snakeGame

Készítette Doxygen 1.9.2

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. ButtonBox struktúrareferencia

Egy menü egy grafikai elemének adatait tartalmazó struct.

`#include <menus.h>`

Adatmezők

- int value
- int colorR
- int colorG
- int colorB
- int posX1
- int posY1
- int posX2
- int posY2
- char * text
- int textColorR
- int textColorG
- int textColorB

3.1.1. Részletes leírás

Egy menü egy grafikai elemének adatait tartalmazó struct.

3.1.2. Adatmezők dokumentációja

3.1.2.1. colorB

`int ButtonBox::colorB`

A téglalap RGB kék színe

3.1.2.2. colorG

```
int ButtonBox::colorG
```

A téglalap RGB zöld színe

3.1.2.3. colorR

```
int ButtonBox::colorR
```

A téglalap RGB piros színe

3.1.2.4. posX1

```
int ButtonBox::posX1
```

A téglalap bal alsó sarkának x koordinátája

3.1.2.5. posX2

```
int ButtonBox::posX2
```

A téglalap jobb felső sarkának x koordinátája

3.1.2.6. posY1

```
int ButtonBox::posY1
```

A téglalap bal alsó sarkának y koordinátája

3.1.2.7. posY2

```
int ButtonBox::posY2
```

A téglalap jobb felső sarkának y koordinátája

3.1.2.8. text

```
char* ButtonBox::text
```

A téglalapban megjelenítendő szöveg

3.1.2.9. textColorB

```
int ButtonBox::textColorB
```

A szöveg RGB kék színe

3.1.2.10. textColorG

```
int ButtonBox::textColorG
```

A szöveg RGB zöld színe

3.1.2.11. textColorR

```
int ButtonBox::textColorR
```

A szöveg RGB piros színe

3.1.2.12. value

```
int ButtonBox::value
```

A checkClick() funkció visszatérési értéke. Ezáltal lehet tovább vezérelni a programot.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- menus.h

3.2. fruit struktúrareferencia

Egy gyümölcs adatait tartalmazó struct.

```
#include <graphics.h>
```

Adatmezők

- int x
- int y
- SDL_Color color
- struct fruit * nextFruit

3.2.1. Részletes leírás

Egy gyümölcs adatait tartalmazó struct.

3.2.2. Adatmezők dokumentációja

3.2.2.1. color

```
SDL_Color fruit::color
```

A gyümölcs színe

3.2.2.2. nextFruit

```
struct fruit* fruit::nextFruit
```

A következő gyümölcs helye a memóriában.

3.2.2.3. x

```
int fruit::x
```

A gyümölcs pozíciója x koordináta szerint.

3.2.2.4. y

```
int fruit::y
```

A gyümölcs pozíciója y koordináta szerint.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- graphics.h

3.3. global_Settings struktúrareferencia

A játék fő beállításait("főflagek") tartalmazó struct.

```
#include <gameLogic.h>
```

Adatmezők

- bool twoPlayerMode
- bool show_mainMenu
- bool init_mainMenu
- bool show_gameSettings
- bool show_mainGame
- bool show_highScoreboard
- bool game_Init
- bool exitGame
- bool init_highScoreboard
- bool isRunning

3.3.1. Részletes leírás

A játék fő beállításait("főflagek") tartalmazó struct.

3.3.2. Adatmezők dokumentációja

3.3.2.1. exitGame

```
bool global_Settings::exitGame
```

Játékot bezáró folyamat elindítása

3.3.2.2. game_Init

```
bool global_Settings::game_Init
```

Játék inicializálása

3.3.2.3. init_highScoreboard

```
bool global_Settings::init_highScoreboard
```

Dicsőségtábla előrenderelése

3.3.2.4. init_mainMenu

```
bool global_Settings::init_mainMenu
```

Főmenü renderelésének megkezdése

3.3.2.5. isRunning

```
bool global_Settings::isRunning
```

Fut-e a program

3.3.2.6. show_gameSettings

```
bool global_Settings::show_gameSettings
```

Játék indítása előtti beállítások mutatása

3.3.2.7. show_highScoreboard

```
bool global_Settings::show_highScoreboard
```

Dicsőségtábla mutatása, vele együtt dicsőségtábla vezérlőinek engedélyezése

3.3.2.8. show_mainGame

```
bool global_Settings::show_mainGame
```

Játék mutatása, vele együtt játék vezérlőinek engedélyezése

3.3.2.9. show_mainMenu

```
bool global_Settings::show_mainMenu
```

Főmenü mutatása, hozzátartozó vezérlő bekapcsolása

3.3.2.10. twoPlayerMode

```
bool global_Settings::twoPlayerMode
```

többjátékos mód kapcsoló

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- gameLogic.h

3.4. highScorePlayer struktúrareferencia

A dicsőségtábla egy elemének az adatai.

```
#include <io.h>
```

Adatmezők

- char name [50]
- int score

3.4.1. Részletes leírás

A dicsőségtábla egy elemének az adatai.

3.4.2. Adatmezők dokumentációja

3.4.2.1. name

```
char highScorePlayer::name[50]
```

a játékos neve. Maximum 50 karakter hosszú.

3.4.2.2. score

```
int highScorePlayer::score
```

A játékos eredménye.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- io.h

3.5. scoreBoard_highscores struktúrareferencia

A dicsőségtáblát tartalmazó struct.

```
#include <io.h>
```

Adatmezők

- highScorePlayer data [10]

3.5.1. Részletes leírás

A dicsőségtáblát tartalmazó struct.

3.5.2. Adatmezők dokumentációja

3.5.2.1. data

```
highScorePlayer scoreBoard_highscores::data[10]
```

A játékosok adatait highScorePlayer structokban tartalmazó tömbje.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- io.h

3.6. scoreBoard_highscores_Elements struktúrareferencia

A dicsőségtábla elemeit tartalmazú struct.

`#include <menus.h>`

Adatmezők

- ButtonBox menuElements [11]

3.6.1. Részletes leírás

A dicsőségtábla elemeit tartalmazú struct.

3.6.2. Adatmezők dokumentációja

3.6.2.1. menuElements

```
ButtonBox scoreBoard_highscores_Elements::menuElements[11]
```

11 hosszú tömb, benne a 10 legjobb játékos adataival és a visszalépés (X) gombbal

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- menus.h

3.7. Snake struktúrareferencia

A kígyó, és fejének adatait tartalmazó struct.

`#include <graphics.h>`

Adatmezők

- int x
- int y
- double vx
- double vy
- unsigned int r
- unsigned int g
- unsigned int b
- int points
- struct SnakeBody * firstBodyElement
- char lastPos

3.7.1. Részletes leírás

A kígyó, és fejének adatait tartalmazó struct.

3.7.2. Adatmezők dokumentációja

3.7.2.1. b

```
unsigned int Snake::b
```

A kígyó RGB bontásban vett kék színe.

3.7.2.2. firstBodyElement

```
struct SnakeBody* Snake::firstBodyElement
```

Az első kígyótestre mutató pointer.

3.7.2.3. g

```
unsigned int Snake::g
```

A kígyó RGB bontásban vett zöld színe.

3.7.2.4. lastPos

```
char Snake::lastPos
```

A legutolsó irány, amerre a kígyó ment. Kezdőértéke 0. U: up, D: down, L: left, R: right. Ez akadályozza meg, hogy a kígyó belefordduljon a saját testébe.

3.7.2.5. points

```
int Snake::points
```

A kígyó pontszáma

3.7.2.6. r

```
unsigned int Snake::r
```

A kígyó színének RGB bontásban vett piros színe.

3.7.2.7. vx

```
double Snake::vx
```

A kígyó fejének x irányú sebessége

3.7.2.8. vy

```
double Snake::vy
```

A kígyó fejének y irányú sebessége

3.7.2.9. x

```
int Snake::x
```

A kígyó fejének pozíciója x koordináta szerint.

3.7.2.10. y

```
int Snake::y
```

A kígyó fejének pozíciója y koordináta szerint.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- graphics.h

3.8. SnakeBody struktúrareferencia

A kígyó testét tartalmazó struct. A fej adatait nem tartalmazza, azt a Snake struct tárolja. Több SnakeBody struct együtt egy duplán láncolt listát alkot.

```
#include <graphics.h>
```

Adatmezők

- int x
- int y
- struct SnakeBody * next
- struct SnakeBody * prev

3.8.1. Részletes leírás

A kígyó testét tartalmazó struct. A fej adatait nem tartalmazza, azt a Snake struct tárolja. Több SnakeBody struct együtt egy duplán láncolt listát alkot.

3.8.2. Adatmezők dokumentációja

3.8.2.1. next

```
struct SnakeBody* SnakeBody::next
```

A következő SnakeBody helye a memóriában.

3.8.2.2. prev

```
struct SnakeBody* SnakeBody::prev
```

Az előző SnakeBody helye a memóriában

3.8.2.3. x

```
int SnakeBody::x
```

A kígyótest egy elemének helyzete x koordináta szerint.

3.8.2.4. y

```
int SnakeBody::y
```

A kígyótest egy elemének helyzete y koordináta szerint.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- graphics.h

3.9. SnakeBodyList struktúrareferencia

A kígyó testének strázsája, és sentinelje.

```
#include <graphics.h>
```

Adatmezők

- SnakeBody * head
- SnakeBody * last

3.9.1. Részletes leírás

A kígyó testének strázsája, és sentinelje.

3.9.2. Adatmezők dokumentációja

3.9.2.1. head

```
SnakeBody* SnakeBodyList::head
```

Strázsa helye a memóriában. A strázsa HASZNOS adatot tárol! Ide másolódik a kígyó fejének x és y koordinátája.

3.9.2.2. last

```
SnakeBody* SnakeBodyList::last
```

Sentinel helye a memőriában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- graphics.h

3.10. Window struktúrareferencia

A megjelenítő ablak adatai.

```
#include <graphics.h>
```

Adatmezők

- int width
- int height

3.10.1. Részletes leírás

A megjelenítő ablak adatai.

3.10.2. Adatmezők dokumentációja

3.10.2.1. height

```
int Window::height
```

Az ablak magassága.

