Star_Trek_Federation_in_Peril

Generated by Doxygen 1.9.5

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 colordata Struct Reference

ColorData.

`#include <torpedo.h>`

Data Fields

- int r
- int g
- int b
- int a

3.1.1 Detailed Description

ColorData.

3.1.2 Field Documentation

3.1.2.1 a

`int a`

3.1.2.2 b

`int b`

3.1.2.3 g

```
int g
```

3.1.2.4 r

```
int r
```

The documentation for this struct was generated from the following file:

- torpedo.h

3.2 DebugmallocData Struct Reference

```
#include <debugmalloc.h>
```

Data Fields

- char logfile [256]
- long max_block_size
- long alloc_count
- long long alloc_bytes
- long all_alloc_count
- long long all_alloc_bytes
- DebugmallocEntry head [debugmalloc_tablesize]
- DebugmallocEntry tail [debugmalloc_tablesize]

3.2.1 Field Documentation

3.2.1.1 all_alloc_bytes

```
long long all_alloc_bytes
```

3.2.1.2 all_alloc_count

```
long all_alloc_count
```

3.2.1.3 alloc_bytes

```
long long alloc_bytes
```

3.2.1.4 alloc_count

```
long alloc_count
```

3.2.1.5 head

```
DebugmallocEntry head[debugmalloc_tablesize]
```

3.2.1.6 logfile

```
char logfile[256]
```

3.2.1.7 max_block_size

```
long max_block_size
```

3.2.1.8 tail

```
DebugmallocEntry tail[debugmalloc_tablesize]
```

The documentation for this struct was generated from the following file:

- debugmalloc.h

3.3 DebugmallocEntry Struct Reference

```
#include <debugmalloc.h>
```

Data Fields

- void * real_mem
- void * user_mem
- size_t size
- char file [64]
- unsigned line
- char func [32]
- char expr [128]
- struct DebugmallocEntry * prev
- struct DebugmallocEntry * next

3.3.1 Field Documentation

3.3.1.1 expr

```
char expr[128]
```

3.3.1.2 file

```
char file[64]
```

3.3.1.3 func

```
char func[32]
```

3.3.1.4 line

```
unsigned line
```

3.3.1.5 next

```
struct DebugmallocEntry * next
```

3.3.1.6 prev

```
struct DebugmallocEntry* prev
```

3.3.1.7 real_mem

```
void* real_mem
```

3.3.1.8 size

```
size_t size
```

3.3.1.9 user_mem

```
void* user_mem
```

The documentation for this struct was generated from the following file:

- debugmalloc.h

3.4 DinStr Struct Reference

```
#include <string_operations.h>
```

Data Fields

- int size
- char * str

3.4.1 Field Documentation

3.4.1.1 size

```
int size
```

3.4.1.2 str

```
char* str
```

The documentation for this struct was generated from the following file:

- string_operations.h

3.5 enemyship Struct Reference

EnemyShip.

```
#include <enemy_ship.h>
```

Data Fields

- int y_coor
  Az ellenseges hajo y koordinataja.
- int x_coor
  Az ellenseges hajo x koordinataja.
- int movement_dir
  Az ellenseges hajo mozgasi iranya.
- int hitbox_beg_coor
  A hitbox bal szele.
- int hitbox_end_coor
  A hitbox jobb szele.
- int centerline_y_coor
  A hitbox also hatara (ahonnan mar erzekeli a sprite a talalatot)
- TextureData texture_data
  Az ellenseges hajo textura adatai (meretek)
- SpriteMapData sprite_map_data
  Az ellenseges hajo spritemap meretei (SDL2 igenyli)
- int speed
  Az ellenseges hajo mozgasi sebessege.
- int health
  Az ellenseges hajo HP-ja.
- int score_value
  Az ellenseges hajo lelovesevel megszerezheto pont.
- struct enemyship * next_ship
  Az ellenseges hajokat tartalmazo lancolt lista kovetkezo eleme.
- struct enemyship * prev_ship
  Az ellenseges hajokat tartalmazo lancolt lista elozo eleme.

3.5.1 Detailed Description

EnemyShip.

Az ellenseges hajokat tarolo lancolt lista listaeleme

3.5.2 Field Documentation

3.5.2.1 centerline_y_coor

```
int centerline_y_coor
```

A hitbox also hatara (ahonnan mar erzekeli a sprite a talalatot)

3.5.2.2 health

```
int health
```

Az ellenseges hajo HP-ja.

3.5.2.3 hitbox_beg_coor

```
int hitbox_beg_coor
```

A hitbox bal szele.

3.5.2.4 hitbox_end_coor

```
int hitbox_end_coor
```

A hitbox jobb szele.

3.5.2.5 movement_dir

```
int movement_dir
```

Az ellenseges hajo mozgasi iranya.

3.5.2.6 next_ship

```
struct enemyship* next_ship
```

Az ellenseges hajokat tartalmazo lancolt lista kovetkezo eleme.

3.5.2.7 prev_ship

```
struct enemyship* prev_ship
```

Az ellenseges hajokat tartalmazo lancolt lista elozo eleme.

3.5.2.8 score_value

```
int score_value
```

Az ellenseges hajo lelovesevel megszerezheto pont.

3.5.2.9 speed

```
int speed
```

Az ellenseges hajo mozgasi sebessege.

3.5.2.10 sprite_map_data

```
SpriteMapData sprite_map_data
```

Az ellenseges hajo spritemap meretei (SDL2 igenyli)

3.5.2.11 texture_data

```
TextureData texture_data
```

Az ellenseges hajo textura adatai (meretek)

3.5.2.12 x_coor

```
int x_coor
```

Az ellenseges hajo x koordinataja.

3.5.2.13 y_coor

```
int y_coor
```

Az ellenseges hajo y koordinataja.

The documentation for this struct was generated from the following file:

- enemy_ship.h

3.6 gameassets Struct Reference

GameAssets.

```
#include <game_assets.h>
```

Data Fields

- StarMap * star_map
- PlayerShip * player_ship
- EnemyShip * enemy_armada
- TorpedoShot * player_torpedoes
- TorpedoShot * enemy_torpedoes

3.6.1 Detailed Description

GameAssets.

3.6.2 Field Documentation

3.6.2.1 enemy_armada

```
EnemyShip* enemy_armada
```

3.6.2.2 enemy_torpedoes

```
TorpedoShot* enemy_torpedoes
```

3.6.2.3 player_ship

```
PlayerShip* player_ship
```

3.6.2.4 player_torpedoes

```
TorpedoShot* player_torpedoes
```

3.6.2.5 star_map

```
StarMap* star_map
```

The documentation for this struct was generated from the following file:

- game_assets.h

3.7 gameattributes Struct Reference

GameAttributes.

```
#include <game_attributes.h>
```

Data Fields

- int width
- int height
- int enemy_armada_size
- int num_of_rows
- int enemy_ships_per_row
- int game_score
- InputStateInterface isi
- SDL_TimerID id

3.7.1 Detailed Description

GameAttributes.

3.7.2 Field Documentation

3.7.2.1 enemy_armada_size

```
int enemy_armada_size
```

3.7.2.2 enemy_ships_per_row

```
int enemy_ships_per_row
```

3.7.2.3 game_score

```
int game_score
```

3.7.2.4 height

```
int height
```

3.7.2.5 id

```
SDL_TimerID id
```

3.7.2.6 isi

```
InputStateInterface isi
```

3.7.2.7 num_of_rows

```
int num_of_rows
```

3.7.2.8 width

```
int width
```

The documentation for this struct was generated from the following file:

- game_attributes.h

3.8 inputstateinterface Struct Reference

InputStateInterface.

```
#include <input_state_interface.h>
```

Data Fields

- bool quit
- bool restart
- bool game_over
- bool up
- bool down
- bool left
- bool right
- bool y
- bool n
- bool torpedo
- bool torpedo_ready

3.8.1 Detailed Description

InputStateInterface.

3.8.2 Field Documentation

3.8.2.1 down

```
bool down
```

3.8.2.2 game_over

```
bool game_over
```

3.8.2.3 left

```
bool left
```

3.8.2.4 n

```
bool n
```

3.8.2.5 quit

```
bool quit
```

3.8.2.6 restart

```
bool restart
```

3.8.2.7 right

```
bool right
```

3.8.2.8 torpedo

```
bool torpedo
```

3.8.2.9 torpedo_ready

```
bool torpedo_ready
```

3.8.2.10 up

```
bool up
```

3.8.2.11 y

```
bool y
```

The documentation for this struct was generated from the following file:

- input_state_interface.h

3.9 keymap Struct Reference

KeyMap.

```
#include <keymap.h>
```

Data Fields

- char * upkey
- char * downkey
- char * leftkey
- char * rightkey
- char * torpedokey

3.9.1 Detailed Description

KeyMap.

3.9.2 Field Documentation

3.9.2.1 downkey

```
char* downkey
```

3.9.2.2 leftkey

```
char* leftkey
```

3.9.2.3 rightkey

```
char* rightkey
```

3.9.2.4 torpedokey

```
char* torpedokey
```

3.9.2.5 upkey

```
char* upkey
```

The documentation for this struct was generated from the following file:

- keymap.h

3.10 playership Struct Reference

PlayerShip.

```
#include <player_ship.h>
```

Data Fields

- int y_coor
  A hajo y koordinataja.
- int x_coor
  A hajo x koordinataja.
- int hitbox_beg_coor
  A hitbox bal szele.
- int hitbox_end_coor
  A hitbox jobb szele.
- int centerline_y_coor
  A hitbox also hatara (ahonnan mar erzekeli a sprite a talalatot)
- TextureData texture_data
  A hajo textura adatai (meretek)
- SpriteMapData sprite_map_data
  A hajo spritemap meretei (SDL2 igenyli)
- int speed
  A hajo mozgasi sebessege.
- int health
  A hajo HP-ja.
- int score_value
  A hajo elpusztulasanal elveszitett pont.

3.10.1 Detailed Description

PlayerShip.

A jatekos hajojat tarolo adatstruktura.

3.10.2 Field Documentation

3.10.2.1 centerline_y_coor

```
int centerline_y_coor
```

A hitbox also hatara (ahonnan mar erzekeli a sprite a talalatot)

3.10.2.2 health

```
int health
```

A hajo HP-ja.

3.10.2.3 hitbox_beg_coor

```
int hitbox_beg_coor
```

A hitbox bal szele.

3.10.2.4 hitbox_end_coor

```
int hitbox_end_coor
```

A hitbox jobb szele.

3.10.2.5 score_value

```
int score_value
```

A hajo elpusztulasanal elveszitett pont.

3.10.2.6 speed

```
int speed
```

A hajo mozgasi sebessege.

3.10.2.7 sprite_map_data

```
SpriteMapData sprite_map_data
```

A hajo spritemap meretei (SDL2 igenyli)

3.10.2.8 texture_data

```
TextureData texture_data
```

A hajo textura adatai (meretek)

3.10.2.9 x_coor

```
int x_coor
```

A hajo x koordinataja.

3.10.2.10 y_coor

```
int y_coor
```

A hajo y koordinataja.

The documentation for this struct was generated from the following file:

- player_ship.h

3.11 shipdtt Struct Reference

ShipDTT.

```
#include <data_transfer_types.h>
```

Data Fields

- int speed

  A hajo sebessege.
- int health

  A hajo HP-ja.
- int score_value

  A hajo pont-erteke.

3.11.1 Detailed Description

ShipDTT.

3.11.2 Field Documentation

3.11.2.1 health

```
int health
```

A hajo HP-ja.

3.11.2.2 score_value

```
int score_value
```

A hajo pont-erteke.

3.11.2.3 speed

```
int speed
```

A hajo sebessege.

The documentation for this struct was generated from the following file:

- data_transfer_types.h

3.12 spritemapdata Struct Reference

SpriteMapData.

`#include <texture_data.h>`

Data Fields

- int x_coor
- int y_coor
- int width
- int height

3.12.1 Detailed Description

SpriteMapData.

3.12.2 Field Documentation

3.12.2.1 height

```
int height
```

3.12.2.2 width

```
int width
```

3.12.2.3 x_coor

```
int x_coor
```

3.12.2.4 y_coor

```
int y_coor
```

The documentation for this struct was generated from the following file:

- texture_data.h

3.13 star Struct Reference

Star.

`#include <star_map.h>`

Data Fields

- int y_coor

  A csillagot jelkepzo kor y koordinataja.
- int x_coor

  A csillagot jelkepzo kor x koordinataja.
- int radius

  A csillagot jelkepzo kor sugara.

3.13.1 Detailed Description

Star.

Ez az adatstruktura tarolja a hatter egy csillagat. Ertekei a csillag kirajzolasahoz szukseges koordinatak es sugar. Ez az adattarolo a fuggvenyhivaskor megadando parameterlistak lerovidíteset, illetve az osszetartozo adatok egy helyen tartasat szolgalja.

3.13.2 Field Documentation

3.13.2.1 radius

`int radius`

A csillagot jelkepzo kor sugara.

3.13.2.2 x_coor

`int x_coor`

A csillagot jelkepzo kor x koordinataja.

3.13.2.3 y_coor

```
int y_coor
```

A csillagot jelkepzo kor y koordinataja.

The documentation for this struct was generated from the following file:

- star_map.h

3.14 starcolor Struct Reference

StarColor.

```
#include <star_map.h>
```

Data Fields

- int r
  A csillag RGBA piros erteke.
- int g
  A csillag RGBA zold erteke.
- int b
  A csillag RGBA kek erteke.
- int a
  A csillag RGBA alfa erteke (ez hatarozza meg a csillag attetszoseget.

3.14.1 Detailed Description

StarColor.

Ez az adatstruktura tarolja a hatter csillaganak szinet. Ertekei a csillag RGBA-ban meghatarozott szinertekei. Ez az adatterolo a fuggvenyhivaskor megadando parameterlistak leroviditeset szolgalja.

3.14.2 Field Documentation

3.14.2.1 a

```
int a
```

A csillag RGBA alfa erteke (ez hatarozza meg a csillag attetszoseget.

3.14.2.2 b

```
int b
```

A csillag RGBA kek erteke.

