Apáczai Csere János Elméleti Líceum

Matematika-Informatika-Intenzív Angol Tagozat

Lucifer's Comeback

Készítette:

Nagy Timea

Fehérvári Zsolt

Vezetőtanár:

Geváld Júlia

Kolozsvár

2020

Felhasználói dokumentáció

ÁLTALÁNOS BEMUTATÁS

A 7 szintes játék kisebb kihívást jelent a játékosoknak mivel szükségük van némi tudásra és odafigyelésre. A játék megalkotásához inspirációt nyújtott Madách Imre: Az ember tragédiája könyve és a Daniel Defoe: Robinson Crusoe és a Lucifer sorozat. A hét szint különböző minigamekből áll, amelyek között van labirintus, labdás játék, quiz, másodfokú egyenlet és stb.

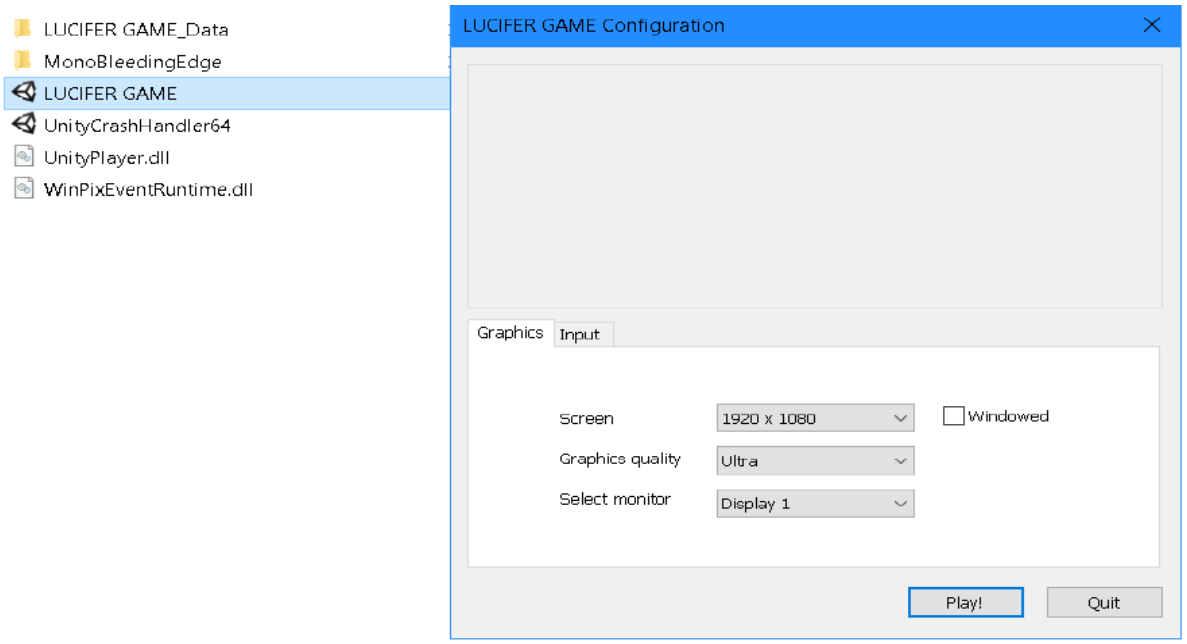
A játék célja az hogy, a karakter (Lucifer) egy lakatlan szigeten találja magát, ahonnan csak akkor juthat ki, ha teljesíti a játékban lévő hét szintet. Az utolsó szint eredménye jóslatként is tekinthető, amely bemutatja a Föld sorsát a karakter szabadulása után.

A szintek úgy változnak, hogy a karakter rá kell lépjen a teleportkristályra és így a játék bedobja a kívánt szintre. Ha a játékos végig tudja vinni a szintet, akkor a karakter automatikusan visszakerül a szigetre, közel a portálhoz.

A játék a Unity programban készült, amelynek programozási nyelve C#-on alapul, a különböző grafikus elemek és videók az Adobe programcsalád által valósultak meg.

PROGRAM INDÍTÁSA

A játék elindításához a LUCIFER GAME kell duplán kattintani, majd kiválasztjuk a kívánt beállításokat és megnyomjuk a play virtuális gombot.



A JÁTÉK IRÁNYÍTÁSA

A játék kezdése után megjelenik a képernyőn egy térkép. A térképet az M gombbal lehet eltüntetni. A karaktert az egérrel kell irányítani és a W gombbal vagy az előre nyíllal lehet mozgatni.

A játék futtatása után elindul az intro videó, amit követ a játék főmenüje, ahol két opció található a Story és a Start Game.

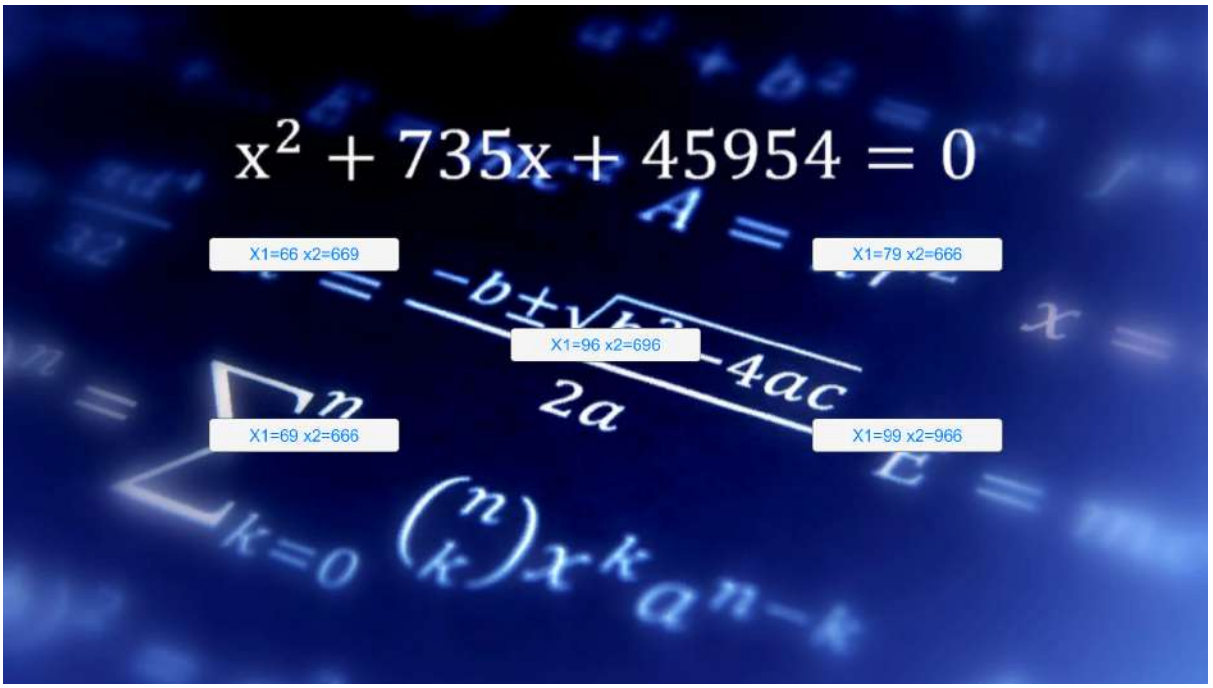
STORY

Itt olvasható a játék rövid története, majd visszatérhetünk a főmenüben Back gombbal vagy tovább kattintunk a Rules gombra, hogy olvassuk el a szabályokat.

