Budapesti Műszaki Szakképzési Centrum
Verebély László Szakgimnáziuma és Szakközépiskolája
54 213 05 Szoftverfejlesztő szakképzés

Snaky

Készítette:
Veres Dávid Márk

2020

NYILATKOZAT A SZAKDOLGOZAT EREDETISÉGÉRŐL

Alulírott Veres Dávid Márk ..... a **BMSZC Verebély László Szakgimnáziuma és Szakközépiskolájának 54 213 05 OKJ Szoftverfejlesztői képzésében** részt vevő hallgatója büntetőjogi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy

a ……..

.....

című szakdolgozat saját, önálló munkám, és abban betartottam az iskola által előírt, a szakdolgozat készítésére vonatkozó szabályokat.

Tudomásul veszem, hogy a szakdolgozatban plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás nélkül,
- tartalmi idézet hivatkozás megjelölése nélkül,
- más publikált gondolatainak saját gondolatként való feltüntetése.

E nyilatkozat aláírásával tudomásul veszem továbbá, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Budapest, 2020. április 9.

Hallgató aláírása

Tartalomjegyzék

1.BEVEZETÉS

A globális piacon manapság a játékok nagy része egy platformon jelenik meg ami a mobil. A játékok bevételét pedig javarészt a mobil platform alkotja. Az emberek nagy része pedig, azért is választhatja a mobil platformot, mivel hordozható, változatos, és némely játék rendkívül addiktív. Az utóbbi időben pedig megjelent 1-2 világklasszis játék, amik pc-n kezdték pályafutásukat, amik később mobilon is megjelentek sokak örömére.

A gyerekek nagy része manapság több időt tölt el a mobilján (játékkal) mint tanulással, persze vannak nagyon jó tanulást elősegítő alkalmazások, de kevés olyan, amely a játékkal ötvözné a tanulást. Ezért is választottam ki, hogy olyan játékot készítek, amely ötvözi a kettőt, tehát a játék élményen kívül lehetőséget ad egy kis tanulásra is. A játék java részt Unity Engine-ben készült amit bárkinek tudok ajánlani játék fejlesztésére. Az unity egy viszonylag egyszerűen elsajátítható program, ezt ajánlják a legtöbben a kezdőknek. Mivel már egy pár éve használom a Unity-t ezért már van egy kis tapasztalatom a programban való fejlesztés terén, legutóbb egy országos versenyen kellett sokat használnom. Az így szerzett tapasztalatokkal nyugodtan vágtam bele a szakdolgozatom elkészítésébe.

1.1.A SZAKDOLGOZAT TÉMÁJA KIVÁLASZTÁSA

A téma kiválasztása során nem volt nagyon saját tervem, ezért a kiadott témák közül választottam egyet. A téma kiválasztása során két téma volt szimpatikus, egy társas játék elkészítése, és egy tanító jellegű action platformer. Hamar eldöntöttem, hogy az action platformer mellett teszem le a voksomat. A tájékoztató dokumentum egy alap leírást adott a témáról, Amiből kiderültek az alapvető feltételek.

2.FELHASZNÁLÓI DOKUMENTÁCIÓ

A játékot úgy fejlesztettem, hogy egyszerűen legyen használható, még annak is, aki keveset, vagy nem is játszott, még ilyen stílusú játékkal. A játékot egyszerű irányítással láttam el például a főmenüben a navigálás egyértelmű ikonokkal láttam el, a szint kiválasztásánál pedig egyértelműen láttszik melyik szintet lehet kiválasztani. Amikor a játékos rányom az egyik pálya gombjára, betöltődik az adott pálya, és már mehet is a játék. A játékban a felhasználó mehet jobbra-balra a bal alsó sarokban található gombokkal, a jobbalsó sarokban található gombokkal pedig ugorhat, üthet, vagy ha fel van oldva a dobás akkor dobhat. A játékban a játékosnak végig kell mennie a pályán ahhoz, hogy teljesítse. A közben szerzett pontok csak akkor kerülnek mentésre, ha a játékos végig megy a pályán, ha közben kilép akkor az addig szerzett érmék elvesznek. A pályákon persze nem olyan egyszerű végig menni, ugyanis

különböző akadályok hátráltatják a játékost, mint például az ellenségek, vagy a tüskék. A játékos ugyanakkor szerezhet különböző bónuszokat, ha felvesz egy könyvet, és eltalálja a benne feltett kérdésre a választ, így megkönnyítve a pályát. Mind a két szint végén található egy főellenség, akinek sokkal több élete van az átlagostól, és különleges képességei is vannak, viszont sokkal több érmét is dobnak a többinél.

2.1.RENDSZERKÖVETELMÉNYEK

Hardver:

- Processzor: 32bites 2 magos 1 GHz-es
- Rendszer memória (RAM): 2GB
- Szabad 40mb tárhely

Szoftver:

- Operációs rendszer: Android 5.0

2.2.JÁTÉKOSOK FELHASZNÁLÓI UTASÍTÁSAI

Telepítési útmutató

1. A DVD-n lévő Szakdolgozat/Snaky mappát másolja át egy tetszőleges helyre a telefonon.
2. Az átmásolt mappában lévő Snaky. Apk fájlal lehet telepíteni a játékot.

