OULUN AMMATTIKORKEAKOULU

Kalle Sirkka

# 3D-pelin toteuttaminen Unreal Enginellä

# 3D-pelin toteuttaminen Unreal Enginellä

Kalle Sirkka
Opinnäytetyö
Kevät 2016
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

**TIIVISTELMÄ**

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, Web-sovelluskehityksen suuntautumisvaihtoehto

---

Tekijä(t): Kalle Sirkka
Opinnäytetyön nimi: 3D-pelin toteuttaminen Unreal Enginellä
Työn ohjaaja: Matti Viitala
Työn valmistumislukukausi- ja vuosi: Kevät 2016 Sivumäärä: 37

---

Opinnäytetyön tavoitteena oli ensisijaisesti luoda 3D-peli portfoliota varten, mutta samalla opetella 3D-mallintamista Blenderillä ja pelimoottori Unreal Enginen käyttöä. Peli on ensimmäisen persoonan seikkailupeli, missä tutkitaan kaksikerroksista rakennusta ja sen pihapiiriä ja ratkotaan etenemistä estäviä pulmia.

Prosessi aloitettiin tutkimalla Unreal Enginen käyttöä ja 3D-mallintamista, sekä perehtymällä kenttäsuunnitteluun. Pelikenttä kehitettiin mallintamalla ensin talo ja sen piha Unreal Enginessä ja kalustamalla talo Blenderissä luoduilla 3D-malleilla. Unreal Enginessä käytettiin blueprint-pohjaista projektia.

Lopputuloksena oli hyvä pohja pelille, mutta jatkokehitystä tarvitaan portfoliokelpoisuuden saavuttamiseksi. Ongelmana oli ajankäytön haasteellisuus ja huonosti tehty suunnitteluvaihe. Taidot 3D-mallintamisessa, kenttäsuunnittelussa ja Unreal Enginen käyttämisessä, sekä käsitys työhön kuluvasta ajasta paranivat huomattavasti. Pelin kehittämistä tullaan jatkamaan opintojen jälkeenkin.

---

Asiasanat: pelinkehitys, pelit

## ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Option of Web Application Development

---

Author(s): Kalle Sirkka
Title of thesis: 3D-pelin toteuttaminen Unreal Enginellä
Supervisor(s): Matti Viitala
Term and year when the thesis was submitted: Spring 2016 Number of pages: 37

---

The main goal for the thesis was to create an example game for portfolio but also to learn more about 3D modelling with Blender and game development with Unreal Engine. The game is first person adventure game where the player character searches for clues in the villa and its surroundings.

The game development process was started by searching for information about Unreal Engine, 3D modelling and level designing. The villa and its surroundings were first created in Unreal Engine and after that filled with furniture modelled in Blender.

The outcome of the thesis is a good base for the game but main goal was not met. The biggest problems were poor time management and only half finished design work. There were significant improvement in 3D modelling, level design and Unreal Engine skills. The development of the game will be continued.

---

Keywords: game development, games

# SISÄLLYS

# 1 JOHDANTO

Olen aina ollut kiinnostunut peleistä, niin pelaajana kuin pelintekijänäkin. Pelit tuntuivat pitkään asialta, mitä haluaa nimenomaan harrastaa, ei tehdä työkseen, mutta erityisesti opinnot Oulu Game Labissa selkeyttivät haaveita ammatista pelien parissa.

Tarkoituksena oli luoda työnäyte portfoliota varten. Ensimmäisen persoonan seikkailupeli tuntui erinomaiselta paikalta esitellä omia taitoja ja erityisesti kehittää niitä entistä pidemmälle. Tarjolla olisi mm. pelisuunnittelua, ohjelmointia, 3D-mallintamista ja äänisuunnittelua joista kaksi jälkimmäistä eivät ole koskaan kuuluneet omiin vahvuuksiin vaikka kiinnostavia asioita ovatkin.

Peli on toteutettu Unreal Engine –pelimoottorilla käyttämällä blueprint-pohjaista projektia ja sitä varten tarvitut 3D-kappaleet Blenderillä. Tekstuurien käsittelyyn on käytetty GIMP-kuvankäsittelyohjelmaa. Äänienkäsittelyyn on käytetty Audacityä.

Suomessa peliteollisuus on jatkuvasti kasvava ala ja sen vuoden 2015 liikevaihdoksi on arvioitu noin 2,4 miljardia euroa eli noin kolmanneksen vuoden 2014 liikevaihtoon verrattuna. Maailmalla liikevaihdoksi on arvioitu 84 miljardia euroa, mikä merkitsee yhdeksän prosentin kasvua aiempaan vuoteen verrattuna. Kasvua on tapahtunut erityisesti Aasian markkinoiden kasvun myötä. (Neogames Finland ry. 2016. Viitattu 28.5.2016.)

# 2 PELIKEHITYKSEN TYÖKALUT

Pelinkehitykseen soveltuvia työkaluja on tarjolla todella paljon ja yhä useampi niistä on saatavilla ilmaiseksi. Madaltuneet hankintakustannukset ovat tuoneet aivan uusia mahdollisuuksia harrastelijoille ja pienille tiimeille, joille satojen tuhansien dollarien sijoitukset ovat olleet mahdottomia.

## 2.1 Pelimoottorit

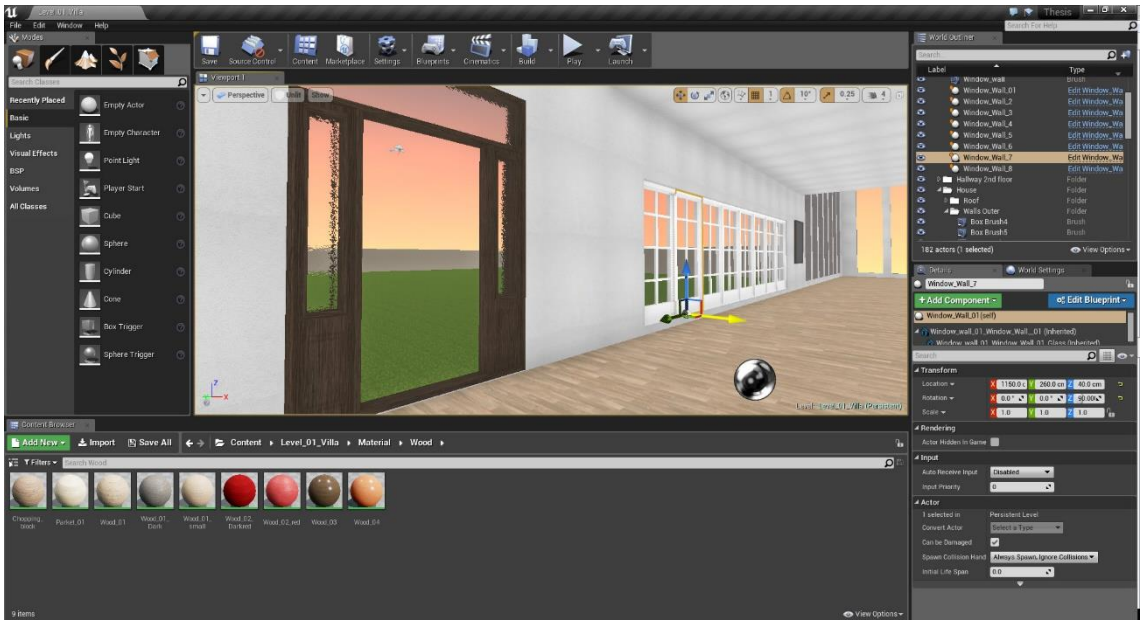
Pelimoottori nimestään huolimatta ei ole vain pelin käynnissä pitävä asia, vaan lajitelma uudelleenkäytettäviä työkaluja joita käyttämällä tai muokkaamalla peli valmistetaan. Valmiiden ja toimiviksi todettujen työkalujen käyttäminen säästää aikaa ja luo vakaamman pohjan pelin sisällön kehittämiselle.

Pelimoottorin tehtävät vaihtelevat, mutta pääsääntöisesti pelimoottorien avulla voidaan ladata ja käyttää assetteja, tarkkailla eri kappaleiden välisiä törmäämisiä, käsitellä pelaajalta tulevaa syötettä ja jopa rakentaa tekoäly pelihahmoille. Nykyään pelimoottorit myös miltei poikkeuksetta sisältävät fysiikanmallinnuksen, joten erilliselle fysiikkamoottorille ei ole tarvetta. (Ward, J. 2008, viitattu 11.5.2016.)

Pelimoottoreista suosituimpia ovat Unity ja Unreal Engine. Unity on nuoresta iästään huolimatta kerännyt suuren suosion erityisesti pienten yritysten käytössä, Unreal Enginellä taas on pitkät juuret suurten pelijulkaisujen kehityksessä. Suomessa Unity on yleisin pelimoottorivaihtoehto pelialan yrityksissä, mutta muiden kilpailijoiden madaltuneet kustannukset sekoittavat tilannetta.

### 2.1.1 Unreal Engine

Epic Gamesin kehittämä Unreal Engine 4 –pelimoottori (KUVIO 1) taipuu tiimin koosta riippumatta paitsi suuriin tietokonepeleihin, myös mobiilipeleihin. Lisensoinnin suhteen Unreal Engine on hyvin suoraviivainen: maksa rojalti peleistä ja sovelluksista joita julkaiset. Alle 3000 dollarin tuotoista ei maksua kerry, mutta summan ylittymisen jälkeen vaaditaan viiden prosentin rojalti vuosineljänneksittäin. (Epic Games 2016, viitattu 24.4.2016.)



*KUVIO 1. Unreal Enginen käyttöliittymä.*

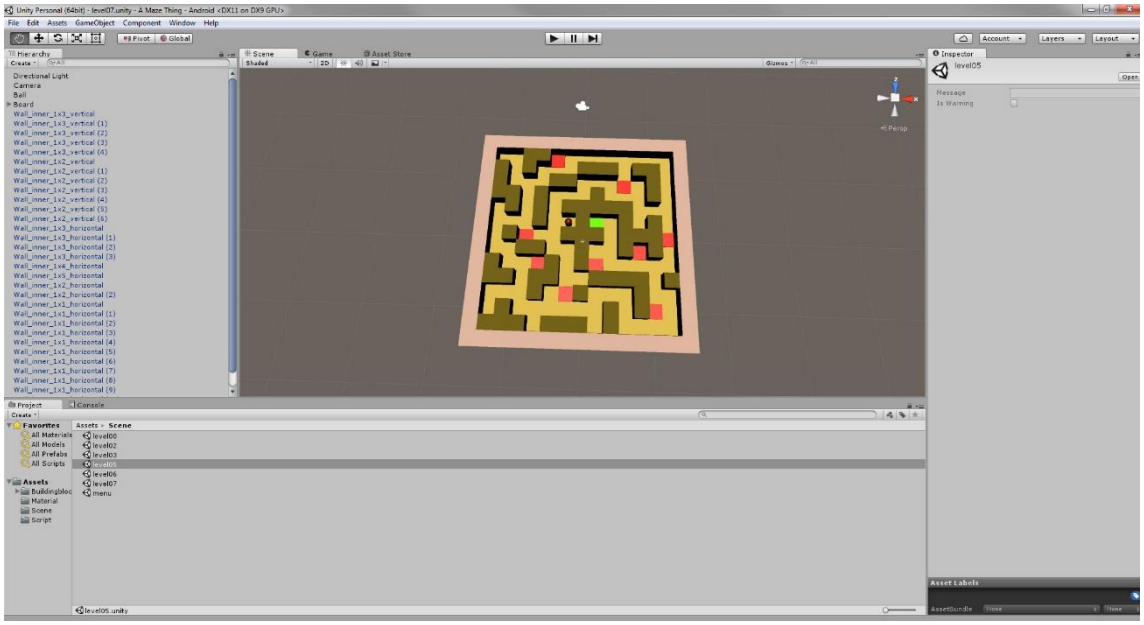
Tunnettuja Unreal Engine 4:llä tehtyjä pelejä ovat mm. Street Fighter V ja ARK: Survival Evolved. Suuria UE4-julkaisuja ei ole vielä montaa, mutta aiemmalla versiolla onkin sitten toteutettu mm. Batman Arkham City, Bioshock Infinite ja Mass Effect (Kim, B. 2014, viitattu 24.4.2016).

Pelien ohjelmointiin Unreal Engine 4:ssä käytetään C++-ohjelmointikieltä tai Blueprinttiä. Blueprint on visuaalinen, solmupohjainen (node-based) tapa käsitellä pelimoottorin toimintoja. Perinteisesti valtaosa pelimoottorin toiminnoista on ollut vain ohjelmoijien käytettävissä, mutta Blueprintit tekevät ohjelmoimisen helpommin lähestyttäväksi myös ohjelmointia osaamattomille. (Epic Games 2015, viitattu 24.4.2016.)

Syinä Unreal Enginen valintaan opinnäytetyön kehitysympäristöksi olivat uteliaisuus Blueprint-järjestelmää kohtaan, sekä upeat valotehosteet. Valojen merkitys oli tärkeä, sillä laadukkaalla valonmallinnuksella yksinkertainenkin grafiikka näyttää paljon aidommalta.

### 2.1.2 Unity

Unity (KUVIO 2) on Unity Technologiesin ylläpitämä monialustainen pelimoottori. Sillä onnistuu 2D ja 3D-pelien kehittäminen paitsi konsoleille tai mobiililaitteille, myös internetiin, älytelevisioihin ja virtual reality –laitteille. Ohjelmointi onnistuu kielillä C# ja Javascript. (Unity Technologies 2016a, viitattu 20.4.2016.)



