Budapesti Műszaki Szakképzési Centrum
Verebély László Szakgimnáziuma és Szakközépiskolája

SZAKDOLGOZAT

2D Platformer játék

Kerekes Adrienne

Konzulens
Juhász Zoltán

Szoftverfejlesztő képzés
OKJ 54 213 05

2020. 04. 19.

NYILATKOZAT A SZAKDOLGOZAT EREDETISÉGÉRŐL

Alulírott Kerekes Adrienne 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 2D Platformer játék

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 10.

Kerekes Adrienne

Hallgató aláírása

Tartalomjegyzék

1. Bevezetés

Számítógépes játék. Ha ez a kifejezés egy mai beszélgetésben elhangzik, a többségnek általában hasonló dolgok jutnak eszébe akár fiatal, akár idősebb személyről van szó. Szórakozás, kikapcsolódás, nevetés, izgalom, csapatjáték, kihívás és lehetne folytatni. Manapság szinte mindenki találkozik valamilyen formában számítógépes vagy egyéb platformon működő játékokkal, illetve játszik is velük. Gondoljunk itt csak a telefonos játékokra. Még régebbi, nyomógombos telefonokon is volt legalább egy játék és biztos, hogy legalább egyszer mindenki kipróbálta őket – még akkor is, ha véletlenül nyitotta meg.

Manapság nem csak szórakoztatásra készítenek játékokat, hanem tanításra, fejlesztésre, illetve terápiás célokra is. A tanító, fejlesztő szoftverek különösebb megerőltetés nélkül, játszva tanítják meg a felhasználót az adott tudásra, ami pozitív élményekkel köti össze a tanulás folyamatát. Ebből következik, hogy a játékos szívesebben és könnyebben sajátítja el a megtanulandó anyagot, készségeket. A terápiás játékok egy része virtuális valóság segítségével segíthetnek traumák, fóbiák, stresszbetegségek kezelésében is. Azáltal, hogy lemodelleznek egy magas erkélyt vagy egy zsúfolt metrót és ezekkel a körülményekkel szembesítik a beteget, segíthetnek a páciensnek enyhíteni vagy akár leküzdeni az adott félelmüket, betegségüket. Mindezt persze szigorú orvosi felügyelet mellett.

Nem csak ezek a speciális játék szoftverek segíthetik vagy fejleszthetik a játékost. Egy egyszerűbb játék is képes additív hatást gyakorolni. Javíthatják a koncentrációs készséget, a reakció időt, a problémamegoldó készséget. A kooperatív vagy többjátékos játékokból sokat tanulhatunk a másik féltől, megismerhetjük a társunk vagy társaink nézőpontját. Kapcsolati nehézségekre, vagy egyéb hiányosságokra is rámutathat egy-egy a játékban kialakult helyzet.

A játékipar mai kínálatában kortól és érdeklődési körtől függetlenül mindenki találhat magának megfelelő játékot. Mindenkit más dolog fog meg egy játékban. Van, aki a kihívás miatt játszik az adott játékkal. Van, akit inkább a történet vagy a gyönyörű képi és hangi világ ragad meg. Olyan is akad, aki a játékokba menekül a hétköznap problémái elől, ami sokszor függősébe torkollik, további problémákat eredményezve. Nagyon fontos megállapítani, hogy mennyi az az idő, amit játékra szánhatunk.

Szakdolgozatom témája egy általam elkészített számítógépes játék. Évek óta játszom játékokkal és mindig is érdekelt a folyamat, ahogy létrejöttek kedvenceim. Természetes volt számomra, hogy a szakdolgozatom témája számítógépes játék lesz, és idő múltával én is szeretnék bekerülni ebbe az iparba, mint játékfejlesztő.

2. Fejlesztői dokumentáció

2.1 Feladat specifikáció

A jelenlegi tudásszintemen képes voltam egy kétdimenziós platformer[1] játékot csinálni, ezért esett erre a műfajra a választásom. A játék grafikájának alapjául szolgáló képeket - amikből később az animációkat összeállítottam -, vagyis „sprite sheeteket” ingyenesen töltöttem le az internetről. Erre egy későbbi fejezetben bővebben kitérek. Miután a játék felépítése ”lentről felfelé” halad a ”balról jobbra” helyett, ezért a harcrendszer közelharcos lett. Saját menürendszert is kapott a játék.

2.1.1 A játék főbb funkciói

Menü funkciók: főmenü és a játék szüneteltetésének menüje, ezeken belüli navigálás.
Karakter irányítása egyszerű módon.
Rövid, akadályokkal, ellenfelekkel nehezített pályán való végig haladás.
Az ellenségek kikerülése vagy megsemmisítése.
A játék végén egy egyszerű összesítő megjelenítése.

2.2 Fejlesztői ütemterv

Az ütemterv nyolc nagyobb lépésből állt. Mindegyik lépésnek része volt a konzulenssel való egyeztetés. A lépések a következők:

- téma, stílus, és fejlesztői eszközök kiválasztása,
- kép- és hanganyag gyűjtése,
- a karakter és az ellenségek megtervezése, animálása, kódolása,
- platformok és játékelemek megtervezése, kódolása,
- GUI[2] és a menük megtervezése, összeállítása, kódolása,
- pálya struktúra megtervezése, megvalósítása,
- tesztelés, finomhangolás,
- dokumentáció megírása.

[1] videójáték-műfaj, lényege: elemeken való ugrálással túl kell jutni az akadályokon a célig
[2] Graphical User Interface, azaz grafikus felhasználói felület

2.2.1 A játék megtervezésének lépései

Első lépésnek meg kellett határozni a szoftver témáját. Több lehetőség is felvetődött, végül egy kétdimenziós, ügyességi, közelharci rendszerű rövid, egyszerű játékra esett a választás. Ezután jöhetett a fejlesztői eszközök kiválasztása. Nagyon fontos a megfelelő IDE[3] kiválasztása minden feladat típushoz a feladatnak és a fejlesztő képességeinek megfelelően. A játék megalkotásához a Unity nevű játékmotort használtam, míg a kód megírását a Microsoft Visual Studio 2019 fejlesztői környezetében végeztem. A választásom indoklását egy későbbi fejezetben ismertetem.

