Izrada virtualne stvarnosti - K34

Rogina, Marko

Sveučilište
Sjever

Završni rad br. 570/MM/2018

Izrada virtualne stvarnosti – K34

Marko Rogina, 0815/336

Varaždin, rujan 2018. godine

Sveučilište
Sjever
Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 570/MM/2018

Izrada virtualne stvarnosti – K34

Student

Marko Rogina, 0815/336

Mentor

dr. sc. Andrija Bernik, pred.

Varaždin, rujan 2018. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za multimediju, oblikovanje i primjenu

PRISTUPNIK: Marko Rogina | MATIČNI BROJ: 0815/336

DATUM: 16.3.2018. | KOLEGIJ: Video animacija

NASLOV RADA: Izrada virtualne stvarnosti - K34

NASLOV RADA NA ENGL. JEZIKU: Creation of virtual reality of the room K34

MENTOR: dr.sc. Andrija Bernik | ZVANJE: Predavač

ČLANOVI POVJERENSTVA:
1. mr.sc. Dragan Matković, v. pred. - predsjednik
2. doc.art. Robert Geček - član
3. dr.sc. Andrija Bernik, pred. - mentor
4. Nikolina Bolčević Horvatić, dipl.ing. - zamjenski član
5.

Zadatak završnog rada

BROJ: 570/MM/2018

OPIS

Unity je game engine koji se prvenstveno koristi za razvoj dvodimenzionalnih i trodimenzionalnih videoigara i simulacija na računalima, konzolama i mobilnim uređajima. Razvijen je of tvrtke Unity Technologies. Podržava 2D i 3D grafiku, funkciju povlačenja i ispuštanja te rađenje skripti pomoću jezika C#.
U ovom radu prikazat će se izrada 3D modela i textura kroz Autodesk Maya program.
Prikazat će se cjelokupan tijek izrade projekta virtualne stvarnosti pa sve do generiranje PC aplikacije u Unity okruženju.
Cilj rada je prikazati mogućnosti Unity alata kao i postupke nužne za kreiranje virtualne stvarnosti.

U radu je potrebno:
- objasniti teorijske koncepte za izradu virtualne stvarnosti.
- prikazati postupke za modeliranje i teksturiranje osnovnih elemenata.
- objasniti korisničko sučelje Unity-a.
- objasniti sustave osvjetljavanja u Unity-u.
- objasniti interaktivnost korisnika i kontrola (tipkovnica, miš).
- izraditi VR PC aplikaciju koja se odnosi na kabinet K34.

ZADATAK URUČEN: 14.3.2018.

REPUBLIKA HRVATSKA SVEUČILIŠTE SJEVER KOPRIVNICA

POTPIS MENTORA

Predgovor

Tema ovoga završnog rada odabrana je zbog strasti prema videoigrama. Cilj ovog rada bio je steći nova znanja u području 3D modeliranja, teksturiranja i virtualne stvarnosti te prikazati mogućnosti Unity alata, kao i postupke nužne za kreiranje virtualne stvarnosti. Također, cilj je bio prikazati cjelokupan tijek izrade projekta virtualne stvarnosti pa sve do generiranja PC (*Personal Computer*) aplikacije u Unity okruženju.

Ovim putem zahvaljujem mentoru, dr. sc. Andriji Berniku, koji je cijelo vrijeme nadzirao proces izrade te mi svojim savjetima pomogao u rješavanju prepreka koje su nastale tijekom izrade završnog rada. Također, zahvaljujem svojoj obitelji na podršci tijekom studiranja.

Sažetak

U ovom radu prikazani su procesi koje je potrebno izvršiti kako bi se napravila aplikacija za osobna računala u virtualnoj stvarnosti. U prvom poglavlju je uvod gdje će se čitatelja detaljnije upoznati s predmetom, svrhom i ciljevima završnog rada. U drugom poglavlju reći će se osnovno o 3D modeliranju, a u trećem općenito o teksturiranju. U četvrtom poglavlju objasnit će se pojam virtualne stvarnosti. Također, pojašnjavaju se i teorijski koncepti za izradu virtualne stvarnosti. U petom poglavlju objasnit će se korisničko sučelje Unityja. U šestom poglavlju govorit će se o sustavu osvjetljavanja u Unityju. U sedmom poglavlju objasnit će se postupak izrade završnog projekta. U osmom poglavlju donosi se zaključak. U devetom poglavlju je navedena literatura.

Ključne riječi: 3D modeliranje, Autodesk Maya, virtualna stvarnost, Unity, aplikacija.

Abstract

This paper presents the processes that need to be performed to make the applications for personal computers in the virtual reality. First chapter contains an introduction where readers will get more familiar with the subject, purpose, and goals of the final project. In the second chapter, we will discuss the basics of a 3D modeling and in third we will say something in general about texturization. The fourth chapter will explain the concept of virtual reality. It also explains theoretical concepts for creating virtual reality. Chapter five will explain the Unity User Interface. Chapter six will discuss the system of illumination in Unity. The seventh chapter will explain the process of making the final project. The eighth chapter gives the conclusions of this study. Chapter nine contains the literature.

Keywords: 3D modeling, Autodesk Maya, Virtual Reality, Unity, application.

Popis korištenih kratica

PC Personal Computer
Osobno računalo

VR Virtual Reality
Virtualna stvarnost

Sadržaj

1. Uvod

Virtualna stvarnost danas je sve aktualniji pojam. Iako virtualna stvarnost još nije dosegla svoj vrhunac te je još u razvitku i usavršavanju, ona se koristi. Virtualna stvarnost je umjetno okruženje koje je stvoreno pomoću softvera i predstavljeno korisniku na takav način da korisnik prihvaća to okruženje kao pravi okoliš. Na računalu se virtualna stvarnost prvenstveno doživljava kroz dva od pet osjetila: vid i zvuk. [1] Kada se govori o virtualnoj stvarnosti, najčešće se misli na videoigre budući da su videoigre primarna primjena tehnologije virtualne stvarnosti. No virtualna stvarnost koristi se i u druge svrhe, kao što su zdravstvena industrija, marketing, obrazovanje, vojska, sport i simulacije raznih vježbi. [2]

U ovom radu fokus će biti na izradi virtualne stvarnosti sobe K34. Za to će nam biti potrebno nekoliko programa, kao što su Autodesk Maya, Unity i Microsoft Visual Studio. Autodesk Maya koristit će se za 3D modeliranje i teksturiranje 3D modela koji će biti na našoj sceni. Autodesk Maya jedan je od vodećih programa za trodimenzionalno modeliranje koji je razvila tvrtka Autodesk. Maya pruža paket alata za stvaranje 3D sadržaja, od modeliranja, animacije, dinamike do renderiranja. [3] Unity će se koristiti za uvođenje Maya scene koju smo prethodno napravili te za dovršetak aplikacije. Unity je „game engine“ koji se prvenstveno koristi za razvoj dvodimenzionalnih i trodimenzionalnih videoigara te simulacija na računalima, konzolama i mobilnim uređajima. Razvila ga je tvrtka Unity Technologies. [4] Podržava 2D i 3D grafiku, funkciju povlačenja i ispuštanja te kreiranje skripti pomoću jezika C#. [5] U Unityju ćemo objasniti sustave osvjetljavanja, interaktivnost korisnika i kontrola te način izvoza iz Unityja u gotovu PC aplikaciju. Microsoft Visual Studio koristit će se za pisanje skripti potrebnih za aplikaciju. Microsoft Visual Studio je interaktivno razvojno okruženje tvrtke Microsoft. Koristi se za pregled, uređivanje koda, ispravljanje, izgradnju i objavljivanje aplikacije. [6]

U radu će biti prikazan cjelokupan tijek izrade projekta. Detaljno će biti objašnjen svaki korak prilikom modeliranja, teksturiranja, uvođenja scene u Unity, postavljanje kamere i osvjetljenja, apliciranje skripta na pojedine objekte, interaktivnost korisnika i kontrola, generiranje gotove aplikacije. Cilj rada je prikazati mogućnosti Unity alata, kao i postupke nužne za kreiranje virtualne stvarnosti.

2. 3D modeliranje

3D modeliranje je proces stvaranja 3D prikaza bilo koje površine ili objekta manipuliranjem poligona, rubova i vrhova u simuliranom 3D prostoru. [7] Koristeći 3D, moguće je napraviti veličinu, oblik i teksturu stvarnog ili zamišljenog objekta.

Prvi 3D modeli nastali su 1960-ih. Tada su u 3D modeliranje bili uključeni samo oni stručnjaci iz područja računalnog inženjerstva i automatizacije koji su radili s matematičkim modelima i analizom podataka. Pionir 3D grafike je Ivan Sutherland, tvorac Sketchpada. Ovaj revolucionarni program pomogao je stvaranju prvih 3D objekata – 3D jest ono što je danas zahvaljujući Sketchpadu. U početku su 3D modeliranje i animacija uglavnom korišteni na televiziji i u oglasima, ali s vremenom je njihova prisutnost u drugim područjima života uvelike povećana. [8]

3D modeliranje se koristi u mnogim različitim industrijama uključujući virtualnu stvarnost, videoigre, 3D tisak, marketing, televiziju i filmove, znanstvenu i medicinsku fotografiju te računalno potpomognuto dizajniranje. 3D modeliranjem softver generira model kroz razne alate i pristupe uključujući: [9]

– jednostavne poligone

– 3D primitivne oblike – jednostavni oblici poligona poput piramida, kockica, kuglica, cilindara i čunjeva

– klinaste krivulje

– NURBS (*non-uniform rational b-spline*) – glatke oblike koji definiraju obrubne krivulje koje su relativno računalno kompleksne

– 2D geometrijski poligonski oblici koriste se u velikoj mjeri u filmskim efektima i 3D videoigrama.

Kod 3D modeliranja gotovo je uvijek dobra ideja početi s jednostavnim te to nadograđivati prema kompleksnijim stvarima. 3D modeliranje zahtijeva precizan tijek rada koji često uključuje pažljivi položaj pojedinih vrhova kako bi se postigle ispravne konture željenog objekta. [10]

Na tržištu postoje brojni softverski programi za 3D modeliranje. Neki od najpoznatijih softvera za 3D modeliranje navedeni su ovdje: Autodesk Maya, AutoCAD, ZBrush, 3DS Max, Blender, SketchUp. [7] 3D modeliranje je bitan dio modernoga digitalnog doba. Također je uzbudljivo i nagrađujuće sredstvo umjetničkog samoizražavanja. [10]

3. Teksturiranje

Teksturiranje 3D modela je proces kojim se oživljava model, daje mu se boja i svojstvo. [11] Prvi i najvažniji cilj korištenja teksture je pokazati materijal iz kojeg se stvara stvarni objekt, na primjer za prikaz cigle iz koje se zid izrađuje u računalnoj igri. Ponekad se 3D teksturiranje koristi za određena fizička svojstva modela objekta, primjerice glatkoću ili hrapavost. Korištenje teksture omogućuje izradu različitih svjetlosnih efekata, kao što su refleksija i sjene. 3D teksture omogućuju stvaranje različitih manjih objekata na površini modela, na primjer bore, ožiljke, pukotine. Različite vrste tekstura i mapiranja koriste se kako bi slika ili 3D model bili realniji. Ovaj proces je vrlo važan za stvaranje dobrog 3D modela. [12] Za dobro teksturiranje poželjno je koristiti dodatne programe za obradu fotografija kako bi model dobio karakteristike koje su potrebne u skladu s namjenom modela. [11]

Postoji nekoliko načina teksturiranja 3D modela i svaki način ima svoje prednosti i nedostatke. Ovisno o kompleksnosti i detaljnosti 3D modela te o namjeni samog 3D modela izvodi se detaljnost i preciznost teksturiranja. Na samom početku teksturiranja odabire se tip materijala koji se postavlja na model. Osnovni tipovi materijala su: [11]

1) lambert – mat materijal bez sjaja

2) blin – reflektirajući, sjajni materijal

3) anisotropic – postepen prijelaz boja

4) phong – stakleni, sjajni materijal s oštrim osvjetljenjem

5) ramp – materijal kojim se kontroliraju prijelazi između više boja i tekstura, mijenjanjem svjetla i kuta pogleda.

Nakon odabira tipa materijala odabire se tehnika teksturiranja. Postoji nekoliko tehnika teksturiranja: [11]

1) UV teksturne koordinate ili UV točke – dvodimenzionalne koordinate koje se razmještaju s „vertex“ komponentama i dijele informacije za poligonalne i „subdivision“ plohe mreže. UV točke kontroliraju položaj teksturne mape na 3D modelu. UV točke postoje da bi definirale dvodimenzionalni teksturni koordinatni sistem koji se zove UV teksturni prostor, a smjerovi u 2D prostoru označavaju se slovima U i V.

2) UV mapiranje – Proces izrade određenih UV točaka za mrežu ploha modela. UV mapiranje je proces gdje se izrađuju i uređuju UV točke koje se pojavljuju kao ravne, dvodimenzionalne mreže ploha na dvodimenzionalnoj slici koja se koristi kao tekstura i pojavljuje se u UV Texture Editoru.

4. Virtualna stvarnost

Pojam virtualne stvarnosti, naravno, dolazi od riječi „virtualna“ i „stvarnost“. Definicija „virtualnog“ je 'blizu', a „stvarnost“ je ono što mi doživljavamo kao ljudska bića. Dakle, pojam „virtualna stvarnost" zapravo znači 'blizu stvarnosti'. To bi, naravno, moglo značiti samo ono što se obično odnosi na određenu vrstu imitacije stvarnosti. [13]

Sve što znamo o našoj stvarnosti dolazi putem naših osjetila. Drugim riječima, naše cjelokupno iskustvo stvarnosti jednostavno je kombinacija osjetilnih informacija i našeg mozga koji ima mehanizam za stvaranje smisla za tu informaciju. Stoga je jasno da ako možete predstaviti svoja osjetila s informacijama koje su izmišljene, vaša percepcija stvarnosti također bi se mijenjala kao odgovor na nju. Bila bi vam predstavljena verzija stvarnosti koja zapravo nije tamo, ali iz vaše perspektive to bi se shvatilo kao stvarno, odnosno nešto što bismo nazvali virtualnom stvarnošću. Dakle, virtualna stvarnost je pojam koji se koristi za opisivanje trodimenzionalnoga, računalno generiranog okruženja koje osoba može istražiti i raditi interakciju s tim okruženjem. Ta osoba postaje dio virtualnog svijeta ili je uronjena u to okruženje i dok se nalazi u tom okruženju, osoba može manipulirati objektima ili izvesti niz akcija. [13]

Danas se virtualna stvarnost obično provodi korištenjem računalne tehnologije. Postoji niz sustava koji se koriste u tu svrhu, kao što su slušalice, kaciga s unutarnjim zaslonom ili rukavice opremljene senzorima. To se sve zajedno koristi za stimuliranje naših osjetila kako bismo stvorili iluziju stvarnosti. U praksi je to teže napraviti nego što zvuči budući da su naša osjetila i mozak razvijeni kako bi nam pružili usklađeno i posredovano iskustvo. Ako je bilo što imalo neobično, tada tu promjenu obično možemo osjetiti. [13]

Virtualna stvarnost (VR) dominirala je tehnološkim naslovima posljednjih godina s mogućnošću uranjanja svojih korisnika u virtualni, ali sigurni svijet. „Gaming“ je jedna od poznatijih primjena virtualne stvarnosti, ali njezin potencijal ne prestaje tamo. [14]

Područja na koja se može primijeniti VR tehnologija su: vojska, zdravstvo, obrazovanje, poslovanje, sport, telekomunikacije, graditeljstvo, filmska industrija. [15] Također, na mjestima gdje je nešto preopširno, skupo ili nepraktično učiti u stvarnosti, virtualna stvarnost je odgovor. Ona omogućuje da preuzmemo virtualne rizike kako bismo stekli iskustvo u stvarnom svijetu. [13]

4.1. Teorijski koncepti virtualne stvarnosti

Danas, tehnologije virtualne stvarnosti temelje se na idejama koje datiraju još iz 1800-ih. Godine 1838. izumljen je prvi stereoskop. Sastojao se od dvostrukih zrcala za projektiranje jedne slike. Stereoskop kao takav danas je poznat pod nazivom „View Master“, a patentiran je 1939. godine. Pojam „virtualna stvarnost“, prvi put se spominje sredinom osamdesetih godina kada je Jaron Lanier, osnivač VPL istraživanja, počeo razvijati opremu potrebnu za iskustvo onoga što je nazvao „virtualnom stvarnošću“. No i prije toga, tehnolozi su razvili simuliranu okolinu. Jedna prekretnica bila je „Sensorama“ 1956. godine. „Sensorama“ je simulirala stvarno gradsko okruženje. Više senzorska stimulacija omogućavala je osobi da vidi cestu, čuje motor, osjeti vibracije i miris motora u dizajniranom svijetu. Heilig je također 1960. godine patentirao i uređaj „Telesphere Mask“ koji se mogao staviti na glavu. Mnogi će se izumitelji temeljiti na njegovom radu. Do 1965. drugi je izumitelj, Ivan Sutherland, izumio uređaj "Ultimate Display". Uređaj koji postavljanjem na glavu služi kao "prozor u virtualni svijet". Sedamdesetih i osamdesetih godina bilo je to uznemirujuće vrijeme na terenu. Današnja trenutna oprema virtualne stvarnosti duguje zahvalnost pionirskim izumiteljima proteklih šest desetljeća koji su olakšali i omogućili put niskim troškovima i visokokvalitetnim uređajima koji su lako dostupni. [16]

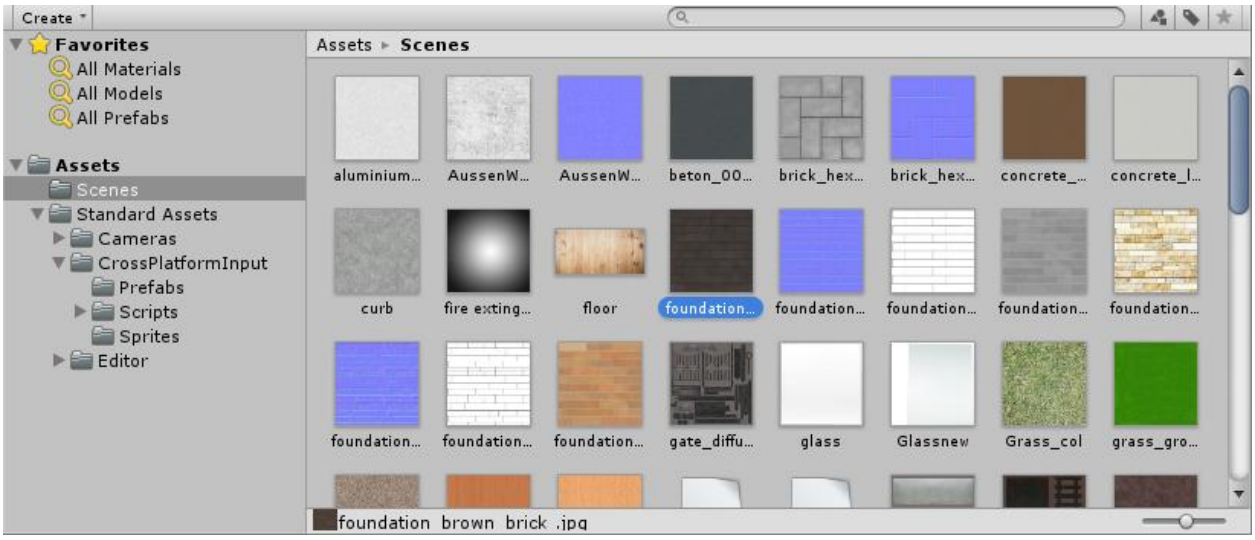
Mozak se temelji na prošlim iskustvima kako bi razvio "pravila" za tumačenje svijeta. Na primjer, zbog neba znamo protumačiti riječ gore. Sjene govore odakle dolazi svjetlost. Relativna veličina stvari govori koji predmet je udaljeniji. Ova pravila pomažu mozgu da djeluje učinkovitije. Razvojni programeri virtualne stvarnosti implementiraju ova pravila i pokušavaju mozgu pružiti iste informacije u virtualnom svijetu. U virtualnom okruženju, objekti koji se kreću trebaju slijediti očekivanja zakona fizike. Sjenčanje i tekstura trebali bi omogućiti određivanje dubine i udaljenosti. Ponekad, kada se virtualni znakovi ne podudaraju s očekivanjima mozga, osoba se može osjećati dezorijentirana. Budući da je ljudski mozak puno složeniji od čak najboljih i najefikasnijih računala, znanstvenici još uvijek pokušavaju shvatiti koji su znakovi najvažniji za određivanje prioriteta u virtualnoj stvarnosti. [17]

5. Korisničko sučelje Unityja

Glavni prozor „Editor“ sastoji se od kartičnih prozora koji se mogu preurediti, grupirati, odvojiti i priključiti. To znači da se izgled „Editora“ može razlikovati od jednoga do drugog, ovisno o osobnoj želji i vrsti posla. [18]

