LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

VIRTUAALINEN NÄYTTELY

LAHDEN AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikka
Ohjelmistotekniikka
Kevät 2019
Juha Luoma

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Luoma, Juha	Opinnäytetyö, AMK	Kevät 2019
	Sivumäärä	
	30	

Työn nimi

Virtuaalinen näyttely

Tutkinto

Tieto- ja viestintätekniikan insinööri

Tiivistelmä

Opinnäytetyössä käsitellään virtuaaliseen todellisuuteen perustuvaa ohjelmistoa, joka on valmistettu Unity-pelimoottorilla sekä Oculus Riftin ohjelmointirajapinnalla. Lahden historialliselle museolle toteutettu ohjelmisto on mahdollista viedä ihmisten luokse näytteille. Ohjelmistoon toteutettiin 100 askelta Lahdessa -näyttelystä oma virtuaalinen versio ja ohjelmiston näyttelyjä on mahdollista muokata tarvittaessa.

Lahden museoiden kiinnostus virtuaalisia näyttelyjä kohtaan on kasvanut. Virtuaalisen todellisuuden teknologian soveltaminen tuo lisää mahdollisuuksia näyttelyiden esittämiseen.

Opinnäytetyössä käsitellään virtuaalista todellisuutta käsitteenä ja sitä, miten Unityssa pystytään käyttämään Oculus Riftin ohjelmointirajapintaa. Samalla tutustutaan tapoihin, miten näyttelyssä esiintyviä kuvia, ääniä sekä videoita pystytään tuomaan ohjelmistoon näytettäväksi. Samalla tutustutaan tapaan, jolla ohjelmistosta pystytään toteuttamaan interaktiivinen ilman ohjaimia.

Opinnäytetyötä oli tekemässä Lahden historiallisen museon henkilökuntaa ja Lahden ammattikorkeakoulusta yksi ohjelmoija. Ohjelmoijan tehtävänä oli toteuttaa ohjelmistoon koodit ja asetella elementit näkymissä. Museon henkilökunta huolehti näyttelyn käsikirjoituksesta ja tarvittavista materiaaleista.

Asiasanat

Unity, Oculus Rift, virtuaalinen todellisuus

Abstract

Author(s)	Type of publication	Published
Luoma, Juha	Bachelor's thesis	Autumn 2019
	Number of pages	
	30	

Title of publication

Virtual exhibition

Name of Degree

Bachelor of Engineering, Information and Communications Technology

Abstract

The objective of the thesis was to make a virtual reality program with Unity game engine and Oculus Rift's software development kit. The program developed for Lahti Historical Museum was intended to be a virtual reality exhibition for people who has no access to the original exhibitions. The program was made about an exhibition called "A Hundred Steps in Lahti" and the virtual exhibition can be modified when needed.

The Historical Museum of Lahti has an increasing interest in virtual exhibitions. The adaptation of virtual reality technology brings more possibilities to present historical events to people.

The thesis deals with the concept of virtual reality and how virtual reality can be developed in the Unity game engine. Topics also include the ways how pictures, audios and videos can be displayed inside virtual reality. The thesis also introduces a way of making a program interactive without using any controllers.

The team consisted of one programmer from Lahti University of Applied Sciences and personnel from Lahti Historical Museum. The programmer's job was to create code for the program and position the elements in the scenes. Personnel from Lahti Historical Museum took care of the scripts and provided the materials included in the exhibition.

Keywords

Unity, Oculus Rift, virtual reality

SISÄLLYS

1 JOHDANTO

Oculus Riftin virtuaalilaitteiston myötä virtuaalisen todellisuuden pelit alkoivat lisääntymään. Virtuaalisen todellisuuden ansiosta pelaajat pääsevät tutustumaan realistisempiin peleihin. Myöskin muualla kuin pelimaailmassa on herännyt kiinnostusta virtuaalista todellisuutta kohtaan. Sitä on alettu käyttämään esimerkiksi myös markkinoinnissa, museoissa ja opetuksessa.

Lahden kaupunginmuseon alaisuudessa toimii Lahden historiallinen museo, joka on Päijät-Hämeen maakuntamuseo. Siellä järjestetään näyttelyitä, joiden teemat vaihtuvat tiettyjen ajanjaksojen välin. Yhdestä näistä näyttelyistä lähdettiin toteuttamaan virtuaaliseen todellisuuteen pohjautuvaa näyttelyä, jota pystytään tarpeen mukaan viemään asiakkaiden luokse. Laitteistoksi kyseiselle ohjelmistolle valittiin Oculus Rift, koska Lahden historiallinen museolla oli Oculus Riftejä jo kolme kappaletta.

Aluksi opinnäytetyössä käsitellään virtuaalista todellisuutta ja sen määritelmää. Sitten käydään läpi virtuaalisen todellisuuden historiaa, minkä myötä nykyaikainen virtuaalinen todellisuus on muodostunut. Samalla käydään läpi Oculus Riftin syntyminen ja sen mukana tulevaa laitteistoa.

Virtuaalisen todellisuuden historian jälkeen käydään läpi Unity-pelimoottoria ja siitä löytyviä kirjastoja, joita käytettiin tämän ohjelmiston toteuttamisessa. Sen lisäksi käydään läpi Unity-pelimoottorin ohjelmistoarkkitehtuuria ja siihen kuuluvia asioita, joiden avulla pystytään ohjelmistoja toteuttamaan. Tämän jälkeen tutustutaan siihen, miten Oculus Riftiä pystytään käyttämään Unityssa ja millaisia komponentteja Oculus Riftin oma ohjelmointirajapinta tarjoaa.

Seuraavaksi opinnäytetyössä käydään läpi sovelluksen valmistusprosessi. Tämän aikana tutustutaan siihen, miten sovellus on rakennettu hyödyntäen Unityn pelimoottorin kirjastoja. Valmistusprosessiin kuului sovelluksen suunnittelua ja ohjelmiston valmistaminen vaadittujen kriteerien mukaisesti. Sovelluksen mukana toteutettiin museon käsikirjoittama esimerkki, jonka pohjalta sovellukseen tarvittavat komponentit valmistettiin.

Lopuksi opinnäytetyössä mietitään sovelluksen onnistumista ja sen mukana esiin tulleita ajatuksia. Tämän lisäksi pohditaan myös mahdollisia muita jatkokehitysideoita ja sitä, miten virtuaalista todellisuutta voitaisiin hyödyntää paremmin.

2 VIRTUAALISEN NÄYTTELYN KÄYTTÖTARKOITUS

Museot järjestävät Lahdessa useita näyttelyitä, joita ihmiset käyvät katsomassa paikan päällä. Näyttelyitä kierretään kävellen ja katsotaan esillä olevia historiallisia asioita, jotka soveltuvat museon näyttelyn teemaan. Nämä näyttelyt ovat museoiden tärkein asia, ja niiden avulla museot kohtaavat yleisönsä. Joidenkin ihmisten on kuitenkin vaikeampaa käydä museoissa vierailulla iän tai fyysisen vamman takia. Jotkut ihmiset taas saattavat asua alueella, jossa ei ole lähellä museota.

Ihmisten rajoittunutta liikkumista varten museolla on ollut kiinnostusta tuoda näyttelyitä ja elämyksiä liikuntarajoitteisille ihmisille virtuaalisessa muodossa. Tämänkaltaisten näyttelyiden toteutuksesta on Lahden historiallisella museolla kokemusta jo aikaisemmin 100 askelta Lahdessa -näyttelystä, jonne toteutettiin virtuaalinen elämysmatka Vesijärvenkadulla. Tämä oli kuitenkin prototyyppi ja toiminnallisuudet eivät toteutuneet halutulla tavalla, jolloin syntyi tarvetta toteuttaa kevyempi virtuaalinen näyttely varsinaisesta 100 askelta Lahdessa -näyttelystä.

100 askelta Lahdessa oli Suomen 100-vuotis juhlan kunniaksi järjestetty näyttely, jossa käytiin läpi Lahden historiaa sadan vuoden ajalta. 100 askelta Lahdessa sijoittui pääasiallisesti Vesijärvenkadulle, josta näyttelyn matka aloitettiin rautatieasemalta ja se päättyi Lahden satamaan. Matkan varrella esitettiin Lahden ikimuistoisempia viihdepaikkoja, kauppoja ja niiden kehittymistä, yleisten saunojen kulttuuria sekä Lahden teollisuutta.

Virtuaalisessa näyttelyssä oli museon väen asettamia kuvia, videoita, ääniä ja tekstejä. Lahden historiallisen museon henkilökunnalla oli tavoitteena viedä näyttelyä mukanaan muistisairaiden ihmisten hoivapaikkoihin ja kouluihin. Siinä haluttiin esittää 100 askelta Lahdessa -näyttelyn eri huoneet niin kuin ne olivat paikan päällä. Näitä huoneita oli yhteensä kahdeksan kappaletta. Ohjelmistoon uudelleenkäytettävyyden takia siihen haluttiin myös mahdollisuus vaihtaa näyttelyn sisältöä tarpeen mukaan. Joidenkin näytettävien kuvien kiinnostavuus saattaa olla toiselle asiakasryhmälle eri kuin toiselle.

3 VIRTUAALINEN TODELLISUUS

3.1 Määritelmä

Virtuaalisella todellisuudella tarkoitetaan todellisuutta, joka toteutetaan keinotekoisessa 3D-ympäristössä. Virtuaalisen todellisuuden toteuttamiseen käytetään interaktiivisia laitteistoja, joilla saadaan luotua käyttäjälle melko todentuntuinen kokemus korvaamaan aito todellisuus. Samalla käyttäjä pystyy vuorovaikuttamaan virtuaalisessa todellisuudessa lähes todentuntuisesti laitteistojen rajoitusten mukaisesti. Kuviossa 1 on havainnoituna kaksi pelaajaa pelaamassa virtuaalisen todellisuuden peliä lisälaitteiden avulla. (Diversified Internet Holdings LLC 2019.)

KUVIO 1. Virtuaalinen todellisuus (Garland 2016)

On olemassa myös lisättyä todellisuutta ja sekoitettua todellisuutta. Virtuaalinen todellisuus eroaa näistä siten, että se on erotettu kokonaan aidosta todellisuudesta. Lisätty todellisuus on taas muoto, jossa virtuaalista sisältöä lisätään aitoon todellisuuteen. Sitä pystytään toteuttamaan ilman erikseen tarvittavia laseja, esimerkiksi omalla älypuhelimella. Tunnetuin esimerkki lisätystä todellisuudesta on Pokemon GO, jossa etsitään virtuaalisia

hahmoja aidosta maailmasta. Seuraavassa kuviossa 2 on esitetty pelaaja etsimässä virtuaalista olentoa sillan luota. Sekoitettu todellisuus on taas lisätyn todellisuuden kehittyneempi muoto. Siinä lisätään virtuaalista sisältöä myös aidon todellisuuden päälle, mutta tämän virtuaalisen sisällön kanssa pystytään vuorovaikuttamaan. (Tokareva 2018.)

