Asteroids – Programozói dokumentáció

A játék grafikus részéért az SDL2 felel.

A játék indításakor a program leelenőrzi, hogy létezik e a „saves” mappa, ha nem akkor létrehozza. Ezután inicializálja az App struktúrát. Ebben tárolja el az legfontosabb információkat, gyakorlatilag minden más, függvényeknek átadott érték itt található.

Ezután a játék menüjét futtató függvény hívja meg: runMenu().

A menüben található a ranglista, a legutóbbi pontszám, a nehézség és a PLAY gomb. Ezekhez a szöveget a text_to_texture() függvény alakítja SDL_Texture* -é.

A játékot a T billentyű vagy a PLAY gomb kattintása indítja runGame() meghívásával.

A játékmenetért egy Player struktúra és két láncolt lista felel. Mindkét listában hasonló függvények felelnek a különböző műveleteket. A játékban két óra felel az időzítésekért. Az egyik minden framen kiszámolja az eltelt időt(deltaTime). A másik BASE_SPAWN_RATE-ig számol felfele majd alaphelyzetbe áll.

Az egyik listában a meteorok adatai vannak(Pozíció, méret, elforgatás). Meteorok egy már létező nem legkisebb meteor szétlövésekor vagy elegendő idő elteltével véletlen pozíción spawnolnak.

A másik listában a lövések adatai vannak(Pozíció, lövés szöge). Lőni bármelyik egérgomb lenyomásával lehet(SDL_MOUSEBUTTONDOWN event)

A játék függőleges szinkronizációval(VSync) kerüli el a végtelenített Képlocla/mp problémát a szokásos SDL_Delay() hívás helyett.

runGame() az adott játékban elért pontszámmal tér vissza miután a játékos visszatér a menübe(ablak bezárása/M – billentyű) vagy eltalálja egy meteor. Ekkor frissül a legutóbbi pontszám és a ranglista, valamint a játék alaphelyzetbe áll(resetGame())

Kilépéskor a legutóbbi pontszám és a ranglista txt fájlba íródik, az első egyszerű számként míg a másik *név pontszám* formátumban.

A menü bezárása után a program felszabadít minden dinamikusan foglalt területet majd meghívja SDL_Quit() -et végül visszatér 0-val.

Alább csatolom a teljes Doxygen által generált dokumentációt amelyben az összes függvény és struktúra valamint a legtöbb változó dokumentálva van:

Programozás alapjai 1. - NHF

Készítette Doxygen 1.9.8

1. fejezet

Adatszerkezet-mutató

1.1. Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

2. fejezet

Fájlmutató

2.1. Fájllista

Az összes fájl listája rövid leírásokkal:

3. fejezet

Adatszerkezetek dokumentációja

3.1. App struktúrareferencia

a játék legtöbb helyen használt változóit fogja össze

`#include <app.h>`

Adatmezők

- char username [51]

  játékos által megadott név
- SDL_Surface * icon
- SDL_Window * menuWindow
- SDL_Renderer * menuRenderer
- SDL_Window * gameWindow
- SDL_Renderer * gameRenderer
- SDL_Texture * reticle
- int screenW

  a képernyő szélessége
- int screenH

  a képernyő magassága
- TTF_Font * font

  a játékban használt normál méretű font
- TTF_Font * font_big

  a játékban használt nagy méretű font
- bool isMenu
- bool isGame
- Player player

  játékos adatait tartalmazó struktúra
- Input input

  a bemeneteket tároló struktúra
- SDL_Texture * backround

  háttér textúrája
- node * meteor_lista_head

  meteorokat tároló láncolt lista feje
- SDL_Texture * meteor_texture

meteorok textúrája

- int latest_score

  legutóbb elért pontszám
- shot_node * shot_lista_head

  lövéseket tároló lista feje
- SDL_Texture * shot_texture

  lövés textúrája
- ranglista_node * ranglista_head

  ranglista feje
- Clock clock

  a játék deltaTime számláló órája
- Spawn_clock spawn_clock

  a játékban meteorok létrehozásának időzítéséért felelős óra
- int targetFPS
- bool hardmode

  nehéz mód állása
- bool succesful_init

  sikeres inicializálás esetén true

3.1.1. Részletes leírás

a játék legtöbb helyen használt változóit fogja össze

Definíció a(z) app.h fájl 22. sorában.

3.1.2. Adatmezők dokumentációja

3.1.2.1. backround

```
SDL_Texture* backround
```

háttér textúrája

Definíció a(z) app.h fájl 54. sorában.

3.1.2.2. clock

```
Clock clock
```

a játék deltaTime számláló órája

Definíció a(z) app.h fájl 68. sorában.

3.1.2.3. font

```
TTF_Font* font
```

a játékban használt normál méretű font

Definíció a(z) app.h fájl 42. sorában.

3.1.2.4. font_big

```
TTF_Font* font_big
```

a játékban használt nagy méretű font

Definíció a(z) app.h fájl 44. sorában.

3.1.2.5. gameRenderer

```
SDL_Renderer* gameRenderer
```

Definíció a(z) app.h fájl 34. sorában.

3.1.2.6. gameWindow

```
SDL_Window* gameWindow
```

Definíció a(z) app.h fájl 32. sorában.

3.1.2.7. hardmode

```
bool hardmode
```

nehéz mód állása

Definíció a(z) app.h fájl 74. sorában.

3.1.2.8. icon

```
SDL_Surface* icon
```

Definíció a(z) app.h fájl 26. sorában.

3.1.2.9. input

```
Input input
```

a bemeneteket tároló struktúra

Definíció a(z) app.h fájl 52. sorában.

3.1.2.10. isGame

```
bool isGame
```

Definíció a(z) app.h fájl 48. sorában.

3.1.2.11. isMenu

```
bool isMenu
```

Definíció a(z) app.h fájl 46. sorában.

3.1.2.12. latest_score

```
int latest_score
```

legutóbb elért pontszám

Definíció a(z) app.h fájl 60. sorában.

3.1.2.13. menuRenderer

```
SDL_Renderer* menuRenderer
```

Definíció a(z) app.h fájl 30. sorában.

3.1.2.14. menuWindow

```
SDL_Window* menuWindow
```

Definíció a(z) app.h fájl 28. sorában.

3.1.2.15. meteor_lista_head

```
node* meteor_lista_head
```

meteorokat tároló láncolt lista feje

Definíció a(z) app.h fájl 56. sorában.

3.1.2.16. meteor_texture

```
SDL_Texture* meteor_texture
```

meteorok textúrája

Definíció a(z) app.h fájl 58. sorában.

3.1.2.17. player

```
Player player
```

játékos adatait tartalmazó struktúra

Definíció a(z) app.h fájl 50. sorában.

3.1.2.18. ranglista_head

```
ranglista_node* ranglista_head
```

ranglista feje

Definíció a(z) app.h fájl 66. sorában.

3.1.2.19. reticle

```
SDL_Texture* reticle
```

Definíció a(z) app.h fájl 36. sorában.

3.1.2.20. screenH

```
int screenH
```

a képernyő magassága

Definíció a(z) app.h fájl 40. sorában.

3.1.2.21. screenW

```
int screenW
```

a képernyő szélessége

Definíció a(z) app.h fájl 38. sorában.

3.1.2.22. shot_lista_head

```
shot_node* shot_lista_head
```

lövéseket tároló lista feje

Definíció a(z) app.h fájl 62. sorában.