RULES

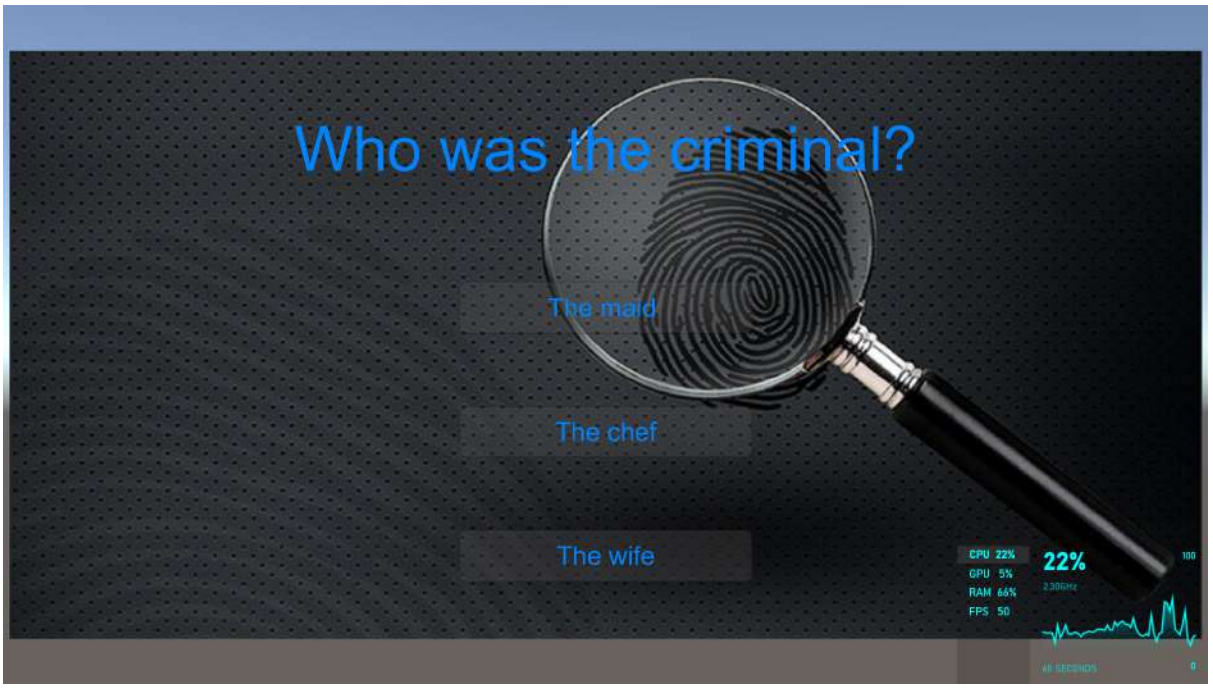
Itt megtalálható, hogy milyen gombokkal irányítható a karakter és, hogy melyik gombokkal cserélhetjük a zenét és hívhatjuk elő a térképet. Egyben tudatjuk a játékossal, hogy a küldetéseket sorrendbe kell elvégezni. Az Story gombra kattintva visszatérhetünk az előző ablakra, viszont hogyha a Back gombra kattintunk akkor visszadob minket a főmenübe.

ELSŐ SZINT



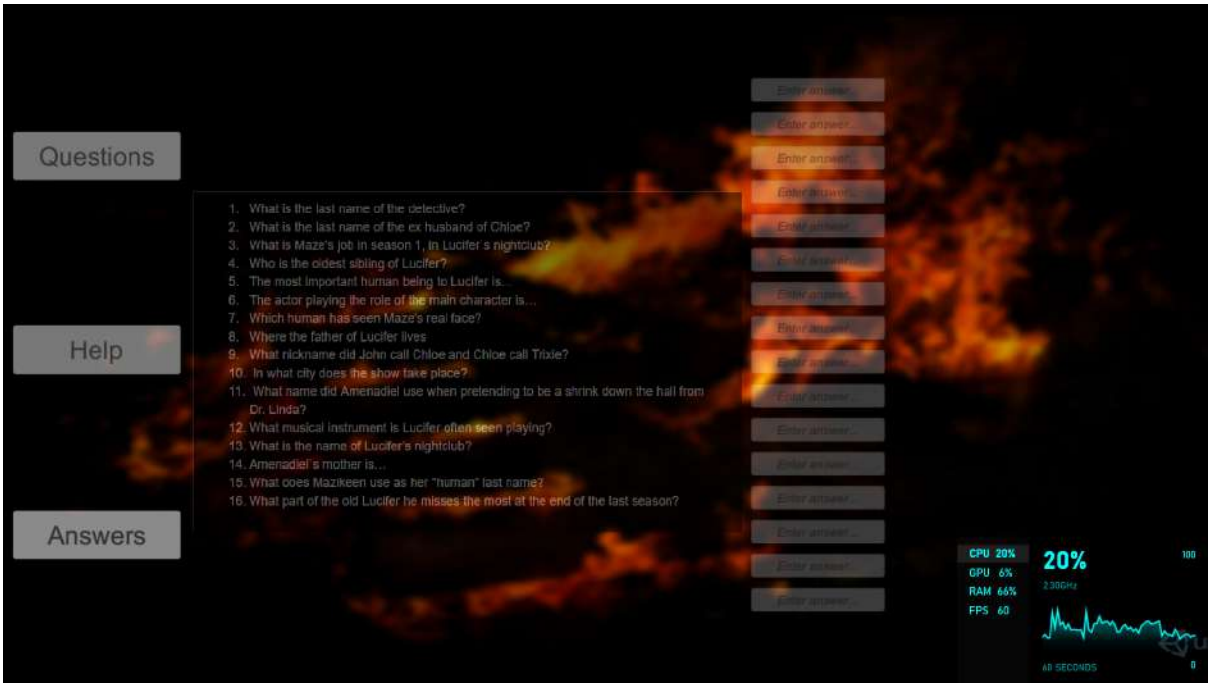
Az első szintnél az új képernyő megjelenítéséhez a kút elé kell menni a karakterrel. Az első szinten meg kell oldani egy másodfokú egyenletet (x^2 -375x+45954=0). Az egérrel lehet kiválasztani a helyes megoldást. A helyes megoldás a 69 és 666. Ez után Enter segítségével lehet visszamenni a szigetre, ahol meg kell keresni a következő missziót.

