Računarska animacija

Blender animacija u Unity razvojnom okruženju

Nikola Vitanović 761

Mentori:

prof. dr. Aleksandar Milosavljević

prof. dr. Aleksandar Dimitrijević

SADRŽAJ

[Unity razvojno okruženje 3](#_Toc65532934)

[Blender 4](#_Toc65532935)

[Priprema za izvoz u FBX 5](#_Toc65532936)

[Izvoz u FBX format 6](#_Toc65532937)

[Uvoz u Unity 7](#_Toc65532938)

[Reprodukovanje animacije 10](#_Toc65532939)

[Upravljanje igrača 12](#_Toc65532940)

[Projekat 14](#_Toc65532941)

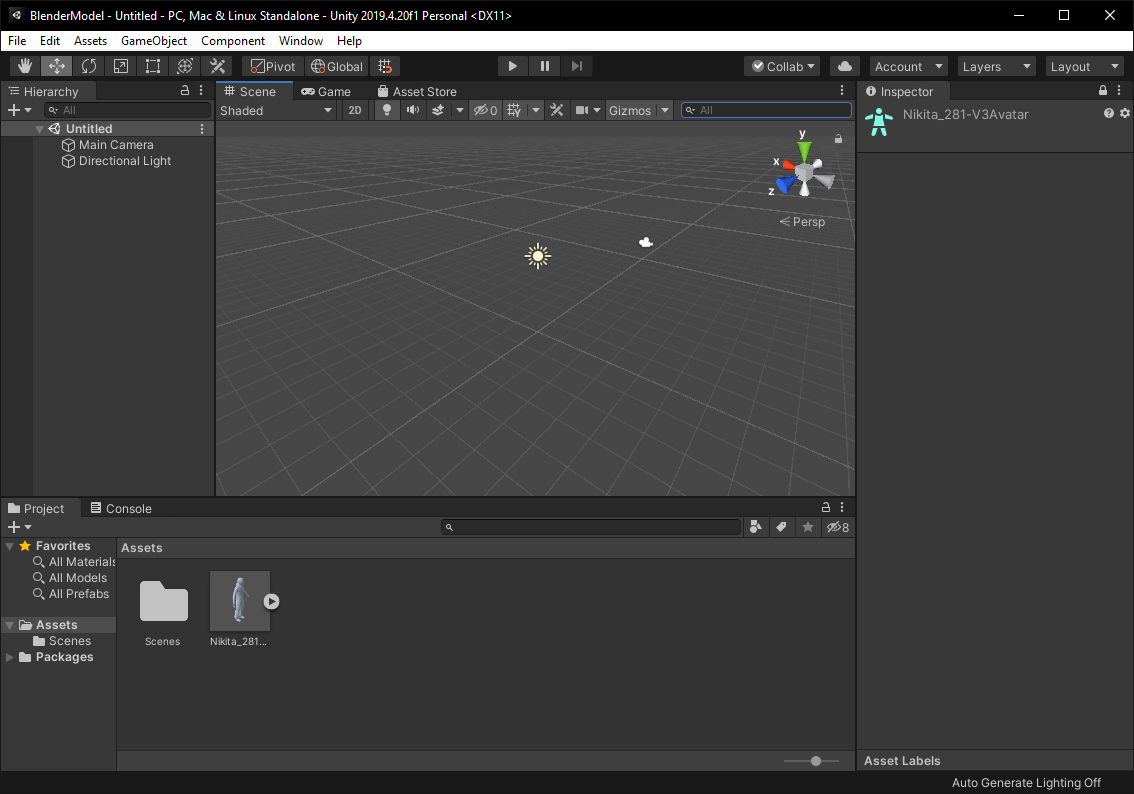
[Zaključak 15](#_Toc65532942)

[Literatura 15](#_Toc65532943)

# Unity razvojno okruženje

Firma “Unity Technologies” osnovana je 2004. godine u malom apartmanu u Kopenhagenu [1]. Predstavlja softversku platformu za izradu kompjuterskih igara. Pored primene u izradi video igara, Unity se može iskoristiti za mnogo više. Sylvio Drouin potpredsednik “Untiy Labs R&D” želi da Unity ne bude samo alat za izradu video igara već da bude 3D operativni sistem ovog sveta. Unity se već koristi za izradu više od polovine igara današnjice, pre svega u indie segmentu zbog svoje lakoće, dokumentacije i velike zajednice.

Unity podržava izradu 3D, 2D, VR, AR igara i simulacija [2], dok se najviše koristi za izradu 3D igara. Program izarđen u Unity razvojnom okruženju moguće je izvesti na više od 25 različitih platformi. Za sve popularniji VR i AR sadržaj, Unity predstavlja najbolju platformu za razvoj. Pored izrade igara od 2010. godine Unity počinje da se dosta koristi u drugim industrijama kao što su automobilska, filmska i 3D industrija.



