INSTITUT ZA MATEMATIKU I INFORMATIKU

PRIRODNO-MATEMATIČKI FAKULTET

UNIVERZITET U KRAGUJEVCU

SEMINARSKI RAD IZ OBRAZOVNOG SOFTVERA 1

„NOĆNA AVANTURA“

STUDENT:

Ilija Nikolić 203/2020

PROFESOR:

dr Tatjana Stojanović

Sadržaj

UVOD

Opis

Noćna avantura je zanimljiva igra koja kombinuje elemente kviza i istraživanja mračne mape. Igrači preuzimaju ulogu istraživača koji istražuje misterioznu mapu noću. Na svakom koraku, igrači će se suočiti s pitanjima iz različitih područja znanja ali uglavnom iz oblasti matematike. Kvizovi unutar igre će se aktivirati kod određenih objekata na mapi kao sto su kuće ili neke druge stvari. Međutim kako bi se kviz pokrenuo potrebno je uneti "key" koji se zapravo nalazi u blizini tog kviza. Kada igrač reši sve kvizove, njegovo vreme će se upisati u scoreboard i tako ćemo imati sve igrače rangirane po vremenu prolaska kroz mapu.

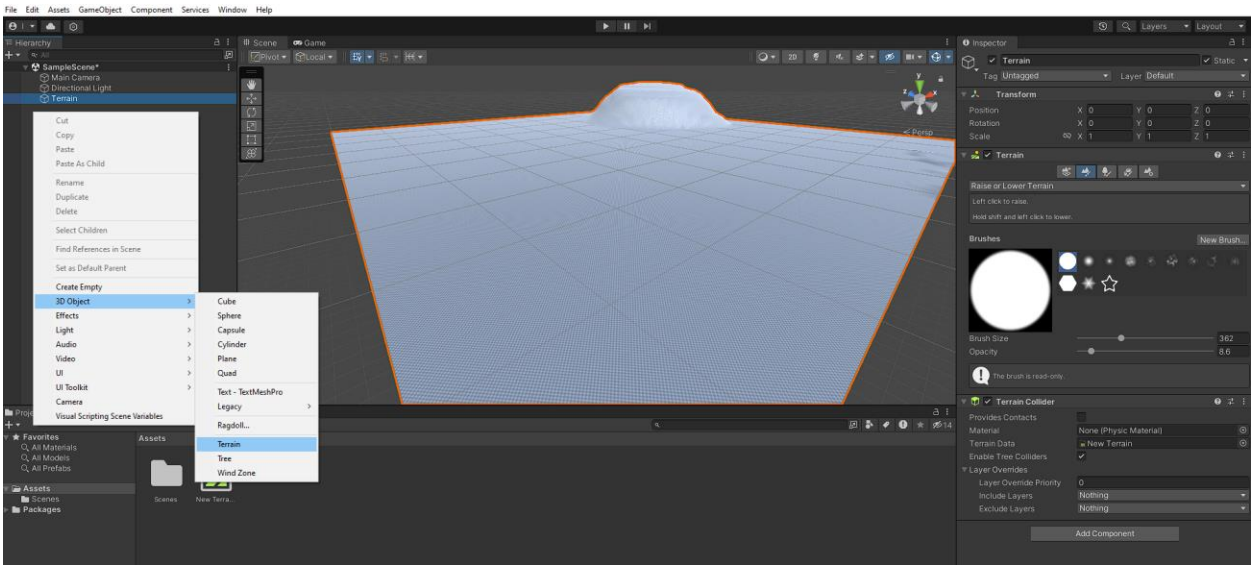
Tehnologije

Igra ce biti izrađena u unity okruženju. Kodiranje ce biti u programskom jeziku C#. Pitanja na koja će moći igrači odgovarati biće unutar baze podataka, vršiće se konekcija na bazu, kako bi mogli dobiti svaki put random pitanje. Što se tiče grafičkog izgleda igrice, koriste se dostupni assets unutar unity assets shop-a.

Izgled

KREIRANJE MAPE

Kako bi kreirali mapu, prvo je potrebno kreirati prazan teren koji ćemo pomoću alata i brojnih assets-a transformisati u željeni izgled. Unity pruža veliki broj mogućnost za izradu igrica.



Nakon što željeno uredimo mapu pomoću unity alati, grafičke elemente možemo preuzeti na zvaničnom unity assets store sajtu.

