TAMPEREEN
AMMATTIKORKEAKOULU

UNITY 5 PELITUOTANTO

Pelidemon tuottaminen Windows-alustalle

Kim Milán

Opinnäytetyö
Huhtikuu 2017
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Kim Milán:
Unity 5 pelituotanto
Pelidemon tuottaminen Windows-alustalle

Opinnäytetyö 27 sivua, joista liitteitä 0 sivua
Huhtikuu 2017

Opinnäytetyön tarkoituksena oli käsitellä pelituotantoa ja pelien kehitystä Unity 5 -pelimoottorilla sekä tuottaa valmis pelidemo, jota testata käyttäjillä. Työn tavoitteena oli tutustua pelien tekemiseen, kehittää saatujen tietojen perusteella pelidemo ja analysoida demosta saatavia palautteita mahdollisen lopullisen pelin tuotannossa. Opinnäytetyön idea ja aihe olivat peräisin omasta kiinnostuksesta peleihin ja niiden kehitykseen sekä aiemmasta harrastuneisuudesta kyseisellä alalla.

Työssä käydään läpi pelien kehityksen teoriaa yleisesti, Unityn käyttöä ja ominaisuuksia sekä tuotetun demon eri osia ja ominaisuuksia. Tuotettu pelidemo itsessään on ensimmäisen persoonan näkökulmasta pelattava ongelmanratkaisupeli. Pelille ei asetettu tiettyä kohderyhmää vaan se julkaistiin useaan paikkaan internetissä anonyymien käyttäjien saataville. Käyttäjäpalautetta kerättiin itse tuotettuun tietokantaan koulun palvelimella sekä Unity Analytics-palveluun.

Työn tavoitteisiin päästiin ja demon tuottamista palautteista voidaan tehdä johtopäätöksiä jatkokehityksen suhteen, vaikka otanta olisi saanut olla suurempi saatujen keskiarvojen paremman luotettavuuden kannalta.

Asiasanat: unity, pelituotanto, ohjelmistokehitys, c#-ohjelmointi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information Technology
Software Engineering

Kim Milán:
Unity 5 game development
Developing a game demo for Windows

Bachelor's thesis 27 pages, appendices 0 pages
April 2017

The purpose of this thesis was to look into game development, developing games using the Unity 5 game engine and to develop a demo for testing. The goal was to learn more about development, develop a working demo based on that knowledge and analyze the feedback received from the demo. The idea for this thesis came from my own interest and experience with games and game development.

The thesis examines game development theory, the use and features of Unity and the different parts and features of the created demo. The game demo itself is a first-person puzzle game. The game had no specific target group and was released to multiple locations on the internet for any anonymous users. User feedback from the demo was collected using Unity Analytics and a separate database on the school's server.

The set goals were achieved and various conclusions about further development can be made from the collected feedback, even though the amount of it could have admittedly been higher for better credibility.

Key words: unity, game development, software development, c#-programming

SISÄLLYS

LYHENTEET JA TERMIT

DAW	Digital Audio Workstation, äänitykseen ja äänitiedostojen käsittelyyn tarkoitettu laite tai ohjelmisto
SQL	Structured Query Language, relaatiotietokantojen kanssa käytettävä kyselykieli
C#	Microsoftin .NET-konseptia varten kehitetty ohjelmointikieli
JSON	JavaScript Object Notation, avoimen standardin mukainen tiedostomuoto tiedonvälitykseen.
PHP	PHP: Hypertext Preprocessor, yleinen ohjelmointikieli
UnityScript	Javascriptin kaltainen ohjelmointikieli, jota käytetään C#-kielen ohella Unityssa.
REST	Representational State Transfer, ohjelmistorajapinnan arkkitehtuurimalli.
HUD	Heads Up Display, pelissä käyttäjän ruudullä näkyvä graafinen käyttöliittymä.

1 JOHDANTO

Tässä työssä käsitellään pelituotantoa Unity 5 -pelimoottorilla. Tuotannon kulun sekä käytettävien työkalujen lisäksi toteutetaan varsinainen pelidemo Windows-alustalle käyttäen Unityn omia, Unity Asset Storen ilmaisia sekä itse tehtyjä komponentteja ja elementtejä. Opinnäytetyön ajatus syntyi tekijän omasta mielenkiinnosta niin peliteollisuuteen alana kuin myös aiemmassa kokemuksesta mobiilipelin tuotannossa harjoittelun yhteydessä.

Työn tavoitteena on selvittää pelin luomisen eri osa-alueita sekä vaiheita, ja tuottaa lopullinen toimiva demo käyttäjien testattavaksi. Ylimääräisenä tavoitteena on kerätä ja analysoida käyttäjien arvioita sekä pelin yleistä käyttäjäkokemusta keräämällä anonyymisti palautetta Unity Analytics palvelun ja oman REST-rajapinnan kautta pelinsisäisellä palautelomakkeella.

Työn ensimmäisessä luvussa käsitellään lyhyesti pelituotantoa yleisesti kattaen vaiheet pelin suunnittelusta valmiin pelin tai testiversion julkaisuun ja testaamiseen. Luvussa ei kuitenkaan keskitytä erityisesti Unity-ympäristön tai tiettyjen alustojen tuotantoon, vaan asiat käsitellään pääosin yleispätevästi.

Toisessa luvussa käsitellään käytettyjä työkaluja ja kehitysympäristöjä, kuten Unity 5, Blender ja Visual Studio. Luvun tarkoitus on käsitellä tarkemmin Unityn toimintaa ja erityisominaisuuksia. Koska työn pääpaino on Unity 5 -pelimoottori, muiden käytettyjen työkalujen toimintaan perehdytään vain niiden käytöstä Unityn kanssa.

Työn jälkimmäisellä puoliskolla käsitellään varsinaisen tuotetun peliprojektin toteutusta sekä käyttäjäpalautetta. Käyttäjäpalautteen tarkoitus on antaa eväitä suuremman peliprojektin jatkokehitykseen, joten pelidemon onnistumista ja jatkokehitystä pohditaan erityisesti niiden pohjalta.

2 PELITUOTANNON VAIHEET

Nykyaikaisten pelien tuotanto on suuri ja usein kallis prosessi, joka koostuu useista osa-alueista ja vaiheista. Projektiryhmä koostuu useista eri rooleista, joilla kaikilla on omat vastuualueensa pelituotannon eri vaiheissa, kuten suunnittelussa, testauksessa ja julkaisussa. Joitain vuosikymmeniä sitten pelit toteutettiin pääasiassa koodaamalla pienen ryhmän voimin ja nykyaikaisten pelien tuotanto muistuttaa jo suurelta osin elokuvatuotantoa niin teknisen vaativuuden, laajuuden kuin rahoituksenkin osalta. (Soininen 2006)

2.1 Esituotanto ja suunnittelu

Pelituotanto alkaa ideoinnilla ja suunnittelulla, jotka määrittelevät alustavasti projektin eri osa-alueet, perusajatuksen toiminnalle, tarinalle ym. sekä käytettävät resurssit ja työkalut. Pelin ideointi luo pohjan esimerkiksi sille, mitä tyylilajia peli edustaa ja ketkä ovat pelin kohderyhmää ja suunnittelu tukee rakentunutta ideaa ja sen toteutusta teknisestä näkökulmasta. (Soininen 2006)