3.14.2.3 g

```
int g
```

A csillag RGBA zold erteke.

3.14.2.4 r

```
int r
```

A csillag RGBA piros erteke.

The documentation for this struct was generated from the following file:

- star_map.h

3.15 starmap Struct Reference

StarMap.

```
#include <star_map.h>
```

Data Fields

- int length
  A lista hossza.
- Star * stars
  A csillagokat tarolo lista.
- StarColor color
  A csillagok szinet tarolo struktura.

3.15.1 Detailed Description

StarMap.

Ez az adatstruktura tarolja a hatter osszes csillagat. Ertekei a lista hossza, a csillagokat tartalmazo lista, illetve azok szine. Ez az adattarolo a hatter csillagainak konnyu letrehozasat, tarolasat es felszabaditasat szolgalja.

3.15.2 Field Documentation

3.15.2.1 color

```
StarColor color
```

A csillagok szinet tarolo struktura.

3.15.2.2 length

```
int length
```

A lista hossza.

3.15.2.3 stars

```
Star* stars
```

A csillagokat tarolo lista.

The documentation for this struct was generated from the following file:

- star_map.h

3.16 texturedata Struct Reference

TextureData.

```
#include <texture_data.h>
```

Data Fields

- int width
- int height
- int texture_center_x
- int texture_center_y

3.16.1 Detailed Description

TextureData.

3.16.2 Field Documentation

3.16.2.1 height

```
int height
```

3.16.2.2 texture_center_x

```
int texture_center_x
```

3.16.2.3 texture_center_y

```
int texture_center_y
```

3.16.2.4 width

```
int width
```

The documentation for this struct was generated from the following file:

- texture_data.h

3.17 torpedocolors Struct Reference

TorpedoColors.

```
#include <torpedo.h>
```

Data Fields

- ColorData outter_ring
- ColorData inner_ring
- ColorData center

3.17.1 Detailed Description

TorpedoColors.

3.17.2 Field Documentation

3.17.2.1 center

```
ColorData center
```

3.17.2.2 inner_ring

```
ColorData inner_ring
```

3.17.2.3 outter_ring

```
ColorData outter_ring
```

The documentation for this struct was generated from the following file:

- torpedo.h

3.18 torpedoshot Struct Reference

TorpedoShot.

```
#include <torpedo.h>
```

Data Fields

- int x_coor
- int y_coor
- int damage
- int speed
- int dir
- TorpedoColors colors
- struct torpedoshot * next_torpedo
- struct torpedoshot * prev_torpedo

3.18.1 Detailed Description

TorpedoShot.

3.18.2 Field Documentation

3.18.2.1 colors

```
TorpedoColors colors
```

3.18.2.2 damage

```
int damage
```

3.18.2.3 dir

```
int dir
```

3.18.2.4 next_torpedo

```
struct torpedoshot* next_torpedo
```

3.18.2.5 prev_torpedo

```
struct torpedoshot* prev_torpedo
```

3.18.2.6 speed

```
int speed
```

3.18.2.7 x_coor

```
int x_coor
```

3.18.2.8 y_coor

```
int y_coor
```

The documentation for this struct was generated from the following file:

- torpedo.h

Chapter 4

File Documentation

4.1 data_transfer_types.h File Reference

Data Structures

- struct shipdtt
  ShipDTT.

Typedefs

- typedef struct shipdtt ShipDTT
  ShipDTT.

4.1.1 Typedef Documentation

4.1.1.1 ShipDTT

```
typedef struct shipdtt ShipDTT
```

ShipDTT.

4.2 data_transfer_types.h

Go to the documentation of this file.

```
1
5 #ifndef DATA_TRANSFER_TYPES_H_INCLUDED
6 #define DATA_TRANSFER_TYPES_H_INCLUDED
7
8
13 typedef struct shipdtt{
14     int speed;
15     int health;
16     int score_value;
17 }ShipDTT;
18
19 #endif // DATA_TRANSFER_TYPES_H_INCLUDED
```

4.3 debugmalloc.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdarg.h>
#include <unistd.h>
```

Data Structures

- struct DebugmallocEntry
- struct DebugmallocData

Macros

- #define malloc(S) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
- #define calloc(N, S) debugmalloc_malloc_full((N)∗(S), "calloc", #N ", " #S, __FILE__, __LINE__, true)
- #define realloc(P, S) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
- #define free(P) debugmalloc_free_full((P), "free", __FILE__, __LINE__)

Typedefs

- typedef struct DebugmallocEntry DebugmallocEntry
- typedef struct DebugmallocData DebugmallocData

Enumerations

- enum { debugmalloc_canary_size = 64 , debugmalloc_canary_char = 'K' , debugmalloc_tablesize = 256 , debugmalloc_max_block_size_default = 1048576 }

4.3.1 Macro Definition Documentation

4.3.1.1 calloc

```
#define calloc(
             N,
             S ) debugmalloc_malloc_full((N)*(S), "calloc", #N ", " #S, __FILE__, __LINE__↩
, true)
```

4.3.1.2 free

```
#define free(
             P ) debugmalloc_free_full((P), "free", __FILE__, __LINE__)
```

4.3.1.3 malloc

```
#define malloc(
             S ) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
```

4.3.1.4 realloc

```
#define realloc(
             P,
             S ) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
```

4.3.2 Typedef Documentation

4.3.2.1 DebugmallocData

```
typedef struct DebugmallocData DebugmallocData
```

4.3.2.2 DebugmallocEntry

```
typedef struct DebugmallocEntry DebugmallocEntry
```

4.3.3 Enumeration Type Documentation

4.3.3.1 anonymous enum

```
anonymous enum
```

Enumerator

debugmalloc_canary_size	
debugmalloc_canary_char	
debugmalloc_tablesize	
debugmalloc_max_block_size_default	

4.4 debugmalloc.h

Go to the documentation of this file.

```
1 #ifndef DEBUGMALLOC_H
2 #define DEBUGMALLOC_H
3
4 #include <stdbool.h>
5 #include <stddef.h>
6 #include <stdlib.h>
7 #include <stdio.h>
8 #include <ctype.h>
9 #include <string.h>
10 #include <stdarg.h>
11
12
13 enum {
14     /* size of canary in bytes. should be multiple of largest alignment
15      * required by any data type (usually 8 or 16) */
16     debugmalloc_canary_size = 64,
17
18     /* canary byte */
19     debugmalloc_canary_char = 'K',
20
21     /* hash table size for allocated entries */
22     debugmalloc_tablesize = 256,
23
24     /* max block size for allocation, can be modified with debugmalloc_max_block_size() */
25     debugmalloc_max_block_size_default = 1048576
26 };
27
28
29 /* make getpid and putenv "crossplatform". deprecated on windows but they work just fine,
30  * however not declared. */
31 #ifdef _WIN32
32     /* windows */
33     #include <process.h>
34     #ifdef _MSC_VER
35         /* visual studio, getenv/getpid deprecated warning */
36         #pragma warning(disable: 4996)
37     #else
38         /* other windows. the declaration is unfortunately hidden
39          * in mingw header files by ifdefs. */
40         int putenv(const char *);
41     #endif
42 #else
43     /* posix */
44     #include <unistd.h>
45 #endif
46
47
48 /* linked list entry for allocated blocks */
49 typedef struct DebugmallocEntry {
50     void *real_mem;     /* the address of the real allocation */
51     void *user_mem;     /* address shown to the user */
52     size_t size;        /* size of block requested by user */
53
54     char file[64];      /* malloc called in this file */
55     unsigned line;      /* malloc called at this line in file */
56     char func[32];      /* allocation function called (malloc, calloc, realloc) */
57     char expr[128];     /* expression calculating the size of allocation */
58
59     struct DebugmallocEntry *prev, *next;  /* for doubly linked list */
60 } DebugmallocEntry;
61
62
63 /* debugmalloc singleton, storing all state */
64 typedef struct DebugmallocData {
65     char logfile[256];    /* log file name or empty string */
66     long max_block_size;  /* max size of a single block allocated */
```

```
67      long alloc_count;     /* currently allocated; decreased with free */
68      long long alloc_bytes;
69      long all_alloc_count; /* all allocations, never decreased */
70      long long all_alloc_bytes;
71      DebugmallocEntry head[debugmalloc_tablesize], tail[debugmalloc_tablesize];  /* head and tail elements
        of allocation lists */
72  } DebugmallocData;
73
74
75  /* this forward declaration is required by the singleton manager function */
76  static DebugmallocData * debugmalloc_create(void);
77
78
79  /* creates singleton instance. as this function is static included to different
80   * translation units, multiple instances of the static variables are created.
81   * to make sure it is really a singleton, these instances must know each other
82   * somethow. an environment variable is used for that purpose, ie. the address
83   * of the singleton allocated is stored by the operating system.
84   * this implementation is not thread-safe. */
85  static DebugmallocData * debugmalloc_singleton(void) {
86      static char envstr[100];
87      static void *instance = NULL;
88
89      /* if we do not know the address of the singleton:
90       * - maybe we are the one to create it (env variable also does not exist)
91       * - or it is already created, and stored in the env variable. */
92      if (instance == NULL) {
93          char envvarname[100] = "";
94          sprintf(envvarname, "%s%d", "debugmallocsingleton", (int) getpid());
95          char *envptr = getenv(envvarname);
96          if (envptr == NULL) {
97              /* no env variable: create singleton. */
98              instance = debugmalloc_create();
99              sprintf(envstr, "%s=%p", envvarname, instance);
100              putenv(envstr);
101          } else {
102              /* another copy of this function already created it. */
103              int ok = sscanf(envptr, "%p", &instance);
104              if (ok != 1) {
105                  fprintf(stderr, "debugmalloc: nem lehet ertelmezni: %s!\n", envptr);
106                  abort();
107              }
108          }
109      }
110
111      return (DebugmallocData *) instance;
112  }
113
114
115  /* better version of strncpy, always terminates string with \0. */
116  static void debugmalloc_strlcpy(char *dest, char const *src, size_t destsize) {
117      strncpy(dest, src, destsize);
118      dest[destsize - 1] = '\0';
119  }
120
121
122  /* set the name of the log file for debugmalloc. empty filename
123   * means logging to stderr. */
124  static void debugmalloc_log_file(char const *logfilename) {
125      if (logfilename == NULL)
126          logfilename = "";
127      DebugmallocData *instance = debugmalloc_singleton();
128      debugmalloc_strlcpy(instance->logfile, logfilename, sizeof(instance->logfile));
129  }
130
131
132  /* set the maximum size of one block. useful for debugging purposes. */
133  static void debugmalloc_max_block_size(long max_block_size) {
134      DebugmallocData *instance = debugmalloc_singleton();
135      instance->max_block_size = max_block_size;
136  }
137
138
139
140  /* printf to the log file, or stderr. */
141  static void debugmalloc_log(char const *format, ...) {
142      DebugmallocData *instance = debugmalloc_singleton();
143      FILE *f = stderr;
144      if (instance->logfile[0] != '\0') {
145          f = fopen(instance->logfile, "at");
146          if (f == NULL) {
147              f = stderr;
148              fprintf(stderr, "debugmalloc: nem tudom megnyitni a %s fajlt irasra!\n", instance->logfile);
149              debugmalloc_strlcpy(instance->logfile, "", sizeof(instance->logfile));
150          }
151      }
152
```

```
153     va_list ap;
154     va_start(ap, format);
155     vfprintf(f, format, ap);
156     va_end(ap);
157 
158     if (f != stderr)
159         fclose(f);
160 }
161 
162 
163 /* initialize a memory block allocated for the user. the start and the end
164  * of the block is initialized with the canary characters. if 'zero' is
165  * true, the user memory area is zero-initialized, otherwise it is also
166  * filled with the canary character to simulate garbage in memory. */
167 static void debugmalloc_memory_init(DebugmallocEntry *elem, bool zero) {
168     unsigned char *real_mem = (unsigned char *) elem->real_mem;
169     unsigned char *user_mem = (unsigned char *) elem->user_mem;
170     unsigned char *canary1 = real_mem;
171     unsigned char *canary2 = real_mem + debugmalloc_canary_size + elem->size;
172     memset(canary1, debugmalloc_canary_char, debugmalloc_canary_size);
173     memset(canary2, debugmalloc_canary_char, debugmalloc_canary_size);
174     memset(user_mem, zero ? 0 : debugmalloc_canary_char, elem->size);
175 }
176 
177 /* check canary, return true if ok, false if corrupted. */
178 static bool debugmalloc_canary_ok(DebugmallocEntry const *elem) {
179     unsigned char *real_mem = (unsigned char *) elem->real_mem;
180     unsigned char *canary1 = real_mem;
181     unsigned char *canary2 = real_mem + debugmalloc_canary_size + elem->size;
182     for (size_t i = 0; i < debugmalloc_canary_size; ++i) {
183         if (canary1[i] != debugmalloc_canary_char)
184             return false;
185         if (canary2[i] != debugmalloc_canary_char)
186             return false;
187     }
188     return true;
189 }
190 
191 
192 /* dump memory contents to log file. */
193 static void debugmalloc_dump_memory(char const *mem, size_t size) {
194     for (unsigned y = 0; y < (size + 15) / 16; y++) {
195         char line[80];
196         int pos = 0;
197         pos += sprintf(line + pos, "      %04x  ", y * 16);
198         for (unsigned x = 0; x < 16; x++) {
199             if (y * 16 + x < size)
200                 pos += sprintf(line + pos, "%02x ", mem[y * 16 + x]);
201             else
202                 pos += sprintf(line + pos, "   ");
203         }
204         pos += sprintf(line + pos, "  ");
205         for (unsigned x = 0; x < 16; x++) {
206             if (y * 16 + x < size) {
207                 unsigned char c = mem[y * 16 + x];
208                 pos += sprintf(line + pos, "%c", isprint(c) ? c : '.');
209             }
210             else {
211                 pos += sprintf(line + pos, " ");
212             }
213         }
214         debugmalloc_log("%s\n", line);
215     }
216 }
217 
218 
219 /* dump data of allocated memory block.
220  * if the canary is corrupted, it is also written to the log. */
221 static void debugmalloc_dump_elem(DebugmallocEntry const *elem) {
222     bool canary_ok = debugmalloc_canary_ok(elem);
223 
224     debugmalloc_log("  %p, %u bajt, kanari: %s\n"
225                     "  %s:%u, %s(%s)\n",
226                     elem->user_mem, (unsigned) elem->size, canary_ok ? "ok" : "**SERULT**",
227                     elem->file, elem->line,
228                     elem->func, elem->expr);
229 
230     if (!canary_ok) {
231         debugmalloc_log("    ELOTTE kanari: \n");
232         debugmalloc_dump_memory((char const *) elem->real_mem, debugmalloc_canary_size);
233     }
234 
235     debugmalloc_dump_memory((char const *) elem->user_mem, elem->size > 64 ? 64 : elem->size);
236 
237     if (!canary_ok) {
238         debugmalloc_log("    UTANA kanari: \n");
239         debugmalloc_dump_memory((char const *) elem->real_mem + debugmalloc_canary_size + elem->size,
```

