Ville Väisänen

PELIPROJEKTIN TAUSTAJÄRJESTELMÄT OSANA PELIKOKONAISUUTTA

Opinnäytetyö
Tietojenkäsittely

2018

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Ville Väisänen	Tradenomi (AMK)	Marraskuu 2018

Opinnäytetyön nimi

Peliprojektin taustajärjestelmät osana pelikokonaisuutta

38 sivua
0 liitesivua

Toimeksiantaja

Kamina Dimension

Ohjaaja

Jukka Selin, Kalle Sievänen

Tiivistelmä

Tämä opinnäytetyö käsittelee Unity-pelimoottoriin perustuvan peliprojektin taustajärjestelmien toimintaa, merkitystä sekä vaikutusta osana toimivaa monipuolista pelikokonaisuutta. Opinnäytetyön tutkimus syventyy tarkasti peliprojektin järjestelmien tehtäviin ja tehokkaan toiminnan vaatimuksiin. Vaatimuskriteereiksi käyttäjäystävälliselle pelaajakokemukselle muodostuivat menujärjestelmän yksinkertaisuus, nopeus sekä pelaajan häiritsemättömyys. Taustajärjestelmien oikeanmukaisen suunnittelun sekä toteutuksen avulla pelin viihdyttävyys nousee arvossaan immersion avulla. Immersio vaikuttaa pelaajan syventyneisyyteen ympäröivään sisältöön kasvattaen pelaajan elämystä sekä pelikokemusta.

Samalla tutkimus käsittelee tallennus- ja latausmenetelmien käyttötapoja yksittäisten menetelmien positiiviset ja negatiiviset puolet huomioon otettuna sekä tutkii peliprojektin tiedonhallinnan tarpeita löytääkseen tehokkaimmat metodit sekä kokonaisuuden monipuolista tiedonhallintajärjestelmää varten. Unity-peliprojektissa tallennusmetodeja on useita käytettävänä, jolloin monia metodeja samanaikaisesti käytettäessä hyödynnetään yksittäisen metodin tuomat hyvät ominaisuudet sekä ehkäistään negatiiviset vaikutukset, jonka seurauksena voidaan peliprojektiin luoda turvallinen, tehokas sekä monipuolisesti toimiva tiedonhallintajärjestelmä.

Opinnäytetyössä tutkitaan aluksi pelin taustajärjestelmien toimintaa sekä taustajärjestelmien luomaa vaikutusta pelikokonaisuuteen aluksi menujärjestelmää tarkastellen. Lisäksi opinnäytetyössä tutkitaan muita taustajärjestelmiä, kuten asetuksien hallintaa sekä navigaatiojärjestelmää. Taustajärjestelmistä tutkimus siirtyy tiedon hallintaan, tallennukseen sekä lataukseen. Tiedonhallintaa seuraavassa osassa tarkastellaan käytännön työn rakennetta, sisältöä, ominaisuuksia sekä toimintoja. Lopuksi viimeisessä osassa opinnäytetyö päättyy tietoon tavoitteen täyttymisestä, projektin haasteista, sekä saadusta kokemuksesta.

Päätuloksena käytännön tuotos toteutui täysin toimeksiantajan suunnitellun toimeksiannon vaatimuksien mukaisesti. Toimeksiantaja Kamina Dimension jatkaa pelikokonaisuuden työstämistä käyttäen opinnäytetyön käytännön tuotosta mukana tulevassa pelissään MindSeizessä. Käytännön tuotoksen visuaalisuus muuttuu mahdollisesti toteutetusta ulkomuodostaan pelin kehittyessä.

Asiasanat

peliohjelmointi, Unity2D, tiedontallennus, taustajärjestelmät

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Ville Väisänen	Bachelor of Business Administration	November 2018

Thesis title

Games' back end systems as a part of a game project

38 pages
0 pages of appendices

Commissioned by

Kamina Dimension

Supervisor

Jukka Selin, Kalle Sievänen

Abstract

The aim of this bachelor's thesis was to study in which ways a game project's back end systems could have an effect on the game project as a whole. A versatile system mostly comprised a fully functioning menu system, game customization system, and an information management system that was developed as a result of this thesis. This thesis was done by utilizing the Unity game engine with C sharp as its programming language.

The first part of the study examined the basis of back end systems in games by investigating how back end systems could be designed to suit the essential need of a satisfying user experience. The second part of the study examined information management as a system and the usability of its varying applicable methods in a game project based on the Unity game engine.

The empirical part of the study gave an overview of the structural and mechanical design of the game's back end systems by viewing its every part developed separately. This part also demonstrated the interaction as well as the variety of unique functions and effects between the separate back end systems.

The main findings of the study were that the game project's back end systems had the best outcome in terms of a satisfying user experience by reducing additional elements of the user interface, consequently keeping the user interface simplified, but easily usable. The user interface of the back end systems had to be developed so that all the usable content was sharply available to the user without disruptive game elements impairing the user experience. The information management system functions the most effectively by utilizing many saving and loading methods together, as a result weakening a unique method's disadvantages and simultaneously enhancing its advantages.

The assignment and its objectives defined by the case company Kamina Dimension were fully accomplished, and the system created will be used in Kamina Dimension's game MindSeize. The created system was developed to be easily modified for future utilization.

Keywords

game programming, Unity2D, information storage, back end systems

SISÄLTÖ

1 JOHDANTO

Suomen peliteollisuus on pelinkehityksen alkuajoista alkaen ylläpitänyt suosiotaan kansainvälisillä markkinoilla saakka uraa uurtavilla peleillään. Tästä esimerkkeinä ovat Remedy Entertainmentin vuonna 2001 julkaisema Max Payne sekä nykyvuosiin asti maailmanlaajuiseen suosioon päässyt Rovio Entertainmentin Angry Birds-mobiilipelisaaga. Uusia pelialan pienyrityksiä luodaan jatkuvasti, ja kysyntä peliohjelmoijista on suuri. Nykypäivänä Rovio Entertainmentin ja Supercellin kaltaiset suomalaiset menestystarinat inspiroivat suomalaista pelialaa, ja samalla vahvistavat suomalaisen tuotannollisen standardin mukaista laatua ohjelmistopuolella. Teknologian kehittyessä uudet laitteet, kuten VR eli Virtual Realityyn perustuvat Oculus Rift ja HTC Vive sekä säännöllisesti julkaistavat uudet konsolit jatkuvasti kehittävät lisää tuotettavuutta sekä joukosta erottuvaa ainutlaatuista sisältöä. Uusi teknologia sekä uudet konsolit vetävät yhä enemmän ihmisiä mukaansa pelien jatkuvasti muuttuvaan ja kehittyvään maailmaan.

Tässä opinnäytetyössä perehdyn peliprojektin välttämättömien taustajärjestelmien valmistukseen, rakenteeseen, merkitykseen osana pelikokonaisuutta sekä eri pelijärjestelmien keskinäiseen interaktioon. Tutustun läheisesti pelin tiedon tallennus- ja lataustoimintoihin sekä tallennustapojen positiivisiin ja negatiivisiin puoliin ja käyttötapoihin. Tallennus- ja latausmetodeista kertoessa otan myös kantaa tallennettavan tiedon suojaamiseen sekä turvalliseen tiedon hallintaan.

Toimeksiannon opinnäytetyölle sain Kuopiossa toimivalta pelialan yritykseltä, Kamina Dimensionilta. Tehtävänäni oli toteuttaa valikkojärjestelmä asetuksineen, tietoluettelojärjestelmä, hahmon kehitysjärjestelmä, tallennus- ja latausjärjestelmä sekä muita pelin tausta- ja tukijärjestelmiä Kamina Dimensionin tulevaan peliin MindSeizeen.

2 PELIN JÄRJESTELMÄRAKENNE

Oma tietämykseni peliohjelmoinnista sekä pelien eri osien, kuten taustajärjestelmien sekä menujärjestelmän tehtävistä ja vaikutuksesta pelikokonaisuuteen, perustuu oppimaani tietoon ammattikorkeakoulun peliohjelmointikurs-

seilta sekä oppimaani käytännön taitoihin tehdessäni harjoittelun ja opinnäytetyöni Kamina Dimensionille. Perustin oman tekemäni työni useista lähteistä keräämääni tietoon, joiden neuvojen avulla osittain muodostin toiminnallisen pohjan työlleni, pitäen mielessä lukemani lähteiden neuvot sekä huomiot. Raportissa käyn läpi tausta- ja menujärjestelmien toimintoja sekä vaatimuksia osana pelikokonaisuutta. Menujärjestelmällä on suuri tehtävä yhdistäessä peliin tehdyn sisällön eri toiminnot navigoitaviksi ja käytettäviksi. Menujärjestelmän toimiessa pelin hallitsevana navigaatiokeskuksena on välttämätöntä tutkia menujärjestelmään lisättävien elementtien tehtävää osana menua sekä menujärjestelmäkokonaisuuden vaikutukset muuhun pelisisältöön ja pelaajan käyttökokemukseen. Tallennus- ja latausmetodit ovat tärkeä osa pelin runkoa. Tallennusjärjestelmät keräävät tietoa useista eri projektin osista sekä vuorovaikuttavat kokonaisuuden kanssa lataamalla tallennettua tietoa takaisin sitä vaativille järjestelmille. Avaan tallennusmetodien toimintoja sekä pohdin eri tallennusmetodien parhaita käyttöympäristöjä pohjautuen tallennusmetodien ominaisuuksien positiivisiin ja negatiivisiin puoliin.

Pelin kokonaisuuden suuri osuus rakentuu pelillistä sisältöä tuottamattomista, mutta peliin merkittävästi vaikuttavista taustajärjestelmistä. Vuosia aiemmin Super Marion kaltaisten pelattavuudeltaan yksinkertaisten, mutta aikansa viihdyttävyydessä kilpailemattomien pelien ollessa suosiossa, pelit eivät antaneet pelaajalle mahdollisuutta räätälöidä pelattavuutta pelaajan omalle tavalleen parhaiten sopivaksi. Osasyynä tähän on se, että vuosia aiemmin pelit olivat sisällöltään sekä pelityyliltään paljon nykyistä yksinkertaisempia. Pelejä usein pelattiin Nintendon kaltaisilla konsoleilla, joiden ohjaimet koostuivat noin kahdeksasta napista, joista vain osaa käytettiin aktiivisesti. Klassisen Super Marion tapauksessa hahmo pystyi liikkumaan vasemmalle ja oikealle, kiipeämään, kyykistymään, hyppäämään sekä ampumaan tai käyttämään erikoiskykyään. Hahmon liikemahdollisuuksien ollessa niin rajoittuneet ei pelaajalle ollut tarvetta antaa toimintoa muuttaa näppäinyhdistelmiä niiden yleiskäyttöisyyden vuoksi. Vanhemmissa peleissä graafisten asetusten muokkauskaan ei ollut suuri prioriteetti, sillä aiemmat konsolit, tietokoneet sekä vanhempi teknologia rajoittivat grafiikan laatutason pieneksi.

2.1 Menujärjestelmä osana pelikokonaisuutta

Menujärjestelmä osana peliä omien toimintojensa lisäksi luo pelin ensivaikutelman. Hyvä menujärjestelmä on yksinkertainen pelaajan heti ymmärrettävissä oleva Unity-pelimoottorin termillä scene. Scenet ovat peliin lisättävän ja käytettävän tiedon kokonaisuuksia, joihin peli rakennetaan ja sisällytetään. Scenejä voidaan ajatella kirjan sivuina, joissa jokainen sivu on toisestaan eroava uniikki osa yhtenevää kokonaisuutta. Lopulta pelin valmistuessa peli kootaan tehdyistä sceneistä. Unity-pelimoottorissa scene aluksi luodessaan sisältää ainoastaan kamerakomponentin, jonka avulla Unityn käyttöliittymässä peli nähdään sitä kokeiltaessa, eli vaadittu minimi kokeilemaan projektia. Myöhemmin sisällön lisääntyessä ja pelin eteenpäin kehittyessään sceneen alkaa lisääntymään luotuja järjestelmiä, jotka myötävaikuttavat pelin toimintaan. Sceneen luodut järjestelmät sisältävät pelin mekaniikat, yleisen toimivuuden sekä hallittavat tiedot.

Menujärjestelmän ehkäpä suurin tehtävä peleissä on toimia scenejä hallitsevana kokonaisuutena yhdistäen tausta- ja tukijärjestelmät käytettäväksi navigoimalla menun scenestä haluttuihin tausta- ja tukijärjestelmien hallitsemiin sceneihin. Esimerkiksi MindSeizen menujärjestelmään lisäsin navigoitavaksi vaihtoehdoksi asetukset-scenen, jota hallitsee asetuksia ylläpitävä tukijärjestelmä. Baker (2017) kertoo menun sekä graafisen käyttöliittymän olevan essentiaalinen osa pelikokonaisuuden rakennetta. Pelin navigointi operoi menun sekä graafisen käyttöliittymän kanssa vuorovaikutuksessa toimiakseen. Hän yhdistää tähän pelin toimivuudelle välttämättömään osaan toiminnot, kuten pelin aloituksen, pelin tallennuksien ja aiempien pelikertojen hallinnan, asetusten muuttamisen sekä taustajärjestelmien kanssa vuorovaikuttamisen. Itse pelattava sisältö rakennetaan menujärjestelmän ympärille, jota menujärjestelmä yhdessä muiden taustajärjestelmien kanssa sitten hallitsee luoden pelaajalle käytettävän järjestelmän.

