VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jani Niemistö

3D-pelin toteuttaminen Unity-kehitysympäristössä

Tekniikka
2018

VAASAN AMMATTIKORKEAKOULU
Insinööri (AMK), tietotekniikka

TIIVISTELMÄ

Tekijä	Jani Niemistö
Opinnäytetyön nimi	3D-pelin toteuttaminen Unity-kehitysympäristössä
Vuosi	2018
Kieli	suomi
Sivumäärä	38 + 2 liitettä
Ohjaaja	Timo Kankaanpää

Tämän opinnäytetyön tarkoitus oli tutustua 3D-pelikehitykseen aloittelevan pelikehittäjän näkökulmasta. Kehitysympäristöksi valittiin Unity, jota on käytetty pelimoottorina muun muassa Pokémon Go ja Angry Birds 2 -peleissä. Opinnäytetyön tehtävä on oppia ja opettaa pelikehityksen eri tekniikoita, sekä työkaluja aloittelijoille.

Käydään läpi käytetyt tekniikat ja pelikehityksen eri vaiheet, jotka ovat: suunnittelu, peliympäristön luominen, objektien animointi, skriptien ohjelmointi ja lopuksi pelin kääntäminen. Tietoa tähän projektiin on haettu Internet-sivustoilta, keskusteluista, videokursseilta ja kokeilun kautta.

Pelin toteuttaminen on hyvä tapa avata ovia pelialan yrityksiin tai saada oppia oman indie-pelistudion aloittamiseen, joten toteutettiin projekti, jossa luotiin tosimaailman ympäristöön perustuva 3D-peli nimeltä Burning Island Auroras. Todettiin, että pelikehitys Unitylla on todella suoraviivaista ja kätevää, mutta kokonaisen 3D-pelin tuottaminen vaatii aina runsaasti työtä ja suuren työryhmän, jotta peli valmistuu kohtuullisessa ajassa.

Avainsanat Pelit, pelikehitys, 3D-mallinnus, ohjelmointi, animointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Bachelor of Engineering, Information Technology

ABSTRACT

Author	Jani Niemistö
Title	Game Development with Unity 3D
Year	2018
Language	Finnish
Pages	38 + 2 Appendices
Name of Supervisor	Timo Kankaanpää

The background of this thesis was to become familiar with 3D game development as beginner level game developer. Game development platform in this thesis was chosen to be Unity, which was used to create games such as Pokémon Go and Angry Birds 2. Thesis is supposed to teach different methods and tools of game development to the people who are interested in the industry.

We will get familiar with used methods and different phases of game development. The phases are: planning, game environment developing, animating, scripting and building the game. Information sources in this project have been web pages, discussions, video tutorials and learning through experiments.

Development of a game is a good way to open doors to the game industry or to get experience to start your own indie game studio. During this thesis we created 3D game named Burning Island Auroras with real-world based environment. As conclusion we noticed that working with Unity is very handy and straightforward, but development of a full game always requires a lot of work and a big team.

Keywords Games, game development, 3D-simulation, programming, animating

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

TERMIT JA LYHENTEET

Blender	Ilmainen ja vapaa 3D-mallinnusohjelma.
Boo	Oliopohjainen ohjelmointikieli, joka ei ole enää käytössä Unityssa.
C#	Unityssa käytettävä oliopohjainen ohjelmointikieli.
Microsoft Visual Studio	Ohjelmankehitysympäristö, jossa voi käyttää useita eri ohjelmointikieliä.
MonoDevelop	Unityn oletuksena asentama ohjelmointiympäristö.
Burning Island Auroras	Unitylla tässä työssä kehitetty 3D-peli.
Peliobjekti	Kaikki pelissä näkyvät ja kuuluvat osat.
PhysX	Nvidian omistama reaaliaikainen fysiikkamoottori.
Prefab	Peliobjekti, joka voi sisältää eri komponentteja ja ominaisuuksia.
Tekstuuri	Kuvatiedosto, joka voidaan liittää Unityn pelimaailmaan.
UnityScript	Unityssa käytettävä ohjelmointikieli, jota usein kutsutaan nimellä JavaScript, johon se pohjautuu.

1 JOHDANTO

Opinnäytetyön aiheeksi valittiin 3D-pelin toteuttaminen Unity-kehitysympäristössä, koska peliala on vieläkin kehittyvä ja opinnäytetyön tekijää kiinnostaa pelikehitys tulevana ammattina. Peliala tarjoaa monenlaisia työtehtäviä, muun muassa grafiikan luomista, joka on opinnäytetyön tekijälle lähellä sydäntä. Opinnäytetyössä lähestymistapana pelialaan on aloittelijan tasolla oleva pelikehittäjä.

Projektin alussa pelillä ei vielä ollut tyylilajia ja päätettiin, että luodaan ensin pelimaailma. Pelimaailmaksi päätettiin tehdä Palosaaren Vetokankaan puoli, joka olisi kooltaan 500x500 metriä. Opinnäytetyön tekijä ajatteli, että tämä edesauttaa projektissa etenemistä ulkoilun kautta, sillä ulkona oli kätevä käydä katselemassa yksityiskohtia peliin. Peliin on otettu myös kuvia oikean maailman rakennuksista, jotka on liitetty pelimaailmaan realistisuuden luomiseksi.

Alusta asti oli selvää, että peli tulee toimimaan ensimmäisestä persoonasta, eli 3D-pelimaailma esitetään pelihahmon näkökulmasta nähtynä. Peliä tehdessä myös tyylilaji alkoi hahmottumaan ja visio oli, että pelissä näkyy kuvitteellisen ytimen tila, joka laskee hitaasti pelin edetessä ja tilaa pystyy nostamaan korjaavilla toimenpiteillä, esimerkiksi sähköverkon korjaamisella. Peli keskittyisi korjaamiseen, eikä tuhoamiseen, kuten suuri osa nykyisistä peleistä.

Pelin nimeä ei oltu vielä alkuun mietitty vaan keskityttiin itse pelin tekemiseen. Projektin aikana kävi muutama vaihtoehto mielessä ja pelin nimeksi tuli lopulta Burning Island Auroras.

2 KÄYTETYT TEKNIIKAT

Tämä luku kertoo projektin aikana käytetyistä tekniikoista, johon kuuluu muun muassa ohjelmointikieli, kehitys- ja ohjelmointiympäristö sekä niiden valintaprosessit. Käydään läpi Unityn tärkeimmät ominaisuudet ja miksi valittiin juuri Unity tässä työssä käytettäväksi kehitysympäristöksi. Pelikehityksessä suuri osa pelin luomista on skriptien ohjelmointi, joten paneudutaan myös ohjelmointikielen ja -ympäristön valintaan.

2.1 Kehitysympäristön valinta

Pelikehityksen yksi tärkeimpiä kysymyksiä on kehitysympäristön valinta, joten se täytyy tehdä heti projektin alussa. Tässä opinnäytetyössä punnittiin valintaa Unreal Enginen ja Unityn välillä, mutta vertailujen ja arvostelujen perusteella päätös oli helposti tehty.

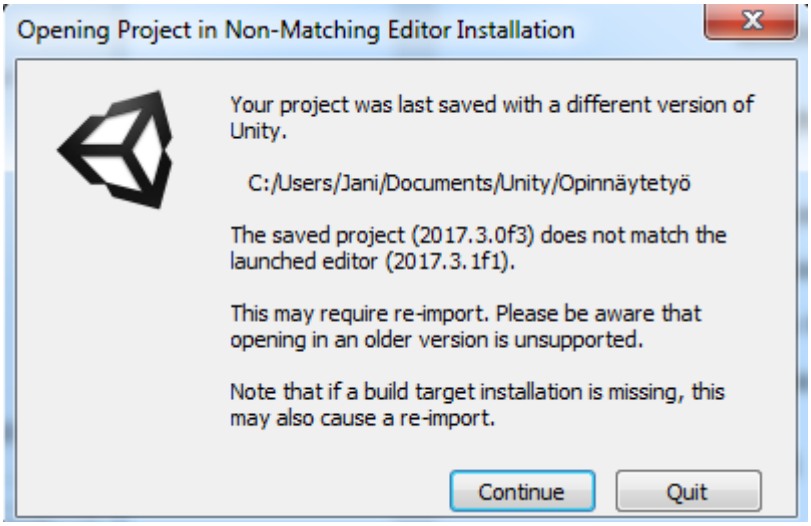
Vertailuun hyvä apu oli Samuli Hannukselan opinnäytetyö ”Unity 5- ja Unreal Engine 4 -pelimoottorien vertailu”, jossa kahta suosituinta pelimoottoria esiteltiin todella kattavasti. Pelimoottorit arvosteltiin karkeasti samanveroisiksi ja loppupäätelmä oli:

> ”Pelimoottorien työkalut ja ominaisuudet mukaan lukien Unity 5 sopii paremmin vähän resursseja omaaviin laitealustoihin ja fysiikkaan pohjautuviin sovelluksiin. Unreal Engine 4 sopii paremmin graafisesti vaativiin sovelluksiin.” (Hannuksela, 2016)

Täyden varmuuden Unityn valintaan tuli työvoimatoimiston järjestämästä vapaaehtoisesta MessiLive-keskustelusta. Keskustelussa haastateltiin Neogames Finland yrityksen ohjaajaa KooPee Hiltusta, Parta Games yrityksen perustajaa Antti Kolehmaista ja Kajaanin ammattikorkeakoulun pelikehityksen opettajaa Antti Seppästä.

Opettaja Antti Seppänen sanoi haastattelussa, että jos haluaa päästä sisään pelialalle, niin kannattaa luoda peli käyttäen Unity-pelimoottoria ja C#-ohjelmointikieltä. Tämän jälkeen Unity-kehitysympäristön valinta oli varma ja alettiin pelin luomisprosessi sillä.

Tässä opinnäytetyössä on käytössä tällä hetkellä uusin Unityn vakaa versio Unity 2017.3, joka ilmestyi joulukuussa 2017. Versio 2017.3.0 julkaistiin sopivasti juuri ennen kuin pelin luominen aloitettiin, joten päästiin testaamaan ja harjoittelemaan uunituoretta versiota. Helmikuussa 2018 julkaistiin taas uusi päivitys 2017.3.1, jota käytettiin työssä tuon ajankohdan jälkeen.



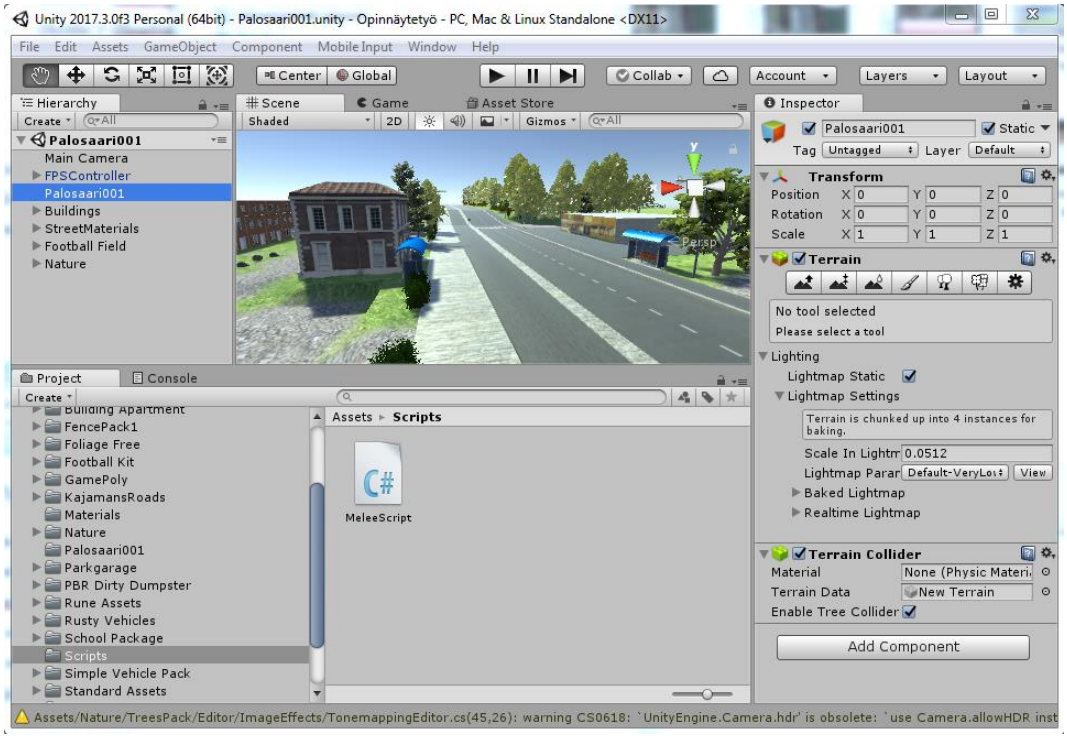
Kuvio 1. Unityn päivitys vaatii projektin uudelleentuomisen (re-import).