*KUVIO 2. Unityn käyttöliittymä.*

Unityllä on tehtyihin peleihin lukeutuvat mm. Firewatch, Kerbal Space Program, Pony Island, Ori and the Blind Forest (Unity Technologies 2016b, viitattu 20.4.2016). Lisensoinnin suhteen Unity on ilmainen yksityiseen käyttöön. Unitylla on 4,5 miljoonaa rekisteröityä käyttäjää ympäri maailman ja sen osuus pelimoottorimarkkinoista on 45 prosenttia (Unity Technologies 2016c, viitattu 8.5.2016).

### 2.1.3 Muita vaihtoehtoja

Amazonin kehittämä Lumberyard on aivan uusi kilpailija pelimoottoririntamalla. Sen valttina on integrointi AWS Cloud –palveluun, mikä tekee moninpelien kehittämisestä paljon helpompaa. Lumberyard on lähdekoodeineen saatavilla ilmaisesti, ainoat maksut tulevat niistä Amazonin palveluista mitä kehittäjä päättää käyttää.

Pelimoottorissa on huomioitu huippusuosittu, peleihin erikoistunut striimauspalvelu Twitch. Tarjolla on nippu työkaluja joiden avulla striimin katsojat voivat osallistua striimattavaan peliin esimerkiksi äänestämällä peliin liittyvissä päätöksissä. (Amazon Web Services 2016a, viitattu 26.4.2016.)

Tällä hetkellä Lumberyardilla voi tehdä pelejä vain tietokoneelle, Xbox Onelle ja Playstation 4:lle (Amazon Web Services 2016b, viitattu 26.4.2016). Pelimoottorin nuoresta iästä johtuen sillä tehtyjä pelejä ei vielä löydy, samoin Amazonin ulkopuolista oppimateriaalia löytyy melko vähän.

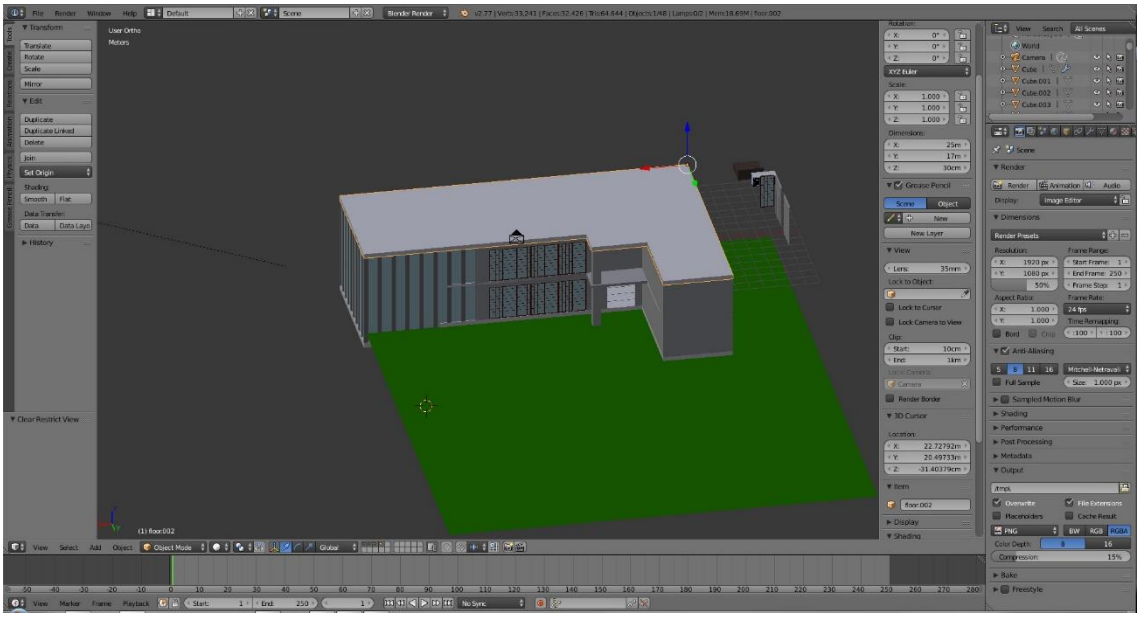
Cryengine on saksalaisen Crytekin tekemä pelimoottori joka on erityisen tunnettu kyvystä esittää äärimmäisen monimutkaista grafiikkaa hyvin avoimella pelialueella. Pelimoottorilla tehtyihin peleihin lukeutuvat esimerkiksi Crysis, Evolve ja Star Citizen (Crytek 2016b, viitattu 26.4.2016).

## 2.2 Assettien tekemiseen käytettävät työkalut

Pelimoottorin lisäksi peliin tarvitaan assetteja, joita ovat mm. 3D-mallit, tekstuurit ja äänitehosteet. Assettien tekemiseen löytyy laaja kirjo erilaisia ohjelmia, niin maksullisia kuin ilmaisiakin.

### 2.2.1 Blender

Blender on ohjelmisto, jota käytetään 3D-mallintamisen ja renderöimisen ohella mm. animoimiseen, simulaatioihin ja videonkäsittelyyn. Sovellus on avointa lähdekoodia ja toimii useissa käyttöjärjestelmissä. (Blender Foundation 2016, viitattu 19.4.2016.)



*KUVIO 3. Blenderin käyttöliittymä.*

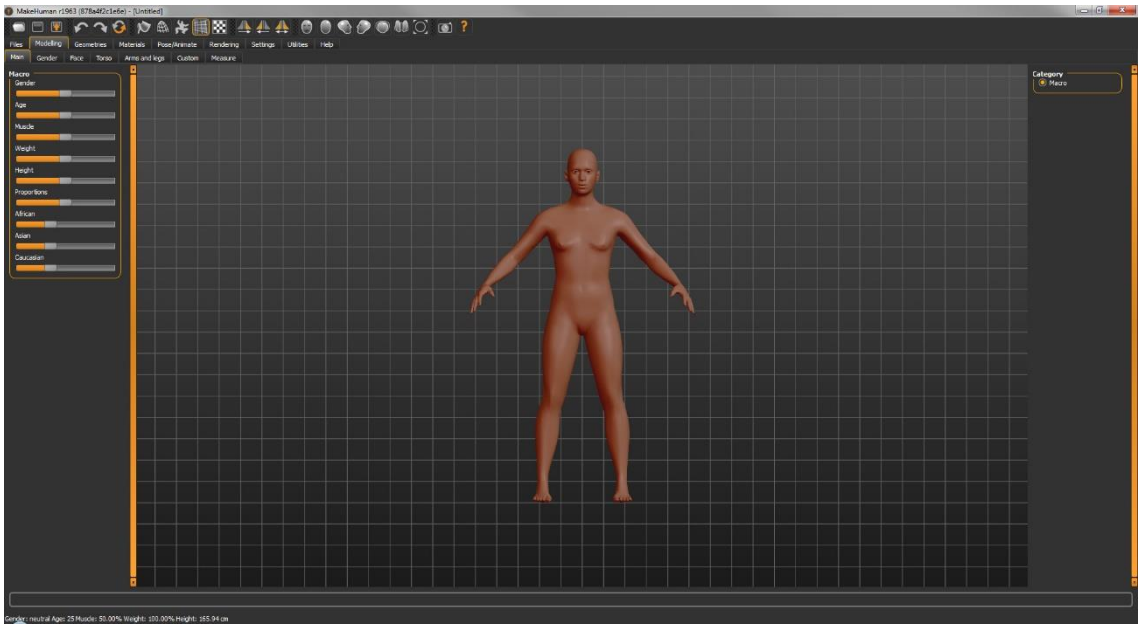
Äkkiseltään monimutkaiselta vaikuttava käyttöliittymä (KUVIO 3) on pohjimmiltaan melko looginen ja sisältää useita tapoja tehdä sama asia. Ominaisuusvalikoima muuttuu eri tilojen mukaan, joten

objekteja tarkastellessa ei juurikaan nähdä tekstuurin maalaamisen liittyviä toimintoja, eikä veistostilassa tarvitse väistellä tekstuurien maalaamiseen tarvittavia työkaluja. Toki eri tilojen käyttö voi alkuun vaatia totuttelua.

Muita 3D-mallinnusohjelmia käyttäneille Blenderiin siirtymistä helpottaa mahdollisuus vaihtaa pikanäppäimet, hiiren käyttäytyminen ja muut syötteeseen liittyvät asiat esimerkiksi 3DS Maxia muistuttaviksi. Muokkausmahdollisuuksia on muutenkin laajasti.

### 2.2.2 MakeHuman

MakeHuman on ilmainen, avoimen lähdekoodin ohjelma ihmismallien luomiseen. Se tarjoaa satoja asetuksia, joilla voi säätää kaikkea vartalon muodoista kasvojen yksityiskohtiin (KUVIO 4). Ohjelmassa käytetään Creative Commons 0 -lisenssiä, eli sillä tehtyjä malleja voidaan käyttää täysin vapaasti myös kaupallisissa tuotteissa. (MakeHuman Team 2016, viitattu 15.4.2016.)




*KUVIO 4. MakeHumanin käyttöliittymä.*

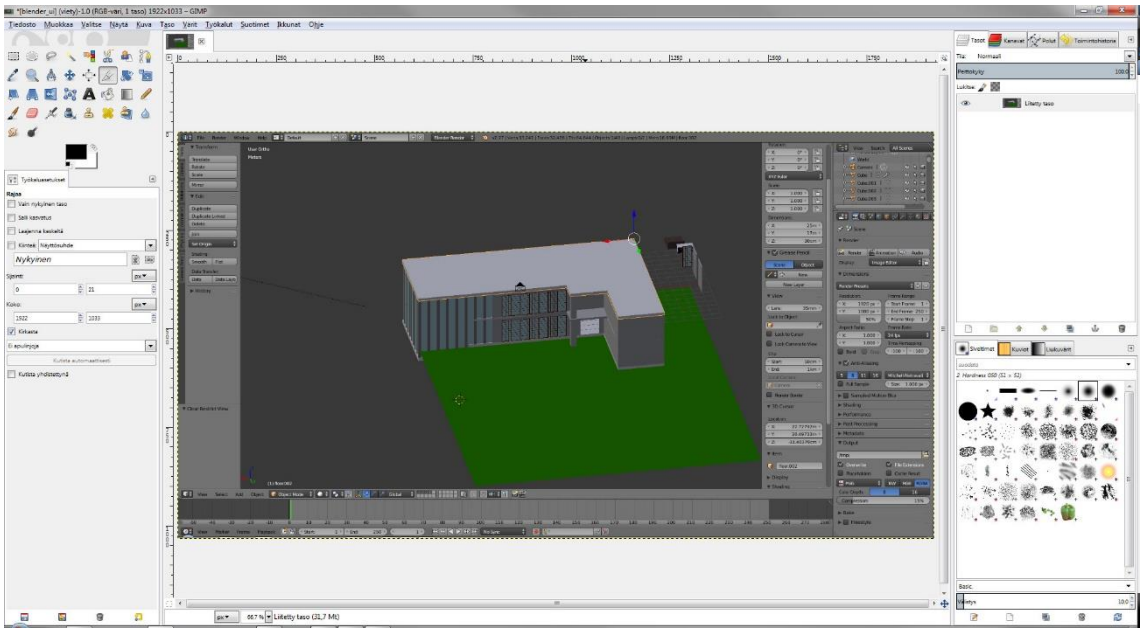
Nopeus on MakeHumanin tärkein valtti. Sen avulla voidaan luoda skinnattu ja rigattu 3D-hahmo normal mappeineen ja tekstuureineen parissa minuutissa. Heikkoutena valikoima kaiken muun paitsi kehonmuotojen suhteen on melko vaisu. Vaatteita, hiuksia ja sen sellaisia ei yksinkertaisesti ole riittävästi ja puuttuvat asiat on sitten toteutettava itse.

MakeHumanilla oli tarkoitus tehdä hahmoja pelissä nähtäviä valokuvia varten, mutta peli ei opinnäytetyön puitteissa edennyt riittävän pitkälle. Sovelluksen rooli opinnäytetyössä jäi koekäytön asteelle. Valokuviin riittävää ilmeikkyyttä oli haastavaa luoda, sillä yksinkertaisetkin ilmeet muodostuvat lukemattomista yksityiskohdista, tämä kuitenkaan ei varsinaisesti ollut sovelluksen syytä.

### 2.2.3 GIMP

GIMP on avoimen lähdekoodin kuvankäsittelyohjelma, mikä tarjoaa työkalut valokuvien käsittelemiseen, digitaaliseen maalaamiseen ja graafiseen suunnitteluun. Lisää ominaisuuksia on mahdollista hankkia laajennusten muodossa tai skripteillä, joita voi kirjoittaa mm. Python- ja Perl-ohjelmointikielillä. Nimi GIMP on lyhenne sanoista GNU Image Manipulation Program. (The GIMP Team 2015, viitattu 24.4.2016.)

Ohjelman kehitys on alkanut jo vuonna 1995 (The GIMP Team 2015, viitattu 24.4.2016). Ikä näkyy käyttöliittymästä (KUVIO 5), mikä jatkuvasta kehitysestä huolimatta on paikoin sekava ja vaatii jonkin verran opettelua yksinkertaisempiin ohjelmiin tottuneelta.