2.3.A JÁTÉKOS JOGOSULTSÁGAI

A játékosnak joga van játszani a játékkal, azaz tudja irányítani a karaktert. Megtekintheti az addig elért pontjait az, hogy hányszor halt meg, hányszor ölt, kapott egy életet vagy elvesztett. Ezen felül vissza állíthatja a játék beállításait. Joga van még vásárolni is a boltban, ha van elegendő pénze.

3.FEJLESZTŐI DOKUMENTÁCIÓ

3.1.FELADAT SPECIFIKÁCIÓ

A játék célja, hogy egy egyszerű, viszonylag könnyű játék menet mellett egy kis tudást is nyújtson a felhasználónak. A játék próbára teszi a felhasználó tudását különböző alaptárgyakban, mint például az irodalom, matematika, történelem. A feladat kidolgozásához a Unity3D nevű szoftvert használtam, de csak a 2Ds részét mivel egy 2Ds platformert terveztem készíteni. A fejlesztés során saját grafikát is alkalmaztam.

A játékot legalább középiskolás szintre terveztem mivel ők már vélhetőleg tanulták a feltett kérdéseket, de egy általános iskolás is tanulhat belőle. A szoftver ajánlott felhasználói tehát a középiskolások vagy idősebb személyek.

3.1.1.A JÁTÉK FŐBB FUNKCIÓI

- A karakter irányítása,
- Több szint, összesen 12db pálya teljesítése,
- Pont szerzési lehetőség,
- Menühasználat,
- Hordók megsemmisítése,
- Kövek felvétele és eldobása,
- Ellenséget kikerülése vagy megölése,

3.2.FEJLESZTŐI ÜTEMTERV

3.2.1.A JÁTÉK MEGTERVEZÉSÉNEK LÉPÉSEI

A játék tervezése először nem tűnt nagy feladatnak, de ahogy belekezdtem rájöttem, hogy fel kell darabolnom a szakaszokat és eltervezni, hogy mikor mit és meddig csinálok. Először a játék témáját kellett kitalálnom. Gondoltam, hogy idő utazós legyen, vagy modern, de végül a *Medieval* stílusnál maradtam, ami egy középkor szerű téma. Így a fő karakter egy középkori harcos lett.

A második lépés pályák megtervezése volt. Itt először is perspektív kamerát használtam nem pedig a szokványos periszkópikust. Így olyan hatást érek el mintha az emberi szemmel látnánk a pályát. A pályák hátterei hat különböző szintből állnak, amik különböző távolságra vannak egymástól. A hátterek ugyan olyanok minden pályán, de a pályák más magasságban vannak a háttérhez képest így viszonylag változatosnak tűnhet. Mivel egy pályán először megrajzoltam az első részt, amin a játékos megy, majd a sötétebb hátteret, végül pedig a növényeket. Ha pedig megvoltam ezzel akkor raktam le hordókat, amiből pénzt, és követ szerezhet a játékos, majd leraktam az ellenségeket és a pályát határoló elemeket. Miután kész volt egy pálya, leteszteltem, hogy ne legyen semmi hiba.

A harmadik lépésben a karaktert terveztem implementálni. Ebben a stádiumban rajzoltam meg a kinézetét, és a mozgását úgy hogy a felhasználó számára könnyű, és egyértelmű legyen, az

irányítás. A karakternek ebben a fázisban terveztem kialakítani a támadás mechanikáját, majd az életet. A játékosnak úgy terveztem, hogy három élete legyen összesen, amiből gazdálkodhat.

A játékban lesz pontszerzési lehetőség is, amit úgy gondoltam megoldani, hogy érméket dobnak az összetört hordók és a megölt ellenségek. Úgy terveztem, hogy kettő fajta érmét teszek a játékba, egy aranyat, ami ötöt ér, és egy ezüstöt, ami kettőt.

A játék fontos elemét képezi még a tanulás, amit egy könyvel oldottam meg ami minden pálya elején feldob egy kérdést, amivel valamilyen előnyhöz juthat a játékos, ha eltalálja a választ.

A játék animálását akkor terveztem elkészíteni amikor a játék fő funkciói megvannak, azaz a karakter, és az ellenségek tudnak már mozogni. A hordó széttörését pedig particle rendszerrel szerettem volna megoldani.

A játék menü rendszerét akkor terveztem megvalósítani amikor már van pár játszható pálya és van már egy alap amire ráépíthetem mondjuk a statisztika ablakot.