3.1.2.23. shot_texture

```
SDL_Texture* shot_texture
```

lövés textúrája

Definíció a(z) app.h fájl 64. sorában.

3.1.2.24. spawn_clock

```
Spawn_clock spawn_clock
```

a játékban meteorok létrehozásának időzítéséért felelős óra

Definíció a(z) app.h fájl 70. sorában.

3.1.2.25. succesful_init

```
bool succesful_init
```

sikeres inicializálás esetén true

Definíció a(z) app.h fájl 76. sorában.

3.1.2.26. targetFPS

```
int targetFPS
```

Megjegyzés

nem használt

Definíció a(z) app.h fájl 72. sorában.

3.1.2.27. username

```
char username[51]
```

játékos által megadott név

Definíció a(z) app.h fájl 24. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/app.h

3.2. Clock struktúrareferencia

a játékban meteorok létrehozásának időzítéséért felelős óra

```
#include <clock.h>
```

Adatmezők

- int last_tick

  előző frame ezen a ticken kezdődött
- int now_tick

  a jelenlegi frame ezen a ticken kezdődött

3.2.1. Részletes leírás

a játékban meteorok létrehozásának időzítéséért felelős óra

Definíció a(z) clock.h fájl 6. sorában.

3.2.2. Adatmezők dokumentációja

3.2.2.1. last_tick

```
int last_tick
```

előző frame ezen a ticken kezdődött

Definíció a(z) clock.h fájl 8. sorában.

3.2.2.2. now_tick

```
int now_tick
```

a jelenlegi frame ezen a ticken kezdődött

Definíció a(z) clock.h fájl 10. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/clock.h

3.3. Input struktúrareferencia

a játékbeli bemeneteket tároló struktúra

```
#include <player.h>
```

Adatmezők

- int up
- int down
- int left
- int right
- int menu

3.3.1. Részletes leírás

a játékbeli bemeneteket tároló struktúra

Definíció a(z) player.h fájl 27. sorában.

3.3.2. Adatmezők dokumentációja

3.3.2.1. down

```
int down
```

Definíció a(z) player.h fájl 28. sorában.

3.3.2.2. left

```
int left
```

Definíció a(z) player.h fájl 28. sorában.

3.3.2.3. menu

```
int menu
```

Definíció a(z) player.h fájl 28. sorában.

3.3.2.4. right

```
int right
```

Definíció a(z) player.h fájl 28. sorában.

3.3.2.5. up

```
int up
```

Definíció a(z) player.h fájl 28. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/player.h

3.4. Meteor struktúrareferencia

egy meteor tulajdonságait tartalmazza

```
#include <aszteroida.h>
```

Adatmezők

- SDL_FRect position

  meteor koordinátáit és méreteit tárolja
- int meret

  meteor mérete: 0-2
- float angle

  beteor középpont körüli elforgatásának szöge(random)

3.4.1. Részletes leírás

egy meteor tulajdonságait tartalmazza

Definíció a(z) aszteroida.h fájl 14. sorában.

3.4.2. Adatmezők dokumentációja

3.4.2.1. angle

```
float angle
```

beteor középpont körüli elforgatásának szöge(random)

Definíció a(z) aszteroida.h fájl 20. sorában.

3.4.2.2. meret

```
int meret
```

meteor mérete: 0-2

Definíció a(z) aszteroida.h fájl 18. sorában.

3.4.2.3. position

```
SDL_FRect position
```

meteor koordinátáit és méreteit tárolja

Definíció a(z) aszteroida.h fájl 16. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/aszteroida.h

3.5. node struktúrareferencia

meteorok láncolt listájában egy elem

`#include <aszteroida.h>`

Adatmezők

- Meteor meteor
  adott listaelemben tárolt meteor adatai
- struct node * next
  következő listaelem

3.5.1. Részletes leírás

meteorok láncolt listájában egy elem

Definíció a(z) aszteroida.h fájl 24. sorában.

3.5.2. Adatmezők dokumentációja

3.5.2.1. meteor

`Meteor meteor`

adott listaelemben tárolt meteor adatai

Definíció a(z) aszteroida.h fájl 26. sorában.

3.5.2.2. next

`struct node* next`

következő listaelem

Definíció a(z) aszteroida.h fájl 28. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/aszteroida.h

3.6. Player struktúrareferencia

jtáékos adatait tároló struktúra

`#include <player.h>`

Adatmezők

- SDL_FRect position
  játékos mérete és koordinátái
- SDL_Texture * texture
  játékos textúrája
- int health
  játékos életereje

3.6.1. Részletes leírás

jtáékos adatait tároló struktúra

Definíció a(z) player.h fájl 16. sorában.

3.6.2. Adatmezők dokumentációja

3.6.2.1. health

```
int health
```

játékos életereje

Megjegyzés

currently partly unused

Definíció a(z) player.h fájl 23. sorában.

3.6.2.2. position

```
SDL_FRect position
```

játékos mérete és koordinátái

Definíció a(z) player.h fájl 18. sorában.

3.6.2.3. texture

```
SDL_Texture* texture
```

játékos textúrája

Definíció a(z) player.h fájl 20. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/player.h

3.7. ranglista_node struktúrareferencia

ranglista láncolt listájában egy elem

`#include <ranglista.h>`

Adatmezők

- rekord adat
  adott elem rekordját tartalmazza
- struct ranglista_node * next
  következő listaelem

3.7.1. Részletes leírás

ranglista láncolt listájában egy elem

Definíció a(z) ranglista.h fájl 22. sorában.

3.7.2. Adatmezők dokumentációja

3.7.2.1. adat

`rekord adat`

adott elem rekordját tartalmazza

Definíció a(z) ranglista.h fájl 24. sorában.

3.7.2.2. next

`struct ranglista_node* next`

következő listaelem

Definíció a(z) ranglista.h fájl 26. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/ranglista.h

3.8. rekord struktúrareferencia

egy eredmény(rekord) adatai a ranglistán

`#include <ranglista.h>`

Adatmezők

- char nev [51]
  játékos neve(50 karakter + '\0')
- int pontszam
  elért pontszám

3.8.1. Részletes leírás

egy eredmény(rekord) adatai a ranglistán

Definíció a(z) ranglista.h fájl 14. sorában.

3.8.2. Adatmezők dokumentációja

3.8.2.1. nev

```
char nev[51]
```

játékos neve(50 karakter + '\0')

Definíció a(z) ranglista.h fájl 16. sorában.

3.8.2.2. pontszam

```
int pontszam
```

elért pontszám

Definíció a(z) ranglista.h fájl 18. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/ranglista.h

3.9. Shot struktúrareferencia

egy lövés adatait tárolja

```
#include <shoot.h>
```

Adatmezők

- double angle
  a lövés vízszinteshez képest vett szöge
- SDL_FRect position
  a lövés mérete és koordinátái

3.9.1. Részletes leírás

egy lövés adatait tárolja

Definíció a(z) shoot.h fájl 14. sorában.

3.9.2. Adatmezők dokumentációja

3.9.2.1. angle

```
double angle
```

a lövés vízszinteshez képest vett szöge

Definíció a(z) shoot.h fájl 16. sorában.

3.9.2.2. position

```
SDL_FRect position
```

a lövés mérete és koordinátái

Definíció a(z) shoot.h fájl 18. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/shoot.h

3.10. shot_node struktúrareferencia

a lövések listájának egy eleme

```
#include <shoot.h>
```

Adatmezők

- Shot shot
  lövés adatait tárolja
- struct shot_node * next
  következő listaelem

3.10.1. Részletes leírás

a lövések listájának egy eleme

Definíció a(z) shoot.h fájl 22. sorában.

