VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jesse Sipiläinen

VR-sovellus Unreal Engine 5:llä

Tekniikka
2024

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Jesse Sipiläinen
Opinnäytetyön nimi	VR-sovellus Unreal Engine 5:llä
Vuosi	2024
Kieli	suomi
Sivumäärä	53 + 1 liitettä
Ohjaaja	Harri Lehtinen

Tämän opinnäytetyön VR-sovellus luotiin Unreal Engine 5.1 -pelimoottorilla. Tässä dokumentissa käydään läpi asetukset, ohjelmat ja liitännäiset, joita käytettiin sovelluksen onnistuneeseen tekemiseen, kääntämiseen ja siirtämiseen Oculus (Meta) Quest 2 -laseille. Pääpaino kuitenkin on itse sovellus ja sen toiminta sekä esitys käytetyistä Unreal Engine 5.1 -ominaisuuksista sovelluksessa.

Yritykseltä, johon tämä sovellus liittyi, on lupa käyttää kuvia opinnäytetyössä sekä näyttää sovellus ohjaajille ja opettajille, jotka ovat mukana arvioinnissa. Liitteenä työhön sisältyy myös manuaali, jota seuraamalla on mahdollista luoda samankaltainen sovellus tai muokata alkuperäistä sovellusta. Liite on salassa pidettävä, joten sitä ei jaeta opinnäytetyön mukana, vaan se on toiminut tukena opinnäytetyötä tehdessä.

Unreal Engine 5.1 oli loistava työkalu virtuaalisen ympäristön rakentamiseen ja testaamiseen. Laitevaatimukset sovelluksen tehokkaalle käytölle ovat melko vaativat, mutta henkilökohtainen tietokone, jolla sovellus kehitettiin, toimi tarpeeksi tehokkaasti sujuvaan ohjelman rakentamiseen. Sovelluksessa on kaikki vaaditut ominaisuudet sekä joitain lisättyjä ominaisuuksia, jotka ovat enemmänkin testivaiheessa kuin valmiita. Silti sovelluksen lopullinen kunto miellytti työn antajaa ja sen katsottiin täyttävän tehtävälle asetetut kriteerit.

Sovelluksessa käytettyihin ratkaisuihin vaikuttaa myös äärimmäisen tiukka aikataulu demovaiheen jälkeen. Jotkin ratkaisut tehtiin puhtaasti ajan säästämiseksi tai käyttäen valmiiksi tiedettyjä käytäntöjä uusien etsimisen sijaan. Laitteiden mallien optimoiminen oli myös pakollinen, mutta valmiiksi saadut mallit rajoittavat jonkin verran optimoimista. Mittava manuaalinen optimointi olisi epäkäytännöllistä, eikä oikeastaan kuulu edes annetun työn tarkoitukseen.

Avainsanat	tietotekniikka, virtuaalitodellisuus, unreal engine 5, ohjelmistokehitys

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Jesse Sipiläinen
Title	VR application made with Unreal Engine 5
Year	2024
Language	Finnish
Pages	53 + 1 Appendices
Name of Supervisor	First name last name

This thesis project´s VR application was created using Unreal Engine 5.1 game engine. This document goes through settings, programs and plugins that were used to successfully create, compile and transfer to Oculus (Meta) Quest 2 glasses. Main point (of this document) is application itself, behaviour of application and showing of used features from Unreal Engine 5.1.

There is approval from the company (regarding this application) to use pictures in thesis and show the application to directors and teachers that are involved in valuation of this thesis. Work include manual as an appendix, what can be followed to produce same kind of application or edit original application. Manual is confidential and it is not included in final thesis, it worked as a support to make this thesis.

Unreal Engine 5.1 is great tool for creating and testing virtual environment. System requirements to use software (Unreal Engine 5.1) efficiently is quite demanding but personal computer that was used to create application, worked efficiently enough to create application fluently. All required features are included in application and few other features that are more in test phase than ready. Applications final form pleased task giver and it was deemed to meet the criteria set for it.

Resolutions used in application was also affected by extremely hard schedule after the demo phase. Optimization of the machine models was also mandatory but already constructed models limited some optimization and extensive manual optimising would be unpractical and wouldn’t even belong in frames of given work.

Keywords Information technology, virtual reality, Unreal Engine 5, Software development.

SISÄLLYS

SANASTO

Actor	Unreal Enginen tasoon lisätty mikä tahansa olio.
Metaverse	Virtuaalinen todellisuus usealle käyttäjälle, jota tarkastellaan esimerkiksi virtuaalilaseilla. Määritteitä on useita ja yhtä oikeaa metaversea ei tällä hetkellä ole.
Unreal Engine/UE	Pelimoottori, jolla voi luoda lähes kaikenlaista 3D sisältöä.
VR	Virtuaalitodellisuus.
Pivot point	Tukipiste, jonka ympärillä actor esimerkiksi kääntyy.
Konemallit/laitteet	Tässä työssä viitataan teollisuuden käyttämiin laitteisiin, joita työnantaja myy. Konemallit/laitteet ovat suunnitteluosaston tiedostoista tulevat oikeat tekniset 3D-piirustukset.
Mesh	Kuvaa 3D-olion muotoa. Käytännössä jokainen näkyvä actori työssä on Mesh (vielä tarkemmin static mesh).
Node	Tässä työssä Unreal Enginen visuaalisen koodaamiseen liittyvät oliot, jotka suorittavat erilaisia funktioita.
Taso/Level	Tasot sisältävät kaiken käyttäjälle kerralla näytettävän ja käytettävän sisällön.

KUVA JA TAULUKKOLUETTELO

LIITELUETTELO

1 TYÖTEHTÄVÄ JA OPINNÄYTETYÖ

Idea opinnäytetyöhön syntyi koulutehtävästä ja yrityksen antamasta tehtävästä. Alkuperäisenä tarkoituksena oli saada Oculus (Meta) Quest 2 -laseilla näkyvä metaverse-ympäristö, jossa suunnitteluosaston valmiita laitemalleja kyetään tarkastelemaan virtuaalisessa ympäristössä. Koulutehtävän tuloksena oli alkeellinen malli siitä, mitä oli mahdollista saavuttaa, mutta metaversen todellinen tila ajoi jo tässä vaiheessa tehtävää uusille urille. Metaversen tila oli niin alkuvaiheessa, että sopivaa alustaa työlle ei löytynyt (Tucci & Needle, 2023). Koulutehtävä ja tässä opinnäytetyössä käytetty tehtävä ovat kuitenkin kaksi täysin eri työtä ja niissä on käytetty myös eri versiota Unreal Enginestä, joten niitä ei voi täysin verrata keskenään.

VR-sovellus, jota tässä opinnäytetyössä käsitellään, alkoi valmistua työsuhteen aikana. Aluksi tehtävänä oli vielä selvittää, mikä on käytännöllisesti mahdollista Oculus (Meta) Quest 2 -laseilla ja tehdä esitettävä demo ennen työsuhteen päättymistä (tässä vaiheessa kyseessä oli toissijainen tehtävä). Demovaiheen vaatimuksina oli toimiva useamman konemallin tarkasteleminen, liikkuminen ympäristössä ja yleinen toimivuus.

Useat esivaatimukset ohjelmiston kohdalla veivät hieman aikaa itse demon rakentamiselta, mutta muutaman yrityksen jälkeen kaikki toimi niin kuin kuului, ja demon rakennus muun työn ohella jatkui.

Demon esittelyn jälkeen työsuhde jatkui ja VR-sovellus sekä manuaali sen tekemisestä tuli primääriksi tehtäväksi. Tässä vaiheessa sovelluksen kehitys oli todella aktiivista ja nopeatempoista. Koko prosessi sujuisi niin, että suunnitteluosasto tuottaa laitemallin 3D-piirustukset, tiedosto ladataan Unreal Engineen Datasmithin avulla ja malli asetetaan valmiiksi rakennettuun tyhjään tasoon ja sovellus päivitetään.

Taulukossa on pelkistettynä työn vaiheet, vaatimukset ja kaikki esivaatimukset. Esivaatimukset käydään läpi taulukon jälkeen, vaikka ne eivät ole olennainen osa itse työtä.

Taulukko 1. Työn vaiheet, vaatimukset ja esivaatimukset.

Työn vaiheet	**Työn vaatimukset**	**Esivaatimukset**
– Esivaatimuksien selvittäminen. – Vaadittavien ohjelmistojen, liitännäisten ja kirjastojen asentaminen. – Demon suunnittelu. – Demon luonti. – Demon esittäminen esimiehelle. – Käytettävän VR-Sovelluksen luominen Oculus (Meta) Quest 2 -laseille. – Ehdotettujen ominaisuuksien lisääminen applikaatioon. – Manuaalin kirjoittaminen. – Testaaminen (kaikissa työn vaiheissa). – Sovelluksen esittäminen esimiehelle ja muille työntekijöille.	– Mahdollisuuksien selvittäminen. – VR-sovellus Oculus (Meta) Quest 2 -laseille. Sovelluksen tulee toimia sujuvasti pelkillä laseilla, eikä esimerkiksi tietokoneesta ajettuna. – Demoversio, jossa on useampi kone tarkkailtavana ja käyttäjä kykenee pyörittämään koneita, sekä liikkumaan virtuaaliympäristössä. – Suunnittelun laitemallit pystytään siirtämään VR-ympäristöön mahdollisimman pienellä vaivalla. – Helposti käytettävä versio, jotka kyetään esittämään laseista mm. messuympäristössä. – Lisää ominaisuuksia: liikkuminen thumbstickeistä (eteen, taakse, sivuille, käännä katsetta), kolme kuljettavaa korkeutta laitteiden tarkastelun helpottamiseksi, laitteiden oikeat nimet, valikko mistä valita koneet niiden oikealla nimellä, taustaäänen lisääminen kenttään ja äänen lisääminen koneen pyöritykseen. – Bugien korjaamista ja ominaisuuksien muokkaamista. – Ympäristön muokkaamista (toissijainen tehtävä, joten niin paljon kuin aikaa muusta jäi). – Manuaali.	– Unreal Engine 5.1. – liitännäiset Unreal Engineen: Datasmith, Bridge ja Modeling Tools Editor Mode. – Android Studio 4.0 (Android SDK Platforms: 10.0, API 33.0. SDK Tools Android SDK build - Tools 34 rc3: 33.0.2, 30.0.3, 29.0.2, 28.0.3. NDK (Side by Side): 25.2.x, 25.1.x,21,4.x,21.1.x. CMake: 3.10.2. Android Emulator: 32.1.11. Android SDK Platform-Tools: 34.0.0. Google USB Driver. Intell x86 Emulator Accelerator (HAXAM installer) (huom. luultavasti toimii useammallakin kokoonpanolla, mutta näitä käytettiin työn tekemiseen). – Visual Studio 2022 (koneella, jolla työ on tehty, on myös VS 2019) ja Workloads osiosta: .Net desktop development, Desktop development with C++, Mobile development with C++, Game development with C++ (installation details: Unreal Engine installer, Android IDE support for Unreal Engine). Individual components osiosta: .Net Core 3.1 Runtime ja .Net 5.0 Runtime. – Java jdk 11.0.3. – Side Quest. (Unreal Enginen asetukset esitetään alempana).

Esivaatimuksena on suuri määrä ohjelmia, kirjastoja ja liitännäisiä. Tässä työssä käytetystä ohjelmien, liitännäisten ja asetusten kokoonpanosta saattaa olla muitakin toimivia ratkaisuja, mutta suuren määrän vuoksi ensimmäinen toimiva kokoonpano on pysynyt sellaisenaan (esim. The Unreal Takeaway, 2023). Tärkeimmät sovellukset ovat kuitenkin Unreal Engine 5.1 (ohjelma, jolla tämä työ luotiin), Android Studio 4.0 (tarvitaan VR-lasien ja Unreal Enginen yhdessä toimimisen vuoksi, sekä sovelluksen rakentamiseen Oculus-laseille), sekä Visual Studio 2022 (sisältää tarvittavia kirjastoja ainakin projektin kääntämisvaiheeseen). Näillä ohjelmilla on mahdollista luoda täysin toimiva virtuaalitodellisuussovellus Oculus (Meta) Quest 2 laseille.

Tärkeimmät liitännäiset työn kannalta ovat Datasmith, joka tuo itse koneiden mallit suunnitteluosaston tiedostoista Unreal Enginen puolelle (Datasmith – Unreal Engine, n.d.). Bridge (Bridge by Quixel – Unreal Engine, n.d.) ympäristön ja aulan luomiseen sekä immersion lisäämiseksi. Modeling Tools Editor Mode helpottaa kentässä olevan actorin pivot pointin siirtämistä ja täten helpottaa actorin kääntämistä sujuvasti.

Opinnäytetyön painopiste kuitenkin on itse VR-sovellus sekä Unreal Engine 5.1:ssä käytetyt ominaisuudet ja siitä syystä työn vaiheet, vaatimukset ja esivaatimukset on esitetty vain lyhyesti ja suppeasti. Liitteenä olevassa manuaalissa on esitetty joitain näistä asioista tarkemmin ja hieman käytännöllisemmin.

Opinnäytetyö etenee sovelluksen kehitysjärjestyksessä, pois lukien asetukset ja testaaminen. Asetukset käydään läpi ensimmäisessä osiossa missä projekti luodaan ja tarvittavat liitännäiset ladataan. Testaamisesta kerrotaan luvussa kahdeksan. Osa opinnäytetyössä näkyvistä kuvista on otettu työn aikana ja osa erikseen opinnäytetyötä varten.

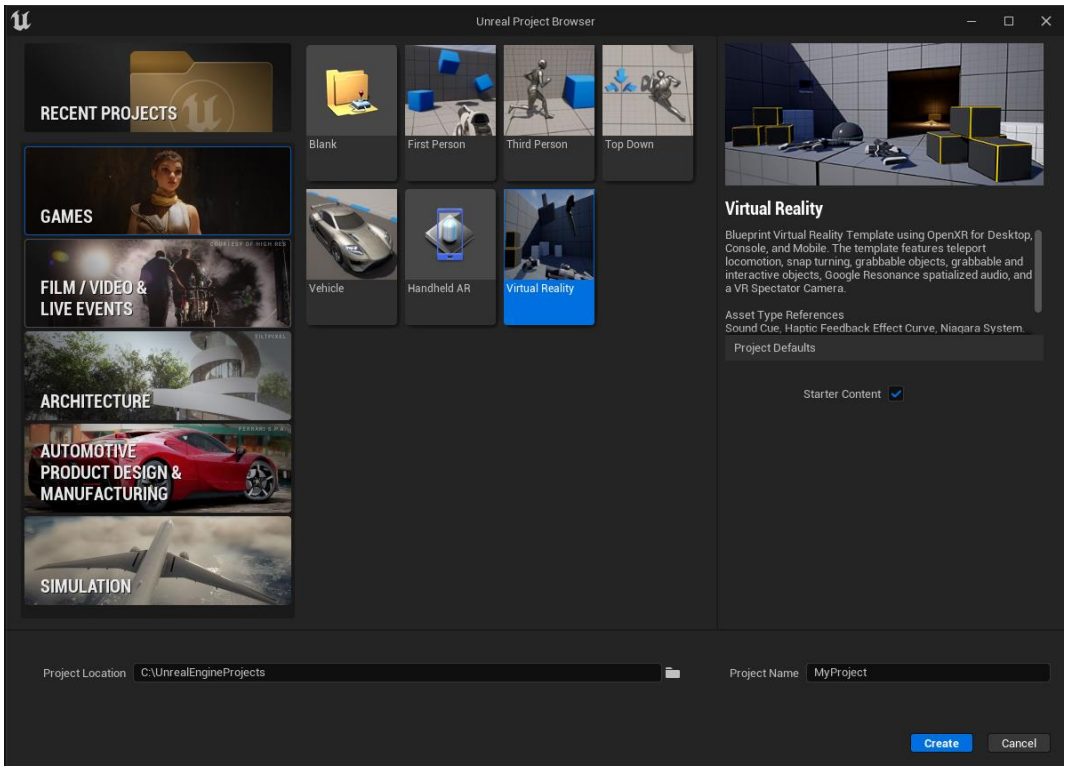
Käyttötarkoituksia tällaiselle ohjelmistolle voisi olla useita, mutta kenties messuympäristössä markkinoitujen koneiden tarkastelu olisi luonnollisin. Asiakkaalle

on helpompaa näyttää todellisen kokoiset ja näköiset virtuaaliset koneet kuin oikeat suuret ja raskaat koneet. Ohjelmistoa ei tarvitse myöskään rajoittaa koneiden määrän vuoksi toisin kuin esitystiloja.

