Opinnäytetyö (AMK)

Tietotekniikka

Mediatekniikka

2015

Jussi Elsilä

ÄÄNIMOOTTORIN KÄYTTÖ OSANA ÄÄNISUUNNITTELUA PELINKEHITYSYMPÄRISTÖSSÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Mediatekniikka

2015 | Sivumäärä 46

Mika Luimula, Yliopettaja, FT

Jussi Elsilä

ÄÄNIMOOTTORIN KÄYTTÖ OSANA ÄÄNISUUNNITTELUA PELINKEHITYSYMPÄRISTÖSSÄ

Opinnäytetyössä käsiteltiin äänimoottoreiden käyttöä väliohjelmistona osana äänisuunnittelua pelinkehitysympäristössä. Työ toteutettiin turkulaiselle startup-yritykselle FakeFish Oy:lle, ja tarkoitus oli kartoittaa oikean äänimoottorin valinta eri vaihtoehdoista. Tärkeimpinä lähteinä työssä toimivat äänimoottoreiden valmistajien internetsivut, käyttöoppaat ja alan kirjallisuus. Lähteistä selvitettiin äänimoottorien tärkeimmät ominaisuudet, äänimoottorien vertailu ja niiden tehokas käyttö.

Työssä perehdyttiin ensin äänimoottoriin väliohjelmistona, sen ominaisuuksiin ja käytön mahdollisuuksiin osana äänisuunnittelua. Erilaisia äänimoottoreita valittiin työtä varten neljältä eri valmistajalta ja äänimoottoreiden soveltavuutta tehokkaaseen käyttöön arvioitiin niiden ominaisuuksien, lisenssien ja käyttöystävällisyyden perusteella. Lopuksi valittiin soveltuvin äänimoottori työkäyttöön. Työssä käytiin läpi käytännön esimerkkien kautta FMOD Studio äänimoottorin käyttöä Unity-pelinkehitysympäristössä.

Työn toteuttamista helpotti aikaisempi kokemus äänitekniikan alan ohjelmien käytöstä, musiikin harrastamisesta ja taidot ohjelmoinnin perusteista. Äänimoottorin käyttö osana äänisuunnittelua ja pelinkehitystä teki työstä enemmän joustavaa ja säästi samalla ohjelmoijan työtaakkaa. Äänimoottorin monipuolinen käyttö saatiin havainnollistettua yksinkertaisten esimerkkien avulla, joita voitiin soveltaa käyttötarkoituksen mukaan.

Lopuksi pohdittiin tavoitteiden ja työn onnistumista sekä miten opinnäytetyöstä olisi voinut saada laajemman kokonaisuuden. Laajemman kokonaisuuden tavoittamiseksi työn määrä olisi ollut kuitenkin moninkertainen.

ASIASANAT:

äänisuunnittelu, äänimoottori, väliohjelmisto, pelinkehitysympäristö, Unity, FMOD

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Digital Media

2015 | Total number of pages 46

Mika Luimula, Principal Lecturer, PhD

Jussi Elsilä

THE USE OF AN AUDIO ENGINE IN AUDIO DESIGN WITHIN THE GAME DEVELOPMENT ENVIRONMENT

The thesis covers the usage of audio engines as middleware in audio design in a game development environment. The thesis was commissioned by a Turku-based start-up company called FakeFish with the goal of mapping out the correct choice of an audio engine for their project. The main sources for the thesis consisted of audio engine developers' web pages, guidebooks and literature from the audio field. By the means of the source material, the most important qualities, differences and effective usage of audio engine were covered.

In the theoretical section of this thesis, the properties and possibilities of using an audio engine as middleware in audio design were explored. Four different audio engines from different developers were selected for closer inspection and their suitability for the project was evaluated according to their features, licences, and ease of use. Based on these factors, an audio engine most suitable for the project was chosen. In the empirical section, the use of the FMOD Studio audio engine within the Unity3D development environment was demonstrated with relevant examples.

Previous experience in audio workstations, working in the music field and basic knowledge of programming helped with the completion of the project. By using an audio engine in audio design and game development, the development was more adaptive and at the same time lessened the programmer's workload. The diverse use of the audio engine was illustrated with simple examples that could be applied for different uses.

A discussion of the goals and results of the thesis took place in the ending chapter, including possible additions to the scope of the thesis. However with a broader scope the amount of work required would have been multiplied.

KEYWORDS:

audio design, audio engine, middleware, game development environment, Unity, FMOD

SISÄLTÖ

KUVAT

TAULUKOT

SANASTO

Aikajana	Visuaalinen alue, jossa ääntä näytetään lineaarisessa tai epälineaarisessa muodossa.
Aikakursori	Kursori, joka näyttää mitä kohtaa aikajanalta toistetaan.
Automaatioraita	Tapa, jolla voidaan vaikuttaa ennalta esimerkiksi äänen voimakkuuteen tai sävelkorkeuteen.
DAW	*Digital Audio Workstation* eli äänityöasema.
Desibeli	Äänenvoimakkuuden mittayksikkö, dB.
Ekvalisaattori	Musiikkityökalu, jolla voidaan säätää äänisignaalin taajuusvastetta.
Implementointi	Äänien tuominen pelimoottoriin ja niiden asettaminen toistumaan oikeisiin kohtiin koodissa.
Kanavaliuku	Mikserien äänikanavien äänenvoimakkuusliuku.
Kompressori	Työkalu äänisignaalin dynamiikan pienentämiseksi.
Kuuntelija	Komentorivi, joka on kiinnitetty peliobjektiin ja kuuntelee peliympäristössä tapahtuvia ääniä.
Mikseri	Äänipöytä, johon voidaan tuoda ja josta voidaan viedä äänisignaalia, jonka luonnetta voidaan erilaisilla säädöillä.
MIDI	*Musical Instrument Digital Interface* eli musiikkisoittimien digitaalinen tiedonsiirtojärjestelmä.
Modulointi	Äänisignaalin luonteen muokkaus, kuten sävelkorkeuden tai äänenvoimakkuuden virittäminen.
Muuttuja	Tietovarasto, josta voidaan kutsua tai johon voidaan tuoda tietoa.
Panorointikiertosäädin	Säädin, jolla voidaan muuttaa äänisignaalin sijaintia stereokuvassa.
Pelimoottori	Pelien ohjelmistokehys, jonka päälle voidaan rakentaa pelejä eri laitteille.
Ryhmälähtö	Ryhmälähtö voi sisältää monen eri äänikanavan signaalin.
Sekvensseri	Musiikkityökalu, jolla voidaan luoda musiikkia.
Suoratoisto	Äänisignaalia luetaan suoraan levyltä muistista lukemisen sijaan.