```
    debugmalloc_canary_size);
240     }
241 }
242
243
244 /* dump data of all memory blocks allocated. */
245 static void debugmalloc_dump(void) {
246     DebugmallocData *instance = debugmalloc_singleton();
247     debugmalloc_log("** DEBUGMALLOC DUMP **********************************\n");
248     int cnt = 0;
249     for (size_t i = 0; i < debugmalloc_tablesize; i++) {
250         DebugmallocEntry *head = &instance->head[i];
251         for (DebugmallocEntry *iter = head->next; iter->next != NULL; iter = iter->next) {
252             ++cnt;
253             debugmalloc_log("** %d/%d. rekord:\n", cnt, instance->alloc_count);
254             debugmalloc_dump_elem(iter);
255         }
256     }
257     debugmalloc_log("** DEBUGMALLOC DUMP VEGE ******************************\n");
258 }
259
260
261 /* called at program exit to dump data if there is a leak,
262  * ie. allocated block remained. */
263 static void debugmalloc_atexit_dump(void) {
264     DebugmallocData *instance = debugmalloc_singleton();
265
266     if (instance->alloc_count > 0) {
267         debugmalloc_log("\n"
268                         "*********************************************************\n"
269                         "* MEMORIASZIVARGAS VAN A PROGRAMBAN!!!\n"
270                         "*********************************************************\n"
271                         "\n");
272         debugmalloc_dump();
273     } else {
274         debugmalloc_log("*************************************************\n"
275                         "* Debugmalloc: nincs memoriaszivargas a programban.\n"
276                         "* Osszes foglalas: %d blokk, %d bajt.\n"
277                         "*************************************************\n",
278                         instance->all_alloc_count, instance->all_alloc_bytes);
279     }
280 }
281
282
283 /* hash function for bucket hash. */
284 static size_t debugmalloc_hash(void *address) {
285     /* the last few bits are ignored, as they are usually zero for
286      * alignment purposes. all tested architectures used 16 byte allocation. */
287     size_t cut = (size_t)address >> 4;
288     return cut % debugmalloc_tablesize;
289 }
290
291
292 /* insert element to hash table. */
293 static void debugmalloc_insert(DebugmallocEntry *entry) {
294     DebugmallocData *instance = debugmalloc_singleton();
295     size_t idx = debugmalloc_hash(entry->user_mem);
296     DebugmallocEntry *head = &instance->head[idx];
297     entry->prev = head;
298     entry->next = head->next;
299     head->next->prev = entry;
300     head->next = entry;
301     instance->alloc_count += 1;
302     instance->alloc_bytes += entry->size;
303     instance->all_alloc_count += 1;
304     instance->all_alloc_bytes += entry->size;
305 }
306
307
308 /* remove element from hash table */
309 static void debugmalloc_remove(DebugmallocEntry *entry) {
310     DebugmallocData *instance = debugmalloc_singleton();
311     entry->next->prev = entry->prev;
312     entry->prev->next = entry->next;
313     instance->alloc_count -= 1;
314     instance->alloc_bytes -= entry->size;
315 }
316
317
318 /* find element in hash table, given with the memory address that the user sees.
319  * @return the linked list entry, or null if not found. */
320 static DebugmallocEntry *debugmalloc_find(void *mem) {
321     DebugmallocData *instance = debugmalloc_singleton();
322     size_t idx = debugmalloc_hash(mem);
323     DebugmallocEntry *head = &instance->head[idx];
324     for (DebugmallocEntry *iter = head->next; iter->next != NULL; iter = iter->next)
325         if (iter->user_mem == mem)
```

```
326             return iter;
327     return NULL;
328 }
329 
330 
331 /* allocate memory. this function is called via the macro. */
332 static void *debugmalloc_malloc_full(size_t size, char const *func, char const *expr, char const *file,
    unsigned line, bool zero) {
333     /* imitate standard malloc: return null if size is zero */
334     if (size == 0)
335         return NULL;
336 
337     /* check max size */
338     DebugmallocData *instance = debugmalloc_singleton();
339     if (size > instance->max_block_size) {
340         debugmalloc_log("debugmalloc: %s @ %s:%u: a blokk merete tul nagy, %u bajt;
    debugmalloc_max_block_size() fuggvennyel novelheto.\n", func, file, line, (unsigned) size);
341         abort();
342     }
343 
344     /* allocate more memory, make room for canary */
345     void *real_mem = malloc(size + 2 * debugmalloc_canary_size);
346     if (real_mem == NULL) {
347         debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerult %u meretu memoriat foglalni!\n", func,
    file, line, (unsigned) size);
348         return NULL;
349     }
350 
351     /* allocate memory for linked list element */
352     DebugmallocEntry *newentry = (DebugmallocEntry *) malloc(sizeof(DebugmallocEntry));
353     if (newentry == NULL) {
354         free(real_mem);
355         debugmalloc_log("debugmalloc: %s @ %s:%u: le tudtam foglalni %u memoriat, de utana a sajatnak
    nem, sry\n", func, file, line, (unsigned) size);
356         abort();
357     }
358 
359     /* metadata of allocation: caller function, code line etc. */
360     debugmalloc_strlcpy(newentry->func, func, sizeof(newentry->func));
361     debugmalloc_strlcpy(newentry->expr, expr, sizeof(newentry->expr));
362     debugmalloc_strlcpy(newentry->file, file, sizeof(newentry->file));
363     newentry->line = line;
364 
365     /* address of allocated memory chunk */
366     newentry->real_mem = real_mem;
367     newentry->user_mem = (unsigned char *) real_mem + debugmalloc_canary_size;
368     newentry->size = size;
369     debugmalloc_memory_init(newentry, zero);
370 
371     /* store in list and return pointer to user area */
372     debugmalloc_insert(newentry);
373     return newentry->user_mem;
374 }
375 
376 
377 /* free memory and remove list item. before deleting, the chuck is filled with
378  * the canary byte to make sure that the user will see garbage if the memory
379  * is accessed after freeing. */
380 static void debugmalloc_free_inner(DebugmallocEntry *deleted) {
381     debugmalloc_remove(deleted);
382 
383     /* fill with garbage, then remove from linked list */
384     memset(deleted->real_mem, debugmalloc_canary_char, deleted->size + 2 * debugmalloc_canary_size);
385     free(deleted->real_mem);
386     free(deleted);
387 }
388 
389 
390 /* free memory - called via the macro.
391  * as all allocations are tracked in the list, this function can terminate the program
392  * if a block is freed twice or the free function is called with an invalid address. */
393 static void debugmalloc_free_full(void *mem, char const *func, char const *file, unsigned line) {
394     /* imitate standard free function: if ptr is null, no operation is performed */
395     if (mem == NULL)
396         return;
397 
398     /* find allocation, abort if not found */
399     DebugmallocEntry *deleted = debugmalloc_find(mem);
400     if (deleted == NULL) {
401         debugmalloc_log("debugmalloc: %s @ %s:%u: olyan teruletet probalsz felszabaditani, ami nincs
    lefoglalva!\n", func, file, line);
402         abort();
403     }
404 
405     /* check canary and then free memory */
406     if (!debugmalloc_canary_ok(deleted)) {
407         debugmalloc_log("debugmalloc: %s @ %s:%u: a %p memoriateruletet tulindexelted!\n", func, file,
```

```
    line, mem);
408         debugmalloc_dump_elem(deleted);
409     }
410     debugmalloc_free_inner(deleted);
411 }
412
413
414 /* realloc-like function. */
415 static void *debugmalloc_realloc_full(void *oldmem, size_t newsize, char const *func, char const *expr,
    char const *file, unsigned line) {
416     /* imitate standard realloc: equivalent to free if size is null. */
417     if (newsize == 0) {
418         debugmalloc_free_full(oldmem, func, file, line);
419         return NULL;
420     }
421     /* imitate standard realloc: equivalent to malloc if first param is NULL */
422     if (oldmem == NULL)
423         return debugmalloc_malloc_full(newsize, func, expr, file, line, 0);
424
425     /* find old allocation. abort if not found. */
426     DebugmallocEntry *oldentry = debugmalloc_find(oldmem);
427     if (oldentry == NULL) {
428         debugmalloc_log("debugmalloc: %s @ %s:%u: olyan teruletet probalsz atmeretezni, ami nincs
    lefoglalva!\n", func, file, line);
429         abort();
430     }
431
432     /* create new allocation, copy & free old data */
433     void *newmem = debugmalloc_malloc_full(newsize, func, expr, file, line, false);
434     if (newmem == NULL) {
435         debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerult uj memoriat foglalni az
    atmeretezeshez!\n", func, file, line);
436         /* imitate standard realloc: original block is untouched, but return NULL */
437         return NULL;
438     }
439     size_t smaller = oldentry->size < newsize ? oldentry->size : newsize;
440     memcpy(newmem, oldmem, smaller);
441     debugmalloc_free_inner(oldentry);
442
443     return newmem;
444 }
445
446
447 /* initialize debugmalloc singleton. returns the newly allocated instance */
448 static DebugmallocData * debugmalloc_create(void) {
449     /* config check */
450     if (debugmalloc_canary_size % 16 != 0) {
451         debugmalloc_log("debugmalloc: a kanari merete legyen 16-tal oszthato\n");
452         abort();
453     }
454     if (debugmalloc_canary_char == 0) {
455         debugmalloc_log("debugmalloc: a kanari legyen 0-tol kulonbozo\n");
456         abort();
457     }
458     /* avoid compiler warning if these functions are not used */
459     (void) debugmalloc_realloc_full;
460     (void) debugmalloc_log_file;
461     (void) debugmalloc_max_block_size;
462
463     /* create and initialize instance */
464     DebugmallocData *instance = (DebugmallocData *) malloc(sizeof(DebugmallocData));
465     if (instance == NULL) {
466         debugmalloc_log("debugmalloc: nem sikerult elinditani a memoriakezelest\n");
467         abort();
468     }
469     debugmalloc_strlcpy(instance->logfile, "", sizeof(instance->logfile));
470     instance->max_block_size = debugmalloc_max_block_size_default;
471     instance->alloc_count = 0;
472     instance->alloc_bytes = 0;
473     instance->all_alloc_count = 0;
474     instance->all_alloc_bytes = 0;
475     for (size_t i = 0; i < debugmalloc_tablesize; i++) {
476         instance->head[i].prev = NULL;
477         instance->head[i].next = &instance->tail[i];
478         instance->tail[i].next = NULL;
479         instance->tail[i].prev = &instance->head[i];
480     }
481
482     atexit(debugmalloc_atexit_dump);
483     return instance;
484 }
485
486
487 /* These macro-like functions forward all allocation/free
488  * calls to debugmalloc. Usage is the same, malloc(size)
489  * gives the address of a new memory block, free(ptr)
490  * deallocates etc.
```

```
491  *
492  * If you use this file, make sure that you include this
493  * in *ALL* translation units (*.c) of your source. The
494  * builtin free() function cannot deallocate a memory block
495  * that was allocated via debugmalloc, yet the name of
496  * the function is the same! */
497
498 #define malloc(S) debugmalloc_malloc_full((S), "malloc", #S, __FILE__, __LINE__, false)
499 #define calloc(N,S) debugmalloc_malloc_full((N)*(S), "calloc", #N ", " #S, __FILE__, __LINE__, true)
500 #define realloc(P,S) debugmalloc_realloc_full((P), (S), "realloc", #S, __FILE__, __LINE__)
501 #define free(P) debugmalloc_free_full((P), "free", __FILE__, __LINE__)
502
503 #endif
```

4.5 enemy_hit_management.h File Reference

```
#include "game_attributes.h"
#include "game_assets.h"
#include "torpedo.h"
#include "debugmalloc.h"
```

Functions

- void explode_enemy_ship_if_dead (EnemyShip **enemy_ship, EnemyShip **temp_ship, GameAttributes *game_attributes)

  explode_enemy_ship_if_dead

4.5.1 Function Documentation

4.5.1.1 explode_enemy_ship_if_dead()

```
void explode_enemy_ship_if_dead (
            EnemyShip ** enemy_ship,
            EnemyShip ** temp_ship,
            GameAttributes * game_attributes )
```

explode_enemy_ship_if_dead

Felszabaditja a felrobbant ellenseges hajot ha annak HP-ja 0 (vagy kevesebb), es kezeli az ahhoz tartozo pointereket.

Parameters

in,out	***enemy_ship*	Az ellenseges hajotkat tartalmazo lancolt lista aktualis elemenek pointere.
in,out	***temp_ship*	Az ellenseges hajot tartalmazo lancolt lista head pointerenek ideiglenes taroloja.
in,out	**game_attributes*	A jatek attributumainak taroloja.

Returns

void

4.6 enemy_hit_management.h

Go to the documentation of this file.

