VR-suunnittelusovelluksen prototyyppi Unreal Enginellä

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
kevät, 2021

Topi Raatikainen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Hämeenlinnan korkeakoulukeskus

Tekijä Topi Raatikainen Vuosi 2021

Työn nimi VR-suunnittelusovelluksen prototyyppi Unreal Enginellä

Ohjaajat Lasse Seppänen

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli kehittää virtuaalitodellisuutta ja visualisoituja tiloja hyödyntävä sovellusprototyyppi visuaaliseen suunnitteluun, ja samalla tutkia sovelluskehitystä Unreal Engine -pelimoottorilla sekä havainnollistaa matalan kynnyksen VR-sovelluskehityksen mahdollisuuksia. Tavoitteena oli tuottaa itsenäinen, kehitysympäristön ulkopuolella toimiva sovellus, jota olisi mahdollista käyttää konseptitodistuksena varsinaiselle, laajemmalle suunnittelusovellukselle. Työllä ei ollut toimeksiantajaa, vaan se toimii ensisijaisesti osaamista kehittävänä projektina tekijälle itselleen.

Opinnäytetyön tietopohjassa kuvataan perusteita pelimoottoreista, ja keskitytään erityisesti kehitysympäristönä toimivan Unreal Enginen lisensointimalleihin ja toimintaan. Tämän lisäksi työssä kuvataan virtuaalitodellisuuden käsitteitä ja historiaa, kuluttaja- ja ammattilaismarkkinoita sekä käyttökohteita peliteollisuuden ulkopuolella. Toiminnallisessa osuudessa kuvataan luodun sovelluksen kehitystä sen keskeisimpien ominaisuuksien kautta.

Kehitystyön tuloksena syntyneen sovelluksen tavoite toimivuudesta kehitysympäristön ulkopuolella ei toteutunut, mutta saavutettu tulos on pitkälti muilta osin ennakkosuunnitelmaa ja vaatimusmäärittelyä vastaava. Vaikka kaikkia asetettuja tavoitteita ei saavutettu, oli se kokonaisuutena tekijälleen tarkoituksenmukainen, ja osaamista ja tietotaitoa kehittävä. Lopputulos tulee toimimaan jatkokehitysalustana tuleville projekteille.

Avainsanat Pelimoottori, Unreal Engine, virtuaalitodellisuus, sovelluskehitys

Sivut 38 sivua ja liitteitä 1 sivu

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Abstract

Author Topi Raatikainen
Year 2021

Subject VR design software prototype with Unreal Engine

Supervisors Lasse Seppänen

ABSTRACT

The purpose of this thesis was to develop an application prototype for visual designing that utilizes virtual reality and visualized environments, and to research software development with Unreal Engine while demonstrating the possibilities of entry-level VR development. The goal was to produce an application that could be run independently outside of its development environment, and which could later be possibly used as a proof of concept for a broader application solution. The thesis was not commissioned by any party, and its primary goal was to educate and train the author in the topic.

In the theoretical part this thesis describes the basics of game engines and especially focuses on the Unreal Engine. Additionally, the thesis unwraps the history, markets, and applications of virtual reality outside of the gaming industry. In the practical part this thesis describes the development process of the application via its key components and functions.

The goal of an independently runnable application was not met, but otherwise the thesis mostly reached its goals. Although all the planned functions were not produced, the process was meaningful for the author and served the purpose of improving skills and know-how of Unreal Engine & VR software development. The product of this thesis process is going to be used and developed further in the author's future projects.

Keywords Game engine, Unreal Engine, virtual reality, software development
Pages 38 pages and appendices 1 page

Sanasto

Actor	Pelimaailmaan lisättävä objekti.
AR	Augmented Reality, lisätty todellisuus.
Blueprint	Unreal Enginen visuaalinen skriptausjärjestelmä, tai sen osa.
Epic Games	Yhdysvaltalainen ohjelmistotalo ja pelijulkaisija.
HMD	Head mounted display, päähän puettavat VR-lasit.
Materiaali	Objektin pinnan määritelmä, mm. heijastavuus, tasaisuus, läpinäkyvyys. Materiaalissa voi olla myös liikettä.
Node	Blueprint-järjestelmässä toiminnallisuuksia tai muuttujia sisältävä ”solmu”.
Pelimaailma	Vakiintunut termi kuvaamaan virtuaalista, pelimoottorilla luotua ympäristöä.
SteamVR	Valve Corporationin kehittämä VR laite- ja ohjelmistoprotokolla, jota mm. HTC Vive -laitteet käyttävät.
UE4	Täsmentävä viittaus Unreal Engine 4 ekosysteemiin.
Unreal Engine, UE	Epic Gamesin kehittämä pelimoottori.
Visualisointi	Tiedon tai asian esittämistä näköaistin havaitsemassa muodossa.
VR	Virtual Reality, virtuaalitodellisuus.
Widget	Käyttöliittymäelementti, jolla näytetään käyttäjälle tietoa tai jolla käyttäjä on vuorovaikutuksessa sovellukseen.

Sisällys

Kuvat ja taulukot

Liitteet

1 Johdanto

Virtuaalitodellisuus on ollut vuosikymmeniä kiehtova konsepti, ja erilaisia sitä hyödyntäviä tai sen ympärillä toimivia teollisia ja kaupallisia tuotteita on ollut olemassa jo 1950-luvulta lähtien. Kuluttajamarkkinoille asianmukaisia laitteita tuli toden teolla vasta 2016 Oculus Riftin ja HTC Viven julkaisujen myötä, ja tuon jälkeen laitteiden kirjo on vain kasvanut. Hetkellisestä buumista huolimatta VR-laitteista ei kuitenkaan vielä 2020 ollut tullut niin isoa myyntimenestystä kuin muutamaa vuotta aiemmin yleisesti oletettiin. Syynä tähän on ollut muun muassa laitteiden huomattavan korkea hinta, mutta myös heikko sisällöllinen tarjonta. Virtuaalitodellisuudella on kuitenkin paljon käyttömahdollisuuksia myös viihdeteollisuuden ulkopuolella. (Lawrie, 2020; Poetker, 2019)

HAMK Smart tutkimusyksikölle tekemäni tietojenkäsittelyn opintoihin liittyneen projektin sekä työharjoittelun myötä heräsi kiinnostukseni pelimoottoreihin ja virtuaalitodellisuusjärjestelmiin, sekä erityisesti niiden hyötykäyttöön ja mahdollisuuksiin pelinkehityksen ohella.

Tämän opinnäytetyön tarkoituksena on luoda konseptitodistus (proof of concept) virtuaalitodellisuutta hyödyntävästä suunnittelusovelluksesta sovellusprototyypin muodossa. Opinnäytetyössä kuvataan sovelluksen luomisprosessia, konseptin tuomia mahdollisuuksia ja haasteita sekä käytänteitä. Tarkoituksena on luoda aidosti jatkokehitettävä alusta, jonka päälle on myöhemmin mahdollista rakentaa laajempaa kokonaisuutta. Opinnäytetyöllä ei ole toimeksiantajaa, vaan sen aihe löytyi aiheen parissa tehdyn työharjoittelujakson aikana pöytälaatikkoon kirjoittamistani muistiinpanoista ja mahdollisista tulevista projekti-ideoista.

Opinnäytetyön keskeisimmät tutkimuskysymykset:

- Kuinka Unreal Enginellä luodaan sovellus?
- Millainen on Unreal Enginen soveltuvuus visuaaliseen suunnitteluun?
- Mitä hyötyjä tai haittoja virtuaalitodellisuus tuo visuaaliseen suunnitteluun?
- Mitä VR-sovelluksen käyttöliittymäsuunnittelussa tulee ottaa huomioon?

2 Pelimoottorit

Vaikka tässä opinnäytetyössä ei tehdäkään peliä, projektin ytimessä on pelimoottori. Sana viittaa pelinkehityksessä käytettävään työkaluun, jonka avulla voidaan hyödyntää kertaalleen luotua sovellusarkkitehtuuria luotaessa uusia pelejä ja nykyään myös hyötyohjelmia tai simulaattoreita.

Alun perin videopelejä kehitettiin laitekohtaisesti. Käytännössä tämä tarkoitti sitä, että mikäli sama peli haluttiin julkaista usealla eri alustalla, se jouduttiin kirjoittamaan jokaisella kerralla alusta asti uudestaan. Modulaarisuuden tulo pelinkehitykseen 1980-luvulla pohjusti jo konsepteja pelimoottoreille, mutta varsinaisesti termi ja käytäntö yleistyivät vasta seuraavalla vuosikymmenellä. (Buttle, 2020; Jones, 2011)

Käännekohtana pelimoottoreiden yleistymiselle viitataan usein Id Softwaren huippusuosittuihin **Doom**- ja **Quake** -peleihin jatko-osineen pelitalon lisensoidessa kehitystyökalujaan ulkopuolisille yrityksille. Laitteistojen ja 3D-tekniikan kehittyessä myös pelit monimutkaistuivat, eikä enää ollut välttämättä taloudellisesti järkevää tehdä kaikkea ohjelmointia itse. Pelimoottoreiden myötä kehittäjät pystyivät keskittymään entistä enemmän itse sisältöön ja ainutlaatuisen kokonaisuuden luomiseen. (Buttle, 2020; Gregory, 2018, s. 11)

Sittemmin pelimoottoreiden määrä ja kirjo on kasvanut huimasti. Markkinoilla on lukuisia eri käyttötarkoituksiin, laitealustoille ja pelityypeille tarkoitettuja, ja eri ohjelmointikielillä toteuttettuja pelimoottoreita, joiden lisensointimallit poikkeavat toisistaan. Suosituimmiksi ovat nousseet mm. Godot, Source, CryEngine, Unity ja Unreal Engine. Tässä opinnäytetyöprojektissa käytetään viimeksi mainittua. (*List of Game Engines*, 2020)

2.1 Unreal Engine

Unreal Engine on Epic Gamesin kehittämä, julkaisema ja ylläpitämä pelimoottori ja reaaliaikaisen 3D-renderöinnin alusta. Yhtiön vuonna 1998 julkaisemaa **Unrealia** varten kehitetty (ja siltä nimensä saanut) pelimoottori oli aikanaan tekninen edelläkävijä kuumilla FPS-pelien markkinoilla. Kehityksessä otettiin jo varhaisessa vaiheessa huomioon työkalujen

soveltuvuus myös muille pelikehittäjille, ja se loi pohjan yhtiön myöhemmälle liiketoimintamallille. (Thomsen, 2010) Kesällä 2020 Epic Games julkisti Unreal Engine 5:n, joka tullaan julkaisemaan vuoden 2021 aikana.

2.1.1 Lisensointi

Alun perin Unreal Enginen lisensointi oli pelimoottorin korkean hinnan vuoksi mahdollista vain suurille pelistudioille. Vuonna 2009 Epic Games julkaisi Unreal Engine 3 pohjaisen Unreal Development Kitin ilmaiseksi ei-kaupalliseen käyttöön, kuten esimerkiksi koulutukseen tai 3D-suunnitteluun. Kaupalliset tuotteet tuli edelleen lisensoida lisenssiehtojen mukaisesti. (IGN, 2009)

Vuonna 2014 julkaistu Unreal Engine 4 uudisti lisensointimallin täysin. Nyt koko pelimoottori lähdekoodeineen oli saatavilla 19 Yhdysvaltain dollarin (USD) kuukausihintaan, ja mikäli tuotteesta tuli kaupallinen, sen bruttotuloista perittiin 3000 USD ylittävästä osasta 5 % provisio. Sittemmin tuo raja on noussut.

Lähes tarkalleen vuotta myöhemmin, maaliskuussa 2015 Epic Games ilmoitti, että kuukausimaksullisesta mallista luovutaan, ja UE4 itsessään on saatavilla kokonaan ilmaiseksi. Epic Gamesin perustaja Tim Sweeney (2015) kiteytti asian tiedotteessaan: ”The state of Unreal is strong, and we’ve realized that as we take away barriers, more people are able to fulfill their creative visions and shape the future of the medium we love. That’s why we’re taking away the last barrier to entry, and going free.” Opinnäytetyön kirjoitushetkellä Unreal Enginestä on tarjolla neljä lisensointimallia (Taulukko 1).

Taulukko 1 Unreal Enginen lisensointimallit (Epic Games, 2021a, 2021b)

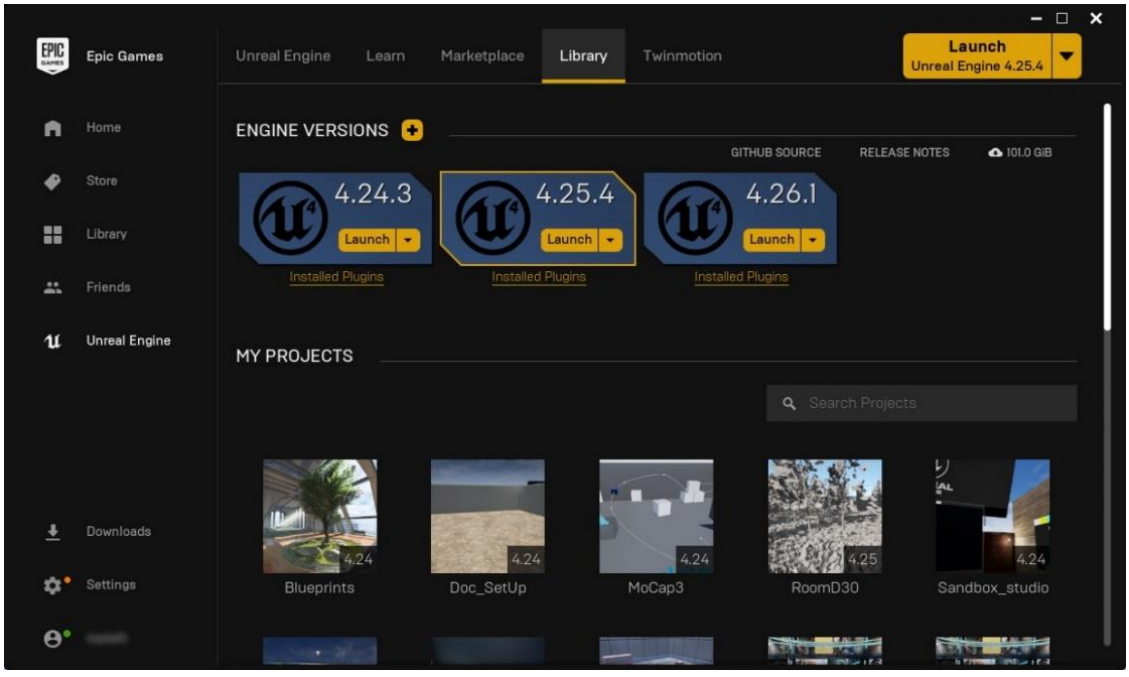
Lisensointimalli	Mihin käyttöön	Kustannukset
Creators License	Sisäisiin, ei kaupallisiin projekteihin. Ei pelinkehitykseen.	Ilmainen, rojaltivapaa.
Publishing License	Pelinkehityksen perusmalli pieniin projekteihin.	Ilmainen käyttää, mutta 5 % provisio 1 milj. USD ylittävältä osalta.
Unreal Enterprise Program	Paikkakohtainen lisenssi (per-seat license) muuhun	1500 USD

	kuin pelinkehitykseen. Läheisempi yhteistyö Epic Gamesin kanssa.	
Custom License	Kaiken kattava, ehdoiltaan ja hinnaltaan yksilökohtaisesti neuvoteltava lisenssi.	Sopimuskohtaiset kustannukset.