Single Player, eli yksin pelattavan pelin menu rakennetaan usein niin, että pelaaja pääsee navigoimaan mahdollisimman pienellä vaivalla haluamaansa kohteeseen. Menun tarkoituksena ei ole pitää pelaajaa paikallaan, menu sen sijaan pyrkii mahdollisimman tehokkaasti siirtämään pelaajan eteenpäin kohti

pelattavaa sisältöä. Baker (2017) pohtii kirjoituksessaan, miksi useissa peleissä pelaajalle annetaan tehtäväksi painaa mitä tahansa näppäintä ennen edes menuun pääsyä. Ainoana selityksenä hän uskoo olevan vanhat suuret arcade-pelilaitteet, jotka vaativat kolikoita pelatakseen. Siitä huolimatta nykyaikaisissa peleissä turha käsky luo yhden vaikkakin pienen ja nopeasti suoritetun vältettävissä olevan lisätehtävän pelaajalle. Samanlaista toimintoa on näkynyt jokaisella pelituotannon vuosikymmenellä Capcomin ja Nintendon klassikoista uusiin suurten peliyhtiöiden julkaisuihin saakka.

Videopeliaiheisen blogisivusto Kotakun toimittaja Hamilton (2015) kertoo hyvän videopelin menun olevan kuin hyvä roudari, sillä se pysyy poissa tieltä. Silti hänen mukaansa liian moni videopelin menujärjestelmä kuluttaa pelaajan aikaa liialti turhiin tai huonosti suunniteltuihin prosesseihin. Pelaajat haluavat pelata hankkimaansa peliä menussa pyörimisen sijasta. Samalla hän myös puhuu kymmenestä säännöstä pelien menujärjestelmille, joiden tehtävänä on nopeuttaa peliin pääsyä sekä vähentää pelaajan vaiheita ennen peliin menoa. Esimerkiksi peliin pääsyä voi nopeuttaa automaattisesti tallentamalla valitut asetukset ilman, että pelaajan täytyy itse erikseen asettaa asetuksensa valmiiksi. Uuden innoitusta herättävän pelin ostaessaan pelin menujärjestelmä voi muodostaa mielenkiitoisen ja mieluisen vaikutelman. Myöhemmin useita kertoja uudelleen menuun tullessa, mahdollisten ylimääräisten prosessien sekä liiallisten vaiheiden aiheuttama vaivaannuttava odotus vain kasvattaa epämukavuutta. Pelien ja varsinkin menujärjestelmien toteutuksessa tulee ottaa huomioon, että jokaisella pelikerralla pelaajan täytyy käydä sama pelin aloitusprosessi läpi. Tästä syystä pelaajaa helpottavalla ajatusmaailmalla kehitin menujärjestelmän, jonka prioriteettina on minimoida pelaajalta vaaditut turhat toiminnot ja samalla automatisoida osia pelin asetuksista.

Tekemässäni menujärjestelmässä pelaajaa ei johdateta suorittamaan ylimääräisiä toimintoja menujärjestelmään päästäkseen. Esimerkkinä ylimääräisestä toiminnosta on pelaajan tarve painaa näppäintä päästäkseen menuun.
Sen sijaan peli avatessaan menee suoraan menuun samalla laittaen pelin asetukset pelaajan käyttämän laitteiston mukaan sopivimmalle tasolle, esimerkiksi muuttamalla pelin resoluution näytön käyttämän resoluution mukaiseksi. Varsinkin ensimmäisellä pelikerralla asetuselementtien automatisointi vaikut-

taa henkilön kykyyn pystyä saamaan otteen pelistä antamalla pelaajalle mahdollisuuden siirtyä suoraan pelin pariin pakollisen asetuksissa käymisen sijaan.

Lovato (2015) kirjoittaa opittujen tapojen tai tottumuksien positiivisesta vaikutuksesta peleissä. Hänen mukaansa opittavat tottumukset, tavat ja rutiinit ovat mekaanisia, joten pelaaja, joka kehittää tavan tai tottumuksen pelata peliä myös todennäköisesti myöhemminkin palaa pelin pariin pelatakseen lisää. Tottumuksien kehittäminen on avain pitkäkestoisen peliyhteisön muodostamiseksi pitkällä aikatähtäimellä (Lovato 2015). Tavoilla on merkittävä positiivinen vaikutus pelin menestymisen kannalta. On siis tärkeää miettiä rutiinin omaisen prosessin vaiheet aloituksesta pelin käynnistymiseen toistuvan osan peliä nopeimmin rutinoituessa. Pienet yksityiskohdat suunnittelussa tulevat vahvemmin esille rutiinin omaisesti kehitetyn tavan vaiheita suoritettaessa, verrattuna muutamia kertoja satunnaisesti peliä pelaamiseen. Aktiiviset tavan muodostaneet pelaajat omaksuvat lihasmuistiinsa pelaamisen lisäksi sen, mitä nappeja täytyy painaa navigoidakseen peliin käsiksi. Esimerkiksi tavan oppinut pelaaja omaksuu pelisovelluksen avauksen menun kautta ’Load game’- näppäimeen ja navigointireitit päämääriin. Jottei tavasta tulisi turhauttava rutiini, ja sen sijaan osa positiivista vaikutusta pelin kannalta, on mahdollisimman nopea ja vaivaton pääsy peliin myöskin ratkaiseva osa pelaajan positiivista käyttökokemusta. Jos menut tehtäisiin mahdollisimman yksinkertaisiksi, mutta samalla visuaalisesti miellyttäviksi, ei pelaajakaan kehitä negatiivista mieltymää vaivaannuttavasta aloitusprosessista. Aloitusprosessiin voi sisältyä useasti nähtyjä ohittamattomia visuaalisia videoklippejä pelistä tai muutoin turhaa odotusta pidättäen pelaajan poissa pelin pääsisällöstä, jota varten pelaaja on pelin avannut.

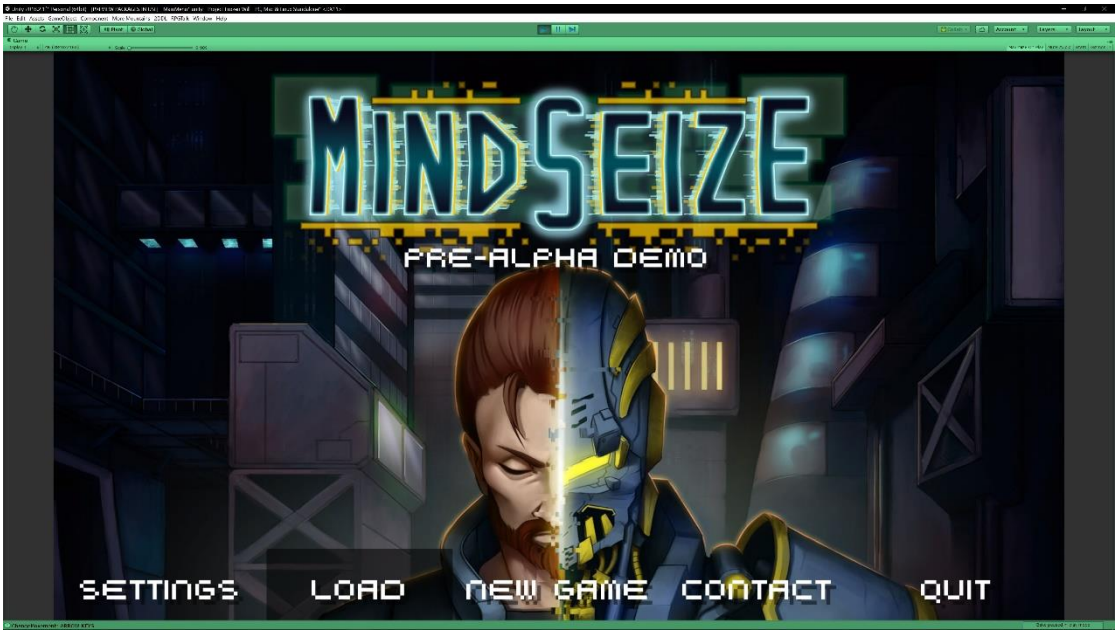
Baker (2017) ehdottaa pelaajan kokemuksen parantamiseksi ohittaa toistuvilla pelikerroilla menun suoraan, ja menun sijasta ladata pelaajalle automaattisesti aiempi tallennus. Pelaaja tarvitsee käyttöönsä täyttä menun toiminnallisuutta ja navigointia ensimmäisellä pelikerralla, jolloin pelaaja voi tutustua peliin ja asettaa asetukset haluamakseen. Tämän ensimmäisen kerran jälkeen pelaajan tarvitsee harvoin muuttaa aiemmin asetettuja tallennettuja asetuksiaan, jonka takia pelaaja voisi suoraan ohittaa menun ja päästä heti jatkamaan peliä (Baker *2017*). Pelaajalla on lähes aina kuitenkin mahdollisuus kesken pelin

muuttaa asetuksia tai palata menujärjestelmään. Oletettavasti useimmiten pelaaja haluaa pelin aloittaessaan asetusten vaihtamisen tai uuden pelin aloittamisen sijaan jatkaa aiempaa pelisessiotaan. Ehdotus nopeuttaisi huomattavasti pelaajan päätavoitteeseen, eli pelaamiseen pääsyä, pelaajan ylimääräisiä askeleita pelin käynnistämisen jälkeen ottamatta.

Pienillä yksityiskohdilla menussa on valtava merkitys pelaajan pelielämykselle. Tästä esimerkkinä on Hamiltonin (2015) artikkelissa sääntö numero 9, jonka mukaan pelin menuun tullessaan ensimmäinen valittava näppäin olisi oltava ’continue’. Tällöin pelaaja pääsee yhden napin painalluksella suoraan aiemmin tallennettuun peliin ylimääräistä navigointia tekemättä. Pelin menua suunnitellessa voisi tai tulisi jopa ajatella, että menu on pelaajalle vaivaa tuottava pelin toiminnalle välttämätön osa pelikokonaisuutta. Pelaaja täytyy ohjata nopeasti eteenpäin suuremmalta haitalta välttyäkseen, varsinkin yksinpeleissä.

Videopelien suosituimpana pelialustana käytetty Steam on lähes kaikille tietokonepelien pelaajille väistämättä tuttu ohjelma, joka sisältää tuhansia indie-yhtiöiden sekä tunnetuimpien peliyhtiöiden pelejä. Steam-pelialustasta halutun pelin kauppasivulle mentäessä pelin trailerit käynnistyvät automaattisesti esittelemään peliä, jolloin pelin ulkoasu sekä tyyli tulevat varmasti selville pelin ostajalle. Jollei henkilö kuitenkaan ole nähnyt traileria on hänellä varmasti muuta tietoa pelistä, ellei ulkopuoliset ihmiset ole vaikuttamassa henkilön ostopäätökseen. Tällöin voisi olettaa henkilön tietävän pelistä pelin aloittaessaan. Silti osa peleistä ennen pelin alkua sisältää ohittamattomia trailereita, jotka pelaaja velvoitetaan katsomaan päästäkseen pelaamaan itse peliä. Peliä pelaamalla juoni epäilemättä selviäisi, joten ohittamattomat trailerit hidastavat itse peliprosessiin pääsyä tuomatta minkäänlaista lisäarvoa pelaajalle.

MindSeizeen tein yksinkertaisen, mutta tehokkaan menujärjestelmän, jonka mekaniikat loin aiemmin mainittujen artikkeleiden neuvojen pohjalta. Tein sen siten, että pidin menun vain porttina eteenpäin minimoiden pelaajan tarpeen odottaa tai navigoida päästäkseen pelaamaan. Tekemäni menu sisältää mahdollisimman vähän huomiota herpaannuttavia elementtejä, jottei pelaajan keskittymisen kohde siirtyisi pois pelaamisesta.



Kuva 1. MindSeize-pelin päämenu Unity-pelimoottorin editorissa

Suunnittelin menun korostamaan taustan tuomaa ensivaikutusta sijoittamalla menussa käytettävät painikkeet näytön alareunaan (kuva 1). Menuun on ohjelmoitu toiminto, joka tarkastaa onko pelaaja aiemmin pelannut peliä ja tallentanut pelisessionsa. Siinä tapauksessa, että peli löytää tallennetun pelitiedon, pelin menun käynnistyessä kohdistettu näppäin on suoraan ”Load”-näppäimen päällä, joka vie yhtä näppäintä painamalla pelaajan vaivatta suoraan takaisin jatkamaan aiempaa pelisessiota. Muutoin jos tallennusta ei löydy, on ”Load”-napin toiminto on sama kuin ”New game”-napillakin, uusi peli käynnistyy. Pelin menun näppäimien asettaminen alapuolelle näyttöä korostaa taustan tuomaa vaikuttavaa ensivaikutelmaa. Näppäimet ovat silti selkeästi huomattavissa. Pelin menusta pääsee myös asetuksiin sekä tarkastamaan peliyhtiön yhteystiedot. MindSeizen tuotanto on vielä demovaiheessa, joten oli tärkeä lisätä yrityksen yhteystiedot napin päähän päämenusta. Menussa mainitaan vielä pelin kehitysvaihe peliotsikon alapuolella. MindSeizen menu kiteyttää menun vaaditut tehtävät pääpiirteisiinsä ja tuo ne pelaajalle käyttöönsä tehokkaasti.