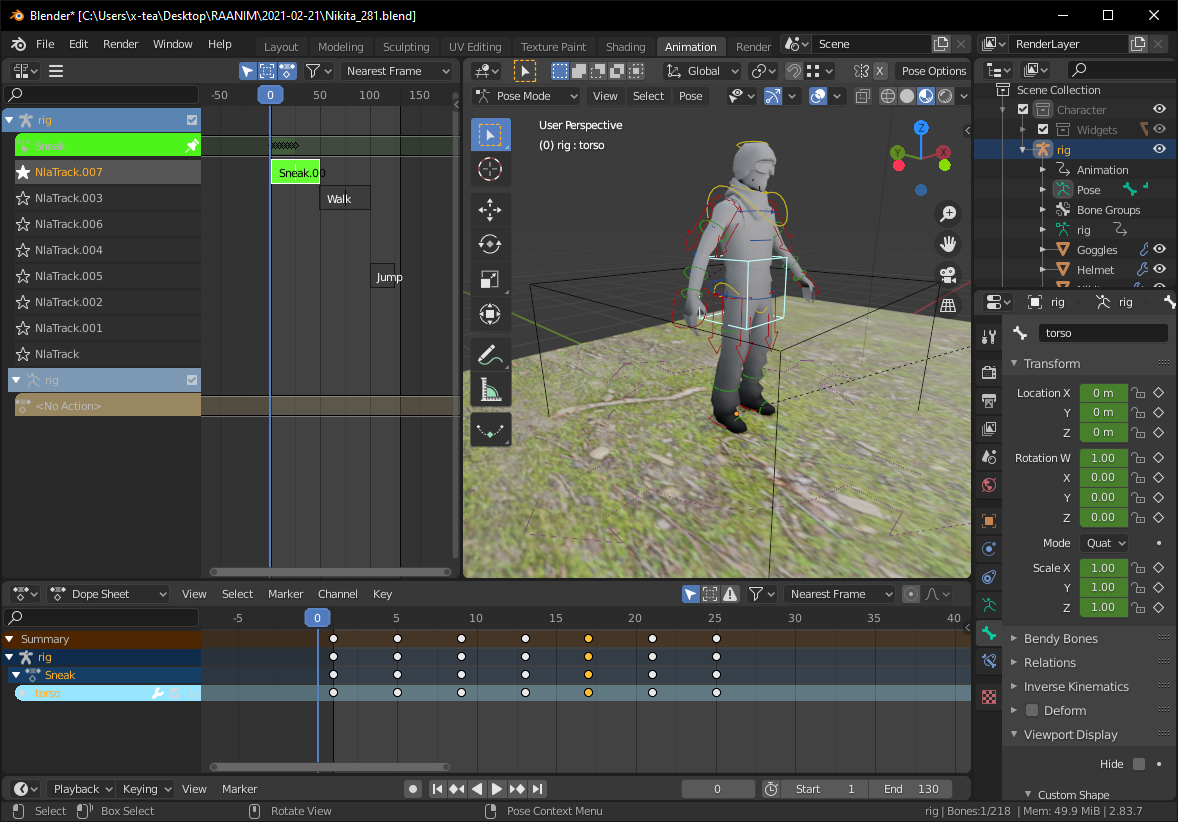
*Slika 1 - Izgled Unity razvojnog okruženja.*

Unity podržava pisanje skripti u C# i Javascript programskom jeziku, pored samog koda u jednom od ponuđenih jezika moćni vizuelni editor omogućava, animaciju modela, dodavanje tekstura, izradu nivoa, rad sa sistemima čestica, simulaciju fizike, audio i drugo.

# Blender

Blender softverski alat [3] je kreiran 02.01.1994. godine, tada je Ton Roosendaal iz Holandije napisao prve izvorne fajlove koda koji su nazvani “Blender”. Inicijalno je planirano da se softver koristi kao aplikacija za Roosendallovu firmu NeoGeo, iz više nezavistnih alata u ceo sistem. U to vreme 3D tehnologija komercijalno nije bila popularna. Po zatvaranju svoje firm NeoGeo, Roosendaal zajedno sa Frank van Beek je osnovao novu kompaniju fokusiranu na razvoj i reklamiranje Blendera. Softver se davao besplatno uz kupovinu dodatnih funkcionalnosti kako se i firma finansirala. Maja 2002. godine osnovana je neprofitna Blender fondacija i izvorni kod Blendera je objavljen pod GNU General Public licencom.

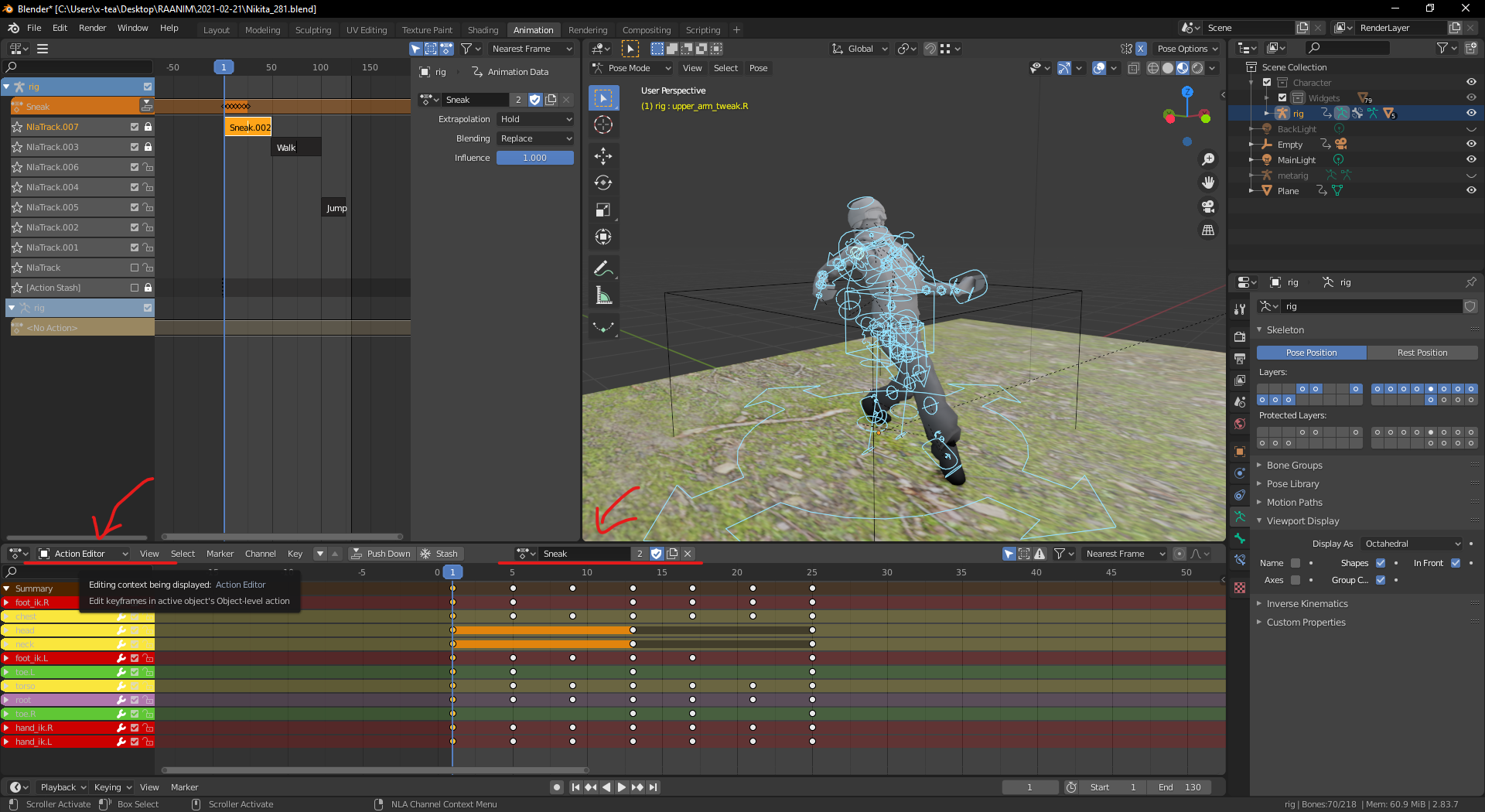
Zbog svoje popularnosti Blender se dosta koristi u indie izradi filmova i igara. Njegova fleksibilnost i dodaci pisani u Python programskom jeziku omogućavaju korišćenje Blendera u raznim industrijama.



