LAPIN AMK
Lapland University of Applied Sciences

MULTIPLATFORM-PELIKEHITYS UNREAL ENGINE -PELIMOOTTORILLA

Irina Luiro

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2024

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Irina Luiro	**Vuosi**	2024
Ohjaaja	Maisa Mielikäinen		
Toimeksiantaja	FrostBit-laboratorio		
Työn nimi	Multiplatform-pelikehitys Unreal Engine -pelimoottorilla		
Sivumäärä	36		

Opinnäytetyö toteutettiin Lapin ammattikorkeakoulun ohjelmistolaboratoriolle FrostBitille. Opinnäytetyössä pyrittiin kehittämään yhteistyökumppanille, Metsämuseolle, VR-versiona toimivaa pelillistettyä visualisointia museon tiloista ja näyttelyistä.

Metsämuseolle toteutettavan pelillistetyn visualisoinnin ohessa tarkasteltiin Multiplatform-pelikehitystä ja kuinka se toimii Unreal Engine -pelimoottorilla. Tämän tarkoituksena oli keskittyä erityisesti Multiplatform-kehityksen haasteisiin sekä tutkia alustakohtaisia vaatimuksia ja kuinka nämä vaatimukset tulee ottaa huomioon itse kehityksessä. Teoriapohjana toimivat Unreal Enginen oma dokumentaatio ja alakohtaiset julkaisut.

Opinnäytetyön pohjalta voidaan todeta Unreal Enginen sopivan aloittelevalle ohjelmoijalle hyvin, mutta itse asetusten hienosäätäminen vaatii kokemusta ja harjoittelua. Myös Unreal Enginen visuaalisten ominaisuuksien täysi hyödyntäminen vaatii tietotaitoa ja sen uudemmat tekniikat, kuten naniitti ja MetaSounds vaativat huomattavasti aikaa perehdytykseen. Joissakin asioissa Unreal Enginen kilpailija Unity tuntui olevan helpompikäyttöisempi Multiplatform-pelikehityksessä, kuten VR-ominaisuuksien testaaminen helposti editorissa ilman VR-laseja. Unreal Enginessä on kuitenkin potentiaalia myös Multiplatform-pelikehityksessä, kunhan on osaamista hyödyntää se mahdollisuus.

Avainsanat ohjelmointi, pelillistäminen, peliohjelmointi, virtuaalitodellisuus, visualisointi

LAPIN AMK
Lapland University of Applied Sciences

Abstract of the thesis

Tieto- ja viestintätekniikka
Insinööri (AMK)

Author	Irina Luiro	**Year**	2024
Supervisor	Maisa Mielikäinen		
Commissioned by	FrostBit-laboratory		
Title	Multiplatform Game Development with Unreal Engine Game Engine		
Number of pages	36		

The aim of this thesis study was to develop a VR version of the gamified visualization of the museum's facilities and exhibitions for the partner, Metsämuseo which is a museum dedicated to Lapland’s forestry. Commissioner was software laboratory FrostBit, Lapland University of Applied Sciences.

Alongside with the gamified visualization to be implemented for the Forest Museum, Multiplatform game development was examined and how it works with the Unreal Engine game engine. The purpose of this was to focus specifically on the challenges of Multiplatform development, and to examine platform-specific requirements and how these requirements should be considered when developing the visualisation. The theoretical basis was Unreal Engine's own documentation and sector-specific publications.

Based on the results of the study, it can be stated that Unreal Engine is well suited for a novice programmer, but fine-tuning the settings requires experience and practice. Also, making full use of the visual features of Unreal Engine requires know-how and its newer technologies, such as nanites and Meta-Sounds, require considerable time to get used to. In some respects, Unreal Engine's competitor Unity seemed easier to use in multiplatform game development, such as easily testing VR features in the editor without VR glasses. However, Unreal Engine also has potential in multiplatform game development, assuming one has the knowledge to utilize that opportunity.

Keywords programming, gamification, game programming, virtual reality, visualization

SISÄLLYS

ALKUSANAT

Haluaisin kiittää ystäviäni, jotka ovat tämän neljä vuotta jaksaneet kuunnella minun koulustressiäni sekä ovat olleet tukenani ylä- että alamäkieni kanssa. Erityiskiitos Inkerille, joka jaksoi oikolukea opinnäytetyötäni.

Suurin kiitos kuitenkin menee isälleni, jota ilman en olisi tähän pystynyt. Hän on pysynyt elämäni varrella minulle vakaana kalliona ja sellaisena hän pysyi myös koulutieni ajan. Lämpimät kiitokset myös äidilleni, joka jaksoi minuun aina uskoa ja joka oli niin ylpeä minun päästessäni opiskelemaan insinööriksi. Sydämeni särkyy, ettei hän näe valmistumistani, mutta olen varma, että äitini olisi ollut enemmän kuin ylpeä!

KÄYTETYT LYHENTEET

Asset	peleissä käytetty hyödyke/objekti
Blueprint	Unreal Enginen käyttämä oma koodausmalli, jossa käytetään visuaalista koodausta sen kirjoittamisen sijaan
Cross-platform	monialusta, sovellus toimii useammalla alustalla ja laitteistojärjestelmällä
Culling	peleissä käytettävä piirtotekniikka, jolla määritetään mallin piirtoetäisyys pelimaailmassa pelaajan kameraan nähden
FPS	Frames Per Second, kuvataajuus, kuinka monta kuvaa näytetään sekunnissa.
Foliage	Unreal Enginen Foliage Mode -työkalulla kenttään asetettu kasvillisuus objekti
Fotogrammetria	tekniikka, jossa valokuvien pohjalta luodaan 3D mallinnus
GUID	Globally Unique Identifier, uniikki tunnus, ID
LOD	Level of Detail, asetus, joka vaikuttaa yksityiskohtien renderöintiin pelissä
Mesh	peleihin tuodut 3D mallit
Multiplatform	monialusta, sovellus toimii useammalla alustalla
PBR	Physically Based Render, pelimateriaalin teksturointi tekniikka
Pelimoottori	peleihin erikoistunut ohjelmistokehys
SDK	Software Development Kit, kokoelma ohjelmistokehityksen työkaluja.
Template	Unreal Enginessä käytettävä moottorin oma koodimalli/sapluuna
Viewport	näytön näkymä

1 JOHDANTO

Metsämuseolla oli tarve luoda digitaalinen esittely tiloista ja kokoelmistaan, ja näin luoda mahdollisesti uutta kiinnostusta Lapin metsähistoriasta uudelle käyttäjäkunnalle. Visualisoinneilla ja simulaatioilla mahdollistetaan useamman henkilön pääsy tietoihin ja paikkoihin, joihin he eivät pääse tai joissa käyminen voisi aiheuttaa vaaratilanteita. Esimerkiksi yhä useammat työnantajat turvaavat virtuaalisiin simulaatioihin perehdyttäessään työntekijöitä riskialttiisiin toimitiloihin, kuten tehtaisiin. Visualisointia hyödynnetään myös matkailussa ja ne mahdollistavat pääsyn hankalasti tavoitettaviin paikkoihin.

Koko opinnäytetyön kehitysprosessin ajan tarkasteltiin mitä kaikkea haasteita monialustainen-pelikehitys pitää sisällään ja mitä kaikkea tulee ottaa huomioon kehityksessä. Pelillistetty visualisointi toteutettiin mahdollisimman usealle alustalle vertailua varten, tässä tapauksessa Virtual Reality -laitteistolle ja tietokoneelle. Näissä vertailtiin ohjautuvuutta, graafisia asetuksia ja ohjelmointia.

Opinnäytetyötä varten on haastateltu projektityöntekijä Joakim Nivalaa, joka työskenteli yhdessä muiden kanssa tehden Metsämuseolle simuloinnin selaimelle museon tiloista. Projekti ja opinnäytetyö olivat erillisiä projektejaan. Haastattelulla haluttiin selvittää fotogrammetrian mahdollisuuksia mallintamisessa.

Tuloksena opinnäytetyöstä saatiin aikaan visualisoinnista prototyyppi. Prototyyppiä on mahdollista hyödyntää jatkokehityksessä. Opinnäytetyön asiasisältöä voidaan hyödyntää tulevissa projekteissa, erityisesti VR-projekteissa.

2 MULTIPLATFORM-KEHITYS

2.1 Multiplatform-kehityksen hyödyt ja haitat

Sanalla ”multiplatform” tarkoitetaan sitä, että samaa koodikantaa käytetään usealle alustalle kehittäessä. Usein tästä käytetään myös nimitystä ”cross-platform”. (Oztel, Yolcu Oztel & Sahin 2023.)

Monialustaisessa pelikehityksessä pelikehittäjät työskentelevät saman yhteisen koodin ja materiaalin parissa. Pelimoottorit, sovelluskirjastot, koodikielet ja SDK luovat alustakohtaiset elementit pelille julkaisua varten. Usealle alustalle kehityksestä hyödytään, sillä eri alustoille ei vaadita erillisten ja erikoistuneiden koodikantojen ylläpitoa, vaan kaikki tapahtuu yhden koodikannan sisällä. Monialustakehityksen tuote on sen julkaisussa useamman käyttäjän saavutettavissa sekä tekee tuotteen päivityksistä helpompaa, kun päivityksiä ei tarvitse tehdä useammalle koodikannalle. Näin pelikehittäjät voivat myös päivityksen jälkeen ottaa tehdyt muutokset käyttöön saman tien jokaisella alustalla säästäen aikaa ja resursseja. (Perforce 2020.)

Ongelmatonta useammalle alustalle kehittäminen ei kuitenkaan ole. Verrattuna alustakohtaiseen kehitykseen monialustaisessa pelissä voi esiintyä enemmän performanssiongelmia, sillä lisäkerros koodin ja natiivin alustan välillä voi hidastaa suoritusaikoja varsinkin grafiikkarikkaissa peleissä. Vaikka yleisesti ottaen monialusta-pelit ja -sovellukset voivat olla helpommin päivitettävissä yhden koodikannan ansiosta, voi niihin kuitenkin saada hitaammin alustakohtaisia päivityksiä ominaisuuksiin tai rajapintoihin. Myös uusimpien ominaisuuksien ja tietoturvakorjausten käyttöönotto voi viivästyä. Virhekorjaukset voivat hankaloitua, jos ongelma on alustakohtainen, ja näiden tunnistaminen sekä korjaus voivat olla haastavia. (Rizwan 2024.)

Monialusta-pelikehityksen yksi ongelmista on myös sovelluskauppojen ja pelijulkaisijoiden hyväksynnän viivästyminen. Julkaisijat tarkistavat pelin koontiversioiden käytettävyyden ja useampi alusta voi pidentää tarkistusaikaa. (Steamworks 2024).

2.2 Kehitystyökalut

2.2.1 Agisoft

Metsämuseolle tuotettavaan projektiin oli tarkoituksena luoda käytettävät mallit peliobjekteja varten fotogrammetrian avulla käyttämällä Agisoft Metashape-sovellusta. Tämä sovellus mahdollistaa mallien kehittämisen kuvista; kuvat syötetään sovellukseen, joka laskee kuvista vaadittavat parametrit ja luo 3D-mallin näiden pohjalta. Ohjelmistoa voidaan hyödyntää kartoituksissa, maanmittauksessa, veden alaisten kohteiden mallinnuksessa, arkkitehtuurissa ja lisäksi myös arkeologiassa. (Agisoft 2024.)

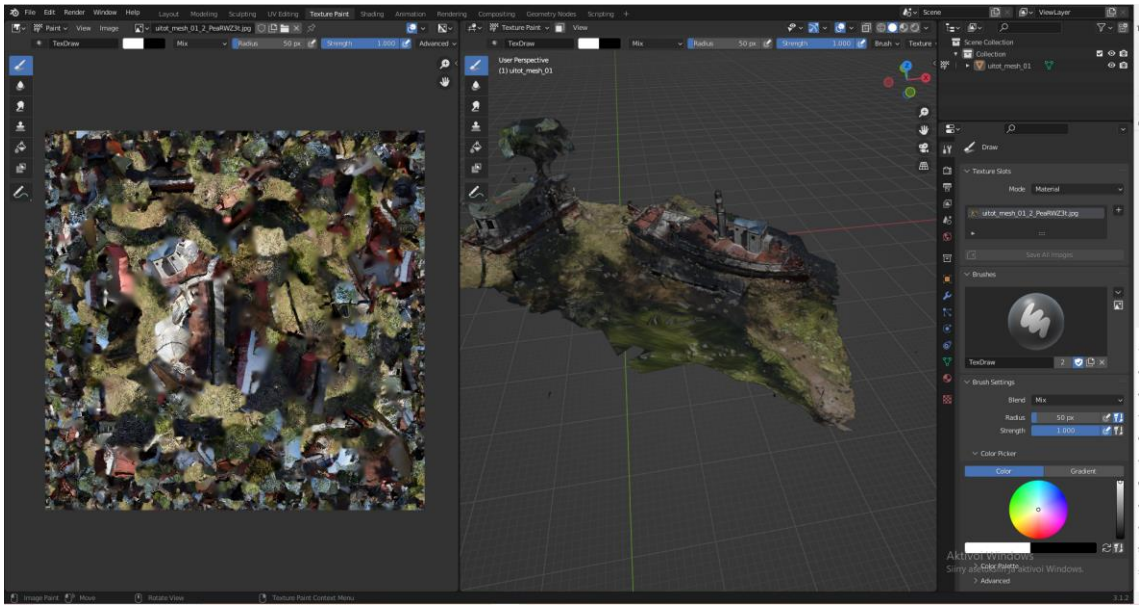
Metashape 2.1 -versiolla on mahdollista luoda myös ilmakuvista tarkkoja kartoituksia, sillä se mahdollistaa automaattisen LiDAR-pistepilven käytön paikallisen geometrian pohjalta käyttämättä väritietoja, jotka voivat olla virheellisiä. Jos väritiedot ovat olemassa, on näitä pistepilviä mahdollista säätää 3D- ja 2D-yhteyspisteiden avulla. (Agisoft 2024.) LiDAR tulee sanoista ”light detection and ranging”, ja se on mittaustekniikka, joka käyttää lasersädettä mittaamaan tarkat etäisyydet ympäristössä (IBM 2024).