Sävelkorkeus	Äänisignaalin taajuus, jota mitataan yleensä puolisävelaskeleina.
Tagi	Tapa, jolla voidaan linkittää peliobjekteja keskenään ja antaa niille tunniste.
Väliohjelmisto	Ohjelman osa, joka suorittaa yksittäisiä tehtäviä ohjelmiston sisällä.
Äänenpakkausmenetelmä	Tapa, jolla äänisignaalia saadaan pakattua tiivimpään muotoon tilan säästämiseksi.
Äänikanava	Kanava, joka ottaa sisään äänisignaalia.
Äänitapahtuma	Äänimoottorissa ääniä sisältävä tapahtuma, jonka käyttäytymistä ja toiminnallisuutta voidaan muokata käyttötarkoituksen mukaan.

1 JOHDANTO

Äänimoottorin käyttö pelinkehitysympäristöissä on tullut jatkuvasti suositummaksi lähinnä sen kehittyneen käyttöliittymän ansiosta ja käyttämisestä ilman laajempaa ohjelmointitaitoa. Perehtyminen äänimoottorin ominaisuuksiin on tarpeellista, koska äänisuunnittelijan on ymmärrettävä niiden tarkoitus ja vaikutus, jotta saadaan tavoiteltava lopputulos aikaiseksi äänimoottorin käyttäytymisessä.

Äänimoottorin tehokas käyttö vaatii ympärilleen myös pelimoottoriin, jossa äänimoottoria voidaan hyödyntää. Opinnäytetyön tarkoitus on kartoittaa äänimoottorin käytön mahdollisuuksia osana pelimoottoria ja havainnollistaa, miten äänisuunnittelua voidaan toteuttaa tehokkaasti väliohjelmistoa käyttäen ja ilman vaativampaa ohjelmointitaitoa.

Opinnäytetyössä tutkitaan äänimoottorien toiminnallisuutta ja ominaisuuksia osana äänisuunnittelua ja vertaillaan neljän eri äänimoottorin ominaisuuksia sekä niiden yhteensopivuutta pelimoottoreihin. Opinnäytetyössä tuodaan esille erilaisia äänimoottorin käyttötapoja esimerkkien kautta. Opinnäytetyö toteutettiin FakeFish Oy:lle Turussa.

Ominaisuuksista on kerrottuina kaikki oleellisimmat, jotka löytyvät hieman eri muodossa jokaisesta esitellystä äänimoottorista. Oikean äänimoottorin valintaan erilaisia projekteja varten vaikuttavat sen käyttökustannukset, tekniset rajoitukset ja käyttökokemus. Lopussa havainnollistetaan käytännön esimerkkien kautta, miten äänimoottorin ominaisuuksia peliympäristössä voidaan käyttää tehokkailla tavoilla. Yhteenvedossa pohditaan tuloksia ja kuinka opinnäytetyötä olisi vielä voinut laajentaa aiheeltaan.

2 ÄÄNIMOOTTORI PELINKEHITYKSESSÄ

Äänimoottorilla tarkoitetaan ohjelman osaa, jonka tehtävänä on toistaa ääntä ja hoitaa sen käyttäytymistä. Äänimoottorilla tarkoitetaan tyypillisesti väliohjelmistoa (engl. middleware), joka auttaa välittämään tietoa kahden eri ohjelman välillä. Väliohjelmiston avulla äänisuunnittelijan on helpompi hallita pelin tai ohjelman äänitapahtumia, ilman vaativampaa ohjelmointitaitoa. (Brandon 2007; Roos 2012.)

Useimmat äänimoottorit ovat erillisiä ohjelmistoja, joita voidaan yhdistää eri pelimoottoreiden kanssa. Suurin osa niistä on kaupallisia, mutta viime aikoina monet valmistajat ovat antaneet ohjelmistojaan vapaaseen käyttöön, kuten opiskelijoille tai ei-kaupallisiin tarkoituksiin. (Emusician.com 2014.)

2.1 Väliohjelmisto

Väliohjelmistolla tarkoitetaan ohjelmistoa, joka yhdistää ohjelman eri osia tai komponentteja tai siirtää tietoa toisesta ohjelmasta toiseen hajautetussa ympäristössä. Väliohjelmistoa voidaan kutsua eräänlaiseksi eri ohjelmien väliseksi liimaksi, joka yhdistää ja helpottaa ohjelmien välistä keskustelua ja käyttöä. (Middleware.org n.d.; Lea Kutvonen 2014, 4.)

Väliohjelmistona äänimoottori toimii yhdessä pelimoottorin kanssa ja käytännössä se toimii yhdistävänä työkaluna äänisuunnittelijan ja ohjelmoijan välillä. Väliohjelmiston tehtävänä on antaa äänisuunnittelijalle enemmän hallintaa ja valtaa määrätä äänen käyttäytymistä pelimoottorin sisällä. (Emusician 2014.)

2.2 Väliohjelmistot osana pelien äänisuunnittelua

Pelien kehittyessä monimutkaisemmiksi ja pelattavuudeltaan laajemmiksi, äänimoottorin kaltaisia väliohjelmistoja on kehitetty helpottamaan työntekoa ja

-kulkua pelimoottorin, ohjelmoijan ja äänisuunnittelijan välillä. Ennen väliohjelmistojen kehittymistä nykyiseen muotoonsa äänisuunnittelija usein toi ohjelmoijalle ääniä erilaisissa tiedostomuodoissa, minkä jälkeen ohjelmoija implementoi ne oikeisiin kohtiin pelin koodiin. Internetin suomien mahdollisuuksien kehittyessä äänisuunnittelija pystyi työskentelemään toiselta puolelta maailmaa ja lähettämään äänityönsä verkon ylitse yleensä tekstitiedoston kanssa, joka sisälsi tietoa äänen nimistä, äänen käyttäytymisestä ja sen mihin kohtaan ne on tarkoitettu toistettavaksi. (Horowitz & Looney 2014, 124.)

Väliohjelmistot ovat nykyisin helppokäyttöisiä kehittyneemmän graafisen käyttöliittymän ansiosta, joka mahdollistaa äänisuunnittelijan tehokkaammaan työskentelyn ja äänien miksaamisen pelissä reaaliajassa. Ennen graafisen käyttöliittymän yleistymistä äänien implementoiminen tapahtui yleensä käyttämällä pelkästään äänimoottoreiden sovellusrajapintaa. (Collins ym. 2014, 447.)

Nykyiset väliohjelmistot, kuten FMOD ja Wwise, ovat usein järjestelmäriippumattomia (engl. cross-platform). Järjestelmäriippumattomuuden ansiosta äänimoottoreita voidaan käyttää usealla eri käyttöjärjestelmäalustalla. Äänisuunnittelijan ei tarvitse miettiä yhteensopivuusongelmia eri järjestelmien välillä, vaan voi keskittyä äänenlaadullisiin seikkoihin eri päätelaitteille implementoidessa. Järjestelmäriippumattomien väliohjelmistojen teknisenä haasteena on kuitenkin monen eri valmistajien käyttöjärjestelmäalustojen jatkuva kehittyminen eteenpäin ja näin ollen väliohjelmistojen valmistajien on hidasta pysyä perässä. (Collins ym. 2008, 82.) Järjestelmäriippumattomien väliohjelmistojen käyttö on kuitenkin suosittua, koska samaa väliohjelmistoa käyttäen eri käyttöjärjestelmäalustoilla voidaan tuotteen rahallinen tuotto maksimoida (Bradleymeyer.com 2013).