3.10.2.2. width

```
int Window::width
```

Az ablak szélessége.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- graphics.h

4. fejezet

Fájlok dokumentációja

4.1. gameLogic.c fájlreferencia

A játék közbeni vezérlés, és a menük előkészítéséért, és rendereléséért felelős modul.

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

Függvények

- void stopGame (global_Settings *g)

  Előkészíti a kilépést.
- void mainMenu_init (global_Settings *g)

  A főmenü előkészítése, majd kirenderelése.
- void mainMenuLogic (global_Settings *g)

  A menü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.
- void render_gameSettingsMenu (global_Settings *g)

  A játék indítása előtti almenü előkészítése, majd kirenderelése.
- void gameSettingsLogic (global_Settings *g, Snake *snake1, Snake *snake2)

  A játék indítása előtti almenü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.
- void render_highScoresMenu (global_Settings *g, scoreBoard_highscores_Elements m)

  A dicsőségtábla kiírását megvalósító függvény.
- void highScoresMenu_Logic (global_Settings *g, scoreBoard_highscores_Elements highScoreMenu)

  A dicsőségtábla almenü vezérlő függvénye egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.
- void inGameButtons (ButtonBox *buttons, int len)

  A játék futása alatt a "Menü" gomb megjelenítése.
- void mainGame_Logic (global_Settings *g, Snake *snake1, Snake *snake2, scoreBoard_highscores *hS)

  A játék futása alatti logikai motor.
- void randomize_snakePos (Snake *s)

  A Kígyó pozíciójának véletlenszerű elhelyezése.
- void resetSnake (Snake *s)

  A Kígyó pozíciójának és utolsó irány karakterének alaphelyzetbe állítása.

- void resetSnakePoints (Snake ∗s1)

  A Kígyó pontszámának visszaállítása.
- void P1_Controller (Snake ∗snake1, SDL_Event ev)

  Az első számú kígyó vezérléséért felelős függvény.
- void P2_Controller (Snake ∗snake2, SDL_Event ev)

  A második számú kígyó vezérléséért felelős függvény.
- int checkScore (Snake ∗s, scoreBoard_highscores hS)

  Megnézi, hogy a játékos által elért eredmény benne van-e a top 10-ben.
- void changeHighScoreList (Snake ∗s, const char ∗playerName, int idx, scoreBoard_highscores ∗h)
- bool enter_text (char ∗where, size_t len, SDL_Rect box, SDL_Color backgroundColor, SDL_Color textColor)
- fruit ∗ add_Fruit (fruit ∗firstFruit, SnakeBodyList snake1_L, SnakeBodyList snake2_L)

  Generál egy új gyümölcsöt, beteszi a gyümölcsöket tartalmazó lista elejére.
- void destroyFruitList (fruit ∗fruitList)

  Felszabadítja a gyümölcsöket tartalmazó láncolt listát.
- fruit ∗ checkCollision (fruit ∗fruitList, Snake s)

  Megnézi, hogy egy kígó ütközik-e egy gymölcssel.
- fruit ∗ deleteFruit (fruit ∗fruitList, fruit ∗toBeDeleted)

  Kitöröl egy gyümölcsöt a láncolt listából.
- bool checkIncomingFruitCollision (fruit newFruit, SnakeBodyList ∗s)

  Megnézi, hogy a frissen beilleszteni kívánt gyümölcs ütközne-e egy játékossal.
- void add_BodyElement (SnakeBodyList ∗o, Snake s)

  Megnöveli a kígyó hosszát 1-gyel.
- void moveBody (SnakeBodyList ∗s, Snake ∗sHead)

  Elmozdítja a kígyó testét.
- void init_SnakeBody (SnakeBodyList ∗sBody)

  Előkészíti a kígyó testének duplán láncolt listáját.
- bool checkWallHit (Snake s)

  Megnézi, hogy a Snake kígyófej ütközött e éppen a fallal.
- void exitProgram (global_Settings ∗g, FILE ∗fp, scoreBoard_highscores ∗highscores)

  Bezárja az SDL könyvtárakat és kiírja az új dicsőségtáblát a fájlba,.
- void destroy_snakeBody (SnakeBodyList ∗s)

  Felszabadítja a kígyó testéhez lefoglalt duplán láncolt listát.
- bool checkBodyCollision (SnakeBodyList ∗sList, Snake ∗sHead)

  Megnézi, hogy két test (a.k.a. az egyik kígyó feje és a másik teste) ütközik-e.
- bool checkHeadCollision (Snake ∗sHead1, Snake ∗sHead2)

  A két fej ütközését vizsgálja meg.
- void calculateNewScoreboard (scoreBoard_highscores ∗hS, Snake snake1, Snake snake2)

  Az új dicsőségtáblát elkészítő modul.
- void highscore_subRoutine (int snakeIndex, scoreBoard_highscores ∗hS, Snake playerSnake, short which↩ Snake)

  Dicsőségtábla szubrutin.

Változók

- global_Settings globalSettings

4.1.1. Részletes leírás

A játék közbeni vezérlés, és a menük előkészítéséért, és rendereléséért felelős modul.

4.1.2. Függvények dokumentációja

4.1.2.1. add_BodyElement()

```
void add_BodyElement (
            SnakeBodyList * o,
            Snake s )
```

Megnöveli a kígyó hosszát 1-gyel.

Paraméterek

o	A kígyó testét tartalmazó SnakeBodyList struct
s	A kígyó fejét tartalmazú struct.

4.1.2.2. add_Fruit()

```
fruit * add_Fruit (
            fruit * firstFruit,
            SnakeBodyList snake1_L,
            SnakeBodyList snake2_L )
```

Generál egy új gyümölcsöt, beteszi a gyümölcsöket tartalmazó lista elejére.

Ha ez a gyümölcs ütközne az egyik kígyóval, akkor újat generál-

Paraméterek

firstFruit	A gyümölcslista első elemére mutató pointer.
snake1_L	Az 1. kígyó testét tartalmazó struct.
snake2_L	A 2. kígyó testét tartalmazó struct.

Visszatérési érték

Visszatér a frissített gyümölcslista elejére mutató pointerre.

4.1.2.3. calculateNewScoreboard()

```
void calculateNewScoreboard (
            scoreBoard_highscores * hS,
```

```
            Snake snake1,
            Snake snake2 )
```

Az új dicsőségtáblát elkészítő modul.

Megkeresi, hogy van-e új rekord, majd az eredmény alapján továbblép, vagy meghívja a hisghscore_subRoutine() függvényt, hogy módosítsa a dicsőségtáblát.

Paraméterek

hS	A program futása során tárolt diőcségtábla.
snake1	Az egyik kígyó feje.
snake2	A másik kígyó feje

4.1.2.4. changeHighScoreList()

```
void changeHighScoreList (
            Snake * s,
            const char * playerName,
            int idx,
            scoreBoard_highscores * h )
```

4.1.2.5. checkBodyCollision()

```
bool checkBodyCollision (
            SnakeBodyList * sList,
            Snake * sHead )
```

Megnézi, hogy két test (a.k.a. az egyik kígyó feje és a másik teste) ütközik-e.

Paraméterek

sHead	Az egyik kígyó feje.
sList	A másik kígyó teste

Visszatérési érték

Ha ütközik a fej a testtel, TRUE-t ad vissza, egyébként FALSE-t.

4.1.2.6. checkCollision()

```
fruit * checkCollision (
            fruit * fruitList,
            Snake s )
```

Megnézi, hogy egy kígó ütközik-e egy gymölcssel.

Paraméterek

fruitList	A gyümölcslista első elemére mutató pointer.
s	a vizsgáladnó kígyó feje.

Visszatérési érték

Ha ütközik a kígyó valamelyik gyümölccsel, akkor a függvény visszter az adott gyümölcsre mutató fruit∗ pointerrel. Egyéb esetben NULLpointert ad vissza.

4.1.2.7. checkHeadCollision()

```
bool checkHeadCollision (
            Snake * sHead1,
            Snake * sHead2 )
```

A két fej ütközését vizsgálja meg.

Paraméterek

sHead1	Az egyik kígyó feje.
sHead2	A másik kígyó feje

Visszatérési érték

Ha ütközik a két fej, TRUE-t ad vissza, egyébként FALSE-t.

4.1.2.8. checkIncomingFruitCollision()

```
bool checkIncomingFruitCollision (
            fruit newFruit,
            SnakeBodyList * s )
```

Megnézi, hogy a frissen beilleszteni kívánt gyümölcs ütközne-e egy játékossal.

Paraméterek

newFruit	Az újonnan beillesztendő gyümölcs
s	A kígyó fejét tartalmazú struct.

Visszatérési érték

true értékkel tér vissza, ha ütközne a gyümölcs, false értékkel, ha nem. true érték esetén új pozíciót generál az add_Fruit() függvény.

4.1.2.9. checkScore()

```
int checkScore (
            Snake * s,
            scoreBoard_highscores hS )
```

Megnézi, hogy a játékos által elért eredmény benne van-e a top 10-ben.

Paraméterek

s	A kígyó adatait tartalmazó struct
hS	az eddigi legjobb 10 adatait tartalmazó scoreBoard_highscores struct.

Visszatérési érték

Ha top10-nek számít, akkor visszatér azzal az indexszel, aminek adatánál nagyobb(, jobb) az elért eredmény. Különben -1 -et ad.

4.1.2.10. checkWallHit()

```
bool checkWallHit (
            Snake s )
```

Megnézi, hogy a Snake kígyófej ütközött e éppen a fallal.

Paraméterek

s	Az adott kígyó feje.

Visszatérési érték

true értékkel tér vissza, ha a kígyó ütközött a fallal, és false-al, ha nem.

4.1.2.11. deleteFruit()

```
fruit * deleteFruit (
            fruit * fruitList,
            fruit * toBeDeleted )
```

Kitöröl egy gyümölcsöt a láncolt listából.

Paraméterek

fruitList	A gyümölcsöket tartalmazó láncolt lista.
toBeDeleted	A kitörlendő gyümölcs memóriacíme.

Visszatérési érték

Visszatér A gyümölcsöket tartalmazó láncolt lista elejére mutató pointerrel.

