

**VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**

SZAKDOLGOZAT

Patkósi Kevin, Fazekas Dávid

2023.

**VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**



SZAKDOLGOZAT

Encircled

Konzulens: Wiezl Csaba

Készítette: Patkósi Kevin

Fazekas Dávid

Hallgatói nyilatkozat

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra.

Patkói Kevin

Fazekas Dávid

Konzultációs lap

Vizsgázók neve: Patkósi Kevin, Fazekas Dávid

Szakdolgozat címe: Encircled

Program nyújtotta szolgáltatások:

- Interaktív játékmenet
- Testreszabható beállítások
- Ügyességi megpróbáltatás
- Történetmesélés

Sorszám	A konzultáció időpontja	A konzulens aláírása
1.	2022.10.10.	
2.	2022.11.14.	
3.	2022.12.12.	
4.	2023.01.16	
5.	2023.02.13.	
6.	2023.03.13	

A szakdolgozat beadható:

Vác, 2023.....

A szakdolgozatot átvettetem:

Vác, 2023.....

Konzulens

A szakképzést folytató
intézmény felelőse

Tartalomjegyzék

Hallgatói nyilatkozat.....	3
Konzultációs lap.....	4
Tartalomjegyzék	5
Témaválasztás	7
1 Fejlesztői dokumentáció	8
1.1 Ismertető.....	8
1.2 Fejlesztői környezet	8
1.2.1 Visual Studio Code.....	8
1.2.2 Unity Engine	8
1.2.3 Almotor – Low Poly Shooter	9
1.2.4 Unity Hub	10
1.2.5 Unity Asset Store – Felhasznált Assetek	10
1.2.6 ChatGPT.....	14
1.2.7 Trello	14
1.3 Kódnyelv	15
1.3.1 C#.....	15
1.4 Adatszerkezet.....	16
1.4.1 3D modell	16
1.4.2 Textúra	17
1.4.3 Hangfájlok	17
1.4.4 Scriptek.....	17
1.4.5 Konfigurációs fájlok	18
1.5 Felhasználói esetmodell.....	18
1.6 Scriptek / Osztályok / Modulok	19
1.6.1 Projectile.cs	19
1.6.2 AiController.cs.....	21
1.6.3 WaveSpawner.cs	23
1.6.4 MainMenu.cs	24
1.6.5 AiController lepéldányosítása a Projectile.cs-ben	25
1.6.6 GameplayController.cs.....	26
1.6.7 HealthBar.cs	27
1.6.8 KillTxtController.cs	28
1.6.9 PlayerHealth.cs.....	29
1.6.10 SceneFade_ToGame.cs	30

1.6.11	SceneFader_Menu.cs	30
1.6.12	SoundManager.cs.....	31
1.6.13	ZombieAudio.cs.....	32
1.7	Jelenetek	33
1.7.1	Main_Menu.unity.....	33
1.7.2	Encircled_map_V1.unity	33
1.7.3	Win_Screen.unity.....	34
1.7.4	Death.unity.....	34
1.7.5	TestMap.unity	35
1.8	Továbbfejlesztési lehetőségek	36
2	Felhasználói dokumentáció.....	37
2.1	Encircled.....	37
2.1.1	Bemutatás	37
2.1.2	Hardver követelmények.....	37
2.1.3	Szoftver követelmények.....	37
2.2	Telepítési útmutató	38
2.2.1	Letöltés.....	38
2.2.2	Kicsomagolás.....	38
2.2.3	Indítás.....	38
2.3	Játékkezelési útmutató	39
2.3.1	Főmenü	39
2.3.2	Beállítások	40
2.3.3	Játékban	40
3	Irodalomjegyzék	42
3.1	Internetes források.....	42
4	Mellékletek	43

Témaválasztás

A csapatunk egy első személyből játszódó, kör alapján haladó játékot teremtett meg, melynek a célja hogy a játékos az őt támadó ellenfeleket, melyek a mi esetünkben zombik lettek, legyőzze a rendelkezésére álló fegyverekkel. A modern játékipar besorolása szerint a Zombie FPS kategóriába sorolható a játékunk.

A témaválasztást több tényező is inspirálta; Mind a ketten részesei voltunk annak a remek pillanatnak a videójátékipar történelmének során, amikor a Treyarch nevű cég az Activision kiadójával megteremtette a Call of Duty: World At War nevű játékot, amelyben a sorozat ötödik megjelenésében létre hoztak egy, a mi mostani játékunkra hasonlító játékmódot. Ez hamar világhíres popularitást szerzett magának, és inspirált minket egy hasonló játék létrehozására.

Ezek mellett a Zombie FPS egy rendkívül nehezen megvalósítható műfaj a videójátékiparban, viszont hatalmas potenciállal rendelkezik siker és popularitás terén ha megfelelő erőbefektetés kerül rá. Ilyenre példa a Valve nevű kiadó Left 4 Dead 1 és Left 4 Dead 2 videójátéka, melyek egyszer Game of the Year díjat; az ipar legnagyobb megtisztelését nyerték el, valamint egyszer nominálásra is kerültek erre a nemes kitüntetésre. Úgy gondoljuk, hogy a sikeres elhelyezkedéshez ebben az iparban szükséges hogy kövessük a trendeket és igényeket a szoftverrel amit létrehozunk, így választottuk ki azt a témát, aminek nagy esélye van sikerre lelnie az iránta lévő érdekeltség miatt.

A GameSight nevű videójátékokkal foglalkozó oldal adatgyűjtése szerint a 2021-es évben kettő és negymillió között mozgott az átlagos napi nézettsége annak a játék műfajnak amelyet a mi játékunk is képvisel a Twitch nevű játékközvetítésre használt weblapon. Ez az információ alátámasztja az elkövetésünket hogy ebben a műfajban történő elhelyezkedés megteremtheti a kellő érdeklődést egy ilyen fajta játékprojektnek.

Tanultunk az igazi fejlesztők és nagy kiadók hibáiból, akik egy kevés érdeklődéssel rendelkező műfajt választottak játékuknak, melynek eredményeképp több éves munkájuk, valamint több millió dollárnyi befektetésük veszik kárba az elenyésző játékosszámok és eladásoknak köszönhetően.

1 Fejlesztői dokumentáció

1.1 Ismertető

Az Encircled egy videójáték, melynek megcélzott felhasználói köre az átlagos számítógép használó akik szabadidejüköt zombikkal foglalkozó videójátékokkal szeretnék tölteni.

A játék célja egyszerű, a játékos egy elkerített területen az élőholtak folyamatosan támadó seregeit kell visszavernie a rendelkezésre álló fegyverekkel, amíg a segítség meg nem érkezik.

Ennek érdekében egy Low Polygon Stílusban felépített játékot alkottunk meg, melyet alacsony gépigényről híres, így több játékoshoz juttatható el a termékünk.

1.2 Fejlesztői környezet

1.2.1 Visual Studio Code

A Visual Studio Code mint az Unity Engine fő kódfejlesztői alkalmazása lett használatba véve a mesterremek megalkotásának érdekében. A felhasználóbarát terméke a Microsoftnak támogatja a saját kódnyelvüket melyben játékunk jelentős mennyisége készült el, a C#-ot, így egyszerűen tudtunk az Engine hibajelzőjéből információt a kódbazisunkban orvosolni.

1.2.2 Unity Engine

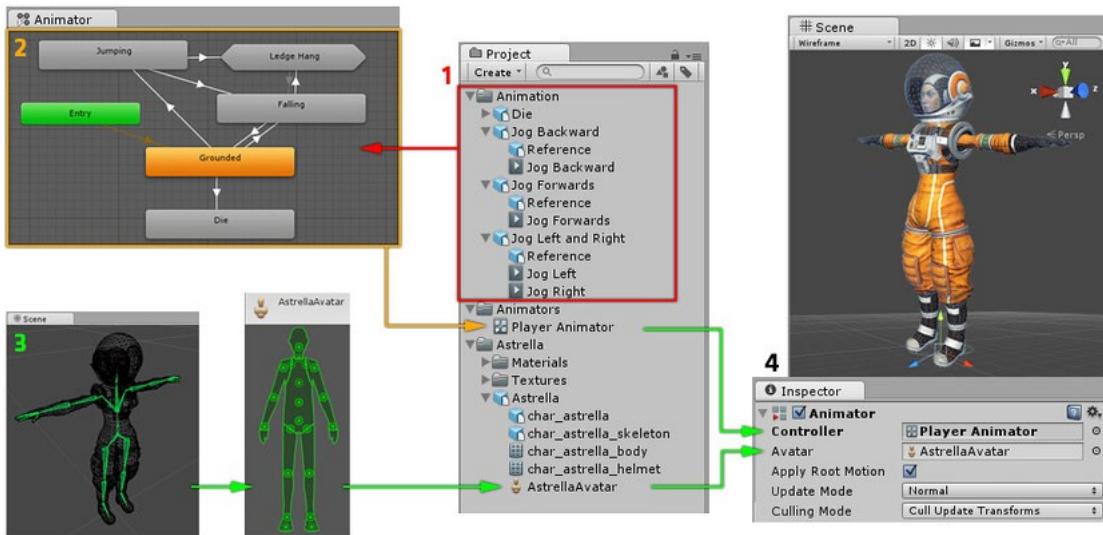
Az Unity Engine egy erőteljes játék motor, amelyet az Unity Technologies fejlesztett ki. Az Unity Engine lehetővé teszi a játékfejlesztők számára, hogy különböző platformokra, például mobil, számítógép, konzol és virtuális valóság eszközökre, játékokat hozzanak létre, így a kétségtelen volt hogy ez lesz végül a választásunk amikor egy stabil alapot kerestünk a játékunknak.

Számos érvek között megtalálható a legfontosabb; A kezdő-barát kivitelezése a programnak, a hatalmas felhasználói bázisból adódó széleskörű információhalmaz, valamint az Unity Asset Store, ahonnan rendkívül sok, ingyenes és minőségi segédanyaghoz lehet hozzáférni, amely megegyezővel a dolgát bármelyik játék fejlesztőjének. A hanganyagok és a textúrák innen történő beszerzésének köszönhetjük hogy kettő ember munkásságával el tudtuk juttatni a játékot a jelenlegi állapotához.

1.2.2.1 Unity Engine – Animator

A játék életre keltésének szempontjából elengedhetetlen animációk beszerzésére, működtetésére és szinkronizálására az Unity Engine Animator felületét vettük használatba. Ez az egyszerűen megtanulható, átlátható és

testreszabható eszköz lehetőséget nyújt előre létrehozott animációk testreszabására, feltételhez kötésére és összekötésére a játék alap funkcióival. A végeredmény pedig egy egyszerű, de nagyszerűen elkészített, animált karakter.



1.2.3 Almotor – Low Poly Shooter



A játék kezdetleges fázisaiban egy almotor is használatba lett véve, melynek a neve Top Down Engine, de a fejlesztés későbbi szakaszaiban az almotor használatából adódó komplikációk miatt ennek elhagyása mellett döntöttünk, és egy új almotor használatával, egy új szemszögből folytattuk a játék fejlesztését;

Harmadik személy nézetéből első személy nézetbe váltottunk. (Third Person Shooter →First Person Shooter)

A Low Poly Shooter használatba vételével használatba tudtunk venni olyan technológiákat, melyek lehetővé teremtették egy ekkora méretű projekt kivitelezését kettő amatőr játékfejlesztő számára is, hiszen az idő és erőforrásigényes szükségleteket mint az animációk, a textúrák, a karaktermodellek és a fegyverek mind megtalálhatóak ebben az almotorban; Csak a kódbeli alapjukat kellett megteremteni, és a saját használatunkra megírni, hogy megfelelően illeszkedjenek a játékunkba. Ezek mellett elengedhetetlenül fontos VFX és SFX mintákkal látta el a projektünket, melyek sajátkezű legyártása elkerpesztően sok időbe, pénzbe és energiába kerül a legnagyobb játékfejlesztő cégeknek is.

1.2.4 Unity Hub

Az Unity Hub lehetőséget nyújt a projektek fejlesztésének több számítógépen történő kivitelezésére, a verziókezelésre és a helyreállításra kritikus adatvesztés vagy kódbeli felülíródás esetén. A Collaborate lehetőség használatbavételével egyszerűen fejleszthettük eltérő gépeken eltérő részeit a játéknak, melyet végül egybe importálhattunk először egy teszt verzióba, majd a stabil verziójába a projektünknek.

1.2.5 Unity Asset Store – Felhasznált Assetek



Az Unity Asset Store egy online bolt, amely a Unity játékfejlesztők számára nyújt hozzáférést egy óriási készlet asset-ekhez, például modellekhez, textúrákhoz, hangokhoz, effektekhez és kódokhoz. Az Asset Store-ban a játékfejlesztők kiválaszthatják azokat az asset-eket, amelyekre szükségük van, és

hozzáadhatják őket saját játékukhoz. Ezek minden minőség és vírusellenőrzöttek az Unity Technologies által, így a fejlesztők biztonságos forrásból javíthatják játékuk minőségét.

