OULUN AMMATTIKORKEAKOULU

Jani Korkiakoski

VIRTUAALILASIEN KÄYTTÖ UNITY-PELIMOOTTORILLA RAKENNETUSSA 3D-VISUALISOINTISOVELLUKSESSA

VIRTUAALILASIEN KÄYTTÖ UNITY-PELIMOOTTORILLA RAKENNETUSSA 3D-VISUALISOINTISOVELLUKSESSA

Jani Korkiakoski
Opinnäytetyö
Kevät 2016
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä(t): Jani Korkiakoski
Opinnäytetyön nimi: Virtuaalilasien käyttö Unity-pelimoottorilla rakennetussa 3D-visualisointisovelluksessa
Työn ohjaaja(t): Risto Korva
Työn valmistumislukukausi ja -vuosi: Kevät 2016 Sivumäärä: 32

Työn tavoitteena oli tutustua virtuaalitodellisuuden kokeelliseen soveltamiseen Unity-kehitysympäristössä sekä soveltaa virtuaalilasien käyttöä visualisointisovellukseen rakennusteollisuuden tarpeisiin. Viimeisenä tavoitteena oli lopputuote, jossa virtuaalilaseista olisi esityksellistä hyötyä visualisointisovelluksen käytössä.

Työssä tutustuttiin aluksi virtuaalitodellisuuden soveltamiseen, tulevaisuuden virtuaalitodellisuuden kehittämiseen sekä oman työn tekniseen toteuttamiseen. Pääasiallisesti sovellusta kehitettiin Oculus Rift -lasien kanssa GB4D-sovelluksen sisällä, ja tutkittiin sen soveltuvuutta sovelluksen tarpeisiin. Koska virtuaalilasit ovat kolmannen osapuolen laitteita, oli työn luonne suurimmaksi osaksi ohjelmistokehitykseen soveltuva. Laitteistollinen puoli käsitteli ainoastaan virtuaalilasien ominaisuuksien sekä vastaanotettavan datan tutkimista.

Lopputuloksena valmistui toteutus, jossa virtuaalilaseja voitiin hyödyntää GB4D-sovelluksen sisällä. Sovelluksesta on jo olemassa valmiina versio kiinteistöjen piirustusten kolmiulotteista tarkastelua varten, joten yritys jää miettimään, onko toteutuksen jatkokehittäminen tai julkaisu kannattavaa.

Asiasanat: Unity, virtuaalitodellisuus, rakennusteollisuus

ALKULAUSE

Haluaisin kiittää koko Group Builderin väkeä opinnäytetyön mahdollistamisesta ja työavusta työpaikalla sekä työajan ulkopuolella. Työn aiheeksi saatiin muodostettua tuore ja mielenkiintoinen aihe, jossa pystyttiin kevyesti mutta tehokkaasti yhdistämään ohjelmistollista sekä hiukan laitteistollista suunnittelua ja kehittämistä. Opinnäytetyön tyylikäs toteutus ei olisi ollut mahdollista ilman Group Builderia ja sen osaavaa henkilöstöä. Erityiskiitokset annan Marko Savolaiselle ja Ari Oinaalle opinnäytetyön sisällön tutkimisesta sekä teknisten termien määrittelystä.

Oulussa 12.4.2016

Jani Korkiakoski

SISÄLLYS

SANASTO

API	Application Programming Interface on ohjelmointirajapinta, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja keskustella keskenään
BIM	Building Information Model on rakennuksen ja rakennusprosessin koko elinkaaren aikaisten tietojen kokonaisuus digitaalisessa muodossa
CAD	Computer-aided Design on ohjelmiston käyttöä apuvälineenä insinöörien ja arkkitehtien harjoittamassa suunnittelutyössä
Event	Tapahtuma, joka kutsuu kaikkia siihen liitettyjä funktioita ohjelmakoodissa
EventSystem	Unityn tapahtumakutsuja prosessoiva ja käsittelevä Unity-komponentti
GB4D	Ohjelmisto, jolla tarkastellaan CAD-ohjelmalla piirrettyä kiinteistöä kolmiulotteisena
GameObject	Kaikkien entiteettien pohjaluokka Unity-sceneissä
GB Core	GB-ohjelman selainympäristö
Google Cardboard	Googlen mobiilityyppinen toteutus virtuaalisovellusalustasta
Huonekortti	Suunnittelua ja kustannuksia varten tarkoitettu informaatio rakennusteollisuudessa
IFC	Industry Foundation Class on kansainvälinen ISO/PAS 16739 -standardi oliopohjaisen tiedon siirtoon
Instantiointi	Objektin esiintymän luominen sovelluksessa
JSON	Javascript Object Notation on yksinkertainen avoimen standardin datamuoto
Oculus Rift	Oculus LCC:n kehittelemät virtuaalilasit tietokonesovelluksille

Prefab	GameObject, joka sisältää valmiiksi asetettuja scriptejä ja komponentteja
Samsung Gear VR	Samsungin kehittämät virtuaalitodellisuuslasit mobiililaitteille
Scene	Yksittäisen pelitilan instanssi Unityn käyttöliittymässä
Script	Ohjelmakooditiedosto Unityssä
Transform	GameObjectin paikka- ja rotaatiotiedot sisältävä komponentti
Unity	Sovelluskehitysympäristö kolmiulotteisten sovellusten, kuten pelien tai muiden ohjelmistojen kehitykseen
WebGL	3D-grafiikkaa esittävä JavaScript-rajapinta WebGL-yhteensopivissa selaimissa

1 JOHDANTO

Työn tarkoituksena oli tutkia ja kehittää Group Builder Oy:n GB4D-sovellusta virtuaalilaseille sopivaksi käyttäen Unity-pelimoottoria. Ensisijainen virtuaalitodellisuuden kehitysvaihtoehto oli Oculus Rift -virtuaalilasit. Vaihtoehtoisista virtuaalilaseista tehtiin tutkimustyötä, jotta voitiin selvittää, voitaisiinko niitä hyödyntää Group Builderin kehittämässä visualisointisovelluksessa. Myöhemmin työn toteutuksen aikana todettiin, että mobiilipohjaiset virtuaalialustatoteutukset kuten Google Cardboard ja Samsung Gear VR osoittautuivat kaupallisesti järkevimmiksi virtuaalitoteutusalustoiksi.

GB4D on visualisointisovelluksena asiakaslähtöiseen rakentamisen avuksi tarkoitettu ohjelma, jolla pohjapiirustuksia saadaan kolmiulotteiseen muotoon asiakasystävälliseksi. GB4D voidaan Unity-sovelluksena kääntää monelle eri alustalle, kuten Android-käyttöliittymälle ja WebGL-sovellukseksi. Ohjelmiston tärkeimpänä saavutuksena oli WebGL-käännöksen onnistuminen, sillä WebGL-versiona sovellusta voidaan isännöidä selainystävällisenä sellaisenaan siten, ettei erillisiä lisäosia selaimeen tarvitse asentaa. Yritysten käyttöön sovellus tulee toimimaan pääasiallisesti verkkosivuun upotettuna WebGL-sovelluksena. Virtuaalilasisovellusta varten ohjelma käännetään Windows-alustalle Oculus Rift -sovellusta varten ja Android-alustalle Google Cardboard -sovellutusta varten.

Group Builder Oy on vuonna 2012 perustettu asiakaslähtöisen rakentamisen yritys. Yrityksen päätuotteisiin kuuluvat muutoksenhallintasovellus, sekä kommunikointityökalu ostajan ja rakentajan välillä. Kiinteistöjen tarkasteluun yritys tarjoaa ohjelmiston, joka esittää BIM-tietomallit kolmiulotteisena representaationa ja jossa kiinteistöä voidaan tarkastella ja sisustaa.

2 UNITY-PELIMOOTTORI

Unity Technologiesin perustivat vuonna 2014 David Helgason, Nicholas Francis ja Joachim Ante. Miehet näkivät potentiaalisuuden pelimoottoreissa sekä ohjelmointityökaluissa ja lähtivät kehittelemään pelimoottoria, jota voitaisiin helposti käyttää edulliseen hintaan. Idean mukaisesti miehet lähtivät kehittämään pelimoottoria, joka tunnetaan nykyisin nimellä Unity. Vuonna 2015 Unityllä on 45 prosentin markkinaosuus pelimoottoreiden myynnissä maailmanlaajuisesti, ja rekisteröityneitä kehittäjiä on noin 4,5 miljoonaa. (Company Facts. 2015; Takahashi 2014.)

