Programozói leírás (Slay)

A Slay játék az SDL könyvtár felhasználásával készült eseményvezérelt grafikus program. A játék hatszög alakú cellákból álló pályán zajlik. A játékos a gép ellen játszik. A játék úgy kezdődik, hogy először megjelenik egy pályaválasztó ablak. A pálya kiválasztása után a program véletlenszerűen szétszór díszítő elemeket a pályán és létrehoz egy-egy négy cellából álló összefüggő területet a gép és a játékos számára egy-egy bázissal. A játékos akkor nyer, ha sikerül elfoglalnia az ellenség bázisát. Fordított esetben a játékos veszít. Minden pályamérethez külön pályacsúcs nyilvántartás tartozik.

Külső könyvtárak
A szabványos C-n kívül szükséges az SDL telepítése.
Használt SDL grafikus könyvtárak a következők:
#include <SDL2/SDL.h> // alap SDL
#include <SDL2/SDL2_gfxPrimitives.h> // kiegészítő, pl. téglalap rajzoláshoz
#include <SDL2/SDL_image.h> // kiegészítő, PNG formátumú képek beolvasásához és kirajzolásához
#include <SDL2/SDL_ttf.h> // kiegészítő, betűtípus használathoz

Fordítás: a Code::Blocks fejlesztőrendszerben a „Build and run” gomb megnyomásával.

Futtatás
A lefordított kesz.exe futtatásához szükségesek:
SDL lefordított függvényei: libFLAC-8.dll, libfreetype-6.dll, libjpeg-9.dll, libmodplug-1.dll, libmpg123-0.dll, libogg-0.dll, libpng16-16.dll, libtiff-5.dll, libvorbis-0.dll, libvorbisfile-3.dll, libwebp-7.dll, SDL2.dll, SDL2_gfx.dll, SDL2_image.dll, SDL2_mixer.dll, SDL2_ttf.dll, zlib1.dll
Font: LiberationSerif-Regular.ttf
Képek:
tartalom.png, toronyfarm.png, egysegek.png, tulaj.png, gomb.png, becsukott_menu.jpg, lenyitott_menu.jpg

Függvények:

bool palyavalaszto(Jatek *jatek, SDL_Window *window, SDL_Renderer *renderer, TTF_Font *font); // pályaválasztó, true ha sikeres

void jatek_init(Jatek *jatek); // játék struktúra inicializálás

bool palya_init(Jatek *jatek, SDL_Renderer *renderer, TTF_Font *font);
//dinamikus pálya foglalás, pálya kezdeti állapotának véletlenszerű feltöltése, képek
//betöltése, pálya kirajzolás, képek kirajzolása, játékos egyenlegének és költségvetésének
//kiírása, aktuális kör sorszám kiírása, csúcsok beolvasása, aktuális csúcs kiírása
//True ha sikeres

bool palyat_lefoglal(Jatek *jatek);
//Lefoglalja a pálya hatszögeit. True ha sikeres.

bool palyat_feltolt(Jatek *j);
//Feltölti a dinamikusan lefoglalt pályát. True, ha minden rendben volt.

void kepeket_betolt(Jatek *jatek, SDL_Renderer *renderer);
//képek betöltése

void palyat_rajzol(Jatek *jatek, SDL_Renderer *renderer);
//kirajzolja a pálya hatszögeit a rajtuk lévő tartalommal

void kepeket_rajzol(Jatek *j, SDL_Renderer *renderer);
//kirajzolja a képeket

void egyenleget_kiir(Jatek *jatek, SDL_Renderer *renderer, TTF_Font *font);
//kiírja a játékos aktuális egyenlegét

void koltsegvetest_kiir(int koltsegvetes, SDL_Renderer *renderer, TTF_Font *font);
//kiszámoltatja és kiírja a játékos költségvetését

int koltsegvetes(Jatek *jatek, Tulajdonos tulaj);
//kiszámolja a játékos költségvetését

void korszamot_kiir(Jatek *jatek, SDL_Renderer *renderer, TTF_Font *font);
//kiírja hányadik körben jár a játékos

void csucs_beolvas(Jatek *j, char const *fajlnev);
//csúcsok fájlból olvasás

void csucs_mentes(Jatek *j, char *fajlnev);
//csúcsok fájlba mentés
void csucsot_kiir(Jatek *j, SDL_Renderer *renderer, TTF_Font *font);
//aktuális csúcs kiírása

static bool lefoglal_szomszed(Hexa *hatszog, int const meret);
//Lefoglalja egy hatszög szomszédait. True ha a foglalás sikeres.

