Aknakereső - Programozói Dokumentáció

Az aknakereső játék

Az aknakereső lényegében egy egyszerű logikai játék, amely egy aknamezőn játszódik, a játékos célja pedig a mezőn található aknák felderítése. Ebben több dolog segíti: megjelölheti az általa aknának vélt mezőket, megtudhatja a már felderített mezők szomszédságában levő aknák számát, továbbá ha üres mezőt talál, akkor az azzal összefüggő további üres mezők is fel lesznek derítve.

A program az aknakereső játék szabályrendszere alapján működik, amely a következő fő pontokból áll: - A játék addig tart, amíg minden egyes aknát nem tartalmazó mező fel nincs fedve. - A játéknak vége, ha a játékos aknát tartalmazó mezőre lép, vagy a beállított időkorlát lejár. - A játékos által elsőként kiválasztott mezőn, valamint annak szomszédságában nem lehet akna. - A mező kiválasztása az egér gombjának felengedésével válik véglegessé. - Az egér jobb gombjával a mező megjelölhető zászlóval, vagy kérdőjellel. Utóbbi ritkán használt.

A játékban előfordulhatnak olyan szituációk, amelyeket pusztán logikai úton nem lehet megoldani. Ezeket az eseteket a játék nem figyeli, ilyenkor sajnos a szerencsére van bízva a játék kimenetele.

A projekt felépítése

A projekt három fő modulból épül fel: a főmodulból, grafikai modulból és játékmodulból. Ezekhez külön forrás- és fejlécfájlok tartoznak. Moduláris felépítése miatt a projekt könnyen átlátható.

Az alábbiakban a projekt fájljainak neve és helye:

```
minesweeper/
│
└──CMakeLists.txt
│
└──src/
│   │──main.c
│   │──graphics.c
│   │──leaderboard.c
│   │──game.c
│   │
│   └──headers/
│       │──graphics.h
│       │──leaderboard.h
│       └──game.h
│
└──docs/
│   │──specification_hu.pdf
│   │──developer_documentation_hu.pdf
│   │──developer_documentation_hu.md
│   │──user_documentation_hu.pdf
│   └──user_documentation_hu.md
│
└──build/
```

```
    |——faces.png
    |——fields.png
    |——mine_numbers.png
    |——numbers.png
    └——(results.csv)
```

A program felépítéséhez az `src/` mappa fájljai szükségesek, a futtatásához viszont a `build/` mappában található `.png` kiterjesztésű képfájlok is, ugyanis ezek nélkül a játék grafikája nem fog működni.

A projekt gyökérkönyvtárában található `CMakeLists.txt` fájl segíthet a projekt build-elésében. (CMake használatával)

A program függvényei ebben a dokumentációban, valamint a forrásfájlokban is dokumentálva vannak. A modulok privát függvényei a `.c` fájlokban, a publikus függvényei pedig a `.h` fájlokban vannak dokumentálva.

Főmodul

A program fő modulja a `main.c` forrásfájlban van implementálva. A feladata a program elindítása, a szükséges erőforrások inicializálása, a játék paramétereinek bekérése. Tartalmazza a játék alapértelmezett paramétereit is, amennyiben a felhasználó nem adna meg mást.

Egyetlen függvénye a `main`, mely a program belépési pontjának felel meg. A fent írt feladatokat hajtja végre.

Grafikai modul

A grafikai modul tartalmaz minden megjelenítéssel kapcsolatos függvényt, változót és konstanst. Ehhez a modulhoz a `graphics.c` és `graphics.h` fájlok tartoznak.

Típusok

GraphicsAssets

Tartalmazza a megjelenítéshez szükséges grafikai erőforrásokat (ún. asseteket). Ezek közé tartoznak a - grafikai elemek (tileset) - betűtípusok - színek

Ezek az assetek egyszer kerülnek inicializálásra az `init_game_window([...])` függvényben.

```
typedef struct GraphicsAssets {
    // Az arc különböző állapotai
    SDL_Texture *face_tiles;

    // A mezők lehetséges textúrái
    SDL_Texture *field_tiles;

    // A számláló számjainak textúrái
```

```
    SDL_Texture *number_tiles;

    // A szomszédos aknák számát jelző mezők textúrái
    SDL_Texture *mine_number_tiles;

    // Betűtípusok
    TTF_Font *font_small;
    TTF_Font *font_large;

    // Betűszín (fekete)
    SDL_Color black;
} GraphicsAssets;
```

Fontosabb belső globális változók

Ezek a változók a modul minden egyes függvényéhez szükségesek; a modul inicializálófüggvényének (`init_game_window`) hívása által kapnak értéket. A modulon kívül nem elérhetőek.