3.10.2. Adatmezők dokumentációja

3.10.2.1. next

```
struct shot_node* next
```

következő listaelem

Definíció a(z) shoot.h fájl 26. sorában.

3.10.2.2. shot

```
Shot shot
```

lövés adatait tárolja

Definíció a(z) shoot.h fájl 24. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/shoot.h

3.11. Spawn_clock struktúrareferencia

a játékban meteorok létrehozásának időzítéséért felelős óra

```
#include <clock.h>
```

Adatmezők

- double time
  játék kezdete óta eltelt idő
- double lastSpawn
  legutóbbi meteor spawn óta eltelt idő

3.11.1. Részletes leírás

a játékban meteorok létrehozásának időzítéséért felelős óra

Definíció a(z) clock.h fájl 14. sorában.

3.11.2. Adatmezők dokumentációja

3.11.2.1. lastSpawn

```
double lastSpawn
```

legutóbbi meteor spawn óta eltelt idő

Definíció a(z) clock.h fájl 18. sorában.

3.11.2.2. time

```
double time
```

játék kezdete óta eltelt idő

Definíció a(z) clock.h fájl 16. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- src/clock.h

4. fejezet

Fájlok dokumentációja

4.1. src/app.h fájlreferencia

```
#include <stdio.h>
#include <stdbool.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_ttf.h>
#include "aszteroida.h"
#include "player.h"
#include "shoot.h"
#include "ranglista.h"
#include "defines.h"
#include "font.h"
#include "../lib/debugmalloc.h"
#include "menu.h"
#include "file.h"
#include "clock.h"
#include "reticle.h"
```

Adatszerkezetek

- struct App

  a játék legtöbb helyen használt változóit fogja össze

Típusdefiníciók

- typedef struct App App

  a játék legtöbb helyen használt változóit fogja össze

Függvények

- App init_App (int screenW, int screenH)

  Inicializálja a játékadatok struktúráját, létrehozza a rendereket, ablakokat, megnyitja a textúrákat és a fontot. Inicializálja a láncolt listák fejét és a legutóbbi pontszámot 0-ra állítja stb.
- void runMenu (App ∗app)

  Futtatja a menüt. Meghívja runGame() -t, ha a játékos a játékot indító jelet adja.
- int runGame (App ∗app)

  A játék futtatásáért felelős. ha a játékos bezárja, visszatérés után a menüben folytatódik a játék.
- void resetGame (App ∗app)

  alaphelyzetbe állítja a játékot: kitörli a meteorokat, lövéseket, visszateszi a játékost a kezdőpozícióra
- void getDisplaySize (int ∗w, int ∗h)

  lekérdezi majd letárolja a képernyő méretét

4.1.1. Típusdefiníciók dokumentációja

4.1.1.1. App

```
typedef struct App App
```

a játék legtöbb helyen használt változóit fogja össze

4.1.2. Függvények dokumentációja

4.1.2.1. getDisplaySize()

```
void getDisplaySize (
            int * w,
            int * h )
```

lekérdezi majd letárolja a képernyő méretét

Paraméterek

w	a képernyő szélességét tároló változóra pointer
h	a képernyő magasságát tároló változóra pointer

4.1.2.2. init_App()

```
App init_App (
            int screenW,
            int screenH )
```

Inicializálja a játékadatok struktúráját, létrehozza a rendereket, ablakokat, megnyitja a textúrákat és a fontot. Inicializálja a láncolt listák fejét és a legutóbbi pontszámot 0-ra állítja stb.

Paraméterek

screenW	a képernyő szélessége
screenH	a képernyő magassága

Visszatérési érték

App visszatér az inicializált struktúrával

4.1.2.3. resetGame()

```
void resetGame (
            App * app )
```

alaphelyzetbe állítja a játékot: kitörli a meteorokat, lövéseket, visszateszi a játékost a kezdőpozícióra

Paraméterek

app	a játék adatait tartalmazó struktúrára mutató pointer

4.1.2.4. runGame()

```
int runGame (
            App * app )
```

A játék futtatásáért felelős. ha a játékos bezárja, visszatérés után a menüben folytatódik a játék.

Paraméterek

app	a játék adatait tartalmazó struktúrára mutató pointer

Visszatérési érték

visszatér a játékos adott sessionben elért pontszámával

4.1.2.5. runMenu()

```
void runMenu (
            App * app )
```

Futtatja a menüt. Meghívja runGame() -t, ha a játékos a játékot indító jelet adja.

Paraméterek

app	a játék adatait tartalmazó struktúrára mutató pointer

4.2. app.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef APP_H
00002 #define APP_H
00003
00004 #include <stdio.h>
00005 #include <stdbool.h>
00006 #include <SDL.h>
00007 #include <SDL_image.h>
00008 #include <SDL_ttf.h>
00009 #include "aszteroida.h"
00010 #include "player.h"
00011 #include "shoot.h"
00012 #include "ranglista.h"
00013 #include "defines.h"
00014 #include "font.h"
00015 #include "../lib/debugmalloc.h"
00016 #include "menu.h"
00017 #include "file.h"
00018 #include "clock.h"
00019 #include "reticle.h"
00020
00022 typedef struct App{
00024     char username[51];
00025
00026     SDL_Surface* icon;
00027
00028     SDL_Window *menuWindow;
00029
00030     SDL_Renderer *menuRenderer;
00031
00032     SDL_Window *gameWindow;
00033
00034     SDL_Renderer *gameRenderer;
00035
00036     SDL_Texture* reticle;
00038     int screenW;
00040     int screenH;
00042     TTF_Font* font;
00044     TTF_Font* font_big;
00045
00046     bool isMenu;
00047
00048     bool isGame;
00050     Player player;
00052     Input input;
00054     SDL_Texture *backround;
00056     node* meteor_lista_head;
00058     SDL_Texture* meteor_texture;
00060     int latest_score;
00062     shot_node* shot_lista_head;
00064     SDL_Texture* shot_texture;
00066     ranglista_node* ranglista_head;
00068     Clock clock;
00070     Spawn_clock spawn_clock;
00072     int targetFPS;
00074     bool hardmode;
00076     bool succesful_init;
00077 }App;
00078
00088 App init_App(int screenW , int screenH);
00089
00095 void runMenu(App* app);
00096
00104 int runGame(App* app);
00105
00108 void resetGame(App* app);
00109
00113 void getDisplaySize(int* w , int* h);
00114
00115 #endif
```

4.3. src/aszteroida.h fájlreferencia

```
#include <SDL.h>
#include <SDL_image.h>
#include <stdbool.h>
```

```
#include <math.h>
#include <stdio.h>
#include "defines.h"
#include "../lib/debugmalloc.h"
```

Adatszerkezetek

- struct Meteor

  egy meteor tulajdonságait tartalmazza
- struct node

  meteorok láncolt listájában egy elem

Típusdefiníciók

- typedef struct Meteor Meteor

  egy meteor tulajdonságait tartalmazza
- typedef struct node node

  meteorok láncolt listájában egy elem

Függvények

- node ∗ spawnMeteors (struct node ∗head, int maxX, int maxY)

  Meteorokat hoz létre.
- node ∗ spawnMeteors_pos (struct node ∗head, int x, int y, int meret)

  Meteorokat hoz létre megadott pozición és megadott mérettel.
- int renderMeteors (struct node ∗head, SDL_Renderer ∗renderer, SDL_Texture ∗texture)

  A rendererre másolja a láncolt listában lévő meteorokat.
- int moveMeteors (node ∗head, float deltaTime)

  mozgatja jobbról balra a meteorokat
- void delete_out_of_bounds_meteors (node ∗∗head)

  kitörli a pályán kívülre kerülő meteorokat
- void delete_meteor_list (node ∗head)

  kitörli az egész meteorlistát

4.3.1. Típusdefiníciók dokumentációja

4.3.1.1. Meteor

