ZÁRÓDOLGOZAT

Tápai Árpád

2019-04-01

**MagIx**

**Tápai Árpád**

**SZÁMALK-Szalézi Szakgimnázium**

**Szoftverfejlesztő**

**Konzulens: Nagymáté Péter**

**Nyilatkozat**

a záródolgozat eredetiségéről

Alulírott Tápai Árpád (név) {Nagy Éva (anyja neve) 097219TA (szem. ig. szám)} büntetőjogi és fegyelmi felelősségem tudatában kijelentem és aláírásommal igazolom, hogy a záródolgozat saját munkám eredménye. A felhasznált irodalmi és egyéb információs forrásokat az előírásoknak megfelelően kezeltem, a záródolgozat készítésre vonatkozó szabályokat betartottam.

Kijelentem, hogy ahol mások eredményeit, szavait vagy gondolatait idéztem, azt a záródolgozatomban minden esetben, beazonosítható módon feltüntettem, a dolgozatban található fotók és ábrák közlésével pedig mások szerzői jogait nem sértem.

Kijelentem, hogy a záródolgozatom elektronikus változata teljes egészében megegyezik a nyomtatott formával.

Hozzájárulok ahhoz, hogy az érvényben lévő jogszabályok és a Számalk-Szalézi Szakgimnázium belső szabályzata alapján az iskola saját könyvtárában megtekinthető (olvasható) legyen a záródolgozatom.

A záródolgozat titkos/nem titkos.

Budapest, 2019. április …..

Tanuló aláírása

Tartalomjegyzék

[1 Bevezetés 4](#_Toc4622346)

[1.1 Miért pont RPG? 4](#_Toc4622347)

[1.2 Futtatási környezet 4](#_Toc4622348)

[2 Fejezet 5](#_Toc4622349)

[2.1 Rendszer követelmények 5](#_Toc4622350)

[2.2 Felhasználói útmutató. 5](#_Toc4622351)

[2.3 Felhasználói felület- HUD 7](#_Toc4622352)

[2.4 Felhasználói felület- JátékMenu 8](#_Toc4622353)

[2.5 Elindítás 8](#_Toc4622354)

[2.6 Felhasználói felület- Játék célja 9](#_Toc4622355)

[3 Fejlesztői dokumentáció 9](#_Toc4622356)

[3.1 Megvalósítás 9](#_Toc4622357)

[3.2 Magáról az engineről 10](#_Toc4622358)

[3.3 Kliens által alkalmazott osztályok és metódusok 11](#_Toc4622359)

[3.3.1 MonoBehaviour 12](#_Toc4622360)

[3.4 GameObejectek és prefábok 12](#_Toc4622361)

[3.4.1 GameObejectek 12](#_Toc4622362)

[3.4.2 Prefábok 13](#_Toc4622363)

[3.4.3 Scene 14](#_Toc4622364)

[3.4.4 EvenSystem 14](#_Toc4622365)

[3.5 Menu scene 14](#_Toc4622366)

[3.6 Szint generálás 15](#_Toc4622367)

[3.7 Játékos karakter 16](#_Toc4622368)

[3.7.1 Követő 18](#_Toc4622369)

[3.8 Ellenség 19](#_Toc4622370)

[3.9 Cél 19](#_Toc4622371)

[3.10 SzunetMenu 20](#_Toc4622372)

[3.11 Adatbázis 20](#_Toc4622373)

[3.11.1 Adatszerkezet 20](#_Toc4622374)

[3.11.2 Funkciók adatszintű leírása 20](#_Toc4622375)

[3.12 Tesztelés 21](#_Toc4622376)

[3.13 Továbbfejlesztési lehetőségek 22](#_Toc4622377)

[4 Összegzés 23](#_Toc4622378)

[5 Irodalomjegyzék 23](#_Toc4622379)

[Mellékletek 26](#_Toc4622380)

Ábrajegyzék

[1. ábra 6](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622757)

[2. ábra 6](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622758)

[3. ábra 7](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622759)

[4. ábra 7](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622760)

[**5**. ábra 8](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622761)

[6. ábra 9](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622762)

[7. ábra-UnityEngine 11](file:///D:\Szakdoga\Zarodolgozat_TápaiÁrpád.docx#_Toc4622763)

# Bevezetés

Szakdolgozatom témájául a MagIx-et válatoszttam mint külsőnézetes, mágikus kalandjátékot, amit leginkább a Zelda inspirált bennem. Szerencsére manapság már rengeteg játék motort lehet találni amivel könnyen lehet 3d-s játékot tervezni, amit a felhasználó is jobban tud élvezni. Amennyire a Zeldát vettem alapul, oly annyira más játékokat is. Inkább szerettem volna egy általános RPG-re hajazó játékot csinálni mintsem egy Zelda kopit. A játék csak is single-game élményt nyújt, és nem tervezek hozzá multiplayert. Az adatbázisban mentjük le a játékos adatait, azt, hogy mennyi pontot szerzett, és melyik pályán tart éppen. A játék elején egy segítő pálya van, ami megtanítjaa játékosokat a karakterek és az egyéb tárgyak kezelésére.