2.1.2 Epic Games Launcher & Unreal Editor

Unreal Engineä käytetään yleensä Epic Games Launcher- ja Unreal Editor -ohjelmien kautta. Epic Games Launcherilla hallinnoidaan pelimoottorin latauksia ja asennuksia, sekä käynnistetään Unreal Editor. Samaan aikaan voi olla asennettuna usea eri versio pelimoottorista (Kuva 1); eri versioilla luodut projektit eivät toimi hyvin tai ollenkaan ristiin. Launcherin kautta on myös mahdollista selata UE Marketplace -kauppapaikkaa ja ostaa tai lunastaa ilmaista lisäsisältöä omiin projekteihin.

Kuva 1 Unreal Engine Epic Games Launcherissa

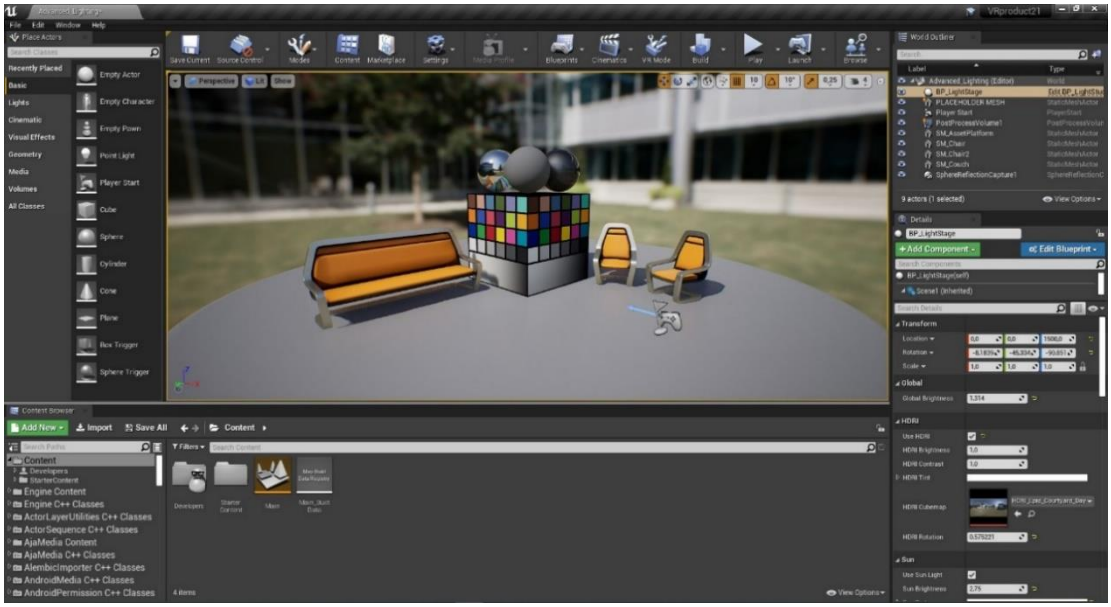


Epic Games Launcher ei kuitenkaan ole välttämätön Unreal Editorin käytössä. Vaihtoehtona on ladata Unreal Enginen viimeisimmän version lähdekoodi GitHubista, ja koota editori itse.

Näin on myös mahdollista tehdä muutoksia ja parannuksia itse moottoriin, sekä saada viimeisin päivitys nopeimmin. Lisävaatimuksena on kuitenkin Visual Studio -ohjelmankehitysympäristö. (Epic Games, n.d.-a)

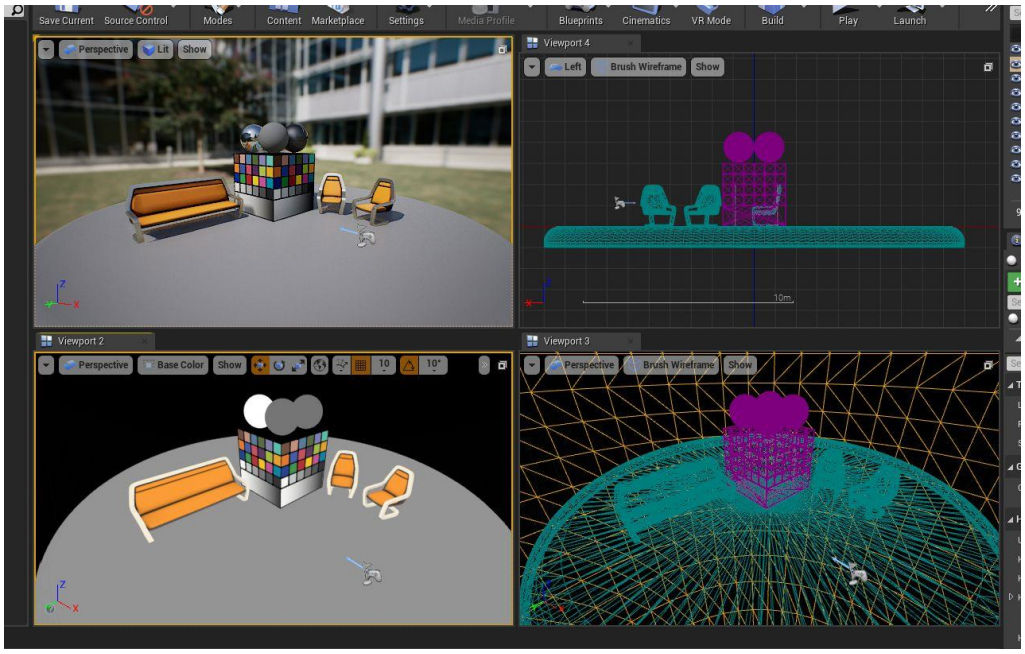
Unreal Editor on nimensä mukaan varsinainen työkalu pelimoottorin käyttöön. Käyttöliittymän näkymä on jaettu moniin työkaluvälilehtiin / ikkunoihin, joiden sijaintia ruudulla voidaan vaihtaa, tuoda näkyviin tai piilottaa tarpeen ja mieltymyksen mukaan. Eri tilanteisiin sopivia asetteluita voidaan tallentaa, ja ottaa käyttöön nopeasti tarvittaessa. Oletuksena näkymä on kuitenkin Kuva 2 mukainen.

Kuva 2 Unreal Editorin perusnäkymä



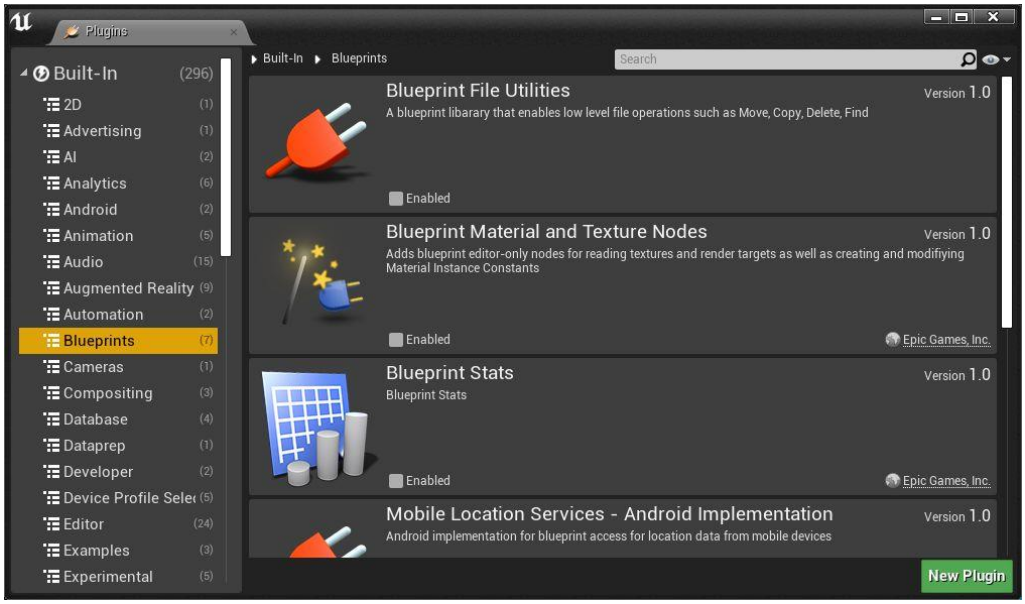
Kuva 2 keskellä olevaa näkymä pelimaailmaan voidaan katsoa yhdestä neljään eri ikkunan, eli **viewportin** läpi. Kuva 3 esimerkin mukaan viewportien näkymä voi olla kolmiulotteinen (vasen yläkulma) tai ortografinen (oikea yläkulma). Näiden ohella kolmiulotteisesta näkymästä voidaan näyttää esimerkiksi vain pohjavärit (vasen alakulma) tai rautalankamalli (oikea alakulma); näkymiä on lukuisia lisää. Viewporteihin on myös mahdollista tuoda kuva erillisistä, pelimaailmaan sijoitettavista virtuaalikameroista.

Kuva 3 Näkymä kenttään jaettuna neljään viewporttiin



Unreal Enginen ja Editorin toiminnallisuuksia on mahdollista laajentaa lisäosilla. Vakioasennuksen mukana tulee paljon laajennuksia, joista vain projektityyppikohtaiset ovat oletuksena lisättynä projektiin. Kaikki ladatut laajennukset löytyvät Plugins Browserin kautta ryhmiteltynä eri kategorioihin (Kuva 4), josta ne voidaan tarvittaessa liittää projektiin. Lisää laajennuksia voi hankkia Unreal Enginen kauppapaikalta, mutta niitä on myös mahdollista tehdä ja julkaista itse. (Epic Games, n.d.-c)

Kuva 4 Unreal Editor Plugins Browser



2.1.3 Blueprint Visual Scripting

Toimintojen ohjelmoimiseen Unreal Enginessä on kaksi tapaa: joko C++ -kielisillä skripteillä tai visuaalisesti ohjelmoimalla Blueprint Visual Scriptingillä. Tehokkainta on kuitenkin käyttää näitä yhdessä.

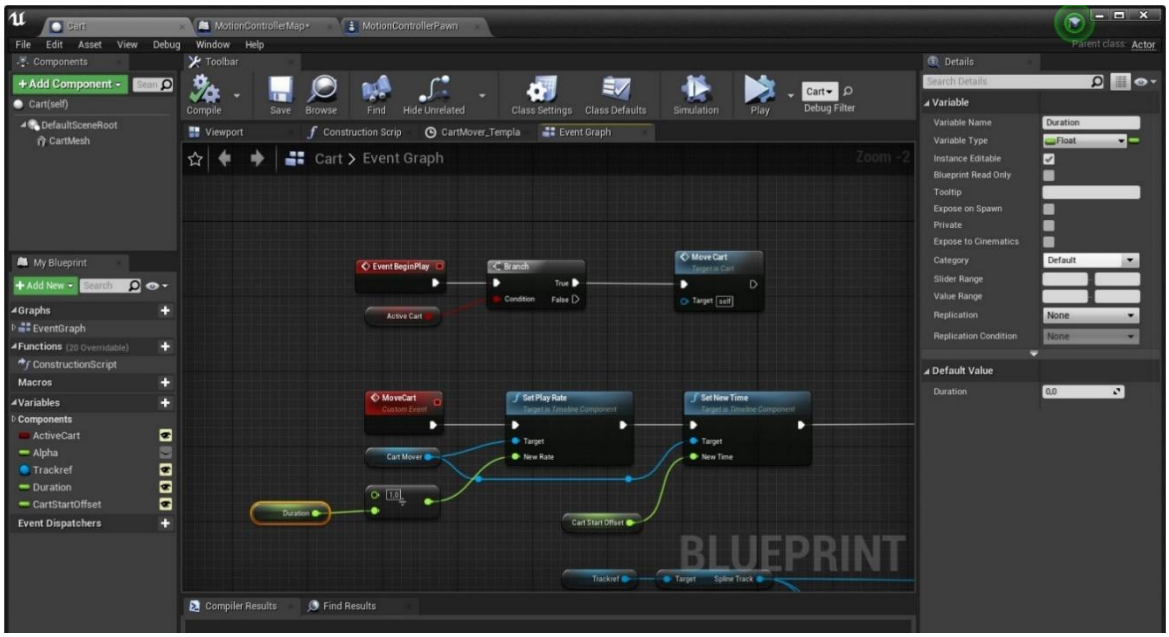
Blueprint Visual Scripting on oliopohjainen skriptikieli. Sen avulla on mahdollista luoda kokonainen ohjelma tai peli varsinaisesti kirjoittamatta riviäkään ohjelmakoodia perinteiseen tapaan, mutta käsitys ohjelmoinnin perusteista ja logiikasta on silti hallittava. Tavallisesti tähän järjestelmään viitataan lyhyemmin sanalla blueprint, mutta termiä käytettäessä on syytä huomioida konteksti, sillä blueprinttejä on eri tyyppisiä käyttötarkoituksen mukaan (Epic Games, n.d.-d):

- **Level Blueprint**: Koko käsittelyssä olevaa kenttää tai tasoa koskeva blueprint, joka luodaan automaattisesti ja on yksilöllinen jokaiselle kentälle. Esimerkiksi kenttäkohtaiset tapahtumat, tai kenttäkohtaisiin objekteihin liittyvät toiminnot luodaan usein tänne.
- **Blueprint Class**: Luokkakohtainen blueprint, jolla määritetään esimerkiksi jonkun tietyn esineen tai actorin ominaisuuksia ja toiminnallisuuksia, ja josta voidaan luoda sitten omia instanssejaan. Yleisin tyyppi puhuttaessa blueprinteistä täsmentämättä tarkemmin. Käytetään usein lyhennettä **BP**.
- **Data-Only Blueprint**: Blueprint Class johdannainen, jolla voidaan varioida vanhempansa ominaisuuksia, mutta ei lisätä mitään uutta.
- **Blueprint Interface**: Ei sisällä omaa koodiaan, mutta toimii nimensä mukaan liittymänä muiden blueprinttien välillä.
- **Blueprint Macro Library**: Kokoelma blueprint-graafeja, joihin voidaan viitata muissa blueprinteissä. Mikäli esimerkiksi tietyt toiminnot toistuvat useissa blueprinteissä, voi olla hyödyllistä koota niistä Macro Library ja hyödyntää sitä.

