Záródolgozat

Szakképesítés neve: Szoftverfejlesztő

OKJ száma: 54 213 05

Chug és Dug

Készítette: Risztics Ádám

Budapest

[2022]

Bevezető

Először is szeretnék köszönetet nyilvánítani a tanáraimnak, akik ebben az évben tanítottak temérdek hasznos dolgot, de legfőképpen a konzulensemnek, Donátnak. Ahol tudott segítette a munkámat és tanácsokkal látott el. Emellet szeretnék köszönetet nyilvánítani a barátaimnak, hogy segítettek letesztelni a játékot és a barátnőmnek akivel közösen találtuk ki a játék koncepcióját.

Manapság ha a fiatalok ki akarnak kicsit kapcsolódni akkor leülnek a számítógép elé és előveszik a kedvenc játékaikat, van aki egyedül szeret játszani és van aki például a barátaival egy gépnél vagy hálózaton keresztül, akár idegenekkel. Sok féle játék típus létezik, mindenki megtalálhatja ami neki tetszik Akció (FPS,TPS), Ügyességi (Platformer), Kaland, Szerepjátékok (MMORPG), Stratégiai játékok (RTS) és sorolhatnám még. Mivel én is szeretek játszani és már van egy kevés tapasztalatom játékfejlesztésben úgy gondoltam, hogy megcsinálom a saját játékomat.

Felhasználói dokumentáció

1. A program általános specifikációja

A Chug és Dug nevű játék egy két dimenziós platformer játék, lényege hogy megöljük az összes ellenséget és összegyűjtsük az arannyal teli üstöket. A játékos az ellenfelet úgy tudja kiiktatni, hogy ha lelövi a kezében lévő fegyverrel vagy a fejére ugrik. Az üstöket pedig úgy tudjuk összegyűjteni, hogy egyszerűen csak belesétálunk. Van lehetőség arra is hogy egy barátunkkal közösen játszunk, de ha egyedül kívánunk játszani akkora arra is van mód.

A karakterünknek 3 szívnyi életereje van, ez azt jelenti hogy háromszor érhet hozzánk az ellenfél, ha többször ér hozzánk mint három akkor meghalunk és újra kell kezdenünk a pályát, ha tüskébe esünk az egyből halálhoz vezet. Ha meghaltunk újra meg kell ölnünk az ellenfeleket és összegyűjteni az üstöket. Az ellenségnek is három szívnyi életereje van ami azt jelenti hogy három lövés kell neki hogy kipurcanjon, viszont ha a fejére ugrunk akkor egyből meghal. A cél, halál nélkül kell teljesíteni a pályát, és úgy tudunk a következő pályára lépni, hogy ha megöljük az összes ellenfelet és összegyűjtjük az összes üstöt, ha ez megtörtént akkor megjelenik a pályán egy ajtó és azon keresztül sétálnunk.

2. Rendszerkövetelmények

A játékprogramnak egyáltalán nincs nagy hardver igénye, szinte bármin elfut, amin 64-bites Windows operációs rendszer van, fontos hogy minimum a DirectX10-es driver telepítve legyen. Az adatbázis szerver nem létszükséglet a játék futtatásához, nélküle is menni fog csak nem fogunk tudni bejelentkezni a fiókunkba, csak vendégként fogunk tudni játszani.

MINIMÁLIS RENDSZERKÖVETELMÉNYEK	Windows	Android
Operációs rendszer (OS)	Windows 7 (SP1+), Windows 10 and Windows 11, csak a 64-bites verziók	KitKat 4.4 (API 19)+
Processzor (CPU)	X64 architektúra SSE2 utasításkészlet támogatással	ARMv7, Neon támogatással (32-bit) vagy ARM64
Grafikus API	DirectX10 vagy DirectX11 vagy DirectX12	OpenGL ES 2.0+, OpenGL ES 3.0+, Vulkan
További követelmények	Hárdver gyártók által hivatalosan támogatott driverek	1GB+ RAM. A hardvernek natívan Android OS-t kell futtatnia. Android emulátor nem támogatott.

A játék a következő hardvereken tesztelve lett és tökéltesen fut:

- Intel Core i7-6700k, 16 GB RAM, NVIDIA GeForce GTX 1060 6GB, Windows 10 Pro (x64)
- Intel Core i7-3517, 8 GB RAM, Intel(R) HD Graphics 4000, Windows 8.1 Pro (x64)
- Intel Core i5-4460, 8 GB RAM, Windows 10 Enterprise (x64)
- Intel Core i5-10900, 8 GB RAM, Geforce GTX 1650 Ti Max-Q, Windows 10 Pro (x64)
- Intel Core i7-6700, 16 GB RAM, Geforce GTX 1070, Windows 10 Enterprise (x64)

3. A program telepítése

A szoftvert a Chug és Dug weboldaláról tudjuk letölteni a *Download* gomb segítségével. Ha megnyomtuk a gombot egyből megkezdődik a *game.zip* fájl letöltése. Ha letöltődött akkor csomagoljuk ki a fájlt. Ha ez is megtörtént akkor nyissuk meg a mappát és keressük a *Chug és dug.exe* alkalmazást. Ha meg van indítsuk el, fel fog ugrani egy szürke kis ablak, itt betudjuk állítani hogy milyen felbontáson, milyen minőségű grafikán és hogy teljesképernyőben induljon-e el a játék.

1. Nyissuk meg a weboldalt (http://gyros-tesok.hu/chugesdug.html)
2. Nyomjuk meg a *Download* gombot
3. Csomagoljuk ki a *game.zip*-et
4. A mappában keressük meg a *Chug és Dug.exe* alkalmazást.
5. Válasszuk ki a kívánt opciókat
6. Nyomjuk meg a *Play!* gombot
7. Élvezzük a játékot

4. A program használatának a részletes leírás

A program fejlesztésekor törekedtem egy áltátható, egységes, egyszerű kinézetet alkotni, hogy mindenki számára átlátható legyen.

Amikor elindítjuk a játékot ez a látvány fog fogadni minket.

