Bottom text

Grupo 2
Versão 1.0
Domingo, 2 de Dezembro de 2018

Índice dos Componentes

Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Classes

Referência da Estrutura Atributos_Data

`#include <funcoes.h>`

Atributos Públicos

- int **classe**
- int **divisao**
- int **time**
- int **nivel**

Atributos

int Atributos_Data::classe

int Atributos_Data::divisao

int Atributos_Data::nivel

int Atributos_Data::time

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Base_Principal

```
#include <funcoes.h>
```

Atributos Públicos

- int **vida**
- int **time**
- int **dim**
- int **i**
- int **j**

Atributos

int Base_Principal::dim

int Base_Principal::i

int Base_Principal::j

int Base_Principal::time

int Base_Principal::vida

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Cell_Mapa

`#include <funcoes.h>`

Atributos Públicos

- int **x**
- int **y**
- int **lado**
- **unidade_movel * pUniMovel**
- **unidade_estatica * pUniImovel**
- **base_principal * pBase**

Atributos

int Cell_Mapa::lado

base_principal* Cell_Mapa::pBase

unidade_estatica* Cell_Mapa::pUniImovel

unidade_movel* Cell_Mapa::pUniMovel

int Cell_Mapa::x

int Cell_Mapa::y

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Imagens_Data

`#include <funcoes.h>`

Atributos Públicos

- unsigned int **textura_grade**
- unsigned int **textura_fundo**
- unsigned int **textura_base**
- unsigned int **textura_menu_principal**
- unsigned int **humano_GER_REC**
- unsigned int **humano_GER_TROP**
- unsigned int **humano_DEF_OFS**
- unsigned int **humano_DEF_PAS**
- unsigned int **mecanico_GER_REC**
- unsigned int **mecanico_GER_TROP**
- unsigned int **mecanico_DEF_OFS**
- unsigned int **mecanico_DEF_PAS**
- unsigned int **eletrico_GER_REC**
- unsigned int **eletrico_GER_TROP**
- unsigned int **eletrico_DEF_OFS**
- unsigned int **eletrico_DEF_PAS**
- unsigned int **minerio**
- unsigned int **raio**
- unsigned int **comida**
- unsigned int **operario**
- unsigned int **botao1**
- unsigned int **botao2**

Atributos

unsigned int Imagens_Data::botao1

unsigned int Imagens_Data::botao2

unsigned int Imagens_Data::comida

unsigned int Imagens_Data::eletrico_DEF_OFS

unsigned int Imagens_Data::eletrico_DEF_PAS

unsigned int Imagens_Data::eletrico_GER_REC

unsigned int Imagens_Data::eletrico_GER_TROP

unsigned int Imagens_Data::humano_DEF_OFS

unsigned int Imagens_Data::humano_DEF_PAS

unsigned int Imagens_Data::humano_GER_REC

unsigned int Imagens_Data::humano_GER_TROP

unsigned int Imagens_Data::mecanico_DEF_OFS

unsigned int Imagens_Data::mecanico_DEF_PAS

unsigned int Imagens_Data::mecanico_GER_REC

unsigned int Imagens_Data::mecanico_GER_TROP

unsigned int Imagens_Data::minerio

unsigned int Imagens_Data::operario

unsigned int Imagens_Data::raio

unsigned int Imagens_Data::textura_base

unsigned int Imagens_Data::textura_fundo

unsigned int Imagens_Data::textura_grade

unsigned int Imagens_Data::textura_menu_principal

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Mouse_Data

`#include <funcoes.h>`

Atributos Públicos

- int **x**
- int **y**
- int **x_mem**
- int **y_mem**
- int **x_agr**
- int **y_agr**
- int **x_botao**
- int **y_botao**
- int **botao_mem**

Atributos

int Mouse_Data::botao_mem

int Mouse_Data::x

int Mouse_Data::x_agr

int Mouse_Data::x_botao

int Mouse_Data::x_mem

int Mouse_Data::y

int Mouse_Data::y_agr

int Mouse_Data::y_botao

int Mouse_Data::y_mem

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Player_Data

`#include <funcoes.h>`

Atributos Públicos

- int **time**
- int **comida**
- int **minerio**
- int **eletricidade**
- int **xp**
- int **nivel**
- int **vida_base**

Atributos

int Player_Data::comida

int Player_Data::eletricidade

int Player_Data::minerio

int Player_Data::nivel

int Player_Data::time

int Player_Data::vida_base

int Player_Data::xp

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Texto_Data

`#include <funcoes.h>`

Atributos Públicos

- unsigned int **numero_textura** [101]
- unsigned int **nome_textura** [35]

Atributos

unsigned int Texto_Data::nome_textura[35]

unsigned int Texto_Data::numero_textura[101]

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Unidade_Estatica

`#include <funcoes.h>`

Atributos Públicos

- int **vida**
- int **ataque**
- int **defesa**
- int **alcance**
- int **classe**
- int **divisao**
- int **time**
- int **nivel**
- int **dim**
- int **i**
- int **j**
- int **custo_minerio**
- int **custo_comida**
- int **custo_eletricidade**
- int **producao**
- bool **acao**
- bool **cont_ataque**

Atributos

bool Unidade_Estatica::acao

int Unidade_Estatica::alcance

int Unidade_Estatica::ataque

int Unidade_Estatica::classe

bool Unidade_Estatica::cont_ataque

int Unidade_Estatica::custo_comida

int Unidade_Estatica::custo_eletricidade

int Unidade_Estatica::custo_minerio

int Unidade_Estatica::defesa

int Unidade_Estatica::dim

int Unidade_Estatica::divisao

int Unidade_Estatica::i

int Unidade_Estatica::j

int Unidade_Estatica::nivel

int Unidade_Estatica::producao

int Unidade_Estatica::time

int Unidade_Estatica::vida

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Referência da Estrutura Unidade_Movel

`#include <funcoes.h>`

Atributos Públicos

- int **vida**
- int **ataque**
- int **defesa**
- int **alcance**
- int **velocidade**
- int **divisao**
- int **time**
- int **nivel**
- int **dim**
- int **i**
- int **j**
- int **custo_minerio**
- int **custo_comida**
- int **custo_eletricidade**
- bool **acao**
- bool **cont_ataque**

Atributos

bool Unidade_Movel::acao

int Unidade_Movel::alcance

int Unidade_Movel::ataque

bool Unidade_Movel::cont_ataque

int Unidade_Movel::custo_comida

int Unidade_Movel::custo_eletricidade

int Unidade_Movel::custo_minerio

int Unidade_Movel::defesa

int Unidade_Movel::dim

int Unidade_Movel::divisao

int Unidade_Movel::i

int Unidade_Movel::j

int Unidade_Movel::nivel

int Unidade_Movel::time

int Unidade_Movel::velocidade

int Unidade_Movel::vida

A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

- **funcoes.h**

Arquivos

Referência do Arquivo cpu.cpp

Arquivo com a aplicação da inteligencia artificial da CPU para simular o oponente.

```
#include "funcoes.h"
```

Funções

- bool **verificador_CPU** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j)
- void **criacoes_iniciais_1** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **criacoes_iniciais_2** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **criacoes_iniciais_3** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_defesa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_geradores** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_evolucoes** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_tropas** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_movimentacao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_destruicao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_combate** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_destruicao_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- void **priorizar_contra_ataque** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player_CPU)
- int **CPU** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **player_data** *player_CPU, int contador_turno)
  Definição da CPU. Função que define ações do jogador adversário.

