

INSTITUTO POLITECNICO DE BEJA

Escola superior de Tecnologia e Gestão

Licenciatura de Engenharia Informática

Sistemas Operativos

Trabalho Prático

Relatório Técnico do

Jogo “***Hunter Hallow***”

Rute Figueiredo, nº

Tiago Campos, nº 13953

25 de Novembro de 2017

***Índice***

[Introdução 3](#_Toc499419599)

[Mecânica do Jogo 4](#_Toc499419600)

[1.1. História do Jogo 4](#_Toc499419601)

[1.2. Mapa do jogo 4](#_Toc499419602)

[1.3. Objetivos do jogo 4](#_Toc499419603)

[1.4. Super User 4](#_Toc499419604)

[Desenvolvimento do Jogo 5](#_Toc499419605)

[1.1. Primeira Fase do Desenvolvimento 5](#_Toc499419606)

[1.1.1. Definição das Estruturas 5](#_Toc499419607)

[1.1.2. Inicialização do Jogador e do Monstro 5](#_Toc499419608)

[1.1.3. Inicialização do Mapa no código 5](#_Toc499419609)

[1.1.4. Inicialização dos Itens e dos Tesouros no código 5](#_Toc499419610)

[1.2. Segunda Fase do Desenvolvimento 5](#_Toc499419611)

[1.2.1. Inicialização do Mapa através da leitura do Ficheiro 5](#_Toc499419612)

[1.2.2. Inicialização dos Itens e dos Tesouros a partir de um Ficheiro 5](#_Toc499419613)

[1.2.3. Desenvolvimento da Função para fazer o Jogador andar no Mapa 5](#_Toc499419614)

[1.2.4. Desenvolvimento da Função para fazer os Monstros andarem no Mapa 5](#_Toc499419615)

[1.2.5. Desenvolvimento da Função para a Batalha 5](#_Toc499419616)

[1.3. Terceira Fase do Desenvolvimento 5](#_Toc499419617)

[1.3.1. Permitir o Jogador apanhar os Itens e os Tesouros 5](#_Toc499419618)

[1.3.2. Implementar o Save e o Load 5](#_Toc499419619)

[1.3.3. Revisão do Código e Correção de Bugs 5](#_Toc499419620)

[1.3.4. Melhorar API 5](#_Toc499419621)

[Conclusão 6](#_Toc499419622)

[Referencias 7](#_Toc499419623)

[Anexos 8](#_Toc499419624)

Indice de Figuras

# ***Introdução***

# ***Mecânica do Jogo***

* 1. ***História do Jogo***
  2. ***Mapa do jogo***
  3. ***Objetivos do jogo***
  4. ***Super User***

***Desenvolvimento do Jogo***

* 1. ***Primeira Fase do Desenvolvimento***
     1. *Definição das Estruturas*

**Player**

struct Player {

char namePlayer[MAX\_NAME];

int energyPlayer;

int damage;

int critic;

int cellPlayer;

int itemPlayer;

int treasurePlayer;

};

**Monster**

struct Monster {

char nameMosnter[MAX\_NAME];

int lifeMonster;

int damageMonster;

int criticMonster;

int cellMonster;

int itemMonster;

int treasureMonster;

int nMonsters;

};

**Item**

struct Item {

int CodItem;

char NameItem[MAX\_NAME];

int DamageItem;

int CriticItem;

int PositionItem;

int LifeItem;

};

**Treasure**

struct Tresure {

int CodTresure;

char NameTreasure[MAX\_NAME];

int Gold;

int PositionTresure;

};

**Cell**

struct Cell {

int north;

int south;

int east;

int west;

int up;

int down;

int treasureCell;

int itemCell;

char descriptionCell[MAX\_DESCRIPTION\_CELL];

};

**Map**

struct Map

{

struct Cell cell[MAX\_CELLS];

struct Item item[MAX\_ITEMS];

struct Tresure treasure[MAX\_TREASURE];

int nCells;