1. Főmenü

1.1. Ezzel a gombbal tudjuk elindítani az egyjátékos módot.

1.2. A Multiplayer gombbal a többjátékos módot tudjuk elindítani.

1.3. A Settings gomb megnyomásával a beállításokhoz férünk hozzá.

1.4. Az Exit gomb megnyomásával bezárjuk a játékot.

2. Beállítások

2.1 Az első helyen egy legördülő lista található, ahol tudunk válogatni a monitorunk által támogatott felbontások között és ha ráklilekelünk az egyik elemre akkor természetesen be is állítja azt a felbontást

2.1. Felbontás

2.2 Itt szintúgy egy legördül listát találhatunk, aminek segítségével a játék kinézetének a minőségét tudjuk állítani.

2.3 A harmadik egy kétállású gomb, amivel a teljesképernyős módból, ablakos módba tudunk váltani és fordítva.

2.2 Grafika

2.4 Az utolsó gomb a *Back,* ezzel a főmenübe tudunk visszalépni.

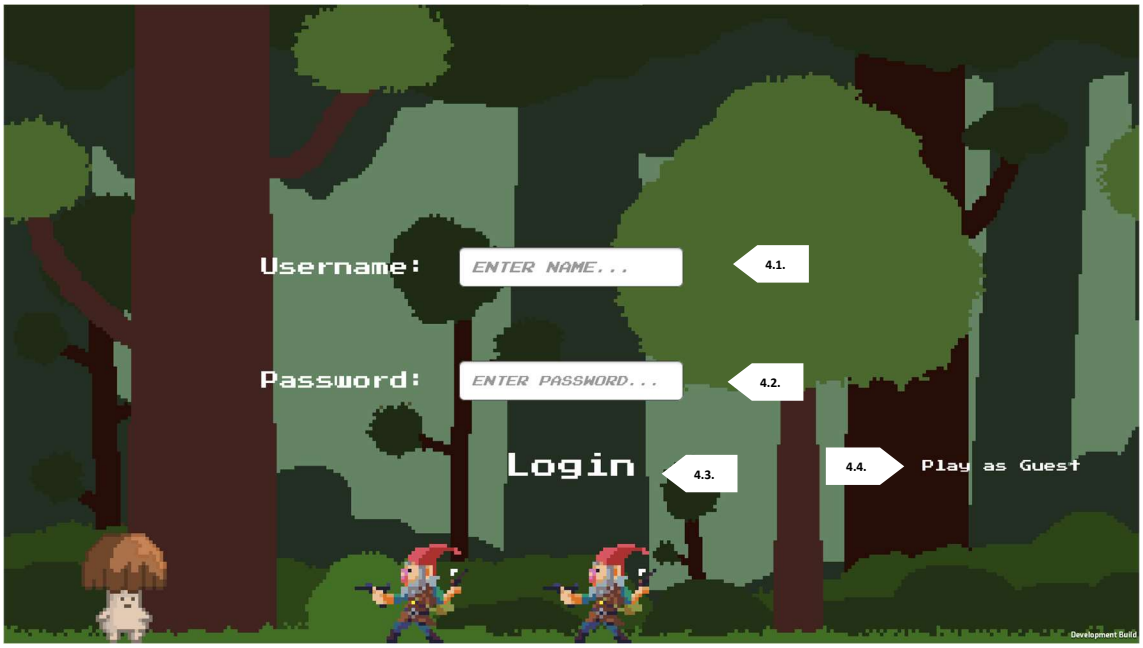
A *Singleplayer* vagy *Multiplayer* gomb megnyomása után erre a felületre kerülünk.

3. Név megadása

3.1. Az első elem egy beviteli mező, itt kell megadnunk a nevünket.

3.2. A *Start Game* feliratú gombbal tudunk tovább lépni a szobákhoz/szobához.

3.3. A *Login* gomb megnyomás után megjelenik a bejelentkezési felület.



4. Bejelentkezési felület

4.1. Ennél a beviteli mezőnél tudjuk megadni a felhasználónevünket.

4.2. Ennél a beviteli mezőnél pedig a jelszavunkat kell megadni.

4.3. A *Login* gomb megnyomásával megjelennek a szobák.

4.4. Ha ezt a gombot nyomjuk meg akkor visszamegyünk arra az oldalra, ahonnan ide kattintottunk.

Ez itt az egyjátékos mód szobája.