Az általunk felhasznált assetek a projekt megvalósításához a következők:

1.2.5.1 Low Poly Cars

Low Poly Cars
Broken Vector | ★★★★☆ (28) | ❤ (1960)
FREE
🕒 352 views in the past week
[Open in Unity](#) [Heart](#)

License agreement: Standard Unity Asset Store EULA
License type: Extension Asset
File size: 2.5 MB
Latest version: 1.1
Latest release date: Jul 9, 2018
Original Unity version: 5.6.0 or higher
Support: [Visit site](#)

1.2.5.2 Low Poly Fence Pack

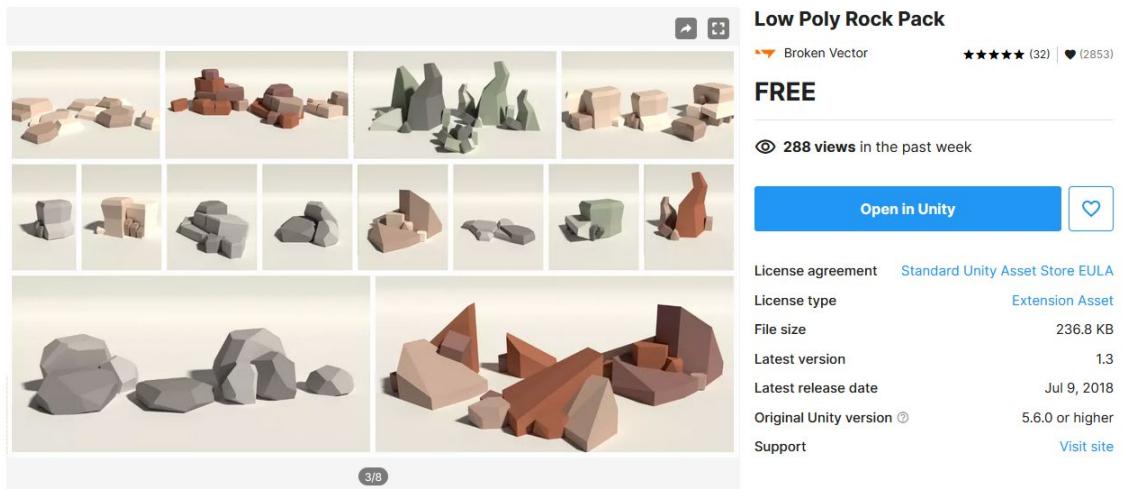
Low Poly Fence Pack
Broken Vector | ★★★★★ (18) | ❤ (1977)
FREE
🕒 306 views in the past week
[Open in Unity](#) [Heart](#)

BitGamey ★★★★★ a month ago
Great Models, Amazing Price
I only needed a specific fence model from this pack but it fitted exactly the purpose I needed. Great model, can't beat the price and no issues at all...
[Read more reviews](#)

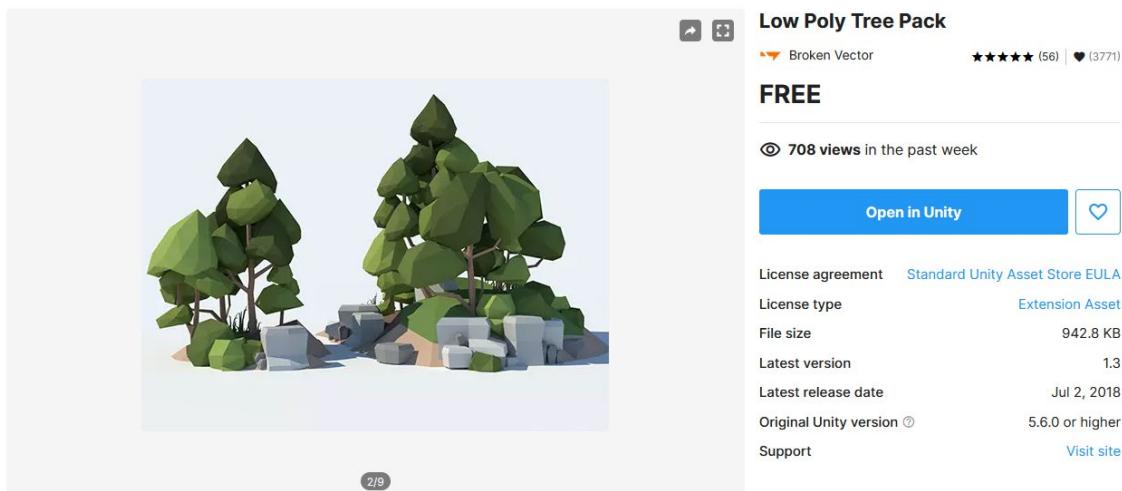
License agreement: Standard Unity Asset Store EULA
License type: Extension Asset
File size: 230.5 KB
Latest version: 1.1
Latest release date: Jul 9, 2018
Original Unity version: 5.6.0 or higher
Support: [Visit site](#)

Overview Package Content Releases Reviews Publisher info Asset Quality

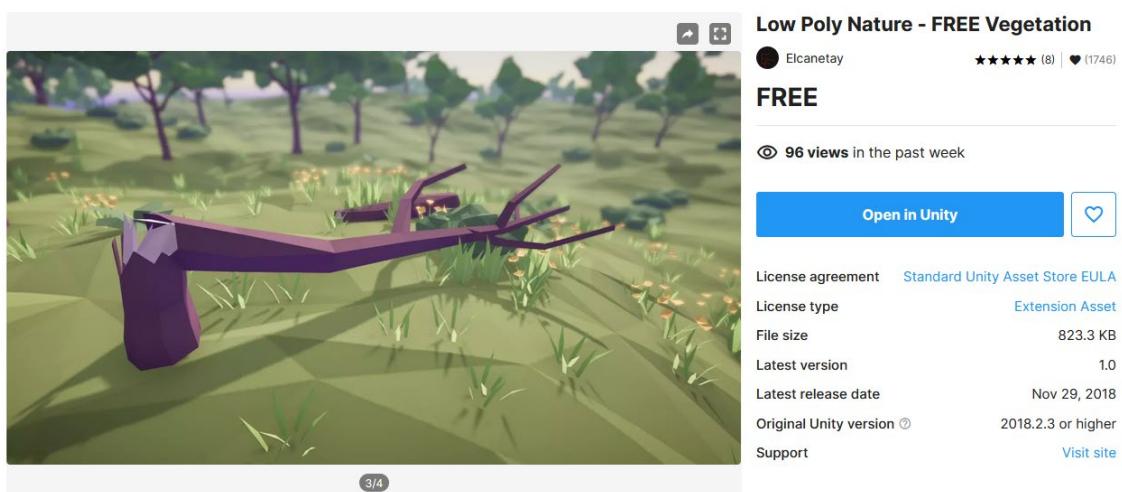
1.2.5.3 Low Poly Rock Pack



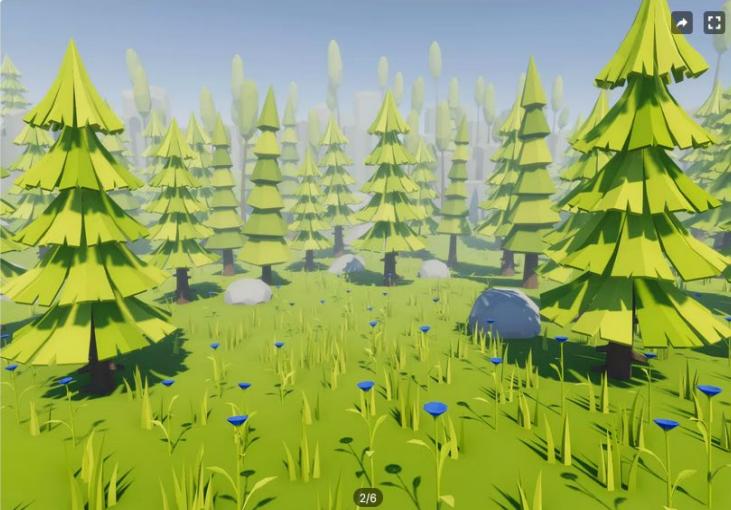
1.2.5.4 Low Poly Tree Pack



1.2.5.5 Low Poly Nature



1.2.5.6 Low Poly Wind



The image shows a low-poly forest scene with tall, green, coniferous trees and a field of small blue flowers. A large blue sphere sits on the grass. The interface includes a navigation bar at the top right and a progress bar at the bottom.

Low Poly Wind

Nicrom ★★★★★ (44) | ❤ (747)

FREE

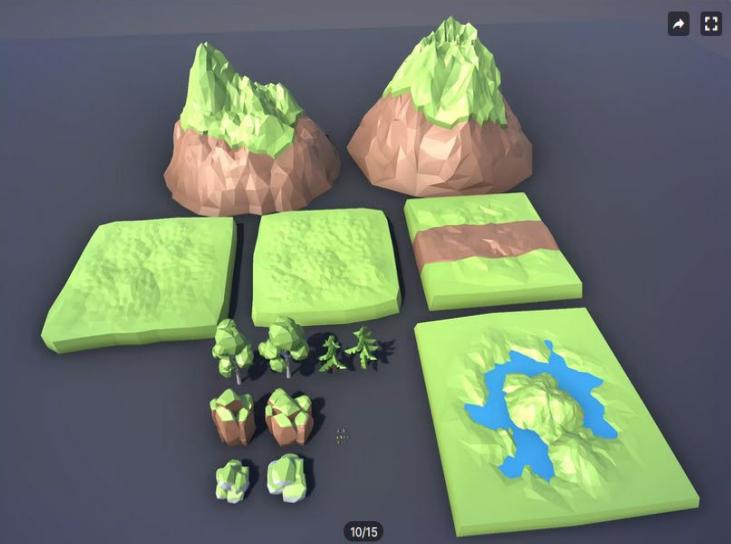
335 views in the past week

[Open in Unity](#) [Heart](#)

wsdsgn ★★★★★ a day ago
Amazing asset and even better support!
I am new to using shaders and still trying to learn my way around Unity, so I had a lot of problems getting the asset to work properly because I wasn't...
[Read more reviews](#)

License agreement	Standard Unity Asset Store EULA
License type	Extension Asset
File size	1.9 MB
Latest version	1.2.0
Latest release date	Feb 17, 2021

1.2.5.7 Low Poly Nature Forest



The image shows several low-poly terrain blocks and foliage assets. There are large green blocks representing hills and smaller brown blocks representing rocks or bushes. A blue water effect is visible on one of the green blocks. The interface includes a navigation bar at the top right and a progress bar at the bottom.

Free Low Poly Nature Forest

Pure Poly ★★★★★ (9) | ❤ (1357)

FREE

1267 views in the past week

[Open in Unity](#) [Heart](#)

Vykaash ★★★★★ 19 days ago
Looks great easy to use
The provided scenes were great for getting a test scene up and running.
[Read more reviews](#)

License agreement	Standard Unity Asset Store EULA
License type	Extension Asset

1.2.6 ChatGPT



Az OpenAI ChatGPT3 nevű mesterséges intelligenciája, mely az elmúlt hónapokban viharszerűen elárasztotta az internetet. Tényleges felhasználhatósága programozáson belül még korlátozott, játékfejlesztés terén pedig még nem vehető megfelelően használatba. Ettől függetlenül rendkívül sok hasznát tudtuk venni az AI-nak debugging és problémamegoldási téren, hiszen a Debugger és a Konzol compiler hibáinak értelmezésében rendkívül sok segítségre tudott lenni. Ennek köszönhetően hamarabb tudtuk oda fordítani a figyelmünket a kódban, ahol beavatkozásra volt szükség.

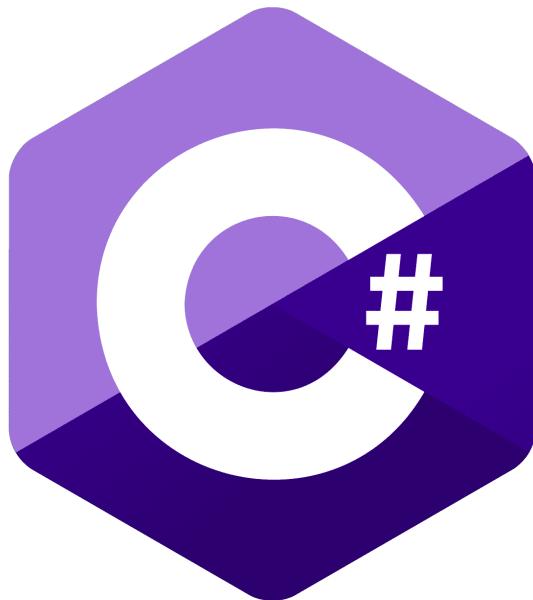
1.2.7 Trello

A Trello egy online feladatelesztásra használt weboldal, melyet csapatok világszerte használnak munkakoordináció érdekében. Az egyszerűsített feladatelesztás érdekében a mi projektünk is használatba vette a Trello szolgáltatásait, és minden egyes fejünkbe pattant ötletet a Trello-n belül helyeztünk el. Ezután mindenkoron nekiláttunk videóanyagot elemezni hogy az általunk elképzelt ötletet hogyan lehet megvalósítani, és a nagyobb hozzáértést elérő személy a kettőnk közül elvállalta az ötletet a weboldalon.

A screenshot of a Trello board titled "Encircled - to do". The board has several columns: "Ötlet", "Folyamatban", "SÜRGÖS", "Kivágott", "Kész", and "Bemutató Után". Each column contains one or more cards with "Add a card" buttons. On the far right, there is a sidebar with a "Board" dropdown and a list of items under "Bemutató Után": Hangerőszabályzó (K), Multiplayer (K), Vizuális HP csík (F), Lőszerek rendszer (K), Játék indulás bevezető (F), Felvételi fegyverek (K), Spawner Rework (F), Fegyver testreszabás (K), Animáció halálon (K-F) (delays), Végtelen játékmód (K), Win Condition - Win screen (K), Kill számítási (F), and Testresszabható beállítások (K). There is also a "+ Add a card" button at the bottom of the sidebar.