Verrattuna yksinpeleihin, moninpelien menujärjestelmän tehtävät ja rooli osana pelikokonaisuutta saattavat erota huomattavasti. Moninpelien menut nykyään usein sisältävät informatiivisia elementtejä kertoen peliyhteisöstä, uutisista sekä peliin liittyvistä tapahtumista tai suorista lähetyksistä. Nykyään

trendaaviksi peleiksi kasvaneissa Battle Royale-moninpeleissä, kuten suositussa Player Unknown's Battlegroundsissa sekä Fortnitessä, menulla on tärkeä rooli ryhmän muodostamisessa ja hallinnassa.

Kuva 2. Esimerkki interaktiivisesta moninpelin menujärjestelmästä

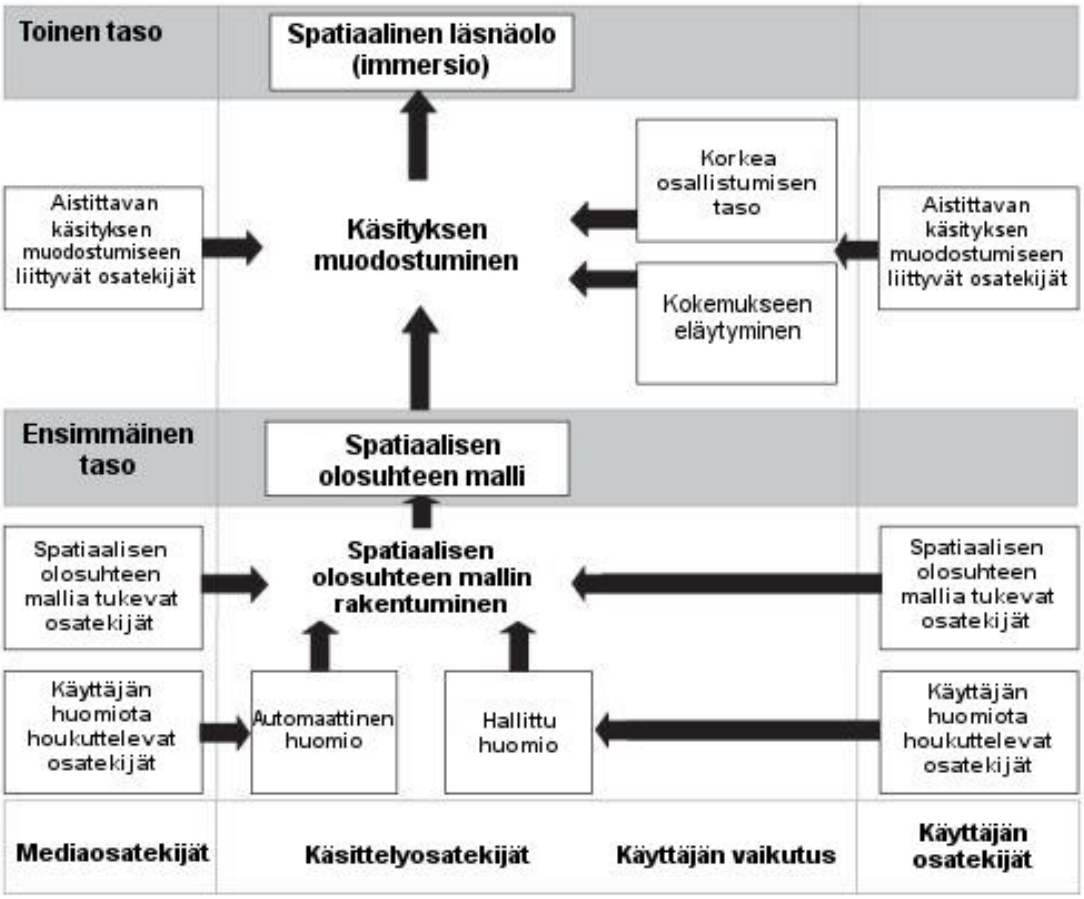
Kuvassa 2 on esimerkki hyvästä yksinkertaisesta menusta, jossa visuaalisesti hahmotetaan pelaajan ryhmän kokoonpanoa ja lukumäärää. Samalla kuvassa 2 esitetään tärkeää tietoa visuaalisten elementtien avulla vaikuttamatta heikentävästi menun käytettävyyteen sekä käyttöliittymään. Varsinkin peleissä, joissa menua käytetään työkaluna ryhmän kokoamisessa, menuun liitetään elementtejä, jotta muita odottaessa pelaaja voi tarkastaa menun kautta peliin liittyviä uutisia tai muiden pelaajien tekemää yhteisössä suosittua sisältöä. Elektronisen urheilun sekä pelitapahtumien suosion kasvaessa monet pelikehittäjät ovat lisänneet peliin tiedotuksen pelin suorista lähetyksistä, varsinkin kilpailullisissa peleissä kasvattaen yhteisön aktiivisuutta.

Muita vertaan otettavia huomioita Hamiltonin (2015) artikkelissa liittyen menujärjestelmän ideaaliseen suunnitteluun on ylimääräisten pelisisältöä tuottamattomien elementtien ohittamattomuuden ongelma. Esimerkkinä ohittamattomista elementeistä ovat ennen pelin menua näkyvät aloitusruudut, jotka esittävät pelin tekijöiden ja pelissä osallisina olleiden logot sekä mahdolliset ennen menua tulevat pelin videotiivistelmät. Mahdollinen ratkaisu aloitusruutujen ohitettavuudelle on koodin kautta tarkastaa, onko pelaaja aiemmin päässyt aloitusruuduista edemmäs pelimenuun saakka. Peli voisi pelaajan pelimenuun

päästyään tallentaa tiedon ja mahdollistaa aloitusruutujen ohittamisen seuraavilla tulevilla pelikerroilla. Pelin sisällölle merkityksettömät elementit tulisivat olla ohitettavissa pelaajat huomioon ottaen pienten yksityiskohtien ollessa niin merkittäviä mieluisuuden kannalta.

2.2 Immersion hyödyntäminen peliprojektissa

Yksi pelin menestymisen ja suosion kulmakivistä on pelin immersio. Ihmisillä on usein tapana sijoittaa itsensä kuultuihin tarinoihin tai samaistua elokuvien pääosassa oleviin hahmoihin. Tällöin pelaaja tuntee itsensä olevan osa tarinaa, jolloin tarinasta tulee henkilökohtaisempi. Saatava tunne tai viihtyvyys henkilön syventyessä aiheeseen on huomattavasti vahvempi verrattuna esimerkiksi toiseen aiheeseen, johon henkilö ei pysty samaistumaan. Madigan (2010) määrittelee immersion videopeleissä olevan tunne, jossa pelaaja kokee olevansa tai sijaitsevansa näkemässään pelimaailmassa. Immersion vaikutus peliin on ilmeinen. Pelaajan tuntiessa olevansa osana koettavaa peliä ja mahdollisesti samaistuessa pelattavaan hahmoon on pelikokemuskin rikkaampi verrattuna peliin, jonka sisältö tai maailma ei saa pelaajaa syventymään näkemäänsä tai sen tarinaan. Immersioon vaikuttaa myös henkilön halukkuus olla syventynyt kohteeseen. Pelin aiheen ollessa lähellä omaa mielenkiintoa, syventyminen ja immersio peliin muiden elementtien sen mahdollistaessa on todennäköisempää.



Kuva 3. Malli immersion rakenteesta ja muodostumisen tekijöistä (Madigan 2010)

Madigan (2010) määrittelee immersion muodostuvan automaattisesta huomiosta, hallitusta huomiosta, käyttäjän osallistumisesta kokemukseen sekä kokemukseen eläytymisestä (kuva 3). Immersio perustuu koettavaan syventyneisyyteen, joten menu- ja taustajärjestelmillä pelissä on suuri vaikutus immersion toteutumiseen. Esimerkkinä voidaan pitää ilmoituspalkkeja tallennuksesta tai mitä tahansa muuta pelistä keskittymisen vievää elementtiä, jonka Madigan (2010) artikkelissaan sanoo sopimattomien visuaalisten vihjeiden tai merkkien puutteen pelin maailmassa olevan yksi mielenkiintoisemmista vaadittavista edeltäjistä pelaajan spatiaaliselle mukanaololle eli immersiolle. Esimerkiksi jos uskottavan tapahtuman pelimaailmassa kokiessaan pelaajaa huomautetaan ilmestyvällä pelielementillä, pelaajan peliin kehittynyt immersio heikkenee. Esimerkkejä immersiota heikentävistä elementeistä peleissä ovat palkin sisällä ilmestyvät ilmoitukset, apuviestit sekä saavutusilmoitukset. Pelaajan immersion täytyy pitää jatkuvana sitä keskeyttämättä parhaan kokemuksen saavuttaakseen. Tästä syystä esimerkiksi menu- ja taustajärjestelmät vaikuttavat suuresti

pelikokemukseen, jonka vuoksi menu- ja taustajärjestelmät on tärkeää suunnitella tarkasti olemaan turhaan häiritsemättä tai keskeyttämättä pelaajan kokemusta.

Hyvänä esimerkkinä peleissä toteutetusta immersion säilyttämisestä, mutta silti tärkeän tiedon esittämisestä pelaajalle on käyttää pelin automaattisen tallennuksen kertomiseen ison huomiota vievän ilmestyvän tekstin ja palkin sijaan peliin liittyvää pelaajan tuntemaa kuvaketta alareunaan sijoitettuna. Tällöin pelaaja voi huomata merkin, muttei joudu keskeyttämään syventyneisyyttään. Immersion muodostumiseen vaikuttaa merkittävästi myös pelikokonaisuuden pelillinen sekä visuaalinen tyyli. Peli voi esimerkiksi tyyliltään suuntautua kohti tarinallisuuteen pohjautuvaa peliä, jolloin pelaaja helposti uppoaa mukaan juoneen. Peli voi myös pohjautua enemmän nopeaan toimintaan pääpainona pelattavuus tarinan sijaan, jolloin pelaaja nauttii sisällöstä, muttei samalla tavalla uppoudu ympäröivään maailmaan ja tarinaan. Immersio on yleisempää yksinpeleissä, jolloin muut pelaajat eivät pääse vaikuttamaan omaan pelilliseen kokemukseen. Yksinpeleissä pelaajalla on oma rauha nähdä sisältö suunnitellusti pelin tekijöiden haluamalla tavalla. Toisaalta moninpeleissä muut pelaajat voivat lisätä kiinnostavuutta vuorovaikuttamalla vastavuoroisesti kehittäen yhteistyöhön perustuvaa pelisisältöä, joka muuttaa pelityyliä alkuperäisestä tyylistään ja mahdollisesti luo pelien sisäisiä peliyhteisöjä. Tällöin immersion vaikutusta peliin ei tarvitse ottaa huomioon samalla tavalla kuin yksinpeleissä, koska yhteistyö muiden pelaajien kanssa helposti rikkoo immersion muiden pelaajien vuorovaikutuksen perusteella.

2.3 Taustajärjestelmät osana pelikokonaisuutta

Nykyaikana teknologian sekä pelien kehittyessä yhä monipuolisimmiksi täytyy peliin luoda mahdollisuus pelaajalle muokata pelin elementtejä. Tästä esimerkkeinä ovat grafiikka, äänenvoimakkuus, käytettävät näppäimet sekä resoluutio. Varsinkin tietokoneen yleistyminen pelillisenä laitteena lisää tarvetta kehittää muokattavuutta tietokonetta käytettävän näppäimistön näppäinmäärän sekä korkean tehokynnyksen takia. Yksi pelin menestyksen takaamisen kulmakivistä on hyvän sekä monipuolisen suunnitelman laatiminen. Toisin kuin aiemmin, nykyaikana monille tulee ongelmaksi pystyä pelaamaan uusimpia yhä korkeampaa tehoa vaativia pelejä parhailla asetuksillaan. Usein pelaajien

täytyy alentaa graafisia asetuksia pystyäkseen pelaamaan ilman pelikokemusta hankaloittavia ongelmia, kuten FPS (Frames Per Second) -vähäisyyttä tai näytön jäätymistä.

Pelin mielenkiinnon ja sisällön kiinnostavuuden säilyttämiseksi sekä kohentamiseksi nykyaikaisiin peleihin luodaan pelillistä sisältöä tukevia taustajärjestelmiä. MindSeizessä menujärjestelmä niputtaa taustajärjestelmät yhteen navigoitavaksi käytettäväksi sisällöksi, jonka avulla menujärjestelmän merkitys osana pelikokonaisuutta kasvaa. Taustajärjestelmät voivat olla suoraan pelin ohella toimivia pelaajaan vuorovaikuttamattomia järjestelmiä, jotka keräävät tietoa ja käyttävät sitä esimerkiksi osoittaakseen pelaajalle hänen keräämänsä pisteet ja käytetyn ajan ja esimerkiksi pelin läpäisyprosentin. Sen lisäksi taustajärjestelmät voivat olla vaikkapa tiedon säilyttäjiä, jotka palauttavat pelaajan hahmon menehtyessään takaisin viimeksi tallennettuun turvapaikkaan. Taustajärjestelmät voivat myös olla interaktiivisia pelaajan muokattavissa tai käytettävissä olevia kokonaisuuksia, kuten esimerkiksi asetusjärjestelmä. Asetusjärjestelmässä pelaaja voi muokata äänenvoimakkuutta tai vaikkapa pelin resoluutioasetusta ja valittuja käytettäviä näppäimiä.