5. Singleplayer szoba

5.1. Itt arról bizonyosodhatunk meg hogy az egyjátékos módot indítottuk el.

5.2. Itt a saját nevünket láthatjuk.

5.3. Ezzel a nyíllal balra tudunk léptetni a karakterek között.

5.4. Ezzel a nyíllal pedig jobbra tudunk léptetni.

5.5. Itt a karakter nevét láthatjuk.

5.6. A *Back to main menu* gombbal vissza tudunk lépni a főmenübe.

5.7. A *Play Game* gombbal elindítjuk a játékot.

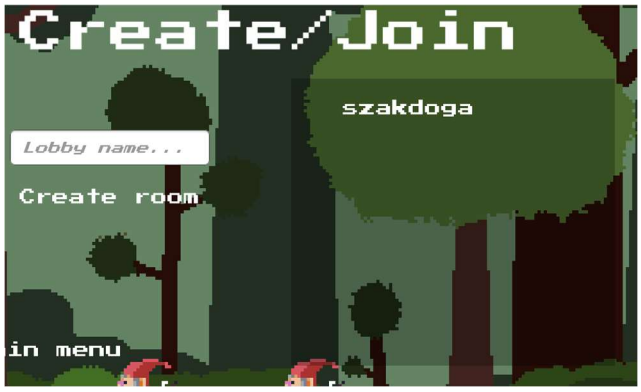
6. Create/Join

6.1. Itt tudjuk megadni a létrehozandó szoba nevét.

6.2. Ezzel a gombbal hozzuk létre a szobát.

6.3. A *Back to main menu* gombbal vissza tudunk lépni a főmenübe.

6.4. Itt találhatóak meg a szobák, ha vannak.



6.1. Szoba

7. Multiplayer lobby

7.1. A szoba neve.

7.2. A játékos neve.

7.3. Ezzel a nyíllal balra tudunk léptetni a karakterek között.

7.4. Ezzel a nyíllal pedig jobbra tudunk léptetni.

7.5. Ez itt a másik játékos.

7.6. Itt a karakter nevét láthatjuk.

7.7. A *Leave Room* gombbal tudjuk elhagyni a szobát.

7.8. A *Play Game* gombbal pedig a játékot tudjuk elindítani.

***8**. UI és Collectable*

8.1. Ez itt a karakterünk életereje.

8.2. Itt pedig azt láthatjuk, hogy mennyi gombát öltünk eddig.

8.3. Ez pedig az arannyal teli üst, amit össze kell gyűjtenünk.

9.1. Itt a játékos található

9.2. Ez itt az ellenfél

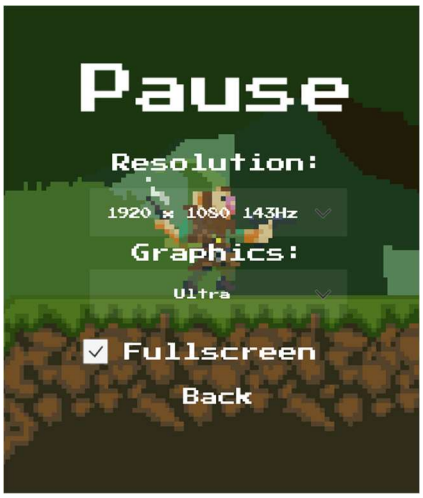
9. Ellenfél

10. Pause menü

10.1. Ezzel a gombbal visszalépsz a főmenübe.

10.2. A *Settings* gomb megjeleníti a beállítások panelt.

10.3. Ez a gomb pedig bezárja a játékot.



11. Beállítások

Fejlesztői dokumentáció

1. Témaválasztás indoklása

Azért választottam ezt a témát mivel szeretek játszani és szívesebben fektetek abba sok energiát, amivel másokat és magam is tudom szórakoztatni. Amikor meghallottam hogy projektmunkát kell csinálni, egyből tudtam, hogy valamilyen játékot fogok fejleszteni. Egy minimális tapasztalatom már volt benne mivel 12.-ben is kellet projektmunkát készíteni és akkor is játékot fejlesztettem. Ennek a témája egy Monopoly társasjáték volt, ez a vonal nem volt rossz, az is megfordult a fejemben, hogy ezt a projektet folytatom, de ezt hamar elvettem mert tiszta lappal szerettem volna indulni. Gondolkoztam hogy 2D-s vagy 3D-s játék legyen de maradtam a 2D-nél, mivel egyszerűbb megrajzolni a karaktereket és a pályát mint 3D-ben lemodellezni azokat. Így szinte magától értetődő volt, hogy platformer játékot fogok készíteni. Aztán jött a dilemma, hogy mi legyen a témája, hogy nézzen ki a játékmenet, meg hasonló kérdések vetődtek fel. A témát, meg a karaktert viszonylag hamar kitaláltam, a barátnőm segített ötletelni és együtt kitaláltuk hogy legyenek manók gombák ellen, és végül az lett

2. Az alkalmazott fejlesztői eszközök

2.1. Unity

„A Unity egy videójáték-motor, amelyet a Unity Technologies fejleszt. A Unity segítségével háromdimenziós illetve kétdimenziós videójátékokat, ezen kívül egyéb interaktív jellegű tartalmakat lehet létrehozni, például építészeti látványterveket vagy valós idejű háromdimenziós animációkat.Többek között előnye, hogy a szoftver képes nagyméretű adatbázisokat kezelni, kihasználni a kölcsönhatások és animációk képességeit, előre kiszámított vagy valós idejű világítást biztosítani. Továbbá használható geometriai eszközcsomagok továbbítására, illetve viselkedési elemek hozzáadására egyes objektumokhoz. Ezek mellett a játékmotor folyamatosan megőrzi a végleges változat megjelenítését.” (1)

2.2. C#

A Visual Studio egy több programozásinyelvet tartalmazó fejlesztői környezet. Az általam használt C# mellett kódolhatunk F#-ban, C++-ban, Visual Basicben vagy XML-ben. Az iskolai tanulmányaimat kiválóan tudtam alkalmazni. Ugyan nem konzolos vagy WinForms applikációban dolgoztam, de a fejlesztői környezetben otthonosan tevékenykedtem.

2.3. Aseprite

Az Aseprite egy képszerkesztő program, ami kifejezetten pixel art rajzokhoz, animációkhoz tervezett program. Mind a három népszerű operációs rendszeren is futtatható, Windows, macOS és Linux. A programra egy youtube videóban bukkantam rá és nagyon szimpatikus volt első ránézésre és ez a későbbiekben sem változott, nagyon kis kezes program átlátható és egyszerű használni. Ezen program segítségével rajzoltam meg a játékban található összes elemet.

2.4. Notepad++

„A Notepad++ egy szöveg- és forráskódszerkesztő Microsoft Windows platformra. Akár Linux vagy Mac OS X alatt is futtatható, a Wine segítségével. A Notepad++ egyik fő előnye a Windows beépített szerkesztőjével szemben a fülekkel ellátott felület, amely több fájl párhuzamos szerkesztését engedi meg, támogatja a UNIX/LINUX sorvégeket.” (Wikipédia). Azért nem Visual Studio

Code-ot használok mert a Notepad++ szimpatikusabb és számomra átláthatóbb, habár a Code-ot is sokszor használom.

2.5. HTML

„A HTML (angolul: HyperText Markup Language=hiperszöveges jelölőnyelv) egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált a W3C (World Wide Web Consortium) támogatásával. Az aktuális változata az 5, mely az SGML általános jelölőnyelv egy konkrét alkalmazása (azaz minden 5-ös HTML dokumentum egyben az SGML dokumentumszabványnak is meg kell hogy feleljen). Ezt tervek szerint lassan kiszorította volna az XHTML, amely a szintén SGML alapú XML leíró nyelven alapul.” (2).

2.6. PHP

„A PHP egy általános *szerveroldali szkriptnyelv* dinamikus weblapok készítésére. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a *webszerver* PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre. Rasmus Lerdorf *1995*-ben indította útjára. Ma a The PHP Group tartja fenn és fejleszti. A PHP szabad szoftver, de licence nem csereszabatos a GNU licenccel, mivel megkötéseket tartalmaz a PHP név használatára.” (3). A PHP használatát a technikumi tanulmányaink alatt kezdtük elsajátítani.

2.7. Javascript

„A JavaScript programozási nyelv egy objektumorientált, prototípus-alapú szkriptnyelv, amelyet weboldalakon elterjedten használnak.” (4).

2.8. CSS

„A CSS (Cascading Style Sheets, magyarul: lépcsőzetes stíluslapok) a számítástechnikában egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. Ezenkívül használható bármilyen XML alapú dokumentum stílusának leírására is, mint például az SVG, XUL stb.” (5).

2.9. XAMPP

„A XAMPP – (kiejtése: /ˈzæmp/ vagy /ˈɛks.æmp/[1]) – egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB (korábban a MySQL[2]) adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői (végrehajtó rendszerei). Ez a szoftvercsomag egy integrált rendszert alkot, amely webes alkalmazások készítését, tesztelését és futtatását célozza, és ehhez egy csomagban minden szükséges összetevőt tartalmaz. A rendszer egyik nagy előnye az összehangolt elemek könnyű telepíthetősége.” (6).

