Pannon Egyetem

Műszaki Informatikai Kar

Villamosmérnöki és Információs Rendszerek Tanszék

Programtervező Informatikus BSc szak

SZAKDOLGOZAT

Tower Defense Stílusú Többszemélyes Játék Fejlesztése

Géringer Ábel Róbert

Témavezető: Dr. Guzsvinecz Tibor

2021

FELADATKIÍRÁS

- Téma: **Tower defense stílusú többszemélyes játék fejlesztése**
- Cím: **Tower defense stílusú többszemélyes játék fejlesztése**
- Végleges cím:
- Oktatók: **Dr. Guzsvinecz Tibor**
- Jelentkezés dátuma: **2021.09.07. 13:14:44**
- Elfogadás dátuma: **2021.09.22. 7:54:26**
- Beadás dátuma:
- Bemutatás dátuma:
- Védés dátuma:
- Külső téma: **Nem**
- Nyelv: **magyar**
- Szervezeti egység: **MIVIR**
- Szakdolgozat státusz:
- Oktatói vélemény: **Támogatott**
- Beosztás eredménye: ✔
- Elfogadó: **Bálint Adrienn**
- Visszavonás dátuma:
- Védés eredménye:
- Titkos: **Nem titkos**
- Url:
- Sorszám: **SZD21092207546484**

- Leírás:

Tower Defense témájú játék. A lényege, hogy egy (vagy több) kijelölt úton jönnek ellenségek és azokat tüzelő tornyok megvásárlásával és elhelyezésével a pályán kell megvalósítani. - Hang: saját zenét és hangeffektek - Multiplayer: egyszerre 2 játékos játszhatna és a pálya fel lenne osztva, hogy ki melyik részen építhet - WaveSpawner: a hullámok random mintában jönnek - Nehézségi fokozatok - Pályaszerkesztő: A játékosok megszabhatnak közel mindent a pályán, mennyi pénzzel kezdjenek, mekkora legyen a pálya, milyen dizájn elemeket tennének rá, mennyi irányból jöjjenek a hullámok, mennyi helyre érkezzenek, mennyi élet legyen, milyen gyakran jöjjenek a hullámok stb... - Custom Maps: az előző pontból jön ez a lehetőség, hogy elmentsék az eddig készített pályákat és bármikor játszhassanak - Saját modellek Kérésre kiírt téma.

NYILATKOZAT

Alulírott Géringer Ábel Róbert diplomázó hallgató, kijelentem, hogy a szakdolgozatot a Pannon Egyetem Villamosmérnöki és Információs Rendszerek Tanszékén készítettem Programtervező informatikus BSc diploma (BSc in Computer Science) megszerzése érdekében.

Kijelentem, hogy a szakdolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy a szakdolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Veszprém, 2021. dec. 03.

aláírás

Alulírott Dr. Guzsvinecz Tibor témavezető kijelentem, hogy a szakdolgozatot Géringer Ábel Róbert a Pannon Egyetem Villamosmérnöki és Információs Rendszerek Tanszékén készítette mérnök informatikus BSc diploma (BSc in Computer Science) megszerzése érdekében.

Kijelentem, hogy a szakdolgozat védésre bocsátását engedélyezem.

Veszprém, 2021. dec. 03.

aláírás

KÖSZÖNETNYILVÁNÍTÁS

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Guzsvinecz Tibornak a bizalmáért, mellyel lehetőséget biztosított számomra a szakdolgozatom elkészítéséhez. Kiváló szakmai tudásával, tapasztalatával és átfogó látásmódjával mindvégig segítségemre volt a munkám során. Szeretnék köszönetet mondani családomnak, akik megértésükkel és odaadó segítségükkel mindig támogattak. Nekik köszönhetem, hogy idáig eljuthattam.

TARTALMI ÖSSZEFOGLALÓ

A szakdolgozat Tower defense stílusú többszemélyes játékot mutat be. Az alkalmazás a Unity Engine segítségével csináltam C# nyelven, melyet Microsoft Visual Studio-ban írtam. A fejlesztés Windows 10 operációs rendszeren végeztem. A program mind egy- és többjátékos módban játszató, melynek lényege a „tower” megvédése. Az ellenség egy adott útvonalon, hullámokban, tart a torony felé, hogy elfoglalhassák. A játékos(ok) feladata ezt megakadályozni, különböző védő tornyok átgondolt elhelyezésével, valamint fejlesztésével a pályán.