};

**SaveGame**

struct SaveGame {

struct Monster saveMonster;

};

* + 1. *Inicialização do Jogador e do Monstro*
* A inicialização do Jogador é feita através do método *InsertPlayer*, que pede o nome do jogador e inicializa-o com esse nome e as características que estão pré-definidas para o início do jogo (Energy=100, Damage=50, Critic=10). No caso do jogador inserir um dos comandos de SuperUser em vez do seu nome, o jogador recebe valores muito mais generosos que vão facilitar bastante o jogo (Energy=10000, Damage=1000, Critic=200, Item=Gold Armor + Sword, Treasure=Cofre Grande).
* A inicialização do Monstro é feita através da função InicializeMonster. São inicializados 5 monstros que permanecem sempre nas mesmas salas e outros 5 que se movimentam de forma aleatória pelo mapa. Um dos monstros é o que vai definir se o jogo é ganho ou não, porque é apenas depois deste ser derrotado que o jogador ganha.
  + 1. *Inicialização do Mapa no código*

Para a inicialização do Mapa inicialmente foi criada uma função (*InitMap*) que inicializasse o mapa estando o mesmo definido nessa mesma função.

* + 1. *Inicialização dos Itens e dos Tesouros no código*
* Inicialmente foi criada uma função que inicializa os itens e os tesouros a partir do código – *InitItemPlusTreasure*
  1. ***Segunda Fase do Desenvolvimento***
     1. *Inicialização do Mapa através da leitura do Ficheiro*

Foi criada uma função (*LoadMapFromFile*) que vai descarregar as características do mapa a um ficheiro com o nome “map.txt” com a seguinte estrutura:

*north south west east up down item treasure*

*descriptionCell*

* + 1. *Inicialização dos Itens e dos Tesouros a partir de um Ficheiro*

Foram criadas duas funções para carregarem os itens e os tesouros a partir de ficheiros “.txt” – *InitObejectItem* e *InitObejectTreasure*

Após isso ainda foi criada outra função para carregar os itens a partir de um ficheiro binário com o nome “objectos.dat” - *InitObejectItemBin*

* + 1. *Desenvolvimento da Função para fazer o Jogador andar no Mapa*

A função criada para definir o movimento do jogador no mapa é a função *PlayerWalk.* Esta função apresenta ao jogador uma lista de opções:

1 – Norte

2 – Sul

3 – Oeste

4 – Este

5 – Subir

6 – Descer

7 – Save

8 – Menu Principal

Consoante o que o jogador escolher pode-se movimentar pelo mapa (caso exista uma forma de passagem na direção escolhida), gravar o jogo ou ir para o menu principal.

* + 1. *Desenvolvimento da Função para fazer os Monstros andarem no Mapa*

Para este objetivo foi criada a função *MonstersWalk,* que escolhe um dos monstros, que se movimentam pelas salas, aleatoriamente e escolhe uma direção aleatória também para este se movimentar. Após isso, é verificado se o monstro se pode movimentar nessa direção, e caso o player seja super user ainda escreve na consola onde se encontra o monstro.

* + 1. *Desenvolvimento da Função para a Batalha*

Para a batalha entre o player e qualquer um dos monstros foi desenvolvida a função *Battle* que vai sortear valores para o ataque crítico do Player e do Monstro que vai depois modificar o valor de ataque de cada um deles. É ainda sorteado se o Player e o Monster atacam ou falham os seus ataques. O Player vai sendo informado sobre todos os momentos de ataque, e quando a energia do Monstro for menor ou igual a zero a luta acaba e o jogador recolhe o item e o tesouro que estejam nessa sala.

* 1. ***Terceira Fase do Desenvolvimento***
     1. *Permitir o Jogador apanhar os Itens e os Tesouros*
     2. *Implementar o Save e o Load*
     3. *Revisão do Código e Correção de Bugs*
     4. *Melhorar API*

# ***Conclusão***

# ***Referencias***

# ***Anexos***