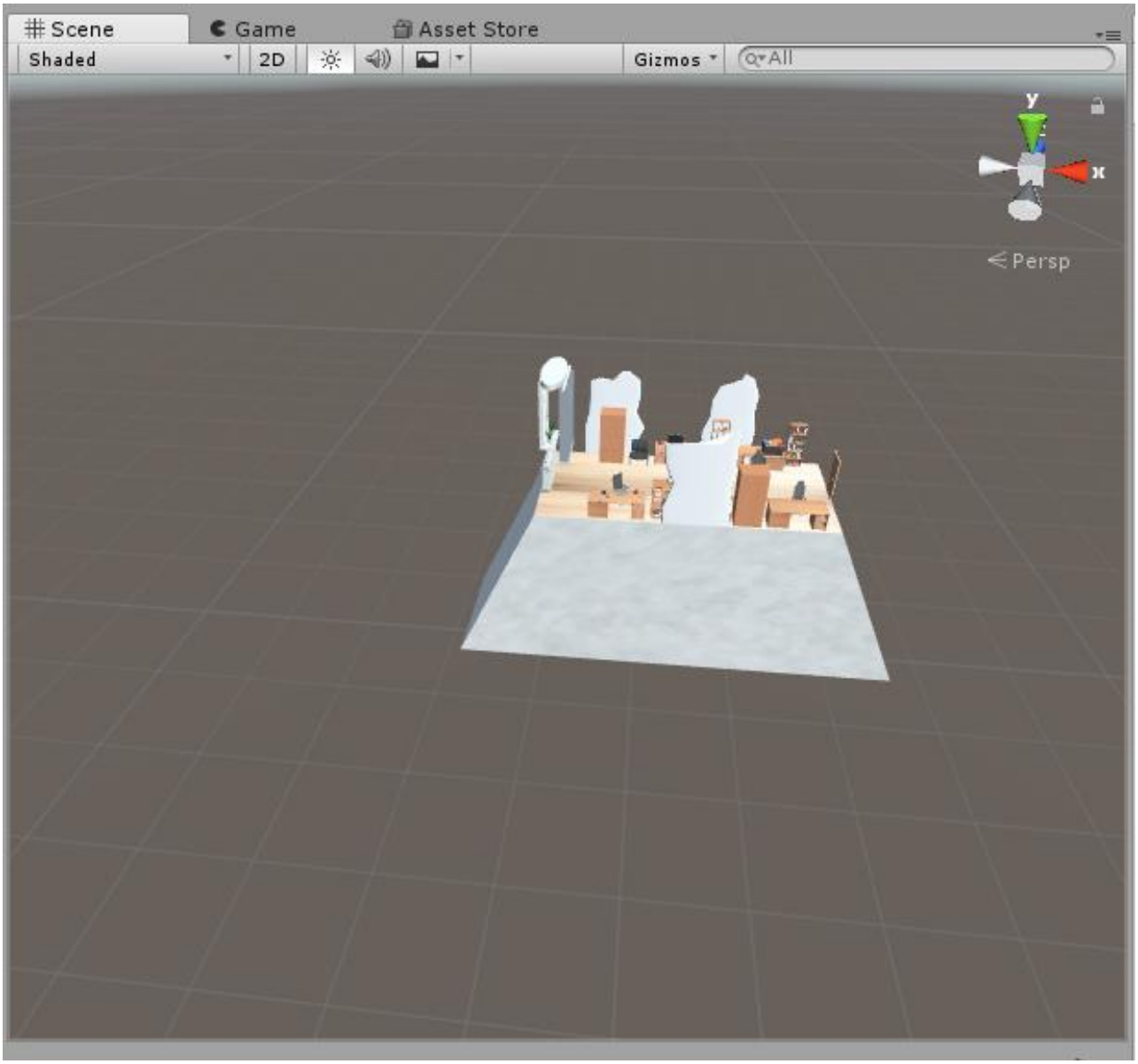
Unity se sastoji od sljedećih dijelova:

1) Prozor projekta (*eng. The Project Window*) – lijeva strana preglednika prikazuje strukturu mapa projekta kao hijerarhijski popis. Može se kliknuti na mali trokut da bi se proširila mapa i prikazala sve ugniježđene mape koje sadrži. Kada se s popisa odabere mapa klikom, njezin će sadržaj biti prikazan na ploči s desne strane. Također, na lijevoj strani alatne trake nalazi se izbornik „Create“ koji omogućuje dodavanje novih sadržaja i podmapa u trenutnu mapu. Sadržaj mape je na desnoj strani ploče i prikazan je kao ikone koje upućuju na njihovu vrstu (skripta, materijal, podmapa). Veličina ikona se može mijenjati pomoću klizača na dnu ploče. Sadržaj će biti zamijenjen hijerarhijskim prikazom ako se klizač povuče na krajnju, lijevu stranu. Prostor koji se nalazi lijevo od klizača prikazuje trenutno odabranu stavku uključujući punu putanju do stavke ako se sadržaj želi naći pomoću trake za pretraživanje koja se nalazi na alatnoj traci. Alatna traka preglednika nalazi se uz gornji rub prozora. Desno od trake za pretraživanje nalaze se tri gumba. Prvi dopušta daljnje filtriranje sadržaja prema vrsti. Sljedeći gumb filtrira sadržaj u skladu s oznakom. Treći gumb je gumb „Favorites“ gdje se mogu spremati često korištene stavke za jednostavan pristup. Stavke iz popisa strukture projekta mogu se povući u „Favorites“ i tako spremiti upite za pretraživanje. [19]



Slika 5.1 Prozor projekta

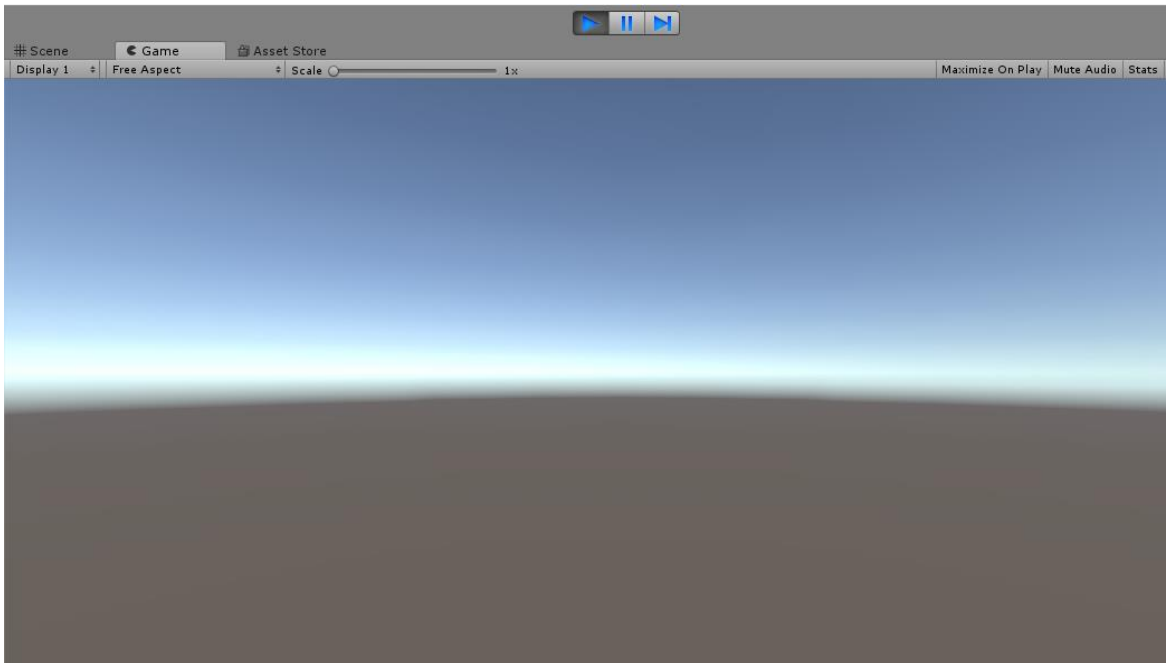
2) Prikaz scene (*eng. The Scene View*) – interaktivni pogled na scenu koja se stvara. Koristi se za odabir i pozicioniranje prizora, likova, kamere, svijetla i sve druge vrste objekta. Ispod kartice „Scene“ nalaze se opcije za način prikaza scene, omogućivanje 2D pregleda, uključivanje i isključivanje osvjetljenja, zvuka i efekata. [20]



Slika 5.2 Prikaz scene

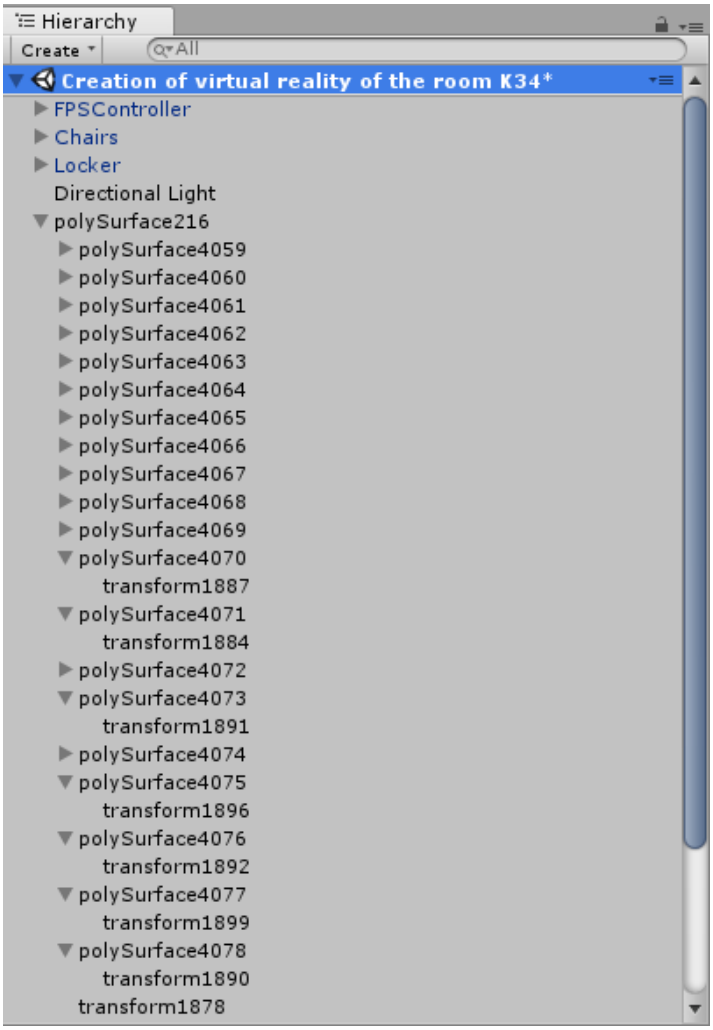
3) Prikaz igre (*eng. The Game View*) – prikazuje konačan izgled igre. Kada se pritisne na „Play“, ulazi se u način reprodukcije i prikazuje se ono što vidi kamera koja je postavljena na sceni. Kada se nalazimo u načinu reprodukcije, sve promjene koje napravimo su privremene i resetiraju se kada izađemo iz načina reprodukcije. Pritiskom na „Pause“ izlazi se iz načina reprodukcije. Opcija „Step“ omogućava preskakanje po koracima. Ispod kartice „Game“ nalazi se nekoliko opcija. Opcija „Display“ omogućuje odabir kamere ako je postavljeno više kamera na

sceni. Sljedeća opcija je „Aspect“ i ona omogućuje promjenu omjera slike. „Scale“ omogućuje uvećanje ili smanjenje prikaza trenutne kamere. Opcija „Maximize On Play“ omogućuje prikaz igre preko cijelog zaslona kada se uđe u način reprodukcije. Opcija „Mute Audio“ služi za isključivanje zvuka kada se uđe u način reprodukcije. Opcija „Stats“ omogućuje praćenje performansa igre kada se uđe u način reprodukcije. [21]



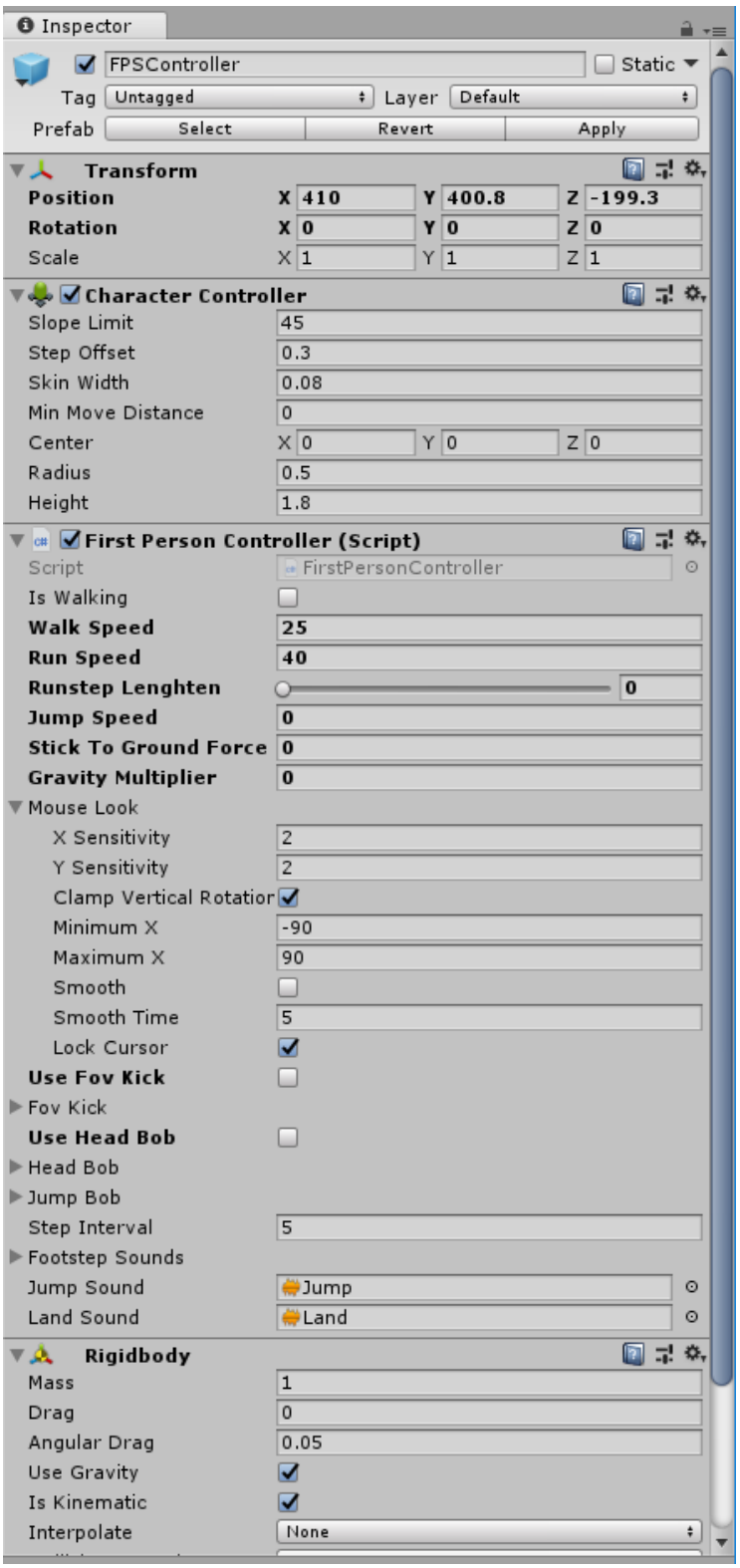
Slika 5.3 Prikaz igre

4) Hijerarhijski prozor (*eng. The Hierarchy Window*) – omogućuje kreiranje novog sadržaja (2D objekta, 3D objekta, efekata, osvjetljenja, zvuka, videa, teksta) i sadrži popis svakoga „GameObjecta“ u trenutnoj sceni. Kada se objekti dodaju na scenu, oni se pojave u popisu hijerarhije, a ako se uklone sa scene, oni nestaju s popisa. Najčešće su objekti navedeni u hijerarhijskom prozoru onim redoslijedom kojim su izrađeni. Objekti se mogu posložiti povlačenjem gore ili dolje po hijerarhiji. Svi objekti koji se nalaze u hijerarhijskom prozoru mogu se pretraživati pomoću trake za pretraživanje. Desno od trake za pretraživanje nalazi se opcija zaključavanja hijerarhije. Također, jedan ili više objekata može se grupirati unutar drugog objekta. U Unityju to se naziva „Parenting“. Kada stvorimo skupinu objekta, najgornji objekt naziva se „Parent Object“, a svi objekti grupirani ispod njega nazivaju se „Child Objects“. Također, mogu se stvoriti i ugniježđeni objekti „Parent-Child Objects“ koje nazivamo „Descendants“ glavnog objekta najviše razine. [22]



Slika 5.4 Hijerarhijski prozor

5) Inspektor (*eng. The Inspector Window*) – prikazuje detaljne informacije o trenutno odabranom „GameObjectu“ uključujući sve priključene komponente i njihova svojstva. Također, omogućuje modificiranje funkcije „GameObjecta“, sadržaja, materijala u sceni te dodavanje novih funkcija. Na primjer, kada „GameObject“ ima priključene komponente skripte, inspektor prikazuje javne varijable skripte što znači da se postavke skripte mogu jednostavno izmijeniti unutar inspektora bez mijenjanja koda. Za promjenu redoslijeda priključenih komponenata, unutar inspektora klikne se na zaglavlje željene komponente te se komponenta povuče gore ili dolje. Važno je napomenuti da se redoslijed komponenata može mijenjati samo unutar istog „GameObjecta“. Komponente između različitih „GameObjecta“ ne mogu se premještati. [23]



Slika 5.5 Inspektor

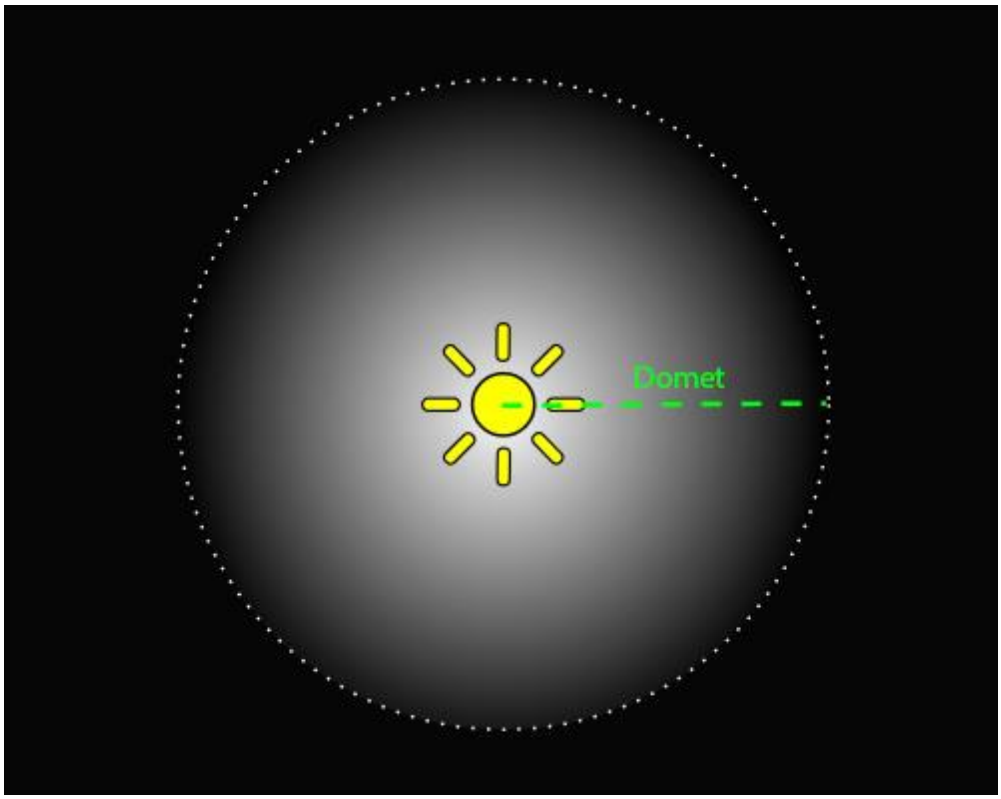
6) Alatna traka (*eng. The Toolbar*) – prvi alat na alatnoj traci je „Hand Tool“ koji služi za pomicanje po sceni. „Move“ omogućuje pomicanje „GameObjecta“ na sceni. Pritiskom na „GameObject“ pojave se tri strelice koje omogućuju pomicanje tog objekta po osima x, y i z. Pomoću alata „Rotate“ objektu možemo promijeniti rotaciju i to na način da kliknemo na „GameObject“ i rotiramo crvene, zelene i plave krugove koji se pojave oko njega. Crvena predstavlja x-os, zelena predstavlja y-os, dok plava predstavlja z-os. Alat „Scale“ služi za skaliranje „GameObjecta“. Kada kliknemo na „GameObject“, pojave se tri kockice koje omogućavaju skaliranje po x, y i z osi. Također, „GameObject“ možemo ravnomjerno skalirati tako da kliknemo i povučemo na kockicu koja se nalazi u sredini triju osi. „RectTransform“ kombinira alate za pomicanje, skaliranje i rotaciju te se koristi za pozicioniranje 2D elemenata, ali može biti koristan za manipulaciju 3D „GameObjecta“. Za premještanje „GameObjecta“ klikne se na pravokutni „Gizmo“ te se povlačenjem miša objekt premjesti na željeno mjesto. Za skaliranje objekta klikne se na bilo koji rub ili kut pravokutnog „Gizmoa“ te se povlačenjem mijenja visina, širina ili dužina „GameObjecta“. Ako se želi skalirati „GameObject“ duž jedne osi klikne se na rub, a ako se želi skalirati duž dvije osi klikne se na kut. Za zakretanje „GameObjecta“ pokazivač miša postavi se točno iza kuta pravokutnika. Kada se kursor promijeni u ikonu rotacije klikne se i povuče mišem kako bi se zarotirao „GameObject“. Alat „Transform“ kombinira alate za pomicanje, skaliranje i rotaciju. Alat „Pivot“ pozicionira „Gizmo“ na stvarnu, početnu točku mreže, dok alat „Center“ postavlja „Gizmo“ u središte postavljenoga „GameObjecta“. Alat „Local“ održava „Gizmoovu“ rotaciju u odnosu na „GameObject“, dok alat „Global“ spaja „Gizmo“ s orijentacijom scene. [24]

Slika 5.6 Alatna traka

6. Sustavi osvjetljavanja u Unityju

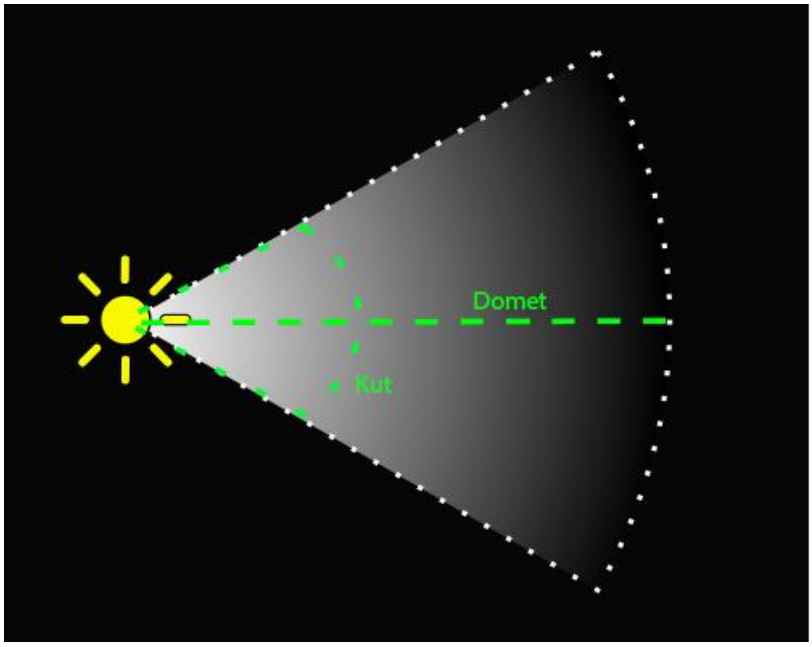
Osvjetljenje je bitan dio svake scene. Ono određuje ton i raspoloženje 3D okruženja. [25] Postoje četiri sustava osvjetljavanja u Unityju: [26]

1) Točkasto svjetlo (*eng. Point Light*) – nalazi se na točki u prostoru i šalje svjetlost u svim smjerovima jednako. Smjer svjetla koji udara na površinu je linija od točke kontakta pa sve do središta svjetlosnog objekta. Intenzitet se smanjuje što je objekt udaljeniji od svjetla, dostižući nulu u određenom rasponu. Jačina svjetla je obrnuto proporcionalna kvadratu udaljenosti od izvora. To je poznato kao „inverzni kvadratni zakon“ i sličan je načinu na koji se svjetlo ponaša u stvarnom svijetu.