Descrição detalhada

Arquivo com a aplicação da inteligencia artificial da CPU para simular o oponente.

Autor:
Grupo 2

Funções

int CPU (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], player_data * *player_CPU*, int *contador_turno*)

Definição da CPU. Função que define ações do jogador adversário.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
player_data	*player_CPU - Informações do adversário.
int	contador_turno - Marcador de turno.

Retorna:

0 - Se o procedimento foi bem sucedido.

void criacoes_iniciais_1 (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void criacoes_iniciais_2 (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void criacoes_iniciais_3 (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_combate (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_contra_ataque (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_defesa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_destruicao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_destruicao_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_evolucoes (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_geradores (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_movimentacao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

void priorizar_tropas (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player_CPU*)

bool verificador_CPU (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*)

Referência do Arquivo criacao.cpp

Arquivo com as funções relacionadas à criação de elementos do jogo.
`#include "funcoes.h"`

Funções

- int **cria_mapa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Cria Mapa.. Função que carrega o mapa do jogo.
- void **cria_player** (**player_data** *player, int **time**)
  Criação de jogador. Função que cria determinado jogador.
- int **cria_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j, int dim, int vida, int **time**)
  Carrega a base do jogador. Função que carrega a base de cada participante do jogo, com seus devidos atributos.
- int **cria_uni_estatico** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j, **atributos_data** atributos, **player_data** *player)
  Cria unidade estática. Função que cria a unidade estática desejada (predios).
- int **cria_uni_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j, **atributos_data** atributos, **player_data** *player)
  Cria a unidade movel. Função que cria as unidades móveis no mapa, com seus atributos respectivos.
- int **construction** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, int cell_i, int cell_j, **unidade_movel** *unit, **player_data** *player)
  Construção. Função que constrói determinada unidade.

Descrição detalhada

Arquivo com as funções relacionadas à criação de elementos do jogo.

Autor:
Grupo 2

Funções

int construction (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, int *cell_i*, int *cell_j*, unidade_movel * *unit*, player_data * *player*)

Construção. Função que constrói determinada unidade.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
atributos_data	atributos - Atributos do jogador.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.
*unidade_movel**	unit -Unidade movel emvolvida na construção .
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int cria_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*, int *dim*, int *vida*, int *time*)

Carrega a base do jogador. Função que carrega a base de cada participante do jogo, com seus devidos atributos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da base.
int	j - Auxiliar de coordenada da base.
int	dim - Dimensão.
int	vida - Vida da base.
int	time - O time a qual a base é atribuída.

Retorna:

0 - Se o procedimento foi bem sucedido.

int cria_mapa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Cria Mapa.. Função que carrega o mapa do jogo.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:

0 - Se o procedimento foi bem sucedido.

void cria_player (player_data * *player*, int *time*)

Criação de jogador. Função que cria determinado jogador.

Parâmetros:

player_data	*player - Informações do jogador.
int	time - tempo da ocorrência.

Retorna:

Nada (função tipo void).

int cria_uni_estatico (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*, atributos_data *atributos*, player_data * *player*)

Cria unidade estática. Função que cria a unidade estática desejada (predios).

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.
atributos_data	atributos - Atributos do jogador.

player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int cria_uni_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*, atributos_data *atributos*, player_data * *player*)

Cria a unidade movel. Função que cria as unidades móveis no mapa, com seus atributos respectivos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

Referência do Arquivo funcoes.h

```
#include <iostream>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#include <sys/time.h>
#include <stdlib.h>
#include <SDL/SDL_ttf.h>
#include <SDL/SDL.h>
#include "SDL/SDL_opengl.h"
#include "SDL/SDL_image.h"
#include "string"
```

Componentes

- struct **Player_Data**
- struct **Unidade_Movel**
- struct **Unidade_Estatica**
- struct **Base_Principal**
- struct **Cell_Mapa**
- struct **Imagens_Data**
- struct **Mouse_Data**
- struct **Atributos_Data**
- struct **Texto_Data**

Definições e Macros

- #define **RX** 900
- #define **RY** 600
- #define **BLOCOS_LINHA** 40
- #define **DIMENSAO_COMANDANTE** 0.2 * **RY**
- #define **TRANSLADA_COMANDANTE** 0.02 * **RX**
- #define **DIVISAO_INFERIOR** 0.08 * **RY**
- #define **DIMENSAO_ICONES** 0.05 * **RY**
- #define **TAMANHO_TEXTO_ICONES** 2.2 * **DIMENSAO_ICONES**
- #define **LARGURA_BARRAS** 0.015 * **RY**
- #define **VIDA_INICIAL_BASE** 99
- #define **COMIDA_INICIAL** 50
- #define **MINERIO_INICIAL** 50
- #define **ELETRICIDADE_INICIAL** 50

Definições de Tipos

- typedef struct **Player_Data player_data**
- typedef struct **Unidade_Movel unidade_movel**
- typedef struct **Unidade_Estatica unidade_estatica**
- typedef struct **Base_Principal base_principal**
- typedef struct **Cell_Mapa cell_mapa**
- typedef struct **Imagens_Data imagens_data**
- typedef struct **Mouse_Data mouse_data**
- typedef struct **Atributos_Data atributos_data**
- typedef struct **Texto_Data texto_data**

Enumerações

- enum **divisao** { **HUMANO** = 0, **MECANICO**, **ELETRICO**, **OPERARIO** }

- enum **classe** { **GERADOR_DE_RECURSO** = 4, **GERADOR_DE_TROPA**, **DEFESA_OFENSIVA**, **DEFESA_PASSIVA** }
- enum **unidades** { **REPLICANTE** = 8, **EXTERMINADOR**, **HATSUNE**, **WALL**, **DROIDES**, **IRON**, **MERCENARIOS**, **CAVALEIROS**, **CHORIS**, **CUSTO** }
- enum **tipo** { **UNIDADE** = 18, **ESTRUTURA**, **BASE**, **VAZIO** }
- enum **criacao** { **GERAR_OPERARIO** = 22, **CRIAR_GER_REC**, **CRIAR_GER_TRO**, **CRIAR_MUR**, **GERAR_TROPA**, **CRIAR_DEFESA_OF**, **CRIAR_DEFESA** }
- enum **niveis** { **EVOLUIR** = 29, **NIVEL_MAXIMO**, **NIVEL_1**, **NIVEL_2**, **NIVEL_3**, **EVOLUIR_NIVEL_INSUFICIENTE** }
- enum **time** { **ALIADO** = 0, **INIMIGO** }

