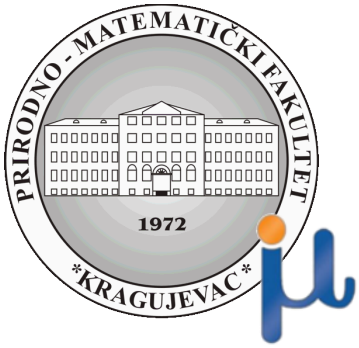
Kragujevac April 2017.



COLORSWITCH

CLONE

AUTORI:

Strahinja Milosavljević

Jovana Radovanović

Sadržaj:

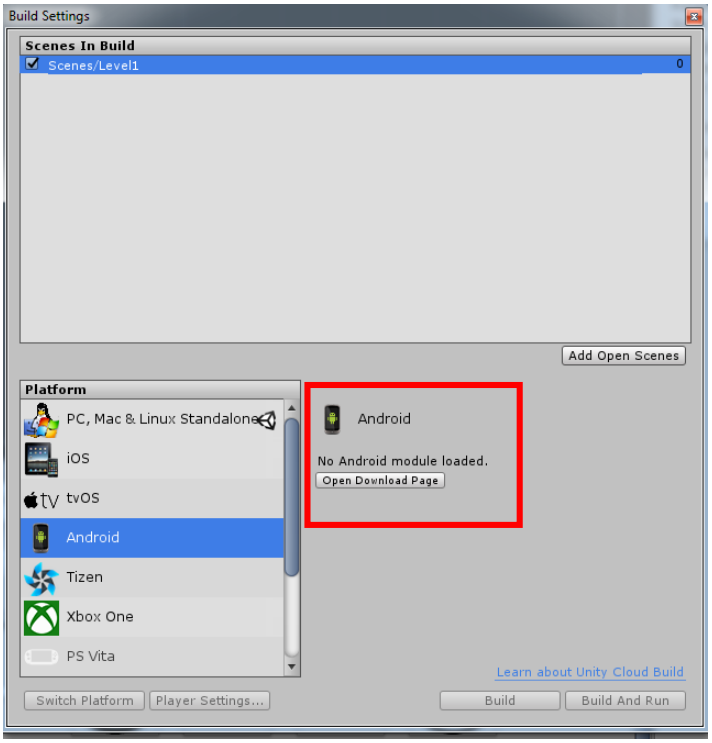
1. Uvod

1.1. Napomene potrebne za izradu projekta

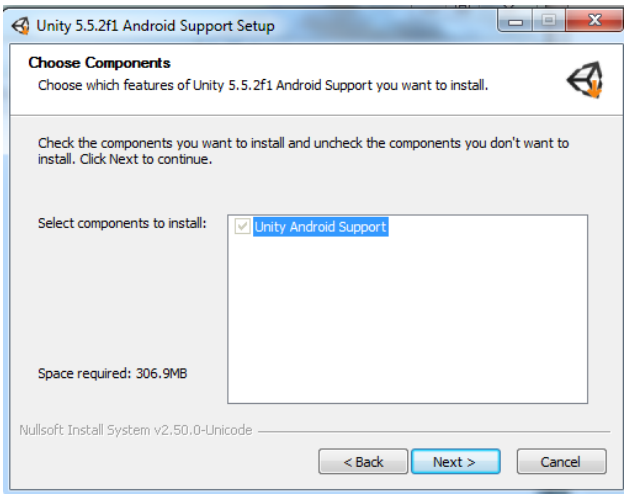
2. Ideja za Color Switch igricu prvenstveno je u vlasništvu “Fortafy Games” i koristi se u ovoj literaturi samo u edukacione svrhe. Mi smo ovde kreirali Clone verziju kako bi pokazali kako ona može da se napravi u Unitiju.
3. Za potrebe kreiranja dizajna (**Front-enda**) ove igrice korišćena je verzija **Unity 5.5.2** i preporučujemo da ažurirate starije verzije na novije zbog nekih pogodnosti i mogućih razlika sa starijim verzijama ukoliko ste imali Unitz pre iyrade ovog projekta.
4. Za kodiranje (**back-End**) pisan je kod u C# u **Visual Studio 2015** ali može i neki drugi editor kao što je npr. Mono Developer-u
5. Voditi računa o čestom čuvanju projekta i Scena kako bi se izbeglo nepredviđeno gubljenje podataka I postignutog učinka. Za potrebe čuvanja scena preporučujemo da se kreira poseban folder u **Assets/Scenes/…** kao i da se daju smislena imena scenama.
6. Davati smislena imena folderima, skriptama, promenljivama i objektima kako biste mogli lakše da pratite strukturu foldera projekta i sam kod.
7. Obratiti pažnju za koju platformu pravite igricu i najbolje bi bilo da se na početku projekta izvrše potrebna podešavanja.
8. Svaka scena da bi mogla da se pokrene kao apk mora prvo biti “izbildovana” a to se radi tako što se odabere **File->Build Settings**
9. Vodite računa da struktura projekta bude uredno raspoređena zbog vase lične preglednosti i boljeg snalaženja kako za Vas tako i za onog ko čita Vaš kod.

1.2. Odabir platforme i orijentacija ekrana

Da bi se određena scena mogla pokrenuti/uključiti u igricu i da bi mogla da prikaže ono što ste joj dodali neophodno je prvo “Izbildovati” je kao i selektovati platformu za koju želimo da kreiramo igricu, a Unity nam u tome jako olakšava posao jer je vrlo jednostavno. Ove opcije snalaze u **File->Build Settings.** Ako imamo neku scenu za bildovanje onda je dodajemo odabirom na Add Open Scenes (ovo dugme dodaje aktivnu scenu). Da bismo promenili platformu moramo prvo imati instaliran modul za platformu koju želite da selektujete. Pošto mi želimo da kreiramo krajnju igricu za android kada selektujete opciju “**Android**” videćete da će Vam tražiti



modul koji verovatno nemate. Kliknite na opciju **Open Download Page** i skinuće Vam se neophodna instalacija koju nakon toga pokrenete.

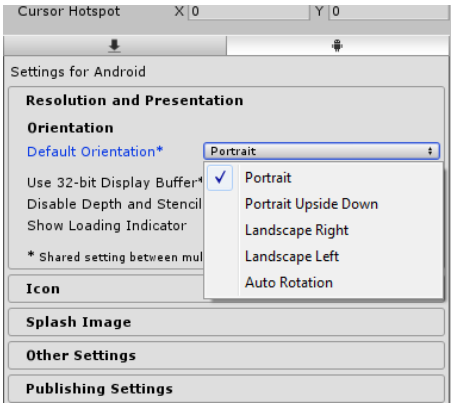


Instalacija će Vam tražiti da ugasite Unity editor kako bi mogao da implementira instalirani modul po završetku tako da to i uradite. Kada se vratite u projekat idaberite opciju **Swtich Platform** koja će Vam sada biti omogućena i **ikonica** za unity će se prebaciti na **Android**.

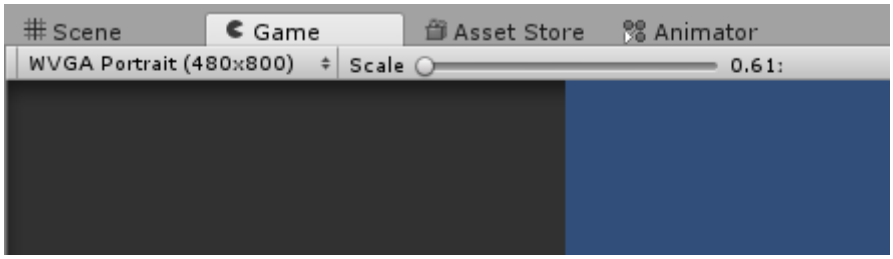
NAPOMENA: NAJBOLJE JE DA PLATFORMU SELEKTUJETE PRE POČETKA IZRADE PROJEKTA ZBOG RAZLIČITIH REZOLUCIJA I DIMENZIJA KAMERE KAKO NE BISTE POSLE NA KRAJU MORALI SVE DA NAMEŠTATE. ZATO ODMA NA POČETKU VAM BUDE SVE PODEŠENO I SPREMNO ZA RAD.

Kada pokrenete ponovo Vaš projekat i u **Build Settings** odaberete android platformu videćete da će sada prozor izgledati drugačije. Omogućeno je i dugme **Player Settings** koje se koristi za neka osnovna podešavanja aplikacije. Pošto ovu igricu pravimo za Android mi ćemo odabrati **Android platformu**. Ali igrica će moći da se pokrene i na **PC računaru**. Vi možete lako da odaberete I **iOS** ili koju god želite.

Ako želite da promenite **automatsko rotiranje ekrana** tj orijentaciju ekrana na androidu manuelno idete na **File->Build Settings->Player Settings** u Inspector prozoru odabrati polje za Android I opciju **Default Orijentation postavite** na **Portrait** kao na slici**.**

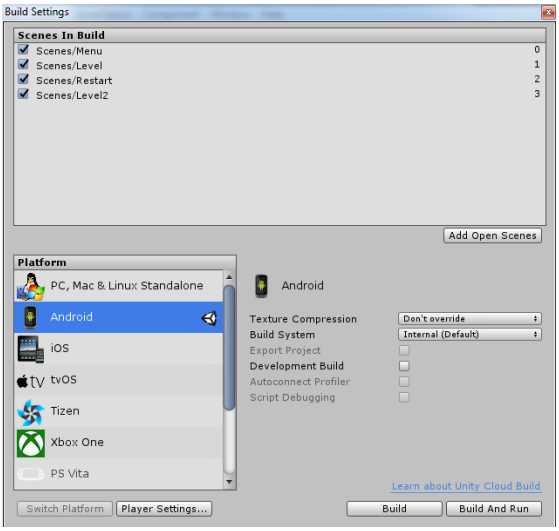


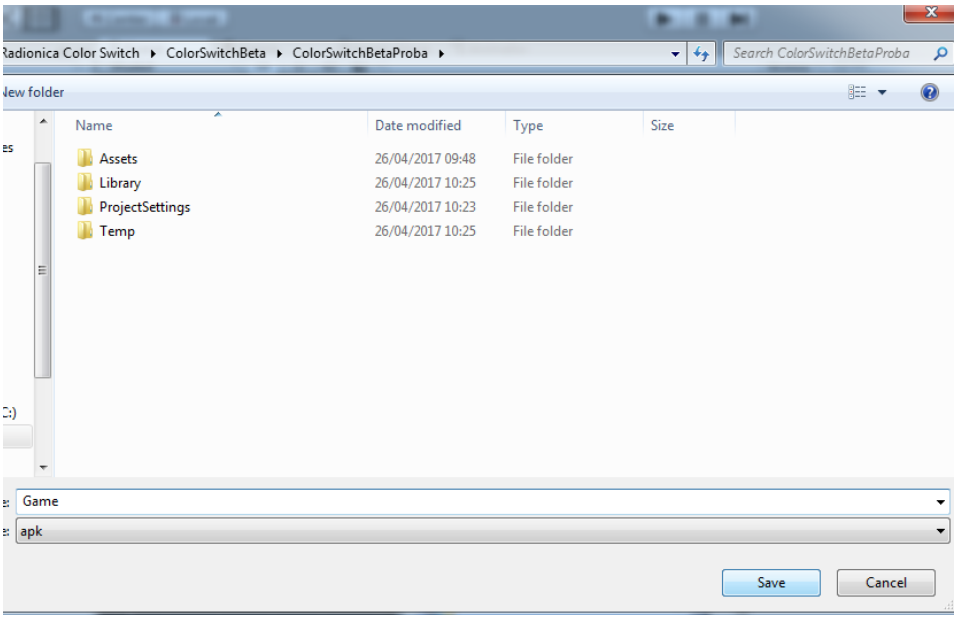
Ostalo je još samo da podesite rezoluciju **Kamere** tj kako će se Vaša igrica prikazivati. Odaberite opciju kao sa slike.



1.3. Bildovanje projekta i priprema za android

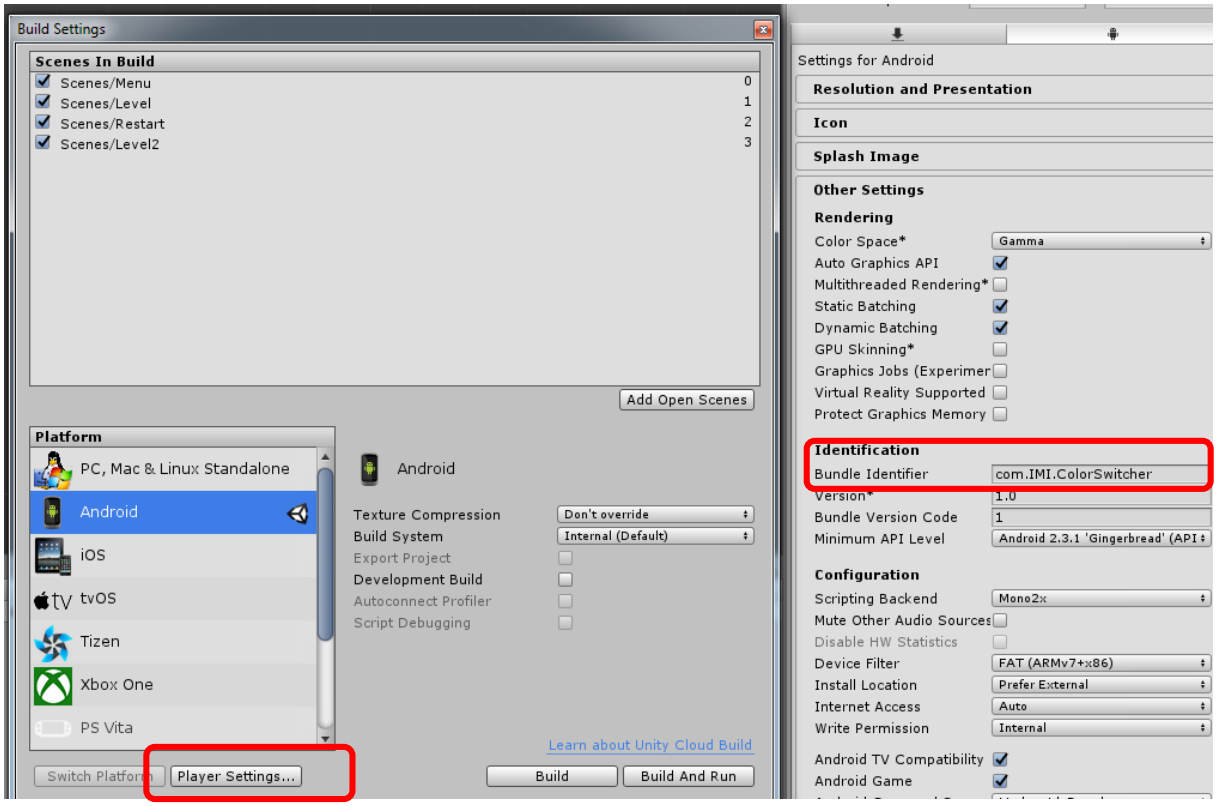
Na slici je dat primer izbildovanih scena. Scenu dodajete u spisak za bildovanje odabirom dugmeta "**Add Open Scenes**" i dodaće onu scenu koja Vam je trenutno aktivna. Ostala polja ostavite po default-u. Kada kliknete na Build dugme tražiće se od Vas da sačuvate fajl kao .**apk** tj to je ekstenzija za krajnju aplikaciju.





Takođe odabirom dugmeta **Player Settings** treba navesti i polje **Bundle Identifier** pošto će Vam Unity ako pokrente Build tražiti. Polje Bundle Identifier treba biti validno popunjeno tako što će se navesti product company i product Name u sledećem opisu:

com.[ProductCompany].[ProductName]



OBRATITE PAŽNJU DA SCENA KOJA SE NALAZI NA VRHU SPISKA SCENA JE DEFAULT-NA SCENA KOJA SE PRVA PRIKAZUJE AUTOMATSKI pri pokretanju igrice na odabranoj platformi!!!!

Nakon što odaberete opciju za Buildovanje tražiće Vam se i putanja do instaliranog **Andoird sdk** foldera. Naredni koraci daju detaljno uputstvo kako da instalirate sdk i zatim nakon toga dodelite putanju koja je uglavnom C:\Users\YourName\AppData\Local\Android\sdk.

Da bi ste uspešno setovali Platformu za Android neohpodno je:

1. Instalirati modul **u File->Build Settings** za android i odabrati **Switch Platform**
2. da se instalira **Android SDK** za Build-ovanje
3. Android SDK će verovatno instalirati i **Android Studio**
4. zatim se u Unity editoru dodeli putanja do instaliranog foldera ako idete na **Build** u kome se nalazi Android SDK (C:\Users\YourName\AppData\Local\Android\sdk)

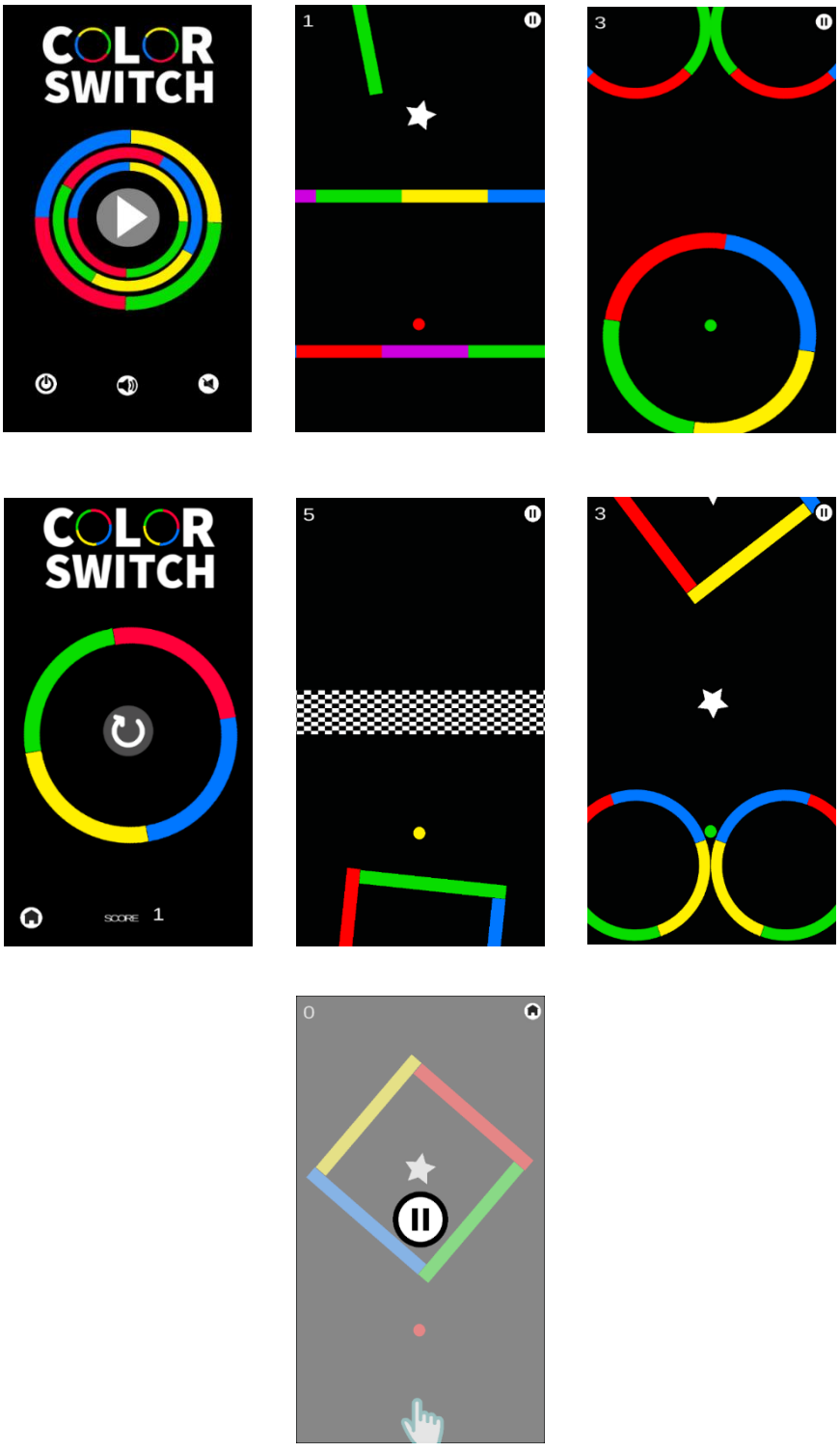
Postoji par načina za prebacivanja projekta na Android (Teža):

1. Povezati Vaš android telefon ukoliko želite da pokrenete aplikaciju nakon Build-ovanja preko **USB kabla** sa Androidom
2. morate omogućiti "**Opcije razvoja**" , uputstvo za nalaženje Developer opcija je na linku https://docs.unity3d.com/Manual/android-sdksetup.html
3. čekirati polje **USB otklanjanje grešaka.**
4. Instalirati **Android Studio** sa sajta **https://developer.android.com/index.html**
5. Instalirati **PC Suite** program za razmenu medijskog sadržaja PC-a i vašeg Androida u slučaju da Vam javlja grešku prilikom Build-ovanja jer nema drajver.
6. Nakon opcije **Build & Run** kada se izbilduje projekat, aplikacija će pronaći Vaš uređaj i pokrenuti automatski projekat tj igricu.

Druga varijanta jeste lakša:

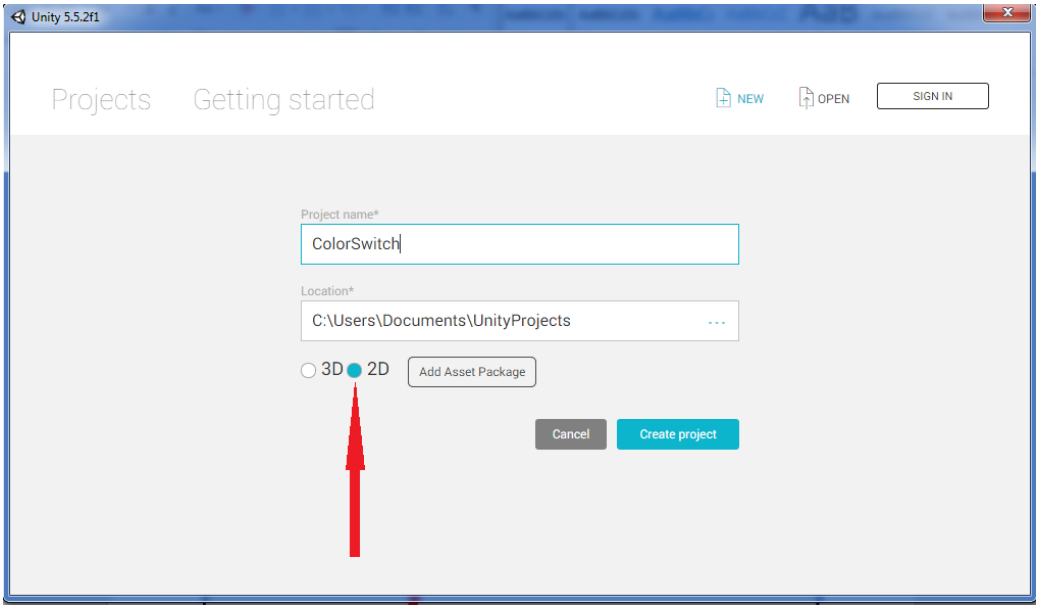
1. Izbildujete projekat **kao .apk** sa tom ekstenzijom u folderu
2. Povežete **USB kabl** sa Androidom
3. zatim jednostavno prebacite taj fajl na Vaš Android telefon
4. Locirate gde se nalazi fajl i pokrenete
5. Android će Vas pitati da li da **instalirate** projekat kažete Da i to je to

1.4. Kratak pregled finalnog proizvoda



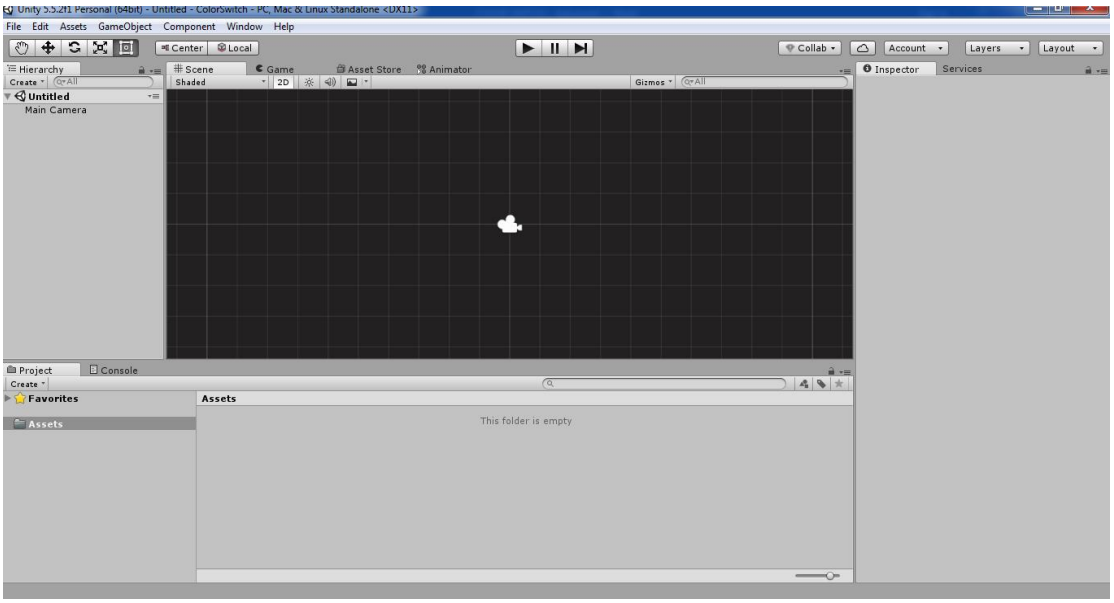
2. Kreranje loptice

U ovom poglavlju započinjemo izradu naše igrice. Pokrenuti program u kome ćemo kreirati našu igricu **Unity** 5.5.0. Nakon startovanja Unity programa prikazuje se prozor za kreiranje novog projekta, koji će biti ***2D***. U okviru polja ***Project name*** unosimo naziv projekta, u ovom slučaju **ColorSwitch** ,a ***Location*** predstavlja mesto gde će se naš projekat čuvati. Lokaciju možemo lako promeniti odabirom „...“.



Slika 1 Početni prozor

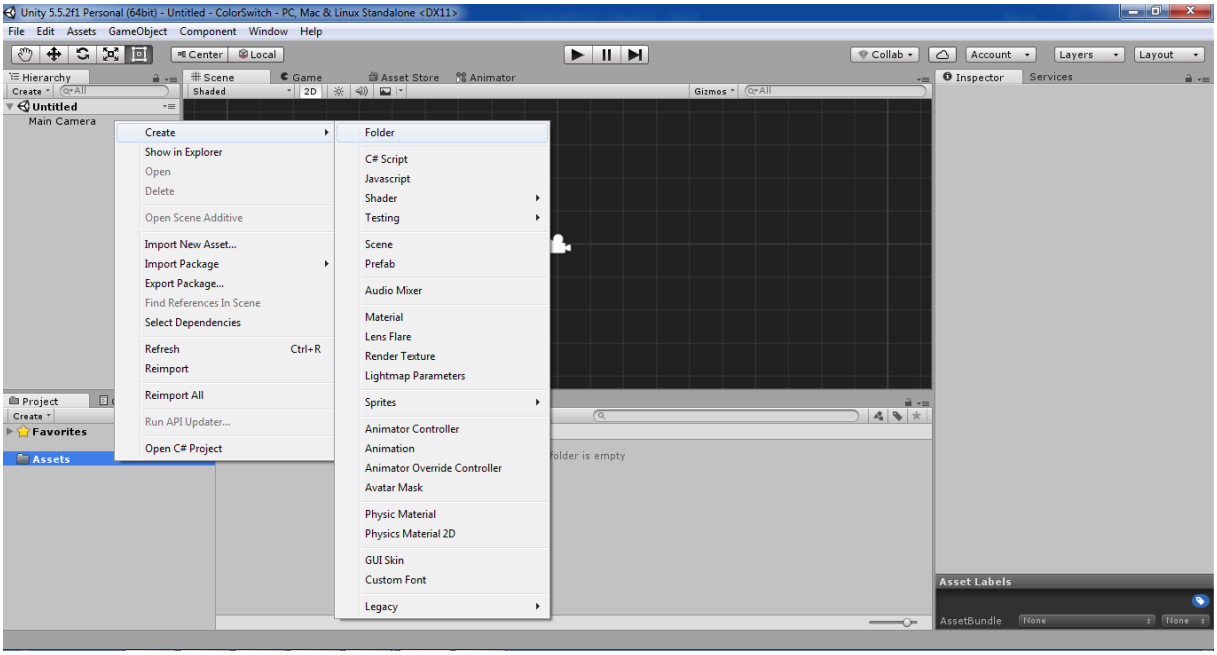
Nakon klika na dugme ***Create project*** otvara se okruženje-Editor Unity programa. **Assets** folder je u početku prazan.



