Izrada 3D računalne igre u Unity razvojnom okruženju

Jagodić, Antonio

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University North / Sveučilište Sjever**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:122:273776

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

University North Digital Repository

Sveučilište
Sjever

Završni rad br. 443/MM/2015

Izrada 3D računalne igre u Unity razvojnom okruženju

Antonio Jagodić, 4686/601

Varaždin, rujan 2016. godine

Sveučilište Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 443/MM/2015

Izrada 3D računalne igre u Unity razvojnom okruženju

Student

Antono Jagodić, 4686/601

Mentor

pred, Andrija Bernik, dipl.inf.

Varaždin, rujan 2016. godine

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

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za multimediju, oblikovanje i primjenu

PRISTUPNIK: Antonio Jagodić | MATIČNI BROJ: 4686/601

DATUM: 11.09.2015. | KOLEGIJ: 3D modeliranje

NASLOV RADA: Izrada 3D računalne igre u Unity razvojnom okruženju

MENTOR: Andrija Bernik, dipl.inf. | ZVANJE: Predavač

ČLANOVI POVJERENSTVA:

1. mr.sc. Dragan Matković, v. predavač - predsjednik
2. mr.sc. Vladimir Stanisavljević, v. predavač - član
3. pred. Andrija Bernik, dipl.inf. - mentor
4. izv.prof.dr.sc. Damir Vusić - zamjenski član
5.

VZ MK

MM I

Zadatak završnog rada

BROJ: 443/MM/2015

OPIS

Zadatak ovog rada je napraviti PC igru korištenjem dva osnovna programa Autodesk Maya i Unity. Ovaj rad prikazuje jednu jednostavnu igru, preko koje se objašnjavaju animacija, modeliranje, skripte i ostali dijelovi za izradu igre. Upoznaje se s osnovnim dijelovima tih programa, povijesti igara, game engine-a, računalnim igrama i njihovim zahtjevima.

Kroz praktični dio će se izraditi igra kroz spomenute programe. Detaljno će se opisati svaki korak koji se koristi u svakom programu. U Autodesk Mayi će se kreirati 3D modeli i objekti koji će se koristiti u igri. Modeliranjem i teksturiranjem će se stvoriti željeni izgled levela u igri. Nastavak rada se prebacuje na program Unity u kojeg će se prebaciti svi modeli i objekti iz Maye. Unity služi da bi na objekte primijenili skripte, odnosno da bi određeni objekti postali dinamični, te da se mogu pokretati željenim prostorom i tako stvorili igru. Dodat će se i zvuk i određeni glasovi koji bi dali što bolju i realističniju igru. Igru će se moći igrati kroz 2 levela, sa početnim menijem. U meniju se podešava jačina zvuka, pokretanje i izlazak iz igre. Svaki level će biti dizajniran na svoj način u kojemu će se kupiti određeni objekti, izbjegavati protivnike i dolazak do određenog mjesta da bi se završio level.

U radu je potrebno prikazati:
- opis programa Autodesk Maya,
- opis programa Unity,
- opis korištenih alata,
- opis izrade modela,
- opis postupka izrade igre u Unity-u.

Pisani rad obuhvaćati će tehničko rješenje uz popratne grafičke prikaze, te će se doprinos ovoj temi analizirati u zaključku rada.

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER
KOPRIVNICA

ZADATAK URUČEN: 25.09.2015. | POTPIS MENTORA: Bernik

Predgovor

Temu ovog završnog rada odabrao sam iz osobnih interesa i istraživanja kako se rade igre i koji proces moraju proći da bi dobili finalni sadržaj. Cilj je bio steći novo znanje i upoznati se sa novim programom za izradu igre, te kroz razne stručne radove i knjige saznati što više o tome području.

Zahvaljujem mentoru Andriji Berniku pred.dipl.inf. koji je pratio cijeli proces nastajanja završnog rada i svojim savjetima i entuzijazmom me usmjeravao kako da riješim probleme koji bi se pojavili prilikom izrade završnog rada.

Također zahvaljujem roditeljima i prijateljima koji su mi bili podrška tijekom studiranja.

Sažetak

Ovim radom je prikazan proces izrade jednostavne PC igre, te alati i programi koji se koriste u njezinoj izradi. Objašnjeni su osnovni pojmovi svakog korištenog programa, njihovi zahtjevi, osnovni dijelovi, povijest i dr.

Igre su najčešće napravljene i razvijene od strane timova koje mogu činiti manja grupa ljudi od nekoliko njih ili čak sam pojedinac. Manji timovi se bave razvojem „lakših“ i manje zahtjevnijim projektima kao što su igre za mobilne plaforme. Veliki timovi najčešće rade za izdavače videoigara. Izdavači videoigara su tvrtke koje su pokrenule i napravile igru koju su sami razvili. Svatko u timu ima neku ulogu u razvoju igre. Dizajneri stvaraju ciljeve, pravila, izgled i ulogu igre, odgovorni su za sve aspekte razvoja igre (od njezinog početka i samog kraja), stvaraju vizualne elemente igre (likove, krajolike, objekte, vozila, odjeću i dr.). Pored navedenog stvaraju i sam koncept igre, odnosno njezinu priču. Za sve kodove i skripte koje su potrebne u igri zaduženi su programeri. Različite platforme imaju različite zahtjeve programiranja, a unutar same igre moraju se ispunjavati zahtjevi kao što su: fizika, umjetna inteligencija, 3D razvoj sučelja, sustav kontrole itd. Dizajneri razina (nivoa) u igri su osobe koje su zadužene za arhitekturu igre. Koriste igru i mijenjaju njezin izgled i kompleksnost igre, definiraju predmete i likove koji su uključeni u igri, njihovo ponašanje prilikom igranja, izazove kao što su zapreke, mjesta za skrivanje, testove vještine i druge elemente za interakciju. Isto tako dosta bitan je audio dizajner koji stvara glazbu, zvučne efekte, karakter glasova, buku, vozila itd. U razvoju igre to su najbitnije uloge, dok su u većim timovima i veće podjele kao što su: animatori, pomoćni producenti, kreativni direktor, glavni direktor, glavni programer, osoba zadužena za marketing, voditelj proizvoda, voditelj projekta, tehnički umjetnik i dr.

Ključne riječi: Aplikacija, Game engine, Grafičko korisničko sučelje, Skripte, Sustav čestica, Unity, Video igra

Summary

This paper presents the process of making a simple PC games, and tools and programs that are used in its preparation. It explains the basic concepts of each used programs, their requirements, basic parts, history and others.

The games are usually designed and developed by teams that may seem small group of people from several of them, or even the individual himself. Smaller teams are engaged in the development of "lighter" and less demanding applications such as games for mobile plaform. Large teams often work for publishers of video games. Video game publishers are companies that have launched and made the game that they have developed themselves. Everyone on the team has a role in the development of the game. Designers create goals, rules, look and role play, are responsible for all aspects of the game (from its beginning and the very end), creating the visual elements of the game (characters, landscapes, buildings, cars, clothes, etc.). Additionally create the concept of the game, and her story. For all codes and scripts that are needed in the game are responsible developers. Different platforms have different requirements programming, and within the game itself must meet requirements such as physics, artificial intelligence, 3D development interface, control system, etc. Designers level (level) in the game are the persons responsible for the architecture of the game. Using the game and changing its appearance and complexity of the game, define the objects and characters involved in the game, their behavior during gameplay challenges such as obstacles, hiding places, testing skills and other elements to interact. Also quite important is a sound designer who creates music, sound effects, character voices, noises, vehicles and so on. In the development of games that are the most important roles, while in larger teams and more divisions such as animators, assistant producer, creative director, CEO, chief programmer, the person responsible for marketing, product manager, project manager, technical artist and others.

Key words: Application, Game engine, Graphical user interface, Autodesk Maya, Unity, Video game

Sadržaj

1. Uvod 1
2. Igre i softveri za razvoj igara (eng. game engine) kroz povijest 2
3. Autodesk Maya 5
3.1. Modeliranje 6
3.2. Teksturiranje 7
3.3. UV mapiranje 10
3.4. Svjetla 11
3.5. Animacija 17
3.6. Prednosti Autodesk Maye, Maya LT i konkurencija 18
4. Unity 22
4.1. Osnovni dijelovi Unitya 23
4.2. Svjetla 25
4.3. Skriptiranje 27
4.4. Grafička korisnička sučelja 28
4.5. Prednosti i nedostatci Unitya 29
4.6. Konkurencija 31
4.7. Kompatibilinost Unity s Mayom , formati i sistemske karakteristike 31
4.8. Sjenčanja, materijali i teksture 33
4.9. Zvuk 34
4.10. Animacije 35
4.11. Sustav čestica 36
5. Računalne igre i zahtjevi 37
5.1. Ograničenja poligona 37
5.2. Grafičke performanse (CPU i GPU) 37
5.3. Grubo modeliranje 38
6. Maya – praktični dio 39

6.1. Snjegović 40
6.2. Neprijatelj (eng. Enemy) 41
6.3. Drvo 46
6.4. Ostali modeli 47
7. Unity – praktični dio 49
7.1. Objekti, komponente i izgled levela 50
7.2. Glavni igrač (player) 53
7.3. Novčić 54
7.4. Neprijatelj i Snjegović 55
7.5. Ostali pomični objekti 59
7.6. Teren i Nebo 61
7.7. Kamera 63
7.8. GUI – grafičko korisničko sučelje 64
7.9. Kraj levela i kontrolna točka (checkpoint) 65
7.10. Skripte 67
7.11. Izrada aplikacije (igre) 74
8. Zaključak 75
9. Literatura 76
10. Popis slika 80

1. Uvod

Cilj ovog rada je napraviti PC igru korištenjem dva osnovna programa Autodesk Maya i Unity i prikazuje jednu jednostavnu igru, preko koje se objašnjavaju animacija, modeliranje, skripte i ostali dijelovi za izradu igre. Kroz teoriju upoznaje se s osnovnim dijelovima tih programa, povijesti igara, game engine-a, računalnim igrama i njihovim zahtjevima.

Kroz poglavlje 2. pokazuje se kako su se igre i njihovi softveri (game engini) razvijali, koje su njihove prednosti i mane, što se moglo postići, poznate tvrtke kao što je Apple i dr. Kroz poglavlje 3. objašnjava se detaljnije o softveru Autodesk Maya koji omogućuje trodimenzionalno modeliranje, animacije, simulacije, renderiranje, teksturiranje. Dijelovi kroz koje se prolazi su: modeliranje, teksturiranje, UV mapiranje, svijetla i animacija, prednosti Maye u odnosu na ostale programe i podržanost alata. U poglavlju 4. upoznaje se softver Unity koji služi za izradu računalnih i mobilnih igara. Saznaje se sučelje, osnovni dijelovi, svjetla, skriptiranje, UI – korisnička sučelja, ulazni formati u program, kompatibilnost s Mayom i konkurencija na tržištu.

U praktičnom dijelu rada izrađuje se igra kroz spomenute programe. Detaljno se opisuje svaki korak koji je korišten u svakom programu. U Autodesk Mayi kreirani su 3D modeli i objekti koji se koriste u igri. Modeliranjem i teksturiranjem stvoren je željeni izgled levela u igri. Nastavak rada prebacuje se na program Unity u kojeg se prebacuju svi modeli i objekti iz Maye. Unity služi da se na objekte primijene skripte, odnosno da bi određeni objekti postali dinamični, te da se mogu pokretati željenim prostorom i tako stvorili igru. Dodaje se zvuk i određeni glasovi koji bi dali što bolju i realističniju igru. Igru se može igrati kroz 2 levela, sa početnim menijem. U meniju se podešava jačina zvuka, pokretanje i izlazak iz igre. Svaki level je dizajniran na svoj način u kojemu se kupuju određeni objekti, izbjegavaju protivnici i dolazak do određenog mjesta da bi se završio level.

2. Igre i softveri za razvoj igara (eng. game engine) kroz povijest

Sve je krenulo 1952. kada je Alexander Douglas napravio igru „tic – tac – toe“ odnosno XOX igra. Igra je bila prikazana na CRT zaslonu veličine 35 x 16 pixela. Igru je igrao korisnik protiv kompjutera, odnosno stroja. Sam projekt Douglas je počeo jer je istraživao interakciju između čovjeka i računala. Igra se igrala pomoću telefona tako da koji se broj pritisnuo na to mjesto išao je znak X ili O.[11]

1958. godine nuklearni fizičar William Higinbotham napravio je simulaciju tenisa na osciloskopu te se može reći da je to prva 2D igra koja je napravljena. Igralo se tako da je svaki igrač imao na osciloskopu dva gumba preko kojih je s jednim mogao udariti loptu u bilo koje vrijeme po izboru, drugi gumb bio je za odabiranje kuta kojim se udarala lopta drugom igraču. Igra je napravljena pomoću releja, tranzistora i pojačala, što je u ono vrijeme tehničarima i Williamu trebalo nekoliko tjedana za projektiranje i izgradnju igre. Kasnih 70–ih N. Bushnell i T. Dabney osnovali su tvrtku Atari koja je odgovorna za stvaranje konzola i komercijalnu upotrebu za svaki dom. Prva od tih konzola sa više igara bila je Atari 2600 VCS (koja je imala 128 bajtova radne memorije).[12]

Steve Jobs, Steve Wozniak i Ronald Wayne osnovali su tvrtku Apple. 1977 Apple II postao je jedan od najpopularnijih računala ikad. Iako je golem napredak u odnosu na Apple I, sadržavao je isti procesor i radio na istoj brzini. Apple II je bio jedno od prvih računala sa zaslonom u boji, a također je i prvi razumljiv sustav. Najvažnija značajka Apple II je osam utora za proširenje. Druga računala nisu imala ovakve mogućnosti. Proširenja su omogućavala jednostavan pristup sustavu matične ploče i proširenje utora.[1]

Početkom 1989. godine napravljena je sci-fi igra po imenu Ultima Underworld. Napravljena je od istoimenog game engin-a (softvera za izradu igre), ali kad je izašla igra Space Roungue, Origin System je preuzeo Ultima Underworld te je razvila algoritam za mapiranje tekstura koje su se mogle primijeniti na podove, stropove, zidove i sl. Velika revolucija u svijetu zabave bila je kad su došla prva 16-bitna računala i tvrtka id Software stvorila je igru Wolferstein 3D 1992. To je prva igra u kojoj su se teksture na sve predmete oslikavale metodom billboarding. Tako se dobivao dojam da se igra 3D igra, iako je ona zapravo bila 2D. Također je bila samo jedna rotacija, odnosno vertikalna (lijevo/desno).[2]

Slika 2.1. *– Wolfenstein 3D [1]*

Godinu poslije id Software povlači ogromne granice i izlazi igra DOOM koja je izgledala drugačije i imala je veće mogućnosti. Igra je radila u višim rezolucijama, teksture se koristile na krovu i podu, sobe nisu više ravne, kamera se pomicala gore/dolje prilikom kretanja, rotacija oko dvije osi (lijevo/desno i gore/dolje). Igra uvodi mrežu Deathmatch i kooperativni mod. John Carmack i id Software postali su dio povijesti od tog trenutka jer su izazvali revoluciju i pomakli ljestvicu igara za stupanj više. Igra je imala nastavak pod nazivom DOOM2. Sljedeća revolucija igara je izdana 1995. godine napravljena od tvtke Parallax. Igra je omogućavala kretanje i rotaciju u svakom smjeru. XnGine (1995) bio je prvi 3D engine koji je razvijen u DOS bazi. To je omogućilo kasnije da se upotrebljavaju grafike visoke rezolucije i da budu kompatibilne sa 3dfx grafičkim karticama. Quake engine (1996) bio je prvi istinski 3D game engine. Imao je jedinstven kapacitet prerade za prikaz mape koji su vidljivi igraču i samo je to prikazivao i očitavao. Renderware (1996) je najpopularniji game engine za multiplatform igre. Podržava PlayStation 2, Wii, GameCube, Xbox, Xbox 360, PlayStation 3 i PSP platforme. OpenGL je višeplatformska specifikacija s podrškom za niz programskih jezika, čiji je cilj omogućiti pisanje aplikacija koje rade s 2D i 3D grafikom. Quake II / id Tech 2 engine podržava OpenGL (bolja svijetla, efekti, C podrška za jezik). Neke uspješne igre koji su na temelju OpenGL su Half-Life i Counter Strike. Najpopularnija je Unreal Engine (1998), koja integrira vlastiti skriptni jezik (UnrealScript) i map editor (UnrealEd). Modificirana verzija Quake II enginea je Quake III osmišljen 1999. koji podržava 32-bitnu boju, sjene i poboljšano umrežavanje.[2]

Croteam je isto tako zaslužan za igricu Serious Sam koja je napravljena Serious engineom (2001) i omogućavao je velike prostore i velik broj modela na zaslonu u bilo kojem trenutku. Ovaj engine je vrlo djelotvoran i podržava desetke animiranih modela čak i na skromnim konfiguracijama računala, te nadmašuje sposobnosti popularnih Quake, Unreal i Half-life engine-a. Podržava DirectX i OpenGL, a optimiziran je za DirectX 7. Današnji njihov engine je Serious Engine 4, te je prvi istinski multi-platform engine s podrškom za mobilne platforme i nove buduće generacijske igre.[3]

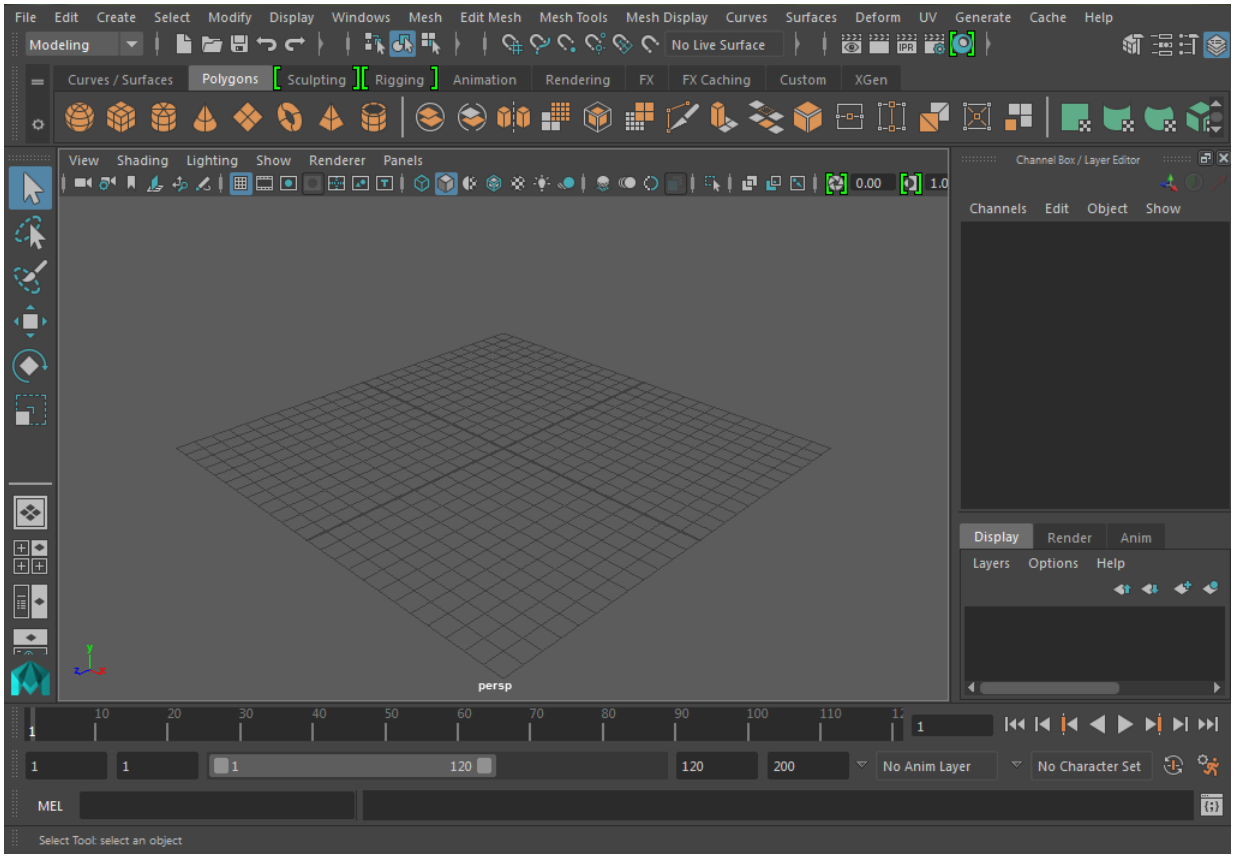
Slika 2.2. *– Serious Engine 4 [2]*

Svaki game engine sadrži grafiku, zvuk, logiku igre i druge elemente. Kad se rade alati za razvoj neke igre na game enginu oni trebaju biti kompitabilni sa pogonom igre da bi infomacije između programerskog i dizajnerskog djela bile bolje. Povezanost tih dijelova možemo podijeliti na svjetla, 3D modele, animacije, zvuk i sl. Također, svaki od tih alata treba biti i kompatibilan s drugim hardverima i softverima. U igrama kvaliteta grafike nije ista kao u filmskoj. Radi se na tome da se smanji hardversko opterećenje, a da se pritom pruži visoka kvaliteta igre.[2]

3. Autodesk Maya

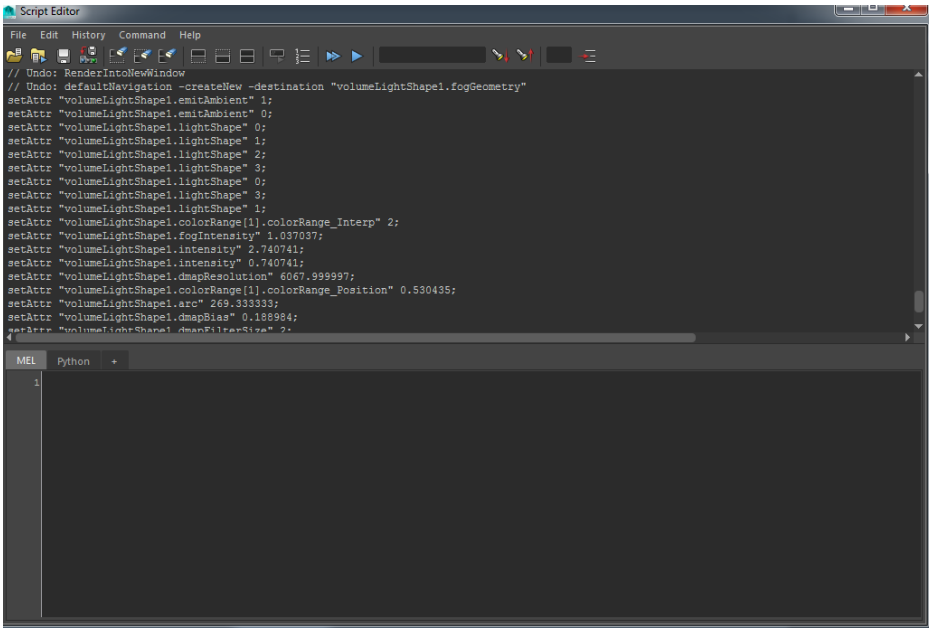
Autodesk Maya je softver koji omogućuje trodimenzionalno modeliranje, animacije, simulacije, renderiranje, teksturiranje. Izvorno je nastao od Alias Systems, a danas je u vlasništvu Autodeska. Pomoću Maye mogu se raditi trodimenzionalne aplikacije, videoigre, animirani film, TV serije, vizualne efekte. Maya se može pohvaliti za mnoge izrade i stvaranje grafike u filmovima kao što su: Matrix, Avatar, Hugo i sl. Vizualni efekti su upotrijebljeni u uvodnoj špici trenutno najbolje ocijenjenoj seriji Game of Thrones.[21]

Maya je do kraja 2009. godine imala dvije verzije: Maya Complete i Maya Unlimited. Razlika je bila u tome što je Maya Complete imala osnovne alate, a Maya Unlimited je imala dodatne dijelove. Problem je bio što je Maya Unlimited bila mnogo skuplja te su 2010. godine ukinute dvije verzije i sve je smješteno u jednu Autodesk Maya 2010 [4]. Najnovija Autodesk Maya 2016. nudi nam: bolje tekuće simulacije (pjena, mjehurići, valovi), atmosferski utjecaji poput dima ili magle, novi alati za oblikovanje modela sa više detalja i rezolucije, poboljšani pregled tijeka prilikom pripreme za renderiranje.[13]



Slika 3.1. *Sučelje Autodesk Maya 2016.*

Maya koristi skriptni jezik Maya Emmbeded Language (MEL) koji je jednostavniji za korištenje i omogućuje mijenjanje izvornih funkcija. Maya Emmbeded Language isto nudi ubrzano i pojednostavljeno korištenje zadataka koji se ponavljaju. Uz MEL podržava i Python, programski jezik koji se koristi za aplikacije, web dizajn i sl.



Slika 3.2. *Script editor - MEL i Python*

3.1. Modeliranje

Modeliranjem u Mayi stvaraju se trodimenzionalni objekti koji se žele prikazati. Modeliranje je temeljna faza u izradi nekog objekta ili tijela, a na njih se primjenjuju dalje animacije, teksture, simulacije itd.

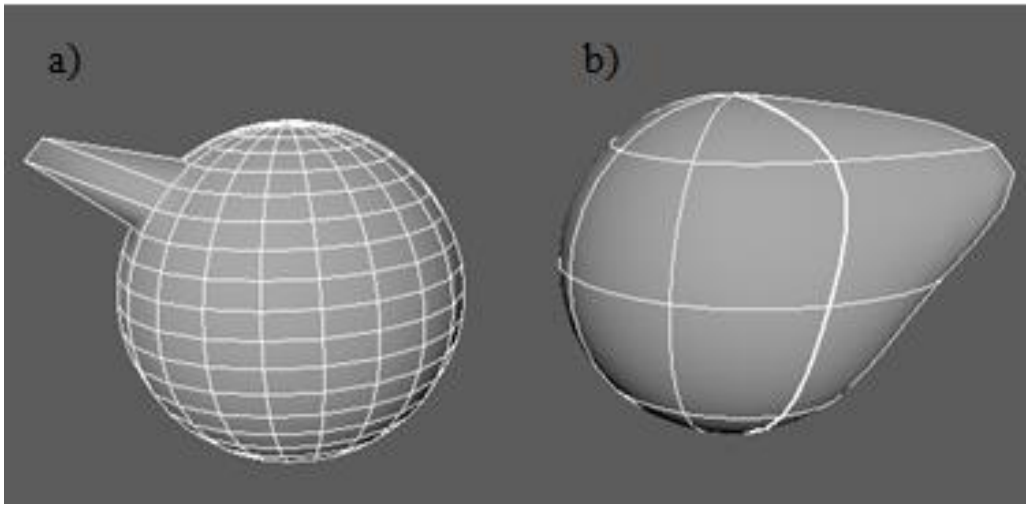
Modelirati možemo na 3 načina [5]:

1) Poligon (eng. Polygon) – čine osnovni poligoni poput kocke, valjka, kugle, piramide, stošca. Kod ovog tipa modeliranja bitna je točka na rubu (eng. vertex). Najjednostavniji poligon je trokut koji čine tri vrha međusobno povezani sa tri ruba. Više trokuta ili drugih oblika čine elemente tj. složenije poligone, odnosno objekte s više od tri vrha.

2) NURBS (eng. NURBS – Non Uniformal Rational Bezier Splines) – modeli su prikazani pomoću Bézierovih krivulja i površina. Krivulje se sastoje od početka,

zakrivljenosti i kraja krivulje. Dodavanjem više vrhova unutar krivulje daje više mogućnosti i točaka za upravljanje,a isto tako nema utjecaja na zaobljenost.

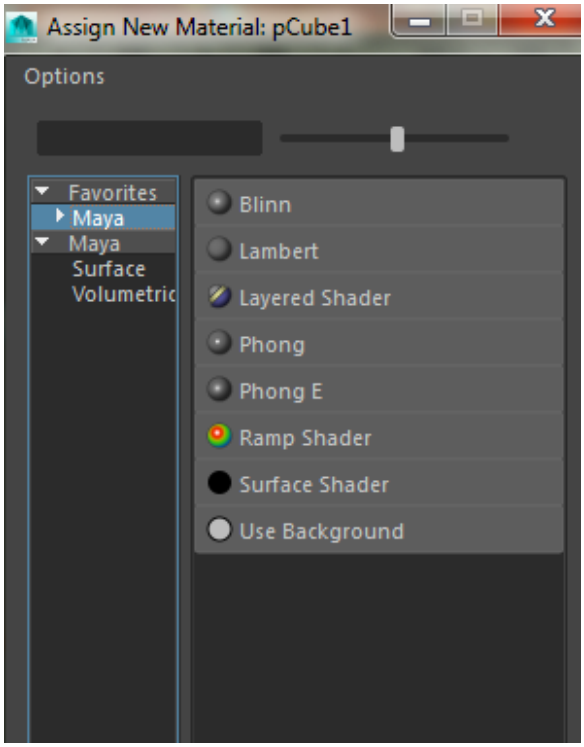
3) Subdivizijsko (eng. Subdivision surfaces) - modeli se dobivaju kombinacijom poligona i krivulja. Baziraju se na poligonalnom modeliranju, a rezultat je glatka površina. Rubovi mogu biti oštri i zaobljeni.