*Slika 2 - Blender korisnički intefejs.*

# Priprema za izvoz u FBX

Nakon rigovanja željenog modela [4] potrebno je animirati ga. Animacije je potrebno snimiti u posebne akcije. Koristeći „Action Editor“ potrebno je napraviti par animacija.



*Slika 3 - Action Editor.*

Kreirane animacije moguće je reprodukovati. Animacije se najčešće prave na način da se početni i poslednji korak slažu tj. da je moguće animaciju ponavljati i napraviti iluziju kretanja. Naravno nije svaka animacija jednaka, moguće je napraviti i animaciju koju je potrebno izvršiti jednom ili samo u nekom posebnom slučaju.

Nakon završetka animacije potebno je pritisnuti „Push Down“ dugme koje će sačuvati vašu animaciju i postaviti je na NLA traku. Ukoliko ne želite da dodate anmaciju kao NLA traku potrebno je zaštiti animaciju, jer se Blender trudi da ukloni sve objekte koji se ne koriste iz datoteke pri gašenju programa, to znači da animacije koje nisu korišćenje biće uklonjene pri zatvaranju editora bez obzira na to da li ste snimili projekat. Zbog toga se savetuje biranje opcije „Fake User“ za svaku animaciju koja je trajna.

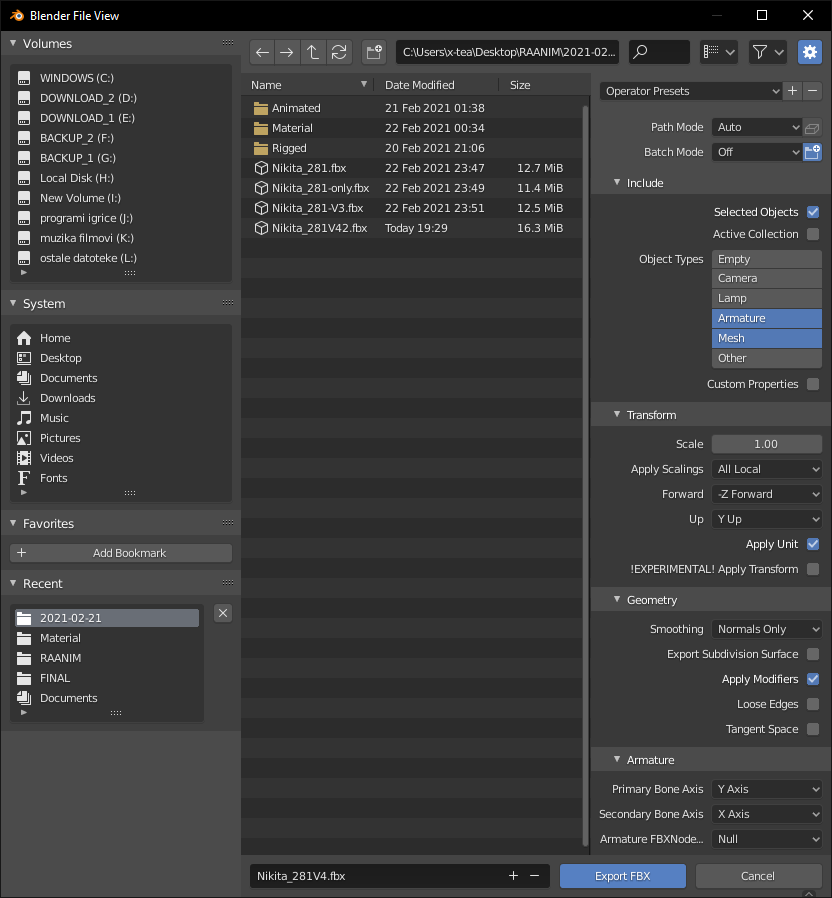


*Slika 4 - Snimanje kreirane akcije bez dodavanja na NLA traku.*

# Izvoz u FBX format

Kako bi koristili model koji smo kreirali i animirali u Blenderu potrebno je izvesti ga u FBX format [5]. Modeli izvezeni u FBX formatu omogućavaju da pored same 3D geometrije modela i teksture izvezemo anmaciju, kretanje, 2D, 3D, audio pa čak i video sadržaj. FBX format je jedan od popularnijih formata. Unity okruženje obuhvata uvoz modela iz FBX formata uključujući i animacije.

