**FarmVik – félkész házi feladat**

**A játék felépítése:**

A programom 4 .c fájlból áll össze, ezekben bontom szét kisebb elemekre a feladatokat.



A *global.c* és a hozzá tartozó *global.h* fájl a globálisan elérhető változók deklarálására és tárolására van. Itt sorolom fel az include listát is, hogy ne kelljen az összes fájlban újra leírni őket.

A **textures.c** fájl két függvénnyel kezdődik, amik arra szolgálnak, hogy a későbbiekben képeket tölthessek be a memóriába, majd megjelenítsük azokat.

SDL\_Texture \*loadTexture(**char** \*path)

A *loadTexture(char \*path)* függvénybe a kívánt elérési útvonalat beírva tudjuk a textúrát létrehozni,

**void** loadImage()

majd a *loadImage()* függvénnyel betölteni a memóriába, hogy később használhassuk.

**void** doRender()  
{  
 **int** NUMBER\_OF\_IMAGES = 3; // ennyi féle terményt lehet gondozni / növelni  
  
 SDL\_SetRenderDrawColor(renderer, 76, 175, 80, 255);  
 SDL\_RenderClear(renderer);  
 SDL\_SetRenderDrawColor(renderer, 0, 0, 0, 255);

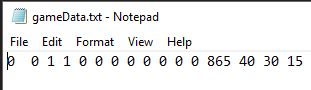
Ezt követi a *doRender()* függvény, amiben a folyamatosan a képernyőn lévő dolgokat renderelem be. Ilyenek például a logo, a kicsi ikonok, a bolt ikonjai, stb. Ezt egy rövid függvény követi, amit mellékszámítások miatt hoztam létre, és arra használom, hogy egy szám számjegyinek a számát adjam meg. Ezt a következő függvényben használom, ahol a bal fent elhelyezkedő ikonok mellett lévő számokat, és az érme melletti számot jelenítem meg. Itt a számok változása miatt mindig változó méretű szövegdobozta van szükség, ezt a problémát orvosolja a *digit()* függvény.

A *game.c* elején hozom létre a függvényt, amivel az SDL-es és TTF-es dolgokat inicializálom.

**int** init() {  
  
 bool success = true;  
   
 // Initializing everything, setting up the playground  
 SDL\_Init( SDL\_INIT\_EVERYTHING );  
 TTF\_Init();

Ekkor hozom létre az ablakot, amiben a játék futni fog, és a renderert, amivel renderelem a későbbiekben a textúrákat. Ez után vannak az egyelőre kezdetleges *scan()* és *send()* függvények, amik a fájlkezelésért felelősek.

**void** scan()  
{  
 FILE\* data;  
 data = fopen("gameData.txt", "r");  
 fscanf(data,

 A játék adatait egy *gameData.txt* fájlban tárolom el, amiben a veteményes mezőinek tulajdonságai vannak, azaz hogy hányas számú növény hányas állapotban van, valamint a játékos vagyonát, és a terményei számát. Ezeket az adatokat a *main.c*-ben lévő játékciklus elején beolvasom, majd az *SDL\_RenderPresent(renderer)* meghívása előtt betöltöm a *send()* függvénnyel a .txt fájlba. Ennek előnye, hogy a játék automatikusan el van mentve minden változtatás után. Ezt a későbbiekben azzal fogom feldobni, hogy a játék elején megjelenő menüben ki lehet majd választani, hogy folytatni szeretnénk-e a játékot, vagy egy teljesen újat akarunk kezdeni, és ebben az esetben minden adatot alapállapotra fog állítani.

A gombok vezérlése, és az ültetés 4 függvény használatával történik, egyelőre még nem tökéletesen.

**int** goods()

A *goods()* függvény feladata, hogy ha a játékos a veteményesen belül kattint valahova, akkor a cella számát visszaadja. Ezek balról lefele, majd jobbról folytatva lefelé növekednek 1-től kezdve. Ha a bal oszlopra kattint, akkor a visszatérési értéke a függvénynek

**return** (buttony / d) + 1;

ahol a *buttony* a legfelső cella teteje és az egér y koordinátája közötti hely, *d* pedig egy cella magassága (és egyben szélessége is). Ha a jobb oszlop beli mezőkre kattint, akkor a visszatérési érték 3-mal nagyobb mint az előző esetben.