2.2 Unityn yleiskatsaus

Unityn on kehittänyt Tanskasta lähtöisin oleva Unity Technologies, joka on perustettu vuonna 2004. Unity Technologies yrityksen tavoite on tehdä pelikehityksestä mahdollisimman kivutonta ja tuoda yhteen maailman parhaat pelikehittäjät. Unity keskittyy kehityksessään tällä hetkellä käytettävyyden laajentamiseen, tehon parantamiseen ja uusien pelialustojen tukemiseen.

Unitylla voidaan kehittää sekä kolmi-, että kaksiulotteisia videopelejä ja simulaatioita. Sillä on kehitetty muun muassa suomalainen Angry Birds 2 (Rovio) ja huippusuosion saanut Pokémon Go (Nintendo). Unitysta on saatavilla neljä eri lisenssiä, jotka ovat: Personal, Plus, Pro ja Enterprise. Tämän opinnäytetyön 3D-pelin toteutukseen soveltui vielä tässä vaiheessa ilmainen Personal lisenssi, jolla voi kehittää pelejä 100 000 dollarin vuosittaisen tuoton edestä.

Unityn grafiikkapuolesta vastaa Nvidian PhysX fysiikkamoottori ja Unity tukee todella kattavasti yhteensä 27 eri alustaa: iOS, Android, Tizen, Windows, Universal Windows Platform, Mac, Linux, WebGL, PlayStation 4, PlayStation Vita, Xbox One, Wii U, 3DS, Oculus Rift, Google Cardboard, Steam VR, Playstation VR, Gear VR, Windows Mixed Reality, Daydream, Android TV, Samsung Smart TV, tvOS, Nintendo Switch, Fire OS, Facebook Gameroom, Apple ARKit, Google ARCore, and Vuforia (Wikipedia, 2018).



Kuvio 2. Unity-kehitysympäristö, jossa valittuna Terrain-objekti.

2.3 Ohjelmointikielen valinta

Yksi tärkeä osa Unitya on skriptien ohjelmointi ja liittäminen pelimaailmaan. Skripteillä voidaan luoda eri toimintoja, muun muassa pyöriviä tai kerättäviä objekteja, suoritettavia tehtäviä pelaajalle tai esimerkiksi useista peleistä tutun kuntopalkin. Unityssa annetaan käyttäjälle mahdollisuus valita ohjelmointikieleksi joko JavaScript (UnityScript) tai C#. Näillä kielillä ohjataan mono nimistä pelilogiikkaa, joka toimii .NET alustalla.

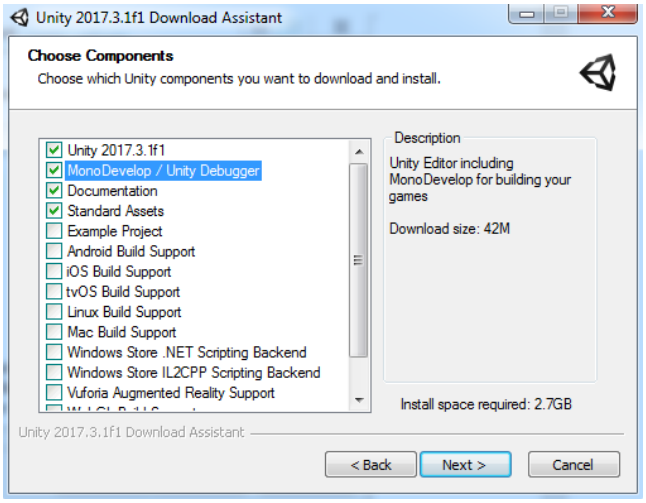
Aiemmin oli käytettävissä myös hieman vieraampi Boo, joka Wikipedian suomenkielisen Unity-artikkelin vanhentuneen tiedon mukaan olisi vielä käytössä. Englanninkielisen Boo-artikkelin alta löytyi kuitenkin tieto, että Boo tippui pois Unitysta vuonna 2014 pienen käyttäjämäärän takia. Boo olisi ollut varteenotettava vaihtoehto ohjelmointiin, sillä se perustuu Pythoniin, jonka syntaksi on todella mielekästä lukea ja kirjoittaa.

Unityn teoriaa on hyvä opiskella katsomalla YouTuben Jimmy Vegas kanavalta kursseja, joissa käytetään ohjelmointikielenä perinteisempää JavaScriptiä. Näissä videoissa kuitenkin puhuttiin, että JavaScript vain esitellään alkuun ja myöhemmässä vaiheessa käytetään C#-ohjelmointikieltä, jolla käytännössä myös melkein koko Unity-kehitysympäristö on rakennettu (Jimmy Vegas Unity Tutorials, 2018). Aloittelijoille JavaScript on anteeksiantavaisempi virheiden sattuessa, mutta se on myös rajoittuneempi, kuin C#.

Tässä projektissa valittiin käytettäväksi C#-ohjelmointikieli, koska Boo ei ole enää käytössä ja C# on hyvin kytköksissä Unityyn. C# on myös uudempi, kuin JavaScript ja opinnäytetyön tekijä on käyttänyt sitä huomattavasti enemmän viime aikoina.

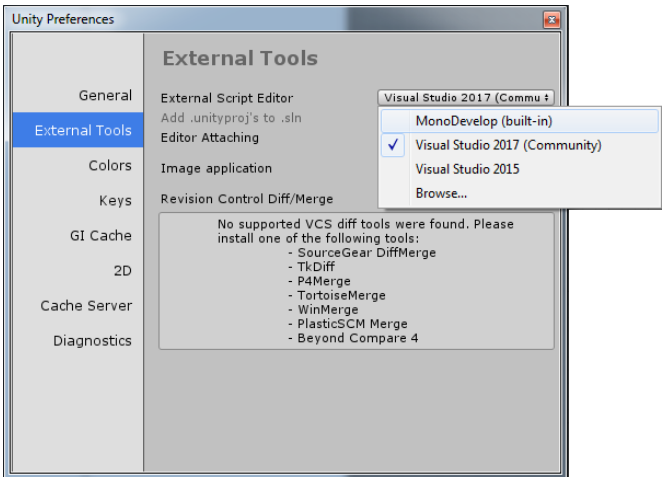
2.4 Ohjelmointiympäristön valinta

MonoDevelop on ohjelmointiympäristö, joka asennetaan oletuksena samalla kun asentaa Unityn. Se voidaan kuitenkin halutessa jättää asentamatta ja käyttää ohjelmointiympäristönä esimerkiksi Microsoftin Visual Studiota.



Kuvio 3. Unity asentaa oletuksena MonoDevelop-ohjelmointiympäristön.

Kummatkin ohjelmointiympäristöt ovat varteenotettavia vaihtoehtoja, joten niitä tuli vertailla toisiinsa. Opinnäytetyön tekijällä oli koneella asennettuna Microsoftin Visual Studio 2017 (Community) ja Unityn mukana asennettu MonoDevelop 5.9.6. Käytettävä ohjelmointiympäristö voidaan vaihtaa Unityn asetuksista External Tools-välilehdeltä (Kuvio 4). Vertailu aloitettiin käynnistymisnopeudesta, joten tehtiin testi, jossa avattiin sama skriptitiedosto kummallakin ohjelmointiympäristöllä. MonoDevelop käynnistyi hieman 5,2 sekunnissa ja Microsoftin Visual Studio 8,2 sekunnissa.



Kuvio 4. Käytettävä ohjelmointiympäristö valitaan External Tools-välilehdeltä.

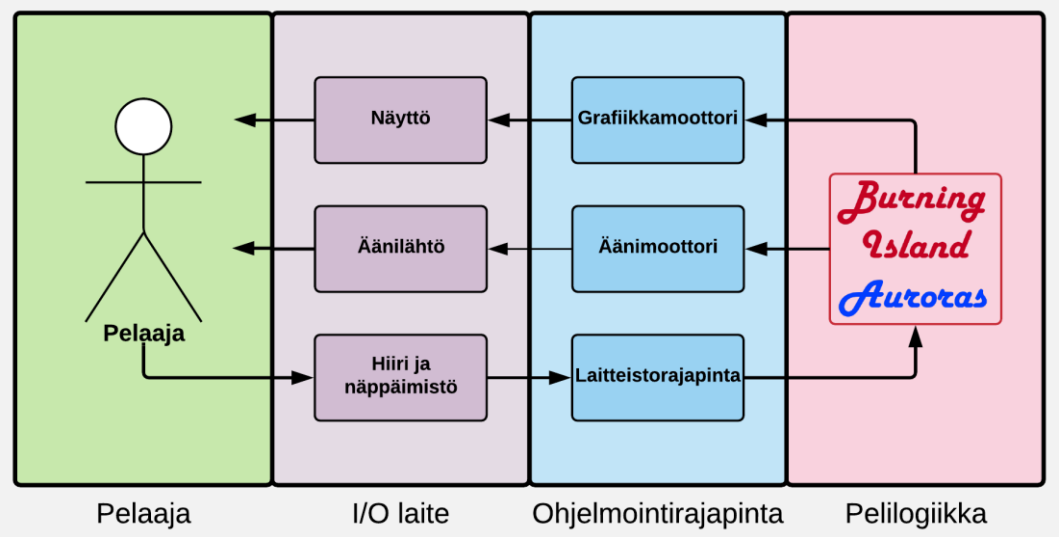
Valintaan vaikutti myös se, että AVG AntiVirus Free 18.3.3051 -tietoturvaohjelmisto näkee MonoDevelopin haittaohjelmana. Unityn keskustelupalstalla kirjoitetaan, että kyseessä on väärä hälytys, mutta valittiin silti ohjelmointiympäristöksi Microsoftin Visual Studio, koska ympäristö on tuttu opinnäytetyön tekijälle entuudestaan, joten päätettiin säästää aikaa MonoDevelopin opiskelusta, jotta saadaan kehitettyä peliä pidemmälle. MonoDevelop olisi säästänyt aikaa käynnistymisessä, mutta ero oli vain 3,2 sekuntia, joten ympäristön opiskeluun olisi kulunut kuitenkin enemmän aikaa.

3 3D-PELIN SUUNNITTELU

Tämä kappale koostuu pelin suunnitteluun kuuluvista osista, johon lukeutuu pelin arkkitehtuuri, peliympäristön suunnittelu ja Burning Island Auroras -pelin hierarkia.

3.1 Arkkitehtuuri

Pelien arkkitehtuuri on hyvin samanlainen ohjelmistojen arkkitehtuurin kanssa, mutta peleissä käytetään hieman erilaisia ohjelmointirajapintoja. Alla olevassa kuvassa on esiteltynä Burning Island Auroras -pelin komponentit arkkitehtuurikaaviossa:



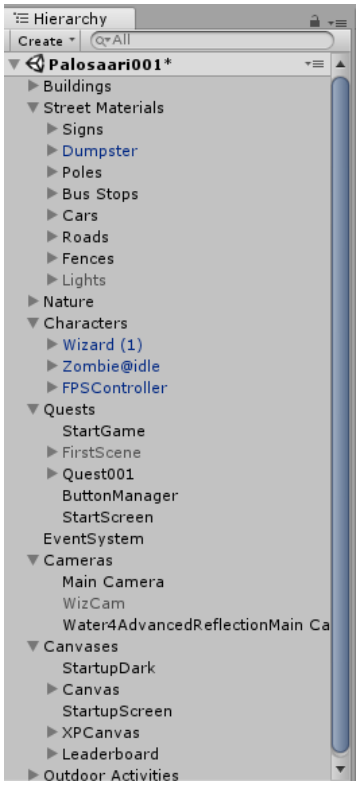
Kuvio 5. Burning Island Auroras -pelin arkkitehtuuri.