Kehittyneen prosessoreiden laskentatehon ja parantuneen suorituskyvyn ansiosta äänisuunnittelussa säästetään nykyään aikaa ja työresursseja reaaliaikaisen miksauksen ja äänen suorituskyvyn kuormituksen profiloimisen myötä. Ennen kuin reaaliaikainen miksaus peleissä tuli mahdolliseksi,

äänisuunnittelija joutui tekemään ääneen liittyvät miksaukset sekä säädöt erillisesti ja sen jälkeen ajamaan pelin uusilla äänen muokkauksilla. Reaaliaikaisella miksauksella voidaan verkon ylitse miksata pelissä tapahtuvia ääniä pelin ollessa käynnissä ja tämän jälkeen tallentaa väliohjelmistoon hyväksi todetut asetukset. Väliohjelmistoihin voidaan liittää miksauksen helpottamiseksi MIDI-yhteensopiva mikseri, jolloin saadaan fyysinen kosketuspinta äänitasojen muokkaamiseksi. (Kuva 1.) (Audiokinetic.org 2014a; Fmod.org 2015c.)

Kuva 1. Midi-yhteensopiva mikseri. (Gearnuts.com 2015.)

Massiiviset ja näyttävät tietokonepelit, kuten myös lyhyemmät ja mobiileille suunnitellut pelit, tuovat mukanaan tekniset rajoitteet. Mitä enemmän pelissä on yleensä tapahtumia ja interaktiivisuutta, sitä enemmän se vaatii äänisuunnittelua, ja tämän tuloksena on usein enemmän äänidataa sekä toistettavia tapahtumia. Pienemmissä mobiililaitteissa saattaa laitteen muisti ja suorituskyky tulla vastaan siinä missä tietokoneiden tekniset rajoitukset tulivat vastaan 1990-luvulla. (Horowitz & Looney 2014, 196). Helpotuksena liialliselle kuormituksena, väliohjelmistot mahdollistavat nykyään reaaliaikaisen miksauksen tavoin äänien suorituskyvyn kuormituksen seurannan suorittimille ja

muistin käytölle. Suorituskyvyn seurannalla vältytään ikäviltä yllätyksiltä pelejä pelattaessa, kuten ohjelman kaatumiselta tai jumittumisilta. (Horowitz & Looney 2014, 127.)

2.3 Äänimoottorin ominaisuuksia

2.3.1 Graafinen käyttöliittymä

Äänimoottoreiden väliohjelmiston graafinen käyttöliittymä on tehty käyttäjäystävälliseksi ja ne muistuttavat kaukaisesti suosituimpien musiikkituotantoon suunnattujen DAW-ohjelmien ja sekvensserien käyttöliittymää, kuten Pro Tools tai Logic Pro. Väliohjelmistot sisältävät samoja osia työympäristöstä, joskin niiden lähestymistapa toteuttaa jokin asia, saattaa poiketa. (Kuva 2.)

Kuva 2. Logic Pro X:n käyttöliittymä.

Väliohjelmisto jakautuu yleensä käyttöliittymältään äänitapahtuma- ja pankkiselaimeen, tapahtumaeditoriin, mikseriin ja hallintapaneeliin (Audiokinetic.com 2015b; FMODTV 2012).

Äänitapahtuma- ja pankkiselaimessa on lueteltuna kaikki projektin sisältämät tiedostot hakemistopuun tapaan. Äänitapahtumat voidaan järjestää erikseen kansioihin käyttötarkoituksen mukaan. Tapahtumaikkunasta äänitapahtuman valitsemalla se aukeaa tapahtumaeditoriin, jossa suurin osa työskentelystä tapahtuu. Tapahtumaeditorissa määritetään äänitapahtuman käyttäytyminen ja sen ominaisuudet. Mikseri sisältää muiden miksauspöytien tapaan äänikanavia, kanavaliu'ut, panorointikiertosäätimet ja ryhmälähdöt sekä ryhmäpaluut. Äänikanaviin voidaan liittää myös erilaisia efektejä ja liitännäisiä, kuten ekvalisaattorin, kompressorin tai kaiun. Hallintapaneelin avulla voidaan äänitapahtumia toistaa, tauottaa, pysäyttää tai laittaa looppaamaan.

2.3.2 Äänitapahtumat

Pelien äänien ja musiikin toisto tapahtuu erilaisista lähteistä. Äänimoottorin yhteydessä lähteiksi kutsutaan äänitapahtumia, jotka sisältävät äänen tai kokoelman ääniä. Äänitapahtumia on erilaisia ja jokainen niistä määrittyy käyttötarkoituksen mukaan, jonka käyttäjä näkee parhaiten soveltuvan pelissä tapahtuvan tapahtuman mukaan. (Harvey 2015, 78; Audiokinetic 2015c.)

Äänitapahtumien äänen tai äänien yleisimmät säätimet ovat äänenvoimakkuus ja sävelkorkeus. Äänitapahtumalle voidaan myös määrittää, onko se itseänsätoistava eli looppaava ja missä järjestyksessä äänet toistetaan. Jokainen äänitapahtuma voidaan priorisoida käyttötarkoituksen mukaan, jolloin vähemmän tärkeille tapahtumille annetaan matalampi prioriteetti kuin tärkeille tapahtumille.

2.3.3 Mikseri

Monia äänitapahtumia voidaan reitittää niputtamalla ne samaan ryhmälähtöön mikserissä, jolloin voidaan yhtäaikaisesti käyttää monelle äänelle samoja efektejä ja liitännäisiä. Tämä helpottaa erityisesti isoissa projekteissa äänien hallinnoimista ja säästää samalla suorituskykyä, koska jokaiselle äänelle ei tarvitse erikseen määritellä efektejä. (Harvey 2015, 43 - 46.)

Snapshotit ovat keino automatisoida mikserin lähtöjen ja efektien asetuksia. Snapshotteja voidaan luoda tilanteen mukaan ja niitä käytetään tilanteissa, jossa peli vaatii erityistä miksausjärjestelyä. Pelitilanteen vaatiessa erityisjärjestelyä, voidaan ottaa sitä varten omansa snapshot käyttöön, milloin se yliajaa mikserin vakioasetukset väliaikaisesti. Erityistilanteita varten voidaan käyttää myös kahta tai useampaa snapshottia, milloin väliohjelmisto osaa sulautuvasti käyttää monta snapshottia kerrallaan, kuitenkin priorisoiden enemmän snapshotin asetuksia, joka on korkeimmalle asetettu hierarkiassa. (Audiokinetic 2010, Video.)