3.3.HATÁRIDŐK

3.3.1 ÜTEMTERV

Feladat neve	Tervezett határidő	Tényleges időpont
2D mozgás terve	2019.09.03	2019.09.03
2D mozgás megvalósítása	2019.09.04	2019.09.15
A karakter kinézetének megtervezése	2019.09.16	2019.09.17
A karakter megrajzolása	2019.09.17	2019.09.21
A pálya hat rétegű hátterének megtervezése	2019.09.21	2019.09.21
A pálya hat rétegű hátterének megrajzolása	2019.09.21	2019.09.23
A pályák megtervezése		
A pályák megtervezése	2020.01.03	2018.01.10

A pályaelemek megrajzolása	2020.01.11	2020.01.20
A pálya összerakása	2020.01.20	2020.01.28

Az ellenségek megtervezése, megrajzolása	2020.01.28	2020.02.01
Az ellenségek meganimálása	2020.02.01	2020.02.04
Az ellenségek mozgásának leprogramozása	2020.02.04	2020.02.06
A hordó széttörésének implementálása	2020.02.06	2020.02.07
Az érmék megtervezése	2020.02.07	2020.02.07
Az érmék megrajzolása, animálása	2020.02.07	2020.02.08
Az érmék leprogramozása	2020.02.08	2020.02.09
A könyv megtervezése	2020.02.09	2020.02.09
A könyv menüjének implementálása	2020.02.09	2020.02.10
A könyv menüjének megtervezése	2020.02.10	2020.02.10
A könyv implementálása	2020.02.11	2020.02.13
Az adattárolás megoldása		
Az adattárolás megtervezése	2020.02.14	2020.02.20
A json fájl rendszer megtervezése	2020.02.21	2020.02.23
A kérdések és válaszok elmentése json fileba	2020.02.23	2020.02.25

A kérdések beállítása a pályákon, bónuszok implementálása	2020.02.26	2020.03.06
A két főellenség megtervezése	2020.03.07	2020.03.08
A két főellenség megrajzolása	2020.03.08	2020.03.12
A főellenségek mozgásának, támadásának implementálása	2020.03.16	2020.03.19
A menü rendszer megtervezése	2020.03.20	2020.03.22
A menük megrajzolása, implementálása	2020.03.22	2020.03.26
A játék átnézése, esetleges hibák kijavítása	2020.03.26	2020.03.28
Publikus béta teszt megkezdése	2020.04.03	2020.04.03
Visszajelzések alapján hibajavítások	2020.04.03	2020.04.03

3.4.KÖVETELMÉNY SPECIFIKÁCIÓ

3.4.1.FEJLESZTÉS CÉLJA

A szakdolgozat kidolgozásáról konkrét terveim voltak. Először is 2D-ben gondoltam mivel a megadott stílus ezt diktálja. A játékban alap 2Ds mozgás van *double jump*-al kiegészítve. Az ugrás közben pedig irányítható legyen a karakter. A *tanuló jellegét* a játéknak úgy terveztem megvalósítani, hogy egy felvezető könyv által feltett kérdés és válaszok közül kellene kiválasztani a helyes választ a *honfoglaló*hoz hasonlóan.

A játék elkészítése során mindent nekem kellet megcsinálni, mint például:

- A játék pályájának, elemeinek megrajzolása,
- A fő karakter meganimálása, leprogramozása,
- Mind a 12 pálya megtervezése, implementálása,

- A pályák szebbé tétele, háttérrel, növényekkel,
- Az ellenségek leprogramozása,
- A pénzrendszer leprogramozása,
- Az akadályok implementálása,
- A menü rendszer létrehozása,
- Az adatok elmentését szolgáló rendszer létrehozása,
- A játék főbb fázisaiban tesztelni, majd az esetleges hibákat kijavítani.

3.4.2.FEJLESZTŐI KÖRNYEZET

A fejlesztői környezet eredetileg a Miscrosoft Visual Studio 2019 volt ahol a C# nyelven programoztam, majd egy hónap múlva váltottam Microsoft Visual Studio Code-ra amin ugyan úgy C#-ot használtam, viszont gyorsabban frissültek a kódok, és valamivel jobban személyre szabhatóbb. A C# nyelvet már négy éve tanulom így már van benne egy kevés tapasztalatom. A játék összerakására a Unity3D-t használtam, ami direkt játék fejlesztésre lett kifejlesztve. A játék grafikáját teljes mértékben a Piskel nevű ingyenes pixelenkénti szerkesztőben végeztem.

```
public void death(){
    for (int i = 0; i < spawnedObjects+PlayerPrefs.GetInt("moreLoot"); i++)
    {
      if (rng.Next(0,10) == 1)
      {
          GameObject hearth =  Instantiate(Hearth,new Vector3(transform.position.x,transform.position.y+1.2f,0),Quaternion.identity);
          hearth.GetComponent<Rigidbody2D>().velocity = new Vector2(rng.Next(2, 4),rng.Next(3, 6));
      }
      else
      {
          int rnd = rng.Next(0,1);
          GameObject coin = Instantiate(Coins[rnd],new Vector3(transform.position.x,transform.position.y+1.2f,0), Quaternion.identity);
          coin.GetComponent<Rigidbody2D>().velocity = new Vector2(rng.Next(2, 4),rng.Next(3, 6));
      }
    }
```

1. ábra, az ellenségek halál metódusa

3.4.3.FUTTATÁSI KÖRNYEZET

A futtatási környezet az a felület, szoftverek összessége, ami a játék futásához elengedhetetlen. Az én esetemben ez nem igényel túl sok magyarázatot, mivel az alkalmazásom Android rendszerre épül, így csak egy 5.0-ás minimum verzió szükséges a futtatáshoz.