Pelaaja käyttää hiirtä ja näppäimistöä pelin ohjaamiseen laitteistorajapinnan kautta. Burning Island Auroras kehitettiin Windows- ja Linux-alustoille, joten laitteistorajapintana toimii Microsoftin kehittämä DirectX, joka toimii siltana ohjelmiston ja laitteiston välillä.

Pelilogiikka taas tuo Unityn grafiikka- ja äänimoottorin avulla kuvan ja peliäänet pelaajan näytölle ja kaiuttimiin. Pelaaja ja I/O -laitteet ovat siis fyysisiä, ohjelmointirajapinta toimii käyttöjärjestelmän avulla ja pelilogiikka taas Unityn pelimoottorilla.

3.2 Hierarkia

Pelikehityksessä on todella tärkeää käyttää alusta asti hyvin suunniteltua hierarkiaa objekteille. Tämän projektin aikana peliin tuli yli tuhat objektia, joten jos niitä ei ole järjestetty hyvin, niin yksittäisen objektin etsiminen on työlästä ja aikaa vievää. Hierarkia on vakiona Unityn vasemmassa reunassa ja sitä voidaan muokata vapaasti omiin tarpeisiin sopivaksi. Seuraavassa kuvassa näkyy tämän projektin hierarkia Unityssa.



Kuvio 6. Unityn hierarkiaikkuna.

Hierarkiaan voidaan lisätä otsikoita painamalla hiiren oikeaa ja valitsemalla ”Create Empty”, jonka jälkeen voidaan nimetä otsikko halutuksi. Tässä projektissa hierarkia on jaoteltu seuraavasti: rakennukset, liikenne, luonto, pelihahmot, tehtävät, kamerat, kanvaasit ja ulkoaktiviteetit. Dokumentin lopussa on liitteenä hierarkiakaavio, josta jaottelun havainnollistaa paremmin (Liite 2).

4 3D-PELIN TOTEUTUS

Tässä luvussa kerrotaan esimerkkiprojektin, eli 3D-pelin nimeltä Burning Island Auroras luomisesta. Käydään läpi muun muassa peliympäristön, skriptien ja animaatioiden luominen projektiin.

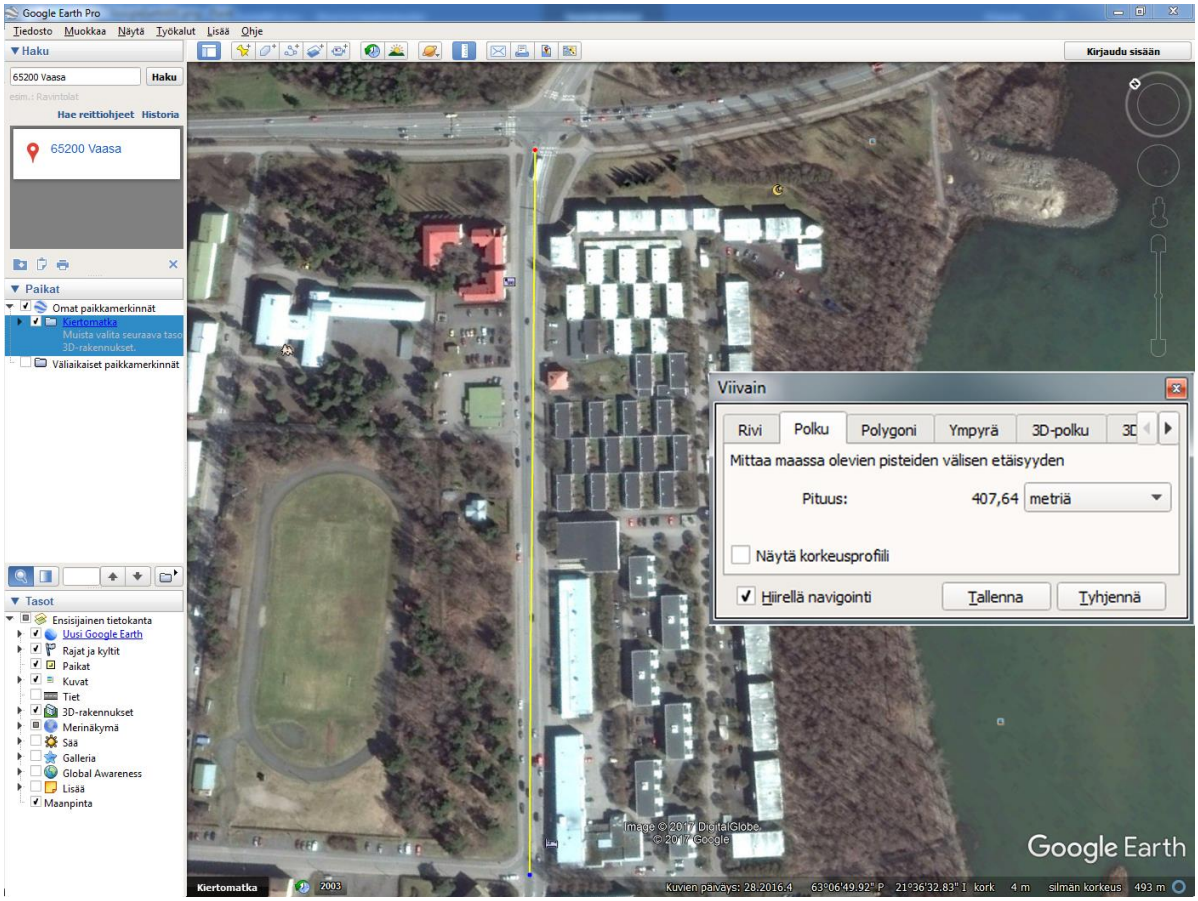
4.1 Peliympäristön luominen

Peliympäristön luominen piti aloittaa suunnittelemalla kartta, jolla pelaaja pystyy liikkua. Päätettiin luoda ympäristö pohjautuen paikkaan, jossa tämän projektin aikana opinnäytetyön tekijä asui, eli Vaasassa sijaitsevan Palosaaren Vetokankaan puoleen. Tällä tavoin opinnäytetyön tekijä pystyi luomaan ympäristöä mahdollisimman realistisesti menemällä kävelylle ulos. Vuodenaika on koko pelin ajan kesä.

4.1.1 Googlen palveluiden hyödyntäminen

Ympäristön luomiseen käytettiin paljon Google Maps ja Google Earth -palveluita. Google Mapsin Street View -työkalua on hyvä käyttää pelimaailman yksityiskohtien mallintamiseen. Sillä voidaan nähdä teiden varsissa olevat rakennukset, kyltit, puskat, aidat jne. Tässä projektissa pelimaailman yksityiskohtien havainnollistaminen onnistui myös kävelemällä ulos katsomaan.

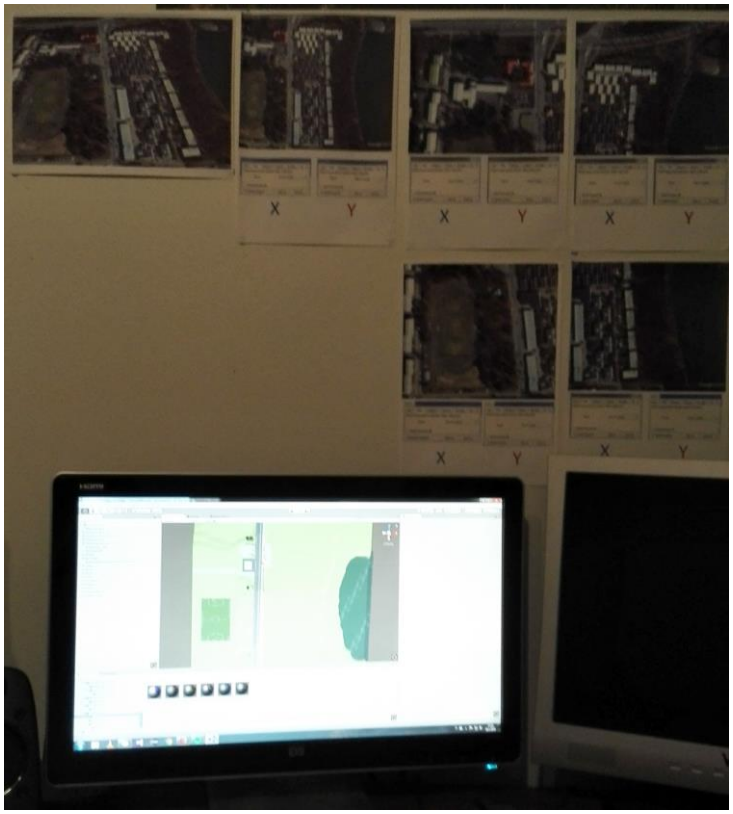
Google Earthia voidaan käyttää peliympäristön mittaamiseen ja korkeuserojen havainnollistamiseen. Google Earth Pro on ladattavissa ilmaiseksi Googlen palvelimilta ja tässä työssä oli käytössä versio 7.3.1.4507. Peliympäristön kooksi päätettiin Google Earthin viivaintyökalun avulla 500 x 500 metriä.



Kuvio 7. Google Earthin viivaintyökalun käyttäminen kartan mittaamiseen.

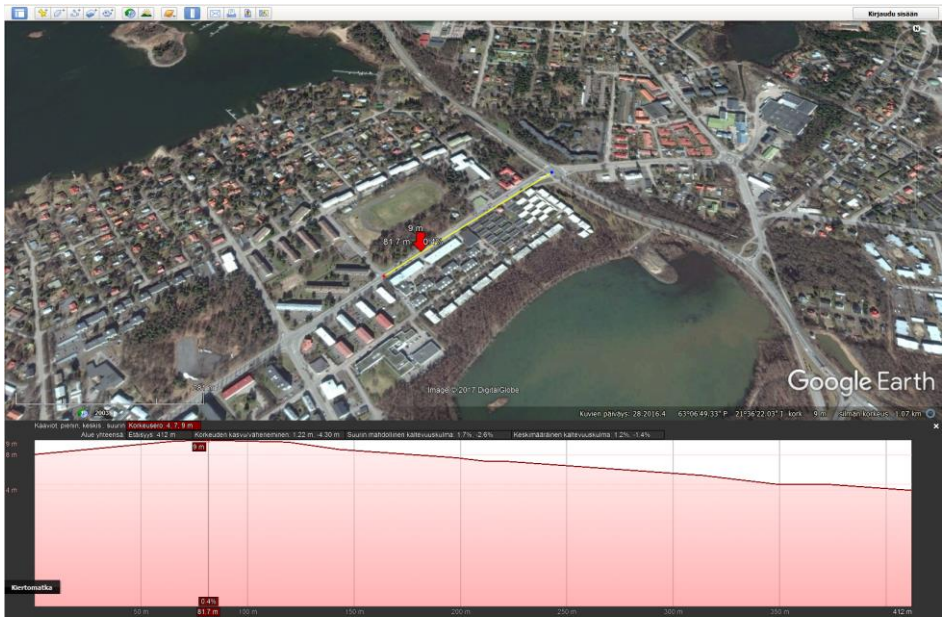
Peliympäristön koon ja sijainnin määrittämisen jälkeen alettiin paneutua alueen korkeuseroihin. Google Earthin viivaintyökalu osoittautui tässäkin tehtävässä todella käteväksi.

Valittiin viivaintyökalulla päätepisteet luotavasta pelimaailmasta ja Kuvio 7:ssa nähtävästä viivaintyökalun asetusikkunasta ”Näytä korkeusprofiili”. Tämä valinta avaa näkymän, joka on esillä havainnollistavassa kuvassa (Kuvio 9).