Ohjelmoiminen blueprinttejä käyttäen tapahtuu yhdistämällä **solmuja** (node) toisiinsa **johdoilla** (wire). Nodeja on monia erilaisia riippuen niiden toimintatavasta ja roolista. Ne voivat olla viittauksia muuttujiin, suoritettavia metodeja, operaattoreita tai vaikkapa laitteelta tulevia syötteitä. Valikoima nodeja voidaan koota metodiksi, joka puolestaan on helpompi ja selkeämpi toistaa blueprintin sisällä tarvittaessa.

Level ja Class blueprinttejä muokataan omassa ikkunassaan Blueprint Editorissa (Kuva 5). Eri blueprintit ja niiden sisältämät metodit ja graafit avautuvat ikkunaan omille välilehdilleen. Muun Unreal Editorin tapaan Blueprint Editor on näkymän osalta modulaarinen, eli työkaluikkunoiden paikkaa voidaan tarvittaessa vaihtaa.

Kuva 5 Kuvakaappaus Blueprint Editorista



2.2 Käyttökohteita peliteollisuuden ulkopuolella

Alun perin pelinkehitykseen kehitetyt pelimoottorit ovat vuosien ja julkaisuversioiden myötä monipuolistuneet kehitystyökaluiksi elokuvateollisuuteen ja TV-tuotantoihin, visuaaliseen suunnitteluun, harjoitussimulaatioihin ja tekniseen suunnitteluun. Usein näissä tapauksissa käytössä on myös joku virtuaalitodellisuusjärjestelmä. Esimerkiksi Unreal Engine on ollut käytössä muun muassa 2019 alkaneessa suositussa Star Wars -televisiosarjassa The Mandalorian ja HAMKin VATTU – Valmistavan teollisuuden virtuaaliset tuotteet -hankkeessa 2019-2020. (Farris, 2020; HAMK, 2019)

Mandalorianin kaltaisissa elokuva- ja TV-tuotannoissa tietokoneella luotuja erikoistehosteita on käytetty jo kauan, mutta ne on perinteisesti lisätty (useimmiten) vihreää tai sinistä taustaa vasten näyteltyihin kohtauksiin jälkituotannossa. Pelimoottoreiden käytön etu perustuu ennen kaikkea reaaliaikaiseen renderöintiin ja dynaamisiin ympäristöihin, jotka

voidaan kuvata samaan aikaan tosimaailman näyttelijöiden kanssa, ja joita voidaan tarvittaessa muokata tuotannon ollessa kuvausvaiheessa. Tällä on mahdollista tehostaa kokonaisprosessia ja saavuttaa paremmin halutunkaltainen lopputulos. (Farris, 2020; Postpace, 2020; Schalk, 2020)

Pelimoottoreiden hyödyt tulevat esille myös teknisessä suunnittelussa. Esimerkiksi 3D-CAD -mallien eli tietokoneavusteisten teknisten piirustusten tuomisella tehokkaasti pelimoottorin käyttöön on mahdollista toteuttaa tuotteiden valmistamista ja testaamista virtuaalisesti ennen fyysisten prototyyppien rakentamista. Suunnittelu ei kuitenkaan rajoitu vain pieniin prototyyppeihin vaan pelimoottoreita käytetään myös isojen rakennushankkeiden parissa. Esimerkiksi ruotsalainen suunnittelutoimisto AFRY on käyttänyt Unreal Engineä Ruotsiin rakentuvan neuronitutkimuslaitoksen, ESS:n sekä maan ensimmäisen suurnopeusrautatien The East Linkin suunnittelussa ja visualisoinnissa. (Ahola, 2020; Helmbäck, 2020; Pimentel, 2019; Torkkel, 2019)

Vuonna 2020 merkittävä paikkatietojärjestelmiä ja palveluita toimittava yritys Esri julkaisi Unitylle ja Unreal Enginelle oman, opinnäytetyön kirjoitushetkellä vielä beta-vaiheessa olevan lisäosan 3D GIS-datansa tuomiseksi kyseisillä pelimoottoreilla käytettäväksi ja visualisoitavaksi. (Hansen, 2020)

3 Virtuaalitodellisuus

Virtuaalitodellisuuden määrittely ja etenkin termin kattavuus voi vaihdella suuresti riippuen siihen viittaavasta tahosta. Siinä missä esimerkiksi kuluttajat ja media määrittävät kokemusta lähinnä pelien ja päähän asetettavan laitteen kautta, on kehittäjillä ja tutkijoilla toisenlainen ja ennen kaikkea laajempi näkemys. Määrityksestä on kuitenkin olemassa yleisiä vakiintuneita käytänteitä. (Sherman & Craig, 2018)

Kirjassaan Sherman ja Craig (2018, ss. 6–15) listaavat viisi keskeistä elementtiä, jotka yhdessä luovat virtuaalitodellisuuden kokemuksen:

Osallistujat ja **kehittäjät** ovat kolikon kaksi puolta. Osallistujat kokevat jokainen omalla tavallaan ja yksilöllisesti sen, mitä kehittäjä tai kehittäjät ovat luoneet. Nämä kaksi eivät välttämättä ole suoraan vuorovaikutuksessa keskenään, mutta kokonaisuuden kannalta molemmat osapuolet ovat oleellinen osa virtuaalitodellisuutta.

Virtuaalimaailma on virtuaalitodellisuuden sisältö, joka ei olemassa ollakseen tarvitse fyysisiä laitteita tai vuorovaikutusta. Pelkkä ajatus, mutta myös esimerkiksi näytelmä tai kirjan tarina kuvastavat virtuaalimaailmaa.

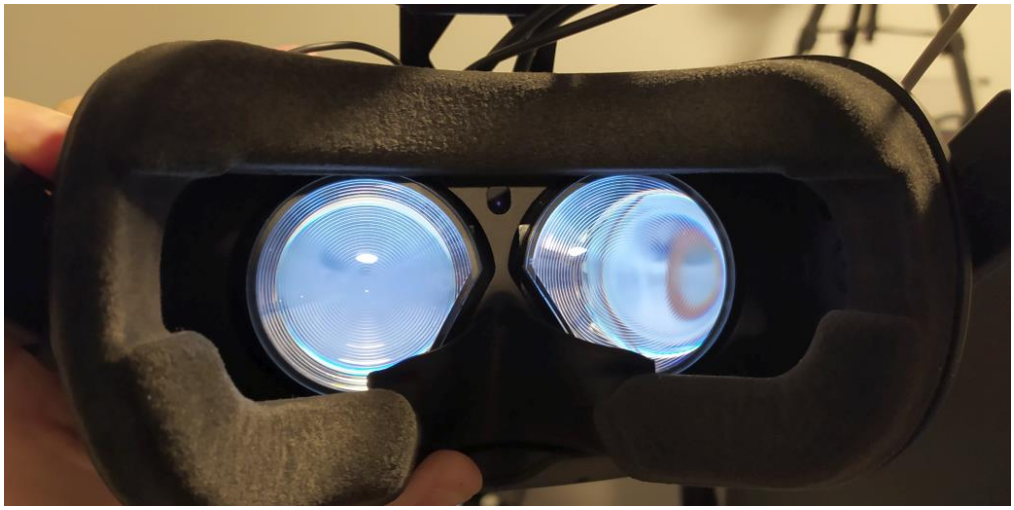
Immersio tarkoittaa käsiteltävään asiaan syventymistä ja sen maailmaan uppoutumista tavalla, joka luo kokemuksen tilaan kuulumisesta. Tunne voi olla psykologinen tai fyysinen, tai molempia.

Vuorovaikutteisuus virtuaalimaailman kanssa luo immersiivisen kokemuksen virtuaalitodellisuudesta. Ilman vuorovaikutusta virtuaalimaailmaa voi kokea esimerkiksi kirjaa lukemalla, mutta tällöin kokemus on vain yksisuuntainen; virtuaalitodellisuudessa käyttäjä voi vaikuttaa siihen tai sen sisältämiin elementteihin.

Yleisesti virtuaalitodellisuutta kuitenkin käytetään ja se käsitetään juuri päähän puettavan laitteen ja ohjaimien kautta. Ne perustuvat laitteiden sijainnin ja asennon paikannukseen, jolloin käyttäjän liike välittyy virtuaalimaailmaan ja luo vaikutelman siellä liikkumisesta sekä ennen kaikkea katseen suunnan kohdistumisesta. Tuntemus liikkumisesta ja olemisesta kolmiulotteisessa ympäristössä luodaan HMD:ssä stereokuvalla. Siinä molemmille silmille

näytetään omilla näytöillään kaksiulotteinen kuva hieman eri perspektiivistä (Kuva 6), jotka tämän jälkeen yhdistyvät näköaistimuksena yhdeksi kolmiulotteiseksi kuvaksi. (Haggrén, 2003)

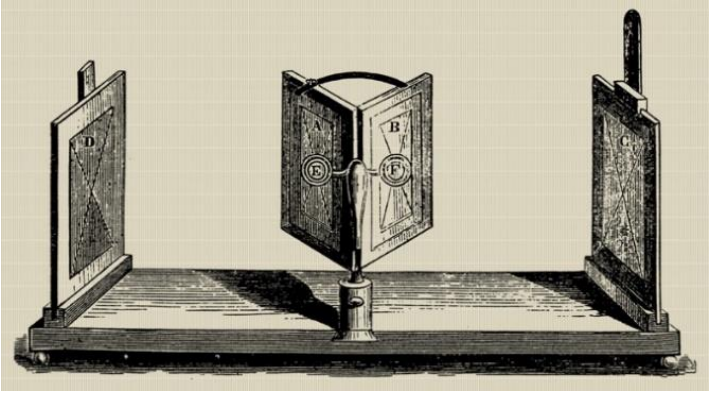
Kuva 6 HTC Vive HMD:n näytöt



3.1 Virtuaalitodellisuuden historiaa

Nykymuotoisen, päälle puettavien laitteiden kautta koettavan virtuaalitodellisuuden mahdollistavan teknologian historia ulottuu vuoteen 1838, jolloin Charles Wheatstone julkaisi havaintonsa tavasta nähdä kaksi näennäisesti samanlaista, mutta hieman eri kulmista kuvattua kaksiulotteista kuvaa yhtenä, kolmiulotteisena kuvana. Tämä tapahtui alun perin Wheatstonen rakentaman ensimmäisen **stereoskoopin** avulla, joka oli pöydälle asetettava laite (Kuva 7) jossa peilien ja kuvien asettelun myötä näköaistimus kahdesta hieman eri kulmasta otetusta tai piirretystä kuvasta muodostuvat aivoissa yhdeksi kolmiulotteiseksi kuvaksi. Myöhemmin samaan ideaan perustuvia, mutta käsissä pidettäviä laitteita valmistettiin monien kehittäjien toimesta, ja niistä tuli suosittuja 1800-luvun puolivälissä. Suosio jatkui 1939 julkaistun View-Masterin myötä vielä uudelle vuosituhannelle (Kuva 8). (C. Thompson, 2017)

Kuva 7 Charles Wheatstonen kehittämä stereoskooppi (n.d.)



Kuva 8 View-Master mallit G (1959–77) ja E (1955–61)

Fiktion keinoin, mutta yllättävän osuvasti nykyisenkaltaista virtuaalitodellisuutta kuvasi Stanley G. Weinbaum vuonna 1935 julkaistussa tarinassaan **Pygmalion's Spectacles**. Siinä silmille asetettavien lasien kautta oli mahdollista näkö-, tunto-, kuulo- ja makuaisteilla kokea ennalta luotua tarinaa. (Weinbaum, 2007)

Ensimmäinen varsinainen virtuaalitodellisuuslaite oli Morton Heiligin 1962 patentoima **Sensorama Simulator**. Laiteen toimintaperiaate oli stereoskooppisen, liikkuvan kuvan ohella tuottaa fyysisiä aistiärsykkeitä luomaan immersio esitettävän lyhytelokuvan kanssa. Ulkoasultaan laite muistutti myöhempien aikojen peliautomaatteja istuimineen, käsikahvoineen ja erikoisine näyttölaitteineen. (Turi, 2014)

Päälle puettavien tai päähän asetettavien HMD tai niitä muistuttavien laitteiden aika oli 1960-luvulla. Laitteet olivat edelleen kömpelöitä, mutta niihin kehitettiin nyt myös liikkeen seurantaa. Päivänvalon näki myös ensimmäinen lisätyn todellisuuden (AR) laite, Ivan Sutherlandin -68 kehittämä **The Sword of Democles**. Raskas, kattoon kiinnitetty HMD tuotti

käyttäjän näkökenttään tietokonegrafiikkaa rautalankamalleina. (Poetker, 2019; Sutherland, 2018)

Populaarikulttuuriin virtuaalitodellisuus tuli 1980-luvulla science fiction -elokuva **Tron** myötä, ja pikkuhiljaa termi VR alkoi yleistyä. Termin popularisoijana pidetään yleisesti Jaron Lanieriä, jonka 1984 perustama VPL Research toi markkinoille ensimmäiset kaupalliset virtuaalitodellisuutta hyödyntävät tuotteet, kuten hansikkaan mallisen syöttölaiteen ja EyePhone HMD:n (Kuva 9) ja 3D-mallinnusmoottorin. (Poetker, 2019; Virtual Reality Society, n.d.)

Kuva 9 VPL Researchin EyePhone ja DataGlove (Pape, 1999)

3.2 VR:n markkinat

Monien teknologisten innovaatioiden tapaan myös virtuaalitodellisuutta hyödyntävät laitteet löysivät tiensä ennen pitkää kuluttaja- ja yritysmarkkinoille sekä sotateollisuuteen; aiheeseen liittyy jonkin asteinen kaupallisuus jo varhaisista stereoskoopeista lähtien.

