COMTRADE

MožešIT 2017

# STAR WARS

GAMING WITH UNITY


**Autori:**

Strahinja Milosavljević

Jovana Radovanović

# Sadržaj:

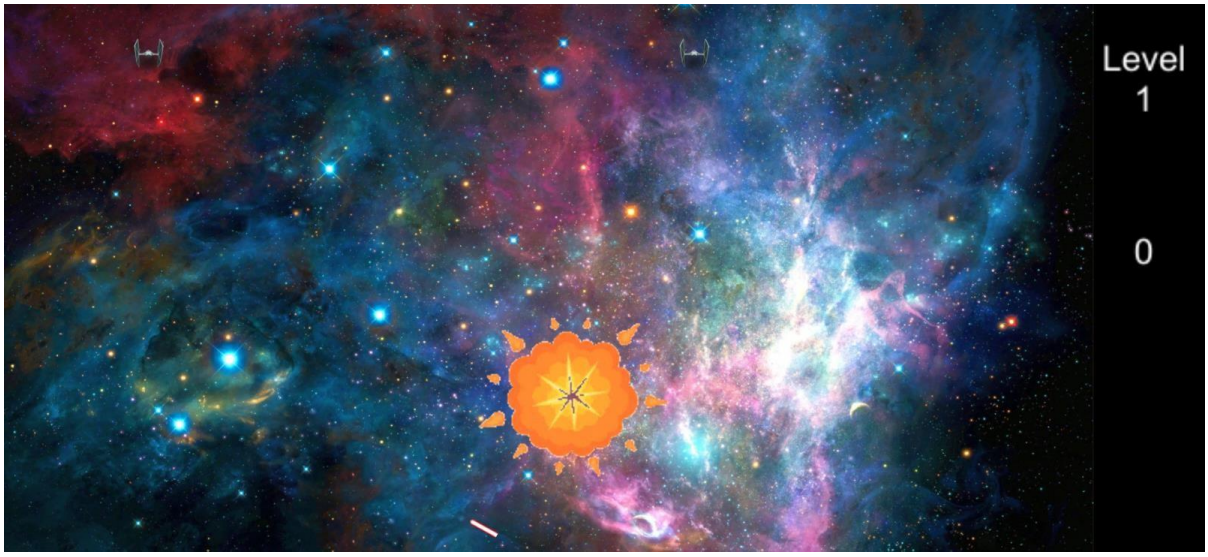
# 1. Kratak pregled - Uvod

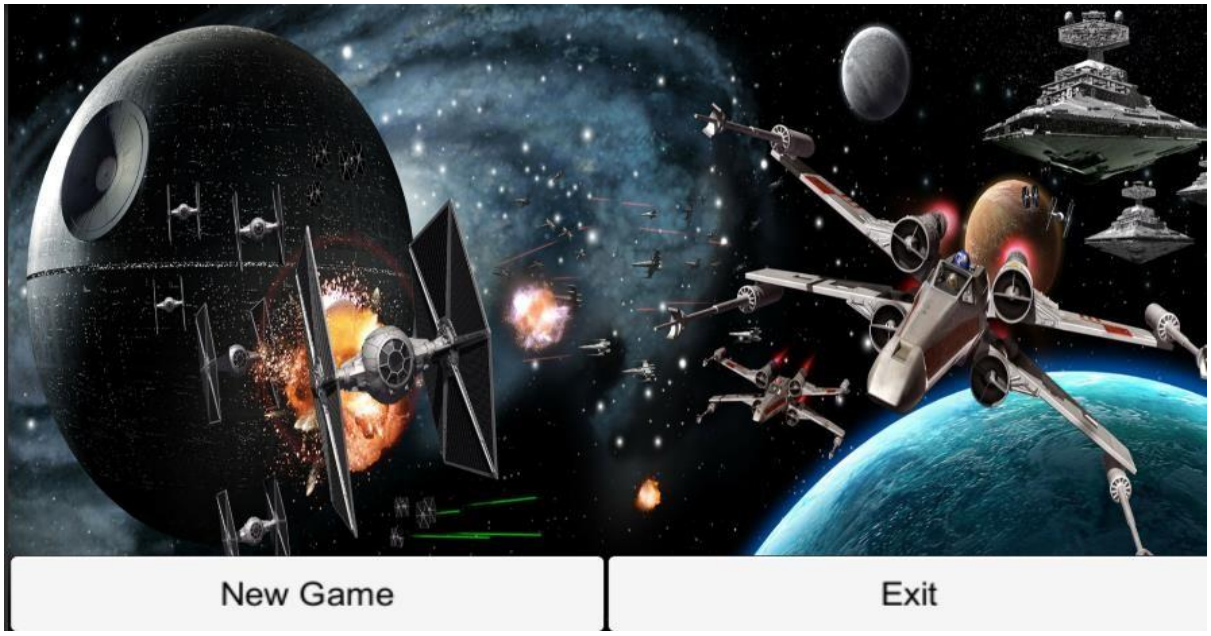
Kreiranje igara je jako zanimljiv i vrlo kretivan proces koji može biti vrlo interesantan i maštovit ako znate šta želite i na koji način da ostvarite to. Kreiranje dizajna sveta, upravljanje stanovnicima istog kao i sam proces pokretanja nečega što ste sami napravili, vaših ruku delo, je veliki uspeh i zato ćemo se u ovoj skripti pozabaviti jednom krajnje prostom igricom koja će predstavljati repliku Star Wars svemirskih ratova sa brodićem kao našim glavnim protogonistom i par njegovih neprijateljskih aviona. Kroz skriptu ćete moći da pratite detaljna uputstva od samog početka kreiranja igrice, kako biste na kraju imali jednu jednostavnu gotovu igricu na osnovu koje ćete imati početno znanje za sve ono što je bitno da jedna igrica sadrži.

Sada ćemo navesti šta Vam je potrebno da imate instalirano na Vašem računaru i na šta da obratite pažnju pre nego što krenemo u kreiranje sveta:

1. **Instaliran Unity editor , najbolje da imate verziju 5.5.2** jer ćemo u njemu raditi.
2. Unity može nekad praviti problem sa verzijom pa da imate kao napomenu da o tome treba viditi računa , poslednja verzija je 5.6.0
3. Unity skinite sa oficijalnog sajta Unitija
4. Neki editor u kome možete da kodirate skripte, Unity Vam nudi MonoDevelop kao default-ni pri instalaciji a možete i Visual Studio 2015 ili neki drugi.
5. Kodiranje ćemo pisati u C# jeziku
6. Često čuvajte projekat i kreirane scene da ne bi došlo do gubljenja nekih podataka pri nepredviđenim situacijama
7. Vodite računa o preglednosti vašeg projekta, trebalo bi da Vam folder *Assets* bude uredno raspoređen kako biste lakše našli greške, a i kako bi neko drugi mogao da ima uvid u to gde se šta nalazi.

## 1.1 Izgled krajnjeg proizvoda

## 1.2 Instalacija Unity editora

Svaka igra/program pre nego što dobije svoju krajnju verziju i ugleda svetlost dana mora da prođe kroz niz procesa kako bi dostigla nivo pokretanja:

- Instalacija okruženja u kome će se igrica dizajnirati - Unity
- Instalacija editor za kodiranje objekata – MonoDevelop/Visual Studio
- Rad u Unity okruženju
- Kodiranje C# skripti
- Provera kompajlerskih grešaka
- Pokretanje igrice

Zato u ovom poglavlju ćemo proći kroz korake koji su Vam neophodni da biste imali spremno okruženje u kome možete da radite. Za početak sa sajta :

https://unity3d.com/get-unity/download/archive

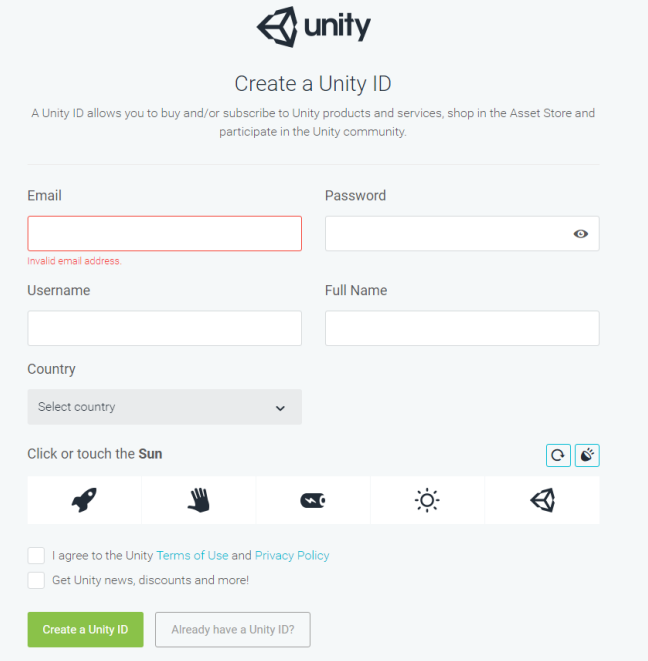
Preuzmite Unity verziju **5.5.2** kako ne bi došlo do problema pri pokretanju projekta na drugim verzijama pošto nove verzije su vrlo česta pojava , mi ćemo u ovoj verziji raditi pa je zato najbolje da nju i vi instalirate.

Kada se skine instalacija jednostavnim klikom pokrenete i pratite korake za pravilno instaliranje editor. Morate da prihvatite njihove uslove korišćenja i u instalaciji ćete biti pitani da odaberete opciju da Vam uz Unity instalira i MonoDevelop editor za kodiranje. Čekirajte opciju ili ako ne želite možete da odaberete editor po svojoj želji ili da instalirate Visual Studio (naša verzija je 2015). Instalaciju Visual Studija ili nekog drugog editor za kodiranje možete naći na You Tube-u , Googlu ili po forumima sa detaljnijim opisom i koracima. ***Mi ćemo se ovde fokusirati na instalaciju Unity okruženja jer je to svima zajedničko za kreiranje naše igrice.***

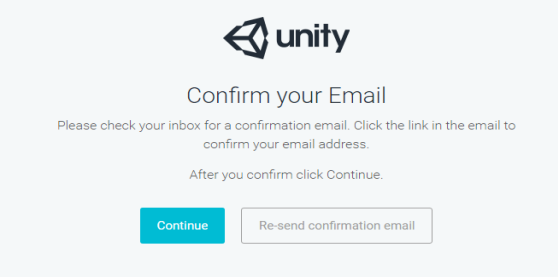
Praćenjem svih koraka doćićete do kraja instačacijonog procesa koji se završava kreiranjem Desktop ikonice Unity editora. Sledeći korak jeste kreiranje Free naloga koji je dostupan svima i potreban za prijavljivanje i izradu projekta i kako bi mogli da pristupite prodavnici sa koje možete skidati već gotove template-ove , sprajtove, animacije i ostalo.

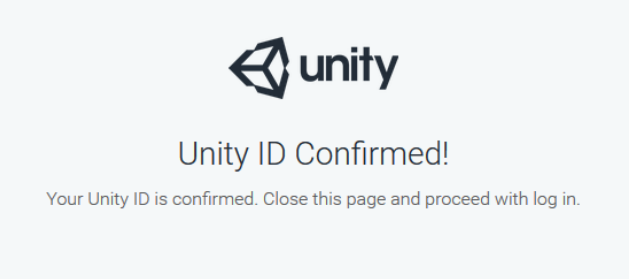
Klikom na link idete na stranu za kreiranje naloga.
https://id.unity.com/en/conversations/5dac9435-a3b1-4a64-a5ae-8526e0929e1d009f?view=register

Pojaviće se prozor kao na slici:



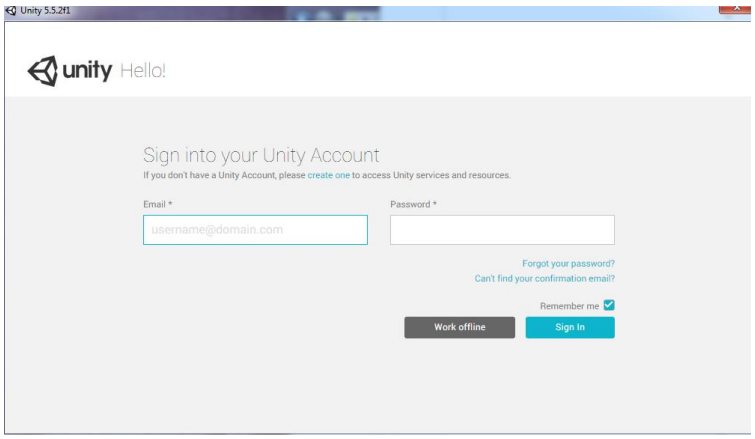

Popunite neophodna polja odaberite sliku koja ilustruje zadatu reč i prihvatite uslove korišćenja i kliknite **Create Unity ID.** Posle toga treba da potvrdite Email i pojaviće Vam se prozor, kada potvrdite preko mejla obavestiće Vas kako ste uspešno kreirali nalog.




Kada ste obavili sve prethodne korake morate se prijaviti na Vaš nalog

Pri prijavljivanju ukoliko Vas pita u koje svrhe ćete koristiti Unity odaberite opciju “Ne u poslovne svrhe već za učenje” . Ukoliko Vam iskoče prozori za dodatna pitanja označite odgovore koji odgovaraju Vašem profile. Ankete ne morate da radite samo kliknete ***Skip.***




***Time je Vaš nalog spreman i možete da kreirate nov projekat.***

# 2. Uvod u radno okruženje

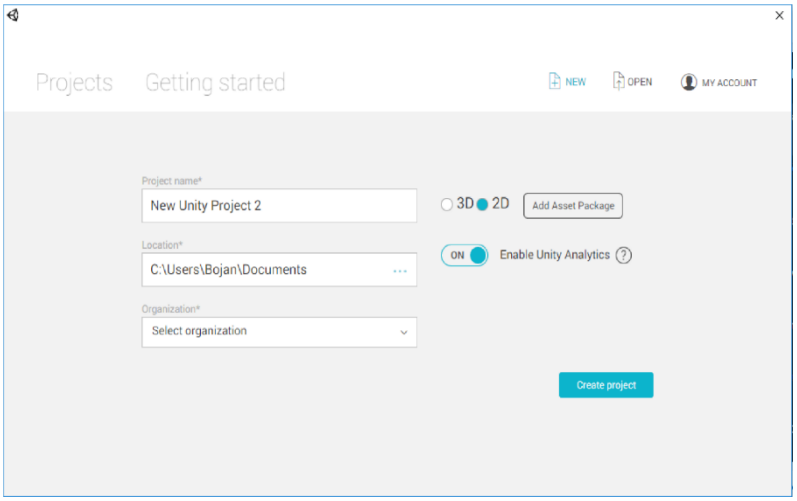
## 2.1 Šta ćemo naučiti?

- Šta je Unity engine?
- Uvod u Okruženje?
- Osnovni pojmovi
- Sprajtovi
- Pisanje skripti i dodeljivanje skripti objektima
- Kreiranje i čuvanje prve Scene
- Šta je Canvas ?
- Kreiranje prvog Nivoa
- Dodavanje neprijatelja
- Kreiranje kolizije
- Dodavanje animacija

## 2.2 Šta je Unity?

- Unity je game engine uz pomoć koga se prave 2D i 3D igre.
- Sa lakoćom se može razvijati za razne platforme.
- Game engine je softver koji pruža osnovne funkcije koje se često koriste za razvoj video igara.

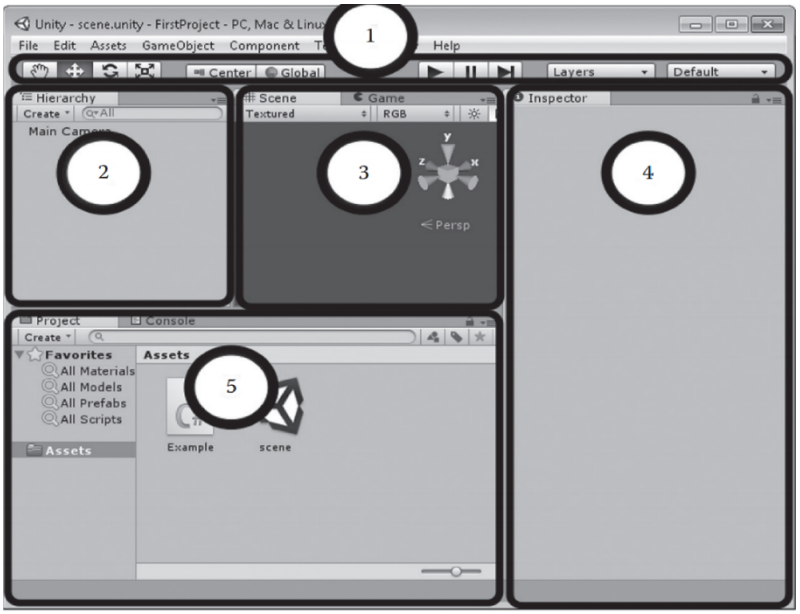
## 2.3 Prozor za kreiranje projekta



- Prilikom kreiranja projekta, Unity u direktorijumu projekta kreira različite fajlove i direktorijume. Ono što je nama bitno jeste folder **Assets**.


- **Assets** – Primarni direktorijum za objekte za igru kao što su modeli, teksture, zvukovi i skripte.
- U okviru ovog foldera bi bila dobra ideja držati fajlove organizovane u raznim direktorijimuma, kako se fajlovi ne bi mešali.