## Miért pont RPG?

Olyan játékot szerettem volna tervezni amit én magam is szívesen játszanék. Gyerekkorom óta Gamer-nek tartom magam, és számtalan játékot vittem végig és élveztem évekig. Így mindenképpen játékot akartam tervezni. illetve a továbbiakban is szívesen foglalkoznék ilyen munkakörben.

## Futtatási környezet

Azért gondoltam a Unity játék motorral való programozásra, mert itt adot a 3D-s felület amiben szerettem volna megvalósítani a programomat. Itt egyszerűen lehet a kódolást grafikai formában megjeleníteni ami élvezhetőbb játékélményt nyújt. C# nyelven programoztam, mert ez volt az alapértelmezett a Unity motorhoz , de a server-adatbázis részt is könnyen lehet vele kezelni.

# Fejezet

## Rendszer követelmények

A játék futtatásához a következő hardver, illetve szoftver követelmények szükségesek:  Windows 7 vagy Windows 10 operációs rendszer,  1 gigahertzes (GHz) vagy gyorsabb processzor.

* 1 gigabájt (GB) RAM (32 bites rendszerhez) vagy 2 GB RAM (64 bites rendszerhez),
* 2Mb szabad lemezterület.
* DirectX 9 grafikus eszköz WDDM 1.0 vagy újabb illesztőprogrammal, illetve  .NET Framework 3.0 vagy nagyobb.

## Felhasználói útmutató.

A játékot egy megfelelő mappába bontsuk ki, majd a MagIx.exe kétszeres kattintásával el tudjuk indítani a játékhoz szükséges opciópanelt. Először egy beállítás menüt kell hogy kihozzon, ahol be tudjuk állítani a képminőséget, képfelbontását, teljes képernyőben szeretnénk-e, illetve melyik monitoron akarunk játszani. Felül van egy input fül ahol megnézhetjük az irányításhoz szükséges billentyűket.

lent jobb oldalt a „play!”-re kattintva elindíthatjuk a játékprogramot, vagy a „quit”-el kiléphetünk.

Felhasználói felület

Mikor elindul a játék, elsőként a játék Engine logója tűnik fel majd utána megjelenik a menü.

1. ábra: A felhasználói felület terve (forrás: Saját készítés)

Felhasználói felület- Menü

A menüben 3 opciót választhatunk.(lásd:1-es ábra)

Legfelül található a **Játék** gomb. Alatta a **Beállítás** gomb, legalul pedig a **Kilépés** gombot találhatjuk.

1. ábra

Játék gombra kattintva előjön egy Almenü amint szintén 3 lehetőségünk van választani.  
 Elsőként az **Új Játék** gomb, amivel az új játék kreálása menüre ugorhatunk ahol a nevünket megadva kezdhetünk egy teljesen játékot

Másodikként a **Betöltés** gomb amivel folytathatjuk az eddig játékunkat.

Harmadikként a **Vissza** gomb amivel vissza térhetünk az Főmenübe.

Beállítás gombra kattintva előjön a beállítás Almenü.

Itt beállíthatjuk a hangok erősségét.

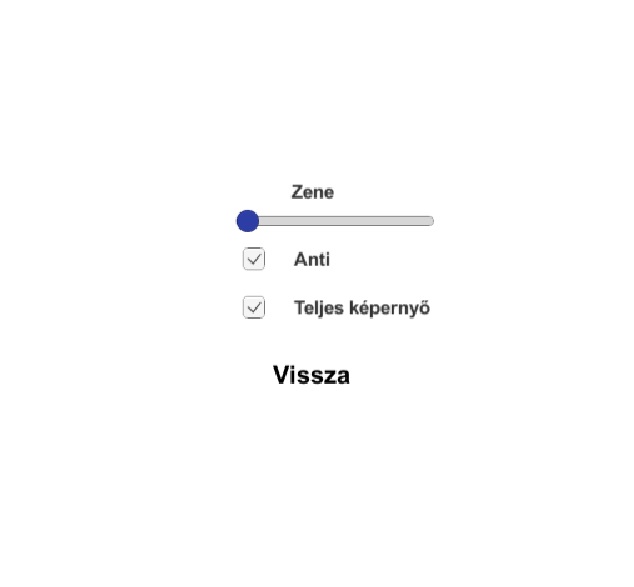
2. ábra

A teljes képernyőnél be lehet állítani az alkalmazás teljes képernyőre állítását

 A **Vissza** gombra kattintva vissza mehetünk a Főmenübe.

A **Kilépés** gombra kattintva pedig leállíthatjuk a játékprogramot.

3. ábra



## Felhasználói felület- HUD

4. ábra

Mikor az Új Játék gombra kattintunk , elindul a játék, és kezdetét veheti a tutorial szint ahol a játékos elsajátíthatja, és megértheti a játék alapvető menetét.

A képernyő tetején található a játékos élete.

A bal oldalt a képességünk használati ideje.

A jobb oldalt meg az eddig gyűjtött aranyérménk.

A karakter irányítását a **W A S D** gombokkal és az egér mozgatásával tudjuk elérni.