3.2.1 Kuluttajamarkkinat

Kuluttajamarkkinoille VR tuli tuotteiden muodossa 1980-luvulla, kun peliteollisuudessa otettiin ensiaskeleita Vectrex 3D Imager- ja Sega 3-D Glasses -lasien sekä Nintendo Power Glove -hansikasohjaimen myötä. Näistä mitkään eivät kuitenkaan olleet pitkäikäisiä. 1990-

luvulla markkinoille oli kuitenkin vielä tulijoita The Virtual Groupin tuodessa pelihalleihin Virtuality-kolikkopeliautomaatit ja Nintendon julkaistessa Virtual Boy -pelikonsolinsa (Kuva 10) sekä Segan kehitellessä omaa laitteistoaan, mutta markkinoille päässeiden laitteiden kaupallisesti pahoin epäonnistuttua alan kiinnostus virtuaalitodellisuutta kohtaan katosi vuosiksi. Syynä tähän pidetään teknologian vielä riittämätöntä tasoa ja toisaalta kalliita kustannuksia, mutta myös internetin tuloa massojen saataville, joka vei markkinoiden huomion uuteen tekniikkaan. (Evenden, 2016; Fowle, 2015; Robertson & Zelenko, 2014; Vectrex Museum, 2012)

Kuva 10 Nintendo Virtual Boy

Virtuaalitodellisuuden paluu kuluttajamarkkinoille tapahtui pitkälti 2010-luvulla. Vuosikymmenen alussa Palmer Luckeyn kehitettyä ensimmäisen Oculus-prototyypin kiinnostus virtuaalitodellisuuden ympärillä heräsi taas eloon. Sen jälkeen kehitys on ollut nopeaa: Vuoden 2016 aikana markkinoille tulivat niin PC-liitännäiset VR-järjestelmät Oculus Rift ja HTC Vive kuin Sonyn PlayStation 4 -lisälaite PSVR. Sittemmin markkinoille on tullut myös muiden kehittämiä laitteita. (GlobalData Technology, 2020)

Opinnäytetyön kirjoittamishetkellä vuonna 2021 suurin julkinen huomio aiheesta on kuitenkin jo hälvennyt. Uudet laitteistot ovat vihdoin teknisesti soveltuvia immersiivisen virtuaalikokemuksen luomiseen, mutta niiden korkea hinta ja jossain tapauksissa myös

laitteistovaatimukset rajoittavat asiakaskuntaa. Laitekanta kehittyy kuitenkin edelleen, ja uusia laitevalmistajia on tullut mukaan. (Lawrie, 2020)

Virtuaalitodellisuuden uuden tulemisen jälkeen modernit laitteistot ovat olleet toimintaperiaatteiltaan hyvin samankaltaisia kuin jo 30 vuotta aiemmin: Asentoa ja sijaintia paikantava HMD, joka kytketään johdolla varsinaista laskentaa suorittavaan tietokoneeseen. Tämä on mahdollistanut korkeatasoisen graafisen ulkoasun HMD-laitteiden näyttötarkkuuden puitteissa, mutta samalla tehden käytöstä paikka- ja laitesidonnaista sekä rajoittaen liikkumisen vapautta. (Paule, 2020; Robertson & Zelenko, 2014)

PC-liitännäisten VR-järjestelmien ohella markkinoilla on jo vuodesta 2015 ollut myös ns. erillisjärjestelmiä sekä älypuhelimiin perustuvia ratkaisuja. Näissä keskeisenä ideana on oma ekosysteeminsä, jossa kaikki toiminnot sijaitsevat HMD:ssä. Yksinkertaisimmillaan kyseessä on ollut Google Cardboard, jossa Android-sovelluksella mikä tahansa sovellusta tukeva älypuhelin muuttuu ”virtuaalilaseiksi” sen asetettaessa vaikkapa nimensä mukaisesti pahviseen tai 3D-tulostettavaan, päähän asetettavaan kehikkoon. Kehityslinjan toisessa päässä on aiemmin mm. Rift- ja Rift S -laitteistaan tunnettu Oculus omalla Quest-sarjallaan, joka on oma itsenäinen VR-alustansa täysverisine HMD-laseineen, ohjaimineen ja sijainnin seurantoineen. Toisin kuin PC-liitännäisissä laitteissa, itsenäiset laitteet eivät vaadi erillisiin kameroihin tai lasermajakoihin perustuvaa käyttöaluetta, joten laitteita voi käyttää missä vain. Tämän lisäksi itsenäiset laitteet ovat hintatasoltaan kiinteitä huomattavasti edullisempia. Varjopuolena on mahdolliset tekniset rajoitukset laitteen laskentatehon, mutta myös rajoitetusti laitekohtaisten sovellusten puolesta. (Paule, 2020; Ratushnyi, n.d.; Wikipedia, 2021)

3.2.2 Ammattiratkaisut

Peliteollisuuden ulkopuolella virtuaalitodellisuuden tutkimuksella ja tuotekehityksellä on alusta alkaen ollut vahvoja hyötynäkökulmia myös muun muassa teollisuudessa ja sotilasteknologiassa. Ensimmäiset käyttökohteet löytyivät sotilaslentokoneista Thomas Furnessin 1960–80 luvuilla Yhdysvaltain ilmavoimille kehittämistä lentosimulaattoreista ja lentäjän kypärään asennettavista näytöistä. Furnessin mukaan tulosten esittelyn aikaan esiin nousi kysymyksiä kehitetyn teknologian mahdollisuuksista mm. lääketieteessä tai

pelastustoiminnassa. 1980-luvulla NASA oli kiinnostunut tekniikasta, ja kehitteli omassa tutkimusyksikössään ratkaisua korvata avaruuskävelyt aseman sisältä etäohjattavilla roboteilla, hyödyntäen tuntoaistiin perustuvia käsineitä ja HMD-laitteita. (Evenden, 2016; Furness, 2015)

90-luvun puolivälissä alkaneen ns. ”VR-talven” aikana termi pitkälti katosi mediasta ja kuluttajamarkkinoilta, mutta tekniikkaa ja sovelluksia kehittävät yritykset monelta osin jatkoivat toimintaansa, joskin kutistunein resurssein. Näihin aikoihin erilaiset armeijan koulutussimulaatiot yleistyvät Yhdysvalloissa, mutta myös virtuaalitodellisuuden mahdollisuuksia lievittää traumaperäisen stressihäiriön oireita alettiin tutkia. (Aitoro, 2016; Robertson & Zelenko, 2014; Tomos Morgan, 2019)

Uuden VR-buumin myötä 2010-luvun puolivälin jälkeen uusien valmistajien tuodessa markkinoille entistä tehokkaampia ja tarkempia, mutta myös edullisempia kuluttajalaitteita, ja sitä myötä myös sovelluskehittäjien kiinnostuessa tekniikasta on virtuaalitodellisuus uusien aluevaltauksien äärellä. Esimerkiksi elokuva- ja muu videotuotantoja tekevä teollisuus hyödyntää pelimoottoreiden ohella VR-järjestelmiä muun muassa mahdollisuudessa ohjata ja synkronoida virtuaaliset ja tosimaailman videokamerat saumattomaksi kokonaisuudeksi tai tutkia virtuaalisia kuvauspaikkoja paikan päällä. (Epic Games, n.d.-e; VR Sync, 2020)

Erilaiset koulutussimulaattorit sekä suunnittelusovellukset vaikkapa auto- ja rakennusteollisuuteen ovat jo nyt VR-tekniikan potentiaalisia markkinoita. Alalla toimivat laite- ja sovelluskehittäjät näkevät Unityn tuottamassa katsauksessa (2020) myös lisätyn todellisuuden (AR) tuovan lähitulevaisuudessa uusia liiketoimintamahdollisuuksia. (S. Thompson, 2019)

4 Kehittämistyön tavoite, tarkoitus ja menetelmät

Opinnäytetyön tavoitteena on rakentaa virtuaalitodellisuutta hyödyntävä sovellus tai sovellusprototyyppi Unreal Enginellä. Termi riippuu siitä, mistä näkökulmasta sitä katsotaan: teknisesti ottaen tarkoitus on luoda ”koottu” (built), itsenäinen eli ilman Unreal Editoria toimiva sovellus. Toisaalta sovelluksen lähtökohta on toimia eräänlaisena konseptitodistuksena ja mahdollisena jatkokehitysalustana, eikä siitä opinnäytetyöprosessin puitteissa tavoitella valmista tuotetta.

Sovelluksessa liikutaan visualisoidussa huoneistossa, jossa ennalta määritettyjen kohteiden ja esineiden, kuten vaikkapa seinän tai kaapistojen ovien väriä tai materiaalia voidaan vaihtaa. VR-tilassa ohjaimesta lähtevällä valintasäteellä osuttaessa kohteeseen, johon voidaan vaikuttaa, sen ääriviivat korostetaan. Korostettua objektia klikatessa aukeaa valikko, jossa kyseisen objektin ominaisuuksiin voidaan vaikuttaa. Visualisoidussa tilassa on useita vuorovaikutettavia kohteita. Toteutettavien ominaisuuksien määrä riippuu lopulta kehitystyön etenemisestä ja sitä myötä käytössä olevasta ajasta.

Tuotettavan sovelluksen tarkoitus on havainnollistaa kuluttajamarkkinoilla olevien, tai jo poistuneiden virtuaalitodellisuusjärjestelmien käyttöä sovelluskehityksessä, ja tuoda esiin tekniikan tuomia mahdollisuuksia matalalla kynnyksellä: kehitysympäristö on käytössä maksuton, VR-järjestelmänä käytössä on markkinoilta jo poistunut ensimmäisen sukupolven HTC Vive, ja useita vuosia vanha PC-laitteisto ei suorituskykynsä puolesta vastaa nykypäivän tehokkainta kärkeä.

Kehitystyökaluna HTC Vive on iästään ja suorituskyvystään huolimatta perusteltu, sillä ollen osa SteamVR-protokollaa, on sovelluksen mahdollinen jatkokehitys ja testaus muilla SteamVR-laitteilla, kuten uudemmilla Vive- tai Valve Index -laitteilla helpompaa.

Työllä ei ole toimeksiantajaa, vaan sen on tarkoitus toimia osaamista ja tietotaitoa kehittävänä projektina työn tekijälle. Ennen kehitysprojektin aloitusta projektille luodaan vapaamuotoinen vaatimusmäärittelydokumentaatio tekemisen tueksi.

5 Sovellusprototyypin kehitys

Kehitysprojektin alussa projektista luotiin vapaamuotoinen vaatimusmäärittelydokumentti, jotta projektin skaala on mahdollista pitää hallinnassa ja rakenne selkeänä.

Suunnitelman mukaan vähintään toteutettavat ominaisuudet:

- Visualisoidussa tilassa liikkuminen
- Ennalta valikoitujen objektien valinta ja korostus
- Objektin materiaalin vaihto
- Objektin mallin vaihto
- Selkeä ja yksinkertainen käyttöliittymä

Näiden ohella mahdollisia lisätoimintoja, jotka toteutetaan suunnitelman mukaisessa järjestyksessä ajan niin salliessa, ovat:

- Ennalta valikoitujen objektien siirto
- Valaistuksen säätö
- Näkyviin kytkettävä mittaruudukko
- Sovelluksen käyttö ilman VR-laitteistoa

Kun halutut ominaisuudet on saatu toteutettua yksittäin tai yhdessä, ne on tarkoitus siirtää visualisoituun tilaan, ja yhdistää toimivaksi kokonaisuudeksi. Tällaisena tilana voidaan käyttää joko itse luotua huoneistoa, valmista Unreal Engine Marketplacesta saatavaa kokonaisuutta tai molempia. Prioriteetti on kuitenkin toiminnallisuuksien kehittämisessä.

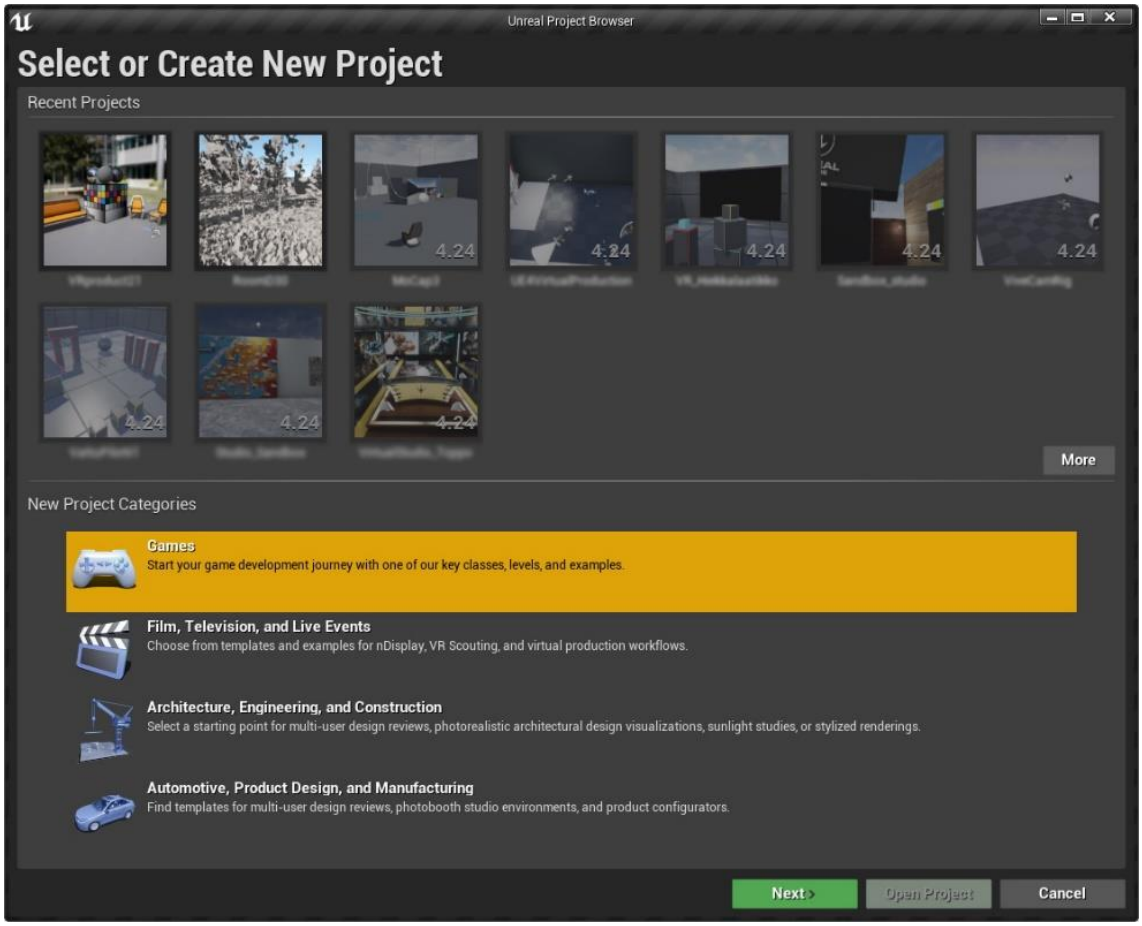
Huomionarvoista on, että vaikka tekijällä on jonkin verran aiempaa kokemusta Unreal Enginen käytöstä, ovat monet nyt toteutettavat toiminnot täysin uusia eikä niiden vaatima toteutustapa ole todennäköisesti ennalta tiedossa.

Opinnäytetyön kirjoitushetkellä pelimoottorin viimeisin julkaistu versio on 4.26.1, mutta käytännön osuudessa kehitettävä sovellus toteutetaan versiolla 4.25.4 paremman taaksepäin yhteensopivuuden varmistamiseksi tekijän edellisten projektien sekä UE-yhteisön keskustelupalstoilla esitettyjen ja ratkottujen ongelmien kanssa, jotka toimivat käytännön läheisinä lähteinä ja muistin virkistyksenä toimintoja luotaessa.