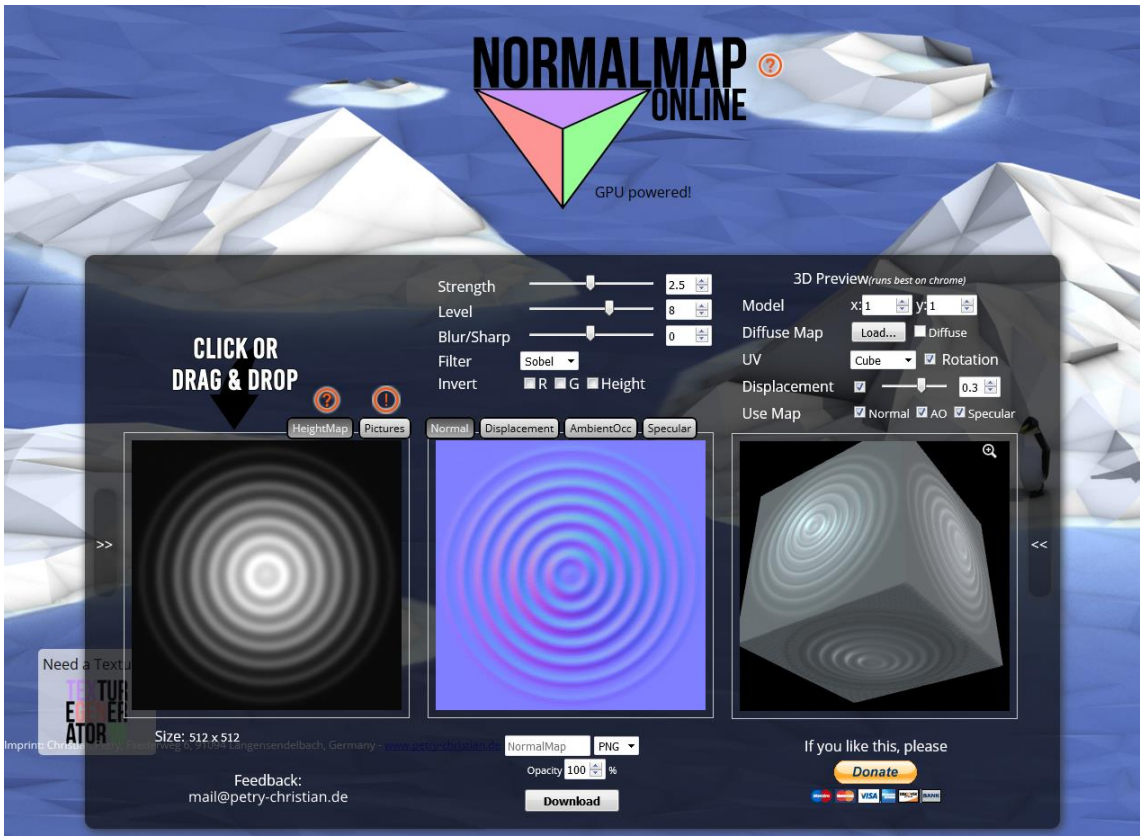


*KUVIO 5. GIMP-kuvankäsittelyohjelman käyttöliittymä.*

GIMPin tärkeimmät tehtävät opinnäytetyötä tehdessä oli kuvakaappausten muokkaamisen ohella tekstuurien käsittely ja pinnan kiiltämistä ilmaisevan specular mapin tekeminen. Valtaosa kuvankäsittelystä oli yksinkertaista värien ja kontrastin säätämistä.

### 2.2.4 NormalMap-Online

Normalmap-tiedostoja voi tehdä monella tapaa, mutta vaivattomin on niiden generoiminen tekstuurin pohjalta. Opinnäytetyössä päädyttiin internet-selaimessa toimivaan NormalMap-Onlineen, mikä on paitsi ilmainen, myös hyvin selkeä käyttöliittymältään (KUVIO 6).




*KUVIO 6: NormalMap-Onlinen käyttöliittymä.*

Ohjelmaan voi raahata halutun kuvan ja asetuksia säätämällä luoda normalmap-tiedostosta halutun kaltaisen. Samassa näkymässä on kokoajan esikatselu, mikä tekee asetusten valitsemisesta helppoa. (Petry C. 2014, viitattu 10.5.2016.)

### 2.2.5 Audacity

Äänen tallentamiseen ja käsittelemiseen ilmaisen, avoimen lähdekoodin ratkaisun tarjoaa Audacity. Ohjelman keveydestä huolimatta sillä pystyy käsittelemään esimerkiksi äänitehosteita hyvin monipuolisesti erilaisten efektien ja suotimien avulla ja lopulta muuntaa tehoste käytetyn pelimoottorin ymmärtämään muotoon. (Audacity Team 2016, viitattu 25.4.2016).

## 2.3 Assetteja tarjoavat palvelut

Kaikkien assettien luominen itse vie paljon aikaa, joten työskentelyprosessia voi suoraviivaistaa hakemalla esimerkiksi tekstuurit niitä tarjoavista palveluista. Tällaisia palveluita ovat mm. aiemmin CGTextures-nimellä tunnettu Textures.com ja Pixabay. Myös monilla pelimoottoreilla on omat, usein suoraa ohjelmistoon integroidut asset-kaupat.

Valmiiden assettejen käyttöpäätöstä tehdessä tulee huomioida nopeuden tuomat säästöt, mutta samalla assettien laatu ja yleisyys. Erityisesti ilmaisissa asseteissa laatu vaihtelee rajusti ja niiden toimintaan saattaminen voi viedä paljonkin aikaa. Muut pelintekijät, mutta myös osa pelaajista kuluttavat niin paljon aikaa pelin parissa, että he voivat tunnistaa muista peleistä tuttuja assetteja ja tämä puolestaan rikkoo pelikokemusta. Prototyyppiä toteuttaessa assettien yleisyydellä ei ole merkitystä, joten valmiille ihmismalleille on paikkansa.

Unreal Enginen asset-kauppa on nimeltään Marketplace. Tarjolla on laaja kirjo malliprojekteja, grafiikkaa, äänitehosteita ja koodia. Osa sisällöstä on pelimoottorin kehittäjän Epicin tuottamaa ja äärimmäisen korkealaatuista, mutta muutenkin kaupan tarjoamat assetit ovat tasokkaita. Ilmaista sisältöä Marketplacessa tosin on melko vähän.

Unityn Asset Store on täynnä assetteja, mutta niiden laatu vaihtelee suuresti. Aivan kuten Unreal Enginessäkin, myös Unityssä kauppa on integroitu pelikehitysympäristöön ja assettien hankkiminen ja projektiin lisääminen on todella helppoa.

Textures.com tarjoaa ilmaiskäyttäjille 15 krediittiä päivässä, joilla voi ladata tekstuureja sivustolta. Isommat tekstuurit vaativat maksullisia krediittejä, mutta ilmaisistakin löytyy paljon pelikäyttöön soveltuvaa materiaalia. Sivuston tarjoamia tekstuureja voi käyttää kaupallisesti, mutta myytävä tuote ei saa kilpailla Textures.comin kanssa. (Textures.com 2016. Viitattu 11.5.2016.)

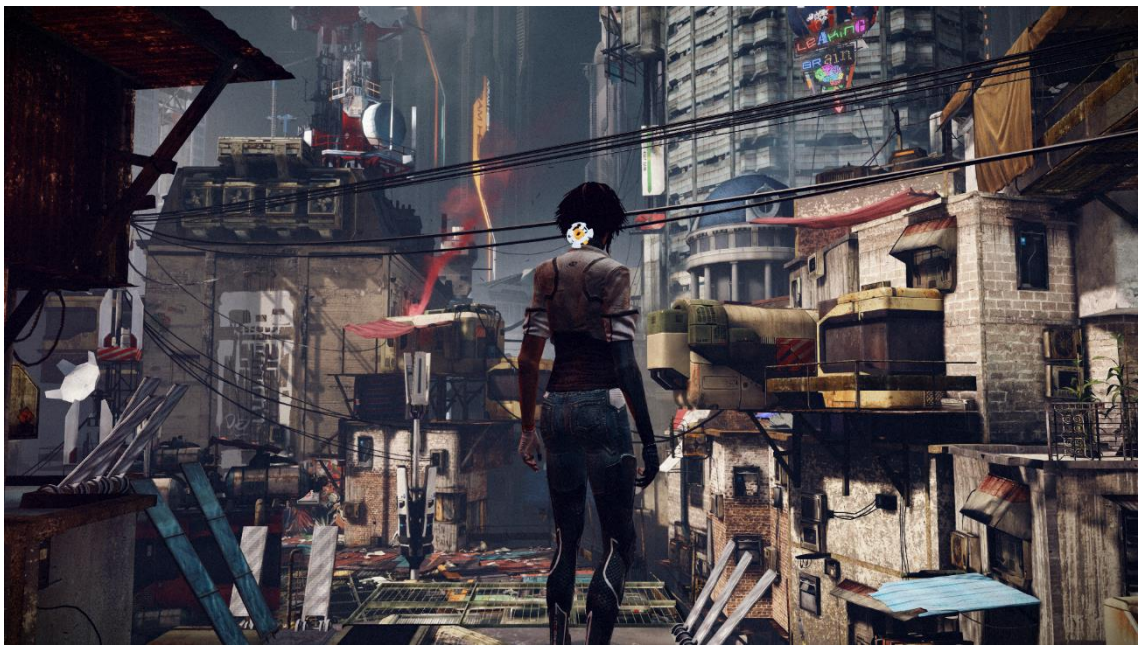
Pixabayn kuvat ovat Creative Commons 0 –lisensoituja, mikä tarkoittaa, että niitä voi kopioida, muokata ja levittää ilman erillisiä lupia tai mainintoja kuvan ottajasta. Lisenssi ei rajoita kaupallistakaan käyttöä, mutta on hyvä huomioida, että kuvissa voi olla esimerkiksi tuotemerkkejä, joiden käyttöön voi liittyä rajoitteita. Kuvat palveluun tulee käyttäjiltä, joten saatavilla oleva materiaali on laadultaan hyvin vaihtelevaa. (Pixabay 2016. Viitattu 7.5.2016.)

Tekstuurien ohella äänitehosteet oli helpompaa etsiä internetistä kuin tehdä itse. Freesound on suuri tietokanta erilaisista äänitehosteista. Nimensä mukaisesti palvelun käyttäminen on ilmaista, sillä äänitehosteet ovat Creative Commons –lisensoituja. Kaikkia tehosteita ei voi käyttää kaupallisesti ja osa vaatii nimeämään tekijän. (Freesound 2015. Viitattu 7.5.2016.)

# 3 KENTTÄSUUNNITTELUN TEORIAA

Monissa peleissä pelaajan pelimaailmassa navigoimista helpotetaan käyttöliittymän avulla. Tämä voi tarkoittaa esimerkiksi tehtäväkompassia, mikä osoittaa aina paikkaa minne pitää mennä seuraavaksi tai jopa pelimaailmaan piirtyvää nuolta mitä seuraamalla pääsee helposti perille. Peliin uppoutumisen kannalta on usein eduksi muokata maailmasta sellainen, missä pelaajan on helppo kulkea ilman keinotekoista apua. Pelaajaa voi opastaa muun muassa valolla, ympäristön muodoilla, liikkeellä, pelimekaniikoilla, väreillä, äänitehosteilla ja kiintopisteillä (Brown, M. 2015, viitattu 26.4.2016).

Valoisat paikat tuntuvat turvallisilta ja siksi valo kiinnittää katseen. Valojen ohella liike on erittäin toimiva huomionherättäjä. Hyvin usein peleissä näkee esimerkiksi lentoon pyrähtäviä lintuja pelin etenemisen kannalta kiinnostavasta suunnasta. Ääni herättää kiinnostuksen erityisesti silloin kun sen lähdettä ei voi nähdä.



*KUVIO 7. Pelaajan ohjaamista pelissä Remember Me. Tavoitteena on päästä baariin nimeltä Leaking Brain, minkä mainos näkyy oikealla ylhäällä.*

Erilaisia muotoja yhdistämällä voidaan kuva-ala täyttää kappaleilla jotka muodostavat käytäviä tai polkuja sellaisiinkin paikkoihin missä niitä ei oikeasti ole (KUVIO 7). Erilaiset kyltit ja esimerkiksi

liikennettä ohjastavat nuolet ovat myös tehokas ja sopivasti käytettynä huomaamaton tapa ohjastaa pelaajaa.

Erilaiset pelimekaniikat ohjaavat pelaajan ajattelua ja ongelmanratkaisutapoja. Hiiviskelypeliä pelaava etsii piilopaikkoja ja karttaa kaikkein valoisimpia alueita kun taas ammuskelupelin parissa pyritään löytämään kaikkea räjähtävältä vaikuttavaa.

Väreillä voidaan ehdollistaa pelaaja huomioimaan vaikkapa kiipeämiseen soveltuvia kohtia pelimaailmassa. Joskus värit saattavat olla hyvinkin selkeästi esillä, kuten Mirrors Edgessä jossa kaupungin pohjaväri on valkoinen ja etenemiseen soveltuvat yksityiskohdat kuten ovet on merkitty punaisella, mutta yleensä ne on piilotettu osaksi kenttää vaikkapa poliisin keltaisen eristysnauhan tai valkoisten köysien muodossa.

Kiintopiste on usein jotain kaukaisuudessa näkyvää. Se voi olla esimerkiksi rakennus tai erikoinen maastonmuoto, mikä pilkottaa lähempänä olevien kappaleiden lomasta. Esimerkiksi kaupungin keskellä saattaa olla jokin erityisen suuri rakennus minne pelaajan pitää päästä suorittaakseen annetun tehtävän. Rakennus on nähtävissä miltei kaikkialta ympäri kaupunkia, joten sokkeloisessakin ympäristössä pelaaja pystyy suunnistamaan kiintopistettä päin.

# 4 TOTEUTUS

## 4.1 Suunnittelu

Pelin suunnittelu alkoi rajoitteiden määrittämisellä. Aikaa olisi vähän ja omat taidot erityisesti 3D-mallintamisen suhteen sen verran heikot, että kovin laajaa peliä ei voisi mitenkään saada tehdyksi. Pelialueeksi muodostui nopeasti talo ja siihen liittyvä piha-alue. Pelistä tulisi seikkailupeli, jossa yksityisetsivänä toimiva pelihahmo koettaa löytää todisteita talosta silloin kun asukkaat ovat muualla.