2.2.2 Blender

Fotogrammetrian ansiosta 3D-malleja ei tarvinnut mallintaa alusta asti itse, mutta kehitysongelmien vuoksi näitä korjailtiin Blenderissä. Blender on lisensoitu GNU General Public -lisenssillä, jonka ansiosta Blender on avoimen lähdekoodin sovellus ja ilmainen käyttää. Sovelluksella on mahdollista mallintaa, muokata ja teksturoida sekä lisäksi sillä voidaan luoda visuaalisia tehosteita ja animoida. (Blender 2024.)

Tyypillinen ongelma fotogrammetrialla tehdyissä malleissa oli reikäisyys, mikä on suhteellisen helppo ratkaista Blenderillä. Reikäisyys ilmenee useimmiten vasta kun mallin epätasaisuuksia lähdetään tasoittelemaan ja muokkaamaan. Tätä varten valitaan malli jota halutaan muokata Blenderin hierarkiasta, ja sitten valitsemalla Modeling-näkymässä Mesh-valikko ja täältä Clean Up -asetus, josta sitten valitaan Merge by Distance. Näin mallia voidaan muokata ilman repeilyä.

Muokkaamisessa kuitenkin ilmeni isompi ongelma, joka oli tekstuurit. Agisoftilla tuotetun mallin tekstuuri ja UV-tekstuuri ovat hyvin sekavia, sillä rajat ovat sulautuneet osittain toisiinsa ja mallin yksityiskohtien tekstuurit voivat olla levittäytyneet tekstuuripohjassa laajalle alueelle (kuvio 1). UV-tekstuuri tai UV-mapping on tasainen pinta, joka kuvastaa 3D-mallin levitettyjä tekstuureja, joiden avulla tekstuurit levittäytyvät itse mallin pintaan oikein (Calvello 2022).



Kuvio 1. Kuva Metashapen automaattisesti luodusta tekstuurista

Malleja on mahdollista muokata Blenderillä, mutta tämä vaatisi kokeneen mallintajan korjailemaan fotogrammetrian virheitä. Agisoftilla mallinnetut 3D-mallit on kuitenkin mahdollista kenties pelastaa näkemällä vaivaa ja aikaa.

2.3 Kehitysympäristöt

Pelimoottoreiden tarkoituksena on tehdä pelikehityksestä helpompaa kirjastoidensa ja kehitystyökalujensa avulla. Niiden avulla pelin renderöintiä on helpompi optimoida, peliin voidaan luoda tarkat fysiikat ja pelejä pystytään kehittämään useammalle alustalle. Tämän lisäksi kehittäjän ei tarvitse joka kerta opetella uutta tapaa kehitykseen, kunhan pelimoottori pysyy samana. Pelimoottorit mahdollistavat myös helpon ulkoisten assettien käytön projektissa, kuten 3D-mallit ja kuvat. Pelimoottoreihin on mahdollista asentaa myös ulkoisia työkaluja helpottamaan kehitystyötä, esimerkiksi virheiden etsintää ja korjausta varten voidaan asennuttaa ulkoisia työkaluja käytettäväksi. Kaikki tämä voidaan toteuttaa

ilman, että pelinkehittäjä joutuu luomaan kaiken alusta asti itse, ja täten mahdollistaa keskittymisen itse pelin kehittämiseen. (Bradfield 2018; Ezpeleta, Raposas, Tiglao & Limjoco 2018.)

Ohjelmoinnissa kirjastoilla tarkoitetaan valmista kokoelmaa, joihin ohjelmoinnissa päästään helpoksi käsiksi ja jotka ovat uudelleen käytettävissä. Nämä pitävät sisällään valmiita ratkaisuja ja metodeja ohjelmointiin. Kirjastojen avulla voidaan luoda joustavasti omaa koodia. (Jalli 2024.)

Yksi tunnetuimmista pelimoottoreista on Unreal Engine. Sen perustaja on Tim Sweeney, joka perusti Potomac Computer Systems-pelitalon vuonna 1991 (Jensen 2023). Tämä tapahtui hänen opiskellessaan Marylandin A. James Clarkin teknillisessä korkeakoulussa. Nimi muutettiin vuonna 1995 muotoon Epic MegaGames ja lopulta vuonna 1999 nykyiseen muotoonsa: Epic Games. (University of Maryland 2019.)

Vuonna 1995 Sweeney yhdisti voimansa James Schmalzin löytämän Digital Extremes -studion kanssa (Digital Extremes 2024). Heidän yhteinen tavoitteensa oli luoda 3D-renderöintimoottori, sillä siihen asti markkinoita hallitsi 2D ”spritet” ja 3D polygonit. Moottoria oli innoittamassa Sweeneyn vuonna 1991 kehittämä ja julkaisema seikkailupeli nimeltä ZZT. Unreal Engine julkaistiin 22. toukokuuta vuonna 1998. Unreal Engine lisensoitiin jo ennen julkaisuaan muille pelistudioille, ja tuolloin kehitysvaiheessa ollut Duke Nukem Forever vaihtoi pelimoottorinsa Quakesta Unreal Engineen. (Jensen 2023.)

Unreal Engine sopii hyvin vaihtoehdoksi pelikehityksessä muun muassa sen blueprinttien ansiosta, mitkä on esitelty neljännestä versiosta eteenpäin. Näiden avulla on mahdollista ohjelmoida visuaalisesti sopien myös kehittäjille, joilla ei ole kokemusta koodauksesta. Se on yksi etu Unreal Enginellä kilpailijoihinsa nähden graafisten mahdollisuuksien lisäksi. (Hillmann 2019.)

2.4 Alustat

Pelillistetty visualisointi pyrittiin toteuttamaan useammalle alustalle, jotta saataisiin hyvin vertailukohteita. Näissä vertailuissa pyrittiin tarkastelemaan ohjautuvuuksien eroavaisuuksia, graafisia asetuksia sekä myös koodillisia eroavaisuuksia. Peli toteutettiin monialusta-kehityksenä useammalle alustalle, eli

siinä on vain yksi koodikanta, joka pystytään julkaisemaan ja pyörittämään useammalla eri laitteella.

Perusversio visualisoinnista kehitettiin PC:lle. Nykytietokoneilla on mahdollista pyörittää graafisesti hyvinkin näyttäviä toteutuksia, jotka voivat olla muille alustoille haastava pyörittää laitteistojen rajallisen muistin vuoksi. Tämän lisäksi PC-versiota on mahdollista pelata myös VR:llä. Tätä varten vaaditaan käyttäjältä vain VR-lasien kytkemistä tietokoneeseen jonka jälkeen päämenusta voidaan avata VR-taso.

3 OPINNÄYTETYÖN TOTEUTUKSEN SUUNNITELU

3.1 Lähtökohdat opinnäytetyön projektille

Opinnäytetyön tekohetkellä Metsämuseolla ei ollut olemassa olevaa digitaalista esittelyä museosta ja sellainen haluttiin toteuttaa. Lähtökohtana projektille sekä opinnäytetyölle oli se mitä halutaan pelillistetyltä visualisoinnilta sekä Metsämuseon miljöö. Metsämuseolta otettiin 360-kuvia referenssiksi toisen tiimin toimesta, näistä esimerkkinä kuva Metsämuseon sisäänkäynniltä (kuvio 2).

Kuvio 2. Metsämuseon Virkatalo (Kuva: Mikko Vaarala. FrostBit.)

Tärkeä ominaisuus mitä pelillistetyltä visualisoinnilta haluttiin oli interaktio museokohteiden kanssa. Visualisoinnilla haluttiin informoida Suomen metsähistoriasta, jätkäkulttuurista sekä sen ajan haasteista ja esitellä museokohteita itseään (kuvio 3). Kuvion ylävasemmalla Luiron kämpän tupa ja tästä oikealla Uittomakasiini, alavasemmalla Ahmakuusiston kämpän keittiö ja sen oikealla Uitto 6 ja ponttoonivarppaaja.

Kuvio 3. Metsämuseon näyttelyitä

Museoesineet oli tarkoitus mallintaa käyttäen fotogrammetriaa, sillä museossa on hyvin yksityiskohtaisia ja erilaisia esineitä, joita on hidasta mallintaa perinteisin keinoin. Tästä ideasta luovuttiin pelillistetyn visualisoinnin osalta, sillä fotogrammetrialla luodut mallit eivät olleet sellaisenaan valmiita käytettäväksi projektiin. Näitä malleja olivat toteuttamassa toinen tiimi FrostBitillä, jota konsultoitiin opinnäytetyötä varten.

Toisen tiimin jäsen Joakim Nivala totesi Agisoftilla työskentelyn olleen haastavaa. Haasteet alkoivat jo kuvausvaiheessa, jolloin heidän oli tarkoitus kuvata alueen maasto ilmasta käsin dronejen avulla ja mallintaa koko alue. Tätä vaikeuttivat puusto sekä maan vaikeat muodot. Agisoft ei ollut tiimille tuttu sovellus, mikä vaikeutti prosessia, sillä Agisoft olisi vaatinut kattavammat tietotaidot eri vaiheista sekä niihin vaadituista parametreistä. (Nivala 2024.)

Nivala mainitsi haastavaa olleen myös sen, että Agisoftin lisenssit olivat konekohtaisia eikä näitä saanut siirrettyä toisiin koneisiin. Nämä koneet, joilta lisenssi löytyi, uloskirjaavat käyttäjät päivän vaihtuessa. Johtuen kuvien määrästä sekä

mallinnuksen prosessista ei vuorokaudessa ollut mahdollista saada prosessia loppuun korkeilla tai edes keskisuurilla asetuksilla. Mallinnuksiin ei myöskään ollut mahdollista käyttää etäkoneita, sillä nämäkin kirjaavat käyttäjän ulos, jos koneella ei ole aktiivisesti. (Nivala 2024.) Näin ollen mallinnuksen prosessi jäi usein kesken työpäivän päätteeksi, ja seuraavana päivänä prosessi jouduttiin aloittamaan alusta.

Nivalan konsultoinnin päätteeksi pohdittiinkin, mitä kaikkea opinnäytetyössä voidaan tehdä, jos ja kun vaadittavia malleja ei saataisikaan valmiiksi. Opinnäytetyönä toteutettava pelillistetty visualisointi suunniteltiin vietävän niin pitkälle kuin mahdollista, mutta pääpainona oli tutkia monialusta-pelikehitystä. Fotogrammetrian avulla luotujen mallien korjailua suunniteltiin testattavan resurssien ja ajan puitteissa.

Alun perin projektiin suunniteltiin kuuluvaksi myös WebGL-versio, mutta Unreal Engine ei tue versiosta 4.24 ylöspäin HTML5-alustaa. Tähän kuitenkin löytyi vielä kesään 2024 asti Unreal Enginen oma 4.27-version dokumentti, joka ohjasi lukijan ulkoisen lisäosan Github-repositorioon, jonka avulla oli mahdollista kehittää vielä WebGL-versiokin. Tämä on sittemmin poistettu oletettavasti vanhentuneen version ja tiedon vuoksi, sillä Githubista löytynyttä lisäosaa oli viimeksi päivitetty ennen vuotta 2020. Tästä kuitenkin löytyy yhä todisteena keskustelut Unreal Enginen foorumeilla (Unreal Engine 2022) sekä Stackoverflown keskustelualustalla (Stackoverflow 2023).

WebGL on web-pohjainen graafinen kirjasto, joka käyttää ohjelmointikielenä JavaScriptiä ja hyödyntää HTML5 canvas -elementtiä. Sen etuna on yhteensopivuus useiden verkkoselainten kautta, ja se on siten helposti saavutettavissa. (Lee & Jang 2019.) Tästä saavutettavuudesta olisi ollut hyötyä Metsämuseon tarpeisiin.

3.2 Valitut alustat ja aikataulutus

VR-laitteille kehittämisen valintaan vaikutti halu toteuttaa mukaansa tempaava pelikokemus käyttäjälle. VR-laseilla on mahdollisuutta käyttää aisteja eri tavoin mitä normaalisti tietokoneella pelatessa, sillä ne on suunniteltu estämään ulkoiset valonlähteet ja yleensä sisältävät lisäksi kuulokkeet. VR-laitteiden mukana oleviin

ohjaimiin kuuluu myös haptiikkaa, minkä avulla fyysiseen interaktioon saadaan syvyyttä. Vaikka ohjaimissa oleva haptiikka onkin vain tärinää aiheuttavia tietyissä kohden peliä tai eri interaktioissa, antaa se silti käyttäjälle paremman immersion. (Mack & Ruud 2019.)