Az alkalmazás funkciói:

- Állítható nehézségi fokozatok
- Pályaszerkesztő
- Többjátékos mód
- Grafikai és egyéb beállítások

Dolgozatom ismerteti a felhasznált technológiákat, és bemutatja az alkalmazás megtervezésének folyamatát. Ismertetem a fejlesztett program működését, majd értékelem az elkészült alkalmazást. Végül szót ejtek a továbbfejlesztési lehetőségekről.

Kulcsszavak: Unity Engine, Tower Defense, C#, Microsoft Visual Studio, játék

ABSTRACT

My thesis is about a Tower Defense styled multiplayer game. The application is made by the Unity Engine and was written in C# in Microsoft Visual Studio. The development was going under Windows 10 operation system. The game can be played both alone and with a friend. The players mission in the game is to defend the tower from the incoming enemy waves by strategically placing and also upgrading turrets on the map to prevent them from occupying it.

The functions of the application:

- Multiple difficulties
- Map creator
- Multiplayer
- Graphical and other options

My thesis reviews the used technologies and presents the process of designing the application's user interface and back-end. It describes how the finished application operates. Lastly, I evaluate the work done and write about the possible improvements.

Keywords: Unity Engine, Tower Defense, C#, Microsoft Visual Studio, game

TARTALOMJEGYZÉK

0. Fejezet

0.1 Bevezetés

Szakdolgozatom egy régebbi játék stílusa adta meg az ihletet, melyet még 10-15 éve még Windows XP-n játszottam a böngészőben a Flash Player-rel 2D-ben. A „Tower Defense” stílus, mely a stratégiai játékok egy alcsoportjába tartozik, aranykora egészen az 1990-es évekre vezethető vissza, az Árkád játékokhoz, amikor megjelent a „Rampart” 1990-ben, melyet az akkor ismertebb Atari Games adott ki. Mindig is érdekelt a játékfejlesztés és gondoltam ez egy remek alkalom lesz arra, hogy elsajátíthassam annak alapjait, miközben egy kedvenc műfajomból csinálok játékot egy kis extrával. Szerettem volna valami kis pluszt is belevinni a játékba, így a játékot 3D-ben csinálom, melyben lesz lehetőség online, barátokkal együttesen visszaverni az érkező ellenségeket. A játékon belül van lehetőség egy egyszerű kampány módra, melyben az ellenségek egy adott számú körön át érkezik, valamint van lehetőség olyan módra melyben csakis kizárólag a mi ügyességünkön múlik, hogy meddig bírjuk, ebben a lehetőségben mindig lesz egy következő hullám, melyben helyt kell állni. Ennek megvalósítására az Unity 3D játékfejlesztő környezetet választottam, ugyanis ez a program az egyik legnépszerűbb jelenleg a piacon, vezető cégek ezzel a programmal dolgoznak hosszú évek óta, mely folyamatos fejlesztés alatt van.

0.2 Tower Defense Stílus

A „Tower Defense” a stratégiai játékok egyik alkategóriája sok más osztállyal együtt, csak hogy egy pár közismertebbet megemlítsek: TBS (Turn-Based Strategy), RTS (Real Time Strategy), MOBA (Multiplayer Online Battle Arena). A TBS egy körökre osztott játék, akár a sakk. Minden egyes körben egy valaki végezheti el stratégiai lépéseit. Az RTS egy olyan stratégiai játék, melyet nem körökre bontva kell játszani, hanem valós időben történik minden. A MOBA lényege, hogy 2 csapat játszik egymás ellen egy előre megadott pályán. Sokan gondolják laikusként, hogy egyetlen pályán játszani minden alkalommal nagyon repetitív, viszont ebben az esetben más elemekkel teszik változatossá a játékokat. Ha példának vesszük a DOTA2 című játékot, minden egyes játék teljesen különböző, bár egy pálya van csak, melyet úgy érnek el, hogy 123 karakterrel lehet játszani, melyeknek különböző képességei, tulajdonságai, szerepei vannak a játékban.