Unity on Unity Technologiesin kehittelemä pelimoottori, jota pääasiallisesti käytetään kolmiulotteisten sovellusten kehittämiseen. Unity käyttää ohjelmointiympäristönään MonoDevelopia, joka on avoimen lähdekoodin ohjelmointiympäristö. MonoDevelopin on kehittänyt Xamarin Incorporation San Franciscossa yhdessä MonoDevelopin yhteisön kanssa ja kehittäjät käyttävät sitä ohjelmakoodin kirjoitukseen yleensä C#- tai JavaScript-kielellä. MonoDevelop tukee kuitenkin muitakin ohjelmointikieliä kuten F#, C, C++ ja Vala. Unityllä ohjelmia voidaan kirjoittaa usealle eri alustalle ja tällä hetkellä sovelluksia voidaan kääntää ainakin viidelletoista eri alustalle, kuten esimerkiksi Android, iOS ja WebGL. (Takahashi 2014; Company Facts. 2015; MonoDevelop. 2015.)

2.1 Unity kehitysympäristönä

Ennen varsinaisen pelimoottorin syntyä Unityn kehittäjät kehittivät vuonna 2004 videopelin nimeltään GooBall, josta tuli varhaisin Unityyn kytköksissä oleva videopeli. Peli ei saavuttanut haluttua suosiota kaupallisilla markkinoilla, joten kehittäjät päättivät siirtyä videopelien kehittämisestä Unity-pelimoottorin kehitykseen. Vuoteen 2015 mennessä Unityllä on kirjoitettu useita tuhansia sovelluksia ja sillä on yli 5 miljoonaa rekisteröitynyttä kehittäjää. (Cook 2015; Unity. 2016.)

Unity on pääasiassa tarkoitettu pelikehittäjille, mutta sillä on myös kehitetty muitakin hyötysovelluksia kuten 3D-animaatioita, grafiikkademoja ja teollisuuteen sekä luomiseen liittyviä sovelluksia kuten Elbphilharmonie 3D, The coherentrx suite ja Minutely (Unity. 2016). Unity-sovellusten kehitysperiaate on kuitenkin avoin ja kehitetyt sovellukset voivat pohjautua mihin ideaan tahansa.

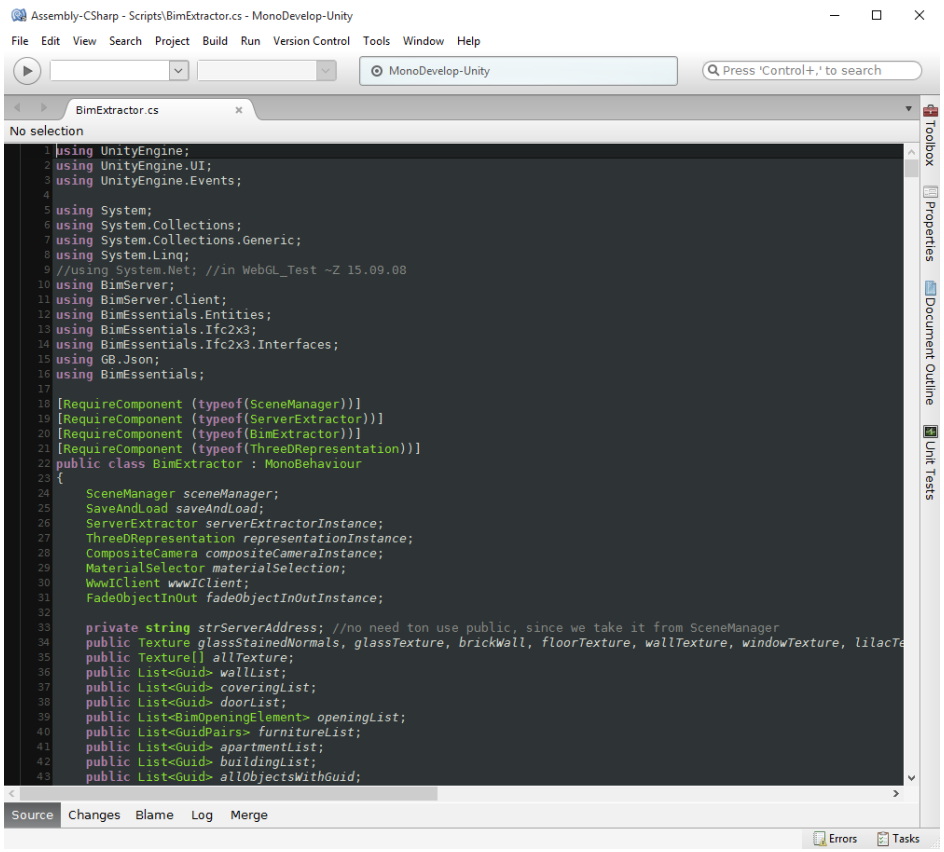
Kehitysympäristönä Unity toimii joustavasti, sillä kehitettävään ohjelmaan voidaan vaikuttaa sen ohjelmakoodissa, sekä audiovisuaalisella puolella. Grafiikkaa, ääntä, animaatioita ja useita muita tiedostotyyppejä voidaan viedä projektiin, mikä sallii kehittäjien luoda minkä tahansa näköisiä sovelluksia. Group Builder Oy valitsi Unityn GB4D:n moottoriksi, sillä kaikilla osallisilla kehittäjillä oli käyttökokemusta Unitystä. Unity tukee myös projektin WebGL-versioksi kääntämistä, mikä on valittu GB4D:n pääasialliseksi sovellusmuodoksi. GB4D:n muita käyttämiä alustoja ovat Android sekä Windows standalone.

2.2 MonoDevelop ohjelmakoodin kehitysympäristönä

MonoDevelop (kuva 1) on avoimen lähdekoodin kehitysympäristö, joka toistaiseksi on tullut jokaisen Unity-version mukana vuoden 2016 alkuun mennessä. MonoDevelopilla Unityyn ohjelmoitaessa voidaan käyttää joko JavaScript-, C#- tai Boo-kieltä. Yhtä sovellusta voidaan myös ohjelmoida usealla eri ohjelmointikielellä. MonoDevelopin käyttämä Mono-kehitysalusta on yhteensopiva .NET Framework -kirjastojen avulla käännettyjen kirjastojen kanssa. MonoDevelop sekä Mono soveltuvat käytettäväksi Unityn kanssa, sillä ne mahdollistavat yhdessä ohjelmien kirjoituksen usealle eri alustalle siten, että ohjelmakoodi on yhteensopivaa .NET Framework-kirjastojen kanssa. (Mono FAQ General. 2015.)

MonoDevelopin virheenkorjausjärjestelmä on toteutettu siten, että kun ohjelmakoodi halutaan suorittaa tiettyyn pisteeseen, virheenkorjaaja liitetään MonoDevelopista Unityn Editorissa suoritettavaan ohjelmaan ja ohjelma suoritetaan.

Ohjelman suoritusta voidaan tarkastella Unity Editorista ja kun ohjelmassa päästään riville, jossa ohjelmakoodiin asetettu pysäytyspiste on, ohjelman suoritus keskeytetään ja ohjelmakoodia voidaan tarkastella kuten useissa muissa ohjelmakoodien kehitysympäristöissä. Ohjelmakoodin kehitysympäristö jota Unity Editor käyttää, on mahdollista vaihtaa asetuksista. Unity 5 -version asennuksessa voi vaihtoehtoisesti asentaa Visual Studio Community 2015 -ympäristön, joka ohjelmakoodin kirjoituksen kannalta toimii teknisesti samoin, mutta sallii Visual Studiota käyttävien kehittäjien käyttää tuttavallisempaa kehitysympäristöä. (Kuva 1.)