3.4.4.FELHASZNÁLÓI FELÜLET

A játék főmenüjébe lévő felület teljesen animálva van, az egyik menüpontból a másikba átcsúszik a kamera. A UI azaz (User Interface) magyarul felhasználói felület.

3.4.5.MINŐSÉGI KÖVETELMÉNY

A program futása közben előkerülhetnek esetleges hibák, legyenek ezek kezeltek vagy kezeletlenek. Alábbiakban összegyűjtöttem a kezelt hibákra vonatkozó hibaüzeneteket, valamint ezekről egy részletesebb leírást.

3.4.6.TERJESZTÉS MÓDJA

A programot tárolhatjuk telefonon, pendrive-on, vagy bárhol. A programot bárki terjesztheti, mivel egy kisméretű apk telepítőről van szó ezért akár üzenetben is elküldhetjük, vagy Bluetooth-on is.

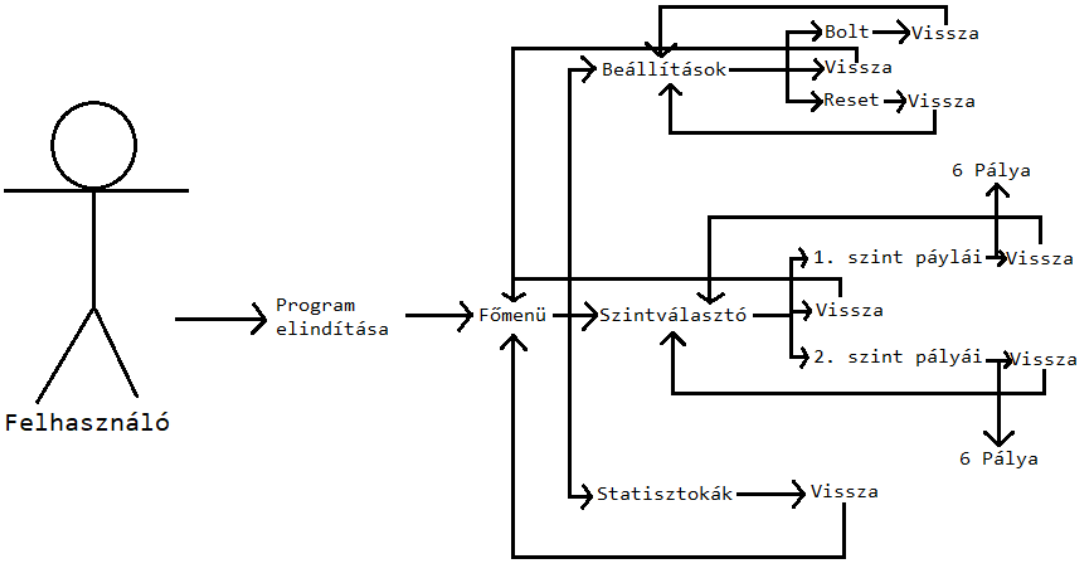
3.5.RENDSZER SPECIFIKÁCIÓ

3.5.1.RENDSZER HASZNÁLÓI

Olyan emberek a használói a szoftveremnek, akik szeretnek telefonon játszani, vagy szeretik ezt a bizonyos műfajt, esetleg szeretnének egy kicsit tanulni. A program más célokra nem is használható fel.

3.6.RENDSZERHASZNÁLATI ESETEK

A felhasználó szemszögéből egyértelműnek kell tűnjön a program használata. Ezt a problémát a Unity3D nagyon jól tudja kezelni, nem kell számítanunk hibákra. A futás során egy fontos feltétel az, hogy a *User* ne kerüljön szembe olyan hibával, hogy belemegy egy menübe és nem tud onnan kilépni. Tehát nemszabad lenni zsákutcának sehol.



2. ábra, a használati esetdiagramm

3.7.MENÜ-HIERARCHIA

A menü-hierarchia fő célja, hogy meghatározzuk a program menü-szerkezetét, és hogy mi melyik után következik.

A játék elindításakor egy főmenü fogad, ahol három választási lehetőség van: Statisztikák, Beállítások, és a játék elindítása, azaz a szint választó. A Beállítások menüben találhatjuk a boltot, amiben tud a felhasználó vásárolni, vagy visszaállíthatja a játékot, azaz kitöröl minden elmentett statisztikát. A Szint választó menüben két lehetőség van a tovább lepéshez, az első, és a második szint. Ha a játékos kiválasztotta szintet megjelenik egy újabb ablak, ahol kiválaszthatja a felhasználó melyik pályán szeretne tovább haladni, itt csak azok a pályák jelennek meg amiket már feloldott a játékos. A pálya választóból kettő fajta van, az 1., és a 2. szintre vonatkozóan. A játékon belül egy megállítás menü van, ami a jobb felső sarokból érhető el. A megállítás menüben van egy egyszerű felület található, amin egy újrakezdés, folytatás, és a menübe visszaléptető gomb található.