Maailma on täynnä erilaisia pelejä, joista monet muistuttavat toisiaan niin usealla tavalla, että niitä voidaan pitää saman tyylilajin eli genren edustajina. Saman genren pelien samankaltaisuus johtaa myös siihen, että niillä on usein sama kohderyhmä. Ammattimaiset tuottajat tietävät, millaiset pelit tietyssä genressä ovat menestyneet ja mitä yhteisiä piirteitä ne jakavat. Usein nykypäivänä peliteollisuuden pitkän kehityksen myötä tämä johtaa siihen, että uudet pelit pyrkivät vain toistamaan ja kopioimaan edellisiä menestyneitä pelejä. Ideoiden ja peruselementtien toistaminen saattaa johtaa pelaajakunnan kyllästymiseen, mutta toisaalta liiallinen poikkeaminen tutuista kaavoista voi myös vieraannuttaa kohderyhmän. (Soininen 2006)

Teknisestä näkökulmasta tärkein päätös pelin suunnittelussa on kohdealusta, koska se vaikuttaa oleellisesti pelituotannon teknisiin rajoituksiin, kuten grafiikan vaativuuteen. Mobiilipelille on voimassa erilaiset laitteistorajoitukset kuin esimerkiksi nykyisille pelikonsoleille, joiden suorituskyky vastaa jo melko tehokasta tietokonetta. Kohdealustan valintaan vaikuttavat myös ideointivaiheen genre ja kohderyhmä suuresti, koska

esimerkiksi tietyn genren pelejä pelataan useimmiten tai se on luontevilta tietynkaltaisella ohjaimella, tai koska tietyt konsolit tai muut alustat ovat jo lähtökohtaisesti suunniteltu tietyille kohderyhmille. (Soininen 2006)

Pelin tuottajien määrä ja ammattitaito, käytettävissä oleva budjetti sekä esimerkiksi kohdealusta määrittävät käytettävät työkalut. Pienemmille, itsenäisille niin sanotuille indie-tuottajille riittävät usein ilmaiset perustyökalut, kuten Unity-pelimoottori. Erilaiset valmiit pelimoottorit, kehitysympäristöt ja muut työkalut soveltuvat vaihtelevasti eri alustoille. Suositun pelin myydessä hyvin aiheutuu kuitenkin valmiin pelimoottorin tai muiden resurssien käytöstä usein rojaltimaksuja, minkä vuoksi monet suuret tuottajat valmistavat pelimoottoria myöden kaiken itse. Pelimoottorin ja muiden resurssien kuten kehitystyökalujen valmistuksella tuottajat saavat lisäksi käyttöönsä juuri sen kaltaiset valmiudet, kuin valitulle kohdealustalle tai vaikka genrelle tarvitaan, ja tuotettuja resursseja voidaan hyödyntää tai myydä eteenpäin myös jatkossa.

2.2 Toteutus

Pelin toteutus aloitetaan resurssien sallimana yleensä usean osa-alueen yhtäaikaisella kehityksellä esituotannossa määritellyn suunnitelman mukaisesti. Varsinaisen tuotannon alkaessa oletetaan resurssien kuten pelimoottorin ja työkalujen olevan jo olemassa, ja tuotantovaiheessa pyritään etenemään kehityksessä mahdollisimman paljon suunnitelman mukaisesti pelimekaanikoita tai muita oleellisia ominaisuuksia muuttamatta. (Lahti 2014, 19)

Toteutusvaiheessa julkaistaan erilaisia testiversiota, joilla on omat vaatimuksensa ja ominaisuutensa tuotannon välimaaleina. Alpha-, Beta- ja Gold-versiot jaksottavat pelin tuotantoa loogisten perusteiden mukaisesti. Alpha-vaiheessa pelin perusrakenteen oletetaan olevan kunnossa ja toimivan, mutta ominaisuuksia muutetaan vielä jatkuvasti testauksen seurauksena ja aikana. Alpha-versioita seuraa Beta-versio, joka vastaa jo lähes valmista tuotetta. Beta-versiossa pelin toiminta ja ominaisuudet ovat lopullisessa muodossaan ja Beta-vaihe keskittyy lähinnä testaukseen ja havaittujen virheiden korjauksiin. Julkaisuvalmista pelikandidaattia kutsutaan Gold-versioksi. Gold-vaiheen aikana valmista peliä testataan viimeisen kerran ennen pelin varsinaista julkaisua. (Lahti 2014, 28-30)

2.3 Testaus

Valmista peliä tai testiversiota tuotannon aikana voidaan testata monin eri tavoin, kuten eri osien funktionaalisella testauksella tai koko tuotteen käyttäjätestauksella. Kattava, jatkuva testaus tuotannon eri vaiheissa on tärkeää virheiden ja parannusta kaipaavien osa-alueiden huomaamiseksi mahdollisimman aikaisin. Hyvin testattu peli on julkaisijan ja kehittäjien etu, sillä vastaanotto ja arvostelut vaikuttavat niin suoraan pelin myyntiin, kuin tuottajaryhmän imagoon ja arvostukseen (Lahti 2014, 21).

Tuotetun koodin testaaminen onnistuu kuten minkä tahansa muun ohjelmiston esimerkiksi yksikkötestein, mutta pelin varsinaista toimintalogiikkaa ja kokonaisuutta on vaikea testata automatisoidusti. Tämän vuoksi suuri osa pelitestauksesta toteutetaan manuaalisesti. Kuten koodikatselmoinnissa, myös pelin logiikan, grafiikan ym. testauksessa on suositeltavaa käyttää testaajia, jotka eivät ole osallistuneet itse pelin tuottamiseen. Tehtävää varten peliprojekteissa on usein mukana erillinen testiryhmä, joka on erikoistunut pelien testaukseen.

Pelitestaus on laaja ala itsessään ja pelitestaajilta vaaditaan nykyään pelistä, alustasta ja tuottajista riippuen erilaisia teknisiä valmiuksia sen mukaan, mitä osa-alueita testataan. Testattaviin alueisiin sisältyvät esimerkiksi graafiset ominaisuudet, äänet, tasojen suunnittelu ja pelattavuus sekä suorituskykyyn ja optimointiin liittyvät eri osa-alueet. Teknistä laadunvarmistusta suorittavan testaajan on ymmärrettävä testaamiensa osa-alueiden ominaisuudet ja rajoitukset tavallista kuluttajaa ja toisaalta valmiin pelin lopputestaajia paremmin, mistä johtuen testaus on alana jopa teknisempi, kuin usein ajatellaan. (Lahti 2014, 22)