```
1 #ifndef ENEMY_HIT_MANAGEMENT_H_INCLUDED
2 #define ENEMY_HIT_MANAGEMENT_H_INCLUDED
3
4 #include "game_attributes.h"
5 #include "game_assets.h"
6 #include "torpedo.h"
7
8 #include "debugmalloc.h"
9
10 void explode_enemy_ship_if_dead(EnemyShip **enemy_ship, EnemyShip **temp_ship, GameAttributes
       *game_attributes);
11
12 #endif // ENEMY_HIT_MANAGEMENT_H_INCLUDED
```

4.7 enemy_hit_mananagement.c File Reference

```
#include "enemy_hit_management.h"
```

Functions

- void explode_enemy_ship_if_dead (EnemyShip **enemy_ship, EnemyShip **temp_ship, GameAttributes *game_attributes)

  explode_enemy_ship_if_dead

4.7.1 Function Documentation

4.7.1.1 explode_enemy_ship_if_dead()

```
void explode_enemy_ship_if_dead (
            EnemyShip ** enemy_ship,
            EnemyShip ** temp_ship,
            GameAttributes * game_attributes )
```

explode_enemy_ship_if_dead

Felszabaditja a felrobbant ellenseges hajot ha annak HP-ja 0 (vagy kevesebb), es kezeli az ahhoz tartozo pointereket.

Parameters

in,out	***enemy_ship*	Az ellenseges hajotkat tartalmazo lancolt lista aktualis elemenek pointere.
in,out	***temp_ship*	Az ellenseges hajot tartalmazo lancolt lista head pointerenek ideiglenes taroloja.
in,out	**game_attributes*	A jatek attributumainak taroloja.

Returns

void

4.8 enemy_ship.c File Reference

```
#include "enemy_ship.h"
```

Functions

- EnemyShip * init_enemy_armada (TextureData texture_data, SpriteMapData sprite_map_data, ShipDTT **ship_dtt, GameAttributes *game_attributes)

  init_enemy_armada
- void move_enemy_armada (EnemyShip *enemy_armada, GameAttributes *game_attributes)

  move_enemy_armada
- int find_max_enemy_armada_y_coor (EnemyShip *enemy_armada)

  find_max_enemy_armada_y_coor
- void pop_enemy_ship (EnemyShip **enemy_ship)

  pop_enemy_ship
- void free_enemy_armada (EnemyShip *enemy_armada)

  free_enemy_armada

4.8.1 Function Documentation

4.8.1.1 find_max_enemy_armada_y_coor()

```
int find_max_enemy_armada_y_coor (
            EnemyShip * enemy_armada )
```

find_max_enemy_armada_y_coor

Megkeresi a kepernyo also szelehez legkozelebb eso ellenseges hajo koordinatajat.

Parameters

in,out	*enemy_armada	Az ellenseges hajokat tartalmazo lancolt lista.

Returns

[out] max_y_coor A kepernyo also szelehez legkozelebb eso ellenseges hajo koordinataja.

4.8.1.2 free_enemy_armada()

```
void free_enemy_armada (
            EnemyShip * enemy_armada )
```

free_enemy_armada

Az ellenseges hadsereg altal elfoglalt memoriaterulet felszabaditasaert felel.

Parameters

[]	enemy_armada Az ellenseges hajotkat tartalmazo lancolt lista head pointere.

Returns

void

4.8.1.3 init_enemy_armada()

```
EnemyShip * init_enemy_armada (
            TextureData texture_data,
            SpriteMapData sprite_map_data,
            ShipDTT ** ship_dtt,
            GameAttributes * game_attributes )
```

init_enemy_armada

Az ellenseges hadsereget inicializalja a bemeneti parameterek alapjan.
VIGYAZAT: a hadsereg es az azt alkoto hajok felszabaditasaert a hivo felel!

Parameters

in	*texture_data*	Az ellenseges hajo textura-koordinata adatait tartalmazo tarolo.
in	*sprite_map_data*	Az ellenseges hajo spritemap koordinata adatait tartalmazo tarolo (az SDL2 igenyli).
in	***ship_dtt*	Az ellenseges hajo attributumait tartalmazo fajlbeolvasasbol szarmazo adatstruktura.
in	*game_attributes*	Az osszes jatekattributumot tartlmazo adatstruktura.

Returns

[out] enemy_armada az ellenseges hajok lancolt listajaval ter vissza.

4.8.1.4 move_enemy_armada()

```
void move_enemy_armada (
            EnemyShip * enemy_armada,
            GameAttributes * game_attributes )
```

move_enemy_armada

Az ellenseges hadsereg mozgasat vezerli.

Parameters

in,out	*enemy_armada	Az ellenseges hajokat tartalmazo lancolt lista.
in	*game_attributes*	Az osszes jatekattributumot tartlmazo adatstruktura.

Returns

void

4.8.1.5 pop_enemy_ship()

```
void pop_enemy_ship (
            EnemyShip ** enemy_ship )
```

pop_enemy_ship

Egy hajo torleset vegzi.

Parameters

in,out	*enemy_ship*	Az ellenseges hadsereg adott hajoja

Returns

void

4.9 enemy_ship.h File Reference

```
#include "game_attributes.h"
#include "data_transfer_types.h"
#include "random_number_in_interval.h"
#include "texture_data.h"
#include "torpedo.h"
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>
#include "debugmalloc.h"
```

Data Structures

- struct enemyship

  EnemyShip.

Typedefs

- typedef struct enemyship EnemyShip

  EnemyShip.

Functions

- EnemyShip * init_enemy_armada (TextureData texture_data, SpriteMapData sprite_map_data, ShipDTT **ship_dtt, GameAttributes *game_attributes)

  init_enemy_armada
- void move_enemy_armada (EnemyShip *enemy_armada, GameAttributes *game_attributes)

  move_enemy_armada
- int find_max_enemy_armada_y_coor (EnemyShip *enemy_armada)

  find_max_enemy_armada_y_coor
- void pop_enemy_ship (EnemyShip **enemy_ship)

  pop_enemy_ship
- void free_enemy_armada (EnemyShip *enemy_armada)

  free_enemy_armada

4.9.1 Typedef Documentation

4.9.1.1 EnemyShip

```
typedef struct enemyship EnemyShip
```

EnemyShip.

Az ellenseges hajokat tarolo lancolt lista listaeleme

4.9.2 Function Documentation

4.9.2.1 find_max_enemy_armada_y_coor()

```
int find_max_enemy_armada_y_coor (
            EnemyShip * enemy_armada )
```

find_max_enemy_armada_y_coor

Megkeresi a kepernyo also szelehez legkozelebb eso ellenseges hajo koordinatajat.

Parameters

in,out	*enemy_armada	Az ellenseges hajokat tartalmazo lancolt lista.

Returns

[out] max_y_coor A kepernyo also szelehez legkozelebb eso ellenseges hajo koordinataja.

4.9.2.2 free_enemy_armada()

```
void free_enemy_armada (
            EnemyShip * enemy_armada )
```

free_enemy_armada

Az ellenseges hadsereg altal elfoglalt memoriaterulet felszabaditasaert felel.

Parameters

[]	enemy_armada Az ellenseges hajotkat tartalmazo lancolt lista head pointere.

Returns

void

4.9.2.3 init_enemy_armada()

```
EnemyShip * init_enemy_armada (
            TextureData texture_data,
            SpriteMapData sprite_map_data,
            ShipDTT ** ship_dtt,
            GameAttributes * game_attributes )
```

init_enemy_armada

Az ellenseges hadsereget inicializalja a bemeneti parameterek alapjan.
VIGYAZAT: a hadsereg es az azt alkoto hajok felszabaditasaert a hivo felel!

Parameters

in	*texture_data*	Az ellenseges hajo textura-koordinata adatait tartalmazo tarolo.
in	*sprite_map_data*	Az ellenseges hajo spritemap koordinata adatait tartalmazo tarolo (az SDL2 igenyli).
in	***ship_dtt*	Az ellenseges hajo attributumait tartalmazo fajlbeolvasasbol szarmazo adatstruktura.
in	*game_attributes*	Az osszes jatekattributumot tartlmazo adatstruktura.

Returns

[out] enemy_armada az ellenseges hajok lancolt listajaval ter vissza.

4.9.2.4 move_enemy_armada()

```
void move_enemy_armada (
            EnemyShip * enemy_armada,
            GameAttributes * game_attributes )
```

move_enemy_armada

Az ellenseges hadsereg mozgasat vezerli.

Parameters

in,out	*enemy_armada	Az ellenseges hajokat tartalmazo lancolt lista.
in	*game_attributes*	Az osszes jatekattributumot tartlmazo adatstruktura.

Returns

void

4.9.2.5 pop_enemy_ship()

```
void pop_enemy_ship (
            EnemyShip ** enemy_ship )
```

pop_enemy_ship

Egy hajo torleset vegzi.

Parameters

in,out	*enemy_ship*	Az ellenseges hadsereg adott hajoja

Returns

void

4.10 enemy_ship.h

Go to the documentation of this file.

```
1
5 #ifndef ENEMY_SHIP_H_INCLUDED
6 #define ENEMY_SHIP_H_INCLUDED
7
8 #include "game_attributes.h"
9 #include "data_transfer_types.h"
10 #include "random_number_in_interval.h"
11 #include "texture_data.h"
12 #include "torpedo.h"
13
14 #include <stdlib.h>
15 #include <stdbool.h>
```

```
16 #include <math.h>
17
18 #include "debugmalloc.h"
19
24 typedef struct enemyship{
25     int y_coor;
26     int x_coor;
27     int movement_dir;
28     int hitbox_beg_coor;
29     int hitbox_end_coor;
30     int centerline_y_coor;
31     TextureData texture_data;
32     SpriteMapData sprite_map_data;
33     int speed;
34     int health;
35     int score_value;
36     struct enemyship *next_ship;
37     struct enemyship *prev_ship;
38 }EnemyShip;
39
40 EnemyShip *init_enemy_armada(TextureData texture_data,
41                              SpriteMapData sprite_map_data,
42                              ShipDTT **ship_dtt,
43                              GameAttributes *game_attributes);
44
45 void move_enemy_armada(EnemyShip *enemy_armada, GameAttributes *game_attributes);
46
47 int find_max_enemy_armada_y_coor(EnemyShip *enemy_armada);
48
49 void pop_enemy_ship(EnemyShip **enemy_ship);
50
51 void free_enemy_armada(EnemyShip *enemy_armada);
52
53 #endif // ENEMY_SHIP_H_INCLUDED
```

4.11 file_management.c File Reference

```
#include "file_management.h"
```

Functions

- ShipDTT ** import_ship_dtt (char *filename, int *num_of_rows)

  import_ship_dtt
- void read_texture_data (char *filename, TextureData *texture_data, SpriteMapData *sprite_map_data)

  read_texture_data

4.11.1 Function Documentation

4.11.1.1 import_ship_dtt()

```
ShipDTT ** import_ship_dtt (
            char * filename,
            int * num_of_rows )
```

import_ship_dtt

Beolvassa az urhajok alapveto attributumainak listait egy adott forrasfajlbol, majd azokbol egy ShipDTT tagokbol allo dinamikus tomb pointerevel ter vissza.

Parameters

in	*filename*	A megnyitando file neve.
out	*num_of_rows*	Az urhajok listajaban talalhato sorok szamaval ter vissza.

Returns

enemy_armada ShipDTT-ket tartalmazo lista pointere.

4.11.1.2 read_texture_data()

```
void read_texture_data (
            char * filename,
            TextureData * texture_data,
            SpriteMapData * sprite_map_data )
```

read_texture_data

Beolvassa az urhajok texturaihoz szukseges adatokat. majd azokbol egy ShipDTT tagokbol allo dinamikus tomb pointerevel ter vissza.

Parameters

in	*filename*	A megnyitando file neve.
out	*texture_data*	Az urhajok megjelenitesehez szukseges textura adatok.
out	*SpriteMapData*	Az urhajok texturainak beolvasasahoz szukseges meretek.

Returns

void

4.12 file_management.h File Reference

```
#include "data_transfer_types.h"
#include "texture_data.h"
#include "keymap.h"
#include <stdio.h>
#include "debugmalloc.h"
```

Functions

- ShipDTT ** import_ship_dtt (char *filename, int *num_of_rows)

  import_ship_dtt
- void read_texture_data (char *filename, TextureData *texture_data, SpriteMapData *sprite_map_data)

  read_texture_data

4.12.1 Function Documentation

4.12.1.1 import_ship_dtt()

```
ShipDTT ** import_ship_dtt (
            char * filename,
            int * num_of_rows )
```

import_ship_dtt

Beolvassa az urhajok alapveto attributumainak listait egy adott forrasfajlbol, majd azokbol egy ShipDTT tagokbol allo dinamikus tomb pointerevel ter vissza.

Parameters

in	*filename*	A megnyitando file neve.
out	*num_of_rows*	Az urhajok listajaban talalhato sorok szamaval ter vissza.

Returns

enemy_armada ShipDTT-ket tartalmazo lista pointere.

4.12.1.2 read_texture_data()

```
void read_texture_data (
            char * filename,
            TextureData * texture_data,
            SpriteMapData * sprite_map_data )
```

read_texture_data

Beolvassa az urhajok texturaihoz szukseges adatokat. majd azokbol egy ShipDTT tagokbol allo dinamikus tomb pointerevel ter vissza.

Parameters

in	*filename*	A megnyitando file neve.
out	*texture_data*	Az urhajok megjelenitesehez szukseges textura adatok.
out	*SpriteMapData*	Az urhajok texturainak beolvasasahoz szukseges meretek.

Returns

void

4.13 file_management.h

Go to the documentation of this file.

```
1 #ifndef FILE_MANAGEMENT_H_INCLUDED
2 #define FILE_MANAGEMENT_H_INCLUDED
3
4 #include "data_transfer_types.h"
5 #include "texture_data.h"
6 #include "keymap.h"
7
8 #include <stdio.h>
9
10 #include "debugmalloc.h"
11
12 ShipDTT **import_ship_dtt(char *filename, int *num_of_rows);
13
14 void read_texture_data(char *filename,
15                        TextureData *texture_data,
16                        SpriteMapData *sprite_map_data);
17
18 #endif // FILE_MANAGEMENT_H_INCLUDED
```

4.14 fire_management.h File Reference

```
#include "game_attributes.h"
#include "game_assets.h"
#include "enemy_ship.h"
#include "debugmalloc.h"
```

Functions

- void fire_player_torpedo (GameAssets *game_assets, GameAttributes *game_attributes)

  fire_player_torpedo
- void fire_enemy_torpedoes (int armada_size, GameAssets *game_assets)

  fire_enemy_torpedoes

4.14.1 Function Documentation

4.14.1.1 fire_enemy_torpedoes()

```
void fire_enemy_torpedoes (
            int armada_size,
            GameAssets * game_assets )
```

fire_enemy_torpedoes

Az ellenseges torpedok kiloveseert felel.

Parameters

in	*armada_size*	Az ellenseges hadseregmerete.
out	**game_assets*	A jatekhoz szukseges assetek taroloja.

Returns

void

4.14.1.2 fire_player_torpedo()

```
void fire_player_torpedo (
            GameAssets * game_assets,
            GameAttributes * game_attributes )
```

fire_player_torpedo

A jatekos torpedok kiloveseert felel.

Parameters

out	**game_assets*	A jatekhoz szukseges assetek taroloja.
out	**game_attributes*	A jatek attributumainak taroloja.

Returns

void

4.15 fire_management.h

Go to the documentation of this file.