3. ábra, a játék menüi egyben

3.8.FIZIKAI KÖRNYEZET

A környezetet a Unity Engine 2D-s részének segítségével úgy tudtam megalkotni, hogy a lehető legjobban illeszkedjen össze a környezet, és a karakter. A játék hátterét hat szint alkotja, azaz van egy kék háttér leghátul, egy nagyon sűrű és világos zöld erdő, majd a következő négy szintben egyre ritkább az erdő és sötétebb. A háttért legjobban a perpektív nézet segítségével lehet a legjobban szemléltetni, amivel úgy láthatjuk a hátteret, mint a valóságban. A játékban elhelyeztem *particle systemet* ami egy effect rendszer amiben mindenféle efekteket lehet animálni mint például füst. Az én esetemben ezt a rendszert a hordók széttörésére utáni hordó darabok efekjeként használtam. A hordó effektjéhez először meg kellett rajzolnom a szétesett darabot, ami 16x16 pixel széles. Utána ezt importálnom kellett unityba ahol a kép típusát nem spritera, hanem textúrára kellett állítanom. A textúrából pedig létrehoztam egy materialt és a particle systemben kijelöltem. A játék környezetét úgy alkottam meg hogy viszonylag változatos legyen, ne legyenek olyan laposak a pályák. A játékban a dupla ugrás igen fontos szerepet játszik mivel a pályákat vertikálisabbra csinálhattam, így változatosabb lehetett. A játékban kész szint található, a legfőbb különbség az, hogy a második szintnek sötétkékes színűek a pályái, és persze kicsit nehezebbek is. A pályák három képszintből tevődnek össze: az első, amin a játékos tud mozogni, ütközni, ez a világosabb. A második ami az első mögött, takarásban helyezkedik el, ez a háttér ami kicsit sötétebb mint az előtte lévő, és a karakter persze nem tud érintkezni vele. A harmadik szinten találhatók a növények, kövek, és a tüskék is mivel egyszerre egy helyen nem lehet 1 szinten 2 elem ezért külön szintet kellett csinálnom az ilyen tárgyaknak.

3.8.1.KÜLSŐ RENDSZEREK

A fejlesztés során felmerült a külső online adatbázis használata, de mivel túl sok új plugint kellett volna hozzáadni és persze internet kapcsolat kellett volna, ami egy ilyen kis 2D s játéknál nem igényelt. Tehát így nincsen semmilyen külső rendszer a játékomban

3.8.2.KÉPERNYŐTERVEK

A játékom mivel Unity Engine-ben fejlesztettem ezért úgynevezett scene-ket azaz színeket alkalmazok, ami egy-egy játék részét reprezentáltja. Úgy terveztem a játékot, hogy legalább két szintje legyen, és azon belül 6-8 pálya. A játék elejére, azaz első indításnál egy kis történet scene-t gondoltam elhelyezni, de ez kikerült mivel ebbe a játékban nincs annyi történet. Végül a menü kapott egy külön scenet, és a két szint összesen 12 pályája. A menü scene-ben zajlik miden kiválasztása, beállítása. Így összesen 13 scene van a programban.

3.9.FEJLESZTŐI ESZKÖZÖK

A fejlesztői eszközök azok, amelyek elősegítik a fejlesztés menetét, és felhasználásra kerülnek. Elsősorban az Unity által fejlesztett Unity Engine-t használtam, ami jobbára 2Ds játékok fejlesztésére lett megtervezve. A program kódokat először Visual Studio 2019-ben írtam, majd váltottam Visual Studio Code-ra, amiben több lehetőség volt kiterjesztésekre, és sokkal gyorsabban lefuttatta a kódot unityhez. A játék kinézetét teljes mértékben a Piskel program segítségével csináltam, ami egy nagyon leegyszerűsített pixel rajzoló, ami könnyen használható, és nekem tökéletesnek bizonyult. Ebben a programban rajzoltam meg a kinézetét, azaz a pályákat, a növényeket, és a karakter, érmék, ellenségek animációját is képkockánként. A játékban változó vászon méreteket használtam. A pályaelemeket mind 16x16os pixel méretben rajzoltam meg, a nagyobb elemeket, mint a karakter, vagy némely ellenség már 32x32, vagy 64x64esben rajzoltam meg, ez persze nem jelenti, hogy maga a karakter 64x64es lenne, viszont a támadás miatt kellett a nagyobb vászon, hogy jól látszódjon az animáció.

3.10.ARCHITEKTÚRÁLIS TERV

Az architektúrális tervezés során meg kellett oldanom, hogy kommunikáljanak egymással a kellő scriptek. Itt elég sok dolgom volt mivel egyszerre sok miden történik, és nem lehet egy scripten minden. A felhasználói felület gombjai különböző scriptekre mutattak, a mozgásgombok a mozgás scriptre a támadás pedig a támadás scriptjére. A játékban vannak még olyan objektek amit egy másik hoz létre, például az egyik főellenség akadály objecteket hoz létre amit az ő pozíciójától hajít el. Mindig eltárolom az aktuális akadályt egy változóba,

hogy legyen valami referenciám a létrehozott objectre. Így, ha meghal a főellenség akkor az éppen eldobott akadály is megsemmisül így elkerülve a kellemetlenségeket.