Pelattava osuus koostuisi ongelmanratkonnasta, esimerkiksi lukitun oven avaamisesta, valvontakameran välttelemisestä ja niin edelleen. Pelin käyttöliittymässä näkyisi kuinka monta todistetta vaaditaan kentän läpäisemiseksi ja kuinka monta pelaaja on löytänyt.

Ajan säästämisen vuoksi peli on kuvattu ensimmäisestä persoonasta jolloin vältetään tarve näyttää animoitua pelihahmoa. Pelaajaa edustava hahmo kuitenkin tarvittaisiin heijastavia pintoja kuten peilejä varten. Kolmas persoona olisi vaatinut paitsi hahmon tekemisen, myös siihen liittyvät animaatiot, kameran liikkeet ja niin edelleen.

Kentän pienuudesta johtuen talon pohjapiirrokselle asettui paljon paineita: sen pitäisi tarjota riittävästi tutkittavaa, mutta myös jollain tapaa antaa vinkkejä pelaajan edistymisestä. Useamman piirroksen ja 3D-visualisoinnin jälkeen rakennuksesta tuli L-kirjaimen muotoinen ja kaksikerroksinen. Pelaajan suunniteltu reitti kulkee niin, että tärkeimmät todisteet sisältävä huone jää viimeiseksi. Reitti on toteutettu talon rakenteen, fyysisten esteiden kuten lukitun oven ja matkan varrella olevien kiinnostavien kohteiden avulla.

Kentän tärkein tila, työhuone, sijaitsee toisessa kerroksessa kauimpana portaista. Osa todisteista on vaikeasti löydettävissä, joten huoneen merkitystä korostamassa ovat punaiset sälekaihtimet (KUVIO 8), jotka myös näkyvät selvästi ulkoa katsottaessa.


*KUVIO 8. Kuvakaappaus keskeneräisestä työhuoneesta.*

## 4.2 Objektin matka

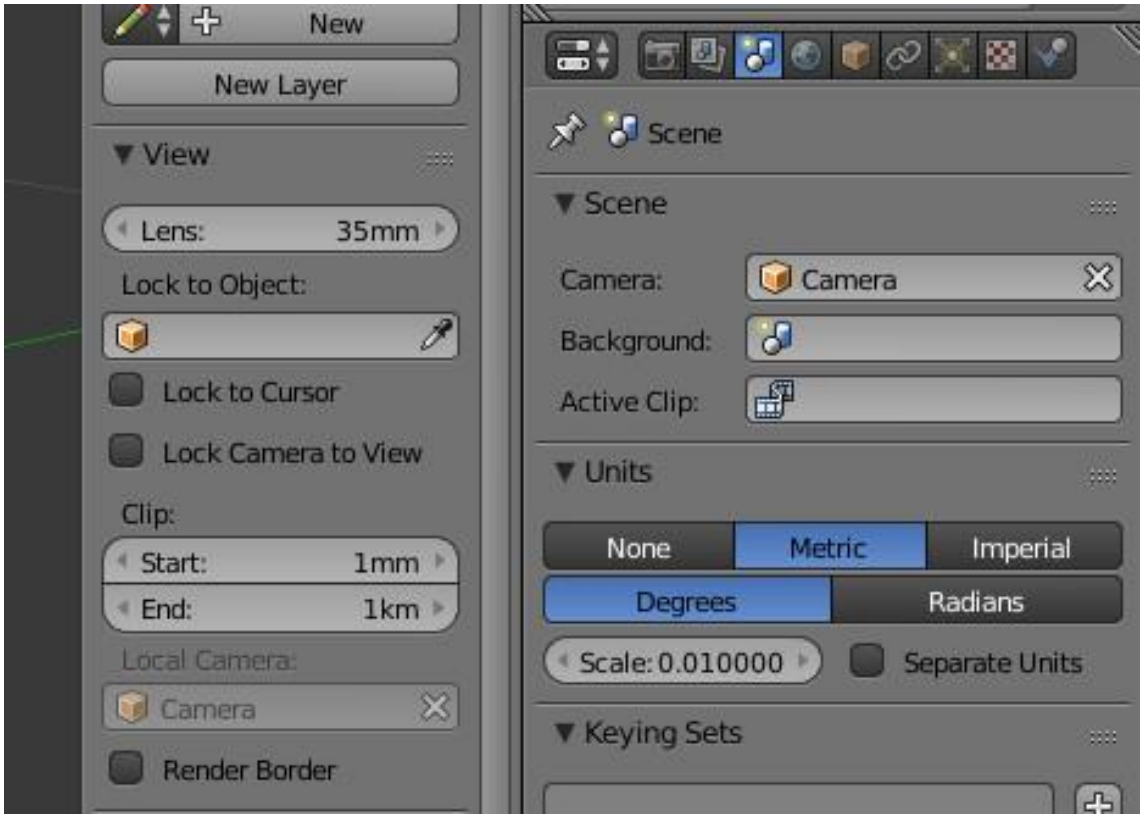
Kappaleessa seurataan yöpöydän matkaa aina suunnitteluvaiheesta pelissä nähtäväksi, interaktiiviseksi kappaleeksi. Työvaiheet on kuvattu mahdollisimman tarkasti, jotta saataisiin välitettyä mahdollisimman tarkka kuva eri työvaiheista ja niiden merkityksistä.

Erilaisilla esineillä, kalusteilla ja sen sellaisilla voi kertoa hyvin paljon niiden omistajista. Ehkä omistaja on elänyt keskinkertaista elämää kunnes yhtäkkiä rahaa onkin alkanut tulla yli tarpeiden, millaisessa talossa tällainen ihminen eläisi? Ehkä hän yrittäisi toteuttaa haaveita huvilasta ja kohta huomaisi, että kalustamaton suuri tila pitäisi täyttääkin jotenkin. Onko henkilöllä tyylitajua vai ostaako hän vain nopeasti jotain? Ovatko huonekalut harmoniassa keskenään vai ei? Tällaiset yksityiskohdat tuovat malleihin ja siten koko peliin elämäntuntua.

Tehtävästä kappaleesta on hyvä olla mallikuvia, erityisesti erilaisista yksityiskohdista ja materiaalista. Suuri apu on myös sellaisista kuvista, missä näkyy kappaleen mitat. Erityisesti huonekaluista mittakuvia löytyy todella runsaasti.

Mallintamiseen käytetään 3D-mallinnusohjelma Blenderiä. Aivan aluksi navigoidaan Blender-ikkunan oikeassa reunassa olevaan välilehtiosioon ja sieltä scene-välilehteen. Blenderin oletusmittayksikkö on Blend Unit. Erityisesti kalusteita mallintaessa on kuitenkin hyvä vaihtaa mittayksikkö tosimaailmassa käytetyksi, milloin mallikuvista löytyviä mittoja voi käyttää helposti. Blenderin metri on

vain senttimetri Unreal Enginessä, joten muutetaan mittakaava muotoon 0,01, mikä mahdollistaa mittojen hyödyntämisen mallintaessa ilman että Blenderistä tuodut kappaleet olisivat mikroskooppisia Unreal Enginessä. Samalla Blenderissä käytettyä näköetäisyyttä pitää muuttaa tai muuten vähänkin isommat kappaleet eivät näy kokonaan. Tämä tapahtuu oikeasta paneelista View-otsikon alta Clip-arvoja muuttamalla. Aloitusarvoa ei tarvitse muuttaa, mutta jälkimmäisen voi nostaa esimerkiksi kilometriin (KUVIO 9).




*KUVIO 9. Mittayksikkö ja –kaava sekä kappaleen näyttöetäisyyteen liittyvät asetukset Blenderissä.*

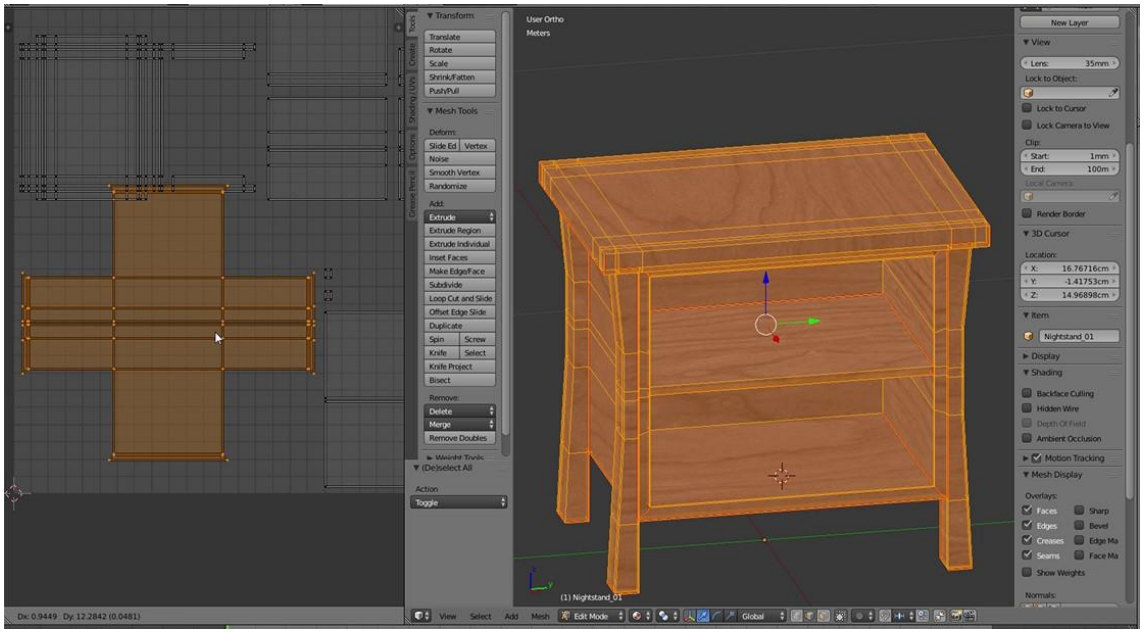
Blenderin numerokentät ovat todella monipuolisia ja mahdollistavat erilaisten mittayksiköiden käytön ja jopa laskutoimitukset. Projektiin, missä mittana käytetään metrejä, voi vaikka tuolin korkeuden syöttää jalkoina, milloin Blender muuntaa arvon metreiksi. Blender myös muuttaa automaattisesti desimaalipilkun pisteeksi, joten arvot on helppo syöttää suomalaisella näppäimistöllä. Unreal Engine ei tätä kuitenkaan osaa, joten siinä pitää käyttää desimaalipistettä.

Mittakaavan muuttamisen vuoksi oletuksena luodut kuutio, kamera ja valo ovat liian pieniä käytettäväksi, helpoin ongelman korjaamiseen on valita kaikki kappaleet a-näppäimellä ja valita x-näppäimellä avattavasta valikosta delete. Uuden, automaattisesti oikeassa mittakaavassa olevan kuution (tai muun primitiivin) saa Add-valikon alta löytyvästä Mesh-valikosta.

Edit-tilassa kappaletta voidaan muokata erilaisilla työkaluilla. Valittuja pintoja voi esimerkiksi pursottaa extrude-työkalulla, mikä löytyy e-näppäimen takaa. Pintoja voidaan leikata control + r-näppäin –yhdistelmän takaa löytyvällä loop cut –työkalulla.

Pelimoottori tarvitsee tiedon mallin geometriasta, jotta se osaisi asetella tekstuurin oikein mallin pinnalle. Tätä tietoa kutsutaan UV-kartaksi ja siinä kolmeulotteinen kappale on purettu kaksiuloitteiseksi. Helpointa on ajatella puusta rakennettua pöytää, missä eri osien kuviointi on toisistaan poikkeavaa. UV-kartan avulla 3D-malli voidaan purkaa eri osiksi ja näin samoja tekstuureja voidaan käyttää eri malleille.

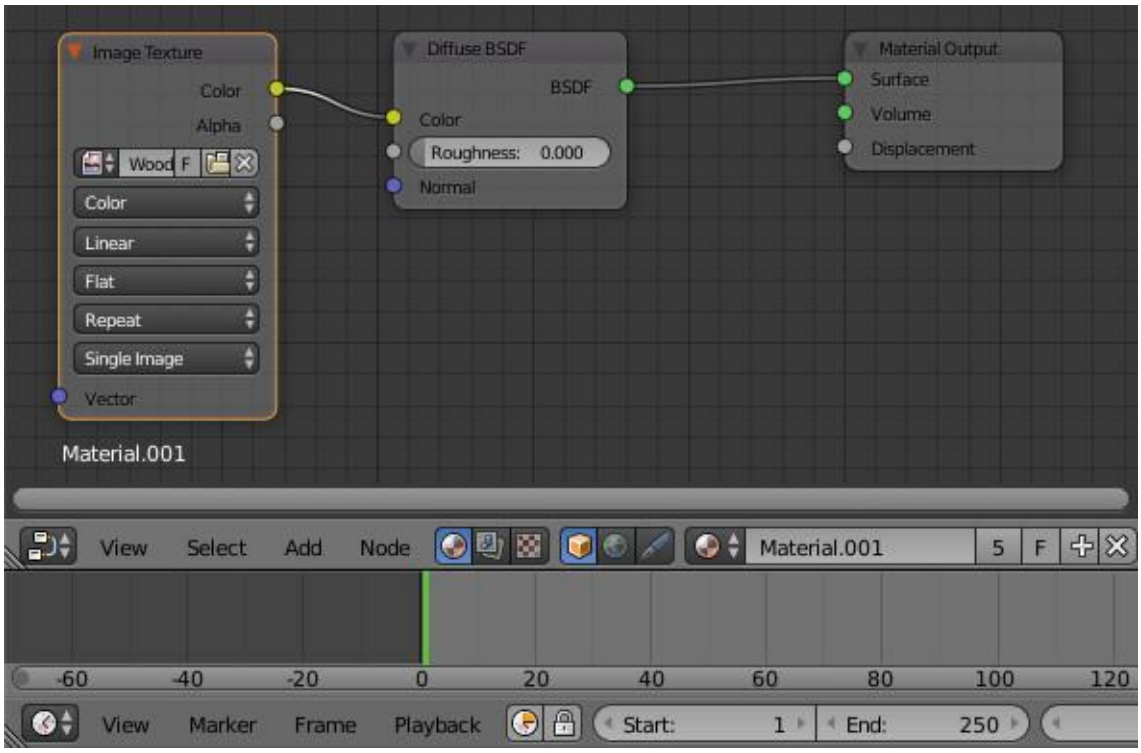
Käytännössä UV-kartan tekeminen on reunojen valitsemista ja niiden saumaksi merkitsemistä painamalla control + e-näppäin ja valitsemalla avautuvasta valikosta Mark Seam (KUVIO 10). Pieleen menneen sauman saa poistettua valitsemalla Clear Seam. Sauma tarkoittaa kirjaimellisesti saumaa tekstuurin asettelussa ja se onkin hyvä mahdollisuuksien mukaan piilottaa luonnollisiin saumakohtiin tai näkymättömiin jääviin kohtiin.




