Házi Feladat

Programozás alapjai 2.

Végleges

Foki Lénárd

Tartalom

[1. Feladat röviden 3](#_Toc164632383)

[2. Terv 3](#_Toc164632384)

[1. Könyvtárak 3](#_Toc164632385)

[2. Fájl struktúra 3](#_Toc164632386)

[3. Fázisok 4](#_Toc164632387)

[4. UI komponensek 4](#_Toc164632388)

[5. Menü 5](#_Toc164632389)

[6. Hőskiválasztás 6](#_Toc164632390)

[7. Pálya 7](#_Toc164632391)

[8. Entitások 8](#_Toc164632392)

[9. Hősök 8](#_Toc164632393)

[10. Hős a hős ellen párharc algoritmus leírása 9](#_Toc164632394)

[11. Más entitások 9](#_Toc164632395)

[12. játék fázis 10](#_Toc164632396)

[13. Lépések 11](#_Toc164632397)

[3. Bemenet és kimenet 12](#_Toc164632398)

[4. Tesztek 13](#_Toc164632399)

# Feladat röviden

A feladat egy a League of Legends (röviden csak: LoL) nevű játékot alapul vevő stratégia felülnézetes játék. Két játékos játszik egymás ellen, de ugyanaz a játékos is játszhat magával, ilyenkor inkább egy szimuláció szerepet tölt be a játék, ami hasznos lehet, hogy stratégiákat dolgozzon ki az eredeti LoL játékhoz. Persze a LoL egy nagyon komplex játék, ezért ennek egy egyszerűbb változatát készítjük el, de a program nyitott lesz a komplexitás bővítésére, ezáltál a jövőben közelebb hozva a szimulációt a valósághoz. Ilyen bővítés lehet például, hogy hasznos statisztikákat mutasson a játék egy meccs elemzése után, vagy másféle lépéseket is végre tudjon hajtani a hős, vagy akár vissza lehessen játszani egy meccset.

A játék nagyban leköveti a LoL felépítését, tehát 10 hős öt az öt ellen küzd egymással a pályán és az a csapat nyer, amelyik az ellenség bázisán lévő erődítményt ledönti. A különbség annyi, hogy az eredeti játékban mindenki egy hőst irányít, itt viszont egy-egy játékos az egész csapatot irányítja, de mivel ez túl megterhelő lenne az eredeti játék mechanikájával, ezért a sakkhoz hasonlóan, felváltva lépnek a karakterekkel, úgy hogy van egy megadott idejük gondolkodni és elvégezni a lépéseket.

A másik fontos része a játéknak, hogy nincsenek kőbe vésve a hősök/tárgyak adatai, hanem mind testre szabható, változtatható a program bemeneteinek megváltoztatásával.

Három fő fázisra (ezeken belül lehetnek alfázisok) lehet osztani a programot az állapota alapján, ezek a következők:

* menü
* hőskiválasztás
* mérkőzés

A menüből lehet eljutni először a hőskiválasztás fázisba, onnan pedig a mérkőzés fázisba. Ha nem megy végbe a hőskiválasztás, vagy a mérkőzés véget ér, akkor onnan a menübe kerül vissza a program.

# Specifikáció változtatások

A fejlesztés során találkoztam különböző problémákkal, amik a terv megírásakor még nem tűntek fel, így a specifikáció kisebb megváltoztatására van szükség.