Eräs syy Unreal Enginen valinnassa kehitysalustaksi oli se, kuinka kehuttu se on graafisten mahdollisuuksiensa puolesta. Mutta lisäsyynä oli sen helppokäyttöisyys ohjelmoinnissa blueprinttiensä ansiosta, sillä visuaalinen koodaus helpottaa keskittymään koodiin paremmin.

Projekti lähdettiin toteuttamaan selvittämällä Metsämuseon tarpeet ja kuinka voitaisiin toteuttaa yksinkertainen ratkaisu näihin tarpeisiin. Tarpeiden pohjalta tarkasteltiin kokonaisuuden suuruutta ja suunniteltiin opinnäytetyölle sekä projektille aikataulu (ks. taulukko 1). Luodun aikataulun puitteissa olisi mahdollista toteuttaa pelillistetty visualisointi, mikä vastaisi joihinkin Metsämuseon tarpeisiin ja myös antaisi tietopohjaa opinnäytetyölle. Toteutuksen ajan pidettiin yksinkertaista päiväkirjaa ja kirjattiin huomioita ylös.

Taulukko 1. Opinnäytetyön prosessi vuodelta 2024

Vaihe	Toukokuu	Kesäkuu	Heinäkuu	Elokuu	Syyskuu	Lokakuu	Marraskuu
Suunnittelu							
Toteutus							
Raportointi							
Päättäminen							

4 VISUALISOINNIN TOTEUTUS

4.1 Unreal Enginen haasteet

Metsämuseolle toteutettava visualisointi aloitettiin konsultoimalla Metsämuseon yhteishenkilöä sekä kuvaamalla museon tiloja. Näistä ympäristökuvista toteutettiin mallinnus fotogrammetrian avulla tärkeistä rakennuksista ja sisätiloista sekä esineistä.

Unreal Enginellä toteutus aloitettiin luomalla maasto sekä luomalla tärkeimmät koodilliset ominaisuudet. VR-toteutusta koodatessa ilmenee heti iso ero Unrealin ja Unityn välillä, sillä siinä missä Unityllä löytyy helposti XR Device Simulator -lisäosa suoraan pelimoottorista editorissa tapahtuvaa testausta varten ilman itse VR-settiä (Unity 2024), tällaista ei Unreal Enginellä ole itsessään valmiina.

Unreal Engineen on mahdollista saada manuaalisesti asennettuna Metan XR Simulator -lisäosa, mutta tämä vaatii erillisen paketin latausta ja sen kopioimista Unreal Enginen kansioon sekä sen lisäksi vaatii projektiasetusten säätämistä. Tälle lisäosalle on myös kunnollinen tuki vain versiosta viisi ylöspäin. (Meta Quest 2024.)

Myöhemmin projektin ensimmäistä testausta varten tehdyssä koontiversiossa huomattiin myös, että osa Unreal Enginen sisäänrakennetuista lisäosista voi aiheuttaa ongelmia. Koontiversiota tehdessä peli ei suostu paketoitumaan oikein, vaan antaa virheilmoituksen. Tämä johtuu siitä, että lisäosat pitää olla Marketplace-kansion alla, joten lisäosat tulee joko siirtää kansiorakenteessa Marketplace-kansion alle tai sitten ottaa käyttöön vain Marketplacesta löytyvät lisäosat. Marketplacen lisäosista löytyi myös Metan XR-lisäosa, jonka käyttöönoton jälkeen huomattiin, että se sisälsi myös XR Device Simulator -lisäosan. Tästä ei löytynyt virallisista ohjeista mitään. Device Simulator vaatii kuitenkin ohjeiden mukaisen polun asetuksen siitä huolimatta, että se tuli itse lisäosan mukana.

VR-ohjelmointi oli tarkoitus tehdä alusta asti katsoen vain Unreal Enginen VR-pelimallista esimerkkiä, mutta tämä osoittautui astetta hankalammaksi, sillä pelimalli oli toteutettu usealla eri osalla. Lopulta päädyttiin tulokseen, että projektissa käytetään Unreal Enginen omaa VR-pelimallia siltään, jota sitten tarpeen tullen

muokataan. Koodillisesti tähän lisättiin objektin tarkastelu mukaillen PC:lle toteutettua tarkastelua ja lisättiin infotekstin näyttäminen käyttämällä pelimallin menua pohjana, sillä VR:llä eivät toimi viewport-menut.

Unreal Enginen 5-versioon on kehitetty uusi edistyneempi syötesysteemi, joka on taaksepäin yhteensopiva Unreal Enginen 4-version oletus syötteeseen. Edistyneempi versio tarjoaa mahdollisuuden laajentaa omaa suodatusta ja raakadatan käsittelyä asset-pohjaisessa ympäristössä. Se myös tarjoaa ominaisuuksia, kuten radiaaliset ”dead zonet”, ”chorded actionit”, kontekstipohjaiset inputit ja priorisoinnin. Näiden ominaisuuksiensa avulla pelin aikana on mahdollista lisätä ja poistaa inputtien kartoituskonteksteja riippuen mitä halutaan. Unreal Engine 5-version moottorin oma ”VR-template” tulee tämän uuden pelisyötejärjestelmän kanssa. (Unreal Engine 2024i.)

Itse maasto luotiin mahdollisimman kevyeksi visuaalisesti, jolloin sen toimivuus useammassa alustassa tehostuu ilman suurempia muutoksia. Tähän voitiin vaikuttaa mallien laadulla, mutta erityisesti tekstuurien koolla, varjoasetuksilla sekä culling-asetuksilla. Culling-asetukset vaikuttavat kuinka pitkälle malli piirtyy kameraan nähden, ja tällä voidaan vaikuttaa tehokkuuteen sekä graafiseen ilmeeseen.

Maasto luotiin Unreal Enginen omalla maastotyökalulla, jota sitten muokattiin suurin piirtein alueeseen sopivaksi. Tähän toisena vaihtoehtona olisi kokeilla maaston muokkaamista raakadatan avulla. Muun muassa Maanmittauslaitokselta on mahdollista ladata ilmaiseksi korkeusmallit tietyltä alueelta, ja tämän avulla saisi suhteellisen tarkan mallin toteutettua.

Demoa varten käytettiin ilmaisassetteja Epicin Marketplacesta kasvillisuutta varten. Näitä malleja pyrittiin muokkaaman siten, että niiden LOD-asetukset olisivat olleet pienemmät kauempana ja lähempänä olisi sijainnut korkeamman yksityiskohtien omaavat mallit. Kuitenkin testattaessa huomattiin, että nämä mallit vaikuttivat suorituskykyyn huomattavasti ja FPS tippui. Poistettaessa kauimmat puumallit saatiin FPS korotettua 50:stä aina 120:een FPS:ään.

Peliasetuksien luominen oli myös yksi tapa vaikuttaa pelin toimivuuteen useammalla laitteella, sillä käyttäjä voi itse valita tekstuurien ja resoluution välillä mahdollistaen paremman kokemuksen käyttäjän omalla laitteellaan. Toteutukseen

tehtiin yksinkertaiset asetukset niin näytön resoluution ja koon muuttamiseksi sekä grafiikoiden vaihtelemisen. Toteutukseen käytettiin GameUserSettings-luokkaa asetusten käyttöönotossa, sekä asetusten tallentamisessa. Tallennukseen on mahdollista käyttää myös Unreal Enginen SaveGame-luokkaa, jolloin saataisiin myös tallennettua audion asetukset, sillä GameUserSettings ei säädä eikä tallenna audion tietoja. Asetukset jäivät demovaiheeseen, sillä nämä olivat enemmänkin esimerkkinä, kuinka saadaan useammalle käyttöjärjestelmälle sopiva peli mukautuvuuden avulla.

Pelikenttää luodessa ensimmäisenä ideana oli luoda yksi kenttä tietokoneella ja VR-laseilla pelattavaksi, ja kentän moodia olisi pystynyt vaihtamaan esimerkiksi menun kautta erillisellä asetuksella. Mutta pelikenttää luodessa tuli ilmi ongelma liittyen tähän ratkaisuun. VR-ohjaimet tarvitsevat NavMeshBoundsVolume-peliobjektin, jotta teleporttaamiseen tarvittavat syötöt toimivat. PC-pelaajaa varten pelialue on rajattu BlockingVolume-peliobjektin avulla, mikä estää pelaajaa menemästä tietyn alueen ulkopuolelle ja pienentäen pelialuetta, vähentäen näin ympäristön mallintamiseen vaadittavia resursseja ja täten tarvittavaa renderöintiä. Jos NavMeshBoundsVolume-peliobjekti ylittää tämän BlockingVolume-peliobjektin, voi VR pelaaja teleportata rajatun alueen ulkopuolelle, mutta tämä ei voi enää teleportata takaisin.

Tämä ratkaistiin pelillistetyn visualisoinnin demoa varten luomalla kaksi erillistä kenttää tekemällä jo luodusta kentästä duplikaatti VR-kentäksi, josta poistettiin BlockingVolume-peliobjektit täysin. Mahdollinen toinen ratkaisu on luoda pelikenttä erittäin huolella ja varmistaa ettei NavMeshBoundsVolume- ja BlockingVolume-peliobjektit ikinä risteä keskenään. Tapa voi olla aikaa vievä, sillä molemmat voluumit ovat alkujaan neliön muotoisia, jota voidaan säätää leveys-, korkeus- ja pituussuunnissa. Tuolloin myös vaaditaan GameModen BeginPlay-funktiossa koodia, joka katsoo, pelataanko kenttää PC:llä vai VR:llä riippuen sille menusta välitetystä tiedosta ja sen mukaan Default Pawnin luomista, kun pelikenttä ladataan.

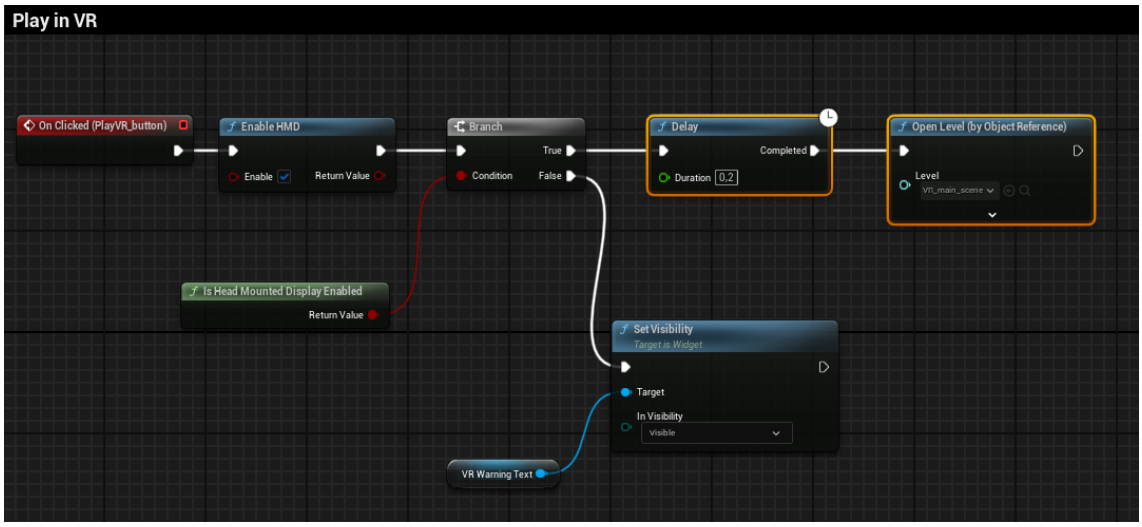
Eräs ongelma havaittiin erillisen VR-kentän kanssa, kun demo pakattiin testausta varten koontiversioon, ja ongelmana oli, ettei VR-laseja automaattisesti havaita. VR-projektin koontiversion projektiasetusten Start in VR-asetus on yksi tapa saada kenttä latautumaan VR-moodissa. Tämä ei kuitenkaan ole hyvä ratkaisu,

jos halutaan vaihdella tietokoneen ja VR-laitteen välillä pelin aikana, sillä koko projekti on silloin VR-valmiudessa eikä tietokoneella ole mahdollista renderöidä näkymää oikein. Tähän vastauksena on käyttää Enable HMD -nodea, sillä se mahdollistaa suorituksen aikaisen vaihdoksen viewportin ja VR-lasien välillä.

Jostain syystä luotu VR-kenttä ei kuitenkaan toiminut oikein, kun käytettiin Enable HMD -nodea, vaan koontiversiossa tämän kentän lataus aiheuttaa pelin kaatumisen virheilmoituksella: ”Assertion failed: !EnumHasAnyFlags(TextureDesc.Flags, ETextureCreateFlags::RenderTargetable | ETextureCreateFlags::ResolveTargetable)”. Ongelma ei ilmene kuitenkaan ilman kyseistä koodia, vaan VR-pelaajan ”blueprint” ei vain ikinä havaitse VR-laseja. Virheilmoitus viittaisi tekstuurionglemiin ja eräs mahdollisuus oli, että VR-kentässä olisi mahdollisesti GUID-ongelmia. Nämä ongelmat kuitenkin tulevat esille vain, kun VR-lasit otetaan käyttöön suorituksen aikana. Ongelmaa ei tullut vastaan, kun VR-kenttä luotiin tyhjästä kentästä ja sinne tuotiin vain testaukseen vaadittavat objektit sekä ominaisuudet, ja kyseiseen kenttään otettiin Enable HMD -noden avulla VR-lasit käyttöön.