```
1 #ifndef FIRE_MANAGEMENT_H_INCLUDED
2 #define FIRE_MANAGEMENT_H_INCLUDED
3
4 #include "game_attributes.h"
5 #include "game_assets.h"
6 #include "enemy_ship.h"
7
8 #include "debugmalloc.h"
9
10 void fire_player_torpedo(GameAssets *game_assets, GameAttributes *game_attributes);
11
12 void fire_enemy_torpedoes(int armada_size, GameAssets *game_assets);
13
14 #endif // FIRE_MANAGEMENT_H_INCLUDED
```

4.16 fire_managemet.c File Reference

```
#include "fire_management.h"
```

Functions

- void fire_player_torpedo (GameAssets *game_assets, GameAttributes *game_attributes)

  fire_player_torpedo
- void fire_enemy_torpedoes (int armada_size, GameAssets *game_assets)

  fire_enemy_torpedoes

4.16.1 Function Documentation

4.16.1.1 fire_enemy_torpedoes()

```
void fire_enemy_torpedoes (
            int armada_size,
            GameAssets * game_assets )
```

fire_enemy_torpedoes

Az ellenseges torpedok kiloveseert felel.

Parameters

in	*armada_size*	Az ellenseges hadseregmerete.
out	**game_assets*	A jatekhoz szukseges assetek taroloja.

Returns

void

4.16.1.2 fire_player_torpedo()

```
void fire_player_torpedo (
            GameAssets * game_assets,
            GameAttributes * game_attributes )
```

fire_player_torpedo

A jatekos torpedok kiloveseert felel.

Parameters

out	**game_assets*	A jatekhoz szukseges assetek taroloja.
out	**game_attributes*	A jatek attributumainak taroloja.

Returns

void

4.17 game_assets.h File Reference

```
#include "star_map.h"
#include "player_ship.h"
#include "enemy_ship.h"
#include "torpedo.h"
```

Data Structures

- struct gameassets

  GameAssets.

Typedefs

- typedef struct gameassets GameAssets

  GameAssets.

4.17.1 Typedef Documentation

4.17.1.1 GameAssets

```
typedef struct gameassets GameAssets
```

GameAssets.

4.18 game_assets.h

Go to the documentation of this file.

```
1 #ifndef GAME_ASSETS_H_INCLUDED
2 #define GAME_ASSETS_H_INCLUDED
3
4 #include "star_map.h"
5 #include "player_ship.h"
6 #include "enemy_ship.h"
7 #include "torpedo.h"
8
13 typedef struct gameassets{
14     StarMap *star_map;
15     PlayerShip *player_ship;
16     EnemyShip *enemy_armada;
17     TorpedoShot *player_torpedoes;
18     TorpedoShot *enemy_torpedoes;
19 }GameAssets;
20
21 #endif // GAME_ASSETS_H_INCLUDED
```

4.19 game_attributes.h File Reference

```
#include "input_state_interface.h"
#include <SDL.h>
```

Data Structures

- struct gameattributes

  GameAttributes.

Typedefs

- typedef struct gameattributes GameAttributes

  GameAttributes.

4.19.1 Typedef Documentation

4.19.1.1 GameAttributes

```
typedef struct gameattributes GameAttributes
```

GameAttributes.

4.20 game_attributes.h

Go to the documentation of this file.

```
1 #ifndef GAME_ATTRIBUTES_H_INCLUDED
2 #define GAME_ATTRIBUTES_H_INCLUDED
3
4 #include "input_state_interface.h"
5 #include <SDL.h>
6
11 typedef struct gameattributes{
12     int width;
13     int height;
14     int enemy_armada_size;
15     int num_of_rows;
16     int enemy_ships_per_row;
17     int game_score;
18     InputStateInterface isi;
19     SDL_TimerID id;
20 }GameAttributes;
21
22 #endif // GAME_ATTRIBUTES_H_INCLUDED
```

4.21 game_engine.c File Reference

```
#include "game_engine.h"
```

Functions

- void free_ship_dtt (ShipDTT **ship_dtt, int num_of_ships)

  free_ship_dtt
- void runtime ()

  runtime
- Uint32 input_timer (Uint32 ms, void *param)

  input_timer

4.21.1 Function Documentation

4.21.1.1 free_ship_dtt()

```
void free_ship_dtt (
            ShipDTT ** ship_dtt,
            int num_of_ships )
```

free_ship_dtt

Hasznalat utan felszabaditja a ShipDTT ideiglenes tarolokat.

Parameters

in	***ship_dtt*	A hajok inicializalasahoz szukseges fajlbol beolvasott adatok ideiglenes taroloja.
in	*num_of_ships*	A torlendo hajok szama.

Returns

void

4.21.1.2 input_timer()

```
Uint32 input_timer (
            Uint32 ms,
            void * param )
```

input_timer

Az inputok beolvasasanak idoziteseert felel.

Parameters

in	*ms*	
in	*param*	

Returns

Uint32

4.21.1.3 runtime()

```
void runtime ( )
```

runtime

Aggregalja az osszes jatek mukodesehez szukseges logikai fuggvenyt. Amikor a jatekos kilep a jatekbol, felszabadit mindent es kilep az SDL2bol.

Returns

void

4.22 game_engine.h File Reference

```
#include "input_state_interface.h"
#include "keymap.h"
#include "graphics.h"
#include "ui_input.h"
#include "game_assets.h"
#include "game_attributes.h"
#include "star_map.h"
#include "player_ship.h"
#include "enemy_ship.h"
#include "fire_management.h"
#include "hit_management.h"
#include "data_transfer_types.h"
#include "file_management.h"
#include "random_number_in_interval.h"
#include "texture_data.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <SDL.h>
#include "debugmalloc.h"
```

Functions

- void runtime ()

  runtime
- Uint32 input_timer (Uint32 ms, void ∗param)

  input_timer

4.22.1 Function Documentation

4.22.1.1 input_timer()

```
Uint32 input_timer (
            Uint32 ms,
            void * param )
```

input_timer

Az inputok beolvasasanak idoziteseert felel.

Parameters

in	*ms*	
in	*param*	

Returns

Uint32

4.22.1.2 runtime()

```
void runtime ( )
```

runtime

Aggregalja az osszes jatek mukodesehez szukseges logikai fuggvenyt. Amikor a jatekos kilep a jatekbol, felszabadit mindent es kilep az SDL2bol.

Returns

void

4.23 game_engine.h

Go to the documentation of this file.

```
1 #ifndef GAME_ENGINE_H_INCLUDED
2 #define GAME_ENGINE_H_INCLUDED
3
4 #include "input_state_interface.h"
5 #include "keymap.h"
6 #include "graphics.h"
7 #include "ui_input.h"
8 #include "game_assets.h"
9 #include "game_attributes.h"
10 #include "star_map.h"
11 #include "player_ship.h"
12 #include "enemy_ship.h"
```

```
13 #include "fire_management.h"
14 #include "hit_management.h"
15 #include "data_transfer_types.h"
16 #include "file_management.h"
17 #include "random_number_in_interval.h"
18 #include "texture_data.h"
19
20 #include <stdio.h>
21 #include <stdlib.h>
22 #include <stdbool.h>
23 #include <SDL.h>
24
25 #include "debugmalloc.h"
26
27 void runtime();
28
29 Uint32  input_timer(Uint32 ms, void *param);
30
31 #endif // GAME_ENGINE_H_INCLUDED
```

4.24 graphics.c File Reference

```
#include "graphics.h"
```

Functions

- SDL_Texture * load_sdl_texture (char *img_name)

  load_sdl_texture
- TTF_Font * open_font (int size)

  open_font
- void sdl_init (char const *title, int width, int height, SDL_Window **pwindow, SDL_Renderer **prenderer)

  sdl_init
- void create_font ()

  create_font
- void create_textures (char *fed, char *enemy)

  create_textures
- void create_window (int width, int height)

  create_window
- void draw_background (StarMap *sm)

  draw_background
- void draw_player_ship (PlayerShip *ps)

  draw_player_ship
- void draw_enemy_ships (EnemyShip *enemy_armada)

  draw_enemy_ships
- void draw_torpedo (TorpedoShot *torpedoes)

  draw_torpedo
- void draw_end_screen ()

  draw_end_screen
- void draw_score (int game_score)

  draw_score
- void draw_LCARS_bacground ()

  draw_LCARS_bacground
- void clear_screen ()

  clear_screen
- void render_screen ()

  render_screen
- void destroy_textures ()

  destroy_textures

4.24.1 Function Documentation

4.24.1.1 clear_screen()

```
void clear_screen ( )
```

clear_screen

Torol mindent a jatekablakbol.

Returns

void

4.24.1.2 create_font()

```
void create_font ( )
```

create_font

Beolvassa a szukseges meretu fontokat.

Returns

void

4.24.1.3 create_textures()

```
void create_textures (
            char * fed,
            char * enemy )
```

create_textures

felepiti a texturakat

Parameters

in	*fed*	A jatekos texturajat tartalmazo fajl neve.
in	*enemy*	A ellenseg texturajat tartalmazo fajl neve.

Returns

void

4.24.1.4 create_window()

```
void create_window (
            int width,
            int height )
```

create_window

Legeneralja a jatekablakot.

Parameters

in	*width*	
in	*height*	

Returns

void

4.24.1.5 destroy_textures()

```
void destroy_textures ( )
```

destroy_textures

Torli a texturakat.

Returns

void

4.24.1.6 draw_background()

```
void draw_background (
            StarMap * sm )
```

draw_background

kirajzolja a hatteret

Parameters

in	*∗sm*	A csillagterkep pointere.

Returns

void

4.24.1.7 draw_end_screen()

```
void draw_end_screen ( )
```

draw_end_screen

Kirajzolja a jatek vege kepernyot.

Returns

void

4.24.1.8 draw_enemy_ships()

```
void draw_enemy_ships (
            EnemyShip * enemy_armada )
```

draw_enemy_ships

Kirajzolja az ellenseges hajokat.

Parameters

in	*∗armada*	Az ellenseges hajok listajanak head pointere.

Returns

void

4.24.1.9 draw_LCARS_bacground()

```
void draw_LCARS_bacground ( )
```

draw_LCARS_bacground

Kirajzolja a jatekos pontjainak hatteret.

Returns

void

4.24.1.10 draw_player_ship()

```
void draw_player_ship (
            PlayerShip * ps )
```

draw_player_ship

Kirajzolja a jatekos hajot.

Parameters

in	*ps	A jatekos hajojanak pointere.

Returns

void

4.24.1.11 draw_score()

```
void draw_score (
            int game_score )
```

draw_score

Kirajzolja a jatekos pontjait kepernyot.

Parameters

in	*game_score*	a jatekos pontjai.

Returns

void

4.24.1.12 draw_torpedo()

```
void draw_torpedo (
            TorpedoShot * torpedoes )
```

draw_torpedo

Kirajzolja a kilott torpedot.

Parameters

in	**torpedoes*	A torpedok listajanak head pointere.

Returns

void

4.24.1.13 load_sdl_texture()

```
SDL_Texture * load_sdl_texture (
            char * img_name )
```

load_sdl_texture

Betolti az SDL2 altal hasznalt texturakat.

Parameters

in	*img_name*	A betoltendo kepfajl neve.

Returns

fontokat A betoltott textura.

4.24.1.14 open_font()

```
TTF_Font * open_font (
            int size )
```

open_font

Betolti az SDL2 altal hasznalt fontokat.

Parameters

in	*size*	A betoltendo font merete.

Returns

font A betoltott font.

4.24.1.15 render_screen()

```
void render_screen ( )
```

render_screen

Rendereli a jatekablakot.

Returns

void

4.24.1.16 sdl_init()

```
void sdl_init (
            char const * title,
            int width,
            int height,
            SDL_Window ** pwindow,
            SDL_Renderer ** prenderer )
```

sdl_init

Inicializalja az SDL2-t.

Parameters

in	*title*	
in	*width*	
in	*height*	
in	*pwindow*	
in	*prenderer*	

Returns

void

4.25 graphics.h File Reference

```
#include "star_map.h"
#include "player_ship.h"
#include "enemy_ship.h"
#include "input_state_interface.h"
#include "torpedo.h"
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_ttf.h>
#include <SDL2_gfxPrimitives.h>
```

```
#include <math.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "debugmalloc.h"
```

Functions

- void create_font ()
  create_font
- void create_textures (char *fed, char *enemy)
  create_textures
- void create_window (int width, int height)
  create_window
- void draw_background (StarMap *sm)
  draw_background
- void draw_player_ship (PlayerShip *ps)
  draw_player_ship
- void draw_enemy_ships (EnemyShip *enemy_armada)
  draw_enemy_ships
- void draw_torpedo (TorpedoShot *torpedoes)
  draw_torpedo
- void draw_end_screen ()
  draw_end_screen
- void draw_score (int game_score)
  draw_score
- void draw_LCARS_bacground ()
  draw_LCARS_bacground
- void clear_screen ()
  clear_screen
- void render_screen ()
  render_screen
- void destroy_textures ()
  destroy_textures

4.25.1 Function Documentation

4.25.1.1 clear_screen()

```
void clear_screen ( )
```

clear_screen

Torol mindent a jatekablakbol.

Returns

void

4.25.1.2 create_font()

