Tijek izrade 3D okruženja u Unreal Engine 4

Mačković, Damir

Sveučilište
Sjever

Završni rad br. 520/MM/2017

Tijek izrade 3D okruženja u Unreal Engine 4

Damir Mačković, 0297/336

Varaždin, lipanj 2017. godine

TIJEK IZRADE 3D OKRUŽENJA U UNREAL ENGINE 4

UNREAL ENGINE 4 ENVIRONMENT WORKFLOW

Damir Mačković
Sveučilište Sjever – Sveučilišni centar Varaždin

Sažetak

Kako bi bilo moguće izraditi 3D okruženje za računalnu igru pokretanu Unreal Engine-om 4 potrebno je prvo proučiti klasične igre te njihove elemente. Također potrebno je razumjeti industriju računalnih igara te uloge dizajnera igara i level dizajnera. Važnost dizajnera igara i level dizajnera zajedno sa igrama, industrijom računalnih igara opisani su u teorijskom dijelu rada. Za vrijeme rada na projektu S.S. Infinity osmišljen je tijek rada koji je dokumentiran u ovom radu. Dokumentirani tijek rada uključuje opise izrade 3D modela, teksturiranja, pakiranja tekstura za Unreal Engine 4, izrade 3D okruženja u Unreal Engine 4, postavljanje osvjetljenja, post proces efekata i izrade animacije. Izrađeno 3D okruženje je u početku izrađeno za računalnu igru, ali je zbog nedostatka vremena iskorišteno za izradu video animacije za potrebe kolegija Video animacija. Video animacija pokazuje rani stadij razvoja računalne igre S.S. Infinity.
Ključne riječi: 3D okruženje, 3D modeliranje, igre, level dizajn, teksturiranje, Unreal Engine 4.

Abstract

In order to understand the creation of 3D environments in Unreal Engine 4 it is necessary to study conventional games and their essential elements. It is also necessary to understand both game and level designer's roles in the computer game industry. The importance of their roles along with games and computer game industry in general are described in the theoretical part of this paper. During the developement of the S.S. Infinity a complete documented workflow had been created and later described in this paper. The documented workflow includes descriptions of creating 3D models, textures, texture packages, lighting, post process effects, animations and creating 3D environment. 3D environmnt was intended to be used for a PC game, but due to the lack of time it was only used for Video Animation classes. The final result of the project is a flythrough animation which shows an early stage of game development.
Keywords: *3D environment, 3D modeling, games,* level design, texturing, Unreal Engine 4.

1. Uvod
1. Introduction

Računalne igre su medij koji kroz integraciju drugih medija pružaju igraču osjećaj uključenosti i zadovoljstva kroz zabavu. Kako bismo računalne igre razumjeli, potrebno je prvo razumjeti elemente koji tvore neku igru. Bitno je znati da se računalne igre u svojoj osnovi ne razlikuju previše od običnih,

klasičnih igara. Za razliku od primjerice filma ili glazbe, računalne igre isporučuju sadržaj na interaktivan način kojim se igraču svijet igre prezentira kao nova stvarnost. Većina 3D računalnih igara današnjice oslanja se na vizualni izgled 3D okruženja u kojem se radnja igre odvija, jer ono je jedan od presudnih faktora u stvaranju osjećaja druge stvarnosti kod igrača. Za vizualni izgled, funkcionalnost 3D okruženja i iskustvo igranja (eng. Gameplay) u računalnim 3D igrama zaslužni su level dizajneri (eng. Level Designer). Oni na osnovu suradnje s producentom, umjetničkim redateljem i dizajnerom igre implementiraju sadržaje igre, i u konačnici izrađuju 3D okruženja. Tijek razvoja računalne igre tako u osnovi ovisi o odlukama četiri ključne osobe u razvojnoj kući: o producentu, umjetničkom redatelju, dizajneru igre i level dizajneru.

Cilj ovog rada je u osnovama pojasniti elemente igara, te uloge dizajnera igre i level dizajnera u procesu razvoja računalne igre, pri tom fokusirajući se na izradu 3D okruženja.

Ovaj rad se fokusira na tijek izrade 3D okruženja za igru pokretanu Unreal Engine 4. Radni tijekovi proizvodnje nekog 3D sadržaja pa i okruženja su različiti na osobnoj i na razini tvrtke koja igru razvija. Tijekom izrade 3D okruženja za projekt „S.S. Infinity“ razvio sam svoj tijek rada kako bi olakšao izradu i što bolje organizirao vrijeme. Tijek rada dokumentiran je u ovom radu, i iako on može poslužiti nekome tko tek ulazi u svijet level dizajna, on nikako nije smjernica kako izraditi dobro 3D okruženje računalne igre.

2. Igre
2. Games

Prije nego bismo mogli početi govoriti o level dizajnu ili dizajnu igara i izradi 3D okruženja računalnih igara, potrebno je odrediti što igre jesu i kako funkcioniraju. Lako je za pretpostaviti kako svatko zna što je igra, ali danas postoje toliko različitih vrsta igara pa je najbolje ne raditi pretpostavke osnovane samo na osobnom iskustvu. Za razumijevanje igara ključno je identificirati potrebne elemente koje neka igra mora posjedovati, zatim definirati što igra je bazirajući se na tim elementima. Tek kada razumijemo elemente igre, onda možemo raspravljati o računalnim igrama i njihovim prednostima nad klasičnim igrama. [1]

3. Tijek rada izrade 3D okruženja u Unreal engine 4
3. Unreal engine environment workflow

Tijekovi rada (eng. workflow) su organiziran način proizvodnje nekog sadržaja. Sve velike razvojne kuće koriste neki svoj, već ustaljeni tijek rada koji im omogućuje maksimalno iskorištavanje vremena uz veliku učinkovitost. Za vrijeme razvoja 3D okruženja za projekt imena S.S. Infinity razvijen je cijelokupan tijek rada koji obuhvaća 3D modeliranje, teksturiranje, izradu 3D okruženja u Unreal Engine 4, osvjetljenja, post proces efekata i animacije.

3.1. Projekt S.S. Infinity
3.1. S.S. Infinity project

Projekt *S.S Infinity* je zamišljen kao 3D računalna avantura iz prvog lica, u kojoj igrač otkriva elemente transportnog svemirskog broda pod imenom *Infinity*. *S.S. Infinity* je samoodrživi svemirski brod kojim ne upravljaju ljudi, a služi za transport raznog opasnog tereta. Glavni cilj igre je pronaći način kako doći do glavnih kontrola svemirskog broda i ispraviti mu putanju koja ga vodi u središte zvijezde. Igra bi se sastojala od 4 prostorije koje zajednički tvore isto 3D okruženje, u kojima lebdeći robotski nosači prenose razne spremnike iz jedne u drugu prostoriju. Ovladavanjem mehanike robotskih

nosača, igrač bi trebao koristiti logičke zaključke kako bi shvatio na koji način svemirski brod funkcionira kako bi došao do kontrolne prostorije i ispravio putanju svemirskog broda.
3D okruženje pokreće pokretač igre Unreal Engine 4. Zamišljeno je za potrebe računalne igra, ali je zbog nedostatka vremena za daljnji razvoj korišteno za potrebe kolegija Video animacija.

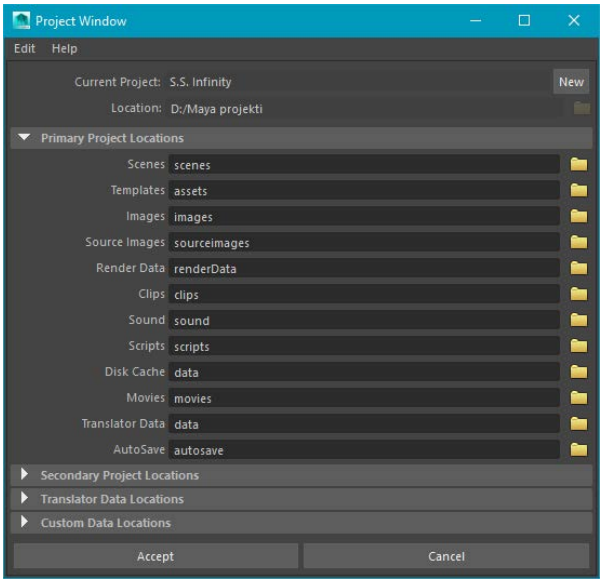
Slika 1 S.S. Infinity logo
Figure 1 S.S. Infinity logo

3.2. Programski paketi
3.2. Software

Kako bi se izradilo 3D okruženje za projekt S.S. Infinity, korišteno je nekoliko programskih paketa. Izrada 3D modela zahtijevala je znanje korištenja Autodesk Maya 2016 paketa za 3D modeliranje, Allegorithmic Substance Painter 2 za izradu tekstura, te Substance Designer 5 za izradu dodatnih tekstura potrebnih za ispravan prikaz u Unreal Engine 4 i pakiranje svih tekstura u jednu datoteku razumljivu Unreal Engine 4. Epic Games Unreal Engine 4 je korišten za konačnu izradu 3D okruženja i animacije.

3.3. Organizacija projekta na hard disku računala
3.3. Project data structure

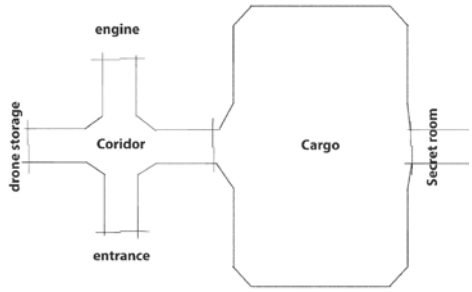
Razvoj 3D okruženja računalnih igara velik je pothvat pa je organizacija projekta na hard disku računala izrazito važna. Opseg projekta vrlo brzo može narasti pa je organizacija mapa i datoteka kao i konvencija njihovih naziva izrazito važna, kako zbog snalaženja tako i zbog uštede vremena. Svoj tijek rada započeo sam koristeći se organizacijskim mogućnostima unutar programskih paketa koje sam koristio.
Maya 2016 ima odličan podsustav za organizaciju datotečne strukture projekta kojeg je moguće modificirati potrebama projekta. Koristio sam predefiniranu strukturu mapa, dok sam konvenciju naziva datoteka osmislio sam.



Slika 2 Project window u Maya 2016
Figure 2 Maya 2016 Project window

3.4. Izrada 3D okruženja
3.4. Making 3D environment

Izrada 3D okruženja projekta S.S. Infinity zahtijevala je znanje 3D modeliranja, izrade tekstura, poznavanje osnova level dizajna za konačnu izradu 3D okruženja, te osnova kadriranja i video montaže za izradu animacije. 3D modeli i njihove teksture vizualno čine veliki postotak konačnog izgleda nekog 3D okruženja, te zajedno s korektnim i vješto izvedenim osvjetljenjem, čine sveukupan dojam i atmosferu 3D okruženja.
Prije izrade 3D modela koji čine 3D okruženje bilo je potrebno izraditi prostorni plan 3D okruženja. Prostorni plan izrađen je na papiru zbog veće kreativnosti.

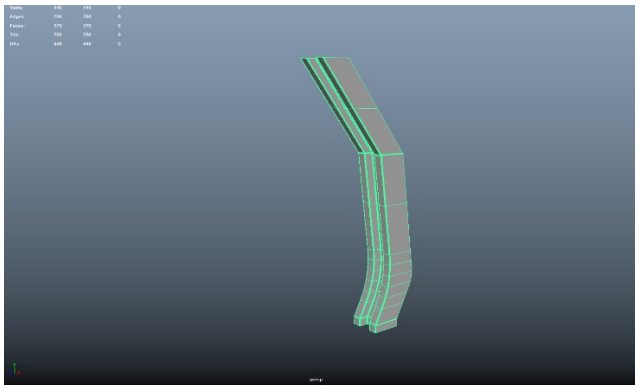


Slika 3 Rekreirani prostorni plan 3D okruženja

Figure 3 Recreated blueprint of the 3D environment

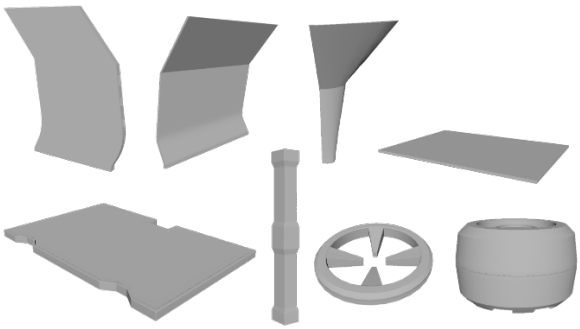
3.4.1. Izrada 3D modela
3.4.1. Making 3D models

Svi 3D modeli izrađeni za potrebe izrade 3D okruženja projekta S.S. Infinity izrađeni su Autodesk Maya 2016 programskim paketom. Nakon što je ugrubo skicom definiran prototip 3D okruženja, odlučio sam kako bi za izradu ovog 3D okruženja bilo najbolje krenuti s modularnim pristupom ili tako zvanim modularnim level dizajnom. Modularni level dizajn podrazumijeva korištenje zasebnih 3D elemenata koji se nadovezuju i zajedno tvore jednu cjelinu, kao što je to u stvarnom svijetu. Glavni princip modularnog level dizajna je krenuti izrađivati 3D modele koji čine kutove 3D okruženja, zatim s 3D modelima zidova, podnica, stropova. Kada su svi ti 3D modeli izrađeni, i sa svim pripadajućim teksturama uvezeni u pokretač igre, tada se kreće s izradom 3D okruženja. Kada je glavni prostor izrađen prema prototipu, tada se kreće s izradom 3D modela rekvizita (eng. props) koji će upotpuniti prostor i tvoriti sveukupnu atmosferu 3D okruženja.



Slika 4 3D model stupa

Figure 4 Column 3D model



Slika 5 Pojedini 3D modeli 3D okruženja S.S. Infinity.

Figure 5 Several 3D models of S.S. Infinity 3D environmen

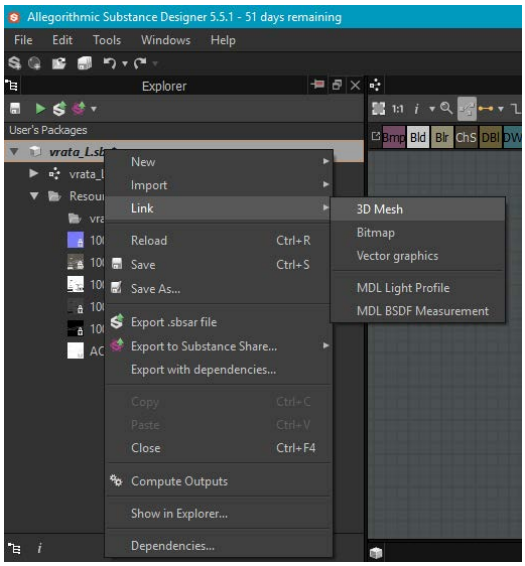
3.4.2. Izrada tekstura
3.4.2. Texturing

Za izradu tekstura korišteni su programski paketi Substance Designer i Substance Painter 2. Unreal Engine 4 implementira *Metalic/Roughness* PBR način prikaza tekstura. PBR je fizički točan prikaz tekstura u pokretaču igre. Substance programski paketi implementiraju isti način prikaza tekstura, te su namijenjeni upravo izradi PBR tekstura. Iako su Substance programski paketi namijenjeni za izradu tekstura i uvelike ubrzavaju proces izrade istih, izrada tekstura i dalje zauzima puno vremena. Kako bi ubrzao izradu tekstura, koristio sam se tijekom rada između Substance Designer 5 i Substance Painter 2 koje preporučuje Allegorithmic, ali uz dodatne modifikacije. [2]

Substance Designer 5 je korišten prvotno za izradu dodatnih tekstura, potrebnih za rad u

Substance Painter 2, dok je Substance Painter 2 korišten za izradu konačnih tekstura 3D modela.. Substance Painter 2 za izradu PBR Metallic/Roughness tekstura treba *Normal map* teksturu od koje će kasnije generirati ostale teksture: *World space normals*, *Ambient Occlusion*, *Curvature*, *Position* i *Thickness*. Te dodatne teksture ne izvažaju se za potrebe prikaza u pokretaču igre, već služe Substance Painter 2 grafičkim algoritmima za generiranje raznih efekata na osnovnim teksturama, poput ogrebotina po rubovima, hrđe i dr. [3,4]

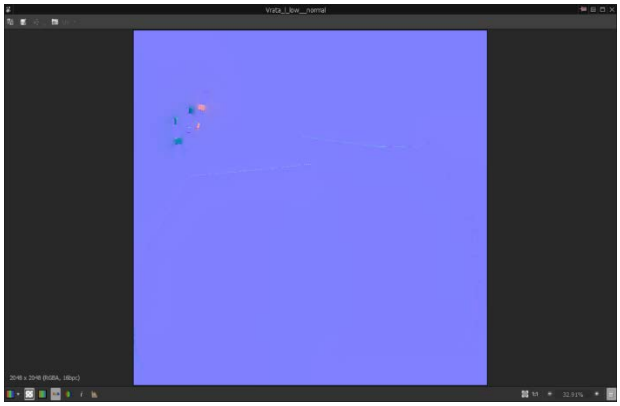
Prvo što je potrebno kod izrade tekstura za 3D model u Substance programskim paketima je izraditi *normal map* i *curvature* teksture. Za primjer uzimam 3D model lijevih vrata. Prvo je bilo potrebno uvesti oba 3D modela (niske i visoke poligonalne rezolucije). Uvažanje 3D modela u Substance Designer 5 se radi tako što se na desni klik miša na glavnu mapu projekta otvori izbornik link u kojem je potrebno odabrati *3D mesh.*



Slika 6 Uvoz 3D modela u Substane Designer 5

Figure 6 Importing 3D meshes in Substance designer 5

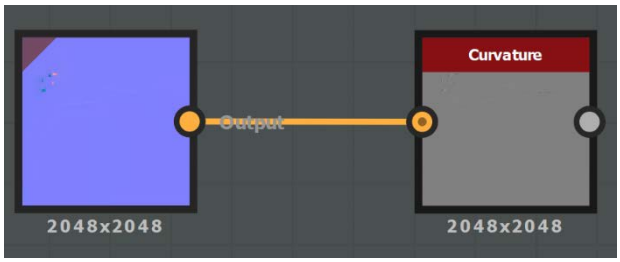
Nakon uvoza 3D modela niske i visoke poligonalne rezolucije, potrebno je bilo izraditi normal map teksture. U kratko normal map tekstura sadrži informacije o tome kako se svjetlost odbija od 3D modela, tj. opisuje ga u 3D prostoru. Uobičajena je tehnika projicirati normal informacije sa 3D modela visoke poligonalne reozolucije na onaj niske poligonalne rezolucije, kako bi dobio više detalja. Izrada normal map teksture omogućena je potprogramom za izradu normal map tekstura u Substance Painter i Designer. Za projekt S.S Infinity korišten je onaj u Substance Designer 5 jer je fleksibilniji. [5]



Slika 7 Izrađena normal map tekstura u Substance Designer 5

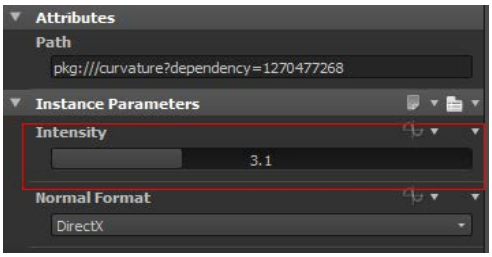
Figure 7 Finished normal map texture in Substance Designer 5

Pomoću izrađene normal map teksture izrađuje se curvature tekstura koju kasnije koristi Substance Painter 2. Izrada te teksture puno je jednostavnija. Potrebno je samo u Substance Designer 5 spojiti čvorište koje sadrži normal map teksturu sa curvature čvorištem i podesiti njen intenzitet.



Slika 8 Izrada curvature teksture u Substance Designer 5

Figure 8 Creating curvature texture in Substance Designer 5

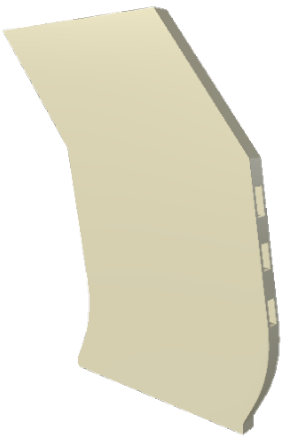


Slika 9 Podešavanje intenziteta curvature teksture

Figure 9 Setting the curvature intensity

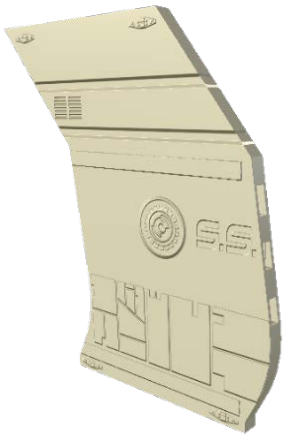
Izrađene normal map i curvature teksture se zatim uvažaju u Substance Painter 2 prilikom otvaranja novom projekta. Nakon što su uvezene, potrebno je izraditi ostale dodatne teksture u prozoru bake textures.

Nakon što postoje sve potrebne dodatne teksture za rad Substance Painter 2, moguće je izrađivati teksture koje će biti izvezene za uporabu s Unreal Engine 4. Svoj proces izrade tih tekstura podijelio sam na nekoliko faza: izrada height detalja, izrada osnovnih materijala, izrada materijala isijavanja svjetlosti (eng. emmisive). Izrada tekstura u te tri faze uvelike mi je ubrzala rad jer sam se mogao fokusirati samo na te tri faze izrade. Izrađene teksture su time dobile uniformiran vizualni stil.



Slika 10 3D model ljevih vrata bez height detalja

Figure 10 Left door 3D model without height details

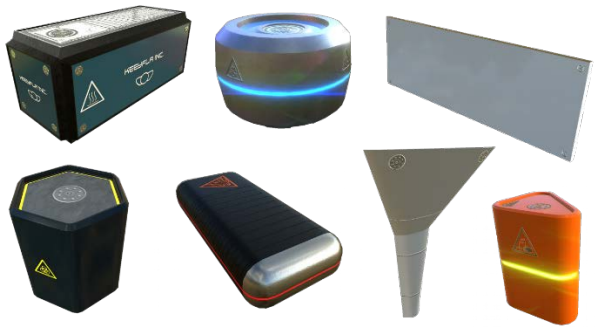


Slika 11 3D model lijevih vrata sa height detaljima

Figure 11 Left door 3D model with height details

Slika 12 3D model lijevih vrata sa završenim teksturama

Figure 12 Left door 3D model with all textures



Slika 13 Neki od 3D modela s izrađenim teksturama

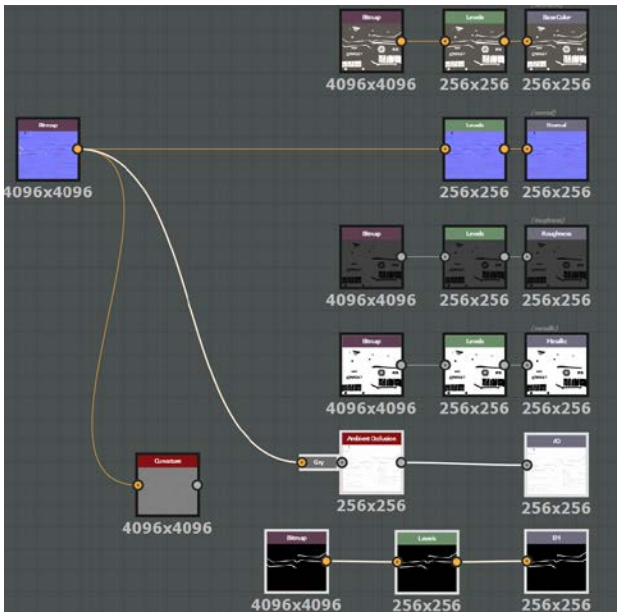
Figure 13 Several 3D models with finished textures

3.4.3. Izvoz i pakiranje tekstura za Unreal Engine 4

3.4.3. Export and packaging texture for Unreal engine 4

Unreal Engine 4 podržava Allegorithmicov programski dodatak za Substance Designer 5. Substance programski dodatak za Unreal Engine 4 omogućuje otvaranje paketa tekstura stvorenih u Substance Designer 5. Unreal Engine 4 implementira *material editor* u zasebnom prozoru unutar kojeg se pomoću vizualno skriptnog računalnog koda upravlja grafičkim sustavom Unreal Engine 4. Kako Unreal Engine 4 implementira *Metallic/Roughness* način prikaza PBR tekstura, tako uključuje ista čvorišta na koje je potrebno spojiti sve potrebne teksture stvorene Substance Designer 5 programskim paketom. Substance programski dodatak za Unreal Engine 4 omogućuje automatsko izrađivanje materijala za 3D model što znatno ubrzava rad s dodjeljivanjem tekstura 3D modelu u *material editoru*. Materijal u Unreal Engine 4 je naziv za pakirani vizualno skriptni računalni kod koji definira izgled 3D modela u 3D okruženju. [6,7]

Sve izrađene teksture u Substance Painter 2 potrebno je uvesti u radni prostor Substance Designer 5 i spojiti ih na odgovarajuća čvorišta. Nakon što su sve teksture povezane sa izlaznim čvorištima potrebno je podesiti relacije skaliranja tekstura. Jednom kada se paket tekstura uveze u Unreal Engine 4 biti će moguć odabir tekstura različitih rezolucija.



Slika 14 Mreža čvorišta u Substance Designer 5

Figure 14 Node grid in Substance Designer 5

Relacija skaliranja je na izvornim čvorištima podešena na relative to parent, dok je na izlaznim podešena na relative to input.

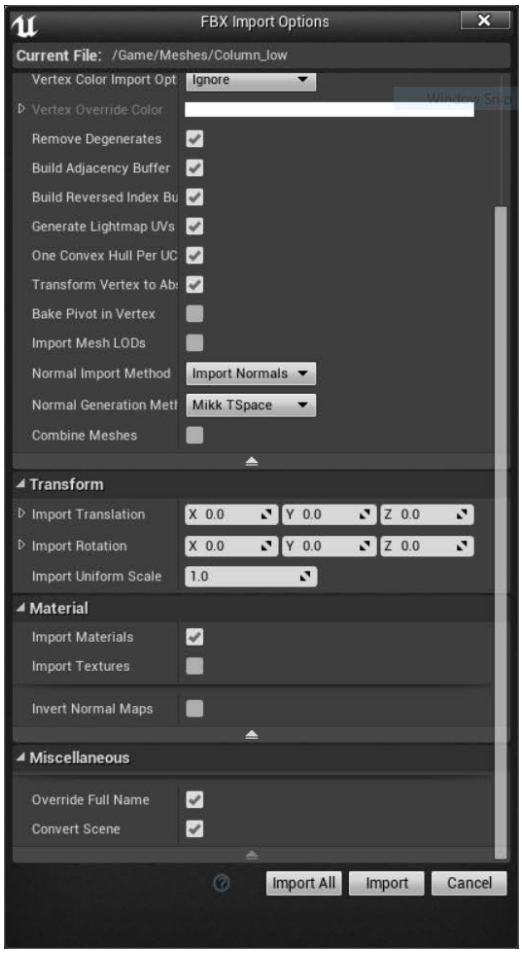
Nakon podešenih postavki skaliranja tekstura, one su spremne za pakiranje u jednu datoteku koja će kasnije biti uvezena Substance programskim dodatkom u Unreal Engine 4. Pakiranje tekstura izvodi se tako što se desnim klikom miša na glavnu mapu projekta pristupi *export .sbsar file* prozoru i pospremi u predefiniranu mapu. Substance Designer 5 će automatski stvoriti Unreal Engine 4 materijal i povezati sve teksture na potrebna čvorišta. U Unreal Engine 4 *material editoru* bit će omogućen odabir rezolucije tekstura. Zbog prirode proceduralnih tekstura, vrlo je lako imati u istom projektu više tekstura različitih rezolucija i tako uštedjeti na računalnim resursima. [8]

Izrada tekstura za projekt S.S. Infinity velik je dio sveukupnog tijeka rada na tom projektu. Razvoj organiziranog tijeka rada prilikom izrade tekstura omogućio mi je ubrzanu izradu istih.

3.4.4. Uvoz 3D modela i pripadajućih tekstura u Unreal Engine 4
3.4.4. Importing 3D models and textures in Unreal Engine 4

Unreal Engine 4 implementira upravitelj mapama i datotekama projekta pod nazivom *content browser*, koji omogućuje izvrsnu organizaciju projekta. *Content browser* upravlja velikim brojem mapa koje sadrže velik broj različitih datoteka potrebnih za izradu 3D okruženja, poput specijalnih efekata, animacija likova, 3D modela, tekstura, materijala, i mnogo drugih. *Content browser* kod uvoza datoteka automatski prepoznaje njihov tip, i u skladu s time otvara specifične prozore za njih. Također, jednom kada su datoteke uvezene, vrlo lako ih je pretraživati, ponovo učitavati i seliti u druge mape. [9]

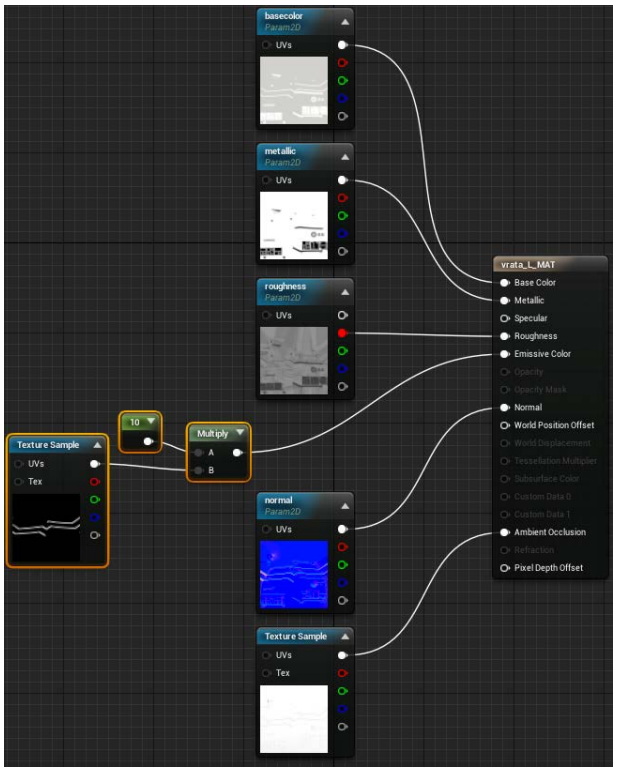
Za uvoz 3D modela i pripadajućih tekstura potrebno je obaviti nekoliko koraka. Prvo je potrebno uvesti 3D modele u prethodno definiranu mapu *meshes*. Uvoz datoteka obavlja se klikom miša na gumb *import* u *content broswseru*. Nakon klika mišem na gumb *import*, potrebno je locirati *fbx* datoteke koje sadrže izvezene 3D modele izrađene Autodesk Maya 2016 programskim paketom i kliknuti na gumb *import all* kako bi se uvezli 3D modeli s osnovnim materijalima koji su pakirani s 3D modelima. Isti postupak je potrebno ponoviti i sa Substance paketima tekstura.



Slika 15 Uvoz 3D modela u content browseru

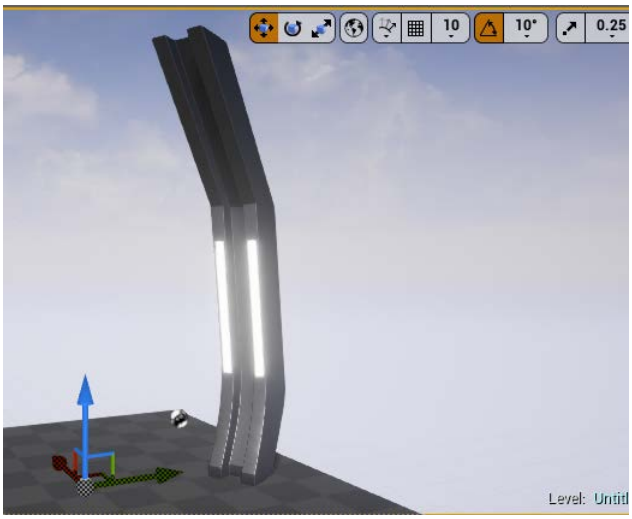
Figure 15 Importing 3D models in content browser

Nakon što su svi 3D modeli sa pripadajućim teksturama uvezeni u Unreal Engine 4, potrebno je u material editoru povezati dodatne teksture ambient occlusion i emissive. Kako Unreal engine 4 implementira identičan način prikaza tekstura na 3D modelima, tako je potrebno sve teksture spojiti na pripadajuća mjesta u izlaznom čvorištu.



Slika 16 Izrađen materijal u material editoru

Figure 16 Finished material in material editor



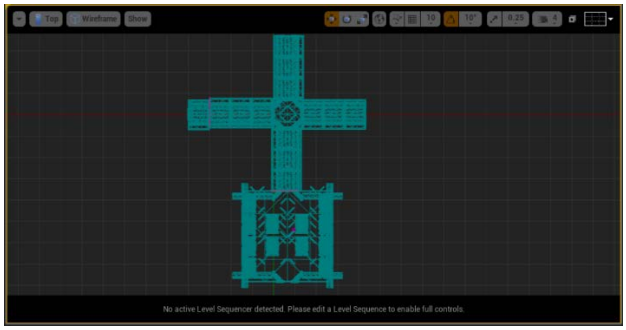
Slika 17 3D model sa svim teksrurama

Figure 17 3D model with all textures

3.4.5. Izrada koridora i skladišta a u Unreal Engine 4

3.4.5. Making a coridor and cargo hold in Unreal Engine 4

Jednom kada su 3D modeli s pripadajućim teksturama bili uvezeni i za njih izrađeni materijali, mogao sam započeti s izradom 3D okruženja koridora i skladišta. Koristio sam se alatima za pomicanje i rotaciju 3D modela kako bih ih smjestio na odgovarajuća mjesta. Koridor sam izradio tako što sam prvo izradio jedan dio koristeći se pripadajućim 3D modelima namijenjenim za izradu koridora. Na početak koridora postavio sam 3D modele vrata koja ću kasnije animirati, dok sam na kraj koridora stavio 3D modele koji tvore kutove. Za smještaj 3D modela na željena mjesta ključno je što sam prilikom njihove izrade uskladio mjerne jedinice i postavke mreže u Maya 2016. Zbog istih mjernih jedinica i postavki mreže između Unreal Engine 4 i Maya 2016, omogućeno je nalijeganje 3D modela jedan do drugoga, bez praznog prostora.



Slika 18 Žičani prikaz gornjeg prikaza S.S. Inifinity 3D okruženja

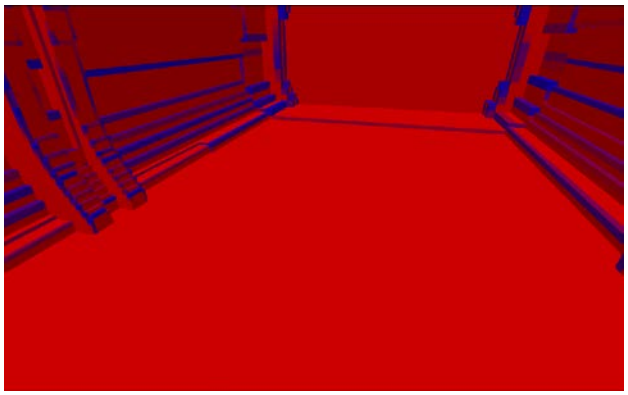
Figure 18 Top view Wireframe of S.S. Infinity 3D environment

Slika 19 Neosvjetljen modul koridora

Figure 19 Unlit coridor module

3.4.6. Osvjetljenje
3.4.6. Lighting

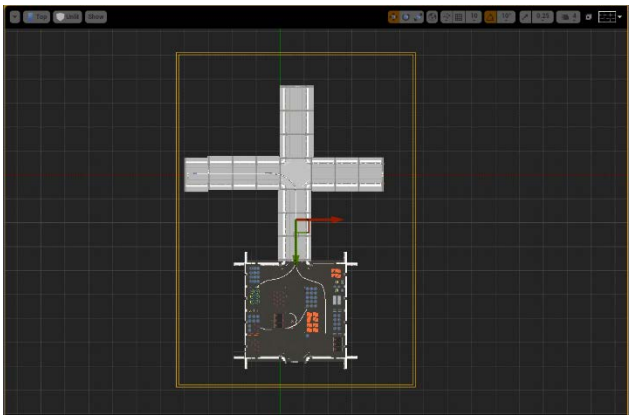
Osvjetljenje u Unreal Engine 4 je izvedeno tako da što više štedi računalne resurse, jer je izuzetno zahtjevno za računala. Uobičajena je praksa izraditi takozvane mape osvjetljenja koje se dodjeljuju 3D modelima na sličan način kao teksture. Mape osvjetljenja sadrže informacije o svjetlu te se tako izbjegava bespotrebno računanje svijetla u svakoj slici koju računalo prikazuje na zaslonu. Iako je uobičajeno izraditi mape osvjetljenja, postoje uvjeti kada je potrebno dinamično osvjetljenje, tj. ono koje se izvodi u stvarnom vremenu. [10]
Za izradu osvjetljenja 3D okruženja S.S. Infinity koristio sam dinamično osvjetljenje. Dinamično osvjetljenje koje sam koristio nije ono implementirano u Unreal Engine 4. Koristio sam način simulacije indirektnog osvjetljenja naziva *VXGI* tvrtke *Nvidia*. Korištenje tog načina osvjetljenja značajno mi je ubrzalo rad na izradi osvjetljenja za projekt S.S. Infinity. *VXGI* je tako značajan dio tijeka rada izrade 3D okruženja.
VXGI simulira indirektno osvjetljenje koristeći *voxel* tehnologiju prikaza. *Voxel* se najjednostavnije može definirati kao 3D pixel. Indirektno osvjetljenje podrazumijeva odbijenu svjetlost od svih površina 3D okruženja i normalna je pojava u stvarnom svijetu. Izračun odbijene svjetlosti izrazito je zahtjevan za računala pa se stoga inače ne implementira u pokretače računalnih igara. VXGI uspješno simulira indirektno osvjetljenje tako da postavlja niz *voxel* stožaca kroz cijelo 3D okruženje i dodaje osvjetljenje tamo gdje bi se inače pojavilo u slučaju indirektnog osvjetljenja. Dodavanje globalne iluminacije u 3D okruženje računalne igre znatno povećava realizam jer prikazuje svu svjetlost u 3D okruženju, pa čak i onu odbijenu od sjajnih površina. [11]



Slika 20 Voxelizirano 3D okruženje
Figure 20 Voxelised 3D environment

3.4.6.1. Refleksije i post proces efekti
3.4.6.1. Reflection and post proces effects

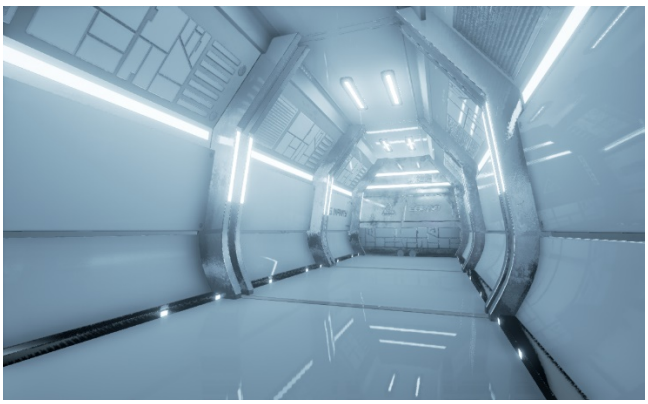
Kako bi bilo moguće korstiti post process efekte u Unreal Engine 4, potrebno je u 3D okruženje uvesti post process volume objekt. Taj objekt se nalazi pod izbornikom volumes. Jednom kada je taj objekt uvezen u 3D okruženje moguće je upravljati raznim post proces efekrima.
Refleksije su odmah iza osvjetljenja po kompleksnosti računanja. Zahtijevaju mnogo računalne moći kako bi se realno prikazivale. Stoga je potrebno ograničiti ih unutar određenih granica. Unreal Engine 4 radi to na domišljat i efikasan način implementacijom zonama hvatanja refleksija *(eng. reflection capture*). Tako postoje 3D objekti sferne i poligonalne zone hvatanja refleksija (*eng. sfere, box reflection capture*).
Za potrebe 3D okruženja S.S. Infinity korištena je samo poligonalna zona za hvatanje refleksija. Poligonalna zona je postavljena tako da obuhvaća cijelo 3D okruženje. Na isti način je mogla biti postavljena i sferna zona hvatanja refleksija, ali zbog korištenja VXGI tehnologije, praktički je svejedno koja zona hvatanja refleksija će se koristiti.



