Documentação - Trabalho Prático

"Final Fantasy"

Universidade Federal de Minas Gerais

Programação e Desenvolvimento de Software I

09/2021

Ítalo Dell Areti

1 - Introdução

O Trabalho Prático sugerido para a disciplina de PDS1 foi baseado na criação de um jogo utilizando a linguagem de programação C e a biblioteca Allegro, que foi apresentada para os alunos durante a execução do curso. Através desta biblioteca, fomos possibilitados a desenvolver um jogo com maior acesso a recursos e ferramentas para a estruturação de elementos que não era possível anteriormente, sem a utilização de cores, temporizadores e outros artifícios.

O jogo é do estilo RPG e tem como objetivo chegar na saída do mapa. Durante o trajeto possui vários inimigos escondidos em posições aleatórias que vão atacar o personagem e entrar no modo de batalha. Existe dois modos de jogo, o modo exploração e modo de combate. O modo de exploração é basicamente o personagem no mapa e a saída. O outro modo é o de batalha, que consiste em no personagem e o inimigo. Neste modo, existe um menu de opções para execução, atacar, especial e fugir. A cada vez que o personagem mata um inimigo ele ganha pontos e retorna ao modo de exploração. Ao final do jogo, quando chega a saída, ele recebe a pontuação de acordo com a quantidade de inimigos mortos.

2- Implementação

2.1 Movimentação e funcionalidade

A movimentação é realizada apenas com as setas do teclado e a o botão de confirmação é o ENTER. No modo de exploração vamos utilizar apenas as setas, e no modo de batalha vamos utilizar as setas para cima e baixo para escolher umas das três opções, atacar, especial e fugir, podendo usar a tecla ENTER para selecionar qualquer uma dessas opções.

2.2 Funções e Procedimentos importantes

As entidades do jogo foram declaradas como structs para o melhor arranjo do sistema. Dentre elas, estão:

Monstro: que possui seu par ordenado da localização (x, y), possui uma cor, vida, vida máxima (que será explicada adiante) e seu dano.

Herói: que possui seu par ordenado da localização (x, y), possui uma cor, vida, pontuação, dano, suas ações e sua posição antes de entrar no modo de batalha

Projetil: que possui seu par ordenado da localização (x, y), dano, velocidade e ativo (0 ou 1 para o disparo).

A função responsável por criar tipo de monstro.

```
void inicializaTipo(Monstro *m, int tipo)
{
    m->vida_max = 100 + tipo * 50;
    m->vida = m->vida_max;
    m->dano = 5 + tipo * 3;
    m->COR = CORES[tipo];
}
```

Como é exigido que exista mais de um tipo de inimigo, utilizei a seguinte metodologia. Escolhi 4 cores, amarela, azul, preta e azul, as posições do vetor de 0 a 3 respectivamente. Então dependendo do tipo do monstro, tem somado a sua vida base a multiplicação de seu tipo por 50. O mesmo acontece com o dano causado pelo inimigo, sendo 5 o dano base e somado o tipo multiplicado por 3.

As funções responsáveis pela geração, organização e tipos de monstros.

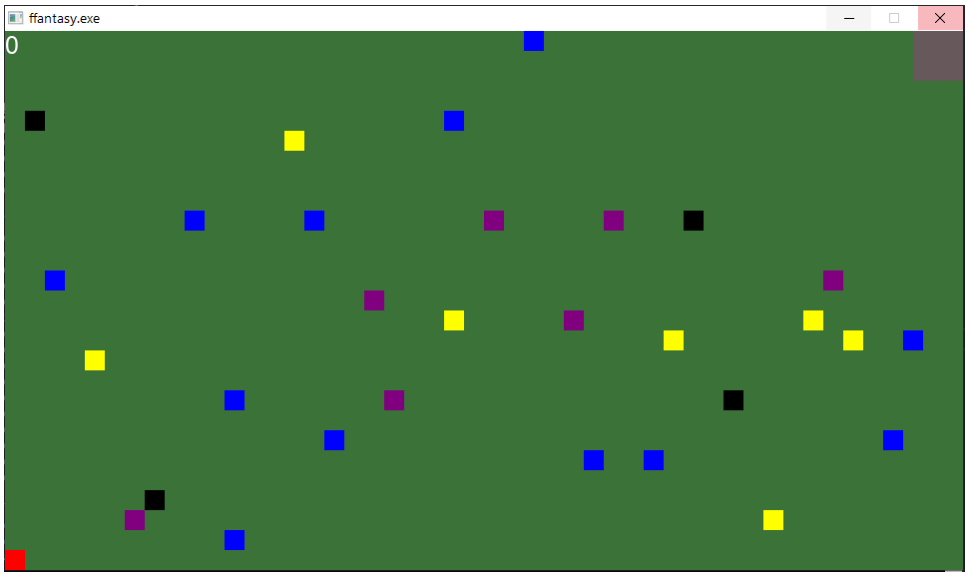
```
int inteiroAleatorio(int min, int max)
{
    return min + rand() % (max - min + 1);
}

int multiploMaisProximo(int num, int mult)
{
    return roundf((float)num / mult) * mult;
}

void initMonstros(Monstro monstros[MAX_MONSTROS])
{
    srand(time(NULL));
    for (int i = 0; i < MAX_MONSTROS; i++)
    {
        monstros[i].x = multiploMaisProximo(inteiroAleatorio((SCREEN_W - TAM_HEROI) / MAX_MONSTROS * i,
                                                             (SCREEN_W - TAM_HEROI) / MAX_MONSTROS * (i + 1)),
                                            PASSO);
        monstros[i].y = multiploMaisProximo(inteiroAleatorio(0, SCREEN_H - TAM_HEROI), PASSO);
        inicializaTipo(&monstros[i], inteiroAleatorio(0, 3));
    }
}
```

