OULUN AMMATTIKORKEAKOULU

Mikke Penttilä

VIRTUAALITODELLISUUSPELIN TOTEUTTAMINEN VISUAALISTA OHJELMOINTILIITTYMÄÄ KÄYTTÄEN

VIRTUAALITODELLISUUSPELIN TOTEUTTAMINEN VISUAALISTA OHJELMOINTILIITTYMÄÄ KÄYTTÄEN

Mikke Penttilä
Opinnäytetyö
Syksy 2018
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely, internet-palvelut ja digitaalinen media

Tekijä: Mikke Penttilä
Opinnäytetyön nimi: Virtuaalitodellisuuspelin toteuttaminen visuaalista ohjelmointiliittymää käyttäen
Työn ohjaaja: Matti Viitala
Työn valmistumislukukausi ja -vuosi: Syksy 2018 Sivumäärä: 46 + 3

Aiheeksi valikoitui virtuaalitodellisuuspelin toteuttaminen, koska pelit ovat mielenkiintoinen tapa tutkia ja oppia uutta. Projektin kautta tarkoituksena oli syventyä peliohjelmointiin ja siihen, miten virtuaalitodellisuus eroaa perinteisestä näytön kautta tapahtuvasta pelaamisesta. Virtuaalitodellisuus kehittyy, ja sillä on potentiaalia niin viihdepeleissä kuin opetuskäytössä.

Toimeksiantajaa tällä projektilla ei ollut, vaan keskityin luomaan pelidemon itsenäisesti. Laitteistona toimi Oculus Rift, pelimoottorina käytettiin Unreal Engineä ja mallinnuksessa käytettiin Blenderiä. Unreal Engine oli pelimoottori, josta itsellä ei ollut kokemusta ja jonka takia sen valitsinkin. Blender oli tuttu mallinnustyökalu, jonka opetteluun ei kulunut aikaa.

Internetistä löytyi videoita, keskustelupalstoja sekä e-kirjoja, joita pystyin hyödyntämään projektissa ja tässä opinnäytetyössä. Oppiminen tapahtui yrityksen ja erehdyksen kautta sekä esimerkkejä hyödyntäen.

Asiasanat: Unreal Engine, Blender, virtuaalitodellisuus, pelikehitys, pelisuunnittelu, Oculus Rift

ABSTRACT

Oulu University of Applied Sciences
Business information systems, internet-services and digital media

---

Author: Mikke Penttilä
Title of thesis: Implementation of a virtual reality game by using visual programming interface
Supervisor: Matti Viitala
Term and year when the thesis was submitted: Fall 2018 Number of pages: 46 + 3

The theme was chosen to be implementation of virtual reality game, as games are an interesting way to study and learn new. Through the project, the aim was to learn more about game programming and how the virtual reality differs from the traditional games. Virtual reality is developing and has potential in entertainment and teaching use.

There was no client on this project, I focused on creating a game demo independently. The hardware used was Oculus Rift, Unreal Engine was the used game engine and Blender was the tool for 3d-models. I chose Unreal Engine because I didn't have much experience about it and I wanted to learn using it. Blender was a familiar modeling tool, that I had used before.

On the internet there are videos, discussion boards and e-books that I used in my project and in this report. Learning was done through trying and discovering things and by using examples.

Keywords: Unreal Engine, Blender, Virtual reality, Game development, Game Design, Oculus Rift

SISÄLLYS

SANASTO

Actor object	Näyttelijä-objekti on mikä tahansa objekti, joka voidaan lisätä tasolle.
Blender	Avoimen lähdekoodin 3d-työkalu, joka sisältää työkalut jokaiseen 3d-mallinnuksen osa-alueeseen.
Blueprint Interface	Liittymäluokan voi lisätä erilaisille actoreille. Liittymäluokka toimii siten, että luokalle voi lisätä funktioita, syöttöjä sekä ulostuloja. Sen jälkeen itse actor-luokan sisällä, johon liittymäluokka on lisätty, voidaan funktioille lisätä toimintoja ja kutsua niitä.
Jana	Kahden verteksin välillä oleva viiva tai reuna
Oculus Rift	Virtuaalilasit ohjaimilla, kehittäjä Oculus VR
Pawn class	Pawn-luokka on perusluokka kaikille actor-objekteille, joita pelaaja tai tekoäly voi ohjata. Pawn-luokka sisältää fyysisen sisällön lisäksi kaikki toiminnallisuudet, jotka kyseinen kappale pystyy tekemään.
Polygoni	Kolmiulotteisessa grafiikassa käytettävä monikulmio, joka muodostuu vertekseistä ja niitä yhdistävistä janoista. Vähintään kolme verteksipistettä vaaditaan, että polygoni muodostuu, mutta polygonissa voi olla myös useampi verteksipiste.
Unreal Engine	Pelimoottori, jolla on pelien lisäksi mahdollista luoda animaatioita.
Verteksi	Kolmiulotteisessa grafiikassa piste, jolla on sijainti mutta ei kokoa.

1 JOHDANTO

Opinnäytetyössä tarkoituksena oli tutustua siihen, miten virtuaalitodellisuuspeli toteutetaan. Kehittämisessä tuli esiin monia osa-alueita: millä pelimoottorilla pelin tekee, kuinka pelimoottori tukee virtuaalitodellisuutta, miten toteuttaminen eroaa perinteiseen peliin verrattuna ja mitä mahdollisuuksia virtuaalitodellisuudella on. Näihin kysymyksiin oli tarkoitus löytää vastaus.

Pelisuunnitelma on myös oltava, jotta tiedetään, minkälainen pelistä tulee. Tässä opinnäytteessä osa-alueina ovat ohjelmointi ja mallinnus, mutta selkeyden vuoksi käydään läpi myös pelisuunnitelma. Suunnitelma käydään lyhyesti läpi, ja kokonaisuudessaan pelisuunnitelma löytyy liitteistä (Liite 1). Tarkoituksena oli kehittää idea, joka on suhteellisen uniikki ja käyttää hyödykseen virtuaalitodellisuuden tarjoamia mahdollisuuksia. Omien kokemuksien pohjalta virtuaalitodellisuus tarjoaa uudenlaisen lähtökohdan peleille. Hyvinkin tavalliset tai tylsänä pidetyt asiat perinteisissä peleissä voivat olla monimutkaisempia ja haastavampia virtuaalitodellisuudessa. Esimerkkinä mainittakoon tähtäystä vaativat toimenpiteet: tietokone- tai konsolipeleissä tähtäys tapahtuu helposti hiiren avulla, mutta virtuaalilaseilla tähtääminen vaatii todellista silmä- ja kädenkoordinaatiota.

Pelimoottoriksi valikoitui Unreal Engine 4. Vaihtoehtona olisi ollut myös Unity. Päädyin valintaan siksi, että Unreal oli itselleni suhteellisen tuntematon ennen opinnäytettä. Halusin oppia Unrealin käytön, sillä se on työkalu, jota käytetään todella monessa muussa pelissä: Fortnite, Eve: Valkyrie ja Street Fighter V ovat esimerkkejä sellaisista peleistä (Wikipedia List Of Unreal Engine Games 2018 b, viitattu 19.9.2018).

Ohjelmointi on osa-alue, johon kuluu eniten aikaa. Tärkeyden ja henkilökohtaisen mielenkiinnon vuoksi opinnäytetyössä käydään läpi myös mallinnusta 3d-työkalu Blenderillä. Blender on jo ennestään tuttu 3d-työkalu, joten sen opetteluun ei kulunut ylimääräistä aikaa. Tarkoituksena oli mallintaa omia malleja, kuten esimerkiksi rakennuksia ja työkaluja, mutta mahdollisuuksien mukaan hyödyntää myös valmiita malleja.

Tavoitteena oli saada toteutettua virtuaalilaseilla pelattava demo käyttäen hyväksi Unreal Engineä ja Blenderiä. Laitteistona toimi Oculus Rift. Demoa suunnitellessa oli oltava tietyt minimivaatimukset toimintojen suhteen, ja tarkoituksena oli, että demoa voisi käyttää hyväksi portfoliossa.

Demon lisäksi tavoitteena oli sen pohjalta luoda myös tämä raportti, joka kertoo erilaisista projektin vaiheista ja haasteista. Raportin oli oltava sellainen, jonka pohjalta samasta aiheesta kiinnostunut pystyisi oppimaan uutta. Raportissa olisi oltava myös pohdintaa ja vertailua erilaisista aiheista. Tässä opinnäytteessä esiintyvät koodinäytteet eivät välttämättä vastaa ihanteellista koodia. Koodi on kuitenkin toimivaa, ja se on tärkeintä.

Henkilökohtaisena tavoitteena oli kehittyä ohjelmoinnissa ja tutustua Unreal Engineen siten, että sen parissa olisi mahdollista myös työskennellä tulevaisuudessa. Virtuaalitodellisuus tarjoaa haasteita ja on käytettynä teknologiana vielä suhteellisen uutta. Opetellessa kehittämistä tätä kautta saa hyödyllistä ja uniikkia taitoa tulevaisuuden varalle.

Projekti alkoi pelisuunnitelmasta. Kun sopiva pelisuunnitelma oli luotu, alkoi Unreal Engineen tutustuminen. Tarkoitus oli tutustua pelimoottoriin yrityksen ja erehdyksen kautta. Internet tarjoaa monenlaisia lähteitä pelimoottorin opetteluun.

Mielenkiinnon ja vaihtelevuuden vuoksi ohjelmointi, mallinnus ja raportin kirjoittaminen tapahtuivat vuorotellen. Ohjelmointiin liittyen tarkoituksena oli suunnitella ja luoda tietty minimivaatimus toimintojen suhteen. Mallinnuksen näkökulmasta tarkoituksena oli luoda ja löytää sellaisia malleja, jotka sopivat teemaan ja ovat mielenkiintoisia sekä optimaalisia peleihin.

Ajanjaksona toimi syksy 2018. Jos tällä ajanjaksolla olisi keskittynyt pelkästään peliin, olisi se voinut riittää jopa pieneen toimivaan peliin. Samalla oli kuitenkin kirjoitettava raporttia, eivätkä kaikki osa-alueet olleet vielä tuttuja, joten tavoitteeksi syntyi saada aikaan pelattava pelidemo, johon raportissa voisi syventyä.

2 OHJELMISTOT JA LAITTEET

Projektissa käytettiin pelimoottorina Unreal Engine 4:ää. 3d-mallinnuksessa käytettiin Blenderiä. Virtuaalitodellisuuslaitteistona toimi Oculus Rift Touch-ohjaimilla.

2.1 Unreal Engine

Unreal Engine 4 on täydellinen kokonaisuus pelinkehitystyökaluja pelintekijöille niin kaupalliseen kuin yksityiseenkin käyttöön. Se soveltuu kaikenkokoisille tiimeille, ja työkaluja löytyy niin ohjelmointiin kuin animointiinkin. Unreal Engine on merkittävä pelimoottori myös siinä mielessä, että sen hankkiminen ei maksa mitään. (Unreal engine 4.x example 2016, viitattu 27.9.2018.) Siinä vaiheessa, kun peli on valmis myyntiin, Epic Games ottaa 5 prosentin osan itselleen myyntituloista (Unreal Engine 2018 a, viitattu 27.9.2018).

Unreal Engine on suosittu pelimoottori, ja sillä on tehty myös useita AAA-luokan pelejä. Tämän lisäksi Unreal Engine on saanut lähes 50 palkintoa vuodesta 2004 lähtien, muiden muassa 2018 Best Game Engine (Unreal Engine Awards 2018 b, viitattu 20.11.2018).

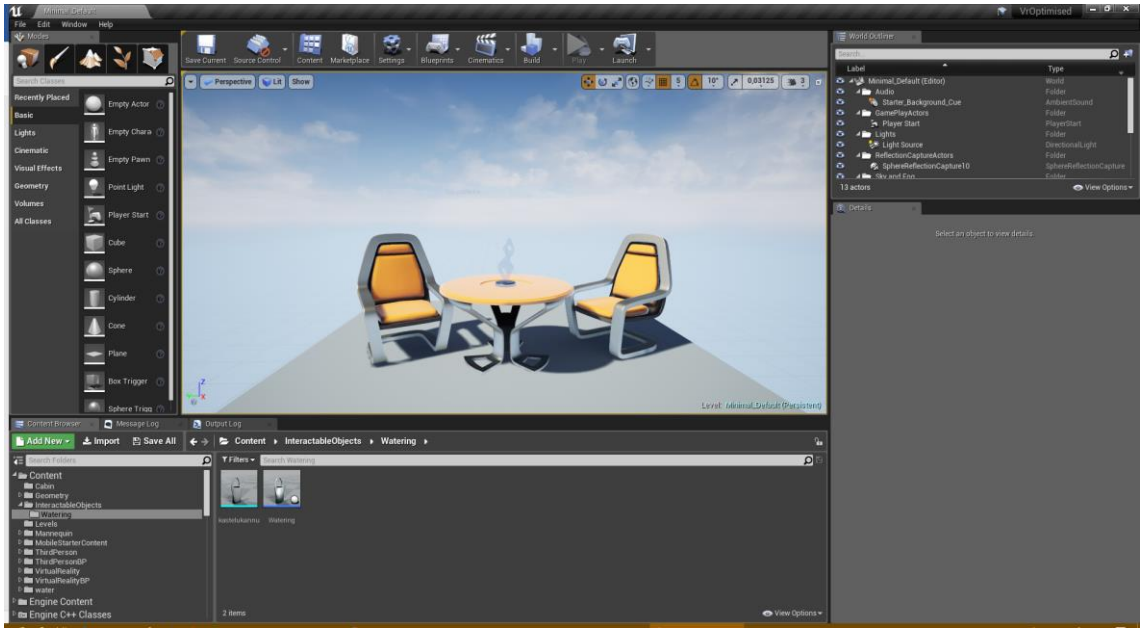
2.1.1 Ohjelmointi

Unreal Engine tarjoaa ohjelmointiin kahta erilaista työkalua. C++-kieltä käyttäen voi hyödyntää pelimoottorin omia ohjelmointielementtejä, ja lopuksi koodin koontivaiheessa tehdyt muutokset saa välittömästi näkymään pelimoottorin sisällä. Koodia voi muokata joko Visual Studiota tai XCodea käyttämällä. Toinen työkalu ohjelmointiin on visuaalinen ohjelmointiliittymä: tämä työkalu toimii siten, että käyttäjä yhdistelee erilaisia toimintolohkoja ja viitteitä toisiinsa visuaalisten johtojen avulla. C++-luokista voi tehdä Blueprint-luokkia, ja näin voi helpottaa tiimin sisäisiä tehtäviä. (Unreal Engine Programming 2018 d, viitattu 27.9.2018.)

Demossa käytettiin ohjelmointiin visuaalista ohjelmointia, eli ohjelmointi tapahtui käyttämällä Blueprint-luokkia. Visuaaliset ohjelmointiliittymät ovat yleistymään päin myös muissa sovelluksissa, joten on hyvä ymmärtää, miten ne eroavat perinteisestä kirjoitetusta koodista.

2.1.2 3D-ympäristö

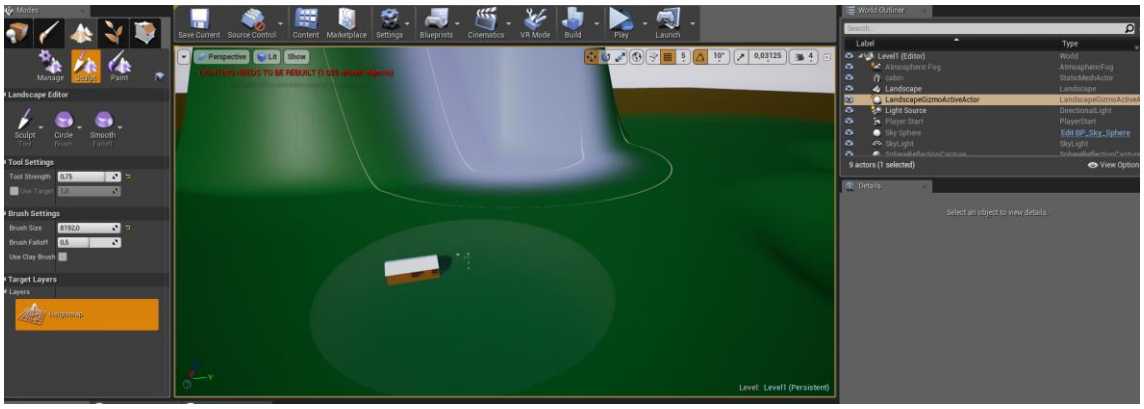
Unreal Engine sisältää 3d-ympäristön muiden pelimoottoreiden tapaan. Kun projekti avataan, ruudulla näkyy isoimpana elementtinä nykyisen tason näkymä (KUVIO 1). Näkymää voi siirtää tarpeen mukaan, ja perspektiivin voi vaihtaa ortografiseen näkymään. Samalla voi saada näkyviin myös neljä näkymää jokaisesta suunnasta. Käyttöliittymä on muokattavissa todella vapaasti, jolloin jokaiseen osa-alueeseen saa halutunlaisen näkymän tarpeellisine elementteineen.



