

My Project

Generated by Doxygen 1.9.5

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 beam Struct Reference	5
3.1.1 Detailed Description	5
3.2 beamcolor Struct Reference	5
3.2.1 Detailed Description	6
3.3 colordata Struct Reference	6
3.3.1 Detailed Description	6
3.4 DebugmallocData Struct Reference	6
3.5 DebugmallocEntry Struct Reference	7
3.6 DinStr Struct Reference	7
3.7 enemyarmada Struct Reference	7
3.7.1 Detailed Description	8
3.8 enemyship Struct Reference	8
3.8.1 Detailed Description	8
3.9 enemysquadronship Struct Reference	8
3.9.1 Detailed Description	9
3.10 gameassets Struct Reference	9
3.10.1 Detailed Description	9
3.11 gameattributes Struct Reference	9
3.11.1 Detailed Description	10
3.12 inputstateinterface Struct Reference	10
3.12.1 Detailed Description	10
3.13 keymap Struct Reference	10
3.13.1 Detailed Description	11
3.14 leveledtt Struct Reference	11
3.14.1 Detailed Description	11
3.15 mouseposition Struct Reference	11
3.15.1 Detailed Description	12
3.16 phaserbeam Struct Reference	12
3.16.1 Detailed Description	12
3.17 playership Struct Reference	12
3.17.1 Detailed Description	13
3.18 shipdtt Struct Reference	13
3.18.1 Detailed Description	13
3.19 spritemapdata Struct Reference	13
3.19.1 Detailed Description	14
3.20 star Struct Reference	14

3.20.1 Detailed Description	14
3.21 starcolor Struct Reference	14
3.21.1 Detailed Description	15
3.22 starmap Struct Reference	15
3.22.1 Detailed Description	15
3.23 texturedata Struct Reference	16
3.23.1 Detailed Description	16
3.24 torpedocolors Struct Reference	16
3.24.1 Detailed Description	16
3.25 torpedoshot Struct Reference	16
3.25.1 Detailed Description	17
4 File Documentation	19
4.1 data_transfer_types.h	19
4.2 debugmalloc.h	19
4.3 enemy_ship.c File Reference	25
4.3.1 Function Documentation	26
4.3.1.1 create_enemy_ship()	26
4.3.1.2 enemy_armada_entry_animation()	26
4.3.1.3 free_enemy_armada()	27
4.3.1.4 free_enemy_squadron()	27
4.3.1.5 free_enemy_squadron_ship()	28
4.3.1.6 init_enemy_armada()	28
4.3.1.7 manage_enemy_dirs()	28
4.3.1.8 modify_enemy_dir()	29
4.3.1.9 move_enemy_armada()	29
4.3.1.10 position_enemy_armada()	30
4.3.1.11 squadron_size()	30
4.4 enemy_ship.h	30
4.5 file_management.c File Reference	31
4.5.1 Function Documentation	31
4.5.1.1 import_ship_dtt()	31
4.6 file_management.h	32
4.7 game_assets.h	32
4.8 game_attributes.h	32
4.9 game_engine.c File Reference	32
4.9.1 Function Documentation	33
4.9.1.1 calculate_game_assets()	33
4.9.1.2 clear_graphics()	34
4.9.1.3 default_keymap_init()	34
4.9.1.4 draw_graphics()	34
4.9.1.5 free_assets()	35

4.9.1.6 free_components()	35
4.9.1.7 game_loop()	36
4.9.1.8 init_game_assets()	36
4.9.1.9 init_game_attributes()	37
4.9.1.10 input_timer()	37
4.9.1.11 keep_enemy_time()	37
4.9.1.12 keep_player_time()	38
4.9.1.13 runtime()	38
4.10 game_engine.h	38
4.11 graphics.c File Reference	38
4.11.1 Function Documentation	39
4.11.1.1 clear_screen()	39
4.11.1.2 create_textures()	39
4.11.1.3 create_window()	40
4.11.1.4 destroy_textures()	40
4.11.1.5 draw_background()	40
4.11.1.6 draw_crosshair()	41
4.11.1.7 draw_enemy_ships()	41
4.11.1.8 draw Phaser()	42
4.11.1.9 draw_player_ship()	42
4.11.1.10 draw_torpedo()	42
4.11.1.11 load_sdl_texture()	43
4.11.1.12 render_screen()	43
4.12 graphics.h	43
4.13 hit_management.c File Reference	44
4.13.1 Function Documentation	44
4.13.1.1 detect_player_hit()	44
4.13.1.2 manage_hits()	45
4.14 hit_management.h	45
4.15 input_state_interface.h	45
4.16 keymap.h	46
4.17 phaser.c File Reference	46
4.17.1 Function Documentation	46
4.17.1.1 create_beam_attributes()	46
4.17.1.2 free Phaser()	47
4.17.1.3 phaser_init()	47
4.18 phaser.h	47
4.19 player_ship.c File Reference	48
4.19.1 Function Documentation	48
4.19.1.1 fire Phaser()	48
4.19.1.2 free_player_ship()	49
4.19.1.3 init_player_ship()	49

4.19.1.4 move_player_ship()	50
4.20 player_ship.h	50
4.21 random_number_in_interval.c File Reference	51
4.21.1 Function Documentation	51
4.21.1.1 random_number_in_range()	51
4.22 random_number_in_interval.h	51
4.23 star_map.c File Reference	51
4.23.1 Function Documentation	52
4.23.1.1 advance_starmap_frame()	52
4.23.1.2 free_starmap()	52
4.23.1.3 starmap_init()	53
4.24 star_map.h	53
4.25 string_operations.c File Reference	54
4.25.1 Function Documentation	54
4.25.1.1 dinstr_alloc()	54
4.26 string_operations.h	54
4.27 texture_data.h	55
4.28 torpedo.c File Reference	55
4.28.1 Function Documentation	55
4.28.1.1 add_torpedo_shot()	56
4.28.1.2 free_torpedoes()	56
4.28.1.3 init_torpedo_colors()	57
4.28.1.4 move_torpedoes()	57
4.28.1.5 remove_torpedo_shot()	57
4.29 torpedo.h	58
4.30 ui_input.c File Reference	58
4.30.1 Function Documentation	58
4.30.1.1 user_input()	59
4.31 ui_input.h	59

Index

61

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

beam	Beam	5
beamcolor	BeamColor	5
colordata	ColorData	6
DebugmallocData	DebugmallocData	6
DebugmallocEntry	DebugmallocEntry	7
DinStr	DinStr	7
enemyarmada	EnemyArmada	7
enemyship	EnemyShip	8
enemysquadronship	EnemySquadronShip	8
gameassets	GameAssets	9
gameattributes	GameAttributes	9
inputstateinterface	InputStateInterface	10
keymap	KeyMap	10
leveldtt	LevelDTT	11
mouseposition	MousePosition	11
phaserbeam	PhaserBeam	12
playership	PlayerShip	12
shipdtt	ShipDTT	13
spritemapdata	SpriteMapData	13

star		
	Star	14
starcolor		
	StarColor	14
starmap		
	StarMap	15
texturedata		
	TextureData	16
torpedocolors		
	TorpedoColors	16
torpedoshot		
	TorpedoShot	16

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

data_transfer_types.h	??
debugmalloc.h	??
enemy_ship.c	25
enemy_ship.h	??
file_management.c	31
file_management.h	??
game_assets.h	??
game_attributes.h	??
game_engine.c	32
game_engine.h	??
graphics.c	38
graphics.h	??
hit_management.c	44
hit_management.h	??
input_state_interface.h	??
keymap.h	??
phaser.c	46
phaser.h	??
player_ship.c	48
player_ship.h	??
random_number_in_interval.c	51
random_number_in_interval.h	??
star_map.c	51
star_map.h	??
string_operations.c	54
string_operations.h	??
texture_data.h	??
torpedo.c	55
torpedo.h	??
ui_input.c	58
ui_input.h	??