Itse peliä pelatessaan pelaaja harvoin ymmärtää tai ajattelee, miten paljon taustalla toimivia näkymättömiä järjestelmiä peli samanaikaisesti hallitsee. Jokainen pelin toiminto vaatii ohjelmoidun järjestelmän toiminnon suorittaakseen. Jotta pelaaja voi esimerkiksi käyttää yksinkertaisempia pelin toimintoja, kuten pysäyttää pelin halutessaan tai vaikkapa keskustella toiselle pelin hahmolle, täytyy kyseinen toiminto olla luotuna käytettäväksi järjestelmään. Jo-

kaista toimintoa varten on joko yksi järjestelmä, joka on koonnut kaikki toiminnot itseensä, tai useita pienempiä järjestelmiä, jotka tekevät yksityiskohtaisesti omat toimintonsa.

Kuva 4. MindSeizen pausemenu Unity-pelimoottorin editorissa

MindSeize-pelissä menuun sisällytettyjä taustajärjestelmiä ovat kartta- ja navigointijärjestelmä, asetukset sekä tietoluettelo. Osa navigoitavista järjestelmistä on sijoitettu päämenuun, kun taas osa on sijoitettuna pelin pysäyttävään menuun, eli pausemenuun (kuva 4). Pausemenun avatessaan pelisessio pysähtyy ja peliä voi jatkaa painamalla ’Resume’- painiketta. Pausemenu visuaalisesti esittää hahmon sijainnin ja kyseisen huoneen, sekä missä huoneissa pelaaja on käynyt. Tekemäni menu sisältää vain pohjan vaadituille mekaniikoille Kamina Dimensionin toimeksiannon mukaisesti. Jotkin taustajärjestelmistä, kuten esimerkiksi tekemäni hahmon kehitysjärjestelmä sisältää informaatiota, joka kehittää pelikokemusta. Hahmon kehitysjärjestelmä venyttää pelisisällön viihtyvyyttä pidemmälle eteenpäin ajoittamalla hahmon kehityksen ja sitä kautta uuden pelattavuuden mahdollistamisen pelistä kerättävien objektien löytämisen avulla. Hahmoa kehittäessään itse pelisession pelityyli muuttuu ja muokkaa pelaajan kokemusta aiemmasta, jonka avulla peli ylläpitää mielenkiinnon pelissä vähentämällä pelillistä toiminnallista toistuvuutta. Pelaaja voi myös halutessaan muokata valittuja kehityksiä sekä uudelleen asettaa ne halutukseen.

Taustajärjestelmien täytyy yleensä tehokkaasti toimiakseen olla vuorovaikutuksessa toisten järjestelmien kanssa. Taustajärjestelmät voivat jakaa oman kerätyn tietonsa muille tallennettavaksi ja käytettäväksi, tai säilyttää tietoa itsessään, jota taustajärjestelmä käyttää myöhemmin tietoa pyydettäessä. Tiedonhallinta on keskeinen elementti pelin toiminnan kannalta. Tiedon tallennusta ja käyttöä voisi jopa pitää yhtenä pelin tärkeimmistä mekaniikoista. Ehkäpä olennaisin käyttötarkoitus tiedon tallennukselle yksinkertaisissa peleissä on pelisession tallennus ja lataus, joka mahdollistaa pelin keskeyttämisen halutessaan. Samalla tiedon hallinta mahdollistaa jatkaa samaa sessiota uudelleen pelin myöhemmin avatessaan. Muita korvaamattoman tärkeitä toimintoja tallennuksella ja latauksella peleissä on esimerkiksi peliscenen tallennus ja tiedon hallinta muihin taustajärjestelmäsceneihin siirtyessä, hahmon kehityksen tallennus ja päivitys, kerättyjen esineiden tallennus (inventaariojärjestelmät yms.), pelin sisäisesti alueelta toiselle siirtyminen sceneä vaihtamalla säilyttäen pelaajan hahmon tiedot, ja niin edelleen. Listaa voi jatkaa eteenpäin lukemattomat eri järjestelmät sekä niiden toiminnot mukaan lukien, riippuen luonnollisesti pelistä ja sen sisällöstä.

MindSeizessä taustajärjestelmien tietoja käytetään esimerkiksi ylläpitämään loogista navigaatiota sekä käyttöympäristökokonaisuutta. Ilman tiedon säilyttämistä tai tallentamista, taustajärjestelmät eivät voi hallita pitkäkestoisesti omia annettuja toimintojaan aiemman tiedon puuttuessa. Tiedon puute aiheuttaa tehotonta toimivuutta järjestelmissä. Esimerkkinä tehottomuudesta ja toiminnottomuudesta otan omassa opinnäytetyössä kohtaamani ongelman, jolloin en ollut luonut tiedonhallintajärjestelmää tukemaan taustajärjestelmiä. Projektissani tuli olla pelatessa mahdollisuutena avata pelin pysäyttävä menu (pausemenu). Pausemenu pysäyttää pelin ja avaa ikkunan, josta voi navigoida toisiin sceneihin, kuten päämenuun sekä asetuksiin. Unity-pelimoottori oletuksena sceneä vaihtaessa poistaa aiemman scenen tiedot ja lataa uuden scenen tilalle omin tietoineen. Ilman tallennusjärjestelmää scenen vaihto onnistuu moitteetta, mutta aiemman scenen tiedot katoavat ja uudelleen pelattaessa palautuvat alkuperäisiin arvoihinsa. Esimerkiksi MindSeizessä asetukset-sceneen mentäessä aiempi peliscene katoaa ja uusi asetukset-scene aukeaa tilalle. Asetukset-sceneen pääsee joko suoraan pelin päämenusta tai pelin sisäisestä menusta, kun taas asetuksista palataan painamalla yhtä palaa-näppäintä, joka palauttaa pelaajan aiempaan sceneen. Ilman tiedon tallennusta

peli ei tiedä, mikä oli aiempi avoin scene. Tässä tilanteessa täytyisi valita palaako pelaaja päämenuun vaiko pelin sisäiseen menuun palauttamatta pelaajaa siihen sceneen, missä pelaaja aiemmin sijaitsi.

Toinen ongelma hahmottui pelattavaan sceneen palatessa asetukset-scenestä. Pelin sisäinen menu oli sulkeutunut, vaikka vaatimuksena pausemenulle oli pysyä auki, jos se oli aiemmin avattu asetuksiin siirtyessä. Tallennusjärjestelmä tuo paljon mahdollisuuksia pelin mekaniikkojen hienosäätöön ja hyvin suunniteltuna tukee ja tehostaa pelin toimivuutta sekä parantaa pelaajan käyttökokemusta. Unity-pelimoottorissa käytin hyväkseni kolmea yleistä tallennusmetodia, joista kerron seuraavaksi. Samalla tutkin näiden kolmen käyttämäni tallennusmetodin käyttötapoja ja niiden hyötytekijöitä.

3 TIEDONHALLINTA UNITY-PROJEKTISSA

Uccello (2017) määrittää Unity-pelimoottorilla tiedon tallennuksen keinot neljään metodiin: playerprefsiin, serialisaatioon, deserialisaatioon sekä JSON-tallennukseen. MindSeizessä käytin näistä tallennuskeinoista playerprefs-metodia sekä serialisaatiota ja deserialisaatiota, jotka itse yhdistän ja tulkitsen molemmat termit yhdessä serialisaationa. JSON-tallennusta en projektissa käyttänyt. Uccellon (2017) mukaan JSON-tallennuksen hyötypuolena on se, että JSON-tekstitiedostoon voi tallentaa tietoa esimerkiksi peliprojektista ja myöhemmin käyttää samaa tallennettua tietoa muussa käyttöympäristössä. Toinen käyttöympäristö voi esimerkiksi käyttää eri ohjelmointikieltä, jolle JSON-tallennuksen avulla lokalisoidaan tallennettu tieto. Lokalisointi tarkoittaa tiedon muokkausta käytettäväksi toisessa käyttöympäristössä. MindSeizessä kaikki käytetty sisältö pysyy Unity-peliprojektin sisällä, jolloin tiedon lokalisointi muiden käyttöympäristöjen käytettäväksi on turha ominaisuus. JSON-tallennuksen sijaan valitsin käytettäväkseni tiedon serialisaation, joka sopii kyseiseen projektiin paremmin verrattuna JSON-tallennusmetodiin.

3.1 PlayerPrefs

Mahdollisesti yleisin ja helpoiten käyttöön otettava tallennusmetodi Unity-pelimoottorissa on playerprefs-metodi. Playerprefs on Unity-pelimoottorin oma tallennusmetodi, joka on suoraan käytettävissä pelin kooditiedostoissa ylimääräisiä lisäyksiä tekemättä. Playerprefs-funktion avulla voi tallentaa ja ladata float-, int-, ja string-arvoja, eli liukuarvoja, kokonaislukuja sekä tekstiä. Playerprefs-tallennusmenetelmä on kätevä siitä syystä, että float-, int-, ja string-arvoja voi käyttää yleisimpiin tarpeisiin. Int-arvoa voi myös käyttää boolean-arvona antamalla boolean-arvolle int-arvot 1 tai 0, eli true tai false. Muihin tallennusmenetelmiin verrattuna Playerprefs vaatii suhteellisen vähän työtä, tiedon tallennuksen ja saman tiedon latauksen voi suorittaa kahdella komennolla. Playerprefsin helppokäyttöisyyden vuoksi varsinkin Unity-pelimoottoria käyttävät peliohjelmointia aloittelevat henkilöt omaksuvat nopeasti Playerprefsin käytön. Playerprefsin avulla voi tehdä nopeasti yksinkertaisen tallennusjärjestelmän, jonka pohjalle saa tallennettua yleisimmät tarpeet, kuten hahmon sijainnit x-, y- ja z-koordinaatistossa liukuarvoilla tai vaikkapa pelaajan keräämät pisteet kokonaislukuja käyttämällä.

Playerprefsillä luodaan itse nimettäviä avaimia, joille samalla annetaan jokin arvo riippuen valitusta arvotyypistä. MindSeizessä tein Playerprefsin HasKey-funktiolla pelin käynnistyessä tarkastuksen, onko pelaaja aiemmin käynnistänyt peliä. Tarkastustiedon perusteella koodi asettaa asetukset sopimaan pelaajan laitteiston kanssa parhaalla tasolla. HasKey-funktio tarkastaa onko kysytyn nimistä playerprefs-avainta tehty, jonka perusteella voi esimerkiksi laittaa pelin käymään läpi asetukset ja asettamaan ne oikein tai siirtymään suoraan menuun.

Nimi	Laji	Data
(oletus)	REG_SZ	(arvoa ei ole asetettu)
aiempiLevel_h3922268426	REG_BINARY	4d 61 69 6e 4d 65 6e 75 31 00
AlotusVolume_h3768426013	REG_DWORD	0x00000001 (1)
BeforeMuteEffect_h3866324930	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
BeforeMuteVolume_h3439047421	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
BrightnessArvo_h169116326	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
DashKey_h3864859532	REG_BINARY	58 00
EffectArvo_h1604163384	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
EffectBloom_h1392835729	REG_DWORD	0x00000000 (0)
EffectPercent_h3029634345	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
EffectsMuted_h1209704556	REG_DWORD	0x00000001 (1)
EffectsMutedCheck_h2576057546	REG_DWORD	0x00000001 (1)
FPSCheck_h2278437254	REG_DWORD	0x00000001 (1)
FPSToggle_h2755302386	REG_DWORD	0x00000000 (0)
FromGame_h369640957	REG_DWORD	0x00000000 (0)
GMCheck_h175081193	REG_DWORD	0x00000000 (0)
GrenadeKey_h1049261676	REG_BINARY	53 00
HealKey_h2008903826	REG_BINARY	41 00
InventoryLoad_h186670345	REG_DWORD	0x00000000 (0)
InventoryOpen_h186502907	REG_DWORD	0x00000000 (0)
InventorySave_h187489870	REG_DWORD	0x00000000 (0)
JumpKey_h198900752	REG_BINARY	5a 00
LevelManagerSingleSpawn_h619533825	REG_DWORD	0x00000001 (1)
MasterMuted_h1123568820	REG_DWORD	0x00000001 (1)
MasterMutedCheck_h3466142866	REG_DWORD	0x00000001 (1)
MeleeKey_h474149046	REG_BINARY	56 00
MenuKey_h1453540961	REG_BINARY	45 73 63 61 70 65 00
Movement_h1609263334	REG_DWORD	0x00000000 (0)
onContinue_h3084976549	REG_DWORD	0x00000000 (0)
PassedFactoryPuzzle_h871469069	REG_DWORD	0x00000000 (0)
SceneLoadChange_h2123827611	REG_DWORD	0x00000000 (0)
SceneLoadLevelManager_h3537199928	REG_DWORD	0x00000000 (0)
SettingsOpen_h1797398644	REG_DWORD	0x00000000 (0)
ShootKey_h217657725	REG_BINARY	43 00
SmallMapKey_h433689617	REG_BINARY	54 61 62 00
ukkoPosX_h2593995947	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
ukkoPosY_h2593995946	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
unity.cloud_userid_h2665564582	REG_BINARY	31 33 63 33 34 39 32 64 64 33 31 61 36 36 32 34 ...
unity.player_session_background_time_h123860221	REG_BINARY	31 35 32 38 32 38 39 37 38 33 34 32 38 00
unity.player_session_elapsed_time_h192694777	REG_BINARY	32 38 31 31 35 30 35 00
unity.player_sessionid_h1351336811	REG_BINARY	33 39 36 31 31 33 37 37 33 30 32 31 34 33 33 32 ...
UnityGraphicsQuality_h1669003810	REG_DWORD	0x00000000 (0)
volumeArvo_h2064554855	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)
VolumePercent_h590789558	REG_DWORD	(virheellinen DWORD (32-bittinen) -arvo)

Kuva 5. PlayerPrefs-metodilla tallennetut tiedot tietokoneen Rekisterieditorissa. Rekisterieditorin kautta pelaaja voi muuttaa tallennettavia arvoja haluamakseen.