```
void create_font ( )
```

create_font

Beolvassa a szukseges meretu fontokat.

Returns

void

4.25.1.3 create_textures()

```
void create_textures (
            char * fed,
            char * enemy )
```

create_textures

felepiti a texturakat

Parameters

in	*fed*	A jatekos texturajat tartalmazo fajl neve.
in	*enemy*	A ellenseg texturajat tartalmazo fajl neve.

Returns

void

4.25.1.4 create_window()

```
void create_window (
            int width,
            int height )
```

create_window

Legeneralja a jatekablakot.

Parameters

in	*width*	
in	*height*	

Returns

void

4.25.1.5 destroy_textures()

```
void destroy_textures ( )
```

destroy_textures

Torli a texturakat.

Returns

void

4.25.1.6 draw_background()

```
void draw_background (
            StarMap * sm )
```

draw_background

kirajzolja a hatteret

Parameters

in	*sm	A csillagterkep pointere.

Returns

void

4.25.1.7 draw_end_screen()

```
void draw_end_screen ( )
```

draw_end_screen

Kirajzolja a jatek vege kepernyot.

Returns

void

4.25.1.8 draw_enemy_ships()

```
void draw_enemy_ships (
            EnemyShip * enemy_armada )
```

draw_enemy_ships

Kirajzolja az ellenseges hajokat.

Parameters

in	*armada	Az ellenseges hajok listajanak head pointere.

Returns

void

4.25.1.9 draw_LCARS_bacground()

```
void draw_LCARS_bacground ( )
```

draw_LCARS_bacground

Kirajzolja a jatekos pontjainak hatteret.

Returns

void

4.25.1.10 draw_player_ship()

```
void draw_player_ship (
            PlayerShip * ps )
```

draw_player_ship

Kirajzolja a jatekos hajot.

Parameters

in	*ps	A jatekos hajojanak pointere.

Returns

void

4.25.1.11 draw_score()

```
void draw_score (
            int game_score )
```

draw_score

Kirajzolja a jatekos pontjait kepernyot.

Parameters

in	*game_score*	a jatekos pontjai.

Returns

void

4.25.1.12 draw_torpedo()

```
void draw_torpedo (
            TorpedoShot * torpedoes )
```

draw_torpedo

Kirajzolja a kilott torpedot.

Parameters

in	**torpedoes*	A torpedok listajanak head pointere.

Returns

void

4.25.1.13 render_screen()

```
void render_screen ( )
```

render_screen

Rendereli a jatekablakot.

Returns

void

4.26 graphics.h

Go to the documentation of this file.

```
1 #ifndef GRAPHICS_H_INCLUDED
2 #define GRAPHICS_H_INCLUDED
3
4 #include "star_map.h"
5 #include "player_ship.h"
6 #include "enemy_ship.h"
7 #include "input_state_interface.h"
8 #include "torpedo.h"
9
10 #include <SDL.h>
11 #include <SDL_image.h>
12 #include <SDL_ttf.h>
13 #include <SDL2_gfxPrimitives.h>
14 #include <math.h>
15 #include <stdlib.h>
16 #include <stdbool.h>
17 #include <string.h>
18
19 #include "debugmalloc.h"
20
21 void create_font();
22
23 void create_textures(char* fed, char* enemy);
24
25 void create_window(int width, int height);
26
27 void draw_background(StarMap *sm);
28
29 void draw_player_ship(PlayerShip *ps);
30
31 void draw_enemy_ships(EnemyShip *enemy_armada);
32
33 void draw_torpedo(TorpedoShot *torpedoes);
34
35 void draw_end_screen();
36
37 void draw_score(int game_score);
38
39 void draw_LCARS_bacground();
40
41 void clear_screen();
42
43 void render_screen();
44
45 void destroy_textures();
46
47 #endif // GRAPHICS_H_INCLUDED
```

4.27 hit_management.c File Reference

```
#include "hit_management.h"
```

Functions

- void remove_torpedo_if_out_of_bounds (TorpedoShot **torpedo, GameAttributes *game_attributes)

  ----------*Externally callable hit managers*----------

- void manage_enemy_hits (EnemyShip **enemy_armada, TorpedoShot **player_torpedoes, GameAssets **game_assets, GameAttributes *game_attributes)

  manage_enemy_hits

- void manage_player_hits (PlayerShip **player_ship, TorpedoShot **enemy_torpedoes, GameAssets **game_assets, GameAttributes *game_attributes)

  manage_player_hits

4.27.1 Function Documentation

4.27.1.1 manage_enemy_hits()

```
void manage_enemy_hits (
            EnemyShip ** enemy_armada,
            TorpedoShot ** player_torpedoes,
            GameAssets ** game_assets,
            GameAttributes * game_attributes )
```

manage_enemy_hits

Menedzseli a jatekos altal bevitt torpedo talalatokat.

Parameters

in,out	*enemy_armada*	Az osszes ellenseges hajot tartalmazo lancolt lista head pointere.
in,out	*player_torpedoes*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointere.

Returns

void

4.27.1.2 manage_player_hits()

```
void manage_player_hits (
            PlayerShip ** player_ship,
            TorpedoShot ** enemy_torpedoes,
            GameAssets ** game_assets,
            GameAttributes * game_attributes )
```

manage_player_hits

Menedzseli a jatekos altal bevitt torpedo talalatokat.

Parameters

in,out	*enemy_armada*	Az osszes ellenseges hajot tartalmazo lancolt lista head pointere.
in,out	*player_torpedoes*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointere.

Returns

void

4.27.1.3 remove_torpedo_if_out_of_bounds()

```
void remove_torpedo_if_out_of_bounds (
            TorpedoShot ** torpedo,
            GameAttributes * game_attributes )
```

----------—Externally callable hit managers----------—

remove_torpedo_if_out_of_bounds

Kitorli a torpedot, ha az elhagyta a jatekteret.

Parameters

in,out	*player_torpedoes*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointere.
in	*game_attributes*	A jatek attributumait tartalmazo adatszerkezet pointere.

Returns

void

4.28 hit_management.h File Reference

```
#include "game_attributes.h"
#include "game_assets.h"
#include "torpedo.h"
#include "enemy_ship.h"
#include "torpedo_hit_management.h"
#include "enemy_hit_management.h"
#include "player_hit_management.h"
#include <stdbool.h>
#include "debugmalloc.h"
```

Functions

- void remove_torpedo_if_out_of_bounds (TorpedoShot **torpedo, GameAttributes *game_attributes)

  ----------—*Externally callable hit managers*----------—
- void manage_enemy_hits (EnemyShip **enemy_armada, TorpedoShot **player_torpedo, GameAssets **game_assets, GameAttributes *game_attributes)

  manage_enemy_hits

4.28.1 Function Documentation

4.28.1.1 manage_enemy_hits()

```
void manage_enemy_hits (
            EnemyShip ** enemy_armada,
            TorpedoShot ** player_torpedoes,
            GameAssets ** game_assets,
            GameAttributes * game_attributes )
```

manage_enemy_hits

Menedzseli a jatekos altal bevitt torpedo talalatokat.

Parameters

in,out	*enemy_armada*	Az osszes ellenseges hajot tartalmazo lancolt lista head pointere.
in,out	*player_torpedoes*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointere.

Returns

void

4.28.1.2 remove_torpedo_if_out_of_bounds()

```
void remove_torpedo_if_out_of_bounds (
            TorpedoShot ** torpedo,
            GameAttributes * game_attributes )
```

----------—Externally callable hit managers----------—

remove_torpedo_if_out_of_bounds

Kitorli a torpedot, ha az elhagyta a jatekteret.

Parameters

in,out	*player_torpedoes*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointere.
in	*game_attributes*	A jatek attributumait tartalmazo adatszerkezet pointere.

Returns

void

4.29 hit_management.h

Go to the documentation of this file.

```
1 #ifndef HIT_MANAGEMENT_H_INCLUDED
2 #define HIT_MANAGEMENT_H_INCLUDED
3
```

```
4 #include "game_attributes.h"
5 #include "game_assets.h"
6 #include "torpedo.h"
7 #include "enemy_ship.h"
8 #include "torpedo_hit_management.h"
9 #include "enemy_hit_management.h"
10 #include "player_hit_management.h"
11
12 #include <stdbool.h>
13
14 #include "debugmalloc.h"
15
16 void remove_torpedo_if_out_of_bounds(TorpedoShot **torpedo, GameAttributes *game_attributes);;
17
18 void manage_enemy_hits(EnemyShip **enemy_armada, TorpedoShot **player_torpedo, GameAssets **game_assets,
       GameAttributes *game_attributes);
19
20 #endif // HIT_MANAGEMENT_H_INCLUDED
```

4.30 input_state_interface.h File Reference

`#include <stdbool.h>`

Data Structures

- struct inputstateinterface

  InputStateInterface.

Typedefs

- typedef struct inputstateinterface InputStateInterface

  InputStateInterface.

4.30.1 Typedef Documentation

4.30.1.1 InputStateInterface

`typedef struct inputstateinterface InputStateInterface`

InputStateInterface.

4.31 input_state_interface.h

Go to the documentation of this file.

```
1 #ifndef INPUT_STATE_INTERFACE_H_INCLUDED
2 #define INPUT_STATE_INTERFACE_H_INCLUDED
3
4 #include <stdbool.h>
5
10 typedef struct inputstateinterface{
11     bool quit;
12     bool restart;
13     bool game_over;
14     bool up;
15     bool down;
16     bool left;
17     bool right;
18     bool y;
19     bool n;
20     bool torpedo;
21     bool torpedo_ready;
22 } InputStateInterface;
23
24 #endif // INPUT_STATE_INTERFACE_H_INCLUDED
```

4.32 keymap.h File Reference

Data Structures

- struct keymap

  KeyMap.

Typedefs

- typedef struct keymap KeyMap

  KeyMap.

4.32.1 Typedef Documentation

4.32.1.1 KeyMap

```
typedef struct keymap KeyMap
```

KeyMap.

4.33 keymap.h

Go to the documentation of this file.

```
1 #ifndef KEYMAP_H_INCLUDED
2 #define KEYMAP_H_INCLUDED
3
8 typedef struct keymap{
9 char *upkey;
10 char *downkey;
11 char *leftkey;
12 char *rightkey;
13 char *torpedokey;
14 } KeyMap;
15
16 #endif // KEYMAP_H_INCLUDED
```

4.34 main.c File Reference

```
#include <stdlib.h>
#include "game_engine.h"
#include "debugmalloc.h"
```

Functions

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

4.34.1 Function Documentation

4.34.1.1 main()

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

4.35 player_hit_management.c File Reference

```
#include "player_hit_management.h"
```

Functions

- void explode_player_ship_if_dead (PlayerShip **player_ship, GameAssets **game_assets, GameAttributes *game_attributes)

  explode_player_ship_if_dead

4.35.1 Function Documentation

4.35.1.1 explode_player_ship_if_dead()

```
void explode_player_ship_if_dead (
            PlayerShip ** player_ship,
            GameAssets ** game_assets,
            GameAttributes * game_attributes )
```

explode_player_ship_if_dead

Felszabaditja a jatekos hajojat, ha annak HP-ja 0 (vagy kevesebb), es kezeli az ahhoz tartozo pointereket.

Parameters

in,out	***player_ship*	A jatekos hajojat tartalmazo adatstruktura pointere.
in,out	***game_assets*	A jatekhoz szukseges assetek taroloja.
in,out	**game_attributes*	A jatek attributumainak taroloja.

Returns

void

4.36 player_hit_management.h File Reference

```
#include "game_assets.h"
#include "player_ship.h"
#include "debugmalloc.h"
```

Functions

- void explode_player_ship_if_dead (PlayerShip **player_ship, GameAssets **game_assets, GameAttributes *game_attributes)

  explode_player_ship_if_dead

4.36.1 Function Documentation

4.36.1.1 explode_player_ship_if_dead()

```
void explode_player_ship_if_dead (
            PlayerShip ** player_ship,
            GameAssets ** game_assets,
            GameAttributes * game_attributes )
```

explode_player_ship_if_dead

Felszabaditja a jatekos hajojat, ha annak HP-ja 0 (vagy kevesebb), es kezeli az ahhoz tartozo pointereket.

Parameters

in,out	***player_ship*	A jatekos hajojat tartalmazo adatstruktura pointere.
in,out	***game_assets*	A jatekhoz szukseges assetek taroloja.
in,out	**game_attributes*	A jatek attributumainak taroloja.

Returns

void

4.37 player_hit_management.h

Go to the documentation of this file.

