Sistemi virtuelne i proširene realnosti

Virtuelna realnost korišćenjem SteamVR dodatka u Unity razvojnom okruženju

Nikola Vitanović 761

Mentor:

prof. dr. Aleksandar Milosavljević

**Sadržaj**

[Unity razvojno okruženje 3](#_Toc73310528)

[Virtuelna realnost 4](#_Toc73310529)

[Vridge 4](#_Toc73310530)

[SteamVR 8](#_Toc73310531)

[Unity SteamVR dodatak 8](#_Toc73310532)

[Projekat 9](#_Toc73310533)

[Podešavanje 9](#_Toc73310534)

[Struktura projekta 13](#_Toc73310535)

[XRRig objekat 13](#_Toc73310536)

[CharacterController 13](#_Toc73310537)

[Kretanje 13](#_Toc73310538)

[Vrata 13](#_Toc73310539)

[Dugme 13](#_Toc73310540)

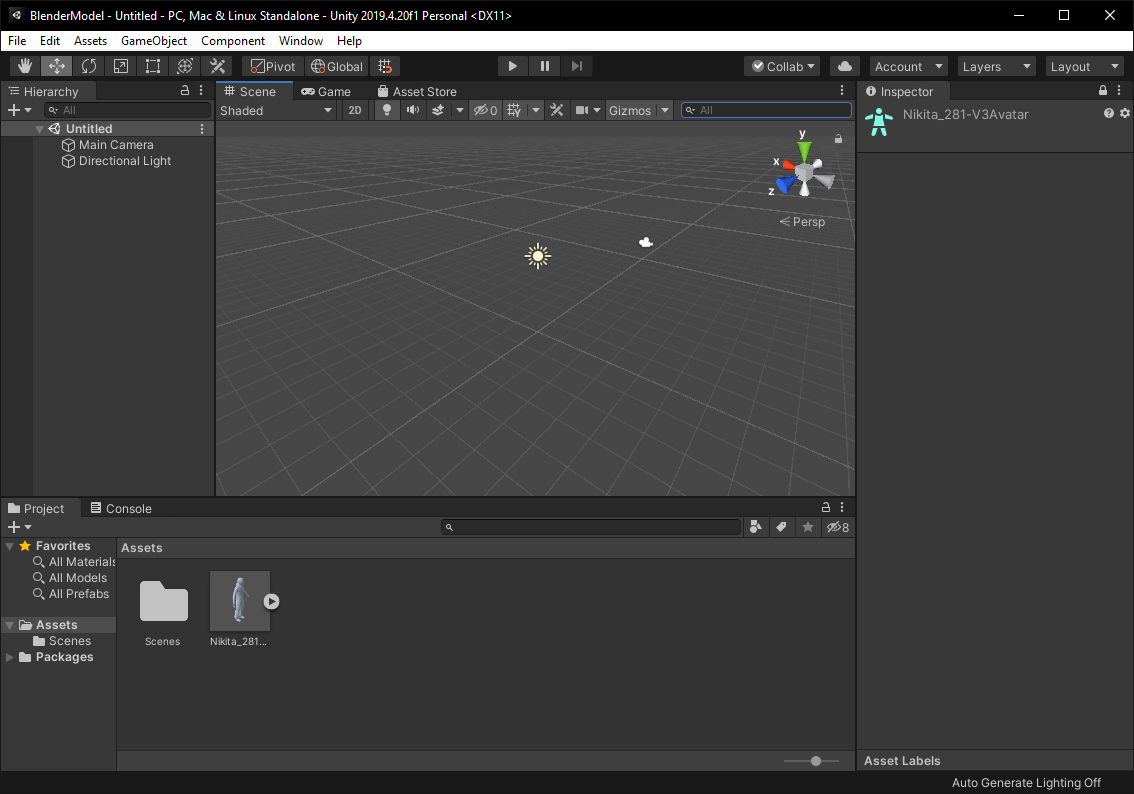
[Kutija 13](#_Toc73310541)

[Zaključak 13](#_Toc73310542)

# 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.

# Virtuelna realnost

Virtuelna realnost predstavlja kompjuterski generisanu simulaciju u kojoj osoba može da vrši interakciju u virtuelnom trodimenzionalnom svetu korišćenjem naočara, rukavica ili kontrolera za virtuelnu realnost. Ovo iskustvo je najpribližnije realnom svetu i pruža razne mogućnosti u IT industriji.

Virtuelna realnost se koristi najčešće za razne vrste simulacija i koristila se još od 1970. Najčešća primena izmežu 1970 do 1990 godine su vojne, medicinske ili avio simulacije. Komericjalno virtuelna realnost se širi od 1992. godine. Kompanija SEGA 1994. je objavila konzolu Sega VR-1 koja je koristila 3D grafiku. Kratko nakon toga kompanija Apple pokušava da zaradi na usponu VR tehnologije i objavljuje QuickTime VR koji ima mogućnost prikaze panorama 360 slika.

Od 2010. godine kreće prava revolucija VR tehnologije sa prvim protoipom Oculus Rift. Ubrzo nakon toga 2012. godine kreirana je igra za ovu platformu koja je demonstrirala mogućnosti koje VR tehnologija nudi. Oculus Rift firma kreće Kickstarter kampanju i već 2013. godine šalje prve verzije Oculus Rift razvojne jedinice.

Danas postoje mnogo firmi koje razvijaju sisteme za virtuelnu realnost kao što su Oculus, HTC, Valve, Sony i drugi. Sistemi zahtevaju veoma jake računare i grafičke kartice, zbog toga postoje alternative kao što su Google Cardboard, koji omogućava korisniku korišćenje mobilnog uređaja u kartonskom kućištu sa dva sočiva za postizanje efekta VR doživljaja. Naravno to je dosta različito od profesionalnih sistema koje nude velike firme, ali Cardboard nudi veoma pristupačan ulaz u VR svet, bez velikog ulaganja.