3 KÄYTETYT KEHITYSTYÖKALUT

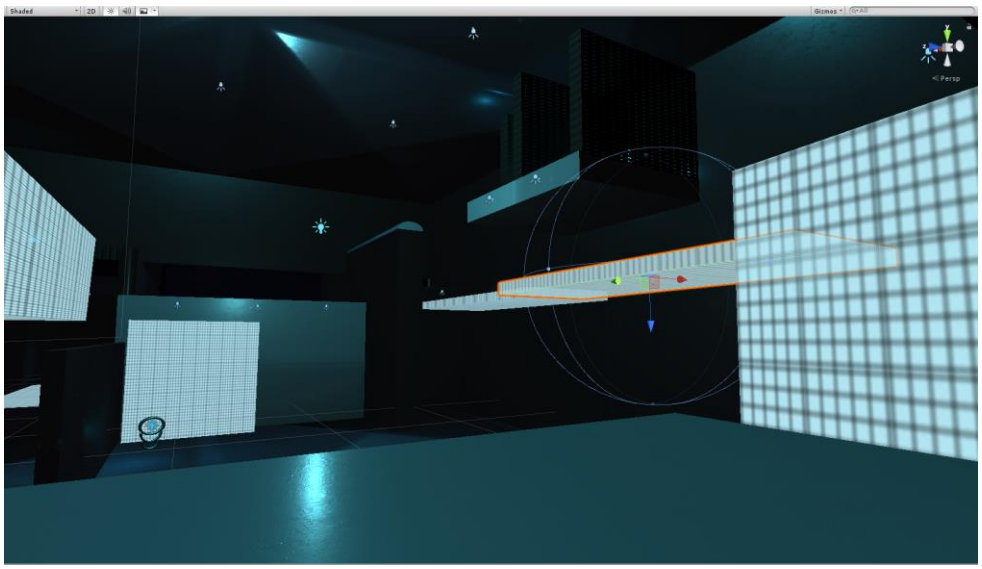
3.1 Unity 5

Unity on monelle eri alustalle soveltuva, pienempien kehittäjien erityisesti suosima pelimoottori. Alun perin 2000-luvun puolivälissä kehitetty pelimoottori on toiminut pitkään halvempana ja yksinkertaisempana työkaluna aloitteleville pelikehittäjille, mutta viidenteen versioonsa mennessä se on kehittynyt varteenotettavaksi vaihtoehdoksi myös isommissa tuotannoissa. Eri alustoille tuotettaessa Unity käyttää Direct3D, Vulkan, OpenGL sekä konsoleiden omia rajapintoja. Pelit Unitylle ohjelmoidaan C#- tai Javascriptin kaltaisella UnityScript-ohjelmointikielellä. (Unity Technologies, 2017)

3.1.1 Käyttöliittymä

Unity 5 sisältää kattavan ja laajasti muokattavan Editor-käyttöliittymän pelien kehitykseen myös ilman ohjelmointia. Käyttöliittymän eri komponentteja voidaan lisätä, poistaa ja siirtää pääosin vapaasti käyttäjän mieltymysten mukaiseksi. Muutamia esimerkkikomponentteja ovat esimerkiksi esikatseluruutu, peliobjektin ominaisuuksien ja komponenttien muokkaus Inspectorilla sekä objektihierarkia. Käyttöliittymäkomponentteja ja työkaluja voidaan myös luoda itse omien koodien avulla. (Unity Technologies, 2017)

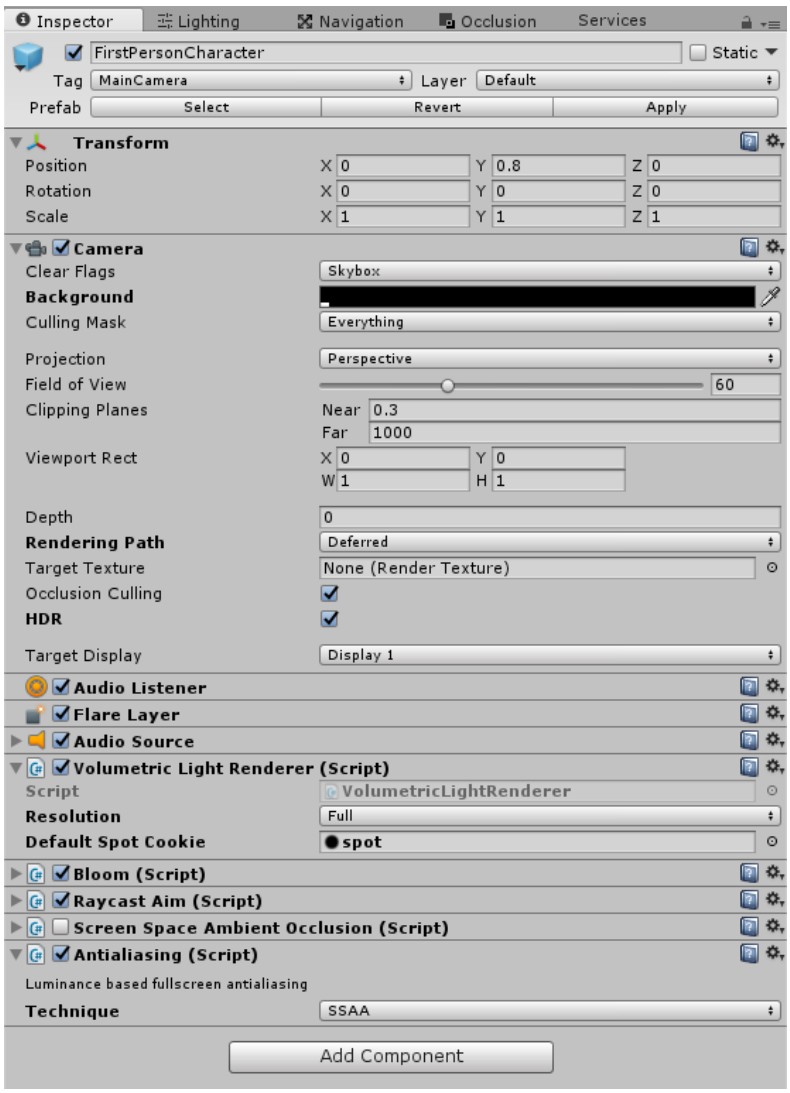
Esikatseluruutu on monikäyttöinen kehitys- ja testityökalu, jolla pelin monia ominaisuuksia voidaan määrittää ja testata käynnistämättä varsinaista peliä. Käyttäjä voi esimerkiksi valita objektien esitystavan materiaalein ja tekstuurein piirretyistä valmiista grafiikoista wireframe- tai erilaisten valotusasetusten määrittämiksi muunlaisiksi kuvioiksi. Esikatseluruudun avulla käyttäjä voi liikkua luomassaan pelimaailmassa täysin vapaasti ja tarkastella peliobjekteja eri kulmista esimerkiksi erilaisten kuvaefektien kanssa tai ilman niitä. Näkymä voidaan myös vaihtaa vapaasta 3D-kamerasta 2D-näkymäksi, mikä helpottaa esimerkiksi sivulta kuvattujen tasohyppelyiden kehitystä. (Unity Technologies, 2017)



KUVA 3.1 Preview-esikatseluruutu

Inspector-komponentilla voidaan muuttaa peliobjektien sisältämiä komponentteja kuten fyysisiä ominaisuuksia ja kaikkia liitettyjä Monobehaviour-koodeja. Mikäli liitettyihin koodeihin sisältyy julkiseksi asetettuja muuttujia, voidaan näiden muuttujien arvoja ja sisältöä muokata suoraan Inspectorissa. Tämä tarkoittaa, että esimerkiksi koodissa viitattaessa kokonaiseen peliobjektiin, voidaan kyseinen peliobjekti määrittää vetämällä objektihierarkiasta tai kansioista haluttu objekti Inspectorissa näkyvään kenttään. Inspectorilla voidaan myös asettaa objektin Tag- ja Layer-muuttujat. Tag-muuttuja määrittää objektin ”luokan” tai ”ryhmän”, eli muut objektit voidaan esimerkiksi törmäyksen sattuessa asettaa reagoimaan tietyllä tavalla tietyn Tag-luokan mukaan. Layer-kerrosmuuttuja määrittää nimensä mukaisesti objektin kerroksen näkymässä varsinaisen nähtävän maailman lisäksi, minkä avulla voidaan esimerkiksi tehdä joistain kerroksista Raycast-säteet läpäiseviä tai määrittää ne osaksi graafista käyttöliittymää tai HUD:ia. (Unity Technologies, 2017)

Kuvassa 3.2 on FirstPersonCharacter-nimisen kameraobjektin sisältö. Kuvan yläreunasta löytyvät objektin nimi, Tag ja Layer, ja niiden jälkeen kaikki muut peliobjektiin liitetyt komponentit. Kamerana toimivalle objektille on siis liitetty erilaisia audiovisuaalisia peruskomponentteja, sekä useita eri kuvaefektejä ja toiminnallisuutta luovia C#-skriptejä.