Slika 21 Poligonalna zona hvatanja refleksija

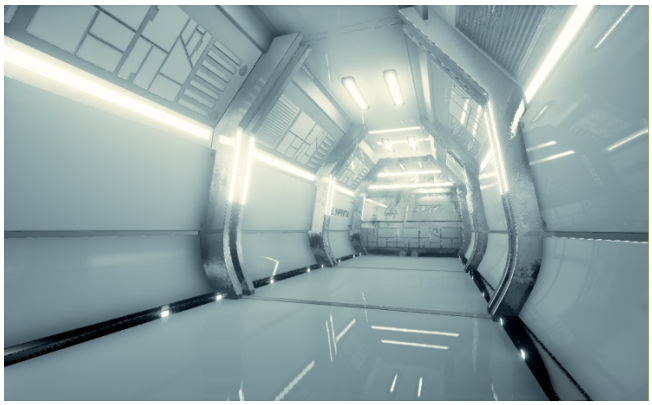
Figure 21 Polygonal reflection capture

Gradacija boja omogućena je u postavkama *post process volume* objekta pod odjeljkom *scene color*. Gradaciju boja omogućuje *LUT* datoteka stvorena u Adobe Photoshop CC. Dovoljno je uzeti jedan snimak zaslona 3D okruženja u Unreal Engine 4, i u Adobe Photoshop CC obraditi sliku te ju izvesti kao *LUT* datoteku. *LUT* datoteku potrebno je učitati u *post proces volume* postavkama i cijeli prikaz 3D okruženja će biti u skladu s obrađenom snimkom zaslona. Gradacijom boja postigao se filmski izgled 3D okruženja.



Slika 22 Prikaz 3D okruženja bez gradacije boja

Figure 22 3D environment without color grading



Slika 23 Prikaz 3D okruženja s gradacijom boja

Figure 23 3D environment with color grading

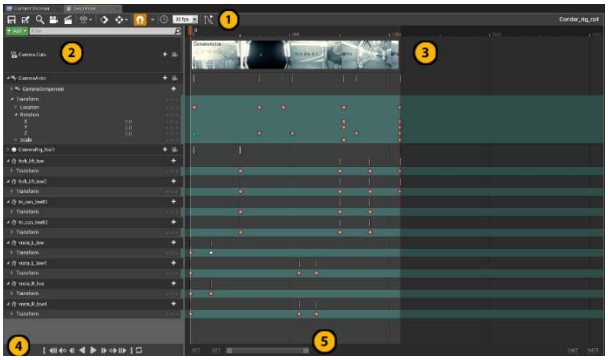
Postavljanjem osvjetljenja, refleksija, i post proces efekata završena je izrada 3D okruženja projekta S.S. Infinity. Izrađeno 3D okruženje koje sadrži koridor i skladište svemirskog broda S.S. Infinity sa grafičke strane u potpunosti je spremno za daljnji razvoj računalne igre. 3D okruženje je kasnije iskorišteno za izradu kratke animacije u kojoj se prikazuje radni stadij razvoja igre S.S. Infinity. Za potrebe računalne igre, bilo bi potrebno uložiti još mnogo truda za implementiranje interaktivnih elemenata, umjetne inteligencije, sustava upravljanja igrom i mnogo drugih sustava koji bi zajedno tvorili računalnu igru S.S. Infinity.

Slika 24 Snimak zaslona skladišta u S.S. Infinity 3D okruženju

Figure 24 S.S. Infinity 3D environment screen capture

4. Izrada animacije
4. Creating Animation

Izrada animacija unutar Unreal Engine 4 programskog paketa omogućena potprogramom Sequencer. Sequencer je moćan dio Unreal Engine 4 u kojem je moguće izrađivati animacije gotovo jednake kvalitete kao i u namjenskim programskim paketima za izradu istih. Za izradu video datoteke animacije, korišten je Adobe Premiere CC programski paket. [12]

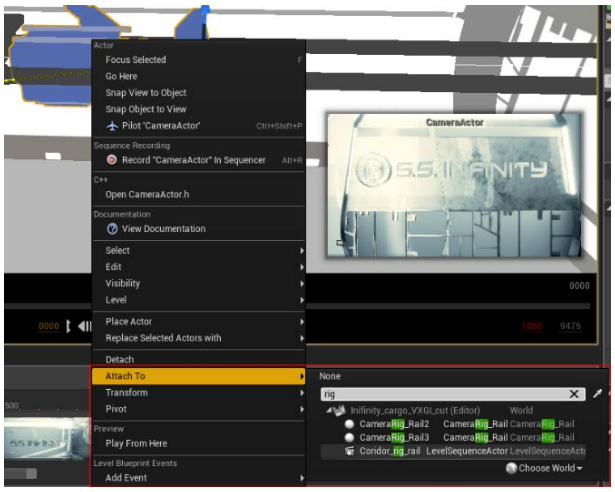


Slika 25 Unreal Engine 4 Sequencer

Figure 25 Unreal Engine 4 Sequencer

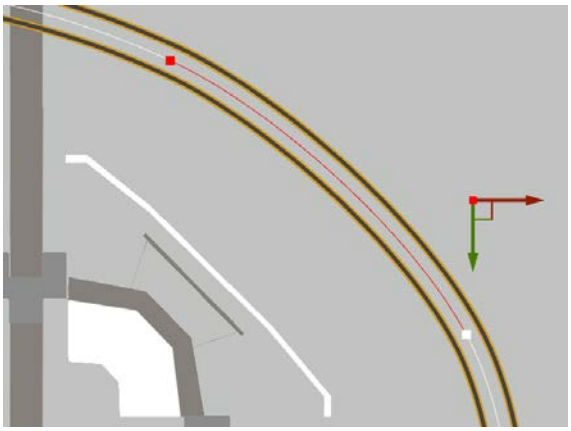
4.1. Animacija kamere i 3D modela
4.1. Camera and 3D model animation

Animacija kamere omogućena je rig rail objektom koji se postavlja u 3D okruženje. Taj objekt se ponaša identično kao i tračnice za kameru u stvarnom svijetu. Dostupan je na modes > cinematics. Kada se rig rail uvede u 3D okruženje i postavi na željeno mjesto, potrebno mu je pridružiti kameru. Kameru je također moguće pronaći u cinematics izborniku. Potrebno ju je povezati s rig rail objektom tako da se pristupi padajućem izborniku desnim klikom miša i odabere attach > CameraRig_Rail. Nakon što su ta dva objekta povezana, potrebno je izraditi putanju kamere. Rig rail objekt je u osnovi vektorska krivulja po kojoj kamera putuje.



Slika 26 Povezivanje kamere s rig rail objektom

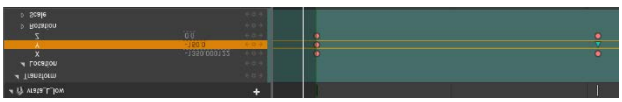
Figure 26 Attaching camera to rig rail



Slika 27 Rig rail objekt

Figure 27 Rig rail object

Za animaciju 3D modela korištena je tehnika postavljanja *keyframeova*. Za primjer uzimam animaciju 3D modela lijevih i desnih vrata. Način izrade animacije otvaranja vrata prilikom prolaska kamere je bio sljedeći; prvo je bilo potrebno dodati početni *keyframe* na nultu sliku. Nakon toga, oba 3D modela vrata postavljena su u otvoreni položaj i dodan im je *keyframe* za translaciju po Y osi na 104. slici.



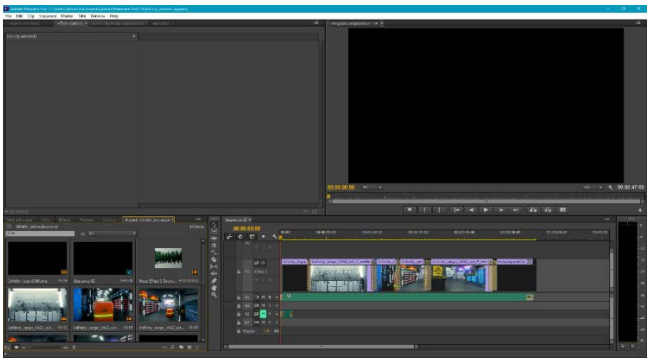
Slika 28 Postavljanje keyframeova za 3D modele lijevih vrata

Figure 28 Keframing left door 3D models

Nakon što su izrađeni svi kadrovi animacije, izvezene su sve slike u png formatu rezolucije 1920x1080px.

4.2. Izrada video datoteke
4.2. Creating video file

Za izradu video datoteke korišten je Adobe Premiere CC, programski paket namijenjen za nelinearnu montažu videa. Za izradu videa u Adobe Premiere CC, potrebno je otvoriti novi projekt i novu sekvencu. Zatim je potrebno uvesti sve izvezene kadrove iz Unreal Engine 4 i uvesti ih u video trake Adobe Premiere CC kako bi se mogle uređivati i spajati s ostalima.
Nakon što su svi kadrovi uvezeni u video traku i posloženi u željeni redoslijed, potrebno je bilo napraviti najavne i odjavne špice te dodati glazbu. Video je nakon toga bilo potrebno izvesti u MPEG-4 kontejner s .h264 načinom kodiranja. U prozoru za izvoz videa, odabrane su predefinirane postavke koje omogućuju kodiranje u .h264 *kodek* koji omogućuje spremanje video datoteke maksimalne kvalitete videa, uz zadovoljavajuću veličinu datoteke na hard disku računala. Pristup prozoru za izvoz videa omogućen je padajućim izbornikom *File > export media.*



Slika 29 Izrađena montaža videa u Adobe Premiere CC
Figure 29 Finished video in Adobe Premiere CC

5. Zaključak
5. Conclusion

Razvoj 3D okruženja za računalne igre velik je pothvat. Za razvoj S.S. Infinity 3D okruženja bilo je potrebno razviti novi tijek rada kako bi se maksimalno iskoristilo vrijeme. Najveći dio tijeka rada na izradi bilo je teksturiranje 3D modela. Substance programski paketi su izrazito moćni alati za izradu tekstura. Izrada tekstura unutar ta dva programska paketa odvija se proceduralno, što omogućuje stvaranje jedinstvenih tekstura. Izrada tekstura tako zahtjeva veliku maštu i odlično poznavanje korištenih programskih paketa.
Unreal Engine 4 je odličan, ako ne i najbolji besplatan pokretač igre kojim je moguće izrađivati izuzetno realna 3D okruženja. Velika prednost kod izrade 3D okruženja u Unreal Engine 4 je što se do velike većine mogućnosti grafičkog sustava može pristupiti uz malo ili skoro ni malo programiranja.
Rezultat projekta S.S Infinity je video animacija preleta kamere kroz 3D okruženje. Dokumentirani tijek rada može poslužiti nekome tko tek ulazi u svijet level dizajna, ali on nikako nije smjernica kako izraditi dobro 3D okruženje za računalnu igru.

6. Reference
6. References

[1] Adams, Ernest. Fundamentals of Game Design, 2nd Edition. Berkley, SAD: New Riders, 2009.

[2] McDermott, Wes The Comprehensive PBR Guide by Allegorithmic - vol. 1. California, SAD: Allegorithmic, 2017., dostupno na: https://www.allegorithmic.com/system/files/software/download/build/PBR_Guide_Vol.1.pdf, pristupano: 15.06.2017.

[3] Allegorithmic, Substance Painter, dostupno na: https://support.allegorithmic.com/documentation/display/SPDOC/Substance+Painter, pristupano: 16.06.2017.

[4] Allegorithmic, Substance Designer, dostupno na: https://support.allegorithmic.com/documentation/display/SD5/Substance+Designer+User+Guide, pristupano: 16.06.2017.

[5] Allegorithmic, Izrada normal map teksture, dostupno na: https://support.allegorithmic.com/documentation/display/SD5/Normal+Map+from+Mesh, pristupano: 16.06.2017.

[6] Epic Games, Material Editor, dostupno na: https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/Editor/, pristupano: 17.06.2017.

[7] Epic Games, Unreal Engine 4, dostupno na: https://www.epicgames.com/about, pristupano: 15.06.2017.

[8] Allegorithmic, Substance Designer pakiranje tekstura, dostupno na: https://support.allegorithmic.com/documentation/display/SD5/Packages, pristupano: 17.06.2017.

[9] Epic Games, Content Browser, dostupno na: https://docs.unrealengine.com/latest/INT/Engine/Content/Browser/index.html, pristupano: 18.06.2017.

[10] Epic Games, tipovi osvjetljenja, dostupno na: https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightTypes/index.html, pristupano: 29.06.2017.

[11] Nvidia, VXGI, dostupno na: https://developer.nvidia.com/vxgi, pristupano: 29.06.2017.

[12] Epic Games, Sequencer, dostupno na: https://docs.unrealengine.com/latest/INT/Engine/Sequencer/index.html, pristupano: 29.06.2017

Slike

Slika 14 Mreža čvorišta u Substance Designer 5
Figure 14 Node grid in Substance Designer 5
Slika 14 je autorska.

Slika 15 Uvoz 3D modela u content browseru
Figure 15 Importing 3D models in content browser
Slika 15 je autorska.

Slika 16 Izrađen materijal u material editoru
Figure 16 Finished material in material editor
Slika 16 je autorska.

Slika 17 3D model sa svim teksrurama
Figure 17 3D model with all textures
Slika 17 je autorska.

Slika 18 Žičani prikaz gornjeg prikaza S.S. Inifinity 3D okruženja
Figure 18 Top view Wireframe of S.S. Infinity 3D environment
Slika 18 je autorska.

Slika 19 Neosvjetljen modul koridora
Figure 19 Unlit coridor module
Slika 19 je autorska.

Slika 20 Voxelizirano 3D okruženje
Figure 20 Voxelised 3D environment
Slika 20 je autorska.

Slika 21 Poligonalna zona hvatanja refleksija
Figure 21 Polygonal reflection capture
Slika 21 je autorska.

Slika 22 Prikaz 3D okruženja bez gradacije boja
Figure 22 3D environment without color grading
Slika 22 je autorska.

Slika 23 Prikaz 3D okruženja s gradacijom boja
Figure 23 3D environment with color grading
Slika 23 je autorska.

Slika 24 Snimak zaslona skladišta u S.S. Infinity 3D okruženju
Figure 24 S.S. Infinity 3D environment screen capture
Slika 24 je autorska.

Slika 25 Unreal Engine 4 Sequencer
Figure 25 Unreal Engine 4 Sequencer
Slika 25 je autorska.

Slika 26 kamere s rig rail objektom
Figure 26 Attaching camera to rig rail
Slika 26 je autorska.

Slika 27 Rig rail objekt
Figure 27 Rig rail object
Slika 27 je autorska.

Slika 28 Postavljanje keyframeova za 3D modele lijevih vrata
Figure 28 Keframing left door 3D models
Slika 28 je autorska.

Slika 29 Izrađena montaža videa u Adobe Premiere CC
Figure 29 Finished video in Adobe Premiere CC
Slika 29 je autorska.

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: Damir Mačković | MATIČNI BROJ: 0297/336

DATUM: 05.03.2017 | KOLEGIJ: 3D animacija

NASLOV RADA: Tijek izrade 3D okruženja u Unreal Engine 4

NASLOV RADA NA ENGL. JEZIKU: Unreal engine 4 enviroment workflow

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

ČLANOVI POVJERENSTVA:

1. mr.sc. Dragan Matković, v. predavač - predsjednik
2. pred. Robert Geček,dipl.ing. - član
3. Andrija Bernik, pred., dipl.inf. - mentor
4. pred. Snježana Ivančić Valenko, dipl.ing.
5.

VŽ KC

MMI

Zadatak završnog rada

BROJ: 520/MM/2017

OPIS

Level dizajn je široko područje koje pokriva mnoge djelatnosti stvaranja 3D sadržaja za računalne igre, poput 3D modeliranja, teksturiranja, programiranja, osvjetljenja i stvaranja kompleksnih shader mreža unutar odabranoga pokretača računalne igre (game engine). Svaki stadij level dizajna se pomno planira kako bi rezultat bilo 3D okruženje privlačno igračima.
Cilj ovog rada je objasniti sve potrebne stadije tijeka razvoja 3D okruženja jednog levela računalne igre kao i programske pakete potrebne u pojedinim koracima razvoja.
Praktični dio rada je izrada 3D okruženja, interijera svemirskog broda. Točnije, glavnih hodnika i skladišta.
Rezultat praktičnog djela završnog rada je Animacija preleta kamere kroz 3D okruženje.

U radu je potrebno:
- teoretski objasniti Level design
- teoretski opisati korištene programske alate i njihovu uporabu
- objasniti tijek rada svih stadija izrade 3D okruženja igre
- izraditi 3D okruženje u Unreal Engine 4
- izraditi animaciju preleta kamere kroz okruženje

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER
KOPRIVNICA

ZADATAK URUČEN: 11.04.2017. | POTPIS MENTORA: Bernik

IZJAVA O AUTORSTVU I SUGLASNOST ZA JAVNU OBJAVU

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

Ja, Damir Mačković *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/~~ica~~ završnog/~~diplomskog~~ (*obrisati nepotrebno*) rada pod naslovom Ljet izrade 3D otiskača u Unreal Engine 5 (*upisati naslov*) te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Damir Mačković
(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, Damir Mačković *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom Ljet izrade 3D otiskača u Unreal Engine 5 (*upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Damir Mačković
(vlastoručni potpis)

Sveučilište
Sjever

Završni rad br. 520/MM/2017

Tijek izrade 3D okruženja u Unreal Engine 4

Damir Mačković, 0297/336

Varaždin, lipanj 2017. godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 520/MM/2017

Tijek izrade 3D okruženja u Unreal Engine 4

Student

Damir Mačković, 0297/336

Mentor

dr.sc. Andrija Bernik, pred.

Varaždin, lipanj 2017. godine

Predgovor

Zahvaljujem tvrtkama Autodesk, Allegorithmic software i Epic Games što su mi ustupili besplatne studentske licence za svoje programske pakete Maya, Substance Painter2, Substance Designer 5 i Unreal Engine 4 te mi tako omogućili izradu ovog završnog rada. Želim se zahvaliti Petri Gal na velikoj pomoći pri ispravljanju grešaka i prevođenju. Također želim se zahvaliti Marku Klobučaru, Magdaleni Ledinšćak, Mariji Magdaleni Čizmadiji, Ani Medvedec, Tjaši Ogrinec i ostalim poznanicima što su me podupirali u proučavanju područja izrade 3D sadržaja, njihova potpora je uvijek bila prisutna u pravo vrijeme.

Sažetak

Kako bi bilo moguće izraditi 3D okruženje za računalnu igru pokretanu Unreal Engine-om 4 potrebno je prvo proučiti klasične igre te njihove elemente. Također potrebno je razumjeti industriju računalnih igara te uloge dizajnera igara i level dizajnera. Igre, industrija računalnih igara te uloge dizajnera igara i level dizajnera opisani su u teorijskom dijelu rada. Za vrijeme rada na projektu S.S. Infinity osmišljen je tijek rada koji je dokumentiran u ovom radu. Dokumentirani tijek rada uključuje opise izrade 3D modela, teksturiranja, pakiranja tekstura za Unreal Engine 4, izrade 3D okruženja u Unreal Engine 4, postavljanje osvjetljenja, post proces efekata i izrade animacije. Izrađeno 3D okruženje je u početku izrađeno za računalnu igru, ali je zbog nedostatka vremena iskorišteno za izradu video animacije za potrebe kolegija Video animacija. Video animacija pokazuje rani stadij razvoja računalne igre S.S. Infinity.

Ključne riječi: 3D okruženje, 3D modeliranje, igre, level dizajn, teksturiranje, Unreal Engine 4.

Popis korištenih kratica

ENG Engleski jezik

PBR Physically Based Rendering
Fizički točan računalni prikaz

LUT Lookup table
Datoteka koja sadrži podatke o gradaciji boje u slici.

MMO Massive multiplayer online game.
Računalna igra koja uključuje više igrača u online okruženju

3D Trodimenzionalan

2D Dvodimenzionalan

MEL Maya Embedded Language
Skriptni jezik koji služi za pojednostavnjivanje operacija unutar Autodesk Maya

Sadržaj

1. Uvod

1.1. Uvod u računalne igre i njihov razvoj

Računalne igre su medij koji kroz integraciju drugih medija pružaju igraču osjećaj uključenosti i zadovoljstva kroz zabavu. Kako bismo računalne igre razumjeli, potrebno je prvo razumjeti elemente koji tvore neku igru. Bitno je znati da se računalne igre u svojoj osnovi ne razlikuju previše od običnih, klasičnih igara. One posjeduju identične elemente kao klasične igre. Računalne igre sve te elemente objedinjuju kroz skup različitih medija dajući pritom igraču osjećaj slobode. Za razliku od primjerice filma ili glazbe, računalne igre isporučuju sadržaj na interaktivan način kojim se igraču svijet igre prezentira kao nova stvarnost. Većina 3D računalnih igara današnjice oslanja se na vizualni izgled 3D okruženja u kojem se radnja igre odvija, jer ono je jedan od presudnih faktora u stvaranju osjećaja druge stvarnosti kod igrača. Upravo kvaliteta izgleda 3D okruženja neke računalne igre može biti presudan faktor koji će privući ciljanu publiku, tj. određeni tip igrača..

Za vizualni izgled, funkcionalnost 3D okruženja i iskustvo igranja (eng. Gameplay) u računalnim 3D igrama zaslužni su level dizajneri (eng. Level Designer). Oni na osnovu suradnje s producentom, umjetničkim redateljem i dizajnerom igre implementiraju sadržaje igre, i u konačnici izrađuju 3D okruženja. Level dizajneri također odlučuju koji će elementi 3D okruženja biti korišteni u interakciji s igračem kako bi što više doprinijeli njegovom iskustvu igranja. Razvojne kuće često zapošljavaju i po nekoliko level dizajnera, ovisno o opsegu projekta. Tijek razvoja računalne igre tako u osnovi ovisi o odlukama četiri ključne osobe u razvojnoj kući: o producentu, umjetničkom redatelju, dizajneru igre i level dizajneru. Level dizajneri su često osobe koje i same stvaraju sadržaj igre (3D modele i teksture, animacije, i drugo). No, većinom se uvelike oslanjaju na umjetničke sposobnosti 3D umjetnika (eng. 3D artist). 3D umjetnik je osoba koja je direktno povezana s izgledom elemenata koji tvore 3D okruženje neke video igre. To je osoba koja posjeduje veliko iskustvo u radu s programskim paketima namijenjenim za 3D modeliranje i teksturiranje pa i animaciju. 3D često umjetnik dobije zadani koncept prema kojem treba stvarati sadržaj, ali nerijetko mu je omogućena velika sloboda u procesu izrade istog.

1.2. Cilj završnog rada

Cilj ovog rada je u osnovama pojasniti elemente igara, te uloge dizajnera igre i level dizajnera u procesu razvoja računalne igre, pri tom fokusirajući se na izradu 3D okruženja. Također bih kroz praktični rad volio objasniti sve potrebne stadije u razvoju 3D okruženja iz aspekta 3D umjetnika.

Ovaj rad se fokusira na tijek izrade 3D okruženja za igru pokretanu Unreal Engine 4. Radni tijekovi proizvodnje nekog 3D sadržaja pa i okruženja su različiti na osobnoj i na razini tvrtke koja igru razvija. Tijekom izrade 3D okruženja za projekt „S.S. Infinity“ razvio sam svoj tijek rada kako bi olakšao izradu i što bolje organizirao vrijeme. Tijek rada dokumentiran je u ovom radu, i iako on može poslužiti nekome tko tek ulazi u svijet level dizajna, on nikako nije smjernica kako izraditi dobro 3D okruženje računalne igre.

2. Igre

Prije nego bismo mogli početi govoriti o level dizajnu ili dizajnu igara te izradi 3D okruženja računalnih igara, potrebno je odrediti što igre jesu i kako funkcioniraju. Lako je za pretpostaviti kako svatko zna što je igra, ali danas postoje toliko različitih vrsta igara pa je najbolje ne raditi pretpostavke osnovane samo na osobnom iskustvu. Za razumijevanje igara ključno je identificirati potrebne elemente koje neka igra mora posjedovati, zatim definirati što igra je bazirajući se na tim elementima. Tek kada razumijemo elemente igre, onda možemo raspravljati o računalnim igrama i njihovim prednostima nad klasičnim igrama. [1]

2.1. Elementi igre

Igre su proizašle iz ljudske želje za zabavom i iz sposobnosti za pretvaranjem. Igranje igara spada u kategoriju neobavezne, i obično rekreativne ljudske aktivnosti koja često ima i socijalan značaj. Pretvaranje je jedan od elemenata igre koji se smatra mentalnom sposobnošću uspostavljanja druge stvarnosti, za koju onaj koji se pretvara zna da je različita od stvarnog svijeta. To je stvarnost koju onaj koji se pretvara može u bilo kojem trenutku stvoriti, mijenjati ili napuštati i ponovno se u nju vraćati. Igranje i pretvaranje su osnovni elementi igranja igara. Oba elementa su intenzivno proučavana kao kulturološki i psihološki fenomen, te su samo jedni od četiri osnovna elementa koja tvore neku igru. Četiri osnovna elementa igre su dakle *igranje, pretvaranje, cilj* i *pravila*. [1]

2.1.1. Igranje

Igranje se smatra elementom igre koji se može definirati kao uključujući oblik zabave, jer za razliku od filma ili knjiga koji su prezentacijske prirode, igranjem igre onaj koji je uključen u igru zabavlja sam sebe. Za usporedbu, iskustvo čitanja sadržaja neke knjige neće se promijeniti neovisno koliko puta knjigu pročitali, dok kada igramo neku igru mi sami odlučujemo slijed događaja i tako utječemo na iskustvo konzumiranja njenog sadržaja. S obzirom na to da svaki puta kada igramo igru mi donosimo različite odluke, u mogućnosti smo dobiti svaki puta drugačije iskustvo. Igranje igara u konačnici uključuje slobodu djelovanja i izbora. Ta sloboda nije neograničena, dakako. Sve mogućnosti izbora unutar igre ograničene su pravilima koja od igrača traže da bude pametan, domišljat i vješt u igranju igre.[1]

2.1.2. Pretvaranje

Pretvaranje je čin stvaranja zamišljene alternativne stvarnosti. Drugi naziv za stvaranje zamišljene alternativne stvarnosti je „čarobni krug“. Čarobni krug ideja koju je definirao Danski povjesničar Johan Huizinga u svojoj knjizi *Homo Ludens*. Čarobni krug je definirao kroz koncept imaginarnih svjetova koji se javljaju kroz fikciju. Za čarobni krug Huzinga vjeruje kako se može primijeniti na ritualne, duhovne i pravne aktivnosti.[1,2]

Kada govorimo o igrama, Huzingin koncept čarobnog kruga možemo primijeniti kako bismo definirali granicu između ideja i aktivnosti koje su značajne u igri od ideja i aktivnosti u stvarnom svijetu. U osnovi, možemo njegov koncept čarobnog kruga koristiti kao definiciju granice između zamišljenog svijeta igara i stvarnosti. Za vrijeme igre igrači se mogu pretvarati do te granice da ideje koje postoje unutar igre ne moraju odgovarati ostvarivosti u stvarnom svijetu. Primjerice, u igri se možemo pretvarati da putujemo brzinom svjetlosti, i iako znamo da to nije moguće u stvarnom svijetu, za vrijeme igre nam to itekako djeluje stvarno.

Za vrijeme igre, unutar čarobnog kruga igrači se vežu za privremeni, umjetni značaj situacija i događaja koji se odvijaju za vrijeme igre. Čarobni krug počinje postojati onda kada se igrači zajedno uključe u igru, drugim riječima kada pristanu ponašati se u skladu pravilima igre. Čarobni krug nestaje u trenutku kada igrači napuste igru ili kada igra završi.[1]

Slika 1.1. Čarobni krug [2]

U igrama koje uključuju samo jednog igrača, igrač sam uspostavlja čarobni krug samim činom što je odabrao igrati igru. U društvenim igrama, svi igrači se u dogovoru složno pretvaraju iste stvari, tj. prihvaćaju sva pravila igre. Iako se pretvarana stvarnost jako uživljenom igraču može itekako činiti stvarna, ona vrlo lako može prestati postojati u trenutku kada se jedan od igrača prestane ponašati po pravilima i počne odbijati igrati dalje, kršeći tako društveni dogovor. Uzmemo li primjerice sportsku igru nogomet, na prvu pomisao pomislili bi kako u toj igri nema puno pretvaranja. Igrači se ne pretvaraju da su netko drugi i njihove aktivnosti se događaju u stvarnom svijetu. Ali igrači itekako uspostavljaju alternativnu stvarnost, i stvari koje se odvijaju tijekom igre nogomet igračima su itekako značajne, što je sami čin pretvaranja. Van čarobnog kruga čin udaranja lopte nogom s ciljem da uđe kroz metalni okvir s mrežom nema neki dodatni značaj. Ali za vrijeme igre igrači i gledatelji se pretvaraju da je udaranje lopte u mrežu dobra stvar i da to doprinosi timu koji to postigne. Ova usporedba s nogometom ilustrira ponašanje u skladu s pravilima igre za vrijeme pretvaranja dok igra traje.[1]

Granica između čarobnog kruga i stvarnog svijeta nije uvijek tako kristalno jasna. Ako su događaji unutar igre značajni i u stvarnom svijetu, tada čarobni krug postaje mutan. Primjerice, igre na sreću ili kockanje zamućuju granicu čarobnog kruga sa stvarnošću. Kada igrač kocka, on se kladi na pravi novac kako bi utjecao na ishod igre. Proces kockanja može ili ne mora biti unutarnji element igre. U jednu ruku, igrač može odabrati igrati domino za novce, šibice, ili za ništa. U drugu ruku kartaška igra poker u kasinu ima ulog novca kao unutarnji element igre i ako ne uložimo novac, ne možemo sudjelovati u igri. [1]

2.1.3. Cilj

Igre moraju imati cilj. Često ih imaju i nekoliko. Čak i kreativne, ne kompetitivne igre imaju cilj, stvaranje. Cilj igre je definiran pravilima i proizvoljan je zato što dizajneri igre mogu definirati cilj proizvoljno kako oni to žele. Cilj igre ne bi trebao biti trivijalan zato što igra mora sadržavati element izazova. Čak i igre na sreću poput ruleta tjeraju igrače da nauče shvatiti izglede i da u skladu s time ulažu svoje uloge koji bi im mogli donijeti profit. Slično tome, kreativne igre tjeraju igrače na kreativnost. Kako bi igrač bio dobar u igri, potrebna mu je vještina. Ako se cilj igre može ostvariti u jednom trenutku, bez prevelikog fizičkog ili mentalnog napora, tada se aktivnost koju igrač obavlja i nije igra. Primjer takve jedne igre je dječja igra par-nepar. Dvoje djece bacaju identičan novčić, i ako oba padnu na različite strane, jedno dijete uzima sav novac. Ako oba padnu na istu stranu, drugo dijete dobiva oba novčića.

Vjerojatnost je 50% i ne postoji način kako je poboljšati, tj. u toj igri ne postoji nikakvo donošenje odluka. Zbog toga se ta igra zapravo ne kvalificira kao igra, već oblik kockanja zbog svog trivijalnog cilja. [1]

Pravila igre često karakteriziraju cilj igre kao stanje pobjede – nedvosmislenu situaciju unutar igre za vrijeme koje jedan ili više igrača odnosi pobjedu. Primjerice, stanje pobjede u šahu diktira pobjednika u slučaju kada jedan igrač matira kralja protivničkog igrača (nedvosmislena situacija). U sportskim igrama koji su određeni vremenom, kao što je npr. košarka. Stanje pobjede nalaže da kada vrijeme istekne (nedvosmislena situacija), pobjedu odnosi tim s najviše bodova. Koncept pobjeđivanja i gubljenja u igrama nije nužno potreban igrama, ali ih čine uzbudljivijima. Igra mora imati cilj, ali taj cilj ne mora biti okarakteriziran pobjedom ili porazom. Pravila i cilj igre su u potpunosti zatvoreni u čarobnom krugu, ali koncept pobjeđivanja i poraza prelaze i u stvarni svijet. Pobjeđivanje se percipira kao zasluženo zadovoljstvo, i nakon što je igra završila igrači su ponosni na pobjedu. Pobjeđivanje u igrama igračima može također donijeti i materijalnu dobit. Koncept pobjede i poraza ne moraju nužno biti uključeni u igru, oni su sporedan element igre koji čine igru uzbudljivom i značajnijom igračima.[1]

2.1.4. Pravila

Pravila igre su definicije i upute koje igrači slijede za vrijeme igre. Svaka igra ima pravila, čak i kada su ta pravila ne zapisana ili se uzimaju zdravo za gotovo. Pravila u igrama imaju nekoliko funkcija. Ona uspostavljaju cilj igre i značenja različitih aktivnosti i događaja koji se događaju unutar čarobnog kruga. Ona također stvaraju kontekstualni okvir koji omogućuje igračima razlikovanje aktivnosti koje su dopuštene za vrijeme igre, a koje nisu. Pravila također omogućuju igračima prepoznavanje najpovoljnijih tijekova događaja kako bi ostvarili cilj igre. Pravila između ostalog definiraju: [1]

- **Semiotiku igre** - značenja i veze između različitih simbola unutar igre. Neki semiotički simboli unutar igara mogu biti apstraktni, poput izlasci i ulasci igrača u američkoj igri *baseball.* Drugi simboli, poput vojske u stolnoj igri *Rizik,* imaju paralelu u stvarnom svijetu koja nam pomaže da ih shvatimo. Semiotika sama po sebi pa i semiotika unutar igara je jako kompleksno i široko područje, te izlazi izvan opsega ovog rada, pa u ovom radu neće biti šire obrađena.

- **Iskustvo igranja (eng. Gameplay) -** kojeg osim pravila definiraju izazovi i djelovanja igrača unutar igre.
- **Slijed igranja** - koji označava napredak aktivnosti unutar igre.
- **Cilj igre**.
- **Meta pravila**. To su pravila o pravilima koja mogu ukazivati u kojim se situacijama pravila mijenjaju ili kada su odstupanja od njih dopuštena.[1]

Pravila bi trebala biti definirana nedvosmisleno kako bi se izbjegli konflikti kod interpretacije. Pravila bi također trebala biti pisana tako da se ne kose s ostalim pravilima u igri. Ako postoji mogućnost da konflikti između pravila nastanu, potrebno je definirati meta pravilo koje bi odredilo koje pravilo prevladava. Dvosmislena i konfliktna pravila su znak lošeg dizajna igara i potrebno ih je izbjegavati. [1]

Prilikom dizajniranja igara potrebno je dakle imati širok pogled na igre, i gledati igre kao ljudsku aktivnost, a ne samo kao skup pravila. Iako sve igre moraju imati pravila, sama pravila ne čine igru. Kako bi igra mogla postojati, mora biti igrana. Inače je samo teoretska apstrakcija. Ako gledamo igre kao ljudsku aktivnost, fokusiramo se na igrače – osobe za koje je igra napravljena. Ljudi većinom igraju igre iz zabave, ali ih ponekad igraju kako bi nešto naučili, ili je koristili kao oblik učenja neke nove sposobnosti u stvarnom životu. U tom kontekstu, definicija igranja igara postaje pomalo mutna. Dobre igre ljudi percipiraju kao zabavne, dok one loše ne. Zabava je emocionalna reakcija na igranje igara i nije unutarnji element igre. Sama činjenica da je neka igra nekome nije zabavna ne znači da je ona loša. Zabava je preuzak koncept kako bi se opisalo ono što igre mogu raditi za igrače.[1]

2.1.5. Natjecanje i suradnja

Osnovni elementi igre definiraju igru, bilo ona računalna ili klasična igra u fizičkom obliku. Svaki od elemenata unutarnje je svojstvo igara bez koje igre ne mogu funkcionirati. Igre mogu pored osnovnih elemenata sadržavati i dodatne elemente koje ih mogu ali i ne moraju činiti boljima. Jedna od dodatnih elemenata igara su natjecanje i suradnja igrača unutar igre. Natjecanje se javlja kada se suoče igrači različitih interesa, tj. kada igrači žele postići uzajamno ekskluzivni cilj ili ciljeve unutar igre. Suradnja igrača se javlja kada više igrača želi postići iste

ili povezane ciljeve. Igrači koji žele postići različite ciljeve, oni se ne natječu niti surađuju pa skladu s time niti ne igraju istu igru.[1]

Igre koje uključuju više igrača uključuju različite načine natjecanja kroz oblike:

- **Natjecanje dvoje igrača** ili „ti protiv mene“. Taj način natjecanja između igrača je najbolje znan jer se nalazi u najstarijim igrama poput šaha.
- **Natjecanje više igrača** ili „svatko za sebe“. Takav način natjecanja nalazimo u kartaškim igrama, stolnim igrama poput monopola i brojnim sportskim igrama.
- **Suradnja više igrača** ili „svi mi zajedno“. Ovakav tip suradnje u igrama javlja se uslijed težnje svih igrača ka istom cilju.
- **Timska suradnja** ili „mi protiv njih“. Takav tip suradnje javlja se kada se igrači grupiraju u timove s ciljem da poraze protivnički tim.
- **Igra jednog igrača** ili eng. single-player. Npr. kao u kartaškoj igri pasijans ili računalnim igrama dizajniranim posebno za ovakav tip igranja, poput Super Mario računalne igre.[1]

Mnogo računalnih igara omogućuje igračima na samom početku igre odabir načina u kojem žele igrati. Najčešće je to igra za jednog igrača, natjecanje više igrača ili nešto drugo. Takav izbor načina igranja igre proširuje tržište takvih igara ali istovremeno povećava opseg projekta izrade igre. Najčešće prilikom dizajniranja igre, razvojni timovi fokusiraju svoj trud ka jednom načinu igranja igre, a dodaju druge tek nakon. Primjer tome su brojne igre koje su u načinu za jednog igrača bile iznad prosječno uspješne, a u načinu za više igrača to nikako nisu uspijevale ponoviti. Razlog tome leži u odluci razvojnog tima da se sav napor razvoja usmjeri ka jednom načinu. Manji razvojni timovi često imaju ovakav pristup jer često nemaju dovoljno velik budžet kako bi ostvarili sve svoje ideje. Također često je prisutan i manjak iskustva u takvim manjim razvojnim timovima i osobne preferencije ljudi u timu, koji su faktor koji na posljetku odredi smjer kojim razvoj igre krene teći. [1]

2.2. Računalne igre

Računalne igre su poseban dio svijeta igara. Računalne igre odvijaju se posredstvom računala, bilo ono malo poput današnjih pametnih telefona ili veliko poput stolnog osobnog računala. Računala omogućuju računalnim igrama posuđivanje tehnika zabavljanja od drugih medija poput knjiga, filma i dr. Računalne igre donose zabavu kroz spoj različitih medija, pri tom uzimajući od svakoga po nešto. To posuđivanje i integracija elemenata iz drugih medija je upravo razlog zašto su računalne igre toliko privlačne igračima. [1]

Za razliku od konvencionalnih igara, računalne igre ne zahtijevaju pisana pravila. Računalne igre i dalje imaju pravila, ali ih računalo implementira i provodi za igrače. Igrači ne trebaju točno znati koja su pravila igre, iako im se mora reći kako da igraju igru. U većini računalnih igara, računala postavljaju granice čarobnog kruga zato što su postupci igrača za vrijeme igre značajni jedino ako ih računalo može detektirati kroz ulazne uređaje. Računalo također određuje kada je cilj postignut. Računalo također određuje stanje pobjede i poraza ako ti koncepti postoje u programskom kodu igre. To znači da igrači više ne moraju razmišljati o igri kao igri. Igrač koji u jednom trenutku igranja igre planira neki postupak jednostavno može odmah isprobati to što želi napraviti, bez da mora čitati sva pravila igre da vidi da li je to što želi dopušteno unutar igre. Time se omogućilo igračima da se bolje užive u igru, da je ne vide samo kao privremeno umjetno okruženje s proizvoljnim pravilima, već kao alternativni svijet čiji dio postaju. [1]

Nemogućnost sagledavanja pravila tijekom igre u računalnim igrama ipak ima jedan nedostatak. Taj nedostatak krije se u tome što ako igrači ne vide pravila tijekom igre onda ne mogu racionalizirati svoje izbore. Igrači mogu naučiti pravila jedino igranjem igre jer računalne igre često uključuju male upute koje navode igrača kako bi trebao igrati igru i što tijekom igre može očekivati. Iako, poneke računalne igre tjeraju igrača da uči metodom pokušaja i pogreške, što igru može učiniti poprilično frustrirajućim. Jedan primjer je serijal računalnih igara Dark Souls. Mnogo igrača voli igrati taj serijal, ali većina njih ne voli učiti igrati igru tako, što značajno limitira tržište na igrače koji mogu tolerirati takvu frustraciju tijekom igranja. [1]

Računalne igre su software kojim dizajneri igre grade model simuliranja ponašanja unutar igre. Pisanje računalnog koda u mnogo tome se razlikuje od primjerice, pisanja proze ili fotografiranja ili snimanja videa. Računalni kod računalnih igara postavlja model mogućih izgleda događaja od kojih su svi podložni pravilima igre. Jedan od naziva za ovakav način reprezentiranja je *proceduralnost*. *Proceduralnost* u računalnim igrama je naziv za pojam koji označava mogućnost računala da izvodi model mogućih ponašanja igrača unutar igre. Računalne igre su reprezentacija tog modela. Stoga u računalnim igrama postoji izvorni sustav, koji dolazi iz stvarnog svijeta i računalni proceduralni model tog sustava. Igrač stoga mora vršiti interakciju s tim modelom kako bi igra mogla funkcionirati. Kada igrač igra igru, on ima neku predodžbu o modeliranom sustavu i sustavu po kojem je napravljen model. Tu predodžbu igrač stvara na osnovu načina na koji je izvorni sustav simuliran modelom. S obzirom na to da igrači stvaraju predodžbe na osnovu modela koji je nastao prema izvornom sustavu, dizajneri igara imaju velike mogućnosti prilikom stvaranja računalnih igara. Primjerice, jedan dizajner igara može stvoriti sportsku igru o strategiji igre, dok drugi dizajner igre može stvoriti igru o dužnostima pojedinih igrača unutar te iste sportske igre. Mogućnosti dizajnera igre da naprave različite igre temeljene na konceptima iz stvarnog svijeta naglašava činjenicu kako izvorni sustav zapravo i ne postoji kao takav. To je tako zato što je doživljaj igrača o nekoj igri u stvarnom svijetu izrazito subjektivan. Računalne igre nasljeđuju tu subjektivnost jer stvaraju disonance, razdore između dizajniranoga proceduralnog modela i igračevog subjektivizma. Zbog toga su igre izrazito izražajne. One potiču igrače da propitkuju stvarnost koja ih okružuje i stvarnosti koje su modelirane u računalnim igrama. [3]

Većinu vremena računalne igre stvaraju proceduralne modele izmišljenih života, kao na primjer životi profesionalnih igrača u nogometnim igrama, ili pak noćnih vilenjaka u igri Warcraft 3 razvojne kuće Blizzard. Tim izmišljenim životima likova u igrama dizajneri računalnih igara pozivaju igrače da vide običan svijet u kojem se nalaze na sasvim novi način. Jedan od razloga zašto se dizajneri igre odlučuju na ovakav način prezentacije igara igračima je za persuaziju ideja o stvarnom svijetu i načinu na koji funkcionira. O raznim oblicima persuazije postoje mnoge teorije koje izlaze van opsega ovog rada. [3]

2.2.1. 3D okruženje računalnih igara

Računalne igre mogu prezentirati svijet igre u dvije ili tri dimenzije. Iako 2D igre mogu biti itekako zabavne i zanimljive, te po elementima ne toliko različite od 3D igara, u ovom radu fokusirat ću se većinom na 3D igre i njihovo okruženje.