2.3.4 3D-äänien tuki

3D-ääni on ääntä, joka toteutetaan manipuloimalla ääntä äänenvoimakkuuden, viiveiden ja kaikujen avulla. Tietokonepeleissä 3D-äänet yleistyivät 1990-luvulla. (Wikipedia.org 2015a.) 3D-äänen käyttäytymistä määritetään minimi- ja maksimietäisyydellä (engl. min and max distance). Etäisyyksien määrittämistä voidaan havainnollistaa kuuntelijalla, joka on sijoitettu keskelle autoa. Auton moottorista lähtevä ääni saavuttaa kuuntelijan noin kahden metrin säteellä, jolloin minimietäisyydeksi määritellään kaksi. Jos minimietäisyys on määritelty nollaksi, ääni alkaa heti vaimeta kohti kuuntelijaa. Tämän lisäksi määritellään myös maksimietäisyys, jonka saavuttaessaan ääni ei ole enää lainkaan kuultavissa. Minimi- ja maksimietäisyyden välissä tapahtuvan vaimenemisen muodon määrittää käyrä, joka voi olla lineaarinen tai epälineaarinen. Poistamalla 3D-ääni käytöstä, saadaan äänitapahtumasta 2D-ääni. 2D-ääneen

ei vaikuta äänilähteen etäisyys, vaan se toistuu peliympäristössä aina samanlaatuisena. (Harvey 2015, 175 - 178.)

2.3.5 Äänipankit

Äänipankit *(*engl. soundbanks*)* ovat kokoelmia, jotka sisältävät äänitapahtumia ja muuta projektissa käytettyä sisältöä. Pankit tuodaan pelimoottoriin ja ne sisältävät kaiken tarvittavan tiedon äänitapahtumista pelimoottoria varten. Pankkeja on oletusarvoisesti yksi, mutta käyttäjä voi halutessa luoda useampia äänipankkeja käyttötarkoituksen mukaan. Kun äänitapahtumat ovat sijoitettu äänipankkiin tai äänipankkeihin, pitää ne rakentaa (engl. build) luettavaksi pelimoottoriin. Äänipankkeja rakentaessa pitää ottaa huomioon, mitä äänenpakkausmenetelmää käytetään ja kuinka paljon ääntä kompressoidaan sekä suoratoistetaanko (engl. stream) ääni suoraan levyltä. (Harvey 2015, 214)

2.3.6 Muuttujat

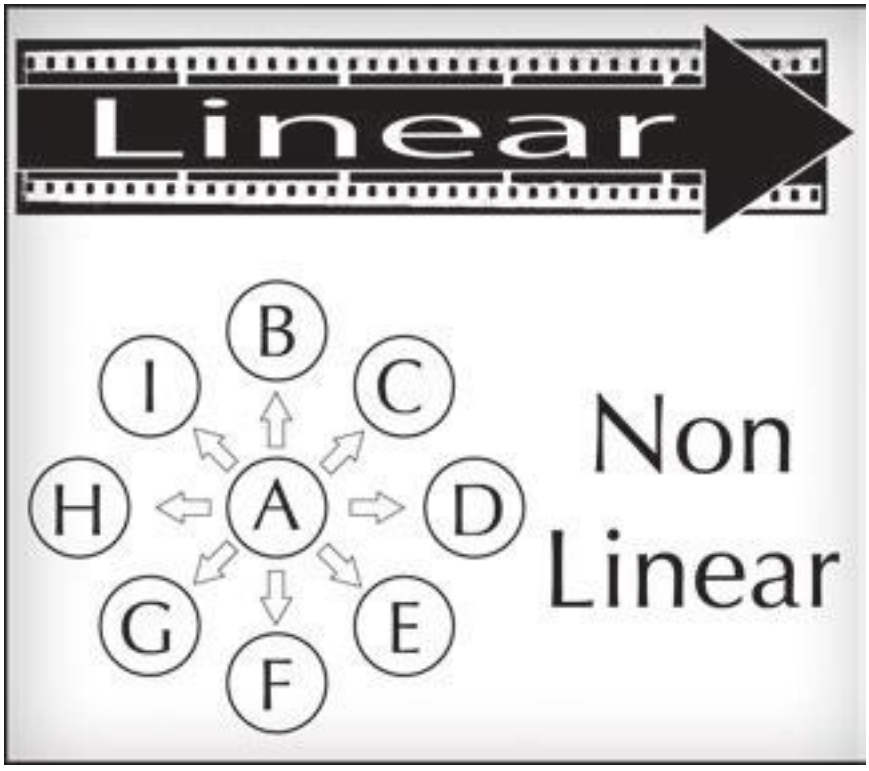
Muuttujalla (engl. parameter) voidaan muuttaa FMODin sisällä, viedä peliin tai tuoda pelistä informaatiota. Muuttujalle voidaan antaa jokin arvo, jonka perusteella esimerkiksi pelistä tuotava arvo asettaa aikakursorin tiettyyn kohtaan aikajanalle, määrittää sävelkorkeus tai -voimakkuus. Muuttujia on kahden tyyppisiä: itsemääriteltyjä tai sisäänrakennettuja. (Harvey 2015, 85-87.)

Itsemääritetyillä muuttujilla voidaan vaikuttaa yleisimmin kursorin käytökseen aikajanalla tai tapahtumassa. Itsemääriteltyjä muuttujia käytetään esimerkiksi äänitapahtumaan, jossa tapahtuu jatkuvaa muutosta. Muutos voi olla esimerkiksi auton moottori, johon kaasua lisäämällä kierrokset kasvavat ja näin ollen ääni muuttuu. Muuttuja voidaan osoittaa reagoimaan auton kierroksiin ja toimimaan se yhteisesti pelissä ajettavan auton kanssa. Muuttujaa voidaan käyttää myös tilanteissa, joissa pelin intensiteetti muuttuu jännittävämmäksi ja muuttujalla voidaan viedä aikakursoria eteenpäin kohtiin, joissa musiikkikin on intensiivisempää. Itsemääritetyillä muuttujilla voidaan myös käyttää yhdessä

siirtymäkohtien kanssa, jolloin muuttujan saadessaan kutsutun arvon siirtymäkohdasta aikakursori siirtyy sille ennalta määritetylle paikalle. (Harvey 2015, 85 & 100 - 110).

2.3.7 Epälineaarinen toisto

Epälineaarinen toisto on lineaarisen toiston vastakohta. Lineaarinen toisto on yleisin tapa miten olemme tottuneet näkemään ja kuulemaan musiikkia. Lineaarisella musiikilla on alku, kesto, aaltomuoto ja loppu – esimerkiksi totuttu tapa on laittaa musiikkikappale soimaan soittimesta, kelailla sitä ja lopettaa kappale. Epälineaarisen toiston ja musiikin ymmärtäminen on paljon haasteellisempaa kuin lineaarisen. (Kuva 3.)