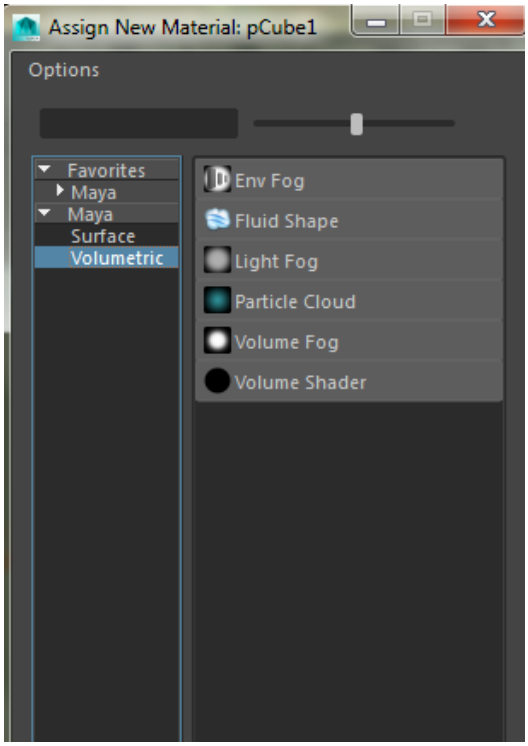
Slika 3.3. *a) Poligonalno modeliranje i b) NURBS modeliranje*

3.2. Teksturiranje

Svaki model koji se napravi može se teksturirati. Teksturiranjem modeli dobivaju boju i materijal od čega su napravljeni, tj. model postaje stvar, živo biće, prirodna pojava i sl. Svaki materijal ima svoju vrstu teksture te se mora dati željena vrsta materijala, tako na primjer čaša mora biti sjajna i prozirna, metal mora biti reflektirajući i sjajan. Najčešće vrste površina koje se koriste su glatke, hrapave, sjajne, zrnaste. Tako postoje osnovne vrste materijala poput – Lambert, Phong, Blinn, Phong E, Anisotropic Material, Shader Ramp, Surface Shader, Background. Također postoje 2D texture, a neke od njih su platno, voda, bitmape, ramp (prozorske refleksije u oku, odsjaj željeza) i dr. Volumetrijski materijali (eng. Volumetric) prikazuju prirodne materijale i pojave kao što su: magla, dim i prašina, te se oni upotrebljavaju kao dodatne teksture. Za dobro teksturiranje poželjno je koristiti dodatne programe za obradu fotografija, kako bi model dobio karakteristike koje su potrebne u skladu s namjenom modela. Veliku uslugu u postavljanju i prilagođavanju teksture na model imaju i UV točke bez kojih bi bilo teško zamisliti proces teksturiranja.[6]



Slika 3.4. *Osnovni materijali za teksturiranje*

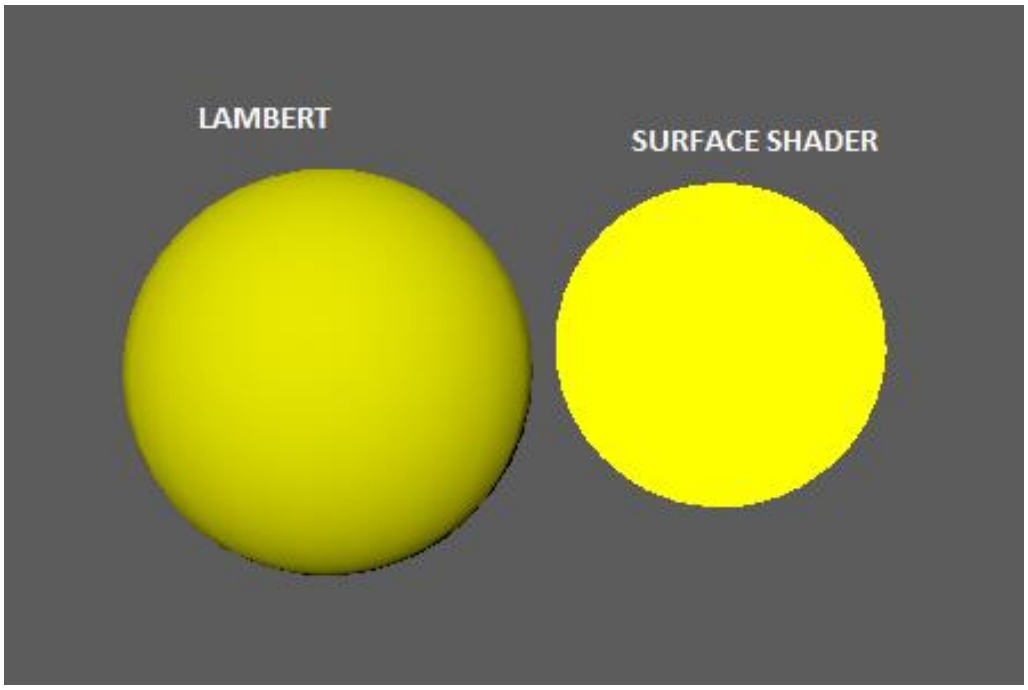


Slika 3.5. *Volumetrijski materijali za teksturiranje*

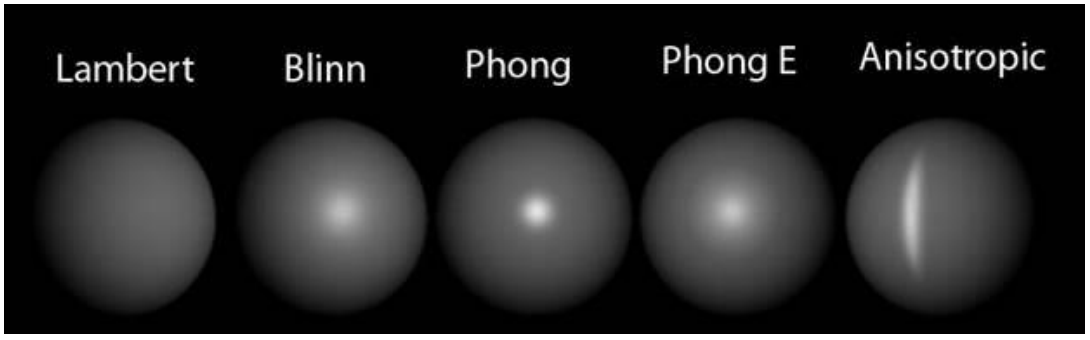
Tablica 3.6. *Opis tipova materijala za teksturiranje*

Lambert	Ravan materijal koji ima glatku podlogu bez ikakvog naglašavanja. Osnovni materijal kod teksturiranja, nema refleksiju, sjaj, mat ili ostale efekte. Lambert se upotrebljava za površine kao što su keramika, kreda, mat boje itd.
Blinn	Materijal koji najbolje prikazuje metalne površine (aluminij, čelik). To je također i računski najskuplji od 3 uobičajena materijala: Lambert, Phong i Blinn. Kod Blinna se može kontrolirati veličina sjaja i površina odsjaja na materijalu.**[7]**
Phong i Phong E	Vrsta materijala koja služi za staklene ili svjetleće površine (kupaonica, automobil, telefon, predmeti od stakla). Na Phongu se može podešavati zrcalnost i refleksija boje. PhongE je sličan kao običan Phong koji ima brže renderiranje i mekše prijelaze nego običan Phong.
Anisotropic	Prikazuje boje koje imaju postepene prijelaze (gradiente) koji se koriste na sjajnim predmetima kao što su sjajne posude, CD

	za računala, dno tave za pečenje. Polukružni prijelazi mogu se orijentirati i rasporediti na različite načine.
Ramp	Materijal koji se koristi za prijelaze između više boja, način promjene boje sa svjetlom, kut gledanja između boja, mogućnost dodavanja broja prijelaza boja. Teksture se također mogu dodati na materijal i mogu se miješati s bojama.
Surface Shader	Materijal koji ne sadrži sjenčanja, te ne uzima u obzir sjene i svjetla. Najbolje je upotrebljiv za pozadine kao što je nebo, jako osvijetljene znakove i za crtane materijale.



Slika 3.7. *Razlika između lambert i surface shader materijala*

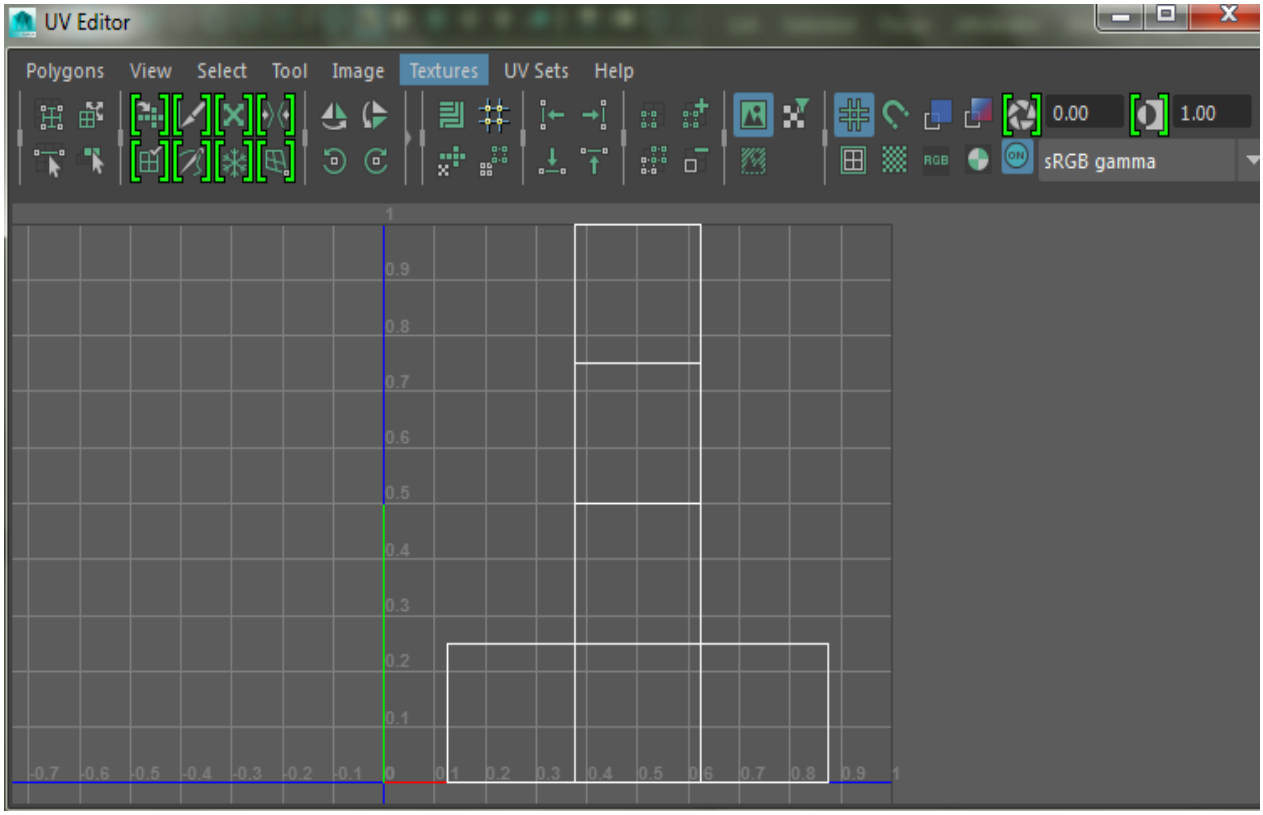


Slika 3.8. *Tipovi materijala za teksturiranje [3]*

3.3. UV mapiranje

Kad se postavi slika na model proces teksturiranja nije gotov jer još je potrebno prilagoditi teksturu da dobro stoji na modelu. Taj se proces naziva mapiranje koje se izvodi pomoću UV editora, Attribute Editor-a i Hypershade-a.[6]

UV točke su prikazane u 2D teksturnom koordinatnom prostoru i označavaju osi sa slovima U i V. Kada se odabere neki 3D model UV točke razmještaju poziciju i prilagođavaju je na 2D teksturnu mapu prema poziciji točaka na modelu. Tako se preko tog prostora točke mogu pomicati i prilagođavati željenom rezultatu. Mapiranje UV točaka je prostor koji služi za uređivanje UV točaka koje se prikazuju u dvodimenzionalnoj mreži, te služi kao tekstura i pojavljuje se u UV Editoru. UV editor sadrži mnogo opcija za manipuliranje UV točkama. Na svaki tip modela može biti primijenjen različiti tip UV mapiranja, a radni prostor mreže počinje na 0 i završava sa 1 .[6]



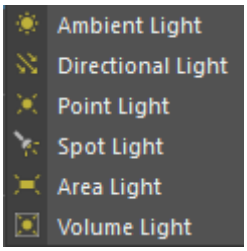
Slika 3.9. *Prikaz UV Editora u Mayi 2016.*

Postoji više tehnika UV mapiranja, a koriste se ovisno o kakvoj se vrsti i obliku modela radi. Međutim UV mapiranje nije uvijek točno, pa je stoga potrebna daljnja obrada UV točaka korištenjem UV Texture Editor-a.[6]

Tablica 3.10. *Tehnike UV mapiranja*

Automatic UV mapping	Izrađuje UV mrežu tako da pronalazi najbolji UV položaj projiciran sa tri strane modela. Automatsko mapiranje koristi se kod pravilnih i nepravilnih oblika modela.
Planar UV mapping	Izrađuje UV na mrežu kroz ravnu plohu. Koristi se za objekte koji su relativno ravni ili su kompletno vidljivi samo iz jednog kuta. Moguće je označiti samo nekoliko elemenata i izraditi projekciju na samo određene elemente modela.
Cylindrical UV mapping	Izrađuje UV točke za objekte bazirane za kružne (cilindrične) oblike i mreža se omotava oko objekta. Najbolja je za objekte koji su kompletno zatvoreni.
Spherical UV mapping	Izrađuje UV mrežu za korištenje projekcije koja je bazirana na kuglastom omotu oko modela. Ova projekcija je najbolja za oblike koji mogu biti cijeli zatvoreni i vidljivi u sklopu okruglog objekta, bez projiciranja praznih ili nepravilnih dijelova.
Camera based UV mapping	Izrađuje UV teksturne koordinate prema trenutnom pogledu kamere. Pogled kamere postaje ploha projekcije na objekt, a funkcionira na principu planar mapiranja.
Contour Stretch UV mapping	Nova vrsta mapiranja koja je izašla u novoj mayi 2016. Model se teksturira onako kako se označi ili odabere njegov dio. Kontura analizira četiri kuta kako bi se utvrdilo kako najbolje rastegnuti UV poligone.

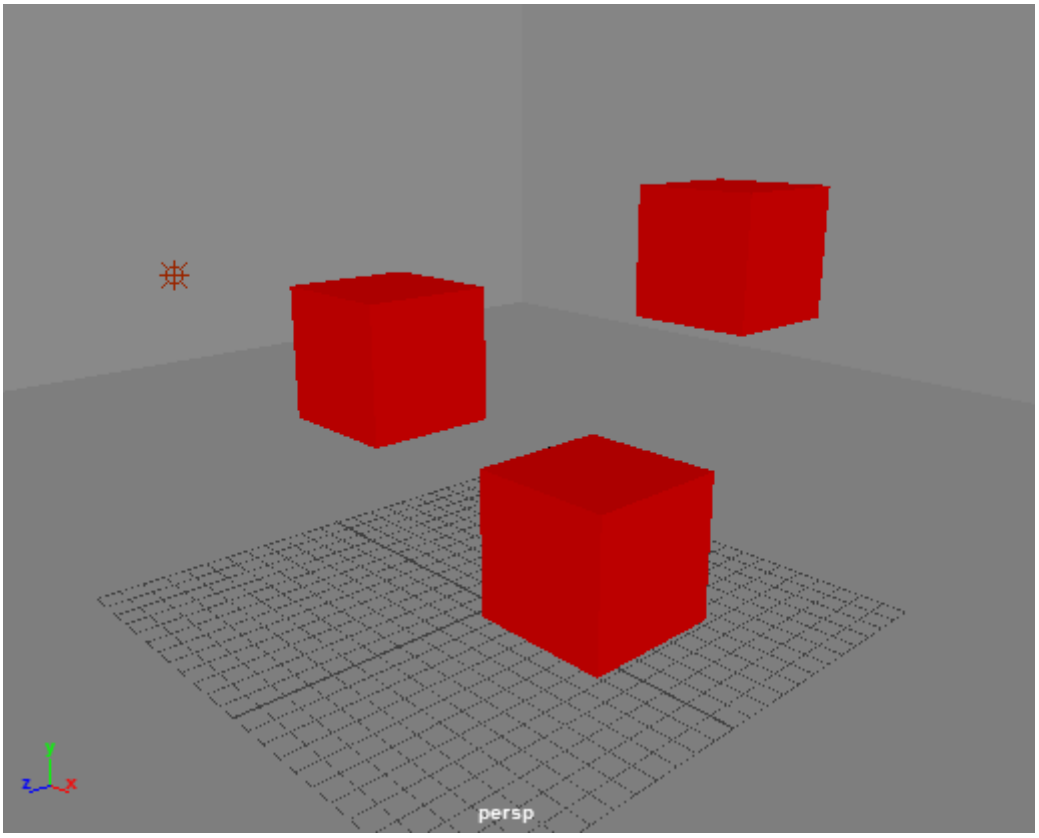
3.4. Svjetla

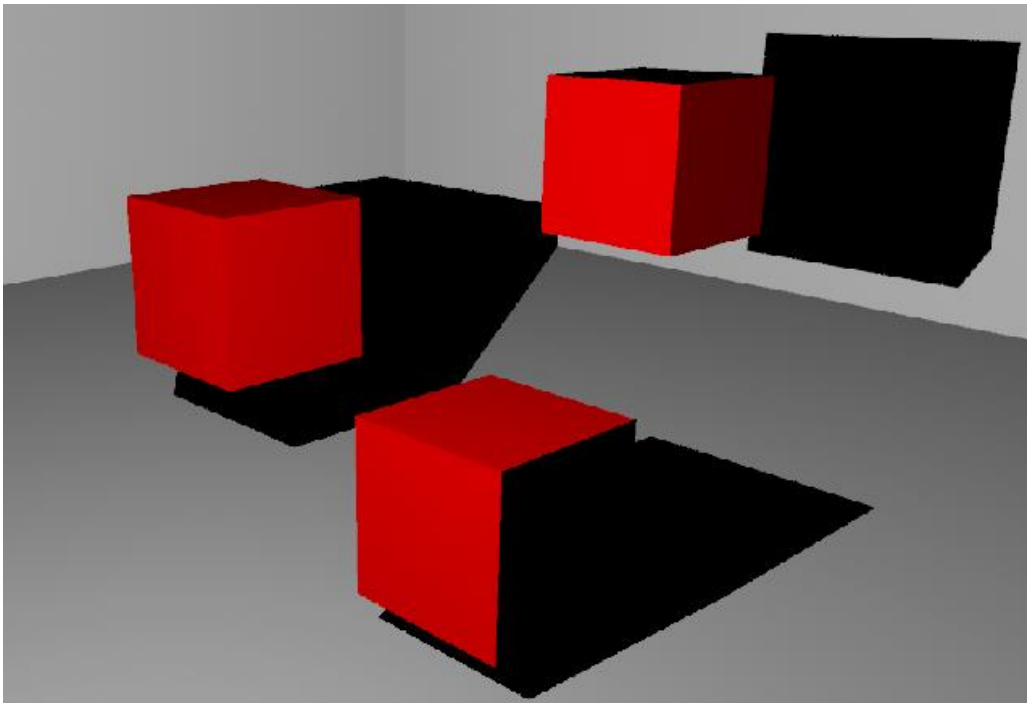


Slika 3.11. *Vrste svjetla u Mayi*

Ambient Light – široko je rasprostranjeno u svakodnevnom životu. Ambient light je neizravno svjetlo koje je odbijeno ili se prenosi putem drugih objekata. Osvjetljava sa svih strana područja na sceni, čak i ona koja nisu izravno osvijetljena drugim izvorom svjetlosti. U stvarnom životu prikuplja svjetlost iz okoliša i dodaje te boje na različite strane objekata, na temelju prikupljenih boja iz okoliša.[17]

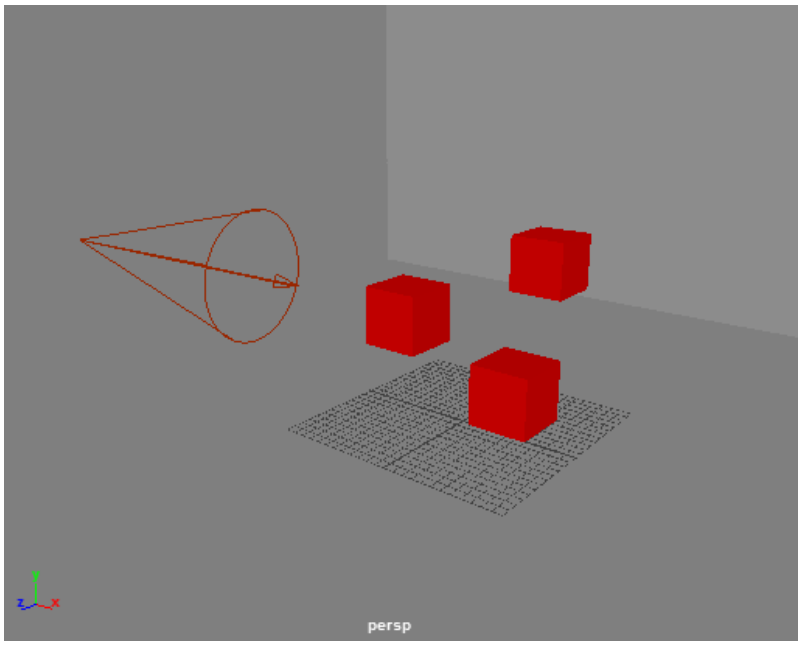
Point Light – simulira sjajne zrake iz jedne beskrajno male točke u prostoru. Emitira svjetlo ravnomjerno u svim smjerovima, poput gole žarulje ili užarene zvijezde u svemiru. Osvjetljenje i sjene prikazuju se ovisno o strani na kojem se svjetlo i objekt nalaze.[17]

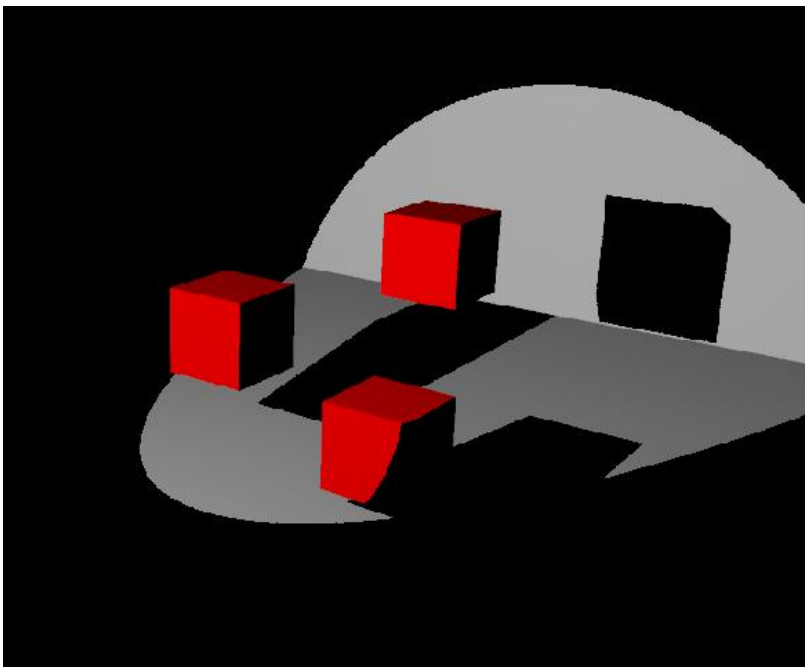




Slika 3.12. i 3.13. *Point Light (prije i poslije renderiranja)*

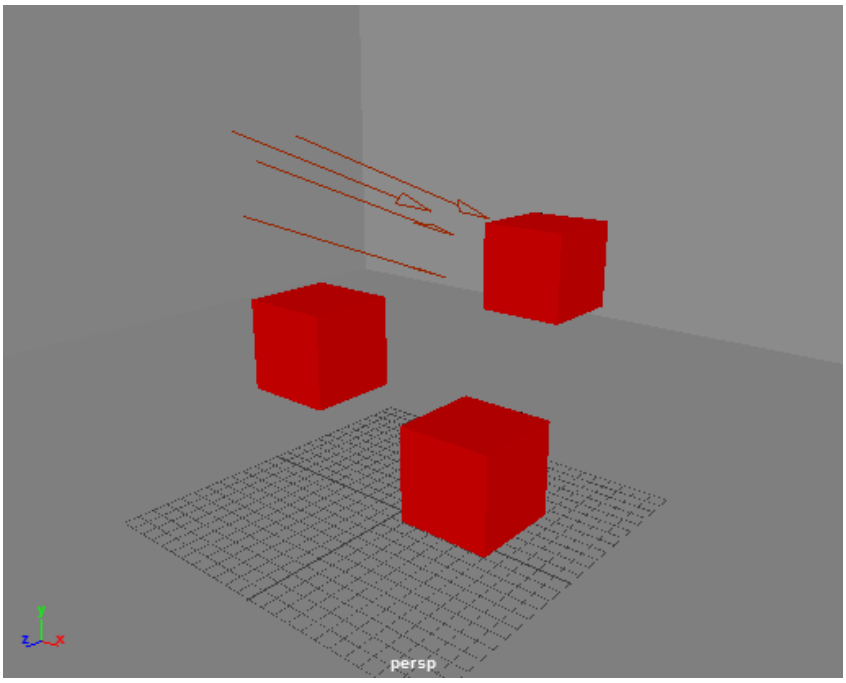
Spot Light – koristi se najviše u dizajnu računalne grafike. Spot light simulira svjetlost iz jedne točke, slično kao point light. Razlika je u tome što spot light ima ograničeno osvjetljenje u obliku stošca, tj. izgleda kao snop svjetla. Rotacijom se određuje gdje će se usmjeriti svjetlo i što se želi osvijetliti. Također pri odabiru ovog svjetla korisnik ima ekstra mogućnosti, kontrole i opcije za podešavanje (primjer: snop svjetla vidljive kroz maglu). Također se može kontrolirati širina kuta osvjetljavanja, te se može imati uski ili široki snop.[17]

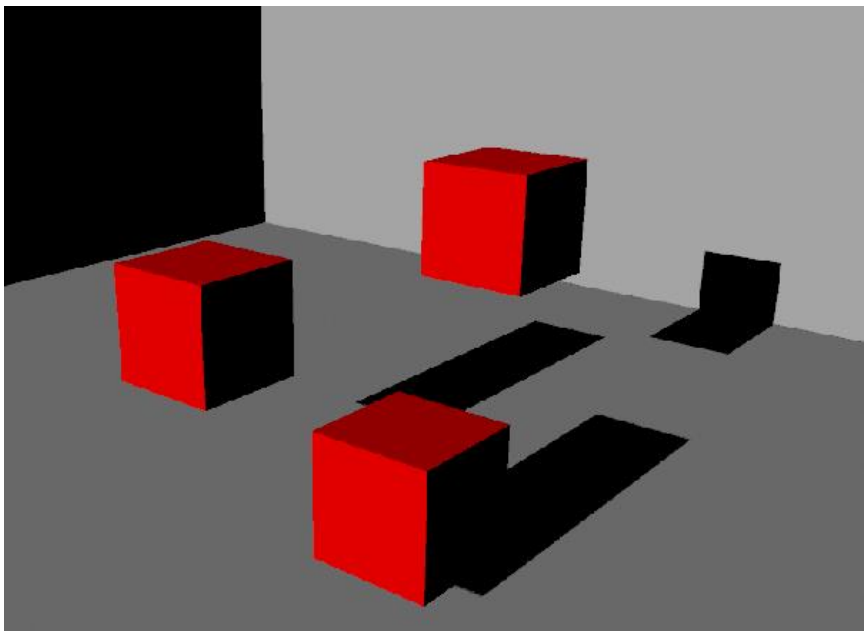




Slika 3.14. i 3.15. *Spot Light (prije i poslije renderiranja)*

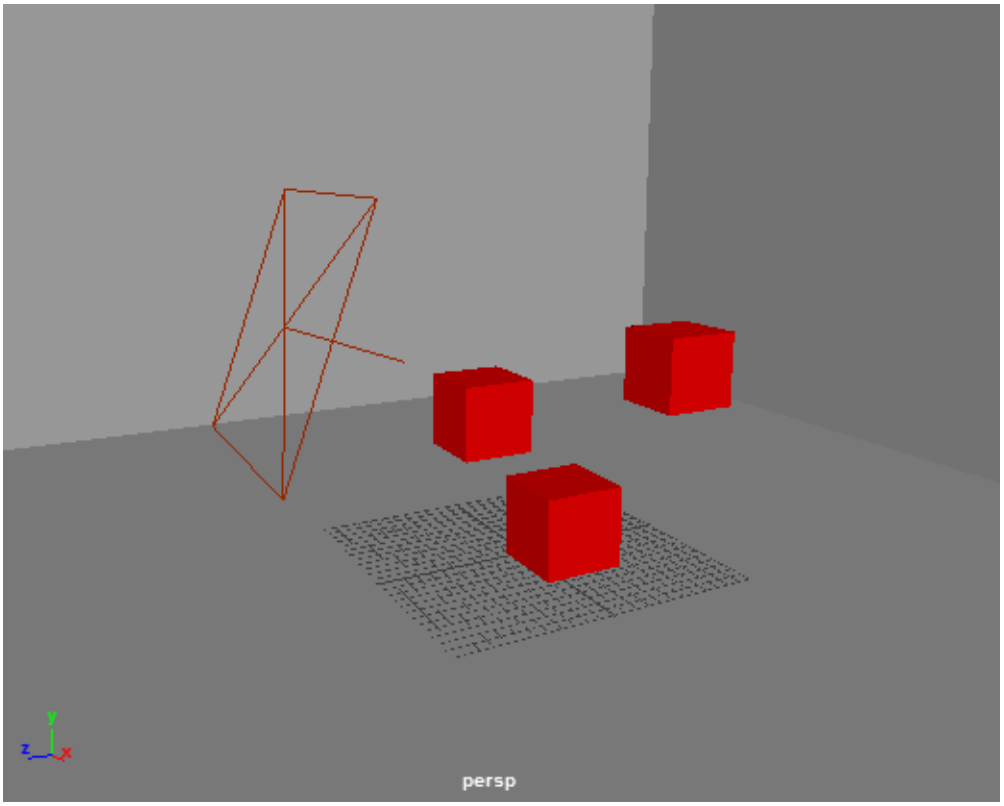
Directional Light – kod ovog svjetla je bitan smjer vektora, odnosno kako se svjetlo rotira tako će padati na svaki objekt bez obzira gdje se on nalazio. Sve se sjene bacaju iz smjera koju svjetlost baca. Nije bitno gdje se svjetlo nalazi u prostoru (iza,ispred, ispod) objekta, nego je bitan kut smjera svjetla. Ovo svjetlo se može usporediti sa sunčevim svjetlom (dnevnim svjetlom), jer ispunjava velike površine. Problem kod ovog svjetla je što se ne može dobro ograničiti na neku određenu površinu, kao što je primjer kod point i spot light, te se tako ovo svjetlo više upotrebljava kao sekundarno svjetlo.[17]