0.3 Hasonló Stílusú Játékok

Ebben a stílusban sok játék jelent meg az évek során, melyek nagy népszerűségnek is örvendenek, csak hogy néhányat említsek, itt van talán a legismertebb a genre-ban a „Plants Vs zombies”melyet az Electronic Arts (EA) leányvállalata a „PopCap Games” fejlesztett. Az első része a játéknak még 2009-ben jelent meg, mely egy 2D-s játék, amit szinte minden platformra kiadtak a fejlesztők. Emellett érdemes megemlíteni még egy jelenleg is nagy népszerűségnek örvendő játékot a „Bloons TD”-t, mely szintén 2D-ben készült először, 2007-ben, valamint egy személyes kedvencemet a „Dungeon Defenders”, mely merőben eltérő az előbb említett 2 játéktól. 3D-ben készült már az első része is még 2011-ben. Nagy újdonság volt akkor az új mechanika, miszerint egy hőst kellett irányítanunk a térképen úgy nevezett „Third Person View”-ban (A karakterünk hátát látjuk és irányítjuk) a több választható és nagyban különböző karakterek közül és vele építeni meg a különböző védőtornyokat és beszállni a védésbe.

1. Fejezet

Felhasználói Dokumentáció

1.1 A program használata

A játékot Windows operációs rendszeren futtathatjuk. Telepíteni nem kell, a TowerDefense.exe fájllal indíthatjuk.

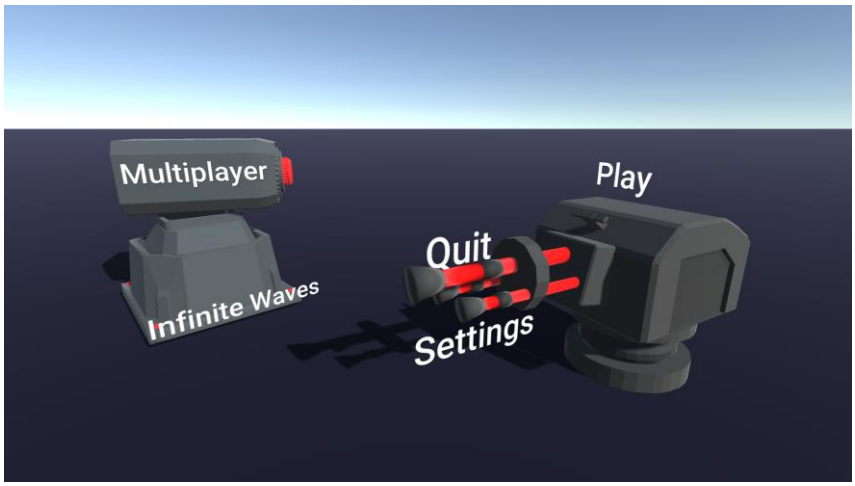
1.2 Irányítás

- jobbra: jobbra nyíl, d, vagy az egér jobbra húzása
- balra: balra nyíl, a, vagy az egér balra húzása
- fel: felfele nyíl, w, vagy az egér felhúzása húzása
- le: lefele nyíl, s, vagy az egér lefele húzása
- közelítés/távolítás: görgővel

1.3 Menürendszer

A program indulásakor a főmenüben 5 menüpont közül lehet választani:

- **SinglePlayer:** ezzel a gombbal mehetünk el a pályakiválasztó menübe
- **Multiplayer:** ezzel a gombbal mehetünk el a többjátékos módhoz
- **Infinite Waves:** ezzel a gombbal jutunk el a végtelen hosszú játékmódba
- **Settings:** ezzel a gombbal mehetünk el a beállítások menübe
- **Quit:** ezzel a gombbal léphetünk ki az alkalmazásból



1.4 Beállítások

1.4.1 Graphics

- **Resolution:** kiválaszthatjuk a játék ablakának felbontását.
- **Fullscreen:** kiválaszthatjuk, hogy teljesképernyőn szeretnénk-e játszani
- **Texture:** kiválaszthatjuk a játék minőségét
- **Anti-Aliasing:** kiválaszthatjuk az élsimítás minőségét
- **V-sync:** kiválaszthatjuk a Vertikális frissítést