Chapter 3

Class Documentation

3.1 beam Struct Reference

Beam.

```
#include <phaser.h>
```

Public Attributes

- [BeamColor](#) **core_color**
- [BeamColor](#) **falloff_color**

3.1.1 Detailed Description

Beam.

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

- phaser.h

3.2 beamcolor Struct Reference

BeamColor.

```
#include <phaser.h>
```

Public Attributes

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

3.2.1 Detailed Description

BeamColor.

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

- phaser.h

3.3 colordata Struct Reference

ColorData.

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

Public Attributes

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

3.3.1 Detailed Description

ColorData.

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

- torpedo.h

3.4 DebugmallocData Struct Reference

Public Attributes

- 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]

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

- debugmalloc.h

3.5 DebugmallocEntry Struct Reference

Public Attributes

- 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**

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

- debugmalloc.h

3.6 DinStr Struct Reference

Public Attributes

- int **size**
- char * **str**

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

- string_operations.h

3.7 enemyarmada Struct Reference

EnemyArmada.

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

Public Attributes

- int **number_of_squadrons**
- int * **no_of_ships_per_sq**
- [EnemySquadronShip](#) ** **enemy_armada**
- int * **squadron_dirs**
- int * **squadron_heights**
- bool * **entry_finished_per_squadron**
- bool **ready_to_move**

3.7.1 Detailed Description

EnemyArmada.

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

- `enemy_ship.h`

3.8 enemyship Struct Reference

EnemyShip.

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

Public Attributes

- `int y_coor`
- `int x_coor`
- [TextureData](#) `texture_data`
- `int speed`
- `int health`
- `int damage`

3.8.1 Detailed Description

EnemyShip.

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

- `enemy_ship.h`

3.9 enemysquadronship Struct Reference

EnemySquadronShip.

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

Public Attributes

- [EnemyShip](#) `ship`
- `struct enemysquadronship * next_ship`
- `struct enemysquadronship * prev_ship`

3.9.1 Detailed Description

EnemySquadronShip.

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

- enemy_ship.h

3.10 gameassets Struct Reference

GameAssets.

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

Public Attributes

- [StarMap](#) * **star_map**
- [PlayerShip](#) * **player_ship**
- [EnemyArmada](#) * **enemy_armada**
- [TorpedoShot](#) * **player_torpedo**
- [TorpedoShot](#) * **quantum_torpedo**
- [TorpedoShot](#) * **enemy_torpedo**

3.10.1 Detailed Description

GameAssets.

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

- game_assets.h

3.11 gameattributes Struct Reference

GameAttributes.

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

Public Attributes

- int **width**
- int **height**
- [InputStateInterface](#) **isi**
- SDL_TimerID **id**

3.11.1 Detailed Description

GameAttributes.

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

- `game_attributes.h`

3.12 inputstateinterface Struct Reference

InputStateInterface.

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

Public Attributes

- `bool quit`
- `bool up`
- `bool down`
- `bool left`
- `bool right`
- `bool torpedo`
- `bool torpedo_ready`
- `bool left_mouse_button`
- `bool right_mouse_button`
- [MousePosition](#) `mouse_position`
- `bool phaser_ready`
- `bool phaser_firing`

3.12.1 Detailed Description

InputStateInterface.

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

- `input_state_interface.h`

3.13 keymap Struct Reference

KeyMap.

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


Public Attributes

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

3.13.1 Detailed Description

KeyMap.

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

- keymap.h

3.14 leveledtt Struct Reference

LevelDTT.

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

Public Attributes

- int **number_of_waves**
- int **number_of_squadrons**
- [ShipDTT](#) ** **shiptypes_per_squadron**
- int * **ships_per_squadron**

3.14.1 Detailed Description

LevelDTT.

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

- data_transfer_types.h

3.15 mouseposition Struct Reference

MousePosition.

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

Public Attributes

- int **mouse_x**
- int **mouse_y**

3.15.1 Detailed Description

MousePosition.

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

- input_state_interface.h

3.16 phaserbeam Struct Reference

PhaserBeam.

```
#include <phaser.h>
```

Public Attributes

- int **beg_x**
- int **beg_y**
- int **end_x**
- int **end_y**
- [Beam](#) **beam_composition**

3.16.1 Detailed Description

PhaserBeam.

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

- phaser.h

3.17 playership Struct Reference

PlayerShip.

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

Public Attributes

- int **y_coor**
- int **x_coor**
- [TextureData](#) **texture_data**
- [PhaserBeam](#) * **phaser_blast**
- int **phaser_timer**
- int **health**
- int **speed**

3.17.1 Detailed Description

PlayerShip.

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

- `player_ship.h`

3.18 shipdtt Struct Reference

ShipDTT.

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

Public Attributes

- int **speed**
- int **health**
- int **damage**

3.18.1 Detailed Description

ShipDTT.

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

- `data_transfer_types.h`

3.19 spritemapdata Struct Reference

SpriteMapData.

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

Public Attributes

- int **x_coor**
- int **y_coor**
- int **width**
- int **hight**

3.19.1 Detailed Description

SpriteMapData.

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

- texture_data.h

3.20 star Struct Reference

Star.

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

Public Attributes

- int **y_coor**
A csillagot jelkezo kor y koordinataja.
- int **x_coor**
A csillagot jelkezo kor x koordinataja.
- int **radius**
A csillagot jelkezo kor sugara.

3.20.1 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 leroviditeset, illetve az osszetartozo adatok egy helyen tartasat szolgalja.

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

- star_map.h

3.21 starcolor Struct Reference

StarColor.

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

Public Attributes

- **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 határozza meg a csillag attetszoseget).

3.21.1 Detailed Description

StarColor.

Ez az adatstruktúra tárolja a hatter csillaganak szinet. Ertekei a csillag RGBA-ban meghatarozott szinertekei. Ez az adatterolo a függvényhivaskor megadando parameterlistak leroviditeset szolgálja.

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

- star_map.h

3.22 starmap Struct Reference

StarMap.

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

Public Attributes

- **int length**
A lista hossza.
- [Star](#) * **stars**
A csillagokat tarolo lista.
- [StarColor](#) **color**
A csillagok szinet tarolo struktura.

3.22.1 Detailed Description

StarMap.

Ez az adatstruktúra tárolja a hatter osszes csillagat. Ertekei a lista hossza, a csillagokat tartalmazo lista, illetve azok szine. Ez az adattarolo a hatter csillagainak konnyu letehozasat, tarolasat es felszabaditasat szolgálja.

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

- star_map.h

3.23 texturedata Struct Reference

TextureData.

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

Public Attributes

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

3.23.1 Detailed Description

TextureData.

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

- texture_data.h

3.24 torpedocolors Struct Reference

TorpedoColors.

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

Public Attributes

- [ColorData](#) **outter_ring**
- [ColorData](#) **inner_ring**
- [ColorData](#) **center**

3.24.1 Detailed Description

TorpedoColors.

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

- torpedo.h

3.25 torpedoshot Struct Reference

TorpedoShot.

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

Public Attributes

- int **x_coor**
- int **y_coor**
- int **damage**
- int **speed**
- int **dir**
- [TorpedoColors](#) **colors**
- struct [torpedoshot](#) * **next_torpedo**
- struct [torpedoshot](#) * **prev_torpedo**

3.25.1 Detailed Description

TorpedoShot.