KUVIO 1. Unreal Enginen käyttöliittymä

3d-mallien lisääminen pelimoottoriin tapahtuu FBX-muodossa. FBX-formaatti on yksi suosituimmista pelinkehityksessä käytetyistä muodoista, sillä se voi mallin lisäksi siirtää myös materiaaliasetukset ja animaatioita. Kaiken lisäksi formaattia tuetaan kaikissa suosituimmissa 3d-sovelluksissa. FBX-muoto on Autodeskin oma tiedostoformaatti. (3d game design with ue4 and blender 2016, viitattu 27.9.2018). Jos 3d-malli ei ole FBX-formaatissa, onnistuu sen konvertointi käyttämällä FBX-konvertteria tai 3d-sovelluksessa formaattia vaihtamalla.

Ympäristön lisäämiseen ja muokkaamiseen on pelimoottorissa omat työkalut. Lisäämällä tasolle oman Landscape-objektin on mahdollista muokata maastoa muun muassa korkeuden ja materiaalin suhteen. Maata voi nostaa ja laskea eri muotojen mukaan, ja muotoja voi lisätä myös itse. Tarkempaa työtä varten Unreal tukee myös kustomoituja korkeuskuvia ja -tasoja. Ympäristöön on mahdollista lisätä kasvillisuutta ja kustomoituja tekstuureita suoraan tasolle piirtämällä (KUVIO 2).



KUVIO 2. Ympäristön muokkaaminen Unreal Enginessä

2.1.3 Virtuaalitodellisuus

Unreal Engine 4 tukee virtuaalitodellisuutta tarjoamalla siihen valmiit työkalut. Koska Unreal Engine on suunniteltu vaativillekin sovelluksille, kuten AAA-luokan peleille ja visuaalisesti vaativille animaatioille, tukee pelimoottori myös virtuaalitodellisuutta niin tietokoneelle, konsolille kuin mobiilillekin. Tasosuunnittelijoille Unreal mahdollistaa tasojen editoinnin virtuaalilasien avulla, jolloin on mahdollista rakentaa mahdollisimman todentuntuinen maailma, joka vastaa oikeaa maailmaa muun muassa skaalan suhteen. Unreal tarjoaa ohjelmointiin kattavat seurantatyökalut suorituskykyyn liittyen sekä valmiita toimintoja ja funktioita lasien ja ohjainten hyödyntämiseen. (Unreal Engine 2018 a, viitattu 27.9.2018.)

Ohjelmointiin liittyen toimintolohkoja löytyy myös virtuaalitodellisuuteen liittyen. Toimintojen avulla kehittäjä pystyy saamaan erilaista tietoa lasien ja ohjaimien sijainnista ja lisätä toimintoja esimerkiksi ohjaimien eri painikkeisiin. Projekteja pystyy myös testaamaan VR-laseilla ilman erillistä odottelua.

2.1.4 Lisäosat

Epic Games -sovelluksen sisältä löytää muun muassa Unreal Enginen oman kauppapaikan. Kauppapaikasta löytää erilaisia työkaluja ja valmista sisältöä esimerkiksi 3d-mallien ja toimintojen muodossa. Hinnat vaihtelevat sadoista dollareista ilmaiseen, mutta pääasiassa suurin osa sisällöstä on maksullista. Epic Games julkaisee omissa demoissaan käytettyä sisältöä ilmaiseksi, joten sieltä on mahdollista löytää myös hyvin korkeatasoista sisältöä ilmaiseksi.

Virtuaalitodellisuuteen liittyvä sisältö on vähäistä. Hakusanalla ”virtual” löytyi kauppapaikasta noin 30 erilaista tuotetta, ja niiden hinnat vaihtelivat ainoastaan yhden ollessa ilmainen. Suurin osa liittyi erilaisiin ohjelmoituihin toimintoihin.

2.2 Blender

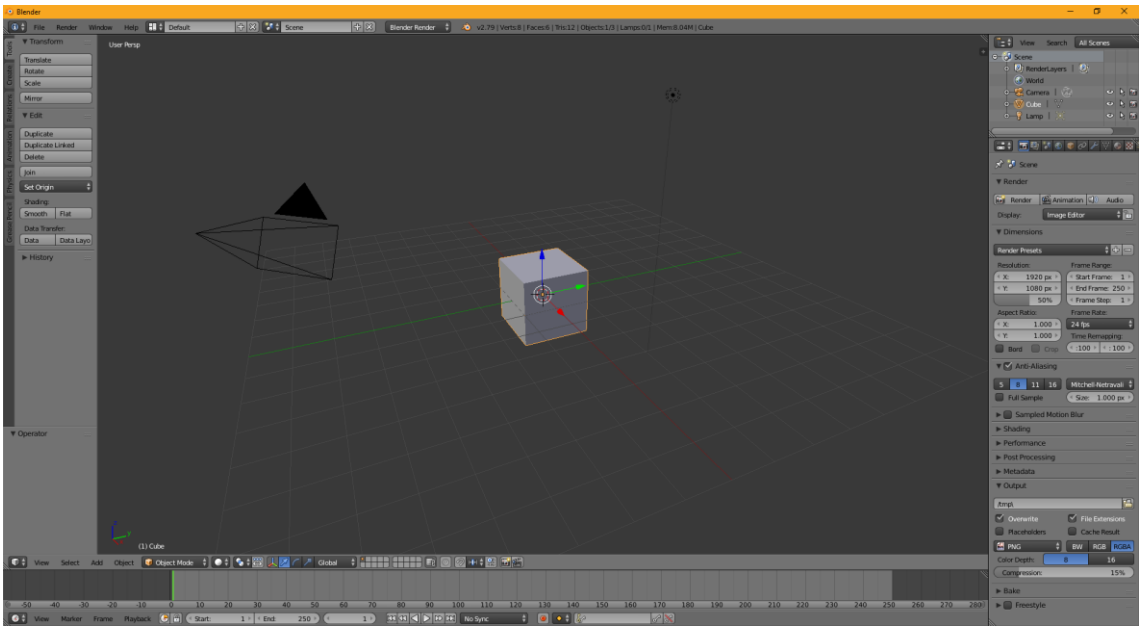
Blender on ilmainen ja avoimeen lähdekoodiin perustuva 3d-työkalu. Se tukee mallinnusta, simulaatioita ja animaatioita. Animaatioihin voi käyttää liikkeenseurantaa. Blender tukee myös video- ja pelieditointia. Myös ohjelmointi Pythonilla on mahdollista, eli käyttäjä voi luoda omia ominaisuuksia ja työkaluja suoraan 3d-työkaluun. Blender soveltuu loistavasti yksityisille tai pienille studioille, sillä se sisältää kaiken tarvittavan samassa ja toimii Windows-, Linux- ja Macintosh-käyttöjärjestelmillä. (Blender About 2018 a, viitattu 7.10.2018.)

Blender tukee monia tiedostomuotoja 3d-mallien vientiin. Blenderin oma tiedostomuoto on Blend, jolla hallitaan projekteja. Tätä muotoa tukevat muutamat muutkin ohjelmat, kuten esimerkiksi pelimoottori Unity. Muita tuettuja tiedostomuotoja ovat muun muassa FBX, 3DS, DFX ja Alembic. (Wikipedia Blender 2018 a, viitattu 7.10.2018.)

2.2.1 Mallintaminen

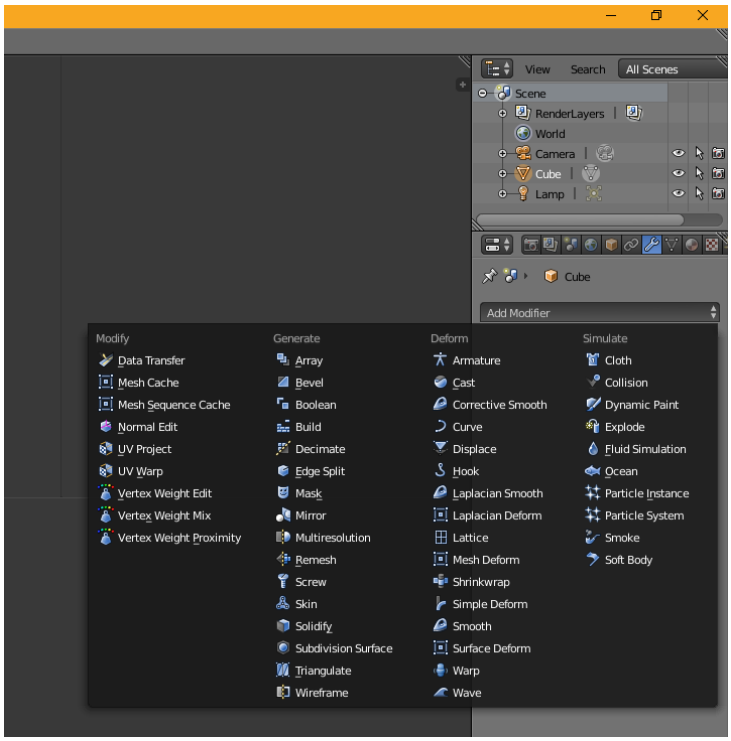
Kun mallintaa peleihin 3d-malleja, täytyy pitää mielessä tietynlainen raja polygonien määrässä. Elokuvien 3d-malleissa ei ole rajaa verteksien määrässä; ainoa rajoite on aika. Peleissä ajan lisäksi rajoittaa puhelimen, konsolin tai tietokoneen suorituskyky. Mitä enemmän on verteksejä, sitä enemmän tehoa vaaditaan laitteistolta. Kun teknologia kehittyy, verteksien määrä kasvaa. Esimerkkinä voidaan käyttää Playstation-pelikonsoleja. Playstation 1:n aikaan pelihahmo sisälsi noin 1 200 polygonia, mutta Playstation 3:n aikana hahmon polygonimäärä oli lisääntynyt jo 12 000 polygoniin. Nykyisin Playstation 4 -peleissä polygonimäärä yksittäisellä hahmolla voi olla jopa 120 000 (Pluralsight 2014, viitattu 7.10.2018.) 1994–2013 välissä polygonien määrä on siis satakertaistunut.

Blenderillä mallintaminen tapahtuu pääosin näyttämössä (KUVIO 3). Oletusnäkymä sisältää näkymän nykyiseen näyttämöön, asetusvalikot, näyttämön objektien sisällysluettelon ja työkalut valittuna olevan objektin muokkaukseen. Näyttämön näkymää voi siirtää hiirtä tai pikanäppäimiä käyttäen, ja lisäksi mahdollista on myös muuttaa näkymän perspektiiviasetuksia.



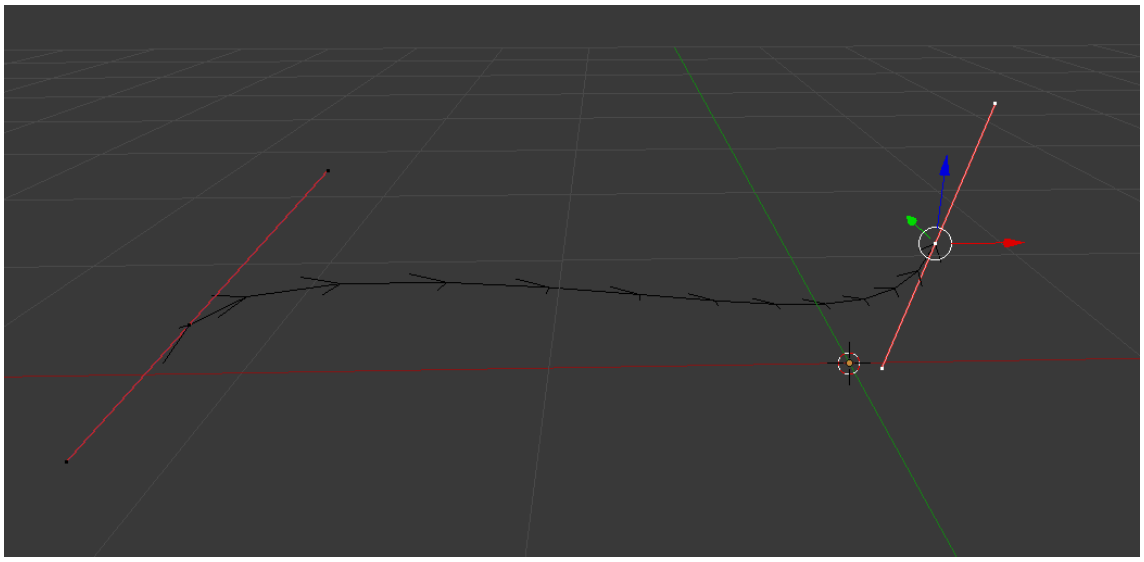
KUVIO 3. Blenderin käyttöliittymä

Mallintaminen tapahtuu Edit-tilassa. Tilaan pääsee, kun mallinnettava objekti on valittuna ja painetaan Tab-näppäintä. Tällöin päästään Edit-tilaan, jossa käyttäjä voi muokata ja siirtää valitun objektin verteksipisteitä sekä luoda uusia. Manuaalisen verteksipisteiden muokkauksen lisäksi Blender tarjoaa monenlaisia työkaluja työn nopeuttamiseksi. Työkalujen avulla voidaan esimerkiksi tehdä kopioita valitusta objektista eli käyttää työkalua Array, leikata objektia toisen objektin avulla eli käyttää työkalua Boolean ja muokata objektin muotoa tietynlaisen ristikkolaatikon eli työkalu Latticen avulla (KUVIO 4).



KUVIO 4. Edit-tilan työkaluja

Verteksien lisäksi 3d-malleja on mahdollista tehdä polku-, bezier- ja nurbs-kurvien avulla. Kurvit ovat joko 2d- tai 3d-viivoja, joita pystyy taivuttamaan ja muokkaamaan halutun muotoisiksi. (KUVIO 5). Niiden avulla on mahdollista tehdä polkuja, joita pitkin voi laittaa kulkemaan esimerkiksi putkia tai tehdä kurvin muodon mukaisia 3d-malleja ruuvityökalulla. Ruuvi eli screw tekee pyöreän muodon noudattaen kurvin muotoa, jolloin on mahdollista tehdä esimerkiksi juomalasi.



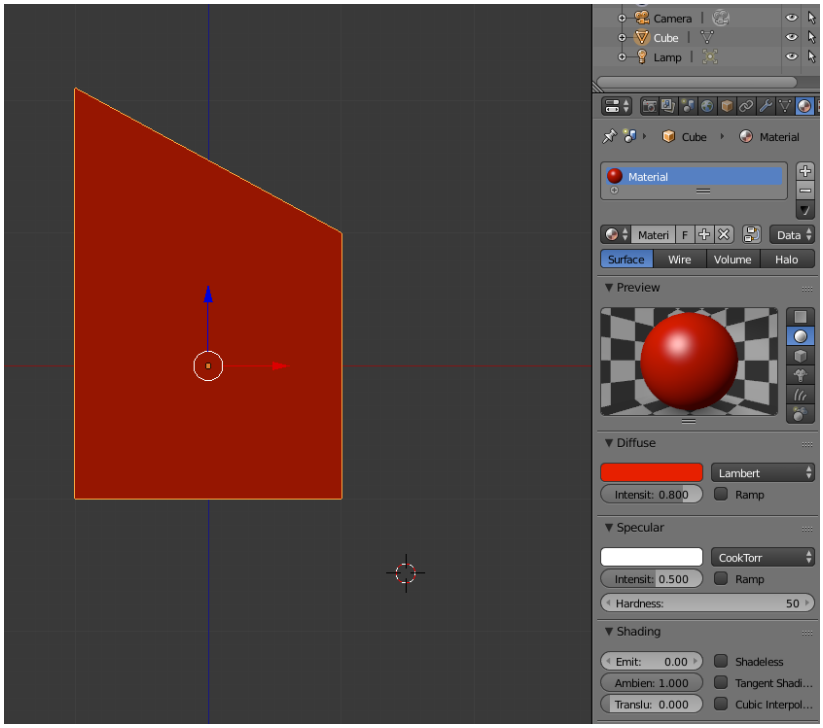
KUVIO 5. Bezier-kurvi

Mallintamista helpottavat myös muut työkalut. Smooth-työkalun avulla saadaan matalapolygoninen malli näyttämään sulavammalta, mikä on erittäin hyödyllistä etenkin peleihin malleja mallintaessa, kun verteksimäärä on rajallinen. Vertekseihin liittyen Blender sisältää myös työkalut verteksien päällekkäisten kopioiden poistamiseksi.

Jos malli alkaa olla liian yksityiskohtainen, voi eri osia piilottaa näkyvistä sekä siirtää ja muokata objektin haluttua osaa eri tasolla. Tämän lisäksi on erilaisia työkaluja muun muassa alkupisteen säätöön, polkujen seurantaan ja eri objektien yhteen liittämiseen.