Kuva 3. Lineaarisen ja epälineaarisen havainnollistaminen kuvana. (Horowitz 2014, 35.)

Epälineaarisessa toistossa äänitapahtuma voi sisältää monia lyhyitä pätkiä musiikkia tai ääniä, mitkä ovat yleensä tarkoitettu looppattavaksi. Epälineaarisen toiston käyttäytymistä säädellään muuttujilla. Esimerkiksi pelissä

tapahtuvan tunnelman kiihtyessä, muuttuja säätelee hetki hetkeltä äänitapahtuman epälineaarisen musiikin toistoa saavuttamalla juuri toivotun intensiteetin pelaamiseen. Epälineaarisessa musiikissa käytetään usein ristihäivytystä (engl. crossfading), joka tarkoittaa kahden äänen sulautumista toisiinsa, jolloin ensimmäisen äänen äänenvoimakkuus vaimennus alkaa samanaikaisesti kun toisen äänen äänenvoimakkuuden nosto alkaa. Epälineaarisella toistolla tavoitetaan enemmän yhtenäisyyttä usein pelin kulkuun, joka usein on myös pelinkerronnaltaan epälineaarista pelaajan päätöksien johdosta edetä pelissä. (Prunotto 2014; Horowitz 2014, 36.)

2.3.8 Modulointi

Moduloinnilla voidaan vaikuttaa äänen muuttujiin, sävelkorkeuteen tai äänenvoimakkuuteen eri tavoilla. Tällä tavalla yksittäiselle äänelle tai monille äänille saadaan enemmän käytettävyyttä ja rikkautta, muuttamalla sen luonnetta. (Harvey 2015, 57 - 63.)

Satunnaisuudella (engl. random) määritellään tietty arvoväli, josta satunnaisesti ohjelma valitsee jonkin arvon. Esimerkiksi askeläääniä toistaessa, voidaan vaikuttaa niiden sävelkorkeuteen joka toistokerralla eri arvoilla, jolloin jokainen toistettu ääni kuulostaa hieman erilaiselta.

Sidechain sallii toisen sisääntulevan arvon moduloida annettua muuttujan arvoa. Kyseessä voi olla esimerkiksi äänen voimakkuus, jolloin halutaan pelissä tapahtuvan toisen äänen hiljentävän toisaalla tapahtuvaa ääntä. (Robinson 2014, 15.17.3.)

2.3.9 Liitännäiset

Väliohjelmistot tukevat usein kolmannen osapuolen liitännäisiä, joiden avulla voidaan laajentaa äänimoottoreiden efektikirjastoa. Useimmiten kyseessä on erilaiset kaiku-, viive, äänimallinnus-, kompressointi- ja ekvalisaattoriliitännäiset. Liitännäisiä on saatavilla sekä ilmaiseksi ja kaupallisesti.

2.4 Unity-pelinkehitysympäristö

Unity on Unity Technologiesin kehittämä monialustainen pelimoottori, jolla voi kehittää pelejä tietokoneille, pelikonsoleille, mobiililaitteille ja internet-sivuille. Unity on alansa käytetyin pelimoottori 45 %:n markkinaosuudella kaikista olemassa olevista pelimoottoreista. Unity on käytössä ilmainen tuottojen pysyessä alle 100 000 dollarissa. Tuottojen ylittäessä kyseisen rajan, pitää kehittäjän ostaa käyttölisenssi, joka maksaa 75 dollaria kuukaudessa tai kertamaksuna 1 500 dollaria. (Unity3d.com 2015a; 2015b; 2015c.)

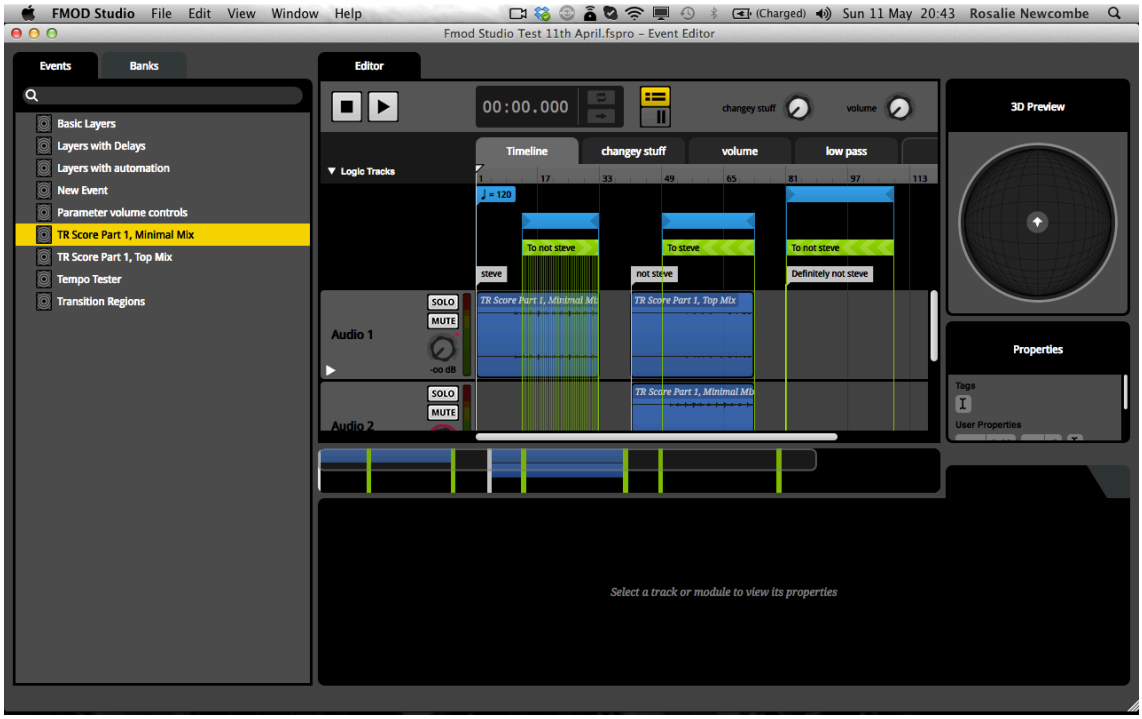
3 ÄÄNIMOOTTORIVAIHTOEHTOJA

3.1 Tutkittavat äänimoottorit

3.1.1 FMOD Studio

FMOD on Firelight Technologiesin kehittämä osaksi kaupallinen äänimoottori ja yritys on perustettu vuonna 2002 Australiassa. FMODin tuoteperheeseen on aiemmin kuulunut FMOD Ex ja FMOD Designer. FMOD Studio on yksi alansa käytetyimpiä äänimoottoreista ja sitä on käytetty väliohjelmistona yli 1 500 pelissä. (Fmod.org 2015a.)