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

- torpedo.h

Chapter 4

File Documentation

4.1 data_transfer_types.h

```
1 #ifndef DATA_TRANSFER_TYPES_H_INCLUDED
2 #define DATA_TRANSFER_TYPES_H_INCLUDED
3
4
5
6
7
8
9 typedef struct shipdtt{
10     int speed;
11     int health;
12     int damage;
13 }ShipDTT;
14
15
16
17
18
19 typedef struct leveledtt{
20     int number_of_waves;
21     int number_of_squadrons;
22     ShipDTT **shiptypes_per_squadron;
23     int *ships_per_squadron;
24 }LevelDTT;
25
26 #endif // DATA_TRANSFER_TYPES_H_INCLUDED
```

4.2 debugmalloc.h

```
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
72     of allocation lists */
73 } DebugmallocData;
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 * somehow. 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_strncpy(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 *logfile) {
125     if (logfile == NULL)
126         logfile = "";
127     DebugmallocData *instance = debugmalloc_singleton();
128     debugmalloc_strncpy(instance->logfile, logfile, 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, "a");
146         if (f == NULL) {
147             f = stderr;
148             fprintf(stderr, "debugmalloc: nem tudom megnyitni a %s fajlt irasra!\n", instance->logfile);
149             debugmalloc_strncpy(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,
240             debugmalloc_canary_size);
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("** DEBUGMALLOD 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("** DEBUGMALLOD 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_strncpy(newentry->func, func, sizeof(newentry->func));
361     debugmalloc_strncpy(newentry->expr, expr, sizeof(newentry->expr));
362     debugmalloc_strncpy(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 chunk 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 területet próbalsz felszabadítani, ami nincs
402         lefoglalva!\n", func, file, line);
403         abort();
404     }
405
406     /* check canary and then free memory */
407     if (!debugmalloc_canary_ok(deleted)) {
408         debugmalloc_log("debugmalloc: %s @ %s:%u: a %p memoriaterületet tulindexelted!\n", func, file,
409         line, mem);
410         debugmalloc_dump_elem(deleted);
411     }
412     debugmalloc_free_inner(deleted);
413 }
414
415 /* realloc-like function. */
416 static void *debugmalloc_realloc_full(void *oldmem, size_t newsize, char const *func, char const *expr,
417 char const *file, unsigned line) {
418     /* imitate standard realloc: equivalent to free if size is null. */
419     if (newsize == 0) {
420         debugmalloc_free_full(oldmem, func, file, line);
421         return NULL;
422     }
423     /* imitate standard realloc: equivalent to malloc if first param is NULL */
424     if (oldmem == NULL)
425         return debugmalloc_malloc_full(newsize, func, expr, file, line, 0);
426
427     /* find old allocation. abort if not found. */
428     DebugmallocEntry *oldentry = debugmalloc_find(oldmem);
429     if (oldentry == NULL) {
430         debugmalloc_log("debugmalloc: %s @ %s:%u: olyan területet próbalsz atmeretezni, ami nincs
431         lefoglalva!\n", func, file, line);
432         abort();
433     }
434
435     /* create new allocation, copy & free old data */
436     void *newmem = debugmalloc_malloc_full(newsize, func, expr, file, line, false);
437     if (newmem == NULL) {
438         debugmalloc_log("debugmalloc: %s @ %s:%u: nem sikerült új memóriát foglalni az
439         atmeretezéshez!\n", func, file, line);
440         /* imitate standard realloc: original block is untouched, but return NULL */
441         return NULL;
442     }
443     size_t smaller = oldentry->size < newsize ? oldentry->size : newsize;
444     memcpy(newmem, oldmem, smaller);
445     debugmalloc_free_inner(oldentry);
446
447     return newmem;
448 }
449
450 /* initialize debugmalloc singleton. returns the newly allocated instance */
451 static DebugmallocData * debugmalloc_create(void) {
452     /* config check */
453     if (debugmalloc_canary_size % 16 != 0) {
454         debugmalloc_log("debugmalloc: a kanari merete legyen 16-tal osztható\n");
455         abort();
456     }
457     if (debugmalloc_canary_char == 0) {
458         debugmalloc_log("debugmalloc: a kanari legyen 0-tól különbozo\n");
459         abort();
460     }
461 }

```

```

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_strncpy(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

```

4.3 enemy_ship.c File Reference

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

Functions

- [EnemyShip](#) [create_enemy_ship](#) ([ShipDTT](#) *ship_dtt, int y_coord)
create_enemy_ship
- [EnemyArmada](#) * [init_enemy_armada](#) ([LevelDTT](#) *level_dtt, [TextureData](#) texture_data, [GameAttributes](#) *game_attributes)
init_enemy_armada
- int [squadron_size](#) ([EnemySquadronShip](#) *squadron)
squadron_size
- void [position_enemy_armada](#) ([EnemySquadronShip](#) *squadron, [GameAttributes](#) *game_attributes, int dir)
position_enemy_armada
- bool [enemy_armada_entry_animation](#) ([EnemySquadronShip](#) *squadron, [GameAttributes](#) *game_attributes, int dir)
enemy_armada_entry_animation
- void [modify_enemy_dir](#) ([EnemyArmada](#) *armada)

- modify_enemy_dir*
- void `manage_enemy_dirs` (`EnemyArmada` *armada, `GameAttributes` *game_attributes)
- manage_enemy_dirs*
- void `move_enemy_armada` (`EnemyArmada` *armada, `GameAttributes` *game_attributes)
- move_enemy_armada*
- void `free_enemy_squadron_ship` (`EnemySquadronShip` **ess)
- free_enemy_squadron_ship*
- void `free_enemy_squadron` (`EnemySquadronShip` *squadron)
- free_enemy_squadron*
- void `free_enemy_armada` (`EnemyArmada` *armada)
- free_enemy_armada*

4.3.1 Function Documentation

4.3.1.1 create_enemy_ship()

```
EnemyShip create_enemy_ship (
    ShipDTT * ship_dtt,
    int y_coord )
```

create_enemy_ship

az ellenseges hajok generalasaert felelos fuggveny. Hivasonkent csak egy hajot general

Parameters

in	<i>ship_dtt</i>	az ellenseges hajo attributumait tartalmazo fajlbeolvasasbol szarmazo adatstruktura
in	<i>y_coord</i>	az ellenseges hajo y koordinataja

Returns

[out] `EnemyShip` ellenseges urhajo tipusokkal ter vissza

4.3.1.2 enemy_armada_entry_animation()

```
bool enemy_armada_entry_animation (
    EnemySquadronShip * squadron,
    GameAttributes * game_attributes,
    int dir )
```

enemy_armada_entry_animation

a jatek kezdo animaciojat vezerli.

Parameters

in, out	<i>squadron</i>	egy hajo-sor egy elemere mutato lancolt lista elem pointer
in	<i>game_attributes</i>	az osszes jatekattributumot tartlmazo adatstruktura
in	<i>dir</i>	a hajo x iranyu mozgasi iranya

Returns

[out] bool true ertekkel ter vissza, ha a kezdo animacio befejezodott.