Második lépésben a már körvonalazódó témához kellett keresni hozzá illő kép- illetve hanganyagot. Kutatás közben egyre tisztább és egyértelműbb lett az elképzelés, hogyan fog kinézni a játék. Mivel eldöntésre került az előző lépésben, hogy kétdimenziós és közelharci lesz, ezért ennek megfelelően, célirányosan kutattam.

A harmadik szakasz több időt ölelt fel, mint bármelyik másik rész. Első lépésnek össze kellett állítani az animációkat, és a közöttük lévő kapcsolatokat, átmeneteket, illetve az ezekhez tartozó változókat, amik a folyamatokat irányították. Csak ezt követően lehetett megírni a szkripteket[4], amik az animációkhoz rendelt változók által befolyásolták az elemek mozgását, reakcióját, irányíthatóságát. Egészen a projekt befejezéséig akadtak kisebb nagyobb nehézségek, ”bugok[5]”, amiket folyamatosan javítani kellett.

A negyedik lépés viszonylag kevés időt igényelt, mivel a platformokat, elemeket nem kellett meganimálni, és a szkriptjük is pár soros.

Az ötödik szakasz A Unity egyik funkciójának köszönhetően rövid volt. A menük egyes elemeihez nem kellett szkripteket írjak, mert volt erre előre megírt és beépített funkció.

A pálya felépítése és formája szintén eldöntésre került még az első szakaszban. Különös figyelem lett fordítva a pálya sokszínűségére, minél több fajta játékelem, és ezek különböző kombinációinak létrehozásával, csökkentve lett a játék kinézetének monotonitása. Számításba kellett venni a karakter és az ellenfelek tulajdonságait.

A tesztelési időszak már a harmadik lépéstől elkezdődött. Ebben a szakaszban már voltak működő elemek, ezeket tesztelni, és hiba esetén javítani kellett.

A dokumentáció írása a projekt elején megkezdődött és párhuzamosan haladt a fejlesztéssel.

[3] Integrated Development Environment, azaz integrált fejlesztői környezet
[4] elemi utasítások sorozata
[5] hibák, rendellenes működések

2.2.2 Határidők

A projekt szakaszainak tervezett és tényleges időintervallumait az *1. számú melléklet***: Gantt-diagram, saját készítésű ábra** szemlélteti. Az ábrán látható egy táblázat és mellette egy egyszerűsített Gantt-diagram. Minden tevékenységhez hozzá vannak rendelve a következő jellemzők: terv kezdete, terv időtartama, tényleges kezdés, tényleges időtartam. Voltak olyan szakaszok, ahol a tervezett és tényleges adatok különböznek, hol kedvezően, hol kedvezőtlenül. A táblázat mellett grafikusan is ábrázolva lettek a tényleges adatok. Ez az ábra jól és átláthatóan mutatja be a projekt feladatainak beosztását hónapokra leosztva.
Decemberben kezdtem neki a munkálatoknak, azonban nem tudtam minden nap foglalkozni a projekttel. A szakdolgozat leadásának határideje 2020.04.19. volt. Az utolsó napig tesztelések és finomhangolások zajlottak a játékon.

2.3 Követelmény specifikáció

A tervezési fázisban elengedhetetlen volt meghatározni a szoftver főbb jellemzőit, és megalkotásának körülményeit. Próbáltam egyedi kinézetet, mechanikát és pálya felépítést választani. Ezt a jellemzőt a tervezés és megalkotás minden fázisában figyelembe kellett venni.

2.3.1 Fejlesztés célja

Az általam írt szoftver célja a szórakoztatás. Kortól függetlenül bárkinek rövid, kellemes időtöltést nyújtása. Évek óta játszom játékokkal és érdekel a játékfejlesztés, illetve a későbbiekben szeretnék ebben az iparban elhelyezkedni. Ezért gondoltam, hogy egy ehhez kapcsolódó témában készítem el a szakdolgozatom.

2.3.2 Fejlesztői környezet és eszközök, alkalmazott technológiák

Két fejlesztői környezet segítségével alkottam meg a programot, Microsoft Visual Studio 2019 és Unity. Előbbi programban a szkripteket írtam, az utóbbiban magát a játékot készítettem el, azaz a grafikát az interakciókat, a karaktert, az ellenfeleket, a pályát és a menüt. Az általam választott programozási nyelv a C# lett. A C, C++, illetve Javascripten kívül ezt is támogatja a Unity, illetve ez az a nyelv, amit majdnem másfél éven át tanultam a szakdolgozat elkezdése előtt.
Azért döntöttem a Visual Studio mellett, mert egy sokoldalú, sokak által használt, népszerű IDE, amihez sok beépülő modul és kiegészítő van, valamint ebben a környezetben tanultam a kódolás alapjait. A program hibakezelése, ami megmutatta, mint a szintaktikai, mint a logikai hibáimat nagyban elősegítette a kódolás menetét.

A Unity 2018-as verziójában[6] dolgoztam, mert ez egy LTS[7], stabil, hibamentes kiadása volt a fejlesztői környezetnek. Ez fontos a projekt szempontjából, mert olyan hibák fordulhatnak elő egy nem stabil verzióban, ami nagyban visszavetheti a fejlesztés menetét.

2.3.3 Futtatási környezet

A szoftver Windows rendszerre lett készítve, amelynek specifikációi a következők:

- processzor: 1 gigahertz-es (GHz) vagy gyorsabb processzor vagy SoC (rendszerchip),
- RAM: 1 gigabájt (GB) a 32 bites rendszerhez és 2 GB a 64 bites rendszerhez,
- merevlemez-terület: 16 GB a 32 bites és 32 GB a 64 bites operációs rendszerhez,
- videokártya: DirectX 9-kompatibilis vagy újabb, WDDM 1.0 típusú illesztőprogrammal.

