Portuguese

PeixeraWars

1.0.0

Gerado por Doxygen 1.7.6.1

Segunda, 30 de Junho de 2014 20:57:54

Contents

Chapter 1

ESPECIFICAÇÃO

1.1 1. INFORMAÇÕES GERAIS

Nome do projeto: Peixera Wars

Projeto de Disciplina: Jogo de Estratégia

Semestre Cursado: 1/2014

Curso: Engenharia de Computação

Disciplina: Métodos de Programação

Professor: Jan Mendonça

Turma: A

1.2 2. GRUPO

No de Integrantes: 4

Nome: Maximillian Fan Xavier

Matrícula: 12/0153271

Função: Estrutura de Dados

Nome: Otávio Alves Dias

Matrícula: 12/0131480

Função: Estrutura de Dados

Nome: Rafael Dias da Costa

Matrícula: 12/0133253

Função: Interface Gráfica

Nome: Túlio Abner de Lima

Matrícula: 12/0137194

Função: Interface Gráfica

1.3 3. DESCRIÇÃO GERAL

1.3.1 3.1. Unidades

Existirão ao todo três(3) unidades de combate, onde cada qual pode ser evoluída em até dois níveis, ou seja, as unidades mais básicas iniciam no nível um(1) e podem chegar até o nível mais alto, sendo este três(3).

As unidades dependem da evolução dos edifícios para assim poderem passar para o próximo nível, ou seja, ao evoluir uma construção, a unidade específica relacionada àquele edifício também evoluirá. Dependem de ouro para serem criadas. Possuem os atributos Vida, Esquiva, Dano, e Nível.

Soldado: Tem vantagem contra lanceiro e desvantagem contra arqueiro. É a unidade com maior vida do jogo;

Arqueiro: Tem vantagem contra soldado e desvantagem contra lanceiro. É a unidade de maior esquiva do jogo;

Lanceiro: Tem vantagem contra arqueiro e desvantagem contra soldado. É a unidade com maior ataque do jogo.

1.3.2 3.2. Edifícios

Existirão ao todo quatro(4) edifícios, onde cada qual pode ser evoluído em até dois níveis, ou seja, as construções mais básicas iniciam no nível um(1) e podem chegar até o nível mais alto, sendo este três(3).

As construções dependem de pontos de evolução para serem evoluídas.

Possuem os atributos:

Quartel: Responsável pela criação de soldados;

Campo de Tiro: Responsável pela criação de arqueiros;

Casa das Lanças: Responsável pela criação de lanceiros;

Comércio: Responsável por gerar o recurso "Ouro".

Todos os edifícios juntos representam seu castelo.

1.3.3 3.3. Recursos

Ponto de Evolução: Recurso utilizado na evolução de edifícios. É adquirido ao fim de cada horda como recompensa por ter sobrevivido.

Ouro: Recurso utilizado na evolução de soldados. É gerado pelo edifício comércio.

1.3.4 3.4. Mecânica

O jogo é de tema medieval baseado em hordas, totalizando oito(10) ataques que o jogador deverá sobreviver, mantendo suas unidades vivas até o fim da rodada.

O mapa será composto por um menu superior que mostra as informações gerais, um menu esquerdo que mostra os edifícios e a janela central onde as unidades realizam o combate. O objetivo do jogador é criar batalhões de defesa que irão resistir aos ataques da CPU.

Imagem ilustrativa do jogo Final Fantasy para Super Nintendo.

O jogador já inicia a partida com todas as construções disponíveis, apenas necessitando evoluí-las conforme sua necessidade. Ao fim de cada horda, todas as unidades em campo são removidas e um ponto de evolução é gerado, possibilitando a evolução dos edifícios. O ouro gerado pelo comércio também é creditado neste instante.

Os edifícios só podem ser evoluídos na transição entre uma horda e outra, sendo representada por um tempo de pausa entre os ataques da CPU.

As unidades são criadas da seguinte forma: o jogador deverá selecionar, no inicio de cada horda, quais unidades deseja comprar. Em seguida, o combate se inicia e o usuário deve escolher que unidades inimigas atacar.

No turno do jogador, todas as unidades atacam, cabendo a ele escolher qual dos inimigos receberá o dano. No fim da rodada, a CPU faz sua investida. O dano é gerado aleatoriamente dentro de uma margem de diferença de até 10 pontos, ou seja, se um guerreiro ataca com 60 pontos, o valor absoluto de seu ataque pode flutuar dentre um intervalo de 50 e 70 pontos. No caso da esquiva, um valor aleatório é gerado entre 0 e 1 e caso este número caia fora do intervalo do inimigo, o ataque é realizado com sucesso, caso contrário, resulta em uma falha e não causa dano.

Imagem ilustrativa do menu de castelo do jogo CastleStorm.

Os comandos evolutivos do jogo serão feitos através de um menu; as informações úteis aparecerão na parte superior da tela, farão parte da HUD(heads- up display); as interações serão efetuadas através das teclas do teclado.

Critério de vitória: Sobreviver as dez(10) hordas inimigas, eliminando todas as unidades da CPU.

Critério de derrota: Ter todas as suas unidades mortas.

Chapter 2

Índice das estruturas de dados

2.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Chapter 3

Índice dos ficheiros

3.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

Chapter 4

Documentação da classe

4.1 Referência à estrutura cabecageral

Estrutura de cabeça geral responsável pela união dos edifícios e das unidades.

`#include <CabecaGeral>`

Campos de Dados

- CabecaPFilas * character
- Castelo * castle

4.1.1 Descrição detalhada

Estrutura de cabeça geral responsável pela união dos edifícios e das unidades.

4.1.2 Documentação dos campos e atributos

4.1.2.1 Castelo* castle

4.1.2.2 CabecaPFilas* character

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- estruturas.h

4.2 Referência à estrutura cabecapfilas

Estrutura de cabeça que organiza as duas filas de ação do jogo.

`#include <CabecaPFilas>`

Campos de Dados

- Unidade * player
- Unidade * cpu

4.2.1 Descrição detalhada

Estrutura de cabeça que organiza as duas filas de ação do jogo.

4.2.2 Documentação dos campos e atributos

4.2.2.1 Unidade* cpu

4.2.2.2 Unidade* player

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- estruturas.h

4.3 Referência à estrutura castelo

Estrutura de cabeça que organiza e dá acesso aos quatro edifícios do jogo, possui informações de vida e quantidade de ouro disponível.

`#include <Castelo>`

Campos de Dados

- Edificio * quartel
- Edificio * campodetiro
- Edificio * casadaslancas
- Comercio * comercio
- int ouro

4.3.1 Descrição detalhada

Estrutura de cabeça que organiza e dá acesso aos quatro edifícios do jogo, possui informações de vida e quantidade de ouro disponível.

4.3.2 Documentação dos campos e atributos

4.3.2.1 Edificio* campodetiro

4.3.2.2 Edificio* casadaslancas

4.3.2.3 Comercio* comercio

4.3.2.4 int ouro

4.3.2.5 Edificio* quartel

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- estruturas.h

4.4 Referência à estrutura comercio

Estrutura do tipo comércio a qual faz parte do castelo. Responsável por controlar a produção de ouro.

`#include <Comercio>`

Campos de Dados

- int taxaouro
- int nivel

4.4.1 Descrição detalhada

Estrutura do tipo comércio a qual faz parte do castelo. Responsável por controlar a produção de ouro.

4.4.2 Documentação dos campos e atributos

4.4.2.1 int nivel

4.4.2.2 int taxaouro

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- estruturas.h

4.5 Referência à estrutura edificio

Estrutura genérica do tipo edificio. Responsável por representar a maioria das construções do jogo.

`#include <Edificio>`

Campos de Dados

- int custounidade
- int nivel

4.5.1 Descrição detalhada

Estrutura genérica do tipo edificio. Responsável por representar a maioria das construções do jogo.

4.5.2 Documentação dos campos e atributos

4.5.2.1 int custounidade

4.5.2.2 int nivel

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- estruturas.h

4.6 Referência à estrutura unidade

Estrutura genérica do tipo unidade. Responsável por representar todas as unidades do jogo.

`#include <Unidade>`

Campos de Dados

- int vida
- int dano
- float esquiva
- int nivel
- int classe
- struct unidade * prox
- struct unidade * ant

4.6.1 Descrição detalhada

Estrutura genérica do tipo unidade. Responsável por representar todas as unidades do jogo.

4.6.2 Documentação dos campos e atributos

4.6.2.1 struct unidade* ant

4.6.2.2 int classe

4.6.2.3 int dano

4.6.2.4 float esquiva

4.6.2.5 int nivel

4.6.2.6 struct unidade* prox

4.6.2.7 int vida

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- estruturas.h

Chapter 5

Documentação do ficheiro

5.1 Referência ao ficheiro engine.c

Módulo de implementação do módulo engine.h.

```
#include <stdio.h> #include <stdlib.h> #include <string.-
h> #include <strings.h> #include <time.h> #include <unistd.-
h>  #include "estruturas.h"  #include "engine.h"  #include
"grafico.h"
```

Macros

- #define HUDH 7
- #define CASTLEW 40
- #define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
- #define CTRLD 4

Funções

- void CriaEstruturas (CabecaGeral *Interface, int novawave, int valor)

  Realiza a alocação das estruturas Castelo e CabecaPFilas e inicializa com valor NULL todas as estruturas internas de cada uma das estruturas alocadas anteriormente.
- void InicializaEstruturas (CabecaGeral *Interface, int status, int cuQ, int nQ, int cuCDT, int nCDT, int cuCDL, int nCDL, int tC, int nC)

  Aloca as estruturas internas de Castelo e CabecaPFilas.
- int VerificaInterface (CabecaGeral *Interface)

  Verifica se a estrutura controladora está alocada ou não.
- void Inicia ()

  Executa a exibição do menu principal para o jogador iniciar, carregar, visualizar os créditos ou sair do jogo.
- void StartGame (CabecaGeral *Cabeca, int horda)

Realiza a chamada para o começo do jogo, verificando se o jogo está sendo iniciado do começo ou a partir de algum save.