4.3.1.3 free_enemy_armada()

```
void free_enemy_armada (
    EnemyArmada * armada )
```

free_enemy_armada

az ellenseges hadsereg által elfoglalt memoriaterulet felszabaditasaert felel

Parameters

[]	armada
----	--------

Returns

void

4.3.1.4 free_enemy_squadron()

```
void free_enemy_squadron (
    EnemySquadronShip * squadron )
```

free_enemy_squadron

az ellenseges hajok által soronként elfoglalt memoriaterulet felszabaditasaert felel

Parameters

[]	squadron
----	----------

Returns

void

4.3.1.5 free_enemy_squadron_ship()

```
void free_enemy_squadron_ship (
    EnemySquadronShip ** ess )
```

free_enemy_squadron_ship

az ellenseges hajok által elfoglalt memoriaterület egyenkenti felszabadításaért felel

Parameters

[]	ess
-----	-----

Returns

void

4.3.1.6 init_enemy_armada()

```
EnemyArmada * init_enemy_armada (
    LevelDTT * level_dtt,
    TextureData texture_data,
    GameAttributes * game_attributes )
```

init_enemy_armada

az ellenseges hadsereget inicializálja a bemeneti paraméterek alapján.

VIGYAZAT: a hadsereg és az azt alkotó hajok felszabadításaért a hívó felel!

MEGJEGYZÉS: ez a függvény és a segédfüggvényei a végleges verzióhoz refaktoralason esnek majd át, a függvény ajelenlegi verzióban működik ugyan, de nem azt a feladatot látja el, amit majd a végleges verzióban kell.

Parameters

in	level_dtt	az ellenseges hadsereg attribútumait tartalmazó fájlbeolvasásból származó adatstruktúra
in	texture_data	az egyes hajok textúra adatai a kirajzoláshoz (közép koordináták, textúra szélesség)
in	game_attributes	az összes jatekattribútumot tartalmazó adatstruktúra

Returns

[out] EnemyArmada az ellenseges hajokkal tér vissza

4.3.1.7 manage_enemy_dirs()

```
void manage_enemy_dirs (
    EnemyArmada * armada,
    GameAttributes * game_attributes )
```

manage_enemy_dirs

visszafordítja az ellenseges hajokat a játék csata animációja közben, amennyiben azok elerték a mozgási szabadságuk végét jelző x koordinátát

Parameters

<code>[]</code>	<code>armada</code>
<code>[]</code>	<code>game_attributes</code>

Returns

void

4.3.1.8 modify_enemy_dir()

```
void modify_enemy_dir (
    EnemyArmada * armada )
```

modify_enemy_dir

az ellenseges hajók mozgási irányát változtatja az ellenkezőjére

Parameters

<code>in, out</code>	<code>armada</code>	az ellenseges hadsereget tartalmazó adatstruktúra
----------------------	---------------------	---

Returns

void

4.3.1.9 move_enemy_armada()

```
void move_enemy_armada (
    EnemyArmada * armada,
    GameAttributes * game_attributes )
```

move_enemy_armada

az egész ellenseges hadsereg mozgásáért felel

Parameters

<code>[]</code>	<code>armada</code>
<code>[]</code>	<code>game_attributes</code>

Returns

void

4.3.1.10 position_enemy_armada()

```
void position_enemy_armada (
    EnemySquadronShip * squadron,
    GameAttributes * game_attributes,
    int dir )
```

position_enemy_armada

az ellenseges hajokat a kepernyon kívül elhelyezi, elokeszitve azokat jatek kezdeti animaciojara

Parameters

in, out	<i>squadron</i>	egy hajo-sor egy elemere mutato lancolt lista elem pointer
in	<i>game_attributes</i>	az osszes jatekattributumot tartlmazo adatstruktura
in	<i>dir</i>	a hajo x iranyu mozgasi iranya

Returns

[out] void

4.3.1.11 squadron_size()

```
int squadron_size (
    EnemySquadronShip * squadron )
```

squadron_size

egy ellenseges hajo-sort tartalmazo lancolt lista alapjan visszater az adott sorban levo hajok szamaval

Parameters

in	<i>squadron</i>	egy adott sor ellenseges hajot tarolo lancolt lista head pointere
----	-----------------	---

Returns

[out] int visszater egy sor hajo meretevel

4.4 enemy_ship.h

```
1 #ifndef ENEMY_SHIP_H_INCLUDED
```

```

2 #define ENEMY_SHIP_H_INCLUDED
3
4 #include "game_attributes.h"
5 #include "data_transfer_types.h"
6 #include "random_number_in_interval.h"
7 #include "texture_data.h"
8
9 #include <stdlib.h>
10 #include <stdbool.h>
11
12 #include "debugmalloc.h"
13
14 typedef struct enemyship{
15     int y_coor;
16     int x_coor;
17     TextureData texture_data;
18     int speed;
19     int health;
20     int damage;
21 }EnemyShip;
22
23 typedef struct enemysquadronship{
24     EnemyShip ship;
25     struct enemysquadronship *next_ship;
26     struct enemysquadronship *prev_ship;
27 }EnemySquadronShip;
28
29 typedef struct enemyarmada{
30     int number_of_squadrons;
31     int *no_of_ships_per_sq;
32     EnemySquadronShip **enemy_armada;
33     int *squadron_dirs;
34     int *squadron_heights;
35     bool *entry_finished_per_squadron;
36     bool ready_to_move;
37 }EnemyArmada;
38
39 #endif // ENEMY_SHIP_H_INCLUDED

```

4.5 file_management.c File Reference

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

Functions

- [ShipDTT * import_ship_dtt](#) (char *filepath)
import_ship_dtt

4.5.1 Function Documentation

4.5.1.1 import_ship_dtt()

```
ShipDTT * import_ship_dtt (
    char * filepath )
```

import_ship_dtt

beimportálja az urhajok alapveto attributumainak listait egy adott forrasfajlbol, majd azokbol egy listak dinamikus tombje pointerrel ter vissza.

Parameters

<code>[]</code>	<code>filepath</code>
-----------------	-----------------------

Returns

ShipDTT

4.6 file_management.h

```
1 #ifndef FILE_MANAGEMENT_H_INCLUDED
2 #define FILE_MANAGEMENT_H_INCLUDED
3
4 #include "data_transfer_types.h"
5
6 #include <stdio.h>
7
8 #include "debugmalloc.h"
9
10 #endif // FILE_MANAGEMENT_H_INCLUDED
```

4.7 game_assets.h

```
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
9
10 typedef struct gameassets{
11     StarMap *star_map;
12     PlayerShip *player_ship;
13     EnemyArmada *enemy_armada;
14     TorpedoShot *player_torpedo;
15     TorpedoShot *quantum_torpedo;
16     TorpedoShot *enemy_torpedo;
17 }GameAssets;
18
19 #endif // GAME_ASSETS_H_INCLUDED
```

4.8 game_attributes.h

```
1 #ifndef GAME_ATTRIBUTES_H_INCLUDED
2 #define GAME_ATTRIBUTES_H_INCLUDED
3
4 #include "input_state_interface.h"
5 #include <SDL.h>
6
7 typedef struct gameattributes{
8     int width;
9     int height;
10     InputStateInterface isi;
11     SDL_TimerID id;
12 }GameAttributes;
13
14 #endif // GAME_ATTRIBUTES_H_INCLUDED
```