`static SDL_Window *window`

A játékablak.

`static SDL_Renderer *renderer`

A grafikai elemek kirajzolásához szükséges változó.

Fontosabb függvények:

`void init_game_window(int rows, int cols)`:

Inicializálja a játékablakot. Kiszámítja az ablak méreteit, inicializálja az `SDL` könyvtárat, létrehozza az ablakot és a renderert. Kirajzolja az alapértelmezet hátteret. Betölti a grafikához szükséges erőforrásokat (képfájlokat).

`void draw_face(FaceType face)`:

Kirajzolja a játékablak felső részében található arcot, amely a játék során különféle arckifejezéseket vesz fel, valamint rá kattintva kezdhető újra a játék.

`void draw_fields(FieldSet fieldSet)`:

Kirajzolja a játékmezőket a játéktáblára. Mindegyiket az adatforrásnak (fieldSet) megfelelő állapotban.

`void draw_mine_counter(int mine_count)`:

Kirajzolja a felderítendő aknák számát (paraméterben kapja) a játékablak bal felső sarkába.

`void draw_time_counter(int seconds)`:

Kirajzolja a játék kezdetétől eltelt másodperceket (paraméterben kapja) a játékablak jobb felső

sarkába

`void refresh_screen()`:

Frissíti a játékablakot.

`void draw_leaderboard(Result results[], int length)`

Kirajzolja az eredménytáblát a játékterület helyére.

Fontosabb segédfüggvények

`void position_to_field_coordinates(int posX, int posY, int rows, int cols, int *coordX, int *coordY)`:

Ez a függvény egy grafikai pozíciót (`posX`, `posY`) egy adott méretű játéktáblára értemezve (`rows`, `cols`) mező-koordinátákká konvertál: `coordX`, `coordY`. A függvény az utols két paraméterét cím szerint veszi át és az eredmény értékét fogja beléjük írni. Amennyiben a kapott grafikai pozíció nincs a játékmezőben, ez esetben `coordX` és `coordY` értéke `-1` lesz.

`bool is_in_range(int x, int y, int minX, int maxX, int minY, int maxY)`:

Ellenőrzi, hogy egy adott pont (`x`, `y`) benne van-e a többi paraméter által határolt téglalapban.

`bool is_position_on_face(int posX, int posY)`:

Ellenőrzi, hogy az adott grafikai pozíció a játékablakon található arcon van-e.

`bool is_position_on_button(int posX, int posY)`

Ellenőrzi, hogy az adott grafikai pozíció az eredménytábla / játék módválasztó gombon van-e.

Játékmodul

Ebben a modulban található a játék logikájához szükséges összes függvény, változó, konstans. A `game.c` és `game.h` fájlok alkotják ezt a modult.

Típusok

Mező (Field)

Egy mezőt jelent.

```
typedef struct Field {
    FieldType type;
    bool is_mine;
    int x;
    int y;
    int neighbour_mines_count;
} Field;
```

A `type` nevű tulajdonság a FieldType valamely lehetséges értékét veszi fel, melyek lentebb láthatók.
Az `is_mine` tulajdonság igaz értéke esetén a mezőt aknaként jelöli meg.
Az `x` és `y` tulajdonságok a mező koordinátáit tárolják.
A `neighbour_mines_count` pedig a mezővel szomszédos aknák számát tárolja.

```
typedef enum FieldType {
    FIELD_STANDARD,
    FIELD_EMPTY,
    FIELD_FLAGGED,
    FIELD_QUESTION,
    FIELD_QUESTION_PRESSED,
    FIELD_MINE_REVEALED,
    FIELD_MINE_EXPLODED,
    FIELD_MINE_CROSSED,
    FIELD_NUMBERED
} FieldType;
```

Aknamező (FieldSet)

Mezők összességét jelenti, melyek együtt egy teljes aknamezőt alkotnak.

```
typedef struct FieldSet {
    Field **fields;
    int rows;
    int cols;
} FieldSet;
```

A `fields` tulajdonság egy mezőket (Field) tartóalmazó kétdimenziós tömb.
A `rows`, valamint a `cols` az aknamező (és a `fields` 2D tömb) méretét határozza meg.