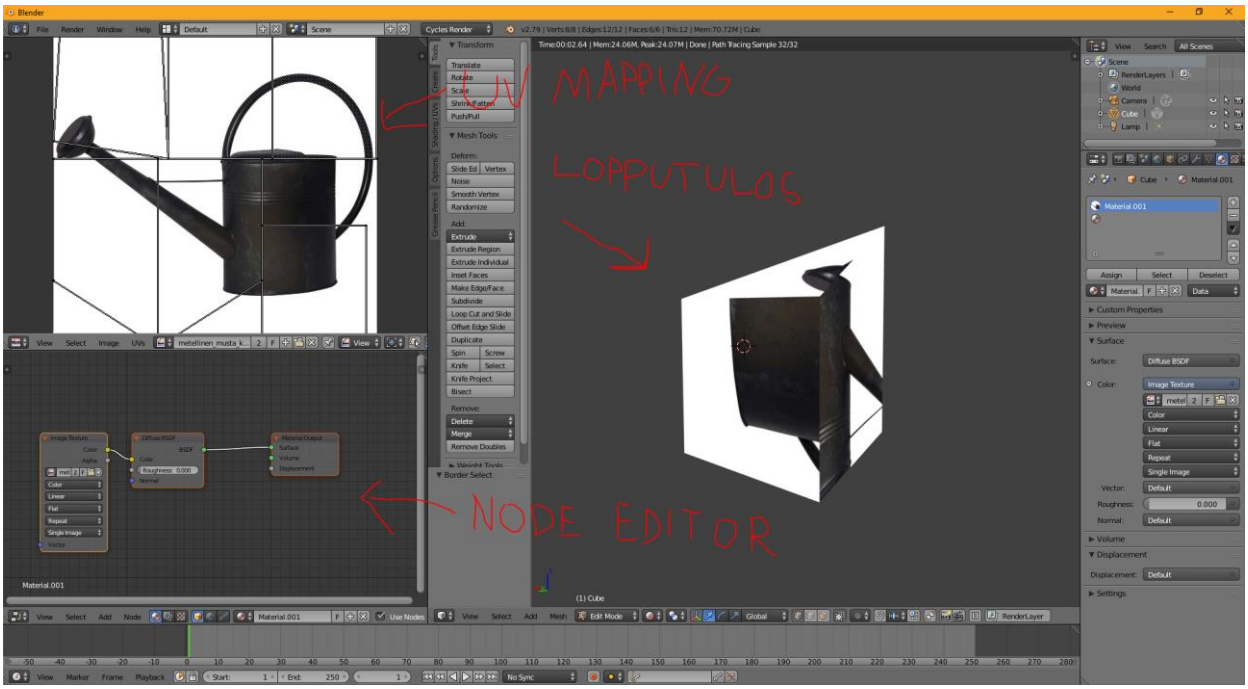
2.2.2 Teksturointi ja pinnat

Teksturointia ja materiaaleja on mahdollista tehdä Blenderillä usealla tavalla. Tavallisesti aloittelija käyttää materiaalit-välilehdestä löytyviä asetuksia yksinkertaisten värien ja pinnan kiiltävyyden muutoksiin. Texture-välilehdeltä löytää asetuksia, jos haluaa tietynlaisen tekstuurin objektille. Tämä tapa toimii yksinkertaisiin materiaaleihin, jotka halutaan koko objektiin tai esimerkiksi vain tietylle polygonille (KUVIO 6).



KUVIO 6. Yksinkertainen tapa lisätä materiaaleja

Edistyneempi tapa, jolla mallin tekstuurit ja materiaalit saa parhaiten näkyviin myös muissa ohjelmistoissa, on vaihtaa renderöintimoottori oletuksen ”Blender Renderistä” ”Cycles Renderiin”. Käyttöliittymässä ei tapahdu suuria muutoksia, mutta renderöinti ja materiaalien luonti muuttuu monipuolisemmaksi. Uv-mapituksen avulla objektin pintaan heijastetaan haluttu kuva. Jokaisesta pinnasta tehdään oma osio, joka heijastetaan haluttuun kohtaan kuvassa. Kun tämä on tehty, siirrytään Blenderissä node-editoriin, jossa materiaaliin liittyvät muokkaukset tehdään. Node-editor muistuttaa Unreal Enginen visuaalista ohjelmointiliittymää siinä mielessä, että siinäkin yhdistetään eri asetuksia toisiinsa visuaalisten viivojen avulla (KUVIO 7).



KUVIO 7. Uv-mapitus ja materiaalien muokkaus Node-editorilla

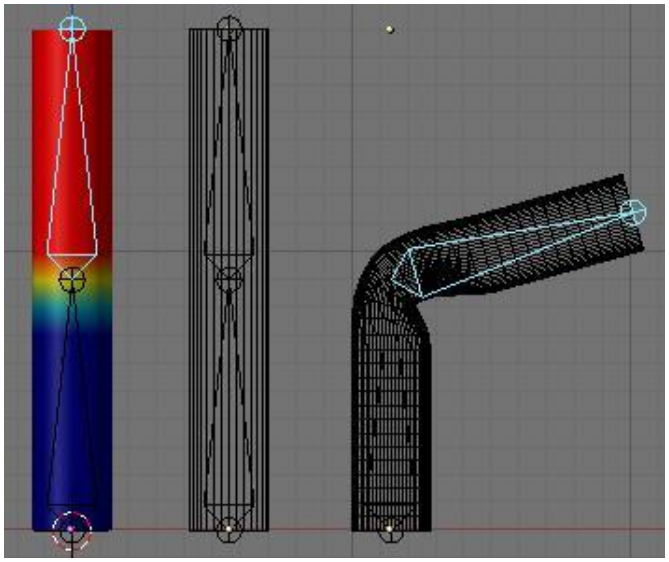
2.2.3 Animointi

Objekteja voi animoida Blenderissä monella tapaa. Yksinkertaisinta on liikuttaa tai kääntää koko objektia. Tämä onnistuu lisäämällä aikajanalle avainkehyksiä ja tallentamalla niihin objektin sijainti, koko ja rotaatio. Tämän jälkeen Blender osaa tehdä avainkehysten välille sulavan animaation, jossa koko objekti voi siirtyä, muuttaa kokoa tai pyöriä halutunlaisesti. (Blender Animation Introduction 2018 b, viitattu 7.10.2018.)

Toinen tapa animoida on muuttaa objektin muotoa. Yksittäisiä verteksejä on mahdollista animoida avainkehysten avulla. Tämän lisäksi voi käyttää muotoavaimia. Nämä toimivat siten, että perusmuodon lisäksi voi lisätä avaimia, joissa objektin verteksejä on siirretty antamaan objektille erilainen muoto. Muotoavain sisältää lukuarvon, jota muuttamalla muoto muuttuu perusmuodon ja muotoavaimen välillä. Avainkehyksen voi lisätä aikajanalle ja antaa sille muotoavaimen lukuarvo, jolloin saa aikaan animaation, jossa muotoavainten lukuarvot muuttuvat ja siten muuttavat objektin muotoa. (Blender Animation Shape Keys 2018 c, viitattu 7.10.2018.)

Kolmatta tapaa käytetään erityisesti hahmojen animoinnissa. Objektia liikutetaan toisen objektin mukaan, kuten omistajan tai hahmon rangan mukaan. Armature eli ranka voidaan lisätä myös muihin objekteihin, sillä ranka sisältyy eri määristä luita, jotka voivat olla minkä kokoisia tahansa. Jokaisen luun välillä on nivel, jolloin luita voi käännellä hallitusti. Näin ollen rangan voi lisätä niin lyhdylle kuin vaikkapa ihmishahmollekin. Avainkehyksiä lisäämällä rankaa voi animoida samalla tavoin sijaintia, rotaatiota ja kokoa muuttaen. (Blender Rigging 2018 d, viitattu 7.10.2018.)

Ranka itsessään ei riitä, jotta objektia voisi alkaa muuttamaan. Ranka pitää liittää haluttuun objektiin ja jokaiselle luulle pitää määrittää oma pinta-alansa, jota luu hallitsee. Tämä onnistuu liittämällä luu objektille joko automaattisesti luomalla objektin pinnalle rajat pinta-aloille tai itse maalamalla. Blenderistä löytyy Weight paint -tila, jolla saa hallittua pintoja, joita eri luut hallitsevat. Pinnalle maalamalla voi joko lisätä tietyn luun pinta-alaa tai vähentää sitä. Punainen väri on aluetta, jota valittu luu hallitsee ja sininen on aluetta, jota se ei hallitse ollenkaan. Vaikein osuus on luiden raja-alueet, jotka pitää testaamalla säätää oikein (KUVIO 8).



KUVIO 8. Luiden hallitsema pinta-ala

2.3 Oculus Rift

Oculus Rift on 2016 julkaistut virtuaalitodellisuuslasit. Kyseiset lasit ovat Oculus Vr -yrityksen ensimmäiset kuluttajille suunnatut virtuaalilasit. Myöhemmin samana vuonna julkaistiin myös Touch-ohjaimet. Kuluttajaversiossa on kaksi OLED-näyttöä, joiden resoluutio on 1080 x 1200 per silmä. Virkistystaajuus on 90 hertsiä, ja näkökentän laajuus on 110 astetta. Laitteisto tunnistaa liikkeen ja rotaation sisäisten sekä usb-sensorien avulla. Lasit sisältävät myös kuulokkeet (KUVIO 9). (Wikipedia Oculus 2018 c, viitattu 9.10.2018.)

KUVIO 9. Oculus Rift Cv1 Touch-ohjaimilla

Ensimmäiset kehityskäyttöön tehdyt lasit julkaistiin 2013, ja tätä versiota kutsuttiin Development Kit 1:ksi. Seuraavan version, eli Development Kit 2:n myynti aloitettiin vuonna 2014 (KUVIO 10). Erot ensimmäisten versioiden välillä olivat muun muassa korkeampi tarkkuus sekä virkistystaajuus. Tämän lisäksi versio 2 tuki liikeseurantaa. (Wikipedia Oculus 2018 c, viitattu 9.10.2018.)

KUVIO 10. Oculus Rift Dk1 ja Dk2

3 PELISUUNNITTELU

Pelisuunnittelu on luovaa työtä, jonka avulla pelit tuodaan eloon niin pelimaailman, hahmojen, tarinan kuin pelattavuudenkin kannalta. Pelisuunnittelijan rooli voi vaihdella sen mukaan, minkä kokoinen tiimi peliä on tekemässä ja mitkä ovat kiinnostuksen ja osaamisen kohteet. Tasosuunnittelija vastaa pelialueista, ympäristön luomisesta sekä tapahtumien ja tehtävien sijoittamisesta. Tarinankirjoittaja vastaa mukaansatempaavasta tarinasta. Sisällöntuottaja luo pelimaailmaan yksittäisiä esineitä ja kohteita yleensä 3d-mallintamalla tai piirtämällä. Ohjelmoija suunnittelee ja ohjelmoi pelattavuutta ja toimintoja mahdollisimman toimivan pelikokemuksen takaamiseksi. Käyttöliittymäsuunnittelijat vastaavat valikkojen ja muiden liittymien ulkonäöstä ja toiminnallisuudesta. (International Student 2018, viitattu 9.10.2018.)

Tässä projektissa oli vain yksi kehittäjä. Tällöin vastuualueena oli koko pelin suunnittelu ja kehitys. Pelidemo on suunniteltu hyödyntäen GDD eli pelisuunnitteludokumenttia. Pelisuunnitteludokumentti löytyy tämän opinnäytetyön liitteistä. (Liite 1.)

3.1 Peliaihe

VR-pelidemon aiheena toimii maanviljelyteema. Pelaaja omistaa maatilan, ja hänen on aloitettava kasvien ja puiden istutus, jotta hän menestyy pelissä ja ansaitsee rahaa esineiden sekä entistä tehokkaampien työkalujen ostoon. Pelin edetessä pelaaja saa kasvatettavakseen erilaisia kasveja ja hän pystyy päivittämään maatilaansa esimerkiksi lisäämällä istutuspaikkoja, aitoja ja hankkimalla entistä tehokkaampia työkaluja. Maatilan läheisyydessä on pieni kylä, johon pelaaja voi kuljettaa satonsa ja jossa hän voi ostaa tarvikkeita lisää. (Liite 1.)

Virtuaalitodellisuudessa pelaaja pystyy monimutkaisempiin toimintoihin kuin perinteisissä peleissä. Tavalliselta tuntuva toiminto, kuten esimerkiksi oven avaaminen, on virtuaalilaseilla todellisempaa kuin perinteisissä peleissä. Pelin aihetta pohtiessa maanviljelyteema tuntui sopivalta useasta syystä: maanviljelyyn liittyy paljon pieniä toistoja, kasveista täytyy huolehtia kastelemalla ja keräämällä niitä, pitää käyttää oikeanlaisia työkaluja sekä nostella, siirrellä ja käyttää monenlaisia esineitä. Nämä toiminnot ovat yleensä hyvinkin yksinkertaisia, eivätkä ne välttämättä perinteisissä

peleissä tuota sen kummempia haasteita. Virtuaalitodellisuudessa nämä toiminnot ovat monimutkaisempia. Esimerkiksi kasvien kastelu vaatii pelaajalta kastelukannun nostamisen, sen kantamisen määränpäähän ja kallistamisen, jotta toiminto saadaan suoritettua. Hiirellä ja näppäimistöllä todentuntuinen tavaroiden kantaminen ja hallinta ei onnistu.

3.2 Pelin toiminnot

Pelisuunnitelmaan on kuvattu tietty määrä erilaisia toimintoja. Minimivaatimuksena oli, että demoon perustoiminnot saadaan toimimaan. Perustoimintoja ovat liikkuminen ja erilaisten esineiden interaktiivisuus, kasvien hoito sekä kauppajärjestelmä.

Näiden lisäksi oli myös muita toimintoja: maatilaa pitää pystyä kehittämään, eli siellä pitäisi voida sijoittaa uusia esineitä haluttuihin paikkoihin. Tämän lisäksi pelaajalla voisi olla myös ajoneuvo, jolla hän voi kuljettaa valmista satoa kylään myytäväksi ja myös tuoda uusia esineitä maatilalle. Virtuaalitodellisuus vaatii omanlaisen käyttöliittymän. Perinteisen näytöllä näkyvän käyttöliittymän lisäksi virtuaalitodellisuudessa voidaan lisätä valikkoja myös käsiin, kuten esimerkiksi ranteisiin. (Liite 1.)

3.3 Ympäristö

Ympäristö muodostuu pelidemossa pelaajan omistamasta maatilasta, sen läheisyydessä sijaitsevasta järvestä sekä kylästä. Maatila sisältää päärakennuksen ja mahdollisia muita tiloja eri tarpeisiin. Pelaaja voi myös muokata maatilaa lisäämällä ja siirtämällä erilaisia tavaroita alueella. Kylässä on useita rakennuksia. Osa niistä on kauppoja, joiden sisään pelaajan tulee päästä.

Maatilan ja kylän välillä on erilaista tietä, metsää ja muuta täydennystä, lisäksi kartan reunamilla on vuoria estämässä pelaajaa menemästä kartan ulkopuolelle. Ympäristössä olevat puut ja pensaat heiluvat tuulen mukana. Ympärillä kuulee tuulen ääntä ja tavallisia luonnon ääniä, kuten lintujen laulua, jotka elävöittävät pelimaailmaa. (Liite 1.)

3.4 Alkuperäinen vastaan valmis pelisuunnitelma

Alkuperäisessä pelisuunnitelmassa tarkoitus oli tehdä useita toimintoja pelidemoon. Joistakin toiminnoista oli kuitenkin pakko luopua aiheen rajaamisen sekä aikataulun takia. Demoa tehdessä toimintoja luotiin niiden tarpeellisuuden mukaan: ensimmäiseksi kehittyi pelaajan liikkuminen ja tavaroiden nosto. Seuraavaksi pelaajalle lisättiin 3d-keho, joka seurasi pään ja käsien liikkeitä. Kasvien hoito loi pohjan maanviljelyteemalle, ja kauppajärjestelmä taas toimii pohjana pelissä edistymiselle.

Demosta jäi uupumaan opinnäytetyön ajalta esimerkiksi ajoneuvo sekä tallennusmahdollisuus. Maatilaa eikä sen työkaluja voi myöskään päivittää. Nämä toiminnot tulevat demoon mahdollisesti myöhemmin opinnäytetyön valmistuttua. Ympäristön suhteen demoon saatiin pelaajan oma maatila, kauppa ja järvi. Kartan reunamilla on vuoristoa. Tarkempi täydennys, kuten kylä ja erilaiset tiet, jäivät puuttumaan.

