

Asteroids fejlesztői dokumentáció

Külső könyvtárak és források:

A program az **SDL2** grafikus könyvtárat használja a grafikus elemek megjelenítésére és hangok lejátszására. Az **SDL2** használata nélkül a program **nem fordul le!**

A program a **debugmalloc.h** könyvtárat használja a memóriaszivárgások megkeresésére.

A program a grafikus és hang forrásait az **assets** könyvtárból tölti be. A zene ezen a könyvtáron belül a **music** könyvtárba van, a hangeffektek pedig az **sfx** könyvtárba vannak.

A dicsőséglista a **top_lista.txt** fájlból olvassa be és menti el a legjobb eredményeket.

A **buid.bat** script a program fordítását végzi Windows 10 operációs rendszer alatt.

A program belső felépítése:

asteroidmanager.h

- Az aszteroidák, robbanások és a főbb játékelemek kezelése itt történik. A érintkezések vizsgálata itt történik meg, a pontok hozzáadása, az aszteroidák idézése, eltüntetése, szétrobbantása itt történik meg.

a robbanás struct, tartalmazza a pozíciót és azt, hogy meddig él még, valamint egy pintert a következő lista elemre:

AsteroidExplosion

- **Vector2D** pos
- **double** lifetime
- **struct AsteroidExplosion*** next_explosion (A lista következő eleme)

A robbanás irányító szerkezete, tartalmazza a textúráját a robbanásnak és a láncolt robbanás lista fejét:

Explosionmanager

- **SDL_Texture*** explosion_atlas
 - **struct AsteroidExplosion*** next_explosion (A lista következő eleme)
-

Az aszteroida szerkezete, van pozíciója, iránya, forgása, rádiusza és típusa, 3 típus van, kicsi, nagy és nagyon nagy aszteroida. Tartalmaz egy pointert a következő aszteroida elemre:

Asteroid

- `SDL_Texture* self_texture`
- `Vector2D vel`
- `Vector2D pos`
- `int type`
- `bool hit`
- `double angle`
- `double radius`
- `int value`
- `struct Asteroid* nextAsteroid` (A lista következő eleme)

Az aszteroida idéző idézi az aszteroidákat, tárolja a robbanásokat, a robbanáskezelőt és az aszteroidák láncolt listáját:

AsteroidSpawner

- `SDL_Texture* sprite_large`
- `SDL_Texture* sprite_mid`
- `SDL_Texture* sprite_small`
- `SDL_Texture* bg_image`
- `double frequency`
- `double timer`
- `struct Explosionmanager* exp_manager`
- `struct Asteroid* nextAsteroid` (A lista következő eleme)

```
void render_background(AsteroidSpawner* asteroid_spawner, RenderingContext* rendering_context):
```

Kirajzolja a háttérbe lévő aszteroidákat.

Paraméterek:

- `AsteroidSpawner* asteroid_spawner`: Referencia az aszteroida kezelőre (Pointer)
- `RenderingContext* rendering_context`: A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

```
void check_player_hit(Player* player, AsteroidSpawner* asteroid_spawner):
```

Megnézi, hogy a játékos el lett-e találva egy aszteroida által.

Paraméterek:

- `Player* player`: Játékos (*Pointer*)
- `AsteroidSpawner* asteroid_spawner`: Referencia az aszteroida kezelőre (*Pointer*)

(Nincs visszatérési értéke)

`void setup_asteroid_spawner(AsteroidSpawner* asteroid_spawner, RenderingContext* context, int difficulty):`

Inicializálja az aszteroida kezelőt, és elemeit, betölti a textúrákat.

Paraméterek:

- `AsteroidSpawner* asteroid_spawner`: Referencia az aszteroida kezelőre (*Pointer*)
- `RenderingContext* context`: A rajzoló kontextus (*Pointer*)
- `int difficulty`: A nehézség azonosítója

(Nincs visszatérési értéke)

`void clear_asteroids(AsteroidSpawner* asteroid_spawner):`

Kitörli az összes aszteroidát.

Paraméterek:

- `AsteroidSpawner* asteroid_spawner`: Referencia az aszteroida kezelőre (*Pointer*)

(Nincs visszatérési értéke)

`void cleanup_asteroid_manager(AsteroidSpawner* asteroid_spawner):`

Felszabadítja az aszteroida kezelő és elemei lefoglalt memóriáját.

Paraméterek:

- `AsteroidSpawner* asteroid_spawner`: Referencia az aszteroida kezelőre (*Pointer*)

(Nincs visszatérési értéke)

`Vector2D get_random_asteroid_start(void):`

Egy véletlenszerű aszteroida kezdőpozíciót ad vissza, a képernyő valamelyik szélén.

Paraméterek:

- (nincs)

Kimenet: Egy véletlenszerű kezdőpozíció

Vector2D get_random_asteroid_vel(Vector2D position):

Véletlenszerű aszteroidasebességet ad vissza (és irányt ami kb. a pálya közepe fele mutat).

Paraméterek:

- Vector2D position: 2 dimenziós vektor, az aszteroida kezdőpozíciója

Kimenet: Egy véletlenszerű kezdősebesség.

void try_append_asteroid(AsteroidSpawner* asteroid_spawner, Asteroid* asteroid):

Hozzáadja az új aszteroidát az aszteroidák listájához.