*KUVIO 10. UV-kartan tekeminen.*

Työn helpottamiseksi Blenderin tila kannattaa vaihtaa Defaultista UV Editingingiksi. Väliaikaisen materiaalin mallille saa muuttamalla näkymän tyypiksi Node Editor ja luomalla siellä uuden materiaalin. Add-valikosta lisätään Image Texture –elementti ja yhdistetään sen color-arvo Diffuse BSDF

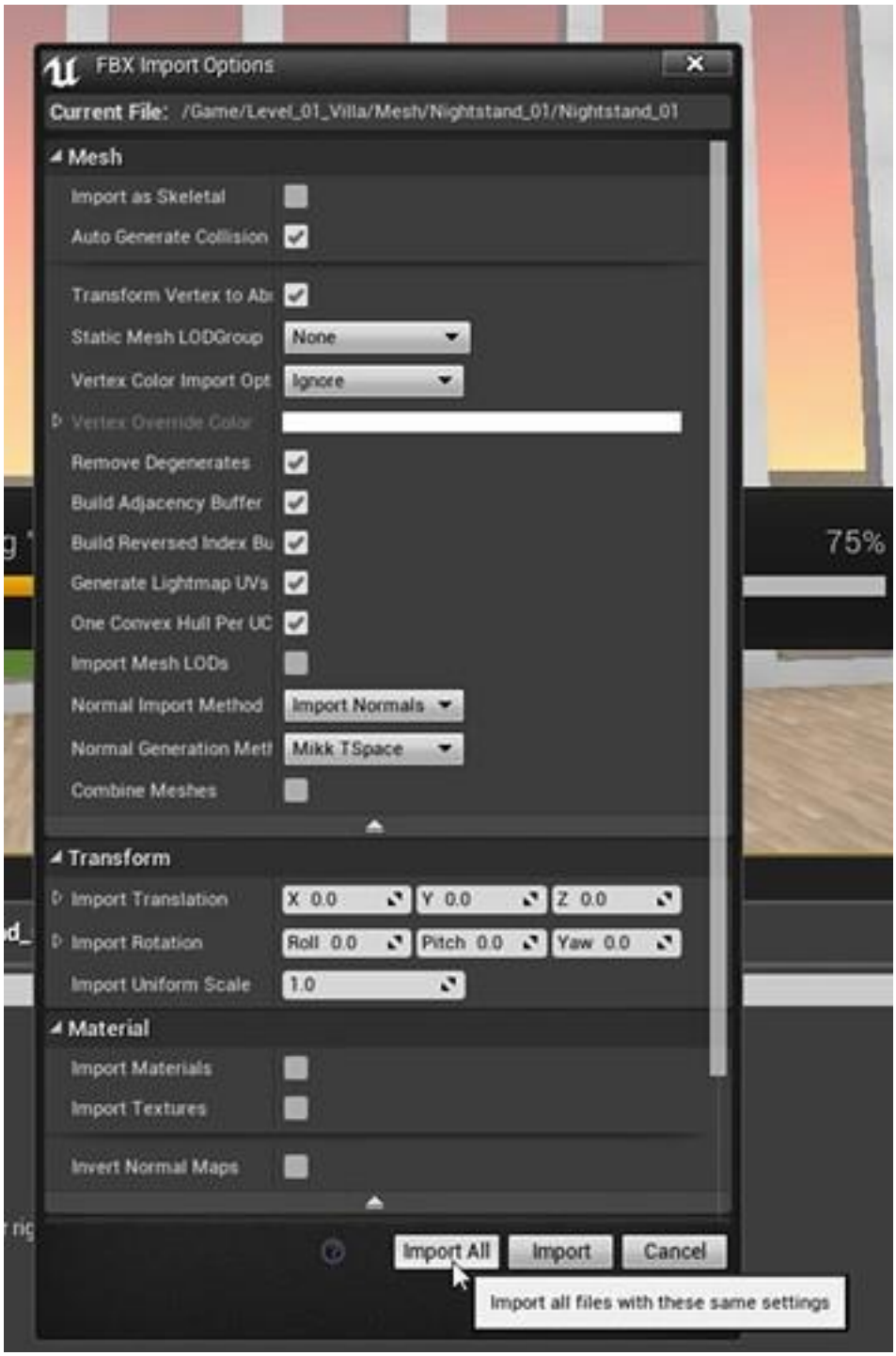
–elementin color-arvoon (KUVIO 11). Kansion kuvaa klikkaamalla Image Elementtiin voi ladata haluamansa tekstuurin. Tekstuurin ei tarvitse olla oikean kokoinen, kunhan vain siitä näkee, ettei UV-kartassa ole häiritseviä vääristymiä.




*KUVIO 11. Väliaikainen tekstuuri mallille.*

Lopuksi vielä tehdään toinen UV-kanava lightmass-karttaa varten. Helpoiten toimenpide onnistuu Data-välilehdeltä löytyvästä UV-maps-valikosta klikkaamalla plus-painiketta. Uusi UV-kartta on oletuksen kopio aiemmasta, nimeksi voi jättää Blenderin tarjoaman, mutta kahden UV-kartan tarpeen muistamiseksi voi olla hyödyllistä nimetä se Lightmassiksi. Lightmass-karttaa käytetään Unreal Enginessä valoihin liittyvän laskennan tallentamiseen ja toisin kuin tekstuurin ohjaamiseen käytetty UV-kartta, se ei missään nimessä saa sisältää päällekkäisyyksiä.

Unreal Engine ei osaa tehdä mitään blend-tiedostolla, joten malli pitää muuntaa Blenderissä fbx-tiedostoksi. Muuntaminen tehdään valitsemalla File-valikon Export-alavalikosta löytyvä fbx. Jos mallintaessa on käytetty Blenderin oletusmittakaavaa, voi skaalauksen tehdä myös fbx-tiedostoksi muuntamisen yhteydessä, muussa tapauksessa sen voi jättää oletusarvoonsa. Ennen Export FBX –painikkeen painamista tulee Geometries-välilehdeltä muuttaa Smoothingin arvoksi Face, muuten Unreal Engine huomauttaa smoothing groupin puuttumisesta.



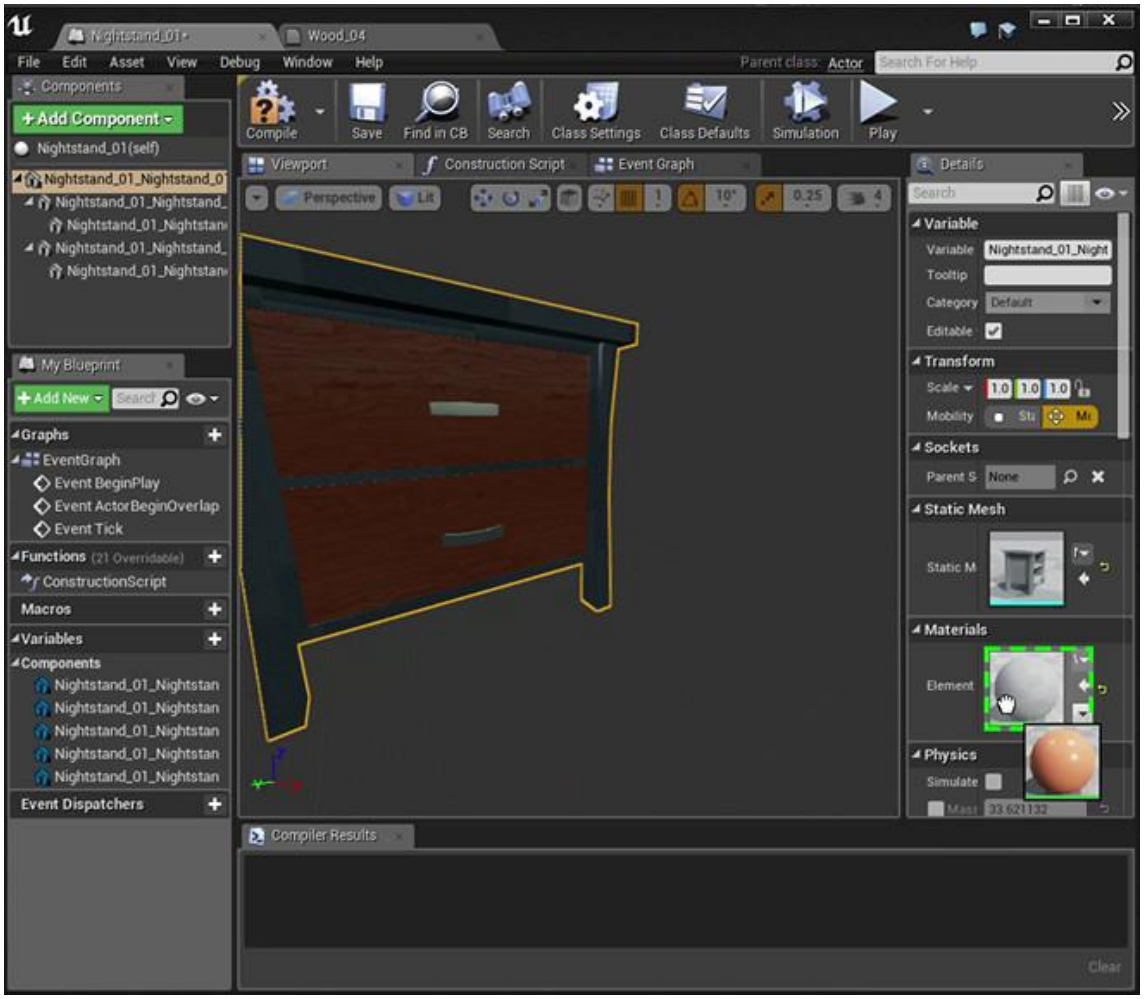

*KUVIO 12. Asetuksia Unreal Engineen tuotavalle kappaleelle.*

Fbx-tiedoston lisääminen Unreal Engine –projektiin onnistuu helposti raahaamalla se tiedostoselaimesta Unreal Engineen. Avautuvasta ikkunasta voidaan valita erilaisia asetuksia kappaleelle (KUVIO 12). Huomionarvoisia asioita ovat Auto Generate Collision ja Combine Meshes joista ensimmäinen luo törmäyksentarkistuksessa käytetyn kehikon mallille ja jälkimmäinen yhdistää kappaleet. Yksinkertaisilla asioilla törmäyksentarkistuksen voi jättää automaattiseksi, monimutkaiset

kappaleet voivat vaatia itse tehtyä kehikkoa. Jotkin kappaleet eivät välttämättä tarvitse törmäyksentarkistusta laisinkaan.

Mikäli peli-tilassa liikkuminen pysähtyy näkymättömään seinään, voi syynä olla huono törmäyksentarkistus, esimerkiksi ovenkarmille Unreal Engine luo karmin ulkomittojen mukaisen laatikon mikä johtaa siihen, että ovenkarmeista ei mahdu läpi. Tällaisessa tapauksessa pelihahmojen läpikulku voidaan sallia muuttamalla törmäyksentarkistuksen asetuksia.