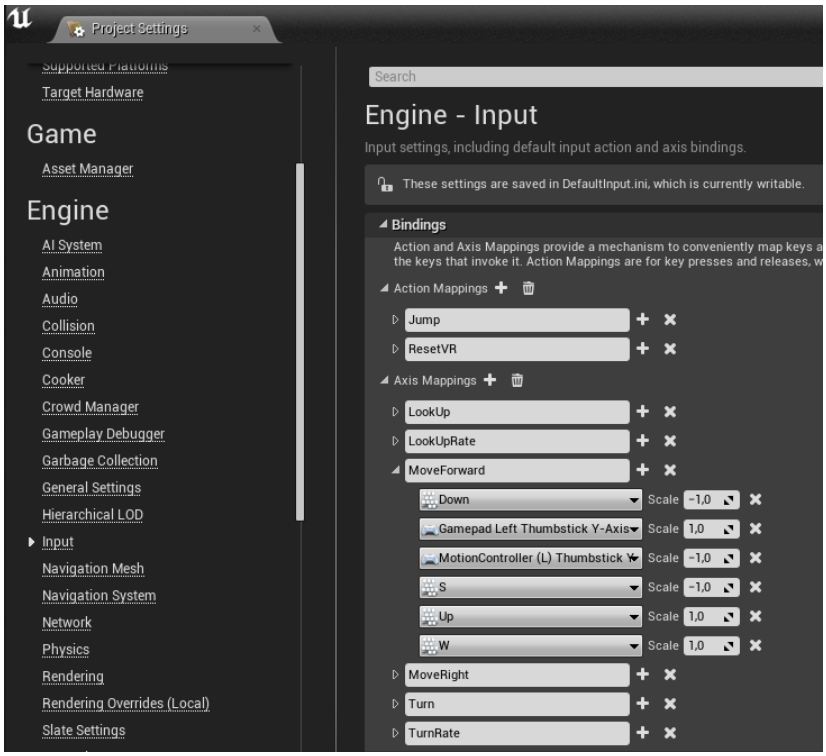
4 PELIN TOTEUTUS

Pelin toteutus alkoi Unreal Engineen tutustumisella. Unrealin virallisilta sivuilta löytyi dokumentaatiota jokaisesta osa-alueesta pelimoottoriin liittyen. Sieltä löytyi myös oma osa-alue virtuaalitodellisuuteen. (Unreal Engine Virtual Reality Development 2018 e, viitattu 29.10.2018.) Tätä kautta pääsi alkuun ja ymmärsi perusasiat pelimoottorin käytössä. Erilaisilta keskustelupalstoilta löytyi aiheita, jotka liittyivät omiin ongelmiin: joku kysyi apua omaan ongelmaansa, ja toisella saattoi olla vastaus siihen. Keskustelupalstoja selaamalla ja oikeita hakusanoja käyttämällä sai ratkaistua monia ongelmia.

Videot ovat kenties parhain keino uuden oppimiseen. YouTubesta löytyi kattavasti videoita liittyen pelimoottoreihin ja virtuaalitodellisuuteen. Erityisen hyvä kanava Unreal Engineen ja virtuaalitodellisuuteen liittyen on Marco Ghislanzonin kanava (YouTube Marco Ghislanzoni 2018 a, viitattu 29.10.2018). Pelidemon koodissa onkin paljon viitteitä kyseisen kanavan sisältöön. YouTubessa on myös Unreal Enginen oma virallinen kanava, josta löytyi pitempiä videoita myös virtuaalitodellisuuteen liittyen (YouTube Unreal Engine 2018 b, viitattu 29.10.2018).

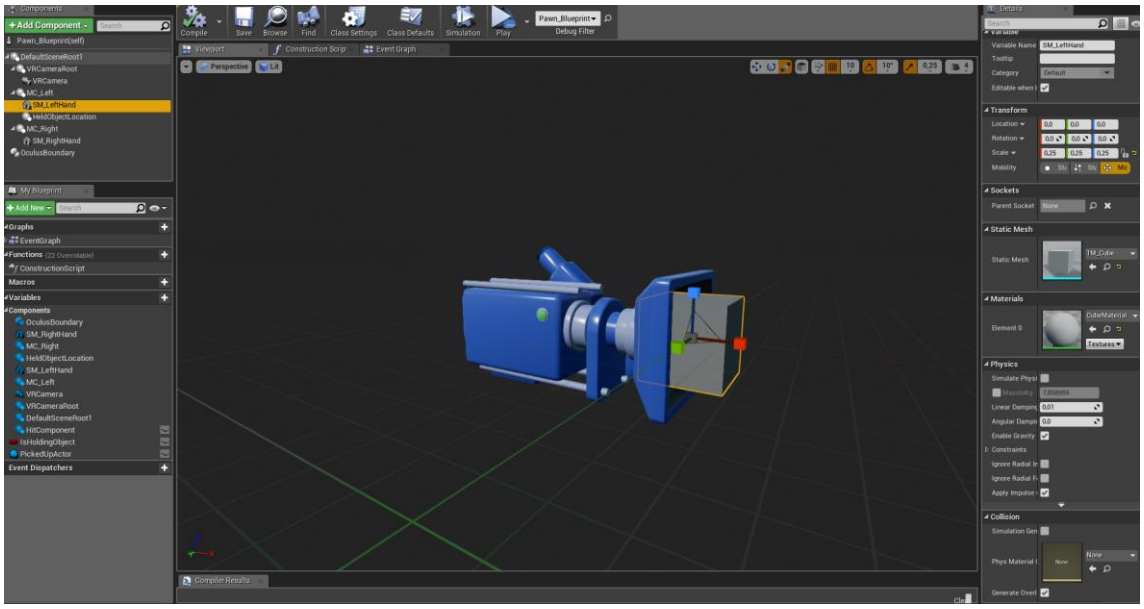
4.1 Liikkuminen

Ensimmäiseksi virtuaalitodellisuuspeliä tehdessä piti saada toimimaan pelaajan liikkuminen pelimaailmassa. Valikosta projektia luodessa on mahdollista valita virtuaalitodellisuusosio, jolla saa projektiin valmiita toiminnallisuuksia ja tasoja, joissa toimintoja on mahdollista testata. Näihin toiminnallisuuksiin kuuluu esimerkiksi perusliikkuminen teleportaation avulla. Tämä pelaajan kontrolloima pawn-blueprint-luokka sisältää myös kädet, jotka reagoivat animaatiolla ja tärinällä tarrautumiseen. Liikkuminen ei kuitenkaan onnistu pelkästään lisäämällä pawn-objekti tasolle. Koodia täytyy muokata ja lisätä Oculuksen touch-ohjaimen näppäimet projektin kontrollien asetuksiin (KUVIO 11). Sitomalla syöttökomennot muuttujiin voidaan koodissa kutsua näitä muuttujia.

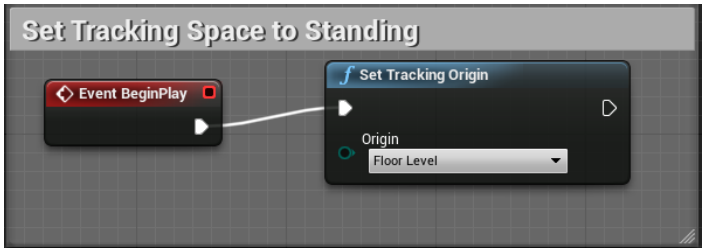


KUVIO 11. Projektin syöttökomentojen sitominen muuttujiin

Valmista pawn-luokkaa ei kuitenkaan käytetty demossa, vaan se oli apuna lähinnä sen hahmottamisessa, miten visuaalinen koodi toimii Unrealissa. Syynä oli myös se, että henkilökohtaisesti mukavampaa on liikkuminen, jossa pelaaja oikeasti liikkuu tietyllä nopeudella eteenpäin, toisin kuin teleportaatiossa, jossa pelaaja siirtyy välittömästi osoittamaansa paikkaan. Työ alkoi siitä, että luotiin uusi pawn-luokka. Avaamalla luokan saa näkyville näkymän, josta näkee komponentit ja muuttujat pawn-luokalle (KUVIO 12). Luomalla juuren ja lisäämällä kamerakomponentin sai yksinkertaisimmillaan tason näkymään myös virtuaalilaseilla. Tämän lisäksi lisäämällä funktion SetTrackingOrigin saa kameran nousemaan pelaajan korkeudelle sen mukaan, pelaako istualtaan vai seisaaltaan (KUVIO 13). Tällöin piti muistaa kuitenkin lisätä kamerakomponentti lattiatasolle antaen z-arvoksi nolla, jotta kamera ei nouse liian korkealle pelaamisvaiheessa.

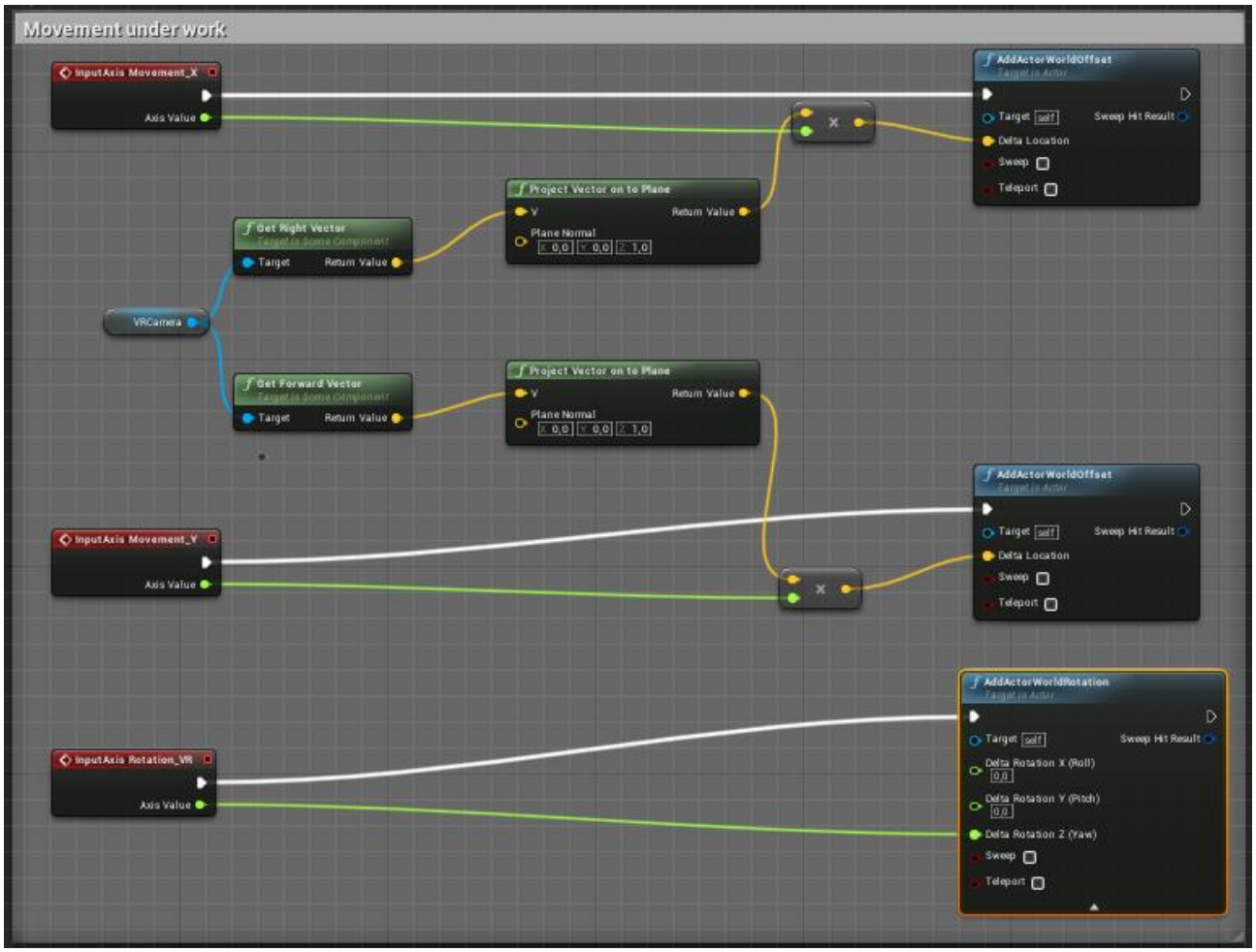


KUVIO 12. Viewport-näkymä sisältäen pawn-luokan komponentit ja muuttujat



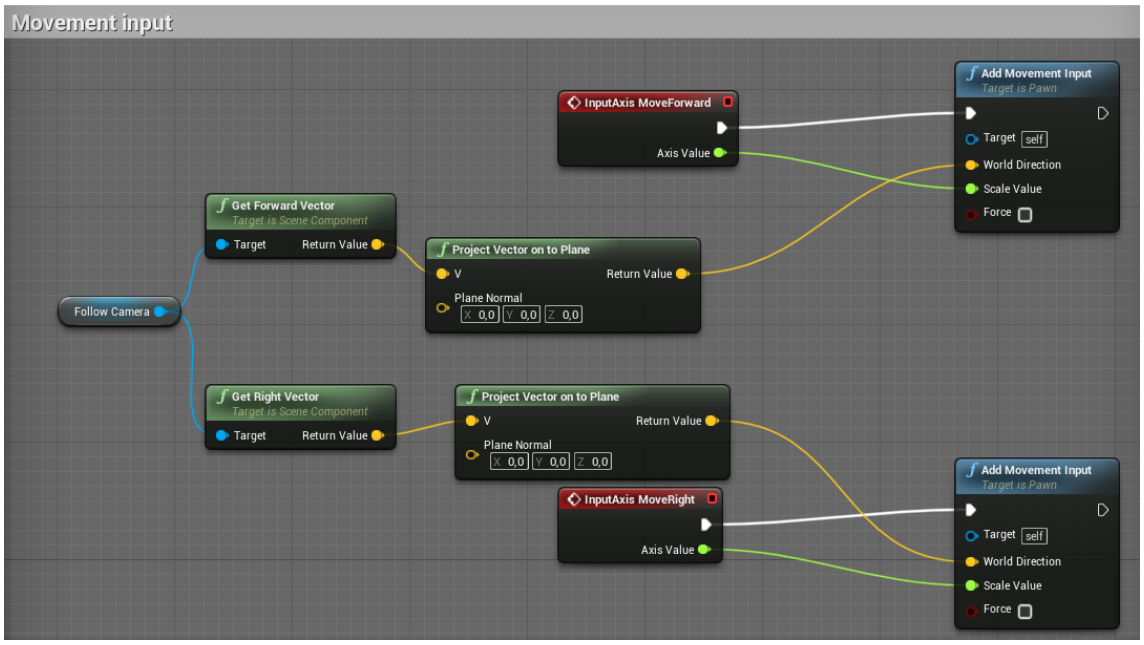
KUVIO 13. SetTrackingOrigin seisaaltaan pelaamiseen

Pawn-luokan ohjaamisessa hyödynnettiin syöttökomentojen muuttujia. Kun vasemman touch-ohjaimen tattia liikutetaan joko eteen tai taaksepäin, y-akseli saa arvon -1:n ja 1:n välillä. Jos tattia liikutetaan vasemmalle tai oikealle, saa x-akseli arvon -1:n ja 1:n väliltä. Näin ollen syöttökomennon avulla saa selville, liikuttaako mihin suuntaan ja millä nopeudella pelaaja tattia. Selvittämällä kamerakomponentin vektorien suunnan ja lopulta yhdistämällä vektorin ja syöttökomennon arvot saadaan pawn-luokkaa liikutettua lisäämällä loppuun AddActorWorldOffset-funktio. Ympäri kääntyminen onnistuu lisäämällä funktio AddActorWorldRotation, ja syöttökomennon arvoa eli oikean touch-ohjaimen x-akselin arvoa katsomalla saadaan kääntymisen nopeutta hallittua (KUVIO 14).