## 2.4 Okruženje Unity programa




*Slika 1 Okruženje Unity-ja*

Kada napravimo projekat, otvara se glavni editorski panel koji je podeljen u pet glavnih delova:

- **Toolbar**
- **Hiararchy panel**
- **Scene and Game view**
- **Inspector Panel**
- **Project view**

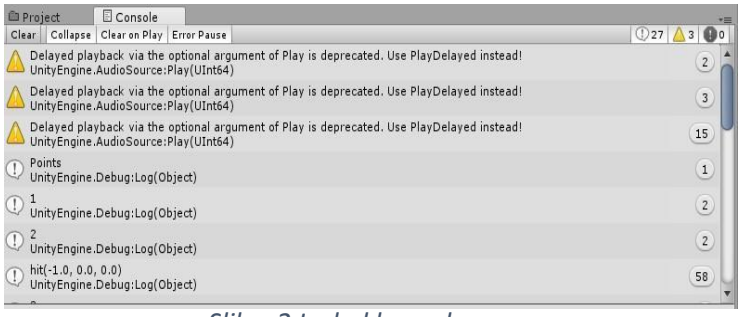
**Toolbar** sadrži u sebi alate za navigaciju, manipulaciju kao i dugmiće Play, Pause i Step Forward. **Play** dugme ćemo uvek koristiti da testiramo kod. Kada pritisnemo **play** dugme svi kodovi i scene će se pokrenuti.

**Hiararchy panel** sadrži sve objekte koji će se naći u okviru scene. Nova scena će sadržati samo „Main Camera“.

**Scene panel** je konstruktivni deo igre i vidljiv je samo programeru. U Scene panel-u se pakuju i rasporedjuju objekti. **Game panel** je pogled na igru kroz glavnu kameru (Main Camera). Tu je i **Assets Store** gde možemo kupiti ili besplatno skinuti gotove delove  za projekte.

U **Inspector prozoru** selektovanjem objekta u okviru Scene, ili Hiararchy panela možete videti i menjati njegove osobine.

**Project panel** sadrži folder Assets koji ima razne objekte koje možemo ubaciti u scenu.




*Slika  2 Izgled konzole*

## 2.5 Pisanje skripti

1. Za programiranje Unity-a koristićemo C# skripte.
2. Skriptu možemo kreirati i dodeliti objektu na više načina.
3. Desnim klikom na Project Panel i izaberemo Create → C# Script.

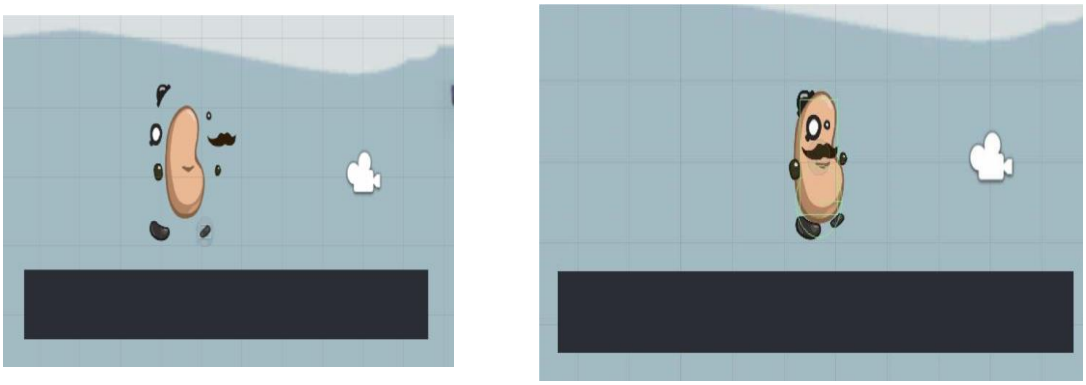
## 2.6 Čuvanje scena i projekata

- Kada završimo sa postavljanjem objekata u scenu možemo snimiti scenu kako bismo je kasnije opet koristili.
- Poželjno je snimiti scenu u **Scene** folder u okviru **Assets** direktorijuma
- Možemo kreirati više scena i lako menjati scene u C# skirpti uz pomoć metode SceneManager.LoadScene(„Ime Scene");

## 2.7 Rad sa sprajtovima

- U unity-u 2D Sprite-ovi su slike koje uglavnom prikazuju jedan objekat.
- Nekoliko Sprite-ova može činiti jedan objekat, u pojedinačnim frejmovima kako bi omogućilo animaciju lika. Na slici možemo videti objekat sačinjen od više Sprite-ova.
- Kada se prenesu na scenu automatski dobijaju **Sprite Renderer** komponentu koji je spreman da se prikaže u igri.
- Sprite Renderer je novi 2D prikazivač koji iscrtava Sprite-ove na ekran.
- U okviru Sprite Renderer komponente možemo videti ime sprite-a

Možemo menjati i boju, okretati Sprite po x i y osi, menjati materijal Sprite-a, birati layer u kome se nalazi i određivati njegov raspored u layer-u (sloju).

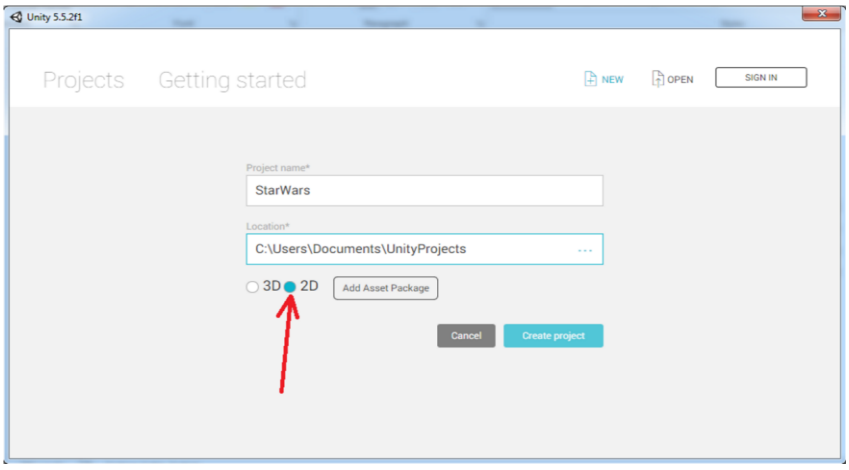


*Slika 3 Sprites*

# 3. Glavni meni prozor za Star Wars

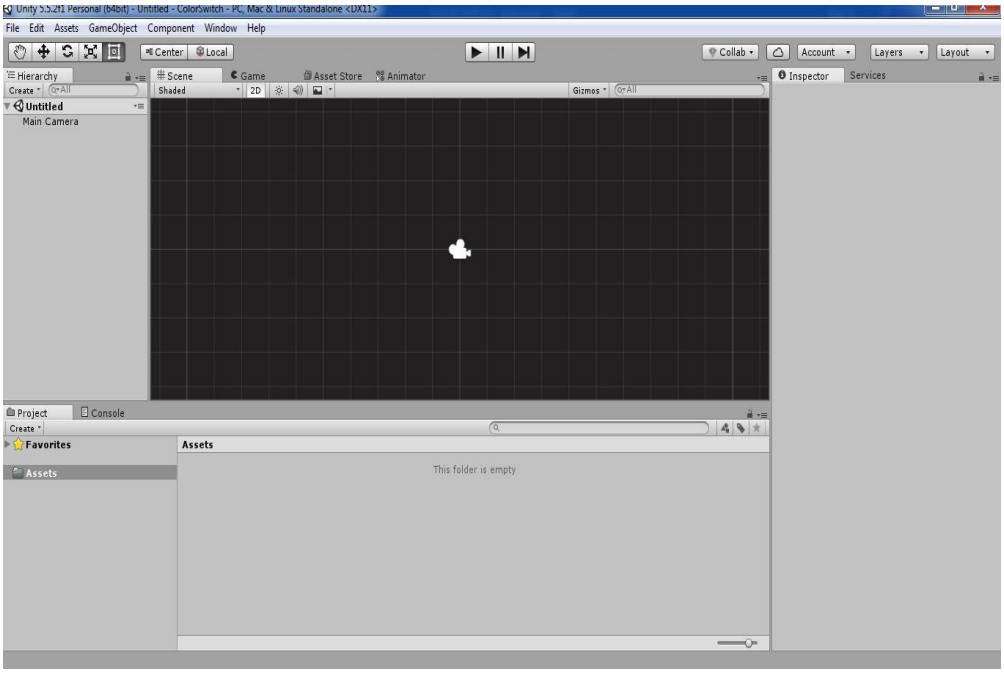
## 3.1 Kreiranje našeg projekta

U ovom poglavlju započinjemo izradu naše igrice. Pokrenuti program u kome ćemo kreirati našu igricu **Unity** 5.5.2. 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 **StarWars**, a ***Location*** predstavlja mesto gde će se naš projekat čuvati. Lokaciju možemo lako promeniti odabirom „...“.



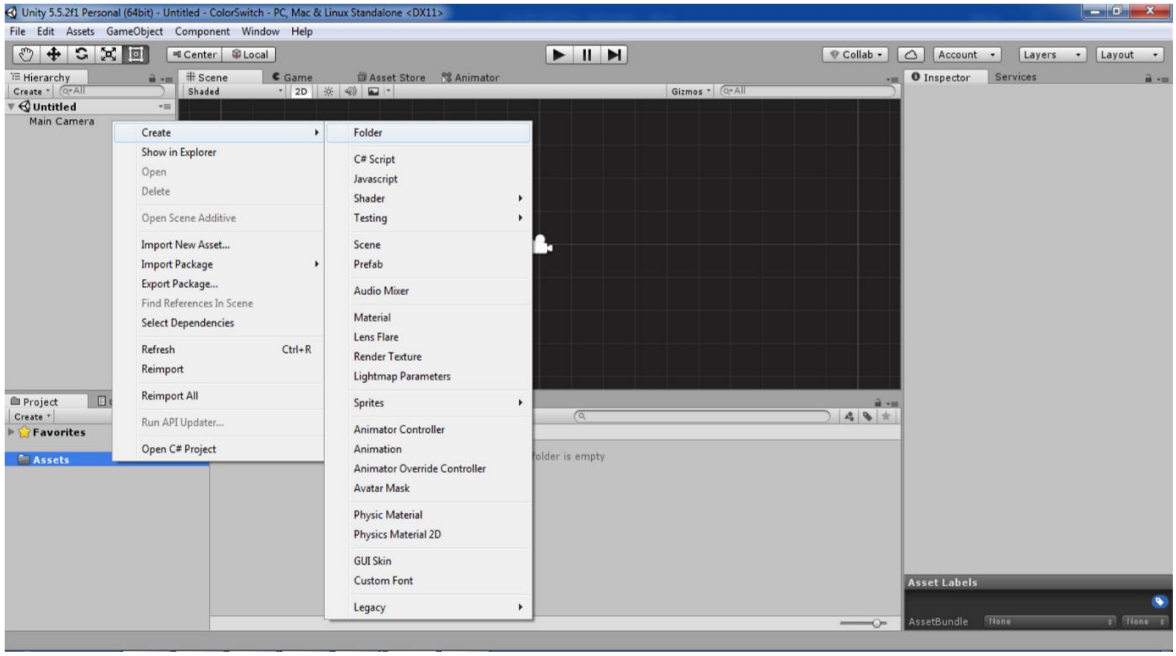

*Slika 4 Kreiranje novog projekta*

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



## 3.2 Kreiranje scene za Meni

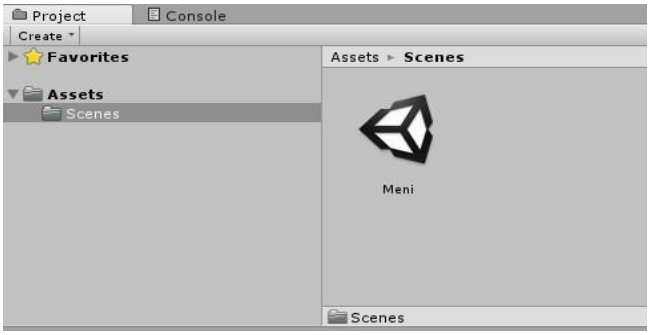
Prvi korak pri kreiranju igrice je snimanje scene. **Scene** ćemo čuvati u okviru foldera ***Scenes***, koji će biti podfolder foldera ***Assets*** (desni klik na folder *Assets >Create ->Folder-> imenujemo ga u Scenes*).




*Slika 5 Kreiranje foldera Scenes*

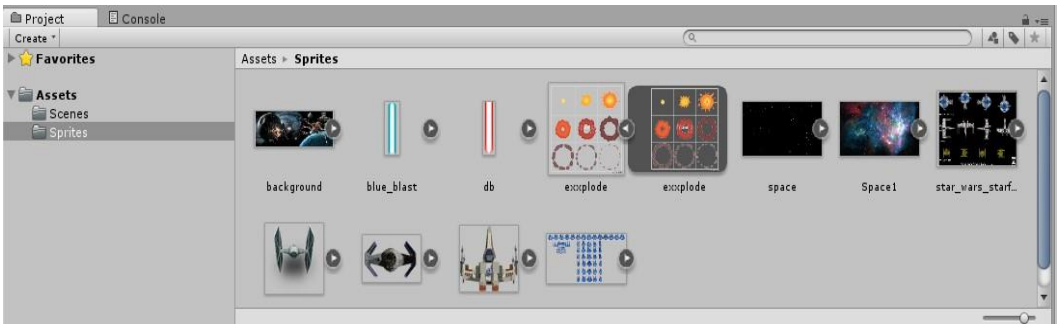
Scenu snimamo na sledeći način:

*File->Save Scenes->nakon čega biramo folder Scenes ->scenu čuvamo podnazivom Meni.*




*Slika 6 Scena Meni*

Sledeći korak jeste kreiranje foldera ***Sprites***. Uvozimo sve Sprite-ove iz foldera Materijali sa Desktop-a, u podfolder **Assets/*Sprites/*.**



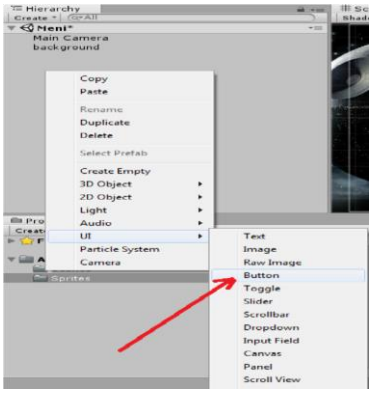

*Slika 7 Dodati Sprite-ovi*

Sada sprite pod nazivom ***background***prevučemo na scenu, kako bi dobili pozadinu scene **Meni**.

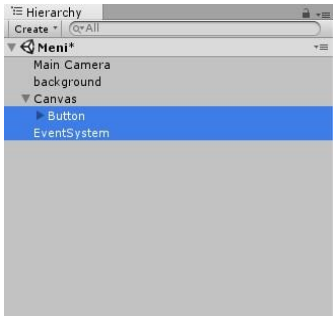

*Slika 8 Prebacivanje background na scenu*

## 3.3 Kreiranje dugmića za meni

Sledeći korak jeste kreiranje dva dugmeta. Jedno koji nas vodi na glavnu scenu, a drugo koje služi za izlaz iz aplikacije. Postupak kreiranja dugmeta je sledeći: U okviru panela ***Hiararhy*** **-> desni klik -> UI -> Button** .

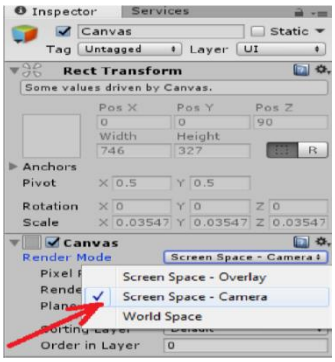



*Slika 9 Kreiranje Button-a*




Nakon ovog koraka u okviru objekta ***Canvas*** kreira se dugme. *Canvas* je prostor u okviru koga se smeštaju **UI elementi** (dugmići, tekstovi, paneli...). Kreiranjem bilo kog UI elementa kreira se i Canvas, u okviru koga se smešta taj element. Dakle, ***Canvas*** je roditelj za bilo koji kreirani ***UI element***.

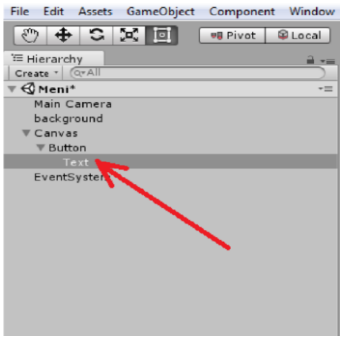
*Slika 10 Button u okviru Canvas-a*




Klikom na kreirani **Canvas** u ***Inspector*** prozoru vršimo podešavanja za isti. Da bi se ***Canvas*** video u okviru **Main Camere** vršimo sledeća podešavanja u okviru komponente ***Canvas***. Za ***Render Mode*** odaberemo *Screen Space – Camera* , a u ***Render Camera*** prevučemo ***Main Cameru*** iz Hiararchy panela.

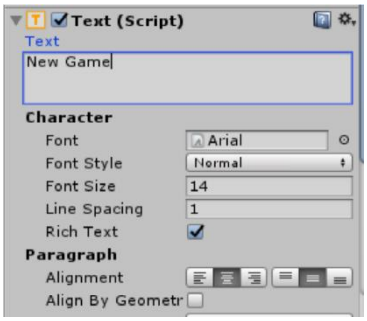

*Slika 12 Prevučemo Main Cameru u polje Render Camera*

Sada pozicioniramo dugme na mesto koje želimo, i promenimo tekst na sledeći način: kliknemo na **Button**, pa na podmeni **Text**.