Arc (FaceType)

Definiálja az egyes arctípusokat:

```
typedef enum FaceType {
    FACE_SMILING,
    FACE_SMILING_PRESSED,
    FACE_THREATENED,
    FACE_SUNGLASSES,
    FACE_DEAD
} FaceType;
```

Játékparaméterek (GameParams)

Tartalmazza a játék felhasználó által (is) megadható paramétereit.

```
typedef struct GameParams {
```

```
    char *player_name;
    int cols;
    int rows;
    int mine_count;
    int time_limit;
} GameParams;
```

Játékállapot (GameState)

Tartalmazza a játék mindenkori állapotához szükséges változókat, melyek a program viselkedését, reakcióját befolyásolják. Az eltelt másodperceket, megtalált aknákat, hogy folyik-e a játék éppen, az eredményjelző meg van-e nyitva, hogy melyik mezőt, vagy esetleg az arcot nyomtuk-e meg.

```
typedef struct GameState {
    int mines_found;
    int time_elapsed;
    bool game_started;
    bool game_finished;
    bool leaderboard_opened;
    bool face_pressed;
    Field *pressed_field;
} GameState;
```

Játék (Game)

Egybefoglal mindent, ami a játék (logikájához) szükséges, a fent definiált strukturák egy-egy példányát.

```
typedef struct Game {
    FieldSet fieldSet;
    GameParams params;
    GameState state;
} Game;
```

Fontosabb belső globális változók

Az alábbi változók csak a modulon belül globálisak, kívülről, más modulokból nem elérhetőek.

`static SDL_TimerID timer`

Időzítő, amely a játék kezdete óta eltelt másodperceket számolja.

Fontosabb függvények

`void start_game(GameParams params)`

Elindítja a játékot, és az ahhoz tartozó event loopot. Inicializálja a játékhoz szükséges összes változót, és addig nem tér vissza, amíg a játékos ki nem lép a programból.

void reset_game(GameParams params)

A játék összes paraméterét a kezdeti állapotba állítja a kapott paramétereknek megfelelően. Inicializálja a játékmezőket (és amennyiben volt, a korábbi aknamezőt felszabadítja), az aknák számát visszaállítja, valamint az eltelt másodperceket nullázza. Újrarajzoltatja a játékterületet.

void handle_mouse_down(Uint8 button, int x, int y, Game *game)

Feldolgozza az egérgomb lenyomását.

Bal egérgomb lenyomása esetén több eset lehet: - amennyiben mezőre kattintott, megjelöli, hogy melyik mező fölött nyomtuk meg az egérgombot. - Ha az arcra kattintott, akkor megjegyzi, hogy az arcra kattintott a felhasználó. - Ha az eredményjelző / játék gombra kattintott, akkor megjelenik az eredményjelző, ha pedig már meg volt nyitva, akkor megjelenik a játéktábla és egy új játék kezdődik.

Jobb egérgomb lenyomása esetén, ha az mezőn történt, akkor megjelöli a mezőt zászlóval, ha már van rajta zászló, akkor kérdőjellel, ha pedig kérdőjel van rajta, akkor törli a jelölést. Zászlóval való jelölés esetén a bal felső sarokban látható aknák száma csökken eggyel. Ha töröljük a zászlót, akkor visszanő.

void handle_mouse_up(Uint8 button, int x, int y, Game *game)

Feldolgozza az egérgomb felengedését

Bal egérgomb lenyomása esetén amennyiben mező fölött engedte fel a gombot, a szabályoknak megfelelő műveletet hajtja végre. Ha aknára kattintott, vége a játéknak. Ha üres mezőre, akkor felfedi azt, valamint a szomszédos üres mezőket is, továbbá kijelzi a szomszédos aknák számát. Ha az arc felett engedte el az egeret, akkor új játék kezdődik a jelenlegivel megegyező paraméterekkel.

void free_field_memory(FieldSet *fieldSet):

Felszabadítja az aknamező által elfoglalt memóriaterületet.

Fontosabb segédfüggvények

void init_field_set(Game *game):

Inicializálja az aknamezőt. Lefoglalja a szükséges memóriaterületet, majd mindegyik mezőt alaphelyzetbe állítja

void place_mines(Field start, FieldSet fieldSet):

Kihelyezi az aknákat a pályán véletlenszerűen. A játék szabályainak megfelelően a felhasználó által először felfedett mező szomszédságában, valamint a mezőn nem lehet akna. Ezt a mezőt a függvény a paramétereként kapja meg.

Field *find_field(int x, int y, FieldSet fieldSet)

Koordináták által megkeres egy mezőt az aknaterületen.