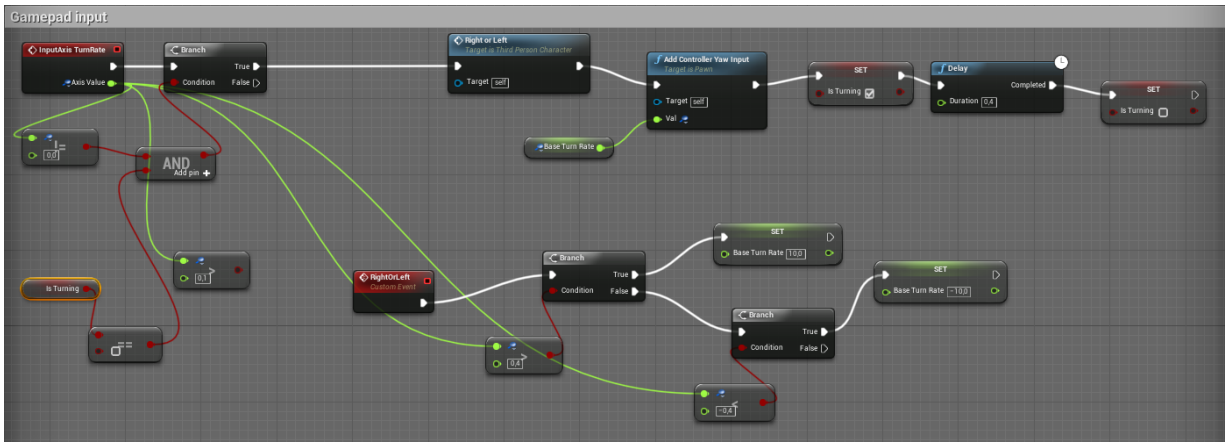


KUVIO 14. Liikkuminen ja kääntyminen AddActorWorldOffset- ja AddActorWorldRotation-funktioilla

Myöhemmin pelaajan pawn-luokka vaihtui. Pelkkien leijuvien käsien sijaan muokattiin jo valmista ThirdPersonCharacter-luokkaa siten, että se toimi virtuaalilaseilla. Kolmannen persoonan hahmon animaatiot eivät enää toimineet AddActorWorldOffset-funktiota käyttämällä. Uusi funktio tilalle oli AddMovementInput (KUVIO 15). Pelaajan kääntymiseen liittyen vapaa kääntyminen muuttui nykäyksittäin tietyn astemäärän kerralla tapahtuvaksi. Tämä vähensi pahoinvointia, kun kääntyminen tapahtui kerralla. Tämä vaati, että lisättiin uusi muuttuja kerralla tapahtuvalle astemäärälle. Muuttujalle annetaan tietty arvo sen mukaan, meneekö pelaaja oikealle vai vasemmalle. Kun kääntyminen tapahtuu, asetetaan pieni viive ennen kuin toiminnon voi suorittaa uudelleen, jotta estetään useampi nykäys kerralla (KUVIO 16).



KUVIO 15. Liikkuminen AddMovementInput-funktiolla

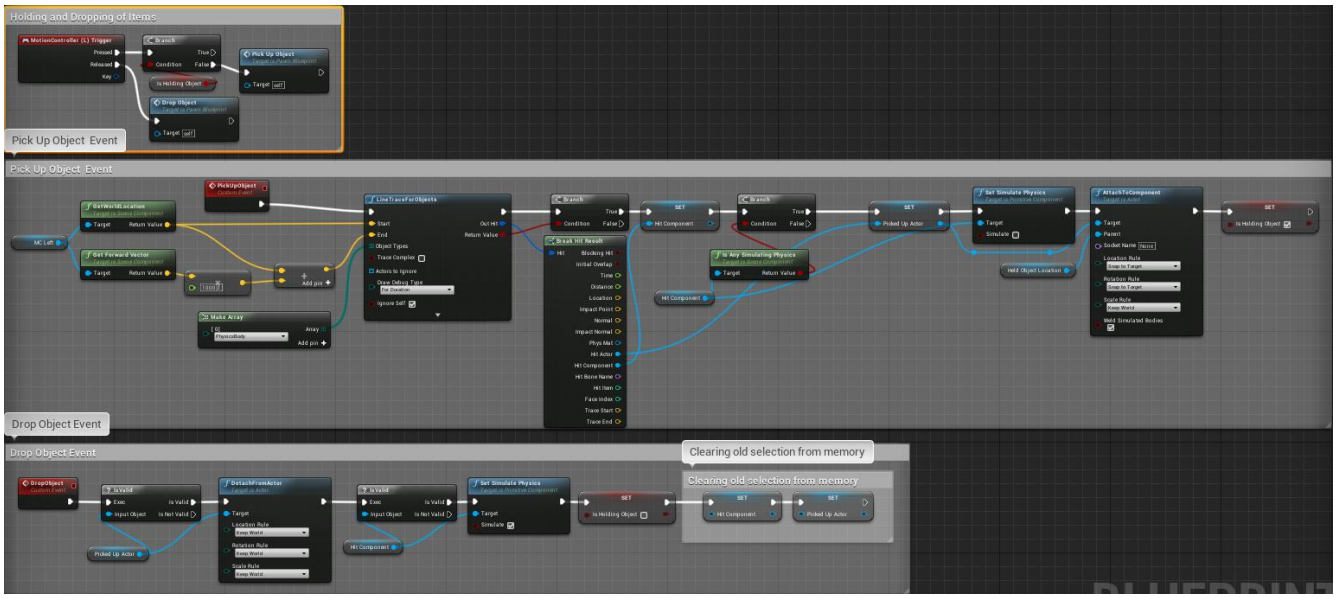


KUVIO 16. Monimutkaisempi ympärikääntyminen, joka tapahtuu tietty astemäärä kerrallaan.

4.2 Tarttuminen esineisiin

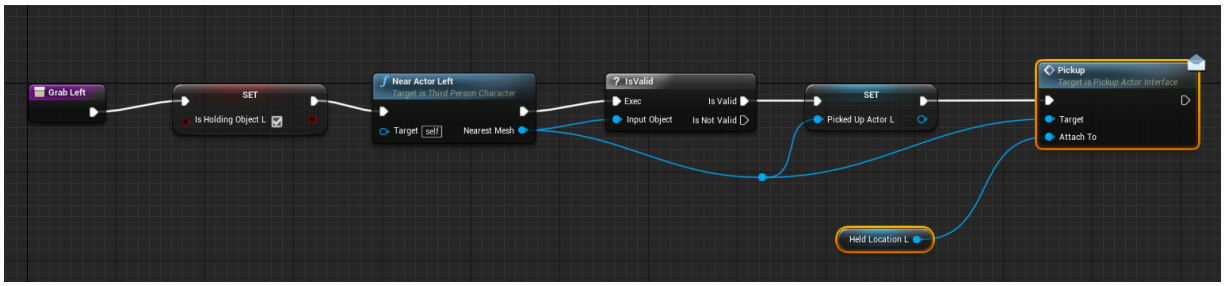
Esineisiin tarttumiseen on myös toimiva pawn-luokka jo valmiina. Se on sama, jossa hallitaan pelaajan liikettä teleportaation avulla. Valmis pohja sisältää funktioita, joissa tarkastetaan ja aktivoidaan tuloksien mukaan tarttumiseen liittyviä objekteja ja koodia. Tässä pohjassa tarkastetaan pelaajan lähellä oleva objekti pallon muotoisella törmäyskomponentilla. Jos objekti on törmäyskomponentin sisällä, tarkastetaan, onko objektilla PickupActorInterfacea. Jos on, niin objekti lisätään osaksi pelaajaa siksi ajaksi, kun objektista pidetään kiinni.

Demon ensimmäisessä versiossa esineillä ei ollut interface-luokkaa. Sen sijaan mikä tahansa esine, jolla oli simuloidut fysiikat, kävi nostettavaksi esineeksi. Törmäyskomponentteja ei myöskään käytetty lähellä olevien objektien löytämiseen, vaan luotiin sen sijaan tietyn mittainen jana pelaajan käden eteen, joka osuessaan kohteeseen aktivoi tarttumisen. Tämä tapahtui funktion LineTraceForObjects avulla. Kun sopiva objekti löytyi, poistettiin kyseisestä objektista fysiikat ja kiinnitettiin se kädessä olevaan komponenttiin. Kun pelaaja ei pitänyt enää kiinni esineestä eli löysäsi ohjaimen liipaisimen, irrotettiin objekti pelaajan kädestä ja lisättiin fysiikat, jolloin objekti käyttäytyi taas itsenäisesti (KUVIO 17).

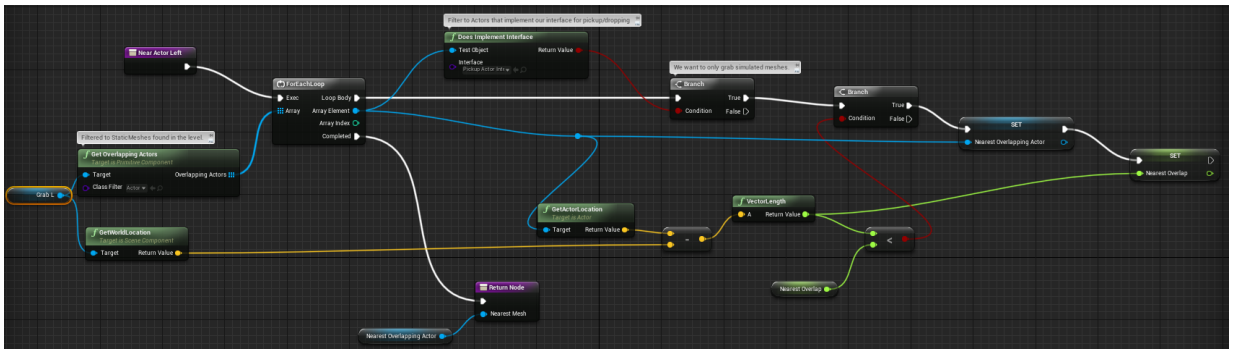


KUVIO 17. Tarttuminen ja pudottaminen käyttäen LineTraceForObject-funktiota

Kohteen etsiminen kuitenkin pelkän ohuen janan avulla ei ollut kovin toimivaa. Törmäyskomponentit sekä interface-luokat otettiin käyttöön, jolloin pystyy hallitsemaan, mitkä esineet ovat tartuttavissa, ja esineisiin pystyi tarttumaan laajemmalla alueella. Funktioiden käyttö auttoi selkeyttämään koodia, jolloin jokainen osio oli omassa osassaan. Kummallekin kädelle on omat funktionsa, jotta ne voivat toimia toisistaan riippumatta. GrabLeft- ja GrabRight-funktiot kiinnittävät pidetyn objektin käsien omiin komponentteihinsa, mutta ennen sitä ne käyvät läpi NearActorLeft- ja NearActorRight-funktiot (KUVIO 18). NearActor-funktiot tarkastavat, että objektilla, joka on törmäyskomponentin sisällä, on PickupActorInterface. Tämän lisäksi funktiot tarkistavat, mikä objekti on lähimpänä, ja tallentavat tämän objektin ja sen sijainnin (KUVIO 19). Nämä tarttumisfunktiot ovat melkein kokonaan hyödynnetty valmiista pawn-luokasta, joka sisälsi liikkumisen ja tarttumisen. Suurin ero on siinä, että omassa pawn-luokassa kummatkin kädet ovat samassa, eivätkä erillisinä pawn-luokkina.



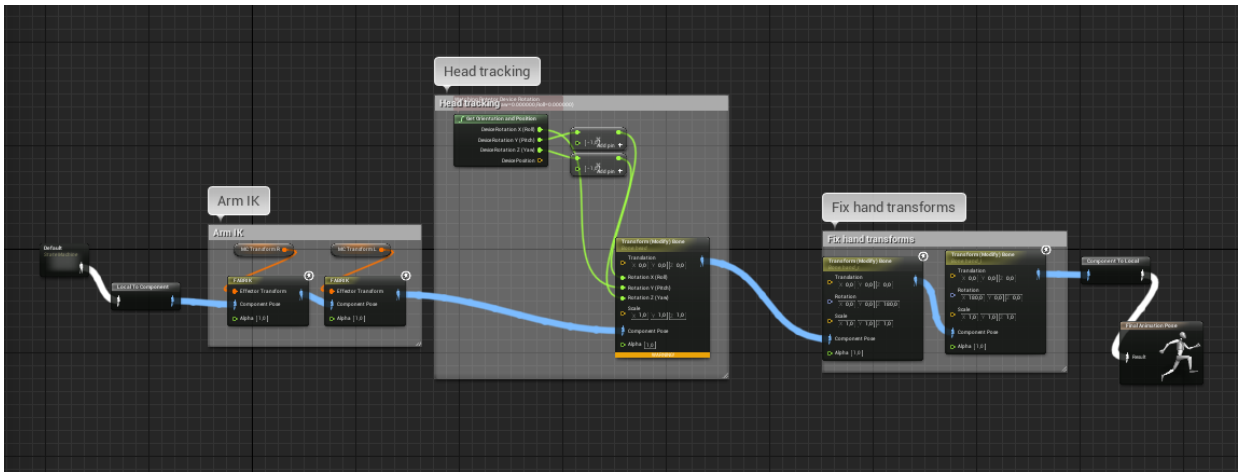
KUVIO 18. GrabLeft-funktio vasemmalle kädelle



KUVIO 19. NearActorLeft-funktio vasemmalle kädelle

4.3 Kehon hallinta

Kolmannen persoonan valmista luokkaa käyttämällä sai demoon näkyviin virtuaalitodellisuushahmon. Kamera seurasi hahmoa luoden aikaan oman virtuaalisen kehon. Ohjaimia ja pään liikkeitä hyödyntämällä sai pelkkien animaatioiden sijaan hallittua hahmon käden ja pään liikkeitä suoraan. Muokkaamalla hahmon animaatioluokkia sai aikaan, että tallentamalla kummankin käden sijainnin muuttujiin, pystyi käsien luut siirtämään näihin kohtiin. FABRIK-funktio laskee annetun arvon perusteella luun sijainnin ja korjaa muiden luiden asentoa sen mukaiseksi. Lopuksi käsien luita täytyi kiertää, jotta ne vastasivat oikeita käsien asentoja. Pään kääntymistä pystyi hallitsemaan yksinkertaisemmin käyttämällä avuksi GetOrientationAndPosition-funktiota, joka nimensä mukaisesti antaa ulostulona virtuaalilasien sijainnin ja kulmat. Kulmat voidaan pienten laskutoimitusten jälkeen kääntää vastaamaan pään luun kulmaa, jolloin hahmon pää saatiin kääntymään virtuaalilasien suuntaan (KUVIO 20).



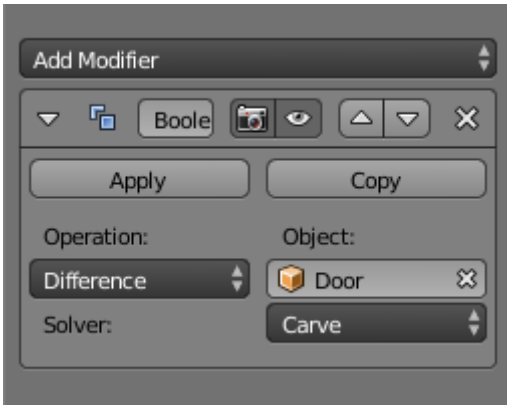
KUVIO 20. Touch-ohjainten ja virtuaalilasien mukaan kääntyvät käsien ja pään luut

4.4 Pelaajan koti

Mallintaminen aloitettiin suunnittelemalla pelaajan talo. Suunnitteilla oli, että mökki olisi suhteellisen kompakti, mutta kumminkin tarpeeksi tilava virtuaalitodellisuus huomioon ottaen. Netistä etsimällä muutaman referenssikuvan, alkoi mökin mallintaminen Blenderillä (KUVIO 21). Samalla etsittiin myös yleisiä kokoja ovien ja ikkunoiden korkeuksille, jotta mökki vastaisi oikeita mittoja.

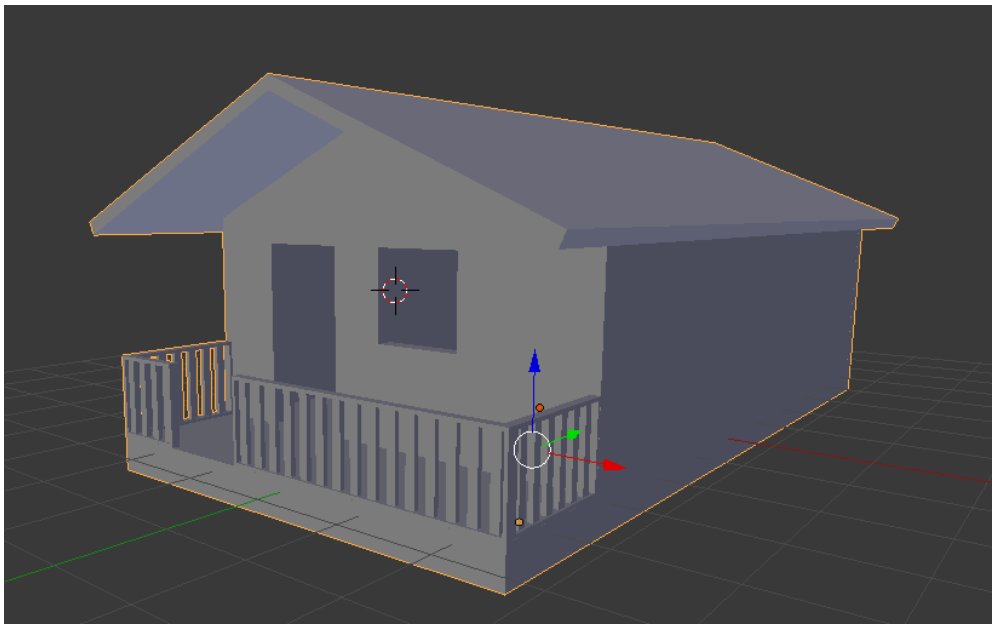
KUVIO 21. Mökin referenssikuvia

Mökin mallintaminen alkoi pohjasta. Leveydeksi tuli 5 metriä ja pituudeksi 10 metriä. Päätyseinät ovat viisikulmaisia eli katto on korkeimmillaan keskeltä. Koska kummatkin päätyseinät ovat samanlaisia, pystyi valmiin päätyseinän kopioimaan haluttuun kohtaan. Boolean työkalua käyttämällä sai leikattua ovelle ja ikkunoille aukon seiniin. Ensin luodaan sopivilla mitoilla ovea esittävä objekti, sen jälkeen lisätään se kohtaan, johon aukko halutaan ja lisätään Boolean-työkalu. Työkalun operaation vaihtoehtoja selaamalla saa erilaisia efektejä siitä, miten haluttu objekti leikkaa muokattavaa objektia (KUVIO 22).