# Vridge

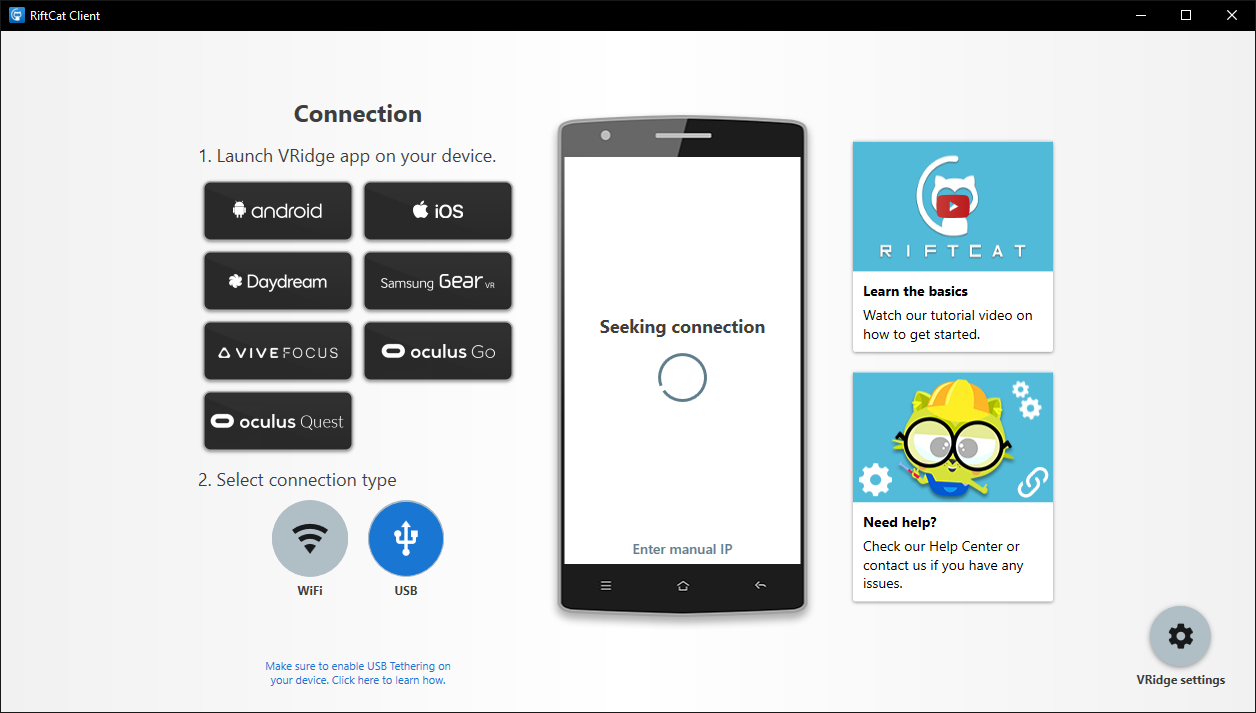
Vridge je softver koji omogućava osobama da bez velikog ulaganja u skupu VR opremu iskuse i igraju poznate VR igre. Korišćenjem mobilnog telefona i VR naočara bez ekrana, gde se umesto ekrana umeće mobilni moguće je iskusiti VR.



Slika 2 – Izgled Google Cardboard naočara za virtuelnu realnost.

Vridge softver razvija firma RiftCat iz Poljske. Postoje dve verzije softvera, besplatna koja omgućava korisniku da igra bilo koju VR igru u sesiji od 10 minuta, kao i plaćena verzija koja omogućava neograničene sesije.

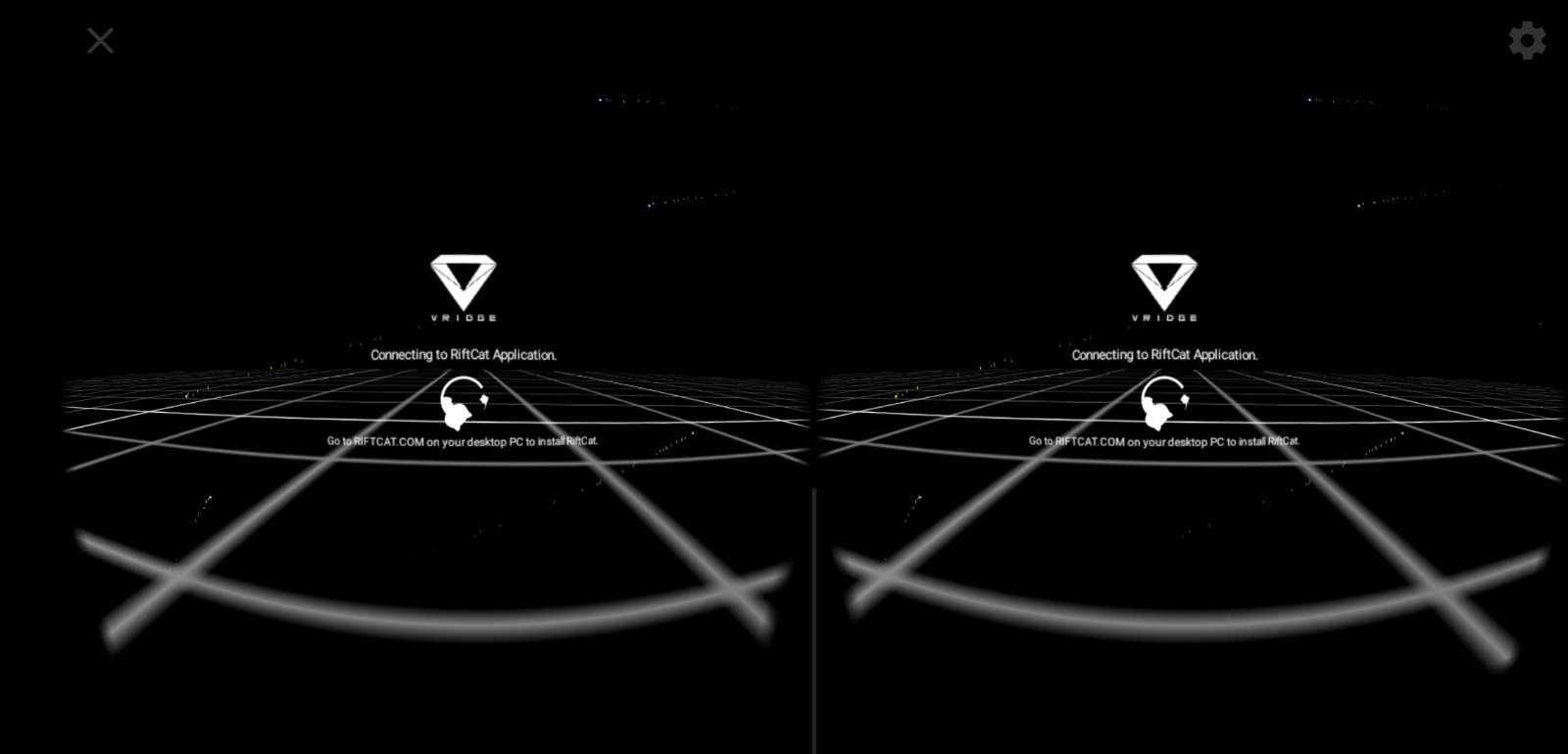
Za potrebe ovog rada kupljena je komercijalna verzija softvera koja omogućava neograničene sesije. Instalacija softvera je veoma jednostavna, potrebno je preuzeti softver sa njihove oficijalne stranice i instalirati. Nakon instalacije softver je spreman i vodi vas kroz čarobnjaka za podešavanje telefona i povezivanje. Na mobilnom telefonu potrebno je takođe preuzeti Vridge aplikaciju sa Google Play prodavnice.



Slika 3 - Izgled RiftCat VRidge aplikacije na računaru.