Paraméterek:

- AsteroidSpawner* asteroid_spawner: Referencia az aszteroida kezelőre (Pointer)
- Asteroid* asteroid: Referencia az aszteroidára (Pointer)

(Nincs visszatérési értéke)

void shatter_asteroid(AsteroidSpawner* asteroid_spawner, int type, Vector2D pos, ScoreManager* score_manager, Player* player):

Széttöri az aszteroidát kisebb darabokra és pontot ad a játékosnak a találatért. Ha nagyon kicsi volt az aszteroida akkor csak eltűnik.

Paraméterek:

- AsteroidSpawner* asteroid_spawner: Referencia az aszteroida kezelőre (Pointer)
- int type: Az aszteroida mérete (típus szerint)
- Vector2D pos: 2 dimenziós vektor, ahol eltalálták az aszteroidát.
- ScoreManager* score_manager: A pontszám kezelő (Pointer)
- Player* player: Játékos (Pointer)

(Nincs visszatérési értéke)

void update_bullet_hits(BulletManager* bullet_manager, AsteroidSpawner* asteroid_spawner, ScoreManager* score_manager, Player* player):

Megnézi, hogy kaptak-e az aszteroidák találatot vagy eltalálták-e az aszteroidák a játékost.

Paraméterek:

- BulletManager* bullet_manager: Referencia a lövedékkezelőre (Pointer)
- AsteroidSpawner* asteroid_spawner: Referencia az aszteroida kezelőre (Pointer)
- ScoreManager* score_manager: A pontszám kezelő (Pointer)
- Player* player: Játékos (Pointer)

(Nincs visszatérési értéke)

void spawn_asteroid(AsteroidSpawner* asteroid_spawner):

Inicializál egy új aszteroidát, lefoglalja a memóriát neki és berakja az aszteroidák láncolt listába.

Paraméterek:

- **AsteroidSpawner* asteroid_spawner:**Referencia az aszteroida kezelőre (Pointer)

(Nincs visszatérési értéke)

void try_spawn_asteroid(AsteroidSpawner* asteroid_spawner, GameState* game_state):

Megpróbál idézni egy aszteroidát, bizonyos időközönként, ami nehézségtől függ.

Paraméterek:

- **AsteroidSpawner* asteroid_spawner:**Referencia az aszteroida kezelőre (Pointer)
- **GameState* game_state:**Referencia a játékállapotr, innen kell a delta idő a pontos idézéshez. (Pointer)

(Nincs visszatérési értéke)

void update_asteroids(AsteroidSpawner* asteroid_spawner, GameState* game_state):

Frissíti, mozgatja az aszteroidákat, ha kimentek a pályáról akkor eltünteti őket a függvény.

Paraméterek:

- **AsteroidSpawner* asteroid_spawner:**Referencia az aszteroida kezelőre (Pointer)
- **GameState* game_state:**Referencia a játékállapotr (Pointer)

(Nincs visszatérési értéke)

void render_asteroid(Asteroid* asteroid, RenderingContext* rendering_context):

Kirajzol egy aszteroidát.

Paraméterek:

- **Asteroid* asteroid:**Referencia az aszteroidára, amit ki kell rajzolni (Pointer)
- **RenderingContext* rendering_context:**A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

```
void render_asteroids(AsteroidSpawner* asteroid_spawner, RenderingContext* rendering_context):
```

Végigmegy minden aszteroidán és meghívja a kirajolást mindegyiken.

Paraméterek:

- AsteroidSpawner* asteroid_spawner:Referencia az aszteroida kezelőre (Pointer)
- RenderingContext* rendering_context:A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

```
void render_explosions(AsteroidSpawner* asteroid_spawner, RenderingContext* rendering_context):
```

Kirajzolja a robbanásokat.

Paraméterek:

- AsteroidSpawner* asteroid_spawner:Referencia az aszteroida kezelőre, amin belül van a robbanáskezelő (Pointer)
- RenderingContext* rendering_context:A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

```
void add_explosion(AsteroidSpawner* asteroid_spawner,AsteroidExplosion* new_explosion):
```

Hozzáad egy robbanást a robbanások láncolt Listájához.

Paraméterek:

- AsteroidSpawner* asteroid_spawner:Referencia az aszteroida kezelőre (Pointer)
- AsteroidExplosion* new_explosion:Referencia a robbanáskezelőre (Pointer)

(Nincs visszatérési értéke)

```
void spawn_explosion(AsteroidSpawner* asteroid_spawner,Vector2D pos):
```

Inicializál egy robbanást egy megadott helyen.

Paraméterek:

- AsteroidSpawner* asteroid_spawner:Referencia az aszteroida kezelőre (Pointer)
- Vector2D pos:2 dimenziós vektor, ezen a pozíción fog a robbanás megjelenni.

(Nincs visszatérési értéke)

```
void update_explosions(AsteroidSpawner* asteroid_spawner, GameState* game_state):
```

Frissíti a robbanásokat. Ha lejárt az élettartamuk akkor kitörli őket.

Paraméterek:

- `AsteroidSpawner* asteroid_spawner`: Referencia az aszteroida kezelőre (Pointer)
- `GameState* game_state`: Referencia a játékállapotr (Pointer)

(Nincs visszatérési értéke)

audiomanager.h

- A hanglejátszáshoz fontos adatstruktúra, függvények, betöltések vannak itt.

Az audio kezelő, eltárolja az audio lejátszó azonosítót:

AudioManager struktúrája tartalmazza:

- `SDL_AudioDeviceID audio_device`: Az SDL hang lejátszásához szükséges változó.

