NHF –programozói dokumentáció

Ez a program az aknakereső nevű játékot akarja megvalósítani SDL grafikus megjelenítéssel.

Memóriaterület lefoglalása a pályának

Ahhoz, hogy a szélső mezőkre is tudjak generálni számokat nagyobb kétdimenziós tömböt kell foglalni. Az így keletkezett szélső sort nem jelenítettem meg és nem is generáltam rá bombát. Ez által a pálya szélére is könnyedén tudtam generálni számokat mivel a generalas függvényem mind a 8 irányba nézi, hogy van-e bomba és így a pálya szélén lévő mezőknek is van mind a 8 irányba szomszédja.

deklaracio.h:

enum Mezo

13 db értéket tartalmaz, amik a mezo.png-beli sorrendben vannak → az így kapott indexek megegyenek a képbeli indexekkel.

enum MERET

A mezok.png-ben lévő mezők mérete.

struct cella

Tartalmazza, hogy:

- mi az adott mező kezdetének az x és y koordinátája
- bomba-e az adott mező
- mi az értéke az adott mezőnek (enum Mezo)
- meg van-e jelölve zászlóval
- meglátogatták-e (fel van-e fedve az értéke)

enum Smiley

Hasonló az enum Meret-hez, csak a smileyk 3 lehetséges értékét tárolja.

enum MERET2

A smiley.png-ben lévő smileyk mérete.

timer.c

A fejlécben megjelenő eltelt idővel és a felhasználható zászlók számával foglalkozik.

Függvényei:

void zaszlosz(SDL_Renderer *renderer, SDL_Surface *felirat, SDL_Texture *felirat_t, TTF_Font *font, SDL_Rect hova, int zaszlodb, int numx, int numy)

Kiírja a felhasználható zászlók számát, ami kezdetben megegyezik a bombák számával.

void timer(SDL_Renderer *renderer, SDL_Surface *felirat, SDL_Texture *felirat_t, TTF_Font *font, SDL_Rect hova, int sec, int numx, int numy)

Kiírja az eltelt időt az első kattintástól számítva.

Uint32 idozit(Uint32 ms, void *param)

Ez a függvény hívódik meg az időzítő által. Betesz a feldolgozandó események közé egy felhasználói eseményt.

jatekmenet.c

void mezo_kirajzol(SDL_Renderer *renderer, SDL_Texture *mezo, Mezo melyik, int x, int y)

Kirajzolja a megfelelő képet. Tudjuk, hogy a typedef enum Mezo-ben a mezők ugyanolyan sorrendben vannak, mint a képen és ez alapján könnyen meg tudjuk határozni a forrást, ami kivágja a megfelelő képet.

void smiley_kirajzol(SDL_Renderer *renderer, SDL_Texture *smiley, Smiley melyik, int x, int y)

Ugyan olyan elven rajzolja ki a megfelelő smiley-t mint a mezo_kirajzol függvény.

void foglalas(cella ***uj, int numx, int numy)

Lefoglalja a megfelelő méterű helyet a 2 dimenziós tömbnek. Ezt fel kell szabadítani.

void felszabadit(cella **m, int numy)

Felszabadítja a dinamikusan foglalt 2 dimenziós tömböt.

void alaphelyzet(SDL_Renderer *renderer, SDL_Texture *mezo, Mezo melyik, cella **m, int numx, int numy)

Megjeleníti a lefedett mezőket és beállítja az alapértékeket a tömb struktúrájának.

void bombak_helye(cella **m, int numx, int numy, int bombasz)

Kisorsolja a bombák helyét a pályán. Létrehoz egy tömböt (bombaszám méretű), amiben eltárolja a különböző sorsolt számokat és a számok alapján megjelöli azokat a mezőket, amiben bomba lesz.

void generalas(SDL_Renderer *renderer, SDL_Texture *mezo, cella **m, int numx, int numy)

Legenerálja a bombák helyét és ez alapján azt, hogy a többi mező milyen számot tartalmaz. Ha nincs körülötte bomba akkor üresen marad az adott mező.

void kitolt(SDL_Renderer *renderer, SDL_Texture *mezo, cella **m, int j, int i, int numx, int numy)

Felfedi az üres mezőket rekurzívan (8 irányban), mindaddig amíg:

- a pályán belül van a kattintás
- nem látogattuk meg az adott mezőt
- nem bomba a mező
- nem zászlós a mező
- üres a mező (ilyenkor a számos mezőt is felfedjük, ami az üres mező szomszédja)

void jatek_vege(SDL_Renderer *renderer, SDL_Texture *mezo, cella **m, int j, int i, int numx, int numy)

Ha bombára kattintunk, akkor felfedi az összes mezőt, és amelyik bombára kattintottunk az piros lesz.

bool nyert(SDL_Renderer *renderer, SDL_Texture *smiley, cella **m, int numx, int numy, int smileyx, int smileyy, int bombasz)

Akkor tér vissza igaz értékkel, ha csak a bombát tartalmazó mezők nincsenek felfedve.

main.c:

- Megkérdezi a felhasználót, hogy milyen nehézségi szinten szeretne játszani
- A nehézségi szinttől függően kirajzolja a megfelelő méretű pályát és elmenti változókba a mezők (numx, numy) és a bombák számát (bombasz) továbbá a fejlécben található smiley x koordinátáját (smileyx)
- Beolvassa a mezok.png (egy mező képe értékétől függően) és a smiley.png (a smiley arcai a játékmenettől függően) képet
- Betölti a betűtípust, amivel majd az eltelt időt és a zászlók számát jeleníti meg a fejlécben
- Létrehozza a pályának a kétdimenziós dinamikusan foglalt tömböt
- A while ciklusban lévő switch:
    - másodpercenként növeli az eltelt időt
    - kilép, ha az ablakot bezáró X-re kattintunk
    - ha a bal egérgombbal kattintunk, akkor megállapítja, hogy az ablakon belül hova kattintottunk és az alapján:
        - alaphelyzetbe állítja a programot, ha a smileyra kattintunk
        - kirajzolja a megfelelő mezőt
    - ha a nyert függvény értéke true, akkor kiírja a legjobb időt és az abban a menetben elért időt és ha új legjobb idő van, akkor a megfelelő txt fájlba kiírja az elért időt, ha az jobb, mint az előző idő
    - ha a jobb egérgombbal kattintunk, akkor megállapítja, hogy az ablakon belül hova kattintottunk és az alapján megjelöli a mezőt vagy eltávolítja a jelölést attól függően, hogy meg volt-e jelölve vagy nem
- Felszabadítja a memóriaterületet