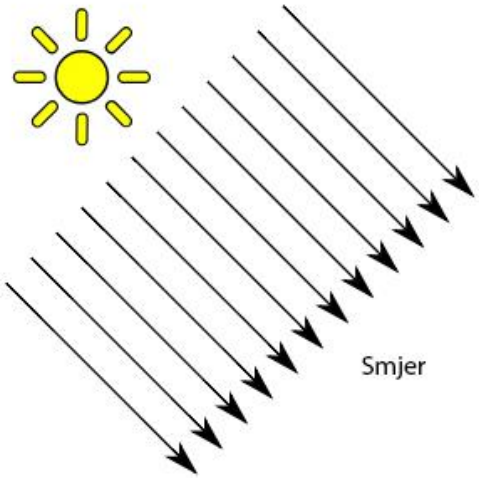
Slika 6.1 Točkasto svjetlo [1]

2) Reflektorsko svjetlo (*eng. Spot Light*) – ima određeno mjesto i raspon preko kojeg svjetlo pada, no ograničeno je na kut što dovodi do stožastog oblika osvjetljenja. Središte stošca usmjereno je prema svjetlosnom objektu. Svjetlost se također smanjuje na rubovima stošca. Širenje kuta povećava širinu stošca te se time smanjuje intenzitet svjetla.



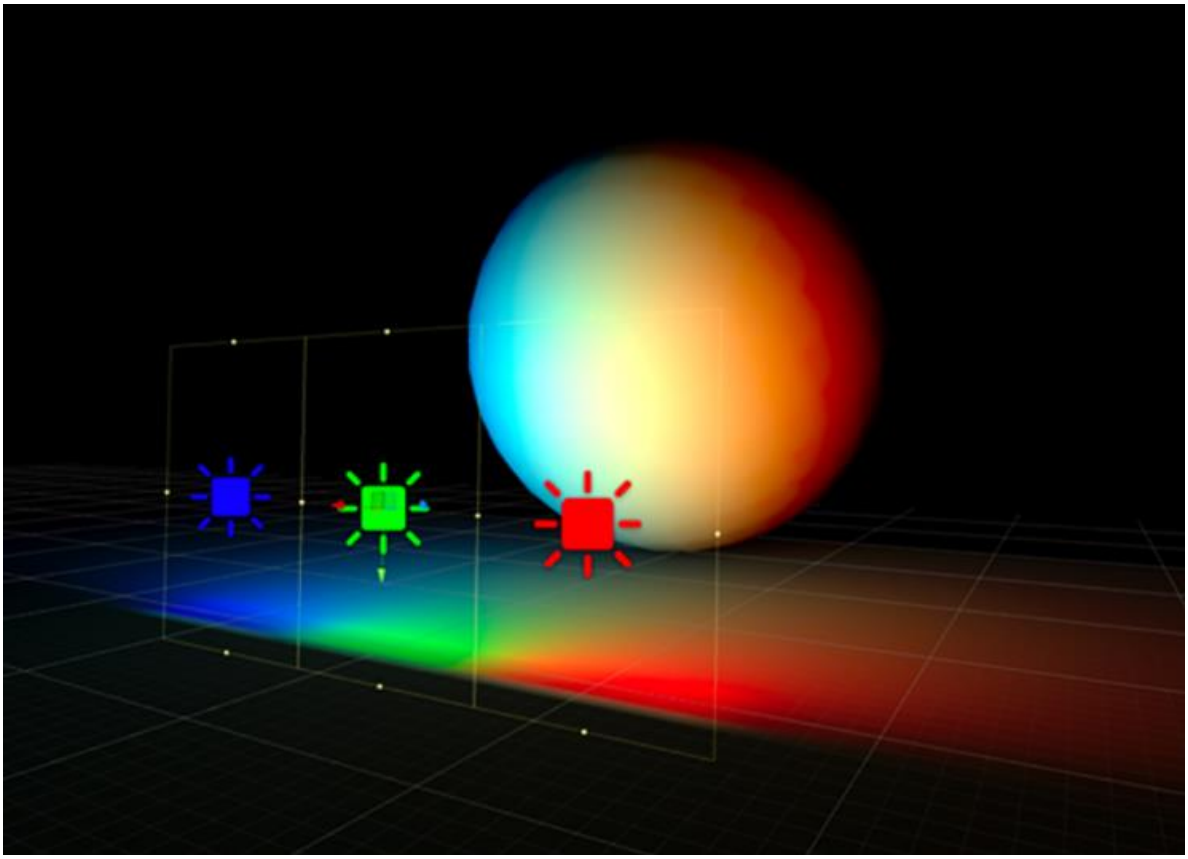
Slika 6.2 Reflektorsko svjetlo [2]

3) Usmjerena svjetlost (*eng. Directional Light*) – vrlo korisna za stvaranje efekata sunčeve svjetlosti. Oponašajući na mnoge načine sunčevu svjetlost, ova vrsta svjetla može se smatrati udaljenim izvorom svjetlosti pa se osvijetljen objekt može postaviti na bilo koje mjesto u sceni. Svi objekti u sceni ponašaju se kao da svjetlo uvijek dolazi u istom smjeru. Udaljenost svjetlosti od objekta nije definirana pa se zbog toga intenzitet svjetla ne smanjuje.



Slika 6.3 Usmjerena svjetlost [3]

4) Prostorna svjetlost (*eng. Area Light*) – određena je pravokutnikom u prostoru. Svjetlost se emitira u svim smjerovima ravnomjerno preko površine, ali samo s jedne strane pravokutnika. Ne možemo kontrolirati domet svjetla, no kako svjetlo putuje dalje od izvora tako se intenzitet svjetla smanjuje.



Slika 6.4 Prostorna svjetlost [4]

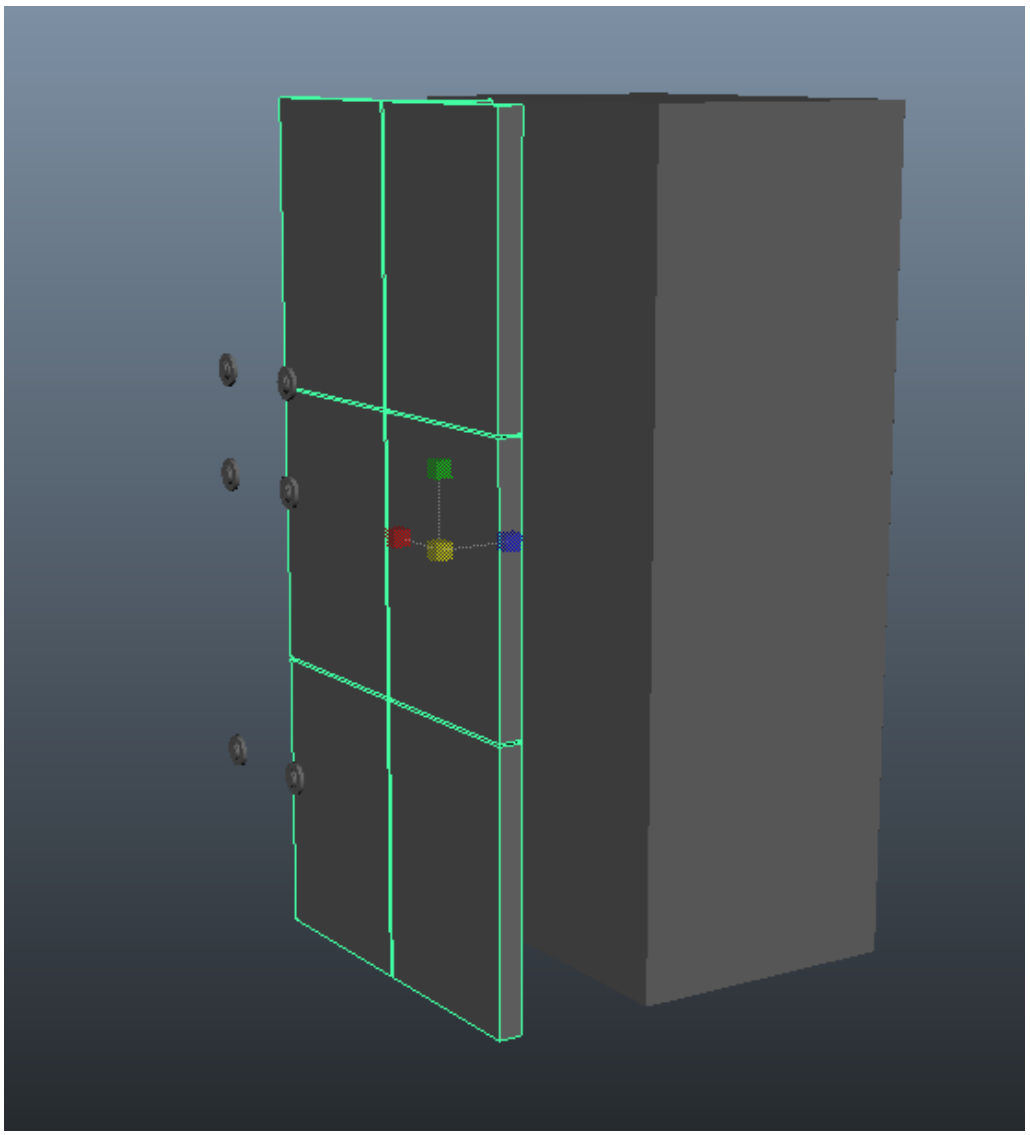
7. Praktični dio

U praktičnom dijelu rada odabrao sam temu izrade virtualne stvarnosti. Praktični dio rada sam započeo u programu Autodesk Maya. Svi 3D modeli koji se nalaze na sceni napravljeni su u tom programu. Za izradu 3D modela korišteni su osnovni poligoni poput kugle, kocke i valjka te su iz osnovnih poligona dobiveni oni složeniji. Također, korišteni su i alati poput: Edit Mesh, Insert Edge Loop Tool, Extrude, Bevel, Bend, Combine, Seperate. Za ravne plohe poput ormara, stolova, vrata, zidova i prozora korištene su kocke, a za zaobljene su korištene kugle i valjci. Nakon izrade scene 3D modelima je trebalo dodati teksture kako bi oni bili što sličniji predmetima u stvarnosti. Sve teksture su također dodane preko Autodesk Maya programa. Nakon dovršetka cijele scene ta ista scena je otvorena u Unity programu. Programi Unity i Microsoft Visual Studio su korišteni za dovršetak projekta. U Unityju je napravljen „360° Skybox“, dodana je kamera, osvjetljenje, tekst i animacija. Većinom su korištene gotove Unity skripte kako bi se ostvarila interakcija, a pojedine skripte pisane su C# i JavaScript jezikom u Microsoft Visual Studiju te su ubačene u Unity. Na kraju, cijeli taj projekt je pretvoren u aplikaciju. Cilj rada je prikazati mogućnosti Unity alata te postupke nužne za izradu virtualne stvarnosti.

7.1. Izrada 3D modela

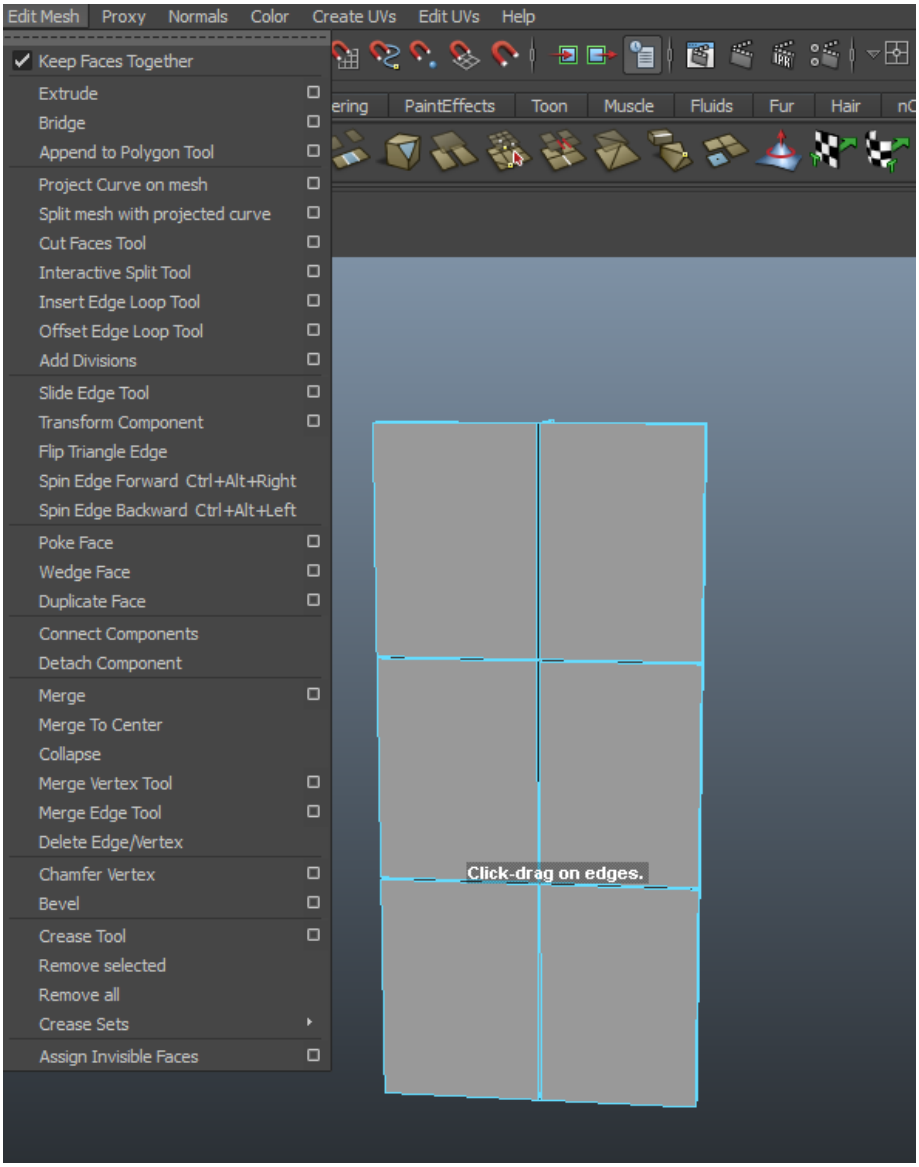
Svi ravni poligoni kao što su ormari, stolovi, police, vrata, zidovi i prozori napravljeni su pomoću kocke te su korišteni isti alati i tehnike kao u navedenom primjeru izrade ormara.

Visina, širina i debljina poligona se mijenja na način da se poligon selektira te se pritisne na tipku R kako bi se uključio „Scale Tool“. Nakon što je „Scale Tool“ uključen pojave se tri kvadratića koja nam omogućuju oblikovanje.



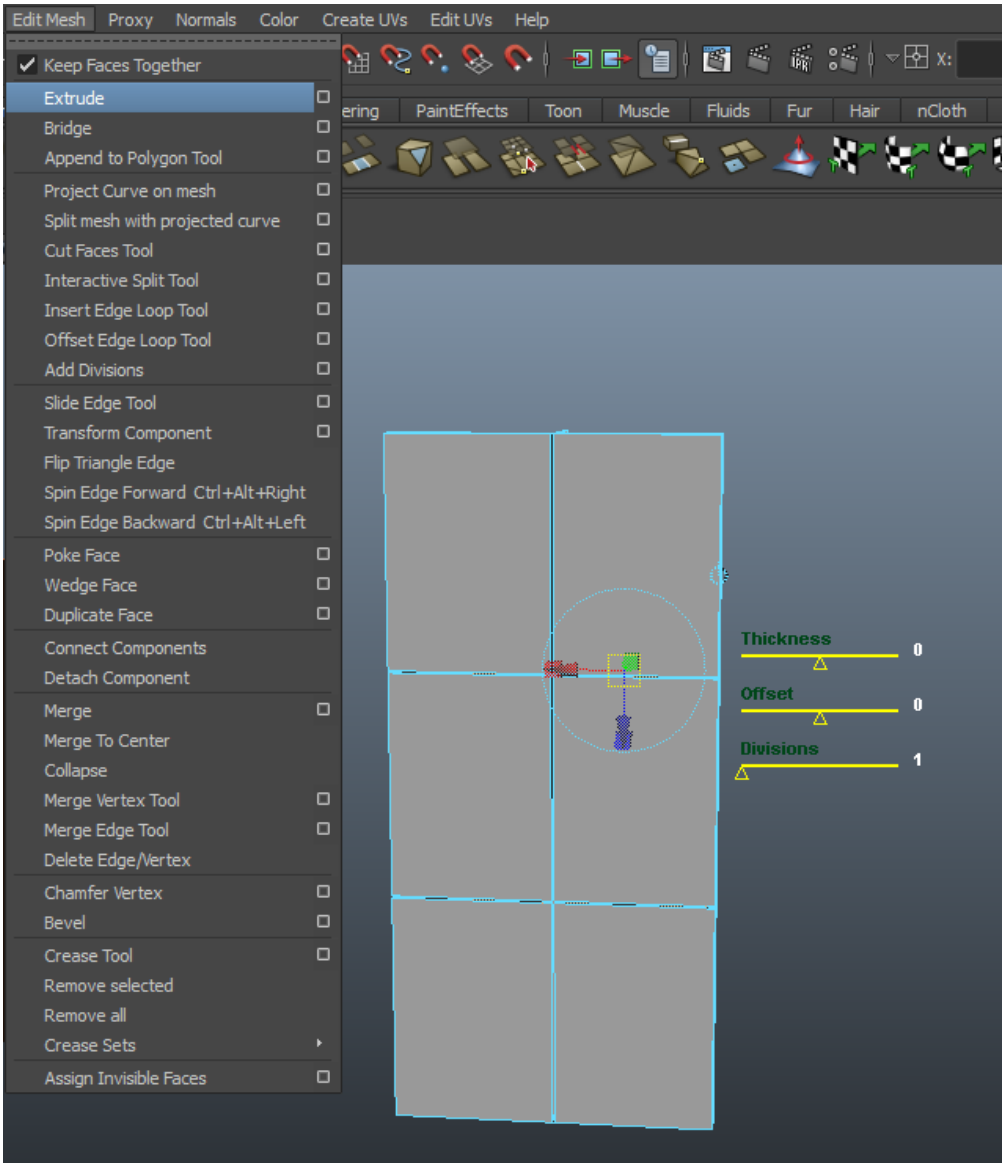
Slika 7.1 Oblikovanje kocki i ključanica

Na tanje oblikovanu kocku dodane su podjele budući da ormar ima više predjela. Podjele se dodaju na način da označimo model te u izborniku „Edit Mesh“ odaberemo „Insert Edge Loop Tool“ te upišemo broj podjela. U ovom slučaju dodane su četiri horizontalne i dvije vertikalne podjele.



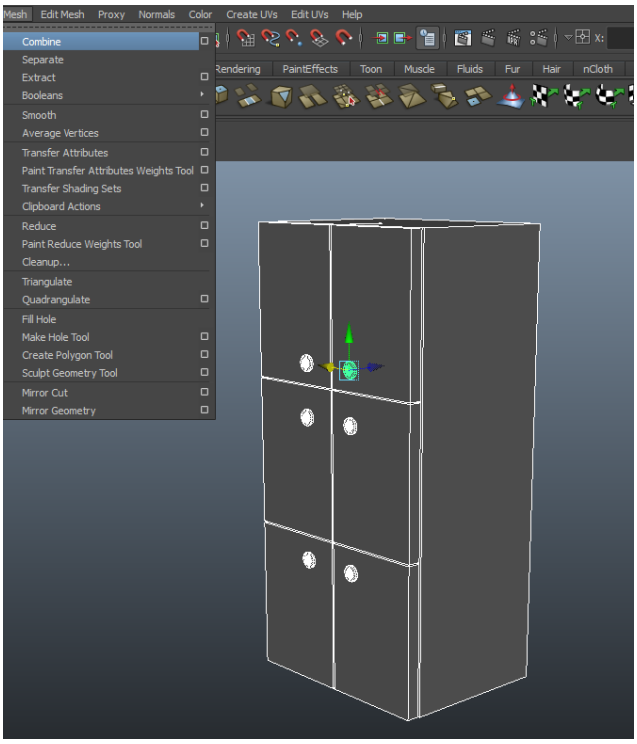
Slika 7.2 Dodavanje podjela

Prelaskom u „Face Mode“ označimo prostor među podjelama te u izborniku „Edit Mesh“ odaberemo opciju „Extrude“. Prostor među podjelama pomaknemo prema unutra kako bi se napravio prazan prostor koji će označavati predjele ormara, odnosno kako bi se istaknulo da ormar ima više predjela.

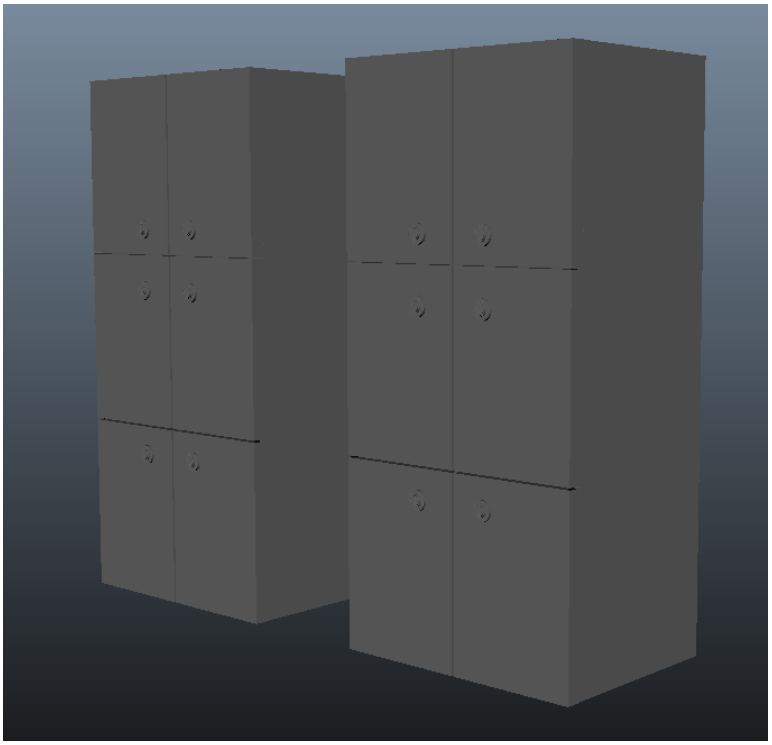


Slika 7.3 Extrude *alat*

Na kraju su se oblikovani pravokutnici s ključanicama označili i spojili klikom na izbornik „Mesh“ i odabirom opcije „Combine“. Klikom na ormar te pritiskom tipki Ctrl+D ormar je dupliciran.