**typedef enum** Hasznalat{*BUY*,*SELL*}Hasznalat;  
  
**int** buttonbuy(Hasznalat transaction)

A következő függvényt egyaránt lehet használni a vételi és eladási szándék érzékelésére is egy enum segítségével. Ha a függvényt *BUY* paraméterrel hívjuk meg, akkor csak a vásárló gombokra kattintást észleli, ellenkező esetben meg csak az eladóakat.

Ezt a két függvényt foglalja össze a *planting()*.

**void** planting()  
{  
 **int** i = buttonbuy(*BUY*);  
 **if**(i == -1)  
 {  
 **return**;  
 }  
 **int** sorszam = goods();  
 money -= buy\_price[i-1];

Ez a függvény kiszámolja, hogy mit, és hova akarunk ültetni. Először lefuttatja az előbb bemutatott *buttonbuy(BUY)* függvényt, ami visszaadja annak a terménynek a sorszámát, amit majd el akarunk ültetni, majd megvárja, hogy kattintsunk egy mezőre, majd visszaadja annak a sorszámát is.

**void** bed(**int** x, **int** y, **int** i)  
{  
 **int** d = (**int**)(agyas\*SCREEN\_WIDTH);  
 SDL\_Rect rect = { x, y, d, d };  
 //printf("\n %d %d \n", x, y);  
 SDL\_SetRenderDrawColor(renderer, 255, 255, 0, 255);  
 SDL\_RenderCopy(renderer, textures[i], NULL, &rect);  
}

Ezeket az értékeket átadja a *bed()* függvénynek, ami majd azt fogja csinálni, hogy a megfelelő adatot változtatja a .txt fájlban eltároltak közül, hogy a játékciklus következő lefutáskor a megfelelő mérettel és terménytípussal kirajzolhasson egy textúrát a képre a renderer.

Egyelőre csak addig rajzolja ki a megfelelő képet, ameddig a bal egérgombot egy helyben lenyomva tartjuk.

Az főprogram, a *main.c* pedig így néz ki:

#include "global.h"  
#include "game.h"  
#include "textures.h"  
  
**int** main( **int** argc, **char** \*\*argv ) {  
  
 init(); // inicializalas  
 loadImage(); // kepek betoltese memoriaba  
  
 bool running = true;

Itt elkezdem azokat a függvényeket meghívni, amiket egyszer kell csak a játék folyamán. Inicializálom a képernyőt, betöltöm a képeket, és létrehozok egy boolean változót ami akkor igaz, ha a játék fut.

**while**( running )  
{  
 scan(); // adatok beolvasása  
 doRender(); // renderelek mindent  
 score(); // a pénz és a takarmányok kiírasa  
 planting(); // mit es hova akarok ultetni?  
 send(); // adatok kiírása .txt-be  
  
 SDL\_RenderPresent(renderer);  
 SDL\_Delay(10); // fps problémák miatt  
  
 **while**( SDL\_PollEvent( &windowEvent ) != 0 )  
 {  
 **if**( windowEvent.type == *SDL\_QUIT* )  
 {  
 running = false;  
 **break**;  
 }  
 }  
}

Ezt követi a játékciklus, ami addig fut, amíg ki nem ikszeljük az ablakot, vagy nem nyomunk Alt+F4-et. Ezen belül futnak a már bemutatott függvények, ahogy a programrészleten is látszik.

// Close and destroy the window and the renderer  
SDL\_DestroyWindow(window);  
SDL\_DestroyRenderer(renderer);  
  
// Clean up  
SDL\_Quit();

Ezt már csak az SDL utáni takarítás követi.

**Amik még hátra vannak:**

A *bed()* függvény nem egyből kirajzol, hanem a .txt fájlban változtat értékeket, majd később annak megfelelően rendereli be az elültetett növényeket, így folyamatosan meg lesz jelenítve

Folyamatos növekedést adni a terményeknek

Öntözést, elrohadást megoldani

Az eladást meg kell még írni hiba mentesen

Menüt csinálni a játéknak

Pintér Tamás

JY4D5L