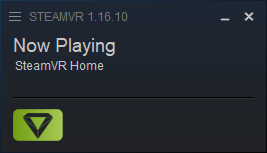
Nakon pokretanja RiftCat Vridge aplikacije potrebno je izabrati način povezivanje aplikacije. Moguće je izabrati bežično ili žičano povezivanje. Preporuka je koristiti žičano USB povezivanje radi kvalitetnije konekcijem, bolje rezolucije i niže latencije. Bežično povezivanje takođe je moguće ali savetuje se korišćenje 5GHz WiFi mreže, ukoliko nemate ruter koji podržava ovu frekvenciju najbolje je povezati putem telefona.

Na telefonu je potrebno pokrenuti Vridge aplikaciju i povezati telefon ili putem USB kabla ili Wifi mreže. Aplikacija na računaru će automatski prepoznati uređaj i povezati ga sa aplikacijom.

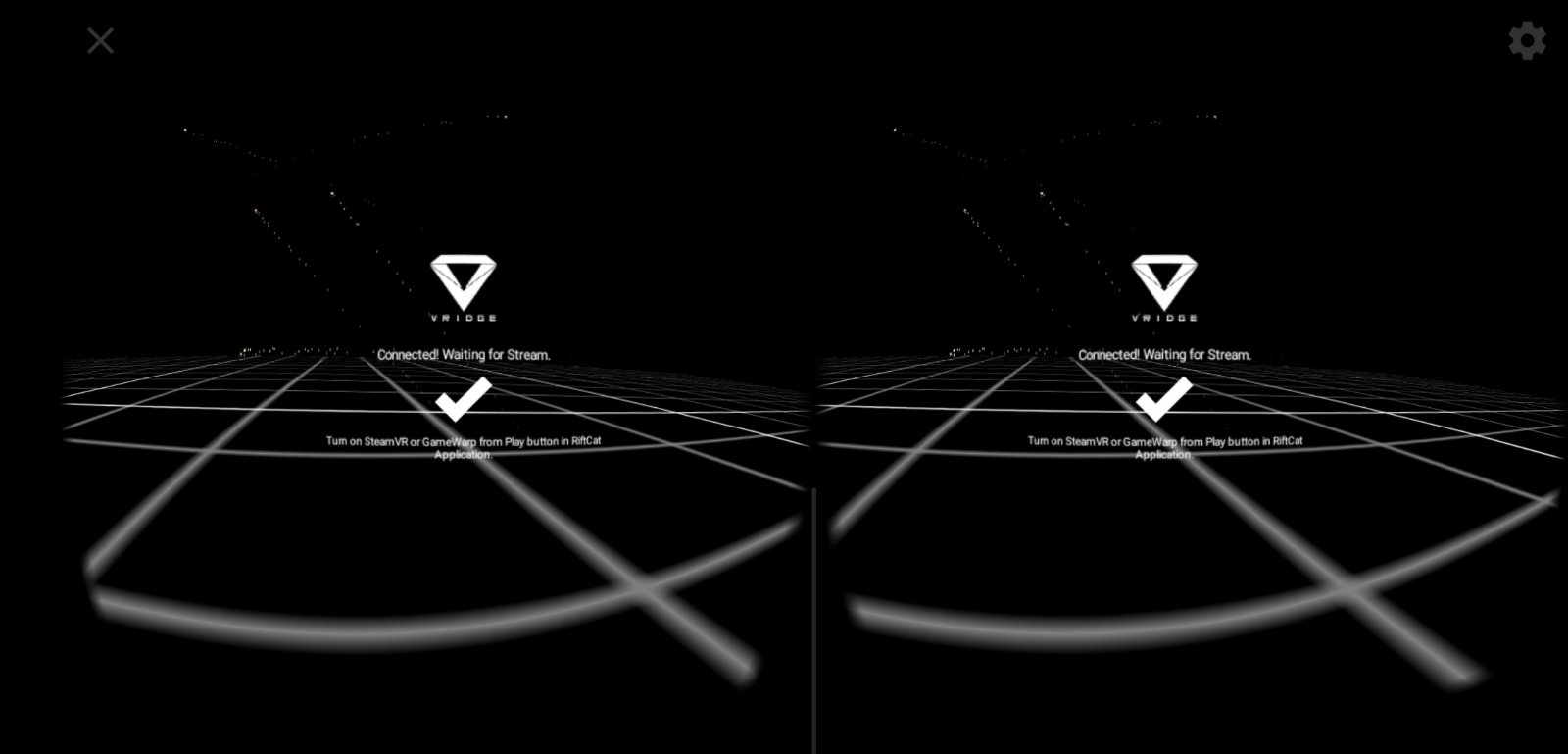


Slika 4 - Izgled VRidge aplikacije na Android uređaju.

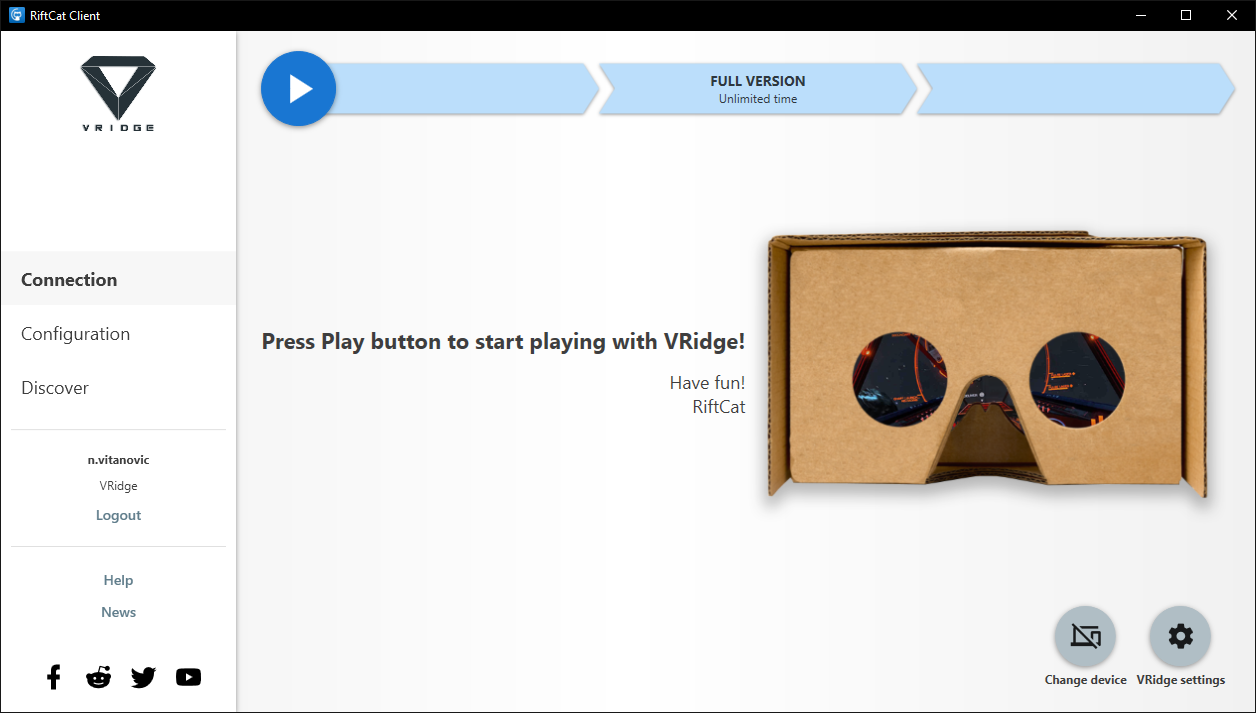
Ukoliko se vrši povezivanje putem USB-a potrebno je na telefonu uključiti opciju *USB debugging* i izabrati opciju *USB tethering*. Nakon prepoznavanja aplikacije na računaru se prikazuje druga forma koja indicira da je telefon uspešno povezan sa Vridge aplikacijom. Na telefonu se takože prikazuje poruka da je veza uspešno uspostavljena. Pritiskom na taster play na aplikaciji pokreće se SteamVR.