Funções

- int **min** (int a, int b)
- int **max** (int a, int b)
- GLuint **loadTexture** (const std::string &fileName)
  Carrega Texturas. Função que carrega as texturas atribuídas.
- GLuint **importText** (const std::string &text, int font_size, int red, int green, int blue)
  Carrega Texto. Função que carrega os textos atribuídos.
- int **cria_mapa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Cria Mapa.. Função que carrega o mapa do jogo.
- int **carrega_interface** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **mouse_data** *mouse, **texto_data** texto, **player_data** *player, **atributos_data** atributos)
  Carrega Texto. Função que carrega os textos atribuídos.
- int **cria_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j, int dim, int vida, int **time**)
  Carrega a base do jogador. Função que carrega a base de cada participante do jogo, com seus devidos atributos.
- int **carrega_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens)
  Carrega base. Função que carrega a base do jogador, com seus atributos.
- bool **verifica_imagem** (const std::string &fileName)
  Verifica a imagem. Função que verifica se uma imagem foi carregada corretamente.
- bool **verifica_espaco** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j)
  Verifica espaço do mapa. Função que verifica se um um espaço no mapa está livre.
- int **verifica_selecao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** mouse)
  Verifica conteúdo espaço do mapa. Função que verifica se um espaço no mapa está ocupado por estrutura determinada.
- int **verifica_unidades** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **player_data** *player, **atributos_data** atributos, **imagens_data** imagens, **texto_data** texto)
  Verifica unidades. Função que verifica um ações entre unidades, como locomoção, combate e destruição.
- bool **verifica_velocidade** (**unidade_movel** *aux, int cell_i, int cell_j)
  Verifica velocidade. Função que verifica determinada velocidade.
- bool **verifica_alcance** (**unidade_movel** *aux, int cell_i, int cell_j)
  Verifica alcance. Função que verifica determinado alcance.
- bool **verifica_alcance_defesa** (**unidade_estatica** *aux, int cell_i, int cell_j)
  Verifica alcance da defesa. Função que verifica determinado alcance de situação de defesa.
- bool **verifica_oposicao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, int i, int j)
  Verifica oposição. Função que verifica oposição de uma ação.

- bool **verifica_oposicao_defesa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, int i, int j)
  Verifica oposição da defesa. Função que verifica oposição à defesa..
- int **cria_uni_estatico** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j, **atributos_data** atributos, **player_data** *player)
  Cria unidade estática. Função que cria a unidade estática desejada (predios).
- int **carrega_uni_estatico** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens)
  Carrega unidade estática. Função que renderiza a unidade estática desejada (predios).
- int **carrega_display_recursos** (**imagens_data** imagens, **texto_data** texto)
  Carrega informações de recursos. Função que carrega os recursos do jogador.
- int **carrega_mapa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **mouse_data** mouse)
  Carrega o mapa. Função que carrega o mapa do jogo.
- int **carrega_layout** ()
  Carrega layout. Função que carrega o layout para a criação de outros elementos do jogo.
- int **colore_espacos_validos** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **player_data** *player)
  Colore espaços. Função que colore espaços no mapa, se estão vazio.
- int **colore_espacos_validos_defesa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, **player_data** *player)
  Colore espaços de defesa. Função que colore espaços no mapa, se são espaços válidos de defesa.
- int **carrega_uni_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **texto_data** texto)
  Carrega a unidade movel. Função que carrega as imagens de unidades móveis no mapa.
- int **cria_uni_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j, **atributos_data** atributos, **player_data** *player)
  Cria a unidade movel. Função que cria as unidades móveis no mapa, com seus atributos respectivos.
- int **carrega_numeros_recurso** (**texto_data** texto, **player_data** *player)
  Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.
- int **move_unidade** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *unit, int i, int j)
  Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.
- int **carrega_barras** (**imagens_data** imagens)
  Carrega barras. Função que carrega a barra lateral da interface do jogador, onde ficam as principais informações.
- int **combate** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **unidade_movel** *aux2, **player_data** *player)
  Combate. Função que executa a atualização de informações de combate entre unidade móveis.
- int **combate_defensivo** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, **unidade_movel** *aux2, **player_data** *player)
  Combate defensivo. Função que executa a atualização de informações de combate (defesa) entre unidade móveis.
- int **destruicao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **unidade_estatica** *aux2, **player_data** *player)
  Destruição de construção. Função que executa a atualização de informações de combate entre uma unidade e um prédio.
- int **destruicao_defensiva** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, **unidade_estatica** *aux2, **player_data** *player)
  Destruição de construção de defesa. Função que executa a atualização de informações de combate entre uma unidade e um prédio.

- int **carrega_comandante** (**imagens_data** imagens)
  Carrega comandante. Função que carrega a imagem de um comandante.
- int **carrega_caixa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **imagens_data** imagens, **texto_data** texto, **atributos_data** atributos, **player_data** *player)
  Carrega a caixa. Função que carrega a caixa do jogador.
- int **escolhe_imagem_estatica** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, int i, int j)
  Escolhe imagem estática. Função que define a imagem de unidade estática desejada.
- int **escolhe_imagem_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **texto_data** texto, int i, int j, int opcao)
  Escolhe imagem móvel. Função que define a imagem de unidade móvel desejada.
- int **escolhe_texto_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **texto_data** texto, int i, int j)
  Escolhe texto. Função que define o texto a ser exibido.
- int **construction** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, int cell_i, int cell_j, **unidade_movel** *unit, **player_data** *player)
  Construção. Função que constrói determinada unidade.
- void **Atualizar_recursos** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **player_data** *player)
  Atualização de recursos. Função que recursos de determinado jogador.
- int **player_level** (**player_data** *player)
  Definição de nível. Função que define nível de determinado jogador.
- int **evolution** (**unidade_estatica** *aux, **player_data** *player)
  Evolui construções. Função que evolui determinada construção.
- int **carrega_botao** (**imagens_data** imagens, **texto_data** texto, **mouse_data** *mouse, int local, int **tipo**, **cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player)
  Carrega botão. Função que carrega botões que interagem com o jogador.
- int **gera_operario** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **atributos_data** atributos, **player_data** *player)
  Gera operário. Função que gera os operários do jogador.
- int **gera_tropa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **atributos_data** atributos, **player_data** *player)
  Geração de tropas. Função que gera as tropas de determinado jogador.
- int **CPU** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **player_data** *player_CPU, int contador_turno)
  Definição da CPU. Função que define ações do jogador adversário.
- void **cria_player** (**player_data** *player, int **time**)
  Criação de jogador. Função que cria determinado jogador.
- void **restaurar_acoes** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Definição de nível. Função que define nível de determinado jogador.
- int **destruicao_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **base_principal** *aux2, **player_data** *player)
  Salvar jogo. Função que carrega o jogo anterior do jogador.
- void **salva_jogo** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Salvar jogo. Função que carrega o jogo anterior do jogador.
- void **carrega_jogo** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Carregar jogo. Função que carrega o jogo anterior do jogador.