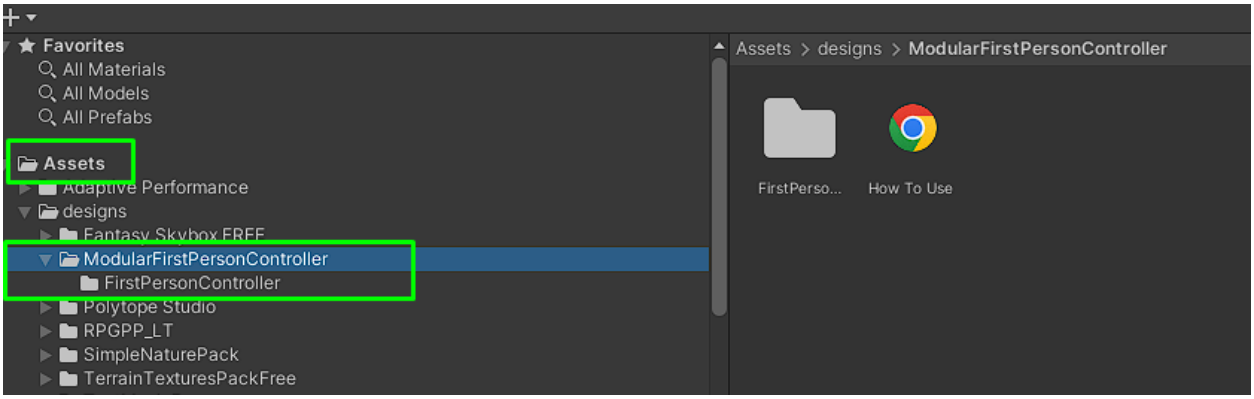
*Mnoge stvari se na slici iznad ne vide, razlog je povećanje FPS u samoj igrici, odnosno neki elementi u dalji se ne renderuju.

KODIRANJE

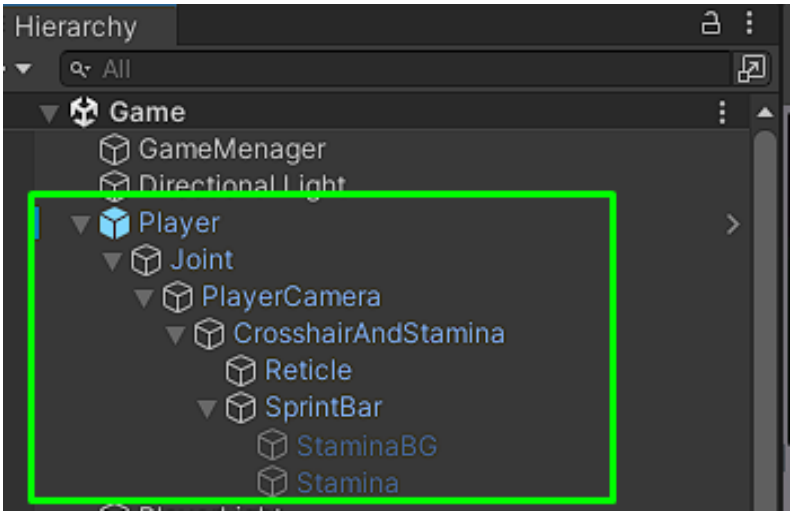
Kretanje igrača

Na unity assets store postoji idealan assets pomoću kog uopšte nije potrebno kodirati samo kretanje igrača, već je dovoljno samo skinuti i ubaciti u projekat. Sama skripta sadrži veliki broj mogućnosti uređivanje samo ponašanje igrača.

First Person Controller



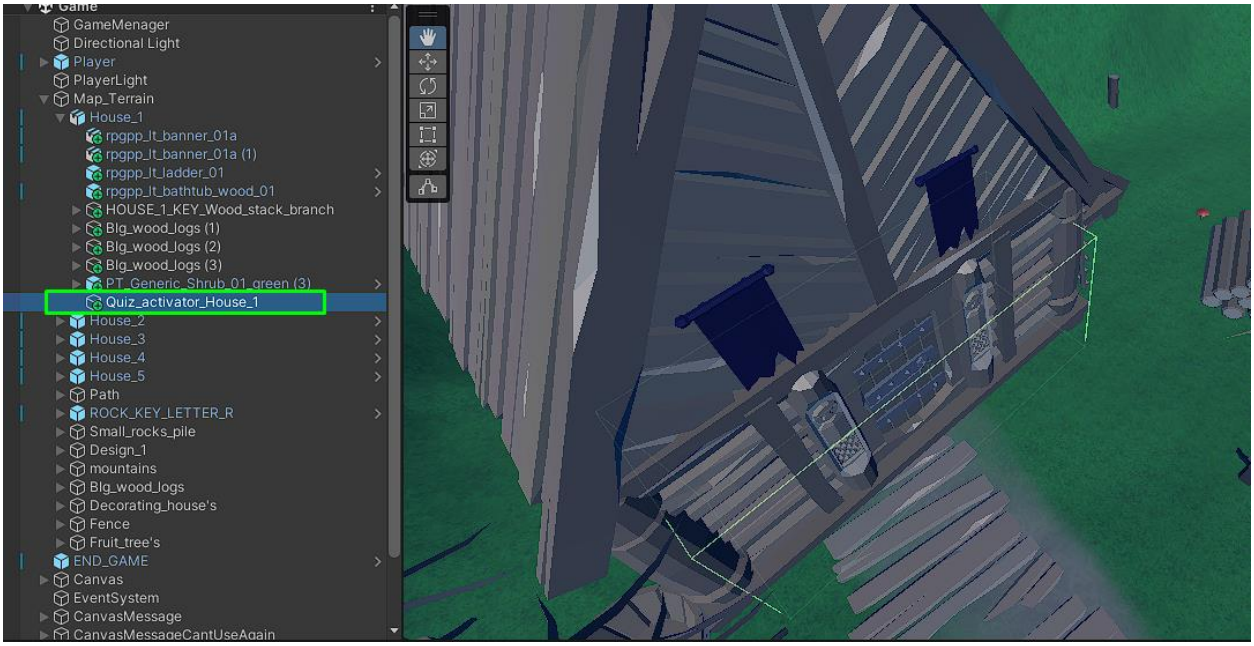
Nakon što se preuzme i importuje u projekat, ceo assets će se nalaziti u podrazumevanom „Assets“ folderu.