* 6 darab minion helyett csak 3 lesz
* 3 torony van egy ösvényen és nem 4
* nincs x darab minion ütése lépés, hanem a sima támadás lépéssel lehet minionokat ütni
* a hősök és tárgyak listájának elmentésére nincs szükség redundánsan minden kimenetnél
* a kimeneti fájl névütközése miatt egyszerűbb és kényelmesebb a felhasználónak, ha csak a prefixét adja meg a fájlnévnek, a vége pedig a meccs kezdetei dátum lesz, így biztosítva egyedi azonosítót, ebből következik hogy a dátum és idő nem a fájl első sorában lesz, hanem a fájl nevében
* a két hős elleni támadás kiszámítása is máshogy történik
* mindig a kék csapat kezd, mivel így működik az eredeti játékban is, így természetesebb

Ami sajnos technikai okok miatt nem valósulhatott meg azok a következők:

* a baron effektus nem a minionoknak add bónuszokat, hanem úgy működik mint a sárkány bónusza
* a sárkány effektust csak az kapja meg aki megöli
* a bokorban egyelőre csak esztétikai elem, és nem működik másképp a láthatóság

# A projekt elkészítése

A projekt megvalósításáról röviden.

## Fáziskezelés

Először is egy fáziskezelő osztályt kellett írnom, amivel könnyedén és memóriahiba mentesen tudok váltogatni a fázisok között, ez a state manager osztály. A rajta megvalósított metódusokkal új fázist lehet előállítani, vagy a jelenleg aktív fázist bezárni. Emellett ami fontos volt a megvalósításban, hogy egy úgynevezett main loop cycle (handle events, update és draw függvények) az végigfusson, és csak azután legyen fázisváltás.

## UI

Ezután a UI megtervezése történt, létrehoztam a szükségem komponenseket (u.n. gridelement), illetve egy olyan komponenst is, a grid-et, amellyel a több gridelement-et megfelelő pozícióban, oszlopban vagy akár sorban elhelyezi. A gombok megvalósításánál a kattintás lekezelésére callback függvényt használtam, vagyis a gomb nem tudja amikor létrehozzák, hogy kattintásnál mit csináljon, de meg lehet neki adni, hogy milyen függvényt hívjon meg ilyenkor.

## Menu

A menüben nincs sok érdekesség, a gombokat létrehoztam, illetve a szöveges beviteli mezőket, ahova a program be- és kimeneteinek a nevét lehet megadni.

## Draft

A specifikáció szerint lett megvalósítva, kisebb eltérés annyi, hogy a kitiltott hősök nem egy sorban vannak, hanem alul egy oszlopban.

## Entitások

A különböző entitások az alap entitás osztályból származnak, így egy heterogén kollekcióként lehet ezeknek egy halmazát kezelni.

A hősök támadásánál kellett változtatni a matekon, hogy jól jöjjön ki a nyertes.

A szám jelzi, hogy melyik hőshöz tartozik

Ezután pedig az első hős relatív ereje:

Ekkor az esély százalékos megkapásához a fenti számot még fel kell szorozni.

Ekkor legenerálunk egy véletlen számot, és ha 0 és az első entitás nyerési esélye közé esik, akkor az első entitás nyer, ha pedig az első entitás nyerési esélye és 100 közé esik, akkor a második.

## Pálya

A pálya létrehozásakor a megfelelő fájlból olvas (ami nem felhasználói bemenetet, tehát feltételezve van, hogy helyes, mivel a felhasználó nem változtathatja). Ekkor a bemeneti kódolás alapján létrehozza a mezőket a megfelelő entitásokkal.

## Szimuláció

A szimuláció megvalósítása úgy történt, hogy bizonyos időközönként (jelenleg három másodperc) felszólítja az entitások, hogy tegyék meg a lépéseiket, és ezt mutatja a kijelzőn.

# Tesztek

A tesztekben legfőképp a beolvasás lett tesztelve, hogy a felhasználói bemenetet megfelelően tudja lekezelni a program, és ne szálljon el. Az érdemi funkcionalitása a játéknak az entitások egymás közötti harca, legfőképp a hősöké, illetve a map létrehozásakor nem ütközik-e hibába.

* Hősök beolvasása fájlból, és leellenőrzése annak, hogy a megfelelő mezők a megfelelő hős tulajdonságot töltötték fel
* tárgyak beolvasása
* két hős csatája, és hogy mennyi az esély arra hogy az egyik nyer
* a hőssel a mapon lépni egyet, és megnézni hogy jó helyre került-e