Zbog prirode 3D računalnih igara, one uključuju imaginarne svjetove koje mogu sadržavati imaginarne ljude, mjesta i situacije. Igrači se mogu poistovjetiti s likovima i u određenom trenutku izgubiti osjećaj stvarnog svijeta. Kod konvencionalnih igara, to poistovjećivanje sa svijetom igre događa se u igračevoj mašti. U jednom smislu se tako može reći kako je svijet konvencionalnih igara koncipiran u glavama igrača. 3D Računalne igre, u drugu ruku, mogu prezentirati direktnije fikcionalne svjetove igračima. Baš kao što filmovi mogu prezentirati fikcionalni svijet stvarnije od knjiga. Korištenjem računalnog ekrana i zvučnika, 3D računalne igre prezentiraju fikcionalan svijet tako da je malo toga ostavljeno mašti. Do praktički nedavno, slabija razvijenost računalne grafike je značila da su igrači puno morali upotrebljavati svoju maštu tijekom igre. Prezentiranje što realnijih fikcionalnih svjetova 3D računalnih igara je od uvijek bio cilj razvojnih timova računalnih igara. Svaka nova 3D računalna igra sa sobom donosi novi dašak realizma. Što se više igra čini realnija, tj. što je realnije njeno 3D okruženje to više privlači igrače. Posebice ako se radi o žanrovima iz prvog lica ili ako se radi o simulacijama raznih sportova. Današnje moderne igre su pune slika, animacija, filmova, glazbe, dijaloga, zvučnih efekata, itd; što konvencionalne igre jednostavno ne mogu pružiti. 3D okruženja računalnih igara su u zadnjih par godina postala toliko foto realistična da pojedini dizajneri igara eksperimentiraju s raznim umjetničkim pravcima. [1]

Na samim rubovima današnje tehnologije 3D računalnih igara, razvojni timovi rade igre s poboljšanom stvarnosti i miješanom stvarnosti, u kojima se računala koriste u spoju s aktivnostima iz stvarnog svijeta kako bi se ostvarilo iskustvo igranja. Takve igre često se razvijaju za moderne pametne telefone, video kamere, GPS sustave, pa čak i za web poslužitelje. Postoje mnogo programskih alata za izradu 3D okruženja poput Unreal Engine 4 ili Unity 5 kojima se level dizajneri služe, koji im omogućuju razvoj za više platforma od jednom. [1]

2.2.2. Pokretač računalne igre (eng. game engine)

Sve računalne igre odvijaju se u stvarnom vremenu. Kako bi cijela mehanika igre mogla funkcionirati, potreban je programski alat koji to omogućuje. Mehanika igre određuje parametre svijeta igre koji funkcionira sam za sebe bilo da igrač vrši interakciju s njim ili ne. Cijela mehanika igre mora funkcionirati kontinuirano na produljeno vrijeme. Primjerice likovi s umjetnom inteligencijom moraju funkcionirati sami za sebe, zamke u igri moraju znati da li se igrač nalazi pored njih, banke moraju funkcionirati potpuno neovisno o igraču, itd. [4]

Programski alati koji se koriste za izradu 2D i 3D računalnih igara zovu se pokretači igara (eng. game engine). Moderni pokretači igara pružaju razvojno okruženje za prikaz elemenata igre koji reagiraju na igračeve akcije. Pokretači igara također omogućuju funkcioniranje umjetne inteligencije te između ostalog, simuliranje fizike i indirektnog osvjetljenja. U industriji računalnih igara postoje mnogo različitih pokretača igre jer skoro svaki veći i ozbiljniji studio razvije svoj. Iako u industriji računalnih igara postoje popularni pokretači igre poput Unreal Engine 4 studija Epic Games, ne postoji dogovorom određen standardan pokretač igre.

Osoba koja provodi vjerojatno najviše vremena koristeći se pokretačem igre je level dizajner. Pokretač igre omogućuje mu ostvarivanje željenog 2D ili 3D okruženja i njegove atmosfere. Pokretač igre je moćan programski alat kojim je moguće izrada ranog prototipa 2D ili 3D okruženja koji onda može služiti za testiranje rane verzije razine igre. Testiranje je važno jer daje uvid u proces razvoja igre kod kojeg vrlo lako stvari mogu krenuti nizbrdo ako se na vrijeme ne rade testovi. Pokretači igre omogućuju rano igranje računalne igre, iako još ne postoje svi elementi koji bi u potpunosti činili tu igru. Rano igranje unutar editora pokretača igre kasnije donosi veliku uštedu vremena i novca. Level dizajner tako može s lakoćom znati koje elemente treba ubaciti u okruženje koje dizajnira, a koje bi bilo bolje izbaciti. Kada stadij razvoja neke računalne igre dosegne razinu gdje više nema povratka, mogućnost ranog testiranja računalne igre unutar pokretača igre za vrijeme razvoja, vrlo često se pokaže kao najvrjednije sredstvo koje jedan razvojni studio može imati.[3]

2.2.3. Svijet igre

Igre zabavljaju kroz iskustvo igre, ali mnogo igara također zabavlja tako što odvode igrača na imaginarno mjesto – svijet igre. Iskustvo igre kod mnogih 3D računalnih igara diktiraju interakcije igrača između njega i svijeta igre. Svijet igre je umjetni svemir, imaginarno mjesto u kojem se igra odvija. Kada igrač uđe u čarobni krug (opisan u ranijim poglavljima) i kada se pretvara da je netko drugi, svijet igre je mjesto gdje se nalazi igrač. Većina računalnih igara prezentira svoj svijet igre sa slikama i zvukom; umjetnošću, animacijama, glazbom, i zvučnim efektima. Nemaju sve igre vizualne ili audio komponente. Svjetovi igara su puno više no skup slika i zvukova. Oni sadrže svoju kulturu, estetiku, skup moralnih vrijednosti i druge elemente koji ih sačinjavaju. Svijet igre također ima vezu sa stvarnošću, bilo da je izrazito apstraktan, s malo sličnosti sa stvarnim svijetom ili visoko reprezentativan i što sličniji stvarnome svijetu. [1]

Igre zabavljaju na nekoliko načina: iskustvom igranja, pričom, socijalnom interakcijom (u slučaju igre za više igrača), itd. U stolnoj igri poput šaha, malo ljudi ima predodžbu kako se u toj igri prezentira svijet srednjovjekovne bitke. Za drugi primjer navodim igru *Monkey Island* u kojoj je svijet skoro pa glavni element igre, bez njega bi ta igra bila sasvim drugačija. Kao glavno pravilo kod igara, što više igrač zna o mehanici igre i načinu na koji igra funkcionira, svijet igre mu postaje manje bitan. Kada igrač ovlada mehanikom igre, to od njega traži apstraktno razmišljanje, prilikom čega mu imaginarni svijet može biti distrakcija. Kada igrači popularne računalne igre *Counter strike* postanu izrazito vješti u igranju igre, oni više ne razmišljaju o tome kako se pretvaraju da su vojnici ili teroristi; oni su fokusirani samo na skrivanje, kretnju, pucanje, zasjedu, zauzimanje idealne pozicije, itd. Takav način apstraktnog igranja igre, prilikom čega se ignorira svijet igre događa se samo kod izrazito iskusnih igrača. Za nekoga tko je tek prvi puta otvorio igru *Counter Strike* , svijet igre je vitalan kako bi se stvorio zadržao njegov interes. [1]

Druga svrha svijeta igre je da proda igru na prvom mjestu. Često mehanika igre je nedovoljna kako bi privukla kupce igara da je pokupe s police u dućanu i kupe je. Najčešće igrače privuče prelijepo 3D okruženje svijeta igre i fikcija koje ono proizvodi. 3D okruženja zahvaljujući današnjem hardware-u i alatima za proizvodnju 3D sadržaja, s lakoćom daju igračima osjećaj stvarnosti prostora. S realnijim 3D okruženjima, igrači imaju puno više mogućnosti za istraživanje svijeta igre i to je ono što ih u velikoj mjeri privuče da kupe igru.

3. Dizajn igara (eng. game design)

Igre su integralni dio svih ljudskih kultura. Računalne igre, u svim svojim različitim formatima i žanrovima su samo novi način izražavanja kroz tu drevnu metodu društvene interakcije, igara. Stvoriti dobru igru je jako zahtjevan proces koji zahtjeva zaigran, ali sistematičan pristup. Dizajneri igara u isto vrijeme moraju biti dijelom zabavljači, dijelom matematičari i dijelom socijalni redatelji. Uloga dizajnera igara je da stvara i postavlja skupove pravila unutar kojih se nalaze načini i motivacija za igrom. Bilo da pričamo o narodnim igrama, stolnim igrama, arkadnim igrama ili o MMO (eng. massive multiplayer online game) igrama, umijeće dizajna igara je od uvijek bilo stvoriti kombinaciju izazova, natjecanja i interakcije koju igrači zovu zabavom. [3]

Kulturološki utjecaj računalnih igara je narastao toliko da se uspješno nadmeće s televizijom i filmom. Prihodi industrije računalnih igara su narasli toliko da se broje u osmeroznamenkastim brojkama prestigavši tako filmsku industriju. Činjenica je da su računalne igre druge po popularnosti iza televizije. Kako je prodaja računalnih igara rasla, interes za dizajnom igara je slijedio je taj rast. Eksplozija porasta zanimanja ljudi za dizajn igara sličan je rastu interesa za pisanjem scenarija koji je pratio rast filmske industrije. Kreativni ljudi današnjice se okreću računalnim igrama, jer one su novi način izražavanja. Na brojnim sveučilištima u svijetu postoje studiji dizajna igara kao rezultat potražnje studenata. [3]

3.1. Proces dizajna igara

Od kad postoje igre, postoje i dizajneri igara. Prvi dizajneri igara možda se nisu smatrali dizajnerima igara, možda su jednostavno tražili nešto čime će se zabaviti tijekom dosadnih kišnih dana ili su jednostavno htjeli zabaviti svoje prijatelje tako što su izmišljali razna natjecanja koristeći objekte koji ih okružuju. Takve jednostavne igre nastale davno su se igrale tisućama godina. Iako povijest igara seže daleko do samih početaka ljudske kulture, kada pomislimo na igre mi često govorimo o računalnim igrama koje su nam nedavno zaokupile maštu. Računalne igre imaju mogućnost odvesti nas do zadivljujućih novih svjetova i do fascinantnih likova i također do impresivnih 3D okruženja.[3]

Računalne igre razvijaju profesionalni timovi specijalizirani za razvoj računalnih igara koji provode mnogo sati radeći na specijaliziranim zadacima. Tehnološki i poslovni aspekti razvijanja računalnih igara su zadivljujući. Iako privlačnost igrača ka računalnim igrama vuče svoje korijene još iz jednostavnih ljudskih impulsa i poriva. Ljudi igraju igre kako bi stekli nove vještine, kako bi dobili osjećaj postignuća, kako bi se družili s prijateljima i obitelji, dok ponekad igre ljudi igraju iz razbibrige. Kako bi dizajner igre mogao dizajnirati kvalitetnu igru, mora se pitati zbog čega on igra igre. Tek kada dizajner igre razumije sam svoj odgovor na to pitanje, i odgovor drugih igrača, tek onda može početi dizajnirati igre.[3]

Prema knjizi autorice *Tracy Fulllerton, Game design workshop: a playcentric aproach to game design*, postoje tri osnovna koraka u pristupu dizajniranja dobrih računalnih igara.

1. korak

Razumjeti kako igre funkcioniraju. Potrebno je dakle razumjeti njihova pravila, procedure, ciljeve i dr. Treba se zapitati što je to igra, što je čini privlačnom za igranje.[3]

2. korak

Potrebno je naučiti izrađivati koncepte, prototipe igre i također testirati te igre igranjem istih u ranim stadijima razvoja. Također bilo bi dobro rani prototip dati igračima kako bi dobili korisne povratne informacije. [3]

3.korak

Razumjeti kako industrija igara funkcionira i mjesto dizajnera igara u toj industriji.[3]

Neki ljudi smatraju kako je dizajn igara vrsta umjetnosti, proces uporabe mašte kao neiscrpan izvor kreativnosti. Dizajneri igara također se smatraju umjetnicima koji troše svoje vrijeme udovoljavajući svojoj mašti. Drugi ljudi pak smatraju kako je dizajn igara inženjersko zanimanje u kojem se koriste metodologije za određivanje i balansiranje pravila igre. Bilo koji od tih pogleda na dizajn igara je nekompletan te se igre ne mogu gledati samo kroz umjetnost ili kroz inženjerstvo. Igre, pa i one računalne se ne mogu vezati samo za rigorozne standarde ili formalne metode. Cilj igre je da zabavi kroz iskustvo igranja pa dizajn igra stoga zahtjeva od dizajnera da u isto vrijeme bude kreativan i da pažljivo kreira svoj proces dizajniranja igre. [3]

3.2. Uloga dizajnera igara u razvoju računalnih igara

Dizajner računalne igre za vrijeme razvoja ima viziju kako bi igra trebala funkcionirati tijekom igranja. On stvara ciljeve i pravila igre, postupke razvoja igre, razmišlja o priči igre i daje joj život. Dizajner računalne igre je također odgovoran za stvaranje privlačnog igračevog iskustva igranja igre. Odgovoran je za igračevo iskustvo igranja igre na isti način kao što je odgovoran arhitekt pri izradi nacrta zgrade ili scenarist koji piše scenarij za neki film. Dizajner igre planira strukturalne elemente sustava koji kada se aktivira igračevim igranjem igre stvara interaktivno iskustvo. [3]

Uloga dizajnera računalne igre na prvom mjestu biti zastupnik interesa igrača. On mora gledati na svijet računalnih igara kroz igračeve oči. To zvuči jako jednostavno, ali iznenađujuća je učestalost koliko je taj koncept ignoriran kod dizajnera igara. To je tako jer je lako zanijeti se u grafičkom izgledu igre, priči ili novim mogućnostima koje je moguće implementirati u igru i zaboraviti što je to što čini igru jako zanimljivom igračima. Iako se igračima mogu svidjeti specijalni efekti, kvaliteta 3D okruženja ili radnja igre, oni neće igrati igru ako ih iskustvo igre pretjerano ne privlači.[3]

Kao i sva zanimanja, dizajn računalnih igara zahtjeva mnogo talenta i vještine. Talent je urođen, dok se vještina stječe vježbom. Dobri dizajneri računalnih igara trebaju posjedovati širok skup vještina kao što su:

- **Mašta** – vizualna, dramska, konceptualna, lateralno razmišljanje i sposobnost opažanja.
- **Tehnička svjesnost** – dizajner igre mora biti jako dobro upoznat s tehničkim mogućnostima računalne platforme za koju dizajnira igru.
- **Estetske kompetencije** – dizajner igre mora imati opće razumijevanje umjetnosti i stilskog izražavanja.
- **Opće znanje** – neki od najmaštovitijih dizajnera igara su ljudi sa širokim obrazovanjem i koji su zainteresirani za sve što ih okružuje.
- **Spisateljske sposobnosti** – dizajneri igara provode većinu svojeg vremena pišući, tako da bi trebali imati jako dobre spisateljske sposobnosti kod tehničkog, fikcionalnog i pisanja dijaloga.
- **Crtačke sposobnosti** – nisu isključivo potrebne, ali bile bi poželjne kako bi se dizajner igre lakše razumio s konceptualnim umjetnikom. [1]

3.3. Uloga Level dizajnera u razvoju računalnih igara

Drugi naziv za level dizajnera je *graditelj svjetova*. Level dizajner uzima osnovne komponente igre koje dobije od ostalih dizajnera poput dizajnera korisničkog sučelja, dizajnera glavnih funkcionalnosti igre, iskustva igranja, 3D umjetnika i drugih. Komponente koje dobije, level dizajner koristi kako bi dizajnirao individualne razine u igri. Level dizajner se često smatra podređenim dizajneru igre, ali moderni level dizajneri često posjeduju mnogo vrijednih vještina poput 3D modeliranja i skriptnog programiranja, koje ih mogu postaviti iznad dizajnera igara zbog veće predodžbe o tehnologijama potrebnim za ostvarivanje raznih funkcionalnosti igre. Kao rezultat posjedovanja tih vještina, level dizajn je sada specijalizirana vještina, ili skup vještina koja se smatraju jednako važne kao i dizajn igara. Kod većih razvojnih kuća, razvojni timovi sadrže i po nekoliko level dizajnera. Neki od tih dizajnera mogu čak biti neki od vodećih ljudi u razvojnom timu.[3]

Level dizajneri stvaraju sljedeće elemente igračevog iskustva igranja:

- **3D okruženje u kojem se igra odvija.** Level dizajn uključuje stvaranje 2D ili 3D okruženja koristeći se 2D ili 3D programskim paketima specijaliziranim za stvaranje 2D ili 3D sadržaja. Dok dizajner igre definira koje sve elemente treba sadržavati 3D okruženje igre, level dizajner precizno određuje kakav i koji vizualan sadržaj će svaki od razina (eng. levels) sadržavati. Level dizajnerova uloga je da ostvaruje ideje koje je postavio dizajner igre.
- **Inicijalno stanje razine igre** koje uključuje stanja raznih promjenjivih sadržaja. Primjerice da li su na nekoj razini vrata otvorena ili zatvorena prvi puta kada igrač stupi u tu razinu igre. Level dizajner također određuje s koliko umjetnih protivnika će se igrač susretati tijekom igranja određene razine igre, koliko će resursa igrač imati na raspolaganju na početku razine i lokaciju tih resursa koji se mogu pronaći unutar 3D okruženja razine igre.
- **Skup izazova** na koje igrač nailazi tijekom igranja pojedine razine igre. Mnogo igara pružaju igračima izazove u nekom linearnom slijedu pa level dizajneri određuju na koji način će dostaviti taj slijed igračima. Također level dizajner određuje mjesta u 2D ili 3D okruženju na kojima će se pojaviti određeni izazovi.
- **Stanje prekida razine igre.** To stanje se još opisuje kao stanje pobjede ili poraza.
- **Povezanost između priče igre koju igra nosi i iskustva igranja.**

- **Vizualni izgled i ugođaj** razine igre. Dok je uloga dizajnera igre i umjetničkog redatelja da specificiraju sveukupni ton razine igre i 3D umjetnika da stvaraju specifične 3D modele i teksture, level dizajneri uzimaju generalne ideje i odlučuju kako implementirati ih. Ako je ideja primjerice da razina u igri sadrži neku ukletu kuću , level dizajner odlučuje kakva će to biti kuća i što će se koristiti kako bi se ona učinila zastrašujućom i ukletom.[1]

Level dizajneri inače konstruiraju sve dijelove razine koristeći se programskim alatima dizajniranim specifično za tu svrhu. Neke igre primjerice, poput Warcraft III ili Half-Life 2, uključuju te alate tako da igrači sami mogu proširivati igru i modificirati 3D okruženje. Igračima je tako omogućeno vježbanje level dizajna koristeći se alatima koje se koriste prilikom izrade igara koje vole igrati. [3]

Level dizajneri su neko vrijeme pokušavali definirati skupove načela koje bi vodili proces dizajniranja razina igara s ciljem izbjegavanja istih grešaka kod razvoja novih igara. Značajne rasprave se vode u svijetu razvoja računalnih igara oko tog pitanja iz razloga jer se ne slažu svi kako bi bilo koje od tih načela moglo biti univerzalno. Proučavanjem važnijih načela podrazumijeva vrijednu vježbu u svakom slučaju. Neki od tih načela su:

- Rane razine igre bi trebale biti one koje uče igrača igranju igre
- Tijek razine bi trebao biti varijabilan. Što je izuzetno važno za igrače kako ne bi rano izgubili interes za igrom u ranim stadijima igre.
- Kada igrač naiđe na izazov koji troši njegove resurse unutar igre, trebalo bi mu omogućiti da do istih može lako doći.
- Trebalo bi izbjegavati slabu povezanost razina igre.
- Igrača bi se trebalo jasno informirati o njegovim kratkotrajnim ciljevima u igri.
- Igraču bi se jasno trebalo naznačiti rizici, nagrade, odluke i posljedice unutar igre.
- Igrača bi trebalo nagraditi za njegovo vješto igranje igre, za njegovu inteligenciju i moć zapažanja.
- 3D okruženja bi se trebala dizajnirati tako da igrače odmah privuku.[1]

Kada se divimo 3D okruženju neke igre ili uživamo u izazovima koja nam igra postavlja, tada se divimo radu level dizajnera. Level dizajner ne stvara samo 3D okruženje igre u kojem se igra odvija, već pažljivo dizajnira i igračevo iskustvo tijekom igranja igre. Uspješni level dizajneri crpe svoju inspiraciju iz osnovnih načela level dizajna koji se mogu primijeniti na bilo koju vrstu igre, kao primjerice načelo osiguravanja igračeve svjesnosti o kratkotrajnim ciljevima i posljedicama njegovih odluka unutar igre. Level dizajneri blisko surađuju s dizajnerom igre kako bi na najbolji način ostvarili određenu atmosferu koja igra isporučuje igračima. Također, level dizajner se brine za brzinu razvoja priče koja je ključna za igračevo iskustvo, zbog kojeg će biti uključen u svijet igre. Level dizajn nije brz i lak posao, pa je potrebna velika strpljivost i predanost poslu kako bi se ostvarili veliki rezultati. [1]

Umjetnički redatelj i glavni level dizajner odlučuju konačan izgled igre; 3D umjetnici izrađuju 3D modele i teksture; audio inženjeri stvaraju audio efekte, itd. Dok svi ti umjetnici izrađuju sadržaj, na level dizajnerima je odgovornost da sastave sav taj sadržaj u formu koja u konačnici čini jednu razinu igre ili level. 3D okruženja moraju estetski odgovarati atmosferi koju igra isporučuje. Level dizajner u osnovi radi sve što bi primjerice kod izrade filma radilo petorica ljudi; scenski redatelj, redatelj osvjetljenja, redatelj specijalnih vizualnih efekata, redatelj zvuka i čak kinematograf. To je tako zato što level dizajner mora vidjeti svijet igre kroz igračeve oči, kroz leće virtualne kamere.[3]

Kako bi ostvario određenu atmosferu računalne igre level dizajner se služi:

- **Osvjetljenjem.** Smještaj i orijentacija svjetla u 3D okruženju može stvoriti razliku između sunčanog dana, noćne scene obasjane mjesečinom ili mračne ulice. Meko jutarnje svjetlo koje lagano prolazi kroz prozore stvara osjećaj topline i dobroćudnosti, dok čudna svijetla raznih boja u strojarnici stvaraju osjećaj opasnosti. Bilo kojim načinom osvjetljenja 3D okruženja se služio level dizajner, osvjetljenje mora funkcionirati s ostalim estetskim elementima koji tvore atmosferu 3D okruženja. Također je važno koje elemente 3D okruženja level dizajner odabire osvijetliti, a koje ne. Osvjetljenje tako igra jednu od najvažnijih elemenata kod postavljanja atmosfere 3D okruženja računalne igre.
- **Paletom boja.** Baš kao što paleta boja može dati osobnost likovima unutar igre, isto tako može reflektirati atmosferu 3D okruženja. Paletu boja 3D okruženja level dizajner odabire na osnovu kombinacije boja koje su korištene na 3D modelima koje stavlja u 3D okruženje i tipa osvjetljenja koje se koristi.

- **Atmosferskim efektima.** Magla, kiša, snijeg i vjetar mogu stvoriti različite impresije. Puno računalnih igara uključuje 3D okruženja koja su interijeri pa se zbog toga često lako može zaboraviti važnost atmosferskih efekata pri stvaranju atmosfere 3D okruženja računalne igre. Mračni, teški oblaci s grmljavinom instinktivno kod igrača izazivaju osjećaj nezaštićenosti i tjera ih da odmah potraže sklonište od kiše i munja. Magla može stvoriti misteriju, dok jaki vjetrovi nagovještaju atmosferske nestabilnosti. Korištenjem atmosferskih efekata u 3D okruženju, level dizajner može stvoriti jako uključujuću atmosferu za igrače koja ih privlači da igru igraju dulje.
- **Specijalnim efektima.** Level dizajner se koristi raznim specijalnim efektima kako bi razvio atmosferu stvarnog 3D okruženja za igrače. Primjerice kada u igri vidimo da lik puni oružje ili kada čujemo škripanje automobilskih guma na cesti, ili dok neka specijalna čarolija proizvede iskre u boji, ili kada vidimo zidove pune mrlja od krvi, tada vidimo specijalne efekte kojima se level dizajner koristio kako bi stvorio atmosferu 3D okruženja.
- **Glazbom.** Level dizajner se koristi glazbom iako on sam ne mora biti glazbenik, ali on je odabire za svoje 3D okruženje surađujući s redateljem zvuka. Ritam glazbe pomaže uspostaviti korak kojim igra teče, dok boja glazbe pomaže uspostaviti ukupnu atmosferu 3D okruženja. Uglavnom, ali ne uvijek, glazba ostaje konzistentna tijekom cijele jedne razine igre.
- **Ambijentalnim zvukovima.** Kao i glazbom, level dizajner se koristi ambijentalnim zvukovima kako bi uspostavio atmosferu 3D okruženja igre. Tako primjerice računalna igra koja simulira sportsku igru golf, koristi zvukove pjeva ptica ili zvukove zrikavaca kako bi uspostavila mirna vanjska atmosfera kakva vlada tijekom igranja golfa u stvarnosti. Ambijentalni zvukovi mogu varirati ovisno o mjestu i vremenu kako bi igraču pomogli kod snalaženja u 3D okruženju. Primjerice, zvuk rada velikih parnih motora stvara osjećaj moći i opasnosti, dok huk sova nagovještava da je noć. Isto tako zvuk velike grupe ljudi nas smješta na gradski trg. Zvuk je jedan od najmoćnijih alata kojima bi se level dizajneri koriste kako bi ostvarili atmosferu 3D okruženja kojeg stvaraju. [4]

Level dizajner se koristi jednim od najvažnijih koncepata u procesu dizajna 3D okruženja računalnih igara – izradom prototipa. Veći dio procesa izrade prototipa 3D okruženja sastoji se od korištenja programskih alata za 3D modeliranje kako bi se konstruirali osnovni modeli koji će tvoriti 3D okruženje i objekte koji se pojavljuju unutar njega. Korištenje zasebnog programskog paketa za izradu privremenih 3D modela nije uvijek potrebno jer neki od pokretača igre poput Unreal Engine 4 omogućuju tu funkcionalnost. Privremeni 3D modeli služe samo kako bi se napravio nacrt po kojem će onda umjetnički tim raditi svoje konačne modele. [4]

Izrada prototipa 3D okruženja uključuje:

- Izradu osnovnih 3D modela 3D okruženja koja može ali i ne mora biti stvorena korištenjem zasebnih programskih paketa za izradu 3D modela.
- Izradu privremenih tekstura koje će biti korištene s privremenim 3D modelima koje će kasnije biti zamijenjene s konačnim teksturama.
- Izradu privremenih 3D modela rekvizita (eng. props) poput drveća, namještaja, zgrada, itd.
- Planiranje kretanja likova s umjetnom inteligencijom
- Planiranje osvjetljenja za 3D okruženje.
- Postavljanje lokacija u 3D okruženju unutar kojih se događa interakcija s igračem[4]

Prilikom izrade prototipa moguće je koristiti i finalne audio efekte, tj. audio efekte koji će završiti u konačnoj verziji računalne igre. Ako finalni audio efekti nisu dostupni, tj. audio razvojni tim nije još stigao napraviti ih, tada level dizajner koristi privremene uz zabilješku da ih treba zamijeniti s konačnima. Nakon izrade konačnog prototipa 3D okruženja, ako su ga programeri osposobili unutar pokretača igre, tada je moguće testirati to okruženje na najosnovnijoj razini. Testiranjem prototipa dobivaju se vrijedne povratne informacije od svih zaposlenih uključenih u razvoj igre. [3]

4. Tijek izrade 3D okruženja u Unreal Engine 4

Tijekovi rada (eng. workflow) su organiziran način proizvodnje nekog sadržaja. Sve velike razvojne kuće koriste neki svoj, već ustaljeni tijek rada koji im omogućuje maksimalno iskorištavanje vremena uz veliku učinkovitost. Prilikom izrade digitalnog sadržaja za igre razvojni timovi često koriste različite programske pakete, pa je ključna dobra suradnja između timova koji proizvode različit digitalan sadržaj, jer se tako štedi vrijeme i novac. Svi zaposlenici unutar razvojnog tima zato moraju jako dobro poznavati tijek rada koji se ustalio u njihovoj razvojnoj kući. Tijekovi rada postoje i na osobnoj razini. Primjerice svaki 3D umjetnik koristi po nekoliko programskih paketa za 3D modeliranje, teksturiranje, animaciju, pa s vremenom razvije svoj tijek rada u tim programskim paketima. Dugogodišnjim iskustvom 3D umjetnici usklade svoj tijek rada s tijekom rada razvojne kuće te tako maksimalno iskorištavaju vrijeme uz veliku učinkovitost.

Za vrijeme razvoja 3D okruženja za projekt imena *S.S. Infinity* na kojem sam radio za potrebe kolegija video animacija razvio sam svoj tijek rada. Tijek rada uključivao je korištenje programskih paketa *Autodesk Maya 2016*, *Allegorithmic Substance Painter* 2, *Substance Designer 5* i *Unreal Engine 4* razvojne kuće Epic Games, te *Adobe Premiere CC*, i Photoshop CC koji neće biti široko objašnjeni, jer čine jako mali dio sveukupnog tijeka rada. Za sve te programske pakete dobio sam besplatne studentske licence u trajanju od jedne godine te se još jednom zahvaljujem tvrtkama koje su mi to omogućile. Rezultat projekta S.S. je video animacija u trajanju od dvije minute i četrdeset i sedam sekundi. Iako sam 3D okruženje izrađivao s planom da ono bude korišteno za jednostavnu 3D računalnu igru. Zbog nedostatka vremena i ostalih obaveza na studiju bio sam vremenski ograničen raditi na tom projektu maksimalno dva do tri sata dnevno, tako da mi je tijek rada kojeg sam osmislio uvelike pomogao na vrijeme uspješno dovršiti taj projekt. U poglavljima koji slijede u osnovama ću pojasniti svaki od programskih paketa koje sam koristio, i koncepte razvoja 3D okruženja u Unreal Engine 4 pri korištenju tijeka razvoja kojeg sam osmislio za potrebe projekta.

4.1. Projekt S.S Infinity

Projekt *S.S Infinity* je zamišljen kao 3D računalna avantura iz prvog lica, u kojoj igrač otkriva elemente transportnog svemirskog broda pod imenom *Infinity*. *S.S. Infinity* je samoodrživi svemirski brod kojim ne upravljaju ljudi, a služi za transport raznog opasnog tereta. Glavni cilj igre je pronaći način kako doći do glavnih kontrola svemirskog broda i ispraviti mu putanju koja ga vodi u središte zvijezde. Igra bi se sastojala od 4 prostorije koje zajednički tvore isto 3D okruženje, u kojima lebdeći robotski nosači prenose razne spremnike iz jedne u drugu prostoriju. Ovladavanjem mehanike robotskih nosača, igrač bi trebao koristiti logičke zaključke kako bi shvatio na koji način svemirski brod funkcionira kako bi došao do kontrolne prostorije i ispravio putanju svemirskog broda.

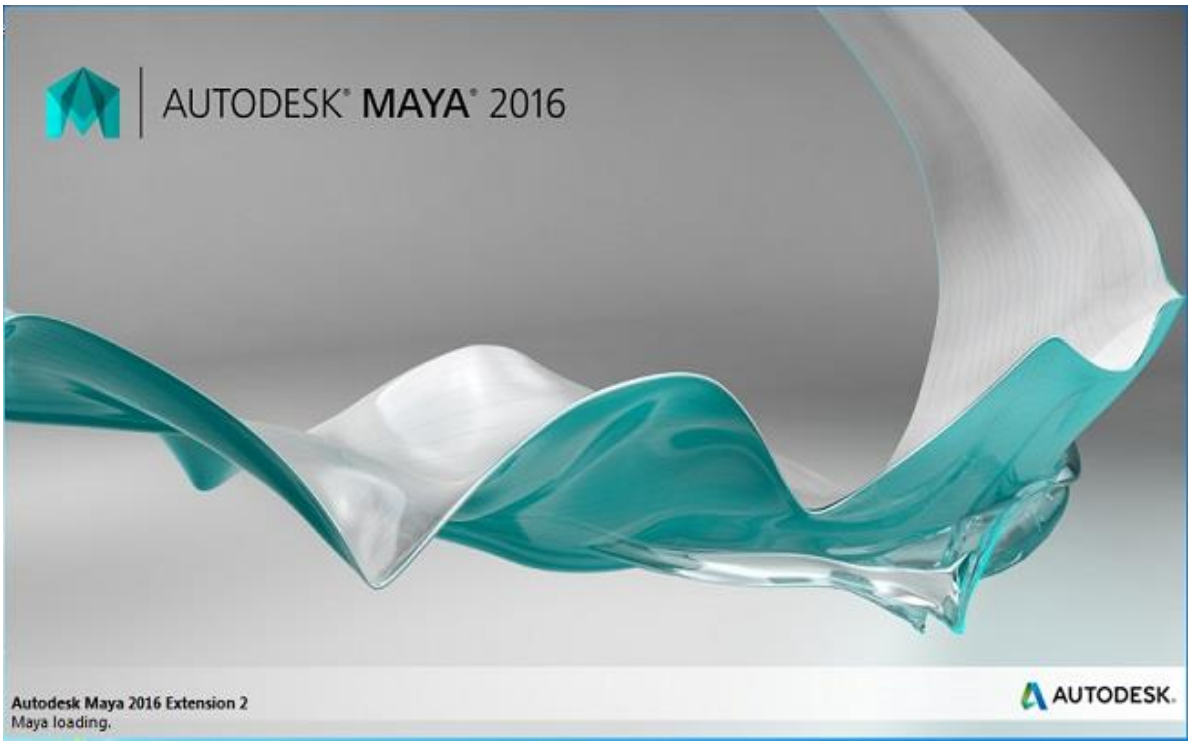
3D okruženje pokreće pokretač igre Unreal Engine 4. Zamišljeno je za potrebe računalne igra, ali je zbog nedostatka vremena za daljnji razvoj korišteno za potrebe kolegija Video animacija. Razvoj projekta zahtijevao je znanje 3D modeliranja, teksturiranja, video animacije, level dizajna, osvjetljenja, renderiranja, post procesiranja, kadriranja i video editiranja. Tijek rada koji je korišten prilikom izrade 3D okruženja uvelike je ubrzao razvoj usmjeren na video animaciju i kretnje kamere. Tijek rada izrade uključivao je pomno planiranje izrade svih elemenata koji će biti korišteni unutar 3D okruženja. Animacijom su u konačnici prikazane dvije prostorije, koridor svemirskog broda i skladište. Iako, su u generalnoj zamisli trebale postojati još prostorija strojarnice i prostorija spremišta automatiziranih lebdećih robota, te tajna kontrolna prostorija. U daljnjim poglavljima bit će prikazan tijek razvoja 3D okruženja koje uključuje koridor i skladište svemirskog broda S.S Infinity.

Slika 4.1. S.S. Infinity Logo.

4.2. Programski paketi

Kako bi se izradilo 3D okruženje za projekt S.S. Infinity, korišteno je nekoliko programskih paketa. Izrada 3D objekata zahtijevala je znanje korištenja Autodesk Maya 2016 paketa za 3D modeliranje, Allegorithmic Substance Painter 2 za izradu tekstura, te Substance Designer 5 za izradu dodatnih tekstura potrebnih za ispravan prikaz u Unreal Engine 4 i pakiranje svih tekstura u jednu datoteku razumljivu Unreal Engine 4. Epic Games Unreal Engine 4 je korišten za konačnu izradu 3D okruženja i animacije. U daljnjem tekstu slijedi u osnovama pojašnjenje korištenih programskih paketa.