4.1.2.12. destroy_snakeBody()

```
void destroy_snakeBody (
            SnakeBodyList * s )
```

Felszabadítja a kígyó testéhez lefoglalt duplán láncolt listát.

Paraméterek

s	A felszabadítandó dupllán láncolt listához tartozó SnakeBodyList struct

4.1.2.13. destroyFruitList()

```
void destroyFruitList (
            fruit * fruitList )
```

Felszabadítja a gyümölcösket tartalmazó láncolt listát.

Paraméterek

fruitList	A gyümölcslista első elemére mutató pointer.

Visszatérési érték

Visszatér a frissített gyümölcslista elejáre mutató pointerre.

4.1.2.14. enter_text()

```
bool enter_text (
            char * where,
            size_t len,
```

```
            SDL_Rect box,
            SDL_Color backgroundColor,
            SDL_Color textColor )
```

4.1.2.15. exitProgram()

```
void exitProgram (
            global_Settings * g,
            FILE * fp,
            scoreBoard_highscores * highscores )
```

Bezárja az SDL könyvtárakat és kiírja az új dicsőségtáblát a fájlba,.

Paraméterek

g	A kígyó adatait tartalmazó struct
fp	A dicsőségtábla adatait tartalmazó fájl
highscores	A program futása során tárolt diőcségtábla.

4.1.2.16. gameSettingsLogic()

```
void gameSettingsLogic (
            global_Settings * g,
            Snake * snake1,
            Snake * snake2 )
```

A játék indítása előtti almenü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
snake1	Az első kígyó fejére/adataira mutató pointer.
snake2	A második kígyó fejére/adataira mutató pointer.

4.1.2.17. highscore_subRoutine()

```
void highscore_subRoutine (
            int snakeIndex,
            scoreBoard_highscores * hS,
            Snake playerSnake,
            short whichSnake )
```

Dicsőségtábla szubrutin.

Megjeleníti azt a menüt, ahol rekord elérése esetén a játékos beírhatja a nevét, és az elért pintszámát. Olyan színt használ, amilyen színt kiválasztott a játék kezdése előtt a játékos.

Paraméterek

snakeIndex	A dicsőségtábla egy indexe. 0 és 9 közt lehet.
hS	A program futása során tárolt diőcségtábla.
playerSnake	Az éppen feldolgozandó kígyó Snake structja.
whichSnake	1, vagy 2 értékű lehet. Megmondja, melyik kígyót kell éppen feldolgozni.

4.1.2.18. highScoresMenu_Logic()

```
void highScoresMenu_Logic (
            global_Settings * g,
            scoreBoard_highscores_Elements highScoreMenu )
```

A dicsőségtábla almenü vezérlő függvénye egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
highScoreMenu	A dicsőségtáblát tartalmazó buttonBoxokat tartalmazó menü.

4.1.2.19. inGameButtons()

```
void inGameButtons (
            ButtonBox * buttons,
            int len )
```

A játék futása alatt a "Menü" gomb megjelenítése.

Paraméterek

buttons	A kirenderelendő gomb(ok)
len	A renderelendő gomb(ok) darabszáma.

4.1.2.20. init_SnakeBody()

```
void init_SnakeBody (
            SnakeBodyList * tmp )
```

Előkészíti a kígyó testének duplán láncolt listáját.

Létrehozza a strázsát és a sentinelt. Összelinkeli őket, hogy egyásra mutassanak.

Paraméterek

sBody	Az inicializálandó SnakeBodyList struct

4.1.2.21. mainGame_Logic()

```
void mainGame_Logic (
            global_Settings * g,
            Snake * snake1,
            Snake * snake2,
            scoreBoard_highscores * hS )
```

A játék futása alatti logikai motor.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
snake1	Az első játékos kígyójának adatai
snake2	A második játékos kígyójának adatai
hS	A dicsőságtáblára mutató pointer

4.1.2.22. mainMenu_init()

```
void mainMenu_init (
            global_Settings * g )
```

A főmenü előkészítése, majd kirenderelése.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.1.2.23. mainMenuLogic()

```
void mainMenuLogic (
            global_Settings * g )
```

A menü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.1.2.24. moveBody()

```
void moveBody (
            SnakeBodyList * s,
            Snake * sHead )
```

Elmozdítja a kígyó testét.

Hátulról bejárja a láncolt listát, és mindig az adott elem előtti elem adatait átmásolja.

A láncolt lista legelején található elem a SnakeBodyList head elemében eltárolt x és y pozíciót másolja át. Ez a Snake fejelemből vevődik át.

Paraméterek

s	A kígyó testének strázsája, és sentinelje.
sHead	Snake típusú struct.

4.1.2.25. P1_Controller()

```
void P1_Controller (
            Snake * snake1,
            SDL_Event ev )
```

Az első számú kígyó vezérléséért felelős függvény.

Azért kell külön P1_Controller() és P2_Controller(), mert más billentyűkre változik a kígyók mozgása. Egyszerűbb különvéve kezelni őket.

Paraméterek

snake1	A kígyó adatait tartalmazó Snake structra mutató pointer
ev	Egy SDL_Event esemény

4.1.2.26. P2_Controller()

```
void P2_Controller (
            Snake * snake2,
            SDL_Event ev )
```

A második számú kígyó vezérléséért felelős függvény.

Azért kell külön P1_Controller() és P2_Controller(), mert más billentyűkre változik a kígyók mozgása. Egyszerűbb különvéve kezelni őket.

Paraméterek

snake2	A kígyó adatait tartalmazó Snake structra mutató pointer
ev	Egy SDL_Event esemény

4.1.2.27. randomize_snakePos()

```
void randomize_snakePos (
            Snake * s )
```

A Kígyó pozíciójának véletlenszerű elhelyezése.

Az eredmény alapján átállítja a Kígyó pozíciójára és sebességére vonatkozó beállításait.

Paraméterek

s	A kígyó adatait tartalmazó struct

4.1.2.28. render_gameSettingsMenu()

```
void render_gameSettingsMenu (
            global_Settings * g )
```

A játék indítása előtti almenü előkészítése, majd kirenderelése.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.1.2.29. render_highScoresMenu()

```
void render_highScoresMenu (
            global_Settings * g,
            scoreBoard_highscores_Elements m )
```

A dicsőségtábla kiírását megvalósító függvény.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
m	a dicsőségtábla grafikus elemeit tartalmazó struct.

4.1.2.30. resetSnake()

```
void resetSnake (
            Snake * s1 )
```

A Kígyó pozíciójának és utolsó irány karakterének alaphelyzetbe állítása.

Paraméterek

s	A kígyó adatait tartalmazó struct

4.1.2.31. resetSnakePoints()

```
void resetSnakePoints (
            Snake * s1 )
```

A Kígyó pontszámának visszaállítása.

Azért nem jó a resetSnake() függvény ehhez, mert a játék véget érése után a dicsőségtáblát kezelő függvényeknek még el kell éríük a kígyó pontszámát.

Paraméterek

s	A kígyó adatait tartalmazó struct

4.1.2.32. stopGame()

```
void stopGame (
            global_Settings * g )
```

Előkészíti a kilépést.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.1.3. Változók dokumentációja

4.1.3.1. globalSettings

```
global_Settings globalSettings
```

Kezdő érték:

```
={false,
                                true,
                                true,
                                false,
                                false,
                                false,
                                false,
                                false,
                                false,
                                false}
```

4.2. gameLogic.h fájlreferencia

```
#include <SDL.h>
#include <SDL2_gfxPrimitives.h>
#include <stdbool.h>
#include "graphics.h"
#include "menus.h"
```

Adatszerkezetek

- struct global_Settings

  A játék fő beállításait("főflagek") tartalmazó struct.

Típusdefiníciók

- typedef struct global_Settings global_Settings

Függvények

- void stopGame (global_Settings ∗g)

  Előkészíti a kilépést.
- void mainMenu_init (global_Settings ∗g)

  A főmenü előkészítése, majd kirenderelése.
- void mainMenuLogic (global_Settings ∗g)

  A menü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.
- void render_gameSettingsMenu (global_Settings ∗g)

  A játék indítása előtti almenü előkészítése, majd kirenderelése.
- void gameSettingsLogic (global_Settings ∗g, Snake ∗snake1, Snake ∗snake2)

A játék indítása előtti almenü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

- void render_highScoresMenu (global_Settings *g, scoreBoard_highscores_Elements m)

  A dicsőségtábla kiírását megvalósító függvény.
- void highScoresMenu_Logic (global_Settings *g, scoreBoard_highscores_Elements)

  A dicsőségtábla almenü vezérlő függvénye egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.
- void inGameButtons (ButtonBox *buttons, int len)

  A játék futása alatt a "Menü" gomb megjelenítése.
- void mainGame_Logic (global_Settings *g, Snake *snake1, Snake *snake2, scoreBoard_highscores *hS)

  A játék futása alatti logikai motor.
- void randomize_snakePos (Snake *s)

  A Kígyó pozíciójának véletlenszerű elhelyezése.
- void resetSnake (Snake *s1)

  A Kígyó pozíciójának és utolsó irány karakterének alaphelyzetbe állítása.
- void resetSnakePoints (Snake *s1)

  A Kígyó pontszámának visszaállítása.
- void P1_Controller (Snake *snake1, SDL_Event ev)

  Az első számú kígyó vezérléséért felelős függvény.
- void P2_Controller (Snake *s2, SDL_Event ev)

  A második számú kígyó vezérléséért felelős függvény.
- void changeHighScoreList (Snake *s, const char *playerName, int idx, scoreBoard_highscores *h)
- int checkScore (Snake *s, scoreBoard_highscores hS)

  Megnézi, hogy a játékos által elért eredmény benne van-e a top 10-ben.
- bool enter_text (char *where, size_t len, SDL_Rect box, SDL_Color backgroundColor, SDL_Color textColor)
- fruit * add_Fruit (fruit *firstFruit, SnakeBodyList snake1_L, SnakeBodyList snake2_L)

  Generál egy új gyümölcsöt, beteszi a gyümölcsöket tartalmazó lista elejére.
- void destroyFruitList (fruit *fruitList)