Playerprefs on mainio tallennuskeino siitä, että se on helppo ottaa käyttöön projektiin. Playerprefs on myös yleisimpiin tarpeisiin monikäyttöinen, playerprefsillä voi tallentaa yleisimmät tarvittavat tiedot, vaikkakin tiedon serialisaatio on monipuolisempi vaihtoehto. Varsinkin jos tietoa tallentaa paljon kerralla, tiedon serialisaatio on tehokkaampi käyttömetodi. Playerprefsin negatiivinen sekä samalla positiivinen puoli on siinä, että playerprefs tallentaa tietonsa käyttäjän tietokoneelle (kuva 5). Omalta tietokoneelta peliä pelaava voi löytää tallennetut arvot ja muokata niitä haluamakseen. Jos pelissä on tallennettu

playerprefs-metodilla esimerkiksi boolean-arvoja tarkastaakseen, onko pelaaja aukaissut uuden alueen käyttäen playerprefsin kokonaislukutallennuksen 0- ja 1- arvoja, voi pelaaja suoraan playerprefs-tiedoista muuttaa arvot. Arvojen muutoksen seurauksena pystyy pelaaja esimerkiksi aukaisemaan pelissä lukitun alueen. Pelin tekijän ollessa huolimaton playerprefs-metodia käyttäessä, voi playerprefs rikkoa koko suunnitellun pelikokemuksen. MindSeizessä playerprefs-tallennus on toteutettu sillä tavoin, että pelaaja pystyy muuttamaan ainoastaan arvoja, jotka vaikuttavat pelin ulkonäköön, äänenvoimakkuuteen sekä kirkkauteen. Pelaajalle ei jää mahdollisuutta muokata pelin etenemistä tallennettuja arvoja muuttamalla.

Verrattuna tiedon serialisaatioon, playerprefs-metodi ei millään tavoin suojaa tietosisältöä, jonka vaikutuksena tiedot altistuvat pelaajan nähtäväksi sekä mahdollistaa tallennetun tietosisällön muokkaamisen. Playerprefs-metodilla kannattaa tallentaa arvoja, jotka eivät riko peliä tietoja mahdollisesti muokatessa. Ongelmana playerprefs-metodissa on myös se, että pelaaja voi tietoja muokkaamalla antaa väärän arvon tehdylle avaimelle, jolloin pahimmassa tapauksessa peli ei voi toimia oikein. Positiivisena puolena playerprefsillä-metodilla on, jos esimerkiksi pelaaja ei pääse muuttamaan asetuksia pelissä ongelman kohdatessa, voi pelaaja siirtyä playerprefs-tallennuksen sijaintiin rekisterieditorissa ja muuttaa asetuksien arvot. Arvot muuttamalla suoraan rekisterieditorissa tallentuvat arvot myös peliin, jolloin seuraavalla kerralla pelin avatessaan uudet arvot tulevat käyttöön heti pelin käynnistyessään.

Playerprefs-metodilla voi tallentaa yhteen tehtyyn avaimeen kerralla vain yhden arvon. Tämän vuoksi playerprefs-metodin käyttö ei sovi suurten tietokokonaisuuksien, kuten esimerkiksi listojen tallennukseen. Playerprefs-metodin käytettävyys heikentyy suuren tietomäärän sisältävissä peliprojekteissa, joissa suuri määrä tietoa halutaan tallentaa tehokkaasti kerralla. Tallennettavaa tietoa on myös helpompi hallita, kun tallennettava tieto on yhdessä paketissa sen sijaan, että joutuisi etsimään eri tallennettuja avaimia sekä yksitellen tallentamaan ja lataamaan tietoa.

3.2 Serialisaatio

Peliprojektissa otin myös käyttöön tiedon serialisaatioon perustuvan tallennusmetodin. Unity-pelimoottorissa oleva tiedon serialisaatiometodi muuttaa halutun tiedon Unity-pelimoottorille väliaikaisesti tallennettavaan muotoon. Serialisoitu tieto voidaan halutessaan uudelleen muodostaa käytettäväksi tiedoksi. Unity-pelimoottorissa on jo valmiiksi serialisaatioliitännäinen, jonka avulla serialisaatiometodi voidaan ottaa käyttöön projektissa lisäämällä muutaman rivin koodia. Unity-pelimoottori käyttää tiedon serialisaatiota esimerkiksi Unity-pelimoottorin käyttöliittymässä näkyvässä inspector-ikkunassa sekä prefabeissa, jotka ovat pelaajan tekemiä valmiita objekteja. Prefab-objektit jakavat alkuperäisen käyttäjän tekemän prefab-objektin arvot, joita käyttäjä voi laittaa peliin sekä muokata monia samanlaisia prefabeja yhtäaikaisesti.

```
 ▯    ÿÿÿÿ▯
    ▯Assembly-CSharp▯▯
PlayerData-
   ▯pelaajasijaintiX▯pelaajasijaintiY▯pelaajasijaintiZ
NPCsijaintiX
NPCsijaintiY
NPCsijaintiZ
pelaajaHealth
pelaajaPower▯pelaajaMaxHealth▯pelaajaMaxPower
CheckPoint▯pelaajaNanobots▯pelaajaMaxNanobots▯NanoBotsPermitte
d
MinePermitted
DashPermitted A1 A2 A3 B1 B2 B3 B4 B5 B6 B7 B9▯B10▯B12▯B13▯B15
▯BOSS1     MINIBOSS1 S1▯XXX1▯XXX2▯XXX3▯XXX4 SceneNimi
GMDoorPoint
GMNextScene
GMLastScene
SadePaalla▯DestroyedForever
MinimapSave          ▯                            ▯▯▯▯ ▯▯




▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯ System.Collections.Generic.Li
st`1[[System.String, mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089]]
▯System.Collections.Generic.List`1[[System.String, mscorlib,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089]]
    õâ¿ÂÕ8 Â                        d   d   ▯▯   ▯SaveCheckPoi
nt1▯   ▯   ▯  ▯                    ▯▯    A1▯▯
LevelStart▯▯    A1
           ▯   ▯    ▯System.Collections.Generic.List`
1[[System.String, mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089]]
▯   ▯_items▯_size▯_version▯  ▯▯          ▯   ▯   ▯▯
           ▯       ▯   ▯
   ▯B7-Gate1
▯▯

```

Kuva 6. Serialisaatiometodilla tallennetut tiedot tekstitiedostossa

Serialisaatiometodi on huomattavasti monipuolisempi ratkaisu eri käytettävien datatyyppien määrässä verrattuna playerprefs-metodiin. Playerprefs-metodin tavoin serialisaatiometodi tallentaa tietonsa pelaajan tietokoneelle tekstitiedostoon (kuva 6). Serialisaatiometodilla voi tallentaa kaikki primitiiviset datatyypit,

kuten yleisimpinä käytetyt int, float, double, bool, string, sekä byte. Primitiivisten datatyyppien lisäksi serialisaatiometodilla voi tallentaa enumeraatioita, jotka muodostuvat ryhmitetyistä vakioarvoista, sekä Unityn omia datatyyppejä, kuten vector, rect, quaternion, color, gradient ja niin edelleen. Serialisaatiometodi mahdollistaa myös tietolistojen suoran tallennuksen. Ainoana vaatimuksena tiedon serialisaatiolle on se, että tallennettavan tiedon täytyy olla luokaltaan public eli avoin tai sisältää SerializeField-attribuutti. MindSeizessä tein tallennettavalle tiedolle oman avoimen luokan, joiden muuttujiin lisäsin haluttavan tallennettavan tiedon ja joista taas latasin halutun tiedon takaisin.

Peliprojektissa käytin tärkeän pelillisen tiedon tallentamisessa serialisaatiometodia. Esimerkkeinä serialisaatiometodilla tallennetuista tiedoista ovat hahmon sijainti, kerätyt tavarat ja niiden vaikutukset, hahmon käytettävien tavaroiden määrät, pelin edistymispykälät, hahmon huoneen ja viimeksi käytetyn tallennuspisteen tiedot, avatut taidot, ympäristön tiedot sekä kerätyt ja käytetyt taitopisteet. Kaikki serialisaatiometodilla tallentamani tiedot ovat sisältävät tietoa, jota ei haluta pelaajan muokattavaksi. Serialisaatiotallennus verrattuna playerprefs-tallennukseen on myös tiedon säilönnässä turvallisempi, vaikkakin serialisaatiotallennuksella tallennetut tiedot tallentuvat playerprefs-tallennuksen tavoin pelaajan tietokoneelle. Serialisaatiotallennuksella tietosisältö on sen sijaan vaikeasti muokattavissa. Serialisaatiotallennuksen tekstitiedostossa näkyy vain tallennettavien tietojen nimet ilman arvoja. Pelaajan on sen lisäksi vaikea hahmottaa miten tallennetulle tiedolle voi antaa arvoja, joka vaikuttaa serialisaatiomenetelmän tiedon salauksen luotettavuuteen.

Serialisaatiometodi on peliohjelmoijien keskuudessa yleinen tapa Unity-pelimoottorilla tallentaa tietoa luotettavasti yhdessä JSON-tallennuksen kanssa. Serialisaatiomenetelmällä tallennus soveltuu parhaiten käytettäväksi suurten tietomäärien tallennuksessa, kuten esimerkiksi pelisession tallennuksessa. Jos tallennettavan tiedon määrä on pienempi eikä tallennettava tieto ole salassapidon kannalta tärkeää, playerprefs-tallennusmetodi on tehokkuudeltaan parempi vaihtoehto. Serialisaatiometodi vaatii aina tiedon väliaikaista muokkausta, joka on huomattavasti hitaampi prosessina verrattuna playerprefs-metodilla tallennukseen. Tarvittaessa tiedon voi tallentaa playerprefs-metodilla väliaikaisesti, jonka jälkeen pelin tallentaessa playerprefs-metodin sisältämät tiedot tallennetaan serialisaatiotallennuksella suojatumpaan muotoon. Tämä

menetelmä vaikuttaa pelin tallennusprosessin nopeuteen, mutta voi mahdollisesti aiheuttaa suojatun tallennetun tiedon pääsyä muokattavaksi käyttäjän toimesta.

3.3 DontDestroyOnLoad

Kolmas peliprojektissa käyttämäni tallennusmenetelmä on dontdestroyonload-metodi. Verrattuna tiedon serialisaatioon sekä playerprefs-metodiin, dontdestroyonload-metodi eroaa tallennustyyliltään siten, ettei dontdestroyonload-metodi tallenna tietoa tietokoneelle. Sen sijaan dontdestroyonload-metodi vaikuttaa pelin sceneissä oleviin objekteihin. Scenessä olevat objektit normaalisti pysyvät sen scenen sisällä, johon objekti on asetettu. Sceneä vaihtaessa aiemman scenen objektit poistetaan sisältämineen tietoineen, ja uuden scenen objektit lisätään aiempien tilalle. Dontdestroyonload-metodin avulla sceneä vaihtaessa halutut objektit eivät poistu, vaan sen sijaan siirtyvät uuteen sceneen säilyttäen objektien sisältämät tiedot aiemmasta scenestä. MindSeize-pelissä lisäsin objekteja sceneihin eri käyttötarkoituksilla. Esimerkiksi GameManager-objekti sisältää scenejen välillä siirtymisen sekä pelin väliaikaisen pysäyttämisen, kun taas SoundManager-objekti hallitsee eri ääniä sisältävien objektien äänenvoimakkuutta ja tilaa. Tämän kaltaiset yleisesti käytössä pidettävät objektit usein halutaan pysyvän mukana käytettävissä jokaisessa pelin scenessä, jolloin DontDestroyOnLoad-funktio on käytännöllinen tiedonhallintakeino.

Ilman dontdestroyonload-metodia, SoundManagerin kaltaiset objektit pitäisi kopioida ja luoda jokaiseen sceneen uudelleen kopiona. Siitä aiheutuisi pelin sisältämän ylimääräisen tietosisällön lisääntyminen, varsinkin jos pelin sisältämiä scenejä on monia, jolloin kopioitakin täytyy tehdä useita. Sen lisäksi joka kerta scenen vaihtaessa SoundManagerin kaltaisten objektien sisältämät arvot tulee päivittää aina uuden scenen ladatessaan. Tiedon poistuminen aiheuttaa tiedon turhaa päivittämistä vaikuttaen lataus- sekä väliprosessien suorituskeston pitkittymiseen, joka lisää tietokoneelta vaadittua työtä. Jos objektien tietoja ei ole tallennettu, objektit sisältävät uuteen sceneen siirtyessä ainoastaan oletusarvot, jotka objekteille annetaan peliä rakentaessa.