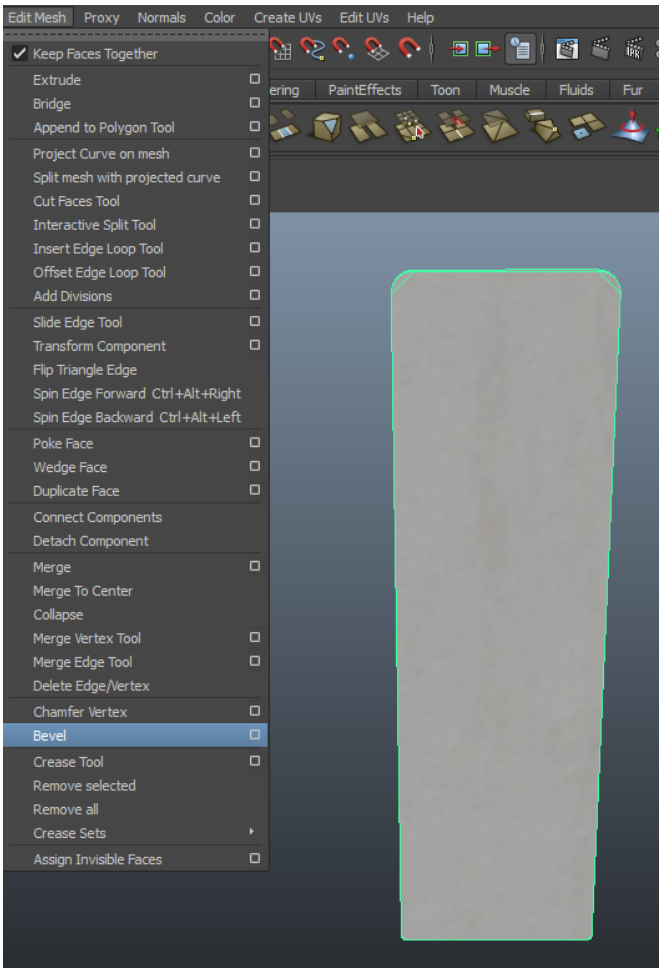
Slika 7.4 Spajanje poligona



Slika 7.5 Dupliciranje ormara

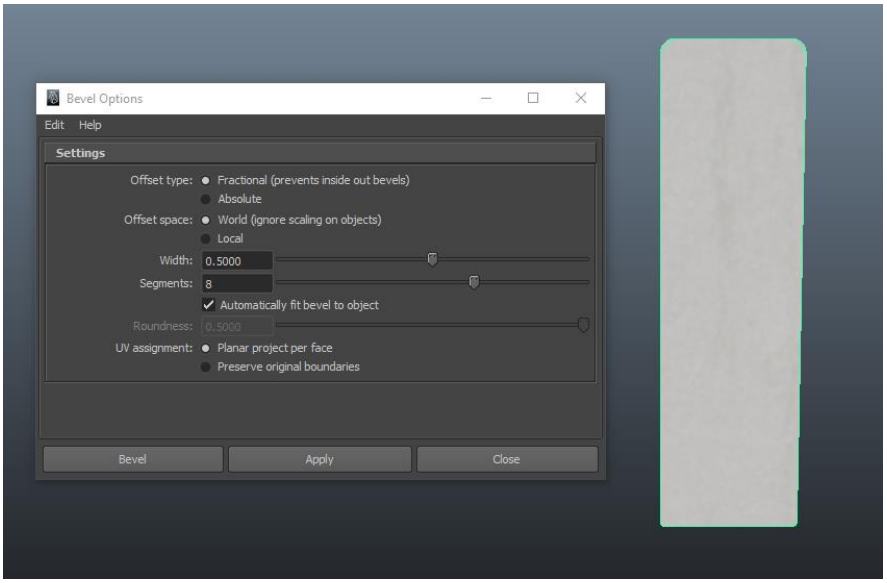
Svi zaobljeni poligoni, kao što su radijatori, držači za zavjese, stolice, zidni ormarići napravljeni su kombinacijom kocke, valjka ili kugle te su korišteni isti alati i tehnike kao u navedenom primjeru izrade radijatora.

Na početku je uzeta kocka koja je preoblikovana. Visina, širina i debljina su preoblikovane pomoću alata „Scale Tool“ koji se uključi pritiskom na tipku R. Rubovi su zaobljeni pomoću alata „Bevel Tool“. Kako bi se rubovi zaoblili, selektira se poligon te se odabere „Edge Mode“. Kad je uključen „Edge Mode“, odabiru se rubovi koji se žele zaobliti te se u izborniku „Edit Mesh“ odabere opcija „Bevel“.



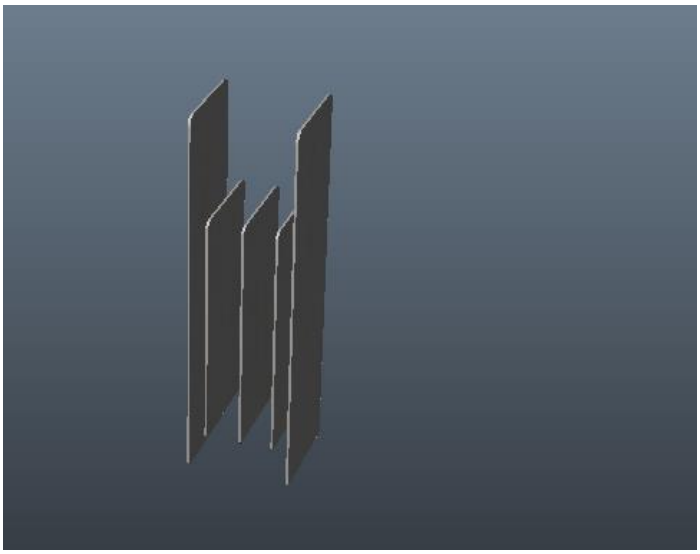
Slika 7.6 Zaobljivanje rubova

Opcija „Bevel“ omogućuje zaobljenje rubova na način da se odabere širina među segmentima te broj segmenata. Što je širina veća, zaobljenje će biti veće, a što je širina manja, zaobljenje će biti manje. Što je broj segmenata manji, to će zaobljenje biti kruće, a što je broj segmenata veći, to će zaobljenje biti glađe.



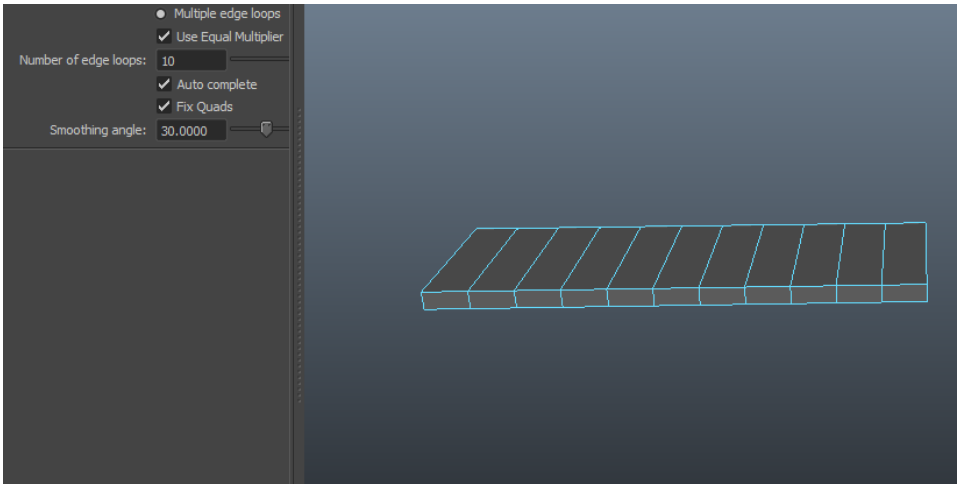
Slika 7.7 Bevel *opcije*

Nakon dobivenoga željenog oblika taj se oblik duplicira nekoliko puta pritiskom na tipke Ctrl+D.



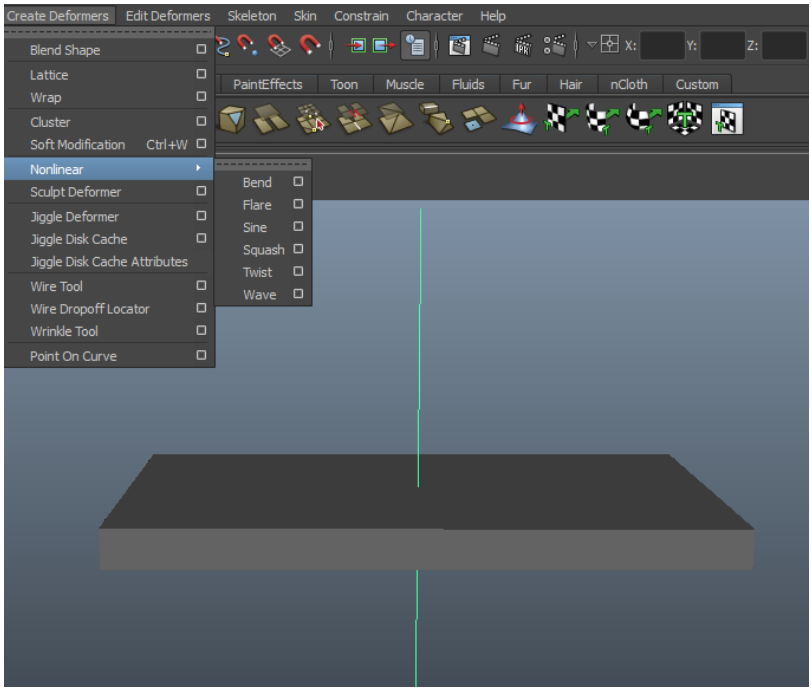
Slika 7.8 Dupliciranje

Zaobljeni dijelovi mogu se dobiti i presavijanjem ravnih poligona. Najprije se napravi ravan poligon koji se onda označi te se iz izbornika „Edit Mesh“ odabere „Insert Edge Loop Tool“. U postavkama se poligon podijeli na deset dijelova kako bi se dobila što glađa površina prilikom zaobljivanja.



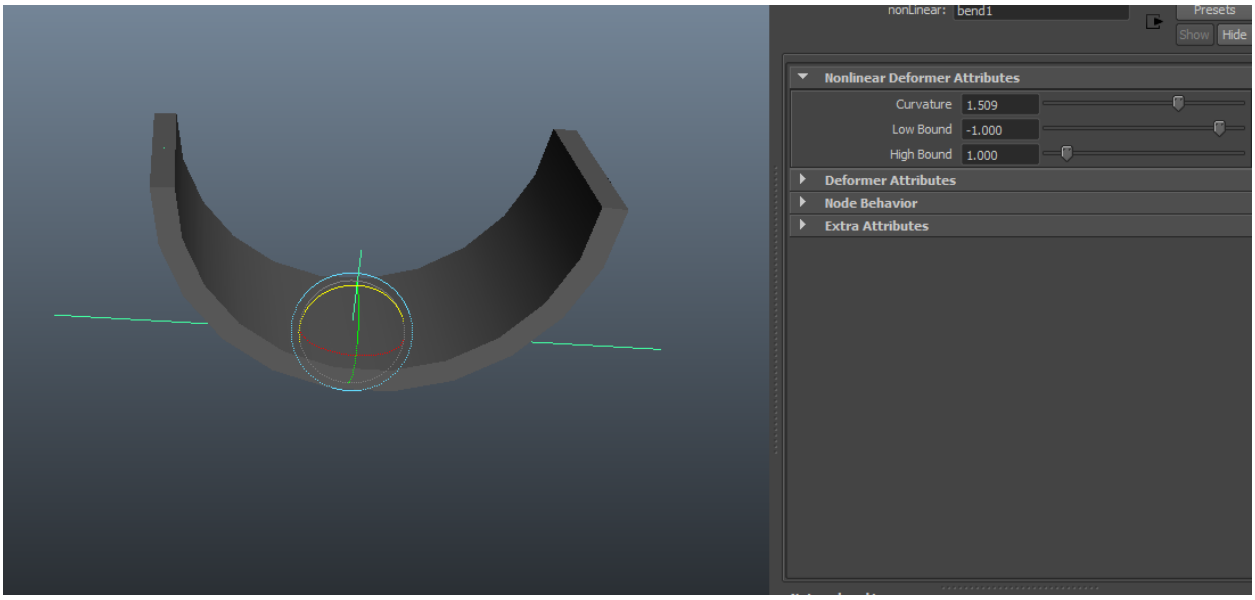
Slika 7.9 Postavljanje podjela

U izborniku „Create Deformers“ odabere se opcija „Nonlinear“ te se klikne na „Bend“. Pojavi se crta koja označava u kojem će se smjeru poligon presavijati.



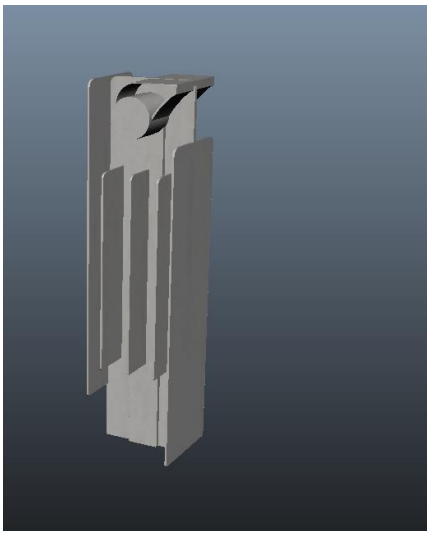
Slika 7.10 Izbornik Create Deformers

Ta se crta mora rotirati kako bi ona bila pod kutom od 90°. Na taj se način ravna ploha može savijati prema gore ili dolje. Označi se crta te se napravi rotacija pomoću alata „Rotation Tool“ koji se uključi pritiskom na tipku E. „Curvature“ označava koliko će ravna ploha biti savijena. Ako je „Curvature“ pozitivan, ploha će se savijati prema gore, a ako je negativan, ploha će se savijati prema dolje. „Low Bound“ mora biti postavljen na minus, a „High Bound“ mora biti postavljen na plus. Broj mora biti isti ako želimo da se lijeva i desna strana savijaju podjednako.



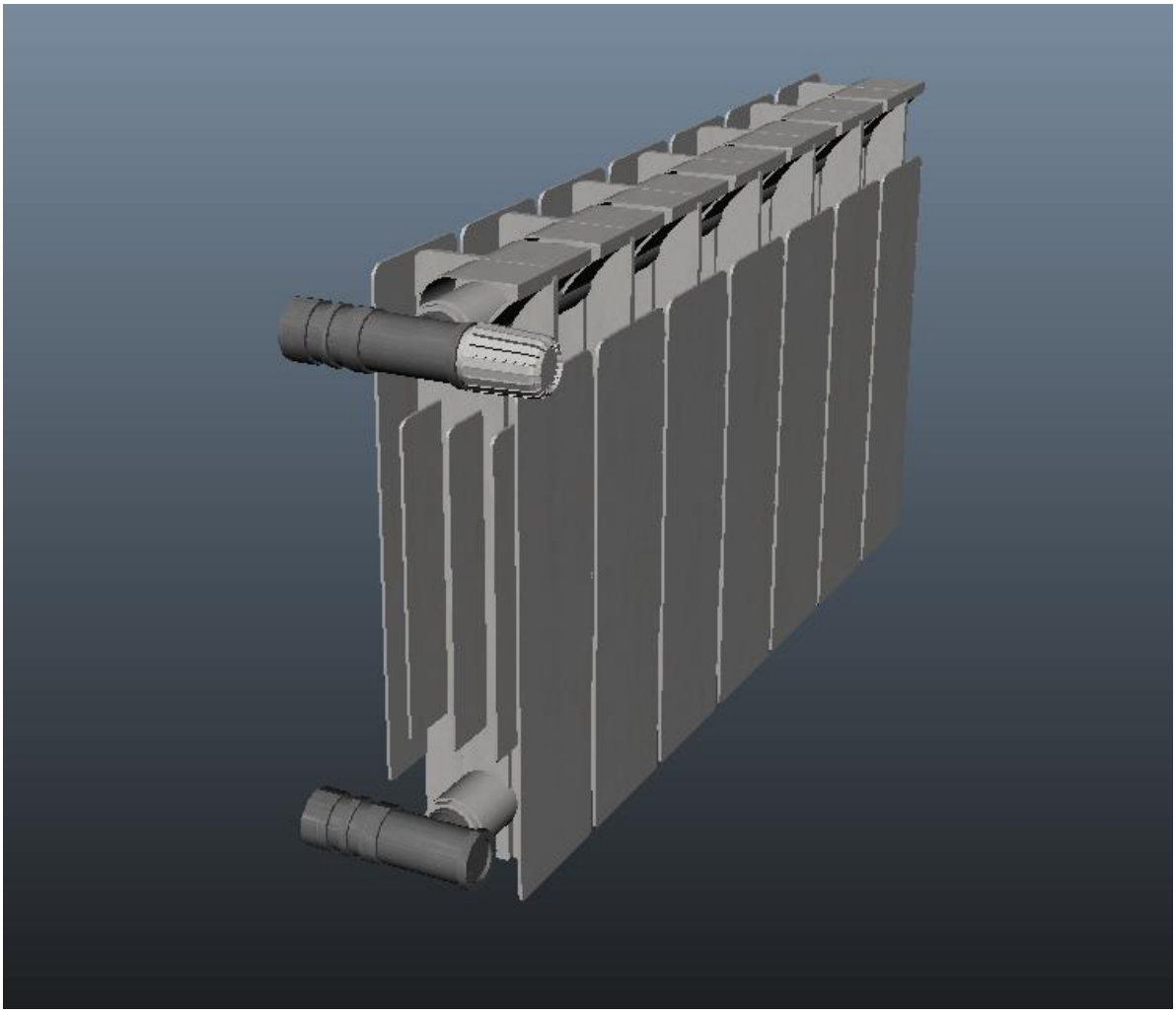
Slika 7.11 Postavke presavijanja

Nakon što je oblik presavijene plohe zadovoljavajući, tu plohu veličinom prilagodimo 3D modelu. Kombiniranjem ravnih ploha, valjaka i presavijenih ploha, dobivena je struktura jednog dijela radijatora.



Slika 7.12 Dio radijatora

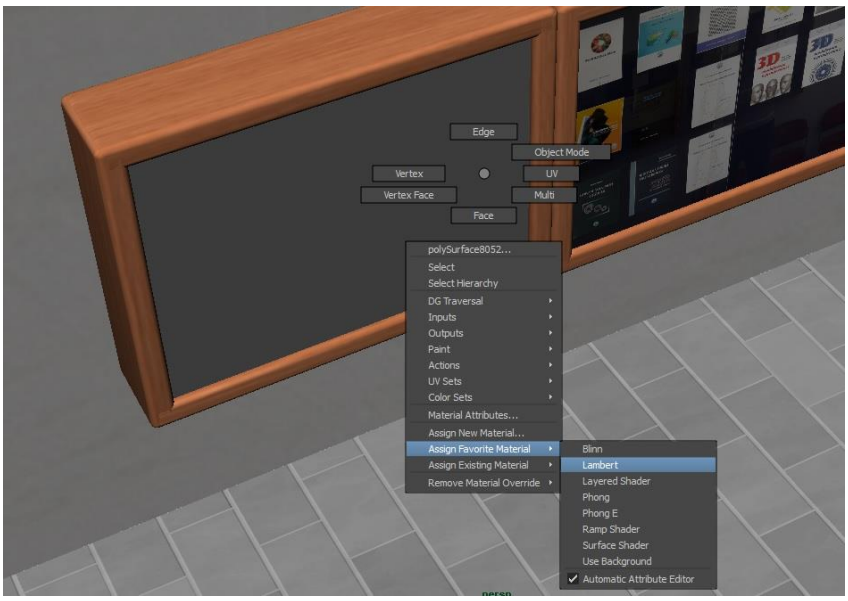
Budući da se izgled radijatora ponavlja, označi se dio napravljenog radijatora te se taj dio duplicira nekoliko puta pritiskom na tipke Ctrl+D. Duplicirani dijelovi se poslože jedan pored drugog te se označe i spoje tako da se u izborniku „Mesh“ odabere opcija „Combine“. Ručke i ventili koji se nalaze na radijatoru također su napravljeni pomoću alata „Bevel“ i „Bend“ koji su prikazani prethodno.



Slika 7.13 Konačan izgled

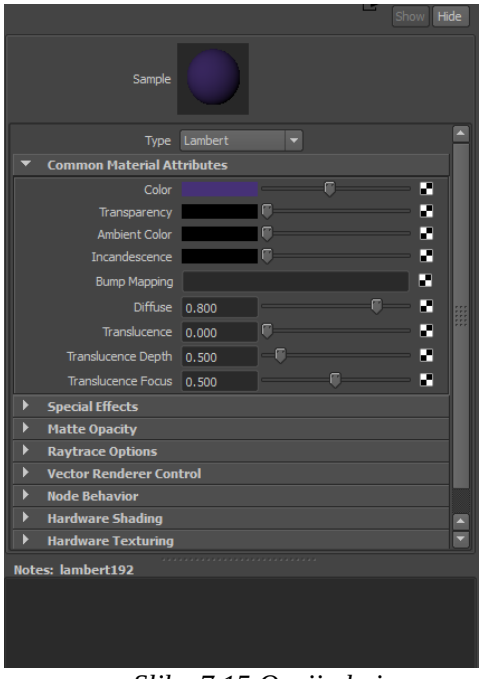
7.2. Teksturiranje 3D modela

Postupak je za postavljanje tekstura isti za svaki model. Označi se model ili dio modela na koji se želi postaviti tekstura. Kada se model označi, klikne se na njega te se nakon toga drži pritisnuta desna tipka miša. Iz izbornika se mišem navigira do opcije „Assign Favorite Material“ te se odabire vrsta materijala. U ovom je projektu odabran „Lambert“ budući da nema reflektirajućih ili sjajnih materijala.