5.1 Uuden UE-projektin luominen

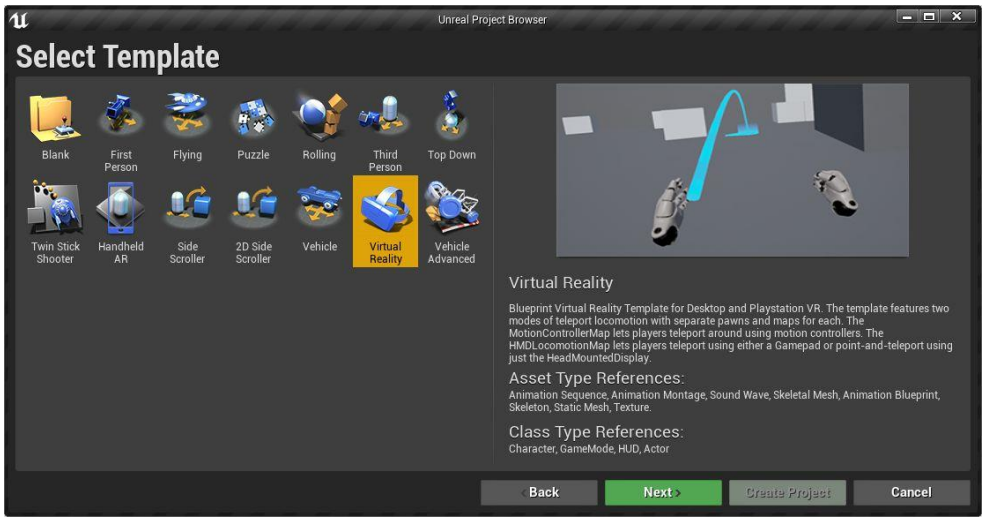
Kun Unreal Engine käynnistetään, ensimmäisenä näkymänä avautuu **Project Browser**, jonka kautta voidaan avata joku projektikansiossa olevista vanhoista projekteista, tai luoda kokonaan uusi. Mikäli aiemmat, mutta samassa polussa sijaitsevat projektit on luotu muulla kuin juuri käytössä olevalla UE-versiolla, niiden versionumerot näkyvät esikatselussa (Kuva 11).

Kuva 11 Unreal Project Browser



Uusille projekteille on olemassa valmiita pohjia, jotka sisältävät yleisimmin käytettyjä toiminnallisuuksia, asetuksia ja laajennuksia. Tässä tapauksessa valitaan Games-kategoriasta Virtual Reality -pohja. Tämä kyseinen pohja sisältää valmiiksi hyödyllisiä ominaisuuksia ja toimintoja VR-järjestelmän käytöstä, joita ei ole tarpeen tehdä itse, kuten Kuva 12 näkyvä teleportaatio, jolla hoidetaan vaatimusmäärittelyssä mainittu liikkuminen.

Kuva 12 Projektipohjan valinta



Tämän jälkeen säädetään projektin perusasetukset, kuten onko kyseessä Blueprint- vai C++ -pohjainen projekti, onko lopputuotetta tarkoitus käyttää tietokoneella / pelikonsolilla vai mobiililaitteilla ja minne ja minkä nimisenä projekti tallennetaan. Tämän jälkeen käynnistetään projektin luomisprosessi, jossa voi vierähtää yllättävän kauan.

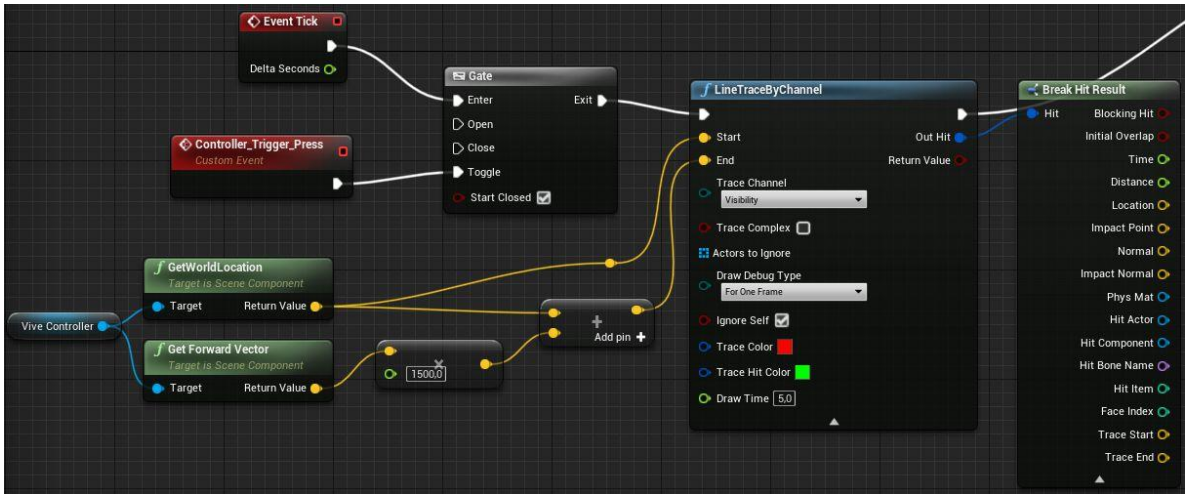
5.2 Objektien valinta

Tärkeä osa sovellusta on mahdollisuus valita objektit, joihin vaikutetaan ja korostaa nämä valinnat näkyvästi käyttäjälle. Valinnassa käytetään nodea nimeltä *Single Line Trace by Channel*, jossa luodaan määritetystä pisteestä haluttuun suuntaan lähtevä säde. Säteen osumat kohteisiin rekisteröidään, jolloin saadaan tieto osumakohteesta, ja jolloin on mahdollista luoda tähän nimenomaiseen osuttuun objektiin vaikuttava tapahtuma.

Säteelle määritetään alku- ja päätepiste sijaintivektoreina. Koska tarkoituksena on luoda liikeohjaimesta lähtevä säde, on lähtöpisteen vastattava ohjaimen sijaintia. Yhtä lailla päätepiste tulee määritellä suhteessa ohjaimen sijaintiin. Säde luodaan aina kun node aktivoidaan, jolloin säde on olemassa Draw Time -parametrilla määritetyn ajan vain näiden kahden pisteen välillä, ja tästä syystä suora liitos ohjaimen painallukseen ei tule kysymykseen. Ratkaisu on liittää säteen luominen **Event Tickiin**, jolloin node laukaistaan jokaisella ruudunpäivityksellä. Mikäli säteen olemassaoloa on tarpeen säädellä napin painalluksella, voidaan hallinta toteuttaa lisäämällä Event Tickin ja LineTraceByChannelin väliin **Gate***,* joka avataan ja suljetaan halutulla painikkeella.

Kuva 13 esitetään malli, jossa ohjaimen painikkeella luodaan säde. Lähtöpiste saadaan ohjaimesta, ja sen päätepiste määritetään 1500 yksikön (15 metrin) päähän lähtöpisteen **forward vectorin** suuntaan. Säteen luomaa nodea kutsutaan joka tickillä, mutta ohjaimen painikkeella avataan tai suljetaan portti, jolla säädellään tickistä lähtevää suorituspulssia. Säteen ja sen osumakohdan väri voidaan määrittää, jotta sen toiminta voidaan havaita ja todeta suorituksen aikana. Säteen mahdollinen osuma voidaan purkaa **Break Hit Result**-nodella osiin, jolloin siitä voidaan poimia yksilölliset, tarvittavat tiedot.

Kuva 13 Esimerkki Line Tracen luomisesta



Jotta Line Trace voi varsinaisesti osua objektiin, on sillä oltava niin sanottu ”törmäysrunko” (collision hull). Usein objektille on määritetty näitä törmäysrunkoja kaksi: yksinkertainen, eli karkeasti objektin ympäröivä, sekä monitahoinen, joka myötäilee tarkasti pinnan muotoja. Tämän projektin toimintojen kannalta on tärkeää, että nämä törmäysrungot löytyvät nimenomaan objektilta itseltään, jotta osuma rekisteröidään varsinaisessa kohteessa; törmäysrunkoja on mahdollista luoda myös omina objekteinaan virtuaaliympäristöön, jolloin niillä on mahdollista luoda esimerkiksi näkymättömiä seiniä.

5.3 Blueprint Interface

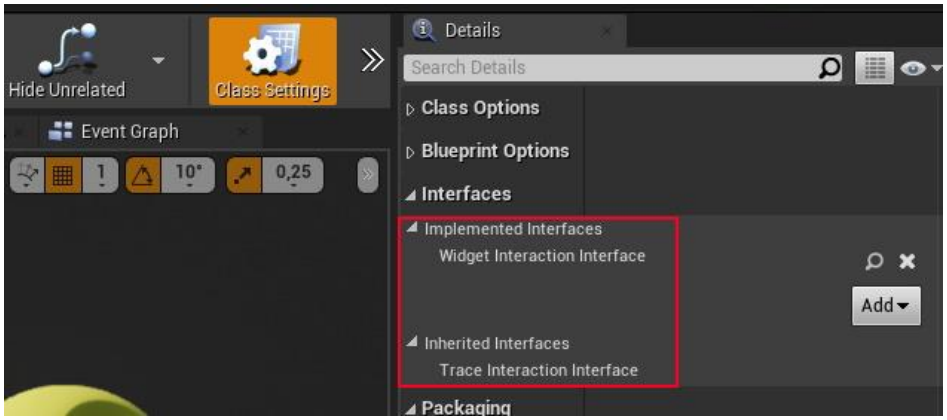
Jotta Line Tracella voidaan vaikuttaa virtuaaliympäristön kohteisiin, otetaan käyttöön Blueprint Interface. Se toimii nimensä mukaisesti rajapintana tai protokollana muiden blueprinttien välillä helpottaen näiden välistä viestintää.

Itsessään Blueprint Interface (myöhemmin rajapinta) ei sisällä toimintoja, mutta siinä määritetään yksi tai useampi funktio, joita voidaan kutsua kaikkien samaa rajapintaa käyttävien blueprinttien toimesta. Nämä ”rajapintafunktiot” ovat blueprintkohtaisia, joten niiden toiminta on yksilöllistä; funktiokutsulle annetaan kohteeksi samaa rajapintaa käyttävä toinen blueprint, jossa toiminto on määritetty.

Funktioille voidaan antaa yksi tai useampi syöte (input) ja tuloste (output), joita ei kuitenkaan ole välttämätöntä käyttää funktiokutsussa, mutta ne ovat kuitenkin määritettävä, että kyseistä funktiota voidaan käyttää. Syötteiden ja tulosteiden avulla on kuitenkin helppo viedä tietoa blueprinteistä toiseen pelkkien suorituskäskyjen ohella.

Rajapinta otetaan käyttöön blueprintin asetuksissa Class Settings > Interfaces (Kuva 14). Niitä voi olla käytössä useita samaan aikaan; myös mahdollisen vanhemman perityt rajapinnat ovat käytössä.

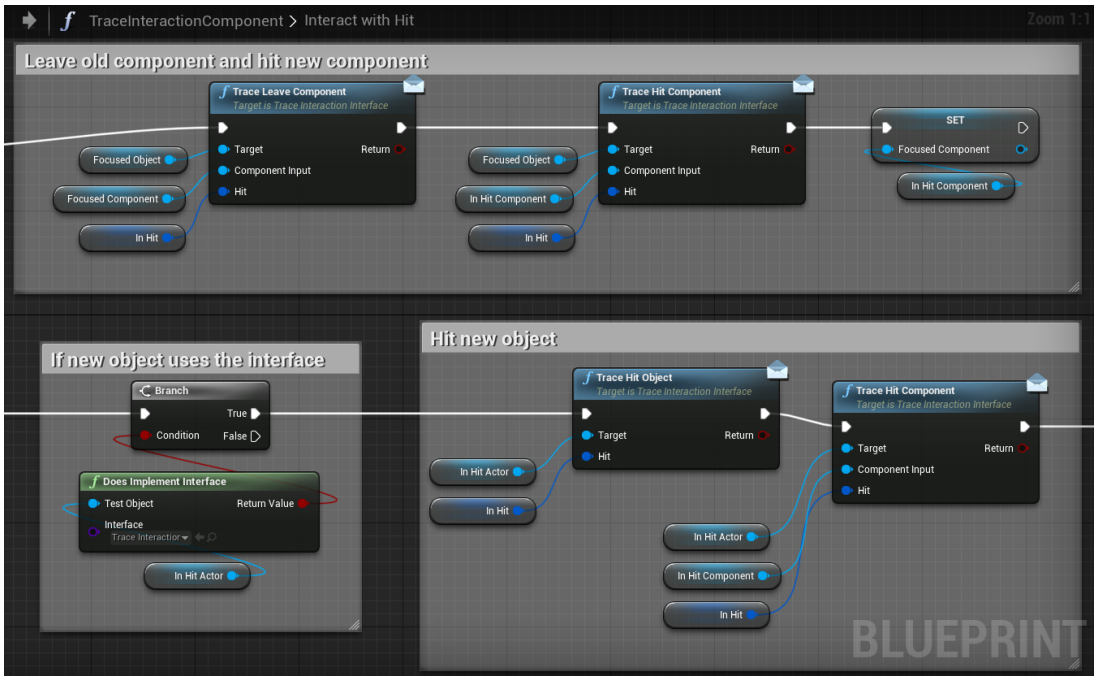
Kuva 14 Blueprint Interface-käyttöönotto



Käytössä olevat rajapintafunktiot näkyvät Blueprint Editorissa My Blueprint -välilehdellä muiden funktioiden ja muuttujien tapaan. Mikäli funktiolle ei ole jostain syystä määritetty syötettä ja tulostusta, se näkyy listalla keltaisena mutta sitä ei voida ottaa käyttöön.

Tässä projektissa blueprint-rajapintojen käyttö alkoi Line Tracen toteutuksen myötä kirjassa Unreal Engine VR Cookbook (McCaffrey, 2017) kuvatulla tavalla, jossa Line Tracen mahdollisen osumakohteen rajapintafunktioita kutsutaan eri tilanteissa (Kuva 15).