  Felszabadítja a gyümölcsöket tartalmazó láncolt listát.
- fruit * checkCollision (fruit *fruitList, Snake s)

  Megnézi, hogy egy kígó ütközik-e egy gymölcssel.
- fruit * deleteFruit (fruit *fruitList, fruit *toBeDeleted)

  Kitöröl egy gyümölcsöt a láncolt listából.
- bool checkIncomingFruitCollision (fruit newFruit, SnakeBodyList *s)

  Megnézi, hogy a frissen beilleszteni kívánt gyümölcs ütközne-e egy játékossal.
- void add_BodyElement (SnakeBodyList *o, Snake s)

  Megnöveli a kígyó hosszát 1-gyel.
- void init_SnakeBody (SnakeBodyList *sBody)

  Előkészíti a kígyó testének duplán láncolt listáját.
- void moveBody (SnakeBodyList *s, Snake *sHead)

  Elmozdítja a kígyó testét.
- void destroy_snakeBody (SnakeBodyList *s)

  Felszabadítja a kígyó testéhez lefoglalt duplán láncolt listát.
- bool checkBodyCollision (SnakeBodyList *sList, Snake *sHead)

  Megnézi, hogy két test (a.k.a. az egyik kígyó feje és a másik teste) ütközik-e.
- bool checkHeadCollision (Snake *sHead1, Snake *sHead2)

  A két fej ütközését vizsgálja meg.
- bool checkWallHit (Snake s)

  Megnézi, hogy a Snake kígyófej ütközött e éppen a fallal.
- void exitProgram (global_Settings *g, FILE *fp, scoreBoard_highscores *highscores)

  Bezárja az SDL könyvtárakat és kiírja az új dicsőségtáblát a fájlba,.

- void calculateNewScoreboard (scoreBoard_highscores *hS, Snake snake1, Snake snake2)

  Az új dicsőségtáblát elkészítő modul.
- void highscore_subRoutine (int snakeIndex, scoreBoard_highscores *hS, Snake playerSnake, short which↩ Snake)

  Dicsőségtábla szubrutin.

Változók

- global_Settings globalSettings

4.2.1. Típusdefiníciók dokumentációja

4.2.1.1. global_Settings

```
typedef struct global_Settings global_Settings
```

4.2.2. Függvények dokumentációja

4.2.2.1. add_BodyElement()

```
void add_BodyElement (
            SnakeBodyList * o,
            Snake s )
```

Megnöveli a kígyó hosszát 1-gyel.

Paraméterek

o	A kígyó testét tartalmazó SnakeBodyList struct
s	A kígyó fejét tartalmazú struct.

4.2.2.2. add_Fruit()

```
fruit * add_Fruit (
            fruit * firstFruit,
            SnakeBodyList snake1_L,
            SnakeBodyList snake2_L )
```

Generál egy új gyümölcsöt, beteszi a gyümölcsöket tartalmazó lista elejére.

Ha ez a gyümölcs ütközne az egyik kígyóval, akkor újat generál-

Paraméterek

firstFruit	A gyümölcslista első elemére mutató pointer.
snake1_L	Az 1. kígyó testét tartalmazó struct.
snake2_L	A 2. kígyó testét tartalmazó struct.

Visszatérési érték

Visszatér a frissített gyümölcslista elejáre mutató pointerre.

4.2.2.3. calculateNewScoreboard()

```
void calculateNewScoreboard (
            scoreBoard_highscores * hS,
            Snake snake1,
            Snake snake2 )
```

Az új dicsőségtáblát elkészítő modul.

Megkeresi, hogy van-e új rekord, majd az eredmény alapján továbblép, vagy meghívja a hisghscore_subRoutine() függvényt, hogy módosítsa a dicsőségtáblát.

Paraméterek

hS	A program futása során tárolt diőcségtábla.
snake1	Az egyik kígyó feje.
snake2	A másik kígyó feje

4.2.2.4. changeHighScoreList()

```
void changeHighScoreList (
            Snake * s,
            const char * playerName,
            int idx,
            scoreBoard_highscores * h )
```

4.2.2.5. checkBodyCollision()

```
bool checkBodyCollision (
            SnakeBodyList * sList,
            Snake * sHead )
```

Megnézi, hogy két test (a.k.a. az egyik kígyó feje és a másik teste) ütközik-e.

Paraméterek

sHead	Az egyik kígyó feje.
sList	A másik kígyó teste

Visszatérési érték

Ha ütközik a fej a testtel, TRUE-t ad vissza, egyébként FALSE-t.

4.2.2.6. checkCollision()

```
fruit * checkCollision (
            fruit * fruitList,
            Snake s )
```

Megnézi, hogy egy kígó ütközik-e egy gymölcssel.

Paraméterek

fruitList	A gyümölcslista első elemére mutató pointer.
s	a vizsgáladnó kígyó feje.

Visszatérési érték

Ha ütközik a kígyó valamelyik gyümölccsel, akkor a függvény vissztér az adott gyümölcsre mutató fruit∗ pointerrel. Egyéb esetben NULLpointert ad vissza.

4.2.2.7. checkHeadCollision()

```
bool checkHeadCollision (
            Snake * sHead1,
            Snake * sHead2 )
```

A két fej ütközését vizsgálja meg.

Paraméterek

sHead1	Az egyik kígyó feje.
sHead2	A másik kígyó feje

Visszatérési érték

Ha ütközik a két fej, TRUE-t ad vissza, egyébként FALSE-t.

4.2.2.8. checkIncomingFruitCollision()

```
bool checkIncomingFruitCollision (
            fruit newFruit,
            SnakeBodyList * s )
```

Megnézi, hogy a frissen beilleszteni kívánt gyümölcs ütközne-e egy játékossal.

Paraméterek

newFruit	Az újonnan beillesztendő gyümölcs
s	A kígyó fejét tartalmazú struct.

Visszatérési érték

true értékkel tér vissza, ha ütközne a gyümölcs, false értékkel, ha nem. true érték esetén új pozíciót generál az add_Fruit() függvény.

4.2.2.9. checkScore()

```
int checkScore (
            Snake * s,
            scoreBoard_highscores hS )
```

Megnézi, hogy a játékos által elért eredmény benne van-e a top 10-ben.

Paraméterek

s	A kígyó adatait tartalmazó struct
hS	az eddigi legjobb 10 adatait tartalmazó scoreBoard_highscores struct.

Visszatérési érték

Ha top10-nek számít, akkor visszatér azzal az indexszel, aminek adatánál nagyobb(, jobb) az elért eredmény. Különben -1 -et ad.

4.2.2.10. checkWallHit()

```
bool checkWallHit (
            Snake s )
```

Megnézi, hogy a Snake kígyófej ütközött e éppen a fallal.

Paraméterek

s	Az adott kígyó feje.

Visszatérési érték

true értékkel tér vissza, ha a kígyó ütközött a fallal, és false-al, ha nem.

4.2.2.11. deleteFruit()

```
fruit * deleteFruit (
            fruit * fruitList,
            fruit * toBeDeleted )
```

Kitöröl egy gyümölcsöt a láncolt listából.

Paraméterek

fruitList	A gyümölcsöket tartalmazó láncolt lista.
toBeDeleted	A kitörlendő gyümölcs memóriacíme.

Visszatérési érték

Visszatér A gyümölcsöket tartalmazó láncolt lista elejére mutató pointerrel.

4.2.2.12. destroy_snakeBody()

```
void destroy_snakeBody (
            SnakeBodyList * s )
```

Felszabadítja a kígyó testéhez lefoglalt duplán láncolt listát.

Paraméterek

s	A felszabadítandó dupllán láncolt listához tartozó SnakeBodyList struct

4.2.2.13. destroyFruitList()

```
void destroyFruitList (
            fruit * fruitList )
```

Felszabadítja a gyümölcösket tartalmazó láncolt listát.

Paraméterek

fruitList	A gyümölcslista első elemére mutató pointer.

Visszatérési érték

Visszatér a frissített gyümölcslista elejáre mutató pointerre.

4.2.2.14. enter_text()

```
bool enter_text (
            char * where,
            size_t len,
            SDL_Rect box,
            SDL_Color backgroundColor,
            SDL_Color textColor )
```

4.2.2.15. exitProgram()

```
void exitProgram (
            global_Settings * g,
            FILE * fp,
            scoreBoard_highscores * highscores )
```

Bezárja az SDL könyvtárakat és kiírja az új dicsőségtáblát a fájlba,.

Paraméterek

g	A kígyó adatait tartalmazó struct
fp	A dicsőségtábla adatait tartalmazó fájl
highscores	A program futása során tárolt diőcségtábla.

4.2.2.16. gameSettingsLogic()

```
void gameSettingsLogic (
            global_Settings * g,
            Snake * snake1,
            Snake * snake2 )
```

A játék indítása előtti almenü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
snake1	Az első kígyó fejére/adataira mutató pointer.
snake2	A második kígyó fejére/adataira mutató pointer.

4.2.2.17. highscore_subRoutine()

```
void highscore_subRoutine (
            int snakeIndex,
            scoreBoard_highscores * hS,
            Snake playerSnake,
            short whichSnake )
```

Dicsőségtábla szubrutin.

Megjeleníti azt a menüt, ahol rekord elérése esetén a játékos beírhatja a nevét, és az elért pintszámát. Olyan színt használ, amilyen színt kiválasztott a játék kezdése előtt a játékos.

Paraméterek

snakeIndex	A dicsőségtábla egy indexe. 0 és 9 közt lehet.
hS	A program futása során tárolt diőcségtábla.
playerSnake	Az éppen feldolgozandó kígyó Snake structja.
whichSnake	1, vagy 2 értékű lehet. Megmondja, melyik kígyót kell éppen feldolgozni.

4.2.2.18. highScoresMenu_Logic()

```
void highScoresMenu_Logic (
            global_Settings * g,
            scoreBoard_highscores_Elements highScoreMenu )
```

A dicsőségtábla almenü vezérlő függvénye egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
highScoreMenu	A dicsőségtáblát tartalmazó buttonBoxokat tartalmazó menü.

4.2.2.19. inGameButtons()

```
void inGameButtons (
```

```
            ButtonBox * buttons,
            int len )
```

A játék futása alatt a "Menü" gomb megjelenítése.