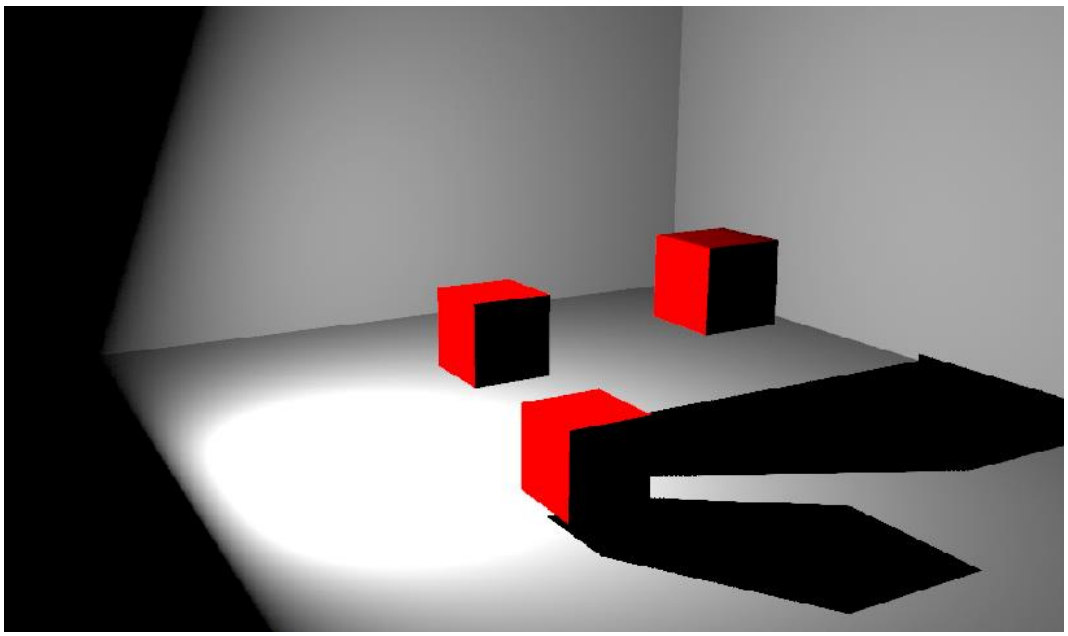




Slika 3.16. i 3.17. *Directional Light (prije i poslije renderiranja)*

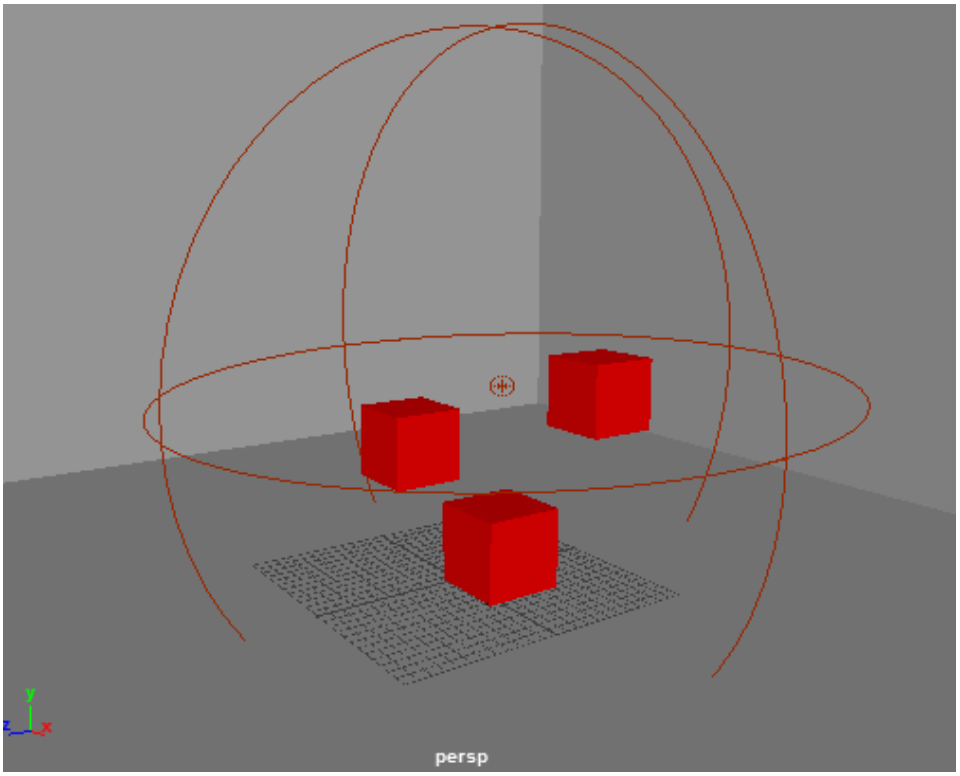
Area Light – dvodimenzionalni pravokutni izvor svjetlosti. Koriste se za simulaciju pravokutnih refleksija na prozorskim površinama. Kod usporedbe sa drugim svjetlima renderiranje je duže, ali kvaliteta svjetla i sjena je mnogo veća. Upotrebljavaju se kod visoko kvalitetnih fotografija, ali nisu povoljni za duže animacije gdje renderiranje može potrajati.[17]

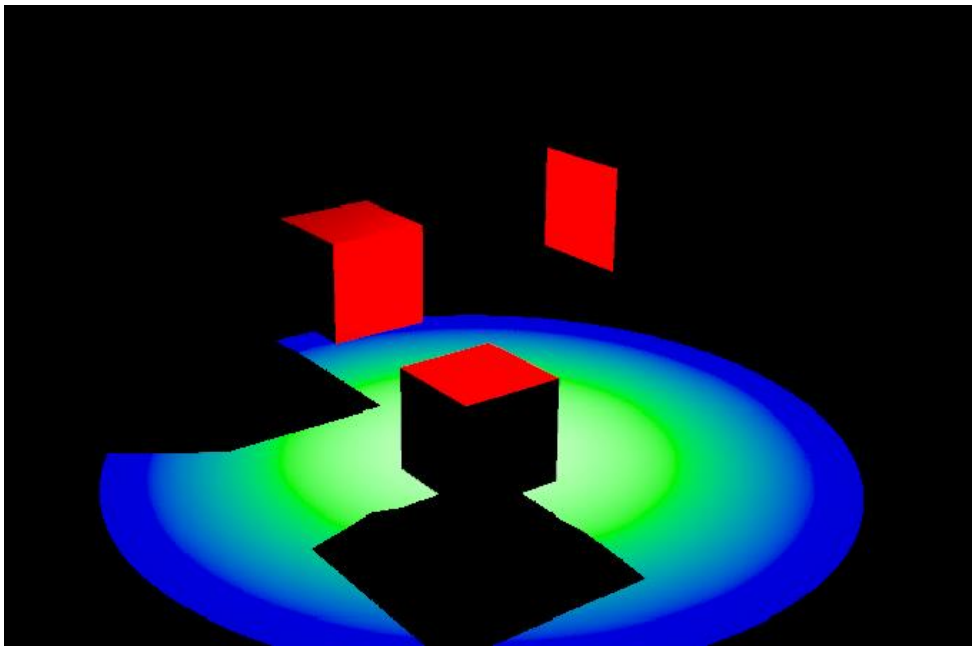




Slika 3.18. i 3.19. *Areal Light (prije i poslije renderiranja)*

Volume Light – glavna prednost kod ovog svjetla je prikaz prostora koji obuhvaća. Svjetlo se može prikazati u više boja (gradijent), koji sprječavaju raspadanja i dodatnu kontrolu. Gradient je također koristan za volumen magle, a mogu se postići i različiti efekti sa usmjerivanjem svjetla. Volume light koristi se za osvjetljavanje unutar određenog prostora, pruža kontrolu smjera svjetla i boje unutar određenog volumena.[17]





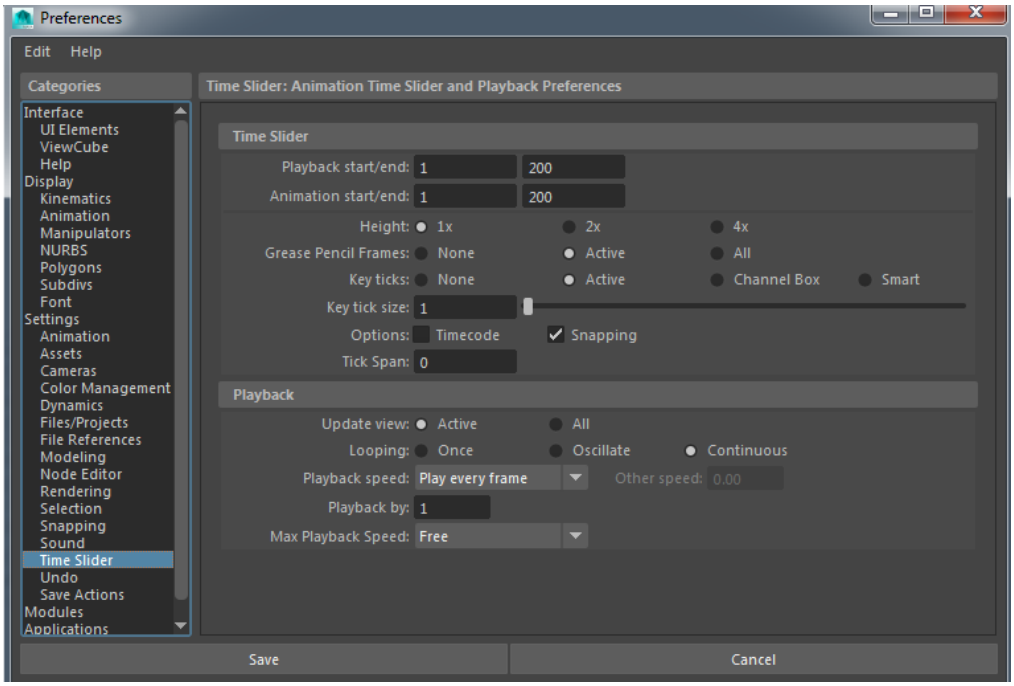
Slika 3.20. i 3.21. *Volume Light (prije i poslije renderiranja)*

3.5. Animacija

U Mayi se može animirati gotovo sve što se može zamisliti, bez obzira koliko je to nestvarno. Da bi se animiralo treba razmisliti unaprijed o pokretima i vremenu kako će to izgledati. Animacije u Mayi kontroliraju se i postavljaju preko trake za animaciju. Na toj traci nalazi se vremensko podešavanje, veličina koju želimo podešavati, animacijske postavke, te gumbovi za pokretanje animacije.[16]

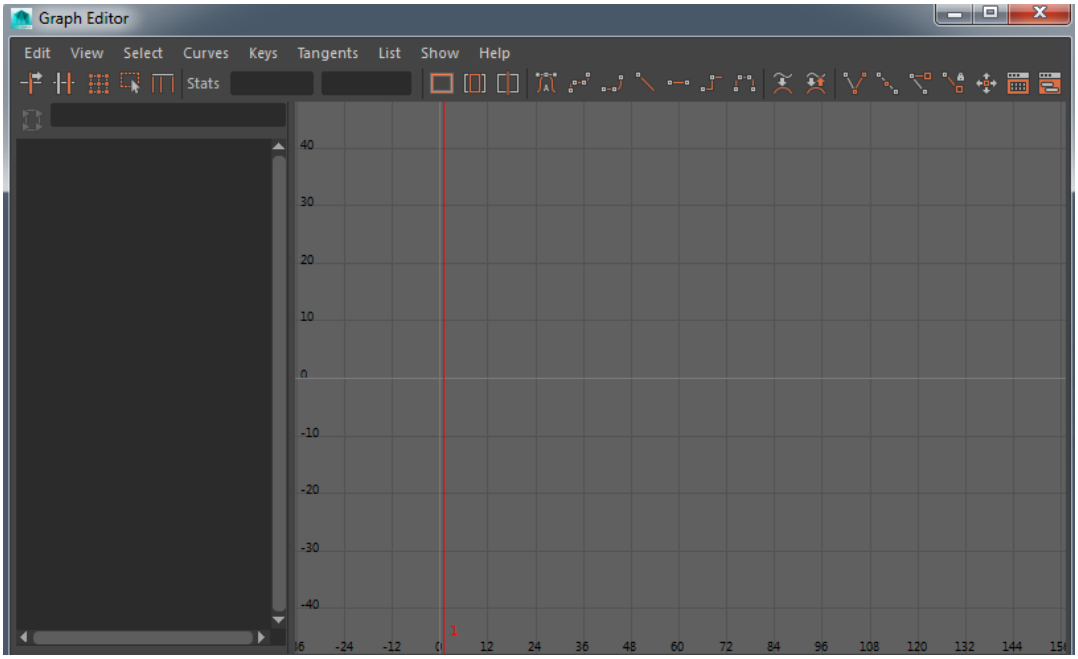
Slika 3.22. *Traka za animaciju*

Traku za animaciju čini glavni okvir (eng. keyframe) koji dodjeljuje vrijednosti na željeni objekt ili model. Većina animacijskih sustava koriste fraemove kao osnovne jedinice mjerenja, jer zbog tromosti oka čovjeku je to prebrzo da vidi svaki dio pokreta koji se stavi za određenu animaciju. Npr. dodaje se od prvog do desetog okvira (eng. Frame-a) na atribut ruke njezina rotacija. Čovjeku je dovoljno 24 sličice po sekundi da on vidi normalan pokret u filmu ili igri zbog već navedene tromosti oka. Također postoji dijaloški okvir, odnosno dodatne animacijske postavke na kojoj se mogu mijenjati vrijednosti vremena i reprodukcije za animaciju,a može se postaviti i ukupno vrijeme animacije.[16]



Slika 3.23. *Dodatne opcije za animaciju*

Graph Editor je alat za bolje poboljšanje vrijednosti za svaki frame. Vizualni prikaz nudi zaobljene crte od atributa koje su animirane za svaki frame. Vrijeme animacije ide s lijeva na desno i prikazuje se kao varijabla koja ide gore ili dolje.[16]

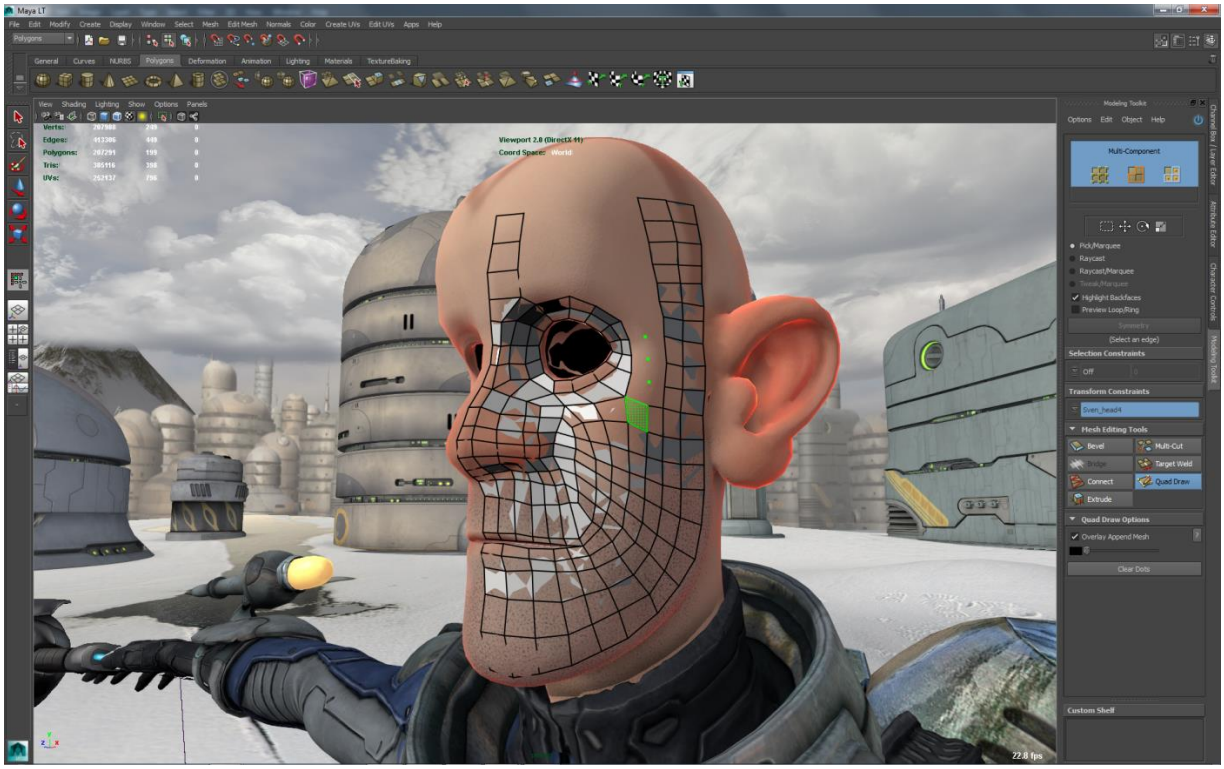


Slika 3.24. *Graph Editor*

3.6. Prednosti Autodesk Maye, Maya LT i konkurencija

Maya je nedvojbeno jedan od najpopularnijih i najkorištenijih programa u 3D svijetu. Program je dobro napravljen za 3D animatore i najčešće se koristi u izradi filma i animacija. U odnosu na druge programe Maya je više program za animiranje, odnosno više se koristi u filmskoj industriji nego za videoigre. U zadnjem djelu Harry Pottera Maya je pomogla u potpuno računalnoj izradi zgrade Hogwarts sa ogromnim 3D okruženjem. Velika prednost programa je široki raspon alata, jednostavnost i pristupačnost što povećava produktivnost i brže modeliranje. Rigging sustav omogućuje brzo i jednostavno stvaranje karaktera i animacije. Međutim Maya se smatra programom težim za učenje, te ako se taj program svlada da je lako naučiti i prijeći na ostale programe. Maya je kompatibilna s različitim dodacima koji pomažu u stvaranju realističnih okruženja i naprednom osvjetljenju. Kada je u pitanju jednostavnost i dostupnost alata za animaciju, ovaj program je u velikoj prednosti u odnosu na druge. Nedostaci kod Maye: spor rad, audio problemi prilikom editiranja, duže vrijeme renderiranja, duže vrijeme učenja u odnosu na ostale programe.[14]

Postoji i druga verzija Maye Maya LT. Osmišljen je kako bi pomogao i olakšao rad sa kreativnim alatima za razvoj indie igara. Glavni alati koji se koriste su za stvaranje 3D elemenata za igru, te su poboljšani alati za bolje modeliranje. Maya LT također sadrži jednostavan izvoz 3D modela i animacija izravno za Unity. Najveća razlika između Maye LT i standardne je da su značajke za renderiranje uklonjene (specijalni efekti, čestice, dinamični sustavi, tkanine, kose, tekućine, dinamike polja, dinamični učinci vatre su također isključeni). Maya LT uključuje sve značajke igre koju programeri trebaju koristiti. Modeliranja uključuju dodatne značajke za modeliranje s poligonima, subdivision i NURBS. Quad Draw alat omogućuje stvaranje i povezivanje novih četverokuta na postojećem modelu (slika 3.25.). Kod teksturiranja uključuje hypershade, 3D alat za bojanje koji se direktno slika na mrežu. Što se tiče formata podržava FBX format koji koriste većina game enginea uključujući i Unity.[15]



Slika 3.25. *Quad Draw u Mayi LT [4]*

Postoje mnogi programi za 3D kompjutersku grafiku. Svaki od njih ima svoje prednosti i mane. Neki su namijenjeni više za modeliranje dok su drugi bolji u animaciji. Tako se neki programi više upotrebljavaju u filmskoj industriji, a drugi u izradi videoigara.

Tablica 3.26. *Usporedba programa za 3D kompjutersku grafiku*

PROGRAM	**PLATFORME**	**GLAVNA UPOTREBA**	**ULAZNI I IZLAZNI FORMATI (najbitniji)**	**CIJENA**
Maya	Windows, Mac OS X, Linux	modeliranje, animacija, svjetla, renderiranje, vizualni 3D effekti	GIF, PNG, BMP, FBX, DXF, OBJ, MEL, JPG	3675 $
3ds Max	Widnows	modeliranje, videoigre, renderiranje, svjetla	FBX, 3DS, XML, DXF, DWG	3675 $
Blender	Windows, Mac OS X, Linux	modeliranje, animacija, renderiranje, svjetla, videoigre, 3D efekti	JPG, PNG, GIF, AVI, MOV, FBX, DXF, OBJ	besplatno
Cinema 4D	Windows, Mac OS X	animacija, svjetla, modeliranje, vizualni 3D efekti, renderiranje, simulacija	FBX, OBJ, 3DS	3695 $,
Modo	Windows, Mac OS X, Linux	modeliranje, animacija, renderiranje	DXF, FBX, 3DS, BMP, GIF, JPG, JPEG, PNG	1495 $

Kao glavni konkurent Mayi su Blender i 3ds Max. 3ds Max koristi više krivulja i zahtjeva malo vremena da bi se naučilo modelirati na osnovnoj razini. Jedna od bitnijih prednosti je da može obrađivati veći broj podataka, poboljšava prenformanse i vizualne kvalitete. Sučelje je relativno jednostavno za shvatiti i naučiti, a UI omogućuje brz i jednostavan pristup najvažnijim alatima. Mana u odnosu na Mayu je što se može koristiti samo u sustavu Windows i zahtjeva 64-bit procesor, te ne podržava Linux ili Mac OS X.[18]. 3ds Max je vodeći izbor za izradu 3D igre, a također se koristi u tehničke svrhe. Razlog je zbog

interakcije sa programom AutoCAD i ostalih programa za dizajn. Interakcija sa objektima, njihovo imenovanje, premještanje, rotiranje je bit programa 3ds Max.[19]

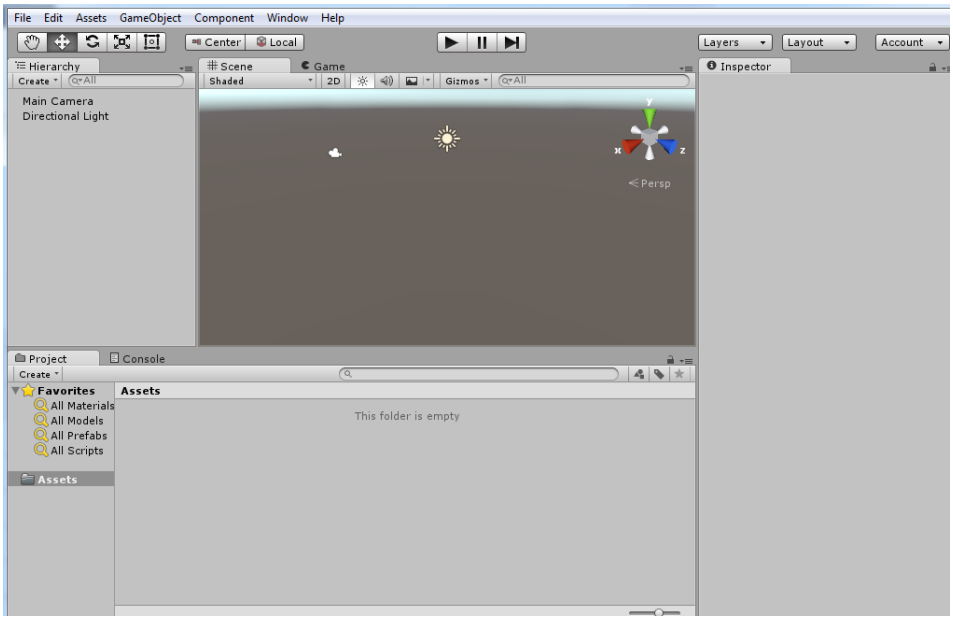
Drugi konkurent Mayi je Blender koji je besplatan. Sve više je popularan među developerima, te konstantno ima nadogradnje i poboljšanja. Jedan od najboljih elemenata Blendera su alati za modeliranje, označavanje objekta, rubova i sl. Postoji poseban Edit mode koji služi za uređivanje posebno za svaki objekt, te se time ne selektiraju modeli koje se ne žele uređivati. Selekcija je obrnuta od Maye (npr. vertexi se označavaju desnom tipkom miša). Blender je dobro prilagođen za pojedince i male skupine ljudi koji žele napraviti svoj projekt. Program radi jednako dobro na Linux, Windows i Mac OS X. Njegovo sučelje koristi OpenGL.[20]

Uz Blender i 3ds Max može se dodati i Cinema 4D. Odličan program za individualne svrhe ili manje timove. Cinema 4D nudi brzo i kvalitetno renderiranje. MoGraph je dodatak koji nudi brz i jednostavan tijek rada. Omogućuje kloniranje predmeta, dodavanje efekata, gibanja, ekstrudirani tekst. Skica i Toon su alati za sjenčanje, crtiće i tehničke crteže. Cinema 4D podržava Windows i Apple OS X i zahtjeva 64-bit procesor.[14]

4. Unity

Unity game engine je program za razvoj igara za PC, mobilne uređaje i racunalnu grafiku koji je razvijen od strane Unity Tachnologies. Prvo je najavljen samo za platformu OS X, no 2005. godine se razvio na ostalo računalno i mobilno tržište (iOS, Android, Windows, Playstation 3 i 4, Xbox One, Mac itd.). Razlog osnivanja tvrtke je da se stvori engine koji će biti relativno dostupan svima u izradi igara, te je u kratkom vremenu privukao velike tvrtke. Program je privukao i programere, jer Unity nudi kvalitetne skriptne jezike. Razvojem iPhonea Unity je bio jedini program koji je u cijelosti podržavao sve funkcije i mogućnosti operacijskog sustava. Broj igara je porastao na više od 50% koje su napravljene preko Unity za Android i iOS uređaje. Što se tiče ulaznih i izlaznih formata Unity podržava većinu programa (Maya, 3D studio MAX, Blender, Adobe PS i Fireworks) za 3D modeliranje, uređivanje slika, videomontažu itd. Grafički engine nudi velike mogućnosti jer radi na više platformi kao što su Direct3D i OpenGL.[8]

Današnja verzija koja je dostupna korisnicima je Unity 5. Dostupna je u Personal verziji koja je besplatna i Proffesional verziji koja se plaća. Unity 5 je dosad najbolja platforma za stvaranje 2D ili 3D igre, te donosi mnogo novih alata i poboljšanja u izgledu. Neki od noviteta u Unity 5 su: sustav sjena koji čine materijale da izgledaju dosljedno u bilo kojem okruženju, Audio mixer je znatno unaprijedio zvuk u igri, poboljšan je vizualni prikaz, ogromna poboljšanja 3D fizike.[23]



Slika 4.1. *Sučelje Unity 5 Personal Edition*

4.1. Osnovni dijelovi Unitya

Projekt (eng. Project view) služi za upravljanje elementima (Assets) koji pripadaju i koji se stvaraju u projektu. Lijevi dio (traka) prikazuje strukturu foldera i hijerarhijsku listu onoga što ima u projektu. Svaki element je prikazan kao ikonica sa slikom koja prikazuje o kojoj se vrsti ikone radi (model, skripta, tekstura itd.). U gornjem lijevom kutu je naredba Create koja omogućuje dodavanje novih elemenata.[36]

Slika 4.2. *Project view [5]*

Inspektor (eng. Inspector view) prikazuje detaljno informacije za odabrani objekt. Ovo je mjesto gdje programeri mogu mijenjati vrijednosti kako bi dobili pravi osjećaj za igru. Bilo koja osobina koja je prikazana i koja se doda u Inspector može se promijeniti unutar ovog pregleda (fizika, zvuk, modeli). Također sve skripte koje se žele primjeniti na željeni objekt se mogu dodati ili mijenjati.[36]

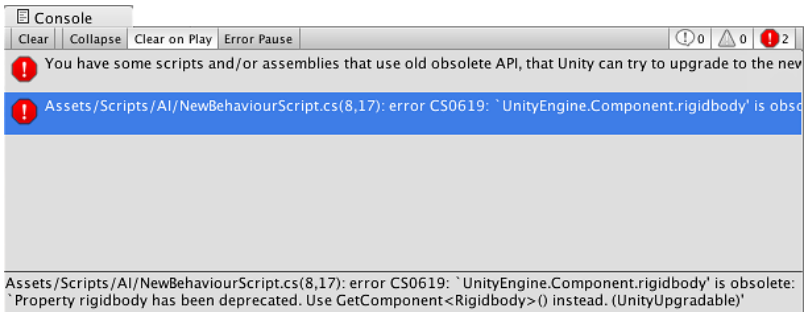
Hierarhijski pregled (eng. Hierarhy view) sadrži sve dijelove scene koji postoje u projektu (kamere, svjetla, 3D modeli i sl.). Kako se objekti dodaju i brišu u sceni, također se u Hierarchy view događaju iste akcije. Hierarchy omogućuje da kad se klikne na željeni objekt označi ga u sceni, a isto tako mogu se prebacivati objekti jedan u drugi.[36]

Alatna traka (eng. Toolbar) je traka koja se nalazi na vrhu programa. Sastoji se od dva dijela: jedan dio je predviđen za scenu (eng. Scene view), a drugi za pogled koji se vidi prilikom igranja (eng. Game view).[36]

Scene pregled (eng. Scene view) prikazuje prozor i prostor u kojem se radi igra. U njemu se odabiru i pomiču svi objekti koji postoje u sceni. U desnom gornjem uglu scene se nalazi Scene Gizmo. On prikazuje trenutnu orijentaciju kamere, te omogućuje brzu promjenu kuta gledanja. Svaka os X, Z, Y je obojana drugom bojom i pritiskom na nju može se postaviti kamera na pogled bez perspektive, promatrajući scenu duž odabrane osi.[36]

Slika 4.3. *Scene view [6]*

Konzola (eng. Console view) se aktivira u izborniku Windows > Console. Console prikazuje poruke, upozorenja, greške prilikom kreiranja igre. Najčešće greške koje se događaju vezane su uz skripte, te ova konzola uvelike pomaže kod traženja greške.[36]



Slika 4.4. *Console view*

Igra (eng. Game view) prikazuje ono što kamera vidi, tj. kako se kamera postavi tako igrač vidi tijekom igre. Na sceni može biti neograničen broj kamera, te se njihovom manipulacijom i prilagođavanjem mogu napraviti dobre scene u igri. Da bi se provjerilo da li igra radi koriste se gumbovi iz Toolbara predviđene za Game pogled.[36]

Slika 4.5. *Game view [7]*

4.2. Svjetla

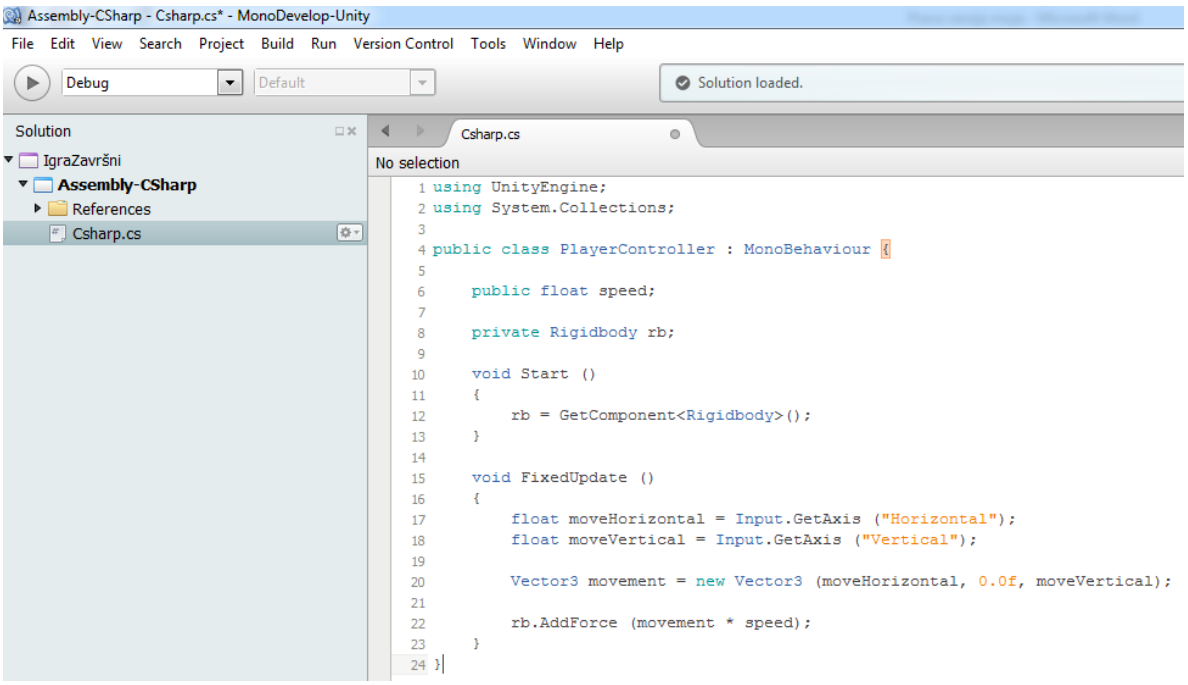
Bez svjetla se ništa ne može vidjeti u igri koja se napravi. Kao i u Mayi svjetla su slična i u Unity – u. Ona mogu simulirati sunce, vatru, eksploziju, lampu, svjetlo u kući itd. Korištenje svjetla je vrlo jednostavno. Treba se samo kreirati svjetlo željenog tipa i staviti ga gdje se želi na sceni. Kod renderiranja se mora odabrati ispravna vrsta, jer utječe na svjetla i sjene ovisno o zahtjevima igre. Kod svakog svjetla postoje određene opcije koje pospješuju detalje, kut, boju i veličinu. Svako svjetlo se koristi za određeni dio i namjenu u sceni. Tako spot svjetla mogu biti vrlo dobra za stvaranje svjetla koja dolaze iz prozora, point svjetla niskog intenziteta pružaju dubinu na sceni itd.[22]

Tablica 4.6. ***Vrste svjetla u Unit-u***

DIRECTIONAL	Svjetla su postavljena daleko i utječu na svaki objekt i sve što postoji u sceni isto kao i u Mayi. Pod kojim kutem se postavi to svjetlo, tako će se primijeniti upadni kut na svaki objekt. Najviše se koriste u otvorenim scenama za sunce i mjesečinu.
POINT	Svjetla sjaje podjednako u svim smjerovima poput žarulje. Intenzitet svjetla se smanjuje s udaljenosti, dosegnuvši nulu na određenom rasponu. Koriste se za simulaciju svjetiljke i drugih lokalnih izvora svjetlosti u sceni. Također mogu se koristiti kao iskra ili eksplozija.
SPOT	Poput point svjetla, spot svjetlo ima određeni položaj i raspon nad kojima svjetlost pada. Međutim, svjetlo je ograničeno na kut, koji je u obliku stošca. Koriste se za umjetne izvore svijetlost, kao što su svjetiljke, auto svjetla, reflektori. Spot svjetla će osvijetliti samo mali prostor na sceni i stvoriti efekte pod nekim željenim kutom.
AREA	Svjetlo se emitira pravokutno u svim smjerovima, ali samo s jedne strane pravokutnika. Svjetlo pada u određenom rasponu. Budući da se svjetlo emitira iz nekoliko smjerova odjednom, sjene su mekane u odnosu na druge vrste. Najbolje mogu prikazivati interijere kućne rasvjete, realne ulične rasvjete itd.