KUVA 1. Kuvankaappaus Monodevelop-ympäristöstä

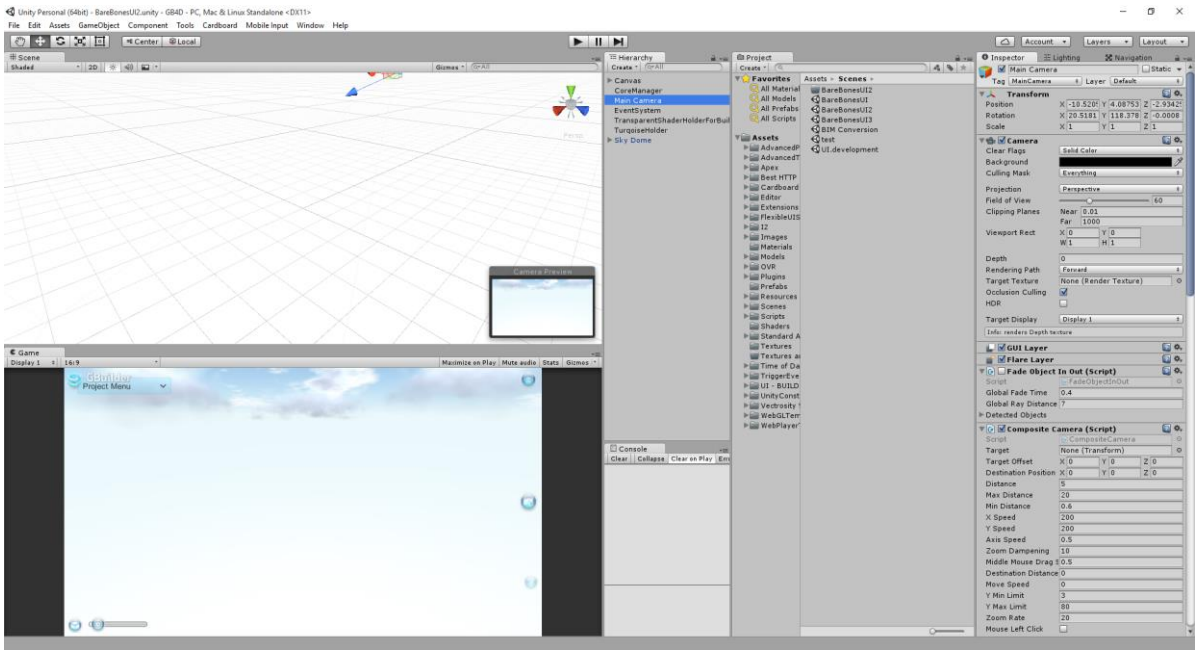
2.3 Mono-ohjelmakoodikirjasto

Unity-ympäristön yleinen ohjelmakoodin kirjoittaminen on toteutettu Mono-kirjastolla, joka on avoimen lähdekoodin UNIX-tyyppinen toteutus Microsoftin .NET-kirjastosta. Monossa toteutetaan myös useita eri C#-ohjelmointikieleen perustuvia komponentteja ja runkoja. Visual Studiolla kirjoitettuja kirjastoja voidaan hyödyntää Mono-kirjaston kanssa. Monon tarkoituksena on mahdollistaa UNIX-kehittäjien .NET-applikaatioiden alustavapaan kehittämisen. (Mono FAQ General. 2015.)

2.4 Unity Editorin käyttöliittymän kuvaus

Unityn pääkäyttöliittymäympäristö on Editor-ikkuna (kuva 2), josta projektin kokonaisuutta kontrolloidaan. Editor-ikkunassa on sijoitettuna oletusasettelussa hierarkia Sceneen sijoitetuista objekteista, lista projektiin sisälletyistä tiedostoista, 3D-tilaa kontrolloivan kameran ikkuna, esikatselukameran ikkuna sekä yksittäisen peliobjektin tarkastelua varten luettelo kyseisen objektin komponenteista (kuva 2).

Kolmiulotteinen tila Unityssä perustuu normaaliin XYZ-koordinaatistoon, missä jokaisella sceneen sijoitetulla gameobjectilla on transform-komponentti, mikä pitää sisällään objektin paikka- ja rotaatiotiedot koordinaatistossa. Gameobject sisältää kaikki komponentit, kuten vaikkapa transform-komponentin. Gameobjectin sisältöä voidaan visuaalisesti tarkastella Inspector-välilehdessä, joka on yleensä Editor-ikkunan oikealla reunalla. Kolmiulotteiseen tilaan sijoitettu gameobject voidaan nähdä scene-ikkunassa vasemmassa ylänurkassa. (Kuva 2.)



KUVA 2 Unityn Editor-ikkuna kehitysympäristönä

Projektia voidaan hallinnoida Editor-ikkunassa visuaalisesti. Esimerkkitapauksessa peliobjektia kontrolloiva C#-tiedosto kirjoitetaan ja tallennetaan projektikansioon. Kun tiedosto on tallennettu, se ilmestyy näkyviin projektitiedostojen hierarkiaan assets-nimiseen kansioon. Ellei ohjelmakooditiedoston luokka peri MonoBehaviour-luokkaa, sitä voidaan kutsua normaalisti luomalla siitä olio. Yleensä Unityn C#-tiedoston luokka perii MonoBehaviour-luokan, jolloin sovellus pystyy hyödyntämään yleisimpiä MonoBehaviour-luokan ominaisuuksia kuten Start- ja Awake-funktioita. (Unity MonoBehaviour. 2016.)

Muiden resurssien kuten kuvien tai 3D-mallien lisääminen aktiiviseen sceneen voidaan toteuttaa samalla tavalla. GameObject luokassa on myös Addcomponent-funktio, jolla erityyppisiä komponentteja voidaan lisätä sceneen sovelluksen jo ollessa käynnissä (Unity Scripting API. 2016). Tällä tavoin sovelluksen toimintaa voidaan automatisoida toisin kuin manuaalisesti Unity Editorissa. Tällöin täytyy kuitenkin muistaa ongelmatilanteiden ehkäisy ohjelmakoodissa, kuten tapauksissa, joissa esimerkiksi tekstuurin tiedosto puuttuu projektitiedostoista.

3 GBUILDER-OHJELMA

GBuilder-ohjelmalla tarkoitetaan kokonaisuutta, johon sisältyy useampi ohjelma tai rajapinta. Ohjelmassa on useita eri komponentteja, jotka tekevät yhteistyötä käyttäjäkokemuksen luomiseksi. GB4D:n kannalta olennainen osa Gbuilder-ohjelmaa on GB Core -rajapinta, joka lähettää esimerkiksi selainympäristössä tehdyn materiaalivalinnan tiedot ja lähettää ne GB4D-ohjelmaan.

3.1 GB4D kiinteistöjen kolmiulotteiseen tarkasteluun

Virtuaalitoteutuksen perustana toimii GB4D-ohjelma, jolla tarkastellaan pohjapiirrustuksia kolmiulotteisena. Group Builderin omalla GBCAD-ohjelmistolla piirretyt pohjapiirustukset viedään palvelimelle, josta ne viedään edelleen eteenpäin sovellukseen. Sovellus käyttää lisäksi BIM-tyyppistä kirjastoa, joka kääntää ifc-tyyppisiä tiedostoja datamuotoon jossa niitä voidaan hyödyntää muissakin sovelluksissa. (Oinas 2016.)

GB4D:n kehitys aloitettiin vuoden 2015 alussa. Kehitys alkoi hitaasti, sillä ohjelman käyttämä BIM-kirjasto oli kirjoitettu uusinta Microsoftin kehittämää .NET Framework kirjaston versiota käyttäen. Unityn käyttämä Mono-kirjasto vastasi vanhempaa .NET Framework versiota, joten valmista BIM-kirjastoa jouduttiin muokkaamaan yhteensopivaksi vanhemman .NET Frameworkin kanssa. Vuoden 2016 tammikuussa GB4D:stä on kuitenkin kehitetty WebGL-käännös, jossa GB Cad -ohjelmistolla piirrettyjä pohjakuvia voidaan tarkastella kaikkine elementteineen. Myös GB Core -ympäristössä tehdyt materiaali- sekä kiintokalustevalinnat ovat nähtävissä GB4D:ssä. (Oinas 2016.)