```
typedef struct Meteor Meteor
```

egy meteor tulajdonságait tartalmazza

4.3.1.2. node

```
typedef struct node node
```

meteorok láncolt listájában egy elem

4.3.2. Függvények dokumentációja

4.3.2.1. delete_meteor_list()

```
void delete_meteor_list (
             node * head )
```

kitörli az egész meteorlistát

Paraméterek

head	láncolt lista első elemére mutató pointer

4.3.2.2. delete_out_of_bounds_meteors()

```
void delete_out_of_bounds_meteors (
            node ** head )
```

kitörli a pályán kívülre kerülő meteorokat

Paraméterek

head	meteorok láncolt listájának fejére mutató pointer pointere (kell, mivel head is törlésre kerülhet)

4.3.2.3. moveMeteors()

```
int moveMeteors (
            node * head,
            float deltaTime )
```

mozgatja jobbról balra a meteorokat

Paraméterek

head	meteorok láncolt listájának feje
deltaTime	deltaTime, eddig tartott a frame (s = v∗t mozgatás mértékének számításához kell)

Visszatérési érték

int -1 ha head==NULL

4.3.2.4. renderMeteors()

```
int renderMeteors (
            struct node * head,
            SDL_Renderer * renderer,
            SDL_Texture * texture )
```

A rendererre másolja a láncolt listában lévő meteorokat.

Paraméterek

head	a meteorokat tartalmazó láncolt lista első elemére mutató pointer
renderer	a játék SDL_Renderer -jére mutató pointer
texture	a meteorokhoz használt textúrára mutató pointer

Visszatérési érték

int -1 ha head==NULL

4.3.2.5. spawnMeteors()

```
node * spawnMeteors (
            struct node * head,
            int maxX,
            int maxY )
```

Meteorokat hoz létre.

Paraméterek

head	a meteorokat tartalmazó láncolt lista első elemére mutató pointer
maxX	maximum X pozíció
maxY	maximum Y pozíció

Visszatérési érték

node∗ az új elemre mutató pointer (új első elem lesz)

4.3.2.6. spawnMeteors_pos()

```
node * spawnMeteors_pos (
            struct node * head,
            int x,
            int y,
            int meret )
```

Meteorokat hoz létre megadott pozíción és megadott mérettel.

Paraméterek

head	a meteorokat tartalmazó láncolt lista első elemére mutató pointer
x	X pozíció
y	Y pozíció
meret	az újmeteor mérete

Visszatérési érték

node∗ az új elemre mutató pointer (új első elem lesz)

4.4. aszteroida.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef ASZTEROIDA_H
00002 #define ASZTEROIDA_H
00003
00004 #include <SDL.h>
00005 #include <SDL_image.h>
00006 #include <stdbool.h>
00007 #include <math.h>
00008 #include <stdio.h>
00009 #include "defines.h"
00010 #include "../lib/debugmalloc.h"
00011
00012
00014 typedef struct Meteor{
00016     SDL_FRect position;
00018     int meret;
00020     float angle;
00021 }Meteor;
00022
00024 typedef struct node{
00026     Meteor meteor;
00028     struct node *next;
00029 }node;
00030
00039 node* spawnMeteors(struct node* head, int maxX , int maxY);
00040
00050 node* spawnMeteors_pos(struct node* head , int x , int y , int meret);
00051
00060 int renderMeteors(struct node* head , SDL_Renderer* renderer , SDL_Texture* texture);
00061
00069 int moveMeteors(node* head , float deltaTime);
00070
00076 void delete_out_of_bounds_meteors(node** head);
00077
00083 void delete_meteor_list(node* head);
00084
00085 #endif
```

4.5. src/clock.h fájlreferencia

`#include <SDL.h>`

Adatszerkezetek

- struct Clock

  a játékban meteorok létrehozásának időzítéséért felelős óra
- struct Spawn_clock

  a játékban meteorok létrehozásának időzítéséért felelős óra

Típusdefiníciók

- typedef struct Clock Clock

  a játékban meteorok létrehozásának időzítéséért felelős óra
- typedef struct Spawn_clock Spawn_clock

  a játékban meteorok létrehozásának időzítéséért felelős óra

Függvények

- void init_clock (Clock ∗clock)
  inicializálja az órát
- void tick_clock (Clock ∗clock)
  lépteti az órát
- float calculate_delta_time (Clock ∗clock)
  kiszámolja az előző frame óta eltelt időt
- void init_spawn_clock (Spawn_clock ∗clock)
  inicializálja az órát(meteor spawnhoz)
- void tick_spawn_clock (Spawn_clock ∗clock)
  lépteti az órát (de csak time növekszik)
- float calculate_spawn_time (Spawn_clock ∗clock)
  kiszámolja a legutóbbi meteor spawn óta eltelt időt
- void resetSpawnClock (Spawn_clock ∗clock)
  meteor spawnoláskor beállítja lastSpawn-t a jelenlegi időre

4.5.1. Típusdefiníciók dokumentációja

4.5.1.1. Clock

```
typedef struct Clock Clock
```

a játékban meteorok létrehozásának időzítéséért felelős óra

4.5.1.2. Spawn_clock

```
typedef struct Spawn_clock Spawn_clock
```

a játékban meteorok létrehozásának időzítéséért felelős óra

4.5.2. Függvények dokumentációja

4.5.2.1. calculate_delta_time()

```
float calculate_delta_time (
            Clock * clock )
```

kiszámolja az előző frame óta eltelt időt

Paraméterek

clock	óra pointere

Visszatérési érték

deltaTime másodpercben

4.5.2.2. calculate_spawn_time()

```
float calculate_spawn_time (
            Spawn_clock * clock )
```

kiszámolja a legutóbbi meteor spawn óta eltelt időt

Paraméterek

clock	Spawn_clock (amiből számol)

Visszatérési érték

legutóbbi spawn óta eltelt idő másodpercben

4.5.2.3. init_clock()

```
void init_clock (
            Clock * clock )
```

inicializálja az órát

Paraméterek

clock	inicializálandó óra pointere

4.5.2.4. init_spawn_clock()

```
void init_spawn_clock (
            Spawn_clock * clock )
```

inicializálja az órát(meteor spawnhoz)

Paraméterek

clock	inicializálandó óra pointere

4.5.2.5. resetSpawnClock()

```
void resetSpawnClock (
            Spawn_clock * clock )
```

meteor spawnoláskor beállítja lastSpawn-t a jelenlegi időre

Paraméterek

clock	állítandó óra pointere

4.5.2.6. tick_clock()

```
void tick_clock (
            Clock * clock )
```

lépteti az órát

Paraméterek

clock	léptetett óra pointere

4.5.2.7. tick_spawn_clock()

```
void tick_spawn_clock (
            Spawn_clock * clock )
```

lépteti az órát (de csak time növekszik)

