# Kígyós játék – Programozói dokumentáció

Czumbel Péter – ODZF0M

## A játék működése – core.c modul

### A játék menete röviden

Egy vagy két kígyó mozog egy négyzet alakú, négyzetrácsos pálya mezőin. A mezőkön almák kerülnek elhelyezésre, amit, ha a kígyók megesznek, hosszabbak lesznek. Ha a kígyók a falba, magukba vagy egymásba ütköznek, akkor meghalnak, a játéknak vége.
A core.c modul végzi az összes ehhez kapcsolódó számítást és műveletet.

### A játékelemek

A kígyók testének darabkái egy-egy láncolt listában vannak tárolva. Ez a tárolási mód a legegyszerűbb erre a feladatra, hiszen a kígyók mozgatásakor elég a lista elejéhez hozzácsatolni a kígyó fejének új helyét, a legutolsó elemet pedig levágni. A kígyók listájának első elemére mutató pointer a kígyó többi paraméterével együtt struktúrában van tárolva:

```
typedef struct snake{
    body* head;
    coord speed;
    int score;
    SDL_Color c;
} snake;
```

A *head* tartalmazza alista első elemét, a *score* a kígyó pontjait, azaz a kígyó hosszát, a C pedig a színét. A *speed* változó a kígyó haladásának irányát tartalmazza X és Y koordinátával.

```
typedef struct body{
    coord pos;
    struct body *next;
} body;
```

Az almának csak egy X és egy Y koordinátája van, ezért eltárolható egy egyszerű koordináta struktúrában. Az alma akkor számít *megevettnek* ha a koordinátái megegyeznek az egyik kígyó fejének koordinátáival.

```
typedef struct coord{
    int x;
    int y;
} coord;
```

### A játékot működtető függvények

- **void grow(snake sn, int n):**
  A paraméterként kapott kígyó méretét és pontját *n*-nel növeli. N-db elemet fűz a kígyó láncolt listájának végére.
- **void init(snake *sn, SDL_Color c):**
  Feltölt értékekkel egy üres kígyó struktúrát. Lefoglalja a láncolt lista első elemét és a koordinátáit random értékekre állítja. A kígyó sebessége 0 lesz x és z irányban is, színe pedig a kapott *c* paraméter.
- **void kill(snake *sn):**
  Felszabadítja a kígyó listájában lefoglalt *body* elemeket.
- **reset(snake *sn):**
  meghívja egy kigyóra a kill(…) majd az init(…) függvényt, azaz a kezdeti állapotba állítja azt.

- **void move(snake *sn):**
  Egy mezővel mozgatja előre a kígyót. A lista elejére beszúr egy elemet, a végéről levág egyet.
- **bool collision(snake sn1, snake sn2, winner *w, gamemode gm):**
  Ellenőrzi, hogy a kígyók beleütköztek-e valamibe. Ha igen, igaz értékkel tér vissza. A *w* értékét az ütközésnek megfelelően állítja be. Ha a játékmód *solo* volt, a *w* értéke *none* lesz. Ha *duo* volt, a *w* a nyertesnek megfelelő értéket kap.

```
typedef enum winner{
    greenw,
    bluew,
    none
} winner;
```

- **void check_food(snake *sn1, snake *sn2, coord *apple, gamemode gm):**
  Ellenőrzi, hogy a kígyók fejének koordinátája megegyezik-e az almáéval. Ha igen, az adott kígyóra meghívja a grow(…) függvényt, az alma koordinátáit pedig random értékekre állítja be.

```
typedef enum gamemode{
    solo,
    duo
} gamemode;
```

# A játék kirajzolása – draw_game.c modul

## A modul feladata

Ez a modul tartalmazza a játék kirajzolásához szükséges függvényeket. A core.c modul grafikus párja.