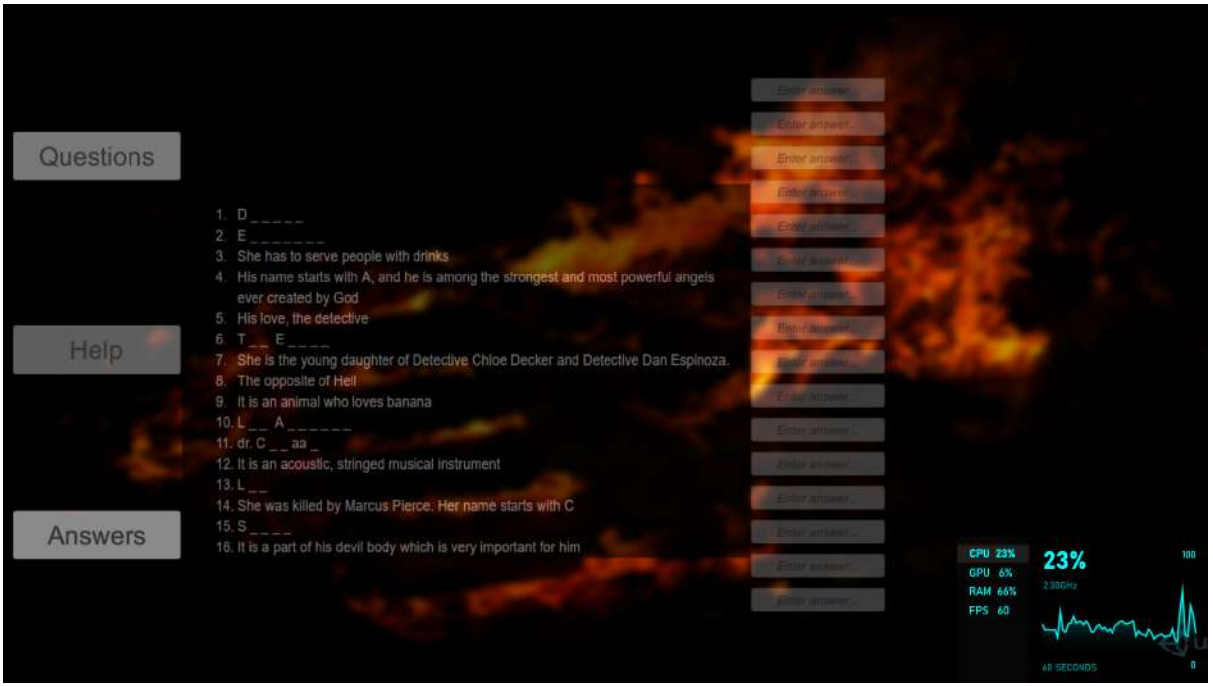
MÁSODIK SZINT



A játékos rá kell jöjjön, hogy ki volt a gyilkos, majd ki kell választania ezt három opció közül. Ha rosszat választ ki, akkor újból meg kell próbálnia ameddig nem sikerül. A szint megnyitásához a térképen megjelölt helyre kell menni és ott pontosan a sátor elé. A helyes megoldás a feleség (wife).

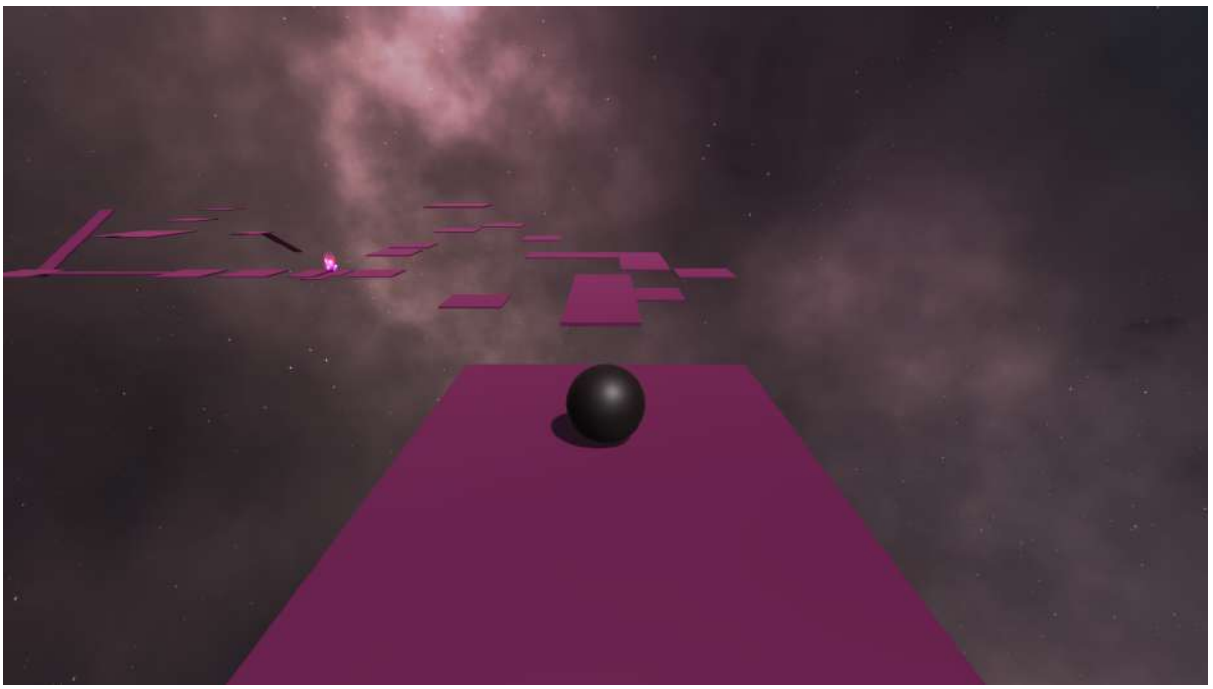
HARMADIK SZINT





A harmadik szintnél a megnyitáshoz a dokk/kikötő előtt található.
A játékosnak meg kell oldania egy keresztrejtvény szerűséget. A kérdések a Lucifer sorozaton alapulnak. Ha a Questions gombra kattintunk, akkor megjelennek a kérdések. Ha a Help gombra kattintunk, akkor kapunk némi segítséget és, hogy ha így sem sikerül megoldani egy bizonyos idő után lehetséges lesz, hogy az Answers gombra kattintva meg tudjuk tekinteni a helyes megoldásokat.
Amint kitöltöttük minden egyes kockát helyesen a játék tovább fog engedni.

NEGYEDIK SZINT



A negyedik misszió megjelenítéséhez az autó mögé kell menni. A labdát el kell juttatni A pontból B (kristály) be, úgy hogy ne essen le, ha leesik újból kell kezdeni az elejéről. A labdát a.W, S, A, D gombokkal kell irányítani és ugrani úgy lehet, hogy lenyomjuk a space-et.