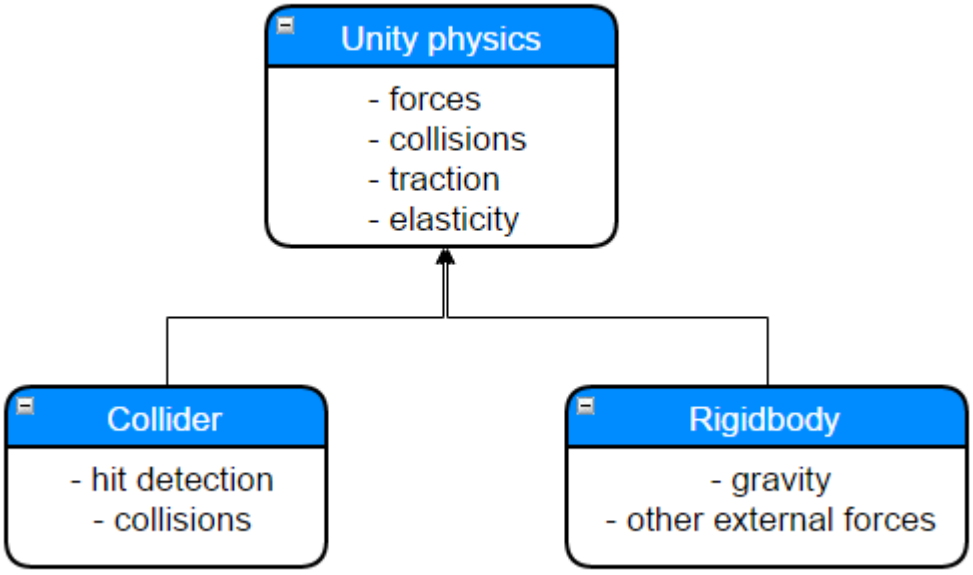


KUVA 3.2 FirstPersonCharacter-kameraobjektin sisältö Inspectorissa

Objektihierarkia sisältää kaikki Scene-tiedoston (ks. luku 4.1) sisältämät objektit aliobjekteineen. Hierarkiasta voidaan helposti määrittää kunkin peliobjektin child- ja parent-objektit ja valita objekteja Inspectorin objektikenttiin niiden sisältämien komponenttien perusteella.

3.1.2 Fysiikka

Unityn sisäänrakennetun fysiikkamoottorin toiminta perustuu Rigidbody- ja Collider-komponentteihin. Molemmat vaikuttavat objektien fyysisiin ominaisuuksiin eri tavalla ja niitä voidaan käyttää joko yhdessä tai erikseen.



KUVIO 3.1 Fysiikkamoottorin rakenne

Rigidbody on komponentti, johon vaikuttavat fysiikkamoottorin eri toiminnot kuten painovoima sekä erilaiset työntövoimat. Kun peliobjektiin on liitetty Rigidbody-komponentti, vaikuttavat objektiin kohdistuvat voimat siihen passiivisesti ilman erillistä ohjelmointia. Ilman Collider-komponenttia kappale ei kuitenkaan reagoi törmäyksiin vaan putoaa suoraan muiden objektien läpi esimerkiksi painovoiman vaikutuksesta. (Unity Technologies, 2017)

Collider-komponentti määrittelee objektin fyysisen muodon ja tarkkailee nimensä mukaisesti kappaleeseen kohdistuvia törmäyksiä. Kun Collider-komponentti saa törmäyksen havaitessaan tietoja osumasta ja osuneesta toisesta kappaleesta, minkä perusteella voidaan esimerkiksi manuaalisesti määritellä ohjelmakoodissa kappale tuhotuksi tai laukaista muita funktioita ja tapahtumia. Collider-komponenttiin eivät kuitenkaan vaikuta ulkoiset voimat kuten painovoima tai törmäyksistä aiheutuvat työntövoimat ilman Rigidbody-komponenttia. (Unity Technologies, 2017)

Kun objektiin on liitetty sekä Rigidbody- että Collider-komponentti, puhutaan Rigidbody Colliderista, johon vaikuttavat kaikki erilaiset voimat, törmäykset sekä objektin materiaalin määrittämät ominaisuudet kuten kitka ja kimmoisuus. (Unity Technologies, 2017) Hyödyntämällä objektien eri fyysisiä ominaisuuksia saadaan aikaan fysikaalisesti realistinen ympäristö, jossa erilaiset kappaleet liikkuvat ja reagoivat toisiinsa kuten tosielämässä.

3.1.3 Grafiikka

Unitylla toteutettujen pelin visuaalinen ilme muodostuu erilaisista valaistus-, materiaali- ja tekstuurielementeistä sekä muokattavasta kamera-komponentista. Käyttäjä voi määrittää kappaleille visuaalisia ominaisuuksia sen perusteella, onko näkyvän esineen tarkoitus olla esimerkiksi metallia tai kiveä ja säädellä sitä, miten kamera näkee kyseiset kappaleet.

Valon tyyppi määrittää sen ominaisuudet ja käyttäytymisen suhteessa muuhun ympäristöön. Peliympäristöön voidaan lisätä erilaisia spotti-, suunta-, alue- sekä pistemäisiä valoja. Näistä spottivalot ja pistemäiset valot päivittyvät jokaisella ruudunpäivityksellä reaaliaikaisesti. Reaaliaikaisia valoja voidaan hyödyntää muuttuvissa olosuhteissa, kuten pelaajan mukana esimerkiksi taskulampun valona. Loput valotyypit ovat staattisia, mikä tarkoittaa, että niiden luoma valo ja sen vaikutus ja heijastukset ympäristössä ovat ennalta laskettuja. Staattisten valojen avulla voidaan luoda valoja, jotka heijastuvat realistisesti kappaleesta toiseen materiaalien ominaisuuksien mukaisesti. (Unity Technologies, 2017)

Materiaalit määrittävät tekstuurien lisäksi sen, miten kappaleet reagoivat valoon ja luovat niihin kohokuvioita ja yksityiskohtia. Erilaisilla Shader-komponenttien ja tekstuurien sekä materiaalin sisäänrakennettujen ominaisuuksien määrittämisellä voidaan luoda näkyvistä kappaleista realistisia tai vaihtoehtoisesti hyvin piirrosmaisia. Materiaalit voivat olla itsensä valaisevia, kiiltäviä kuin metalli tai läpinäkyviä esimerkiksi lasia varten. (Unity Technologies, 2017)

Kamera näkee luodun 3D-peliympäristön ja piirtää näkymän pelaajalle sille asetettujen ominaisuuksien ja parametrien mukaisesti. Kameraa käytetään myös Raycast-laskennassa, jota hyödynnetään esimerkiksi ensimmäisen persoonan perspektiivistä tähtäyksen ja osumien määrittämiseen. Kameraan liitettäviä erilaisia efektejä luovia komponentteja on monia. Niistä yleisimpinä voidaan mainita esimerkiksi liikesumennus, antialiasointi sekä bloom-hohto, joita kaikkia käytetään laajasti nykyaikaisissa peleissä kuvan elävöittämiseksi ja kuvanlaadun parantamiseksi. (Unity Technologies, 2017)

