Henri Lagerroos

Työkalu avoimen 3D-maailman luontiin Unityssä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

2.5.2022

Tiivistelmä

Tekijä:	Henri Lagerroos
Otsikko:	Työkalu avoimen 3D-maailman luontiin Unityssä
Sivumäärä:	53 sivua
Aika:	2.5.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisovellukset
Ohjaajat:	Lehtori Miikka Mäki-Uuro Lehtori Antti Laiho

Insinöörityön tavoitteena oli tehdä työkalu avoimen 3D-maailman kehittämisen helpottamiseksi Unity-pelimoottorilla. Työkalu suunniteltiin siten, että se voitaisiin myöhemmin laittaa myyntiin Unity Asset Store -kauppapaikkaan. Työkalun avulla oli tarkoitus pystyä suoratoistamaan avoimen maailman alueita. Pelaajan ollessa vuorovaikutuksessa maailman kanssa maailman alueita ladattaisiin tilannekohtaisesti kytkimiä aktivoimalla. Näin avoimen maailman alueita voitaisiin ladata kontrolloidusti ja saumattomasti pelin taustalla ilman latausikkunoita alueiden välissä.

Maailman eri alueiden välillä liikkumista varten tehtiin suoratoistaja sekä muita avoimen maailman kehittämistä varten luotuja komponentteja. Komponentteja olivat muun muassa maailman lataamista käynnistävät kytkimet, tasojen viittaustiedostot, erikseen tallennettavat aloituspisteet ja pelimaailman asetukset.

Työkalun oli myös tarkoitus avustaa pelin tasojen hallinnassa, jolloin pelimaailman erilaisia tilanteita voitaisiin simuloida painiketta painamalla ja editori avaisi pelitilanteen tasot editorin muokkaustilassa. Lisäksi työkalun oli tarkoitus avustaa muiden avoimen maailman luomiseen käytettävien komponenttien luomisessa ja muokkaamisessa. Kehittäjän avustamista varten tehtiin editoriin laajennuksia.

Insinöörityön tuloksena syntyi työkalu, jossa on avoimen maailman suoratoistaja komponentteineen sekä erilaisia editoria laajentavia ominaisuuksia, kuten valikkoja, painikkeita, ikkunoita, tiedostoja ja erilaisia muokattuja näkymiä. Laajennuksien avulla pelimaailman tasojen ja komponenttien käsittelystä tuli nopeampaa ja miellyttävämpää. Työkalua varten tehtiin myös kattava esimerkkitoteutus työkalulla tehtävästä avoimen maailman pelistä. Käännetty peli voitiin käynnistää itsenäisesti pelivalikosta ja aloittaa uudelleen tallennuspisteestä. Koepelissä oli esirakennettu eri osa-alueet, kuten valaistus, navigaatioverkot, alueellinen objektin piilottaminen ja heijastusluotaimet. Myös ei-pelattava hahmo seurasi pelattavaa hahmoa tasolta toiselle.

Työkalu implementoitiin koemielessä myös vanhaan henkilökohtaiseen peliprojektiin, joka sisälsi maastoja, erilaisia 3D-malleja, vihollisia ja pelitehtäviä. Koepelin oikeata maailmaa vastaava pinta-ala oli 0,75 km². Työkalu suoriutui koepelistä moitteetta suoratoistajan ja valikkotyökalujen osalta.

Avainsanat: Unity, avoin pelimaailma, tasojenhallinta, suoratoistokytkimet, editorin laajentaminen, pelikehitys, työkalu

Abstract

Author:	Henri Lagerroos
Title:	Tool for creating an open 3D world in Unity
Number of Pages:	53 pages
Date:	2 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Game Applications
Supervisors:	Miikka Mäki-Uuro, Senior Lecturer Antti Laiho, Senior Lecturer

---

The goal of this final year project was to make a tool to help create an open 3D world game in the Unity game engine. The tool was planned in such a way that it would be possible to later publish it on the Unity Asset Store. The main goal of the tool was to be able to stream open world areas in the game. Players should be able to interact with triggers in the game to load and unload the game areas. This way, the player can traverse the game world without pauses and loading screens.

To be able to seamlessly traverse between different areas of the world, a world streamer was created along with related components. The components included triggers for streaming, scene abstraction, a save point mechanism, and game settings.

The additional purpose of the tool was to help open world game development. Most importantly, developers should be able to easily simulate different game stages in the editor. The editor would then load the scenes for the game stage. Another set of tools was also intended to help create and modify different open world components. To aid development, different extensions were created for the Unity editor.

As a result of the project, a tool with a world streamer and related components was created. A variety of different extensions were added, such as menus, menu buttons, windows, files, and different visual editor customizations. With the extensions, editing the scenes and different open world components turned out to be more pleasant and efficient. The tool included a sample of an open world game created with the tool. The sample could be built and played as a standalone game. Different types of scene data were also baked into the game, such as lighting, navigation meshes, occlusion culling, and reflection probes. The game also included a simple main menu to start the game, along with the ability to save start points and restart the game from a start point. One non-playable character was added to follow the player around the world.

To properly test the tool, it was implemented into an old personal 3D game. The game area on the real-world scale was 0.75 km² and included terrain, 3D models, enemies, and game missions. Both the world streamer and the extensions were able to handle the game successfully.

Keywords: Unity, open world, scene management, trigger streaming, editor extension, game development, tool

Sisällys

Lyhenteet ja käsitteet

Kytkin: Suoratoistajan käyttämä objekti, jonka välityksellä aktivoidaan suoratoisto-operaatio.

LOD: *Level of Detail*. Mekanismi, joka mahdollistaa 3D-mallin yksityiskohtaisuuden muuttamisen

LTS: *Long Term Support*. Tuotteen pitkäaikainen tuki, jolloin tuotteessa havaitut virheet korjataan pitkäjaksoisesti.

Objekti: Tason tai pelin sisällä oleva asia tai esine.

OC: *Occlusion Culling*. Mekanismi, joka mahdollistaa objektin piirron estämisen, kun objekti jää toisen objektin taakse piiloon.

Suoratoisto: Pelimaailman jatkuva latautuminen ilman pelin keskeytymistä.

Taso: Pelimaailman osa tai kenttä. Unityssä käytetään englanninkielistä termiä *scene*.

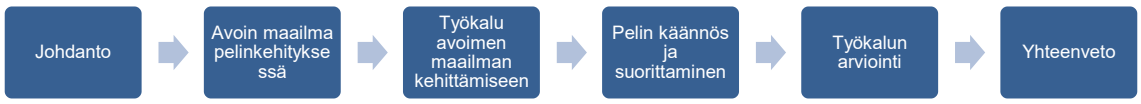
1 Johdanto

Insinöörityön tarkoitus oli luoda työkalu avoimen 3D-maailman luontiin Unityssä ja raportoida siitä saaduista tuloksista ja päätelmistä. Raportissa esitellään aineistoa, työssä käytettyjä menetelmiä ja työn kulkua sekä tarkastellaan tuloksia ja etsitään mahdollisia virheitä.

Työn toiminnallisessa toteutuksessa tehtiin työkalu Unityyn. Työkalu kirjoitettiin Unity-pelimoottorin käyttämällä C#-ohjelmointikielellä. Työkalu sisälsi avoimen maailman suoratoistajan komponentteineen, erilaisia editoria laajentavia aputyökaluja avoimen maailman komponenttien ja tasojen hallintaan sekä työkalulla toteutetun minimalistisen avoimen maailman esimerkkipelin. Työkalu implementoitiin koemielessä myös vanhaan henkilökohtaiseen peliprojektiin, jotta työkalun kelvollisuutta pystyttiin arvioimaan luotettavammin.

Unity valittiin työkalun alustaksi sen vahvan markkina-aseman vuoksi. Unityä käytetään yli 190 maassa ja yli 20 alustassa. Vuonna 2021 yli puolet mobiili-, konsoli- tai tietokonealustoilla julkaistuista peleistä sekä 72 prosenttia tuhannesta suosituimmasta mobiilipelistä tehtiin Unityllä. (1.) Koska Unity valittiin alustaksi, sitä painotetaan raportin eri vaiheissa.

Raportissa käydään läpi kuvan 1 mukaisesti työn aikana tehdyt havainnot.



Kuva 1. Raportin rakenne.

Työn toisessa luvussa kerrotaan avoimen maailman kehityksestä sekä avoimesta maailmasta käsitteenä. Tämän luvun tarkoituksena on antaa lukijalle hyvä kuva avoimen maailman suunnittelun lähestymiskulmista pelikehittäjän näkökulmasta. Lisäksi pohditaan mahdollisia avoimen maailman pelikehitykseen liittyviä ongelmia.

Kolmannessa luvussa esitellään tämän insinöörityön tuloksena syntynyt työkalu avoimen maailman kehittämisen apuna. Tässä luvussa käydään vaihe vaiheelta avoimen maailman kehittämisen apuun tehdyn työkalun käyttöä.

Neljännessä luvussa esitellään työkalulla tehdyn pelin kääntäminen ja suorittaminen. Tässä luvussa käydään läpi avoimen maailman suoratoistaminen, kääntämisen ja pelitilan esivalmistelut sekä pelin kääntäminen. Luvussa havainnollistetaan maailman käynnistämistä esimerkkikoodien avulla.

Viidennessä luvussa on työkalun arviointi. Tarkoituksena on arvioida insinöörityön tuloksena syntynyttä työkalua, sen vaikutusta avoimen maailman pelikehityksen työskentelyyn ja työkalun suorituskykyä sekä esitellä mahdollisia jatkokehitysideoita.

2 Avoin maailma pelinkehityksessä

Tässä luvussa käydään aluksi läpi, mitä avoin maailma tarkoittaa pelimekaniikkana ja annetaan muutamia esimerkkejä toteutuksista olemassa olevissa peleissä. Seuraavaksi pohditaan erilaisia näkökulmia avoimen pelimaailman toteuttamiseksi pelinkehittäjän näkökulmasta. Tapoja maailman suoratoistamiseksi (engl. stream) mainitaan kaksi: etäisyyteen ja tapahtumiin perustuva suoratoistaminen. Lopuksi pohditaan avoimen maailman peleissä pelinkehittäjän kohtaamia haasteita, kuten maailman täyttäminen yksityiskohdilla ja tapahtumilla luontevasti, suorituskyvyn rajoitteet ja projektityöskentelyn sujumattomuus.

2.1 Avoin maailma

Avoin maailma (engl. open world) pelimekaniikkana tarkoittaa virtuaalimaailmaa, jossa pelaaja voi pelin sääntöjen puitteissa vapaasti liikkua ja tutkia maailmaa. Pelin maailma latautuu pelin aikana pelaajan liikkuessa maailmassa ilman, että alueelta toiselle siirryttäessä tulee ruutuun näkyville latausikkuna. Avoin maailma antaa pelaajalle vapauksia suorittaa annettuja tehtäviä

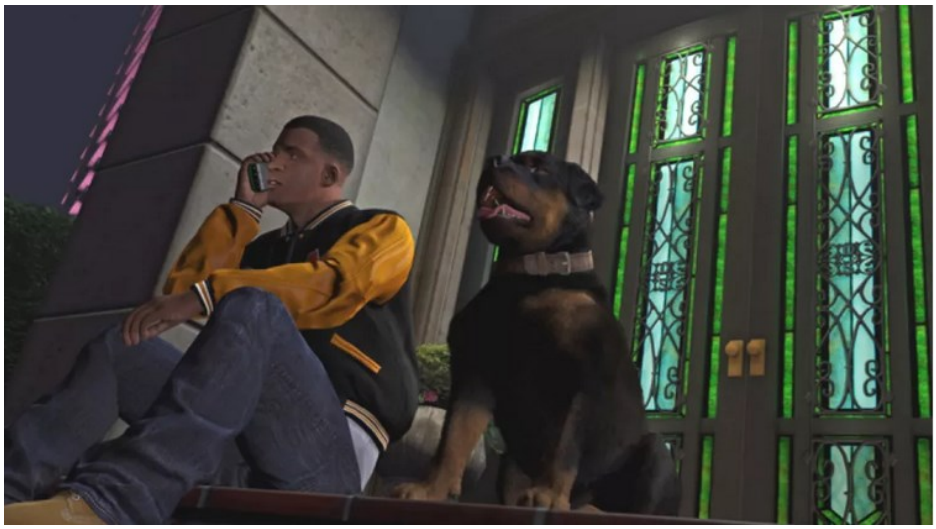
haluamassaan järjestyksessä. (2.) Tyypillisesti avoimen maailman pelissä ei välttämättä jokaiselle alueelle tarvitse mennä eikä jokaista sivutehtävää suorittaa päästäkseen pelin läpi.

Laadukkaan pelin tekijältä vaaditaan kehitysresursseja ja luovuutta maailman täyttämiseksi. Tyypillisesti valveutunut pelaaja havaitsee pelissä useasti toistuvia elementtejä, joita on vaikea välttää kehittäjän haasteiden lisäksi myös pelin kohdeyleisön tietokoneiden asettamien teknisten rajoitusten takia. Pelikehittäjät voivatkin joutua tekemään kompromisseja pelin visuaalisen sisällön suhteen parantaakseen pelin suorituskykyä ja piilottaakseen sisällön suoratoistamisesta johtuvia latauspiikkejä. Usein avoimen maailman peleissä ovat ongelmina myös tyhjyys, maailmassa kulkemisen vaivalloisuus, turhalta tuntuvat tehtävät ja määrä-ennen-laatua-ajattelutapa. (3.)

Avoimen maailman ideaalitilanteessa peli on suuri ja monipuolinen seikkailu, jossa pelaaja tutkii loputtamalta tuntuvia monipuolisia alueita rikkaassa maailmassa lukuisine uniikkeineen hahmoineen, tehtävineen ja yksityiskohtineen (4).

GamesRadar on julkaissut tuoreen, sen mukaan tämän hetken parhaita avoimen maailman pelejä käsittelevän artikkelin. Artikkelissa olevassa listassa mainitaan muun muassa pelit Grand Theft Auto 5 (julkaistu 2013) ja Red Dead Redemption 2 (julkaistu 2018). (5.)

Grand Theft Auto 5 -pelissä (kuva 2) pelaaja pääsee rötöstelemään monipuolisessa maailmassa. Pelissä voi hurjastella erilaisilla ajoneuvoilla, tehdä mielipuolisia rikoksia ja suorittaa erilaisia tehtäviä. (5.)



Kuva 2. Grand Theft Auto 5 -pelin kohtaus (5).

Red Dead Redemption 2 -pelissä (kuva 3) pelaaja pääsee leikkimään villin lännen sankaria. Pelin päähahmo kuuluu lainsuojattomien ryhmään ja aiheuttaa erilaista hämmennystä pelimaailmassa. Pelissä on suuri ja monipuolinen maailma, jossa on paljon erilaisia tehtäviä. Maailmassa pelaaja kokee villin lännen aikalaisten ihmisten haasteita ja yhteiskunnan kehittymisen kiihtyvän. (5.)

Kuva 3. Red Dead Redemption 2 -pelin kohtaus (5).

Molemmat pelit ovat pelattavan maailman osalta suurehkoja, ja niissä liikkuminen maailman päästä päähän ilman ajoneuvoa vie aikaa. Grand Theft Auto

5:ssä pelattava alue vastaa oikean maailman pinta-alassa noin 80 km²:n aluetta. Red Dead Redemption 2:ssa se vastaa noin 75 km²:n aluetta. (6.)

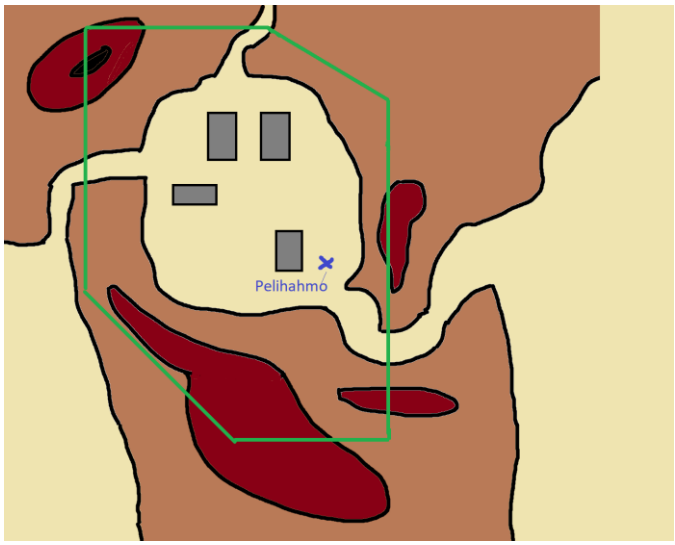
Game Rantin kymmenestä pienestä avoimen maailman pelistä kertovan artikkelin pienin peli vastaa oikean maailman pinta-alassa 0,26 km²:n suuruista pelattavaa aluetta. Listan suurimman pelin pinta-ala vastaa 7,8 km²:n aluetta. Suurin peli listalla on mielenkiintoisesti Grand Theft Auto 3 (julkaistu 2001). Grand Theft Auto -pelisarjan viides osa onkin siis noin kymmenen kertaa suurempi pelattavalta alueeltaan kuin kolmas osa. (7.)

2.2 Suunnittelun lähestymiskulmat

Avoimen maailman suoratoistamisessa on perinteisesti kaksi metodia: etäisyyteen tai tapahtumiin perustuva pelimaailmojen lataus. Näiden metodien avulla pelinkehittäjä määrittelee, mitkä pelimaailman osat milloinkin ladataan. (8.)

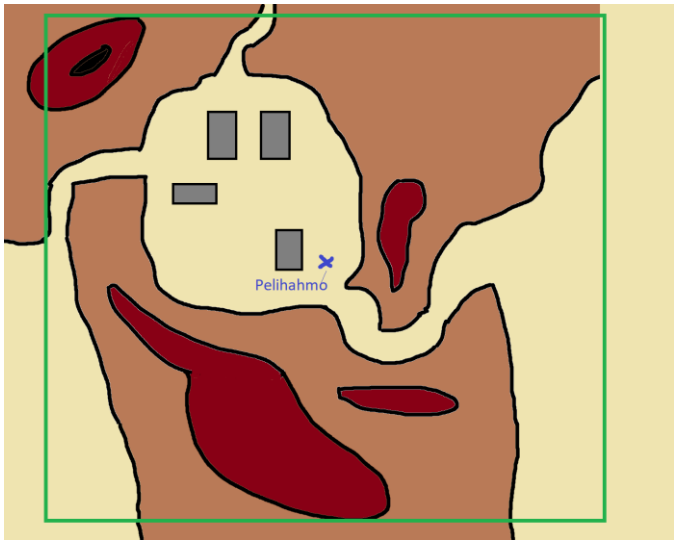
Etäisyyttä käytettäessä jatkuvasti tarkastetaan, mitkä kohteet ovat tietyn etäisyyden säteellä pelaajan kamerasta. Tällöin ladataan tai poistetaan pelialueita etäisyyden mukaan. Tapahtumia käytettäessä kohteiden lataaminen ja poistaminen on sidottu logiikassa tapahtuviin muutoksiin, kuten esimerkiksi pelaajan astuessa kytkimen (engl. trigger) törmäyksentarkistuskomponenttiin tai esimerkiksi pelissä tapahtuvaan juoneen, kuten portin aukeamiseen tai tehtävän suorittamiseen. (8.)