Viallisella kentällä tehtiin testejä, jotta paljastuisi miksi tämä virheilmoitus tapahtui. Lopulta paljastui sen johtuneen vesielementistä kentässä. Vesielementti oli luotu peliobjektiin asetetun materiaalin avulla. Mahdollisesti VR-lasit eivät pystyneet renderöimään vesimateriaalia osittaisen läpinäkyvyyden vuoksi. Tarkkaa syytä ei löytynyt, mutta kokeiltuja keinoja oli muokata vesimateriaalia sekä poistaa forward shading -asetus käytöstä. Nämä eivät ratkaisseet ongelmaa. Kenttään ei kokeiltu tehdä vesielementtiä Unrealin oman lisäosan avulla, mutta tämä voisi olla eräs mahdollinen vaihtoehto vesielementin luomiselle.

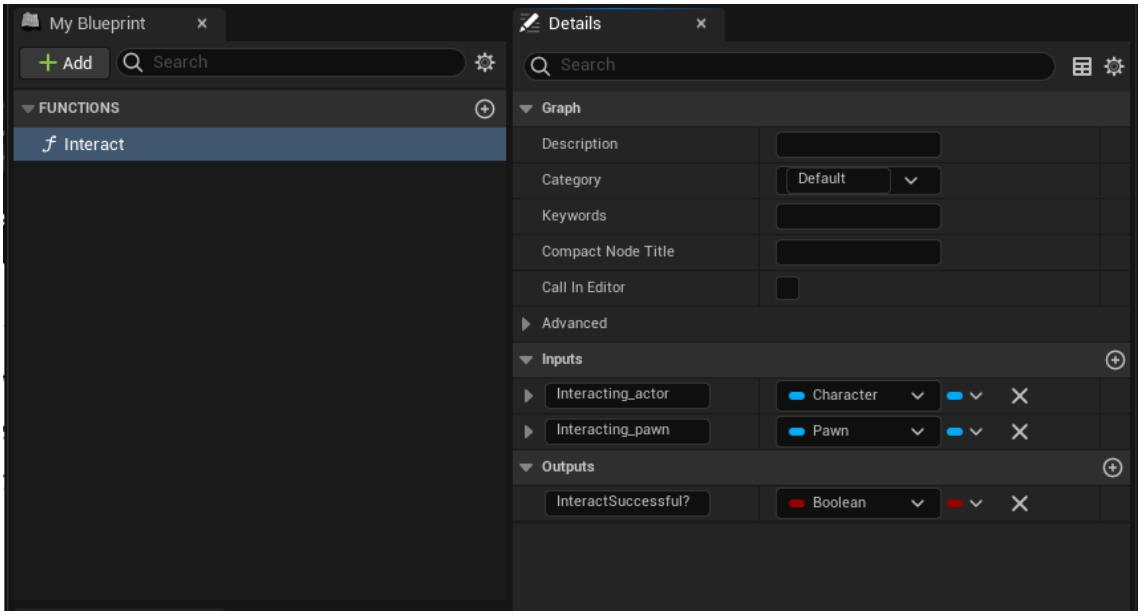
Enable HMD suoritetaan heti päämenussa ja samalla suoritetaan tarkistus siitä, löytyykö käyttäjältä XR-laitteistoa vai ei, ja ilmoitetaan käyttäjälle, jos tätä ei ole havaittu. Tällä tavalla mahdollistetaan, ettei käyttäjä turhaan lataa kenttää ja vähennetään turhaa koodia kentän puolella (kuvio 4).



Kuvio 4. VR-lasien käyttöönotto heti päämenussa

4.2 Esineiden tarkastelun toiminnallisuus

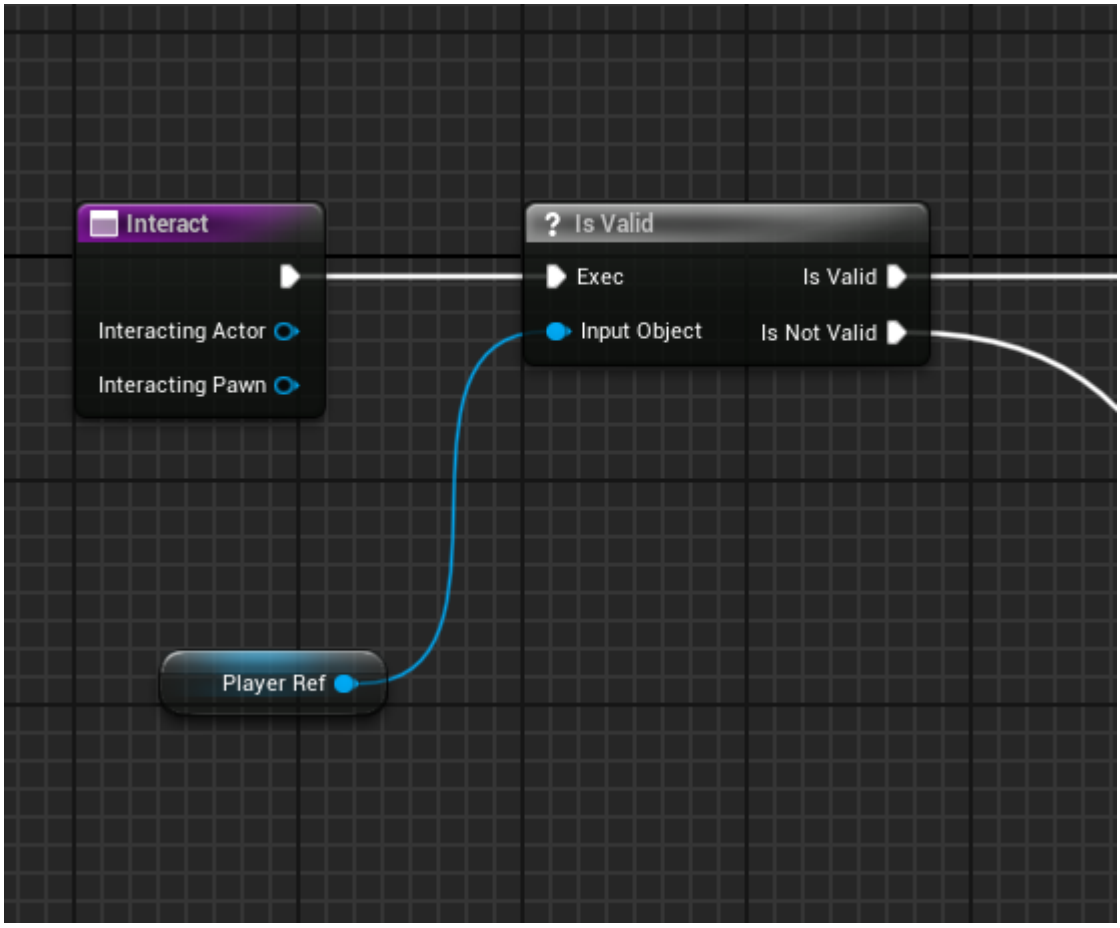
Eräs ominaisuus joka pelillistetyltä visualisoinnilta haluttiin, oli esineiden tarkastelu ja samalla olisi mahdollista lukea näistä tietoa. Tätä varten luotiin ”interface” (kuvio 5), joka välittää tiedon funktion avulla, kun esineen kanssa ollaan vuorovaikutuksessa. Unreal Enginessä tähän on oma ”blueprint”.



Kuvio 5. Interface interaktiolle

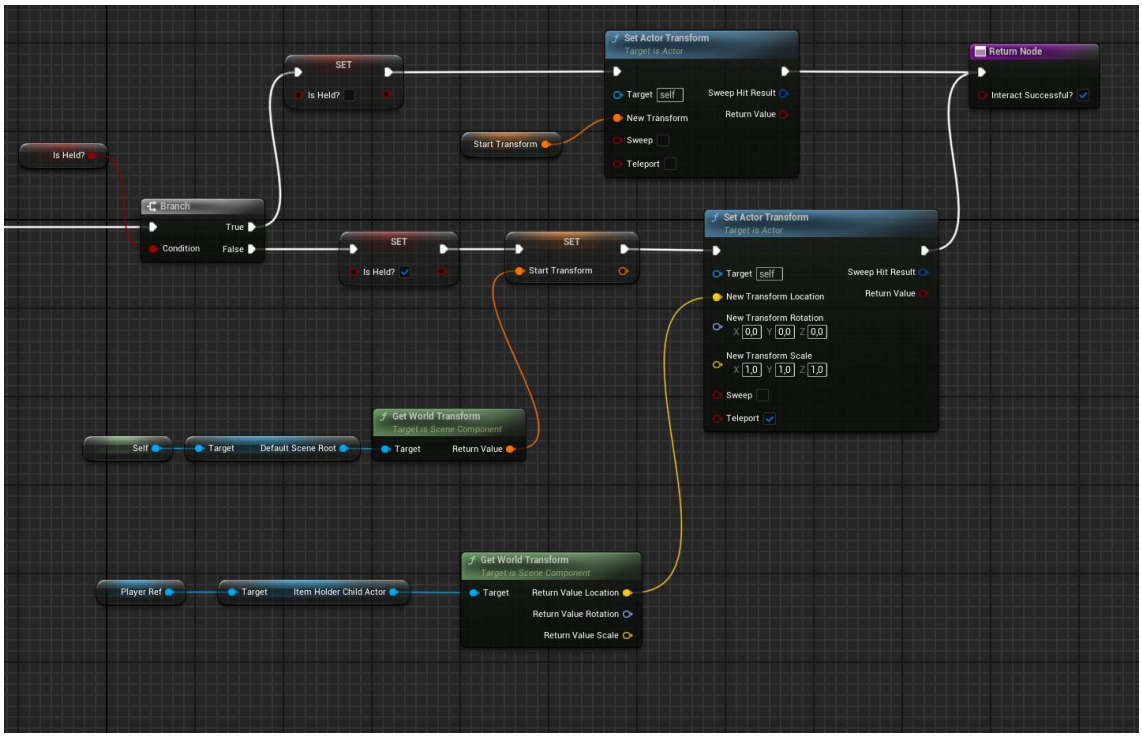
Jotta ominaisuutta voitaisiin hyödyntää luotiin erillinen ”actor blueprint”, joka toimisi pohjana eri esineille, ja tämä nimettiin ItemParent_bb nimellä selkeyden

vuoksi. Tämän ”blueprintin” Class Settings -kohdasta lisättiin aiemmin luotu ”interface” Implemented Interfaces -asetuksessa. Tämä kertoo ”blueprintille”, että se on käsiteltävä objekti, kun sille lähetetään ”interfacen” kautta tieto. Interact-funktio ylikirjoitetaan ”blueprintissä” (kuvio 6).



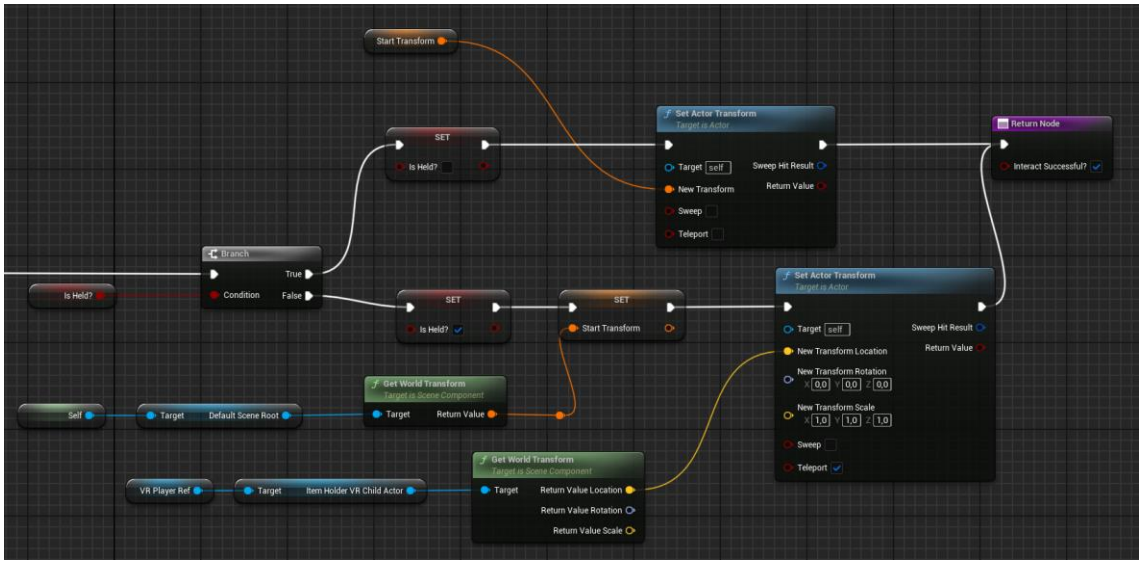
Kuvio 6. ”Interact-interfacen” ylikirjoituksen aloitus

Interface-funktiolle kerrotaan tiedot, mitä käsittelyssä tapahtuu itse esineelle. Pelaajalle haetaan referenssi BeginPlay-funktiossa, jossa ensin tarkastetaan, onko PC pawn olemassa ja asetetaan referenssi PlayerRef. Jos pelaaja on tietokoneella, toteutetaan sille funktio, mikä asettaa esineen koordinaatit PC-pelaajassa määritettyyn kohtaan nimeltä Item Holder Child Actor (kuvio 7).