3.2 BimVault GB4D:n projektipalvelimena

GB4D vastaanottaa datan pohjapiirustuksista JSON-formaatissa olevan datan tekstinä. Koska pohjapiirustukset ovat kuitenkin ifc-muodossa tiedostoina, ne täytyy kääntää pilvipalvelussa haluttuun muotoon, jotta niistä saataisiin JSON-tyyppistä dataa. BimVault on pilvipalvelu, joka suorittaa kyseisen ifc-tyyppisten tiedostojen käsittelyn JSON-tyyppiseksi dataksi.

BimVault itse sijaitsee pilvipalvelussa yhdessä IFC-tiedostojen kanssa, ja reagoi GB4D-sovelluksen rajapintakutsuihin. BimVault kommunikoi GB4D-sovelluksen kanssa siten, että sovelluksen suorituksen alussa lähetetään rajapintakutsu, jonka palautustuotteena BimVault lähettää JSON-muotoista dataa kutsuvalle ohjelmalle.

Dataobjektit on luokiteltu siten että esimerkiksi rakennuksen aliobjekteiksi on loogisesti määritelty asunto ja asunnon aliobjektiksi huone. Yksittäisen projektin data tulee useassa eri osassa. Esimerkiksi yhdellä rajapinta-kutsulla saadaan data rakennuksen kiinteistön seinistä aliobjekteineen ja toisella kutsulla kiinteistöjen huoneiden data aliobjekteineen. Tarvittavan JSON-datan lähettämiseen BimVaultilla menee noin 2–7 sekuntia riippuen rakennuksen sisältämän informaation laajuudesta. BimVault voi tallentaa käännetyn JSON-datan muistiin, jolloin vastaanottavan sovelluksen on nopeampi ladata sitä.

Vastaanotettu JSON-data jäsennetään sovelluksen sisällä luokkaobjekteiksi C#-ohjelmointikielellä ja sitä hyödynnetään kokonaisuutena näytettävän kiinteistön piirtämiseksi. Kaikkea dataa ei kuitenkaan tarvitse hyödyntää piirtämiseen, kuten esimerkiksi yleisessä tapauksessa jossa halutaan esittää vain yksi kiinteistö rakennuskokonaisuudesta.

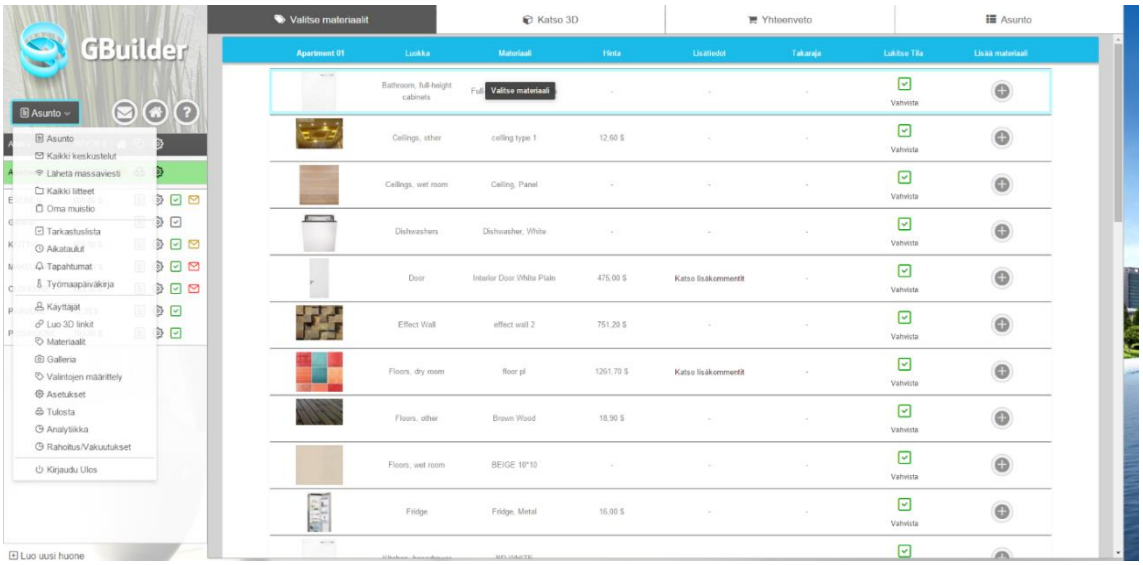
3.3 Rakennuksen tietomallin merkitys GB4D:ssä ja BimVaultissa

BIM eli Building Information Model on suomennettuna rakennuksen tietomalli. Rakennuksen tietomallin määritys on digitaalinen esitys rakennuksen fyysisistä ja toiminnallisista piirteistä. (National Bim Standard. 2016.) GB4D:n kannalta rakennuksen tietomalli käsitteenä kattaa kiinteistön kolmiulotteiseen esittämiseen liian suuren määrän tietoa, joten se data, mitä GB4D:ssä esitetään, käsittää vain rakennuksen esittämisen. BimVault, jota GB4D käyttää, toimii eräänlaisena BIM-muuntajana. BimVault kääntää ifc-muodossa olevat tiedostot GB-ohjelman omiksi luokkaobjekteiksi BIM-periaatteella, jotta lopputuloksena sovelluksille lähetettävä data olisi JSON-muotoista.

3.4 GB4D-sovellus GB Core -ympäristössä

GBuilder-ohjelman pääympäristönä toimii selaimessa näkyvä GB Core -ympäristö, joka pitää sisällään yleisimmät Gbuilder-ohjelmaa käyttävän asiakkaan toiminnot. GB Core on ohjelmoitu Java-kielellä ja esimerkiksi GB4D:n suuntaan se pystyy lähettämään JSON-tyyppistä dataa rajapintakutsun kautta. Vuoden 2016 alussa GB4D käyttää GB Coren lähettämää dataa pääasiassa kalusteiden ja kiintokalusteiden määrittämiseen. Rajapinnan läpi voi lähettää myös muuta hyödyllistä dataa, kuten vaikkapa huoneen tunnisteen, jota sovellus voi hyödyntää kohdistettaessa kameraa huoneeseen.

GB Core -ympäristö tukee tällä hetkellä GB4D:tä vain siten, että GB Coressa suoritetut materiaalivalinnat esitetään GB4D:ssä. Tulevaisuuden tavoitteena kuitenkin on myös, että materiaalivalintoja suoritettaisiin GB4D:n sisäisesti ja materiaalivalinnat tallentuisivat GB4D:stä GB Coreen. (Kuva 3.)



KUVA 3 GB Core-ympäristön materiaalivalintaikkuna

GB Core -ympäristössä tehdyt kalustemuutokset tallentuvat projektikohtaisesti GB Coren tietokantaan. Esimerkkitapauksessa projektiin on tehty kiintokaluste- ja materiaalivalintoja. Kun asuntoprojekti avataan GB Core -ympäristössä GB4D-sovelluksella, GB Core lähettää projektin materiaalivalinnat vastaaviin elementteihin esitettäväksi sovelluksessa. Yhden ifc-tiedoston elementit sisältävät vain yleistä informaatiota JSON-datan muodossa sovellukseen saapuessaan, mutta GB Coresta saatavien valintojen avulla ne voidaan esittää oikeina 3D-objekteinaan. (Kuva 4.)