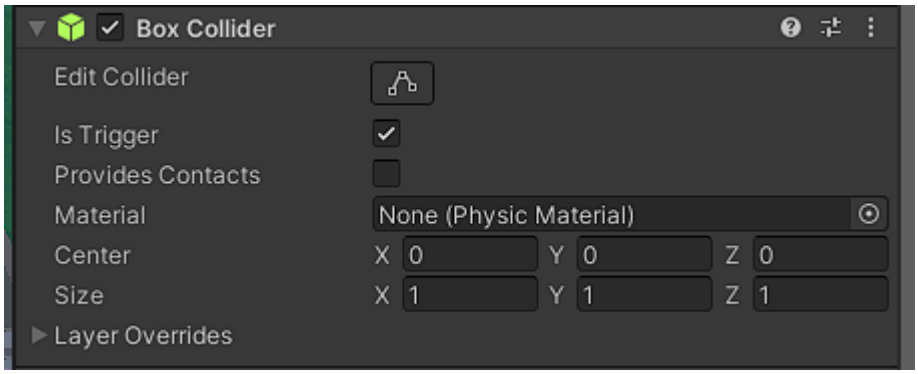
Nakon toga je samo potrebno dodati objekat Player unutar „Game“ scene.

Pokretanje kviza

Na mapi se nalazi 6 kviza koje takmičar treba rešiti, svaki od tih kvizova se aktivira kada se igrač približi i unese ključ.



Svaki kviz element ima empty objekat „Quiz activator“, koji detektuje igrača. Takođe sadrži i komponentu **BoxCollider**, kojom ćemo preko koda detektovati igrača. Važno je označiti **Is Trigger** kako bi sve radilo.



Kada igrač dodirne box collider dobija poruku o potrebnom ključu za aktivaciju samog kviza.

```
private void OnTriggerEnter(Collider other)
{
    if(other.CompareTag("Igrac"))
    {
        StartCoroutine(ShowTextForDuration(2f));
    }
}
```

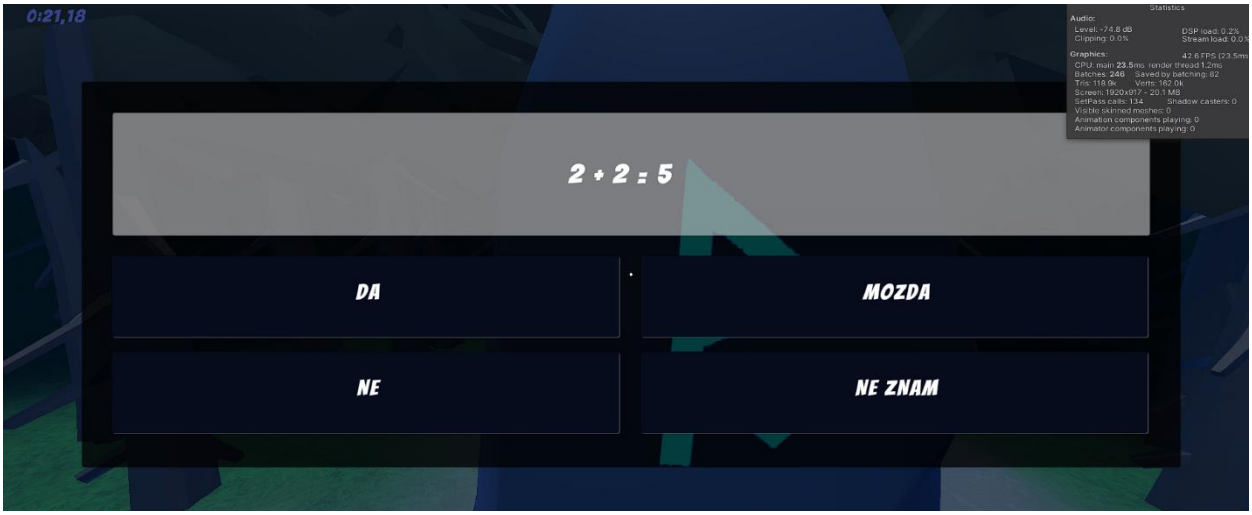
Ovom metodom omogućavamo ispisivanje poruke o potrebnom ključu.

```
private void OnTriggerStay(Collider other)
{
    if (obj.CompareTag("Rock_key") && other.CompareTag("Igrac") &&
Input.GetKey(KeyCode.R))
    {
        if (keys[0] == 0)
        {
            Debug.Log("R is pressed");
            QuizCanvasEnable();
            keys[0] = 1;
        }
        else
            StartCoroutine(ShowAnswerCheckForDuration(cantUseAgainCanvas));
    }
    if (obj.CompareTag("House_1_key") && other.CompareTag("Igrac") &&
Input.GetKey(KeyCode.Alpha6))
    {
        if (keys[1] == 0)
        {
            Debug.Log("6 is pressed");
            QuizCanvasEnable();
            keys[1] = 1;
        }
       else
            StartCoroutine(ShowAnswerCheckForDuration(cantUseAgainCanvas));

    }
    if (obj.CompareTag("House_2_key") && other.CompareTag("Igrac") &&
Input.GetKey(KeyCode.Alpha3))
    {
        if (keys[2] == 0)
        {
            Debug.Log("3 is pressed");
            QuizCanvasEnable();
            keys[2] = 1;
        }
        else
            StartCoroutine(ShowAnswerCheckForDuration(cantUseAgainCanvas));
    }
    if (obj.CompareTag("House_3_key") && other.CompareTag("Igrac") &&
Input.GetKey(KeyCode.Alpha7))
    {
        if (keys[3] == 0)
        {
```

```
            Debug.Log("7 is pressed");
            QuizCanvasEnable();
            keys[3] = 1;
        }
        else
            StartCoroutine(ShowAnswerCheckForDuration(cantUseAgainCanvas));
    }
    if (obj.CompareTag("House_4_key") && other.CompareTag("Igrac") &&
Input.GetKey(KeyCode.Alpha8))
    {
        if (keys[4] == 0)
        {
            Debug.Log("8 is pressed");
            QuizCanvasEnable();
            keys[4] = 1;
        }
        else
            StartCoroutine(ShowAnswerCheckForDuration(cantUseAgainCanvas));
    }
    if (obj.CompareTag("House_5_key") && other.CompareTag("Igrac") &&
Input.GetKey(KeyCode.Alpha5))
    {
        if (keys[5] == 0)
        {
            Debug.Log("5 is pressed");
            QuizCanvasEnable();
            keys[5] = 1;
        }
        else
            StartCoroutine(ShowAnswerCheckForDuration(cantUseAgainCanvas));
    }
}
```