2 UUDEN PROJEKTIN LUONTI

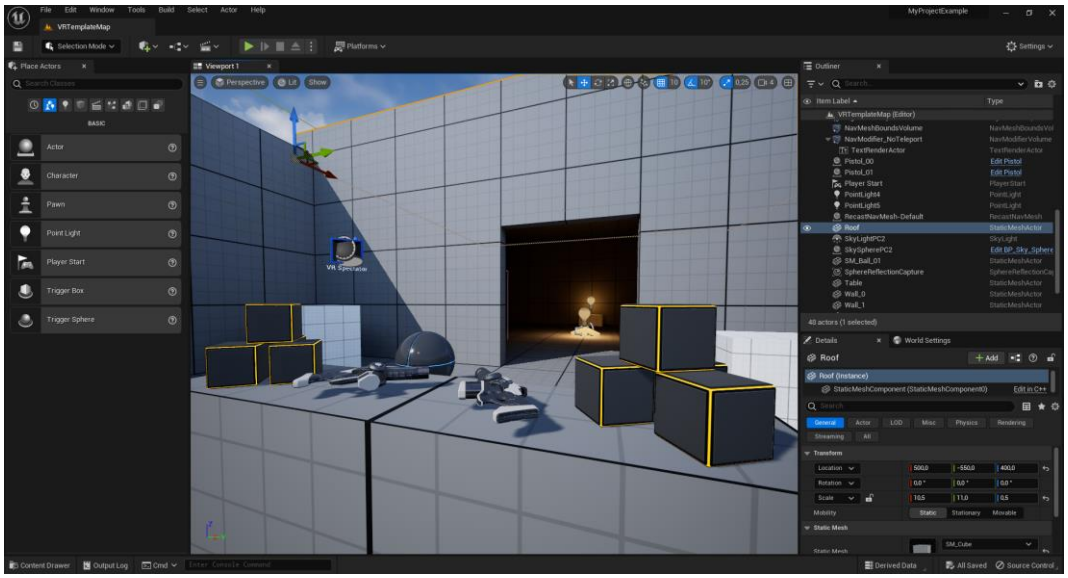
Aloitettaessa uutta projektia Unreal Enginellä on ensin mietittävä, mikä on projektin tyyppi. Unreal Engine tarjoaa valmiita pohjia erilaisille ohjelmistoille, joista on nopeaa ja yksinkertaista lähteä kehittämään halutunlaista ohjelmaa. Päävalinnat ovat pelit, elokuva/video ja livetapahtumat, autojen tuotesuunnittelu ja valmistus sekä simulaatiot. Valittava pohja (template) vaikuttaa Unreal Enginen projektissa valmiina oleviin asetuksiin ja liitännäisiin (Unreal Engine Templates Reference, n.d.).

Tässä opinnäytetyössä käsiteltävä sovellus on lähimpänä VR-peliä, joten valitaan Games ja pohjaksi (template) Virtual Reality. Valitaan haluttu kohdekansio, usein halutaan myös starter content mukaan projektiin, jolloin se valitaan valintaruudusta ja sen jälkeen painetaan create (Kuva 1).



Kuva 1. Unreal Project Browser ja templatet.

Projektin luominen kestää jonkin verran kauemmin kuin projektin avaaminen jatkossa. Seuraavaksi aukeaa Unreal Engine ja luotu projekti. Aluksi katseluikkunassa (jatkossa käytetään englanninkielistä nimitystä viewport) näkyvä taso on täynnä starter content -sisältöä (Kuva 2). Nämä on helppo poistaa valitsemalla actorit joko viewportista tai outlinerista ja painamalla delete.



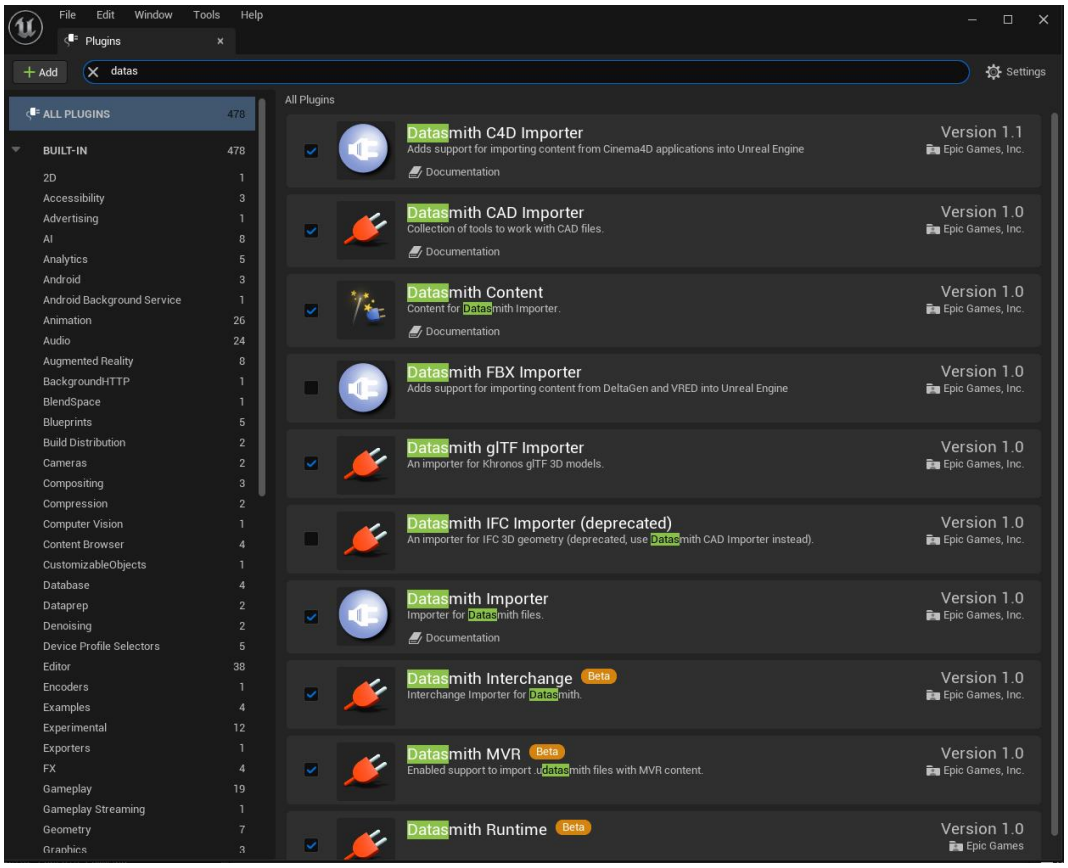
Kuva 2. Unreal Enginen aloitusnäkymä. Viewport ja outliner.

Templatessa olevat asetukset ja ominaisuudet (mm. VR pawn, menu ja näppäin asetukset) ovat syy sille, miksi templatea käyttämällä säästää runsaasti aikaa. Actoreiden poistaminen käy hyvin nopeasti verrattuna siihen, kuinka paljon valmiita ominaisuuksia template antaa pohjaksi.

2.1 Liitännäiset

Liitännäiset valitaan Plugin Browserista, joka löytyy yläpalkin Edit -osion alta nimellä plugins. Tässä työssä käytetyt kolme tiettyä liitännäistä helpottavat tai mahdollistavat halutut ominaisuudet. Näistä tärkein on Datasmith, joka tuo suunnitteluohjelmistoilla tuotettuja tiedostoja suoraan Unreal Engineen muokattavaksi. Tässä työssä oli useampia testattuja suunnitteluohjelmiston tiedostotyyppejä, minkä vuoksi useampi Datasmith plugin on valittuna (Kuva 3). Jt- ja stp-tiedostot

kyettiin tuomaan onnistuneesti, jos suunnitteluosaston tiedostossa oli osien sijaintidata kunnossa.



Kuva 3. Käytetyt Datasmith liitännäiset (kaikki eivät välttämättömiä).

Quixel Bridge plugin saattaa olla käynnistettäessä jo valittuna päälle. Quixel Bridge antaa nopean keinon hakea valmiita 3D-malleja ja tekstuureita Quixel Bridgen valmiista kirjastosta. Näitä malleja käytettiin elävöittämään tehdashallia ja varsinkin aulaa. Halutut mallit tai tekstuurit voi ladata suoraan Quixel Bridgestä Unreal Enginen käynnissä olevaan projektiin. Aulassa käytettiin myös Unreal Enginen marketplacen sisältöä (Unreal Engine Marketplace, n.d.).

Modeling Tools Editor Mode antaa suuren määrän työkaluja muokata meshiä suoraan viewportissa. Tässä työssä ei ollut tarvetta muille kuin pivot pointin siirtämi-

selle. Koneen mallin kääntyminen toimii sen keskipisteen mukaan ja joissain malleissa keskipiste ei ollut keskellä mallia. Tämä sai koneen pyörimään epätasaisesti ja vaikutti huomattavasti kokemukseen. Pivot pointin siirtäminen manuaalisesti ilman liitännäistä olisi ollut huomattavan paljon työläämpää.

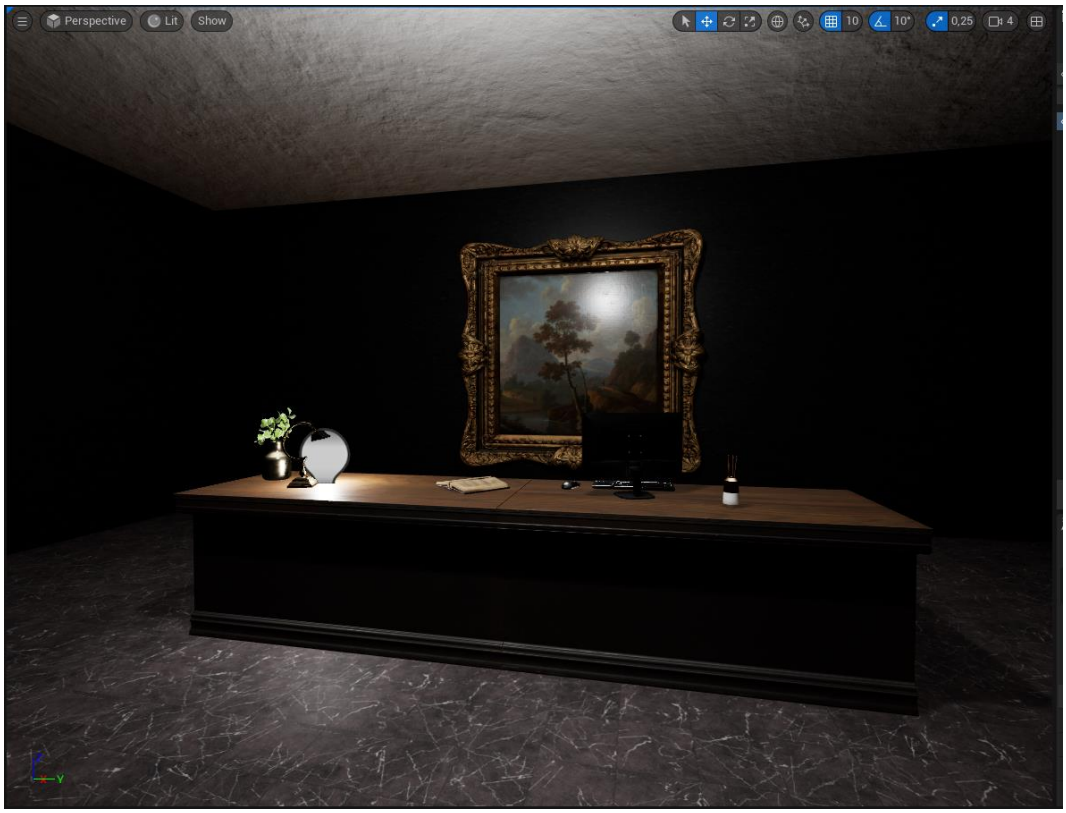
Nämä käytetyt liitännäiset antavat hyvän kuvan siitä, kuinka Unreal Enginen liitännäiset säästävät aikaa, mahdollistavat uusia toimintoja ja helpottavat yleistä työskentelyä. Unreal Engine Marketplacessa on myös käyttäjien tekemiä liitännäisiä, mutta useat niistä ovat maksullisia. Liitännäiset ovat ehdottoman tärkeä osa sujuvaa ohjelmistokehitystä Unreal Enginellä.

2.2 Unreal Enginen asetukset

Luodakseen VR sovelluksen Oculus (Meta) Quest 2 -laseille, on ehdottoman tärkeää muokata joitain asetuksia Unreal Enginessä. Unreal Engine tarvitsee myös java-polun, polun Androidin SDK/NDK ja tiedon SDK ja NDK API tasosta (Ks. Liite 1: kuva 10), voidakseen pakata Android-sovelluksen kohdelaitteelle. Muut tärkeimmät asetukset löytyvät project settings -osion alta kohdasta Android ja nämä tärkeimmät ovat minimum SDK version ja target SDK version. On tärkeää laittaa minimum SDK version 23 koska Unreal Engine 5.x ei pakkaa version 23 alapuolelle. Package game data inside apk, Enable FullScreen immersive on KitKat and above devices, support arm64 ja support Vulcan on myös tärkeää olla päällä (tarkemmat kuvat käytetyistä asetuksista löytyvät Liite 1: kuvat 10,11 ja 12). Asteusten jälkeen on muistettava myös painaa saman asetusvalikon yläosasta löytyvää accept SDK license.

