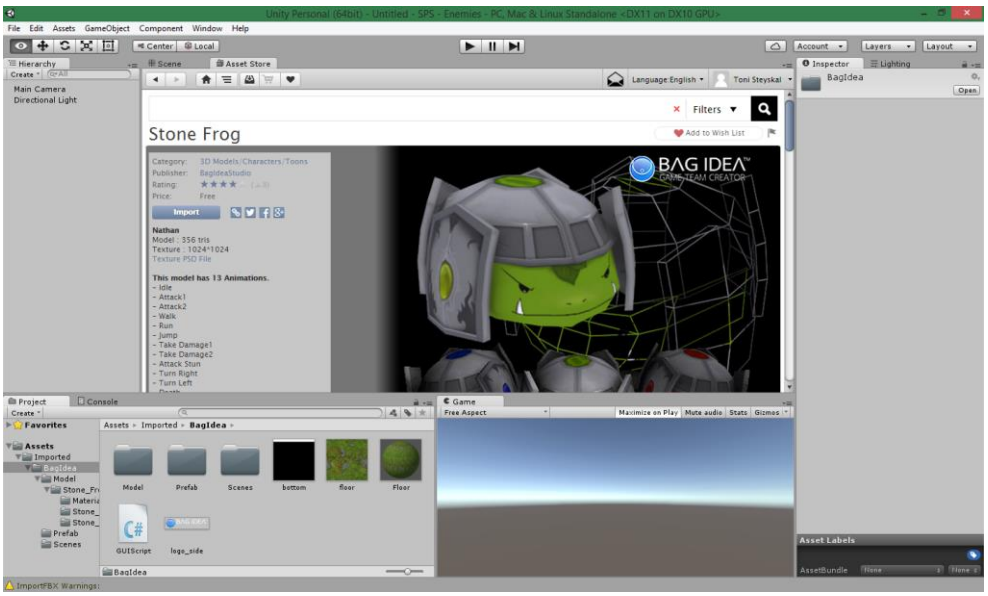
Sveučilište u Zagrebu
Fakultet organizacije i informatike, Varaždin
MTlab@FOI