Slika 2O kruženje Unity programa

Prvi korak pri kreiranju igrice je snimanje glavne scene. **Scene** ćemo čuvati u okviru foldera ***Scenes***, koji će biti podfolder foldera ***Assets*** (desni klik na folder

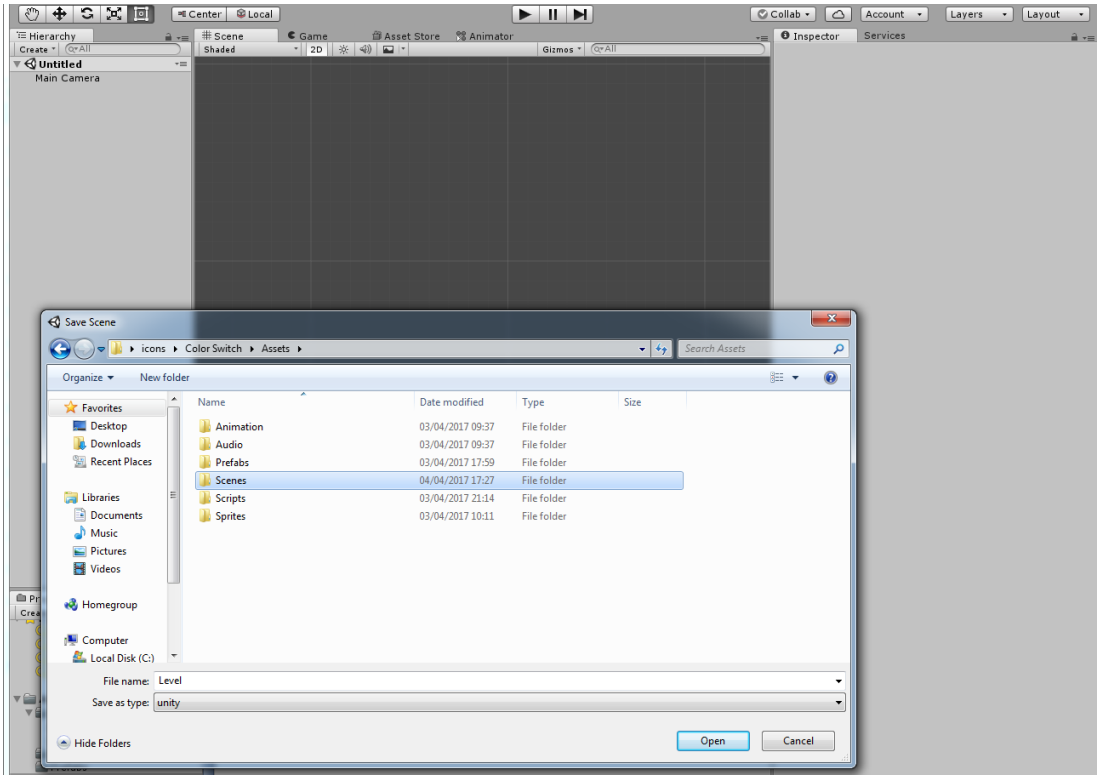
Assets ->Create ->Folder-> imenujemo ga Scenes).



Slika 3 Kreiranje foldera Scenes

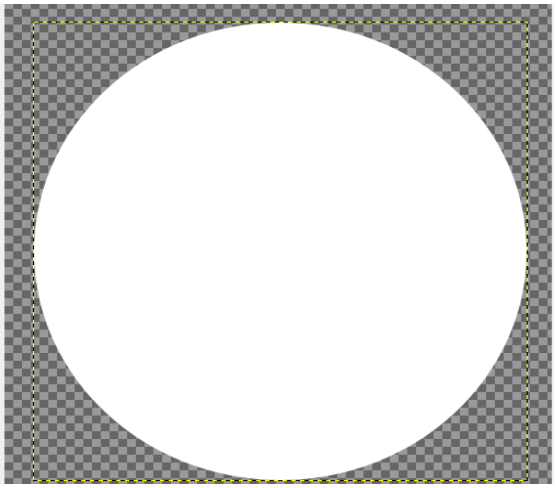
Scenu snimamo na sledeći način:

File->Save Scenes->nakon čega biramo folder Scenes ->scenu čuvamo pod nazivom Level.



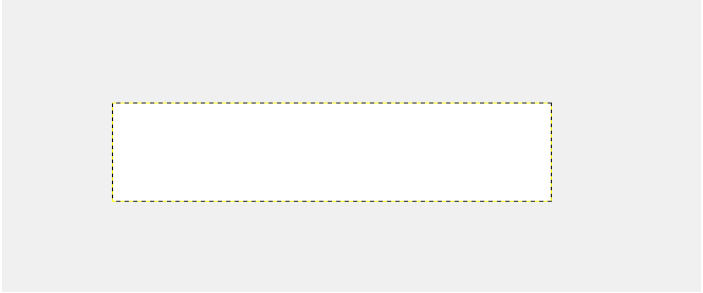
Slika 4 Snimanje scene Level

Sledeći korak jeste kreiranje. Uvozimo dva sprajta, u podfolder Assets/*Sprites/.* Prvi pod nazivom ***sphere*** , koji predstavlja lopticu i ima sledeći izgled:



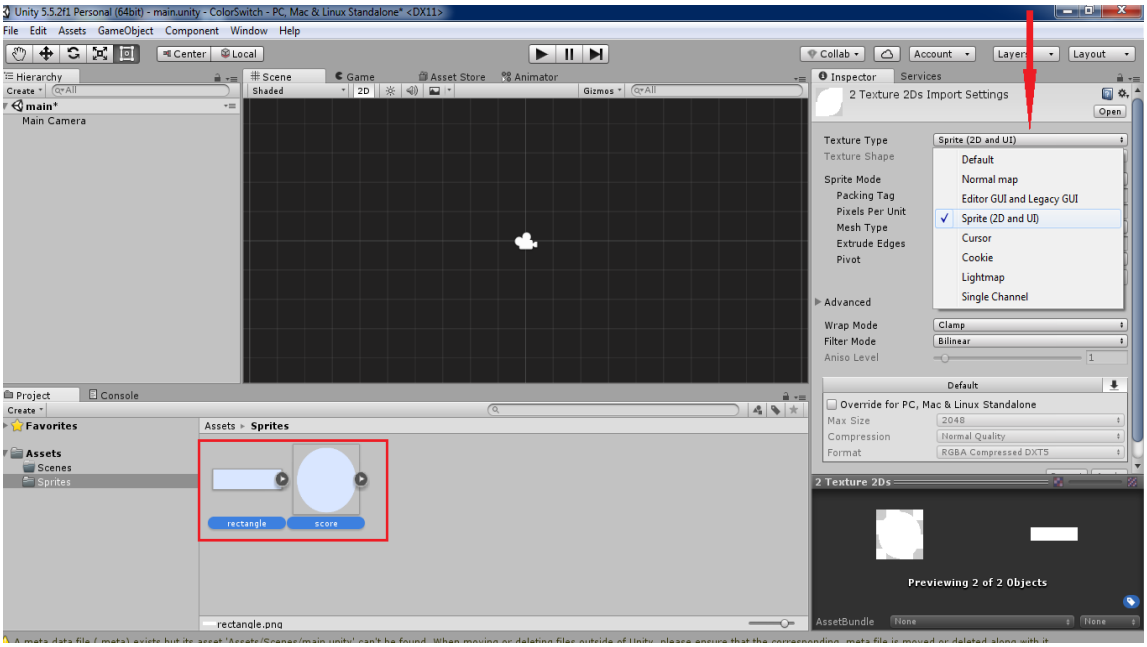
Slika 5 Sphere sprite

Drugi pod nazivom ***rectangle*** koji ćemo koristiti za kreiranje prepreka/objekata



Slika 6 Rectangle sprite

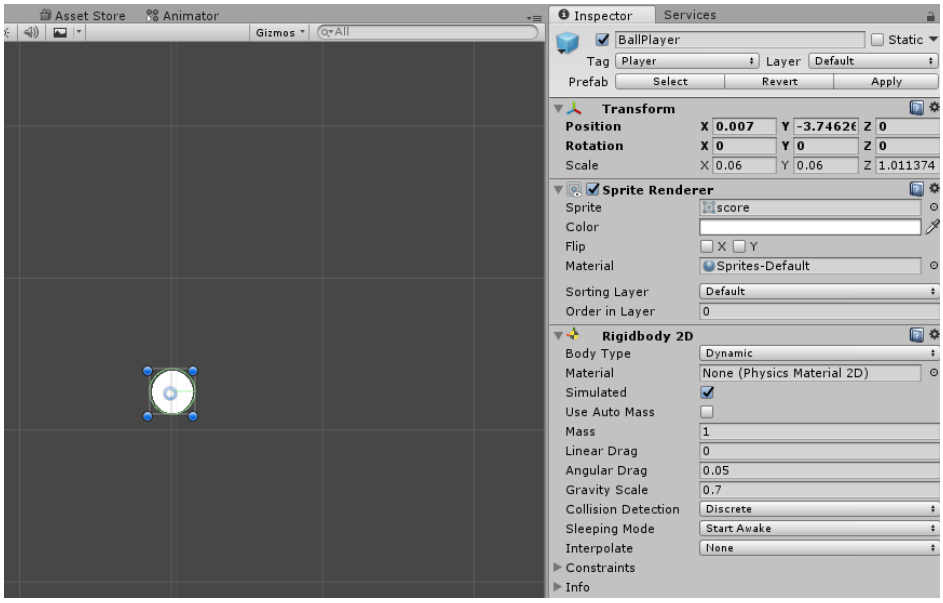
Nakon što prebacimo sprite-ove u folder **Sprites**, selektujemo oba sprite-a, i u Inspector prozoru sa desne strane za ***Texture Type*** postavimo ***Sprite(2D and UI)*** , kao na sledećoj slici:



Slika 7 Podešavanje za dva dodata sprite-a

Iz foldera Sprites oba sprite-a prevučemo na glavnu scenu (*Level*), u ***Hierarchy*** prozor. Klikom na sprite ***sphere*** možemo sada da mu promenimo ime i nazvaćemo ga ***BallPlayer*** kao igračeva loptica.

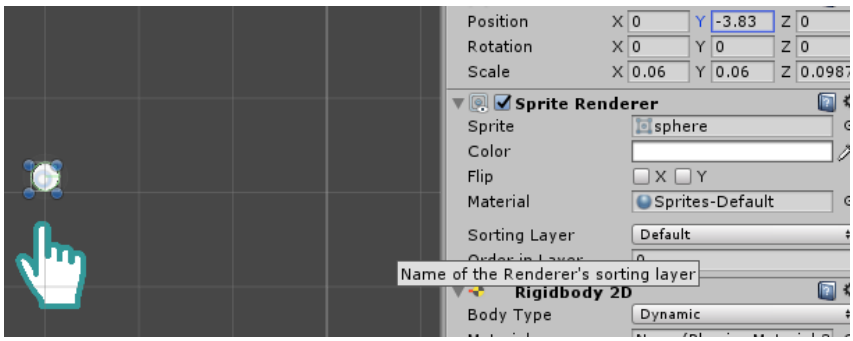
U ***Hierarchy*** prozoru, dobijamo mogućnost podešavanja tog sprite-a u okviru ***Inspector*** prozora (isto važi i za sprite ***rectangle***).



Slika 8 Sprite-ovi prebačeni na main scenu

Kada kreiramo objekat, Unity mu automatski dodeljuje komponentu ***Transform*** pomoću koje možemo da podešavamo:

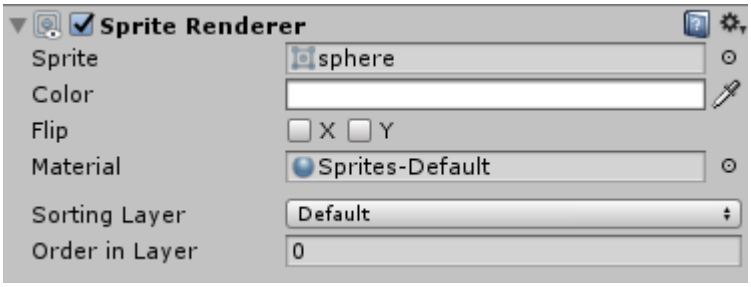
- **Position** – menjamo poziciju objekta
- **Rotation** - rotiramo naš objekat
- **Scale** – menjamo dimenzije objekta



Slika 9 Transform komponenta za sphere

Pozicije i dimenzije postavljamo proizvoljno.

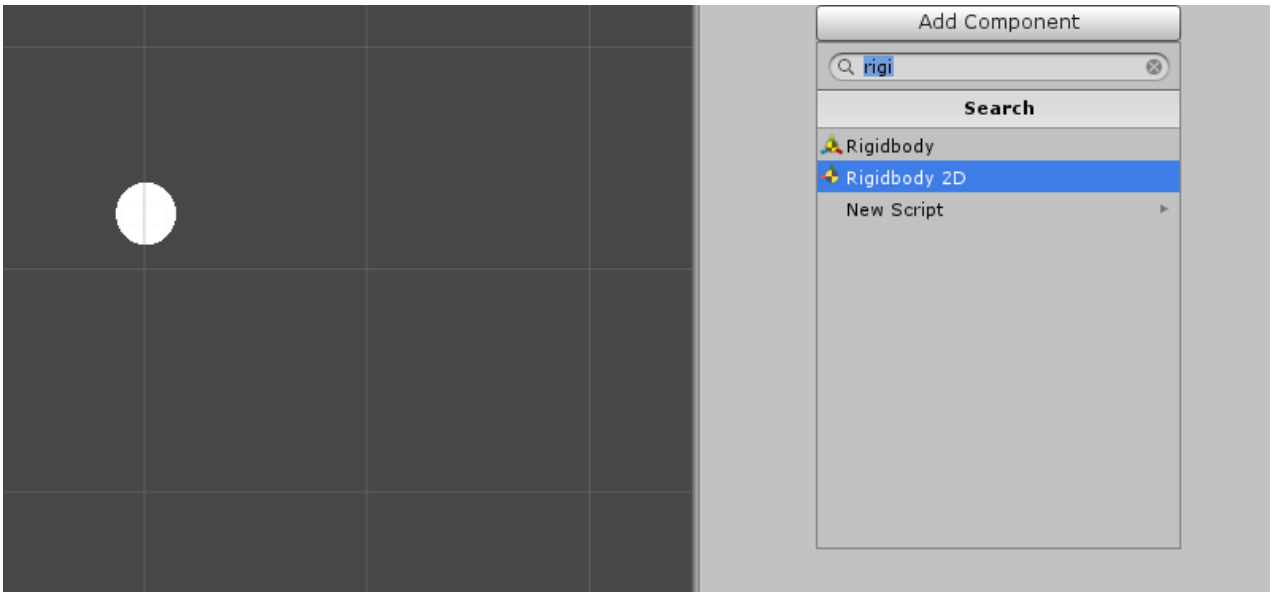
Kada je reč o sprite-u kao u našem slučaju, pored ***Transform*** komponente Unity dodaje ***Sprite Renderer*** komponentu, koja je pokazivač na sprite i u kome se pored ostalih mogućnosti nalazi i polje ***Color*** kao i ime sprajta:



Slika 10 Sprite Render komponenta za sphere

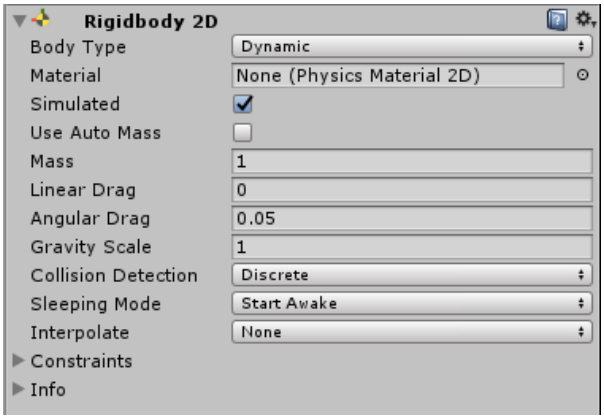
Sledeće što je potrebno sprite-ovima dodati je fizika.

1. Selektujemo ***BallPlayer*** u ***Hierarchy*** *prozoru -> Component -> Phisics 2D -> Rigidbody 2D ili*
2. *Klikom na* ***BallPlayer*** u Inspector prozoru odaberemo opciju ***Add Component*** i nađemo ***RigidBody2D*** što ćemo mi uraditi.



Slika 11 Dodavanje Rigidbody komponente

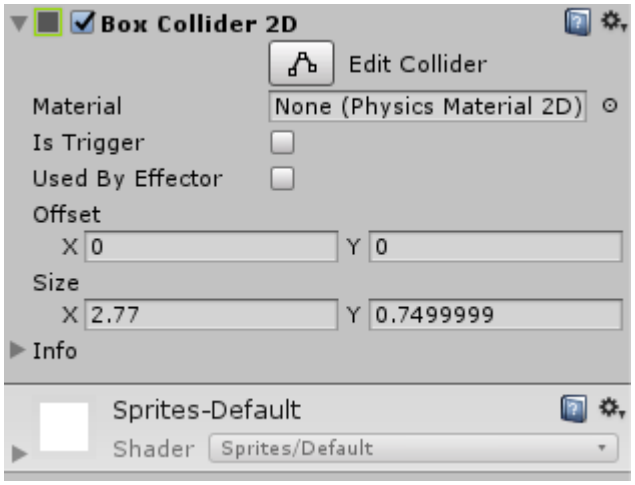
Nakon čega nam se u Inspector prozoru za selektovani sprite dodaje ***Rigidbody2D*** komponenta:



Slika 12 Rigidbody 2D za sphere

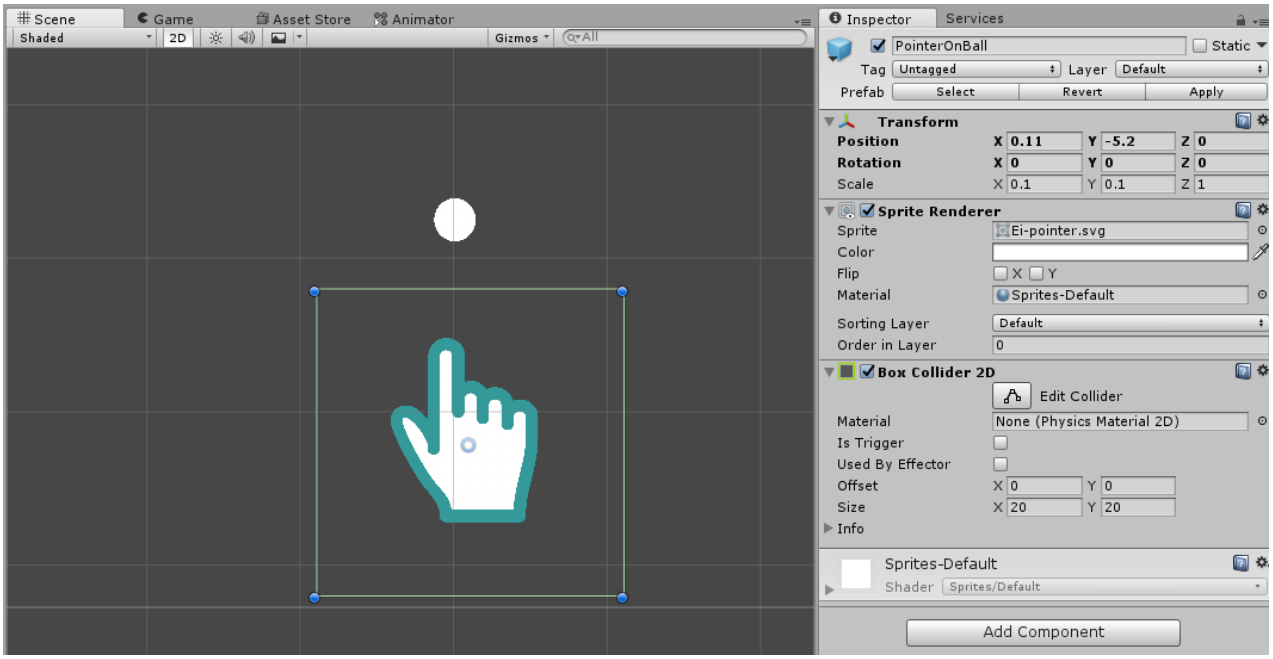
Ova komponenta stavlja sprite-ove pod kontrolu engine-a za **fiziku**. To omogućava da sprite bude pod uticajem gravitacije(***Gravity Scale*** je 1 – koliko jako će gravitacija uticati na objekat). Ukoliko sada pokrenemo projekat, klikom na dugme ***play***, primetićemo da naša loptica pada, jer na nju sada deluje sila gravitacije. Međutim mi želimo da naša loptica ostane u mestu i da ne pada. Zato ćemo za sada kreirati podlogu koja će joj omogućiti da ne propadne već da stoji u mestu kada dodirne sprite ***PointerOnBall*** .

Iz tog razloga selektujemo sprite iz foldera ***Assets/Sprites/pointeronball,*** *i* dodajemo u Hijararhiju kao objekat, a zatim mu dodajemo komponentu **BoxCollider2D** ***(Add Component -> Box Collider 2D)***. Postavićemo ga ispod loptice. Takođe dodajemo i **BoxCollider2D** na sprajt **rectangle** kao sa slike 13.



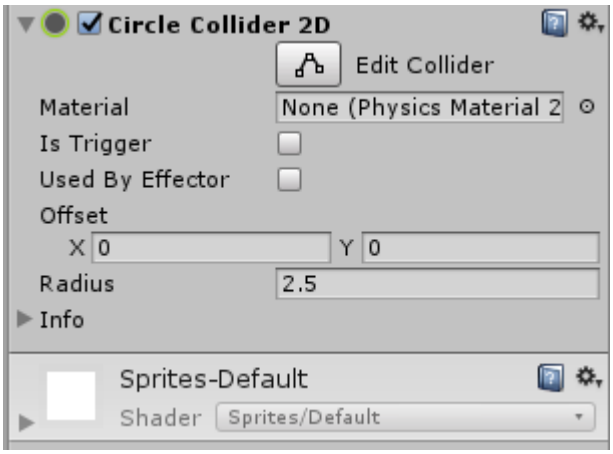
Slika 13 Box Collider 2D za rectangle sprite

Dakle, BoxCollider2D dodajemo kao komponentu za sprite-ove kojima želimo da omogućimo „težinu“ odnosno fiziku sudaranja sa drugim objektima (kolizijom). Nakon dodavanja BoxCollider2D-a, sprite ***rectangle,*** postaje oivičen zelenom linijom, što se može primetiti u ***Scene*** prozoru:



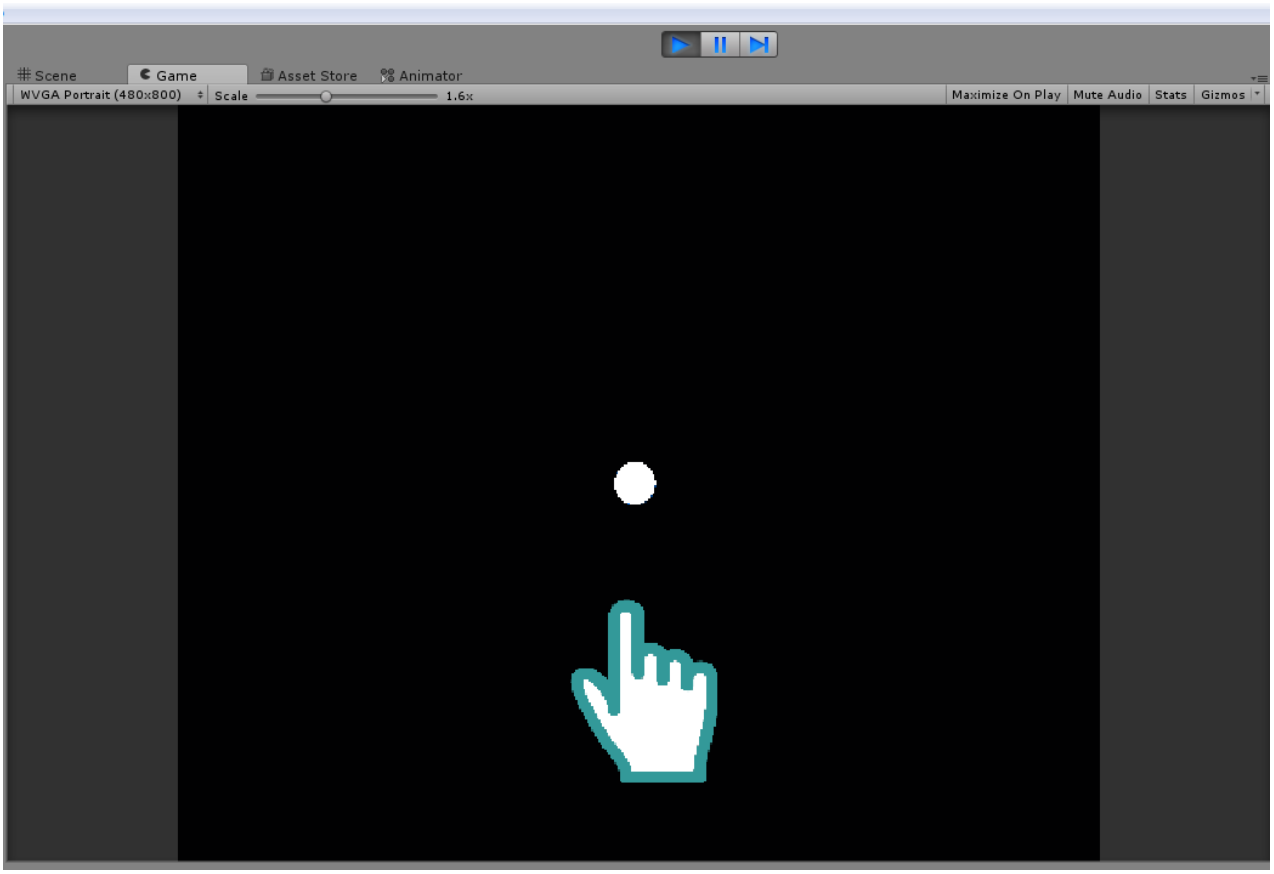
Slika 14 Izgled sprite-a nakon dodavanja Box Collider – a 2D

Za sprite ***BallPlayer*** dodajemo (u skladu sa oblikom) ***CircleCollider2D*** *(Add Component -> Circle Collider 2D).*



Slika 15 Circle Collider 2D za sphere

Kada pokrenemo projekat videćemo da dolazi do kolizije ova dva objekta:



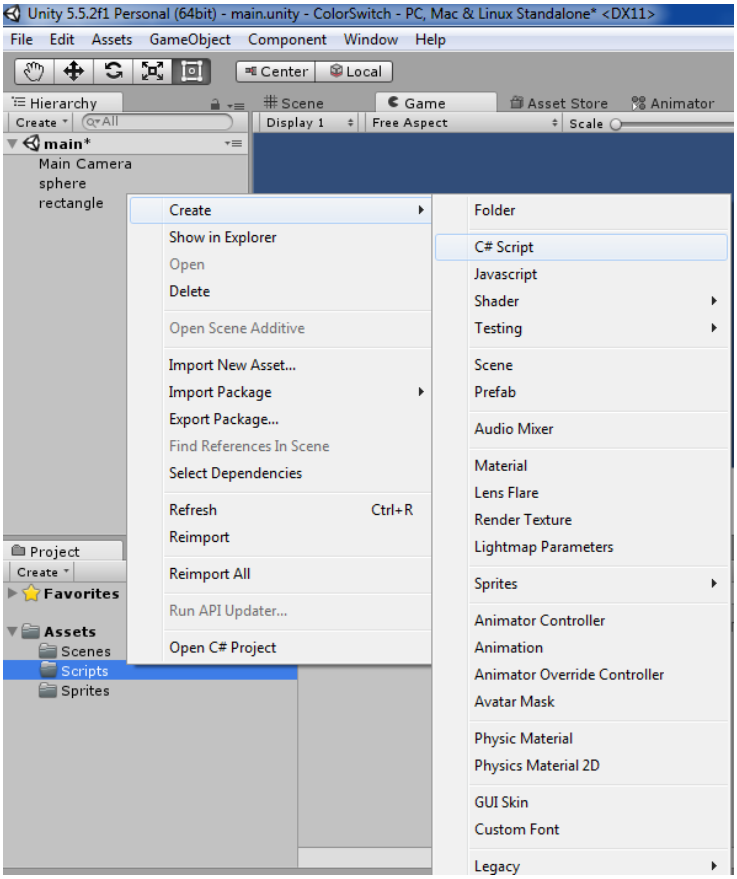
Slika 16 Loptica se zaustavlja pri dodiru sa objektom PointerOnBall

2.1. Kretanje loptice – igrača

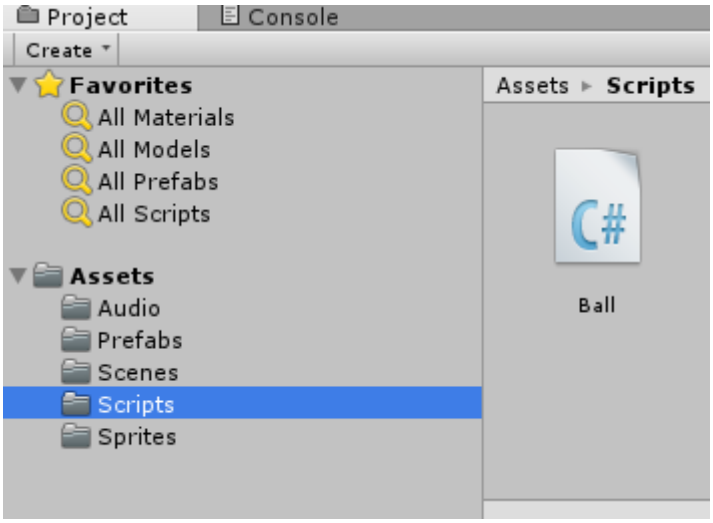
Sledeća funkcionalnost koja je jako važna kod igrice **Color Switch** jeste da omogućimo da se loptica svaki put kada kliknemo na taster miša/touch kreće nagore. Da bismo ovo omogućili kreiramo *C#* skriptu, u okviru foldera **Assets/*Scripts/*** na sledeći način: Desni klik na koren foldera

Scripts -> Create -> C# Script -> Ball.cs.