4.2.1. Autodesk Maya 2016



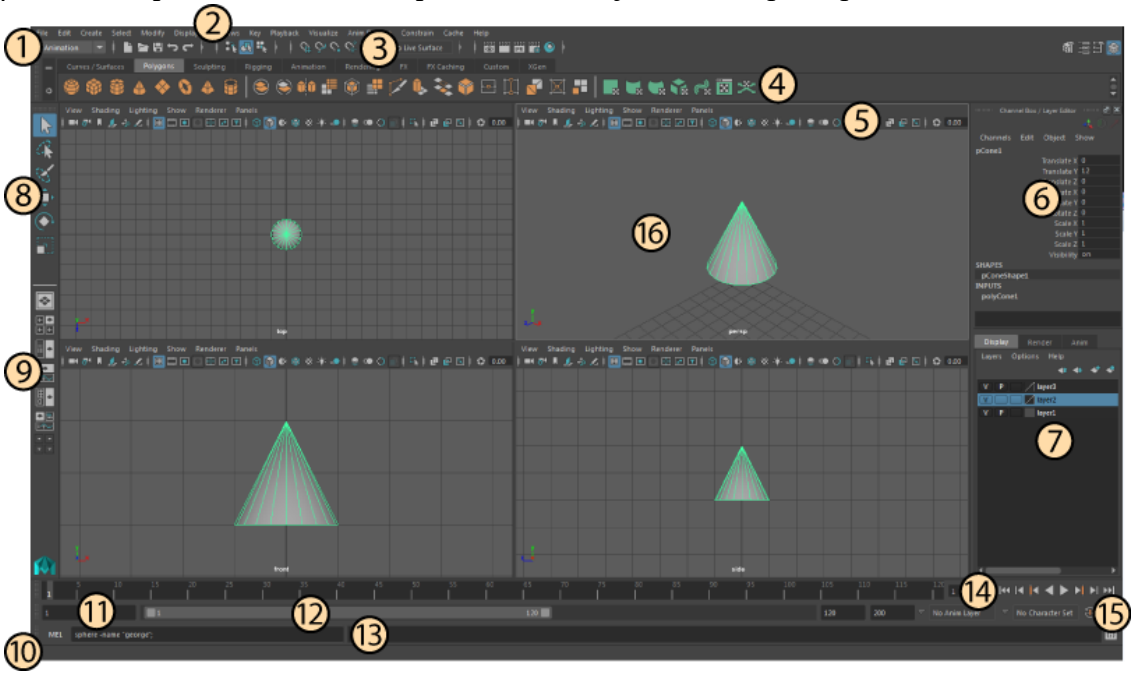
Slika 4.2. Početni zaslon Maya 2016. [5]

Autodesk je tvrtka koja se bavi razvojem raznih programskih alata za korištenje u proizvodnji, arhitekturi, gradnji, i zabavnoj industriji. U zabavnoj, i industriji igara poznatiji su po programskim paketima *Maya, 3DS studio max, Motion Builder, Mudbox* i mnogim drugim alatima. [5]

Za potrebe izrade 3D modela za 3D okruženje projekta *S.S Infinity*, korišten je Autodeskov programski paket *Maya 2016*. *Maya 2016* je opširan programski paket kojim se koriste mnogi razvojni studiji u industriji računalnih igara za izradu 3D modela, animacija likova, specijalnih efekata, simulacije fizike, i mnogih drugih potreba.

Sučelje Autodesk Maya 2016

Kada novi korisnik po prvi puta otvori *Maya 2016*, zasigurno će biti preplašen s opsežnosti ikona i izbornika grafičkog sučelja, ali ubrzo će shvatiti kako je zapravo izuzetno logički osmišljeno kako bi mu se olakšao rad. *Maya 2016* također ima i radijalni izbornik s najčešće korištenim mogućnostima, kojem se pristupa dugim držanjem tipke desne tipke miša, *space*, *shift* i *ctrl* na tipkovnici, ovisno o tipu izbornika kojem želimo pristupiti.



Slika 4.3. Pregled grafičkog sučelja Maya 2016. [5]

Pregled grafičkog sučelja Autodesk Maya 2016:

1 - Padajući izbornici. Sedam padajućih izbornika je uvijek dostupno neovisno o izborniku skupova alata, a to su :

- **File**
- **Edit**
- **Select**
- **Modify**
- **Display**
- **Windows**

1.1 - Izbornik skupova alata koji određuje koji će se padajući izbornici pojaviti s odabirom pojedinog skupa izbornika. Skupovi izbornika koji se nalaze u ovom padajućem izborniku su:

- **Modeling**
- **Rigging**
- **Animation**
- **FX**
- **Rendering**

2 - Padajući izbornici koji sadrže alate i akcije za manipuliranje objektima, postavljanje scene i ostalih mogućnosti. Ovaj predio sučelja sadrži prethodno nabrojanih sedam standardnih padajućih izbornika i dodatne koji se pojavljuju nakon što je odabran određen skup izbornika. Ovakav način organizacije sličan je kao i u Adobeovim programskim paketima koji omogućuje odličnu organizaciju padajućih izbornika koje korisnik bira na osnovu željenih akcija koje želi ostvariti. Tako je vješto izbjegnuta prenatrpanost sučelja i optimalan razmještaj padajućih izbornika kako se ne bi previše uzelo od prostora radnog prostora.

3 - Statusna linija s ikonama za brzi pristup alatima i za selekciju i poravnanje (eng. snapping) objekta te brzi pristup prozoru za renderiranje i spremanje/otvaranje projekta.

4 - Polica (eng. shelf) koja može sadržavati predefinirane poligonalne ili NURBS objekte te alate ovisno koji skup alata je odabran. Ako npr. u izborniku skupa alata korisnik odabere *Polygons*, tada će mu se prikazivati osam predefiniranih poligonalnih objekata i svi alati potrebni za manipuliranje poligonima. Na policu je moguće dodavati alate koji će se često koristiti, ali isto tako moguće je stvoriti i vlastitu policu s alatima i akcijama koje se najčešće koriste.

5. - Traka s alatima. Ovaj predio sučelja sadrži najčešće korištene alate i stavke padajućih izbornika koje se koriste kod modeliranja ili postavljanja scene. U statusnoj liniji moguće je pristupiti npr. postavkama kamere, uključiti ili isključiti mrežu u radnom prostoru, uključiti žičani prikaz (eng. wireframe) način prikaza objekta, uključiti prikaz objekta s teksturom i puno drugih mogućnosti.

5.1 - Iznad trake s alatima nalaze se padajući izbornici koji se odnose na radni prostor. Tako postoje *view*, *shading*, *lightining*, *show*, *render*, *panels* izbornici u kojima korisnik može odabrati koju vrstu objekta želi prikazati u radnom prostoru, odabrati vrstu potprograma za prikaz objekata u realnom vremenu i dr.

6 - Channel box. Ovaj predio sučelja omogućuje korisniku manipuliranje objektom i njegovim vrijednostima kao što su npr. njegove dimenzije, pozicija u prostoru, rotacija i vidljivost. U Channel box-u moguće je također pristupati povijesti, izvedenim akcijama nad objektom i mijenjati ih.

7 - Layer editor. Maya 2016 ima 3 vrste slojeva (eng. layer), *display*, *render* i *animation*. Display slojevi se koriste kod manipuliranje scenom, render slojevi se koriste za postavljanje render prolaza (eng. render pass) kod post produkcije (eng. compositioning).

8 - Traka s alatima za manipuliranje objektima, zove se još i QWERTY traka s alatima. Ona sadrži alate za selekciju, pomicanje, skaliranje i rotaciju objekta kojima se isto tako može pristupati tipkama Q,W,E,R,T,Y na tipkovnici.

9 - Alati za brzo pristupanje panelima. U ovom predjelu sučelja organizirane su ikone za brzo pristupanje npr. *outliner* prozoru za pristupanje organizaciji objekata u sceni, i drugih panela kao što su odabir tlocrtnog, bokocrtog, nacrtnog i 3D radnog prostora.

10 - Linija u kojoj se ispisuju pomoćna objašnjenja alata. Na ovoj liniji također mogu biti ispisana uputstva kako iskoristiti neki od alata.

11 - Vremenski klizač (eng. time slider). Prikazuje raspon vremena animacije definiranog u klizaču raspona (eng. range slider). Vremenski klizač također pokazuje *keyframe* svakog objekta u sceni.

12 - Klizač raspona animacije (eng. range slider) koji služi za definiranje početka i kraja animacije ali i raspon duljine reprodukcije animacije ako se npr. korisnik želi fokusirati na mali dio animacije.

13 - Komandna linija u koju se mogu unositi MEL naredbe za modifikaciju sučelja i mogućnosti samog programa.

14 - Kontrole za reprodukciju animacije koje omogućuju pokretanje animacije u duljini zadanoj s vremenskim klizačem. S ovim skupom alata moguće je reproducirati animaciju od početka do kraja kako je zadano klizačem raspona. Moguće je pomicati kursor kroz animaciju u stupnju po jedan *keyframe*, i također vratiti se na početak i na kraj animacije.

15 - Alati za kontrolu animiranih objekata u sceni pomoću kojeg je moguće stvoriti skupove objekata koji će kasnije biti korišteni u animaciji. Isto tako moguće je prebacivati način rada iz animacijskog sloja u *character set* s ovim izbornicima.

16 - Radni prostor koji se sastoji od mnogo panela za upravljanjem scenom i koji velikim dijelom služi za prikaz objekta kojeg modeliramo. Uvijek je poželjno da radni prostor bude što veći za ugodniji rad na sceni pa se zbog toga isključuju poneki elementi sučelja ako nisu potrebni. Primjerice ako korisnik radi na modeliranju objekta, nije mu potreban dio sučelja koji služi za animaciju pa ga može isključiti u standardnom padajućem izborniku *Display* ako izabere pod izbornik *UI elements*.

Sučelje Maya 2016 na prvi pogled je zastrašujuće, ali nakon što se prouče upute s Autodeskove stranice i kako korisnik sve više vremena provodi koristeći program i prilagođuje ga svojim potrebama, tako sučelje postaje ugodnije za korištenje i počinje imati sve više smisla. Sučelje Maya 2016 je dizajnirano kako bi se maksimalno iskoristio prostor na ekranu računala, te odlično balansira između pravilnog vizualnog rasporeda elemenata i funkcionalnosti. Velika prednost Maya 2016 je što koristi MEL skriptni jezik koji omogućuje naprednim korisnicima modificiranje raznih elemenata sučelja, kao i same funkcionalnosti programa.

4.2.2. Allegorithmic Substance Painter2

Allegorithmic je razvojna kuća koja razvija programske pakete Substance Designer i Substance Painter koji služe za izradu proceduralnih tekstura za 3D modele. Razvojna kuća postoji od 2002. koja je do danas postavila svojevrstan standard izrade proceduralnih tekstura za 3D računalne modele. Proceduralne teksture su u potpunosti računalno generirane bit mape pomoću kompleksnih grafičkih algoritama i kao takve pogoduju raznim pokretačima igara. Substance Painter 2 je programski paket koji omoguće izradu fizički točnih tekstura (eng. PBR). Program je logički osmišljen slično Adobe Photoshopu u smislu rada sa slojevima i maskama. [6]

Sučelje Substance Painter2

Slika 4.4. Sučelje Substance Painter 2.

Pregled grafičkog sučelja Substance Painter 2:

1. - Padajući izbornici *File, Edit, Mode, Plugins, Help* kojima se pristupa standardnim mogućnostima programa.

2. - Alatna traka koja sadrži alate za izradu maski, načina bojanja tekstura, omogućavanje efekta zrcaljenja i dr.

3. - Radni prostor u kojem se prikazuje rezultat izrađenih tekstura u stvarnom vremenu.

4. - Slojevi (eng. Layers). Prostor u kojem se prikazuju slojevi koji sadrže osnovne materijale. Postoje dvije vrste slojeva, *Fill* i *standardni* sloj koji sadrže sve osnovne kanale potrebne za rad, ovisno o načinu rada koji korisnik odabere kod stvaranja novog projekta. Kanali zajedno tvore fizički točne teksture.

5. - Prostor za prikaz skupova tekstura (eng. texture set). Skupovi tekstura se koriste kod većih modela koje je potrebno podijeliti na nekoliko skupova tekstura kako bi se za njih lakše izradile teksture.

6. - Svojstva. Prostor u kojem se prikazuju svojstva sloja, kista i kanala na kojima korisnik želi raditi.

7. - Postavke skupa tekstura. U ovom prostoru se prikazuju postavke PBR kanala, izrade dodatnih tekstura potrebnih za rad na izradi glavnih tekstura. Također se u ovom prostoru prikazuju postavke prikaza pozadine, osvjetljenja, specijalnih efekata i dr.

8. - Polica (eng. Shelf). Polica je dio Substance Painter 2 koja sadrži sve potrebne resurse potrebne za izradu tekstura kao što su mapa s vanjskim teksturama, alfa kanal teksture, razni alati, filteri, i proceduralne teksture, osnovni materijali, napredni materijali i mnogo drugih.

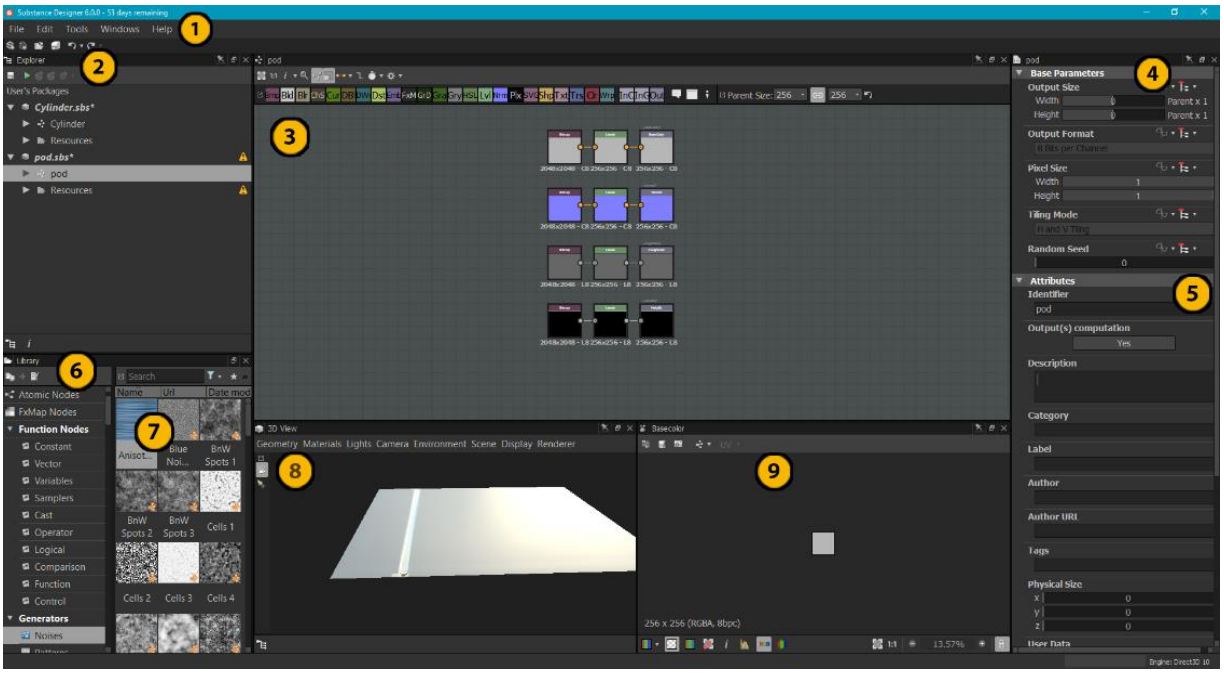
9. - Prostor za prikaz odabrane mape resursa. Na slici 4.4. su trenutno prikazane alfa kanal teksture koje služe za izradu maski.

Substance Painter 2 je izrazito moćan programski paket za izradu proceduralnih, fizički točnih tekstura. Logika rada jako je slična Adobe Photoshopu u smislu rada sa slojevima i maskama. Izrada tekstura u Substance Painter 2 odvija se tako da prilikom stvaranja novog projekta korisnik mora odabrati način izrade PBR tekstura. Između ostalih postoje dva glavna načina, *Metal/Glossiness* i *Metal/Roughness.* Oba su u osnovi jako slična, razlika je većinom u načinu interpretacije tih tekstura unutar grafičkog podsustava pokretača igre. Nakon što korisnik odabere način izrade tekstura, prvi korak mu je izrada dodatnih tekstura nakon čega slijedi rad na kanalima. Ovisno o odabranom načinu izrade tekstura, korisnik dalje radi na osnovnim kanalima. Nakon završetka, potrebno je napraviti izvoz tekstura za što Substance Painter 2 ima predefinirane postavke. Cijeli proces izrade tekstura bit će prikazan u daljnjim poglavljima.

4.2.3. Allegorithmic Substance Designer 5

Allegorithmicov Substance Designer 5 je također izrazito moćan alat koji služi za izradu osnovnih materijala, koji se kasnije koriste u kombinaciji sa Substance Painter 2 kako bi se izradile konačne, fizički točne teksture. Substance Designer 5 je programski paket koji omogućuje izradu grafičkih mreža, tj. omogućuje vizualno skriptno programiranje. Korisnik se služi predefiniranim, osnovnim čvorištima (eng. node) i u kombinaciji s proceduralnim generatorima izrađuje osnovni materijal, ili Substance koji će kasnije biti korišten u Substance Painter 2 za izradu konačnih tekstura. [7]

Sučelje Substance Designer 5



Slika 4.5. Sučelje Substance Designer 5.

Pregled sučelja Substance Designer 5:

1. - Padajući izbornici *File*, *Edit*, *Tools*, *Windows*, *Help* kojima se pristupa standardnim mogućnostima programa.

2. - *Explorer*. Prostor u programu koji služi za upravljanje datotečnom strukturom projekta na disku.

3. - Radni prostor za rad s čvorištima i izradu vizualnog skriptnog koda koji tvori osnovni materijal ili Substance.

4. - Osnovni parametri za izradu tekstura. U ovom dijelu sučelja korisniku su omogućene kontrole nad izlaznim postavkama tekstura kao što su izlazna rezolucija, način nadovezivanja, i naslijeđena rezolucija tekstura.

5. - Atributi projekta. U ovom dijelu sučelja moguće je unijeti osnovne informacije o projektu kako bi se olakšala organizacija.

6. - Knjižnica (eng. library). Kao što polica u Substance Painter 2 sadrži sve dostupne resurse za izradu tekstura, tako knjižnica sadrži sve dostupne predefinirane resurse za izradu osnovnih materijala ili Substancea.

7. - Prikaz sadržaja odabrane mape u knjižnici.

8. - 3D prikaz osnovnog materijala. Ovaj prozor sadrži dodatne padajuće izbornike *Geometry*, *Materials*, *Lights*, *Camera*, *Environment*, *Scene*, *Display*, *Renderer*. Dodatni izbornici omogućuju korisniku prilagodbu prikaza osnovnog materijala kako bi bio u skladu s određenim 3D okruženjem u kojem će biti korišten.

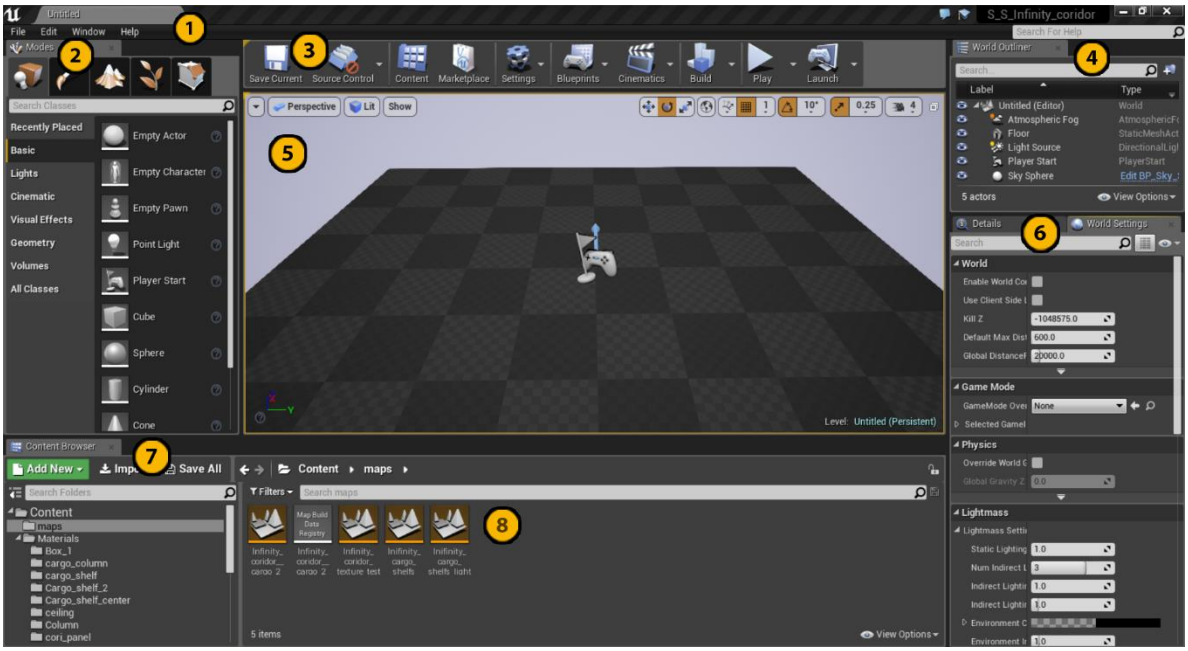
9. - 2D prikaz određenog kanala koji se prikazuje kada korisnik klikne desnim klikom miša na jedan od izlaznih čvorišta koji predstavljaju određene kanale. Na slici 4.5. prikazuje se osnovna boja teksture rezolucije 512x512px.

Substance Designer 5 omogućuje izradu kompliciranih *shader mreža* koje su u osnovi vizualni skriptni računalni kod. Takav način rada, iako zahtjeva jako puno strpljenja pri učenju načina na koji program funkcionira, omogućuje izradu jako personaliziranih osnovnih materijala. Substance Designer 5 je također potreban za pakiranje sveukupnih tekstura za korištenje s većim i poznatijim pokretačima igre poput Unreal Engine 4. Substance Designer 5 korišten je za potrebu projekta S.S. Infinity kod izrade dodatnih tekstura i pakiranje svih tekstura za Unreal Engine 4. Cijeli postupak rada u Substance Designer 5 bit će prikazan u daljnjim poglavljima.

4.2.4. Epic Games Unreal Engine 4

Epic Games je razvojna kuća osnovana 1991. godine s trenutnim sjedištem u Californiji, S.A.D. Epic Games su poznati po *Unreal tournament*, *Gears of war* i *Infinity Blade* serijalom računalnih igara. Također tvrtka je poznata po razvoju svojeg pokretača igara Unreal Engine koji je u trenutku pisanja ovog rada dosegao inačicu 4.16. Unreal Engine je slobodno dostupan i besplatan pokretač igre koji uključuje i besplatan editor. Epic Games konstantno unosi nove mogućnosti u Unreal Engine 4, stoga on predstavlja vodeći alat za izradu različitih računalnih igara koji podržava velik broj računalnih platformi. Igre koje se pomoću Unreal Engine 4 razviju za računalno, s lakoćom se mogu prilagoditi za primjerice Android pametne telefone ili igrače konzole. Upravo su grafičke mogućnosti Unreal Engine 4 bile presudan faktor za odabir pokretača za 3D okruženje S.S. Infinity projekta. [8]

Sučelje Unreal Engine 4



Slika 4.6. Sučelje Unreal Engine 4.

1. - Padajući izbornici *File*, *Edit*, *Window*, *Help* kojima se pristupa standardnim mogućnostima programa.

2. - Izbornik Načina rada. Postoje pet osnovnih načina rada *Place*, *Paint*, *Landscape*, *Foliage* i *Geometry* . U *Place* načinu rada stoje pod izbornici *Basic*, *Lights*, *Cinematic*, *Visual*

Effects, *Geometry*, *Volumes*, *Classes* koji omogućuju smještanje osnovnih elemenata, osvjetljenja, vizualnih efekata i mnogih drugih mogućnosti.

3. - Alatna traka na kojoj su smješteni alati u obliku ikona koji služe za mogućnosti poput spremanje projekta, pregleda dostupnog sadržaja, online trgovine, generalnih postavki editora, pristup *blueprint* editoru, izrada animacija, simulacija igre, izrada osvjetljenja, igranje igre.

4. - *World outliner*. Dio sučelja koje sadrži listu svih elemenata koji tvore 3D okruženje. Ovaj dio sučelja omogućuje lako lociranje pojedinog elementa u sceni i manipuliranje njime.

5. - 3D radni prostor unutar kojeg korisnik slaže svoju 3D scenu u potpunosti. U gornjem lijevom i desnom kutu 3D radnog prostora se nalaze ikone za prikaz scene, kao i ikone za smještaj sadržaja i kretanje kroz scenu.

6. - *World Settings*. U ovom dijelu sučelja dostupne su napredne postavke pokretača igre, poput *Lightmass* sustava za simulaciju indirektnog osvjetljenja i postavke simulacije fizike, i dr.

6.1. - *Details*. Ovaj dio sučelja prikazuje informacije o selektiranom 3D objektu. Tu se prikazuju informacije poput lokacije objekta u 3D sceni, materijali, postavke osvjetljenja, fizike objekta, i dr.

7. - *Content Browser*. Dio sučelja koji omogućuje prikaz strukture projekta na hard disku računala na kojem se nalazi. *Content browser* omogućuje uvoz objekata i tekstura, te prikaz istih.

8. - Prikaz sadržaja u odabranoj mapi.

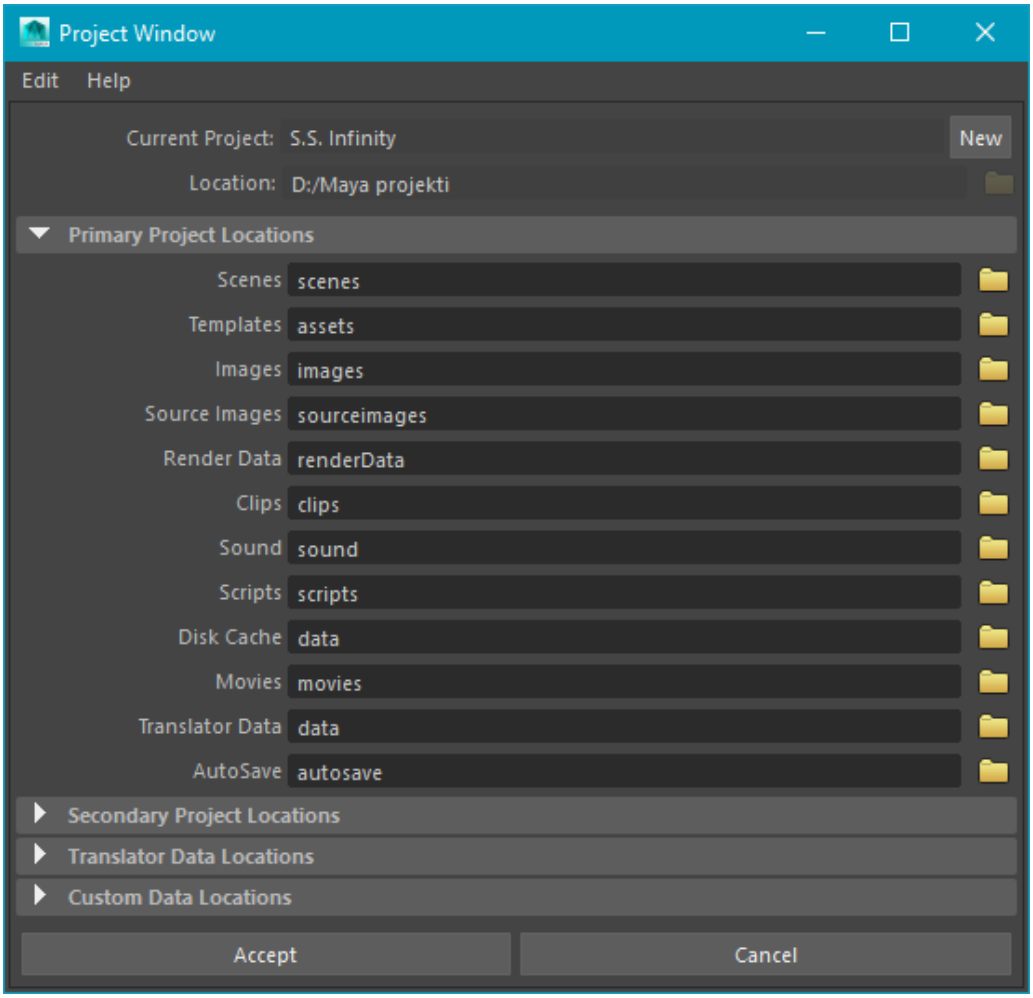
Unreal Engine 4 je izrazito moćan pokretač igre koji omogućuje izradu vizualno privlačnih 3D okruženja. Unreal Engine 4 to postiže implementacijom PBR načina prikazivanja tekstura i osvjetljenja. Kao i kod svakog programskog paketa, važno je znati gdje se određene mogućnosti u sučelju nalaze i zašto se tamo nalaze. Kada je korisnik upoznat s načinom na koji sučelje Unreal Engine 4 funkcionira, s lakoćom može shvatiti cijelu mehaniku rada.

4.3. Organizacija projekta na hard disku računala

Razvoj 3D okruženja računalnih igara velik je pothvat pa je organizacija projekta na hard disku računala izrazito važna. Opseg projekta vrlo brzo može narasti pa je organizacija mapa i datoteka kao i konvencija njihovih naziva izrazito važna, kako zbog snalaženja tako i zbog uštede vremena. Svoj tijek rada započeo sam koristeći se organizacijskim mogućnostima unutar programskih paketa koje sam koristio.

4.3.1. Organizacija mapa

Maya 2016 ima odličan podsustav za organizaciju datotečne strukture projekta kojeg je moguće modificirati potrebama projekta. Koristio sam predefiniranu strukturu mapa, dok sam konvenciju naziva datoteka osmislio sam.



Slika 4.7. Project window u Maya 2016.

Struktura mapa prikazana na slici 4.7. je poslužila kao odličan početak u organiziranju mapa projekta S.S. Infinity. *Scenes* mapa je korištena za spremanje izrađenih 3D modela, kao i kompletnih dijelova 3D okruženja koji su kasnije bili uvezeni u Unreal Engine 4. Mapa *Data* je korištena za spremanje 3D modela u FBX datoteke za potrebe teksturiranja i uvoz u Unreal Engine 4. *Data* mapa sadrži podmape *Meshes* koja sadrži podmape nazvane *Corridor* i *Cargo* u koje će biti spremane fbx datoteke male i velike poligonalne rezolucije. Mapa *Images* bit će korištena za spremanje referentnih slika koje služe kao inspiracija za izradu i teksturiranje 3D modela. Nakon prihvaćanja predefiniranih mapa u *Project window* u Maya 2016, na lokaciji projekta je dodana dodatna mapa naziva *Substance* koja će služiti za spremanje izrađenih tekstura izrađenim programskim paketima Substance Painter 2 i Substance Designer 5. U mapi *Substance* dodane su još podmape *Designer_painter_files* za pospremanje izvornih datoteka programskih paketa Substance Painter 2 i Substance Designer 5. Dodane su još mape *tex* za pospremanje izrađenih tekstura, *UE4_materials* za pospremanje pakiranih tekstura, *alphas* za pospremanje alfa tekstura izrađenih u Adobe Photoshopu, i *bakes* za pospremanje dodatnih tekstura izrađenih u Substance Designer 5.

Unutar Unreal Engine 4 editora, u *content browseru* stvorene su mape *maps*, *meshes*, i *materials*. Mapa *maps* služit će za spremanje izrađenih scena i sekvenci kamere, mapa *meshes* služit će za spremanje 3D modela, dok će mapa *materials* sadržavati sve teksture i materijale 3D modela. Kasnije u razvoju projekta dodana je mapa PP_LUT koja sadrži post proces efekte korištene kod postavljanja osvjetljenja.

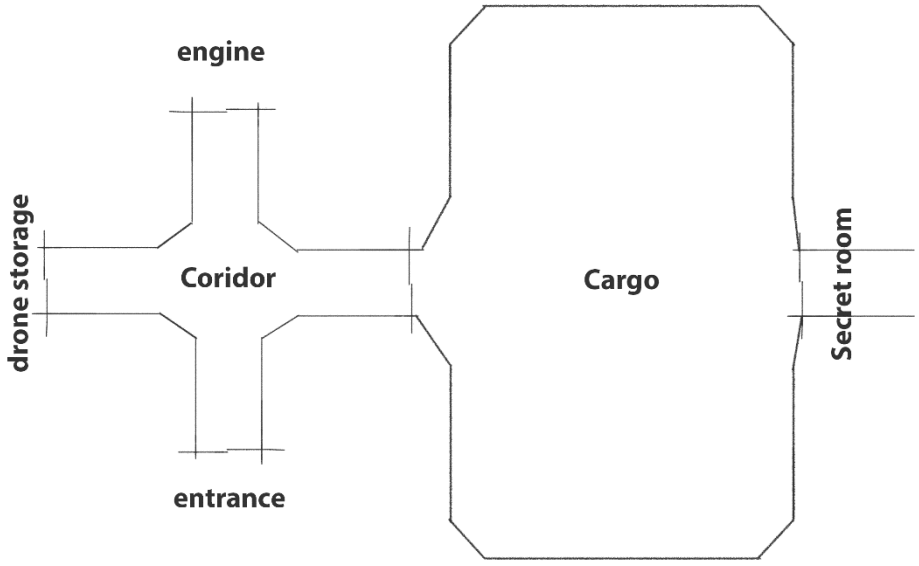
Nakon što su sve mape bile definirane, bilo je potrebno definirati konvenciju imenovanja 3D modela i njihovih materijala kako bi se lakše snalazio u Unreal Engine 4 i Substance programskim paketima. 3D modeli niske poligonalne rezolucije tako imaju konvenciju imenovanja *naziv_low*, dok 3D modeli visoke poligonalne rezolucije imaju naziv *naziv_high*. Kod istih naziva datoteka, označavao sam datoteke tako da su u svom nazivu imale naziv mape u kojem se nalaze. Primjerice 3D model podnice za 3D okruženje koridora i skladišta su nosile naziv *cargo_floor_low* i *cori_floor_low*. Tako se unutar Unreal Engine 4 editora izbjegla redundantnost naziva. Za imenovanje mapa i ostalih datoteka često sam koristio engleski jezik zbog kraćih naziva i bolje preglednosti. Definiranje mapa odmah na početku izrade projekta se pokazala kao dobra odluka jer je donijela velike dobitke u organizaciji vremena. Kasnije u procesu izrade projekta, dodao sam još prečace u kontekstne izbornike Windows 10 operacijskog sustava.

4.4. Izrada 3D okruženja

Izrada 3D okruženja projekta S.S. Infinity zahtijevala je znanje 3D modeliranja, izrade tekstura, poznavanje osnova level dizajna za konačnu izradu 3D okruženja, te osnova kadriranja i video montaže za izradu animacije. 3D modeli i njihove teksture vizualno čine veliki postotak konačnog izgleda nekog 3D okruženja, te zajedno s korektnim i vješto izvedenim osvjetljenjem, čine sveukupan dojam i atmosferu 3D okruženja.

4.4.1. Izrada prostornog plana

Prije izrade 3D modela potrebno je bilo napraviti detaljan prostorni plan 3D okruženja i u grubo definirati kakve 3D objekte je bilo potrebno izraditi. Unreal Engine 4 editor ima ugrađene alate za izradu prototipa 3D okruženja, no nisam se koristio njima. Umjesto alata u Unreal Engine 4, koristio sam olovku i papir jer sam smatrao kako je to najbolji način da se stvori kreativno 3D okruženje. Na slici 4.8. nalazi se digitalno rekreirana verzija prostornog plana 3D okruženja.

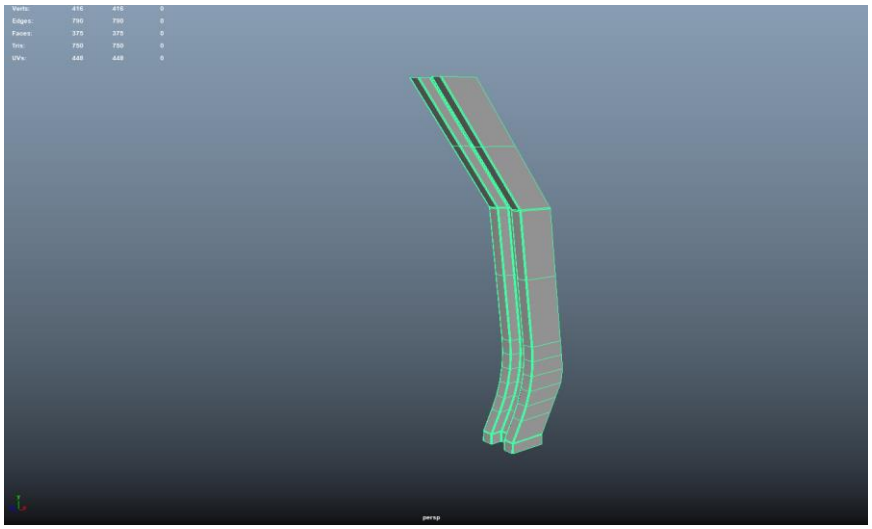


Slika 4.8. Rekreirani prostorni plan 3D okruženja.

4.4.2. Izrada 3D modela

Svi 3D modeli izrađeni za potrebe izrade 3D okruženja projekta S.S. Infinity izrađeni su Autodesk Maya 2016 programskim paketom. Nakon što je ugrubo skicom definiran prototip 3D okruženja, odlučio sam kako bi za izradu ovog 3D okruženja bilo najbolje krenuti s modularnim pristupom ili tako zvanim modularnim level dizajnom. Modularni level dizajn podrazumijeva korištenje zasebnih 3D elemenata koji se nadovezuju i zajedno tvore jednu cjelinu, kao što je to u stvarnom svijetu. Glavni princip modularnog level dizajna je krenuti izrađivati 3D modele koji čine kutove 3D okruženja, zatim s 3D modelima zidova, podnica, stropova. Kada su svi ti 3D modeli izrađeni, i sa svim pripadajućim teksturama uvezeni u pokretač igre, tada se kreće s izradom 3D okruženja. Kada je glavni prostor izrađen prema prototipu, tada se kreće s izradom 3D modela rekvizita (eng. props) koji će upotpuniti prostor i tvoriti sveukupnu atmosferu 3D okruženja.

Prije početka izrade 3D modela, podesio sam mjerne jedinice i postavke mreže (eng. grid) u Maya 2016 kako bi odgovarale onima u Unreal Engine 4 s ciljem postizanja uniformiranosti 3D modela. Nakon nekoliko prikupljenih referentnih slika iz znanstveno fantastičnih serija i igara, započeo sam izradu 3D modela nosećeg stupa, koji je posljedično diktirao cjelokupan stil interijera broda. Pošto sam bio vremenski ograničen raditi dva do tri sata na jednom 3D modelu, odlučio sam izrađivati vizualno jednostavnije 3D modele. U sučelju Maya 2016 uključio sam prikaz broja poligona kako bi se ograničio na poligonalnu rezoluciju modela do 1000 poligona. Ta poligonalna rezolucija pokazala se izrazito učinkovita nakon izrade svih 3D modela. Za računalne igre je bitno da 3D modeli imaju što je moguće mani broj poligona uz što više moguće detalja, kako bi zauzimali manje resursa računala prilikom izvođenja igre.