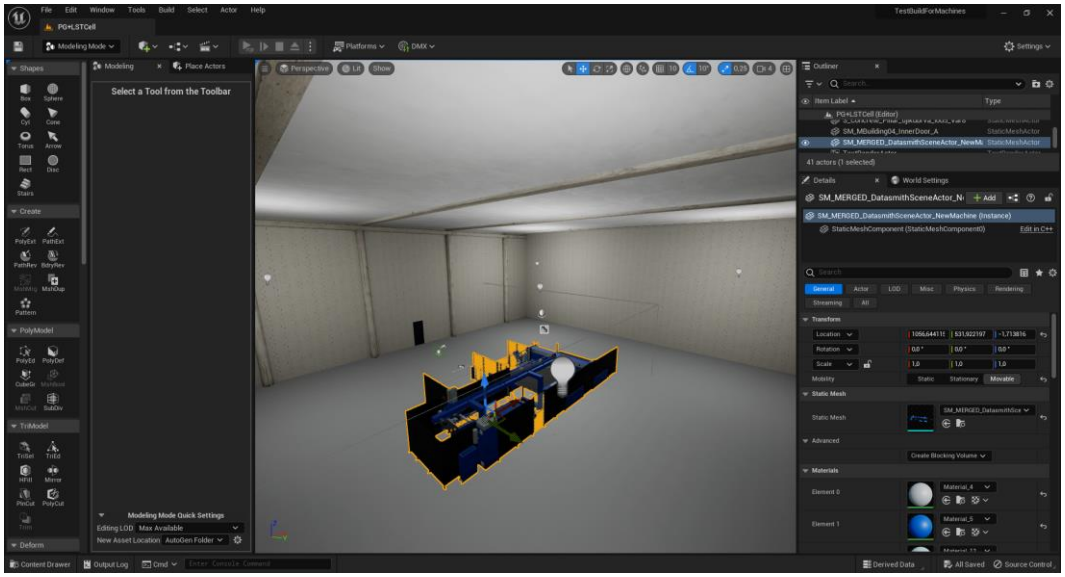
2.3 Tasosuunnittelu työhön sopivalla tavalla

Suunnittelu on aina tärkeä osa mitä tahansa projektia tai työtä. Tässä työssä tärkeimmät asiat suunnittelun kannalta olivat toimivuus, helppokäyttöisyys ja koneiden laatu. Syy sille, että ensimmäinen taso on aula eikä koneen esittelyhuone, on antaa käyttäjälle kosketus ohjelmaan ilman tärkeää tekemistä. Aulassa käyttäjän on helppo testata ohjelman käyttämistä. Se lisää myös hyvin immersiota, ja seinällä oleva yrityksen logo muistuttaa kuitenkin virallisesta asiasta (yrityksen logo ei ole näkyvissä tässä opinnäytetyössä). Aulan rakentaminen ei kestänyt kovin pitkää aikaa, kiitos Bridgestä tuotujen mallien, sekä Unreal Engine Marketplacesta ladatun sisällön (Kuva 4). Myöhemmin koneen esittelyihin tarkoitetun tason suunnitteluun vaikutti myös aikataulu.



Kuva 4. Yksi osa aulasta.

Koneen sisältävän tason täytyi olla tarpeeksi tilava suuremmillekin koneille, hyvin valaistu ja hallimainen. Prioriteettina kuitenkin oli käytettävyys ja itse konemallit, joten ympäristö jäi hyvin pelkistetyksi ja yksinkertaiseksi (Kuva 5). Valaistuksen optimointi suureen huoneeseen oli hieman haastavaa, joten siihen käytettiin enemmän aikaa, kuin huoneen visuaalisiin yksityiskohtiin, jotka eivät olleet prioriteettina missään vaiheessa.



Kuva 5. Koneiden esittelyyn tarkoitettu halli.

Oculus (Meta) Quest 2 -lasien teho pakotti myös tasojen sisällä olevan vain yhden koneen. Konemalleissa oli tuodessa tuhansia osia, yleensä yli kymmenentuhatta, ja se vaikutti suuresti ohjelman toimivuuteen. Edes kaksi konetta tasossa ei toiminut sujuvasti, joten tasosuunnittelussa täytyi ottaa tämä huomioon.

Koneiden laatu pysyi todella hyvänä, ja suunnittelun lähettämät mallit tulivat sellaisenaan Datasmithin avulla. Saadut mallit olivat yksinkertaistettuja, mikä näkyi mm. joidenkin materiaalien fuusioitumisena ja karvamattojen karvojen puuttumisena (pois lukien yksi konemalli, joka oli pienemmästä koostaan huolimatta äärimmäisen raskas käytetyille VR-laseille). Yksinkertaistetusta mallista huolimatta koneet olivat erittäin yksityiskohtaisia ja näyttivät todella hyviltä VR-laseilla.

Halutut ominaisuudet, kuten liikkuminen koneen ympärillä ja kolmessa korkeustasossa liikkuminen, vaikuttivat myös suunnitteluun. Toiminnallisuuden ja helppokäyttöisyyden maksimoimiseksi tasossa täytyi olla tarpeeksi tilaa liikkua koneen ympärillä sekä yläpuolella. Jotkin koneet itsessään olivat todella korkeita tai leveitä ja tästä syystä tilaa täytyi olla reilusti. Liian suuri avonainen tila ei myöskään ole hyvä kokemuksen kannalta, joten sopivan kokoisen tason tekeminen vaati jonkin verran testaamista. Tasosuunnittelu lähti tarpeesta varsinkin, kun aikataulu on tiukka tai tason ulkonäkö ei ole keskeinen seikka.

3 SUUNNITTELUN MALLIT, ACTORIT, TEKSTUURIT JA MATERIAALIT

Suunnittelusta saadut mallit tuotiin Datasmith-liitännäistä käyttäen Unreal Engineen ja liitännäinen muutti jt-tiedoston (yhdessä tapauksessa stp.) DatasmithSceneActoriksi, joka koostuu tuhansista actoreista. Tämän jälkeen kaikki Datasmithin tuomat actorit yhdistettiin yhdeksi actoriksi Unreal Enginen Merge -ominaisuudella. Merge luo uuden yhdistetyn actorin, jonka jälkeen alkuperäisen voi poistaa tasosta ja korvata yhdistetyllä actorilla. Tämä säästää hurjasti tehoja, ja tässä työssä hyvä toimivuus oli yksi kriteereistä.

Suunnittelun mallien tekstuurit ja materiaalit tulevat sellaisenaan actoreihin. Ulosviennin yhteydessä malleja oli yksinkertaistettu. Tästä syystä jotkin materiaalit eivät vastaa todellista laitetta, mutta tämä katsottiin kuitenkin pieneksi haitaksi lopputuloksen muutoin hyvän laadun vuoksi. Saatuja tiedostoja käytettiin sellaisenaan ohjelman kehitykseen, vaikka työssä näytettiin toteen, että Unreal Enginessä voi myös manuaalisesti tehdä saaduista malleista hieman parempia. Työn tarkoituksena ei kuitenkaan ole parantaa saatuja malleja, vaan näyttää ne sellaisenaan virtuaalisessa ympäristössä.

3.1 Datasmith ja mallit

Datasmith on kehitetty helpottamaan muissa ohjelmistoissa suunniteltujen 3D-mallien ja ympäristöjen tuontia Unreal Engineen. Vaikutuksen teki kuitenkin liitännäisen yksinkertaisuus ja nopeus. Liitännäisen asentamisen jälkeen quickly add to the project -alasvetovalikosta voi valita datasmith file import -valinnan, jolla liitännäinen tuo automaattisesti valitun tiedoston Unreal Enginessä auki olevaan tasoon. Tässä vaiheessa voi myös muokata, mitkä osat mallista haluaa tuoda Unreal Enginen puolelle (tässä työssä käytettiin oletusasetuksia ja tuotiin kaikki saatavilla oleva tieto suunnittelun antamista tiedostoista).

Tuotu malli koostuu kaikista tiedostossa olleista osista ja tässä työssä jokaisessa mallissa oli tuhansia osia. Unreal Enginessä on kuitenkin Merge Actors -työkalu,

jolla useampi actor voidaan yhdistää yhdeksi actoriksi. Sen jälkeen tasosta poistetaan Datasmithin tuoma malli ja viedään content drawerista (projektin tiedostojenhallintaikkuna, joka sisältää kaikki projektissa olevat tiedostot) uusi yhdistetty actor haluttuun paikkaan tasossa.

3.2 Actorit

Unreal Enginen actorit ovat minkälaisia tahansa olioita, jotka voidaan lisätä tasoon (Actors | Unreal Engine Documentation, n.d.). Actorit voivat sisältää esim. koodia, meshin, tekstuureita, animaatioita tai ääniä. Outlinerissa näkyvät kaikki avoinna olevan tason actorit ja niiden muuttujat, joita voi muokata suoraan avaamatta actoria.

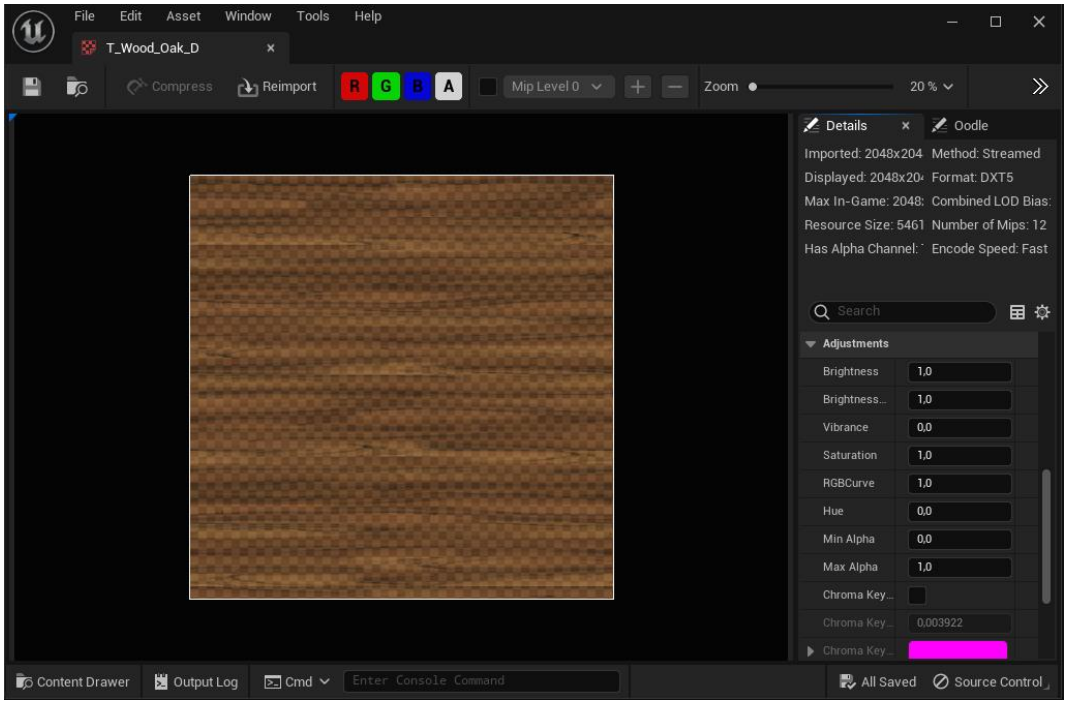
Actorit ovat Unreal Enginen ydin luodessa lähes mitä tahansa sovellusta. Tässä työssä tärkeimmät actorit ovat VRPawn, tasossa olevat meshit, jotka luovat huoneet, valot ja laitemallit. VRPawn toimii käyttäjän virtuaalisena paikkana ja antaa mahdollisuuden tarkkailla ympäristöä. Meshit luovat ympäristön ja saavat tilan tuntumaan luontevalta. Ilman valoja näkymä olisi täysin musta ja oikeanlaisella valoisuudella pystyy luomaan ympäristöön tunnelmaa. Tämän työn ydin on kuitenkin laitemallit ja nekin ovat actoreita, jotka koostuvat mm. monista eri materiaaleista sekä kaikista Unreal Enginen antamista muuttujista, jotka ovat oletuksena jokaisessa mesh actorissa. Toki suuri osa valmiista muuttujista on oletusarvoisesti pois päältä tai vaatii lisää dataa toimiakseen, mutta esimerkkinä tässä vaiheessa mainittakoon cast shadow, joka on oletusarvoisesti päällä ja joka tätä työtä varten täytyy ottaa pois päältä toimivuuden ja laitteiden tarkastelun vuoksi.

3.3 Tekstuurit

Opinnäytetyössä käytetyt tekstuurit tulevat neljästä eri paikasta. Ensimmäisenä on starter content, joka sisältää valmiiksi yksinkertaisia tekstuureita. Toisena on

Unreal Engine marketplace, josta suurin osa aulan tekstuureista on hankittu. Kolmantena on Bridge, josta suurin osa hallin tekstuureista on ladattu. Neljäs on suunnittelun mallit, joiden mukana tulee kaikki tiedostoon tallennetut tekstuurit.

Yleisimmät asetukset tekstuurin (Kuva 6) nopeaan muuttamiseen ovat kirkkaus tai väri. Tämä käy helposti muuttamalla tekstuurin muuttujia esimerkiksi brightness, brightness curve tai hue muuttujia säätämällä. Jos tekstuuria käytetään enemmän kuin yhdellä pinnalla tai sen mahdollisuus on olemassa, on suositeltua luoda tekstuurista oma instanssi ja asettaa se haluttuun pintaan tai materiaaliin.



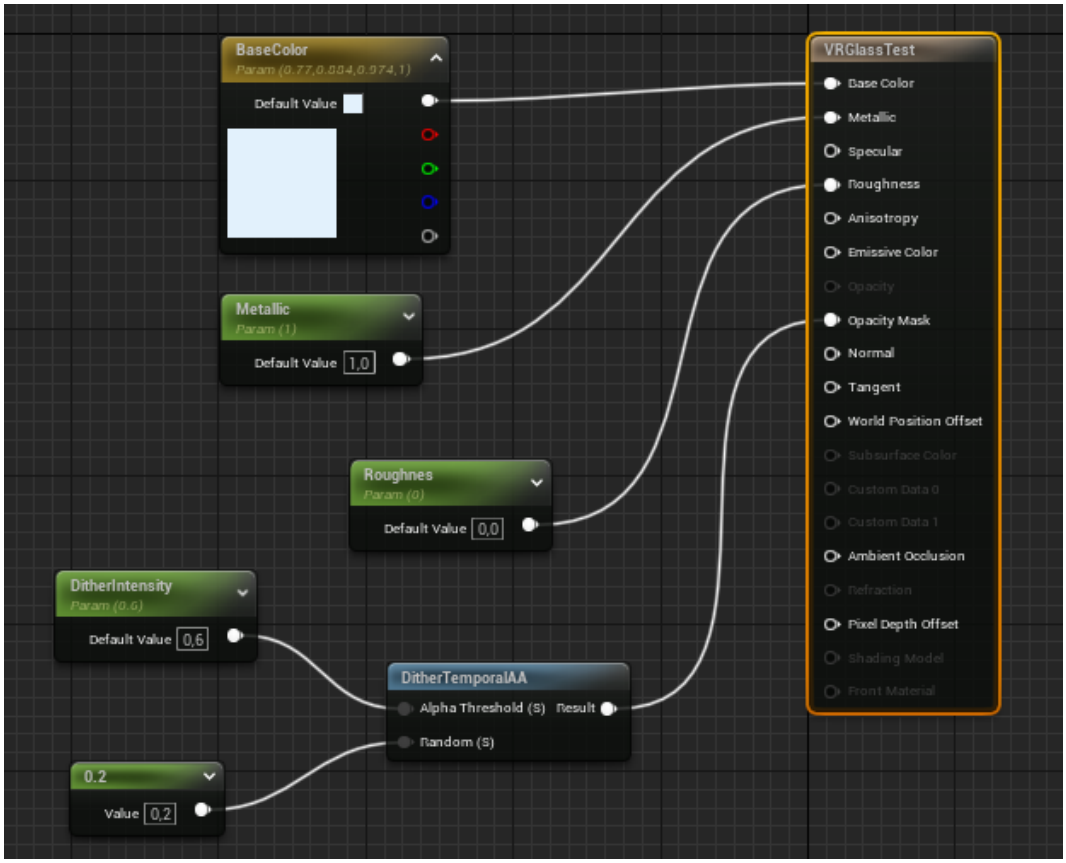
Kuva 6. Tekstuuri ja sen säädöt.

Tässä työssä esimerkiksi hallin pintojen väriä muokattiin tummemmaksi nopeasti säätämällä kirkkaus arvoja. Myös hue (värisävy) ja saturation (saturaatio) olivat nopean pinnan ulkonäön muutoksen kannalta käytännöllisiä muuttujia.