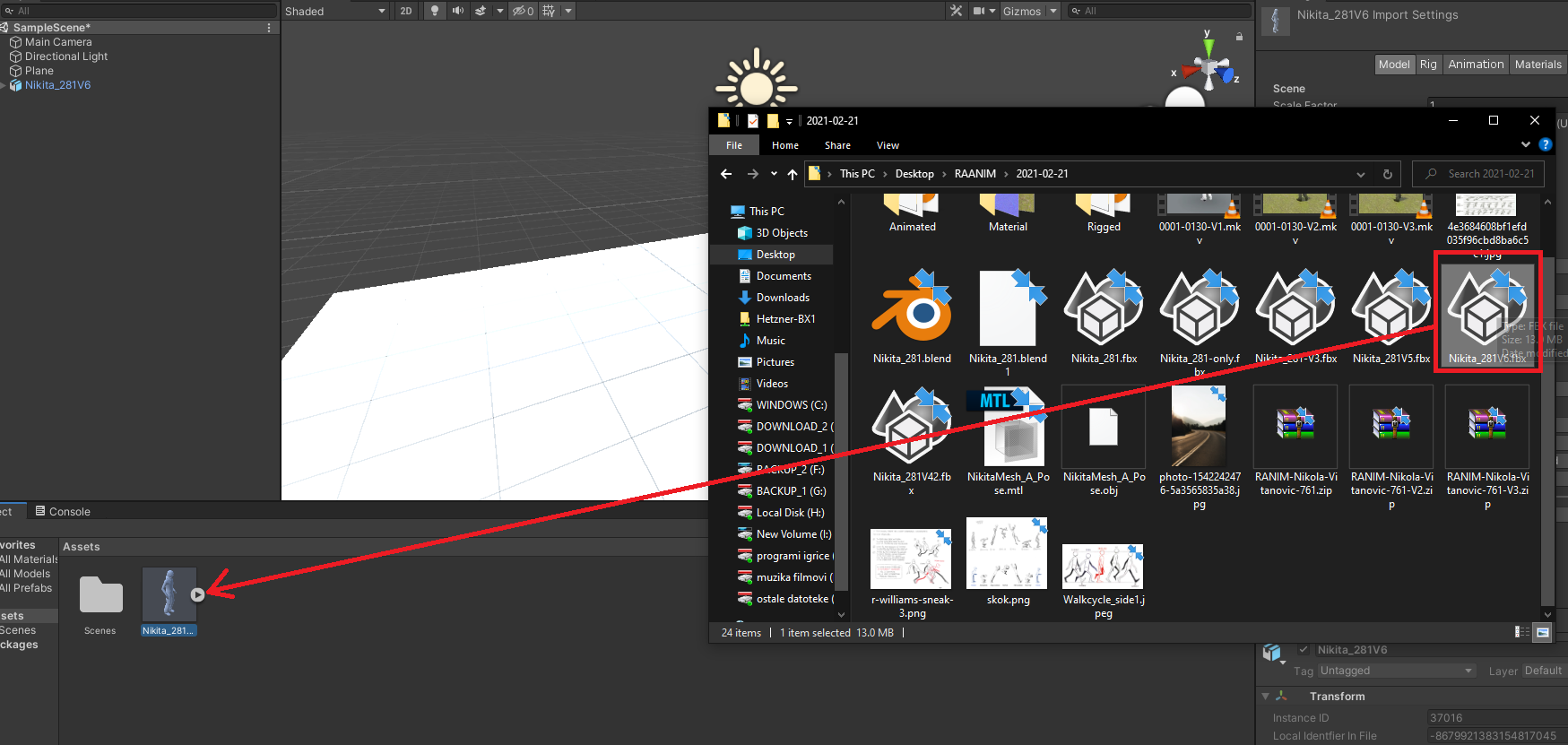
U Blenderu potrebno je selektovati model sa svim svojim delovima. Nakon selektovanja potrebno je izabrati „File -> Export -> FBX“. S obzirom da za prikaz animacije u Unity okruženju nama je potreban model i animacije biramo opcije kao na slici.



*Slika 5 - Izvoz animiranog modela u FBX format.*

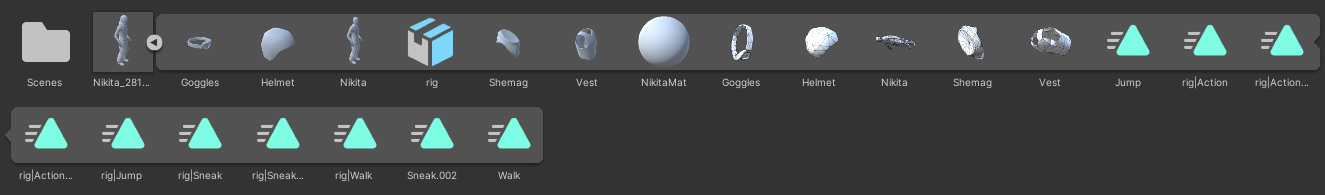
# Uvoz u Unity

Dovoljno je prevući FBX datoteku u „Assets“ folder i Unity će automatski pokušati da učita sve informacije o modelu.



*Slika 6 - Uvoz FBX datoteke u Unity.*

FBX asset koji se uveze u Unity je predstavljen kao Prefab. Taj Prefb u sebi sadrži različite korisne objekte. U našem slučaju možemo tačno videti svaki deo objekta iz Blendera. Pored 3D modela možemo da vidimo da postoje dodatni objekti koji predstavljaju naše kreirane animacije.