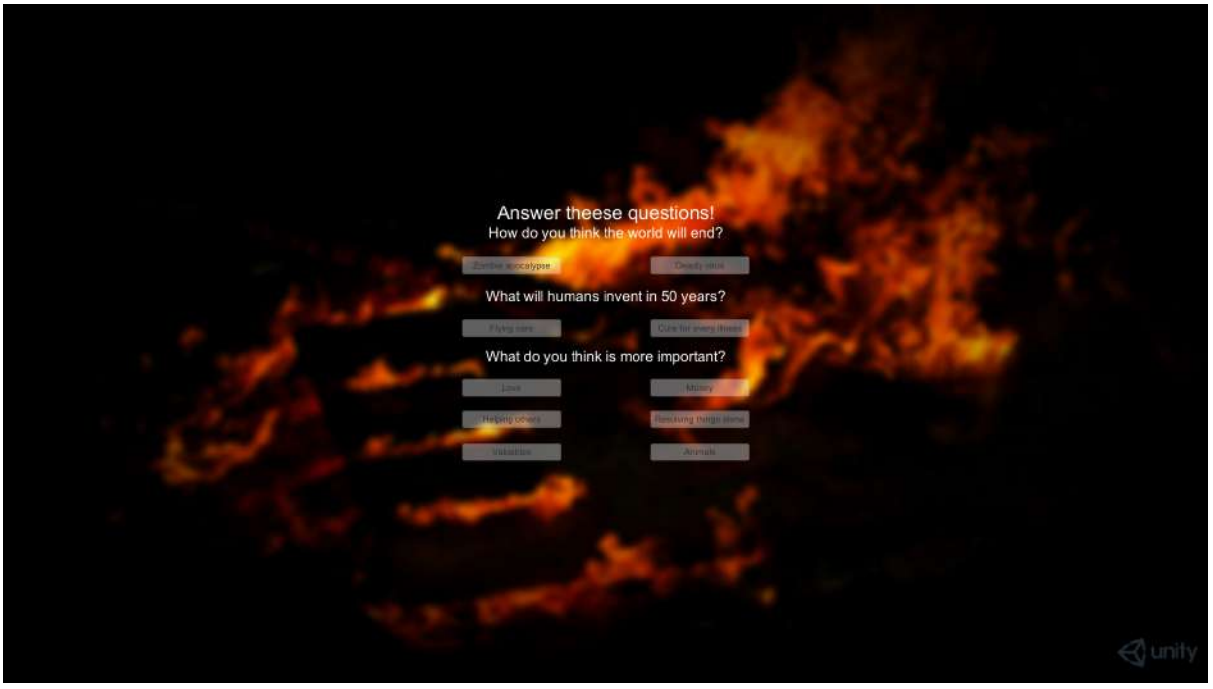
ÖTÖDIK SZINT

Az ötödik szint megnyitásához meg kell keresni az elhagyatott házat és ott a bejárathoz érve megjelenik az új szint. A játékosnak meg kell találnia a szobába nem illő tárgyakat. Összesen 9 tárgy van, ha a játékos többször kattint mint 13 akkor újból kell kezdenie a szintet. Amint kiválasztotta a 9 nem odaillő tárgyat a játék visszadobja a szigetre.

HATODIK SZINT

A hatodik szint az elhagyatott konténerek közül az egyikben található.
A játékosnak végig kell menni egy labirintusban, a játékost ugyanúgy mozgatva, mint eddig, amint megtalálta a kiutat, a játék visszafogja teleportálni a szigetre.

HETEDIK SZINT



Az utolsó szint a világítótorony bejáratánál van elhelyezve, mely egy pár kérdésre alapul. A válaszok alapján a játék el fogja dönteni, hogy milyen képsor jelenik meg, azaz egy szöveg amely leírja a világ hogy fogja végezni ha kiszabadul Lucifer a szigetről és a pokolból.

Programozói dokumentáció

A JÁTÉKMOTOR

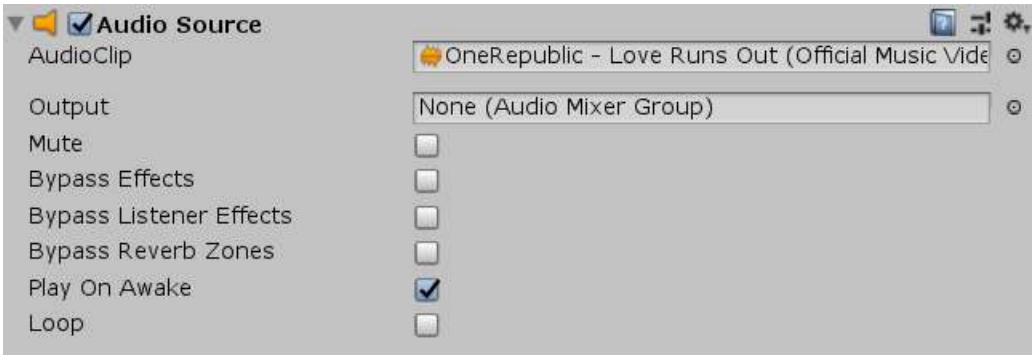
A Unity egy videójáték-motor, amelyet a Unity Technologies fejleszt. A Unity segítségével háromdimenziós illetve kétdimenziós videójátékokat, ezen kívül egyéb interaktív jellegű tartalmakat lehet létrehozni. Többek között előnye, hogy a szoftver képes nagyméretű adatbázisokat kezelni, kihasználni a kölcsönhatások és animációk képességeit, előre kiszámított vagy valós idejű világítást biztosítani. A Unity szoftverrel való videojáték-készítés lehetséges Microsoft Windows vagy Mac OS X operációs rendszerek használatával, a játékmotor segítségével létrehozott játékok pedig futtathatók Windows, Mac OS X, Xbox 360, PlayStation 3, Wii, iPad, iPhone vagy Android alatt.

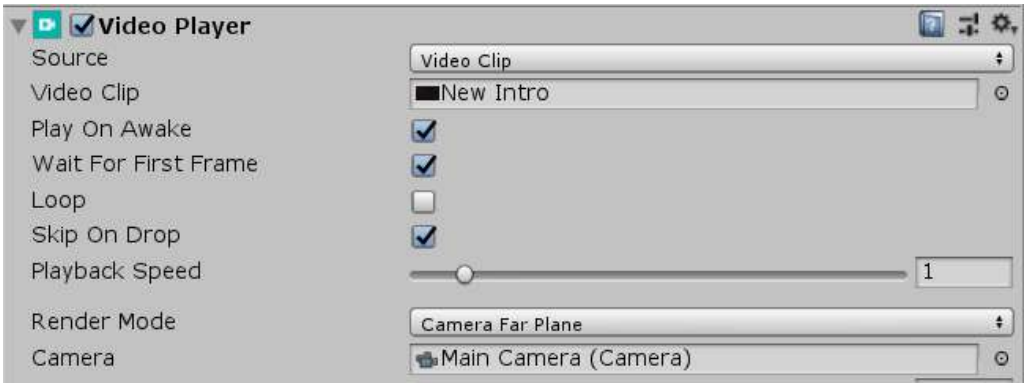
A legtöbbet használt programozási nyelv a C#, de emellett még a C vagy C++ programozási nyelv is használható.