Kuvio 7. Esineen sijainti asetetaan PC-pelihahmon koordinaatteihin

Jos PlayerRefin haku epäonnistuu, niin asetetaan referenssi VRPlayerRef, jonka mukaan sitten muutetaan interact-funktion toiminnallisuutta (kuvio 8). Tällä varmistetaan toimivuus myös VR-pelaajalle.

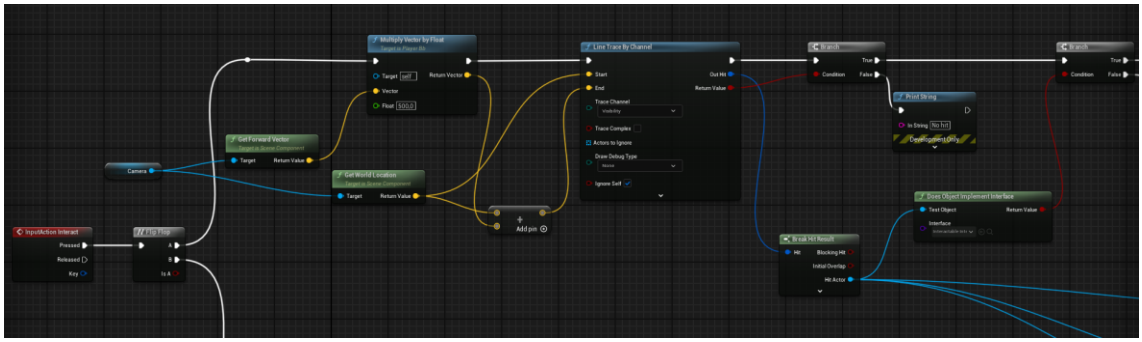


Kuvio 8. VR-pelaajan interaktio-interface ylikirjoitus

”ItemParent_bb-blueprintissä” on erinäisiä muuttujia, joita käytetään interact-funktiossa, mutta myös tyypin ja infotekstin määrittämiseen. Näitä on sitten mah-

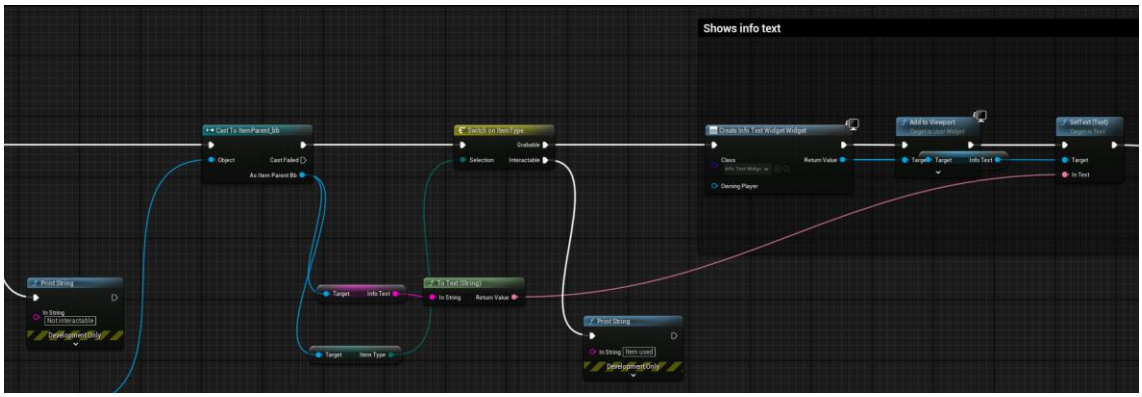
dollisuus muuttaa esine kohtaisesti, kuten myös esineen 3D-mallia, kun ”blueprintistä” luodaan ”Child Blueprint”. Tällä tavalla samaa koodia ei tarvitse toistaa useaan kertaan ja on mahdollista luoda helposti erilaisia esineitä kenttää varten, mitä siten voidaan asettaa kenttäkohtaisesti.

Ensimmäisenä luotiin PC-pelaajalle tarkoitettu funktio, jota sitten muokattiin VR:lle sopivaksi myöhemmin. PC:llä interaction tapahtuma on sidottu näppäimeen E ja esinettä voidaan pyörittää hiirtä liikuttamalla (kuvio 9). Kun pelaaja on antanut interaktio käskyn syötteellään, tarkastellaan ensin, onko pelaajan kameran edessä esinettä, jotta varmistetaan esineen olevan pelaajan ulottuvissa.



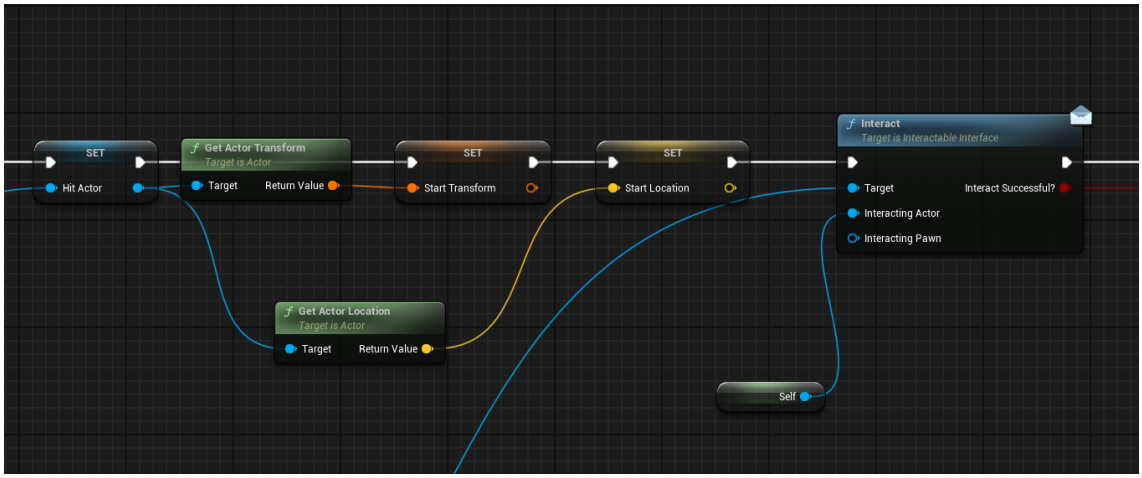
Kuvio 9. Interaktion aloitus pelaajan syötteellä

Jos esine on sellainen, jonka kanssa voidaan olla vuorovaikutuksessa, haetaan ”ItemParent_bb-blueprintistä” esineen tyyppitiedot. Tällä tavalla voidaan luoda useampia ominaisuuksia kuin vain tarkastelu. Sen jälkeen haetaan infoteksti, jos esine on tartuttava, ja asetetaan viewportiin käyttämällä ”widget-blueprinttia” nimeltä InfoText_widget. ”Widgetin” tarkoitus on luoda visualisointi tekstille (kuvio 10).



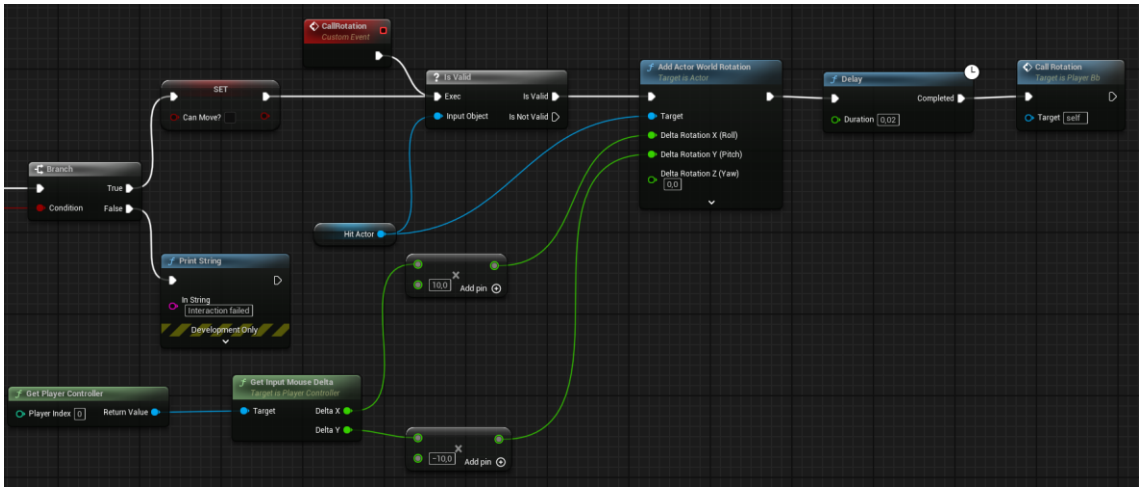
Kuvio 10. Interaktion infoaminen pelaajalle

Tämän jälkeen pelaajan muuttujiin tallennetaan tieto esineestä sekä sen aloitustiedoista ja lähetetään tieto ”interfacen” kautta esineen ”blueprintille”. Esine asetetaan pelaajan eteen käyttäen pelaajan ”blueprintissä” sijaitsevaa Child Actor -komponenttia kun esineen ”blueprint” saa ”interfacen” kautta tiedon (kuvio 11).



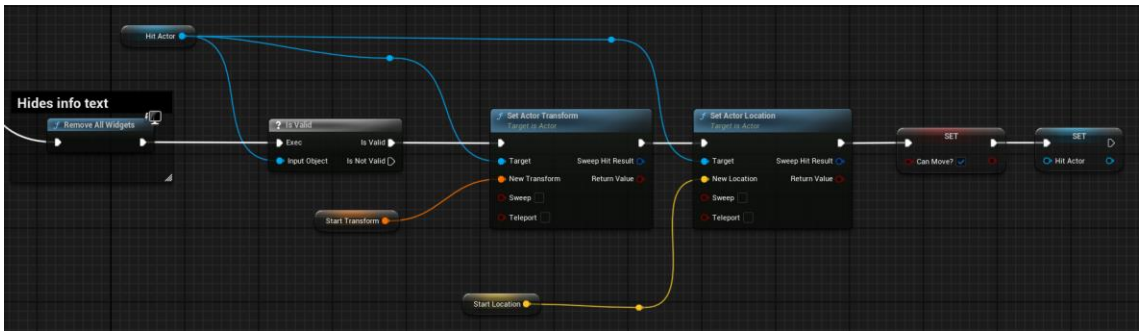
Kuvio 11. Esineen pyöritys hiirellä

Esineelle sitten luodaan rotaatio hiiren inputin avulla ja kutsutaan uudelleen tietyn ajan jälkeen, jotta pyöritys onnistuu jatkuvana. Tämä toteutetaan erillisenä Custom Event -inputtina (kuvio 12).



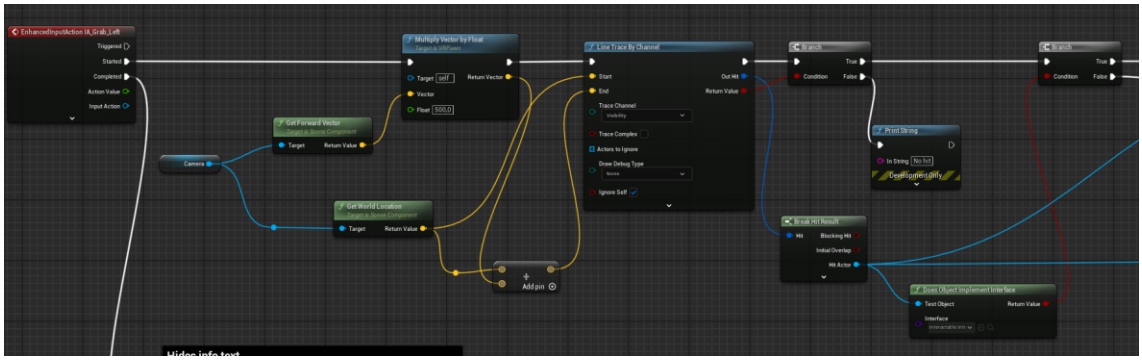
Kuvio 12. Custom Event -syötteen toiminnallisuus ja esineen pyöritys hiirellä

Painamalla uudelleen E-näppäintä piilotetaan infotekstit ja asetetaan esineelle takaisin alkuasetusarvot (kuvio 13). Tällä varmistetaan, että se jää sille asetettuun paikkaan vuorovaikutuksen jälkeen.



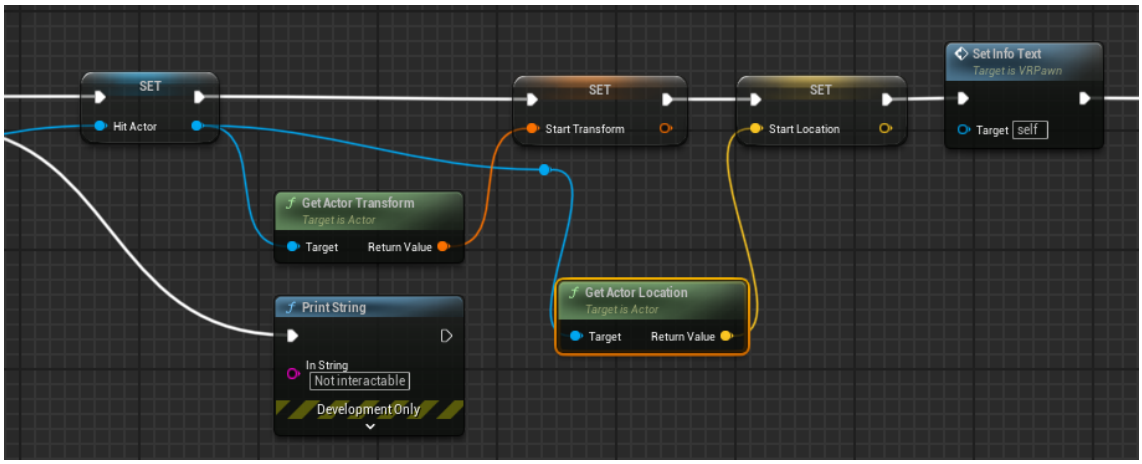
Kuvio 13. Interaktion lopetus

VR:lle tämä tapahtuma muuttui huomattavasti joissakin kohden. Flip flop -nodea ei myöskään vaadittu tässä tapauksessa, sillä Grap-syöte vastaa siitä, milloin tapahtuma alkaa ja milloin loppuu (kuvio 14). Mutta samalla tavoin VR:llä aloitetaan kameran suhteesta esineeseen.