static void free_szomszed(Jatek *j);
//Minden hatszög szomszédtömbjét felszabaditja.

static int szamol(Meret palya, int y, int x);
//Megszámolja és visszaadja hány szomszédja van egy hatszögnek.

static void szomszedokat_berak(Hexa *hatszog, Meret palya, int y, int x);
//Egy hatszög dinamikusan lefoglalt szomszéd tömbjébe bepakolja szomszédai pozícióját.
//Alatta lévőtől kezdve óramutató járásával megegyező irányban.

static void free_PozicioLista(PozicioLista *eleje);
//Felszabadít egy pozíció listát.

static bool elore_beszur(PozicioLista **eleje, int y, int x);
//Pozíció lista elejére beszúr egy egy új pozíciót.
//**eleje = A pozíció lista elejére mutató pointerre mutató pointer.
//True ha a beszúrás sikeres.

static bool volt(PozicioLista *voltak, int y, int x);
//Megnézi, hogy szerepel-e a pozíció listában a megadott pozíció.
//true, ha szerepel.

static bool uresen_vegigmegy(Jatek *j, PozicioLista **voltak, int y, int x, int cel);
//Rekurzívan végigmegy az üres hatszögeken szomszédról szomszédra.
//Pozíció lista segítségével jegyzi meg, hogy hol járt már (így nem lép vissza oda ahol már
//járt).
//**voltak = A pozíció lista elejére mutató pointerre mutató pointer.
//cel = Egy bizonyos sor aminek az elérése a cél.
//true, ha elértük a célt.

static bool elvag_e(Jatek *j, int y, int x);
//Átadja az uresen_vegigmegy függvénynek a megfelelő kezdő pozíciót.
//És lefoglalja az elsõ lista elemet az elore_beszur függvénnyel.
//true, ha az uresen_vegigmegy false (ha nem találunk más kiutat akkor elvágna).

bool szomszedos(Jatek *j, Tulajdonos tulaj, int y, int x);
//Megnézi, hogy egy bizonyos pozíció szomszédjában van-e megadott tulajdonú terület.
//true, ha van.

static bool tulajdonok(Jatek *j, Tulajdonos tulaj, int sor, int oszlop);
//Egy kisorsolt pozícióra kirakja a bázist és a tulajdonokat mellésorsolja feltételezve, hogy van
//elég hely.
//true, ha sikeresen lefoglalódik a láncolt lista.

static bool hanyadik_ures(Jatek *j, Pozicio *hova, int hanyadik);
//Egy kisorsolt számhoz (hanyadik üres hatszög) megtalálja a pozíciót (a bázisok sorsolódnak
//egy üres hatszögre).
//*hova = A pozícióra mutató pointer amit átír ha megvan.
//true, ha sikerült megtalálni.

static bool van_eleg_hely(Jatek *j, PozicioLista **voltak, int y, int x, int cel);
//Megnézi, hogy egy adott kisorsolt pozíció környékén van-e elég hely.
//Rekurzívan végigmegy a senki tulajdonú hatszögeken szomszédról szomszédra.
//Pozíció lista segítségével jegyzi meg hogy hol járt már (így nem lép vissza oda ahol már
//járt).
//**voltak = A pozíció lista elejére mutató pointerre mutató pointer.
//cel = Hány területnek kell (még) elférnie.
//true, ha a cél nullára vált (akkor van elég hely).

static bool bazisokat_kisorsol(Jatek *j, int uresek);
//A palyat_feltolt függvénybõl megkapja mennyi az üres terület.
//Ezekbõl kisorsol egyet a játékos bázisának késõbb a gép bázisának.
//Meghívja a tulajdonok függvényt ami lerakja a bázist és kisorsol mellé tulajdonokat.
//true, ha minden rendben volt.

bool palyat_feltolt(Jatek *j);
//Feltölti a dinamikusan lefoglalt pályát.