FMOD Studio on suunniteltu käyttöliittymältään helppokäyttöiseksi, kuitenkin unohtamatta sen kattavia ja monipuolisia editointiominaisuuksia. FMOD Studio tukee 3D-ääniä, äänien reaaliaikaista miksausta ja suorituskyvyn seurantaa sovellustestauksen aikana ja ohjelman mukana tulee kokoelma erilaisia efektejä. FMOD Studiota voidaan käyttää yhdessä C-, C++- ja C# -ohjelmointikielten kanssa. FMOD Studio toimii Windows- ja Mac OSX -alustoilla. (Kuva 4.) (Fmod.org 2015c.)



Kuva 4. FMOD Studio.

FMOD Studion yhteensopivuusliitännäinen julkaistiin Unity-pelimoottorille vuonna 2013. FMODin käyttäytymistä ohjataan Unitystä joko mukana tulevilla komentosarjoilla tai itse ohjelmoimalla. (Fmod.org 2013.)

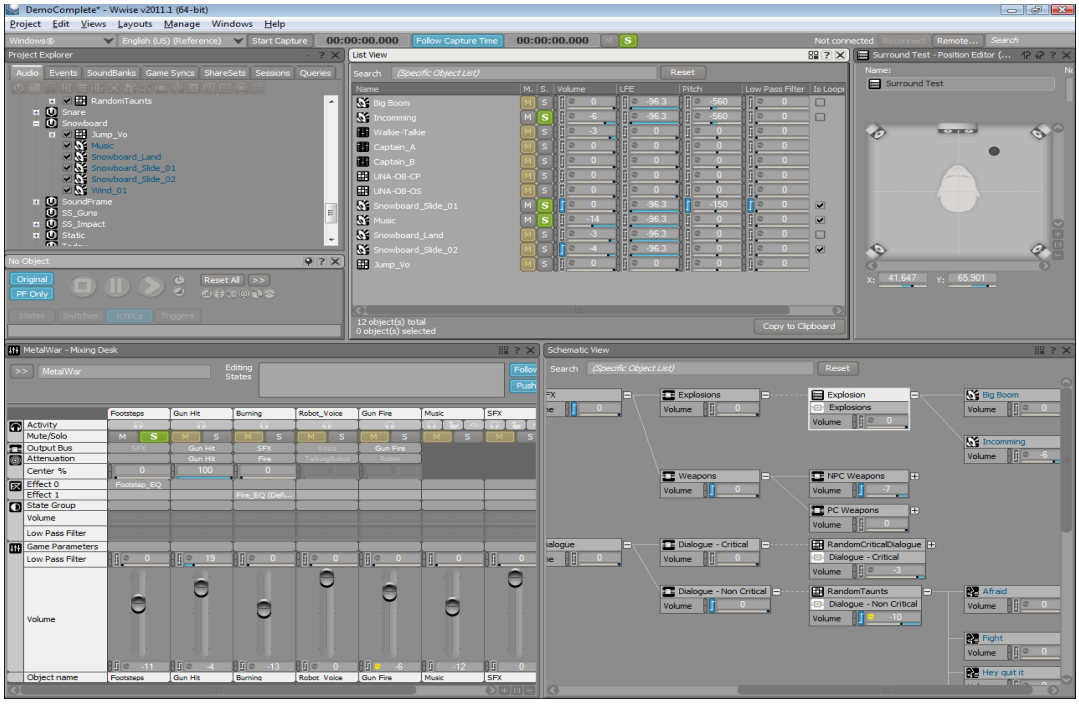
FMOD Studion yhteensopivuusliitännäinen julkaistiin Unreal Engine 4 -pelimoottorille vuonna 2014. Yhteensopivuusliitännäinen sulautuu Unreal Engine 4:sen graafiseen käyttöliittymään Blueprintiin, joka auttaa äänisuunnittelijoita tuoda äänimaailmaa peliin ilman ohjelmointitaitoa. (Fmod.org 2014.)

3.1.2 Audiokinetic Wwise

Wwisen (Wave Works Interactive Sound Engine) on kehittänyt kanadalainen Audiokinetic, mikä esiteltiin alun perin vuonna 2006. Wwise on vahva kilpailija em. FMOD Studioon ja moni tunnettu pelitalo on ottanut sen käyttöönsä. Wwisen käyttölisenssi määräytyy tarkoituksen mukaan ja lisenssejä on kolme:

kaupallinen, rajoitettu kaupallinen ja ei-kaupallinen lisenssi. (Audiokinetic.com 2015a; Wikipedia.org 2015b.)

Wwise on käyttöliittymältään hieman monimutkaisempi, mutta sen muokkausmahdollisuudet riittävät erittäin vaativiin ja tehokkaisiin ratkaisuihin. Wwiseä voidaan käyttää reaaliaikaisesti sovellustestauksen aikana miksaamiseen ja ohjelman mukana tulee myös työkalu äänien suorituskyvyn seurantaan. Wwise tukee 3D-ääniä ja ohjelman erinomainen ominaisuus on tuki MIDI:lle. (Kuva 5.) (Audiokinetic.org 2014b.)