`void reveal_mines(FieldSet fieldSet)`

Vesztett játék esetén felfedi az aknákat. Pirossal jelöli a felrobbant aknát, a tévesen megjelölt aknát pedig áthúzott aknával jelöli.

`int count_neighbour_mines(Field field, FieldSet fieldSet)`:

Megszámolja, hogy egy adott mező körül hány darab akna található.

`autodiscover_empty_fields(Field *field, FieldSet fieldSet)`:

Felfedi az adott mezővel összefüggő üres területeket. Rekurzív módon működik.

`bool is_winner(FieldSet fieldSet)`:

Ellenőrzi, hogy a játékos nyert-e.

Eredményelző modul

Ebben a modulban tallálható az eredményhelző (leaderboard) betöltéséhez és kijelzéséhel található minden függvény.

Típusok

Eredmény (Result)

Egy elért eredményt reprezentál. Eltárolja a játékosnevet, a pálya méretét, az aknák számát, valamint a játék befejezéséig eltelt másodperceket.

```
typedef struct Result {
    char player_name[100 + 1];
    int seconds;
    int mines;
    int cols;
    int rows;
} Result;
```

Fontos makrók

`min(a, b)`

Visszatér két szám mimimumával.

`MAXIMUM_RESULTS`

A betöltendő (és megjelenítendő) eredmények maximális száma. (Alapértelemezetten 5)

Fontosabb belső globális változók

`DEFAULT_FILENAME`

A fájl neve, ahol az eredmények tárolva vannak.

`top_results`

A top [`MAXIMUM_RESULTS`] eredményt tárolja. (Alapértelemezetten 5)

`fp`

A fájlkezeléshez szükséges `FILE` pointer.

Fontosabb függvények

`bool init_leaderboard(int p_cols, int p_rows, int p_mines)`

Ezzel a függvénnyel kell inicializálni az eredményjelzőt. Paraméterként kéri a játékterület méreteit, valamint az aknák számát. Ez alapján olvassa majd be a megfelelő eredményeket.

`void save_game_results(char *player_name, int seconds)`

Ezzel a függvénnyel el lehet menteni egy játék eredményét. Paraméterként a játékos nevét kapja, valamint a játék kezdete után eltelt másodpercek számát.

`void load_top_results()`

Betölti a top [`MAXIMUM_RESULTS`] (5) eredményt egy tömbbe. A 0. elem a legjobb eredménynek felel meg, és a másodpercek szerint növekvő sorrendben kerül a tömb feltöltésre.

`void show_leaderboard()`

Megjeleníti az eredményjelzőt.

`void close_leaderboard()`

Bezárja az eredményjelző modult, felszabadítja a felhasznált memóriát.

Fontosabb segédfüggvények

`void write_header()`

Üres fájl esetén a CSV headert beleírja a fájlba. (Azaz az oszlopok nevét)

`void put_result(Result result)`

Behelyez egy eredményt (`result`) a legjobb eredményeket tartalmazó tömbbe, a megfelelő helyre. (Amennyiben belefér.)

Működés

A program belépési pontja a főmodulban van. Itt kerülnek beállításra a játék paraméterei a parancssori argumentumoknak megfelelően. Ha valamelyik nincs megadva, az alapértelmezett értékek kerülnek beállításra. A parancssori argumentumok: `minesweeper [aknamezo_szelessege] [aknamezo_magassage] [aknak_szama] [idolimit_masodpercekben]`

Megjelenik az ablak, minden grafikus elem alaphelyzetbe kerül, a játék állapota is kezdeti állapotba áll, majd elindul az event loop. Minden ismétlésnél az ablak frissítésre kerül. A játék kezdeti állapotba állításakor a program inicializálja a játékmodulban a belső globális változókat, lefoglalja a memóriaterületet a mezőknek, majd kirajzolja őket.

Egérkattintás esetén a játékmodul megfelelő függvénye feldolgozza az eseményt. A függvény működése fentebb van leírva.

Amennyiben az Eredmények gombra (szövegre) kattint, amely a bal felső sarokban található, megjelenik az eredményjelző. A program frissíti a betöltött eredményeket. Ez a gomb csak akkor kattintható, ha nincs éppen játék folyamatban.

Ha már meg van nyitva az eredményjelző, akkor ismét a gombra - melynek a felirata már "Játék" - kattintva újra megjelenik a játéktábla és új játék kezdődik.

Amennyiben a játékos az ablakon található piros x-re kattint, a játékmezők, a grafika és az eredményjelző által lefoglalt memória felszabadításra kerül, majd a program kilép.