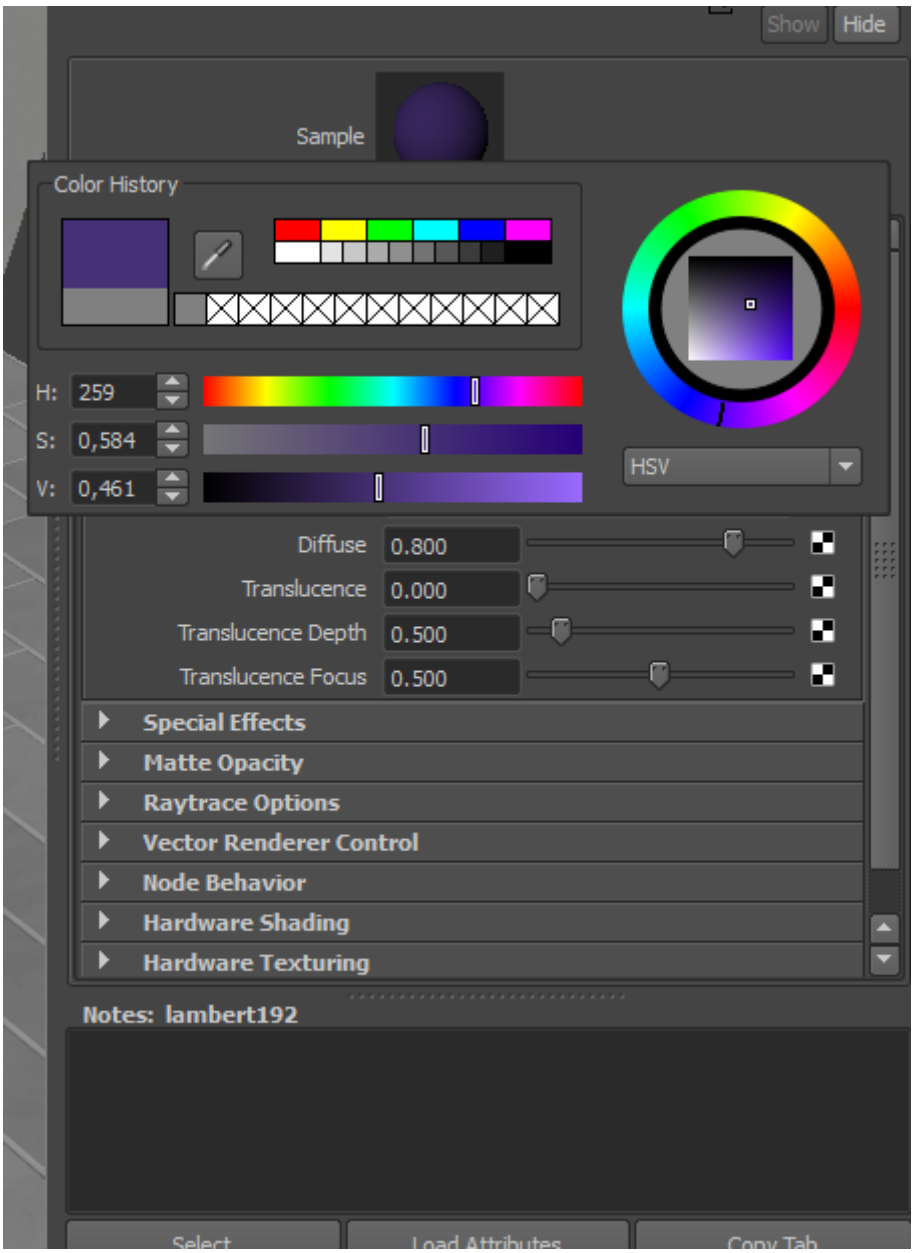


Slika 7.14 Odabir vrste materijala

Nakon što je vrsta materijala odabrana, s desne strane se pojavi nekoliko opcija koje daju mogućnost stavljanja boje na model, određivanje razine prozirnosti modela, boju ambijenta i izbljeđenost boje. Također se nudi i opcija postavljanja vlastite teksture na model klikom na kvadratić koji se nalazi desno od boje. Na nama je da odlučimo hoće li se koristiti boja ili tekstura za pojedini 3D model. Važno je napomenuti da ako se koristi boja, tada se ne može koristiti tekstura i obrnuto. Ako se želi primijeniti boja na 3D model, klikne se na opciju „Color“ koja se nalazi u prozoru s desne strane.

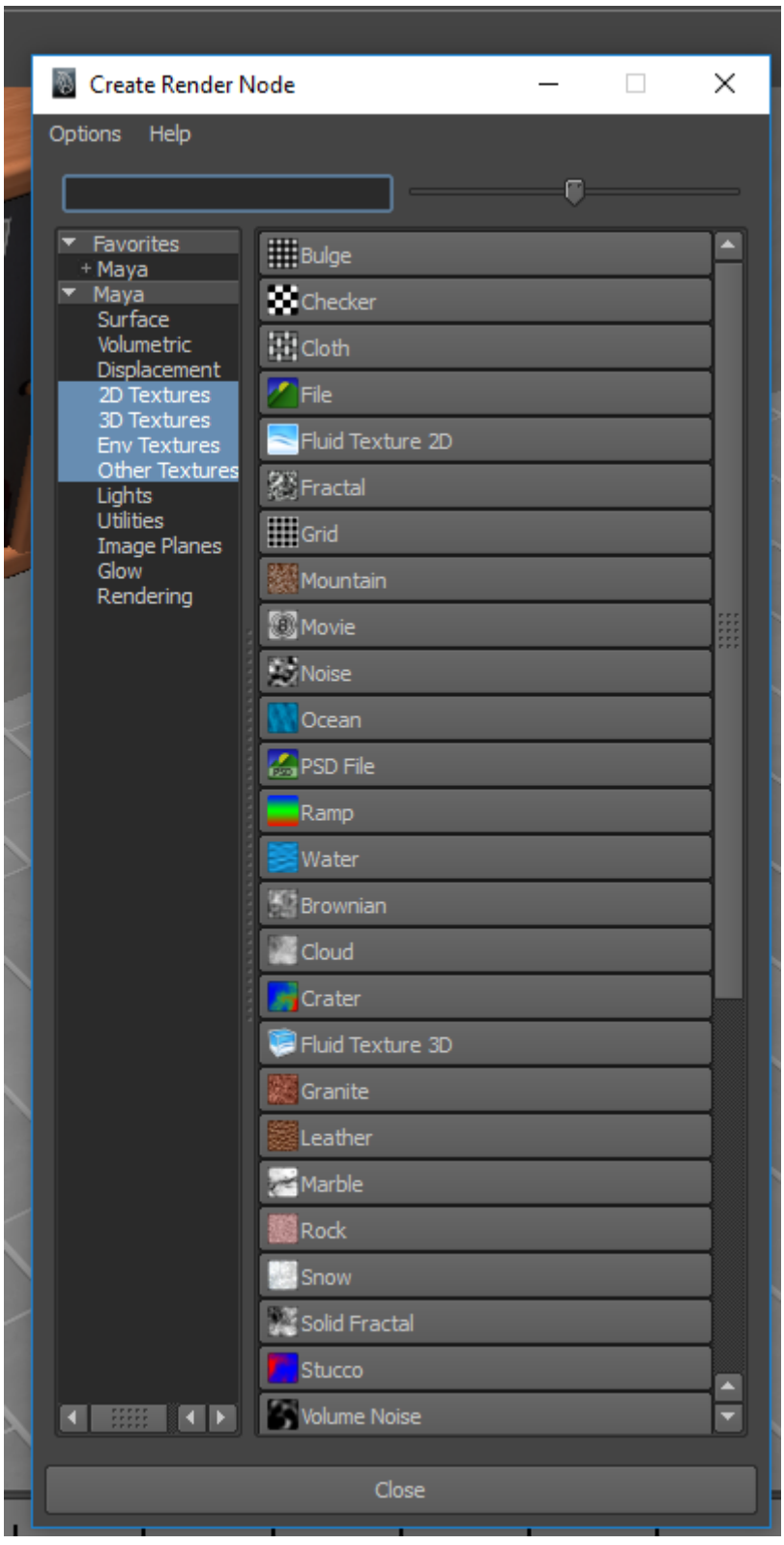


Slika 7.15 Opcija boje



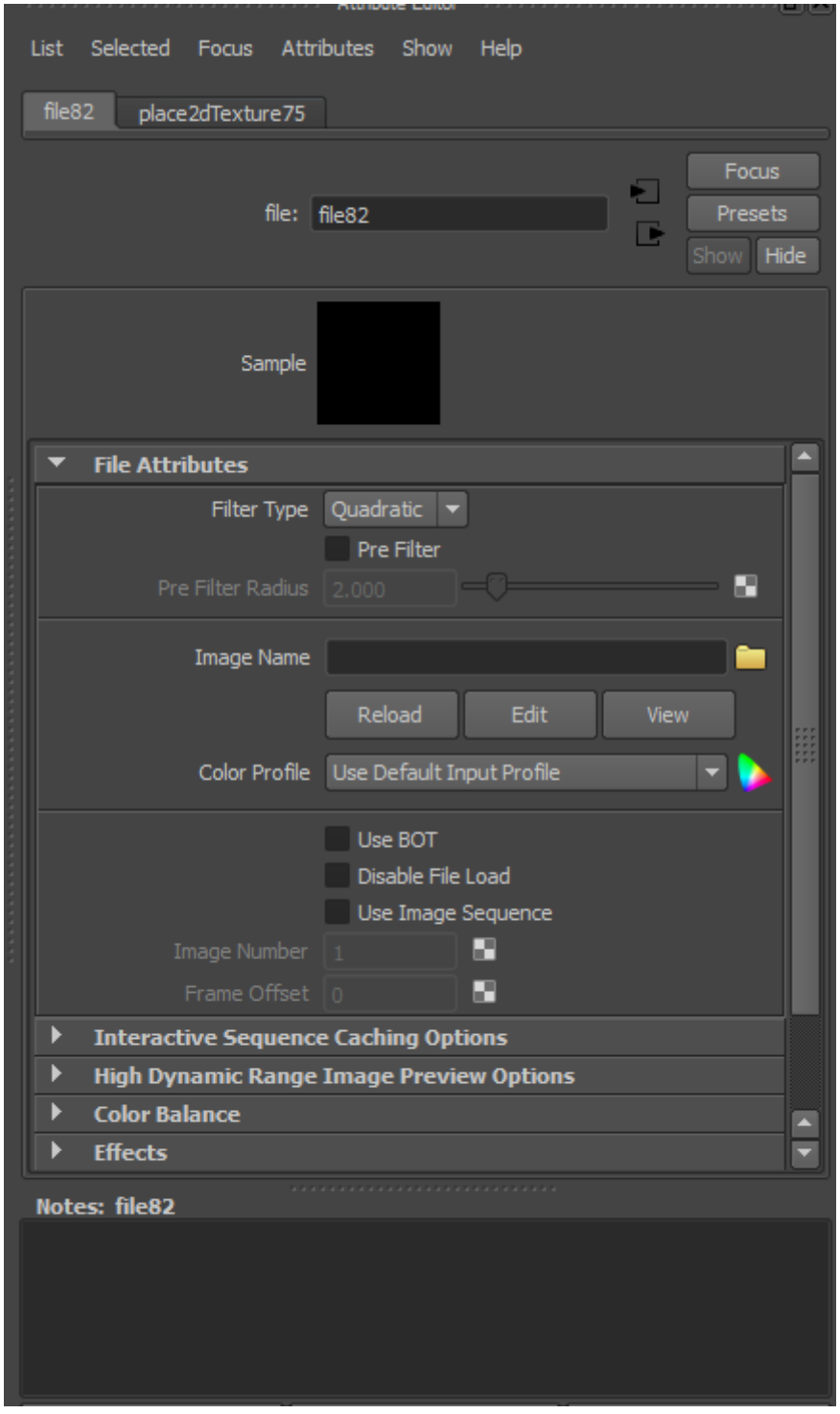
Slika 7.16 Apliciranje boje

Ako se želi primijeniti tekstura na 3D model, klikne se na kvadratić koji se nalazi desno od opcije za boju. Otvori se novi prozor u kojem se odabere vrsta datoteke. U ovom sam slučaju odabrao „File“ budući da je moja tekstura bila slika.



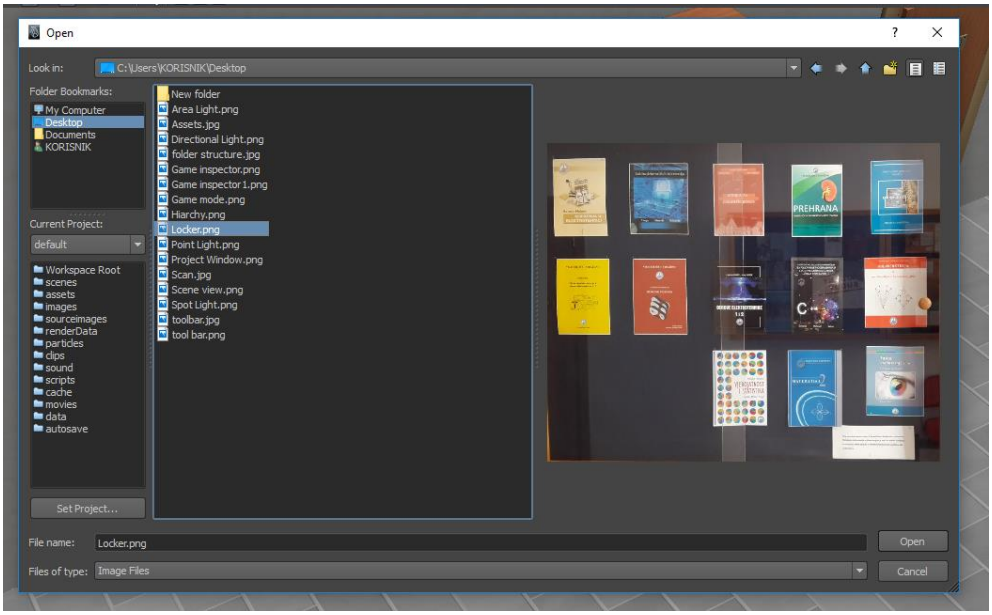
Slika 7.17 Odabir vrste datoteke

U desnom prozoru se pojavi mapa na koju se klikne da bi se definirala putanja teksture koja će biti aplicirana na 3D model.



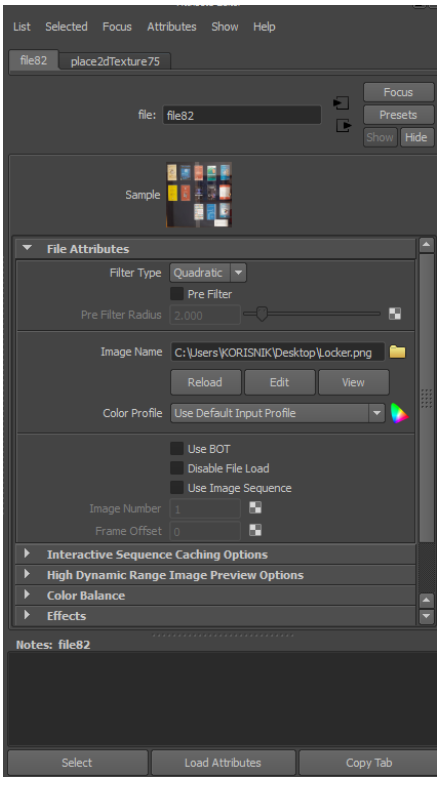
Slika 7.18 Odabir putanje teksture

Kada se pronađe željena tekstura, ona se označi i klikne se na „Open“.



Slika 7.19 Otvaranje teksture

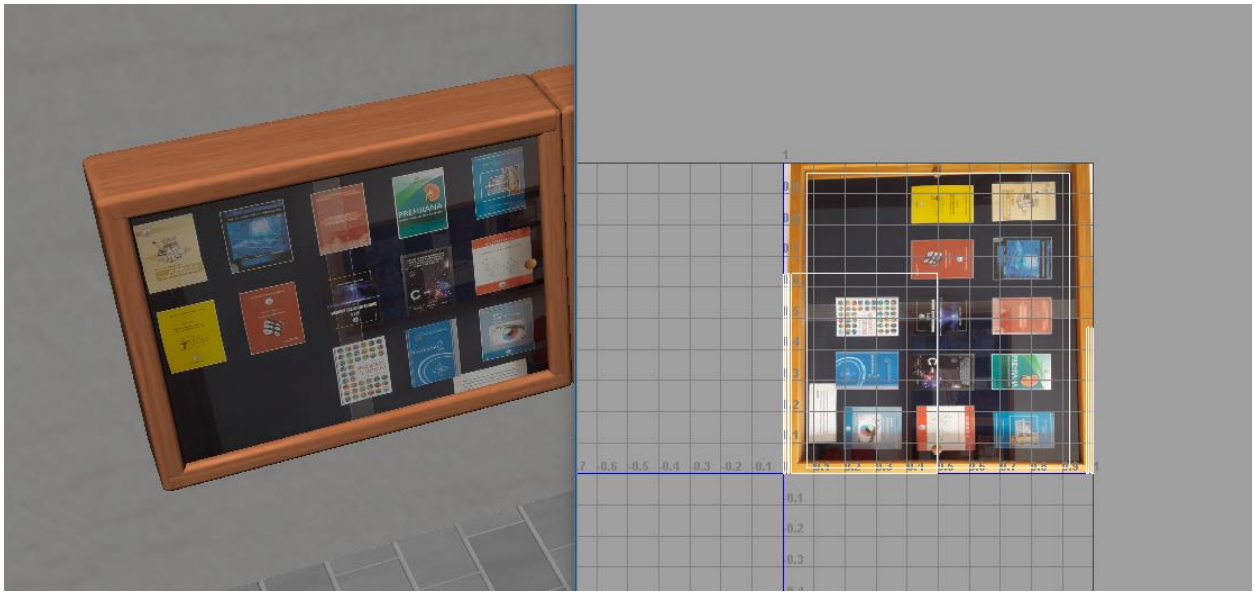
Nakon što se klikne na „Open“, putanja mape bi trebala biti prikazana i „Sample“ bi trebao prikazivati umanjenu teksturu, što znači da je tekstura uspješno aplicirana na model.



Slika 7.20 Izgled nakon otvaranja teksture

Kada se tekstura otvori, ona najčešće neće biti dobro postavljena na 3D model. UV mapa teksture mora se posložiti u „Texture Editoru“.

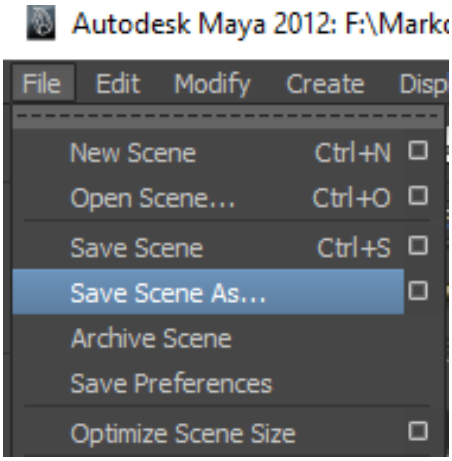
Slika 7.21 Aplicirana tekstura prije UV mapiranja



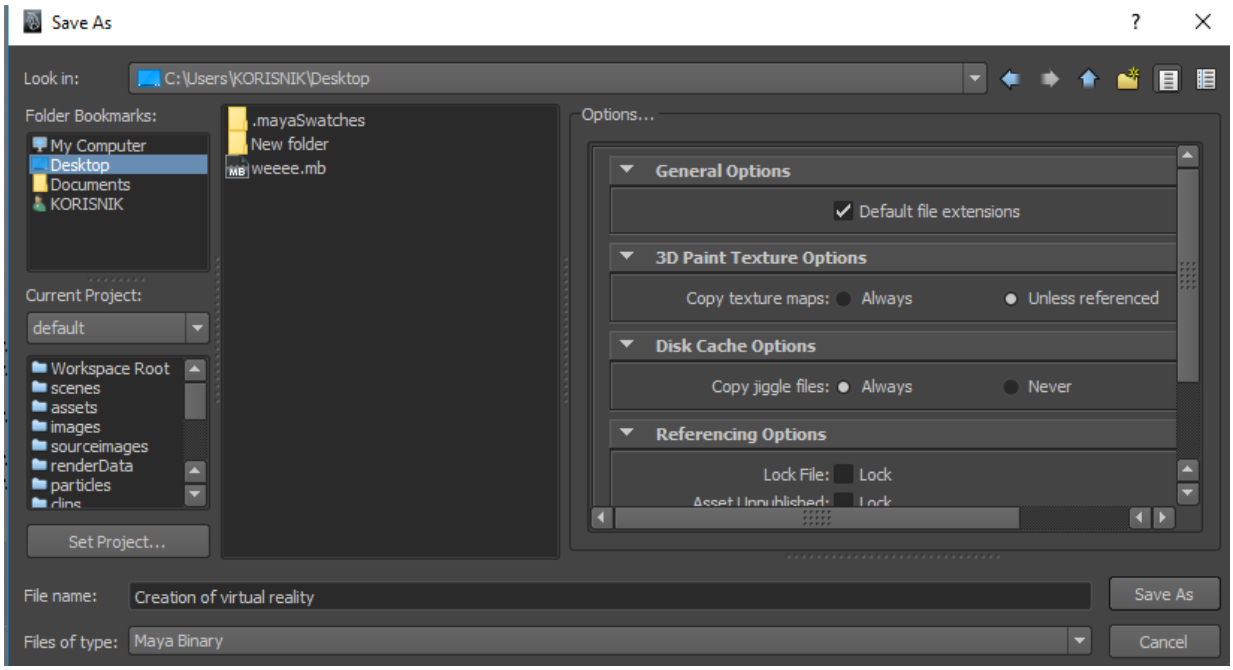
Slika 7.22 Izgled nakon što je tekstura posložena u Texture Editoru

7.3. Spremanje scene

Kada je završena izrada svih 3D modela i kada su postavljene sve teksture na te modele, scenu je trebalo spremiti. Scena se sprema klikom na izbornik „File“ te se iz izbornika odabere „Save Scene As“. Pojavi se prozor u kojem se određuje odredište ili mapa u koju će se spremiti scena. Također, scena se imenuje i određuje se format u kojem će biti spremljena. Ja sam se odlučio za „Maya Binary“ format zato što ću projekt završiti u Unityju, a on podržava taj format.