KUVIO 2. Lisätty todellisuus (McCarthy 2017)

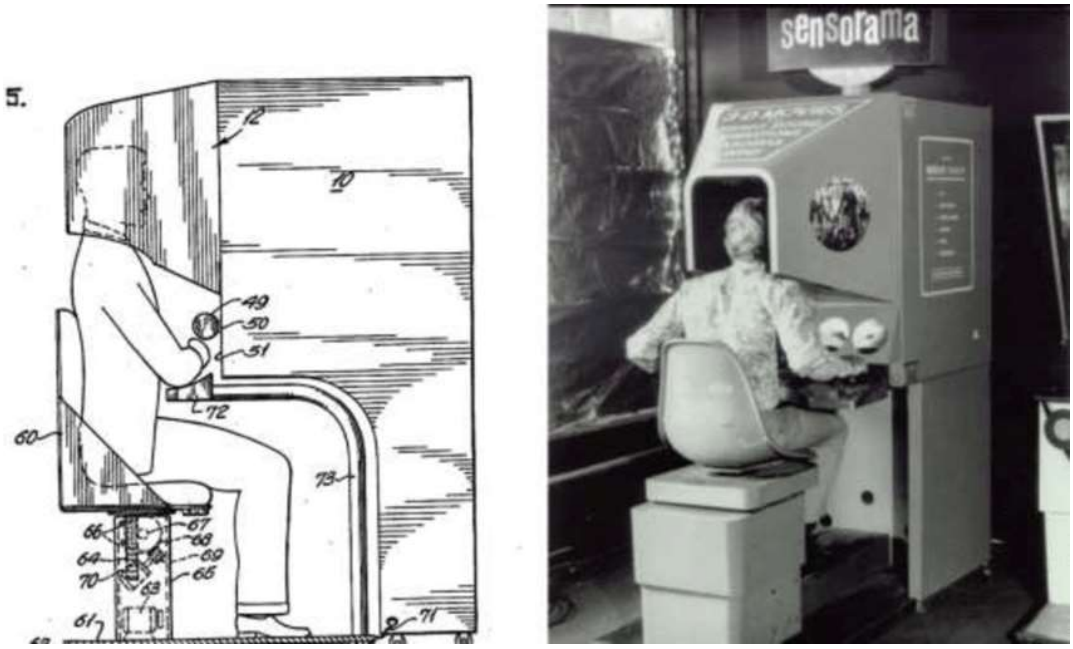
3.2 Historia

Virtuaalisen todellisuuden ajatus sijoittuu kauas historiaan, kun alettiin maalaamaan huoneen ympäröiviä maalauksia. Maalausten tarkoituksena oli tuoda katsojalle todentuntuinen kokemus siitä, että he olivat itse mukana maalauksen tapahtumissa, esimerkiksi taistelussa tai ympäristössä. (Virtual Reality Society 2017.)

Vuonna 1838 Charles Wheatstonen tekemässä tutkimuksessa osoitettiin, että aivot sulauttivat eri silmillä nähdyt kuvat yhdeksi kuvaksi, jolloin pystyttiin luomaan pienimuotoista immersiota ja keinotekoista todellisuutta. Nykyään samaa immersion luontia käytetään halvoissa virtuaalisissa laitteissa, esimerkiksi älypuhelimiin liitettävissä matalan hintatason virtuaalilaitteistoissa. (Virtual Reality Society 2017.)

Vuonna 1930 esiintyi ensimmäinen ajatus nykyaikaisista virtuaalilaseista, kun Stanley G. Weinbaum kirjoitti kirjassaan ajatuksen niistä. Näiden lasien avulla henkilö pystyi kokemaan keinotekoista maailmaa hologrammien, hajun, maun sekä tunnon avulla. (Virtual Reality Society 2017.)

1950-luvulla rakennettiin ensimmäinen virtuaalinen todellisuuden laite: Sensorama, jota käyttäjä kokeilee kuviossa 3. Morton Heiligin Sensorama oli arcade-pelikoneen tapainen laite, jonka avulla pystyttiin stimuloimaan kaikkia aisteja. Laitteeseen oli rakennettu sisäisesti äänentoisto, stereoskooppinen 3D-näyttö, tuulettimet, hajugeneraattorit sekä värisytin. Sensoramaan tuotettiin yhteensä 6 kappaletta lyhytelokuvia, jotka Morton Heilig itse tuotti. (Virtual Reality Society 2017.)



KUVIO 3. Sensorama (Spence, Obrist & Velasco 2017)

Vuonna 1960 Morton Heilig toteutti seuraavan laitteensa eli ensimmäiset virtuaalilasit. Laitteessa oli stereoskooppinen 3D näyttö, mutta ei liikkeentunnistusta. Vuotta myöhemmin kaksi insinööriä Philco-korporaatiosta kehittivät uuden virtuaalilasin, Headsightin. Headsightista löytyi uutena ominaisuutena näytöt molemmille silmille ja magneettinen liikkeentunnistus. Laite oli yhdistetty suoraan kameraan, mutta kyseiselle laitteelle ei ollut kehitetty vielä yhtään sovelluksia, koska virtuaalinen todellisuus käsitteenä ei ollut vielä olemassa vaan laitetta käytettiin armeijassa helpottamaan vaarallisten tilanteiden seurantaa. (Virtual Reality Society 2017.)

3.3 Oculus Rift

2012-luvulla virtuaalitodellisuus alkoi taas kukoistamaan, kun Valve kehitti yksinkertaisen virtuaalisen todellisuuden laitteiston. Tämä laitteisto sisälsi virtuaalilasit, kameran ja siinä oli myös liikkeentunnistus. Liikkeentunnistus toteutettiin käyttämällä AprilTageja, jotka ovat

QR-koodin tyyppisiä mustavalkoisia kuvia koodeista. Samaan aikaan Oculus alkoi kehittämään omaa virtuaaliseen todellisuuteen kykenevää laitteistoa ja Oculus saikin tukijoilta tarpeeksi rahaa kehitystyöhön, jolloin Valve päätti luopua omasta suunnitelmastaan.

Valve ryhtyi tukemaan Oculusta, mutta yhteiset suunnitelmat kariutuivat erimielisyyksiin siitä, mitä virtuaalisen todellisuuden pitäisi olla. Tämän jälkeen Facebook osti Oculuksen, jolloin kyseinen polku sulkeutui. Samaan aikaan puhelinkehittäjä HTC oli luomassa omia virtuaalilaseja ja Valve päätyi tekemään HTC:n kanssa yhteistyötä. Molemman yhtiön visiot olivat samansuuntaiset, jolloin kehitys pystyi alkamaan kunnolla. (Souppouris 2016.) Oculus sai julkaistua loppujen lopuksi omat virtuaalilasinsa vuonna 2016 (Alex 2018).

3.4 Laitteisto

Oculus Rift sisältää vakiona virtuaalilasit, kaksi kappaletta sensoreita sekä Oculus Touch -ohjaimet. Virtuaalilasit asetetaan päähän siihen mukana kuuluvilla nauhoilla, ja nauhoista löytyvät korvakuulokkeet ääntä varten. Virtuaalilasien sisältä löytyvät tarvittavat magnetometri, kiihtyvyysanturi, gyroskooppi ja ledit. Laitteisto hyödyntää ledejä tarkemman sijainnin seuraamisessa. Tällä hetkellä kyseisten antureiden ansiosta virtuaalilasien sijaintia ja liikettä pystytään havaitsemaan yli 250 kertaa sekunnissa. Virtuaalilaseissa on kaksi näyttöä ja kaksi kappaletta linssejä, joiden avulla vältytään suuremmilta pahoinvointia aiheuttavilta ongelmilta tai näytön sumeudelta. (Nield 2016.)

KUVIO 4. Oculus Riftin virtuaalilasit ja ohjaimet (OneBonsai 2018)

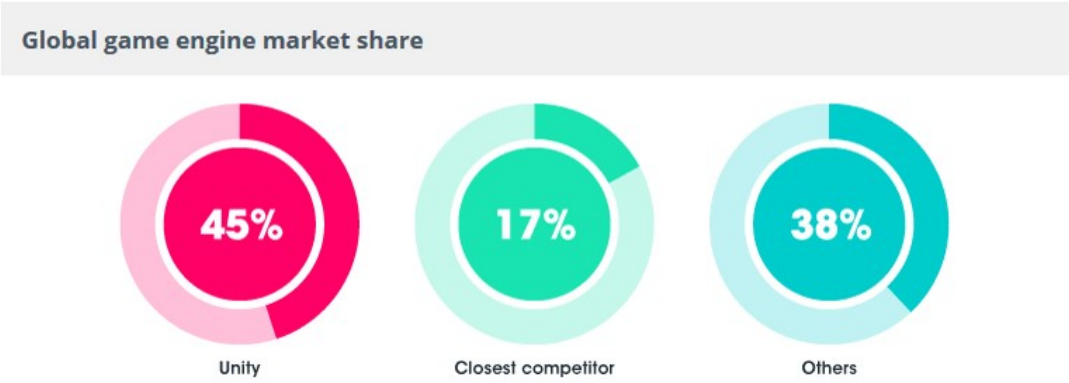
Oculus Riftin mukana tulevien sensoreiden avulla pystytään havaitsemaan virtuaalilasien sijainti sekä ulkoisten ohjainten sijainti tarkasti. Mukana tulevat ohjaimet on suunniteltu molemmille käsille, ja niistä pidetään kiinni rumpukapuloiden tapaan. Kuviossa 4 on nähtävissä Oculus Riftin virtuaalilasit ja ohjaimet. (Laukkonen 2018.)

4 UNITY

4.1 Historia

Unity on pelimoottori, jolla pystytään luomaan pelejä sekä sovelluksia yli 25:lle eri alustalle (Unity Technologies 2019o). Unityn on kehittänyt Unity Technologies, joka sai alkunsa Tanskassa kolmen henkilön toimesta: David Helgason, Joachim Ante ja Nicholas Francis. Heidän ajatuksenaan oli luoda amatööreille pelintekijöille sovellus, jonka avulla pelien valmistus olisi helpompaa.

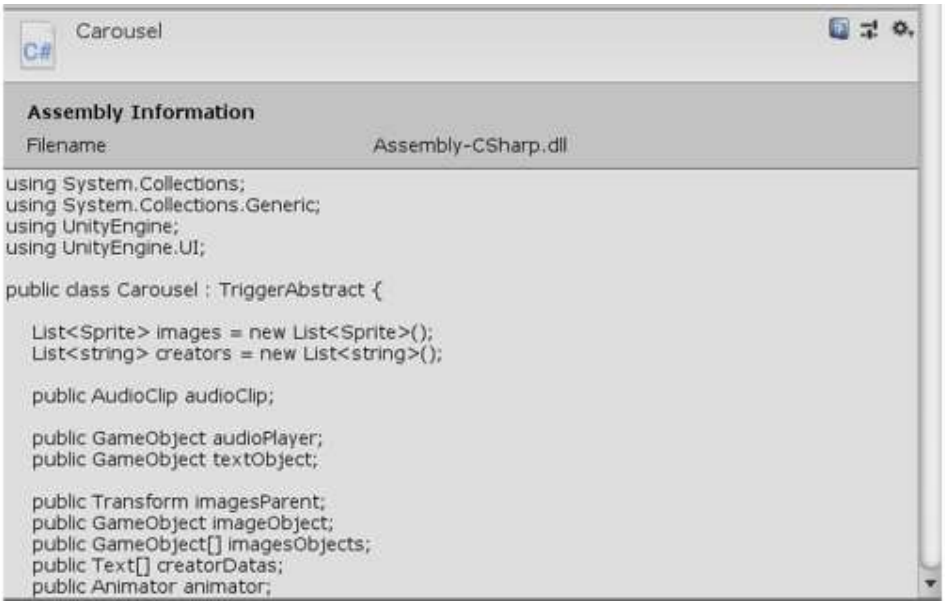
Alkuun Unity toimi vain Mac:lla, mutta laajeni nopeasti PC:lle sekä verkkoselaimille. Vuonna 2008 Apple julkisti oman kauppa-alustan iPhonelle, ja Unityn kehittäjät loivat sille nopeasti tuen. Nopean tuen myötä monet pelialan kehittäjät alkoivat käyttämään Unitya ja se alkoi laajenemaan suosituimmaksi pelimoottoriksi. Seuraavassa kuviossa 5 on esitetty Unityn markkinaosuus. (Brodkin 2013.)