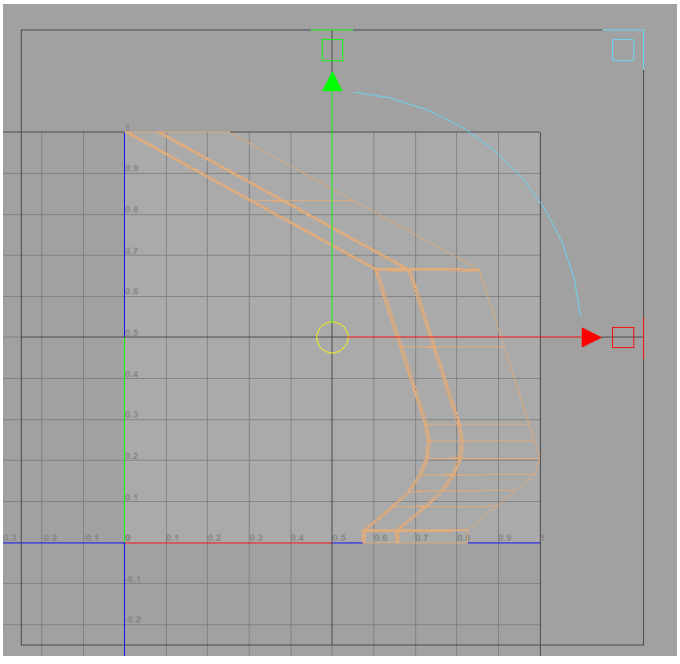
Slika 4.9. Prikaz 3D modela stupa.

Na svim izrađenim 3D modelima brisani su pozadinski poligoni, jer se tako štedi na poligonalnoj rezoluciji. Pozadinski poligoni ne vide se tijekom prikaza u pokretaču igre stoga je logično izbrisati ih radi uštede računalnih resursa. Također, olakšan je proces UV mapiranja.

UV mapiranje je proces izrade 2D koordinata kako bi se 2D slika ili tekstura mogla omotati oko 3D modela. U i V su nazivi za osi koordinatnog sustava, a odgovaraju X i Y osima koordinatnog sustava. U osnovi, kada 3D modelar izrađuje UV mapu, tada on polaže projekcije iz 3D prostora u 2D prostor. Tada računalo na osnovu tih projekcija prikazuje teksturu na 3D modelu. UV prostor je površina unutar koje je smještena tekstura i odgovara vrijednostima od 0 do 1 u U i V smjeru. To znači kako izrađena tekstura mora biti jednake *pixel* rezolucije po širini i dužini. Primjerice tekstura rezolucije 2048x2048px. 3D model može biti sastavljen od više zasebnih 3D objekata, tada bi se sve UV mape tih objekata posložile unutar istog UV prostora na organiziran način kako bi se maksimalno iskoristio 0-1 prostor. Maksimalno iskorištenje 0-1 prostora bitno je zbog oštrog prikaza tekstura na 3D modelu. Kako sam odlučio koristiti se modularnim level dizajnom, tako svaki 3D model ima svoj zaseban 0-1 prostor ili UV mapu. To nije najučinkovitiji pristup, ali u konačnici daje najveću fleksibilnost izrade 3D okruženja.

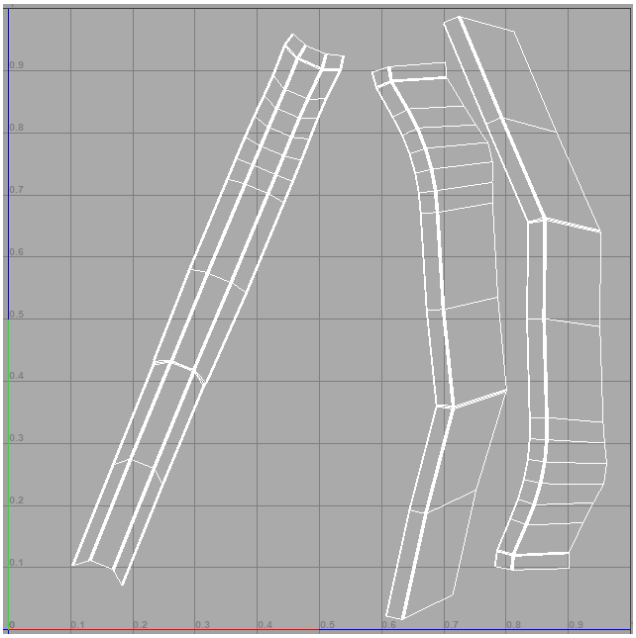
4.4.2.1. UV mapiranje

Postupak UV mapiranja je sljedeći. Prvo je potrebno selektirati cijeli 3D model i zatim odabrati *UV > Planar* alat za projekciju, te zatim u postavkama tog alata odabrati s koje osi u 3D prostoru se želi projicirati 2D slika u 0-1 prostor.



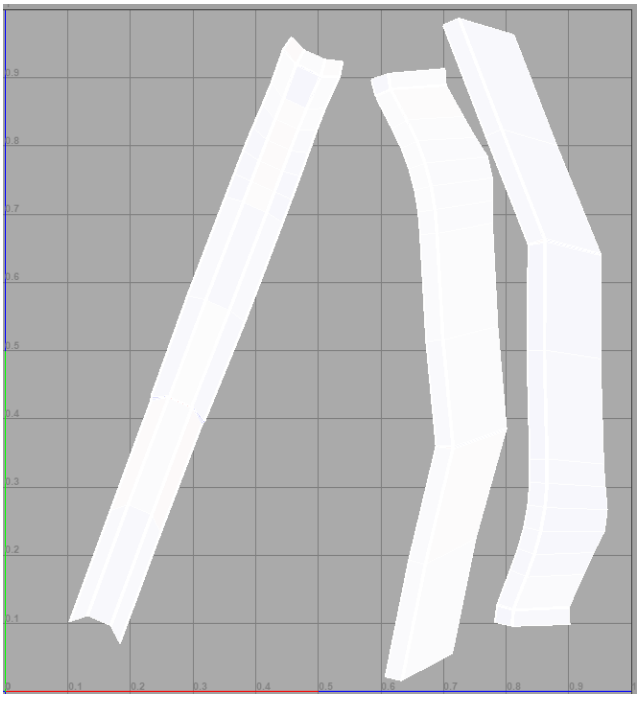
Slika 4.10. Projekcija sa Z osi.

Nakon što je napravljena projekcija u 0-1 prostoru, potrebno je selektirati rubne dijelove 3D modela. Zatim je potrebno unutar UV editora odabrati opciju *cut UV edges* kojom se od projekcije naprave podijeljene UV mape koje je potrebno odmotati naredbom *unfold* u te ih posložiti na organiziran način naredbom *layout*.



Slika 4.11. Odmotane UV mape naredbom unfold.

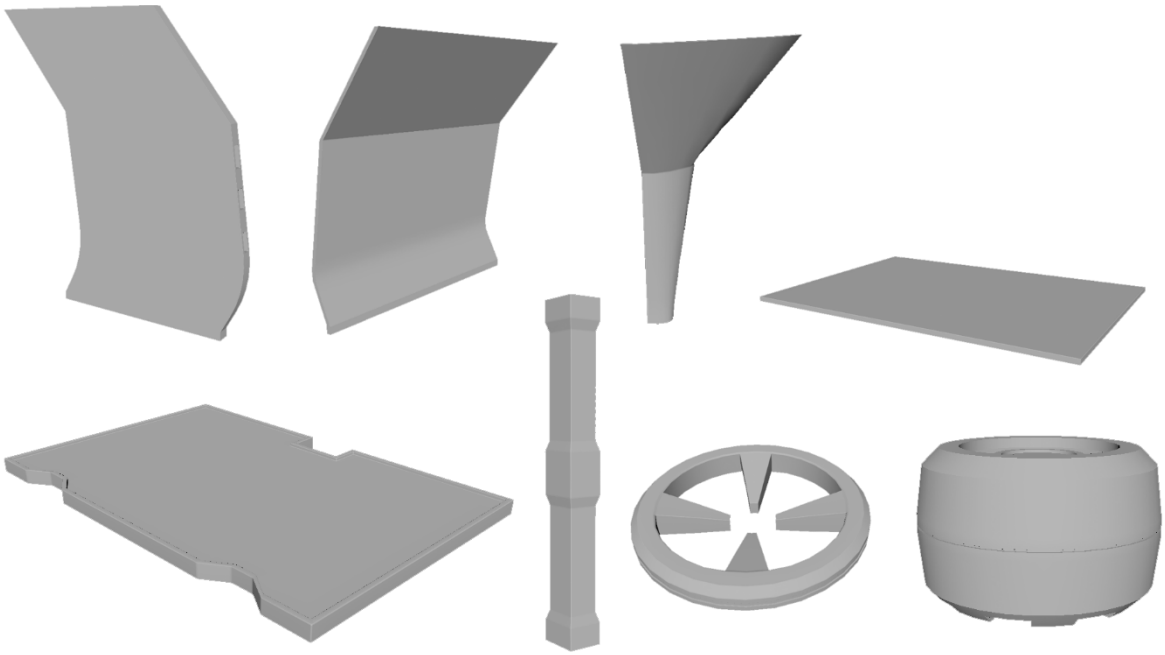
Naredba *unfold* doslovno odmata projiciranu sliku iz 3D prostora na osnovu UV koordinata dobivenih projekcijom. Bitno je provjeriti da li izrađene UV mape imaju izobličenja, tj. hoće li se primijenjena tekstura ispravno prikazivati na 3D modelu bez izobličenja.



Slika 4.12. Provjera izobličenja bojom u UV editoru.

Maya 2016 omogućuje provjeru izobličenja u UV editoru na nekoliko načina, uzorkom šahovnice ili bojama. Oba su jednako pouzdana i bitno ih je koristiti u kombinaciji. Na slici 4.12. prikazana je provjera izobličenja bojom unutar UV editora. Ovakva provjera izobličenja daje odlične rezultate uz veliku brzinu rada. Provjera izobličenja vrši se provjerom triju boja, bijele - nema izobličenja, plave - izobličenje rastezanjem, crvene - izobličenje skupljanjem. Na slici je vidljivo da provjera izobličenja pokazuje UV mape većinom bijele boje uz malo svijetlo plave boje. To izobličenje je prihvatljivo jer ga je nemoguće primijetiti jednom kada je tekstura primijenjena na 3D model.

Nakon izrade 3D modela stupa bilo je potrebno izraditi identičan 3D model veće poligonalne rezolucije. 3D model veće poligonalne rezolucije potreban je za izradu tekstura u Substance Painter 2 i Substance Designer 5. 3D model stupa veće poligonalne rezolucije može se izraditi jednostavnom naredbom *smooth* unutar Maya 2016. Daljnji tijek rada izrade tekstura bit će opisan u daljnjim poglavljima. Opisan postupak 3D modeliranja i UV mapiranja korišten je kod izrade svih 3D modela koji su korišteni za izradu 3D okruženja. Neki od 3D modela 3D okruženja prikazani su na slici 4.13.



Slika 4.13. Pojedini 3D modeli 3D okruženja S.S. Infinity.

4.4.3. Izrada tekstura

Za izradu tekstura korišteni su programski paketi Substance Designer 5 i Substance Painter 2. Unreal Engine 4 implementira *Metalic/Roughness* PBR način prikaza tekstura. PBR je fizički točan prikaz tekstura u pokretaču igre. Substance programski paketi implementiraju isti način prikaza tekstura, te su namijenjeni upravo izradi PBR tekstura. Iako su Substance programski paketi namijenjeni za izradu tekstura i uvelike ubrzavaju proces izrade istih, izrada tekstura i dalje zauzima puno vremena. Kako bi ubrzao izradu tekstura, koristio sam se tijekom rada između Substance Designer 5 i Substance Painter 2 koje preporučuje Allegorithmic, ali uz dodatne modifikacije. [9]

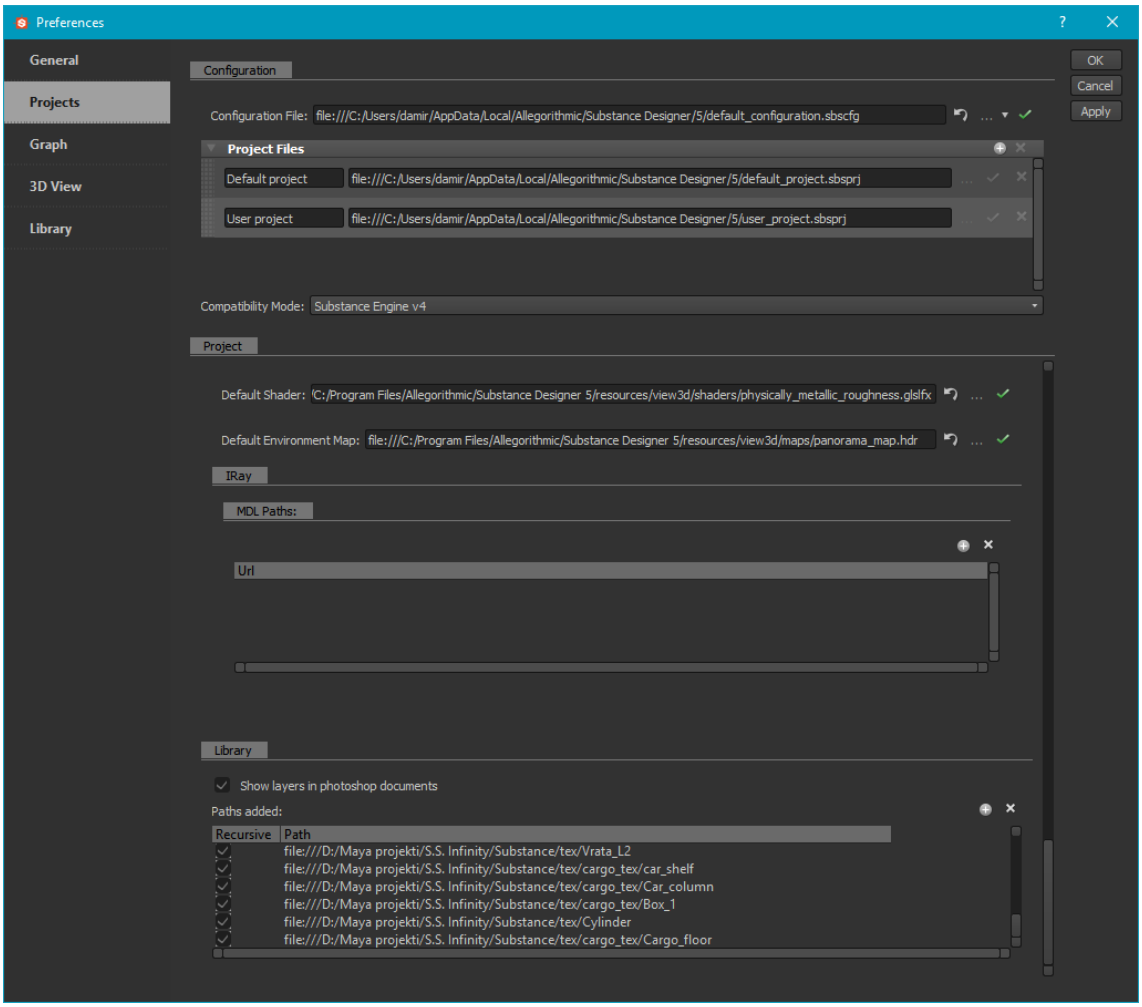
Metallic/roughness način izrade tekstura uključuje izradu sljedećih tekstura:

1. **Base color**
2. **Metallic**
3. **Roughness**
4. **Normal map**
5. **Emissive (dodatno)**
6. **Ambient oclusion (dodatno)**

Sve teksture moguće je automatski izrađivati istovremeno unutar Substance programskih paketa. Rad na PBR teksturama odvija se u obliku kanala u Substance Painter 2 ili *output* čvorova u Substance Designer 5. *Base color* tekstura sadrži informacije o boji, *Metallic* tekstura označava koliko je 3D model metalan, *Roughness* tekstura označava koliko je hrapav ili gladak, dok *Normal map* tekstura označava udubine i izbočine na 3D modelu. *Emissive* tekstura služi za isijavanje svjetla, dok *Ambient occlusion* tekstura služi za simulaciju sitnih sjena na 3D modelu. [9]

Kako bi povezano mogao raditi između Substance Designer 5 i Substance Painter 2, u Substance Designer 5 je bilo potrebno podesiti postavke knjižnice tako da su unutar nje vidljive mape u koje su izvezene teksture iz Substance Paintera 2. Jednom kada se to podesi, te postavke ostaju sačuvane za sve nove projekte koji se stvore u Substance Designeru 5 i moguće je povezano raditi između Substance programskih paketa.

U novo otvorenom projektu, u postavkama pod *Edit* > *Preferences* > *Library* potrebno je dodati mape izvezenih tekstura za sve 3D modele. Nakon toga, bit će moguć povezan rad između Substance programskih paketa.



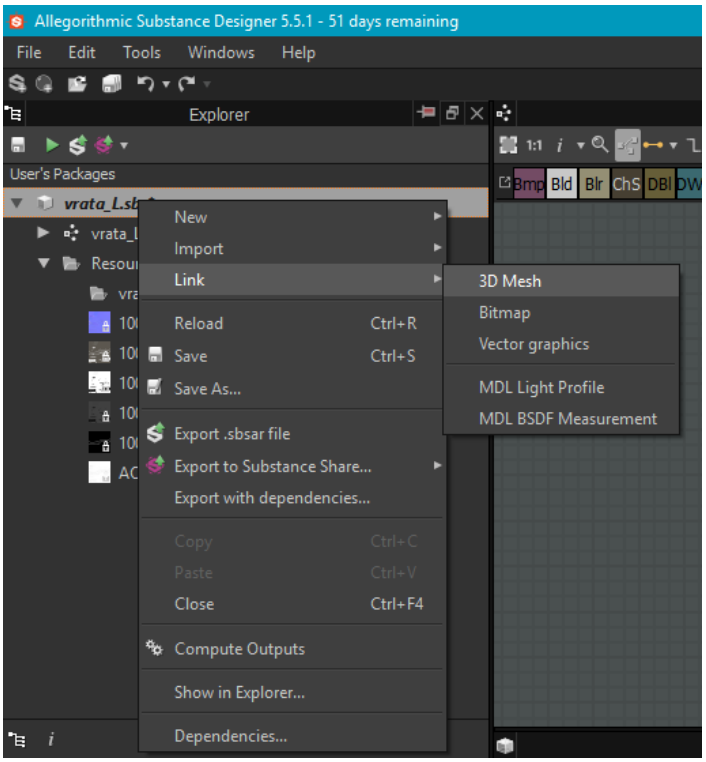
Slika 4.14. Povezivanje Substance programskih paketa.

4.4.3.1. Postupak izrade tekstura (Substance Designer 5)

Nakon što su povezani Substance programski paketi, potrebno je otvoriti novi projekt s nazivom 3D modela za koji će se izraditi paket tekstura razumljiv Unreal Engine 4. Tijekom izrade tekstura, naizmjenice su korištena oba Substance programska paketa. Substance Designer 5 je korišten prvotno za izradu dodatnih tekstura, potrebnih za rad u Substance Painter 2, dok je Substance Painter 2 korišten za izradu konačnih tekstura 3D modela.. Substance Painter 2 za izradu PBR Metallic/Roughness tekstura treba *Normal map* teksturu od koje će kasnije generirati ostale teksture: *World space normals*, *Ambient Occlusion*, *Curvature*, *Position* i *Thickness*. Te dodatne teksture ne izvažaju se za potrebe prikaza u pokretaču igre, već služe Substance Painter 2 grafičkim algoritmima za generiranje raznih efekata na osnovnim teksturama, poput ogrebotina po rubovima, hrđe i dr.

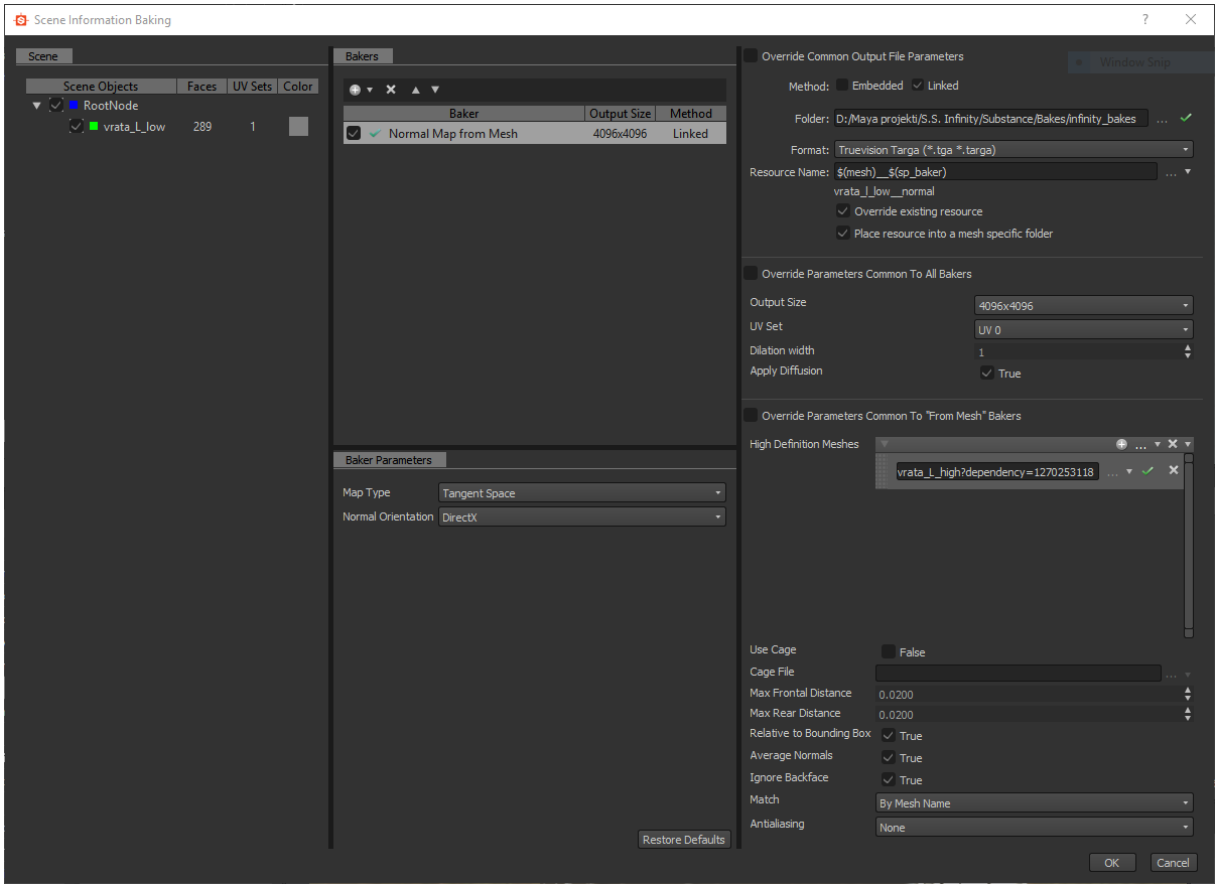
Normal map tekstura sadrži informacije o normalama, matematičkim linijama koje su smještene okomito na poligone 3D modela. *Normal* linije govore grafičkom sustavu pokretača igre na koji se način treba odbijati svjetlo od 3D objekta. Drugim riječima, *normal* linije opisuju 3D model u 3D prostoru. U industriji računalnih igara, često se koristi tehnika izrade *normal map* tekstura koje sadrže *normal* informacije od 3D modela s velikom poligonalnom rezolucijom. Tako izrađena *normal map* tekstura se onda primjenjuje na 3D model niske poligonalne rezolucije kako bi se prikazivao kao onaj s visokom poligonalnom rezolucijom. Ta tehnika uvelike štedi računalne resurse i omogućuje prikaz vizualno atraktivnijih 3D modela, jer im je omogućen prikaz sa zaglađenim rubovima. Oba Substance programska paketa implementiraju mogućnost izrade *normal map* tekstura koje sadrže *normal* informacije od 3D modela visoke poligonalne rezolucije. Koristio sam onaj implementiran u Substance Designer 5 radi veće fleksibilnosti i kasnijeg pakiranja svih tekstura. [10]

Prvo što je potrebno kod izrade tekstura za 3D model u Substance programskim paketima je izraditi *normal map* i *curvature* teksture. Za primjer uzimam 3D model lijevih vrata. Prvo je bilo potrebno uvesti oba 3D modela (niske i visoke poligonalne rezolucije). Uvažanje 3D modela u Substance Designer 5 se radi tako što se na desni klik miša na glavnu mapu projekta otvori izbornik link u kojem je potrebno odabrati *3D mesh*.



Slika 4.15. Uvoz 3D modela u Substane Designer 5.

Nakon uvoza 3D modela, bilo je potrebno izraditi *normal map* i *curvature* teksture koje će biti uvezene u Substance Painter 2. Izrada *normal map* teksture izvodi se u potprogramu (eng. baker) Substance Designera 5 za izradu *normal map* teksture na sljedeći način. Prvo je potrebno pristupiti potprogramu desnim klikom miša na 3D model niske poligonalne rezolucije i odabrati opciju *bake model information* u padajućem izborniku.

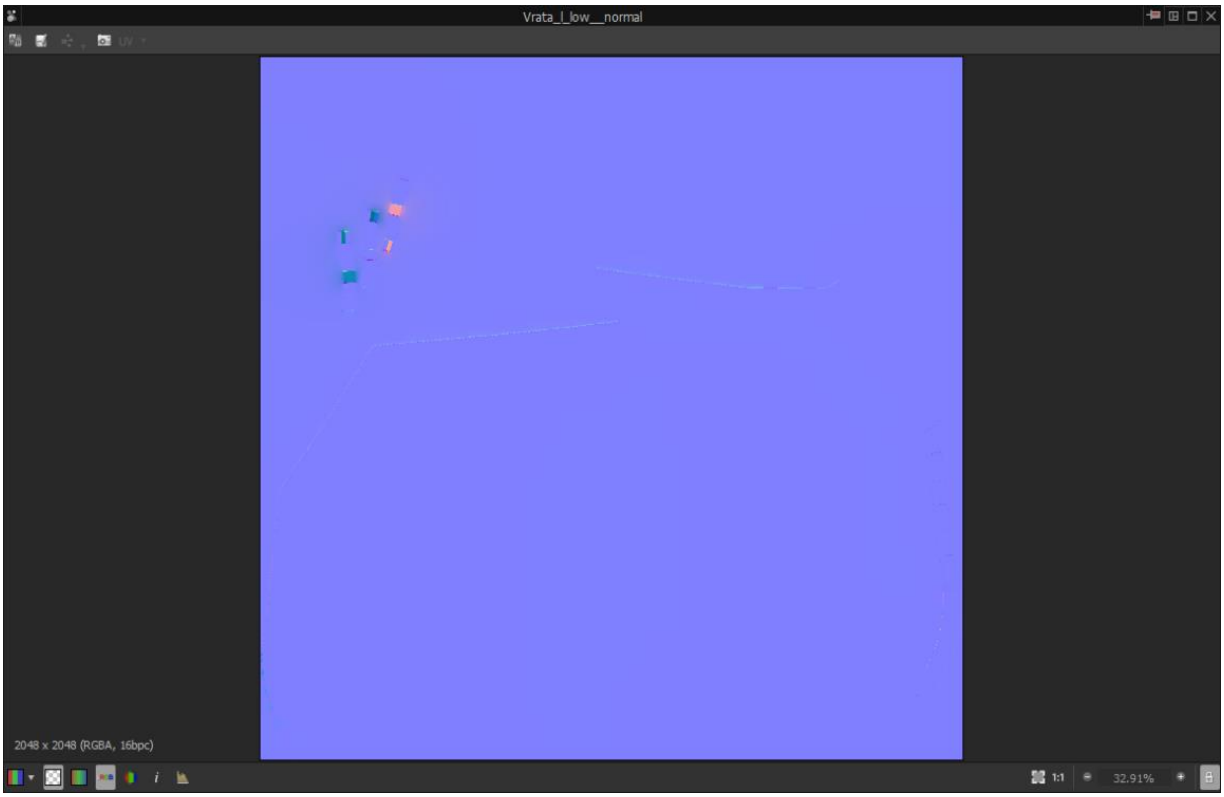


Slika 4.16. Potprogram za izradu normal map tekstura u Substance Designer5. [10]

Potrebno je u potprogramu za izradu *normal map* teksture podesiti lokaciju na kojoj će se pospremiti izrađena *normal map* tekstura. Zatim, sljedeći korak je podešavanje konvencije imenovanja. Substance programski paketi omogućuju automatsko imenovanje datoteka korištenjem posebne sintakse. Potrebno je u polje *Resource name* upisati *$(mesh)_$(sp_baker)* kako bi Substance Designer automatski imenovao *normal map* teksturu onako kako je imenovan 3D model, te će nakon toga dodati sufiks *_normal*. Nakon što je podešeno automatsko imenovanje datoteke, potrebno je podesiti izlaznu rezoluciju. Odabrana izlazna rezolucija je 4096x4096px radi oštrih detalja koji će biti izrađeni u Substance Painter 2. Pod odjeljkom *High*

Definition Meshes odabire se 3D model visoke poligonalne rezolucije od kojeg će se preuzeti *normal* informacije. Nakon toga potrebno je podesiti minimalnu i maksimalnu udaljenost zrake koja projicira *normal* informacije iz 3D prostora u 2D prostor. Kako sam na početku definiranja projekta odredio stvarne jedinice u centimetrima, izradio sam 3D modele u pravoj veličini. Izrada 3D modela u pravoj veličini omogućila je olakšanu izradu *normal map* teksture, bez izobličenja. Minimalna i maksimalna udaljenost je podešena na 0.02 cm. Postavke *relative to bounding box*, *average normals* i *ignore backface* nisu mijenjane, jer za to nije bio potrebe.

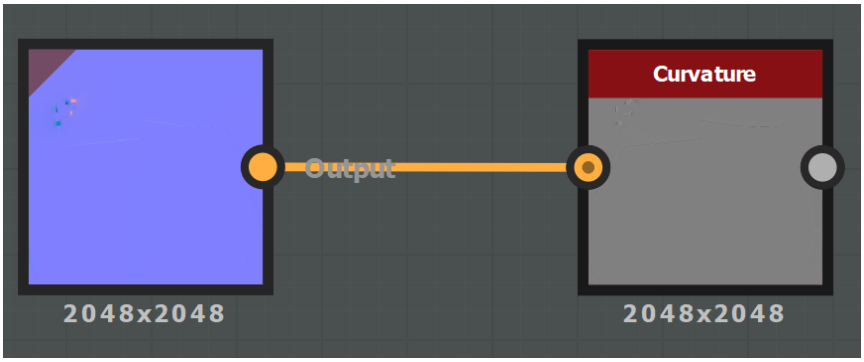
Bitna postavka kod izrade *normal map* teksture u Substance programskim paketima je postavka načina na koji će se *normal* informacije s 3D modela visoke poligonalne rezolucije prenijeti iz 3D u 2D prostor. Postoje dva načina za to, po imenu 3D modela (eng. by mesh name) ili uvijek (eng. always). Ako se odabere postavka da se *normal* informacije prenose uvijek, tada će se one projicirati u 2D prostor bez uzimanja u obzir okolnih 3D objekata. Naime, 3D model niske i visoke poligonalne rezolucije dijele identične 3D koordinate, tako da su kod ove postavke vrlo vjerojatno moguća izobličenja u *normal map* teksturi. Kod postavke po imenu 3D objekta, potprogram za projiciranje *normal* informacija razdvaja oba 3D modela i tek onda vrši projekciju. Za ovu postavku, potrebno je koristiti konvenciju imenovanja 3D modela niske i visoke poligonalne rezolucije definirane na počeku razvoja projekta.



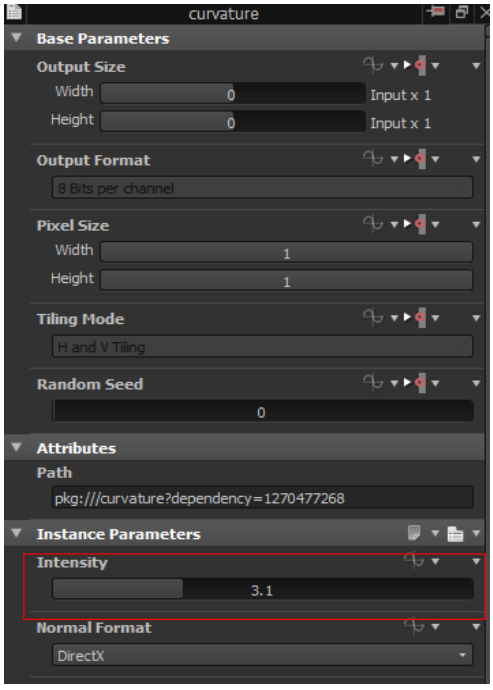
Slika 4.17. Izrađena normal map tekstura u Substance Designer 5.

Izrada Curvature teksture u Substance Designer 5 mnogo je jednostavnija od izrade *normal map* teksture. Substance Designer 5 implementira vizualno skriptni kod u obliku čvorišta kojima je moguće stvoriti kompleksne mreže za prikaz visoko realističnih materijala. *Curvature* teksturu moguće je izraditi sa samo dva čvorišta, *bitmap* i *curvature*. Kada se iz knjižnice prenese u radni prostor neka tekstura, ona automatski dobije svoje bitmap čvorište, s kojim je kasnije moguće raditi. Na slici 4.18. prikazan je način izrade *curvature* teksture pomoću *normal map* teksture.

Intenzitet *curvature* teksture moguće je podešavati odjeljku sučelja s osnovnim parametrima. Intenzitet za sve izrađene *curvature* teksture projekta S.S. Infinity bio je podešen na 3.1 zbog najboljih rezultata.



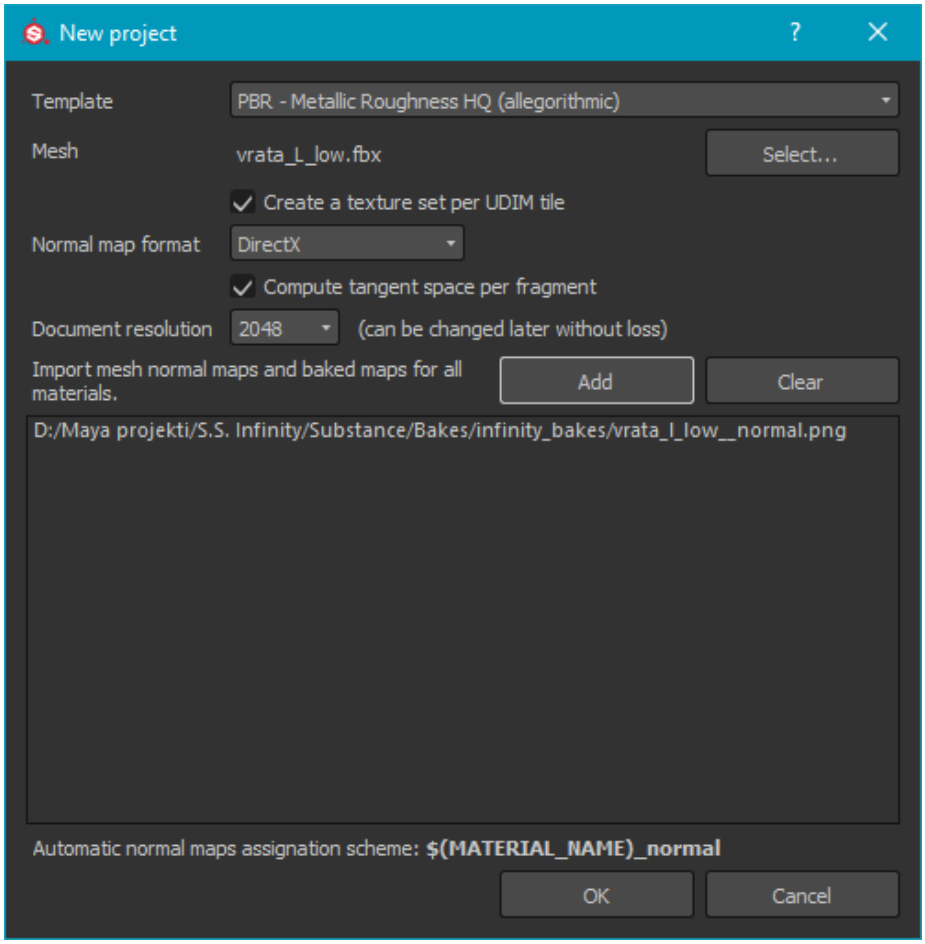
Slika 4.18. Izrada curvature teksture u Substance Designer 5.



Slika 4.19. Podešavanje intenziteta *curvature* teksture.

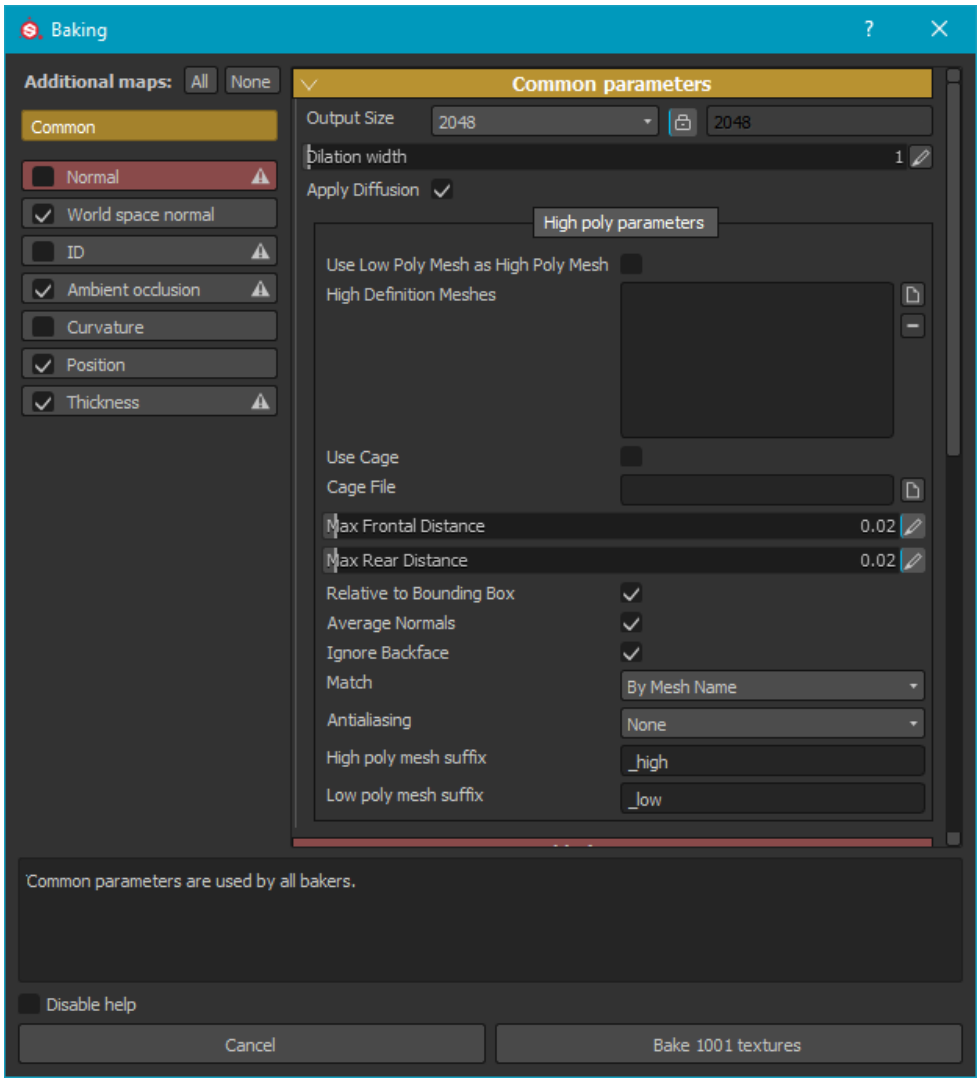
4.4.3.2. Postupak izrade tekstura (Substance Painter 2)

Nakon što su izrađene *normal map* i *curvature* teksture, potrebno je bilo otvoriti novi dokument u Substance Painter 2 i u prozoru za stvaranje novog dokumenta odabrati 3D model niske poligonalne rezolucije i izrađenu *normal map* teksturu. Izrađena *normal map* tekstura automatski će biti pridružena *normal* kanalu zbog definirane konvencije imenovanja u Substance Designer 5.