4.3. Skriptiranje

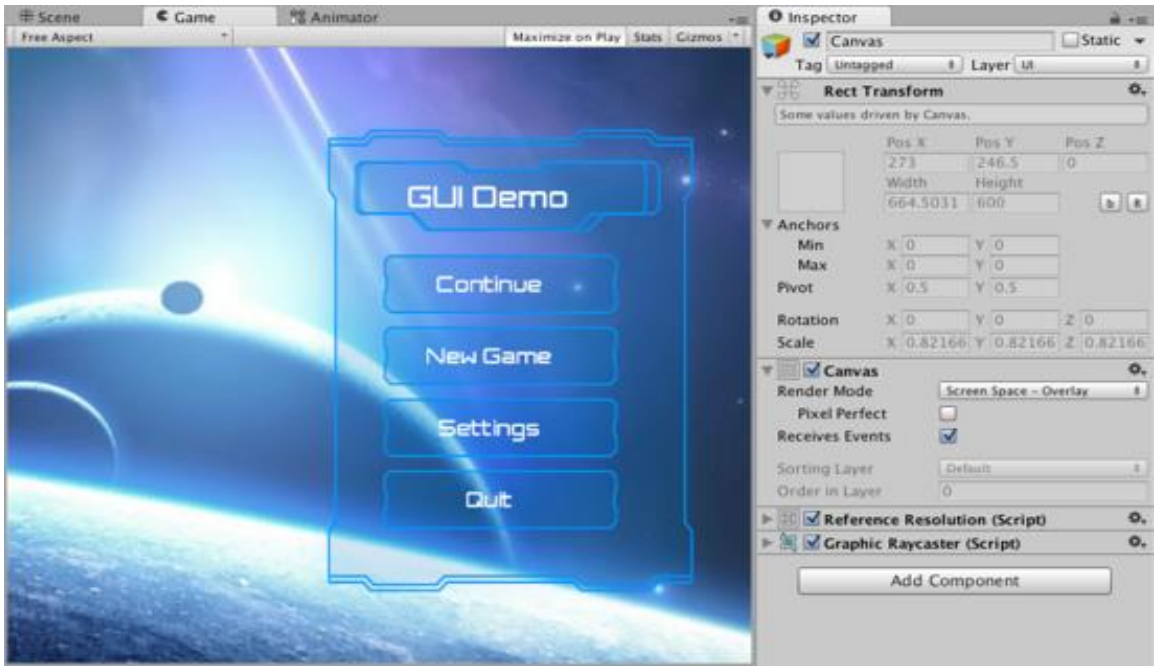
Unity nudi tri načina za programiranje: JavaScript (UnityScript), C# i Boo (uklonjen u Unity 5). Na početku se koristio najviše UnityScript/JavaScript i korišten je u većini turtorialа, dokumentacija i primjera kodova. Međutim danas je C # postao mnogo popularniji jezik u Unity, moćniji od UnityScript i dobro je prilagođen za mobilne uređaje. Najviše primjera sintaksi i kodova može se naći u formatu C#. Sa skriptama se mogu stvarati komponente koje se želi izvesti i koje će obavljati željenu funkciju. Skriptiranje se izvodi u Mono Develop programu koji obuhvaća većinu skriptnih jezika. Program je posebno dizajniran za skriptiranje, te korisniku pruža prednosti u rješavanju problema i grešaka na koje nailazi. UnityScript je JavaScript programski jezik, te je ujedno i najbolji izbor za početnike u programiranju. Većina developera danas koristi ovaj jezik, pa postoji mnogo primjera i lakše je doći do pomoći na raznim forumima i sličnim stranicama za programiranje. Privatne varijable stvaraju se na isti način kao i u C # sa naredbom private prije deklariranja nekog tipa. C# je malo teži programski jezik nego UnityScript, ali omogućuje programeru da ima potpunu i preciznu kontrolu. C# podržava neke značajke koje UnityScrit ne, kao što su događanja i generika. Boo koristi Python kao sintaksu te je strukturirano slično kao UnityScript. Vrlo malo ljudi koristi Boo, pa ako dođe do problema kod nekog koda, vrlo je teško pronaći pomoć. U Unity 5 Boo skriptni jezik je odbačen, a s njim i njegova sva dokumentacija.[24]



Slika 4.7. *Mono Develop - program za skriptiranje u Unity*

4.4. Grafička korisnička sučelja

UI omogućuje stvaranje korisničkog sučelja brzo i intuitivno. UI zapravo prikazuje platno, unutar kojeg su elementi sučelja. To je zapravo sučelje koje se pojavljuje prilikom pokretanja igre gdje se može odabrati nova igra, podešavati slideri (zvuk, brzina miša itd.). UI se koristi za početni meni u igricama u kojima se postavljaju opcije kojima se podešava, pokreće ili izlazi iz igre. Canavas je platno koje je u sceni prikazano kao pravokutnik. Na to platno se primjenjuju svi elementi tog platna i oni moraju biti dio njega odnosno njegovi elementi. Svaki element koji se stvori automatski postaje u hijerarhiji platna njegovo „dijete“. Elementi koji se koriste su najčešće slike, tekstovi, klizači. Platno ima Render postavku na kojem se može podesiti pogled na zaslonu ili prostor okoline. Tako postoje tri render moda: prekrivanje, kamera, prostor. Overlay stavlja elemente korisničkog sučelja na zaslonu renderirane na vrhu scene. Prilikom promjene razlučivosti zaslona, platno će automatski promijeniti veličinu. Kamera je slična kao prostor, ali u ovom se platno nalazi ispred određene kamere. Postavke kamere utječu na izgled korisničkog sučelja. Ako je kamera stavljena na perspektivni pogled, elementi će biti prikazani u tom pogledu,ako se mijenja rezolucija na kameri, platno će automatski promijeniti veličinu. World Space platno se ponaša kako i bilo koji drugi objekt u sceni. Veličina platna se mijenja ručno, tako se može staviti ispred ili iza drugih objekata. [10]



Slika 4.8. *Primjer platna sa elementima [8]*

4.5. Prednosti i nedostatci Unitya

Unity se može podijeliti na 3 dijela [25]:

1. Game engine – omogućuje stvaranje, testiranje, igranje igre u različitim vrstama okruženja.
2. Aplikacija - dizajn ili korisničko sučelje je povezano zajednički sa grafičkim pregledom opcija i kontrolom za igranje.
3. Code editor – IDE (integrirano razvojno okruženje) sadrži text editor za pisanje koda.

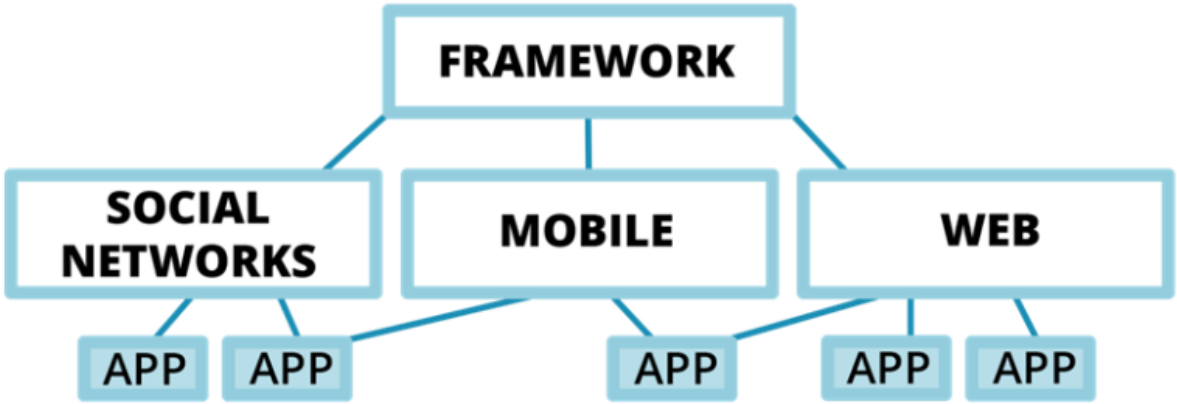
Unity je odlična platforma za početnike koji žele saznati više o razvoju igara. Osim toga, pruža programerima online tutorijale za razvoj igre, te različite tehnike za izgradnju igre visoke kvalitete. Sučelje je složeno za početnike, te je sam dizajn i pozadina napravljena jednostavnije da bi se smanjila kompleksnost programa. Programeri mogu napraviti audio i video u igrama, stvoriti likove i scene izravno pomoću vizualnog sučelja. Što se tiče renderiranja Unity koristi OpenGL, Direct3D i OpenGIES (mobilne platforme).[25]

Tablica 4.9 *Prednosti Unitya*

PODRŠKA OSTALIH PLATFORMI (CROSS PLATFORM)	Velika podrška na razne platforme (oko 95%) uključuje računalo, iOS, web i mobilne platforme.
IDE	Odlična grafika nudi napredne vizualne i zvučne efekte sa velikom materijalnom podrškom. Optimizacija pixela je odlična i za mobilne uređaje.
ODLIČNA GRAFIKA	Integrirano razvojno okruženje za sve platforme je najmoćniji alat u korisničkom okruženju.
DOKUMENTACIJA	Detaljna dokumentacija i podrška za programere.
KOD	Kod je vrlo stabilan u odnosu na druge jezike, a sastoji se od velike arhitekture za dobre performanse i smanjenje pogrešaka (C##)
PROFILER	Profiler se koristi za optimizaciju igre i spriječava „curenje“ memorije.

Kao nedostatak kod Unitya je cijena. Cijena Unity Pro uz sve dodatke (iOS, Android) je 4500 $. Za početnika je možda previše novca da bi obavljao aplikaciju za iOS i Android tržišta. Ipak tu postoji i Personal Edition koji ne sadrži sve opcije i nadogradnje kao Unity Pro, pa zbog toga služi za velike početnike i one koji se tek upoznavaju sa Unity. No ako to korisniku nije dovoljno može uzeti Unity Pro licencu za 75 $ mjesečno.[26]

Kao dodatne prednosti može se još navesti i Unity Asset Store koji sadrži mnoge korisne dodatke i sredstva za razvoj igre. Mogu se koristiti besplatni alati i alati koji se plaćaju. Kada korisnik pronađe alat koji želi, on se može direktno preuzeti i staviti u Unity. Postoji još alat koji olakšava mnoge aspekte razvoja igre poput pohrane datoteka na poslužitelju, primanje sadržaja s poslužitelja, pohrane igračkih profila, integracija s društvenim mrežama. Taj alat se naziva Multiplatform Solutions Freamework (MSF). Prije svega MSF omogućuje da koristite igrači profil na različitim uređajima. Igrač može igrati igru na iOS, Android ili preko weba i njihov napredak bit će spremljen svaki put u jednom profilu bez obzira na kojoj platformi korisnik igra.[26]



Slika 4.10. *MSF logika za svaku platformu [9]*

Spremanje i primanje datoteka je druga mogućnost MSF-a. Ako je igra cross-platforma ili ima neke proširene konfiguracije (npr. sposobnost za promjenu levela nakon objavljivanja dodataka) trebalo bi imati neki mehanizam za ažuriranje podataka i konfiguraciju igre. Treća mogućnost je integracija s društvenim mrežama (pozivanje prijatelja, slanje poruka, obavijesti, poslati darove itd.).[26]

4.6. Konkurencija

Kao konkurencija Unityu danas su najpopularniji UE4 (Unreal Engine 4), Marmelada, Cocos2D, Source2, CryEngine. Svaki od njih ima svoje prednosti, no ako bi se odabrala cross-platforma Unity je na samom vrhu.

Najveći konkurent Unityu trenutno je UE4. U grafičkom pogledu Unreal Engine je bolji od Unitya nudi veće mogućnosti i bolji prikaz. Korisnici mogu stvoriti grafiku u rangu sa igrama koje su puštene na konzole. Mogu se dodati čestice složenih simulacijskih sustava za napredno dinamičnu rasvjetu. Može se reći da se u UE4 mogu stvoriti bilo koji tipovi vizualnog stila koje korisnik želi u 2D ili 3D.Što se tiče cijene ovaj program je besplatan, ali pri izradi igre korisnik je dužan dati naknadu od 5% od svega što zaradi. Što se tiče programskog jezika UE4 koristi C++ skriptni jezik, a ima i vizualno skriptiranje u kojoj tehnički se ne treba napisati niti jedna linija koda. Odličan je za brzo programiranje koristeći prototipove, a s njim se može stvoriti čak i cijela igra. Odličan je za korisnike koji baš i nisu dobri u programiranju, a žele napraviti igru. Isto ako i Unity UE4 ima mogućnost preuzimanja raznih dodataka za igru (zvukovi, likovi, rekviziti itd.).[27]

CryENGINE je iznimno snažan game engine koje je uveden u prvoj Far Cry igri. Osmišljen je kako bi se koristio na PC platformi i konzolama (Playstation4 i Xbox One). Grafičke mogućnosti nadmašuju Unty, ali su u rangu s UE4. Program je teže naučiti, ako korisnik nema iskustvo sa drugim game engine-om.[28]

4.7. Kompatibilinost Unity s Mayom , formati i sistemske karakteristike

Unity podržava većinu formata za uvoz iz drugih programa kao što su: Mya, Cinema 4D. 3DS Max, Cheetah 3D, Blender, SketchUP itd. Kod ulaznih tekstura trebaju se teksture pohraniti u mapu po imenu „Textures“, te će se teksture automatski spojiti u Unity iz drugog programa. 3D formati koji se uvoze u Unity su: .FBX ili .OBJ, .Max, .Bled. Uz te formate Unity može uvesti formate kroz konverziju kao što su .MAX, .MB, .MA itd. Prednosti kod tih uvoza su što je brz proces uvoza a nedostatak je što velike datoteke mogu usporiti Unity, te su manje valjanosti i dolazi do poteškoća.[29]

Kod uvoza datoteka iz Maye, da bi uvezli .mb i .ma datoteke korisnik mora imati instaliranu Mayu, te preko Unity uvesti te vrste datoteka. To je potrebno zbog toga kad se uvozi file iz Maye ona se automatski otvara u pozadini, pa tako Unity komunicira sa Mayom i konventira .mb file u format koji Unity može prepoznati. Ako korisnik nema Mayu na

računalu, a želi uvesti Maya datoteku s drugog računala, koristi se .FBX format. Pri izvozu FBX datoteke u cjelini, može se izvesti najviše 65000 poligona po datoteci. Unity iz Maye može uvesti sve čvorove, rotacije, rastezanja, imena, mrežu sa vertex bojama, materijale sa teksturama, animacije, Bone animacije, BlendShapes.[30]

Tablica 4.11. *Neki od glavnih ulaznih formata koje podržava Unity3D*

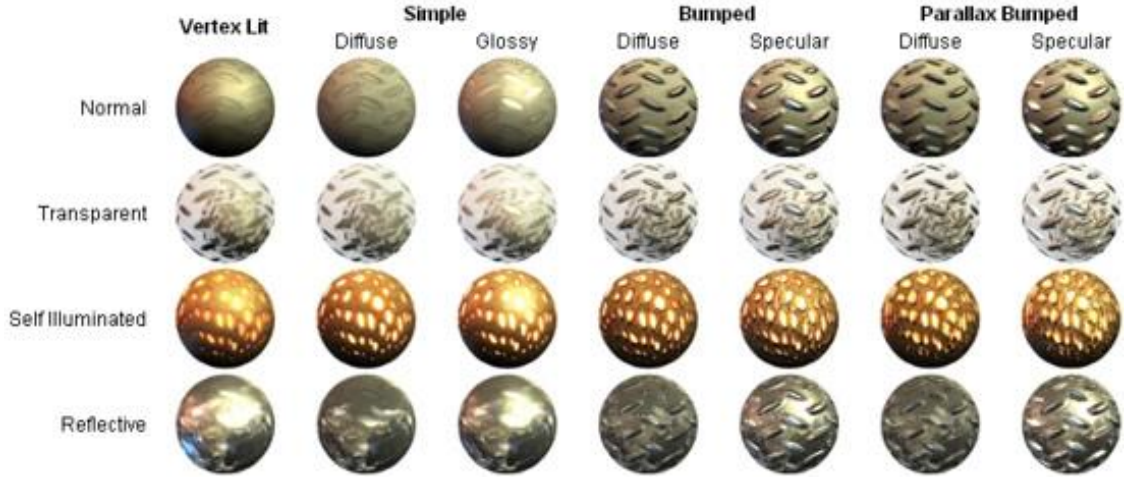
3D FORMATI	.fbx, .3DS, .dxf, .obj, .max, .mb, .ma, .blend
AUDIO FORMATI	.mp3, .ogg, .wav, .mod, .it, .s3m, .xm, .aif
MOVIE FORMATI	.mp4, .3gp, .mov, .mpv

Sistemske karakteristike koje zahtjeva Unity 5.2 se mogu podijeliti na 2 dijela: za izradu igre i za pokretanje igre. Za izradu igre potrebno je imati operacijski sustav Windows ili Mac OS X. Što se tiče grafičkog zahtjeva potrebna je grafička kartica sa DX9, odnosno sve što je napravljeno od 2004. godine bi trebalo raditi. Pokretanje igre sa Unityem nije toliko zahtjevno, te se može pokrenuti poprilično i sa manjim zahtjevima ovisno o veličini i složenosti projekta. Grafički zahtjevi su isti kao i za izradu igre. Web player podržava Chrome, Safari, Firefox, IE, Safari i dr. iOS zahtjeva iOS 6.0 ili noviji, dok Android verzija mora biti OS 2.3.1 ili novija.[31]

4.8. Sjenčanja, materijali i teksture

Renderiranje u Unityu temelji se na materijalima, sjenčanjima i teksturama. Sjenčanja sadrže kod (skripte) koji definira kakva svojstva se upotrebljavaju, a materijali omogućuju podešavanje svojstava, te definiraju kako podloga treba biti renderirana. Kako bi se stvorio novi materijal u glavnom izborniku se odabere Assets -> Create -> Material. Kad je materijal kreiran, može se primijeniti na objekt, te sva njegova svojstva koja se mogu podešavati u inspectoru. Kako bi se primijenio, samo se povuče mišem na željeni objekt u sceni. Inspector sadrži svojstva za podešavanje i odabiranje sjenčanja koje korisnik želi koristiti za željeni objekt. Svojstva mogu biti boje, klizači. Tekstura, brojevi ili vektori. Uz standardno sjenčanje postoje i druge kategorije za specijalne namjere kao što su: FX (rasvjeta i stakleni predmeti), GUI (grafičko korisničko sučelje), Mobitel, Priroda, Čestice, Skybox (prikaz i podešavanje geometrije neba) itd. [36]

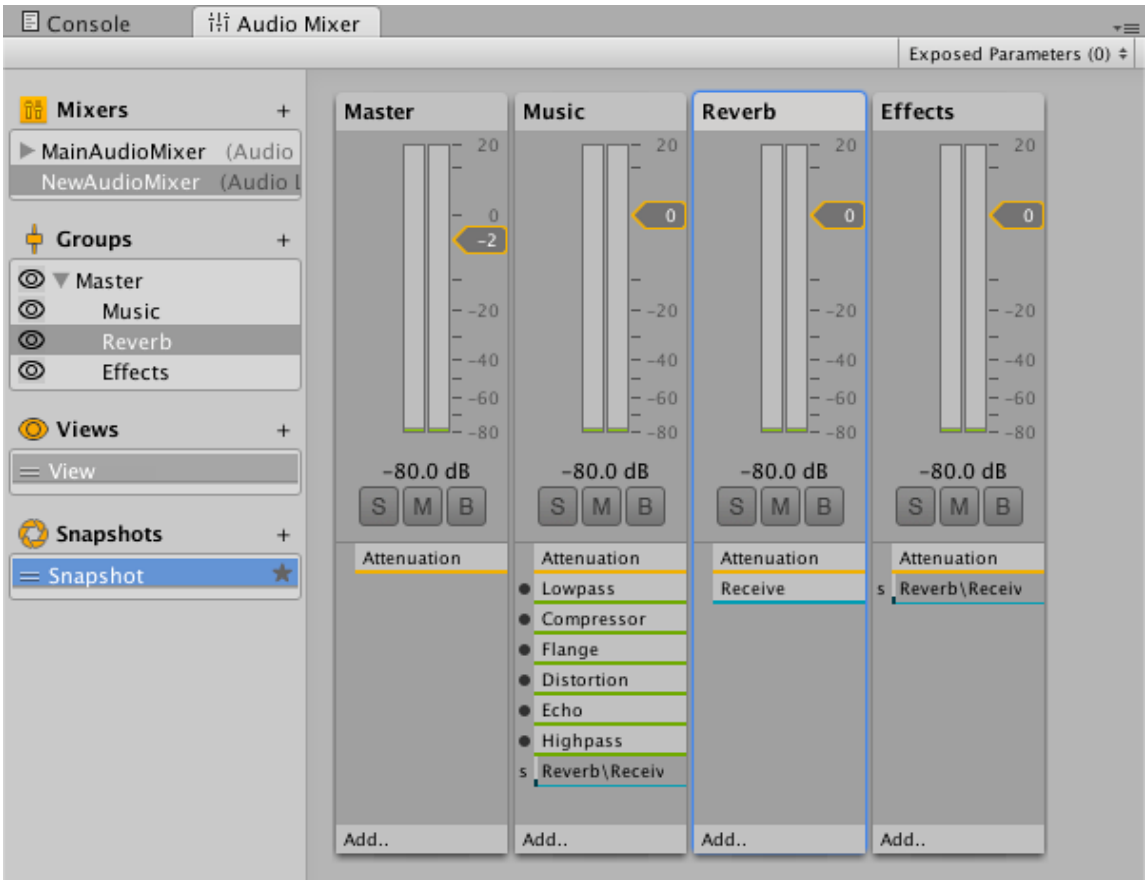
Standardno sjenčanje je sjenčanje koje se najviše koristi u sceni. Omogućuje velik raspon vrsta sjenčanja kao što su difuzno, reflektirajuće, refleksije. Sve vrste se mogu kombinirati u jedno sjenčanje, sa svim njihovim značajkama. Prednost toga je što se dobiva realna i uvjerljiva distribucija svjetla i sjene u svim modelima. Najviše se upotrebljava za materijale kao što su drvo, zid, asfalt, staklo, plastika, metal itd. Uz standardno sjenčanje koristi se i standardno refleksno sjenčanje koje odbija svjetlost tj. reflektiraju je. Unity 5 je prva verzija koja nudi taj način renderiranja za rasvjetu modela zvanu Fizičko sjenčanje (eng. Physically Based Shading PBS), koje simulira interakciju između materijala i svjetlosti u stvarnom vremenu (real-time graphic). Najviše se koristi za materijale kao što su ogledalo, tekućine, ispolirano drvo, glatke površine i sl. [36]



Slika 4.11. *Vrste sjenčanja i njihov izgled [10]*

4.9. Zvuk

Svaka igra je nepotpuna bez zvuka (eng. Audio). Za simulaciju zvuka u Unity potrebno je da se zvuk podesi na željeni objekt, te tako zvuk potječe iz njega. Prijemnik zvuka (eng. Audio Listener) se postavlja u objekt kao što je kamera. Omogućuje igraču da čuje zvukove u sceni, prema udaljenosti. Važno je spomenuti kako je za ispravan rad unutar scene potreban samo jedan prijemnik zvuka. Drugi dio zvuka je izvor zvuka (eng. Audio Source) koji na određenom objektu pušta zvuk unutar svog dometa u sceni. Zvuk može postići Doplerov efekt ovisno o tome koliko je igrač udaljen od zone izobličenja (eng. Reverb Zones). Zona se može podešavati do određene veličine, a koristi se najviše za prelaske područja gdje postoji zvuk u prostor zvuka gdje ga nema. Za simulaciju zvuka u tunelu, pećinama i sl. koriste se audio filteri (eng. Audio Filters). Unity sadrži i Audio mixer koji se koristi za podešavanje efekata, performanse i razine audio zvuka. Svaki izvor zvuka omogućuje opciju ponavljanja (eng. Loop), kojom se mogu simulirati kratki zvukovi kao što je voda, vatra, vodopad i sl. Napredne postavke omogućuju funkcije prigušenja, prioriteta, postavljanja minimalne i maksimalne udaljenosti itd. [36]

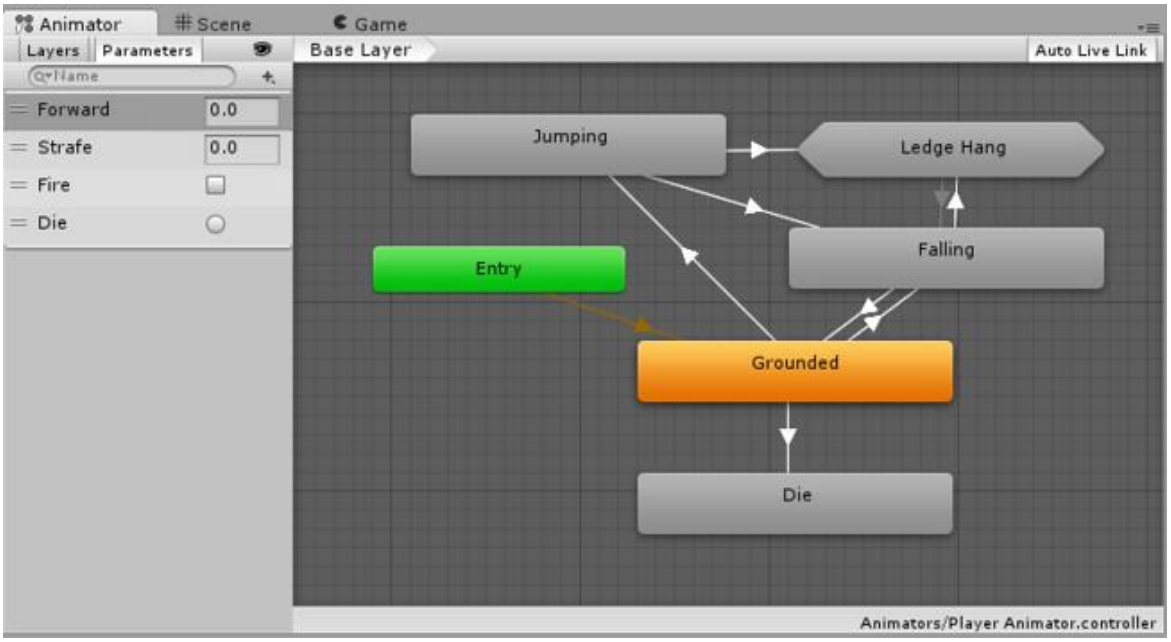


Slika 4.12. *Audio mixer [11]*

4.10. Animacije

Animcije u programu Unity moguće su na više načina: stvaranje animacija izravno u programu, uvozom animacija iz drugih programa, te animiranje pomoću skripti (eng. Mecanim). Animacije koje mogu biti uvezene iz vanjskih izvora mogu biti : Humanoidi, animacije izrađene u vanjskom programu 3D Max, Maya, animacijski setovi (Unity trgovina). Animacije stvorene u Unityu omogućuju: položaj, rotaciju, obujam objekta, svojstva (materijali, bojem intenzitet svjetla, volumen zvuka), vrijeme pozivanja funkcija unutar vlastite skripte. [36]