KUVIO 5. Unityn suosio

4.2 Unityn ohjelmistoarkkitehtuuri

Pelimoottorit ovat kehikoita pelinkehittäjille. Pelimoottoreiden avulla pelinkehittäjät pystyvät nopeuttamaan pelinkehitysprosessiaan. Niiden avulla yhdistetään tarvittavat ominaisuudet, joita peleissä tarvitaan. Näihin kuuluvat esimerkiksi fysiikkamoottori, käyttöliittymä, verkkopeliominaisuudet, grafiikat ja ohjelmistokoodit. Näiden avulla pystytään toteuttamaan peleihin tarvittavia näkymiä ja ympäristöjä sekä lisäämään niihin interaktiivisuutta, erikoisefektejä, luomaan valaistusta tai optimoimaan ohjelmistoa toimimaan halutulla alustalla. (Unity Technologies 2019f) Suurin osuus on ohjelmistokoodilla, jonka avulla peleihin pystytään toteuttamaan uusia käyttäytymismalleja sekä hallinnoimaan peliobjektien välisiä suhteita (Kuvio 6). Ohjelmistokoodilla voidaan luoda peliobjekteille esimerkiksi liikettä pelaajan painaessa näppäintä tai luoda graafisia objekteja. (Unity Technologies 2019r.)



KUVIO 6. Ohjelmistokoodia

4.2.1 MonoBehaviour

MonoBehaviour-luokka on Unityn ohjelmistokoodin perusta, jolla kehittäjän komponentit näkymässä saadaan yhdistettyä Unityn sisäisiin pelimoottorisilmukoihin. Kaikkien luotujen komponenttien tulee periytyä tästä luokasta, jos ne halutaan lisätä ohjelmistosta löytyviin peliobjekteihin. MonoBehaviour-luokasta löytyy valmiita funktioita, joita Unity kutsuu ohjelmiston suorituksen aikana. Yleisimpiä näistä ovat funktiot Awake, Start ja Update. (Unity Technologies 2019i.)

Awake-funktiota kutsutaan, kun instanssi on ladattu ensimmäisen kerran ennen ohjelmiston suorituksen alkua. Tämä on hyvä funktio alustaa komponentti. Awake funktiota kutsutaan satunnaisessa järjestyksessä ja vain kerran koodin elinkaaren aikana. Kuviossa 7 asetetaan delegaattifunktiota Awake-funktiossa. Tämä tapahtuu ennen Start-funktion kutsua. (Unity Technologies 2019j.)

```
public Jukebox jukebox;
AudioClip audioClip;

private void Awake()
{
    onTriggerOpened = () => jukebox.PlayAudio(audioClip);
}
```

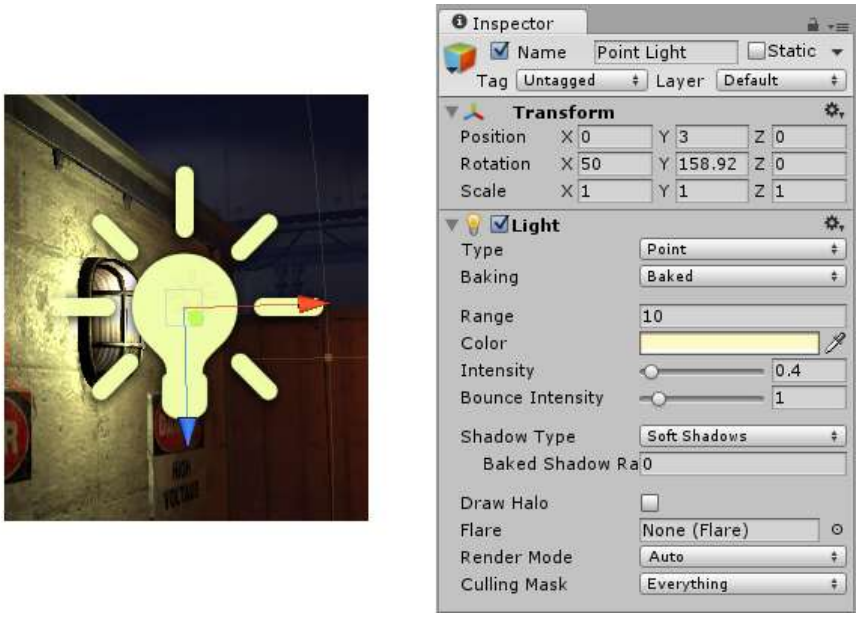
KUVIO 7. Awake funktio

Start-funktiota kutsutaan aina ennen Update-funktiota, kun komponentti tulee ensimmäistä kertaa aktiiviseksi. Start-funktiota kutsutaan vain kerran sen elinaikana. Eroavaisuutena Awake-funktioon on se, että Awake-funktiota kutsutaan, vaikka komponentti ei olisikaan aktiivinen. (Unity Technologies 2019k.)

Update-funktiota kutsutaan yhden kerran ennen ruudunpäivitystä. Se on funktio, jota kutsutaan toistuvasti, jolloin sitä voidaan käyttää esimerkiksi laskemaan pelissä kulunutta aikaa. Kaikki komponentit eivät kuitenkaan Update-funktiota tarvitse. (Unity Technologies 2019l.)

4.2.2 Peliobjektit ja komponentit

Unityssa hahmojen, esineiden ja muiden asioiden esittäminen on toteutettu peliobjekteilla. Ne ovat eräänlaisia säiliöitä, jotka eivät itsessään toteuta mitään, vaan ne tarvitsevat komponentteja. Kaikilla peliobjekteilla on aina muunnoskomponentti, jonka avulla niiden sijaintia (Position), suuntausta (Rotation) ja kokoa (Scale) pystytään muokkaamaan. Kuviossa 8 on esitetty näkymässä toimiva valonlähde lisäämällä valokomponentti peliobjektiin. (Unity Technologies 2019g.)



KUVIO 8. Valonlähde näkymässä

Ohjelmistokoodin avulla pystytään luomaan uusia komponentteja, joilla peliobjekteihin saadaan syvällisempiä toiminnallisuuksia, esimerkiksi liikuttamaan tarvittavia peliobjekteja tai luomaan ääntä taustalle. Unitysta löytyy valmiita komponentteja, mutta niiden käyttötarkoitukset ovat rajattuja. Monimutkaisempia toiminnallisuuksia varten on komponentteja ohjelmoitava itse. Luotujen komponenttien luokan nimien täytyy myös täsmätä

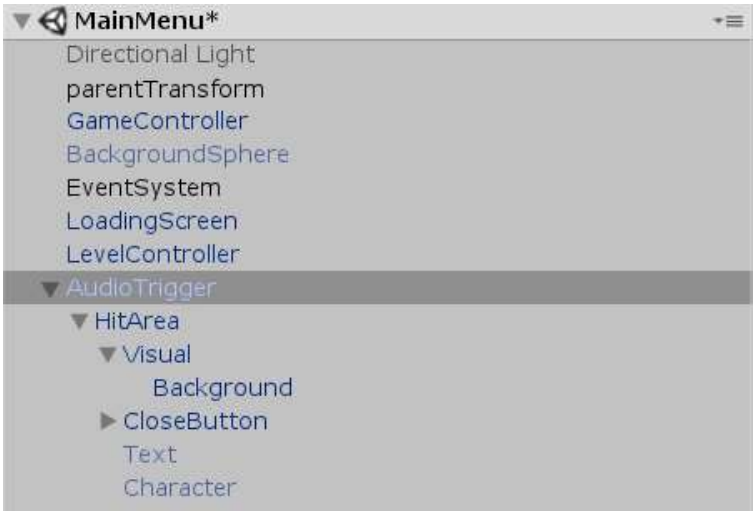
komponentin tiedoston nimen kanssa, muuten niitä ei voida asettaa peliobjekteihin. (Unity Technologies 2019e.)

KUVIO 9. Valmis elementti

Peliobjekteista pystytään luomaan myös valmiita elementtejä (Prefab), joiden avulla voidaan luoda instansseja tarvittavista peliobjekteista (Kuvio 9). Valmiit elementit sisältävät peliobjektiin asetetut komponentit sekä lapsipeliobjekit. Tämän ansiosta pelissä tarvittavien peliobjektien tiedot ovat samanlaiset, mutta jokaisen luodun instanssin komponenttien asetuksia pystytään säätämään erilaisiksi. Kaikki muutokset, mitä valmiiseen elementtiin tehdään kehitysympäristössä, heijastuvat jokaiseen kyseisen elementin instanssiin näkymässä. Ohjelmiston ajon aikana luodut instanssit ovat tarpeellisia esimerkiksi, kun halutaan luoda lentäviä ammuksia tai muita vaihtuvia peliobjekteja. (Unity Technologies 2019n.)

4.2.3 Hierarkiapuu

Hierarkiapuu on Unityn kehitysympäristössä paikka, jonne kaikki näkymässä olevat peliobjektit listataan. Ne listataan hierarkiapuuhun luontijärjestyksessä, mutta niiden uudelleenjärjesteleminen on mahdollista. Kun peliobjekti poistetaan näkymästä, niin se katoaa myös hierarkiapuusta. Peliobjektit on mahdollista asettaa isä-lapsi-peliobjekteiksi, jolloin luotu peliobjekti asetetaan hierarkiassa kuviossa 10 esitetyllä tavalla toisen peliobjektin sisälle. Tämä onnistuu joko vetämällä tai luomalla uusi peliobjekti suoraan toisen peliobjektin päälle. Lapsipeliobjekteja voidaan luoda useita ja jokaisella lapsella voi olla omia lapsipeliobjekteja. (Unity Technologies 2019t.)



KUVIO 10. Peliobjekteja hierarkiapuussa

4.3 Oculuksen ohjelmointirajapinta

Oculus on toteuttanut oman ohjelmointirajapinnan, jonka avulla pystytään luomaan virtuaaliseen todellisuuteen perustuvia pelejä, kokemuksia tai opetussovelluksia. Sen saa ladattua Oculuksen omilta kehittäjille suunnatulta sivulta. Oculuksen ohjelmointirajapinta sisältää tarvittavat komponentit Oculus Riftin hallintaa varten virtuaalisessa todellisuudessa. Sillä pystytään myös ohjelmoimaan sovelluksia Oculuksen mobiilivirtuaalilaseille, Oculus Questille ja Oculus Golle. Ohjelmointirajapinnasta löytyy suorat tuet Unity- ja Unreal Engine-pelimoottoreille. (Facebook Technologies 2019b.)