FONTOSABB KÓDRÉSZLETEK BEMUTATÁSA

`Az intro video`

Video komponens és audio komponens segítségével





Az intro video utáni játék betöltése

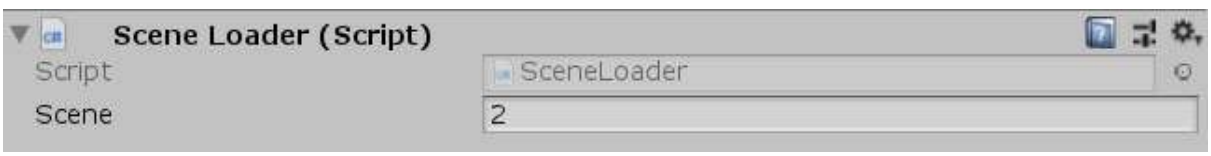
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
//this script plays the "Main" scene after the intro video
//the length of the intro is 10 seconds, so the "Main" scene will be played
//10 seconds after starting the game:)
public class IntroTimer : MonoBehaviour
{
    public float timer=10f;
    void Update()
    {
        timer -= Time.deltaTime;
        if(timer<=0)
        {
            SceneManager.LoadScene("Main");
        }
    }
}
```

A jelenetek betöltése

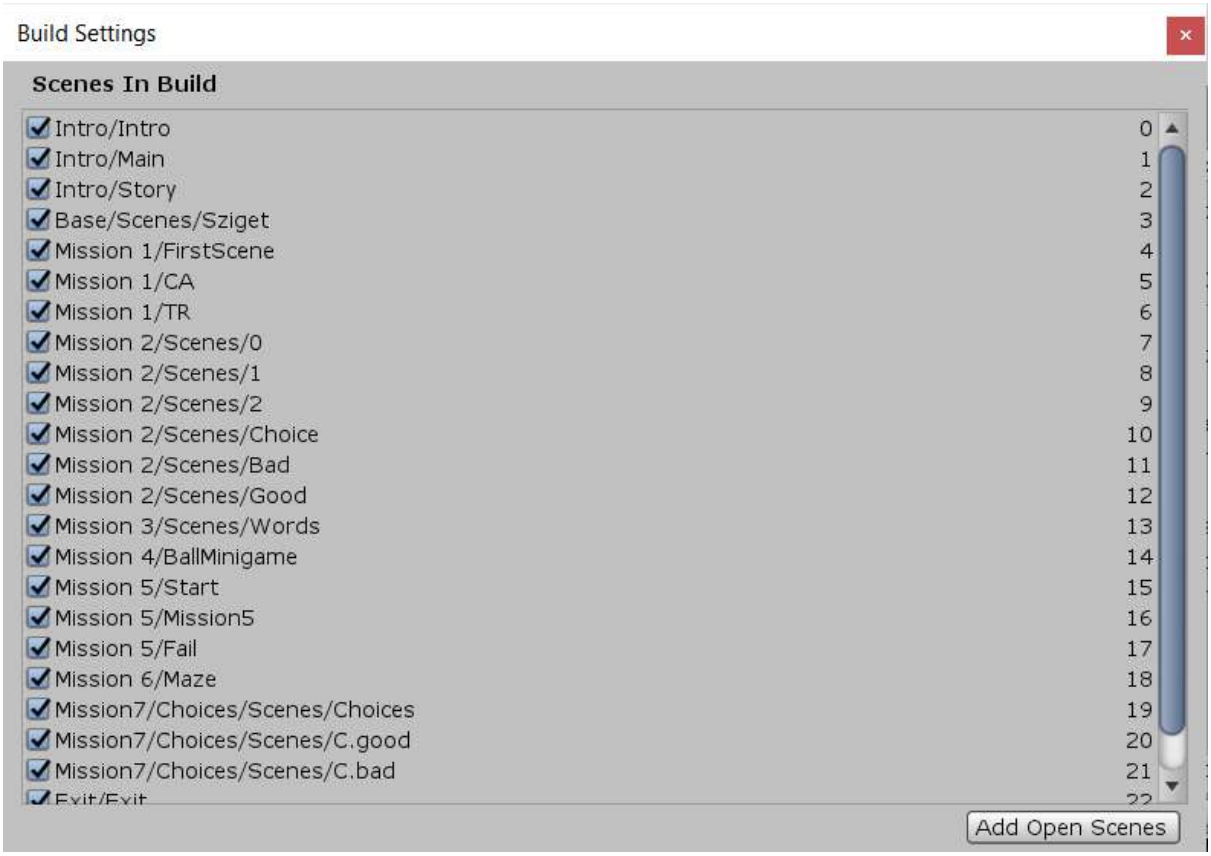
A kód

```
public class SceneLoader : MonoBehaviour
{
    public int scene;
    public void Change(int scene)
    {
        SceneManager.LoadScene(scene);
    }
}
```

Így használtuk a kódot Unityben
A scene mezőbe beírjuk a betölteni kívánt jelenet indexét
A kód egy komponensként hozzáadható a jelenetben szereplő bármely objektumhoz.

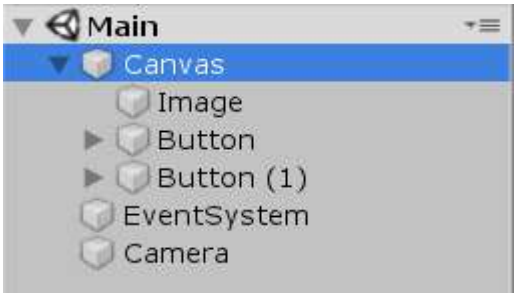


A jelenetek indexe megtalálható a File -> Build Settings gombokra kattintva

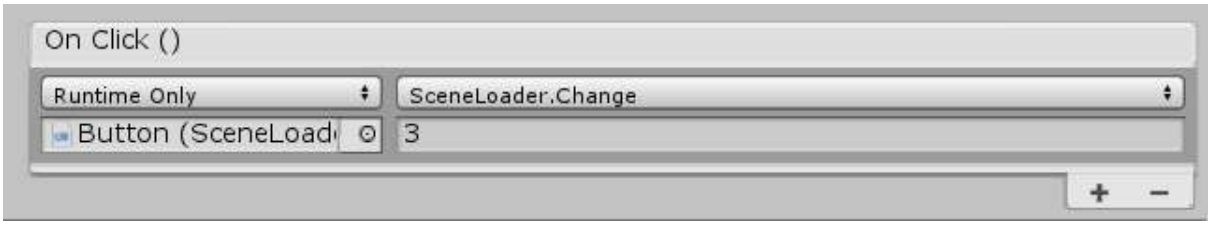


User Interface

A Unity UI segítségével hoztuk létre a Menüt. Először is egy Canvas-t adtunk hozzá a jelenethez, majd a szükséges gombokat. A logót is külön adtuk hozzá.



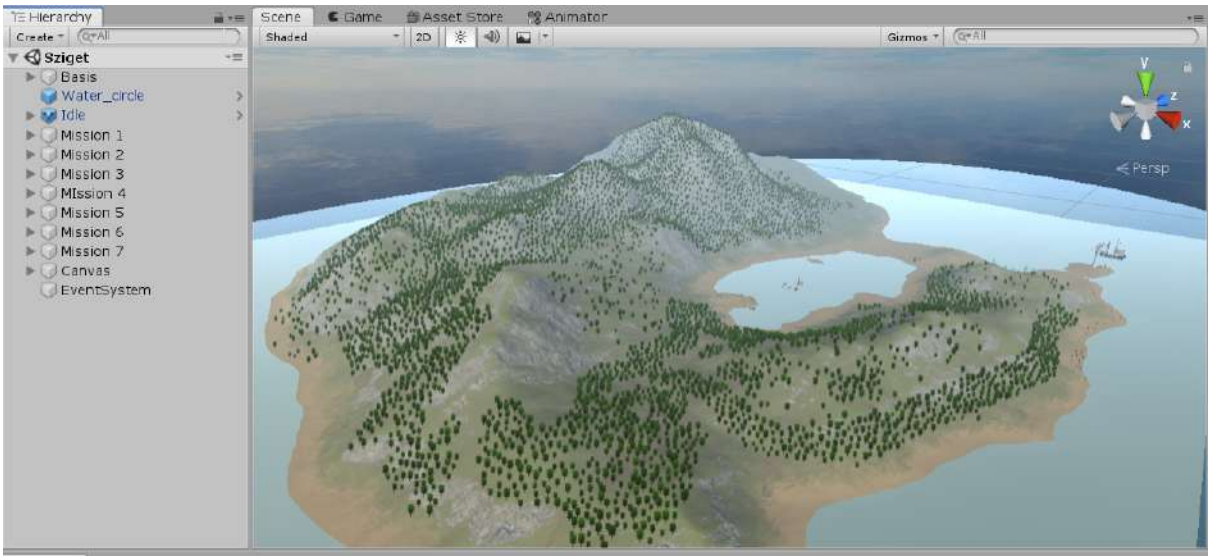
A gombokra kattintva megjelenik a kiválasztott jelenet, a Change függvény segítségével.



A Change függvény a SceneLoader scriptben található.

A sziget

Így néz ki a sziget felülről.



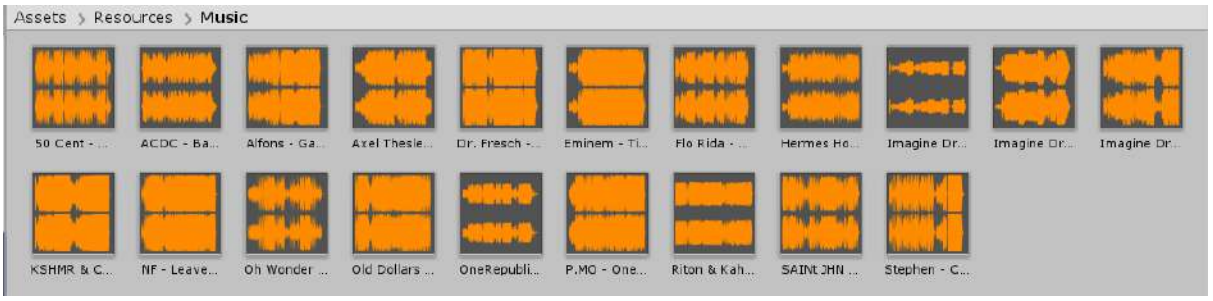
A térkép megjelenítése

