Programozás alapjai 1

Czap Balázs

Nagy házi feladat
Programozói Dokumentáció

A program működése:

Az alapértelmezett értékek beállítása után belép a program az event loopba.
Itt az egér lekezelésén kívül 10 ms-enként a program fő részére kerül sor.
Először, kiszámolódik, hogy melyik cellába klikkeltünk, majd ha az élettéren rajzolni vagy törölni szeretnénk, akkor ez elvégződik.
Egy (gombbal állítható) belső számláló hatására bizonyos időközönként kiszámolódik az élettér új állapota, ha futó állapotban van.
Végül a menü esetleges lekezelése, és az új adatok kirajzolása történik meg.

A menü interaktív gombokat tartalmaz, hoverelés és klikkelés hatására színt vált, stb.
Lehetséges törölni az életteret, elindítani a szimulációt, megállítani azt, csak egyet lépni előre a szimulációban, elmenteni a jelenlegi állapotot, betölteni egy állapotot, „zoomolni”, a szimuláció sebességét változtatni, és az élettér méretét növelni.

Az élettér kiszámítása úgy működik, hogy egy ideiglenes élettérre beíródik, az első alapján, az összes olyan cella ami a következő körben életben van. Ez a szokásos szabályok alapján történik, minden sejtnek megnézzük a szomszédjainak a számát, és ez alapján eldöntjük hogy élni fog-e.
Az ideiglenes élettér azért kell, mert ha az eredeti életteret módosítanánk, akkor az újonnan létrejött sejtek megmenthetnék az elhalálozókat, és az elhalálozók megakadályozhatnák a születést.
Miután minden cellát kiszámoltunk, az ideiglenes életteret visszamásoljuk az újba.

Adatszerkezetek:

Az élettér egy leképzett, x*y hosszúságú dinamikusan foglalt tömbben tárolódik, és emellé jár egy pointertömb, amiben a négyzetrács sorainak az elejére találunk pointereket.

Mivel az élettér logikai értékeket tartalmaz csak, ehhez nem kell külön adatszerkezet.

A **Pont** struktúra elsősorban egy pont koordinátáit határozza meg (vagy az élettér, vagy a teljes képernyő koordinátáját), de fel lett használva az élettér teljes méretének a tárolására is (azaz a legutolsó pont x és y koordinátájának a tárolására).

A **Menu_btn** struktúra egy gomb adatait tárolja el:

- a színét mind a négy esetre (alapértelmezett, hover, lenyomva, és inaktív) vonatkozóank
- a sarkainak a koordinátáit
- a gombra írt szöveget
- egy **Btnstate** enumot
- és egy clicked értéket, ami elsősorban a hosszú nyomvatartás esetének a lekezelésére van.

A **Btnstate** enum egy gomb lehetséges állapotait határozza meg, azaz alapértelmezett, hover, lenyomva, és inaktív.

Végül a **clickaction** enum a klikkelések lehetséges állapotait határozza meg, azaz vagy rajzolunk az élettéren, vagy törlünk az élettéren, vagy a menü területére klikkeltünk, vagy ezek közül egyik se.

A program függvényei:

Az élettér függvényei:

int eletter_create(Pont size);**
Dinamikusan foglal egy életteret a feljebb leírt módon (az első paramétere az élettér mérete), és a sorok elejére mutató pointerek tömbjével tér vissza.

void eletter_free(int **eletter);
Felszabadítja a paraméterként kapott életteret.

void eletter_nullaz(int **eletter, Pont size);
Feltölti az élettér összes celláját 0 értékűre.

void eletter_draw(SDL_Surface *screen,int **eletter,Pont screensize, Pont size,int cellsize, int menu_width, int color, int correction);
Le kell hagyni a széléről correction-nyi sort (alapértelmezetten 3); a képernyőméret, menu_width és correction paraméterek alapján kialakít két padding értéket, hogy az életteret a képernyő közepére rajzolja, majd ezt felhasználva kirajzolja az összes cellát, majd a négyzetrácsot.

int szomszedok_szamol(int **eletter, int y, int x);
Az élettér y. sorában található x. cellának megszámolja, hány aktív szomszédja van, majd a számukkal tér vissza.

void eletter_szamol(int **eletter, Pont size);
Felveszi a következő kör állapotát.