*Slika 13 Text u okviru Buttona*

Nakon ovog koraka u ***Inspector*** prozoru se pojavljuje komponenta *Text,* gde unosimo naziv dugmeta:




*Slika 14 Ovde unosimo naziv dugmeta*

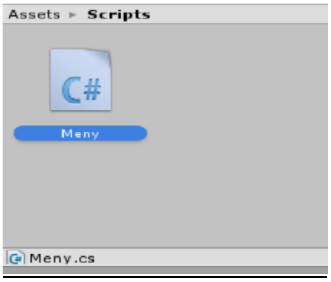
Sada kreiramo još jedno dugme za izlaz iz aplikacije u *okviru istog* ***Canvas*** *objekta*. Nazvaćemo ga ***Exit***.


*Slika 15 Meni sa dva dugmeta*

## 3.4 Kreiranje skripte Meni

Dugme **New Game** se koristi za prelazak na novu scenu, koju ćemo kasnije nazvati ***level1.*** **Exit** koristimo za izlaz iz aplikacije. Da bi dugmići mogli da obave ove funkcije, neophodno je kreirati **C#,** koje smeštamo u folder ***Scripts***, koji je

potrebno napraviti u ***Assets*** folderu. Sada kreiramo skriptu za meni: *desni klik u folderu Scripts -> Create -> C# Script-> pod nazivom Meny.*



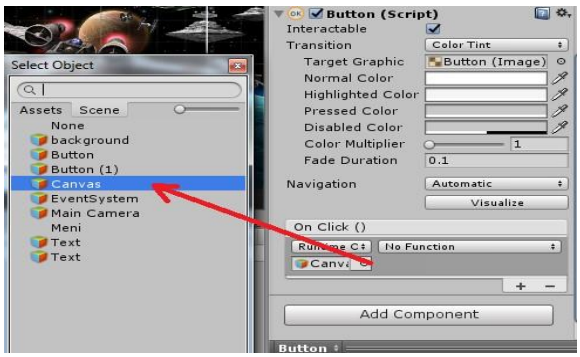

*Slika 16 Meny skripta*

Skripta ima sledeći sadržaj:

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

public class Meny : MonoBehaviour
{

      //metoda za ucitavanje nove scene
      public void NewGame()
          {
                SceneManager.LoadScene("Level");
          }

      //metoda za izlaz iz aplikacije
      Public void Exit()
          {
                Application.Quit();
          }
}
```

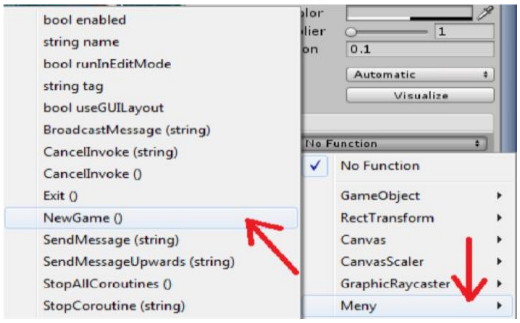
Sada u Unity okruženju dodelimo kreiranu skriptu *Canvas* objektu, tako što je prevučemo na objekat Canvas u Hiararchy panel-u.

Potrebno je dugmićima dodeliti kreirane metode u skripti Meny, kako bi obavljali definisane uloge. Klikom na prvi ***Button***, u okviru Inspector prozora , u okviru *Button (Script)* komponente podesimo šta će se desiti na klik dugmeta. U delu OnClick() te komponente na dugme **+** dodajemo događaj na klik. Kliknemo na točkić sa poljem *None*, nakon čega nam se otvara prozor gde selektujemo *Canvas*.




*Slika 17 Podešavanje klika na dugme*

Sada klikom na *NoFunction*, biramo metodu (funkciju) koju dodeljujemo dugmetu *New Game*. Biramo funkciju iz skripte *Meny* pod nazivom ***NewGame().***

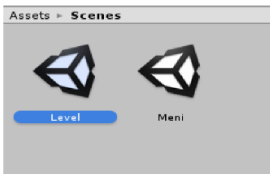



*Slika 18 Dodela funkcije dugmetu*

Postupak ponavljamo i za drugo dugme (Exit), samo što mu dodeljujemo funkciju ***Exit().***

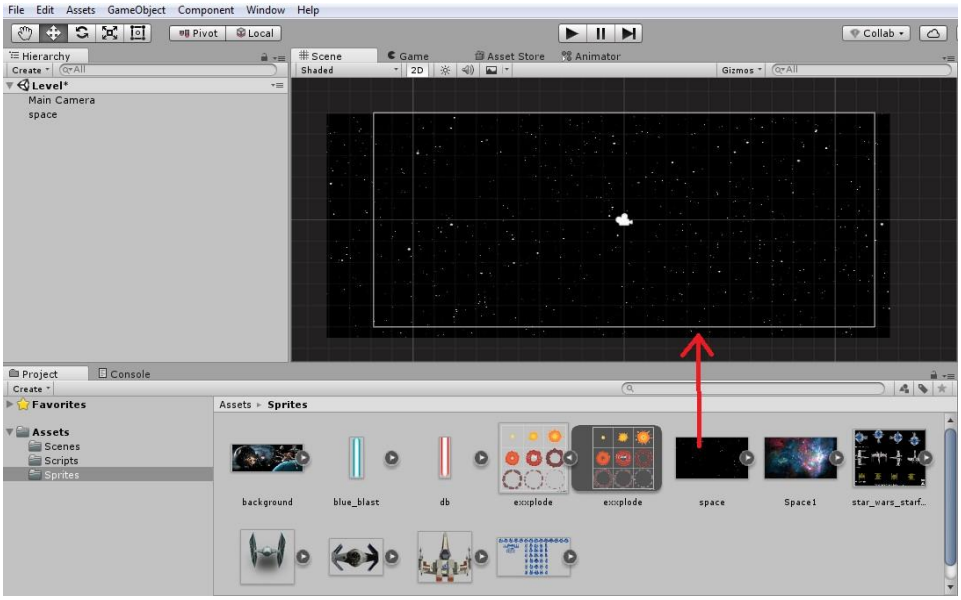
## 3.5 Kreiranje nove scene

U okviru foldera ***Scenes*** kreiramo scenu sa nazivom ***Level*** (na isti način kao što smo kreirali prethodnu scenu), kako bi se klikom na dugme New Game ta scena učitala.




*Slika 19 Level scena*

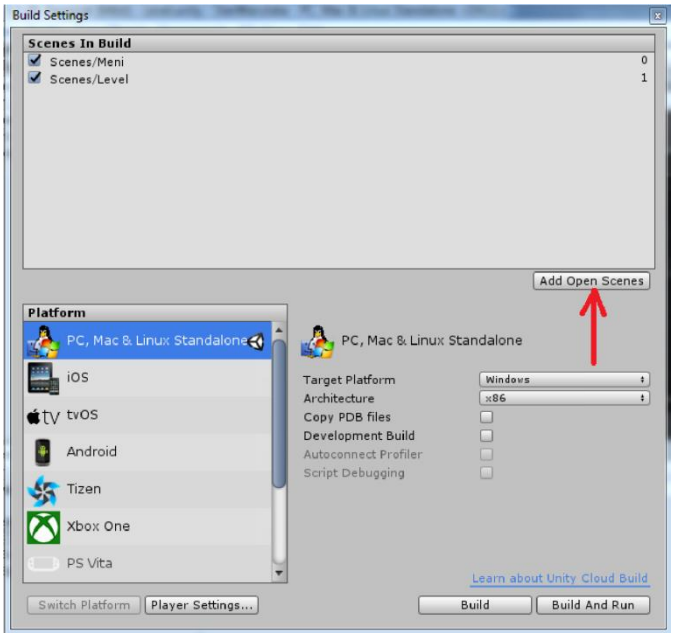
Na ovu scenu za početak samo prevučemo pozadinu iz foldera *Sprites* pod nazivom ***space***.




*Slika 20 Pozadina nove scene*

## 3.6 Bildovanje scena

Da bi mogli da pokrenemo igricu, neophodno je izbildovati obe scene. Klikom na *File-> BuildSettings* pojavljuje se prozor sa slike ispod.

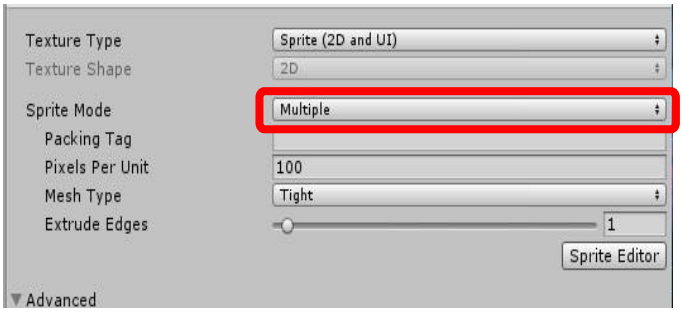



*Slika 21 Bildovanje scena*

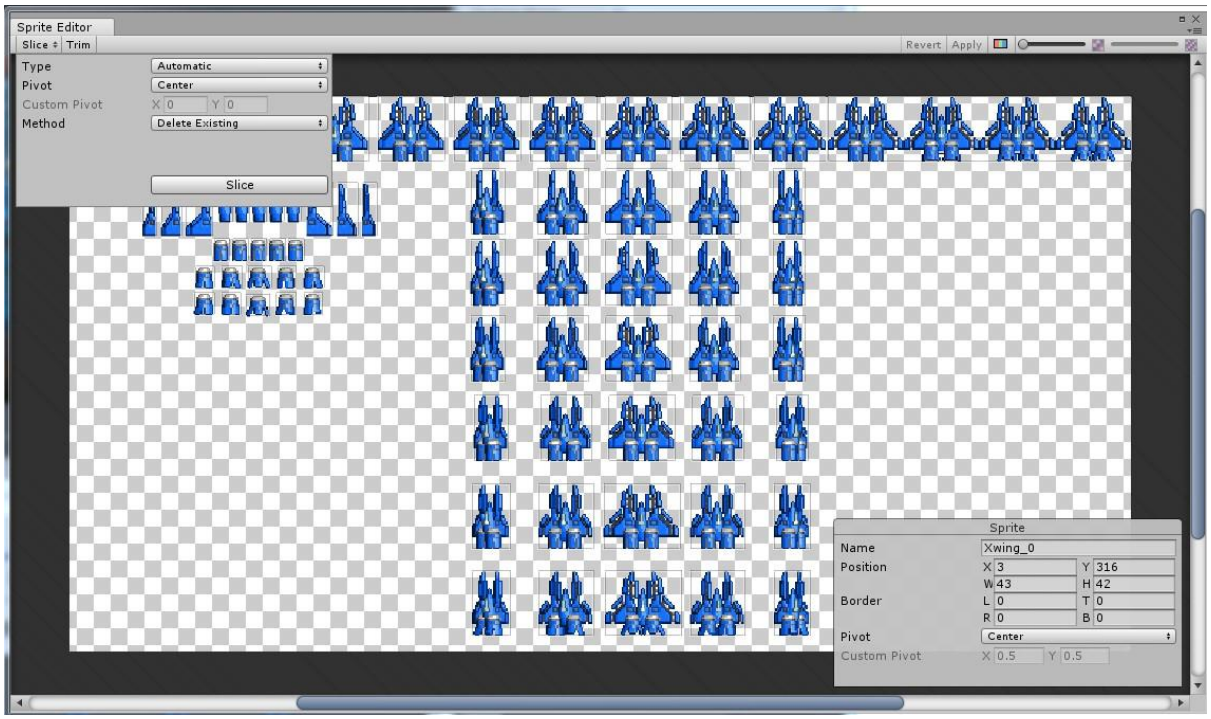
U okviru prozora *ScenesInBuild,* dodajemo scene koje smo kreirali. Klikom na dugme *Build* pravimo *.exe* file pod proizvoljnim nazivom, npr. *Game.exe*. Klikom na exe file, možemo pokrenuti igricu i videti kako radi. Prva scena koju dodate u ovo polje će biti i **default**-na scena koja će se prva pokrenuti.

# 4. Kreiranje igračevog brodića

Nakon što smo dodali scenu vreme je da dodamo i glavni brod na našu scenu. Prvo što nam treba jeste da uzmemo iz foldera **Assets/Sprites/** sprajt **Xwing** i da ga isečemo na pojedinačne sprajtove. U **Inspector** Prozoru klikom na **Xwing** pojaviće se polje **Spritemode** i tu selektujete **Multiple** da naš Sprajt od jednog napravi više pojedinačnih kao na slici.

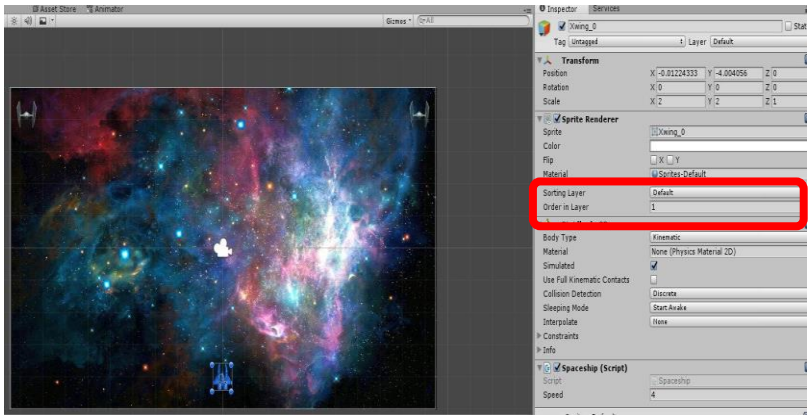


Zatim odaberete Sprite editor dugme i tu će se otvoriti prozor :




Odaberite Polje **Slice** a zatim dugme **Slice** da sam automatski editor iseče sa predefinisanim opcijama Sprajt, njih nećemo menjati. Zatim ćete u **Sprites** folderu dobiti nešto ovako:

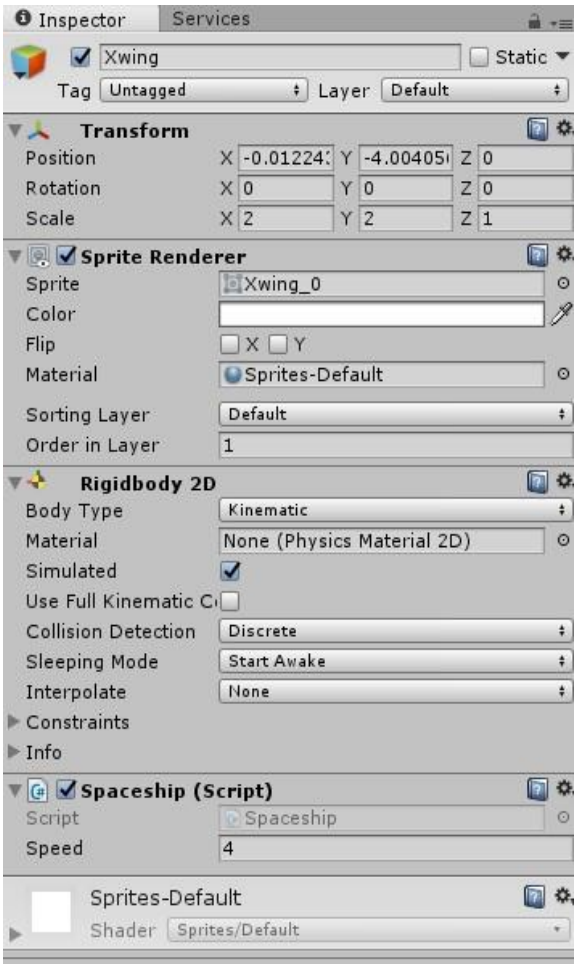

Nakon ovoga treba da odaberemo prvi Sprajt u nizu **Xwing_0** prevučemo ga na našu scenu i podesimo da sloj pozadine bude iza našeg brodića pošto Unity sve dodaje u Default-ni sloj i preklopiće **background** na brod. U inspector prozoru postavite **Order in Layer na 1** da bude ispred **background** pozadine i dobijemo sledeći izgled:



Sada je naš brod postavljen u svet. U hijararhiji promenite ime ako želite desnim klikom na **Xwing_0** ime i preimenujete u samo **Xwing**.

## 4.1 Kretanje brodića

Sada sledi da pokrenemo brod da može da se kreće i sa kojim ćemo da upravljamo na strelice tastature u zavisnosti na koju stranu želimo da idemo.




Prvo dodajemo komponentu u Inspector prozoru na Add component dugme **Rigidbody2D** za kretanje broda kao sa slike i čekiramo **isKinematic** opciju ili odaberemo iz drop down menija **Kinematic**.

Kreiraćemo novu C# skriptu u folderu **Assets/Scripts/Spaceship.cs** i možemo zatim da je prevučemo na brod, a možemo i nakon kreiranja ali je bitno da skripta bude prikačena za brod.

Skriptu možete dodati prevlačenjem miša na objekat **Xwing** ili prevlačenjem u Inspector prozor objekta xwing ili dodavanjem na **Add Component**.

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

public class Spaceship : MonoBehaviour
{
    public float speed = 4.0f; //brzina kretanja brodića
    private Rigidbody2D rb; //rigidbody za kretanje
    void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
    }
  //sada za svaki pritisnut taster  Update() metod računa novu
  poziciju koju množi sa vremenom između frejmova i brzinom
```