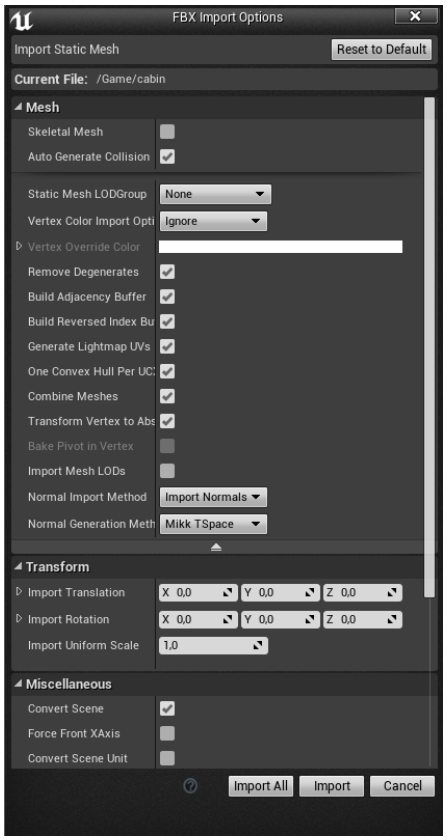
KUVIO 22. Boolean-työkalu

Mökkiin kuului myös kuisti kaiteineen. Kaiteissa voi hyödyntää Booleania, mutta laudat kaiteen ympärille sai myös joko manuaalisesti luomalla tai kopioimalla kaiteen ja skaalaamalla kopiota pienemmäksi ja sitten sisäpinnat poistaen. Laudat kaiteeseen saa yksinkertaisesti tekemällä yhden laudan mitat valmiiksi ja hyödyntämällä Array-työkalua. Tämän työkalun avulla on mahdollista luoda useita kopioita halutusta objektista tietyille etäisyyksille toisistaan. Näin jokaiselle kaiteelle saa halutun määrän lautoja ja tietyn etäisyyden jokaisen laudan väliin. Tämän jälkeen sivuseinät lisäämällä ja muita yksityiskohtia tekemällä mökki alkoi näyttämään seuraavanlaiselta (KUVIO 23).



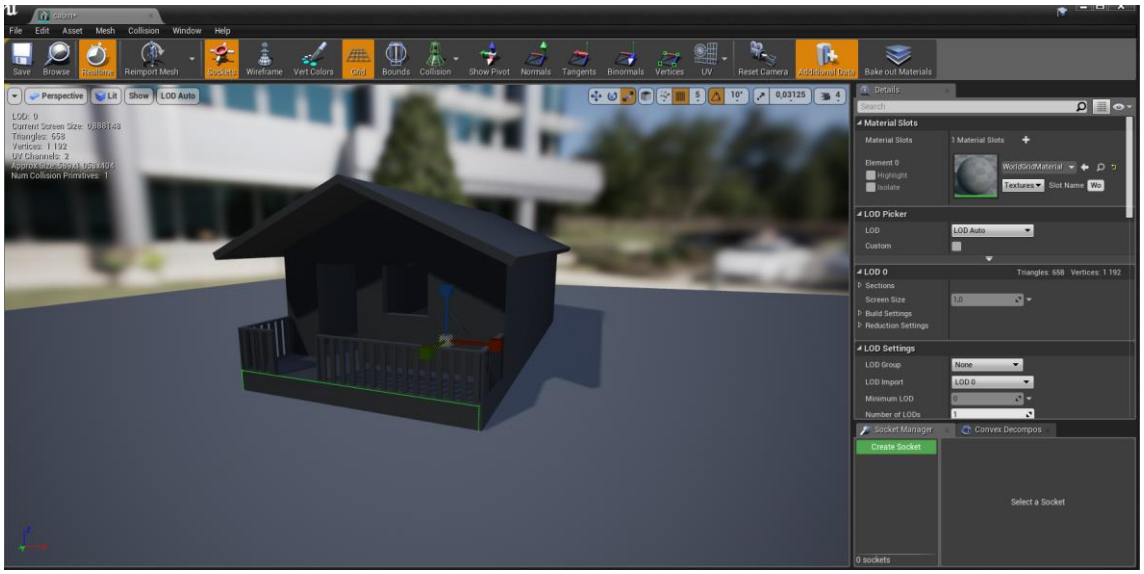
KUVIO 23. Mökki perusmuodoiltaan

Tämän jälkeen testattiin mökin toimivuutta Unrealin puolella. Ennen vientiä piti muistaa resetoida mökin skaala ja asettaa jokaisen osan alkupiste oikein 3d-työkalussa. Tämän jälkeen Blend tiedostomuoto vietiin FBX-muotoon, jolloin Unreal tunnistaa 3d-mallin. Unrealin sisällä tuominen aukaisee valikon, josta voi valita eri asetuksia, kuten esimerkiksi sen, haluaako yhdistää eri osat yhteen vai haluaako Unrealiin tuoda myös materiaalit ja tekstuurit (KUVIO 24).



KUVIO 24. Unrealin FBX-tiedoston tuontiasetukset

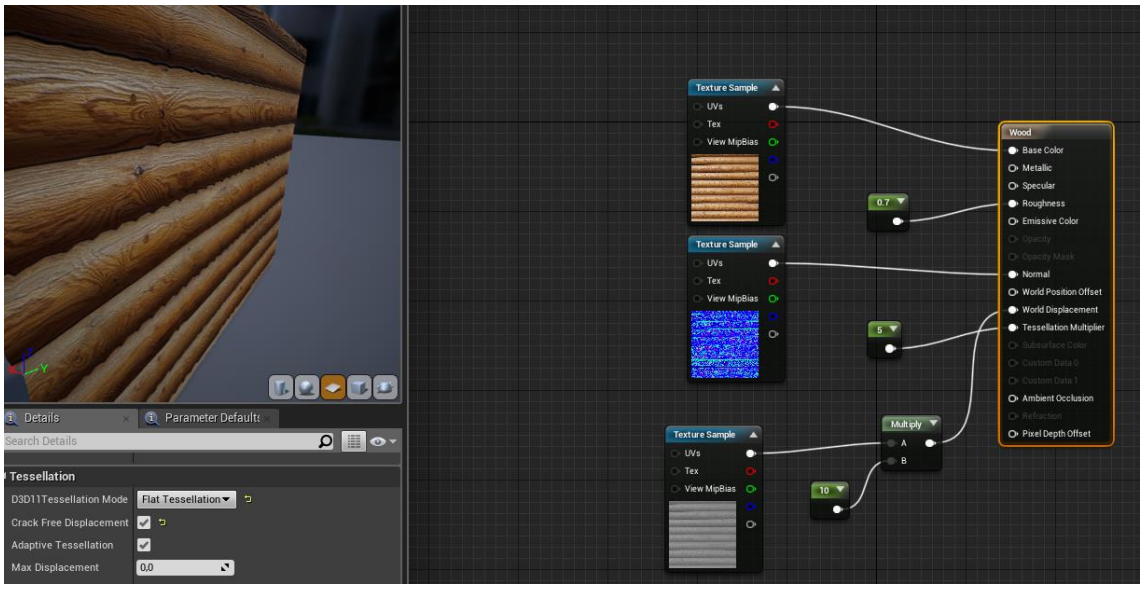
Asetuksista oli mahdollista valita, että Unreal luo automaattisesti törmäysalueen mallille. Tämä kuitenkin mökin tapauksessa tarkoitti sitä, että pelimoottori lisäsi ympärille alueen, jonka sisälle ei päässyt ollenkaan. Törmäysalueiden lisäys onnistuu Unrealissa manuaalisesti aukaisemalla 3d-mallin uuteen ikkunaan. Sieltä käsin on mahdollista luoda eri muotoisia komponentteja törmäysalueille sekä samalla muokata myös esimerkiksi materiaaleja (KUVIO 25). Manuaalisesti törmäysalueiden lisäys osoittautui kuitenkin hyvin epätarkaksi, sillä esimerkiksi kokoa oli hyvin vaikea hallita, jolloin törmäysalueesta saattoi tulla liian suuri. Automaattisesti törmäysalueiden luonti onnistuu helposti, mutta vaarana voi olla liian yksityiskohtainen törmäysalue.



KUVIO 25. 3d-mallin näkymä

Toinen tapa lisätä törmäyskomponentit 3d-mallille on tehdä ne suoraan 3d-editorissa. Näin sijaintia ja kokoa on paljon helpompi hallita. Nimeämällä osat editorissa nimellä UBX_(osan nimi) saa aikaan laatikon muotoisen törmäysalueen. UCP_(osan nimi) saa aikaan kapselin, ja USP_(osan nimi) saa aikaan pallon muotoisen törmäyskapselin ja UCX_(osan nimi) saa aikaan kuperan muotoisen törmäysalueen. Törmäysalueita tehdessä yksittäisiä verteksejä ei saa siirtää. (Unreal Engine FBX Static Mesh Pipeline 2018 c, viitattu 12.9.2018.)

Mökin tekstuureihin palattiin projektin loppupuolella. Blenderin kautta materiaalien siirto ei onnistunut, joten materiaalit täytyi tehdä Unrealin kautta. Mökillä on neljä materiaalia: lattia, seinä, katto ja kivijalka. Materiaaleista saa todemman näköisiä, kun käyttää normal- sekä displacement-tekstuureja pinnan muotoihin. Normal-tekstuuri luo pintaan keinotekoisen valon mukaan heijastavan pinnan, ja displacement-tekstuuri tekee pintaan näkyvää muotoa. Seinämateriaalin saa yksinkertaisesti lisäämällä sopivan tekstuurin Base-coloriin. Kun ottaa materiaalista Tessellation-ominaisuuden käyttöön, voi uppoamatekstuuria käyttää pinnan muotoon. Normaalitekstuuri yhdistetään normal-kohtaan, ja materiaali on valmis (KUVIO 26). Erilaisia tekstuureja kuvasta saa aikaan tähän tarkoitukseen tehdyillä työkaluilla. Esimerkkinä Normal Map Online, joka on netistä löytyvä ilmainen selaimessa toimiva ohjelma, jota käytettiin materiaalien luonnissa (Normal Map Online, viitattu 19.11.2018). Lopputuloksena mökki näytti Unrealissa seuraavanlaiselta (KUVIO 27).

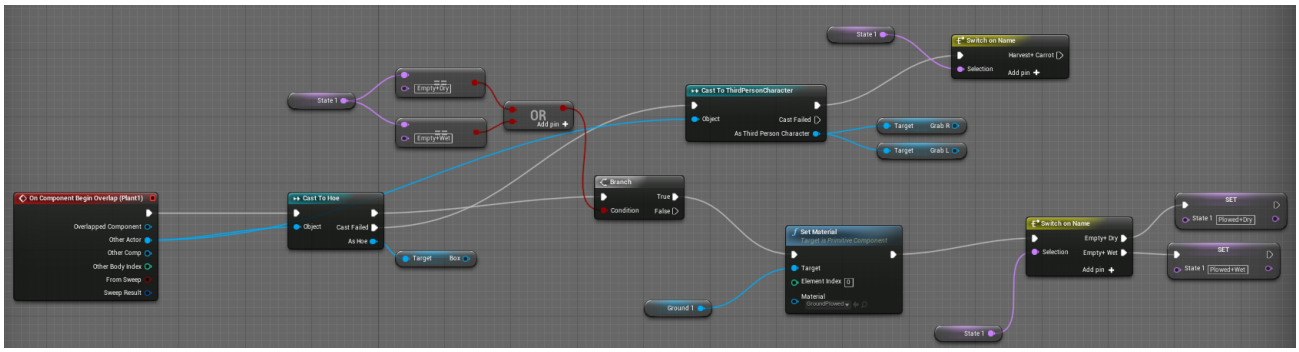


KUVIO 26. Seinämateriaali normal- ja displacement-mapilla

KUVIO 27. Mökki teksturoituna

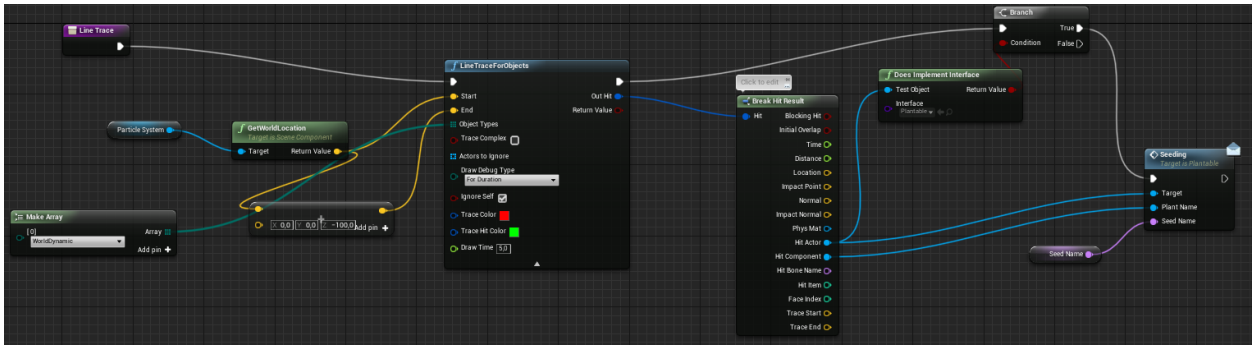
4.5 Kasvien hoito

Istutus on kaikille kasveille sama. Tämä helpottaa ohjelmointia ja pelattavuutta. Istutukseen vaikuttavat erilaiset työkalut, kuten kastelukannu, kuokka ja istutettavat siemenet. Istutuksen hahmottaminen alkoi luomalla väliaikaiset mallit ja neljä istutuspaikkaa. Begin overlap -toimintoa hyödyntäen voidaan tarkastaa, mikä työkalu osuu kohteeseen. Kuokkaan liittyen tehdään cast to -toiminto eli katsotaan, onko päällekkäinen actor tosiaan kuokka ja, jos se on, niin muokataan maan materiaali oikeanlaiseksi ja muutetaan kyseisen istutuksen state-muuttuja Empty+Dry tai Empty+Wet sen mukaan, onko maata kasteltu. Ajatuksena oli, että state-muuttujien avulla voisi tallentaa edistymisen, jotta peliä voisi jatkaa samasta kohtaa eri pelikerralla. Samalla se toimii myös tarkistajana, joka tallentaa, mitkä toiminnot istutukselle on jo tehty (KUVIO 28).



KUVIO 28. Istutuksen overlap-eventit kuokalle ja pelaajan kädelle

Samalla tavalla toimivat myös siemenet ja kastelukannu. Ainoana poikkeuksena on Raycast, jolloin janan avulla tarkistetaan, onko alapuolella osumia. Jos on, niin lisätään tyhjään sijaintiin haluttu istutus staattisena mallina (KUVIO 29).



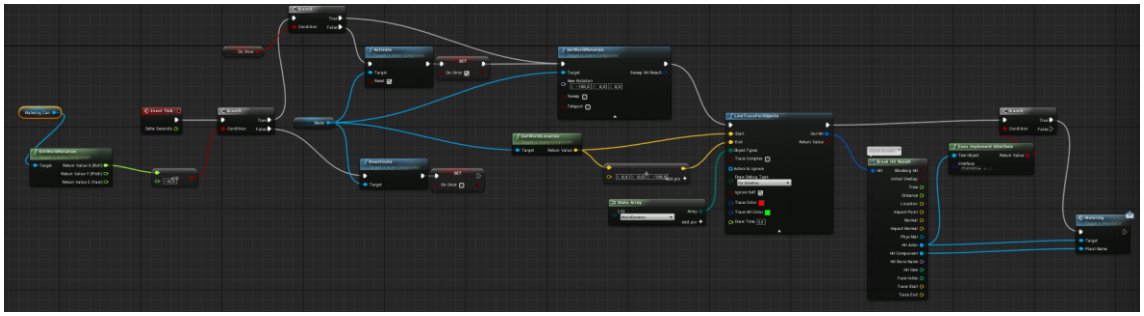
KUVIO 29. Funktio LineTrace, jolla tarkistetaan alapuolella oleva actor ja siirrytään Seeding-funktioon.