Paraméterek

clock	léptetendő óra pointere

4.6. clock.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CLOCK_H
00002 #define CLOCK_H
00003 #include <SDL.h>
00004
00006 typedef struct Clock{
00008     int last_tick;
00010     int now_tick;
00011 }Clock;
00012
00014 typedef struct Spawn_clock{
00016     double time;
00018     double lastSpawn;
00019 }Spawn_clock;
00020
00023 void init_clock(Clock* clock);
00024
00027 void tick_clock(Clock* clock);
00028
00032 float calculate_delta_time(Clock* clock);
00033
00036 void init_spawn_clock(Spawn_clock* clock);
00037
00040 void tick_spawn_clock(Spawn_clock* clock);
00041
00045 float calculate_spawn_time(Spawn_clock* clock);
00046
00049 void resetSpawnClock(Spawn_clock* clock);
00050
00051 #endif
```

4.7. src/defines.h fájlreferencia

Makródefiníciók

- #define PI 3.14159265358

- #define BASE_SPAWN_RATE 2

  ennyi másodpercenként spawnol egy meteor
- #define SHOT_TIME 10

  két lövés közt legalább ennyi időnek kell eltelnie
- #define DIE 1

  ha 0 akkor a játékos nem halhat meg
- #define SHOT_SPEED 2100

  a lövések sebessége
- #define MENU_W 800

  a menü ablakának szélessége
- #define MENU_H 400

  a menü ablakának magassága
- #define RANGLISTA_SIZE 7

  ennyi rekord jelenik meg a ranglistáról a menüben
- #define REKORD_SIZE 59

  ilyen hosszú maximum egy rekord stringje a ranglistán
- #define MENU_COLOR 150, 222, 255, 0

  a menü háttérszíne
- #define METEOR_SPEED 270

  meteorok sebessége
- #define PLAYER_SPEED 500

  a játékos sebessége
- #define PLAYER_SIZE 64

  a játékos szélessége és magassága
- #define DEFINITELY_AN_ERROR_MESSAGE system("..\\materials\\Error.url");

  What could this be.

4.7.1. Makródefiníciók dokumentációja

4.7.1.1. BASE_SPAWN_RATE

```
#define BASE_SPAWN_RATE 2
```

ennyi másodpercenként spawnol egy meteor

Definíció a(z) defines.h fájl 9. sorában.

4.7.1.2. DEFINITELY_AN_ERROR_MESSAGE

```
#define DEFINITELY_AN_ERROR_MESSAGE system("..\\materials\\Error.url");
```

What could this be.

Definíció a(z) defines.h fájl 43. sorában.

4.7.1.3. DIE

```
#define DIE 1
```

ha 0 akkor a játékos nem halhat meg

Megjegyzés

for debug purposes

Definíció a(z) defines.h fájl 14. sorában.

4.7.1.4. MENU_COLOR

```
#define MENU_COLOR 150, 222, 255, 0
```

a menü háttérszíne

Definíció a(z) defines.h fájl 30. sorában.

4.7.1.5. MENU_H

```
#define MENU_H 400
```

a menü ablakának magassága

Definíció a(z) defines.h fájl 21. sorában.

4.7.1.6. MENU_W

```
#define MENU_W 800
```

a menü ablakának szélessége

Definíció a(z) defines.h fájl 19. sorában.

4.7.1.7. METEOR_SPEED

```
#define METEOR_SPEED 270
```

meteorok sebessége

Definíció a(z) defines.h fájl 33. sorában.

4.7.1.8. PI

```
#define PI 3.14159265358
```

Definíció a(z) defines.h fájl 6. sorában.

4.7.1.9. PLAYER_SIZE

```
#define PLAYER_SIZE 64
```

a játékos szélessége és magassága

Megjegyzés

a legjobb eredményhez előnyös 2 hatványára állítani

Definíció a(z) defines.h fájl 40. sorában.

4.7.1.10. PLAYER_SPEED

```
#define PLAYER_SPEED 500
```

a játékos sebessége

Definíció a(z) defines.h fájl 36. sorában.

4.7.1.11. RANGLISTA_SIZE

```
#define RANGLISTA_SIZE 7
```

ennyi rekord jelenik meg a ranglistáról a menüben

Megjegyzés

8 fölé nem érdemes vinni mert belelóg a többi menürészbe

Definíció a(z) defines.h fájl 25. sorában.

4.7.1.12. REKORD_SIZE

```
#define REKORD_SIZE 59
```

ilyen hosszú maximum egy rekord stringje a ranglistán

Definíció a(z) defines.h fájl 27. sorában.

4.7.1.13. SHOT_SPEED

```
#define SHOT_SPEED 2100
```

a lövések sebessége

Definíció a(z) defines.h fájl 16. sorában.

4.7.1.14. SHOT_TIME

```
#define SHOT_TIME 10
```

két lövés közt legalább ennyi időnek kell eltelnie

Definíció a(z) defines.h fájl 11. sorában.

4.8. defines.h

Ugrás a fájl dokumentációjához.

```
00001
00002
00003 #ifndef DEFINES_H
00004 #define DEFINES_H
00005
00006 #define PI 3.14159265358
00007
00009 #define BASE_SPAWN_RATE 2
00011 #define SHOT_TIME 10
00014 #define DIE 1
00016 #define SHOT_SPEED 2100
00017
00019 #define MENU_W 800
00021 #define MENU_H 400
00022
00025 #define RANGLISTA_SIZE 7
00027 #define REKORD_SIZE 59 //50+1+7+1 50= max nev hossza , 1 = szokoz , 7 = INT_MAX karakterszama , 1 =
      '\0'
00028
00030 #define MENU_COLOR 150, 222, 255, 0
00031
00033 #define METEOR_SPEED 270
00034
00036 #define PLAYER_SPEED 500
00037
00040 #define PLAYER_SIZE 64
00041
00043 #define DEFINITELY_AN_ERROR_MESSAGE system("..\\materials\\Error.url");
00044
00045 #endif
```

4.9. src/file.h fájlreferencia

```
#include <stdio.h>
#include <dirent.h>
#include "ranglista.h"
```

Függvények

- int load_latest_score (char *path)

  betölti a legutóbbi sessionben elért pontszámot
- int write_latest_score (char *path, int latest_score)

  kiírja a legutóbbi sessionben elért pontszámot fájlba
- void check_for_saves_folder (void)

  leelenőrzi, hogy létezik -e "saves" mappa és ha nem akkor létrehozza

4.9.1. Függvények dokumentációja

4.9.1.1. check_for_saves_folder()

```
void check_for_saves_folder (
            void  )
```

leelenőrzi, hogy létezik -e "saves" mappa és ha nem akkor létrehozza

Megjegyzés

(mkdir "../saves"), macOS-en lehetséges hogy nem fut

4.9.1.2. load_latest_score()

```
int load_latest_score (
            char * path )
```

betölti a legutóbbi sessionben elért pontszámot

Paraméterek

path	fájl elérési útja

Visszatérési érték

int legutóbbi sessionben elért pontszám

4.9.1.3. write_latest_score()

```
int write_latest_score (
            char * path,
            int latest_score )
```

kiírja a legutóbbi sessionben elért pontszámot fájlba

Paraméterek

path	fájl elérési útja
latest_score	elért pontszám

Visszatérési érték

int | -1 ha nem sikerült megnyitni a fájlt, 0 ha sikeres a kiírás.

4.10. file.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef FILE_H
00002 #define FILE_H
00003 #include <stdio.h>
00004 #include <dirent.h>
00005 #include "ranglista.h"
00006
00013 int load_latest_score(char* path);
00014
00022 int write_latest_score(char* path, int latest_score);
00023
00026 void check_for_saves_folder(void);
00027 #endif
```