Paraméterek

buttons	A kirenderelendő gomb(ok)
len	A renderelendő gomb(ok) darabszáma.

4.2.2.20. init_SnakeBody()

```
void init_SnakeBody (
            SnakeBodyList * sBody )
```

Előkészíti a kígyó testének duplán láncolt listáját.

Létrehozza a strázsát és a sentinelt. Összelinkeli őket, hogy egyásra mutassanak.

Paraméterek

sBody	Az inicializálandó SnakeBodyList struct

4.2.2.21. mainGame_Logic()

```
void mainGame_Logic (
            global_Settings * g,
            Snake * snake1,
            Snake * snake2,
            scoreBoard_highscores * hS )
```

A játék futása alatti logikai motor.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
snake1	Az első játékos kígyójának adatai
snake2	A második játékos kígyójának adatai
hS	A dicsőségtáblára mutató pointer

4.2.2.22. mainMenu_init()

```
void mainMenu_init (
```

```
            global_Settings * g )
```

A főmenü előkészítése, majd kirenderelése.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.2.2.23. mainMenuLogic()

```
void mainMenuLogic (
            global_Settings * g )
```

A menü vezérlését kezelő függvény. egérgomlenyomás esetén meghívja a checkClick() függvényt, ellenőrizve, hogy rákattintott-e valamire a játékos. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.2.2.24. moveBody()

```
void moveBody (
            SnakeBodyList * s,
            Snake * sHead )
```

Elmozdítja a kígyó testét.

Hátulról bejárja a láncolt listát, és mindig az adott elem előtti elem adatait átmásolja.

A láncolt lista legelején található elem a SnakeBodyList head elemében eltárolt x és y pozíciót másolja át. Ez a Snake fejelemből vevődik át.

Paraméterek

s	A kígyó testének strázsája, és sentinelje.
sHead	Snake típusú struct.

4.2.2.25. P1_Controller()

```
void P1_Controller (
            Snake * snake1,
            SDL_Event ev )
```

Az első számú kígyó vezérléséért felelős függvény.

Azért kell külön P1_Controller() és P2_Controller(), mert más billentyűkre változik a kígyók mozgása. Egyszerűbb különvéve kezelni őket.

Paraméterek

snake1	A kígyó adatait tartalmazó Snake structra mutató pointer
ev	Egy SDL_Event esemény

4.2.2.26. P2_Controller()

```
void P2_Controller (
            Snake * snake2,
            SDL_Event ev )
```

A második számú kígyó vezérléséért felelős függvény.

Azért kell külön P1_Controller() és P2_Controller(), mert más billentyűkre változik a kígyók mozgása. Egyszerűbb különvéve kezelni őket.

Paraméterek

snake2	A kígyó adatait tartalmazó Snake structra mutató pointer
ev	Egy SDL_Event esemény

4.2.2.27. randomize_snakePos()

```
void randomize_snakePos (
            Snake * s )
```

A Kígyó pozíciójának véletlenszerű elhelyezése.

Az eredmény alapján átállítja a Kígyó pozíciójára és sebességére vonatkozó beállításait.

Paraméterek

s	A kígyó adatait tartalmazó struct

4.2.2.28. render_gameSettingsMenu()

```
void render_gameSettingsMenu (
            global_Settings * g )
```

A játék indítása előtti almenü előkészítése, majd kirenderelése.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.2.2.29. render_highScoresMenu()

```
void render_highScoresMenu (
            global_Settings * g,
            scoreBoard_highscores_Elements m )
```

A dicsőségtábla kiírását megvalósító függvény.

Paraméterek

g	A játék globális beállításait tartalmazó struct.
m	a dicsőségtábla grafikus elemeit tartalmazó struct.

4.2.2.30. resetSnake()

```
void resetSnake (
            Snake * s )
```

A Kígyó pozíciójának és utolsó irány karakterének alaphelyzetbe állítása.

Paraméterek

s	A kígyó adatait tartalmazó struct

4.2.2.31. resetSnakePoints()

```
void resetSnakePoints (
            Snake * s1 )
```

A Kígyó pontszámának visszaállítása.

Azért nem jó a resetSnake() függvény ehhez, mert a játék véget érése után a dicsőségtáblát kezelő függvényeknek még el kell éríük a kígyó pontszámát.

Paraméterek

s	A kígyó adatait tartalmazó struct

4.2.2.32. stopGame()

```
void stopGame (
            global_Settings * g )
```

Előkészíti a kilépést.

Paraméterek

g	A játék globális beállításait tartalmazó struct.

4.2.3. Változók dokumentációja

4.2.3.1. globalSettings

```
global_Settings globalSettings  [extern]
```

4.3. gameLogic.h

Ugrás a fájl dokumentációjához.

```
1
2 #ifndef SNAKEGAME_GAMELOGIC_H
3 #define SNAKEGAME_GAMELOGIC_H
4
5 #include <SDL.h>
6 #include <SDL2_gfxPrimitives.h>
7 #include <stdbool.h>
8 #include "graphics.h"
9 #include "menus.h"
10
14 typedef struct global_Settings{
15     bool twoPlayerMode;
16     bool show_mainMenu;
17     bool init_mainMenu;
18     bool show_gameSettings;
20     bool show_mainGame;
21     bool show_highScoreboard;
22     bool game_Init;
23     bool exitGame;
24     bool init_highScoreboard;
25     bool isRunning;
26 }global_Settings;
27 extern global_Settings globalSettings;
28
29 void stopGame(global_Settings *g);
30
31 void mainMenu_init(global_Settings *g);
32 void mainMenuLogic(global_Settings *g);
33
```

```
34 void render_gameSettingsMenu(global_Settings *g);
35 void gameSettingsLogic(global_Settings *g,Snake *snake1,Snake *snake2);
36
37 void render_highScoresMenu(global_Settings *g,scoreBoard_highscores_Elements m);
38 void highScoresMenu_Logic(global_Settings *g,scoreBoard_highscores_Elements);
39 void inGameButtons(ButtonBox *buttons,int len);
40
41 void mainGame_Logic(global_Settings *g, Snake *snake1, Snake *snake2,scoreBoard_highscores *hS);
       //,scoreBoard_highscores hS
42 void randomize_snakePos(Snake *s);
43
44 void resetSnake(Snake *s1);
45 void resetSnakePoints(Snake *s1);
46 void P1_Controller(Snake *snake1, SDL_Event ev);
47 void P2_Controller(Snake *s2, SDL_Event ev);
48
49 void changeHighScoreList(Snake *s, const char* playerName,int idx, scoreBoard_highscores *h);
50 int checkScore(Snake *s, scoreBoard_highscores hS);
51 bool enter_text(char *where, size_t len, SDL_Rect box, SDL_Color backgroundColor, SDL_Color textColor);
52 //Gyümölcsök
53 fruit* add_Fruit(fruit* firstFruit, SnakeBodyList snake1_L, SnakeBodyList snake2_L);
54 void destroyFruitList(fruit* fruitList);
55 fruit* checkCollision(fruit* fruitList,Snake s);
56 fruit* deleteFruit(fruit* fruitList,fruit *toBeDeleted);
57 bool checkIncomingFruitCollision(fruit newFruit,SnakeBodyList *s);
58
59 //Kígyótest
60 void add_BodyElement(SnakeBodyList *o,Snake s);
61 void init_SnakeBody(SnakeBodyList *sBody);
62 void moveBody(SnakeBodyList *s, Snake *sHead);
63 //void traverse_snakeBody(SnakeBodyList s);
64 void destroy_snakeBody(SnakeBodyList *s);
65 bool checkBodyCollision(SnakeBodyList *sList,Snake *sHead);
66 bool checkHeadCollision(Snake *sHead1, Snake *sHead2);
67 bool checkWallHit(Snake s);
68
69 //IO
70 void exitProgram(global_Settings *g,FILE *fp,scoreBoard_highscores *highscores);
71 void calculateNewScoreboard(scoreBoard_highscores *hS,Snake snake1, Snake snake2);
72 void highscore_subRoutine(int snakeIndex,scoreBoard_highscores *hS, Snake playerSnake,short whichSnake);
73 #endif //SNAKEGAME_GAMELOGIC_H
74
```

4.4. graphics.c fájlreferencia

A játék grafikájával, renderelésével foglalkozó modul.

```
#include "graphics.h"
#include "io.h"
#include "menus.h"
#include <SDL.h>
#include <SDL2_gfxPrimitives.h>
#include <SDL_ttf.h>
#include <SDL_image.h>
#include "debugmalloc.h"
```

Függvények

- Uint32 add_waitEvent (Uint32 ms, void *param)
- int initSDL_everything ()

  Betölti a játékhoz szükséges SDL könyvtárat, illetve az SDL2gfx további szükséges almoduljait.
- int renderText (TTF_Font *textFont, SDL_Surface *textSurface, SDL_Texture *textTexture, SDL_Rect where, int r, int g, int b, char *Text)
- int renderText_middle (TTF_Font *textFont, SDL_Surface *textSurface, SDL_Texture *textTexture, SDL_Rect where, int r, int g, int b, char *Text)
- void renderMenu_middle (const TTF_Font *textFont, SDL_Surface *textSurface, SDL_Texture *textTexture, const ButtonBox *buttons, const int lenMenu)

Előkészít a renderer számára egy menüt. A szövegek középre igazítva jelennek majd meg benne.

- void renderMenu (const TTF_Font ∗textFont, SDL_Surface ∗textSurface, SDL_Texture ∗textTexture, const ButtonBox ∗buttons, const int lenMenu)

  Előkészít a renderer számára egy menüt. A szövegek balra zártan jelennek majd meg benne.
- void setFPS (int fps)

  Beállítja, hány képkocka/másodperccel működjön a játék.
- void renderSnakeBody (SnakeBodyList ∗o, Snake s)

  Megjeleníti a kígyó testét.
- Uint32 allow_fruitAdd (Uint32 ms, void ∗param)

  Engedélyez egy gyümölcs hozzáadását.
- void renderFruits (fruit ∗fruitList)

  Megjeleníti a gyümölcsöket,.
- Uint32 allow_moveSnake (Uint32 ms, void ∗param)

  Engedélyezi a kígyó mozgatását.