Yleistäen tapahtumiin perustuva toiminta voi mahdollistaa paremman suorituskyvyn tai suuremman määrän yksityiskohtaisempaa sisältöä tiheämmin sijoitettuna, koska se tarjoaa paremman kontrollin ladattavan sisällön määrään kullakin hetkellä. Tämä kuitenkin vaatii, että kehittäjä kykenee tehokkaasti suunnittelemaan pelialueet suorituskyvyn ja näkyvyyden pohjalta sekä huomioi sudenkuoppia, kuten suuria tasankoja tai alueita, joista näkee useita muita alueita. Kuvassa 4 on esimerkki tapahtumilla ohjatun maailman ladattavan pelialueen rajoista.



Kuva 4. Tapahtumametodilla ladattu maailma. Vain näkyvien tasojen osat ladattiin. Sininen x vastaa pelihahmoa ja vihreä viiva ladattavaa aluetta.

Etäisyyteen perustava toiminta ei itsessään juurikaan mahdollista aluekohtaista optimointia, koska silloin tasot ladataan pelkästään etäisyyden mukaan. Etäisyyden käyttö voi vapauttaa pelimaailman suunnittelua isossa kuvassa, ja silloin menetetään pelin suorituskykyä tai sisällön tiheyttä ja laatua. Kuvassa 5 on esimerkki etäisyyden perusteella ohjatun maailman ladattavan pelialueen rajoista.



Kuva 5. Etäisyyttä käyttämällä ladattu maailma. Pelihahmon ympäriltä piirretään tasaisen etäisyyden päästä alueita. Etäisyys valittiin siten, että se riittää katkaisemaan näkyvyyden kaikkiin suuntiin kaikissa pelin tilanteissa. Sininen x vastaa pelihahmoa ja vihreä viiva ladattavaa aluetta.

Metodin valintaan vaikuttavat erilaiset tekijät. Tekijöitä yhdistelemällä voi saada lukemattoman määrän erilaisia yhtälöitä suoratoistometodin valintaan. Lopulta pohdinta jää pelinkehittäjän tehtäväksi. Pelinkehittäjä voi pohtia seuraavia kysymyksiä:

- Näkeekö kamera vain ylhäältä alas?
- Onko kamera pelihahmon luona ja vapaasti pelaajan käänneltävissä?
- Liikkuuko pelaaja maan pinnalla vai voiko hahmo lentää esimerkiksi vuorien ylitse?
- Kuuluuko pelaajan nähdä tyhjiä latautumattomia alueita vai tavoitellaanko maailman saumatonta jatkumista?
- Kuinka suuri on pelimaailman pinta-ala?
- Kuinka tiheästi maailmassa on sisältöä ja kuinka yksityiskohtaista sisältö on?
- Onko pelimaailmassa selkeitä reittejä esimerkiksi vuorien välistä ja kaupunkien läpi, joilla voi katkaista näkyvyyttä?
- Onko pelimaailma pelkkiä suuria avoimia alueita, kuten tasankoja, metsiä, soita, merta tai avaruutta?
- Voidaanko käyttää tehosteita, kuten sumua, rajoittamaan näkyvyyttä?
- Näkeekö pelaaja samanaikaisesti toisia suuria alueita, esimerkiksi ollessaan vuoren huipulla?
- Halutaanko pelimaailman alueista näyttää erilaisia versioita, esimerkiksi pommituksen jälkeen?

Selkeää kaavaa metodin valintaan on mahdotonta tehdä, sillä vastaus on aina projektikohtainen. Taulukossa 1 on arvioitu erilaisia pelikonsepteja, joihin voidaan pelejä suunnitellessa törmätä. Taulukossa vasemmalla puolella selvitetään pelin toiminnallisuutta ja käyttäjätarinaa ja oikealla ehdotetaan mahdollista lähestymistapaa pelin suoratoistamiseen.

Taulukko 1. Erilaisia pelikonsepteja ja niille ehdotettuja suoratoistometodeja.

Pelikonsepti	**Etäisyys**	**Tapahtuma**
Kamera sivusta. Kaksiulotteinen maailma.	X	
Kamera ylhäällä. Jatkuva maailma.	X	
Kamera ylhäällä. Rajatut alueet ja näkyvyys. Esimerkiksi luolia tai piilotettuja alueita.		X
Kamera maantasossa. Monipuolinen katseen katkaiseva maailma. Ns. perinteinen kolmiulotteinen seikkailupeli.		X
Kamera maantasossa. Maailmassa vähän yksityiskohtia, eikä suorituskyky huoleta.	X	
Kamera maantasossa. Maailmassa edetään putkimaisesti.	X	
Kamera maantasossa. Maailma tasankoa. Sumulla katkaistaan näköetäisyys tai ei välitetä ladattavien asioiden ilmestymisestä.	X	
Kamera lennossa. Maailma tunnelimainen tai vuorien yli ei pääse.		X
Kamera lennossa. Voidaan lentää vapaasti.	X	
Pelin alueissa on suurta satunnaisuutta, jota on vaikea ennakoida.	X	

Metodeja voi hypoteettisesti erityistilanteissa myös yhdistellä. Yksi vaihtoehto voisi olla, että tietyn korkeuskoordinaatin alapuolella käytettäisiin kytkimiä ja yläpuolella etäisyyttä. Toinen vaihtoehto voisi olla, että isot linjaukset, kuten maastot, tehdään kytkimillä ja pienemmät tason objektit etäisyyden perusteella.

Ennen päätöstä on hyvä myös tietää, että moderneissa pelimoottoreissa, kuten Unity ja Unreal Engine, on jo molempia metodeja muistuttavat sisäänrakennetut mekanismit. Mekanismit Level Of Detail (LOD) ja Occlusion Culling (OC)

estävät tilanteen mukaan objektin piirtämisen. Tasojen lataamiseen verrattuna mekanismit kasvattavat objekteihin käytettävän muistin määrää. Keskusyksikön jatkuva työkuorma myös kasvaa, koska objekteja arvioidaan jatkuvasti. Tasojen latauksessa latauksen aiheuttama kuorma tapahtuu kerralla. (10; 11; 12.) Joka tapauksessa ennen metodin lopullista valintaa kannattaa kehittäjän huolellisesti tutustua aiheeseen ja puntaroida edellä mainittujen mekanismien hyötyjä ja haittoja. Sekä LOD että OC toimivat kummankin metodin kanssa, ja molemmat todettiin toimivaksi ja hyödylliseksi työkalun testauksessa.

LOD-mekanismi mahdollistaa objektien yksityiskohtaisuuden muuttamisen etäisyyden käsityksen mukaan. Todellisuudessa LOD perustuu siihen, kuinka monta prosenttia objekti peittää ruudusta. LOD mahdollistaa myös objektin näkyvyyden piilottamisen kokonaan. Objekti säilyy silti tason muistissa niin kauan, kuin taso on ladattuna. (10.) LOD-mekanismin voi tunnistaa monia moderneja kolmiulotteisia pelejä pelatessa, ja se tuntuu olevan pelialalla yleinen tapa.

Occlusion Culling -mekanismi mahdollistaa objektien piilottamisen ennalta laskettujen kuutionmuotoisten volyymien mukaan. Yksinkertaistetusti periaatteena on suorittaa esirakentamisprosessi, joka jakaa maailma kuutioihin ja tallentaa jokaiseen kuutioon tiedon, mitä muita kuutioita se voi nähdä. Pelaamisen aikana kamera selvittää, minkä kuution sisällä kamera on, ja piirtää kuution määrittelemien kuutioiden sisällä olevat objektit. Mekanismi voi projektista riippuen olla tehokas suorituskyvyn kasvattaja tai hyödytön muistin haaskaaja. (11.)

Sekä tapahtumia että etäisyyttä latausperusteena käytettäessä voidaan pelimaailmaa optimoida, tilanteen salliessa, lataamalla yksinkertaistettuja versioita tasoista kaukaisille maailman osille. Tasot voidaan vaihtaa alkuperäisiin tasoihin niitä lähestyttäessä.

Tapahtumiin perustuvan algoritmin käyttö pelissä lisää pelinkehittäjän työmäärää. Työskentelyvaiheessa tapahtumien ja logiikan toteuttamien, niiden perusteella tehtävät tasojen valinnat sekä taustalla suorittava algoritmi lisäävät työmäärää, sillä ne pitää erikseen toteuttaa. Etäisyysalgoritmiin verrattuna työmäärä on tapahtumia käytettäessä huomattavasti suurempi. (9.)

2.3 Kehitystyön haasteet

Tyypillisesti avoimen maailman peleissä haasteina ovat suuren maailman täyttämisen vaikeudesta aiheutuva toistuvuus tai tyhjyys. Nämä haasteet voivat johtua esimerkiksi peliä kehittävän ryhmän pienestä budjetista ja luovuuden tai taidon puutteesta. Käytettävä teknologia voi tuoda myös teknisiä rajoitteita pelin suorituskyvyn ja muun toteutuksen suhteen. (3.)

Projektityöskentely voi myös tuottaa haasteita muun muassa konfliktien tai versioiden erojen myötä. Samojen tiedostojen muokkaus on esimerkiksi Unityssä huonosti tuettua ja tasojen osittaminen vaikeuttaa maailman visualisointia ja altistaa virhetilanteille. Yksi kehittäjä voisi esimerkiksi siirtää objektia, mikä vaikuttaisi toiseen tasoon. Kehittäjä ei kuitenkaan voisi välttämättä vapaasti koskea toiseen tasoon, jos toinen kehittäjä muokkaisi sitä samanaikaisesti. (8; 13.)

Avoimen maailman kehitystyössä maailman osat eli tasot ovat keskeisessä roolissa. Unityn tasot ovat oletuksena binäärimuotoisia, eikä niiden versionhallintaan ole ollut hyvää tapaa. Unityn oma versionhallinta ei tunnista varsinaisia muutoksia, vaan koko tiedosto yliajetaan. Usean kehittäjän muokatessa samaa tiedostoa voi syntyä työläästi ratkaistavia tai ratkaisemattomia konflikteja. Unityssä on myös keino tallentaa tasojen tiedostot tekstimuotoisena käyttäen YAML-teknologiaa, mutta tason lataaminen tekstimuotoisena on binäärimuotoista hitaampaa. (13.)

Avoimen maailman projektityöskentelyn haasteita on käsitellyt vuoden 2016 GDC-konferenssissa Jane Ng. Esitelmässä Ng kertoo Firewatch-pelistä, jonka pääartisti ja yhteistuottaja hän oli. Firewatch (julkaistu 2016) on kolmiulotteinen avoimen maailman seikkailupeli. Esitelmä koskee pelin maailman luomista, ja sen aikana Ng kiinnittää huomiota tiiminsä yhteystyöhön ja korostaa edellisessä kappaleessa mainittuja Unityn ongelmia tasojen hallinnassa. Firewatch-projektissa käytetään olemassa olevaa kaupallista työkalua nimeltään Sectr. Sen avulla tiimi pilkkoo maailman pienempiin osiin editorissa muokkausta varten. Pelin aikana pelaaja kulkee kytkinten läpi määrittäen ladattavat alueet, minkä jälkeen Sectr yhdistää osat yhdeksi tasoksi. Pääsyy Sectr-työkalun käyttöön on

puhujan mukaan tiimin käyttämä vanhempi Unity-versio, johon ei löydy valmiiksi järkeviä ratkaisuja tasojen osittamiseen tai useiden tasojen lataamiseen. (8.)

Unityn versiossa 5.3 (julkaistu 2015 joulukuussa) esiteltiin SceneManager-rajapinta, joka mahdollistaa useiden tasojen lataamisen pelimoottorissa (14). Rajapintaa käytetään tässä työssä pelimaailman suoratoistamiseen. Unityn editori tukee useiden tasojen aukipitämistä, joten maailman osittaminen voidaan suorittaa perinteisillä tasotiedostoilla. Kaikki tasot käyttävät aina samaa koordinaatistoa, jolloin tason objektit voidaan kohdistaa helposti oikein toisen tason objekteihin nähden. (15.)

3 Työkalu avoimen maailman kehittämiseen

Tässä luvussa kerrotaan tärkeimpiä teknisiä yksityiskohtia insinöörityön projektina syntyneestä työkalusta. Työkalu tehtiin helpottamaan ja nopeuttamaan avoimen 3D-maailman pelien kehittämistä. Luvussa esitellään projektin suunnittelua ja ominaisuuksia. Teknisten toteutusten osalta käydään läpi tasojen viittausten käyttämää abstrahointia, kytkinten käyttämistä suoratoistamisessa ja aloituspisteiden hyödyntämistä maailman käynnistämiseksi. Lisäksi esitellään avoimen maailman komponenttien lisäämisen ja muokkaamisen helpottamiseksi tehtyjä aputyökaluja, jotka laajentavat editorin toimintaa. Aputyökaluina tehtiin tasojen ja kytkinten hallintavalikoita, asetusten muokkausikkuna sekä aloituspisteiden lisäys- ja muokkaustyökalu. Laajennuksia tehtiin myös minimoimaan virhetilanteita sekä parantamaan editorin visuaalista selkeyttä ja työskentelyn käyttökokemusta.

3.1 Työkalun ominaisuudet

Työkalun ensisijainen tehtävä oli helpottaa ja nopeuttaa avoimen maailman toteuttamista sekä helpottaa tasojen ja pelin muiden avoimeen maailmaan liittyvien komponenttien hallintaa. Avoimen maailman toteutukseen luotiin suoratoistaja, joka käyttää kytkimiä tasojen viittausten kanssa. Maailman

käynnistämiseksi tehtiin järjestelmä, jolla voidaan tallentaa aloituspisteitä ja ladata maailma tallennetusta aloituspisteestä.

Avoimen pelimaailman kehityksen helpottamiseksi luotiin valikkotyökalu, jonka avulla maailman voi muokkaustilassa avata helposti simuloimaan eri tilanteita pelissä. Kytkinten muokkaamista varten luotiin valikkotyökalu, jolla kytkimiä voi etsiä, piilottaa ja näyttää sekä korvata niiden tasoviittauksia. Maailman asetuksien muokkaamiseksi luotiin asetusikkuna. Myös aloituspisteen visualisoimiseen, muokkaamiseen ja lisäämiseen luotiin oma aputyökalu.

Työkalu suunniteltiin tukemaan kahta viimeisintä Unityn Long Term Support -versiota (LTS), jotka kirjoitushetkellä olivat 2019.4.37f1 ja 2020.3.32f1. LTS tarkoittaa Unity-version pitkäaikaista tukea, ja versiot saavat editorin virheiden korjauspäivityksiä. Unity suosittelee verkkosivuillaan vakavasti otettavassa tuotannossa olevia projekteja käyttämään LTS-versiota (19). LTS-versiota käytettäessä toiveena oli työkalun tukevan mahdollisimman montaa peliprojektia. Muiden versioiden tukea arvioitiin seuraamalla Unityn ohjelmointirajapinnan dokumentaatiota. Dokumentaatio kertoisi jonkin asian olevan vanhentunut tai vanhentumassa. Vanhentumista ei kuitenkaan havaittu projektissa käytettävien ominaisuuksien osalta.

Työkalussa haettiin inspiraatiota Jane Ngin GDC-puheessaan esittelemästä Firewatch-pelistä (8). Puhe esiteltiin tarkemmin luvussa 2.3. Tässä projektityössä oli kuitenkin ajatus lähestyä avoimen maailman toteutusta SceneManager-rajapinnan näkökulmasta ja tehdä oma kokonaisvaltainen ratkaisu tasojen ja avoimen maailman komponenttien hallintaan maailmaa muokatessa.

Työkalun avoimen maailman kohdepelisovellutukseksi valittiin ensimmäisen tai kolmannen persoonan peli, jossa pelihahmo liikkuu maan tasolla ja vuoria tai kaupunkeja voidaan käyttää alueiden jakamiseksi. Kohdepelin yksityiskohtia ja pinta-alaa haluttiin peliin mahdollisimman paljon, jolloin aluekohtaista suoratoistamisen optimointia tarvittaisiin. Yleisvaikutelma haluttiin pitää kohdepelissä realistisena ja säilyttää pelaajan immersio pelaamisen aikana. Jotta pelaajan

käsitys pelimaailmasta säilyisi yhtenäisenä, ei pelaajan kuuluisi nähdä tasojen latautumista.

Kohdepelissä voisi tulla myös tilanteita, joissa esimerkiksi vuoren huipulta voisi nähdä alueita, joihin pelaajalla ei ole senhetkisestä paikasta pääsyä. Tällöin pelinkehittäjä voisi käyttää yksinkertaistettua versiota sillä hetkellä saavuttamattomissa olevista tasosta. Pelaaja edelleen näkisi tason olemassaolon käyttämättä liikaa ylimääräisiä resursseja tietokoneestaan.

Edellä mainitut vaatimukset jättivät vaihtoehdoksi tapahtumiin perustuvan suoratoistamisen metodin. Tällöin pelin suoratoisto tapahtuisi kytkimiä käyttämällä. Johtopäätökseen päädyttiin luvun 2.2 pohdintojen perusteella. Kytkinten hallinnan helpottamaiseksi tehtiin myös valikkotyökalu Unity-editorin laajennuksena.

Työssä esitellyn työkalun tarkoitus ei ollut aukottomasti tehdä joka tilanteessa toimivaa avoimen maailman pelialustaa. Erilaisilla peliprojekteilla on erilaisia vaatimuksia, ja tällöin käytetään eri keinoja toteuttaa ja optimoida peliä. Suunnittelussa valittiin lähestymistavaksi tasojen lataaminen asynkronisesti. Sen on tarkoitus toimia joissain projekteissa koko ratkaisuna ja joissain loogisena lähtötilanteena sekä avustajana avoimen maailman toteutuksessa.