Dontdestroyonload-metodi on käytännöllinen esimerkiksi väliaikaisen tiedon säilyttämisessä. Dontdestroyonloadilla voi säilyttää esimerkiksi objektin, joka hallitsee ja kerää pelin tärkeää tietoa. Säilytettyä objektia voi myöhemmin käyttää tiedon tallentamisessa serialisaatiometodin avulla. Sen sijaan, että tietoa aina tallennettaisiin hidastaen pelisessiota, objekti voi dontdestroyonload-metodilla kerätä väliaikatietoja, jotka halutaan tietyssä vaiheessa peliä tallentaa, esimerkiksi pelisession lopettaessa. Tällöin pelin ei tarvitse käyttää resurssejaan jatkuvaan tiedon tallentamiseen silti samalla ylläpitäen tärkeää tietoa käytettävänään.

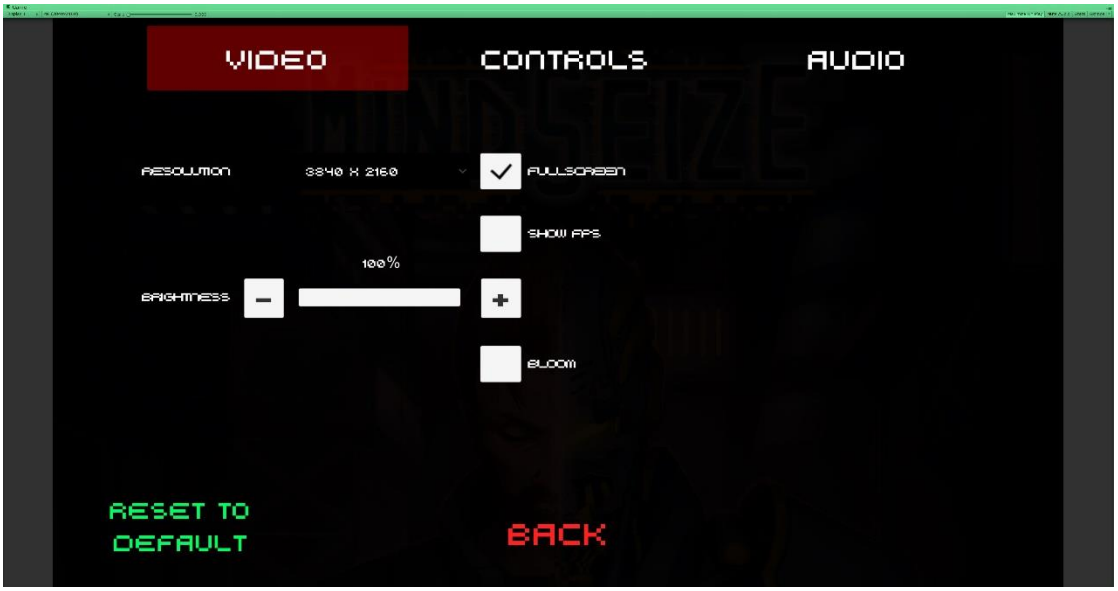
Dontdestroyonload-metodi ei siis suoranaisesti tallenna tietoa vaan sen sijaan hallitsee tietoa suotuisasti peliprojektissa. Dontdestroyonload-metodin avulla voi vähentää tietoa sisältäviä kopioita peliprojektissa, jotka lisäisivät peliprojektin kokoa ja vaadittua tehoa. Samanaikaisesti dontdestroyonload-metodi vähentäisi arvojen päivitysprosessien määrää. Dontdestroyonload yhdistettynä tiedon serialisaatiometodiin mahdollistaa tehokkaan tiedon hallinta- sekä tallennusjärjestelmän projektissa hyväksikäyttämällä jokaisen tallennusmenetelmän positiivisia puolia. MindSeizessä yhdistin serialisaatio-, dontdestroyonload-, sekä playerprefs-metodit muodostaakseni monipuolisen tiedonhallintajärjestelmän peliin. Tallennusmetodien yhdistämisen avulla tiedonhallintajärjestelmä toimii tehokkaasti jokaisessa pelin vaiheessa ongelmitta sekä joustavasti. Joustavuuden järjestelmään tuo playerprefs-metodi, joka tiedon puuttuessa valitsee oletusarvot käytettäviksi arvoiksi. Samalla playerprefs-metodin avulla tallennetaan tiedoiksi oletusarvot, jonka jälkeen pelaaja voi halutessaan muuttaa arvoja halutukseen uudelleen. Ilman playerprefs-metodia sekä sen HasKey-funktiota, esimerkiksi arvotta olevat asetukset oletuksena olisivat poissa käytöstä, jos asetuksen arvo ei olisi löydettävissä.

4 MINDSEIZE

Opinnäytetyön toimeksiantona tein MindSeizeen monia pelillisiä sekä peliä tukevia taustajärjestelmiä. Taustajärjestelmät sisältävät, käyttävät sekä hallitsevat tietoa, joiden merkitys pelattavuudelle sekä pelin etenemiselle on suuri. Seuraavaksi esittelen tekemäni työni osissa.

4.1 MindSeizen asetukset

Tietosisällöltään suurin taustajärjestelmistä on peliasetuksia hallitseva järjestelmä, jonka avulla voi muokata sekä asettaa järjestelmän sisältämiä asetuksia haluamakseen. Pelin asetukset tallentuvat playerprefs-metodilla pelaajan tietokoneelle. Jaoin pelin muokattavat asetukset kolmeen ryhmään, joille tein omat ikkunat asetukset-scenen sisälle.



Kuva 7. Videoasetusten ikkuna MindSeizen asetuksissa

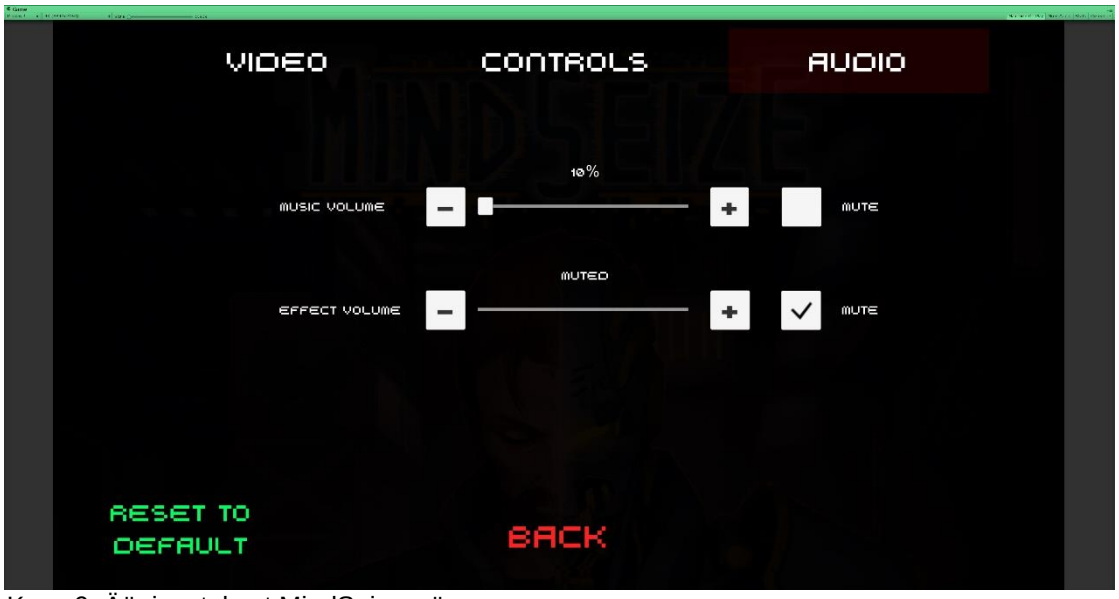
Ensimmäinen ryhmä, video, sisältää näkyvyyteen sekä tarkkuuteen liittyviä asetuksia (kuva 7). Videoasetuksista voi valita resoluution, säätää kirkkautta, muuttaa pelin näkymään koko näytön tilassa tai ikkunana, asettaa FPS-laskurin (frames per second) näkyviin sekä laittaa graafiseen näkyvyyteen vaikuttavan bloom-asetuksen päältä tai pois. Peliasetuksia hallitseva järjestelmä sisältää varmistavan järjestelmän, joka tarkastaa pelaajan muutokset sekä tarjoaa mahdollisuuden muutetun arvon palautukseen. Esimerkiksi resoluutiota muuttaessa pelaajalta kysytään, haluaako hän pitää valitun resoluution vai palauttaa resoluution aiempaan arvoonsa. Jos pelaaja on painamatta näppäintä tai

peruuttaa tapahtuman, peli palautuu aiemmin käytettyyn resoluutioon. Muutoin uusi resoluutio otetaan käyttöön, jolloin resoluutiolista päivittää tietonsa.

Kuva 8. Kontrolliasetukset MindSeizessä. Näppäinarvon puuttuminen esitetään pelaajalle punaisella värillä sekä ilmoituksella asetuksista lähtiessä.

Toinen ryhmä, controls eli kontrollit, sisältää hahmolle määritettävät näppäimet pelaamista varten (kuva 8). Pelaaja voi halutessaan muuttaa käytettävät näppäimet haluamakseen valitsemalla muutettavan näppäimen ja tarkastusikkunan ilmestyessä painamalla haluamaansa näppäintä, jolloin valittu näppäin tulee käyttöön aiemmin valitun näppäimen tilalle. Jos esimerkiksi pelaaja valitsee hyppynäppäimeksi näppäimen, joka on jo käytössä menunäppäimenä, menunäppäimelle määritetty näppäin poistuu ja siirtyy hyppynäppäimen uudeksi arvoksi.

Kontrolliasetuksissa järjestelmä tarkastaa aina onko jokaiselle vaaditulle näppäimelle annettu näppäinarvo. Vaaditun näppäimen arvon ollessa tyhjänä, näppäimen nimi muuttuu punaiseksi, jolloin pelaajalta vaaditaan annettavaksi näppäimelle arvo päästäkseen pelisceneen.

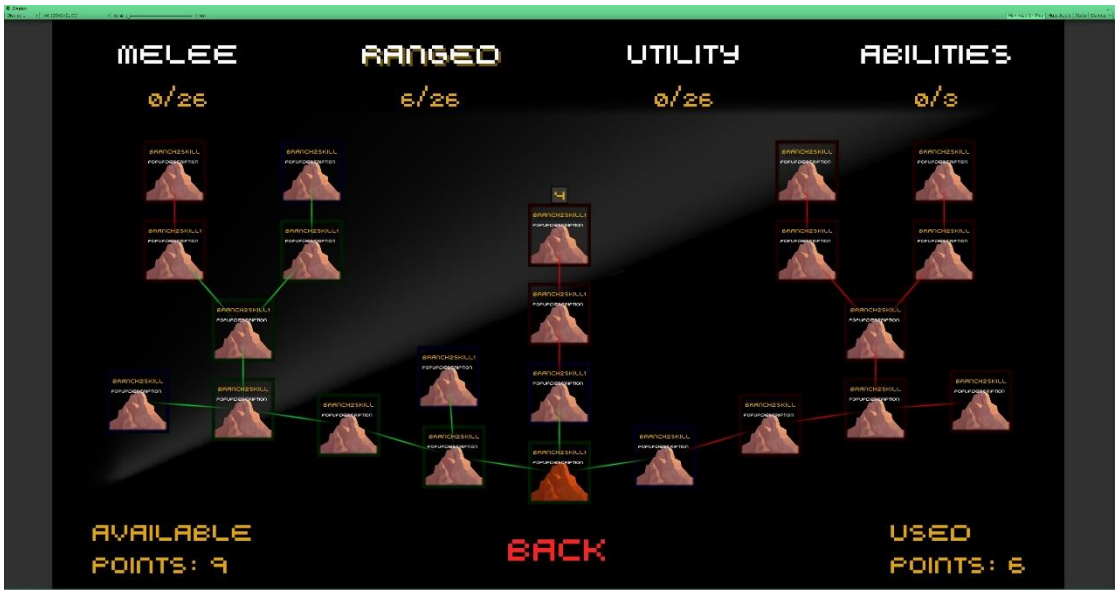


Kuva 9. Ääniasetukset MindSeizessä

Kolmas ryhmä, audio, sisältää kaksi liukuvaa palkkia, jotka määrittävät pelin musiikin äänenvoimakkuuden sekä pelin sisältämien efektien äänenvoimakkuuden (kuva 9). Molemmat äänenvoimakkuudet voidaan myös hiljentää kokonaan mute-painikkeella. Ääniasetukset arvoineen tallentuvat muiden asetusten mukaisesti tiedostoon, jolloin pelin käynnistyessään äänenvoimakkuudet ovat aiemmin tallennetuissa arvoissaan. Asetukset-sceneen siirtyessä kaikki nähtävät arvot päivitetään näkyviksi hakemalla asetetut arvot tallennetuista tiedoista.