1.4.2 Volume

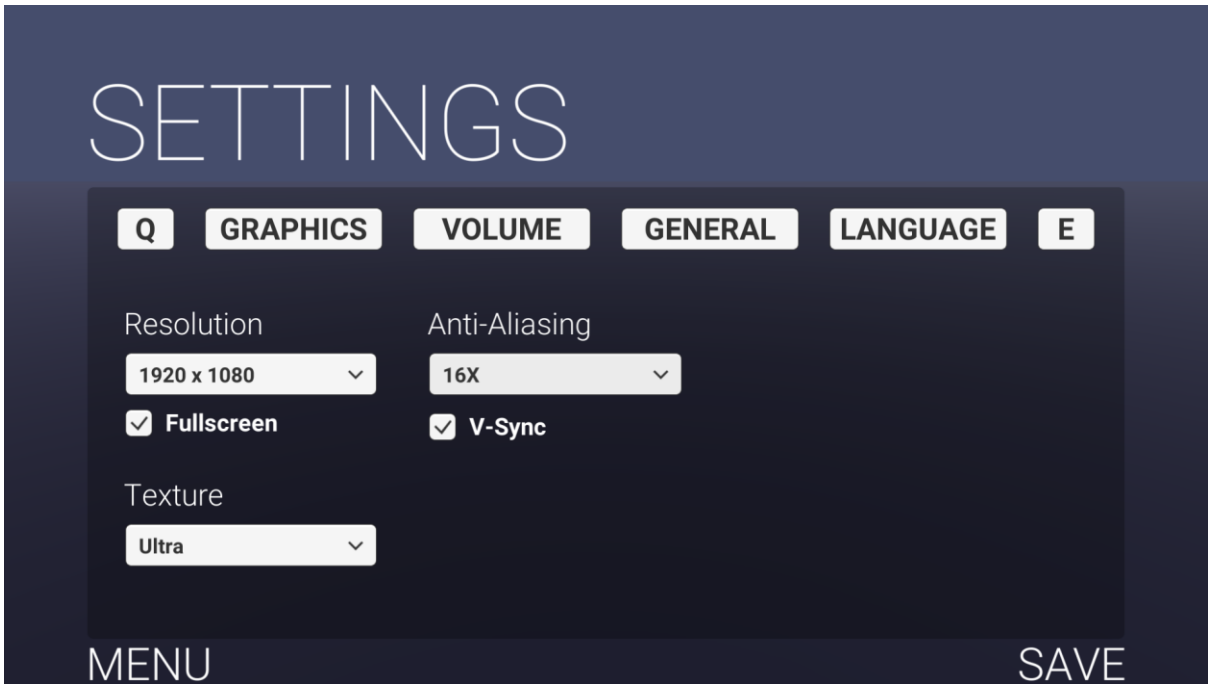
- TBD

1.4.3 Display

- TBD

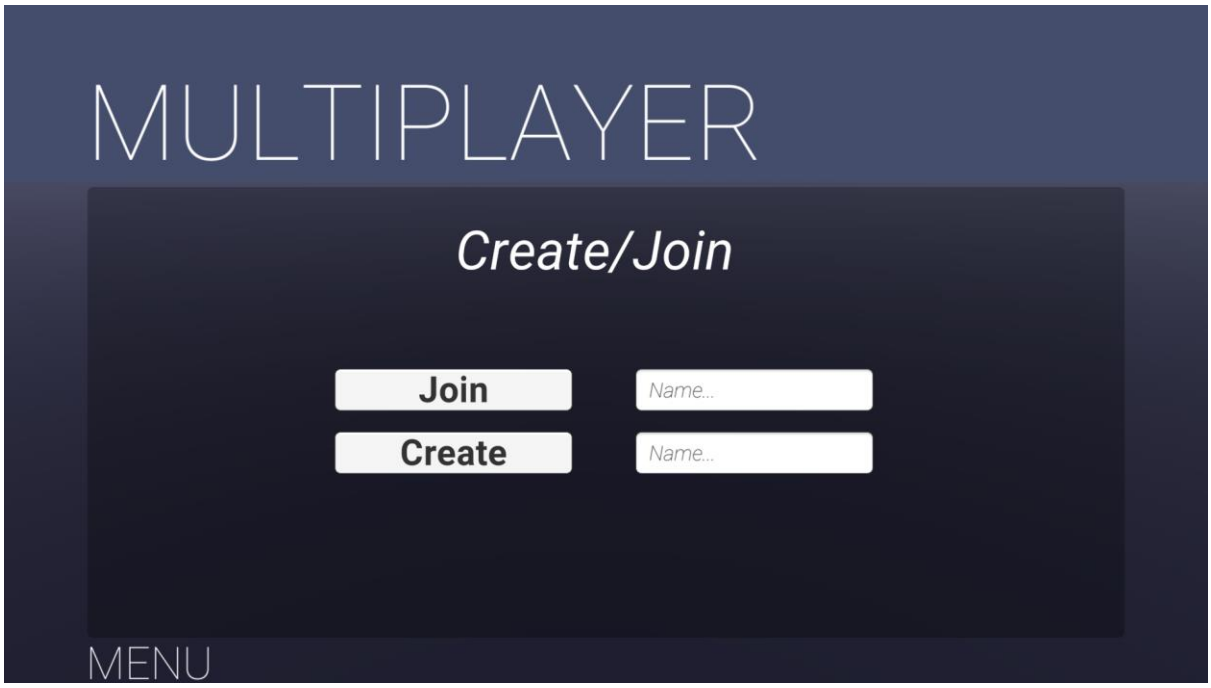
1.4.4 Language

- TBD



1.5 Multiplayer

Ebben a menüben tudunk bárkivel együtt játszani, akinek szintén van hozzáférése az internezhez. Annyit kell tennünk, hogy beírjuk a szobánk nevét, amelyben szeretnénk játszani és létrehozzuk a create gombbal, amellyel rögtön elindítjuk a játékot. Ha már létre van hozva a szoba és csak be szeretnénk csatlakozni a barátunkhoz, akkor a szobanév beírása után csak rá kell kattintanunk a join gombra, mely egyből be fog dobni minket a játékba. Ebben az opcióban, egyelőre csak egy pályán tudunk játszani online, de később lehetőség lesz arra, hogy egy lobbyban kiválaszthassuk a pályát is.



1.6 Szabályok

A játék célja, hogy a hullámokban érkező ellenségeket, különböző lőtornyokkal megsemmisítsük és meggátoljuk, hogy elérjék a tornyot. A megsemmisített ellenségekért a játék pénzel jutalmaz meg, melyet újabb tornyokra, vagy pedig fejlesztésekre költhetünk. Minden egyes pályán, meg van adva, hogy a játékos mennyi ellenséges egységet engedhet oda a várhoz a pálya tetején, bal oldalán, hogy hányadik hullámnál tartunk, valamint