2.10. Microsoft Word

„A Microsoft Word (a 2003-as, Windowsra készült változat óta Microsoft Office Word) a Microsoft által készített dokumentumszerkesztő (szövegfeldolgozó) program. Első változatát Richard Brodie írta 1983-ban DOS-t futtató IBM PC-kre. Azóta Apple Macintosh-ra (1984), SCO UNIX-ra és Microsoft Windowsra (1989) is megjelent. Ma a Word része a Microsoft Office irodai alkalmazáscsomagnak (és egyúttal az Office Rendszernek is). A Word 6 for Windowstól kezdődően a Windowsra készült Word-verziók magyar nyelven is megjelentek. A Word angol szó, jelentése „szó”.” (7).

2.10. Microsoft PowerPoint

„A Microsoft PowerPoint egy számítógépes prezentáció készítő program, amelyet Robert Gaskin és Dennis Austinkészítettek 1987-ben, a Forethought Inc. nevű cégnél. 1987 áprilisában jelent meg eleinte csak Mac számítógépekre. Három hónappal később a Microsoft megvásárolta a PowerPointot, 14 millió dollárért. Ezt követően a Microsoft új központot hozott létre a Szilícium-völgyben.” (8).

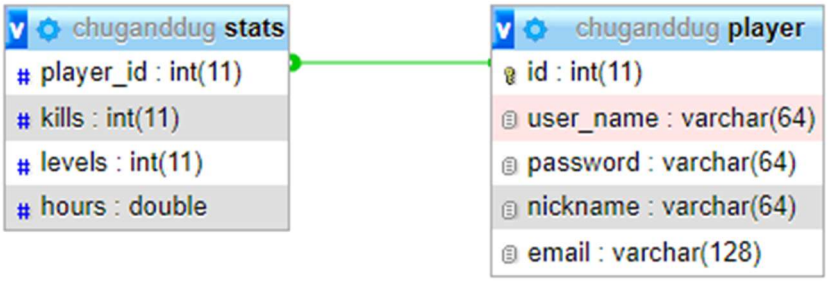
2.12. Adobe Photoshop

„Az Adobe Photoshop egy képszerkesztő és fényképfeldolgozó szoftver, melynek a fejlesztője az Adobe Systems. Első verziója, a 0.63-as 1988 októberében jelent meg Macintoshra.” (9).

2.12. Adobe PremierPro

„Az Adobe Premiere Pro mindenekelőtt videószerkesztő program. Ez egy nemlineáris szerkesztési rendszer. Alkalmas mind amatőr szakemberek, mind rajongók számára. Megvásárolható és használható kizárólag, vagy különféle alkalmazásokkal, például Adobe Photoshop, After Effects stb. Mellett. Ez az Adobe Creative Cloud szoftvergyűjtemény, valamint az Adobe CS6 vagy az Adobe Creative Suite része. Videók, beleértve a filmeket, szerkesztésére használják, és a világ minden tájáról gyorsan mozgóprogrammá válik.” (10).

3. Adatmodell leírása



12. Egyed kapcsolat

id	int(11)	Elsődleges kulcs Automatikusan generált szám, 0-tól indul, ez alapján azonosítom a játékost, mindenkinek egyedi.
user_name	varchar(64)	Ez a játékos felhasználó neve, ezzel tud bejelentkezni. Regisztrációnál adja meg a játékos. Szöveg típusú adat, ami maximum 64 karakter hosszúságú lehet.
password	varchar(64)	Ez a játékos jelszava, regisztrációnál adod meg, ez is szüksége a bejelentkezésnél. SHA-1 hash függvénnyel van titkosítva. Szöveg típusú adat, ami maximum 64 karakter hosszúságú lehet.
nickname	varchar(64)	Ez a játékos beceneve, ezt látja a többi játékos. Szöveg típusú adat, ami maximum 64 karakter hosszúságú lehet.
email	varchar(128)	Ez a játékos email címe, regisztációkor kell megadni, szöveg típusú adat, ami maximum 128 karakter hosszúságú lehet.

player_id	int(11)	Idegen kulcs A player táblában lévő player.id-ra mutat. Egész típusú szám lehet és maximum 11 hosszúságú.
kills	int(11)	Ez tárolja a játékos öléseinek számát. Egész típusú szám lehet és maximum 11 hosszúságú.
levels	int(11)	Ez a tárolja el hogy hanyadik pályáig jutott a játékos. Egész típusú szám lehet és maximum 11 hosszúságú.
hours	double	Ez számolja hogy hány órát játszott eddig a játékos. Csak valós szám lehet.

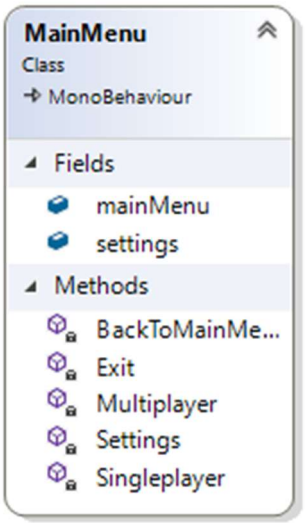
4. Részletes feladatspecifikáció, algoritmusok

4.1. C#

4.1.1. Menu

4.1.1.1. MainMenu.cs

A MainMenu nevű forráskód azért lett létrehozva, hogy a főmenüt kezelje. Megtalálható benne kettő darab *public GameObject* melyeknek neveik *mainMenu* és *settings*, ezeket azért hoztam létre hogy tudjak hivatkozni az editorban lévő MainMenu és Settings GameObjectekre amik legfőképpen gombokat tartalmaznak, ezenkívül még öt darab eljárás szerepel a fájlban. Az itt lévő eljárásokat gombokhoz rendeltem hozzá.



13. MainMenu class

Az első eljárás amit írtam a *private void Singleplayer(),* ebben két darab eljárást hívok meg és egy darab függvényt, az első eljárás a *PhotonNetwork.Disconnect();* amit a *using Photon.Pun* névtér segítségével hívok meg. Ez az eljárás annyit tesz, hogy a klienst lekapcsolja a photon kiszolgálóról. A következő a *PhotonNetwork.OfflineMode* függvény, ennek a visszatérése egy bool változó amit itt igazra állítok, ez azért fontos mert így lett használni egyjátékos módban azokat a kódokat amik a többjátékos módhoz lett írva. A második eljárás az paraméterrel is rendelkezik *SceneManager.LoadScene("Loading");* , itt annak a színtérnek a nevét adjuk át amit szeretnénk betölteni, ezt is a *using Photon.Pun* névtér segítségével hívom meg.

A második a *private void Multiplayer()* ebben az előzökben említett *PhotonNetwork.OfflineMode* függvényt ismét meghívom és itt most hamisra állítom és szintén meghívom a *SceneManager.LoadScene("Loading");* .

A harmadik eljárás a *private void Settings()* itt kap szerepet a két GameObject amit az elején deklaráltam. A GameObjecteknek van egy *SetActive()* nevű eljárása ami egy bool típusú paramétert vár el és ez alapján állítja láthatóvá magát.