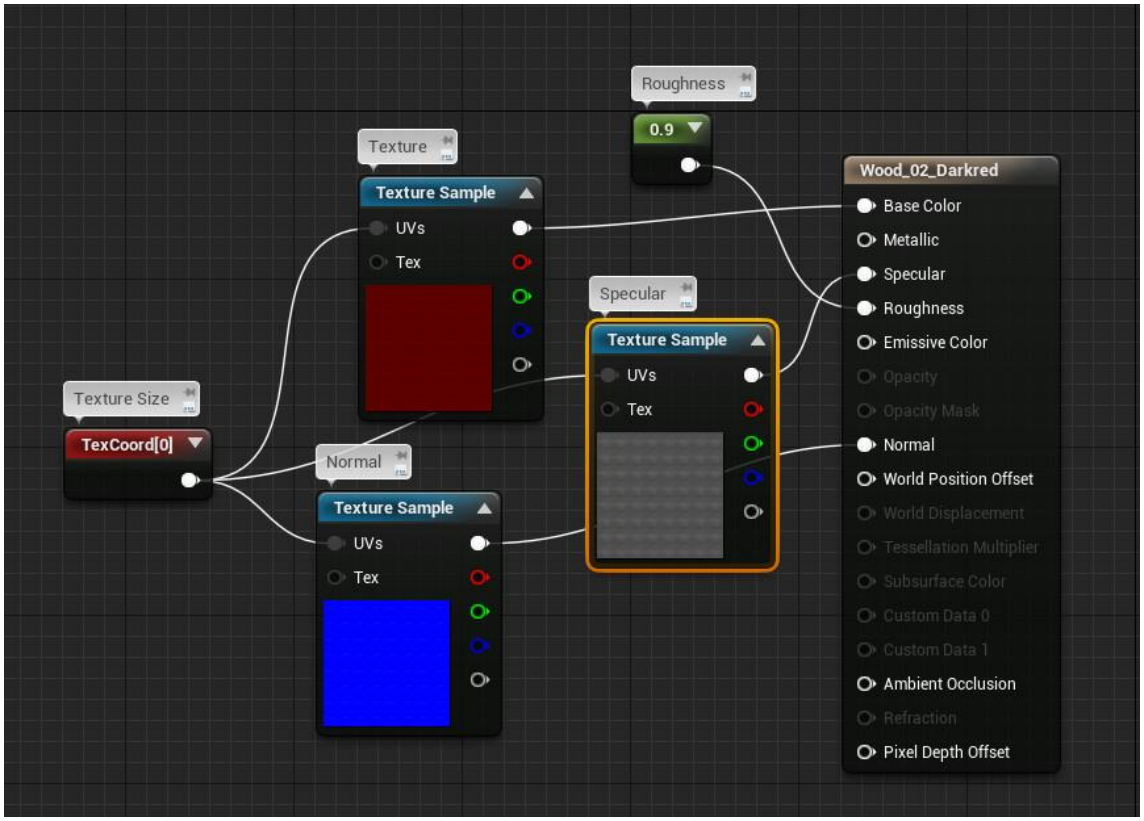
Mikäli mallissa on jotain huomautettavaa, Unreal Engine antaa virheilmoituksen. Yleisimmät syyt virheilmoituksille ovat puuttuva UV-kanava lightmass-kartalle, smoothing groupin puuttuminen ja kappaleen pieni koko, mikä voi kieliä väärästä mittakaavasta.




*KUVIO 13. 3D-mallit on liitetty blueprintiin.*

Mallia voi käyttää sellaisenaan Unreal Enginessä, mutta sille pitää asettaa materiaali joka kerta kun uusi kopio mallista lisätään pelikenttään, mutta ongelman voi kiertää käyttämällä blueprintiä. Lisätään projektiin uusi Blueprint class, minkä tyypiksi tulee Actor. Blueprint on valkoista palloa vaille tyhjä, joten avataan se muokattavaksi ja lisätään sen komponenteiksi aiemmin tuodut 3D-mallit. Pallosta pääsee eroon raahaamalla 3D-malli sen päälle komponentti-valikossa. Mallit voidaan laittaa toisten alaisiksi, milloin esimerkiksi ovenkahva perii ovelta tämän sijainnin. Materiaalit malleille voidaan asettaa Details-näkymästä (KUVIO 13).

Mallille voi asettaa jo olemassa olevan materiaalin tai luoda uuden. Uusi materiaali luodaan hiiren kakkospainiketta painamalla Unreal Enginen sisältöselaimessa. Create Basic Asset –otsikon alta valitaan Material.




*KUVIO 14. Valmis materiaali.*

Kun uuden materiaalin avaa muokattavaksi, on tarjolla vain materiaalin nimellä varustettu elementti täynnä ominaisuuksia kuten Base Color, Metallic, Specular, Roughness ja Normal (KUVIO 14). Materiaalin ominaisuuksista tärkein on Base Color, sillä se nimensä mukaisesti kuvaa materiaalin pintakuviointia. Arvoksi Base Colorille voidaan asettaa tasainen väri tai tekstuuri. Unreal Enginessä ei ole värin säilömiseen omaa muuttujaa vaan halutut rgb-arvot asetetaan 3D-vektoriin. Tekstuuri

puolestaan lisätään Texture Sample –elementin avulla. Elementistä löytyy ominaisuudet UVs ja Tex, sekä Details-välilehdestä mm. valittu kuva ja Sampler type. Jos tekstuurit on tuotu Unreal Engineen oikein, Sampler type on automaattisesti oikein eli color muilla paitsi normal mapilla.

Värin lisäksi tekstuurit tarvitaan specularille ja normalille. Specular map kuvaa kappaleen pinnan heijastavuutta ja sillä saa paljon eloa pintoihin, sillä eiväthän pinnat oikeassakaan maailmassa heijasta kaikkialta tismalleen samalla tavalla. Toinen tärkeä pinnan elävöittäjä on normal map, jolla voidaan luoda yksityiskohtia kuten kuhmuja ja naarmuja ilman tarvetta lisägeometrialle.

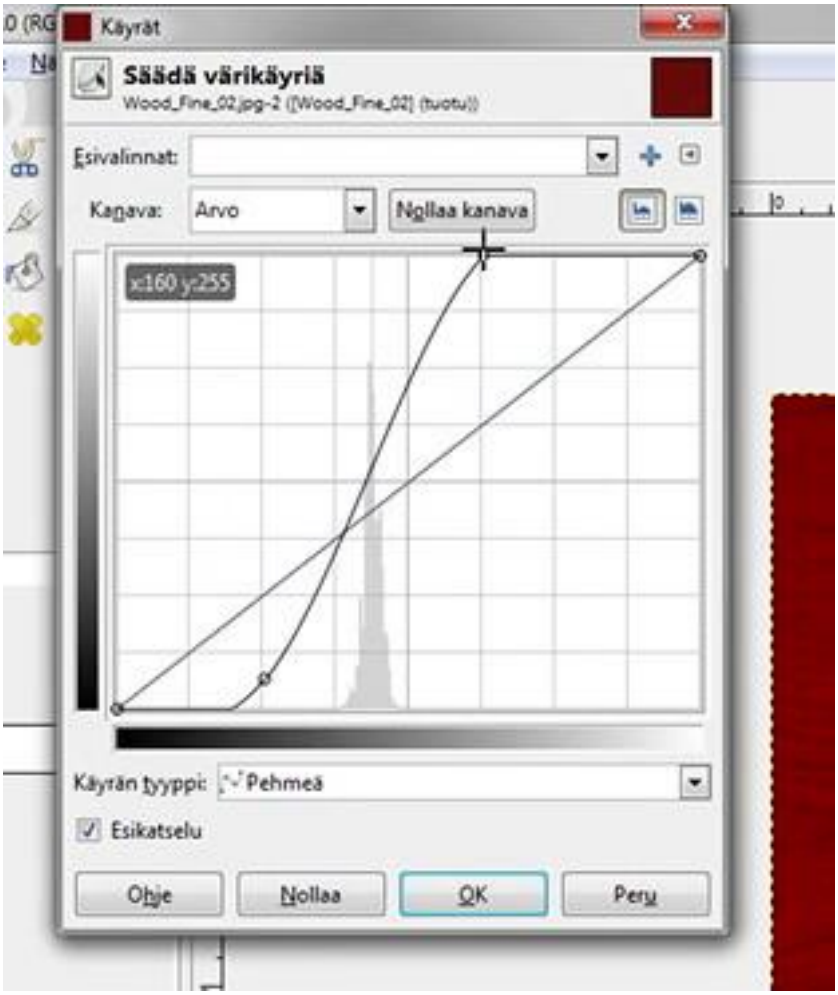
Kun kaikki kolme arvoa on asetettu, lisätään vielä roughnessiin liukuluvun sisältävä yksiulotteinen vektori säätämään pinnan kiiltävyyttä. Mitä suurempi arvo, sitä kiiltävämmäksi kappaleen pinta muuttuu.

Tekstuuri ei välttämättä ole juuri oikean kokoinen käyttötarkoitusta varten, joten sen koon muuttamiseen voi joutua käyttämään Texture Coordinates –elementtiä. Elementti yhdistetään tekstuurien UV-sisääntuloon ja details-välilehden alta löytyviä UTiling ja VTiling –arvoja muuttamalla tekstuuri saadaan laatoittumaan toivotulla tavalla.

Aina juuri sopivanlaista tekstuuria ei löydy, mutta yksinkertaisella kuvankäsittelyllä pystyy helposti muuttamaan kuvan värisävyjä, korjata virheitä ja sen sellaista. Yksinkertaisuuden vuoksi esimerkiksi Textures.comia käytettäessä kannattaa pitää silmällä seamless-avainsanaa, mikä kertoo että tekstuurista on saumattomasti laatoittuva versio saatavilla. Tämä on hyvin arvokas asia, sillä saumattomia tekstuureita hyväksi käyttäen voidaan luoda materiaaleja, joita voi käyttää hyvin erilaisissa kappaleissa sen sijaan, että jokaisella esineellä olisi oma tekstuurinsa. Kannattaa myös huomioida, että Unreal Engine suosii kuvia, joiden kanta ja korkeus ovat kahden potensseja. Kuvan resoluutiolla on väliä, sillä 1024*1024 kokoinen tekstuuri on hyvin epätarkka verrattuna 8192*8192 kokoiseen, mutta jälkimmäinen vie moninkertaisesti enemmän muistia. Sopivan koon valinta on aina kompromissi tarkkuuden ja muistinkulutuksen välillä.

Valmistakin tekstuuria voi yleensä parantaa esimerkiksi kuvankäsittelyohjelma Gimpin Käyrät-työkalulla. Käyrillä voidaan tehdä vaikka tummista väreistä hieman tummempia ja vaaleista vaaleampia ja näin korostaa tekstuurin yksityiskohtia. Tämä onnistuu muuttamalla suoraa kulmasta kulmaan osoittavasta suorasta hieman s-kirjainta muistuttava (KUVIO 15).

Valmis kuva tallennetaan Vie nimellä –toiminnon avulla, sillä Gimpissä tallennus-toiminto on varattu sen omalle xcf-tiedostoformaatille. Kuvalle valitaan formaatiksi jpg tai png, jälkimmäinen jos kuva sisältää läpinäkyvyyttä. Yksinkertaisessakin pelissä voi olla kymmeniä tekstuureita, joten on hyvä noudattaa aina samanlaista nimeämiskäytäntöä.



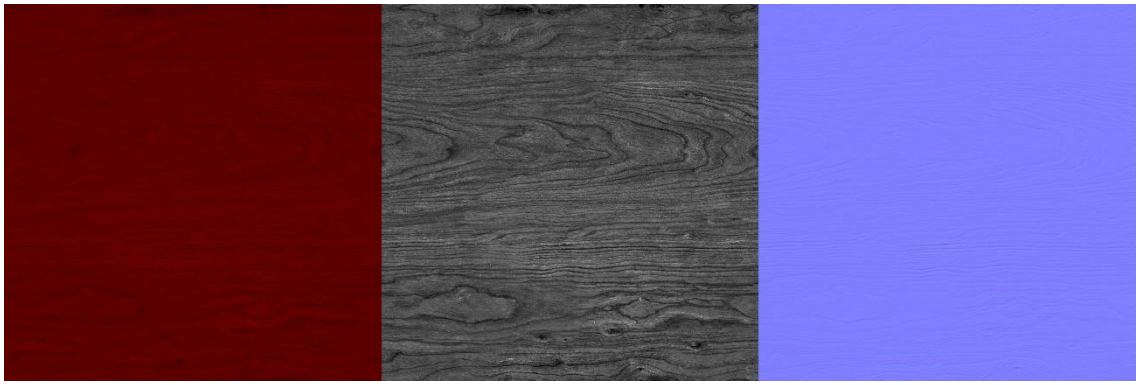

*KUVIO 15. Käyrät-työkalu Gimpissä.*

Samasta tekstuurista voidaan muokata myös kiilloista vastaava specular map -tiedosto. Kuvan tila muutetaan harmaasävyksi ja jälleen käyrien avulla tuodaan haluttuja yksityiskohtia esille. Specular map kannattaa selkeyden vuoksi tallentaa samalla nimellä kuin itse tekstuuri, mutta lisätä nimeen ”_specular”.

Normal map -tiedoston luominen ei onnistu suoraan Gimpissä, tosin lisäosilla sekin on mahdollista. Helpointa on käyttää palvelua nimeltä NormalMap-Online. Internet-selaimessa tarkastettavalle sivulle raahataan haluttu tekstuuri ja säädetään efektin voimakkuutta, korkeutta ja pehmeyttä. Voimakkuus ja korkeus vaikuttavat suoraan siihen, miltä kappaleen pinta näyttää Unreal Enginessä,

pehmeydellä voidaan vähentää tekstuurissa olevaa kohinaa ja näin saada sulavampi lopputulos. Unreal Engine etsii tekstuureista viitteitä normal map -tiedosta, joten ennen tiedostopäätettä tekstuurin tiedostonimen lopussa tulee olla ”_normal”.

Valmiit tekstuurit (KUVIO 16) voidaan importata Unreal Engineen tuttuun tapaan raahaamalla ne tiedostoselaimesta. Tässä vaiheessa Unreal Enginen pitäisi ilmoittaa normal mapin löytämisestä, jos näin on, asialle ei tarvitse tehdä muuta. Mikäli normal mappia ei löytynyt, tekstuuria tuplaklikkaamalla pääsee käsiksi sen asetuksiin, mistä Compression Settings –kohdasta valitaan arvoksi Normalmap.

