Életjáték

programozói dokumentáció

A játék leírása

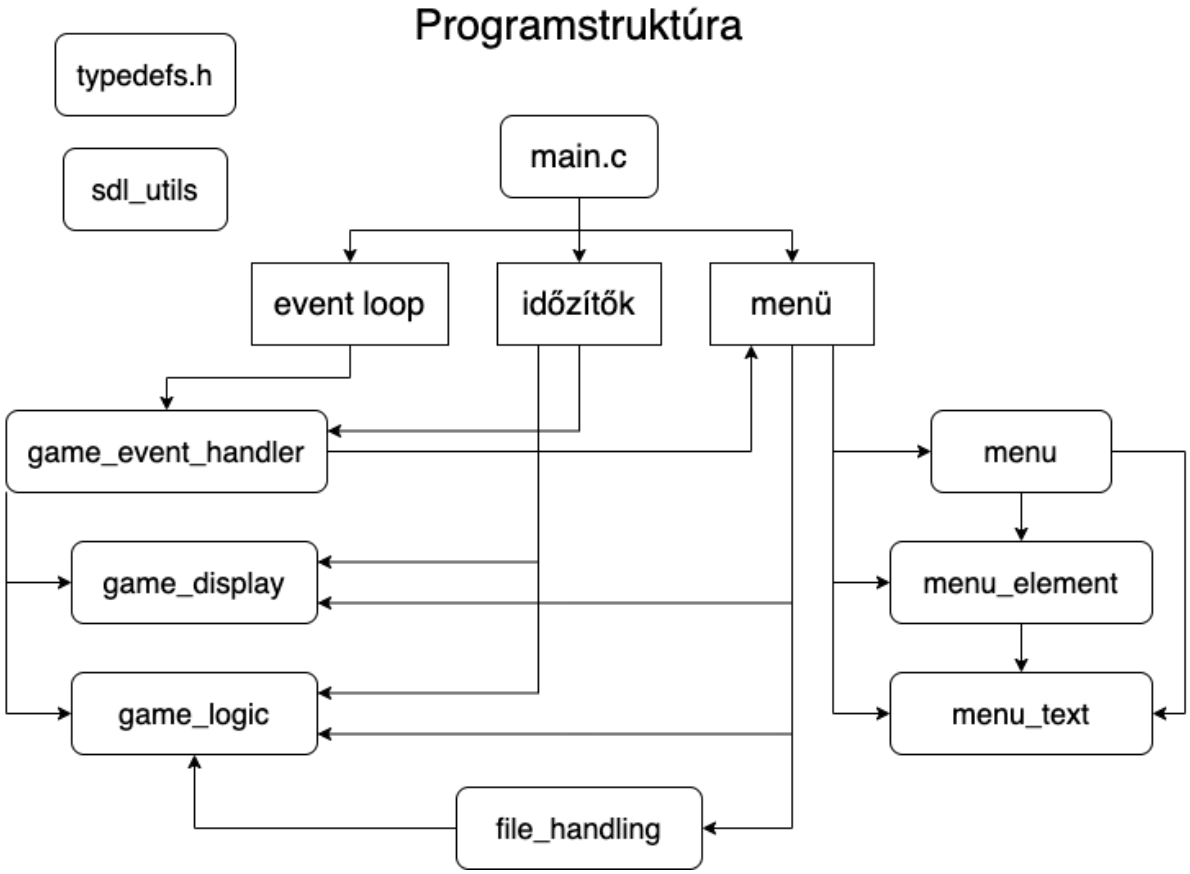
A program a Conway-féle életjátékot valósítja meg grafikus, menüvezérelt formában. A játéktér egy kétdimenziós négyzetháló, melyben egy-egy cella lehet élő, vagy halott. A játék egyetlen kezdeti állapotot („seed”) igényel és a következő állapot mindig kiszámolható a jelenlegi állapotból. A játéknak végtelen állapota létezik.

A kiszámítás abból áll, hogy cellánként meg kell vizsgálni az adott cella 8 szomszédos celláját, és szabályok alapján eldönteni, az adott cella milyen állapotot fog felvenni a következő lépésben. Ha a pálya szélén helyezkedik el a cella, akkor a pálya túlsó szélén levő cellák a szomszédok, ezt úgy kell elképzelni, mintha egy gömb felülete lenne síkra kiterítve.