KUVA 4 GB4D-sovellus CB Core-ympäristössä

3.5 Industry Foundation Class projektitiedostona

Kansainvälisenä oliopohjaisen tiedon formaattina Industry Foundation Classia eli ifc:tä käytetään laajalti rakennusteollisuuden informaatioteknologiassa. ifc:n tarkoitus GB-ohjelmassa on toimia GB Cadilla piirrettyjen pohjapiirustusten perustana. GB Cadilla piirretyt projektit ovat yhteensopivia ifc 2x3 version määrittelyjen kanssa

Ifc-datan sekä informaation lähteenä sisältää GB4D:n käyttöön dataa projektista, rakennuksesta, asunnosta ja sen sisällöstä kokonaisuudessaan. GB Cadissa piirretty kiinteistökokonaisuus viedään ifc-tiedostoksi, jolloin sen sisältämä data voidaan hyödyntää varsinaisessa 3D-sovelluksessa. Erillisiä projektin komponentteja kutsutaan elementeiksi. Esimerkiksi seinä, ikkuna tai kiintokaluste ovat elementtejä. Elementit itsessään voivat sisältää monenlaista dataa, kuten vaikkapa koko-, paikka tai suuntatietoa. Elementtejä ei välttämättä tarvitse piirtää näkyväksi kolmiulotteisena, kuten vaikkapa huonetila, mutta niiden sisältämä data on tarpeellinen, kun halutaan esimerkiksi sijoittaa huonetilan sisältämä informaatio täsmällisesti. (IFC-Based Data Exchange. 2016.)

4 SOVELLUKSEN VIRTUAL REALITY -TOTEUTTAMINEN

Tässä luvussa kuvaillaan virtuaalitoteutuksen työn vaiheita ja tuloksia. Toteutusvaiheessa työ pyrittiin aloittamaan tarvittavien laitteiden taustatutkimuksella, sekä laitteiston valinnalla. Valitsemisen kriteerinä oli, että virtual reality -toteutus GB4D-sovelluksesta olisi mahdollista toteuttaa valitun laitteiston kanssa. Kehityskohteiksi valittiin Oculus Rift -virtuaalilasit ja Google Cardboard -alusta.

Varsinaiseen kehittelyyn ei pyritty käyttämään liikaa aikaa tai resursseja, sillä virtuaalitoteutuksen kehittäminen GB4D-sovellukseen koettiin nopeaksi ja kevyeksi toimenpiteeksi. Toteutus suoritettiin niin, että luoduista ominaisuuksista toteutusvaiheessa valittiin yrityksen sisällä yksimielisesti tärkeimmät ja vaaditut ominaisuudet. Virtual Reality -toteutus ideana oli periaatteessa kokeilu ja yritys ei toistaiseksi tavoitellut tuotantoon menevää valmista toteutusta.

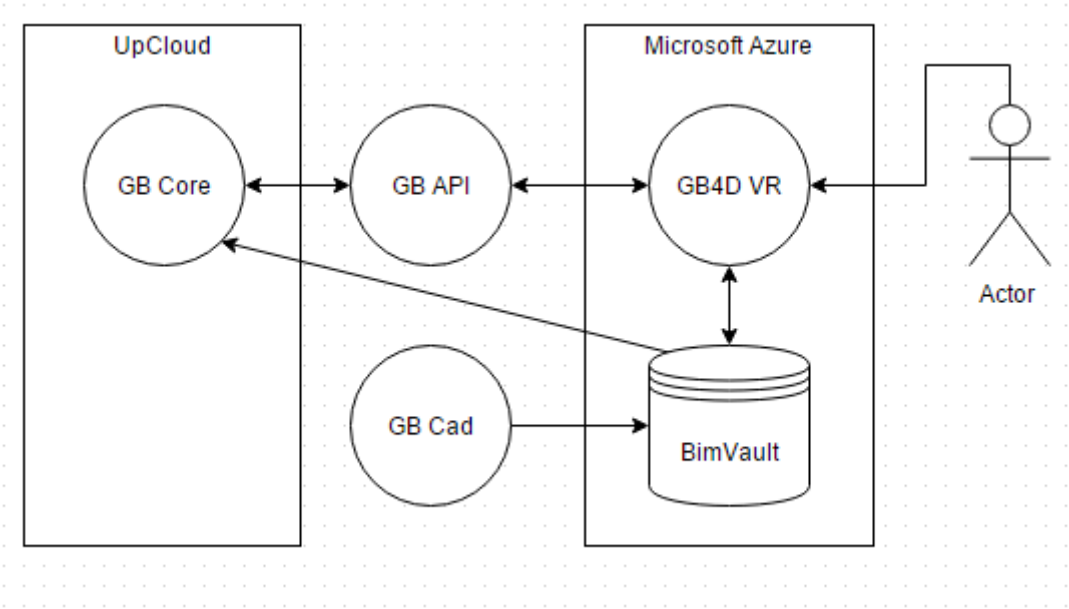
4.1 Virtual Reality rakennusteollisuuden tarpeisiin

Ohjelmistojen tarjoamalla tuella on koettu olevan huomattava merkitys rakennusteollisuudessa suunnittelu- ja esitysvaiheessa. Esimerkiksi teoreettista tietoa rakenteilla olevasta kiinteistöstä voidaan hyödyntää havainnollistamistarkoituksiin ja kiinteistöstä voidaan GB4D:llä luoda kolmiulotteinen malli jo ennen varsinaisen kiinteistön valmistumista. Teoreettiseen informaation perustuva kokonaisuus kiinteistöstä edistäisi sen valmistumista suunnitelmasta myytäväksi asunnoksi. Virtual Reality -toteuttaminen voi viedä ajatusta vielä pidemmälle, ja toteutettu kolmiulotteinen malli voidaan esittää virtuaalisena kiinteistönä virtuaalitodellisuutta havainnollistavien laitteiden avulla, kuten Oculus Rift -laseilla, Google Cardboard -alustalla tai Samsung Gear VR -laseilla.

Ohjelmistotekniikan ja laitteiden kehittyessä jatkuvasti myös GB4D-sovelluksen koettiin tarvitsevan kehittyneempiä esitysalustavaihtoehtoja. Virtuaalitodellisuusteollisuus on saavuttanut suosiota videopelimarkkinoilla, sekä visuaalisella puolella, joten Group Builder Oy totesi virtuaalitodellisuuskokoonpanon hyväksi kolmiulotteisen kiinteistön esitystavaksi markkinointi- ja esittelytarkoituksiin.

Vuonna 2015 Group Builder Oy aloitti virtuaalitodellisuustoteutuksen kehityksen Oculus Rift -virtuaalilaseille. Alustava toteutus, jossa käyttäjän oli mahdollista liikkua BIM-pohjaisen kiinteistön sisällä virtuaalitodellisuudessa, valmistui nopeasti. Oculus Rift -virtuaalilasien kuluttajaversion julkaisu kuitenkin viivästyi ja toteutuksen kehittäminen päätettiin myöhemmin keskeyttää. Vuoden 2015 viimeisellä neljänneksellä mietittiin vaihtoehtoisia virtuaalitodellisuusvaihtoehtoja ja Group Builder alkoi kehittämään virtuaalitodellisuutta myös mobiilialustoilla toimiviin virtuaalilaseihin, kuten Google Cardboard ja Samsung Gear VR.

Virtuaalitodellisuuden toteutus päätettiin suorittaa suoraan olemassa olevan GB4D-sovelluksen päälle siten, että virtuaalitodellisuudessa näkyvä malli näkyisi samanlaisena, kuin se näkyy normaalissa PC:llä suoritettavassa sovelluksessa, mutta vain poikkeuksellisesti virtuaalisen kävelevän hahmon näkökulmasta. Toteutukseen liittyvä idea saatiin tällöin pidettyä mahdollisimman yksinkertaisena, eikä GBuilder-ohjelman toimijakomponenttien hierarkiaan tarvittu tehdä muutoksia, kuin itse GB4D-sovelluksen virtuaalitodellisuustoteutukseen. Unityn kääntämät WebGL-sovellukset eivät tukeneet virtuaalitodellisuustoteutusta, eikä Group Builder Oy:n suunnitelma sitä vaatinut, joten mobiilialustojen virtuaalitodellisuustoteutukset päätettiin kääntää ensisijaisesti Android-alustalle, ja Oculus Rift -virtuaalilaseilla Windows-alustalle. (Kuva 5.)