1.3 Kódnyelv

1.3.1 C#



A C# egy modern, általános célú programozási nyelv, amelyet a Microsoft fejlesztett ki. Az egyik legfontosabb alkalmazási területe a játékfejlesztés, különösen az Unity Engine-ben.

Az Unity Engine-ben a C# kódot az Unity Editor-on belüli MonoDevelop vagy a Visual Studio használatával lehet szerkeszteni. A kód írása során a C#-ban megszokott alapelvek érvényesek, például a szigorú típusellenőrzés, az objektumorientált programozás és a dinamikus memóriakezelés. Az Unity Engine kiterjedt API-jának köszönhetően a C#-ban írt kód könnyen hozzáférhet az engine funkcióihoz, például a fizikához, a hanghoz, a grafikához és a játékmenethez.

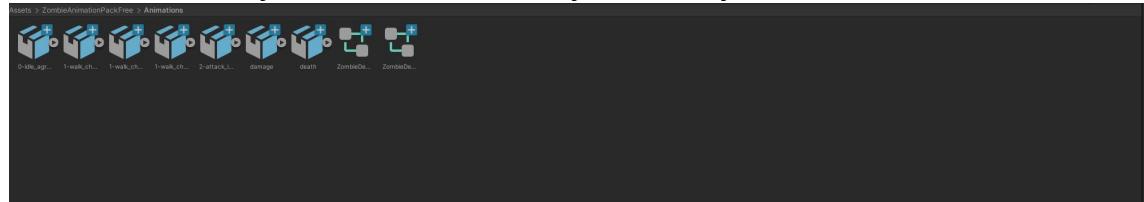
A C# használata nagyon fontos szerepet játszik az egyéni játékelemek, scriptek és funkciók létrehozásában. A játékfejlesztőknek lehetőségük van saját scriptek írására, amelyek később a játék logikáját és működését irányítják. Az engine beépített scripteket is tartalmaz, amelyek könnyen testreszabhatók a játékfejlesztők igényei szerint.

Az Engine-ben a C# nyelv és az engine API-jának hatékony használata lehetővé teszi a játékfejlesztők számára a magas minőségű és nagy hatékonyságú játékok létrehozását. Az Unity Engine-ben a C#-ban írt kódhoz rengeteg dokumentáció, tananyag és fórum segítségével lehet hozzáférni, így bárki könnyen elsajátíthatja a C# nyelv és az engine API-jának használatát.

1.4 Adatszerkezet

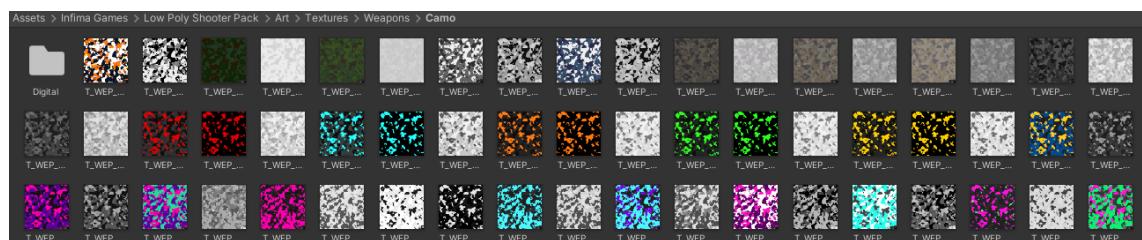
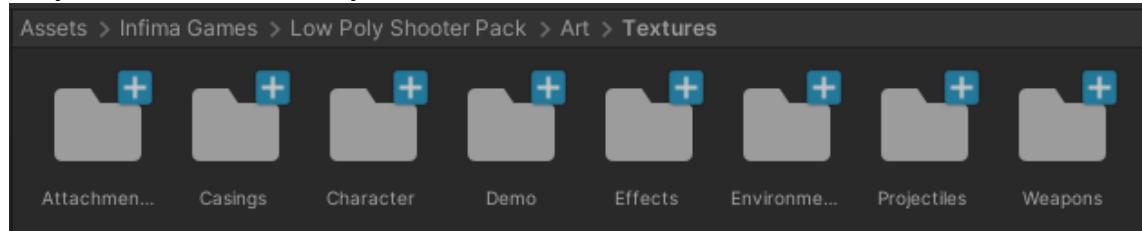
1.4.1 3D modell

A játék világában található objektumok, karakterek, épületek és környezetek 3D modelljei a videójáték fájlstruktúrájának egyik fontos elemei. Ez a modell tartalmazza az objektum méretét, formáját, textúráját és animációit.



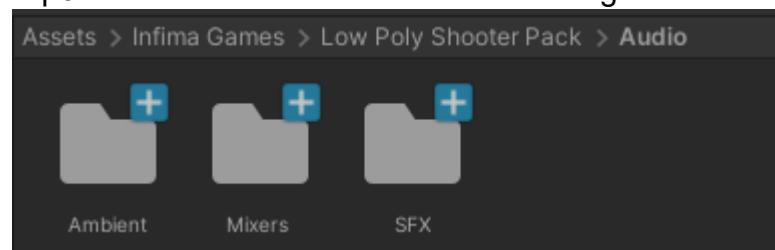
1.4.2 Textúra

A játék grafikus megjelenítéséért felelős textúrafájlok tartalmazzák az objektumok kinézetének részleteit, mint például a színeket, a mintákat, a fényhatásokat és az árnyékokat.



1.4.3 Hangfájlok

A játékban található hanghatások, zenei aláfestés szintén fontos adatszerkezet. A hangfájlok mp3 és wav formátumban találhatóak meg.



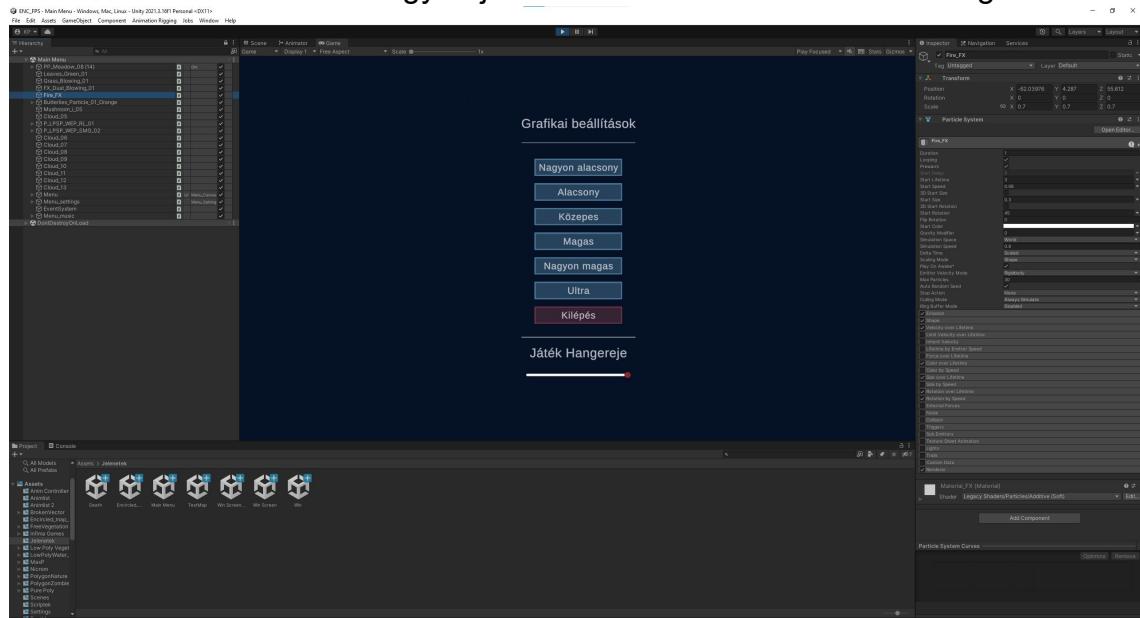
1.4.4 Scriptek

A játék működésének és interakcióinak irányításához használt kódok, amelyek C# programnyelven íródtak.

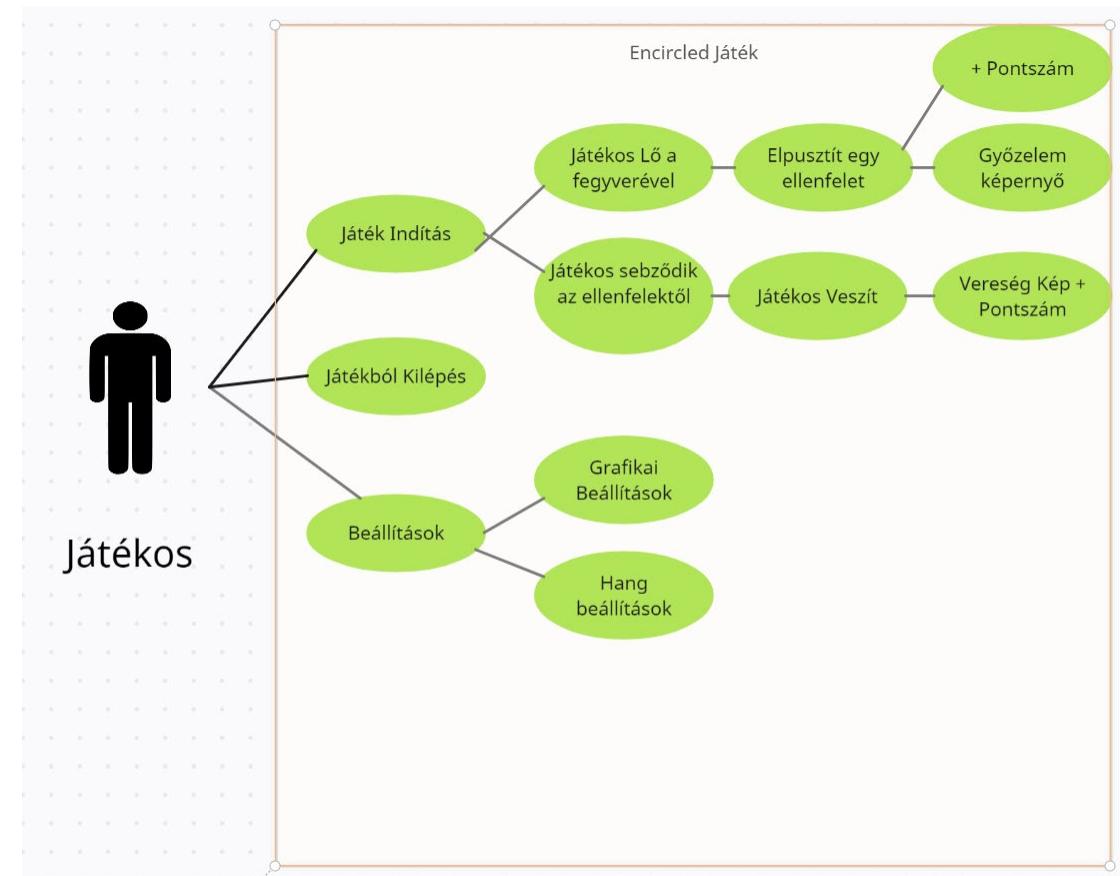


1.4.5 Konfigurációs fájlok

A játék beállításainak, mint például a grafikus beállítások, a hang beállítások, a vezérlés beállítások és az egyéb játékbeli beállítások tárolására szolgálnak.