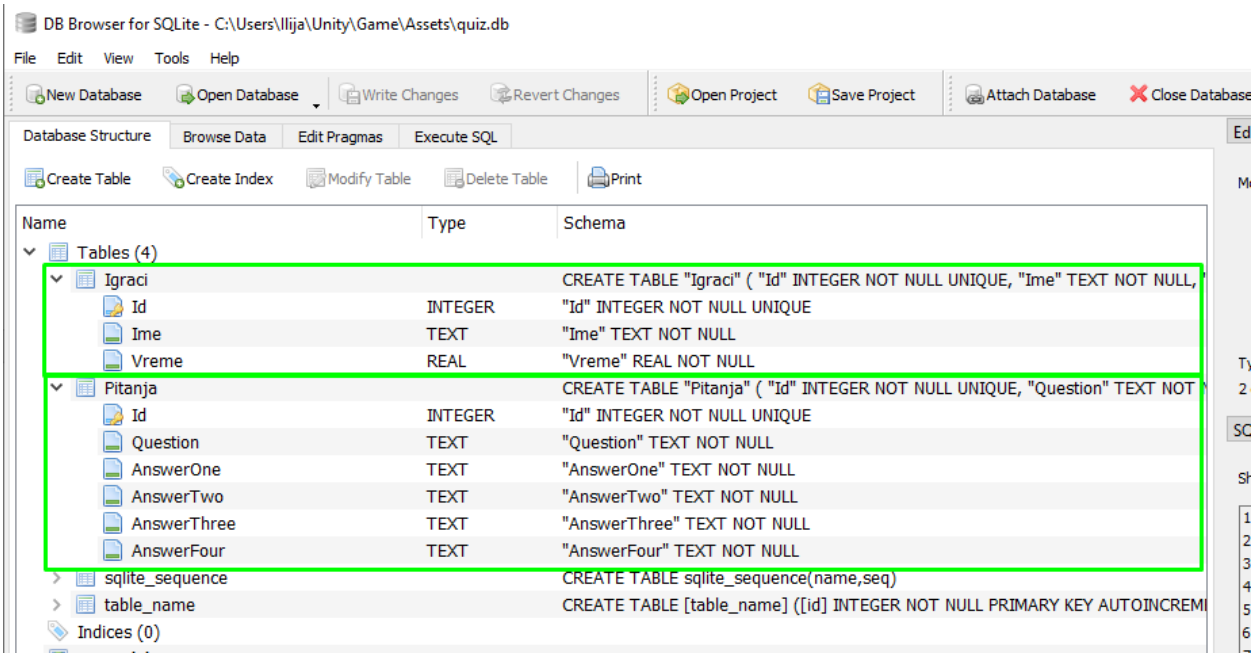
Deo koja koji nam omogućava da proveravamo koji kviz igrač pokušava da pokrene i da li je taj kviz već rešen.

Nakon što je unet odgovarajući ključ igraču će biti omogućen pristup samom kvizu, na ekranu će se pojaviti pitanje na koje je potrebno odgovoriti.



Izvlačenje pitanja

Sva pitanja su smeštena u bazi podataka. Kao baza podatak je porišćena SQLite baza. Samoj bazi možemo pristupiti preko DB Browser for SQLite.