4.2 Taitopuujärjestelmä

Pelillisesti merkittävin tekemäni taustajärjestelmä on hahmon taitopuujärjestelmä (skill tree). Taitopuujärjestelmässä hahmon taitoja voi kehittää pelin edetessä saatavilla taitopisteillä (skill point). Taitopisteet käytetään taitopuun järjestelmän scenessä. Taitopuun scene rakentuu useista napeista, joille on ohjelmoitu avaamisjärjestys. Pelaajan vahvistaessa tietyn taidon avauksen, taidon avaus tallennetaan taitopuujärjestelmän omaan serialisaatiotallennukseen. Kun taito on avattu, voi samaa taitolinjaa kehittää eteenpäin. Taitojen avaamisjärjestys luo puumaisen kokonaisuuden, joka sisältää myös risteyksiä, joissa pelaajan täytyy valita itselleen jompikumpi reitti.



Kuva 10. MindSeizen taitopuujärjestelmä Unityn editorissa

Aluksi taitopuujärjestelmää varten tein taitonapin, joka sisältää mahdollisuuden avata kyseisen napin sisältämän taidon napin sisältämää pistemäärää vastaan. Yhden napin tehtyäni muodostin napista prefab-objektin, jota pystyy kopioimaan moniksi samanlaisiksi objekteiksi. Tämän jälkeen lisäsin napeille avausjärjestyksen boolean-arvoja käyttämällä. Pelaajan täytyy avata ensimmäinen avattava taito, jonka jälkeen seuraavien näppäimien boolean-arvot päivittyvät muuttaen seuraavat taidot avattaviksi. Boolean-arvojen avulla rakensin komponentin, jota lisäämällä haluttaviin objekteihin muodostuu taitopuujärjestelmä (kuva 10).

Taitopuujärjestelmän pohjan valmistuessa loin järjestelmään mukaan taitopuuikkunoita. Muut taitopuuikkunat sisältävät duplikaatteja ensimmäisestä taitopuusta eri arvoilla sekä taidoilla. Taitopuiden rakenteiden ollessa toiminnallisia, lisäsin taitopuujärjestelmään taitopisteet sekä tarkastusjärjestelmän. Taitopisteet ylläpitävät käytettävien sekä käytettyjen pisteiden määrää. Tarkastusjärjestelmän toiminta pohjautuu näppäinkomponentin tietoihin sekä taitopisteisiin. Esimerkiksi taitopisteiden ollessa liian vähäiset vaaditun taidon avaamiseen, tarkastusjärjestelmä peruuttaa taidon avausprosessin sekä ilmoittaa ongelmasta pelaajalle. Aivan lopuksi rakensin näppäinkomponenttiin toiminnon, joka luo viivan näppäimien välille, jonka avulla pelaaja hahmottaa taitopuun kehitysjärjestystä. Viivat toimivat yhdessä tarkastusjärjestelmän kanssa värjäten viivat sen mukaan, onko näppäintä seuraava taito avattavissa vai ei.

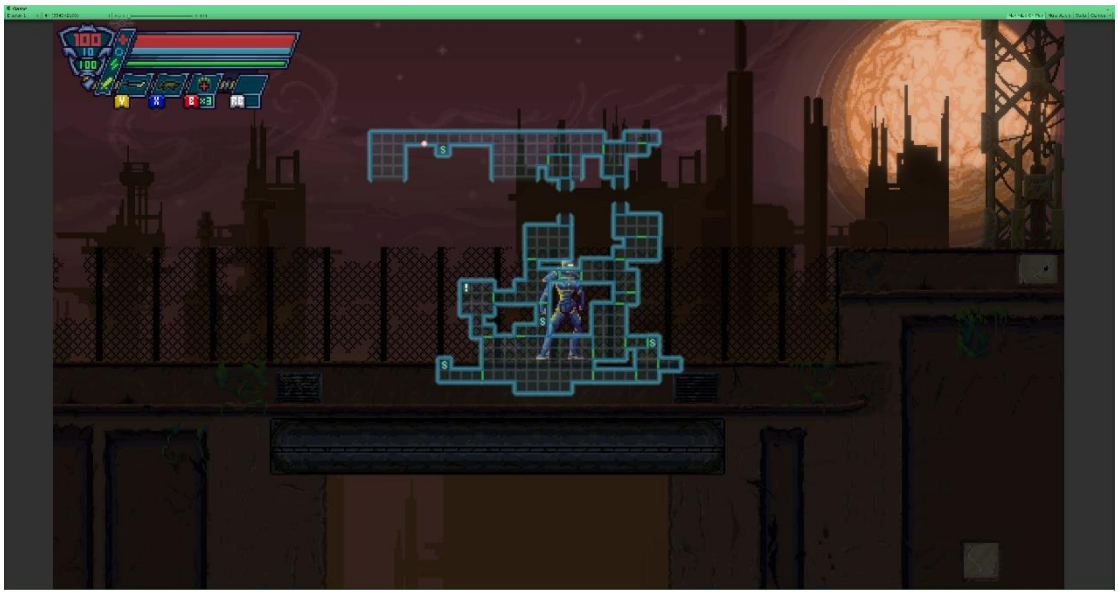
Taitopuujärjestelmän merkitys pelille on huomattava. Se antaa pelaajalle joustavan muokkausympäristön kehittääkseen hahmoa pelaajan haluamalla sekä muista pelaajista poikkeavalla tavalla, jolloin pelityyli sekä pelin eteneminen myöskin poikkeavat muiden pelaajien omista kokemuksista. Monipuolinen pelin personalisointi vahvistaa peliin kehittyvää immersiota antamalla pelaajan vaikuttaa ja tehdä itse valintoja perustuen hahmon toimintoihin. Samasta syystä pelin uudelleen pelaamisen halu kasvaa, samalla lisäten pelin pelattavuutta, jolloin viihdyttävyyskynnys nousee. Taitopuusta voi myös nähdä tulevia avattavia taitoja, joista saatua tietoa pelaaja voi suunnitelmallisesti hyödyntää omassa pelisessiossaan. Usein taitopuujärjestelmissä avattavia taitoja on enemmän kuin pisteitä, joilla taitoja avataan. Tästä syystä pelaajan on mietittävä mitä taitoja haluaa käyttöönsä ja mitkä taidoista on jätettävä avaamatta. Tekemäni taitopuun avattavia reittejä on selkeästi hahmotettu näkyvillä värjätyillä viivoilla. Taitopuu havainnollistaa myös pelaajan tämän hetkiset avatut taidot sekä seuraavat mahdolliset avattavat taidot.

Monet pelit käyttävät hahmon muokkausta ja kehittämistä peleissään myös siitä syystä, että pelaajat luovat omia peliryhmiään ja sivustoja, joissa pelaajat voivat kertoa muille oman pelityylinsä vahvuuksista ja toiminnoista. Peliryhmät kehittävät myös vuoropuhelua muiden pelaajien kanssa, joka kasvattaa yhteisöjen aktiivisuutta ja sitä kautta myös pelin suosiota.

4.3 Kartta- ja navigaatiojärjestelmä

Kolmas merkittävä pelin informatiivinen järjestelmä on kartta- ja navigaatiojärjestelmä, joka toimii yhteistyössä pelin tiedonhallintajärjestelmän kanssa. Karttajärjestelmä koostuu monista spriteistä eli kuvista, jotka on yhdistetty palapelin tavoin kartaksi. Tiedonhallintajärjestelmä tallentaa järjestelmäänsä jokaisen scenen, jossa pelaaja on käynyt. Pelaajalle näytetään jokaisen käydyn scenen kuvat karttakokonaisuudessa, kun taas pelaajan käymättömät scenet ovat karttakokonaisuudessa näkymättömiä. Aina scenen vaihtuessa tiedonhallintajärjestelmä tarkistaa aktiivisen scenen tiedot sekä tallennustiedot pelaajan käydyistä sceneistä, samalla päivittäen tallennustiedot tiedon muuttuessa. Aktiivisessa scenen kartassa näytetään hahmon sijainti, joka on määritetty järjes-

telmässä. Kartta- ja navigaatiojärjestelmä etsii hahmon prosentuaalisen sijainnin x- ja y-koordinaatistossa ja siirtää karttakokonaisuudessa hahmoa esittävän pisteen samaan prosentuaaliseen sijaintiin aktiivista sceneä mallintavan kuvan alueeseen rajattuna. Karttakokonaisuuden kuvat ollen samassa mittasuhteessa scenen alueiden kanssa, lopputuloksena hahmoa esittävän kuvan sijainti kartassa on realistinen suhteessa scenessä liikkuvan hahmon sijaintiin.

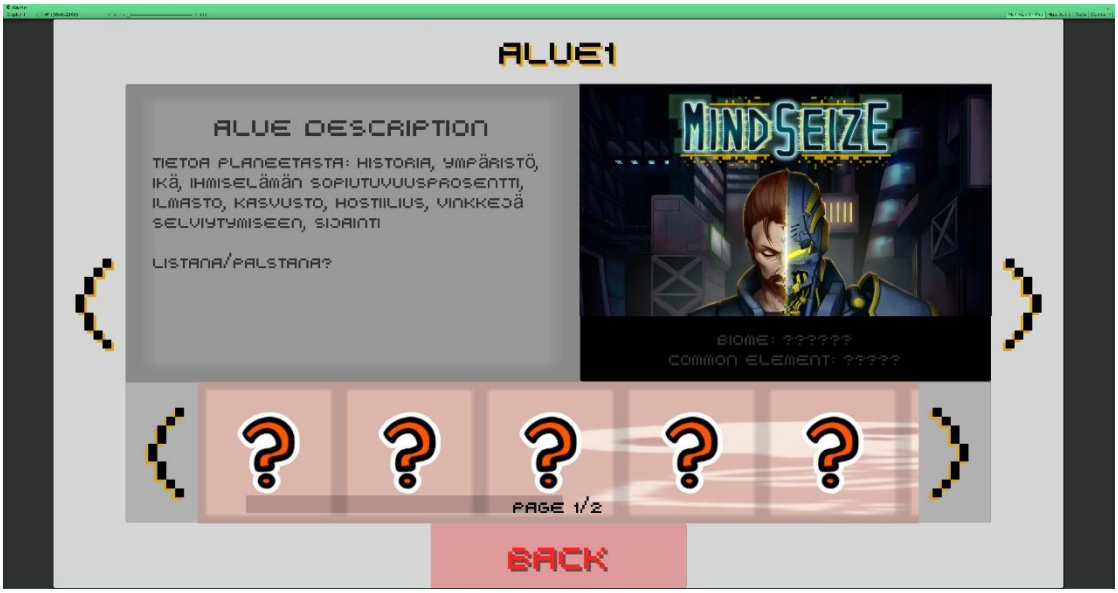


Kuva 11. MindSeizen kartta- ja navigaatiojärjestelmä pienoisversiossa (mini map). Hahmon sijainti näkyy kartassa punaisena pisteenä.

Kartta- ja navigaatiojärjestelmä auttaa pelaajaa hahmottamaan nykyisen sijaintinsa sekä sen, missä reitit seuraaviin pelin alueisiin sijaitsevat. Kartta auttaa pelaajaa myös hahmottamaan missä on hahmon kannalta vaara tai minne ei ole mahdollisuutta edetä. Ilman kartta- ja navigaatiojärjestelmää pelaajan ymmärtäminen etenemisestä on heikentynyt, sen lisäksi mahdollinen määrätiedottomuus voi vaikuttaa alentavasti viihdyttävyyteen. Pelaajan on myös mahdollista avata pienoisversio kartasta näytölle peliä pelatessa (kuva 11). Pienoiskartta esittää pelaajan sijainnin pelissä jatkuvasti, mahdollistaen tarkan sijainnin paikannuksen.

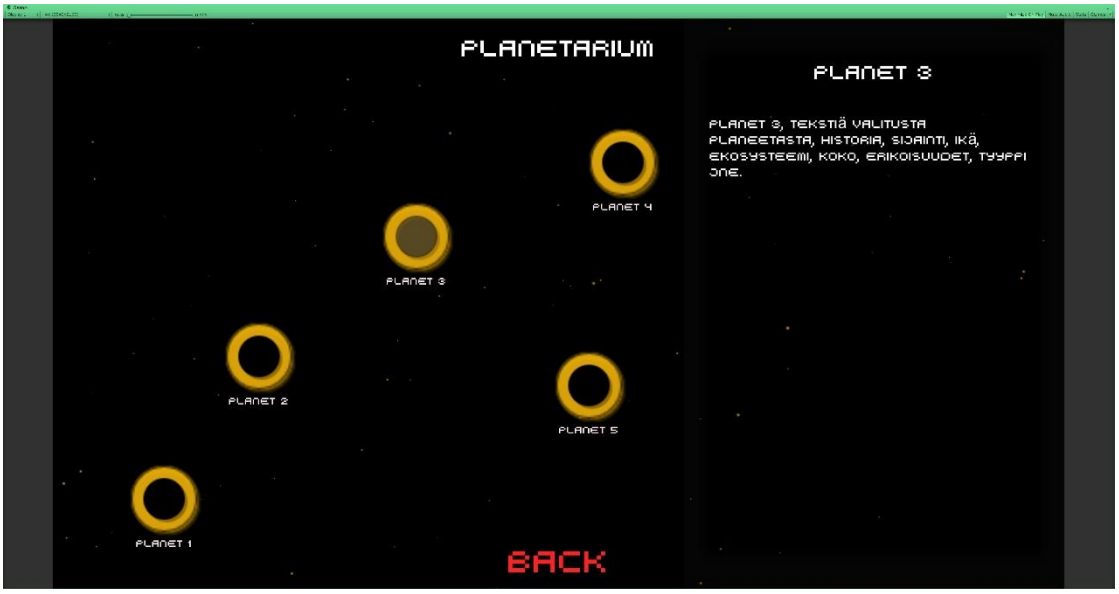
4.4 Tietoluettelojärjestelmä

Loin MindSeizeen myös tietoluettelojärjestelmän, joka sisältää tietoa sekä neuvoja pelin alueista sekä hahmoista. Aloitin tietoluettelon luonnin tekemällä luettelopohjan yhdelle pelin alueista. Luettelo sisältää näytön kokoisen sivun jokaisesta pelin alueesta.