Variáveis

- const int **LADO = RY/BLOCOS_LINHA**

Definições e macros

#define BLOCOS_LINHA 40

#define COMIDA_INICIAL 50

#define DIMENSAO_COMANDANTE 0.2 * RY

#define DIMENSAO_ICONES 0.05 * RY

#define DIVISAO_INFERIOR 0.08 * RY

#define ELETRICIDADE_INICIAL 50

#define LARGURA_BARRAS 0.015 * RY

#define MINERIO_INICIAL 50

#define RX 900

#define RY 600

#define TAMANHO_TEXTO_ICONES 2.2 * DIMENSAO_ICONES

#define TRANSLADA_COMANDANTE 0.02 * RX

#define VIDA_INICIAL_BASE 99

Definições dos tipos

typedef struct Atributos_Data atributos_data

typedef struct Base_Principal base_principal

typedef struct Cell_Mapa cell_mapa

typedef struct Imagens_Data imagens_data

typedef struct Mouse_Data mouse_data

typedef struct Player_Data player_data

typedef struct Texto_Data texto_data

typedef struct Unidade_Estatica unidade_estatica

typedef struct Unidade_Movel unidade_movel

Enumerações

enum classe

Enumeradores:

GERADOR_DE_RECURSO
GERADOR_DE_TROPA
DEFESA_OFENSIVA
DEFESA_PASSIVA

enum criacao

Enumeradores:

GERAR_OPERARIO
CRIAR_GER_REC
CRIAR_GER_TRO
CRIAR_MUR
GERAR_TROPA
CRIAR_DEFESA_OF
CRIAR_DEFESA

enum divisao

Enumeradores:

HUMANO
MECANICO
ELETRICO
OPERARIO

enum niveis

Enumeradores:

EVOLUIR
NIVEL_MAXIMO
NIVEL_1
NIVEL_2
NIVEL_3
EVOLUIR_NIVEL_INSUFICIENTE

enum time

Enumeradores:

ALIADO
INIMIGO

enum tipo

Enumeradores:

UNIDADE
ESTRUTURA
BASE
VAZIO

enum unidades

Enumeradores:

REPLICANTE
EXTERMINADOR
HATSUNE
WALL
DROIDES
IRON
MERCENARIOS
CAVALEIROS
CHORIS
CUSTO

Funções

void Atualizar_recursos (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], player_data * *player*)

Atualização de recursos. Função que recursos de determinado jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_barras (imagens_data *imagens*)

Carrega barras. Função que carrega a barra lateral da interface do jogador, onde ficam as principais informações.

Parâmetros:

imagens_data	imagens - Imagens carregadas.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*)

Carrega base. Função que carrega a base do jogador, com seus atributos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_botao (imagens_data *imagens*, texto_data *texto*, mouse_data * *mouse*, int *local*, int *tipo*, cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player*)

Carrega botão. Função que carrega botões que interagem com o jogador.

Parâmetros:

imagens_data	imagens - Imagens carregadas.
mouse_data	mouse - Informações de localização do mouse do usuário.
int	local - local onde se amostra o botão.
int	tipo - tipo de botão.
cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_caixa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, imagens_data *imagens*, texto_data *texto*, atributos_data *atributos*, player_data * *player*)

Carrega a caixa. Função que carrega a caixa do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_comandante (imagens_data *imagens*)

Carrega comandante. Função que carrega a imagem de um comandante.

Parâmetros:

imagens_data	imagens - Imagens carregadas.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_display_recursos (imagens_data *imagens*, texto_data *texto*)

Carrega informações de recursos. Função que carrega os recursos do jogador.

Parâmetros:

imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_interface (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, mouse_data * *mouse*, texto_data *texto*, player_data * *player*, atributos_data *atributos*)

Carrega Texto. Função que carrega os textos atribuídos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
mouse_data	mouse - Informações de localização do mouse do usuário.
texto_data	texto - Texto a ser carregado.

player_data	*player - Informações de jogador.
atributos_data	atributos - Atributos do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

void carrega_jogo (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Carregar jogo. Função que carrega o jogo anterior do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:

Nada (função tipo void).

int carrega_layout ()

Carrega layout. Função que carrega o layout para a criação de outros elementos do jogo.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_mapa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, mouse_data *mouse*)

Carrega o mapa. Função que carrega o mapa do jogo.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
mouse_data	mouse - Informações de localização do mouse do usuário.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_numeros_recurso (texto_data *texto*, player_data * *player*)

Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.

Parâmetros:

texto_data	texto - Texto a ser carregado.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_uni_estatico (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*)

Carrega unidade estática. Função que renderiza a unidade estática desejada (predios).

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_uni_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, texto_data *texto*)

Carrega a unidade movel. Função que carrega as imagens de unidades móveis no mapa.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.

Retorna:

0 - Se o procedimento foi bem sucedido.

int colore_espacos_validos (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, player_data * *player*)

Colore espaços. Função que colore espaços no mapa, se estão vazio.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	aux - A unidade móvel selecionada.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int colore_espacos_validos_defesa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, player_data * *player*)

Colore espaços de defesa. Função que colore espaços no mapa, se são espaços válidos de defesa.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_estatica**	aux - A unidade estatica selecionada.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int combate (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, unidade_movel * *aux2*, player_data * *player*)

Combate. Função que executa a atualização de informações de combate entre unidade móveis.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate
unidade_movel	*aux2 - Unidade envolvida no combate
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int combate_defensivo (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, unidade_movel * *aux2*, player_data * *player*)

Combate defensivo. Função que executa a atualização de informações de combate (defesa) entre unidade móveis.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate
unidade_movel	*aux2 - Unidade envolvida no combate
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int construction (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, int *cell_i*, int *cell_j*, unidade_movel * *unit*, player_data * *player*)

Construção. Função que constrói determinada unidade.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
atributos_data	atributos - Atributos do jogador.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.
*unidade_movel**	unit -Unidade movel emvolvida na construção .
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int CPU (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], player_data * *player_CPU*, int *contador_turno*)

Definição da CPU. Função que define ações do jogador adversário.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
player_data	*player_CPU - Informações do adversário.
int	contador_turno - Marcador de turno.

Retorna:

0 - Se o procedimento foi bem sucedido.

int cria_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*, int *dim*, int *vida*, int *time*)

Carrega a base do jogador. Função que carrega a base de cada participante do jogo, com seus devidos atributos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da base.
int	j - Auxiliar de coordenada da base.
int	dim - Dimensão.
int	vida - Vida da base.
int	time - O time a qual a base é atribuída.