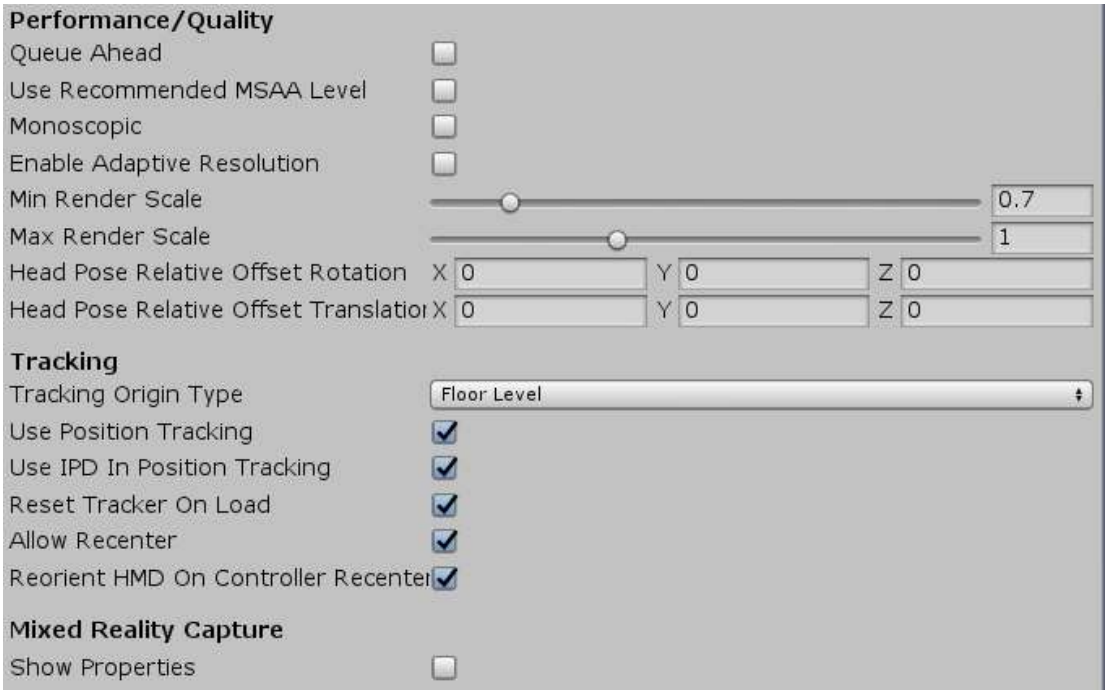
Kaikki Unityn versiot 5.1 ja siitä ylöspäin omaavat valmiiksi tuen virtuaaliselle todellisuudelle. Tämän asetuksen saa päälle Unityn asetuksista. Kun asetuksen laittaa päälle, niin Unityn kaikki kamerat alkavat seuraamaan sijaintiaan kamerakomponentin omaavan peliobjektin sijainnista. Jos näkymään on tuotu OVRCameraRig peliobjekti, niin tämä sijainti määritellään kyseisen peliobjektin mukaisesti. (Facebook Technologies, LLC 2019a.)

4.3.1 OVRCameraRig-elementti

OVRCameraRig on Oculuksen ohjelmointirajapinnasta löytyvä valmis elementti, jolla pystytään korvaamaan Unityn kamerakomponentti. Se tuodaan näkymään vetämällä se hierarkiapuuhun. Kyseinen elementti sisältää kolme Unityn kameraa, joista 2 on eri silmille ja 1 yksi molemmille, peliobjektin, joka tarkkailee asentoa, ja peliobjektin, jolla pystytään tarkkailemaan Oculus Riftin tarvitsemaa aluetta. OVRCameraRig sisältää myös OVRManager-komponentin, jonka päätehtävänä on hallita Oculus Riftin laitteistoa. (Facebook Technologies, LLC 2019a.)

4.3.2 OVRManager-komponentti

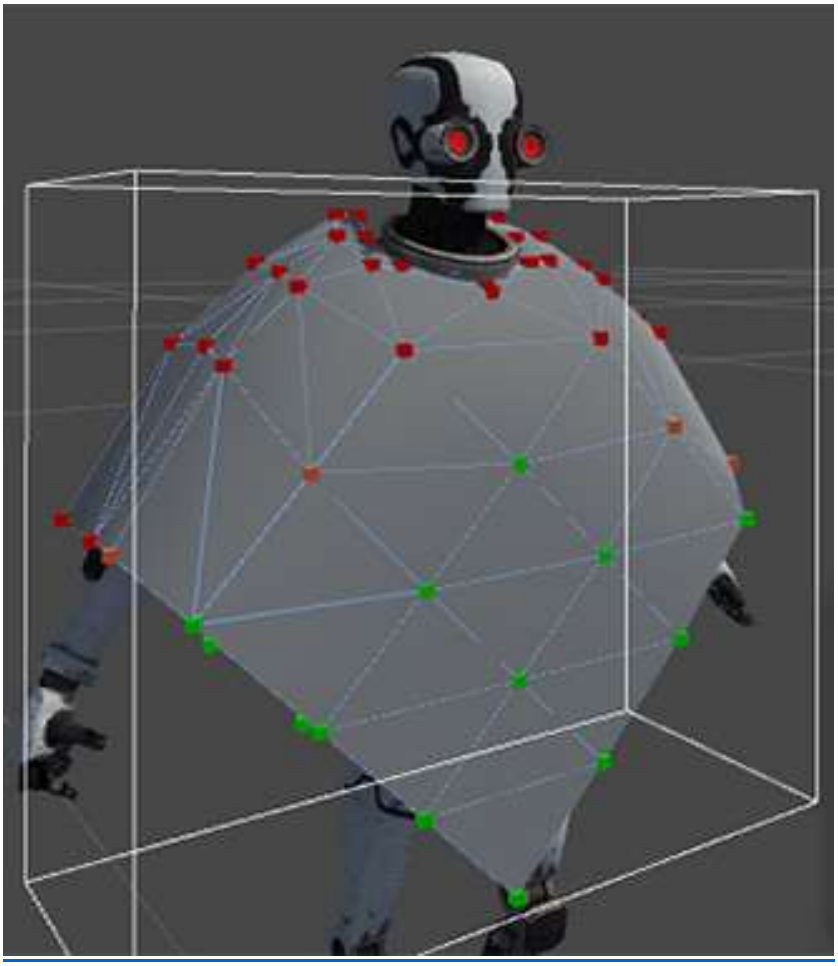
OVRManager-komponentti on singletoni, joka on päärajapinta Oculus Riftin laitteistoon. Siitä löytyy julkisia muuttujia, joiden avulla Oculus Riftiä pystytään säätämään. Yhtenä säätönä löytyy ominaisuus, jonka avulla ohjelmisto asettaa suositellun resoluution automaattisesti nykyiselle tietokoneen laitteistolle. Sillä pystytään myös määrittämään se, seurataanko virtuaalilasien sijaintia lattiatasosta, vai silmien tasolta. OVRManager sisältää myös monia muita ominaisuuksia (Kuvio 11), joiden ansiosta laitteisto saadaan käyttäytymään ohjelmistossa halutulla tavalla. Jos kyseinen komponentti löytyy näkymästä, niin Unityn virtuaalisen todellisuuden tuki aktivoidaan automaattisesti. (Facebook Technologies, LLC 2019a.)



KUVIO 11. OVRManagerin asetuksia

4.4 Fysiikkamoottori

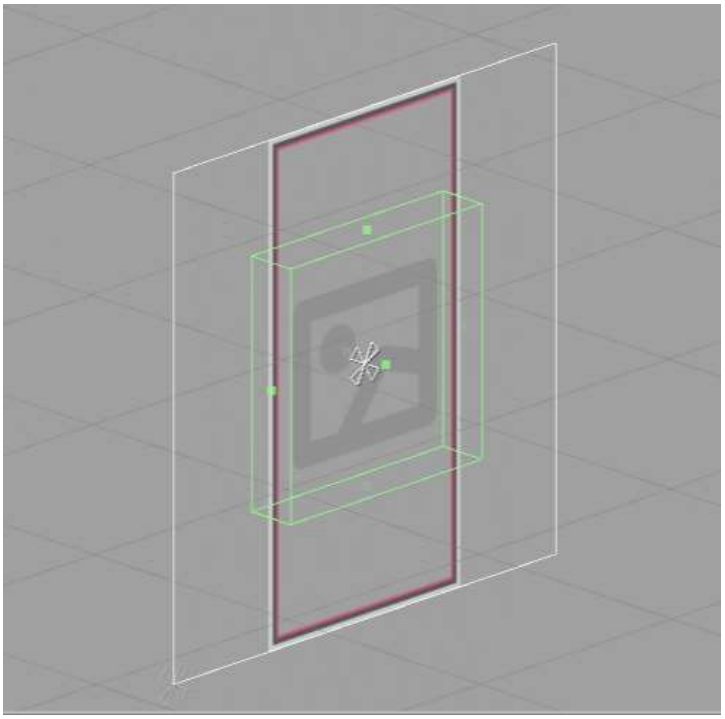
Unityn fysiikkamoottorin ansiosta peliobjekteille pystytään toteuttamaan fysiikan lakeja mallintavia ominaisuuksia, kuten törmäyttimiä, kiihtyvyyttä ja massaa. Fysiikkamoottoreita on Unityssa kaksi kappaletta, toinen kaksiulotteiselle ja toinen kolmeulotteiselle mallinnukselle. Ohjelmistokoodin avulla pystytään luomaan peliobjekteille erilaisia käyttäytymisiä kuten liikkumista, auton hallintaan tarvittavia fysiikoita tai esimerkiksi kangasta mallintavia tiloja (Kuvio 12). (Unity Technologies 2019m.)



KUVIO 12. Kangasfysiikka

4.4.1 Collider

Törmäyttimet (Collider) ovat Unitysta löytyviä komponentteja, joiden avuilla peliobjektit havaitsevat törmäyksiä. Näitä törmäyttimiä löytyy Unitysta primitiivisinä tai ne voivat seurata peliobjektin mallin muotoja. Primitiivisiä törmäyttimiä ovat laatikkotörmäytin (Box Collider), kapselitörmäytin (Capsule Collider) ja pallotörmäytin (Sphere Collider). Seuraavassa kuviossa 13 on asetettuna peliobjektiin primitiivinen laatikkotörmäytin. Peliobjektiin asetettujen törmäyttimien ei tarvitse noudattaa peliobjektin muotoja vaan niitä pystyy vapaasti muokkaamaan isommiksi tai pienemmiksi. Niitä voidaan myös asettaa yhteen peliobjektiin useampia kappaleita tai luoda lapsipeliobjekteja, joihin asetetaan omat törmäyttimet. (Unity Technologies 2019d.)



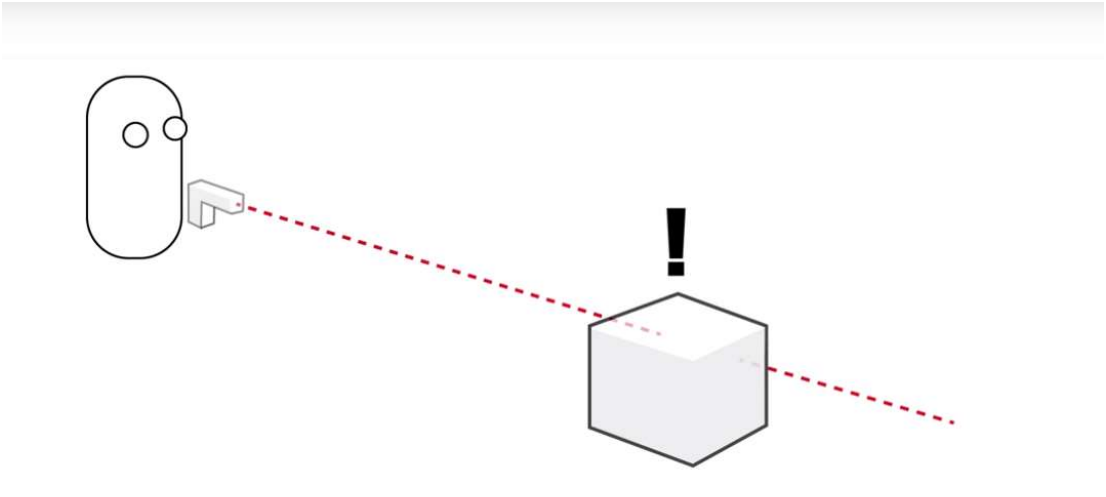
KUVIO 13. Laatikkotörmäytin

Primitiiviset törmäyttimet rasittavat prosessoria vähemmän, minkä takia niitä tulisi käyttää liikkuvissa peliobjekteissa. 3D-mallien muotoja seuraavia törmäyttimiä taas käytetään useimmiten maastossa löytyvistä peliobjekteissa, koska niiden ei tarvitse liikkua. Näitä paikallaan olevia törmäyttimiä kutsutaan staattisiksi törmäyttimiksi. Unityn fysiikkamoottori tulkitsee staattiset törmäyttimet paikallaan oleviksi ja tämän ansiosta pystyy optimoimaan niiden toimintaa paremmin. Jos kyseisiä törmäyttimiä liikutetaan tai niiden suuntausta vaihdetaan, niin se aiheuttaa isoja kuormituspiikkejä laskelmissa ja pahimmassa tapauksessa törmäytin saattaa esiintyä ohjelmistossa paikassa, jossa se ei oikeasti ole. (Unity Technologies 2019d.)