`int init_audio(AudioManager* aman):`

Inicializálja a paraméterként kapott hanglejátszót. Negatív visszatérési érték jelzi a hibát.

Paraméterek:

- `AudioManager* aman`: A hanglejátszó amit inicializálni kell. (Pointer)

Kimenet: Egy egész szám, ami indikálja, hogy volt-e hiba. A negatív kimenet jelzi a hibát.

`int play_sound(AudioManager* aman, const char* file_path):`

Lejátszik egy hangot a paraméterként kapott hanglejátszóval. a file_path az a .wav hang file útvonala.

Paraméterek:

- `AudioManager* aman`: A hanglejátszó aminek le kell játszani a hangot (Pointer)
- `const char* file_path`: A lejátszandó .wav hangfájl elérési útvonala (Pointer)

Kimenet: Egy egész szám, ami indikálja, hogy volt-e hiba. A negatív kimenet jelzi a hibát.

`void cleanup_audio(AudioManager* aman):`

Felszabadítja a paraméterként kapott hanglejátszót.

Paraméterek:

- `AudioManager* aman`: A hanglejátszó, amit fel kell szabadítani. (Pointer)

(Nincs visszatérési értéke)

bulletmanager.h

- A lövedékkezelő és a lövedékhez szükséges adatstruktúrák, függvények, definíciók vannak itt.

A lövedék elem adatstruktúrája, tartalmazza a pozíciót, az irányt, azt, hogy talált-e és a szögét, valamint a következő lövedékre mutató pointert:

Bullet

- Vector2D pos
- Vector2D vel
- bool hit
- double angle
- struct Bullet* nextBullet (A lista következő eleme)

A lövedék kezelő struktúrája, tartalmazza a lövedékek láncolt listájának a fejét és a lövedék textúrát:

BulletManager

- SDL_Texture* bullet_texture
- struct Bullet* nextBullet (A lista feje)

void setup_bullet_manager(BulletManager* bullet_manager, RenderingContext* context):

Inicializálja a lövedék kezelőt, betölti a lövedék textúráját.

Paraméterek:

- BulletManager* bullet_manager: Referencia a lövedékkezelőre (Pointer)
- RenderingContext* context: A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

void shoot_bullet(BulletManager* bullet_manager, Player* player):

Csinál egy lövedéket a játékos iránya és pozíciója alapján és berakja a láncolt listába.

Paraméterek:

- BulletManager* bullet_manager: Referencia a lövedékkezelőre (Pointer)
- Player* player: Játékos, aminek a helyére és irányába fog a lövedék idéződni. (Pointer)

(Nincs visszatérési értéke)

void update_bullets(BulletManager* bullet_manager, GameState* game_state):

Frissíti a lövedékek helyét, letörli őket, ha kimentek a pályáról vagy eltaláltak

egy aszteroidát.

Paraméterek:

- `BulletManager* bullet_manager`: Referencia a Lövedékkezelőre (Pointer)
- `GameState* game_state`: A játékállapot adatai (Pointer)

(Nincs visszatérési értéke)

`void render_bullets(BulletManager* bullet_manager, RenderingContext* context):`

Kirajzolja a lövedékeket egyesével.

Paraméterek:

- `BulletManager* bullet_manager`: Referencia a Lövedékkezelőre (Pointer)
- `RenderingContext* context`: A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

`void cleanup_bullets(BulletManager* bullet_manager):`

Az összes lövedéket kitörli és a foglalt memóriájukat felszabadítja.

Paraméterek:

- `BulletManager* bullet_manager`: Referencia a Lövedékkezelőre (Pointer)

(Nincs visszatérési értéke)

extramath.h

- A kiegészített matematikai struktúrák, függvények vannak itt tárolva.

Egy kétdimenziós vektor struktúrája, tartalmazza az x és y koordinátát:

`Vector2D`

- `double x`
- `double y`

`Vector2D init_vec2(double x1, double y1):`

Inicializál egy 2 dimenziós vektort a paraméterként kapott x és y pozíción és visszaadja azt.

Paraméterek:

- `double x1`: Egy valós érték
- `double y1`: Egy valós érték

Kimenet: `Vector2D`

`Vector2D multiply_vec2(Vector2D v1, double lambda):`

Megszoroz egy paraméterként kapott vektort egy lambda valós számmal. Visszaadja a szorzatvektort.

Paraméterek:

- `Vector2D v1:2 dimenziós vektor`
- `double lambda:Egy valós érték`

Kimenet: `Vector2D`

`Vector2D add_vec2(Vector2D v1, Vector2D v2):`

Összead 2 darab 2 dimenziós vektort és visszaadja az összeget.

Paraméterek:

- `Vector2D v1:2 dimenziós vektor`
- `Vector2D v2:2 dimenziós vektor`

Kimenet: `Vector2D`

`double distance_vec2(Vector2D v1, Vector2D v2):`

Visszaadja az távolságot 2 db 2 dimenziós vektor között.

Paraméterek:

- `Vector2D v1:2 dimenziós vektor`
- `Vector2D v2:2 dimenziós vektor`

Kimenet: A 2 vektor közötti távolság valós értéként.

`bool overlapping(Vector2D pos1, double radius1, Vector2D pos2, double radius2):`

Megnézi, hogy 2 körnek van-e metszete. A paraméterek a 2 kör koordinátája és sugarai.