Az egyes cellák következő állapotának megállapítása az alábbi szabályok szerint történik:

- Az élő cella meghal, ha kevesebb, mint 2 élő szomszédja van. (népességcsökkenés)
- Az élő cella életben marad, ha 2 vagy 3 élő szomszédja van.
- Az élő cella meghal, ha több, mint 3 élő szomszédja van. (túlnépesedés)
- A halott cella, ha pontosan 3 élő szomszédja van, élővé válik. (szaporodás)

A program felépítése



typedefs.h

A program által használt összes típusdefiníciót tartalmazza, enumokat és struktúrákat. Ezenkívül a programban sok helyen használt könyvtárak include-ja is itt szerepel ahelyett, hogy sok fájlban ismételve lennének ugyanazok az include-ok.

ENUM | CellState
Egy cella lehetséges állapotai: DEAD, LIVE

ENUM | MenuAction
Menüelemek által végezhető tevékenységek. Lényegében a megnyomott menüelem beazonosítására szolgál.

ENUM | MenuElementType
A menüelemek lehetséges típusai: BUTTON, TEXTFIELD (gomb vagy szövegbeviteli mező)

STRUCT | Vector2s
2 db **short** típusú számot (x, y) tárol.

STRUCT | Vector2d
2 db **double** típusú számot (x, y) tárol.

STRUCT | GameField
A játéktér celláinak kétdimenziós tömbjét tárolja (cells) illetve a hozzátartozó tömbméretet (size). Ezenkívül a 'newCells' tömb arra van, hogy a következő állapotot valahova lehessen írni számítás közben. A következő állapot kiszámolta után a 2 tömb (pointere) helyet cserél, így 'cells' mindig az éppen aktuális állapotra mutat

STRUCT | GridParams
A négyzetháló kirajzolásához szükséges paraméterek.

SDL_Rect gameArea: A teljes játéktér területe.

SDL_Rect gridArea: A négyzetháló által elfoglalt terület (gameArea – margók).

Vector2s borderWidth: A négyzetháló szegélyeinek szélessége/vastagsága.

Vector2s padding: A négyzetháló margója.

Vector2s cells: A cellák száma függőleges és vízszintes irányban.

Vector2d cellSize: Egy cella mérete.

SDL_Color deadColor: A halott cellák színe.

SDL_Color liveColor: Az élő cellák színe.

SDL_Color borderColor: A négyzetháló szegélyének színe.

SDL_Color bgColor: A játéktér háttérszíne.

STRUCT | MenuElementColors
Menüelemek színei.

SDL_Color edgeColor: A menüelem szélének a színe.

SDL_Color normalColor: A menüelem háttérszíne normál állapotban.

SDL_Color interactColor: A menüelem háttérszíne amikor a kurzor felette van, vagy le van nyomva.

SDL_Color selectColor: A menüelem háttérszíne, amikor ki van választva (szövegmezőnél használt).

STRUCT | Text
Egy megjelenítendő szöveg.

SDL_Rect area: A szöveg helye.

char *text: A szöveg nyers szövege.

TTF_Font *textFont: A szöveg betűtípusa.

SDL_Color textColor: A szöveg színe.

SDL_Texture *texture: A szöveg megrajzolt példánya, melyet közvetlenül ki lehet renderelni.

STRUCT | MenuElement
Egy menüelem.

MenuElementType type: A menüelem típusa.

MenuAction action: A menüelem művelete.

MenuElementColors colors: A menüelem megjelenítéséhez használt színek.

SDL_Rect area: A menüelem helye.

Text *text: A menüelemhez tartozó szöveg.

Uint8 interactAlpha: Szám, ami megmondja, hogy az interactColor éppen mennyire legyen áttetszően renderelve.

bool clicked: A menüelem éppen kattintva van-e.

STRUCT | Menu
Egy menü.

SDL_Rect area: A menü helye.

SDL_Color bgColor: A menü háttérszíne.

int textCount: A menüben levő szövegek száma.

Text **texts: A menüben levő szövegek.

int elementCount: A menüben levő menüelemek száma.