U bazi se nalaze dve tabele, tabela „Igrači“ i tabela „Pitanja“, koje sadrže odgovarajuće promenljive. Svako pitanje se čuva u tabeli tako sto je u koloni „AnswerOne“ tačan odgovor, dok u ostale tri kolone netačni odgovori.

Da bi izvukli pitanja iz baze potrebno je uspostaviti konekciju izmedju same baze i igrice.

```
public void GetQuestionFromDB()
{
    string conn = "URI=file:" + Application.dataPath + "/quiz.db";
    IDbConnection dbconn;
    dbconn = (IDbConnection)new SqliteConnection(conn);
    dbconn.Open();
    IDbCommand dbcmd = dbconn.CreateCommand();

    randomLine = UnityEngine.Random.Range(1, 62);
    string sqlQuery = "SELECT * FROM Pitanja where Id LIKE "+ randomLine;
    dbcmd.CommandText = sqlQuery;
    IDataReader reader = dbcmd.ExecuteReader();

    while (reader.Read())
    {
            int id = reader.GetInt32(0);
            string questionStr = reader.GetString(1);
            correctAnswer = reader.GetString(2);
            List<string> answers = new List<string>();
            for (int i = 2; i < 6; i++)
                answers.Add(reader.GetString(i));
```

```
            question.text = questionStr;
            RotateList(answers);
            answerOne.text = answers[0];
            answerTwo.text = answers[1];
            answerThree.text = answers[2];
            answerFour.text = answers[3];
    }
    reader.Close();
    reader = null;
    dbcmd.Dispose();
    dbcmd = null;
    dbconn.Close();
    dbconn = null;
}
```

Kako je u bazi, prva kolona uvek tačan odgovor, potrebno je odgovore izmešati, to nam omogućava metoda **RotateList(answers);**

```
static void RotateList<T>(List<T> list)
{
    int places = Random.Range(0, 4);
    int n = list.Count;

    // Korišćenje LINQ za dobijanje ciklički pomerene liste
    List<T> rotatedList = list.Skip(places % n).Concat(list.Take(places %
n)).ToList();

    // Kopiranje elemenata nazad u originalnu listu
    for (int i = 0; i < n; i++)
    {
        list[i] = rotatedList[i];
    }
}
```

Metodom **ButoonPressed()** proveravamo ispravnost datog odgovora, ukoliko je netačan igrač dobija kazneno vreme, odnosno dodaju mu se 15 sekunde.

```
public void ButoonPressed(UnityEngine.UI.Button button)
{
    if (button.GetComponentInChildren<TextMeshProUGUI>().text.Equals(correctAnswer))
    {
        quizCanvas.enabled = false;
        StartCoroutine(ShowAnswerCheckForDuration(correctAnswerCanvas));
    }
    else
    {
        Timer.startTime -= 15;
        quizCanvas.enabled = false;
        StartCoroutine(ShowAnswerCheckForDuration(wrongAnswerCanvas));
    }
    counter++;
}
```

Vreme

Kako bi napravili high score tabelu, svakom igraču se meri vreme koje mu je potrebno da završi igru. Prilikom pokretanja igre, pokreće se i timer, koji se zaustavlja ako igrač dođe do kraja igre.

Timer je realizovan preko nekoliko jednostavnih linija koda:

```
void Start()
{
    startTime = Time.time;
}

void Update()
{
    if (finnished)
        return;

    t = Time.time - startTime;

    minutes = ((int) t / 60).ToString();
    seconds = (t % 60).ToString("f2");

    timerText.text = minutes + ":" + seconds;
}
public void Finnished()
{
    finnished = true;
    timerText.color = Color.red;
}
```

Metoda **void Start()** se pokreće samo jednom, prilikom pokretanja igre, dok se **void Update()** pokreće u svakom frame-u. Vreme dobijamo tako što od trenutnog vremena oduzimamo početno vreme, tako da dobijamo vreme koje je igrač proveo u igri.

```
public Timer timerScript;
private void OnTriggerEnter(Collider other)
{
    if(other.CompareTag("Igrac") && QuizActivator.counter >= 6)
    {
        timerScript.Finnished();
        UpisiRezultat(PlayGame.ime, (int)Timer.t);
        StartCoroutine(WaitBeforeLoad());
    }
}
```

Igra se završava ako je igrač došao do kraja i odgovorio na sva pitanja.

Metoda **UpisiRezultat(PlayGame.ime, (int)Timer.t);** omogućava upisivanje rezultata u high score tabelu igrača.

```
public void UpisiRezultat(string ime, int vreme)
{
    string conn = "URI=file:" + Application.dataPath + "/quiz.db
    IDbConnection dbconn;
    dbconn = (IDbConnection)new SqliteConnection(conn);
    dbconn.Open();
    IDbCommand dbcmd = dbconn.CreateCommand();
    string sqlQuery = "INSERT INTO Igraci (Ime, Vreme) VALUES ('" + ime + "','" +
(int)vreme + "')";
    dbcmd.CommandText = sqlQuery;
    IDataReader reader = dbcmd.ExecuteReader();
    dbconn.Close();
}
```

Kako se tabela igrača nalazi u bazi, prilikom upisa je potrebno otvoriti konekciju prema bazi.