3.1.4 Äänet

Kamera-komponenttiin liitetty AudioListener-kuuntelija komponentti kuuntelee aktiivisesti pelimaailman eri AudioSource-äänilähteitä. Äänilähteet voivat toimia esimerkiksi taustamusiikkina kuuluen kaikkialla tai ne voidaan määrittää 3D-ympäristön mukaisiksi, jolloin kuuntelijan etäisyys ja liike (Doppler-efekti) vaikuttavat äänen voimakkuuteen ja taajuuteen. (Unity Technologies, 2017)

Erilaiset kaiut ja muut ympäristön luomat vaikutukset saavutetaan AudioFilter-suodatinkomponenteilla, jotka määrittävät alueen, jonka sisällä äänet kuuluvat efektin mukaisesti. Suodattimilla voidaan esimerkiksi aiheuttaa äänien kaikuminen luolan kaltaisissa ympäristöissä. (Unity Technologies, 2017)

3.1.5 Event System-komponentti

Peliobjektiin liitetty Event System -komponentti hallitsee käyttäjien syötteitä sekä erilaisia tapahtumia kuten Raycast-tarkistuksia pelin ollessa käynnissä. Käytännössä Event System havaitsee syötteen esimerkiksi näppäimistöltä tai ohjaimesta ja lähettää siitä tiedon sen tarvitseville komponenteille, kuten pelaajahahmon liikettä ohjaavalle koodille. Komponentti itsessään näkyy käytössä passiivisena eikä sen toimintaan ole tarkoituskaan liioin vaikuttaa. Event System toimiikin siis pääasiassa vain erilaisten viestien ja komentojen välittäjänä pelimoottorin sisällä. Raycast-laskentaa käytetään esimerkiksi yhdessä kamera-komponentin kanssa määrittämään mihin pelaaja katsoo tai tähtää, mitä objektia kursori osoittaa tai kuinka lähellä toinen kappale, kuten lattia on lähdekappaleeseen nähden. (Unity Technologies, 2017)

3.1.6 Unity Analytics

Analytics on Unityn analytiikkapalvelu, jota käytetään tietojen kuten pelaajien käytösmallien tarkasteluun. Kun Analytics on lisätty peliin, se saa automaattisesti tietoa pelaajamääristä, uusista pelaajista, pelinsisäisisten ostosten tuotosta jne. Analytics-palvelua voidaan käyttää myös itse määritetyn datan keräämiseen. Peli voi esimerkiksi lähettää viestejä tietyissä kohdissa tai tietyn ajan kuluessa pelatessa, minkä perusteella voidaan nähdä lopettavatko käyttäjät pelaamisen esimerkiksi tietyssä kohdassa ja päätellä, onko kyseisessä kohdassa jotain vialla.

3.2 Microsoft Visual Studio 2015

Visual Studio on Microsoftin luoma ohjelmointiympäristö, joka tukee monia eri ohjelmointikieliä, kuten Unitylle ohjelmoitaessa käytettävää C#-kieltä. Visual Studio integroitavissa Unity 5:en kanssa, jolloin Unityn käyttöliittymästä avatut koodit avautuvat suoraan Visual Studiossa, jossa pelimoottorin luokat ja toiminnot ovat suoraan käytettävissä.

Visual Studiossa on C#-kääntäjä, mutta Unity käyttää kuitenkin omaa kääntäjäänsä koodien kääntämiseen pelille. Tästä ja kääntäjien eri ominaisuuksista johtuen jotkin koodit voivat aiheuttaa virheitä Unityn kääntäjässä, vaikka ne eivät Visual Studion kääntäjästä ole virheellisiä. (Unity Technologies, 2017)

3.3 Blender 3D

Blender 3D on avoimeen lähdekoodiin perustuva 3D-mallinnusohjelmisto, jota käytetään erilaisten hahmojen, esineiden ja muiden kappaleiden luomiseen ja animaatioihin. Oletuksena Blender luo jokaisesta projektista Unityn Scenen kaltaisen tiedoston, joka sisältää kameran, valoja, jne.

Käytettävät 3D-mallit tuodaan Unityyn FBX-tiedostoina, jolloin niitä voidaan lisätä pelimaailmaan ja muokata Unityn omien mallien tapaan. Halutessaan käyttäjä voi myös tuoda Blenderistä malleille luodut materiaalit ja animaatiot, ja integroida ne Unityn omiksi vastaaviksi komponenteiksi. (Unity Technologies, 2017)

3.4 SONAR Steam Edition

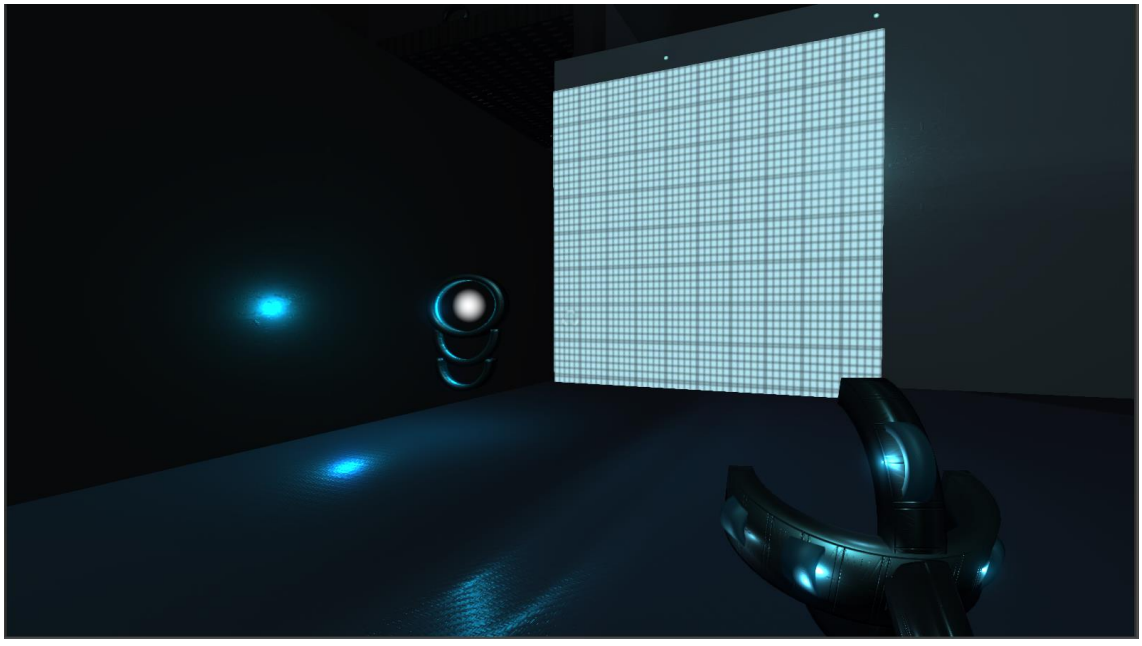
SONAR Steam Edition on Cakewalkin kehittämä DAW-ohjelmisto, jolla voidaan luoda pelien äänimaailma erilaisista efekteistä taustamusiikkiin. SONAR toimii kuten mikä tahansa äänen ja musiikin äänitykseen ja muokkaamiseen tarkoitettu ohjelmisto, mutta