A Unity lehetővé teszi, hogy OSX vagy Linux alapú rendszerekre is le lehessen fordítani a programot.

Jelen szoftver futtatásához nincs szükség kiegészítő program telepítésére.

2.3.4 Felhasználói felület

A felhasználói felület megalkotásakor fontos szempont volt az egyszerűség, átláthatóság. A menürendszert - kép- és hangvilágban egyaránt- hozzá kellett igazítani a játék kinézetéhez és hangulatához.

1. ábra: A főmenü képernyőképe

[6] Verziószám: 2018.4.14f1 (64bit)

[7] Long Term Support, azaz hosszú távú támogatás

A játékos nyomógombok segítségével – bal egérgomb kattintásával - tud navigálni a menüben, a játékban pedig billentyűk lenyomásával, ami a főmenüben, az irányítás menüponton belül fel van tüntetve.

2. ábra: A beállítások képernyőképe

A játék elindulásakor a játékost a főmenü fogadja, négy menüponttal: játék, beállítások, irányítás, illetve kilépés.

A játék menügombra való kattintáskor azonnal elindul a játék.

A második menüpont esetében megjelennek a játék lehetséges beállításai: felbontás, grafika, hang. A felbontás lenyíló menüjében, az éppen használt monitor felbontásának összes variációja megtalálható és beállítható. Az érték alapértelmezetten 1920 x 1080[8]. A grafika lenyíló menüjében a következő beállítási lehetőségek állnak rendelkezésre: nagyon alacsony, alacsony, közepes, magas, nagyon magas, ultra. Ennek a funkciónak az alapértelmezése a nagyon alacsony. A hang erőssége egy csúszkán állítható. Ez a program indulásakor a maximumon van. A vissza gombbal a főmenübe lehet navigálni.

Az irányítás menügombra való kattintással megjelennek a karakter irányításával kapcsolatos információk. A billentyűzeten való ”D” gomb lenyomásával jobbra, vagyis előre indul el a karakter, az ”A” gomb megnyomásával pedig balra, azaz hátra. A ”Szóköz” gomb leütésével képes ugrani. Ahhoz, hogy támadni tudjon a játékos által irányított karakter, a bal egérgombbal kell kattintani.

[8] 1920 képpont széles és 1080 képpont magas

3. ábra: Az irányítás menüpont képernyőképe

A főmenüben a kilépés gombbal a szoftver bezáródik.

Ha a játékot szüneteltetni akarjuk, akkor az ”Esc” billentyűt kell lenyomni a billentyűzeten. Ekkor megjelenik a szünet menü.

4. ábra: A játék szüneteltetés menüjének képernyőképe

A játék szüneteltetésének a menüje a következő menüpontokból áll: folytatás, irányítás, főmenü. Amíg ez a képernyő látható, addig a játék meg van állítva. A folytatás gombra kattintva visszatérünk a játékhoz és a felhasználó tovább játszhat. Az irányítás menüben ugyanaz a leírás található, mint a főmenüben. Ha a játékos bizonytalan az irányítást illetően, akkor bármikor tájékozódhat anélkül, hogy ki kellene lépnie a játékból. A főmenü gomb megnyomásával a felhasználó visszatér a főmenübe, viszont az eddig elért eredménye és állapota elveszik. Az ”Esc” billentyű újbóli lenyomásával is folytathatja a játékot a játékos.

Felületnek tekinthető még maga a játék, a karakter állapotáról információt nyújtó elem, a győzelemkor, illetve a karakter halálakor megjelenő képernyő.

A karakter állapotáról egy egyenes szolgáltat információt a játék bal felső sarkában. Azt jelzi mennyi élete van a karakternek. A játék kezdetekor ez az érték maximumon van.

5. ábra: A karakter életét mutató elem

Ha eléri a 0-át, akkor a karakter meghal, és a játéknak vége.

Ekkor megjelenik egy képernyő, ami tájékoztatja a játékost arról, hogy mi történt és lehetőséget nyújt a játék újrakezdésére, vagy a főmenübe való visszatérésre gombok által. Az új játék feliratú gombbal azonnal az elejétől elindul a játék, a menü gombbal visszatérhet a felhasználó a főmenübe.

6. ábra: A játék elvesztésekor megjelenő képernyő

Ha a játékos elér a pálya végére, megtalálja a kincses ládát és odamegy hozzá az irányított karakterrel, akkor megnyerte a játékot. Egy képernyő informálja erről a játékost, felkínálja egy új játék kezdésének a lehetőségét, vagy a főmenübe való ugrást. Továbbá kiírja a pályán megölt ellenségek számát.

7. ábra: A játék megnyerésekor megjelenő képernyő

2.3.5 Minőségi követelmény

A rengeteg tesztelésnek és finomhangolásnak köszönhetően a program hiba nélkül futtatható. A meghatározott feladatait teljeskörűen ellátja. Párhuzamosan működtethető több programmal egyszerre. A program egyszerűségéből adódóan nincs nagy rendszerkövetelménye.

2.3.6 Terjesztés módja

A program terjesztése több módon és egyszerűen történhet. Ajánlott betömöríteni, utána interneten, illetve bármilyen adathordozó eszközön feltétel nélkül terjeszthető. A játék csak Windows operációs rendszeren működik.