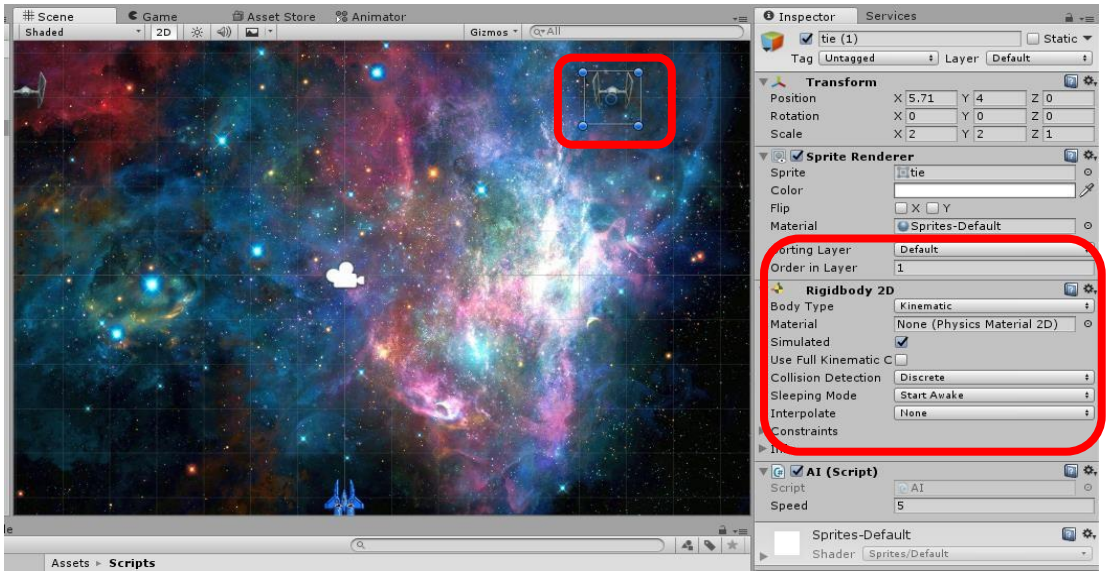
```
   void Update ()
    {
     if (Input.GetKey(KeyCode.LeftArrow) && transform.localPosition.x > -8)
        {
            transform.position += Vector3.left * speed * Time.deltaTime;
       }
if (Input.GetKey(KeyCode.RightArrow) && transform.localPosition.x < 6)
        {
            transform.position += Vector3.right * speed * Time.deltaTime;
        }
if (Input.GetKey(KeyCode.UpArrow) && transform.localPosition.y < 6)
        {
            transform.position += Vector3.up * speed * Time.deltaTime;
       }
if (Input.GetKey(KeyCode.DownArrow) && transform.localPosition.y > -4)
        {
            transform.position += Vector3.down * speed * Time.deltaTime;
        }
    }
}
```

**NAPOMENA**: Vrednosti pozicija do kojih brod može da se kreće mogu da variraju u vašem projektu drugačije od našeg tako da to podesite kako Vama odgovara i kako želite

## 4.2 Kreiranje protivnika i kretanje

Naš brod ne može da ostane sam na sceni već mora da ima i neprijatelje protiv kojih se bori. Za protivnike dovoljno je da samo da prevučemo na scenu sprajt ***tie*** iz foldera **Assets/Sprites/tie** pošto njih ne moramo da sečemo na pojedinačne. Dodaćemo dva broda koji će se kretati horizontalno po x osi sa gornje strane naše scene i isto kao i za naš glavni brod promeniti u Inspector prozoru u komponenti Sprite Renderer **Order in Layer bude 1** kako bi se nalazili ispred background pozadine. Dodajte samo da pozicija oba neprijateljska aviona bude **po y osi 4** kako bi bili u ravni.

U **Inspector** prozoru dodajte **Rigidbody2d** kao za naš brod kako bi mogli da se kreću i čekirajte kinematic opciju kao na slici.



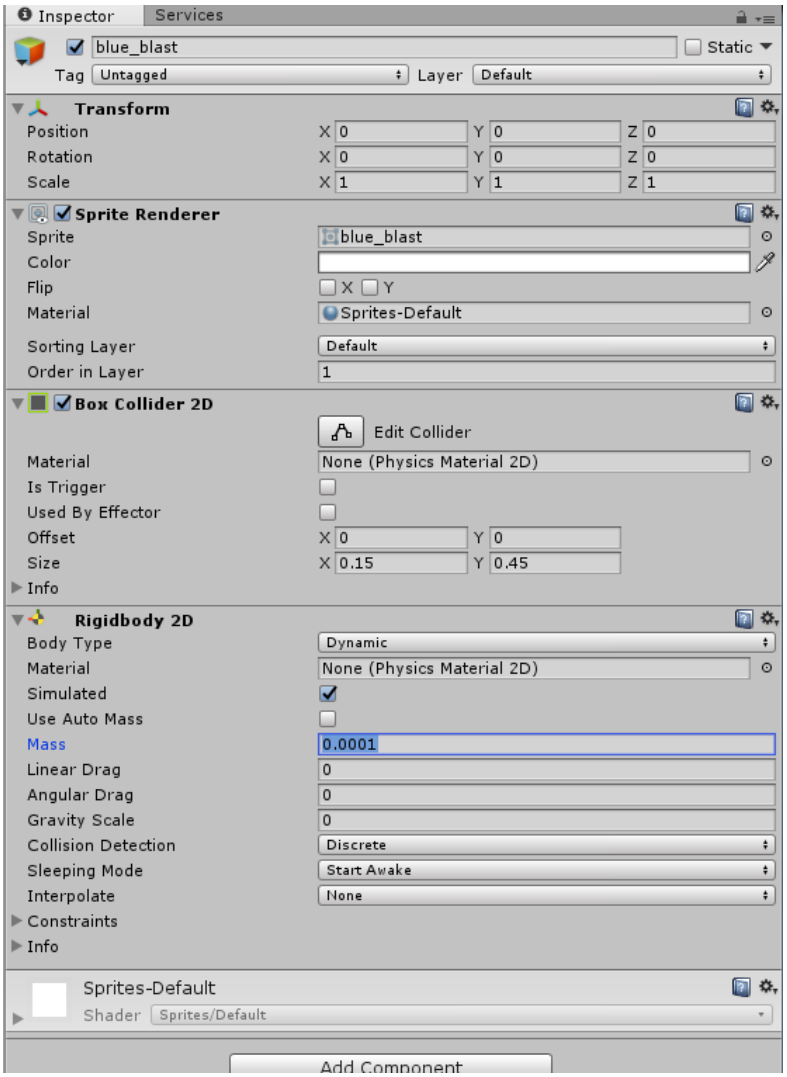

Dakle sada treba da kreiramo i skriptu za ova dva broda i da ih svakome od njih pridružimo. Napisaćemo skriptu **AI.cs** u folderu **Assets/Scripts/** sa sledećim kodom:

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

public class AI : MonoBehaviour
{
        bool movingLeft = false;
        public float speed = 4.0f; //brzina avioncica

        void Update ()
        {
             if (!movingLeft) //ako je ispunjen uslov
             if (transform.localPosition.x < 5)
             {
                  transform.position += Vector3.right * speed * Time.deltaTime;
             }
             else movingLeft = true;

             if (movingLeft)
             if (transform.localPosition.x > -7)
             {
                  transform.position += Vector3.left * speed * Time.deltaTime;
             }
            else movingLeft = false;
        }
}
```

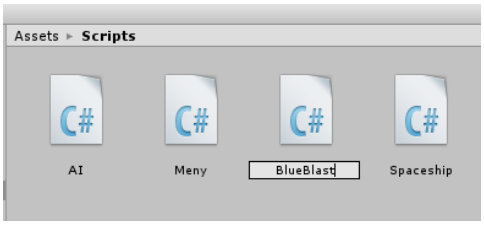
**NAPOMENA**: Vrednosti pozicija do kojih brod može da se kreće mogu da variraju u vašem projektu drugačije od našeg tako da to podesite kako Vama odgovara i kako želite .

# 5. Pucanje igračevog brodića

Sada imamo na našoj sceni naš brod i protivničke brodiće ali nam fali poenta igrice, a to je pucanje jednih na druge. Mi ćemo ovde kreirati dve vrste metaka "dobre" (plave) i "loše" (crvene). Za početak prevucimo sprajt iz foldera **Assets/Sprites/blue_blast** u hijararhiju naše scene kao sa slike. Na slici je prikazano šta da dodate od komponenti u Inspector prozoru blue_blast sprajta. Ovde postavljamo u **RigidBody2D**-u sve vrednosti na **0** jer želimo da se metak kreće pri pucanju na gore tj. da gravitacija i masa ne utiču na njega.




Sada treba da kreiramo skriptu u folderu **Assets/Scripts/BlueBlast.cs** kao sa slike kako bi metku dodelili brzinu, poziciju kao i šta će se desiti pri koliziji sa istim.

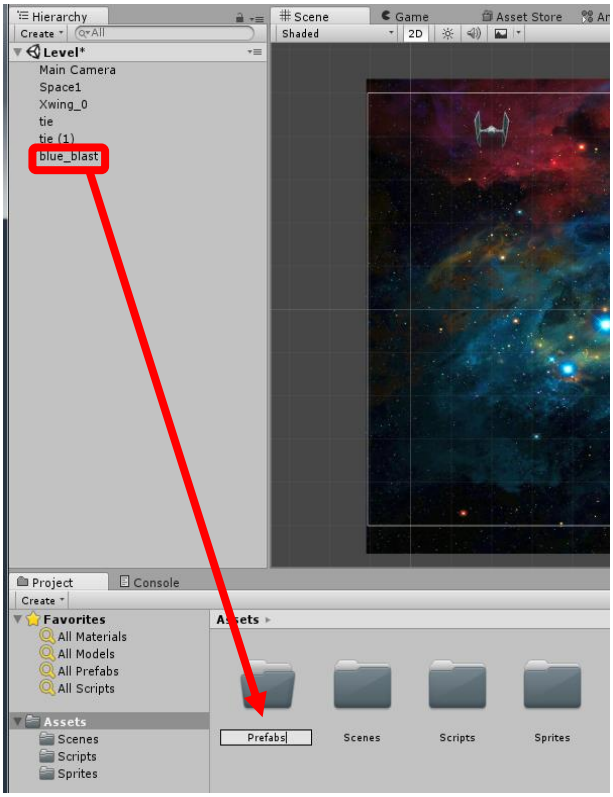



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BlueBlast : MonoBehaviour
{
    public float speed = 10f; // brzina kretanja metka
      // Update is called once per frame
      void Update ()
      {
        transform.position += Vector3.up * speed * Time.deltaTime; //pozicija
        metka i gde da se krece
      }
```

```
    void OnBecameInvisible() // kada metak ode van main kamere
    {
            enabled = false; //ugradjena bool promenljiva koja postaje false
            //posto dok se metak vidi ona je true
            Destroy(gameObject);//unisti objekat tj metak
    }

    // sta ce se desiti kada dodje u koliziju metak sa drugim objektom
    void OnCollisionEnter2D(Collision2D col)
    {
      if (col.gameObject.name != "Xwing_0" && col.gameObject.name != "blue_blast(Clone)")
         {
            Destroy(col.gameObject);
         }
    }
}
```

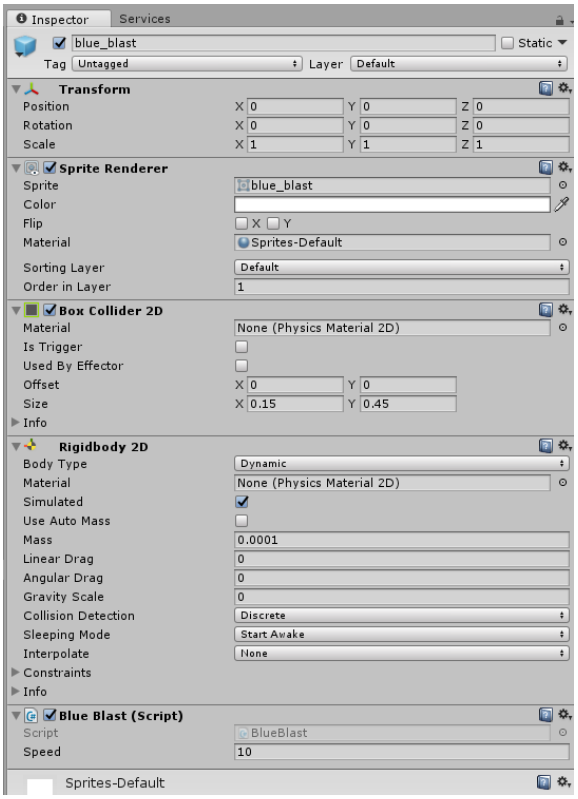
Po završetku kucanja skripte **dodelite** skriptu našem objektu **blue_blast** prevlačenjem ili preko **Add Component**. Nakon što smo to uradili treba da izmenimo još par linija koda u skripti **Assets/Scripts/Spaceship.cs** koju smo prethodno kreirali kako bi naš metak bio prikačen na naš svemirski brodić.

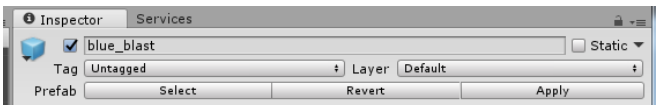
```
using UnityEngine.SceneManagement; //dodajemo zbog koriscenja metode
SceneManager koaj //vodi na drugu scenu
using UnityEngine;


public class Spaceship : MonoBehaviour {

      public GameObject blueblast;

      void Update ()
      {
            if (Input.GetKey(KeyCode.Escape))
            {
                  SceneManager.LoadScene("Meny"); //ucitaj meni scenu
            }

            .
            .
            .

            if (Input.GetKeyDown(KeyCode.Space))
            {
                  Instantiate(blueblast, transform.position,
                  Quaternion.identity);
                  // ovo je za kreiranje vise objekata
                  // koji prosledjujemo kao 1 argument sa odredjene pozicije
                  sto je //drugi argument dok treci argument sluzi da ne dodje
                  do rotacije vec //da objekat prati pravac roditelja odnosno
                  naseg broda
            }

        }

}
```



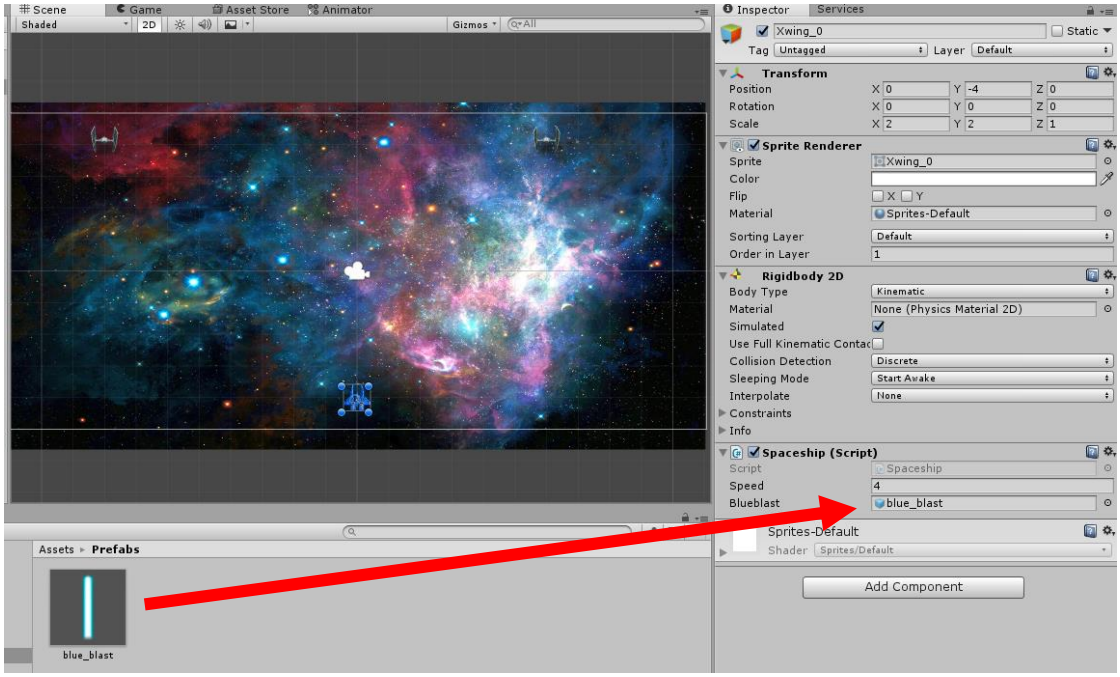

Nakon modifikovanja ove skripte kreiramo folder u **Assets/Prefabs/** i u njega prevlačimo sprajt ***blue_blast*** iz hijararhije pošto nam ne treba na sceni trenutni već samo pri pritisku na **space** tastature brišemo ga iz hijararhije i ono što možemo da primetimo jeste da u **prefabs** folderu imamo kreiran objekat koji smo izbrisali za kasniju upotrebu sa svim istim vrednostima koje smo mu prethodno dodelili. Razlog tome jeste što takav jedan folder čuva objekte koji se u **Unity** sistemu nazivaju **Prefabs** (Vi ste folder mogli nazvati kako god ali smo ipak ostali dosledni imenu onoga što se u njemu nalazi). Kada god želite da koristite više objekata na sceni nema razloga da kreirate svaki od tih objekata pojedinačno i zato definišete na jednom mestu objekat, a zatim ga samo dodeljujete kasnije gde Vam je potrebno jer zapravo Prefab-ovi su objekti sa **predefinisanim** vrednostima koje kasnije možete pozivate koliko god puta želite.

Ono što je još bitno jeste da sve dok je neki prefabs objekat na sceni (u **Hijararhiji**) i ako mu menjate bilo kakve vrednosti u **Inspector** prozoru ili dodajete komponente morate nakon toga kliknuti "**Apply**" da bi tom objektu bila prihvaćena svojstva i unutar **Prefabs** (ili kako ste već nazvali folder u koji smeštate prefabove) foldera. Ako ipak menjate svojstva tom istom objektu ali unutar **Prefabs** foldera onda **nema** potrebe za stalnim kliktanjem "**Apply**" dugmeta. Slika ispod je za objekat **blue_blast** iz hijararhije (on ima opciju Apply) dok druga slika je iz prefabs foldera u kome je objekat **blue_blast** nakon prevlačenja iz hijararhije.