SPS - Unity – Enemy

Sadržaj

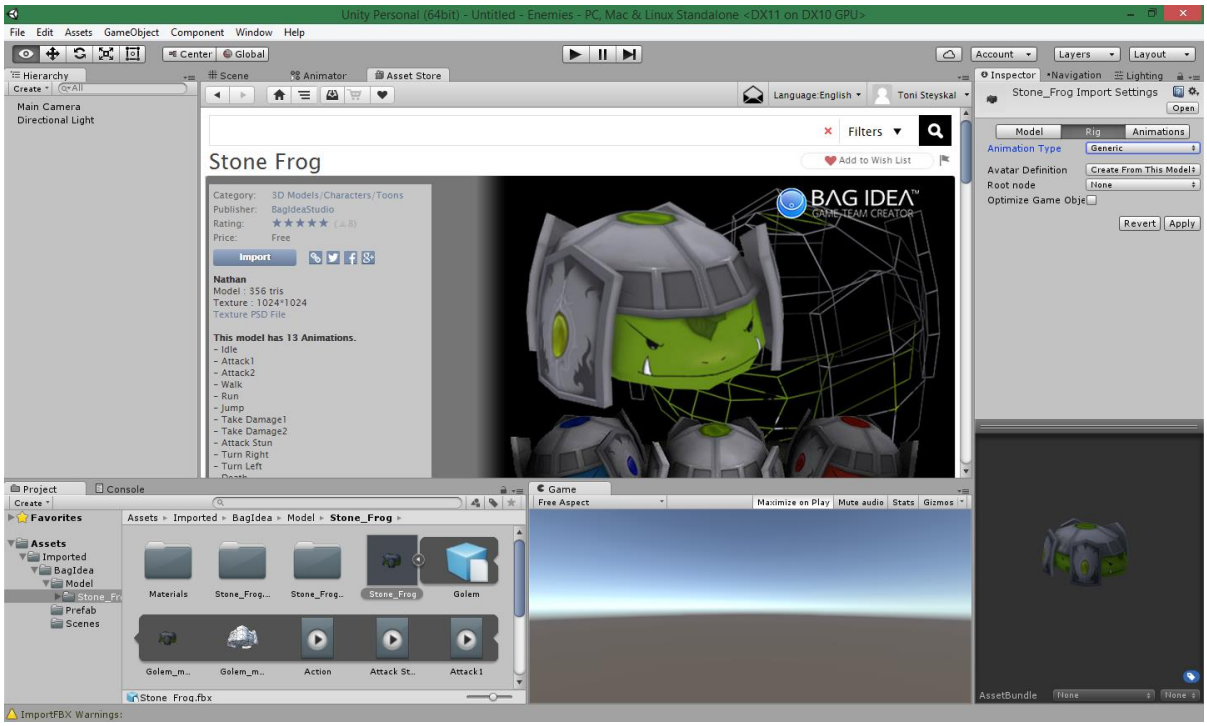
1. Preuzimanje modela

Modeli potrebni za izradu neprijatelja se preuzimaju sa **Unity Asset Store**-a na linku: https://www.assetstore.unity3d.com/en/#!/content/18022. Unutar **Editor**-a kreirajte datoteku **Imported** te u nju preuzmite navedene modele prema slici 1.1.



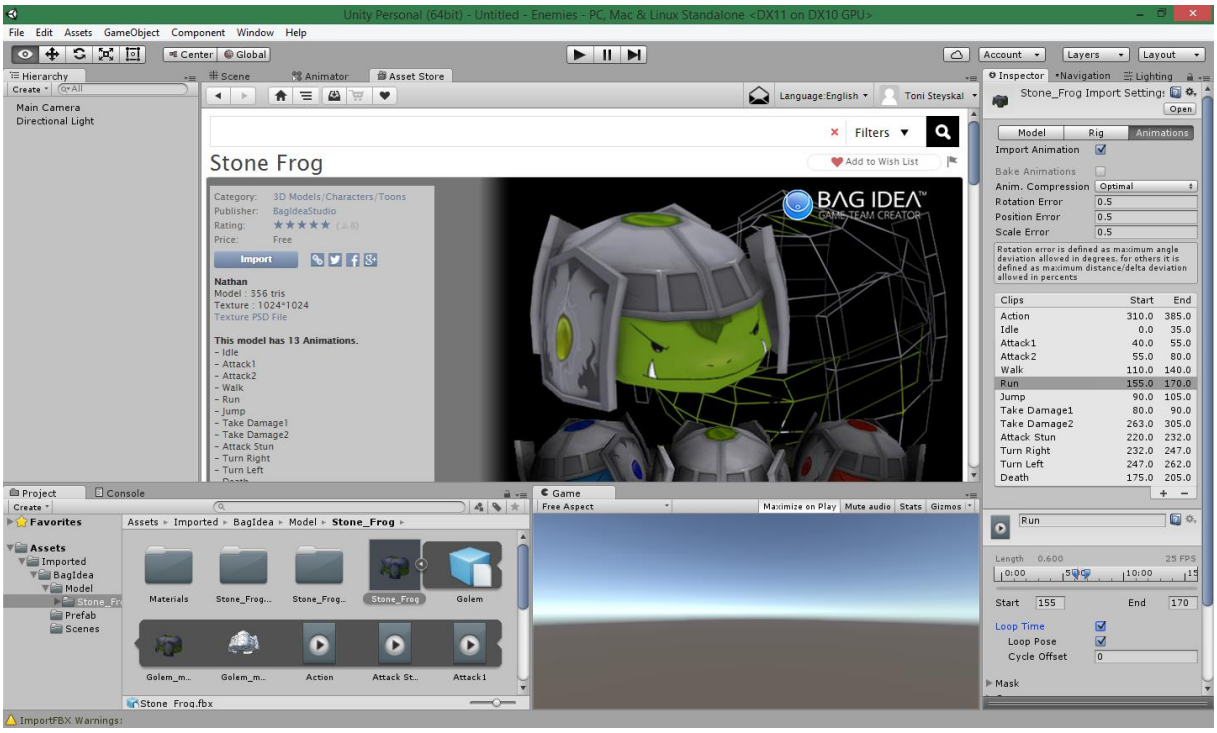
Slika 1.1: *Preuzimanje modela neprijatelja*

Pronađite model **Stone_Frog** unutar uvezenog foldera i promijenite mu tip animacije na **Generic** kao što je prikazano na slici 1.2.