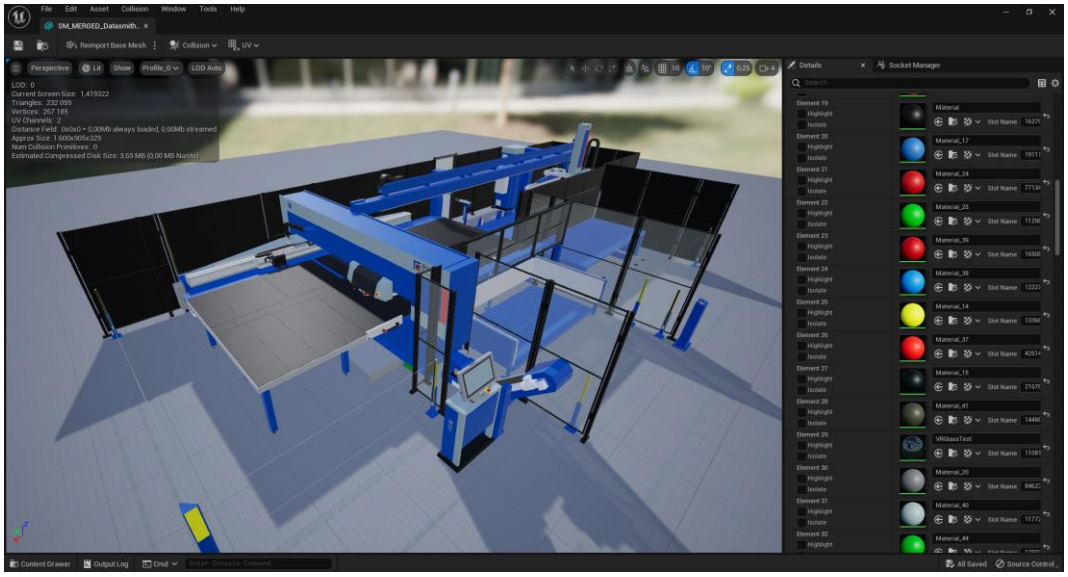
Tekstuurit voidaan asettaa suoraan pinnoille tai ne voidaan asettaa osaksi materiaalia. Suoraan pinnalle asetettuna ne antavat tyydyttävän lopputuloksen, mutta osana materiaaleja ne voivat näyttää huomattavasti paremmalta (Textures in Unreal Engine | Unreal Engine 5.1 Documentation, (n.d.).

3.4 Materiaalit

Unreal Enginessä materiaalit ovat yleisin tapa antaa ohjelmalle visuaalista kauneutta (Unreal Engine Materials | Unreal Engine 5.1 Documentation, n.d.). Materiaaleissa on huomattavasti enemmän mahdollisuuksia muokkauksiin, kuin pelkissä tekstuureissa. Materiaaleja muokataan node-ohjelmoinnilla, joka muistuttaa huomattavasti myös Unreal Enginen blueprint-ohjelmointia (Kuva 7).



Kuva 7. Itse tehty keinotekoinen lasimateriaali.



Kuva 8. Koneen läpinäkyvät osat sisältävät itsetehdyn keinotekoisen lasimateriaalin.

Oculus (Meta) Quest 2 -lasit eivät toistaneet valmiina olleita lasimateriaaleja oikein, joten työssä tehtiin itse oma materiaali lasiksi. Tarkoituksena oli testata pystyykö mallien materiaaleja järkevästi muokkaamaan itse lähemmäs realistista konetta. Työssä näytettiin toteen, että se on täysin mahdollista, mutta manuaalisesti materiaalien vaihtaminen ja koneiden muokkaaminen jälkikäteen on aikaa vievää ja epäkäytännöllistä sen vuoksi. Myös käytetyt VR-lasit vaikeuttivat joidenkin materiaalien käyttöä, näistä esimerkkinä juuri kaksi erillistä lasimateriaalia starter contentista ja Unreal Engine marketplacesta.

Perusmateriaali node (ks. Kuva 7), joka on työssä nimetty VRGlassTest, saa tässä tapauksessa neljä sisääntuloa. Base Color antaa materiaalille sen yleisen värin. Metallic antaa materiaalille metallista ominaisuutta, mikä tässä tapauksessa antaa lasinomaisen ympäristön heijastuksen ja aavistuksen tiheyttä materiaalin omaavalle pinnalle. Roughness antaa tässä esimerkissä materiaalille sileyttä, mikä antaa enemmän kiiltoa ja heijastusta varsinkin läheltä tarkasteltuna. Neljäs sisääntulo opacity mask antaa läpinäkyvyyden materiaalille.

Tällä tavalla tehdyssä materiaalissa on joitain lasin ominaisuuksia, mutta ei silti voida puhua aidosta lasimateriaalista, koska valon reagoiminen materiaaliin ei vastaa lasin ominaisuuksia. Valon reagoiminen materiaaliin on luultavasti syy, miksi aidommat lasimateriaalit eivät toimineet Oculus (Meta) Quest 2 -laseilla. Materiaali toimi kuitenkin hyvin

4 BLUEPRINT-OHJELMOINTI, KÄYTETYIMMÄT BLUEPRINTIT JA TOIMINNALLISET OMINAISUUDET OHJELMASSA

Tässä työssä käytettiin Unreal Enginen blueprint-ohjelmointia toiminnallisten ominaisuuksien saavuttamiseksi. Blueprint-ohjelmointi on visuaalista c++ ohjelmointia, joka helpottaa ja nopeuttaa sisällön tuottamista. Blueprintit mahdollistavat visuaalisen ohjelmoinnin ja tekemällä actorista blueprintin voi lisätä paljon toiminnallisuuksia itse actoriin. Tässä työssä suurin osa toiminnallisuuksista lisättiin kuitenkin level blueprinttiin sekä VRPawn blueprinttiin.

Ohjelmoinnin perusteet on välttämätön taito ymmärtää logiikkaa ja syvemmät taidot ovat ehdottomasti hyödyksi tämän tason visuaalisessa koodaamisessa. Spesifistinen ymmärrys nodeista Unreal Enginessä tulee tutustumalla manuaaleihin ja opetusmateriaaliin Unreal Enginen omilla sivuilla.

4.1 Blueprint-ohjelmointi

Unreal Enginen visuaalinen ohjelmointi on tehokas työkalu pieniin ja keskikokoisiin projekteihin tai prototyyppeihin (Overview of Blueprints Visual Scripting in Unreal Engine | Unreal Enginge 5.1 Documentation, n.d.). Pääsääntöisesti kaikki toiminnallisuus, joka ei juokse taustalla taukoamatta, on järkevää laittaa blueprint-luokkiin. Unreal Enginessä on suuri määrä blueprint-luokkia ja niitä voi luoda myös itse. Tässä työssä käytettiin VRPawn blueprinttiä, Level blueprinttiä ja widget blueprinttiä.

Blueprint-ohjelmointi on visuaalista ohjelmointia, jonka pohjalla on C++. Tämän vuoksi on mahdollista myös haluttaessa sekoittaa blueprint-ohjelmointia ja C++-ohjelmointia sujuvasti keskenään. Tässä työssä ei ollut tarpeellista ohjelmoida C++:lla, koska kaikki halutut ominaisuudet olivat ajallisesti tehokkaampaa tehdä blueprint-ohjelmoinnilla. Toinen suuri hyöty blueprint-ohjelmoinnissa on se, että on mahdotonta tehdä syntaksi- ja kirjoitusvirheitä.

4.2 Yleisimmät blueprinteissä käytetyt node tyypit ohjelmassa

Event nodet eli punaiset nodet, ovat hyvin yleisiä kaikissa ohjelmistoissa, jotka on luotu visuaalisella koodaamisella (Events | Unreal Engine 5.1 Documentation, n.d.). Event node käynnistyy, kun sille omistettu tapahtuma käynnistyy, on käynnissä tai päättyy. Näitä käytetään pääsääntöisesti kaikkeen toiminnallisuuteen, niin ohjelmiston sujuvassa toiminnassa kuin näppäinten seurannassa. Tässä työssä käytetyt event nodet olivat On Actor End Overlap, useat EnchanchedInputInputAction nodet (näppäinten seuranta ja toiminnallisuus), EventBegin Play (pitää huolen kaikesta VR lasien toiminnasta, mikä tulee VR templaten mukana) ja On Clicked (valikon näppäinten painamisesta huolehtivat event nodet).

Function Call Nodet eli siniset nodet, nimensä mukaan tarjoavat erilaisia toiminnallisuuksia (Function Calls in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.). Yleisimmät tässä työssä käytetyt function call nodet, jotka eivät tulleet valmiina templaten mukana, ovat Set Actor Transform (muokkaa actorin kääntymistä, lokaatiota tai kokoa), Delay (hidastaa ohjelman juoksua halutulla ajalla), muuttujien lukeminen, setterit ja Open Level (by Object Reference). Open Level (by Name) aiheutti ongelmia ja epävarmuutta toimivuuden kanssa testeissä, joten se korvattiin varmemmin toimivalla Open Level (by Object Reference) nodella.

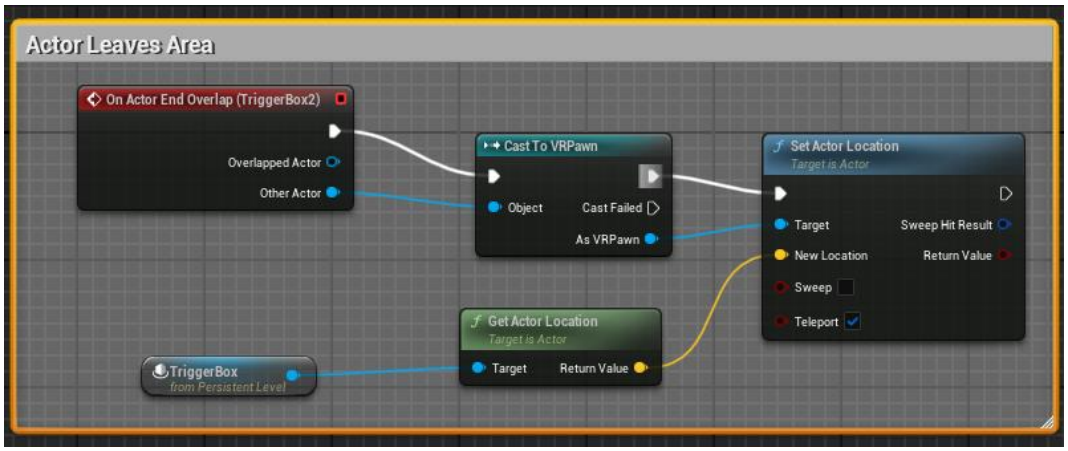
Vihreät nodet pääsääntöisesti sisältävät gettereitä tai niihin liittyviä toiminnallisuuksia. Get Actor nodet ovat käytetyin näistä nodeista tässä työssä käytetyistä get nodeista. Esimerkiksi jotkin set nodet saattavat olla vihreitä, jos ne ensin hakevat jonkin tiedon get-ominaisuudella.

Harmaat nodet ovat pääsääntöisesti virtauksen hallintaan (Flow Control) ja loogisia nodeja. Tässä työssä käytettiin Do Once-, Branch- ja Compare-nodeja hallitsemaan virtausta ja logiikkaa (Flow Controll in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.).

Unreal Enginessä on huomattava määrä nodeja, ja värikoodit antavat vahvan viittauksen, millaisiin toimintoihin niitä voi käyttää. Myös muuttujien tyypeillä on värikoodeja sekä vektoreilla oma värinsä. Tyypit ovat pääsääntöisesti nähtävissä väreistä ja pääsääntöisesti erityyppisiä muuttujia ei kyetä ilman datan muuttamista yhdistämään toisiinsa. Myös Cast nodet, joita käytetään mm. datan saamiseen silloin kun se ei muutoin olisi mahdollista, omaavat oman värinsä (Nodes in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.).

4.3 Lisätty toiminnallisuus ohjelmassa

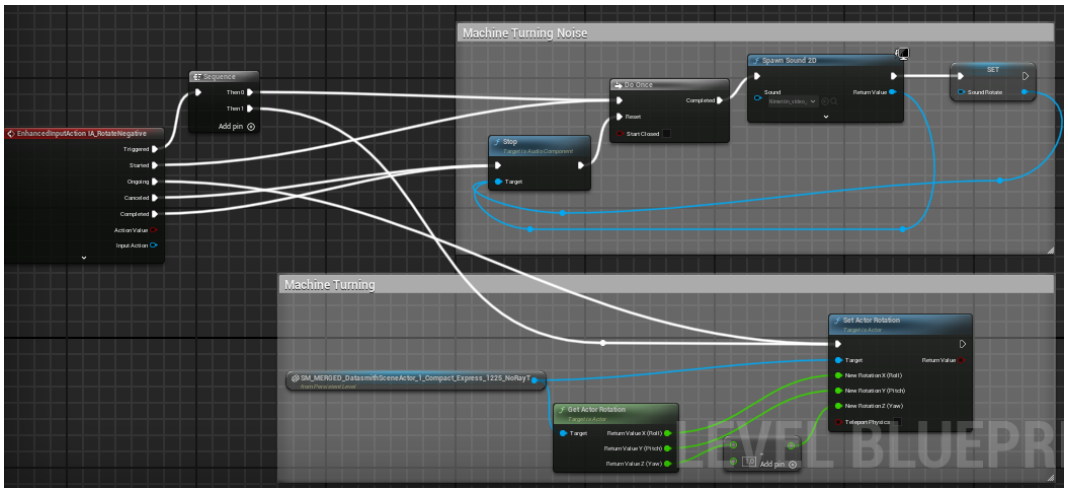
Tässä osassa käydään läpi lisätyt toiminnallisuudet ja niiden rakenteet pois lukien UI ja menu, jotka käsitellään erikseen. Kontrolleista puhutaan tarkemmin luvussa 6, mutta kontrollien toiminnallisuus käsitellään tässä luvussa.



Kuva 9. Käyttäjä kulkee alueen ulkopuolelle.

Ensimmäisenä käsitellään koodi (Kuva 9), joka on lisätty jokaiseen tasoon. Tasoon lisätään Trigger Box ja tason blueprinttiin (Level blueprint) luodaan tapahtuma ja toiminnallisuus, joka palauttaa käyttäjän, jos käyttäjä yrittää kulkea tason ulkopuolelle. Tässä ongelma ratkaistiin siirtämällä pelaaja toisen tasoon sijoitetun Trigger Boxin kohdalle. Koska VRPawnia ei ole tasossa, sitä ei voida käyttää suoraan sen sijaan joudutaan käyttämään Cast To nodea (VRPawn Actor tuodaan pelin alussa starting location actorilla). Cast To yrittää hakea halutun actorin tasosta ja tehdä sille halutun toiminnan. Get Actor location node hakee toisen TriggerBoxin

ja vie sen lokaation Set Actor Location nodelle. Set Actor Location node saa siis kohteekseen VRPawn Actorin ja siirtää sen TriggerBox actorin lokaatioon Teleport-toimintoa käyttäen.