Változók

- SDL_Rect hova = { 0, 0, 0, 0 }
- int moveMentScale =80

4.4.1. Részletes leírás

A játék grafikájával, renderelésével foglalkozó modul.

4.4.2. Függvények dokumentációja

4.4.2.1. add_waitEvent()

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

4.4.2.2. allow_fruitAdd()

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

Engedélyez egy gyümölcs hozzáadását.

Paraméterek

ms	Az idő, amit két engedélyezés közt várjon a függvény
param	Egyéb poaraméterek, ha vannak

Visszatérési érték

Betesz egy SDL_USEREVENT eseményt az event stackra. Ennek az eventnek 42-es számú egyedi kódot ad.

4.4.2.3. allow_moveSnake()

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

Engedélyezi a kígyó mozgatását.

Paraméterek

ms	Az idő, amit két engedélyezés közt várjon a függvény
param	Egyéb poaraméterek, ha vannak

Visszatérési érték

Betesz egy SDL_USEREVENT eseményt az event stackra. Ennek az eventnek 43-as számú egyedi kódot ad.

4.4.2.4. initSDL_everything()

```
int initSDL_everything ( )
```

Betölti a játékhoz szükséges SDL könyvtárat, illetve az SDL2gfx további szükséges almoduljait.

Ha valamelyik almodul betöltése sikertelen, akkor a programot a hibatáblázatban megfelelő kóddal megállítja.

(Kódszám, és a hozzátartozó hiba) #| Hiba jelentése 1| Nem indítható az SDL. 2| Nem hozható létre az ablak. 3| Nem hozható létre a renderer (megjelenítő). 4| Nem sikerült betölteni az 1. fontot. 5| Nem sikerült betölteni a 2. fontot.

Visszatérési érték

visszaad 1 -et, ha sikerült a betöltés.

4.4.2.5. renderFruits()

```
void renderFruits (
            fruit * fruitList )
```

Megjeleníti a gyümölcsöket,.

Paraméterek

fruitList	A gyümölcsöket tartalmazó láncolt lista első elemére mutató pointer.

Visszatérési érték

Azonnal visszatér, ha fruitList NULL-ra mutat. .

4.4.2.6. renderMenu()

```
void renderMenu (
            const TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            const ButtonBox * buttons,
            const int lenMenu )
```

Előkészít a renderer számára egy menüt. A szövegek balra zártan jelennek majd meg benne.

Paraméterek

textFont	A használandó betűtípus.
textSurface	A renderernek szükséges felület, amire rárendereli majd a szöveget.
textTexture	A szöveg textúrája. Ezt felhasználva készíti majd el a renderer a megjeleníthető szöveget a textSurface -re.
renderer	Az SDL renderelője.
buttons	A megjelenítendő gombokat, grafikai elemeket tartalmazó ButtonBox struct.
lenMenu	A megjelenítendő grafikai elemek száma.

4.4.2.7. renderMenu_middle()

```
void renderMenu_middle (
            const TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            const ButtonBox * buttons,
            const int lenMenu )
```

Előkészít a renderer számára egy menüt. A szövegek középre igazítva jelennek majd meg benne.

Paraméterek

textFont	A használandó betűtípus.
textSurface	A RAM-ban tárolt, szoftveresen renderelt pixelhalmaz.
textTexture	A VRAM-ban tárolt, szoftveresen renderelt pixelhalmaz.
buttons	A megjelenítendő gombokat, grafikai elemeket tartalmazó ButtonBox struct.
lenMenu	A megjelenítendő grafikai elemek száma.

4.4.2.8. renderSnakeBody()

```
void renderSnakeBody (
            SnakeBodyList * o,
            Snake s )
```

Megjeleníti a kígyó testét.

Paraméterek

o	A Kígyó testének strázsáját és sentineljét tartalmazza.
s	A Kígyó feje, és az adaiatit tartalmazó Snake struct.

4.4.2.9. renderText()

```
int renderText (
            TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            SDL_Rect where,
            int r,
            int g,
            int b,
            char * Text )
```

4.4.2.10. renderText_middle()

```
int renderText_middle (
            TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            SDL_Rect where,
            int r,
            int g,
            int b,
            char * Text )
```

4.4.2.11. setFPS()

```
void setFPS (
            int fps )
```

Beállítja, hány képkocka/másodperccel működjön a játék.

Paraméterek

fps	A beállítandó képkockasebesség.

4.4.3. Változók dokumentációja

4.4.3.1. hova

```
SDL_Rect hova = { 0, 0, 0, 0 }
```

4.4.3.2. moveMentScale

```
int moveMentScale =80
```

Szorzó, amivel a mozgás finomhangolható. Mivel a kígyó egy-egy eleme 20x20px méretű, a P1_Controller() és a P2_Controller() 0.25-értéket ad a sebességekhez, a moveMentscale alapértelmezetten 80, hogy 20px távot tegyen meg a kígyó egy-egy mozgásnál.

4.5. graphics.h fájlreferencia

A játék grafikájával, renderelésével foglalkozó modul fejléce.

```
#include <SDL.h>
#include <SDL2_gfxPrimitives.h>
#include <SDL_ttf.h>
#include <stdbool.h>
#include "menus.h"
```

Adatszerkezetek

- struct Window

  A megjelenítő ablak adatai.
- struct SnakeBody

  A kígyó testét tartalmazó struct. A fej adatait nem tartalmazza, azt a Snake struct tárolja. Több SnakeBody struct együtt egy duplán láncolt listát alkot.
- struct SnakeBodyList

  A kígyó testének strázsája, és sentinelje.
- struct Snake

  A kígyó, és fejének adatait tartalmazó struct.
- struct fruit

  Egy gyümölcs adatait tartalmazó struct.

Típusdefiníciók

- typedef struct Window Window
- typedef struct SnakeBody SnakeBody
- typedef struct SnakeBodyList SnakeBodyList
- typedef struct Snake Snake
- typedef struct fruit fruit

Függvények

- Uint32 add_waitEvent (Uint32 ms, void ∗param)
- int initSDL_everything ()

  Betölti a játékhoz szükséges SDL könyvtárat, illetve az SDL2gfx további szükséges almoduljait.
- int renderText (TTF_Font ∗textFont, SDL_Surface ∗textSurface, SDL_Texture ∗textTexture, SDL_Rect where, int r, int g, int b, char ∗Text)
- void renderMenu (const TTF_Font ∗textFont, SDL_Surface ∗textSurface, SDL_Texture ∗textTexture, const ButtonBox ∗buttons, int lenMenu)

  Előkészít a renderer számára egy menüt. A szövegek balra zártan jelennek majd meg benne.
- void renderMenu_middle (const TTF_Font ∗textFont, SDL_Surface ∗textSurface, SDL_Texture ∗textTexture, const ButtonBox ∗buttons, int lenMenu)

  Előkészít a renderer számára egy menüt. A szövegek középre igazítva jelennek majd meg benne.
- Uint32 allow_fruitAdd (Uint32 ms, void ∗param)

  Engedélyez egy gyümölcs hozzáadását.
- Uint32 allow_moveSnake (Uint32 ms, void ∗param)

  Engedélyezi a kígyó mozgatását.
- void renderFruits (fruit ∗fruitList)

  Megjeleníti a gyümölcsöket,.
- void renderSnakeBody (SnakeBodyList ∗o, Snake s)

  Megjeleníti a kígyó testét.
- void setFPS (int fps)

  Beállítja, hány képkocka/másodperccel működjön a játék.

Változók

- SDL_Event event
- SDL_Renderer ∗ renderer
- SDL_Window ∗ window
- SDL_TimerID id
- SDL_Surface ∗ text_Surface
- SDL_Texture ∗ text_Texture
- SDL_Rect rect_where
- TTF_Font ∗ font1
- TTF_Font ∗ font2
- int moveMentScale

4.5.1. Részletes leírás

A játék grafikájával, renderelésével foglalkozó modul fejléce.

4.5.2. Típusdefiníciók dokumentációja

4.5.2.1. fruit

```
typedef struct fruit fruit
```

4.5.2.2. Snake

```
typedef struct Snake Snake
```

4.5.2.3. SnakeBody

```
typedef struct SnakeBody SnakeBody
```

4.5.2.4. SnakeBodyList

```
typedef struct SnakeBodyList SnakeBodyList
```

4.5.2.5. Window

```
typedef struct Window Window
```

4.5.3. Függvények dokumentációja

4.5.3.1. add_waitEvent()

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

4.5.3.2. allow_fruitAdd()

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

Engedélyez egy gyümölcs hozzáadását.

Paraméterek

ms	Az idő, amit két engedélyezés közt várjon a függvény
param	Egyéb poaraméterek, ha vannak

Visszatérési érték

Betesz egy SDL_USEREVENT eseményt az event stackra. Ennek az eventnek 42-es számú egyedi kódot ad.

4.5.3.3. allow_moveSnake()

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

Engedélyezi a kígyó mozgatását.

Paraméterek

ms	Az idő, amit két engedélyezés közt várjon a függvény
param	Egyéb poaraméterek, ha vannak

Visszatérési érték

Betesz egy SDL_USEREVENT eseményt az event stackra. Ennek az eventnek 43-as számú egyedi kódot ad.

4.5.3.4. initSDL_everything()

```
int initSDL_everything ( )
```

Betölti a játékhoz szükséges SDL könyvtárat, illetve az SDL2gfx további szükséges almoduljait.

Ha valamelyik almodul betöltése sikertelen, akkor a programot a hibatáblázatban megfelelő kóddal megállítja.

(Kódszám, és a hozzátartozó hiba) #| Hiba jelentése 1| Nem indítható az SDL. 2| Nem hozható létre az ablak. 3| Nem hozható létre a renderer (megjelenítő). 4| Nem sikerült betölteni az 1. fontot. 5| Nem sikerült betölteni a 2. fontot.

Visszatérési érték

visszaad 1 -et, ha sikerült a betöltés.

4.5.3.5. renderFruits()

```
void renderFruits (
            fruit * fruitList )
```

Megjeleníti a gyümölcsöket,.