- void SaveGame (CabecaGeral ∗Interface, int horda)

  Realiza a gravação dos do game em um arquivo binário para posterior leitura e continuação do jogo.
- void LoadGame ()

  Realiza a leitura do arquivo binário gerado pela função Save Game e continua com o jogo.
- int MenuEvolucaoEdificio (CabecaGeral ∗Interface)

  Realiza a evolução de um dos edifícios do jogo.
- int MenuEscolhaUnidade (CabecaGeral ∗Interface)

  Realiza a escolha do personagem a ser criado no jogo.
- void Run (CabecaGeral ∗Interface, int new, int wave)

  Realiza a execução do loop principal do game, avançando as hordas de inimigos e executando as funções de criação e evolução de edifícios ∗ e personagens.
- int GameLoop (CabecaGeral ∗Interface, WINDOW ∗winfield)

  Realiza o loop de batalhas entre o player e o cpu no game.
- int Batalha (CabecaGeral ∗Interface, WINDOW ∗winfield)

  Realiza a seleção do inimigo a ser atacado pelo player, um ataque para cada um de seus personagens.
- Unidade ∗ Atacar (Unidade ∗atacante, Unidade ∗vitima)

  Realiza o ataque de um personagem do player sobre um dos personagens do cpu.
- int Batalha2 (CabecaGeral ∗Interface, WINDOW ∗winfield)

  Função que executa o loop de ataque da cpu contra os personagens do player.
- Unidade ∗ buscaAlvo (Unidade ∗vitima)

  Realiza a busca pelo personagem mais fraco do player para que a cpu possa realizar o ataque.
- Unidade ∗ AtaqueInimigo (int ∗hit, Unidade ∗atacante, Unidade ∗vitima)

  Realiza o ataque de um personagem do cpu sobre um personagem do player.
- int TemQuatro (Unidade ∗lista)

  Função para verificar se já existem 4 personagens na lista do player, para que ele possa escolher seus personagens.
- int GerarValor (int a, int b)

  Função que gera um valor aleatório entre 1 e 3 que vai indicar o tipo de personagem a ser criado pelo cpu.
- void InsereUnidadeInimiga (CabecaGeral ∗Interface, int horda)

  Realiza a inserção de um personagem na lista de personagens do cpu, de acordo com os valores gerados para a classe e nível.
- void InsereUnidadeBoss (CabecaGeral ∗Interface)

  Realiza a inserção do personagem Boss na lista de personagens do cpu quando o player vencer as 10 hordas de inimigos.
- int GerarNivel (int horda)

  Função que verifica o nível do personagem a ser criado pela cpu, de acordo com a horda de inimigos do game.

5.1.1 Descrição detalhada

Módulo de implementação do módulo engine.h.

5.1.2 Documentação das macros

5.1.2.1 #define ARRAY_SIZE(*a*) (sizeof(a) / sizeof(a[0]))

5.1.2.2 #define CASTLEW 40

5.1.2.3 #define CTRLD 4

5.1.2.4 #define HUDH 7

5.1.3 Documentação das funções

5.1.3.1 Unidade * Atacar (Unidade * *atacante,* Unidade * *vitima*)

Realiza o ataque de um personagem do player sobre um dos personagens do cpu.

Parâmetros

atacante	- ponteiro para o personagem do player (Unidade*) que vai realizar o ataque
vitima	- ponteiro para o personagem do cpu (Unidade*) que vai receber o ataque

Retorna

Retorna a lista de personagen da cpu atualizada.

Assertiva de entrada

As estruturas de personagens enviadas para a função precisam ter sido declaradas e alocadas.

5.1.3.2 Unidade * AtaqueInimigo (int * *hit,* Unidade * *atacante,* Unidade * *vitima*)

Realiza o ataque de um personagem do cpu sobre um personagem do player.

Parâmetros

hit	- ponteiro para um valor inteiro que vai indicar se foi realizado o ataque com eficiencia ou não
atacante	- ponteiro para o personagem da cpu (Unidade*) que vai realizar o ataque
vitima	- ponteiro para o personagem do player (Unidade*) que vai ser atacado

Retorna

Retorna a lista de personagens do player atualizada após o ataque.

Assertiva de entrada

A lista de personagens do player precisa ter sido declarada e alocada. O personagem da cpu que vai realizar o ataque precisa estar alocado O ponteiro para a variavel inteira precisa ter sido declarado.

5.1.3.3 int Batalha (CabecaGeral ∗ *Interface,* WINDOW ∗ *winfield*)

Realiza a seleção do inimigo a ser atacado pelo player, um ataque para cada um de seus personagens.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral∗)
winfield	- ponteiro para a estrutura WINDOW (estrutura da biblioteca ncurses) para realizar a visualização do game.

Retorna

Retorna o valor inteiro 1 caso o player tenha vencido a batalha ou 0 caso todos os ataques tenham sido realizados e ainda existem inimigos na cpu.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada. A estrutura WINDOW precisa ter sido declarada e alocada para correta visualização do game.

5.1.3.4 int Batalha2 (CabecaGeral ∗ *Interface,* WINDOW ∗ *winfield*)

Função que executa o loop de ataque da cpu contra os personagens do player.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral∗) do game.
winfield	- ponteiro para WINDOW (parâmetro da biblioteca ncurses) para atualizar a janela de exibição do game.

Retorna

Retorna o valor inteiro 0 caso todos os ataques tenham sido feitos e ainda existam personagens do player, ou 2 caso todos os personagens ∗ do player tenham sido eliminados.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada. A estrutura WINDOW precisa ser válida para poder exibir corretamente o game na tela.

5.1.3.5 Unidade ∗ buscaAlvo (Unidade ∗ *vitima*)

Realiza a busca pelo personagem mais fraco do player para que a cpu possa realizar o ataque.

Parâmetros

vitima	- ponteiro para a lista de personagens do player (Unidade∗).

Retorna

Retorna o ponteiro para o personagem (Unidade∗) a ser atacado.

Assertiva de entrada

A lista de personagens do player enviada para a função precisa ter sido declarada e alocada.

5.1.3.6 CriaEstruturas (CabecaGeral ∗ *Interface,* int *novawave,* int *valor*)

Realiza a alocação das estruturas Castelo e CabecaPFilas e inicializa com valor NULL todas as estruturas internas de cada uma das estruturas alocadas anteriormente.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral∗) do game.
novawave	- determina o valor inicial.
valor	- informa a quantidade de ouro

Retorna

Retorna a cabeça da estrutura com todas as estruturas Castelo e CabecaPFilas alocadas com seus ponteiros internos apontando para NULL.

Assertiva de entrada

A estrutura controladora precisa ter sido alocada.

5.1.3.7 int GameLoop (CabecaGeral ∗ *Interface,* WINDOW ∗ *winfield*)

Realiza o loop de batalhas entre o player e o cpu no game.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral∗)
winfield	- ponteiro para a estrutura WINDOW (estrutura da biblioteca ncurses) para realizar a visualização dos combates.

Retorna

Retorna o valor inteiro 0 caso o player tenha vencido a batalha ou 1 caso o cpu tenha vencido a batalha.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada. A estrutura WINDOW precisa ter sido declarada e alocada para realizar a visualização.

5.1.3.8 int GerarNivel (int *horda*)

Função que verifica o nível do personagem a ser criado pela cpu, de acordo com a horda de inimigos do game.

Parâmetros

horda	- valor inteiro que representa a horda de inimigos do jogo.

Retorna

Retorna um valor inteiro entre 1 e 3, indicando o nível do personagem a ser criado.

Assertiva de entrada

A variável horda precisa estar de acordo com o jogo, ou seja, estar entre 1 e 10.

5.1.3.9 int GerarValor (int *a,* int *b*)

Função que gera um valor aleatório entre 1 e 3 que vai indicar o tipo de personagem a ser criado pelo cpu.

Parâmetros

a	- valor inteiro que indica o valor máximo entre os dois.
b	- valor inteiro que indica o valor mínimo entre os dois.

Retorna

Retorna o valor inteiro, entre gerado pelo função rand.

Assertiva de entrada

Os valores de a e b precisam ser valores inteiros.

5.1.3.10 Inicia ( )

Executa a exibição do menu principal para o jogador iniciar, carregar, visualizar os créditos ou sair do jogo.

Retorna

void - não possui retorno.

Assertiva de entrada

Não possui assertivas de entrada.

5.1.3.11 void InicializaEstruturas (CabecaGeral * *Interface*, int *status*, int *cuQ*, int *nQ*, int *cuCDT*, int *nCDT*, int *cuCDL*, int *nCDL*, int *tC*, int *nC*)

Aloca as estruturas internas de Castelo e CabecaPFilas.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral*)
status	- valor inteiro que indica se as estruturas serão alocadas com os valores iniciais do game ou a partir de um valor específico
cuQ	- valor inteiro que representa o custo para a criação de um personagem do edifício Quartel
nQ	- valor inteiro que representa o nível do edifício quartel
cuCDT	- valor inteiro que representa o custo para a criação de um personagem do edifício Campo de Tiro
nCDT	- valor inteiro que representa o nível do edifício Campo de Tiro
cuCDL	- valor inteiro que representa o custo para a criação de um personagem do edifício Cada das Lanças
nCDL	- valor inteiro que representa o nível do edifício Casa das Lanças
tC	- valor inteiro que representa a quantidade de recursos gerados pelo edifício Comercio
nC	- valor inteiro que representa o nível do edifício Comercio

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido alocada A variavel status precisa obedecer os padrões, ser o valor inteiro 0 ou o valor inteiro 1 Os valores das variáveis de inicialização precisam de acordo com o game.

5.1.3.12 InsereUnidadeBoss (CabecaGeral * *Interface*)

Realiza a inserção do personagem Boss na lista de personagens do cpu quando o player vencer as 10 hordas de inimigos.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral*)

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada.

5.1.3.13 void InsereUnidadeInimiga (CabecaGeral ∗ *Interface,* int *horda*)

Realiza a inserção de um personagem na lista de personagens do cpu, de acordo com os valores gerados para a classe e nível.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral∗)
horda	- valor inteiro que indica a horda de inimigos no jogo.

Retorna

Retorna o valor inteiro 0 caso o player tenha vencido a batalha ou 1 caso o cpu tenha vencido a batalha.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada. A variável horda precisa estar de acordo com os padroões do jogo, ou seja, ser um valor entre 1 e 10.

5.1.3.14 void LoadGame ( )

Realiza a leitura do arquivo binário gerado pela função Save Game e continua com o jogo.

Não possui .

Retorna

void - não possui retorno.

Assertiva de entrada

O arquivo binário a ser lido precisa ser válido, ou seja, ter sido gerado pelo game.

5.1.3.15 MenuEscolhaUnidade (CabecaGeral ∗ *Interface*)

Realiza a escolha do personagem a ser criado no jogo.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral∗) do game.

Retorna

Retorna o valor inteiro 1 caso um dos personagens tenha sido criado, ou 0 caso nenhum tenha sido criado.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada.

5.1.3.16 int MenuEvolucaoEdificio (CabecaGeral * *Interface*)

Realiza a evolução de um dos edifícios do jogo.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.

Retorna

EvoluiuEdificio - valor inteiro que indica se foi realizada a evolução de um dos edifícios.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada.

5.1.3.17 void Run (CabecaGeral * *Interface,* int *new,* int *wave*)

Realiza a execução do loop principal do game, avançando as hordas de inimigos e executando as funções de criação e evolução de edifícios * e personagens.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.
new	- valor inteiro que indica se o jogo está sendo iniciado do começo ou a partir de um save.
wave	- valor inteiro que representa a horda de inimigos caso seja executado o jogo a partir de um load.

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada. As variáveis new e wave precisam estar de acordo com o jogo, ou seja, new precisa ser 0 ou 1 e wave precisa estar entre 1 e 10.

5.1.3.18 SaveGame (CabecaGeral ∗ *Interface,* int *horda*)

Realiza a gravação dos do game em um arquivo binário para posterior leitura e continuação do jogo.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral∗) do game.
horda	- valor inteiro que representa a horda atual de inimigos do jogo.

