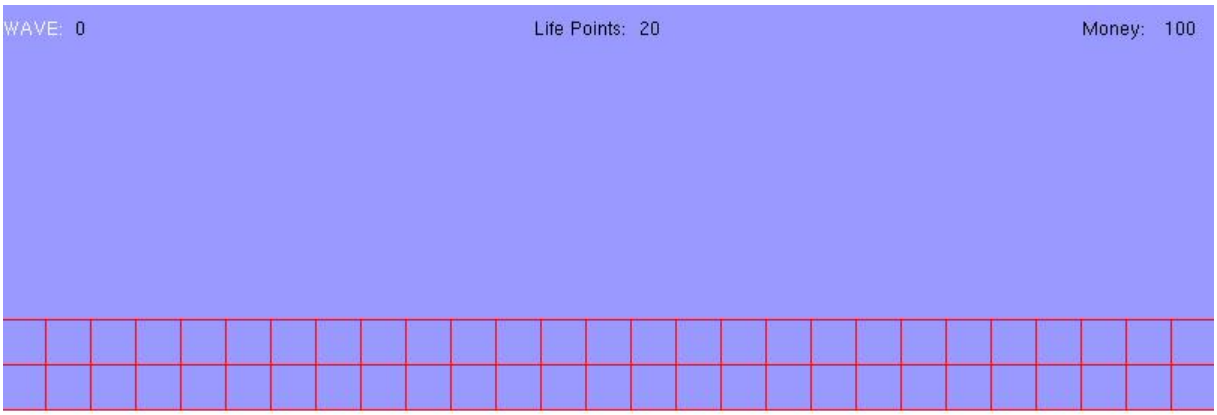
FELHASZNÁLÓI DOKUMENTÁCIÓ

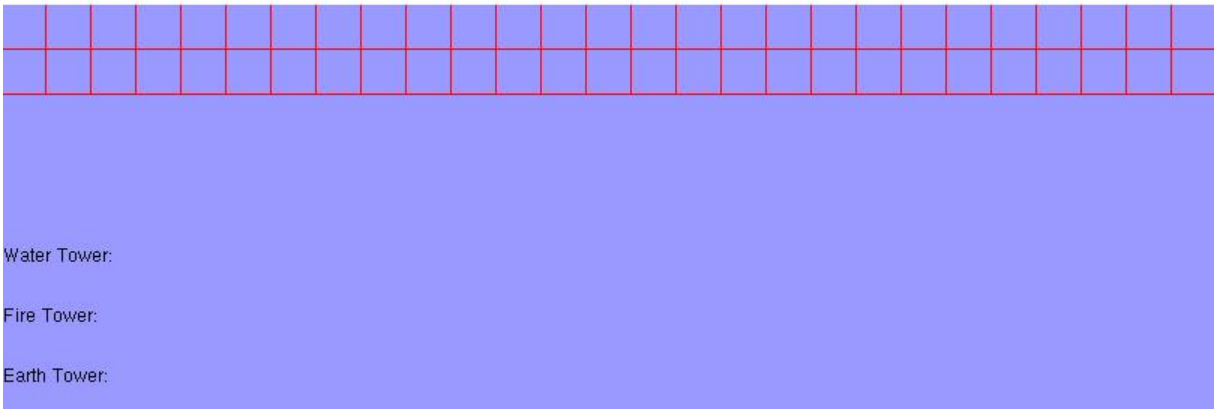
WAVE: Jelzi, hogy melyik hullámban vagyunk éppen és milyen fajta ellenséggel kell megküzdenünk.

Life Points: Kijelzi, hogy mennyi életünk maradt ha a golyózáporon keresztül jutna egy ellenség akkor egy életünk elveszik.

Money: Kijelzi, hogy mennyi pénzünk van így ebből tudunk gazdálkodni amikor tornyokat vásárolunk.

A pálya alatt látszik három felirat Water tower, Earth tower és Fire tower. Ezekből a tornyfajtákból lehet letenni a pálya négyzethálóval kijelölt részére.





Irányítás

Első lépésben ki kell választani milyen tornyot szeretnénk lerakni. Ezt úgy tudjuk megtenni, hogy rákattintunk a valamelyik toronyfajtára.

Miután kiválasztottuk a megfelelő tornyot rákattintunk valamelyik négyzetre. Ekkor lehelyeztünk egy tornyot és a pénz mennyisége is csökkenni fog.

Ha leraktunk minden tornyot amit szerettünk volna vagy elfogyott a pénzünk elindíthatjuk a hullámot. A WAVE gombra kattintva ezt megtehetjük.

Miután elment az összes ellenség a program kiértékeli mennyi pénzt nyertünk illetve mennyi életünk maradt majd folytathatjuk a játékot egy új hullám elindításával.

FEJLESZTÉS

Első felmerülő probléma a képfrissítési frekvencia beállítása volt a Game Loop-ban. Erre szerencsére voltak előre elkészített függvények így nem kellett kitalálni új algoritmust.