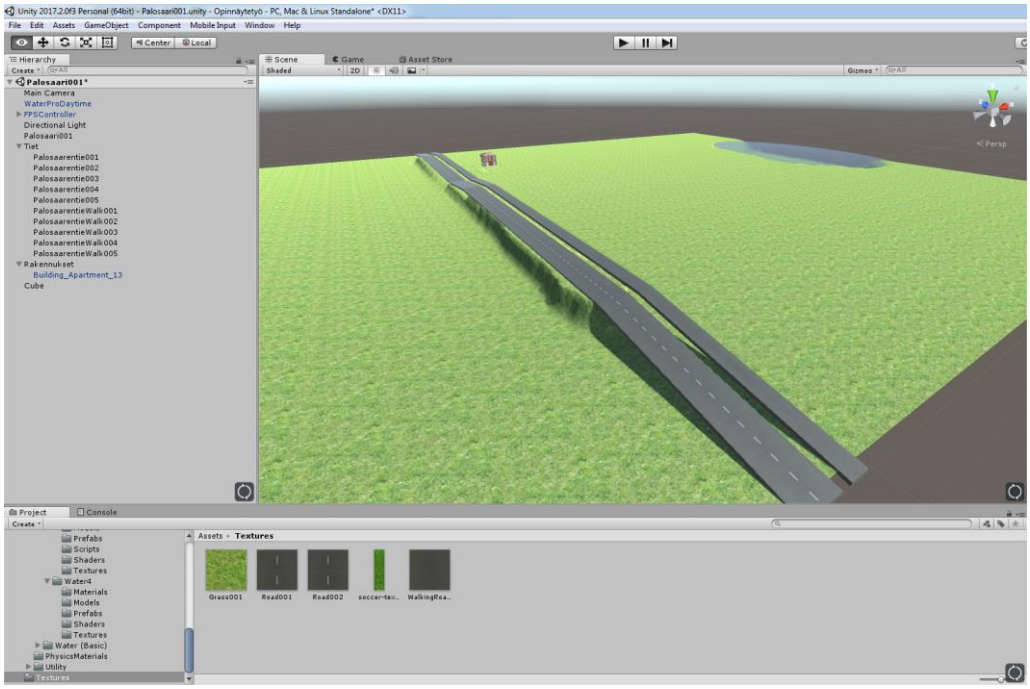
Kuvio 8. Google Earthista tulostettuja etäisyyksistä kertovia kuvia seinällä.

Tämän näkymän avulla on kätevä tehdä mäet ja teiden kaltevuudet mahdollisimman realistisesti. Avaamalla toiselle näytölle Google Earthin ja toiselle Unityn voi tehokkaasti rakentaa tietä Cube-objekteilla. Tämän voi havainnollistaa katsomalla dokumentin lopusta liitteen numero yksi (Liite 1).



Kuvio 9. Viivaintyökalulla voidaan katsoa mittapisteiden välisiä korkeuseroja.

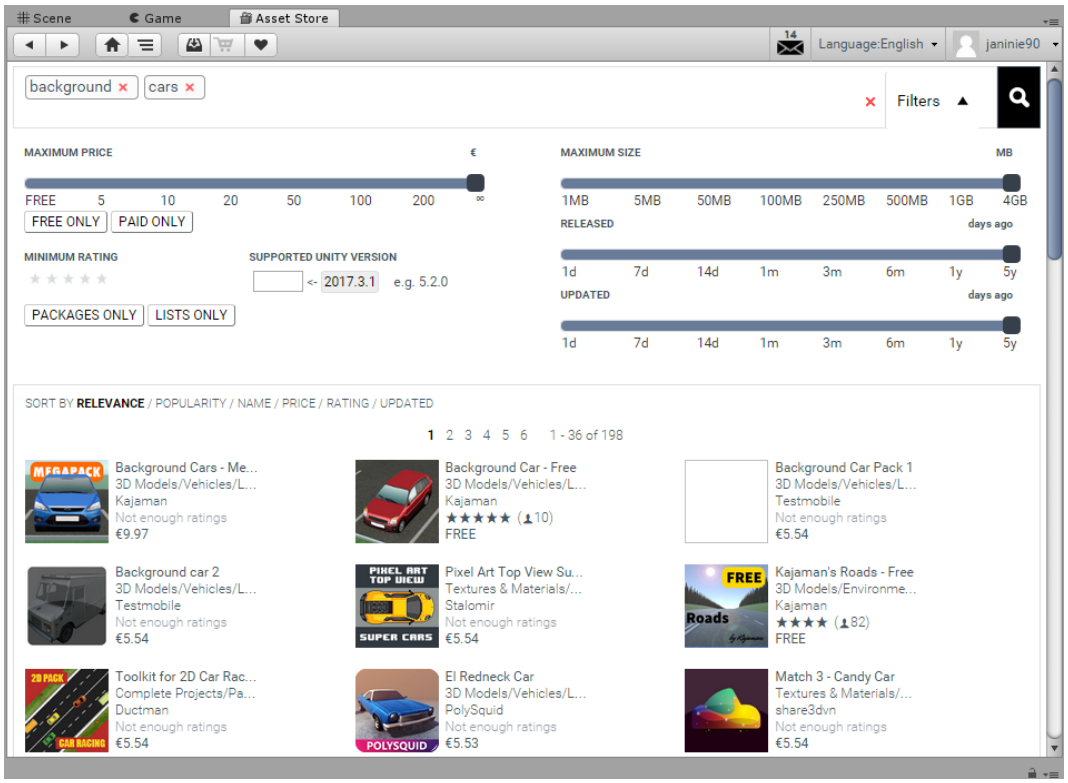
Kun Palosaarentie oli rakennettu fyysisesti, niin liitettiin autotiehen harmaa tekstuuri, jossa näkyvät tien keskiviivat. Tämä tekstuuri löytyy Unityn Asset Storesta nimellä ”Kajaman’s Roads - Free”. Tekstuuri tuli skaalata sopivaksi luotuihin Cube-objekteihin, jonka jälkeen tie oli valmis ja alettiin keskittymään muun ympäristön korkeuksien muokkaamiseen.



Kuvio 10. Palosaarentien luominen Unity-kehitysympäristössä.

4.1.2 Unity Asset Store

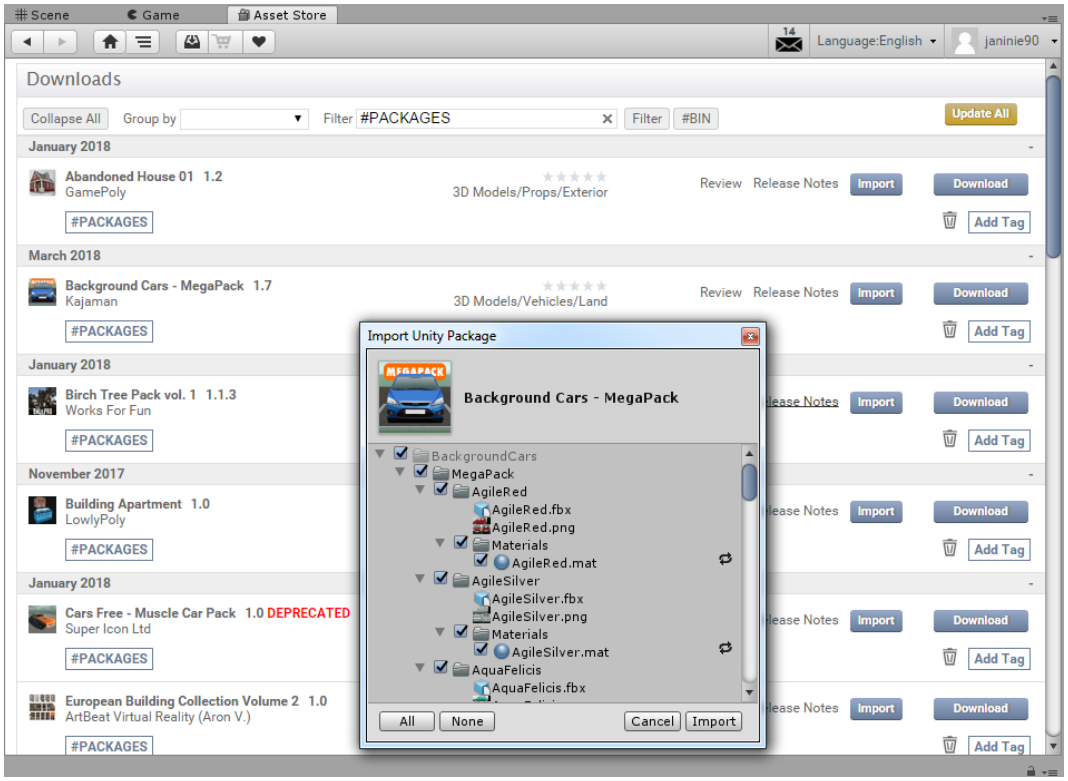
Unity Asset Storesta voi ladata muun muassa ilmaisia tai maksullisia työkaluja, grafiikkaa, laajennuksia ja palveluita Unityyn. Asset Store on pelikehityksessä äärimmäisen kätevä, sillä 3D-mallien luominen esimerkiksi Blenderillä vie paljon aikaa. Asset Storen avulla valmiiksi tehdyt mallit voidaan ladata vain napin painalluksella joko selaimella tai suoraan Unitylla.



Kuvio 11. Asset Storesta voi ostaa peliin esimerkiksi 3D-malleja autoista.

Tässä projektissa käytettiin seuraavia paketteja Asset Storesta: Abandoned House 01, Background Cars – Megapack, Birch Tree Pack vol. 1, Building Apartment, Cloth Animation Based Flag, Crowbar, European Building Collection Volume 2, Fence pack 1, Foliage Pack Free, Football Kit, Free Night Sky, Kajaman's Roads – Free, Nature Starter Kit 2, PBR Dirty Dumpster, Playground, Realistic Birches Pack, Realistic Pine Tree, Rock Pack, School Package, Scots Pine Trees Package, Simple Urban Buildings Pack 1, Street Environment Pack, Street Lamp, Ultimate Fantasy Creator LITE ja White Wizard.

Kun käyttäjä ostaa Asset Storesta jonkin paketin, niin se ilmestyy Asset Storen Download Manager välilehdelle. Täältä voidaan ladata paketti painamalla Download painiketta, jonka jälkeen paketti tuodaan Unityyn Import-painikkeen avulla. Alla näkyvässä kuvassa (Kuvio 12) on ladattu ”Background Cars – MegaPack” paketti, jota ollaan juuri tuomassa Unityyn. Tämä paketti sisältää 3D-mallinnettuja autoja, joita voidaan sijoittaa pelimaailmaan.



Kuvio 12. Download Managerilla voi tuoda komponentteja projektiin.