4.11. src/font.h fájlreferencia

```
#include <SDL.h>
#include <SDL_ttf.h>
```

Függvények

- SDL_Texture * text_to_texture (TTF_Font *font, char *text, SDL_Renderer *renderer, SDL_Rect *pos)
  szöveget textúrává alakít
- SDL_Texture * text_to_texture_white (TTF_Font *font, char *text, SDL_Renderer *renderer, SDL_Rect *pos)
  szöveget textúrává alakít fehér kijelzéshez

4.11.1. Függvények dokumentációja

4.11.1.1. text_to_texture()

```
SDL_Texture * text_to_texture (
            TTF_Font * font,
            char * text,
            SDL_Renderer * renderer,
            SDL_Rect * pos )
```

szöveget textúrává alakít

Paraméterek

font	font amivel a szöveget szeretnénk kiírni
text	szöveg amit kiírunk
renderer	renderer amire kiírjuk a szöveget
pos	SDL_Rect* pozíció ahová a szöveget kiírjuk

Visszatérési érték

SDL_Texture* a konvertálás végeredménye

4.11.1.2. text_to_texture_white()

```
SDL_Texture * text_to_texture_white (
              TTF_Font * font,
              char * text,
              SDL_Renderer * renderer,
              SDL_Rect * pos )
```

szöveget textúrává alakít fehér kijelzéshez

Paraméterek

font	font amivel a szöveget szeretnénk kiírni
text	szöveg amit kiírunk
renderer	renderer amire kiírjuk a szöveget
pos	SDL_Rect* pozíció ahová a szöveget kiírjuk

Visszatérési érték

SDL_Texture* a konvertálás végeredménye

4.12. font.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef FONT_H
00002 #define FONT_H
00003
00004 #include <SDL.h>
00005 #include <SDL_ttf.h>
00006
00016 SDL_Texture* text_to_texture(TTF_Font* font, char* text, SDL_Renderer* renderer, SDL_Rect* pos);
00026 SDL_Texture* text_to_texture_white(TTF_Font* font, char* text, SDL_Renderer* renderer, SDL_Rect* pos);
00027
00028 #endif
```

4.13. src/menu.h fájlreferencia

```
#include <SDL.h>
#include <SDL_ttf.h>
#include "font.h"
#include "defines.h"
#include <stdio.h>
#include <stdbool.h>
```

Függvények

- int renderCopyMenuContents (SDL_Renderer ∗renderer, TTF_Font ∗font, char ∗username, int latestpoint)
  rendereli a menü tartalmát amihez nem kell különösen sok adat(pl. latest score)
- void render_get_username (SDL_Renderer ∗renderer, TTF_Font ∗font, char ∗username)
  rendereli a játékos számára a felhasználónév bekérés üzenetet
- int renderPlayButton (SDL_Renderer ∗renderer, TTF_Font ∗font)
  rendereli a play gombot
- SDL_Rect calculatePlayButtonSize (SDL_Renderer ∗renderer, TTF_Font ∗font)
  kiszámolja a play gomb méretét(kattintás ellenőrzéshez szükséges)
- bool checkPlayButton (SDL_Rect ∗play_pos)
  leelenőrzi hogy le lett -e nyomva a play gomb
- int renderHardButton (SDL_Renderer ∗renderer, TTF_Font ∗font, bool ∗hardmode)
  rendererre másolja a nehézséget jelző és állító szöveget
- SDL_Rect calculateHardButtonSize (SDL_Renderer ∗renderer, TTF_Font ∗font, bool ∗hardmode)
  kiszámolja a play gomb méretét(kattintás ellenőrzéshez szükséges)
- bool checkHardButton (SDL_Rect ∗hard_pos)
  leelenőrzi le lett -e nyomva a gomb

4.13.1. Függvények dokumentációja

4.13.1.1. calculateHardButtonSize()

```
SDL_Rect calculateHardButtonSize (
            SDL_Renderer * renderer,
            TTF_Font * font,
            bool * hardmode )
```

kiszámolja a play gomb méretét(kattintás ellenőrzéshez szükséges)

Paraméterek

renderer	menü rendererje
font	használt font
hardmode	be van e kapcsolva a nehéz játékmód

Visszatérési érték

SDL_Rect a gombot leíró téglatest

4.13.1.2. calculatePlayButtonSize()

```
SDL_Rect calculatePlayButtonSize (
            SDL_Renderer * renderer,
            TTF_Font * font )
```

kiszámolja a play gomb méretét(kattintás ellenőrzéshez szükséges)

Paraméterek

renderer	menü rendererje
font	használt font

Visszatérési érték

SDL_Rect a gombot leíró téglatest

4.13.1.3. checkHardButton()

```
bool checkHardButton (
            SDL_Rect * hard_pos )
```

leelenőrzi le lett -e nyomva a gomb

Paraméterek

hard_pos	gomb pozíciója

Visszatérési érték

igaz ha le lett nyomva, hamsi ha nem

4.13.1.4. checkPlayButton()

```
bool checkPlayButton (
            SDL_Rect * play_pos )
```

leelenőrzi hogy le lett -e nyomva a play gomb

Paraméterek

play_pos	a gombot leíró téglatest

Visszatérési érték

ture ha lenyomták, flase ha nem

4.13.1.5. render_get_username()

```
void render_get_username (
            SDL_Renderer * renderer,
            TTF_Font * font,
            char * username )
```

rendereli a játékos számára a felhasználónév bekérés üzenetet

Paraméterek

renderer	menü rendererje
font	használt font
username	a játékos által megadott nevet tároló char tömb

4.13.1.6. renderCopyMenuContents()

```
int renderCopyMenuContents (
            SDL_Renderer * renderer,
            TTF_Font * font,
            char * username,
            int latestpoint )
```

rendereli a menü tartalmát amihez nem kell különösen sok adat(pl. latest score)

Paraméterek

renderer	menü rendererje
font	használt font
username	a játékos által megadott név
latestpoint	legutóbb elért pontszám

Visszatérési érték

0 ha sikeres a konvertálás | -1 ha nem

4.13.1.7. renderHardButton()

```
int renderHardButton (
            SDL_Renderer * renderer,
            TTF_Font * font,
            bool * hardmode )
```

rendererre másolja a nehézséget jelző és állító szöveget

Paraméterek

renderer	menü rendererje
font	használt font
hardmode	be van e kapcsolva a nehéz játékmód

Visszatérési érték

0 ha sikeres

4.13.1.8. renderPlayButton()

```
int renderPlayButton (
            SDL_Renderer * renderer,
            TTF_Font * font )
```

rendereli a play gombot

Paraméterek

renderer	menü rendererje
font	használt font

Visszatérési érték

0 ha sikeres

4.14. menu.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef _MENU_H
00002 #define _MENU_H
00003
00004 #include <SDL.h>
00005 #include <SDL_ttf.h>
00006 #include "font.h"
00007 #include "defines.h"
00008 #include <stdio.h>
00009 #include <stdbool.h>
00010
00017 int renderCopyMenuContents(SDL_Renderer* renderer , TTF_Font* font , char* username , int
      latestpoint);
00018
00023 void render_get_username(SDL_Renderer* renderer , TTF_Font* font , char* username);
00024
00029 int renderPlayButton(SDL_Renderer* renderer , TTF_Font* font);
00030
00035 SDL_Rect calculatePlayButtonSize(SDL_Renderer* renderer , TTF_Font* font);
00036
00040 bool checkPlayButton(SDL_Rect* play_pos);
00041
00047 int renderHardButton(SDL_Renderer* renderer , TTF_Font* font , bool* hardmode);
00048
00054 SDL_Rect calculateHardButtonSize(SDL_Renderer* renderer , TTF_Font* font , bool* hardmode);
00055
00059 bool checkHardButton(SDL_Rect* hard_pos);
00060
00061 #endif
```