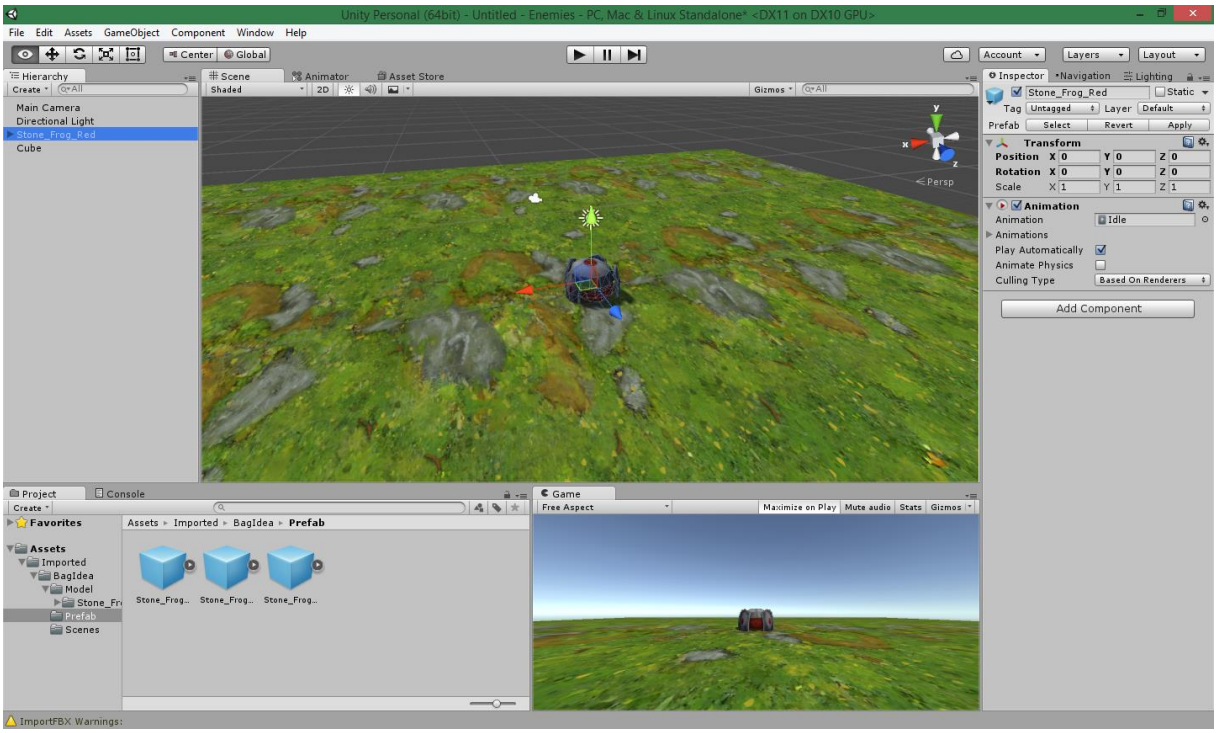
Slika 1.2: *Tip animacije*

Animacije **Idle**, **Walk** i **Run** promijenite u **Loop** animacije prema sljedećoj slici:



Slika 1.3: *Loop animacije*

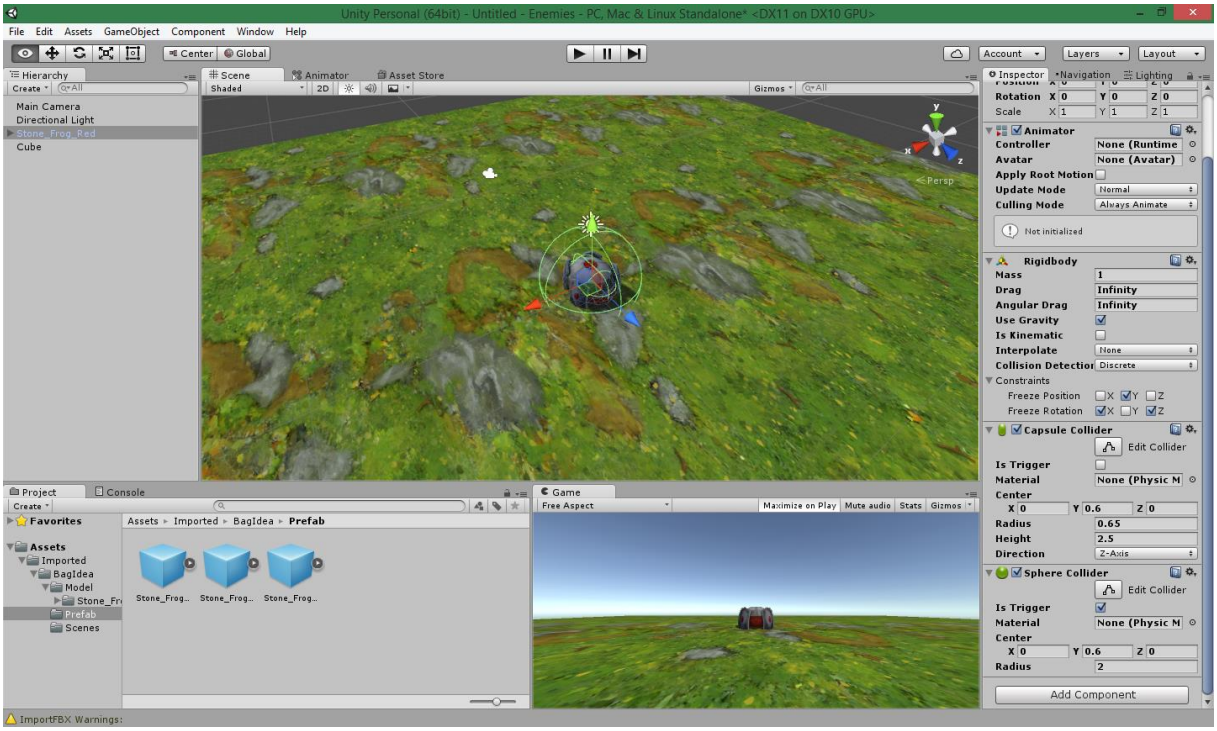
Nakon preuzimanja modela potrebno je **Prefab** jednog od neprijatelja ubaciti u scenu (u ovom primjeru uzet je crveni neprijatelj) te ga pozicionirat na površinu (slika 1.4).