//true, ha minden rendben volt.

void free_minden(Jatek *j);
//Felszabadítja a hatszögek szomszédait és a kétdimenziós hatszög tömböt.

void ujra_rajzol(Jatek *j, SDL_Renderer *renderer, TTF_Font *font, int *nyert);
//újrajalzolja a pályát, képeket, egyenleget, költségvetést, kör számlálót, csúcs érteket

bool esemeny_hurok(Jatek *jatek, SDL_Window *window, SDL_Renderer *renderer, TTF_Font *font);
//eseményvezérelt hurok

Pozicio cella_pozicio(SDL_Event *esemeny);
//visszaadja, hogy melyik cellán történt a bal egér gomb lenyomása vagy felengedése

int vedelmi_szint(Jatek *j, Tulajdonos tulaj, int y, int x);
//visszaadja az adott ellenséges cella védelmi szintjét

void egyseg_pusztulas(Jatek *jatek, Tulajdonos tulaj);
//csőd esetén egyenleg nullázás és egység pusztulás

void mind_lephet(Jatek *jatek);
//az összes cella léphetőre állítása

void free_hatszog(Jatek *jatek);
//Felszabadítja a kétdimenziós hatszög tömböt.

void gep_kore(Jatek *j, bool *jatekvege, int *nyert);
//gép köre

bool kurzor_palya_teglalapon(SDL_Event *esemeny, Jatek *j);
//true, ha az egér művelet a pályát befoglaló téglalpon belül történik

bool kurzor_nyomogomb_teglalapon(SDL_Event *esemeny, Jatek *j);
//true, ha az egér művelet a nyomógombot befoglaló téglalapon belül történik

bool kurzor_egysegek_teglalapon(SDL_Event *esemeny, Jatek *j);
//true, ha az egér művelet a vásárolható egységek vagy ingatlanok képén történik

bool lehet_foglalni(Jatek *j, Tartalom *kivalasztva, Pozicio *pozicio);
//true, ha a kiválasztott egységgel az adott cella foglalható

bool lehet_sajatra(Jatek *j, Tartalom *kivalasztva, Pozicio *pozicio);
//true, ha a kiválasztott tartalom lerakható az adott saját cellára

bool valaszthato_egyseg(Jatek *j, Pozicio *pozicio);
//true, ha az adott cella a játékos tulajdonában van, egység a tartalma és el lehet lépni róla

void kivalaszt(SDL_Event *esemeny, Jatek *j, Tartalom *kivalasztva);

//eltárolja, hogy melyik egységen vagy ingatlanon nyomták le a játékos a bal egér gombot

void jatekos_nyert_kiir(Jatek *j, SDL_Renderer *renderer, TTF_Font *font, int *nyert);
//a nyert paraméter értéke szerint kiírja, hogy nyert = 1 esetén "Gratulálunk Ön nyert!", nyert //= 2 esetén "Gratulálunk Ön új csúccsal nyert!", nyert = 3 esetén "Sajnos Ön veszített!"

void eger_gomb_le(SDL_Event *esemeny, bool *quit, bool *menu_nyitva, bool *rajzolni, int *nyert, bool *uj_jatek, bool *jatekvege, Jatek *jatek, SDL_Window *window, SDL_Renderer *renderer, TTF_Font *font, Pozicio *egyseg, bool *belso, Tartalom *kivalasztva);
//bal egér gomb lenyomásának kezelése

void eger_gomb_fel(SDL_Event *esemeny, bool *rajzolni, int *nyert, bool *jatekvege, Jatek *jatek, SDL_Window *window, SDL_Renderer *renderer, TTF_Font *font, Pozicio *egyseg, bool *belso, Tartalom *kivalasztva);
//bal egér gomb felengedésének kezelése

void gep_lep(Jatek *j);
//Gep lepese.