Paraméterek:

- `Vector2D pos1:2 dimenziós vektor`
- `double radius1:Egy valós érték`
- `Vector2D pos2:2 dimenziós vektor`
- `double radius2:Egy valós érték`

Kimenet: Igaz, ha a 2 kör metszi egymást, különben hamis.

`double clampd(double num, double min, double max):`

Ha az 1. paraméter kisebb, mint a minimum paraméter, akkor a minimum paramétert adja vissza, ha nagyobb mint a maximum paraméter akkor a maximum paramétert adja vissza, különben csak visszaadja az 1. paramétert.

Paraméterek:

- `double num:Egy valós érték`

- `double min:A minimum való érték`
- `double max:A maximum való érték`

Kimenet: *A megszorított érték.*

fontmanager.h

- A betűtípus, betűk kirajzolása, betöltése, felszabadítása és kirajzolásáért feleős függvények vannak itt.

A betűfelület struktúrája, egy betűfelület tartalmazza a szöveget, amit majd ki kell rajzolni és egy helyet, ami mutatja, hogy hova kell a szöveget kirajzolni:

FontSurface

- `Vector2D pos`
- `char data[]`

A betűtípuskezelő tartalmazza a betűtípushoz tartozó textúraatlaszt:

FontManager

- `SDL_Texture* cur_font`

```
void load_font_manager(FontManager* font_manager, RenderingContext* context, char path[]):
```

Betölt egy paraméterként kapott betűtípus-atlaszt a szintén paraméterként kapott betűtípuskezelőbe.

Paraméterek:

- `FontManager* font_manager:TODO (Pointer)`
- `RenderingContext* context:A rajzoló kontextus (Pointer)`
- `char path[]: Ez az atlaszkép elérési útvonala`

(Nincs visszatérési értéke)

```
void draw_letter(RenderingContext* context, FontSurface* written_surface, SDL_Texture* font_texture, int offset_x, int global_offset, int y_adder):
```

Kirajzol egy betűt egy megadott betűtípussal. Az x és y pozícióját lehet tolni a paraméterekkel.

Paraméterek:

- `RenderingContext* context:A rajzoló kontextus (Pointer)`
- `FontSurface* written_surface:Referencia a betűfelületre (Pointer)`
- `SDL_Texture* font_texture:Referencia a használt betűtípus textúrájára (Pointer)`
- `int offset_x:Az X tengelyen való eltolási érték`
- `int global_offset:A tabulációt szimuláló eltolás`
- `int y_adder:Az Y tengelyen való eltolási érték`

(Nincs visszatérési értéke)

```
void draw_letters(RenderingContext* context,FontManager*  
font_manager,FontSurface* written_surface):
```

Kirajzol egy betűfelületet egy paraméterként megadott betűtípussal.

Paraméterek:

- RenderingContext* context:A rajzoló kontextus (Pointer)
- FontManager* font_manager:A betűtípuskezelő (Pointer)
- FontSurface* written_surface:Referencia a betűfelületre (Pointer)

(Nincs visszatérési értéke)

gamemanager.h

- A játékállapot struktúráját, függvényeit, az állást tároló struktúrát tartalmazó header file.

A játékállapot helyzet megértésére ad lehetőséget :

- ```
enum State
```
- QUIT 0,
  - INGAME 1,
  - INMENU 2,
  - RESET\_SCORE 3,
  - DISPLAY\_DEATH 4,
  - LOAD\_TOPLIST 5,
  - DISPLAY\_TOPLIST 6,
  - SAVE\_STATS 7

---

*A játékállást tárolja el, tárolja a delta időt 2 frame között, az egér x és y pozícióját, hogy kattintott-e a játékos, a nehézséget és a játékállapotot:*

- ```
GameState
```
- State state
 - int last_frame_time
 - double delta_time
 - int mouseposx
 - int mouseposy
 - bool mousedown
 - int difficulty

```
void switch_difficulty(GameState* game_state):
```

Megváltoztatja a nehézséget a paraméterként kapott játékállapotban.

Paraméterek:

- GameState* game_state: A játékállapot, ahol a nehézséget állítani kell (Pointer)

(Nincs visszatérési értéke)

const char* get_difficulty_label(int difficulty):

Visszaad egy szöveget, ami jellemzi a nehézséget.

Paraméterek:

- int difficulty: Az új nehézség azonosítója

Kimenet: A nehézség neve, az azonosítótól függően.

void setup_game(GameState* gamestate):

Inicializálja a játékállást alapértékekkel.

Paraméterek:

- GameState* gamestate: A játékállás, amit inicializálni kell. (Pointer)

(Nincs visszatérési értéke)

void update(GameState* gamestate):

Frissíti a játékállást. Újra kiszámolja a delta időt.