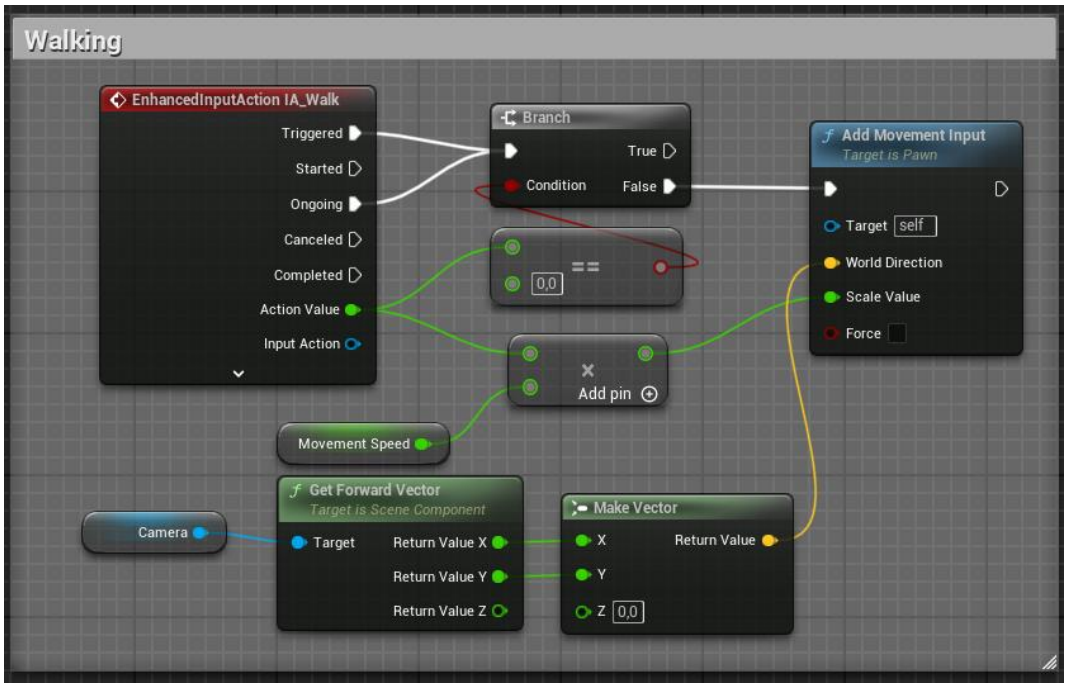


Kuva 10. Koneen kääntäminen.

Tässä (Kuva 10) käyttäjän näppäimen painallustapahtuma (Event) aloitetaan IA nodella (InputAction eli käytännössä käyttäjän näppäinten tai muiden mahdollisten sisääntulojen tapahtuma). Sequence node on lisätty Triggered-kohdalle, ettei ääni pyörisi kahta kertaa päällekkäin. Myös Do Once node varmistaa saman asian. Triggered ja Started liittyvät siihen, kun käyttäjä ensimmäisen kerran painaa näppäintä. Ongoin tapahtuu, kun käyttäjä pitää näppäintä alas painettuna. Canceled ja Completed taas liittyvät siihen, kun näppäin nostetaan tai toiminta on suoritettu. Machine Turning Noise -osa koodista toistaa siis lyhyen koneäänen, joka pysähtyy, kun ääni on toistettu joko kokonaan tai käyttäjä nostaa kääntöön osoitetun näppäimen ylös.

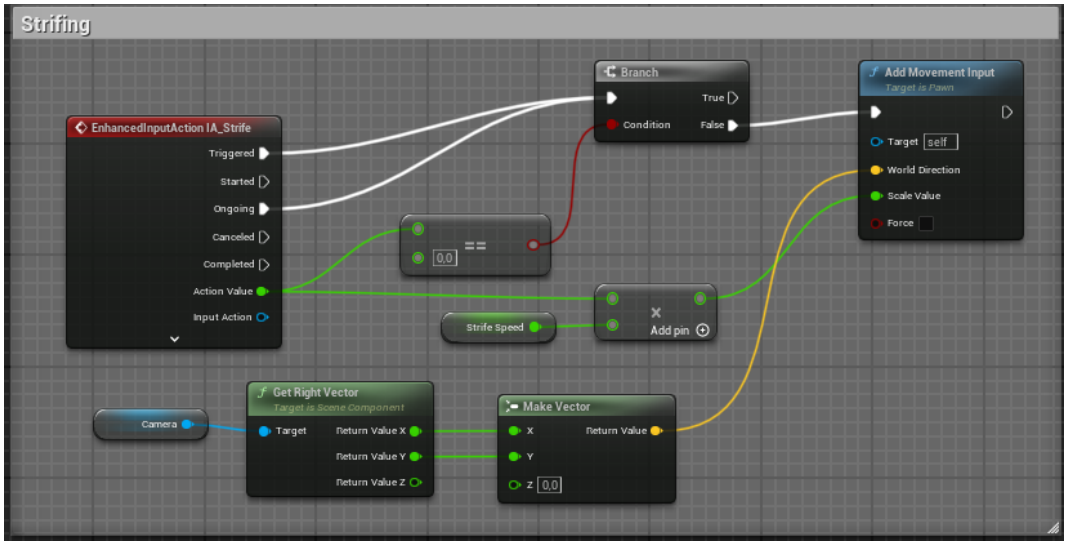
Machine Turning osa taas pitää huolen koneen kääntämisestä. Kone lähtee kääntymään heti, kun näppäintä painetaan ja kääntyy niin kauan kuin näppäin on alas painettuna. Get Actor Rotation hakee koneen nykyisen rotaation ja Set Actor Ro-

tation asettaa sen uuden rotaation suhteessa edelliseen. Yksinkertainen yhden asteen vähentäminen z akselilla pyörittää konetta. Toisessa vastaavassa koodissa on vain lisätty yksi aste vähentämisen sijaan, jolloin kone pyörii toiseen suuntaan.



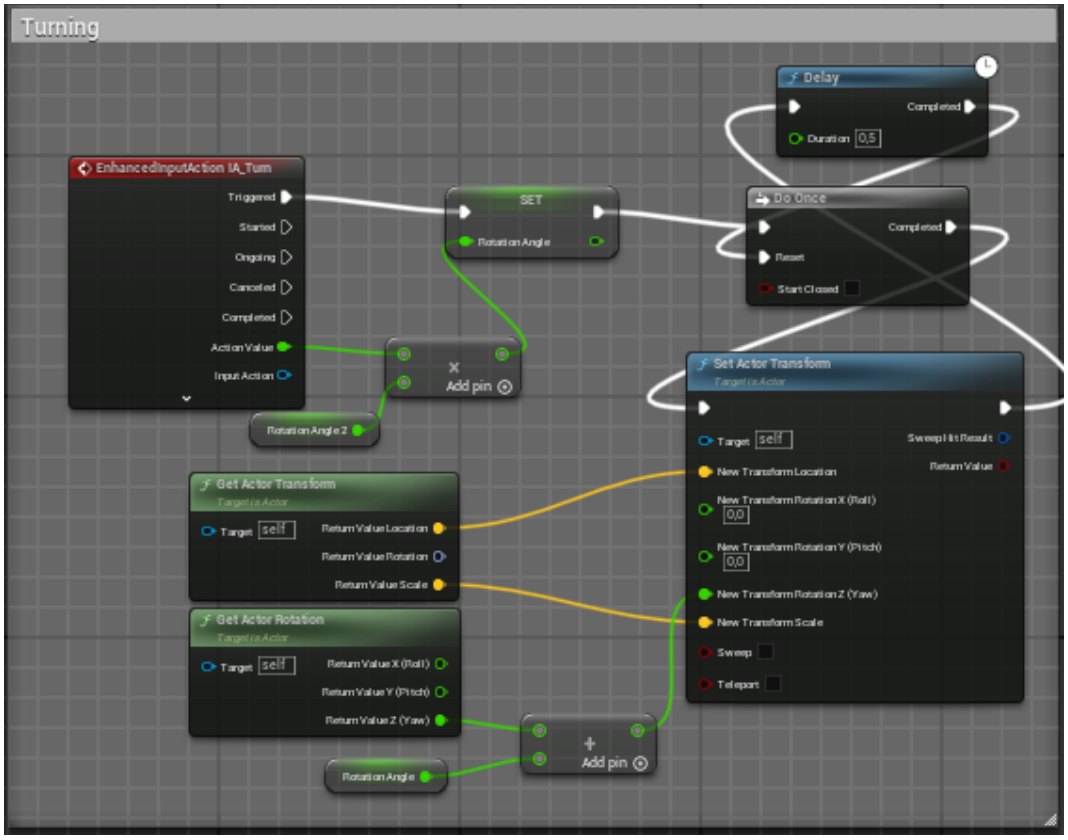
Kuva 11. Walking-koodi.

Tässä koodissa (Kuva 11) IA node käynnistää toiminnallisuuden tilassa liikkumiselle mitä kutsutaan kävelyksi. Kun käyttäjä käyttää liikkumiseen tarkoitettua näppäintä, niin käyttäjä liikkuu Movement Speed muuttujan, sekä näppäimen antaman arvon kertoimella eteen tai taaksepäin. Branch on lisätty varmistamaan se, että toiminta ei juokse silloin kun käyttäjä ei liiku mihinkään suuntaan. Add Movement (Target is Pawn) kohdistaa tapahtuman käyttäjän ohjaamaan Pawn actoriin ja saa suunnan kamerasta. Koska työssä ei haluttu minkäänlaista sulavaa liikkumista ylös tai alas, ei oteta huomioon Z-akselia. Scale Value on käytännössä liikkumisen määrä ja tätä määrää hallittiin sujuvasti Movement Speed muuttujalla.



Kuva 12. Strifing eli sivuttainen liike.

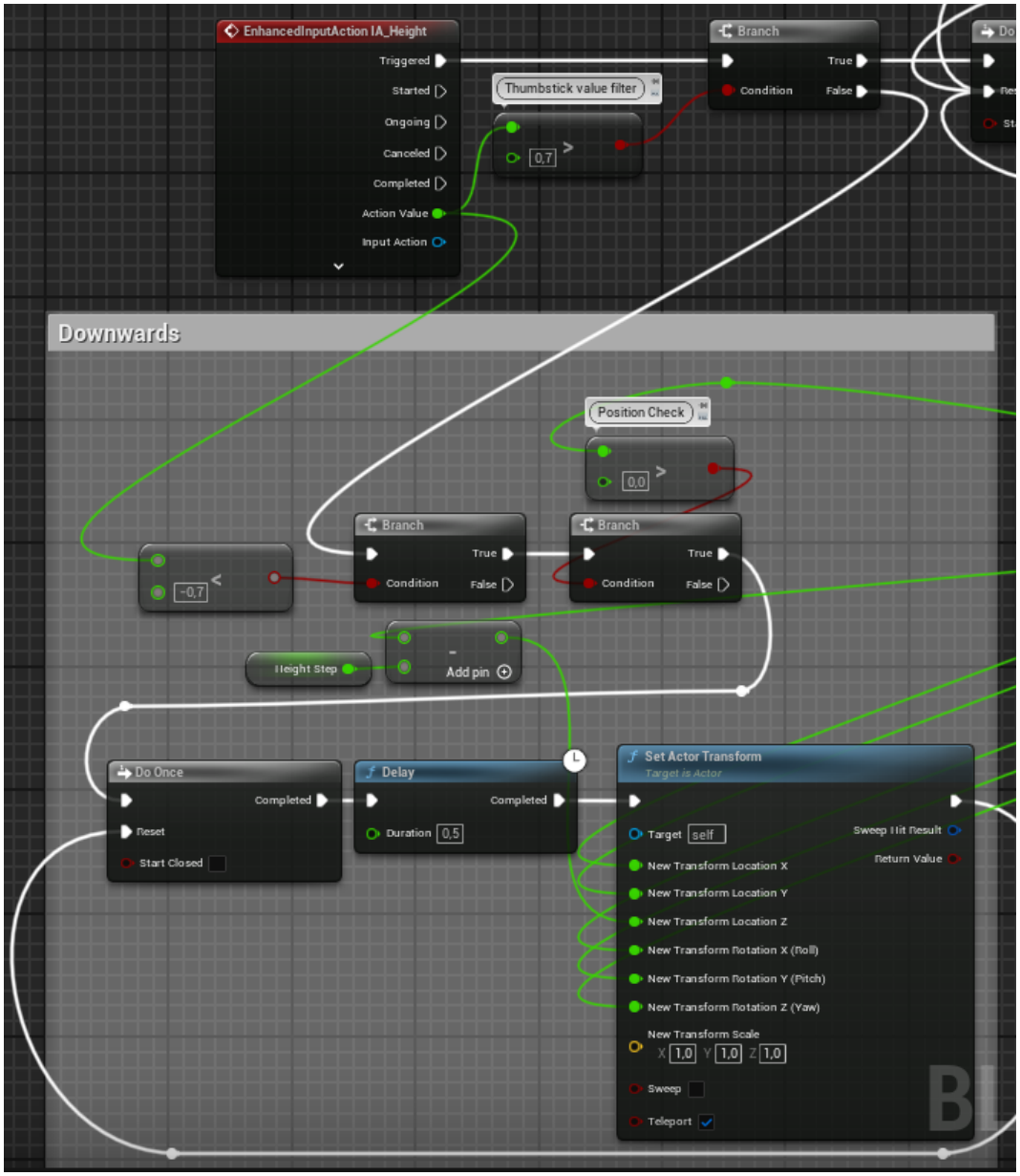
Sivuttain liikkuminen (Kuva 12) toimii täysin samalla periaatteella, kuin eteenpäin liikkuminenkin, mutta saa eri suunnan World Directioniin. Koska liikkumiseen tarkoitettu näppäin on thumbstick, arvot ovat väliltä -1.0 ja 1.0. Tämä mahdollistaa sujuvan liikkumisen myös viistoon, jolloin Walking ja Strifing saavat molemmat arvoja.



Kuva 13. Turning-koodi.

Kääntyminen (Kuva 13) oli toiminnallisesti yksi VR-laseille raskaimmista toiminnoista. Tästä syystä paras lopputulos saatiin kerta kääntymisellä maksimissaan puolen sekunnin välein. Sujuva kääntyminen oli liian raskasta koneiden hurjan yksityiskohtaisuuden vuoksi ja kuvataajuuden siedettävänä pitämiseksi ainoa ratkaisu oli vähentää kääntymisen sulavuutta. Pääpiirteittäin tämä koodi seuraa aikaisempien periaatteita. Delay ja Do Once pitävät huolen, että käyttäjä ei voi kääntyä liian usein. Set Actor Transform saa kohteen itsestään, koska koodi sijaitsee VRPawn Actorin sisällä ja node vaati lokaation ja skaalan, joten haetaan nykyiset Get Actor Transform nodella. Koska Rotation Angle haetaan toisesta thumbstickistä, se saa arvoja 1.0 ja -1.0 välillä, joten vain yksi koodi riittää saamaan kääntämisen kumpaankin suuntaan z akselilla. IA noden Action Value otetaan talteen ja muokataan Rotation Angle 2 muuttujalla sopivan suuruiseksi ja asetetaan todelliseen Rotation Angle muuttujaan ennen Delay ja Do Once nodeja.