Retorna:

0 - Se o procedimento foi bem sucedido.

int cria_mapa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Cria Mapa.. Função que carrega o mapa do jogo.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:

0 - Se o procedimento foi bem sucedido.

void cria_player (player_data * *player*, int *time*)

Criação de jogador. Função que cria determinado jogador.

Parâmetros:

player_data	*player - Informações do jogador.
int	time - tempo da ocorrência.

Retorna:

Nada (função tipo void).

int cria_uni_estatico (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*, atributos_data *atributos*, player_data * *player*)

Cria unidade estática. Função que cria a unidade estática desejada (predios).

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int cria_uni_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*, atributos_data *atributos*, player_data * *player*)

Cria a unidade movel. Função que cria as unidades móveis no mapa, com seus atributos respectivos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int destruicao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, unidade_estatica * *aux2*, player_data * *player*)

Destruição de construção. Função que executa a atualização de informações de combate entre uma unidade e um prédio.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate.
unidade_estatica	*aux2 - Unidade envolvida no combate.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int destruicao_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, base_principal * *aux2*, player_data * *player*)

Salvar jogo. Função que carrega o jogo anterior do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux -Unidade movel envolvida na destruição .
base_principal	*aux2 - Base envolvida na destruição .

player_data	*player - Informações de jogador.

Retorna:

Nada (função tipo void).

int destruicao_defensiva (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, unidade_estatica * *aux2*, player_data * *player*)

Destruição de construção de defesa. Função que executa a atualização de informações de combate entre uma unidade e um prédio.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate.
unidade_estatica	*aux2 - Unidade envolvida no combate.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int escolhe_imagem_estatica (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, int *i*, int *j*)

Escolhe imagem estática. Função que define a imagem de unidade estática desejada.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

0 - Se o procedimento foi bem sucedido.

int escolhe_imagem_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, texto_data *texto*, int *i*, int *j*, int *opcao*)

Escolhe imagem móvel. Função que define a imagem de unidade móvel desejada.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.
int	opcao - Auxiliar que define caso de execução

Retorna:

0 - Se o procedimento foi bem sucedido.

int escolhe_texto_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], texto_data *texto*, int *i*, int *j*)

Escolhe texto. Função que define o texto a ser exibido.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
texto_data	texto - Texto a ser carregado.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

0 - Se o procedimento foi bem sucedido.

int evolution (unidade_estatica * *aux*, player_data * *player*)

Evolui construções. Função que evolui determinada construção.

Parâmetros:

unidade_estatica	aux2 - Unidade envolvida na evolução.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int gera_operario (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, atributos_data *atributos*, player_data * *player*)

Gera operário. Função que gera os operários do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int gera_tropa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, atributos_data *atributos*, player_data * *player*)

Geração de tropas. Função que gera as tropas de determinado jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

GLuint importText (const std::string & *text*, int *font_size*, int *red*, int *green*, int *blue*)

Carrega Texto. Função que carrega os textos atribuídos.

Parâmetros:

const	std::string &text - Texto a ser renderizado.
int	font_size - Tamanho do texto.
int	red - Valor de vermelho.
int	green - Valor de verde.
int	blue - Valor de azul.

Retorna:

Texto- O texto renderizado.

GLuint loadTexture (const std::string & *fileName*)

Carrega Texturas. Função que carrega as texturas atribuídas.

Parâmetros:

const	std::string&fileName - Nome de arquivo.

Retorna:

Imagem- O objeto renderizado.

int max (int *a*, int *b*)

int min (int *a*, int *b*)

int move_unidade (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *unit*, int *i*, int *j*)

Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	unit - A unidade móvel selecionada.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

0 - Se o procedimento foi bem sucedido.

int player_level (player_data * *player*)

Definição de nível. Função que define nível de determinado jogador.

Parâmetros:

player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

void restaurar_acoes (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Definição de nível. Função que define nível de determinado jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:

Nada (função tipo void).

void salva_jogo (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Salvar jogo. Função que carrega o jogo anterior do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:

Nada (função tipo void).

bool verifica_alcance (unidade_movel * *aux*, int *cell_i*, int *cell_j*)

Verifica alcance. Função que verifica determinado alcance.

Parâmetros:

*unidade_movel**	aux - A unidade móvel selecionada.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

bool verifica_alcance_defesa (unidade_estatica * *aux*, int *cell_i*, int *cell_j*)

Verifica alcance da defesa. Função que verifica determinado alcance de situação de defesa.

Parâmetros:

*unidade_movel**	aux - A unidade móvel selecionada.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

bool verifica_espaco (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*)

Verifica espaço do mapa. Função que verifica se um um espaço no mapa está livre.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

true - Se o procedimento foi bem sucedido.

bool verifica_imagem (const std::string & *fileName*)

Verifica a imagem. Função que verifica se uma imagem foi carregada corretamente.

Parâmetros:

const	std::string&fileName - Arquivo que se deseja verificar.

Retorna:

true - Se o procedimento foi bem sucedido.

bool verifica_oposicao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, int *i*, int *j*)

Verifica oposição. Função que verifica oposição de uma ação.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	aux - A unidade móvel selecionada.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

bool verifica_oposicao_defesa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, int *i*, int *j*)

Verifica oposição da defesa. Função que verifica oposição à defesa..

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	aux - A unidade móvel selecionada.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

int verifica_selecao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data *mouse*)

Verifica conteúdo espaço do mapa. Função que verifica se um espaço no mapa está ocupado por estrutura determinada.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.

Retorna:

0 - Se o espaço não está ocupado por nenhuma estrutura definida.

int verifica_unidades (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, player_data * *player*, atributos_data *atributos*, imagens_data *imagens*, texto_data *texto*)

Verifica unidades. Função que verifica um ações entre unidades, como locomoção, combate e destruição.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
player_data	*player - Informações de jogador.
atributos_data	atributos - Atributos do jogador.

Retorna:

0 - Se tudo ocorre dentro do esperado.

bool verifica_velocidade (unidade_movel * *aux*, int *cell_i*, int *cell_j*)

Verifica velocidade. Função que verifica determinada velocidade.