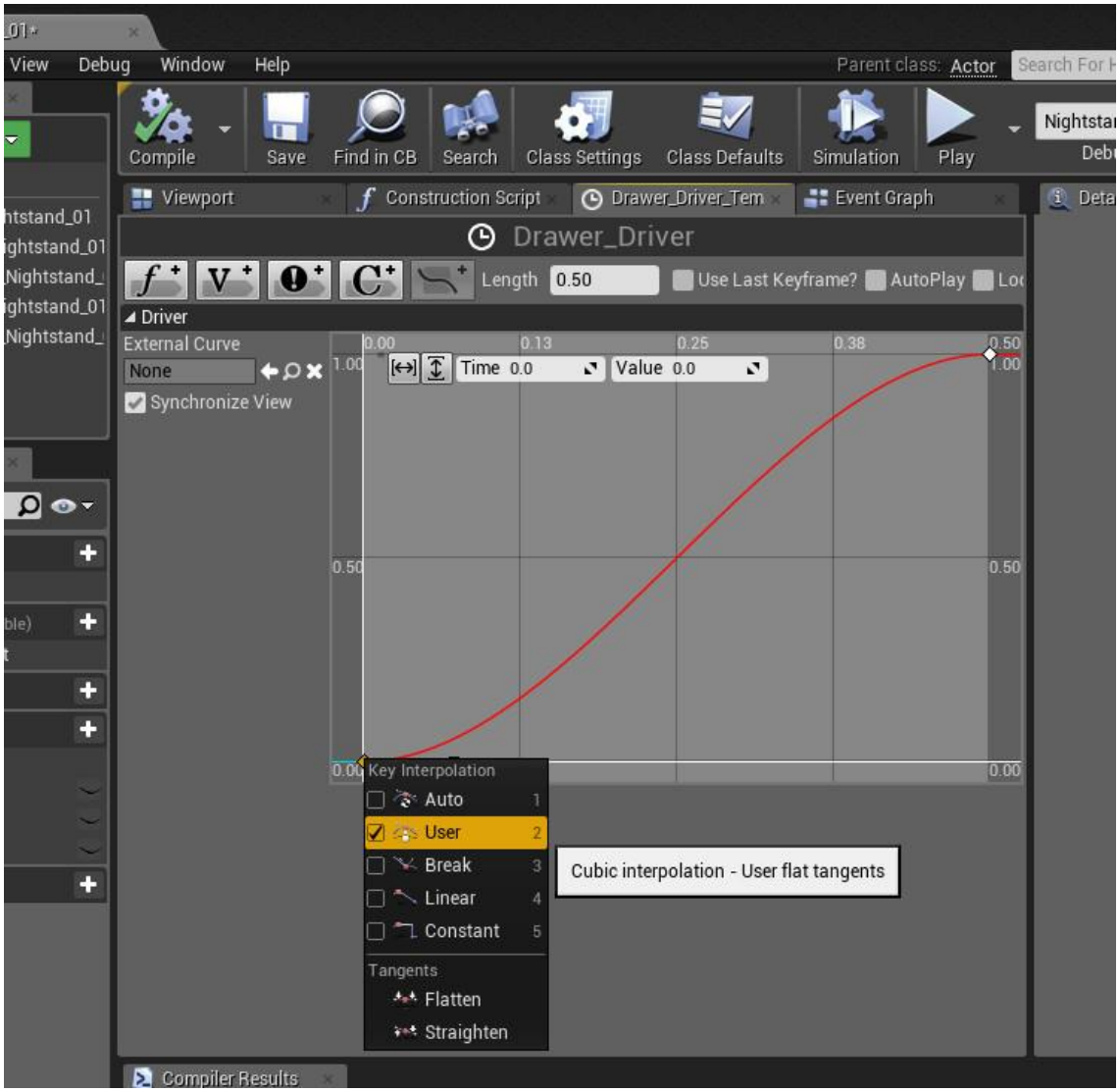


*KUVIO 16. Tekstuuri, specular map ja normal map.*

Kun kappale näyttää valmiilta, se pitää kääntää Unreal Enginessä suoritettavaksi Compile-painikkeella. Kääntämisen yhteydessä tarkistetaan, ettei blueprintissä ole virheitä. Lopulta blueprintin voi asettaa pelikenttään tarkasteltavaksi ja jos kaikki näyttää olevan kunnossa, edetään toiminnollisuuden pariin. Kannattaa huomata, että uudet kappaleet vaativat kentän valojen uudelleenlaskemisen, mikä onnistuu Build-painikkeen vieressä olevaa nuolelta klikkaamalla ja valitsemalla Build Lightning Only. Mikäli valoihin liittyvä esilaskenta ei ole ajantasainen, voi kentässä olla outoja valoja ja varjoja. Yleensä valojen laaduksi riittää esikatselu, mutta aina silloin tällöin voi olla hyvä tarkistaa, miltä maailma näyttää tarkemmilla valoilla.

Yöpöydän vetolaatikon toiminta on yksinkertaista: sen voi aukaista tai sulkea, milloin laatikko siirtyy yhden akselin suuntaisesti eteen tai taakse. Toiminnollisuuden tekeminen Blueprinttien avulla on hyvin suoraviivaista. Aluksi siirrytään tapahtumia kuvaavaan Event Graph –välilehteen ja luodaan sinne Custom Event –elementti, mikä yöpöydän kohdalla voidaan nimetä laatikon aukaisemista kuvaavaksi. Valittu elementti saadaan kahdennettua control + w-näppäimellä ja kopio nimetään

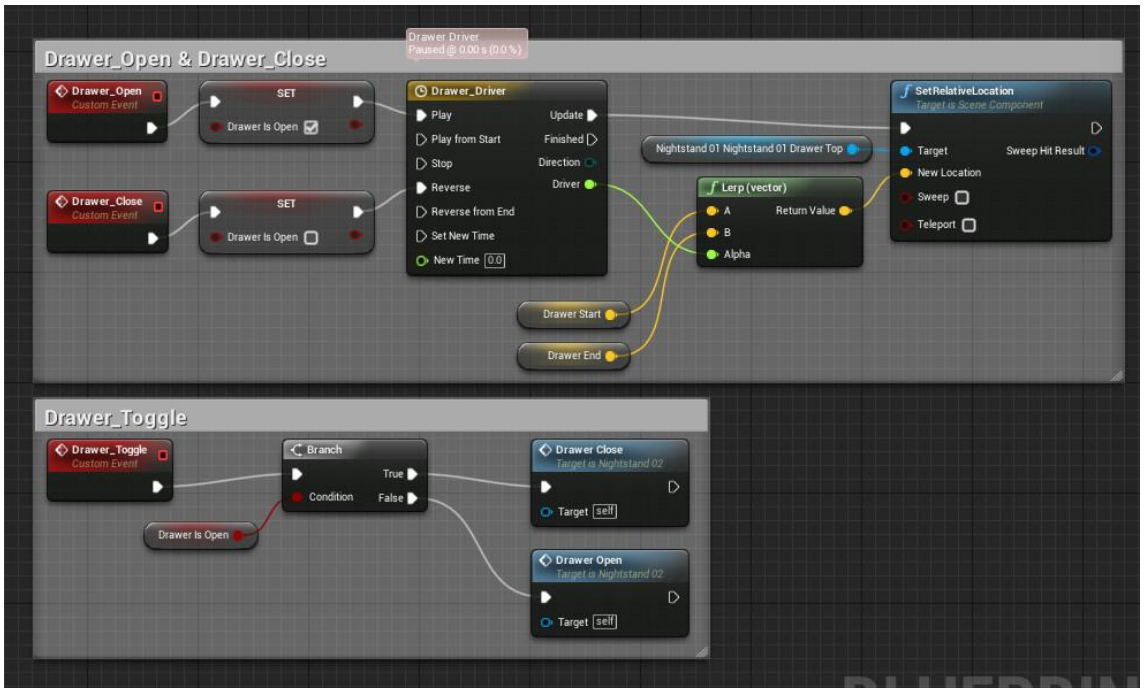
sulkemista kuvaavaksi. Seuraavaksi lisätään timeline-elementti, mitä käytetään ohjaamaan laatikon tekemää siirtymää. Timelinen saa muokattavaksi kaksoisklikkaamalla sitä. Elementille luodaan Float Track –tyyppinen käyrä klikkaamalla työkalurivillä sijaitsevaa f+-painiketta. Käyrän voi nimetä esimerkiksi Driveriksi. Aloituspiste animaatiolle asetetaan klikkaamalla hiiren kakkospainikkeella ja valitsemalla Add key to käyrän nimi. Pistettä ei tarvitse saada kerralla oikeaan kohtaan vaan aikaa ja arvoa voidaan muokata syöttämällä halutut arvot niille osoitettuihin kenttiin. Aloituspisteen ajaksi ja arvoksi tulee nolla, lopetuspisteelle ajaksi tulee liikkeen pituus ja arvoksi yksi. Liikkeen aloituksen ja lopetuksen voi pehmentää vaihtamalla Key Interpolation muotoon User (KUVIO 17). Hyvin loiva s-kirjain johtaa siihen, että liikkeen nopeus kiihtyy puoliväliin asti ja hidastuu sitten taas nollaan.




*KUVIO 17. Aikajanalla visualisoitua tietoa voidaan käyttää esimerkiksi kappaleen sijainnin muuttamiseen vaihtelevalla nopeudella.*

Lopulta palataan takaisin Event Graph –välilehdelle, missä aiemmin lisättyyn Timeline-elementtiin on ilmaantunut käyrän nimellä koristettu kenttä. Yhdistetään aukaisua varten tehty Custom Event Timelinen kohtaan Play ja sulkeminen kohtaan Reverse. Timelinen Update-kenttä yhdistetään uuteen elementtiin, tällä kertaa SetRelativeLocation. Mikäli liikuteltavaksi haluttu 3D-malli on valittuna uutta elementtiä lisätessä, sitä tarjotaan automaattisesti elementin Target-arvoksi.

Luodaan Lerp (vector) –elementti, minkä avulla voidaan kahdesta vektorista alfa-arvon avulla muodostaa uusi vektori. Alfa-arvoksi tulee Timelineen lisätty käyrä ja vektori-kentistä tehdään muuttujia klikkaamalla niitä hiiren kakkospainikkeella ja valitsemalla Promote to variable. Ensimmäinen muuttuja kertoo siirtymän aloituspisteen ja toinen päätöspisteen, joten nimetään ne asiaan kuuluvalla tavalla. Tässä vaiheessa muuttujiin ei voi vielä asettaa oletusarvoja, mutta kun blueprint on käännetty, onnistuu arvojen asettaminen Details-paneelin kautta. Lerp-elementin palauttama arvo yhdistetään SetRelativeLocation-elementin New Location kenttään ja toiminnollisuus on periaatteessa valmis.



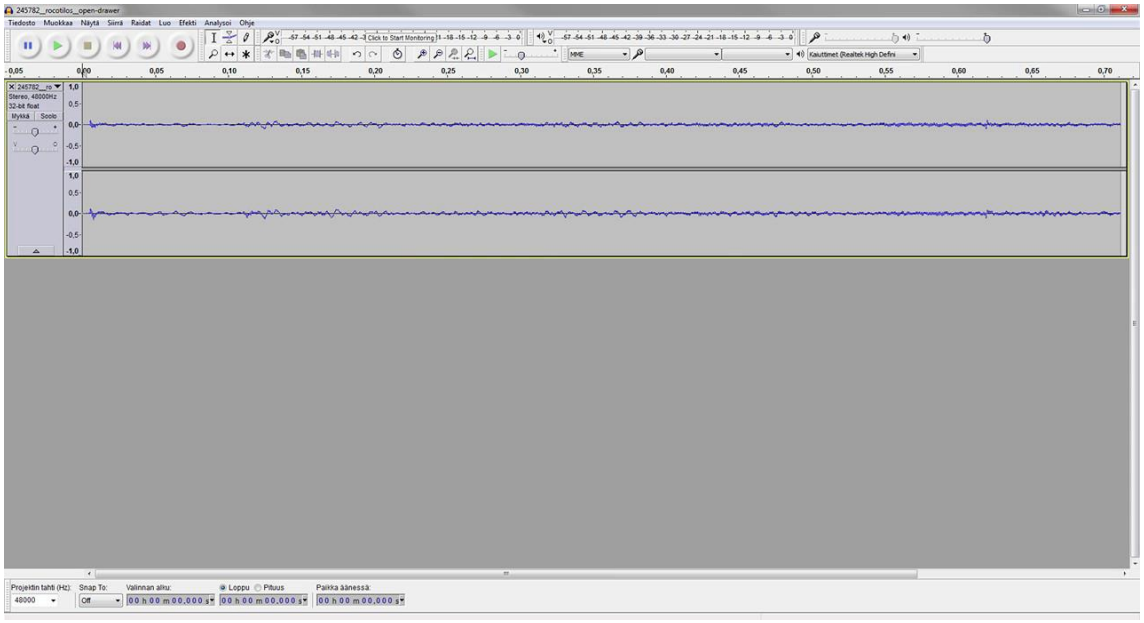

*KUVIO 18. Yöpöydän laatikon avaamiseen ja sulkeminen onnistuu hyvin yksinkertaisesti Unreal Enginen Blueprint-järjestelmällä.*

Käytännössä on kätevää tehdä kolmas Custon Event, mitä toimii kytkimenä avaamisen ja sulkemisen välillä. Luodaan totuusarvo-tyyppinen muuttuja jonka nimeksi tulee esimerkiksi is_open. Muuttujaa käytetään toiminnollisuuden haaroittavan Branch-elementin toimintaehtona. Jos ehto on totta

eli laatikko on vedetty auki, suljetaan se ja toisinpäin. Lopuksi vielä lisätään avaamisen ja sulkemisen ja Timelinen väliin Set-elementti, minkä avulla is_open-muuttujaa muutetaan vastaamaan kappaleen uutta tilaa (KUVIO 18).

Yöpöydän laatikon avaaminen ja sulkeminen tarvitsevat äänitehosteet. Sopivat, CC0-lisensoidut tallenteet löytyivät Freesound.org-sivustosta. Valtaosa laatikoiden, ovien ja sen sellaisten käyttämiseen liittyvistä äänistä on miltei täysin hiljaisia, joten sopivaa tehostetta voi joutua etsimään jonkin aikaa.