Skriptu ***Ball.cs*** pokrećemo preko *Visual Studija.* Osnovne metode koje kreira Unity su ***Start()***, koja se poziva prilikom pokretanja skripte i ***Update()*** metode. Umesto Update() metode, koristimo **FixedUpdate()** koja se uglavnom primenjuje kod vremenski osetljivih kodova ili za fiziku kao što je ovde slučaj.



Slika 17 Kreiranje C# skripte

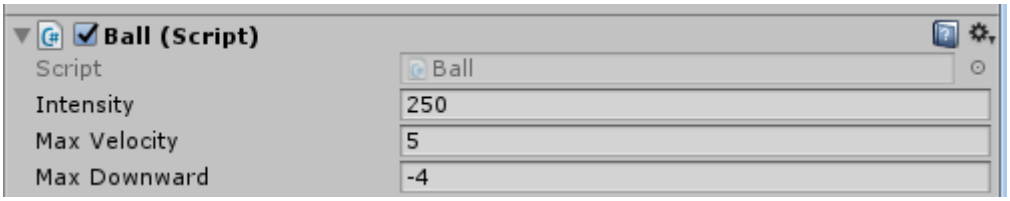


Slika 18 Ball.cs

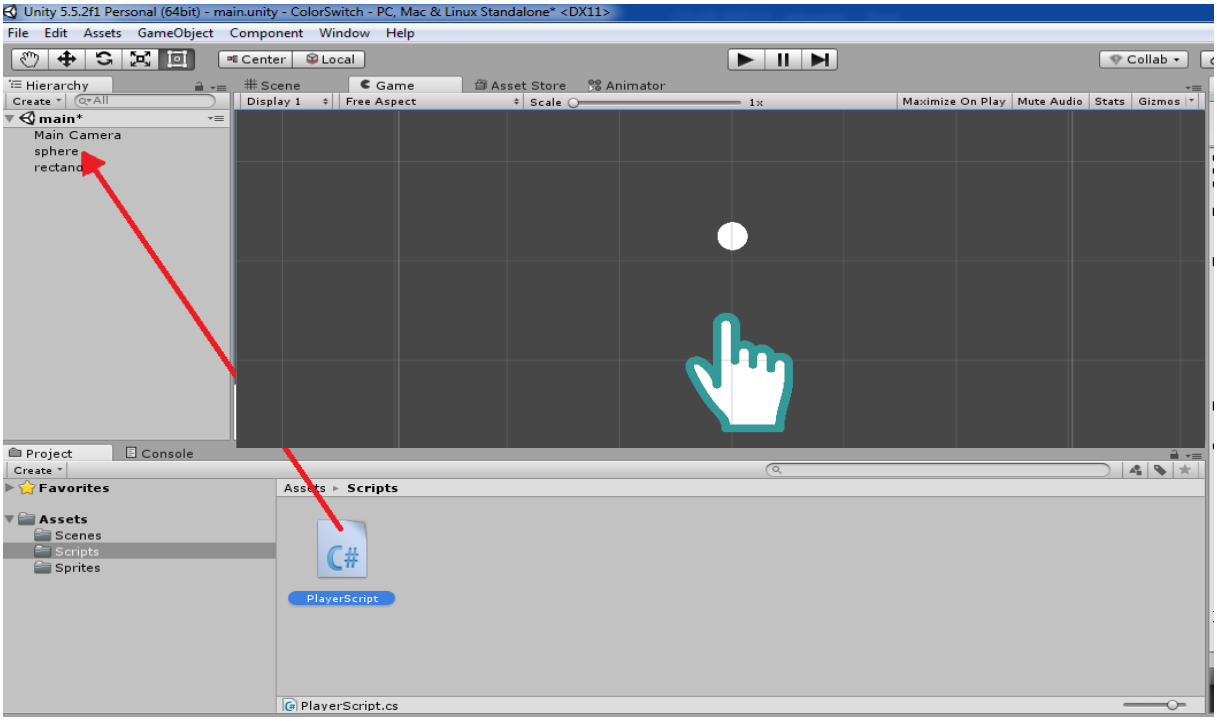
Skripta ***Ball.cs*** ima sledeći sadržaj:

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

public class Ball : MonoBehaviour
{

    public float intensity; //jačina sa kojom loptica odskače
    private Rigidbody2D rb2d; //referenca na kompnentu Rigidbody2D
    public float MaxVelocity ; // dokle najviše da promeni poziciju po y osi -skok
    public float MaxDownward;    // dokle da padne

    // Use this for initialization
    void Start()
    {
       rb2d = GetComponent<Rigidbody2D>();
    }

    void FixedUpdate()
    {
        if (Input.GetMouseButtonDown(0)) //ako kliknemo levim tasterom misa
        {
            //add force upward, or add impulse upward
            if (rb2d != null)
            {
                rb2d.AddForce(Vector2.up * intensity);
            }
        }
        if (rb2d.velocity.y > MaxVelocity) // da onemoguci lopti da skace previsoko
        {
            rb2d.AddForce(Vector2.down * 30);
        }

        if (rb2d.velocity.y <  MaxDownward) // max velocity kada pada y ispod 0 dokle moze
da pada
        {
            rb2d.AddForce(Vector2.up * intensity);
        }
    }

}
```

Sada je potrebno ovu skriptu dodeliti objektu ***BallPlayer,*** a to radimo tako sto skriptu prevučemo na objekat ili odabirom objekta pa **Add component** i nađemo skriptu po imenu.. U inspector prozoru dodeliti veličine trima promenljivama po vašem nahođenju ali na slici 19 je dat prikaz kako smo mi našli „zlatnu“ sredinu.



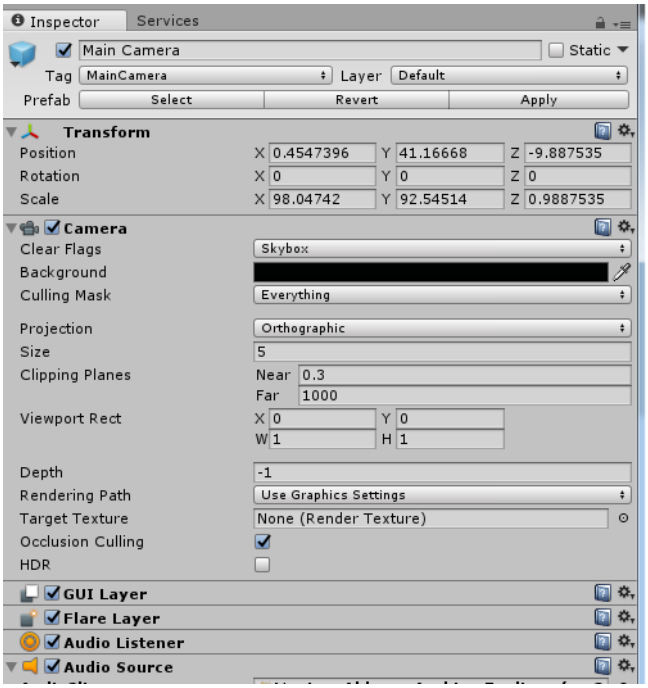
Slika 19 Podešavanje promenljivih za kretanje loptice



Slika 20 Prevlačenje skripte na objekat

Sada se loptica na klik miša kreće na gore, međutim nju ne prati kamera. Da bismo to obezbedili ***Main Cameru*** prevučemo na objekat ***BallPlayer***, čime taj objekat postaje roditelj objekata *Main Camera*. Na slici 21 je prikazano kako se ***Main Camera*** dodeljuje objektu ***BallPlayer*** čime postaje loptica roditelj kameri i time smo rešili problem praćenja loptice kamerom.

Slika 21 Main camera kao dete loptice



Slika 22 Main Camera

Podesiti da kamera **Main Camera** bude veličine otrpilike 5. To se radi tako što se promeni vrednost polja ***Size*** u prozoru Inspector.

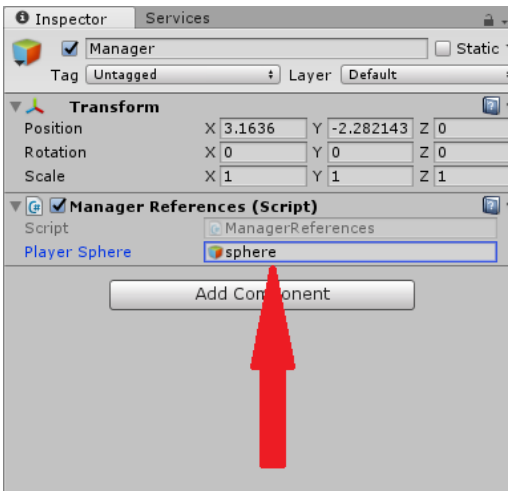
2.2. Menjanje boje loptice

Zbog boljeg vizuelnog efekta, boju pozadine, koja je plava, menjamo u **crnu**, na sledeći način: klikom na ***Main Camera*** u **Hiararchy** panelu, dobijamo mogućnost podešavanja pozadine u okviru ***Inspector*** panela. Tu okviru komponente ***Camera***, imamo ***Background*** sekciju gde biramo boju pozadine, postavite da bude **crna**.

Trenutna boja loptice je bela. Naš cilj je da omogućimo da loptica menja boje nasumično ali pošto ne zna koje boje sme da koristi mi moramo to da joj kažemo na osnovu predefinisanog intervala boja odnosno niza. Iz tog razloga kreiramo prazan objekat koji ćemo nazvati **Menager** *(Game Object -> Create Empty -> objekat imenujemo Menager).* Ovo je objekat koji će kontrolisati celu scenu i biće referenca na poene, na lopticu itd. Sada kreiramo skriptu ***MenagerReferences.cs*** u folderu ***Assets/Scripts/*** koju dodeljujemo objektu ***Menager***. Skripta izgleda ovako:

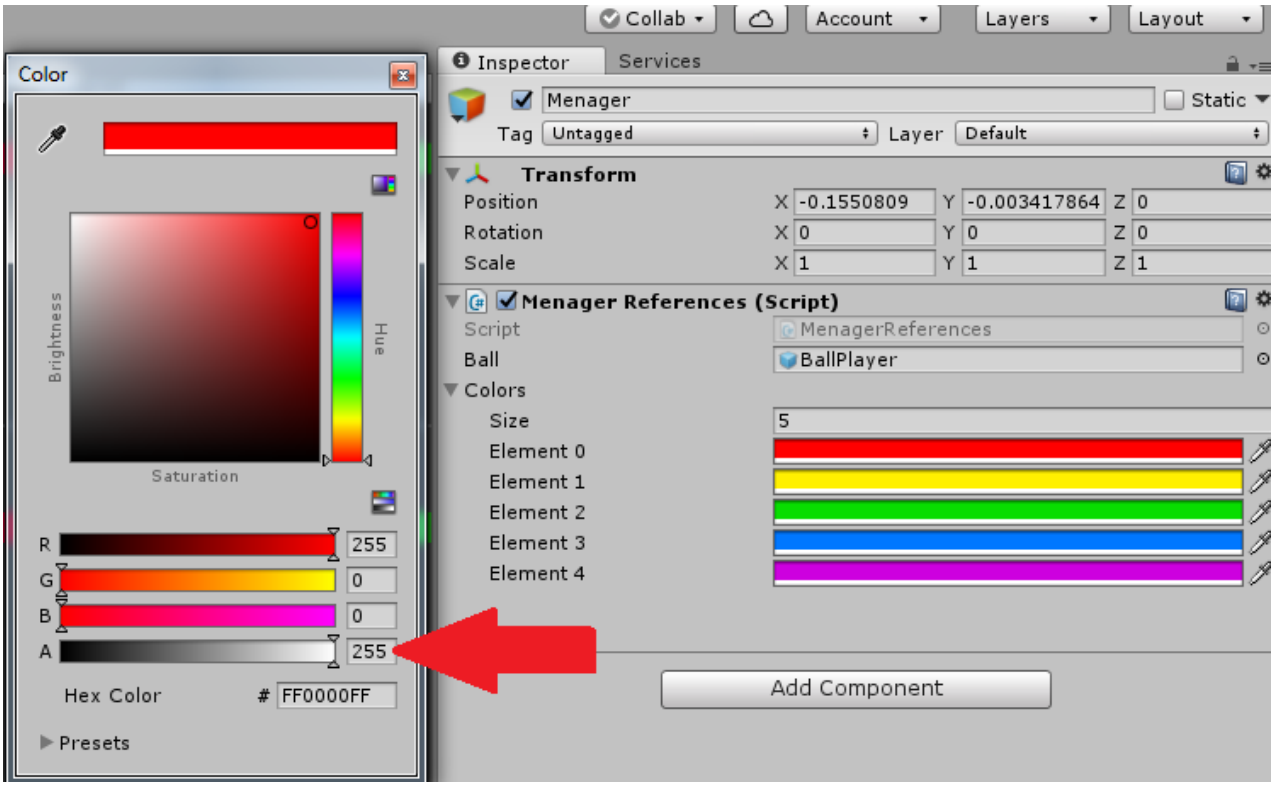
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MenagerReferences : MonoBehaviour
{
    public GameObject ball; //referenca na objekat sphere
    public Color[] Colors;     //referenca na niz boja
    // Use this for initialization
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Nakon sto u skripti navedemo referencu na neki objekat iz Unity programa, npr. referenca na objekat ***BallPlayer***, u okviru Inspector prozora te skripte pojavljuje se polje sa nazivom reference, gde je potrebno prevući objekat, na koji se ta referenca odnosi.



Slika 23 Dodavanje reference skripti

Nakon dodavanja reference na niz boja, u Unity programu, u okviru komponte koja definiše ***ManagerReferences*** skriptu, dobijamo mogućnost definisanja prvo veličine niza, a zatim i boja koje želimo da se nadju u nizu ***Colors***:



Slika 24 Dodavanje boja

Strelica na prethodnoj slici ukazuje da alfa uvek mora da ima vrednost **255** da bi loptica mogla da primi boju inače će imati boju crnu kao osnova i neće se primećivati na kameri. Sada je potrebno dodeliti boje objektima. Loptica treba da ima random boje, svaki put kada se igrica startuje. Da bismo to obezbedili modifikujemo skriptu ***Ball.cs*** na sledeci način:

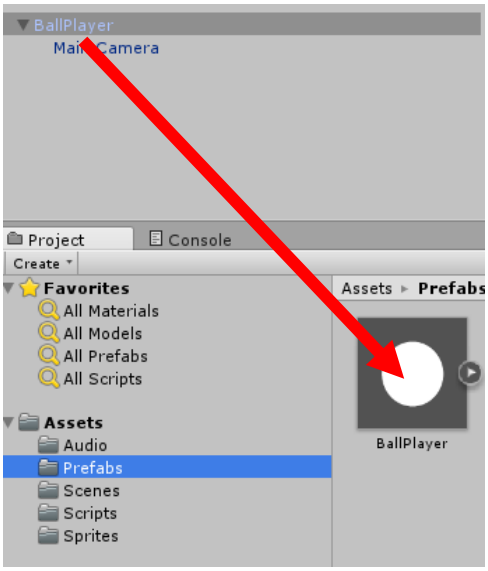
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Ball : MonoBehaviour
{
    public float intensity; //jačina sa kojom loptica odskače
    private Rigidbody2D rb2d; //referenca na kompnentu Rigidbody2D
    public float MaxVelocity ; // dokle najviše da promeni poziciju po y osi -skok
    public float MaxDownward;    // dokle da padne

    // Use this for initialization
    void Start()
    {
       rb2d = GetComponent<Rigidbody2D>();
       InitializeColor();
    }
    void FixedUpdate()
    {
        if (Input.GetMouseButtonDown(0)) //ako kliknemo levim tasterom misa
        {
            //add force upward, or add impulse upward
            if (body != null)
            {
                Rb2d.AddForce(Vector2.up * intensity);
            }
        }
```

```
        if (rb2d.velocity.y > MaxVelocity) // da onemoguci lopti da skace previsoko
        {
            rb2d.AddForce(Vector2.down * 30);
        }

        if (rb2d.velocity.y <  MaxDownward) // max velocity kada pada y ispod 0 dokle moze
da pada
        {
            rb2d.AddForce(Vector2.up * intensity);
        }
    }

    void InitializeColor()
    {
        //pravimo referencu na objekat Manager
              MenagerReferences refs =
              GameObject.Find("Menager").GetComponent<MenagerReferences>();
        if (refs!=null)
        {
              //broj boja
            int colorCount = refs.Colors.Length;
              //random boje u opsegu od 0 do maksimalnog broja boja
            int randomIndex = Random.Range(0,colorCount-1);
            //kreiranje nove random boje
            Color newColor = refs.Colors[randomIndex];
              //posto je loptica(sphere) Sprite, uzimamo tu komponentu i dodeljujemo joj
       boju
            SpriteRenderer render = GetComponent<SpriteRenderer>();
            render.color = newColor; //nova boja
        }
   }

}
```

2.3. Kreiranje Prefab elemenata

Kopiranje obejekata će sigurno proizvesti duplikate, ali osobine duplikata se moraju posebno menjati. Da bi se izbeglo ovako nešto, Unity podržava ***Prefeb assets*** tip sredstva koje omogućava da sačuvamo **GameObject** objekte zajedno sa komponentama i svojstvima.

Na kraju treba da kreiramo ***Prefab*** folder u ***Assets/Prefabs/*** kako bi tu smeštali kreirane objekte da ih ne bi duplirali i pojedinačno svakom podešavali opcije u Inspector prozoru prevlačenjem iz hijararhije projekta u folder prefabs dobijamo objekat sačuvan za ubuduće korišćenje na onoliko mesta koliko mi želimo sa predefinisanim opcijama. Sada Lopticu ćemo prevući u taj folder kako bi mogli kasnije istu lopticu za naredni nivo da samo prebacimo iz ovog foldera kao na slici 25.



Slika 25 Kreiranje Prefabs-a

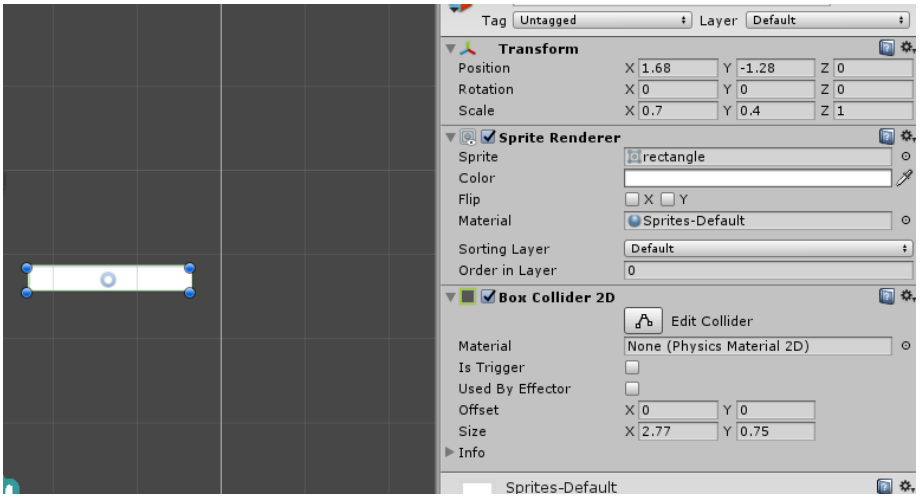
Ono što je Važno za Prefabs elemente jeste da ukoliko se nakon kreiranja prefab-a promeni neka vrednost objektu mora se odabrati u gornjem uglu opcija *„**Apply**“* kako bi se nove vrednosti dodelile svim istim objektima na sceni ukoliko želimo ako ne onda ostavljamo samo za taj element novu vrednost bez odabira apply opcije za sve.

3. Dodavanje drugog objekta

Dodaćemo običan ***Rectangle*** sprajt (ako ga niste dodali u prethodnom poglavlju) iz foldera **Assets/Sprites/** na scenu kako bi videli kako se loptica sudara sa drugim objektom i šta treba da objekat radi kao I podešavanja za objekte od kojih će se kasnije praviti razne druge prepreke.

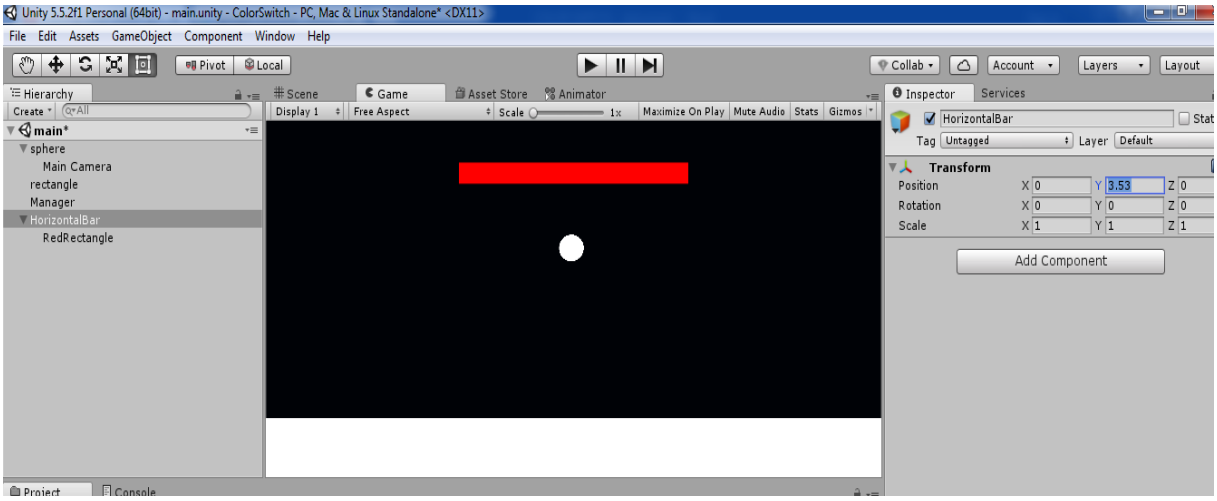
3.1. Kreiranje objekta za koliziju

Nakon što smo dodali ***rectangle*** treba da podesimo širinu i visinu pravougaonika kao na slici. Posle kada ovaj objekat dodelimo jednom objektu i kreiramo više onda možemo da ga obrišemo iz hijararhije.



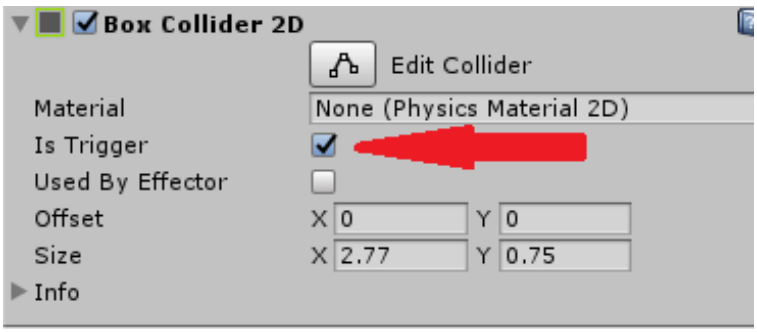
Slika 26 Podešavanja rectangle objekta

Zatim kreiramo prazan objekat ***HorizontalBar*** (*GameObject ->CreateEmpty).* Ovom objektu kao podobjekat dodajemo prethodno kreirani sprite ***rectangle,*** i imenujemo ga u ***RedRectangle***. Bojimo ga u crveno, a u okviru komponente ***Transform*** za taj objekat postavimo ***Position*** X:0 i Y:0. Zatim objekat ***HorizontalBar*** pozicioniramo iznad loptice kao na slici ispod:



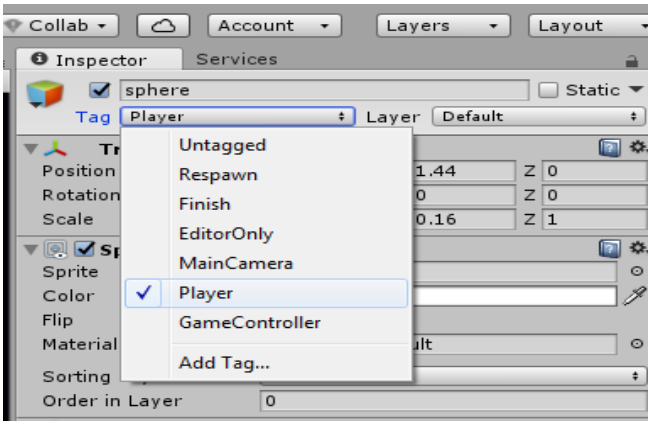
Slika 27 Dodavanje prve prepreke

Objektu ***RedRectangle*** sada dodajemo komponentu ***BoxCollider2D***, kako bi detektovali sudar sa lopticom. Kako bi loptica prolazila kroz objekat ***RedRectangle*** u komponenti *BoxCollider2D* potrebno je čekirati ociju ***Is Trigger*** da bi bila ***true*** .



Slika 28 Dodavanje BoxCollider 2D komponente objektu RedRectangle

Za objekat ***BallPlayer*** potrebno je postaviti tag na ***Player,*** Unity sam nudi u predefinisanim tagovima i tag ***Player*** ,ali i mi sami možemo da kreiramo tagove ako nam trebaju:



Slika 29 Za sphere tag Player

Sledeći korak jeste kreiranje skripte za koliziju sa drugim objektima i koristi se f-ja **OnTriggerEnter2D()**, koju dodajemo objektu ***RedRectangle,*** i imenujemo je **Assets/Scripts/*TriggerCollision.cs*** .