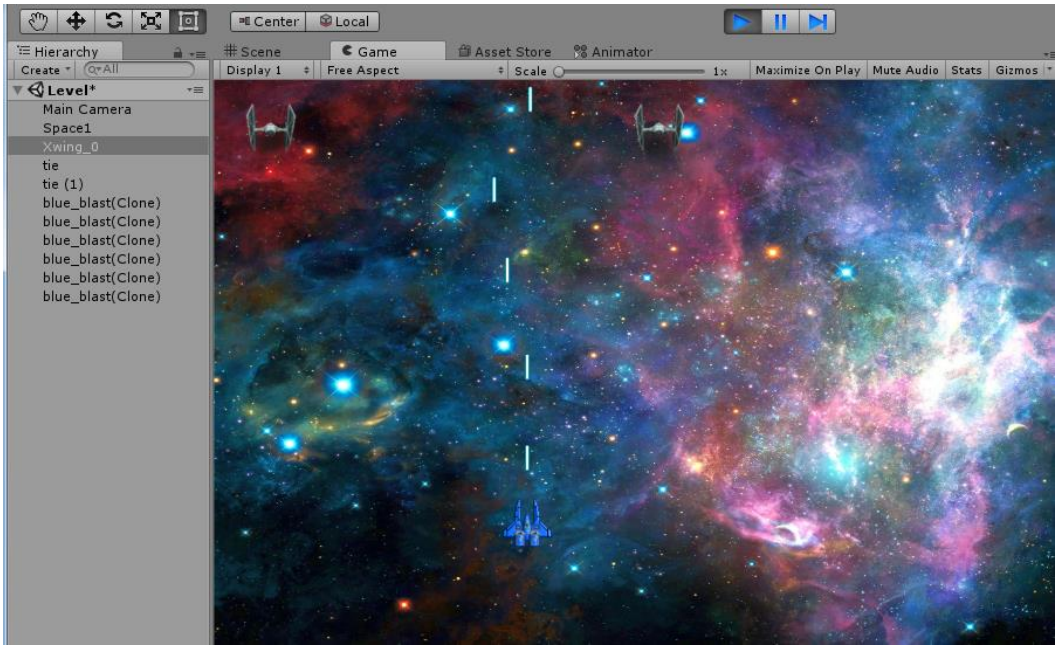


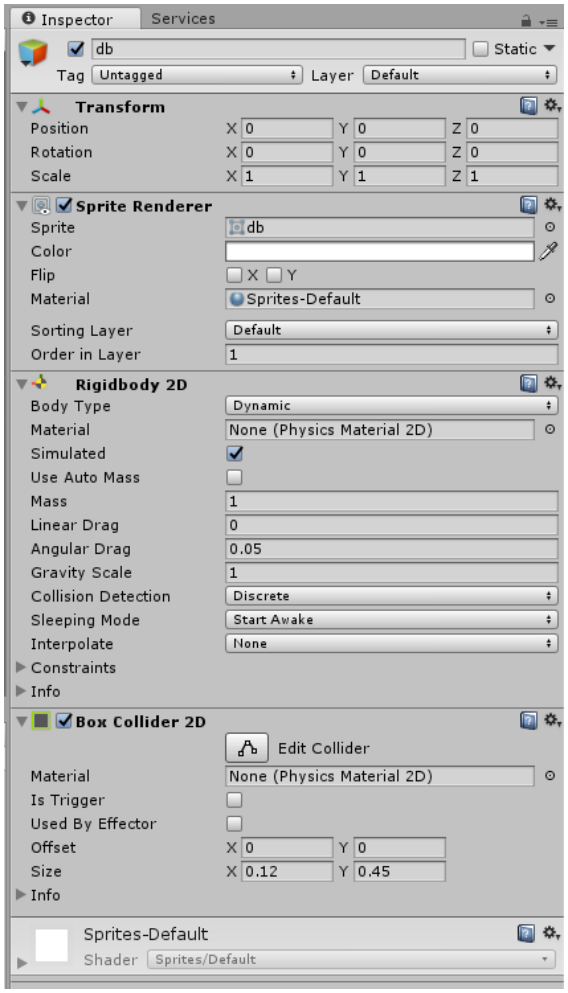
Kada smo dodali objekat blue_blast u Prefabs folder ako niste slobodno ga **obrišite** iz hijararhije. Sada je ostalo samo da ovaj objekat iz foldera dodelimo skripti **spaceship.cs** koja se nalazi nakačena na **Xwing_0** tj naš brod kao na slici. Takođe možete da prevučete i Xwing_0 u Prefab folder ili bilo koji drugi objekat ali ne morate pošto je on jedini objekat svoje vrste na sceni.




Sada kada pokrenemo igricu videćemo da naš brodić ispaljuje metke pristiskom na **space.** Sa leve strane u **Hiarachy** prozoru vidimo sve klonove koje kreira space i to je ime koje smo trebali da damo u prethodnoj skripti kada proveravamo da li je došlo do kolizije sa metkom.



# 5.1. Pucanje neprijateljskih brodića




Nakon što smo omogućili našem brodiću da može da puca treba da omogućimo i našim neprijateljskim brodićima da se brane odnosno da pucaju na naš brodić. Razlika je što njima ne upravljamo mi već "računar" pa ćemo im dodeliti neki brojač koji će na svakih 100 frejmova ispaljivati metak.

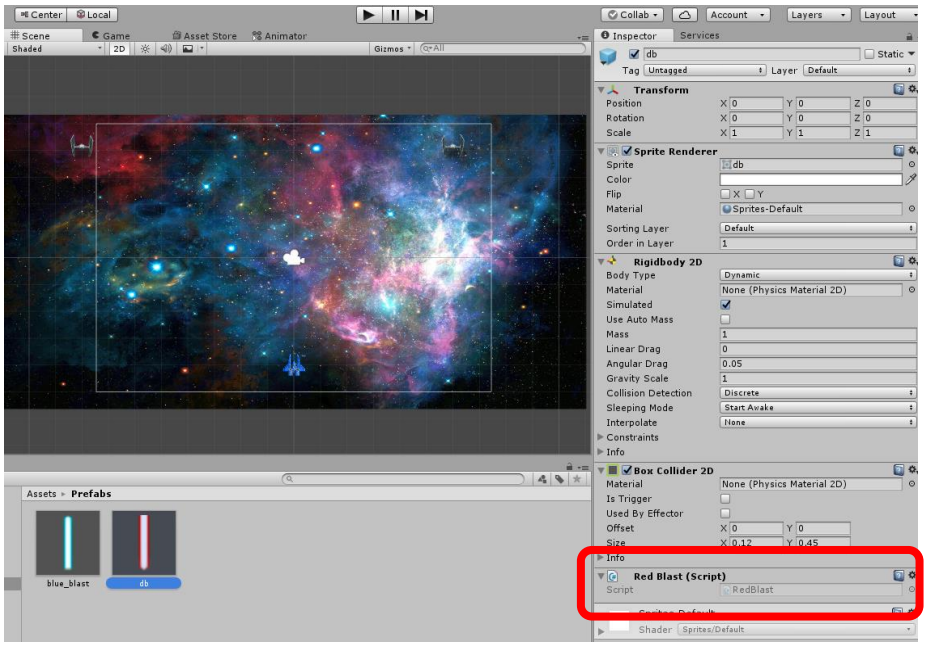
Na isti način dodajemo iz foldera **Assets/Sprites/db** sprajt u Hijararhiju i na sličan način dodeljujemo komponente jedina razlika je što sada ne želimo da u komponenti **Rigidbody2D** masa i gravitacija budu **0** već da ostanu kako je definisano kao **1,** jer treba da padaju na dole kada se ispale.

Isto sada prevlačimo i objekat **db** iz hijararhije u folder **Prefabs** . Kada objekat postane **prefab** on dobija plavu boju u editoru čime se označava da se radi o prefab objektu. Sada ga možemo obrisati iz hijararhije jer nam više nije potreban na sceni već samo kada ga ispaljuju protivnici.

Napisaćemo skriptu i za crveni pucanj i dodeliti je **prefab**-u **db na kraju** prevlačenjem kao što smo dodali I za plavi metak tako da kreiramo **C#** skriptu u folderu **Assets/Scripts/RedBlast.cs** na sledeći način:

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

public class RedBlast : MonoBehaviour
{
    // ovde metak samo treba da registruje kolizju i da
    //unistava sebe kada dodje van main kamere
    void OnBecameInvisible()
    {
        enabled = false;
        Destroy(gameObject);
    }
```

```
    void OnCollisionEnter2D(Collision2D col)
    {
        if (col.gameObject.name == "Xwing_0")
        {
            Destroy(gameObject); // ne unistava drugi objekat vec sebe
        }
    }
}
```

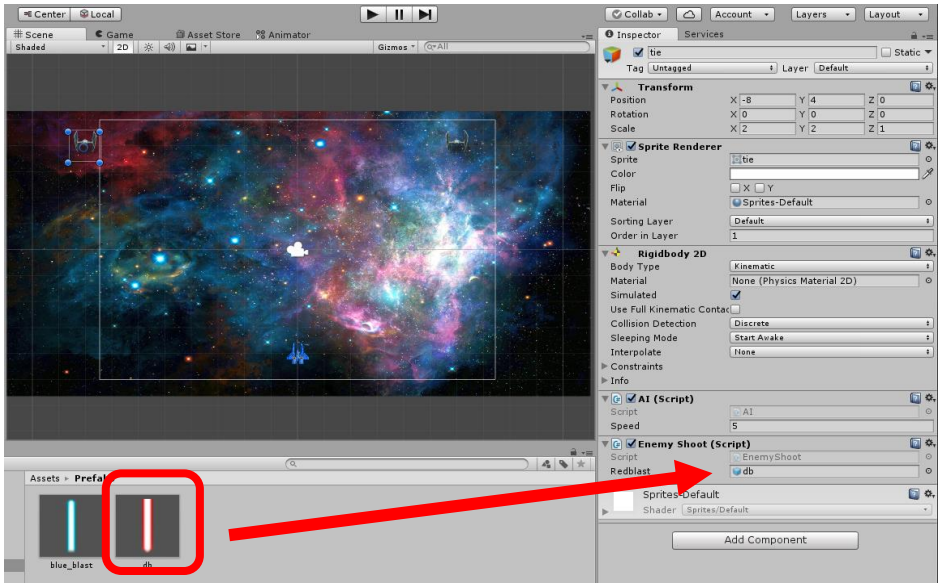



Sada treba da dodelimo neprijateljima ovaj metak i da podesimo vreme njihovog ispaljivanja. Zato ćemo kreirati C# skriptu u folderu **Assets/Scripts/EnemyShoot.cs** I modifikovati je na sledeći način:

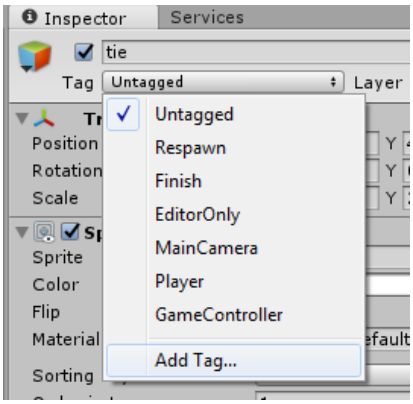
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyShoot : MonoBehaviour
{
    public GameObject redblast;
    private int time = 0;

     // Update is called once per frame
     void Update ()
    {
        time += 1; //pucaj na svakih 100 frejmova

        if (time > 100) //ako je preslo 100 vrati time na 0
        {
            time = 0;
            Instantiate(redblast, transform.position, Quaternion.identity);
                        // ovo je za kreiranje vise objekata koji
```

```
            // prosledjujemo kao 1 argument sa odredjene pozicije sto je drugi
            // argument dok treci argument sluzi da ne dodje do rotacije vec
            // da objekat prati pravac roditelja odnosno neprijatelja
        }
    }
}
```

Sada ovu skriptu dodelimo našim neprijateljima **tie** i **tie(1)** iz hijararhije. I prevučemo **db prefab metak** iz foldera **Prefabs** u polje kao na slici.

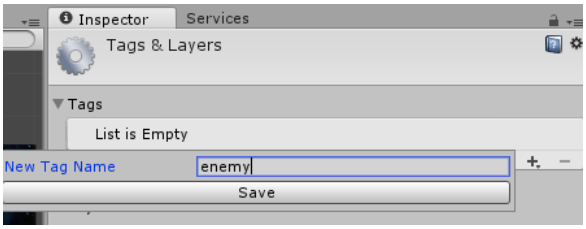



## 5.2. Tagovanje i uništavanje brodova




Sada još treba da namestimo da se brodići međusobno uništavaju. Mi smo namestili malopre da uništava metak ali ne i neprijateljske brodove. Zato je potrebno ovde uvesti pojam **TAGOVA**.

**Tagovanje** se koristi kada želimo da neke osobine/svojstva/akcije dodelimo na više objekata u našoj igri. Lakše je navesti tag i onda u skriptama samo pozivati sve objekte koji su grupisani sa istim imenom taga nego ići preko imena pojedinačno. Pristup preko imena je lakši kada je igrica jednostavna ali što su igrice složenije to postaje dosta konfuzno i teško za pratiti.




Zato se kreira **tag** kao na slici i mi ćemo ga imenovati kao **enemy**. Po kreiranju taga dodelite tag **enemy** svim neprijateljskim brodovima. Tako će svi neprijatelji imati taj tag a onda ćemo modifikovati samo skriptu iz foldera **Assets/scripts/BlueBlast.cs** i dodeliti još jedan uslov u metodi **OnCollisionEnter2D()** ali da bi došlo do kolizije ovaj metod će se pozvati samo ako svi objekti koji učestvuju u sudaru imaju **BoxCollider2D komponentu,** pa dodelimo i ovo na neprijateljske brodove **tie** i **tie(1)** u **Inspector** prozoru**.**

```
void OnCollisionEnter2D(Collision2D col)
{
        // u ovom ifu pitamo da se  objekat sa kojim je doslo do kolizije razlikuje od
        //naseg broad I kopije svakog metka koji je nas brod ispalio (ime ovo dobijate iz
        //hijararhije kada pokrente igricu i kliknete space dugme ispisivace vam naziv)
        if (col.gameObject.name != "Xwing_0" && col.gameObject.name !=
"blue_blast(Clone)" && col.gameObject.tag == "enemy")
        {
            Destroy(col.gameObject);
        }

}
```

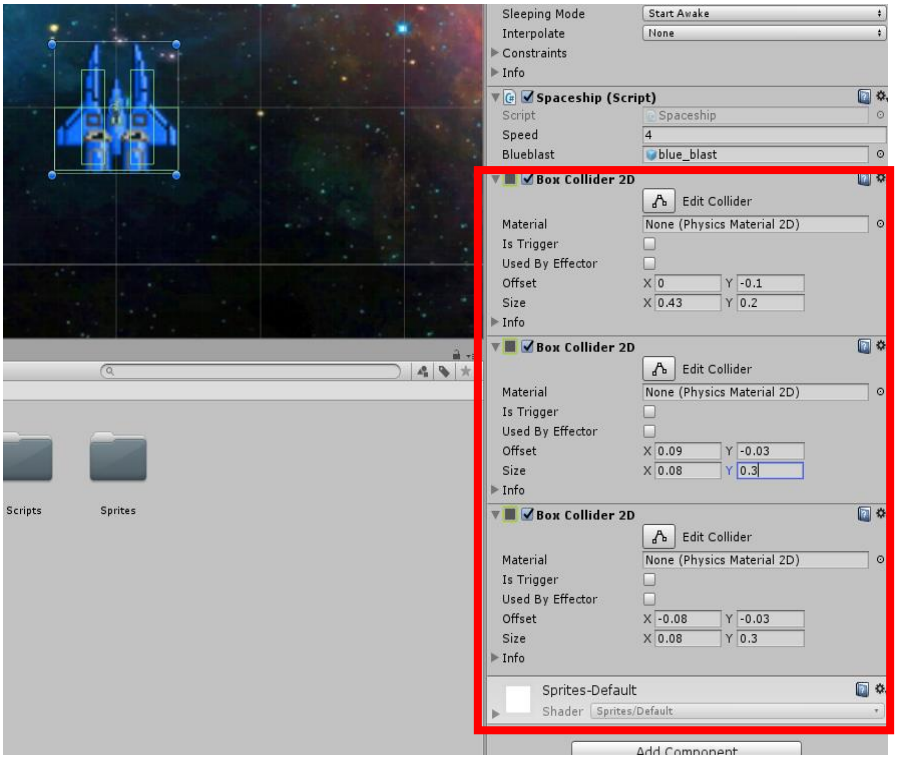
Na isti način ćemo omogućiti i uništavanje našeg brodića kada je pogođen neprijateljskim metkom s razlikom što **ne moramo** davati tag (ali možete ako želite) našem igraču, jer je samo jedan u igrici kao glavni protogonista svemira. Kada se on uništi **ZA SADA** će učitati scenu **Meny** pošto je **Game Over** u toj situaciji. Dakle skriptu iz foldera **Assets/Scripts/Spaceship.cs** modifikovati na sledeći način:

```
    void OnCollisionEnter2D(Collision2D col)
    {
        //kada dodje u sudar sa drugim objektom tj metkom ili brodom
        if (col.gameObject.name != "blue_blast(Clone)")
        {
            SceneManager.LoadScene("Meny");
        }
    }
```

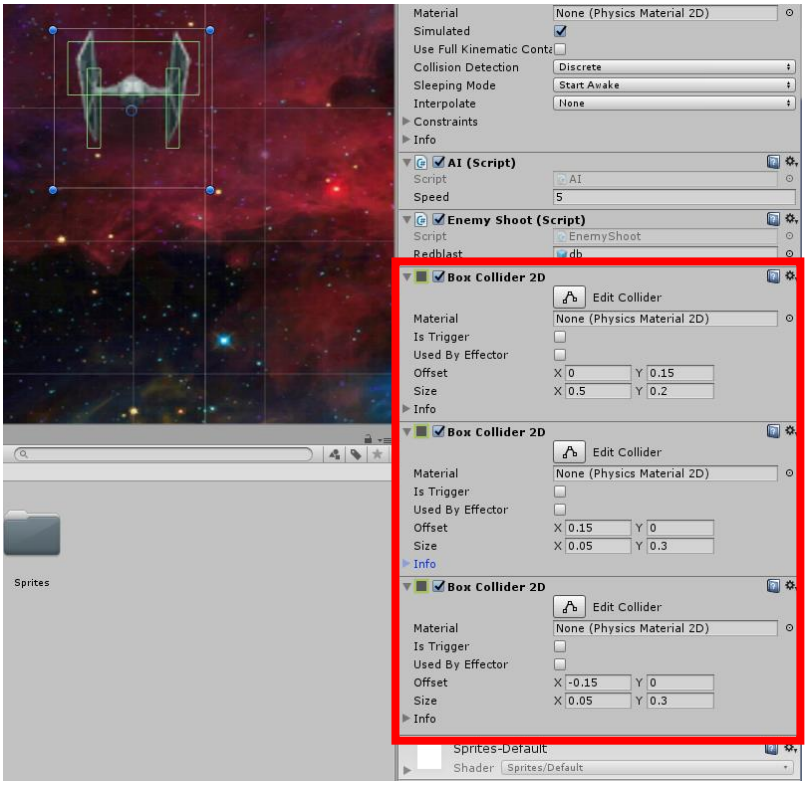
Isto kao i za neprijateljske brodove dodelimo našem **brodu Xwing_0 BoxCollider2D** komponentu i pokrenite da vidimo krajnji rezultat.