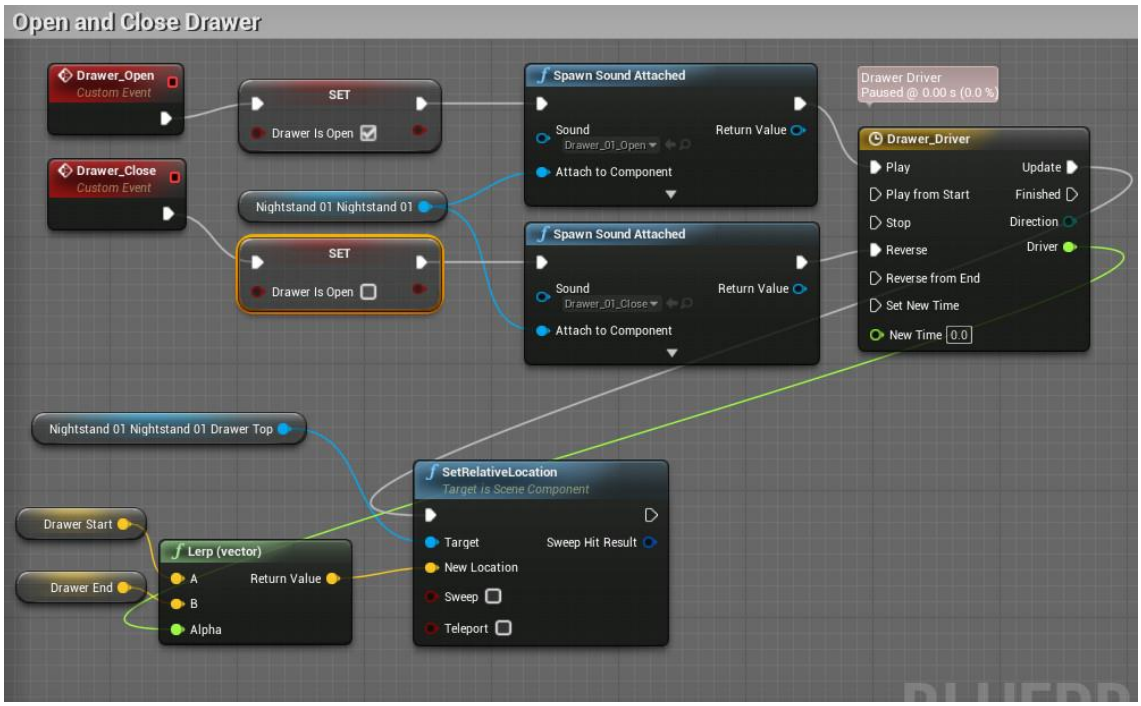
Sivustolta ladatut WAV-muotoiset äänitiedostot saadaan käsiteltäväksi raahaamalla ne Audacityyn (KUVIO 19). Valinta- ja aikakorjaustyökalulla voidaan muokata tehosteen kestoa ja poistaa turha materiaali. Efekteistä puolestaan löytyy normalisointi-työkalu, joka jo oletusasetuksilla tekee äänisignaalista huomattavasti selkeämmän. Äänen dynamiikkaa eli hiljaisten ja kovien äänten välimatkaa voidaan pienentää kompressori-efektillä. Efektillä saadaan äänestä vahvemman tuntuinen. Lopuksi vielä valitaan pätkä tehosteen alusta ja käytetään siihen efektiä häivytä sisään. Tehosteen lopulle tehdään sama, paitsi häivytä ulos –efektillä. Häivytykset muuttavat sulavasti äänisignaalin voimakkuuden nollaan, milloin vältytään ikäviltä poksahduksilta tehostetta toistettaessa.



*KUVIO 19. Äänitehoste ladattuna äänentallennus ja –käsittelyohjelma Audacityyn.*

Kun äänitehoste on valmis, se voidaan siirtää Unreal Enginelle sopivaksi Export Audio –toiminnolla jolloin tallennusmuodoksi tulee WAV. Äänitiedoston metatiedoilla ei ole merkitystä Unreal Enginen kannalta.

Kuten muutkin assetit, myös äänet voidaan raahata Unreal Engineen tiedostoselaimesta. Blueprinttiin niiden lisääminen on hyvin yksinkertaista, avataan jälleen aiemmin tehty blueprint muokattavaksi ja lisätään kaksi Spawn Sound Attached –elementtiä aikajanan eteen (KUVIO 20). Soundkenttään tulee haluttu äänitehoste ja Attach to Component –kenttään kappale, mistä ääni tulee. Elementillä on paljon muitakin asetuksia, joita voi tarkastella alareunassa olevaa nuolta klikkaamalla. Erityisesti äänenvoimakkuuden säätö on hyödyllinen ominaisuus.




*KUVIO 20. Blueprintit tulisi tehdä vasemmalta oikealle, jotta solmujen väliset suhteet näkyisivät selvemmin.*

## 4.3 Kohdattuja ongelmia

Yleisin ongelma eri oppaiden kanssa oli tiedon ajantasaisuus. Ohjelmistot kehittyvät niin kovalla vauhdilla, että jo pari vuotta vanha artikkeli voi olla täysin käyttökelvoton. Suurin osa ongelmista oli kuitenkin ratkaistavissa päättelyn avulla, sillä periaatteet yleensä säilyvät samoina. Käyttöliittymän muodostavat nappulat ja nimet voivat muuttua paljonkin ja opasta seuratessa jää monesti etsimään

sitä rastitettavaa ruutua joka ei olekaan siellä missä sen pitäisi olla. Pääsääntöisin kuitenkin etenkin kaupallisten ohjelmien omat ohjeet ovat erinomaisen kattavia ja ajantasaisia.

Välillä valoihin liittyvä laskenta vaati enemmän muistia kuin mitä käytössä olleella kahdeksan gigatavun keskusmuistilla varustetulla tietokoneella oli tarjota. Ongelman syyksi paljastui tekemäni virhe Blenderin puolella: Unreal Engine tarvitsee kaksi UV-karttaa tuotavalta kappaleelta, ensimmäistä käytetään tekstuurien asettamiseen ja toista lightmass-kartan generoimiseen.

Kansiorakenteen muutokset pitää tehdä Unreal Enginessä, sillä muutosten tekeminen tiedostoselaimen kautta rikkoo tiedostojen väliset riippuvuudet. Unreal Enginessäkään tiedoston siirtäminen ei aina tapahdu toivotulla tavalla ja yhdessä vaiheessa kaikki projektiin liittyvät tekstuurit hävisivät. Ongelman toistaminen ei onnistunut, mutta oletettavasti sillä on jotain tekemistä sivuutettujen varoitusten kanssa.

Joskus materiaalit saattoivat hohtaa vaikka niille ei minkäänlaista hohto-ominaisuutta ollut annettu (KUVIO 21). Ongelma korjaantui muuttamalla objekti liikuteltavasta paikallaanpysyväksi. Perimmäinen syy mystiselle hohtamiselle ei selvinnyt.


*KUVIO 21. Kattolampun katkaisijan mystinen hohto.*

Eräs mystisimmistä ongelmista oli kappaleiden sijainnin muuttuminen blueprintiä käsitellessä. Mitään sijaintiin tai rotaatioon liittyviä muutoksia ei tapahtunut, mutta jostain syystä esimerkiksi kaikki talon sisäovet siirtyivät kuutisen metriä alkuperäistä korkeammalle. Blueprintiä tarkastellessa ei

löytynyt mitään selitystä oudolle käytökselle, eikä se myöskään ollut väliaikainen, Unreal Enginen uudelleenkäynnistyksellä korjaantuva ongelma.

Kolmeulotteisen grafiikan teko peleihin on jatkuvaa tasapainottelua yksityiskohtaisuuden ja suurpiirteisyyden välillä. Kokemattomuus 3D-mallintamisessa näkyi liian usein yksityiskohdissa, joita on mahdoton nähdä itse pelissä. Tällaiset virheet veivät paitsi aikaa, myös tekivät pelin grafiikoista raskaammat.

Ajankäytöllisesti asiat olisivat voineet mennä huomattavasti paremmin. Ryhtiä tekemiseen alkoi löytyä oikeastaan vasta ihan loppumetreillä. Ensimmäinen virhe ajan suhteen oli suunnitteluvaiheen aikatauluttamatta jättäminen. Kun aikaa ajatustyölle tuntuu olevan paljon, aikaa myös kuluu paljon. Lisäksi työ usein jäi mielen sisälle ja vähät fyysiset suunnitelmat vanhenivat nopeasti tai olivat liian suurpiirteisiä.

# 5 YHTEENVETO

Loppujen lopuksi tavoitteessa luoda 3D-peli Unreal Enginellä ei onnistuttu. Tulos on kyllä interaktiivinen ympäristö missä kaappeja voi avata ja valoja laittaa päälle ja pois, mutta varsinainen pelillinen sisältö puuttuu. Lopputulos on kuitenkin hyvä pohja, jonka päälle voi rakentaa portfolio-kelpoisen pelin. Seuraava askel pelin kehittämisessä on talon kalustaminen, mutta sen jälkeen tuleekin interaktiivisuuden syventäminen, esimerkiksi lisäämällä lukkoja. Peli tarvitsee myös päävalikon.

Suurin ongelma oli ajanhallinta ja tuloksettomaan suunnitteluun jumiutuminen. Perinteinen liian moneen rooliin ryhtyminen ei auttanut asiaa, vaan osa pelin kannalta tärkeistä osa-alueista jäi miltei täysin vaille huomiota. Lisää huomiota olisi kaivannut esimerkiksi eri kehitysvaiheiden selkeämpi erottaminen toisistaan: suunnittelun ja ohjelmiin tutustumisen aikana tuli jo tehtyä asioita peliä varten, mutta työ piti monet kerrat aloittaa alusta kun joku asia oli jäänyt tekemättä. Erityisesti 3D-mallit vaativat monta yritystä kun liian usein lopputuloksena oli kaoottinen ja miltei mallinnuskelvoton sotku. Hyvän 3D-mallin teoriaan olisi pitänyt tutustua paremmin, jotta edes osa yrityksistä ja erehdyksistä olisi saatu vältettyä.

Moni pelikehityksen osa-alue jäi kartoittamatta, tällaisista esimerkkinä toimii pelin optimoiminen. 3D-malleja tai tekstuureja ei luotu erityinen budjetti mielessä, joten pelin suorituskyvyssä on varmasti huomattavasti parantamisen varaa.

Opinnäytetyön tekemisen myötä sain ennen kaikkea kehitettyä Unreal Engine 4:n tuntemusta ja laajennettua omaa 3D-mallinnusosaamista, erityisesti tekstuurien kanssa työskentelyn osalta. Tuntuma eri tehtävien vaatimasta ajasta parani huomattavasti, mikä on tulevaisuudessa hyvä asia aikataulujen kanssa toimiessa.

# LÄHTEET


Amazon Web Services 2016b. Details. Viitattu 26.4.2016. https://aws.amazon.com/lumberyard/details/

Amazon Web Services 2016b. Amazon Lumberyard FAQ. Viitattu 26.4.2016. https://aws.amazon.com/lumberyard/faq/.

Audacity Team. 2016. About Audacity. Viitattu 25.5.2016. http://www.audacityteam.org/about/.

Blender Foundation. 2016. About. Viitattu 19.4.2016, https://www.blender.org/about/.

Blender Foundation. 2013. History. Viitattu 25.4.2016. https://www.blender.org/foundation/history/.

Brown, M. 2015. Game Maker's Toolkit – Why Nathan Drake Doesn't Need a Compass. Viitattu 26.4.2016. https://www.youtube.com/watch?v=k70_jvVOcG0.

Crytek 2016b. CRYENGINE Showcase. Viitattu 26.4.2016. https://www.cryengine.com/showcase.

Epic Games 2015. Blueprints Visual Scripting. Viitattu 24.4.2016. https://docs.unrealengine.com/latest/INT/Engine/Blueprints/index.html.

Epic Games. 2016. What is Unreal Engine 4. Viitattu 24.4.2016. https://www.unrealengine.com/what-is-unreal-engine-4.

Freesound. 2015. Frequently Asked Questions. Viitattu 7.5.2016. https://www.freesound.org/help/faq/.

Kim, B. 2014. The 10 Best Unreal Engine 3 Games. Viitattu 24.4.2016. http://www.pcgamer.com/10-best-unreal-engine-3-games/.

MakeHuman team. 2016. MakeHuman. Viitattu 15.4.2016, http://www.makehuman.org/index.php.

Neogames Finland ry. 2016. Viitattu 28.5.2016. http://www.neogames.fi/neogames-finland-ry-peli-toimialan-raportti-2015/.

Petry, C. 2014. NormalMap-Online. Viitattu 10.5.2016. https://cpetry.github.io/NormalMap-Online/.

Pixabay. 2016. FAQ. Viitattu 7.5.2016. https://pixabay.com/en/service/faq/.

Textures.com. 2016. How It Works. Viitattu 11.5.2016. https://www.textures.com/howItWorks.

The GIMP Team. 2015. GIMP – GNU Image Manipulation Program. Viitattu 24.4.2016. https://www.gimp.org/.

The GIMP Team. 2015. How It All Started…. Viitattu 24.4.2016. https://www.gimp.org/about/pre-history.html.

Unity Technologies 2016a. Unity – Game engine, tools and multiplatform. Viitattu 20.4.2016. https://unity3d.com/unity.

Unity Technologies 2016b. Made with Unity – Games. Viitattu 20.4.2016. https://made-with.unity.com/games.

Unity Technologies 2016c. Fast Facts. Viitattu 8.5.2016. https://unity3d.com/public-relations.

Ward, J. 2008. What is a Game Engine. Viitattu 11.5.2016. http://www.gamecareerguide.com/features/529/what_is_a_game_.php.