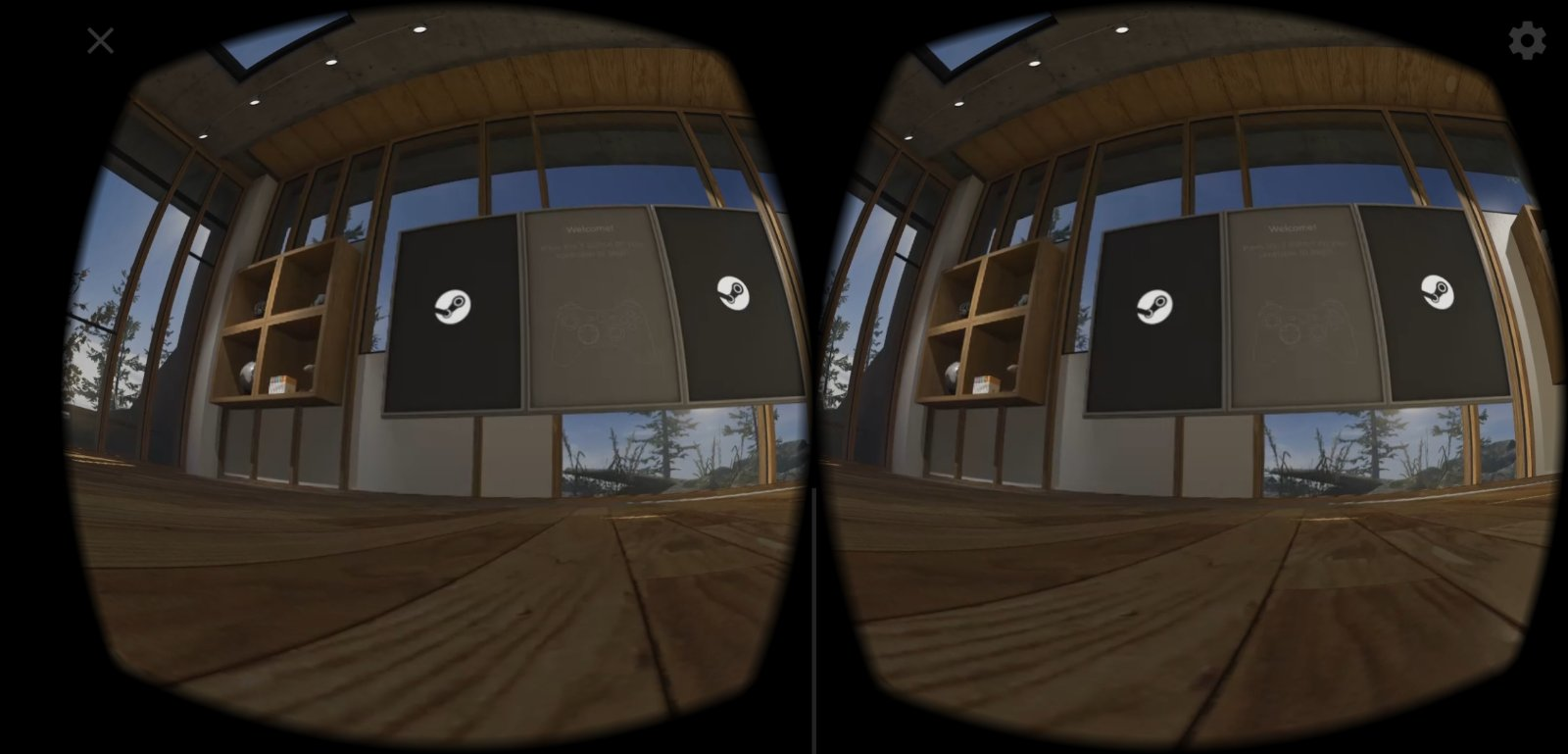
Slika 5 - Izgled SteamVR aplikacije.



Slika 6 - VRidge Android aplikacije je povezana sa računarom.



Slika 7 - RiftCat VRdige Windows aplikacija je povezana sa telefonom.



Slika 8 - Izgled SteamVR početne sobe pre pokretanja VR igre.

# SteamVR

SteamVR predstavlja Valve razvojno okruženje za najpopularniju Game platofrmu Steam. SteamVR pruža programerima da lakše razviju igre koje zahtevaju virtuelnu realnost jer se svi delovi apstrakuju maksimalno moguće. Na primer, čitanje senzora za poziciju glave igrača se automatski obrađuju, bez obzira koji tip VR uređaja koristite. SteamVR se bazira na OpenVR inicijativi koja pokušava da standardizuje razvoj VR aplikacija i igara.

Pre igranja bilo koje VR igre potrebno je preuzeti SteamVR ili će se on automatski preuzeti ukoliko pokrećete neku SteamVR igru. Nakon instalacije i pokretanja moraćete proći kroz čarobnjaka za kalibraciju, tek nakon kalibracije vašeg VR uređaja pojavljuje se dijalog kao na slici 5, dok na slici 8 možete videti početni meni kroz VR naočare.

# Unity SteamVR dodatak

Vavle firma održava dodataka za Unity razvojno okruženje koji olakšava intrakciju sa SteamVR. Moderan SteamVR Unity dodatak je zadužen za tri glavne stavke:

* učitavanje 3D modela za VR kontrolere;
* upravljanje VR kontrolerima;
* odlučivanje o izgledu vaših VR ruku u zavisnosti od kontrolera.

Pored ove tri stavke SteamVR Unity dodatak takođe poseduje primere interakcije u okviru Unity igara pomoću SteamVR platoforme, koji mogu da pomognu programeru u bržem razvoju VR igara.

Za korišćenje SteamVR Unity dodatka potrebno je da imate instaliran SteamVR, poželjno Beta verziju zbog novih funkcionalnosti.

Dodatak se može preuzeti sa Unity prodavnice i potpuno je besplatan za korišćenje kako u akademske tako i komercijalne svrhe.