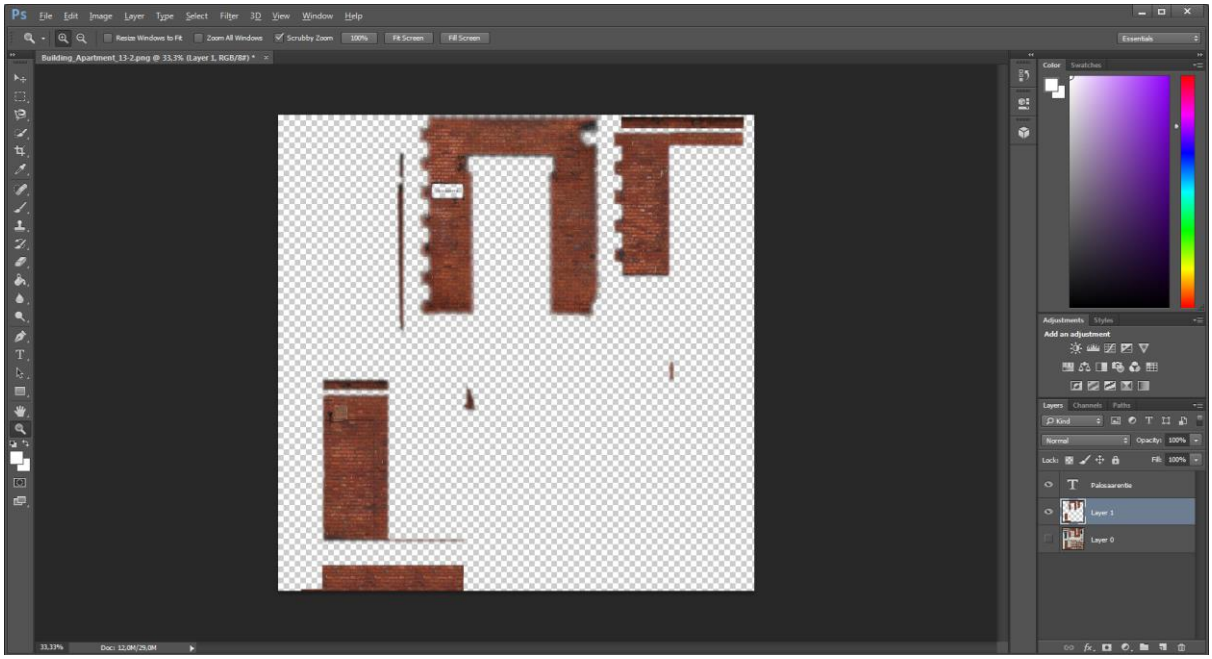
Kun paketti on tuotu Unityyn, niin sen kansio ilmestyy alapaneelin projektivälilehdelle. Kansio sisältää varsinkin tässä tapauksessa paljon alihakemistoja, sillä kyseessä on ”MegaPack”. Kun avataan esimerkiksi ”AgileRed” hakemisto (Kuvio 13), niin nähdään 3D-mallin tekstuuri, eli kuvatiedosto ja valmis 3D-malli, eli Unity-kielellä prefab. Prefab-tiedoston tunnistaa Unityssä esikatselukuvassa olevasta play-painikkeesta. Seuraavassa kuvassa käyttäjä on vetänyt prefab-tiedoston pelimaailmaan (Kuvio 13).

Kuvio 13. 3D-mallinnetun auton siirtäminen pelimaailmaan.

4.1.3 Tekstuurien muokkaus Adobe Photoshopilla

Jotta pelimaailmasta saa haluamansa näköisen on muokattava tekstuureja, eli kuvatiedostoja, jotka voidaan sijoittaa pelimaailmaan. Yksi hyvä työkalu tähän on Adobe Photoshop, jonka versiota CC 2014 käytettiin tässä projektissa. Photoshopilla voidaan muokata tekstuurien väritystä ja vaikka tekstejä, joita on joissakin tekstuureissa. Tässä esimerkissä muutettiin talon seinien väri ja talon seinässä olevaan tekstiin tien nimeksi Palosaarentie.

Tekstuurit löytyvät hakemistosta, johon Unity-projekti on tallennettu. Oletuksena tämän hakemiston polku on Windows-käyttöjärjestelmässä:
”C:\Users\#Käyttäjänimi#\Documents\Unity\#Projektin nimi#\Assets”.
Seuraavassa kuvassa on avattu tästä hakemistosta Talo-objektin kuvatiedosto Photoshopilla. Talon seinien väri on kätevä muuttaa Photoshopin Hue/Saturation -työkalulla, joka löytyy CC 2014 versiossa yläpalkin Image → Adjustments -valikosta.

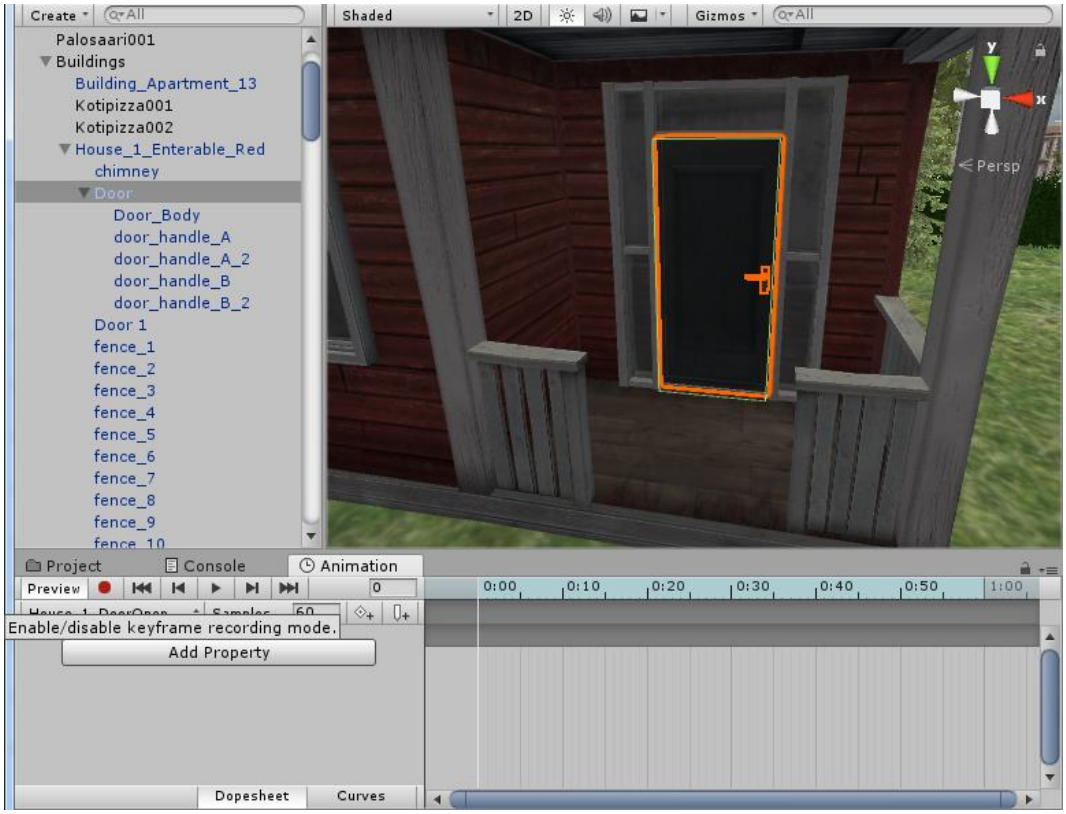


Kuvio 14. Talo-objektin seinien värien muuttaminen Adobe Photoshopilla.

4.2 Objektien animointi

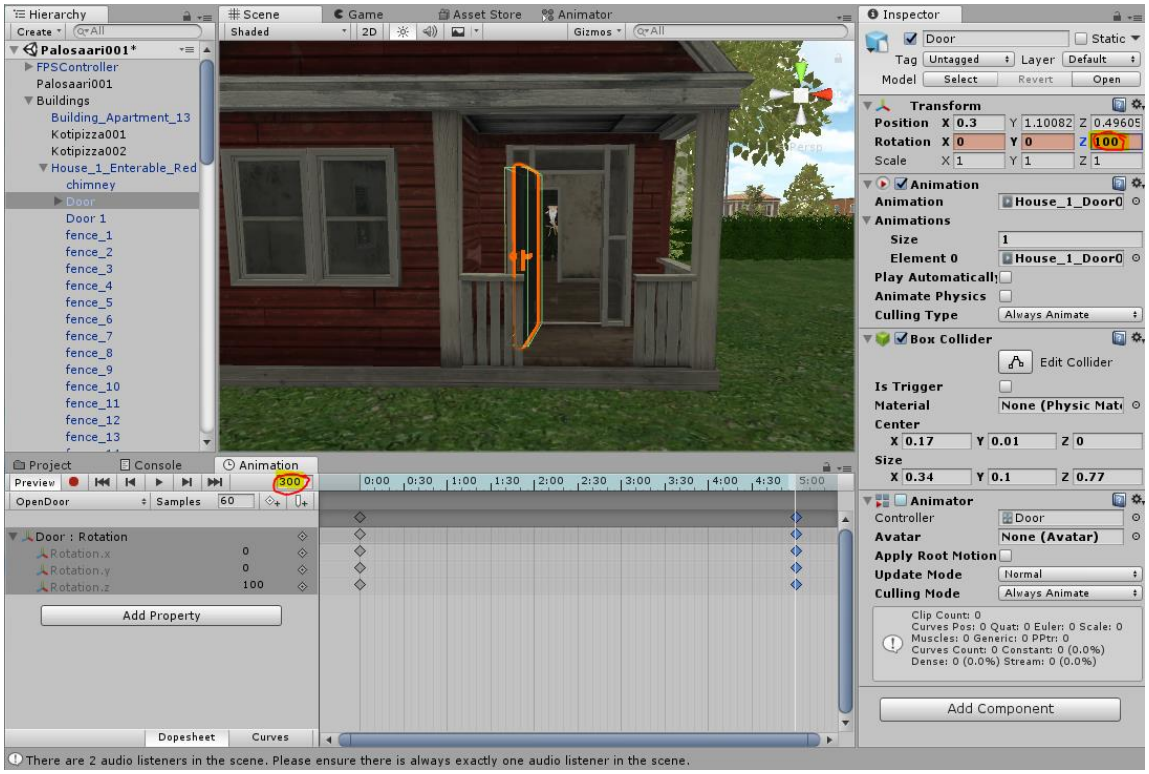
Jotta peliin saa interaktiivisuutta ja eloa, niin tulee joitakin pelimaailman objekteja animoida. Tässä projektissa tuli animoida muun muassa ovi, josta pääsee sisälle taloon hakemaan tehtäviä. Animointi aloitetaan valitsemalla haluttu objekti ja painamalla Animation-välilehden tallennuspainiketta (Kuvio 15).

Animaatioon tulee tehdä avainkehyksiä, joille tallennetaan objekti esimerkiksi eri asennossa, kuten tässä oven aukeamisessa. Animaatioiden kuvataajuus (FPS) on 60, eli sekunnissa näytölle piirretään 60 kuvaa. Tämä tarkoittaa, että kuvaruutu 30 on puolen sekunnin kohdalla, kuvaruutu 60 yhden sekunnin kohdalla ja niin edespäin.



Kuvio 15. Animation-välilehden tallennuspainike.

Ensimmäisellä kerralla kun animaatio oven avaamiseen tehtiin, niin lisättiin animaatioon puolen sekunnin välein avainkehyksiä, joissa z-akselin arvoa muutettiin 10 asteella. Huomattiin kuitenkin opettajan avustuksella ja kokeiluilla, että jos lisätään pelkästään alkuun avainkehys, jossa ovi on kiinni ja loppuun taas kehys, jossa z-akselin arvoa kasvatetaan 100 asteella, niin Unity osaa animoida oven aukeamisen alusta loppuun. Ovi aukeaa siis tässä tapauksessa hieman yli suorakulman verran.



Kuvio 16. Animaation kuvaruudussa 300 avataan ovea 100 astetta.

4.3 Skriptien ohjelmointi

Joka pelissä tulee olla myös skriptejä, joiden avulla pelikehittäjä voi ohjelmoida interaktiivisuutta peliin. Unityssa on valtava määrä sisäänrakennettuja skriptejä, joten hyvin yksinkertainen peli on mahdollista rakentaa pelkästään niiden avulla. Tässä työssä täytyi kuitenkin ohjelmoida omia skriptejä muun muassa ovien avaamiseen, tarinan tehtävien hallintaan ja peliympäristön ominaisuuksien muutoksiin.