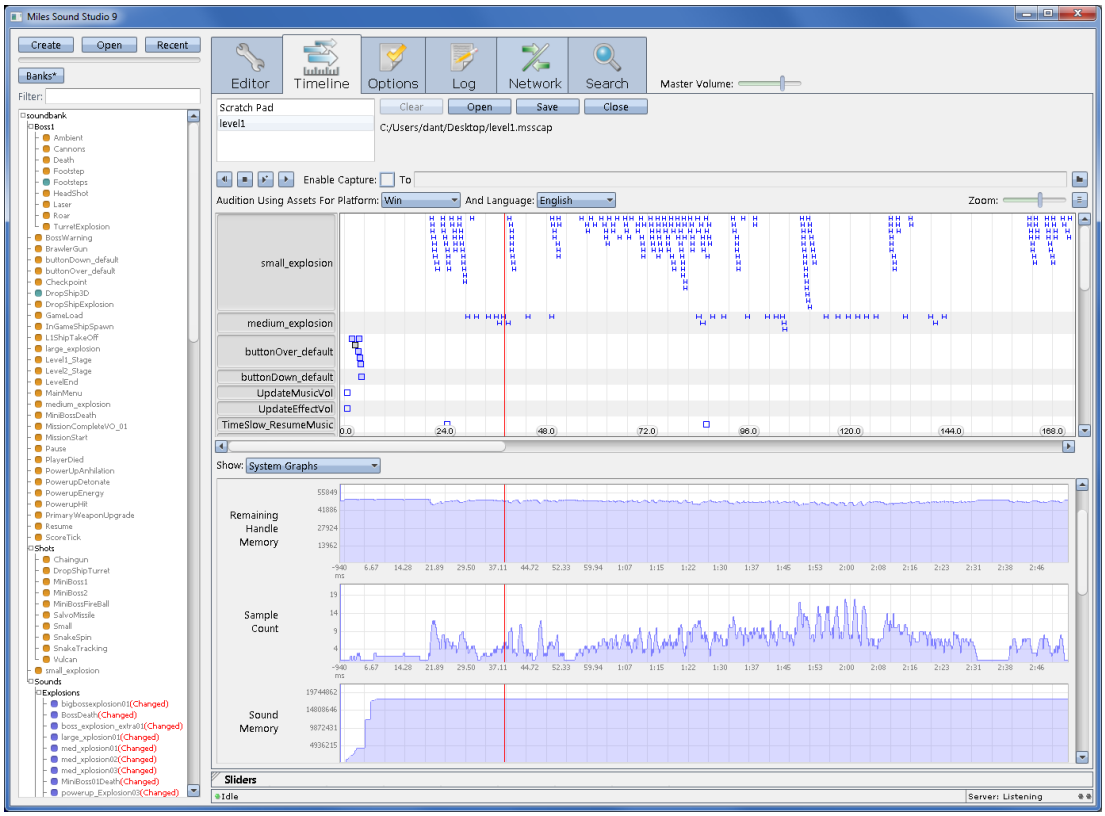


Kuva 5. Audiokinetic Wwise. (Snipview 2012.)

3.1.3 RAD Game Tools Miles Sound System

Miles Sound System (MSS) on RAD Game Toolsin kehittämä äänimoottori, jota on kehitetty vuodesta 1991 saakka. Miles Sound System on lisensoitu yli 5200 peliin 14 eri sovellusalustalla. RAD Game Tools tarjoaa erilaisia sekä räätälöityjä lisenssejä käyttötarkoituksen mukaan. (Kuva 6.) (Radgametools.com 2015.)

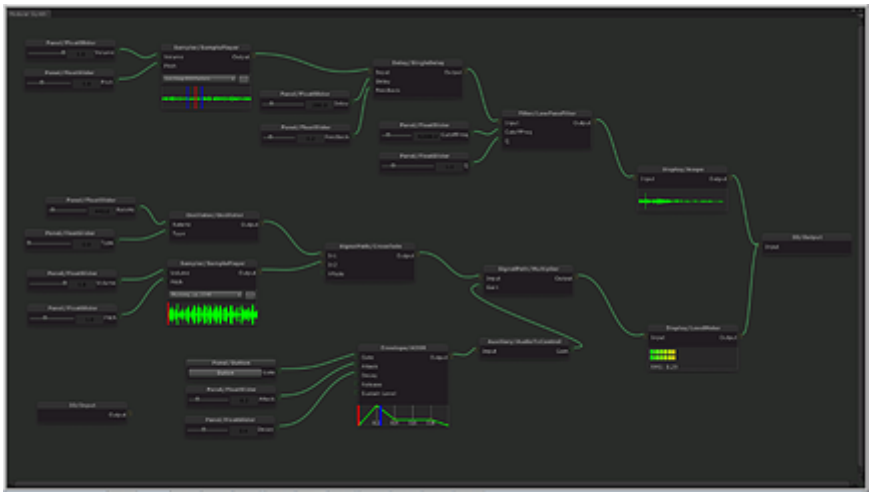
Miles Sound System tukee 2D- ja 3D-ääniä, monikanavaista miksausta ja käyttää äänen pakkaamiseen muiden pakkaamiskoodekkien lisäksi omaa Bink Audio -koodekkia. Käyttöliittymältään Miles Sound System on melko karu ja vaatii hieman enemmän tutustumista.



Kuva 6. Miles Sound System. (Radgametools.com 2015)

3.1.4 Tazman-Audio Fabric

Fabric on Tazman-Audion kehittämä äänityökalulaajennus Unity-pelimoottoriin. Vaikka Fabric ei ole itsessään äänimoottori, se laajentaa Unityn äänenkäsittely- ja käyttäytymisominaisuuksia. Fabric Unityn äänenmuokkausominaisuuksien lisäksi sisältää monia samoja ominaisuuksia kuin muut äänimoottorit. Yhtenä heikkoutena Fabricissa on kolmannen osapuolen liitännäisten sopivuuden tuen puute. Fabricia on saatavilla kolmella eri käyttölisenssillä, joista yksi on ilmainen alle 100 000 dollarin budjetin pelituotannoissa. (Kuva 7.) (Tazman-audio.co.uk 2015a, 2015b.)



Kuva 7. Tazman-Audio Fabric. (Tazman-audio.co.uk 2015.)

3.2 Lisenssit

Äänimoottoreista FMODista, Wwisesta ja äänilaajennuksesta Fabricista on tarjolla kolme erilaista käyttölisenssiä: kaupallinen, ei-kaupallinen ja rajoitettu käyttölisenssi. Ne määräytyvät käyttötarkoituksen sekä peliin tai ohjelmaan käytettävien tuotantokulujen mukaan. (Fmod.org 2015b; Audiokinetic.com 2015d; Tazman-audio.co.uk 2015b.)

Fabricin ja FMODin kohdalla rajoitettua käyttölisenssiä voidaan käyttää kun peliin tai ohjelmaan käytettävät tuotantokulut pysyvät alle 100 000 dollarissa. Wwise tarjoaa rajoitetun käyttölisenssin kohdalla erilaisen ratkaisun: kaupalliseen lisenssiin tarvitsee siirtyä silloin, kun Wwise-projektin assettien määrä ylittää yli 200 (Audiokinetic.com 2015d). Rajoitetussa käyttölisenssissä ei ole rajoituksia julkaisualustasta tai maksuja.

Kaupallista käyttölisenssiä vaaditaan viimeistään kun peliin tai ohjelmaan käytettävät tuotantokulut ylittävät FMODin ja Fabricin kohdalla yli 100 000 dollaria ja Wwisen kohdalla 200 assetin määrän. Käyttölisenssin hinta nousee portaittain tuotantokulujen ja julkaistavien alustojen lukumäärien mukaan. Esimerkiksi peliä julkaistaessa yhdelle julkaistualustalle ja tuotantokulujen ollessa 200 000 - 500 000 dollaria käyttäen FMODia käyttölisenssin hinnaksi muodostuu 3 000 dollaria. Wwisen käyttölisenssin hinta samaa esimerkkiä

käyttäen olisi 6 000 dollaria olettaen 200 assetin käyttömäärän ylittyneen. Miles Sound Systemin käyttölisenssistä ei ole valmistajan sivuilla mainintaa, mutta eräässä haastattelussa käyttölisenssin hinnaksi mainitaan 5 000 dollaria. (Develop-online.net 2012.)