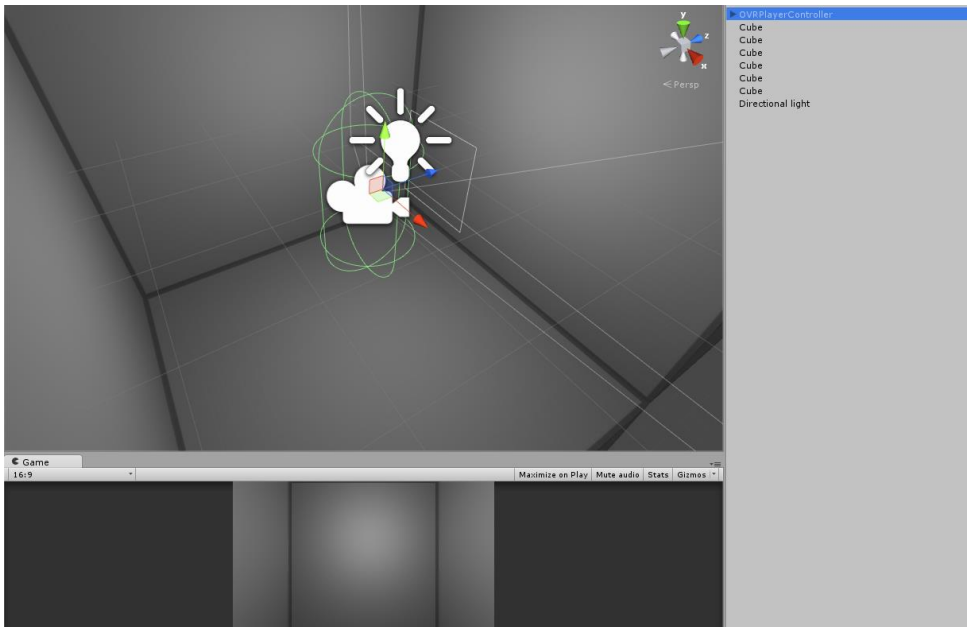


KUVA 5 Yksinkertaistettu komponenttikaavio Gbuilder-ohjelmasta

4.2 Oculus Rift-toteutus GB4D-ohjelmaan

Oculus VR LLC on kehittänyt Oculus Rift -virtuaalilaseille sovelluskehittäjän ohjelmakoodipaketin Unityn kehitysympäristöä varten. Pakettiin kuuluu esimerkiksi yksinkertaiset script-tiedostot virtuaalilasien fysiikkaa ja ohjausominaisuuksia varten. Pakettia ja sen esimerkkejä voidaan hyödyntää ohjelmien kirjoituksessa ja esimerkiksi first person controller -nimistä scriptiä voidaan hyödyntää GB4D:ssä, kun tehdään kameratilaa, jossa käyttäjä voi liikkua esimerkiksi kiinteistön sisällä virtuaalisesti. (Oculus VR LLC. 2016.)

Työ aloitettiin kehityspaketin lataamisella ja sen sisällön tutkimisella. Kehityspaketti sisälsi scriptit lasien kontrollointiin, esimerkin scenestä, joissa demonstroidaan lasien käyttöä Unityn sisällä, sekä valmiit esimerkkielementit, joilla käyttäjä voidaan sijoittaa virtuaalitilan sisälle ohjaamaan kuviteltua hahmoa. Käytännössä yksinkertaisen ohjausesimerkin saa aikaiseksi vetämällä pelaajahahmon prefabin Unityn scene-ikkunaan. Kun ohjelma suoritetaan, lasit reagoivat pään liikkeisiin ja hahmon liikkumista voidaan jo kontrolloida tietokoneen näppäimistöllä ja hiirellä valmiin ohjelmakoodin, sekä esiasetettujen gameobjectien eli prefabien avulla. (Kuva 6.)



KUVA 6 Pelaajahahmo 3D-tilassa Virtual Reality -toteutuksella

Virtuaalitodellisuuden soveltaminen toteutettiin ensisijaisesti siten, että valmiista kontrolloitavasta prefabista luodaan ilmentymä suoritettavaan sceneen silloin, kun kiinteistö on ladattu ja piirretty kolmiulotteiseen tilaan. BimVaultista tulevan datan avulla voidaan laskea ohjelmakoodissa esimerkiksi kiinteistön suurin huone ja sijoittaa prefab siihen. Tällöin käyttäjä pääsee suoraan kiinteistön sisään liikkumaan virtuaalitilaan. (Kuva 7.)

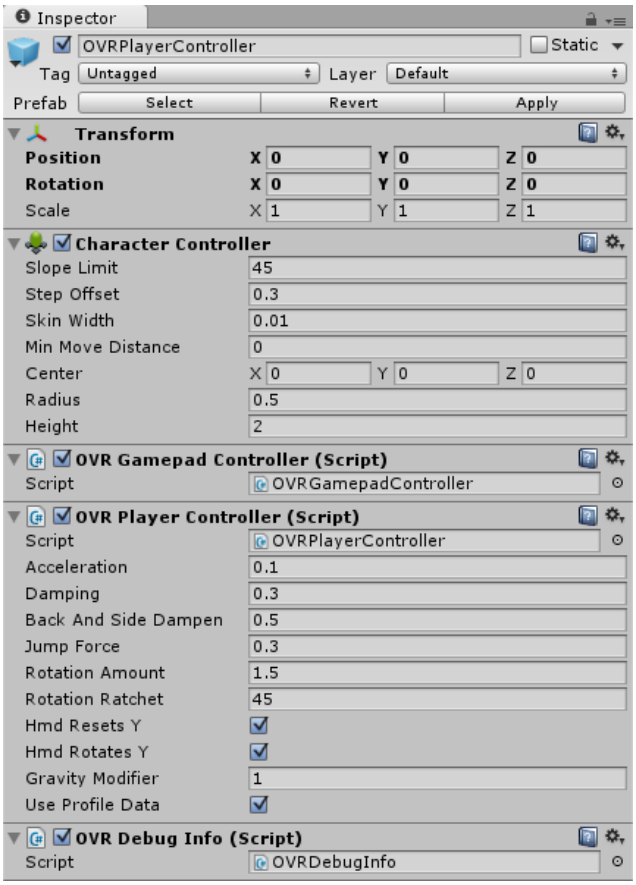
```
var vr = GameObject.Instantiate (Resources.Load<GameObject> ("OVRPlayerController"));
vr.transform.position = SelectedProject.ChildViewModels.Instances.
OfType<SpaceViewModel> ().FirstOrDefault ().GetLargestSpace;
```

KUVA 7 Ohjelmakoodi pelaajahahmon ilmentymän luomisesta Instantiate-funktiolla

Ohjelmakoodissa luodaan instanssi gameobjectista Instantiate-nimisellä funktiolla, joka sisältyy gameobject-luokkaan. Instantiate vaatii parametrikseen viittauksen olemassa olevasta gameobjectista, jolloin parametriksi syötetään resources-luokasta Load-funktio. Load-funktio vaatii parametrikseen string-tyyppisen muuttujan, joka on pelaajahahmon prefabin nimi. Ohjelmakoodin rivin suorituksen jälkeen GameObject on luotu ja sen sijainti kolmiulotteisessa tilassa voidaan asettaa GetLargestSpace-nimisellä funktiolla, joka palauttaa gameobjectin sijainniksi kiinteistön suurimman huoneen sijainnin. (Unity Scripting API. 2016.)

4.2.1 Kävelijähahmon kontrollien kehittäminen

Oculus-kehityspakettiin kuuluu valmiiksi luotuja elementtejä, joita voidaan sijoittaa pelitilaan. Esimerkiksi GB4D-toteutuksessa käytettiin elementtiä, jossa näkökulma sijoittuu henkilön pään tasolle mahdollisimman realistista vaikutelmaa varten. Elementissä on kiinni C#-scriptejä komponentteina, jotka suorittavat virtuaalilasien kontrolloinnin. Scripteissä on valmiina julkisia muuttuja-arvoja, joita muuttamalla voidaan säätää sitä, miten virtuaalilasit reagoivat kontrollointiin. Muuttujat joita säädettiin, olivat kävelynopeus, kameran kääntymisen nopeus, sekä hahmon juoksemisen nopeus. Yhdessä prefabin scripteistä oli myös valmiina hyppyominaisuus, joka päätettiin kommentoida pois ohjelmakoodista, sillä se koettiin turhaksi kiinteistön tarkastelua varten. (Kuva 8.)