Slika 4.20. Stvaranje novog projekta u Substance Painter 2

Odmah nakon stvaranja novog projekta, potrebno je uvesti *curvature* teksturu u policu, koja će se automatski pridružiti pripadajućem kanalu. Nakon što je projekt uspješno stvoren, potrebno je stvoriti ostale dodatne teksture koje su potrebne za izradu osnovnih tekstura. Dodatne teksture izrađuju se u prozoru *Bake textures* koji je prikazan na slici 4.21.

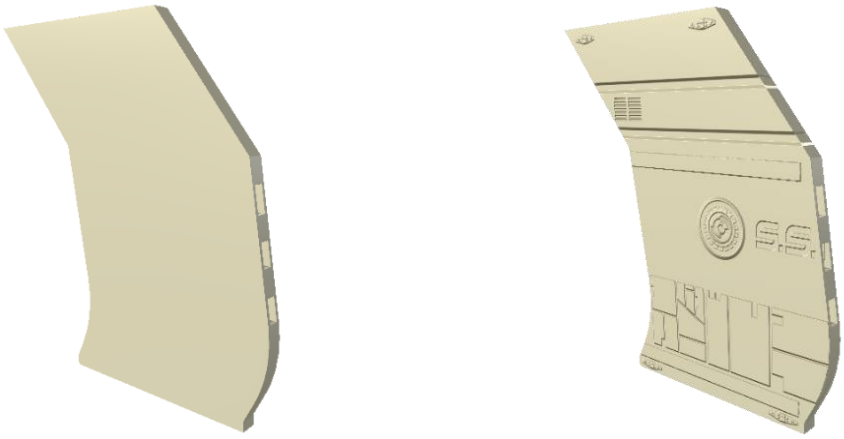


Slika 4.21. Bake textures prozor u Substance Painter 2

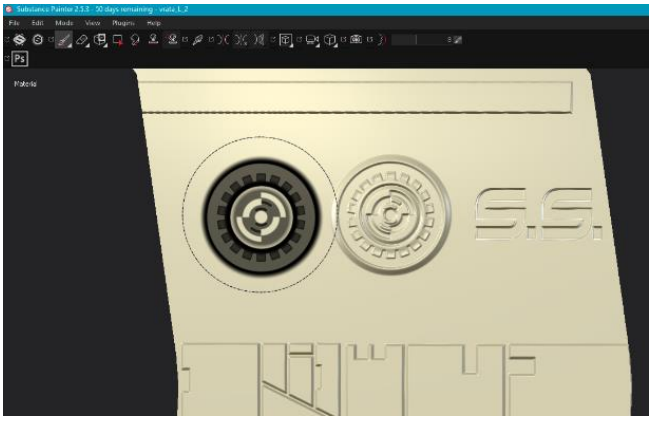
U prozoru *bake textures* automatski su odznačene *normal map* i *curvature* teksture zbog konvencije imenovanja u Substance Designer 5. U ovom prozoru nije potrebno učitavati 3D model visoke poligonalne rezolucije, jer su sve potrebne informacije s tog modela već projicirane u *normal map* i *curvature* teksturama, iz kojih će Substance Painter 2 izraditi sve potrebne teksture (eng. *world space normal*, *ambient occlusion*, *thickness* i *position*).

Nakon što postoje sve potrebne dodatne teksture za rad Substance Painter 2, moguće je izrađivati teksture koje će biti izvezene za uporabu s Unreal Engine 4. Svoj proces izrade tih tekstura podijelio sam na nekoliko faza: izrada *height* detalja, izrada osnovnih materijala, izrada materijala isijavanja svjetlosti (eng. emmisive). Izrada tekstura u te tri faze uvelike mi je ubrzala rad jer sam se mogao fokusirati samo na te tri faze izrade. Izrađene teksture su time dobile uniformiran vizualni stil.

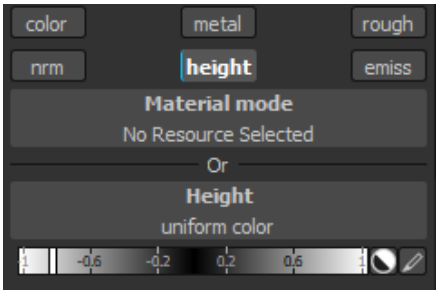
Za izradu *height* detalja potrebno je bilo stvoriti jedan *fill* sloj i uključiti mu samo *height* kanal. Pod *height* detaljima podrazumijevaju se detalji udubina i izbočina na 3D modelu. Zatim je bilo potrebno stvoriti novi klasičan sloj kojem se također trebalo uključiti samo *height* kanal, te ga postaviti iznad *fill* sloja. Tako se postigla kontrola nad *height* detaljima i ubrzan je rad u toj fazi izrade tekstura. Na klasičnom sloju su se zatim pomoću kist alata koji koristi *alpha* kanal teksturu izrađenu u Adobe Photoshopu izradili detalji na *normal map* teksturi pomoću *hieght* kanala. Primjer kist alata s *alpha* kanal teksturom prikazan je na slici 4.23. Visinu udubina ili izbočina moguće je kontrolirati podešavanjem vrijednosti ispod naziva kanala u vrijednostima od 0 do 1 u pozitivnom i negativnom smjeru. Primjer podešavanje visine udubina ili izbočina prikazan je na slici 4.24.



Slika 4.22. 3D model bez height detalja lijevo, 3D model sa height detaljima desno.



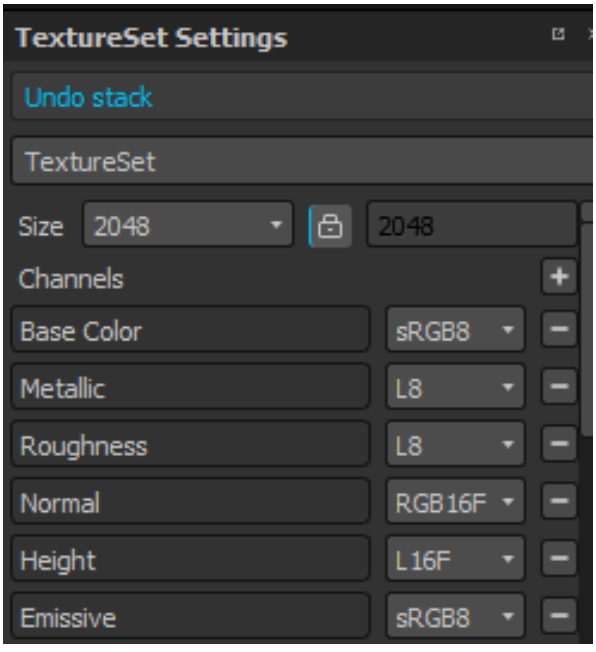
Slika 4.23. Kist alat s alpha kanal teksturom.



Slika 4.24. Podešavanje visine udubina i izbočina.

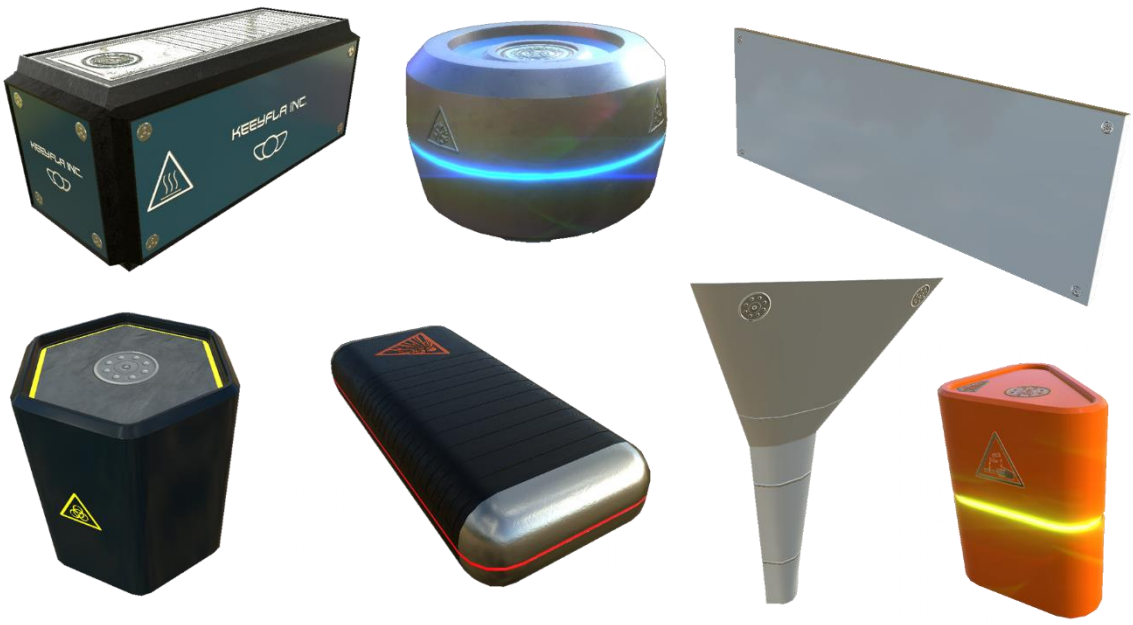
Nakon izrade *height* detalja, potrebno je bilo stvoriti novu mapu koja sadrži novi *fill* sloj s predefiniranim oštećenim metalom. Za izradu osnovnih materijala poput metala, koristio sam predefinirane materijale koje sam koristio za prijašnje projekte. Substance Painter 2 s lakoćom dopušta pristup prijašnje stvorenim materijalima što mi je uvelike ubrzalo izrađivanje tekstura.

Poslije stvaranja materijala oštećenog metala, dodan je novi *fill* sloj s materijalom sjajne plastike kojem je neznatno korigiran sjaj na *roughness* kanalu. Pomoću kist alata s *alpha* teksturom, materijal plastike nanesen je samo na izbočena područja. Nanošenje materijala plastike omogućeno je zbog *aplha* kanal teksture koja se primjenjuje na kistu, a koja djeluje kao maska između slojeva. Odmah nakon završetka bojanja materijala plastike, stvoren je novi *fill* sloj s *emissive* kanalom bijele boje. *Emissive* kanal po otvaranju novog projekta u Substance Painter 2 nije omogućen pa ga je potrebno omogućiti u postavkama skupa tekstura, što je prikazano na slici 4.25. Ponovno je pomoću kista nanesen taj isijavajući materijal čime je proces teksturiranja 3D modela lijevih vrata bio završen. Ovakav proces je ponovljen za sve 3D modele koji tvore 3D okruženje projekta S.S. Infinity. Neki od 3D modela s izrađenim teksturama prikazani su na slici 2.7.



Slika 4.25. Uključivanje emissive kanala.

Slika 4.26. 3D model lijevih vrata sa završenim teksturama.



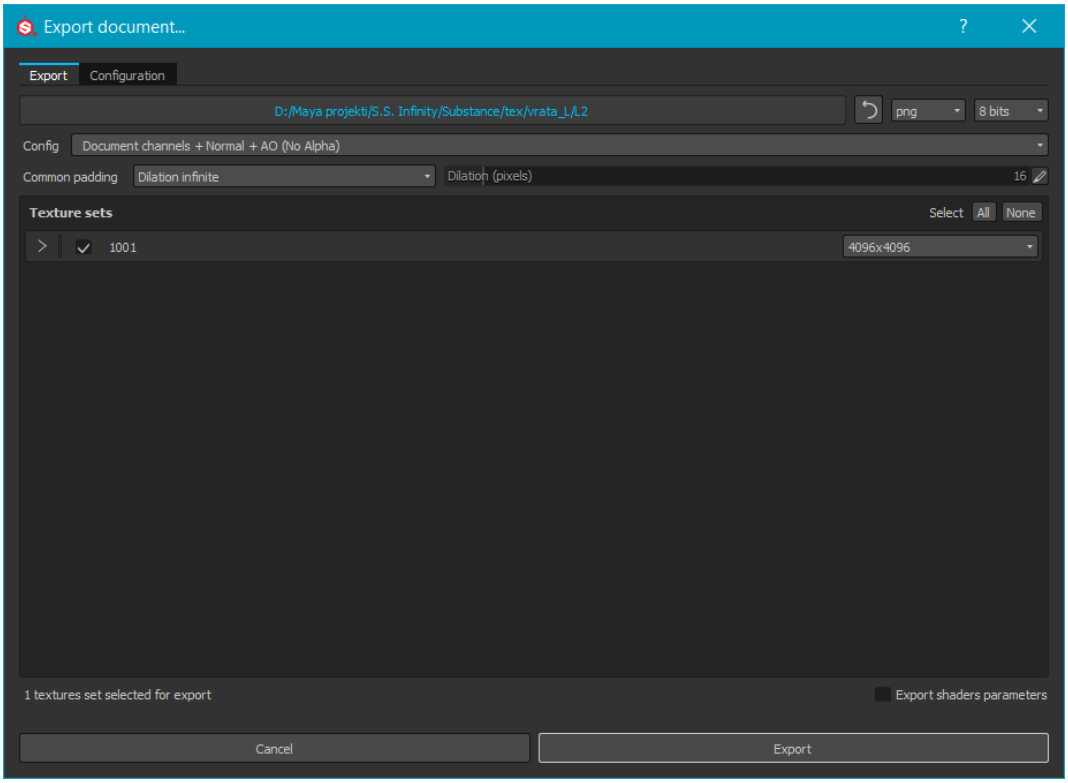
Slika 4.27. Neki od 3D modela s izrađenim teksturama.

4.4.3.3. Izvoz i pakiranje tekstura za Unreal Engine 4

Unreal Engine 4 podržava Allegorithmicov programski dodatak za Substance Designer 5. Substance programski dodatak za Unreal Engine 4 omogućuje otvaranje paketa tekstura stvorenih u Substance Designer 5. Unreal Engine 4 implementira *material editor* u zasebnom prozoru unutar kojeg se pomoću vizualno skriptnog računalnog koda upravlja grafičkim sustavom Unreal Engine 4. Kako Unreal Engine 4 implementira *Metallic/Roughness* način prikaza PBR tekstura, tako uključuje ista čvorišta na koje je potrebno spojiti sve potrebne teksture stvorene Substance Designer 5 programskim paketom. Substance programski dodatak za Unreal Engine 4 omogućuje automatsko izrađivanje materijala za 3D model što znatno ubrzava rad s dodjeljivanjem tekstura 3D modelu u *material editoru*. Materijal u Unreal Engine 4 je naziv za pakirani vizualno skriptni računalni kod koji definira izgled 3D modela u 3D okruženju. [13]

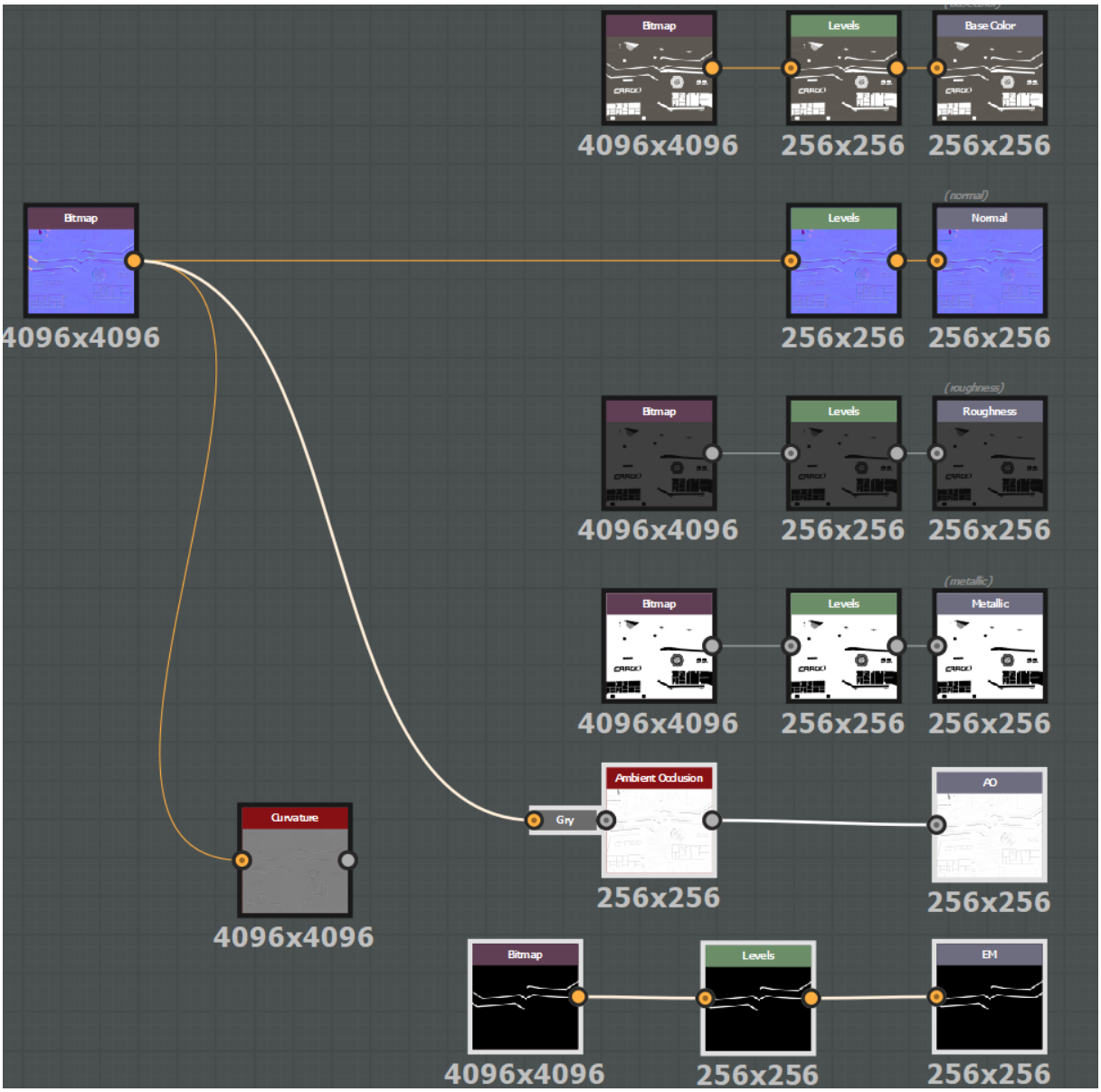
Za pakiranje i izvoz svih tekstura pojedinog 3D modela u jednu datoteku, bitno je podesiti osnovne rezolucije tekstura u Substance Designer 5 i odrediti im način skaliranja kako bi se unutar Unreal Engine 4 *material editora* moglo upravljati s rezolucijom svih tekstura odjednom. Teksture stvorene Substance Programskim paketima izrađene su proceduralno što omogućuje njihovo skaliranje bez gubitka kvalitete prikaza.

Nakon što su teksture u Substance Painter 2 izrađene, potrebno je napraviti izvoz tih tekstura u mapu koja će kasnije biti vidljiva u Substance Designer 5 knjižnici. Proces pakiranja tekstura unutar Substance Designer 5 značajno ubrzava ranije definirana povezanost knjižnice sa Substance Painter 2 mapama za izvoz. Izvoz tekstura iz Substance Painter 2 prikazan je na slici 4.28.



Slika 4.28. Izvoz tekstura iz Substance Painter 2.

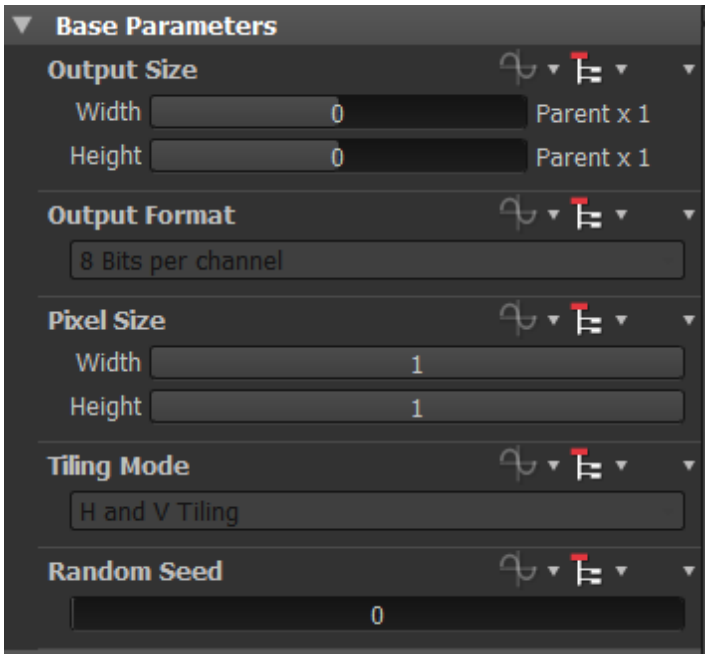
Pristup prozoru za izvoz tekstura u Substance Painter 2 omogućen je u padajućem izborniku *File > export textures* ili prečicom na tipkovnici pritiskom tipaka *ctrl + shift + E*. U tom prozoru je potrebno definirati putanju mape za izvoz na hard disku računala, format datoteke i rezoluciju tekstura. Za izvoz tekstura koristio sam 8-bit png format zbog male kompresije i 4096x4096px rezoluciju ili tako zvani 4K. Nakon izvoza tekstura, u definiranoj mapi za izvoz nalaze se sve potrebne teksture za prikaz 3D modela u Unreal Engine 4 (eng. *Base color*, *Metallic*, *Roughness*, *Normal map*, *Emissive*). *Ambient occlusion* teksturu izrađivao sam u Substance Designer 5 radi bolje kontrole nad željenim detaljima. Proces izrade *Ambient occlusion* teksture moguć je na identičan način kao kod ranije opisanog procesa izrade *curvature* teksture. Potrebno je uvesti *curvature* teksturu u radni prostor Substance Designer 5 i spojiti je s *Ambient occlusion* čvorištem. Kako bi *Ambient occlusion* bila omogućena u Unreal Engine 4, potrebno je za nju stvoriti izlazno čvorište i spojiti bitmap čvorište s njim. Također, sve pripadajuće teksture izrađene u Substance Painter 2 je potrebno spojiti na njihova izlazna čvorišta što je prikazano na slici 4.29. [11]



Slika 4.29. Mreža čvorišta u Substance Designer 5.

Isti postupak kao kod stvaranja izlaznog čvorišta *Ambient occlusion* teksture potrebno je bilo ponoviti i za *Emmisive* teksturu.

Još jedna bitna postavka u Substance Designer 5 kod pakiranja tekstura za Unreal Engine 4 je podesiti relacije između čvorišta koji sadrže izvorne teksture izrađene u Substance Painter 2 i izlaznih čvorišta. Relacije između izvornih i izlaznih čvorišta određuju mogućnost skaliranja tekstura jednom kada se uvezu u Unreal Engine 4. Definiranje relacija čvorišta izvodi se tako što se cijelom grafu definira relacija *relative to parent*, koja će postaviti izvorni čvor na maksimalnu rezoluciju. Nakon što je cijelom grafu definirana relacija skaliranja, potrebno je na izlaznim čvorovima definirati *relative to input*. Definiranje relacija skaliranja omogućeno je u odjeljku osnovnih parametara (eng. base parameter), što je prikazano na slici 4.30.



Slika 4.30. Podešavanje odnosa skaliranja tekstura u Substance Designer 5.

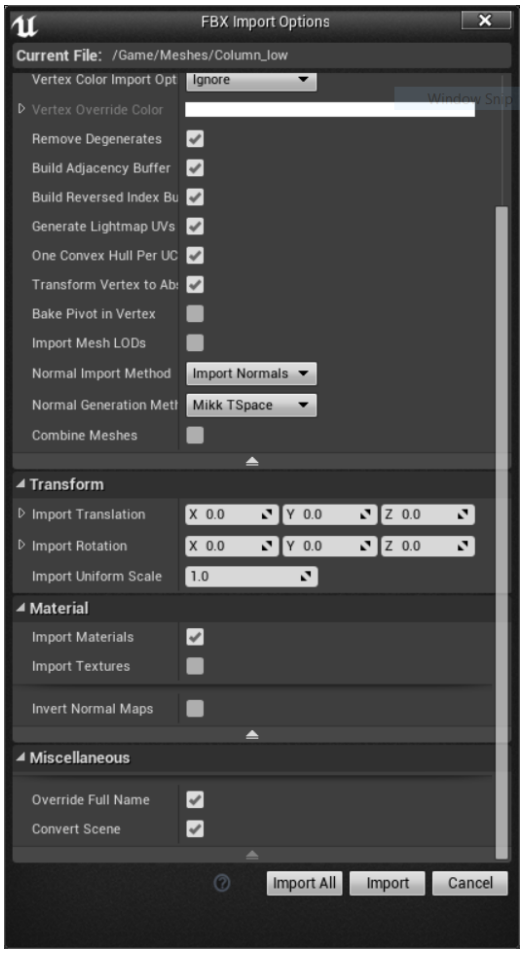
Nakon podešenih postavki skaliranja tekstura, one su spremne za pakiranje u jednu datoteku koja će kasnije biti uvezena Substance programskim dodatkom u Unreal Engine 4. Pakiranje tekstura izvodi se tako što se desnim klikom miša na glavnu mapu projekta pristupi *export .sbsar file* prozoru i pospremi u predefiniranu mapu. Substance Designer 5 će automatski stvoriti Unreal Engine 4 materijal i povezati sve teksture na potrebna čvorišta. U Unreal Engine 4 *material editoru* bit će omogućen odabir rezolucije tekstura. Zbog prirode proceduralnih tekstura, vrlo je lako imati u istom projektu više tekstura različitih rezolucija i tako uštedjeti na računalnim resursima. [12]

Izrada tekstura za projekt S.S. Infinity velik je dio sveukupnog tijeka rada na tom projektu. Razvoj organiziranog tijeka rada prilikom izrade tekstura omogućio mi je ubrzanu izradu istih. Važno je napomenuti kako razvojne kuće u industriji igara zapošljavaju i po nekoliko ljudi koji se bave izradom tekstura. Izrada tekstura za 3D modele izrazito je kreativan posao koji zahtjeva veliku maštu i odlično poznavanje programskih paketa za izradu tekstura. Substance programski paketi omogućuju izradu izrazito kreativnih tekstura prilikom čega bitno olakšavaju rad. Teksture izrađene za projekt S.S. Infinity su jednostavno izrađene teksture u tri faze, spomenute ranije. Zbog izrade tekstura u tri faze, one imaju specifičan grafički stil koji sam osmislio tijekom izrade tekstura. Opisan proces izrade i pakiranja tekstura koristio sam za sve 3D modele koji tvore 3D okruženje projekta S.S. Infinity.

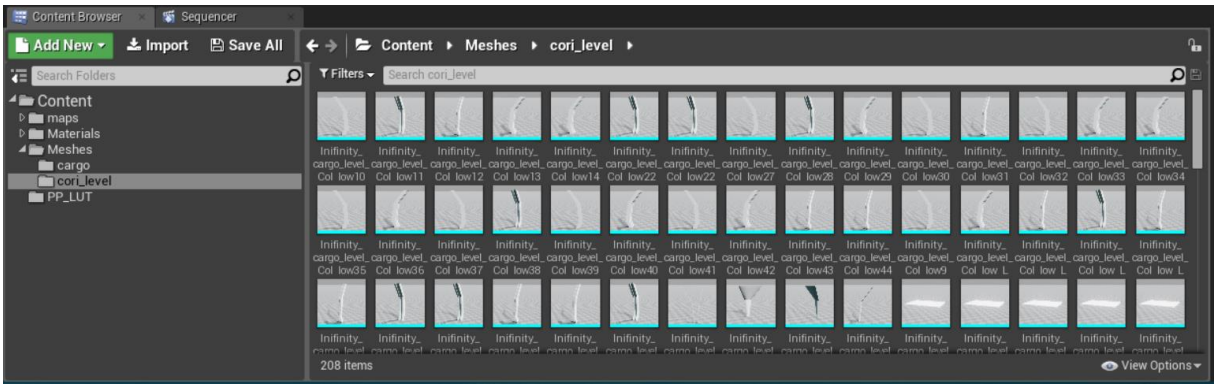
4.4.4. Uvoz 3D modela i pripadajućih tekstura u Unreal Engine 4

Unreal Engine 4 implementira upravitelj mapama i datotekama projekta pod nazivom *content browser*, koji omogućuje izvrsnu organizaciju projekta. *Content browser* upravlja velikim brojem mapa koje sadrže velik broj različitih datoteka potrebnih za izradu 3D okruženja, poput specijalnih efekata, animacija likova, 3D modela, tekstura, materijala, i mnogo drugih. *Content browser* kod uvoza datoteka automatski prepoznaje njihov tip, i u skladu s time otvara specifične prozore za njih. Također, jednom kada su datoteke uvezene, vrlo lako ih je pretraživati, ponovo učitavati i seliti u druge mape. [14]

Za uvoz 3D modela i pripadajućih tekstura potrebno je obaviti nekoliko koraka. Prvo je potrebno uvesti 3D modele u prethodno definiranu mapu *meshes*. Uvoz datoteka obavlja se klikom miša na gumb *import* u *content broswseru*. Nakon klika mišem na gumb *import*, potrebno je locirati *fbx* datoteke koje sadrže izvezene 3D modele izrađene Autodesk Maya 2016 programskim paketom i kliknuti na gumb *import all* kako bi se uvezli 3D modeli s osnovnim materijalima koji su pakirani s 3D modelima. Isti postupak je potrebno ponoviti i sa Substance paketima tekstura.

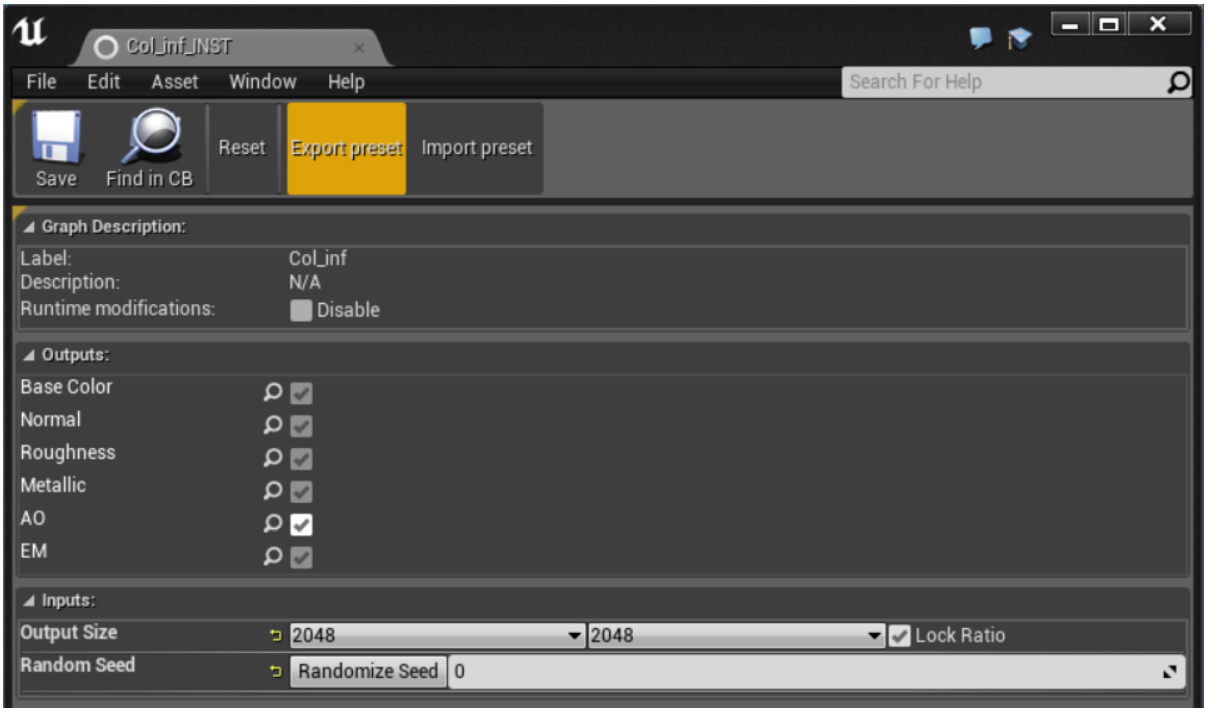


Slika 4.31. Uvoz 3D modela u mapu meshes u content browseru.



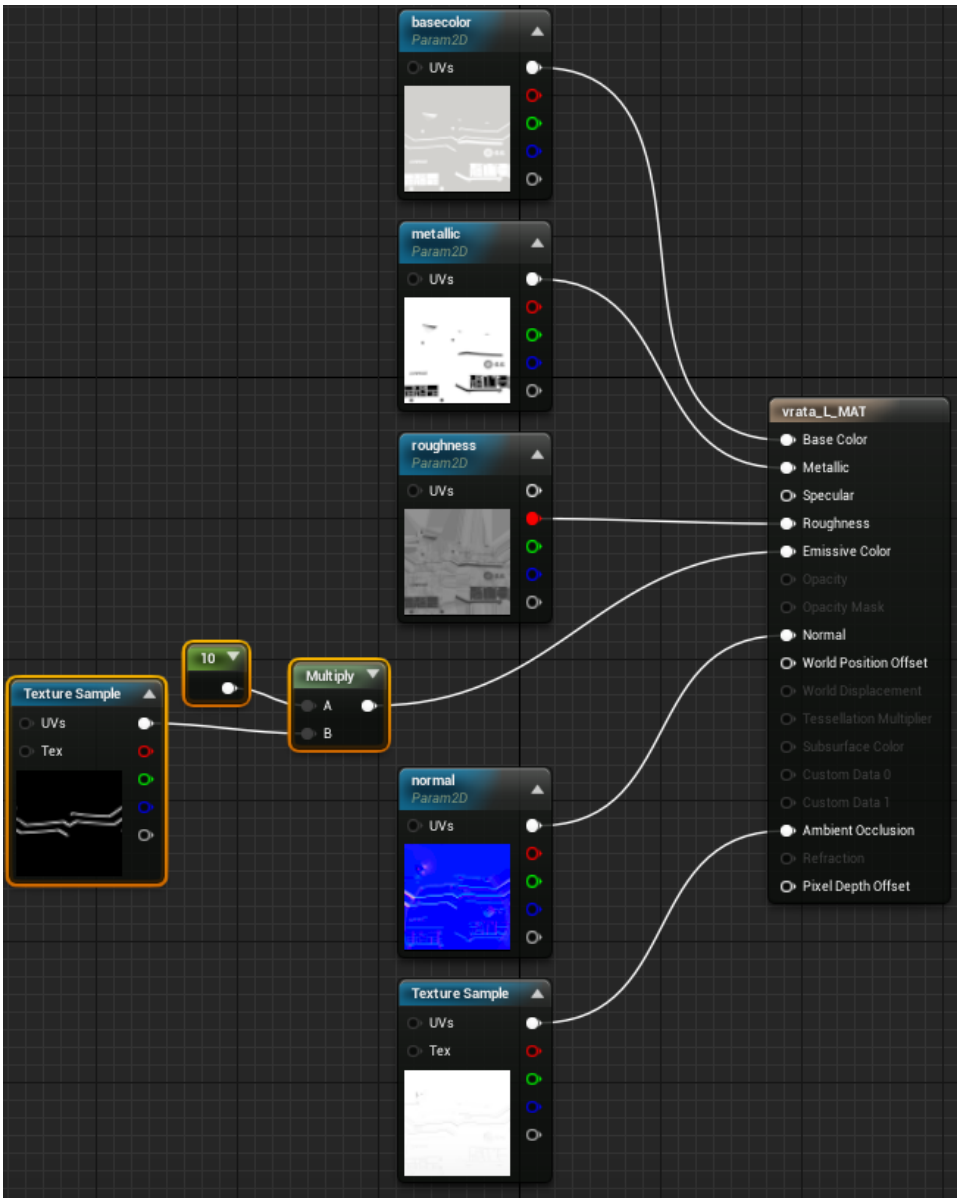
Slika 4.32. Uvezeni 3D modeli koji služe za izradu koridora u 3D okruženju.

Uvezeni 3D modeli organizirani su u mape *cargo* i *cori_level*, dok su njihove pripadajuće teksture smještene u mapu *materials* i podmape s nazivima 3D modela kojima pripadaju. Nakon uvoza paketa tekstura, Substance programski dodatak za Unreal Engine 4 stvorio je materijal s bitmap čvorištima koja sadrže sve teksture i spojio ih na odgovarajuća mjesta u glavnom čvorištu. Iako je ovaj programski dodatak napravio većinu posla koje bi inače ručno morao raditi sam i tako značajno ubrzao rad, i dalje je potrebno pojedine postavke podesiti kako bi se teksture ispravno prikazivale na 3D modelu. Potrebno je otvoriti uvezeni paket tekstura u mapi koja nosi naziv po 3D modelu kojem pripadaju. Zatim je potrebno odabrati dodatne teksture i podesiti željenu rezoluciju u prozoru koji se otvori nakon klika miša na ikonu paketa tekstura u *content browseru*. Odabir dodatnih tekstura omogućen je u odjeljku *outputs*, dok je odabir rezolucije u odjeljku *output size*. Odabir dodatnih tekstura i sveukupne rezolucije tekstura prikazan je na slici 4.33.

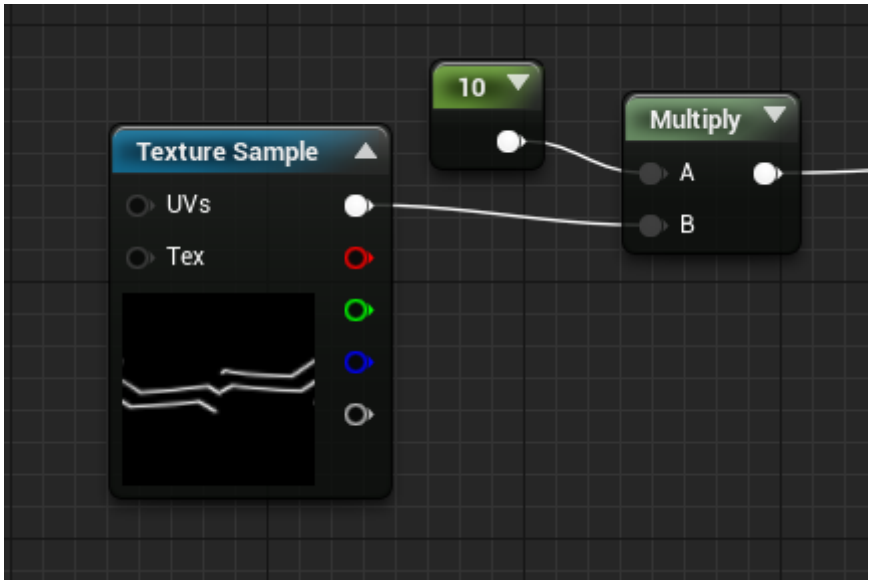


Slika 4.33. Odabir dodatnih tekstura i sveukupne rezolucije tekstura.

Dodatne teksture (*Ambient occlusion* i *emissive*) nisu automatski povezane s odgovarajućim mjestom u glavnom čvorištu, pa ih je potrebno povezati ručno. Kod *emissive* teksture, potrebno je dodati još dva dodatna čvorišta kako bi bilo moguće kontrolirati jačinu svjetlosti koju *emissive* tekstura opisuje. Potrebno je dodati čvorište koje vrši matematičku operaciju množenja (eng. multiply) i čvorište konstantne vrijednosti. Čvorište *multiply* prima dva ulazna argumenta, a i b. Kao ulazni argument (b) postavio sam izlazne podatke iz bitmap čvorišta koje sadrži *emissive* teksturu, dok sam kao *a* argument postavio konstantnu vrijednost 10. Time sam dobio veći sjaj na 3D modelu. Prikaz izrađenog materijala za 3D model lijevih vrata prikazan na slici 4.34. Prikaz ostvarenog načina kontrole jačine svjetlosti *emissive* teksture prikazan na slici 4.35. Uvezen 3D model sa svim teksturama prikazan je na slici 4.36.