4.15. src/player.h fájlreferencia

```
#include <SDL.h>
#include <SDL_image.h>
#include <stdbool.h>
#include <stdio.h>
#include "aszteroida.h"
#include "../lib/debugmalloc.h"
#include "defines.h"
```

Adatszerkezetek

- struct Player

  jtáékos adatait tároló struktúra
- struct Input

  a játékbeli bemeneteket tároló struktúra

Függvények

- void init_player (int x, int y, int health, SDL_Renderer ∗renderer, char ∗path, Player ∗player)

  Inicializálja a játékost.
- void move_player (Player ∗player, Input input, float deltaTime, int maxX, int maxY)

  Játékos mozgatásáért felelős függvény.
- void keyDown (Input ∗input, SDL_KeyboardEvent ∗event)

  Kezeli a billentyűk lenyomását.
- void keyUp (Input ∗input, SDL_KeyboardEvent ∗event)

  Kezeli a billentyűk felengedését.
- void reset_input (Input ∗input)

  alaphelyzetbe állítja a bemenetet tároló struktúrát
- Meteor utkozes_ellenorzese (struct node ∗∗head, Player ∗player)

  ellenőrzi hogy a játékost eltalálja -e egy aszteroida

4.15.1. Függvények dokumentációja

4.15.1.1. init_player()

```
void init_player (
            int x,
            int y,
            int health,
            SDL_Renderer * renderer,
            char * path,
            Player * player )
```

Inicializálja a játékost.

Paraméterek

x	a játékos kezdő x koordinátája
y	a játékos kezdő y koordinátája
health	a játékos kezdő életereje
renderer	a játék SDL_Renderer -jére mutató pointer
path	a játékos textúrájának elérési útja(string)

Visszatérési érték

Player | visszatér az inicializált játékossal

4.15.1.2. keyDown()

```
void keyDown (
             Input * input,
             SDL_KeyboardEvent * event )
```

Kezeli a billentyűk lenyomását.

Paraméterek

input	bemeneteket kezelő structra pointer
event	keyboardevent, ez alapján kapjuk meg milyen gomb(ok) lettek lenyomva

4.15.1.3. keyUp()

```
void keyUp (
             Input * input,
             SDL_KeyboardEvent * event )
```

Kezeli a billentyűk felengedését.

Paraméterek

input	bemeneteket kezelő structra pointer
event	keyboardevent, ez alapján kapjuk meg milyen gomb(ok) lettek felengedve

4.15.1.4. move_player()

```
void move_player (
             Player * player,
             Input input,
             float deltaTime,
             int maxX,
             int maxY )
```

Játékos mozgatásáért felelős függvény.

Paraméterek

player	inicializált, mozgatandó játékosra mutató pointer
input	bemeneteket tároló struct

4.15.1.5. reset_input()

```
void reset_input (
             Input * input )
```

alaphelyzetbe állítja a bemenetet tároló struktúrát

Paraméterek

input	bemenetet tároló struktúrára pointer

4.15.1.6. utkozes_ellenorzese()

```
Meteor utkozes_ellenorzese (
            struct node ** head,
            Player * player )
```

ellenőrzi hogy a játékost eltalálja -e egy aszteroida

Paraméterek

head	aszteroida lista fejére mutató pointer pointere
player	játékos adatait tároló struktúra pointere

Visszatérési érték

Meteor visszatér a meteorral ami eltalálta a játékost

Megjegyzés

jelenleg sok felesleges dolog van ebben a függvényben de később tervem van velük

4.16. player.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef PLAYER_H
00002 #define PLAYER_H
00003
00004 #include <SDL.h>
00005 #include <SDL_image.h>
00006 #include <stdbool.h>
00007 #include <stdio.h>
00008 #include "aszteroida.h"
00009 #include "../lib/debugmalloc.h"
00010 #include "defines.h"
00011
00012 //előre deklaráció(valószínűleg eltávolítható, will fix)
00013 struct node;
00014
00016 typedef struct{
00018     SDL_FRect position;
00020     SDL_Texture *texture;
00023     int health;
00024 }Player;
00025
00027 typedef struct{
00028     int up , down , left , right , menu;
00029 }Input;
00030
00041 void init_player(int x , int y, int health , SDL_Renderer *renderer , char* path , Player* player);
00042
00049 void move_player(Player* player , Input input , float deltaTime , int maxX , int maxY);
00056 void keyDown(Input* input , SDL_KeyboardEvent* event);
00057
00064 void keyUp(Input* input , SDL_KeyboardEvent* event);
00065
00071 void reset_input(Input* input);
00072
00081 Meteor utkozes_ellenorzese(struct node** head , Player *player);
00082
00083 #endif
```

4.17. src/ranglista.h fájlreferencia

```
#include "stdlib.h"
#include "string.h"
#include "stdio.h"
#include "defines.h"
#include "font.h"
#include <SDL_ttf.h>
#include <SDL.h>
#include "../lib/debugmalloc.h"
```

Adatszerkezetek

- struct rekord

  egy eredmény(rekord) adatai a ranglistán
- struct ranglista_node

  ranglista láncolt listájában egy elem

Típusdefiníciók

- typedef struct rekord rekord

  egy eredmény(rekord) adatai a ranglistán
- typedef struct ranglista_node ranglista_node

  ranglista láncolt listájában egy elem

Függvények

- void insert_ranking (ranglista_node ∗∗head, char ∗nev, int pont)

  beilleszt egy eredményt a listába, megtartva annak csökkenő sorrendjét
- ranglista_node ∗ read_ranglista_from_file ()

  beolvassa a ranglistát fileból
- int print_ranglista_to_file (ranglista_node ∗head)

  kiírja a ranglistát fájlba
- int renderRanglista (SDL_Renderer ∗renderer, TTF_Font ∗font, ranglista_node ∗head)

  rendereli a listát
- void delete_ranglista (ranglista_node ∗head)

  felszabadítja az egész ranglistát

4.17.1. Típusdefiníciók dokumentációja

4.17.1.1. ranglista_node

```
typedef struct ranglista_node ranglista_node
```

ranglista láncolt listájában egy elem

4.17.1.2. rekord

```
typedef struct rekord rekord
```

egy eredmény(rekord) adatai a ranglistán

4.17.2. Függvények dokumentációja

4.17.2.1. delete_ranglista()

```
void delete_ranglista (
            ranglista_node * head )
```

felszabadítja az egész ranglistát

Paraméterek

head	lista legelső elemére pointer

4.17.2.2. insert_ranking()

```
void insert_ranking (
            ranglista_node ** head,
            char * nev,
            int pont )
```

beilleszt egy eredményt a listába, megtartva annak csökkenő sorrendjét

Paraméterek

head	ranglista fej pointer pointere
nev	játékos neve
pont	elért pontszám

4.17.2.3. print_ranglista_to_file()