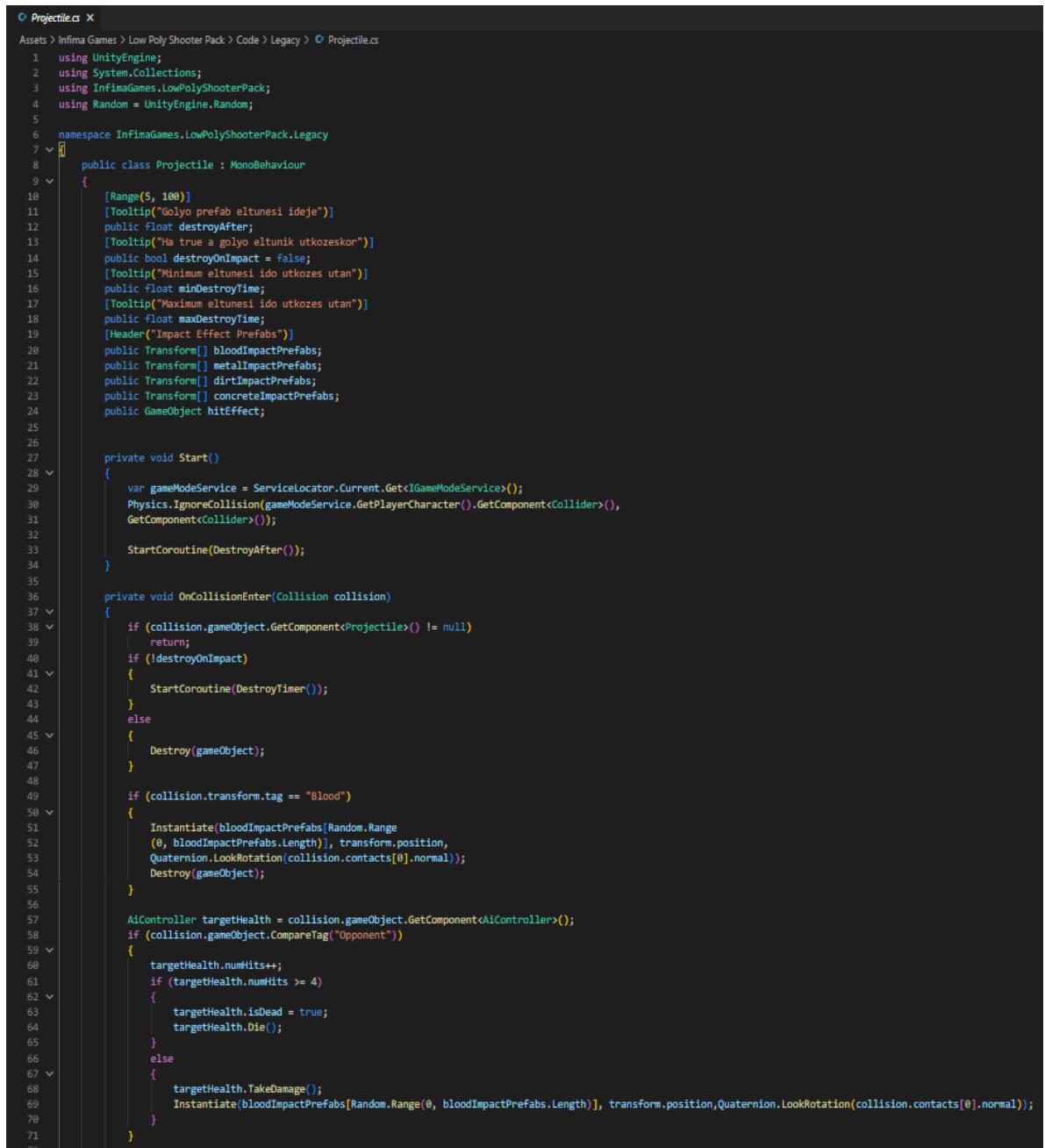


1.5 Felhasználói esetmodell



1.6 Scriptek / Osztályok / Modulok

1.6.1 Projectile.cs



```
Projectile.cs
Assets > Infima Games > Low Poly Shooter Pack > Code > Legacy > Projectile.cs

1  using UnityEngine;
2  using System.Collections;
3  using InfimaGames.LowPolyShooterPack;
4  using Random = UnityEngine.Random;
5
6  namespace InfimaGames.LowPolyShooterPack.Legacy
7  {
8      public class Projectile : MonoBehaviour
9      {
10         [Range(5, 100)]
11         [Tooltip("Golyó prefab eltunesi ideje")]
12         public float destroyAfter;
13         [Tooltip("Ha true a golyó eltulnuk utkozeskor")]
14         public bool destroyOnImpact = false;
15         [Tooltip("Minimum eltunesi idő utkozes után")]
16         public float minDestroyTime;
17         [Tooltip("Maximum eltunesi idő utkozes után")]
18         public float maxDestroyTime;
19         [Header("Impact Effect Prefabs")]
20         public Transform[] bloodImpactPrefabs;
21         public Transform[] metalImpactPrefabs;
22         public Transform[] dirtImpactPrefabs;
23         public Transform[] concreteImpactPrefabs;
24         public GameObject hitEffect;
25
26
27         private void Start()
28         {
29             var gameModeService = ServiceLocator.Current.Get<IGameModeService>();
30             Physics.IgnoreCollision(gameModeService.GetPlayerCharacter().GetComponent<Collider>(),
31                                     GetComponent<Collider>());
32
33             StartCoroutine(DestroyAfter());
34         }
35
36         private void OnCollisionEnter(Collision collision)
37         {
38             if (collision.gameObject.GetComponent<Projectile>() != null)
39                 return;
40             if (!destroyOnImpact)
41             {
42                 StartCoroutine(DestroyTimer());
43             }
44             else
45             {
46                 Destroy(gameObject);
47             }
48
49             if (collision.transform.tag == "Blood")
50             {
51                 Instantiate(bloodImpactPrefabs[Random.Range
52                     (0, bloodImpactPrefabs.Length)], transform.position,
53                     Quaternion.LookRotation(collision.contacts[0].normal));
54                 Destroy(gameObject);
55             }
56
57             AiController targetHealth = collision.gameObject.GetComponent<AiController>();
58             if (collision.gameObject.CompareTag("Opponent"))
59             {
60                 targetHealth.numHits++;
61                 if (targetHealth.numHits >= 4)
62                 {
63                     targetHealth.isDead = true;
64                     targetHealth.Die();
65                 }
66                 else
67                 {
68                     targetHealth.TakeDamage();
69                     Instantiate(bloodImpactPrefabs[Random.Range(0, bloodImpactPrefabs.Length)], transform.position, Quaternion.LookRotation(collision.contacts[0].normal));
70                 }
71             }
72         }
73     }
```

A Projectile.cs felelős a találat érzékelés megfelelő működéséért, és alapvető része a Low Poly Shooter almotornak. A TransformTag funkcióval létrehoztunk ebben a scriptben egy új taget Opponent néven, mely a játék ellenfeleire kerül ráruházásra. Ennek köszönhetően minden egyes lövés érzékeli, ha a mi általunk létrehozott ellenfeleket találja el, és megfelelően jár el.

```

72         if (collision.transform.tag == "Metal")
73     {
74         Instantiate(metalImpactPrefabs[Random.Range
75             (0, bloodImpactPrefabs.Length)], transform.position,
76             Quaternion.LookRotation(collision.contacts[0].normal));
77         Destroy(gameObject);
78     }
79
80
81     if (collision.transform.tag == "Dirt")
82     {
83         Instantiate(dirtImpactPrefabs[Random.Range
84             (0, bloodImpactPrefabs.Length)], transform.position,
85             Quaternion.LookRotation(collision.contacts[0].normal));
86         Destroy(gameObject);
87     }
88
89     if (collision.transform.tag == "Concrete")
90     {
91         Instantiate(concreteImpactPrefabs[Random.Range
92             (0, bloodImpactPrefabs.Length)], transform.position,
93             Quaternion.LookRotation(collision.contacts[0].normal));
94         Destroy(gameObject);
95     }
96
97     if (collision.transform.tag == "Target")
98     {
99         collision.transform.gameObject.GetComponent
100            <TargetScript>().isHit = true;
101         Destroy(gameObject);
102     }
103
104     if (collision.transform.tag == "ExplosiveBarrel")
105     {
106         collision.transform.gameObject.GetComponent
107            <ExplosiveBarrelScript>().explode = true;
108         Destroy(gameObject);
109     }
110
111     if (collision.transform.tag == "GasTank")
112     {
113         collision.transform.gameObject.GetComponent
114            <GasTankScript>().isHit = true;
115         Destroy(gameObject);
116     }
117 }
118
119 private IEnumerator DestroyTimer()
120 {
121     yield return new WaitForSeconds
122         (Random.Range(minDestroyTime, maxDestroyTime));
123     Destroy(gameObject);
124 }
125
126 private IEnumerator DestroyAfter()
127 {
128     yield return new WaitForSeconds(destroyAfter);
129     Destroy(gameObject);
130 }
131 }
```

1.6.2 AiController.cs

Készítette : Patkói Kevin

```
❶ AiController.cs ✘
Assets > PERSONAL > Scripts > ❷ AiController.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.AI;
5  using System;
6
7  public class AiController : MonoBehaviour
8  {
9      NavMeshAgent agent;
10     Animator anim;
11     public Transform target;
12     public float distance;
13     public int minTime = 0;
14     public int maxTime = 4;
15     public bool killed = false;
16     public int maxHealth = 100;
17     public int currentHealth;
18     public int enemiesKilled = 0;
19     public int numHits = 0;
20     public bool isDead = false;
21     public float deathTimer = 10.0f;
22     WaveSpawner spawner;
23     Collider _collider;
24     Rigidbody rb;
25
26
27     void Start()
28     {
29         currentHealth = maxHealth;
30         rb = GetComponent<Rigidbody>();
31         agent = GetComponent<NavMeshAgent>();
32         anim = GetComponent<Animator>();
33         _collider = GetComponent<Collider>();
34         StartCoroutine(WaitBefore());
35         target = GameObject.FindGameObjectWithTag("Player").transform;
36     }
37
38     private IEnumerator WaitBefore()
39     {
40         float waitTime = UnityEngine.Random.Range(minTime, maxTime);
41         yield return new WaitForSeconds(waitTime);
42     }
43
44     void Update()
45     {
46         if(killed == false)
47         {
48             agent.SetDestination(target.position);
49             anim.SetFloat("speed", agent.velocity.magnitude);
50             float distance = Vector3.Distance(agent.transform.position, target.transform.position);
51             if(Input.GetKeyDown(KeyCode.O))
52             {
53                 anim.SetTrigger("distanceT");
54             }
55             if(Input.GetKeyDown(KeyCode.K))
56             {
57                 anim.SetTrigger("death");
58             }
59         }
60     }
61 }
```

```

60     else if (_killed == true)
61     {
62         gameObject.GetComponent<NavMeshAgent>().velocity = Vector3.zero;
63         agent.SetDestination(agent.transform.position);
64         anim.SetFloat("speed", 0);
65     }
66     if(Input.GetKeyDown(KeyCode.L))
67     {
68         killed = true;
69     }
70     if(Input.GetKeyDown(KeyCode.N))
71     {
72         killed = false;
73     }
74 }
75
76
77 public void TakeDamage()
78 {
79     currentHealth -= 25;
80     if (currentHealth <= 0 && !isDead)
81     {
82         isDead = true;
83         enemiesKilled+=1;
84         Die();
85     }
86 }
87
88
89 public void Die()
90 {
91     if(spawner != null) spawner.currentEnemy.Remove(gameObject);
92     rb.isKinematic = true;
93     rb.detectCollisions = false;
94     _collider.enabled = false;
95     gameObject.GetComponent<NavMeshAgent>().isStopped = true;
96     GameplayController.instance.EnemyKilled();
97     Debug.Log("Target has been destroyed!");
98     anim.SetTrigger("isDead");
99     Destroy(gameObject,10);
100 }
101
102 public void setSpawner(WaveSpawner _spawner){
103     spawner = _spawner;
104 }
105 }

```

A legelső általunk elkészített modul, mely az Unity Engine beépített feltérképezési módszerét használva életre kelti a játék ellenségeit.

Az AI Controller felelős az ellenfelek pályán történő navigációjáért, a játék minden pillanatában friss információt ad az ellenfeleknek a játékos pozíciójáról, mely felé elkezdi vezetni őket, a pálya akadályait megkerülve. Ezt az Unity Engine Nav Mesh funkciójának köszönhetően tudtuk megvalósítani. Ezek mellett egy WaitBefore() Metódust is alkalmaztunk, melyben a játék késlelteti az ellenfelek elindulását, hogy az animációjuk ne legyen szinkronizálva, és egydinek tűnjön a haladásuk a játékos felé.

A fejlesztés végső szakaszaiban az egyszerűbb munkálatok és az animátorral történő szinkronizáció érdekében Dávid beleépítette a találat érzékelésért felelős scriptet az AI Controllerbe, így szinte az ellenfél teljes létezéséért egy darab komponens felel. Ez egyszerűbbé teszi a hibák orvoslását, az AI fejlesztését, és az animációk összehangolását.

1.6.3 WaveSpawner.cs

Készítette : Fazekas Dávid

```
WaveSpawner.cs
Assets > PERSONAL > WaveSpawner.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using TMPro;
6
7  public class WaveSpawner : MonoBehaviour{
8
9      [System.Serializable]
10     public class waveContent{
11         [SerializeField] GameObject[] enemySpawner;
12         [SerializeField] Transform[] spawnLocations;
13
14         public GameObject[] getMonsterSpawnList(){
15             return enemySpawner;
16         }
17
18         public Transform[] getSpawnLocations(){
19             return spawnLocations;
20         }
21     }
22
23     [SerializeField][NonReorderable] waveContent[] waves;
24     int currentWave = 0;
25     float spawnRange = 0;
26     public int enemiesKilled;
27     public List<GameObject> currentEnemy;
28     public TMP_Text waveText;
29
30     private bool allWavesSpawned = false;
31
32     void Start()
33     {
34         StartCoroutine(SpawnWavesWithDelay(5f, 22f));
35     }
36
37     IEnumerator SpawnWavesWithDelay(float delayBetweenWaves, float initialDelay)
38     {
39         yield return new WaitForSeconds(initialDelay);
40         allWavesSpawned = false;
41         currentWave = 0;
42         while (!allWavesSpawned)
43         {
44             spawnWave();
45             yield return new WaitUntil(() => currentEnemy.Count == 0);
46             currentWave++;
47             if (currentWave >= waves.Length)
48             {
49                 allWavesSpawned = true;
50                 yield break;
51             }
52             yield return new WaitForSeconds(delayBetweenWaves);
53         }
54     }
55
56
57     void spawnWave()
58     {
59         waveContent currentWaveContent = waves[currentWave];
60         Transform[] spawnLocations = currentWaveContent.getSpawnLocations();
61         for(int i = 0; i < currentWaveContent.getMonsterSpawnList().Length; i++){
62             int randomIndex = Random.Range(0, spawnLocations.Length);
63             Transform spawnTransform = spawnLocations[randomIndex];
64
65             GameObject newSpawn = Instantiate(currentWaveContent.getMonsterSpawnList()[i],FindSpawnLoc(spawnTransform),Quaternion.identity);
66             currentEnemy.Add(newSpawn);
67
68             AiController zombie = newSpawn.GetComponent<AiController>();
69             zombie.setSpawner(this);
70             waveText.text = (currentWave + 1).ToString() + " hullám";
71         }
72     }
73
74     Vector3 GetSpawnLocation(waveContent waveContent)
75     {
76         Transform[] spawnLocations = waveContent.getSpawnLocations();
77         int randomIndex = Random.Range(0, spawnLocations.Length);
78         Vector3 spawnPos = spawnLocations[randomIndex].position;
79
80         if (Physics.Raycast(spawnPos, Vector3.down, 5))
81         {
82             return spawnPos;
83         } else {
84             return GetSpawnLocation(waveContent);
85         }
86     }
87
88     Vector3 FindSpawnLoc(Transform spawnTransform)
89     {
90         Vector3 spawnPos;
91         float xLoc = Random.Range(-spawnRange, spawnRange) + spawnTransform.position.x;
92         float zLoc = Random.Range(-spawnRange, spawnRange) + spawnTransform.position.z;
93         float yLoc = spawnTransform.position.y;
94         spawnPos = new Vector3(xLoc,yLoc,zLoc);
95
96         if(Physics.Raycast(spawnPos, Vector3.down, 5)){
97             return spawnPos;
98         } else {
99             return FindSpawnLoc(spawnTransform);
100        }
101    }
102 }
```

A játék körorientált haladásmenetét megvalósító modul, a játék indításánál kerül automatikusan meghívásra. Általunk megadott láthatatlan pontokra kerül ráhelyezésre ez a script, mely azokat a pontokat használja az ellenségek időközbeni megteremtésére. A cél ezzel az, hogy a zombik a távolból, a játékos szemszögén kívül kerüljenek lehelyezésre.