A képesség használatát az **F** gomb lenyomásával, míg az interakciókat az **E** gomb lenyomásával tudjuk megtenni

Az ellenség életerejét a fejük fölötti piros indikátor jelzi

## Felhasználói felület- JátékMenu

Ha játék közben ki akarunk lépni, vagy szüneteltetni akarjuk a játékot azt az **ESC** gomb megnyomásával tudjuk elérni.

Ilyenkor felugrik 2 opció. **Folytatás** Gomb amivel folytatni lehet a játékot, vagy a **Kilépés** gomb amivel vissza térünk a Főmenübe.

**5**. ábra

## Elindítás

Amikor új karakter kezdünk először szembe találjuk magunkat a tutorial szinttel ahol elsajátíthatjuk a karakterünk mozgatását, és képességei használatát. Továbbá megismerhetjük ellenségeink támadását és a játék irányítását.



6. ábra

## Felhasználói felület- Játék célja

A játékos random generált szinteken vesz részt, ahol végig kell mennie a szobákon, míg el nem ér a MagIx kőhöz, vagy meg nem hal. Ha elérkezünk a kívánt kőhöz két opciónk marad, vagy megöljük az összes ellenséget ezen a szinten, vagy tovább megyünk a következőre ahol újabb megpróbáltatásokon megyünk majd keresztül, de az igazi cél, hogy minél több ellenséget öljünk meg, így kerüljünk a ranglétra tetejére.

A pályán fellelhetőek a képességekkel rendelkező őrzők, akik hozzánk verődve ruháznak fel minket képességeikkel. Ezek segítségével tudjuk legyőzni a 2 típusú ellenséget, és szerezhetünk minél több pontokat.

Az első fajta ellenség a közeli, akihez ha közel kerülünk egyből el kezd sebezni minket. A másik típusú a távoli aki ha észre vesz, egyből el kezd minket lőni, amivel nagy sebzést mér ránk, így mindenképpen el kell kerülni ezeket. Szerencsére ezek a lövedékek másokat is sebeznek, így ki lehet játszani az ellenségeket, egymás ellen. Megfontolt mozgás kötelező, hiszen az ellenséges lövedékek nagyot szakíthatnak az életünkből, de nem csak a lövedékekre, hanem a szoba minden részét átvizsgálva járjuk a különböző helyszíneket, nehogy egy váratlan támadás esetén elfogyjon az életpontunk. Az élményt tovább fokozza, hogy a pályák, az ellenségek és a cél helye is változik szintéként, így elképzelhető hogy akár egyből előttünk lesz a cél, de az is lehetséges, hogy át kell verekednünk magunkat megannyi szobán a cél elérése érdekében.