Slika 1.4: *Prefab neprijatelja*

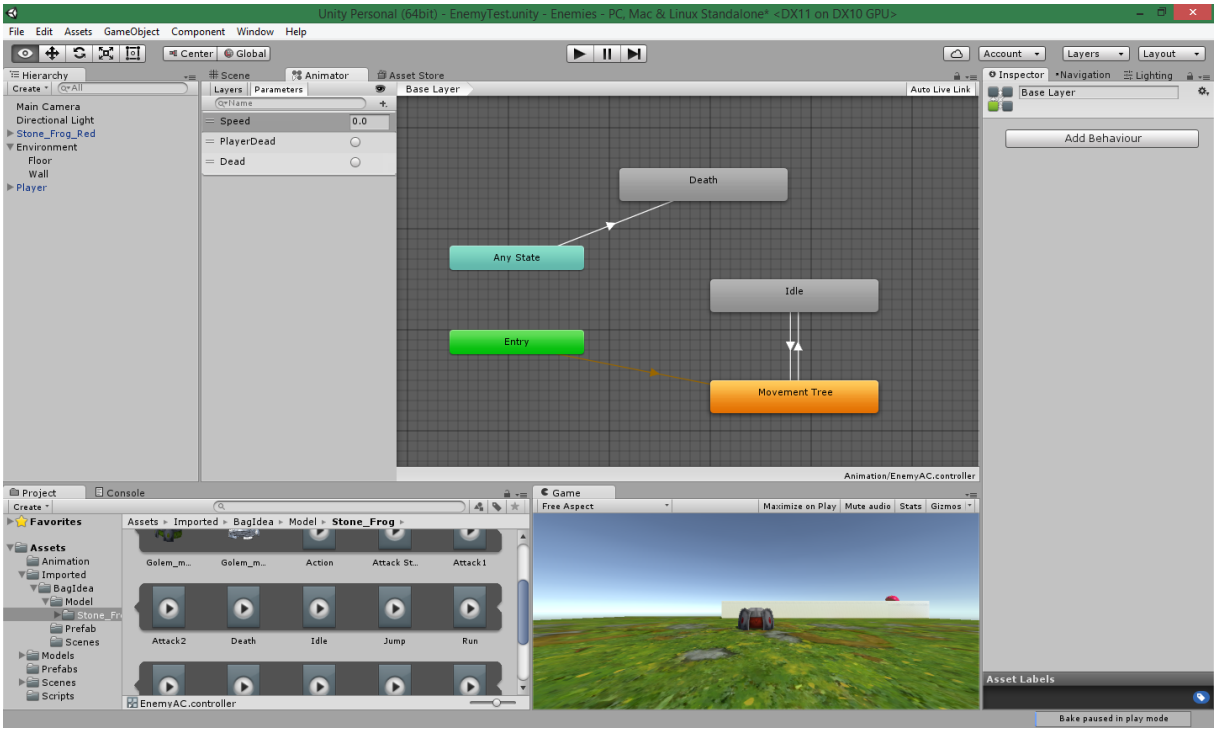
2. Kreiranje neprijatelja

Kako nam komponenta **Animation** unutar **Inspector**-a neprijatelja nije potrebna treba ju ukloniti. Zatim istom dodajte komponente **Animator**, **Rigidbody**, **Capsule Collider** i **Sphere Collider** te ih postavite prema slici 2.1:



Slika 2.1: *Postavke unutar inspector-a*

Sada je potrebno kreirati **Animator Controller** („EnemyAC“) unutar datoteke **Animation** i povezati ga sa **Animator**-om neprijatelja. Otvorite dodani **EnemyAC**, unutar njega kreirajte novi **Blend Tree** („Movement Tree“) i dodajte animacije **Idle** i **Death**. Stanje **Movement Tree** postavite kao **Default State** ukoliko ono to već nije. Potrebno je dodati i određene parametre **PlayerDead** (trigger), **Dead** (trigger) i preimenovati parametar **Blend** u **Speed** (slika 2.2).