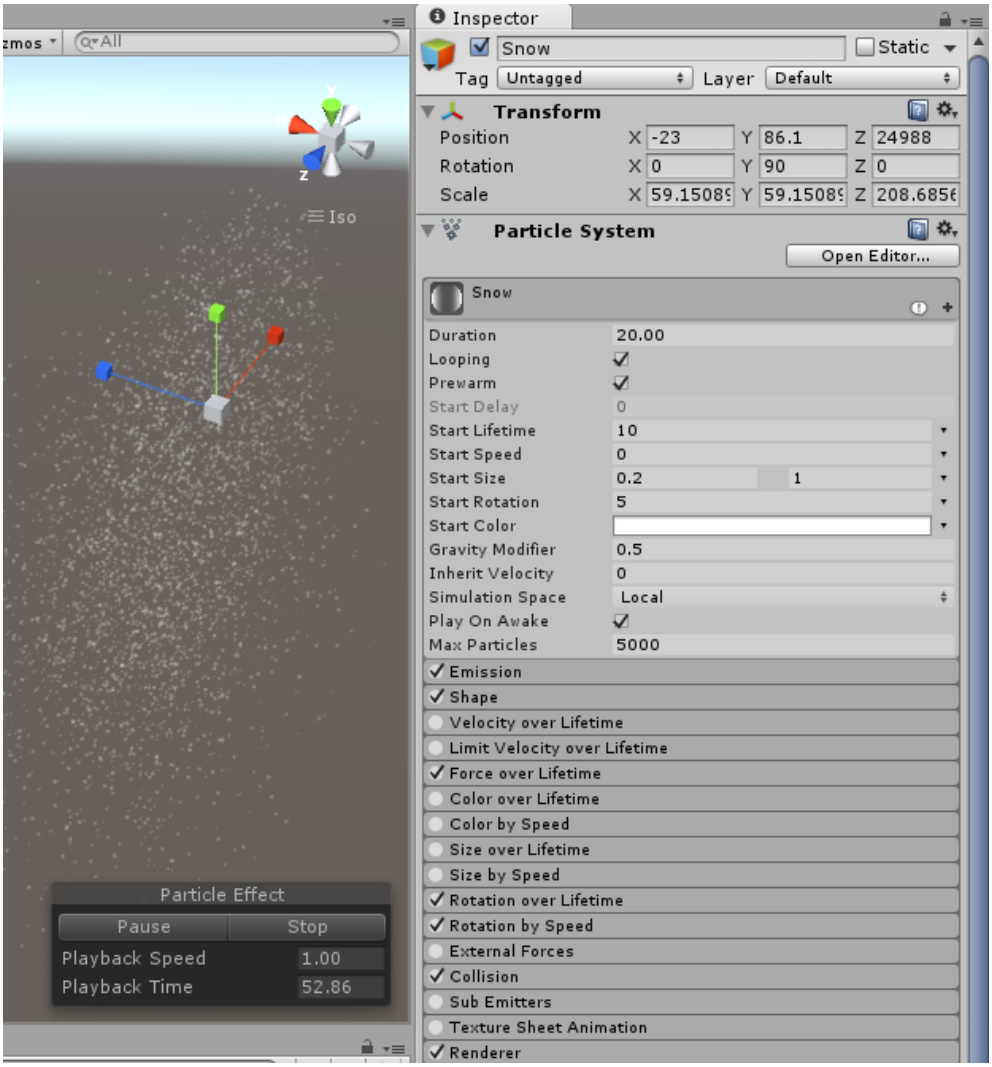
Unityu kontroler animacije (engl Animtor Controller) omogućuje organiziranje skupa animacija za objekt ili animacije lika. Kontroler upravlja razne animacije stanja i prijelaze između njih pomoću stanja stroja (eng State Machine). Kontroler animacije može se koristiti kod više animacija i prebacivanja između njih (primjer: prebacivanje iz šetnje animacije, na skok kad god se pritisne razmak). Animacijski parametri su varijable koje su definirane u animacijskom kontroleru, te se njima može pristupiti iz skripte u kojoj se postavlja vrijednost i što određeni parametar radi. Uz parametre tu su i prijelazi koji omogućuju stanje prijelaza iz jednog u drugi (eng. State Machine Transitions). [36]



Slika 4.13. *Kontroler animacije i parametri [12]*

4.11. Sustav čestica

Čestice su male slike koje se prikazuju u velikom broju te tako čine njezin sustav. Svaka čestica je mali dio tekućine ili fluida kojom se može dobiti simulacije dima, oblaka, snijega, kiše itd. Svaka čestica ima svoj vijek trajanja (početak i kraj). Počinje emitiranjem iz jedne točke, te nasumice kako su postavljene postavke proteže se u određenom smjeru, određenog oblika. Kad dođe do zadanog krajnjeg vremena čestica se uništava. Čestice se mogu širiti u razne geometrijske likove, te tako definiraju oblik sustava čestica kao cjelinu (kugla, polukugla, stožac, kutija, mreža, kružnica, kut). Čestice se dodaju odlaskom na glavni izbornik u GameObject -> Particle Syste. U inspektoru se podešavaju određene karakteristike čestice, njezina brzina, sile, emisija, oblik, vijek trajanja, boja i sl. [36]



Slika 4.14. *Sustav čestica (snijeg) i njezine postavke*

5. Računalne igre i zahtjevi

Kako napreduje grafika i njezina tehnologija tako zahtjevi za igre postaju veći. Prvo što se razmatra je broj poligona, što je njihov broj veći za pojedini model više je potrebno sustavu da učita taj model. Mnogo različitih faktora utječe na vrijeme izrade igre, no nije uvijek slučaj kada se koristi mali broj poligona da će vrijeme očitavanja biti brže. Mnogi ostali faktori utječu na zahtjeve igre kao što su teksture, materijali, fizika, skripte itd.[33]

U igri postoji način da bi se uklonili nevidljivi poligoni, odnosno poligoni koji se ne vide u sceni. Dobar primjer je kod FPS igara gdje je prikazano oružje. Tako dijelovi koje korisnik ne vidi i koje nikad neće vidjeti u igri su uklonjeni ili napravljeni sa što manje detalja. Nakon što se naprave sve teksture moraju se izgraditi sjene za igru. Ovaj proces uključuje kombiniranje različitih vrsta tekstura zajedno u jedan objekt poznat kao Shader ili materijal, tako da oni zajedno daju modelu izgled i dojam koji se želi dobiti.[35]

5.1. Ograničenja poligona

Broj poligona koji se koriste u sceni ovise o kvaliteti i zahtjevima koje su potrebne korisniku. Za mobilne uređaje broj poligona po objektu je negdje između 300 i 1500, dok je za desktop platforme idealan raspon oko 1500 i 4000. Ako igra ima mnogo objekata na zaslonu može doći do usporenja, te je potrebno smanjiti broj poligona da bi se dobila idealna brzina igre. Igra Half Life 2 koristi 2500 – 5000 poligonalnih trokuta po karakteru, a trenutne AAA (najbolje i najzahtjevnije) igre za PS3 ili Xbox 360 rade na 5000 – 7000 poligonalnih trokuta.[32]

5.2. Grafičke performanse (CPU i GPU)

Grafički problemi i zahtjevi za igru se nalaze na GPU (grafička procesorska jedinica) ili CPU (procesor). Prvo pravilo kod optimizacije je naći gdje je problem, jer optimizacija GPU u odnosu na CPU su drugačije. GPU je ograničen na memorijsku propusnost, što znači da prilikom pokretanja igre u nižim rezolucijama igra radi brže. CPU je ograničen na broj serija koje su potrebne za renderiranje. Broj skripti ili fizike koje sadrži igra mnogo utječu na brzinu. Prilikom renderiranja bilo kojeg objekta na zaslonu, CPU ima zadaću očitati koja svjetla utječu na taj objekt, postavljanje sjena i shader parametri, slanje naredbi za grafičku karticu. Zato kad ima mnogo vidljivih objekata u sceni CPU to sve mora zbrojiti i to može

potrajati. Kako bi se ubrzao proces očitavanja dobro je da se smanji broj vidljivih objekata.[33]

Optimizacija CPU[33]:

1. Kombiniranje bliskih objekata zajedno, ručno ili pomoću Unity.
2. Korištenje manje materijala u objektima, stavljanjem odvojenih tekstura u veću teksturu.
3. Korištenje manje stvari koji uzrokuju da se predmeti očitavaju više puta (refleksije sjene, svjetla itd.).

Optimizacija GPU[33]:

1. Smanjenje korištenja više trokuta nego što je potrebno.
2. Broj UV mapiranja i oštrih rubova smanjiti na minimum.

Treba imati na umu da stvarni broj vrhova koji grafički hardver mora obraditi nije isti kod 3D aplikacija. Za grafičku karticu neki geometrijski vrhovi morat će se podijeliti u dvije ili više logičkih vrhova prilikom renderirnja. Što se tiče tekstura za GPU komprimiranjem tekstura će se smanjiti veličina, što rezultira bržem vremenu očitanja i manje memorije, a može dramatično povećati učinkovitost renderiranja. Komprimirane teksture koriste samo dio memorije koje su potrebne.[33]

5.3. Grubo modeliranje

Low poly ima različite tehničke razlike u odnosu na glavni realni stil. Prvi primjetni atribut je polycount i teksture. Idealna razlučivost za lika u modernoj igri je u rasponu od 1500 do 4000 poligona. U slučaju da je za lik potreban veći broj poligona može ići do 7000 ili dalje ovisno o procesorskoj snazi. Za teksturne mape veličina varira između 1024x1024 i 2048x2048 piksela. Što se tiče broja poligona kreće se oko 1000 poligonalnih trokuta, prema nekoliko stotina. Teksture su općenito što manje u rasponu od 128x128 do 512x512 piksela.[34]

6. Maya – praktični dio

Modeli koji se koristiti u aplikaciji napravljeni su u programu Autodesk Maya 2016. Prvi korak u izradi modela je modeliranje. Svi objekti koji se koriste u ovoj igri rađeni su poligonalnim modeliranjem. Drugi korak je teksturiranje gdje su korišteni materijali Lambert, Blinn i Phong. Neki od modela su animirani, te je potrebno napraviti kostura (eng. Riginng) prilagođeno tom objektu, da bi se dobili dijelovi koji se žele savijati na mjestu kojem se želi. Na samom kraju dolazi animacija u kojem se rade željeni pokreti na objektu. Svaki objekt se sprema u formatu .fbx.

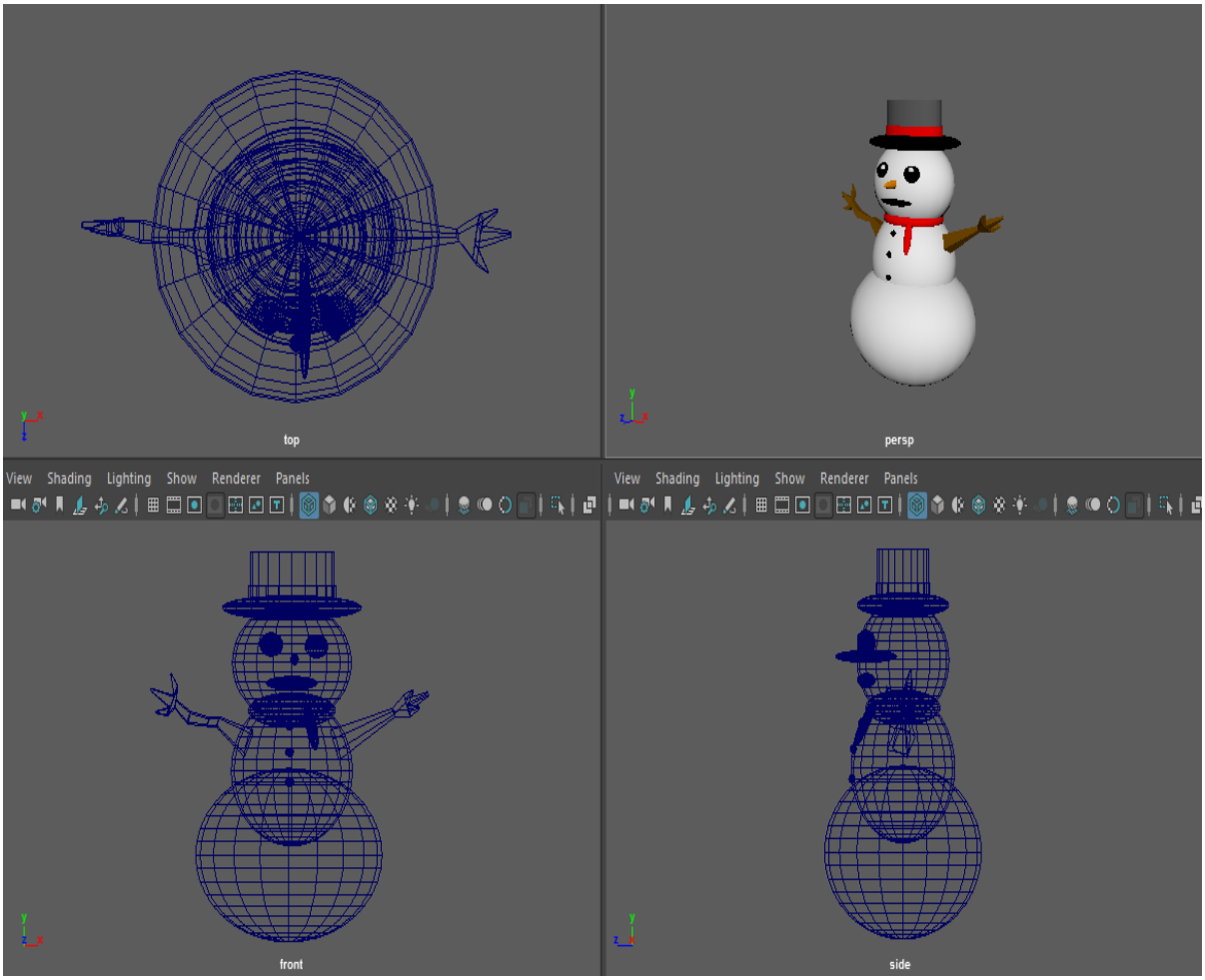
Modeli koji su rađeni u Mayi potrebni za ovu igru su:

Tablica 6.1. *Modeli rađeni u Mayi*

1.	Snjegović	**11.**	Kuća
2.	Neprijatelj (eng. Enemy)	**12.**	Lampa
3.	Kontrolna točka (eng. Checkpoint)	**13.**	Level_1_2
4.	Danger_do_not_enter	**14.**	Metal
5.	Hammer	**15.**	Novic
6.	Drvo	**16.**	Ograda
7.	Finish	**17.**	Poklon_1_2_3
8.	Drvo_pomicanje	**18.**	Slika_alpine_path
9.	Staklo	**19.**	Strelica
10.	Dizalo_ice	**20.**	Tree

6.1. Snjegović

Prilikom modeliranja naviše se koristi alat Polygon Sphere. Dodavanjem većih i manjih Sphere-a dobiva se oblik tijela snjegovića. Ruke su napravljene pomoću alata Cube, te pomoću alata extrude dobiva se željeni i konačni oblik ruku. Odabirom točaka (eng. vertex) pomiču se na željeni način da bi se dobio oblik ruke i prstiju. Šešir snjegovića dobije se pomoću alata Cylinder. Materijal korišten za teskuriranje modela je Lambert, preko kojeg se daju željene boje na objektu. Svi dijelovi se spoje u jedno tijelo pomoću alata Combine.

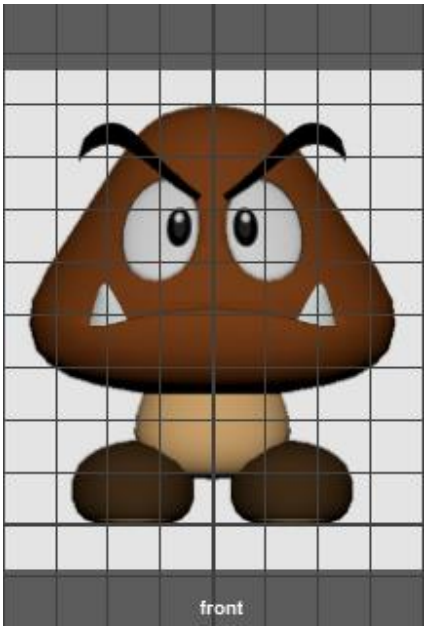


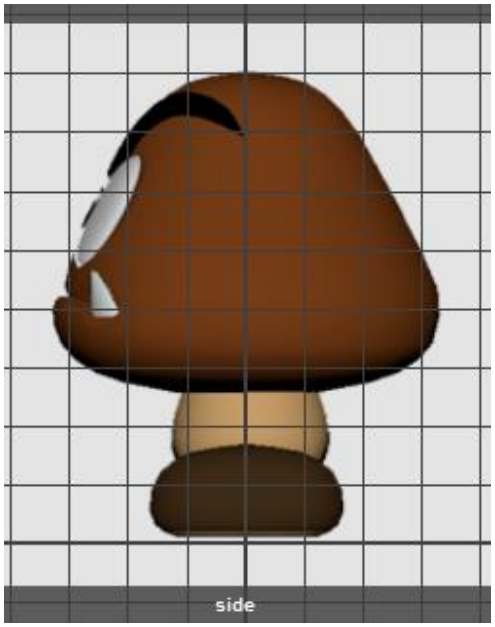
Slika 6.2. *Model snjegovića (top, front, side i persp view)*

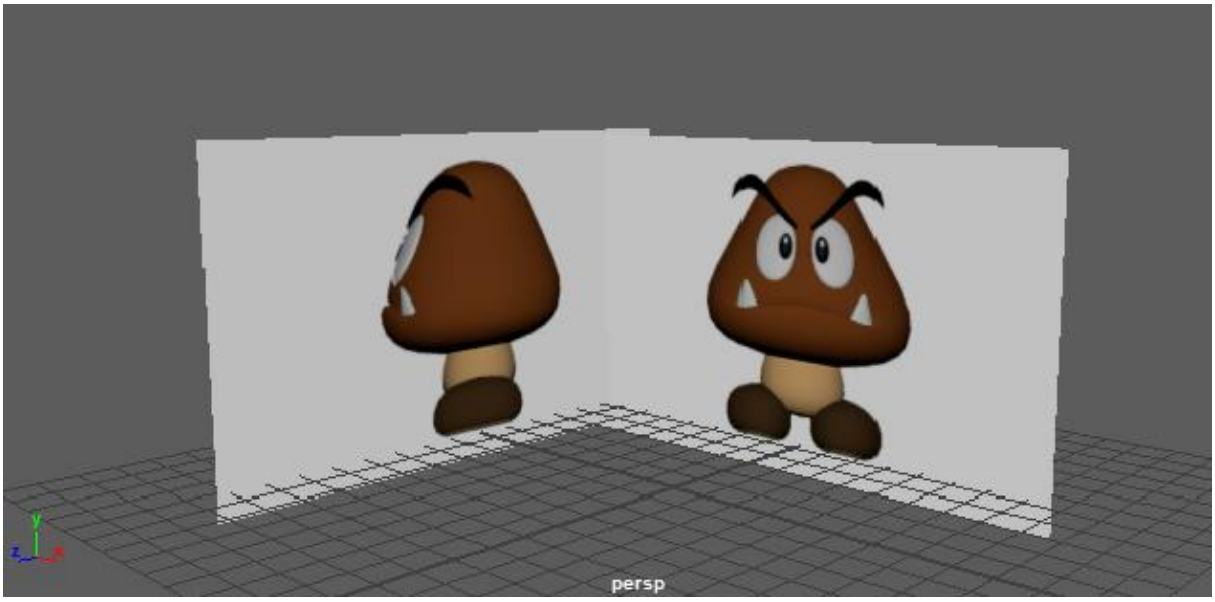
Animacija „Snjegovića“ nije rađena u Mayi, nego u programu Unity što će se kasnije objasniti u poglavlju o animacijama u Unityu. Animacija pokreta je jednostavna, te ju nije potrebno raditi u Mayi, jer se svi ti pokreti mogu napraviti u Unityu.

6.2. Neprijatelj (eng. Enemy)

Prije početka modeliranja potrebno je dodati dva Polygon Plane-a (plohe) na koje se stavljaju slike preko kojih se izrađuje model. Postave se pod kutovima za pogled sa strane i pogled sprijeda (eng. front i side view).

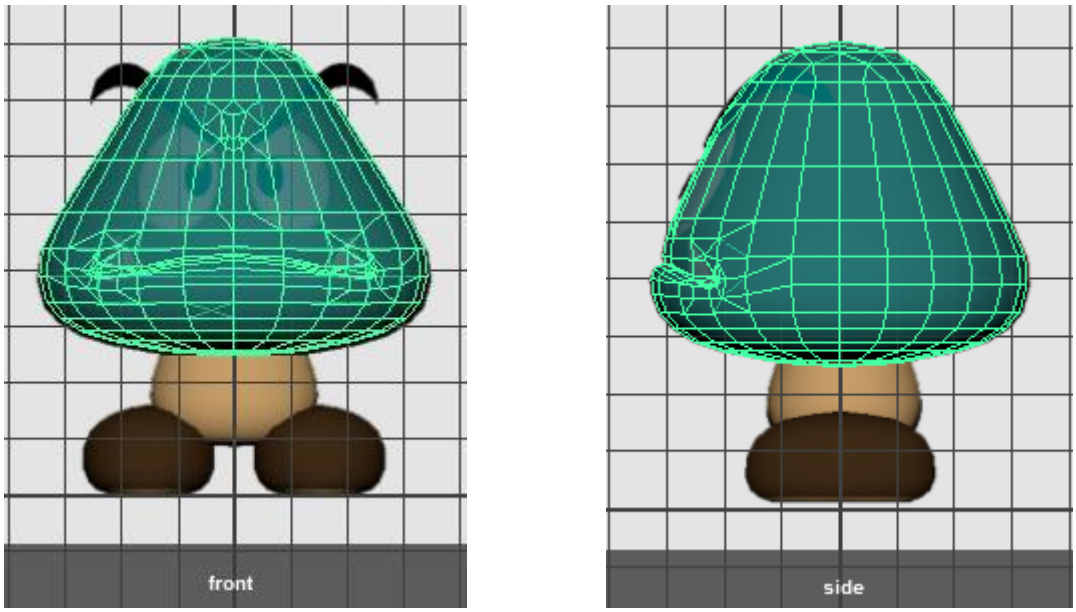






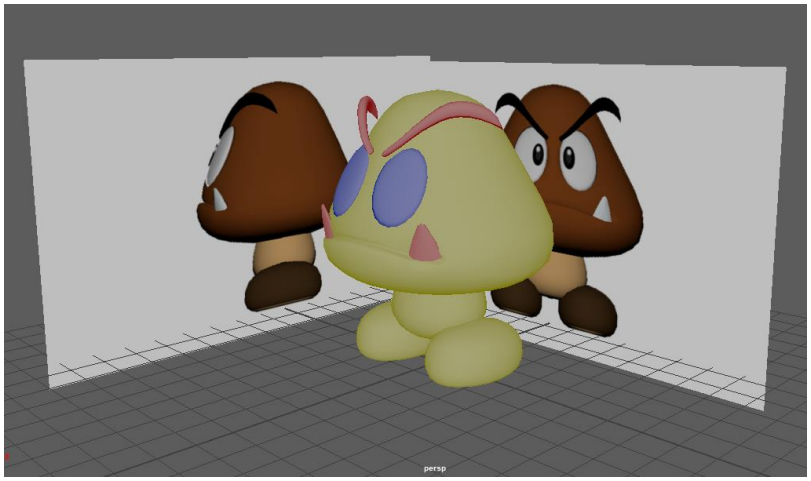
Slika 6.3., 6.4. i 6.5. *Prikaz slike stavljene postavljene na dvije plohe preko kojih će se modelirati model neprijatelja (front, side i persp view)*

Za modeliranje tijela koristi se alat Polygon Cylinder (valjak) te se postavlja u sredinu modela. Koristeći alat Extrude dodaju se dodatne točke koje se postavljaju prema obliku tijela. Preciznije postavljanje točaka i konačni oblik glave dobiva se koristeći točke (eng. vertex), te se tako dobiva finalni oblik glave gljive. Tijelo i noge izrađuju se iz osnovnog Polygon Sphere (kugla), te na isti način kako se modelira glava oblikuju se željeni poligoni, da bi se dobio konačni oblik tijela i nogu.



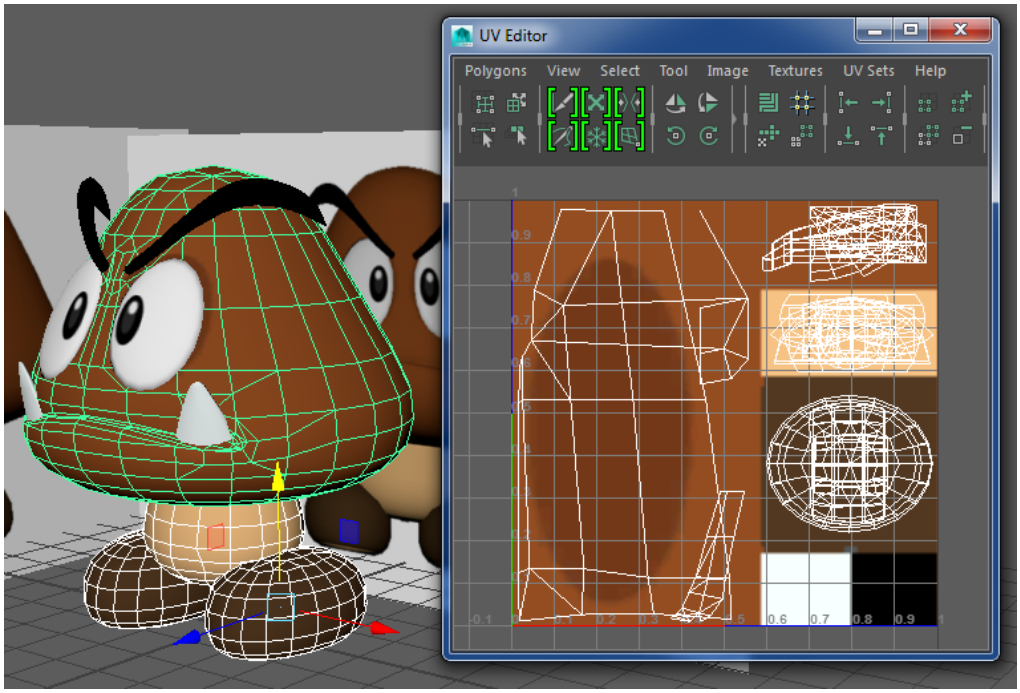
Slika 6.6. i 6.7. *Prikaz modelirane glave prema pozadinskoj slici (front i side view)*

Modeliranje obrva se radi na način da se koristi alat Polygon Cylinder, te alatom Extrude se oblikuju obrve. Druga se obrva dublicira te se postavi translate na -1 da bi se dobila obrnuta strana obrve. Oči se izrađuju Polygon Sphere-om te se oblikuju prema željenom obliku.

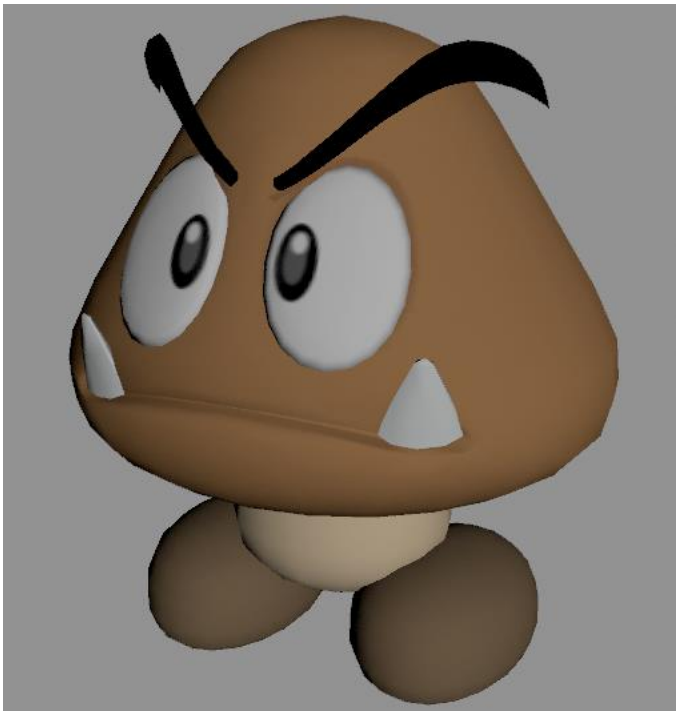


Slika 6.8. *Izgled modela (bez tekstura)*

Nakon modeliranja slijedi teksturiranje koje se izvodi tako da se označi poligon koji se želi teksturirati, te se dodaje novi materijal (eng. Assign new material). Odabire se materijal Lambert, na kojemu se klikom na dodatne opcije boje odabere opcija file. Odabire se file (slika) koja će se postaviti (mapirati) kako se želi preko UV editor-a.

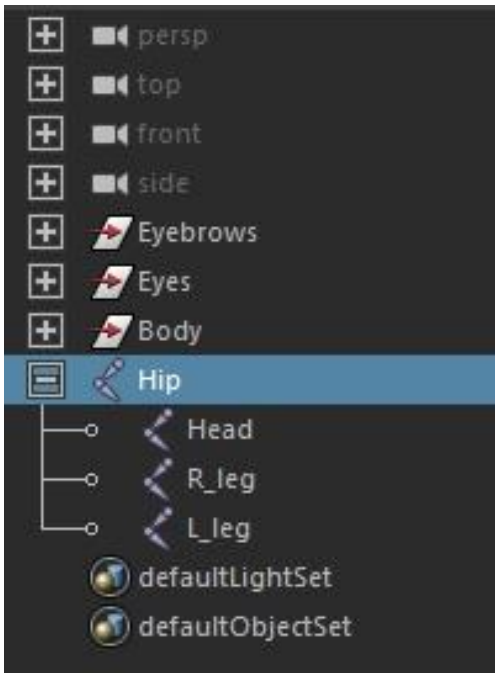


Slika 6.9. *Mapiranje i pozicioniranje tekstura u UV Editor-u*



Slika 6.10. *Konačan izgled modela nakon renderiranja*

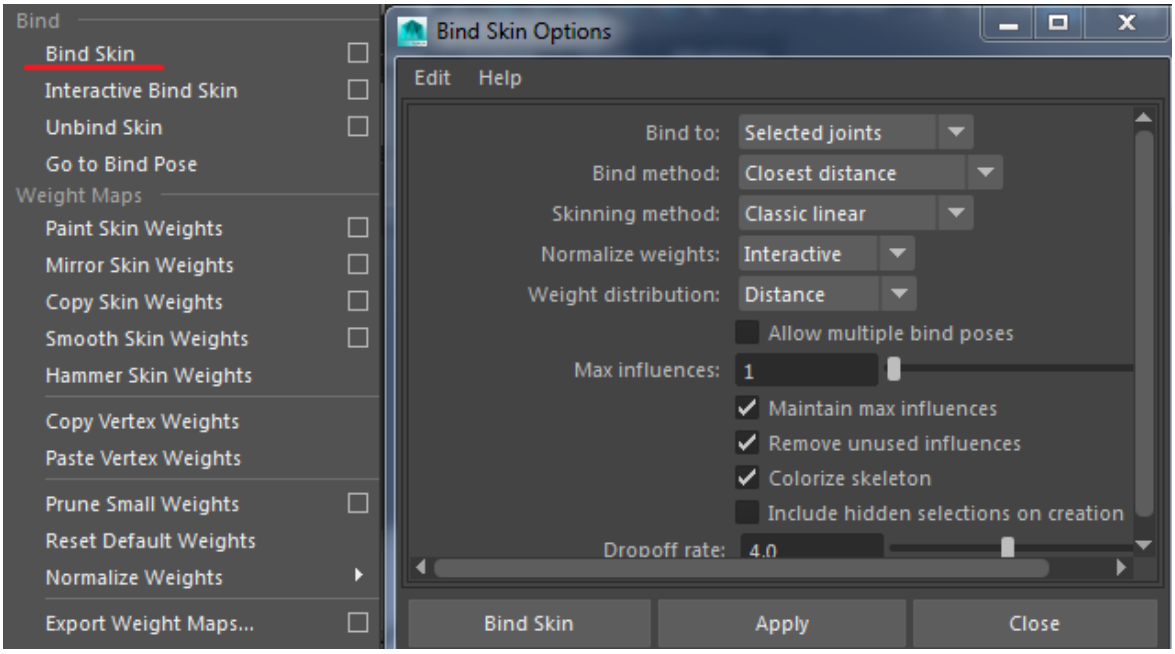
Idući korak je izrada kostura (eng. rigging). Da bi se kreirao kostur koristi se alat Create Joint, te se kreiraju i povezuju kosti po željenom objektu. Tako se postavlja glavna kost (eng. root), te iz nje se spajaju dvije noge i glava. Kostur se postavi na sredinu modela, da bi se pri animaciji dobili pripadajući pokreti tijela. Svakoj kosti se doda ime da se lakše pronađe prilikom spajanja i raspoznavanja. Da bi se došlo do hijerarhije odabere se opcija Window -> Outliner te se preko nje vide svi objekti izrađeni u sceni.