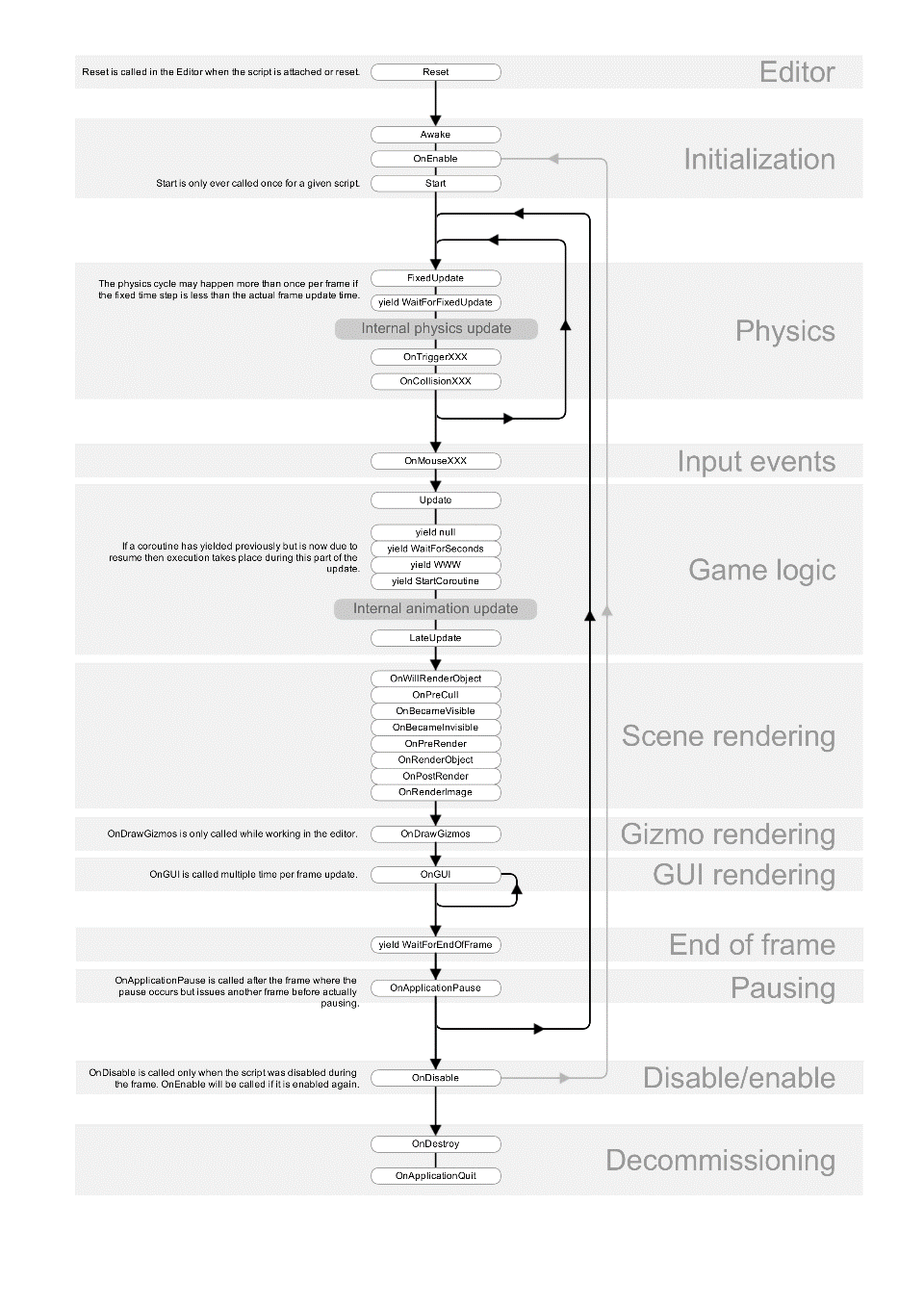
# Fejlesztői dokumentáció

## Megvalósítás

A megvalósítás során Unity enginben három különálló sceneken és projekteken kellett dolgoznom. Kellett egy adatbázist létrehozni ahol a játékosok eddigi pontjaikat lehetet tárolni, és magát a pályákat ahol a játékos mozog illetve a mögötte lévő scripteket is létre kellett hozni.

## Magáról az engineről

A Unity egy olyan videojáték-motor, amivel háromdimenziós illetve kettődimezniós videójátékokat

A **Unity**  egy olyan videojáték-motor, amelyet segítségével háromdimenziós illetve kettődimenziós játékokat, 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.

7. ábra-UnityEngine

A Unity szoftverrel való videojáték-készítés lehetséges Microsoft Windows vagy Mac OS X operációs rendszerek használatával, a játékmotor segítségével létrehozott játékok pedig futtathatók Windows, a Mac OS X, Xbox 360,PlayStation 3, Wii, iPad, iPhone vagy Android alatt. A Unitynek két főbb alkotó része van: az egyik játékok fejlesztésére és tervezésére használható szerkesztő, a másik pedig maga a videojáték-motor, amely a végleges változat kivitelezésében nyújt segítséget, ám

háromdimenziós modelleket nem képes létrehozni, így azokhoz mindenképp szükséges egy 3D modellező program.

## Kliens által alkalmazott osztályok és metódusok

A kliens program az, amit a felhasználók használnak. A játék csak egyjátékos módra lett tervezve. A projectet UNITY–ben készítettem. Két főbb részből tevődik össze: a menüből és magából a játékból. Ezzel a két főbb résszel több osztály az-az scriptek is foglalkoznak. Míg a menuben a Menu osztály a fő addig a játékban a Mindkét osztály egy WPF form, így egy XAML fájl is és egy .cs fájl is tartozik hozzá. A többi három osztály (név szerint Card, CardMini és CardBig) csak a másik kettő osztály teljes értékű működéséhez szükséges.

### MonoBehaviour

Mi is az a MonoBehaviour és miért fontos ez a Unity program programozásában?  
A[1] MonoBehaviour az alaposztály, amelyből minden Unity script származik. A C # használatakor kifejezetten a MonoBehaviorből kell származnia. Olyan életciklus-funkciókat kínál, amelyek könnyebbé teszik az alkalmazás és a játék fejlesztését.

Például néhány fontos methodus:

Start ()

Update ()

FixedUpdate ()

LateUpdate ()

OnGUI ()

OnDisable ()

OnEnable ()

## GameObejectek és prefábok

### GameObejectek

Mivel a Unity programozásához elengethetetlen a grafikus felület kezelése, elkerülhetetlen, hogy létrehozzunk egy-egy GameObjectet és prefábot a könnyebb proramozás érdekében.

Ezek a Untiy alapvető objektumai, amelyek kellékeket, karaktereket és képekekt, geometriai objektumokat képviselnek. Nem önmagukban, hanem a komponensek gyűjtőjeként működnek, amelyek megvalósítják a funkcionalitást.

Például egy Kockaobjektumot hoz létre, ha egy kocka komponenst egy GameObjecthez csatol.

Leginkább ezekre a GameOjbektekre helyezve a scripteket tudjuk életre kelteni a grafikus felületen lévő c#-ban megírt methodusokat. Itt lehetőségünk van a változókat vagy másik GameObjektum azonosítást akár grafikus felületen megadni.

A GameObjektek a sceneben léteznek (ez megjelenik a Hierarchiában, és a Unity egyéni adatbázisformátumába sorolódik a Könyvtár mappában). A GameObejectek más, különálló Gameobjektumokat is tartalmazhatnak mint gyerek.