4.4.2 Raycast ja RaycastHit

Unitysta löytyy säteitä, joita pystytään määrittelemään kameran näkymän avulla. Jokainen piste, joka näkyy kamerasta vastaa yhtä viivaa maailmassa. Näille viivoille löytyy Unitysta oma olio Ray. Unityn kameraluokalla on olemassa valmiiksi kaksi funktiota, joiden avulla säteitä pystytään määrittelemään tarkemmin; ScreenPointToRay ja ViewportPointToRay. ScreenPointToRay olettaa, että säteen piste on määritelty pikselin mukainen koordinaatti. Pikselin koko vaihtelee näytön koon mukaan. ViewportPointToRay taas olettaa säteen pisteen olevan normalisoitu vektori, jossa nolla on pienin arvo ja yksi on isoin arvo.

Normalisoidulla vektorilla (0,0) tarkoitetaan näytön vasenta alakulmaa ja vektorilla (1,1) näytön oikeata yläkulmaa. (Unity Technologies 2019b.)



KUVIO 14. Säteen heijastus (Unity Technologies 2013)

Säteen heijastus (Raycast) on tapa, jolla pystytään havaitsemaan pelimaailmasta löytyviä törmäyttimiä. Kuten kuviossa 14 näkyy, niin siinä lähetetään säde tutkimaan aseen piipusta eteenpäin löytyviä törmäyttimiä. Säteestä löytyy asetukset lähtöpaikalle, suunnalle ja sille miten pitkälle säde matkaa. Osuttuaan törmäyttimeen säteestä saadaan tieto mihin peliobjektiin se on osunut ja mikä on osumakohta. (Unity Technologies 2019q.)

Säteen osuman kohde saadaan tallennettua RaycastHit-tietueeseen, jolloin siitä saadaan selville tarkempia tietoja. RaycastHit omaa tiedot esimerkiksi kohteen etäisyydestä tai osuman sijainnista koordinaatteina maailman mukaisesti. Siihen tallentuu myös osuman saaneen peliobjektin muunnoskomponentti ja se mihin kohtaan tekstuurissa säde on osunut. (Unity Technologies 2019p.)

4.5 WWW

Unitysta löytyy valmis WWW-luokka, jonka avulla pystytään lataamaan tiedostoja ohjelmiston ulkopuolelta, esimerkiksi verkkosivuilta ja kansioista. WWW-luokan rakentaja hyväksyy parametrina merkkijonon URL muodossa. Sen hyväksytyt protokollat ovat http, https ja file. Kun file-protokollaa käytetään Windows tai Windows store-ympäristössä, niin sen pitää alkaa muodossa file:///. WWW-luokan lataus on asynkroninen operaatio, jolloin palautuva arvo saadaan vasta silloin kun lataus on suoritettu. Latauksen tilaa pystytään seuraamaan WWW-luokan isDone arvoa tarkkailemalla tai se voidaan toteuttaa yieldinä

IEnumeraattorissa (Kuvio 16), jolloin ohjelmiston suoritus jatkuu, kun data on ladattu. (Unity Technologies 2019v.)

```
string url = "file:///" + (CurrentLevelFolder + "images/" + backgroundFile).Replace("\\", "/");
using (WWW www = new WWW(url))
{
    yield return www;
    Texture2D texture = new Texture2D(www.texture.width, www.texture.height, www.texture.format, false);
    www.LoadImageIntoTexture(texture);
    texture.Apply(false, true);
    LevelController.instance.SetBackgroundImage(texture);
}
```

KUVIO 16. WWW-olio

4.6 IEnumerator ja yield

IEnumerator on .NET kirjaston rajapinta, jonka avulla pystytään hallitsemaan listamuotoisia objekteja. Esimerkiksi yleinen foreach-silmukka on toteutettu IEnumerator rajapinnalla. IEnumeratorilla pystytään lukemaan palautettuja arvoja, mutta niiden muokkaaminen ei ole mahdollista. (Microsoft 2019a.) Yieldin avulla pystytään toteuttamaan iteraatio IEnumeraattorille, jolloin ylimääräisten luokkien määrittämiselle ei ole tarvetta. Yieldin palautettavan arvon täytyy olla tyyppiä IEnumerable, IEnumerable<T>, IEnumerator tai IEnumerator<T>. (Microsoft 2019b.)

4.7 SimpleJSON

JSON on kevyt ja helppolukuinen tiedostomuoto tiedonvälitykseen. Sen käytänteet muistuttavat etäisesti C luokan ohjelmistoperheiden käytänteitä, joten sen käyttäminen on tuttua ja sillä ei ole kielirajoitteita. JSON:n rakenne on JavaScriptille tutun objektin muotoinen ja se sisältää avainarvopareja sekä taulukoita. Kuviossa 17 on JSON-muotoinen objekti, jossa on asetettuna avaimia ja arvoja. Miltei jokainen nykyajan ohjelmointikieli tukee taulukoita sekä kokoelmia, joten JSON käyttäminen on luonnollista. (Crockford 2019.)

```
"1":{
    "type" : "carousel",
    "rotation": "0",
    "angle" : "110",
    "width" : 2,
    "height": 2,
    "yheight": 0,
```

KUVIO 17. JSON muotoinen objekti

SimpleJSON on Unify kommuunin toteuttama yksinkertainen JSON-parseri. Sillä on helppo lukea JSON-objekteista löytyviä arvoja sekä tallentaa niitä. Kuviossa 18 näkyvässä

esimerkissä näytetään, miten JSON-oliosta saadaan luettua arvoja SimpleJSON-parserin avulla. SimpleJSON omaa omat luokat jokaiselle tarvittavalle JSONin tietotyypille, jolloin tarvittavien arvojen tarkastelu on helpompaa. (Unity Wiki 2017.)

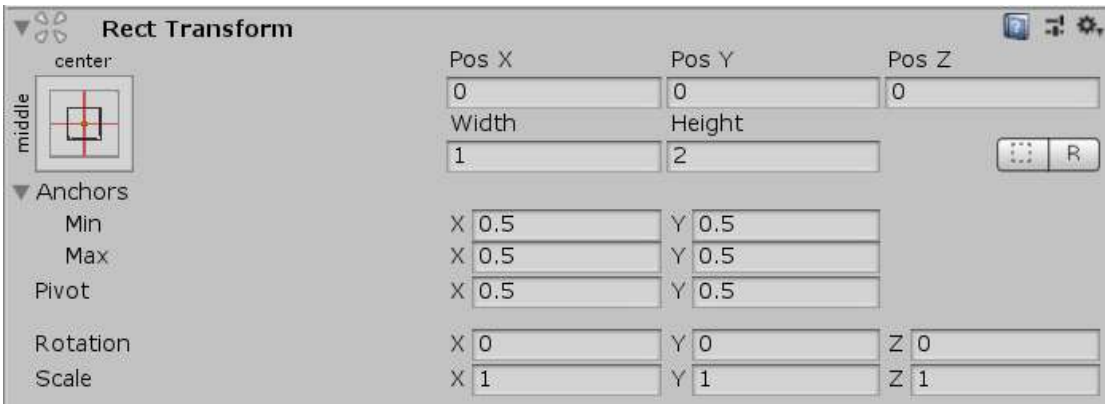
```
var N = JSON.Parse(the_JSON_string);
var versionString = N["version"].Value;        // versionString will be a string containing "1.0"
var versionNumber = N["version"].AsFloat;      // versionNumber will be a float containing 1.0
var name = N["data"]["sampleArray"][2]["name"];// name will be a string containing "sub object"

//C#
string val = N["data"]["sampleArray"][0];      // val contains "string value"
```

KUVIO 18. SimpleJSON (Unity Wiki 2017)

4.8 Unity UI

Unitysta löytyy kirjasto käyttöliittymän luontia varten. Kyseisestä kirjastosta löytyy valmiita elementtejä käyttöliittymän visuaalisia ja interaktiivisia toiminnallisuuksia ajatellen. Kaikkien käyttöliittymäelementtien muutoskomponentti on korvattu suorakulmiomuutoskomponentilla (RectTransform) (Kuvio 19). Suorakulmiomuutoskomponentin avulla pystytään muuttamaan käyttöliittymäelementtien kokoa, suuntausta ja paikkaa. Tämä tapahtuu joko syöttämällä halutut arvot julkisiin muuttujiin tai vetämällä suorakulmamuutoskomponentin kulmia näkymässä. Jos hiiren asettaa lähelle komponentin kulmaa alueen ulkopuolelle, pystytään kääntämään haluttua elementtiä.



KUVIO 19. Suorakulmiomuutoskomponentti

Suorakulmiomuutoskomponentin koon vaihtuessa komponenttia ei skaalata normaalin muutoskomponentin tapaan, vaan sen kokoa muutetaan. Tämä koon muutos ei vaikuta tekstien kokoihin tai pilkottujen kuvien reunoihin. Kyseisestä komponentista löytyy myös ankkurit, joiden avulla pystytään asettamaan elementti halutulle paikalle. Ankkurin asettamista varten voidaan käyttää siihen löytyvää ankkurityökalua, josta voidaan määrittää halutut valmiit paikat. Ankkurin voi myös asettaa suorakulmiomuutoskomponentille vetämällä

näkyviä pieniä kolmioita kulmista, jolloin saadaan toteutettua monimutkaisempi ankkurointi. Ankkuroinnin avulla pystytään venyttämään lapsipeliobjekteja, kun näytön tai piirtoaluekomponentin koko muuttuu, tai asettamaan lapsipeliobjektit olemaan aina tietyssä paikassa käyttöliittymää piirrettäessä. (Unity Technologies 2019a.)

4.8.1 Piirtoaluekomponentti

Suorakulmamuutoskomponenttien omaavien peliobjektien isäpeliobjektina tulee olla piirtoaluekomponentin (Canvas) omaava peliobjekti. Piirtoaluekomponentti mahdollistaa käyttöliittymäelementtien näyttämisen ohjelmiston suorituksen aikana. Jos kyseistä komponenttia ei ole näkymässä käyttöliittymäelementin luonnin aikana, niin se luodaan näkymään automaattisesti. Piirtoaluekomponentti pystytään asettamaan kolmeen erilaiseen tilaan. Ensimmäinen tila on päällystäminen (Overlay), jolloin piirtoalue piirretään näytön kokoiselle alueelle. Piirtoalueen koko vaihtuu tällöin vastaamaan näytön kokoista aluetta. Toinen tila on kamera, milloin piirtoalueelle asetetaan haluttu kamera. Tällöin piirtoalue piirretään kyseisen kameran alueelle. Kolmantena tilana on maailmantila (World), missä piirtoalue näytetään kolmiulotteisessa maailmassa. Tämä on kätevää silloin kun halutaan, että käyttöliittymäelementit näytetään osana maailmaa, kuten kuviossa 20 näkyy. (Unity Technologies 2019c.)