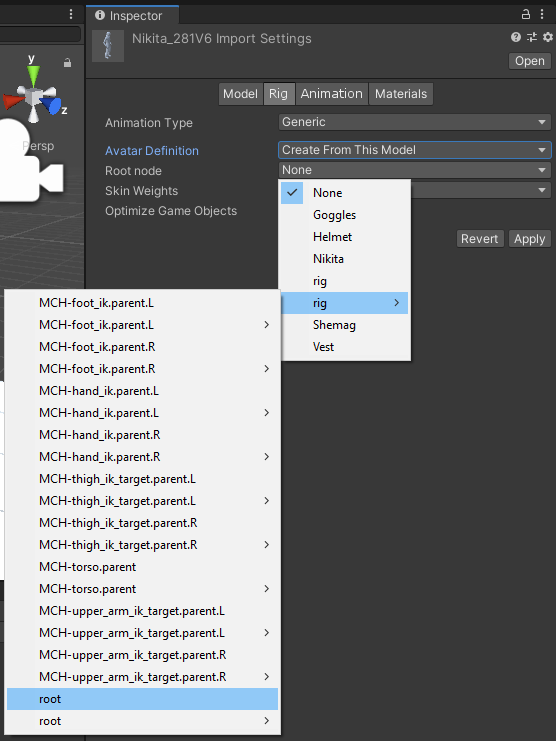


*Slika 7 - Delovi FBX asseta koji su izvezeni iz Blendera.*

Pre nego što je moguće koristiti asset koji smo uvezli, potrebno je podesiti kakav tip animiranog objekta je u pitanju. Unity podržava nekoliko:

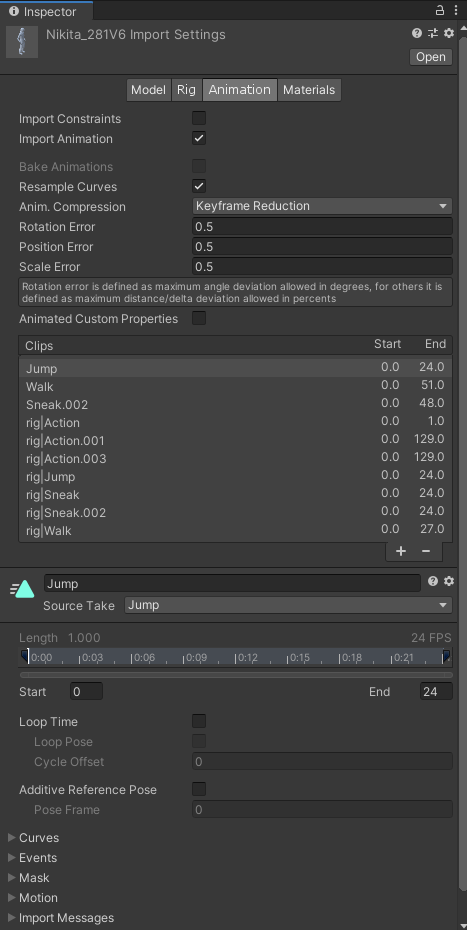
* None – objekat nije animiran.
* Legacy – koristi stari način za animiranje korišćen u Unity 3.x i ranije.
* Generic – najčešće se koristi ako je u pitanju model koji nije humanoidni, moguće je izabrati drugačiju početnu kost. Naravno nije ograničeno izabrati ovu opciju i u slučaju da je vaš model humanoidni.
* Humanoid – ukoliko model predstavlja čovekoliki oblik, Unity automatski detektuje kostur objekta i mapira ga. Ukoliko nije lepo detektovan potrebno je uraditi dodatnu konfiguraciju. Neki softveri ne prate konvenciju koju Unity očekuje u tim slučajevima može biti lakše upotrebiti Generic model umesto humanoidnog.

Potrebno je izabrati model i kroz “Inspector” koji se nalazi sa desne strane izabrati tip animacije koju želite u jezičku „Rig“, podesiti parametre animacije, nakon čega se promena primenjuje klikom na „Apply“ dugme.



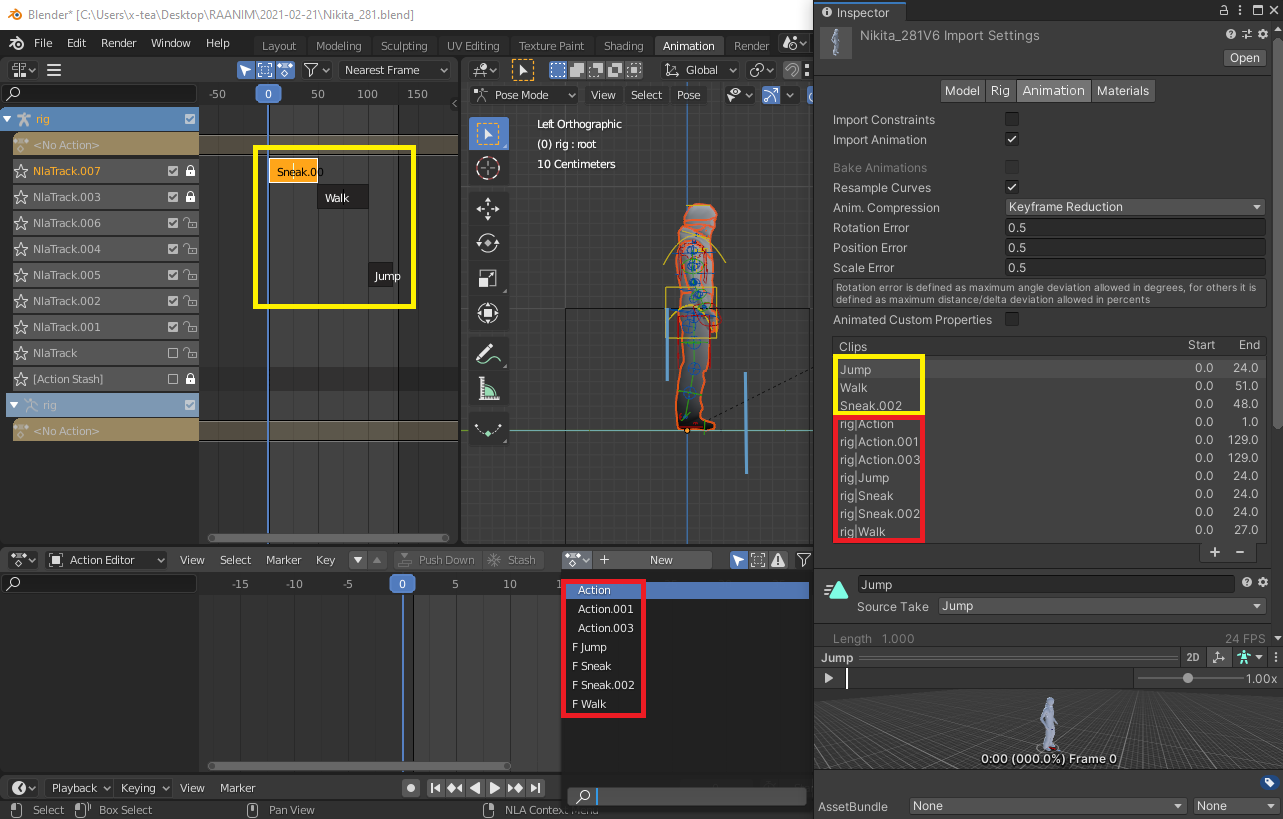
*Slika 8 - Podešavanje uveženog modela u Unity okruženju.*