4.9 game_engine.c File Reference

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

Functions

- Uint32 [input_timer](#) (Uint32 ms, void *param)
input_timer
- [GameAssets](#) * [init_game_assets](#) ([GameAttributes](#) *game_attributes)
init_game_assets
- [GameAttributes](#) * [init_game_attributes](#) ()
init_game_attributes
- [KeyMap](#) * [default_keymap_init](#) ()
default_keymap_init
- void [clear_graphics](#) ([GameAssets](#) *game_assets)
clear_graphics
- void [draw_graphics](#) (int player_ship_time, [GameAssets](#) *game_assets, [GameAttributes](#) *game_attributes)
draw_graphics
- void [calculate_game_assets](#) ([GameAssets](#) *game_assets, [GameAttributes](#) *game_attributes, int enemy_ship_time)
calculate_game_assets
- void [free_assets](#) ([GameAssets](#) *game_assets)
free_assets
- void [free_components](#) ([GameAssets](#) *game_assets, [GameAttributes](#) *game_attributes)
free_components
- int [keep_player_time](#) ()
keep_player_time
- int [keep_enemy_time](#) ()
keep_enemy_time
- void [game_loop](#) ([GameAssets](#) *game_assets, [KeyMap](#) *key_map, [GameAttributes](#) *game_attributes)
game_loop
- void [runtime](#) ()
game

4.9.1 Function Documentation

4.9.1.1 calculate_game_assets()

```
void calculate_game_assets (
    GameAssets * game_assets,
    GameAttributes * game_attributes,
    int enemy_ship_time )
```

[calculate_game_assets](#)

a jatek assetek mozgassat es mukodeset vezerlo szamitasok aggregalo fuggvenye

Parameters

in	<i>game_assets</i>	
in	<i>game_attributes</i>	
in	<i>enemy_ship_time</i>	

Returns

void

4.9.1.2 clear_graphics()

```
void clear_graphics (
    GameAssets * game_assets )
```

clear_graphics

kirajzoltatas előtt mindent torol a kepernyorol.

Parameters

<code>[]</code>	game_assets
-----------------	-------------

Returns

void

4.9.1.3 default_keymap_init()

```
KeyMap * default_keymap_init ( )
```

default_keymap_init

az iranyitashoz hasznalt alapertelmezett billentyuket inicializalja egy KeyMap tipusba

Returns

KeyMap

4.9.1.4 draw_graphics()

```
void draw_graphics (
    int player_ship_time,
    GameAssets * game_assets,
    GameAttributes * game_attributes )
```

draw_graphics

az osszes asset kirajzolasaert felel

Parameters

in	<i>player_ship_time</i>	
in	<i>game_assets</i>	
in	<i>game_attributes</i>	

Returns

void

4.9.1.5 free_assets()

```
void free_assets (
    GameAssets * game_assets )
```

free_assets

felszabadítja a játék asseteket a játék bezarasa előtt.

Parameters

in	<i>game_assets</i>	
----	--------------------	--

Returns

void

4.9.1.6 free_components()

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

free_components

az osszes jatekkomponens felszabaditasaert felel.

Parameters

[]	game_assets
[]	game_attributes

Returns

void

4.9.1.7 game_loop()

```
void game_loop (
    GameAssets * game_assets,
    KeyMap * key_map,
    GameAttributes * game_attributes )
```

game_loop

a jatek fo vezerlesi logikaja

Parameters

<i>[]</i>	game_assets
<i>[]</i>	key_map
<i>[]</i>	game_attributes

Returns

void

4.9.1.8 init_game_assets()

```
GameAssets * init_game_assets (
    GameAttributes * game_attributes )
```

init_game_assets

inicializalja az osszes jatekhoz szukseges assetet.

Parameters

in, out	game_attributes	
---------	-----------------	--

Returns

GameAssets

4.9.1.9 init_game_attributes()

```
GameAttributes * init_game_attributes ( )
```

init_game_attributes

inicializálja a játék attribútumait, amelyeket aztán a vezérles használ

Returns

GameAttributes

4.9.1.10 input_timer()

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

input_timer

az inputok beolvasásának időzítéseért felel

Parameters

in	<i>ms</i>	
in	<i>param</i>	

Returns

Uint32

4.9.1.11 keep_enemy_time()

```
int keep_enemy_time ( )
```

keep_enemy_time

az ellenseges hajók időzítéséhez szükséges számítást végzi el

Returns

int

4.9.1.12 keep_player_time()

```
int keep_player_time ( )
```

keep_player_time

a játékos hajo időzítéséhez szükséges számítást végzi el

Returns

int

4.9.1.13 runtime()

```
void runtime ( )
```

game

aggregálja az összes játék működéséhez szükséges logikai függvényt. Amikor a game_loop kilep a ciklusból felszabadít mindent és kilep az SDL2ből.

Returns

void

4.10 game_engine.h

```
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 "hit_management.h"
14 #include "data_transfer_types.h"
15 #include "file_management.h"
16 #include "random_number_in_interval.h"
17 #include "texture_data.h"
18
19 #include <stdio.h>
20 #include <stdlib.h>
21 #include <stdbool.h>
22 #include <SDL.h>
23
24 #include "debugmalloc.h"
25
26 void runtime();
27
28 #endif // GAME_ENGINE_H_INCLUDED
```

4.11 graphics.c File Reference

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

Functions

- `SDL_Texture *` [load_sdl_texture](#) (`char *img_name`)
load_sdl_texture
- `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_crosshair](#) (`int x_coor`, `int y_coor`)
draw_crosshair
- `void` [draw_enemy_ships](#) (`EnemyArmada *armada`)
draw_enemy_ships
- `void` [draw_phaser](#) (`PhaserBeam *phaser`)
draw_phaser
- `void` [draw_torpedo](#) (`TorpedoShot *torpedoes`)
draw_torpedo
- `void` [clear_screen](#) ()
clear_screen
- `void` [render_screen](#) ()
render_screen
- `void` [destroy_textures](#) ()
destroy_textures

4.11.1 Function Documentation

4.11.1.1 clear_screen()

```
void clear_screen ( )
```

`clear_screen`

torol mindent a jatekablakbol

Returns

`void`

4.11.1.2 create_textures()

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

`create_textures`

felepiti a texturakat

Parameters

<code>[]</code>	fed
<code>[]</code>	enemy

Returns

void

4.11.1.3 create_window()

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

create_window

legeneralja a jatekablakot

Parameters

<code>[]</code>	width
<code>[]</code>	height

Returns

void

4.11.1.4 destroy_textures()

```
void destroy_textures ( )
```

destroy_textures

torli a texturakat

Returns

void

4.11.1.5 draw_background()

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

draw_background

kirajzolja a hatteret

Parameters

<code>[]</code>	<code>sm</code>
-----------------	-----------------

Returns

`void`

4.11.1.6 draw_crosshair()

```
void draw_crosshair (
    int x_coor,
    int y_coor )
```

`draw_crosshair`

kirajzolja a celkeresztet

Parameters

<code>[]</code>	<code>x_coor</code>
<code>[]</code>	<code>y_coor</code>

Returns

`void`

4.11.1.7 draw_enemy_ships()

```
void draw_enemy_ships (
    EnemyArmada * armada )
```

`draw_enemy_ships`

kirajzolja az ellenseges hajokat

Parameters

<code>[]</code>	<code>armada</code>
-----------------	---------------------

Returns

`void`

4.11.1.8 draw Phaser()

```
void draw Phaser (
    PhaserBeam * Phaser )
```

draw Phaser

kirajzolja a fezer sugarat

Parameters

<i>[]</i>	Phaser
-----------	--------

Returns

void

4.11.1.9 draw Player Ship()

```
void draw Player Ship (
    PlayerShip * ps )
```

draw Player Ship

kirajzolja a jatekos hajot

Parameters

<i>[]</i>	ps
-----------	----

Returns

void

4.11.1.10 draw Torpedo()

```
void draw Torpedo (
    TorpedoShot * torpedoes )
```

draw Torpedo

kirajzolja a kilott torpedot

Parameters

<code>[]</code>	<code>torpedoes</code>
-----------------	------------------------