Parâmetros:

*unidade_movel**	aux - A unidade móvel selecionada.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

Variáveis

const int LADO = RY/BLOCOS_LINHA

Referência do Arquivo interacoes.cpp

Arquivo com a aplicação das funções da interação entre elementos do jogo.
`#include "funcoes.h"`

Funções

- int **move_unidade** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *unit, int i, int j)
  Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.
- int **combate** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **unidade_movel** *aux2, **player_data** *player)
  Combate. Função que executa a atualização de informações de combate entre unidade móveis.
- int **combate_defensivo** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, **unidade_movel** *aux2, **player_data** *player)
  Combate defensivo. Função que executa a atualização de informações de combate (defesa) entre unidade móveis.
- int **destruicao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **unidade_estatica** *aux2, **player_data** *player)
  Destruição de construção. Função que executa a atualização de informações de combate entre uma unidade e um prédio.
- int **destruicao_defensiva** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, **unidade_estatica** *aux2, **player_data** *player)
  Destruição de construção de defesa. Função que executa a atualização de informações de combate entre uma unidade e um prédio.
- int **destruicao_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **base_principal** *aux2, **player_data** *player)
  Salvar jogo. Função que carrega o jogo anterior do jogador.
- void **Atualizar_recursos** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **player_data** *player)
  Atualização de recursos. Função que recursos de determinado jogador.
- int **player_level** (**player_data** *player)
  Definição de nível. Função que define nível de determinado jogador.
- int **evolution** (**unidade_estatica** *aux, **player_data** *player)
  Evolui construções. Função que evolui determinada construção.
- int **gera_operario** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **atributos_data** atributos, **player_data** *player)
  Gera operário. Função que gera os operários do jogador.
- int **gera_tropa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **atributos_data** atributos, **player_data** *player)
  Geração de tropas. Função que gera as tropas de determinado jogador.
- void **restaurar_acoes** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Definição de nível. Função que define nível de determinado jogador.

Descrição detalhada

Arquivo com a aplicação das funções da interação entre elementos do jogo.

Autor:
Grupo 2

Funções

void Atualizar_recursos (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], player_data * *player*)

Atualização de recursos. Função que recursos de determinado jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int combate (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, unidade_movel * *aux2*, player_data * *player*)

Combate. Função que executa a atualização de informações de combate entre unidade móveis.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate
unidade_movel	*aux2 - Unidade envolvida no combate
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int combate_defensivo (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, unidade_movel * *aux2*, player_data * *player*)

Combate defensivo. Função que executa a atualização de informações de combate (defesa) entre unidade móveis.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate
unidade_movel	*aux2 - Unidade envolvida no combate
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int destruicao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, unidade_estatica * *aux2*, player_data * *player*)

Destruição de construção. Função que executa a atualização de informações de combate entre uma unidade e um prédio.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate.
unidade_estatica	*aux2 - Unidade envolvida no combate.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int destruicao_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, base_principal * *aux2*, player_data * *player*)

Salvar jogo. Função que carrega o jogo anterior do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux -Unidade movel envolvida na destruição .
base_principal	*aux2 - Base envolvida na destruição .
player_data	*player - Informações de jogador.

Retorna:

Nada (função tipo void).

int destruicao_defensiva (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, unidade_estatica * *aux2*, player_data * *player*)

Destruição de construção de defesa. Função que executa a atualização de informações de combate entre uma unidade e um prédio.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
unidade_movel	*aux - Unidade envolvida no combate.
unidade_estatica	*aux2 - Unidade envolvida no combate.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int evolution (unidade_estatica * *aux*, player_data * *player*)

Evolui construções. Função que evolui determinada construção.

Parâmetros:

unidade_estatica	aux2 - Unidade envolvida na evolução.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int gera_operario (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, atributos_data *atributos*, player_data * *player*)

Gera operário. Função que gera os operários do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int gera_tropa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, atributos_data *atributos*, player_data * *player*)

Geração de tropas. Função que gera as tropas de determinado jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int move_unidade (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *unit*, int *i*, int *j*)

Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	unit - A unidade móvel selecionada.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

0 - Se o procedimento foi bem sucedido.

int player_level (player_data * *player*)

Definição de nível. Função que define nível de determinado jogador.

Parâmetros:

player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

void restaurar_acoes (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Definição de nível. Função que define nível de determinado jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:

Nada (função tipo void).

Referência do Arquivo interface.cpp

Arquivo com a aplicação das funções da interface do jogo.

```
#include "funcoes.h"
```

Funções

- GLuint **loadTexture** (const std::string &fileName)
  Carrega Texturas. Função que carrega as texturas atribuídas.
- GLuint **importText** (const std::string &text, int font_size, int red, int green, int blue)
  Carrega Texto. Função que carrega os textos atribuídos.
- int **carrega_interface** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **mouse_data** *mouse, **texto_data** texto, **player_data** *player, **atributos_data** atributos)
  Carrega Texto. Função que carrega os textos atribuídos.
- int **carrega_layout** ()
  Carrega layout. Função que carrega o layout para a criação de outros elementos do jogo.
- int **carrega_mapa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **mouse_data** mouse)
  Carrega o mapa. Função que carrega o mapa do jogo.
- int **carrega_numeros_recurso** (**texto_data** texto, **player_data** *player)
  Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.
- int **carrega_display_recursos** (**imagens_data** imagens, **texto_data** texto)
  Carrega informações de recursos. Função que carrega os recursos do jogador.
- int **carrega_base** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens)
  Carrega base. Função que carrega a base do jogador, com seus atributos.
- int **carrega_uni_estatico** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens)
  Carrega unidade estática. Função que renderiza a unidade estática desejada (predios).
- int **escolhe_imagem_estatica** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, int i, int j)
  Escolhe imagem estática. Função que define a imagem de unidade estática desejada.
- int **carrega_uni_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **texto_data** texto)
  Carrega a unidade movel. Função que carrega as imagens de unidades móveis no mapa.
- int **escolhe_imagem_movel** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **imagens_data** imagens, **texto_data** texto, int i, int j, int opcao)
  Escolhe imagem móvel. Função que define a imagem de unidade móvel desejada.
- int **carrega_barras** (**imagens_data** imagens)
  Carrega barras. Função que carrega a barra lateral da interface do jogador, onde ficam as principais informações.
- int **carrega_comandante** (**imagens_data** imagens)
  Carrega comandante. Função que carrega a imagem de um comandante.
- int **carrega_caixa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **imagens_data** imagens, **texto_data** texto, **atributos_data** atributos, **player_data** *player)
  Carrega a caixa. Função que carrega a caixa do jogador.
- int **carrega_botao** (**imagens_data** imagens, **texto_data** texto, **mouse_data** *mouse, int local, int **tipo**, **cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **atributos_data** atributos, **player_data** *player)
  Carrega botão. Função que carrega botões que interagem com o jogador.

- int **colore_espacos_validos** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, **player_data** *player)
  Colore espaços. Função que colore espaços no mapa, se estão vazio.
- int **colore_espacos_validos_defesa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, **player_data** *player)
  Colore espaços de defesa. Função que colore espaços no mapa, se são espaços válidos de defesa.

Descrição detalhada

Arquivo com a aplicação das funções da interface do jogo.

Autor:
Grupo 2

Funções

int carrega_barras (imagens_data *imagens*)

Carrega barras. Função que carrega a barra lateral da interface do jogador, onde ficam as principais informações.