Kuva 15 Esimerkki blueprint-rajapinnan käytöstä Line Tracen kanssa

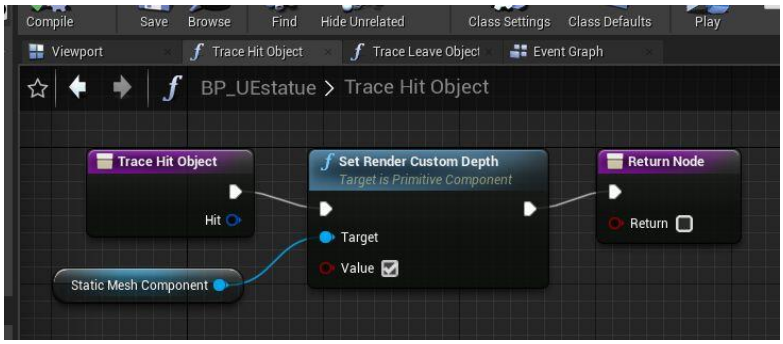


5.4 Objektin korostus

Kun ohjaimesta lähtevä valintasäde osuu kohteeseen, johon voidaan vaikuttaa, halutaan että tuo kohde on mahdollista selvästi korostaa ja tuoda ilmi, että vuorovaikutus on mahdollinen. Objektin korostukseen on useita eri tapoja, mutta pääosin ne liittyvät Post Process Volumen käyttöön. Post Process -efektit, vapaasti käännettynä jälkitehosteet, luodaan kentässä ennen renderöintiä, ja niillä voidaan vielä hienosäätää sovelluksen ulkonäköä suorituksen aikana.

Valitussa tavassa toteuttaa korostus ensimmäisenä luodaan uusi materiaali, joka määrittää miltä korostus näyttää. Tämä materiaali lisätään projektin kenttäkohtaiseen PostProcessVolume elementtiin, jonka jälkeen kohteissa, jotka halutaan korostaa, kytketään tapauskohtaisesti **Render Custom Depth Pass** päälle kutsumalla niiden rajapintafunktiota aina Line Trace-säteen osuessa kohteeseen (Kuva 15 ja Kuva 16). Vastaavasti säteen poistuessa Render Custom Depth Pass kytketään pois.

Kuva 16 Korostuksen aktivoiminen



Jälkikäsittelytehosteilla on yleisesti mahdollista muokata pelin tai sovelluksen ulkoasua laajemminkin erilaisilla efekteillä; valmiita tähän käyttötarkoitukseen luotuja materiaaleja ja efektejä löytyy muun muassa UE Marketplacesta. Tässä projektissa käytettiin mallin mukaan tehtyä yksinkertaista, yksiväristä hohtavaa materiaalia (Kuva 17), jonka voimakkuutta (eli reunan paksuutta), väriä ja hehkun tehoa voidaan helposti säätää käyttötarpeen mukaan.

Kuva 17 Korostettu kohde

5.5 Valikot VR-tilassa

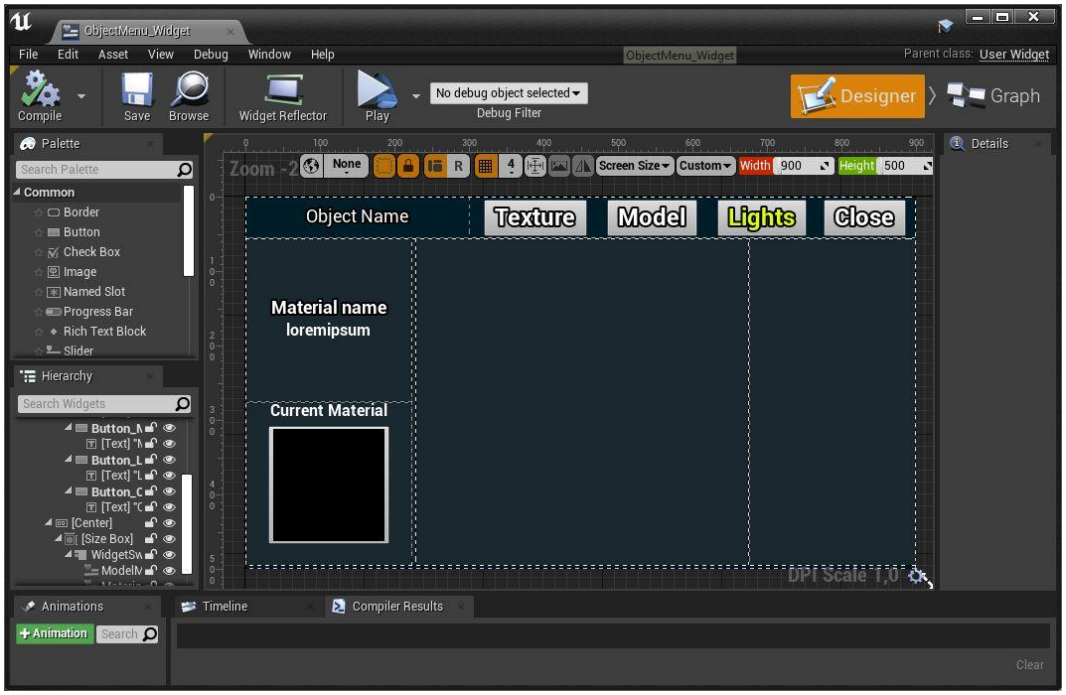
Sovelluksen keskiössä on virtuaaliympäristössä valitun kohteen tai esineen ominaisuuksiin, kuten sen pintamateriaaliin vaikuttaminen. Virtuaalitodellisuus käyttöympäristönä tuo käyttöliittymän toteutukseen omat erityispiirteensä ja haasteensa, sillä esimerkiksi villisti liikkuvat, animoidut tai muuten paikkaa vaihtavat elementit voivat aiheuttaa käyttäjillä VR-

tilassa pahoinvointia. Toisaalta uusille ja luoville toteutuksille on enemmän tilaa, sillä valikkoja voidaan rakentaa vaikkapa käyttäjän liikkeisiin perustuvaksi toiminnaksi.

Tavallisesti UE-sovellusten valikot rakennetaan Widget Blueprinteistä Unreal Motion Graphics UI Designerillä (UMG), ja ne piirretään suoraan käyttäjän näkymään (eli viewporttiin), jolloin ne eivät varsinaisesti ole olemassa kolmiulotteisessa ympäristössä. Virtuaalitodellisuus vaatii hieman toisenlaisen lähestymistavan, sillä valikoiden oltava olemassa virtuaalisessa ympäristössä jossain muodossa. Kaksiulotteinen, UMG:llä luotu valikko on edelleen mahdollista toteuttaa, mutta sitä varten on luotava ja asetettava kenttään actor-objekti. Valikon actorin myötä avautuu mahdollisuus yhdistää kaksi- ja kolmiulotteisia elementtejä.

Englanninkielinen sana **widget** tarkoittaa tässä yhteydessä käyttöliittymän elementtiä, jolla näytetään käyttäjälle tietoja tai jonka avulla käyttäjä voi olla vuorovaikutuksessa sovelluksen kanssa. Esimerkiksi jokainen painike, tekstilaatikko ja kuva ovat kaikki yksittäisiä widget-elementtejä.

Kuva 18 UMG Editor

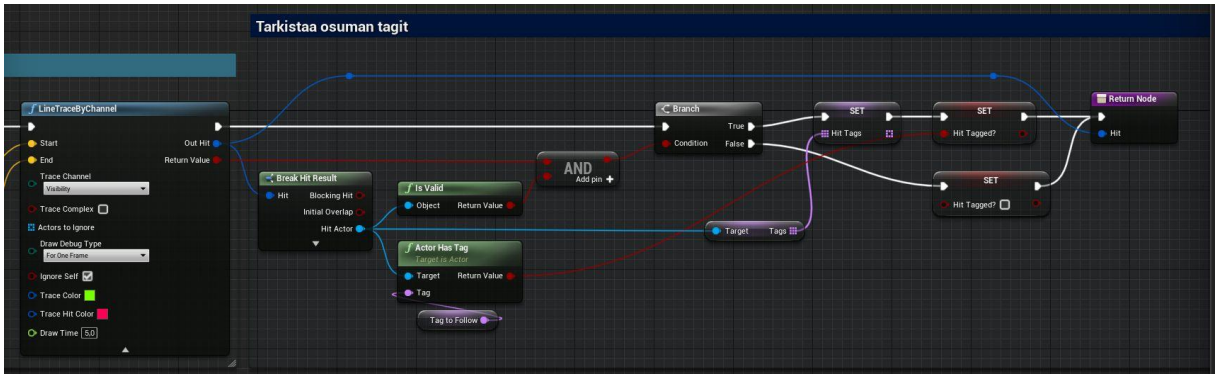


UMG Editorissa (Kuva 18) on kaksi puolta: visuaalisen ilmeen toteutukseen **Designer**, jossa määritetään widgettien sijainti ja ulkonäkö sekä ominaisuudet, ja tavanomaisempi blueprint muokkain **Graph,** jossa määritetään widgettien ja koko blueprintin toiminta.

Koska tarkoituksena on kehittää sovellusprototyyppi, oli projektin kannalta olennaisinta keskittyä valikoiden toimintaan ulkonäön kustannuksella. Täysin pelkistettyyn ratkaisuun ei kuitenkaan päädytty.

Nyt luotavassa sovelluksessa muokattavat kohteet valitaan ”merkkien” (tag) avulla. Kun Line Trace osuu objektiin, tarkistetaan, onko kohteella haluttu merkki (joka on asetettu Tag to Follow -nimiseen muuttujaan), ja jonka perusteella asetetaan Hit Tagged? -totuusarvomuuttuja (Kuva 19). Jos merkki löytyy, eli Hit Tagged? on tosi, muutetaan seuraavan ohjaimen painalluksen luoman suorituspulssin kulkusuuntaa niin, että Line Trace suljetaan ja avataan kohdekohtainen valikko.

Kuva 19 Osuman tagin tarkistus



Avattava valikko (Kuva 20) koostuu kolmesta eri Widget Blueprintistä, eli päävalikosta, joka toimii kaiken pohjana sekä materiaalin ja mallin vaihtosivuista. Päävalikon asettelussa yläpalkista löytyy tekstikenttä, jonne asetetaan juuri valitun objektin nimi (kohtaan Object Name), Texture-painike materiaalin vaihtosivulle, Model-painike mallin vaihtosivulle, Lights-painike sekä Close-painike, jolla nimensä mukaisesti suljetaan valikko.

Lights-painike tulee näkyviin vain silloin, kun valitun kohteen merkeistä (tageista) löytyy merkki, joka kertoo objektin olevan valaisin. Painikkeella voidaan sytyttää ja sammuttaa kyseisen valaisimen valo.

Kuva 20 Päävalikko UMG Designerissä

Yläpalkin alla sijaitsee **Widget Switcher** -paneeli, jonka sisälle asetetaan materiaalin ja mallin vaihtosivujen Widget Blueprintit ns. lapsiobjekteina. Yläpalkin painikkeet vaihtavat Widget Switcherin näytettävää sivua; oletuksena näkymässä on materiaalivalikko.

5.5.1 Materiaalin vaihto

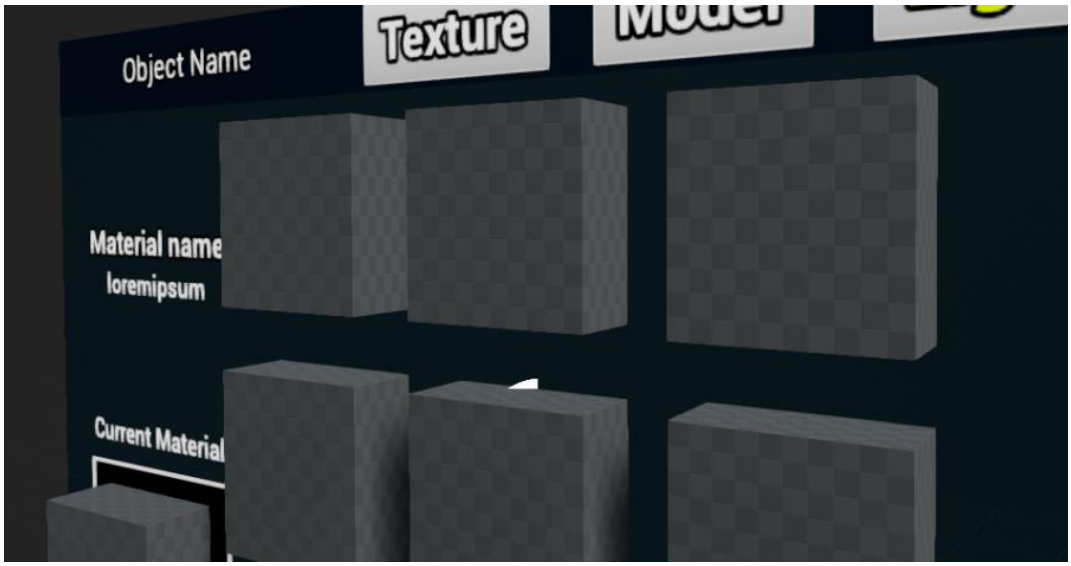
Kun objekti valitaan ja avataan valikko, aktivoidaan materiaalivalikko. Materiaalivalikko on päävalikon tapaan koottu kolmesta eri sivusta, joista jokainen on oma Widget Blueprinttinsä. Vasempaan laitaan tuodaan valitun kohteen nykyisen materiaalin nimi, ja sen alapuolelle luodaan Static Mesh -komponentti, jolle asetetaan kyseinen materiaali. Kohteilla voi olla myös useampi kuin yksi materiaali, mutta käytössä olleen ajan puitteissa päädyttiin käsittelemään vain ensimmäistä materiaalipaikkaa.

Keskellä pääsivua sijaitsee Widget Switcher, jonka sisältönä on kaksi materiaalisivua. Materiaalisivut 1 ja 2 sisältävät taustavärin ohella ainoastaan **Uniform Grid Panelin**, eli taulukkoelementin, joka jakaa tilansa tasaisesti kaikkien lapsiobjektiensa kesken. Sivuille luodaan molempiin yhdeksän painiketta, joista jokainen on niin ikään oma Widget Blueprint. Luomisprosessissa näille painikkeille, siis ”Material Slot” -nimisen Widget Blueprintin instansseille, haetaan tätä tarkoitusta varten luodusta **Data Table** -taulusta sinne asetetun materiaalin nimi, ryhmä sekä viittaus itse materiaalin. Jokainen painike vastaa siis yhtä tietotaulukon riviä.