Returns

void

4.11.1.11 load_sdl_texture()

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

load_sdl_texture

betölti az SDL által használt texturákat

Parameters

<code>[]</code>	<code>img_name</code>
-----------------	-----------------------

Returns

SDL_Texture

4.11.1.12 render_screen()

```
void render_screen ( )
```

render_screen

rendereli a jatekablakot

Returns

void

4.12 graphics.h

```
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 "phaser.h"
9 #include "torpedo.h"
```

```

10
11 #include <SDL.h>
12 #include <SDL_image.h>
13 #include <SDL2_gfxPrimitives.h>
14 #include <math.h>
15 #include <stdlib.h>
16 #include <stdbool.h>
17
18 #include "debugmalloc.h"
19
20 void create_window(int width, int height);
21
22 void draw_background(StarMap *sm);
23
24 void clear_background(StarMap *sm);
25
26 void draw_player_ship(PlayerShip *ps);
27
28 void clear_player_ship(PlayerShip *ps);
29
30 void draw_enemy_ship(EnemyArmada *armada);
31
32 void clear_enemy_ship(EnemyArmada *armad);
33
34 void render_screen();
35
36 #endif // GRAPHICS_H_INCLUDED

```

4.13 hit_management.c File Reference

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

Functions

- void [detect_player_hit](#) ([GameAssets](#) *game_assets, [GameAttributes](#) *game_attributes)
detect_player_hit
- void [manage_hits](#) ([GameAssets](#) *game_assets, [GameAttributes](#) *game_attributes)
manage_hits

4.13.1 Function Documentation

4.13.1.1 detect_player_hit()

```

void detect_player_hit (
    GameAssets * game_assets,
    GameAttributes * game_attributes )

```

[detect_player_hit](#)

ellenorzi, hogy barmelyik ellenseges hajo talalatot kapott-e.

Parameters

<i>[]</i>	game_assets
<i>[]</i>	game_attributes

Returns

void

4.13.1.2 manage_hits()

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

manage_hits

aggregalo fuggveny a hit management logikahoz

Parameters

<code>[]</code>	game_assets
<code>[]</code>	game_attributes

Returns

void

4.14 hit_management.h

```
1 #ifndef HIT_MANAGEMENT_H_INCLUDED
2 #define HIT_MANAGEMENT_H_INCLUDED
3
4 #include "game_assets.h"
5 #include "game_attributes.h"
6 #include "torpedo.h"
7 #include "enemy_ship.h"
8
9 #include <stdbool.h>
10
11 #include "debugmalloc.h"
12
13 #endif // HIT_MANAGEMENT_H_INCLUDED
```

4.15 input_state_interface.h

```
1 #ifndef INPUT_STATE_INTERFACE_H_INCLUDED
2 #define INPUT_STATE_INTERFACE_H_INCLUDED
3
4 #include <stdbool.h>
5
10 typedef struct mouseposition{
11     int mouse_x;
12     int mouse_y;
13 }MousePosition;
14
19 typedef struct inputstateinterface{
20     bool quit;
21     bool up;
22     bool down;
23     bool left;
24     bool right;
25     bool torpedo;
26     bool torpedo_ready;
27     bool left_mouse_button;
```

```

28     bool right_mouse_button;
29     MousePosition mouse_position;
30     bool phaser_ready;
31     bool phaser_firing;
32 } InputStateInterface;
33
34 #endif // INPUT_STATE_INTERFACE_H_INCLUDED

```

4.16 keymap.h

```

1 #ifndef KEYMAP_H_INCLUDED
2 #define KEYMAP_H_INCLUDED
3
4 typedef struct keymap{
5     char *upkey;
6     char *downkey;
7     char *leftkey;
8     char *rightkey;
9     char *torpedokey;
10 } KeyMap;
11
12 #endif // KEYMAP_H_INCLUDED

```

4.17 phaser.c File Reference

```
#include "phaser.h"
```

Functions

- [Beam](#) [create_beam_attributes](#) ([BeamColor](#) core, [BeamColor](#) falloff)
create_beam_attributes
- [PhaserBeam](#) * [phaser_init](#) ([Beam](#) beam_att, int x_coor, int y_coor)
phaser_init
- void [free Phaser](#) ([PhaserBeam](#) *phaser)
phaser_init

4.17.1 Function Documentation

4.17.1.1 create_beam_attributes()

```

Beam create_beam_attributes (
    BeamColor core,
    BeamColor falloff )

```

[create_beam_attributes](#)

legeneralja a fezer sugar kirajzolasahoz szukseges attributum tipust.

Parameters

<i>[]</i>	core
<i>[]</i>	falloff

Returns

void

4.17.1.2 free Phaser()

```
void free Phaser (
    PhaserBeam * Phaser )
```

Phaser_init

felszabadítja a Phaser memóriaterületet.

Parameters

[]	*Phaser
----	---------

Returns

void

4.17.1.3 Phaser_init()

```
PhaserBeam * Phaser_init (
    Beam beam_att,
    int x_coor,
    int y_coor )
```

Phaser_init

inicializálja a sugarat.

Parameters

[]	beam_att
[]	x_coor
[]	y_coor

Returns

void

4.18 Phaser.h

```
1 #ifndef PHASER_H_INCLUDED
```

```

2 #define PHASER_H_INCLUDED
3
4 #include "input_state_interface.h"
5 #include "debugmalloc.h"
6
11 typedef struct beamcolor{
12     int r;
13     int g;
14     int b;
15     int a;
16 }BeamColor;
17
22 typedef struct beam{
23     BeamColor core_color;
24     BeamColor falloff_color;
25 }Beam;
26
31 typedef struct phaserbeam{
32     int beg_x;
33     int beg_y;
34     int end_x;
35     int end_y;
36     Beam beam_composition;
37 }PhaserBeam;
38
39 #endif // PHASER_H_INCLUDED

```

4.19 player_ship.c File Reference

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

Functions

- [PlayerShip](#) * [init_player_ship](#) (int width, int height, [TextureData](#) texture_data, int health, int speed)
init_player_ship
- void [move_player_ship](#) ([PlayerShip](#) *ps, [InputStateInterface](#) *isi, int width, int height)
move_player_ship
- void [fire Phaser](#) ([InputStateInterface](#) *isi, [PlayerShip](#) *ps, int elapsed_interval)
fire Phaser
- void [free_player_ship](#) ([PlayerShip](#) *ps)
move_player_ship

4.19.1 Function Documentation

4.19.1.1 fire Phaser()

```

void fire Phaser (
    InputStateInterface * isi,
    PlayerShip * ps,
    int elapsed_interval )

```

fire Phaser

Kilo egy fezersugarat.