Paraméterek:

- GameState* gamestate: A játékállás, amit frissíteni kell. (Pointer)

(Nincs visszatérési értéke)

konstansok.h

- A konstansok 1 header-be vannak, hogy könnyen lehessen konfigurálni a játékot

Konstansok:

JATEKTER_HOSSZ	1000	A játéklablak hossza pixelben
JATEKTER_MAGASSAG	600	A játéklablak magassága pixelben
FPS	50	A megcélzott FPS
FRAME_TIME	(1000/FPS)	Egy képkocka meddig látszik
RAD	0.0174532925	1 PI radián
MAX_asteroid_SPEED	2.5	Az aszteroidák maximum sebessége
BULLET_SPEED	300.0	A játékos lövedékeinek a sebessége
BULLET_RADIUS	6.0	A lövedékek kör alakú hitboxának sugara.
RELOAD_COOLDOWN	0.5	Az az idő másodpercben, amennyi idő kell, hogy minimum eltelljen 2 lövés között.
FONT_WIDTH	16	A betűk szélessége pixelben
FONT_HEIGHT	16	A betűk magassága pixelben
OUTLINE_SIZE	4	A gombon lévő fehér

		kijelöléseffektus nagysága pixelben.
PLAYER_IFRAMES	1.0	A játékos sebezhetetlenségi ideje.
TOP_SCORES_SIZE	10	Az, hogy a dicsőséglisaból hány embert tölt be a játék.
MAX_NAME_LENGTH	12	A maximum hossz, amekkora lehet a játékos neve a dicsőséglisában.
MAX_SCORE_VALUE	999999	A maximum pontszám
TOP_LIST_FILE	"top_lista.txt"	A toplista fájl neve.

main.c

- A fő forrásfájl, ebbe van a game loop, a felszabadítás és a fő inicializáló függvények.

```
int init_context(RenderingContext* context){
```

A renderert és az ablakot inicializálja a függvény;

Paraméterek:

- RenderingContext* context: A rajzóló kontextus (Pointer)

Kimenet: Egész szám, ha 0 az érték akkor nem volt hiba inicializáláskor, ha nem 0, akkor volt.

```
void init_gui(RenderingContext* context, FontManager* font_manager,
ScoreManager* score_manager, HighScoreManager* h_score_man, TopListScreen*
top_list_gui, DeathScreen* death_screen, MainMenu* main_menu){
```

A kattintható és/vagy szöveges elemeket inicializálja a függvény

Paraméterek:

- RenderingContext* context: A rajzóló kontextus (Pointer)
- FontManager* font_manager: A betűtípuskezelő (Pointer)
- ScoreManager* score_manager: A pontszám kezelő (Pointer)
- HighScoreManager* h_score_man: A toplista kezelő (Pointer)
- TopListScreen* top_list_gui: Referencia a toplista menüjére (Pointer)
- DeathScreen* death_screen: A halálképernyő (Pointer)
- MainMenu* main_menu: A főmenü (Pointer)

(Nincs visszatérési értéke)

menumanager.h

- A menük, gombok, struktúráit, függvényeit tartalmazó header file.

A gomb struktúrája, tartalmazza a funkcióját, a betűfelületet a gomb szövegével, egy nagyságot, egy pozíciót, egy attribútumot, hogy kattintottak-e rá, egy attribútumot, hogy rajta van-e az egért, és egy hanglejátszót a kattintás esetére:

Button

- Vector2D pos
- Vector2D size
- int func_type
- bool isclicked
- bool inmouse
- FontSurface* fontSurface
- AudioManager btn_click

A főmenü struktúrája, tartalmazza a főmenübe lévő textúrákat, egy hanglejátszót a zenének a főmenüben, és pointereket a start, kilépés, toplista, nehézség gombokra:

MainMenu

- Button* start_btn
- Button* quit_btn
- Button* top_list_btn
- Button* difficulty_btn
- double icon_angle
- SDL_Texture* main_title
- SDL_Texture* main_icon
- SDL_Rect* title_dst
- AudioManager music_player

void play_music(MainMenu* main_menu):

Elkezdi lejátszani a zenét a főmenübe.

Paraméterek:

- MainMenu* main_menu: A főmenü (Pointer)

(Nincs visszatérési értéke)

void check_button(Button* btn, GameState* game_state):

Megnézi, hogy egy kapott gombra kattintottak-e vagy rajta van-e a kurzor.

Paraméterek:

- Button* btn: Referencia a gombra (Pointer)
- GameState* game_state: A játékállás, ahonnan olvasható az egér lenyomása (Pointer)

(Nincs visszatérési értéke)

void setup_button(Button btn, Vector2D posi, Vector2D sizei, int func, Vector2D font_start, char datai[]):**

inicialiál egy gombot és visszaadja a gomb címét kettős indirekcióval.

Paraméterek:

- `Button** btn`:Referencia a gomb címének címére (Pointer)
- `Vector2D posi`:2 dimenziós vektor
- `Vector2D sizei`:2 dimenziós vektor
- `int func`:A gomb funkciójának azonosítója
- `Vector2D font_start`:2 dimenziós vektor
- `char datai[]`:A gombon lévő szöveg

(Nincs visszatérési értéke)

`void cleanup_button(Button* btn):`

Felszabadítja a gomb felületét, hanglejátszóját.

Paraméterek:

- `Button* btn`:Referencia a gombra (Pointer)

(Nincs visszatérési értéke)

`void draw_button(Button* btn,RenderingContext* context,FontManager* font_manager):`

Kirajzolja a kapott gombot egy kapott betűtípussal.

Paraméterek:

- `Button* btn`:Referencia a gombra (Pointer)
- `RenderingContext* context`:A rajzoló kontextus (Pointer)
- `FontManager* font_manager`:A betűtípuskezelő (Pointer)

(Nincs visszatérési értéke)

`void main_menu_init(MainMenu* main_menu, RenderingContext* context):`

Inicializálja a főmenüt, betölti a hanglejátszót, gombokat, zenét, textúrákat.