```
1 #ifndef PLAYER_HIT_MANAGEMENT_H_INCLUDED
2 #define PLAYER_HIT_MANAGEMENT_H_INCLUDED
3
4 #include "game_assets.h"
5 #include "player_ship.h"
6
7 #include "debugmalloc.h"
8
9 void explode_player_ship_if_dead(PlayerShip **player_ship,
10                                  GameAssets **game_assets,
11                                  GameAttributes *game_attributes);
12
13 #endif // PLAYER_HIT_MANAGEMENT_H_INCLUDED
```

4.38 player_ship.c File Reference

```
#include "player_ship.h"
```

Functions

- PlayerShip * init_player_ship (GameAttributes *game_attributes, ShipDTT *ship_dtt, TextureData texture_↩ data, SpriteMapData sprite_map_data)

  init_player_ship
- void move_player_ship (PlayerShip *ps, InputStateInterface *isi, int width, int height)

  move_player_ship
- void free_player_ship (PlayerShip *ps)

  move_player_ship

4.38.1 Function Documentation

4.38.1.1 free_player_ship()

```
void free_player_ship (
            PlayerShip * ps )
```

move_player_ship

Felszabaditja a jatekos hajojat.

Parameters

in	*ps	A jatekos hajojanak pointere.

Returns

void

4.38.1.2 init_player_ship()

```
PlayerShip * init_player_ship (
            GameAttributes * game_attributes,
            ShipDTT * ship_dtt,
            TextureData texture_data,
            SpriteMapData sprite_map_data )
```

init_player_ship

Inicializalja a jatekos hajojat.

Parameters

in	*∗game_attributes*	A jatek attributumainak taroloja.
in	*∗ship_dtt*	A hajo adatainak ideiglenes taroloja.
in	*texture_data*	A textura megjelenitesehez szukseges adatok.
in	*sprite_map_data*	A spritemap beolvasasahoz szukseges adatok.

Returns

∗ps A jatekos hajojanak pointere.

4.38.1.3 move_player_ship()

```
void move_player_ship (
            PlayerShip * ps,
            InputStateInterface * isi,
            int width,
            int height )
```

move_player_ship

A jatekos hajojanak mozgasaert felelos szamitasokat vegzi.

Parameters

in,out	*∗ps*	A jatekos hajojanak pointere.
in	*∗isi*	A jatek belso allapotainak taroloja.
in	*width*	Jatekablak szelessege.
in	*height*	Jatekablak magassaga.

Returns

void

4.39 player_ship.h File Reference

```
#include "game_attributes.h"
#include "input_state_interface.h"
#include "data_transfer_types.h"
#include "texture_data.h"
#include "player_ship.h"
#include <stdbool.h>
#include <stdio.h>
#include "debugmalloc.h"
```

Data Structures

- struct playership

  PlayerShip.

Typedefs

- typedef struct playership PlayerShip

  PlayerShip.

Functions

- PlayerShip * init_player_ship (GameAttributes *game_attributes, ShipDTT *ship_dtt, TextureData texture_↩ data, SpriteMapData sprite_map_data)

  init_player_ship
- void move_player_ship (PlayerShip *ps, InputStateInterface *isi, int width, int height)

  move_player_ship
- void free_player_ship (PlayerShip *ps)

  move_player_ship

4.39.1 Typedef Documentation

4.39.1.1 PlayerShip

```
typedef struct playership PlayerShip
```

PlayerShip.

A jatekos hajojat tarolo adatstruktura.

4.39.2 Function Documentation

4.39.2.1 free_player_ship()

```
void free_player_ship (
            PlayerShip * ps )
```

move_player_ship

Felszabaditja a jatekos hajojat.

Parameters

in	*ps	A jatekos hajojanak pointere.

Returns

void

4.39.2.2 init_player_ship()

```
PlayerShip * init_player_ship (
            GameAttributes * game_attributes,
            ShipDTT * ship_dtt,
            TextureData texture_data,
            SpriteMapData sprite_map_data )
```

init_player_ship

Inicializalja a jatekos hajojat.

Parameters

in	*game_attributes	A jatek attributumainak taroloja.
in	*ship_dtt	A hajo adatainak ideiglenes taroloja.
in	texture_data	A textura megjelenitesehez szukseges adatok.
in	sprite_map_data	A spritemap beolvasasahoz szukseges adatok.

Returns

*ps A jatekos hajojanak pointere.

4.39.2.3 move_player_ship()

```
void move_player_ship (
            PlayerShip * ps,
            InputStateInterface * isi,
            int width,
            int height )
```

move_player_ship

A jatekos hajojanak mozgasaert felelos szamitasokat vegzi.

Parameters

in,out	*ps	A jatekos hajojanak pointere.
~~in~~	~~*isi~~	~~A jatek belso allapotainak taroloja.~~
in	width	Jatekablak szelessege.
in	height	Jatekablak magassaga.

Returns

void

4.40 player_ship.h

Go to the documentation of this file.

```
1
5 #ifndef PLAYER_SHIP_H_INCLUDED
6 #define PLAYER_SHIP_H_INCLUDED
7
8 #include "game_attributes.h"
9 #include "input_state_interface.h"
10 #include "data_transfer_types.h"
11 #include "texture_data.h"
12 #include "player_ship.h"
13
14 #include <stdbool.h>
15 #include <stdio.h>
16
17 #include "debugmalloc.h"
18
23 typedef struct playership{
24     int y_coor;
25     int x_coor;
26     int hitbox_beg_coor;
27     int hitbox_end_coor;
28     int centerline_y_coor;
29     TextureData texture_data;
30     SpriteMapData sprite_map_data;
31     int speed;
32     int health;
33     int score_value;
34 }PlayerShip;
35
36 PlayerShip *init_player_ship(GameAttributes *game_attributes,
37                              ShipDTT *ship_dtt,
38                              TextureData texture_data,
39                              SpriteMapData sprite_map_data);
40
41 void move_player_ship(PlayerShip *ps,
42                       InputStateInterface *isi,
43                       int width,
44                       int height);
45
46 void free_player_ship(PlayerShip *ps);
47
48 #endif // PLAYER_SHIP_H_INCLUDED
```

4.41 random_number_in_interval.c File Reference

```
#include "random_number_in_interval.h"
```

Functions

- int random_number_in_range (int lower, int upper)

  random_number_in_range

4.41.1 Function Documentation

4.41.1.1 random_number_in_range()

```
int random_number_in_range (
            int lower,
            int upper )
```

random_number_in_range

Egy random szammal ter vissza egy meghatarozott intervallumon belul. Csak ez a modul hivhatja.

Parameters

in	*Lower*	az intervallum also hatara.
in	*Upper*	az intervallum felso hatara.

Returns

int A generalt random szam.

4.42 random_number_in_interval.h File Reference

```
#include "debugmalloc.h"
```

Functions

- int random_number_in_range (int lower, int upper)
  random_number_in_range

4.42.1 Function Documentation

4.42.1.1 random_number_in_range()

```
int random_number_in_range (
            int lower,
            int upper )
```

random_number_in_range

Egy random szammal ter vissza egy meghatarozott intervallumon belul. Csak ez a modul hivhatja.

Parameters

in	*Lower*	az intervallum also hatara.
in	*Upper*	az intervallum felso hatara.

Returns

int A generalt random szam.

4.43 random_number_in_interval.h

Go to the documentation of this file.

```
1 #ifndef RANDOM_NUMBER_IN_INTERVAL_H_INCLUDED
2 #define RANDOM_NUMBER_IN_INTERVAL_H_INCLUDED
3
4 #include "debugmalloc.h"
5
6 int random_number_in_range(int lower, int upper);
7
8 #endif // RANDOM_NUMBER_IN_INTERVAL_H_INCLUDED
```

4.44 star_map.c File Reference

```
#include "star_map.h"
```

Functions

- StarMap ∗ starmap_init (int width, int height)

  starmap_init
- void advance_starmap_frame (StarMap ∗sm, int width, int height)

  advance_starmap_frame
- void free_starmap (StarMap ∗sm)

  free_starmap

4.44.1 Function Documentation

4.44.1.1 advance_starmap_frame()

```
void advance_starmap_frame (
            StarMap * sm,
            int width,
            int height )
```

advance_starmap_frame

A csillagterkepet eloremozditja egy kockaval. Vegigmegy a csillagokat tartalmazo dinamikus listan, es mindnek egyel noveli az y koordinatajat, amennyiben az nem 10-el nagyobb az ablak magassaganal. Amennyiben ennel az erteknel magasabb az adott csillag y erteke, ugy az y koordinatat 0-ra, az x koordinatat pedig egy, a kepernyo szelessegeben talalhato random ertekre allitja.

Parameters

out	*sm	Egy StarMap tipusu pointer, a jatek StarMap tipusaban tarolt csillagok koordinatait tarolja.
in	*width*	A kepernyo szelessege. Erre a random szam generalasahoz van szukseg.
in	*height*	A kepernyo magassaga. Erre a csillag y koordinatajanak ellenorzesehez van szukseg.

Returns

void

4.44.1.2 free_starmap()

```
void free_starmap (
            StarMap * sm )
```

free_starmap

Ez a fuggveny a parameterkent kapott csillagterkep csillagainak listajat, majd magat a csillagterkepet szabaditja fel.

Parameters

in	*sm	A felszabaditando csillagterkep pointer.

Returns

void

4.44.1.3 starmap_init()

```
StarMap * starmap_init (
            int width,
            int height )
```

starmap_init

Ez a fuggveny inicializalja a StarMap csillagterkep listajat. letrehoz egy, a csillagok vart szamanak megfelelo hosszusagu dinamikus tombot, majd abban elhelyezi a sorban generalt csillagokat. Visszateresi erteke egy csillagterkep.

Parameters

in	*width*	A kepernyo szelessege. Erre a csillagok x koordinatajanak generalasahoz van szukseg.
in	*height*	A kepernyo magassaga. Erre a csillagok y koordinatajanak generalasahoz van szukseg.

Returns

StarMap

4.45 star_map.h File Reference

```
#include "random_number_in_interval.h"
#include "debugmalloc.h"
```

Data Structures

- struct starcolor
  StarColor.
- struct star
  Star.
- struct starmap
  StarMap.

Typedefs

- typedef struct starcolor StarColor
  StarColor.
- typedef struct star Star
  Star.
- typedef struct starmap StarMap
  StarMap.

Functions

- StarMap * starmap_init (int width, int height)
  starmap_init
- void advance_starmap_frame (StarMap *sm, int width, int height)
  advance_starmap_frame
- void free_starmap (StarMap *sm)
  free_starmap

4.45.1 Typedef Documentation

4.45.1.1 Star

```
typedef struct star Star
```

Star.

Ez az adatstruktura tarolja a hatter egy csillagat. Ertekei a csillag kirajzolasahoz szukseges koordinatak es sugar. Ez az adattarolo a fuggvenyhivaskor megadando parameterlistak lerovidíteset, illetve az osszetartozo adatok egy helyen tartasat szolgalja.

4.45.1.2 StarColor

```
typedef struct starcolor StarColor
```

StarColor.

Ez az adatstruktura tarolja a hatter csillaganak szinet. Ertekei a csillag RGBA-ban meghatarozott szinertekei. Ez az adatterolo a fuggvenyhivaskor megadando parameterlistak leroviditeset szolgalja.

4.45.1.3 StarMap

```
typedef struct starmap StarMap
```

StarMap.

Ez az adatstruktura tarolja a hatter osszes csillagat. Ertekei a lista hossza, a csillagokat tartalmazo lista, illetve azok szine. Ez az adattarolo a hatter csillagainak konnyu letrehozasat, tarolasat es felszabaditasat szolgalja.

4.45.2 Function Documentation

4.45.2.1 advance_starmap_frame()

```
void advance_starmap_frame (
            StarMap * sm,
            int width,
            int height )
```

advance_starmap_frame

A csillagterkepet eloremozditja egy kockaval. Vegigmegy a csillagokat tartalmazo dinamikus listan, es mindnek egyel noveli az y koordinatajat, amennyiben az nem 10-el nagyobb az ablak magassaganal. Amennyiben ennel az erteknel magasabb az adott csillag y erteke, ugy az y koordinatat 0-ra, az x koordinatat pedig egy, a kepernyo szelessegeben talalhato random ertekre allitja.

Parameters

out	*sm	Egy StarMap tipusu pointer, a jatek StarMap tipusaban tarolt csillagok koordinatait tarolja.
in	*width*	A kepernyo szelessege. Erre a random szam generalasahoz van szukseg.
in	*height*	A kepernyo magassaga. Erre a csillag y koordinatajanak ellenorzesehez van szukseg.

Returns

void

4.45.2.2 free_starmap()

```
void free_starmap (
            StarMap * sm )
```

free_starmap

Ez a fuggveny a parameterkent kapott csillagterkep csillagainak listajat, majd magat a csillagterkepet szabaditja fel.

Parameters

in	*sm	A felszabaditando csillagterkep pointer.

Returns

void

4.45.2.3 starmap_init()

```
StarMap * starmap_init (
            int width,
            int height )
```

starmap_init

Ez a fuggveny inicializalja a StarMap csillagterkep listajat. letrehoz egy, a csillagok vart szamanak megfelelo hosszusagu dinamikus tombot, majd abban elhelyezi a sorban generalt csillagokat. Visszateresi erteke egy csillagterkep.

Parameters

in	*width*	A kepernyo szelessege. Erre a csillagok x koordinatajanak generalasahoz van szukseg.
in	*height*	A kepernyo magassaga. Erre a csillagok y koordinatajanak generalasahoz van szukseg.

Returns

StarMap

4.46 star_map.h

Go to the documentation of this file.

```
1
5 #ifndef STAR_MAP_H_INCLUDED
6 #define STAR_MAP_H_INCLUDED
7
8 #include "random_number_in_interval.h"
9
10 #include "debugmalloc.h"
11
18 typedef struct starcolor{
```

```
19     int r;
20     int g;
21     int b;
22     int a;
23 }StarColor;
24
31 typedef struct star{
32     int y_coor;
33     int x_coor;
34     int radius;
35 }Star;
36
42 typedef struct starmap{
43     int length;
44     Star *stars;
45     StarColor color;
46 }StarMap;
47
48
49 StarMap *starmap_init(int width, int height);
50
51
52 void advance_starmap_frame(StarMap *sm, int width, int height);
53
54
55 void free_starmap(StarMap *sm);
56
57 #endif // STAR_MAP_H_INCLUDED
```

4.47 string_operations.c File Reference

```
#include "string_operations.h"
```

Functions

- bool dinstr_alloc (DinStr *str, int size)

  dinstr_alloc

4.47.1 Function Documentation

4.47.1.1 dinstr_alloc()

```
bool dinstr_alloc (
            DinStr * str,
            int size )
```

dinstr_alloc

dinamikus sztringet allokal

Parameters

[]	str
[]	size

Returns

bool

4.48 string_operations.h File Reference

```
#include <string.h>
#include "debugmalloc.h"
```

Data Structures

- struct DinStr

Typedefs

- typedef struct DinStr DinStr

4.48.1 Typedef Documentation

4.48.1.1 DinStr

```
typedef struct DinStr DinStr
```

4.49 string_operations.h

Go to the documentation of this file.

```
1 #ifndef STRING_OPERATIONS_H_INCLUDED
2 #define STRING_OPERATIONS_H_INCLUDED
3
4 #include <string.h>
5
6 #include "debugmalloc.h"
7
8 typedef struct DinStr {
9     int size;
10     char *str;
11 } DinStr;
12
13
14 #endif // STRING_OPERATIONS_H_INCLUDED
```

4.50 texture_data.h File Reference

```
#include "debugmalloc.h"
```

Data Structures

- struct spritemapdata

  SpriteMapData.
- struct texturedata

  TextureData.

Typedefs

- typedef struct spritemapdata SpriteMapData

  SpriteMapData.
- typedef struct texturedata TextureData

  TextureData.

4.50.1 Typedef Documentation

4.50.1.1 SpriteMapData

```
typedef struct spritemapdata SpriteMapData
```

SpriteMapData.

4.50.1.2 TextureData

```
typedef struct texturedata TextureData
```

TextureData.

4.51 texture_data.h

Go to the documentation of this file.