**NAPOMENA**: Prilikom dodavanja **BoxCollider2D**-a objektima vodite računa jer se metak koji ispaljuje svaki brod pojedinačno krivi pošto čim se ispali sudari se BoxCollider2D-om i poremeti mu se rotacija. Ovo možete da rešite ako dodate **više** ovih komponenti objektu i napravite precizniji oblik oko broda **Xwing_0** ali tako da proverite odakle Vam kreće **blue_blast** i taj deo da ne označite kao potencijalno

mesto za sudar. Pogledajte sliku ispod i isto napravite i za **neprijateljske brodove**. Da podsetimo **zelena** boja na sprajtu predstavlja BoxCollider2D.




Svaki objekat na sceni može imati i **više** ovih **BoxCollider2D** komponenti kao što je ovde primer ako je to potrebno za detaljnije obeležavanje koji delovi sprajta mogu doći u koliziju. U **inspector** prozoru im možete menjati poziciju kao i dimenziju.



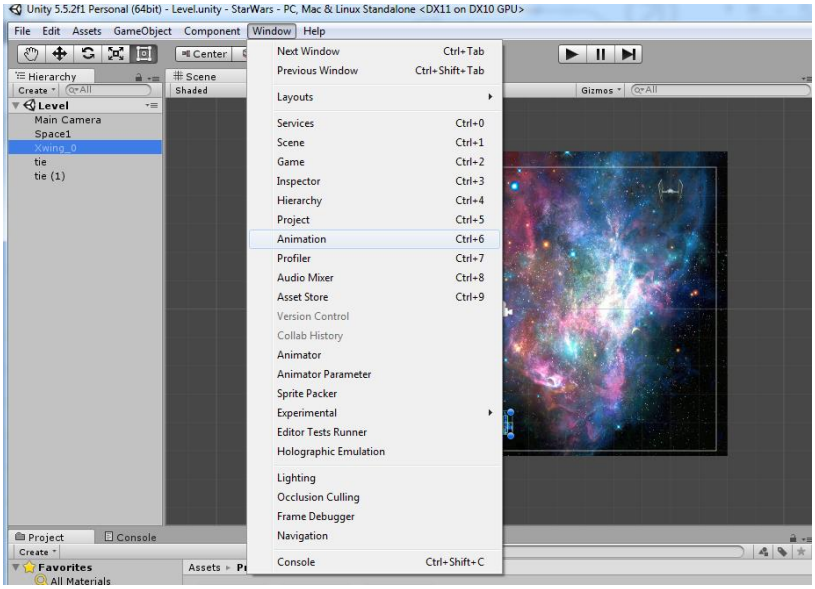
# 6. Animacije igračevog broda

Naš brod trenutno postoji u svemiru I kreće se ali potrebno je da mu omogućimo što realniji prikaz I da ne bude stalno statičan. Zato ćemo kreirati animacije koje će to realizovati I to za:

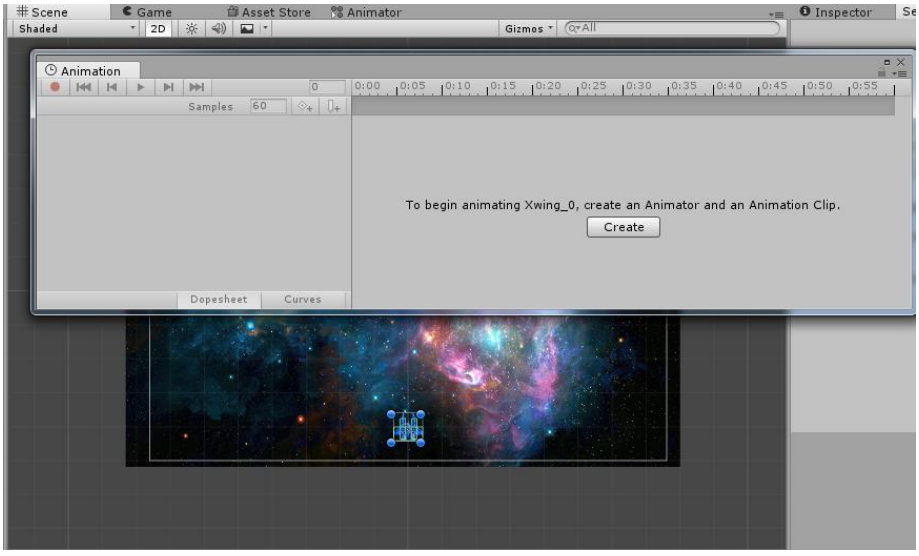
- **POČETNO STANJE,**
- **ROTIRANJE,**
- **EKSPLOZIJU**

Prvo treba da odaberemo objekat nad kojim želimo da kreiramo animaciju, u našem slučaju na **Xwing_0** a zatim odaberemo iz gornjeg toolbar-a

***Window -> Animation***

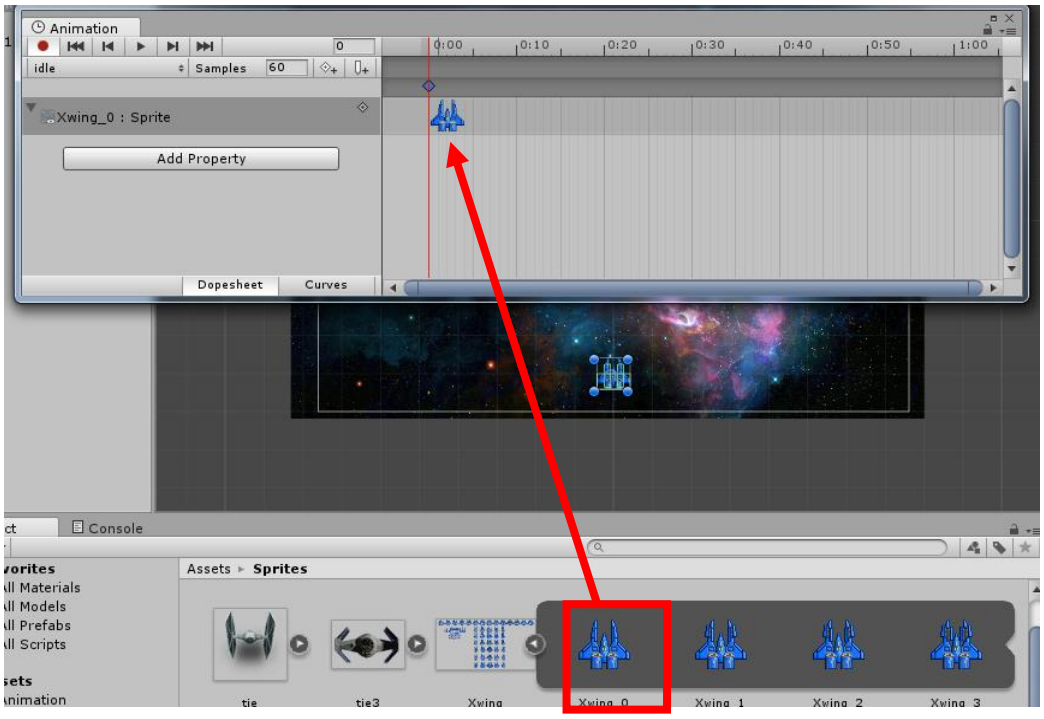



Prozor za kreiranje klipova animacije će izgledati ovako:

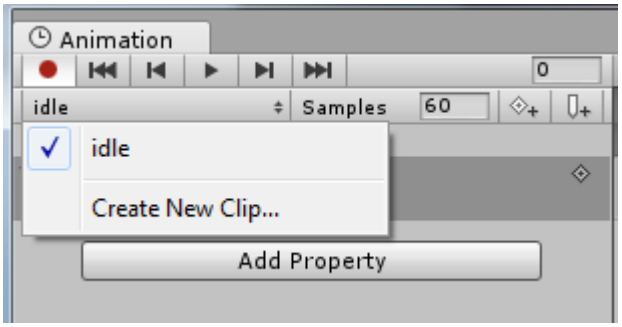



Za **početno stanje** nam ne treba ništa specijalno , **uzećemo isti sprajt** kao za naš brod kako bi taj sprajt predstavljao prvobitno stanje pri pokretanju igrice, za njega ne želimo da se niti pomera niti rotira. Odabirom opcije **Create** dodajte novi klip u nov folder **Assets/Animation/** i nazovite klip **idle.anim** Ova ekstenzija služi da označi da je u pitanju animacija i svi klipovi za animaciju će je imati pri čuvanju.

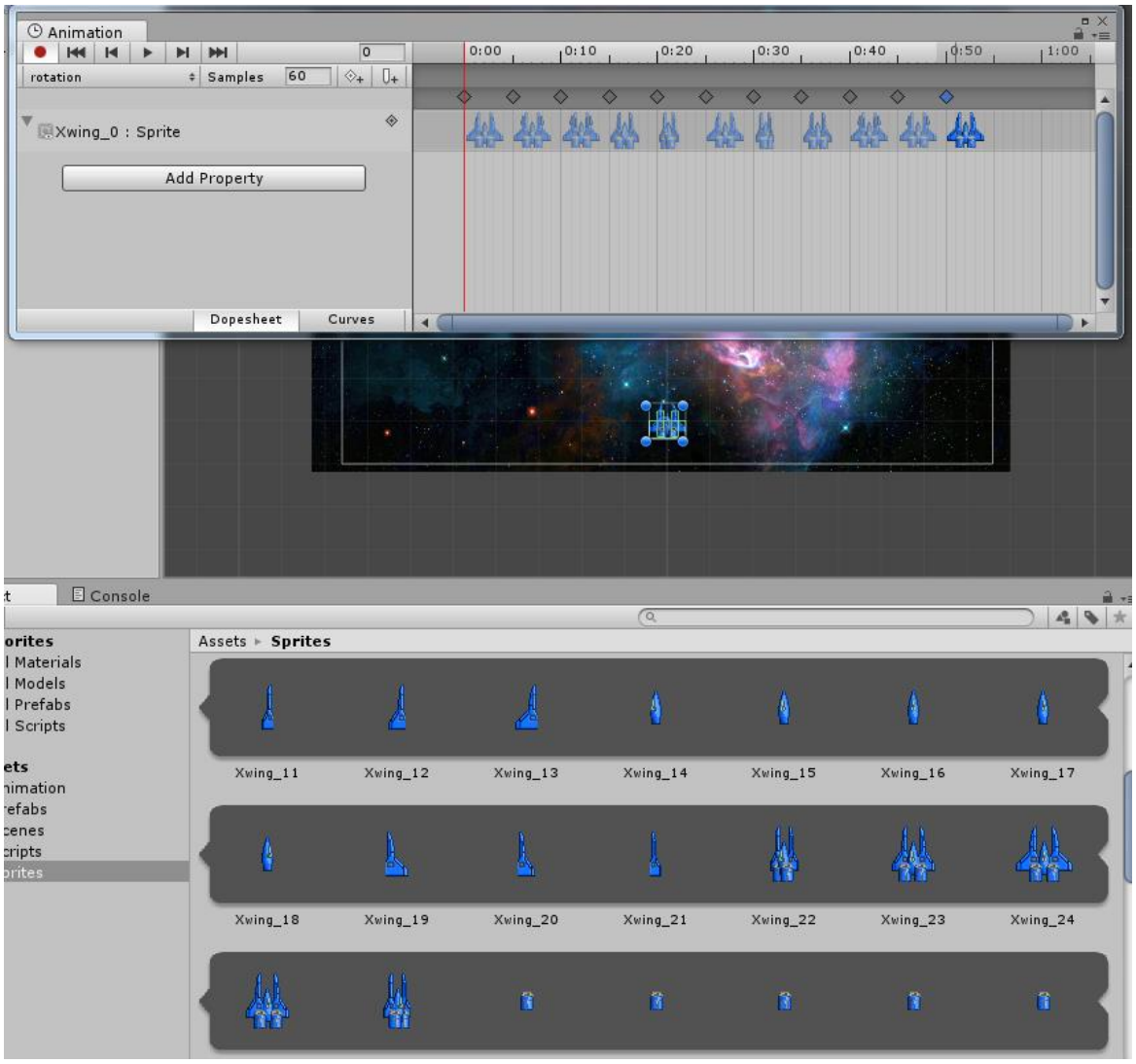
Nakon čuvanja animacije/klipa pojaviće se u Unity-ju sledeći prozor i na njega iz foldera **Assets/Sprites/** prevučemo **Xwing_0.**




Ovo će biti sve za ovaj klip. Sada za rotaciju i eksploziju ćemo nešto složenije da napravimo jer ipak su oni sačinjeni od većeg broja sprajtova. Na isti način kreiramo novi klip ali sada pošto postoji već jedan klip biramo opciju **Create New Clip**:

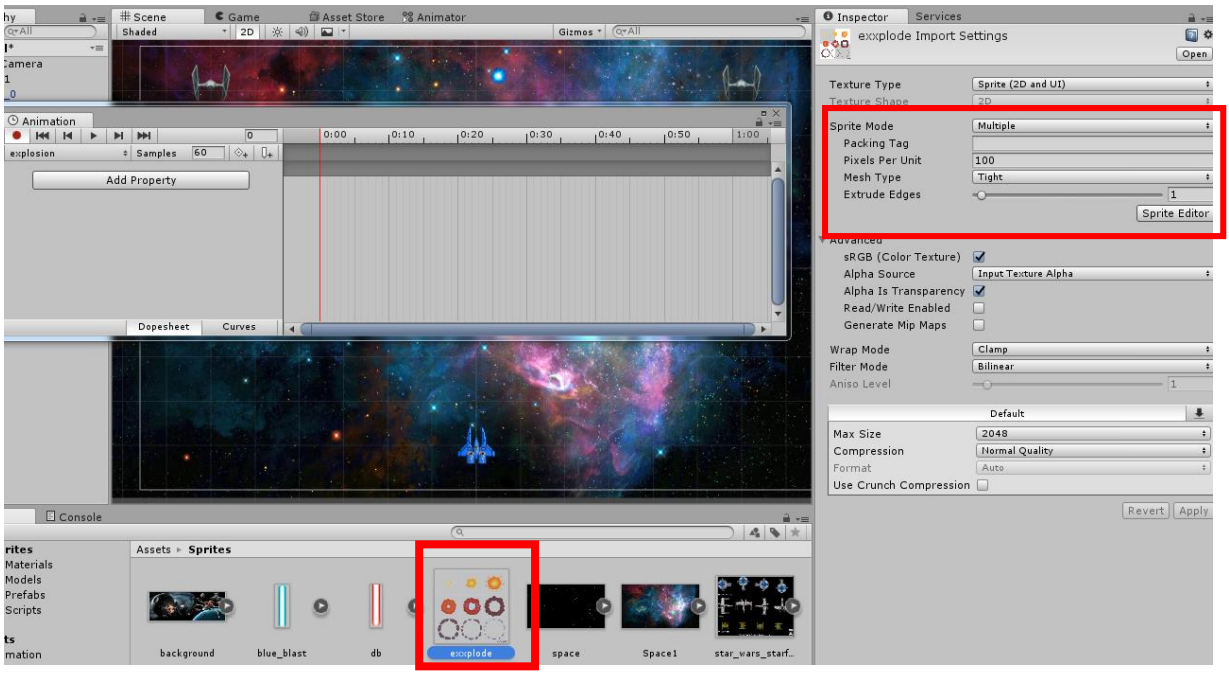


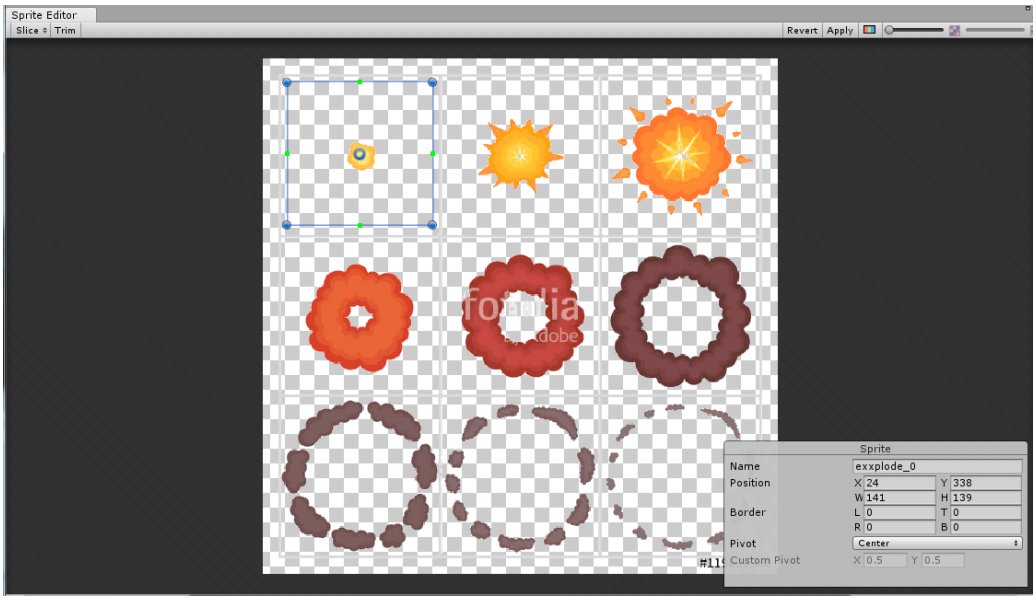

Nazovite novu animaciju **rotation.anim** i sačuvajte isto u folderu Animation. Sada na Vama je kako želite da rotacija izgleda ali mi ćemo iskoristiti sprajtove koje imamo I kada ubacujete sprajtove za rotaciju gledajte da imaju slične logičke sprajtove koji bi sledili u situacijama kada bi došlo do rotacije, dakle treba Vam broj koji se okreće. Mi smo izabrali sprajtove redom**: Xwing_0, Xwing_1, Xwing_2, Xwing_23, Xwing_22, Xwing_24, Xwing_26, Xwing_25, Xwing_2, Xwing_1, Xwing_0** kao sa slike ispod.