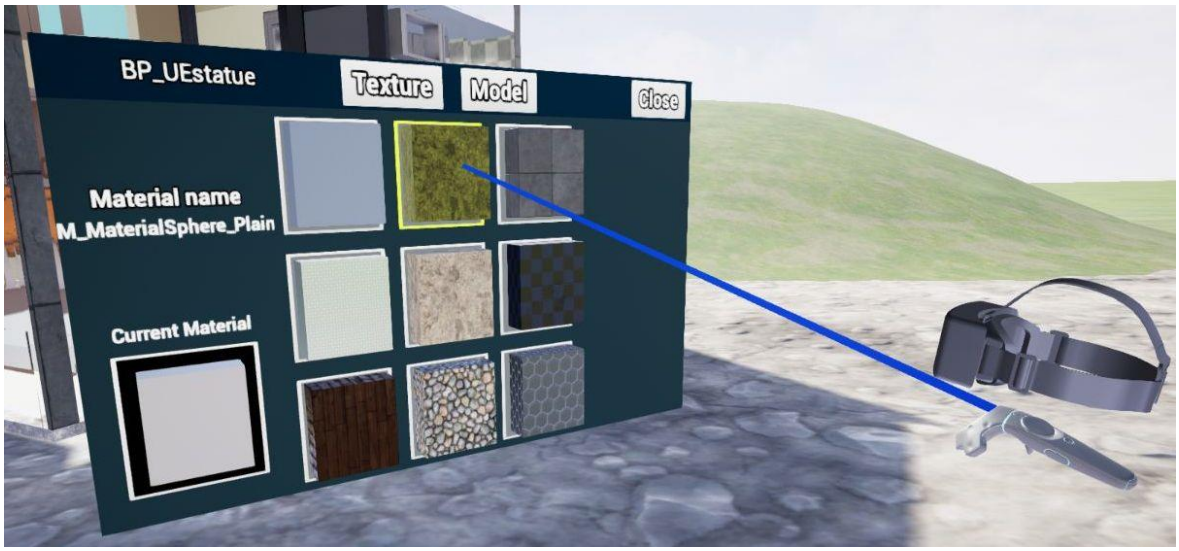
Materiaaleja sisältävään tietotauluun on asetettu yhteensä 18 riviä. Kun kyseessä on materiaalisivu 1, haetaan luotaviin painikkeisiin taulun rivit 1–9 ja kun tarkoituksena on näyttää materiaalisivu 2, painikkeisiin haetaan taulun rivit 10–18. Näitä materiaalisivuja vaihdetaan painamalla sivun vasemmassa alanurkassa olevaa, nykyisen materiaalin esittävän kuution alla olevaa painiketta. Kaikille painikkeille on asetettu korostusväri ”kursorin” vietäessä niiden päälle valinnan selkeyden lisäämiseksi.

Koska Widget Blueprintit ovat täysin kaksiulotteisia, ei niissä voida esittää samoja, esineiden pinnoille tarkoitettuja materiaaleja sellaisenaan niiden mahdollisen korkeusvaihtelun tai heijastusominaisuuksien takia vaan käytössä olevien materiaalien tulee olla luokaltaan käyttöliittymään sopivia. Näin ollen olemassa olevien materiaalivaihtoehtojen esitys sellaisenaan ei onnistu, jonka vuoksi kaksiulotteiseen valikkoon on yhdistetty kolmiulotteisia elementtejä: kenttään luotavassa widget actorissa on itse widgetin ohella ”materiaalikuutioiksi” nimettyjä static mesh -komponentteja, jotka on asetettu painikkeiden sijainteihin (Kuva 21). Kuutioiden törmäysrunko on asetettu olemaan havaitsematta osumia, jolloin valikoita käytettävällä säteellä voidaan osua niiden alla oleviin painikkeisiin. Näihin kuutioihin vaihdetaan materiaalisivujen 1 ja 2 mukaiset materiaalit (Kuva 22). Materiaalikuutiot asetetaan näkyviin tai piilotetaan sen mukaan, onko päävalikossa auki materiaalin vai mallin vaihtosivu.

Kuva 21 Materiaalikuutiot kaksiulotteisessa valikossa



Kuva 22 Materiaalin vaihtosivu käytössä

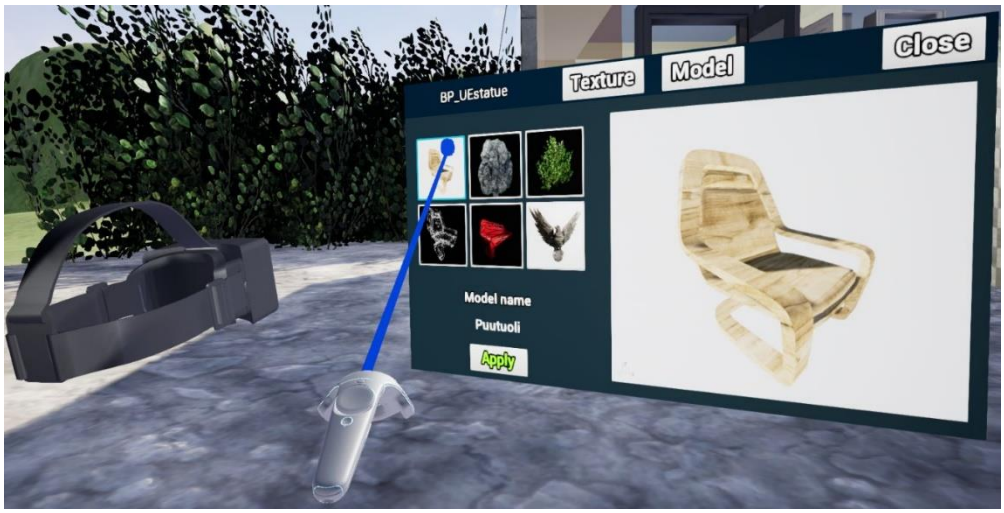


5.5.2 Mallin vaihto

Valitun kohteen mallin (static mesh) vaihdolle on oma, materiaalin vaihtoa yksinkertaisempi sivunsa (Kuva 23). Sivu on jaettu osiin, joista vasemmalla esitetään vaihtoehtoja malleista, joihin olisi mahdollista vaihtaa sekä mallin nimi. Tässä toteutus on samanlainen kuin materiaalien vaihdossa: Uniform Grid Panelin kaltaiseen, mutta hieman toisin toimivaan **Wrap Boxiin** luodaan tietotaulun pohjalta painikkeita, joihin asetetaan päälle esikatselukuva kyseisestä mallista. Painiketta painamalla samasta kuvasta tulee suurempi versio sivun oikeaan laitaan. Toisin kuin materiaalien tapauksessa, tämä esikatselupainike ei vielä vaihda itse mallia.

Valinta hyväksytään Apply-painikkeella, jolloin kohteelle vaihdetaan valittu malli. Mikäli vaihto epäonnistuu, siitä ilmoitetaan käyttäjälle Model name -tekstin päälle tulevalla, punaisella ilmoituksella.

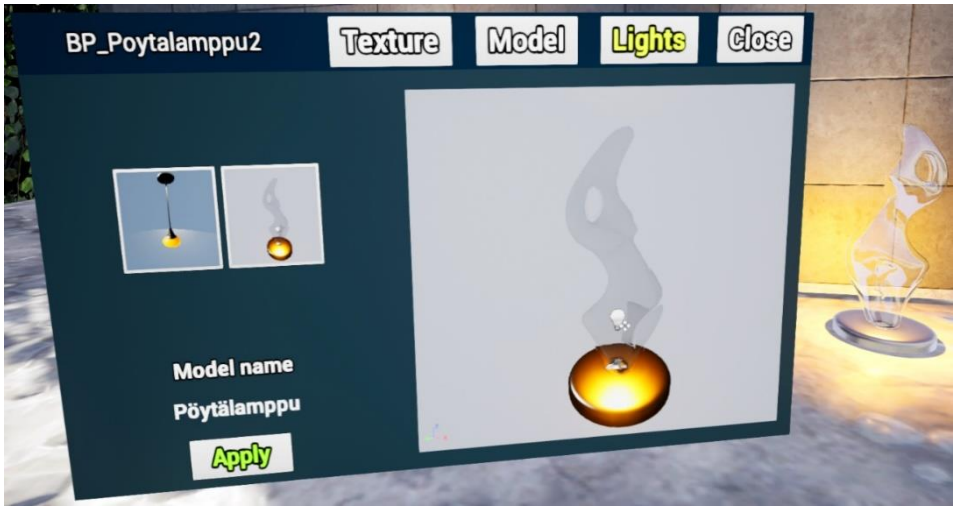
Kuva 23 Mallin vaihtosivu



Mallin vaihtosivu esittää halutulla tavalla sitä, miten vaihtotoimenpide tapahtuisi ja miltä valikko voisi näyttää, mutta itse toimintoa ei saatu toimimaan, ja tuloksena on aina edellä kuvatun kaltainen virheilmoitus. Projektiin käytössä olleen ajan vuoksi tätä ominaisuutta ei ollut mahdollista korjata vaan pääpaino asetettiin toiminnon visuaaliseen esittämiseen valikon muodossa.

Varianttina mallin vaihtoon ja päävalikon toimintaan on mallin vaihtosivu, kun valittu objekti on merkillä ”Lamppu” tunnistettu valaisimeksi (Kuva 24). Tässä tapauksessa vaihtoehtoja esitetään vain kaksi eli tässä tapauksessa kaikki ne tietotaulun rivit, joista merkintä ”Lamppu” löytyy.

Kuva 24 Valaisinvalikko

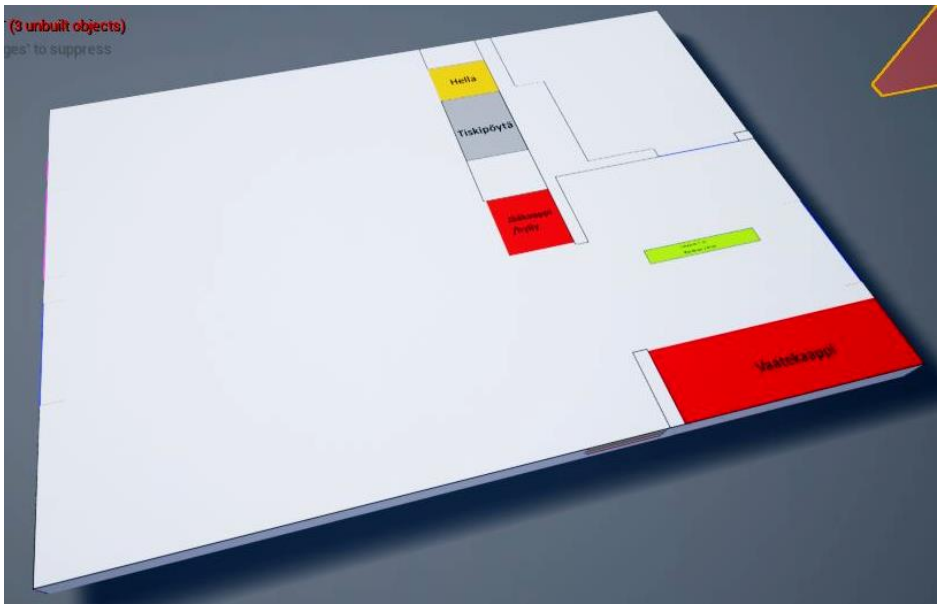


5.6 Tilan visualisointi

Kehitystyön tavoitteiden mukaan toimintoja kehitettiin ensin omassa testiympäristössään, jonka jälkeen toiminnot tuotiin varsinaiseen projektiin, jossa niitä voidaan käyttää ja esitellä parhaalla mahdollisella tavalla. Osa tätä viimeistä vaihetta oli käyttää joko itse visualisoitua tilaa tai esimerkiksi Unreal Engine Marketplacesta valmiiksi saatavilla olevaa kenttää. Vaikka valmiin tilan käyttäminen olisi ajallisesti ollut tehokkaampi vaihtoehto, päädyttiin kokonaisuuden kannalta mahdollisimman omavaraiseen ratkaisuun.

Tilan rakentaminen alkoi pohjalevyn päälle asetetusta pohjapiirustuksesta (Kuva 25). Pohjapiirrokseen oli ennestään merkitty vihreä 1m x 14cm ”skaalainkuvio”, jonka päälle asetettiin läpinäkyvä, Unreal Editorin mittasuhteiden mukaan saman kokoinen esine. Tämän jälkeen pohjan kokoa muokattiin niin, että skaalainkuvio vastasi sen päällä olleen esineen pinta-alaa.

Kuva 25 Visualisoidun tilan pohjapiirros



Pohjalevyn päälle, mahdollisimman tarkasti piirrosten mukaan asetettiin sisä- ja ulkoseinät, jotka muokattiin **Box Brush** -elementeistä, koska niiden jokaiselle pinnalle voidaan asettaa oma materiaalinsa. Tämä ja muut **Geometry Brushes**-työkalut soveltuvat yksinkertaiseen ja nopeaan kenttien ja objektien luomiseen esimerkiksi prototyyppivaiheessa, mutta niitä ei varsinaisesti suositella käytettäväksi lopullisessa kehitysversiossa. (Epic Games, n.d.-b)

Alkuperäisen idean mukaan tila olisi ollut visualisoitu kalusteita myöten mahdollisimman aidon näköiseksi, mutta pian kävi ilmi, että kiinteiden kalusteiden osalta oli syytä tyytyä vain niiden paikkaa esittäviin, läpinäkyviin laatikoihin; laatikot nimettiin niiden esittämien kohteiden mukaan. Kuva 26 näkyvät kirjahylly (DeepDreamDimension, 2020) ja maalaus (Flindigo, 2018) on haettu TurboSquid-sivustolta, mutta muut visualisoinnissa käytetyt esineet ja materiaalit ovat osa Unreal Editor Starter Packia.

Kuva 26 Valmis, visualisoitu tila

6 Johtopäätökset ja pohdinta

Kokonaisuudessaan opinnäytetyöprosessin aikana syntynyt sovellusprototyyppi vastaa monelta osin sitä pohjaideaa, joka työn aihetta valitessa nousi esiin. Alkuperäinen visio oli samaan aikaan selkeä, mutta myös varsin epämääräinen etenkin teknisen toteutuksen osalta; oli tiedossa mitä haluttiin tehdä, mutta yksityiskohdat siitä, miten se toteutetaan, muotoutuivat vasta kehitysprosessin aikana.

Kehitystyöhön käytetyn ajan jäädessä kireän aikataulun vuoksi valitettavan lyhyeksi ilmenivät puutteet vaatimusmäärittelyssä ja siihen liittyvä toimeksiantajan puuttuminen: työn mittakaavan ja tarkoituksen määritys jätti auki sen oleellisen kysymyksen, tulisiko sovelluksen ensisijaisesti toimia teknisesti oikein toteuttaessaan määritettyjä toimintoja, vai onko tärkeämpää luoda vain toimintoja visuaalisesti kuvaava sovellus, joka tuo esiin sen mitä sillä voisi tehdä ja miten se voisi toimia.

Teknisesti työssä toteutetut toiminnot toimivat halutulla tavalla, pois lukien objektin mallin vaihto. Ratkaisuissa ei otettu huomioon resurssien käytön optimointia, vaan ensisijaisena tavoitteena oli toiminnallisuus ja luotujen ominaisuuksien esittäminen.

Työn tavoitteiden mukaista, itsenäistä ja koottua sovellusta projektin aikana ei valmistunut. Valmistettu prototyyppi toimii Unreal Editorin sisällä suunnitellusti, mutta julkaisuvalmiin version hiomiseen ei lopulta jäänyt aikaa; vaatimusmäärittelyssä ei otettu kantaa tähän asiaan erikseen, mutta tekijälle muodostui projektin edetessä vähimmäisvaatimukset ominaisuuksista, joita julkaistavassa versiossa tulisi olla (mm. käynnistysvalikko, testaus) ja kun niihin ei ollut mahdollista päästä, julkaisuversion paketoinnista opinnäytetyöprosessin puitteissa luovuttiin. Julkaisuversion käyttötestaukseen ei myöskään olisi ollut aikaa.