MenuElement **elements: A menüelemek.

MenuElement *foundElement: A jelenlegi kurzorpozíción található menüelem.

MenuElement *selTextField: A jelenleg kijelölt szövegmező.

STRUCT | Game
A játékhoz használt összes változót egybefogó struct.

SDL_Renderer *renderer: A renderer pointere.

SDL_Window *window: Az ablak pointere.

SDL_Rect windowArea: Az ablak területe (mérete).

GameField *gameField: A játéktér példány.

GridParams *gridParams: A négyzetháló paraméterek példány.

Menu *menu: A menü példány.

int simSpeedMs: A szimuláció jelenlegi sebessége.

bool simRunning: Fut-e jelenleg a szimuláció.

bool drawing: A felhasználó jelenleg rajzol-e.

CellState drawMode: A jelenlegi rajzolási mód (élő vagy halott).

SDL_Point cursorPos: A jelenlegi kurzorpozíció.

main.c

A program kiindulópontja.

Itt vannak az időzítők:

```
// Időzítő, mely FRAMETIME_MS-enként generál egy SDL_USEREVENT-et.
Uint32 render_tick(Uint32 ms, void *param);

// Változtatható gyorsaságú időzítő,
// mely a szimuláció következő iterációját számolja ki.
Uint32 sim_tick(Uint32 ms, void *param);
```

És itt vannak a program keretét adó függvények, melyek a main() függvényben a megfelelő sorrendben vannak hívva:

```
// Létrehozza a program futásához szükséges adatstruktúrákat, és
inicializálja az SDL-t.
Game init();

// Felépíti a menü elemeit, szövegeket, gombokat, szövegmezőket.
void build_menu(Game *game);

// Elindítja az időzítőket.
void start_timers(Game *game) ;

// Továbbítja a kapott event-et az event handler megfelelő függvényéhez.
void forward_event(Game *game, SDL_Event *event);

// Felszabadítja a program memóriafoglalásait.
void end(Game *game);
```

game_event_handler

A main.c forward_event() függvénye ennek a fájlnak a megfelelő függvényét hívja meg, hogy az éppen kapott event feldolgozása megtörténjen.

```
/**
 * Átméretezi a játékteret. (Csökkenti/növeli az oszlopainak vagy sorainak
a számát.)
 * @param game A játék példány.
 * @param action A menü művelet ami alapján meg lesz állapítva, milyen
átméretezésről van szó.
 */
void resize_game(Game *game, MenuAction action);

/**
 * Feldolgoz egy menüelem kattintást.
 * @param game A játék példány.
 * @param event Az event, ami előidézte a kattintást.
 * @param element A kattintott menüelem.
 */
void process_element_click(Game *game, SDL_Event *event, MenuElement
*element);

// SDL_MOUSEBUTTONDOWN event esetén meghívandó függvény.
void mouse_button_down(Game *game, SDL_Event *event);

// SDL_MOUSEBUTTONUP event esetén meghívandó függvény.
void mouse_button_up(Game *game, SDL_Event *event);

// SDL_MOUSEMOTION event esetén meghívandó függvény.
void mouse_motion(Game *game, SDL_Event *event);

// SDL_KEYDOWN event esetén meghívandó függvény.
void key_down(Game *game, SDL_Event *event);

// SDL_TEXTINPUT event esetén meghívandó függvény.
void text_input(Game *game, SDL_Event *event);

// SDL_USEREVENT event esetén meghívandó függvény.
void user_event(Game *game, SDL_Event *event);

// SDL_WINDOWEVENT event esetén meghívandó függvény.
void window_event(Game *game, SDL_Event *event);
```