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada. A variavel horda precisa estar de acordo com os padrões do jogo, ou seja, estar entre 1 e 10.

5.1.3.19 StartGame (CabecaGeral ∗ *Cabeca,* int *horda*)

Realiza a chamada para o começo do jogo, verificando se o jogo está sendo iniciado do começo ou a partir de algum save.

Parâmetros

Cabeca	- ponteiro para a controladora (CabecaGeral∗) do game.
horda	- valor inteiro que representa a horda de inimigos da cpu, caso seja utilizada a função de Load Game.

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada A variavel horda precisa estar de acordo com os valores do jogo, ou seja, estar entre 1 e 10.

5.1.3.20 TemQuatro (Unidade ∗ *lista*)

Função para verificar se já existem 4 personagens na lista do player, para que ele possa escolher seus personagens.

Parâmetros

lista	- ponteiro para a lista de personagens (Unidade∗)

Retorna

Retorna o valor inteiro 1 caso já existam 4 personagens, ou 0 caso caso existam menos de 4.

Assertiva de entrada

A lista de personagens do player precisa ter sido declarada e alocada.

5.1.3.21 VerificaInterface (CabecaGeral * *Interface*)

Verifica se a estrutura controladora está alocada ou não.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.

Retorna

Retorna o valor inteiro 1 caso a estrutura esteja alocada, caso contrário retorna 0.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada.

5.2 Referência ao ficheiro engine.h

Módulo para gerenciar a engine.

`#include <stdio.h> #include <ncurses.h>`

Funções

- void CriaEstruturas (CabecaGeral *Interface, int novawave, int valor)

  Realiza a alocação das estruturas Castelo e CabecaPFilas e inicializa com valor NULL todas as estruturas internas de cada uma das estruturas alocadas anteriormente.
- void InicializaEstruturas (CabecaGeral *Interface, int status, int cuQ, int nQ, int cuCDT, int nCDT, int cuCDL, int nCDL, int tC, int nC)

  Aloca as estruturas internas de Castelo e CabecaPFilas.
- int VerificaInterface (CabecaGeral *Interface)

  Verifica se a estrutura controladora está alocada ou não.
- void StartGame (CabecaGeral *Cabeca, int horda)

  Realiza a chamada para o começo do jogo, verificando se o jogo está sendo iniciado do começo ou a partir de algum save.
- void SaveGame (CabecaGeral *Interface, int horda)

  Realiza a gravação dos do game em um arquivo binário para posterior leitura e continuação do jogo.

- void LoadGame ()

  Realiza a leitura do arquivo binário gerado pela função Save Game e continua com o jogo.
- int TemQuatro (Unidade ∗lista)

  Função para verificar se já existem 4 personagens na lista do player, para que ele possa escolher seus personagens.
- int GerarValor (int a, int b)

  Função que gera um valor aleatório entre 1 e 3 que vai indicar o tipo de personagem a ser criado pelo cpu.
- int GerarNivel (int horda)

  Função que verifica o nível do personagem a ser criado pela cpu, de acordo com a horda de inimigos do game.
- void InsereUnidadeInimiga (CabecaGeral ∗Interface, int horda)

  Realiza a inserção de um personagem na lista de personagens do cpu, de acordo com os valores gerados para a classe e nível.
- void InsereUnidadeBoss (CabecaGeral ∗Interface)

  Realiza a inserção do personagem Boss na lista de personagens do cpu quando o player vencer as 10 hordas de inimigos.
- int Batalha (CabecaGeral ∗Interface, WINDOW ∗winfield)

  Realiza a seleção do inimigo a ser atacado pelo player, um ataque para cada um de seus personagens.
- void Run (CabecaGeral ∗Interface, int new, int wave)

  Realiza a execução do loop principal do game, avançando as hordas de inimigos e executando as funções de criação e evolução de edifícios ∗ e personagens.
- int Batalha2 (CabecaGeral ∗Interface, WINDOW ∗winfield)

  Função que executa o loop de ataque da cpu contra os personagens do player.
- Unidade ∗ Atacar (Unidade ∗atacante, Unidade ∗vitima)

  Realiza o ataque de um personagem do player sobre um dos personagens do cpu.
- Unidade ∗ buscaAlvo (Unidade ∗vitima)

  Realiza a busca pelo personagem mais fraco do player para que a cpu possa realizar o ataque.
- Unidade ∗ AtaqueInimigo (int ∗hit, Unidade ∗atacante, Unidade ∗vitima)

  Realiza o ataque de um personagem do cpu sobre um personagem do player.
- int GameLoop (CabecaGeral ∗Interface, WINDOW ∗winfield)

  Realiza o loop de batalhas entre o player e o cpu no game.
- void Inicia ()

  Executa a exibição do menu principal para o jogador iniciar, carregar, visualizar os créditos ou sair do jogo.
- int MenuEvolucaoEdificio (CabecaGeral ∗Interface)

  Realiza a evolução de um dos edifícios do jogo.
- int MenuEscolhaUnidade (CabecaGeral ∗Interface)

  Realiza a escolha do personagem a ser criado no jogo.

5.2.1 Descrição detalhada

Módulo para gerenciar a engine.

5.2.2 Documentação das funções

5.2.2.1 Unidade* Atacar (Unidade * *atacante,* Unidade * *vitima*)

Realiza o ataque de um personagem do player sobre um dos personagens do cpu.

Parâmetros

atacante	- ponteiro para o personagem do player (Unidade*) que vai realizar o ataque
vitima	- ponteiro para o personagem do cpu (Unidade*) que vai receber o ataque

Retorna

Retorna a lista de personagen da cpu atualizada.

Assertiva de entrada

As estruturas de personagens enviadas para a função precisam ter sido declaradas e alocadas.

5.2.2.2 Unidade* AtaqueInimigo (int * *hit,* Unidade * *atacante,* Unidade * *vitima*)

Realiza o ataque de um personagem do cpu sobre um personagem do player.

Parâmetros

hit	- ponteiro para um valor inteiro que vai indicar se foi realizado o ataque com eficiencia ou não
atacante	- ponteiro para o personagem da cpu (Unidade*) que vai realizar o ataque
vitima	- ponteiro para o personagem do player (Unidade*) que vai ser atacado

Retorna

Retorna a lista de personagens do player atualizada após o ataque.

Assertiva de entrada

A lista de personagens do player precisa ter sido declarada e alocada. O personagem da cpu que vai realizar o ataque precisa estar alocado O ponteiro para a variavel inteira precisa ter sido declarado.

5.2.2.3 int Batalha (CabecaGeral * *Interface,* WINDOW * *winfield*)

Realiza a seleção do inimigo a ser atacado pelo player, um ataque para cada um de seus personagens.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral∗)
winfield	- ponteiro para a estrutura WINDOW (estrutura da biblioteca ncurses) para realizar a visualização do game.

Retorna

Retorna o valor inteiro 1 caso o player tenha vencido a batalha ou 0 caso todos os ataques tenham sido realizados e ainda existem inimigos na cpu.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada. A estrutura WINDOW precisa ter sido declarada e alocada para correta visualização do game.

5.2.2.4 int Batalha2 (CabecaGeral ∗ *Interface,* WINDOW ∗ *winfield*)

Função que executa o loop de ataque da cpu contra os personagens do player.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral∗) do game.
winfield	- ponteiro para WINDOW (parâmetro da biblioteca ncurses) para atualizar a janela de exibição do game.

Retorna

Retorna o valor inteiro 0 caso todos os ataques tenham sido feitos e ainda existam personagens do player, ou 2 caso todos os personagens ∗ do player tenham sido eliminados.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada. A estrutura WINDOW precisa ser válida para poder exibir corretamente o game na tela.

5.2.2.5 Unidade∗ buscaAlvo (Unidade ∗ *vitima*)

Realiza a busca pelo personagem mais fraco do player para que a cpu possa realizar o ataque.

Parâmetros

vitima	- ponteiro para a lista de personagens do player (Unidade∗).

Retorna

Retorna o ponteiro para o personagem (Unidade∗) a ser atacado.

Assertiva de entrada

A lista de personagens do player enviada para a função precisa ter sido declarada e alocada.

5.2.2.6 void CriaEstruturas (CabecaGeral ∗ *Interface,* int *novawave,* int *valor*)

Realiza a alocação das estruturas Castelo e CabecaPFilas e inicializa com valor NULL todas as estruturas internas de cada uma das estruturas alocadas anteriormente.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral∗) do game.
novawave	- determina o valor inicial.
valor	- informa a quantidade de ouro

Retorna

Retorna a cabeça da estrutura com todas as estruturas Castelo e CabecaPFilas alocadas com seus ponteiros internos apontando para NULL.

Assertiva de entrada

A estrutura controladora precisa ter sido alocada.

5.2.2.7 int GameLoop (CabecaGeral ∗ *Interface,* WINDOW ∗ *winfield*)

Realiza o loop de batalhas entre o player e o cpu no game.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral∗)
winfield	- ponteiro para a estrutura WINDOW (estrutura da biblioteca ncurses) para realizar a visualização dos combates.

Retorna

Retorna o valor inteiro 0 caso o player tenha vencido a batalha ou 1 caso o cpu tenha vencido a batalha.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada. A estrutura WINDOW precisa ter sido declarada e alocada para realizar a visualização.

5.2.2.8 int GerarNivel (int *horda*)

Função que verifica o nível do personagem a ser criado pela cpu, de acordo com a horda de inimigos do game.

Parâmetros

horda	- valor inteiro que representa a horda de inimigos do jogo.

Retorna

Retorna um valor inteiro entre 1 e 3, indicando o nível do personagem a ser criado.

Assertiva de entrada

A variável horda precisa estar de acordo com o jogo, ou seja, estar entre 1 e 10.

5.2.2.9 int GerarValor (int *a,* int *b*)

Função que gera um valor aleatório entre 1 e 3 que vai indicar o tipo de personagem a ser criado pelo cpu.

Parâmetros

a	- valor inteiro que indica o valor máximo entre os dois.
b	- valor inteiro que indica o valor mínimo entre os dois.

Retorna

Retorna o valor inteiro, entre gerado pelo função rand.

Assertiva de entrada

Os valores de a e b precisam ser valores inteiros.

5.2.2.10 void Inicia ( )

Executa a exibição do menu principal para o jogador iniciar, carregar, visualizar os créditos ou sair do jogo.

Retorna

void - não possui retorno.

Assertiva de entrada

Não possui assertivas de entrada.