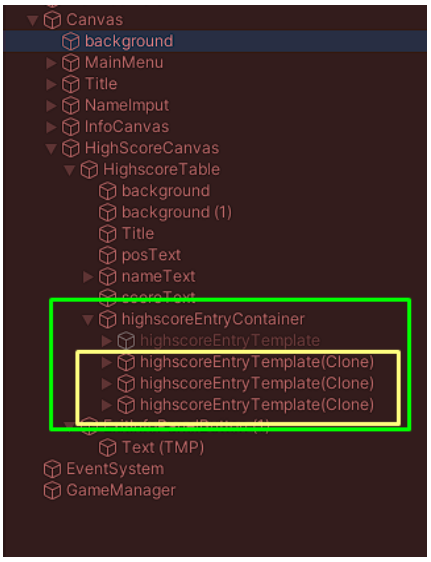
High Score Tabela

```
using System.Collections;
using System.Collections.Generic;
using System.Data;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using Mono.Data.Sqlite;
using System;

public class HighScoreTable : MonoBehaviour
{
    public Transform entryContainer;
    public Transform entryTemplate;
    public TextMeshProUGUI posText;
    public TextMeshProUGUI nameText;
    public TextMeshProUGUI scoreText;
    int minutes, seconds, time;
    private void Awake()
    {
        entryTemplate.gameObject.SetActive(false);
        float high = 30f;

        string conn = "URI=file:" + Application.dataPath + "/quiz.db";
        IDbConnection dbconn;
        dbconn = (IDbConnection)new SqliteConnection(conn);
        dbconn.Open();
        IDbCommand dbcmd = dbconn.CreateCommand();
        string sqlQuery = "SELECT * FROM Igraci ORDER BY Vreme ASC LIMIT 9";
        dbcmd.CommandText = sqlQuery;
        IDataReader reader = dbcmd.ExecuteReader();
        int i = 0;
        while(reader.Read())
        {

            Transform entryTransform = Instantiate(entryTemplate, entryContainer);
```

```
            RectTransform entryRectTransform =
entryTransform.GetComponent<RectTransform>();
            entryRectTransform.anchoredPosition = new Vector2(0, -high * i);
            entryTransform.gameObject.SetActive(true);
            int rank = i + 1;
            string ranking;
            switch (rank)
            {
                default: ranking = rank + "TH"; break;
                case 1: ranking = "1ST"; break;
                case 2: ranking = "2ND"; break;
                case 3: ranking = "3RD"; break;
            }

            posText.text = ranking;
            nameText.text = reader.GetString(1);
            time = (int)reader.GetFloat(2);
            minutes = (time % 3600) / 60;
            seconds = time % 60;
            scoreText.text = minutes+":"+seconds;
            i++;
        }

    dbconn.Close();

    }
}
```

Cela funkcionalnost klase HighScore je napisana u metodi **void Awake()**, ova metoda se pokreće samo prilikom pokretanja objekta za koje je skripta vezana, tako da svaki put kada želimo da proverimo HighScore tabelu ona će se iznova popuniti.

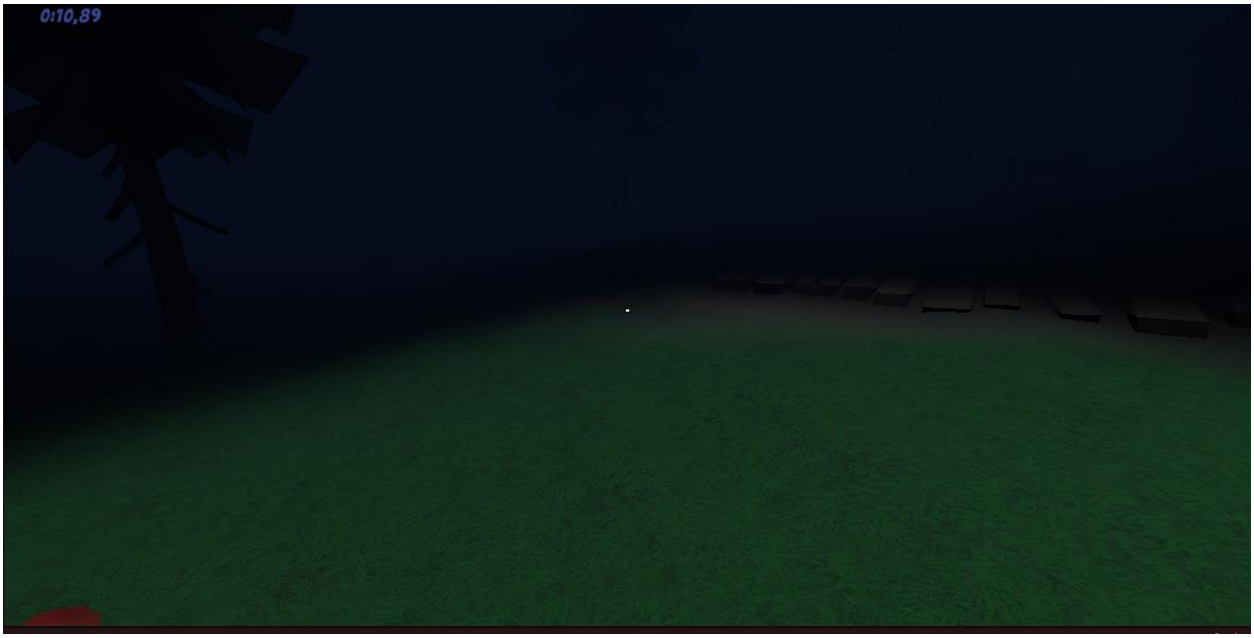
HighScore tabela predstavlja TOP 8 igrača koji su najbrže odigrali.