Tasojen lataamiseen päädyttiin, koska Unityn tuki useiden tasojen lataamiseen on hyvä. Lisäksi useiden tasojen muokkaaminen on käytännöllistä muokkaustilassa. Unity myös tukee muita mekanismeja useiden tasojen ollessa auki kerrallaan, kuten valon ja navigaatioverkon esirakentamista ja lataamista. (15.) Tasojen lataus osoittautui myös toimivaksi tavaksi alustavissa testeissä.

Tasojen suoratoistamisessa sisältö poistetaan muistinkeräilijän toimesta muistista, mikä voi aiheuttaa pientä lisätyötä prosessorille hetkellisesti. Jos sen sijaan suoratoistettaisiin objekteja ja ne piilotettaisiin, säilyisi niistä silloin tieto muistissa, mikä rajoittaisi pelimaailman kokoa. (16.) Tasojen suoratoistamisen käyttämisestä Unityllä ovat raportoineet myös ainakin Spellcast Studios (17) ja Myriad Games Studio (18).

Pelinkehittäjä saa aloitustyökalun, jolla hän pääsee aloittamaan ja testaamaan avoimen maailman peliprojektin vaatimuksia. Työkalua muokkaamalla pelinkehittäjä voisi lähteä tarvittaessa esimerkiksi kokeilemaan erilaisia tapoja sisällön suoratoistamiselle käyttäen kytkimiä. Tasojen sijaan pelinkehittäjä voisi haluta ladata esimerkiksi Prefab-, AssetBundle- tai Addressable-tyyppisiä paketteja pelkästään tai Scene-tiedostojen ohella (20). Pelinkehittäjä voisi myös haluta rakentaa omanlaisia koontityökaluja. Työkalun muokattavuus huomioitiin suunnitelmassa ja toteutuksessa. Tasojen abstrahointia ja suorakytkimiä voidaan muokata käyttämään objekteja tai paketteja tasojen sijaan.

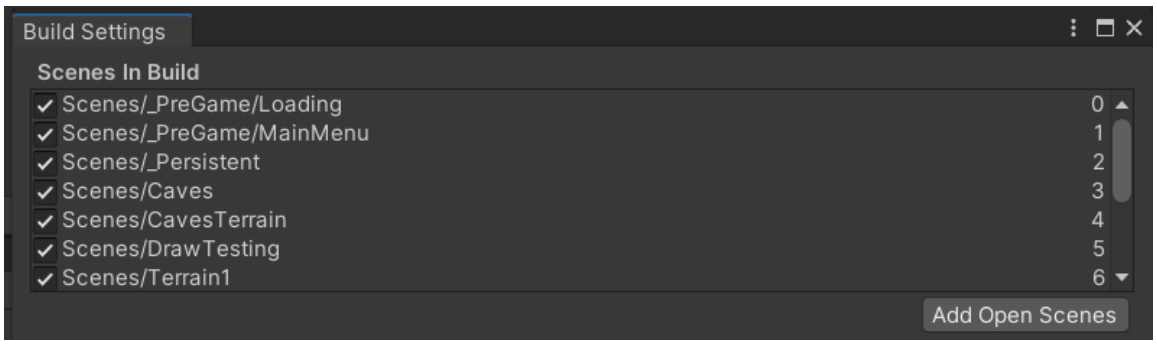
Työkalulla oli tarkoitus pystyä myös hallitsemaan tasoja nopeasti ja tehokkaasti editorin muokkaustilassa. Erilaisia tilanteita, joihin pelitilassa päästään suoratoiston kautta, haluttiin pystyä simuloimaan nopeasti. Tähän tarkoitukseen tehtiin editoria laajentavia uusia valikoita, painikkeita, ikkunoita, visuaalisia muokkauksia ja muita lisäosia. Kuitenkin ajatus työkalun monipuolisuuden kannalta oli tehdä muokkaustilan tasojenhallintavalikosta sellainen, että sitä voisi käyttää itsenäisesti ja ilman avoimen maailman komponentteja. Tällöin tasojen hallintaa voisi hyödyntää projektissa, jossa on esimerkiksi paljon tasonkehittäjiä tai suuria tasoja. Suurien tasojen osalta tasot voitaisiin esimerkiksi osittaa. Eri kehittäjät voisivat muokata saman tason eri osia huolettomasti samanaikaisesti. Näin voitaisiin myös vähentää konflikteja samassa projektissa työskentelevien eri pelikehittäjien välillä. Silti pelinkehittäjä voisi helposti editorissa visualisoida koko tason ja sen eri osia.

Muokkaustilan tasojenhallintavalikko oli tarkoitus haluttaessa voida eriyttää kokonaan suoratoistajasta. Tämän seurauksena sovelluskaupassa myyntiin voitaisiin laittaa kolme eri versiota työkalusta: pelkkä avoimen maailman suoratoistaja komponentteineen, pelkkä tasojenhallintavalikko tai molemmat yhdessä.

3.2 Tasojen abstrahointi

Tasoja ladattaessa tarvitaan jonkinlainen viittaus tasoon, jonka avulla Unity-pelimoottori löytää tason. Unity luo pelitilan käynnistyessä jokaiselle tasolle Scene-luokan olion. Olio sisältää identifioivat tiedot tason löytämiseksi. (21.)

Tasojen identifioivina tietoina toimivat tasojen käännöskokoonpanon (kuva 6) indeksi kokonaislukuna (build index) sekä kaksi merkkijonomuotoista muuttujaa: nimi (name) ja polku (path). (21.)



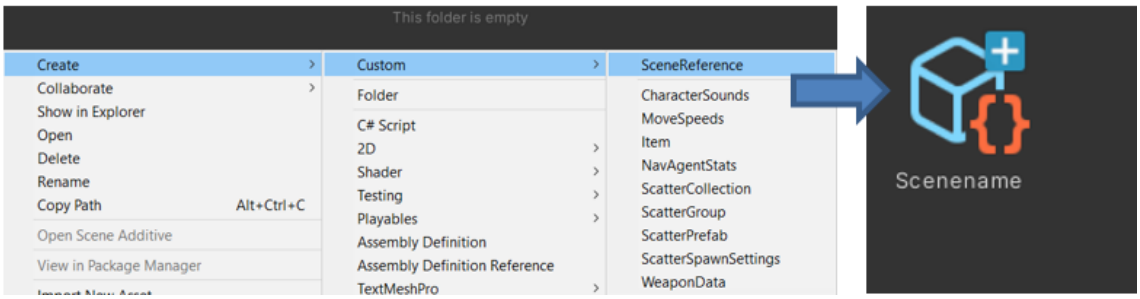
Kuva 6. Käännettävien tasojen polut ja indeksit käännöskokoonpanossa.

Tasoja voidaan ladata editorin pelitilassa ja käännetyssä pelissä käyttäen edellä mainittuja identifioiva tietoja. Tiedoista vain indeksi ja polku lataavat tasot eksplisiittisesti. Pelkän nimen käyttäminen lataa ensimmäisen löydetyn samannimisen tason (22). Muokkaustilassa eli Scene- ja Hierarchy-ikkunoissa tasoja voidaan avata vain täyttä polkua käyttämällä (23).

Työkalussa ja sillä tehdyssä peliprojektissa tasojen viittauksia käytetään monissa paikoissa. Tasojen viittaukset voivat erityisesti toistua useita kertoja eri kytkimissä. Jos indeksiä tai polkua käytettäisiin tasoviittauksina, se johtaisi pelinkehittäjän työn lisääntymiseen ja altistaisi virhetilanteisiin. Esimerkiksi tason nimeä tai järjestystä käännöskokoonpanossa muutettaessa pitäisi muunnos korjata lukuisiin paikkoihin. Vaihtoehtoisesti taso voidaan haluta tilapäisesti poistaa käännöskokoonpanosta, mutta säilyttää silti viittaus kytkimissä ja peliprojektin

asetuksissa. Tasojen abstraktiokerros työkalussa on ratkaisu edellä mainittuihin ongelmiin.

Työkalun SceneReference-luokka toimii abstraktiokerroksena tasojen ja pelitilan suoratoistajan välillä ja viittauksena tasoon. Luokalle voidaan luoda instansseja Create-valikon kautta kuvan 7 mukaisesti, minkä seurauksena syntyy vapaavalintaiseen sijaintiin ScriptableObject-tyyppinen asset-tiedosto.



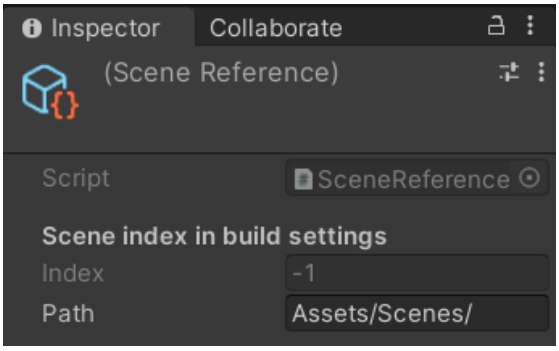
Kuva 7. Viittaustiedoston luonti ja luotu tiedosto.

Seuraavaksi viittaustiedosto nimetään samannimiseksi kuin viitattava taso. Tasot ovat Unity-tyyppisiä tiedostoja. Viitattavan tason tiedoston sijainti tarkistetaan. Viittaustiedoston path-muuttuja laitetaan vastaamaan tason tiedoston sijaintia. Vaihtoehtoisesti voidaan molemmissa käyttää työkalun määrittämää oletussijaintia *Assets/Scenes/*.

Jos tiedoston nimeä vastaavaa tasoa ei löydy käännöskokoonpanosta, antaa SceneReference-luokka varoituksen tiedoston puutteesta muun muassa sitä muokatessa tai editorin pelitilaan siirryttäessä. Puutteellinen viittaustiedosto voidaan haluttaessa jättää käyttöön työkalun muihin osiin, jolloin sen viittaus hylätään ja puutteellisuus ilmoitetaan editorissa operoitaessa. Näin muut työkalun osat jatkavat toimintaa tavalliseen tapaan kaikissa tilanteissa.

Viittaustiedosto sisältää tiedon tason käännöskokoonpanon indeksistä, ja työkalu sekä kohdistaa että ylläpitää indeksin automaattisesti Unityn muokkaustilassa. Indeksin muokkaaminen manuaalisesti pelinkehittäjän toimesta on virhealtista, joten sen estämiseksi tehtiin editoriin laajennus käyttäen

PropertyAttribute- ja CustomPropertyDrawer-rajapintoja. Laajennus mahdollistaa indeksin näyttämisen editorissa, mutta estää muokkaamisen, kuten kuvassa 8 näkyy. Indeksi näkyy vain viittaustiedoston Inspector-ikkunassa.

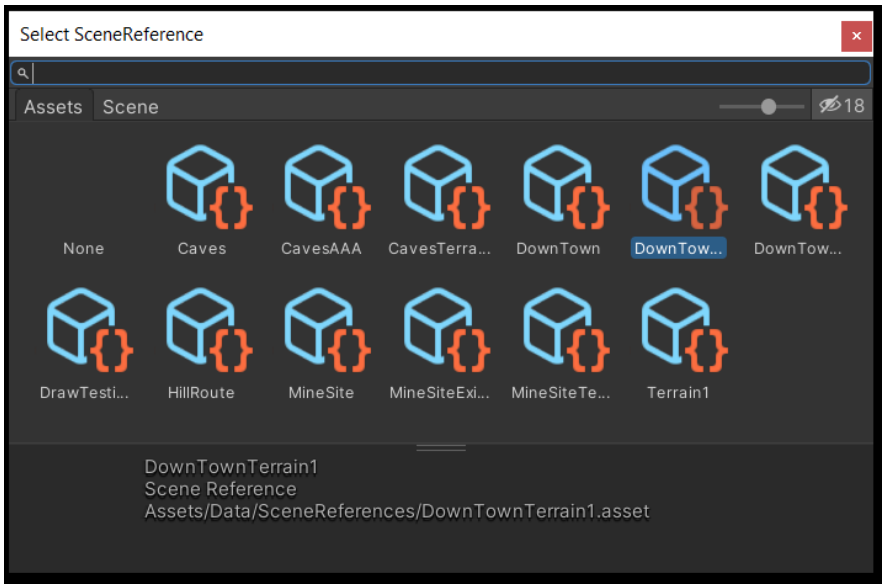


Kuva 8. Viittaustiedosto Inspector-näkymässä. Indeksin muokkaaminen on estetty. Indeksin arvo on -1, koska Scenename-nimellä ei löydy tasoa.

Kun viittaustietoja käsitellään muissa komponenteissa, tulee aina tason nimi käyttöön ja näkyviin. Kuvassa 9 näkyvät viittaustiedot kytkimen taulukossa. Näkymä on yksinkertainen ja nopeasti luettavissa.

Kuva 9. Viittaustiedostot kytkimen taulukossa.

Viittaustiedostojen käyttämisessä pääteltiin myös eduksi, että tasojen kehittäjän ei tarvitse osata koodata ja kuka tahansa kehittäjistä voi luoda viittaustiedostoja. Viitattavien tasojen valitseminen kytkimissä on myös helppoa. Editori tuo kaikki mahdolliset vaihtoehdot näkyviin listauksena (kuva 10), jolloin pelinkehittäjä voi valita ainoastaan olemassa olevan viittaustiedoston.



Kuva 10. Valittavissa olevat viittaukset samassa ikkunassa.

Pelitilassa tasoja ladataan työkalussa käännöskokoonpanon indeksin perusteella, koska kokonaislukujen vertailun ja hallinnan havaittiin olevan tehokkaampaa verrattuna merkkijonojen käyttöön. Tasojen viittauksia säilytetään taulukkoina, ja niiden hallintaa varten on luotu SceneReferenceList-luokka. Luokka sisältää SceneReference-viittauksia taulukossa, ja niitä on helppo käsitellä editorissa. Taulukko muunnetaan pelitilaan siirryttäessä yksinkertaiseksi kokonaislukutaulukoksi.

Muokkaustilassa tasoja avatessa tarvitaan aina tiedoston polku (23). Polku haetaan työkalussa automaattisesti EditorBuildSettings-rajapinnasta. Tasoja voidaan ladata muokkaustilassa luvussa 3.6 esiteltävästä tasojenlatausvalikosta.

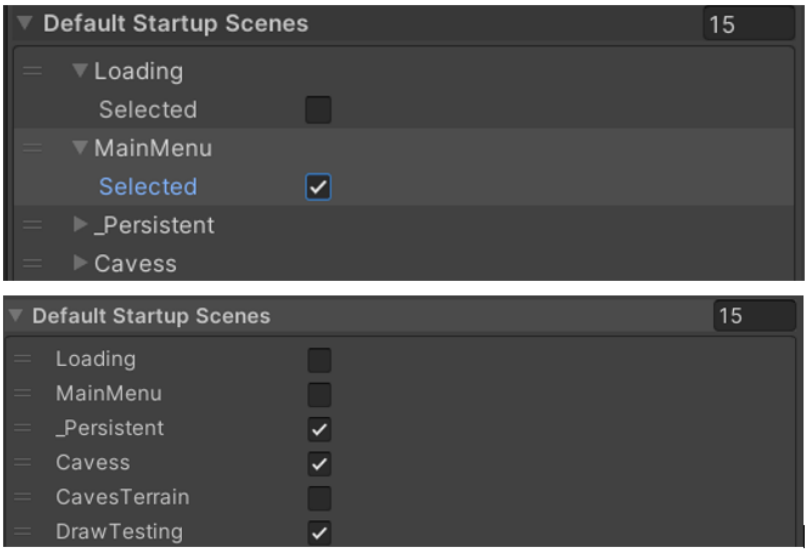
Editorin muokkaustilassa voidaan simuloida pelaamisen aloitusta tallennuspisteestä painamalla editorin tasojenlatausvalikon painiketta. Tällöin tasojen lataamiseen käytetään viittaustiedostoja. Työkalu hakee tasoviittausten polut automaattisesti taustalla.

Muita tasojenlatausvalikon painikkeita käytettäessä ei käytetä viittaustiedostoja. Valikkoa haluttiin voida käyttää ilman avoimen maailman sovellutusta tai viittaustiedostojen luontia. Tällöin tasojenlatausvalikon voi ottaa käyttöön ilman

ylimääräistä valmistelua. Työkalu ylläpitää viittauksia automaattisesti mahdollisuuksien mukaan. Muutostilanteissa ylimääräistä työtä viittaustiedostojen käyttöön havaittiin tulevan vähän. Tällöinkin muutokset jouduttiin korjaamaan vain latausvalikossa. Latausvalikkoa käytetään vain editorissa, jolloin tasojen latauksen vapaus on houkuttelevampi vaihtoehto kuin mahdollisesti työkalun käyttäjän huolimattomuuttaan lataama väärä taso. Viittaukset voi myös halutessaan ottaa käyttöön pienellä ohjelmamuutoksella.

Tasojen muutokset päivitetään valikon asetusikkunan avauksessa, ja lisäksi tapahtumien kuuntelija kuuntelee Unity-editorin EditorBuildSettings-luokkaa ja reagoi automaattisesti muutoksiin käännöskokoonpanossa halliten virhetilanteita. Asetusikkuna päivittää automaattisesti tasolistaukset käännöskokoonpanon mukaisesti.

Muokkaustilan tasojen abstrahointi on tehty työkalun SceneArrayElement-luokalla, joka mahdollistaa tason nimen näytön listauksessa, polun tallentamisen ja valinnan totuusarvon. Luokalle on tehty myös oma editorin laajennus nopeuttamaan käyttöä ja siistimään visuaalista näkymää käyttäen PropertyDrawer-luokkaa. Kuvassa 11 on yläpuolella alkuperäinen näkymä ja alapuolella työkalun oma muokattu näkymä.



Kuva 11. Muokkaustilan tasojen viittauksia listattuna. Yläpuolella alkuperäinen näkymä ja alapuolella työkalun muokattu näkymä.

Unityssä taulukoiden elementtien sisältämät muuttujat on oletuksena minimoitu, kuten kuvassa 11 näkyi. Oletusnäkymää käytettäessä tason valintaa ei voi havaita tai muokata klikkaamatta tason viittausta erikseen auki. Editorin laajennus tuo valinnat helpommin näkyville ja muokattavaksi.