KUVA 8 Lista pelaajahahmon GameObjectin komponenteista Unity Editor-ikkunassa

4.2.2 Toteutuksen käyttöliittymä

Koska GB4D-sovellus vaatii käyttöliittymän tiettyjä ominaisuuksia varten, kuten vaikkapa kameratilan vaihtaminen tai seinätapettien piilottaminen, tarvitsi myös virtuaalitoteutus oman käyttöliittymänsä. Virtuaalitoteutuksen valmis kävelijähahmo-prefab ei tue käyttöliittymää, joka sijaitsee sovelluksen kameratilassa, joten käyttöliittymä täytyi sijoittaa pelin kolmiulotteisen tilan sisään.

Käyttöliittymän sijoittaminen kolmiulotteiseen tilaan toi uusia haasteita toteuttamiseen, sillä käyttöliittymän tuli olla kameran ulottuvissa ja käytettävissä, mutta samalla myös hyödyllinen niin, ettei se haittaisi kameranäkymää. Käyttöliittymä päätettiin toteuttaa siten, että se pystyttäisiin piilottamaan ja palauttamaan käyttäjän edestä napin painalluksella gameobjectin SetActive-nimisellä funktiolla,

joka vastaanottaa parametrikseen bool-tyyppisen muuttujan, kun käyttöliittymä halutaan joko piilottaa tai esittää.

Käyttöliittymä oli canvas eli piirtoalue-tyyppinen gameobject, joten sen sijaintia ja käyttäytymistä pystyttiin ohjelmoimaan, kuten mitä tahansa gameobjectia. Käyttöliittymän sijainnin siirtämisessä päätettiin hyödyntää monobehaviourin Update-nimistä funktiota, jota sovellus kutsuu jokaisella kuvanpäivityksellä. Update-funktiossa napin painalluksia voitiin tarkastella kevyesti, sillä sovelluksen optimoinnin kannalta ne eivät käytä liian suurta määrää prosessorin resursseja. (Kuva 9.)

```
void Update ()
{
        if (Input.GetKeyDown("C"))
        {
                GameObject.Find("Canvas").SetActive(!boolToggleCanvas);
                GameObject.Find("Canvas").transform.position = ovrPlayerController.transform.
                position + ovrPlayerController.transform.forward
        }
}
```

KUVA 9 Esimerkki Canvas-nimisen käyttöliittymän uudelleen piilotuksesta ja sijoituksesta

4.3 Google Cardboard-toteutus GB4D-ohjelmaan

GBuilder-ohjelma tarvitsi virtual reality -toteuttamisen kehitysalustoiksi myös muita vaihtoehtoja, joten toiseksi kehityskohteeksi otettiin Googlen kehittämä Google Cardboard -alusta. Google Cardboard oli virtuaalitodellisuusalustana edullinen sekä helposti kehitettävä, joten yritys näki siinä enemmän kiinteistöjen visuaaliseen esittämiseen liittyvää potentiaalia kuin Oculus Rift -laseissa. Alustaan sisältyi tuki Unity-ympäristölle sekä Cardboard-sovelluskehityspaketti, jonka avulla Google Cardboard -toteutuksesta GB4D-ohjelmaan pystyttiin myös nopeasti luomaan varhainen käytettävä versio. (Kuva 10.)

KUVA 10 Varhainen Google Cardboard -toteutus GB4D-ohjelmasta

4.3.1 Unity SDK toteutuksen kehittämiseen

Cardboard toteutuksen avuksi otettiin käyttöön valmis Unity SDK-niminen sovelluskehityspaketti, johon sisältyi toteutuksessa hyödynnettäviä scriptejä, kuten Cardboard- ja CardboardHead-tiedosto. Toteutusta varten käytettiin esimerkiksi C#-ohjelmointikieleen perustuvia eventejä, joita sovellus kutsuu Cardboard-alustan tietyillä komennoilla. Esimerkiksi OnTrigger-nimistä eventiä sovellus kutsuu joka kerta, kun Google Cardboard-alustan magneettikytkintä painetaan. OnTrigger-eventiin oli mahdollista merkitä kaikki funktiot, joita eventin tapahtuessa kutsuttaisiin. (Plugin References. 2016.)

4.3.2 Google Cardboard-toteutuksen kontrollien kehittäminen

Cardboard-alustassa hyödyllisimmäksi event-kutsuksi todettiin OnTrigger-event, jota kutsumalla virtuaalista hahmoa päätettiin liikuttaa eteenpäin muutamia as-

keleita. Funktion merkitseminen suoritettiin MonoBehaviour-luokan Start-funktiossa, joka suoritetaan sovelluksen käynnistyksen yhteydessä. Merkitsemisen jälkeen jokainen merkitty funktio kutsuttiin eventin tapahtuessa. Kutsuttavassa funktiossa täytyi ottaa huomioon GB4D-sovelluksen ympäristö, sillä magneettikytkintä painamalla päätettiin tarkistaa, mitä objekteja Cardboard-toteutuksen kameran edessä oli. Objekti kameran edessä saattoi olla esimerkiksi ovi, seinä, ikkuna tai käyttöliittymä elementti, jolloin tarvittiin liikkumisesta poiketen toisenlaista käyttäytymistä. (Kuva 11.)

Esimerkkifunktiossa kerättiin Unityn EventSystem-luokan RaycastAll-funktiolla listaan kaikki objektit, jotka olisivat kameran edessä ruututilan keskipisteessä. Listan sisällöstä pystyttiin päättelemään, mitä käyttäytymistä funktion tuli noudattaa. Jos kameran edessä oli ovi, kutsuttiin funktiota joka aukaisee oven tai jos kameran edessä oli esimerkiksi käyttöliittymän painike, funktion suoritus lopetettiin ja Unityn eventsystem suoritti painikkeen painalluksen automaattisesti. Jollei estäviä tekijöitä ollut, pystyttiin kutsumaan funktiota joka suoritti virtuaalihahmon liikuttamisen. (Kuva 11.)

```
void Start ()
{
	Cardboard.SDK.OnTrigger += () => {CheckCardboardTrigger();};
}


public void CheckCardboardTrigger()
{
	//UI raycasts
	var pointer = new PointerEventData(EventSystem.current);
	pointer.position = new Vector2(Screen.width/2, Screen.height/2);
	List<RaycastResult> raycastResults = new List<RaycastResult>();
	EventSystem.current.RaycastAll(pointer, raycastResults);

	// Rotateable raycasts
	var bimWrapperList = Physics.RaycastAll(cardBoardMain.transform.position,
	cardBoardMain.transform.FindChild("Head").forward, 2.5f).Where(p => p.collider.gameObject.GetComponent<DoorRotate>()).ToList();

	if (raycastResults.Exists(p => p.gameObject.layer == 5)) // If hits UI, return
		return;
	else if (bimWrapperList.Count() > 0){ // Else if theres doors or windows, trigger and return
		foreach ( var door in bimWrapperList){
			if(door.collider.gameObject.GetComponent<DoorRotate>() &&
			door.collider.gameObject.GetComponent<DoorRotate>().coroutinesRunning == false)
				StartCoroutine(door.collider.gameObject.GetComponent<DoorRotate>().TriggerDoor());
		}
	return;
	}
	else // If nothing else, just move
		StartCoroutine(CardBoardMove());
}
```

Kuva 11 Esimerkki funktion merkitsemisestä OnTrigger-eventiin

Sovelluskohtaisten toteutusten lisäksi Unity SDK:n mukana oli valmiina kamera-prefab, johon sisällytetyt scriptit reagoivat puhelimen kiihtyvyysanturin liikkeisiin. Kiihtyvyysanturin informaatio siirrettiin sovellukseen, joka liikutti kameraa puhelimen liikkeiden mukaisesti. Näitä ominaisuuksia hyödynnettiin kameran osoittamisen kanssa siten, että kolmiulotteisessa tilassa pystyttiin liikkumaan kameran osoittamaan suuntaan.