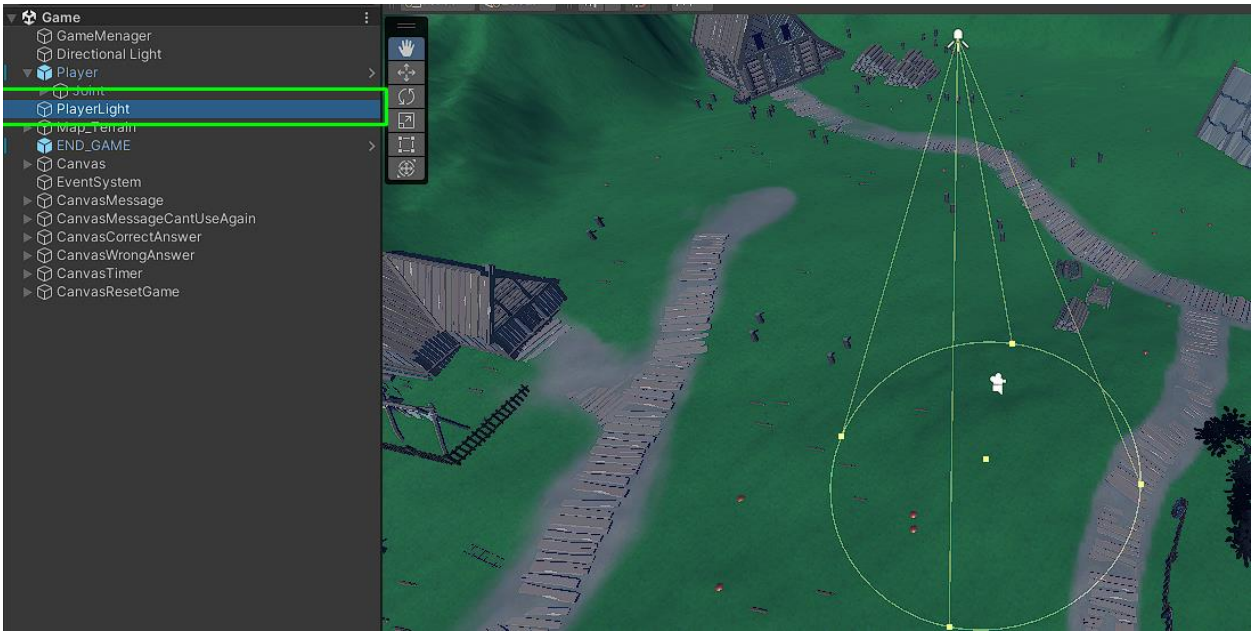
Cilj klase je da prilikom pokretanja kreira klonove objekata koji sadrže podatke o igračima, ti klonovi se nakon završetka brišu automatski.

Svetlo na igraču

Oko igrača se nalazi osvetljeni deo, koji omogućava određenu vidljivost, kao što je prikazano na slici.



Ovaj efekat se realizuje tako što kreiramo objekat Light I dodelimo mu određene funkcije.



*Nije dovoljno samo objekat zakačiti na igrača, već mora preko koda.

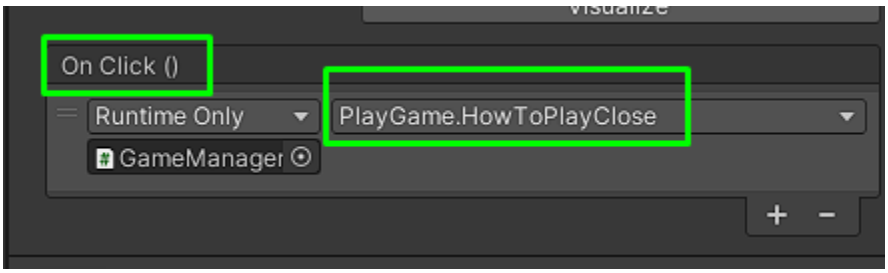
```
public class SpotLightFollowPlayer : MonoBehaviour
{
    public GameObject player;
    public float distance = 16.0f;

    void Update()
    {
        transform.position = player.transform.position + Vector3.up * distance;
    }
}
```

SpotLightFollowPlayer skripta je zakačena na svetlo, koja u svakom frame-u, svoju poziciju pomera na poziciju igrača, time omogućava praćenje igrača.

Igraj igru

Sama igra se sa stoji iz scena, početna scena koji igrači vidi je:

Svaki od button-a ima posebnu funkcionalnost koju realizuje određena skripta. Međutim nije samo dvoljna skripta, već moramo svakom button-u dodeliti event.