Közvetlenül létrehozhat egy új GameObejectek, de létrehozhat egyet úgy is , hogy előhívja a GameObejectet, amit aztán projekt nézetből a scenere húzhat . Az előlapon végrehajtott változtatások az összes példányt érintik. Az előfordulások értékei megfordulhatnak az előlap felülbírálásához, de ha hozzáad vagy eltávolít egy alkatrészt, akkor az előcsatlakozás megszakad.

### Prefábok

A prefab egy újrafelhasználható eszközzé alakított GameObject másolata - megjelenik a Projekt mappában, és a eltárolva fájlként sorosítva van. Az előlapok tartalmazhatnak GameObject hierarchiáját.

Prefáb létrehozásához először létre kell hoznia egy üres prefabot majd bele kell húzni egy GameObjectet (a Hierarchiából) .

Az Unity Prefab rendszere lehetővé teszi a GameObject létrehozását, konfigurálását és tárolását az összes összetevőjével, tulajdonságértékével és a GameObjects gyermekével újrafelhasználható eszközként.

A Prefab eszköz sablonként működik, amellyel új előkészítő példányokat hozhat létre a jelenetben

### Scene

A Scenek tartalmazzák a játékos felé bemutatásra szánt tartalmat, mintha egy jelenet lenne. Ide tartozik a bevezető videó, a menü, a játék különböző szintjei mind scenekben van elmentve.

### EvenSystem

Az EvenSystem Kezeli a bemeneti, eseményeket és küldő eseményeket. Az EventSystem felelős az Unity Scene eseményeinek feldolgozásáért és kezeléséért.

## Menu scene

A menu sceneben található a menu prefab ami egy canvas objektum. Ez 6 további objektumra van felosztva. Az első a háttér mely a látó felület háttér képi megjelenéséért hivatott. Az első fontosabb objektum a **Jatek menu** amiben található további 3 gomb illetve a **Menu script** is. A gomboknak a text gyerek felelős a gombon lévő szöveg megjelenítéséért. Az első a játék gomb amin kettő **On Click()** beépített metodus található. Az első láthatóvá teszi a **játék obejctumot**, a második pedig eltünteti a jelenleg látható objectumot. A második gomb az Opcio gomb ami ugyan ezen az elevn működik csak az Opcio Objectumot nyitja meg. A harmadik gombbal pedig Kilépésgomb ahol a gomb rákattintással meghívjuk a **Menu** **void kilép()** metodusát ami meghívja az **Application.quit()** beépített metódust amivel leállítja a játék programot.

A Jatek objectum alatt szintén 3 gomb van. Az első gomb az **Új Játék** gomb szintén kettő On Click() függvénye tartalmaz ami a **Krealas objeckt** betöltésére hivatott. A második gomb a **Betoltes** gombal tudjuk elérni az eddigi jatekos adatait. A **Vissza** gombal pedig bekapcsoljuk a **Jatek menut** miközben a **Jatek** objecktumot kikapcsoljuk.

A **Krealas** objectumban 2 gomb található és egy input field. A **Vissza** gomb szintén megjeleníti a **Jatek Menut** miközben eltünteni a **Krealas** objectet. Az **nevmegad** input filedben megadhatjuk a karakterünk nevét amivel szeretnénk játszani. Az Új Játék gomb egyik On Click() függvénye meghívja a **menu.játszás()** methodusát ami pedig a **SceneManager.LoadScene(int)** függvényét hívja meg a megadott scennel.

## Szint generálás

Ez az Osztály felelős azért, hogy új pálya generálódjon amit a játékosnak majd teljesítenie kell. Elsősorban a tagek alapján bekért GameObjektek tömböket meg kell formázni, és név sorrendbe kell tenni a könnyebb kezelhetőség érdekében ezért létrehozunk egy **Gameobjekt**[] **listarendezo(string)** függvényt ami a beépített **Array.Sort()** függvénnyel végig megyünk és sorba rakjuk az **int nevosszehasnolit(gameObject, Gameobejct)** függvény eredményeit annak a tömbnek amit vizsgálunk. Ezek után kezdődhet a pálya alap generálása.

**void Alapgen():** metódus a felelős az alap terület generálásáért. Az előre bekért pálya területe az Z és az X koordináta mentén, illetve az előre bekért fal típusokból és padló típusból felhúzza az alap falakat és a padlókat a unity beépített **Instantiate**(Object, vector3 , quaternion) függvényével

Ezek után lefut a **void Szobagen():** metodus **.** Itt létrehozunk kettő 2 dimenziós tömböt amiben a falak generálódnak , 2 változót amiben majd a random számok lesznek generálva a **Random.range(**int, int**)** függvény segítségével a falak típusa hosszából. Ezek adják majd meg hogy milyen típusú falat generáljon a program a következő szobában a **Instantiate**(Object, vector3 , quaternion) függvénnyel. Előszőr az **X** koordináta mentén generáljuk le a falakat. kettő **for()** ciklus segítségével ami végig megy a megadott terület nagyságán. Minden generáláskor vizsgáljuk a generált falat, és ha nem átjárós fal akkor egy változót növelünk egyel. Ha abban a sorban az utolsó falig akkor összesírjük az adatokat és megnézzük, hogy volt-e legalább 1 átjáró, ha nem akkor az utolsó mindenképpen az, így elkerülve a zsákutcákat.