5.2.2.11 void InicializaEstruturas (CabecaGeral * *Interface,* int *status,* int *cuQ,* int *nQ,* int *cuCDT,* int *nCDT,* int *cuCDL,* int *nCDL,* int *tC,* int *nC*)

Aloca as estruturas internas de Castelo e CabecaPFilas.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral*)
status	- valor inteiro que indica se as estruturas serão alocadas com os valores iniciais do game ou a partir de um valor específico
cuQ	- valor inteiro que representa o custo para a criação de um personagem do edifício Quartel
nQ	- valor inteiro que representa o nível do edifício quartel
cuCDT	- valor inteiro que representa o custo para a criação de um personagem do edifício Campo de Tiro
nCDT	- valor inteiro que representa o nível do edifício Campo de Tiro
cuCDL	- valor inteiro que representa o custo para a criação de um personagem do edifício Cada das Lanças
nCDL	- valor inteiro que representa o nível do edifício Casa das Lanças
tC	- valor inteiro que representa a quantidade de recursos gerados pelo edifício Comercio
nC	- valor inteiro que representa o nível do edifício Comercio

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido alocada A variavel status precisa obedecer os padrões, ser o valor inteiro 0 ou o valor inteiro 1 Os valores das variáveis de inicialização precisam de acordo com o game.

5.2.2.12 void InsereUnidadeBoss (CabecaGeral * *Interface*)

Realiza a inserção do personagem Boss na lista de personagens do cpu quando o player vencer as 10 hordas de inimigos.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral*)

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada.

5.2.2.13 void InsereUnidadeInimiga (CabecaGeral * *Interface,* int *horda*)

Realiza a inserção de um personagem na lista de personagens do cpu, de acordo com os valores gerados para a classe e nível.

Parâmetros

Interface	- ponteiro para a controladora do game (CabecaGeral*)
horda	- valor inteiro que indica a horda de inimigos no jogo.

Retorna

Retorna o valor inteiro 0 caso o player tenha vencido a batalha ou 1 caso o cpu tenha vencido a batalha.

Assertiva de entrada

A estrutura controladora do game precisa ter sido declarada e alocada. A variável horda precisa estar de acordo com os padroões do jogo, ou seja, ser um valor entre 1 e 10.

5.2.2.14 void LoadGame ( )

Realiza a leitura do arquivo binário gerado pela função Save Game e continua com o jogo.

Não possui .

Retorna

void - não possui retorno.

Assertiva de entrada

O arquivo binário a ser lido precisa ser válido, ou seja, ter sido gerado pelo game.

5.2.2.15 int MenuEscolhaUnidade (CabecaGeral * *Interface*)

Realiza a escolha do personagem a ser criado no jogo.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.

Retorna

Retorna o valor inteiro 1 caso um dos personagens tenha sido criado, ou 0 caso nenhum tenha sido criado.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada.

5.2.2.16 int MenuEvolucaoEdificio (CabecaGeral * *Interface*)

Realiza a evolução de um dos edifícios do jogo.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.

Retorna

EvoluiuEdificio - valor inteiro que indica se foi realizada a evolução de um dos edifícios.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada.

5.2.2.17 void Run (CabecaGeral * *Interface,* int *new,* int *wave*)

Realiza a execução do loop principal do game, avançando as hordas de inimigos e executando as funções de criação e evolução de edifícios * e personagens.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.
new	- valor inteiro que indica se o jogo está sendo iniciado do começo ou a partir de um save.
wave	- valor inteiro que representa a horda de inimigos caso seja executado o jogo a partir de um load.

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada. As variáveis new e wave precisam estar de acordo com o jogo, ou seja, new precisa ser 0 ou 1 e wave precisa estar entre 1 e 10.

5.2.2.18 void SaveGame (CabecaGeral * *Interface,* int *horda*)

Realiza a gravação dos do game em um arquivo binário para posterior leitura e continuação do jogo.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.
horda	- valor inteiro que representa a horda atual de inimigos do jogo.

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada e alocada. A variavel horda precisa estar de acordo com os padrões do jogo, ou seja, estar entre 1 e 10.

5.2.2.19 void StartGame (CabecaGeral * *Cabeca,* int *horda*)

Realiza a chamada para o começo do jogo, verificando se o jogo está sendo iniciado do começo ou a partir de algum save.

Parâmetros

Cabeca	- ponteiro para a controladora (CabecaGeral*) do game.
horda	- valor inteiro que representa a horda de inimigos da cpu, caso seja utilizada a função de Load Game.

Retorna

void - não possui retorno.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada A variavel horda precisa estar de acordo com os valores do jogo, ou seja, estar entre 1 e 10.

5.2.2.20 int TemQuatro (Unidade * *lista*)

Função para verificar se já existem 4 personagens na lista do player, para que ele possa escolher seus personagens.

Parâmetros

lista	- ponteiro para a lista de personagens (Unidade*)

Retorna

Retorna o valor inteiro 1 caso já existam 4 personagens, ou 0 caso caso existam menos de 4.

Assertiva de entrada

A lista de personagens do player precisa ter sido declarada e alocada.

5.2.2.21 int VerificaInterface (CabecaGeral * *Interface*)

Verifica se a estrutura controladora está alocada ou não.

Parâmetros

Interface	- ponteiro para a controladora (CabecaGeral*) do game.

Retorna

Retorna o valor inteiro 1 caso a estrutura esteja alocada, caso contrário retorna 0.

Assertiva de entrada

A estrutura controladora enviada para a função precisa ter sido declarada.

5.3 Referência ao ficheiro especificacao.hh

5.4 Referência ao ficheiro estruturas.h

Módulo para gerenciar as estruturas.

```
#include <stdio.h>
```

Estruturas de Dados

- struct unidade

  Estrutura genérica do tipo unidade. Responsável por representar todas as unidades do jogo.
- struct edificio

  Estrutura genérica do tipo edificio. Responsável por representar a maioria das construções do jogo.
- struct comercio

  Estrutura do tipo comércio a qual faz parte do castelo. Responsável por controlar a produção de ouro.
- struct cabecapfilas

  Estrutura de cabeça que organiza as duas filas de ação do jogo.
- struct castelo

  Estrutura de cabeça que organiza e dá acesso aos quatro edifícios do jogo, possui informações de vida e quantidade de ouro disponível.
- struct cabecageral

  Estrutura de cabeça geral responsável pela união dos edifícios e das unidades.

Definições de tipos

- typedef struct unidade Unidade
- typedef struct edificio Edificio
- typedef struct comercio Comercio
- typedef struct cabecapfilas CabecaPFilas
- typedef struct castelo Castelo
- typedef struct cabecageral CabecaGeral

Funções

- CabecaGeral ∗ CriaCabecaGeral ()

  Aloca a estrutura controladora do game e faz com que seus ponteiros internos para as estruturas do game apontem para NULL.
- int CabecaGeralVazia (CabecaGeral ∗lis)

  Verifica se a estrutura controladora do game é nula ou não.
- int VerificarCabecaGeral (CabecaGeral ∗CG)

  Verifica se as estruturas internas do controlador são nulas ou não.
- CabecaPFilas ∗ CriaCabecaPFilas ()

  Aloca a estrutura que contém as duas listas de personagens do game (player e cpu), fazendo com que seus ponteiros internos apontem para ∗ NULL.
- int CabecaPFilasVazia (CabecaPFilas ∗lis)

  Verifica se a estrutura que contém as listas de personagens é nula ou não.
- Castelo ∗ CriaCastelo (int novawave, int valor)

  Aloca a estrutura que contém os ponteiros para cada edifício do game e faz com que esses ponteiros apontem para NULL.
- int CasteloVazia (Castelo ∗lis)

  Verifica se a estrutura que contém os edifícios é nula ou não.
- int VerificarCastelo (Castelo ∗C)

  Verifica se cada edifício da estrutura Castelo é nulo ou não.
- Castelo ∗ AtualizarCasteloOuro (Castelo ∗C, int ouro)

  Atualiza a quantidade de ouro (recurso) do castelo do game.
- Unidade ∗ CriaUnidade ()

  Inicializa uma estrutura de Unidade com o valor NULL.
- int UnidadeVazia (Unidade ∗lis)

  Verifica se a estrutura de Unidade é nula ou não.
- Unidade ∗ InserirUnidadeFinal (Unidade ∗, int, int, float, int, int)

  Realiza a inserção de um personagem na lista de personagens do game sempre a última posição da lista.
- int VerificaEstrutura (Unidade ∗lista)

  Assertiva estrutural para garantir que a estrutura de lista duplamente encadeada foi implementada corretamente.
- Unidade ∗ RemoverUnidadeFinal (Unidade ∗lis)

  Realiza a remoção do último nó (personagem) da lista de personagens.
- Unidade ∗ RemoverUnidade (Unidade ∗lis, Unidade ∗retira)

Realiza a remoção do nó (personagem) específico que foi enviado para a função.

- Unidade ∗ LimpaLista (Unidade ∗lis)

  Realiza a limpeza de uma lista de personagens, removendo todos os seus nós.

- Comercio ∗ CriaComercio ()

  Inicializa a estrutura Comércio com o valor NULL.

- int ComercioVazia (Comercio ∗lis)

  Verifica se a estrutura Comercio está alocada ou não.

- Comercio ∗ InicializarComercio (Comercio ∗C, int taxaouro, int nivel)

  Aloca a estrutura Comercio com os devidos atributos.

- Edificio ∗ CriaEdificio ()

  Inicializa a estrutura Edificio com o valor NULL.

- int EdificioVazia (Edificio ∗lis)

  Verifica se a estrutura Edificio é nula ou não.

- Edificio ∗ InicializarEdificio (Edificio ∗E, int custounidade, int nivel)

  Aloca a estrutura Edifício que pode representar no game um Quartel, Casa da Lanças ou Campo de Tiro.

5.4.1 Descrição detalhada

Módulo para gerenciar as estruturas.

5.4.2 Documentação dos tipos

5.4.2.1 typedef struct cabecageral CabecaGeral

5.4.2.2 typedef struct cabecapfilas CabecaPFilas

5.4.2.3 typedef struct castelo Castelo

5.4.2.4 typedef struct comercio Comercio

5.4.2.5 typedef struct edificio Edificio

5.4.2.6 typedef struct unidade Unidade

5.4.3 Documentação das funções

5.4.3.1 Castelo ∗ AtualizarCasteloOuro (Castelo ∗ *C,* int *ouro*)

Atualiza a quantidade de ouro (recurso) do castelo do game.

Parâmetros

C	- ponteiro para a estrutura de Castelo do game.
ouro	- valor inteiro que representa a quantidade de ouro a ser adicionada ou removida do Castelo.

Retorna

Retorna a estrutura de Castelo com a quantidade de ouro atualizada.

Assertiva de entrada

A estrutura de Castelo enviada precisa ter sido declarada. A quantidade de ouro precisa ser um valor inteiro.

5.4.3.2 int CabecaGeralVazia (CabecaGeral * *lis*)

Verifica se a estrutura controladora do game é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura controladora

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso não seja.

Assertiva de entrada

A estrutura enviada precisa ter sido declarada.

5.4.3.3 int CabecaPFilasVazia (CabecaPFilas * *lis*)

Verifica se a estrutura que contém as listas de personagens é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura dos personagens

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso não seja.

Assertiva de entrada

A estrutura enviada para a função precisa ter sido declarada.

5.4.3.4 int CasteloVazia (Castelo * *lis*)

Verifica se a estrutura que contém os edifícios é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura que contém os edifícios

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso não seja.