3.3 Kytkimet tasojen suoratoistossa

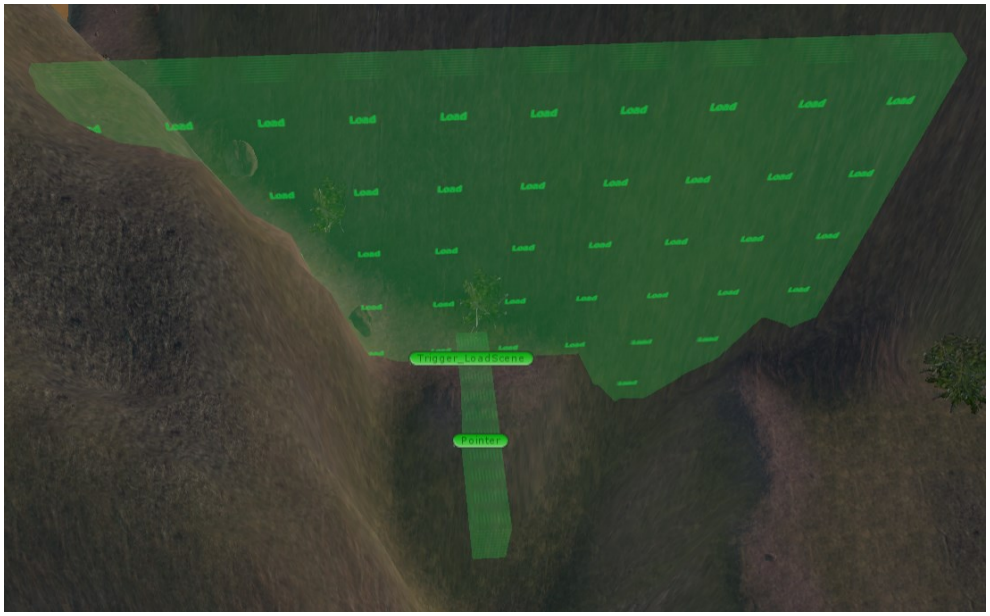
Tasojen suoratoistaminen perustuu työkalussa kytkimiin, jotka ovat kenttään asetettavia objekteja, joiden tehtävä on ladata ja poistaa tasoja pelin tapahtumien mukaisesti. Ennen kytkinten käyttöä luodaan Persistent-taso, joka on erityinen työkaluun merkitty jatkuvasti avoinna oleva taso. Persistent-tasoa ei voi ladata tai poistaa erikseen kytkinten kautta, vaan se ladataan automaattisesti aloituksessa.

Persistent-taso pitää sisällään kaikki suoratoistokytkimet ja muut projektikohtaiset maailmaan liittyvät asetukset, objektit ja logiikan. Tyypillisesti työkalulla tehdyn projektin globaaleihin käsitteisiin kuuluvat muun muassa yleisvalo, auringon tai kuun kohdevalo, jälkikäsittely, yleisäänet, pelilogiikka, pelihahmo ja kamera sekä yksi pakollinen Unityn luoma EventSystem-objekti (24).

Suoratoistokytkimillä on LevelStreamTrigger-komponentti, joka käynnistää suoratoisto-operaation. Komponenttia voi käyttää kahdella tapaa, joista tyypillisempi käyttö on Unityn törmäystarkistusten kanssa. Kun pelihahmo kulkee törmättävän objektin läpi, suoratoistaja aktivoituu. Toinen keino on kutsua komponenttia muualta koodista pelin tapahtumien mukaan, esimerkiksi portin auetessa tai pelaajan suoritettua tehtävän.

Törmäystarkistusta käytettäessä kytkimellä on BoxCollider-komponentti, jossa asetetaan IsTrigger valituksi. Jotta törmättäviä objekteja voidaan rajata, pitää pelihahmon objektin Tag-muuttujan olla nimeltään ”Player”. Halutessaan työkalun käyttäjä voi muokata törmäyksen käyttämään objektin Layer-muuttujaa. Tällöin projektin asetuksista valittaisiin, mitkä Layer-muuttujat vaikuttavat toisiinsa törmäystarkistuksessa (25).

Projektin mukana tulee valmiiksi kaksi esitehtyä prefab-muotoista kytkintä. Tasoja lataavan objektin MeshRenderer-komponentilla on vihreän sävyinen (kuva 12) ja poistavalla punaisen sävyinen tekstuuri. Prefab-objektissa on myös uloke, jolla ei ole minkäänlaista vaikutusta kytkimen toimintaan. Uloketta voidaan haluttaessa käyttää osoittamaan latausoperaation kohteena olevien tasojen suunnan.



Kuva 12. Prefab-muotoinen projektin mukana tuleva esimerkkikytkin. Vihreä väri tarkoittaa, että kytkin lataa tasoja.

Pelin tapahtumien perusteella käynnistettäviä suoratoisto-operaatioita varten LevelStreamTrigger-komponentti voidaan sijoittaa myös tyhjään objektiin Persistent-tasossa. Tällöin operaatio voidaan käynnistää kutsumalla komponentin TriggeredSceneStreaming-metodia muualta koodista kaikilla perinteisillä Unityn tukemilla tavoilla. Yleisen selkeyden ja kehityksen läpinäkyvyyden kannalta on hyvä pitää logiikka yhdenmukaisesti editorissa näkyvillä, jolloin keinona voisi esimerkiksi käyttää UnityEvent-luokkaa. UnityEvent annetaan muuttujana toisen objektin komponentille, kuten esimerkkikoodissa 1. Objektin ollessa valittuna UnityEvent näyttää Inspector-näkymässä listan kutsuttavista tapahtumista, joita kutsutaan, kun muuttujan Invoke-metodia kutsutaan. (26.)

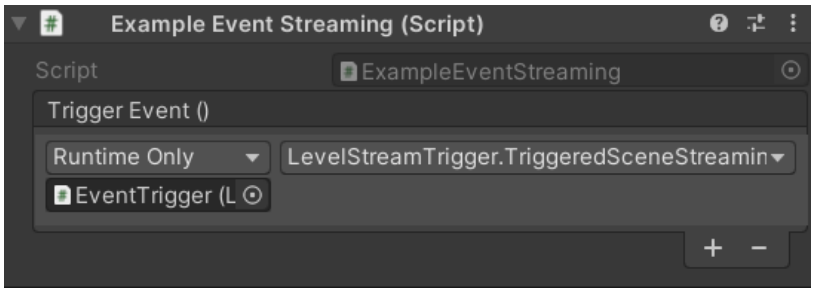
```
using UnityEngine;
using UnityEngine.Events;

public class ExampleEventStreaming : MonoBehaviour
{
    public UnityEvent triggerEvent;

    public void Execute()
    {
        triggerEvent.Invoke();
    }
}
```

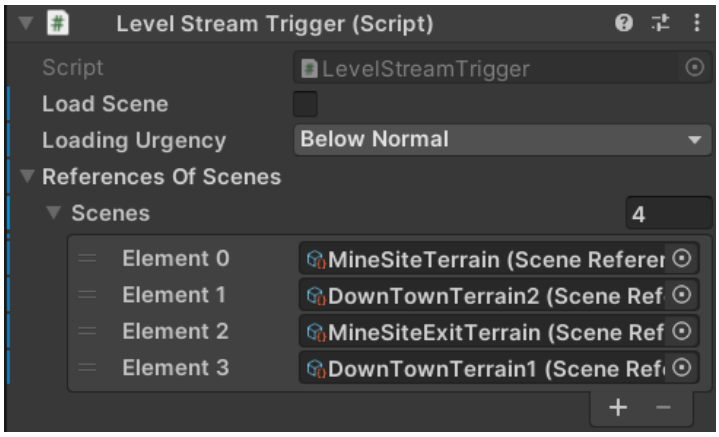
Esimerkkikoodi 1. UnityEvent-tapahtuman käynnistys.

Tapahtuma lisätään UnityEvent-näkymän tapahtumalistaan (kuva 13). Ensimmäiseksi painetaan pluspainiketta. Seuraavaksi objektivalintaan vedetään ja pudotetaan tasosta kytkimen sisältävä objekti. Metodin valinnaksi pudotusvalikosta valitaan ensin LevelStreamTrigger ja sitten TriggeredSceneStreaming. Työkalu sisältää esimerkkitoteutuksen UnityEvent-tapahtuman käytöstä.



Kuva 13. UnityEvent-muuttujan kautta kutsuttu suoratoisto-operaatio.

LevelStreamTrigger-komponentissa (kuva 14) on LoadScene-totuusarvo. Arvon ollessa tosi kytkin lataa tasoja ja arvon ollessa väärä kytkin poistaa tasoja. Operaation kohteena olevat tasot määräytyvät Scenes-nimisen taulukon mukaan. Suoratoistaja hyväksyy vain Persistent-tasoa suuremmat tasojen indeksit.



Kuva 14. LevelStreamTrigger-komponentti Inspector-näkymässä.

Pelitilassa Scenes-taulukko muunnetaan kokonaislukutaulukoksi. Kytkimen aktivoituessa valitaan LevelStreamer-luokan lataus- tai poistometodi. Metodin parametrina on kokonaislukutaulukko. Metodi välitetään uuteen Coroutine-olioon. Metodit luovat uuden AsyncOperation-olion ja käsittelevät taulukon tasot yksi kerrallaan odottaen yhden valmistumista, ennen kuin seuraava taso aloitetaan. Ennen lataus- tai poistotoimenpidettä ohjelma varmistaa, että taso on olemassa eikä se jo ole halutussa tilassa. Taso pitää myös voida saattaa haluttuun tilaan indeksin perusteella. AsyncOperation-oliossa tasot ladataan asynkronisesti SceneManager-luokan LoadSceneAsync-metodin kautta additiivisesti ja poistetaan UnloadSceneAsync-metodilla.

Tasojen lataaminen ja poistaminen tehdään suoratoistamisen periaatteella, eli pelaaminen ei keskeydy pelitasoja ladattaessa. Tasojen määrästä ja koosta riippuen operaatio tapahtuu täysin huomaamattomasti tai lyhyen latauspiikin kera. Latauspiikillä tarkoitetaan muun pelin pysähtymistä pelaamisen aikana pelimaailman tasojen lataamisen ajaksi.

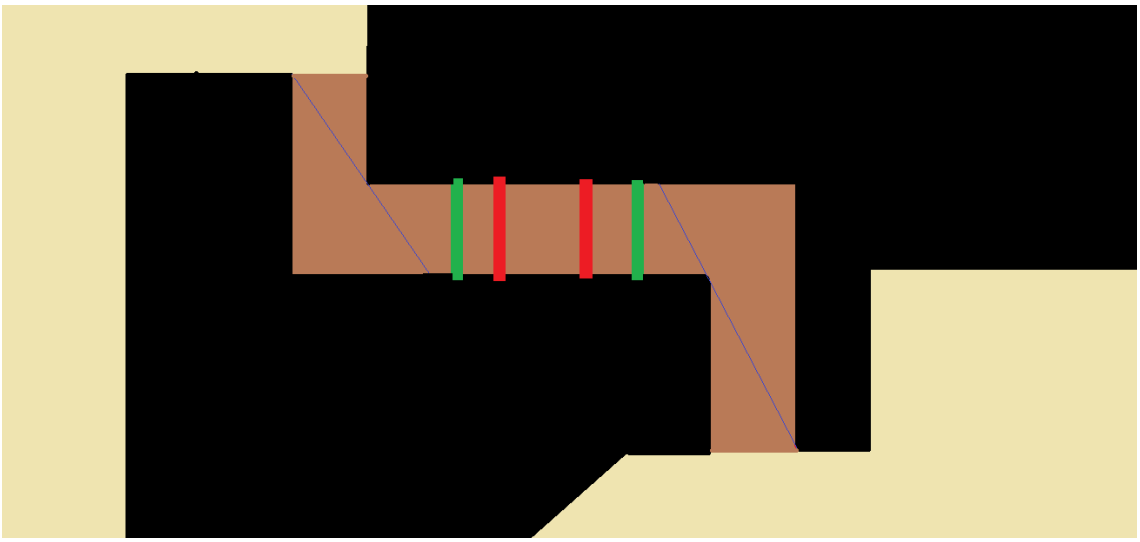
Latauspiikkiin pystytään vaikuttamaan asettamalla LevelStreamTrigger-komponentin LoadingUrgency-muuttuja. Muuttujan valinta vaikuttaa Application.backgroundLoadingPriority-muuttujaan, joka määrittää, kuinka monta millisekuntia asynkroniset operaatiot voivat korkeintaan käyttää yhdestä keskusyksikön pääsäikeen kehyksestä (engl. frame). Vaihtoehtoina taustalla

lataamisen maksimikestoihin ovat Low (2 ms), BelowNormal (4 ms), Normal (10 ms) ja High (50 ms). (27.) Mitä vähemmän millisekunteja valitsee, sen vähemmän latauspiikkiä huomaa. Tällöin kuitenkin tarvitaan enemmän latausaikaa ja sitä myötä matkaa pelaajalle kuljettavaksi tasojen välille.

Kytkinten kehitysvaiheessa havaittiin, että pelissä olevat lataavat ja poistavat kytkimet tulee aina sijoittaa erilleen toisistaan, mielellään reilun etäisyyden päähän. Tällöin pelaajan satunnainen edestakainen askellus ei aiheuta turhia latauksia ja poistoja. Lataava kytkin sijoitetaan jokaisesta kohdastaan lähemmäksi ladattavaa tasoa ja poistava kytkin poistettavaa tasoa. Tällöin halutun kytkimen vaikutus jää aina viimeiseksi eivätkä maailmassa jää väärät tasot näkyviin tai näkymättömiin. Samasta syystä saman tason lataavat ja poistavat kytkimet eivät voi leikata toistensa läpi.

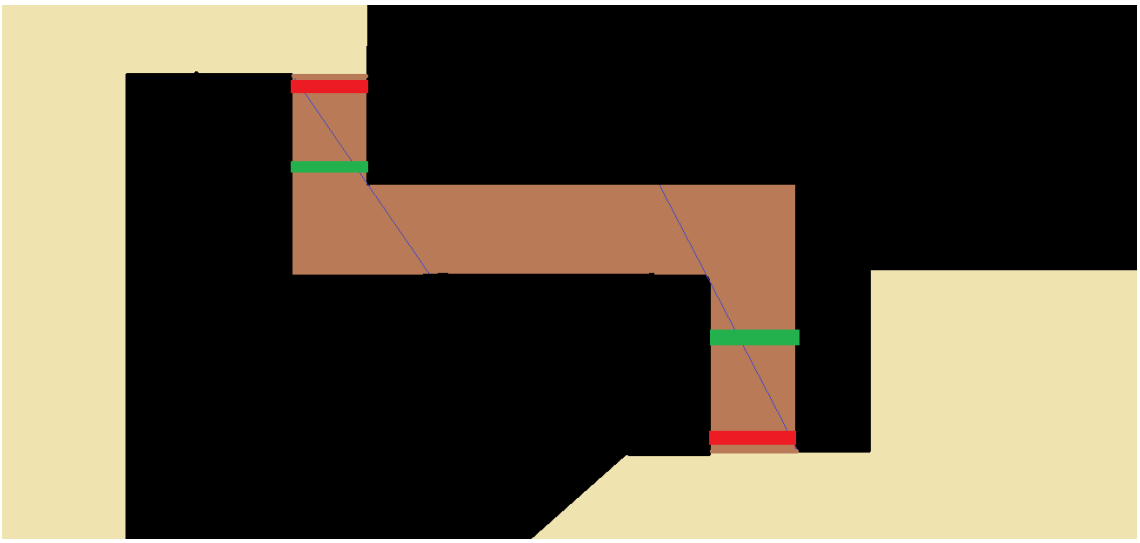
Tasot olisi syytä järjestää siten, että kytkin pystyy kattamaan koko reitin eikä kytkimen ohi pääse koskematta sitä. Lisäksi immersion kannalta voi pitää nyrkkisääntönä, ettei kytkimen tärkeimmistä kohdista ole näköyhteyttä ladattaviin tasoihin, jotta pelaaja ei näe tason latautuvan tai puuttuvan. Lataamiselle on hyvä jättää myös riittävästi etäisyyttä. Tällöin ladattava taso ehtii latautua, ennen kuin pelaaja ehtii kulkea kohtaan, josta tason näkee.

Kuvassa 15 on esimerkki kytkinten asettelusta tiukassa käytävässä. Ratkaisussa vain toinen käytävän ulkopuolinen alue on kerrallaan muistissa. Matkaa lataamiselle ei kuitenkaan jää paljoa.



Kuva 15. Kytkinten asettelu yksinkertaistetusti. Vasemmanpuoleiset kytkimet vaikuttavat vasemmalla puolella olevaan tasoon ja päinvastoin. Kuvan ohuet siniset viivat kuvastavat käytävien katkaisemaa näkyvyyttä. Vihreä kytkin lataa ja punainen kytkin poistaa tason.

Kuvassa 16 on vaihtoehtoinen ratkaisu, jossa nopeasti käytävään astuessa aloitetaan käytävän toisessa päässä olevan tason lataus. Tässä ratkaisussa käytävän molemmin puolin olevat tasot ovat hetken ladattuna muistissa. Siirryttävän tason lataamiselle jätetään kuitenkin aikaa.



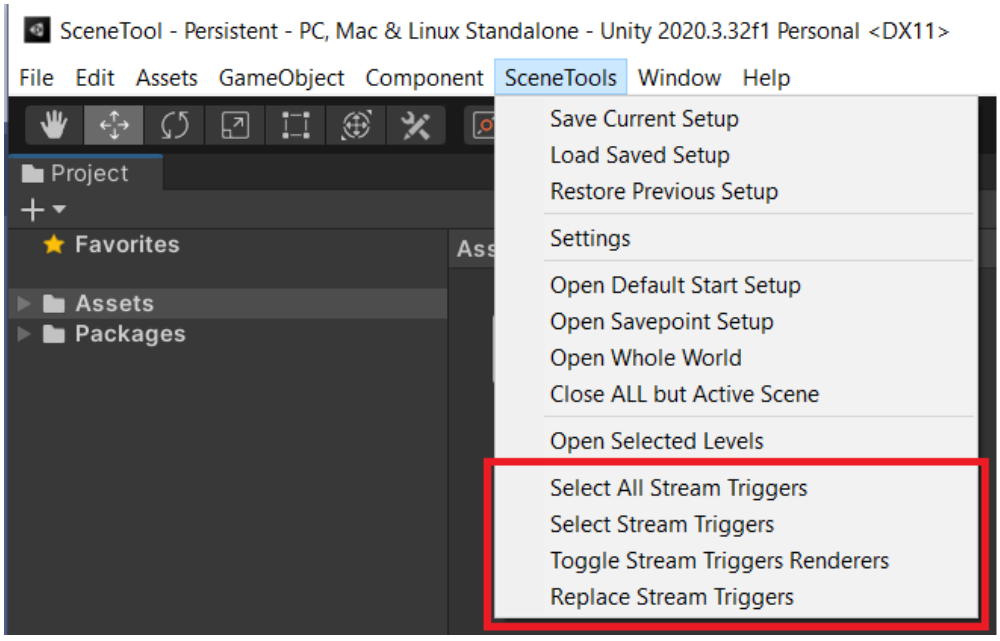
Kuva 16. Vaihtoehtoinen kytkinten asettelu. Vasemmanpuoleiset kytkimet vaikuttavat oikealla puolella olevaan tasoon ja päinvastoin.