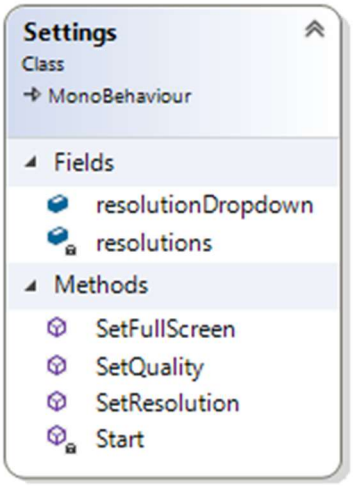
A *mainMenu* láthatóságát hamisra állítom *mainMenu.SetActive(false);* a *settings* láthatóságát pedig igazra settings.SetActive(true);.

A negyedik a *private void BackToMainMenu()* szinte ugyan az mint az előző eljárás csak most a *setting* láthatóságát hamisra állítom settings.SetActive(false); , a *mainMenu* láthatóságát pedig igazra *mainMenu.SetActive(true);* .

Az ötödik eljárásban csak egy eljárás van ami bezárja az alkalmazást, ez a *Application.Quit();* az Application osztály egyik eljárása.

4.1.1.2. Settings.cs

A Settings.cs azért lett létrehozva, hogy a személyes igényünkre és az eszközünk igényeihez tudjuk igazítani a felbontást, a játék kinézetének minőségét és azt, hogy teljesképernyőben fusson a játék vagy sem. A kód elején deklaráltam egy publikus *TMP_Dropdown resolutionDropdown amit* a *using TMPro* névtér segítségével érek el. A *TextMeshPro* jobb szerkeszthetőséget nyújt, mint az alap *Unity Text*. A következő sorban pedig egy *Resolution* típusú tömböt deklaráltam *Resolution[] resolutions.*



14. Settings class

A *void Start()* funkcióban az első utasítás a *resolutions=Screen.resolutions;* itt feltölti *resolutions* tömböt az az éppen használt monitor által támogatott összes teljesképernyős felbontással. A következő sorban kiürítem a *resolutionDropdown* elemeit *resolutionDropdown.ClearOptions();*. Ez után létrehozok egy *string* típusú listát *options* néven, erre azért van szükség mert ennek a listának az elemeivel fogjuk feltölteni a *resolutionDropdown*-t. Utána inicializáltam egy *int* típusú változót, neve *currentResolutionIndex* értékét 0-ra állítottam. A következő sorokban az *options* listát töltöm fel egy *for* ciklus segítségével. A ciklust 0-tól a *resolutions* tömb utolsó előtti eleméig egyesével léptem. A ciklusmagban pedig inicializáltam egy *string* típusú változót *option* néven ennek az értékét így állítottam be: *resolutions[i].width+" x "+resolutions[i].height+" "+resolutions.refreshRate+"Hz";* ez annyit takar hogy veszi a *resolutions*

tömbnek az i-edik elemét és kiírja a képernyő felbontását és képfrissítését ami így néz ki *1920 x 1080 60Hz.* Ezt követően a ciklus hozzáadja az *options* listához az *option* nevű *stringet.* A ciklus törzsben még van egy ha függvény aminek az a szerepe, hogy kiválasztja annak a felbontásnak az indexét ami megegyezik a képernyőnk felbontásával és ezt eltárolja a *currentResolutionIndex* változóban, ezután vége a ciklusnak. A következő három utasítás a következőket csinálja: feltölti a *resolutionDropdown* nevű legördülő listást az *options* nevű listával *resolutionDropdown.AddOptions(options);* , ezután beállítja a *resolutionDropdown* értékét a *currentResolutionIndex* értékre *resolutionDropdown.value=currentResolutionIndex;,* ez azért jó mert a felhasználó az éppen aktuális felbontást látja először, a következő pedig azért felel ha felhasználó kiválaszt egy másik felbontást akkor azt állítsa be aktuálisnak *resolutionDropdown.RefreshShownValue()*.

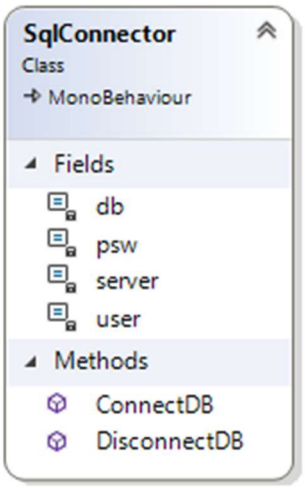
A *public void SetResolution(int resolutionIndex)* egy *int* típusú paramétert vár el ami alapján beállítja a felbontás a *resolutions* tömb *resolutionIndex*-edik elemére. Ezt hozzá rendeltem a színtérben található *Resolution(DropDown)* nevű *TMP_DropDown*-hoz.

A *public void SetQuality(int qualityIndex)* szintúgy egy *int* típusú paramétert vár el ami alapján beállítja a kívánt grafika minőséget. Ezt a *Graphics(DropDown)* nevű *TMP_DropDown*-hoz rendeltem a mi szintén a színtérben található.

A *public void SetFullScreen(bool isFullScreen)* ez egy *bool* típusú változót kér amit a színtérben elhelyezett FullScreen(Toggle) kéttállású kapcsológomb ad meg neki.

4.1.1.3. SqlConnector.cs

Ennél a forráskódnál az volt az elsőlépésem hogy telepítettem a MySql.Data NuGet-et, enélkül nem tudtam volna létrehozni a kapcsolatot a program és az adatbázis szerver között. Miután befejeződött a telepítés és a programkód tetejében elhelyeztem *using MySql.Data.MySqlClient;* névteret, ez által elérhető válik az osztály minden típusának a minősítés nélküli használat. Ezután megvalósítottam egy egyszerű kapcsolatot a program és az adatbázis szerver között azok alapján, amit a suliban tanultunk.



15. SqlConnector class

Az első négy sorba felvettem négy konstans *string* típusú változót az első a *server,* ennek az értékét a localhost IP-címére állítottam, a másik háromnak (*user*, *psw*, *db*) üres értéket adtam, mivel most ennél a projektnél nincs szerepe.

Az első eljárás *public static MySqlConnection ConnectDB()* fontos hogy *static* legyen mert el kell érjük bárhonnan. Itt csatlakoztatjuk a programot az adatbázis szerverhez.