Ukoliko igrač želi da prekine igru, to može izvesti klikom na dugme “X”, to mu omogućava sledeći kod:

```
public void ResetGame()
{
    if(Input.GetKey(KeyCode.X))
    {
        ResetGamePanel.enabled = true;
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }
}
public void YesReset()
{
    SceneManager.LoadScene(0);
}
public void NoReset()
{
    ResetGamePanel.enabled = false;
    Cursor.lockState = CursorLockMode.Locked;
    Cursor.visible = false;
}
```

Zvuk

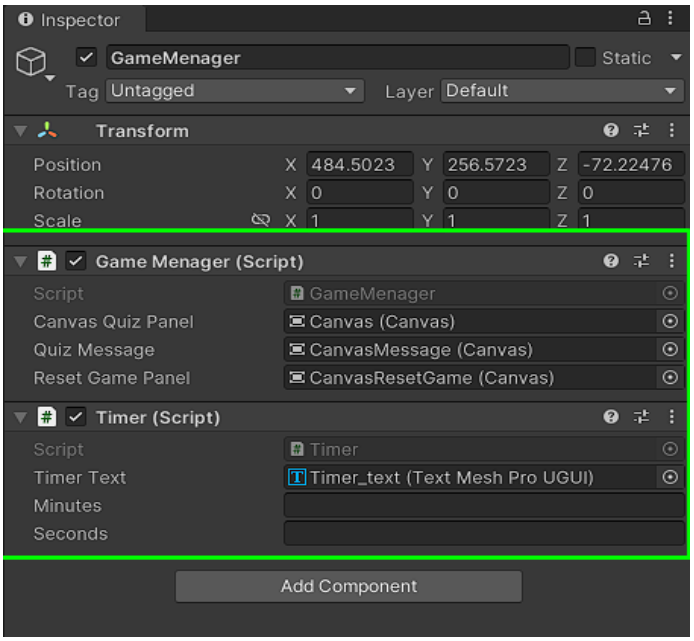
Kako igra ne bi bila monotona potrebno je imati neke zvukove, audio možemo preuzeti i dodati u unity projektu, unity podržava audio source.

Jednostavnim kodom možemo aktivirati ove zvukove tako se oni aktiviraju odgovarajucim kretanjem igrača.

UNITY

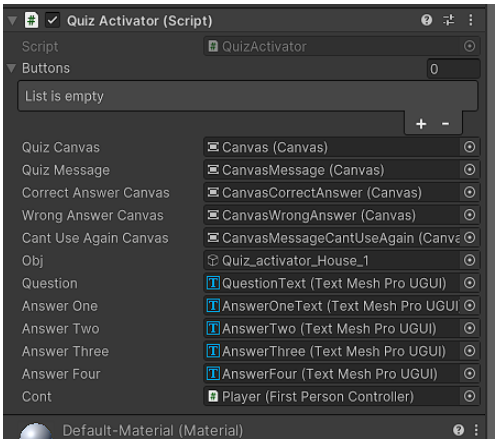
GameManager objekti

Game manager objekti u unity obično služe da određene stvari vezane za igru, a ne za određene objekte, izvršavaju.
Game manager objekat je zapravo kreiran u unity kao empty object, na koga su zakačene sve skripte koje on izvršava.



Unity scripts

Unity skripte se pišu u programskom jeziku C#, isključivo se dodaju na objekte koje treba da izvršavaju kod koji se nalazi unutar skripte.



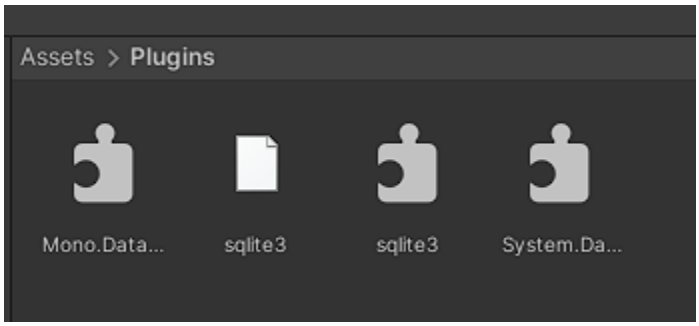
Skripte se čuvaju, kao i sve ostale stvari u assets folderu.

Unity i baze podataka

Iako unity radi sa C#, rad sa bazom podataka nije isti kao običnom C#. Unity nema podršku za baze, tako da mora ručno da se dodaju .dll fajlovi kako bi bio omogućen rad. Ukoliko se radi sa SQLite bazom potrebno je sa sajta SQLite.com skinuti odgovarajući .dll fajl.
Takođe je potrebno dva .dll fajl-a prebaciti u projekat.

. Copy **System.Data.dll** and **Mono.Data.Sqlite.dll** from **C:\Program Files (x86)\Unity \Editor\Data\Mono\lib\mono\2.0* and paste them in your Assets/Plugins* folder in your unity project.

Sve .dll fajlove je potrebno ubaciti u folderu “Plugins” koji je potrebno kreirati unutar “Assets” foldera.
Unity prilikom pokretanja “traži” folder “Plugins” jer je rezervisano ime i zbog toga moraju .dll biti u njemu.



KORISNI LINKOVI

Unity assets store

Unity dokumentacija

Konekcija baze podataka i unity 3D

SQLite