Pont eletter_koord(Pont size, Pont screensize,int menu_width, int cellsize, Pont clickpos, int correction);
Meghatározza, hogy a klikk helyzete az élettér melyik cellájára esik, és visszatér ezzel. Lényegében konverziót végez el a képernyőkoordináta és az élettérkoordináta között. Ha az élettéren belül klikkelünk, majd kihúzzuk róla az egeret, akkor az élettér oldalsó soraiba és oszlopaiba rajzolunk.

void eletter_save(int **eletter, Pont size);
A paraméterként kapott életteret elmenti egy .txt fájlba.
A fájl neve a jelenlegi dátum lesz HHH-NN-ÓÓ-PP-MP formátumban.
Az első sorban tárolódik az élettér mérete (oszlopok, majd sorok száma), utána pedig minden sor kiírva, és egy newline karakterrel elválasztva.
Az aktív értéket az ’x’, az inaktívat a ’.’ jelöli.

void eletter_load(int ***eletter, Pont *size, const char *filename);
A paraméterként kapott fájlból beolvassa először a méreti értékeket, létrehoz egy ekkora életteret, és azt feltölti a fájl maradék tartalmával.

Az menü függvényei:

void btn_draw(SDL_Surface *screen, Menu_btn btn);

Kirajzolja a paraméterként kapott gombot kerettel, a szövegével, és Btnstatejétől függő színnel.

int gombon_e(Menu_btn btn, Pont clickpos);

Az klikkelés képernyőkoordinátájáról megmondja, hogy a gomb kerületén belül esik-e. Visszatér ennek a logikai értékével.

int melyik_gombon(Menu_btn *Menu, int gombokszama, Pont clickpos);

Megnézi az összes gombot, hogy aktívak-e. Mivel „definíció szerint” csak egyetlen gomb lehet aktív, így vissza tud térni egy inttel, ami megmondja hogy hanyas számú gomb van megnyomva.

void menu_generate(struct Menu_btn *Menu, int gombokszama, int menu_width);

Saját elképzelés, („design”) alapján feltölti a paraméterként kapott Menu tömböt gombokkal.
A Menu tömb statikus méretűre van meghatározva, a main függvényben; de ez nem gond mert a gombok száma nem változik.

Menu_btn nyil_btn_generate(int irany, int p1_x,int p1_y,int p2_x,int p2_y, int no_hover_color);

A menu_generate függvény használja csak fel, több, hasonló nyíl gombot kell létrehozni, ez a függvény azt próbálja egyszerűsíteni.

void menu_szamol(SDL_Surface *screen, int ***eletter, Pont *size, Menu_btn *Menu, int gombokszama, int *running, clickaction click, Pont clickpos, clickaction screenpart, int *cellsize, int *timermax);

A legbonyolultabb függvénye a programnak.
Először a mentések könyvtárban található .txt kiterjesztésű fájlok neveit betölti egy tömbbe. Létre van hozva egy statikus számláló, ami ezt a tömböt indexeli, és ez az indexelt szöveg ki van írva a load gomb alá. A load gomb ezt a fájl tölti be mindig.

A kellő gombokat aktívvá teszi a függvény ha nincs a szimuláció futásban, illetve inaktívvá, ha futásban van. Ha egy gomb nem inaktív, és meg van nyomva, akkor a Btnstate-jét clicked-dé teszi, ha csak rajta van az egér, akkor hovering-gé, egyéb esetben no_hover-ré.
Kirajzolja az összes gombot a btn_draw függvénnyel, és kirajzolja az esetleges feliratokat (pl a nyíl gombok fölé).

Ezután a gombok megnyomásának a kezelése következik. Három eset van: a hosszú nyomvatartás nem okoz gondot, a hosszú nyomvatartást mindenképpen meg kell szüntetni, illetve a hosszú nyomvatartás alternatív működést eredményez.

- A reset gomb meghívja az eletter_nullaz fuggvényt az élettérre.
- A run gomb elindítja a szimulációt.
- A stop gomb leállítja a szimulációt.
- Az 1-step gomb függ az előző állásától, ez a *clicked* értékében tárolódik: ha nem fut a szimuláció, és nem volt előzőleg megnyomva, akkor elindítja szimulációt, ha fut a szimuláció és meg volt előzőleg nyomva akkor leállítja a szimulációt.
- A save gomb meghívja az eletter_save függvényt.
- A load gomb meghívja az eletter_load függvényt a fent leírt fájlra.
- A load gomb melletti nyilak növelik illetve csökkentik a betölthető fájlok tömböt, ha max illetve min értéket érnek el akkor visszatérnek az elejére illetve a végére. Ha hosszan nyomva vannak tartva, akkor lassan lapoznak a fáljok között.

- A cellanagyítás-és kicsinyítés nyilak funkciói egyértelműek.
- A szimuláció gyorsítása és lassítása a paraméterként kapott timermax értéket csökkenti illetve növeli, ezzel felgyorsítva illetve lelassítva a szimuláció sebességét.
- Élettér szélesség-hosszúság csökkentése: a méret csökkentése egy minimális értékig. Nem szükséges újrafoglalni, stb. az életteret, mert nem történhet túlindexelés a méret csökkentése miatt.
- Élettér szélesség-hosszúság növelése: a méret növelése egy maximális értékig, és az újonnan kapott értékekkel egy új élettér fevétele, majd az előző belemásolása ebbe.