A inteiroAleatório é responsável por gerar os números aleatórios que serão utilizados como coordenadas dos monstros.

A função multiploMaisProximo possui como tarefa o arredondamento para próximo dos múltiplos do tamanho do passo do personagem, que é 20. Dessa forma, os inimigos ficam mais fáceis de serem encontrados, ficando alinhados pra colidir com o herói e um pouco mais espalhados no mapa.

A initMonstros percorre o vetor monstros, atribuindo as coordenadas e arredondando elas e atribuindo o tipo do monstro, até que se tenha o máximo de monstros possíveis para a resolução do jogo. A cada vez que o jogo for executado novamente, a distribuição será diferente também.

Um exemplo da distribuição dos inimigos no mapa.

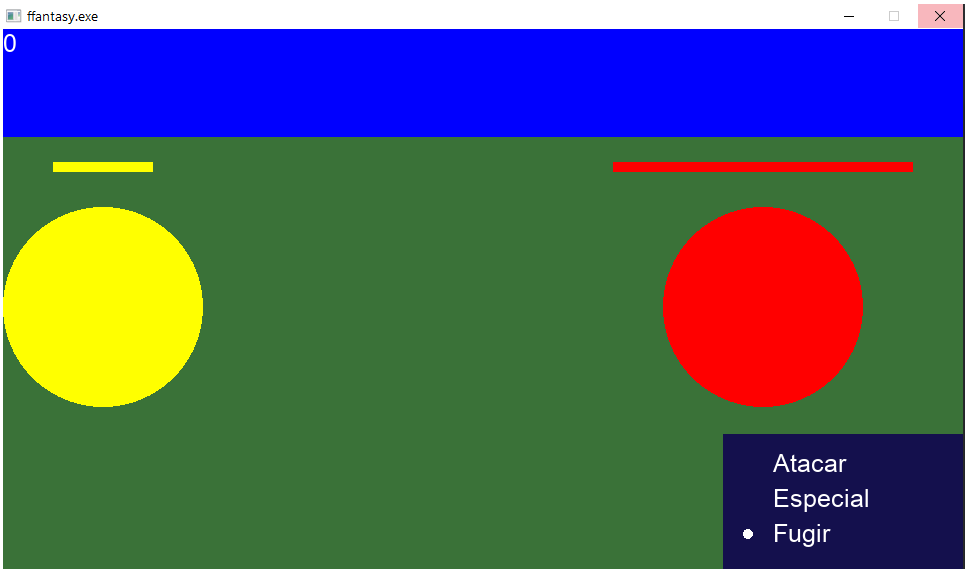


A pontuação indicada no lado superior esquerdo é calculada da seguinte forma:

vida máxima do inimigo + vida máxima do herói - atual

Sendo que a vida máxima é a vida base + bonificação do tipo.

Cenário de batalha.



A vida dos personagens é proporcional a barra da vida, quando maior a vida, maior a barra. O personagem possui vida fixa em 300 e a vida do inimigo é variável de acordo com os 4 tipos de inimigo. O ataque possui do herói possui dano fixo de 25 e o especial possui 33% de chance de ter o dano a baixo de 25, 33% de chance de ter o dano igual a 25 e 33% de ter um dano bem acima de 25. O efeito visual do ataque é uma bola e, quanto maior o dano, maior é o raio do círculo do ataque. A porcentagem de fuga do herói é 80%, sendo 20% de falha.

Ao final, após chagar ao ponto de saída, é printado na tela a mensagem “você venceu”, sua pontuação e o recorde de pontos. Este recorde é salvo no arquivo recorde.dat, e só é alterado quando uma nova pontuação superar o recorde armazenado.

Outras funções importantes, em sua grande maioria são responsáveis por processar a ação do jogador ou herói e realizar o desenho e a cor de cada elemento. Algumas delas são:

void desenhaPontuacao(função responsável por mostrar todo o tempo a pontuação do jogador)

void desenhaVitoria(Mostra a mensagem de vitória e a pontuação final após vencer)

int detectouMonstro(Detecta o monstro se ele não estiver morto ou se encontrar com o herói)

void processaTeclaBatalha(fica verificando quando uma das 3 opções foram selecionadas).