Steam edition tuo mukanaan lisäksi pelikehitykseen tarkoitettuja ominaisuuksia ja valmiita ääniefektejä. (Cakewalk Inc. 2015)

4 PELIDEMON TOTEUTUS

4.1 Pelin idea

Tuotettu pelidemo Light on ensimmäisestä persoonasta kuvattu ongelmanratkaisupeli. Demossa pelaajan on tarkoitus selvittää tiensä kahden varsinaisen ongelmanratkaisukammion läpi ja toimittaa kaksi Orb-valoa maalina toimivalle Grand Beacon -soihdulle. Kammiot sisältävät useita Orb-valoja sekä Beacon-soihtuja, jotka aktivoivat/deaktivoivat kappaleita, jotka toimivat esteinä, siltoina ja portaina pelaajalle. Soihduilla ja pelaajalla voi kerrallaan olla vain yksi valo, joten esimerkiksi deaktivoidakseen soihdun pelaajan täytyy mahdollisesti ensin hankkiutua eroon kantamastaan valosta. Kuvassa 4.1 kuvakaappaus pelin sisältä. Kuvassa näkyy vasemmalla Beacon-soihtu, jonka takana aktivoinnin seurauksena luotu kappale sekä pelaajan työkalu, joka kuvassa ei kanna Orb-valoa.



KUVA 4.1 Kuvakaappaus ensimmäisestä kammiosta

4.2 Scene-tiedostot

Scene-tiedostot sisältävät pelin eri osa-alueiden, kuten eri tasojen, kaikki komponentit järjestettynä valmiiksi ympäristöiksi. Scenet eivät sisällä omaa toiminnallisuutta, koska

ne toimivat vain projektin osien jakajina tai säiliöinä. Scenen lataaminen toisen Scenen sisältä suoritetaan Unityn SceneManagement-luokan LoadScene-funktiolla.

4.2.1 Loader

Loader-scene on käynnistettäessä ensimmäinen latautuva Scene ja toimii palautekomponentin alustajana. Palautekomponentti sisältää koodin, joka laskee peliaikaa ja lähettää ajan sekä käyttäjältä kerätyn palautteen Analyticsiin ja REST-rajapinnan kautta tietokantaan. Komponentille on koodissa määritetty lisäksi DontDestroyOnLoad-ominaisuus, joka pitää sen olemassa ja aktiivisena myös Scenen vaihtuessa, jolloin jokaiselle Scenelle ei tarvitse luoda omaa vastaavaa komponenttiaan ja tietojen lähetys onnistuu mistä tahansa Scenestä.

4.2.2 Menu

Menu-scene sisältää pelin päävalikon ja se ladataan Loaderin jälkeen. Päävalikko sisältää vain painikkeet, jotka avaavat varsinaisen pelin tai palautelomakkeen, tai sulkevat ohjelman. Scene toimii siis vain solmukohtana muiden välillä.

4.2.3 Main

Main-scene on pelin varsinainen pelattava osio ja sisältää demon ainoan ”tason”. Suuremmassa tuotoksessa Main-scenen kaltaisia tasoja olisi useita, esimerkiksi yksi jokaiselle selvitettävälle huoneelle tai kammiolle.

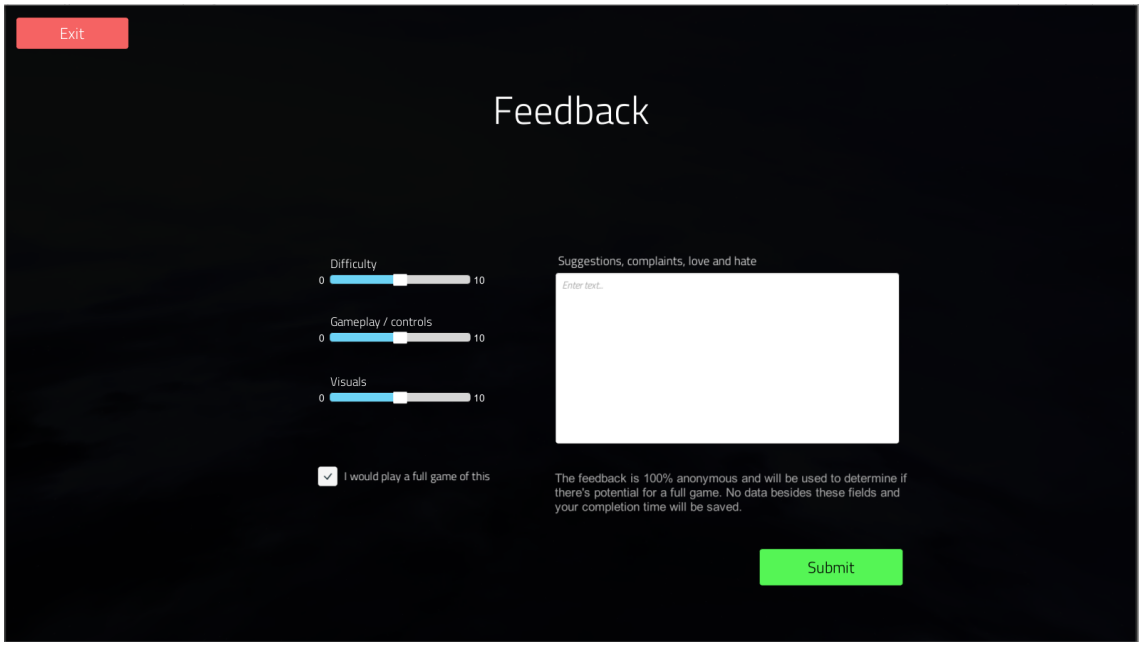
Kammiot ja muu ympäristö rakentuvat pääosin Unityn omista 3D-kappaleista, kuten kuutioista ja sylintereistä, mutta myös joistakin itse tuotetuista yksinkertaisista kappaleista. Demoa varten yksinkertaiset kuutiot ym. geometriset kappaleet toimivat riittävän hyvin, ja erilaiset yksityiskohdat ja koristeet ovat toteutettu materiaalien ja tekstuurien avulla, jotta demo olisi mahdollisimman kevyt myös vähemmän suorituskykyisillä tietokoneilla.

4.2.4 Feedback

Feedback-scene sisältää palautelomakkeen, jonka avulla käyttäjiltä kysytään mielipidettä pelin kyseisestä tilasta sekä mahdollisia jatkokehitysehdotuksia. Feedback on siis Menun kaltainen valikko-Scene, joka sisältää seuraavat komponentit:

- Vaikeusaste-liuku (0-10)
- Pelattavuus-liuku (0-10)
- Visuaalinen ilme -liuku (0-10)
- ”Pelaisinko tällaista peliä”-valintaruutu
- Vapaa sana -tekstikenttä
- Poistu-painike
- Lähetä-painike

Kun käyttäjä on syöttänyt tiedot ja painaa Lähetä-painiketta, tiedot lähetetään Loader-scenen luomalla palautekomponentille, joka välittää ne eteenpäin eri rajapintoihin. Kuvassa 4.2 kyseinen Scene ajossa.



KUVA 4.2 Feedback-scene

4.3 Monobehaviour-luokka

Unityn sisäisiä toimintoja hyödyntävät oliot ja luokat perivät Monobehaviour-luokan, joka sisältää rajapinnat pelimoottorin eri osien resursseihin. Luokan avulla voidaan esimerkiksi etsiä koodin sisältävästä peliobjektista eri komponentteja ja käyttää niiden toimintoja. Kaikki demoa varten luodut luokat perivät Monobehaviour-luokan ja ne ovat listattuna taulukossa 4.3 käyttötarkoituksineen.