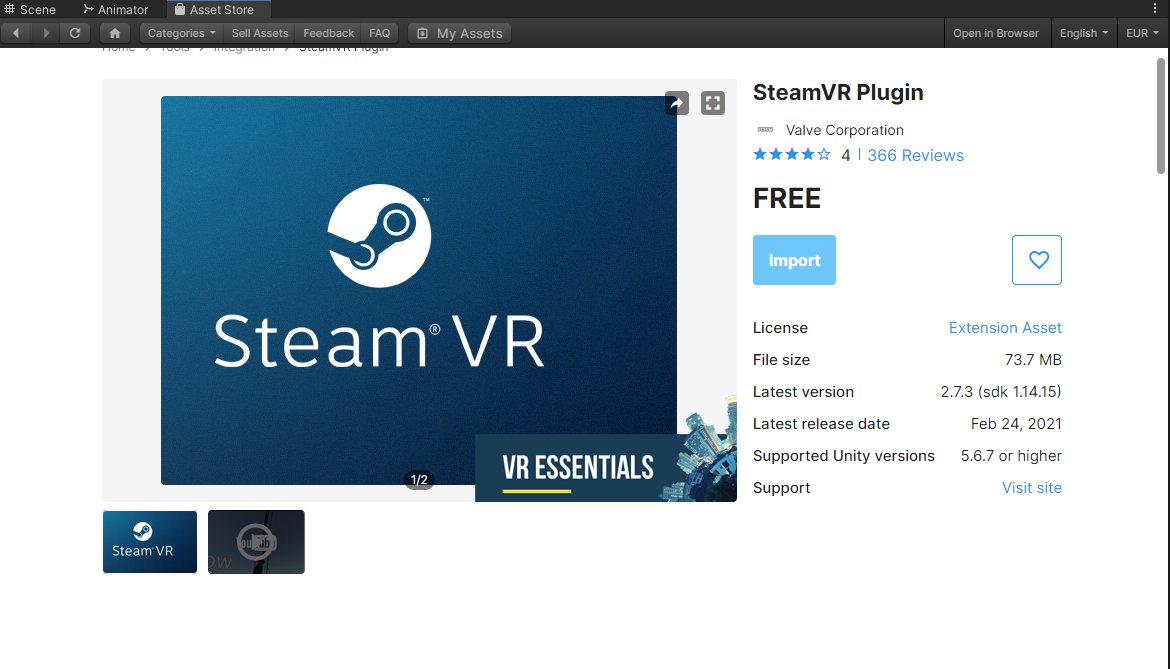
# Projekat

Ideja aplikacije je da pokaže integraciju SteamVR Unity dodatka sa Vridge aplikacijom. Sama aplikacija radi na Windows računaru dok se VR prenosi na Vridge aplikaciju na telefonu.

Cilj igre je da korisnik pređe iz jedne sobe u drugu, slično konceptu escape room, rešavanjem kratkih zagonetki. Igra se sastoji iz 3 nivoa i glavnog menija. Igrač je odmah od menija u virtuelnom svetu.

## Podešavanje

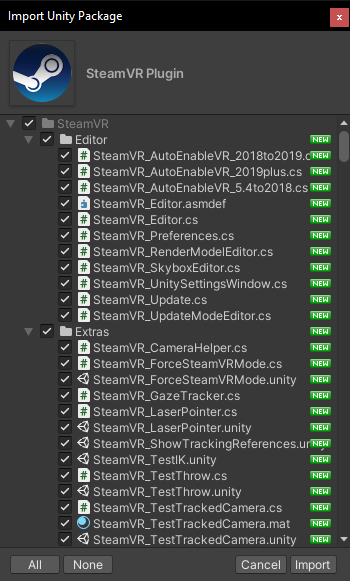
Za kreiranje VR igre potrebno je izabrati 3D projekat u Unity okruženju i dodati SteamVR dodatak.



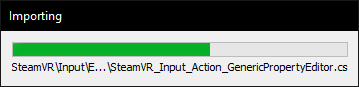
Slika 9 - Uvoz SteamVR dodatka iz Unity prodavnice.

Pritiskom na dugme import sa slike 9. prikazuje se dijalog na slici 10. Dijalog pokazuje dodatne fajlove koji će biti dodati u postojeći projekat. Pritiskom na dugme *Import* uvoz SteamVR dodatka počinje i prikazuje se prozor sa slike 11 koji prikazuje dokle je Unity stigao sa uvozom dodatka. Nakon par minuta pojviće se prozor sa slike 12 koji nudi izbor dva tipa VR podešavanja (za potrebe projekta biramo *Unity XR*):

* Legacy VR – stariji način izrade VR aplikacija.
* Untiy XR – novi pristup izrade VR aplikacija koji podržava veliki broj platformi.



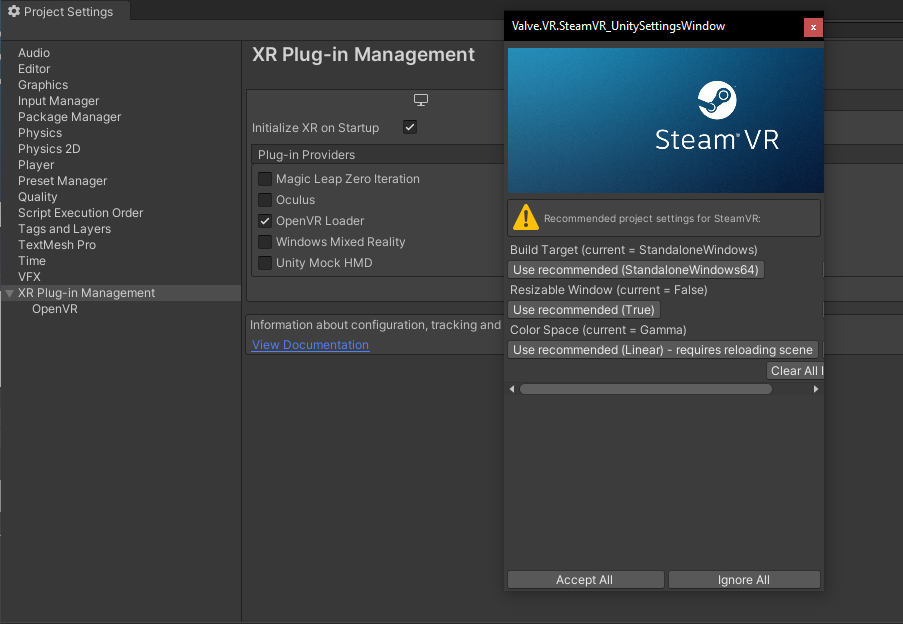
Slika 10 - Uvoz SteamVR dodatka.



Slika 11 - Uvoz dodatka je u toku.



Slika 12 - Izbor tipa VR projekta.