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

public class TriggerCollision : MonoBehaviour
{
    //za prepreke takodje koristimo boje definisena u MenagerReferences
    public ColorUsed colorIndex; //u Unity-ju imamo mogucnost selektovanja željene boje

    void Start()
    {
        //referenciramo se na skriptu ManagerReferences
        ManagerReferences refs =
GameObject.Find("Menager").GetComponent<ManagerReferences>();
        if (refs != null)
        {
            Color newColor = refs.Colors[(int)colorIndex];
            gameObject.GetComponent<SpriteRenderer>().color = newColor;//dodeljujemo
selektovanu boju objektu RedRectangle
        }
    }
```

```
    //funkcija koja detektuje koliziju izmedju prepreke na koju je postavljen triger,
    //i Collider2D objekta, u ovom slucaju loptice(koja ima Circle Collider2D komponentu)
    void OnTriggerEnter2D(Collider2D other)
    {
        //ako je objekat koji dolazi u koliziju loptica(koja ima tag Player)
        if (other.gameObject.tag == "Player")
        {
            //implementiramo logiku igre
            Color playerColor = other.gameObject.GetComponent<SpriteRenderer>().color;
//boja loptice
            Color thisColor = this.gameObject.GetComponent<SpriteRenderer>().color; //boja
prepreke
            if (playerColor != thisColor)
            {
                Debug.Log("GAME OVER!"); //ispis u konzoli a kasnije ćemo dodati šta će da
radi kada se sudari sa drugom bojom
            }
        }
    }
}
```

Verovatno će Vam bojiti u skriptu red gde je tip ***ColorUsed***. To je zato što nismo još definisali opseg boja koje smemo da koristimo odakle će on uzimati trenutnu boju objekta. Zato modifikujemo skriptu **MenagerReferences.cs** sa sledećim kodom koji dodajemo pre klase jer je u pitanju ***enum***. ***Enum*** predstavlja skup imenovanih integer promenljiva. Kreću od 0 i raste za +1. Voditi računa jer enum elementi dobijaju **integer** vrednosti iako su **imenovani** po **želji** i to istim redom kako su definisani, dakle prvi element ima vrednost indeksa 0 naredni 1... Nama su ovde boje pa ćemo definisati istim redom koji smo dodelili i **Menager** objektu.

```
public enum ColorUsed
{
    Red,
    Yellow,
    Green,
    Blue,
    Purple
}

public class MenagerReferences : MonoBehaviour
{
            .
            .
            .
}
```

3.2. Kretanje prepreke

Kako prepreka ne bi bila statična, potrebno je omogućiti njeno pomeranje levo i desno. Za to pišemo novu skriptu, koju dodeljujemo objektu ***HorizontalBar*** , roditelju objekta ***RedRectangle***, pod nazivom ***HorizontalMovement.cs*** .Ovom skriptom ćemo definisati koliko levo, odnosno desno će prepreka moći da se pomera.

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

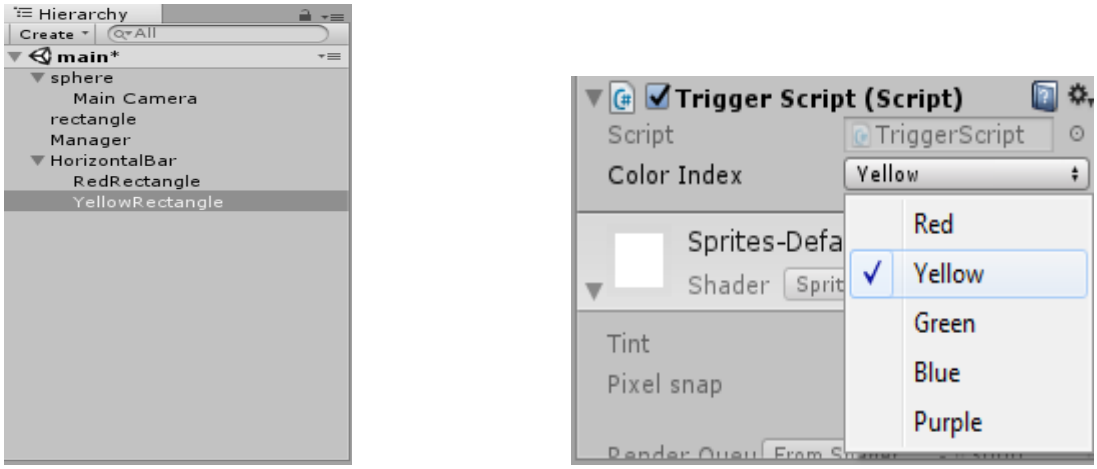
```
public class HorizontalMovement : MonoBehaviour
{

    public float rangeMax = 10;
    public float direction = 1; //ako je 1 pomera se desno, -1 levo
    public float speed = 1; //brzina varira od objekta do objekta I posle se dodaje sve
brže I teže za više nivoe
    public float initialPositionX; //početna pozicija objekta pošto se pomera samo po X
osi

    // Use this for initialization
    void Start()
    {
        initialPositionX = this.transform.position.x;
    }

    // Update is called once per frame
    void Update()
    {
        //pomeranje prepreke
        this.transform.position = new Vector2(this.transform.position.x + (direction *
Time.deltaTime * speed),this.transform.position.y);

        if (this.transform.position.x > initialPositionX + rangeMax)
        {
            direction = -1;
        }
        if (this.transform.position.x < initialPositionX - rangeMax)
        {
            direction = 1;
        }
    }
}
```

4. Kreiranje ostalih prepreka

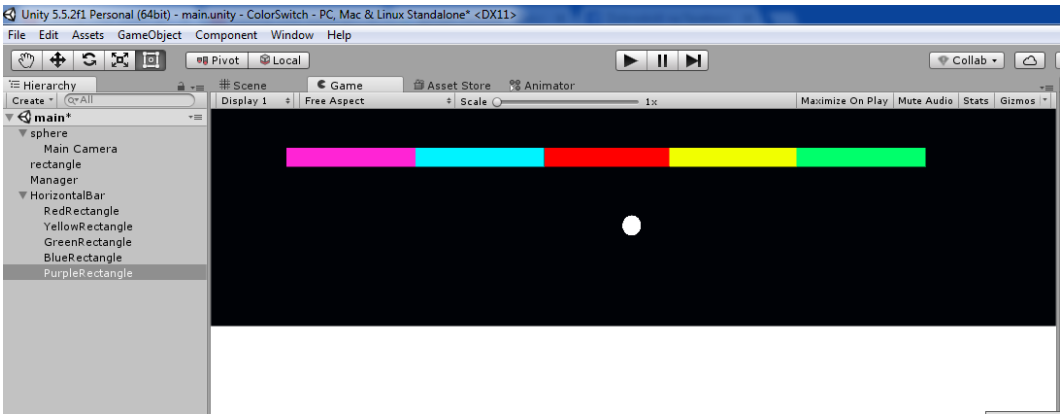
Sada je potrebno dodati prepreke, tako da loptica može prolaziti samo kroz prepreke one boje koje su iste kao ona. Ukoliko se sudari za objektom drugačije boje od sebe dolazi do **Game Over**-a I redirektuje se na **Restart** Scenu. U ovom poglavlju ćemo opisati različite prepreke kako ih kreiramo I kako od njih posle nadograđujemo druge složenije objekte/prepreke.

4.1. Horizonal Bar objekat

Sada kreiramo više prepreka dupliranjem ***RedRectangle*** objekta, onoliko puta koliko imamo definisanih boja ili koliko mi želimo da prepreka ima različitih boja. Za svaki nov objekat preimenujemo u ime boje koju predstavlja radi lakšeg prepoznavanja npr. ***GreenRectangle***,a u okviru **TriggerCollision.cs** komponente, selektujemo boju koja odgovara postavljenoj boji objekta u ovom slučaju **green**, i pozicioniramo ih na sceni jedan pored drugog:

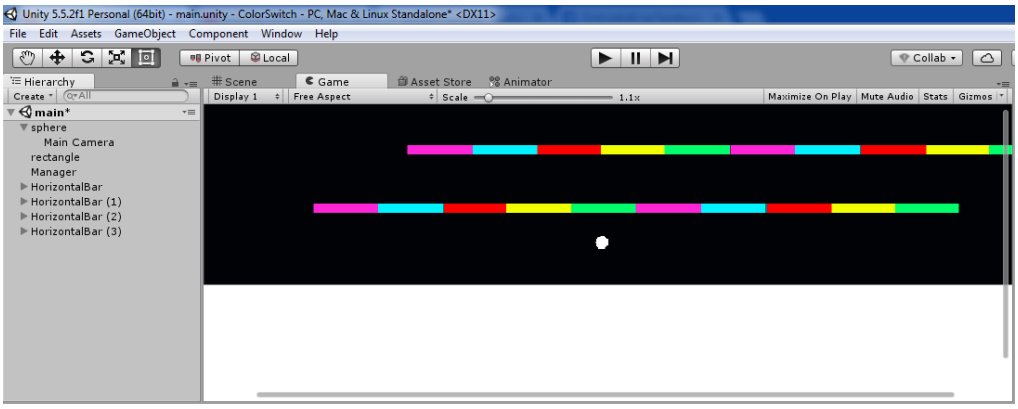


Slika 30 Novi objekat Rectangle



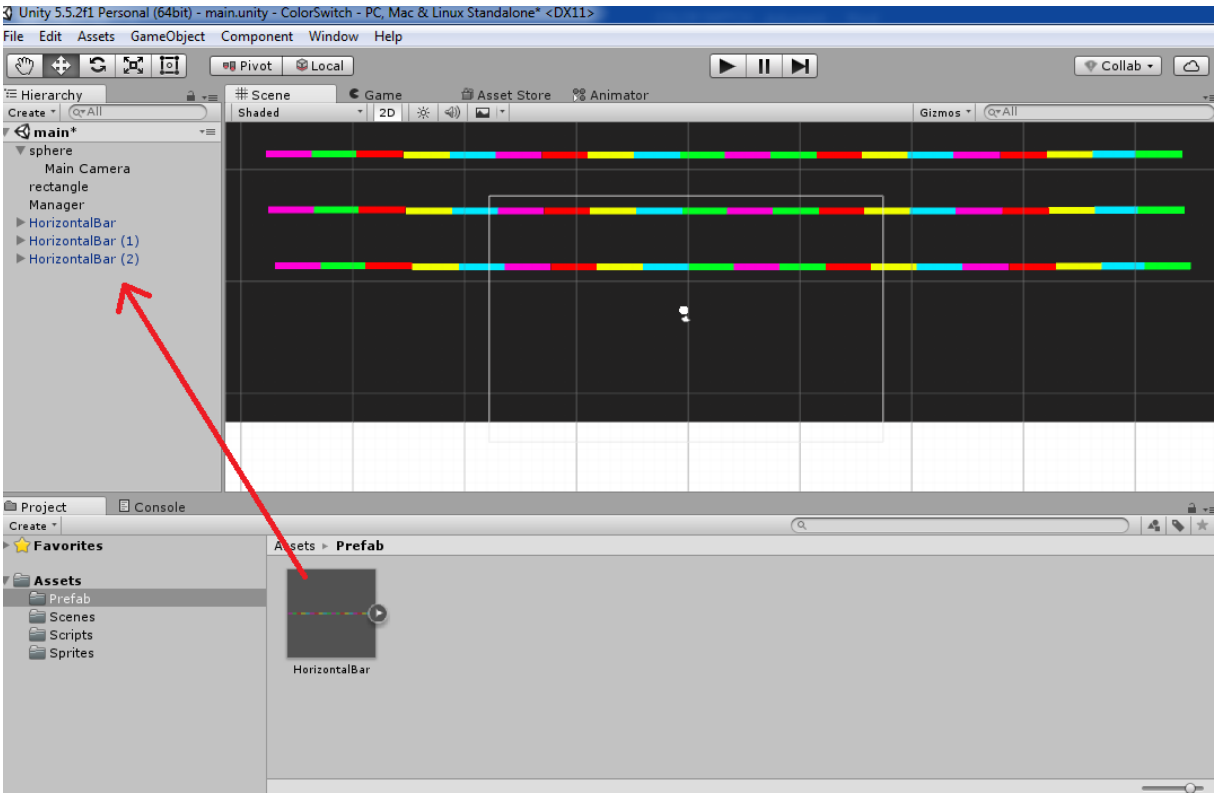
Slika 31 Izgled scene nakon dodavanja vise Rectangle objekata

Isto možemo duplirati više ***HorizontalBar*** objekata, i dobiti scenu sličnu ovoj:



Slika 32 Scena sa vise HorizontalBar objekata

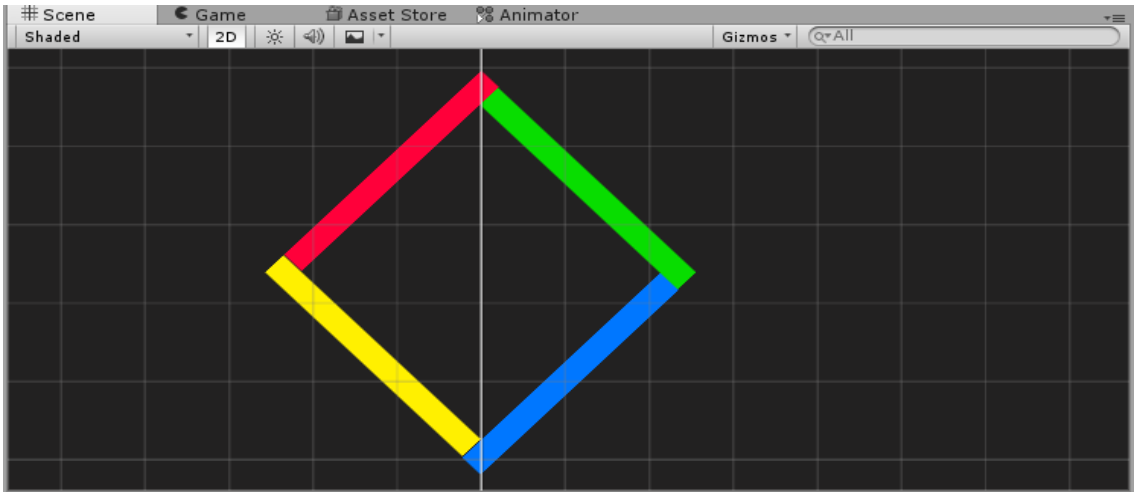
Na kraju selektujemo ceo objekat ***HorizontalBar*** i prevučemo ga u ***Assets/Prefabs/*** folder a zatim možemo da iz Prefab-a dodamo koliko želimo još istih objekata na scenu ali sa drugačijom brzinom što se menja u roditelju objekta dece.



Slika 33 Primer vi[e prefab objekata

4.2. Rectangle - Pravougaonik

Spajanjem ***rectangle*** sprite-ova iz foldera ***Asssets/Sprites/*** različitih boja možemo formirati raznovrsne objekte, kao npr **Rectagle**. Prvo kreiramo prazan objekat , i imenujemo ga Rectangle, u njega prekopiramo 4 posebna rectangle sprite-a iz foldera sa sprajtovima i postavimo im različitih boje.Treba voditi računa da boje budu iz opsega boja koje smo dodelili **Menager** objektu kako bi loptica i boje unutar objekta mogle da se podudaraju. **Nijanse boja ne bi smele da se razlikuju**! Selektovanjem svakog sprite-a pojedinačno menjamo Rotation po Z osi na 45(-45), i pozicioniramo ih tako da formiraju pravougaonik, kao na slici:

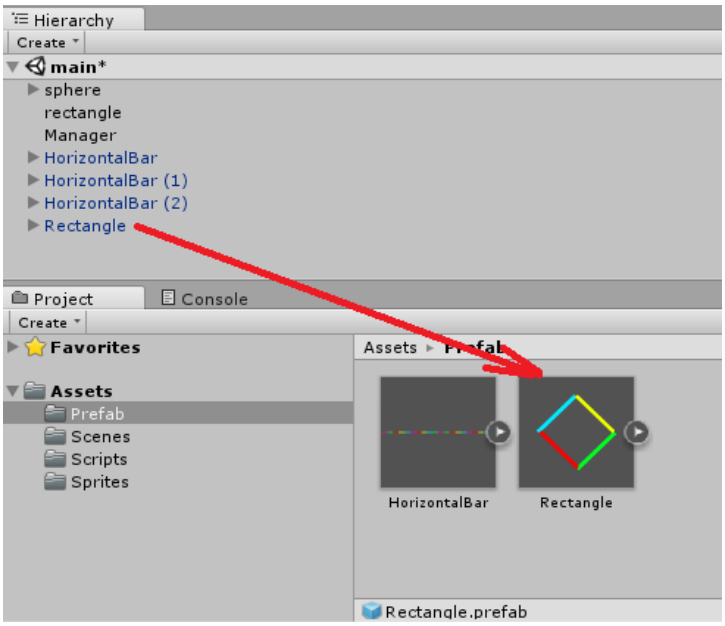


Slika 34 Pravougaona prepreka

Sada želimo da kreiramo mogućnost pravougaoniku da se okreće oko svoje ose kako bi mogle sve boje da u jednom trenutku dođu do loptice . Zato Kako bi se ovaj objekat rotirao, pravimo skriptu koja to omogućava pod nazivom ***RotationMovemen.***cs i po njenom ažuriranju dodajemo je na objekat ***Rectangle*** prevlačenjem*:*

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

public class RotationMovement : MonoBehaviour {

    public float RotationSpeed = 1;

       // Update is called once per frame
       void Update ()
    {
        //za rotaciju romba oko z ose
        Vector3 vec = new Vector3(0, 0, 1+RotationSpeed );
            this.transform.Rotate(vec);
       }
}
```

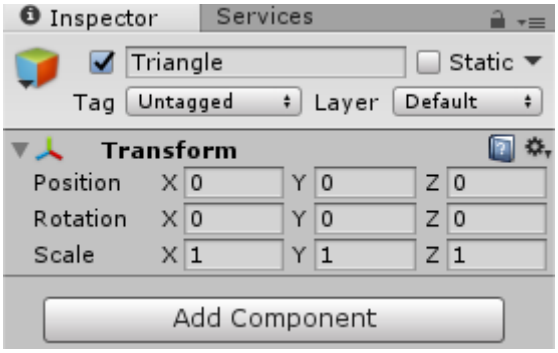
NAPOMENA: Kako bi se pravougaonik rotirao oko svoje ose možda Vam neće centar biti odmah podešen na vrednosti 0 ,0, 0 pa zato morate to ručno proveriti, jer kada se kreira objekat iz više delova Unity ne može sam da prepozna njegov centar rotacije.U Unity-ju je potrebno podesiti da se **Rectangle** rotira oko odredjene tačke pošto mu dodeljujemo skriptu za Rotiranje, iz tog razloga kopiramo **PRIVREMENO** objekat ***BallPlayer*** u **Rectangle** objekat, tako što uradimo desni klik na objekat i **Copy (Ctrl+C),** kreiranu kopiju prebacimo u objekat **Rectangle, uklonimo MainCameru** (da ne bi pravila problem oko pomeranja loptice) i postavimo sve pozicije po **x,y i z na 0**, nakon čega **sva 4 objekta** koja čine **Rectangle** objekat pozicioniraju se tako da im centar bude ***BallPlayer kopija***. Posle ovog podešavanja, možemo obrisati lopticu kopije i vratiti Circle objekat na mesto koje želite. Ovo radimo samo kako bi videli cenatr kruga i mogli da ga pozicioniramo za rotaciju, bez ovog koraka Vaš krug će se okretati oko cele kamere i bice razbacano i neuredno na ekranu.Za Rectangle je potrebno kreirati **prefab**, slično kao što smo uradili u za Horizontal Bar. Dakle u okviru foldera Prefab, jednostavnim prevlačenjem objekta iz hijararhije kreiramo prepreku Prefab.



Slika 35 Kreiranje Rectangle prefab-a

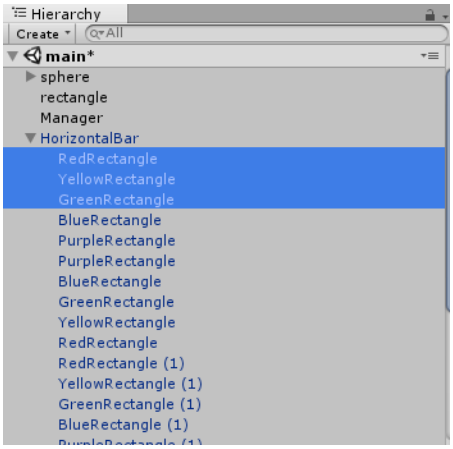
4.3. Triangle – Kreiranje prepreke u obliku trougla

Ono što sledi je kreiranje prepreke u obliku trougla. U Hiararchy panel-u kreiramo prazan objekat pod nazivom ***Triangle*** *(desni klik –> Create Empty->Triangle)*. Sada podesimo ***Transform*** komponentu u Inspector panel-u za ovaj objekat , tako sto Position postavimo na 0, kao na slici:

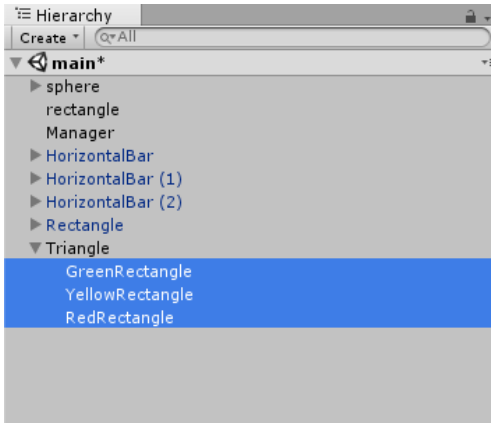


Slika 36 Pozicija se postavlja na 0

Kako bi formirali trougao, kopiraćemo već korišćene objekte iz Horizontal Bar-a (kopiramo tri objekta različite boje: Red, Yellow and Green ili boje koje vi želite), i dodajemo ih u objekat ***Triangle*** (Triangle postaje roditelj dodatim objektima).

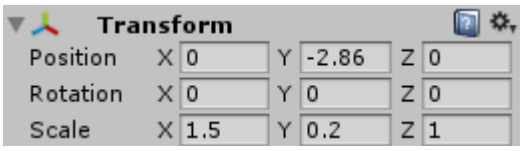


Slika 38 Selektujemo objekte koje kopiramo



Slika 37 Kopirane objekte dodajemo u Triangle

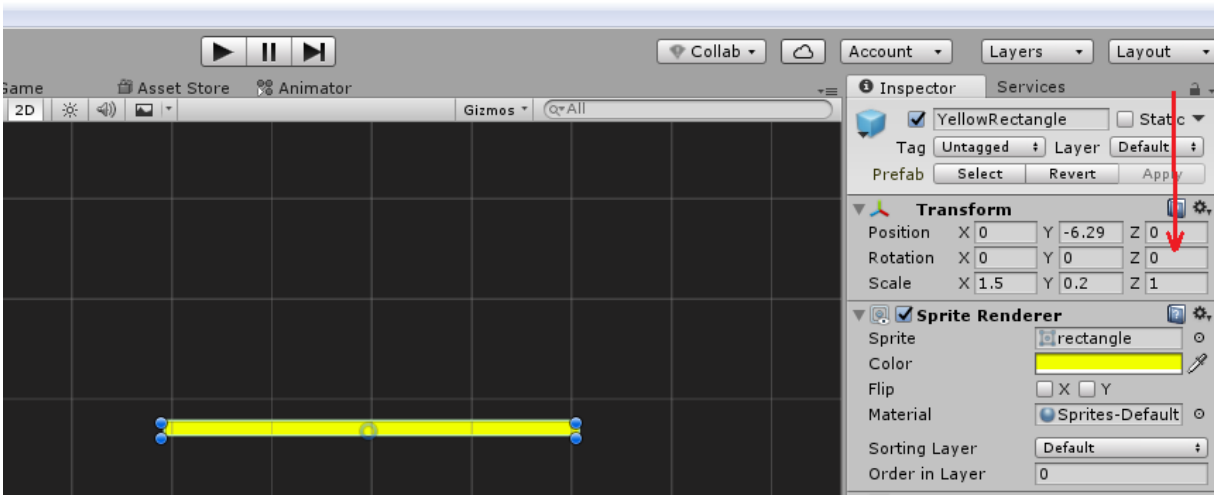
Selektujemo sva tri prekopirana objekta i podesimo ***Scale*** u ***Transform*** componenti na sledeći način:



Slika 39 Podesimo Scale kao na slici

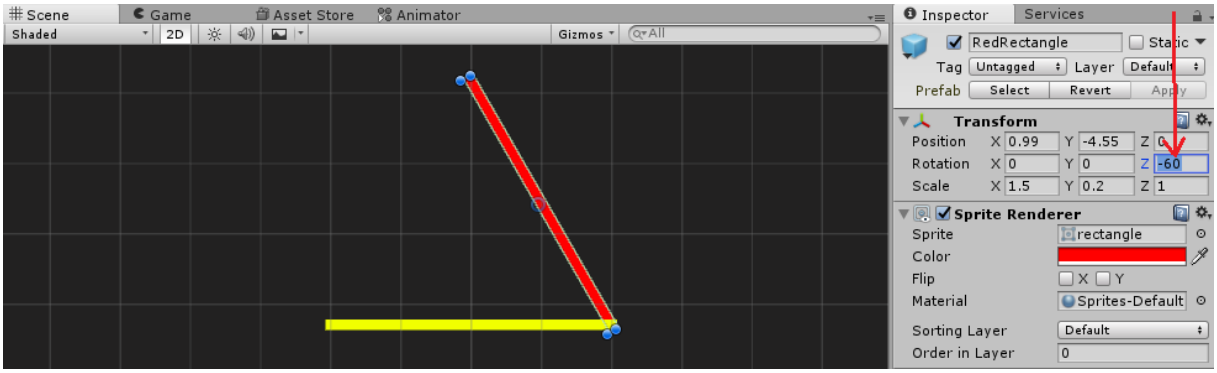
Kako bi napravili trougao od tri boje, za svaku boju podesimo rotaciju. Sledite naredne korake:

- Selektovanjem YellowRectangle objekta rotaciju postavimo na 0:



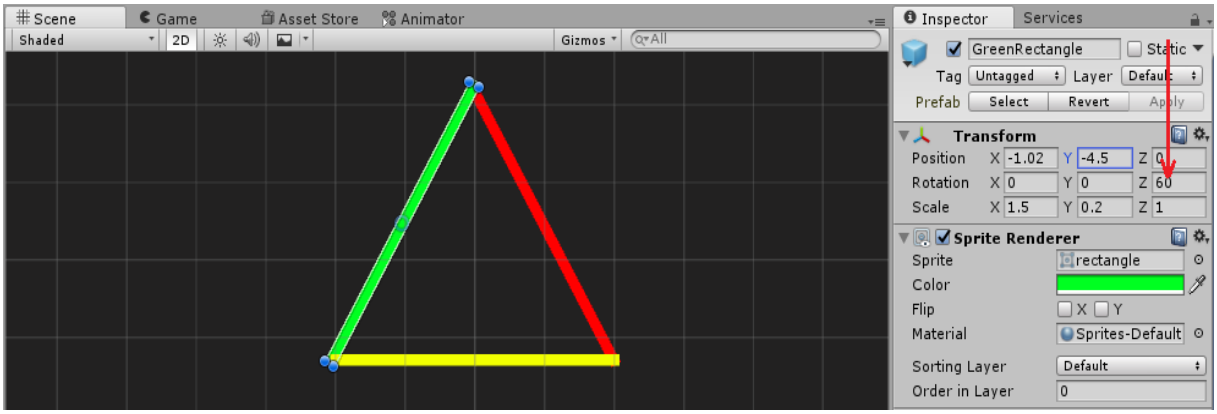
Slika 40 korak 1

- Selektovanjem RedRectangle objekta postavimo rataciju na -60, pozicioniramo tako da dobijemo izgled kao na slici ispod:



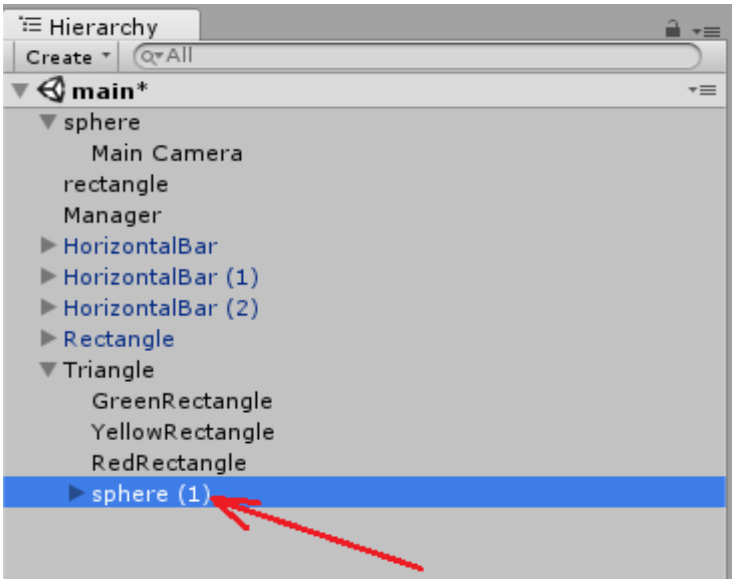
Slika 41 Korak 2

- Selektovanjem GreenRectangle objekta postavimo rataciju na 60, pozicioniramo tako da dobijemo izgled kao na slici ispod:



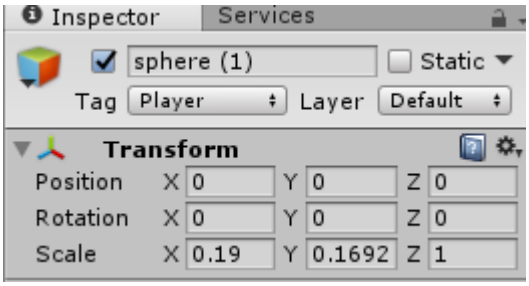
Slika 42 Korak 3

- Kako bi se trougao rotirao, koristimo isti trik kao i za Rectangle. Kopiramo objekat sphere u Triangle:



Slika 43 Korak 4

I postavimo pozicije sphere objekta na 0.