Ezek után a **Z** kordináta generáljuk le a random falakat kettő **for()** ciklus segítségével. Itt vizsgálni kell a **X** koordináta mentén lévő falakat is, hogy van-e minimum 2 átjáró a szobában, mivel így lehet elkerülni a zsákutcákat, amik megakasztanák a játék folyását.

Aztán jön a tartalom generálás amit a **void tartalomgen()** metodus biztosít. Itt szintén 2 **for()** cikluson megyünk végig a szobákon úgy hogy közben random helyekre random ellenséget, és segítőket generálunk a **Instantiate**(Object, vector3 , quaternion) függvény és a **Random.range(**int, int**)** függvény segítségével.

## Játékos karakter

A játékos karakter objecktben több script is található. Az egyik fő script a JatekosStat. Ez felelős azért, hogy a karakter adatait mozgatni lehessen az egyik sceneről a másikra, mivel amikor teljesítünk egy pályát egy új pálya generálására kerül sor egy másik scenen. Itt tároljuk a játékos jelenlegi életét, pontjait, és nevét, illetve innen változtatjuk meg a canvasra kiírt értékeket is. Továbbá itt található a void **EletValtoz(float)** metodus is ami a játékos életének a változásáért felelős. Ezen felül még találhatóak itt az adatok megváltoztatására és lekérdezésére setterek és getterek.

A karakter irányításáért a Jatekosiranyito és a Jatekoskarakteralap a felelős. A játékosiranyíto scriptben meghatározzuk a fő kamerának a nézési irányát és átadjuk a Move() metodussal a Jatekoskarakteralap osztálynak ahol a bemeneti adatok alapján meghatározza a karakter irányát.

A kamera mozgatásáért a Kamera class felel. Itt bekérjük a Játekos pozicíióját, és a kamera pozicióját megegyezővé teszem, hogy amikor a Játékos elindul, az végig tudja követni. Mivel a karakter figyeli a kamera nézési irányát, ezért a kamera mindig a karakterre néz a **LookAt()** beépített függvénnyel. Mivel a középpontban a karakter áll könnyedén tudom körülötte mozgatni a kamerát mivel lekérem az egér irányát az **Input.GetAxis()** beépített függvénnyel majd a **Quaternion.Euler()** beépített függvényben átadjuk ezeket az adatokat.

Tudom növelni illetve csökkenteni a kamera és a játékos közötti távolságát a **void görgetés()** metódussal , ahol lekérem az egér görgőjének a mozgását az **Input.GetAxis()** beépített függvénnyel és ennek float értékét vizsgálom egy bizonyos értéken belül, majd növelem vagy csökkentem a **tavolsagkarakter** változót ami meghatározza milyen távol legyen a kamera a játékostól. Továbbá a játék kezdetén itt kapcsoljuk ki az egér láthatóságát a **cursor.visible** parameter valtoztatasaval és az egeret a monitor közepére lockoljuk a **Cursor.lockState** parameter változtatásával.

A Jatékosnak van egy alap támadása amit az Alaptámadas osztály irányít. Amikor elkezdődik a játék bekérjük a lövedéket és a lövedék pozícióját. Majd vizsgáljuk, hogy lenyomtuk-e a bal billentyűt az egerén, és ha igen akkor létrehozunk egy klónt a Instantiate() beépített függvénnyel játékos pozíciójában plusz amennyivel el akarjuk tolni és adunk annak a golyónak egy erőt arra amerre a játékos néz az AddForce() függvénnyel. A golyó a beépített Rigidbody-nak köszönhetően ami fizikai környezetbe helyezei az objektumot és amit hozzá adtunk a Golyóobjektumnak, érzékeli ezt az erőt és elindul előre.

### Követő

A pályán fellelhető néhány erősítő kocka amit ha megszerzünk egy követő szegődik mellénk akinek külön támadása van. Kezdetben a követő objektum rejtve van majd amikor hozzá ér a játékos collidere a erősítő kocka collideréhez a követő megjelenik az erősítő pedig eltűnik az erősítőben található **void OnCollisionEnter()** függvény segítségével.

A követőnek van egy támadás scriptje ami kezeli a követő támadását, és egy másik a helyzetét.

A Golyo script az ami a támadást kezeli. Ez nagyon hasonlít a játékos alaptámadás scripthez, annyival van kiegészítve hogy itt az F gomb lenyomásával tudunk lőni, és van egy időzítő, hogy csak bizonyos időközönként tudjunk lőni.

A follower script az ami irányítja a követőnket. Itt amikor bekapcsolódik elkezdődik egy időzítő a beépített **StartCorutine()** függvény, ami a megadott ideig fut le. Természetesen ezt ki is írja úgy hogy átadja az ido változó parameteret a játékos HUD-ban lévő képtölt text-be. Ezt a változót minden másodpercben 1-el csökkentünk, és ha lejár a megadott idő, eltűnik a követőnk és nem tudjuk igénybe venni a képességét sem. A követő helyzetét a játékos helyzetéhez kötöm a **koveto.transform.position**-val, amiben átadom a játékos helyzetét illetve forgatom a **Quaternion.Euler()** függvénnyel az egér irányának megfelelően úgy mint a kamerát, ennek köszönhetően egyszerre forogna.