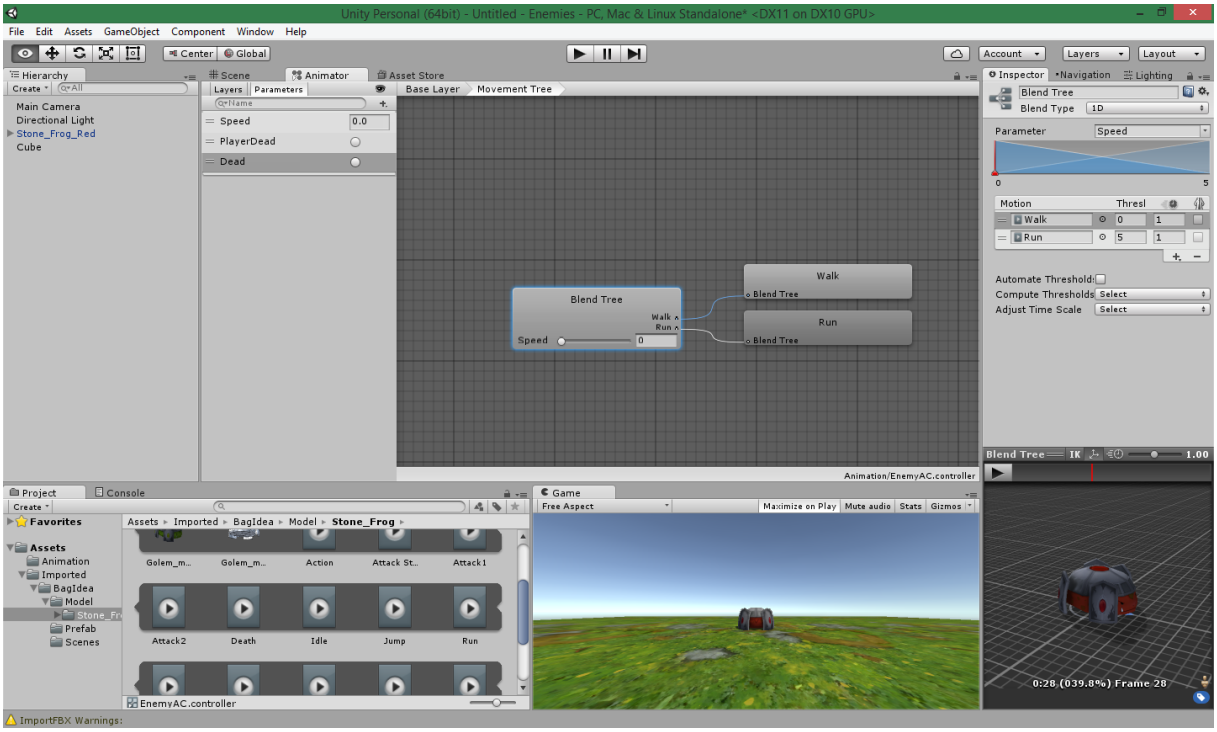
Slika 2.2: *Dodavanje animacija*

Nakon dodanih animacija potrebno je stvoriti tranzicije između njih i povezati s odgovarajućim parametrima prema tablici 2.1.

Tablica 2.1: *Tranzicije između animacija*

From:	To:	Condition:
Idle	Movement Tree	Speed **Greater** 0.1
Movement Tree	Idle	PlayerDead
AnyState	Death	Dead