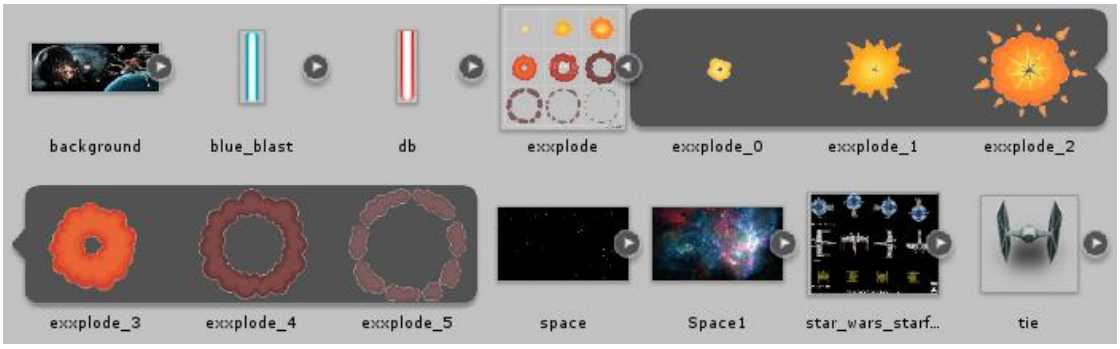

Podesićemo polje "**Samples**" kao broj koji predstavlja vreme okretanja pojedinačnih sprajtova pa samim tim predstavlja i brzinu čime usporava odnosno ubrzava i celu animaciju. Ovde ćemo podesiti da bude **50**.

I poslednje je animacija za eksploziju, nju dodajete i čuvate na isti način kao prethodne dve (**explosion.anim**) samo dodajemo druge sprajtove. U prozor **Animation** prevlačimo sprajtove iz sprajta **Assets/Sprites/explode** koje prvo moramo da isečemo kao što smo radili u poglavlju **4**. Kada smo kreirali sprajt za naš glavni brod. U inspector prozoru ovog sprajta odaberite **Sprite mode – Multiple**. Ako imate problema sa sečenjem **Multiple** sprajtova jer Unity nekad ne može automatski da prepozna granice za sečenje onda sami mišem iscrtajte sprajtove veličine koje vi želite. Nakon toga samo idite na **Apply** bez kliktanja Slice dugmeta kao što je pokazano na slikama ispod.

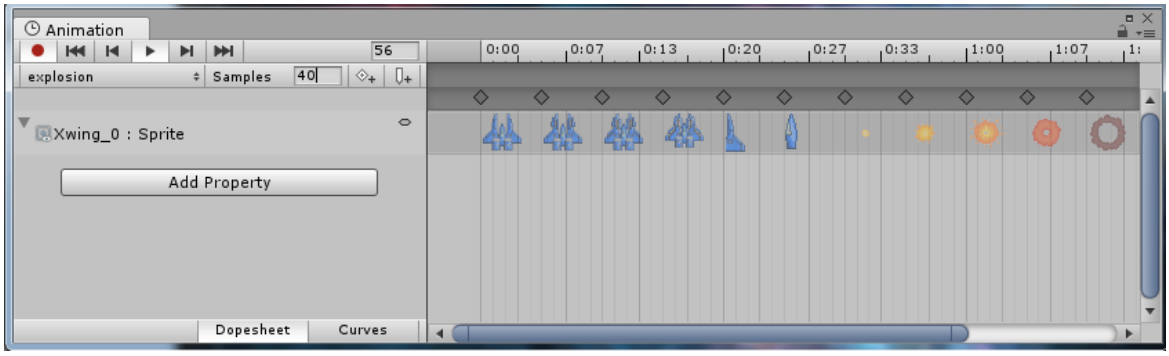



Dobićete sprajtove kao na slici ispod:

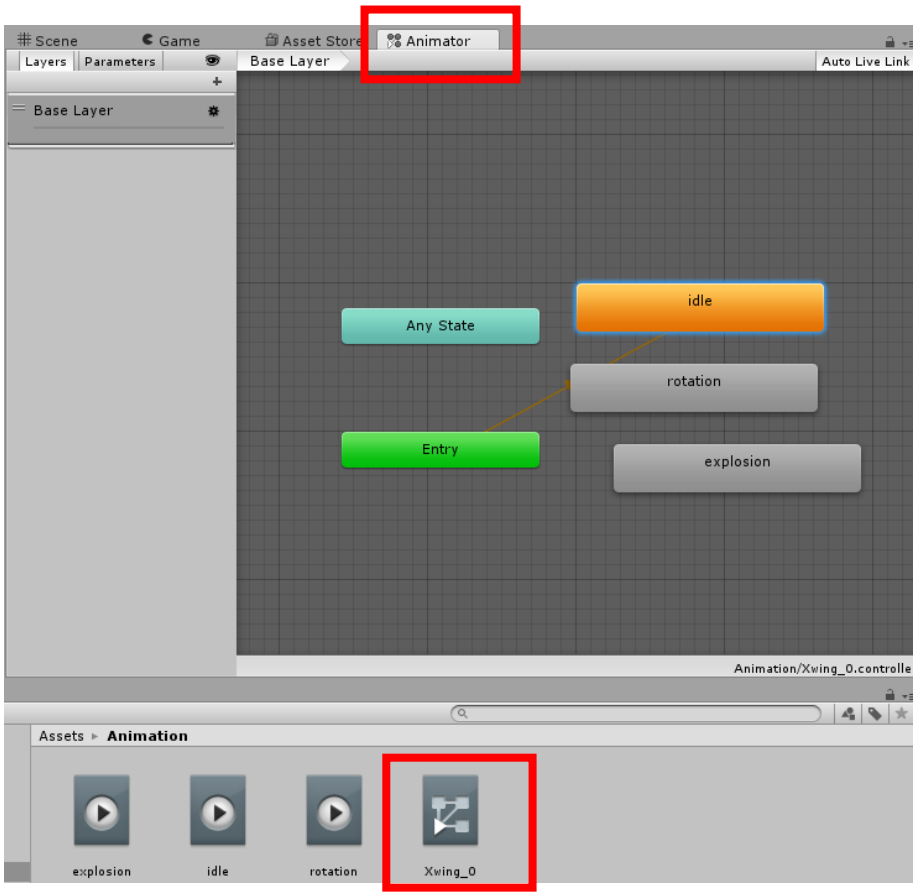


Sada je još ostalo da ih prevučemo u **Animation** window:

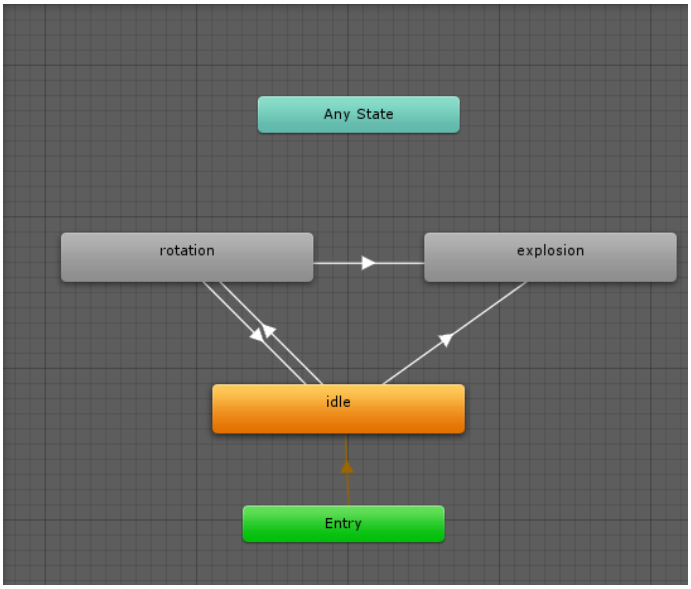



## 6.1. Mašina stanja - Kontroler

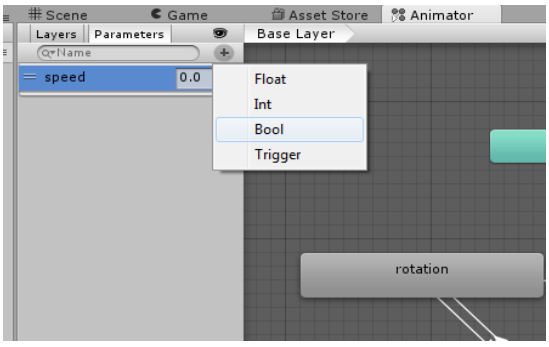
Još jedan pojam koji ovde treba da uvedemo jeste **mašina stanja** odnosno **kontroler** koji upravlja sa animacijama/klipovima. Preko njega mi određujemo kada će i iz kog stanja , naš objekat nad kojim smo kreirali animacije, preći. On kada kreirate klipove u pozadini **sam dodaje** te klipove u animator i kreira kontroler. Kada odaberete prozor kao sa slike dobićete nešto ovako:



Zašto je jedno stanje **narandžasto**? Pa odgovor je vrlo jednostavan. To je početno stanje koje će biti **default-no stanje** pri pokretanju igrice iz kog će da brod prelazi u druga stanja tj animacije. Vi možete da promenite ako ne želite stanje koje Vam je postavio **Unity** na **desni** klik na **sivo** polje i kliknete opciju **Set As Default State.**

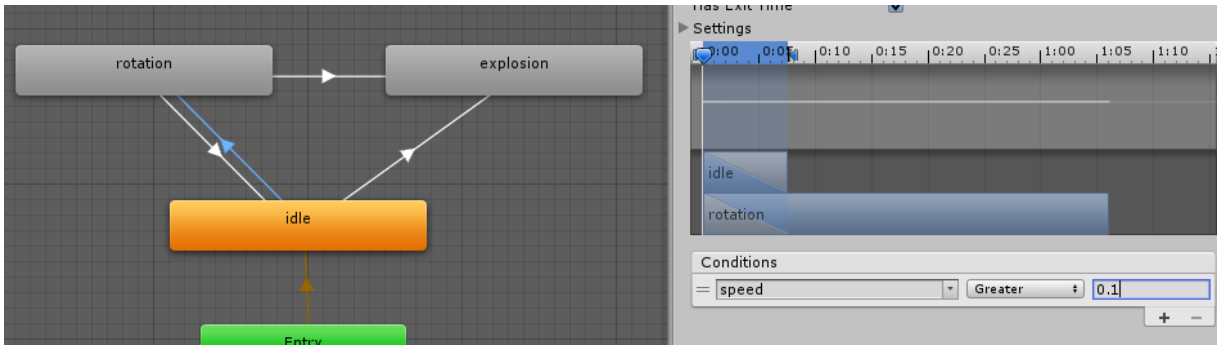



Sada treba da povežemo stanja i da kažemo kontroleru kada želimo da naš brod pređe iz jedne animacije u drugu. Povežite stanja kao na slici ispod tako što **desni** klik miša na neko od stanja a zatim opciju **Make Transition**:

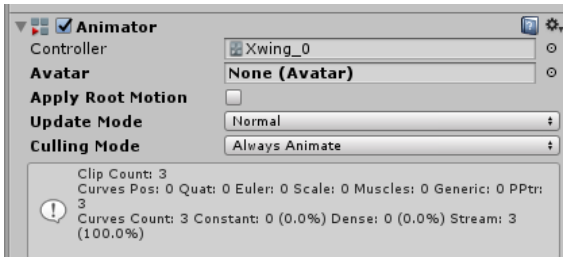



**Da sumiramo**: Početno stanje u našem slučaju ostaje **idle** zatim želimo da iz tog stanja brod može na levu ili desnu stranu da se rotira dakle moćiće da pređe iz stanja idle u stanje **rotation** i obrnuto. U stanje **explosion** će preći kada ga pogodi metak a to može da bude ili iz idle ili iz rotation stanja ali iz eksplozije ne može da se vrati u prethodno stanje tu se igra završava. Ostalo je još da definišemo dva parametra **speed** tipa **float** i **die** tipa **bool** koji će predstavljati promenljive koje će omogućiti stanjima da pređu iz jednog stanja u drugo pošto to još uvek ne možemo da uradimo iako smo stanja samo povezali. **Die ostavite da bude nečekirano (false)**

Nakon setovanja parametara klikom na strelice u Animatoru dodelite sad ove parametre kako bi mašina stanja znala šta treba da radi kad prelazi u drugo stanje.



Kada prelazi **iz idle->rotation** onda speed se **uvećava za 0.1** (Greater) kada prelazi iz **rotation->idle** onda se **smanjuje za 0.1** (Less). Kada prelazi iz bilo kog stanja u stanje **explosion** onda samo die treba da postane **true**. Editor zna nad kojim sprajtom kreirate animacije i sam će dodeliti komponentu **Animator** u **Inspector** prozoru kao što je i ovde slučaj sa **Xwing_0**, naravno komponeta sadrži **kontroler** koji smo definisali kao mašinu stanja i koju smo prethodno kreirali tako da je sad vezana za objekat našeg broda. Samo još treba da skripti **Spaceship.cs** dodelimo animator kako bi znao šta da radi sa njim tako da modifikujte skriptu na sledeći način:




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

public class Spaceship : MonoBehaviour {
    public float speed = 4.0f;
    private Rigidbody2D rb;
    private int time = 0;
    public GameObject blueblast;
    private Animator anim;

    void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
    }
    // Use this for initialization
    void Start () {
        anim.SetBool("die",false);
      }

    void Update ()
    {
        if (Input.GetKey(KeyCode.Escape))
        {
            SceneManager.LoadScene("Meny");
        }
            if (Input.GetKey(KeyCode.LeftArrow) &&
            transform.localPosition.x > -7)
        {
            transform.position = transform.position +
                Vector3.left * speed * Time.deltaTime;
            anim.Play("rotation");
        }
        if (Input.GetKey(KeyCode.RightArrow) &&
            transform.localPosition.x < 7)
        {
            transform.position = transform.position +
                Vector3.right * speed * Time.deltaTime;
            anim.Play("rotation");
```

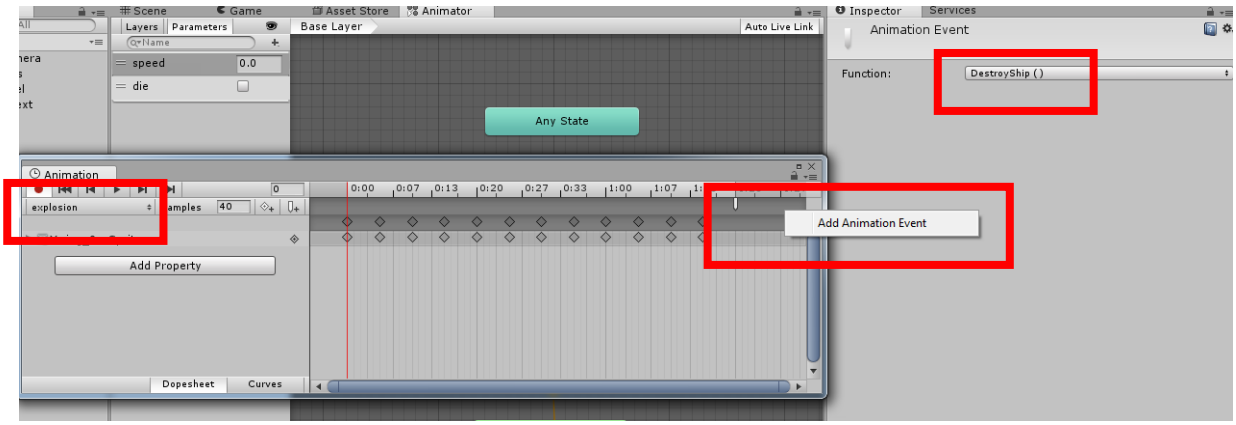
```
            }
            if (Input.GetKey(KeyCode.UpArrow) &&
                transform.localPosition.y < 6)
            {
                transform.position = transform.position +
                    Vector3.up *
                    speed * Time.deltaTime;
            }
            if (Input.GetKey(KeyCode.DownArrow) &&
                transform.localPosition.y > -4)
            {
                transform.position = transform.position +
                    Vector3.down * speed * Time.deltaTime;
            }

            if (Input.GetKeyDown(KeyCode.Space))
            {
                      Instantiate(blueblast, transform.position,
      Quaternion.identity);
            }
        }

        void OnCollisionEnter2D(Collision2D col)
        {
            //kada dodje u sudar sa drugim objektom tj metkom ili brodom
            if (col.gameObject.name != "blue_blast(Clone)")
            {
                anim.SetBool("die", true);
                anim.Play("explosion"); //umesto prebacivanja na meni scenu
            }
        }
    }
```

## 6.2. Dodavanje skripte za uništavanje animacije

Sada kada je kreirana animacija za eksploziju treba da namestimo da po završetku animacije za eksploziju nas odvede na scenu Meny gde možemo da biramo dal želimo novu partiju. Da bi se to uradilo kreiramo skriptu u folderu assets/Scripts/Destroy.cs koja će biti vrlo jednostavna :

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

public class Destroy : MonoBehaviour
 {

    void DestroyShip()
    {
        Destroy(gameObject);
        SceneManager.LoadScene("Meny");
    }
}
```

Ovu skriptu dodate našem brodu **Xwing_0** a zatim otvorite animaciju **explosion** i na kraj svih sprajtova animacije dodelite **Event** (događaj) kao sa slike koji će se pokrenuti kada se završe svi sprajtovi u animaciji. Desni klik zatim **Add animation Event** i u **Inspector** prozoru odaberete iz padajuće liste metodu koju smo dali skripti **Destroy.cs -> DestroyShip().** Sada će se igrica prekinuti po uništavanju broda.