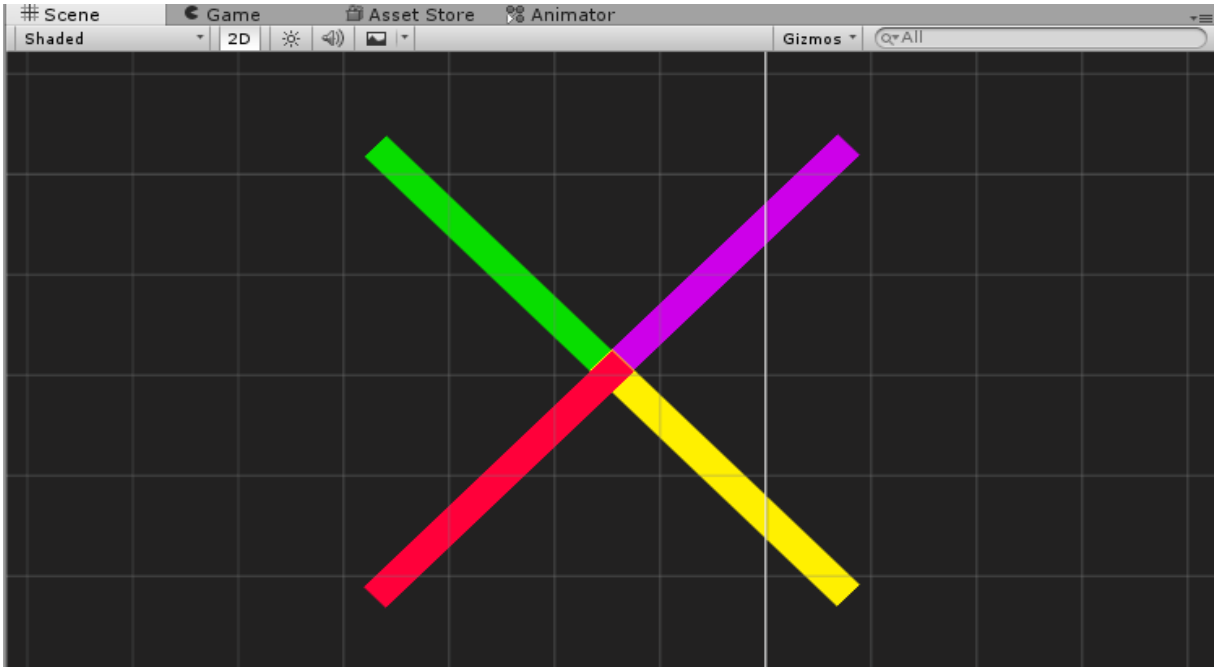


Slika 44 Pozicije postavimo na 0

Nakon toga pozicioniramo ***Triangle***, tako da loptica bude u centru, kako bi to bio centar oko koga se trougao rotira. Zatim brišemo ***BallPlayer*** objekat (**objašenje detaljno u Napomeni u poglavlju 4.2.**). Objektu ***Triangle*** dodelimo već kreiranu skriptu ***RotationMovement.cs***. Možemo da promenimo polje ***speed*** ako želimo da se kreće sporije ili brže. Ostalo je da trougao pozicionirate na scenu iznad *HorizontalBar*-ova ili možete kasnije da ga dodate ali da bi objekat sačuvali isto ga ubacujemo u ***Assets/Prefabs/*** kako bi kreirali Prefab objekat.

4.4. Cross – Prepreka u obliku X / krsta

Sledeća prepreka koju kreiramo će imati oblik ***krsta***. Kao i kod objekta ***Triangle***, kopiramo 4 pravougaonika (boje po želji) iz ***HorizintalBar***-a, I formiramo objekat ***Cross***, koji pozicioniramo iznad objekta *Rectangle* ili gde god želiter svakako iz Prefab foldera na kraju možete da ih postavljate gde vi želite:



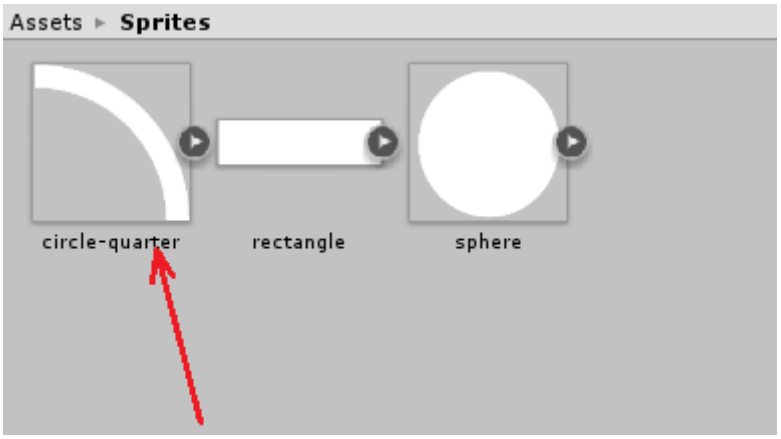
Slika 45 Cross objekat

Postupak dodavanja ***BallPlayer*** objekta u ***Cross*** objekat se ponavlja kao kod prethodnih objekata (**glava 4.2.**) zbog nalaženja centra za rotaciju koje smo kreirali, isto dodajemo i skriptu ***RotationMovement.cs*** Cross objektu.

Na kraju objekat prevlačimo u ***Assets/Prefabs/*** folder I time kreiramo Prefab objekat. Vodite računa samo kod Cross objekta da loptica ne prolazi kroz centar već ga malo pomerite u levo ili u desno kako bi BallPlayer prolazio kroz obojene delove Cross-a.

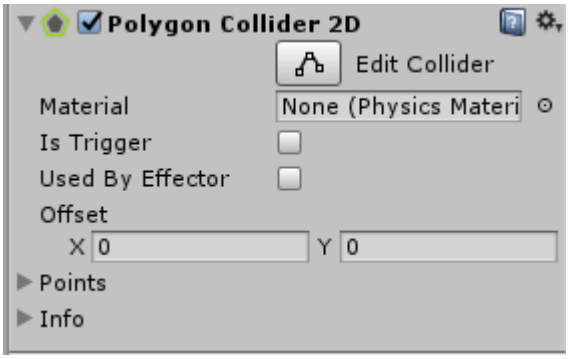
4.5. Circle – Prepreka u obliku kruga/sfere

Postupak kreiranja okruglih objekata u ovom slučaju kruga je sličan samo treba voditi računa da imate četvrtinu kruga koju spajate u krug rotiranjem I dipliranjem. U folder ***Assets/Sprites/*** importujemo sprite pod nazivom ***circle-quarter****:*

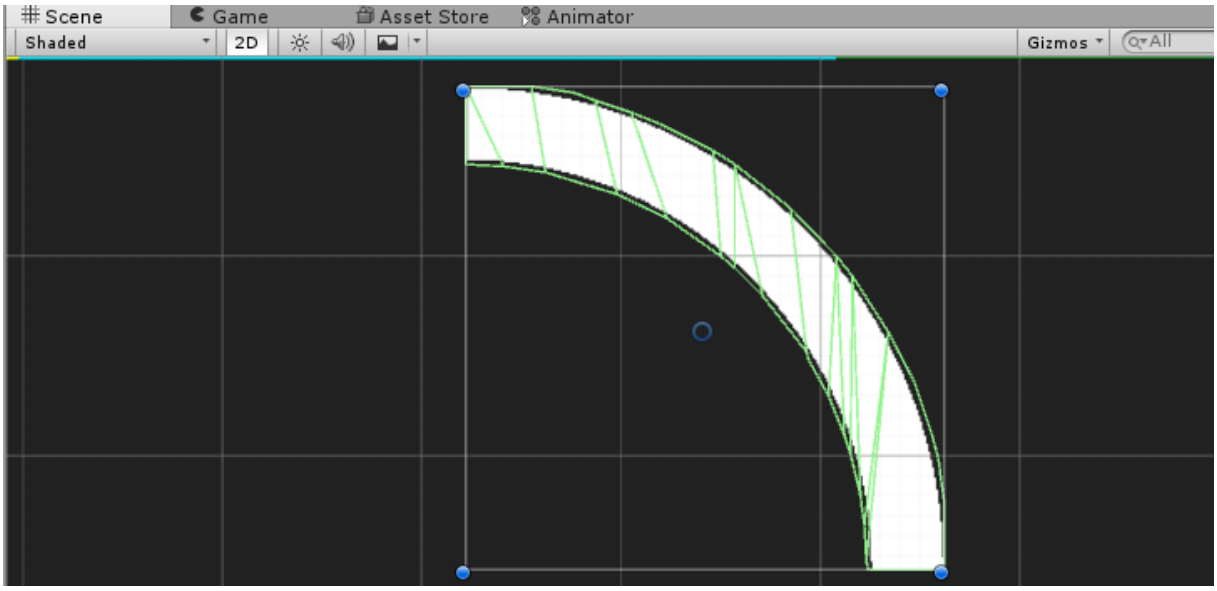


Slika 46 Importovanje sprajta circle

Prebacimo sprite u ***Hiararchy*** panel, i dodamo mu komponentu ***PolygonCollider2D*** *:*

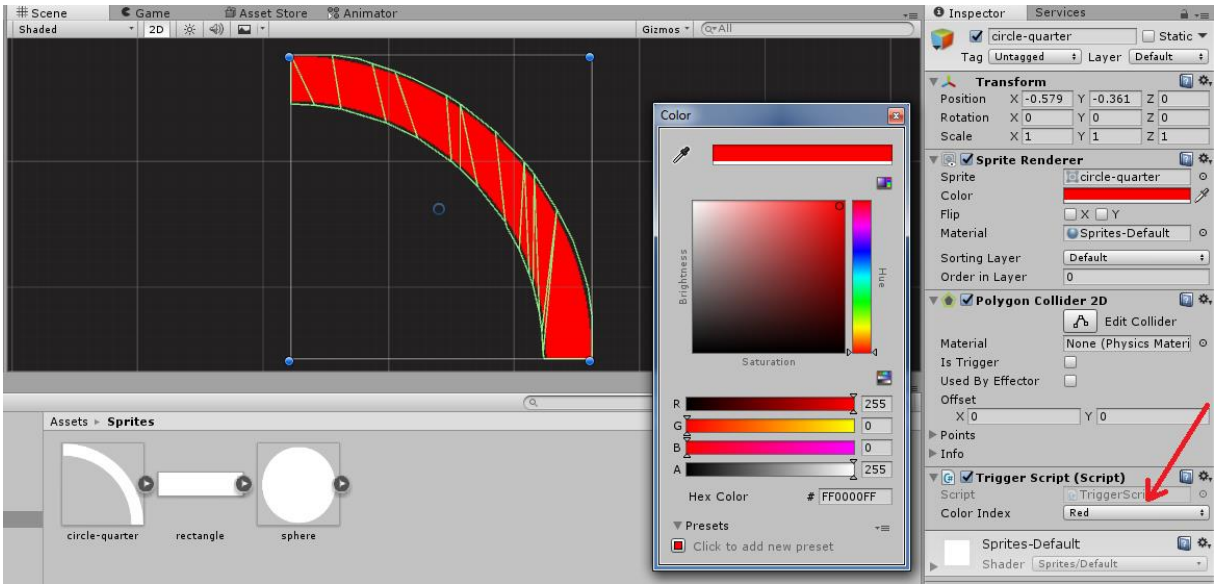


Slika 47 Komponenta Polzgon Collider 2D



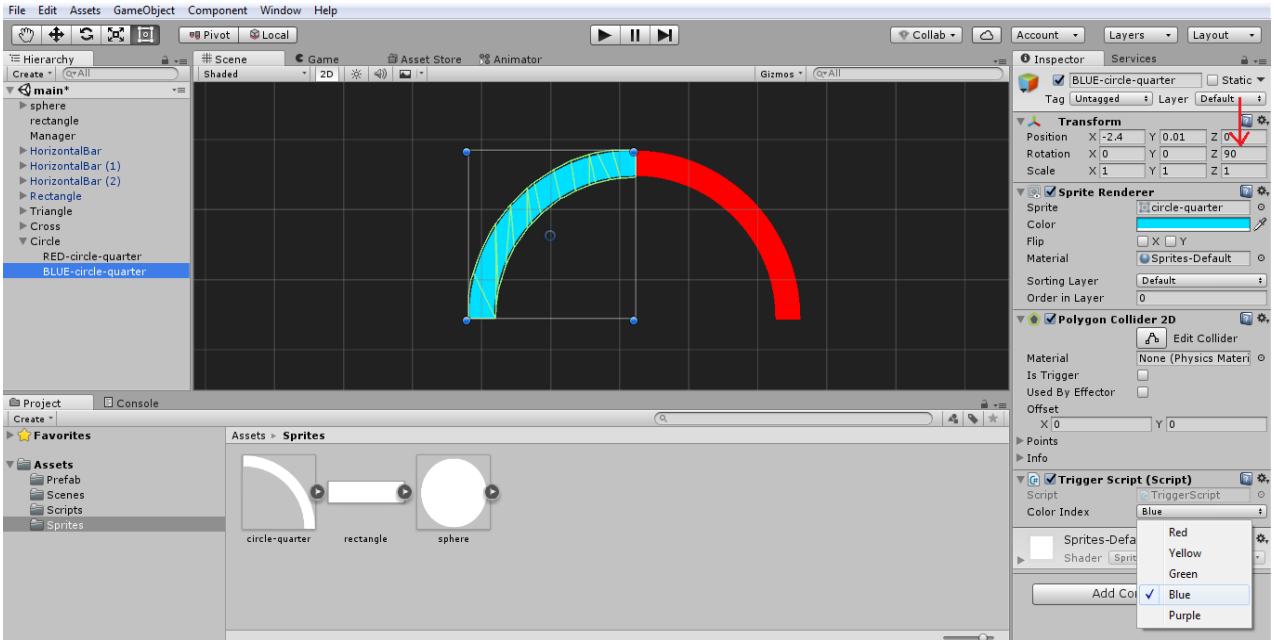
Slika 48 Sprite nakon dodavanja Polygon Collider komponente

Ovom sprite-u dodajemo ***TriggerScript.cs*** skriptu i biramo crvenu boju, kao na slici:

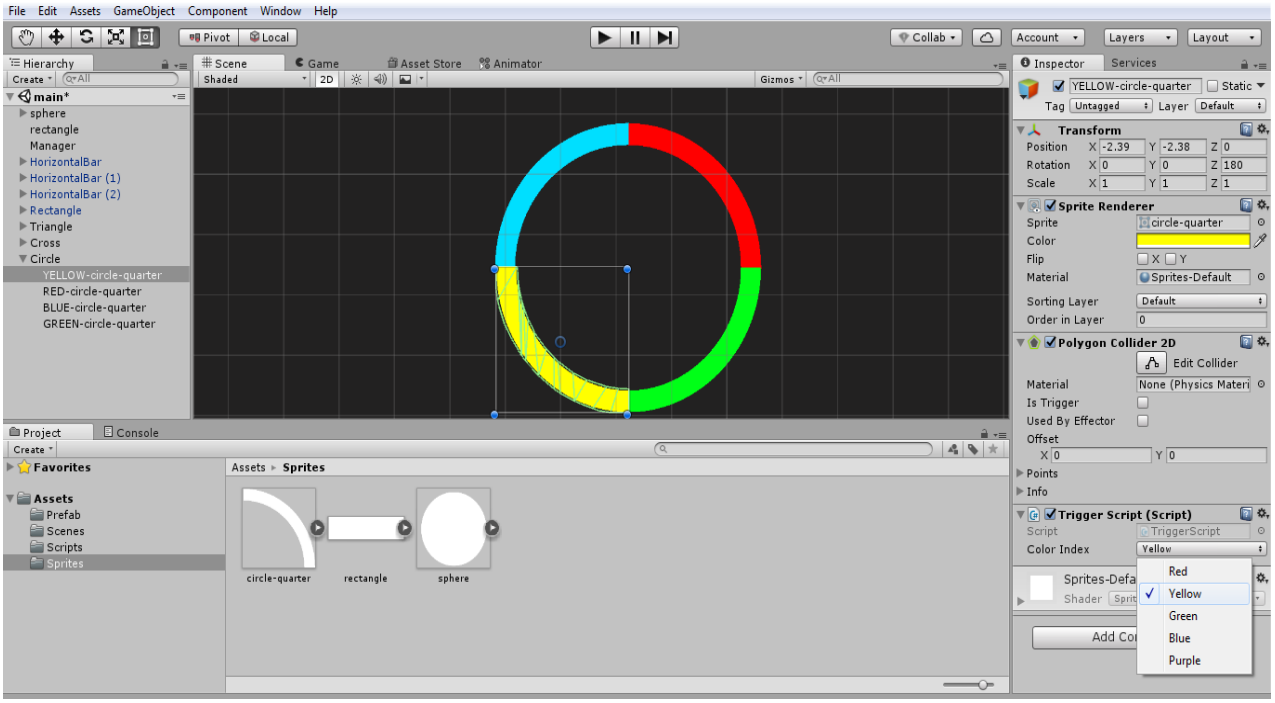


Slika 49 Dodela boje sprite-u i podešavanje

Naš cilj je da napravimo objekat kruga, spajanjem više ***circle*** sprit-ova. Iz tog razloga, kao i u prethodnim poglavljima, kreiramo prazan objekat pod nazivom ***Circle***. U njega prevlačimo *circle-quarter* sprite, koga preimenujemo u ***Red***. Napravimo još tri duplikata ovog sprite-a, i dodelimo im različite boje i podesimo rotaciju:



Slika 50 Dodavanje sprite-a plave boje



Slika 51 Dodavanje ostalih sprite-ova

Postupak dodavanja ***BallPlayer*** objekta, kako bi se objekat ***Circle*** rotirao, isti je kao kod prethodnih objekata(**objašnjenje detaljno u Napomeni u poglavlju 4.2.**). Na kraju objektu ***Circle*** dodeljujemo skriptu ***RotationMovement***. Objekat Circle prebacimo na kraju u **Prefab** kako bi kreirali kasnije složenije objekte od njega.

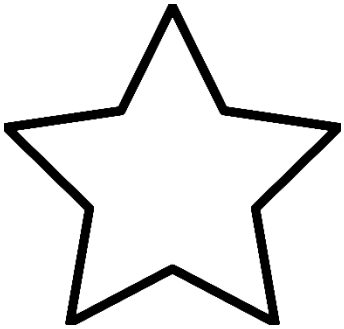
NAPOMENA: Sve druge objekte/Prefabs koje sada kreirate možete po svojoj želji I mašti. Ako želite da se rotira dodajete ***RotationMovement***, ako želite da ide horizontalno dodajete ***HorizontalMovement*** skripte.

5. Poeni i Menjanje boje loptice

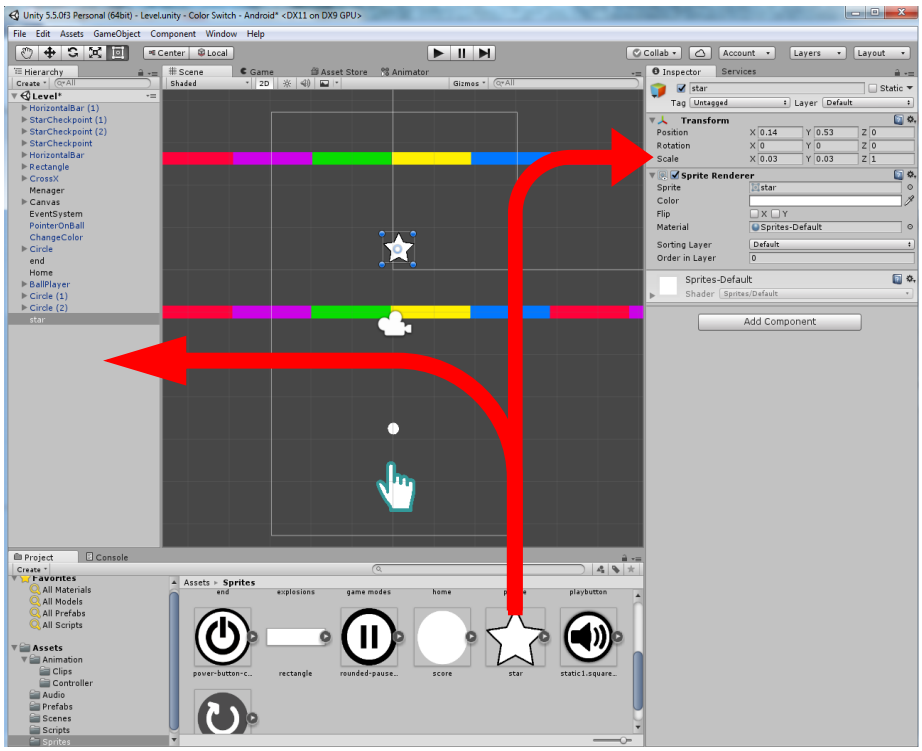
U ovom poglavlju ćemo se pozabaviti objektima koje loptica prikuplja u svakom nivou. To su poeni i color switch objekat kojim se menja boja u zavisnosti od predefinisanih vrednosti.

5.1. Prikupljanje poena - Score points

Nakon pojedinog pređenog objekta stavićemo da se prikuplja po jedan poen eventualno dva i taj objekat ćemo u ovom slučaju staviti da bude zvezda.



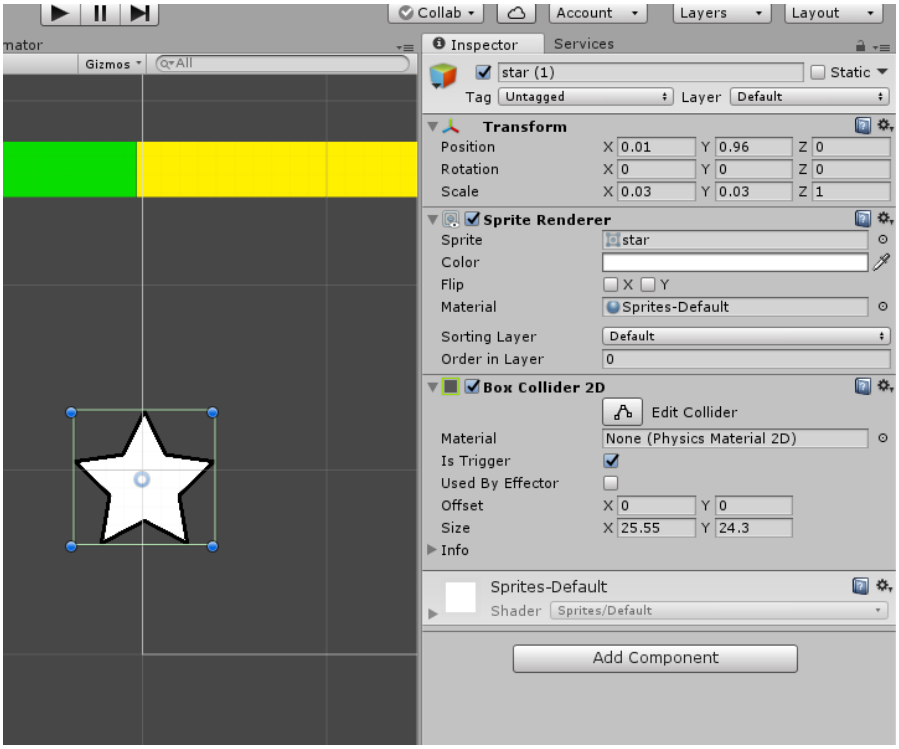
Slika 52 Zvezda



Slika 53 Prevlačenje zvezde na scenu

U folderu ***Assets/Sprites/*** nalazi se sprite „**Star**“ levim držanjem miša prevući zvezdu u hijararhiju sa leve strane Unity prozora i u Inspector prozoru podesiti polje scale da veličina bude **x=0.03 , y=0.03**. Zatim prevlačenjem mišem namestiti na željenu poziciju.

Sledeće što treba da podesimo u Inspector prozoru jeste da se doda komponenta **Box Collider 2D** kako bi loptica mogla da se sudari sa zvezdom ali uz to treba i da postavimo vrednost **Is Trigger** na true jer loptica treba da prodje kroz zvezdu i da ona nestane pri koliziji a ne da se odbije od nje.



Slika 54 Dodavanje Box Collider 2D komponente

Zatim kreiramo novu skriptu u folderu Assets/Scripts/ i nazvaćemo je **StarScore.cs** i preko nje ćemo da namestimo da loptica u dodiru sa zvezdom je uništava i uvećava brojač poena za 1. Skriptu prevući kao što je objašnjeno u prethodnim poglavljima na objekat zvezde u hijararhiji ili preko Add component u Inspector prozoru.

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

public class StarScore : MonoBehaviour
{
    public int scoreAmount = 1; //da se pri sakupljanju zvezdice uvek uvecava za 1

    void OnTriggerEnter2D(Collider2D other) //provera ada li je doslo do kolizije
    {
        if (other.gameObject.tag == "Player") // da li je objekat igrac
        {
            //referenca na menadzera preko koga  se povecava score
           MenagerReferences refs =
           GameObject.Find("Menager").GetComponent<MenagerReferences>();
            refs.IncreaseScore(scoreAmount); //da u menadzeru poveca vrednost poena
            Destroy(gameObject); // da unisti zvezdu
        }
    }
}
```

A sam metod za uvećanje poena ćemo definisati u skripti **MenagerReferences.cs** jer toj skripti mogu lako svi objekti da pristupe pa je bolje uvećavanje poena stavljati u skriptu koja je dostupna svima kao sto je menadžer.

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

public enum ColorUsed
{
    Red,
    Yellow,
    Green,
    Blue,
    Purple
}
public class MenagerReferences : MonoBehaviour
{

    public GameObject ball; //referenca na objekat loptice
    public Color[] Colors; //niz boja
    public int score ; //brojac za poene

    void Start ()
    {
        score = 0;

    }

    public void IncreaseScore(int amount)
    {
        score += amount;
       //posle se dodaje da prikaze score

    }
}
```

Ako želite možete dodeliti zvezdi I skriptu za rotaciju kako bi se okretala **RotationMovement.cs.** Zatim kada završimo sa kreiranjem zvezde možemo je levim držanjem miša prevući u Assets/Prefabs/ I kreirati prefab koji posle moeže da koristite kao I prethodno kreirane objekte više puta u igrici I scenama.