Slika 6.11. *Hijerarhija „Enemy-a“*

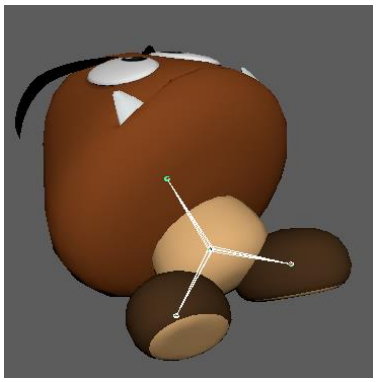
Slika 6.12. *Alatna traka za izradu kostura i označeni glavni alat Create Joint*

Nakon izrade kostura slijedi njegovo spajanje sa modelom. Spajanje se radi alatom Bind Skin, te se označavaju željene kosti i dijelovi tijela modela na koje se želi primijeniti. Tako se kosti noge spoje sa modelom lijeve i desne noge, te kost glave sa glavom, obrvama i očima, a glavna kost (eng. root) se povezuje sa cijelim tijelom. Kod kompleksnijeg spajanja kostura se spajaju kosti sa pravcima, te preko njih se pomiču i rotiraju željene kosti. Kod kosturiranja čovjeka koristi se Human IK, na kojemu se podešavaju broj kostiju koji se želi imati, broj prstiju, veličina kostura itd.



Slika 6.13. *Bind Skin i dodatne opcije za podešavanje*

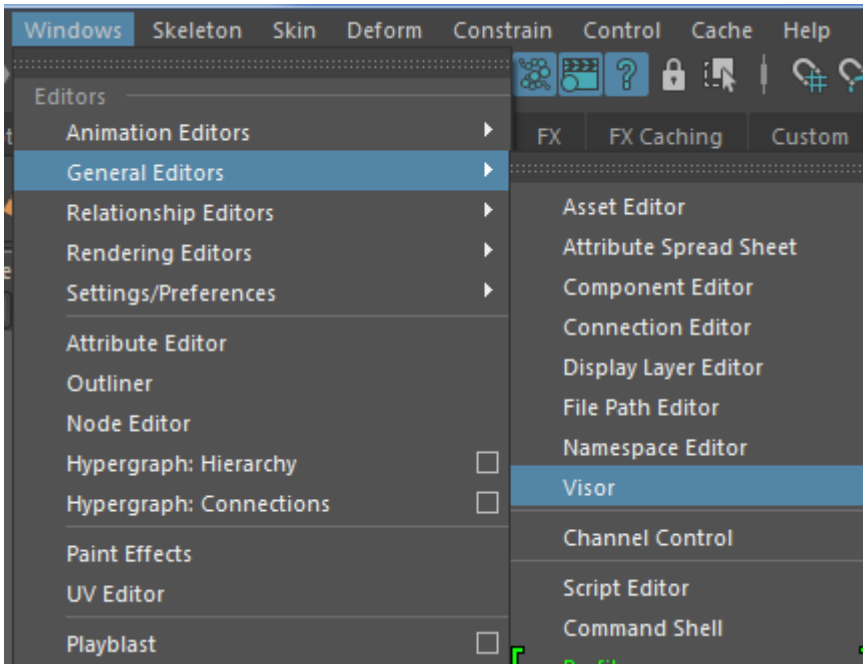
Animacija se izvodi na principu ključnih okvira (eng. keyframe), na kojem se broj postavlja koliko se želi (1 sek = 24 framea), te se oni zaključavaju na vremenskoj traci. Tako se na ovom modelu primijenjuju animacije i pokreti nogu prema naprijed – nazad i mali pokreti glave u stranu. Označi se željeni dio kosti koji se želi pomicati i postavi se na kojoj će poziciji biti u određenom okviru. Tako će se izvoditi dvije animacije na ovom modelu. Prva animacija će biti simulacija hoda, dok će druga biti simulacija pada (prekretanja na leđa). Kad se završi animacija potreban je izlazni format koji će se koristiti, a to je fbx. file, te će biti napravljen svaki zasebno (hodanje i pad).



Slika 6.14. i 6.15. *Dio tj. jedan ključni okvir iz animacije „pada“ i animacije „hodanja“*

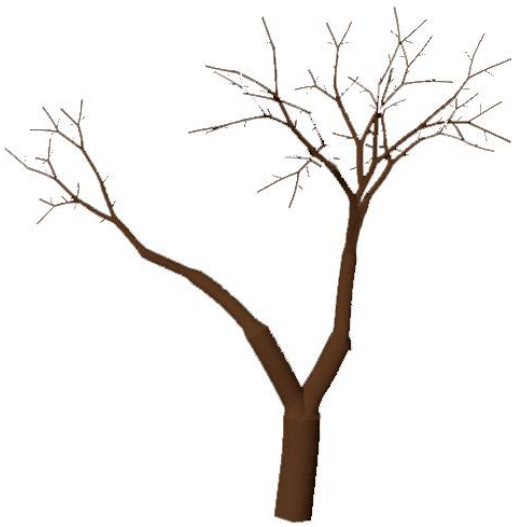
6.3. Drvo

Drvo se izrađuje pomoću Visor Editora. Vizor sadrži teksture, slike i sjenčanje, eksplozije, fluide, kose i ostale primjere i gotove oblike.



Slika 6.16. *Postupak otvaranja Visor Editora : Windows -> General Editors -> Visor*

Kada se odabere odabrani objekt, u ovom slučaju drvo automatski se odabire naredba Paint Effect tool preko koje se dobiva određeni broj objekata koji se žele prikazati. Kada se završi „crtanje“ slijedi pretvorba u poligone. To se dobije naredbom Modify -> Convert -> Paint Effects to Polygons.



Slika 6.17. *Konačan izgled „drveta“ nakon renderiranja*

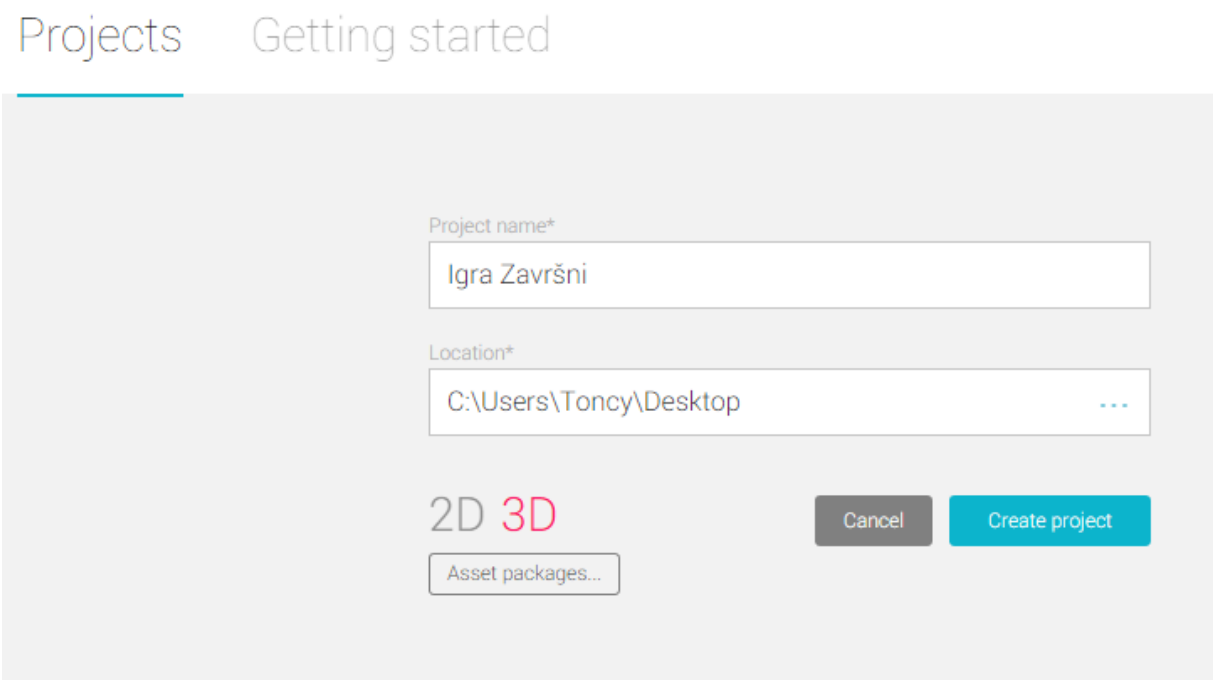
6.4. Ostali modeli

Za izradu modela koriste se osnovni alati Polygon Sphere, Polygon Plane-a, Polygon Cylinder. Daljne doradivanje modela je rađeno pomoću alata Extrude. Modeliranje je rađeno pomoću istih alata kao i u prethodna dva modela. Teksture koje se koriste su Lambert, Phong i Blinn. Transparentnost se koristi na dijelove modela kao što su prozor na kući, Checkpoint, glass itd. Slike na nekim određenim modelima koje se koriste su rađene u programu Adobe Photoshop CS4.

Slika 6.18. i 6.19. *Ostali modeli izrađeni u progrmu Autodesk Maya 2016.*

7. Unity – praktični dio

Pri prvom pokretanju aplikacije potrebno je stvoriti projekt. Unity nudi mogućnost stvaranja 2D i 3D projekta, no u ovom radu odabran je 3D projekt. Također postoje dodatne opcije koje se mogu ubaciti, tj. paketi dodataka (eng. Asset Packages). Tako se mogu odabrati dodatci poput kamere, karaktera, efekata, okoline, čestica, vozila itd. Kako bi se izradila aplikacija projekt mora imati najmanje jednu ili više scena koja se kasnije pretvara u aplikaciju. Svaka scena služe za izgradnju levela (razina), glavnog izbornika ili bilo kojeg drugog dijela (elementa) te aplikacije. Ova aplikacija će sadržavati četiri scene od koje su dvije scene level 1 i 2, a druge dvije početni i završni izbornik.

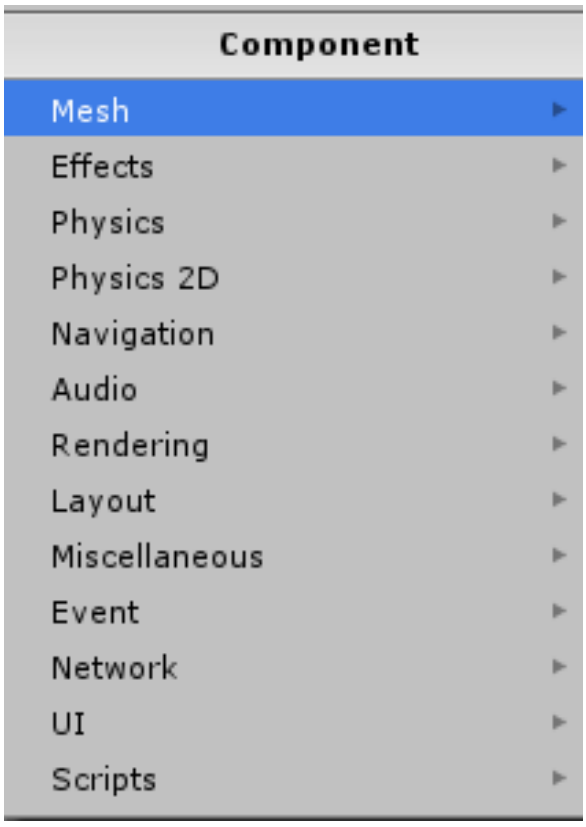


Slika 7.1. *Kreiranje projekta u Unity 5*

Kada se kreira projekt dolazi se do korisničkog sučelja od Unitya koji se sastoji od: glavnog izbornika, prozora scene (igre), prozora projekta (konzole), inspektora i hijerarhije. Nakon toga u prozoru projekta se kreiraju dvije nove mape u koje će se ubaciti .fbx modeli i teksture napravljeni u Mayi. Svi ti modeli se dodaju kao dodatak, korištenjem opcije ubacivanja novog dodatka (eng. Import New Asset) te se stavljaju u mapu.

7.1. Objekti, komponente i izgled levela

Objekti koji se nalaze u Unity programu su: prazni objekti, 3D objekti, 2D objekti, svjetlo, zvuk, korisničko sučelje, sustav čestica i kamera. Što se tiče podjele 2D i 3D objekata oni se dijele na osnovne oblike: kocka, kapsula, kugla. Objekti koji se koriste u ovoj aplikaciji (igri) su: 3D objekt sfera, svijetla, čestice, korisničko sučelje, zvuk, svjetlo i prazni objekti, a svi ostali objekti baziraju se na navedenim dodacima (modelima) iz Maye. Izgled igre i način na koji se igra ova igra bazirana je na platformama. Glavni objekt igre je Player (kugla), kojom se skakanjem po platformama dolazi do kraja levela ili kraja igre u završni izbornik. Da bi objekt postao „živ“ na njega se dodaju odgovarajuće komponente kojima se želi prikazati čemu taj objekt služi u toj aplikaciji. Svaki objekt ima osnovnu komponentu transformacija (eng. Transform) koja govori gdje se objekt nalazi u prostoru na kojoj poziciji, rotaciji, te njegova veličina (x,y,z).



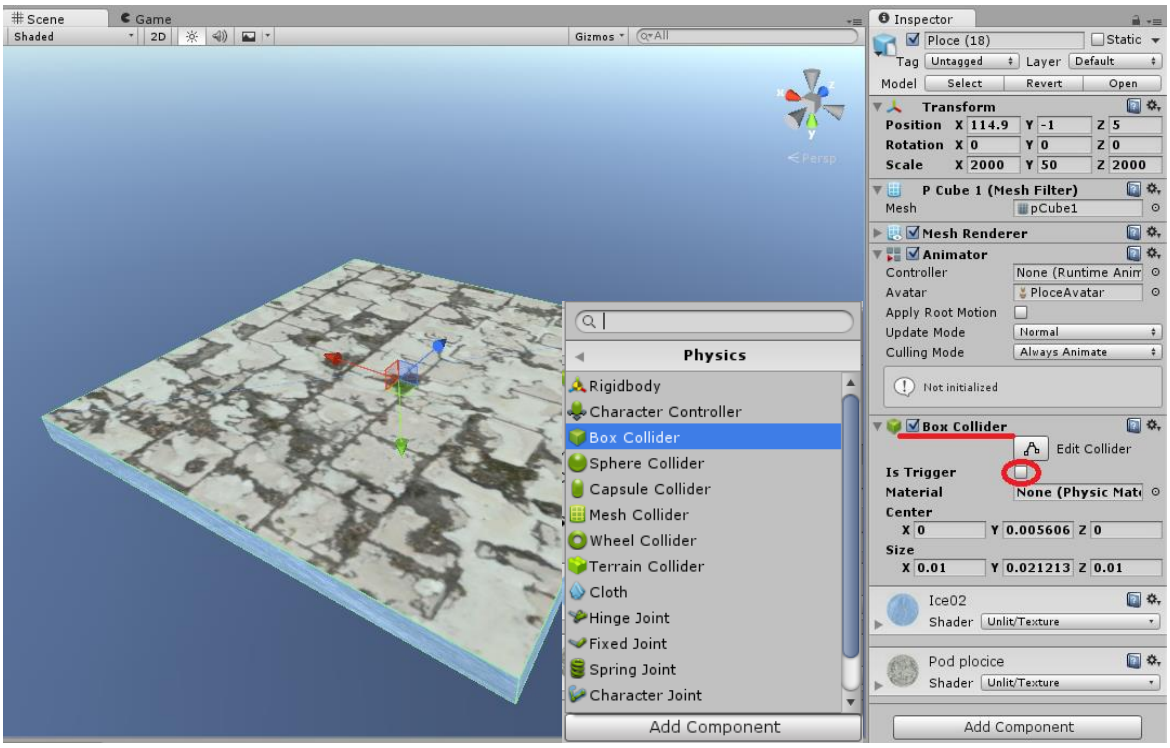
Slika 7.2. *Osnovana podjela komponenti*

Tablica 7.3. *Detaljne informacije o komponentama*

1.	**Geometrija** (eng. Mesh) sastoji se od dijelova koji služe za renderiranje, dodavanje uređivanje teksta i sl. Tako se opcijom za renderiranje (eng. Mesh Renderer) objektu mogu dodavati i micati sjene, mijenjati materijali, dodavati odsjaj prilikom renderiranja i dr.
2.	**Efekti** (eng. Effects) sadrži razne efekte kao što su sustavi čestica, halo svijetla (aureole) koja se koriste da bi se dobio dojam čestica prašine, simulacije efekta svjetla unutar objektiva kamere, efekti za iscrtavanje pri pokretu itd.
3.	**Fizika i 2D Fizika** (eng. Physics 2D) najčešće korištene komponente. Rigidbody je glavna komponenta koja omogućuje fizičke postavke za objekt kao što je gravitacija, rotacija, veličina i druga fizička svojstva objekta. Collideri tj. komponente koje definiraju oblik objekta te se koriste za razne oblike kao što su kutije, kugle, kapsule, kotača itd. Tu su također i opcije zglobnih pomicanja (eng. Joints) koji omogućuju rotaciju oko određene točke, rastezanje između više objekata, lomljenje ako određena sila prelazi postavljeni prag. Za 2D fiziku vrijedi slično kao i za 3D samo što je predviđeno za 2D prostor.
4.	**Navigacija** (eng. Navigation) navigacija omogućuje objektima da se kreću tako da pronađu put od jednog mjesta na drugo, izbjegavaju drugi objekt dok se kreću prema cilju, skakanje preko određenih objekata i dr.
5.	**Zvuk** (eng. Audio) sastoji se od raznih opcija za reproduciranje, kontroliranje, raspoređivanje zvuka. Tako se svakom objektu mogu dodati određeni zvukovi, zvučni efekti, pozadinska glazba i dr.
6.	**Renderiranje** (eng Rendering) sadrži razne elemente za korisničko sučelje, generiranja u igri, rasvjetu, specijalne efekte i dr. Postoji kubno mapiranje (eng. cubemaps) koje sadrži šest kvadrata koji čine imaginarnu kocku, pa se tim može dobiti realan odsjaj od nekog objekta. Nebo (eng. Skybox) se također nalazi u ovoj opciji, a služi kao simulacija neba.
7.	**Layouts** (eng. Layouts) koristi se za moderna korisnička sučelja, te omogućuje postavljanje elemenata u horizontalne, vertikalne grupe ili rešetke. Tako se može grupirati više dugmadi u jednu cjelinu (horizontalno, vertikalno).
8.	**Mješoviti** (eng. Miscellaneous) sadrži mješovite komponente kao što su tren, vjetar, animator, animacija i dr.
9.	**Događaj** (eng. Event) sastoji se od nekoliko dijelova koji rade zajedno da mogu slati događaje odnosno klik miša, tipkovnice, dodir i dr.

10.	**Mreža** (eng. Network) sadrži sinkronizaciju animacije preko mreže, skripte koje izvode razne naredbe i funkcije, upravljanje mreža s poslužiteljem, slanje i primanje poruka u mreži, povezivanja mreže sa klijentima itd.
11.	**Korisničko sučelje** (eng. UI) izrada korisničkog sučelja, dodavanje teksta, trake za pojačavanje, slike, maske, gumba, slidera itd. Također mogu se dodati efekti kao što su sjene, pozicioniranje slova, vanjske unutarnje linije i sl.
12.	**Skripte** (eng. Scripts) su jedne od najbitnijih dijelova za Unity. Preko njih objektima dajemo željene komande i funkcionalnost što i kako želimo da taj objekt sudjeluje u sceni. Tako će se glavnom igraču dati mogućnost pokreta, skakanja, uništavanja.

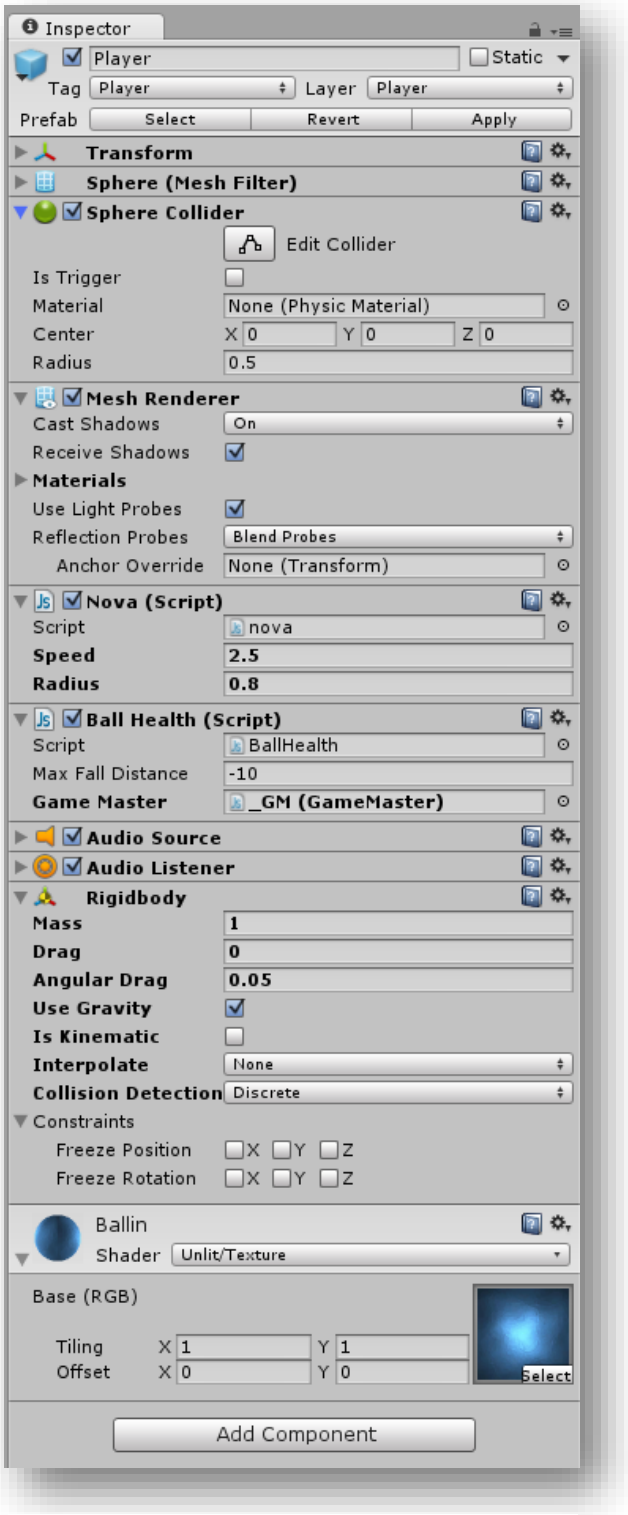
Na platforme se dodaje komponenta Box Collider kojim se postiže da kugla ne propada kroz nju, te da se može pomicati po njoj. Dodaje se tako da se u izborniku Inspector pritisne Add Component -> Physics -> Box Collider. Širina i visina će se postaviti da bude jednaka dimenziji objekta naredbom na Edit Collider. Postavljanjem atributa Is Trigger dozvoljava prolaz kroz Box Collider što se u ovom slučaju ne želi, te se ta oznaka mora maknuti. Uz platforme na isti način dodaje se komponenta i na druge objekte kao što su ograda, kuće, cijevi i ostala okolina po kojoj se je potrebno kretati.



Slika 7.4. *Dodavanje Box Collidera objektu*

7.2. Glavni igrač (player)

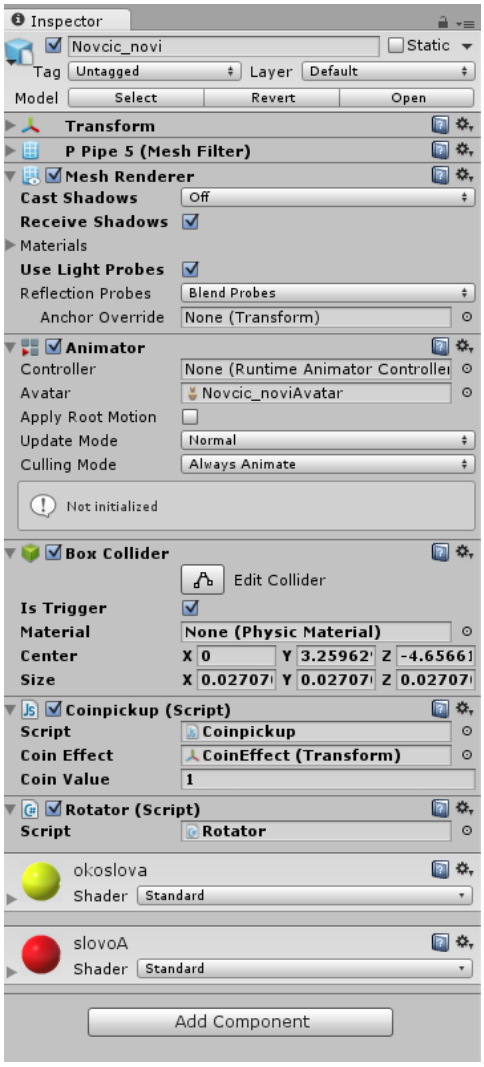
Glavni objekt (player) napravljen je od kugle (eng Spjere) osnovne 3D komponente u Unityu. Kugla se treba kretati u smjerovima gore – dolje, lijevo – desno, skakanje u vis. Da bi se to dobilo treba se napraviti skripta koja će omogućiti pomicanje odgovarajućih komandi. Skripta nudi mogućnost brzine kugle, postavljanje mase u odnosu na radius, ponašanje lopte na zemlji, njezino ubrzanje rotacija, pomicanje kugle ako pritišćemo tipku i ako ne pritišćemo. Uz to su dodane i dodatne fizike koje utječu na pomicanje pri pritisku tipke, njezinom skoku. Dodane su i dodatne opcije kamere te njezino ponašanje uz kuglu, brzina slijeda kamere kugle i njezino pomicanje. Druga skripta koja je dodana služi pri padu kugle sa platforme. Kako objekt kugla padne sa platforme ispod određene visine tako se ona vraća na početak levela odnosno na checkpoint. Pri padu kugle sa platforme dodana je audio komponenta kojom se pali zvuk nakon pada glavnog igrača ispod određene razine. Textura na loptu je dodana opcijom na Unlit/Texture, te je primijenjena slika Ballin. Također je bitno da je dodana fizika Sphere Collider jer bez nje kugla se ne bi mogla pomicati po platformi, skupljati objekte (novčiće) i uništavati protivnike.



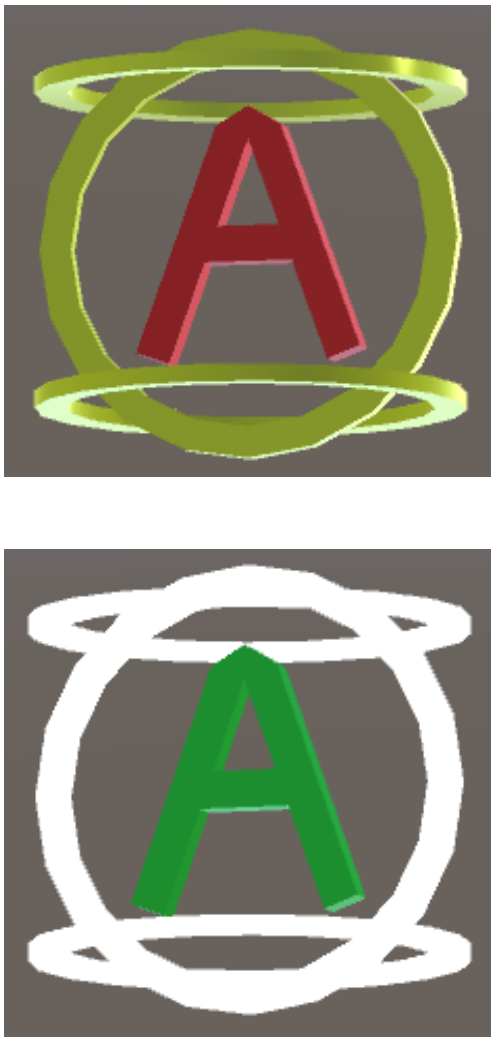
Slika 7.4. *Glavni objekt (Player) i sve njegove komponente*

7.3. Novčić

Objekti koji su zamišljeni da ih glavni igrač skuplja, te se skupljanjem svakog od njih dobiva po jedan ili dva boda. Vrijednost novčića se podešava u skripti, te se tako zadaje da žuto -crveni novčić vrijedi 1 bod, a bijelo – zeleni 2 boda. Također u skripti je dodano da se taj objekt može zbrajati, tj. dodati konačnom rezultatu. Pri dodiru i kontaktu glavnog igrača sa novčićem objekt se uništi i nestaje. Kako bi se to postiglo potrebno je staviti na njega komponentu „Box Collider“ na kojemu je potrebno oznaku „Is Trigger“ aktivirati, te se time dobije da kroz njega glavni objekt može proći. Druga skripta koja je dodana je rotacija, kojom se objekt rotira na mjestu u određenom smjeru kako je u skripti zadano. Pri uništenju je dodana komponenta sustava čestica (emg. Particles) radi boljeg vizualnog izgleda kod sakupljanja objekta.



Slika 7.5. *Komponente objekta „Novčić“*



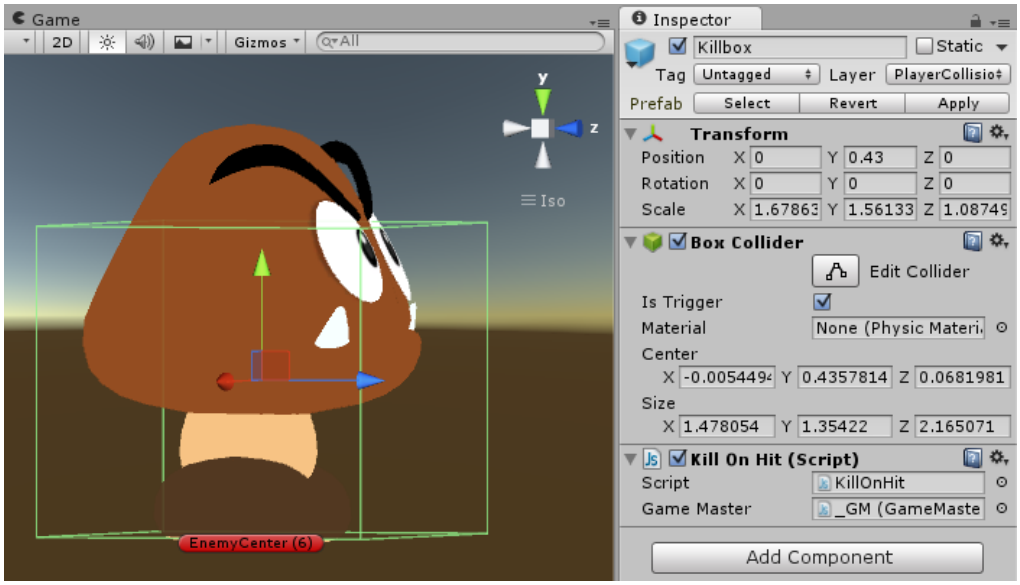
Slika 7.6. i 7.7. *Izgled objekta „Novčić“ vrijednosi 1 i 2 boda*

7.4. Neprijatelj i Snjegović

Kako je poznato mnoštvo igara na platformama sadrže i negativnu stranu, odnosno objekte koji eliminiraju glavnog igrača ili on može eliminirati njih. Ova aplikacija sadrži takva dva objekta iz Maye „Neprijatelj“ („Enemy“) i „Snjegović“. „Enemy“ je kompleksniji objekt koji sadrži više animacija i mogućnosti ponovljenog skoka igrača na njega nakon njegovog uništenja. Glavna skripta sadrži visinu odskoka igrača pri uništenju objekta i sustav čestica koji se prikaže nakon uništenja objekta. Ista skripta se primjenjuje na oba objekta, samo drugog naziva.

