***MineSweeper fejlesztői dokumentáció***

A program 2 részre bontható. Az egyik rész a User Interface (*main.c*), amely kezeli a felhasználói grafikai megjelenítést. A másik a játék logika (*game.c*), amely lényegében a játék agyát adja és kezeli az összes folyamatot, amely felelős azért, hogy a játék az elvárások szerint működjön.

***Adatszerkezetek:***

typedef struct Game  
{  
 GameMode mode;  
 Field field;  
} Game;

A **Game** struktúra szerepe, hogy a felhasználó által megadott *nehézségi fokozatot* és *páya méretet* szimbolizálja. Ezeket felsorolt típusok segítségével teszi meg. Ezek mind számokat jelölnek, csak a könnyebb használat miatt van így kezelve.

typedef struct Cell  
{  
 CellType type;  
 bool shown;  
 bool marked;  
} Cell;

A **Cell** struktúra szerepe, hogy a minden egyes mezőt egyesével szimbolizáljon. Ez teszi lehetővé, hogy a játék folyamán egyesével, a felhasználó kattintása alapján lehessen változtatni az adott mezőt. A *CellType* a mező típusa, az alapján, hogy bomba vagy, hogy milyen közel van a bombához. A másik kettő tulajdonság azt jelöli, hogy a mezőre már *kattintott* e a felhasználó, iletve, esetleg *bejelölte* e.

***Legfontosabb függvények:***

Logikai függvények (game.c):

void new\_game(Game \*game, GameMode mode, Field field)

Ez a függvény inicializálja az adott játékmenetet. A paraméterként kapott *game*-et állítja be, a szintén paraméterként kapott *játékmód* és *páya nagyság* alapján.

Cell \*\*setup\_cells(Game \*game)

A paraméterként kapott *game* alapján dinamikusan lefoglalja a megfelelő méretű memóriát a mezők számára, amelyek 2D tömbként vannak kezelve, majd ezeket inicializálja is az alap mező beállításokkal. Erre a *lefoglalt területre mutató pointerrel tér vissza*.

void show(Cell \*\*\*c, int x, int y)

void mark(Cell \*\*\*c, int x, int y)

Ez a kettő függvény csupán annyit tesz, hogy a paraméterként kapott *2D tömb pointer*ének az *x.* sorú és *y.* oszlopú elemének (ami ugye egy mező) a megfelelő struktúra tulajdonságát (shown vagy marked) igazra állítja.

UI függvények (main.c):

static void setup\_ui(Game \*game, Cell \*\*cells)

Ez a fő függvény, ami a megjelenítésért felel és ez kezeli a többi függvényt. Paraméterként kapja a *game*-et és a *mezők címeire mutató pointert* (későbbiekben ez fog mutatni a dinamikusan foglalt memória területre).

static void game\_view(SDL\_Window \*window, SDL\_Renderer \*\*prenderer, SDL\_Texture \*background)

static void menu\_view(SDL\_Window \*window, SDL\_Renderer \*\*prenderer, SDL\_Texture \*background)

Ezek a függvények jelenítik meg a menüben és a játék alatt a háttérképet. Paraméterként az SDL változóit kapja (*ablak*, *renderer*), illetve a *háttérképre mutató pointer*t.

static void detect\_menu\_click(SDL\_Event ev, Game \*game, SDL\_Texture \*background, SDL\_Renderer \*renderer, bool \*menu\_on)

static void detect\_game\_click(SDL\_Renderer \*renderer, SDL\_Event ev, Game \*game, Cell \*\*cells, double field\_start\_pixel\_x, double field\_start\_pixel\_y, double cell\_size, SDL\_Texture \*cell\_img)

A menüben és a játék alatti egér kattintásokat kezelik ezek a függvények. Ezek hívják meg a legtöbb logika függvényt. Paramétereik többek között azonosak az előző függvényekével. Kiegészülnek pl. a *mező pixel beállításai*val(detect\_game\_click).

static void render\_field(SDL\_Renderer \*renderer, Game \*game, Cell \*\*cells, double cell\_size, double field\_start\_pixel\_x, double field\_start\_pixel\_y, SDL\_Texture \*cell\_img)

Ez a függvény felel a játékmező frissítéséért. Minden kattintás után ez is meghívódik és a frissített mezők alapján tölti be a megfelelő mező képet. Paraméterek a szokványosakon kívül a *mező pixel beállítás*ok.