## A modul függvényei

- **static void draw_grid():** kirajzolja a tábla négyzetrácsát.
- **static void draw_apple(coord apple, SDL_Texture *img):**
  Kirajzolja az alma képét a koordinátáira.
- **static void draw_snake(snake sn):**
  Kirajzolja a paraméterként kapott kígyót.
- **static void show_score(snake sn, int x, int y, TTF_Font *font):**
  Kiírja a paraméterként kapott kígyó pontjait az *x* és *y* koordinátákra.
- **void draw_game(snake sn1, snake sn2, coord apple, TTF_Font *font, gamemode gm, SDL_Texture *apple_img):**
  A fenti függvények összecsomagolva egy nagyobb függvénybe, így elég ezt meghívni.
- **void draw_screen(SDL_Texture *screen):**
  A játék helyére a kapott texture-t rajzolja ki.

# A menü – sidebar.c modul

## A modul feladata

Ez a modul kezeli és rajzolja ki a menü gombjait és hajtja végre a gomboknak azokat a funkcióit, amik nem a *main*ben történnek. Ez a modul kezeli a dicsőséglista fájlból beolvasását és kiírását, illetve megjelenítését is.

## A menü gombjai

```
button **menu = (button**)malloc(10 * sizeof(button*));
menu[0] = create_button(825, 25, 100, 50, "start", false);
menu[1] = create_button(950, 25, 100, 50, "stop", false);
menu[2] = create_button(1075, 25, 100, 50, "reset", false);
menu[3] = create_button(825, 100, 350, 50, "leaderboard", false);
menu[4] = create_button(825, 175, 100, 50, "slow", false);
menu[5] = create_button(945, 175, 110, 50, "normal", false);
menu[6] = create_button(1075, 175, 100, 50, "fast", false);
menu[7] = create_button(825, 250, 165, 50, "solo", false);
menu[8] = create_button(1010, 250, 165, 50, "duo", false);
menu[9] = create_button(825, 325, 350, 50, "change name", false);
```

A fenti kód a build_menu() függvény, ami tíz gombot hoz létre a játék irányítására. A gombok feladataikat a main()-ben, SDL_MOUSEBUTTONDOWN eseményre hajtják végre. A gombok és feladatuk a kódnak megfelelő sorrendben:

```
typedef enum gamestate{
    startup,
    running,
    paused,
    finished,
    leaderboard
} gamestate;
```

- **start:** elindítja a játékot.
  a játékállapotot *running* értékre állítja
- **stop:** megállítja a játékot
  a játékállapotot *paused* értékre állítja
- **reset:** visszaállítja a játékot a kezdeti állapotba
  mindkét kígyóra meghívja a reset(…) függvényt.
  **leaderboard:** megjeleníti a dicsőséglistát
  a játékállapotot *leaderboard* értékre állítja
- **slow:** a játék sebességét lassúra állítja
  a *speed* változó értékét 3-ra állítja, azaz kígyók 12 tickenként fognak mozogni.
- **normal:** a játék sebességét közepesre állítja
  a *speed* változó értékét 3-ra állítja, azaz kígyók 6 tickenként fognak mozogni.
- **fast:** a játék sebességét gyorsra állítja
  a *speed* változó értékét 3-ra állítja, azaz kígyók 3 tickenként fognak mozogni.
- **solo:** egyjátékosra állítja a játékmódot
  a játékmódot *solo* értékre állíja
- **duo:** kétjátékosra állítja a játékmódot
  a játékmódot *duo* értékre állíja
- **change name:** megnyit egy szövegdobozt a név megadására
  meghívja az input_text(…) függvényt a *name* változóra

```
typedef enum gamemode{
    solo,
    duo
} gamemode;
```

## A modul függvényei

- **static button* create_button(int x1, int y1, int width, int height, char text[], bool clicked):**
  Lefoglal egy gomb típusú struktúrát, és visszaadja visszatérési értékként a címét.

```
typedef struct button{
    int xl;
    int yl;
    int width;
    int height;
    char text[20];
    bool clicked;
} button;
```