Paraméterek:

- `MainMenu* main_menu`:A főmenü (Pointer)
- `RenderingContext* context`:A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

`void check_main_menu_inputs(MainMenu* main_menu,GameState* game_state):`

Megnézi, hogy kattintottak-e a főmenü gombjaira.

Paraméterek:

- `MainMenu* main_menu`:A főmenü (Pointer)
- `GameState* game_state`:A játékállás, ahonnan olvasható az egér lenyomása (Pointer)

(Nincs visszatérési értéke)

```
void draw_main_menu(MainMenu* main_menu,RenderingContext* context,FontManager* font_manager):
```

Kirajzolja a főmenü gombjait, grafikus elemeit.

Paraméterek:

- MainMenu* main_menu:A főmenü (Pointer)
- RenderingContext* context:A rajzoló kontextus (Pointer)
- FontManager* font_manager:A betűtípuskezelő (Pointer)

(Nincs visszatérési értéke)

A toplista vizualizálásához szükséges struktúra. Tartalmaz egy vissza gombot, ami a főmenübe visz, egy felső szöveget, és egy szöveget, ami kiírja a TOP X helyezettet, pontszámait és a nehézséget ahol elérték azt:

TopListScreen

- FontSurface* self_label
- FontSurface* top_string
- Button* back_btn

```
void cleanup_menus(MainMenu* mainmenu,TopListScreen* top_screen):
```

Felzabadítja a toplista menüt és a főmenüt.

Paraméterek:

- MainMenu* mainmenu:A főmenü (Pointer)
- TopListScreen* top_screen:Referencia a toplista menüjére (Pointer)

(Nincs visszatérési értéke)

```
void init_top_list_screen(TopListScreen* top_list_screen,RenderingContext* context):
```

Inicializálja a top lista vizualizációs menüjét.

Paraméterek:

- TopListScreen* top_list_screen:Referencia a toplista menüjére (Pointer)
- RenderingContext* context:A rajzoló kontextus (Pointer)

(Nincs visszatérési értéke)

```
void draw_top_list_screen(TopListScreen* top_list_screen,GameState* game_state,RenderingContext* context,FontManager* font_manager):
```

Kirajzolja a toplistát

Paraméterek:

- TopListScreen* top_list_screen:Referencia a toplista menüjére (Pointer)
- GameState* game_state:A játékállás, ahonnan olvasható az egér lenyomása (Pointer)

- `RenderingContext* context:A rajzoló kontextus (Pointer)`
- `FontManager* font_manager:A betűtípuskezelő (Pointer)`

(Nincs visszatérési értéke)

A halálképernyő struktúrája, kiírja, hogy a játékos hány pontot ért el, és hogy bekerült-e a toplistára. Ezen felül itt választja ki a játékos a nevét, hogy ha bekerült a toplisztába:

DeathScreen

- `FontSurface* name_label //Ez rajzolja ki a nevet amivel a játékos akar szerepelni`
- `FontSurface* self_label`
- `Button* back_btn: Referencia a vissza gombra.`
- `SDL_Texture* toplist_label`
- `bool freed: Fel van-e szabadítva? (A dupla felszabadítás elkerülés miatti változó)`
- `bool ontoplist: Fent van-e a játékos a toplistán?`
- `bool done: A kész a játékos a neve megadásával, ez igaz lesz.`
- `double timer`

```
void init_death_screen(DeathScreen* death_screen,RenderingContext* context):
```

A halálképernyőt inicializálja.

Paraméterek:

- `DeathScreen* death_screen:A halálképernyő (Pointer)`
- `RenderingContext* context:A rajzoló kontextus (Pointer)`

(Nincs visszatérési értéke)

```
void update_death_button(DeathScreen* death_screen,GameState* game_state,RenderingContext* context,FontManager* font_manager):
```

Halálképernyő kirajzolásáért felelős függvény, függő a játékalapotha szerepelt értékektől és a paraméterként megadott betűtípust fogja használni szöveg rajzolásakor.

Paraméterek:

- `DeathScreen* death_screen:A halálképernyő (Pointer)`
- `GameState* game_state:A játékalapotha, ahonnan olvasható az egér lenyomása (Pointer)`
- `RenderingContext* context:A rajzoló kontextus (Pointer)`
- `FontManager* font_manager:A betűtípuskezelő (Pointer)`

(Nincs visszatérési értéke)

player.h

- A játékos struktúra, függvényei és az input függvényeit tartalmazó forrásfájl.

A játékos struktúra-úrája, eltárolja a játékos pozícióját, irányát, gyorsulását, sebességét, a lövés cooldown-t, azt, hogy halott-e, az életeket, a bónusz pontokat, a textúrát, a hanglejátszót és az invincibility frame-eket:

Player

- Vector2D pos
- double angle
- double velocity
- double accel
- bool moving
- double cooldown
- bool shooting
- bool is_dead
- double player_radius
- int lives
- int bonus_score
- double iframes
- SDL_Texture* sprite
- AudioManager player_sounds

void init_player(Player* player, RenderingContext* current_context):

Inicializálja a játékos, betölti a textúrát, a hanglejátszót, alap értékeket ad a játékosnak.

Paraméterek:

- Player* player: Játékos (Pointer)
- RenderingContext* current_context: A rajzoló kontextus (Pointer)

(Nincs visszatérési érték)

void cleanup_player(Player* player):

Felszabadítja a játékos által lefoglalt memóriát és a hanglejátszót.