a pálya jobbalsó sarkában látjuk, mennyi idő van még vissza a következő csapat ellenségig. A játékablak alján, kiválaszthatjuk a nekünk kellő védőtornyokat, melyek ára, valamint feladata is sok mindenben különbözik! A játékban szereptő tornyok modelljeit, egy ingyenes Asset Pack-ból szereztem (https://devassets.com/assets/tower-defense-assets/).

- **Rocket Launcher:** rakétákat lő, melynek robbanása következtében, egy kisebb sugarú körön belül minden ellenséges egységet sebez. Vigyázz, sebzése nem nagy, cserébe messzire lő!

- **Leaser Beamer:** egy erős, zöld lézersugarat lő ki magából, melyel egy rosszakarót tud sebezni és lassítani is egyszerre egy kisebb sugarú körben.

- **Standard Turret:** kis töltényeket lő ki magából, melyel egy a Rocket Launcher-nél kisebb de a Leaser Beamer-nél nagyobb sugarú körben tudja sebezni az ellent.

A játék során figyelni kell, hogy a pénzedet okosan használd fel! Nem mindegy, hogy egy tornyot fejlesztesz, vagy esetleg veszel egy másikat, valamint nem mindegy, hogy a pálya mely pontjain helyezed el melyik tornyot! A játékban többféle ellenség van, amikből több különböző erősségű is van, melyekre érdemes odafigyelni a játék előrehaladtával.