Molemmat vaihtoehdot ovat toimivia, ja pelinkehittäjä voi valita haluamansa tavan tilanteen mukaan. Suurelle alueelle siirryttäessä on kuitenkin hyvä jättää riittävästi tilaa tasojen välille.

Kytkinten lisäys peliin käy nopeasti pelinkehityksessä, jolloin pelinkehittäjä pääsee halutessaan liikkumaan pelimaailmassa heti alusta asti. Suurempaa työtä pelinkehityksessä vaatii pelattavan avoimen maailman huolellinen suunnittelu, jotta tasot latautuvat tehokkaasti ja näin pelimaailma näyttäytyy pelaajalle yhtenäisenä.

3.4 Kytkinten hallintatyökalut

Kytkinten hallinnan helpottamiseksi on editorin yläpalkkiin lisätyssä SceneTools-valikossa neljä erilaista kytkinten hallintaa helpottavaa painiketta. Painikkeet on kuvassa 17 rajattu punaisella värillä. Näillä painikkeilla voi helpommin valita kytkimiä, muuttaa niiden näkyvyyttä, etsiä tasoviittauksia kytkimistä ja korvata tasoviittauksia kytkimissä.

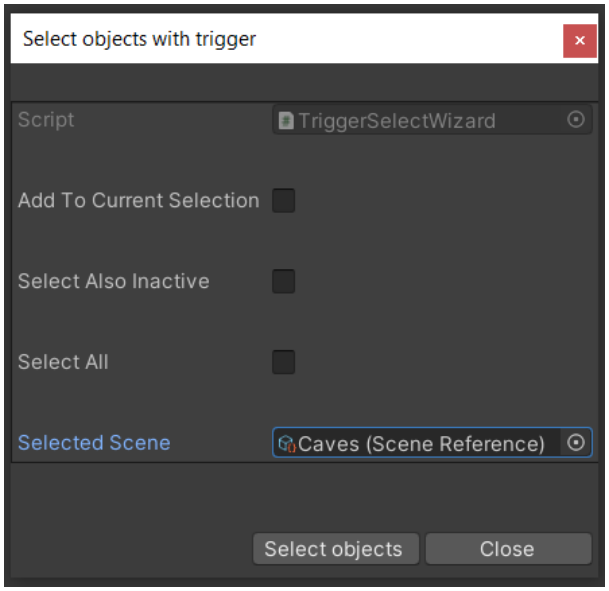


Kuva 17. Valikkotyökalu kytkinten hallintaan. Punaisella rajattu kytkintenhallinnan osio.

Valikkotyökaluissa on oletettu, että kytkimet löytyvät aina Persistent-tasosta, mikä tekee sen käytöstä potentiaalisesti paljon nopeampaa, koska kaikkia tasoja ei tarvitse käydä objektikohtaisesti läpi. Tämän työn luvussa 5 esitetyssä testiympäristössä vaikutus oli silmänräpäys verrattuna muutamiin sekunteihin kyseisen optimoinnin kohdalla. Mitä isompi peli, sen isompi ero on latausajassa.

Select All Stream Triggers -painikkeen avulla voi valita kaikki aktiiviset objektit tasohierarkiassa, joilla on LevelStreamTrigger-komponentti. Epäaktiiviset kytkimet hylätään valinnasta.

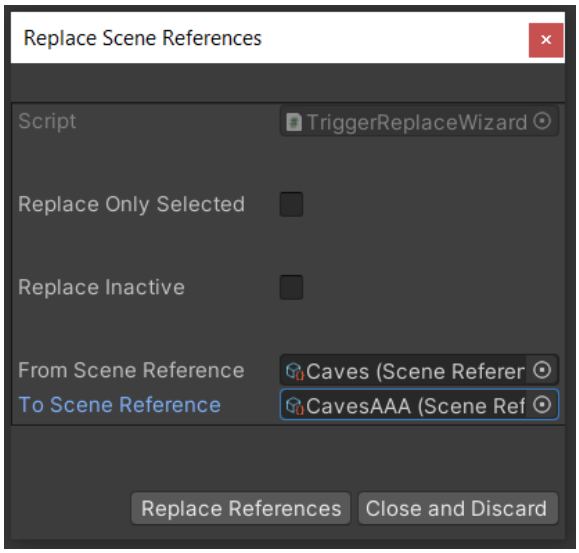
Select Stream Triggers -painike avaa kuvan 18 mukaisen ponnahdusikkunan. Ponnahdusikkunassa voidaan valita ja etsiä Persistent-tason objekteja, joilla on LevelStreamTrigger-komponentin taulukossa SelectedScene-muuttujan mukainen SceneReference-tasoviittaus. Objektien valitsemistavan voi valita. Oletusvaihtoehto on hylätä senhetkinen muokkaustilan objektien valinta ja valita vain kyseiset objektit. Toinen vaihtoehto on lisätä objektit nykyiseen valintaan. Lisäksi voidaan vaikuttaa siihen, valitaanko myös aktiivisten objektien lisäksi epäaktiiviset objektit. SelectAll-totuusarvo ohittaa tasoviittauksen, ja sitä voikin yhdessä SelectAllInactive-totuusarvon kanssa käyttää valitsemaan kaikki aktiiviset ja epäaktiiviset kytkimet.



Kuva 18. Valikkotyökalu kytkinobjektien valintaan.

Toggle Stream Triggers Renderers -painike muuttaa kaikkien aktiivisten kytkinten visuaaliseen näkyvyyteen vaikuttavien MeshRenderer-komponenttien tilaa vuoron perään päälle ja pois. Painike vaikuttaa editorin muokkaus- ja pelitilassa. Käännetyssä pelissä kytkimet ovat aina automaattisesti visuaalisesti piilotettuja.

Replace Stream Triggers -painike avaa kuvan 19 mukaisen ponnahdusikkunan. Ikkunassa on kaksi SceneReference-muuttujaa, joista ensimmäiseen valitaan haluttu korvattava tasoviittaus ja jälkimmäiseen korvaava tasoviittaus. Lisäksi voidaan valita korvaamaan vain editorissa valittuna olevista objekteista tai vaihtoehtoisesti kaikista kytkimistä viittaukset. Ikkunassa on myös lisävalinta, joka mahdollistaa myös epäaktiivisissa objekteissa olevien viittausten korvaamisen.



Kuva 19. Valikkotyökalu kytkinten tasoviittausten muuttamiseen.

Kytkimiä muokataan ja tarkastellaan avoimen maailman työstämisen yhteydessä usein, joten valikkotyökalut auttavat ja nopeuttavat työskentelyä, luovat selkeyttä ja ennaltaehkäisevät virheitä.

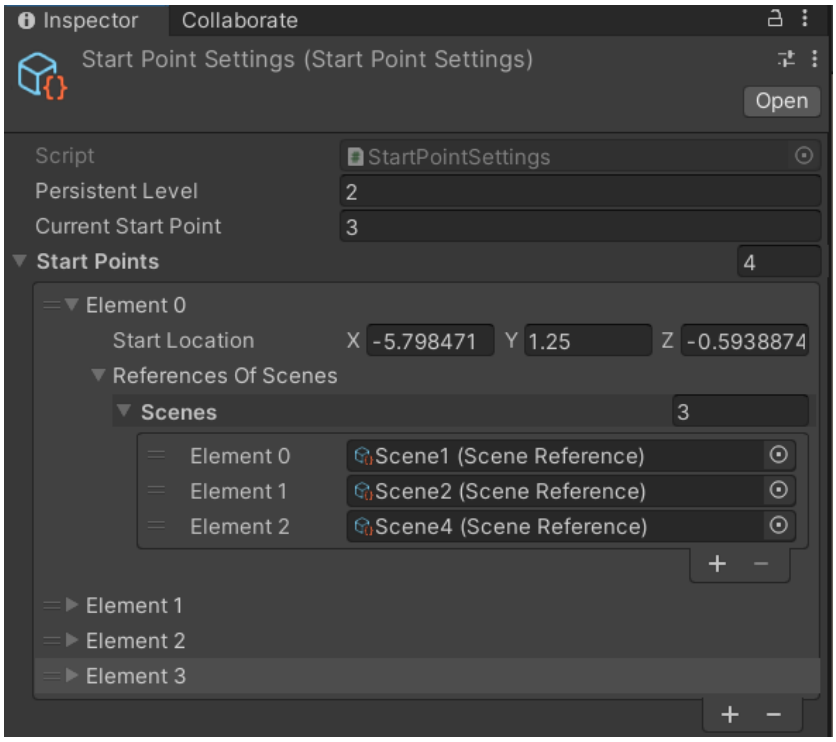
3.5 Aloituspisteiden käyttäminen

Työkalussa tallennuspisteitä lähestytään periaatteella, että tallennus- ja aloituspisteet ovat esimääriteltyjä, jolloin ennakkoon tiedetään, mitkä tasot ovat ladattuina aloitettaessa miltäkin pelaajan aloitussijainnilta. Vaihtoehtoinen lähestymistapa peleissä on antaa pelaajalle vapaus valita tallennuspisteen sijainti itse, jolloin sillä hetkellä auki olevat tasot ja sijainti tallennettaisiin.

Esitallennetut tallennuspisteet ovat hyödyllisiä jo kehitysvaiheessa, vaikka lopullisessa pelissä haluttaisiin käyttää vapaata tallennusta. Testaaminen ja tasojen muokkaaminen nopeutuu, kun pelinkehittäjä voi automaattisesti siirtää pelihahmon haluttuun kohtaan ja ladata muokkauksen kohteena olevan alueen. Työkalu tukee kuitenkin suoraan kumpaakin tapaa tarjoamalla toteutusesimerkin ja rajapinnan, johon tarvittavat tiedot voidaan tallentaa ja hakea sieltä. Lopullinen implementaatio rajapinnan käyttöön on aina projektikohtainen, koska hahmon ja pelilogiikan hallinta riippuu muista projektin komponenteista.

Aloituspisteet tallennetaan taulukkomuodossa Assets/Recourses-kansioon automaattisesti luotavaan StartPointSettings-nimiseen ScriptableObject-tiedostoon, jolloin sitä voidaan muokata myös manuaalisesti tiedoston kautta (kuva 20). Tiedosto sisältää taulukon aloituspisteistä. Jokainen aloituspiste sisältää listan tasoviittauksista ja aloituspisteen sijainnin pelimaailman koordinaatistossa.

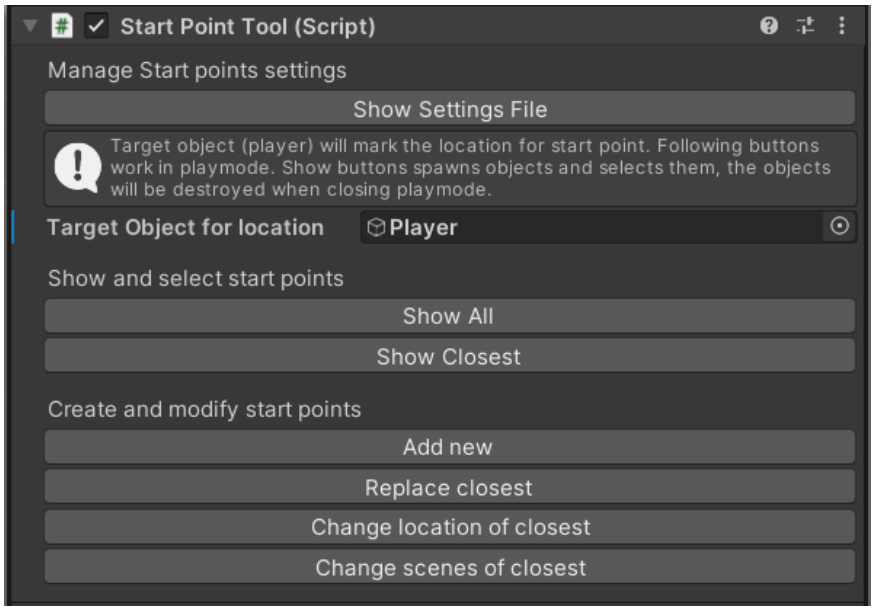
Muokkaustilassa valittu aloituspisteen indeksi tallennetaan StartPointSettings-tiedostoon seuraavassa luvussa esiteltävän tasojen kokoonpanovalikon asetusikkunan kautta. Pelitilassa aloituspisteen indeksi on kuitenkin eritelty ja suojattu muokkaustilan valinnasta ja se tallennetaan erillisesti metodia kutsumalla automaattisesti luotuun binääriformatoituun tiedostoon. Näin pelinkehittäjä ei vahingossa tule sotkeneeksi lopullisen pelin tietoja ja voi sujuvasti siirtyä sekä muokkaus- että pelitilassa tallennettuun pelitilanteeseen. Muokkaustilassa auki olevat tasot pidetään oletuksena auki editorin pelitilaan siirryttäessä, ellei pelimaailmaa käynnistetä erikseen metodia kutsumalla.



Kuva 20. Tiedostoon tallennetut aloituspisteet.

Työkalun mukana tulee myös StartPointTool-niminen Prefab-objekti helpottamaan aloituspisteen hallintaa. Objekti sijoitetaan Persistent-tasoon ja sitä käytetään editorin pelitilassa. Muokkaustilassa objektin käyttäminen ei ole mahdollista. Käännetyssä pelissä objekti poistetaan automaattisesti.

Aloituspisteiden hallintaobjekti sisältää StartPointTool-komponentin. Komponentti muokkaa julkisen ja staattisen StartPointHandler-rajapinnan kautta aloituspisteitä. Komponentille on muokattu oma laajennettu kuvan 21 mukainen ikkunanäkymä. Asetukset sisältävän tiedoston voi paikallistaa painamalla ikkunan *Show Settings File* -painiketta. Aloituspisteiden luomiseksi tai muokkaamiseksi ikkunasta valitaan ensiksi pelaajan objekti, jota käytetään aloituspisteen sijainnin määrittämisessä.



Kuva 21. Valikkotyökalu tallennuspisteiden visualisoimiseksi, lisäämiseksi ja muokkaamiseksi.

Aloituspisteitä lisätäkseen tai muokatakseen pelinkehittäjä voi editorin pelitilassa aloittaa pelaamisen maailman ensimmäisestä aloituspisteestä ja liikkua pelissä seuraaville pelialueille. Päästyään haluttuun kohtaan pelinkehittäjä voi aloituspisteiden hallintaikkunasta painaa *Add New* -painiketta lisätäkseen aloituspisteen. Painikkeen painaminen lisää automaattisesti aloituspisteiden listaan uuden elementin.

Tasojen hallintavalikon *Open Default Start Setup* -painiketta voidaan muun muassa käyttää helpottamaan maailman ensitilan lataamista kehityksen alkuvaiheessa. Painike esitellään tarkemmin seuraavassa luvussa.

Muokatakseen aloituspistettä voidaan hallintaikkunasta painaa *Replace closest* -painiketta, joka muuttaa lähimmän aloituspisteen ladattavat tasot ja aloitussijainnin. *Change location of closest* -painike muokkaa lähimmän aloituspisteen aloitussijaintia. *Change scenes of closest* -painike muokkaa lähimmän aloituspisteen ladattavia tasoja.

Aloituspisteiden sijaintien visualisoimiksesi voidaan painaa *Show All* -painiketta, joka näyttää kaikki aloituspisteet, tai *Show Closest* -painiketta, joka näyttää

lähimmän aloituspisteen. Painikkeiden painamisen seurauksena joko jokaisen tai lähimmän aloituspisteen aloitussijaintiin luodaan uusi visuaalinen pallonmuotoinen peliobjekti. Pallon nimessä näytetään myös aloituspisteeseen liitettyjen tasojen nimet. Luodut pallot valitaan automaattisesti muokkausikkunassa, jotta ne ovat helpompia havaita. Lisätyt objektit poistetaan automaattisesti, kun pelitilasta poistutaan.

Pelitilaan siirryttäessä pelin logiikka on aina projektikohtaista, jolloin jokaisessa tilanteessa toimivaa valmista mallia tallennuspisteiden käyttämisestä ei voida antaa. Työkalun mukana tulee kuitenkin uudessa projektissa toimivat esimerkkikoodit, jolla työkalua voi käyttää aloituspisteen lataamisessa editorin pelitilassa ja käännetyssä pelissä.

Työkalun StartPointHandler-luokka toimii rajapintana aloituspisteiden käytölle. Aloituspisteen tallentamiseksi pelitilanteessa pelinkehittäjä voi kutsua luokan SaveCurrentStartPoint-metodia esimerkkikoodin 2 mukaisesti. Metodille annetaan parametrinä tallennetun aloituspisteen indeksi.

```
private void SetStartPoint(int index)
{
     StartPointHandler.SaveCurrentStartPoint(index);
}
```

Esimerkkikoodi 2. Aloituspisteen muutoksen tallentaminen pelitilanteessa.