```
1 #ifndef TEXTURE_DATA_H_INCLUDED
2 #define TEXTURE_DATA_H_INCLUDED
3
4 #include "debugmalloc.h"
5
10 typedef struct spritemapdata
11 {
12     int x_coor;
13     int y_coor;
14     int width;
15     int height;
16 }SpriteMapData;
17
18
23 typedef struct texturedata
24 {
25     int width;
26     int height;
27     int texture_center_x;
28     int texture_center_y;
29 }TextureData;
30
31 #endif // TEXTURE_DATA_H_INCLUDED
```

4.52 torpedo.c File Reference

```
#include "torpedo.h"
```

Functions

- TorpedoShot * add_torpedo_shot (TorpedoShot *torpedoes, int damage, int speed, int x_coor, int y_coor, bool is_enemy_torpedo)

  add_torpedo_shot
- void move_torpedoes (TorpedoShot **torpedo)

  move_torpedoes
- void pop_torpedo_shot (TorpedoShot **torpedo)

  pop_torpedo_shot
- void free_torpedoes (TorpedoShot *torpedoes)

  free_torpedoes

4.52.1 Function Documentation

4.52.1.1 add_torpedo_shot()

```
TorpedoShot * add_torpedo_shot (
            TorpedoShot * torpedoes,
            int damage,
            int speed,
            int x_coor,
            int y_coor,
            bool is_enemy_torpedo )
```

add_torpedo_shot

Hozzaad a kilott torpedok listajahoz egy ujabb elemet.

Parameters

in	*torpedoes*	
in	*damage*	
in	*speed*	
in	*x_coor*	
in	*y_coor*	
in	*is_enemy_torpedo*	

Returns

TorpedoShot

4.52.1.2 free_torpedoes()

```
void free_torpedoes (
            TorpedoShot * torpedoes )
```

free_torpedoes

Felszabaditja a torpedok listajat.

Parameters

in	*torpedoes*	

Returns

void

4.52.1.3 move_torpedoes()

```
void move_torpedoes (
            TorpedoShot ** torpedo )
```

move_torpedoes

A torpedok mozgasahoz szukseges szamitasokat vegzi.

Parameters

in,out	***torpedo*	

Returns

void

4.52.1.4 pop_torpedo_shot()

```
void pop_torpedo_shot (
            TorpedoShot ** torpedo )
```

pop_torpedo_shot

amennyiben a torpedo eltalal valamit, vagy kimegy a jatekterbol, torli azt.

Parameters

in,out	**torpedo	

Returns

void

4.53 torpedo.h File Reference

```
#include "game_attributes.h"
#include <stdio.h>
#include <stdbool.h>
#include "debugmalloc.h"
```

Data Structures

- struct colordata

  ColorData.
- struct torpedocolors

  TorpedoColors.
- struct torpedoshot

  TorpedoShot.

Typedefs

- typedef struct colordata ColorData

  ColorData.
- typedef struct torpedocolors TorpedoColors

  TorpedoColors.
- typedef struct torpedoshot TorpedoShot

  TorpedoShot.

Functions

- TorpedoShot * add_torpedo_shot (TorpedoShot *torpedoes, int damage, int speed, int x_coor, int y_coor, bool is_enemy_torpedo)

  add_torpedo_shot
- void move_torpedoes (TorpedoShot **torpedo)

  move_torpedoes
- void pop_torpedo_shot (TorpedoShot **torpedo)

  pop_torpedo_shot
- void free_torpedoes (TorpedoShot *torpedoes)

  free_torpedoes

4.53.1 Typedef Documentation

4.53.1.1 ColorData

```
typedef struct colordata ColorData
```

ColorData.

4.53.1.2 TorpedoColors

```
typedef struct torpedocolors TorpedoColors
```

TorpedoColors.

4.53.1.3 TorpedoShot

```
typedef struct torpedoshot TorpedoShot
```

TorpedoShot.

4.53.2 Function Documentation

4.53.2.1 add_torpedo_shot()

```
TorpedoShot * add_torpedo_shot (
            TorpedoShot * torpedoes,
            int damage,
            int speed,
            int x_coor,
            int y_coor,
            bool is_enemy_torpedo )
```

add_torpedo_shot

Hozzaad a kilott torpedok listajahoz egy ujabb elemet.

Parameters

in	*torpedoes*	
in	*damage*	
in	*speed*	
in	*x_coor*	
in	*y_coor*	
in	*is_enemy_torpedo*	

Returns

TorpedoShot

4.53.2.2 free_torpedoes()

```
void free_torpedoes (
            TorpedoShot * torpedoes )
```

free_torpedoes

Felszabaditja a torpedok listajat.

Parameters

in	*torpedoes*	

Returns

void

4.53.2.3 move_torpedoes()

```
void move_torpedoes (
            TorpedoShot ** torpedo )
```

move_torpedoes

A torpedok mozgasahoz szukseges szamitasokat vegzi.

Parameters

in,out	***torpedo*	

Returns

void

4.53.2.4 pop_torpedo_shot()

```
void pop_torpedo_shot (
            TorpedoShot ** torpedo )
```

pop_torpedo_shot

amennyiben a torpedo eltalal valamit, vagy kimegy a jatekterbol, torli azt.

Parameters

in,out	**torpedo	

Returns

void

4.54 torpedo.h

Go to the documentation of this file.

```
1 #ifndef TORPEDO_H_INCLUDED
2 #define TORPEDO_H_INCLUDED
3
4 #include "game_attributes.h"
5 #include <stdio.h>
6
7 #include <stdbool.h>
8
9 #include "debugmalloc.h"
10
15 typedef struct colordata
16 {
17     int r;
18     int g;
19     int b;
20     int a;
21 }ColorData;
22
27 typedef struct torpedocolors
28 {
29     ColorData outter_ring;
30     ColorData inner_ring;
31     ColorData center;
32 }TorpedoColors;
33
38 typedef struct torpedoshot
39 {
40     int x_coor;
41     int y_coor;
42     int damage;
43     int speed;
44     int dir;
45     TorpedoColors colors;
46     struct torpedoshot *next_torpedo;
47     struct torpedoshot *prev_torpedo;
48 }TorpedoShot;
49
50 TorpedoShot *add_torpedo_shot(TorpedoShot *torpedoes, int damage, int speed, int x_coor, int y_coor, bool
       is_enemy_torpedo);
51
52 void move_torpedoes(TorpedoShot **torpedo);
53
54 void pop_torpedo_shot(TorpedoShot **torpedo);
55
56 void free_torpedoes(TorpedoShot *torpedoes);
57
58 #endif // TORPEDO_H_INCLUDED
```

4.55 torpedo_hit_management.c File Reference

```
#include "torpedo_hit_management.h"
```

Functions

- bool is_torpedo_out_of_bounds (TorpedoShot **torpedo, GameAttributes *game_attributes)

  is_torpedo_out_of_bounds
- void explode_torpedo (TorpedoShot **torpedo, TorpedoShot **temp_torpedo)

  explode_torpedo

4.55.1 Function Documentation

4.55.1.1 explode_torpedo()

```
void explode_torpedo (
            TorpedoShot ** torpedo,
            TorpedoShot ** temp_torpedo )
```

explode_torpedo

Felszabaditja a jatekos felrobbant torpedojat es kezeli az ahhoz tartozo pointereket.

Parameters

in,out	***torpedo*	A jatekos altal kilott torpedokat tartalmazo lancolt lista aktualis elemenek pointere.
in,out	***temp_torpedo*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointerenek ideiglenes taroloja.

Returns

void

4.55.1.2 is_torpedo_out_of_bounds()

```
bool is_torpedo_out_of_bounds (
            TorpedoShot ** torpedo,
            GameAttributes * game_attributes )
```

is_torpedo_out_of_bounds

Erzekeli, ha a torpedo kilepett a jatekterbol.

Parameters

in	***torpedo*	A torpedokat tartalmazo lancolt lista adott eleme.
in	**game_attributes*	A jatek attributumait tartalmazo adatszerkezet pointere.

Returns

bool igaz erteket ad, ha az adott torpedo kilepett a jatekterbol, egyebkent hamis erteket ad.

4.56 torpedo_hit_management.h File Reference

```
#include "game_attributes.h"
```

```
#include "torpedo.h"
#include <stdbool.h>
#include "debugmalloc.h"
```

Functions

- bool is_torpedo_out_of_bounds (TorpedoShot **torpedo, GameAttributes *game_attributes)

  is_torpedo_out_of_bounds
- void explode_torpedo (TorpedoShot **player_torpedo, TorpedoShot **temp_torpedo)

  explode_torpedo

4.56.1 Function Documentation

4.56.1.1 explode_torpedo()

```
void explode_torpedo (
            TorpedoShot ** torpedo,
            TorpedoShot ** temp_torpedo )
```

explode_torpedo

Felszabaditja a jatekos felrobbant torpedojat es kezeli az ahhoz tartozo pointereket.

Parameters

in,out	***torpedo*	A jatekos altal kilott torpedokat tartalmazo lancolt lista aktualis elemenek pointere.
in,out	***temp_torpedo*	A jatekos altal kilott torpedokat tartalmazo lancolt lista head pointerenek ideiglenes taroloja.

Returns

void

4.56.1.2 is_torpedo_out_of_bounds()

```
bool is_torpedo_out_of_bounds (
            TorpedoShot ** torpedo,
            GameAttributes * game_attributes )
```

is_torpedo_out_of_bounds

Erzekeli, ha a torpedo kilepett a jatekterbol.

Parameters

in	∗∗*torpedo*	A torpedokat tartalmazo lancolt lista adott eleme.
in	∗*game_attributes*	A jatek attributumait tartalmazo adatszerkezet pointere.

Returns

bool igaz erteket ad, ha az adott torpedo kilepett a jatekterbol, egyebkent hamis erteket ad.

4.57 torpedo_hit_management.h

Go to the documentation of this file.

```
1 #ifndef TORPEDO_HIT_MANAGEMENT_H_INCLUDED
2 #define TORPEDO_HIT_MANAGEMENT_H_INCLUDED
3
4 #include "game_attributes.h"
5 #include "torpedo.h"
6
7 #include <stdbool.h>
8
9 #include "debugmalloc.h"
10
11 bool is_torpedo_out_of_bounds(TorpedoShot **torpedo, GameAttributes *game_attributes);
12
13 void explode_torpedo(TorpedoShot **player_torpedo, TorpedoShot **temp_torpedo);
14
15
16 #endif // TORPEDO_HIT_MANAGEMENT_H_INCLUDED
```

4.58 ui_input.c File Reference

```
#include "ui_input.h"
```

Functions

- void user_input (InputStateInterface ∗isi, KeyMap ∗key_map, SDL_TimerID id)
  user_input

4.58.1 Function Documentation

4.58.1.1 user_input()

```
void user_input (
            InputStateInterface * isi,
            KeyMap * key_map,
            SDL_TimerID id )
```

user_input

A felhasznalotol erkezo billentyuparancsokat ertelmezi, es egy interface-n keresztel adja at a program tobbi reszenek

Parameters

in,out	*isi	a jatek InputStateInterface-re mutato pointer. Ezen keresztul kommunikalnak egymassal a vezerlomodulok.
in	*key_map	ez a vezerlo KeyMap interfacen keresztul hasonlitja ossze a bejovo billentyuparancsokat a valid vezerlo gombokkal.
in	id	egy SDL_TimerID tipusu idozito. Feladata, hogy general egy SDL_USEREVENTet, amennyiben az idozito lejartaval nincs beerkezo esemeny/parancs (enelkul a vezerlo blokkolna a program futasat, nem mukodne a hatter animacio, es semmi nem tortenne, amig nincs felhasznaloi interakcio).

Returns

void

4.59 ui_input.h File Reference

```
#include "input_state_interface.h"
#include "keymap.h"
#include "SDL_timer.h"
#include <stdbool.h>
#include <SDL.h>
#include <SDL2_gfxPrimitives.h>
#include <stdio.h>
#include "debugmalloc.h"
```

Functions

- void user_input (InputStateInterface *isi, KeyMap *key_map, SDL_TimerID id)

  user_input

4.59.1 Function Documentation

4.59.1.1 user_input()

```
void user_input (
            InputStateInterface * isi,
            KeyMap * key_map,
            SDL_TimerID id )
```

user_input

A felhasznalotol erkezo billentyuparancsokat ertelmezi, es egy interface-n keresztel adja at a program tobbi reszenek

Parameters

in,out	*isi	a jatek InputStateInterface-re mutato pointer. Ezen keresztul kommunikalnak egymassal a vezerlomodulok.
in	*key_map	ez a vezerlo KeyMap interfacen keresztul hasonlitja ossze a bejovo billentyuparancsokat a valid vezerlo gombokkal.
in	id	egy SDL_TimerID tipusu idozito. Feladata, hogy general egy SDL_USEREVENTet, amennyiben az idozito lejartaval nincs beerkezo esemeny/parancs (enelkul a vezerlo blokkolna a program futasat, nem mukodne a hatter animacio, es semmi nem tortenne, amig nincs felhasznaloi interakcio).

Returns

void

4.60 ui_input.h

Go to the documentation of this file.

```
1 #ifndef UI_INPUT_H_INCLUDED
2 #define UI_INPUT_H_INCLUDED
3
4 #include "input_state_interface.h"
5 #include "keymap.h"
6
7 #include "SDL_timer.h"
8 #include <stdbool.h>
9 #include <SDL.h>
10 #include <SDL2_gfxPrimitives.h>
11
12 #include <stdio.h>
13
14 #include "debugmalloc.h"
15
16 void user_input(InputStateInterface *isi, KeyMap *key_map, SDL_TimerID id);
17
18 #endif // UI_INPUT_H_INCLUDED
```

Index