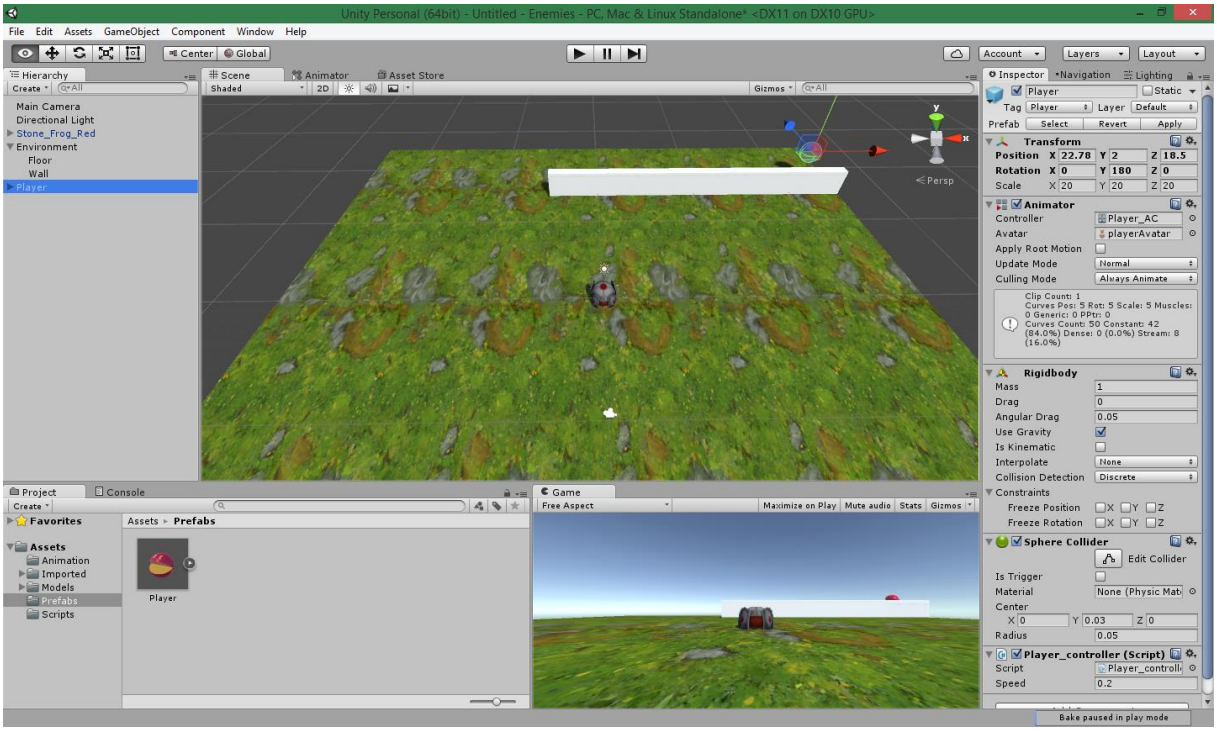
Sada je još potrebno definirati **Movement Tree** animacije, otvorite stanje i u njega dodajte animacije **Walk** i **Run**. Definirajte im postavke prema sljedećoj slici:



Slika 2.4: *Stablo kretanja*

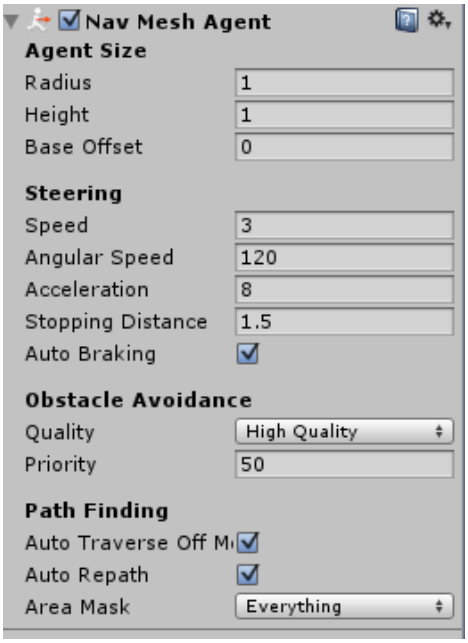
3. Navigacija neprijatelja

Da bi se navigacija pravilno složila potrebno je prvo okolinu definirati kao **Static** i cilj navigacije označiti oznakom **Player** (iskoristiti **Prefab** kreiranog igrača). Primjer okoline i cilja vidljiv je na slici 3.1.