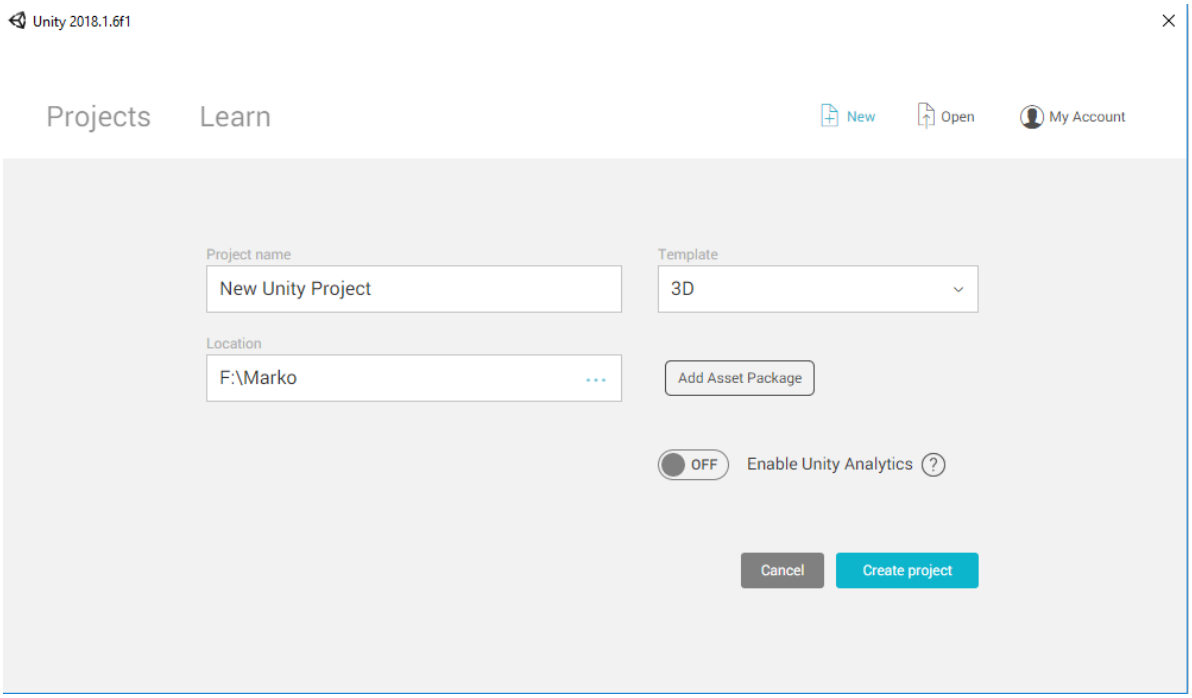
Slika 7.23 Izbornik File



Slika 7.24 Spremanje scene

7.4. Stvaranje novog Unity projekta

Kada se Unity pokrene, pojavi se prozor koji nudi kreiranje novog projekta ili otvaranje postojećeg. Klikom na „New“ odabire se kreiranje novog projekta. Tada se otvori prozor u kojem se projektu određuje ime, tip i mjesto gdje će biti spremljen te se klikne na „Create project“.

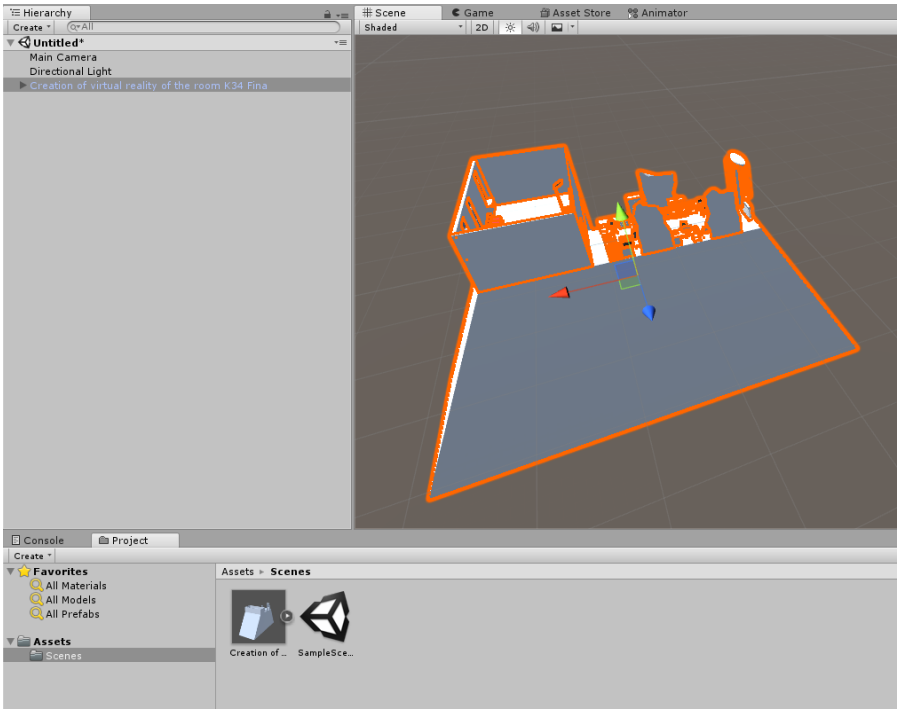


Slika 7.25 Kreiranje projekta

Kada je projekt kreiran, otvara se „Sample scene“. Klikne se na izbornik „File“ te se odabere „New Scene“. Stvorit će se početna scena koja će biti imenovana „Untitled“ te će u sebi imati glavnu kameru i osvjetljenje. Scena se može pri spremanju preimenovati po želji pomoću opcije „Save Scene As“.

7.5. Otvaranje Maya scene u Unityju

Postoje dva načina za otvaranje Maya scene. Prvi način je da se u Unityju pronađe i otvori mapa pod nazivom „Scenes“ te da se povuče i ispusti Maya scena u tu Unity mapu. Drugi način je da se pronađe lokacija mape na računalu gdje je spremljen Unity projekt. Kada se lokacija mape pronađe, mapa se otvori. Unutar glavne mape otvori se prva podmapa „Assets“, a u podmapi „Assets“ otvori se druga podmapa „Scenes“ te se u nju stavlja Maya scena. Nakon toga scena će biti ubačena u prozor projekta u Unityju, no prikaz scene neće biti prazan. Kako bi se omogućio prikaz, scena iz prozora projekta mora se povući u hijerarhijski prozor.

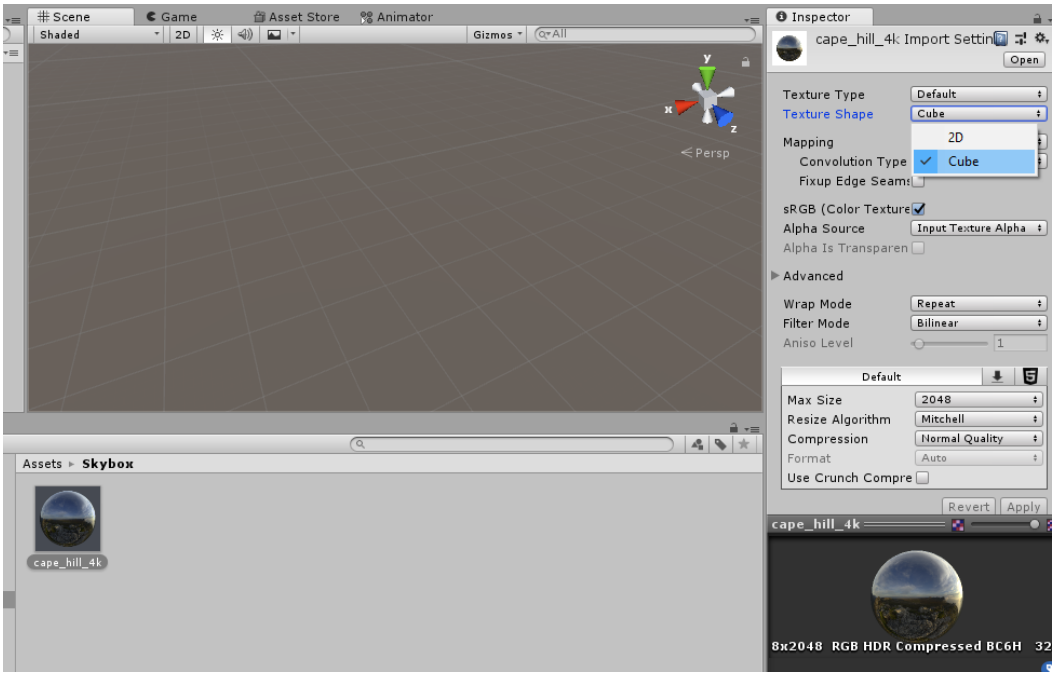


Slika 7.26 Omogućivanje prikaza scene

Nakon toga se u Unityju napravi nova mapa unutar mape „Asstes“. Mapa je nazvana „Textures“ s obzirom na to da će se u mapi nalaziti materijali 3D modela, odnosno teksture. Postupak je ubacivanja tekstura isti kao i postupak ubacivanja scene. Teksture povučemo u mapu „Textures“ te pričekamo da se teksture ubace. U ovom slučaju teksture su automatski aplicirane na 3D modele u sceni budući da Unity prepoznaje da je scena već prije imala aplicirane teksture u Mayi.

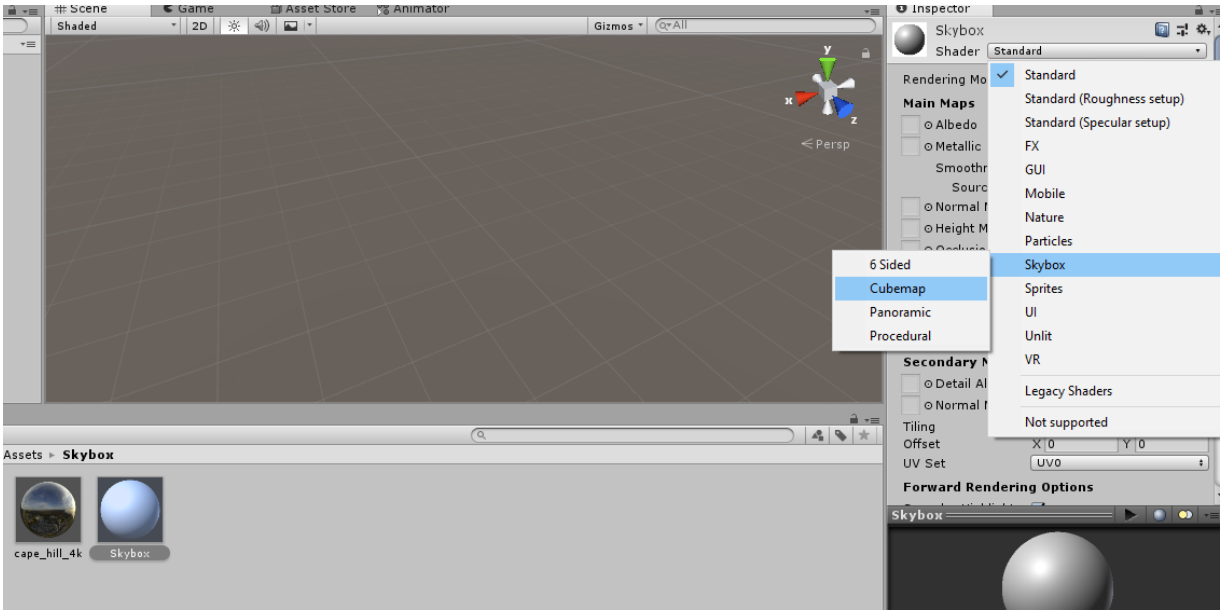
7.6. Postavljanje 360° Skyboxa

Napravi se nova mapa unutar mape „Assets“. Mapa se može nazvati bilo kako, no logično je da se nazove „Skybox“ s obzirom na to da se postavlja „Skybox“. Otvori se mapa „Skybox“ te u se u mapu povlači slika koja će se koristiti za „Skybox“. Klikne se na sliku te se u prozoru inspektor, koji se nalazi s desne strane, „Texture Shape“ iz 2D promijeni u „Cube“.



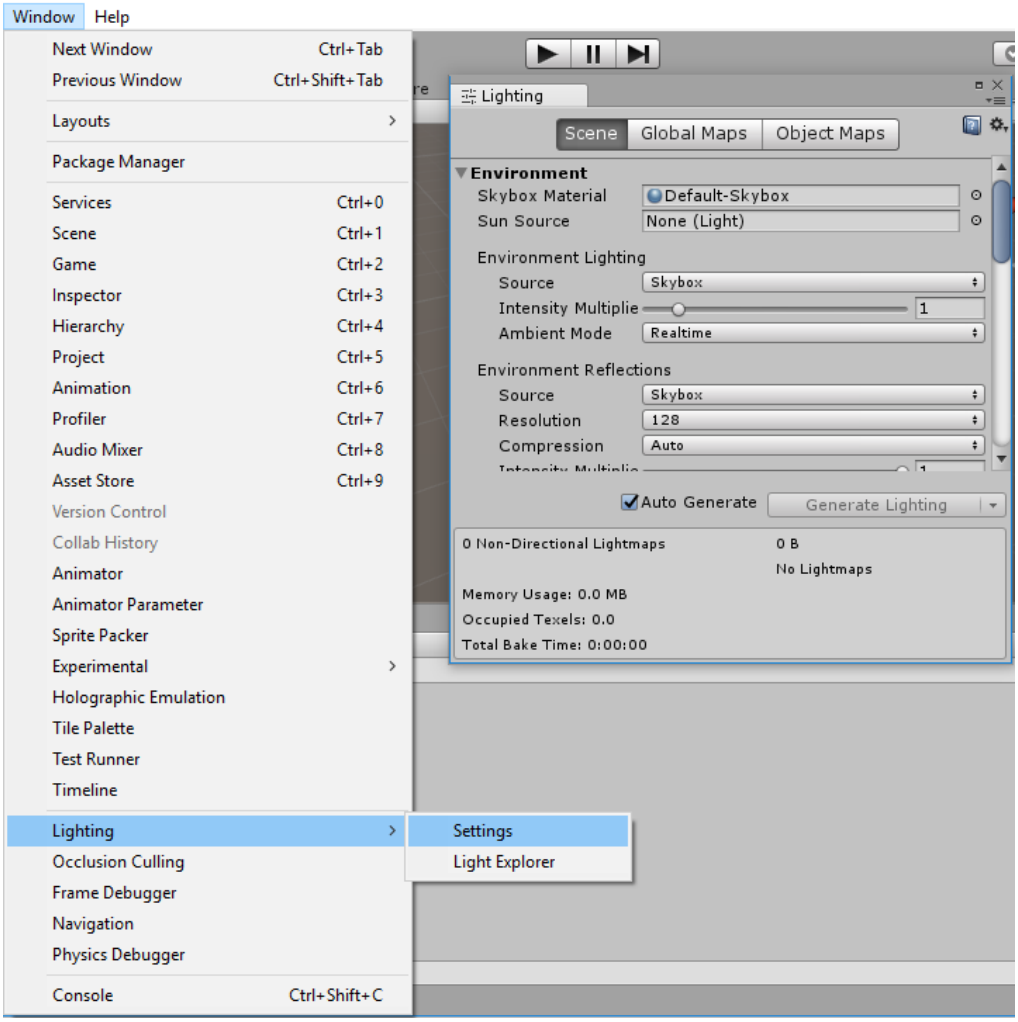
Slika 7.27 Promjena iz 2D u Cube

Klikne se desnim klikom pokraj slike te se iz izbornika „Create“ odabere opcija „Material“. Označi se kreirani materijal te se u prozoru „Inspector“ promijeni tip „shadera“. Za tip „shadera“ navigira se do opcije „Skybox“ te se odabere opcija „Cubemap“.



Slika 7.28 Odabir tipa shadera *materijala*

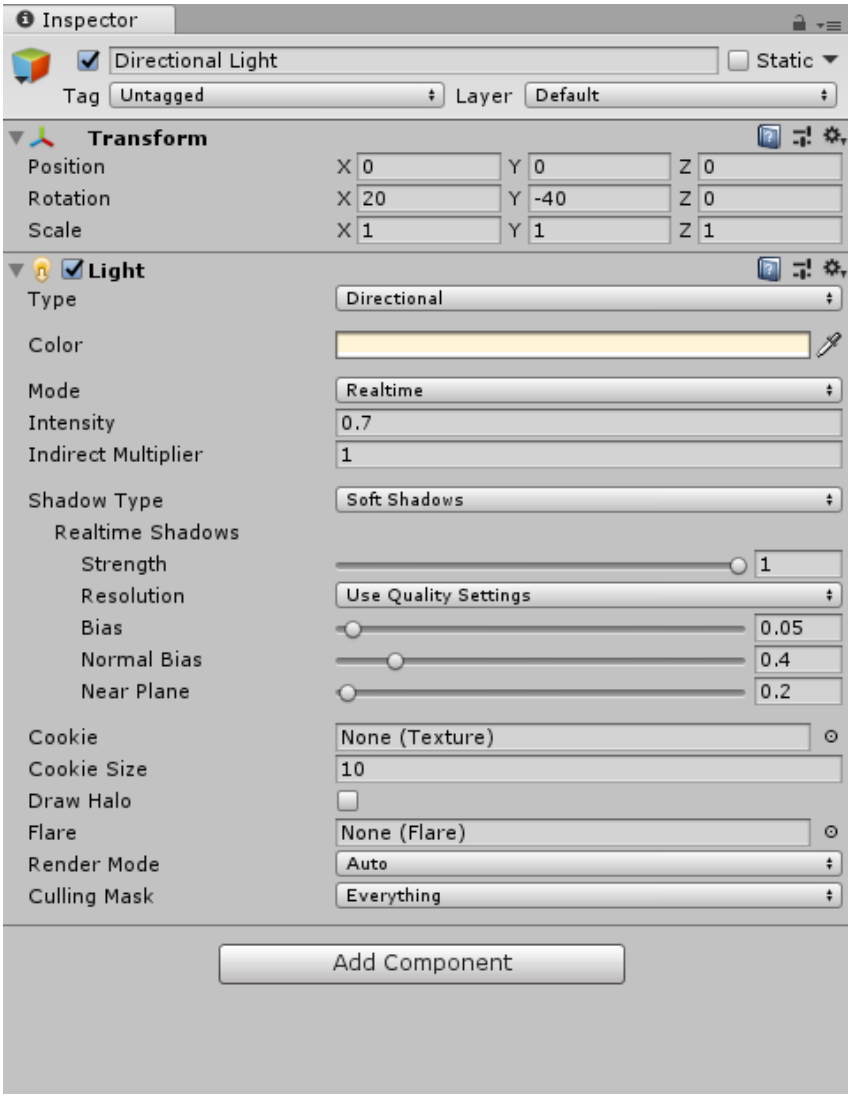
Prijašnja slika, koja je pretvorena iz 2D u „Cube“, povuče se i ubaci u „Cubemap“. Kreirani „Skybox“ ne vidi se na sceni zato što nije postavljen kao „Skybox Material“. Kako bi se postavio, iz izbornika „Window“ navigira se do opcije „Lightning“, a zatim se odabere opcija „Settings“. Otvori se novi prozor u kojem se klikne na kružić pokraj „Default-Skyboxa“ te se odabere željeni „Skybox“.



Slika 7.29 Postavljanje Skyboxa *na scenu*

7.7. Postavljanje osvjetljenja

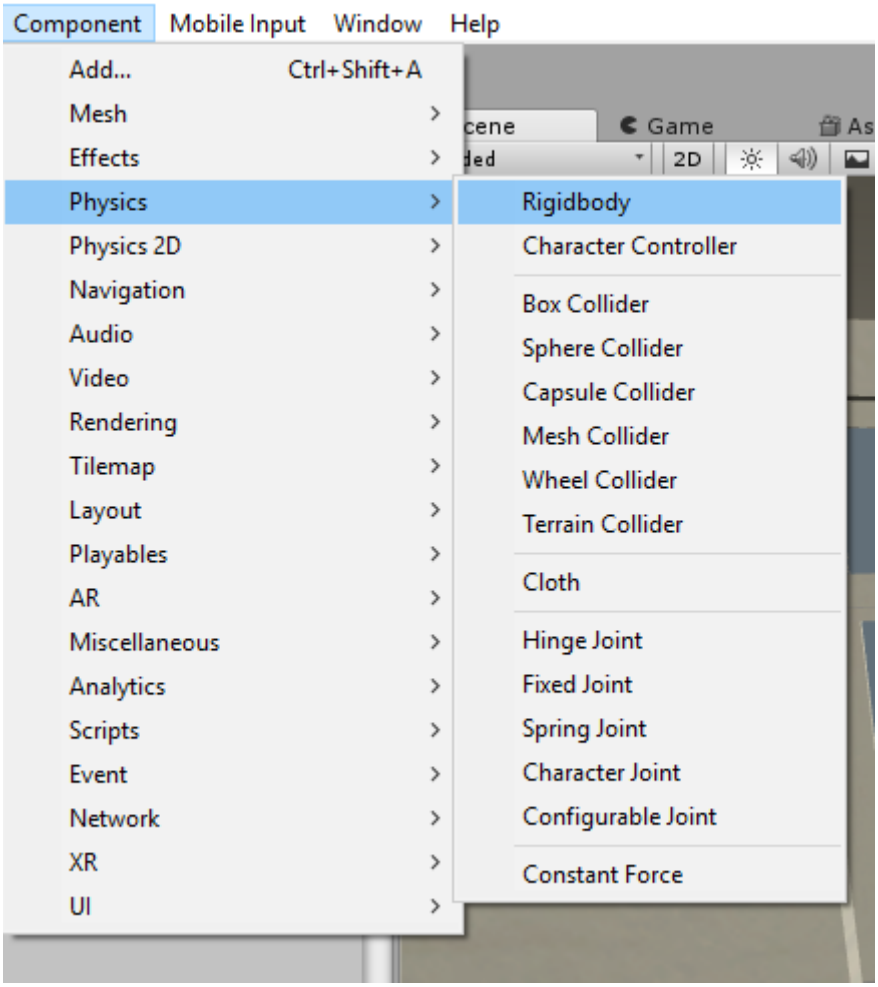
Nakon postavljanja „Skyboxa“ postavljeno je osvjetljenje. U sceni je već bila glavna kamera i glavno svjetlo, no oni nisu bili podešeni. U podešavanje osvjetljenja treba uložiti dosta vremena da bi se dobilo željeno osvjetljenje. Odabran je „Directional“ tip svjetla budući da se ponaša kao sunčevo svjetlo i intenzitet se ne smanjuje bez obzira na to koliko je objekt udaljen od svjetla. Za boju svjetla postavljena je žuta, slična suncu. Postavljeni način svjetla je „Realtime“ budući da najbolje odgovara mojoj sceni. Intenzitet svjetla je postavljen na 0,7 jer najbolje odgovara mojoj sceni. Tip, boja, intenzitet i rotacija svjetla postavljeni su na način da se dobije zalazak sunca. U podešavanje rotacije svjetla također treba uložiti dosta vremena. Osvjetljenje ovisi od scene do scene.