Kuva 12. MindSeizen tietoluettelojärjestelmän pohja toiminnallinen pohja

Pelin alueita tietoluettelojärjestelmässä voi vaihtaa näytön sivuissa olevilla napeilla (kuva 12). Suunnittelin tietoluettelojärjestelmän toimimaan miellyttävästi näppäimistöllä, hiirellä sekä peliohjaimella. Monilla eri oheislaitteilla toimiakseen jouduin suunnittelemaan käytettävästä oheislaitteistosta riippuen tietoluetteloon hieman eri käyttötoiminnot sopimaan parhaiten käytettävää oheislaitteistoa varten. Tein jokaisen alueen tietoluetteloon oman liukuvan listan, joka sisältää listan alueen hahmoja varten. Valittua hahmoa painaessaan avautuu uusi ikkuna hahmon tiedoista. Tein liukuvan listan käytettäväksi peliohjaimilla, hiirellä sekä näppäimistöllä. Esimerkiksi listaa voi vetää hiirellä tai vierittää listan molemmilla sivuilla olevia nappeja käyttämällä.

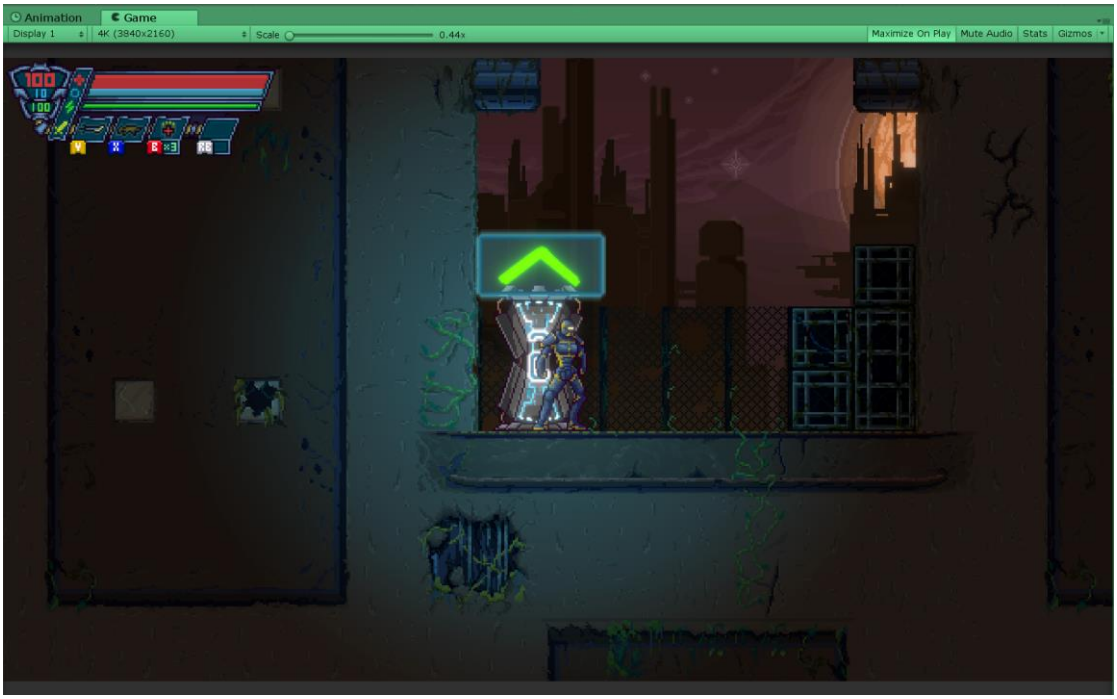


Kuva 13. MindSeizen Planetarium-tietoluetteloscene

Tein myös toisen tietoluettelojärjestelmän, planetariumin, ensimmäisen tietoluettelojärjestelmän tavoin samoilla periaatteilla toimien. Planetarium esittää avaruutta ja pelin viittä planeettaa, joita valitsemalla voi nähdä pelaajan sijainnin sekä taustatarinaa alueille (kuva 13). Planetariumille tein taustalle partikkeliefektin esittämään avaruudessa näkyviä tähtiä. Planetarium sisältää myös tekemäni viivakomponentin, joka piirtää viivan kahden tai useamman objektin välille hahmottaakseen navigointireittiä näppäimistöllä tai ohjaimella. Viivakomponentin voi halutessaan asettaa päälle tai sulkea, riippuen lopullisesta tarpeesta.

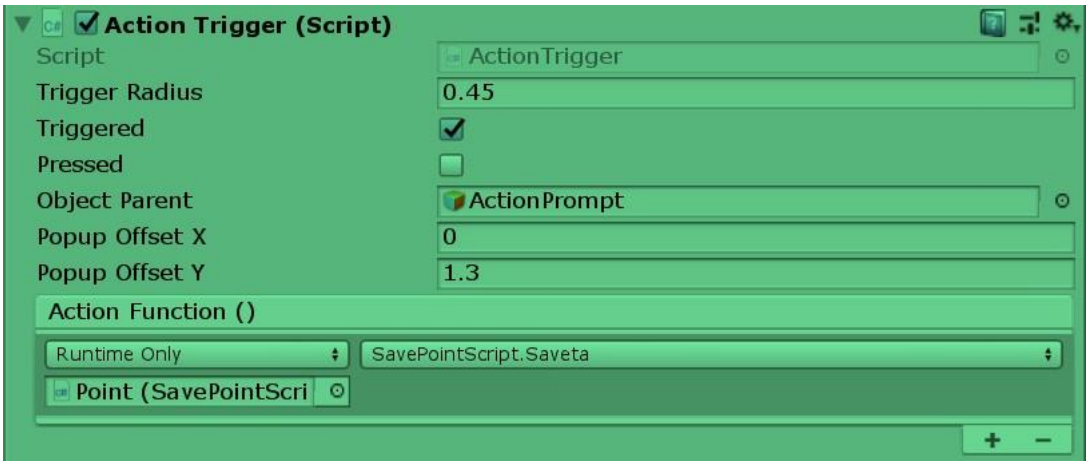
4.5 Interaktiojärjestelmä

Alun perin toimeksiantoon kuului myös tehdä keskustelujärjestelmä. Toimeksianto muuttui keskustelujärjestelmästä niin, että keskustelujärjestelmän sijaan tein interaktiojärjestelmän. Interaktiojärjestelmän kautta voi esimerkiksi käynnistää keskustelun tai siirtyä seuraavalle alueelle tai kutsua hissin (kuva 14). Kamina Dimensionilla oli itsellään jo käytössä vaadittu työkalu keskustelujärjestelmää varten, muttei mitään järjestelmää aktivoivaa toimintoa, joka vaadittiin peliin.



Kuva 14. Esimerkki interaktiojärjestelmän toiminnasta pelin tallennuspisteellä. Pelissä tapahtumia varten luotu interaktiojärjestelmä näyttää pelaajalle, mitä näppäintä pelaajan täytyy painaa tapahtuman aktivoidakseen (ylänuoli).

Interaktiojärjestelmän rakensin lisättäväksi vuorovaikutettavaan hahmoon tai objektiin. Interaktiokomponentin sisältö aktivoituu pelaajan siirtyessä tarpeeksi lähelle interaktiokomponentin sisältämää objektia. Pelaajan lähestyessä vuorovaikutettavaa objektia ilmestyy siihen merkki, joka osoittaa mitä näppäintä pelaajan täytyy painaa käynnistääkseen interaktiokomponenttiin lisätyn toiminnon.



Kuva 15. Interaktioita hallitseva Action Trigger-komponentti

Interaktiokomponentin sisälle voi vetää halutun toiminnon sisältävän funktion, jonka interaktiojärjestelmä käynnistää vaatimusten täyttyessä. Interaktiojärjestelmä toimii Action Trigger-komponentilla (kuva 15). Action Trigger-komponentti vaatii pelaajaa pysymään komponentissa määritetyn alueen sisällä boo-

lean-arvot muuttaakseen komponentissa asetetun funktion aktivoimista varten. Action Trigger-komponentista voi myös muuttaa määritetyn toiminta-alueen kokoa sekä ilmestyvän ilmoituksen sijaintia. Ilmoituksen sijaintia voi muuttaa tarkasti käyttämällä Action Trigger-komponentin popup offset x- ja y-arvoja.

5 PÄÄTÄNTÖ

Tavoitteena opinnäytetyölläni oli toteuttaa toimiva monipuolinen menu- ja taustajärjestelmäkokonaisuus Kamina Dimensionin MindSeize-peliin. Taustajärjestelmien tuli toimia sujuvasti näppäimistöllä sekä peliohjaimella ottaen huomioon pelaajan käyttökokemuksen tärkeyden järjestelmien suunnitteluvaiheessa. Lähdin tekemään monia opinnäytetyöni toimeksiannon antamia järjestelmiä ilman aiempaa kokemusta kyseisistä järjestelmistä. Pohjan opilleni sain ammattikorkeakoulun peliohjelmointikursseilta, itsenäisestä peliohjelmoinnin opiskelusta sekä harjoitteluvaiheesta. Suoritin harjoitteluvaiheen myös peliohjelmoinnin parissa. Olin siis jo tutustunut Unity-pelimoottoriin sekä sen pääpiirteisiin ennen opinnäytetyön aloittamista, siitä huolimatta opinnäytetyön aikana minulle avartui monia uusia toimintoja Unity-pelimoottorista sekä parempia käyttötapoja liittyen pelimoottorin käyttöliittymään.

Tekemäni järjestelmäkokonaisuus täytti annetun toimeksiannon täysin. Opinnäytetyöprosessi oli todella kehittävä kokemus. Opin paljon peliohjelmoinnista, pelillisten järjestelmien suunnittelusta, peliyhtiössä työskentelystä sekä pelin kehityksen rakenteesta ja kehityksen vaiheista. Sain Kamina Dimensionilta avoimet kädet toteuttaakseni järjestelmät omalla tavallani annettujen toimeksiannon vaatimusten täyttyessä. Avoin työ- ja kehitysympäristö kannusti minua kokeilemaan monia eri vaihtoehtoja vastaantuleviin ongelmiin sekä auttoi minua kehittämään lopputuloksesta hyvin toimivan kokonaisuuden.

Suurin haaste projektissa oli jokaisen tekemäni järjestelmän yhdistäminen osaksi toimivaa kokonaisuutta. Alun perin järjestelmiä tehdessäni en ottanut huomioon järjestelmien keskenäistä vuorovaikutusta ja siihen liittyvää yhteensopivuuden tarvetta. Myöhemmin ongelman kohdatessani minun täytyi muuttaa tekemieni järjestelmien rakennetta, jotta järjestelmä toimisi kokonaisuutena suorituskykyisesti ja ilman ongelmia. MindSeizeen tekemäni järjestelmä

tulee käyttöön MindSeizen demoversioon, joten järjestelmään tullaan pelin kehittyessä lisäämään toimintoja sekä muuttamaan ulkonäköä. Ulkonäön toteutusta peliin ei vaadittu toimeksiannossa. Taustajärjestelmiin on helppo lisätä uusia toimintoja tai sisältöä, joka helpottaa pelin tulevaa kehitystyötä.

LÄHTEET

Hamilton, K. 2015. The Ten Commandments Of Video Game Menus. WWW-dokumentti. Päivitetty 29.5.2015. Saatavissa: https://kotaku.com/5955855/the-ten-commandments-of-video-game-menus [viitattu 25.9.2018].

Madigan, J. 2010. The Psychology of Immersion in Video Games. WWW-dokumentti. Päivitetty 27.7.2010. Saatavissa: http://www.psychologyofgames.com/2010/07/the-psychology-of-immersion-in-video-games/ [viitattu 27.9.2018].

Baker, B. 2017. Video game menu design – My thoughts to best deliver features to your gamers. Blogi. Päivitetty 31.3.2017. Saatavissa: https://turntosideb.com/blog/2017/3/31/video-game-menu-design-my-thoughts-to-best-deliver-features-to-your-gamers [viitattu 28.9.2018].

Lovato, N. 2015. Why Habit Formation Is The Key To Long Term Retention. Blogi. Päivitetty 14.4.2015. Saatavissa: https://gameanalytics.com/blog/habit-formation-key-long-term-retention.html [viitattu 2.10.2018].

Uccello, A. 2017. How to Save and Load a Game in Unity. WWW-dokumentti. Päivitetty 1.9.2017. Saatavissa: https://www.raywenderlich.com/418-how-to-save-and-load-a-game-in-unity [viitattu 16.10.2018].

Unity Technologies s.a. PlayerPrefs. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html [viitattu 17.10.2018].

Unity Technologies s.a. Script Serialization. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/Manual/script-Serialization.html [viitattu 18.10.2018].

Unity Technologies s.a. Object.DontDestroyOnLoad. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html [viitattu 19.10.2018].