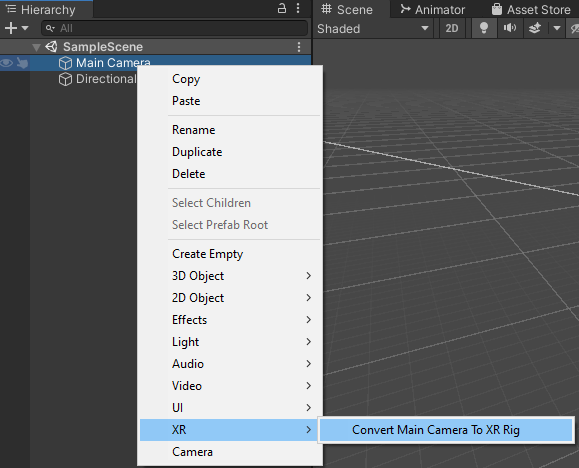


Slika 13 - Početna podešavanja Unity XR aplikacije sa SteamVR integracijom.

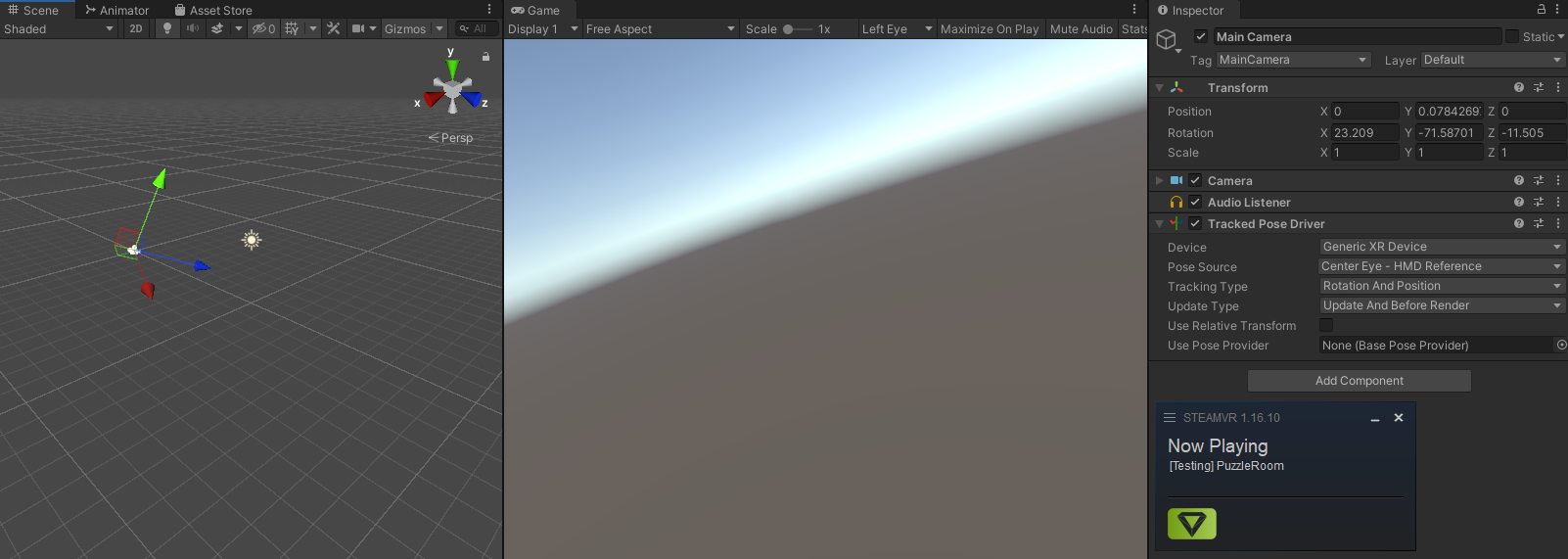
Konačno na slici 13 prikazuju se dva prozora, potrebno je prihvatiti sve predloge SteamVR dodatka pritiskom na dugme *Accept All* time su podrazumevana podešavanja postavljena u projektu i aplikacija je skoro spremna za VR naočare kroz SteamVR.

Poslednji korak predstavlja konvertovanje kamere u XR kameru. Desnim klikom na glavnu kameru u aplikaciji izborom podmenija *XR -> Convert Main Camera to XR Rig* kao na slici 14kamera biva pretvorena u XRRig objekat.

Nakon pokretanja igre klikom na dugme Play u Unity editor SteamVR se pokreće i naša igra se prikazuje na VRdige aplikaciji na Android telefonu. Pomeranjem mobilnog telefona pomera se rotacija kamere u igri. Za prenos rotacije telefona u rotaciju kamere zadužena je komponenta T*racked Pose Driver* koja je dodeljena objektu *Main Camera.*



Slika 14 - Pretvaranje kamere u XR kameru.



Slika 15 - Pokrenuta igra i prikaza pomerene kamere na osnovu lokacije VR naočara.

## Struktura projekta

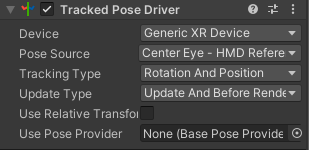
## XRRig objekat

*XRRig* objekat predstavlja igrača. Najbintiji deo *XRRig* objekta predstavlja glavna kamera, koja prikazuje svet igraču kroz VR naočare. Struktura objekta je kao na slici.



Slika 16 - Struktura XRRig objekta.

Podešavanja vezana za VR prikaz se nalaze u okviru glavne kamere. Komponenta koja je zadužena za VR prikaz je *Tracked Pose Driver*, on sadrži podešavanja VR naočara i kako se ona odnose na virtuelni svet.



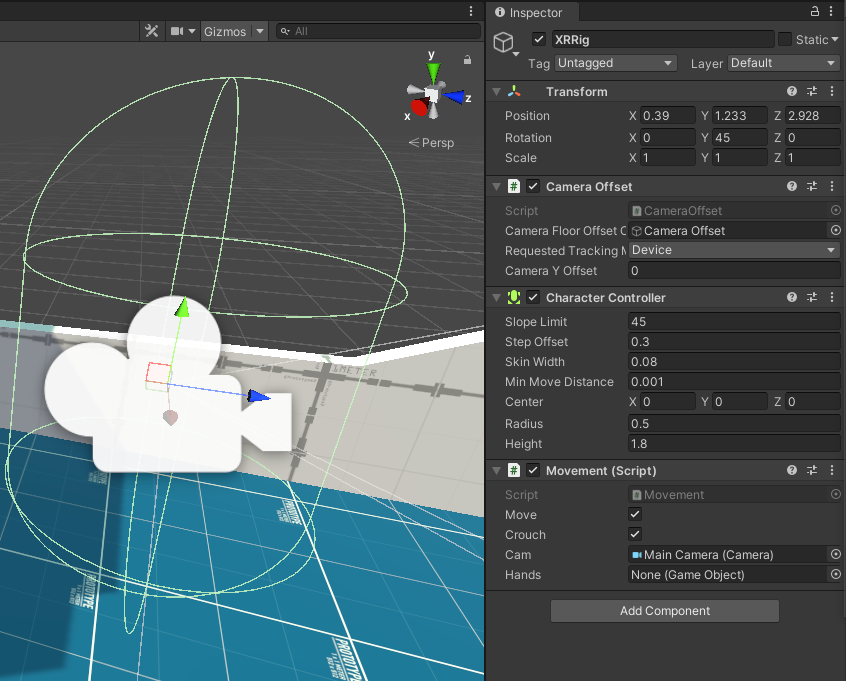
Slika 17 - Podešavanja Tracked Pose Driver komponente.