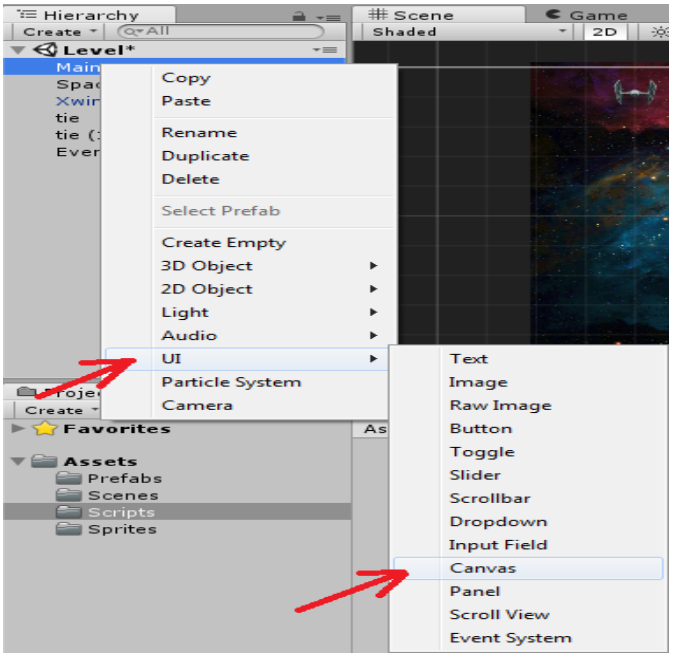


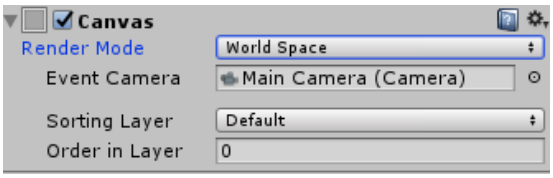

## 7. Dodavanje poena – Score

Sada je potrebno dodati **score**, koji se uvećava, samo ako nas brodić ubije protivnika. Polje za score ćemo postaviti odmah pored pozadine. Prvo smanjujemo širinu pozadine sa desne strane. Pošto smo smanjili pozadinu moramo da smanjimo i granice do kojih se mogu kretati igrač i neprijatelji. To se podešava u okviru već kreiranih skripti ***Spaceship.cs*** i ***AI.cs*** i to u sledećim delovima koda:

```
if (Input.GetKey(KeyCode.RightArrow) &&
    transform.localPosition.x < 7)
{
    transform.position = transform.position +
        Vector3.right * speed * Time.deltaTime;
}
```

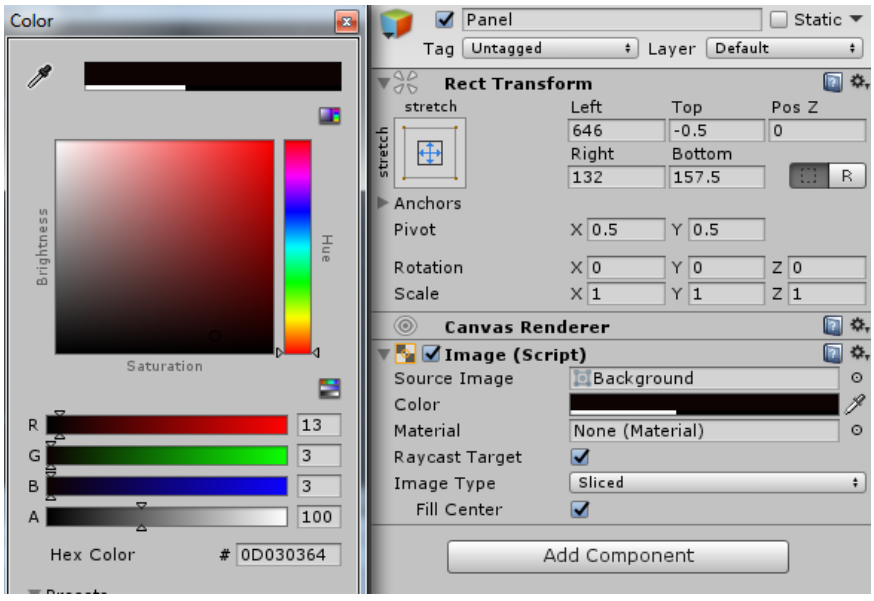
Sledeći korak jeste kreiranje polja za score. U **Hiararchy** panel-u u desnim klikom na ***Main Camera*** kreiramo ***Canvas***, kao na slici ispod:



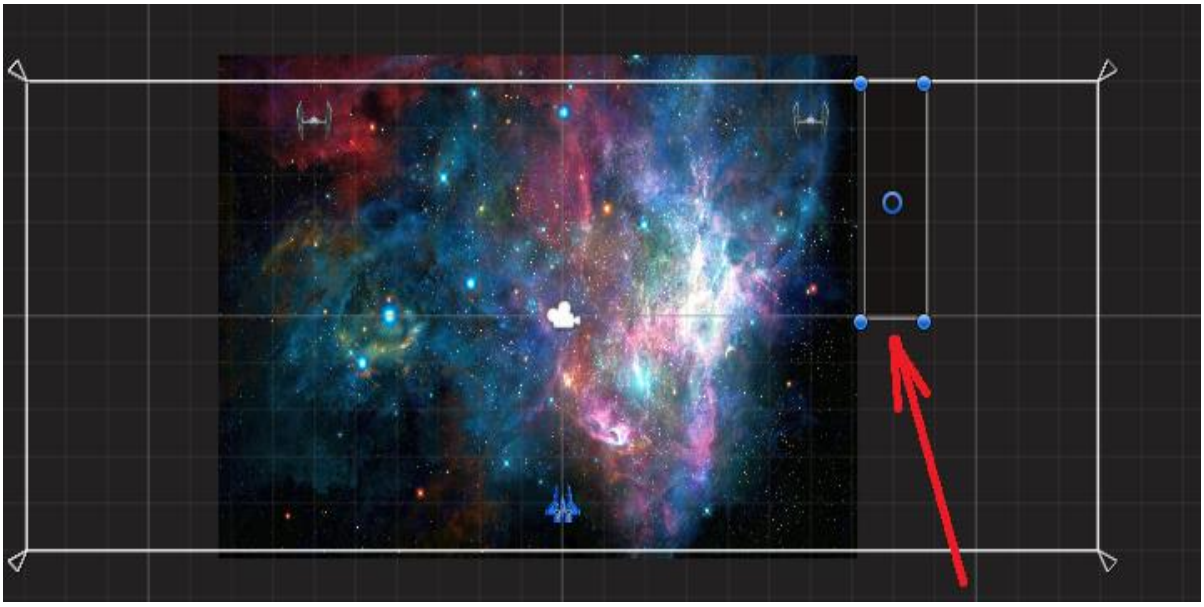




Klikom na **Canvas** u **Hiararchy** panel-u otvaraju nam se podešavanja za isti u okviru **Inspector** panela-a sa desne strane. Tu vršimo sledeća podešavanja u okviru Canvas komponente. Za ***Render Mode*** odaberemo ***World Space,*** u polje *Event Camera* prevučemo *Main Cameru*. Ako Vam ne uspe postavite na **Screen Space – Camera** pa vratite na World Space jer se Unity nekad buni.

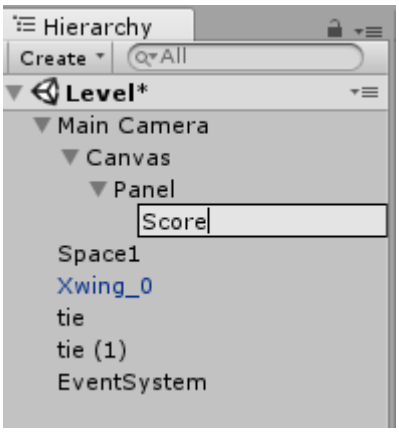
U okviru ***Canvas***-a u Hiararchy panelu kreiramo **Panel** komponentu (desni klik na ***Canvas -> UI->Panel***). Panel komponentu podesimo pored gornjeg desnog ugla pozadine. Sada u okviru ***Image****(Script)* komponente u Inspector prozoru postavimo boju panela na crnu.

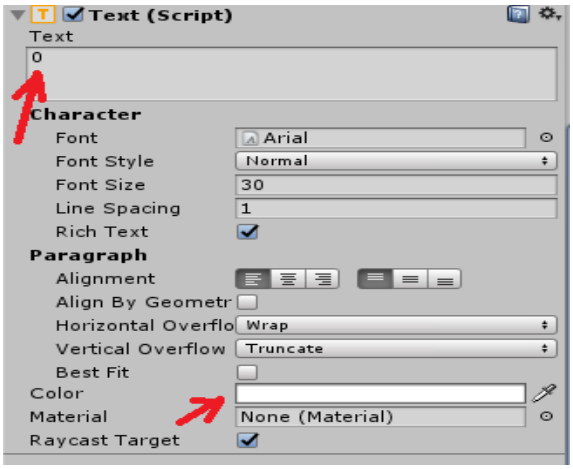


Panel pozicioniramo kao na sledećoj slici:



U kreirani ***Panel*** dodajemo dva text polja. Prvo **text** polje koristimo za prikazivanje **scora**, a drugo **levela**. Desni klik na Panel, klik na **UI**, pa klik na **Text**, kreira se Text polje, kome menjamo naziv u *Score*. Text polje pozicioniramo u okviru Panela, a u Inspector prozoru vršimo sledeća podešavanja. U komponenti *Text(Script)* , boju postavimo na belu, a u polje Text unesemo **0**.

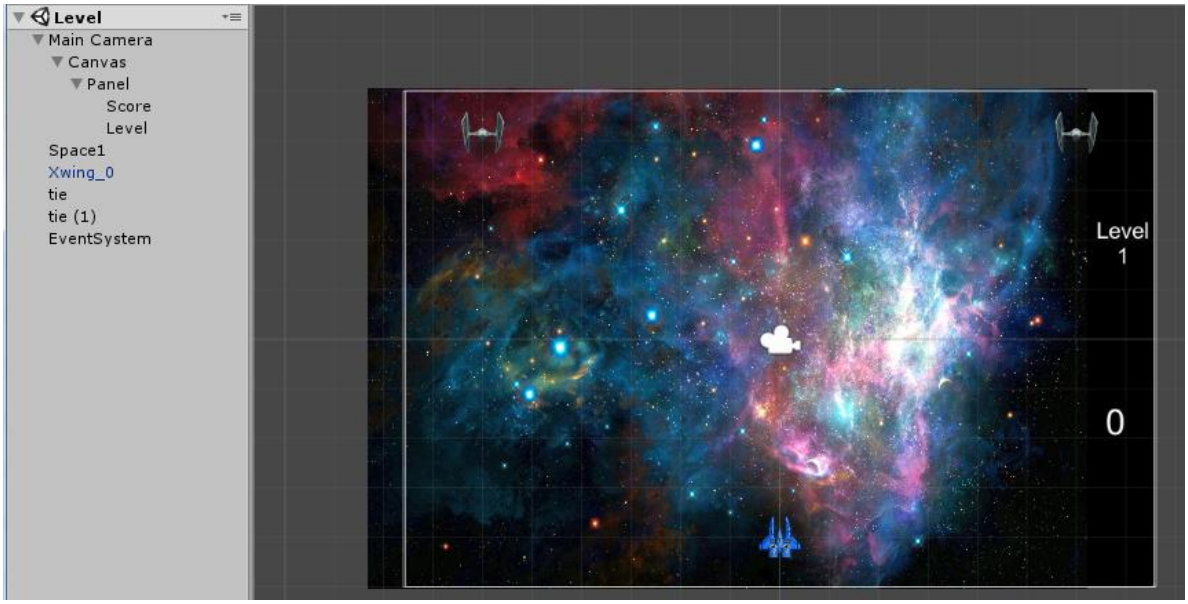



Kako bi score radio, modifikujemo skriptu ***BlueBlast.cs***. Na početku definišemo promenljivu koja predstavlja score. Dodajemo metod **Awake(),** i dopunjujemo metod **OnCollisionEnter2D**. Skripta nakon dopune ima sledeći sadržaj:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class BlueBlast : MonoBehaviour
{
    public float speed = 10f; // bryina kretanja metka
    private Text score; //definisimo promeljivu koja predstavlja score
    private void Awake()
    {
        //pristupamo Text polju koje smo kreirali u Unity-u i nazvali ga Score
        score = GameObject.Find("Score").GetComponent<Text>();
    }

    // Update is called once per frame
    void Update()
    {
                    transform.position += Vector3.up * speed * Time.deltaTime;
                   //pozicija metka i gde da se krece
    }

    void OnBecameInvisible() // kada metak ode van main kamere
    {
        enabled = false; //ugradjena bool promenljiva koja postaje false posto dok se
metak vidi ona je true
        Destroy(gameObject);//unisti objekat tj metak
    }

    // sta ce se desiti kada dodje u koliziju metak sa drugim objektom
    void OnCollisionEnter2D(Collision2D col)
    {

        if (col.gameObject.name != "Xwing_0" && col.gameObject.name !=
"blue_blast(Clone)" && col.gameObject.tag == "enemy")
        {
            //zabelezi rezultat
            string t = score.text;
            int x = System.Int32.Parse(t) + 1;
            t = x.ToString();
            score.text = t;
            Destroy(col.gameObject);
        }
      else if (col.gameObject.name == "db(Clone)") Destroy(gameObject); // ovaj
      //if se dodaje kako ne bi povecavao score i kada pogodi
       //neprijateljski metak, tada ga samo unisti
    }
}
```


Preostaje nam da u okviru Panela napavimo jos jedno Text polje koje će da prikazuje nivo. Nazovemo ga *Level*. I u okviru *Text polja Text(Script)* komponente unesemo broj 1.




## 8. Kraj igre - Pobeda

Ako je igrač tj naš brod ubio sve neprijatelje treba da ga igra vrati na scenu **Meny** gde on može da bira šta želi dalje od akcija da uradi. To se radi tako što definišemo niz **enemys** u skripti **BlueBlast.cs** i modifikujemo kod na sledeći način, deo koda koji nije naveden ostaje nepromenjen:

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

public class BlueBlast : MonoBehaviour
{
    public float speed = 10f; // bryina kretanja metka
    private Text score;
    private GameObject[] enemys;

     .
     .
     .

     void Update () {
        transform.position += Vector3.up * speed * Time.deltaTime;
        enemys = GameObject.FindGameObjectsWithTag("enemy");
        if (enemys == null || enemys.Length == 0)
        {
            SceneManager.LoadScene("Meny");
        }
     }
}
```

I sada imate uvid u osnove kreiranja igara. Sve što Vam treba su par početnih koraka koji će Vam pokazati koliko **Unity** može biti interesantan i kreativan način za ispoljavanje vaše mašte samo ako to volite i želite. Nadamo se da Vam je ovaj tutorijal malo pomogao da shvatite na koji način Unity funkcioniše i kako se kreiraju 2D igrice preko njega. Nama je bilo zabavno a nadamo se i Vama kroz ovo putovanje svemirom.