game_display

A játékállapot megjelenítéséért felelős fájl.

```
/**
 * Létrehozza a grid paramétereket tartalmazó structot dinamikus
memóriában.
 * @param gameArea A játéktér mérete pixelben (margókkal együtt).
 * @param cells A cellák száma vízszintesen és függőlegesen.
 * @param deadColor A halott cellák színe.
 * @param liveColor Az élő cellák színe.
 * @param borderColor A szegély színe.
 * @param bgColor A háttérszín, a margónak a színe.
 * @return A létrehozott struct példány pointere, a hívó kötelessége
felszabadítani a free_grid_params() függvény hívásával.
 */
GridParams *create_grid_params(SDL_Rect gameArea, Vector2s cells, Uint32
deadColor, Uint32 liveColor, Uint32 borderColor, Uint32 bgColor);

/**
 * Újraszámolja a grid paraméterek struct értékeit a megadott új értékek
használatával.
 * @param params A grid paraméterek példány.
 * @param gameArea A játéktér mérete pixelben (margókkal együtt).
 * @param cells A cellák száma vízszintesen és függőlegesen.
 */
void resize_grid_params(GridParams *params, SDL_Rect gameArea, Vector2s
cells);

/**
 * Felszabadítja a memóriából a megadott grid paraméterek példányt.
 * @param params A grid paraméterek példány.
 */
void free_grid_params(GridParams *params);

/**
 * Kirajzolja a játéktér élő és halott celláit a megadott játéktér és grid
paraméterek alapján.
 * A renderert szükséges renderelésre meghívni a függvény visszatérte után.
 * @param renderer A renderer.
 * @param params A grid paraméterek.
 * @param field A játéktér.
 */
void draw_cells(SDL_Renderer *renderer, GridParams *params, GameField
*field);

/**
 * Egy négyzetrácsot rajzol a rendererbe a megadott grid paraméterekkel.
 * A renderert szükséges renderelésre meghívni a függvény visszatérte után.
 * @param renderer A renderer.
 * @param params A grid paraméterek, amikkel a négyzetrács pozícionálása és
színezése történik.
 */
void draw_grid(SDL_Renderer *renderer, GridParams *params);

/**
 * Kirajzolja a játéktér celláit és a köztük levő szegélyeket.
 * Hívása előtt szükséges törölni a játéktér korábbi tartalmát, és
 * a renderert szükséges renderelésre meghívni a függvény visszatérte után.
 * @param game A játék példány.
 */
void draw_game(Game *game);
```

game_logic

A játéktér adatstruktúra létrehozásáért, módosításáért és léptetéséért felelős fájl.

```
/**
 * A játéktér összes celláját halottra állítja.
 * @param field A játéktér.
 */
void clear_cells(GameField *field);

/**
 * Dinamikusan létrehoz egy 2D-s CellState tömböt.
 * @param sizeX A tömb vízszintes mérete.
 * @param sizeY A tömb függőleges mérete.
 * @return A létrehozott tömb pointere.
 */
CellState **create_2D_array(short sizeX, short sizeY);

/**
 * Létrehoz egy megadott méretű játékteret.
 * @param size A játéktér hány cella széles és magas legyen.
 * @return A létrehozott játéktér példány pointere, a hívó kötelessége
felszabadítani a free_field() függvény hívásával.
 */
GameField *create_field(Vector2s size);

/**
 * Átméretezi a megadott játékteret.
 * @param field A játéktér.
 * @param newSize Az új méret (cellák mennyisége mindkét irányban).
 */
void resize_field(GameField *field, Vector2s newSize);

/**
 * Felszabadítja a memóriából a megadott játékteret.
 * @param field A játéktér.
 */
void free_field(GameField *field);

/**
 * Kiszámítja, hogy a jelenlegi x és y kurzor pozición melyik cella van,
 * és megváltoztatja annak állapotát a megadott állapotra.
 * @param game A játék példány.
 */
void change_cell(Game *game);

/**
 * A játéktér egy cellájának lekérdezése átfordulással,
 * vagyis ha -1-et kérdezünk, akkor átugrik a pálya másik végére
 * és az lesz a lekérdezett cella, másik irányú túlindexelésnél szintúgy.
 * @param field A játéktér.
 * @param x A kérdezett cella x indexe. Lehet -1 vagy játéktér mérete
értékű is.
 * @param y A kérdezett cella y indexe. Lehet -1 vagy játéktér mérete
értékű is.
 * @return A lekérdezett cellaérték.
 */
CellState get_cell(GameField *field, int x, int y);
```

```
/**
 * Megszámolja, a megadott indexű cellának mennyi élő szomszédja van a 8-
ból.
 * @param field A játéktér.
 * @param x A kérdezett cella x indexe.
 * @param y A kérdezett cella y indexe.
 * @return Élő szomszédok száma. (min 0, max 8)
 */
int get_neighbor_count(GameField *field, int x, int y);

/**
 * A játéktér következő iterációját kiszámolja a newCells tömbben,
 * majd megcseréli a cells és newCells tömböket.
 * @param field A játéktér.
 */
void evolve(GameField *field);
```