TAULUKKO 4.3 Pelidemon omat luokat

Activator	Activator Beacon-objektia hallitseva luokka, vastaanottaa Raycastin mukaan lähetettyjä aktivointikomentoja.
Feedback	Feedback-scenen pääluokka, jonka muuttujien arvoja käyttöliitttymän eri komponenttien arvojen muuttaminen asettaa. Lähettää palautedatan Stats-luokalle.
Goal	Grand Beacon-objektia hallitseva luokka, joka muuttaa objektin sekä aliobjektin materiaaleja ja animaatioita tilan mukaan.
Loader	Loader-scenen luokka, joka toimii paneelien häivytysten avulla kosmeettisena parannuksena.
Menu	Menu-scenen pääluokka, joka vastaanottaa valikon painikkeiden kutsut ja lataa seuraavan Scenen.
MusicController	Hallitsee äänilähdettä, josta pelin taustamusiikki kuuluu MusicTrigger-luokan objektien kutsujen mukaan.
MusicTrigger	Lähettää MusicControllerille tiedon, kun pelaaja astuu liitetyn Colliderin (trigger) sisään.

PauseMenu	Main-scenen pysäytysvalikon toimintoja kuunteleva luokka.
RaycastAim	Luokka, joka tarkastelee pelaajan kamerasta lähteviä Raycasteja sekä sitä, mihin ne osuvat. Lähettää aktivointikomentoja Activator- ja Toggler-luokille sen mukaisesti, kantaako pelaaja tai kohde valoa vai ei.
Respawn	Pudotuskohtiin pelimaailmassa sijoitetut Colliderit (trigger) lähettävät liitetyille Respawn-luokilleen komennon siirtää pelaaja takaisin turvalliseen sijaintiin.
RespawnChange	Luokka, joka muuttaa Respawn-luokan siirtosijaintia esimerkiksi, jos samaan aukkoon pudotessa tulosuunnalla on väliä pelaajan jumiin jäämisen estämiseksi.
Ring	Tarkastelee Grand Beacon-objektin Ring-aliobjektin materiaalia ja käynnistää tai pysäyttää animaation.
Stats	Palautedatan lähettämisestä Unity Analyticsiin ja omaan REST-rajapintaan vastaava luokka. Stats-luokka alustetaan Loader-scenessä ja se pysyy aktiivisena koko pelin käynnissä olon ajan.
Toggler	Vastaava luokka kuin Activator, mutta hallitsee Toggler Beacon-objektia.

4.4 Käyttäjäpalautteen REST-rajapinta

Käyttäjien palautetta kerätään Unity Analytics -palvelun lisäksi oman rajapinnan kautta TAMKin palvelimella sijaitsevaan SQL-tietokantaan. Ylimääräinen tietokanta luotiin, koska Analytics ei pysty vastaanottamaan yli 500-merkin viestejä, joita vapaan sanan

sisältävässä kentässä saattaa olla. Lisäksi Analytics ei vielä tue raakadatan latausta, joten kaikkien tietojen yksityiskohtainen tarkastelu on vaikeaa.

Rajapinta on Slim-frameworkin mukaan tehty SlimDB PHP-toteutus koulun palvelimella, jonne pelin Stats-luokka lähettää kerätyn palautedatan JSON-muodossa HTTP POST kutsulla. Lähetetty JSON parsitaan rajapinnassa ja lähetetään sen jälkeen koulun palvelimella sijaitsevaan SQL-tietokantaan talteen.

REST-kutsu suoritetaan WWW-luokan avulla, joka muodostuu kohde-URL:sta, varsinaisesta lähetettävästä sisällöstä eli palautteen kentistä JSON-muodossa sekä header-osasta, joka määrittää lähetettävän sisällön tyypin olevan muotoa JSON:

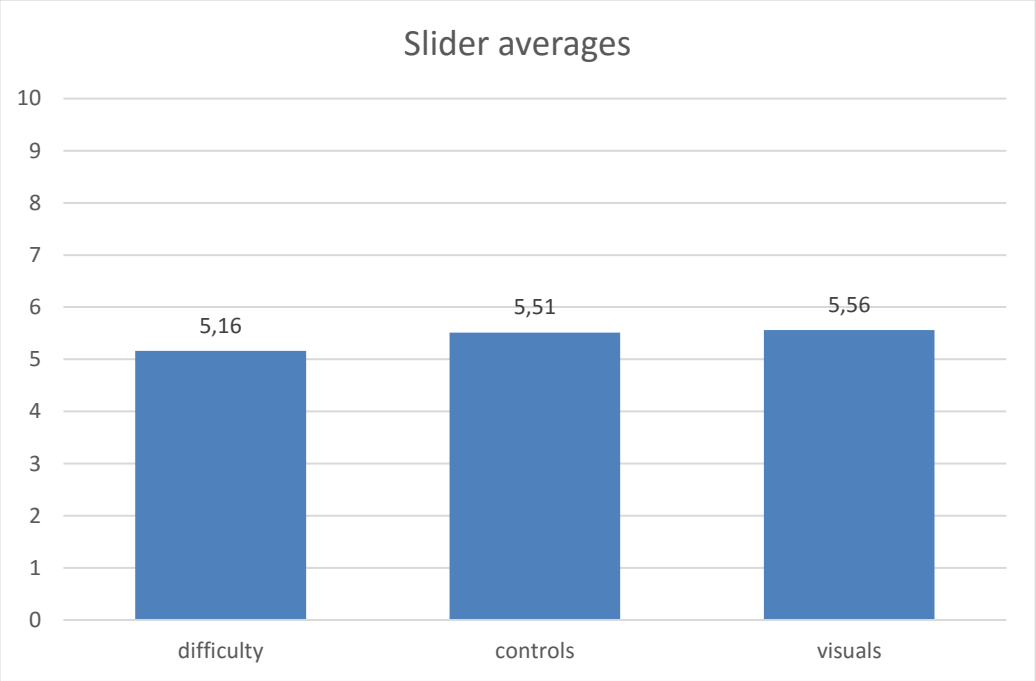
```
Dictionary<string, string> header = new Dictionary<string, string>();
header.Add("Content-Type", "application/json");
WWW www = new WWW(URL, System.Text.Encoding.UTF8.GetBytes(jsonString), header);
StartCoroutine(WaitForRequest(www));
```

Käyttäjien syötteet puhdistetaan välimerkeistä (esimerkiksi ”- ja ’-merkeistä) jo ennen REST-kutsusta vastaavalle Stats-luokalle lähettämistä, joten palvelin voi parsia ja lähettää peliltä tulevan sisällön pääosin sellaisenaan ilman vaaraa SQL-injektioista.

5 TESTAAJIEN PALAUTE

Käyttäjiltä kerättyä palautetta Unity Analytics palveluun sekä SQL-tietokantaan vastaanotettiin odotettua vähemmän, mutta saaduista palautteista voidaan kuitenkin tehdä johtopäätöksiä demon onnistumisesta ja ominaisuuksista. Otanta palautteissa on noin 20 satunnaista, anonyymiä käyttäjää.