Assertiva de entrada

A estrutura enviada para a função precisa ter sido declarada.

5.4.3.5 int ComercioVazia (Comercio * *lis*)

Verifica se a estrutura Comercio está alocada ou não.

Parâmetros

lis	- ponteiro para a estrutura Comercio (geradora de recursos).

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Comercio precisa ter sido declarada.

5.4.3.6 CabecaGeral * CriaCabecaGeral ( )

Aloca a estrutura controladora do game e faz com que seus ponteiros internos para as estruturas do game apontem para NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna a estrutura alocada com os ponteiros internos apontando para NULL.

Assertiva de entrada

Não possui assertivas de entrada.

5.4.3.7 CabecaPFilas * CriaCabecaPFilas ( )

Aloca a estrutura que contém as duas listas de personagens do game (player e cpu), fazendo com que seus ponteiros internos apontem para * NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna a estrutura alocada com os ponteiros internos apontando para NULL.

Assertiva de entrada

Não possui assertivas de entrada.

5.4.3.8 Castelo * CriaCastelo (int *novawave,* int *valor*)

Aloca a estrutura que contém os ponteiros para cada edifício do game e faz com que esses ponteiros apontem para NULL.

Parâmetros

novawave	- determina o valor inicial
valor	- informa a quantidade de ouro

Retorna

Retorna o a estrutura alocada com os ponteiros internos apontando para NULL.

Assertiva de entrada

Não possui assertivas de entrada.

5.4.3.9 Comercio * CriaComercio ( )

Inicializa a estrutura Comércio com o valor NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna o valor NULL para o ponteiro do tipo Comercio.

Assertiva de entrada

Não possui assertivas de entrada.

5.4.3.10 Edificio * CriaEdificio ( )

Inicializa a estrutura Edificio com o valor NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna o valor NULL para o ponteiro do tipo Edificio.

Assertiva de entrada

Não possui assertivas de entrada.

5.4.3.11 Unidade * CriaUnidade ( )

Inicializa uma estrutura de Unidade com o valor NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna o valor NULL para o ponteiro do tipo Unidade.

Assertiva de entrada

Não possui assertivas de entrada.

5.4.3.12 int EdificioVazia (Edificio * *lis*)

Verifica se a estrutura Edificio é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura Edificio.

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Edificio enviada para a função precisa ter sido declarada.

5.4.3.13 Comercio * InicializarComercio (Comercio * *C,* int *taxaouro,* int *nivel*)

Aloca a estrutura Comercio com os devidos atributos.

Parâmetros

C	- ponteiro para a estrutura Comercio.
taxaouro	- valor inteiro que representa a quantidade de ouro gerada por wave no game.
nivel	- valor inteiro que representa o nível do edifício Comércio.

Retorna

Retorna a estrutura alocada.

Assertiva de entrada

A estrutura de Comercio enviada para a função precisa ser válida. Os tipos dos parâmetros precisam ser obedecidos.

5.4.3.14 Edificio * InicializarEdificio (Edificio * *E,* int *custounidade,* int *nivel*)

Aloca a estrutura Edifício que pode representar no game um Quartel, Casa da Lanças ou Campo de Tiro.

Parâmetros

E	- ponteiro para a estrutura Edificio.
custounidade	- valor inteiro que representa a taxa para a criação de uma unidade para aquele tipo de edifício.
nivel	- valor inteiro que representa o nível do edifício.

Retorna

Retorna a estrutura alocada.

Assertiva de entrada

A estrutura de Edificio enviada para a função precisa ter sido declarada. Os tipos de cada parâmetros precisam estar de acordo com a função.

5.4.3.15 Unidade * InserirUnidadeFinal (Unidade * *lis,* int *vida,* int *dano,* float *esquiva,* int *nivel,* int *classe*)

Realiza a inserção de um personagem na lista de personagens do game sempre a última posição da lista.

Parâmetros

lis	- ponteiro para a estrutura Unidade (lista de personagens)
vida	- valor inteiro que representa a vida do personagem.
dano	- valor inteiro que representa o dano causado por esse personagem.
esquiva	- valor decimal que representa a capacidade de esquiva do personagem.
nivel	- valor inteiro que representa o nível do personagem.
classe	- valor inteiro que representa qual o tipo do personagem.

Retorna

Retorna a lista de personagens atualizada.

Assertiva de entrada

A estrutura Unidade enviada para a função precisa ter sido declarada. Os tipos de cada parâmetros precisam estar de acordo.

5.4.3.16 Unidade * LimpaLista (Unidade * *lis*)

Realiza a limpeza de uma lista de personagens, removendo todos os seus nós.

Parâmetros

lis	- ponteiro para a estrutura Unidade (lista de personagens).

Retorna

Retorna a lista sem nenhum nó com o valor NULL.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ser válida.

5.4.3.17 Unidade * RemoverUnidade (Unidade * *lis,* Unidade * *retira*)

Realiza a remoção do nó (personagem) específico que foi enviado para a função.

Parâmetros

lis	- ponteiro para estrutura Unidade (lista de personagens).
retira	- ponteiro para o nó (personagem) específico a ser removido.

Retorna

Retorna a lista atualizada.

Assertiva de entrada

A estrutura de Unidade que contém os personagens precisa ter sido declarada. A estrutura de Unidade que será removida precisa estar contida na lista enviada.

5.4.3.18 Unidade * RemoverUnidadeFinal (Unidade * *lis*)

Realiza a remoção do último nó (personagem) da lista de personagens.

Parâmetros

lis	- ponteiro para a estrutura Unidade (lista de personagens).

Retorna

Retorna a lista atualizada.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ter sido declarada.

5.4.3.19 int UnidadeVazia (Unidade * *lis*)

Verifica se a estrutura de Unidade é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura Unidade.

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ter sido declarada corretamente.

5.4.3.20 int VerificaEstrutura (Unidade * *lista*)

Assertiva estrutural para garantir que a estrutura de lista duplamente encadeada foi implementada corretamente.

Parâmetros

lista	- ponteiro para a estrutura Unidade (lista de personagens).

Retorna

Retorna o valor inteiro 1 caso a estrutura esteja correta, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ter sido declarada.

5.4.3.21 int VerificarCabecaGeral (CabecaGeral * *CG*)

Verifica se as estruturas internas do controlador são nulas ou não.

Parâmetros

CG	- ponteiro para a estrutura controladora

Retorna

Retorna o valor inteiro 1 caso as suas estruturas não sejam nulas, ou 0 caso as estruturas sejam nulas.

Assertiva de entrada

A estrutura controladora precisa ter sido declarada.

5.4.3.22 int VerificarCastelo (Castelo * *C*)

Verifica se cada edifício da estrutura Castelo é nulo ou não.

Parâmetros

C	- ponteiro para a estrutura que contém os edifícios

Retorna

Retorna o valor inteiro 1 caso todos os edifícios não sejam nulos, ou 0 caso algum deles seja.

Assertiva de entrada

A estrutura enviada para a função precisa ter sido declarada para que a análise funcione corretamente.

5.5 Referência ao ficheiro funcoes.c

Módulo para gerenciar as estruturas.

```
#include <stdio.h> #include <stdlib.h> #include <string.-
h> #include <strings.h> #include "estruturas.h" #include
"engine.h"
```

Funções

- CabecaGeral ∗ CriaCabecaGeral ()

  Aloca a estrutura controladora do game e faz com que seus ponteiros internos para as estruturas do game apontem para NULL.
- CabecaPFilas ∗ CriaCabecaPFilas ()

  Aloca a estrutura que contém as duas listas de personagens do game (player e cpu), fazendo com que seus ponteiros internos apontem para ∗ NULL.
- Castelo ∗ CriaCastelo (int novawave, int valor)

  Aloca a estrutura que contém os ponteiros para cada edifício do game e faz com que esses ponteiros apontem para NULL.
- Unidade ∗ CriaUnidade ()

  Inicializa uma estrutura de Unidade com o valor NULL.
- Edificio ∗ CriaEdificio ()

  Inicializa a estrutura Edificio com o valor NULL.
- Comercio ∗ CriaComercio ()

  Inicializa a estrutura Comércio com o valor NULL.
- int CabecaGeralVazia (CabecaGeral ∗lis)

  Verifica se a estrutura controladora do game é nula ou não.
- int CabecaPFilasVazia (CabecaPFilas ∗lis)

  Verifica se a estrutura que contém as listas de personagens é nula ou não.
- int CasteloVazia (Castelo ∗lis)

  Verifica se a estrutura que contém os edifícios é nula ou não.
- int UnidadeVazia (Unidade ∗lis)

  Verifica se a estrutura de Unidade é nula ou não.
- int ComercioVazia (Comercio ∗lis)

  Verifica se a estrutura Comercio está alocada ou não.

- int EdificioVazia (Edificio ∗lis)

  Verifica se a estrutura Edificio é nula ou não.
- Unidade ∗ InserirUnidadeFinal (Unidade ∗lis, int vida, int dano, float esquiva, int nivel, int classe)

  Realiza a inserção de um personagem na lista de personagens do game sempre a última posição da lista.
- Unidade ∗ RemoverUnidade (Unidade ∗lis, Unidade ∗retira)

  Realiza a remoção do nó (personagem) específico que foi enviado para a função.
- Unidade ∗ LimpaLista (Unidade ∗lis)

  Realiza a limpeza de uma lista de personagens, removendo todos os seus nós.
- Unidade ∗ RemoverUnidadeFinal (Unidade ∗lis)

  Realiza a remoção do último nó (personagem) da lista de personagens.
- Edificio ∗ InicializarEdificio (Edificio ∗E, int custounidade, int nivel)

  Aloca a estrutura Edifício que pode representar no game um Quartel, Casa da Lanças ou Campo de Tiro.
- Comercio ∗ InicializarComercio (Comercio ∗C, int taxaouro, int nivel)

  Aloca a estrutura Comercio com os devidos atributos.
- int VerificarCastelo (Castelo ∗C)

  Verifica se cada edifício da estrutura Castelo é nulo ou não.
- int VerificarCabecaGeral (CabecaGeral ∗CG)

  Verifica se as estruturas internas do controlador são nulas ou não.
- int VerificaEstrutura (Unidade ∗lista)

  Assertiva estrutural para garantir que a estrutura de lista duplamente encadeada foi implementada corretamente.
- Castelo ∗ AtualizarCasteloOuro (Castelo ∗C, int ouro)

  Atualiza a quantidade de ouro (recurso) do castelo do game.

5.5.1 Descrição detalhada

Módulo para gerenciar as estruturas.

5.5.2 Documentação das funções

5.5.2.1 Castelo∗ AtualizarCasteloOuro (Castelo ∗ *C,* int *ouro*)

Atualiza a quantidade de ouro (recurso) do castelo do game.

Parâmetros

C	- ponteiro para a estrutura de Castelo do game.
ouro	- valor inteiro que representa a quantidade de ouro a ser adicionada ou removida do Castelo.

Retorna

Retorna a estrutura de Castelo com a quantidade de ouro atualizada.