4.4 Virtuaalitoteutusten jatkokehittäminen ja ideat

GB4D:n virtuaalisia toteutustapoja kehitellessä löydettiin useita vaihtoehtoisia kehityskohteita Google Cardboardin ja Oculus Riftin lisäksi kuten Microsoft Hololens, Samsung Gear VR ja HTC Vive. Toteutusmahdollisuuksia Group Builderissa pohdittiin lisää ja toteuttamisen mahdollistamisen ratkaisevina kriteereinä pidettiin alustojen suosiota kaupallisilla markkinoilla, edullisuutta sekä alustan laitteiston tehoa GB4D:tä varten.

Yleiseksi linjaukseksi virtuaalitodellisuuslaitteissa todettiin, että mobiilialustoilla suoritettavat virtuaalisovellustoteutukset olisivat hyödyllisimpiä, sillä kiinteistöjen tarkastelun tulisi toimia mahdollisimman kevyesti ja joustavasti. Myös tapa, millä virtuaalinen representaatio saataisiin asiakkaan ulottuviin, koettiin ratkaisevaksi tekijäksi. Kalliiden sekä monimutkaisten virtuaalisovellusalustoiden sovelluskehittäminen, sekä saattaminen asiakkaan käyttöön monimutkaistaisivat liikaa varsinaista käytettävyyttä ja hyödyllisyyttä.

Myöhemmin yhdeksi jatkokehityskohteeksi Group Builder valitsi myös Samsung Gear VR-virtuaalisovellusalustan mobiililaitteille, edistyneemmäksi kehityskohteeksi Google Cardboardin seuraajaksi. Samsung Gear VR:ssä koettiin hyödylliseksi virtuaalisen esittämisen tehostuminen virtuaalilasikehyksissä, sekä monipuolisemmat kontrollointivaihtoehdot. Google Cardboardiin verrattuna Samsung Gear VR:ssä oli yhden kytkimen sijaan useita eri vaihtoehtoja käyttää virtuaalilasikehyksissä olevaa kosketusalustaa, jonka ansiosta niille olisi mahdollista kehittää useampia eri käskyjä. (Interaction in VR. 2016; VR Overview. 2016.)

Samsung Gear VR:n toteutus sovellukseen tapahtui lähestulkoon samalla tavalla, kuin Google Cardboard -toteutus. Unityn dokumentaatiosta kävi ilmi että Unityssä oli sisäänrakennettu tuki tietyille virtual Reality -laitteille joihin Samsung Gear VR kuului. Projektiin ladattiin Oculus LCC:n sivuilta saatava sovelluskehittäjän-paketti jolla laitteiston toiminnot ohjelmoitiin sovelluksen sisään.

Virtual Reality -toteutusten toteuttamista monimutkaisti sovellusten graafinen raskaus. Oculus Rift -lasit vaativat tietokoneen näytönohjaimesta paikan HDMI-kaapelille, mikä vaikeutti varsinaisen laitteiston asennusta varsinkin, jos kyseessä oli kannettava tietokone. Lasit vaativat näytönohjaimesta oletusarvoisesti paljon suoritustehoa (Powering the Rift. 2016) jolloin katsottiin parhaaksi siirtää Oculus Rift -lasien virtuaalitoteuttaminen myöhemmäksi. Nykyaikaisilla mobiililaitteilla suoritettavat virtuaalitodellisuustoteutukset koettiin graafisesti paremmaksi Group Builder Oy:ssä joten niiden kehittämiseen päätettiin panostaa enemmän.

5 YHTEENVETO

Opinnäytetyön aiheena oli kokeellisesti soveltaa ja luoda virtuaalitodellisuustoteutus GB4D-sovelluksesta. Virtuaalitodellisuuden soveltaminen rakennusteollisuuden sovelluksessa oli haastavaa, mutta esimerkiksi Oculus Rift -laseilla ja Google Cardboard -alustalla onnistuttiin luomaan toteutukset, jossa käyttäjä pystyi oma-aloitteisesti liikkumaan virtuaalisessa kiinteistössä. Suurin haaste nykypäivän virtuaalilaseissa on laitteiston tehokkuus, jota tarvitaan, kun kuvaa pitää piirtää nopeasti käyttäjän katseltavaksi samanaikaisesti, kun liikutaan kolmiulotteisessa tilassa. Nähtäväksi jää, kuinka tärkeään asemaan virtuaalitodellisuuden soveltaminen tuotteena Group Builderissa jää.

Virtuaalitodellisuusalustojen ohjauksen ohjelmoiminen oli suhteellisen helppoa sovelluskehittäjien pakettien mukana tulevan sisällön avulla. Kehittäjän tehtäväksi jää ainoastaan tutkia arvoja, joita virtuaalilasien komponentit tuottavat, sekä soveltaa virtuaalitodellisuuden toteutus omaan sovellukseen. Käyttäjäkokemuksen parantamiseksi tuli kiinnittää erityistä huomiota siihen, miten käyttäjät keskimäärin toivoisivat virtuaalitilan kontrolloitavaksi. Tämän takia haluttiin kiinnittää erityistä huomiota siihen, mikä olisi käyttäjäystävällisin tapa ohjata virtuaalista hahmoa etenkin mobiilialustoilla, joissa ohjausvaihtoehdot olivat erittäin rajalliset.

Virtuaalisovelluksille jää vielä paljon kehitysvaraa, sillä VR-teollisuus tulee todennäköisesti kehittymään tulevaisuudessa ja varsinaista käyttäjäkokemusta voidaan parantaa sitä mukaan, kun palautetta käyttäjäkokemuksista saadaan. Virtuaalinen representaatio on hyvä, muttei välttämätön lisä rakennusteollisuuden visualisointisovelluksessa.

LÄHTEET

Company Facts. 2015. Unity Technologies. Saatavissa: http://unity3D.com/public-relations. Hakupäivä 7.11.2015.

Cook, Dave 2015. Unity interview: engineering democracy. Saatavissa: http://www.vg247.com/2012/10/18/unity-interview-engineering-democracy/. Hakupäivä 1.2.2016

Takahashi, Dean 2014. VentureBeat. Saatavissa: http://venturebeat.com/2014/10/23/john-riccitiello-sets-out-to-identify-the-engine-of-growth-for-unity-technologies-interview/. Hakupäivä 8.11.2015

Oinas, Ari 2016. Ohjelmistokehittäjä, Group Builder Oy. Pikaviestintäkeskustelu 14.3.2016

IFC-Based Data Exchange. 2016 FAQ. Saatavissa: https://www.graphisoft.com/downloads/addons/ifc/FAQ/index.html. Hakupäivä: 11.2.2016

Interaction in VR. 2016. Unity. Saatavissa: https://unity3d.com/learn/tutorials/topics/virtual-reality/interaction-vr. Hakupäivä: 29.3.2016

Mono FAQ General. 2016. Mono. Saatavissa: http://www.mono-project.com/docs/faq/general/. Hakupäivä 1.2.2016

MonoDevelop. 2016. MonoDevelop. Saatavissa: http://www.monodevelop.com/. Hakupäivä 1.2.2016

National Bim Standard. 2016. FREQUENTLY ASKED QUESTIONS ABOUT THE NATIONAL BIM STANDARD-UNITED STATES. Saatavissa: https://www.nationalbimstandard.org/faqs. Hakupäivä 14.3.2016

Oculus VR LLC. 2016. Documentation. Saatavissa: https://developer.oculus.com/documentation/. Hakupäivä 5.2.2016

Plugin References. 2016. Google. Saatavissa: https://developers.google.com/cardboard/unity/reference#package_contents. Hakupäivä: 26.3.2016

Powering the Rift. 2016. Oculus VR, LCC. Saatavissa: https://www.oculus.com/en-us/blog/powering-the-rift/. Hakupäivä: 4.6.2016

Unity MonoBehaviour. 2016. Unity. Saatavissa: http://docs.unity3d.com/ScriptReference/MonoBehaviour.html. Hakupäivä: 15.3.2016

Unity Scripting API. 2016. Unity. Saatavissa: http://docs.unity3d.com/ScriptReference/. Hakupäivä: 30.3.2016

Unity. 2016. Made with Unity. Saatavissa: https://unity3d.com/showcase/gallery. Hakupäivä 1.2.2016

VR Overview. 2016. Unity. Saatavissa: https://unity3d.com/learn/tutorials/topics/virtual-reality/vr-overview. Hakupäivä 4.4.2016