U jezičku “Animation” Unity pokušava da detektuje postojeće animacije koje su došle sa modelom kroz FBX datoteku. Animacije će biti vidljive u delu “Clips”. Animacije imaju podešen početni i završni frejm, moguće je modifikovati dužinu ukoliko Unity nije detektovao pravu dužinu. Takođe moguće je kreirati animacioni klip od dela duže animacije.



*Slika 9 - Uvezene animacije dostupne za izabrani model.*

Animacije kreirane u Blenderu mogu biti NLA trake i Akcije. Oba vida animacija su vidljiva u Unity okruženju, samo je imenovanje drugačije. Ukoliko je NLA traka animacija u Unity okruženju će se zvati isto kao ime iz Blender NLA trake, dok animacije iz „Action Editor“ dela dobijaju prefiks. Prefiks koji se dobija zavisi od imena koji Rigify ekstenzija generiše, najčešće je to „rig“, mada može biti i nešto drugo ukoliko se promeni.



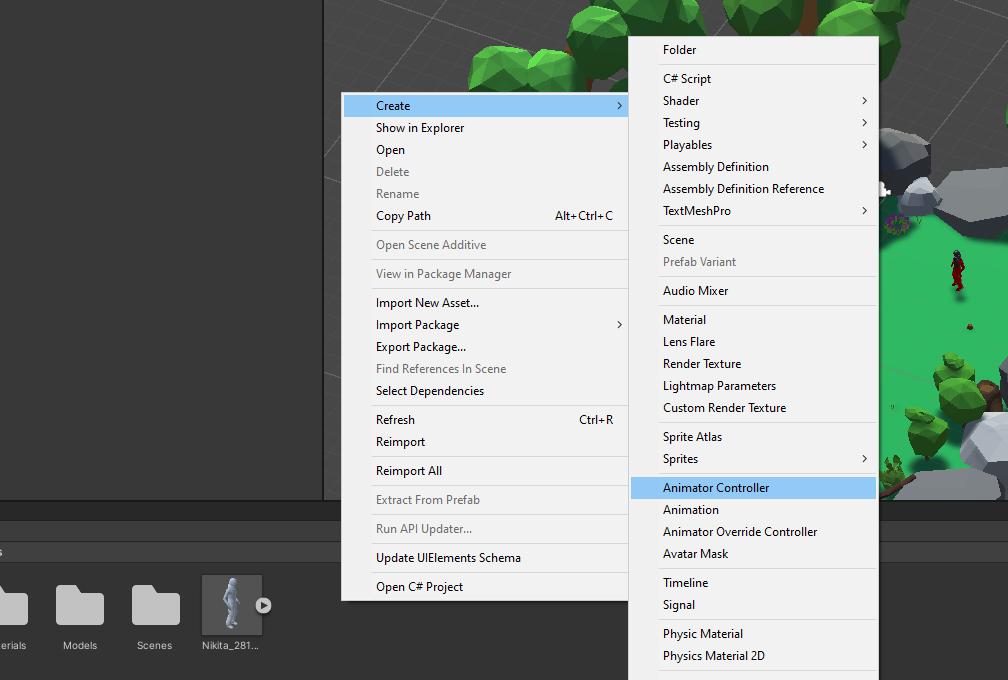
*Slika 10 - Mapiranje animacija iz Blendera u Unity okruženju.*

Moguće je izmeniti i adaptirati svaki animacioni klip ponaosob ukoliko je potrebna korekcija u Unity okruženju.

Pored uvoza FBX datoteke moguće je koristiti blender datoteku direktno. Unity će automatski učitati blender datoteku kao asset ukoliko se ona nalazi u Assets folderu Unity projekta. Ovo omogućuje da uživo modifikujete model u Blenderu bez potrebe da se ažuriraju delovi. U blenderu model mora biti rigovan i podešen za animaciju. Važe ista pravila kao za FBX datoteku. Najčešća greška koju korisnici prave pri ovakvom korišćenju je ostavljanje zvezdice na neku od NLA traka, na taj način se sve animacije osim te iz NLA trake se blokiraju.

# Reprodukovanje animacije

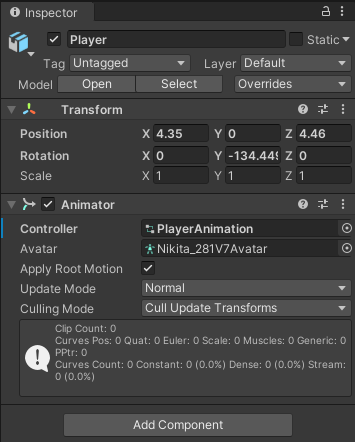
Za reprodukovanje animacije potrebno je iskoristiti Unity animacioni kontroler [6]. Animacioni kontroler omogućava jednostavan rad sa animacionim klipovima koji su uvezeni iz Blendera. Da bi animacija bila reprodukovana potrebno je da se stavi u animacionom kontroleru tog objekta. Potrebno je dodati animacioni kontroler i povezati ga sa objektom na sceni.