2.4 Rendszer specifikáció

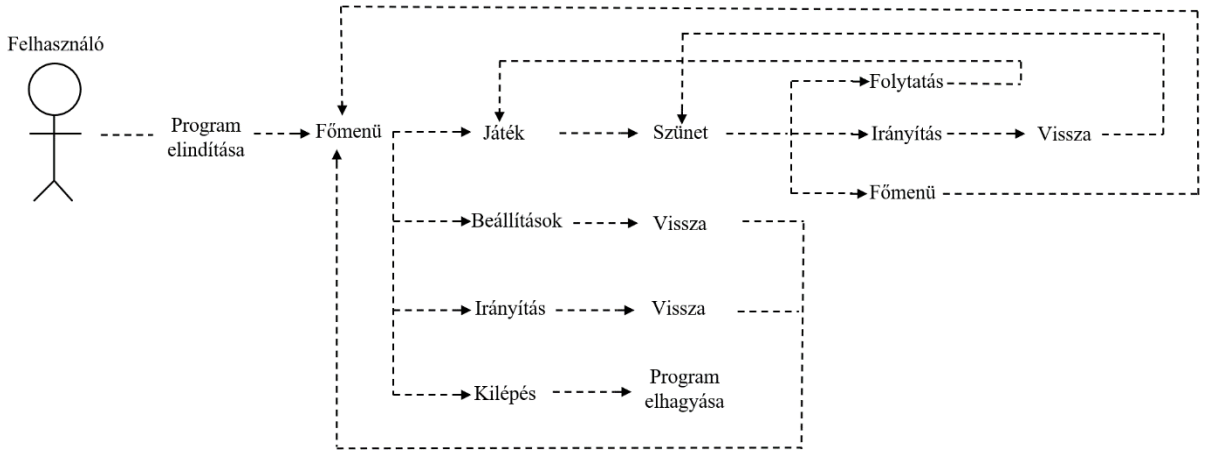
A rendszer megalkotásának fő szempontja az egyszerűség mellett a bővíthetőség volt. A játékelemek úgy lettek meghatározva és felépítve, hogy bármikor hozzá lehessen adni valamilyen funkciót, vagy lehetőséget. Erről bővebben egy későbbi fejezetben ejtek szót.

2.4.1 Rendszer használói

Az aktorok olyan személyek vagy dolgok összessége, melyek ugyan nem képezik szerves részét a rendszernek, de hatással vannak rá. Ebben az esetben csak egy tényező van, ami hatással van a szoftverre. Az elsődleges aktor a szoftver használója, azaz a játékos.

2.4.2 Rendszerhasználati esetek

A rendszerhasználati eseteknél a következő szempontoknak kell eleget tennie a programnak: a felhasználó probléma nélkül kapcsolatba tud lépni a szoftverrel, komplikációk nélkül használni tudja azt, és a folyamat sikerrel befejeződik.



8. ábra: HE-diagram

2.4.3 Menü-hierarchia

A játékom két ”Scene[9]”-ből áll: a főmenü és maga a játék. A főmenü felépítése 4 fő gombból áll: játék, beállítások, irányítás és kilépés. A játék gomb megnyomásakor elindul a játék, a kilépésre kattintva pedig a program bezáródik.

[9] helyszín

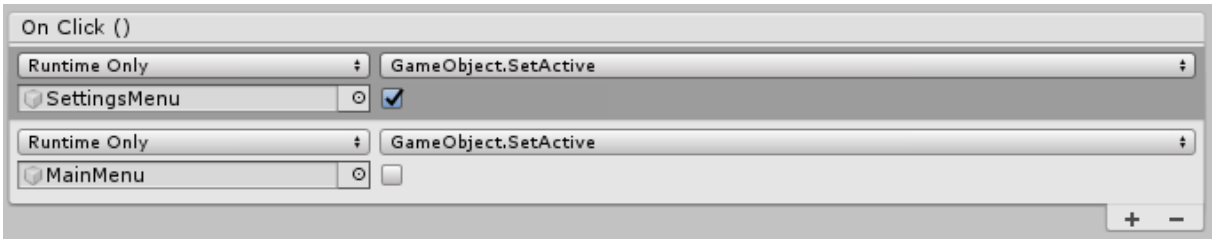
Ennek a két funkciónak a kódját egy scriptbe írtam bele a következőképpen:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

0 references
public class MainMenu : MonoBehaviour
{
    0 references
    public void PlayGame()
    {
        SceneManager.LoadScene("GameScene");
    }

    0 references
    public void QuitGame()
    {
        Application.Quit();
    }
}
```

10. ábra: A főmenü játék és kilépés gombjának kódja

A PlayGame() metódus indítja el a játékot, A QuitGame() metódus pedig bezárja a programot. E két metóduson megírásához feltétlenül ki kellett egészítenem a használt névtereket a következővel: UnityEngine.SceneManagement. Ezáltal elérhetővé vált a scene-ek kezelése kódon keresztül.

A főmenüben beállítások gombra kattintva megjelenik egy felület, amin keresztül állíthatóak a játék tulajdonságai. A Unity beépített lehetőségeinek köszönhetően könnyen lehet a gombok funkcióihoz metódusokat rendelni. Vannak előre beépített funkciók, azonban az általam megírtakat is használhatom. A *10. ábrán* látható, hogy egy már előre megírt funkciót használtam. Ez a része a Unity-nek megkönnyítette és felgyorsította a munkálatok egy részét.



9. ábra: Gombok funkcióinak beállítása

A beállítások menüponton belül változtatható a felbontás tulajdonsága, a játék grafikájának minősége, illetve a hang erőssége. A kód megírásakor itt is ki kellett egészítenem a névtereket a következőkkel: UnityEngine.Audio, UnityEngine.UI.

A UnityEngine.Audio névtér hozzáférést biztosított a játékon belüli hangok irányításához. A hang erősségét a SetVolume() metódus teszi állíthatóvá. A UnityEngine.UI segítségével kódon keresztül el tudom érni az összes grafikus elemet, amit a beállítások menü felületén helyeztem el.

A képernyő felbontását a SetResolution() metódus, a játék grafikájának minőségét a SetQuality() nevű metódus állítja be. A SetFullScreen() metódus változtatja a játék megjelenítésének módját, ami lehet teljes képernyős vagy ablakos.