Parâmetros:

imagens_data	imagens - Imagens carregadas.

Retorna:
0 - Se o procedimento foi bem sucedido.

int carrega_base (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*)

Carrega base. Função que carrega a base do jogador, com seus atributos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.

Retorna:
0 - Se o procedimento foi bem sucedido.

int carrega_botao (imagens_data *imagens*, texto_data *texto*, mouse_data * *mouse*, int *local*, int *tipo*, cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], atributos_data *atributos*, player_data * *player*)

Carrega botão. Função que carrega botões que interagem com o jogador.

Parâmetros:

imagens_data	imagens - Imagens carregadas.
mouse_data	mouse - Informações de localização do mouse do usuário.

int	local - local onde se amostra o botão.
int	tipo - tipo de botão.
cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
atributos_data	atributos - Atributos do jogador.
player_data	*player - Informações do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_caixa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, imagens_data *imagens*, texto_data *texto*, atributos_data *atributos*, player_data * *player*)

Carrega a caixa. Função que carrega a caixa do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_comandante (imagens_data *imagens*)

Carrega comandante. Função que carrega a imagem de um comandante.

Parâmetros:

imagens_data	imagens - Imagens carregadas.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_display_recursos (imagens_data *imagens*, texto_data *texto*)

Carrega informações de recursos. Função que carrega os recursos do jogador.

Parâmetros:

imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_interface (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, mouse_data * *mouse*, texto_data *texto*, player_data * *player*, atributos_data *atributos*)

Carrega Texto. Função que carrega os textos atribuídos.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
mouse_data	mouse - Informações de localização do mouse do usuário.
texto_data	texto - Texto a ser carregado.
player_data	*player - Informações de jogador.
atributos_data	atributos - Atributos do jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_layout ()

Carrega layout. Função que carrega o layout para a criação de outros elementos do jogo.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_mapa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, mouse_data *mouse*)

Carrega o mapa. Função que carrega o mapa do jogo.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
mouse_data	mouse - Informações de localização do mouse do usuário.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_numeros_recurso (texto_data *texto*, player_data * *player*)

Carrega quantidade de recurso. Função que carrega a quantidade de recursos do jogador.

Parâmetros:

texto_data	texto - Texto a ser carregado.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_uni_estatico (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*)

Carrega unidade estática. Função que renderiza a unidade estática desejada (predios).

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.

Retorna:

0 - Se o procedimento foi bem sucedido.

int carrega_uni_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, texto_data *texto*)

Carrega a unidade movel. Função que carrega as imagens de unidades móveis no mapa.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.

Retorna:

0 - Se o procedimento foi bem sucedido.

int colore_espacos_validos (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, player_data * *player*)

Colore espaços. Função que colore espaços no mapa, se estão vazio.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	aux - A unidade móvel selecionada.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int colore_espacos_validos_defesa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, player_data * *player*)

Colore espaços de defesa. Função que colore espaços no mapa, se são espaços válidos de defesa.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_estatica**	aux - A unidade estatica selecionada.
player_data	*player - Informações de jogador.

Retorna:

0 - Se o procedimento foi bem sucedido.

int escolhe_imagem_estatica (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, int *i*, int *j*)

Escolhe imagem estática. Função que define a imagem de unidade estática desejada.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

0 - Se o procedimento foi bem sucedido.

int escolhe_imagem_movel (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], imagens_data *imagens*, texto_data *texto*, int *i*, int *j*, int *opcao*)

Escolhe imagem móvel. Função que define a imagem de unidade móvel desejada.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
imagens_data	imagens - Imagens carregadas.
texto_data	texto - Texto a ser carregado.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.
int	opcao - Auxiliar que define caso de execução

Retorna:

0 - Se o procedimento foi bem sucedido.

GLuint importText (const std::string & *text*, int *font_size*, int *red*, int *green*, int *blue*)

Carrega Texto. Função que carrega os textos atribuídos.

Parâmetros:

const	std::string &text - Texto a ser renderizado.
int	font_size - Tamanho do texto.
int	red - Valor de vermelho.
int	green - Valor de verde.
int	blue - Valor de azul.

Retorna:

Texto- O texto renderizado.

GLuint loadTexture (const std::string & *fileName*)

Carrega Texturas. Função que carrega as texturas atribuídas.

Parâmetros:

const	std::string&fileName - Nome de arquivo.

Retorna:

Imagem- O objeto renderizado.

Referência do Arquivo main.cpp

Arquivo principal da execução do jogo.
`#include "funcoes.h"`

Funções

- int **main** ()

Descrição detalhada

Arquivo principal da execução do jogo.

Autor:
Grupo 2

Funções

int main ()

Referência do Arquivo salva_carrega.cpp

Arquivo com a opcao de salvar e carregar o estado do jogo.
`#include "funcoes.h"`

Funções

- void **salva_jogo** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Salvar jogo. Função que carrega o jogo anterior do jogador.
- void **carrega_jogo** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**])
  Carregar jogo. Função que carrega o jogo anterior do jogador.

Descrição detalhada

Arquivo com a opcao de salvar e carregar o estado do jogo.

Autor:
Grupo 2

Funções

void carrega_jogo (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Carregar jogo. Função que carrega o jogo anterior do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:
Nada (função tipo void).

void salva_jogo (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA])

Salvar jogo. Função que carrega o jogo anterior do jogador.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.

Retorna:
Nada (função tipo void).

Referência do Arquivo testa_main.cpp

Arquivo com testes das funções do jogo.

```
#include <gtest/gtest.h>
#include "funcoes.h"
```

Funções

- **TEST** (Testa, gtest_instalado)
- **TEST** (Testa, **cria_mapa**)
- **TEST** (Testa, foto_existe)
- **TEST** (Testa, foto_n_existe)
- **TEST** (Testa, **carrega_numeros_recurso**)
- **TEST** (Testa, comandante_falha)
- **TEST** (Testa, espaco_verifica)
- **TEST** (Testa, selecao_valida)
- **TEST** (Testa, **cria_base**)
- **TEST** (Testa, minimo)
- **TEST** (Testa, maximo)
- **TEST** (Testa, cria_base_espaco_ocupado)
- **TEST** (Testa, cria_uni_movel_recurso_MAX)
- **TEST** (Testa, cria_uni_movel_espaco_ocupado)
- **TEST** (Testa, cria_uni_movel_sem_recurso)
- **TEST** (Testa, cria_uni_estatico_recurso_MAX)
- **TEST** (Testa, cria_uni_estatica_espaco_ocupado)
- **TEST** (Testa, cria_uni_estatica_sem_recurso)
- **TEST** (Testa, cria_uni_movel_2)
- **TEST** (Testa, cria_uni_movel_3)
- **TEST** (Testa, **move_unidade**)
- **TEST** (Testa, **combate**)
- **TEST** (Testa, **combate_defensivo**)
- **TEST** (Testa, **destruicao**)
- **TEST** (Testa, **destruicao_defensiva**)
- **TEST** (Testa, **destruicao_base**)
- **TEST** (Testa, **player_level**)
- **TEST** (Testa, **evolution**)
- **TEST** (Testa, **gera_operario**)
- **TEST** (Testa, **gera_tropa**)
- **TEST** (Testa, CPU_module)
- int **main** (int argc, char *argv[])

Descrição detalhada

Arquivo com testes das funções do jogo.