4.3.1 Skripti etäisyyden mittaamiseen

Tähän projektiin tarvittiin ensimmäisenä skripti, jolla voidaan mitata etäisyyttä pelaajan näkökulmasta eteenpäin. Skriptiä tarvitaan esimerkiksi, jos pelaaja kävelee kohti ovea ja sen läheisyydessä pelin tulee ilmoittaa näppäin, jolla ovi avataan. Toiminnenäppäimen, joka avaa oven tulee myös toimia ainoastaan oven edessä, jotta pelaaja ei voi avata ovea toiselta puolelta pelikenttää ja näppäin säilyy muuhun käyttöön. Ohjelmoitiin siis skripti, jota voidaan kutsua muista luokista.

```
using System.Collections.Generic;
using UnityEngine;

public class PlayerDistance : MonoBehaviour {

	public static float PlayerDistanceTo;
	public float FromObject;

	// Tämä funktio kutsutaan joka ruudussa
	void Update () {
		RaycastHit Hit;
	// Tämä ehtolauseke määrittää etäisyyden arvon
		if (Physics.Raycast (transform.position, trans-
		form.TransformDirection (Vector3.forward), out Hit)) {
			FromObject = Hit.distance;
			PlayerDistanceTo = FromObject;
		}
	}
}
```

Ehtolausekkeen, joka määrittää etäisyyden tuli sijaita update-funktion sisällä, jotta sitä kutsutaan pelin joka ruudussa ja etäisyys päivittyy reaaliaikaisesti. Skriptissä käytetään apuna Physics.Raycast -luokkaa ja määritetään sille, että etäisyys mitataan eteenpäin: `transform.TransformDirection (Vector3.forward).`

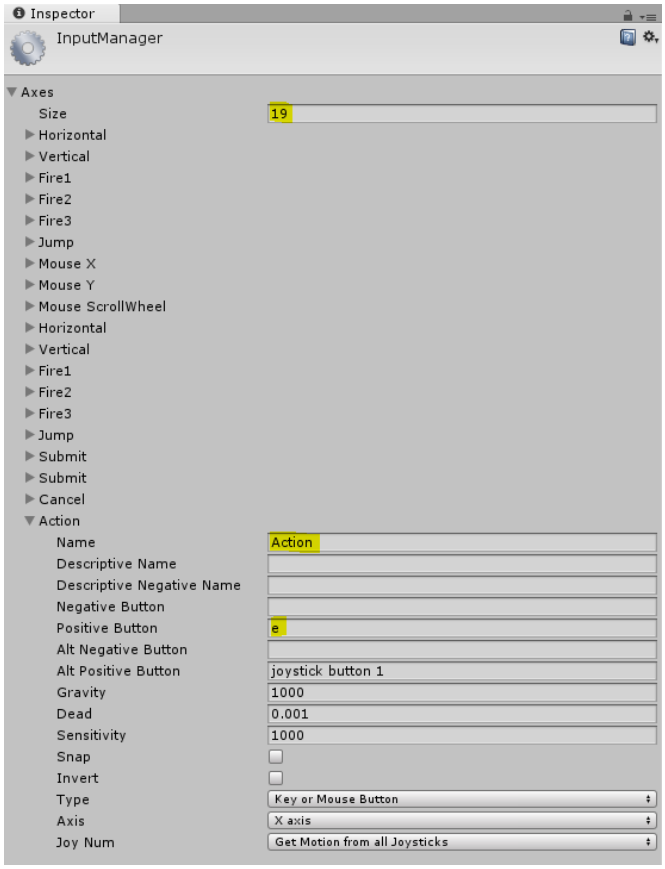
Määritetään pisteeksi, josta etäisyyttä lähdetään mittaamaan muuttuja nimeltä Hit. Etäisyys tästä pisteestä saadaan yksinkertaisesti Hit.distance -komennolla, jonka arvo sijoitetaan PlayerDistanceTo nimiseen float-muuttujaan. Jatkossa voidaan kutsua tätä muuttujaa muista luokista.

4.3.2 Skripti oven avaamiseen

Kun etäisyyden mittaava skripti oli tehty, niin voitiin jatkaa ohjelmoimaan skriptiä, jolla voidaan avata ovi taloon, josta pelaaja voi hakea tehtäviä. Tämä skripti vaati toimiakseen mahdollisuutta mitata etäisyyttä pelaajasta katsoen eteenpäin. Kun pelaaja on oikealla etäisyydellä ovesta ja painaa toiminnenäppäintä, niin käynnistyy animaatio, jossa ovi aukeaa ja taloon pääsee sisään. Tämä vaati aiemmin luodun animaation ja ohjelmoitavan skriptin yhteistyötä.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class O001OpenDoor : MonoBehaviour {

	public float YourDistance;
	public GameObject Door;
	public GameObject AcDisplay;
	public GameObject AcText;

	void Update () {
	 // Käytetään aiemmin luotua PlayerDistance-luokkaa
	 YourDistance = PlayerDistance.PlayerDistanceTarget;
	}

	void OnMouseOver () {
		if (YourDistance <= 1.5) {
	     // Jos etäisyys <= 1.5 tulostetaan käyttöliittymään tekstiä
	     AcText.GetComponent<Text> ().text = "Open Door";
			AcText.SetActive (true);
			AcDisplay.SetActive (true);
		}

		if (Input.GetButtonDown ("Action")) {
			if (YourDistance <= 1.5) {
				this.GetComponent<BoxCollider> ().enabled =
false;
		  /* Kun painetaan toiminnenäppäintä ja etäisyys <= 1.5
		     käynnistetään animaatio */
		  Door.GetComponent<Animation> ().Play ("House_1_DoorOpen");
		  // Otetaan käyttöliittymän tekstit pois
				AcText.SetActive (false);
				AcDisplay.SetActive (false);
			}
		}
	}

	void OnMouseExit () {
		AcDisplay.SetActive (false);
		AcText.SetActive (false);
	}
}
```

Koodissa on määritelty toiminnenäppäin ”Action”, joka on tullut määrittää projektin asetuksissa. Näppäimet määritellään valitsemalla Unityssa: Edit → Project Settings → Input. Kun käyttäjä navigoi Input-asetuksiin, niin Unityn oikeassa palkissa sijaitsevaan Inspectoriin aukeaa seuraavassa kuvassa oleva näkymä (Kuvio 17).



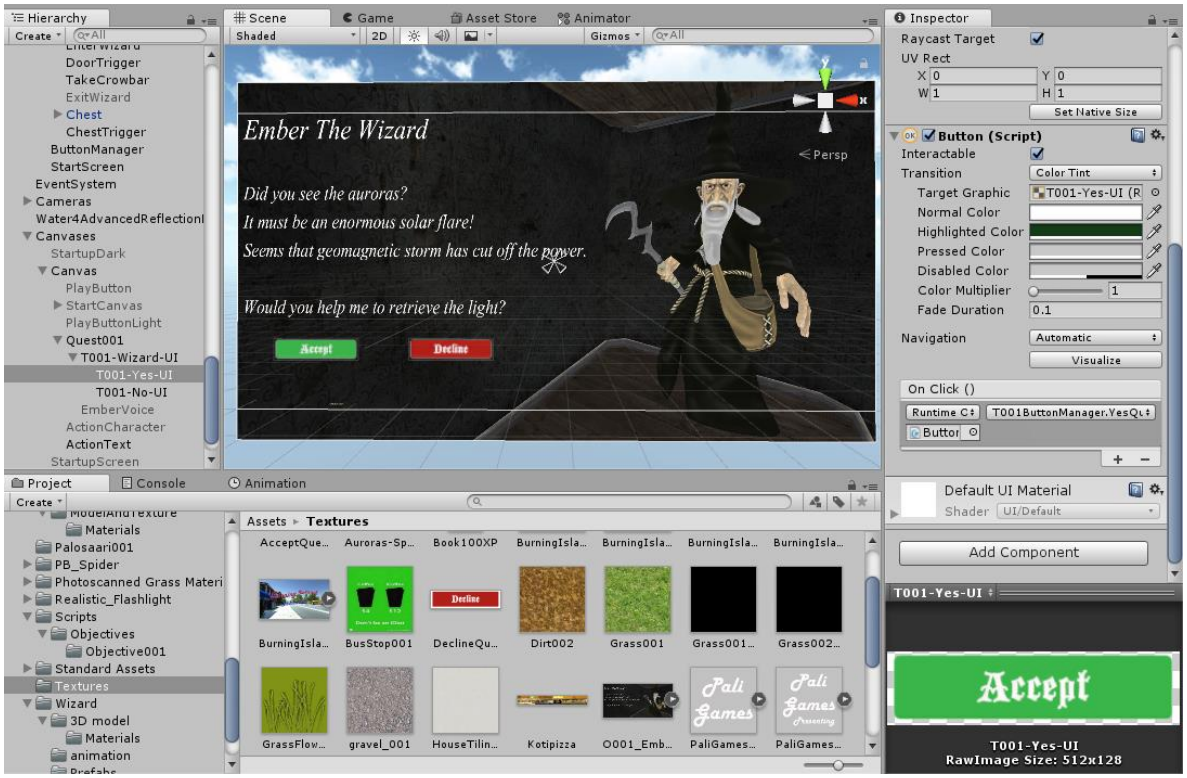
Kuvio 17. Uuden pelissä käytettävän näppäimen määritys Unityssa.

Jotta voidaan lisätä täysin uusi näppäin, niin tulee Size-kohdan arvoa kasvattaa yhdellä, eli tässä tapauksessa 18+1=19. Kun muutos on tehnyt, niin Unity monistaa viimeisen näppäimen listassa. Uudelleennimettiin siis toinen Cancel-näppäimistä Action-nimiseksi. Positive Button kohtaan tulee taas laittaa sen näppäimen arvo, jolla halutaan käyttää tätä Action-näppäintä. Tässä tapauksessa siis käytetään e-näppäintä.

Tämän määrityksen jälkeen voidaan koodissa viitata toiminnenäppäimeen nimellä ”Action”. Esimerkiksi aiemman skriptin ehtolausekkeessa määritellään, että ”House_1_DoorOpen”-animaatio lähtee käyntiin, kun painetaan määriteltyä toiminnenäppäintä: `if (Input.GetButtonDown ("Action")).`

4.3.3 Skripti tehtäväpainikkeiden hallintaan

Burning Island Auroras on tyylilajiltaan roolipeli, joten peli tarvitsee tehtäviä, josta pelaaja saa kokemuspisteitä. Tehtävät taas tarvitsevat välianimaatioita tai ruutuja, joista selviää mitä tehtävän suorittamiseksi pitää tehdä. Seuraavassa kuvassa (Kuvio 18) nähdään tällainen ruutu ja painikkeet, josta tehtävän voi hyväksyä tai hylätä. Tehtäväpainikkeet vaativat funktiot, jotka ajetaan kun pelaaja painaa niitä. Kuvassa on valittuna hyväksy-painike, johon on määritetty käynnistyvä funktio Inspector-ikkunassa. On Click () -kohdassa on valittu skripti ButtonManager ja funktio Yes-Quest.



Kuvio 18. Tehtäväpainikkeet vaativat ohjelmointia toimiakseen.

Itse skriptissä tuli olla funktio kummallekin painikkeelle, joten ohjelmoitiin NoQuest ja YesQuest -funktiot. Tämä tehtävä on tarkoitettu esittelemään Unityn ominaisuuksia ja antamaan kuvaa minkälainen pelistä on tulossa. Jos pelaaja painaa hylkää-painiketta, käynnistyy NoQuest-funktio, eli keskeytetään tehtävänantajan puhe, sekä poistetaan näytöltä tehtäväruutu ja hiiren kursori.

Jos pelaaja taas hyväksyy tehtävän, käynnistyy YesQuest-funktio, jossa:

- Poistetaan tehtäväruutu näytöltä ja revontulet taivaalta
- Muutetaan skybox, eli yötaivas kirkkaammaksi taivaaksi
    - `RenderSettings.skybox = DaySkybox;`
- Muutetaan valaistusta kirkkaammaksi
    - `RenderSettings.ambientLight = Color.white;`
- Otetaan pelaajalta taskulamppu pois kädestä
- Keskeytetään alun aurinkomyrsky -ääni
- Keskeytetään pelihahmon ääni ja käynnistetään musiikki
- Poistetaan kursori näkyvistä

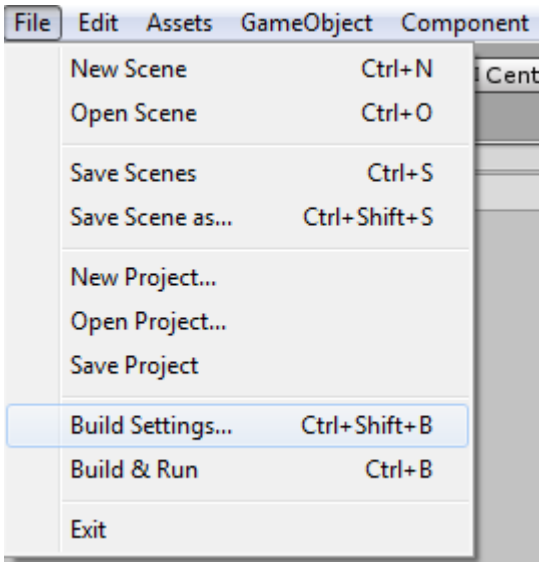
Kaikki nämä vaiheet ovat nähtävissä alla olevasta O001ButtonManager -skriptistä.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
// Tarvitaan UI-luokka, jotta voidaan tulostaa käyttöliittymälle
using UnityEngine.UI;

public class O001ButtonManager : MonoBehaviour {

    public GameObject UIQuest;
    public GameObject WizCam;
    public GameObject NoQuestButton;
    public GameObject YesQuestButton;
    public Material DaySkybox;
    public GameObject SunLight;
    public GameObject Auroras;
    public GameObject FlashLight;
    public GameObject SolarFlareSound;
    public GameObject EmberVoice;
    public GameObject ExitVoice;
    public GameObject CurrentObjective;

    // Kun pelaaja hylkää tehtävän suoritetaan tämä funktio
    public void NoQuest()
    {
        UIQuest.SetActive(false);
        WizCam.SetActive(false);
        NoQuestButton.SetActive(false);
        YesQuestButton.SetActive(false);
        EmberVoice.SetActive(false);
        CurrentObjective.SetActive(true);
        // Piilotetaan ja lukitaan hiiren kursori
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;

    }

    // Kun pelaaja ottaa tehtävän vastaan suoritetaan tämä funktio
```

```
    public void YesQuest()
    {
        Auroras.SetActive(false);
    // Muutetaan Skybox- ja valaistusasetuksia Render-asetuksista
        RenderSettings.skybox = DaySkybox;
        RenderSettings.ambientLight = Color.white;
        SunLight.SetActive(true);
    // Otetaan tehtävänäyttö pois ruudulta
        UIQuest.SetActive(false);
        WizCam.SetActive(false);
        NoQuestButton.SetActive(false);
        YesQuestButton.SetActive(false);
        FlashLight.SetActive(false);
        SolarFlareSound.SetActive(false);
    // Keskeytetään pelihahmon ääni ja käynnistetään musiikki
        EmberVoice.SetActive(false);
        ExitVoice.SetActive(true);
        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;
    }
}
```