- **button** build_menu():**
  Ez a függvény hozza létre a gombokat és teszi bele őket egy dinamikus tömbbe, aminek címével vissza is tér. Pl.:
  Ez a függvény azért hasznos, mivel az általa készített tömbön végig lépkedve egy ciklussal kirajzolhatjuk az összes gombot.
- **bool button_check(button *bt, int mousex, int mousey, bool click):**
  Igaz értéket ad vissza, ha a kurzor x és y koordinátái rajta vannak a gombon és a bal egérgomb le van nyomva. Ennek megfelelően a gomb *clicked* változóját is beállítja.
  Ha a gomb lenyomott állapotban van, más színnel lesz kirajzolva.
- **static void draw_button(button *bt, TTF_Font *font):**
  Kirajzol egy gombot.
- **draw_menu(button **menu, TTF_Font *font, int mousex, int mousey, bool click):**
  Kirajzolja az egész menüt.

```
typedef struct highscore{
    char name[20];
    int score;
} highscore;
```

- **highscore** read_leaderboard(char filename[]):**
  A kapott fájlból struktúrába olvassa a benne levő adatokat, ezeket a struktúrákat pedig egy dinamikus tömbben helyezi el, aminek a címével tér vissza. Ha hibás adatot olvas, a név helyére „Placeholder”-t, pontnak nullát ír. Ezeket az adatokat a többi függvény ignorálja vagy átírja.
- **bool update_leaderboard(char filename[], highscore **ldr, snake sn, char name[]):**
  Ez a függvény akkor hívódik meg, amikor egyjátékos módban egy a játék véget ér. Ha a kígyó pontja ekkor nagyobb, mint a dicsőséglista tizedik elemének pontja, a játékos felkerül a listára, az új dicsőséglistát visszaírja a fájlba, a függvény pedig igaz értékkel tér vissza.
- **void destroy_menu(button **menu):**
  Felszabadítja a menü lefoglalt elemeit, azaz a gombokat.
- **void destroy_leaderboard(highscore **ldr):**
  Felszabadítja a dicsőséglista elemeit.
- **bool input_text(char *dest, size_t hossz, TTF_Font *font):**
  Megjelenít egy szövegdobozt, amibe írni lehet. A beírt szöveget a paraméterként kapott stringbe írja.

# A program menete – main.c

## A vezérlő változók

```
typedef enum gamestate{
    startup,
    running,
    paused,
    finished,
    leaderboard
} gamestate;

typedef enum gamemode{
    solo,
    duo
} gamemode;

typedef enum winner{
    greenw,
    bluew,
    none
} winner;
```

A program folyását három állapotváltozó irányítja. Ez a három a játékállapot, a játékmód és a győztes.
A játék elindításakor az játékállapot *startup* értéket kap. Amíg ez az érték tart, a játék helyén a *titlescreen* nevű kép van megjelenítve.

A *start* gombra kattintva a játékállapot *running* állapotba kerül. Ekkor a játék jelenik meg, irányíthatjuk a kígyóinkat.

Ha valamelyik kígyó ütközött, a játékállapot *finished* értéket vesz fel, és az ütközést vizsgáló függvény által meghatározott győzelmi értéknek megfelelő képernyőt fogunk látni. Egyjátékos módban ez mindig a *none* lesz, és a „you died” feliratot fogjuk látni. Játékmódot a *solo* vagy *duo* gombra kattintva válthatunk. Running állapotba a start gomb újbóli lenyomásával kerül vissza a program.

A *leaderboard* gombra kattintva *leaderboard* állapotot vesz fel a program. Ekkor a játék helyén a dicsőséglista lesz megjelenítve. A gomb újbóli megnyomása *running* módba vezet vissza.

A main végén szükség van a dinamikusan lefoglalt változók felszabadítására, illetve az ezt végző függvények meghívására, valamint a betöltött képek eldobására.

```
SDL_DestroyTexture(titlescreen);
SDL_DestroyTexture(deathscreen);
SDL_DestroyTexture(greenWin);
SDL_DestroyTexture(blueWin);
SDL_DestroyTexture(apple_img);
SDL_RemoveTimer(id);
TTF_CloseFont(font);
destroy_menu(menu);
destroy_leaderboard(ldr);
kill(&green);
kill(&blue);
```