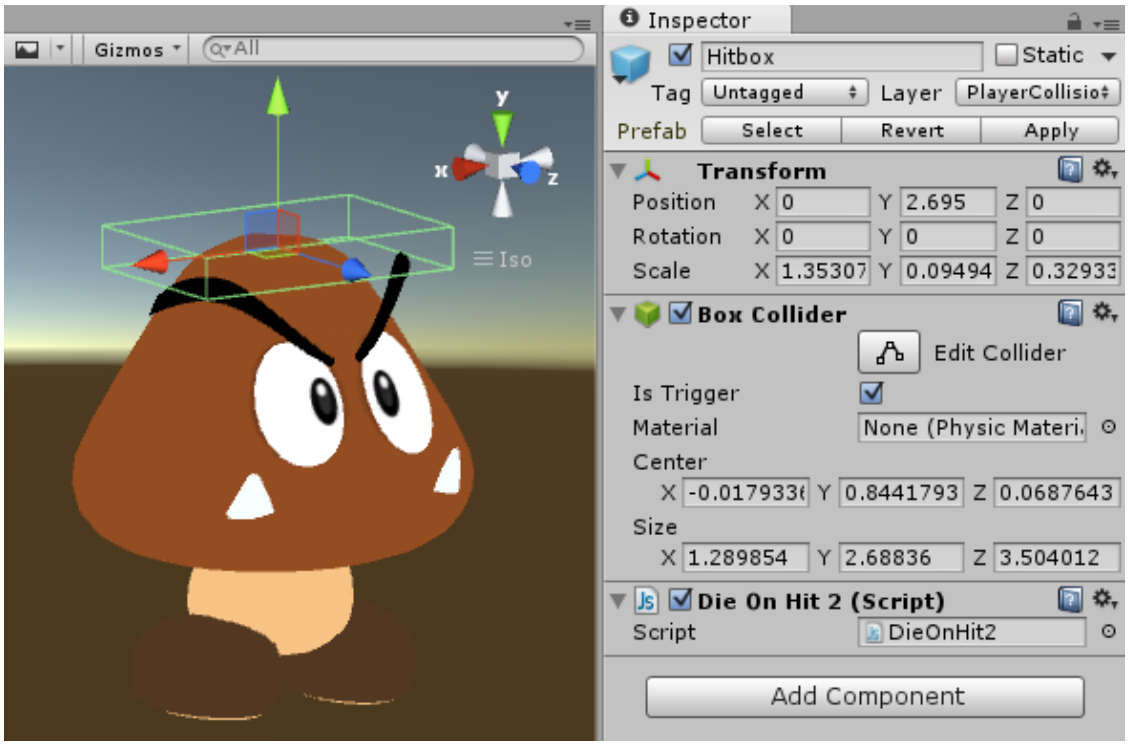
Slika 7.8. *Dijelovi skripte „Enemy“*

Pri dodiru glavnog igrača sa jednim od objekata („Enemy“ i „Snjegović) vraćaju glavnog igrača na početak levela ili kontrolnu točku koju je igrač dostigao (eng. checkpoint). Kako bi se to ostvarilo dodana je skripta za uništavanja glavnog igrača. Dodaje se prazni objekt pod nazivom „Killbox“, te se na njega primjenjuje Box Collider, a pri njegovom dodiru se odvija funkcija restartiranja igrača na početak ili kontrolnu točku levela.



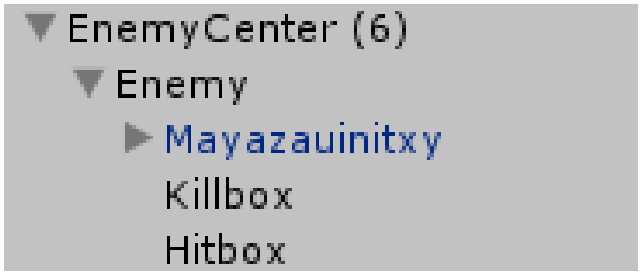
Slika 7.9. *Izgled kutije „Killbox“ na objektu „Enemy“ i njezine komponente*

Drugi prazni objekt koji se dodaje pod imenom „Hitbox“ služi za uništavanje objekta glavnog igrača. Komponente objekta su iste kao i na prethodnom promijenjene skripte, te dodirom tog objekta glavnim igračem se uništi. Pozicija Box Collidera je postavljena na vrh objekta, da ga igrač može dotaknuti samo ako skoči na njega.



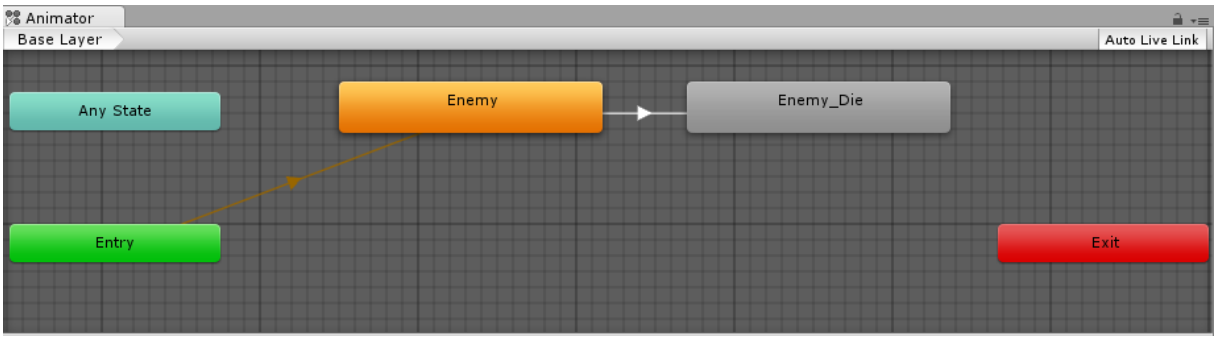
Slika 7.10. *Izgled kutije „Hitbox“ na objektu „Enemy“ i njezine komponente*

Kako bi svi objekti bili dio tog modela potrebno ga je grupirati. Tako njegova glavna hierarhija počinje centrom (“EnemyCenter”) kojeg nazivamo roditeljom (eng. Parent), a u njega se ubacuju ostali objekti koji su njegov podskup. Takva hierarhija omogućuje posebno upravljanje svakog dijela objekata kojeg se želi kontrolirati (animirati, pomaknuti, rotirati itd.).



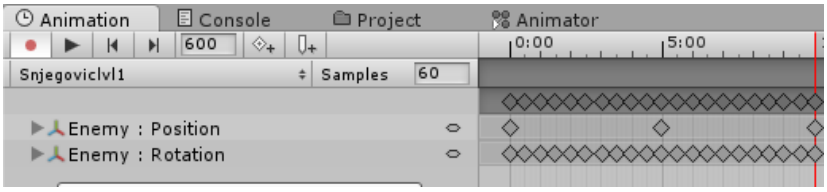
Slika 7.11. *Hierarhija za objekt „Enemy“*

Animacije koje se koriste za objekt "Enemy" su gotove animacije rađene u Mayi i animacije rađene u Unityu. Animacije iz Maye ubacuju se kao dodatak u projekt (eng. Import). Tako je osnovna animacija pokreta po platformi rađena u Unityu, a animacija hoda i pada objekta u Mayi. Kako bi animacije radile naredbe koje se želi zadati, potrebno ih je podesiti u animatoru (eng. Animator). Dodavanjem iskaza (create state) i primjenom animacija na njih se dobivaju željene akcije na objekt (animacija hodanja i animacija prekretanja objekta na leđa prilikom njegovog uništenja).

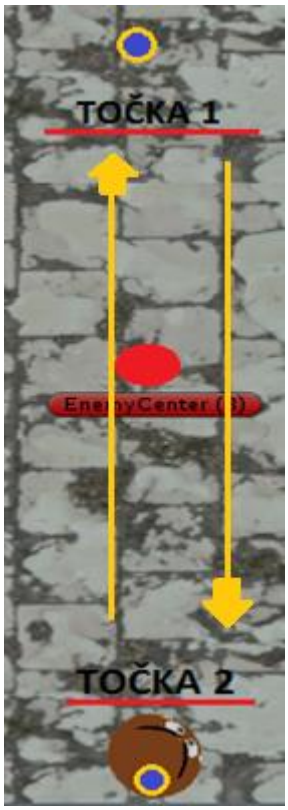


Slika 7.12. *Animator za objekt „Enemy"*

Animacija koja je napravljena u programu Unity je jednostavna animacija, koja služi za jednostavno pomicanje objekta po pravcu. U prozoru projeka Animacija (eng Animation) podešavaju se parametri po X, Y, i Z osi kojom se određuju pozicije, povećanja i rotacije objekta. Brzina rotacije pokreta u određenoj jedinici vremena i detaljniji pokreti se podešavaju u opciji krivine (eng. curves). Objekt kreće iz centralne točke prema točki 1 i 2 na kojima se rotira. Tako se animacija ponavlja sve dok se ne uništi taj objekt, pa u tom trenutku objekt staje na poziciji u kojoj je uništen. Kada je objekt uništen on ostaje na platformi, te glavni igrač ima mogućnost ponovnog skakanja na njega.



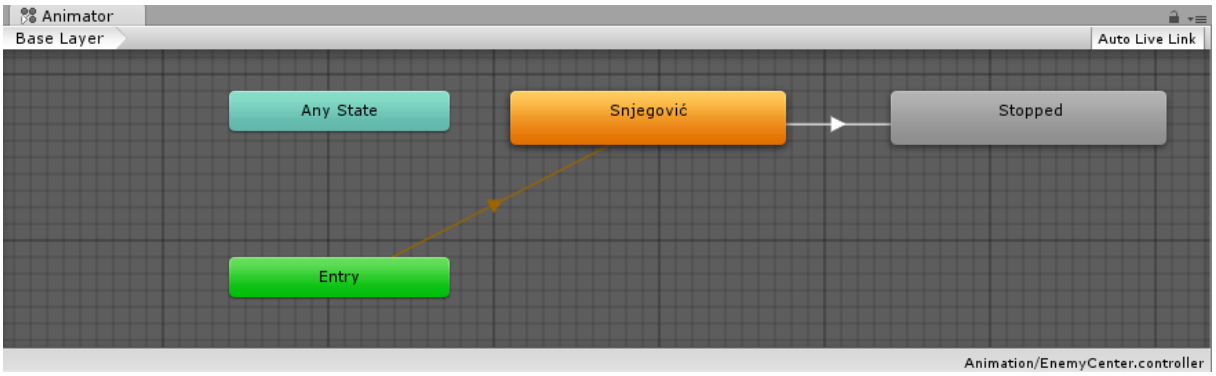
Slika 7.13. *Prozor animacije za objekt „Enemy"*



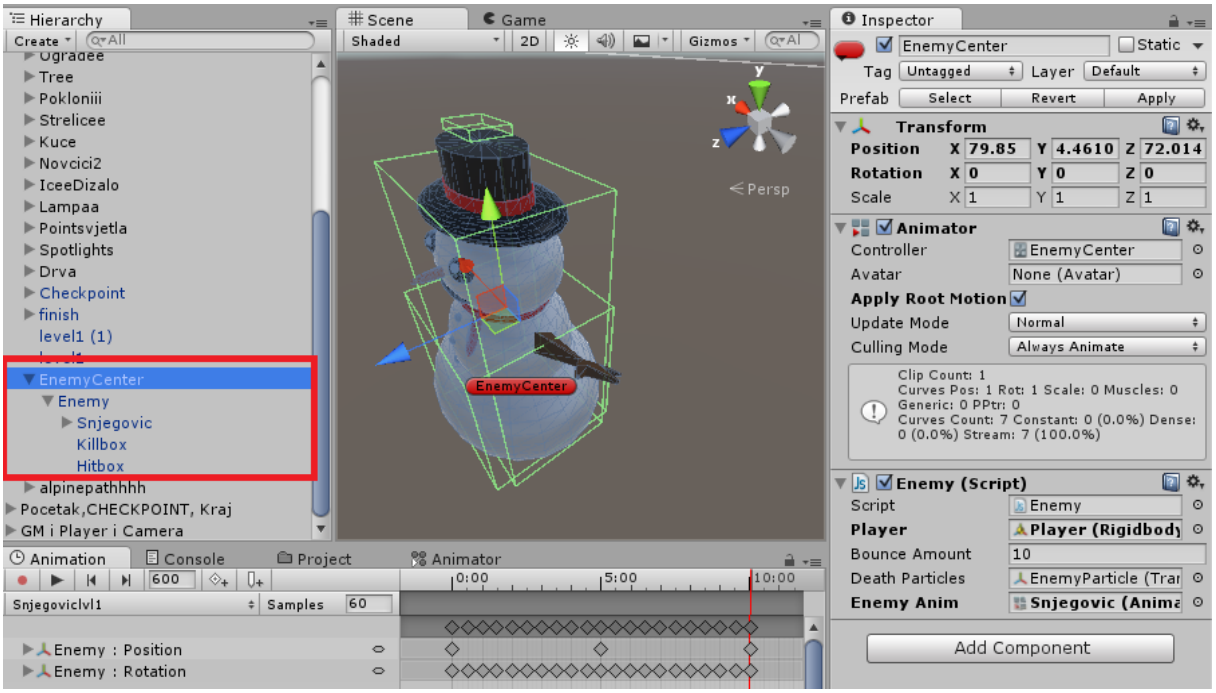
Slika 7.14. *Zamišljeno kretanje objekta „Enemy"*

Objekt “Snjegović” koristi samo animacije napravljene u Unityu (objekt neće imati animacije hoda i padanja na leđa). Pokreti i rotacije su napravljene isto kao i kod objekta “Enemy”, no pri uništenju tog objekta on nestaje. Tako će se kretati isto iz centra prema točki 1 i točki 2 sve dok ga se ne uništi.

Oba objekta koriste istu skripta samo što je za objekt “Enemy” u skripti uklonjeno njegovo uništenje i dodane su dvije aktivacije (eng. Trigger) za animaciju. Tako se koriste dvije skripte jedna za “Snjegovića”, a druga za “Enemy”.



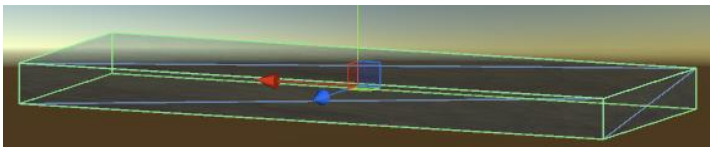
Slika 7.15. *„Animator za objekt „Snjegović“*



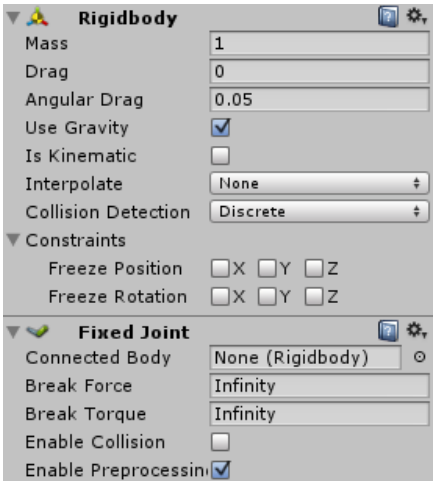
Slika 7.16. *Hierarhija, animacija, kutije „Hitbox“ i „Killbox“ i komponente objekta „Snjegović“*

7.5. Ostali pomični objekti

Elastična opruga napravljena je od dvije vrste objekta. Jedan objekt služi kao učvršćeni (nepomični) dio, a drugi objekt je pomični objekt, tj. elastična opruga koja se rasteže i prčvršćena je za nepomični objekt. Glavna komponenta koja je potrebna da se odredi nepomični objekt je fiksni spoj (eng. Fixed Joint). Komponenta se dodaje kao fizika, koja ima mogućnost podešavanja sile loma i obrtne sile. Također se dodaje komponenta eng. Rigidbody kojom se postavlja masa toga tijela.

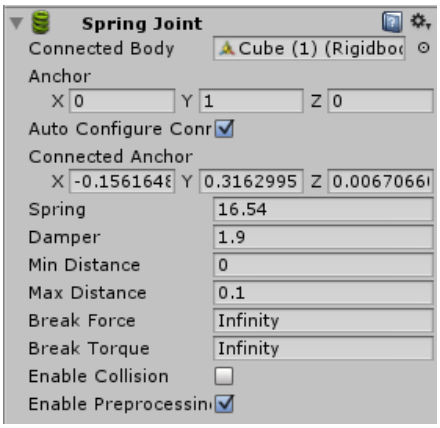


Slika 7.17. *Nepomičnio dio „Elastične opruge“ (Fixed Joint)*

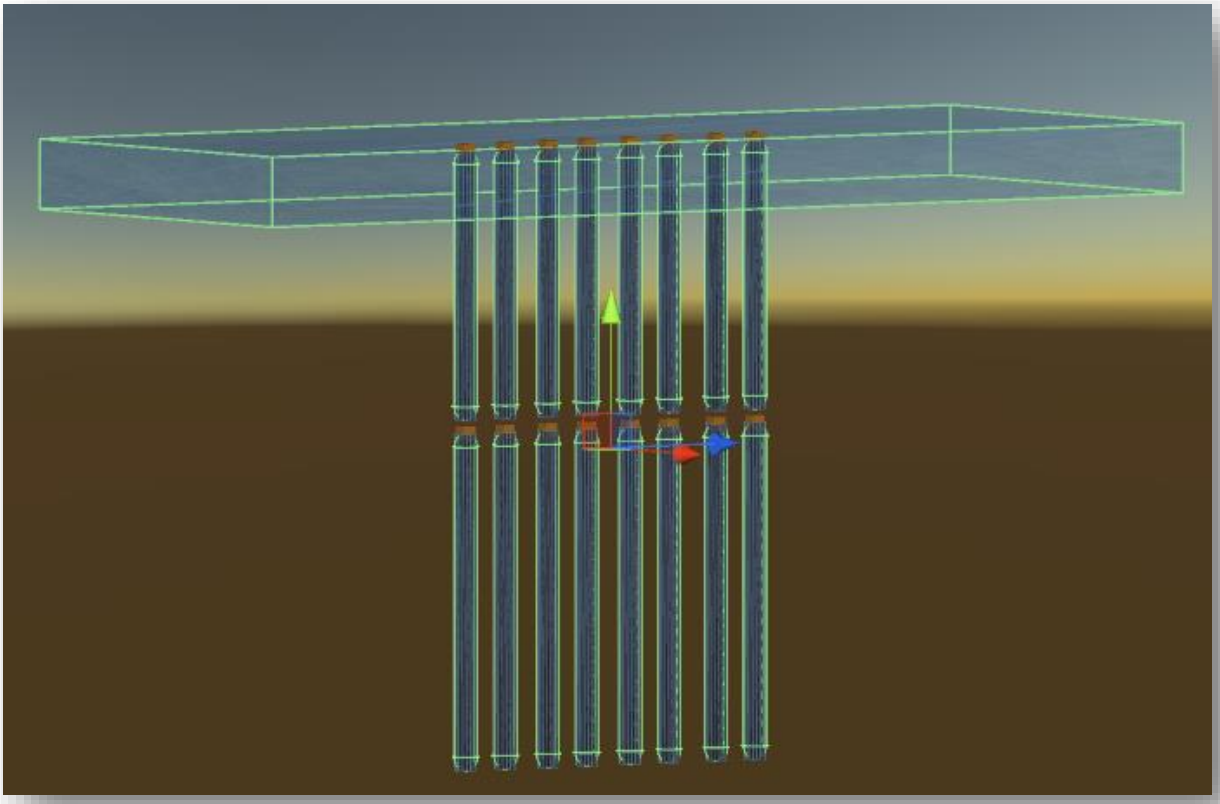


Slika 7.18. *Komponente nepomičnog dijela*

Drugi je objekt u obliku cilindra, na kojeg se postavi kapsula oblik (eng. Capsule Collider). Objekt koji se rasteže dobiva se komponentom spoj opruge (eng. Spring Joint). Tom komponentom se podešava minimalna i maksimalna duljina rastezanja, gibanje (rastezljivost) i brzina rastezanja. Da bi objekt bio povezan potrebno ga je spojiti sa nepomičnim objektom “Cube” postavkom “Connected Body”. Također se dodaje fizika eng. Rigidbody, na kojoj se podešava masa objekta i gravitacija. Pri dodiru glavnog igrača s elastičnom oprugom, ona se elastično pomiče u različitim smjerovima kako ju glavni igrač pomiče.

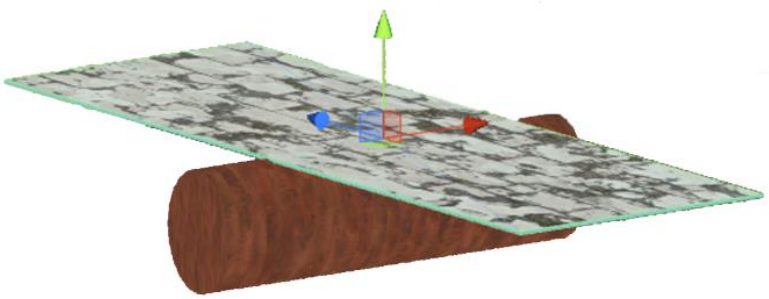


Slika 7.19. *Komponente elastičnog dijela*

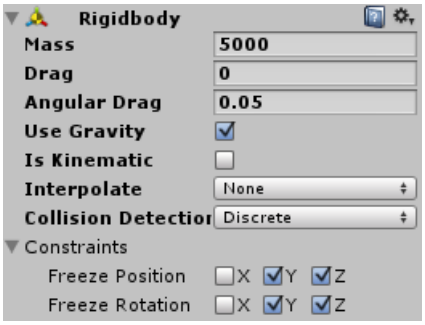


Slika 7.20. *Konačan oblik „Elastična opruga“*

Klackalica je model koji se sastoji od 2 oblika. Jedan je ravna ploča, a drugi valjak na kojemu stoji ravna ploča. Na ploču je primijenjena fizika Rigidbody na kojoj je podešena gravitacija i masa. Kako bi se ploča kretala samo u jednom smjeru (osi) zaključavaju se (eng. Freeze) pozicije osi Y i Z, dok X os ostaje otključana i ploča se može pomicati samo po osi X. Na valjak i ploču se primjenjuje Box Collider, kako bi objekt mogao biti jedan na drugome.

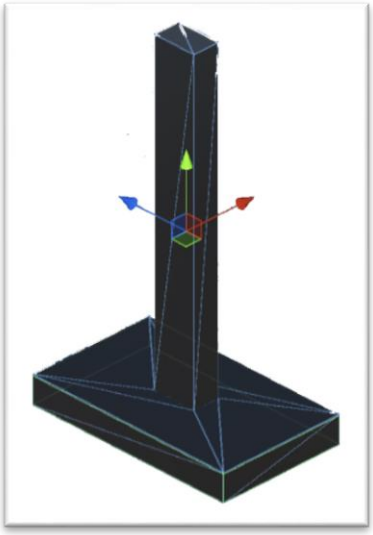


Slika 7.21. *Objekt „Klackalica“*



Slika 7.22. *Komponenta Rigidbody objekta „Ploča“*

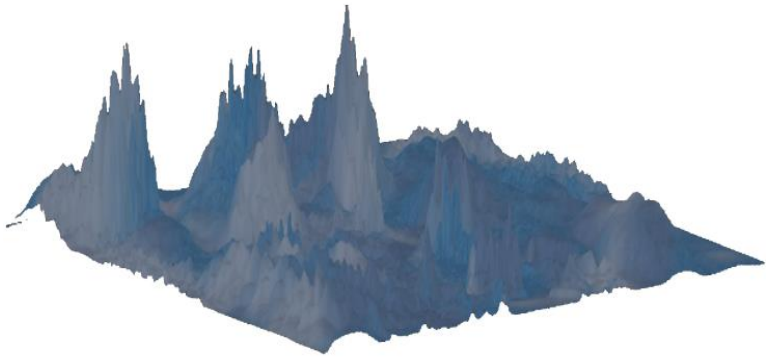
Čekić je pomični objekt koji pri dodirom s glavnim igračem ga uništava. Na objekt je primijenjena animacija napravljena u Unity, te se objekt pomiče gore dolje u određenom vremenu. Animira se samo pozicija objekta u određenom vremenu (X, Y, Z). Na objekt je primijenjena skripta "KillOnHit" koja se koristi na modelima "Enemy i "Snjegović". Skripta dakle pri doticaju "Čekića" sa glavnim igračem uništava glavnog igrača.



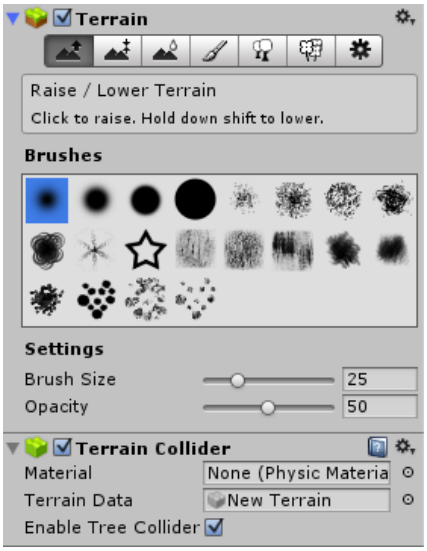
Slika 7.23. *Objekt „Čekić"*

7.6. Teren i Nebo

Teren (eng. Terrain) se dodaje kao 3D objekt. Kreiranjem objekta dobiva se ravna ploča u ravnini. Pritiskom na objekt u inspektoru dobivamo njezine komponente za podešavanje terena. Pritiskom određenog alata i odabirom željenog oblika pomiče se miš tim alatom po terenu i dobiva se željeni oblik, boja, veličina i sl. Teren se može povisiti – sniziti, texturirati, glatki oblici, visina boje, postavljanje drveća, postavljanje detalja (cvijeće, drveće) i detaljne postavke terena koje se sastoje od: glavnog terena, drva i detalji objekata, vjetar i rezolucija.



Slika 7.24. *Teren*



Slika 7.25. *Komponenta Terrain i njezini alati*

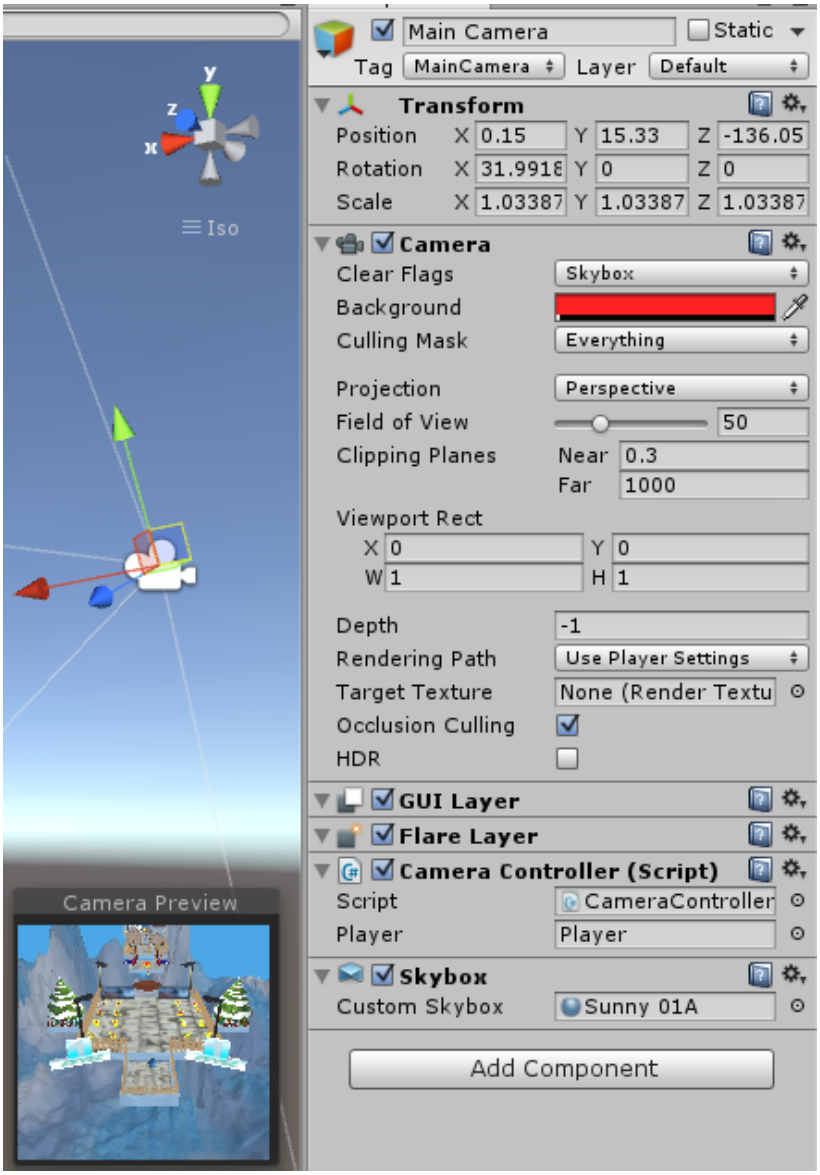
Nebo (eng. Skybox) prikazuje i simulira nebo u cijeloj sceni. Nebo nema svoj početak i kraj, te pomicanjem po sceni se ne može izaći iz njega. Kako bi se dodalo nebo odabire se u alatnoj traci Component -> Rendering -> Skybox. Nebo se primjenjuje kao komponenta na kameru. Unity sadrži početno nebo koje se može zamjeniti kako korisnik želi. Preko Unity trgovine dodataka preuzima se željeno nebo (Windows -> Asset Store). Nebo se sastoji od šest slika koje se postavljaju po određenoj osi (Front +Z, Back –Z, Left +X, Right –X, Up +Y, Down –Y), te tako tvore jednu određenu okolinu. Odabirom željenog neba korisnik podešava slike po svakoj osi.

Izvor dodatka iz Unity trgovine (Nebo):
https://www.assetstore.unity3d.com/#!/content/18353

Slika 7.26. *Komponenta za podešavanje osi slika Neba*

7.7. Kamera

Glavna kamera (eng. Main Camera) postavljena je tako da gleda glavnog igrača. Da bi pratila glavnog igrača kako se kreće potrebna je skripta koja povezuje glavnog igrača i kameru. Povezuju se dvije skripte glavna skripta i skripta za kontroliranje kamere. U skriptama se podešavaju brzina kamere, pomicanje igrača na osnovu kamere, praćenje kamere za glavnim igračem. Osnovna postavka kamere je vidno polje (eng. Field of View) kojom se podešava veličina vidnog polja igrača i okoline. Pomicanjem kamere u sceni u doljnjem desnom kutu se prikazuje pregled kamere u igračem modu, tj. što će kamera vidjeti i prikazati.