```
5   public class MapScript : MonoBehaviour
6   {
7       public GameObject map;
8       void Start()
9       {
10          map.SetActive(true);
11      }
12      void Update()
13      {
14          if(Input.GetKeyDown(KeyCode.M))
15          {
16              if(!map.activeSelf)
17              {
18                  map.SetActive(true);
19              }
20              else
21              if(map.activeSelf)
22              {
23                  map.SetActive(false);
24              }
25          }
26      }
27  }
```

Az alábbi kód hatására, a játék indításakor megjelenik a térkép. Az M gomb megnyomásával eltüntethető illetve megjeleníthető a térkép.

A zene

A Resources >> Music folderbe találhatóak a zenék.



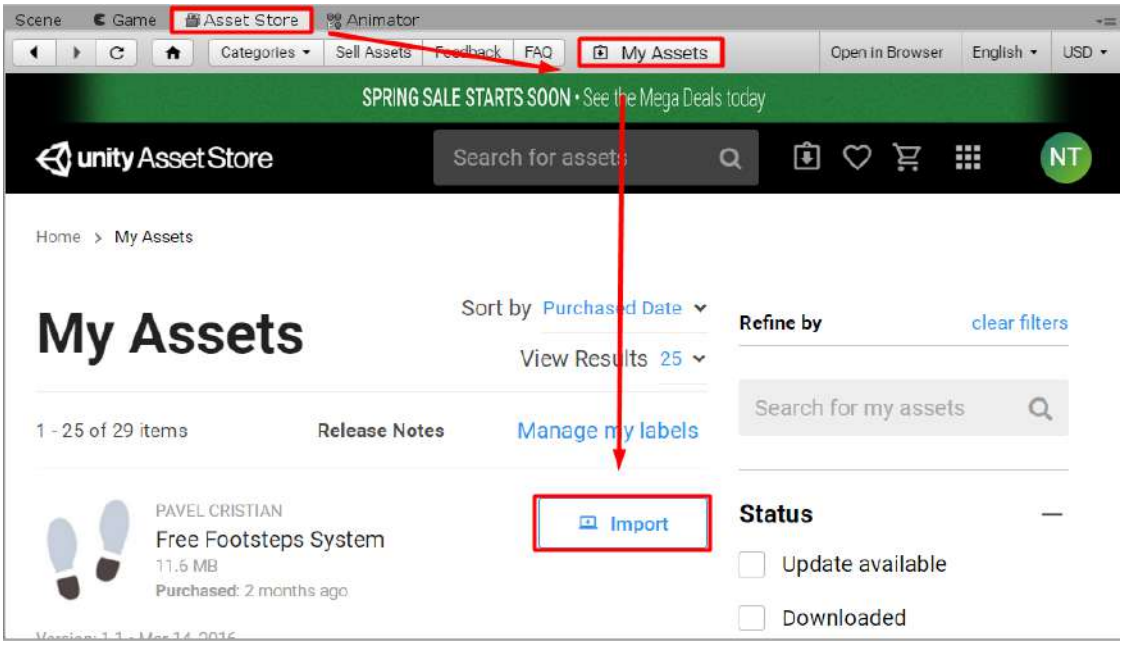
Az N gombra klikkelve ki lehet cserélni a zenét, ha az a játékosnak nem tetszik, vagy már megunta.

```
public class MusicPlayer : MonoBehaviour
{
    Object[] myMusic;

    void Awake ()
    {
        myMusic = Resources.LoadAll("Music",typeof(AudioClip));
    }

    void Start ()
    {
        GetComponent<AudioSource>().Play();
    }

    void Update ()
    {
        if(Input.GetKeyDown(KeyCode.N))
            playRandomMusic();
        if(!GetComponent<AudioSource>().isPlaying)
            playRandomMusic();
    }

    void playRandomMusic()
    {
        GetComponent<AudioSource>().clip = myMusic[Random.Range(0,myMusic.Length)] as AudioClip;
        GetComponent<AudioSource>().Play();
    }
}
```

Asseetek

Az Objects folderben tárolunk minden olyan Asseet-et amit letöltöttünk a Unity Asseet Storeból.

A különböző Asseetokat így lehet letölteni

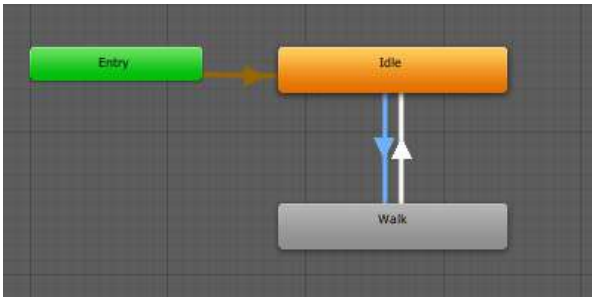


A karakter

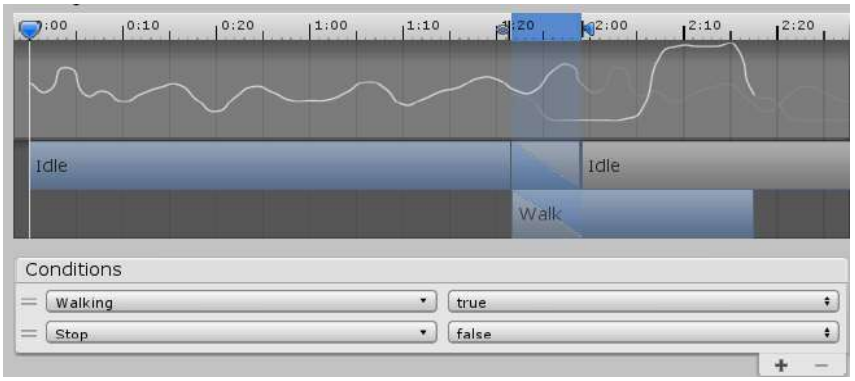
A karakterünk Adobe Fuse-ban volt készítve, majd Mixamo-ban animálva.

A karakter animálása

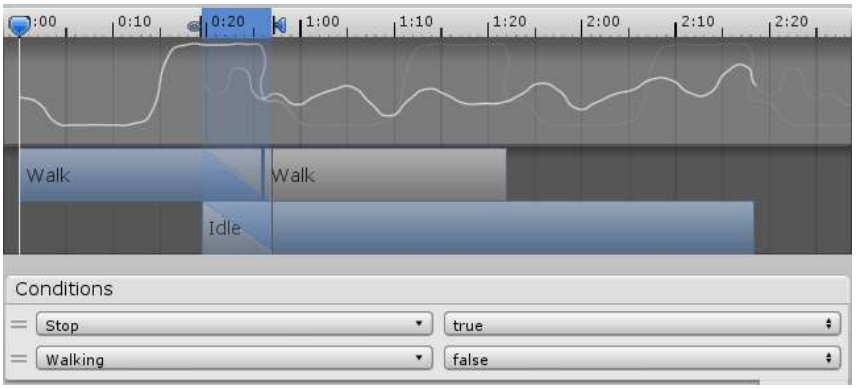
A karakter feje mögött található a kamera, a lábainál pedig a lépteinek hangja.
Így néz ki az animátor:



Váltás állásból sétálásba



Sétálásból állásba

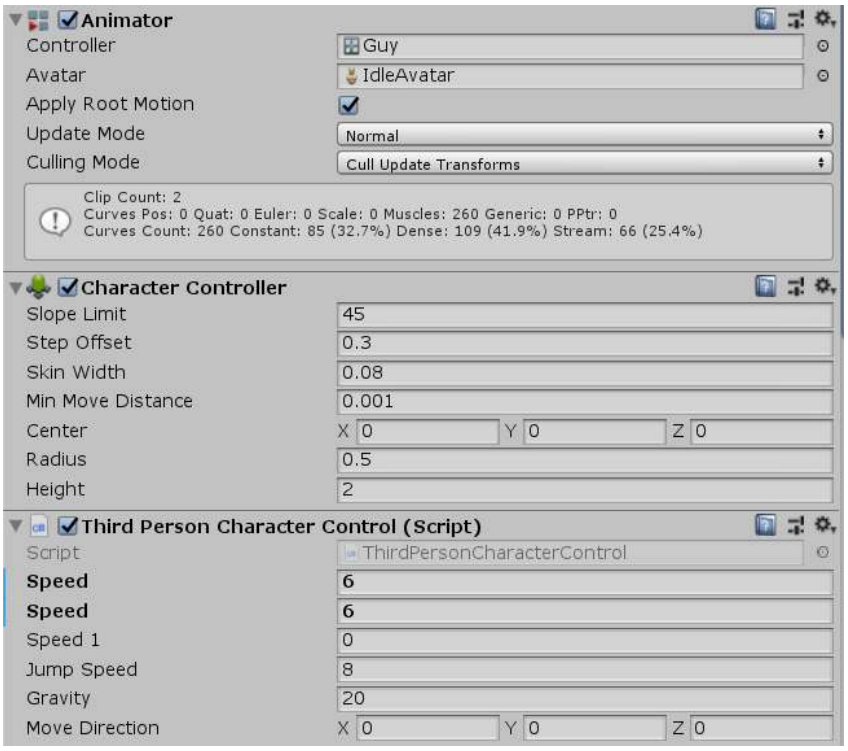


A kódrészlet a “Walking” és “Stop” logikai változók állítgatására a “ThirdPersonCharacterControl” scriptből. Ebbe a kódrészletben látszik még a karakter mozgásához kapcsolódó hang állítása is. A karakternek van egy audiokomponense ami egy járás hangot tartalmaz.