Pidemmälle edetessä hyvä idea olisi ollut tehdä istutukset erillisinä actoreina ja lisätä ne lapsiobjekteina istutukselle. Tällä tavoin olisi säästynyt pieneltä toistolta, joka jokaiselle istutukselle pitää tehdä.

Istutukset kastellaan kastelukannulla. Perinteisen kastelukannun mallintaminen sujui aika vaivattomasti. Kahvan sai tehtyä hyödyntäen Bezier-kurvia, jolla sai kurvin muodon mukaisen mallin tehtyä. Kun muoto oli valmis, muokattiin kahva meshiksi. Kastelukannun rungon ja putken sai yhdistettyä boolean-työkalulla. Valmis työ näytti kuvan mukaiselta (KUVIO 30).

KUVIO 30. Kastelukannun muoto

Ohjelmoinnin näkökulmasta kastelukannun täytyi kastella alla olevat istutukset. Tämä tapahtuu siten, että kun kulma on tietty, testataan LineTracella, onko alapuolella istutuksia. Jos alapuolella olevalla Actorilla on Plantable-niminen interface, suoritetaan Watering-funktio osuneelle istutukselle. Samalla kun kastelukannun kulma on tietty, tehdään näkyväksi kastelupartikkeliefekti (KUVIO 31).

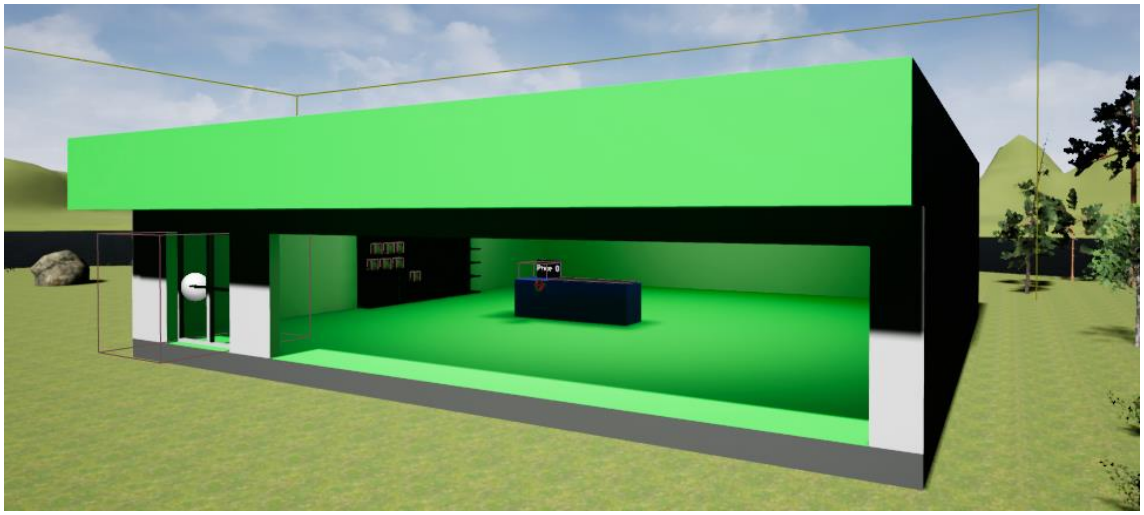


KUVIO 31. Kastelukannun toiminta

4.6 Kauppa ja ostaminen

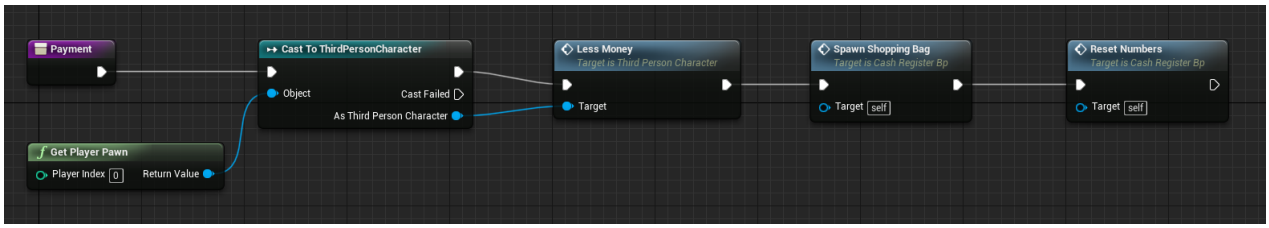
Ajatuksena oli käyttää kaupan pohjana käyttöliittymä-pohjaista järjestelmää. Pelin pelattavuutta ajatellen, oli kuitenkin parempi, että ostaminen tapahtuisi kuten oikeasti: pelaaja etsii tuotteen kaupan hyllyiltä, kuljettaa tuotteen kassalle ja maksaa sen. Tämä vaati enemmän työtä, sillä täytyi suunnitella toiminnot; miten tuotteet lisätään kaupan hyllyille, miten niitä kuljetetaan sekä miten maksaminen tapahtuu.

Kaupan luonti alkoi mallintamisesta. Ensin suunniteltiin kauppa yksinkertaisimmillaan, sillä tärkeintä oli saada se toimimaan ohjelmoinnin kannalta. Kaupan täytyi olla tarpeeksi tilava, jotta siellä sopii liikkumaan, ja hyllyjen täytyy olla pelaajan ulottuvilla virtuaalitodellisuudessa. Tähän hyvä keino oli hyödyntää oikeita mittoja (KUVIO 32).

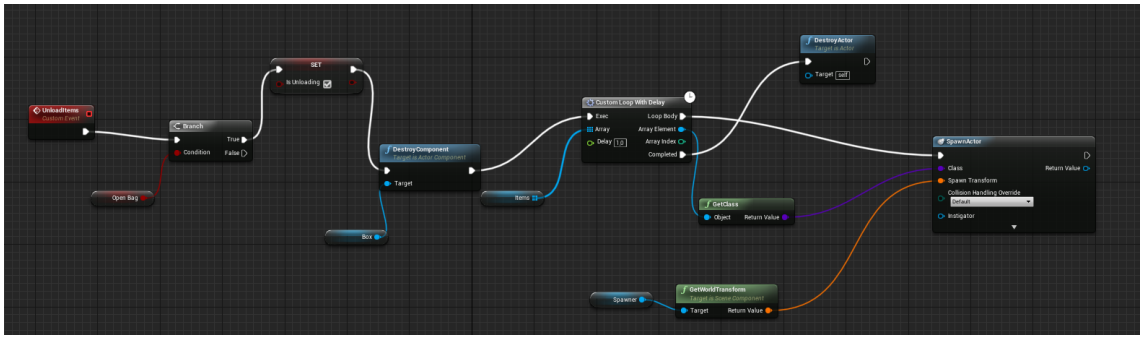


KUVIO 32. Kaupan 3d-malli. Sisällä erillisinä objekteina hyllyt ja kassa.

Ostaminen tapahtuu kantamalla ostokset kassan hihnalle. Tämän jälkeen hihna kuljettaa ostokset ja lopuksi siirtää ne listaan muistiin. Kun pelaaja pitää kättä maksupäätteellä kolme sekuntia, häneltä vähennetään tuotteiden arvo ja ostoskassi ilmestyy kassalle (KUVIO 33). Kun ostoskassi on päällekkäin kummankin käden törmäyskomponentin kanssa ja kun painetaan ohjaimen a-nappia, ostokset spawnaavat kassista pienellä viiveellä, minkä jälkeen ostoskassi poistetaan (KUVIO 34).



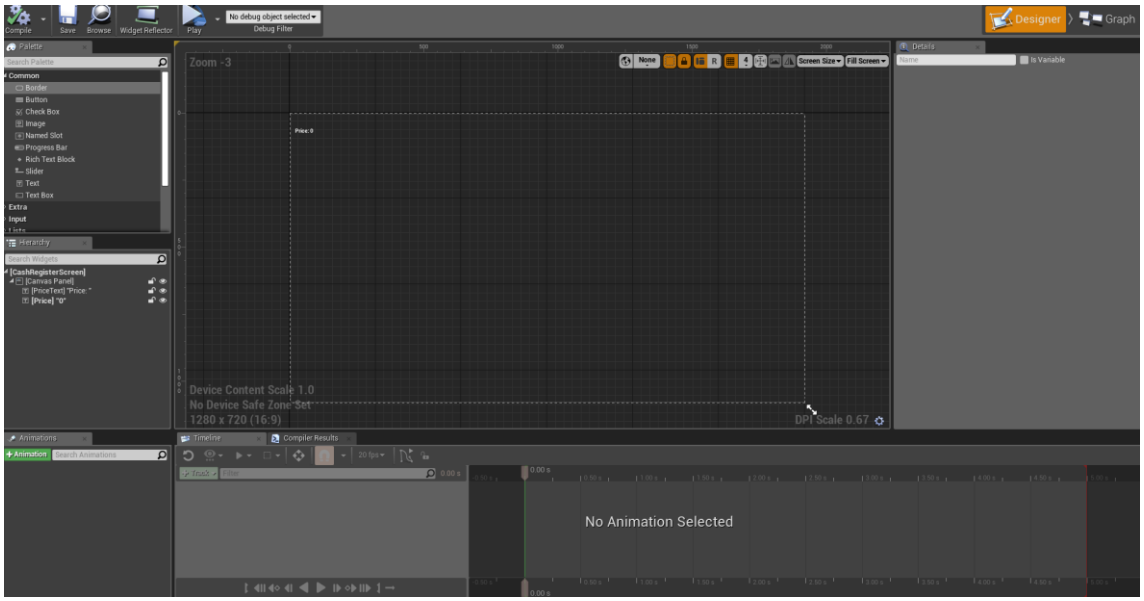
KUVIO 33. Kassa tarkistaa, että pelaajan käsi on kohteella, vähentää rahamäärän, spawnaa ostoskassin ja nollaa kassan lukemat.



KUVIO 34. Ostoskassin tuotteiden spawnaus kustomoidulla loopilla, jossa on viive jokaisella välillä.

4.7 Hud ja käyttöliittymä

Käyttöliittymälle ja hudille on oma blueprinttinsä, eli widget blueprint. Käyttöliittymä eroaa tavallisesta siten, että näkymässä voidaan muokata suoraan hudin ulkonäköä erilaisten osien avulla (KUVIO 35). Näkymässä on Designer ja Graph -osiot, joista Designerissä suunnitellaan ulkonäkö, kun taas Graph on visuaalista ohjelmointiliittymää, jossa esimerkiksi päivitetään lukuarvoja tiettyjen muuttujien mukaan. Jotta widgetin saa näkymään, täytyy lisätä widget-komponentti halutulle kohteelle ja valita widget-luokkaan haluttu blueprint. Sen jälkeen komponentti siirretään haluttuun kohtaan, jossa hudin halutaan näkyvän.

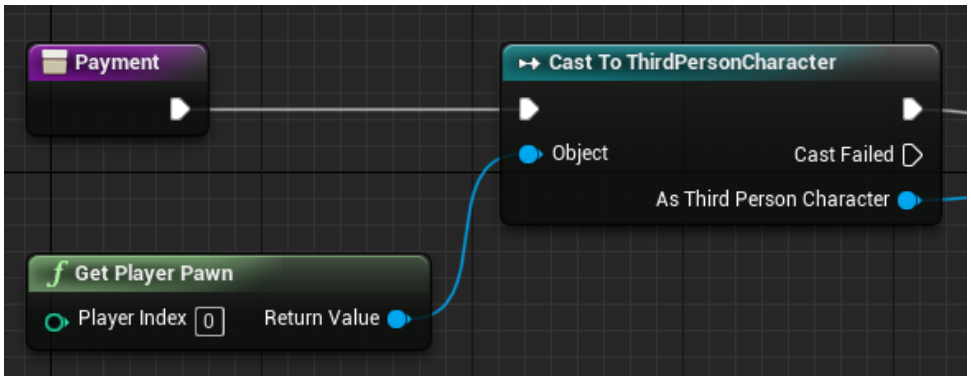


KUVIO 35. Widget-Blueprintin käyttöliittymä

4.8 Haasteet ja ongelmat

Haasteita oli ohjelmoinnin kannalta useita. Alussa haasteena oli löytää oikeat toiminnot. Unreal ehdottaa useita toimintoja, joista aloittelija ei välttämättä tiedä mitään. Joskus myös tavalliset toiminnot olivat erilaisella nimellä. Esimerkiksi If-toteamus löytyy sanalla ”Branch”.

Läpi projektin haastavinta oli luoda viittauksia toisiin actoreihin. Pelejä ohjelmoitaessa tavallista on, että eri objektit ovat keskenään tekemisissä ja aiheuttavat toisilleen tiettyjä toimintoja. Unrealissa viittaaminen toiseen onnistuu helpoiten osumien ja päällekkäisyyksien avulla. Toimintolohkosta on helppo yhdistää osuva ja osuttu kohde viittauksiin liittyen. Tilanteissa, joissa tällainen ei kuitenkaan ollut mahdollista, oli vaihtoehtona myös Cast to -toimintablokin käyttäminen. Cast to tarkistaa, että kohde todella on haluttu kohde. Sen jälkeen pääsee käsiksi halutun kohteen muuttujiin ja toimintoihin. Haastavaa oli saada oikea kohde objektiin, joka Cast to -toimintolohkoon täytyi yhdistää, jotta koodin sai toimimaan. Esimerkkinä voisi käyttää kauppajärjestelmän maksutapahtumaa: pelaajaan referenssin saa käyttämällä GetPlayerPawn-lohkoa (KUVIO 36). Vaatii miettimistä, mitä kautta objektireferenssin saa parhaiten.



KUVIO 36. Cast to funktiossa

Haasteena oli myös erilaisten työkalujen määrä Unreal Enginessä. Partikkeleiden luonnille on oma työkalu niin kuin käyttöliittymien ja valikoiden tekoon. Myös materiaalien sekä animaatioiden luontiin on omanlaiset työkalut. Nämä olivat esimerkkejä monista ja olivat haasteellisia rajallisen ajan kannalta: jokaisesta osa-alueesta kokonaisuudessaan saisi tehtyä opinnäytteen. Tämän vuoksi oli pakko rajoittaa joidenkin työkalujen käyttöä minimiin, jotta aihe ei kasvanut liian suureksi.

Haasteita oli myös widgettien kanssa, kuten siinä, miten pääsee käsiksi toisten actoreiden muuttujiin. Yhteyden widgetteihin sai kuitenkin hakemalla kaikki saman luokan widgetit ja kustomoituja eventtejä sitä kautta kutsuen. Widgettejä on mahdollista muokata hyvin paljon, mutta se olisi vaatinut oman aikansa, joten käyttö rajoittui vain muutamiin tapauksiin

Suoranaisia ongelmia projektissa ei ollut. Suurin ongelma liittyi Blenderin ja Unrealin väliseen yhtenäisyyteen. Mallien siirto Unrealiin onnistuu helposti ja ilman ongelmia. Asetuksista oli mahdollista valita myös materiaalien siirto mukana. Tämä toiminto ei kuitenkaan käytännössä toiminut. Tämä tarkoitti siis sitä, että Blenderissä ei kannattanut tehdä materiaaleja loppuun asti, sillä ne eivät siirtyneet Unrealiin. Käytännössä kannatti Blenderissä tehdä vain uv-mapitus kohteille ja tehdä materiaalit Unrealissa. Varsinkin kohteissa, joihin kuuluivat tekstuurien lisäksi normaali- ja tasotekstuurit pinnan epätasaisuudelle ja kiiltävyydelle, veivät viimeistelyssä oman aikansa.

Ongelmia seurasi myös siinä vaiheessa, kun pelimoottori päivittyi uudempaan. Projektissa käytettiin suurimmaksi osaksi Unrealin versiota 4.20. Projektin loppupuolella pelimoottori päivitettiin kuitenkin 4.21:een, ja vaikka päivitys toikin mukanaan uusia ominaisuuksia ja korjauksia, toi se myös joitain ongelmia: esimerkiksi pelaajan sijainnin nollaaminen onnistui ennen ilman, että sijainnin lisäksi nollattiin myös pelaajan korkeus, jolloin kamera siirtyi lattiatasoon.

5 POHDINTAA

Projektin ja opinnäytetyön aikana tuli esiin monia asioita ja pohdinnan kohteita. Samalla kehityin peliohjelmoinnissa ja virtuaalitodellisuuden ymmärtämisessä, sekä huomasin hyviä ja huonoja puolia pelimoottori Unrealissa. Aiheen kannalta tämä opinnäytetyö oli aika laaja, joten rajausta tapahtui varsinkin loppupuolella ajan puutteen vuoksi. Loppujen lopuksi tuloksena oli kuitenkin pelidemo, joka osoittaa, että olen kehittynyt alun tietämättömyydestä pidemmälle Unrealin käyttäjänä.