Slika 7.27. *Kamera i njezine komponente*

7.8. GUI – grafičko korisničko sučelje

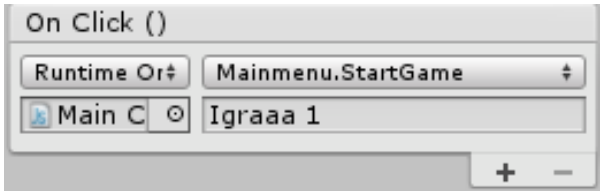
U aplikaciji se koriste dva korisnička sučelja početno i završno. Početno korisničko sučelje sadržavat će ulazak u igru, podešavanje glasnoće zvuka i izlaz iz igre. Izgled i pozadina početnog GUI –a koristi objekte, teren i animacije iz igre. Dugme (eng. Button) "Play" grafički prikazuje tipku "Play Game", te pritiskom na njega ulazi se u igru (level 1). Skripta "Mainmenu" omogućuje funkcije za pokretanje igre, izlazak i podešavanje jačine zvuka. Podešavanjem opcije "On Click" na kojemu se podešava funkcija Mainmenu.StartGame dobiva se željeni prelazak kamere iz početnog GUI –a u level 1. Dugme "Quit" koristi istu skriptu kao i dugme "Play" samo druge funkcije Mainmenu.QuitGame koja omogućuje izlazak iz igre. Funkcija Mainmenu.GameVolume omogućuje pojačavanje i smanjivanje zvuka koristeći klizač. Pojačavanje zvuka postavlja se u smjeru s lijeva na desno minimalne vrijednosti 0, te maksimalne vrijednosti 1. Kod klizača "Slider" podešavaju se parametri jačine zvuka zadane početne vrijednosti 0.5, te ulaskom u izbornik jačina zvuka je podešena na pola.

Slika 7.28. *Početni izbornik*

Završno korisničko sučelje “End” sastoji se od dugmeta za ponovno pokretanje igre i izlazak iz igre. Ponovno pokretanje igre koristi isto dugme kao i u početnom izborniku “Play”, kojom aplikacija ulazi u level 1. Tekst za konačni rezultat (eng. Final score) koristi skriptu za izračun, kojom se zbrajaju bodovi iz oba levela, te se prikazuje brojka skupljenih novčića.

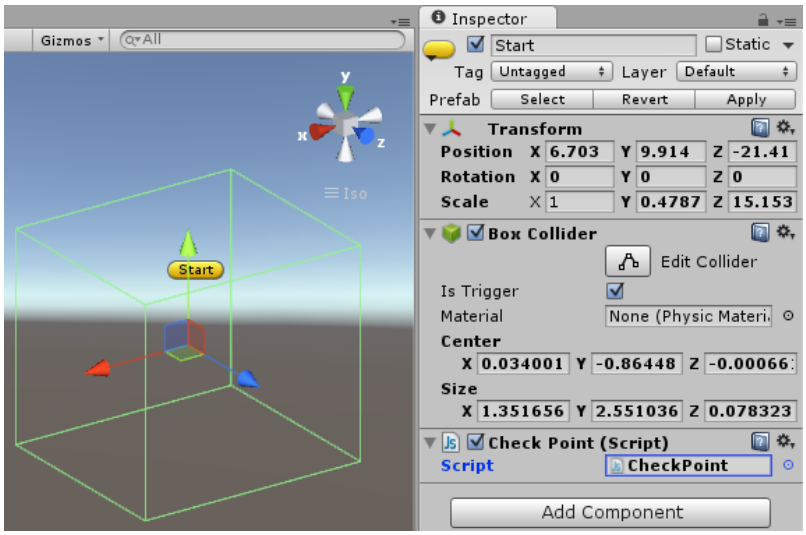
Slika 7.29. *Završni izbornik*

7.9. Kraj levela i kontrolna točka (checkpoint)

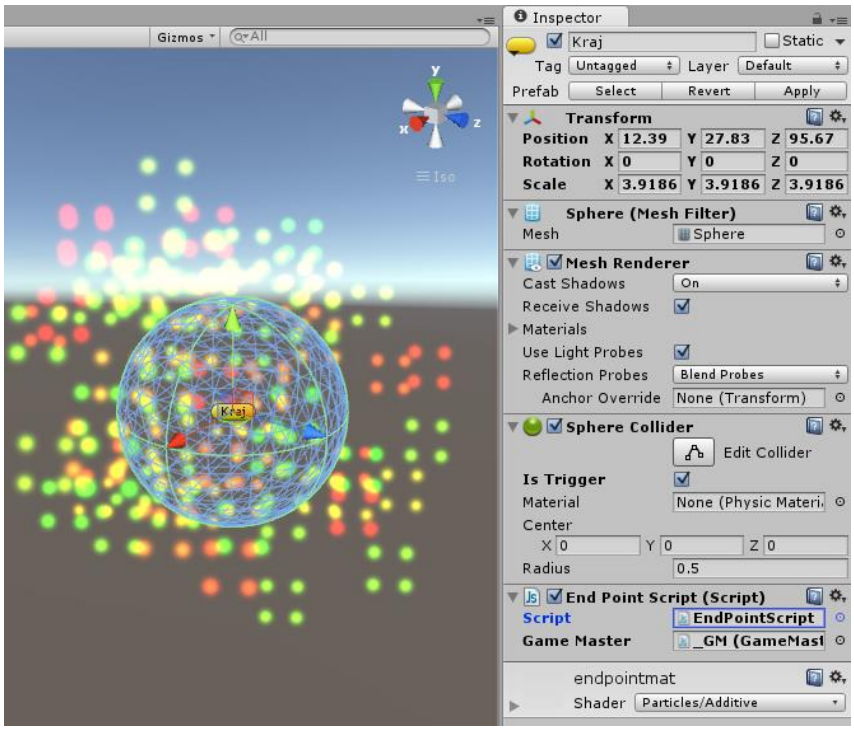


Slika 7.30. *Funkcija On Click za pokretanje igre „Play“*

Kontrolna točka (eng. Checkpoint) dobiva se dodavanjem skripte na prazni objekt. Kada glavni igrač dodirne kontrolnu točku prilikom pada sa platforme ili uništenja glavnog igrača vraća ga na zadnju kontrolnu točku koju je dodirnuo. Ista komponenta se primjenjuje za početak levela.



Slika 7.31. *Kontrolna točka „Checkpoint“ i njezine komponente*



Slika 7.32. *Završni objekt „Kraj“ i njezine komponente*

Kraj levela dobivamo skriptom za završetak levela dodanu na prazni objekt, kojom se glavnog igrača prilikom dodira prebacuje na iduću scenu (level). Korištene su dvije skripte gdje jedna

služi za učitavanje sljedećeg levela “EndPointScript”, a druga za zbrajanje trenutnog rezultata sakupljenih novčića, te povezivanje sa prvom skriptom “GameMaster”. Kako bi objekt bio vidljiv dodan je sustav čestica raznih boja.

7.10. Skripte

1) “Nova” - skripta glavnog igrača

```
// brzina lopte
 var speed = 5.0;
 var radius = 1.0;
 private var velocity : Vector3 = Vector3.zero;


 function Start(){

     transform.localScale = Vector3.one * radius * 2;
     var hit : RaycastHit;
     if(Physics.Linecast(transform.position, transform.position -
Vector3.up * 500, hit)){
         transform.position = hit.point + Vector3.up * radius;
     }
     // rigidbody ako nema
     if(!GetComponent.<Rigidbody>())
         gameObject.AddComponent(Rigidbody);
     // postavljanje mase na osnovu radiusa
     GetComponent.<Rigidbody>().mass = 600 * radius;
 }

 function FixedUpdate () {

     // ponašanje lopte na zemlji
     var hit : RaycastHit;
     var isGrounded = Physics.Raycast(transform.position, -
Vector3.up, hit, radius * 1.2);

     // osnovna brzina kamere.
     // resetiranje kamere X osi na 0 da uvijek gleda horizontalno
     var x = Camera.main.transform.localEulerAngles.x;
     Camera.main.transform.localEulerAngles.x = 0;

     // kretanje (now collect the movement stuff This is generic
direction.)
     var direction =
Vector3(Input.GetAxis("Horizontal"),0,Input.GetAxis("Vertical"));

     // spriječavanje bržeg pomicanje kugle dijagonalno
     if(direction.magnitude > 1.0) direction.Normalize();

     // skok
     if(isGrounded && Input.GetKeyDown(KeyCode.Space))
     GetComponent.<Rigidbody>().AddForce(Vector3.up *
GetComponent.<Rigidbody>().mass * 450);
```

```
     // fizika kugle na mjestu
     if(direction.magnitude > 0){
         // isGrounded u neku informaciju
         var modifier = isGrounded ? 3.0 : 0.5;
         // pomicanje na osnovu kamere
         direction =
Camera.main.transform.TransformDirection(direction) * speed * 2;
         // velocity
         direction.y = GetComponent.<Rigidbody>().velocity.y;


         // pomicanje kugle ako ništa ne pritišćemo
         GetComponent.<Rigidbody>().velocity =
Vector3.Lerp(GetComponent.<Rigidbody>().velocity, direction,
modifier * Time.deltaTime);
         // rotacija
         var rotation =
Vector3(GetComponent.<Rigidbody>().velocity.z,0,-
GetComponent.<Rigidbody>().velocity.x) * 20;


         // dodavanje vrtnje (ubrzanje) da lopta ide bolje
         GetComponent.<Rigidbody>().angularVelocity =
Vector3.Lerp(GetComponent.<Rigidbody>().angularVelocity, rotation,
modifier * Time.deltaTime);
     }


     // kamera x rotacija
     Camera.main.transform.localEulerAngles.x = x;
 }
```

2) "Mainmenu"

```
#pragma strict

var music : AudioSource;

function QuitGame (){
     Application.Quit ();
}

function StartGame (level : String){
     Application.LoadLevel (level);
}

function SetGameVolume (vol : float) {
     music.volume = vol;
}
```

3) "KillOnHit"

```
#pragma strict
```

```
var gameMaster : GameMaster;

function OnTriggerEnter () {
     if (GameMaster.isRestarting == false) {
     gameMaster.RestartLevel();
     }
}
```

4) “GameMaster”

```
#pragma strict

static var currentScore : int = 0;
static var isRestarting = false;
static var finalScore : int = 0;

var player : Transform;

var musicPrefab : Transform;

var GameOverSound : AudioClip;



function Start (){
currentScore = 0;

if    (!GameObject.FindGameObjectWithTag("mm")){
     var mManager = Instantiate (musicPrefab, transform.position,
Quaternion.identity);
     mManager.name = musicPrefab.name;
     DontDestroyOnLoad (mManager);
     }
}

function RestartLevel () {
          isRestarting = true;
          GetComponent.<AudioSource>().Play();
          yield WaitForSeconds
(GetComponent.<AudioSource>().clip.length);

          player.position = CheckPoint.ReachedPoint;
          isRestarting = false;
}

function LoadNextLevel()
{
     finalScore += currentScore;
     Application.LoadLevel (Application.loadedLevel + 1);
}
 function Update () {
     if(Input.GetKeyDown("escape")) {//Kada je pritisnuta tipka esc
izlazi iz igre
         Application.Quit(); // Quits the game
     }
 }
```

5) “FinalScoreGui”

```
#pragma strict

import UnityEngine.UI;

var scoreText : Text;

function Start ()
{
     scoreText.text = "FINAL SCORE: " + GameMaster.finalScore;
}
```

6) “Enemy”

```
#pragma strict

var player : Rigidbody;
var bounceAmount = 10f;

var deathParticles : Transform;

var enemyAnim : Animator;
private var centerAnim: Animator;

function Awake () {
     centerAnim = transform.GetComponent ("Animator") as Animator;
}
function Die ()
{
     player.GetComponent.<Rigidbody>().velocity.y = bounceAmount;
     Instantiate (deathParticles, enemyAnim.transform.position,
enemyAnim.transform.rotation);


     Destroy (gameObject);

}
```

7) “Enemy2”

```
#pragma strict

var player : Rigidbody;
var bounceAmount = 10f;

var deathParticles : Transform;

var enemyAnim : Animator;
private var centerAnim: Animator;

function Awake () {
     centerAnim = transform.GetComponent ("Animator") as Animator;
}
function Die ()
```

```
{
      player.GetComponent.<Rigidbody>().velocity.y = bounceAmount;
      Instantiate (deathParticles, enemyAnim.transform.position,
enemyAnim.transform.rotation);
      enemyAnim.SetTrigger ("Die");
      centerAnim.SetTrigger ("Stop");

      //Destroy (gameObject);

}
```

8) "EndPointScript"

```
#pragma strict

var gameMaster : GameMaster;

function OnTriggerEnter (colInfo: Collider)
{
      if (colInfo.tag == "Player")
      {
            gameMaster.LoadNextLevel();
      }
}
```

9) "DieOnHit"

```
#pragma strict

function OnTriggerEnter ()

{
      var enemy = transform.GetComponentInParent (Enemy);
      enemy.Die();
}
```

10) "DieOnHit2"

```
#pragma strict

function OnTriggerEnter ()

{
      var enemy = transform.GetComponentInParent (Enemy2);
      enemy.Die();
}
```

11) "CurrentScoreGUI"

```
#pragma strict

import UnityEngine.UI;

var scoreText : Text;

function Update ()
```

```
{
      scoreText.text = "COINS COLLECTED: " + GameMaster.currentScore;
}
```

12) "Coinpickup"

```
#pragma strict

var CoinEffect : Transform;
var coinValue = 1;

function OnTriggerEnter (info: Collider)
{
      if (info.tag == "Player")
      {
      GameMaster.currentScore += coinValue;
      var effect = Instantiate(CoinEffect, transform.position,
transform.rotation);
      Destroy(effect.gameObject, 3);
      Destroy(gameObject);
      }
}
```

13) "CheckPoint"

```
#pragma strict

static var ReachedPoint : Vector3;

function OnTriggerEnter (col : Collider)
{
 if (col. tag == "Player")
 {
           ReachedPoint = transform.position;
 }
}
```

14) "BallHealth"

```
#pragma strict

var maxFallDistance = -10;

var gameMaster : GameMaster;

function Update ()
{
      if (transform.position.y <= maxFallDistance)
      {
            if (GameMaster.isRestarting == false)
            {
            gameMaster.RestartLevel();
            }
      }
}
```

15) “2CoinPickup”

```
#pragma strict

var CoinEffect : Transform;
var coinValue = 2;

function OnTriggerEnter (info: Collider)
{
      if (info.tag == "Player")
      {
      GameMaster.currentScore += coinValue;
      var effect = Instantiate(CoinEffect, transform.position,
transform.rotation);
      Destroy(effect.gameObject, 3);
      Destroy(gameObject);
      }
}
```

16) “CameraController”

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour {

      public GameObject player;

      private Vector3 offset;

      void Start () {

            offset = transform.position - player.transform.position;

      }

      void LateUpdate () {
            transform.position = player.transform.position + offset;
      }

}
```

17) “Rotator”

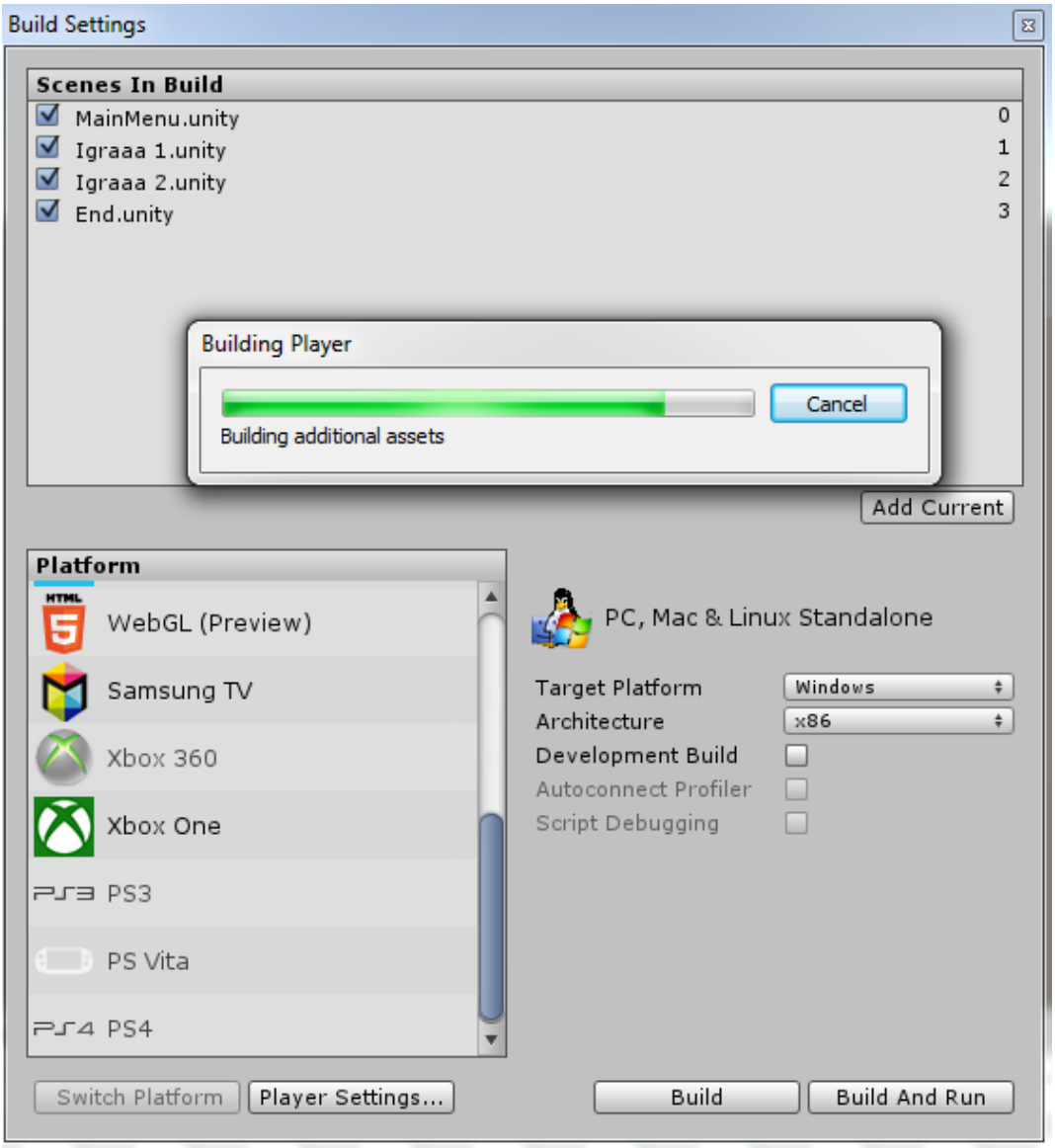
```
using UnityEngine;
using System.Collections;

public class Rotator : MonoBehaviour {

      void Update ()
      {
            transform.Rotate (new Vector3 (15, 30, 45) *
Time.deltaTime);
      }
}
```

7.11. Izrada aplikacije (igre)

Završni korak u izradi igre je pretvaranje projekta u aplikaciju. Odabirom File -> Build Settings pojavljuje se prozor u kojemu se dodaju trenutne scene projekta. Kad su sve scene dodane potrebno ih je poredati po željenom slijedu (1. Main Menu -> 2. Level 1 -> 3. Level 2 -> 4. Kraj). Odabirom platforme stvara se konačna aplikacija, kojom se dobiva nova datoteka, odnosno aplikacija (igra). Detaljne postavke nude izgled ikone za pokretanje igre odabirom željene slike, rezoluciju, pokretanje u cijelom zaslonu, optimizaciju itd. Za svaku platformu postoje određene postavke koje su potrebne radi bolje optimizacije igre.



Slika 7.33. *Postavke izgradnje aplikacije*

8. Zaključak

Rad je osmišljen i prikazan kroz dva programa za razvoj i izradu igre. Detaljno su opisani alati koji se koriste za izradu određenih objekata, njihov izgled i sučelje svakog programa. Cilj rada je upoznati svaki od korištenih programa, te njihovo spajanje u jednu cjelinu (gotovu aplikaciju). Kroz svaki od tih programa upoznaju se određeni alati i njihova uporaba u izradi određenih objekata.

Većina trodimenzionalnih modela je napravljena u programu Autodesk Maya. Modeli su rađeni poligonalnim modeliranjem, koje se može brzo naučiti i nije kompliciranо za jednostavne modele koji su korišteni u ovom radu. Alati se koriste ovisno o primjeni i području koje je korisniku potrebno. Postoje alati koji se koriste za usko područje primjene, ali i kompliciraniji koji se mogu koristiti gotovo za svaki proces modeliranja. Ovisno o korisnikovom znanju i kompleksnosti alata često se koriste lakši alati koji korisniku omogućavaju finalni sadržaj koji želi dobiti. Kroz Unity upoznaje se izrada korisničkog sučelja preko kojeg korisnik komunicira s igrom. Program je pristupačan i jednostavan za izradu, kao i sučelje, kako bi se smanjila kompleksnost programa. Ubacivanjem modela iz Autodesk Maya dobiva se konačna igra s dva levela i korisničkim sučeljem.

Danas svatko može napraviti igru na svojem osobnom računalu. Kako bi igra izgledala dobro i realno potrebno je mnogo vremena i znanja u različitim područjima. Spajanjem više različitih programa i izrada pojedinih dijelova aplikacije u određenom programu omogućuju detaljnu izradu i izgled. Stoga se može zaključiti kako s razvojem igara su se razvili i različiti programi kojima se izrađuju igre. Programi su zainteresirali velik broj ljudi što je vidljivo kroz povećanje broja igrica i igranje istih posljednjih nekoliko godina.

U Varaždinu _____ _____

Datum Potpis

9. Literatura

[1] Oldcomputers (2015.) Apple II – 1977

Dostupno na: http://www.oldcomputers.net/appleii.html, kolovoz 2015.

[2] Partha Sarathi Paul, Surajit Goon, Abhishek Bhattacharya: History and comparative study of modern game engines - evolution of game engines str. 246 – 247.

Dostupno na: http://bipublication.com/files/IJCMS-V3I2-2012-07.pdf, kolovoz 2015.

[3] Croteam (2014.) – Serious Engine 4

Dostupno na: http://www.croteam.com/technology/, kolovoz 2015.

[4] Krunoslav Čovran, Andrija Bernik, Damir Vusić (ožujak 2014): Autodesk Maya – Dinamična bolja u računalnoj animaciji, Tehnički Glasnik: Časopis Veleučilišta u Varaždinu. Varaždin, Hrvatska str. 102.

Dostupno na: http://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=177665, kolovoz 2015.

[5] Andrija Bernik (prosinac 2010.): Vrste i tehnike 3D modeliranja, Tehnički Glasnik: Časopis Veleučilišta u Varaždinu. Varaždin, Hrvatska str. 45.

Dostupno na: http://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=127863, kolovoz 2015.

[6] Andrija Bernik, Kelnarić D. (srpnj 2011.): Vrste i tehnike 3D teksturiranja, Tehnički Glasnik: Časopis Veleučilišta u Varaždinu. Varaždin, Hrvatska str. 30 - 33.

Dostupno na: http://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=127725, kolovoz 2015.

[7] Autodesk Knowledge Network (9. rujan 2014.): Blinn

Dostupno na: http://help.autodesk.com/view/MAYAUL/2015/ENU/?guid=Shading_Nodes_Blinn, kolovoz 2015.

[8] Sveučilište u Zagrebu / Fakultet organizacije i informatike (2012.): Računalna grafika – Projekt

Dostupno na: http://rg.c-hip.net/2012/seminari/kudeljnjak-lozic/works.html, kolovoz 2015.

[9] Unity: Unity 5.0 (2015.)

Dostupno na: https://unity3d.com/unity/whats-new/unity-5.0, kolovoz 2015.

[10] Unity: Unity documentation: Priručnik (2015.)

Dostupno na: http://docs.unity3d.com/460/Documentation/Manual/UICanvas.html, kolovoz 2015.

[11] David Winte: Noughts And Crosses - The oldest graphical computer gamer

Dostupno na: http://www.pong-story.com/1952.htm, kolovoz 2015.

[12] Windell Oskay: Resurrecting Tennis for Two, a videogame from 1958 (2008)

Dostupno na: http://www.evilmadscientist.com/2008/resurrecting-tennis-for-two-a-video-game-from-1958/, kolovoz 2015.

[13] Autodesk Maya: New Maya 2016 features

Dostupno na: http://www.autodesk.com/products/maya/features/new/list-view, kolovoz 2015.

[14] Triplet 3D: Maya vs 3DS Max vs Cinema 4D

Dostupno na: http://www.triplet3d.com/maya-vs-3ds-max-vs-cinema-4d, rujan 2015.

[15] Kelly L.Murdock: Autodesk Maya LT 2014 Review (2013)

Dostupno na: http://www.gamedev.net/page/resources/_/creative/visual-arts/autodesk-maya-lt-2014-review-r3355, rujan 2015.

[16] Maya 2014 Basic Animation & The Graph Editor

Dostupno na: https://www.wellesley.edu/sites/default/files/assets/departments/art/files/maya2014-basicanimationthegrapheditor.pdf, rujan 2015.

[17] University of Washington,: Types of Lights in Maya (2005)

Dostupno na: https://courses.cs.washington.edu/courses/cse458/05au/reading/lighting_tutorial/light_types.html, rujan 2015.

[18] Digital-tutors: 3ds Max, Blender, Maya LT (2015.)

Dostupno na: http://blog.digitaltutors.com/3ds-max-maya-lt-blender-3d-software-choose-asset-creation/, rujan 2015.

[19] Tiffani Tay: 3DS Max vs Maya: A Friendly Comparison (2014.)

Dostupno na: https://blog.udemy.com/3ds-max-vs-maya/, rujan 2015.

[20] Digital-tutors: Where Blender Functionality is Better Than Maya's Edit mode, Selection (2014.)

Dostupno na: http://blog.digitaltutors.com/where-blender-functionality-is-better-than-mayas/, rujan 2015.

[21] Digital media world: Autodesk Maya & 3ds Max 2016 Streamline Animation Tools and Simulations (2015.)

Dostupno na: http://www.digitalmedia-world.com/VFX/autodesk-maya-3ds-max-2016-animation-simulations, rujan 2015.

[22] Unity documentation: Light

Dostupno na: http://docs.unity3d.com/Manual/class-Light.html, rujan 2015.

[23] Unity 5.0

Dostupno na: https://unity3d.com/unity/whats-new/unity-5.0, rujan 2015.

[24] A History of the Unity Game Engine (2015.)

Dostupno na: https://www.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf , rujan 2015.

[25] Advantages of the Unity Game Engine – The ultimate Tool for Game Development

Dostupno na: https://blog.udemy.com/unity-game-engine/, rujan 2015.

[26] Simon Milbert IT Outsourcing Proffessional : Main benefits of Unity3D + grat tool for simpliflying game development,

Dostupno na: https://www.linkedin.com/pulse/main-benefits-unity3d-great-tool-simplifying-game-simon, rujan 2015.

[27] Digital-turtos: Unreal Engine 4 vs Unity: Which Game Engine is best for you? (2014.)

Dostupno na :http://blog.digitaltutors.com/unreal-engine-4-vs-unity-game-engine-best/, rujan 2015.

[28] Digital-tutors: Unity, Source 2, Unreal Engine 4, or CryEngine-Which Game Engine Should I Choose? (2015.)

Dostupno na: http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/, rujan 2015.

[29] Unity: Documentation - 3D formats (2015.)

Dostupno na: http://docs.unity3d.com/Manual/3D-formats.html, rujan 2015.

[30] Unity: Documentation – Importing Objects From Maya (2015.)

Dostupno na: http://docs.unity3d.com/Manual/HOWTO-ImportObjectMaya.html, rujan 2015.

[31] Unity: System Requirements for Unity 5.2 (2015.)

Dostupno na: https://unity3d.com/unity/system-requirements, rujan 2015.

[32] Unity: Documentation – Modeling Characters for Optimal Performance (2015.)

Dostupno na: http://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html, rujan 2015.

[33] Unity: Documentation – Optimizing Graphics Performance (2015.)

Dostupno na: http://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html, rujan 2015.

[34] Lahti University of Applierd Sciences: Animated low poly characters (March 2014.)

Dostupno na: http://www.theseus.fi/xmlui/bitstream/handle/10024/72726/Jolma_Valtteri.pdf?sequence=1 , rujan 2015.

[35] Game development: 3D Primer for Game Developers : An Overview of 3D Modeling in Games (March 2013.)

Dostupno na: http://gamedevelopment.tutsplus.com/articles/3d-primer-for-game-developers-an-overview-of-3d-modeling-in-games--gamedev-5704, rujan 2015.

[36] Unity documentation (2016.)

Dostupno na: http://docs.unity3d.com/Manual, svibanj 2016.

10. Popis slika

[1] Slika 2.1. – Wolfenstein 3D Izvor: http://www.gamespot.com/wolfenstein-3d/images

[2] Slika 2.2. – Serious Engine 4 Izvor: http://www.dsogaming.com/wp-content/uploads/2014/06

[3] Slika 3.8. Tipovi materijala za teksturiranje Izvor:http://www.3dvia.com/blog/components-of-a-3d-model/

[4] Slika 3.25. Quad Draw u Mayi LT Izvor:http://www.jigsaw24.com/news/wp content/uploads/2013/08/MayaLT_QuadDraw_Retopo.png

[5] Slika 4.2. Project view Izvor: http://docs.unity3d.com/401/Documentation/Images/manual/ProjectView40-0.jpg

[6] Slika 4.3. Scene view Izvor: https://cdn.tutsplus.com/active/uploads/legacy/tuts/270_IntroToUnityPart1/images/unity-scene-view.jpg

[7] Slika 4.5. Game view Izvor: http://www.takeinitiative.co.uk/wp-content/2009/12/unity_gameView.jpg

[8] Slika 4.8. Primjer platna sa elementima Izvor: http://docs.unity3d.com/460/Documentation/Manual/UICanvas.html

[9] Slika 4.10. MSF logika za svaku platformu Izvor: https://www.linkedin.com/pulse/main-benefits-unity3d-great-tool-simplifying-game-simon

[10] Slika 4.11. Vrste sjenčanja i njihov izgled Izvor: http://docs.unity3d.com/430/Documentation/Manual/Materials.html

[11] Slika 4.12. Audio mixer Izvor: http://docs.unity3d.com/Manual/AudioMixer.html

[12] Slika 4.13. Kontroler animacije i parametri Izvor: http://docs.unity3d.com/Manual/Animator.html

Prilozi:

DVD

Sveučilište
Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni 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, Antonio Jagodić pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor završnog rada pod naslovom Izrada 3D igre u Unity razvojnom okruženju, te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student:
Antonio Jagodić

(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, Antonio Jagodić neopozivo izjavljujem da sam suglasan s javnom objavom završnog rada pod naslovom Izrada 3D igre u Unity razvojnom okruženju čiji sam autor.

Student:
Antonio Jagodić

(vlastoručni potpis)