```
if (Input.GetKeyDown(KeyCode.W) || Input.GetKeyDown(KeyCode.UpArrow))
{
    if(!audioData.isPlaying)
        audioData.Play(0);
    animIdle.SetBool("Walking", true);
    animIdle.SetBool("Stop", false);
    audioData.UnPause();
}

if (Input.GetKeyUp(KeyCode.W) ||  Input.GetKeyUp(KeyCode.UpArrow))
{
    animIdle.SetBool("Walking", false);
    animIdle.SetBool("Stop", true);
    audioData.Pause();
}
```

A karakter audiokomponense:

Más komponensek, amik szükségesek a karakter mozgatásához:



A karakter mozgása

A karakter irányítása a “ThirdPersonCharacterControl” scriptben történik.

```
if (controller.isGrounded)
{
    moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));
    moveDirection = transform.TransformDirection(moveDirection);
    moveDirection = moveDirection * speed;
}
```

```
moveDirection.y = moveDirection.y - (gravity * Time.deltaTime);
controller.Move(moveDirection * Time.deltaTime);
```

A Shift billentyű nyomva tartásával a karakter gyorsabban fog mozogni. A használt kódrészlet szintén a “ThirdPersonCharacterControl” scriptben található.

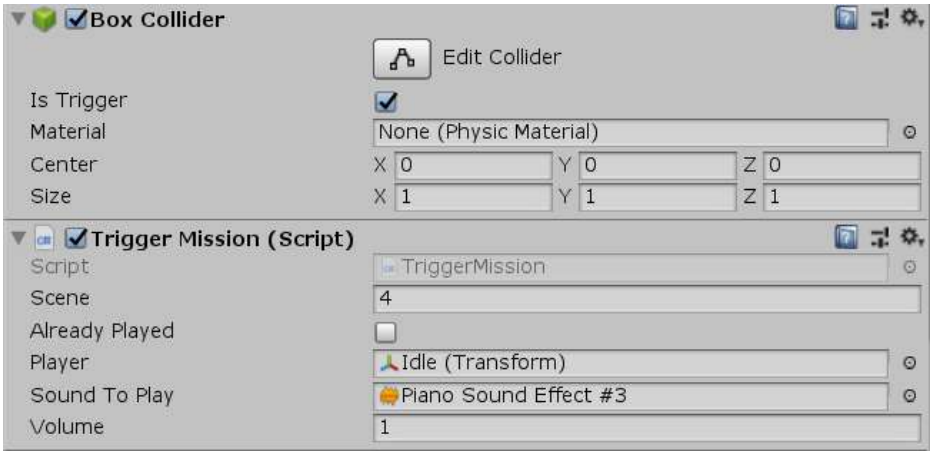
```
if(Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift))
    speed = speed1;
else
    speed = Speed;
```

A portálok

A portálokat arra használjuk, hogy rajtuk keresztül történjen az átváltás más más jelenetekre. A használt portálok kristály alakúak. A program megírásában minden portál mögé elhelyeztünk egy átlátszó kockát amit triggernek használtunk. A trigger hatására a játék átvált új jelenetre és egy hang játszódik le, hogy a játék élvezetesebb legyen.

```
public class Trigger : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        SceneManager.LoadScene(0);
    }
}
```

Az átlátszó kockának a komponensei:



A portálok hangja és a cursor állítása:

```
IEnumerator Wait1Second()
{
    yield return new WaitForSeconds(1);
    SceneManager.LoadScene(scene);
}

void OnTriggerEnter(Collider other)
{
    player.position = new Vector3(ThirdPersonCharacterControl.x ,
     ThirdPersonCharacterControl.y , ThirdPersonCharacterControl.z );
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
    if (!alreadyPlayed)
    {
        audio.PlayOneShot(SoundToPlay, Volume);
        alreadyPlayed = true;
    }
    StartCoroutine(Wait1Second());
    alreadyPlayed = false;
}
```

A karakter koordinátái

Minden szint után a karaktert visszahelyezzük a Szigetre, ezért meg kellett őrizzük a koordinátáit, hogy a portálhoz közel tegyük vissza a karaktert, de ne is rá a portálra.
A karakter koordinátáinak megőrzése, szintén a “ThirdPersonCharacterControl” scriptben történik:

```
if (SceneManager.GetActiveScene().name == "Sziget")
{
    x = transform.position.x;
    y = transform.position.y;
    z = transform.position.z;
}
```

Ezekben az esetekben a “BackToPlay” scriptet használtuk amelyben megadjuk a karakternek az új koordinátáit, amelyek kicsit eltérnek a legutóbb észlelt koordinátáitól, annak érdekében, hogy a karakter ne kerüljön újból a portálra.