KUVIO 20. Maailmatila

4.8.2 Tekstikomponentti

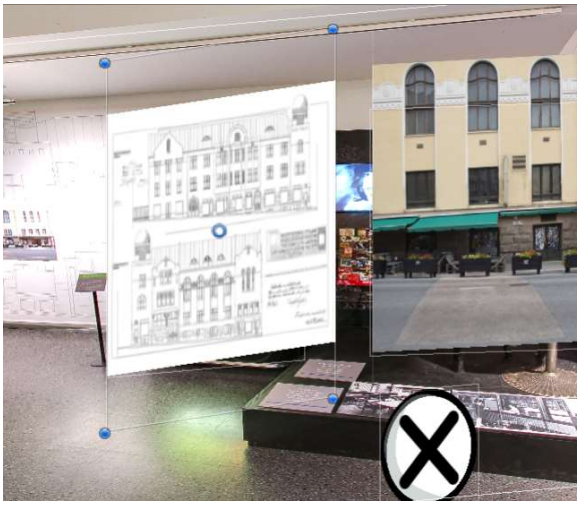
Käyttöliittymän luontia varten Unitysta löytyy visuaalisia ja interaktiivisia komponentteja. Visuaaliset komponentit ovat näkyviä ja niillä ei ole valmiiksi toteutettuja toiminnallisuuksia. Näitä komponentteja on esimerkiksi teksti (Text) ja kuva (Image). Tekstikomponenttien tarkoituksena on näyttää käyttöliittymässä tekstiä (Kuvio 21) ja sille on määritelty alue, johon tekstiä pystytään asettamaan. Tekstikomponentista pystytään muokkaamaan tekstin fonttia, fontin kokoa ja sitä onko kyseinen teksti rikastettua. Rikastettu teksti hyväksyy markup-muotoisia muotoiluja, joiden avulla voidaan tarkemmin määritellä esimerkiksi tekstin paksuutta ja onko tietty osa kursivoitua. Tekstikomponentista löytyy myös asetus parhaalle sovitukselle, jolloin teksti näytetään niin isona, kun vain pystytään. (Unity Technologies 2019s.)

KUVIO 21. Tekstikomponentti

4.8.3 Kuvakomponentti

Kuvakomponentin on tarkoitus näyttää haluttu sprite-muotoinen kuva käyttöliittymässä (Kuvio 22). Halutun näytettävän kuvan voi asettaa komponentin Source Image-muuttujaan, jolloin se tulee näkyviin. Asetetun kuvan pystyy näyttämään neljässä eri mahdollisessa muodossa. Yksinkertainen (Simple) tilassa kuva näytetään sellaisenaan. Pilkotussa (Sliced) muodossa kuva käyttää niin sanottua 3x3 leikkausta, jolloin halutun kuvan kulmat eivät veny, ainoastaan keskimmäiset osat venyvät. Tiilitetty (Tiled) kuva taas on saman tapainen kuin pilkottu, mutta siinä vain keskiosaa toistetaan. Täytetty (Filled) näyttää

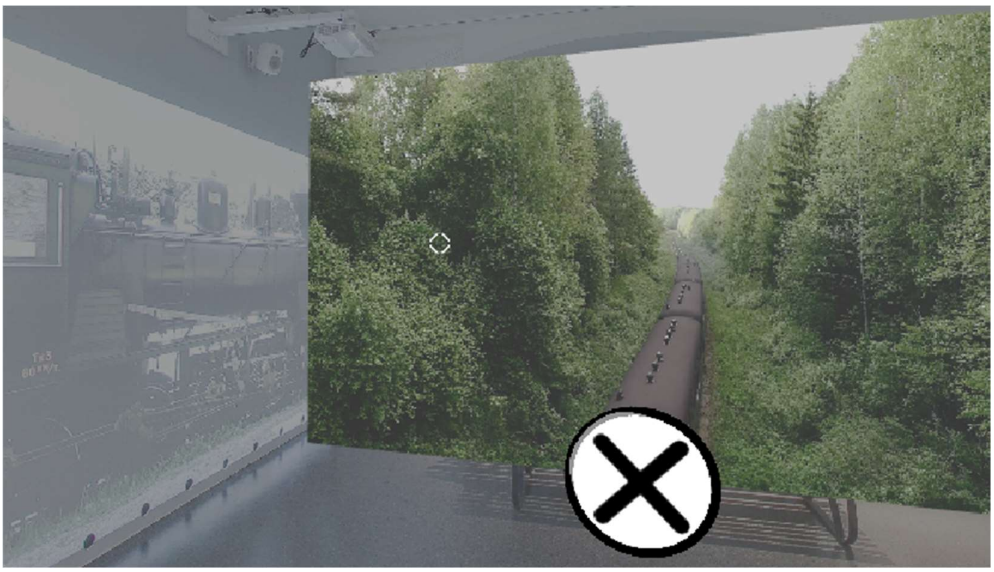
kuvan osittain tai kokonaan, riippuen määritellystä suunnasta ja täyttömäärästä. (Unity Technologies 2019h.)



KUVIO 22. Kuvakomponentti

4.9 VideoPlayer

Unityssa pystytään näyttämään videoita videosoitin-komponentilla (Videoplayer). Videosoitin komponentin videon pystyy asettamaan näkymiin peliobjektien materiaalien tekstuuripaikkaan kuten kuviossa 23, renderöinti tekstuuriin, suoraan kameraan tai tekstuuripaikkoihin komponenteissa. Videon lähteeksi voi valita joko asennuspaketin mukana olevan videon tai urlin, jolloin toistettava video voidaan hakea verkosta. (Unity Technologies 2019u.)

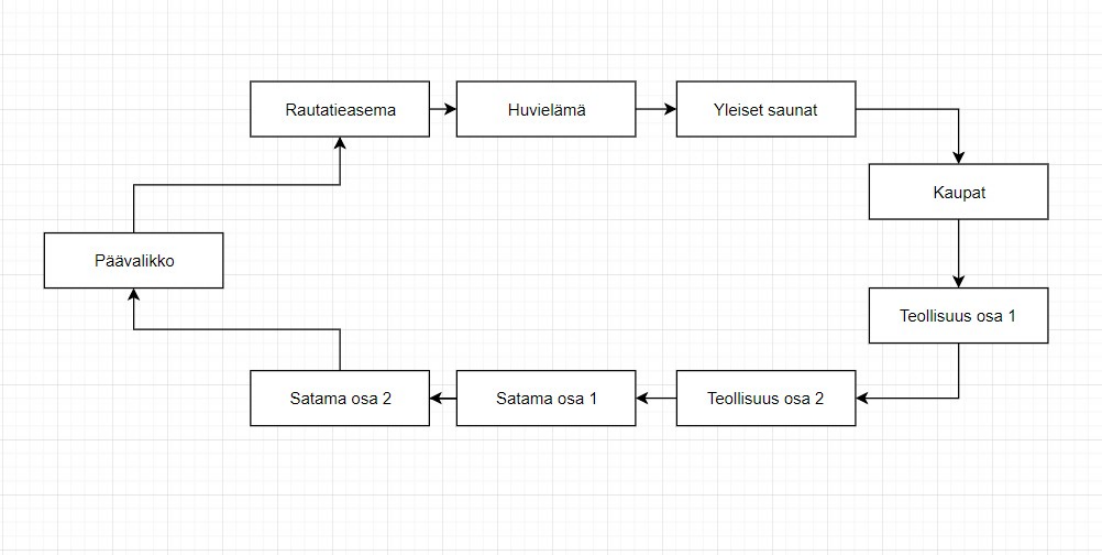


KUVIO 23. Videosoitin

5 OHJELMISTON TOTEUTUS

5.1 Ohjelmiston suunnittelu

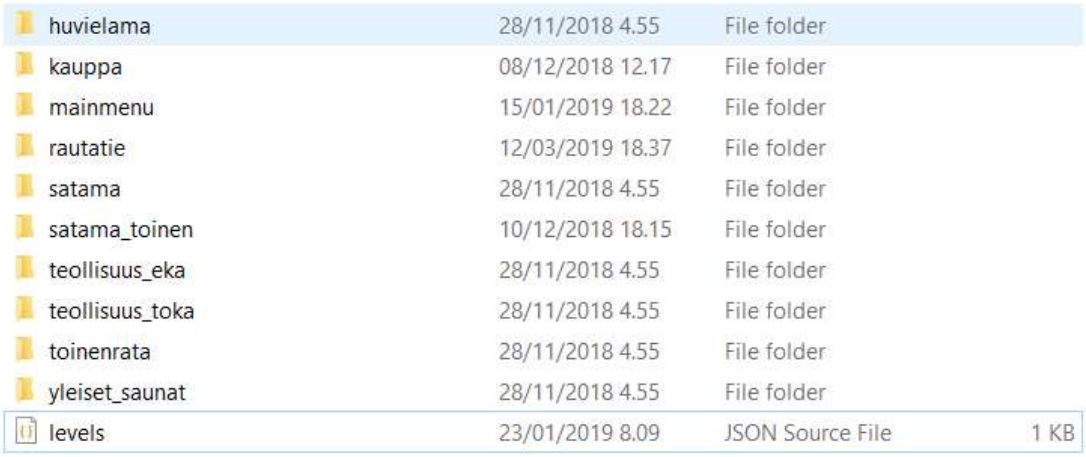
Tavoitteena oli ohjelmistossa toteuttaa todentuntuinen näyttely virtuaalisena. Lahden historialliselta museolta saaduissa suunnitelmissa oli mukana kahdeksan erilaista paikkaa, joissa oli näytillä vanhoja kuvia Lahdesta. Kyseiset paikat kuvastivat joko huoneita tai eri sijaintia samassa huoneessa. Jokaisella huoneistolla tai paikalla oli omat teemat, joiden myötä toteutettiin tarpeelliset peliobjektien elementit niiden esittämistä varten. Ohjelmiston näkökulmasta näiden huoneiden tai paikkojen virtualisoiminen toteutettiin käyttämällä pohjana samaa näkymää ja tuomalla tarpeellisia materiaaleja tai elementtejä huoneen vaihtuessa. Huoneistosta toiseen kulkeminen toteutettiin käyttämällä interaktiivista elementtiä, jossa näkyi jalanjäljet tai tarpeen vaatiessa näppäimistöä käyttämällä. Seuraavassa kuviossa 24 on esiteltynä kaavio, joka näyttää huoneistojen teemat sekä niiden kulun.



KUVIO 24. Huonerakenne

5.2 Kansiorakenne

Ohjelmistoon toteutettiin ulkoinen kansiorakenne, joka toimii selkärankana ohjelmiston logiikalle. Kansiorakenne sijoitettiin ohjelmiston exe-tiedoston kanssa samaan kansioon. Sen suunnittelussa otettiin huomioon tarve useammille kentille ja kenttien erilaisille sisällöille kuten kuviossa 25 näkyy. Tällä ajatuksella haluttiin taata se, että ohjelmiston sisällön pystyi vaihtamaan tarvittaessa kasaamisen jälkeen.



KUVIO 25. Kansiorakenne

Kansiorakenteen sisälle toteutettiin JSON-tiedostoja, joita sovellus luki. JSON-tiedostot ladataan normaalisti merkkijonomuotoisena ensin ohjelmistoon ja siitä parseroidaan SimpleJSON parserin avulla. Kansiorakenteen juureen toteutettiin pelin kenttiä hallitseva tiedosto, josta löytyvät kenttien järjestykselle tärkeät nimet ja indeksit. Jokainen kenttä tarvitsi myös oman kansion. Kenttien latautuessa ladattavan kentän kansion sisältä käydään lukemassa kenttää hallitseva JSON-tiedosto, jolloin saadaan oikeat määritykset näytettäville elementeille.