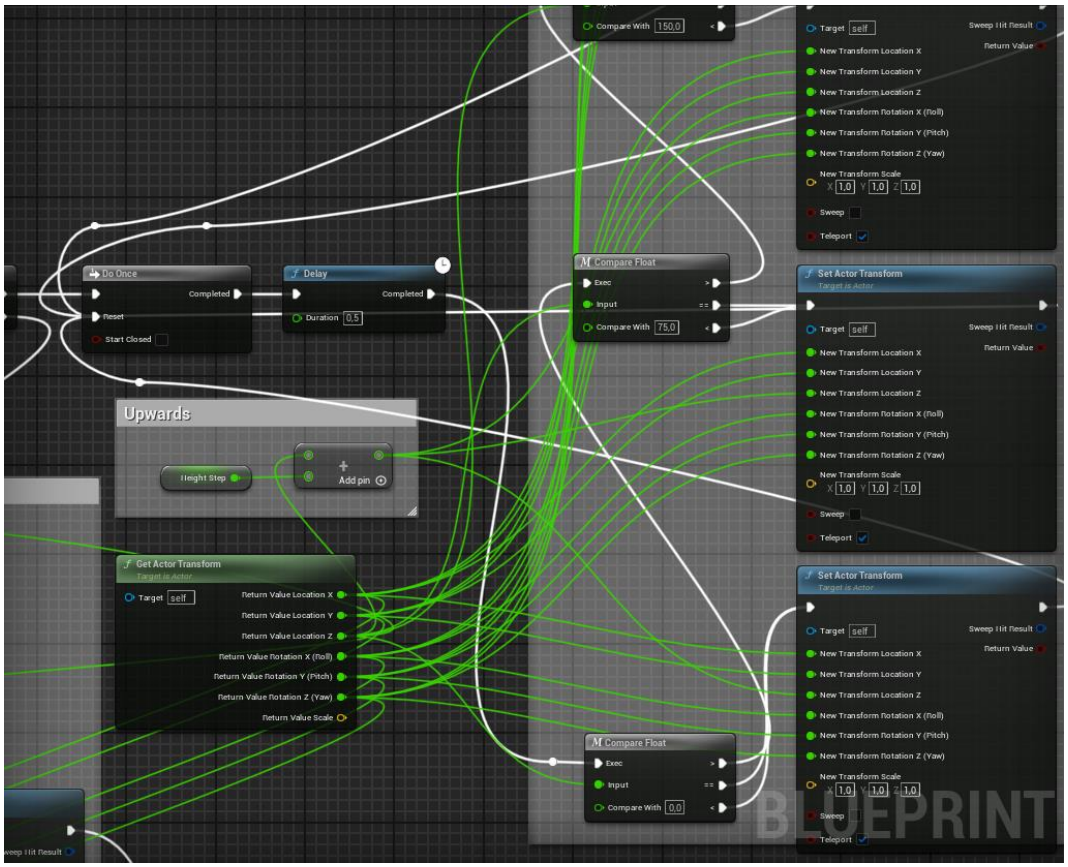
Yksi työssä vaadituista asioista oli saada käyttäjä kulkemaan kolmella korkeudella pystyäkseen tarkastelemaan koneita paremmin. Tämän koodin (Kuva 14, Kuva 15 ja Kuva 16) visuaalisesti esittäminen on hieman sotkuisen näköistä lukuisten yhteyksien vuoksi. Käytännössä Unreal Enginessä kykenee tarkistamaan yhteyden viemällä hiiren sen kohdalle, jolloin se korostaa kyseisen yhteyden ja himmentää muut yhteydet.



Kuva 14. Downwards sekä filtterit korkeutta säätävälle thumbstickille.

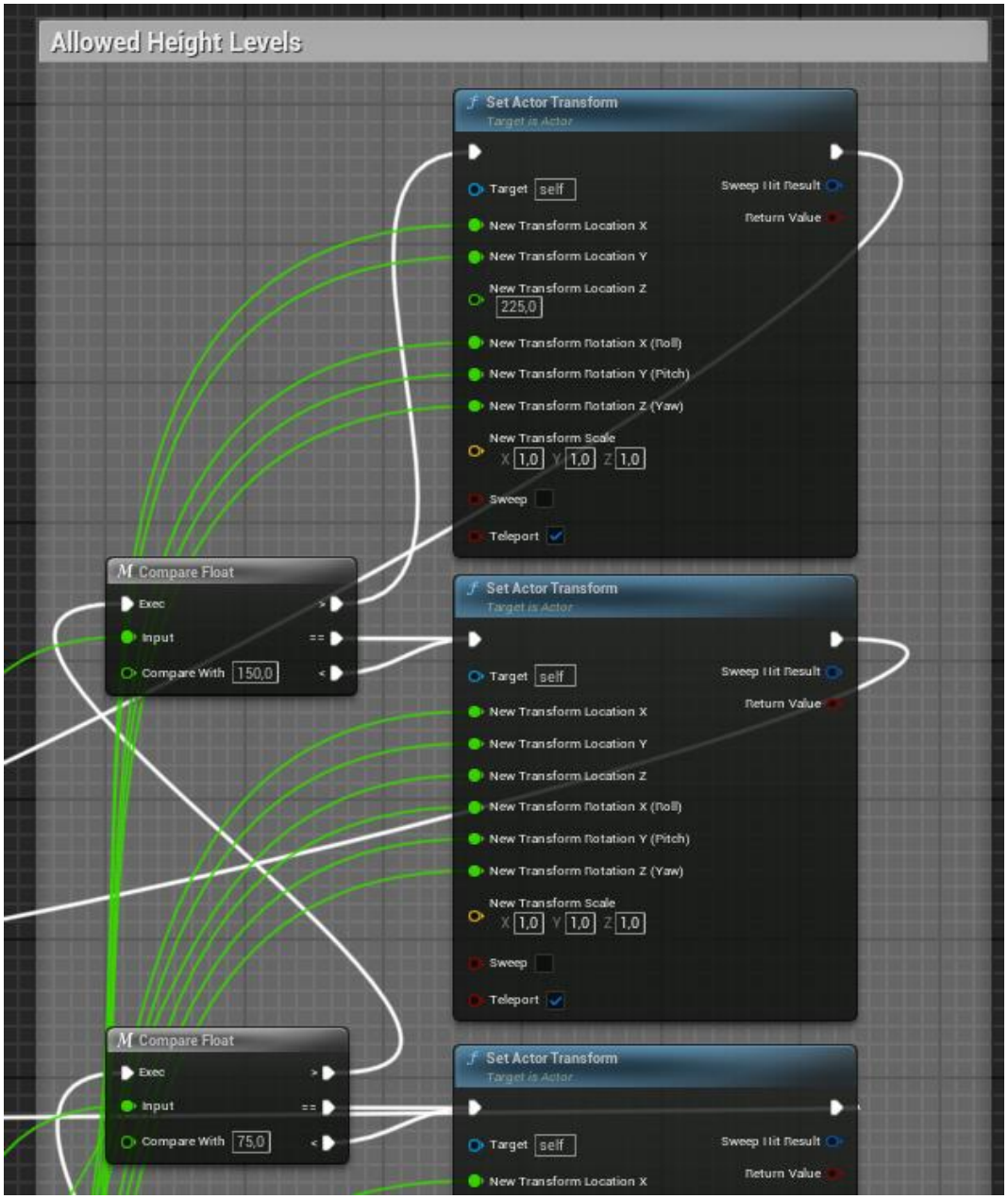
IA nodesta saatava arvo ajetaan filttereihin, jotka katsovat, onko thumbstick siirretty enemmän kuin seitsemänkymmentä prosenttia ohjaimessa halutulla akselilla. Ensimmäinen Branch tarkistaa, onko saatu arvo enemmän kuin 0.7 ja jos ei, niin tarkistetaan, onko saatu arvo vähemmän kuin -0.7. Seuraavaksi tarkistetaan, onko käyttäjän sijainti korkeammalla kuin 0.0 ja jos on, niin käyttäjän sijainti asetetaan Height Step -muuttujan arvon verran alemmas (Height Step -muuttuja on

lisätty alkuperäisen toiminnon jälkeen, joka oli siirtää pelaaja kolmelle tasolle ylöspäin ja sen jälkeen palauttaa lattiatasoon).



Kuva 15. Upwards, Get Actor Transform ja alimmat Comparet sekä alimmat sallitut tasot.

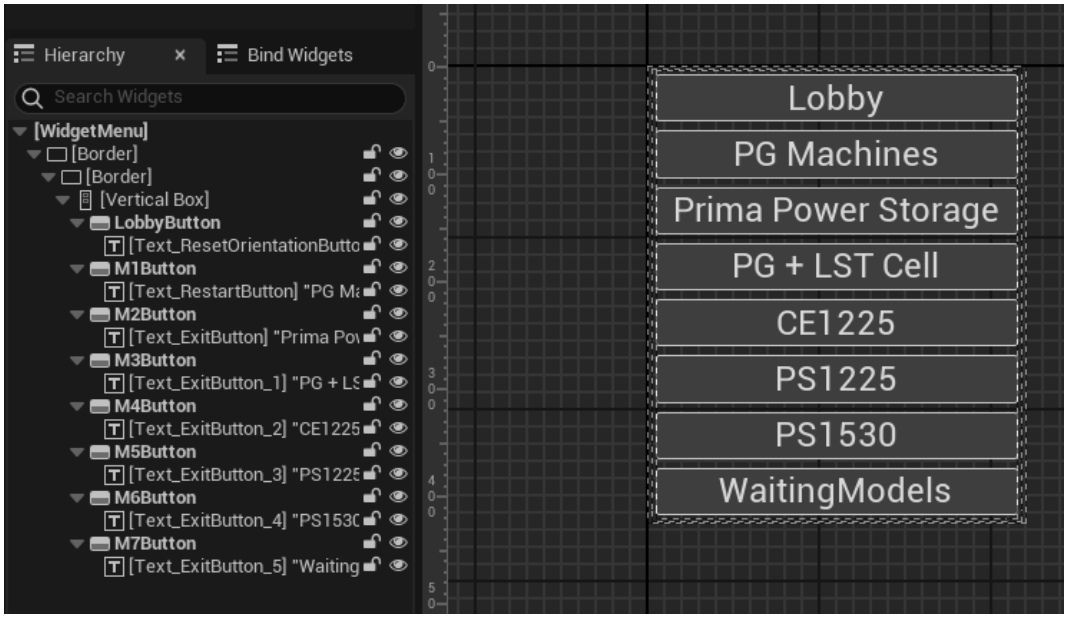
Comparet pitävät huolen sallituista korkeuksista. Get Actor Transformista kaikki nodet, jotka tarvitsevat actorin lokaation, hakevat sen. Koodissa on havaittavissa selvästi kehityssuunta kohti muuttujia kolmen kiinteän tason sijaan, mikä oli alkuperäinen tarkoitus ennen palautetta ja kehitysehdotuksia. Koodi on siinä pisteessä, johon se ehdittiin työn aikana tekemään. Sen kehitystä ei ole sen jälkeen jatkettu (pois lukien muutama noden siirto ja kommentti kuvien parantelemiseksi). Esimerkkinä järkevä lisä olisi asettaa muuttuja kertaa haluttu taso (0, 1, 2,3) Compare With kohtaan Compare Float nodeja.



Kuva 16. Loput Allowed Height Levels -koodista

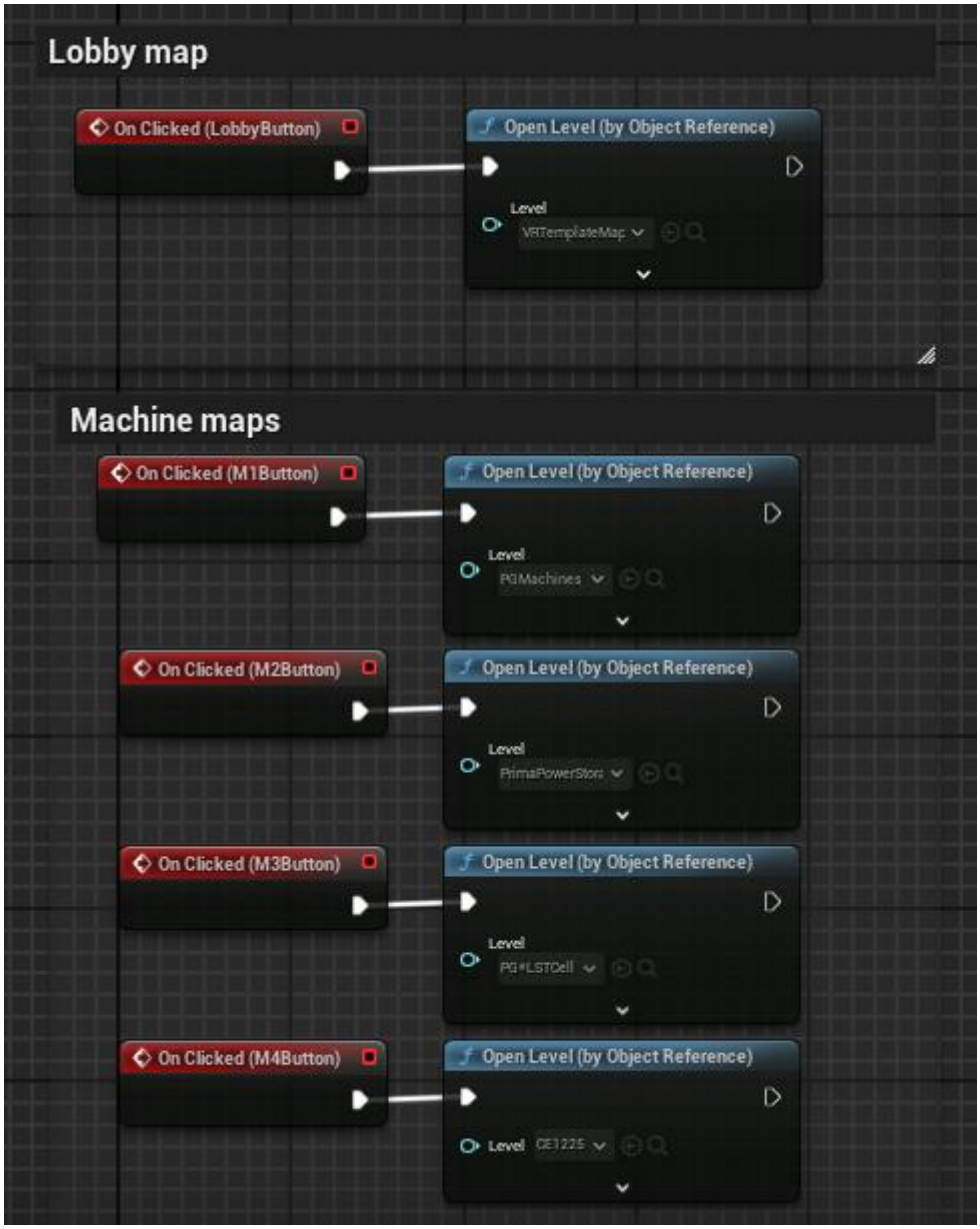
5 UI JA MENU

Unreal Enginessä UI (user interface) luodaan pääsääntöisesti widget blueprinttejä käyttämällä (Creating Widgets in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.). Widget blueprintit mahdollistavat esimerkiksi helpon menun luonnin ja tässä työssä muokattiin VR templaten mukana tullutta menua sopimaan haluttuun lopputulokseen. Yleisesti widget blueprinttejä käytetään lähes kaiken sen informaation näyttämiseen, joka näytetään käyttäjälle suoraan osana käyttöliittymää tai valikoissa. Tässä työssä UI:ssa ei oikeastaan näytetä mitään, paitsi painaessa menun avaavaa näppäintä.



Kuva 17. Menun hierarkia ja ulkoasu editorissa.

Tässä työssä menu piti huolen tason (Level) valinnasta ja tasot oli nimetty niiden sisältävän koneen mukaan (Kuva 17). Ensimmäinen taso on aula, jossa käyttäjä saa tutustua kontrolleihin ja viimeinen taso on tyhjä halli, joka sisältää koneen mallia lukuun ottamatta kaiken koodin. Uusia tasoja kykenee lisäämää menuun helposti kopioimalla ja liittämällä uuden painikkeen (mikä näkyykin tekstin alkuperäisen nimen osalta), mutta tärkeää on muistaa osoittaa uudelle painikkeelle uusi taso.



Kuva 18. Esimerkki yksinkertaisesta painikkeeseen sidotusta eventistä, joka vaihtaa tason (Level).

Yksi painikkeiden (Button) ominaisuuksista on valmis On Clicked -tapahtuma, joka tekee painikkeiden käytöstä erittäin yksinkertaista. Tietenkin on mahdollista sisällyttää suurempiakin toiminnallisuuksia, mutta tässä työssä siihen ei ollut tarvetta. Valikosta siirtyminen haluttuun tasoon oli pyydetty ominaisuus ja sen toiminta oli melko yksinkertaista lisätä työhön (Kuva 18).