menu

A menü adatstruktúra létrehozásáért, menüelemek kezeléséért és megjelenítéséért felelős fájl.

```
/**
 * Létrehoz egy Menu struct példányt a megfelelő kezdőadatokkal.
 * @param menuArea A menü területe az ablakon belül.
 * @param bgColor A menü háttérszíne.
 * @return A létrehozott struct példány pointere, a hívó kötelessége
felszabadítani a free_menu() függvény hívásával.
 */
Menu *create_menu(SDL_Rect menuArea, Uint32 bgColor);

/*
 * Kiszámolja a menüben levő menüelemek interact színéhez használt
átlátszóságot.
 * (Hover animáció)
 */
void calc_interact_alphas(Menu *menu);

/**
 * A rendererbe rajzolja a menüt.
 * A renderert szükséges renderelésre meghívni a függvény visszatérte után.
 * @param renderer A renderer.
 * @param menu A menü.
 */
void draw_menu(SDL_Renderer *renderer, Menu *menu);

/**
 * Felszabadítja a megadott menüt, és az összes benne levő menüelemet.
 * @param menu A menü.
 */
void free_menu(Menu *menu);

/**
 * Hozzáadja a megadott szöveget a menühöz.
 * @param menu A menü.
 * @param text A szöveg.
 */
void add_text(Menu *menu, Text *text);

/**
 * Hozzáadja a megadott menüelemet és a hozzá tartozó szöveget a menühöz.
 * @param menu A menü.
 * @param element A menüelem.
 */
void add_element(Menu *menu, MenuElement *element);

/**
 * Megkeresi az első menüelemet, ami a megadott képernyőkoordinátákon van
 * és a Menu struct foundElement értékét átállítja a talált elem
pointerére,
 * vagy NULL-ra, ha nincs találat.
 * @param menu A menü.
 * @param point A képernyőkoordináták.
 * @return A megtalált menüelem pointere, vagy NULL, ha nincs találat.
 */
MenuElement *find_element(Menu *menu, SDL_Point point);
```

menu_element

Menüelemek (gombok, szövegmezők) létrehozásáért és megjelenítéséért felelős fájl.

```
/**
 * Létrehoz egy dinamikusan foglalt gombot a megadott paraméterekkel.
 * @param renderer A renderer.
 * @param area A gomb helye a menühöz relatívan.
 * @param action A gomb milyen műveletet végez kattintáskor.
 * @param text A gomb szövege.
 * @param textFont A gomb szövegének betűtípusa.
 * @param textColor A gomb szövegének színe.
 * @param colors A gomb színei
 * @return A gombra mutató pointer, a hívó kötelessége felszabadítani a
free_menu_element() függvény hívásával.
 */
MenuElement *create_button(SDL_Renderer *renderer, SDL_Rect area,
MenuAction action, char *text, TTF_Font *textFont, Uint32 textColor,
MenuElementColors colors);

/**
 * Létrehoz egy dinamikusan foglalt szövegmezőt a megadott paraméterekkel.
 * @param renderer A renderer.
 * @param area A szövegmező helye a menühöz relatívan.
 * @param action A szövegmező milyen műveletet végez kattintáskor.
 * @param text A szövegmező szövege.
 * @param textFont A szövegmező szövegének betűtípusa.
 * @param textColor A szövegmező szövegének színe.
 * @param colors A szövegmező színei
 * @return A szövegmezőre mutató pointer, a hívó kötelessége felszabadítani
a free_menu_element() függvény hívásával.
 */
MenuElement *create_text_field(SDL_Renderer *renderer, SDL_Rect area,
MenuAction action, char *text, TTF_Font *textFont, Uint32 textColor,
MenuElementColors colors);

/**
 * Szerkeszti egy menüelem szövegét. A betűtípus, a betűméret és a szöveg
színe nem változik.
 * @param renderer A renderer.
 * @param element A szerkesztendő menüelem.
 * @param newText Az új szöveg.
 */
void edit_element_text(SDL_Renderer *renderer, MenuElement *element, char
*newText);

/**
 * A rendererbe rajzolja a megadott menüelemet.
 * @param renderer A renderer.
 * @param element A menüelem.
 * @param offset A menüelem területéhez képest mennyivel legyen eltolva a
menüelem helye.
 */
void draw_element(SDL_Renderer *renderer, MenuElement *element, Vector2s
offset);

/**
 * Felszabadítja a megadott menüelemet.
 * @param element A menüelem.
 */
void free_element(MenuElement *element);
```

```
/**
 * Beállítja a 'colors' struct értékeit a megadott értékekre.
 * @param edgeColor A menüelem szélének a színe.
 * @param normalColor A menüelem háttérszíne normál állapotban.
 * @param interactColor A menüelem háttérszíne amikor a kurzor felette van,
vagy meg van nyomva.
 * @param selectColor A menüelem háttérszíne, amikor ki van választva
(szövegmezőnél használt).
 * @param colors A struct, aminek az értékei változni fognak.
 */
void set_menu_element_colors(Uint32 edgeColor, Uint32 normalColor, Uint32
interactColor, Uint32 selectColor, MenuElementColors *colors);
```