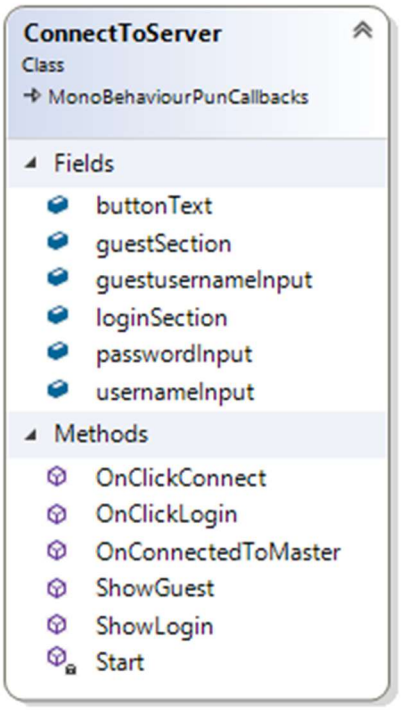
A második eljárás *public static void DisconnectDB(ref MySqlConnection Con)* itt paraméterként egy referencia *ref MySqlConnection Con* kell átadnunk, ennek az a feladata hogy lekapcsolja a felhasználót a szerverről.

4.1.1.4. ConnectToServer.cs

A ConnectToServer.cs legfőbb feladata hogy csatlakoztassa a játékost a photon kiszolgálóra, emellett ez kezeli a fiókunkban való bejelentkezést az adatbázis szerveren keresztül.

Először felvettem három *public TMP_InputField*-et *usernameInput, passwordInput, guestusernameInput* , egy *public TMP_Text buttonText*-t és kettő *public GameObject guestSection, loginSection*.

A *void Start()* funkcióban csak egy feltételes utasítás van, a felétele pedig egy *bool* változó ami a *PhotonNetwork.OfflineMode* ha igazként tér vissza akkor a *buttonText* szövegét erre állítja át: *"Start Game"* viszont ha hamisként akkor erre *"Connect"*.



16. ConnectToServer class

A következő eljárás a *public void ShowGuest()* ez igazából csak annyit csinál hogy a loginSection-t láthatóságát hamisra állítja, a *guestSection*-ét pedig igazra.

A *public void ShowLogin()* szinte ugyan azt csinálja mint a *ShowGuest()* csak itt a *guestSection* láthatóságát hamisra állítja, a *loginSection*-ét pedig igazra.

A *public void OnClickConnect()* ezt a *Connect(Button)* gombhoz rendeltem. Ebben az eljárásban egy ha függvény található aminek feltétele hogy a *guestusernameInput* szövege egy betűnél hosszabbnak lennie, ha ez teljesül akkor a *PhotonNetwork.NickName* egyenlő lesz a *guestusernameInput.text*-el. Ezután van még egy ha függvény, aminek feltétele negálva a *PhotonNetwork.OfflineMode* ha ez teljesül akkor három utasítás hajtódik végre. Az első a *PhotonNetwork.AutomaticallySyncScene* amit igazra állítok ez annyit jelent, hogy mindenkinek ugyan azt a színteret tölti be. A második a *PhotonNetwork.ConnectUsingSettings();* ami annyit takar, hogy úgy csatlakozik a Photon szerverre, mint ahogy beállítottuk a PhotonServerSettings fájlban. A harmadik pedig beállítja a *buttonText* szövegét

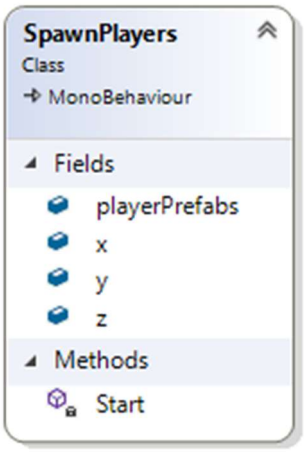
erre: ”Connecting…”. Azonban ha a *PhotonNetwork.OfflineMode* igaz akkor betölti a ”SingleplayerLobby” nevű színteret.

A következő eljárás a *public void OnClickLogin()* ezzel tudsz bejelentkezni az adatbázisba, ezt a Login(Button) gombhoz rendeltem. Az *usernameInput*-ról beolvassa a felhasználónevet és beleteszi *string* tipusú *username*-be, a *passwordInput*-ról pedig a jelszót és beleteszi a szintúgy *string* típusú *psw* változóba. Aztán meghívom a *Login* nevű tárolt eljárást és felparaméterezem a *username*-el és a *psw*-vel. Ezután ha *reader.Read()* igaz értékkel tér vissza akkor beállítja a játékos becenevét a lekérdezett becenévre és szintúgy beállítja a *PhotonNetwork.AutomaticallySyncScene*-et igazra és meg hívja a *PhotonNetwork.ConnectUsingSettings()* mint a OnClickConnect() eljárásban. A ha függvény után lecsatlakozik az adatbázisról.

A *public override void OnConnectedToMaster()* eljárás annyit tesz hogy a játékos csatlakozott a Master szerverhez és készen áll akkor betölti a *”Lobby”* nevű színteret.

4.1.2. Game

4.1.2.4. SpawnPlayers.cs



17. SpawnPlayers

A *SpawnPlayers.cs* idézi meg a játékosokat. Először egy *public GameObject[]* típusú tömböt vettem fel *playerPrefabs* néven. Ezután három *public float* típusú *x,y,z* nevű változót definiáltam, ezek lesznek a koordináták ahol megidéződnek a játékosok.

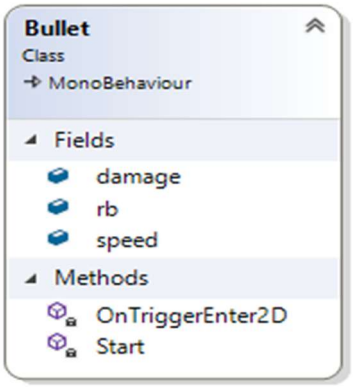
A *void Start()* funkcióban létrehozok egy *Vector3* típusú *position* nevű pontot aminek átadom az *x,y,z* változókat. Ezt követően definiálok egy *GameObject*-et *playerToSpawn* néven. A következő utasítás egy ha elágazás ami azt vizsgálja hogy ha nem választott karaktert akkor a *playerToSpawn* értékét a *playerPrefabs* nulladik elmére állítja, viszont ha választott akkor a *playerToSpawn* értékét a *playerPrefabs* választott karakter index értékére állítja. Végül megidézi a kiválasztott karaktert, a beállított ponton, a

PhotonNetwork.Instanite(playerToSpawn.name, position, Quaternion.identity); paraméterezett függvény segítségével.