- **Standard enemy:** ez egy alap ellenség, melynek nincsen kiemelkedő tulajdonsága, élete, sebessége, valamint a vele elnyert pénz mennyisége is a többi között átlagos. Ismertető jele, hogy a kék 3 különböző árnyalatát veszi fel.
- **Speedy:** ennek a lénynek nagy jellemzője, hogy sokkal gyorsabb az összes többinél, és egy kicsivel többet is kapni belőle, és ennek van a legkevesebb élete az összes többi közül. Onnan lehet felismerni, hogy sárga.
- **Heavy:** az egyik leglassabb fajta rosszakaró, melyért kicsivel több pénz jár, viszont annál nehezebb megsemmisíteni, mivel több élete van, mint az eddigieknél. Felismerni onnan lehet, hogy piros.
- **BOSS:** a legnehezebb ellenfél az egész játékban, nagyon sok pénzt lehet nyerni leküzdése után, nagyon sok élete van és gyorsabb, mint egy standard enemy. A játékban csak bizonyos hullámokban kerül elő, hisz fel kell készülni rá a játékosnak. Arról ismerni fel, hogy ez a legnagyobb féle ellenség a játékban.

▪

2. Fejezet

Fejlesztői Dokumentáció

2.1 Felhasznált Technológiák

Unity játékfejlesztő környezetben készült a program, a scipt-eket pedig az JetBrains Rider-ben írom. Választásom azért jutott ezekre, mivel az egyetem biztosít lehetőséget a JetBrains termékeinek használatára, melyeket nagyon szeretek és nagyon kényelmes használni őket, legyen akár jelen esetben a C# scriptek írása, vagy Java, Kotlin, de akár PHP is. Az Unity-t pedig a hatalmas közössége miatt választottam, valamint hogy a kezdő Unity fejlesztők is könnyebben tudnak elindulni. A projektet verziókezelése Github segítségével valósult meg, mert a legtöbb nagy cég is ezt használja és biztonságos, valamint könnyedén követhető a projekt fejlődése, hiba esetén egy korábbi verzió visszaállítása. A többjátékos mód a Photon PUN mely egy Realtime Cloud service, hogy a világon bárki játszhasson egymással, ha van hozzáférése internethez. Ez a technológia nagyon sokoldalú és kiváló integrációja van az Unity játékfejlesztő környezettel, bármely „Room Based” (kisebb online szobákra alapozó) játék megteremtéséhez. Választásom azért került a Photon PUN-ra, mivelhogy ez egy olyan ingyenes lehetőség a többjátékos mód megvalósítására, melyben, ha meghaladnám az ingyenes keretet is véletlen, sem kapnék elsőre egy csekket, amit be kellene fizetnem. A legbiztosabb alapja a Photon sebességének a „client-2-server-architecture”, ez egy olyan modell melyben a szerver irányítja az erőforrások nagy részét és a szolgáltatásokat pedig a kliens. Más ismertebb elnevezései a „networking computing model” vagy „client/server network”, mert minden a hálózaton keresztül történik. Ha ez mind nem volna elég, akkor lehetőség van „Cross Platform”-ra is, tehát nem számítana, hogy milyen operációs rendszeren, vagy eszközön játszunk, van lehetőség együtt, egy online szobában visszaverni az ellent.

2.2 Követelményspecifikáció

2.2.1 Funkcionális követelmények

- A programban van lehetőség különböző grafikai beállításokra, mint például:
    - Felbontás