Assertiva de entrada

A estrutura de Castelo enviada precisa ter sido declarada. A quantidade de ouro precisa ser um valor inteiro.

5.5.2.2 int CabecaGeralVazia (CabecaGeral * *lis*)

Verifica se a estrutura controladora do game é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura controladora

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso não seja.

Assertiva de entrada

A estrutura enviada precisa ter sido declarada.

5.5.2.3 int CabecaPFilasVazia (CabecaPFilas * *lis*)

Verifica se a estrutura que contém as listas de personagens é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura dos personagens

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso não seja.

Assertiva de entrada

A estrutura enviada para a função precisa ter sido declarada.

5.5.2.4 int CasteloVazia (Castelo * *lis*)

Verifica se a estrutura que contém os edifícios é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura que contém os edifícios

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso não seja.

Assertiva de entrada

A estrutura enviada para a função precisa ter sido declarada.

5.5.2.5 int ComercioVazia (Comercio * *lis*)

Verifica se a estrutura Comercio está alocada ou não.

Parâmetros

lis	- ponteiro para a estrutura Comercio (geradora de recursos).

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Comercio precisa ter sido declarada.

5.5.2.6 CabecaGeral* CriaCabecaGeral ( )

Aloca a estrutura controladora do game e faz com que seus ponteiros internos para as estruturas do game apontem para NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna a estrutura alocada com os ponteiros internos apontando para NULL.

Assertiva de entrada

Não possui assertivas de entrada.

5.5.2.7 CabecaPFilas* CriaCabecaPFilas ( )

Aloca a estrutura que contém as duas listas de personagens do game (player e cpu), fazendo com que seus ponteiros internos apontem para * NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna a estrutura alocada com os ponteiros internos apontando para NULL.

Assertiva de entrada

Não possui assertivas de entrada.

5.5.2.8 Castelo* CriaCastelo (int *novawave,* int *valor*)

Aloca a estrutura que contém os ponteiros para cada edifício do game e faz com que esses ponteiros apontem para NULL.

Parâmetros

novawave	- determina o valor inicial
valor	- informa a quantidade de ouro

Retorna

Retorna o a estrutura alocada com os ponteiros internos apontando para NULL.

Assertiva de entrada

Não possui assertivas de entrada.

5.5.2.9 Comercio* CriaComercio ( )

Inicializa a estrutura Comércio com o valor NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna o valor NULL para o ponteiro do tipo Comercio.

Assertiva de entrada

Não possui assertivas de entrada.

5.5.2.10 Edificio* CriaEdificio ( )

Inicializa a estrutura Edificio com o valor NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna o valor NULL para o ponteiro do tipo Edificio.

Assertiva de entrada

Não possui assertivas de entrada.

5.5.2.11 Unidade* CriaUnidade ( )

Inicializa uma estrutura de Unidade com o valor NULL.

void - Nenhum parâmetro enviada para a função.

Retorna

Retorna o valor NULL para o ponteiro do tipo Unidade.

Assertiva de entrada

Não possui assertivas de entrada.

5.5.2.12 int EdificioVazia (Edificio * *lis*)

Verifica se a estrutura Edificio é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura Edificio.

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Edificio enviada para a função precisa ter sido declarada.

5.5.2.13 Comercio* InicializarComercio (Comercio * *C,* int *taxaouro,* int *nivel*)

Aloca a estrutura Comercio com os devidos atributos.

Parâmetros

C	- ponteiro para a estrutura Comercio.
taxaouro	- valor inteiro que representa a quantidade de ouro gerada por wave no game.
nivel	- valor inteiro que representa o nível do edifício Comércio.

Retorna

Retorna a estrutura alocada.

Assertiva de entrada

A estrutura de Comercio enviada para a função precisa ser válida. Os tipos dos parâmetros precisam ser obedecidos.

5.5.2.14 Edificio* InicializarEdificio (Edificio * *E,* int *custounidade,* int *nivel*)

Aloca a estrutura Edifício que pode representar no game um Quartel, Casa da Lanças ou Campo de Tiro.

Parâmetros

E	- ponteiro para a estrutura Edificio.
custounidade	- valor inteiro que representa a taxa para a criação de uma unidade para aquele tipo de edifício.
nivel	- valor inteiro que representa o nível do edifício.

Retorna

Retorna a estrutura alocada.

Assertiva de entrada

A estrutura de Edificio enviada para a função precisa ter sido declarada. Os tipos de cada parâmetros precisam estar de acordo com a função.

5.5.2.15 Unidade* InserirUnidadeFinal (Unidade * *lis,* int *vida,* int *dano,* float *esquiva,* int *nivel,* int *classe*)

Realiza a inserção de um personagem na lista de personagens do game sempre a última posição da lista.

Parâmetros

lis	- ponteiro para a estrutura Unidade (lista de personagens)
vida	- valor inteiro que representa a vida do personagem.
dano	- valor inteiro que representa o dano causado por esse personagem.
esquiva	- valor decimal que representa a capacidade de esquiva do personagem.
nivel	- valor inteiro que representa o nível do personagem.
classe	- valor inteiro que representa qual o tipo do personagem.

Retorna

Retorna a lista de personagens atualizada.

Assertiva de entrada

A estrutura Unidade enviada para a função precisa ter sido declarada. Os tipos de cada parâmetros precisam estar de acordo.

5.5.2.16 Unidade* LimpaLista (Unidade * *lis*)

Realiza a limpeza de uma lista de personagens, removendo todos os seus nós.

Parâmetros

lis	- ponteiro para a estrutura Unidade (lista de personagens).

Retorna

Retorna a lista sem nenhum nó com o valor NULL.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ser válida.

5.5.2.17 Unidade* RemoverUnidade (Unidade * *lis*, Unidade * *retira*)

Realiza a remoção do nó (personagem) específico que foi enviado para a função.

Parâmetros

lis	- ponteiro para estrutura Unidade (lista de personagens).
retira	- ponteiro para o nó (personagem) específico a ser removido.

Retorna

Retorna a lista atualizada.

Assertiva de entrada

A estrutura de Unidade que contém os personagens precisa ter sido declarada. A estrutura de Unidade que será removida precisa estar contida na lista enviada.

5.5.2.18 Unidade* RemoverUnidadeFinal (Unidade * *lis*)

Realiza a remoção do último nó (personagem) da lista de personagens.

Parâmetros

lis	- ponteiro para a estrutura Unidade (lista de personagens).

Retorna

Retorna a lista atualizada.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ter sido declarada.

5.5.2.19 int UnidadeVazia (Unidade * *lis*)

Verifica se a estrutura de Unidade é nula ou não.

Parâmetros

lis	- ponteiro para a estrutura Unidade.

Retorna

Retorna o valor inteiro 1 caso a estrutura seja nula, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ter sido declarada corretamente.

5.5.2.20 int VerificaEstrutura (Unidade * *lista*)

Assertiva estrutural para garantir que a estrutura de lista duplamente encadeada foi implementada corretamente.

Parâmetros

lista	- ponteiro para a estrutura Unidade (lista de personagens).

Retorna

Retorna o valor inteiro 1 caso a estrutura esteja correta, ou 0 caso contrário.

Assertiva de entrada

A estrutura de Unidade enviada para a função precisa ter sido declarada.

5.5.2.21 int VerificarCabecaGeral (CabecaGeral * *CG*)

Verifica se as estruturas internas do controlador são nulas ou não.

Parâmetros

CG	- ponteiro para a estrutura controladora

Retorna

Retorna o valor inteiro 1 caso as suas estruturas não sejam nulas, ou 0 caso as estruturas sejam nulas.

Assertiva de entrada

A estrutura controladora precisa ter sido declarada.

5.5.2.22 int VerificarCastelo (Castelo * *C*)

Verifica se cada edifício da estrutura Castelo é nulo ou não.

Parâmetros

C	- ponteiro para a estrutura que contém os edifícios

Retorna

Retorna o valor inteiro 1 caso todos os edifícios não sejam nulos, ou 0 caso algum deles seja.

Assertiva de entrada

A estrutura enviada para a função precisa ter sido declarada para que a análise funcione corretamente.

5.6 Referência ao ficheiro grafico.c

Módulo de implementação do módulo grafico.h.

```
#include <ncurses.h> #include <menu.h> #include <stdio.-
h> #include <stdlib.h> #include <string.h> #include <time.-
h> #include <unistd.h> #include "grafico.h" #include "estruturas.-
h" #include "engine.h"
```

Macros

- #define HUDH 7
- #define CASTLEW 40
- #define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
- #define CTRLD 4

Funções

- void inicia_ncurses ()

  Função responsável por iniciar o modo ncurses.
- void finaliza_ncurses ()

  Função responsável por finalizar o modo ncurses.
- void destroy_win (WINDOW ∗local_win)

  Função responsável por apagar as informações de uma janela e deletar sua estrutura.
- int menu_edificio ()

  Função responsável pela criação do menu de evolução dos edifícios.
- int menu_unidade (Castelo ∗castelo)

  Função responsável pela criação do menu de evolução dos edifícios.
- void mensagem (int y, int x, char frase[])

  Função responsável pela impressão de uma mensagem na tela.
- char ∗ mensagem_entrada (int y, int x, char frase[])

  Função responsável por imprimir uma mensagem na tela e guardar a informação digitada pelo usuário.
- void tela_inicial ()

  Função responsável pela impressão da tela inicial, com desenhos e opções do menu.

- void tela_final ()

  Função responsável por imprimir a mensagem de despedida e encerrar o programa.
- void TelaGameOver ()

  Função responsável por imprimir a mensagem de fim de jogo e encerrar o programa.
- void creditos (int y, int x)

  Função responsável pela impressão dos créditos.
- void print_hud (WINDOW ∗hud, int gold, int wave)

  Função responsável pela impressão do HUD superior, contendo título e informações sobre o jogo(ouro disponível e numero da horda).
- void print_field (WINDOW ∗field, CabecaPFilas ∗filas)

  Função responsável pela impressão do campo de batalha.
- void print_castle (WINDOW ∗castle, Castelo ∗castelo)

  Função responsável pela impressão da área de edifícios.
- int menu (WINDOW ∗menuwin)

  Função que cria o menu de opções disponíveis para o usuário na tela inicial.
- void desenhaunidade (WINDOW ∗win, int unidade, int y, int x)

  Função responsável pela impressão do desenho da unidade escolhida nas coordenadas indicadas.
- void desenhainimigo (WINDOW ∗win, int inimigo, int y, int x)

  Função responsável pela impressão de unidades inimigas nas coordenadas escolhidas.
- void desenhaedificio (WINDOW ∗win, int nivel, int edificio, int y, int x)

  Função responsável pela impressão do desenho dos edifícios.
- void seta (WINDOW ∗win, int tipo, int y, int x)

  Função responsável pela impressão das setas.
- void seta_batalha (WINDOW ∗winfield, int j, int i)

  Função responsável pela atualização do desenho de seta durante uma batalha.

5.6.1 Descrição detalhada

Módulo de implementação do módulo grafico.h.