4.4 Pelin kääntäminen

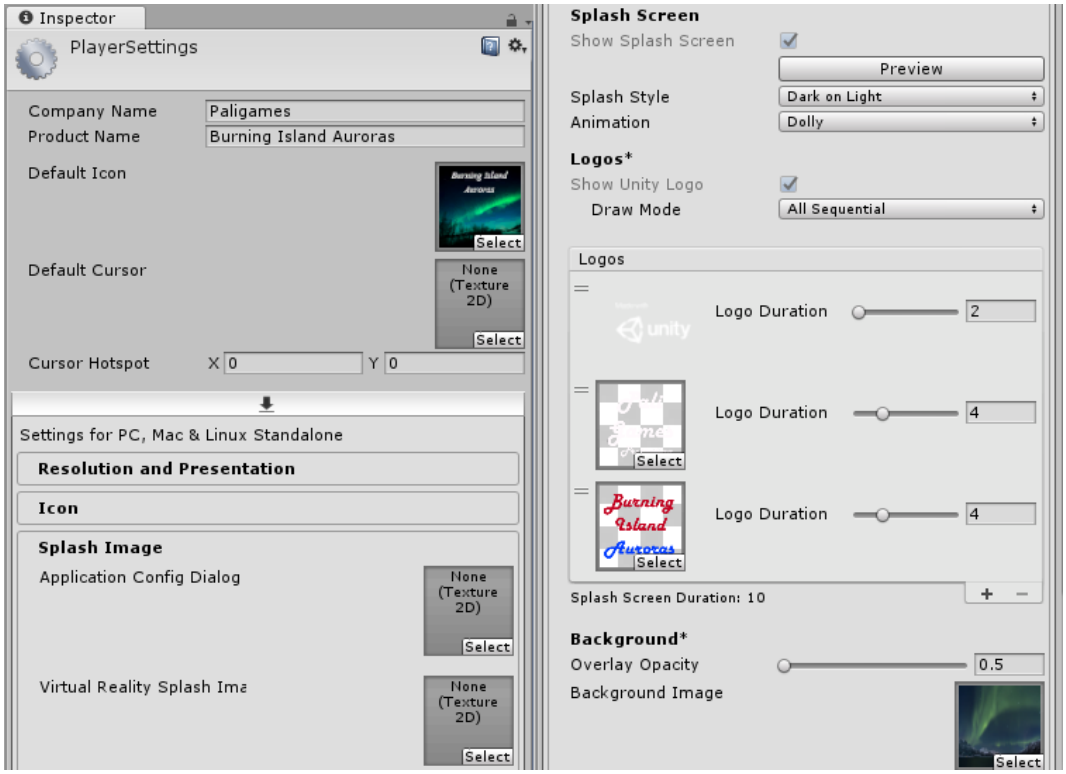
Kun peli on valmis tai sitä halutaan testata lopullisessa muodossaan, niin se tulee kääntää samaan tyyliin kuin ohjelmoinnissa käännetään koodi lopuksi tietokoneen ymmärtämään muotoon. Kuten Unityssa yleensä, myös kääntäjä on ohjelmassa sisäänrakennettuna. Ensiksi tulee laittaa kääntämisen asetukset kuntoon ja se onnistuu Unityn File-valikon Build Settings -kohdasta (Kuvio 19).



Kuvio 19. Pelin kääntäminen aloitetaan Unityn File-valikosta.

Kun avataan Build Settings, painetaan Player Settings, jolloin Inspector-ikkunaan aukeaa asetukset, jotka tulee määrittää ennen kääntämistä (Kuvio 22). Jos näitä ei määritetä, niin Unity kääntää pelin oletusasetuksilla. Muutettiin Company Name -kohtaan kuvitteellinen peliyrityksen nimi Paligames. Product Name -kohtaan kirjoitettiin pelin nimi, eli tässä tapauksessa Burning Island Auroras.

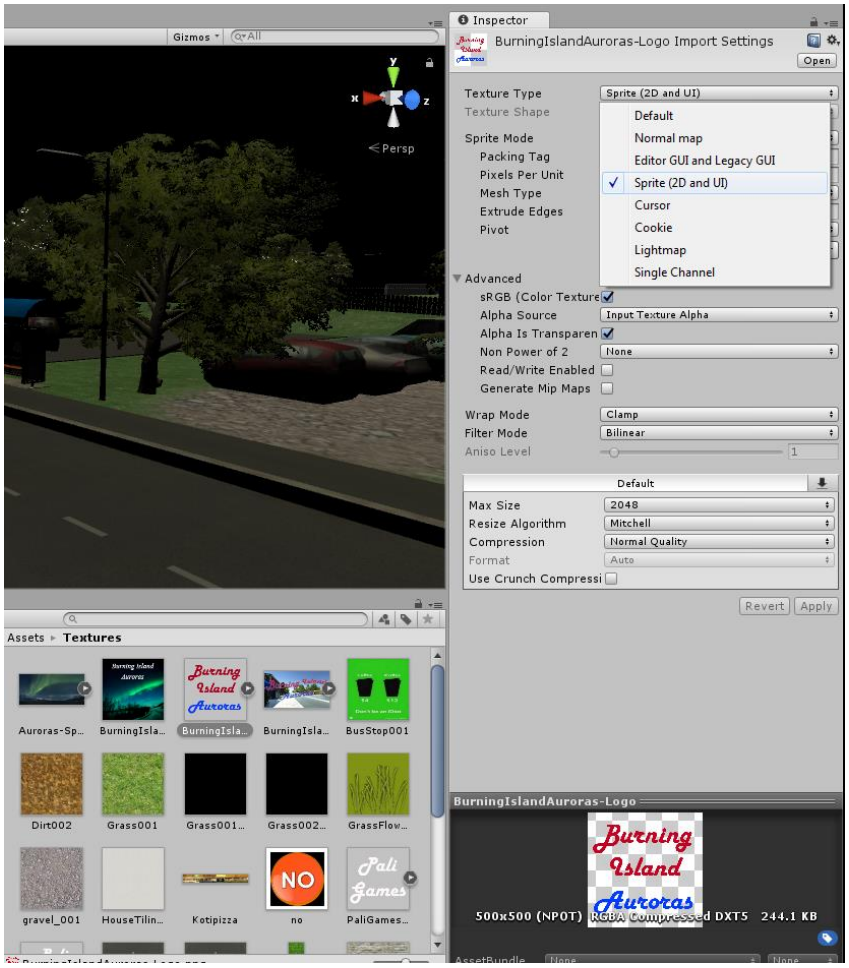
Default Icon -kohta taas määrittää pelin ikonin, joka näkyy muun muassa käyttöjärjestelmän resurssienhallinnassa. Vedettiin tähän aiemmin luotu kuvatiedosto Unityn Project-välilehdeltä. Application Config Dialog -kohdassa voidaan lisätä tekstuuri ennen peliä aukeavalle asetusikkunalle, mutta jätetään se vielä tässä vaiheessa tyhjäksi.



Kuvio 20. Asetukset, jotka aukeavat Player Settings -kohdasta.

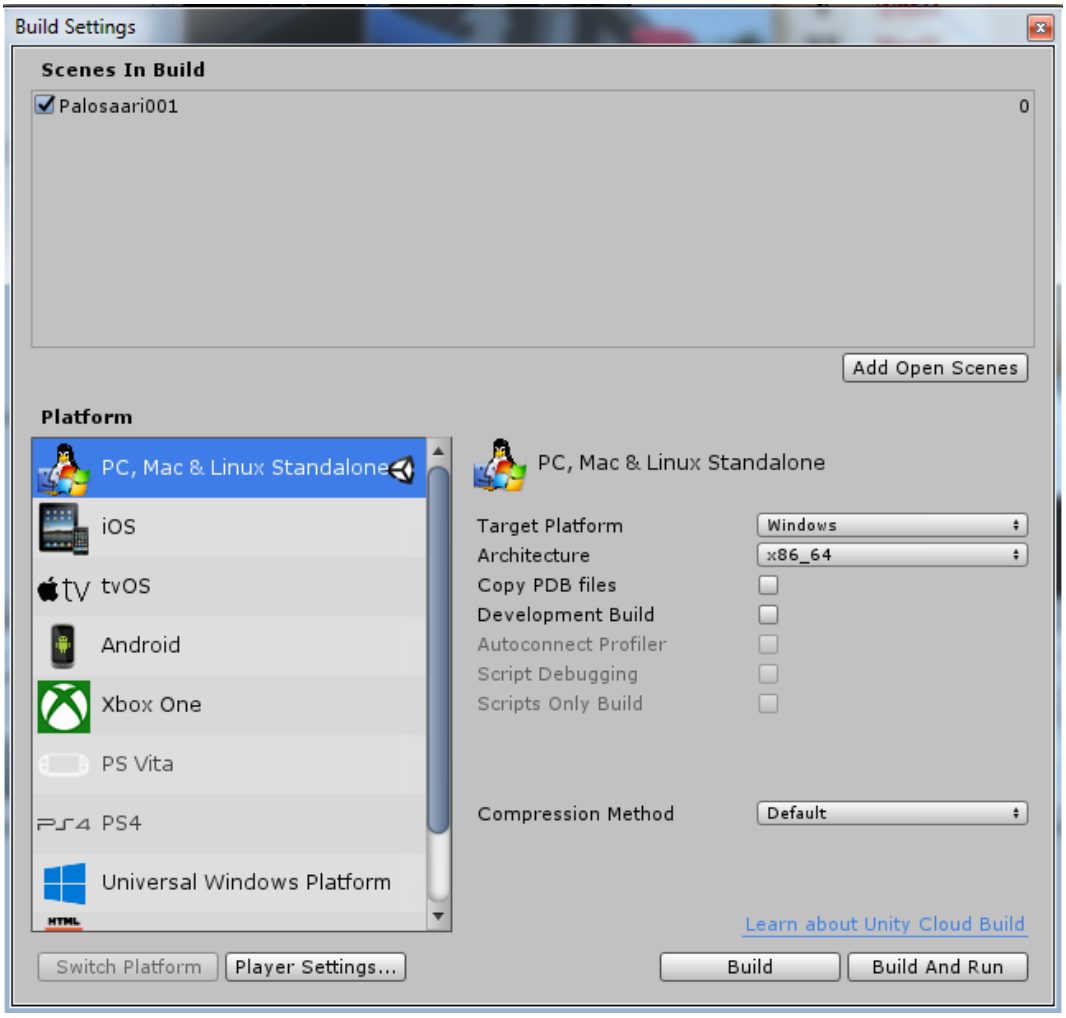
Splash Screen -kohdan asetukset määrittävät monista peleistä tutun käynnistyskuvan, joka pysyy näytöllä Logo Duration -kohdassa annetun ajan. Logos-kehyksen plusmerkillä voidaan lisätä logoja niin monta kuin tarvitaan. Yllä olevassa kuvassa näkyy tämän projektin kolme logoa, eli Unityn oma logo, kuvitteellisen peliyrityksen nimi Paligames, sekä pelin nimi Burning Island Auroras.