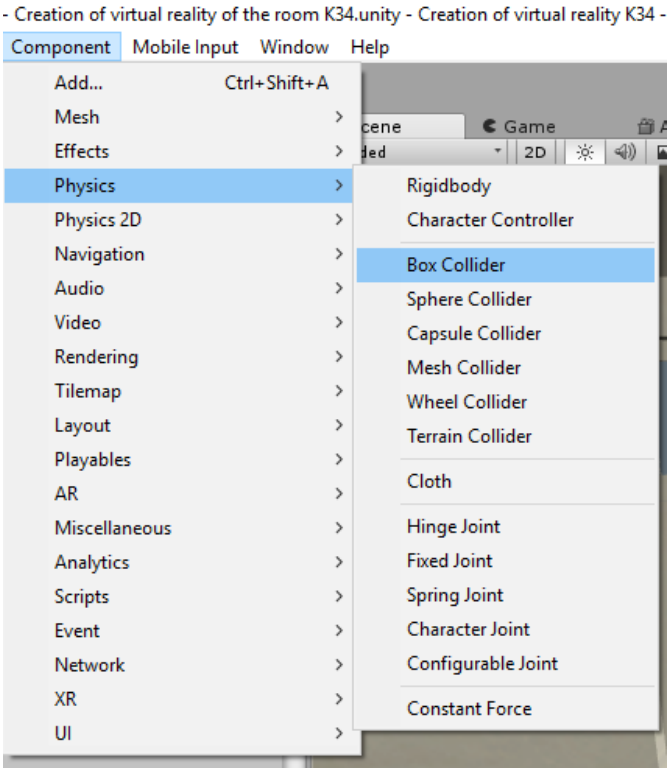
Slika 7.30 Postavke osvjetljenja

7.8. Postavljanje kolizije

Kolizija služi kako kamera ili karakter ne bi mogli prolaziti kroz 3D modele na sceni. Proces je postavljanja kolizije isti za svaki 3D model. U primjeru je pokazan taj proces. Prvo se označi 3D model oko kojeg želimo postaviti koliziju. Kako bi kolizija funkcionirala, na model moraju biti postavljeni „Rigidbody“ i „Collider“ odgovarajuće vrste. Vrsta „Collidera“ ovisi o tome je li 3D model ravan, okrugao ili valjkast. U ovom je projektu korišten „Box Collider“ budući da su 3D modeli na sceni više-manje ravne plohe. Iz izbornika „Component“ odabere se opcija „Physics“ pa zatim opcija „Rigidbody“. Opet se klikne na izbornik „Component“ i odabere se opcija „Physics“ pa opcija „Box Collider“.

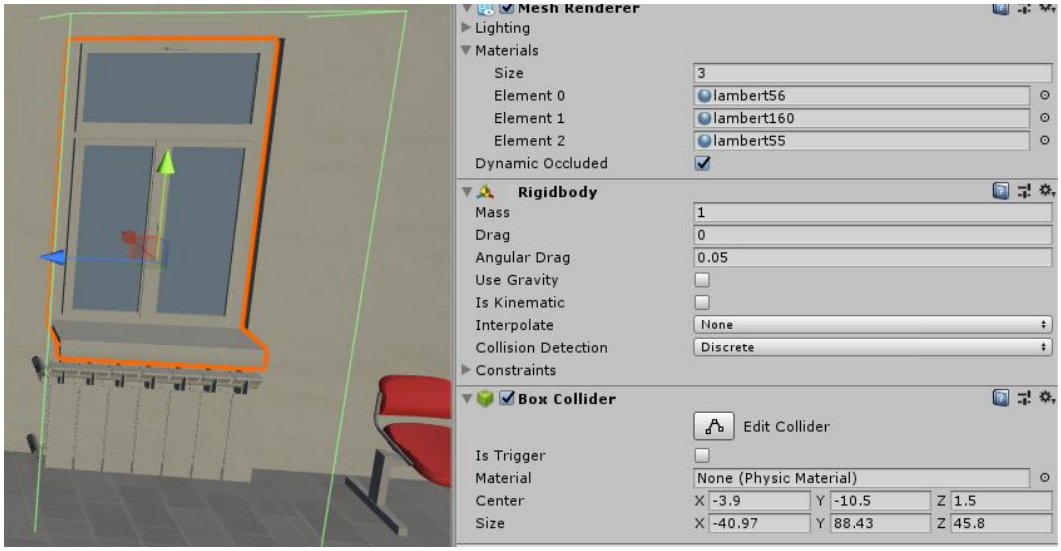


Slika 7.31 Postavljanje Rigidbodyja *na 3D model*



Slika 7.32 Postavljenje Box Collidera *na 3D model*

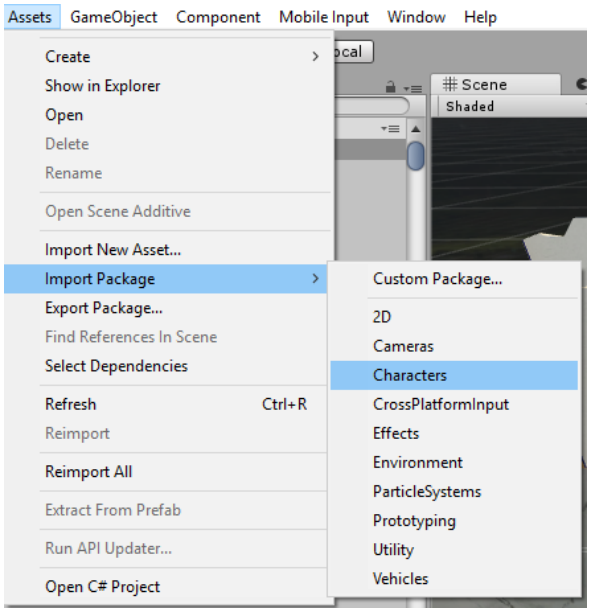
U kartici „Inspector“ pojave se opcije za uređivanje „Rigidbodyja“ i „Box Collidera“. Važno je da se na „Rigidbodyju“ isključi opcija „Use Gravity“ i „Is Kinematic“ budući da je 3D model statičan. U opcijama za „Box Collider“ posložimo visinu, širinu i debljinu kolizije.



Slika 7.33 Editiranje Rigidbodyja *i* Box Collidera

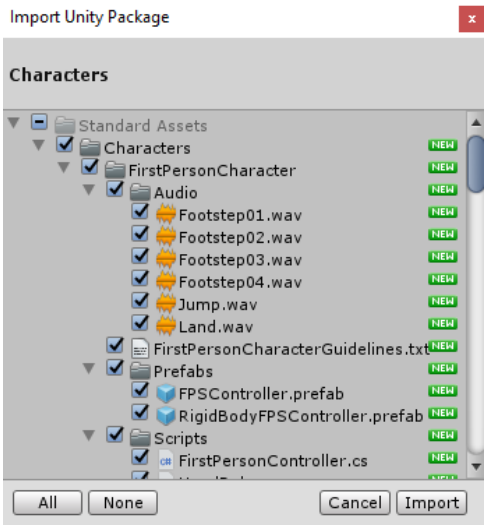
7.9. Kamera

Za kameru je korištena gotova Unity skripta. S obzirom na to da se radi o virtualnoj stvarnosti, na scenu je postavljena „First Person Character“ kamera. Iz izbornika „Assets“ navigira se do opcije „Import Package“ te se odabere opcija „Characters“.



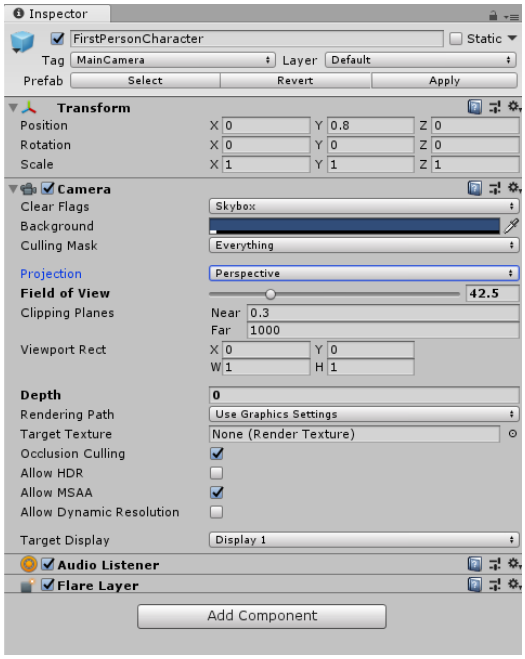
Slika 7.34 Kako uvesti Unity sadržaj

Otvori se novi prozor koji prikazuje sadržaj paketa. Odabere se željeni sadržaj te se klikne na „Import“.

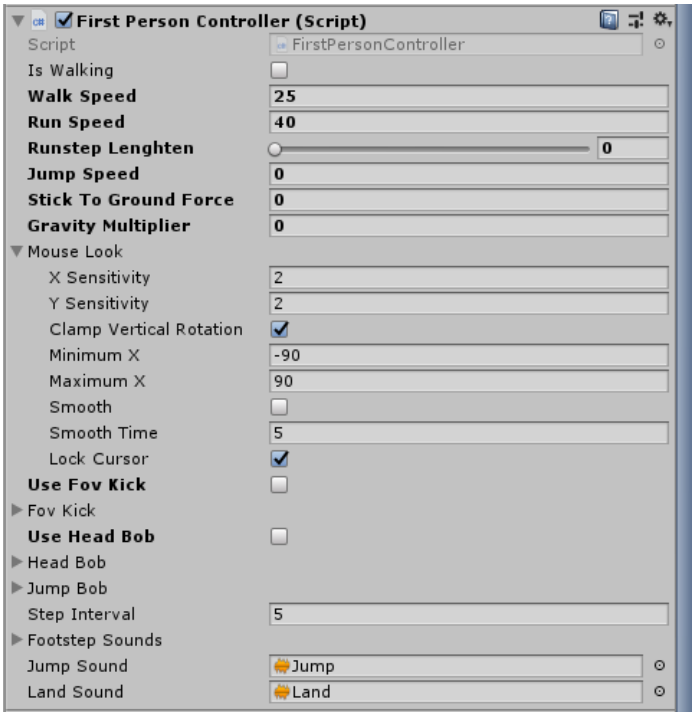


Slika 7.35 Character *prozor*

Nakon što je kamera ubačena u Unity, ta se kamera postavlja na scenu. To se napravi povlačenjem kamere u hijerarhijski prozor. Također, trebalo je podesiti i postavke kamere. Podesio sam vidno polje kamere, način projiciranja slike iz kamere, brzinu pomicanja, način rotacije kamere i senzitivnost miša.



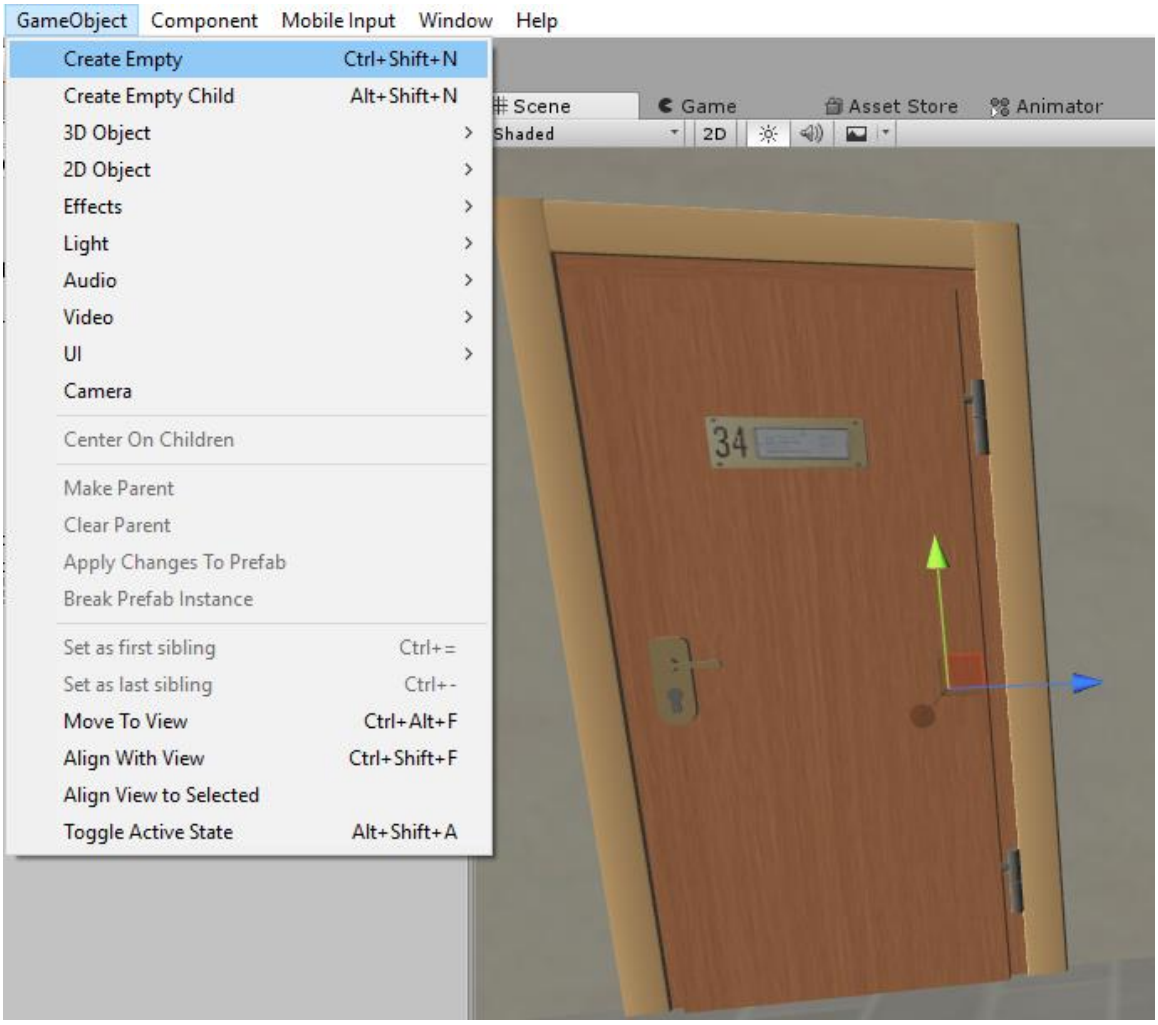
Slika 7.36 Postavke vidnog polja i projiciranja



Slika 7.37 Postavke hoda, rotacije i senzitivnosti

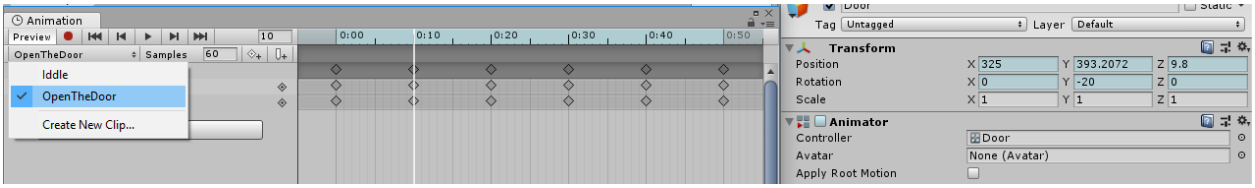
7.10. Animacija

Na vrata je postavljena animacija, a nakon toga je dodana skripta koja pritiskom na određenu tipku na tipkovnici pokreće tu animaciju. Kako bi se napravila animacija, najprije se doda prazan „GameObject“ te se postavi između držača za vrata. Kako bi se vrata grupirala s praznim „GameObjectom“, vrata se povuku unutar „GameObjecta“ na način da „GameObject“ predstavlja „Parent“, a vrata predstavljaju „Child“ .



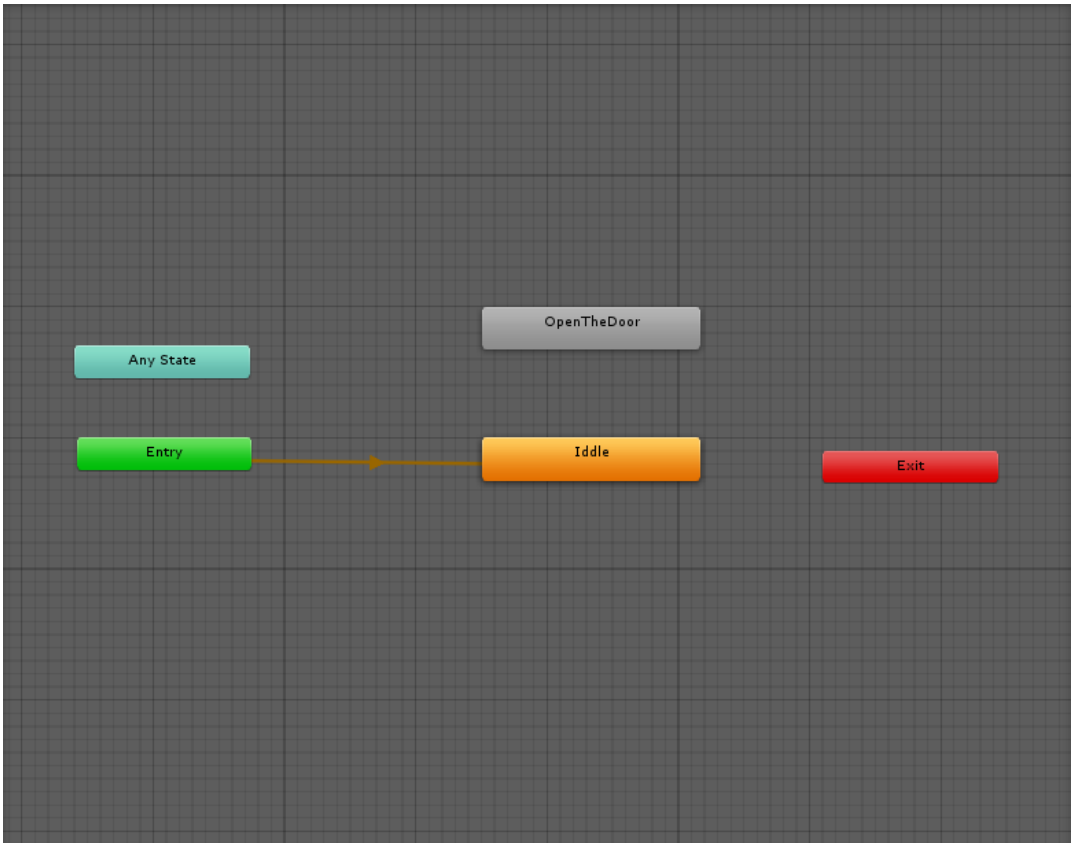
Slika 7.38 Dodavanje praznoga GameObjecta

Nakon toga označi se roditelj grupe i preko izbornika „Window“ odabere se opcija „Animation“ te se započinje s animacijom. Napravio sam dvije vrste animacije. Jedna animacija predstavljala je stanje mirovanja, a druga otvaranje vrata. Animacija koja je predstavljala stanje mirovanja bila je postavljena da ne radi ništa. Animacija koja je predstavljala otvaranje vrata bila je postavljena na način da se svakih deset sekundi y-os zarotira za -20 stupnjeva. Nakon rotacije pozicija vrata popravlja se prema potrebi.



Slika 7.39 Postavljanje animacije za otvaranje vrata

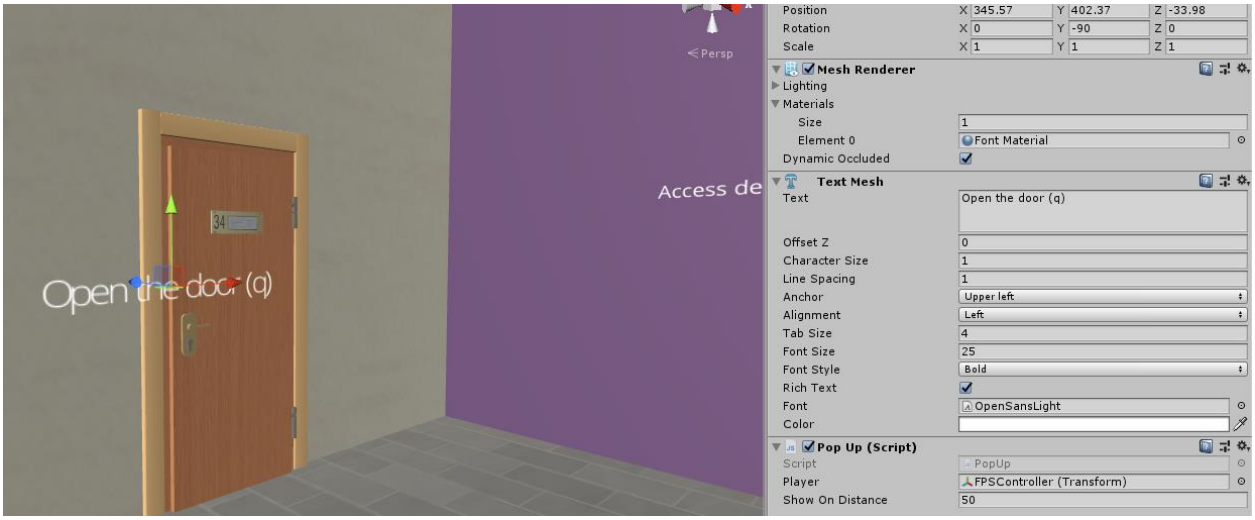
Nakon što su animacije bile postavljene u animatoru je trebalo postaviti stanja na animacije kako bi se dobile željene akcije. Zapravo, u animatoru je postavljena logika, odnosno tijek izvođenja animacija. Kada se uđe u igru, animacija vrata postavljena je u stanje mirovanja i ona ostaje u stanju mirovanja tako dugo dok se ne pritisne određena tipka na tipkovnici. Animacija za otvaranje vrata pod nazivom „OpenTheDoor“ se nalazi u animatoru, ali nije povezana s prijašnjom animacijom. Ona je povezana sa skriptom te se animacija izvršava samo onda kada se pritisne tipka na tipkovnici koja je definirana putem skripte. U suprotnom vrata su u stanju mirovanja kao što je prikazano u animatoru. Važno je još naglasiti da kod animacije „OpenTheDoor“ treba isključiti uzastopno ponavljanje kako bi se vrata otvarala samo putem skripte.



Slika 7.40 Tijek izvođenja animacije

7.11. Tekst

Kako bi se informiralo igrača o tome kako da napravi interakciju s vratima, dodan je tekst. Tekst je dodan desnim klikom na hijerarhiju te je iz izbornika „UI“ odabrana opcija „Text“. Nakon što je tekst ubačen, oblikovan je po želji putem „Inspectora“ koji se nalazi s desne strane. Nakon oblikovanja na tekst se primijenila skripta koja otkriva tekst kako se kamera približava vratima. Važno je napomenuti da u skripti mora biti definirano koji objekt na sceni predstavlja igrača da bi skripta znala koju udaljenost od objekta treba pratiti.



Slika 7.41 Uređivanje teksta

7.12. Interaktivnost korisnika i kontrola

Kao što je već gore napomenuto, interakcija korisnika i kontrola ostvarena je pomoću skripti. Skripta koja se koristi za kretanje kamere po sceni pisana je u C# jeziku i omogućuje kretanje kamere po sceni pomoću tipki W (gore), S (dolje), A (lijevo), D (desno) te rotiranje kamere lijevo, desno, gore i dolje pomoću miša. Skripta za otvaranje vrata također je pisana u C# jeziku i omogućuje korisniku da pomoću specifične tipke na tipkovnici napravi interakciju s vratima, odnosno otvori vrata. Skripta za prikaz teksta pisana je u JavaScript jeziku i informira korisnika na koji način da izvrši interakciju s objektima na sceni.