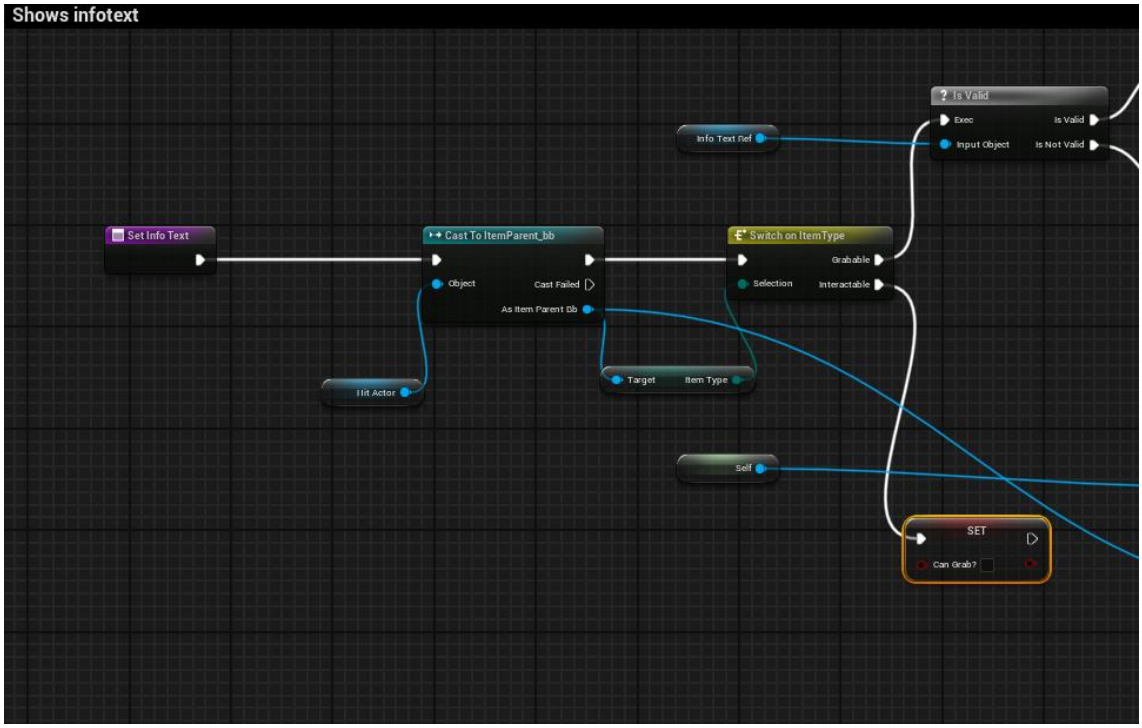
Kuvio 14. VR-pelaajan interaktion aloitus

Tämän jälkeen katsotaan mihin esineeseen kamera on osunut ja tallennetaan nämä erillisiin muuttujiinsa (kuvio 15). Näiden muuttujien avulla esine on helppo sitten interaktion jälkeen resetoida oikealle kohdalle takaisin.



Kuvio 15. Aloitusarvojen tallennus muuttujiin

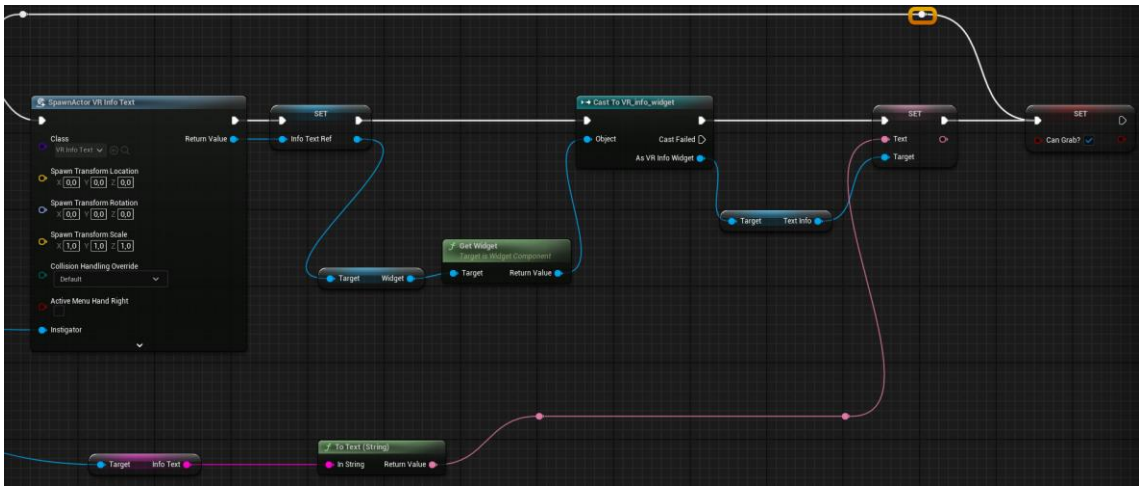
SetInfoText on oma funktionsa, joka on tehty romahduttamalla osa koodia. Funktiossa ensin haetaan HitActor-muuttujaa objekti referenssinä käyttäen ensin ”ItemParent_bb” ja vaihdetaan CanGrab-booleania riippuen halutusta vuorovaikutuksesta samoin mitä PC-pelaajallakin (kuvio 16).



Kuvio 16. VR-pelaajan infotekstien funktion aloitus

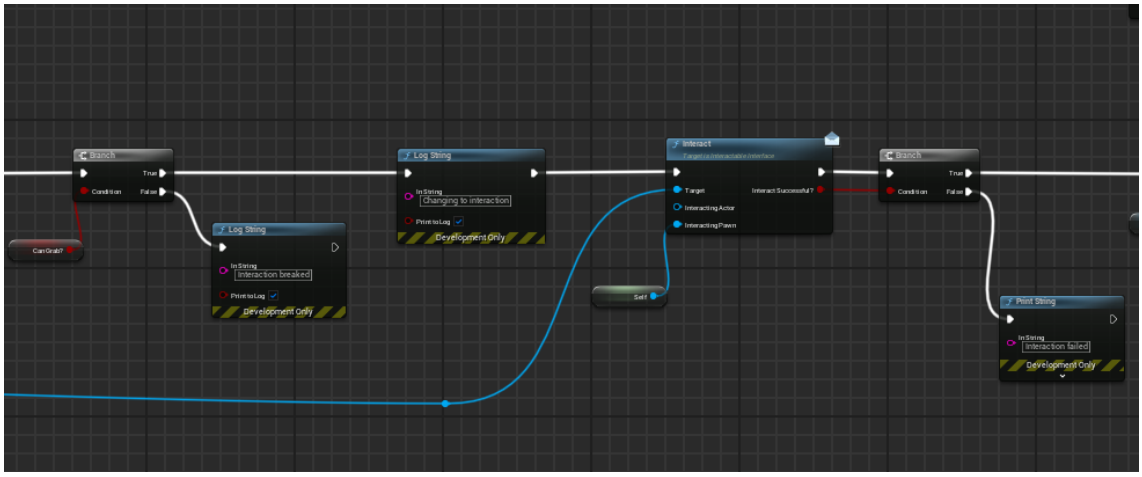
Infotekstin näyttäminen ei ollut yhtä yksinkertainen luoda, mitä PC-pelaajalle, sillä VR ei tue ”viewportissa” näkyviä ”widgettejä” tai ainakaan näitä ei suositella. VR:llä käytetään menuna erillistä ”actoria”, johon ”widget” asetetaan ja näytetään

se pelimaailmassa. VR- pelimallin mukana tulleesta ”menu actorista” luotiin duplikaatti, jota käytetään infotekstin pohjana ja se näytetään sitten ranteessa. Ongelmana oli kuinka ”actorissa” sijaitsevalle ”widgetille” pystytään välittämään tieto, mitä tekstiä halutaan näyttää esineen tarkastelun yhteydessä. Tämä ratkesi hakemalla InfoTextRef-referenssistä, joka on ranteessa näkyvä ”actor”, ”widget”-komponentti, ja käyttämällä sitä objektireferenssinä ”widgetin” hakua varten (kuvio 17).



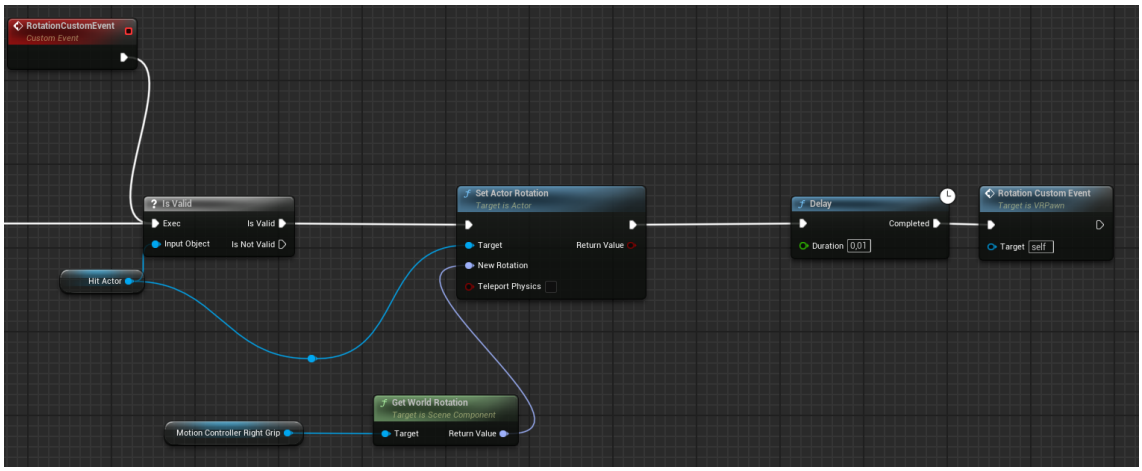
Kuvio 17. Referenssin haku

Seuraava ongelma tuli esineen tyypin mukaan esineeseen tarttuminen. Esineet, joihin ei olisi pitänyt pystyä tarttumaan toteuttivat tartuttaville esineille tarkoitetun ohjelmoinnin. Tämä siitä huolimatta, että ehtolauseke Branch-node, oli ajanut false-haaran käyttäen apuna CanGrab-booleania (kuvio 18). Joten true-haaran ohjelmointia ei olisi pitänyt suorittaa, kerta ehtolauseke oli ollut kielteinen.



Kuvio 18. Branch-haara interaktiolle

Ongelma ratkesi lopulta vaihtamalla loppuun EventTick-funktion RotationCustomEvent-funktioon ja kutsumalla sitä viiveellä uudelleen koodin lopuksi (kuvio 19). EventTick-funktio toimi PC-pelaajalla hyvin, mutta sekin on sittemmin vaihdettu samalla tavoin varmuudeksi CustomEvent-funktioksi. VR-syötteen kanssa funktio ei toiminut selkeästi, miten sen olisi tullut toimia.



Kuvio 19. CustomEvent-funktio, jota kutsutaan VR-pelaajan interaktion lopuksi

4.3 Asetusten optimointi pelimoottorilla

Tietyillä asetuksilla on vaikutusta kuinka pelitoteutus pyörii eri laitteilla. Varsinkin XR-laitteistolla toimivuuteen näillä voi olla paljonkin vaikutusta. Kaikkia listattuja asetuksia tai ehdotuksia ei ole käytetty projektissa, mutta näistä voi olla jatkokehityksessä hyötyä.

Unreal Enginessä on useita mahdollisuuksia renderöintiin, ja yleisasetuksena siinä on käytössä ”Deferred Renderer”. Vaikka tämä onkin yleisesti ottaen parhain vaihtoehto, ei se välttämättä sovi yhteen VR:n kanssa. VR:n kanssa suositellaan käyttämään **Forward shading** -asetusta, sillä se mahdollistaa nopeamman renderöinnin, mikä voi vaikuttaa parempaan suorituskykyyn VR-laitteilla. ”Forward shading” mahdollistaa paremmat anti-aliasing -asetukset tarjoten paremman mahdollisuuden vaikuttaa visuaaliseen ilmeeseen ja sen kanssa on myös mahdollista käyttää niin MSAA- ja TAA anti-aliasing -metodeja. (Unreal Engine 2024a.)