```
0 references
public void SetVolume(float volume)
{
    audioMixer.SetFloat("volume", volume);
}

0 references
public void SetQuality(int qualityIndex)
{
    QualitySettings.SetQualityLevel(qualityIndex);
}

0 references
public void SetFullScreen(bool isFullScreen)
{
    Screen.fullScreen = isFullScreen;
}

0 references
public void SetResolution(int resolutionIndex)
{
    Resolution resolution = resolutions[resolutionIndex];
    Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);
}
```

11. ábra: A beállítások menü kódrészlete

Az irányítás gombra kattintva megjelenik egy tájékoztató képernyő, ami a játékban lévő karakter irányításáról, illetve a játék egyéb funkcióiról nyújt információt. Ehhez a részhez nem kellett szkriptet írjak, a Unity-n belül lett megalkotva.

2.5 Fizikai környezet

A játék Windows operációs rendszerre lett fejlesztve, illetve nincs nagy hardver igénye.

2.5.1 Külső rendszerek

A szoftver nem használ külső rendszereket, viszont bármikor hozzá lehetne adni. A szoftver továbbfejlesztésének lehetőségei egy későbbi fejezetben lettek kifejtve.

2.5.2 Képernyőtervek

A tervezés fázisában nem volt meghatározott számú képernyőterv. A menük megtervezése és létrehozása a projekt második felében történt, ebből adódóan a fejlesztés előrehaladtával bővült a tervezet ezen része.

Ahogy korábban leírtam, nagyon fontos volt a menük, a játékelemek, a vizuális- és hangeffektek stílusának összehangolása. Ez főként a tervezés és anyaggyűjtés fázisában játszott nagy szerepet, ahol eldöntésre került, milyen elemek legyenek felhasználva a játékban.

2.6 Architektúrális-terv

A játék Windows x64 architektúrájú operációs rendszerre lett tervezve és megírva.
A szoftver jelen esetben nem használ külső rendszereket, ezért nem áll fent annak a veszélye, hogy nem megfelelő a kommunikáció a külön álló elemek között. Mivel az elemek egy rendszerben helyezkednek el, ezért a kommunikáció gyors és akadálymentes közöttük, publikus változók, metódusok, illetve funkciók által.
Minden szkripttel rendelkező játékelemhez a kód egyedileg van hozzárendelve, ezzel biztosítva a könnyebb továbbfejlesztését és bővítését a szoftvernek.
A játéknak egyszerre csak egy felhasználója lehet, ezáltal nem kellett a rendszer túlterhelődésének eshetőségét vizsgálni.

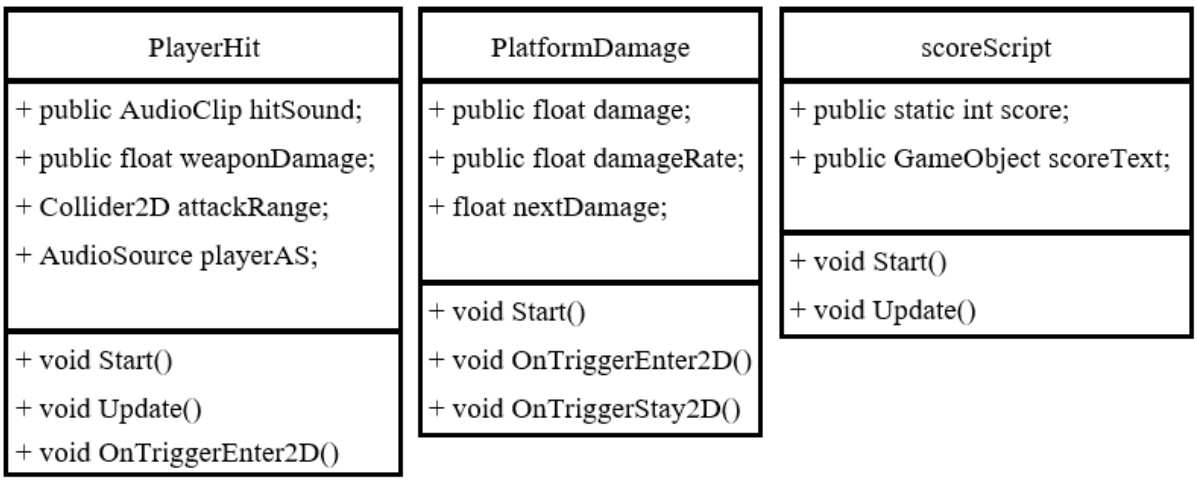
2.7 Implementációs terv

Miután a játékelemek mindegyike megalkotásra és külön-külön ellenőrzésre került, elkezdődhetett az elemek egymáshoz való hangolása és tesztelése. Először kevesebb elemet a könnyebb hibaelhárítás érdekében, majd egyre többet. Ezek után került összerakásra a pálya egésze, és kezdődött el a széles körű tesztelés. Nagyon fontos volt az elemekhez tartozó értékek megfelelő beállítása, ugyanis ez nagyban befolyásolhatja a játékélményt.

2.8 UML ábra

A tervezési fázishoz hozzátartozott az UML[10] ábrák elkészítése. Az UML szabványos, egymásra épülő és egymással kölcsönhatásban lévő diagrammok és modellezési eszközök összessége. *„Eredetileg Unified Method-nak hívták, ezt a nevet azonban félrevezetőnek találták, hiszen ez nem egy módszertan, hanem egy eszközrendszer, így átnevezték Unified Modeling Language-re. Ahhoz, hogy módszertanként szerepelhessen szükség lenne egy eszközrendszerre, illetve jelölésre és egy technikára. Technikát azonban az UML nem szolgáltat, teljesen módszersemleges.”*[11]
A tervezési időszakban elkészítettem azon kódok UML ábráit, amiben biztos voltam, hogy kelleni fog.

[10] Unified Modeling Language
[11] Tarczali Tünde: UML diagrammok a gyakorlatban 12.oldal



12. ábra: UML-diagramok

Balról jobbra haladva az első ábra a karakter sebzés kódja, a második a platform sebzés kódja, az utolsó a megölt ellenségek számlálójának a kódja. Egy ábra felépítése fentről lefele értelmezve: az első sorban a kód neve található, a második sorban a változók nevei, a harmadik sorban a metódusok nevei.

Az ábrákon látható metódusok magyarázata: a Start() metódus úgy viselkedik, mint egy konstruktor, itt történik az értékek inicializálása. Az Update() metódusban lévő kód annyiszor fut le, amennyi a monitor frame rate-je. Első sorban a grafikai elemek kódrészletei szerepelnek itt. Az OnTriggerEnter2D egy olyan metódus, ami akkor fut le, ha két megfelelő collider érintkezik. Colliderek segítségével teremtettem interakciót bizonyos játékelemek között. Az OnTriggerStay2D pedig akkor fut le és addig, amíg a két meghatározott collider folyamatosan egymásba ér.

2.9 Kódolás

A játékon belül 16 db különálló szkriptet írtam meg C# nyelven a különböző játékelemekhez: cameraFollow.cs, cleaner.cs, destroyMe.cs, enemyDamage.cs enemyHealth.cs, enemyMovement.cs, GUI.cs, healthPickUp.cs MainMenu.cs, platformDamage.cs, playerController.cs playerHealth,cs, playerHit.cs, scoreScript.cs, settingsMenu.cs, winGame.cs.

A cameraFollow szkript a Unity kamera eleméhez van hozzárendelve. Ennek segítségével állítottam be, hogy a kamera hogyan kövesse a karaktert a pályán. A cleaner kódban írtam le, hogy mi történjen a játékossal vagy az ellenségekkel, ha leesnek a pályáról. Amelyik játékelemhez hozzá van rendelve a destroyMe szkript az egy előre meghatározott idő után eltűnik a pályáról.