void elvago(Jatek *j, Tulajdonos tulaj, int y, int x);
//Felterkepezi egy adott pozicio korul az elvagando teruleteket.
Amit el kell vagni arra meghivja az elvag fuggvenyt.
//tulaj = Kinek a teruleteit kell elvagni.

static bool jatekos_szomszedban(Jatek *j, Pozicio* hol);
//Megnezi, hogy a jatekos a gep szomszedjaban van-e.
//true, ha igen.

static bool van_hely_szomszedban(Jatek *j, Pozicio *hol);
//Megnezi hogy egy pozicio mellett van-e a gepnek ures terulete.
// true, ha igen.

bool van_tornya(Jatek *j)
// Megnezi, hogy a gepnek van-e tornya.

static void tornyot_lerak(Jatek *j, Pozicio *hol);
// Megprobalja lerakni a gepnek a tornyat a jatekos teruletevel szomszedos (gepi) terulet valamelyik (gepi) szomszedjara. Ha nem sikerul akkor a jatekos teruletevel szomszedos (gepi) teruletre probalja rakni. Ha az sem sikerul akkor veletlenszruen. (A fuggveny feltetelezi, hogy a gepnek van ures terulete.)
//hol = Megkapja egy olyan gepi terulet poziciojat ami mellett jatekos terulete van.

static void bazist_megkeres(Jatek *j, Pozicio *hol);
// Megkaresi a gep bazisat.
//hol = Visszaadja a gep bazisanak poziciojat.

static bool van_hely(Jatek *j);
//Van-e ures terulet a gep teruleten.
//true, ha van.

static void farm_lerak(Jatek *j, Pozicio *hol);
//Megprobalja eloszor lerakni a farmot a bazis mellé, ha nem tudja akkor mashova.
//hol A bazis pozicioja.

static void jatekos_terulete(Jatek *j, Pozicio *hol);
//A megadott pozicio mellett megkeres egy jatekos teruletet.
//param hol Jatekos teruletenek a pozicioja.

static void kardos_lerak(Jatek *j, Pozicio *hol);
//Lerak egy kardost a megadott poziciora.

static bool landzsas_lerak(Jatek *j, Pozicio *hol);
A megadott pozicio mellett megprobal elfoglalni egy jatekos teruletet landzsassal.
hol = Ha sikerult atadja hova rakta le.
true, ha sikerult a foglalas.

static void elhal_e(Jatek *j, PozicioLista **voltak, Tulajdonos tulaj, bool *elhal, int y, int x);
//Rekurzivan vegigmegy a gep teruletein es eldonti, hogy le kell-e vagni.
elhal = alapbol true, ha bazist talal false lesz.
//true, ha le kell vagni.

static void elvag(Jatek *j, PozicioLista *voltak);
//Az elvagott teruleteket letorli.
//voltak = Az elvagando teruletek listaja.

void elvago(Jatek *j, Tulajdonos tulaj, int y, int x);
//Felterkepezi egy adott pozicio korul az elvagando teruleteket.
//Amit el kell vagni arra meghivja az elvag fuggvenyt.
//tulaj = Kinek a teruleteit kell elvagni.

static int mit_foglalhat(Jatek *j, Tartalom foglalo, PozicioLista **szomszedosak);
//A foglalo egyseg altal elfoglalhato teruleteket dinamikus listaba teszi.
// db A lista elemeinek szama.

static bool paraszt_lerak(Jatek *j, Pozicio *hol);
//Egy parasztot lerak egy veletlenszeru szomszedos teruletre.
// hol = Ha sikerult atadja hova rakta le.
// true, ha sikerult lerakni.

bool egyseg(Jatek *j, int y, int x);
//true, ha a pozicion egyseg van,

void egyseg_lep(Jatek *j, Tartalom foglalo, int y, int x);
//Egy veletlenszeru elfoglalhato teruletre lepteti az egyseget ha tudja.
//Ha nem tudja nem lepteti.

void egysegekkel_lep(Jatek *j);
//Minden egyseget atad az egyseg_lep fuggvenybek.