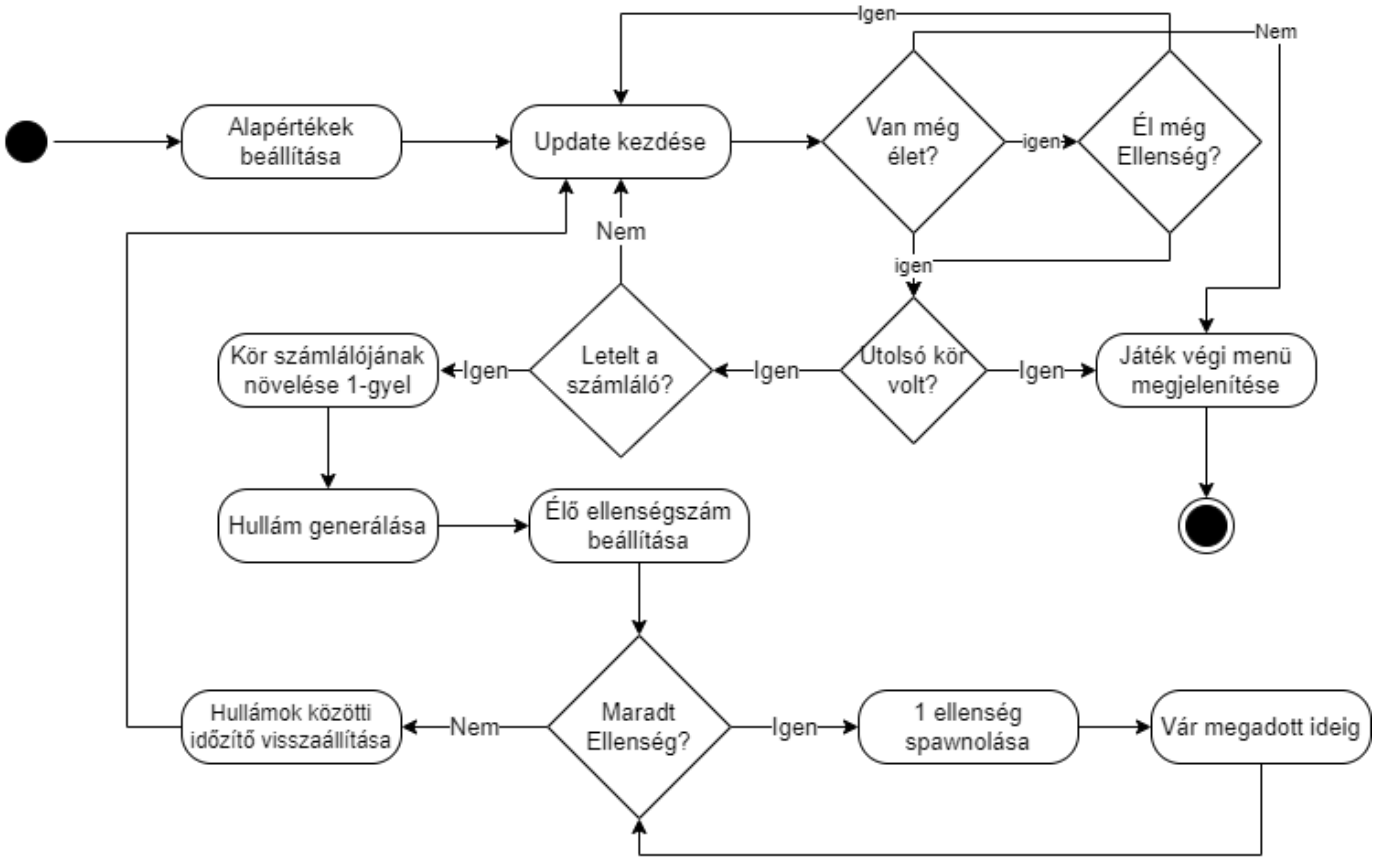
    - Teljes képernyős mód (Fullscreen és Windowed)
    - Textúrák minősége
    - Élsimítás (Anti-aliasing)
    - Vertikális szinkronizáció (V-sync)
    - Nyelv (angol vagy magyar)
- A programban a játékos tud egyedül és barátokkal is játszani
- A programban lehet saját pályát készíteni
- A saját pályákat el lehessen menteni
- A programban lehet random generált pályán is játszani
- A programban az ellenségek véletlenszerűen jönnek a nehézségi beállítástól függően
- A programban az egyjátékos módban, több pálya is van melyeket csak sorban lehet játszani elsőre.
- A programban az egyjátékos módban a pályák kioldhatók legyenek csak, hogyha nyertek az előtte lévő pályán
- A programban könnyen lehessen navigálni
- A játék közben, fontosabb információk, mint a pénz, hányadik hullám és a hátralévő életek legyenek láthatók
- A játékban legyen minimum 3 féle torony, melyet a játékos letehet
- A játékban legyen minimum 8 féle ellenség
- A program indításakor az elmentett beállítások betöltődnek
- A játék közben lehessen szünetet tartani és újrakezdeni az adott pályát

2.2.2 Nem Funkcionális Követelmények

- Kis rendszerigénye legyen
- Lehessen futtatni PC-n, Linuxon és MAC-en
- A játék átlátható legyen
- Könnyen lehessen navigálni a menüpontok között
- Ne sértse meg a szerzői jogok védelmét