Az ellenség sebzését az enemyDamage kóddal, az életét az enemyHelath kóddal, a mozgását pedig az enemyMovement kóddal alkottam meg. A GUI nevű szkripttel vezéreltem a szünet menü gombjait. A healthPickUp kód vezérli, hogy mennyi életet nyerjen vissza az ellenség, akárhányszor felvesz egy szívecskét a pályáról. A MainMenu.cs irányítja a főmenüt. A platformDamage szkriptben határoztam meg, mennyit sebezzenek a szúrós platformok. A playerController, a playerHealth, és a playerHit kódok rendre a játékos irányítását, életét és a sebzését irányítják. A scoreScript a megölt ellenségek számát számolja, a settingsMenu kód a beállítások menüt irányítja, végül a winGame szkriptben van meghatározva, mikor nyer a játékos.

Az interneten és a Unity oldalán megtalálható oktató videókból ismertem meg a játékfejlesztés alapjait, illetve ötleteket, módszereket, praktikákat sajátítottam el. Miután a programozási nyelv használatában már rendelkeztem némi tapasztalattal, ezért a játék megalkotásának alapjait könnyebb volt megérteni.

Két nagyobb hiba volt, ami visszavetette a munkámat. Az ellenség megtervezése során írt szkriptet sajnos a későbbiekben teljesen újra kellett írnom, mert a módszer, amit használtam nem volt megfelelő a játék követelményeinek, ezáltal nem működött hibamentesen. Ezt a fennakadást viszonylag kevés idő alatt el lehetett hárítani. A másik hiba a karakter összeállításának alapjainál kezdődött és csak a projekt végén sikerült megoldani. Jóval kisebb változtatásokkal lett megoldva, mint az előbbi esetben. A hibákat részletesebben egy későbbi fejezetben kifejtettem.

2.10 Teszterv

A tesztelés szinte az egész projekt alatt jelen volt, hiszen minden elem működését a fejlesztés közben ellenőrizni kellet. Minél több elem volt kész, annál több variációban kellett megfigyelni az egymással való interakciójukat, és ezeknek a működését. Miután én ismertem a játékelemek összetételét, jellemzőit használati módjukat, ezért nem tudtam volna minden hibát felfedni és kijavítani. Ezért a tesztelési fázis utolsó szakaszaiban szükséges volt más személyek bevonása is, akik véleményt alkottak a játékról. A véleményekben szereplő hibák elemzése és javítása után lehetett véglegesíteni a szoftvert.

3. Felhasználói dokumentáció

A szoftver egy platformer játék, melynek lényege játékelemről játékelemre való átugrással, ellenségek leküzdése vagy kikerülése által elérni a kijelölt célt. A játék harcrendszere közelharcos. Alaptörténet:
Egy kitalált világ mesebeli erdejében járunk, ahol óriási fákkal van karakterünk körülvéve. Főhősünk, aki nem mellesleg egy kincsvadász, egyetlen céllal van megbízva. Meg kell szereznie eme varázslatos erdő kincseit. Ellenségekkel, veszélyekkel és csapdákkal teli út vezet arra a bizonyos helyre, amiről minden este meséltek neki a vének. Hogyan jut el oda főhősünk? Vajon megöli az útjába kerülő lényeket, akik a kincset védelmezik?

3.1 Rendszerkövetelmények

A játék PC-re lett fejlesztve, Windows operációs rendszerre.
Minimális rendszerkövetelmény:

- processzor: 1 gigahertz-es (GHz) vagy gyorsabb processzor vagy SoC (rendszerchip),
- RAM: 2 GB vagy ennél több,
- merevlemez-terület: minimum 50 MB szabad terület,
- videokártya: DirectX 9-kompatibilis vagy újabb, WDDM 1.0 típusú illesztőprogrammal.

3.2 Játékosok felhasználói utasításai

A program futtatásához feltétlenül szükségesek a következő fájlok:

- MonoBleedingEdge (mappa),
- Játék_Data (mappa),
- Játék (alkalmazás),
- UnityCrashHandler64 (alkalmazás),
- UnityPlayer.dll (alkalmazáskiterjesztés),
- WinPixEventRuntime.dll (alkalmazáskiterjesztés).

A program a Játék nevű alkalmazással indul, elindítása után a főmenü fogadja a játékost. Itt a ”JÁTÉK” gombbal elindíthatja a játékot, A ”BEÁLLÍTÁSOK” gombbal elérhetőek a játék beállításai, az ”IRÁNYÍTÁS” menüpontban tájékozódhat a karakter irányításáról, illetve egyéb funkciókról, végül a ”KILÉPÉS” gombbal bezárja a programot.
A beállítások menüben a következő beállítások elérhetőek:

- felbontás (monitortól függ),
- teljes képernyős mód / ablakos mód,
- grafikai beállítások (nagyon alacsony, alacsony, közepes, magas, nagyon magas, ultra),
- játék hangereje.

A játék közben az Esc billentyű lenyomásával szüneteltethető a játék és elérhetővé válik a szünet menü. A ”FOLYTATÁS” gomb megnyomásával a játék folytatódik, az ”IRÁNYÍTÁS” gombra való kattintással megjelenik az irányítással kapcsolatos információ, végül a ”KILÉPÉS” gombbal visszatérhetünk a főmenübe, azonban az összes eddigi eredmény elveszik.
A játék használatához nem szükséges előzetes regisztráció, vagy bármilyen más adat megadása.

3.3 A játékosok jogosultságai

A szoftver megnyitása, bezárása. A játék teljes körű használata. A beállításokhoz való hozzáférés, és azok változtatása. Az irányításról és egyéb funkciókról szóló menüpont megnyitása. A menüben gombokkal való navigálás.

4. Tesztelés, hibák

A tesztelésnek a fejlesztési folyamat nagy részében fontos szerepe volt. Mindegy egyes szakasz után szükséges volt bizonyos mennyiségű teszt lefuttatása a hibák felfedése céljából. Egy-egy hibás tesztet azonnali hibakeresés, finomhangolás és újabb tesztelés követett, mindaddig, amíg a tervnek megfelelő lett az adott állapot és folytatódhatott a következő szakasz. Elsősorban a Unity-ban történtek nagy számban a tesztelések. Két nagyobb hibával szembesültem, amit később sikerült megoldanom, azonban negatívan befolyásolta a fejlesztés folyamatát.

Az első hiba a karakter megalkotásakor keletkezett és csak a fejlesztés végén került megoldásra. A hiba természete egyszerű volt, könnyen előidézhető. A karakter ugráskor a beállított értéknél jóval nagyobbra ugrott. Ez egy ”game breaking” probléma volt. Annyit jelent, hogy ez a hiba a játékot játszhatatlanná teszi, és erősen negatívan hat a játékélményre. A hibák a következő tényezők miatt keletkeztek. Elsősorban a kódban rossz helyre lett írva egy metódus.