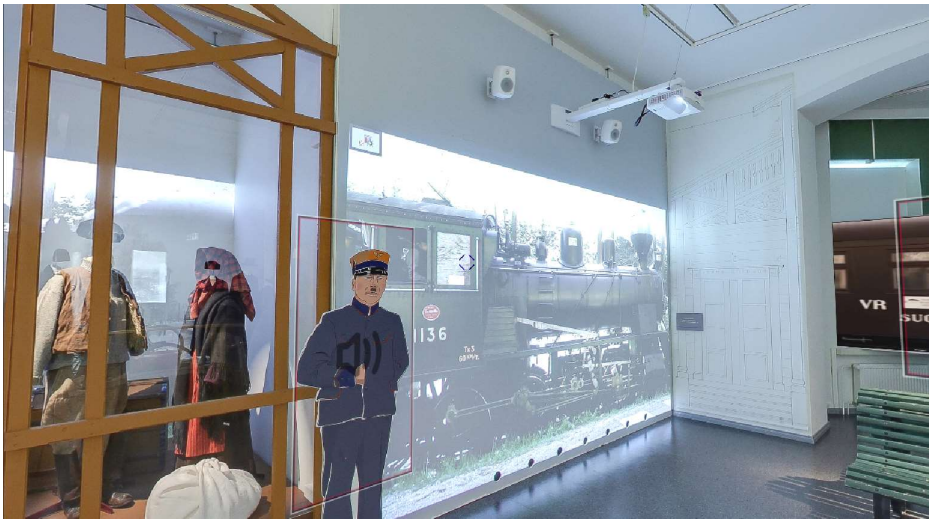
5.3 Kenttien rakenne

Sovellukseen toteutettiin kaksi kappaletta kenttiä. Toinen kentistä on päävalikko (Kuvio 26) ja toinen on pelikenttä. Päävalikon tarkoituksena oli luoda selvä tapa asettaa tarvittavat referenssit sovelluksen komponenteille. Päävalikon taustaääneksi laitettiin myös opasteääni, jolloin pelaaja saa opastuksen pelin alussa. Pelikentässä referenssejä ei aseteta, vaan sovellus suorittaa ainoastaan tarvittavien äänien, videoiden, kuvien ja taustakuvien lataamisen.

KUVIO 26. Päävalikkokenttä

5.3.1 Taustakuvan asettaminen

Sovelluksen kenttien taustana käytettiin 360 asteen kuvia. Nämä kuvat ladataan kansiorakenteesta Unityn WWW-luokan avulla ensimmäisenä. Halutun kuvan lataamisessa kestää pienen hetken, koska se pitää ladata ensin ulkoisesta kansiosta muistiin ja sen jälkeen luoda uudelleen uutena tekstuurina. Tämän jälkeen tekstuurista poistetaan mipmapit, joita käytetään, kun halutaan luoda huonolaatuisempi kuva kuvaa kauempaa katsottaessa. Ilman mipmappien poistoa taustakuvassa näkyvän kuvan saumaan ilmestyy suurempi rako.



KUVIO 27. Taustakuva

Taustakuvan latauksen jälkeen se asetetaan materiaaliin, joka on asetettu asetuksissa taivaslaatikolle (Skybox). Samaan materiaaliin on määritelty myös shader, joka tuo Unityyn tuen 360 kuvien heijastamiselle. Kuviossa 27 on näyttelyn tiloista otettu 360 asteen kuva ohjelmistossa taustakuvana.

5.3.2 Peliobjektien luonti

Kun taustakuva on ladattu, niin sovellus lataa tämänhetkisen näyttelyhuoneen JSON-tiedostosta kokoelman nimeltään ”hotspots”. Kyseisestä kokoelmasta löytyy avainarvopareja. Nämä arvot kuvastavat eri peliobjekteja, joita pelikentässä näkyy (Kuvio 28). Sovellukseen on luotu kahdeksan kappaletta erilaisia mahdollisia peliobjekteja; edellinen tai seuraava kenttä, kuvaobjekti, tekstiobjekti, kuvakaruselli, ääni, video sekä jukeboksi. Näistä kaksi, kuvaobjekti ja tekstiobjekti, ovat ainoat ei-interaktiiviset peliobjektit.


```
"hotspots":{
    "0":{
        "type" : "carousel",
        "rotation": "0",
        "angle" : "165",
        "width" : 4,
        "height": 2,
        "images" : {
            "0": "1.jpg",
            "1": "2.jpg",
            "2": "3.jpg",
            "3": "4.jpg",
            "4": "5.jpg",
            "5": "6.jpg"
        },
        "creators": {
            "0": "Eino Heinonen, Helsingin kaupunginmuseo",
            "1": "Nils Andersson, Helsingin kaupunginmuseo",
            "2": "Kari Hakli, Helsingin kaupunginmuseo",
            "3": "Kari Hakli, Helsingin kaupunginmuseo",
            "4": "Signe Brander, Helsingin kaupunginmuseo",
            "5": "Kari Hakli, Helsingin kaupunginmuseo"
        },
        "audio" : "sauna.WAV",
        "imagetext": "Helsinkiläisiä yleisiä saunoja 1910-luvulta 1970-luvulle."
    },
```


KUVIO 28. Peliobjekti JSON muodossa

Jokaiselle näistä annettiin yhteisiä muuttujia, joita peli hyödyntää, kuten suuntaus, korkeus, leveys, sijainti y-akselilla sekä oman akselin suuntaus. Sovellus lukee näiden tarvittavien muuttujien arvot ja luo uuden peliobjektin antaen tälle kyseiset arvot. Erilaisilla peliobjekteilla on myös omia arvoja, joita sovellus hyödyntää, jotta tarvittavat elementit

saadaan näytettyä. Jokaiselle interaktiiviselle objektille pystytään asettamaan myös taustaääni erikseen, jolloin kuville saadaan syvällisempää kerrontaa.

5.3.3 Kuvien ja äänien lataaminen

Peliobjektien sisältämät kuvat ja äänet ladataan Unityn WWW-luokalla. Kentän latautuessa on huomioitavaa pidempi latausaika kuvien latausten takia. Peliobjektien kuvat on toteutettu käyttäen Unityn UI kirjastoa ja sen valmiita peliobjekteja. Näissä tarkempien kuvien näyttämisen takia ladatut kuvat käännetään ladattaessa Sprite-muotoon, jolloin Unityn kuvakomponentti pystyy sen lukemaan. Ladattujen kuvien asetuksia on säädetty niin, ettei niissä esiinny liikaa väreilyä ja että ne ovat mahdollisimman kevyitä. Liiallisen muistin käytön takia kuvat poistetaan latauksen jälkeen tietokoneen keskusmuistista, jolloin niitä ei enää asettamisen jälkeen pysty muokkaamaan.

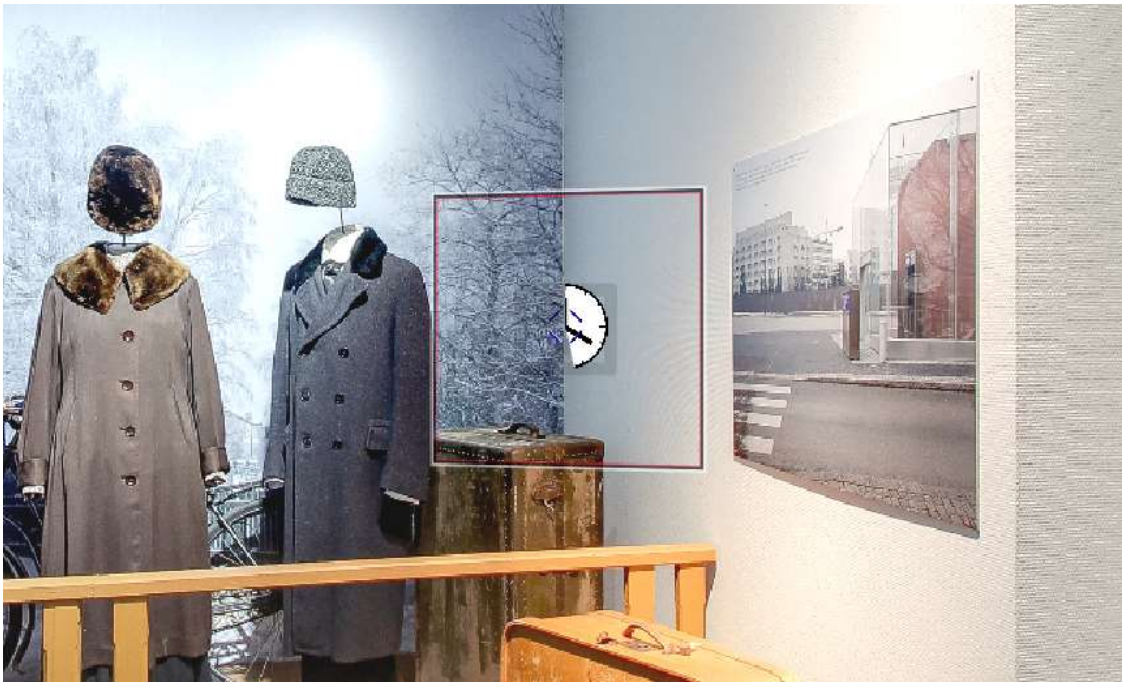
Kentän peliobjektien luonnin jälkeen sovellus siirtyy lataamaan kentän omia aloitus- ja taustaääniä. Kentän omista asetuksista riippuen pelaajan on joko kuunneltava alkupuhe tai ei.

5.4 UX

Pelin interaktiivisuus suunniteltiin mahdollisimman yksinkertaiseksi kohdeyleisöä ajatellen. Peli oli tarkoitettu alun perin ikääntyneemmille ihmisille, jolloin monimutkaisia ohjaimia ei haluttu käyttää. Kenttien tekstien määrä pidettiin maltillisena ja puhetta kenttiin lisättiin, jolloin kokemuksen seuraaminen on miellyttävämpää, kuten kuviossa 29 huomataan.

KUVIO 29. Karusellielementti

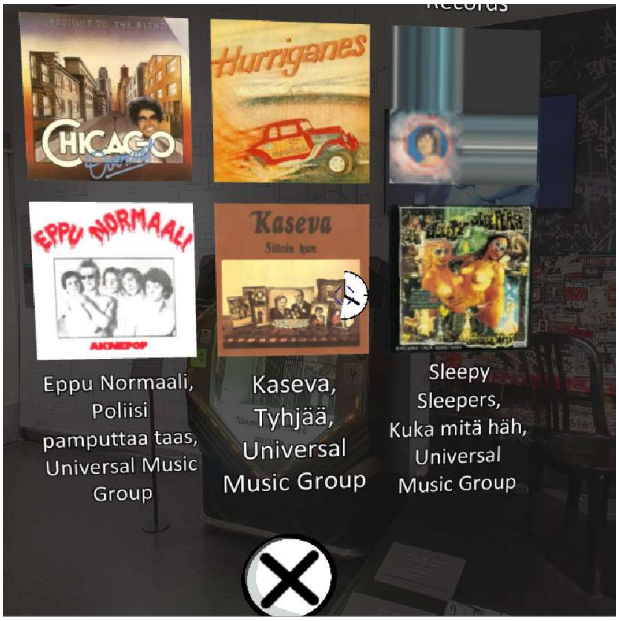
Käyttöliittymä on toteutettu sovellukseen käyttämällä Unityn UI kirjastoa. Kirjaston avulla jokaiselle peliobjektille pystyttiin luomaan tarvittava piirtoaluekomponentti, joka näkyy kolmiulotteisena näkymässä. Painikkeiden sijasta peliobjektin interaktiiviset alueet on toteutettu käyttämällä kuvakomponenttia sekä asettamalla törmäytinkomponentti, jonka ansiosta Raycaster pystyy sen havaitsemaan. Tämän ansiosta ohjelmistossa ei ollut tarvetta Oculus Riftin ohjaimille, vaan interaktiivisuus toteutettiin määrittelemällä Raycastin lähtöpiste keskeltä näyttöä suoraan eteenpäin, jolloin koodilla pystytään aktivoimaan interaktiivisia komponentteja. Visuaaliseksi lisäksi avautumiselle asetettiin kahden sekunnin viive ja esille tuleva latausmittari, joka on toteutettu käyttäen täytettyä kuvakomponenttia (Kuvio 30).