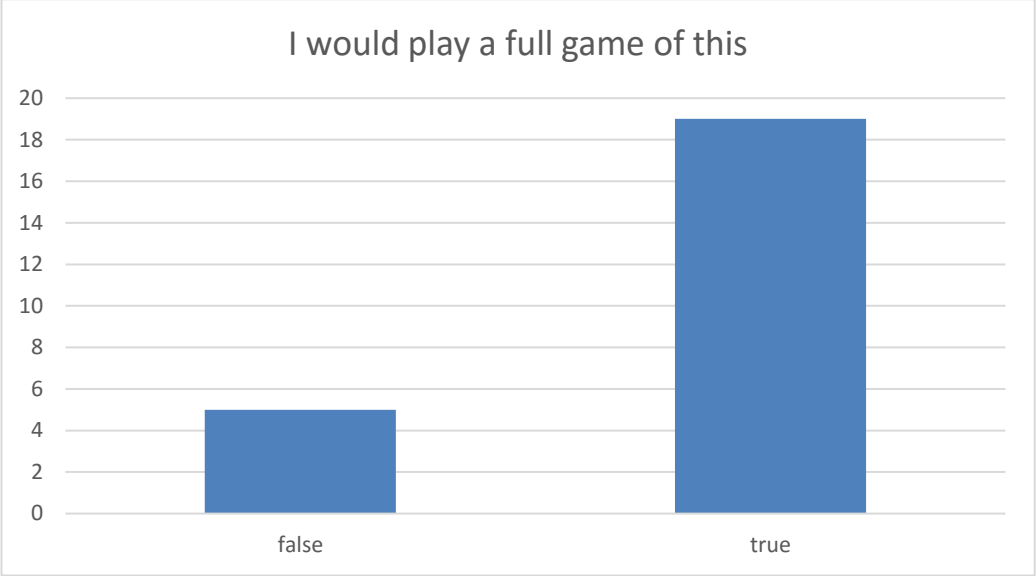
Kuviossa 5.1 ovat liukuri-komponenttien eli vaikeusasteen, visuaalisen ilmeen ja ohjauksen/pelattavuuden keskiarvot palautteista.



KUVIO 5.1. Palautekeskiarvot

Arvosteluasteikolla 0-10 saadut tulokset ovat kaikkien kolmen arvon osalta keskitasoa.

Kuten kuviosta 5.2 näkyy, suurin osa testaajista pelaisi myös pelin täyttä versiota, keskitasoisista arvosanoista huolimatta.



KUVIO 5.2. Testaajien mielipide demoa vastaavasta kokonaisesta pelistä

Palautteen mukana lähetettiin myös testaajan peliaika, mikäli palaute lähetettiin onnistuneen läpipeluun seurauksena. Suurin osa palautteista ei kuitenkaan sisältänyt tätä tietoa, mikä tarkoittaa, ettei kovin moni testaajista pelannut demoa loppuun asti. Näin ollen saaduista peliajoista ja arvosteluista ei voida vetää hyviä johtopäätöksiä eikä aikoja käsitellä tarkemmin.

Kaiken kaikkiaan palautteista vedettävät johtopäätökset kuvaisivat todellisuutta paremmin suuremmalla otannalla, mutta jo noin 20 testaajan palautteista pystytään pääosin päättelemään missä demo on onnistunut ja mitkä asiat kaipaavat parannusta. Tässä päättelyssä toimi oleellisena osana vapaa tekstikenttä, johon testaajat saivat kirjoittaa mielipiteitään ja avata arvosteluperusteitaan tarkemmin.

6 POHDINTA JA JATKOKEHITYS

Kaiken kaikkiaan demoprojektin toteutus Windowsille ja pelin tuottamisen oppimisprosessi voidaan katsoa onnistuneeksi. Projektin suunnittelu ja toteutus eivät vastaa täysin teoreettisesti käsiteltyjä toimintatapoja ja ominaisuuksia, mikä johtuu pääasiassa varsinaisen projektiryhmän puuttumisesta. Koska aikaa oli vähän ja tekijöitä vain yksi, tietyissä vaiheissa kuten testaamisessa ja tuotoksen yleisessä viimeistelyssä oikaistiin joissain kohdin. Valmis demo tietokantoineen ja rajapintoineen saatiin joka tapauksessa julkaistua ja käyttäjiltä kerättyä tietoa vaihtelevalla menestyksellä sekä SQL- että Analytics-tietokantaan. Oman SQL-kannan ottaminen mukaan tiedonkeruuseen oli välttämätöntä, kun selvisi, ettei Unity Analytics sovellu kaikenlaisten omien viestien tallentamiseen esimerkiksi merkkirajoitusten vuoksi.

Saadun palautteen perusteella voidaan tehdä johtopäätön, että peli on perustuksiltaan kunnossa, mutta käyttäjät jäivät kaipaamaan tarinaa, loivempaa vaikeusastekäyrää sekä käyttäjäystävällisempää visuaalista ilmettä. Tarinan ja vaikeusasteen ongelmat ovat selitettävissä tehdyn demon lyhyydellä, sillä käytössä olevassa ajassa oli toteutettava riittävän monipuolinen demo kuvaamaan peliä mahdollisimman tarkasti. Tästä syystä varsinaisen tarinan suunnittelu ja toteutus jätettiin lähes kokonaan huomiotta, sekä vaikeusaste nostettiin sellaiselle tasolle, ettei sen asteittaiseen nostamiseen vaadittaisi pitkää pelijaksoa, mikä olisi teettänyt huomattavan määrän työtä tasojen suunnittelun ja toteutuksen myötä. Visuaalinen ilme oli käyttäjien mukaan miellyttävä, mutta liian synkkä joillakin näytöillä. Lopullisen pelin kehityksessä tulisi siis ottaa huomioon myös erilaiset peliympäristöt ja näyttötyypit esimerkiksi säädettävällä gamma-arvolla.

Palautteista voidaan myös päätellä, että toteutetun demon kaltaiselle pelille löytyisi pelaajia. Suuremman pelin suunnittelussa ja toteutuksessa täytyisi kuitenkin ottaa tarkemmin huomioon peliprojektien yleisiä toimintatapoja ja suunnitella prosessi yleisesti hyväksi todettujen käytänteiden mukaisesti. Tällä hetkellä on epäselvää, tuotetaanko lopullista peliä tulevaisuudessa, mutta projektin aikana ja ansiosta lisääntynyt ymmärrys ja osaaminen pelituotannossa voidaan katsoa hyödyksi yleisesti, joten työn tekeminen oli kannattavaa.

LÄHTEET

Unity Technologies. Unity User Manual (5.5). Käyttöohje. Luettu 14.3.2017
https://docs.unity3d.com/Manual/index.html

Slim Framework Team. Slim Framework Documentation. Käyttöohje. Luettu 28.3.2017
https://www.slimframework.com/docs/

Lahti, M. 2014. Game testing in Finnish game companies. Aalto-yliopisto. Maisterityö. PDF.
https://aaltodoc.aalto.fi/bitstream/handle/123456789/15193/master_Lahti_Mikko_2015.pdf

Soininen, T. 2006. Pelituotanto. Tampereen Teknillinen yliopisto. PDF.
http://www.cs.tut.fi/kurssit/OHJ-2710/2006JaVanhemmat/Pelisuunnittelu2006_4.pdf

Cakewalk Inc. 2015. Cakewalk Announces SONAR Steam Edition now available on Steam Platform. Julkaisutiedote. 19.10.2015.
https://www.cakewalk.com/Press/Release/1235/Cakewalk-Announces-SONAR-Steam-Edition-now-available-on-Steam-Platform