7.13. Skripte

1) Kamera

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;
using UnityStandardAssets.Utility;
using Random = UnityEngine.Random;

namespace UnityStandardAssets.Characters.FirstPerson
{
    [RequireComponent(typeof (CharacterController))]
    [RequireComponent(typeof (AudioSource))]
    public class FirstPersonController : MonoBehaviour
    {
        [SerializeField] private bool m_IsWalking;
        [SerializeField] private float m_WalkSpeed;
        [SerializeField] private float m_RunSpeed;
        [SerializeField] [Range(0f, 1f)] private float
         m_RunstepLenghten;
        [SerializeField] private float m_JumpSpeed;
        [SerializeField] private float m_StickToGroundForce;
        [SerializeField] private float m_GravityMultiplier;
        [SerializeField] private MouseLook m_MouseLook;
        [SerializeField] private bool m_UseFovKick;
        [SerializeField] private FOVKick m_FovKick = new FOVKick();
        [SerializeField] private bool m_UseHeadBob;
        [SerializeField] private CurveControlledBob m_HeadBob = new
         CurveControlledBob();
        [SerializeField] private LerpControlledBob m_JumpBob = new
         LerpControlledBob();
        [SerializeField] private float m_StepInterval;
        [SerializeField] private AudioClip[] m_FootstepSounds;
        [SerializeField] private AudioClip m_JumpSound;
        [SerializeField] private AudioClip m_LandSound;

        private Camera m_Camera;
        private bool m_Jump;
        private float m_YRotation;
        private Vector2 m_Input;
        private Vector3 m_MoveDir = Vector3.zero;
        private CharacterController m_CharacterController;
        private CollisionFlags m_CollisionFlags;
        private bool m_PreviouslyGrounded;
        private Vector3 m_OriginalCameraPosition;
        private float m_StepCycle;
        private float m_NextStep;
        private bool m_Jumping;
        private AudioSource m_AudioSource;
```

```
private void Start()
{
    m_CharacterController =
    GetComponent<CharacterController>();
    m_Camera = Camera.main;
    m_OriginalCameraPosition =
    m_Camera.transform.localPosition;
    m_FovKick.Setup(m_Camera);
    m_HeadBob.Setup(m_Camera, m_StepInterval);
    m_StepCycle = 0f;
    m_NextStep = m_StepCycle/2f;
    m_Jumping = false;
    m_AudioSource = GetComponent<AudioSource>();
    m_MouseLook.Init(transform , m_Camera.transform);
}


private void Update()
{
    RotateView();

    if (!m_Jump)
    {
        m_Jump =
        CrossPlatformInputManager.GetButtonDown("Jump");
    }

    if (!m_PreviouslyGrounded &&
         m_CharacterController.isGrounded)
    {
        StartCoroutine(m_JumpBob.DoBobCycle());
        PlayLandingSound();
        m_MoveDir.y = 0f;
        m_Jumping = false;
    }
    if (!m_CharacterController.isGrounded && !m_Jumping &&
         m_PreviouslyGrounded)
    {
        m_MoveDir.y = 0f;
    }

    m_PreviouslyGrounded = m_CharacterController.isGrounded;
}


private void PlayLandingSound()
{
    m_AudioSource.clip = m_LandSound;
    m_AudioSource.Play();
    m_NextStep = m_StepCycle + .5f;
}


private void FixedUpdate()
{
    float speed;
    GetInput(out speed);
```

```
    Vector3 desiredMove = transform.forward*m_Input.y +
    transform.right*m_Input.x;

    RaycastHit hitInfo;
    Physics.SphereCast(transform.position,
                       m_CharacterController.radius,
                       Vector3.down, out hitInfo,
                       m_CharacterController.height/2f,
                       Physics.AllLayers,
                       QueryTriggerInteraction.Ignore);
    desiredMove = Vector3.ProjectOnPlane(desiredMove,
                  hitInfo.normal).normalized;

    m_MoveDir.x = desiredMove.x*speed;
    m_MoveDir.z = desiredMove.z*speed;


    if (m_CharacterController.isGrounded)
    {
        m_MoveDir.y = -m_StickToGroundForce;

        if (m_Jump)
        {
            m_MoveDir.y = m_JumpSpeed;
            PlayJumpSound();
            m_Jump = false;
            m_Jumping = true;
        }
    }
    else
    {
        m_MoveDir += Physics.gravity*m_GravityMultiplier*
                     Time.fixedDeltaTime;
    }
    m_CollisionFlags = m_CharacterController.Move
                       (m_MoveDir*Time.fixedDeltaTime);

    ProgressStepCycle(speed);
    UpdateCameraPosition(speed);

    m_MouseLook.UpdateCursorLock();
}


private void PlayJumpSound()
{
    m_AudioSource.clip = m_JumpSound;
    m_AudioSource.Play();
}


private void ProgressStepCycle(float speed)
{
    if (m_CharacterController.velocity.sqrMagnitude > 0 &&
       (m_Input.x != 0 || m_Input.y != 0))
    {
        m_StepCycle += (m_CharacterController.velocity.
```

```
                        magnitude + (speed*(m_IsWalking ?
                        1f : m_RunstepLenghten)))*
                        Time.fixedDeltaTime;
    }

    if (!(m_StepCycle > m_NextStep))
    {
        return;
    }

    m_NextStep = m_StepCycle + m_StepInterval;

    PlayFootStepAudio();
}


private void PlayFootStepAudio()
{
    if (!m_CharacterController.isGrounded)
    {
        return;
    }
    int n = Random.Range(1, m_FootstepSounds.Length);
    m_AudioSource.clip = m_FootstepSounds[n];
    m_AudioSource.PlayOneShot(m_AudioSource.clip);
    m_FootstepSounds[n] = m_FootstepSounds[0];
    m_FootstepSounds[0] = m_AudioSource.clip;
}


private void UpdateCameraPosition(float speed)
{
    Vector3 newCameraPosition;
    if (!m_UseHeadBob)
    {
        return;
    }
    if (m_CharacterController.velocity.magnitude > 0 &&
        m_CharacterController.isGrounded)
    {
        m_Camera.transform.localPosition =
        m_HeadBob.DoHeadBob (m_CharacterController.velocity
                             .magnitude +(speed*(m_IsWalking
                             ? 1f : m_RunstepLenghten)));
        newCameraPosition = m_Camera.transform.localPosition;
        newCameraPosition.y = m_Camera.transform
                              .localPosition.y -
                              m_JumpBob.Offset();
    }
    else
    {
        newCameraPosition = m_Camera.transform.localPosition;
        newCameraPosition.y = m_OriginalCameraPosition.y -
                              m_JumpBob.Offset();
    }
    m_Camera.transform.localPosition = newCameraPosition;
}
```

```
        private void GetInput(out float speed)
        {
            float horizontal = CrossPlatformInputManager.GetAxis
                                ("Horizontal");
            float vertical = CrossPlatformInputManager.GetAxis
                            ("Vertical");

            bool waswalking = m_IsWalking;

         #if !MOBILE_INPUT
            m_IsWalking = !Input.GetKey(KeyCode.LeftShift);
         #endif
            speed = m_IsWalking ? m_WalkSpeed : m_RunSpeed;
            m_Input = new Vector2(horizontal, vertical);

            if (m_Input.sqrMagnitude > 1)
            {
                m_Input.Normalize();
            }

            if (m_IsWalking != waswalking && m_UseFovKick &&
                m_CharacterController.velocity.sqrMagnitude > 0)
            {
                StopAllCoroutines();
                StartCoroutine(!m_IsWalking ? m_FovKick.FOVKickUp():
                m_FovKick.FOVKickDown());
            }
        }


        private void RotateView()
        {
            m_MouseLook.LookRotation (transform, m_Camera.transform);
        }


        private void OnControllerColliderHit(ControllerColliderHit
                                             hit)
        {
            Rigidbody body = hit.collider.attachedRigidbody;
            if (m_CollisionFlags == CollisionFlags.Below)
            {
                return;
            }

            if (body == null || body.isKinematic)
            {
                return;
            }

body.AddForceAtPosition(m_CharacterController.velocity*0.1f,
hit.point, ForceMode.Impulse);
        }
    }
}
```

Kod u nastavku ovog rada prilagođen je prema „CubicBrain“ koji se nalazi u popisu literature pod brojem 27

2) Otvaranje vrata

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

public class Doorcontroller : MonoBehaviour
{

    private Animator anim;

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

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown("q"))
        {
            anim.Play("OpenTheDoor");

        }
    }
}
```

Kod u nastavku ovog rada prilagođen je prema „Unity3D WAREHOUSE“ koji se nalazi u popisu literature pod brojem 28

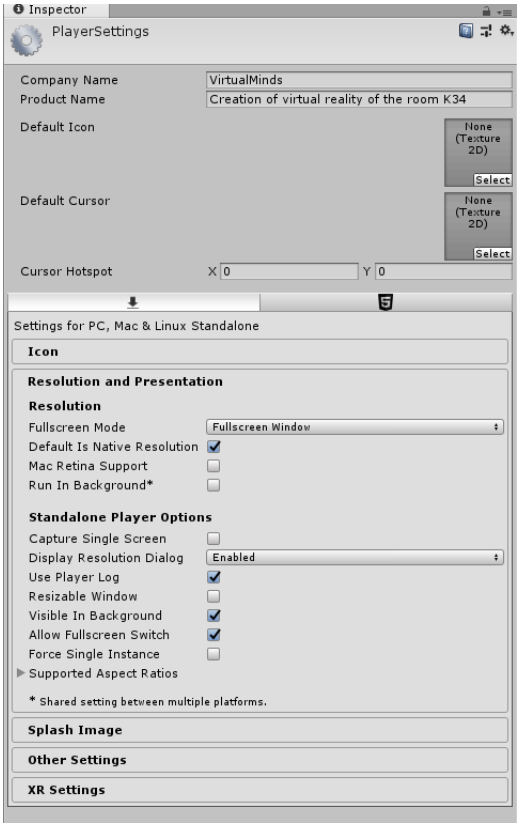
3) Pojava teksta

```
#pragma strict

var player: Transform;
var showOnDistance: float = 2;
private var textMesh: MeshRenderer;

function Start ()
{
  textMesh = gameObject.GetComponent(MeshRenderer);
}

function Update ()
{

  if (Vector3.Distance(transform.position, player.position) <
      showOnDistance)
```

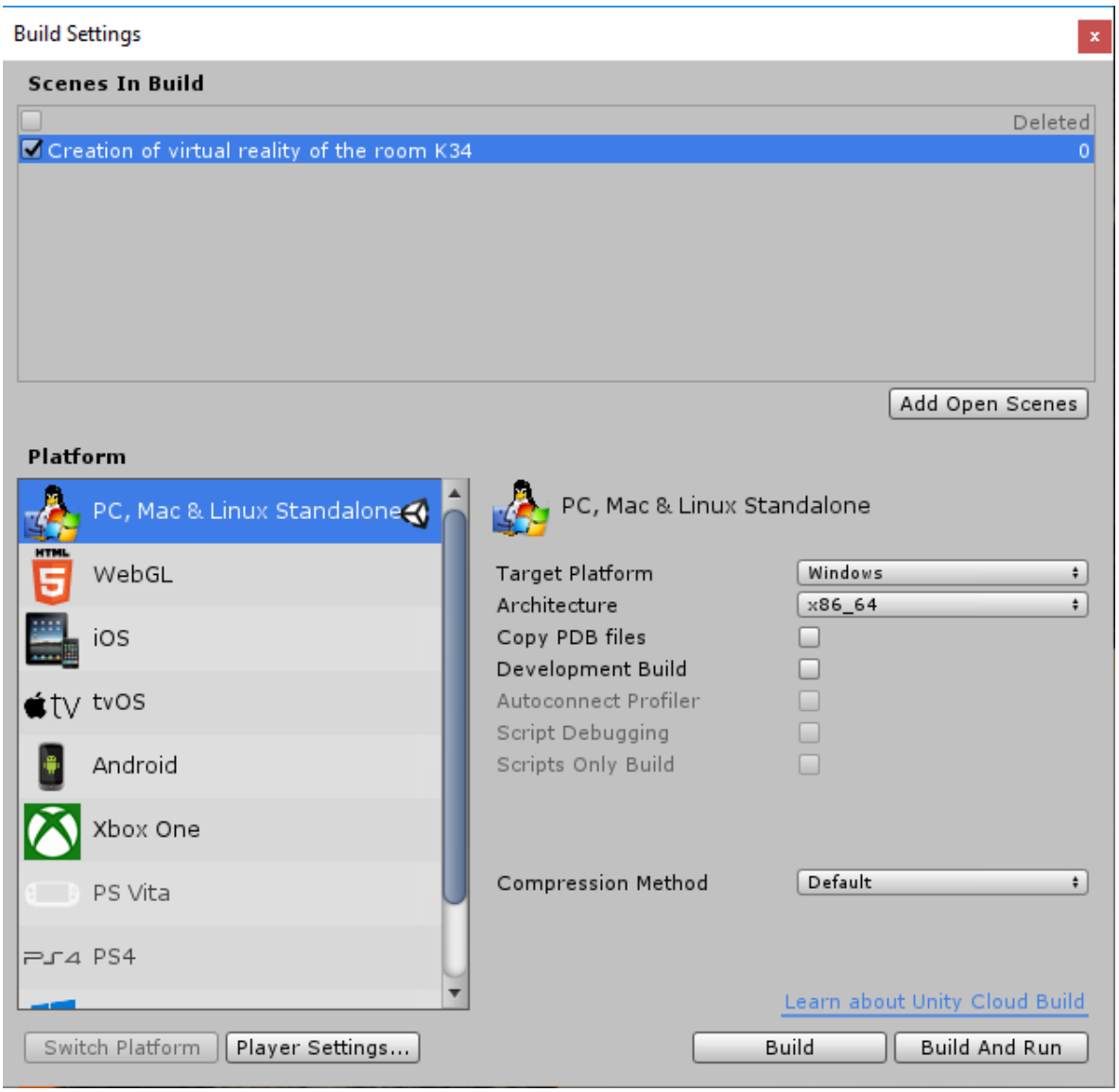
```
    textMesh.enabled = true;

  else
     textMesh.enabled = false;

}
```

7.14. Izrada aplikacije

Nakon završetka izrade projekta, taj projekt trebalo je pretvoriti u aplikaciju. Prije pretvorbe projekta u aplikaciju treba posložiti postavke projekta. Klikom na izbornik „Edit“ navigira se mišem do „Project Settingsa“ te se odabire opcija „Player“. U „PlayerSettings“ kartici možemo postaviti ime kompanije i aplikacije te sliku. Također, možemo konfigurirati rezoluciju, zahtjeve sustava koji su potrebni da bi se aplikacija pokrenula te optimizirati aplikaciju. Postavke projekta bile su posložene i projekt je bio spreman za pretvaranje u aplikaciju. Iz izbornika „File“ odabere se opcija „Build Settings“. Otvori se prozor u kojem trebamo označiti scenu iz koje želimo napraviti aplikaciju. Nakon toga se odabere platforma za koju će ta aplikacija biti namijenjena. Nakon odabira platforme odabere se sustav na kojem će aplikacija raditi. Također, moramo specificirati i arhitekturu sustava. Kada smo zadovoljni s postavkama, kliknemo mišem na gumb „Build“ te pričekamo nekoliko minuta da Unity napravi aplikaciju.



Slika 7.42 Postavke aplikacije



Slika 7.43 Postavke za pretvorbu projekta u aplikaciju

8. Zaključak

Cilj ovog rada bio je prikazati mogućnosti Unity alata te postupke nužne za izradu virtualne stvarnosti. Za realizaciju ovog rada korišteno je više razvojnih programa. Za izradu 3D scene korišten je program Autodesk Maya. 3D scena napravljena je poligonalnim modeliranjem. Za izradu skripti korišten je program Microsoft Visual Studio. Skripte su pisane C# i JavaScript jezikom. Za dovršetak projekta korišten je Unity. Budući da Unity ima širok izbor snažnih alata, u njemu su napravljene interakcije, osvjetljenje i animacija te je postavljena kolizija i kamera. S obzirom na alate i mogućnosti te jednostavnost koju pruža Unity, možemo zaključiti da je Unity odličan izbor prilikom izrade projekata i aplikacija te aplikacija za virtualnu stvarnost. Također, možemo zaključiti da su svi navedeni razvojni programi korisniku znatno olakšali posao prilikom izrade složenijih i kompliciranijih dijelova projekata te mu omogućili da brže i lakše napravi složenije stvari.

U današnje vrijeme za izradu virtualne stvarnosti potrebno je više različitih programa, kao što su Autodesk Maya, Unity, Microsoft Visual Studio i mnogi drugi slični navedenim programima, te znanja u području 3D modeliranja, teksturiranja, programiranja i virtualne stvarnosti. Zahvaljujući odlično razvijenim programima i tehnologiji, svaka osoba koja zna osnove 3D modeliranja, virtualne stvarnosti i programiranja može napraviti aplikaciju za virtualnu stvarnost. Pomoću 3D tehnologije možemo stvoriti realističan ili manje realističan svijet oko sebe, a sve ovisi o tome što želimo postići. Pomoću virtualne stvarnosti možemo prikazati zamišljenu okolinu te prizore koji u stvarnosti nisu mogući, ali isto tako i simulirati stvarni svijet oko sebe. U virtualnoj stvarnosti nemamo ograničenja, dok smo u stvarnom svijetu ograničeni vremenom, prostorom i resursima. Virtualna stvarnost ima široku primjenu pa iz toga možemo zaključiti da se može primijeniti u svim segmentima ljudskog života, kao što je obavljanje različitih poslova, obrazovanje, industrija, vojska, sport, turizam i slično. Zapravo, ona se može primijeniti u svim područjima u kojima je potrebna praksa i eksperimentiranje. Jedna od prednosti virtualne stvarnosti je stjecanje iskustva bez posljedica na stvarni svijet budući da je svijet virtualne stvarnosti računalno stvoren. No bez obzira na to nudi osjećaj stvarnog pa se može steći pravo iskustvo. Uz sve navedeno možemo zaključiti da će u bliskoj budućnosti virtualna stvarnost biti od sve većeg značaja u našemu svakodnevnom životu.

U Varaždinu, _____ _____

Datum Potpis

Sveučilište
Sjever

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, MARKO ROGINA *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom IZRADA VIRTUALNE STVARNOSTI - K34 *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Marko Rogina
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, MARKO ROGINA *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom IZRADA VIRTUALNE STVARNOSTI - K34 *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Marko Rogina

9. Literatura

[1] Margaret Rouse, virtual reality, dostupno na: https://whatis.techtarget.com/definition/virtual-reality

[2] ThinkMobiles: Virtual Reality in Gaming, dostupno na: https://thinkmobiles.com/blog/virtual-reality-gaming/

[3] Autodesk Maya 2014: Introduction, dostupno na: http://download.autodesk.com/us/maya/Maya_2014_GettingStarted/index.html?url

[4] Dean Takahashi: John Riccitiello sets out to identify the engine of growth for Unity Technologies, dostupno na: https://venturebeat.com/2014/10/23/john-riccitiello-sets-out-to-identify-the-engine-of-growth-for-unity-technologies-interview/

[5] Unity: Unity 5.0, dostupno na: https://unity3d.com/unity/whats-new/unity-5.0

[6] Genevieve Warren, Gordon Hogenson, Kemp Brown, Saisang Cai, Greg Van Liew, Matthew Sebolt, Mike Jones, Mike B., Kraig Brockschmidt, tglee, Colin Robertson: Visual Studio overview, dostupno na: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide

[7] Justin Slick: What Is 3D Modeling?, dostupno na: https://www.lifewire.com/what-is-3d-modeling-2164

[8] ARCHI CGI: WHAT IS A 3D MODELING? THINGS YOU HAVE TO KNOW NOWADAYS, dostupno na: https://archicgi.com/3d-modeling-things-youve-got-know/

[9] Margaret Rouse: 3D modeling, dostupno na: https://whatis.techtarget.com/definition/3D-modeling

[10] Josh Petty: What is 3D Modeling & What's It Used For?, dostupno na: https://conceptartempire.com/what-is-3d-modeling/

[11] A. Bernik, D. Kelnarić: Vrste i tehnike 3D teksturiranja, dostupno na: https://www.unin.hr/data/knjiznica/tehnicki_glasnik/tehnickiglasnik_1_2011.pdf

[12] WOW HOW: 3D TEXTURING, dostupno na: https://wow-how.com/3d-texturing

[13] Virtual Reality Society: What is Virtual Reality?, dostupno na: https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html

[14] Alayna Mansell: 5 Uses for Virtual Reality, dostupno na: https://www.fdmgroup.com/5-uses-for-virtual-reality/

[15] Virtual Reality Society: Applications Of Virtual Reality, dostupno na: https://www.vrs.org.uk/virtual-reality-applications/

[16] The Franklin Institute: History of Virtual Reality, dostupno na: https://www.fi.edu/virtual-reality/history-of-virtual-reality

[17] The Franklin Institute: The Science of Virtual Reality, dostupno na: https://www.fi.edu/virtual-reality/the-science-of-virtual-reality

[18] Unity: Learning the interface, dostupno na: https://docs.unity3d.com/Manual/LearningtheInterface.html

[19] Unity: The Project Window, dostupno na: https://docs.unity3d.com/Manual/ProjectView.html

[20] Unity: The Scene View, dostupno na: https://docs.unity3d.com/Manual/UsingTheSceneView.html

[21] Unity: The Game view, dostupno na: https://docs.unity3d.com/Manual/GameView.html

[22] Unity: The Hierarchy window, dostupno na: https://docs.unity3d.com/Manual/Hierarchy.html

[23] Unity: The Inspector window, dostupno na: https://docs.unity3d.com/Manual/UsingTheInspector.html

[24] Unity: Positioning GameObjects, dostupno na: https://docs.unity3d.com/Manual/PositioningGameObjects.html

[25] Unity: Lights, dostupno na: https://docs.unity3d.com/Manual/Lights.html

[26] Unity: Types of light, dostupno na: https://docs.unity3d.com/Manual/Lighting.html

[27] CubicBrain: Unity3D – Trigger animations (keypress trigger), dostupno na: https://www.youtube.com/watch?v=N73EWquTGSY

[28] Unity3D WAREHOUSE: Unity 3D Simple Pop Up Text Tutorial, dostupno na: http://unity3dmgames.blogspot.com/2015/09/unity-3d-simple-pop-up-text-tutorial.html

Popis slika

Prilozi:

CD