Slika 4.34. Izrađen materijal u material editoru.



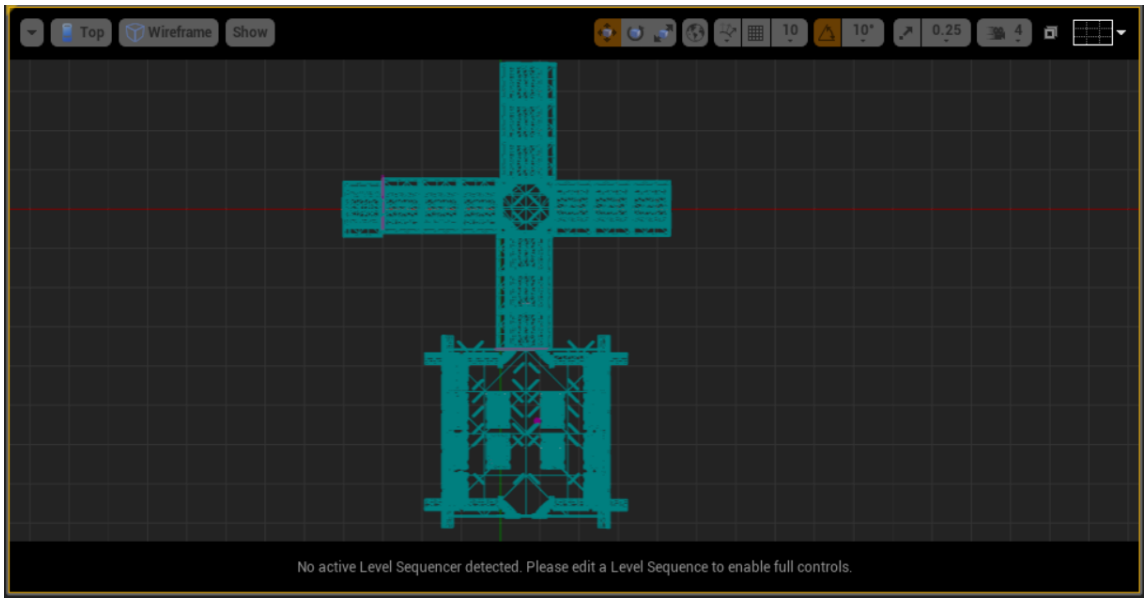
Slika 4.35. Kontrola jačine svjetlosti emmisive teksture.

Slika 4.36. 3D model sa svim teksturama.

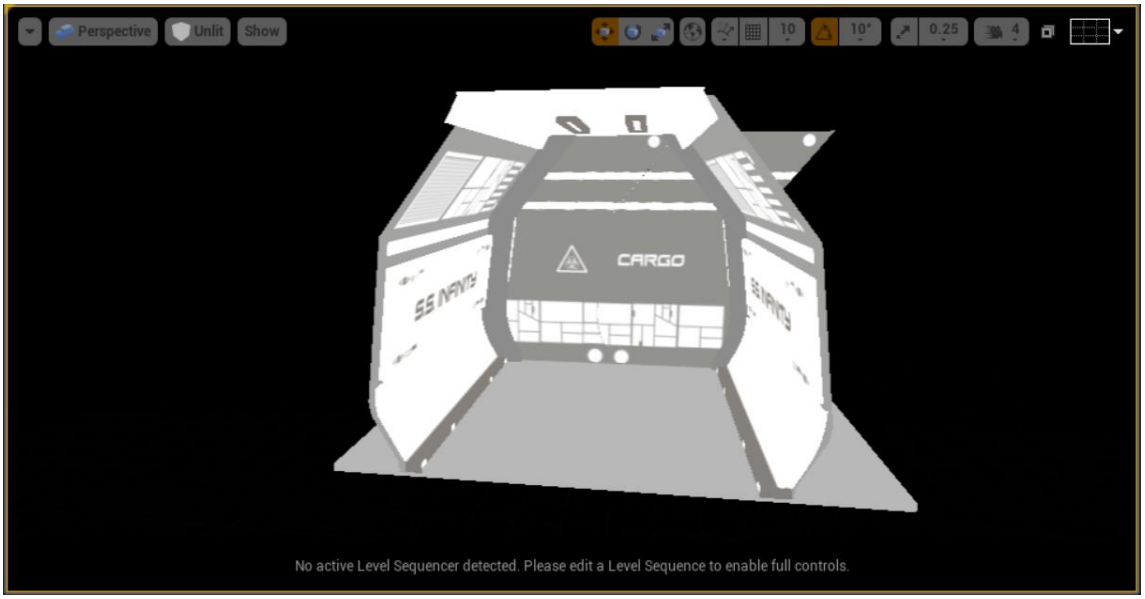
4.4.5. Izrada koridora i skladišta u Unreal Enigine 4

Unreal Engine 4 editor izvrstan je programski alat za izradu 3D okruženja. Njegove mogućnosti izlaze izvan opsega ovog rada pa ću opisivati samo one korištene za izradu 3D okruženja S.S. Infinity. Jednom kada su 3D modeli s pripadajućim teksturama bili uvezeni i za njih izrađeni materijali, mogao sam započeti s izradom 3D okruženja koridora i skladišta. Koristio sam se alatima za pomicanje i rotaciju 3D modela kako bih ih smjestio na odgovarajuća mjesta. Koridor sam izradio tako što sam prvo izradio jedan dio koristeći se pripadajućim 3D modelima namijenjenim za izradu koridora. Na početak koridora postavio sam 3D modele vrata koja ću kasnije animirati, dok sam na kraj koridora stavio 3D modele koji tvore kutove. Za smještaj 3D modela na željena mjesta ključno je što sam prilikom njihove izrade uskladio mjerne jedinice i postavke mreže u Maya 2016. Zbog istih mjernih jedinica i postavki mreže između Unreal Engine 4 i Maya 2016, omogućeno je nalijeganje 3D modela jedan do drugoga, bez praznog prostora. Prazan prostor između 3D modela nije poželjan zbog postavljanja osvijetljena. Tako izrađen koridor dupliciran je i rotiran za 90 stupnjeva tri puta. Izrada 3D okruženja na ovaj način tipičan je za modularni level dizajn. Skladište je izrađeno od istih 3D modela koji tvore koridor uz par dodatnih koji tvore police na kojima su smješteni 3D modeli rekvizita raznih spremnika.

Slika 4.37. Koridor S.S. Infinity, s osnovnim osvjetljenjem.



Slika 4.38. Žičani prikaz gornjeg prikaza S.S. Inifinity 3D okruženja.



Slika 4.39. Neosvijetljen modul koridora.

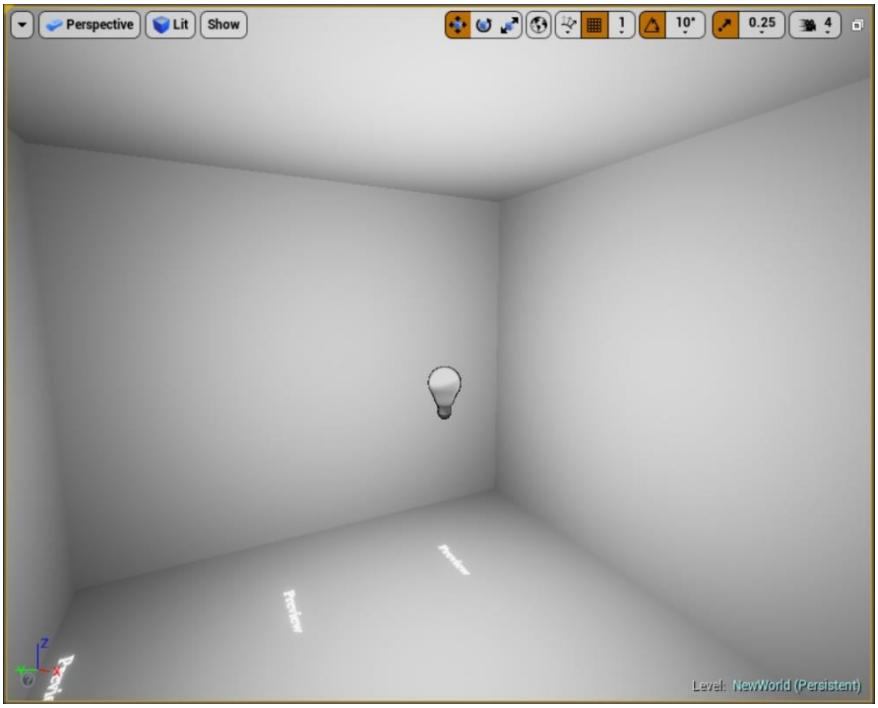
Nakon što su svi 3D modeli posloženi tako da tvore željeno 3D okruženje, potrebno je bilo dodati osvjetljenje. Velik dio osvjetljenja čine *emissive* teksture na 3D modelima stupova, lampi na zidovima i vratima. Iako su *emissive* teksture zamišljene da budu glavno osvjetljenje, zbog nedostatka vremena za izradu spremljenog osvjetljenja (eng. baked) koristio sam dinamično osvjetljenje i VXGI simulaciju indirektnog osvijetljena.

4.4.6. Osvjetljenje

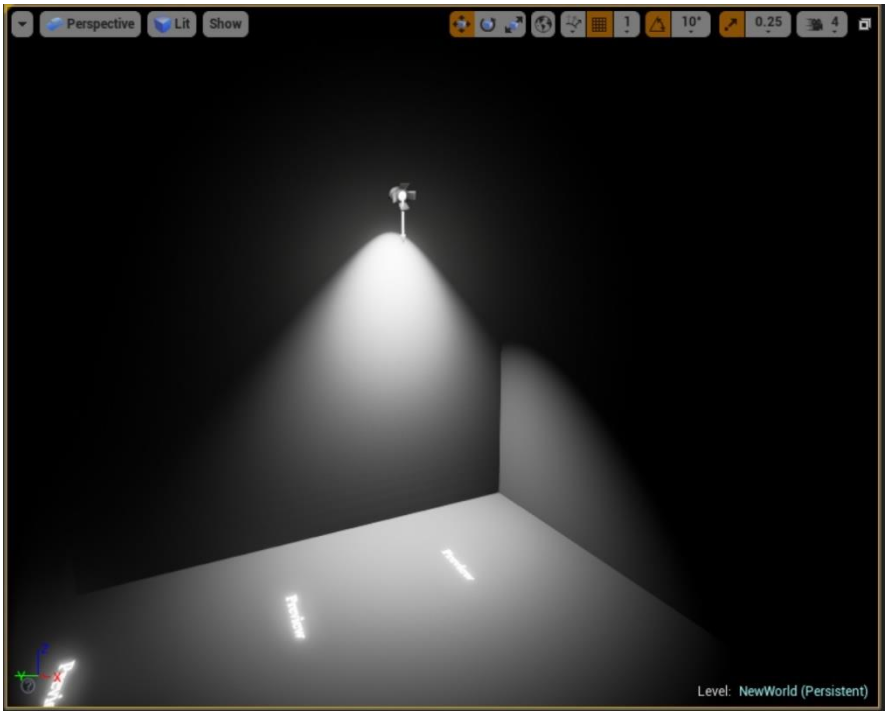
Osvjetljenje u Unreal Engine 4 je izvedeno tako da što više štedi računalne resurse, jer je izuzetno zahtjevno za računala. Uobičajena je praksa izraditi takozvane mape osvjetljenja koje se dodjeljuju 3D modelima na sličan način kao teksture. Mape osvjetljenja sadrže informacije o svjetlu te se tako izbjegava bespotrebno računanje svijetla u svakoj slici koju računalo prikazuje na zaslonu. Iako je uobičajeno izraditi mape osvjetljenja, postoje uvjeti kada je potrebno dinamično osvjetljenje, tj. ono koje se izvodi u stvarnom vremenu.

U Unreal Engine 4 moguće je koristiti četiri tipa osvjetljenja; direkciono (eng. direction light), sferno (eng. point light), usmjereno (eng. spot light) i nebo (eng. skylight). Pod tip osvjetljenja spadaju još i *emissive* teksture, ali one kod klasične izrade mape osvjetljenja ne doprinose puno. *Direction light* je tip osvjetljenja koji je najsličniji suncu i nužan je kod 3D okruženja koji su eksterijeri. *Point light* je najsličniji zidnoj lampi, koji ima konusan prostor osvjetljenja. *Spot light* je tip osvjetljenja koji ima radijalan prostor osvjetljenja i idealan je za rasvjetu zatvorenih 3D okruženja. *Point* i *spot light* tipovi osvjetljenja mogu biti statični (eng. static), stacionarni (eng. stationary) ili dinamični (eng. movable). Statični tip osvjetljenja uvijek zauzima najmanje računalnih resursa i svi podaci o osvjetljenju pospremaju se u mape osvjetljenja. Stacionarno osvjetljenje nema tu mogućnost, jer je predviđeno za promjene za vrijeme izvođenja igre. Dinamično osvjetljenje zauzima najviše računalnih resursa prilikom izvođenja igre. Unreal Engine 4 implementira sustav za izradu mapa osvjetljenja naziva *eng. Lightmass*. Taj sustav za izradu mapa osvjetljenja dopušta korištenje samo četiri ista tipa osvjetljenja prilikom izrade mape osvjetljenja. To na prvi pogled izgleda jako malo, ali lako je za shvatiti kako jednom kada se mape osvjetljenja stvore, nestaje potreba za izvorom osvjetljenja. Dinamičko osvjetljenje nema mogućnost pospremanja informacija o osvjetljenju u mape osvjetljenja, pa stoga zauzima najviše računalnih resursa. [15]

Za izradu osvjetljenja 3D okruženja S.S. Infinity koristio sam dinamično osvjetljenje. Dinamično osvjetljenje koje sam koristio nije ono implementirano u Unreal Engine 4. Koristio sam način simulacije indirektnog osvjetljenja naziva *VXGI* tvrtke *Nvidia*. Korištenje tog načina osvjetljenja značajno mi je ubrzalo rad na izradi osvjetljenja za projekt S.S. Infinity. *VXGI* je tako značajan dio tijeka rada izrade 3D okruženja.



Slika 4.40. Point light.



Slika 4.41. Spot light.

Na slikama 41 i 42 prikazano je jednostavno 3D okruženje s *point light* i *spot light* tipovima osvjetljenja. Prije izrade mape osvjetljenja, na teksturama su ispisani natpisi *preview*. Natpis *preview* označava da trenutni prikaz osvjetljenja privremen i da je potrebno izraditi mapu osvjetljenja.

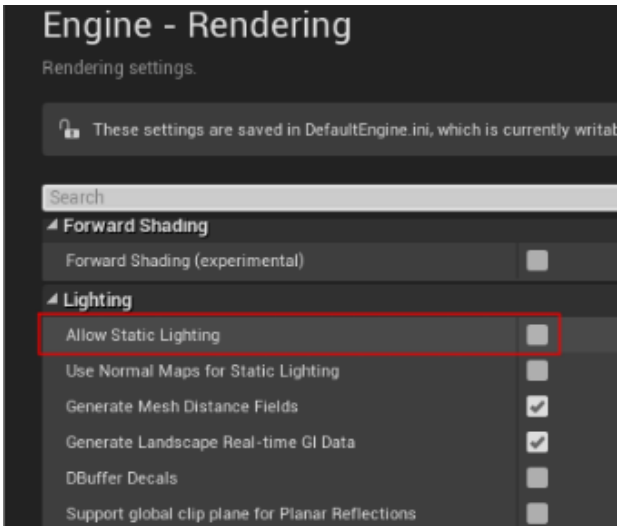
4.4.6.1. NVIDIA VXGI

Slika 4.42. Nvidia VXGI logo (izrađen s Adobe Photoshop CC).

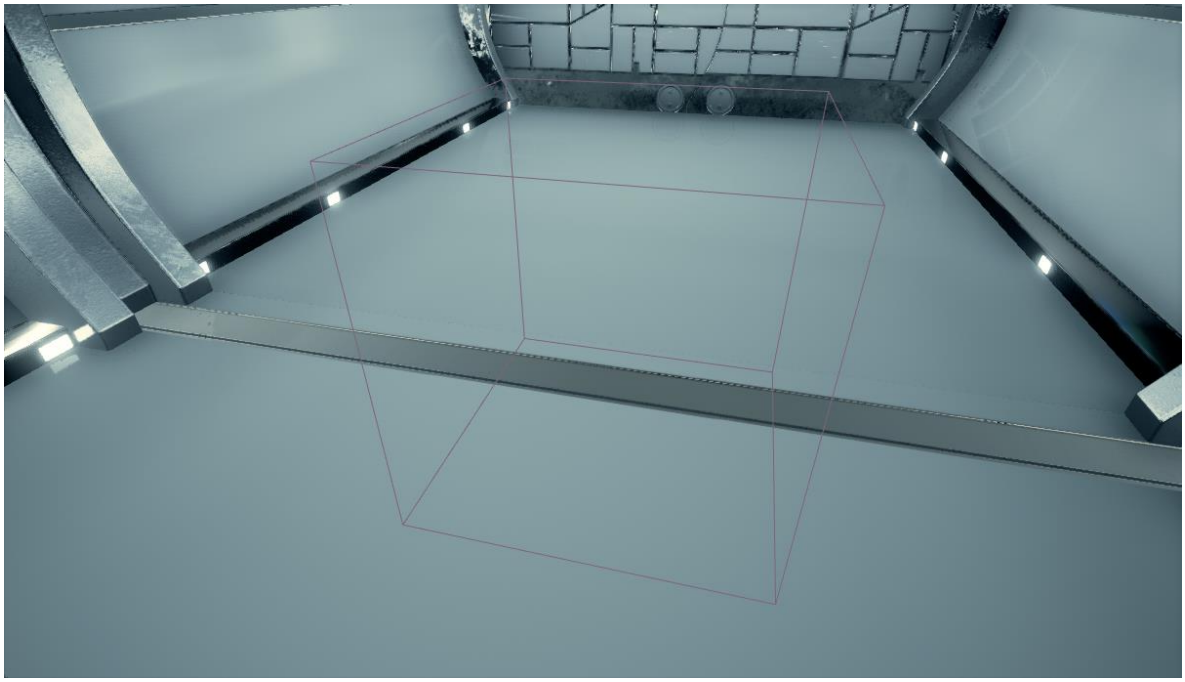
VXGI je grafički podsustav koji razvija tvrtka Nvidia. Tvrtka Nvidia je poznata po izradi grafičkih kartica za računala. Također, Nvidia konstantno ulaže u razvoj novih tehnologija u računalnim igrama. VXGI je slobodno dostupna, revolucionarna tehnologija osvjetljenja 3D okruženja računalnih igara, koja koristi dinamično osvjetljenje uz bitno manje zahtjeve od klasičnog dinamičnog osvjetljenja implementiranog u Unreal Engine 4. VXGI standardno ne dolazi implementiran s Unreal Engine 4, pa je potrebno s Nvidia *GIT* repozitorija preuzeti inačicu Unreal Engine 4 s već implementiranim VXGI. [16]

VXGI simulira indirektno osvjetljenje koristeći *voxel* tehnologiju prikaza. *Voxel* se najjednostavnije može definirati kao 3D pixel. Indirektno osvjetljenje podrazumijeva odbijenu svjetlost od svih površina 3D okruženja i normalna je pojava u stvarnom svijetu. Izračun odbijene svjetlosti izrazito je zahtjevan za računala pa se stoga inače ne implementira u pokretače računalnih igara. VXGI uspješno simulira indirektno osvjetljenje tako da postavlja niz *voxel* stožaca kroz cijelo 3D okruženje i dodaje osvjetljenje tamo gdje bi se inače pojavilo u slučaju indirektnog osvjetljenja. VXGI je dalje znatno zahtjevan za računala, ali pruža veliku fleksibilnost postavki, pa je vrlo lako podesiti zadovoljavajuću kvalitetu osvjetljenja uz prihvatljive računalne zahtjeve. Dodavanje globalne iluminacije u 3D okruženje računalne igre znatno povećava realizam jer prikazuje svu svjetlost u 3D okruženju, pa čak i onu odbijenu od sjajnih površina. [16]

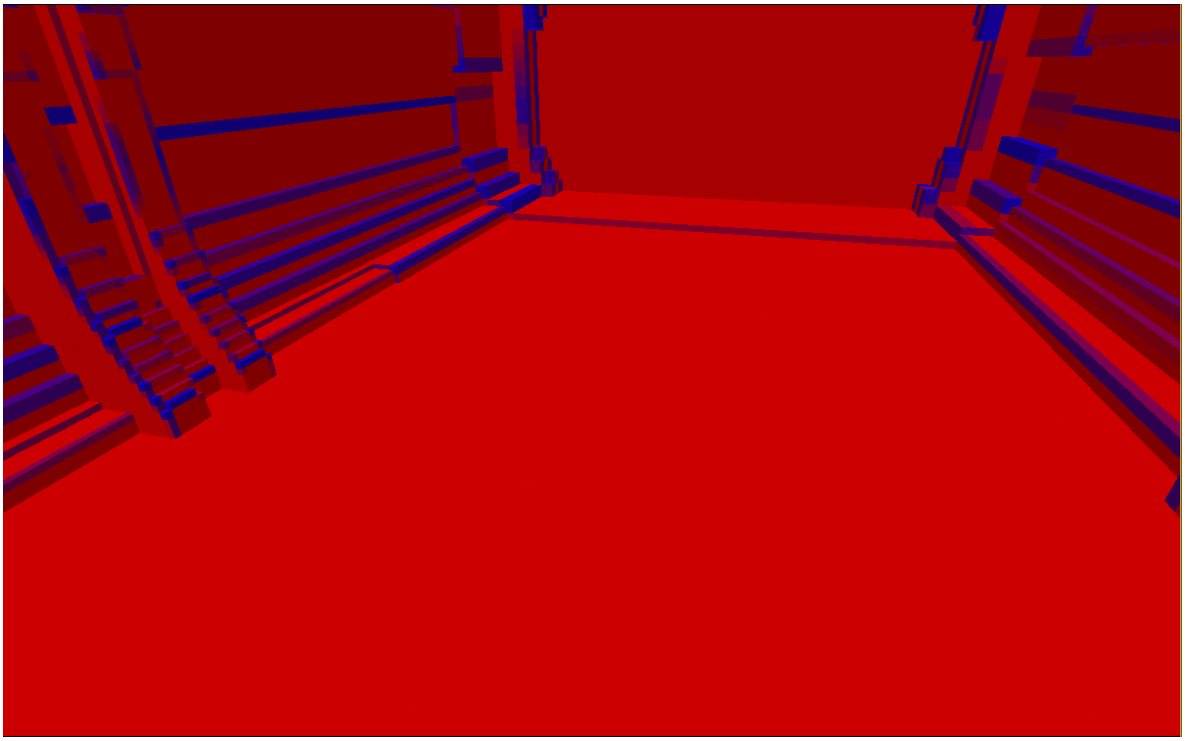
Jednom kada se pokrene inačica Unreal Engine 4 za implementiranim VXGI sustavom, potrebno je podesiti pojedine postavke grafičkog sustava Unreal Engine 4. Potrebno je pristupiti postavkama osvjetljenja (eng. lighting) kojima se pristupa padajućim izbornikom na *edit* > *project settings* > *lightning*. U *lightning* postavkama potrebno je odznačiti opciju *allow static lighting* kako bi se u potpunosti mogla iskoristiti VXGI tehnologija. Postupak onemogućavanja statičnog osvjetljenja prikazan je na slici 4.43. Sljedeće što je potrebno napraviti je postaviti u 3D okruženje objekt koji služi za upravljanje post proces efektima i omogućiti mu opciju *unbound*, kako bi mogao djelovati na cijelo 3D okruženje. Post proces objekt se nalazi u *modes* izborniku pod odjeljkom *volumes*. Nakon što je u 3D okruženju prisutan *post proces volume* objekt, potrebno je selektirati ga i zatim u *details* izborniku pod odjeljkom *VXGI diffuse* omogućiti opciju *enable diffuse*. Tom opcijom uključit će se VXGI tehnologija. Nakon što je VXGI omogućen, potrebno je podesiti broj obijanja simuliranih zraka svjetlosti pod *multi-bounce* i broj *voxel* stožaca. Za potrebe osvjetljenja projekta S.S. Infinity, broj odbijanja svjetlosti postavljen je na 1, dok je broj *voxel* stožaca postavljen na 16. Ove dvije postavke znatno utječu na računalne zahtjeve, pa je potrebno pronaći idealnu kombinaciju vrijednosti kako bi se postigla kvaliteta prikaza uz zadovoljavajuće računalne zahtjeve. Nakon što su te dvije postavke postavljene, potrebno je dodatno optimizirati VXGI osvjetljenje. Postavka *cone rotation* mora biti uključena i *tracing sparsity* bi trebao biti postavljen oko vrijednosti 3. Te dvije postavke ograničavaju djelovanje VXGI tehnologije u smjeru kamere. Također je potrebno omogućiti opciju VXGI *specular*, jer se time omogućuje računanje odbijenog svjetla od sjajnih površina.



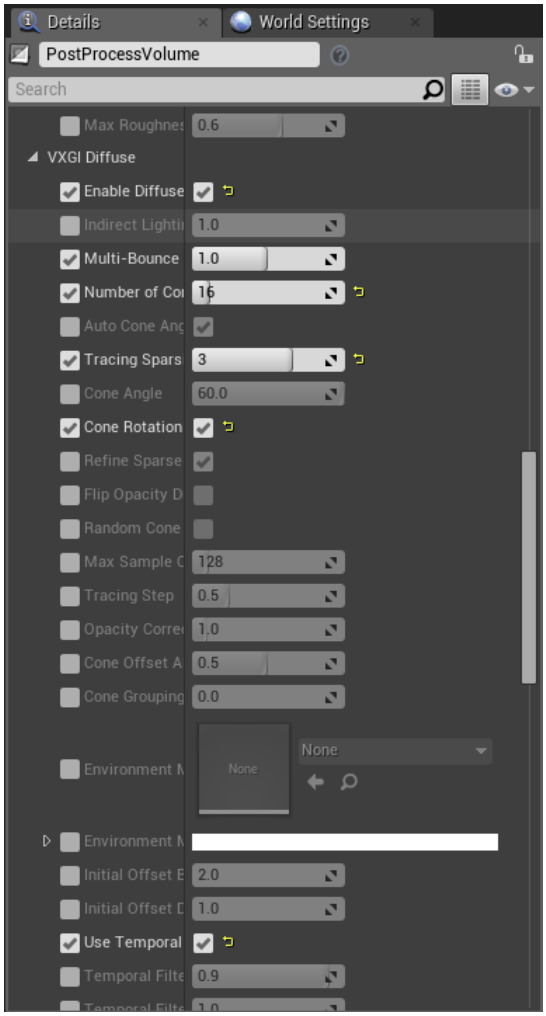
Slika 4.43. Onemogućavanje statičnog osvjetljenja.



Slika 4.44. Post process volume objekt.



Slika 4.45. Voxelizirano 3D okruženje.



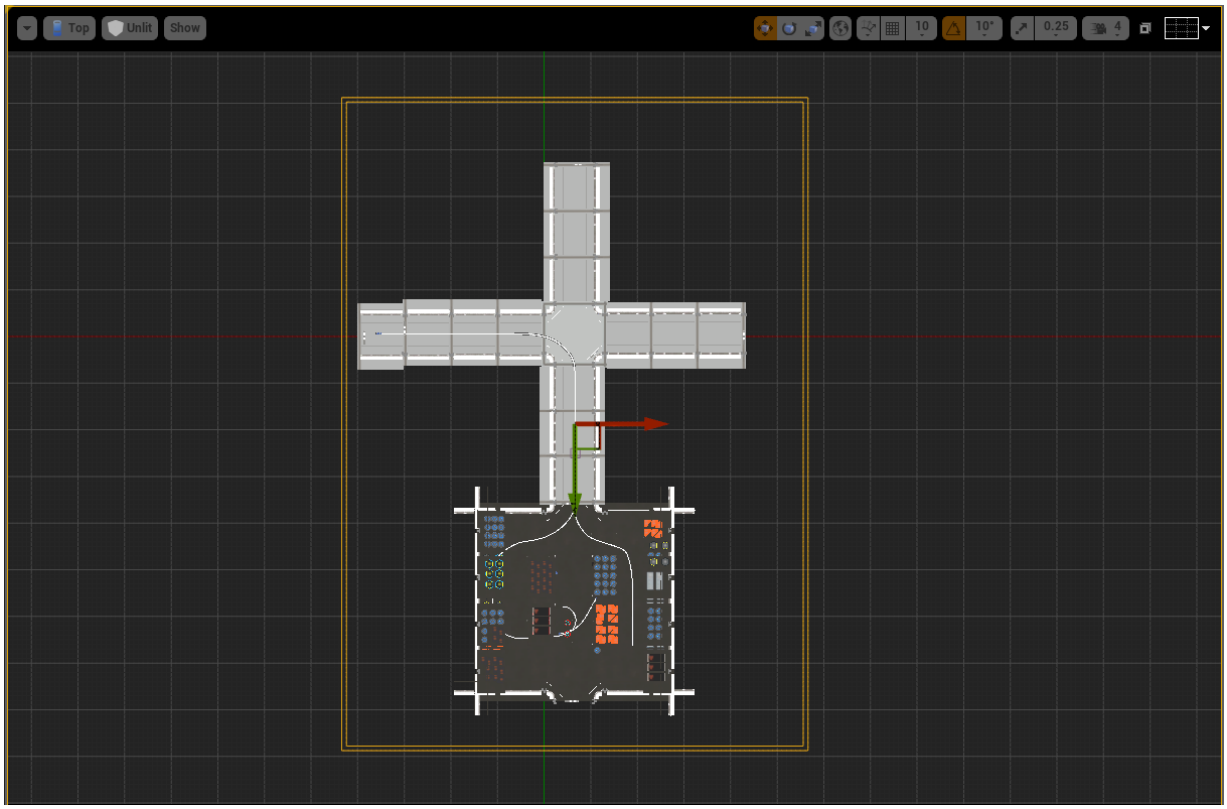
Slika 4.46. Neke od važnijih postavki VXGI tehnologije.

Postavljanje navedenih postavki VXGI tehnologije osvjetljenja individualno je i ovisi o vrsti 3D okruženja. Za potrebe projekta S.S. Infinity podešene su idealne postavke kako bi se maksimalno iskoristila kvaliteta prikaza koju omogućuje VXGI. Iako, podešene postavke su namijenjene izvozu png datoteka za potrebe animacije i nikako ne bi bile zadovoljavajuće u slučaju potreba izrade računalne igre.

4.4.6.2. Refleksije i post proces efekti

Refleksije su odmah iza osvjetljenja po kompleksnosti računanja. Zahtijevaju mnogo računalne moći kako bi se realno prikazivale. Stoga je potrebno ograničiti ih unutar određenih granica. Unreal Engine 4 radi to na domišljat i efikasan način implementacijom zonama hvatanja refleksija *(eng. reflection capture*). Tako postoje 3D objekti sferne i poligonalne zone hvatanja refleksija (*eng. sfere, box reflection capture*). Ti 3D objekti mogu se kombinirati zajedno i tako stvoriti realne refleksije. Svakako je preporučljivo ne koristiti ih previše, zbog zahtjevnosti računanja pri izvođenju igre.

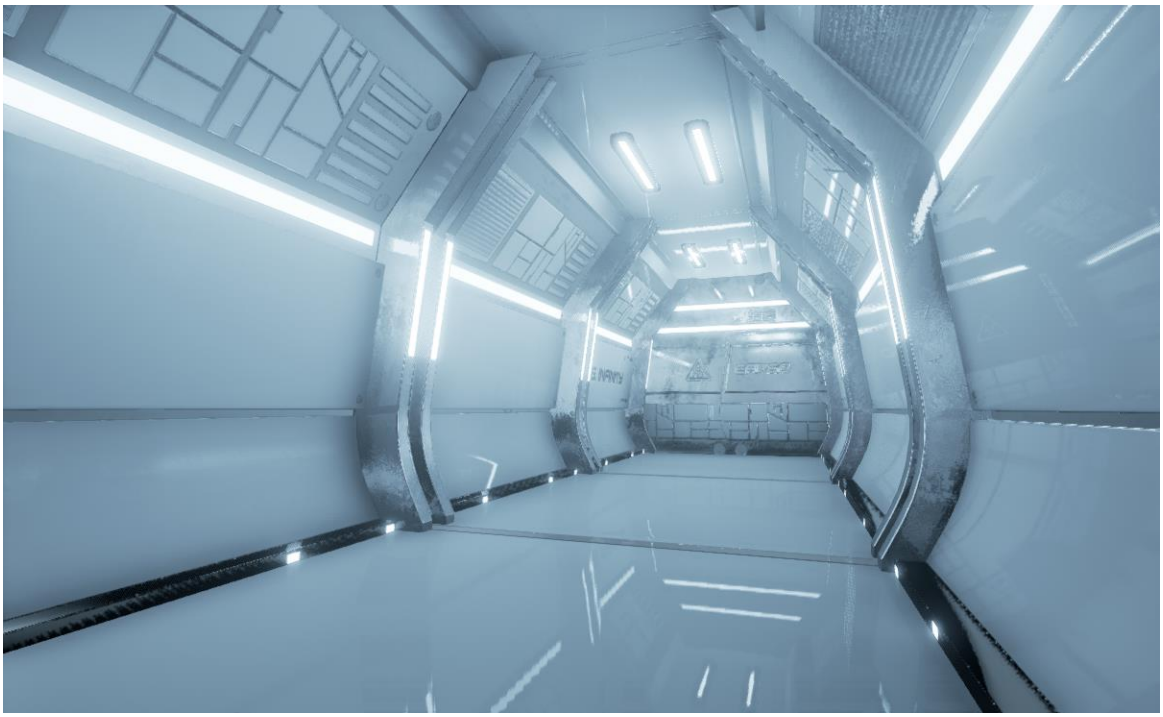
Za potrebe 3D okruženja S.S. Infinity korištena je samo poligonalna zona za hvatanje refleksija. Poligonalna zona je postavljena tako da obuhvaća cijelo 3D okruženje. Na isti način je mogla biti postavljena i sferna zona hvatanja refleksija, ali zbog korištenja VXGI tehnologije, praktički je svejedno koja zona hvatanja refleksija će se koristiti.



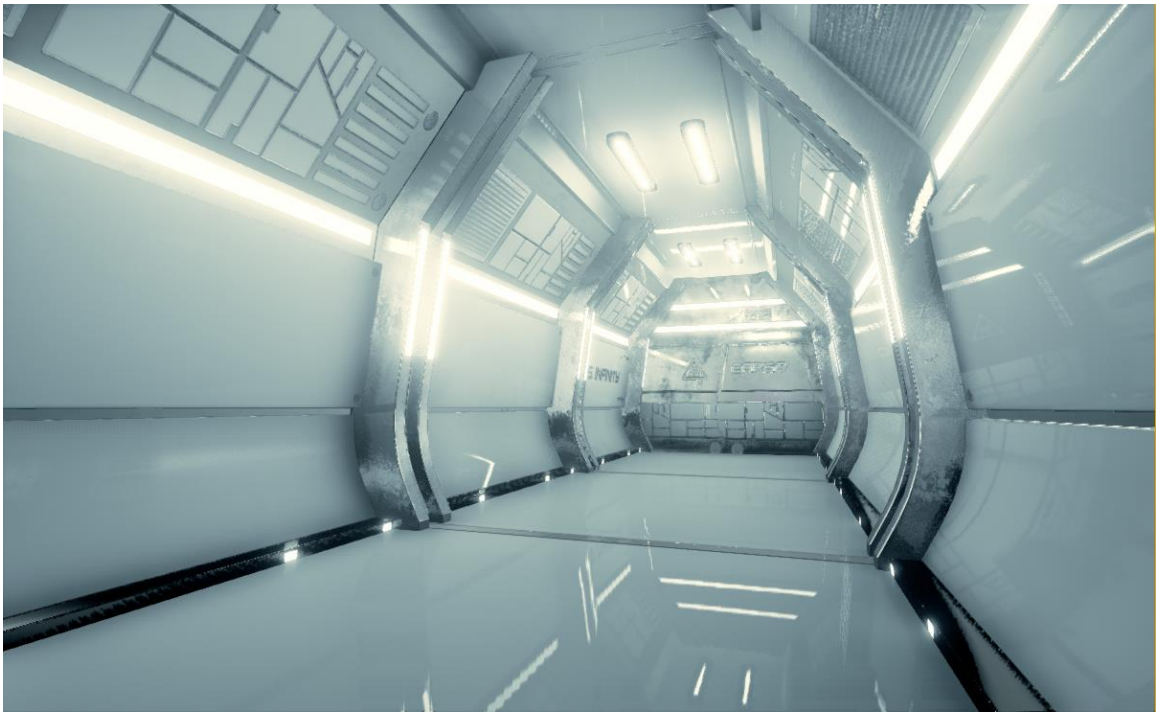
Slika 4.47. Poligonalna zona hvatanja refleksija.

Zone hvatanja refleksija nalaze se u *modes* odjeljku sučelja Unreal Engine 4 u pod izborniku *volumes*. Nakon što je postavljena zona refleksije, i u 3D okruženju postoje vizualno prihvatljive refleksije, potrebno je postaviti određene post proces efekte. Post proces efekti korišteni za 3D okruženje S.S. Infinity su gradacija boja, automatska ekspozicija, polje dubinske oštrine (kod animiranih kamera) i morfološko zaobljenje rubova (eng. anti-aliasing).

Gradacija boja omogućena je u postavkama *post process volume* objekta pod odjeljkom *scene color*. Gradaciju boja omogućuje *LUT* datoteka stvorena u Adobe Photoshop CC. Dovoljno je uzeti jedan snimak zaslona 3D okruženja u Unreal Engine 4, i u Adobe Photoshop CC obraditi sliku te ju izvesti kao *LUT* datoteku. *LUT* datoteku potrebno je učitati u *post proces volume* postavkama i cijeli prikaz 3D okruženja će biti u skladu s obrađenom snimkom zaslona. Gradacijom boja postigao se filmski izgled 3D okruženja.



Slika 4.48. Prikaz 3D okruženja bez gradacije boja.



Slika 4.49. Prikaz 3D okruženja s gradacijom boja.

Postavka automatske ekspozicije korisna je zbog ujednačene ekspozicije tijekom preleta kamere za vrijeme animacije. Postavke automatske ekspozicije nalaze se pod izbornikom *Auto exposure* te imaju dvije vrijednosti, minimalnu i maksimalnu ekspoziciju. Obije vrijednosti potrebno je postaviti na 1 kako bi se postigla ujednačena ekspozicija tijekom preleta kamere kroz 3D okruženje. Polje dubinske oštrine također je potrebno uključiti kako bi se kasnije na kamerama moglo iskoristiti plitko polje dubinske oštrine. Omogućavanje polja dubinske oštrine omogućeno je pod izbornikom *depth of field*. Morfološko zaobljenje rubova omogućeno je pod izbornikom *misc > AA method*. Za potrebe projekta S.S. Ifinity, korištena je FXAA metoda zaobljenja rubova s faktorom 4. To u osnovi znači da će se podešena rezolucija prikaza dodatno multiplicirati četiri puta, čime će se postići zaglađeni rubovi u prikazu 3D okruženja.