A körök között a script vár egy megadott időközt, majd megkezdi a következő adag ellenfél kiosztását a pályára. Ezekből a pontokból több darab került elhelyezésre a pályán az ellenfelek szétszórása és a teljesítmény optimalizálásának érdekében.

1.6.4 MainMenu.cs

Készítette: Fazekas Dávid & Patkói Kevin



```
C:\MainMenu.cs X
C:\MainMenu.cs

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class MainMenu : MonoBehaviour
7  {
8      public GameObject canvas_menu_settings;
9      public GameObject canvas_menu;
10     public AudioSource menuMusic;
11
12     void Start()
13     {
14         GameObject canvas_menu_settings = GameObject.FindGameObjectWithTag("Menu_Settings");
15         GameObject canvas_menu = GameObject.FindGameObjectWithTag("Menu_Canvas");
16         menuMusic.Play();
17     }
18
19     public void CloseCanvas()
20     {
21         canvas_menu.SetActive(true);
22         canvas_menu_settings.SetActive(false);
23     }
24
25     public void Play()
26     {
27         SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
28     }
29
30     public void Settings(){
31         Debug.Log("Beállítások megnyomva");
32         canvas_menu_settings.SetActive(true);
33     }
34
35     public void Quit(){
36         Application.Quit();
37         Debug.Log("A játékos kilépett a játékból.");
38     }
39 }
40
```

A játék megnyitásakor először meghívott modul, a játék fő menüjének betöltéséért felelős, valamint az onnan történő navigációnak megoldásáért. Az

Unity Engine jelenetekre osztva vizualizál minden, amelyek között váltakozva jutunk el bizonyos pontokra. Ezek alapján a fő menü is egy jelenet, melyet előre szerkesztettünk meg a játékból. A gombok lenyomása a jelenetek közti váltakozást lépteti életbe.

A menüből elérhető a játék „Beállítások” füle. Az Unity Engine Canvasing rendszerét kihasználva a játék töltés nélkül elérhető rétegelt képeit kihasználva egy gyorsan elérhető beállítás menüt prezentál a játékos elé, ahol a grafikai komplexitást állíthatja a játékos. Az alacsony polygonnal rendelkező textúrák melyet felhasználtuk lehetőséget adnak széleskörű beállítások elérhetésére, melyet Nagyon Alacsony szintről Ultra szintig kategorizáltunk be. A haladás megfigyelhető az árnyékolás részletességében, a textúrák részletességében és a teljesítményben.



A főmenü animációját Patkósi Kevin, a kódalapját pedig Fazekas Dávid készítette el.

1.6.5 AiController lepéldányosítása a Projectile.cs-ben

Készítette: Fazekas Dávid

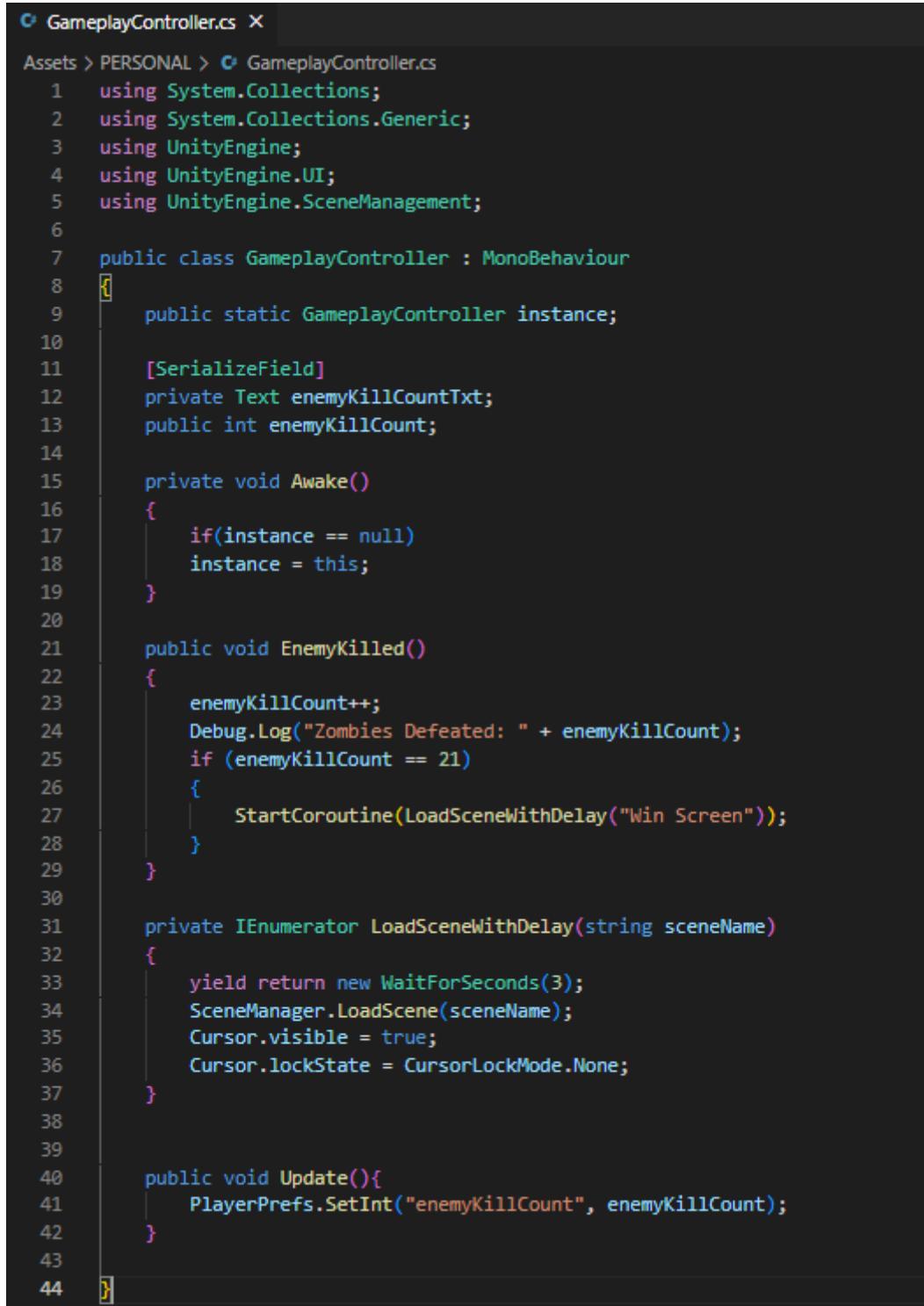
```
AIController targetHealth = collision.gameObject.GetComponent<AIController>();
if (collision.gameObject.CompareTag("Opponent"))
{
    targetHealth.numHits++;
    if (targetHealth.numHits >= 4)
    {
        targetHealth.isDead = true;
        targetHealth.Die();
    }
    else
    {
        targetHealth.TakeDamage();
        Instantiate(bloodImpactPrefabs[Random.Range(0, bloodImpactPrefabs.Length)], transform.position, Quaternion.LookRotation(collision.contacts[0].normal));
    }
}
```

Ahhoz hogy megkerüljük az Unity Engine RayCast rendszerének használatát, Dávid egy objektum alapú rendszert használt fel a töltények és lövések életre keltéséhez a játékunkban. A töltény egy tárgyként mozog végig a pályán a lövés leadásakor, mely egy colliderrel ütközés során ellenőrzéseket hajt végre, hogy

eldöntse hogy minek kell megtörténnie. Ennek köszönhetően Dávid létrehozott egy „Opponent” nevű ismertető címkét, melyet az összes ellenfélre ráruházunk amikor a játékba kerülnek. Innentől minden egyes találat felismeri hogy egy ellenfelet ért, és megfelelően meghívja a sebződés és az elpusztulással kapcsolatos funkciókat.

1.6.6 GameController.cs

Készítette: Fazekas Dávid

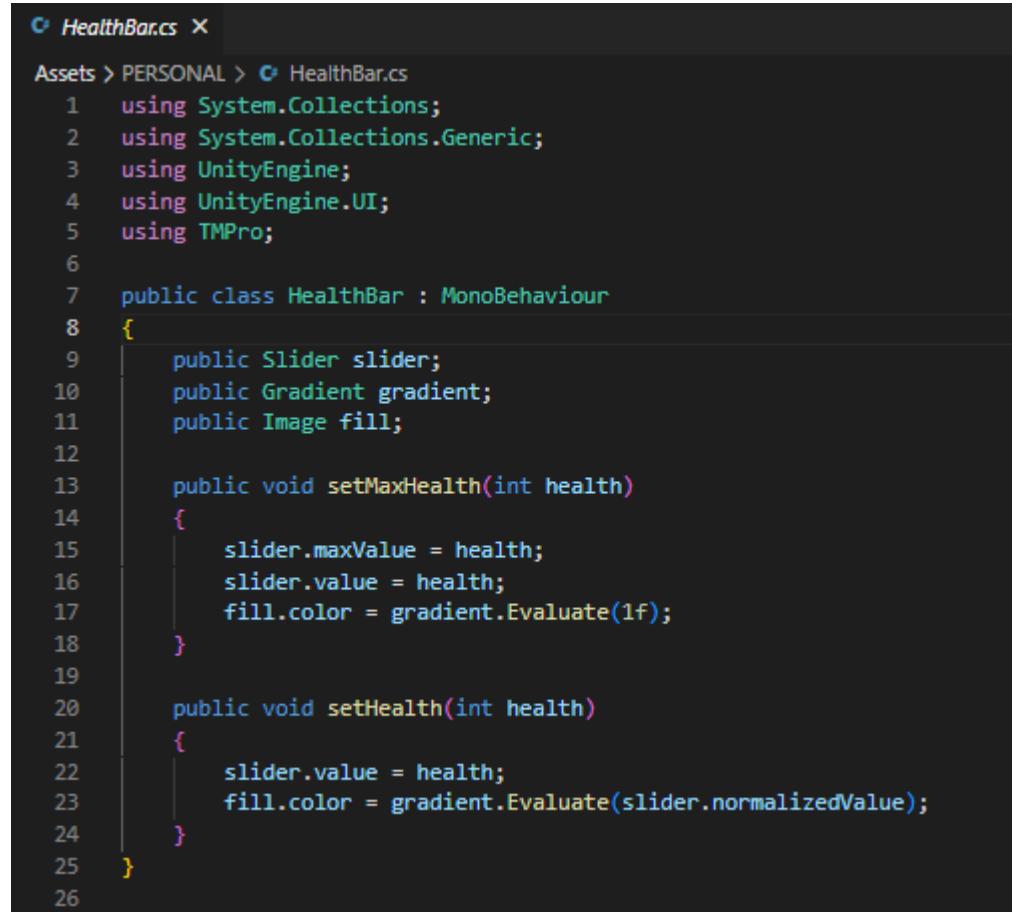


```
Assets > PERSONAL > GameController.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6
7  public class GameController : MonoBehaviour
8  {
9      public static GameController instance;
10
11     [SerializeField]
12     private Text enemyKillCountTxt;
13     public int enemyKillCount;
14
15     private void Awake()
16     {
17         if(instance == null)
18             instance = this;
19     }
20
21     public void EnemyKilled()
22     {
23         enemyKillCount++;
24         Debug.Log("Zombies Defeated: " + enemyKillCount);
25         if (enemyKillCount == 21)
26         {
27             StartCoroutine(LoadSceneWithDelay("Win Screen"));
28         }
29     }
30
31     private IEnumerator LoadSceneWithDelay(string sceneName)
32     {
33         yield return new WaitForSeconds(3);
34         SceneManager.LoadScene(sceneName);
35         Cursor.visible = true;
36         Cursor.lockState = CursorLockMode.None;
37     }
38
39
40     public void Update(){
41         PlayerPrefs.SetInt("enemyKillCount", enemyKillCount);
42     }
43
44 }
```

A GameplayController felelős a játék haladásának ellenőrzéséért és léptetéséért. A játék figyelembe tartja a játékos által elpusztított ellenfelek számát, majd ha egy általunk előre megadott követelményt elér, amely a bemutató példány esetében 21 legyőzött ellenfél, meghívja a Win Screen nevű jelenetet, amely a győzelmi képernyőhöz viszi át a játékost.