menu_text

Szövegek létrehozásáért és megjelenítéséért felelős fájl.

```
/**
 * Létrehoz egy dinamikusan foglalt szövegelemet a megadott paraméterekkel.
 * @param renderer A renderer.
 * @param area A szöveg helye a menühöz relatívan.
 * @param text A megjelenítendő szöveg.
 * @param textFont A szöveg betűtípusa.
 * @param textColor A szöveg színe.
 * @return A szövegelemre mutató pointer, a hívó kötelessége felszabadítani
a free_text() függvény hívásával.
 */
Text *create_text(SDL_Renderer *renderer, SDL_Rect area, char *text,
TTF_Font *textFont, Uint32 textColor);

/**
 * Szerkeszti egy szövegelem szövegét. A betűtípus, a betűméret és a szöveg
színe nem változik.
 * @param renderer A renderer.
 * @param text A szerkesztendő szövegelem.
 * @param area A szöveg helye a menühöz relatívan.
 * @param newText Az új szöveg.
 */
void edit_text(SDL_Renderer *renderer, Text *text, SDL_Rect area, char
*newText);

/**
 * A rendererbe rajzolja a megadott szövegelemet.
 * @param renderer A renderer.
 * @param text A szövegelem.
 * @param offset A szövegelem területéhez képest mennyivel legyen eltolva a
szövegelem helye.
 */
void draw_text(SDL_Renderer *renderer, Text *text, Vector2s offset);

/**
 * Felszabadítja a megadott szövegelemet.
 * @param text A szövegelem.
 */
void free_text(Text *text);
```