Mérföldkövek

A fejlesztés során a következő főbb feladatokat kellett megvalósítani.

1. A lövésnél legyen lövedék ami mozog a cél felé, és amikor eltalálja akkor vesszük le a hpját és kezdünk újra lőni

2. Egér vezérlés, azaz lehessen tornyot azzal lerakni, wavet elkezdeni

3. Tornyot csak a megfelelő pozícióra lehessen rakni, és ne lehessen rakni oda ahol már van

4. Tornyokból és ellenségekből is lehessen többet létrehozni valamilyen dinamikus adatszerkezetben (vagy ellenségnél lehet tömb is elég)

5. Fájlból kérje be az ellenségeket

6. A játék játszhatóvá tétele, azaz lehessen nyerni ha elfogynak a wavek a fájlban, veszíteni ha elfogy a hp, pénzt szerezz a szörnyek megöléséért és vonj le a tornyok vételéért stb

7. Több torony és ellenségnél pedig 1 torony csak 1 ellenfelet lőjön, mégpedig a legelsőt (erre szerintem lesz valamiféle algoritmus ötletem majd csak előtte kell h több torony és enemy lehessen tárolva)

Adatok beolvasása fájlból

Ennél a részfeladatnál az volt a lényeg, hogy mindig egy sort olvassunk be egy függvényhívásra. Illetve, hogy az adott sorban lévő ellenséghez tartozó adatokat kinyerjük és a konstruktort a megfelelő adatokkal hívjuk meg. Ezt az strtok_s függvény segítségével tettük meg. A függvény minden hívás után megvizsgálja, hogy a fájl végén vagyunk-e és a megfelelő értékkel tér vissza.

Pálya kirajzolása

A pálya kirajzolását a drawMap függvény végzi el. A for ciklusban végig lépkedünk amíg el nem jutunk a maximum oszlopszámig. Minden oszlopban egy for ciklus végzi el a négyszögek kirajzolását ami az aktuális x,y koordinátán helyezünk el.

Torony osztály

A Tower osztály a tornyok létrehozására alkalmas. Öt tagváltozóval: Type,Range,Damage,RateOfFire,x,y amiket le lehet kérdezni és be lehet állítani a megfelelő getter metódusokkal. Az osztály tartalmaz egy kirajzol metódust is ami egy négyzet formájában jeleníti meg a tornyot. Illetve egy „loves” metódust ami megnézi, hogy az ellenség benne van-e a torony hatósugarában illetve, hogy az ellenség élete nagyobb-e mint nulla.

Game osztály

A játék létrehozására készült. Tartalmaz Money, LifePoints, ActualWave, Status tagváltozókat és ezek getter függvényeit. Illetve további függvényeket amik a pénz mennyiségét növelik az életet csökkentik az aktuális hullámot állítják be. Illetve egy aktuális információkat képernyőre kiíró függvényt.

Enemy osztály

Az ellenségek létrehozására alkalmas osztály. Tartalmazza az ellenség nevét, életét, sebességét, értékét x és y koordinátáját. Tartalmaz getter metódusokat a kirajzoláshoz szükséges metódust és az aktuális hp-t kiíró metódust.

Szöveg kiíratása OpenGL segítségével

Az egyik felmerülő probléma a grafikus szöveg kirajzolása volt. Ezt a setFont(GLUT_BITMAP_HELVETICA_12) és drawstring(-1.0, 0.9, 0.0, "WAVE") függvényekkel lehetett megoldani.

Egérvezérlés megvalósítása

A glutMouseFunc által ciklikusan lefuttatott mouse függvényben van megvalósítva a vezérlés. Lekérdezzük az egérkurzor x és y koordinátáját, hogy tudjuk melyik területre kattintott a felhasználó. Így beállíthatjuk a vezérléshez fontos megfelelő változókat.

GLUT függvények

glutInit: Eljárás ami inicializálja a GLUT libraryt. Paraméterei megegyeznek a main függvény paramétereivel.

glutInitDisplayMode: A képernyő megjelenítési módját specifikálja.

glutInitWindowSize: Az ablak méreteit specifikálja pixelben.

glutCreateWindow: Megnyitja az ablakot a fenti rutinokban beállított jellemzőkkel.

glutDisplayFunc: Azt a függvényt specifikálja, amelyet akkor kell meghívni, ha az ablak tartalmát újra akarjuk rajzoltatni.

glutReshapeFunc: Azt a függvényt specifikálja, amelyet akkor kell meghívni, ha az ablak mérete vagy pozíciója megváltozik.

glutMouseFunc: A függvényt specifikálja, amely egy egér gomb lenyomásakor illetve elengedésekor hívódik meg. Button paraméter mondja meg melyik gombot nyomtuk meg, a state, hogy milyen állapotban van az adott gomb lenyomott vagy felengedett, illetve a x,y a kurzor pozícióját adja meg.