```
private void CheckGrounded()
{
    grounded = Physics2D.OverlapCircle(groundCheck.position, groundCheckRadius, groundLayer);
    anim.SetBool("isGrounded", grounded);
    anim.SetFloat("verticalSpeed", RB.velocity.y);
}
```

13. ábra: A karakterhez tartozó kód részlete

Az ábrán látható metódus azt vizsgálta, mikor van a levegőben a karakter és mikor áll egy platformon. Ez az Update szekcióban helyezkedett el, a FixedUpdate helyett. A különbség a kettő között a következő. Az Update részben lévő metódusok annyiszor futnak le, amennyi a monitor frame rate-je[12]. Ellenben a FixedUpdate-ben annyiszor futnak le a metódusok, amennyi a grafikus motorban meg van adva, ezért monitortól függetlenül ez az érték ugyanannyi, így nem okoz gondot. Az összes grafikus vonatkozású elemet az Update, az irányításért felelős elemet pedig a FixedUpdate részben kellett elhelyezni. A másik tényező, ami előidézte ezt a hibát annak az OverlapCircle-nek a nagysága, ami a karakter állapotának ellenőrzésében segített. Ez egy olyan kör alakú elem a játékban, ami, ha érintkezik a platformmal, akkor a karakter nem a levegőben van. Ha nem érintkezik semmivel, akkor a karakter a levegőben van. Ennek a körnek a mérete túl nagy volt, ezért csökkenteni kellett, illetve a karakteren való elhelyezkedését is módosítani kellett.

[12] Másodpercenkénti képkockasebesség

A másik nagyobb hiba, amivel szembesültem a fejlesztés során az az ellenség megalkotásakor jelentkezett. A felépítése nem volt megfelelő a játékhoz, ezért teljesen újra kellett írni a hozzá tartozó szkriptet, illetve változtatni kellett a modelljén. Ez is hasonlóan ”game-breaking” hiba, amit mindenképp orvosolni kellett. Ennek a problémának a fennállási ideje kevesebb volt az előzőhöz viszonyítva, hamar megoldást sikerült találnom rá.
Miután elkészült az első prototípus, a játék ”buildelésre” került. Ez annyit tesz, hogy a szerkesztő program véglegesíti, tömöríti a fájlokat, létrehozza az alkalmazást, a könyvtárakat, amivel elindítható a játék, illetve futtathatóvá teszi az adott platformon bármilyen segédprogram nélkül. Voltak olyan hibák, amik csak ebben a verzióban bukkantak fel. Kisebb hibákról volt szó, amik azonnali javításra kerültek.

4.1 Tapasztalatok

Hatalmas kihívás volt ezt a projektet egyedül megalkotni. A fejlesztés közben derült ki, mennyire fontos a szakaszok előzetes megtervezése. Az elemek megalkotása, tesztelése, finomhangolása egy nagyon aprólékos munka, ami hatalmas kitartást követel.
Nagyon fontos volt a játék elemeinek megalkotása során figyelembe venni azt, hogy hogyan fognak egymásra hatni, egymáshoz viszonyulni. Ki kellett találni egy egységes rendszert, amiben mindennek meg volt a helye, és megfelelően a tervek szerint működött. Természetesen az alkotási folyamat közben ez alakult, változott. Egyre több ötlet vetődött fel, aminek egy része egy kis átalakítással vagy változatlanul beépítésre került a folyamatokba. Némely gondolat viszont elvetésre került, mert nem lehetett megfelelően implementálni a játékba.