## Ellenség

Az **AIcontroll** osztály felelős a sima ellenség irányításáért. Az ellenség és a játékos közti távolságot a **Vector3.Distance()** beépített függvénnyel vizsgálom. Két fázisa van ennek, a néz és a támad metodus. Ha egy x távolságra van a játékos akkor csak rá néz a játékosra, az-az a beépített **Quaternion.LookRotation()** függvény és a **Quaternion.Slerp()** függvény segítségével követi a játékosunkat Amennyiben csökken ez a távolság egy bizonyos pontig, az ellenséges karakter **transform.Translate()** függvény segítségével elkezd a játékos felé menni, és sebezni őt amit a **JatekosStatban** lévő **EletValtoz(float)** metodussal hívásával ér el.

Az ellenség még hordoz egy másik sciptet az Élet() scriptet ami az ellenség életének a megjelenítéséért, tárolásáért, és változásáért felel. Az itt található EletValtoz(int) metodus hívásával lehet az ellenség életét változtatni. A bekért értékkel növeljük a jelenlegi életét amit , majd kiíratjuk ezt egy csúszkába. Azt szeretnénk hogy a játékos mindig jól lássa az ellenség életét, így ez a csúszka mindig követi a kamerát a **LookAt()** függvény segítségével.

A másik ellenség típus annyiban különbözik ettől, hogy nem mozogni, csak a játékos felé nézni, viszont ő a játékos lövéséhez hasonlóan van egy lövés képessége amivel a játékost támadja.

## Cél

A pályák újra generálását és a pályák végét a cél nevű prefáb elérésével tudjuk megvalósítani. Az itt megtalálható Ujrageneral script vizsgálja a játékos távolságát a **Vector3.Distance()** függvénnyel, és ha megfelelő távolságban van akkor a játékos HUD-on aktívvá teszi az E text-tet amivel jelezzük a játékosnak hogy az E-gomb lenyomásával új pályát tudunk generálni. Ekkor a játékos adatai feltöltődnek az adatbázisba az **Adatokfel()** metodussal, majd újra legenerálja ezt a scenet a **SceneManager.LoadScene()** metodussal.

## SzunetMenu

A játék során meg tudjuk állítani az időt és ki tudunk lépni a menübe az esc lenyomásával. A SzunetMenu canvasra helyezett script a felelős azért, hogy ha lenyomjuk az esc-gombot megálljon mindenki és aktívvá teszi a canvas-on lévő gombokat, illetve az egér kurzorját is vissza állítja láthatóra, és mozgathatóra a **void Szünet()** metodus által. Itt a folytat gomb lenyomásával meghívóik a **void Folytat()** metodus, amivel az idő újra indul, a kurzor lezáródik és eltűnik, illetve a SzunetMenu canvas is eltűnik. A Kilépés gombra kattintva meghívóik a **void Menütöltés()** amivel betöltődik a Menu scene.

## Adatbázis

A játékban az egyik fő élvezeti faktort az nyújtja hogy minél tovább és minél több pontot érjünk el. Ezt természetesen menteni szeretnénk valahol, hogy aztán vissza nézzük így adott volt az adatbázis gondolata. Mivel nem kell több felhasználónak egyszerre hozzá férnie az adatbázishoz így az SQLite adatbázis-motort választottam. A könnyebb kezelhetőség érdekében a játékban lévő kapcsolat scriptben hozom létre kódból az adatbázist, majd a metódusai kezelik a játékos adatai feltöltését és legkérdezését.

### Adatszerkezet

JATEKOS – egy felhasználó adatait, nevét, pontjait, és életét tartalmazzák

JATEKOS: id, name, pont, elet

### Funkciók adatszintű leírása

**Felhasználók kezelése**:

Új felhasználó:

Új sor JATEKOS-ban: id auto increment, nev és elet megadása.

## Tesztelés

|  |  |  |
| --- | --- | --- |
| **Feketedobozos tesztelés** | | |
| **Funkció** | **Teszt** | **Eredmény** |
| **Első Játék indítása** | **Új játék indítása, új felhasználóval. Pálya indítása felhasználó kreálás nélkül.** | **Játék rendesen indul, nem ír ki hibát.**  **Játék elindul, viszont a pálya scene belépésekor a játékos élete 0-a mivel nincs adatbázisban érték.** |
| **Pálya generálás** | **Csak egyféle fallal generálás.**  **Csak 2 féle fallal generálás.**  **Szélesebb pálya generálás** | **Nem hajtódott végre a generálás ha az az 1 fal nem átjáró volt. Különben igen.**  **Szintén ha az egyik nem ajtó nem generálódott le a pálya.**  **Más bemeneti adatokkal is megfelelő pályát generált.** |

|  |  |  |
| --- | --- | --- |
| **Fehérdobozos tesztelés** | |  |
| **Funkció** | **Teszt** | **Eredmény** |
| **Boost megszerzése**  **JAVÍTVA** | **Több boost megszerzése egyidejűleg / aktív állapotban** | **Boost megszerzése esetén kapunk egy időzítőt, ha többet szereztünk meg hozzá adódott az időzítőhez.**  **Javítva, már nincs hatással ha megszerzünk többet, csak ha lejárt az előző** |
| **Ellenség támad** | **Több ellenség támad** | **Ha több ellenség támad össze adódik a sebzés. A távoli ellenségek golyója sebzi önmagukat is, és az eltalált ellenséget is.** |