KUVIO 30. Katseenkohdistus

Peliobjekteilla oli yleisesti kaksi tilannetta, joissa niiden kanssa voidaan vuorovaikuttaa: silloin kun ne avataan ja silloin kun ne suljetaan. Näille molemmille toteutettiin funktiot sekä delegaatit. Kaikki näkymässä olevat interaktiiviset peliobjektit on toteutettu käyttämällä samaa abstraktia luokkaa, jolloin niiden toiminnallisuudet ovat samanlaiset. Ainoastaan jukeboksina toimivassa elementissä pystyy vuorovaikuttamaan levyjen kansien

kanssa, jolloin niistä lähtee soimaan musiikkia (Kuvio 31).



KUVIO 31. Jukeboxi

5.5 Äänien toistaminen

Ohjelmistoon tuli tarve toteuttaa useammalle äänelle tuki, joten se toteutettiin käyttäen pooling-menetelmää. Siinä toteutettiin 25 kappaletta peliobjekteja, joissa on äänilähde asetettuna. Näitä peliobjekteja hallitsee yksi isäpeliobjekti, jonka avulla pystytään tarvittaessa kutsumaan vapaata äänilähdettä vaihtamaan paikkaa määriteltyyn pisteeseen. Kun vapaa äänilähde vaihtaa paikkaa, niin sen äänileike vaihdetaan avautuneen elementin äänileikkeen mukaiseksi.

5.6 Videon toistaminen

Sovellukseen toteutettiin myös mahdollisuus näyttää tarvittaessa videoita. Peliobjektin avautuessa avataan laaja elementti, jonka materiaaliin video heijastetaan. Kuviossa 32 on heijastettuna videosoitinkomponentin video isomman neliskulmaisen suorakulmion materiaaliin, jolloin video saadaan näkyviin. Videosoitinkomponentin ansiosta isoja mp4-tiedostoja ei tarvitse kokonaan ladata muistiin, vaan ne pystytään näyttämään suoratoiston avulla.

KUVIO 32. Video esitettynä peliobjektin materiaalissa.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa Lahden historialliselle museolle ohjelmisto, jonka avulla museon henkilökunta pystyy toteuttamaan omia näytelmiä virtuaalisesti ja kuljettamaan niitä mukanaan. Ohjelmiston mukana toteutettiin museon käsikirjoittama virtuaalinen näyttely heidän edeltävästä näyttelystään, 100 askelta Lahdessa, jonka avulla tarvittavat komponentit toteutettiin. Museo tarjosi kaikki tarvittavat kuvat, äänet, videot ja tekstit ohjelmiston sisällön tuottamista varten.

Virtuaalisen näyttelyn toteuttamiseksi piti ymmärtää, miten Unity toimii, miten ohjelmoidaan c# ohjelmointikielellä ja miten Oculus Riftin sai yhdistettyä Unityyn. Unity omaa itsessään kattavan dokumentaation verkossa, ja sen lisäksi Unitylla on erittäin aktiivinen foorumi, jonka ansiosta kohdattuihin ongelmiin sai helposti ratkaisun. Oculus Riftin ohjelmointirajapinnan sai taas ladattua helposti Oculuksen verkkosivuilta tai vaihtoehtoisesti Unityn omasta kaupasta.

Oculus Rift ei vaatinut suurempia säädöksiä toimiakseen. Unity itseasiassa tunnisti Oculus Riftin virtuaalilasien näytön jo ennen kuin Oculuksen rajapinta oli Unityyn ladattu. Ilman kyseistä rajapintaa, säädöt olisi pitänyt itse ohjelmoida laseihin. Ohjelmointirajapinnan jälkeen tarvitsi vain vetää yksi valmis elementti kansiosta pelikenttään, jonka jälkeen päästiinkin tekemään pelialuetta. Tällöin pään liikkeitä pystyttiin seuraamaan ja säätämään Oculuksen näytölle näytettävän kuvan asetuksia. Mitään suurempia bugeja tai ongelmia ei ilmestynyt, vaan ainoastaan kuvien kanssa piti olla tarkkana, että ladatut kuvat poistettiin keskusmuistista.

Tällaiset virtuaaliset elämykset vaikuttavat lupaavilta ja niissä on varaa suurempiikin parannuksiin. Jos sovellukseen kehitettäisiin paikkakohtainen haku verkon kautta ja kuvat saataisiin ladattua esimerkiksi tietokannasta, niin elämys voisi parantua huomattavasti. Virtuaalista näyttelyä toteuttaessa tuli esille ajatus, jos kyseisen ohjelmiston pystyisi toteuttamaan mobiililaitteille ja niille tarkoitetuille virtuaalilaseille. Tällöin yhä useampi pystyisi katselemaan virtuaalista näyttelyä omassa kodissaan.

LÄHTEET

Alex 2018. Oculus Rift history - How it all started [viitattu 5.3.2019]. Saatavissa: https://riftinfo.com/oculus-rift-history-how-it-all-started

Brodkin, J. 2013. How Unity3D became a gamedevelopment beast [viitattu 5.3.2019]. Saatavissa: https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

Crockford, D. 2019. Introducting JSON [viitattu 23.3.2019]. Saatavissa: https://www.json.org/

Diversified Internet Holdings LLC. 2019. What is virtual reality (VR)? Ultimate guide to virtual reality (VR) [viitattu 7.3.2019]. Saatavissa: https://www.realitytechnologies.com/virtual-reality/

Facebook Technologies, LLC. 2019a. Oculus utilities for Unity [viitattu 22.3.2019]. Saatavissa: https://developer.oculus.com/documentation/unity/latest/concepts/unity-utilities-overview/#unity-utilities-overview

Facebook Technologies, LLC. 2019b. Turn your ideas into reality [viitattu 22.3.2019]. Saatavissa: https://developer.oculus.com/

Garland, A. 2016. How Virtual Reality Will Shape the Future of Gaming [viitattu 7.3.2019]. Saatavissa: https://tweakyourbiz.com/technology/virtual-reality-will-shape-future-gaming

Laukkonen, J. 2018. What is Oculus Touch [viitattu 15.3.2019]. Saatavissa: https://www.lifewire.com/oculus-touch-4159174

McCarthy, J. 2017. Did Pokemon Go really change how marketers view augmented reality [viitattu 8.3.2019]. Saatavissa: https://www.thedrum.com/news/2017/07/07/did-pokemon-go-really-change-how-marketers-view-augmented-reality

Microsoft 2019a. IEnumerator<T> Interface [viitattu 21.3.2019]. Saatavissa: https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerator-1?view=netframework-4.7.2

Microsoft 2019b. yield (C# Reference) [viitattu 21.3.2019]. Saatavissa: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/yield

Nield, D. 2016. How Oculus Rift works: Everything you need to know about the VR sensation [viitattu 9.3.2019]. Saatavissa: https://www.wareable.com/vr/how-oculus-rift-works

OneBonsai 2018. Looking for a Oculus Rift for rent? We rent all Virtual Reality headsets available [viitattu 5.3.2019]. Saatavissa: https://onebonsai.com/services/rent-virtual-reality-headset/rent-oculus-rift/

Spence, C., Obrist, M., Velasco, C. 2017. Digitizing the chemical senses: Possibilities & pitfalls [viitattu 6.3.2019]. Saatavissa: https://www.researchgate.net/publication/317640892_Digitizing_the_chemical_senses_Possibilities_pitfalls

Souppouris, A. 2016. How HTC and Valve built the Vive [viitattu 6.3.2019]. Saatavissa: https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/?guccounter=1&guce_referrer_us=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_cs=FloPXg0WESsb-uaC5X0O6w

Tokareva, J. 2018. The difference between virtual reality, augmented reality and mixed reality [viitattu 3.4.2019]. Saatavissa: https://www.forbes.com/sites/quora/2018/02/02/the-difference-between-virtual-reality-augmented-reality-and-mixed-reality/#2c500b992d07

Unity Technologies 2019a. Basic Layout [viitattu 4.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/UIBasicLayout.html

Unity Technologies 2019b. Camera.ViewportPointToRay [viitattu 4.4.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/Camera.ViewportPointToRay.html

Unity Technologies 2019c. Canvas [viitattu 5.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/UICanvas.html

Unity Technologies 2019d. Colliders [viitattu 23.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/CollidersOverview.html

Unity Technologies 2019e. Creating and using scripts [viitattu 23.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

Unity Technologies 2019f. Game engines - how do they work [viitattu 1.4.2019]. Saatavissa: https://unity3d.com/what-is-a-game-engine

Unity Technologies 2019g. GameObject [viitattu 3.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/class-GameObject.html

Unity Technologies 2019h. Image [viitattu 8.4.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/UI.Image.html

Unity Technologies 2019i. MonoBehaviour [viitattu 7.4.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

Unity Technologies 2019j. MonoBehaviour.Awake() [viitattu 6.4.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html

Unity Technologies 2019k. MonoBehaviour.Start() [viitattu 6.4.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html

Unity Technologies 2019l. MonoBehaviour.Update() [viitattu 6.4.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html

Unity Technologies 2019m. Physics [viitattu 25.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/PhysicsSection.html

Unity Technologies 2019n. Prefabs [viitattu 6.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/Prefabs.html

Unity Technologies 2019o. Products [viitattu 5.3.2019]. Saatavissa: https://unity3d.com/unity

Unity Technologies 2019p. RaycastHit [viitattu 9.3.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/RaycastHit.html

Unity Technologies 2013. Raycasting - Unity Official Tutorials [viitattu 9.3.2019]. Saatavissa: https://www.youtube.com/watch?time_continue=3&v=EINgloTG8D4

Unity Technologies 2019q. Rays from the Camera [viitattu 9.3.2019]. Saatavissa: https://docs.unity3d.com/Manual/CameraRays.html

Unity Technologies 2019r. Scripting [viitattu 2.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/ScriptingSection.html?_ga=2.150289461.2092785172.1553015985-32365663.1534174112

Unity Technologies 2019s. Text [viitattu 17.3.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/UI.Text.html

Unity Technologies 2019t. The hierarchy window [viitattu 3.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/Hierarchy.html

Unity Technologies 2019u. Video Player component [viitattu 5.4.2019]. Saatavissa: https://docs.unity3d.com/Manual/class-VideoPlayer.html

Unity Technologies 2019v. WWW [viitattu 17.3.2019]. Saatavissa: https://docs.unity3d.com/ScriptReference/WWW.html

Unity Wiki 2017. SimpleJSON [viitattu 18.3.2019]. Saatavissa: https://wiki.unity3d.com/index.php/SimpleJSON

Virtual Reality Society 2017. History of virtual reality [viitattu 7.3.2019]. Saatavissa: https://www.vrs.org.uk/virtual-reality/history.html