Parameters

in, out	<i>isi</i>	pointer to an InputStateInterface type
in, out	<i>ps</i>	pointer to a PlayerShip type
in	<i>elapsed_interval</i>	marker of elapsed ticks since program started

Returns

void

4.19.1.2 free_player_ship()

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

move_player_ship

felszabadítja a játékos hajókat

Parameters

[]	ps
----	----

Returns

void

4.19.1.3 init_player_ship()

```
PlayerShip * init_player_ship (
    int width,
    int height,
    TextureData texture_data,
    int health,
    int speed )
```

init_player_ship

inicializálja a játékos hajókat

Parameters

[]	width
[]	height
[]	texture_data
[]	health
[]	speed

Returns

PlayerShip

4.19.1.4 move_player_ship()

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

move_player_ship

a játékos hajójának mozgásáért felelős számításokat végzi

Parameters

[]	ps
[]	isi
[]	width
[]	height

Returns

void

4.20 player_ship.h

```
1 #ifndef PLAYER_SHIP_H_INCLUDED
2 #define PLAYER_SHIP_H_INCLUDED
3
4 #include "input_state_interface.h"
5 #include "texture_data.h"
6 #include "player_ship.h"
7 #include "phaser.h"
8
9 #include <stdbool.h>
10
11 #include "debugmalloc.h"
12
13 typedef struct playership{
14     int y_coor;
15     int x_coor;
16     TextureData texture_data;
17     PhaserBeam *phaser_blast;
18     int phaser_timer;
19     int health;
20     int speed;
21 }PlayerShip;
22
23 PlayerShip *init_player_ship(int width, int height, TextureData texture_data, int health, int speed);
24
25 void move_player_ship(PlayerShip *ps, InputStateInterface *isi, int width, int height);
26
27 void free_player_ship(PlayerShip *ps);
28
29 #endif // PLAYER_SHIP_H_INCLUDED
```


4.21 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

4.21.1 Function Documentation

4.21.1.1 random_number_in_range()

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

random_number_in_range

egy random számmal ter vissza egy meghatározott intervallumon belül. Csak ez a modul hívhatja

Parameters

in	<i>lower</i>	az intervallum alsó határa.
in	<i>upper</i>	az intervallum felső határa.

Returns

int

4.22 random_number_in_interval.h

```
1 #ifndef RANDOM_NUMBER_IN_INTERVAL_H_INCLUDED  
2 #define RANDOM_NUMBER_IN_INTERVAL_H_INCLUDED  
3  
4 int random_number_in_range(int lower, int upper);  
5  
6 #endif // RANDOM_NUMBER_IN_INTERVAL_H_INCLUDED
```

4.23 star_map.c File Reference

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

Functions

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

4.23.1 Function Documentation

4.23.1.1 advance_starmap_frame()

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

`advance_starmap_frame`

a csillagterketet 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 ennél az értéknél magasabb az adott csillag y értéke, úgy az y koordinatát 0-ra, az x koordinatát pedig egy, a képernyő szélességében található random értékre állítja.

Parameters

out	<i>sm</i>	egy StarMap típusu pointer, a játék StarMap típusában tarolt csillagok koordinatait tarolja.
in	<i>width</i>	a képernyő szélessége. Erre a random szám generálásához van szükség.
in	<i>height</i>	a képernyő magassága. Erre a csillag y koordinatajának ellenőrzéséhez van szükség.

Returns

void

4.23.1.2 free_starmap()

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

`free_starmap`

Ez a függvény a parameterként kapott csillagterket csillagainak listáját, majd magát a csillagterketet szabadítja fel.

Parameters

in	sm	a felszabaditando csillagterkep pointer.
----	----	--

Returns

void

4.23.1.3 starmap_init()

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

starmap_init

Ez a függvény inicializálja a StarMap csillagterkep listáját. létrehoz egy, a csillagok vart szamanak megfelelo hosszúsagu 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

4.24 star_map.h

```
1 #ifndef STAR_MAP_H_INCLUDED
2 #define STAR_MAP_H_INCLUDED
3
4 #include "random_number_in_interval.h"
5
6 #include "debugmalloc.h"
7
14 typedef struct starcolor{
15     int r;
16     int g;
17     int b;
18     int a;
19 }StarColor;
20
27 typedef struct star{
28     int y_coor;
29     int x_coor;
30     int radius;
31 }Star;
32
38 typedef struct starmap{
39     int length;
40     Star *stars;
41     StarColor color;
42 }StarMap;
43
44
```

```

45 StarMap *starmap_init(int width, int height);
46
47
48 void advance_starmap_frame(StarMap *sm, int width, int height);
49
50
51 void free_starmap(StarMap *sm);
52
53 #endif // STAR_MAP_H_INCLUDED

```

4.25 string_operations.c File Reference

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

Functions

- bool [dinstr_alloc](#) ([DinStr](#) *str, int size)
dinstr_alloc

4.25.1 Function Documentation

4.25.1.1 dinstr_alloc()

```

bool dinstr_alloc (
    DinStr * str,
    int size )

```

[dinstr_alloc](#)

dinamikus sztringet allokal

Parameters

<i>[]</i>	str
<i>[]</i>	size

Returns

bool

4.26 string_operations.h

```

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

```

4.27 texture_data.h

```

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

```

4.28 torpedo.c File Reference

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

Functions

- [TorpedoColors](#) [init_torpedo_colors](#) (bool is_enemy_torpedo, bool is_quantum_torpedo)
init_torpedo_colors
- [TorpedoShot](#) * [add_torpedo_shot](#) ([TorpedoShot](#) *torpedoes, int damage, int speed, int x_coor, int y_coor, bool is_enemy_torpedo, bool is_quantum_torpedo)
add_torpedo_shot
- void [move_torpedoes](#) ([TorpedoShot](#) **torpedo, [GameAttributes](#) *game_attributes)
move_torpedoes
- void [remove_torpedo_shot](#) ([TorpedoShot](#) **torpedo)
remove_torpedo_shot
- void [free_torpedoes](#) ([TorpedoShot](#) *torpedoes)
free_torpedoes

4.28.1 Function Documentation

4.28.1.1 add_torpedo_shot()

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

add_torpedo_shot

hozzaad a kilott torpedok listajahoz egy ujabb elemet

Parameters

[]	torpedoes
[]	damage
[]	speed
[]	x_coor
[]	y_coor
[]	is_enemy_torpedo
[]	is_quantum_torpedo

Returns

TorpedoShot

4.28.1.2 free_torpedoes()

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

free_torpedoes

felszabaditja a torpedok listajat

Parameters

[]	torpedoes
----	-----------

Returns

void

4.28.1.3 init_torpedo_colors()

```
TorpedoColors init_torpedo_colors (
    bool is_enemy_torpedo,
    bool is_quantum_torpedo )
```

init_torpedo_colors

inicializálja a torpedók színeit adó TorpedoColors structot

Parameters

[]	is_enemy_torpedo
[]	is_quantum_torpedo

Returns

TorpedoColors

4.28.1.4 move_torpedoes()

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

move_torpedoes

a torpedók mozgáshoz szükséges számításokat végzi

Parameters

[]	torpedo
[]	game_attributes

Returns

void

4.28.1.5 remove_torpedo_shot()

```
void remove_torpedo_shot (
    TorpedoShot ** torpedo )
```

remove_torpedo_shot

amennyiben a torpedo eltalál valamit, vagy kimegy a játéktérből, törli azt.