Autor:

Grupo 2

Funções

int main (int *argc*, char * *argv*[])

TEST (Testa , gtest_instalado)

TEST (Testa , cria_mapa)

TEST (Testa , foto_existe)

TEST (Testa , foto_n_existe)

TEST (Testa , carrega_numeros_recurso)

TEST (Testa , comandante_falha)

TEST (Testa , espaco_verifica)

TEST (Testa , selecao_valida)

TEST (Testa , cria_base)

TEST (Testa , minimo)

TEST (Testa , maximo)

TEST (Testa , cria_base_espaco_ocupado)

TEST (Testa , cria_uni_movel_recurso_MAX)

TEST (Testa , cria_uni_movel_espaco_ocupado)

TEST (Testa , cria_uni_movel_sem_recurso)

TEST (Testa , cria_uni_estatico_recurso_MAX)

TEST (Testa , cria_uni_estatica_espaco_ocupado)

TEST (Testa , cria_uni_estatica_sem_recurso)

TEST (Testa , cria_uni_movel_2)

TEST (Testa , cria_uni_movel_3)

TEST (Testa , move_unidade)

TEST (Testa , combate)

TEST (Testa , combate_defensivo)

TEST (Testa , destruicao)

TEST (Testa , destruicao_defensiva)

TEST (Testa , destruicao_base)

TEST (Testa , player_level)

TEST (Testa , evolution)

TEST (Testa , gera_operario)

TEST (Testa , gera_tropa)

TEST (Testa , CPU_module)

Referência do Arquivo verificacao.cpp

Arquivo com as funções de verificação de atos no jogo.

```
#include <stdlib.h>
#include "funcoes.h"
```

Funções

- int **min** (int a, int b)
- int **max** (int a, int b)
- bool **verifica_imagem** (const std::string &fileName)
  Verifica a imagem. Função que verifica se uma imagem foi carregada corretamente.
- bool **verifica_espaco** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], int i, int j)
  Verifica espaço do mapa. Função que verifica se um um espaço no mapa está livre.
- int **verifica_selecao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** mouse)
  Verifica conteúdo espaço do mapa. Função que verifica se um espaço no mapa está ocupado por estrutura determinada.
- int **verifica_unidades** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **mouse_data** *mouse, **player_data** *player, **atributos_data** atributos, **imagens_data** imagens, **texto_data** texto)
  Verifica unidades. Função que verifica um ações entre unidades, como locomoção, combate e destruição.
- bool **verifica_velocidade** (**unidade_movel** *aux, int cell_i, int cell_j)
  Verifica velocidade. Função que verifica determinada velocidade.
- bool **verifica_alcance** (**unidade_movel** *aux, int cell_i, int cell_j)
  Verifica alcance. Função que verifica determinado alcance.
- bool **verifica_alcance_defesa** (**unidade_estatica** *aux, int cell_i, int cell_j)
  Verifica alcance da defesa. Função que verifica determinado alcance de situação de defesa.
- bool **verifica_oposicao** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_movel** *aux, int i, int j)
  Verifica oposição. Função que verifica oposição de uma ação.
- bool **verifica_oposicao_defesa** (**cell_mapa** mapa[**BLOCOS_LINHA**][**BLOCOS_LINHA**], **unidade_estatica** *aux, int i, int j)
  Verifica oposição da defesa. Função que verifica oposição à defesa..

Descrição detalhada

Arquivo com as funções de verificação de atos no jogo.

Autor:
Grupo 2

Funções

int max (int *a*, int *b*)

int min (int *a*, int *b*)

bool verifica_alcance (unidade_movel * *aux*, int *cell_i*, int *cell_j*)

Verifica alcance. Função que verifica determinado alcance.

Parâmetros:

*unidade_movel**	aux - A unidade móvel selecionada.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

bool verifica_alcance_defesa (unidade_estatica * *aux*, int *cell_i*, int *cell_j*)

Verifica alcance da defesa. Função que verifica determinado alcance de situação de defesa.

Parâmetros:

*unidade_movel**	aux - A unidade móvel selecionada.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

bool verifica_espaco (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], int *i*, int *j*)

Verifica espaço do mapa. Função que verifica se um um espaço no mapa está livre.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

true - Se o procedimento foi bem sucedido.

bool verifica_imagem (const std::string & *fileName*)

Verifica a imagem. Função que verifica se uma imagem foi carregada corretamente.

Parâmetros:

const	std::string&fileName - Arquivo que se deseja verificar.

Retorna:

true - Se o procedimento foi bem sucedido.

bool verifica_oposicao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_movel * *aux*, int *i*, int *j*)

Verifica oposição. Função que verifica oposição de uma ação.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	aux - A unidade móvel selecionada.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

bool verifica_oposicao_defesa (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], unidade_estatica * *aux*, int *i*, int *j*)

Verifica oposição da defesa. Função que verifica oposição à defesa..

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
*unidade_movel**	aux - A unidade móvel selecionada.
int	i - Auxiliar de coordenada da posição.
int	j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.

int verifica_selecao (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data *mouse*)

Verifica conteúdo espaço do mapa. Função que verifica se um espaço no mapa está ocupado por estrutura determinada.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.

Retorna:

0 - Se o espaço não está ocupado por nenhuma estrutura definida.

int verifica_unidades (cell_mapa *mapa*[BLOCOS_LINHA][BLOCOS_LINHA], mouse_data * *mouse*, player_data * *player*, atributos_data *atributos*, imagens_data *imagens*, texto_data *texto*)

Verifica unidades. Função que verifica um ações entre unidades, como locomoção, combate e destruição.

Parâmetros:

cell_mapa	mapa[BLOCOS_LINHA][BLOCOS_LINHA] - Mapa quadriculado do jogo.
mouse_data	mouse - Informações de localização do mouse do usuário.
player_data	*player - Informações de jogador.

atributos_data	atributos - Atributos do jogador.

Retorna:

0 - Se tudo ocorre dentro do esperado.

bool verifica_velocidade (unidade_movel * *aux*, int *cell_i*, int *cell_j*)

Verifica velocidade. Função que verifica determinada velocidade.

Parâmetros:

*unidade_movel**	aux - A unidade móvel selecionada.
int	cell_i - Auxiliar de coordenada da posição.
int	cell_j - Auxiliar de coordenada da posição.

Retorna:

True ou false - Se tudo ocorre dentro do esperado ou não.