2.3 UML Diagramok

2.3.1 WaveSpawner Activity Diagram



2.3.2 WaveSpawner leírása

Az első lépésben inicializálni kell néhány alap változót, hányadik hullám ellenségnégnél tart a játék valamint hányan vannak pályán életben még, ezeket mind a kettőt 0-ra, egyelőre. Az update metódus, minden frame-ben, ezt a függvényt használják a legtöbbet, ha valamilyen játékon belüli scriptet írunk. Minden egyes híváskor meg kell vizsgálnunk, hogy vége lett-e a játéknak, vagy még folytathatjuk, melyhez meg kell vizsgálni, hogy van-e még ellenség, van-e még hátralévő hullám, amelyet le kell győznünk. Ha teljesen elfogytak az ellenségek, akkor nyertünk, megjelen a győzelmi menü, és eldönthetjük, hogy újra próbálnánk, haladnánk a következő pályára, vagy ki szeretnénk lépni a menübe. Abban az esetben, ha van még hullám, akkor elindul egy visszaszámláló, majd elkezdődik a spawn (valamely GameObject létrehozása a pályán). A spawn egy előre, statikusan megadott hullámot kezd el legenerálni, mely elindul a kijelölt útvonalon.

2.4 Multiplayer

A játék a Photon Pun-nal készül, mely mind online mind offline módban is nagyon hasznos, hiszen nem kell implementálni bizonyos függvényeket singleplayer és multiplayer módra külön-külön, ezzel is elkerülve a tömeges kódismétléseket. A játék indulásakor automatikusan betesz minket a játék az offline módba, illetve egy szobába melynek neve „OfflineRoom”. Ez amiatt szükséges, hogy használhatóak legyenek a játékban az „Instantiate” függvények, melyekkel az ellenségeket, illetve a tornyokat spawnoljuk, mert nem az Unity Instantiate metódusa van meghívva, hanem a PhotonNetwork Instantiate függvénye. Ezzel biztosíthatjuk, hogy online és offline is ugyan azzal a kóddal tudjunk spawnolni. Ha a multiplayer menübe megyünk, akkor a játék online módba kerül a PhotonNetwork CreateUsingSettings meghívásával, majd ezután kezdődhet a szoba létrehozása vagy a szobába való belépés. Ezeket a lépéseket a JoinRoom, CreateRoom illetve a LeaveRoom függvényeket tudjuk végrehajtani. A multiplayer során úgy tudjuk megoldani azt az akadályt, hogy mindenkinél külön történjenek meg a spawnolások, toronyfejlesztés, -eladás és ehhez hasonló szinkronizálásra szoruló scriptek, hogy a függvény fejléce felett egy sorral meghatározzuk, hogy ez egy RPC (Remote Procedure Calls) függvény a [PunRPC] attribútummal. Ez biztosítani fogja, hogy a hálózaton mindenhol szinkronizálva legyen az az adott dolog, amit a script csinál.

3. Fejezet

Jövőbeli Tervek

3.1 Multiplayer Továbbfejlesztése

Multiplayer Lobby

3.2 Pályaszerkesztő

A játékosoknak alkalma lesz arra, hogy maguknak állítsanak össze egy saját pályát, melyet elmenthetnek. Szinte mindent beállíthatnak rajta, ha szeretnének. Milyen ellenségek jöjjenek, mennyi jöjjön, mennyi érhessen el a toronyhoz, mennyi időközönként jöjjenek a hullámok, vagy a hullámokban az ellenségek, ő szeretné-e beállítani az ellenségeket vagy random legyen, mekkora legyen a pálya, milyen útvonalon menjenek az ellenségek, mekkora legyen a kezdőpénz, stb.…

3.3 Random hullámokpályák

A körök lényege az lesz, hogy sosem lesz ugyan olyan, mindig más sorrendben, időközönként és más ellenségek jönnek majd.

3.5 Beállítások

3.5.1 Nyelvi beállítások

A játékosnak alkalma lesz az angol és a magyar nyelv kiválasztására.

3.5.2 Álltalános beállítások

- Be lehet majd állítani a játék fényerejét is, hogy világosabb szobában is jól látható legyen minden és este is könnyebb legyen játszani vele.
- A maximum FPS, amivel szeretne a játékos játszani.

3.5.3 Grafikai Beállítások

- V-Sync állítható lesz külön 15-30 FPS-re is nem csak 60-ra.
- Az árnyékok minősége beállítható lesz.

3.5.4 Hang beállításai

- Fő hangerő beállítása.
- Zene hangerejének változtatása.
- különböző effektek hangerejének beállítása.

4. Fejezet

Összefoglalás