Projekt felépítése

Modulok:

main.c // Főprogram

Header fájlok:

- #include <stdlib.h> //srand()
- #include <time.h> //time()
- #include "adatbekero.h" //SDL, stdbool, strukturak, palyavalaszto()
- #include "jatekmenet.h"
  //sdl_init(), sdl_close(), jatek_init(), palya_init() esemeny_hurok()
- #include "palya_init.h" //free_minden()

Függvényhívások:

- sdl_init, sdl_close, jatek_init, palyavalaszto, palya_init, esemeny_hurok, free_minden

palya_init.c // Pálya foglalás, feltöltés, felszabadítás

Header fájlok:

- #include <stdlib.h> //malloc(), free(), rand()
- #include "palya_init.h" //stdbool, strukturak

Függvények:

- palyat_lefoglal, lefoglal_szomszed, free_szomszed, szamol, szomszedokat_berak, free_PozicioLista, elore_beszur, volt, uresen_vegigmegy, elvag_e, szomszedos, tulajdonok, hanyadik_ures, van_eleg_hely, bazisokat_kisorsol, palyat_feltolt, free_minden

adatbekero.c // Pályaméret kiválasztás

Header fájlok:

- #include "adatbekero.h" //SDL, stdbool, strukturak)

Függvények:

- palyavalaszto

gepi_jatekos.c // Gép köre

Header fájlok:

- #include <stdlib.h> //rand()
- #include "gepi_jatekos.h" //stdbool, strukturak
- #include "palya_init.h" //szomszedos(), elore_beszur(), free_PozicioLista(), volt()
- #include "jatekmenet.h" //vedelmi_szint()

Függvények:

- jatekos_szomszedban, van_hely_szomszedban, van_tornya, tornyot_lerak, bazist_megkeres, van_hely, farm_lerak, jatekos_terulete, kardos_lerak, landzsas_lerak, elhal_e, elvag, elvago, mit_foglalhat, paraszt_lerak, egyseg, egyseg_lep, egysegekkel_lep, gep_lep

jatekmenet.c // Játékmenet

Header fájlok:

- #include <stdio.h> //fopen(), fscanf(), fprintf(), sprintf(), FILE
- #include <stdlib.h> //exit()

- #include "jatekmenet.h" //SDL, stdbool, strukturak
- #include "palya_init.h" //szomszedos(), palyat_lefoglal(), palyat_feltolt()
- #include "gepi_jatekos.h" //elvago(), gep_lep()

Függvények:

- sdl_init, sdl_close, csucs_beolvas, csucs_mentes, jatek_init, kepeket_betolt, kepeket_rajzol, palyat_rajzol, gep_kore, koltsegvetes, egyseg_pusztulas, mind_lephet, koltsegvetest_kiir, korszamot_kiir, egyenleget_kiir, csucsot_kiir, palya_init, jatekos_nyert_kiir, ujra_rajzol, cella_pozicio, vedelmi_szint, kurzor_palya_teglalapon, kurzor_nyomogomb_teglalapon, kurzor_egysegek_teglalapon, lehet_foglalni, lehet_sajatra, valaszthato_egyseg, kivalaszt, eger_gomb_le, eger_gomb_fel, esemeny_hurok

Konstansok:

int const **hexa_oldal** = 26; //a hatszög cellák oldalhosszúsága pixelben
int const **hexa_tavol** = 39; //az egy sorban egymás mellett lévő cellák bal szélső //csúcspontjainak x irányú távolsága pixelben
int const **hexa_magas** = 45; //a hatszög cellák magassága pixelben (közelítő érték)
int const **x_palya** = 100; //a pálya bal felső sarka helye az ablakban x irányban
int const **y_palya** = 100; //a pálya bal felső sarka helye az ablakban y irányban
int const **ablak_szeles_init** = 400; //kezdetben ablak szélessége
int const **ablak_magas_init** = 200; //kezdetben ablak magassága
SDL_Color const **feher** = {255, 255, 255}, **fekete** = { 0, 0, 0 }, **piros** = { 255, 0, 0};
//feliratokhoz szín konstansok
char const ***gomb_felirat[3]** = {"Kis pálya", "Közepes pálya", "Nagy pálya"};
//Nyomógomb konstans feliratok
int const **arak[10]** = {0, 20, 10, 0, 25, 20, 40, 0, 0, 0}; //mennyiért lehet egységeket és //ingatlanokat vásárolni, a 0 azt jelenti, hogy nem megvásárolható
SDL_Rect const **menu_helye** = { 10, 10, 80, 65 }; //menü helye konstans

```
Enum {
        farm_bevetel = 3, //körönkénti ennyit termel egy farm
        cella_bevetel = 1, //körönként ennyit termel egy cella
        paraszt_kiadas = 2, //körönkénti kiadás egy paraszt egység után
        landzsas_kiadas = 6, //körönkénti kiadás egy lándzsás egység után
        kardos_kiadas = 18 //körönkénti kiadás egy kardos egység után
};
```

Struktúrák: (adatbekero.h)

```
//Pálya mérete hatszögekben
typedef struct Meret {
   int szeles, magas;
} Meret;

//Hatszög koordináta
typedef struct Pozicio {
   int sor, oszlop;
} Pozicio;

//Hatszögek koordinátáinak listája, pozíciók egyesével való összegyűjtésére hasznos
typedef struct PozicioLista {
   Pozicio hely;
   struct PozicioLista *kov;
} PozicioLista;

//Hatszögek tartalma szándékosan növekvő erősségi sorrendben, ami hasznos a hatszögek
védelmének kiszámításakor
typedef enum Tartalom {
   ures, farm,
   paraszt, bazis,
   landzsas, torony,
   kardos,
   hegy, to, erdo
} Tartalom;

//Hatszögek tulajdonosa
typedef enum Tulajdonos {
   senki, jatekos, gep
} Tulajdonos;

//Maga a hatszög: pozíciója, tartalma, tulajdonosa, szomszédai tömb, szomszédok száma, el
lehet-e lépni róla
A szomszédok eltárolása hasznos, mert így csak egyszer kell kiszámítani.
typedef struct Hexa {
   int x; // hatszög cellát befoglaló téglalap bal felső sarka x koordináta pixelben
   int y; // hatszög cellát befoglaló téglalap bal felső sarka y koordináta pixelben
   Tartalom tartalom; // cella tartalma
   Tulajdonos tulaj; // cella tulajdonosa
   Pozicio *szomszedok; // cella szomszédai
   int mennyi; // mennyi szomszéda van a cellának
   bool lephet; // el lehet-e lépni a celláról
 } Hexa;

//A játékmenet adatai
typedef struct Jatek {
   int palya_tipus; //pálya típusa: 0 = Kis pálya, 1 = Közepes pálya, 2 = Nagy pálya
   int palya_csucs; // új pálya csúcs
```

```
    Meret palya;  //pálya hány sorból és oszlopból áll
    Meret palyapx; //pálya mérete pixelben
    Meret ablak; // ablak mérete pixelben
    int kis_csucs; //eddigi kis pályás csúcs
    int koz_csucs; //eddigi közepes pályás csúcs
    int nagy_csucs; //eddigi nagy pályás csúcs
    Hexa **hatszog; // pálya hatszög celláit leíró kétdimenziós tömb
    SDL_Texture *becsukott_menu, *lenyitott_menu, *tulaj, *tartalom, *egysegek,
*toronyfarm, *nyomogomb; // betöltött képek pointerei
    int korok; // kör számláló
    int jatekos_arany; // játékos egyenlege
    int gep_arany; // gép egyenlege
} Jatek;
```

Eseménykezelő hurok:

Események:

- bal oldali egér gomb lenyomásra
    1. Menü lenyitása
    2. Menü összecsukása
    3. Menüpontok kiválasztása
    4. egység vagy ingatlan kiválasztása vásárláshoz
    5. pályán lévő egység kiválasztása lépéshez
- bal oldali egér gomb felengedésre
    1. kiválasztott egység vagy ingatlan lerakása
- ablak bezárása