## Továbbfejlesztési lehetőségek

Rengeteg továbbfejlesztési lehetőség és potenciál van a játékban. Ha csak abba megyünk bele, hogy ezernyi ellenség típust lehetne kreálni, vagy pálya tárgyakat az élmény fokozására is. Illetve mindenképpen véges pályákat generálnék csomópontokkal, ahol mondjuk egy Fő ellenség várja a játékosokat, majd minden újabb szinttel egyre erősebbek lehetnének az ellenségek, de egyre több aranyat is adnának. Mindenképpen kellene a játékba egy tároló rendszer, ahol tárgyakat képeségekkel felszerelve lehet gyűjteni. Kell egy árus, ahol lehet venni és eladni akár ilyen dolgokat, így lehetne valamit kezdeni az össze gyűjtött pontokkal. Ezek a tárgyak képességeket adhatnának a játékosnak. Több interakciós tárgy kellene, illetve lehetne berakni egyszerű puzzle elemeket is, amik megoldása vagy valami értékes tárgyat adna, vagy pedig a tovább haladáshoz szolgálna kapuként. Továbbá tárgyaknak lehetne létrehozni egy külön adatbázist, meg az ellenségeknek is, és akár meg lehetne szabni, hogy ki melyiket tudja csak használni. Adatbázisból lehetne megcsinálni a pálya generáláshoz szükséges feltételek megszabását, hogy melyik pályára milyen eszközök és ellenségek lehetnek.  
Végül lehetne tárolni a legyőzött Főellenségeket és a kivégzett ellenségek számát is. Akár statisztikai adatokat is csinálhatunk ki mennyi idő alatt tudta teljesíteni az adott szintet.

# **Összegzés**

A program készítése során rengeteg mindent tanultam. Mikor elkezdtem írni a programomat, előtte még nem dolgoztam semmilyen másik motorral, vagy hasonló 3dtervező programmal. Elsősorban meg kellett tanulni a kezelő felületét, majd hogy hogyan van felépítve a program. Illetve a programozás része is egyedinek mondható. Ezért előtte készítettem több kisebb programot is, hogy megtanuljam ennek a működését. Minden esetben játék programokat fejlesztettem 3d térben, hogy később akar fel tudjam használni más projektekben. A motorban rengeteg beépített funkció használatára van lehetőség, ami megkönnyíti a dolgunkat. Az első nehézséget az jelentette számomra, hogy-hogyan érik el egymást scriptek. Rá kellett jönnöm hogy az objektumokon lévő scriptek elérését vizsgálni kell, hogy a jelenlegi scenen helyezkednek-e el, és ténylegesen azt akarom-e elérni. Ugyan így az objektek bekérésnél különbözőnek számít a scenere kihelyezett objektum és annak lementett változata az asset mappában. Illetve a programozás részében figyelembe kellett vennem, hogy frame alapján fut a program és hívja meg a metodusokat. A másik legnagyobb nehézséget az adatbázis használata jelentette. Nem tudtam, hogy a játék kibuildelése utn hogyan kezeli az adatbázist, és mivel sajátot akartam, nem szervereset, mindenképpen local adatbázist szerettem volna csinálni amit a felhasználó minden előre telepítés nélkül tud használni. De szerencsére egy kis utána nézéssel rájöttem hogy az SQLite az amivel ezt a leg egyszerűbben megoldhatom.

Nagyon sok idő telt el az első terv és a befejezett munka között. Ha elölről kezdeném akkor rengeteg mindent másképp csinálnék. Maga a tervezés folyamatát is az engine képességeihez mérném.

# Irodalomjegyzék

(2019. 03 31). Forrás: https://www.sqlite.org/about.html

(2019. 03 31). Forrás: Unity: https://unity.com/

(2019. 03 31). Forrás: SqlIte: https://www.sqlite.org/index.html

(2019. 03 31). Forrás: Youtube: https://www.youtube.com/watch?v=-m9tDF2v-5Q

(2019. 03 31). Forrás: Youtube: https://www.youtube.com/watch?v=e9l2wxkYg2E

(2019. 03 31). Forrás: Youtube: https://www.youtube.com/watch?v=TAGZxRMloyU&t=484s

(2019. 03 31). Forrás: Youtube: https://www.youtube.com/watch?v=TAGZxRMloyU&t=484s

(2019. 03 31). Forrás: Youtube: https://www.youtube.com/watch?v=zc8ac\_qUXQY&t=186s

(2019. 03 31). Forrás: Unity: https://docs.unity3d.com/uploads/Main/monobehaviour\_flowchart.svg

*GameObject*. (2019. 03 31). Forrás: Unity: https://docs.unity3d.com/ScriptReference/GameObject.html

*MonoBahviour*. (2019. 03 31). Forrás: Unity: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

*Prefabs*. (2019. 03 31). Forrás: Unity: https://docs.unity3d.com/Manual/Prefabs.html