1.6.7 HealthBar.cs

Készítette: Patkói Kevin



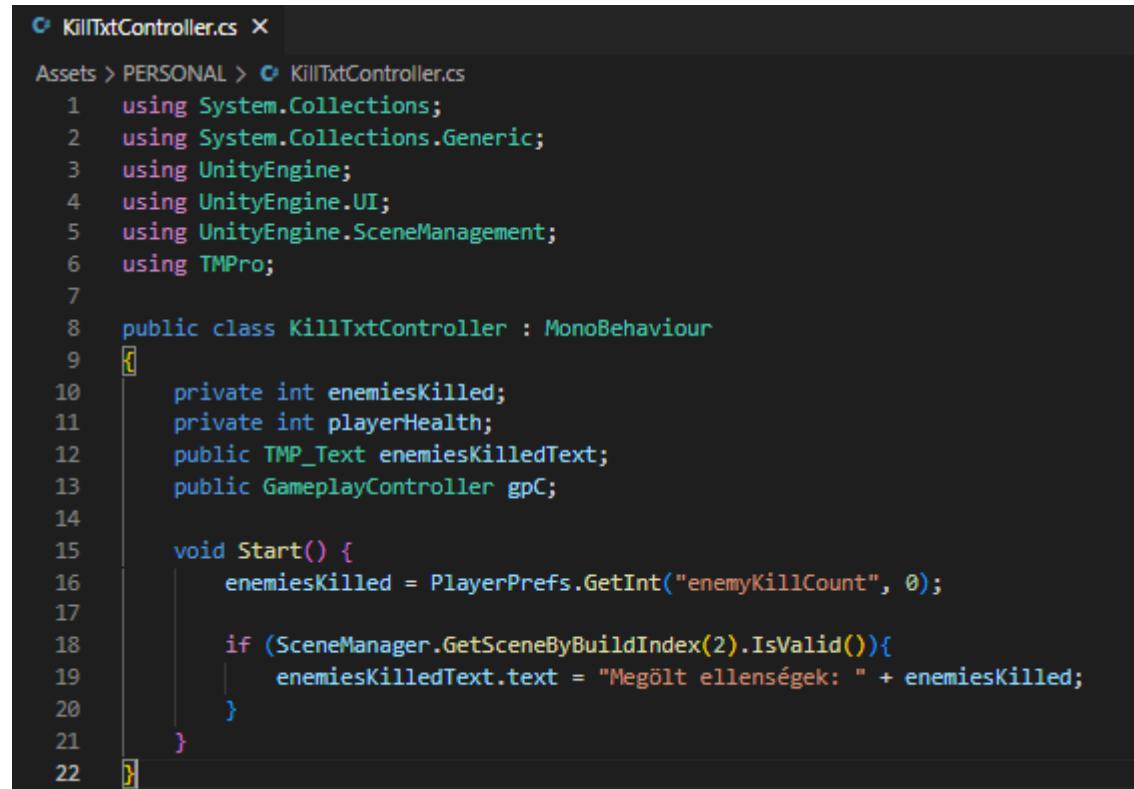
The screenshot shows a Unity code editor window with the file 'HealthBar.cs' open. The code defines a class 'HealthBar' that inherits from 'MonoBehaviour'. It uses several Unity namespaces: System.Collections, System.Collections.Generic, UnityEngine, UnityEngine.UI, and TMPro. The class contains two public fields: 'Slider slider' and 'Gradient gradient', and one public field 'Image fill'. It has two public methods: 'setMaxHealth(int health)' and 'setHealth(int health)'. The 'setMaxHealth' method sets the slider's maxValue to the given health value and its current value to that value, then sets the fill's color to the gradient evaluated at 1f. The 'setHealth' method sets the slider's value to the given health value and the fill's color to the gradient evaluated at the slider's normalized value.

```
Assets > PERSONAL > HealthBar.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using TMPro;
6
7  public class HealthBar : MonoBehaviour
8  {
9      public Slider slider;
10     public Gradient gradient;
11     public Image fill;
12
13     public void setMaxHealth(int health)
14     {
15         slider.MaxValue = health;
16         slider.value = health;
17         fill.color = gradient.Evaluate(1f);
18     }
19
20     public void setHealth(int health)
21     {
22         slider.value = health;
23         fill.color = gradient.Evaluate(slider.normalizedValue);
24     }
25 }
26
```

A HealthBar.cs felelős a képernyőn játék közben látható életerő csík vizuális megjelenéséért, és annak színalapú változásáért a játék haladása közben.

1.6.8 KillTxtController.cs

Készítette: Fazekas Dávid



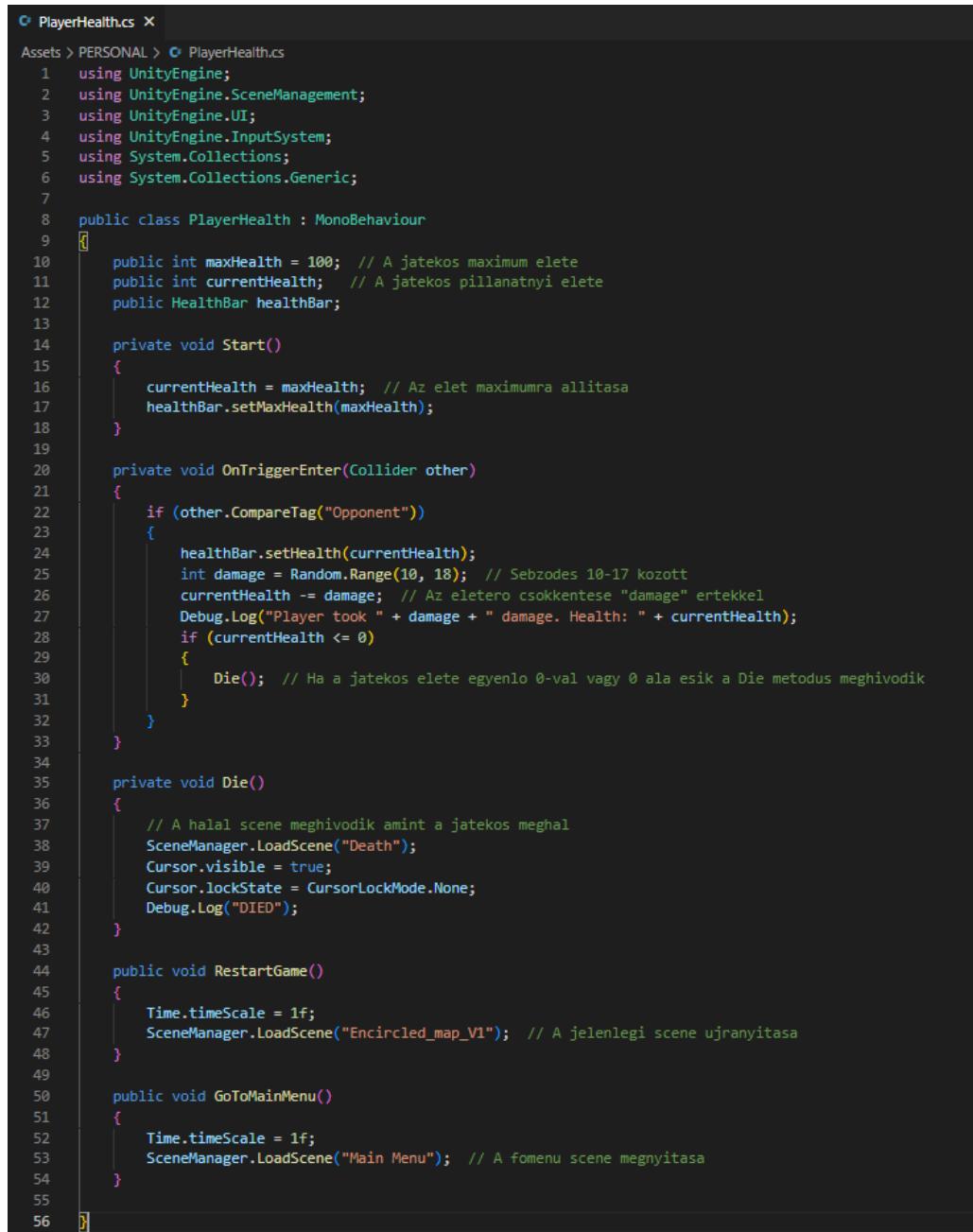
The screenshot shows a code editor window with the file 'KillTxtController.cs' open. The code is written in C# and defines a class 'KillTxtController' that inherits from 'MonoBehaviour'. The class has private fields for 'enemiesKilled' and 'playerHealth', and public fields for 'enemiesKilledText' (of type 'TMP_Text') and 'gpC' (of type 'GameplayController'). The 'Start' method initializes 'enemiesKilled' to 0 and checks if the current scene is valid. If it is, it sets the text of 'enemiesKilledText' to "Megölt ellenségek: " followed by the value of 'enemiesKilled'. The code uses Unity's PlayerPrefs to store the kill count.

```
Assets > PERSONAL > KillTxtController.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.SceneManagement;
6  using TMPro;
7
8  public class KillTxtController : MonoBehaviour
9  {
10     private int enemiesKilled;
11     private int playerHealth;
12     public TMP_Text enemiesKilledText;
13     public GameplayController gpC;
14
15     void Start() {
16         enemiesKilled = PlayerPrefs.GetInt("enemyKillCount", 0);
17
18         if (SceneManager.GetSceneByBuildIndex(2).IsValid()){
19             enemiesKilledText.text = "Megölt ellenségek: " + enemiesKilled;
20         }
21     }
22 }
```

A KillTxtController jeleníti meg a vereség képernyőjén megjelenő számot, amely a játékosnak mutatja meg az általa elpusztított ellenségeket. A GameplayController.cs „enemyKillCount” változójából kap friss értéket az aktuálisan elért pontszámról.

1.6.9 PlayerHealth.cs

Készítette: Fazekas Dávid & Patkósi Kevin



```
PlayerHealth.cs
Assets > PERSONAL > PlayerHealth.cs
1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3  using UnityEngine.UI;
4  using UnityEngine.InputSystem;
5  using System.Collections;
6  using System.Collections.Generic;
7
8  public class PlayerHealth : MonoBehaviour
9  {
10     public int maxHealth = 100; // A játékos maximum elete
11     public int currentHealth; // A játékos pillanatnyi elete
12     public HealthBar healthBar;
13
14     private void Start()
15     {
16         currentHealth = maxHealth; // Az elet maximumra állítása
17         healthBar.setMaxHealth(maxHealth);
18     }
19
20     private void OnTriggerEnter(Collider other)
21     {
22         if (other.CompareTag("Opponent"))
23         {
24             healthBar.setHealth(currentHealth);
25             int damage = Random.Range(10, 18); // Sebződés 10-17 között
26             currentHealth -= damage; // Az elterő csökkenése "damage" ertékkel
27             Debug.Log("Player took " + damage + " damage. Health: " + currentHealth);
28             if (currentHealth <= 0)
29             {
30                 Die(); // Ha a játékos elete egyenlő 0-val vagy 0 alá esik a Die metódus meghívódik
31             }
32         }
33     }
34
35     private void Die()
36     {
37         // A halál scene meghívódik amint a játékos meghal
38         SceneManager.LoadScene("Death");
39         Cursor.visible = true;
40         Cursor.lockState = CursorLockMode.None;
41         Debug.Log("DIED");
42     }
43
44     public void RestartGame()
45     {
46         Time.timeScale = 1f;
47         SceneManager.LoadScene("Encircled_map_V1"); // A jelenlegi scene újranyitása
48     }
49
50     public void GoToMainMenu()
51     {
52         Time.timeScale = 1f;
53         SceneManager.LoadScene("Main Menu"); // A fómenü scene megnyitása
54     }
55
56 }
```

A játékos életerejéért felelős script, a PlayerHealth.cs kezeli a játékos életét, a sebződést amelyet az ellenfelektől kap, valamint a vereség során történő átvezetést a vereség képernyőjére. A Scene Manager segítségével a képernyőn a játékos lehetőséget kap visszatérni a fómenübe vereség esetén, de egy új játékmenetet is kezdeményezhet.

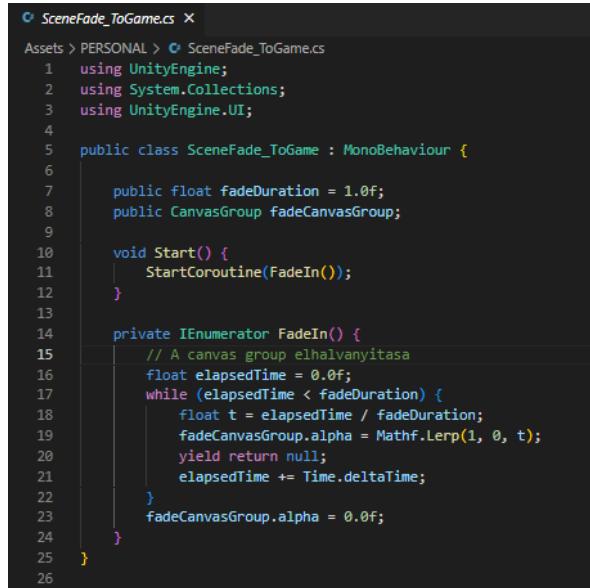
A sebződés a fejlesztői változatban konzolosan is eltárolásra kerül egyszerű visszajelzés érdekében.