*Slika 11 - Dodavanje akcionog kontrolera.*

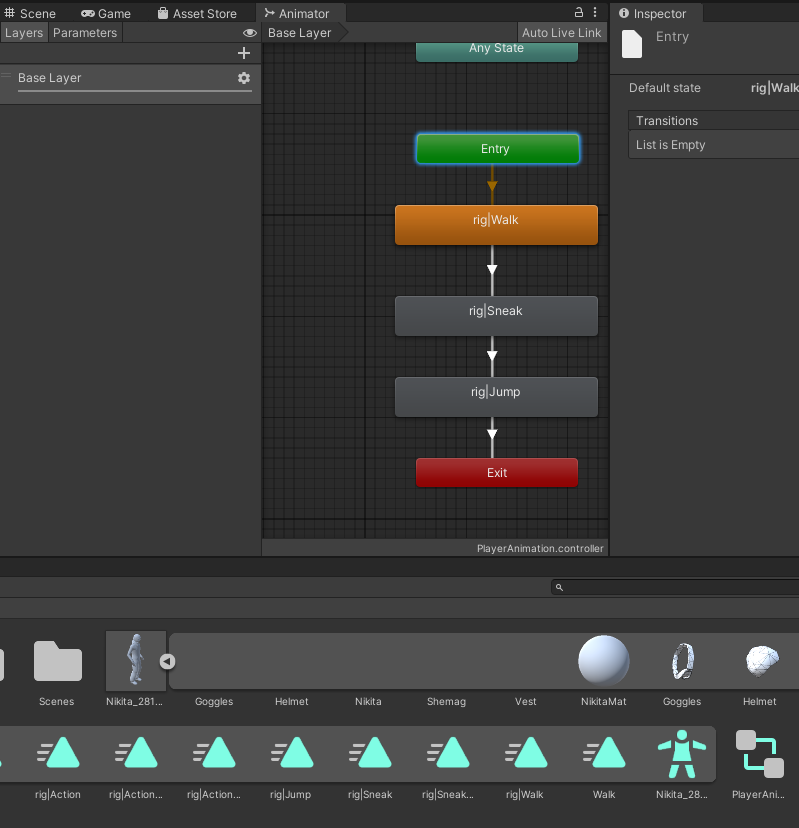
Nakon dodavanja animacionog kontrolera, potrebno je povezati objekat na sceni i novi akcioni kontroler. To je moguće odraditi kroz inspektor u sekciji „Animator“. Kada je dodat animacioni kontroler objekta povezan sa njegovom „Animator“ komponentom moguće je kreirati sekvencu animacija u zavisnosti od stanja u kome se objekat nalazi. Ukoliko se animacije međusobno povežu od „Entry“ do „Exit“ stanja sve animacije će se pokrenuti po redosledu povezivanja grafa.

Sama animacija iz Blendera može da predstavlja stanje na grafu. Ukoliko karakter treba da hoda njegovo stanje bi trebalo da bude „rig|Walk“ u našem slučaju. Dovoljno je uhvatiti animaciju iz Asseta i prevući je na Animacioni graf, ta animacija će se pojaviti kao stanje na grafu.



*Slika 12 - Objekat sa povezanim akcionim kontrolerom.*

Da bi se animacija repordukovala potrebno je povezati je sa početnim stanjem. Ukoliko je povežemo is a izlaznim stanjem animacija će da se ponavlja zauvek. Primer na slici ispod pokazuje kako animacija prolazi iz stanja hodanja u stanje šunjanja i finalno završava u stanje skoka. Nakon finalnog stanja animacija se pokreće ponovo.



*Slika 13 - Dodavanje animacija i povezivanje.*

Naravno nama je bitno da se animacija pokrene u slučaju da korsnik vrši neku akciju. Primera radi, kada korisnik pritisne taster “W” želimo da naš igrač krene da hoda, dok kada pritsnemo “Shift + W” naš igrač treba da se šunja.

# Upravljanje igrača

Kako bi igrač upravljao karakterom potrebno je kreirati skriptu sa imenom „PlayerController.cs“, nakon toga napisati kratak deo koda i zakačiti ga za igrača na sceni. Skripta reprodukuje animaciju i pomera karaktera u zavisnosti od pritisnutih tastera.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerController : MonoBehaviour

{

public Animator animator;

void Start()

{

animator = GetComponent<Animator>();

}

void Update()

{

if(Input.GetKey(KeyCode.LeftShift) && Input.GetKey(KeyCode.W))

{

// Sneak animation

animator.SetBool("isSneaking", true);

animator.SetBool("isWalking", false);

// Move the player twice slower

transform.Translate(Vector3.forward \* 2 \* Time.deltaTime);

}

else if(Input.GetKey(KeyCode.W))

{

// Walk

animator.SetBool("isSneaking", false);

animator.SetBool("isWalking", true);

// Move the player forward

transform.Translate(Vector3.forward \* Time.deltaTime);

}

else

{

// Stop all animations

animator.SetBool("isWalking", false);

animator.SetBool("isSneaking", false);

}

// Rotation

if (Input.GetKey(KeyCode.A))

{

// Rotate the player left

transform.Rotate(Vector3.down \* 50 \* Time.deltaTime, Space.Self);

}

else if (Input.GetKey(KeyCode.D))

{

// Rotate the player right

transform.Rotate(Vector3.up \* 50 \* Time.deltaTime, Space.Self);