Paraméterek

fruitList	A gyümölcsöket tartalmazó láncolt lista első elemére mutató pointer.

Visszatérési érték

Azonnal visszatér, ha fruitList NULL-ra mutat. .

4.5.3.6. renderMenu()

```
void renderMenu (
            const TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            const ButtonBox * buttons,
            const int lenMenu )
```

Előkészít a renderer számára egy menüt. A szövegek balra zártan jelennek majd meg benne.

Paraméterek

textFont	A használandó betűtípus.
textSurface	A renderernek szükséges felület, amire rárendereli majd a szöveget.
textTexture	A szöveg textúrája. Ezt felhasználva készíti majd el a renderer a megjeleníthető szöveget a textSurface -re.
renderer	Az SDL renderelője.
buttons	A megjelenítendő gombokat, grafikai elemeket tartalmazó ButtonBox struct.
lenMenu	A megjelenítendő grafikai elemek száma.

4.5.3.7. renderMenu_middle()

```
void renderMenu_middle (
            const TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            const ButtonBox * buttons,
            const int lenMenu )
```

Előkészít a renderer számára egy menüt. A szövegek középre igazítva jelennek majd meg benne.

Paraméterek

textFont	A használandó betűtípus.
textSurface	A RAM-ban tárolt, szoftveresen renderelt pixelhalmaz.
textTexture	A VRAM-ban tárolt, szoftveresen renderelt pixelhalmaz.
buttons	A megjelenítendő gombokat, grafikai elemeket tartalmazó ButtonBox struct.
lenMenu	A megjelenítendő grafikai elemek száma.

4.5.3.8. renderSnakeBody()

```
void renderSnakeBody (
            SnakeBodyList * o,
            Snake s )
```

Megjeleníti a kígyó testét.

Paraméterek

o	A Kígyó testének strázsáját és sentineljét tartalmazza.
s	A Kígyó feje, és az adaiatit tartalmazó Snake struct.

4.5.3.9. renderText()

```
int renderText (
            TTF_Font * textFont,
            SDL_Surface * textSurface,
            SDL_Texture * textTexture,
            SDL_Rect where,
            int r,
            int g,
            int b,
            char * Text )
```

4.5.3.10. setFPS()

```
void setFPS (
            int fps )
```

Beállítja, hány képkocka/másodperccel működjön a játék.

Paraméterek

fps	A beállítandó képkockasebesség.

4.5.4. Változók dokumentációja

4.5.4.1. event

```
SDL_Event event
```

4.5.4.2. font1

```
TTF_Font* font1
```

Elsődleges betűtípus.

4.5.4.3. font2

```
TTF_Font* font2
```

Másodlagos betűtípus.

4.5.4.4. id

```
SDL_TimerID id
```

A megfelelő képcsokasebesség betartásáért felelős időzítő.

4.5.4.5. moveMentScale

```
int moveMentScale  [extern]
```

Szorzó, amivel a mozgás finomhangolható. Mivel a kígyó egy-egy eleme 20x20px méretű, a P1_Controller() és a P2_Controller() 0.25-értéket ad a sebességekhez, a moveMentscale alapértelmezetten 80, hogy 20px távot tegyen meg a kígyó egy-egy mozgásnál.

4.5.4.6. rect_where

```
SDL_Rect rect_where  [extern]
```

Tartalmazza, hova másolódjona kijelzőn egy textúra. lásd: renderText() függvény.

4.5.4.7. renderer

```
SDL_Renderer* renderer
```

A futás alatt használt megjelenítő

4.5.4.8. text_Surface

```
SDL_Surface* text_Surface
```

4.5.4.9. text_Texture

```
SDL_Texture* text_Texture
```

4.5.4.10. window

```
SDL_Window* window
```

Az ablak, amiben a játék fut.

4.6. graphics.h

Ugrás a fájl dokumentációjához.

```
1 #ifndef SNAKEGAME_GRAPHICS_H
2 #define SNAKEGAME_GRAPHICS_H
3
8 #include <SDL.h>
9 #include <SDL2_gfxPrimitives.h>
10 #include <SDL_ttf.h>
11 #include <stdbool.h>
12 #include "menus.h"
13
17 typedef struct Window {
18     int width;
19     int height;
20 }Window;
21
26 typedef struct SnakeBody{
27     int x;
28     int y;
29     struct SnakeBody *next;
30     struct SnakeBody *prev;
31 }SnakeBody;
32
36 typedef struct SnakeBodyList{
37     SnakeBody *head;
38     SnakeBody *last;
39 }SnakeBodyList;
40
44 typedef struct Snake{
45     int x;
46     int y;
47     double vx;
48     double vy;
49     unsigned int r;
50     unsigned int g;
51     unsigned int b;
52     int points;
53     struct SnakeBody *firstBodyElement;
54     char lastPos;
55 } Snake;
56
60 typedef struct fruit{
61     int x;
62     int y;
63     SDL_Color color;
64     struct fruit *nextFruit;
65 }fruit;
66
```

```
67
68 Uint32 add_waitEvent(Uint32 ms, void *param);
69
70 SDL_Event event;
71 SDL_Renderer *renderer;
72 SDL_Window *window;
73 SDL_TimerID id;
74 SDL_Surface *text_Surface;
75 SDL_Texture *text_Texture;
76
77 extern SDL_Rect rect_where;
78 TTF_Font *font1;
79 TTF_Font *font2;
81 extern int moveMentScale;
102 int initSDL_everything();
116 int renderText(TTF_Font *textFont,SDL_Surface *textSurface,SDL_Texture *textTexture, SDL_Rect where,int
       r, int g, int b, char* Text);
117
127 void renderMenu(const TTF_Font *textFont, SDL_Surface *textSurface, SDL_Texture *textTexture,const
       ButtonBox *buttons, int lenMenu);
136 void renderMenu_middle(const TTF_Font *textFont, SDL_Surface *textSurface, SDL_Texture
       *textTexture,const ButtonBox *buttons, int lenMenu);
149 Uint32 allow_fruitAdd(Uint32 ms, void *param);
150
158 Uint32 allow_moveSnake(Uint32 ms, void *param);
159
165 void renderFruits(fruit *fruitList);
166
172 void renderSnakeBody(SnakeBodyList *o,Snake s);
173
174 void setFPS(int fps);
175
176
177 #endif
```

4.7. io.c fájlreferencia

A játék fájlkezeléssel foglalkozó adatait tartalmazó modul.

```
#include "io.h"
```

Függvények

- scoreBoard_highscores loadScoreBoard (FILE ∗scoreboardTxt)

  Betölti a dicsőségtábla adatait. Az eredmény alapján átállítja a játék globális beállításait.
- void writeScoreBoardToFile (FILE ∗scoreboardTxt, scoreBoard_highscores hS)

  Elmenti a dicsőségtáblát egy külső fájlba.

4.7.1. Részletes leírás

A játék fájlkezeléssel foglalkozó adatait tartalmazó modul.

4.7.2. Függvények dokumentációja

4.7.2.1. loadScoreBoard()

```
scoreBoard_highscores loadScoreBoard (
            FILE * scoreboardTxt )
```

Betölti a dicsőségtábla adatait. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

scoreboardTxt	A dicsőségtábla adatait tartalmazó szüveges fájl.

Visszatérési érték

visszaad egy scoreBoard_highscores structot, benne a dicsőségtábla adataival.

4.7.2.2. writeScoreBoardToFile()

```
void writeScoreBoardToFile (
            FILE * scoreboardTxt,
            scoreBoard_highscores hS )
```

Elmenti a dicsőségtáblát egy külső fájlba.

Figyelmeztetés

Windows rendszeren ki kell hagyni a fájlnév elől a "../" karaktereket!

Paraméterek

scoreboardTxt	A dicsőségtábla adatait tartalmazó szüveges fájl.
hS	A program futása alatt a frissített dicsőségtáblát tartalmazó struct.

4.8. io.h fájlreferencia

A játék fájlkezeléssel foglalkozó adatait tartalmazó modul.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include "string.h"
```

Adatszerkezetek

- struct highScorePlayer

  A dicsőségtábla egy elemének az adatai.
- struct scoreBoard_highscores

  A dicsőségtáblát tartalmazú struct.

Típusdefiníciók

- typedef struct highScorePlayer highScorePlayer
- typedef struct scoreBoard_highscores scoreBoard_highscores

Függvények

- scoreBoard_highscores loadScoreBoard (FILE ∗scoreboardTxt)

  Betölti a dicsőségtábla adatait. Az eredmény alapján átállítja a játék globális beállításait.
- void writeScoreBoardToFile (FILE ∗scoreboardTxt, scoreBoard_highscores hS)

  Elmenti a dicsőségtáblát egy külső fájlba.
- void loadSettings ()
- void saveSettings ()

4.8.1. Részletes leírás

A játék fájlkezeléssel foglalkozó adatait tartalmazó modul.

4.8.2. Típusdefiníciók dokumentációja

4.8.2.1. highScorePlayer

```
typedef struct highScorePlayer highScorePlayer
```

4.8.2.2. scoreBoard_highscores

```
typedef struct scoreBoard_highscores scoreBoard_highscores
```

4.8.3. Függvények dokumentációja

4.8.3.1. loadScoreBoard()

```
scoreBoard_highscores loadScoreBoard (
            FILE * scoreboardTxt )
```

Betölti a dicsőségtábla adatait. Az eredmény alapján átállítja a játék globális beállításait.

Paraméterek

scoreboardTxt	A dicsőségtábla adatait tartalmazó szüveges fájl.

Visszatérési érték

visszaad egy scoreBoard_highscores structot, benne a dicsőségtábla adataival.

4.8.3.2. loadSettings()

```
void loadSettings ( )
```

4.8.3.3. saveSettings()

```
void saveSettings ( )
```

4.8.3.4. writeScoreBoardToFile()

```
void writeScoreBoardToFile (
            FILE * scoreboardTxt,
            scoreBoard_highscores hS )
```

Elmenti a dicsőségtáblát egy külső fájlba.

Figyelmeztetés

Windows rendszeren ki kell hagyni a fájlnév elől a "../" karaktereket!

Paraméterek

scoreboardTxt	A dicsőségtábla adatait tartalmazó szüveges fájl.
hS	A program futása alatt a frissített dicsőségtáblát tartalmazó struct.

4.9. io.h

Ugrás a fájl dokumentációjához.

