***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.

Status STATUS = *ingame*;

Ez a globális változó határozza meg a játék állapotát. Ez teszi lehetővé, hogy egyszerűen minden felhasználói beavatkozásnál ellenőrizve legyen, hogy a játék *folyamatban* van, esetleg *elvesztett*, vagy *megnyert*.

clock\_t start\_time, loaded\_time, game\_time;

Ezek a globális változók mind a játékban lévő számlálóhoz kellenek. Ezek segítenek a processzor idő alapján meghatározni, hogy mennyi idő telt el a játék indítása óta. Ehhez el kell tárolni az *indítási idő*t és (amennyiben van) a *mentésből kinyert korábbi idő*t.

***Legfontosabb függvények:***

Játék 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 set\_bombs(Game \*game, Cell \*\*\*cells)

Ez a függvény felelős a bombák elhelyezéséért (random generálással) és a szomszédos mezők megfelelő számmal való ellátásáért (ennek a lényegi részét egy másik függvény végzi). Paraméterként a *mezők 2D tömbjének pointer*ét és a *game pointer*t kapja.

void show(Game \*game, Cell \*\*\*cells, int x, int y)

void mark(Cell \*\*\*cells, 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, illetve az első függvény esetében rekurzív módon a szomszédos mezőkkel is ezt teszi.

void save(Game \*game, Cell \*\*\*cells)

bool load(Game \*game, Cell \*\*\*cells)

Ez a kettő függvény felelős a játék állapotának mentéséért és betöltéséért. Paraméterként a *mezők 2D tömbjének pointer*ét és a *game pointer*t kapja. Az első függvénynél fájlba írás, a másiknál fájlból olvasás történik. A fájlból olvasás előtt ellenőrizve van, hogy egyáltalán létezik e a mentés (ezért *tér vissza egy logikai változó*val).

void free\_memory(Cell \*\*cells, Game \*game)

Ez a függvény szabadítja fel mezőknek foglalt memóriát. Adott játék befejezésekor van meghívva. Paraméterként a *mezők 2D tömbjének pointer*ét és a *game pointer*t kapja.

void set\_time(double t)

void update\_time()

double get\_time()

Ezek a függvények a számlálót kezelik. Mindegyik a processzor idő alapján dolgozik és elmentik / frissítik / kikérik adott pillanatban az időt. Felhasználói beavatkozásnál és mentésnél / betöltésnél hívodnak meg. A már korábban bemutatott globális változókkal dolgoznak.

void set\_status(Status type)

Status get\_status()

Ezek a függvények a játék állapotát kontrollálják. Az előbbi frissíti a korábban bemutatott globális változót, az utóbbival pedig a változó értéke kérhető ki. Ezek a függvények vannak meghívva minden felhasználói beavatkozásnál.

UI függvények (main.c):

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).

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

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

void result\_view(SDL\_Renderer \*renderer, SDL\_Texture \*result\_background)

Ezek a függvények jelenítik meg a menüben és a játék alatt a háttérképet, illetve a játék végekor a megfelelő befejező képet. Paraméterként az SDL változóit kapják (*ablak*, *renderer*), illetve a *képre mutató pointer*t.

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

void detect\_game\_click(SDL\_Renderer \*renderer, SDL\_Event ev, Game \*game, Cell \*\*cells, FieldPixelSetting \*fpd, SDL\_Texture \*cell\_img)

A menüben és a játék alatti egér kattintásokat kezelik ezek a függvények. Ezek hívják meg a legtöbb játék logikai függvényt. Paramétereik többek között azonosak az előző függvényekével. Kiegészülnek pl. a *játék nehézségét tároló változó*kkal vagy a *mező pixel beállításai*val.

void render\_field(SDL\_Renderer \*renderer, Game \*game, Cell \*\*cells, FieldPixelSetting \*fpd, 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 képeket. Ebben a függvényben történik a lefedett mezők számlálása is, amely a játék végének észleléséhez fontos (innentől már egy játék logikai függvény kezeli). Paraméterek a szokványosakon kívül a *mező pixel beállítás*ok.

void render\_clock(SDL\_Renderer \*renderer, SDL\_Surface \*clock, SDL\_Texture \*clock\_t)

Ez a függvény a számlálót frissíti folyamatosan. Az adott időt a függvényen belül egy függvényhívással kapja meg.

void destroy\_sdl(SDL\_Renderer \*renderer, SDL\_Window \*window, SDL\_Texture \*background, SDL\_Texture \*cell\_img, SDL\_Texture \*result\_background, SDL\_Surface \*clock, SDL\_Texture \*clock\_t)

Ez a függvény a játék bezárásakor (vagy újrakezdésekor) hivatott felszabadítani az SDL-es elemeknek foglalt területeket. Paraméterként kapja az összes *SDL-es elem*et.