Forward-renderöinti toimii käyttämällä culling-metodia valoihin ja heijastuksen kaappauksiin frustum-tilaruudukossa. Jokaisen pikselin forward-piirto iteroi sitten

itsensä valojen ja heijastuskaappaustensa yli, jakaen materiaalit näiden kanssa. Käyttämällä forward renderöintiä ei ole mahdollista käyttää Screen Space -tekniikoita, kuten SSR, SSAO tai Contact Shadows. Myös läpinäkyvät materiaalit eivät toimi täydellisesti forward-renderöinnillä, kuten Dynamically Shadowed Translucency eli dynaamisesti varjostettu läpinäkyvyys tai läpinäkyvyyttä, joka saa varjostuksensa kiinteistä valoista. (Unreal Engine 2024a.)

”Bloom” on tehoste, jolla pyritään simuloimaan erittäin kirkkaita objekteja, joita näytöt eivät yleensä pysty renderöimään. Tämä on eräänlainen lumetehoste, jossa kirkkaana pidetylle objektille luodaan valontaitto, mikä on havaittavissa silmillä tai kameran filmillä. Bloom luodaan käyttämällä Single Gaussian -sumennusta. Parempaa laatua varten voidaan käyttää Multiple Gaussian -sumennusta. (Unreal Engine 2024b.) **”Auto exposure”** on post process volyymin asetuksia, jolla vaikutetaan kuinka valot ja varjot mukautuvat ympäristön muutoksiin kuten kävellessä ulkoa sisään (Unreal Engine 2024c).

”Motion blur” nimensä mukaisesti sumentaa objektin pohjautuen liikkeeseen ja se on suosittu tehokeino luoda immersiota peleihin. VR-laseilla tämä voi kuitenkin aiheuttaa vähintäänkin matkapahoinvointia. (Unreal Engine 2024d.) Vaikeuksistaan huolimatta voi ”motion blur” olla oikein toteutettuna tärkeä aspekti realistisessa VR-kokemuksessa; sillä voidaan luoda todellisuuden tuntoa, oikein toteutettuna pehmentää näkökokemusta ja parantaa suorituskykyä. Kaikki eivät kuitenkaan suosittele sen käyttämistä, sillä se voi vaikuttaa luonnottomalta ihmissilmään. Motion blur -tehosteen ongelma voi usein tulla vastaan XR-laitteiden rajallisuuden vuoksi, sillä osalla laitteista puuttuu kyky renderöidä tehoste oikein. (Szameitat 2023; Mack & Ruud 2019.)

Eräs matkapahoinvoinnin sekä silmärasituksen aiheuttajista VR-laseilla on liian korkea latenssi, ja onkin haastava tasapainottaa oikeaoppisesti matala latenssi samalla kun motion blur -tehoste renderöidään. Latenssilla tarkoitetaan applikaation reagointia, kun käyttäjä esimerkiksi kääntää päätänsä ja näkee tästä johtuvan visuaalisen tuloksen. Syy mahdolliseen matkapahoinvointiin johtuu siitä, että henkilö saa visuaalisen ärsykkeen liikkeestä, mutta ei saa tätä samaa signaalia sisäkorvaansa, mikä vastaa tasapainosta. Näin aistiärsykkeet ovat ristiriidassa toistensa kanssa. Yksi tavoista pienentää latenssia on tiputtaa grafiikoiden laatua nopeuden takaamiseksi. (Szameitat 2023; Mack & Ruud 2019.)

Laitekohtaiset ominaisuudet ja suorituskyky tulee ottaa huomioon, kun luodaan 3D-malleja VR-peleihin ja mikä on mallien vaikutus FPS:ään. Tehdessä malleja täytyy karsia polygoneista ja tekstuureista sieltä mistä pystyy. (Hillmann 2019.)

”Lens flare” simuloi valon taittuvuutta kirkkaasta objektista ja on nähtävissä valokehänä näkökentässä (Unreal Engine 2024e). Hajavalot, kuten lens flare, voivat häiritä visuaalista huomiota. Kameroissa hajavaloa voidaan ehkäistä fyysisesti taikka muilla keinoilla, mutta VR-tekniikoille tutkimusmahdollisuudet ovat rajallisemmat. (Li, Tsai & Lee 2022.)

Anti-aliasing-metodilla poistetaan kovia reunoja objekteista, joiden on tarkoitus olla tasaisia. Forward-renderöinnin kanssa yhteensopivia metodeja ovat MSAA eli Multi-Sample Anti-Aliasing ja TAAU eli Temporal Anti-Aliasing Upsampling.

TAAU ottaa näytteitä eri pakoista jokaisessa framessa ja käyttää aiempia frameja häivyttämään näytteet keskenään poistaen sekä pehmentäen teräviä reunoja. Upsampling-metodi on mahdollista ottaa Rendering-kohdasta pois Default- asetuksissa. MSAA käyttää tekniikkaa, jossa objektin reunat pehmennetään manipuloimalla kahden reunapikselin väriä. Tämä luo vaikutuksen pehmeämmistä ja tasaisimmista reunoista. (Unreal Engine 2024f.)

HDR mahdollistaa korkeammat kontrastit ja laajemman väriskaalan (Unreal Engine 2024g). Tämä ei välttämättä ole parhain asetus pitää VR-laitteistolle kehittäessä, sillä vahvat kontrastit voivat vahvistaa liikkeen tuntua peliympäristössä. Myös himmeämmät valot ja väriskaalat ovat parempia VR-pelikehityksen kannalta. Näin vältytään aiheuttamasta VR-käyttäjille pahoinvointia. (Mack & Ruud 2019.)

Eräs ongelma, mikä nousi esille projektin aikana, oli puiden reunojen värinä, johon anti-aliasing asetuksien muuttaminen ei toiminut. Tähän ongelmaan löytyi lopulta ratkaisuna asettaa VolumetricRenderTarget-moodi kakkoseen manuaalisesti VR-pelaajan blueprintissä. Ratkaisu löytyi Unrealin foorumeilta, sillä virallisissa dokumenteissa ei tästä löytynyt erikseen mainintaa. (Unreal Engine 2023.)

Ainoa maininta VolumetricRenderTarget moodista virallisissa dokumenteissa löytyy volumetristen pilvien dokumenteista sivuhuomiona. Dokumenteissa maini-

taan kakkosasetuksen tukevan maanäkymiä ja keskittyvän korkeampaan laatuun, mutta asetus ei tue pilvien leikkautumista läpinäkymättömien mallien kanssa toisin kuin nolla- ja ykkösasetus. (Unreal Engine 2024h.).

Volumetristä renderöintitekniikkaa on mahdollista käyttää luomaan efektejä, kuten savua, pilviä, nesteitä, sumua ja pölyä, eli kaikelle, jota on mahdoton mallintaa fyysisesti siltään (Hansen, Ikits, Kniss & Lefohn 2024). Volumetrinen renderöintitekniikka pohjautuu Beerin ja Lambertin lakiin, jolla lasketaan massan läpäisevien fotonien määrä läpäisevyyden määrittämiseksi. Näillä tiedoilla on mahdollista laskea lopullinen valaisu. Massan hiukkasilla on lisäksi vaikutusta, kuinka valo taittuu massan läpäistessään, mutta myös kuinka valoa heijastuu pois massasta. (Tarton yliopisto 2024.)

VR:llä ilmenevät ongelmat volumetrisian pilvien kanssa voivat hyvin johtua pilvien taittumisesta läpinäkymättömistä materiaaleista, sillä VR-lasit eivät toimi hyvin läpinäkyvien materiaalien kanssa, jolloin kiinteän objektin läpäisevä valo voi aiheuttaa samaa efektiä. Muita mahdollisia syitä tälle voi olla itse valon taittuminen objektin pinnasta itse volumetrisen renderöinnin yhteydessä, jolloin VR-laseilla on hankaluuksia piirtää tätä oikein. VolumetricRenderTarget-moodin laittamisen jälkeen testailtu vaihtaa anti-aliasing asetuksia ja huomattu ongelman palaavan MSAA anti-aliasing -asetuksella, joten asetukset jätettiin TAA-asetuksille.

Myöhemmin projektista otettiin volumetriset pilvet kokonaan pois käytöstä VR-kentässä, sillä nämä ovat graafisesti raskaita pyörittää (Taikina-Aho 2024). Kuitenkin on mahdollista käyttää myös VR-projektissa volumetrisiä pilviä, mutta on suotavaa katsoa, kuinka nämä vaikuttavat performanssiin. Myös asetuksia tulee säätää, mutta näihin on listattuna yllä muutamia ratkaisuja.

Materiaalien tekstuureissa ei ole suotavaa käyttää normal map -tekstuuria VR-pelikehityksessä ja esimerkiksi pintamateriaaleissa normal map toimii vain hieman kauempaa katsottuna. Tämä johtuu siitä, ettei normal map -tekstuuri ole stereoskooppinen ja VR-laseilla näyttää liian tasaiselta katsottaessa lähietäisyydeltä. Täysin turha normal map ei kuitenkaan ole, sillä sitä käytetään valojen apuna. PBR-materiaalit voivat olla suorituskykyä heikentäviä yhdessä dynaamisten valojen kanssa. Yksi ratkaisu tähän on käyttää tietyllä alueella tai

tietyssä esineessä tarkempia yksityiskohtia ja loput tämän ympärillä tulisi vetää baked-metodilla yksiväriseksi kartaksi. (Hillmann 2019.)

Esirenderöidyt grafiikat ovat useimmiten kustannustehokkaampia VR-projekteja varten. Tämä tapa vähentää pelistä ja visualisoinnista interaktiota, mutta sen avulla saadaan aikaan näyttävämmät grafiikat. Renderöinnissä voidaan käyttää 360 asteen kuvia, joita varten Unreal Enginessä on hyvä lisäosa nimeltä Stereo Panoramic Movie Capture. 360-kuvia käytettäessä tulee kuitenkin huomioida niiden renderöintiin kuluva aika. (Hillmann 2019.)

“Depth of Field” -asetusta ei tulisi käyttää, sillä ei ole varmuutta käyttäjän katseen suunnasta juuri sillä hetkellä. Suurin osa nykypäivän VR-tekniikoista ei tue tällaista ominaisuutta. Toinen asetus mihin ei saisi koskea on **“Field of View”**. Ihmissilmät eivät toimi kuten kameranlinssi, ja tämän kaltaiset asetukset voivat aiheuttaa pahoinvointia pelaajassa. (Mack & Ruud 2019.)

Pelaajan kameraa ei tulisi kiihdyttää taikka hidastaa eikä myöskään liikutella irti pelaajasta. Hyviä tapoja toteuttaa hahmon liikkuminen pelikentässä on tasainen liike tai sitten teleporttauksella toimiva liike, ja liikkeen tulisi lähteä aina pelaajan syötteestä. Näin varmistetaan ettei pelaajalle aiheudu yhtäkkistä matkapahoinvointia. (Mack, K., & Ruud, R. 2019.)

4.4 Visualisoinnin jatkokehitys

Opinnäytetyötä varten toteutetussa projektissa on mahdollisuus muun muassa tarkastella eri objekteja sekä lukea näistä lisätietoja samalla. Tähän on helppo jatkokehittää ja tehdä visualisoinnista enemmän interaktiivinen, sillä esineisiin on jätetty mahdollisuus käyttää niitä myös muuten kuin ottamalla esine käteen pelimaailmassa. Jatkokehitysideana voisi toimia toiminnallisuuden kannalta esineen ”blueprintin” muuttamista hieman siten, että interaktiossa esine ei automaattisesti teleporttaisi sille osoitettuun pisteeseen pelaajahahmoa, vaan tarkistaisi ensin mitä tyyppiä esine on ja sitten toteuttaisi halutun haaran. Pelaajien ”blueprinteissä” toteutukseen ei tarvittaisi muuta muutosta kuin esineen tyypin tarkastuksen jälkeen toisen interaktio viestin esineelle.

Prototyyppi toteutettiin käytettävissä olevien resurssien kuten ajan ja mallien puitteissa. Lopullista koontiversiota varten kenttiä tulisi joko jatkaa lopulliseen suuntaansa taikka luoda uudet kentät tilalle. Eräs kenttien luomiseen vaikuttava asia oli tarkkojen mallien puute niin museoesineistä ja rakennuksista, mutta myös oikeanlaisten ympäristömallien puute. Metsämuseon tilat ovat kasvustoltaan hyvin perus suomalaista mäntymetsää eikä näihin ole aivan kaikkia mahdollisia 3D-malleja tarjolla ilmaiseksi, jotka olisivat erittäin tähdellisiä, ottaen huomioon suomalaisten ja museon luontosuhteen.

Kenttään ei myöskään lisätty kuin testimielessä ääniä, joten äänimaailmaan tulisi panostaa, jotta pelillistetty visualisointi olisi immersioiva. Tätä varten vaadittaisiin sopivien ääniraitojen luontia ja asettamista pelimaailmaan. Myös valotuksen säätämisellä voitaisiin vaikuttaa immersioon. Kaikkia sen mahdollisuuksia pelimoottorina ei tämän opinnäytetyön aikana käyty edes läpi, kuten nanite-teknologiaa tai MetaSound-audiotekniikkaa, mitkä ovat Unreal Enginen tarjoamaa uutta tekniikkaa. Näitä kahta olisi hyvä jatkotutkia ja mahdollisesti hyödyntää tulevissa projekteissa.