[20:52:46] Player took 16 damage. Health: 84
0x00007ff61eb569cd (Unity) StackWalker::GetCurrentCallstack

1.6.10 SceneFade_ToGame.cs

Készítette: Fazekas Dávid

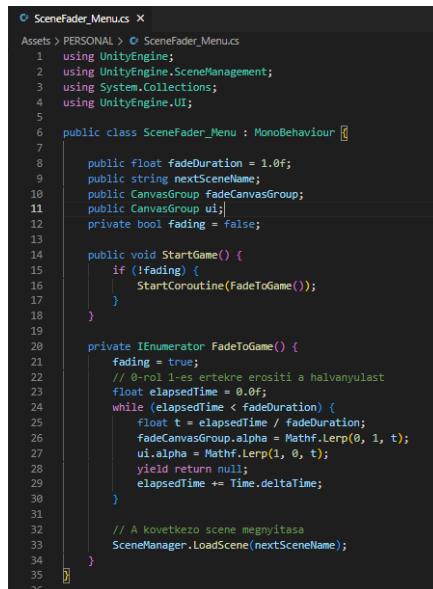


```
SceneFade_ToGame.cs
Assets > PERSONAL > SceneFade_ToGame.cs
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class SceneFade_ToGame : MonoBehaviour {
6
7      public float fadeDuration = 1.0f;
8      public CanvasGroup fadeCanvasGroup;
9
10     void Start() {
11         StartCoroutine(FadeIn());
12     }
13
14     private IEnumerator FadeIn() {
15         // A canvas group elhalványítása
16         float elapsedTime = 0.0f;
17         while (elapsedTime < fadeDuration) {
18             float t = elapsedTime / fadeDuration;
19             fadeCanvasGroup.alpha = Mathf.Lerp(1, 0, t);
20             yield return null;
21             elapsedTime += Time.deltaTime;
22         }
23         fadeCanvasGroup.alpha = 0.0f;
24     }
25 }
26
```

A játék főmenüjének minőségének növelése érdekében a jelenetek közti átvezetést egy SceneFade nevű megoldással, a képernyő elhalványulásával egészítettük ki. Ez az átvezető a menüből a játékba vezet.

1.6.11 SceneFader_Menu.cs

Készítette: Fazekas Dávid



```
SceneFader_Menu.cs
Assets > PERSONAL > SceneFader_Menu.cs
1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3  using System.Collections;
4  using UnityEngine.UI;
5
6  public class SceneFader_Menu : MonoBehaviour {
7
8      public float fadeDuration = 1.0f;
9      public string nextSceneName;
10     public CanvasGroup fadeCanvasGroup;
11     public CanvasGroup ui;
12     private bool fading = false;
13
14     public void StartGame() {
15         if (!fading) {
16             StartCoroutine(FadeToGame());
17         }
18     }
19
20     private IEnumerator FadeToGame() {
21         fading = true;
22         // 0-ról 1-ig erősen lecsökken a halványulás
23         float elapsedTime = 0.0f;
24         while (elapsedTime < fadeDuration) {
25             float t = elapsedTime / fadeDuration;
26             fadeCanvasGroup.alpha = Mathf.Lerp(0, 1, t);
27             ui.alpha = Mathf.Lerp(1, 0, t);
28             yield return null;
29             elapsedTime += Time.deltaTime;
30         }
31
32         // A következő scene megnyitása
33         SceneManager.LoadScene(nextSceneName);
34     }
35 }
36
```

A játék főmenüjének minőségének növelése érdekében a jelenetek közti átvezetést egy SceneFade nevű megoldással, a képernyő elhalványulásával egészítettük ki.

1.6.12 SoundManager.cs

Készítette: Patkósi Kevin



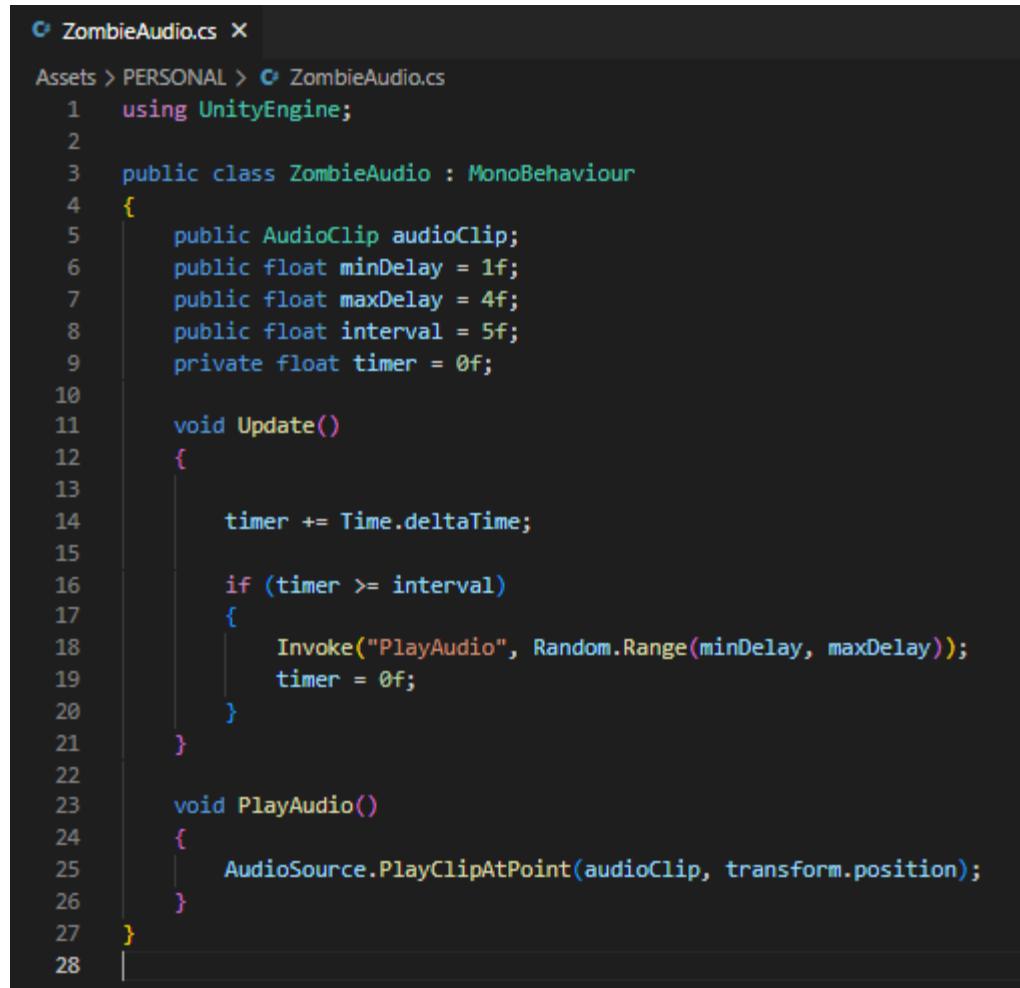
The screenshot shows a code editor window with the file "SoundManager.cs" open. The code is written in C# and defines a MonoBehaviour class named SoundManager. It includes methods for initializing the volume slider, changing the volume, loading the volume from PlayerPrefs, and saving the volume back to PlayerPrefs.

```
 1  using System.Collections;
 2  using System.Collections.Generic;
 3  using UnityEngine;
 4  using UnityEngine.UI;
 5
 6  public class SoundManager : MonoBehaviour
 7  {
 8
 9      [SerializeField] Slider volumeSlider;
10
11     void Start()
12     {
13         if(!PlayerPrefs.HasKey("musicVolume"))
14         {
15             PlayerPrefs.SetFloat("musicVolume", 1);
16             Load();
17         }
18     }
19
20     public void ChangeVolume()
21     {
22         AudioListener.volume = volumeSlider.value;
23         Save();
24     }
25
26     private void Load()
27     {
28         volumeSlider.value = PlayerPrefs.GetFloat("musicVolume");
29     }
30
31     private void Save()
32     {
33         PlayerPrefs.SetFloat("musicVolume", volumeSlider.value);
34     }
35 }
```

A játék beállításaiban megtalálható hangerőcsúszka a SoundManager.cs alapján működik, a játék hangvezérlése innen irányítható egyszerűen.

1.6.13 ZombieAudio.cs

Készítette: Fazekas Dávid



The screenshot shows a Unity code editor window with the file "ZombieAudio.cs" selected. The code is written in C# and defines a MonoBehaviour class named "ZombieAudio". The class has fields for an AudioClip, minDelay, maxDelay, interval, and a timer. It contains an Update() method that increments the timer and calls PlayAudio() if the timer reaches or exceeds the interval. The PlayAudio() method uses AudioSource.PlayClipAtPoint to play the audio at the current transform position.

```
1  using UnityEngine;
2
3  public class ZombieAudio : MonoBehaviour
4  {
5      public AudioClip audioClip;
6      public float minDelay = 1f;
7      public float maxDelay = 4f;
8      public float interval = 5f;
9      private float timer = 0f;
10
11     void Update()
12     {
13
14         timer += Time.deltaTime;
15
16         if (timer >= interval)
17         {
18             Invoke("PlayAudio", Random.Range(minDelay, maxDelay));
19             timer = 0f;
20         }
21     }
22
23     void PlayAudio()
24     {
25         AudioSource.PlayClipAtPoint(audioClip, transform.position);
26     }
27 }
28 |
```

A játék ellenfeleire szabott ZombieAudio.cs felelős az általuk kiadott hangeffektek lejátszásáért játék közben.

1.7 Jelenetek

1.7.1 Main_Menu.unity

Készítette: Patkói Kevin



A játék elindítása során a játékos fogadó képernyő, ez az animált főmenü vezeti át a játékosat a játékba és a beállításokba, valamint ide kerülnek vissza ha végeztek a játékkal.

1.7.2 Encircled_map_V1.unity

Készítette: Fazekas Dávid



A játék pályájaként szolgáló jelenet, a játékosnak itt kell a feladatait elvégeznie az ellenfelek ellen. A szétszórt növényzet és terepelemek ellenére a NavMesh útvonalkövetése megfelelően kezeli az ellenfelek vezetését a játékos irányába.

1.7.3 Win_Screen.unity

Készítette: Patkói Kevin



Ez a statikus képernyő fogadja a játékosot ha megnyerte a játékot. A vereség során előjövő képpel azonosan a játékos innen újrakezdheti a játékot, vagy visszatérhet a főmenübe.

1.7.4 Death.unity

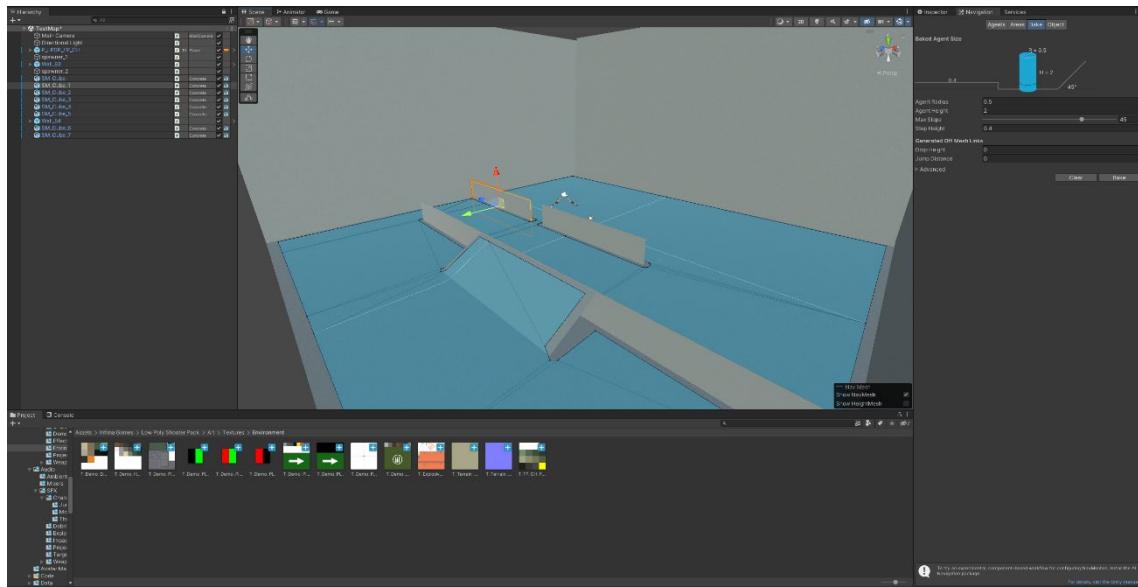
Készítette: Fazekas Dávid



A vereség során előjövő jelenet, ahonnan a játékos ismét megpróbálkozhat a játékkal, illetve visszatérhet a főmenübe.

1.7.5 TestMap.unity

Készítette: Patkósi Kevin



A NavMesh útvonaltérképező technológia megismerésének és implementálásának érdekében létrehoztunk egy egyszerű útvonalpályát, ahol az ellenfélnek három legfőbb dilemmát kell megfelelően megoldania;

- A legrövidebb út kiválasztása
- Levágó út használata (Rámpa)
- Átjárhatatlan akadályok megkerülése

Minden NavMesh-t érintő változás után egy rövid teszten esett át az AiController.cs-el ellátott ellenfél.

1.8 Továbbfejlesztési lehetőségek

A projektünk alapjánál fontosnak találtuk, hogy a továbbfejlesztés egyedüli akadálya a képzeletünk legyen. Az Unity Engine és az általunk választott témának köszönhetően ez beigazolódott, így a Trello feladatkövető táblánkon folyamatosan bővítjük a listát olyan lehetőségekkel, melyet a játékba szeretnénk megtervezni az idő haladásával.



2 Felhasználói dokumentáció

2.1 Encircled

2.1.1 Bemutatás

Az Encircled egy FPS Zombie Survival játék, mely a Golden Joystick díjas Call of Duty: World at War Zombies nevű korszakalkotó játékból inspirálódott Patkosi Kevin és Fazekas Dávid közös munkája.