Az egész szoftver megalkotását, a hibák és nehézségek ellenére élveztem, és szeretnék továbbra is foglalkozni ezzel. Rengeteg lehetőség rejlik a Unity-ben, aminek csak töredékét használtam. Eddig is tisztában voltam, mekkora munka egy játékot megcsinálni, de csak e projekt befejezése után jöttem rá igazán, mennyire aprólékos és munkaigényes feladat ez.

5. Továbbfejlesztési lehetőségek

A szoftver úgy lett megalkotva, hogy könnyen bővíthető legyen. Az alábbi lehetőségekkel lehetne továbbfejleszteni a játékot.

Több fajta karakter és ahhoz tartozó karakter választó képernyő létrehozása. A karakterhez számos egyéb elemet hozzá lehetne még adni például felszereléseket, több fajta támadási módot, energia rendszert.

Több fajta ellenség megtervezése, a pálya végén egy erősebb ellenféllel való harc bevezetése. Az ellenfelek ”okosabbá” tétele kódjuk kiegészítésével.

Több pálya megépítése és hozzákapcsolása a meglévő játékmenethez.

Többféle értékelőrendszer bevezetése a játékba például idő szerinti értékelés.

Külső rendszereket is lehetne alkalmazni. A játékosokat és eredményeiket egy adatbázisban eltárolni és nyilvántartani.

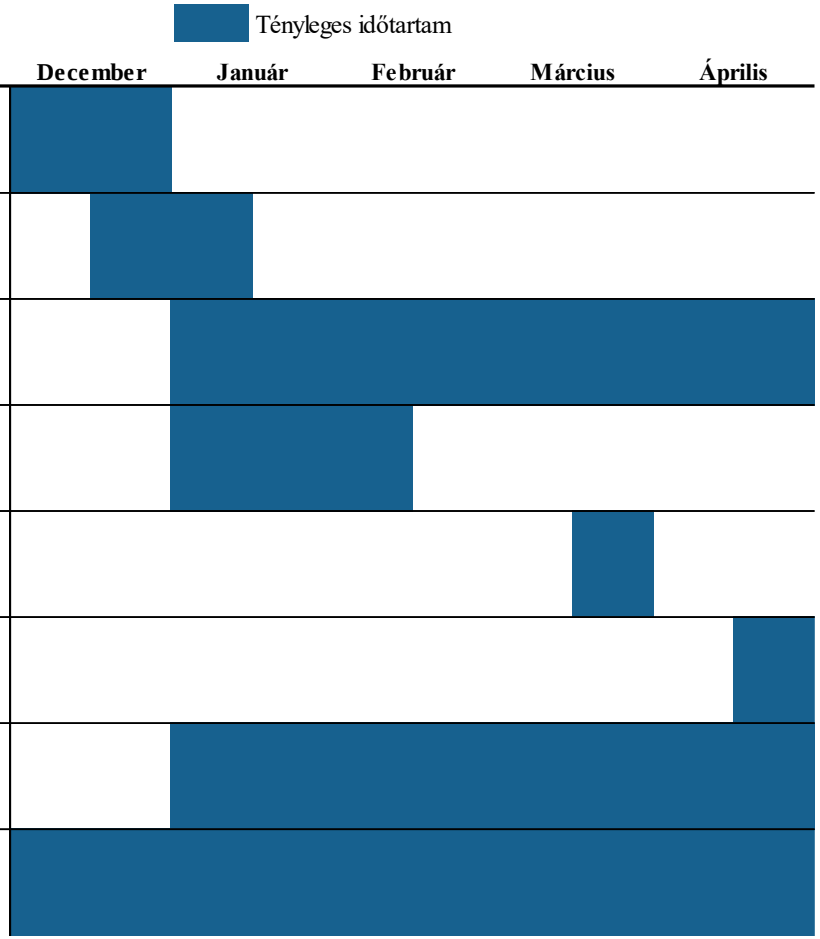
Rekordrendszer bevezetése.

A projekt befejeztével ezek a lehetőségek jutottak az eszembe. Mindenképpen ki szeretném próbálni ezeknek az elemeknek a megalkotását, akár ehhez a játékhoz, akár egy másikhoz.

6. Mellékletek

1. számú melléklet: Gantt-diagram, saját készítésű ábra

TEVÉKENYSÉG	TERV KEZDETE	TERV IDŐTARTAMA	TÉNYLEGES KEZDÉS	TÉNYLEGES IDŐTARTAM
téma, stílus, és fejlesztői eszközök kiválasztása	2019.12.01	fél hónap	2019.12.01	1 hónap
kép- és hanganyag gyűjtése	2019.12.15	fél hónap	2019.12.15	1 hónap
a karakter és az ellenségek megtervezése, animálása, kódolása	2020.01.15	4 hónap	2020.01.15	4 hónap
platformok és játékelemek megtervezése, kódolása	2020.01.15	másfél hónap	2020.01.15	másfél hónap
GUI és a menük megtervezése, összeállítása, kódolása	2020.03.15	1 hónap	2020.03.15	fél hónap
pálya struktúra megtervezése, megvalósítása	2020.04.10	fél hónap	2020.04.10	fél hónap
tesztelés, finomhangolás	2020.01.15	4 hónap	2020.01.15	4 hónap
dokumentáció megírása	2019.12.01	5 hónap	2019.12.01	5 hónap



7. Forrásjegyzék

A játékban található ingyenes vizuális elemek:

1. https://craftpix.net/freebies/free-cartoon-forest-game-backgrounds/
2. https://craftpix.net/freebies/free-golems-chibi-2d-game-sprites/
3. https://craftpix.net/freebies/free-jump-game-items/
4. https://craftpix.net/freebies/free-reaper-man-chibi-2d-game-sprites/
5. https://www.gamedevmarket.net/asset/fantasy-gui-set/
6. https://www.gamedevmarket.net/asset/fantasy-icon-pack-1178/

A játékban található ingyenes hang effektek:

7. https://assetstore.unity.com/packages/audio/sound-fx/fantasy-sfx-32833
8. https://assetstore.unity.com/packages/audio/music/premium-fantasy-music-pack-1-119264
9. https://assetstore.unity.com/packages/audio/sound-fx/free-sound-effects-pack-155776#description

8. Ábrajegyzék