3.11.ADATBÁZIS TERV

A programomban az adatok tárolását json könytárban oldottam meg mivel az sqlite 4hét próbálkozás után sem sikerült megoldani, hogy működjön telefonon. Mivel nem akartam platformot váltani ezért más módszerhez folyamodtam, ami a json lett. A játékban a felvehető könyvek kérdéseit és válaszait tároltam el. Csak beállítottam, hogy hányas kérdést kell előhozni az adott pályán és a többit a program intézte. A json fájl szerkezetét úgy alakítottam ki hogy a későbbiekben bővíteni lehessen a filet. A json fájl szerkezete úgy épül fel hogy egy „gameData" nevű objectben(ami az adatbázisnak felelhetne meg) van a „kerdesobjektek amiknek van egy „id", egy kérdés, és négy válasz van. Ezekből tevődnek össze a kérdések. A c# kódomban viszonylag egyszerű dolgom volt mivel csak a fájlkezeléshez kellett importálnom a „System.IO"-t. A kódomat úgy építettem fel hogy a játék első lefutásánál lementődnek az adatok a telefon játék számára kijelölt helyén. Majd ha valamelyik pályán felveszi a játékos a könyvet, akkor kikeresi az adott pályához tartozó könyvet az id alapján, majd ezeket feltölti a könyv menüjébe, a gombok helyére véletlenszerű sorrendben. Ha a játékos jó válaszra nyomott, akkor az adott pályának megfelelő bónuszt szerez, ami valamilyen előnyhöz juttatja.

```
public void checkAnsver(int index)
{
    if(helyes == valaszok[index].text)
    {
        if(kerdesi == 1 && player.GetComponent<HealthConroller>().health != 3)
        {
            player.GetComponent<HealthConroller>().addHealth();
            Destroy(gameObject);
        }
        if(kerdesi == 2)
        {
            player.GetComponent<PlayerMovement>().rundpeed = 50;
            StartCoroutine("stopBonuses");
        }
```

4. ábra, a választ ellenőrző metódus

3.12.IMPLEMENTÁCIÓS TERV

A munkám már a szakdolgozat kiválasztása után nemsokkal elkezdődött, jöttek az ötletelések, miként tudnám kivitelezni a játékot. A játék fejlesztését, mint az előző pontokban leírtam sok

munkát vett igénybe, de ügyeltem rá, hogy ne egyszerre csináljak mindent. Részletekben haladtam, először csináltam egy alap vázlatot az adott részről, majd leprogramoztam, aztán megrajzoltam a hozzá tartozó elemeket. Ügyeltem rá, hogy jól legyen minden megrajzolva, és hogy minél kevesebb hibát kelljen a későbbiekben kijavítani, de persze elkerülhetetlen, hogy ne csússzon be egykét hiba.

3.13.UML ÁBRA

A szakdolgozatom UML ábráiból, csak a legfontosabb kódokról készítek, mivel 28db script van a játékban. Amint a képeken látható, szinte minden scriptben szerepel a *Start()* eljárás amely a Unity sajátossága. Ez az utasítás mindig akkor fut le amikor az adott scriptet létrehoztuk a játékban. Mindig csak egyszer fut le. Némely scriptben szerepel az Update() eljárás ami minden egyes képkockánként lefut, így tudunk egy-egy változást észlelni, vagy valamit változtatni.

3.14.KÓDOLÁS

A játékom elkészítéséhez rengeteg dolgoztam az Unity-ben, és itt szinte csak húzogatni és beállítgatni kellett dolgokat. Persze nem csak ebből állt a munkám, összesen 31 Scrip-et készítettem ,amikben leprogramoztam a játék funkcióit, ezekben intéztem el a json fájl mentését. A scriptek magukban nem tudnak sokat, hozzá kellett csatolnom őket egy-egy „Game Object”-hez ami a játékon belül lehet bármi, minden gameobjectnek van egy pozíciója, ez a minimum ezen kívül szinte bármit lehet rárakni. A játék fejlesztése során sokat tanultam YouTube videókból, de jobbára egy Google keresés megoldotta a problémámat. Mivel nagyon sok kód van ezért csak pár példát mutatok be milyen eljárásokat használtam. Az egyik legfontosabb szerintem a karakter életének növelése, és csökkentése.

```
public void takeDamage()
{
    PlayerPrefs.SetInt("hpLoss", PlayerPrefs.GetInt("hpLoss")+1);
    health--;
    if (health == 2) hearth3.GetComponent<Image>().sprite = EmptyHearth;
    else if (health == 1) hearth2.GetComponent<Image>().sprite = EmptyHearth;
    else if (health == 0) hearth1.GetComponent<Image>().sprite = EmptyHearth;
    if (health <= 0) death();
}
3 references
public void addHealth()
{
    if(health <= 0) death();
    if(health != 3) health++;
    PlayerPrefs.SetInt("hpGain", PlayerPrefs.GetInt("hpGain")+1);
    if(health == 3) hearth3.GetComponent<Image>().sprite = FullHearth;
    else if(health == 2) hearth2.GetComponent<Image>().sprite = FullHearth;
}
```

5. ábra, élet vesztés és szerzés metódus

A takeDamage() a nevéből érthetően egy életet vesz le a játékostól. Az első sorban megnövelem az összes levont életet eggyel, ez a statisztika. A másodig sorban vonom le az életet. A következő sorokban annak függvényében cserélem ki az egyik szív képet a felhasználói felület bal felső sarkában, hogy éppen mennyi az élete a játékosnak. Ha nulla akkor pedig meghívja a death() metódust. Az addHealth() eljárás hasonlóan működik, csak itt növeli az életet eggyel.