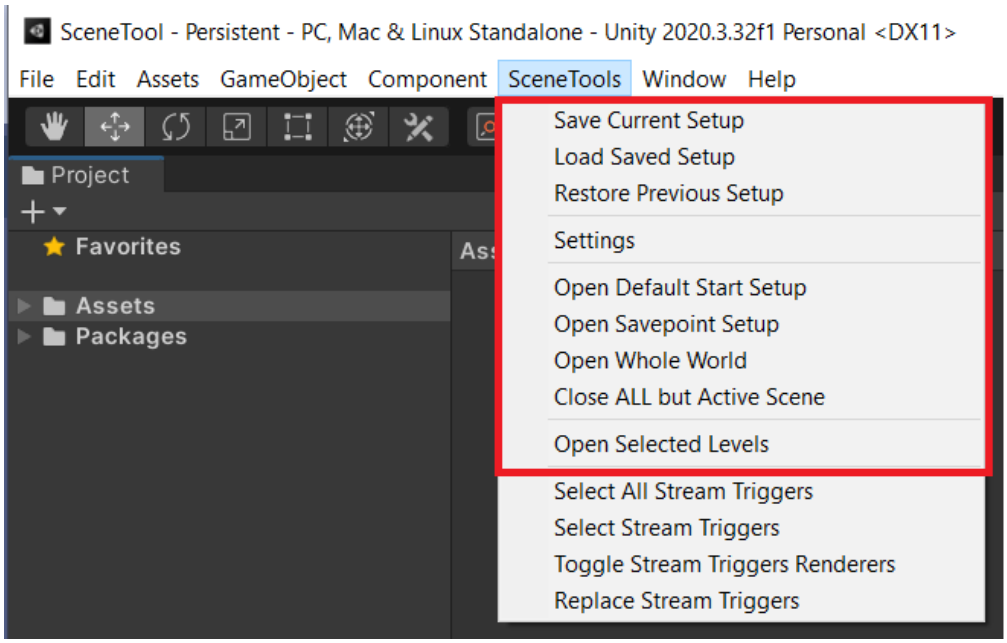
Peliä pelatakseen maailman voi käynnistää käyttämällä LevelStreamer-luokan LoadStartScenes- tai LoadRestartScenes-metodeja. Kumpikin metodi lataa aiemmin SaveCurrentStartPoint-metodilla valitun aloituspisteen tasot automaattisesti. Pelihahmo voidaan manuaalisesti siirtää aloituspisteeseen. Aloitussijainti haetaan StartPointHandler-luokan CurrentStartPointLocation-metodista. Maailman käynnistäminen on tarkemmin dokumentoitu luvussa 4.1, josta löytyvät myös tarkemmat ohjeet rajapintojen käytöstä sekä esimerkkikoodit.

Editorin yläpalkin SceneTools-valikossa on *Open Save Point* -painike, josta voi muokkaustilassa simuloida pelin aloittamista tallennuspisteestä. Asetuksista voidaan valita aloituspisteen indeksi ja päättää, siirretäänkö muokkaustilan

kamera aloituspisteeseen tallennettuun aloitussijaintiin. Painiketta painaessa avataan valitun aloituspisteen tasot muokkaustilaan ja kamera siirretään aloituspisteen aloitussijaintiin, jos niin valittiin. Valikkotyökalu esitellään seuraavassa luvussa tarkemmin.

3.6 Tasojen kokoonpanotyökalut editorissa

Tasojen hallintaa varten on työkalussa editoriin lisätty uusi SceneTools-niminen valikko painikkeineen (kuva 22) editorin yläpalkkiin. Valikon kautta voidaan avata erilaisia tasokokoonpanoja eli tasojen joukkoja. Tasot aukeavat editorin muokkaustilassa Scene- ja Hierarchy-ikkunaan. Osa painikkeista toimii dynaamisesti ja osa esikonfiguroidusti. Lisäksi valikossa on painike, josta voidaan vapaamuotoisesti avata tasoja. Kaikki tasojen hallintaan tarkoitettujen painikkeiden toiminnot, paitsi tallennuspisteet, ovat käytettävissä itsenäisesti ilman avoimen maailman sovellutusta ja erillisiä tasojen viittaustiedostoja. Tallennuspisteet käyttävät viittaustiedostoja, koska ne yhdistyvät avoimen maailman mekanismeihin, eikä tallennuspisteitä haluta luoda toiseen kertaan editorin käyttöön.



Kuva 22. Valikkotyökalu tasojen avaamiseksi. Punaisella rajattu tasojenhallintaan käytetty osio.

Kaikki esikonfiguroidut painikkeet asettavat automaattisesti asetusikkunassa määritetyn Persistent-tason aktiiviseksi. Aktiivinen taso vaikuttaa siihen, miten objekteja lisätään objektien lisäyksessä ohjelmallisesti ja mitä asetuksia käytetään esirakennusvaiheessa. Unityssä pitää aina olla yksi aktiivinen taso auki. (28.)

Joka kerta, kun tasokokoonpanoja yritetään avata valikosta, työkalu kutsuu tallennusmetodia. Metodissa tarjotaan jokaisen avoimena olevan muutoksia sisältävän tasotiedoston osalta Unityn perustallennusta. Tallennuksesta tulee ponnahdusikkuna, josta voidaan tallentaa tai hylätä muutokset. Ponnahdusikkunasta voidaan painaa myös Cancel-painiketta, joka peruuttaa tallentamisen ja kokoonpanojen avaamisen. Tallentamisessa huomioidaan myös tasojen avaus Prefab-muokkaustilan ollessa avoinna.

Dynaamiset painikkeet

Save Current Setup -painike tallentaa tiedon painamishetkellä editorissa auki olevista tasoista. Painike ei ole riippuvainen käännöskokoonpanosta tai tasojen abstrahoinnista.

Load Saved Setup -painike lataa *Save Current Setup* -painikkeella tallennetun tasokokoonpanon editorissa. Painike ei ole riippuvainen käännöskokoonpanosta tai tasojen abstrahoinnista.

Restore Previous Setup -painike palauttaa edellisen editorissa olleen tasokokoonpanon. Painike on käytännöllinen, kun halutaan hyppiä edestakaisin kahden tasokokoonpanon välillä. Painike ei ole riippuvainen käännöskokoonpanosta tai tasojen abstrahoinnista.

Close ALL but active scene -painike sulkee kaikki muut tasot paitsi sen, joka on editorissa määritetty aktiiviseksi tasoksi. Aktiivinen taso voidaan valita klikkaamalla tasoa Hierarchy-ikkunassa hiiren oikealla painikkeella ja klikkaamalla aukeavasta valikosta Set Active Scene -painiketta. Aktiiviseksi muutetun tason nimi saa indikaatioksi lihavoidun fontin Hierarchy-ikkunassa.

Esikonfiguroidut painikkeet

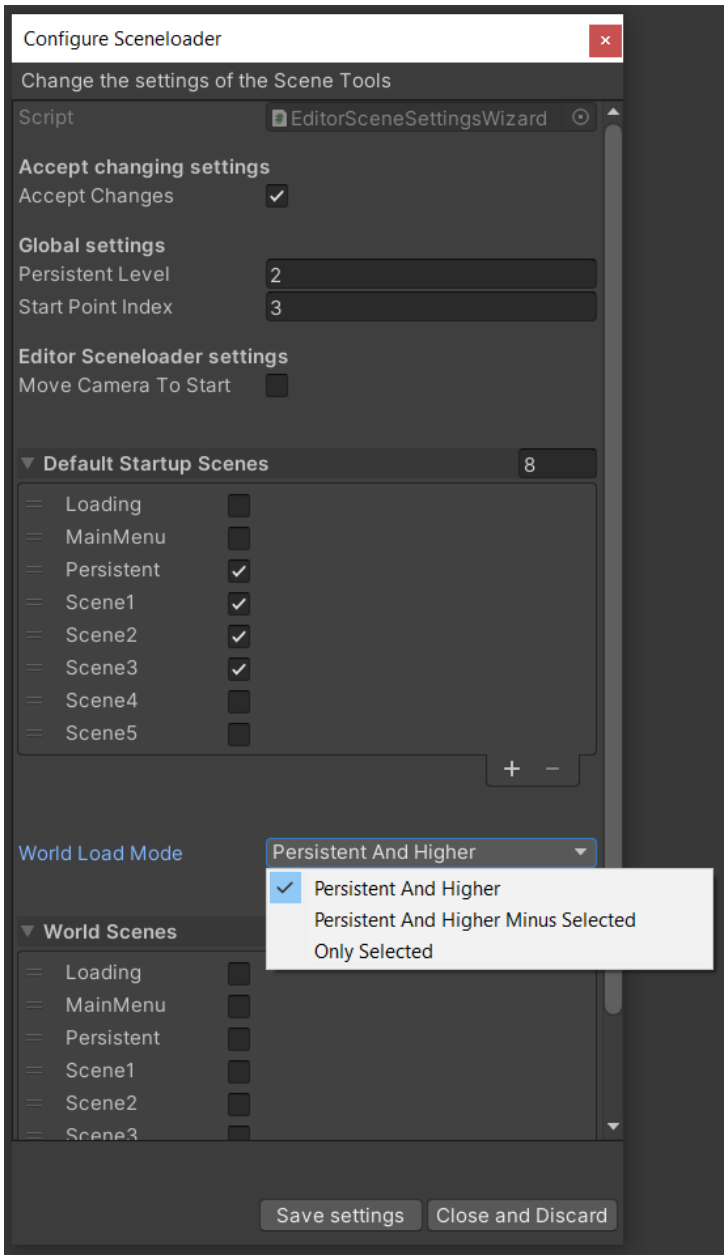
Settings-painikkeen painaminen avaa kuvan 23 mukaisen ponnahdusikkunan, josta voidaan määrittää Persistent-taso ja oletustasokokoonpanot pelin aloitukselle ja koko maailmalle. Koko maailman lataamiselle on valittavissa tapa kolmesta valinnasta:

1. Persistent-taso ja kaikki sen jälkeen indeksissä tulevat tasot

2. sama kuin ensimmäinen, mutta valitut tasot hylätään

3. vain valitut tasot.

Asetusvalikosta voidaan myös valita ennalta tallennetuiden pisteiden joukosta aloituspisteen indeksi, joka määrittää senhetkisen aloituspisteen valikkotyökalussa, ja se, siirretäänkö muokkaustilan kamera aloituspisteen sijaintiin. Ennen asetusten tallennusta pitää muutokset hyväksyä muuttamalla AcceptChanges-valinta todeksi. Yleinen tasojenavausikkuna on samankaltainen, ja vahvistaminen tehdään, jotta ei vahingossa tulla muuttaneeksi oletusarvoja. Tallennus tehdään *Save Changes* -painikkeesta ja muutokset hylätään *Close and Discard* -painikkeesta. Asetukset tallennetaan työkalun luomaan EditorSceneSettings-nimiseen tiedostoon projektin *Assets/Settings*-kansioon, joten valinnat muistetaan editorin uudelleenkäynnistyksen jälkeen. Asetuksia voidaan muokata myös suoraan tiedoston kautta.



Kuva 23. Asetusvalikko, josta muutetaan tasokokoonpanojen ja maailman oletusarvoja.

Open Default Start Setup -painike avaa asetustenmukaisen tasokokoonpanon, jonka tarkoituksena on toimia pelin aloitustilanteena. Tasojen valinnat voi tehdä vapaasti. Jos Persistent-taso ei ole valittuna, annetaan asetusvalikossa varoitus, mutta tallentamista ei estetä.

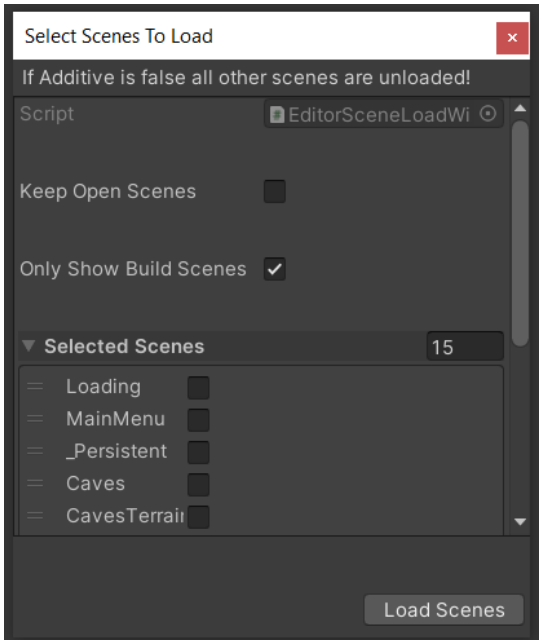
Open Savepoint Setup -painike simuloi asetuksissa valitun tallennuspisteen tilannetta ja avaa tallennetut tasot muokkaustilaan. Tallennuspiste sisältää tiedon

pelaajan sijainnista. Muokkaustilan kamera siirretään aloituspisteeseen, jos niin on valittu asetuksissa. Painike on käytössä vain, jos avoimen maailman sovellutusta käytetään.

Open Whole World -painike avaa asetustenmukaisen tasokokoonpanon, jonka tarkoituksena on edustaa koko maailmaa, johon pelaaja voi kulkea. Tasojen valinnat voi tehdä myös vapaasti. Tämä painike kannattaa laittaa vastaamaan koko maailmaa. Tällöin sitä voidaan hyödyntää tasojen saumattomuuden ja koko maailman tarkastelussa sekä esirakennusvaiheen nopeuttamisessa.

Vapaamuotoinen painike

Open Selected Levels -painike avaa kuvan 24 mukaisen ponnahdusikkunan, jonka kautta editoriin voi avata listasta valittuja tasoja. Ikkunasta on mahdollista valita, pidetäänkö jo auki olevat tasot auki vai suljetaanko ne, kun valitut tasot avataan. Aukipitämisen valinta säilytetään aina muistissa. Ikkunasta on myös mahdollista valita, onko tasolistauksessa listattu kaikki tasot käännöskokoonpanosta (BuildIndex), tai vaihtoehtoisesti, käydäänkö kaikki projektin kansiot läpi ja haetaanko kaikki löydetyt tasotiedostot.



Kuva 24. Ponnahdusikkuna, josta voi avata vapaamuotoisesti tasoja.

Tasojen yleinen avausikkuna voidaan haluttaessa pitää jatkuvasti auki, jolloin tasojen etsimiseen monimutkaisesta kansioiden joukosta ei tarvitse käyttää aikaa. Työkalu myös helpottaa esimerkiksi projekteja, joissa on paljon ulkopuolisten pakettien tasoja. Pakettien mukana tulevat esimerkkitasot löytyvät helposti.

4 Pelin käännös ja suorittaminen

Tässä luvussa kerrotaan pelin kääntämisestä ja suorittamisesta. Luku koostuu kolmesta alaluvusta, joita ovat maailman suoratoistaminen, kääntämisen ja pelitilan esivalmistelut sekä pelin kääntäminen. Luvussa käydään läpi tapoja käynnistää pelimaailma esimerkkikoodien avulla ja se, miten peliä voidaan pelata itsenäisenä sovelluksena.

Työkalulla tehtävän pelin ensimmäisessä kehitysvaiheessa lisätään peliin Persistent-taso, maailman pelattavia alueita edustavia tasoja sekä kytkimiä maailmassa liikkumista varten luvun 3 ohjeiden mukaisesti. Editorin pelitilassa voidaan pelimaailmaa pelata pitämällä halutut tasot auki muokkaustilassa. Jotta peliä voi luontevammin testata editorissa tai viimeistään siirryttäessä käännettyyn peliin, pitää muiden osien olla viritettynä toimimaan suoratoistamisen kautta.

Peli tyypillisesti käynnistetään painamalla valikkopainiketta, joka lataa aloituksessa tarvittavat oikeat tasot ja objektit maailmaan. Joissain tilanteissa pelimaailma voidaan haluta ladata uudelleen maailman eri kohdasta. Esimerkiksi pelaajan hahmon kuoltua voidaan pelihahmo haluta palauttaa edelliselle alueelle.

Pelin toiminta itsenäisenä sovelluksena vaatii pelin kääntämistä. Ennen pelin kääntöä voidaan projektikohtaisesti joutua myös esirakentamaan erilaisia osa-alueita, kuten valaistusta tai navigaatioverkkoja.

4.1 Maailman suoratoistaminen

Pelimaailma käynnistetään pelaajan aloitteesta, tyypillisesti valikkopainiketta painamalla. Pelaaja voi aloittaa pelaamisen pelimaailman ensimmäisestä sijainnista tai pelaajan edistymisen seurauksena tallennetusta aloituspisteestä. Työkalun mukana tulee esimerkkitoteutukset, joita voidaan käyttää molemmissa tilanteissa. Lopullinen toteutus on kuitenkin projektikohtainen.

Maailman käynnistyksessä kutsutaan LevelStreamer-luokan metodeja. Luokan metodit osaavat itse hakea StartPointHandler-luokan välityksellä tallennetun aloituspisteen tasot.

Ensimmäistä kertaa maailmaa käynnistettäessä kutsutaan LoadStartScenes-metodia. Tällöin metodi poistaa automaattisesti aiemmat tasot ja aukaisee ensimmäiseksi Persistent-tason ja sitten muut tallennuspisteen tasot.

Pelihahmon kuollessa kutsutaan LoadRestartScenes-metodia maailman uudelleenkäynnistämiseksi. Metodia voidaan käyttää myös pikaiseen siirtymiseen alueelta toiselle (engl. fast travel). Latausvaiheessa auki oleva Persistent-taso säilytetään ja muut tasot ladataan tai poistetaan asynkronisesti riippuen siitä, onko ne tallennuspisteessä määritetty. Ohjelma tarkistaa, että maailma on jo ladattu kertaalleen. Jos maailmaa ei olla ladattu aikaisemmin annetaan virhe.

Molempia metodeja käytetään täysin samalla tavalla. Metodeja kutsutaan käyttämällä StartCoroutine-metodia, jolloin kutsun on tapahduttava luokasta, joka perii MonoBehaviour-luokan. Metodeja voidaan kutsua parametrin kanssa tai ilman. Parametrin avulla voidaan hakea latautumisen edistymisaste. Edistymisaste lasketaan kaavan 1 mukaisesti, ja metodi palauttaa luvun nollan ja yhden välillä.

$$Edistymisaste = \frac{Ladattavan\ tason\ edistymisaste + Ladattujen\ tasojen\ määrä}{Ladattavien\ tasojen\ kokonaismäärä} \quad (1)$$

Edistymisastetta voidaan käyttää esimerkiksi latausikkunassa visualisoimaan lataamisen edistymistä. Jotta edistymisastetta voidaan ylläpitää jatkuvasti,

katkaisematta samassa metodissa olevan koodin suorittamista, käytetään metodin kutsunnassa Coroutine-Lambda-Action-CallBack-yhdistelmää. Kutsuja käyttää esimerkkikoodin 3 mukaista toteutusta, jossa Lambda-funktion aaltosulkeiden sisällä oleva osa jää päivittämään edistymisasteen määrää ja loppu metodista jatkaa suorittamista normaalisti.

```
public class WorldStarter : MonoBehaviour
{
    public delegate void WorldDelegate();
    public static WorldDelegate worldLoaded;

    public void StartGameLoadingBar()
    {
        StartCoroutine(LevelStreamer.LoadStartScenes(progress =>
        {
            Debug.Log(progress);
            if (Mathf.Approximately(progress, 1.0f))
            {
                worldLoaded();
            }
        }));
     //Koodin suorittaminen jatkuu odottamatta Coroutinen valmistumista
    }
}
```

Esimerkkikoodi 3. Yksinkertaistettu toteutus maailman aloitusvaiheesta.