```
1
5 #ifndef SNAKEGAME_IO_H
6 #define SNAKEGAME_IO_H
7
8 #include <stdio.h>
9 #include <stdbool.h>
10 #include <stdlib.h>
11 #include "string.h"
12
13
17 typedef struct highScorePlayer{
18     char name[50];
19     int score;
```

```
20 }highScorePlayer;
21
22
23
27 typedef struct scoreBoard_highscores {
28     highScorePlayer data[10];
29 } scoreBoard_highscores;
30
31 scoreBoard_highscores loadScoreBoard(FILE* scoreboardTxt);
32 void writeScoreBoardToFile(FILE* scoreboardTxt,scoreBoard_highscores hS);
33
34
35 void loadSettings();
36 void saveSettings();
37
38
39 #endif //SNAKEGAME_IO_H
```

4.10. main.c fájlreferencia

A játék főmodulja.

```
#include <stdlib.h>
#include <time.h>
#include "stdbool.h"
#include "gameLogic.h"
#include "menus.h"
#include "io.h"
#include "graphics.h"
```

Függvények

- int main (int argc, char ∗argv[ ])

4.10.1. Részletes leírás

A játék főmodulja.

Figyelmeztetés

Windows rendszeren ki kell hagyni a projekten belül minden olyan fájlnév elől a "../" karaktereket, ahol meg akarunk nyitni egy fájlt. pl.: fopen();

4.10.2. Függvények dokumentációja

4.10.2.1. main()

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

< Dicsőségtáblát tartalmazó fájl..

4.11. menus.c fájlreferencia

```
#include "menus.h"
#include "io.h"
```

Függvények

- scoreBoard_highscores_Elements create_highscores_menuElements (scoreBoard_highscores sB_H)

  A dicsőségtábla betöltése ButtonBox -okba, így megjeleníthetővé téve azokat.
- int checkClick (ButtonBox *buttons, int menuLen, int mousePosX, int mousePosY)

  Megnézi, hogy az ablakban kattintás érvényes volt e, azaz, egy grafikai elemre történt-e a kattintás.

Változók

- ButtonBox mainMenu [3]
- ButtonBox gameSettingsMenu [10]
- ButtonBox gameSettingsMenu_multi [6]
- ButtonBox inGameMenu [1]

4.11.1. Függvények dokumentációja

4.11.1.1. checkClick()

```
int checkClick (
            ButtonBox * buttons,
            int menuLen,
            int mousePosX,
            int mousePosY )
```

Megnézi, hogy az ablakban kattintás érvényes volt e, azaz, egy grafikai elemre történt-e a kattintás.

Visszatérési érték

Ha érvényes a kattintás, visszatér az érvényes ButtonBox ButtonBox.value értékével. Egyébként -1 -et ad.

4.11.1.2. create_highscores_menuElements()

```
scoreBoard_highscores_Elements create_highscores_menuElements (
            scoreBoard_highscores sB_H )
```

A dicsőségtábla betöltése ButtonBox -okba, így megjeleníthetővé téve azokat.

Paraméterek

sB↩ _H	A dicsőségtábla structja.

4.11.2. Változók dokumentációja

4.11.2.1. gameSettingsMenu

```
ButtonBox gameSettingsMenu[10]
```

Kezdő érték:

```
={
        {1,216,233,168,150,120,320,160,"1 Játékos",26,26,25},
        {2,77,77,77,400,120,570,160,"2 játékos",26,26,25},
        {-1,213,113,73,60,310,130,385,"P1",0,0,0},
        {103,213,113,73,175+25,322,225+25,372,"",26,26,25},
        {104,21, 109, 150,240+25,322,290+25,372,"",26,26,25},
        {105,1212, 192, 174,305+25,322,355+25,372,"",26,26,25},
        {106,26, 193, 199,370+25,322,420+25,372,"",26,26,25},
        {107,205, 230, 69,435+25,322,485+25,372,"",26,26,25},
        {5,216,233,168,(720/2-100),720/2+190,(720/2+100),720/2+230,"Kezdés",26,26,25},
        {3,216,233,168,(720/2-100),720/2+250,(720/2+100),720/2+290,"Menü",26,26,25}
}
```

A játék indítás előtti menüjének adatait tartalmazza.

4.11.2.2. gameSettingsMenu_multi

```
ButtonBox gameSettingsMenu_multi[6]
```

Kezdő érték:

```
={
        {-1,109,152,134,60,410,130,485,"P2",0,0,0},
        {201,109,152,134,175+25,422,225+25,472,"",26,26,25},
        {202,176, 19, 90,240+25,422,290+25,472,"",26,26,25},
        {203,85, 62, 168,305+25,422,355+25,472,"",26,26,25},
        {204,35, 68, 186,370+25,422,420+25,472,"",26,26,25},
        {205,176, 114, 77,435+25,422,485+25,472,"",26,26,25},
}
```

4.11.2.3. inGameMenu

```
ButtonBox inGameMenu[1]
```

Kezdő érték:

```
={
        {0,255,255,53,15,645,100,675,"Menü",149,30,1}
}
```

A játék játék közbeni menüjének adatait tartalmazza.

4.11.2.4. mainMenu

```
ButtonBox mainMenu[3]
```

Kezdő érték:

```
={
        {1,216,233,168,360-200,720/2-100,360+200,720/2-50,"Játék",26,26,25},
        {2,216,233,168,720/2-200,720/2-25,720/2+200,720/2+25,"Dicsőségtábla",26,26,25},
        {3,216,233,168,720/2-200,720/2+50,720/2+200,720/2+100,"Kilépés",149,1,1}
}
```

A főmenü grafikai adatait tartalmazza.

4.12. menus.h fájlreferencia

A menükkal kapcsolatos függvényeket tartlalmazó modulhoz tartozó header.

```
#include "io.h"
```

Adatszerkezetek

- struct ButtonBox

  Egy menü egy grafikai elemének adatait tartalmazó struct.
- struct scoreBoard_highscores_Elements

  A dicsőségtábla elemeit tartalmazú struct.

Típusdefiníciók

- typedef struct ButtonBox ButtonBox
- typedef struct scoreBoard_highscores_Elements scoreBoard_highscores_Elements

Függvények

- scoreBoard_highscores_Elements create_highscores_menuElements (scoreBoard_highscores sB_H)

  A dicsőségtábla betöltése ButtonBox -okba, így megjeleníthetővé téve azokat.
- int checkClick (ButtonBox ∗buttons, int menuLen, int mousePosX, int mousePosY)

  Megnézi, hogy az ablakban kattintás érvényes volt e, azaz, egy grafikai elemre történt-e a kattintás.

Változók

- ButtonBox mainMenu [3]
- ButtonBox gameSettingsMenu [10]
- ButtonBox gameSettingsMenu_multi [6]
- ButtonBox inGameMenu [1]

4.12.1. Részletes leírás

A menükkal kapcsolatos függvényeket tartlalmazó modulhoz tartozó header.

4.12.2. Típusdefiníciók dokumentációja

4.12.2.1. ButtonBox

```
typedef struct ButtonBox ButtonBox
```

4.12.2.2. scoreBoard_highscores_Elements

```
typedef struct scoreBoard_highscores_Elements scoreBoard_highscores_Elements
```

4.12.3. Függvények dokumentációja

4.12.3.1. checkClick()

```
int checkClick (
            ButtonBox * buttons,
            int menuLen,
            int mousePosX,
            int mousePosY )
```

Megnézi, hogy az ablakban kattintás érvényes volt e, azaz, egy grafikai elemre történt-e a kattintás.

Visszatérési érték

Ha érvényes a kattintás, visszatér az érvényes ButtonBox ButtonBox.value értékével. Egyébként -1 -et ad.

4.12.3.2. create_highscores_menuElements()

```
scoreBoard_highscores_Elements create_highscores_menuElements (
            scoreBoard_highscores sB_H )
```

A dicsőségtábla betöltése ButtonBox -okba, így megjeleníthetővé téve azokat.

Paraméterek

sB_H	A dicsőségtábla structja.

4.12.4. Változók dokumentációja

4.12.4.1. gameSettingsMenu

```
ButtonBox gameSettingsMenu[10]  [extern]
```

A játék indítás előtti menüjének adatait tartalmazza.

4.12.4.2. gameSettingsMenu_multi

```
ButtonBox gameSettingsMenu_multi[6]  [extern]
```

4.12.4.3. inGameMenu

```
ButtonBox inGameMenu[1]  [extern]
```

A játék játék közbeni menüjének adatait tartalmazza.

4.12.4.4. mainMenu

```
ButtonBox mainMenu[3]  [extern]
```

A főmenü grafikai adatait tartalmazza.

4.13. menus.h

Ugrás a fájl dokumentációjához.

```
1 //
2 // Created by Gutási Ádám on 2021. 11. 01..
3 //
4
5 #ifndef SNAKEGAME_MENUS_H
6 #define SNAKEGAME_MENUS_H
7
12 #include "io.h"
13
17 typedef struct ButtonBox{
18     int value;
19     int colorR;
20     int colorG;
21     int colorB;
22     int posX1;
23     int posY1;
24     int posX2;
25     int posY2;
26     char* text;
27     int textColorR;
28     int textColorG;
29     int textColorB;
30 }ButtonBox;
31
32
36 typedef struct scoreBoard_highscores_Elements{
37     ButtonBox menuElements[11];
38 }scoreBoard_highscores_Elements;
39
40
41 extern ButtonBox mainMenu[3];
42 extern ButtonBox gameSettingsMenu[10];
43 extern ButtonBox gameSettingsMenu_multi[6];
44 extern ButtonBox inGameMenu[1];
52 scoreBoard_highscores_Elements create_highscores_menuElements(scoreBoard_highscores sB_H);
53
58 int checkClick(ButtonBox *buttons,int menuLen,int mousePosX, int mousePosY);
59
60 #endif //SNAKEGAME_MENUS_H
```