Paraméterek:

- Player* player: Játékos (Pointer)

(Nincs visszatérési érték)

void spawn_player(Player* player):

Leidézi a játékos és ad neki alap értékeket. A játékos a pálya közepére idézi le.

Paraméterek:

- `Player* player:Játékos (Pointer)`

(Nincs visszatérési értéke)

`void setup_player(Player* player):`

A játékosnak ad 3 életet és nullázza a bónuszpontjait, hiszen ekkor indul a játék.
Paraméterek:

- `Player* player:Játékos (Pointer)`

(Nincs visszatérési értéke)

`void try_respawn_player(Player* player,GameState* game_state):`

Ha a játékosnak van bónusz élete akkor újraéleszti a játékost és egy extra életet elvesz, ha nincs akkor a halálképernyőt előhossa.
Paraméterek:

- `Player* player:Játékos (Pointer)`
- `GameState* game_state:A játékállás (Pointer)`

(Nincs visszatérési értéke)

`void update_player(Player* player,GameState* gamestate):`

Frissíti a játékos gyorsaságát, lövését, gyorsulását, irányát a bemenet alapján.
Paraméterek:

- `Player* player:Játékos (Pointer)`
- `GameState* gamestate:A játékállás (Pointer)`

(Nincs visszatérési értéke)

`void render_player(Player* player,RenderingContext* context):`

Kirajzolja a játékost.
Paraméterek:

- `Player* player:Játékos (Pointer)`
- `RenderingContext* context:A rajzoló kontextus (Pointer)`

(Nincs visszatérési értéke)

`void process_input(Player* player,GameState* game_state):`

Az inputok kezelése, egérmozgás, egérleenyomás, betűleenyomások kezelése, eltárolása.
Paraméterek:

- `Player* player:Játékos (Pointer)`
- `GameState* game_state:A játékállás (Pointer)`

(Nincs visszatérési értéke)

renderingmanager.h

- A kirajzolásért felelős függvények és struktúrák:

A rajzoló kontextus struktúrája, tartalmazza az egér pozícióját csak olvasásra, A játék ablakát és az SDL renderer-t:

RenderingContext

- SDL_Window* ablak
- SDL_Renderer* renderer
- Vector2D mouse_pos

SDL_Texture* load_sprite(RenderingContext* context, char path[]):

Betölt egy textúrát a kapott paraméteren lévő helyről (.bmp) és visszaadja azt.
Paraméterek:

- RenderingContext* context: *A rajzoló kontextus (Pointer)*
- char path[]: *A betöltendő kép elérési útvonala*

Kimenet: *Egy pointer ami a betöltött textúrára mutat*

bool ablak_init(RenderingContext* rendering_context):

Inicializálja az ablakot. Hiba esetén hamisat ad vissza, másképp igazat.
Paraméterek:

- RenderingContext* rendering_context: *A rajzoló kontextus (Pointer)*

Kimenet: *sikerült-e az ablakot hiba nélkül inicializálni?*

bool renderer_init(RenderingContext* rendering_context):

Inicializálja az SDL renderert. Hiba esetén hamisat ad vissza, másképp igazat.
Paraméterek:

- RenderingContext* rendering_context: *A rajzoló kontextus (Pointer)*

Kimenet: *sikerült-e az SDL renderert hiba nélkül inicializálni?*

void cleanup_context(RenderingContext* context):

Felszabadítja a rajzoló kontextust és elemeit.
Paraméterek:

- RenderingContext* context: *A rajzoló kontextus (Pointer)*

(Nincs visszatérési értéke)

scoremanager.h

- A pontszámok kirajzolásáért, beolvasásáért, elmentéséért felelő függvények és struktúrák vannak itt.

Egy pontszám rekord, eltárolja a játékos nevét, pontját és a nehézséget, valamint a következő láncolt lista elemre egy pointert:

ScoreEntry

- int cur_score
- int entry_difficulty
- char entry_name[13]: max név hossz: 12 + \0
- struct ScoreEntry* next_score (A lista következő eleme)

A legmagasabb pontszámkezeőt tartalmazza a pontszám rekordok láncolt listáját és a legkevesebb pontszámot olvasásra:

HighScoreManager

- int lowest: A toplistába a legkisebb érték
- struct ScoreEntry* next_score (A lista következő eleme)

**void load_top_list_screen(TopListScreen*
top_list_screen, HighScoreManager* high_score_manager):**

Betölti a top pontszámokat a toplista menübe, szöveggént egybe.

Paraméterek:

- TopListScreen* top_list_screen: Referencia a toplista menüjére (Pointer)
- HighScoreManager* high_score_manager: A pontszám kezelő (Pointer)

(Nincs visszatérési értéke)

void save_highscores(HighScoreManager* high_score_manager):

Elmenti a dicsőséglistát a toplista.txt file-ba.

Paraméterek:

- HighScoreManager* high_score_manager: A pontszám kezelő (Pointer)

(Nincs visszatérési értéke)

**void insert_to_highscores(HighScoreManager* high_score_manager,
ScoreEntry* insertion_value):**

Berakja az új pontrekordot a dicsőséglistába a megfelelő helyezéshez.