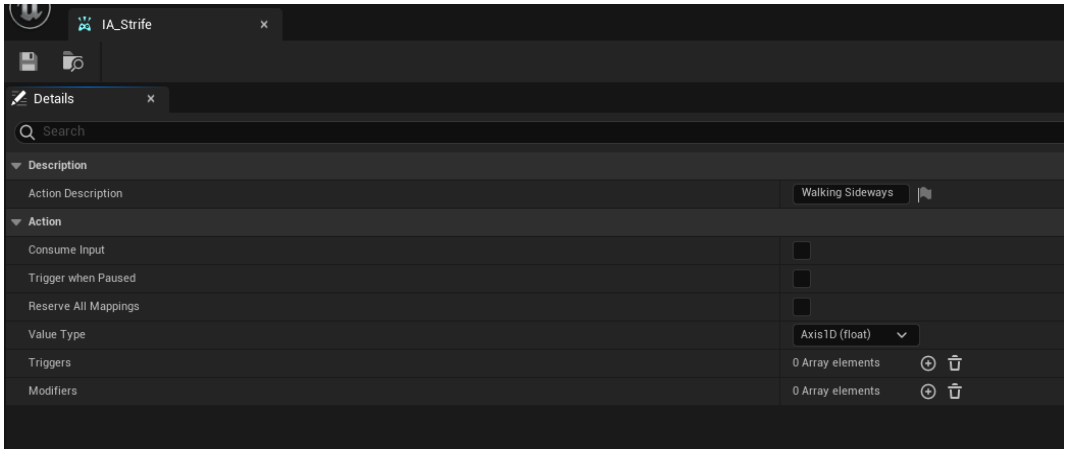
6 KONTROLLIT

Kontrollit Unreal Engine 5.x versioissa asetetaan hieman eritavoin kuin aiemmissa versioissa (Enhanced Input in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.). Nykyinen tapa antaa suurempaa vapautta ja helpottaa toiminnallisuuden lisäämistä. Kontrollit asetetaan Input Mapping Context -taulukkoon (IMC, Kuva 19).

Kuva 19. IMC_Default

Esimerkkinä IA_Strife (IA Strife toiminnallisuus käsiteltiin aiemmin ks. Lisätty toiminnallisuus ohjelmassa) asetetaan vasemman Oculus Touch -ohjaimen thumbstickin x-akseliin. Että tämä saataisiin asetettua IMC-listaan, tarvitaan ensin IA_Strife Input Action assetti (Kuva 20). IMC_Default-listassa nähdään hyvin selvästi, mihin ohjaimen näppäimeen on asetettu mikäkin Input Action assetti.

Kuva 20. Input Action assetit, joista osa templaten mukana tulleita.

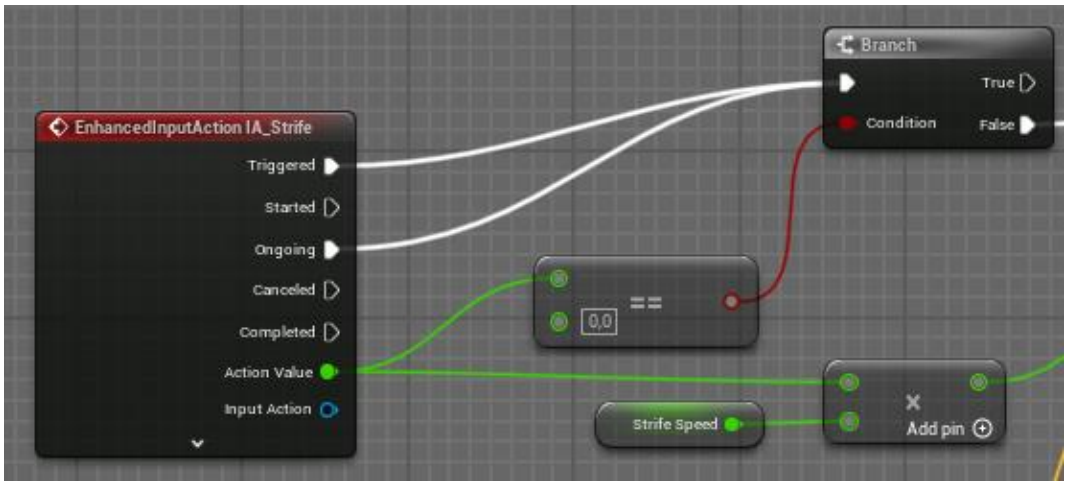


Kuva 21. IA_Strife-ominaisuudet.

Tämän jälkeen koodissa voidaan käyttää näppäinten käyttämistä halutulla tavalla. Järkevintä on tietysti asettaa käyttäjän liikkumiseen liittyvät ominaisuudet suoraan VRPawn actoriin, joka liikkuu tasojen välillä. Koneen pyörittäminen on asetettu tasoihin koska se on tason ominaisuus eikä käyttäjän.

Input Action assettien nimeäminen kuvaavilla nimillä on ehdottoman tärkeää helpon käytön varmistamiseksi. Varsinkin siinä tapauksessa, jos on toivottua saada ohjelma toimimaan usealla eri alustalla. Tässä työssä vaatimus oli nimenomaisesti

Oculus (Meta) Quest 2 VR -lasit, mikä helpottaa IMC taulukon lukemista. Taulukossa näkyy myös edelleen osa VR templaten mukana tuomista valmiista asetuksista muillekin VR laseille (valve index ja vive).



Kuva 22. IA_Strife koodissa.

Näppäinten käyttö koodissa on tällä tavalla toteutettuna hyvin yksinkertaista ja niiden toimintaa kyetään hallitsemaan täysin. Esimerkkinä mainittu IA_Strife tapahtuma node koodissa siis käyttää IA_Strife assettia ja tästä syystä Action Value on Axis-1D (float) -tyyppiä, eli arvoltaan yhden ja miinus yhden välillä. Muita arvoja on boolean, Axis-2D (vector 2D) ja Axis-3d (vector).

Input Action assetteihin voidaan asettaa myös jonkin verran lisätoiminnallisuuksia, esimerkiksi näppäimen täytyy olla painettuna tietyn aikaa tai usean näppäimen täytyy olla painettuna yhtä aikaa. Joidenkin muunteiden asettaminen onnistuu myös, mutta tässä työssä ei käytetty muunteita tai muitakaan lisätoiminnallisuuksia.

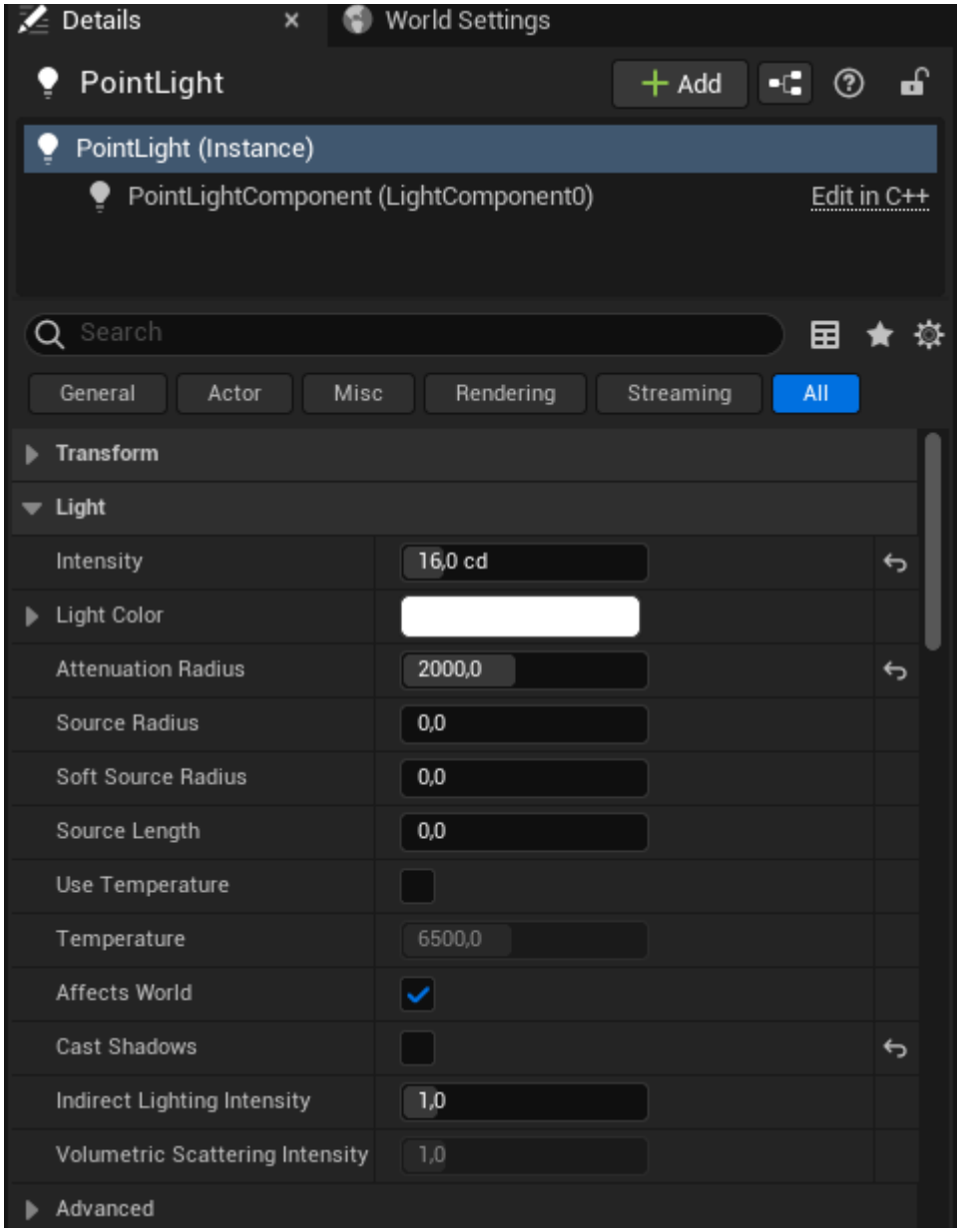
7 VALOT

Valoihin perehtyminen syvällisesti olisi vienyt turhan paljon aikaa työltä, joten testaamisella käytännössä korvattiin syvempi perehtyminen. Unreal Engine 5 on loistava ohjelma valaistuksen osalta, mutta Oculus Quest 2 ei tukenut tai kyennyt tehokkaasti juoksemaan kaikkia Unreal Enginen ominaisuuksia ja valoja. Valotyyppejä Unreal Enginessä on viisi: Directional Light, Sky Light, Point Light, Spot Light ja Rectangular Area Light (Light Types and Their Mobility | Unreal engine 5.1 Documentation, n.d.). Näistä viidestä Spot Light ja Rectangular Area Light aiheutti eniten ongelmia ja ne jäivät kokonaan pois ohjelmasta Oculukselle rakentamisen jälkeen. Unrealin Lumen ominaisuus ei myöskään ollut tuettuna VR-ympäristöissä aikana, jolloin työ tehtiin (Lumen Technical Details in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.).

Sisätilojen valaiseminen tarpeeksi ja mahdollisimman pienellä määrällä valoactoreita oli suuri haaste ja vaati käytännössä enemmän aikaa. Point Light (Point Lights in Unreal Engine| Unreal Engine 5.1 Documentation, n.d.) toimi sisätiloissa hyvin, mutta usea valonlähde vaikutti nopeasti kuvataajuuteen. Koneiden varjot täytyi ottaa pois käytöstä samasta syystä. Valot oli myös asetettu static actoreiksi, jolloin ne saatiin rakennettua skenen kanssa, eikä niitä tarvinnut laskea ohjelmistonollessa käynnissä.

Kuva 23. Tasossa olevat valot.

Tyhjässä tasossa näkyy mahdollisimman optimoidut valot (Kuva 23). Toimivassa versiossa oli aluksi yhdeksän valoa, mutta määrä saatiin putoamaan viiteen säätämällä valojen ominaisuuksia ja matkaa toisistaan.



Kuva 24. Tärkeimmät valon ominaisuudet.

Valojen asetukset (Kuva 24) säädettiin mahdollisimman kirkkaaksi ilman huomattavaa kuvataajuuden laskua. Cast Shadows oli ymmärrettävästi myös liian raskas ominaisuus Oculus Quest 2 VR laseille.

8 AJANKÄYTTÖ, TESTAUS, OPTIMOINTI JA ESIMERKKEJÄ RATKAISUISTA

Ominaisuuksien testaaminen oli kolmiosaista. Ensin testattiin muutokset ja ominaisuudet Unreal Enginen testiympäristössä (In-Editor Testing (Play & Simulate) in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.), joka pyörii käyttäen tietokoneen tehoja ja toistaa sen VR-laseilla. Seuraavaksi rakennettiin ohjelmisto Oculus Quest 2 -laseille ja testattiin se käyttäen pelkästään VR-laseja. Jokainen testauskerta vei täten aikaa noin viidestä kymmeneen minuuttia ja oli yksi suurimmista ajan käytöistä työn tekemisessä. Vaatimuksena kuitenkin oli, että sovellus toimii pelkästään laseilla, ilman tietokoneen avustusta. Vaatimus yksityiskohtaisista ja realistisista koneista vaikutti siihen, että tehoja laseissa ei riitä muuhun visuaaliseen toimintaan. Sujuvan käyttäjäkokemuksen varmistamiseksi testaus oli koko työn ajan läsnä kehityksessä. Jokainen muutos täytyi testata käytännössä VR-laseilla, siedettävän kuvantaajuuden ja muiden odottamattomien virheiden varalta. Viimeisenä testauksen vaiheena lähetettiin uusin versio ohjelmasta testaajalle, joka tarkasteli haluttujen ominaisuuksien toimintaa.

Ajan riittämättömyys, oli yksi suurimmista ratkaisuihin vaikuttavista tekijöistä. Käytännössä aikaa oli hieman yli kolme viikkoa saada ensimmäisestä toimivasta versiosta käytännöllinen kokemus, jonka jatkokehitys ja käyttö olisi mahdollisimman yksinkertaista. Viimeisen viikon varasin manuaalin kirjoittamiseen. Manuaalin avulla työ kyettäisiin toistamaan alusta loppuun ja sitä kyettäisiin käyttämään myös kehityksessä. Käytettävän sovelluksen optimointiin varasin kaksi päivää aikaa, mikä riitti pääominaisuuksien optimointiin melko hyvin. Suuri osa ominaisuuksista jäi kuitenkin vaille täyttä optimointia tai visuaalista viimeistelyä.

Valojen testaus ja optimointi toimivaksi oli ajallisesti pisin yksittäinen prosessi. Noin viikon ajan valaistuksen testaamista, optimointia ja käytännöllisyyttä säädettiin yleisesti, ennen kuin lopputulos miellytti ja kuvataajuus pysyi hyvänä koneiden liikkumisesta huolimatta. Valojen toiminnasta VR ympäristössä oli vaikea löytää

lähdemateriaalia ja ottaen huomioon, että suuri osa sovelluksista on tarkoitettu tietokoneelta ajettavaksi, lähdemateriaalia oli entistä vaikeampi seuloa sujuvasti. Ratkaisuna ainut, joka tuntui ajallisesti järkevältä, oli käyttää erilaisia valoja erilaisilla asetuksilla, kunnes sopivat ja toimivat valot löytyisivät. Nopeasti huomasin point light -valojen toimivan sisätiloissa parhaiten ja antavan mukavan näköisen lopputuloksen myös koneille. Varjot täytyi ottaa pois niin koneen malleista kuin valonlähteistä tai kuvataajuus laski välittömästi. Parasta määrää ja kirkkautta valoihin haettiin pitkään optimoimalla valojen määrää, kirkkautta ja ulottuvuutta. Varjojen pois ottaminen ratkaisuna oli muutenkin hyvä, koska monimutkaiset koneet loivat varjoja itsensä päälle ja häiritsivät koneen tutkimista selvästi. Valon tullessa jotakuinkin tasaisesti useasta suunnasta, koneen materiaalit näkyivät selvästi ja yksityiskohdat erottuivat hienosti.