A másik példa kódrészlet, ami még egy fontos része a játéknak, az a különböző eldobott dolgok felvételére szolgál. Ebben a kódrészletben egy olyan metódus van, ami a nevéből adódóan akkor fut le amikor egy object hozzáér a játékoshoz, és ott marad, tehát folyamatosan egymás után amíg meg nem szűnik a kapcsolat a két object közt. A metóduson belül négy if található, ami lehet két fajta érme, egy, ami 5öt ad, és egy, ami 3-at. A másik 2 az élet és a kövek felvételét figyeli. A mintában látható, hogy a program megnézi, hogy az éppen ért object egy érme-e. Majd le ellenőrzi, hogy mi veszi fel, ebben az esetben a 0 a playert jelképezi. Majd le ellenőrzi, hogy éppen nem történik-e felvétel. Ha mindez igaz akkor ezután a program kitörli a felvett objectet, beállítja a colliding-ot igazra, azaz hogy éppen valaminek a felvétele történik. Majd lefut egy Wait metódus amiben a collidin-ot 0.01mp után hamisra állítja, így időt ad arra hogy lefusson a többi kód, és ne akadozzon a játék ha egyszerre sok érmét kell felvenni. Ezután hozzáad az eddigi érmékhez és ezt kiírja a megfelelő helyre.

```
void OnTriggerStay2D(Collider2D col)
{
    if(col.gameObject.CompareTag("Coin") && gameObject.layer == 0 && colliding == false)
    {
        Destroy(col.transform.parent.gameObject);
        colliding = true;
        StartCoroutine("wait");
        currentCoins += 5;
        coinText.text = currentCoins.ToString("000");
    }
```

6. ábra, érmék felvételéért felelős metódus

3.15.TESZTTERV

A játék tesztelése során rengeteg hibára derült fény, de a végére sikerült mindent kijavítanom. A teszteléseket csak akkor végeztem el, ha egy adott programrész késznek találtam és hiba nélkül lefutott. Így, hogy viszonylag működött rengeteget tudtam fejleszteni a kódokon, és a játékmeneten. A programom először csak én teszteltem, de miután odaadtam próbára egyből találtam legalább három új hibát. Az elején az irányítással voltak gondok, ha már nem nyomott a felhasználó egy mozgás gombot akkor is mozgott a karakter. Ezt egy event rendszer segítségével sikerült kiküszöbölnem, ami után nem is volt vele probléma. Ami még rengeteg tesztelés után derült ki az, hogy a karakter fel tud akadni a pálya tetejére, ezt úgy tudtam megoldani, hogy egy kicsit nagyobb csúszós érintkezőt raktam a karakter felső részéhez, így lecsúszott nem pedig odaragadt. A másik probléma, ami többször is előjött az a pálya határoló collider-ek (ütközők) voltak, amiket gyakran át kellett helyezni, vagy megnövelni mert a játékos át tudta ugrani. A későbbiekben ahogy nőtt a játék, és létrehoztam a dobás képességet, az eldobott kő egy helyett, kettőt sebzett, amit később, mint kiderült két collider volt az eldobott kövön így kétszer érintkezett az ellenséggel. Egy másik hiba, ami nagyon sokáig a játékban volt, de csak később oldottam meg, az az érmék felvétele volt. Először, ha a játékos ráugrott egy érmére akár három érmét is felvehetett az egy helyett. Másodszor is ha a játékos ráugrott egyszerre több érmére, úgy ugrott vissza fel a játékos mintha trambulinra lépett volna. Az első problémát úgy tudtam megoldani hogy egy késleltetést rakok az érmék felvételéhez. A második probléma komplikáltabb volt mivel át kellett rendeztem az érmék ütközését a pályával. Egy adott érmén így volt egy collider ami a földel érintkezett hogy ne essen ki a pályáról, a másik pedig kicsit nagyobb volt, de “átlátszó”, ez biztosította az érme felvételét a játékos számára, az érme script-jébe még bele kellett írnom azt hogy az érme kemény collidere, és a karakter collidere ignorálják egymást, így nem tudott létrejönni az a trambulin hatás.