Unityn oma logo näkyy ruudulla kaksi sekuntia ja pelin logot neljä sekuntia. Logot on tehty Adobe Photoshop CC 2014 -kuvankäsittelyohjelmalla, jonka jälkeen ne tuli viedä Unityn Project-ikkunaan. Unityssa kuvat tuli muuttaa Sprite-muotoon, jotta ne toimivat pelin käyttöliittymässä ja jotta ne voidaan ensinnäkin lisätä logoksi Unityyn. Seuraavassa kuvassa nähdään, miten logon tyyppi muunnetaan oletusarvosta Sprite (2D and UI) -muotoon.



Kuvio 21. Kuvan tyypin muuttaminen Sprite-muotoon.

Viimeisenä määritetään taustakuva, joka näkyy saman ajan, kuin logot ovat määritelty näkymään. Kuva näkyy logojen taustalla ja sen ei tarvitse olla Sprite-muodossa, vaan esimerkiksi normaali png-kuvatiedosto kelpaa. Tässä projektissa lisättiin taustakuvaksi revontulet, koska ne ovat suuri osa peliä myöhemmin. Kun kaikki asetukset ovat määritelty, voidaan kääntää peli Build Settings -ikkunan Build-painikkeesta (Kuvio 22). Burning Island Auroras pelin kääntämiseen meni keskiverto PC:llä vain 30 sekuntia, vaikka peli sisältää runsaasti grafiikkaa ja objekteja.



Kuvio 22. Pelin kääntäminen onnistuu Build Settings -ikkunasta.

5 JOHTOPÄÄTÖKSET

Tämän projektin pohjalta voi sanoa, että pelikehitys vaatii todella paljon aikaa, varsinkin jos luodaan iso graafinen peli. Vaikka kehitysympäristöt ovat kehittyneet todella pitkälle ja niiden avulla pelikehitys on valonnopeaa entiseen verrattuna, niin kehittyneet ovat myös vaatimukset pelin sisällöstä ja pelimaailman graafisuudesta. Tämä tarkoittaa sitä, että hiemankin suuremman luokan pelikehitys tarvitsee suuren työryhmän taustalle.

Näin ison projektin ottaminen työn alle oli kuitenkin hyvä päätös, sillä pienemmällä tavoitteella Unityn kaikkia ominaisuuksia ei olisi päässyt testaamaan yhtä laajasti. Päästiin myös koittamaan useita eri osa-alueita pelikehityksestä, joka taas helpottaa valitsemaan oikean polun matkalla pelialalle. Tämän tyylinen pelikehitys parantaa luovaa ajattelua ja teknistä osaamista.

Peliympäristö oli haastava luoda realistisen näköiseksi, joten siihen kului paljon aikaa ja se ei tullut vielä tämän opinnäytetyön aikana täysin valmiiksi. Se ei ollut tarkoituskaan sillä peliä aiotaan kehittää tästä eteenpäinkin, niin kauan kuin innostusta riittää. Projektin aikana huomattiin, että pelin luominen on koukuttavaa puuhaa, sillä työn jäljen näkee hyvin konkreettisesti ruudulta.

Kaikille joita peliala kiinnostaa, voidaan tämän työn pohjalta suositella Unity-kehitysympäristöä, YouTube-kursseja, Unityn omia ohjeita ja rohkeaa kokeilemista. Älä pelkää, että peli menisi hajalle, se on vain virtuaalinen.

LÄHTEET

C# vs. JavaScript | Unity 5 Comparison 2017. YouTube.
https://www.youtube.com/watch?v=lVegV8k0mlA

Jimmy Vegas Unity Tutorials 2018. YouTube.
https://www.youtube.com/channel/UCRMXHQ2rJ9_0CHS7mhL7erg

MessiLive - Game Industry in Finland: study, work, start your own business 2018.
https://www.youtube.com/watch?v=hmA9sGdJiyk

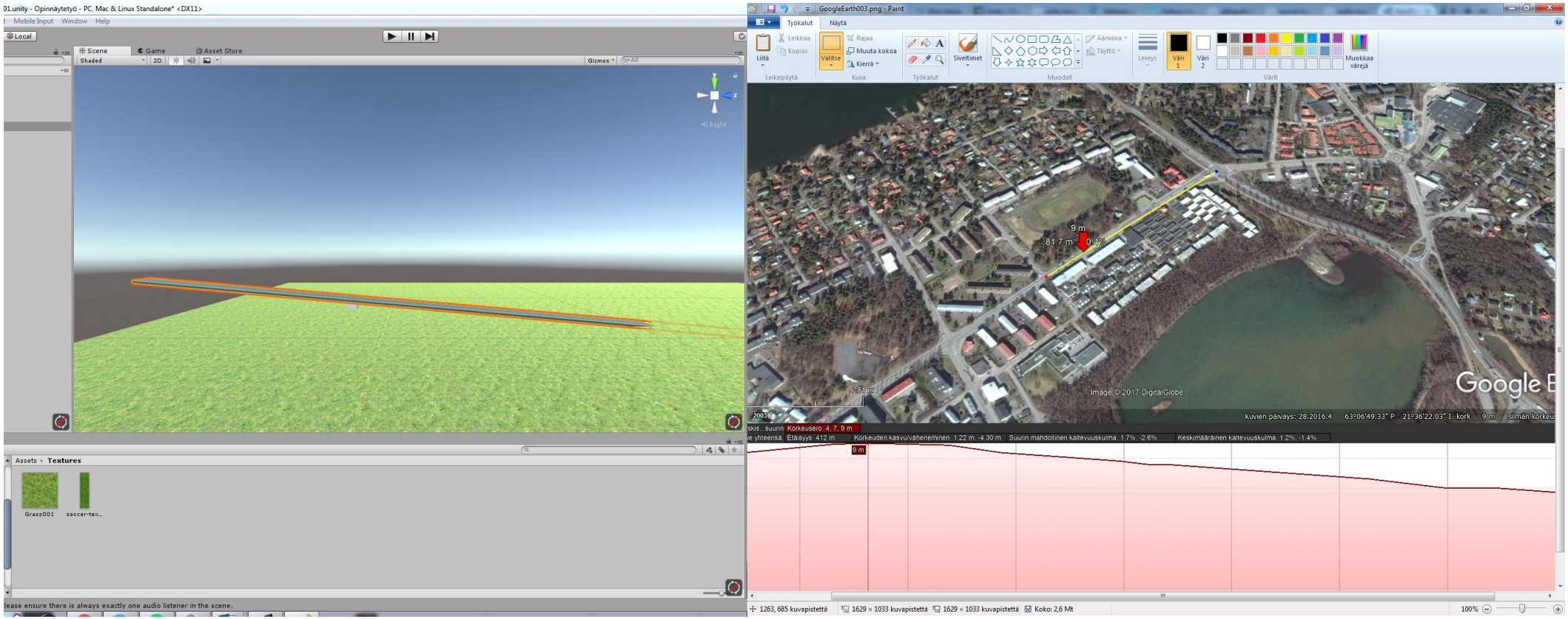
Game Development with Unity 2018. Study Tonight.
https://www.studytonight.com/3d-game-engineering-with-unity/game-development-architecture

Unity (pelimoottori) 2018. Wikipedia. Viitattu 22.1.2017.
https://fi.wikipedia.org/wiki/Unity_(pelimoottori)

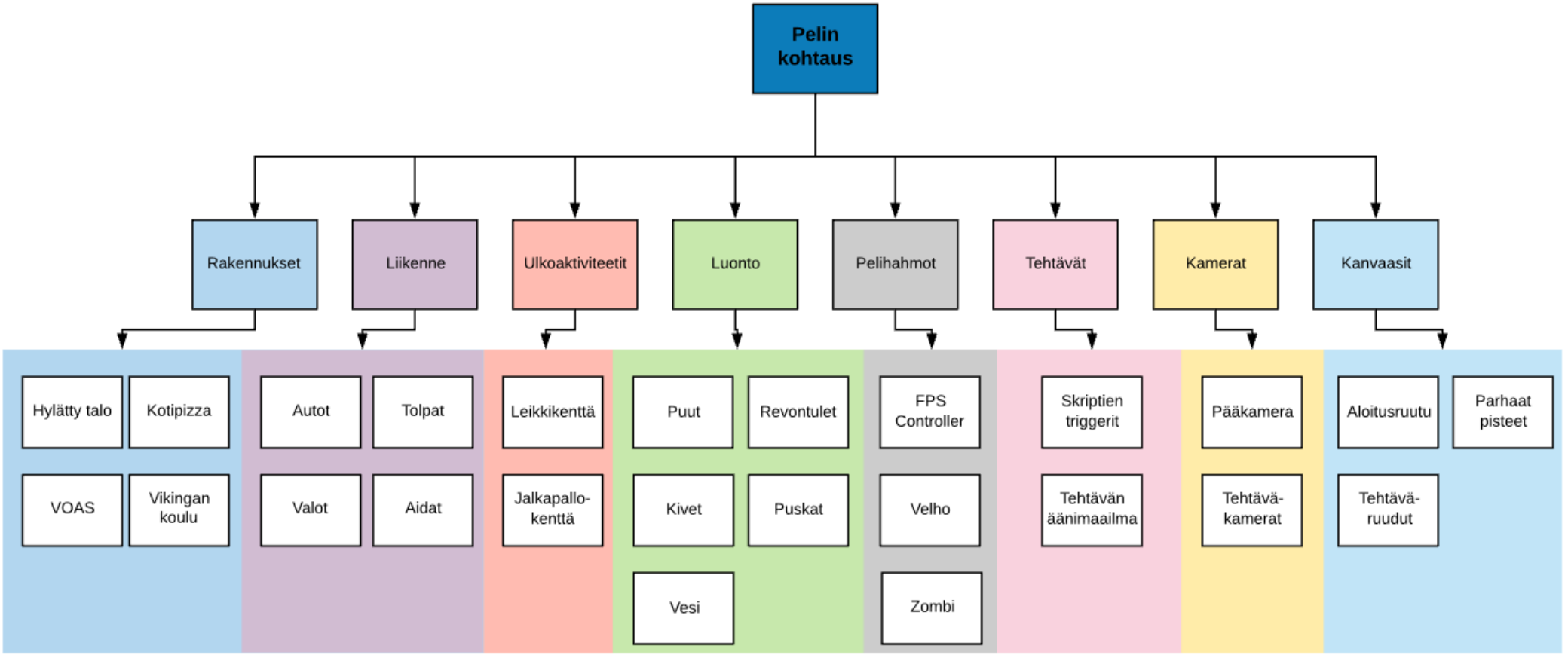
Unity Documentation – Camera Motion Blur 2018. Unity3D.
https://docs.unity3d.com/550/Documentation/Manual/script-CameraMotionBlur.html

Unity Documentation - MonoDevelop 2018. Unity3D.
https://docs.unity3d.com/Manual/MonoDevelop.html

Unityn omat verkkosivut 2018. Unity3D.
https://unity3d.com



Liite 1. Palosaarentien rakentaminen Google Earthin ja Unityn avulla kahdella näytöllä.



Liite 2. Burning Island Auroras -pelin objektien hierarkia.