5.1 Perinteisen ja virtuaalitodellisuuden erot

Olen tehnyt toimeksiantoja ja muutenkin omien harrastusten sekä opintojen kautta tutustunut myös perinteisten 2d- ja 3d-pelien tekoon. Opinnäytteen kannalta myös virtuaalitodellisuus on hieman tuttua. Perinteisten ja virtuaalitodellisuuspelien välillä on useita eroja. Ehkäpä suurin ero on pelattavuudessa. VR-maailmassa seikkaileminen on kokemukseltaan lähes aina aistirikkaampaa kuin perinteisten pelien pelaaminen näytön välityksellä. Tämä tarkoittaa sitä, että niin pelimaailma kuin pelattavuuskin on monimutkaisempaa ja palkitsevampaa virtuaalitodellisuudessa.

Aistirikkaampi kokemus saadaan sillä hinnalla, että kehittäminen virtuaalilaitteistolle vaatii enemmän aikaa ja suunnittelua. Koska toiminnot ovat monimutkaisempia, tarkoittaa tämä myös ohjelmoinnin kannalta sitä, että täytyy miettiä, mitä toimintoja pelaaja voi tehdä, miten ne pystytään tekemään väärin ja miten estää huijaaminen. Vaikka pelaaja tuntee olevansa enemmän fyysisesti mukana pelimaailmassa, ei mikään kuitenkaan lopulta estä häntä menemästä seinien läpi tai kuljettamasta pelin esineitä paikkoihin, joihin niitä ei pitäisi viedä. Myöskin 3d-mallintajien ja ympäristönsuunnittelijoiden täytyy testata malleja tarkemmin, sillä virtuaalilasit päässä huomaa helpommin puutteita ja kuten sen, ovatko esineet skaalan suhteen realistisia. Puutteita huomaa erityisesti peleissä, jotka on myöhemmin muokattu VR-muotoon. Joskus työtä voi olla myös vähemmän: esimerkiksi animaatioita tarvitaan vähemmän, sillä VR-peleissä pelaaja suorittaa nämä toiminnot yleensä itse.

Vaikka VR-pelit ovat yleensä kokemukseltaan parempia, on myös poikkeuksia. Kaikki lajityypit eivät välttämättä sovi virtuaalitodellisuudelle. Esimerkiksi vauhdikkaammat tasohyppelyt tai pelit, joissa liikutaan paljon, voivat pidemmän päälle aiheuttaa pahoinvointia ja vaativat myös parempaa

kuntoa. Tällaisissa tapauksissa vaihtoehtona on yleensä tottuminen pelattavuuteen tai toisenlainen lähestymistapa. Monesti pelit, joissa liikutaan paljon, vaikuttavat myös negatiivisesti pelin koettavuuteen. Virtuaalitodellisuudesta puuttuu vielä nykyteknologialla jalkojen liike, joten liike tapahtuu yleensä ohjainten avulla tai joskus käsien liikkeellä. Aivot luulevat, että liikutaan, mutta jalat eivät kuitenkaan liiku. Tähän voi auttaa, jos pelaaja teeskentelee kävelevänsä pelin aikana liikuttamalla jalkoja.

Yksi suurimmista eroista perinteisen ja virtuaalitodellisuuden välillä liittyy nykyiseen teknologiaan. VR-pelit ovat tällä hetkellä vaativampia perinteisiin peleihin verrattuna, sillä ne vaativat korkeamman virkistystaajuuden ja pelit pitää piirtää kummallekin silmälle. Perinteiset pelit ovat tarkkuudeltaan huimasti parempia yltäen jopa moninkertaisiin Full Hd -tarkkuuksiin. Virtuaalilasit ovat yleisimmillään tarkkuudeltaan Full Hd -tasoa, mutta koska näytöt ovat niin lähellä silmää ja vaatimukset ovat kovemmat, eivät ne yllä tarkkuuden puolesta lainkaan perinteisten pelien tasolle.

Viihdepelit toimivat erinomaisesti, mutta näen VR-peleille erityistä potentiaalia myös hyötypeleissä. VR-laitteistolla onnistuu hyvin erityisesti silmä- ja käsikoordinaatio. Tätä voisi hyödyntää esimerkiksi lääkärien kouluttamisessa. Kyseessä voisi olla peli, jossa pelaaja suorittaa leikkauksen potilaalle. Muita aiheita voisi olla myös tilanteissa, joita olisi vaarallista suorittaa oikeissa ympäristöissä, kuten sukellusta, palon sammutusta tai sotatilanteita.

5.2 Henkilökohtainen kehittyminen

Alussa Unreal Enginen käyttö oli melkein täysin uutta. Projektin loppupuolella Unreal alkoi jo käytettävyyden puolesta tuntumaan tutummalta. Pelimoottori on helppokäyttöinen, kun sen oppii, ja visuaalinen ohjelmointi on mukavaa vaihtelua ainaisen kirjoittamisen sijaan. Projektin loppupuolella en ollut tietoinen läheskään kaikista toiminnoista Unrealissa, sillä pelimoottori on niin laaja. Erilaisia työkaluja löytyy moneen käyttöön, ja jokainen toimii hieman eri tavalla. Parhaiten kehityin mielestäni visuaalisessa ohjelmoinnissa. Kun toiminnan tajuaa, on hyvinkin loogista pohtia johtojen avulla, mitä tapahtuu seuraavaksi. Muuttujia ja toimintoja oppi käyttämään enemmän aina sen mukaan, kun tuli esiin tilanne, jossa uutta toimintoa käytettiin.

Virtuaalitodellisuuteen liittyen opin ymmärtämään sen monimutkaisuuden ja toisaalta sen yksinkertaisuudenkin. Vaikka toimintoja pitää pohtia enemmän, toimii virtuaalipeli kuitenkin suurimmalta

osin kuin tavallinen peli. Haasteita tulee yleensä pelaajan toimintoja pohtiessa, sillä se, mitä pelaaja saa pelimaailmassa tehdä, vaikuttaa paljon pelin mielekkyyteen ja koettavuuteen. Ei esimerkiksi ole enää järkeä, että pelaaja saa tavaran esineluetteloonsa pelkällä napin painalluksella, vaan esine pitää voida ottaa käteen ja laittaa talteen tavalla tai toisella.

3d-mallinnukseen liittyen tapahtui myös pientä kehittymistä. Eniten se oli vanhan palauttamista mieleen. 3d-malleja oli mielenkiintoista tutkia, kun niihin pääsi oikeasti sisään, kuten esimerkiksi talossa. Huomasin myös omiin mielenkiintoihin liittyen, että 3d-mallinnus kaiken kaikkiaan kiinnosti enemmän kuin ohjelmointi. Tarkoituksena oli keskittyä ohjelmointiin enemmän, mutta minusta tuntui, että mallinnusvaiheet olivat mielenkiintoisempia ja motivoivampia. Ohjelmointi liittyen malleihin, kuten esimerkiksi toiminnot ovien avaamiseen, olivat mielenkiintoisia saada toimimaan.

5.3 Mitä tekisin toisin

Sitä mukaan, kun kehittyi, huomasi varsinkin ohjelmointiin liittyen asioita, joita olisi voinut tehdä toisin. Projektin alussa tehdyt toiminnot olisi voinut ohjelmoida paremmin ja käyttää esimerkiksi samoja luokkia erilaisille objekteille. Näin kuitenkin tapahtuu melkein joka osa-alueella, kun aiheessa kehittyy.

Toisin alusta alkaen olisi voinut tehdä myös ohjelmistojen valinnassa. Unrealin sijaan olisi voinut valita esimerkiksi Unityn, joka olisi ollut joillain osa-alueilla parempi vaihtoehto. Visuaalisen ohjelmoinnin sijasta olisi kehittynyt yleisimmin käytetyissä kielissä, ja virtuaalitodellisuudelle olisi ollut enemmän lisäosia ilmaisella saatavilla. On vaikea sanoa, kummalla pelimoottorilla kehittäminen yksin olisi sujunut nopeammin, sillä toisaalta Unrealissa on ohjelmointi visuaalisella käyttöliittymällä nopeampaa, mutta Unityssä olisi saatavilla valmista ilmaista sisältöä paljon enemmän, jolloin ei tarvitsisi ohjelmoida kaikkea itse. Unrealin valitsin kuitenkin siitä syystä, että halusin oppia sen käytön.

LÄHTEET

Blender 2018 a. About. Viitattu 7.10.18,
https://www.blender.org/about/.

Blender 2018 b. Animation introduction. Viitattu 7.10.2018,
https://docs.blender.org/manual/en/latest/animation/introduction.html.

Blender 2018 c. Animation shape keys. Viitattu 7.10.2018,
https://docs.blender.org/manual/en/latest/animation/shape_keys/index.html.

Blender 2018 d. Rigging. Viitattu 7.10.2018,
https://docs.blender.org/manual/en/latest/rigging/index.html.

Carnall, B. 2016. Unreal Engine 4.x by example. Birmingham: Packt Publishing.

International Student 2018. What is game design. Viitattu 9.10.2018,
https://www.internationalstudent.com/study-game-design/what-is-game-design/.

Normal Map Online. Viitattu 19.11.2018,
http://cpetry.github.io/NormalMap-Online/.

Plowman, J. 2016. 3D game design with unreal engine 4 and blender. Birmingham: Packt Publishing.

Pluralsight 2014. What's the difference? A comparison of modeling for games and modeling for movies. Viitattu 7.10.2018,
https://www.pluralsight.com/blog/film-games/whats-the-difference-a-comparison-of-modeling-for-games-and-modeling-for-movies.

Unreal Engine 2018 a. Viitattu 27.9.2018,
https://www.unrealengine.com.

Unreal Engine 2018 b. Awards. Viitattu 20.11.2018,
https://www.unrealengine.com/en-US/awards.

Unreal Engine 2018 c. FBX Static Mesh Pipeline. Viitattu 12.9.2018,
https://docs.unrealengine.com/en-US/Engine/Content/FBX/StaticMeshes.

Unreal Engine 2018 d. Programming. Viitattu 27.9.2018,
https://docs.unrealengine.com/en-us/Programming.

Unreal Engine 2018 e. Virtual Reality Development. Viitattu 29.10.2018,
https://docs.unrealengine.com/en-us/Platforms/VR.

Wikipedia 2018 a. Blender(software). Viitattu 7.10.2018,
https://en.wikipedia.org/wiki/Blender_(software).

Wikipedia 2018 b. List of unreal engine games. Viitattu 19.9.2018,
https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games.

Wikipedia 2018 c. Oculus Rift. Viitattu 9.10.2018,
https://en.wikipedia.org/wiki/Oculus_Rift.

Youtube 2018 a. Marco Ghislanzoni. Viitattu 29.10.2018,
https://www.youtube.com/user/marcoghislanzoni.

Youtube 2018 b. UnrealEngine. Viitattu 29.10.2018,
https://www.youtube.com/user/UnrealDevelopmentKit.

Game Design Document - Farming game

Mikke Penttilä

Introduction

Aloita oman farmisi kasvattaminen jo tänään. Unohda nykyaikaiset teho kasvatukset erilaisilla isoilla laitteilla. Kasvata kasveja ja puita, pitäen niistä huolta: kastele, hoida ja myy. Osta, paranna ja kehitä maatilaa. Oma farmisi tulee kehittymään ajan myötä. Farmin lisäksi voit omistaa ajoneuvoja, joilla voit kuljettaa erilaista tavaraa pelimaailman ympäri. Farmin lähellä on kylä, josta voi ostaa erilaisia tarvikkeita eri tarpeisiin. Odottaessa sadon valmistumista voi seikkailla, löytää aarteita ja muuta, sekä esimerkiksi kaataa puita myyntiin tai rakennusmateriaaliksi. Immersiivinen, rento kokemus odottaa lasien takana, maanviljelijä.

Gameplay Description

Pelaaja aloittaa matkansa mökistään. Peilin kautta hahmo muokataan sopivaksi ja sen jälkeen astutaan ulkomaailmaan. Siellä pelaajaa odottaa jo pieni valmis erä kerättäviä kasveja, sekä siemeniä istutettavaksi. Pelaajalla on myös aloitus ajoneuvo, jolla hän voi kuljettaa satonsa myyntiin ja ostaa tarvikkeita kylästä. Pelaajalla on myös perustyökalut maatilan hoitamiseen ja pelin edetessä voi ostaa uusia/päivitettyjä välineitä.

Myymällä valmista hyvää satoa, pelaaja ansaitsee rahaa, jolla voi ostaa uusia siemeniä, sekä muita tarvikkeita maatilan päivittämiseen. Myös pelimaailmasta löytyy sekalainen joukko erilaista löydettävää ja koettavaa. Yön tullessa pelaajan täytyy mennä mökkiinsä nukkumaan, jolloin peli myös tallentuu.

Function 1

Perusliikkuminen ja esineiden nappaaminen ja interaktiivisuus. Esim: pelaaja liikkuu, voi aukaista oven, pitää kädessä kastelukannua, siirtää objektia ja painaa nappeja yms.

Function 2

Kauppa myynti ja ostaminen. Rahan saaminen ja menettäminen. Tallennus. Eriarvoiset kasvit.

Function 3

Farmin kehittäminen. Uusien objektien ostaminen ja liittäminen osaksi farmia. Esim pelto, aita, huonekalut. Tallentaminen.

Function 4

Ajoneuvo jota voi ohjata ja jonka kyytiin asettaa esineitä.

Function 5

Oman hahmon näkeminen ja editointi

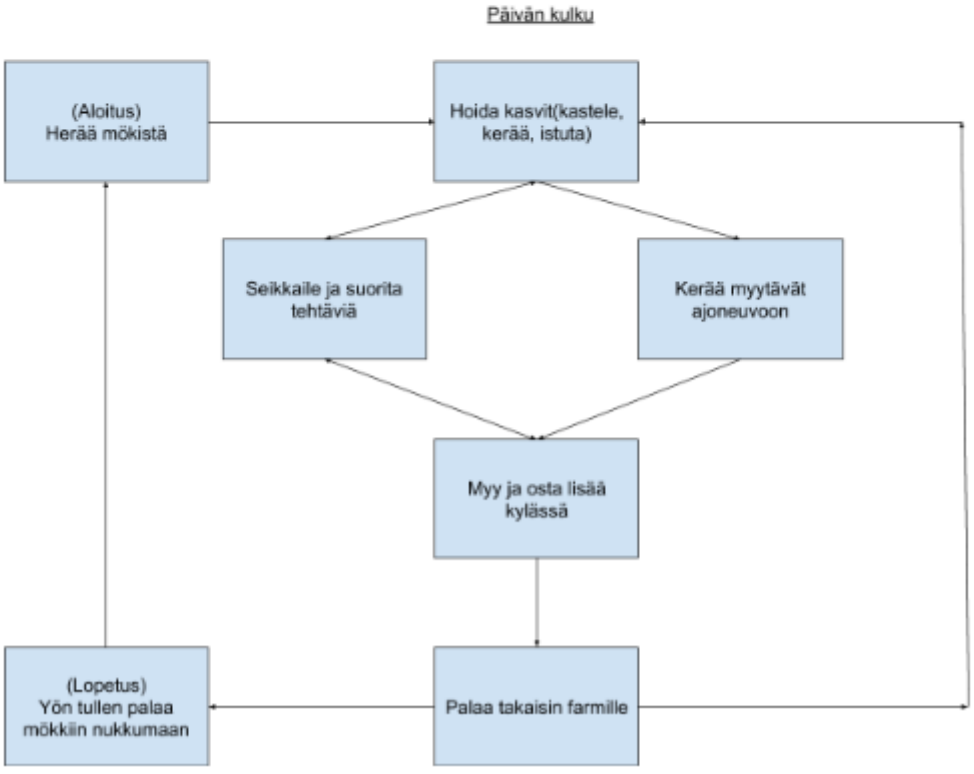
Function 6

Objektien toiminnan ohjelmointi. Kastelukannu kastelee ja niin edelleen.

Function 7

Valikot ja niiden säätäminen vr:lle sopiviksi.

Basic Game Flow



Character Bios

Pelihahmo mahdollisuuksien mukaan näkyvillä. Mahdollisuuksien mukaan kustomoitava, mutta riippuu ajasta ja taidoista.

Game World

Pelimaailmaan sisältyy pelaajan farmi, joka muuttuu ajan myötä, sekä lähistöllä oleva kylä, josta eri kaupoista voi ostaa erilaisia välineitä ja tavaroita. Ympärillä jonkinlaista metsää ja vuoristoa, jokia ja puroja.

User Interface

Käyttöliittymä vr-ohjaimille sopivaksi, napista aukeaa valikko tietyille toiminnoille ja aloitusruutu, josta peli aloitetaan. Nähtäväksi jää tuleeko peliin esim vöitä yms muita vr-todellisuuden työkaluja, joihin voi ripustaa esim työkaluja ja siemenpusseja yms.

Artistic Style

Realistisen oloista, mutta mukavan värikästä. Maaseutu-tunnelmaa.