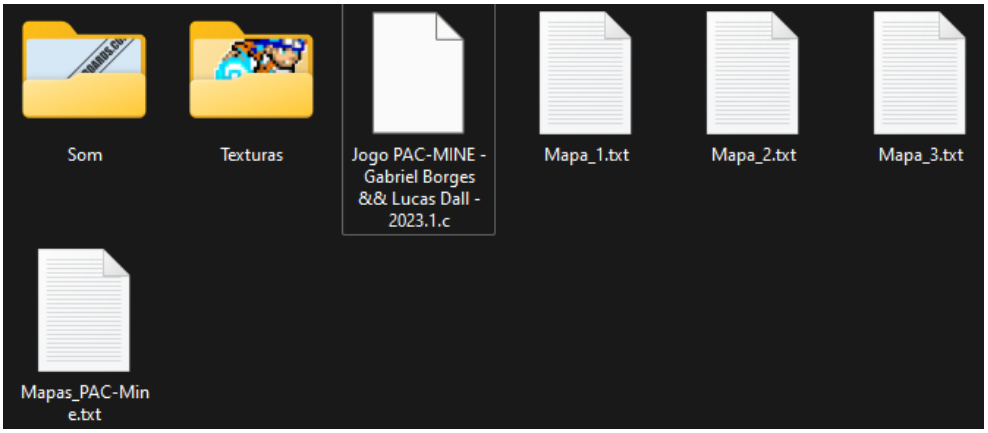
Leitura do arquivo mapa, preenchimento da matriz mapa, aquisição da posição do jogador e dos inimigos, além da inicialização do tiro.

```
252     //LEITURA DO MAPA
253         if((upload_mapa("Mapa_1.txt", mapa, &posicao_do_jogador, &toupeiras[0])!= 1)){
254            printf("Erro na leitura do mapa 1\n");
255         }
256
257     //INICIALIZAÇÃO DA POSIÇÃO DO JOGADOR
258         posX = posicao_do_jogador.posX*ARESTA_BLOCO;
259         posY = posicao_do_jogador.posY*ARESTA_BLOCO;
260
261     //INICIALIZAÇÃO DA POSIÇÃO DAS TOUPEIRAS CONFORME O MAPA E DESLOXAMENTO ALEATÓRIO
262         for (t=0; t < TOUPEIRA_MAX; t++){
263                     toupeiras[t].TposX = toupeiras[t].TposX * ARESTA_BLOCO;
264                     toupeiras[t].TposY = toupeiras[t].TposY * ARESTA_BLOCO;
265                     toupeiras[t].vida = 1;
266         }
267         do{
268             toupeiras[t].TdesX = GetRandomValue(-1, 1) * ARESTA_BLOCO;
269             toupeiras[t].TdesY = GetRandomValue(-1, 1) * ARESTA_BLOCO;
270         }while(toupeiras[t].TdesX == 0 && toupeiras[t].TdesY == 0);
271
272     //INICIALIZAÇÃO DO PROJÉTIL (TIRO)
273         Vector2 Tiro_pos = {0,0};
274         bool Tiro_ativo = false;
```

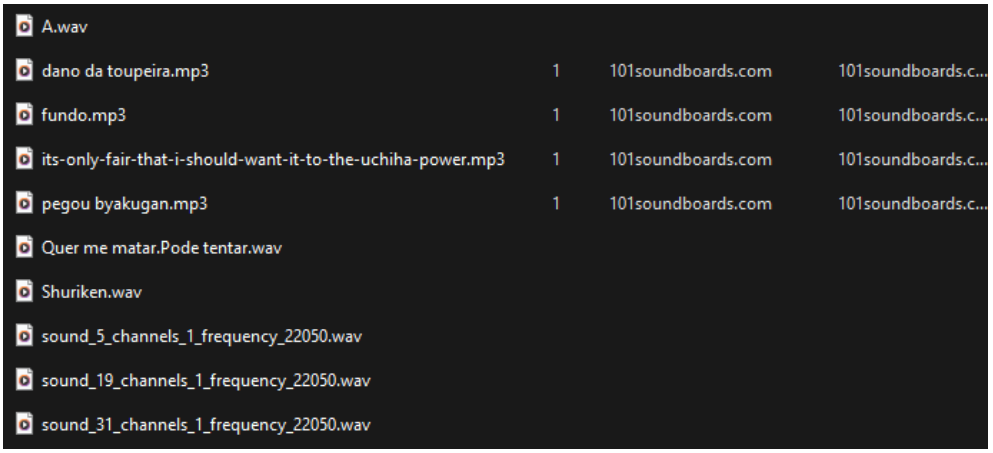
O restante do código está comentado.

Os inimigos foram representados por vetores de structs, enquanto o jogador foi representado principalmente por struct, embora tenham sido utilizadas outras representações para uso visual, como Vector2 e Rectangle.

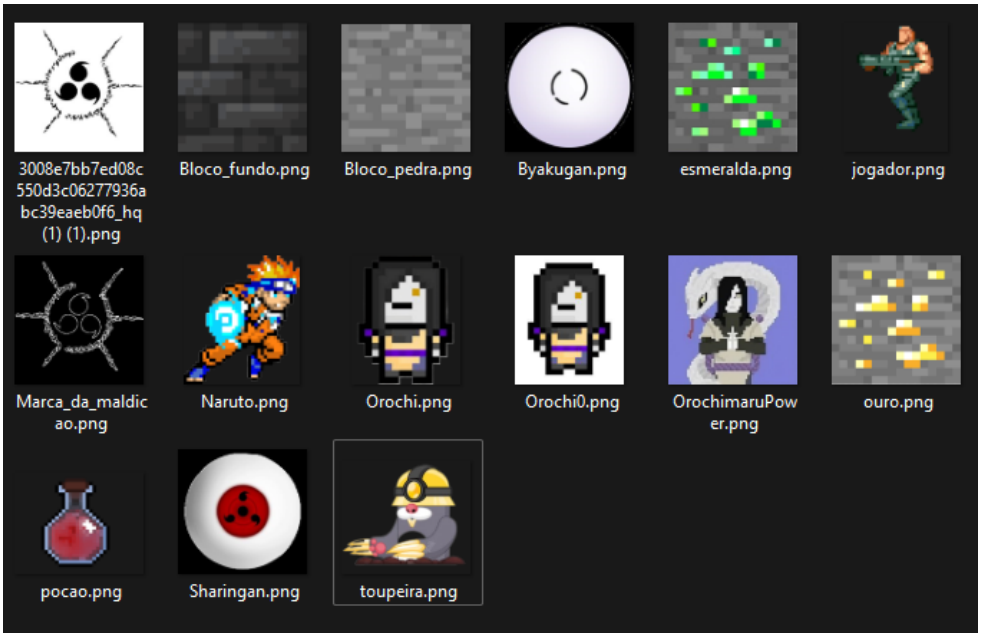
ORGANIZAÇÃO DOS ARQUIVOS DO JOGO



PASTA SOM



PASTA TEXTURAS



CONCLUSÃO

Foi divertido passar por cima das dificuldades, mesmo que tenha sido aos poucos, e personalizar o jogo com um tema que gosto bastante. A parte mais difícil foi a criatividade. Embora não seja um trabalho artístico, a criatividade pesa muito e é ela que quebra os limites da programação.
Nunca imaginei que teria capacidade de fazer um jogo, principalmente porque é o meu primeiro contato com “uma conversa com uma máquina”. Foi muito ver que, embora eu seja visivelmente limitado quando comparado com meus colegas programadores, eu consigo fazer algo (parcialmente) funcional também!