4.ÖSSZEGZÉS

4.1.TESZTELÉSEK

A játékom fejlesztése során próbáltam minél jobban letesztelni, hogy rájöjjek a hibákra és kijavítsam őket. A fejlesztés során több fő tesztelési fázis volt. Az elsőben a mozgást kellett letesztelnem, amit folyamatosan teszteltem a játék fejlesztése során ha akartam ha nem, és így a végére, hibátlanul működik. A második tesztelési fázis a karakter támadásáról szólt. Itt is sok hibába ütköztem, de a sikerült kijavítanom. A fejlesztések során ahogy sikerült lebuildelni egy verziót telefonra, mindig végig teszteltem a funkciókat a játék egészén. A következő tesztelési fázis az ellenségek letesztelése volt. Itt teszteltem le a mozgásukat, és a támadásaikat. Ezután jöttek az érmék tesztelése. Itt teszteltem le hogy működik-e az érmék felvétele, és annyit vesz fel a karakter amennyit kell. A következő tesztelést az első főellenségen végeztem. Leteszteltem, hogy mennyire nehéz, és hogy megfelelően működik. A következő nagy tesztelés fázis a könyvekről szólt. Leteszteltem, hogy betöltődnek-e a kérdések, és a válaszok véletlenszerűen töltődnek be a fájlból. Ezután következett a helyes válaszok által szerzett képességek tesztelése. Mivel nagyon különböző bónuszokat kap a játékos ezért ezeknek a funkcióknak a tesztelése kicsit tovább tartott, mint a többi. A játék befejezése felé észrevettem, hogy az én 5éves telefonomon egészen máshogy néz ki mint az osztálytársam féléves teljes kijelzős telefonján. Ezért tesztelhettem újra teljes játékot az új kijelzőn, hogy minden látszik, és semmi se lóg ki a képből.

4.2.HIBÁK

A fejlesztés során sok hibába ütköztem. Valamelyik kisebb valamelyik nagyobb mértékben befolyásolja a játékmenetet. A legtöbb hibát sikerült pár perc alatt kijavítanom, de voltak bőven olyanok is, amik több órát, sőt több napot vették igénybe. A hibák nagyrésze a saját gondatlanságomból keletkezett, de volt olyan is amit a Unity játékmotor miatt kellett kijavítanom. Az első nagyobb hiba, amit ki kellett javítanom, az a támadás volt. Mivel még nem nagyon készítettem ilyet, ezért ez egy több napos problémának bizonyult. A probléma az volt, hogy amikor a játékos támadott, akkor fel tudta venni a karaktertől távoli érméket is. A problémán sokat törtem a fejem míg rájöttem a megoldásra. Annyi volt a dolgom hogy átállítsam a támadás által létrehozott ütközőt úgy hogy ne player legyen a tag-ja, hanem kard és így, már nem vette fel az érméket csak ha a karakter ért hozzájuk.

4.3.TAPASZTALATOK

A feladat során sok akadályba ütköztem. A szakdolgozatom ösztönzött, hogy kitartó legyek a végéig, és hogy ne hibázzak benne, vagy legalábbis javítsam ki a hibáim. A játékomat a

későbbiekben szeretném tovább fejleszteni. A legtöbb tapasztalat magából a játék fejlesztéséből jött. De sikerült egy kicsi tapasztalatot szereznem a pixel-es rajzolásban is mivel a játékban mindent magamnak kellett megrajzolnom. A rajzolásban segítségemre szolgáltak YouTube videók, ahol elmagyarázták az alapokat a pixel art stílushoz. A játék fejlesztése során rengeteg tapasztalatot szereztem a C# nyelven való programozásban is, mivel a játékban 31 kód fájl van, és mindegyik script egy-egy fontos funkciót lát el a játék során. A program kódok egyrésze a karakterért volt felelős, mint például a mozgás, támadás, vagy a dobott érmék, és szívek felvételéért volt felelős. Egy másik script csoport a json fájlból való beolvasásért volt felelős, ami beolvasta a kérdést és a válaszokat, majd ezeket beillesztette a megfelelő gomb alá véletlen sorrendben. Az így keletkezett kérdéseket, pedig kezeli, hogy ha eltalálta a játékos, akkor a megfelelő bónuszt adja a játékosnak. Sok tapasztalatot szereztem a json fájlok tárolása felépítése során, figyelembe kellett vennem, hogy hova rakja a program a jsont, hogy ha esetleg a felhasználó törli az alkalmazáshoz tartozó adatokat ne kelljen újra telepíteni a játékot, hanem újra létrehozza a fájlt. A játék Unitys részét tekintve is tanultam. A játékot a Unity Engine segítségével raktam össze, a fejlesztés során sokat tanultam, hogy hogy kell összerakni egy komplexebb játékelemeket mint például a karaktert, ellenségeket, és az érméket. Sok scriptet használtam még a menü rendszer elkészítéséhez, mivel úgy kellett megoldanom, hogy csak az eddig feloldott pályákat lássa a felhasználó.

5.ÁBRAJEGYZÉK

6.FORRÁSOK

Minden, ami előre segített a szakdolgozatom elkészülésében:

A játékban lévő betűtípust Kővágó Dávid készítette.

Segédletek a Unityben való fejlesztéshez, programozáshoz:

A Unity dokumentáció:

https://docs.unity3d.com/Manual/index.html

A tileset-et leíró dokumentáció:

https://docs.unity3d.com/Manual/class-Grid.html

A getcomponent metódushoz tartozó oktatóvideó:

https://unity3d.com/learn/tutorials/topics/scripting/getcomponent?playlist=17117

A static típusok unityben:

https://learn.unity.com/tutorial/c-static-types

Az Awake és Start metódusok:

https://unity3d.com/learn/tutorials/topics/scripting/awake-and-start?playlist=17117

Az Update és a FixedUpdate metódusok:

https://unity3d.com/learn/tutorials/topics/scripting/update-and-fixedupdate?playlist=17117

Továbbá a tesztelésben segített Kiss Gábor.