4.1.3. Player

4.1.3.2. Bullet.cs

A *Bullet.cs* a játékos fegyveréből kijövő golyót kezeli. A forráskód elején deklarálok egy publikus *float* típusú *speed* nevű változót és értékét *20f*-re állítom, itt tudjuk beállítani a golyó sebességét, ezután felveszek egy szintúgy publikus *Rigidbody2D* típusú *rb* nevű változót. Az utolsó *a public int* típusú *damage* nevű változó aminek értékét 40-re állítottam, ez azt jelenti hogy ha a golyó eltalálja az ellenfelet akkor az életereje 40-el csökken.



18. Bullet class

A *void Start()* funkcióban bírjuk mozgásra a golyót.

A *void OnTriggerEnter2D(Collider2D hitInfo)* ez a paraméterezett eljárás azt vizsgálja, hogy mivel érintkezik a golyó. Első sorában hivatkozok az *enemy.cs*-re, a következő utasítás egy feltételes utasítás, ami azt vizsgálja hogy ha az *enemy* nem egyenlő *null*-al akkor az ellen fél sebződik. A következő utasítás szintúgy egy feltételes utasítás ami pedig azt vizsgálja hogy milyen Tag-gel rendelkezik az a *GameObject* amivel érintkezik, ha a Tag nem a *"Turn", "Player", "Ladder"* vagy a *"Collectable"* akkor a golyó megsemmisül.

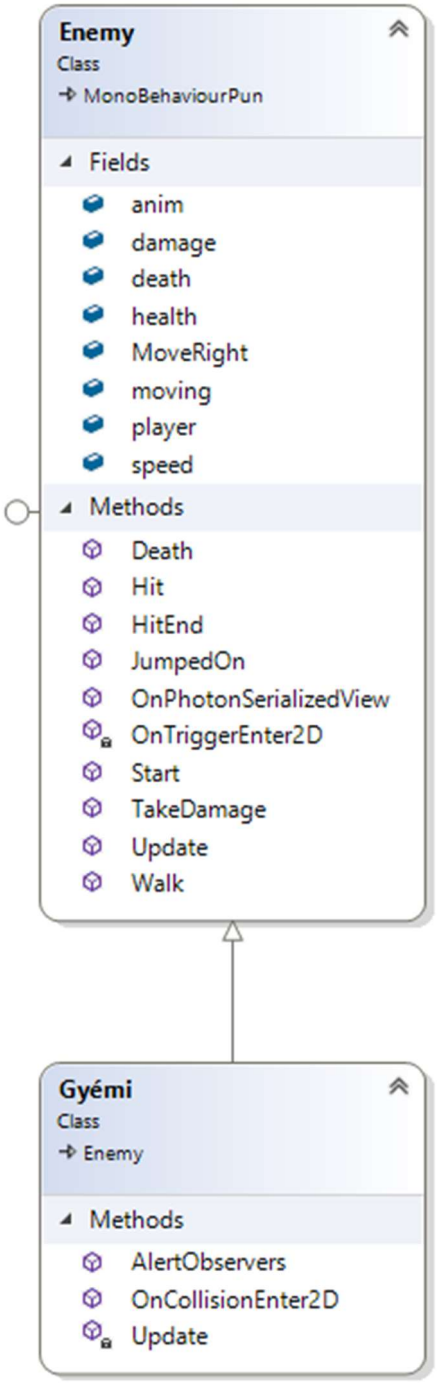
4.1.4. Enemys

4.1.4.1. Enemy.cs

Az E*nemy.cs* maga az ellenfél, ez a forráskód kezeli az összes vele kapcsolódó eseményt.

Először deklaráltam három *int* típusú változót az első a *health* ennek értékét 100-ra állítottam, ez az ellenfél életereje, a második a *damage* ennek értékét 40-re állítottam, ennyit fog veszíteni a játékos az életerejéből akihez hozzáér az ellenfél, a harmadik pedig a *death* ez számolja az ellenfél halálainak számát. Aztán

felveszek egy *Animator* típusú *anim* nevű változót ez azért kell, hogy ha majd példányosítom akkor hozzá férjek az Animator metódusaihoz, szintúgy a *Player-Controller* típusú *player* nevű változónál. Ezután definiálok egy *float* típusú *speed* nevű változót ennek az értéket a színtérben adom meg. Végül két *bool* típusú változót deklarálok az egyiket *moveRight* néven és *false*-ra állítom ezzel fogja majd a program megvizsgálni hogy merre megy és az alapján változtatja a mozgás irányát, a másikat pedig *moving* néven és *true*-ra állítom, ennek szerepe hogy tudjuk hogy mikor mozog az ellenfél.



19. ábra

A *void Start()* funkcióban példányosítom az *anim* változót.

A *void Update()* funkcióban pedig a *player* változót példányosítom, még pedig úgy hogy a színtérben megkeresi a *Player.cs*-vel rendelkező *GameObject*-et, plusz meghívom a *Walk()* eljárást, ha ezt nem tenném akkor nem mozogna az ellenfél.

A *public void Walk()* eljárással bírom mozgásra a karaktert, három darab ha függvény található egymásban az első ha függvénybe akkor lépünk bele, hogy ha az életereje az ellenfélnek nagyobb nullánál. A másodikba pedig akkor, hogy ha a *moving* változó igazra van állítva, a harmadiknak pedig az a feltétele, hogy ha a *moveRight* változó érteke igaz akkor az ellenfelet függölegesen jobbra tükrözi és jobbra kezd el haladni, az ”Walk” animációt aktiválja és a moving-ot true-ra állítja. Különben ha a moveRight értéke hamis

akkor balra tükrözi, balra kezd le haladni és szintúgy aktiválja a ”Walk” animációt és a moving igaz lesz.

A következő egy paraméterezett eljárás public void TakeDamage(int damage) paramétere int típusú. Feladata hogy életerőt veszítsen az ellenfél, ezt az eljárást a bullet.cs-ben hívom meg. A health változóból kivonja a damage változót, ezután egy ha függvény következik ami azt vizsgálja hogy az ellenfél életereje kisebb vagy egyenlő mint nulla akkor a ”Death” animáció lejátszódik.

A következező a public void Hit() ez a sebződés animációjáért felelős. A moving bool típusú változót hamisra állítja ez által megáll a karakter, ezután pedig a ”Walk” animációt megállítja és elindítja a ”Hit” animációt.

A public void Hitend() akkor hívódik meg ha a ”Hit” animáció befejeződött az ellenkezőjét csinálja a Hit()-nek.

A public void JumpedOn() azt az esetett kezeli amikor a játékos az ellenfél fejére ugrik, ezután aktiválódik a ”Death” animáció aminek a végén meghívódik a Death() eljárás amit a következő bekezdésben részletezek.

A public void Death() a death változó értékét 1-re állítja, meghívja a player.KillCounter(death) paraméterezett eljárást és át adja a death változót, ezután törli a játékmenetből az ellenfelet.