}

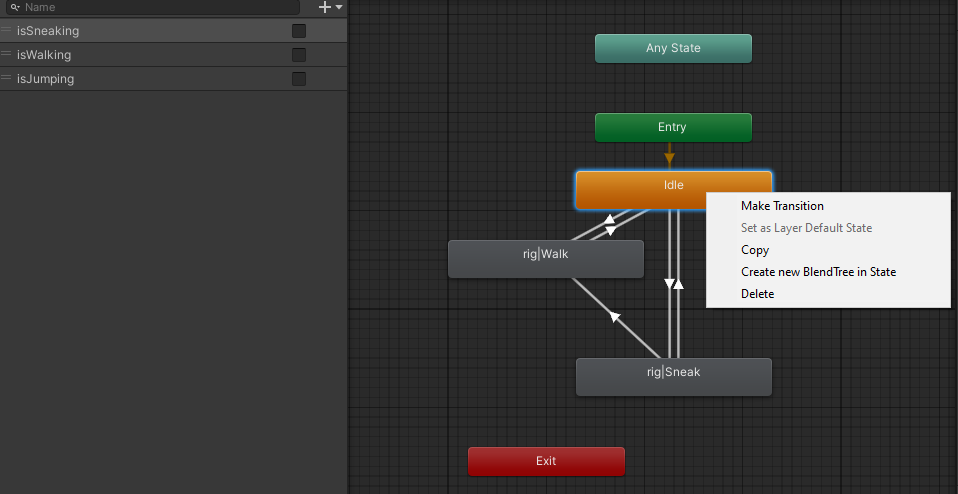
}

}

*Deo koda 1 - PlayerController.cs vrši upravljanje igrača.*

Kako bi ovaj kod bio funkcionalan moramo izmenit naš animacioni graf za igrača na sceni. Potrebno je dodati parametre: „isWalking“ i „isSneaking“ tipa boolean, takođe moramo izmeniti naš animacioni graf da sadrži povratne akcije. Svaka od strelica na grafu prikazuje smer animacije. Smer animacije je veoma bitan za prelaz iz jednog stanja u drugi. Svaka od strelica može da ispuni određeni uslov definisan u listi parametara.

Potrebno je dodati prazno stanje koje ćemo nazvati „Idle“, i prebaciti ga u početno stanje desnim klikom i izborom opcije „Set as Layer Default State“. Sada animacija počinje u tom stanju. U našem slučajo to je prazno stanje koje ne reprodukuje animaciju, tako da naš karakter samo stoji u mestu.



*Slika 14 - Stanja na animacionom grafu.*

Dodaćemo ostale strelice za prelaz iz jednog stanja u drugo kao na slici. Svaka od strelica može imati određeni uslov za prelaz iz početnog stanja u neko drugo stanje. Recimo da prelazimo iz „Idle“ u „rig|Walk“ stanje, potrebno je kliknuti na strelicu i sa desne strane dodati uslov da je „isWalking“ podešen na „true“. Ovo omogućava našem kodu da aktivira u „rig|Walk“ stanje koje pusta animaciju hodanja. Povratna strelica ka „Idle“ stanju sadrži obrnut uslov da je „isWalking“ podešen na „false“ i to prekida animaciju hodanja.

# Projekat

Igrač upravlja u istom trenutku oba modela, s tim što su modeli različito rotirani. To mu otežava kretanje kroz mapu, jer cilj igre je da oba modela u istom trenutku dotaknu loptu. Nakon doticanja lopte igrači se teleportuju na različite pozicije i lopta se ponovo pojavljuje sa neba. Mehanika igre je prosta, ali zbog suprotnog upravljanja, igra može biti veoma frustrirajuća. Takođe na svakih 10 sekundi pojavljuje se nova prepreka na mapi koja otežava kretanje igraču. Kako igrač povećava svoj rezultat tako se i loptica smanjuje.



*Slika 15 - Izgled igre.*

Model u projektu je uvežen kao generička animacija. Koriste se samo animacije koje su dodate kroz „Action Editor“ u Blenderu. Animacija je prikazana na slici 14.

Najglavniji deo igre predstavlja „GameManager.cs“. Funkcija ovog dela koda je upravljanje igrom i praćenje rezultata. Pored ove skripte, postoji i „BallController.cs“ koji prati da li oba igrača dodiruju loptu. Ukoliko je to ispunjeno rezultat se povećava.

private void OnCollisionEnter(Collision collision)

{

collisions.Add(collision.transform.tag);

if (collisions.Contains("P1") && collisions.Contains("P2"))

{

Debug.Log("Collision WIN!");

gm.ReinitializeGame();

}

}

*Deo koda 2 - Provera kolzije igrača i lopte.*

Konačno „PlayerController.cs“ prikazan u delu koda 1, predstavlja jednostavno upravljanje igrača.

Moguća unapređenja igre su dodavanje novih funkcionalnosti, kao što je skuplanje paketa koji bi promenili način kretanja igrača, npr. fiksirali jedan model u mestu dok bi se drugi kretao po komandama igrača.



*Slika 16 - Izgled igre sa većim rezultatom.*

# Zaključak

Blender kao besplatan softver otvorenog koda za modeliranje i animiranje predstavlja moćan alat kako za početnike tako i za napredne korsinike, verzije od 2.8 i novije mogu da pariraju ogromnim softverima kao što su 3D Studio Max i Maya. Unity je postao standard za izradu indie igara, pa čak i većih projekata uz mnoštvo materijala na internetu moguće je izraditi koncept igre za manje od 24h.

U 2021. godini veoma je jednostavno razviti igricu iz koncepta koji zamislite, sa mnoštvom sadržaja dostupnim putem interneta i uz dovoljno vrmena svako bi mogao da kreira igru od modela do finalnog proizvoda.

# Literatura

|  |  |
| --- | --- |
| [1] | „Unity (game engine),“ Wikipedia, 2021. [Na mreži]. Available: https://en.wikipedia.org/wiki/Unity\_(game\_engine). |
| [2] | U. Technologies, „Unity User Manual (2019.4 LTS),“ 2021. [Na mreži]. Available: https://docs.unity3d.com/Manual/index.html. |
| [3] | „Blender (software),“ Wikipedia, 2021. [Na mreži]. Available: https://en.wikipedia.org/wiki/Blender\_(software). |
| [4] | L. P. Level, „Do I Build My Own Rig Or Use Rigify? | Blender 2.8 - Rigging For Animation,“ Youtube, [Na mreži]. Available: https://www.youtube.com/watch?v=J\_BYqfckdLU. |
| [5] | R. S. LLC, „Blender 2.8 Exporting FBXs to Unity 3D (In 60 Seconds!),“ Youtube, 30 06 2019. [Na mreži]. Available: https://www.youtube.com/watch?v=ysl0qYq5p9w. |
| [6] | Imphenzia, „Blender 2.82 - Rigify to Unity Tutorial - How to Export a Rigify Character and Import it into Unity,“ Youtube, 2020. [Na mreži]. Available: https://www.youtube.com/watch?v=vrN9duEoA6g. |