Maailman latautumisen ajaksi voidaan haluta esimerkiksi muuttaa pelaajan kameran näkymää tai kytkeä komponentteja pois päältä. Muuten pelaaja voisi esimerkiksi nähdä maailman latautuvan. Toisaalta pelihahmo voisi pudota tyhjyyteen tai päätyä muuten saavuttamattomalle alueelle. Maailman latauduttua pelihahmon toiminnat halutaan palauttaa normaaliin pelitilaan.

Esimerkkikoodissa 3 käytetään delegate-mekanismia, jonka avulla voidaan lähettää julkinen tapahtuman lähetys worldLoaded()-komennolla. Tapahtumaan voi rekisteröityä kuuntelijaksi, jolloin latauksen valmistuessa koodia voidaan suorittaa eri paikoissa. Esimerkkikoodissa 4 rekisteröidytään kuuntelemaan staattista rajapintaa. EnablePlayer-metodia kutsutaan, kun maailma on latautunut.

```
using UnityEngine;

public class ExampleCalls : MonoBehaviour
{
    void Start()
    {
        WorldStarter.worldLoaded += EnablePlayer;
    }

    private void EnablePlayer()
    {
        Vector3 pos = StartPointHandler.CurrentStartPointLocation;
        gameObject.transform.position = pos;
     …
    }
}
```

Esimerkkikoodi 4. Tapahtumien kuuntelija, joka vastaanottaa tiedon latauksen valmistumisesta ja kutsuu EnablePlayer-metodia.

Pelaajan hahmon aloitussijainti haetaan myös esimerkkikoodissa 4 käyttämällä StartPointHandler-luokan CurrentStartPointLocation-rajapintaa. Rajapinta palauttaa Vector3-tyyppisen arvon, jonka avulla asetetaan pelaajan sijainti.

Pelin edistyessä pelaaja aktivoi kytkimiä kulkemalla niiden läpi tai suorittamalla tehtäviä pelissä. Kytkimen aktivoituessa tasoja ladataan ja poistetaan asynkronisesti taustalla. Onnistuneessa pelikokemuksessa pelaajan immersio säilyy ja tasot vaihtuvat saumattomasti ilman keskeytyksiä pelissä.

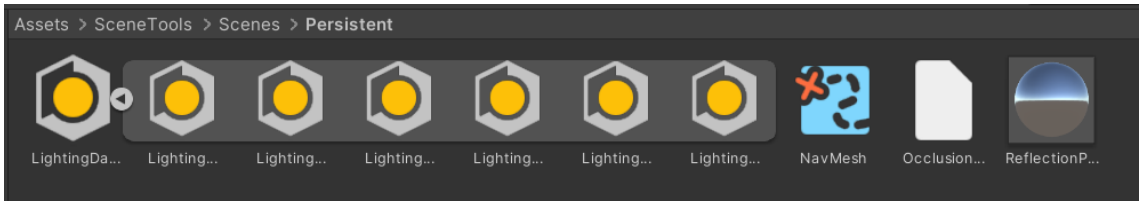
4.2 Kääntämisen ja pelitilan esivalmistelut

Tyypillisessä Unityllä tehdyssä pelissä on osa-alueita, joita pitää esirakentaa (engl. bake). Osa-alueet voidaan myös jättää esirakentamatta, mutta silloin ne eivät toimi pelatessa. Esirakentamista voivat tarvita projektista riippuen valaistus (lightning), heijastusluotaimet (reflection probe), navigaatioverkot (nav mesh) ja alueellinen objektin piilottaminen (occlusion culling). (29.)

Esirakentamisessa rakennusprosessi käyttää tasokohtaisia asetuksia osa-alueilla. Yksittäisen tason asetuksia voidaan muuttaa avaamalla taso erikseen ja muuttamalla asetuksia kunkin osa-alueen ikkunasta. Useiden tasojen ollessa

auki editorissa Unity käyttää aktiiviseksi merkityn tason asetuksia. Tällöin asetukset voidaan tehdä Persistent-tasossa. (30.)

Avoimen maailman työkalua käytettäessä staattisten osa-alueiden mahdollista automaattista rakentamista ei sovi käyttää, vaan ne rakennetaan manuaalisesti. Rakentaminen tehdään helpoiten ja suositellusti ensiksi avaamalla muokkaustilassa Persistent-taso ja kaikki tasot, joihin pelaaja voi koko maailmassa kulkea. Seuraavaksi varmistetaan, että Persistent-taso on aktiivinen taso, jolloin tason nimi on paksunnettu Hierarchy-ikkunassa. Lopuksi suoritetaan esirakentaminen, kuten normaalissa Unity-projektissa käyttäen kunkin osa-alueen omaa ikkunaa. Lopputuloksena työkalun esimerkkipelissä syntyvät kuvan 25 mukaiset tiedostot, jossa eri tasojen valaistuksen data on niputettuna Persistent-tason alaisuuteen.



Kuva 25. Esirakentamisen luomat tiedostot Persistent-nimisessä kansiossa.

Työkalun *Open Whole World* -painikkeeseen kannattaa konfiguroida avautumaan Persistent-taso ja kaikki maailmassa saavutettavissa olevat tasot. Tällöin esirakentaminen on helpompaa eikä vahingossa tule rakennettua tasoa niin, että siihen liittyviä muita tasoja ei olisi ladattu. Muuten seurauksena voisi olla esimerkiksi, etteivät yhdessä tasossa olevat puut tekisikään luonnonmukaisia varjoja toisen tason maastoon. Tasojenhallintavalikon kautta avatessa laitetaan Persistent-taso myös automaattisesti aktiiviseksi tasoksi.

Yksittäisiä tasoja voi joissain tapauksessa esirakentaa erikseen. Tämä kuitenkin vaatii syvällisempää ymmärrystä esirakennusprosesseista ja erityistä huolellisuutta, ja siksi sitä ei suositella työkalun peruskäyttäjälle. Joissain tapauksissa osa-alueesta riippuen pidetään koko maailman esirakennettu data muistissa.

Tällöin voidaan joutua muistin loppumisen takia kiinnittämään yksityiskohtaisempaa huomiota esirakentamiseen. (29.)

Unity tallentaa kaikkien auki olevien tasojen valaistusten datan nippuun aktiivisen tason alaisuuteen. Data sijoitetaan aktiivisen tason nimiseen kansioon. Eri tasojen valaistusten esirakennetun datan Unity osaa kuitenkin erotella automaattisesti. Tason valaistuksen data ladataan ja poistetaan samanaikaisesti yhdessä tason mukana. Tällöin tasojen yksitellen rakentaminen ei vaikuta valaistuksen muistinkäyttöön. Osaava pelinkehittäjä voi kuitenkin pystyä rakentamaan joitain tasoja erikseen ja säästää aikaa rakentamisen kestossa. Tasot on yleensä kuitenkin hyvä rakentaa yhdessä, jotta ne voivat vaikuttaa toistensa valaistukseen. (29.)

Heijastusluotainten osalta Unity aina lataa kerralla samaan aikaan esirakennetun datan. Tällöin koko maailman luotaimet ovat aina ladattuna. Luotaimet asetellaan pelinkehittäjän toimesta, jolloin niiden lukumäärällä voi vaikuttaa datan määrän. (29.)

Alueellisessa pintojen piilottamisessa piilotetaan ja näytetään objekteja esirakennetun datan mukaan. Tasoa ladattaessa etsitään muistista kyseisen tasoon viittauksena tallennettua dataa ja toimitaan sen mukaan. Kaikki saman datan alla olevat tasot käsitellään yhtenä tasona. Tällöin jos kaikki tasot rakennetaan Persistent-tasoon, on koko maailman data ladattuna muistissa. Tasot voivat silloin myös käyttää toisen tason objekteja pintojen piilottamiseen. Yksittäin tasoja rakennettaessa voidaan piilottaa vain saman tason objektien peittämiä pintoja. Yksittäisiä tasoja rakennettaessa ei kuitenkaan koko maailman dataa tarvitse pitää muistissa. (29.) Peleissä, joissa on paljon pelaajan liikkumiselta rajattuja alueita, voidaan käyttää Occlusion Area -objekteja rajoittamaan datan kokonaismäärää (31).

Navigaatioverkot rakennetaan aktiivisen tason nimiseen kansioon yhteen tiedostoon, joka sisältää datan kaikista avoinna olevista tasoista. Tällöin jos kaikki tasot rakennetaan Persistent-tasoon, on koko maailman data ladattuna muistissa. Muistin säästämiseksi navigaatioverkkoja voidaan myös rakentaa tasoille

erikseen, jolloin Unity-pelimoottori lataa ne automaattisesti tasojen latauksessa. (29.) Unityn NavMeshAgent (navigoinnista vastaava tekoälykomponentti) ei kuitenkaan osaa suoraan liikkua erikseen rakennetulta tasolta toiselle. Jotta NavMeshAgent osaisi kulkea eri tiedostoissa olevien datojen välillä, voidaan käyttää Off-Mesh link -objekteja (32). Tasojen navigaatioverkkojen esirakentaminen yksitellen lisää näin pelinkehityksen työmäärää sekä esirakentamisvaiheessa että ylimääräisten Off-Mesh link -objektien luonnissa.

Muistiin ladatuissa navigaatioverkoissa NavMeshAgent pystyy kulkemaan myös muuten lataamattomilla alueilla, koska NavMeshAgent on riippuvainen pelkästään navigaatioverkosta. Jos NavMeshAgent liikkuu alueella, jolta navigaatioverkot poistetaan, ei NavMeshAgent osaa toimia. (33.) Tällöin pitää tehdä myös erilliset ratkaisut puuttuvien navigaatioverkon osalta. Esimerkiksi ei-pelattavan hahmon NavMeshAgent voitaisiin kytkeä pois päältä ja hahmon sijainti voitaisiin laskea lineaarista interpolointia käyttäen ennalta päätetyn pelimaailman aikataulun mukaisesti. Kun pelattava hahmo siirtyy takaisin ladattavalle alueelle, siirrettäisiin ei-pelattava hahmo interpoloimalla laskettuun sijaintiin välittömästi ja NavMeshAgent kytkettäisiin takaisin päälle.

4.3 Pelin kääntäminen

Ilman kääntämistä peliä voi pelata editorissa, mutta ei sen ulkopuolella. Kun peliä halutaan pelata editorin ulkopuolella itsenäisenä sovelluksena, pitää se kääntää.

Työkalulla tehty peli käännetään samalla tavalla, kuin Unityllä tehty peli käännetään tavanomaisesti. Käännöskokoonpanosta (BuildSettings) tarkistetaan, että Persistent-taso ja muut halutut tasot löytyvät listalta. Seuraavaksi varmistetaan, että tarvittavat vaiheet esirakennusvaiheessa on tehty. Lopulta kääntäminen käynnistetään tavanomaisella tavalla File-valikon *Build Settings*- tai *Build And Run* -painikkeiden kautta. Käännösvaiheessa työkalu ei tee erillisiä toimenpiteitä. Käännetty peli voidaan käynnistää tavalliseen tapaan exe-tiedostosta tai laittaa käynnistymään automaattisesti käännöksen valmistuessa.

Käännöskokoonpanosta voidaan jättää tasoja pois testikäytössä. Tällöin työkalun kytkimet hylkäävät automaattisesti pois jätetyn tason viittauksen käännetyssä pelissä. Kun taso palautetaan käännöskokoonpanoon ja peli käännetään uudelleen, palautettu taso on taas automaattisesti pelattavissa.

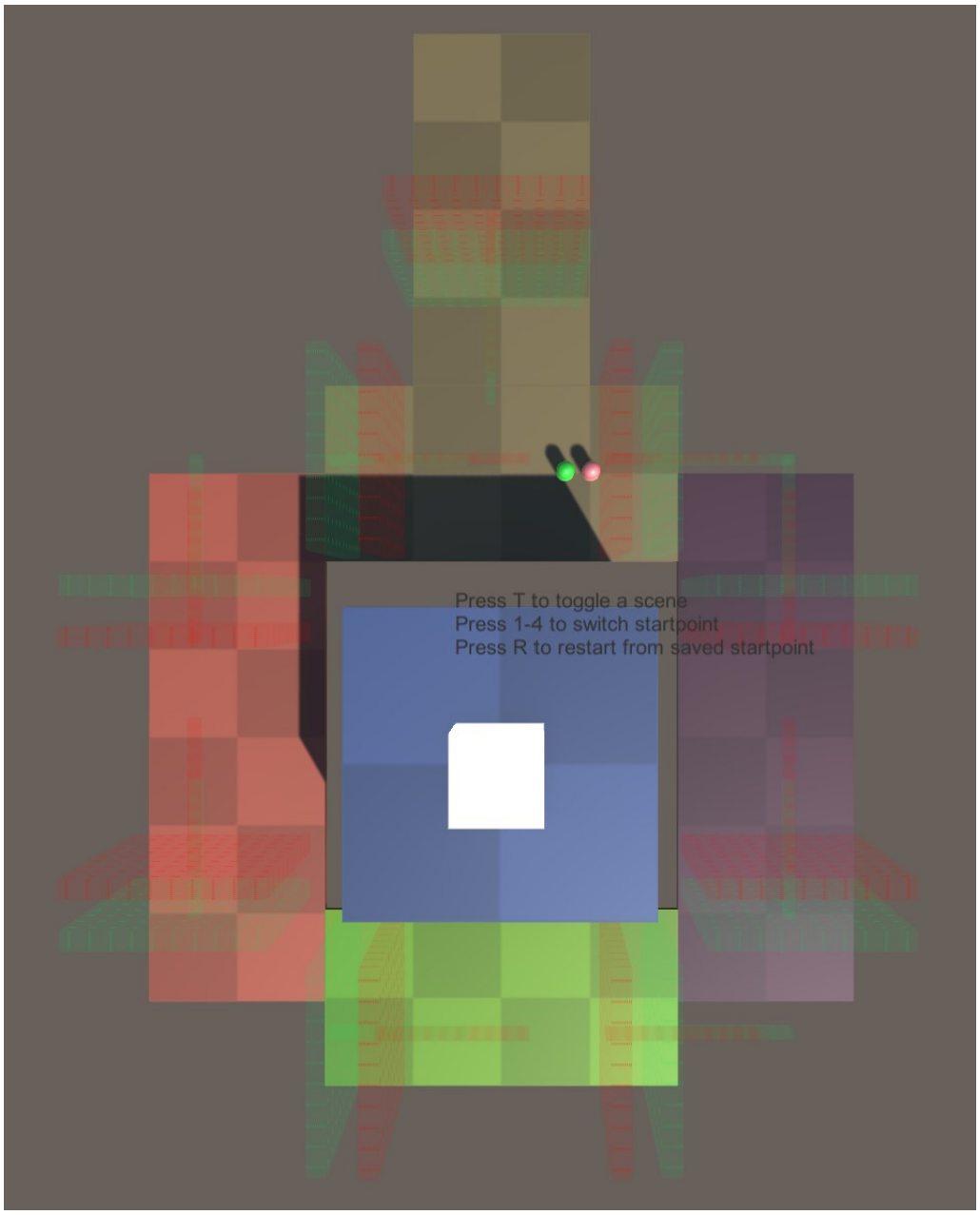
5 Työkalun arviointi

Tässä luvussa arvioidaan työkalun vaikutusta pelinkehityksen työskentelyyn, työkalun suorituskykyä sekä pohditaan mahdollisia jatkotutkimus- ja kehitysideoita.

Työkalusta tuli paketti, joka sisältää apuvälineitä avoimen maailman luomiseen ja hallintaan sekä tarvittavat mekaniikat maailman suoratoistamiseen. Työkaluun tuli tehtyä 28 operatiivista kooditiedostoa ja kuusi esimerkkikooditiedostoa toteutuksesta.

Projektin loppuvaiheessa tehdyssä esimerkkitoteutuksessa on sisällytetty kokonaisvaltainen, minimalistinen toteutus avoimen maailman pelin suoratoistomekaniikoista työkalua käytettäessä. Kuvassa 26 on toteutuksen yksinkertainen pelimaailma. Toteutuksessa on yhteensä kahdeksan tasoa, ja se sisältää latastason, valikkotason ja maailman tasot.

Esimerkkipelin maailma käynnistetään pelivalikosta nappulaa painamalla. Pelimaailman käytäviä pitkin kävellessä keskellä olevan sinisen kuution piilottamat tasot ladataan ja poistetaan pelihahmon koskiessa suoratoistokytkinten törmäystarkastajia. Ei-pelattava hahmo seuraa pelihahmoa tasolta toiselle, eikä se kykene aktivoimaan suoratoistokytkimiä. Pelissä voi vaihtaa aloituspisteitä numeronäppäimistä 1–4 ja käynnistää maailman uudelleen R-näppäimestä. T-näppäimen painaminen lataa ja poistaa vuoron perään maailman keskellä olevan valkoisen kuution sisältämän tason. Esimerkkipelissä on esirakennettu kaikki osa-alueet ja pelin toiminta varmistettiin käännetyssä pelissä.



Kuva 26. Työkalun esimerkkitoteutuksen maailma.

Työkalua rakennettiin alusta alkaen aiemmin tehdyn henkilökohtaisen 3D-peliprojektin (kuva 27) ympärille, jotta mielikuva oikeasta pelistä pysyi mielessä ja mekaniikkoja pystyi testaamaan aidommantuntuisessa ympäristössä. Alusta alkaen työkalun komponentit pidettiin irrallaan pelin komponenteista. Tällöin työkalu voitiin lopuksi irrottaa pelistä helposti omaan projektiinsa.

Kuva 27. Työkalun testauksen kohteena oleva vanha peliprojekti.

Peliprojektissa on käytetty maastossa yhteensä kahtatoista 250 yksikön (engl. unit) levyistä neliönmuotoista Terrain-objektia. Yksi yksikkö pelin maailmassa vastaa skaalauksen mukaisesti yhtä oikean maailman metriä. Pisimmälle kehitetyillä alueilla objektit ovat omissa tasoissaan. Kahdessa tasossa on kuitenkin pelin keskeneräisyyden takia neljä maasto-objektia tasoa kohden. Lisäksi pelissä on jonkin verran erilaisia koristeobjekteja, jotka ovat alueittain omissa tasoissa. Vuorten näkyessä kauempaa voidaan koristeobjekteja sisältävät tasot ladata hieman lyhyemmällä etäisyydellä kuin niiden kohdalla olevat maastot. Yhteensä pelin maailma koostuu 12 tasosta.