sdl_utils

Megjelenítést és SDL-kezelést segítő segédfüggvények.

```
/**
 * Inicializálja az SDL-t és létrehoz egy ablakot és annak a rendererjét.
 * @param width Ablak szélessége (pixel).
 * @param height Ablak magassága (pixel).
 * @param title Ablak neve.
 * @param pwindow A létrehozott ablak pointere milyen memóriacímre legyen
írva.
 * @param prenderer A létrehozott renderer pointere milyen memóriacímre
legyen írva.
 */
void sdl_init(int width, int height, char *title, SDL_Window **pwindow,
SDL_Renderer **prenderer);

/**
 * Megsemmisíti a megadott ablakot és renderert, majd meghívja az
SDL_Quit() függvényt.
 * @param pwindow Az ablak pointere.
 * @param prenderer A renderer pointere.
 */
void sdl_exit(SDL_Window **pwindow, SDL_Renderer **prenderer);

/**
 * A megadott 32 bites szám R, G, B és A értékeit beleírja a cím szerint
megadott 'color' struct-ba.
 * @param value Egy 32 bites szám, melynek bájtjai rendre az R, G, B, A
értékeket határozzák meg.
 * @param color SDL_Color struct példányra mutató pointer, aminek az
értékei át lesznek állítva.
 */
void set_color(Uint32 value, SDL_Color *color);

/**
 * A megadott értékeket beleírja a cím szerint megadott 'rect' struct-ba.
 * @param rect SDL_Rect struct példányra mutató pointer, aminek az értékei
át lesznek állítva.
 */
void set_rect(int x, int y, int w, int h, SDL_Rect *rect);

/**
 * Létrehoz egy fontot a megadott betűtípus elérési út és betűméret
alapján.
 * Ezen függvény hívása előtt a TTF_Init() mindenképpen történjen meg!
 * @param fontPath A betűtípus fájl elérési útja.
 * @param fontSize A betűtípus milyen mérettel kerüljön betöltésre.
 * @return A létrehozott font, a hívó kötelessége felszabadítani a
TTF_CloseFont(); függvény hívásával.
 */
TTF_Font *create_font(char *fontPath, int fontSize);

/**
 * Kitölti a megadott területet a megadott színnel.
 * A renderert szükséges renderelésre meghívni a függvény visszatérte után.
 * @param renderer A renderer.
 * @param area A terület, amit át szeretnénk színezni.
 * @param color A színezéshez használt szín.
 */
void fill_rect(SDL_Renderer *renderer, SDL_Rect *area, SDL_Color color);
```

```
/**
 * Kitölti a megadott területet a megadott színnel.
 * A renderert szükséges renderelésre meghívni a függvény visszatérte után.
 * @param renderer A renderer.
 * @param area A terület, amit át szeretnénk színezni.
 * @param color A színezéshez használt szín.
 * @param offset A színezni kívánt terület mennyivel legyen eltolva.
 */
void fill_rect_offset(SDL_Renderer *renderer, SDL_Rect *area, SDL_Color
color, Vector2s offset);

/**
 * Megmondja, hogy a 'point' rajta van-e az 'offset'-tel eltolt 'rect'
területen.
 * @return Igaz, ha benne van, hamis, ha nincs benne.
 */
bool inside_rect(SDL_Rect *rect, SDL_Point *point, SDL_Rect *offset);
```

A program építése

A programot Mac OS X operációs rendszeren, CLion fejlesztőkörnyezettel fejlesztettem. Építéshez a cmake nevű szoftvert használom, amivel a CLion alapértelmezetten épít. A standard C könyvtárakon kívül használt könyvtárak: SDL2, SDL2_gfx és SDL2_ttf. A könyvtárak a gépen való megtalálására egy github repóból származó cmake fájlokat használok.

Az építéshez szükséges lépések

1. Homebrew-el telepíteni az sdl2, sdl2_gfx és sdl2_ttf könyvtárakat.
   > brew install sdl2
   > brew install sdl2_gfx
   > brew install sdl2_ttf
2. CLion-ba importálni a program fájljait, mely automatikusan bekonfigurálja a CMake-et.
3. A fent említett github repót klónozni az újonnan létrejött projekt „cmake-build-debug” mappájába, azon belül egy „sdl2” mappába.
4. A „CMakeLists.txt”-ben megmondani a CMake-nek, hogyan találja meg az SDL2 könyvtárakat:

```
cmake_minimum_required(VERSION 3.20)
project(eletjatek C)

set(CMAKE_C_STANDARD 99)

list(APPEND CMAKE_MODULE_PATH ${CMAKE_CURRENT_SOURCE_DIR}/cmake-
build-debug/sdl2)

add_executable(eletjatek {PROJECT_SOURCE_DIR}/ ...)

target_include_directories(${PROJECT_NAME} PRIVATE include)

# Add SDL2 library
find_package(SDL2 REQUIRED)
target_link_libraries(${PROJECT_NAME} SDL2::Main)

# Add SDL2_ttf library
find_package(SDL2_ttf REQUIRED)
target_link_libraries(${PROJECT_NAME} SDL2::TTF)

# Add SDL2_gfx library
find_package(SDL2_gfx REQUIRED)
target_link_libraries(${PROJECT_NAME} SDL2::GFX)
```

5. CLion használatával megépíteni a projektet. (Build -> Build Project)