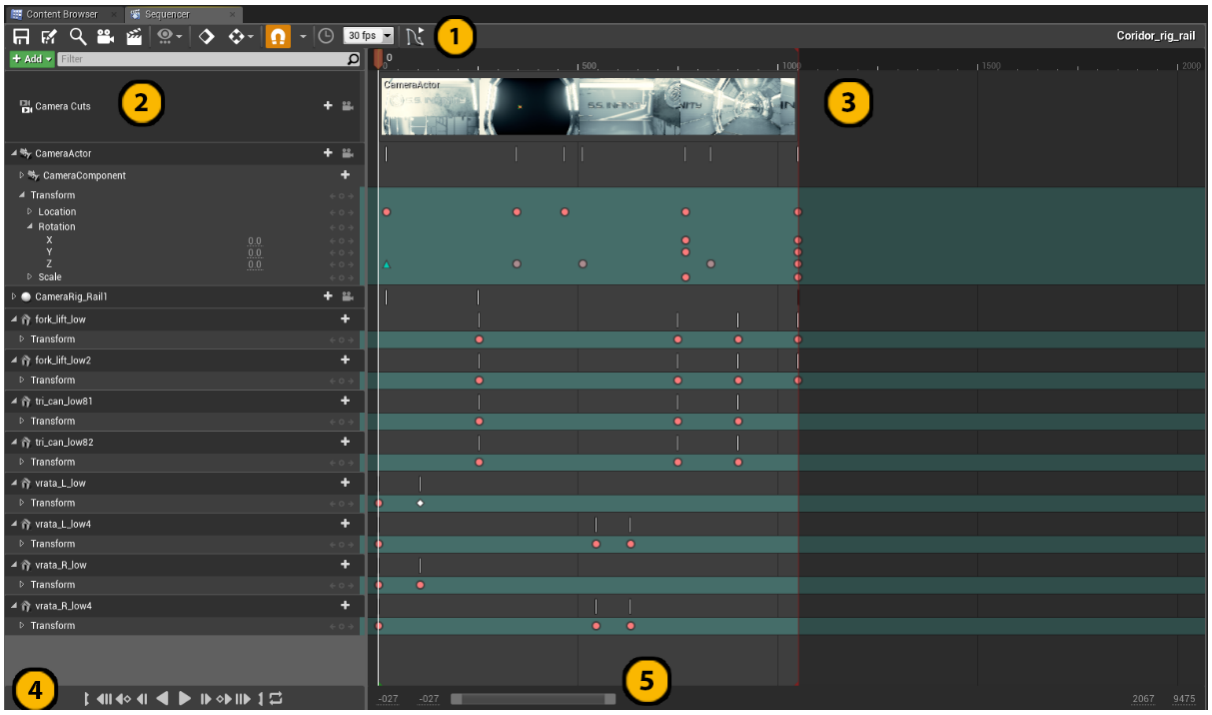
Postavljanjem osvjetljenja završena je izrada 3D okruženja projekta S.S. Infinity. Izrađeno 3D okruženje koje sadrži koridor i skladište svemirskog broda S.S. Infinity sa grafičke strane u potpunosti je spremno za daljnji razvoj računalne igre. 3D okruženje je kasnije iskorišteno za izradu kratke animacije u kojoj se prikazuje radni stadij razvoja igre S.S. Infinity. Za potrebe računalne igre, bilo bi potrebno uložiti još mnogo truda za implementiranje interaktivnih elemenata, umjetne inteligencije, sustava upravljanja igrom i mnogo drugih sustava koji bi zajedno tvorili računalnu igru S.S. Infinity.

Slika 4.50. Snimak zaslona skladišta u S.S. Infinity 3D okruženju.

5. izrada animacije

Izrada animacija unutar Unreal Engine 4 programskog paketa omogućena potprogramom *Sequencer*. *Sequencer* je moćan dio Unreal Engine 4 u kojem je moguće izrađivati animacije gotovo jednake kvalitete kao i u namjenskim programskim paketima za izradu istih. *Sequencer* upravlja svim animacijama tako da izrađuje sekvence koje sadrže kompletne animirane elemente u 3D sučelju. Tako spremljene animacije moguće je montirati u video, slično kao i kod namjenskih programskih paketa za to. Također, omogućen je izvoz svih slika (eng. frame) potrebnih za izradu video datoteke u nekom od programskih paketa namijenjenih specifično za taj zadatak. Za izradu finalne animacije, korišten je Adobe Premiere CC programski paket. [17]

5.1. Sequencer



Slika 5.1. Unreal Engine 4 Sequencer.

Pregled sučelja Sequencer potprograma:

1. - Traka s alatima koja sadrži sve alate potrebne za manipulaciju potprogramom, spremanjem napretka, kontrole *keyframeova* i načina broja slika u sekundi. Traka alata s alatima također sadrži ikonu za pristup *curve editoru* unutar kojeg je moguće manipulirati interpolacijom između dva ili više *keyframea*.

2. - Prostor za prikaz svih elementa sekvence. Tu se prikazuju kadrovi i svi animirani elementi u njima.

3. - Grafički prikaz kadra i svih *keyframeova* u njemu.

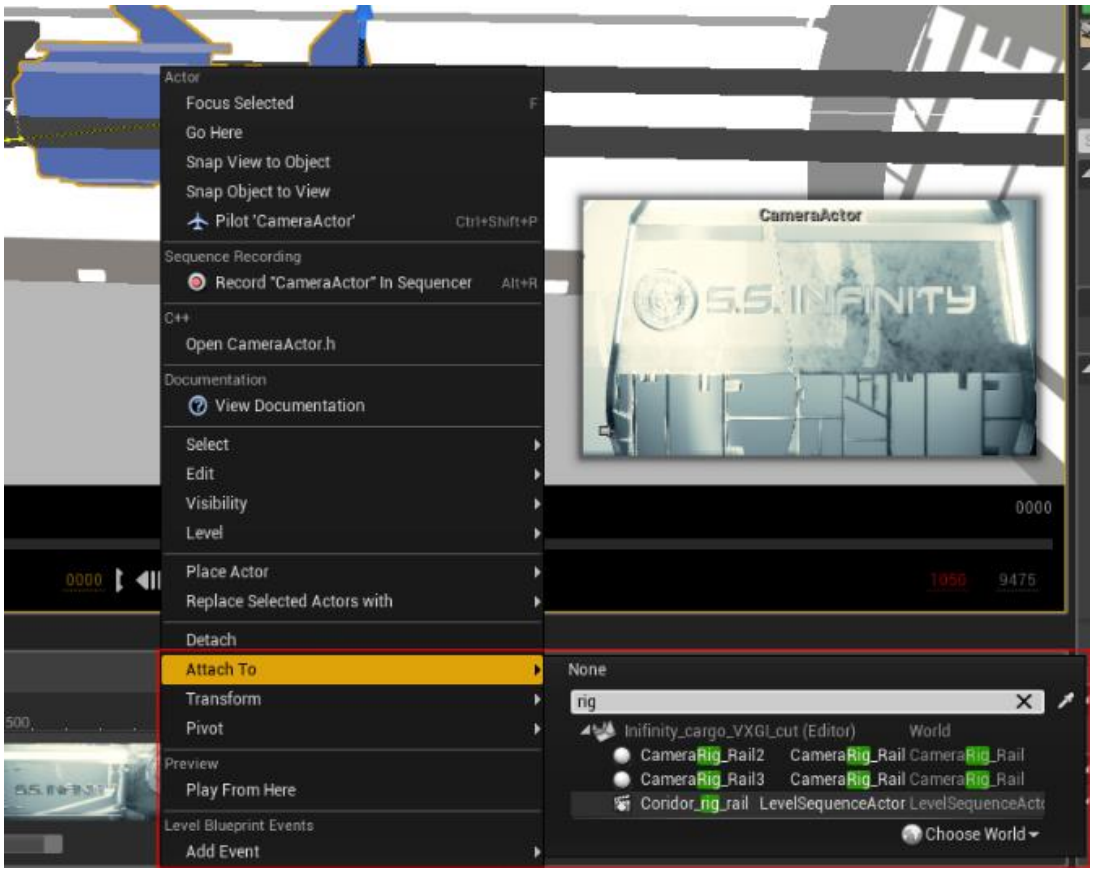
4. - Reprodukcijska traka koja omogućuje puštanje, zaustavljanje, pauziranje, postavljanje na kraj i početak animacije. Kontrolna traka također omogućuje premotavanje po *keyframeovima*.

5. - *Range slider* kojim je moguće postavljati raspon animacije kadra brojem slika. Također, *range slider* omogućuje pomicanje kroz prostor za prikaz svih elemenata sekvence.

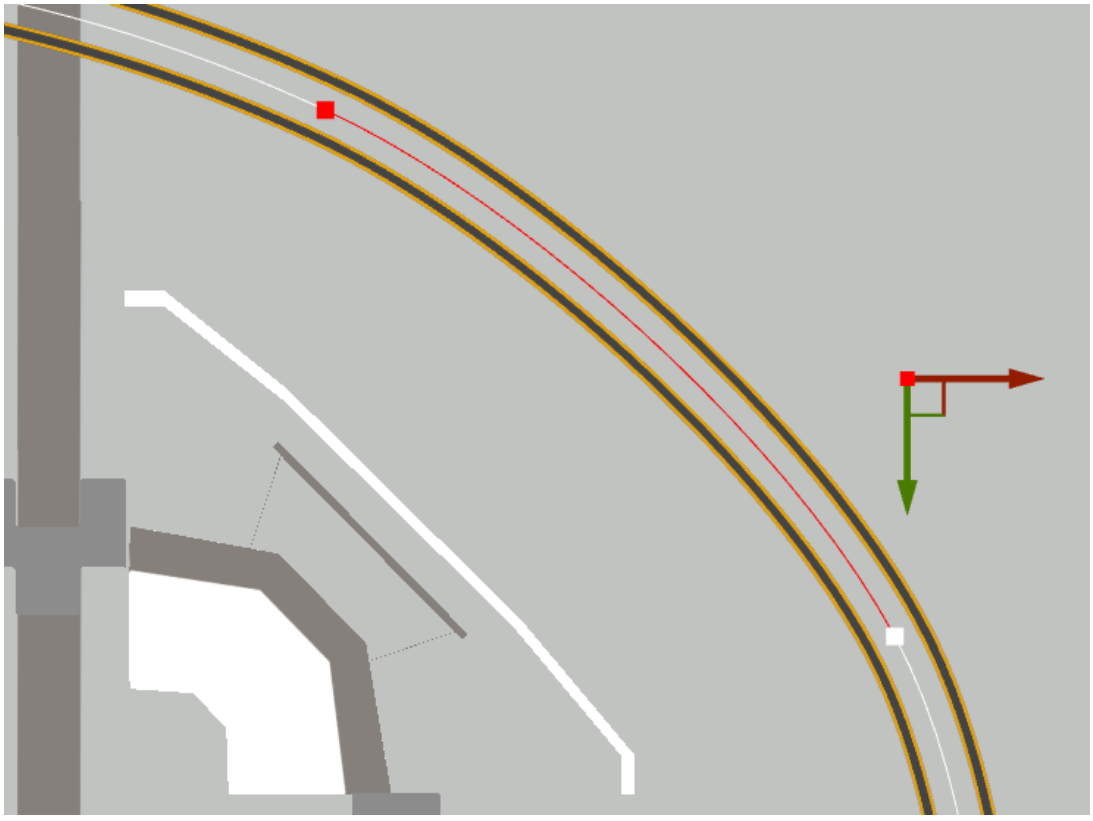
5.2 Animacija kamere i 3D modela

Prvo što je potrebno kod izrade animacija u Unreal Engine 4 *Sequenceru* je stvoriti novu sekvencu. Stvaranje nove sekvence omogućeno je u alatnoj traci sučelja Unreal Engine 4 pod *Cinematics > Add new level sequence.* Nakon što postoji sekvenca, potrebno je dodati traku s kadrovima. Dodavanje nove trake kadrova omogućeno je u padajućem izborniku *sequencera* na gumbu *Add > camera cut track*

Animacija kamere omogućena je *rig rail* objektom koji se postavlja u 3D okruženje. Taj objekt se ponaša identično kao i tračnice za kameru u stvarnom svijetu. Dostupan je na *modes > cinematics*. Kada se *rig rail* uvede u 3D okruženje i postavi na željeno mjesto, potrebno mu je pridružiti kameru. Kameru je također moguće pronaći u *cinematics* izborniku. Potrebno ju je povezati s *rig rail* objektom tako da se pristupi padajućem izborniku desnim klikom miša i odabere a*ttach > CameraRig_Rail*. Nakon što su ta dva objekta povezana, potrebno je izraditi putanju kamere. *Rig rail* objekt je u osnovi vektorska krivulja po kojoj kamera putuje. On posjeduje elemente vektorske krivulje, čvorišta i manipulacijske kontrole. Nakon što je zadana putanja, *rig rail* objektu je potrebno zadati krajnju vrijednost. On može imati vrijednosti od 0 do 1, pa mu je tako vrijednost 0 automatski pridružena na prvu sliku, dok mu vrijednost 1 treba postaviti na zadnjoj slici. To se radi tako što se u *sequence editoru* odabire taj objekt, i u odjeljku *details* pod *rail controls* postavi vrijednost 1 u polje *current position*. Izradom krivulje *rig rail* objekta i postavljanjem vrijednosti njegove pozicije 1 na zadnjoj slici u kadru, izrađena je animacija kamere. Isti postupak je ponovljen za ostala 3 kadra, koji se odvijaju u skladištu.



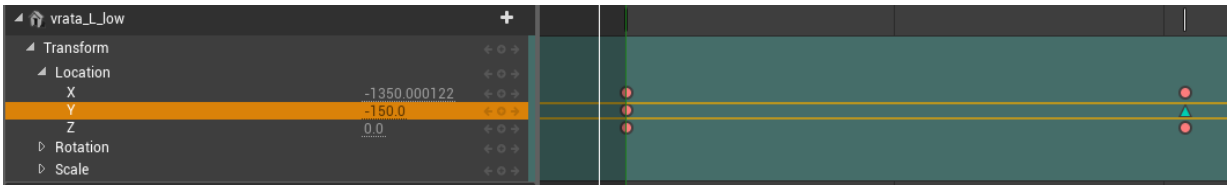
Slika 5.2. Povezivanje kamere s rig rail objektom.



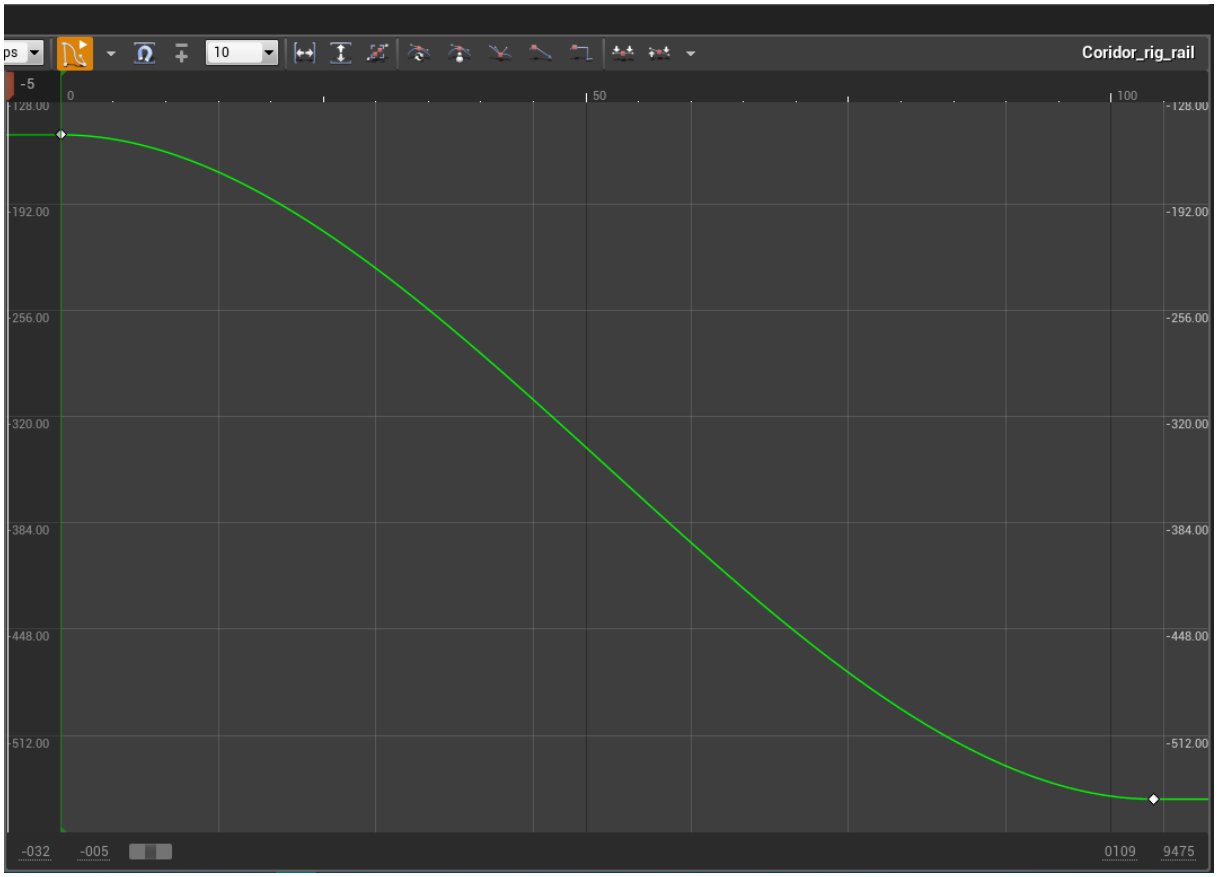
Slika 5.3. Kontrolna čvorišta rig rail objekta.

Animacija 3D modela vrata i lebdećih robota je izrađena u potpunosti u *sequenceru.* Iako, uobičajeno je da se takve animacije izrade u nekom od programskih paketa za animaciju, pa onda zajedno s 3D modelima izvezu i uvezu u pokretač igre. Izrada animacija 3D modela izrađena je u *sequanceru* zbog jednostavnosti samih animacija. Također, rad u *sequenceru* dodatno je ubrzao rad na projektu, jer nije bilo potrebe za radom u više programskih paketa.

Za animaciju 3D modela korištena je tehnika postavljanja *keyframeova*. Za primjer uzimam animaciju 3D modela lijevih i desnih vrata. Prvo što je potrebno kod izrade animacije nekog 3D modela je selektirati ga i dodati ga u *sequencer.* Nakon što je 3D model dodan, za njega je automatski postavljena nova traka na koju je moguće postavljati *keyframe.* Moguće je omogućiti opciju automatskog postavljanja *keyframea. Sequencer* će tako automatski dodati *keyframe* nakon što se dogodi neka promjena. Loša strana automatskog dodavana keyframeova je što se dodaju nepotrebni *keyframeovi* za sve atribute. Primjerice, ako se izrađuje animacija pokreta po jednoj osi, bespotrebne su informacije o skaliranju 3D modela ili njegovoj rotaciji. Način izrade animacije otvaranja vrata prilikom prolaska kamere je bio sljedeći; prvo je bilo potrebno dodati početni *keyframe* na nultu sliku. Nakon toga, oba 3D modela vrata postavljena su u otvoreni položaj i dodan im je *keyframe* za translaciju po Y osi na 104. slici Zatim je u *curve editoru* bilo potrebno podesiti kubičnu interpolaciju na početni *keyframe,* te linearnu interpolaciju na zadnji frame. Podešavanjem interpolacije dobilo se na glatkoći animacije. Opisani postupak animacije 3D modela vrata korišten je u velikoj sličnosti kod svih animacija 3D modela za sve kadrove. Prikaz izrađenog kadra u *sequenceru* prikazan je na slici 5.1. pod poglavljem *Sequencer.*



Slika 5.4. Postavljanje keyframeova za 3D modele lijevih vrata.



Slika 5.5. Curve editor u sequenceru.

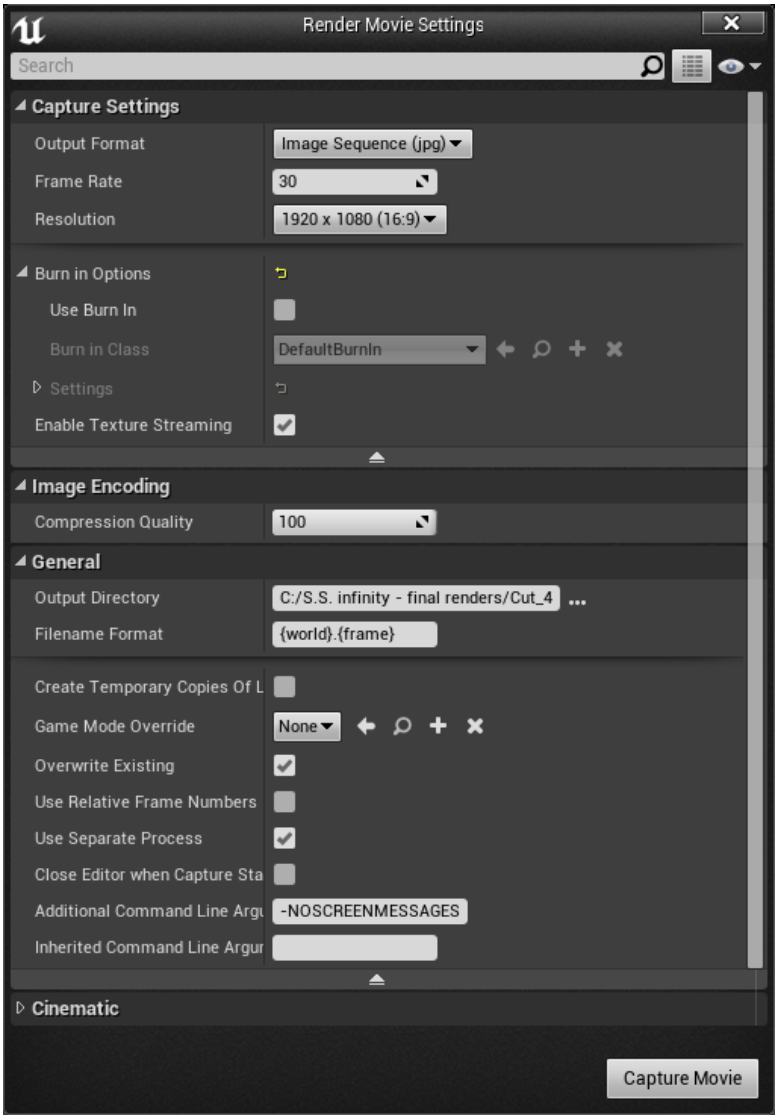
5.3. Izvoz animacije u png datoteke (eng. render)

Nakon što su svi kadrovi izrađeni, pospremljeni su u obliku level sekvenci i moguće ih je pronaći u *content browseru* pod odgovarajućim nazivima. Iako *sequence editor* implementira mogućnost izvoza animacija u avi kontejner, nedostaju mu naprednije kontrole kako nad *kontejnerom*, tako nad *kodekom*. Zbog nedostataka tih kontrola, veća kvaliteta izvezenih slika može se dobiti na način za da se izvezu u *png* datoteke bez kompresije. Izvezene slike je zatim potrebno spojiti i pakirati u neki kvalitetniji *kontejner* s kvalitetnijim *kodekom*.

Prije izvoza kadrova, potrebno je bilo podesiti određene postavke post procesiranja na kamerama svih kadrova. Potrebno je bilo podesiti morfološko zaobljenje rubova na FXAA i množenje istog na 4x. Također, potrebno je podesiti *bloom* efekt na 0 kako bi izvezene slike bile što oštrije. Još jedna jako važna post proces postavka je *VXGI specular > max sample count.* Ta postavka direktno utječe na kvalitetu prikaza refleksija pa je postavljena na 128.

Izvoz svih slika pojedinog kadra omogućen je klikom na ikonu za izvoz slika animacije u alatnoj traci *sequence editora*. U prozoru za izvoz slika potrebno je podesiti format datoteke, lokaciju izvoza, broj slika u sekundi, rezoluciju slika te razinu kompresije. Također je važno onemogućiti opciju *enable texture streaming*, kako bi se izbjegle zamućene slike prilikom izvoza. Opisani način izvoza kadra primijenjen je na sve ostale kadrove.

Slika 5.6. Ikona za izvoz slika kadra (eng. render).

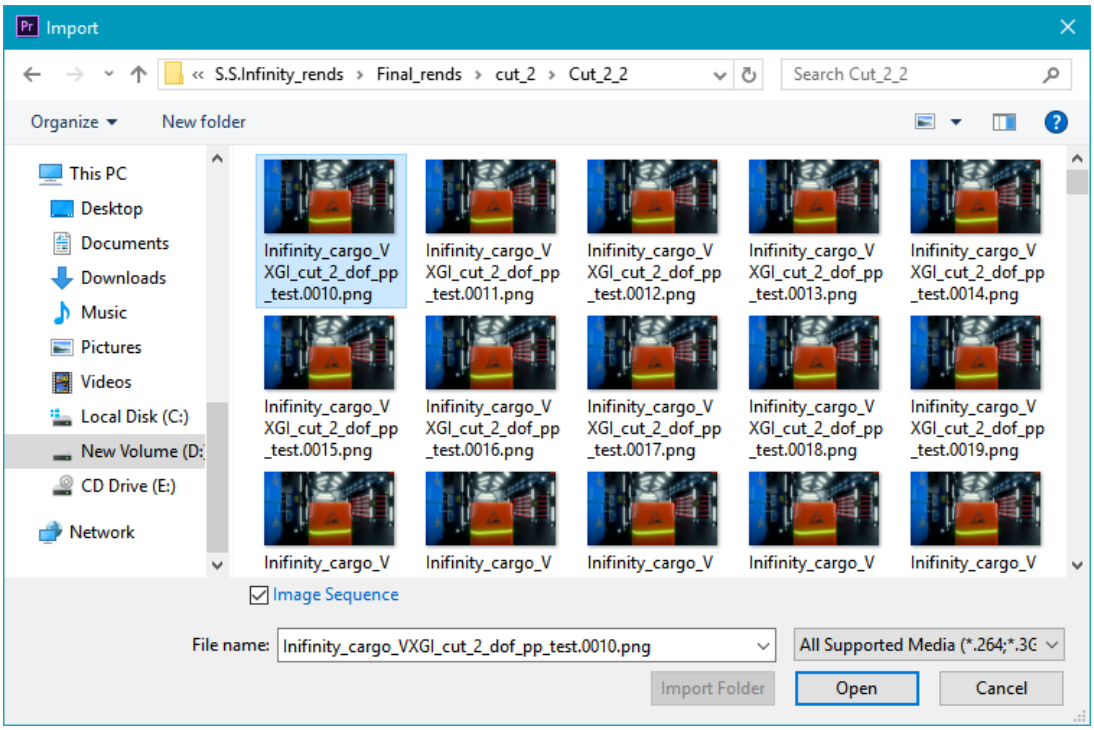


Slika 5.7. Prozor za izvoz slika kadra.

5.4. Izrada video datoteke

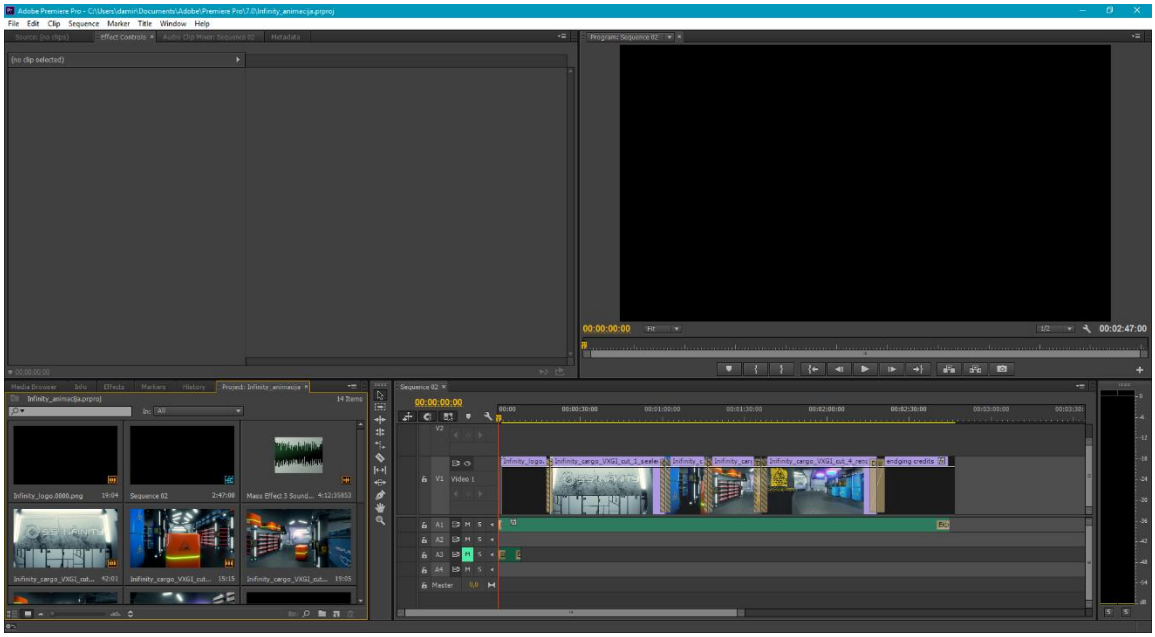
Za izradu video datoteke korišten je Adobe Premiere CC, programski paket namijenjen za nelinearnu montažu videa. Za izradu videa u Adobe Premiere CC, potrebno je otvoriti novi projekt i novu sekvencu. Zatim je potrebno uvesti sve izvezene kadrove iz Unreal Engine 4 i uvesti ih u video trake Adobe Premiere CC kako bi se mogle uređivati i spajati s ostalima.

Otvaranje novog projekta omogućeno je u polaznom prozoru prilikom otvaranja Adobe Premiere CC. Nova sekvenca dodaje se padajućim izbornikom *File* > *new* > *sequence*. Uvoz datoteka se postiže pristupom prozoru za uvoz datoteka na *File* > *import*. U prozoru za izvoz datoteka potrebno je označiti opciju *image sequence* kako bi program znao da je potrebno uvesti seriju slika, a ne samo jednu.

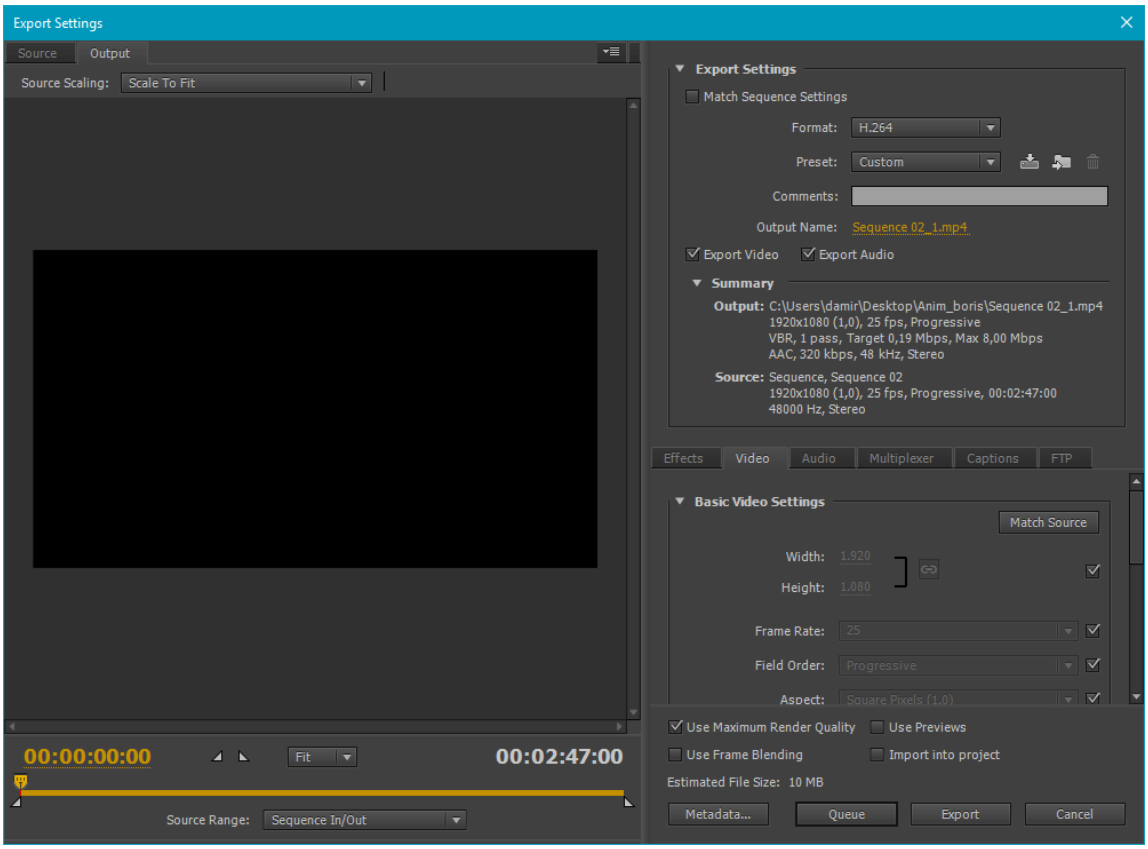


Slika 5.8. Uvoz slika kadrova.

Nakon što su svi kadrovi uvezeni u video traku i posloženi u željeni redoslijed, potrebno je bilo napraviti najavne i odjavne špice te dodati glazbu. Video je nakon toga bilo potrebno izvesti u MPEG-4 kontejner s .h264 načinom kodiranja. U prozoru za izvoz videa, odabrane su predefinirane postavke koje omogućuju kodiranje u .h264 *kodek* koji omogućuje spremanje video datoteke maksimalne kvalitete videa, uz zadovoljavajuću veličinu datoteke na hard disku računala. Pristup prozoru za izvoz videa omogućen je padajućim izbornikom *File* > *export media*.



Slika 5.9. Izrađena montaža videa u Adobe Premiere CC.



Slika 5.10. Prozor za izvoz video datoteke u Adobe Premiere CC.

Finalna animacija traje dulje no što traje u *sequenceru*. To je tako zbog različitog broja slika u sekundi. Naime, slike su izvezene u formatu 30 slika u sekundi. Dok je u Adobe Premiere CC sekvenca podešena na 25 slika u sekundi. Izvoz po 30 slika u sekundi omogućio je glatku animaciju konačne video datoteke. Unreal Engine 4 prikazuje video sliku različito od obične video datoteke, tako da je broj slika u sekundi ključan za glatko izvođenje videa, s toga je bio podešen na 30 slika u sekundi.

6. Zaključak

Kako bi se uopće moglo krenuti izrađivati 3D okruženje za računalnu igru potrebno je u osnovama poznavati elemente koji tvore neku igru, bilo ona računalna ili ne. Također, potrebno je znati što je to 3D okruženje i u koju svrhu se izrađuje. Taj posao obavlja level dizajner koji u suradnji s producentom, umjetničkim redateljem i dizajnerom igre izrađuje 3D okruženje. Level dizajner ne izrađuje samo 3D okruženje neke računalne igre. On koristi osvjetljenje, ambijentalne zvukove i mnoge druge elemente kako bi stvorio sveukupnu atmosferu igre koja čini igračevo iskustvo igranja.

Razvoj 3D okruženja za računalne igre velik je pothvat. Razvojni timovi računalnih igara koriste već ustaljene tijekove rada kako bi što učinkovitije iskoristili vrijeme. Tijekovi rada postoje i na osobnoj razini svih zaposlenika u razvojnim kućama. Zaposlenici razvojnih timova često usklade svoj tijek rada s onim razvojne kuće. Za razvoj S.S. Infinity 3D okruženja bilo je potrebno razviti novi tijek rada kako bi se maksimalno iskoristilo vrijeme. Najveći dio tijeka rada na izradi bilo je teksturiranje 3D modela. Substance programski paketi su izrazito moćni alati za izradu tekstura. Izrada tekstura unutar ta dva programska paketa odvija se proceduralno, što omogućuje stvaranje jedinstvenih tekstura. Izrada tekstura tako zahtjeva veliku maštu i odlično poznavanje korištenih programskih paketa. Tako izrađene teksture vrlo je lako skalirati na više ili niže rezolucije jednom kada su uvezeni pokretač igre kao što je to Unreal Engine 4. Istovremena povezanost između oba Substance programska paketa najviše je doprinijela cjelokupnom tijeku rada projekta S.S. Infinity.

Unreal Engine 4 je odličan, ako ne i najbolji besplatan pokretač igre kojim je moguće izrađivati izuzetno realna 3D okruženja. Velika prednost kod izrade 3D okruženja u Unreal Engine 4 je što se do velike većine mogućnosti grafičkog sustava može pristupiti uz malo ili skoro ni malo programiranja. Unreal Engine 4 implementira PBR način prikaza tekstura kojim se postiže fizikalno točan prikaz tekstura. Ono što ga stvarno čini izuzetnim je kvaliteta osvjetljenja koja se može postići njime. Sustavom *lightmass* moguće je ostvariti gotovo fizički realno osvjetljenje koje se izvodi u stvarnom vremenu. Grafičke mogućnosti Unreal Engine 4 privlače sve više ljudi koji se bave vizualizacijama arhitekture što definitivno doprinosi njegovom daljnjem razvoju. U trenutku pisanja ovog rada Unreal Engine 4 je dostupan u verziji 4.16.

Rezultat projekta S.S Infinity je video animacija preleta kamere kroz 3D okruženje u trajanju od 2:47 minute. Animacijom se pokazuje rani stadij razvoja računalne igre. Također, u animaciji je prikazan prototip mehanike igre. Izrađeno 3D okruženje u potpunosti je moguće iskoristiti za daljnji razvoj računalne igre. Za potrebe računalne igre S.S. Infinity bilo bi potrebno izraditi još tri prostorije. No, potpuno 3D okruženje je samo dio sveukupnog opsega posla kojeg je potrebno obaviti kako bi igra bila funkcionalna. Dokumentirani tijek rada može poslužiti nekome tko tek ulazi u svijet level dizajna, ali on nikako nije smjernica kako izraditi dobro 3D okruženje za računalnu igru.

Potpis

_____

U Varaždinu, srpanj 2017.

7. Literatura

[1] Adams, Ernest. Fundamentals of Game Design, 2nd Edition. Berkley, SAD: New Riders, 2009.

[2] Huzinga, Johan. Homo Ludens, Zagreb: Naprijed, 1992.

[3] Fullerton, Tracy. Game design workshop, 2nd Edition. Burlington, SAD: Elsevier, 2008.

[4] W. Toten, Christopher. An architectual aproach to level design. Virginia, SAD: CRC press, 2014.

[5] Autodesk.com - https://www.autodesk.com/
[Pristupljeno: 13.06.2017.]

[6] Allegorithmic, Substance Painter-
https://support.allegorithmic.com/documentation/display/SPDOC/Substance+Painter
[Pristupljeno: 16.06.2017.]

[7] Allegorithmic, Substance Designer-
https://support.allegorithmic.com/documentation/display/SD5/Substance+Designer+User+Guide
[Pristupljeno: 16.06.2017.]

[8] Epic Games- https://www.epicgames.com/about
[Pristupljeno: 15.06.2017.]

[9] McDermott, Wes The Comprehensive PBR Guide by Allegorithmic - vol. 1. California, SAD: Allegorithmic, 2017.
[Dostupno na
https://www.allegorithmic.com/system/files/software/download/build/PBR_Guide_Vol.1.pdf
Pristupljeno: 15.06.2017.]

[10] Allegorithmic, Izrada normal map teksture - https://support.allegorithmic.com/documentation/display/SD5/Normal+Map+from+Mesh
[Pristupljeno: 16.06.2017.]

[11] Allegorithmic, Substance Painter 2 izvoz tekstura - https://support.allegorithmic.com/documentation/display/SPDOC/Exporting+textures
[Pristupljeno: 16.06.2017.]

[12] Allegorithmic, Substance Designer pakiranje tekstura - https://support.allegorithmic.com/documentation/display/SD5/Packages
[Pristupljeno: 17.06.2017.]

[13] Epic Games, Material Editor - https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/Editor/
[Pristupljeno: 17.06.2017.]

[14] Epic Games, Content Browser - https://docs.unrealengine.com/latest/INT/Engine/Content/Browser/index.html
[Pristupljeno: 18.06.2017.]

[15] Epic Games, tipovi osvjetljenja - https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/LightTypes/index.html
[Pristupljeno: 29.06.2017.]

[16] Nvidia, VXGI - https://developer.nvidia.com/vxgi
[Pristupljeno: 29.06.2017.]

[17] Epic Games, Sequencer - https://docs.unrealengine.com/latest/INT/Engine/Sequencer/index.html
[Pristupljeno: 29.06.2017.

8. Prilozi

Video animacija dostupna je na:

https://vimeo.com/197947676?ref=fb-share

CD