Parameters

<code>[]</code>	<code>torpedo</code>
-----------------	----------------------

Returns

void

4.29 torpedo.h

```

1 #ifndef TORPEDO_H_INCLUDED
2 #define TORPEDO_H_INCLUDED
3
4 #include "game_attributes.h"
5
6 #include <stdbool.h>
7
8 #include "debugmalloc.h"
9
14 typedef struct colordata{
15     int r;
16     int g;
17     int b;
18     int a;
19 }ColorData;
20
25 typedef struct torpedocolors{
26     ColorData outter_ring;
27     ColorData inner_ring;
28     ColorData center;
29 }TorpedoColors;
30
35 typedef struct torpedoshot{
36     int x_coor;
37     int y_coor;
38     int damage;
39     int speed;
40     int dir;
41     TorpedoColors colors;
42     struct torpedoshot *next_torpedo;
43     struct torpedoshot *prev_torpedo;
44 }TorpedoShot;
45
46 #endif // TORPEDO_H_INCLUDED

```

4.30 ui_input.c File Reference

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

Functions

- void `user_input` ([InputStateInterface](#) *isi, [KeyMap](#) *key_map, SDL_TimerID id)
user_input

4.30.1 Function Documentation

4.30.1.1 user_input()

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

user_input

A felhasználótól érkező billentyűparancsokat értelmezi, és egy interface-n keresztül adja át a program többi részének

Parameters

in, out	<i>isi</i>	a játék InputStateInterface-re mutató pointer. Ezen keresztül kommunikálnak egymással a vezérlőmodulok.
in	<i>key_map</i>	ez a vezérlő KeyMap interfácen keresztül hasonlítja össze a bejövő billentyűparancsokat a valid vezérlő gombokkal.
in	<i>id</i>	egy SDL_TimerID típusú időzítő. Feladata, hogy generáljon egy SDL_USEREVENT-et, amennyiben az időzítő lejártával nincs beérkező esemény/parancs (enélkül a vezérlő blokkolná a program futását, nem működne a háttér animáció, és semmi nem történne, amíg nincs felhasználói interakció).

Returns

void

4.31 ui_input.h

```
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 "debugmalloc.h"
13
14 void user_input(InputStateInterface *isi, KeyMap *key_map, SDL_TimerID id);
15
16 #endif // UI_INPUT_H_INCLUDED
```


Index

- add_torpedo_shot
 - torpedo.c, [55](#)
- advance_starmap_frame
 - star_map.c, [52](#)
- beam, [5](#)
- beamcolor, [5](#)
- calculate_game_assets
 - game_engine.c, [33](#)
- clear_graphics
 - game_engine.c, [34](#)
- clear_screen
 - graphics.c, [39](#)
- colordata, [6](#)
- create_beam_attributes
 - phaser.c, [46](#)
- create_enemy_ship
 - enemy_ship.c, [26](#)
- create_textures
 - graphics.c, [39](#)
- create_window
 - graphics.c, [40](#)
- DebugmallocData, [6](#)
- DebugmallocEntry, [7](#)
- default_keymap_init
 - game_engine.c, [34](#)
- destroy_textures
 - graphics.c, [40](#)
- detect_player_hit
 - hit_management.c, [44](#)
- DinStr, [7](#)
- dinstr_alloc
 - string_operations.c, [54](#)
- draw_background
 - graphics.c, [40](#)
- draw_crosshair
 - graphics.c, [41](#)
- draw_enemy_ships
 - graphics.c, [41](#)
- draw_graphics
 - game_engine.c, [34](#)
- draw_phaser
 - graphics.c, [41](#)
- draw_player_ship
 - graphics.c, [42](#)
- draw_torpedo
 - graphics.c, [42](#)
- enemy_armada_entry_animation
 - enemy_ship.c, [26](#)
- enemy_ship.c, [25](#)
 - create_enemy_ship, [26](#)
 - enemy_armada_entry_animation, [26](#)
 - free_enemy_armada, [27](#)
 - free_enemy_squadron, [27](#)
 - free_enemy_squadron_ship, [27](#)
 - init_enemy_armada, [28](#)
 - manage_enemy_dirs, [28](#)
 - modify_enemy_dir, [29](#)
 - move_enemy_armada, [29](#)
 - position_enemy_armada, [30](#)
 - squadron_size, [30](#)
- enemyarmada, [7](#)
- enemyship, [8](#)
- enemysquadronship, [8](#)
- file_management.c, [31](#)
 - import_ship_dtt, [31](#)
- fire_phaser
 - player_ship.c, [48](#)
- free_assets
 - game_engine.c, [35](#)
- free_components
 - game_engine.c, [35](#)
- free_enemy_armada
 - enemy_ship.c, [27](#)
- free_enemy_squadron
 - enemy_ship.c, [27](#)
- free_enemy_squadron_ship
 - enemy_ship.c, [27](#)
- free_phaser
 - phaser.c, [47](#)
- free_player_ship
 - player_ship.c, [49](#)
- free_starmap
 - star_map.c, [52](#)
- free_torpedoes
 - torpedo.c, [56](#)
- game_engine.c, [32](#)
 - calculate_game_assets, [33](#)
 - clear_graphics, [34](#)
 - default_keymap_init, [34](#)
 - draw_graphics, [34](#)
 - free_assets, [35](#)
 - free_components, [35](#)
 - game_loop, [36](#)
 - init_game_assets, [36](#)
 - init_game_attributes, [36](#)

- input_timer, 37
- keep_enemy_time, 37
- keep_player_time, 37
- runtime, 38
- game_loop
 - game_engine.c, 36
- gameassets, 9
- gameattributes, 9
- graphics.c, 38
 - clear_screen, 39
 - create_textures, 39
 - create_window, 40
 - destroy_textures, 40
 - draw_background, 40
 - draw_crosshair, 41
 - draw_enemy_ships, 41
 - draw Phaser, 41
 - draw_player_ship, 42
 - draw_torpedo, 42
 - load_sdl_texture, 43
 - render_screen, 43
- hit_management.c, 44
 - detect_player_hit, 44
 - manage_hits, 45
- import_ship_dtt
 - file_management.c, 31
- init_enemy_armada
 - enemy_ship.c, 28
- init_game_assets
 - game_engine.c, 36
- init_game_attributes
 - game_engine.c, 36
- init_player_ship
 - player_ship.c, 49
- init_torpedo_colors
 - torpedo.c, 56
- input_timer
 - game_engine.c, 37
- inputstateinterface, 10
- keep_enemy_time
 - game_engine.c, 37
- keep_player_time
 - game_engine.c, 37
- keymap, 10
- leveldtt, 11
- load_sdl_texture
 - graphics.c, 43
- manage_enemy_dirs
 - enemy_ship.c, 28
- manage_hits
 - hit_management.c, 45
- modify_enemy_dir
 - enemy_ship.c, 29
- mouseposition, 11
- move_enemy_armada
 - enemy_ship.c, 29
- move_player_ship
 - player_ship.c, 50
- move_torpedoes
 - torpedo.c, 57
- Phaser.c, 46
 - create_beam_attributes, 46
 - free Phaser, 47
 - Phaser_init, 47
- Phaser_init
 - Phaser.c, 47
- Phaserbeam, 12
- player_ship.c, 48
 - fire Phaser, 48
 - free_player_ship, 49
 - init_player_ship, 49
 - move_player_ship, 50
- playership, 12
- position_enemy_armada
 - enemy_ship.c, 30
- random_number_in_interval.c, 51
 - random_number_in_range, 51
- random_number_in_range
 - random_number_in_interval.c, 51
- remove_torpedo_shot
 - torpedo.c, 57
- render_screen
 - graphics.c, 43
- runtime
 - game_engine.c, 38
- shipdtt, 13
- spritemapdata, 13
- squadron_size
 - enemy_ship.c, 30
- star, 14
- star_map.c, 51
 - advance_starmap_frame, 52
 - free_starmap, 52
 - starmap_init, 53
- starcolor, 14
- starmap, 15
- starmap_init
 - star_map.c, 53
- string_operations.c, 54
 - dinstr_alloc, 54
- texturedata, 16
- torpedo.c, 55
 - add_torpedo_shot, 55
 - free_torpedoes, 56
 - init_torpedo_colors, 56
 - move_torpedoes, 57
 - remove_torpedo_shot, 57
- torpedocolors, 16
- torpedoshot, 16

ui_input.c, [58](#)
 user_input, [58](#)
user_input
 ui_input.c, [58](#)