* *Device* – predstavlja uredjaj kojim se vrši praćenje. U našem slučaju to su VR naočare ali mogao bi da bude kontroler ili neki drugi daljinski uređaj.
* *Pose Source* – predstavlja poziciju uređaja, najčešće se koristi opcija kao na slici koja prestavlja tačku između očiju. Kamera u virtuelnom svetu bi srazmerno tome trebalo da bude postavljena.
* *Update type* – kada se vrši čitanje informacija o VR uređjaju.

Objekat *Camera Offse*t predstavlja pomeraj kamere od zemlje, u našem slučaju VR naočare ne podržavaju relativno kretanje tj. ovaj offset nema uticaj, zbog toga je fiksiran na vrednost 0 i sama glavna kamera je pomerena na željenu visinu u okviru *XRRig* objekta.

### Character controller

*Character controller* komponenta predstavlja telo igrača. Telo je u obliku kapsule i automatski dobija svojstva 3D objekta koji reaguje na koliziju sa ostalim 3D objektima. Ova komponenta se dodaje na *XRRig* objekat. Sadrži podešavanja visine, oblika objekta, kao i karakteristika o kretanju objekta.



Slika 18 - Izgled Character Controller komponente na XRRig objektu sa osnovnim podešavanjima.

### Kretanje

Kretanje u VR svetu prestavlja izazov. Radi postizanja najrealnijeg doživljaja VR igara, potrebno je translirati fizičko kretanje u stvarnom svetu u virtuelno kretanje. Da bi tako nešto bilo moguće morali bismo da imamo neogračien prostor po kome bi se kretali. Naravno to nije moguće, iz tog razloga razni proizvođači igara su smislili razne načine kretanja u virtuelnom svetu.

Kretanje u ovoj igri je jednostavno, gledanjem blago na dole u realnom svetu translira se na kretanje igrača u virtuelnom svetu u smeru gledanja. Gledanjem pod većim uglom prema podu igrač prelazi u mod čučanja i sporije se kreće u smeru gledanja. To je omogućeno kodom iz skripte *Movement.*cs koja je zakačena za *XRRig* objekat i omogućava mu pomeranje.

public class Movement : MonoBehaviour

{

private CharacterController controller;

private float playerSpeed = 1.0f;

public bool move = true;

public bool crouch = true;

public Camera cam;

public GameObject hands;

private float originalHeight;

private float crouchHeight;

void Start()

{

controller = GetComponent<CharacterController>();

originalHeight = controller.height;

crouchHeight = originalHeight / 4;

}

void Update()

{

//Check for angle for movement or crouch

if(cam.transform.localEulerAngles.x > 40 && cam.transform.localEulerAngles.x < 60)

{

move = true;

crouch = false;

}

else if(cam.transform.localEulerAngles.x > 75 && cam.transform.localEulerAngles.x < 85)

{

crouch = true;

}

else

{

move = false;

crouch = false;

}

// Actual movement

if (move)

{

Vector3 moveDistance = Camera.main.transform.forward;

controller.SimpleMove(moveDistance \* playerSpeed);

}

else if(crouch)

{

controller.height = crouchHeight;

controller.SimpleMove(Camera.main.transform.forward);

}

else

{

controller.height = originalHeight;

}

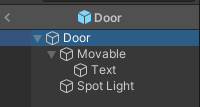
}

}

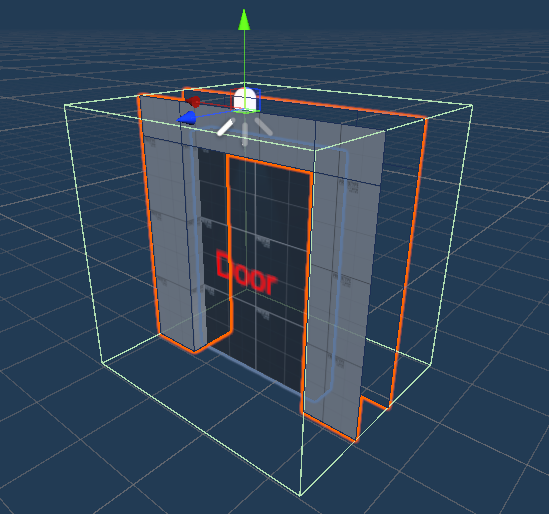
Deo koda 1 - Skripta za kretanje igrača Movement.cs.

## Vrata

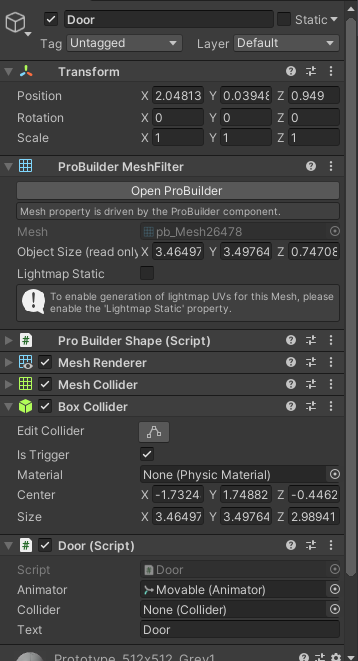
Vrata u svakoj igri predstavljaju osnovnu mehaniku. Vrata su složen objekat koji se sastoji iz nekoliko komponenti. Fiksna komponenta su sam ram vrata *Door* dok sama vrata prestavlja objekat pod nazivom *Movable*.



Slika 19 - Struktura objekta vrata.



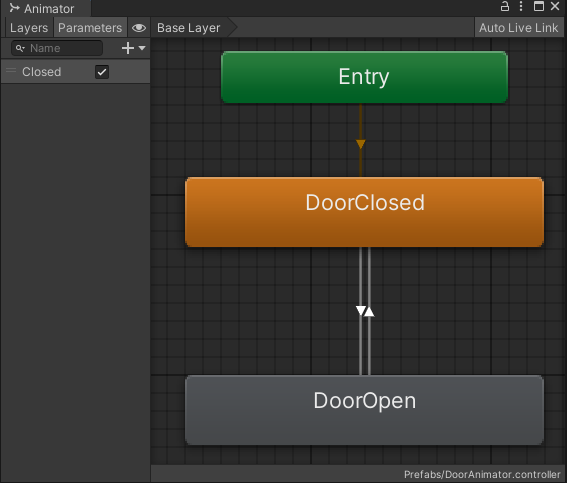
Slika 20 - Izgled vrata u Unity editoru.