```
x = ThirdPersonCharacterControl.x + 15;
y = ThirdPersonCharacterControl.y + 50;
z = ThirdPersonCharacterControl.z + 15;

SceneManager.LoadScene("Sziget");
```

Kilépés

A játék során bármikor ki lehet lépni a játékból az esc billentyű segítségével. Ez után megjelenik a képernyőn egy gomb, amit megnyomva kiléphetünk a játékból.

```
if(Input.GetKeyDown(KeyCode.Escape))
{
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
    SceneManager.LoadScene(22);
}
```

A 22.es jelenet az a jelenet, amelyben meg van kérdezve, ha biztosan ki akar lépni a játékos. Ha megnyomja az igen gombot, akkor a quit alprogram lesz meghivva.

```
public void quit()
{
    Application.Quit();
}
```

NEHEZEBB MISSZIÓK

Mission 3

A harmadik missziót IField segítségével oldottuk meg. Ez a Unity által létrehozott segédeszköz, mely arra szolgál, hogy a játékos bele írhat dolgokat. 16 IFiled van, mindegyikhez jár külön két kép, egy pipa és egy x, melyek a szó beírása után jelennek meg, aszerint, hogy a beírt szó helyes vagy sem. Ezt úgy oldottuk meg, hogy a két képet elhelyeztük, majd kikapcsoltuk a láthatóságukat egészen addig, ameddig az adott IFieldben entert nyomunk. Ekkor ellenőrzi a beírt szót (szavakat) és bekapcsolja a láthatóságát a megfelelő képnek. Itt található a kód hozzá.

```
void Start()
{
    img1.SetActive(false);
    img2.SetActive(false);
}

void Update()
{

    if(img1.activeSelf)
       img1.SetActive(true);
    else
       img1.SetActive(false);
    if(img2.activeSelf)
       img2.SetActive(true);
    else
       img2.SetActive(false);
    if (Input.GetKeyDown (KeyCode.Return))
    {
        checkInput ();
    }
}

public void checkInput()
{
    if (input == iField.text)
    {
        Debug.Log("Correct!");
        img1.SetActive(false);
        img2.SetActive(true);
    }
    else
    {
        Debug.Log("Incorrect!");
        img1.SetActive(true);
        img2.SetActive(false);
    }
}
```

Az Answers gomb csak egy adott idő után lesz kattintható azért, hogy a játékos gondolkodjon megoldani a kérdéseket. A használt kód:

```
void Start()
{
    btn.interactable = false;
}
void Update()
{
    timer -= Time.deltaTime;
    if (timer <= 0)
        btn.interactable = true;
}
```

A szint végén, amikor az összes beírt válasz helyes, megjelenik egy Correct üzenet és egy továbblépési gomb. Ezt kód szempontjából úgy oldottuk meg, hogy akkor jelenítettük meg az üzenetet és a gombot, amikor mind a 16 pipa látható volt. Az ehhez felhasznált kód egy hosszú if feltételből áll és két parancsból:


```
image.SetActive(true);
btn.SetActive(true);
```

Mission 4

Ebben a szintben a labda kinematikus platformokon gurul. Ha leesik, visszakerül a kezdőpozícióba. Ehhez a platformok alá elhelyeztünk egy nagy, átlátszó platformot amit tiggerként használtunk.

```
5   public class PutBack : MonoBehaviour
6   {
7       public GameObject obj ;
8       private Vector3 moveDirection = Vector3.zero;
9       void OnTriggerEnter(Collider other)
10      {
11          obj.transform.position = new Vector3( 0, 5, 0);
12      }
13  }
```

A labda rigidbodyként szerepel. A mozgatása egyszerű. A moveforward és moverightleft változókba pillanatnyilag megőrizzük a nyilakkal vagy A,W,S,D gombokkal megadott irányokat. Ha a labda a levegőben van, akkor kevesebbet mozgatjuk, mintha a platformokon lenne. Így látványosabb és valóságszerűbb a labda mozgása. A Space gombbal változtatjuk a labda y (felfelé) koordinátáit.

```
17      private void Update()
18      {
19          float moveforward = Input.GetAxis("Vertical"); ///up and down
20          float moverightleft = Input.GetAxis("Horizontal"); ///right and left
21          if(!onGround)
22              ApplyInput(moveforward/2, moverightleft/2);
23          else
24              ApplyInput(moveforward, moverightleft);
25          if(Input.GetKeyDown("space") && onGround)
26          {
27              rigidBody.AddForce(Vector3.up*jumpHeight, ForceMode.Impulse);
28              onGround = false;
29          }
30      }
31
32      void OnCollisionEnter(Collision collision)
33      {
34          onGround = true;
35      }
36
37      private void ApplyInput(float move1, float move2)
38      {
39          if(move1 != 0)
40          {
41              transform.Translate(Vector3.forward * move1 * moveSpeed);
42          }
43          if (move2 != 0)
44          {
45              transform.Translate(Vector3.right * move2 * moveSpeed);
46          }
47      }
```

Mission 5

A dolgok eltüntetéséhez használt kódrészlet:

```
void OnMouseDown()
{
    nameofobj = gameObject.name;
    Destroy(gameObject);
    Destroy(objnametext);
    if (objnametext != null && gameObject.activeInHierarchy)
    {
        remainItems--;
        Debug.Log(remainItems);
    }
}
void Update()
{
    if (remainItems == 0)
        BackToPlay.BACK();
}
```

A klikkek számlálása:

```
void Update ()
{
    if (Input.GetKeyDown (mouseclick))
        totalclicks += 1;

    if (totalclicks > 13)
    {
        SceneManager.LoadScene("Fail");
        totalclicks = 0;
    }
}
```

Mission 7

A jó és rossz válaszokat megszámoljuk, majd ezek összehasonlításával jelenik meg a játék végén alkotott világ.

```
29      public void AddGoodanswer()
30      {
31          x++;
32      }
33      public void AddBadanswer()
34      {
35          y++;
36      }
37      void Update()
38      {
39          if(x+y>=3)
40              LoadnextScene();
41      }
42      public void LoadnextScene()
43      {
44          if(x > y) ///good answers
45              SceneManager.LoadScene(20);
46          else ///bad answers
47              SceneManager.LoadScene(21);
48      }
```

Továbbfejlesztési ötletek

A játékot szeretnénk tovább fejleszteni egy pontozási rendszerrel, hogy ha több játékos játszik kiderüljön ki a jobbik. Be szeretnénk vezetni, hogy számítson az idő, amennyit a játékos játszott, tehát minél hamarabb kelljen befejezni a játékot. Ezenkívül még szeretnénk kialakítani egy olyan algoritmust amely a szintek sorrendjét befolyásolja. Utolsó sorban pedig azt hogy a játékos tudja kiválasztani majd a saját karakterét, a játékban meg legyen adva neki több előre elkészített avatar amik közül majd választhat.

Tartalomjegyzék

Általános Bemutatás 1
Program Indítása 1
A Játék Irányítása 2
Story 3
Rules 3
Első Szint 4
Második Szint 5
Harmadik Szint 6
Negyedik Szint 7
Ötödik Szint 8
Hatodik Szint 9
Hetedik Szint 10
Az Intro Video 12
Az Intro Video Utáni Játék Betöltése 12
A Jelenetek Betöltése 13
User Interface 14
A Sziget 14
A Térkép Megjelenítése 15
A Zene 15
Asseetek 16
A Karakter 16
A Karakter Animálása 16
A Karakter Mozgása 18
A Portálok 19
A Karakter Koordinátái 20
Kilépés 20
Mission 3 21
Mission 4 22
Mission 5 23
Mission 7 23
Továbbfejlesztési Ötletek
23