5.6.2 Documentação das macros

5.6.2.1 #define ARRAY_SIZE(*a*) (sizeof(a) / sizeof(a[0]))

5.6.2.2 #define CASTLEW 40

5.6.2.3 #define CTRLD 4

5.6.2.4 #define HUDH 7

5.6.3 Documentação das funções

5.6.3.1 void creditos (int *y*, int *x*)

Função responsável pela impressão dos créditos.

REQUISITO: O usuário deve passar as coordenadas da impressão. HIPÓTESE: A função irá imprimir na tela os créditos dos criadores do jogo. ASSERTIVAS DE ENTRADA: os inteiros devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

y	Coordenada x desejada
x	Coordenada y desejada

Retorna

void

5.6.3.2 desenhaedificio (WINDOW * *win*, int *nivel*, int *edificio*, int *y*, int *x*)

Função responsável pela impressão do desenho dos edifícios.

REQUISITO: O usuário deve passar o ponteiro para a janela, o nível do edifício, o tipo do edifício e as coordenadas. HIPÓTESE: A função irá imprimir os desenhos na área de edifícios. ASSERTIVAS DE ENTRADA: win não pode ser nulo, nivel deve estar entre 1 e 3, edificio deve estar entre 0 e 3, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW *win, int nivel, int edificio, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela em que será desenhada o edificio.
nivel	Nível atual do edifício que será desenhado.
edificio	Indentificador de edificio. Dependendo do número, desenha um edificio diferente.
y	Coordenada y desejada.
x	Coordenada x desejada.

Retorna

void

5.6.3.3 void desenhainimigo (WINDOW * *win*, int *inimigo*, int *y*, int *x*)

Função responsável pela impressão de unidades inimigas nas coordenadas escolhidas.

REQUISITO: O usuário deve passar o ponteiro para a janela, o tipo de unidade e as coordenadas. HIPÓTESE: A função irá imprimir no campo de batalha as unidades geradas pelo inimigo. ASSERTIVAS DE ENTRADA: win não pode ser nulo, inimigo deve estar entre 1 e 4, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗win, int inimigo, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela em que será desenhada a unidade
inimigo	Indentificador de unidade. Dependendo do número, desenha uma unidade diferente
y	Coordenada y desejada
x	Coordenada x desejada

Retorna

void

5.6.3.4 void desenhaunidade (WINDOW ∗ *win,* int *unidade,* int *y,* int *x*)

Função responsável pela impressão do desenho da unidade escolhida nas coordenadas indicadas.

REQUISITO: O usuário deve passar o ponteiro para a janela, o tipo de unidade e as coordenadas. HIPÓTESE: A função irá imprimir no campo de batalha as unidades adiquiridas pelo jogador. ASSERTIVAS DE ENTRADA: win não pode ser nulo, unidade deve estar entre 1 e 4, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗win, int unidade, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela em que será desenhada a unidade
unidade	Indentificador de unidade. Dependendo do número, desenha uma unidade diferente
y	Coordenada y desejada
x	Coordenada x desejada

Retorna

void

5.6.3.5 void destroy_win (WINDOW ∗ *local_win*)

Função responsável por apagar as informações de uma janela e deletar sua estrutura.

Parâmetros

local_win	Uma estrutura do tipo WINDOW previamente inicializada.

Retorna

void

5.6.3.6 void finaliza_ncurses ( )

Função responsável por finalizar o modo ncurses.

Retorna

void

5.6.3.7 void inicia_ncurses ( )

Função responsável por iniciar o modo ncurses.

Retorna

void

5.6.3.8 void mensagem (int *y*, int *x*, char *frase[]*)

Função responsável pela impressão de uma mensagem na tela.

REQUISITO: O usuário deve passar as coordenadas da impressão e a mensagem. HIPÓTESE: A função irá imprimir na tela a mensagem passada como parâmetro. ASSERTIVAS DE ENTRADA: os inteiros devem estar entre 0 e LINES e COLS respectivamente, e a frase seve ser uma string não nula. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros int y, int x, char frase[]. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

y	Coordenada y desejada
x	Coordenada x desejada
frase[]	String a ser mostrada ao jogador

Retorna

void

5.6.3.9 char ∗ mensagem_entrada (int *y,* int *x,* char *frase[]*)

Função responsável por imprimir uma mensagem na tela e guardar a informação digitada pelo usuário.

REQUISITO: O usuário deve passar as coordenadas da impressão e a mensagem. - HIPÓTESE: A função irá imprimir na tela a mensagem passada como parâmetro e irá retornar a string digitada. ASSERTIVAS DE ENTRADA: os inteiros devem estar entre 0 e LINES e COLS respectivamente, e a frase seve ser uma string não nula. ASSERTIVAS DE SAÍDA: A saída deve ser um ponteiro para char. INTERFACE EXPLÍCITA: Tipo de retorno char∗, com parâmetros int y, int x, char frase[]. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

y	Coordenada y desejada
x	Coordenada x desejada
frase[]	String a ser mostrada ao jogador

Retorna

string Texto inserido pelo usuario

5.6.3.10 menu (WINDOW ∗ *menuwin*)

Função que cria o menu de opções disponíveis para o usuário na tela inicial.

REQUISITO: O usuário deve passar o ponteiro para a janela. HIPÓTESE: A função irá imprimir na tela o menu. ASSERTIVAS DE ENTRADA: menuwin não pode ser nulo. ASSERTIVAS DE SAÍDA: A saída deve estar entre -1 e 3. INTERFACE EXPLÍCITA-: Tipo de retorno int, com parâmetro WINDOW ∗menuwin. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

menuwin	Ponteiro da janela de menu

Retorna

Opção selecionada pelo jogador

5.6.3.11 int menu_edificio (void)

Função responsável pela criação do menu de evolução dos edifícios.

REQUISITO: A função deve ser chamada no final de uma rodada. HIPÓTESE: A função irá retornar um valor correspondente à uma das opções do menu. ASSERTIVAS DE ENTRADA: A função não possui parâmetros. ASSERTIVAS DE SAÍDA: O retorno deve

ser um número entre 1 e 4. INTERFACE EXPLÍCITA: Tipo de retorno int. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Retorna

Opção selecionada do menu

5.6.3.12 int menu_unidade (Castelo * *castelo*)

Função responsável pela criação do menu de evolução dos edifícios.

REQUISITO: A função deve ser chamada no início de uma rodada. HIPÓTESE: A função irá retornar um valor correspondente à uma das opções do menu. ASSERTIVAS DE ENTRADA: castelo != NULL. ASSERTIVAS DE SAÍDA: O retorno deve ser um número entre 1 e 3. INTERFACE EXPLÍCITA: Tipo de retorno int. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

castelo	Estrutura de Castelo previamente inicializada.

Retorna

Opção selecionada do menu

5.6.3.13 void print_castle (WINDOW * *castle,* Castelo * *castelo*)

Função responsável pela impressão da área de edifícios.

REQUISITO: O usuário deve passar o ponteiro para a janela e o ponteiro para os edifícios. HIPÓTESE: A função irá imprimir na tela a área de edifícios. ASSERTIVAS DE ENTRADA: castle e castelo não podem ser nulos. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW *castle, Castelo *castelo. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

castle	Ponteiro para a janela do castelo, onde ficam os edifícios.
castelo	Ponteiro para a estrutura Castelo

Retorna

void

5.6.3.14 void print_field (WINDOW * *field,* CabecaPFilas * *filas*)

Função responsável pela impressão do campo de batalha.

REQUISITO: O usuário deve passar o ponteiro para a janela e o ponteiro para as filas de personagens. HIPÓTESE: A função irá imprimir na tela o campo de batalha. ASSERTIVAS DE ENTRADA: field e filas não podem ser nulos. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗field, CabecaPFilas ∗filas. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

field	Ponteiro para a janela do campo de batalha do jogo
filas	Ponteiro para a estrutura de filas de unidades e inimigos.

Retorna

void

5.6.3.15 void print_hud (WINDOW ∗ *hud,* int *gold,* int *wave*)

Função responsável pela impressão do HUD superior, contendo título e informações sobre o jogo(ouro disponível e numero da horda).

REQUISITO: O usuário deve passar o ponteiro para a janela, a quantidade de ouro e o número da horda. HIPÓTESE: A função irá imprimir na tela o HUD. ASSERTIVAS DE ENTRADA: hud não pode ser nulo, gold não pode ser negativo e wave deve estar entre 1 e 10. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE - EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗hud,int gold, int wave. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

hud	Ponteiro para a janela do HUD do jogo (Head's Up Display)
wave	Inteiro que determina em qual wave o jogador se encontra
gold	Inteiro que determina o gold atual do usuário

Retorna

void

5.6.3.16 void seta (WINDOW ∗ *win,* int *tipo,* int *y,* int *x*)

Função responsável pela impressão das setas.

REQUISITO: O usuário deve passar o ponteiro para a janela, o tipo da seta e as coordenadas. HIPÓTESE: A função irá imprimir a seta escolhida nas coordenadas. ASSERTIVAS DE ENTRADA: win não pode ser nulo, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗win, int tipo, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela.
tipo	Direção da seta.
y	Coordenada y da impressão.
x	Coordenada x da impressão.

Retorna

void

5.6.3.17 void seta_batalha (WINDOW * *winfield,* int *j,* int *i*)

Função responsável pela atualização do desenho de seta durante uma batalha.

REQUISITO: O usuário deve passar o ponteiro para a janela, a direção e o tipo da seta. HIPÓTESE: A função irá imprimir a seta escolhida. ASSERTIVAS DE ENTRADA: winfield não pode ser nulo, i deve estar entre 1 e 4. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW *winfield, int j, int i. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

winfield	Ponteiro para a janela.
j	Direção da seta.
i	Posição da seta.

Retorna

void

5.6.3.18 void tela_final ( )

Função responsável por imprimir a mensagem de despedida e encerrar o programa.

REQUISITO: Não há requisitos. HIPÓTESE: A função irá imprimir na tela o design da tela final e irá encerrar o programa. ASSERTIVAS DE ENTRADA: A função não possui entrada. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Retorna

void

5.6.3.19 void tela_inicial ( )

Função responsável pela impressão da tela inicial, com desenhos e opções do menu.

REQUISITO: Não há requisitos. HIPÓTESE: A função irá imprimir na tela o design da tela inicial. ASSERTIVAS DE ENTRADA: A função não possui entrada. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Retorna

void

5.6.3.20 TelaGameOver ( )

Função responsável por imprimir a mensagem de fim de jogo e encerrar o programa.

Retorna

void

5.7 Referência ao ficheiro grafico.h

Módulo para gerenciar a interface gráfica.