Slika 21 - Komponente od kojih su vrata sačinjena.

Komponenta BoxCollider je služi za detektovanja prisustva objekta blizu vrata. U slučaju da se bilo koji objekat uđe u zonu za otvaranje vrata, pokreće se animacija za otvaranje vrata, podešavanjem stanja *Closed* na *false* u okviru animacionog kontrolera. Animacioni kontroler je postavljen na deo vrata koji se pomera tj. na *Moveable* objekat.

Na početku vrata se nalazue u zatvorenom položaju tj. stanju *DoorClosed* animacija tog stanja su zatvorena vrata. Iz tog stanja je moguće preći u stanje *DoorOpen* koje predstavlja otvaranje vrata i reprodukuje animaciju za otvaranje vrata.



Slika 22 - Animacioni graf koji omogućava otvaranje i zatvaranje vrata.

Otvaranje i zatvaranje vrata ne bi bilo moguće bez skripte *Door.cs* koja se stavlja na glavni objekat *Door* i uz njega se povezuje deo vrata koji se pomera. Pored animatora moguće je podesiti i tekst koji se prikazuje na vratima.

public class Door : MonoBehaviour

{

// Start is called before the first frame update

public Animator animator;

public Collider collider;

public string text;

private TextMesh textMesh;

void Start()

{

textMesh = GetComponentInChildren<TextMesh>();

textMesh.text = text;

}

private void OnTriggerEnter(Collider other)

{

animator.SetBool("Closed", false);

}

private void OnTriggerExit(Collider other)

{

animator.SetBool("Closed", true);

}

}

Deo koda 2 - koji omogućuje otvaranje i zatvaranje vrata.

## Dugme

Dugme omogućava aktiviranje određenih interaktivnih komponenti. Pritiskanje dugmeta se vrši gledanjem do 3 sekunde. Za detektovanje objekata koji su interaktivni koristi se tehnika *ray tracing* koja projektuje zamišljenu liniju od centra kamere ka objektu prema kome igrač gleda. Princip je sličan laseru, gde bi tačka lasera na objektu predstavljala gde igrač gleda dok je sam laserski zrak ta zamišljena linija. Unity podržava funkcije za projektovanje tog zamišljenog zraka, on sam po sebi se ne vidi u igri, ali je moguće prikazati ga u modu debagiranja.

Objekti kojima možemo interagovati na sceni korišćenjem principa *ray tracing* implementiraju interfejs *Iinteractable.* Takođe objekti koji sadrže neku prostu akciju kao vrata takođe implementiraju pomenuti interfejs.

public interface IInteractable

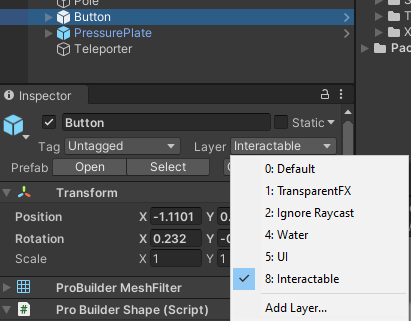
{

bool Interact();

}

Deo koda 3 - interfejs koji omogućava interakciju sa objektima.

Kako bi princip *ray tracing* funkcionisao potrebno je dodati interakcioni kod *XRRig* komponenti. Interakciona skripta poziva funkciju *Interact()* nakog određenog vremena gledanja u željeni objekat. Takođe promena boje objekta ka kome korisnik gleda vizualizuje korisniku koji je objekat trenutno aktivan. Bitno je navesti da objekti kojima korisnik može da interaguje se nalaze u specijalnom sloju pod brojem 8. Veoma je bitno imati ovaj sloj kako se interakcija ne bi pokušavala sa svakim objektom sa scene.



Slika 23 - poseban sloj za interakciju sa komponentama ka kojima igrač gleda.

public class Interaction : MonoBehaviour

{

private Camera playerCamera;

private float totalLookingTime;

private GameObject lastObject;

private Color previousObjectColor;

// Start is called before the first frame update

void Start()

{

playerCamera = GetComponentInChildren<Camera>();

lastObject = null;

}

// Update is called once per frame

void Update()

{

var ray = playerCamera.ViewportPointToRay(new Vector3(0.5F, 0.5F, 0));

RaycastHit hit;

if (Physics.Raycast(ray, out hit, 1.5f) && hit.collider.gameObject.layer == 8)

{

print("I'm looking at " + hit.transform.name);

var objectRenderer = hit.collider.gameObject.GetComponent<Renderer>();

if (lastObject != hit.collider.gameObject)

{

totalLookingTime = 0;

if (lastObject != null)

lastObject.GetComponent<Renderer>().material.color = previousObjectColor;

previousObjectColor = hit.collider.gameObject.GetComponent<Renderer>().material.color;

}

else

{

totalLookingTime += Time.deltaTime;

}

if(totalLookingTime > 3)

{

lastObject.GetComponent<Renderer>().material.color = previousObjectColor;

hit.collider.gameObject.GetComponent<IInteractable>().Interact();

totalLookingTime = 0;

}

else

{

//highlight the object

objectRenderer.material.color = Color.yellow;

}

// save the last object that the player was looking at for comparison

lastObject = hit.collider.gameObject;

Debug.DrawRay(playerCamera.transform.position, playerCamera.transform.forward, Color.green);

}

else

{

totalLookingTime = 0;

print("I'm looking at nothing!");

if(lastObject != null)

lastObject.GetComponent<Renderer>().material.color = previousObjectColor;

Debug.DrawRay(playerCamera.transform.position, playerCamera.transform.forward, Color.red);

}

}

}

Deo koda 4 - Skripta koja omogućava interakciju igrača sa okolinom.

Dok sam kod dugmeta na sceni predstavlja jednostavnu implementaciju pomenutog interfejsa i pozivanje objekta kojim se interaguje nakon pritiska dugmeta. Ovim se omogućuje dinamičko kreiranje scene korišćenjem predefinisanih objekata koji nasleđuju ovaj interfejs i dodaju dodatnu funkcionalnost.

public class Button : MonoBehaviour, IInteractable

{

public bool pressed = false;

public string text = "";

public GameObject objectToTrigger = null;

private Animator animator;

private Text textObj;

public bool Interact()

{

//Call the next item

if (objectToTrigger != null)

objectToTrigger.GetComponent<IInteractable>().Interact();

return pressed = !pressed;

}

// Start is called before the first frame update

void Start()

{

animator = GetComponent<Animator>();

textObj = GetComponentInChildren<Text>();

textObj.text = text;

}

// Update is called once per frame

void Update()

{

if(pressed)

{

animator.SetBool("pressed", true);

}

else

{

animator.SetBool("pressed", false);

}

textObj.text = text;

}

}

Deo koda 5 - Primer koda dugmeta.

Animirnaje objekta je odrađeno na isti način kao i animacija vrata opisanih u prethodnoj sekciji.

# Zaključak

# Reference