Nyt kehitetyn sovellusprototyypin kaltaisia sovelluksia on ollut olemassa markkinoilla jo kauan. Niiden käytön yleisyys ei ole opinnäytetyötä kirjoittaessa tekijällä tiedossa, mutta olettama on, että tämän kaltaisille, matalan kynnyksen sovelluksille voisi olla tilausta. Nyt kehitetty sovellusprototyyppi, tai siinä toteutetut ratkaisut, voivat päätyä myöhemmin osaksi tekijän laajempaa, saman aihepiirin sovellusta.

7 Yhteenveto

Opinnäytetyö vastasi ennalta asetettuihin tutkimuskysymyksiin osittain: Kysymyksiin kuinka Unreal Enginellä luodaan sovellus ja mitä VR-sovelluksen käyttöliittymäsuunnittelussa tulee ottaa huomioon saatiin mielestäni kuvaavat vastaukset, mutta siitä, millainen Unreal Enginen soveltuvuus on visuaaliseen suunnitteluun, tai mitä hyötyjä ja haittoja virtuaalitodellisuus tuo visuaaliseen suunnitteluun opinnäytetyö ei valitettavasti vastaa, sillä näihin vastatakseen sovelluksesta olisi pitänyt saada käyttötestattava julkaisuversio.

Opinnäytetyöprosessi kasvatti omaa osaamistani Unreal Enginen käytöstä etenkin uusien, entuudestaan tuntemattomien toimintojen myötä, joka oli eräs työn tärkeimmistä tavoitteista. Myös käsitys siitä, miten käyttöliittymäratkaisuja virtuaalitodellisuussovelluksissa voi, kannattaa ja ei kannata rakentaa, karttui.

Projekti oli mielenkiintoinen ja monelta osin odotuksieni mukainen. Itseohjautuvuus toi omat haasteensa, mutta uskon että siltäkin osin kokemus oli kehittävä; asioiden priorisoimisessa ja kokonaisuuden hahmottamisessa on itselläni kehitettävää, mutta tämän projektin esiin tuomien haasteiden kautta niihin on helpompi kiinnittää jatkossa suurempaa huomiota. Sovellus syntyi aikatauluhaasteistaan huolimatta nykymuodossaan lyhyessä ajassa, noin kolmessa viikossa, johon voinen olla tyytyväinen.

Toteutettua sovellusta on tarkoitus jatkokehittää opinnäytetyöprosessin jälkeen niin, että ensimmäisenä se saatetaan julkaisukelpoiseksi ja opinnäytetyötä vastaavaksi korjaamatta vielä sen teknisiä puutteita, ja myöhemmin keskittyen jo olemassa olevien ominaisuuksien korjaamiseen ja parantamiseen. Tulevaisuudessa on mahdollista, että siihen lisätään vielä uusia ominaisuuksia; sovelluksen on ennen kaikkea tarkoitus toimia, kuten opinnäytetyöprosessin aikanakin, itseopiskelualustana sovelluskehitykseen Unreal Enginellä.

Lähteet

Ahola, T. (2020). *CaseStudy CAD-mallien hyödynnettävyydestä: ESS - gamEngine lab*. GamEngine Lab - HAMK Smart. https://blog.hamk.fi/ge/cad-mallien-hyodynnettavyys-ess/

Aitoro, J. (2016). 30 Years: Virtual Reality — Training Transformation. *Defense News*. https://www.defensenews.com/30th-annivesary/2016/10/25/30-years-virtual-reality-training-transformation/

Buttle, P. (2020). *The Power Behind Video Games: A Look at Game Engines | by Paul Buttle | We The Players | Medium*. Medium. https://medium.com/wetheplayers/the-power-behind-video-games-a-look-at-game-engines-2731315086e0

DeepDreamDimension. (2020). *BOOKCASE 3D model*. TurboSquid. https://www.turbosquid.com/3d-models/bookcase-3d-model-1550714

Epic Games. (n.d.-a). *Downloading Unreal Engine Source Code | Unreal Engine Documentation*. Unreal Engine 4 Documentation. Retrieved 4 February 2021, from https://docs.unrealengine.com/en-US/ProgrammingAndScripting/ProgrammingWithCPP/DownloadingSourceCode/index.html

Epic Games. (n.d.-b). *Geometry Brush Actors*. Unreal Engine 4 Documentation. Retrieved 17 March 2021, from https://docs.unrealengine.com/en-US/Basics/Actors/Brushes/index.html

Epic Games. (n.d.-c). *Plugins*. Unreal Engine 4 Documentation. Retrieved 23 February 2021, from https://docs.unrealengine.com/en-US/ProductionPipelines/Plugins/index.html

Epic Games. (n.d.-d). *Types of Blueprints*. Unreal Engine 4 Documentation. Retrieved 17 February 2021, from https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/index.html

Epic Games. (n.d.-e). *Virtual Scouting Overview*. Unreal Engine 4 Documentation. Retrieved 12 February 2021, from https://docs.unrealengine.com/en-US/BuildingWorlds/VRMode/VirtualScouting/Overview/index.html

Epic Games. (2021a). *Licensing Options - Unreal Engine*. https://www.unrealengine.com/en-US/download

Epic Games. (2021b). *Other licensing options - Unreal Engine*. https://www.unrealengine.com/en-US/custom-license

Evenden, I. (2016). *The history of virtual reality - Science Focus - BBC Focus Magazine*. Science Focus. https://www.sciencefocus.com/future-technology/the-history-of-virtual-reality/

Farris, J. (2020). *Forging new paths for filmmakers on The Mandalorian*. Unreal Engine Blog. https://www.unrealengine.com/en-US/blog/forging-new-paths-for-filmmakers-on-the-mandalorian

Flindigo. (2018). *North style nature painting October 3D model*. TurboSquid. https://www.turbosquid.com/3d-models/furniture-october-paintings-picture-3d-model-1334926

Fowle, K. (2015). *The Vanguard of Virtual Reality: An Embarrassing Arcade Game*. The Atlantic. https://www.theatlantic.com/entertainment/archive/2015/02/when-vr-was-an-arcade-game/385139/

Furness, T. (2015). *Tom Furness (Grandfather of AR and VR) - 'Being the Future' at AWE 2015*. Augmented World Expo. https://www.youtube.com/watch?v=zr89F88AuUI

GlobalData Technology. (2020). *History of virtual reality: timeline*. Verdict. https://www.verdict.co.uk/history-virtual-reality-timeline/

Gregory, J. (2018). *Game Engine Architecture, Third Edition*. CRC Press LLC. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=5150818

Haggrén, H. (2003). *Fotogrammetrian perusteet, luento 2*. Maa-57.300 Fotogrammetrian Perusteet | Aalto-Yliopisto. https://foto.aalto.fi/opetus/300/luennot/2/2.html

HAMK. (2019). *VATTU – Valmistavan teollisuuden virtuaaliset tuotteet - Hämeen ammattikorkeakoulu*. https://www.hamk.fi/projektit/vattu/

Hansen, R. (2020). *Announcing the first beta release of the ArcGIS Maps SDK for game engines*. ArcGIS Blog. https://www.esri.com/arcgis-blog/products/developers/announcements/announcing-the-first-beta-release-of-the-arcgis-maps-sdk-for-game-engines/

Helmbäck, M. (2020). *How Unreal was Used in the Design of the World's Most Advanced Research Facility for Neutron Science [video]*. Unreal Fest Online 2020. https://youtu.be/fiVC3nRZoGQ

IGN. (2009). *Epic Games Announces Unreal Development Kit, Powered By Unreal Engine 3*. https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3

Jones, S. (2011). *Investigation into modular design within computer games. BSc (Hons)*

Computer Games Design. *May*. http://wiki.polycount.com/w/images/7/70/Investigation_into_modular_design_within_computer_games_v1.0.pdf

Lawrie, E. (2020). *What went wrong with virtual reality? - BBC News*. BBC News. https://www.bbc.com/news/business-50265414

List of game engines. (2020). Wikipedia. https://en.wikipedia.org/wiki/List_of_game_engines

McCaffrey, M. (2017). *Unreal Engine VR Cookbook* (1st ed.). Addison-Wesley.

Pape, D. (1999). *File:VPL Eyephone and Dataglove.jpg - Wikimedia Commons*. Wikipedia. https://commons.wikimedia.org/wiki/File:VPL_Eyephone_and_Dataglove.jpg

Paule, L. (2020). *Standalone Virtual Reality headsets and their uses*. Laval Virtual. https://blog.laval-virtual.com/en/standalone-virtual-reality-headsets-and-their-uses/

Pimentel, K. (2019). *Visualizing Sweden's first high-speed railway with real-time technology*. Unreal Engine Spotlights. https://www.unrealengine.com/en-US/spotlights/visualizing-sweden-s-first-high-speed-railway-with-real-time-technology

Poetker, B. (2019). *The Very Real History of Virtual Reality (+ A Look Ahead)*. https://learn.g2.com/history-of-virtual-reality

Postpace. (2020). *Using Game Engines For Cinematography and Film Production*. Postpace. https://postpace.io/blog/using-game-engines-for-cinematography-and-film-production/

Ratushnyi, Y. (n.d.). *Why You Should Give Up on PC VR and Focus on Standalone VR Apps*. Visartech. Retrieved 11 February 2021, from https://www.visartech.com/blog/advantages-of-standalone-vr-over-pc-vr/

Robertson, A., & Zelenko, M. (2014). *VOICES FROM A VIRTUAL PAST*. The Verge. https://www.theverge.com/a/virtual-reality/oral_history

Schalk, J. van der. (2020). *Game engines are revolutionizing filmmaking*. FreedomLab. https://freedomlab.org/game-engines-are-revolutionizing-filmmaking/

Sherman, W. R., & Craig, A. B. (2018). *Understanding Virtual Reality : Interface, Application, and Design* (2nd ed.). Elsevier Science & Technology. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=5754454

Sutherland, I. (2018). *VR@50: Celebrating Ivan Sutherland's 1968 Head-Mounted 3D Display System*. ACMSIGGRAPH. https://www.youtube.com/watch?v=LZCx0yH9gLM

Sweeney, T. (2015). *If you love something, set it free*. https://www.unrealengine.com/en-

US/blog/ue4-is-free

Thompson, C. (2017). Stereographs were the original virtual reality. In *Smithsonian* (Vol. 2017, Issue October). https://www.smithsonianmag.com/innovation/sterographs-original-virtual-reality-180964771/

Thompson, S. (2019). *VR Applications: 21 Industries already using Virtual Reality*. VirtualSpeech. https://virtualspeech.com/blog/vr-applications

Thomsen, M. (2010). *History of the Unreal Engine*. IGN. https://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine

Tomos Morgan. (2019). *Virtual reality PTSD treatment has 'big impact' for veterans*. BBC Wales News. https://www.bbc.com/news/uk-wales-49880915

Torkkel, J.-M. (2019). *Valmistavan teollisuuden virtuaaliset tuotteet - gamEngine lab*. GamEngine Lab - HAMK Smart. https://blog.hamk.fi/ge/vattu/

Turi, J. (2014). *The sights and scents of the Sensorama Simulator*. Engaget. https://www.engadget.com/2014-02-16-morton-heiligs-sensorama-simulator.html

Unity Technologies. (2020). Top 2020 Trends: Enterprise AR & VR. *Unity Technologies*.

Vectrex Museum. (2012). *3D Imager - Vectrex Wiki*. https://vectrex.fandom.com/wiki/3D_Imager

Virtual Reality Society. (n.d.). *VPL Research Jaron Lanier | Virtual Reality Site*. Virtual Reality Society. Retrieved 9 February 2021, from https://www.vrs.org.uk/virtual-reality-profiles/vpl-research.html

VR Sync. (2020). *How the movie industry uses virtual reality*. VR Sync. https://vr-sync.com/de/how-the-movie-industry-uses-virtual-reality/

Weinbaum, S. G. (2007). *Pygmalion's Spectacles*. Project Gutenberg's Pygmalion's Spectacles. https://www.gutenberg.org/files/22893/22893-h/22893-h.htm

Wheatstone, C. (n.d.). *Charles Wheatstone-mirror stereoscope*. Wikipedia; Wikipedia. Retrieved 8 February 2021, from https://en.wikipedia.org/wiki/Stereoscope#/media/File:Charles_Wheatstone-mirror_stereoscope_XIXc.jpg

Wikipedia. (2021). *Comparison of virtual reality headsets*. Wikipedia. https://en.wikipedia.org/wiki/Comparison_of_virtual_reality_headsets

Liite 1: Aineistonhallintasuunnitelma

Kehitysprojektin aikana pidetään päiväkirjamaista työkirjaa, johon kerätään teknistä tietoa projektista sen edetessä. Tämä tieto analysoidaan opinnäytetyötä varten. Projektista kertyy myös kuvakaappauksia ja mahdollisesti videokaappauksia. Kuvakaappauksia käytetään soveltuvilta osin osana opinnäytetyöraporttia, videokaappauksia yleisesti taustamateriaalina kuvaamaan tiettyjä toiminnallisuuksia tarpeen mukaan. Työkirjaa, sekä kuva- ja videokaappauksia säilytetään projektin aikana henkilökohtaisella HAMKin OneDrive-tilillä, ja niistä otetaan varmuuskopioita opinnäytetyöprojektille varatulle ulkoiselle kovalevylle.

Projektin kehitystyön kohde, VR-suunnittelusovellus ja sen kehitysvaiheiden projektitiedostot, säilytetään opinnäytetyöprojektin ajan tekijän C-asemalla, ja niistä otetaan varmuuskopioita opinnäytetyöprojektille varatulle ulkoiselle kovalevylle.

Opinnäytetyössä kertynyttä aineistoa säilytetään tekijän tietokoneella SSD-massamuistiasemalla (ei järjestelmälevyllä), sekä varmuuskopiota opinnäytetyöprojektille varatulla ulkoisella kovalevyllä vähintään yhden (1) vuoden verran opinnäytetyön valmistumisesta.

Opinnäytetyössä tuotetun aineiston ja tulokset omistaa tekijä itse. Kolmannen osapuolen aineistoa, kuten kuvia, grafiikkaa tai 3D-malleja käytettäessä varmistetaan niiden käyttöoikeus.

Mikäli opinnäytetyössä käytettyjä lähteitä on ladattu verkosta käytettäväksi offline-tilassa, näitä lähdedokumentteja säilytetään muiden varmuuskopioiden yhteydessä vähintään yksi (1) vuosi opinnäytetyön valmistumisesta.