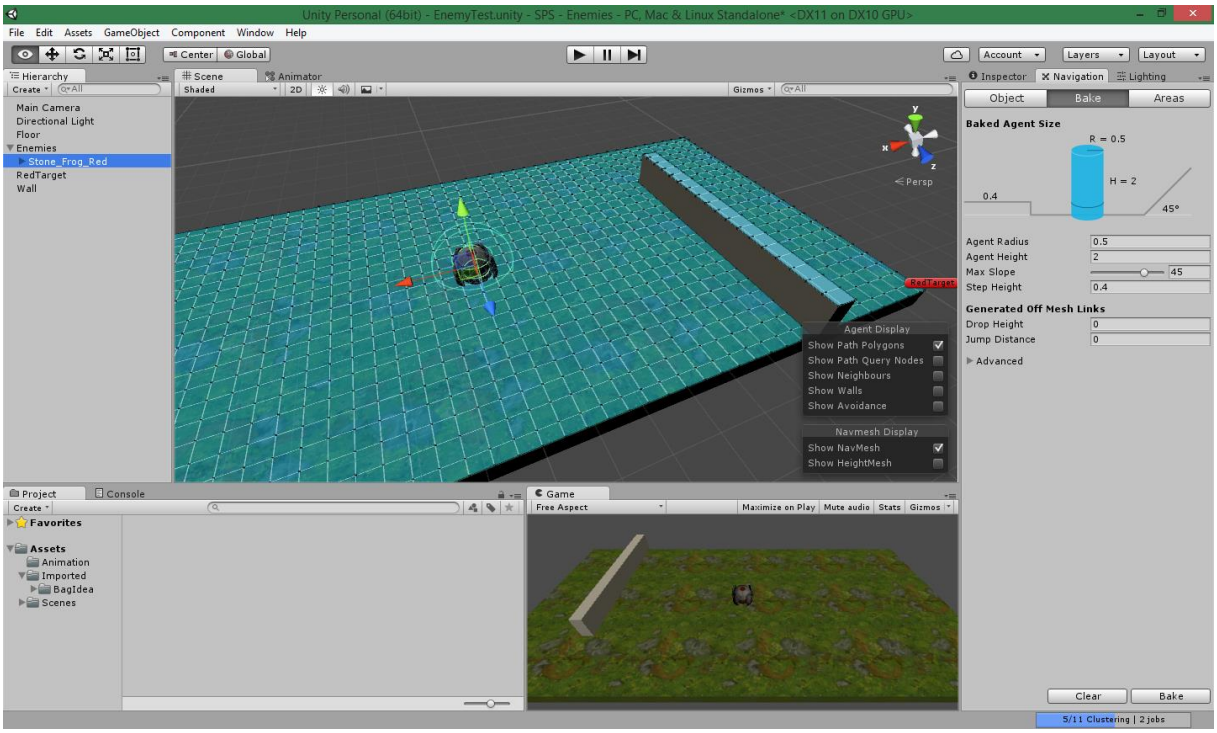
Slika 3.1: *Testna okolina kretanja*

Odaberite neprijatelja i dodajte mu komponentu **Nav Mesh Agent** (slika 3.2).



Slika 3.2: *Nav Mesh Agent komponenta*

Pod **Window > Navigation** se nalazi panel s opcijama za navigaciju, i odaberite **Bake**, nakon završene izrade puteva trebali bi dobiti nešto poput slike 3.3.