Kokonaispinta-ala koepelissä vastaa oikeassa maailmassa 0,75 km²:n aluetta. Pinta-ala on pieni verrattuna useisiin avoimen maailman peleihin, eikä objektien tiheys ole kovin suuri. Game Rantin pienimmistä avoimen maailman peleistä kertova artikkeli osoittaa kuitenkin, ettei maailman tarvitse aina olla valtava. Joukossa on tunnettuja pelejä, ja niiden pelimaailman oikeaa maailmaa vastaavat pinta-alat ovat väliltä 0,26–7,7 km². (7.) Peli tuntui kuitenkin riittävältä simuloimaan järkevänkokoista osaa avoimesta maailmasta, ja sitä käytettiin arvioimaan kaikkia työkalun ominaisuuksia. Koepelistä jäi mielikuva, että maailmaa voisi suorituskyvyn puolesta kasvattaa, jolloin peli olisi pidempi ja pelattavaa olisi enemmän.

5.1 Vaikutus työskentelyyn

Tasojen abstrahointi teki kytkinten kehitystyöstä helppoa ja huoletonta. Niiden ansiosta ei tasoja nimetessä tai järjestystä muokatessa tule vahingossa enää ladattua vääriä tasoja eikä tarvinnut nähdä vaivaa kytkinten korjauksessa. Myös tasoja pystyi editorin suorituskyvyn parantamiseksi ottamaan pois latauksesta ilman, että mitään rikkoutuisi pysyvästi maailmassa.

Kytkinten hallintatyökalut helpottivat kytkinten käyttöä. Kytkimiä ei enää tarvinnut yksitellen tarkistaa sen varalta, mitä tasoja ne lataisivat. Myös muutokset maailman alueissa olivat helppoja korjata, kun tasoviittauksia pystyi etsimään ja korvaamaan. Kytkinten näkyvyyden pystyi myös helposti ja pysyvästi piilottamaan tai palauttamaan painikkeen painalluksella vaikuttamatta niiden toimintaan. Näihin ongelmiin työkalu toimi mielestäni hyvänä ratkaisuna.

Tässä kohtaa on kysymyksenä, kuinka paljon työkalu oikeasti säästää aikaa ja vähentää virheitä pelinkehittäjältä. Vastaus riippunee projektin koosta ja monimutkaisuudesta, mutta ainakin koekehittäjä pystyi havaitsemaan pelinkehityksen mielekkyyden ja käyttökokemuksen parantumisen kytkinten kanssa työskentelyssä.

Koeympäristö myös mahdollisti tasojenhallintavalikon testaamisen monipuolisesti. Hallintavalikko toimi sujuvasti ja ilahduttavasti. Päällimmäisenä kysymyksenä mieleen jäi kuitenkin, kannattiko valikon tekeminen ajallisesti. Ilman valikkoa tasojen availu on kömpelöä ja hieman ärsyttävää, mutta käytetyn ajan kestoa on helppo liioitella. Valikon tekemisessä kesti kuitenkin kauan, joten sitä voisi omassa käytössä joutua käyttämään pitkään, ennen kuin ajallinen sijoitus kannattaisi. Peliprojektin kehityksen vaihe toki vaikuttaa avattavien tasojen määrään. Lopulta kuitenkin valikon tekemistä perusteltiin työkalua suunnitellessa ensisijaisesti pelinkehittäjän käyttömukavuuden kannalta.

Hallintavalikon oli tarkoitus myös helpottaa tasojen konflikteja, tiimityöskentelyä ja virhetilanteita. Konflikteja ei päästy testaamaan yhden kehittäjän projektissa, mutta tasojen osittaminen ja tarkastelu oli hyvin sujuvaa, mikä helpottaisi versionhallintaa. Työkalusta jäi hyvä yleisvaikutelma. Helposti konfiguroitavat esivalinnat varmistivat myös, että kaikki tasot olivat ladattuina esirakennusvaiheessa.

5.2 Suorituskyky

Koepeliin mahtui kolme isompaa pääaluetta ja pieniä osioita niiden sisällä, mikä mahdollisti työkalun kokeilun hieman luontevammassa ympäristössä. Koeympäristössä suoratoistaja suoriutui kelvollisesti eikä merkityksellistä suoratoistoviivettä syntynyt. Koepeli oli pienehkö eikä varmuutta jäänyt, miten suoratoistaja suoriutuisi valtavista alueista tai suuresta sisällön tiheydestä. Koepeli oli kuitenkin optimoimaton, eikä se silti indikoinut suorituskyvyn laskemista. Vertailukelpoinen testaus vaatisi suuremman, mutta huolellisesti optimoidun pelimaailman.

Tasojen hallintatyökalujen painikkeiden vasteet olivat välittömän tuntuisia koeprojektissa. Käytettäessä vapaavalinnaisen tason avausikkunaa voidaan tasolistaan valita näkymään kaikki projektin tasot. Käytäessä kaikki projektin tiedostot läpi, on tiedostojen lukumäärällä suora vaikutus vasteaikaan. Työkalu ei kuitenkaan havaittavasti lisännyt vasteaikaa verrattuna ilman työkalua operoimiseen. Muissa painikkeissa tasot luetaan käännöskokoonpanosta tai valmiista viittauksista, jolloin tarvittavien iteraatioiden määrä pysyy hillitympänä.

Kytkinten hallintatyökalujen painikkeet käyvät kaikki Persistent-tason objektit ja niiden kaikki lapsiobjektit läpi yksi kerrallaan etsiessään kytkinkomponentteja. Tämän kesto on suoraan verrannollinen tason objektien määrän. Koepelissä vaste oli välittömän tuntuinen, kun Persistent-tason objektien määrä oli sadoissa.

5.3 Jatkokehitysideoita

Projektin tarkoituksena oli luoda myyntikelpoinen työkalu. Erilaisia ideoita lisäominaisuuksiin tuli projektin aikana. Raportin kirjoitushetkellä työkalua haluttiin vielä laajentaa ennen potentiaalista julkaisua. Työkalu olisi kuitenkin julkaisukelpoinen.

Työkalu on kompleksinen ja siinä on paljon erilaisia ominaisuuksia ja koodeja. Työkalun ominaisuuksien testaaminen jatkuvien muutosten tapahtuessa oli työlästä ja aikaa vievää. Muutos yhdessä paikassa saattoi aiheuttaa yllättävän ongelman toisessa paikassa. Näin ollen työkalun ja koodien automaattisen testaamisen implementoiminen tuntuisi olevan erityisen tärkeä lisäys työkalua jatkokehittäessä. Erilaisia testauskeinoja voisi olla esimerkiksi yksikkö- tai integraatiotestaaminen. Samoin työkalun toiminnallisuutta erilaisten Unity-versioiden kesken olisi hyvä testata automaattisesti ennen kuin työkalu julkaistaan.

Työkalun jatkokehitysideoista tärkeimmältä vaikuttava ominaisuus oli automaattinen tasojen yhdistäminen käännösvaiheessa. Ajatuksena oli, että valikosta voitaisiin valita listamuotoisesti tasoja, jotka käännösvaiheessa sulautettaisiin automaattisesti yhteen toisten valittujen tasojen kanssa. Tätä harkittiin tehtäväksi insinöörityöprojektin aikana, mutta ominaisuus rajautui pois, koska se oli irrallaan ydinajatuksesta.

Tasojen viittaustiedostoja varten voisi olla valikossa lisäpainike. Viittaustiedostojen automaattinen luonti käännöskokoonpanon pohjalta olisi mielekäs lisäys.

Työkalun esimerkkipeli voisi olla laadukkaampi, ja myös joitain visuaalisia parannuksia voisi työkaluun tehdä. Ensisijaisena visuaalisena muutoksena olisi

parantaa kytkimiä, jolloin editori näyttäisi kytkimen kohteena olevat tasot kytkimen päällä tekstinä.

Editori voisi automaattisesti piirtää viivan kytkimen kohteena olevien tasojen keskipisteeseen tai näyttää tasoja lähimpänä olevan kytkimen kyljen. Ominaisuus voitaisiin virittää myös havaitsemaan epäjohdonmukaisuuksia kohteena olevien tasojen sijainnissa.

Monipuolisuuden vuoksi työkaluun voisi harkita etäisyyspohjaisen suoratoistamisen implementaatiota. Samoin voisi harkita erilaisten Prefab-, AssetBundle tai Addressable-tyyppisten pakettien suoratoistamisen mahdollistamista tasojen ohella. Lisäksi voisi selvittää, millaisia muita koonti- ja optimointiratkaisuja on olemassa, joita voisi työkalussa tukea suoraan.

Unity vaatii, että Unity Asset Storessa julkaistu paketti toimii uusimmalla Unity-versiolla (34). On myös hyvän tavan mukaista, että paketti toimisi useimmilla uusimmilla versioilla. Tiedossa ei ole, että jokin Unityn versio 2019.4.37f1:n jälkeen ei tukisi työkalua. Tiedon varmistaminen on kuitenkin oleellista, jos työkalu laitetaan myyntiin.

Erilaisia pelialustoja ei Windows-käyttöjärjestelmän lisäksi kokeiltu. Tiedossa ei ole, että jokin alusta ei tukisi työkalua. Kuitenkin ennen julkaisua tämä olisi syytä varmistaa.

6 Yhteenveto

Insinöörityössä perehdyttiin avoimen pelimaailman toteuttamiseen ja tehtiin työkalu avoimen 3D-maailman luomisen helpottamiseksi. Työkalu suunniteltiin siten, että se voitaisiin myöhemmin laittaa myyntiin. Työkalu koostui suoratoistajasta komponentteineen sekä avoimen maailman kehitystä helpottavista aputyökaluista.

Työkaluun tehtiin keskeisimpänä osana avoimen pelimaailman suoratoistaja, jonka avulla pelaaja voi saumattomasti liikkua pelattavassa maailmassa ilman

peliä keskeyttäviä latausikkunoita. Suoratoistaja tehtiin noudattamaan pelimaailmassa tapahtuvia muutoksia käyttämällä kytkimiä. Suoratoistotapahtuman käynnistäviä tapahtumia voisi esimerkiksi olla pelaajan kulkeminen tietyn käytävän läpi tai tehtävän suorittaminen pelimaailmassa.

Suoratoistajan kannalta oli oleellista tehdä keinot käynnistää pelimaailma pelin aloituksesta ja pelaajan edistymisen seurauksena tallennetuista aloituspisteistä. Tallennettuja aloituspisteitä varten luotiin oma mekanismi. Maailman käynnistämiseksi implementoitiin työkaluun esimerkkitoteutus avoimen maailman pelistä kokonaisuudessaan.

Työkalu tehtiin myös nopeuttamaan ja helpottamaan kehitystyötä. Työkaluun lisättiin käyttömukavuutta parantavia ja virheitä minimoivia aputyökaluja. Aputyökaluina oli erilaisia editorin laajennuksena tehtäviä elementtejä, kuten valikkoja, painikkeita, ikkunoita, tiedostoja sekä erilaisia muokattuja näkymiä. Aputyökaluja tehtiin suoratoistajan ydinosan valmistumisen jälkeen koepelin muokkauksen ohessa. Tällöin yhtenä tavoitteena ollut aputyökalujen tuoma kehitystyön mielekkyyden parantuminen voitiin huomata jo projektityöskentelyn aikana.

Aputyökalut auttoivat muun muassa maailman asetusten muokkaamisessa, tallennuspisteiden luomisessa ja muokkaamisessa, tasojen hallinnassa ja avaamisessa editorin muokkaustilassa sekä suoratoistokytkinten tarkistelussa ja muokkaamisessa. Pelitasoille tehtiin oma abstraktiokerros vähentämään tasojen muutoksista aiheutuvaa lisätyötä erityisesti kytkinten päivittämisen tarpeen poistamiseksi. Lisäksi muuttujien ja ikkunoiden käytettävyyttä ja näkymien selkeyttä parannettiin. Myös virhetilanteita estettiin ohjelmallisesti.

Työkalu implementoitiin kokonaisuudessaan vanhaan henkilökohtaiseen peliprojektiin, joka sisälsi maastoja, erilaisia 3D-malleja ja pelattavaa sisältöä. Koepeli oli pienehkö, oikeata maailmaa vastaavalta pinta-alaltaan 0,75 km².

Koepelin todettiin kuitenkin vertailussa olevan riittävän suuri joihinkin pienempiin olemassa oleviin avoimen maailman peleihin nähden. Näin voitiin luottavaisemmin mielin testata kaikki työkalun ominaisuudet läpi koepelin kanssa. Työkalu

suoriutui moitteetta valikkotyökalujen osalta. Myös suoratoistaja latasi alueita sulavasti ja mielikuva maailman kasvattamisen mahdollisuudesta säilyi. Jotta suurempaa maailmaa olisi voitu vertailukelpoisesti testata suoratoistajassa, olisi pelimaailman pitänyt olla optimoitu. Myös aputyökaluista jäi yksin työskennellessä pienehkön projektin kanssa epäselvyys niiden todellisesta potentiaalista kehitystyön nopeuttamiseksi. Työkalusta jäi hyvä yleiskuva, ja työkalun kehitystyötä erilaisten ominaisuuksien lisäämiseksi voisi hyvin jatkaa tulevaisuudessa. Työkalusta tuli julkaisukelpoinen ja näin ollen se voitaisiin laittaa myyntiin.

Lähteet

1 Welcome to Unity. 2022. Verkkoaineisto. Unity Technologies. <https://unity.com/our-company https://unity.com/our-company>. Luettu 28.3.2022.

2 Moss, Richard. 2017. Roam free: A history of open-world gaming. Verkkoaineisto. Ars Technica. <https://arstechnica.com/gaming/2017/03/youre-now-free-to-move-about-vice-city-a-history-of-open-world-gaming/>. 25.3.2017. Luettu 8.4.2022.

3 Pedroza-O'Leary, Joshua. 2019. 10 Massive Problems With Open-World Games (That We All Need To Admit). Verkkoaineisto. Game Rant. <https://gamerant.com/big-problems-open-world-video-games/>. 22.11.2019. Luettu 9.2.2022.

4 Buchalter, Jacob. 2022. The Best Open-World Games Of All Time, Ranked. Verkkoaineisto. TheGamer. <https://www.thegamer.com/best-open-world-games-all-time-officially-ranked/>. 14.3.2022. Luettu 10.4.2022.

5 Loveridge, Sam. 2022. Best open world games to play right now and completely forget real life exists. Verkkoaineisto. GamesRadar. <https://www.gamesradar.com/best-open-world-games/>. 8.4.2022. Luettu 10.4.2022.

6 Delatte, Thomas. 2019. The 25 Biggest Open-World Video Games, Ranked By Size. Verkkoaineisto. Screen Rant. <https://screenrant.com/biggest-open-world-video-games-ranked-size/>. 1.4.2019. Luettu 8.4.2022.

7 Llewellyn, Michael. 2020. The 10 Smallest Open World Games (Based On The Size Of Their Maps). Verkkoaineisto. Game Rant. <https://gamerant.com/yakuza-gta-3-small-world-maps-open-world-games/>. 15.1.2020. Luettu 8.4.2022.

8 Making the World of Firewatch. 2016. Verkkoaineisto. Alphabet Inc. <https://www.youtube.com/watch?v=hTqmk1Zs_1I>. 20.5.2016. Luettu 5.1.2022.

9 Jenkins, Barry. 2021. The Journey To Open-World Gaming. Verkkoaineisto. Forbes. <https://www.forbes.com/sites/forbestechcouncil/2021/07/21/the-journey-to-open-world-gaming/?sh=2835939a126b>. 21.1.2021. Luettu 14.1.2022.

10 Level of Detail. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/LevelOfDetail.html>. Luettu 2.3.2022.

11 Occlusion culling. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/OcclusionCulling.html>. Luettu 2.3.2022.

12 Visibility and Occlusion Culling. 2022. Verkkoaineisto. Epic Games. <https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/VisibilityCulling/ >. Luettu 4.4.2022.

13 Text-Based Scene Files. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/TextSceneFormat.html>. Luettu 4.2.2022.

14 Unity 5.3. 2015. Verkkoaineisto. Unity Technologies. <https://unity3d.com/unity/whats-new/unity-5.3>. Luettu 10.4.2022.

15 Work with multiple scenes in Unity. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/MultiSceneEditing.html>. Luettu 11.1.2022.

16 Garbage collector overview. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/performance-garbage-collector.html>. Luettu 29.1.2022.

17 Open World Streaming in Unity. 2019. Verkkoaineisto. Spellcast Studios. <https://ardenfall.com/blog/world-streaming-in-unity>. 25.9.2019. Luettu 17.1.2022.

18 How to use the Unity SceneManager. 2017. Verkkoaineisto. Myriad Games Studio. <http://myriadgamesstudio.com/how-to-use-the-unity-scenemanager/>. 29.8.2017. Luettu 1.2.2022.

19 Long Term Support. 2022. Verkkoaineisto. Unity Technologies. <https://unity.com/releases/lts-vs-tech-stream>. Luettu 4.4.2022.

20 How Addressables interact with other project assets. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Packages/com.unity.addressables@1.18/manual/ManagingAssets.html>. Luettu 1.2.2022.

21 Scene. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/ScriptReference/SceneManagement.Scene.html>. Luettu 15.1.2022.

22 SceneManager.LoadSceneAsync. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadSceneAsync.html>. Luettu 15.1.2022.

23 EditorSceneManager.OpenScene. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/ScriptReference/SceneManagement.EditorSceneManager.OpenScene.html>. Luettu 15.1.2022.

24 Working with the Event System. 2022. Verkkoaineisto. Unity Technologies. <https://learn.unity.com/tutorial/working-with-the-event-system#5fb9711aedbc2a36e711497c>. Luettu 15.3.2022.

25 Layer-based collision detection. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/LayerBasedCollision.html>. Luettu 19.4.2022.

26 UnityEvents. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/UnityEvents.html>. Luettu 23.3.2022.

27 Application.backgroundLoadingPriority. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/ScriptReference/Application-backgroundLoadingPriority.html>. Luettu 19.1.2022.

28 SceneManager.SetActiveScene. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.SetActiveScene.html>. Luettu 1.2.2022.

29 Bake data in multiple scenes. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/bakemultiplescenes.html>. Luettu 13.2.2022.

30 Setup multiple scenes. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/setupmultiplescenes.html>. Luettu 30.1.2022.

31 Occlusion Areas. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/class-OcclusionArea.html>. Luettu 1.3.2022.

32 Loading Multiple NavMeshes using Additive Loading. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/nav-AdditiveLoading.html>. Luettu 1.3.2022.

33 Navigation System in Unity. 2022. Verkkoaineisto. Unity Technologies. <https://docs.unity3d.com/Manual/nav-NavigationSystem.html>. Luettu 12.3.2022.

34 Submission Guidelines. 2022. Verkkoaineisto. Unity Technologies. <https://assetstore.unity.com/publishing/submission-guidelines#>. Luettu 1.3.2022.