5.2. Prikazivanje poena na ekranu

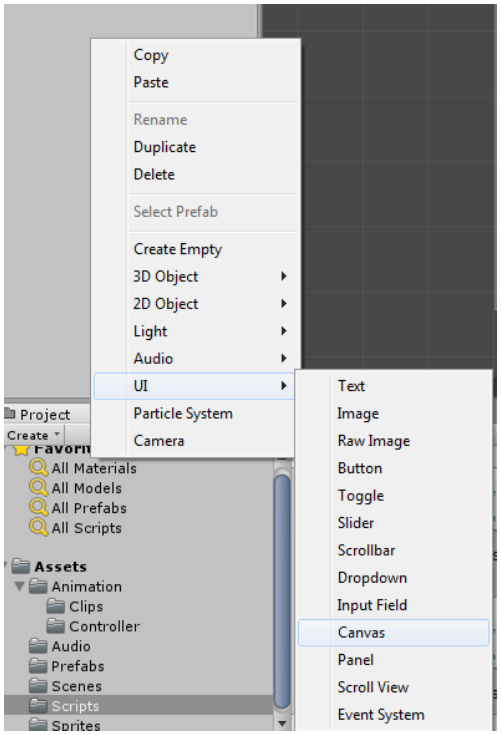
Sada kada smo kreirali poene i kako se sakupljaju treba negde da ih prikažemo krajnjem korisniku. Da bi se prikazalo bilo šta na ekranu kao što su UI elementi npr. neki tekst ,dugme ili poeni uglavnom se koriste objekti preko Canvasa. Najlakše je i vrlo je jednostavno pa ćemo ih ovde koristiti.

U unity editoru odabirom na desni klik u hijararhiji kreiramo Canvas kao na slici 33. Canvas posmatrate kao neki novi “prozor” nalepljen na Vašu scenu na koji možete da dodajete nove UI elemente, a da ostanete u istoj sceni. Svi UI elementi moraju biti “deca” Canvasa tj. da mu pripadaju. Može postojati I više UI Canvasa ali ako se doda npr. Dugme kome nije definisan Canvas element, Unity će sam kreirati i dodeliti Canvasu objekat dugme.

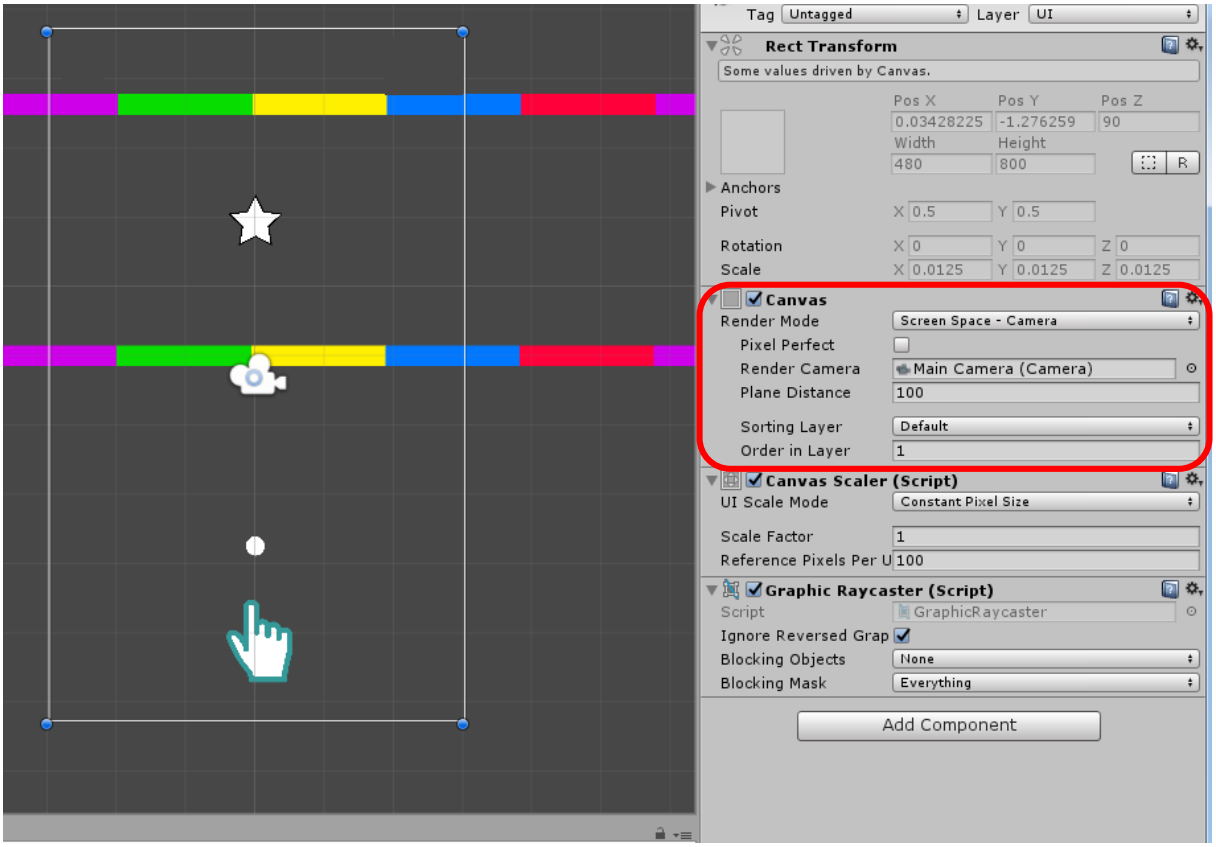
Kada se odabere Canvas u Inspector prozoru se pojavljuje polje **Canvas** na kome piše **Redner Mode** opcija. Po default-u editor sam podešava da bude **Screen Space – Overlay** koji će sam postaviti automatski razmere Canvasa i izračunati za svaki UI objekat sa Canvasa poziciju u **Rect Transform**.

U našem slučaju možemo to da promenimo tako što ćemo odabrati opciju **Screen Space – Camera** i u polje **Render Camera** prevući ćemo našu **Main Cameru** koju će on pozicionirati , a opciju **Order in Layer** ćemo postaviti da bude 1 kako bi Canvas bio ispred naše kamere kada se prikazuje.

NAPOMENA: Ukoliko ostavite default-nu opciju **Screen Space – Overlay** samo dok nameštate UI elemente na kanvas prebacite se u **Game mode** kako bi videli sve elemente bolje raspoređene, jer su na Scene modu jako uvećani.



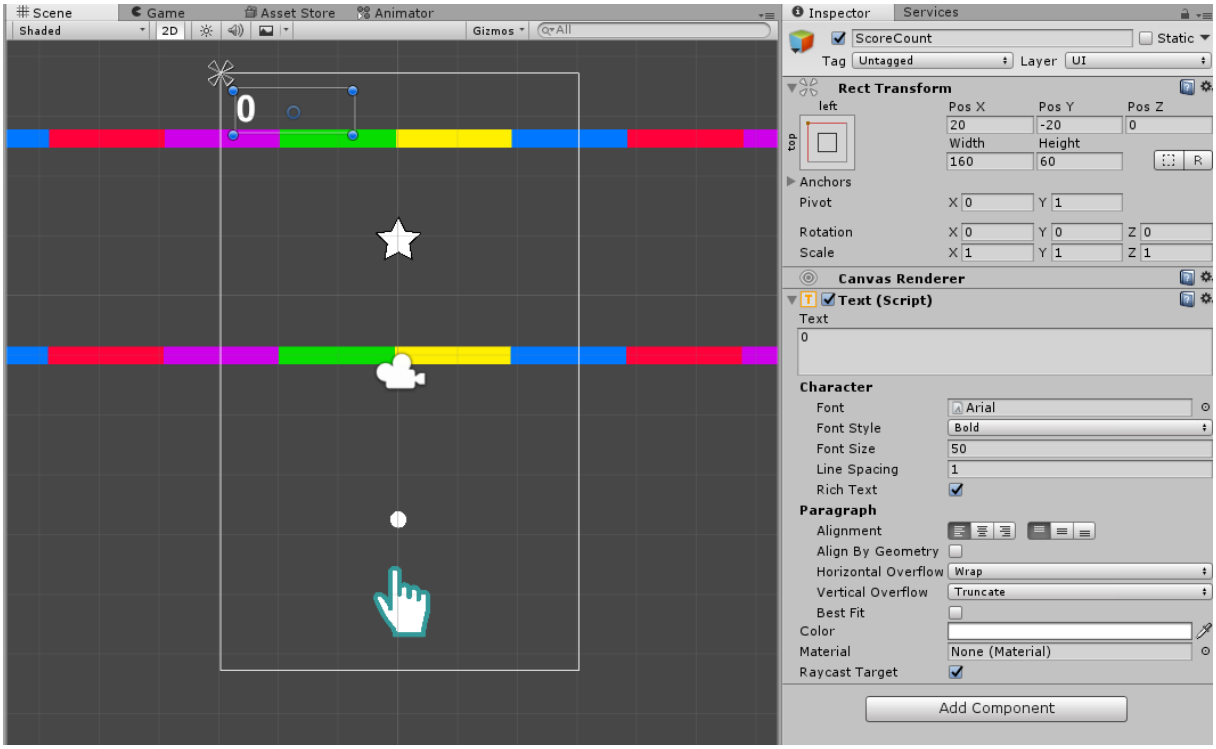
Slika 55 Kreiranje UI Canvas-a



Slika 56 Podešavanja Canvasa

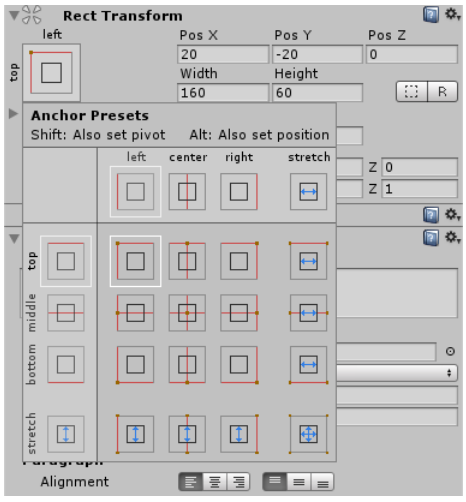
Kada se kreira Canvas objekat u hijararhiji će se pojaviti i **EventSystem** koji ide propratno uz UI elemente kao objekat koji pomaže pri "osluškivanju" događaja i razmenom informacija kao što je npr za dugme event **OnClick().**

Sada u Canvas dodajemo još jedan objekat a to je **Text objekat** koji želimo da nam prikaže rezultat poena tj. Broj sakupljenih zvezdica. Desni klik na **Canvas->UI->Text.** Preimenovaćemo ga u **ScoreCount ,** a zatim podesiti vrednosti u Inspector prozoru kao na slici 35. Tu možemo da se igramo sa tekstom. U polju Rect Transform podešavamo visinu , širinu, poziciju elementa zatim imamo i druge opcije koji će font da se koristi,koji text da prikazuje, njegova veličina, boja i slično.



Slika 57 Podešavanje UI elementa Text

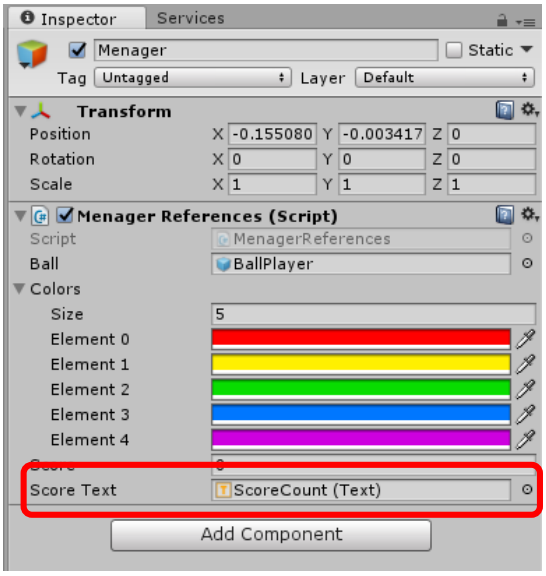
Polje **Anchor Presets** selektuje u kom kvadrantu Scene će se kreirati element i u zavisnosti gde je kreiran vrednosti u polju za poziciju **Rect Transform**-a će se prilagoditi tome. Mi ovde podešavamo da bude u gornjem levom uglu kao na slici 35. Dakle ako opdaberemo levi gornji ugao on će cvrednosti za njega setovati kao **poseban koordinatni sistem** sa poz 0. Na vama je da podesite polja **PosX i PosY** gde želite da stoji.



Slika 58 Achor Presets

Sledeći korak je nam ispiše Score, njega ćemo takođe prikazati preko Menadžera pa odabirom u hijararhiji na **Menager** objekat treba da u **MenagerReferences.cs** dodamo javnu promenljivu tipa **Text** kojoj ćemo dodeliti objekat **Text** iz Canvas-a koji smo prethodno kreirali kako bi preko Menadžera mogli da upisujemo vrednost u Text polje kao na slici 36. Dakle prevučete Text element u polje u Menager objektu nakon ažuriranja skripte u nastavku.

NAPOMENA: Kada se radi sa UI elementima mora da se uključi biblioteka **UnityEngine.UI;**



Slika 59 Dodavanje Score Text polja u Menager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //kada koristimo text za prikazivanje skora
//posto smo koristili objekat tipa UI

public enum ColorUsed
{
    Red,
    Yellow,
    Green,
    Blue,
    Purple
}

public class MenagerReferences : MonoBehaviour
{
    public GameObject ball;
    public Color[] Colors;
    public int score ;
    public Text scoreText; // dodali smo Text promenljivu

    void Start ()
    {
        score = 0;

    }

    public void IncreaseScore(int amount)
    {
        score += amount;
       scoreText.text = score.ToString(); //dodeljujemo vrednost Text polju za
ispis                    na ekranu

    }
}
```

5.3. Menjanje boje igrača odnosno loptice

Sledeći korak u kreiranju naše igrice jeste da omogućimo menjanje boja loptici pošto pojedini objekti mogu imati manje ili više boja koje variraju od objekta do objekta pa zbog toga u zavisnosti koje boje smemo koristiti moramo da omogućimo loptici opseg boja u koje može da se oboji, a zatim objekat koji će menjati loptici boju kreiramo na sličan način kao i zvezdu (da nestaje pri koliziji sa igračevom lopticom).

Uzećemo sprite iz foldera **Assets/Sprites/ colorthing** i prevući ga u hijararhiju, a zatim preimenovati u **Color Switch**. Zatim objektu u Inspector prozoru dodeljujemo komponentu **Box Collider 2D** i postavljamo vrednost **Is Trigger** na **true**. Moramo voditi računa da tzv. **Color Switch** objekat bude pozicioniran ispred objekata koji će koristiti druge boja kao i da opseg boja bude iz prethodno definisanog niza boja **Enum**-a koji smo definisali u **MenagerReferences.cs.**

Kreiramo skriptu u folderu Assets/Scripts/ nazvaćemo je **ColorSwitch.cs** i modifikovaćemo je na sledeći način:

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

public class ColorSwitch : MonoBehaviour
{
    public ColorUsed[] AllowedColors; //niz koji ce pamtiti boje iz enuma

    void OnTriggerEnter2D(Collider2D other) // kad dodje do kolizije sa
lopticom
    {
        if (other.gameObject.tag == "Player") //objekat igrac za kog zelimo da
promenimo boju
        {
            MenagerReferences refs =
GameObject.Find("Menager").GetComponent<MenagerReferences>(); // referenca na
menadžera u kome se nalazi niz enum sa spiskom boja

            int index = Random.Range(0, AllowedColors.Length - 1); //
nasumicno bira boju iz intervala boja
            ColorUsed newColor = AllowedColors[index]; //nova boja
            refs.SwitchPlayerColor(newColor); //metod iz menagera, promena
boje
            Destroy(gameObject); // da unisti objekat za promenu boja
        }
    }
}
```

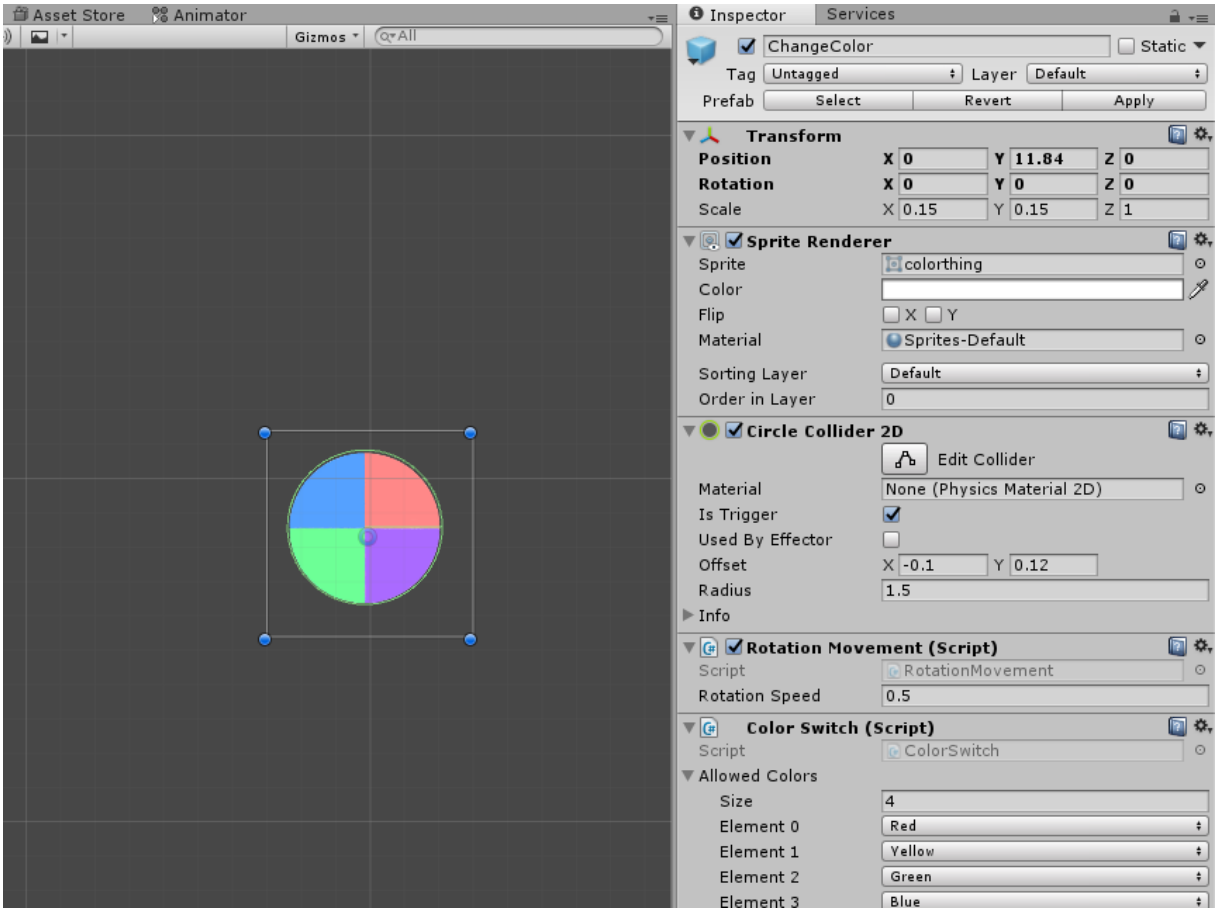
Pošto još uvek nismo kreirali metod **SwitchPlayerColor(newColor)** sada ćemo ga kreirati unutar Menadžera pa sledeći kod ažurira skriptu **MenagerReferences.cs**.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //kada koristimo text za prikazivanje skora
//posto smo koristili objekat tipa UI
```

```
public enum ColorUsed
{
    Red,
    Yellow,
    Green,
    Blue,
    Purple
}

public class MenagerReferences : MonoBehaviour
{
    public GameObject ball;
    public Color[] Colors;
    public int score ;
    public Text scoreText;

    void Start ()
    {
        score = 0;

    }

    public void IncreaseScore(int amount)
    {
        score += amount;
        scoreText.text = score.ToString();

    }


    public void SwitchPlayerColor(ColorUsed newColor)
    {
        SpriteRenderer  renderer = ball.GetComponent<SpriteRenderer>(); //referenca na
objekat loptice I uzima sprite Renderer kako bi pristupio bojama
        switch(newColor) // prosledjuje index boje koju dajemo nasumicno fji
        { // ovde uzima vrednosti iz niza boja koje smo dodali u menagera -tim redom
            case ColorUsed.Red:
                renderer.color = Colors[0];
                break;
           case ColorUsed.Yellow:
                renderer.color = Colors[1];
                break;
            case ColorUsed.Green:
                renderer.color = Colors[2];
                break;
            case ColorUsed.Blue:
                renderer.color = Colors[3];
                break;
            case ColorUsed.Purple:
                renderer.color = Colors[4];
                break;
            default: //ovo je stavljeno radi reda inace nikada nece doci u default ali ya
svaki slucaj smo stavili npr red može bilo koja boja
                renderer.color = Colors[0];
                break;
        }
    }
}
```

Sada kada smo kreirali skriptu i uspešno je sačuvali, dodajemo je držanjem i prevlačenjem levog klika miša na objekat **Color Switch** ili preko Add Component to je na izboru programera kako je kome lakše.

Primećujemo da sada postoji dodato polje za unos broja elemenata niza kao I elementi niza koje želimo. Dakle ako pozicionirate **Color Switch** ispred trougla koji ima crvenu, zelenu i plavu boju onda Color Switch objektu u ovom polju dodeljujete broj elemenata 3 i birate ove tri boje koje sme da izabere za igračevu lopticu. Tako će u koliziji sa istom doći do promene u jednoj od te tri boje sve do narednog objekta za promenu boja se neće menjati. Ukoliko želite možete dodati I skriptu **RotationMovement.cs** I postaviti željenu brzinu okretanja za ovaj objekat ako želimo da se i on okreće. Ostalo je samo da pozicionirate objekat po Vašem ukusu.



Slika 60 Inspector prozor za Color Switch objekat

6. Game Over & Finish

Svaka igra ima svoj cilj I svoj kraj (smrt) pa je došlo vreme da nakon kreiranja svih delova kreiramo I deo koji čini samu poentu I smisao igrice.

CILJ - Loptica treba da prolazeći kroz različite prepreke sakuplja zvezde (poene) probijajući se do kraja nivoa.

KRAJ IGRE – Ukoliko loptica dođe u dodir sa objektom (delom objekta) koji nije iste boje kao i sama igračeva loptica treba da "pogine" tj. Da pri koliziji prekine dalji tog igre.

6.1. Kraj Igre – Game Over

Kao što smo spomenuli završetak igre je onda kada loptica dođe u koliziju sa nekim objektom koji je drugačije boje od same loptice. Tada se pojavljuje Nova scena za Restart koju ćemo kasnije kreirati a sada da podesimo u kodu mesto šta treba da radi kada dođe do sudara.

Ažuriraćemo skriptu **TriggerCollision.cs** I dodaćemo samo poziv funkcije **ShowRestartPanel()** iz Menadžera pošto smo na početku već kreirali šta će se desiti ukoliko dodirne loptica neki deo drugog objekta koji ima boju drugačiju od nje.

Sada red u kome smo definisali da prikaže ispis u konzoli kako bi znali da je došlo do kolizije zakomentarišemo/obrišemo I dodamo sledeći deo koda.

```
 void OnTriggerEnter2D(Collider2D other)
 {
       if (other.gameObject.tag == "Player")
       {
            Color playerColor =
other.gameObject.GetComponent<SpriteRenderer>().color; //boja igračeve loptice
            Color thisColor =
this.gameObject.GetComponent<SpriteRenderer>().color; //boja drugog objekta sa
kojim se dešava kolizija pošto je skripta dodata na pojedinačne objekte

            if (playerColor != thisColor )
            {
                MenagerReferences refs =
GameObject.Find("Menager").GetComponent<MenagerReferences>();
                if (refs != null)
                {
                    refs.ShowRestartPanel();
                }
            }
       }
    }
}
```

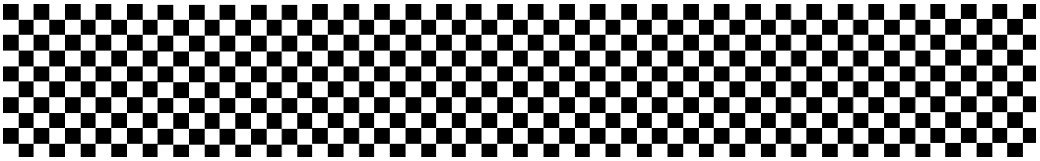
U skriptu za Menadžera dodajemo sada metod za pozivanje nove scene koju ćemo kreirati u narednom poglavlju I šaljemo joj informaciju o osvojenim poenima preko **PlayerPrefs** metoda koji može da čuva neka bitna svojstva igrača između različitih scena u našem slučaju poene treba da nam prikaže na sceni za **Restart**. Dodaćemo poziv sledeće scene pri sudaru sa drugom bojom objekta ali ćemo scenu klreirati u narednom poglavlju (9). **PlayerPrefs** metod može da postavlja bilo koju vrednost sa **SET** ali mora I da uzima sa **GET**. Dakle U ovoj skripti mi setujemo promenljivu koju ćemo u drugoj sceni (Restart) imenovati kao **FinalScore** I dodeljujemo joj vrednost iz trenutne scene koju pamti u **score** promenljivoj.

Sledeći metod dodati u **MenagerReferences.cs** skriptu u Assets/Scripts/ folderu:

```
    public void ShowRestartPanel()
    {
       PlayerPrefs.SetInt("FinalScore", score); //za čuvanje rezultata tj poena
       Application.LoadLevel("Restart"); // da nas odvede na novu scenu

    }
```

6.2. Dolazak do cilja – Finish line

Rekli smo da kada igrač prođe (uspešno) kroz sve prepreke treba da dodirne zastavu kao sa slike 38 što bi označilo dolazak na cilj I odvelo ga na drugi nivo.



Slika 61 Finish Line - Cilj

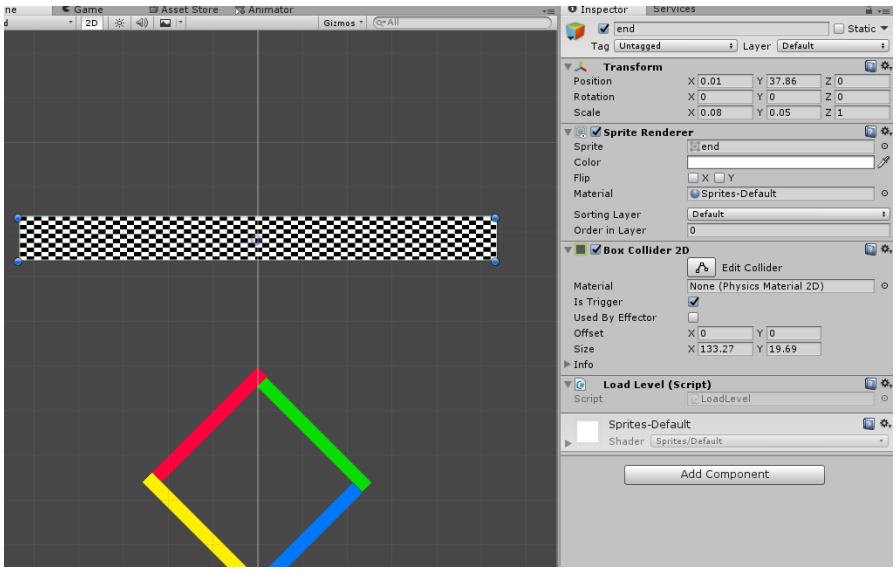
U folderu **Assets/Sprites/** se nalazi sprite kao sa slike pod nazivom **end.** Prevući ga u hijararhiju a zatim dodati u Inspector prozoru **Box Collider 2D** I postaviti **Is Trigger** na **true**. Pozicionirati tako da loptica mora doći u koliziju sa zastavicom.

Kada smo završili kreiranje ove scene sačuvaćemo je pod imenom "**Level**". Kreirati skriptu u folderu **Assets/Scripts/LoadLevel.cs** koja će pamtiti metode za prebacivanje između scena/nivoa ali I događaje dugmića I podešavanja igrice.

NAPOMENA: Za sada još nismo kreirali scene za **Level2 I Menu** ali ćemo ih navesti ovde jer po njihovom kreiranju u narednom poglavlju(9) će morati da se doda ovaj deo koda, pa da ne bi bilo nepregledno razmetanje kodovima najbolje je da bude skripta definisana u poglavlju u kome pripada kako bi bilo preglednije za prebacivanje na drugi nivo/scenu.

Otvoriti skriptu I dodati metod **OnTriggerEnter2D()** koji će u sudaru sa lopticom pozvati narednu scenu:

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

public class LoadLevel : MonoBehaviour
{
    public void OnTriggerEnter2D(Collider2D other) // ova f-ja se koristi u
slucaju dodira loptice sa ciljem posto skripta nema OnClick() event kao sto
ima dugme
    {
       if (Application.loadedLevelName == "Level2")
        {
            // pita da li je aktivna scena poslednja ako jeste onda na kraju
levela      loaduj meni pošto imamo u planu dva nivoa Level2 će biti poslednja
            Application.LoadLevel("Menu");
        }
        else Application.LoadLevel("Level2"); // ako nije poslednja znaci da
je prošao prvi nivo pa učitaj drugi
    }

}
```



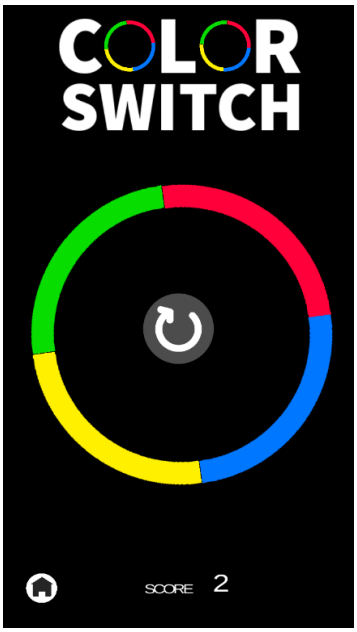
Slika 62 Inspector prozor za flag/kraj igre

7. Kreiranje drugih scena i Panela

Osim osnovne scene Level treba da kreiramo i scene za Menu – početnu scenu, zatim za Game Over – kraj igre , eventualno Level2 i Panel za Pauzu, pa ćemo ovo poglavlje posvetiti tome.