Slika 3.3: *Navigacija*

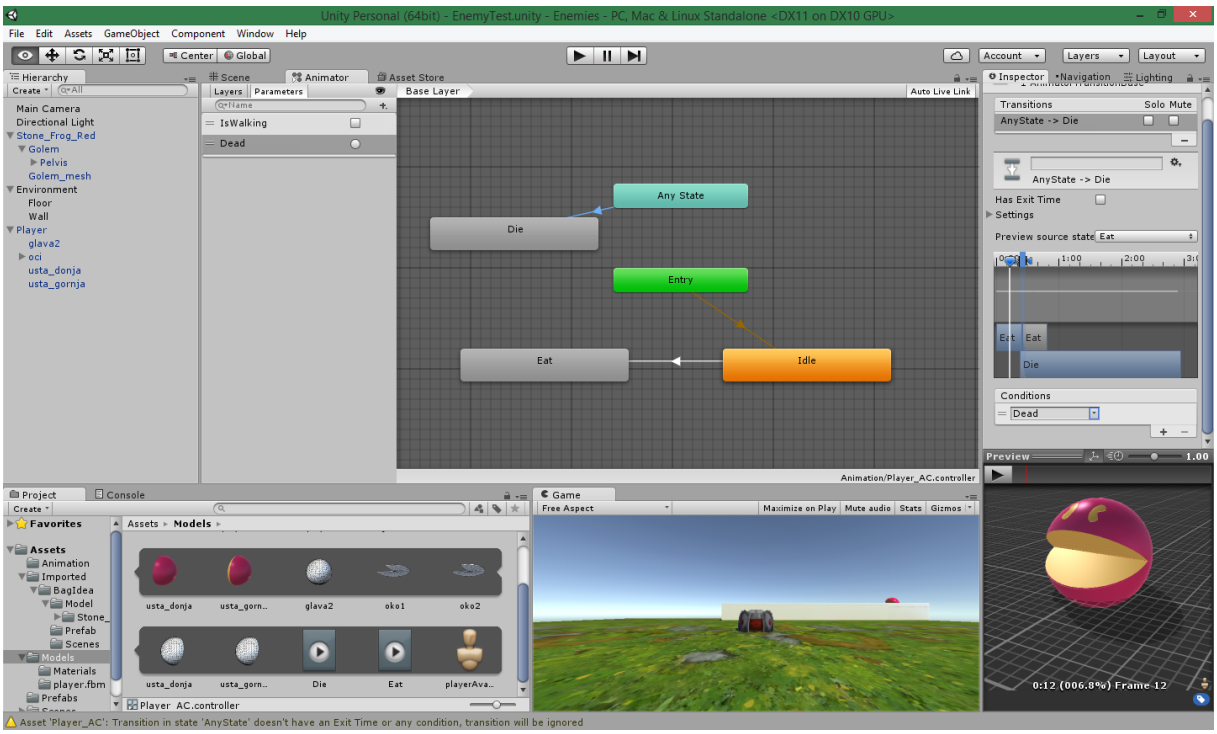
Kreirajte skriptu **EnemyMovement** unutar datoteke **Scripts** i dodajte ju neprijatelju, zatim ju otvorite i nadopunite sljedećim:

```
using UnityEngine;
using System.Collections;

public class EnemyMovement : MonoBehaviour {

    Transform player;               // Reference to the player's position.
    NavMeshAgent nav;               // Reference to the nav mesh agent.

    void Awake ()
    {
        // Set up the references.
        player = GameObject.FindGameObjectWithTag ("Player").transform;
        nav = GetComponent <NavMeshAgent> ();
    }

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        nav.SetDestination (player.position);
    }
}
```

Slika 3.4: *Enemy Movement skripta*

4. Interakcija s igračem

Sada kada naš kreirani neprijatelj prati igrača potrebno je složiti interakciju, odnosno kada ga dodirne da igrač izgubi život. Za to je prvo potrebno dodati parametar **Dead** (trigger) unutar animator igrača, stanje **Dead** s animacijom **Die** i modificirati skriptu igrača **Player_controller** dodavanjem funkcije **Death** i **Die** prema slikama 4.1 i 4.2.



Slika 4.1: *Animator igrača*

```
80      public void Death(){
81          // ... stop the player
82          speed = 0.0f;
83
84          // ... set the Dead parameter inside Animation Controller
85          anim.SetTrigger("Dead");
86
87          // ... call a method after certain amount of time
88          Invoke ("Die", 2.0f);
89      }
90
91      void Die(){
92          // ... remove players collision property
93          GetComponent<SphereCollider>().isTrigger = true;
94
95          // ... destroy this game object after a certain amount of time
96          Destroy(gameObject, 2.0f);
97      }
```

Slika 4.2: *Skripta igrača*

Nakon postavljanja igrača potrebno je dodati još i određenu logiku (funkciju **OnCollisionEnter**) unutar skripte neprijatelja (slika 4.3).

```
public class EnemyMovement : MonoBehaviour {

    Transform player;               // Reference to the player's position.
    Animator animator;              // Reference to the player's animator.
    NavMeshAgent nav;               // Reference to the nav mesh agent.

    void Awake ()
    {
        // Set up the references.
        player = GameObject.FindGameObjectWithTag ("Player01").transform;
        animator = GetComponent<Animator>();
        nav = GetComponent <NavMeshAgent> ();
    }

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        // ... set the destination of the nav mesh agent to the player.
        nav.SetDestination(player.position);
        // ... set the Speed parameter inside Animator Controller
        animator.SetFloat("Speed", nav.velocity.magnitude);
    }

    void OnCollisionEnter(Collision collision){
        if(collision.gameObject.tag == "Player"){
            // ... stop the nav mesh agent
            nav.Stop();
            // ... set the PlayerDead parameter inside Animator Controller
            animator.SetTrigger("PlayerDead");
            // ... call players Death function
            Player_controller playerController = player.gameObject.GetComponent<Player_controller>();
            playerController.Death();
        }
    }
}
```

Slika 4.3: *Skripta neprijatelja*

Sada kada smo u potpunosti kreirali neprijatelja potrebno ga je preimenovati u **EnemyRed** i spremiti kao **Prefab**, jednostavno povucite objekt iz **Hierarchy** u datoteku **Prefabs**. On se kao takav može iskoristiti u više nivoa ili čak u drugoj igri.