Paraméterek:

- HighScoreManager* high_score_manager: A pontszám kezelő (Pointer)

- `ScoreEntry* insertion_value`:A játékos elért pontszáma, neve és a nehézség, amin játszott (Pointer)

(Nincs visszatérési értéke)

`void load_highscores(HighScoreManager* high_score_manager):`

Betölti a dicsőséglistát a toplista.txt file-ból.

Paraméterek:

- `HighScoreManager* high_score_manager`:A dicsőséglista kezelő (Pointer)

(Nincs visszatérési értéke)

Ez a marker a szétlőtt aszteroidáért kapott pontot jelzi amíg az élettartama el nem fogy. Tartalmazza a pointert a láncolt listájának következő elemére:

ScoreMarker

- `double timeleft`
- `FontSurface* fontSurface`
- `struct ScoreMarker* next_marker` (A lista következő eleme)

A pontmenedzser mutatja, hogy a jelenlegi játékba hány pontot ért el a játékos és ad életeteket, ha elég pontot szerzett a játékos :

ScoreManager

- `Button* scorelabel`
- `char* base_string`
- `int given_lives`
- `int score`
- `ScoreMarker* next_marker` (A lista következő eleme)

`void cleanup_scores(HighScoreManager* hscoreman,ScoreManager* sman):`

Felszabadítja a dicsőséglistát ami be lett töltve és kiszedi a memóriából. A markereket pedig kitörli és felszabadítja.

Paraméterek:

- `HighScoreManager* hscoreman`:A dicsőséglista kezelő (Pointer)
- `ScoreManager* sman`:A pontszám kezelő (Pointer)

(Nincs visszatérési értéke)

`void cleanup_death_screen(DeathScreen* death_screen):`

A halálmenüt és elemeit, gombjait felszabadítja.

Paraméterek:

- `DeathScreen* death_screen`: *A halálképernyő (Pointer)*

(Nincs visszatérési értéke)

`void reset_score_manager(ScoreManager* score_manager):`

nullázza a pont számolót. Új játék kezdésekor hívódik meg.

Paraméterek:

- `ScoreManager* score_manager`: *A pontszám kezelő (Pointer)*

(Nincs visszatérési értéke)

`void setup_score_manager(ScoreManager* score_manager):`

Inicializálja a pontszámlálót alap értékekkel.

Paraméterek:

- `ScoreManager* score_manager`: *A pontszám kezelő (Pointer)*

(Nincs visszatérési értéke)

`void draw_score_manager(ScoreManager* score_manager, RenderingContext* context, FontManager* font_manager, Player* player):`

Kirajzolja a pontszámlálót és a markereit.

Paraméterek:

- `ScoreManager* score_manager`: *A pontszám kezelő (Pointer)*
- `RenderingContext* context`: *A rajzoló kontextus (Pointer)*
- `FontManager* font_manager`: *TODO (Pointer)*
- `Player* player`: *Játékos (Pointer)*

(Nincs visszatérési értéke)

`void append_score(ScoreManager* score_manager, Player* player):`

A pontszámláló hozzáadja magához a játékos által szerzett pontot.

Paraméterek:

- `ScoreManager* score_manager`: *A pontszám kezelő (Pointer)*
- `Player* player`: *Játékos (Pointer)*

(Nincs visszatérési értéke)

`void append_score_marker(ScoreManager* score_manager, int a_score, Vector2D pos):`

Hozzáad egy markert a pontszámlálóhoz, egy bizonyos helyre bizonyos értékkel.
Paraméterek:

- `ScoreManager* score_manager`:A pontszám kezelő (Pointer)
- `int a_score`:A pontszám, ami a markeren fog szerepelni
- `Vector2D pos`:A pozíció, ahol lesz a marker

(Nincs visszatérési értéke)

void update_score_manager(`ScoreManager* score_manager`,`GameState* game_state`):

A pontszámlálót frissíti, ad extra életet, ha kell és a markereket frissíti, vagy felszabadítja, ha kell.

Paraméterek:

- `ScoreManager* score_manager`:A pontszám kezelő (Pointer)
- `GameState* game_state`:A játékállás (Pointer)

(Nincs visszatérési értéke)

void setup_stats(`DeathScreen* death_screen`,`ScoreManager* score_manager`,`HighScoreManager* high_score_manager`):

Megváltoztatja a halálképernyőre kiírt adatokat az elért pontszám alapján.

Paraméterek:

- `DeathScreen* death_screen`:A halálképernyő (Pointer)
- `ScoreManager* score_manager`:A pontszám kezelő (Pointer)
- `HighScoreManager* high_score_manager`:A pontszám kezelő (Pointer)

(Nincs visszatérési értéke)

int register_name(`char* username`):

Beolvassa a nevet addig amíg nem éri el a név hossza a 12 karaktert vagy a játékos nem nyom enter. Ha entert nyom akkor felveszi a toplistába azzal a névvel.

Paraméterek:

- `char* username`:A játékos neve (Pointer)

Kimenet: (Igaz/Hamis) Kész van-e a függvény a beolvasással?

void save_score(`HighScoreManager* hman`,`char* player_name`,`ScoreManager* score_manager`,`GameState* gamestate`):

Elmenti a pontszámot.

Paraméterek:

- `HighScoreManager* hman`:A dicsőséglista kezelő (Pointer)
- `char* player_name`:A játékos, akinek a pontszámát el kell menteni (Pointer)

- `ScoreManager* score_manager:A pontszám kezelő (Pointer)`
- `GameState* gamestate:A játékállás (Pointer)`

(Nincs visszatérési értéke)