```
int print_ranglista_to_file (
            ranglista_node * head )
```

kiírja a ranglistát fájlba

Paraméterek

head	lista legelső elemére pointer

Visszatérési érték

-1 ha a kiírás sikertelen, 0 ha sikeres

4.17.2.4. read_ranglista_from_file()

```
ranglista_node * read_ranglista_from_file ( )
```

beolvassa a ranglistát fileból

Visszatérési érték

a lista legelső elemére pointer (az egész lista dinamikusan foglalt, felsazabdítására van külön függvény)

4.17.2.5. renderRanglista()

```
int renderRanglista (
            SDL_Renderer * renderer,
            TTF_Font * font,
            ranglista_node * head )
```

rendereli a listát

Paraméterek

renderer	használt renderer
font	használt font
head	lista legelső elemére pointer

Visszatérési érték

-1 ha head==NULL, 0 ha sikeres

4.18. ranglista.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef RANGLISTA_H
00002 #define RANGLISTA_H
00003
00004 #include "stdlib.h"
00005 #include "string.h"
00006 #include "stdio.h"
00007 #include "defines.h"
00008 #include "font.h"
00009 #include <SDL_ttf.h>
00010 #include <SDL.h>
00011 #include "../lib/debugmalloc.h"
00012
00014 typedef struct rekord{
00016     char nev[51];
00018     int pontszam;
00019 }rekord;
00020
00022 typedef struct ranglista_node{
00024     rekord adat;
00026     struct ranglista_node* next;
```

```
00027 }ranglista_node;
00028
00033 void insert_ranking(ranglista_node** head , char* nev , int pont);
00034
00037 ranglista_node* read_ranglista_from_file();
00038
00042 int print_ranglista_to_file(ranglista_node* head);
00043
00049 int renderRanglista(SDL_Renderer* renderer, TTF_Font* font , ranglista_node* head);
00050
00053 void delete_ranglista(ranglista_node* head);
00054
00055 #endif
```

4.19. src/reticle.h fájlreferencia

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

Függvények

- void renderReticle (SDL_Renderer ∗renderer, SDL_Texture ∗texture)

  rendereli a célkeresztet a játékban

4.19.1. Függvények dokumentációja

4.19.1.1. renderReticle()

```
void renderReticle (
            SDL_Renderer * renderer,
            SDL_Texture * texture )
```

rendereli a célkeresztet a játékban

Paraméterek

renderer	játék rendererje
texture	a célkereszt textúrája

4.20. reticle.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef RETICLE_H
00002 #define RETICLE_H
00003
00004 #include <SDL.h>
00005
00009 void renderReticle(SDL_Renderer* renderer, SDL_Texture* texture);
00010
00011 #endif
```

4.21. src/shoot.h fájlreferencia

```
#include <SDL.h>
#include <SDL_image.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "aszteroida.h"
#include "../lib/debugmalloc.h"
#include "defines.h"
```

Adatszerkezetek

- struct Shot

  egy lövés adatait tárolja
- struct shot_node

  a lövések listájának egy eleme

Típusdefiníciók

- typedef struct shot_node shot_node

  a lövések listájának egy eleme

Függvények

- double calculate_angle_for_shot (int shipX, int shipY)

  kiszámolja egy kattintás és a játékos közti vektor hajlásszögét
- int render_shots (shot_node ∗head, SDL_Renderer ∗renderer, SDL_Texture ∗texture)

  lövések másolása rendererre
- struct shot_node ∗ add_new_shot (struct shot_node ∗head, double angle, int shipX, int shipY)

  hozzáad egy lövést a lista elejéhez
- Meteor check_hits (shot_node ∗∗head, node ∗∗meteor_head)

  ellenőrzi hogy a játékos eltalált -e egy meteort a lövéseivel
- int move_shots (struct shot_node ∗head, float deltaTime)

  lövések mozgatása
- int delete_shot_list (struct shot_node ∗head)

  felszabadítja a lövések listáját

4.21.1. Típusdefiníciók dokumentációja

4.21.1.1. shot_node

```
typedef struct shot_node shot_node
```

a lövések listájának egy eleme

4.21.2. Függvények dokumentációja

4.21.2.1. add_new_shot()

```
struct shot_node * add_new_shot (
            struct shot_node * head,
            double angle,
            int shipX,
            int shipY )
```

hozzáad egy lövést a lista elejéhez

Paraméterek

head	lövések listájának head-jére mutató pointer
angle	lövés szöge
shipX	játékos X pozíció
shipY	játékos Y pozíció

Visszatérési érték

struct shot_node* visszatér a hozzáadott lövés pointerjével(dinamikusan foglalt, az egész lista felszabadítására van függvény)

4.21.2.2. calculate_angle_for_shot()

```
double calculate_angle_for_shot (
            int shipX,
            int shipY )
```

kiszámolja egy kattintás és a játékos közti vektor hajlásszögét

Paraméterek

shipX	játékos X pozíció
shipY	játékos Y pozíció

Visszatérési érték

double hajlászög radiánban x: [-pi/2 ; 3pi/2]

4.21.2.3. check_hits()

```
Meteor check_hits (
            shot_node ** head,
            node ** meteor_head )
```

ellenőrzi hogy a játékos eltalált -e egy meteort a lövéseivel

Paraméterek

head	lövések listájának head-jére mutató pointer pointere
meteor_head	meteorok listájának head-jére mutató pointer pointere

Visszatérési érték

Meteor visszatér az eltalált meteor adataival(meret=-1 ha nem volt találat, -2 ha valamelyik head NULL)

4.21.2.4. delete_shot_list()

```
int delete_shot_list (
            struct shot_node * head )
```

felszabadítja a lövések listáját

Paraméterek

head	lövések listájának head-jére pointer

Visszatérési érték

int: -1 ha head==NULL | 0 ha sikeres

4.21.2.5. move_shots()

```
int move_shots (
            struct shot_node * head,
            float deltaTime )
```

lövések mozgatása

Paraméterek

head	lövések listájának head-jére mutató pointer

Visszatérési érték

int: -1 ha head==NULL | 0 ha sikeres

4.21.2.6. render_shots()

```
int render_shots (
            shot_node * head,
            SDL_Renderer * renderer,
            SDL_Texture * texture )
```

lövések másolása rendererre

Paraméterek

head	lövések listájának head-jére mutató pointer
renderer	gameRenderer
texture	lövések textúrája

Visszatérési érték

int: 0 ha sikeres | -1 ha sikertelen

4.22. shoot.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef SHOOT_H
00002 #define SHOOT_H
00003
00004 #include <SDL.h>
00005 #include <SDL_image.h>
00006 #include <math.h>
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include "aszteroida.h"
00010 #include "../lib/debugmalloc.h"
00011 #include "defines.h"
00012
00014 typedef struct{
00016     double angle;
00018     SDL_FRect position;
00019 }Shot;
00020
00022 typedef struct shot_node{
00024     Shot shot;
00026     struct shot_node* next;
00027 }shot_node;
00028
00036 double calculate_angle_for_shot(int shipX , int shipY);
00037
00046 int render_shots(shot_node* head, SDL_Renderer* renderer , SDL_Texture* texture);
00047
00057 struct shot_node* add_new_shot(struct shot_node* head , double angle, int shipX , int shipY);
00058
00066 Meteor check_hits(shot_node** head, node** meteor_head);
00067
00074 int move_shots(struct shot_node* head , float deltaTime);
00075
00082 int delete_shot_list(struct shot_node* head);
00083
00084 #endif
```

Tárgymutató