7.1. Replay/Restart - Scena za restartovanje nivoa

Novu scenu kreiramo u folderu **Assets/Scenes/** i dodelimo joj ime koje želimo. Prethodno treba da sačuvamo prethodnu Scenu koju smo kreirali i to treba raditi često za slučaj da dođe do nekih nepredviđenih situacija I gubitka učinka. Novu scenu preimenujte u "**Restart**" kako bi najbolje znali na šta se odnosi.



Slika 63 Restart scene

Ovu scenu želimo da pokrenemo kada dođe do kolizije igračeve loptice sa drugim objektima ukoliko im se boje ne slažu.S obzirom da je većina objekata koji se koriste u igrici već napravljena i da se nalazi u folderu **Assets/Prefabs/** možemo izabrati objekte iz tog foldera po želji I prevući ga na našu scenu. Primer je dat na slici 40.

Za natpis (Logo) naziva igrice smo uzeli sprite iz foldera **Assets/Sprites/ColorSwitch** , umesto dva O postavite dva objekta kruga iz Prefab foldera I treći na sredinu Scene. Promenite im poziciju i polje Scale u Inspector prozoru u polju Transform na vrednosti približne slici 40. U prethodnom poglavlju smo spomenuli da je za ovu scenu potrebno da kreiramo promenljivu **FinalScore** preko **PlayerPrefs** pa ćemo sada da kreiramo skriptu **FinalScore.cs** u folderu **Assets/Scripts/** i sa Get ćemo pokupiti njenu vrednost kako bi preko Canvasa prikazali osvojeni rezultat igrača pre nego što je izgubio.

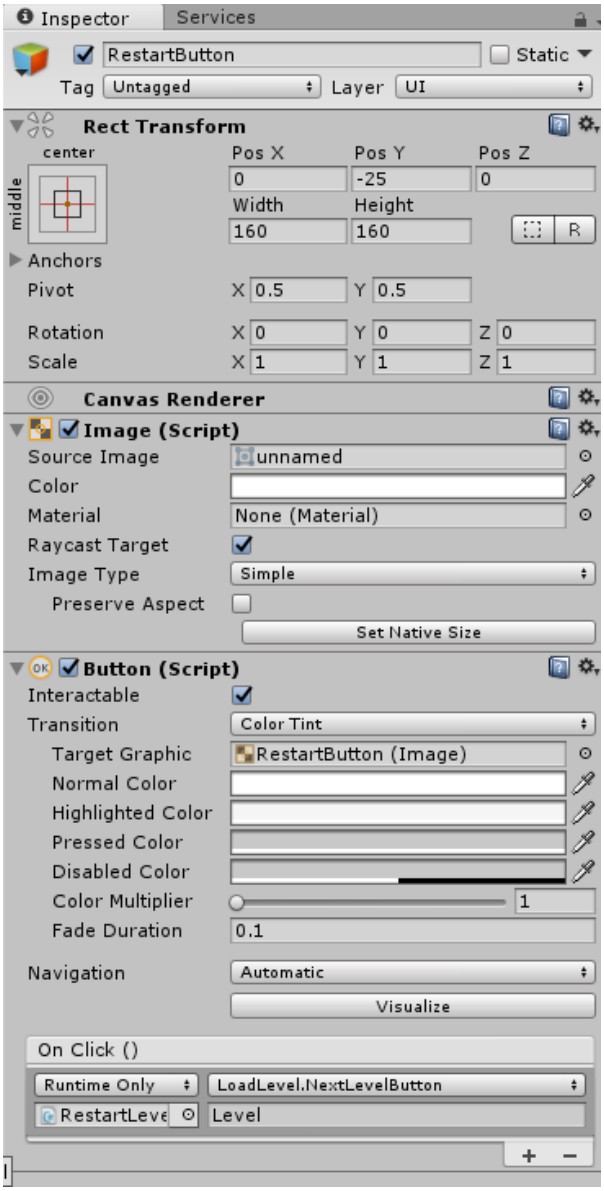
Skriptu ***FinalScore.cs*** popuniti na sledeći način:

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

public class FinalScore : MonoBehaviour {

    int Final = PlayerPrefs.GetInt("FinalScore"); //dodeljujemo
vrednost promenljive FinalScore preko Gettera iz registra PlayerPrefs
    public Text score; // opet dodeljujemo objekat tipa Text kome ćemo
dodeliti Text iz Canvas-a
     void Start ()
     {
        score.text = Final.ToString(); //postavljanje polja Text na
vrednost poena
     }
}
```

U poglavlju **7.2.** je objašnjen postupak kreiranja Text-a unutar Canvas elementa i kako se rukovodi sa UI elementima kao i njihovo pozicioniranje i sređivanje pa se ovde nećemo time ponovo baviti. Kreirajte canvas I jedan **Text** I ponovite korake iz napomenutog poglavlja.

Kreiramo jedan prazan objekat u hijararhiji i imenujemo ga kao **FinalScore**, zatim mu dodeljujemo skriptu **FinalScore.cs** i u polje za Text prevlačimo prethodno definisanu Text labelu iz **UI Canvas**-a kako bi je povezali sa sumom poenta iz skripte.

Kao što se vidi sa slike 40 u centru kruga se nalazi dugme za restart igrice nakon gubitka. Ono se takođe kreira unutar Canvas-a ali se bira **Button** sada. Po default Unity mu dodaje Text ali to može u Inspector prozoru da se podesi po vašem ukusu. Mi smo uklonili Text polje, a u polju **Source Image** dodelili sprajt foldera ***Assets/Sprites/replay*** i nazvali ga **RestartButton.**



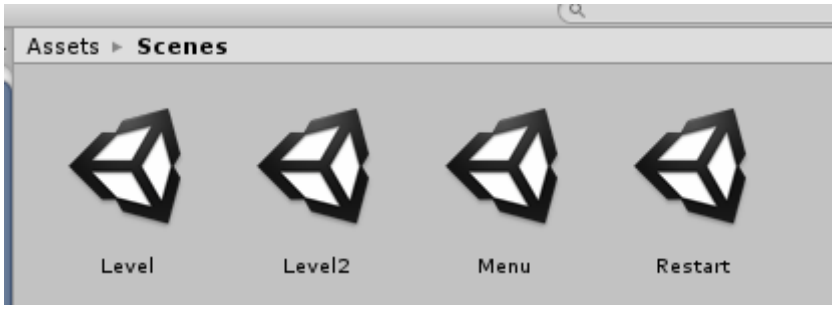
Slika 64 Dugme za restart

Kao što se može primetiti dugme ima dodatnu opciju koji poziva metod **OnClick().** U hijararhiji kreiramo još jedan prazan objekat I dodeljujemo mu ime **RestartLevel**. Zatim kreiramo novu skriptu **LoadLevel.cs u Assets/Scripts/.**

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

public class LoadLevel : MonoBehaviour
{
    //funkcija za pookretanje igre pritiskom na play ili za restart, za bilo koju scenu ,prima  string "naziv scene"

    public void NextLevelButton(string levelName)
    {
        Application.LoadLevel(levelName); //na osnovu datog stringa ucitava narednu scenu
    }

    public void OnTriggerEnter2D(Collider2D other) // ova f-ja se koristi u slucaju dodira loptice sa ciljem posto skripta nema OnClick() event kao sto ima dugme
    {
       if (Application.loadedLevelName == "Level2")
        {
            // pita da li je aktivna scena poslednja ako jeste onda na kraju levela      loaduj meni pošto imamo u planu dva nivoa Level2 će biti poslednja
            Application.LoadLevel("Menu");
        }
        else Application.LoadLevel("Level2"); // ako nije poslednja znaci da je prošao prvi nivo pa učitaj drugi
    }

}
```

Ovu skriptu dodeljujemo novokreiranom objektu **RestartLevel**, a zatim njega levim klikom miša prevlačimo u metod **OnClick()** kao na slici 41 u prozoru dugmeta za Restart. Sa desne strane će se zatim pojaviti opcije za GameObject, Transform, i LoadLevel koje je objekat nasledio iz skripte koju smo mu dodelili. Pritiskom na LoadLevel Automatski će očitati sve metode koje poseduje objekat preko skripte LoadLevel.cs. Izabrati metod **NextLevelButton()** jer nam on treba da loadira scenu. Metoda prima naziv scene **tipa String** i u polju ispod dajemo puno ime (osetljivo na velika ili mala slova) scene koju želimo da nam se očita pritiskom na to dugme. Vi ćete dati naziv scene koja vodi na prvi nivo koji smo pravili u prethodnim poglavljima (Mi smo nazvali **Level**).

Na isti način kreirati prazan objekat **Home** , dodeliti mu skriptu **LoadLevel.cs** a zatim u Canvas-u kreirati još jedan UI element i nazvati ga slobodno istim imenom "**Home**" I na isti način kao malopre kreirati dugme i u metodi **OnClick()** prevući objekat home iz hijararhije I izabrati metod NextLevelButton() samo sada proslediti naziv scene koja void na početak tj **Menu** (koji ćemo sada kreirati) za slučaj da ne želi da restartuje nivo mora imati I ovu opciju.



Slika 65 Primer naših imena scena

7.2. Menu – Početni ekran igrice

Kao što možemo da primetimo ova scena je jako slična prethodnoj, opet koristimo predefinisane objekte kao što je logo I centrirani krugovi iz foldera **Assets/Prefab/,** jedina razlika u ovoj sceni jesu dugmići za Play kao i tri dugmeta za izlazak iz aplikacije, za zvuk i mutiranje zvuka.Kako se kreiraju ovi dugmići već smo objasnili u prethodnom poglavlju 9.1. Ukratko:

- Kreirati **Button** u Canvas objektu I imenovati ga kao **PlayButton,**
- Podesiti mu vrednosti u Inspector prozoru
- Dodeliti mu sprite iz foldera ***Assets/Sprites/playbutton***
- Kreirati prazan objekat u hijararhiji I imenovati ga kao **Play** te mu dodeliti skriptu **LoadlLevel.cs**
- Unutar dugmeta u canvas-u u polju za metodu **OnClick()** prevući prazan objekat iz hijararhije **Play** i on će učitati preko skripte **LoadLevel.cs** i metodu **NextLevelButton()** na isti način kao u poglavlju 9.1. kojoj će se proslediti ime sledeće scene u našem slučaju **Level.**

Slika 66 Početna scena - Menu

Ono što je ovde novitet jesu tri dugmeta u donjem delu ekrana kojima ćemo više posvetiti pažnje. Dakle najlakše je da imamo jednu skriptu koja će se baviti podešavanjima u manipulacijom između scena pa ćemo mi za to svojstvo odrediti skriptu **LoadLevel.cs** pošto ona se već bavi prelaskom sa jedne scene na drugu što se dešava pozivom događaja klikom na neko dugme pa je usko povezana I sa ostalim funkcijama dugmića.

Dugmići se kreiraju na standardan način samo umesto Text-a dodeliti im sprajtove iz foldera :

- ***Assets/Sprites/sound*** – za dugme uključi zvuk
- ***Assets/Sprites/mute*** – za dugme isključi zvuk
- ***Assets/Sprites/quit***– za dugme ugasi igricu EXIT

Podešavanja za svakog od njih uradite na isti način po vašem nahođenju gde želite ali primer je dat na slici 43 kako smo mi napravili. Jedino što ćemo ovde promeniti jeste ono što će dugmići raditi. Dakle otvorite skriptu LoadLevel.cs I ažurirajte kod kao u priloženom:

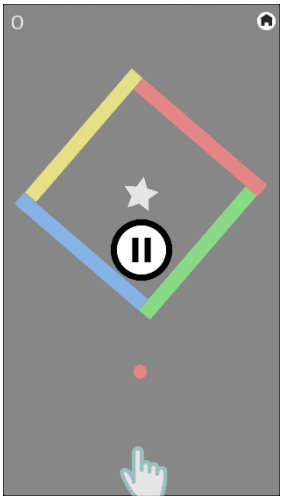
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadLevel : MonoBehaviour
{
    //funkcija za pookretanje igre pritiskom na play ili za restart, za bilo koju
scenu prima  string "naziv scene"

    public void NextLevelButton(string levelName)
    {
        Application.LoadLevel(levelName);
    }

    public void OnTriggerEnter2D(Collider2D other) // ova f-ja se koristi u slucaju
dodira loptice sa ciljem posto skripta nema OnClick() event kao sto ima dugme
    {
       if (Application.loadedLevelName == "Level2")
        {
            // pita da li je aktivna scena poslednja ako jeste onda na kraju levela
loaduj meni pošto imamo u planu dva nivoa Level2 će biti poslednja
            Application.LoadLevel("Menu");
        }
        else Application.LoadLevel("Level2"); // ako nije poslednja znaci da je prošao
prvi nivo pa učitaj drugi
    }

    public void Exit()
    {
        Application.Quit(); //za izlazak iz aplikacije
    }

    public void MuteSound()
    {
        //pauziraj zvuk I postavi jačinu zvuka na 0
        AudioListener.pause = true;
        AudioListener.volume = 0;
    }

    public void PlaySound()
    {
        //suprotno od mutiranja, I zvuk je sad 1 pošto može biti 0/1
        AudioListener.pause = false;
        AudioListener.volume = 1;
    }