Demo toteutettiin toimivaksi vain tietokoneelle ja VR-laitteille. Unreal Enginellä on kuitenkin mahdollista toteuttaa mobiiliversiokin visualisoinnista. Mobiiliversiota varten pelikenttä tulisi suunnitella mahdollisimman kevyeksi ja malleiltaan suhteellisen yksinkertaiseksi, sillä mobiililaitteet ovat renderöintitehokkuudeltaan ja muistiltaan heikompilaatuisia mitä esimerkiksi tietokone. Tällä hetkellä demossa ei ole mitään mobiiliversioon sopivaa koodia eikä assetteja, joten sitä varten vaadittaisiin huomattavaa työmäärää. Unrealilla ei ole enää mahdollisuutta luoda WebGL koontia peleistä, joten tätä varten voisi käyttää Unitya. Eräs mahdollisuus onkin toteuttaa WebGL ja mobiiliversiot omana Unity-projektinaan, sillä molemmat vaativat suhteellisen kevyen toteutuksen.

5 POHDINTA

Prototyyppi visualisoinnista toteutettiin ajan ja mallien puitteissa, ja toteutuksessa painotettiinkin tutkimaan kuinka Unreal Engine toimii monialusta-pelikehityksessä ja sen mahdollisia haasteita. Olettamus ennen tutkimustyön aloittamista oli, että Unreal Engine antaisi uusien ominaisuuksiensa, kuten naniittien, ylivoimaisesti näyttävän tuloksen vähintäänkin tietokoneella ja VR-laitteella. Unreal Engine onkin kehuttu pelimoottori johtuen kuinka näyttäviä ja realistisia tuloksia sillä voidaan saada aikaan. Tämä olettamus paljastuikin astetta hankalammaksi toteuttaa.

Vaikka monessa asiassa Unreal Engine onkin parempi kilpailijoihinsa nähden, on se kuitenkin monialusta-kehityksessä haastavampi alusta. Sillä on heikommin tuettuja alustoja ja varsinkin WebGL-tuen puute on haittaava tekijä, johtuen kuinka kysytty tekniikka se on varsinkin joidenkin yritysten puolesta. Myös jotkin tärkeät kehitystyökalut ovat hankalia tuoda projektiin, sillä monet näistä eivät tule moottorin mukana vaan pitää hankkia joko Marketplacesta tai työkalujen omilta sivustoilta. Eräs tärkeä työkalu VR-kehityksen kannalta on muun muassa XR-simulaattori, mutta ainoa tarjoaja tälle oli Meta. Näin vahva yhteistyö suuren ja kaupallisen teknologiayrityksen kanssa, jolla on melkeinpä monopoliasema, voi vaikuttaa joidenkin kehittäjien päätöksiin kehitysalustan valinnassa.

Eräs itse opinnäytetyön raportin kirjoittamista haittaava seikka oli lähteiden muuttuminen kesken kirjoitusprosessia. Vielä 2024 vuoden kesään asti WebGL-lisäosasta löytyi dokumentaatio Unreal Enginen virallisista dokumentaatioista, mutta tämä poistettiin kesken opinnäytetyötä. Poiston taustalla oleva syy oli luultavimmin vanhentunut Unreal Enginen versio, jolle ulkoinen WebGL-lisäosakin oli, sillä Unreal Enginen dokumentaatio alkaa nykyisellään 4.27 ylöspäin. WebGL:stä kertovaa osiota ei kuitenkaan haluttu poistaa, joten se sisällytettiin opinnäytetyön raporttiin.

Lukuisia haasteita tuli vastaan myös dokumentoinnin sekavuudessa ja yhtä aikaa sen puutteessa, sillä moni asetus mitä VR-laitteillekin suositellaan ei vastannut tärkeään kysymykseen; miksi asetus on suositeltu. Tämä varsinkin aloittelijan näkökulmasta on ikävää, sillä se tieto olisi tärkeä aspekti pelikehityksessä.

Prototyypissä onnistuttiin luomaan toiminnallisuus esineen tarkastelulle mitä Metsämuseo halusikin mahdolliselta VR-toteutukselta. Esineiden tarkastelussa toteutuu myös esineiden historiikin kertominen pelaajalle. Prototyyppiä on helppo lähteä jatkohyödyntämään toiminnallisuutensa puolesta.

Osa lähteistä antoi vinkkiä mitä kannattaisi tehdä, kuten auto exposure, bloom ja ambient occlusion asetukset on kannattavaa laittaa pois päältä. Mutta lähteistä ei aina käynyt ilmi miksi juuri näin kannattaisi tehdä. Myös ristiriitaista tietoa tuli vastaan. Esimerkiksi Mack ja Ruud toteavat kirjassaan, ettei motion blur asetusta olisi suotava käyttää VR-projektissa (Mack & Ruud 2019), mutta Szameitat toteaa artikkelissaan, että oikein käytettynä motion blur-asetus voi tehostaa VR-kokemusta (Szameitat).

Tästä huolimatta Unreal Enginellä on potentiaalia monialusta-pelikehityksessä. Hillman totesi kirjassaan Unreal Enginen olevan helppokäyttöinen ”blueprint-tiensä” avulla (Hillmann 2019) ja tämä todettiin opinnäytetyön aikana pitävän paikkansa. ”Blueprintit” auttavat huomattavasti ohjelmoinnissa, sillä niiden avulla koodista tulee luettavampaa eikä virheitä tarvitse metsästää erillisiltä koodiriveiltä. Kaikkia Unreal Enginen mahdollisuuksia ja ominaisuuksia ei tutkittu opinnäytetyön aikana ja sen täyttä potentiaalia ei päästy näin testaamaan.

LÄHTEET

Agisoft 2024. Metashape 2.1 Photogrammetry + LiDAR. Viitattu 4.9.2024 https://www.agisoft.com/pdf/metashape_presentation.pdf

Jalli, A. 2024. What Is a Library in Programming? A Complete Guide. Codingem.com, blogi. Viitattu 23.10.2024 https://www.codingem.com/what-is-a-library/

Blender 2024. Home of the Blender project – Free and Open 3D Creation Software. Viitattu 14.10.2024 https://www.blender.org/

Bradfield, C. 2018. Godot Engine game development projects: Build five cross-platform 2D and 3D games with Godot 3.0. Birmingham: Packt Publishing. Viitattu 22.8.2024 https://ebookcentral-proquest-com.ez.lapinamk.fi/lib/ulapland-ebooks/detail.action?docID=5446052

Calvello, M. 2022. What Is UV Mapping? How It Makes 3D Models Come to Life. G2.com, Inc. Viitattu 31.10.2024 https://www.g2.com/articles/uv-mapping

Digital Extremes 2024. About us. Viitattu 22.8.2024 https://www.digitalextremes.com/about

Ezpeleta, A. J. P., Raposas, K. A. M., Tiglao, N. M. C. & Limjoco, W. J. 2018. Application Development for the Smart Surface Using Game Engines Unity and Unreal Engine. Viitattu 30.10.2024 https://d1wqtxts1xzle7.cloudfront.net/60619674/UCL01_Almost_Final_Draft20190916-978-1fahat3-libre.pdf?1568703427=&response-content-disposition=inline%3B+filename%3DUCL01_Almost_Final_Draft.pdf&Expires=1729676237&Signature=LYN-UzNfhgpCMObFFjF204PU-FUIdL4q7gcpFde5hMeVgeQsMyuBrCpwislr00zmgacwIke7iM-mpXOfifEVY0YJsTBAfpbNoOwnN-ME5Z6GiZ-RmnLLKk2PlJzmBmdadAtTpnsZcod6Bmv9CWlK8R8Nh2-8V4fubipI0Qtuismrs8kxDXFkGC4QX0b6omf37lGjzWIOL6AWeE-30LGZhPWpdTeg2UQ6bGtKjHohMjJzryHh935O0ocxfaxwdTY-FvWIHd3Y7wzDBsysPNdism-tEGCYpQAEOta1Ksw6CIJkGo9RGKd9W2Q7SRaCz~P2CGpDfGNQV-r4LwfM4xGcdgnCNhQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Hansen, C., Ikits, M., Kniss, J. & Lefohn, A. 2024. Volume Rendering Techniques. NVIDIA Corporation, GPU Gems part 7, chapter 39. Viitattu 13.9.2024 https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques

Hillmann, C. 2019. Unreal for mobile and standalone VR: Create Professional VR apps without coding. Apress. Viitattu 30.10.2024 https://books.google.fi/books?hl=fi&lr=&id=YjuSDwAAQ-BAJ&oi=fnd&pg=PR5&dq=mobile+hdr+and+vr&ots=ppfimT7Q9-&sig=8x6YWjWyefYNXc8fXgONfJy0k6Q&redir_esc=y#v=onepage&q=mobile%20hdr%20and%20vr&f=false

IBM 2024. What is LiDAR? Viitattu 20.11.2024 https://www.ibm.com/topics/lidar

Jensen K. 2023. 25 Years Later: The History of Unreal and an Epic Dynasty. PC Mag. Viitattu 22.8.2024 https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty

Lee, A. & Jang, I. 2019. Implementation of an open platform for 3D spatial information based on WebGL. *ETRI Journal*, *41*(3), 277–288. Viitattu 24.10.2024 https://doi.org/10.4218/etrij.2018-0352

Li, H. C., Tsai, M. C. & Lee, T. X. 2022. A stray light detection model for VR head-mounted display based on visual perception. *Applied Sciences*, *12*(13), 6311. Viitattu 30.10.2024 https://doi.org/10.3390/app12136311

Mack, K. & Ruud, R. 2019. *Unreal Engine 4 virtual reality projects: build immersive, real-world VR applications using UE4, C++, and unreal blueprints*. Packt Publishing Ltd. Viitattu 30.10.2024 https://books.google.fi/books?id=ZiyWDwAAQBAJ&lpg=PP1&ots=0Ve03bldIs&dq=vr%20and%20unreal%20engine&lr&hl=fi&pg=PA25#v=onepage&q=vr%20and%20unreal%20engine&f=false

Meta Quest 2024. Getting Started with Meta XR Simulator. Viitattu 13.9.2024 https://developer.oculus.com/documentation/unreal/xrsim-getting-started/

Nivala, J. 2024. FrostBit. Keskustelu projektityöntekijän kanssa 30.8.2024

Oztel, I., Yolcu Oztel, G. & Sahin, V. H. 2023. Deep Learning-Based Skin Diseases Classification using Smartphones. *Advanced Intelligent Systems*, *5*(12), 2300211. Viitattu 24.10.2024 https://doi.org/10.1002/aisy.202300211

Perforce 2020. What You Need For Cross Platform Game Development. Viitattu 31.7.2024 https://www.perforce.com/blog/vcs/cross-platform-game-development

Rizwan, I. 2024. Beyond native: An overview of cross-platform development. Educative. Viitattu 31.7.2024 https://www.educative.io/blog/cross-platform-development-overview

Stackoverflow 2023. Unreal Engine 5 support HTML5? Viitattu 26.8.2024 https://stackoverflow.com/questions/76291206/unreal-engine-5-support-html5

Steamworks 2024. Tarkistusprosessi. Valve. Viitattu 31.7.2024 https://partner.steamgames.com/doc/store/review_process

Szameitat, C. 2023. Motion Blur in XR (S01/E30). Tech Insights by The LBMA 27.10.2023. Viitattu 1.8.2024 https://thelbma.substack.com/p/motion-blur-in-xr-s01e30

Taikina-Aho, J. 2024. FrostBit. Keskustelu projektipäällikön kanssa 12.11.2024.

Tarton yliopisto 2024. Advanced Computer Graphics (WIP): Volumetric Rendering. Viitattu 30.4.2024 https://cglearn.eu/pub/advanced-computer-graphics/volumetric-rendering

Unity 2024. XR Device Simulator. Viitattu 13.9.2024 https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-device-simulator.html

University of Maryland 2019. The Epic World of Tim Sweeney. Viitattu 22.8.2024 https://mage.umd.edu/news/story/the-epic-world-of-tim-sweeney

Unreal Engine 2022. “Does Unreal Engine 5 supports WebGL? If not, will some day be supported?” Viitattu 26.8.2024 https://forums.unrealengine.com/t/does-unreal-engine-5-supports-webgl-if-not-will-some-day-be-supported/541157

Unreal Engine 2023. Flickering problems in UE5 using Megascans trees. Forums. Viitattu 30.7.2024 https://forums.unrealengine.com/t/flickering-problems-in-ue5-using-megascans-trees/720115/4

Unreal Engine 2024a. Forward Shading Renderer. Viitattu 29.7.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/forward-shading-renderer-in-unreal-engine

Unreal Engine 2024b. Bloom. Viitattu 29.7.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/bloom-in-unreal-engine

Unreal Engine 2024c. Auto Exposure. Viitattu 29.7.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/auto-exposure-in-unreal-engine

Unreal Engine 2024 d. 1.12 - Motion Blur. Viitattu 1.8.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/1.12---motion-blur?application_version=4.27

Unreal Engine 2024e. Lens Flare. Viitattu 29.7.2024 https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/LensFlare/

Unreal Engine 2024f. Anti-Aliasing and Upscaling. Viitattu 29.7.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/anti-aliasing-and-upscaling-in-unreal-engine

Unreal Engine 2024g. High Dynamic Range Display Output. Viitattu 29.7.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/high-dynamic-range-display-output-in-unreal-engine

Unreal Engine 2024h. Volumetric Cloud Component. Viitattu 30.7.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/volumetric-cloud-component-in-unreal-engine?application_version=5.4

Unreal Engine 2024i. Enhanced Input. Viitattu 26.8.2024 https://dev.epicgames.com/documentation/en-us/unreal-engine/enhanced-input-in-unreal-engine