Ensimmäinen toimiva valaistusasetelma oli yhdeksän valoa hallissa kolmessa rivissä, mutta joissain koneissa kääntäessä se oli liikaa ja aiheutti selvästi kuvataajuuden laskua. Lopputuloksena viisi valoa hallissa, jossa yksi valo antoi koneelle valon ylhäältäpäin ja neljä muuta vastakkaisista suunnista, toimi yhtä hyvin valaistuksen osalta ja selvästi paransi käyttäjäkokemusta. Kirkkaus säädettiin miellyttämään silmää, olematta liian pimeä. Ainut hieman häiritsevä asia visuaalisesti oli, että aika ei riittänyt etsimään tai tekemään minkäänlaista valojen kohdalla sijaitsevaa valaisimen static meshiä (Static Mesh Actor in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.), joten kirkkaat näkymättömät valonlähteet ovat hieman häiritsevän näköisiä suoraan kohti katsoessa.

Toiminnallisuuksista eniten kehitystyötä vaatinut yksittäinen asia oli käyttäjän liikkuminen ympäristössä. Ensimmäisenä liikkumisen muotona oli VR-templatessa mukana tuleva teleporttaus haluttuun kohteeseen. Tämä liikkumisen tapa katsotaan aiheuttavan vähiten pahoinvointia ja sopivan parhaiten uusille käyttäjille, mutta vaatimuksena oli sujuva ja portaaton liikkuminen ympäristössä. Tämän saavuttamiseksi täytyi luoda useita toiminnallisuuksia ja asettaa niille sopivat näppäi-

met (nämä toiminnallisuudet on esitetty kohdassa Lisätty toiminnallisuus ohjelmassa). Liikkumisen nopeus täytyi hioa ajatellen pahoinvointia, mutta myös käytettävyyttä. Vaikka nopeuden muuttaminen oli muuttujaa muuttamalla helppoa, sen testaaminen ei ollut pelkästään työn tekijän, vaan myös testaajan vastuulla. Täten ohjelman siirto laitteelta toiselle ja sopivan ajan löytäminen veivät myös aikaa.

Testaajan aika oli myös syy, miksi usein muutoksia, lisäominaisuuksia ja optimointia tehtiin yhtäaikaisesti useamman toiminnan kanssa kerrallaan. Esimerkiksi liikkumista ja tasojen korkeuksia kehitettiin samanaikaisesti kun valaistusta optimoitiin. Ympäristöä muokattiin ja menua viimeisteltiin samanaikaisesti kun ääniä lisättiin ympäristöön. Nämä ohjelmiston siirtämiset hoidettiin internetin välityksellä ja siirroissa ilmeni kahdesti ongelmia, jotka myös hidastivat hieman kehitystyötä.

Yksi vaatimuksista oli myös, että käyttäjän täytyi kyetä tarkastelemaan koneita muistakin kulmista kuin lattiatasosta. Aluksi ennen liikkumisen muuttamista tämä oli ratkaistu konkreettisilla tasoilla ja rampeilla koneiden ympärillä, mutta sulavan liikkumisen asettaminen näin monimutkaiseen ympäristöön VR-laseilla oli uutta ja aika hyvin rajallista, päädyttiin ratkaisuun, jossa konkreettiset rampit ja tasot korvattiin puhtaasti käyttäjän korkeuden säädöllä. Ensimmäinen versio tästä oli kolme määrättyä korkeutta ylöspäin ja sen jälkeen palautus lattiatasolle. Tämä täytti kriteerit, mutta sitä optimoitiin käyttäjäkokemusta silmällä pitäen, ja muutaman kehityskerran jälkeen tuloksena oli ylös ja alaspäin toimiva korkeuden säätö, lähes kokonaan muuttujien määräämänä. Harmillisesti optimointi jäi hieman puutteelliseksi ja siitä syystä joudutaan tekemään myös manuaalista säätöä koodiin muuttujia muuttaessa.

Liikkumisen muuttaminen erilaiseksi kuin templatessa aiheutti myös yllättävän ongelman. Teleporttaamisen esti suurin osa actoreista, mutta VRPawnia olikin huomattavasti hankalampi rajoittaa menemästä esteiden lävitse. Koska aiempi osaamiseni oli muusta kuin VR-projekteista se, miten käyttäjän estäminen esimerkiksi seinien läpi menemiseen, ei se pätenytkään VR ympäristössä. Nopein keksimäni

ratkaisu tälle ongelmalle oli käyttää TriggerBox actoreita (Trigger Volume Actors in Unreal Engine | Unreal Engine 5.1 Documentation, n.d.). Toinen skaalattiin huoneen sisämittojen mukaan ja toinen asetettiin siihen kohtaan minkä keskelle käyttäjän haluttiin palaavan, jos tämä ylittäisi ensimmäisen mainitun trigger boxin rajat. Tämä toimi käytännössä melko hyvin, ja ratkaisu jäi sovellukseen pysyvästi ratkaisemaan käyttäjän mahdollisuus mennä tarkoitetun alueen ulkopuolelle. Ainut bugi oli yhdessä tasossa oleva outo hyppiminen teleporttauksen jälkeen, mutta tämä korjaantui lähes kokonaan siirtämällä kohteeksi otettu trigger box hieman kauemmas laitteesta.

Äänten lisääminen tasoon oli viimeisimpiä lisäyksiä ja vaikka äänet eivät olleet lähelläkään täydellistä, niin ne ajoivat asiansa. Kyse oli enemmänkin demo- tai placeholder -tyyppisistä äänistä, joista toinen ajettiin taustaäänenä tehdasympäristön simuloimiseksi ja toinen konetta käännettäessä. Aiempaa kokemusta äänien lisäämisestä ei ollut, joten vaikka se oli melko yksinkertaista, tämäkin vei hieman aikaa.

Ympäristön ulkonäön hiominen oli tärkeydeltään samaa luokkaa äänten kanssa, eli toteutettiin lähes viimeisenä. Aikaa ei jäänyt juurikaan yli vaadituilta toiminnallisuuksilta ja optimoinnin jälkeen, joten ympäristö jäi hyvin pelkistetyksi. Bridgen avulla ladattiin kyllä useampia seinä ja lattia materiaaleja, sekä esimerkiksi trukkilavojen ja ilmastointiputkien malleja, mutta näitä ei päästy käyttämään ajan loppuessa. Ympäristön testaaminen ja optimointi tapahtui lähes yksinomaan muiden ominaisuuksien testaamisen yhteydessä ja tähän ei käytetty samaa aikaa kuin tärkeämmäksi priorisoituihin asioihin.

Yleisellä tasolla tiukka aikaraja sai ratkaisut, testaamisen ja optimoinnin tapahtumaan usein päällekkäin ja tästä syystä sovelluksen kehitys oli hyvin intensiivistä. Useasti ominaisuuksia lisättiin tai kehitettiin jo ennen kuin edelliset olivat käyneet testaajalla tarkistettavana. Tästä ei tässä työssä ilmennyt haittaa, mutta potentiaalinen uhka olisi se, että kehitetyn ominaisuuden pohjalta kehitetty toinen ominaisuus olisi jouduttu poistamaan ja olisi menetetty enemmän aikaa. Samaten osa

koodin ratkaisuista perustui jo ennestään hankittuun tietoon, sen sijaan että olisi opeteltu uutta ja tehokkaammin toimivaa tapaa. Nämä ratkaisut perustuvat myös ajan puutteeseen.

Tässä työssä testaaja oli käytännössä lähiesimies, jolla oli myös kyky päättää halutuista ominaisuuksista ja laadusta, esimiehen aluksi antamien ohjeiden perusteella.

Suurin painoarvo optimoinnin tasoon oli ajalla ja seuraavaksi suurin käyttäjäkokemuksella. Suurin painoarvo testauksessa oli toimivuudella ja seuraavaksi suurin käyttäjäkokemuksella. Ohjelmointiratkaisuihin, projektin ongelmien ratkaisuihin sekä ominaisuuksien lisäämisen suurin vaikuttava tekijä oli hyvin rajallinen aika, sen jälkeen käyttäjäkokemus ja viimeisenä tietotaito. Nämä asiat ajoivat kaikkea testauksessa, optimoinnissa, ongelmanratkaisussa sekä suunnitelluissa ratkaisuissa.

9 PÄÄTELMÄT

Ottaen huomioon työn tavoitteet, kehitysehdotukset ja tiukan aikataulun, työ onnistui hyvin. Intensiivisen kehityksen vaiheet olivat välillä hieman epäjohdonmukaiset, mutta sujuvan vuorovaikutuksen ansiosta tästä ei ollut suurtakaan haittaa. Tekniset ongelmat tiedostojen siirrossa oli oikeastaan ainut tekninen ongelma koko projektin aikana. Saatu palaute oli positiivista ja vaaditut ominaisuudet löytyivät sovelluksesta. Useat pienemmät ominaisuudet onnistuttiin lisäämään nopeallakin aikataululla. Esimerkiksi otettakoon keinotekoinen lasimateriaali, jonka oli tarkoitus näyttää mahdollisuus muokata laitteiden malleja myös jälkikäteen, mutta manuaalinen muokkaaminen ei ollut toivottua, joten materiaaleja ei lähdetty muokkaamaan laajemmin.

Alkuperäisen tavoitteen (Metaverse) ja nykyisten mahdollisuuksien arviointi ja toteutus jätettiin suurelta osin kehittäjälle, mikä antoi vetovastuun projektin muotoutumisesta aluksi pelkästään kehittäjän käsiin. Tämän jälkeen ensimmäinen demo, joka oli myös menestys esiteltäessä esimiehelle, määräsi lopullisen suunnan projektille. Tässä vaiheessa toinen työntekijä alkoi valvomaan projektin edistymistä, käyttäjän näkökulmaa ja antamaan kehitysehdotuksia (viitattu nimillä testaaja ja lähiesimies). Mahdollisuuksien rajaaminen oli edelleen kehittäjän vastuulla, mutta vain harva ehdotetuista ominaisuuksista ei päätynyt kehitykseen ja osaksi sovellusta. Työskentely oli hyvin sujuvaa ja tiedostojen siirto sekä testaaminen toimivat pääsääntöisesti hyvin. Palaute optimoinnin, bugien ja lisäominaisuuksien suhteen oli sujuvaa ja sovelluksen kehittäjälle annettiin kyky kertoa mihin annettu aika riittää. Lähes kaikki sujui odotettua paremmin tai odotetulla tavalla. Lopputuloksen visuaalisuus ja käyttökokemus tuntui yleisesti hyvältä ja näytti hyvältä, tästä myös saatiin positiivista palautetta.

Videot tai kuvat VR-ympäristöstä eivät vastaa itse kokemusta, mikä on huomattavasti sujuvampi ja visuaalisesti parempi, kuin siitä tallennetut esitemateriaalit. Kokemus Oculus (Meta) Quest 2 -laseilla oli erinomainen visuaalisesti ja yleiseltä kokemukseltaan.

Useasti tässä työssä on painotettu tiukkaa aikataulua ja se on välttämätöntä, koska se vaikutti tämän työn tekemisen jokaiseen vaiheeseen. Kova ajallinen paine, useat asiat, joita ei ollut aiemmin tehty, ja asiat, jotka täytyi oppia itsenäisesti sekä suuri vastuu sovelluksen lopullisesta muodosta, antoivat intensiivisen, mutta positiivisen kokemuksen vaativasta projektista. Tyypilliset projektin vaiheet jäivät lähes kokonaan pois ja useaa asiaa työstettiin samanaikaisesti. Vaati suurta mukautumista, ajallista hahmottamista ja tiukkaakin päätöksentekoa saada projekti siihen valmiuteen, jossa sen kehitys päättyi.

Työn lopputulos, aika huomioon ottaen, miellytti suuresti. Mitä olisi voitu tehdä toisin, olisi yksinkertaisesti varata aikaa enemmän tämänkaltaiselle projektille. Myös optimointi jäi kesken niin koodin kuin käyttäjäkokemuksen osalta, vaikka työ olikin pääsääntöisesti onnistunut ja hyvä.

Unreal Engine oli erittäin laadukas ja sopiva työkalu tämänkaltaisen projektin rakentamiseen. Datasmith, Bridge ja monipuoliset työkalut Unreal Enginessä antoivat kyvyn luoda nopeasti ja tehokkaasti VR sovellus Oculus (Meta) Quest 2 -laseille. Tehtävän jälkeen Unreal Engine on saanut päivityksiä, myös VR projektien tekemiseen ja nykyään esimerkiksi Lumene ominaisuus toimii hyvälaatuisilla VR laseilla. Nykyisistä ominaisuuksista saattaa löytyä muitakin muutoksia tässä työssä tehtyjen ratkaisujen osalta. Unreal Engineä käyttämällä kyetään toistamaan tämän työn kaltainen VR-projekti kohtuullisen helposti ja nopeasti.

LÄHTEET

Actors | Unreal Engine Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 18.1.2024. https://docs.unrealengine.com/5.1/en-US/actors-in-unreal-engine/

Bridge by Quixel – Unreal Engine (n.d.) Unreal Engine. Viitattu 18.1.2024. https://www.unrealengine.com/en-US/bridge

Creating Widgets in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/creating-widgets-in-unreal-engine/

Datasmith – Unreal Engine. (n.d.). Unreal engine. Viitattu 9.1.2024. https://www.unrealengine.com/en-US/datasmith

Enhanced Input in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/enhanced-input-in-unreal-engine/

Events | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/events-in-unreal-engine/

Flow Controll in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/flow-control-in-unreal-engine/

Function Calls in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/function-calls-in-unreal-engine/

In-Editor Testing (Play & Simulate) in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/in-editor-testing-play-and-simulate-in-unreal-engine/

Light Types and Their Mobility | Unreal engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/light-types-and-their-mobility-in-unreal-engine/

Lumen Technical Details in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/lumen-technical-details-in-unreal-engine/

Overview of Blueprints Visual Scripting in Unreal Engine | Unreal Enginge 5.1 Documentation. (n.d.) Unreal Engine 5.1 Documentation. Viitattu 18.1.2024. https://docs.unrealengine.com/5.1/en-US/overview-of-blueprints-visual-scripting-in-unreal-engine/

Point Lights in Unreal Engine| Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/point-lights-in-unreal-engine/

Static Mesh Actor in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/static-mesh-actors-in-unreal-engine/

Textures in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 18.1.2024. https://docs.unrealengine.com/5.1/en-US/textures-in-unreal-engine/

The Unreal Takeaway. (8.1.2023). UE5.1 META QUEST VR Development set up STEP BY STEP! [Video]. Viitattu 9.1.2024. YouTube. https://www.youtube.com/watch?v=bSS1P_ABamc

Tucci, L. & Needle, D. (18.10.2023). What is the metaverse? An explanation and in-depth guide. TechTarget. Viitattu 9.1.2024. https://www.techtarget.com/whatis/feature/The-metaverse-explained-Everything-you-need-to-know

Trigger Volume Actors in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 19.1.2024. https://docs.unrealengine.com/5.1/en-US/trigger-volume-actors-in-unreal-engine/

Unreal Engine Marketplace | Store of UE Assets for Games and 3D Rendering – UE Marketplace. (n.d.). Unreal Engine Marketplace. Viitattu 18.1.2024. https://www.unrealengine.com/marketplace/en-US/store

Unreal Engine Materials | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 18.1.2024. https://docs.unrealengine.com/5.1/en-US/unreal-engine-materials/

Unreal Engine Templates Reference | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 18.1.2024. https://docs.unrealengine.com/5.1/en-US/unreal-engine-templates-reference/

Nodes in Unreal Engine | Unreal Engine 5.1 Documentation. (n.d.). Unreal Engine 5.1 Documentation. Viitattu 18.1.2024. https://docs.unrealengine.com/5.1/en-US/nodes-in-unreal-engine/