`#include <stdio.h> #include <ncurses.h>`

Funções

- void inicia_ncurses ()

  Função responsável por iniciar o modo ncurses.
- void finaliza_ncurses ()

  Função responsável por finalizar o modo ncurses.
- void destroy_win (WINDOW ∗local_win)

  Função responsável por apagar as informações de uma janela e deletar sua estrutura.
- void print_hud (WINDOW ∗hub, int gold, int wave)

  Função responsável pela impressão do HUD superior, contendo título e informações sobre o jogo(ouro disponível e numero da horda).
- void print_field ()
- void print_castle ()
- void tela_inicial ()

  Função responsável pela impressão da tela inicial, com desenhos e opções do menu.
- void tela_final ()

  Função responsável por imprimir a mensagem de despedida e encerrar o programa.
- void TelaGameOver ()

  Função responsável por imprimir a mensagem de fim de jogo e encerrar o programa.
- void creditos (int y, int x)

  Função responsável pela impressão dos créditos.

- int menu ()
- int menu_edificio ()

  Função responsável pela criação do menu de evolução dos edifícios.
- int menu_unidade ()
- void desenhaunidade (WINDOW *win, int unidade, int y, int x)

  Função responsável pela impressão do desenho da unidade escolhida nas coordenadas indicadas.
- void desenhaedificio (WINDOW *win, int nivel, int edificio, int y, int x)

  Função responsável pela impressão do desenho dos edifícios.
- void desenhainimigo (WINDOW *win, int inimigo, int y, int x)

  Função responsável pela impressão de unidades inimigas nas coordenadas escolhidas.
- void mensagem (int y, int x, char frase[])

  Função responsável pela impressão de uma mensagem na tela.
- char * mensagem_entrada (int y, int x, char frase[])

  Função responsável por imprimir uma mensagem na tela e guardar a informação digitada pelo usuário.
- void seta (WINDOW *win, int tipo, int y, int x)

  Função responsável pela impressão das setas.
- void seta_batalha (WINDOW *winfield, int j, int i)

  Função responsável pela atualização do desenho de seta durante uma batalha.

5.7.1 Descrição detalhada

Módulo para gerenciar a interface gráfica.

5.7.2 Documentação das funções

5.7.2.1 void creditos (int *y*, int *x*)

Função responsável pela impressão dos créditos.

REQUISITO: O usuário deve passar as coordenadas da impressão. HIPÓTESE: A função irá imprimir na tela os créditos dos criadores do jogo. ASSERTIVAS DE ENTRADA: os inteiros devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

y	Coordenada x desejada
x	Coordenada y desejada

Retorna

void

5.7.2.2 void **desenhaedificio** (WINDOW ∗ *win,* int *nivel,* int *edificio,* int *y,* int *x*)

Função responsável pela impressão do desenho dos edifícios.

REQUISITO: O usuário deve passar o ponteiro para a janela, o nível do edifício, o tipo do edifício e as coordenadas. HIPÓTESE: A função irá imprimir os desenhos na área de edifícios. ASSERTIVAS DE ENTRADA: win não pode ser nulo, nivel deve estar entre 1 e 3, edificio deve estar entre 0 e 3, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗win, int nivel, int edificio, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela em que será desenhada o edificio.
nivel	Nível atual do edifício que será desenhado.
edificio	Indentificador de edificio. Dependendo do número, desenha um edificio diferente.
y	Coordenada y desejada.
x	Coordenada x desejada.

Retorna

void

5.7.2.3 void **desenhainimigo** (WINDOW ∗ *win,* int *inimigo,* int *y,* int *x*)

Função responsável pela impressão de unidades inimigas nas coordenadas escolhidas.

REQUISITO: O usuário deve passar o ponteiro para a janela, o tipo de unidade e as coordenadas. HIPÓTESE: A função irá imprimir no campo de batalha as unidades geradas pelo inimigo. ASSERTIVAS DE ENTRADA: win não pode ser nulo, inimigo deve estar entre 1 e 4, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗win, int inimigo, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela em que será desenhada a unidade
inimigo	Indentificador de unidade. Dependendo do número, desenha uma unidade diferente
y	Coordenada y desejada
x	Coordenada x desejada

Retorna

void

5.7.2.4 void **desenhaunidade** (WINDOW ∗ *win,* int *unidade,* int *y,* int *x*)

Função responsável pela impressão do desenho da unidade escolhida nas coordenadas indicadas.

REQUISITO: O usuário deve passar o ponteiro para a janela, o tipo de unidade e as coordenadas. HIPÓTESE: A função irá imprimir no campo de batalha as unidades adiquiridas pelo jogador. ASSERTIVAS DE ENTRADA: win não pode ser nulo, unidade deve estar entre 1 e 4, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW ∗win, int unidade, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela em que será desenhada a unidade
unidade	Indentificador de unidade. Dependendo do número, desenha uma unidade diferente
y	Coordenada y desejada
x	Coordenada x desejada

Retorna

void

5.7.2.5 void **destroy_win** (WINDOW ∗ *local_win*)

Função responsável por apagar as informações de uma janela e deletar sua estrutura.

Parâmetros

local_win	Uma estrutura do tipo WINDOW previamente inicializada.

Retorna

void

5.7.2.6 void **finaliza_ncurses** ( )

Função responsável por finalizar o modo ncurses.

Retorna

void

5.7.2.7 void inicia_ncurses ( )

Função responsável por iniciar o modo ncurses.

Retorna

void

5.7.2.8 void mensagem (int *y,* int *x,* char *frase[]*)

Função responsável pela impressão de uma mensagem na tela.

REQUISITO: O usuário deve passar as coordenadas da impressão e a mensagem. HIPÓTESE: A função irá imprimir na tela a mensagem passada como parâmetro. ASSERTIVAS DE ENTRADA: os inteiros devem estar entre 0 e LINES e COLS respectivamente, e a frase seve ser uma string não nula. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros int y, int x, char frase[]. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

y	Coordenada y desejada
x	Coordenada x desejada
frase[]	String a ser mostrada ao jogador

Retorna

void

5.7.2.9 char* mensagem_entrada (int *y,* int *x,* char *frase[]*)

Função responsável por imprimir uma mensagem na tela e guardar a informação digitada pelo usuário.

REQUISITO: O usuário deve passar as coordenadas da impressão e a mensagem. - HIPÓTESE: A função irá imprimir na tela a mensagem passada como parâmetro e irá retornar a string digitada. ASSERTIVAS DE ENTRADA: os inteiros devem estar entre 0 e LINES e COLS respectivamente, e a frase seve ser uma string não nula. ASSERTIVAS DE SAÍDA: A saída deve ser um ponteiro para char. INTERFACE EXPLÍCITA: Tipo de retorno char*, com parâmetros int y, int x, char frase[]. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

y	Coordenada y desejada
x	Coordenada x desejada
frase[]	String a ser mostrada ao jogador

Retorna

string Texto inserido pelo usuario

5.7.2.10 int menu ( )

5.7.2.11 int menu_edificio (void)

Função responsável pela criação do menu de evolução dos edifícios.

REQUISITO: A função deve ser chamada no final de uma rodada. HIPÓTESE: A função irá retornar um valor correspondente à uma das opções do menu. ASSERTIVAS DE ENTRADA: A função não possui parâmetros. ASSERTIVAS DE SAÍDA: O retorno deve ser um número entre 1 e 4. INTERFACE EXPLÍCITA: Tipo de retorno int. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Retorna

Opção selecionada do menu

5.7.2.12 int menu_unidade ( )

5.7.2.13 void print_castle ( )

5.7.2.14 void print_field ( )

5.7.2.15 void print_hud (WINDOW * *hud,* int *gold,* int *wave*)

Função responsável pela impressão do HUD superior, contendo título e informações sobre o jogo(ouro disponível e numero da horda).

REQUISITO: O usuário deve passar o ponteiro para a janela, a quantidade de ouro e o número da horda. HIPÓTESE: A função irá imprimir na tela o HUD. ASSERTIVAS DE ENTRADA: hud não pode ser nulo, gold não pode ser negativo e wave deve estar entre 1 e 10. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE - EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW *hud,int gold, int wave. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

hud	Ponteiro para a janela do HUD do jogo (Head's Up Display)
wave	Inteiro que determina em qual wave o jogador se encontra
gold	Inteiro que determina o gold atual do usuário

Retorna

void

5.7.2.16 void seta (WINDOW * *win*, int *tipo*, int *y*, int *x*)

Função responsável pela impressão das setas.

REQUISITO: O usuário deve passar o ponteiro para a janela, o tipo da seta e as coordenadas. HIPÓTESE: A função irá imprimir a seta escolhida nas coordenadas. ASSERTIVAS DE ENTRADA: win não pode ser nulo, y e x devem estar entre 0 e LINES e COLS respectivamente. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW *win, int tipo, int y, int x. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

win	Ponteiro para a janela.
tipo	Direção da seta.
y	Coordenada y da impressão.
x	Coordenada x da impressão.

Retorna

void

5.7.2.17 void seta_batalha (WINDOW * *winfield*, int *j*, int *i*)

Função responsável pela atualização do desenho de seta durante uma batalha.

REQUISITO: O usuário deve passar o ponteiro para a janela, a direção e o tipo da seta. HIPÓTESE: A função irá imprimir a seta escolhida. ASSERTIVAS DE ENTRADA: winfield não pode ser nulo, i deve estar entre 1 e 4. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void, com parâmetros WINDOW *winfield, int j, int i. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Parâmetros

winfield	Ponteiro para a janela.
j	Direção da seta.
i	Posição da seta.

Retorna

void

5.7.2.18 void tela_final ( )

Função responsável por imprimir a mensagem de despedida e encerrar o programa.

REQUISITO: Não há requisitos. HIPÓTESE: A função irá imprimir na tela o design da tela final e irá encerrar o programa. ASSERTIVAS DE ENTRADA: A função não possui entrada. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Retorna

void

5.7.2.19 void tela_inicial ( )

Função responsável pela impressão da tela inicial, com desenhos e opções do menu.

REQUISITO: Não há requisitos. HIPÓTESE: A função irá imprimir na tela o design da tela inicial. ASSERTIVAS DE ENTRADA: A função não possui entrada. ASSERTIVAS DE SAÍDA: A função não possui saída. INTERFACE EXPLÍCITA: Tipo de retorno void. INTERFACE IMPLÍCITA: variáveis e funções auxiliares.

Retorna

void

5.7.2.20 void TelaGameOver ( )

Função responsável por imprimir a mensagem de fim de jogo e encerrar o programa.

Retorna

void

5.8 Referência ao ficheiro peixera.c

Módulo da main do jogo.

```
#include <ncurses.h> #include <menu.h> #include <unistd.-
h> #include <stdio.h> #include <stdlib.h> #include <string.-
h> #include <time.h> #include "grafico.h" #include "estruturas.-
h" #include "engine.h"
```

Macros

- #define HUBH 7
- #define CASTLEW 40
- #define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
- #define CTRLD 4

Funções

- int main ()

5.8.1 Descrição detalhada

Módulo da main do jogo.

5.8.2 Documentação das macros

5.8.2.1 #define ARRAY_SIZE(*a*) (sizeof(a) / sizeof(a[0]))

5.8.2.2 #define CASTLEW 40

5.8.2.3 #define CTRLD 4

5.8.2.4 #define HUBH 7

5.8.3 Documentação das funções

5.8.3.1 int main ( )