A void OnTriggerEnter2D(Collider2D trig) ez egy paraméterezet eljárás ami azt vizsgálja hogy mivel ütközik a karakter. Ha a karakter a ”Turn”-el taggelt GameObject-hez ér, akkor megfordítja járásának az irányát.

4.1.4.2. Gyemi.cs

Gyémi nem egy sima ellenfél, de ennek ellenére is megörököl minden fontos alap tulajdonságot az *Enemy* osztálytól.

A *private new void Update()* ebben a funkcióban ismét a *player* változót példányosítom, még pedig úgy hogy a színtérben megkeresi a *Player.cs*-vel rendelkező *GameObject*-et, plusz meghívom a *Walk()* eljárást. Aztán feltételes utasítással megvizsgálóm azt hogy ha a játékos 5x-nyi távolságra van ha igen akkor a *moving* változót hamisra állítom így a mozgás megáll és lefut a *”Hide”* animáció.

A következő eljárás *public void AlertObservers(string message)* egy *string* típusú *message* névű paramétert vár el. Ezt a függvényt az animációs sávban elhelyezett eseményen hívom meg és itt meg adom a kívánt szöveget amire majd a scriptben hivatkozom. Összeségében ez annyit tesz hogy ha az animációs folyamat elér az elhelyezett eseményig akkor lefut az utasítás amit megadunk.

A *public void OnCollisionEnter2D(Collision2D other)* egy paraméterezett eljárás. Ebben az esetben azt vizsgálja hogy, ha hozzáér egy másik *GameObject* és *Tag*-je egyenlő a *"Player"*-el, akkor lejátssza a *"Show"* animációt.

5. Tesztelési dokumentáció

A játékot számtalan alakalommal leteszteltem fejlesztés közben. Sok esetben fedeztem fel bugokat ezeket próbáltam javítani több kevesebb sikerrel. Volt amikor ketten teszteltük többjátékos módban a játékot, akkor általában elég sok hiba előjött például, Nándóval játszottunk és az ellenfelek nem voltak szinkronizálva köztünk, ez eléggé zavaró mert így volt hogy meghalt az egyikünknél a másik, viszont a másik nézetéből pedig nem történt semmi, az általános hogy a másik játékos laggol a te szemszögedből.Volt olyan is hogy ha sebződtél akkor az életedet szimbolizáló szívek a bal felső sarokban rossz sorrendben tűnnek el. Akkor olyan is volt hogy ha karakter választásnál nem változtattad meg a karaktert akkor nem ídéződött meg a karakter amikor elindítottad a játékot. Sokáig bármennyit lehetett ugrani a levegőben, hogy ha a szorosan a fal mellett álltunk, ezt úgy javítottam, hogy létrehoztam még egy *collider*-t ami csak a lábánál van mert eddig a *groundcheck*-nél a játékos egész teste számított most csak a talprésze és így már nem lehet bármennyit ugrani a fal mellett. Volt amikor szóltam Máténak és Barnának hogy töltsék le a játékot és teszteljék le és én csak figyeltem hogy mit csinálnak, általában mindig találtak valamit.

6. Továbbfejlesztési lehetőségek, Összegzés

Sajnos itt a próbavédés közeledtével eléggé küzdöttem az idővel, így volt pár dolog, amit nem sikerült megvalósítani. Terveztem több pályát és ellenfelet rajzolni, de ezek rengeteg időbe telnek, van pár ellenfél ötlet már tervben, amint lesz időm meg valósítom őket. A másik fontos dolog amit terveztem, hogy lehessen pályát választani amikor a lobby-ban vagy, még pedig úgy hogy ha a játékos be van jelentkezve, akkor lekérdezi az adatbázisból hogy hányadik pályáig jutott és az alapján tud válogatni a köztük. Ezen kívül amit nem teljesen sikerült megvalósítani az a többjátékos mód, működik, lehet együtt játszani viszont, nincs minden tökéletesen szinkronizálva, például van olyan hogy a játékos1 megöl egy ellenfelet de ez a játékos2-nél nem hal meg és így ha a játékos1 bele sétál az ellenfélbe ami a játékos2-nél még ottvan akkor a játékos2 szemszögében meghal a játékos1, míg a játékos1 szemszögéből folytatódik tovább a játék, ez egy súlyos hiba, mihamarabb tervezem kijavítani mivel így nem élvezhető a többjátékos mód. Volt egy ellenfél amit már megrajzoltam bele is tettem a játékba, de nem teljesen azt csinálta amit szertettem volna, úgyhogy ez jelenleg nem található meg a játékban. Ez az ellenfél azt csinálta volna, hogy ha közel mész hozzá akkor ilyen mortár szerűen rád lő egy spóra golyót, itt azzal gyűlt meg a bajom, hogy nem tudtam kiszámítani, hogy hol lesz a játékos pár másodperc múlva. Van még egy rejtés hiba amire nem sikerült a jelen pillanatig rájönnöm hogy mi okozhatja, ez pedig az hogy ha többjátékos módban játszol és ahogyan telik az idő szépen lassan villódzva eltűnik a karakter, nem mindig csinálja.

Összeségében nem bántam meg hogy ezt a témát választottam, sőt kifejezetten élveztem dolgozni rajta, nem sajnáltam beleölni az időt. Az biztos hogy ez nem az utolsó játék amit tervezni fogok, későbbiekben szeretnék majd valamilyen három dimenziós játékot is készíteni.

7. Irodalomjegyzék, forrásmegjelölés

1/ Unity - https://hu.wikipedia.org/wiki/Unity_(j%C3%A1t%C3%A9kmotor) – Wikipédia

2/ HTML - https://hu.wikipedia.org/wiki/HTML - Wikipédia

3/ PHP - https://hu.wikipedia.org/wiki/PHP - Wikipédia

4/ JavaScript - https://hu.wikipedia.org/wiki/JavaScript - Wikipédia

5/ Cascadin Style Sheets - https://hu.wikipedia.org/wiki/Cascading_Style_Sheets - Wikipédia

6/ XAMPP - https://hu.wikipedia.org/wiki/XAMPP - Wikipédia

7/ Microsoft Word - https://hu.wikipedia.org/wiki/Microsoft_Word - Wikipédia

8/ Microsoft PowerPoint - https://hu.wikipedia.org/wiki/Microsoft_PowerPoint - Wikipédia

9/ Adobe Photoshop - https://hu.wikipedia.org/wiki/Adobe_Photoshop - Wikipédia

10/ Mi az Adobe Premiere Pro - https://hu.education-wiki.com/5419181-what-is-adobe-premiere-pro - Educatio-wiki