}
```

7.3. Pause – Pauziranje igrice

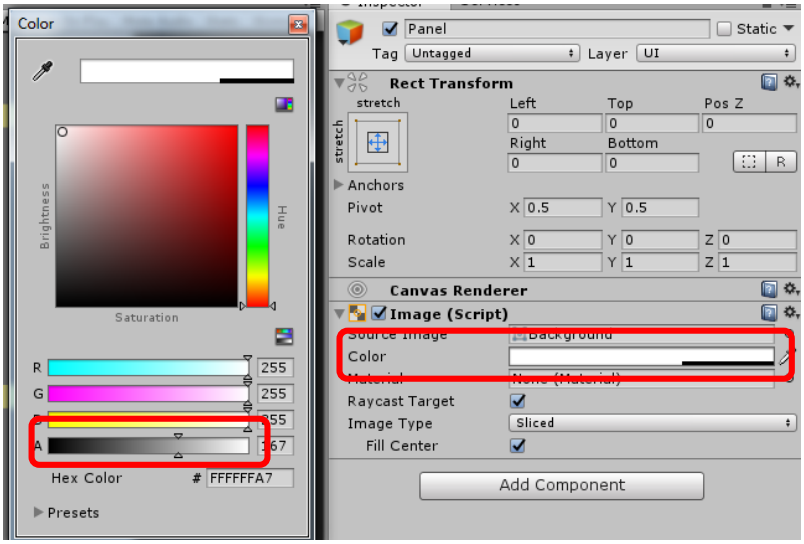
Svaka igrica mora imati I pauzu za korisnike igrice u nepredviđenim situacijama kada je potrebno igricu pauzirati. Ono što je bitno za ovaj deo jeste to da **NEĆEMO** učitavati novu scenu već ćemo kreirati **UI Panel** preko postojaće scene kao **Overlay** (drugi sloj) sa **Opacity**-jem (prozirnost) kako bi se u pozadini videla igrica I dokle smo stigli sa kretanjem loptice.

Takođe je bitno da pozovemo metod koji će zalediti igračevu lopticu u samoj igrici dok je aktivan **UI element Panel.**

Dakle kreirajte UI element unutar Canvasa desnim klikom mišem I odabirom **Panel** objekta. Zatim desni klik na Kreirani panel I dodelite dva dugmeta jedan za "**Home**" jedan za Pauzu "**Unpause**" kao sa slike 45 I 47.

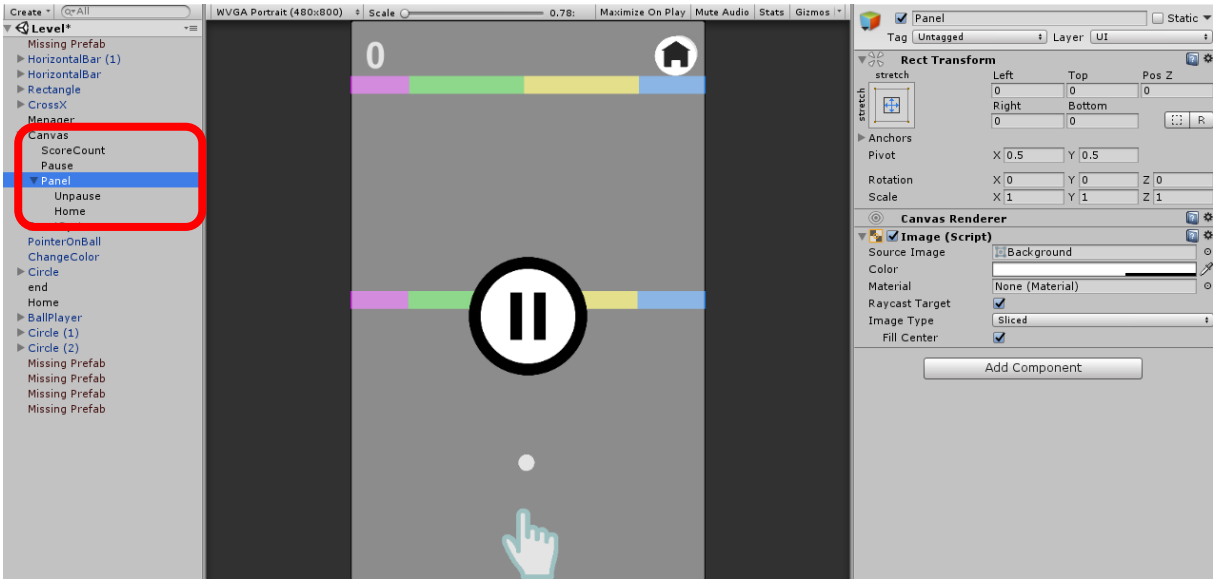


Slika 67 Pause Menu - Pauziranje igrice



Slika 68 Opacity za Panel

Panel treba da ima opacity koji se podešava klikom u inspector prozoru na polje **Color** I odabir kolone **A** kao na slici 46. A prozirnost se može proveriti na Sceni koliko jako treba da bude setovana povlačenjem **Brightness** jačine levo ili desno na koloni A. Panel **Panel** treba da bude roditelj dugmićima **Home** i **Unpause.**



Slika 69 Kreiranje Panela i izgled na sceni

Sad treba ažurirati skriptu I dodeliti metodu za reakciju na događaje dugmića. Prvo ćemo kreirati metode u skripti **Assets/Scripts/LoadLevel.cs:**

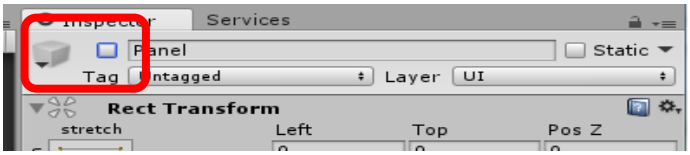
```
    public void Pause()
    {
        if (Time.timeScale == 1) //ako vreme teče tj igrica je u pokretu
        {
            Time.timeScale = 0; // klikom na pause onemogući vreme tj
stiopiraj igricu
        }
        else
        {
            Time.timeScale = 1; // u suprotnom ako je stopirana I opet
kliknemo na pause pokreni timer
        }
    }
```

Dodajemo na našoj glavnoj sceni dugme za pauzu **Pause** kojom će se pokrenuti kreirani **Panel** . To uradimo kao I u prethodnim poglavljima samo izaberemo sprajt iz foldera

- ***Assets/Sprites/Pause***

Podesiti da kreirani panel bude **nevidljiv** pri pokretanju **Level** scene ali da pritiskom na dugme **Pause** se prikaže tj postane Aktivan. To se radi na sledeći način:

1. Onemogućiti vidljivost Panela odabirom Panela u Inspector prozoru a z.atim odčekirati opciju kao na slici

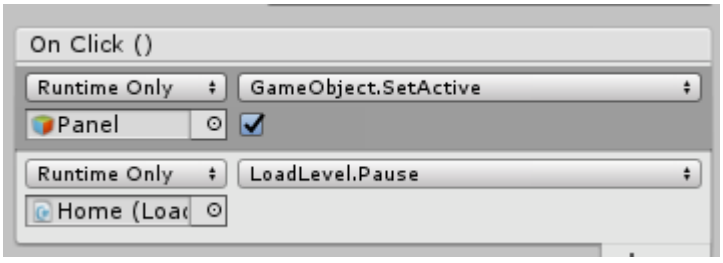


Slika 70 Postavljanje Panel na nevidljivo

2. Klikom na dugme **Pause** u Inspector prozoru u metodi **OnClick()** prevući ceo **Panel** (ili kako ste ga nazvali) I odabrati iz padajućeg menija **GameObject** kako bi uzeo kreirani objekat I odaberite metod **setActive(bool)** I čekirate da bude **True**!

3. Sada mora da klikom na **ISTO** dugme se pozove I metod za pauziranje pa će se taj metod pozvati iz objekta koji smo kreirai u hijararhiji I dodelili mu skriptu **LoadLevel.cs** (nama se zove **Home**) I postaviti vrednost na poziv metoda **Pause()** kao na slici 49

NAPOMENA: OnClick() može pozivati više metoda istovremeno ili različitih objekata koji mogu da imaju događaje ali koji će se izvršiti istovremeno u trenutku pritiska dugmeta.

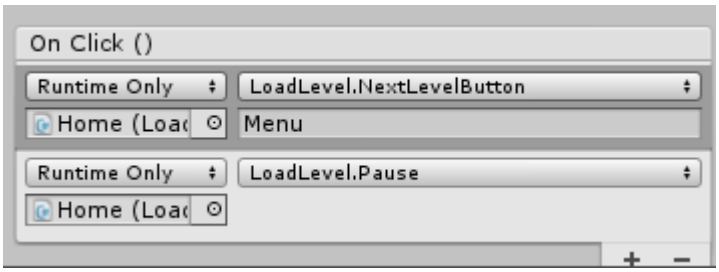


Slika 71 Podešavanja OnClick() za dugme Pause

Sada još preostaje da dodelimo metode za **OnClick()** događaj na dugmiće. Za **Home** dugme:

1. Iskoristićemo kreiran objekat **Home** iz hijararhije pošto on već ima nakačenu skriptu **LoadLevel.cs**
2. Prevučemo objekat u **Home** dugme u metod **OnClick()**
3. Odaberemo opciju "+" da dodamo još jedan **event** I dodaćemo još jednom isti objekat **Home.**
4. Podesiti da prvi učita novu scenu **Menu**
5. Drugi će morati da bude preventive za slučaj da ukoliko odaberemo dugme za **Menu** scenu on I dalje nije **POKRENUO** igricu već I dalje je drži pauziranu pa zbog toga moramo da kažemo –> odvedi me na početni ekran ali sada pri pokretanju želim da krenem ispočetka

Na slici 50 se vidi prethodno uputstvo realizovano ukoliko Vam je lakše da pratite sliku:

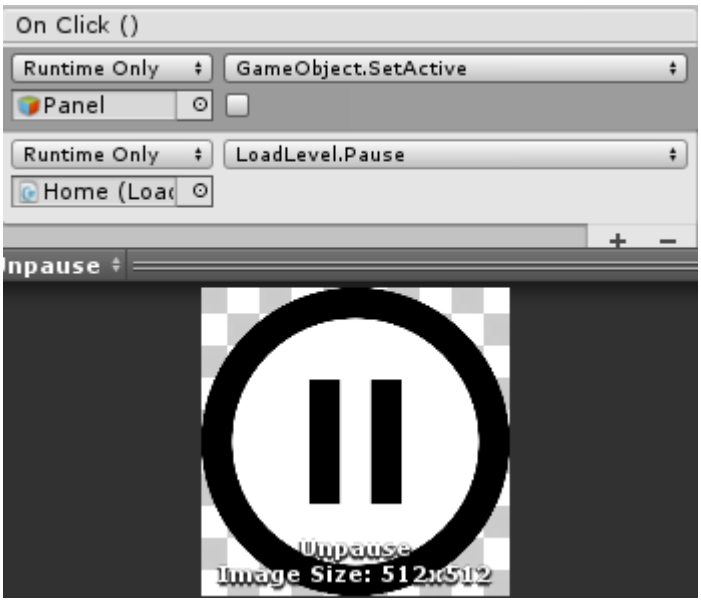


Slika 72 OnClick() podešavanje za dugme Home

Za dugme **Unpause :**

1. Podesimo poziciju za dugme kao I veličinu I sprajt dodelimo da bude isti kao za dugme Pause sa glavne Scene
2. Treba da aktiviramo sada opet igricu pa ćemo **Panel** da postavimo opet na Invisible setovanjem **setActive()** metode na **false**.
3. Prevućićemo opet objekat iz hijararhije **Home** kao drugi događaj u metod **OnClick()** i odabraćemo metod iz skripte LoadLevel.cs **Pause()** kako bi odpauzirali zaleđeno vreme i nastavili igricu dalje.

Na slici 51 se vidi prethodno uputstvo realizovano ukoliko Vam je lakše da pratite sliku:



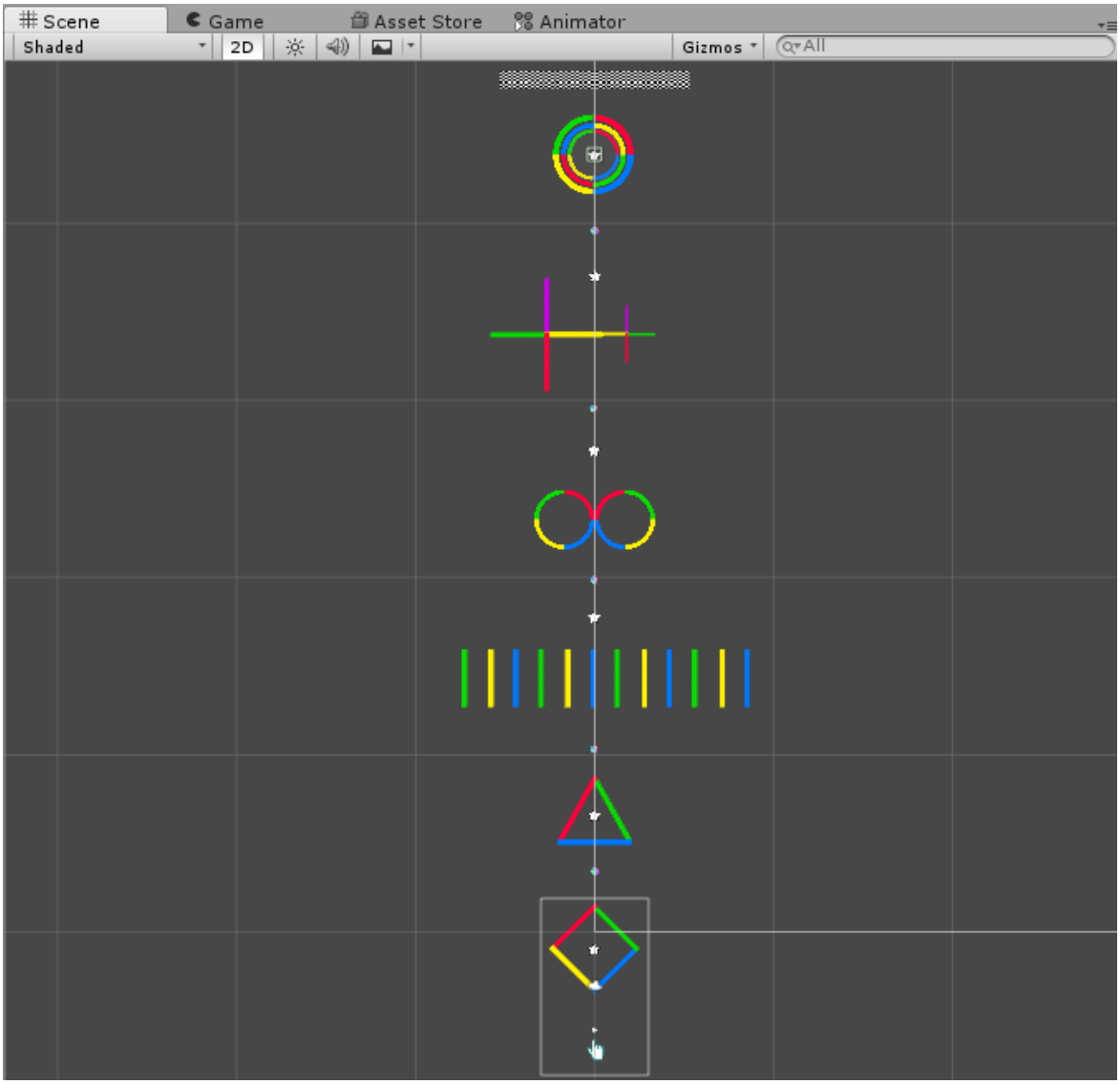
Slika 73 Podešavanje metode OnClick() za dugme Unpause

7.4. Level2 – Kreiranje scene za novi nivo

Kada loptica igrača dođe do **flag**-a koji je trigger , odnosno kraja u **Level** sceni ona pri koliziji sa njom pokreće metod **OnTriggerEnter2D()** kao što smo pomenuli u poglavlju 8.2. gde smo i pozvali novu Scenu. Dakle kreirajte novu scenu u folderu **Assets/Scenes/Level2** i sada je na Vama kako želite da dizajnirate naredne nivoe. S obzirom da folder **Prefabs** igra jako važnu ulogu u kreiranju igrica inače ali i kod nas , jer sada možemo da spajamo više prefabova I kreiramo nove objekte tj prefabove ili da koristimo postojeće to je na Vama da se igrate i pustite mašti na volju. Sada je sve Copy/Paste ali ćemo mi Vama ovde na slici 52 prikazati kako smo mi kreirali drugi nivo.

Evo nekih saveta za kreiranje narednih nivoa/scena:

1. Možete da stavljate teže prepreke
2. Da poene prenosite iz prethodnog nivoa
3. Da povećate brzinu rotacije objekta u skripti za rotaciju (**RotationMovement**) koja je prikačena za objekte
4. Da dodajete više boja
5. Postavljanje većeg broja prepreka

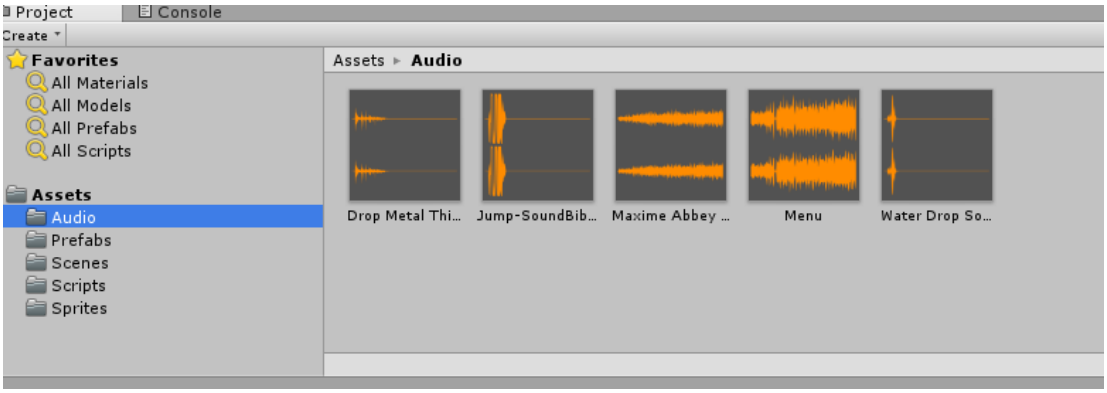


Slika 74 Level2 Izgled scene

8. Audio Source – Dodavanje zvuka

Što se tiče dodavanja **zvuka** vašoj igrici (Što daje poseban utisak i bolju dinamiku igrici) to se vrlo lako realizuje. Sve što je potrebno jeste da svaki objekat ima **Audio Source** ili **Audio Listener** koji Unity sam dodeljuje tj. postoji kod nekih objekata kao već definisana komponenta. Mi ćemo kreirati folder **Assets/Audio/** i dodati neke zvukove za našu igricu.

NAPOMENA: Važno je voditi računa o veličiti fajlova audio izvora pošto što su veći fajlovi ili duži igrica će zauzimati znatno više prostora , samim tim će doći i do bagova I kočenja igrice. Možda će i vremenski više učitavati određene scene/nivoe. Unity podržava različite Audio fajlove ali gledajte da budu **MP3, OGG** kako ne ne biste morali da konvertujete.



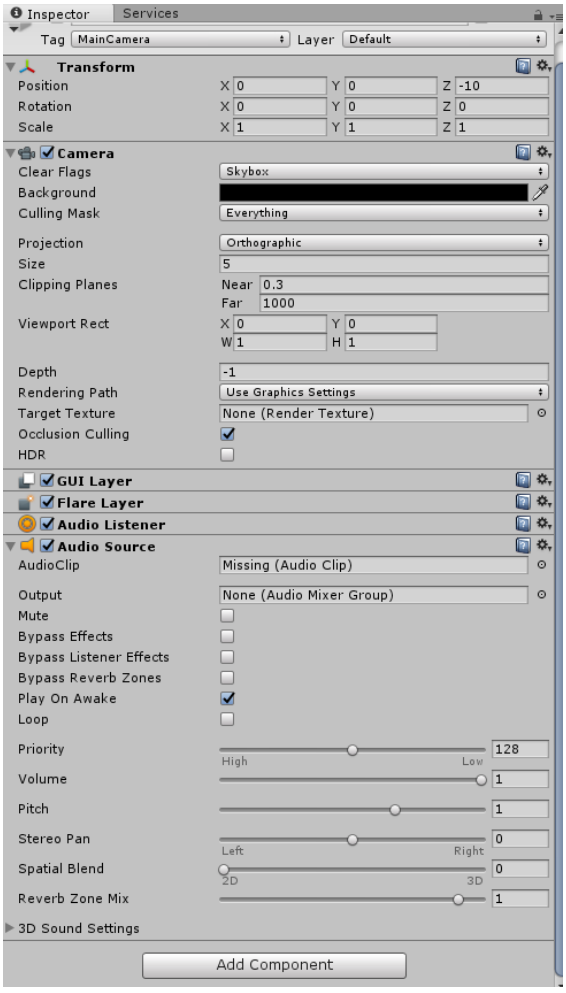
Slika 75 Folder Audio sa ubačenim zvukovima

Audio koji želite da postavite za vodeći zvuk igrice odnosno nivoa se postavlja na **Main Cameru** pošto svaka scena mora imati ***jednu*** kameru onda može na nju imati I dodat zvuk koji se po pokretanju scene/nivoa automatki pokreće.

Main Camera već ima komponentu ***Audio Listener*** čime je omogućeno da može da pokreće I osluškuje zvuk. A u **inspector** prozoru ćete u komponenti za editovanje I dodavanje zvuka ***Audio source*** dodati u polju **Audio Clip** Audio z foldera **Assets/Audio/**.

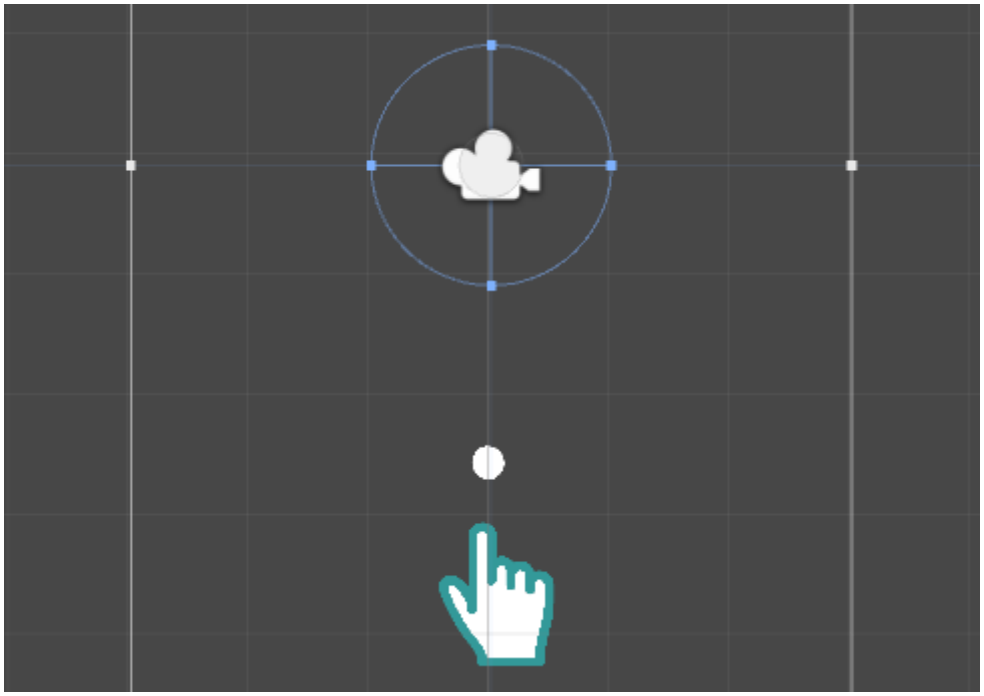
Obratite pažnju na to da dodajete na Main Cameru unutar naše loptice ***BallPlayer***. Tako da ako kreirate drugi nivo glavnu kameru stavite na nevidljivo pošto Vam 2 kamere nisu potrebne, jer jedna pripada loptici.

Što se tiče ostalih podešavanja ostavite automatska/defaultn-a pošto ostala polja su za neke složenije I delikatnije igrice a kod nas to nije potrebno.



Slika 76 Main Camera Audio

Sada kada smo dodali zvuk ono bi trebalo da izgleda ovako:



Slika 77 Izgled nakon dodavanja Audio Clip-a

Svaka scena/nivo može imati različite melodije. Mi smo dodali dva audio elementa za dva nivoa I početnu Menu scenu ali vi se možete igrati I biti kreativni u kombinovanju različitih zvukova. U folderu Assets/Audio/ imao sledeća 2 Audia za scene koje možete da dodate:

- **Maxime Abbey - Arabian Feelings.mp3**, pesma je u vlasništvu http://www.arachnosoft.com
- **Menu.mp3**

8.1. Dodavanje Audija objektima

Sada kada smo dodali muziku našoj igrici ona će se pokrenuti kada pokrenemo I igricu na "Play". Sada želimo da dodamo zvukove za objekte Zvezdu, lopticu I color Switch. Bitno je da se audio doda u skripte kako bi pri interakciji/koliziji došlo do reprodukcije zvuka.

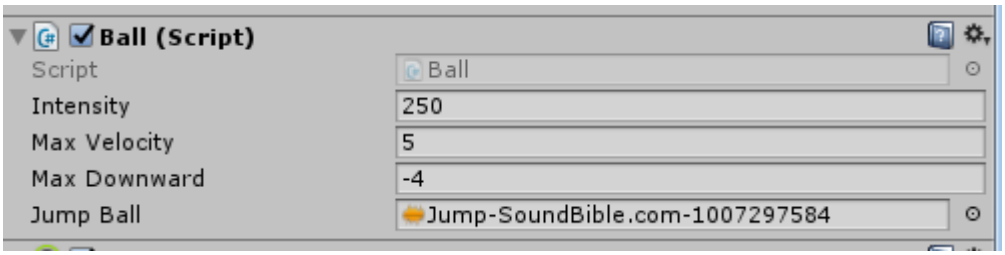
LOPTICA

Za našu glavnu lopticu želimo da se zvuk skoka čuje svaki put pri pritiski toucha/mouse dugmeta kao dinamičniji efekat I bolji osećaj odskakivanja loptice (Bouncing).

Pošto audio treba da dodamo pri svakom skoku onda se u metod za skaknje dodaje I audio. Ažuriraćemo skriptu ***Ball.cs*** I dodati sledeće linije koda u C# skriptu.

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


public class Ball : MonoBehaviour
{
    public AudioClip jumpBall;
```

```
        .
        .
        .
      void FixedUpdate ()
      {
         if (Input.GetMouseButtonDown(0))
         {
             if (rb2d != null)
             {
                 rb2d.AddForce(Vector2.up * intensity );
                  //dajemo metodu zvuk I poziciju loptice
                 AudioSource.PlayClipAtPoint(jumpBall, transform.position);
             }
         }
         if (rb2d.velocity.y > MaxVelocity) // da onemoguci lopti da skace
previsoko
         {
             rb2d.AddForce(Vector2.down * 30);
         }

         if (rb2d.velocity.y <  MaxDownward) // max velocity kada pada y ispod
0 dokle moze da pada
         {
             rb2d.AddForce(Vector2.up * intensity);
         }
     }

        .

        .

        .
}
```



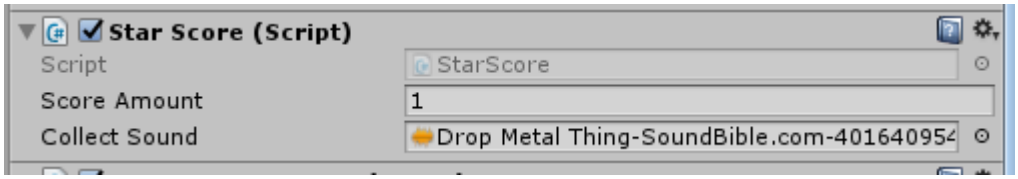
Slika 78 Naziv Audio Clipa za lopticu

ZVEZDA

Za zvezdu želimo da se čuje zvuk samo pri koliziji sa lopticom koja će da sakupi poene dodirom zvezdice. Ažuriramo skritpu ***StarScore.cs*** I u Inspector prozoru dodajemo Audio clip iz foldera **Assets/Audio/** kao u prethodnom primeru prevlačenjem u polje ***collectsound***.

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

public class StarScore : MonoBehaviour
{
    public int scoreAmount = 1;
```

```
    public AudioClip collectSound;

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.tag == "Player")
        {
            MenagerReferences refs =
GameObject.Find("Menager").GetComponent<MenagerReferences>();
            refs.IncreaseScore(scoreAmount);
            AudioSource.PlayClipAtPoint(collectSound, transform.position);
            //this.gameObject.SetActive(false); // da bude nevidljiva zvezda
ili
            Destroy(gameObject); // da unisti zvezdu
        }
    }
}
```

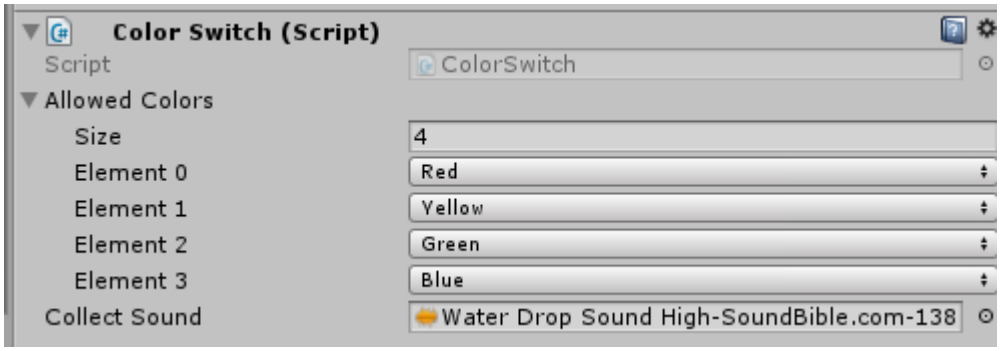


Slika 79 dodavanje audio clipa za zvezdu

COLOR SWITCH

ZIsto kao I za zvezdu želimo da se čuje zvuk samo pri koliziji sa lopticom koja će da sakupi objekat za promenu boje loptice nakon čega se objekat uništava. Ažuriramo skritpu ***ColorSwitch.cs*** I u Inspector prozoru dodajemo Audio clip iz foldera **Assets/Audio/** kao u prethodnom primeru prevlačenjem u polje ***collectsound***.

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

public class ColorSwitch : MonoBehaviour {
    public ColorUsed[] AllowedColors; //niz koji ce pamtiti boje iz enuma
    public AudioClip collectSound;
    void OnTriggerEnter2D(Collider2D other) // kad dodje do kolizije sa lopticom
    {
        if (other.gameObject.tag == "Player") //objekat igrac za kojeg zelimo da
      promenimo boju
        {
            MenagerReferences refs =
GameObject.Find("Menager").GetComponent<MenagerReferences>();

            int index = Random.Range(0, AllowedColors.Length - 1); // nasumicno bira
boju iz intervala boja
            ColorUsed newColor = AllowedColors[index];
            refs.SwitchPlayerColor(newColor); //metod iz menagera
            AudioSource.PlayClipAtPoint(collectSound, transform.position);
                        //this.gameObject.SetActive(false); // da bude nevidljiva
             zvezda ili
            Destroy(gameObject); // da unisti zvezdu , ili gameObject ili this
        }
    }
}
```



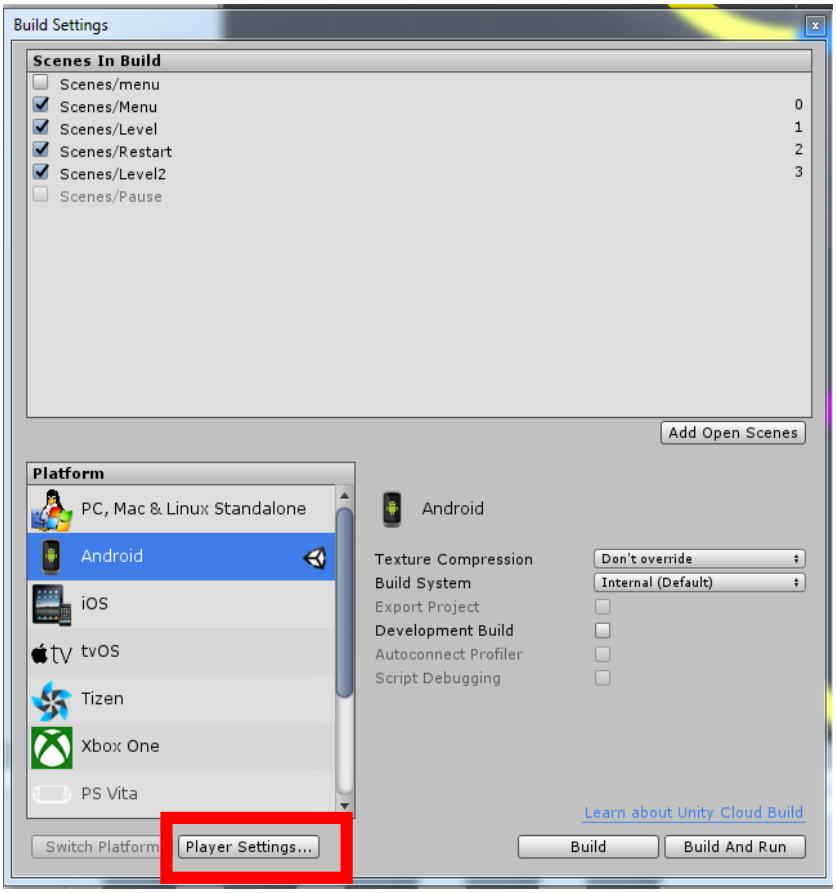
Slika 80 Dodavanje audio Clipa za color switch

9. Puštanje igrice u svet - Otpremljivanje Color Switch-a za Google Play

Sada kada je naša igrica napravljena I imamo krajnji proizvod **Color Switch** igricu ako želimo da je otpremimo (upload) na internet treba da znamo par koraka. Pošto je za android odabraćemo **Google Play**. Ako pravite za Apple iOS onda idete na Apple Store.

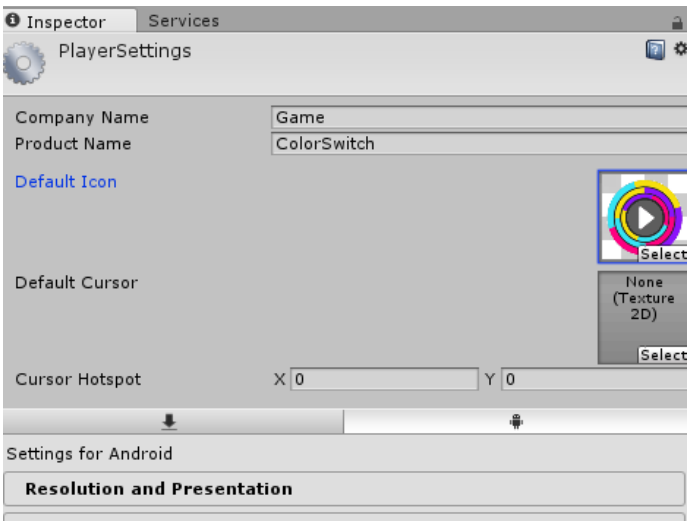
Prvo što treba da podesimo jeste Player Settings koji se nalazi u

File->Build Settings-> Player Settings kao na slici 80.



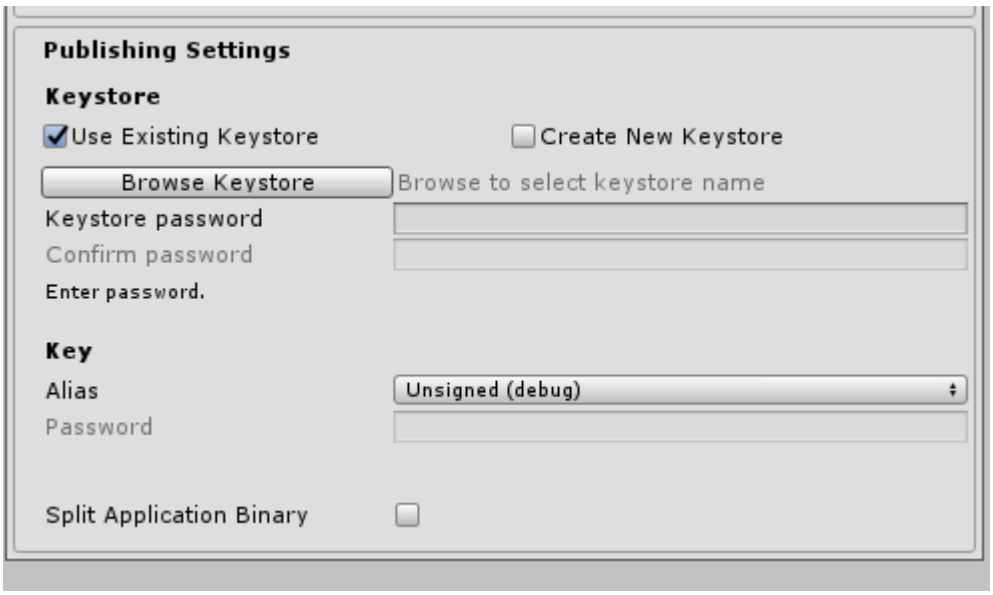
Slika 81 Player Settings

Odabirom Opcije Player Settings u Inspector prozoru će se otvoriti podešavanja I videćemo da u gornjem desnom uglu stoji polje za ikonicu naše igrice , iz foldera **Assets/Sprites/color-switch** dodajte ikonicu odabirom na dugme ***Select.***



Slika 82 Dodavanje ikonice igrici za android

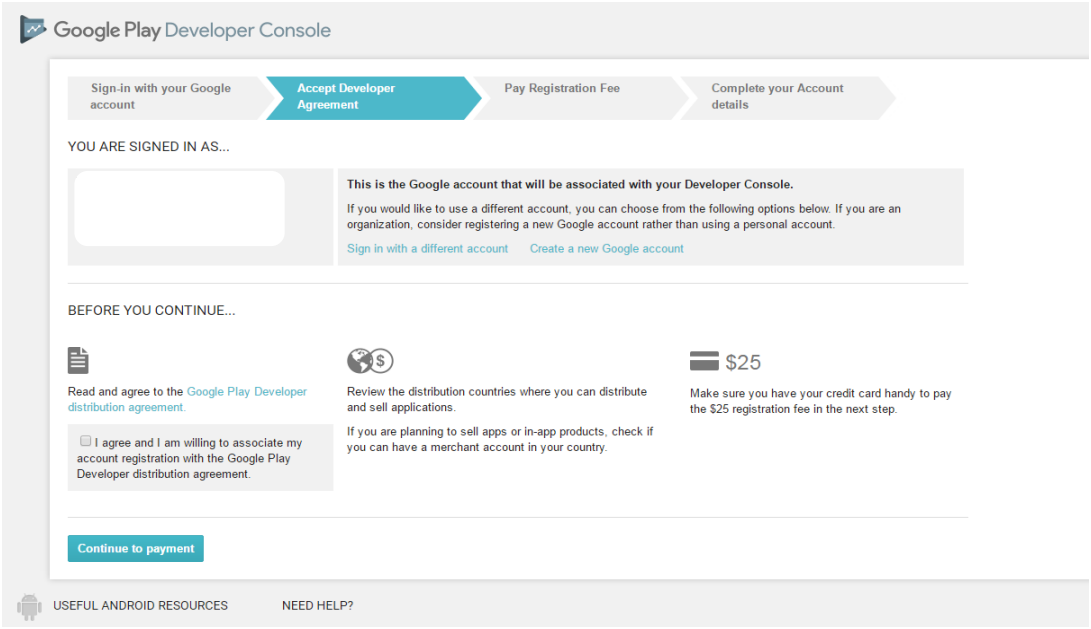
Odaberite opciju **ikonice** za **android** u donjem desnom uglu kao sa slike 81. Sada treba da se u opcijama Publishing Settings kreira Key kako mi Apk aplikacija mogla da se posle ažurira. Ukoliko kreirate novi keystore onda odaberete opciju ***create new keyStore*** a zatim unesete šifru po kojoj ćete imati pristup da aćurirate vašu aplikaciju. Kada imate već postojeći keystore onda samo odaberete opciju ***use Existing Keystore*** I ukucate šifru. Keystore tražite u **browse keystore**.



Slika 83 Keystore za ažuriranja APK

Da bi postali ***Google Developer*** morate za android uplatiti 25$ godišnje što je znatno jeftinije od članarine za iOS koji je 99$. Odete na link a posle pratite uputstva sa googl-u ili na internet nađite kako se objavljuje igrica da svi mogu da je preuzmu.

https://play.google.com/apps/publish/signup/



Slika 84 Google Play objavljivanje apk