A játékos célja egy elkerített és zombikkal ellepett erdőben történő túlélés az élőholtak hordái ellen. Tartsa meg a pozíóját és harcoljon, vagy támadjon futás közben, a kihívás garantált.



2.1.2 Hardver követelmények

2.5 GHz Intel Core i3 processzor / AMD FX szériajú processzor

2 GB RAM

minimum 2 GB tárhely

1GB VRAM-al rendelkező videókártya

2.1.3 Szoftver követelmények

Windows 10-11 Operációs Rendszer

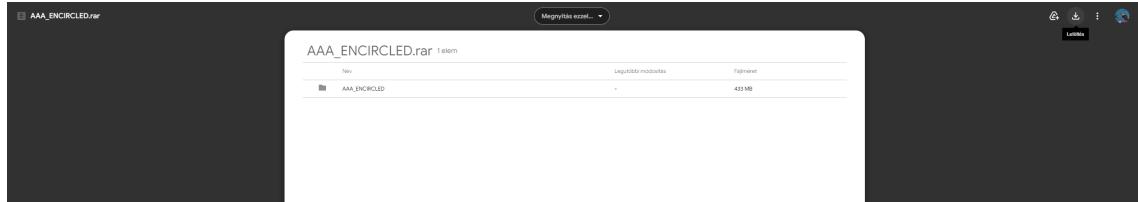
DirectX 9 Runtime Environment

Frissített grafikus vezérlőprogram

2.2 Telepítési útmutató

2.2.1 Letöltés

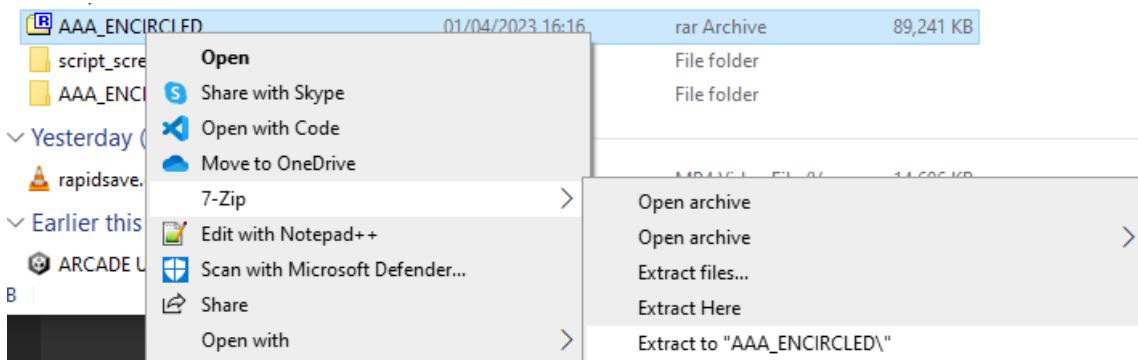
Ha nincsen még a játékból példánya, töltse azt le a számára nyújtott letöltési linkről.



A sávszélességtől eltérően változó a letöltési sebesség.

2.2.2 Kicsomagolás

A becsomagolt fájlt melyet letöltött csomagolja ki tetszőleges archiváló programmal. (WinRar, 7Zip)



2.2.3 Indítás

A kicsomagolt mappában nyissa meg az ENC_FPS nevű fájlt. Rövidesen a játék menüjében fogja találni magát.

ENC_FPS_BurstDebugInformation_DoNot...	01/04/2023 15:32	File folder
ENC_FPS_Data	01/04/2023 15:32	File folder
MonoBleedingEdge	28/03/2023 20:27	File folder
ENC_FPS	28/03/2023 20:27	Application 639 KB
UnityCrashHandler64	28/03/2023 20:27	Application 1,098 KB
UnityPlayer.dll	28/03/2023 20:27	Application exten... 28,580 KB

2.3 Játékkezelési útmutató

2.3.1 Főmenü



2.3.1.1 *Játék indítása*

A játék indítása gombbal megkezdi a játékmenetet. Első indítás során érdemes először a beállítások fül alatt a személyes preferenciákat beállítani grafika és hangerő terén.

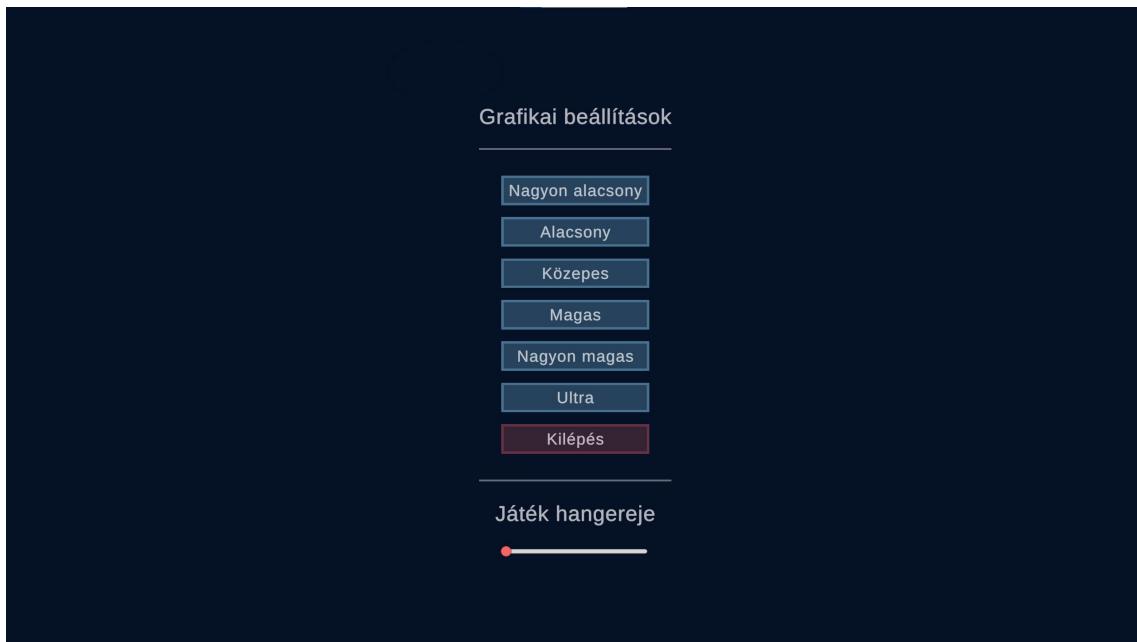
2.3.1.2 *Beállítások*

Megnyitja a játék beállításainak menüpontját. Itt szabhatja testre a grafikai komplexitást és a hangerőt.

2.3.1.3 *Kilépés*

Visszatérés az Asztalra a játék bezárásával.

2.3.2 Beállítások



2.3.2.1 *Grafikai beállítások*

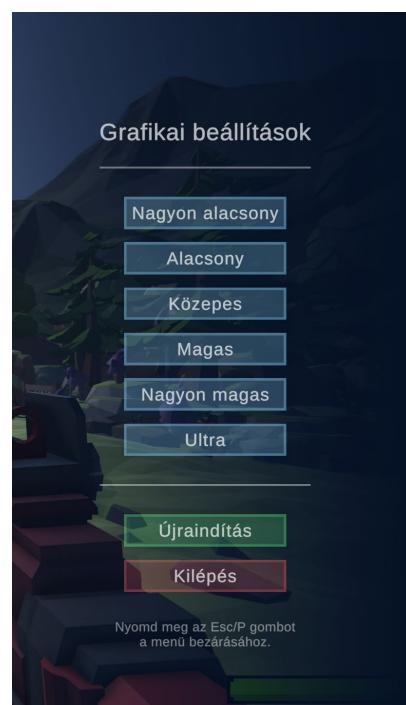
Állítsa be a számítógéphez megfelelőnek ítélt grafikai beállítást. A játékban bármikor meg tudja ezt változtatni.

2.3.2.2 *Játék hangereje*

A játék hangerejét a csúszka használatával itt tudja állítani.

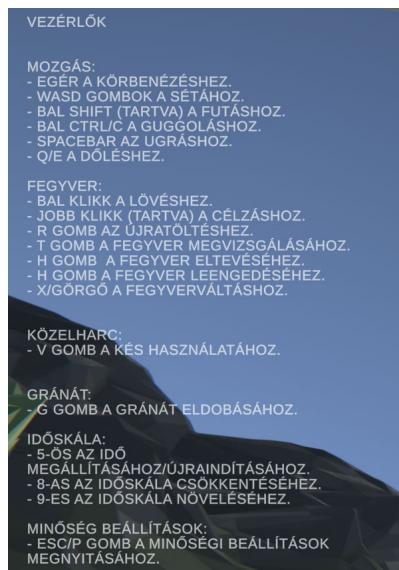
2.3.3 Játékban

Az Escape gomb megnyomásával előhozható menüben szintúgy testreszabhatja a menüből is elérhető lehetőségeket.



2.3.3.1 Irányítások

A TAB gomb lenyomásával előhozza a játék irányításait. (A Gránát lehetőség a jelenlegi verzióban nem elérhető)



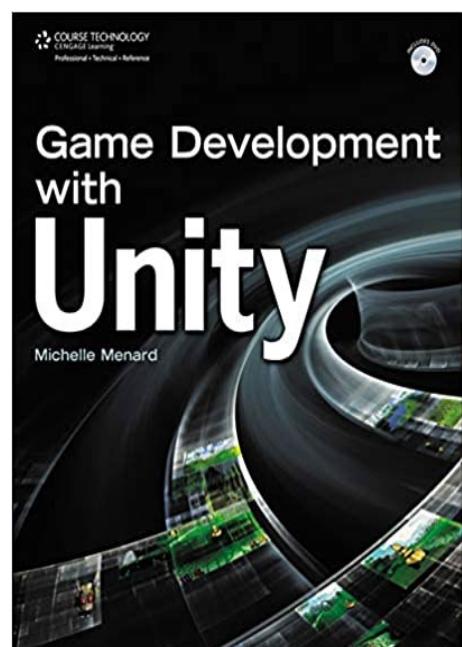
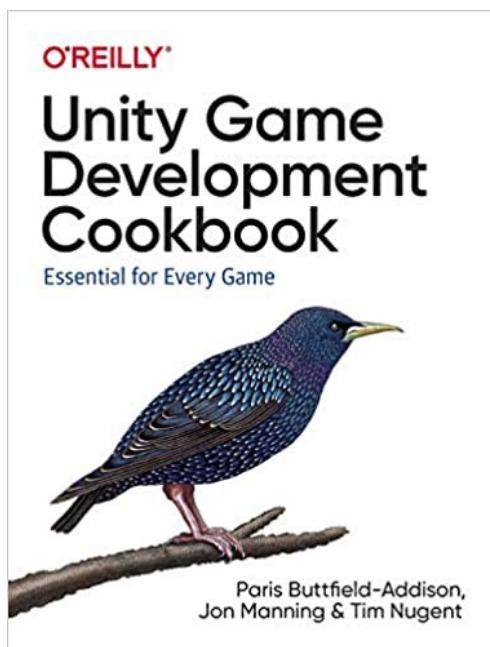
3 Irodalomjegyzék

3.1 Internetes források

- <https://assetstore.unity.com/packages/3d/vehicles/land/low-poly-cars-101798>
- <https://assetstore.unity.com/packages/3d/props/exterior/low-poly-fence-pack-61661>
- <https://assetstore.unity.com/packages/3d/environments/low-poly-rock-pack-57874>
- <https://assetstore.unity.com/packages/3d/vegetation/trees/low-poly-tree-pack-57866>
- <https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153>
- <https://assetstore.unity.com/packages/vfx/shaders/low-poly-wind-182586>
- <https://assetstore.unity.com/packages/3d/environments/landscapes/free-low-poly-nature-forest-205742>
- <https://assetstore.unity.com/packages/templates/systems/low-poly-shooter-pack-free-sample-144839>
- <https://unity.com/download>
- <https://www.youtube.com/watch?v=atCOd4o7tG4&pp=ygUUdW5pdHkgZW5naW5lIG5hdmlc2g%3D>
- https://www.youtube.com/watch?v=BLfNP4Sc_iA&t=470s&pp=ygUJSGVhbHRoQmFy
- <https://www.youtube.com/watch?v=mkErt53EEFY&pp=ygUTaGI0IGRIdGVjdGlvbiB1bmI0eQ%3D%3D>

3.2 Könyvek

<https://www.amazon.co.uk/dp/1491999152?tag=hackr-21&geniuslink=true>



<https://www.amazon.com/Game-Development-Unity-Michelle-Menard/dp/1435456580>

4 Mellékletek



A játék fejlesztése során több alkalommal is költözünk át egymáshoz hogy meggyorsítsuk a fejlesztési folyamatot. A képen is egy ilyen összeülés látható, amikor újult erőből nekiláttunk az első tesztelhető verzió megteremtésének.

Az projekt fontosabb kódjai megtalálhatóak a GitHubon

<https://github.com/Fazi2613/ENCIRCLED/tree/main/script>

A projekt teljes anyaga, beleértve a kódokat, animátorokat, textúrákat és egyéb forrásfájlokat is, elérhető a Google Driveon.

A teljes projekt:

<https://drive.google.com/drive/folders/1bezi9D1esggd1UNAZx-SrXrfPgUuWh1Q?usp=sharing>

A játszható verzió:

https://drive.google.com/drive/folders/1O4ypKpk_JYalnQkoAmlDM_YhLo4oxc?usp=sharing