Muiden kuin Miles Sound Systemin ei-kaupallinen käyttölisenssi on ilmainen ei-kaupalliseen tai opetuskäyttöön. Projektia tai sen sisältämää kirjastoa, ei saa edelleen myydä tai käyttää muuhun kaupalliseen tarkoitukseen. Lisenssi ei kata myöskään voittoa tavoittelemattomia nimikkeitä, jotka julkaistaan kaupallisesti. Opetustarkoituksessa oppilaiden käyttämä lisenssi on ei-kaupallinen ja kaikessa ei-kaupallisessa tarkoituksessa tuotteessa pitää yleensä mainita äänimoottorin valmistaja ja tuote lopputeksteissä.

Äänimoottoreihin voidaan myös ostaa käyttölisenssien lisäksi esimerkiksi teknistä tukea valmistajalta tietyksi ajaksi, tai erillisiä lisenssejä kolmannen osapuolen liitännäisille.

3.3 Äänimoottoreiden vertailu ja yhteensopivuus pelimoottoreihin

Äänimoottorien ominaisuuksia vertaillessa, lähtökohdan muodostivat taulukossa 1 äänimoottorin toimivuus käyttöjärjestelmissä, dokumentaation saatavuus, laajennettavuus, tehokasta työskentelyä tukevat ominaisuudet, käyttöliittymän käyttäjäystävällisyys ja taulukossa 2 yhteensopivuus pelimoottoreiden kanssa.

Äänimoottorin dokumentaation saatavuutta ja käyttöliittymän käyttäjäystävällisyyttä on arvioitu asteikolla 1 - 5, jossa yksi (1) on heikko ja viisi (5) on paras. Käyttäjäystävällisyys perustuu käyttöliittymän selkeyteen ja helppokäyttöisyyteen. (Taulukko 1.)

Taulukko 1. Äänimoottoreiden ominaisuuksien vertailu.

	FMOD Studio	**Wwise**	**Miles Sound System**	**Fabric**
Käyttöjärjestelmäalustat	Windows, Macintosh	Windows, Macintosh	Windows	Windows, Mac
3D-äänet	x	x	x	x
Reaaliaikainen miksaus	x	x	x	x
Reaaliaikainen profilointi	x	x	x	x
Ulkoisen mikserin tuki	x	x		
3. osapuolen liitännäistuki	x	x		
Tuki MIDI-tiedostoille		x	x	
Käyttöliittymän käyttäjäystävällisyys	5	4	2	3
Dokumentaation saatavuus	4	4	1	2

FMOD Studio ja Wwise osoittautuvat vertailun monipuolisimmiksi äänimoottoreiksi. Niiden välistä suhdetta tasapainottaa Wwisen tuki MIDI-tiedostoille kun taas FMOD Studio on käyttäjäystävällisempi. Fabric vaikuttaa luontevalta valinnalta, kun Unityn ääniominaisuudet riittävät, mutta niiden käyttöä halutaan laajentaa. Miles Sound System sopii ennemmin ohjelmointiluontoiselle äänisuunnittelijalle.

Äänimoottoreista kaikki tukevat 3D-ääniä ja tehokkaan äänityöskentelyn kannalta reaaliaikaista miksausta sekä profilointia. Laajennettavuuden kannalta tuki ulkopuolisille liitännäisille ja miksereille löytyy FMODista ja Wwisesta. Tuki ulkopuoliselle liittännäisille tulee tärkeäksi siinä vaiheessa, kun äänimoottoreiden omat liitännäiset eivät enää vastaa äänisuunnittelijan

vaatimuksia ja tuen puute mikserille hidastaa reaaliaikaista miksausta. MIDI-tiedostoja on käytetty tietokonepeleissä 1980-luvulta asti ja se on edelleen tehokas tapa saada pakattua äänidataa pieneen kokoon (Queststudios.com n.d.). Tuki MIDI-tiedostoille löytyvät Wwisesta ja Miles Sound Systemistä.

Dokumentaation saatavuus perustuu valmistajien sivuilta löytyviin oppaisiin, materiaaliin ja tietopankkeihin. Dokumentaation saatavuutta lisää myös internetin keskustelupalstat ja äänimoottoreiden käyttäjien itse tekemät tutoriaalit.

Käyttäjäystävällisyys osoittautui parhaimmaksi FMOD Studiossa. Wwisen käyttökokemus ei kuitenkaan jää kauaksi FMODin käyttökokemuksesta, johtuen lähinnä enemmän näytettävän informaation määrästä ruudulla, joka tekee siitä hieman hitaammin luettavan. Fabric sulautuu käyttöliittymältään yhteen Unityn kanssa, joten Unityä käyttävälle Fabric on suhteellisen helppoa käyttää. Fabricista kuitenkin puuttuu oma selkeä työympäristönsä. Miles Sound System on käyttöliittymältään yksinkertainen, mutta jää ulkoasultaan ja toimivuudeltaan kauaksi äänisuunnitteluun kaivatusta tehokkaasta ympäristöstä.

Taulukossa 2 edellä esiteltyjen äänimoottorien kehittäjät lukuun ottamatta Miles Sound Systemiä ja Fabricia ovat myös tehneet pelimoottorin ja väliohjelmiston välille yhteensopivuusliitännäisen, jonka tarkoitus on helpottaa pelimoottorin ja väliohjelmiston välistä kommunikointia ja käytettävyyttä.

Taulukko 2. Äänimoottoreiden yhteensopivuus pelimoottoreihin.

	Unity	**Unreal Engine 4**	**Unreal Engine 3**	**CryEngine**	**Marmalade**
FMOD Studio	1	1	-	-	-
Wwise	1	1	1	2	1
Miles Sound System	-	-	-	-	-
Fabric	1	-	-	-	-

(- = Ei yhteensopivuutta, 1 = Yhteensopiva, 2 = Pitää erikseen kysyä valmistajalta lisätietoa)

Wwise erottuu taulukon 2 vertailusta selvästi parhaiten pelimoottoreiden yhteensopivuutta tukevaksi, joka on Audiokineticin valttikortti äänimoottoria valitessa. FMOD Studio tarjoaa tukea kahdelle taulukon suurimmista pelimoottoreista, mutta esimerkiksi Unreal Enginen vanhemmille versioille ei ole tukea. Fabric taas käyttää Unity-pelimoottorin ominaisuuksia, eikä se ole laajentanut tukeaan muille pelimoottoreille. Miles Sound System toimii täysin omana sovellusohjelmointirajapintana ja sitä pitää käyttää itsenäisesti kunkin pelimoottorin kohdalla. Joidenkin pelimoottoreiden yhteensopivuusliitännäisen käyttö vaatii erikseen lisenssin, jonka käyttöoikeuden äänimoottorin valmistaja voi antaa käyttäjälle erikseen.

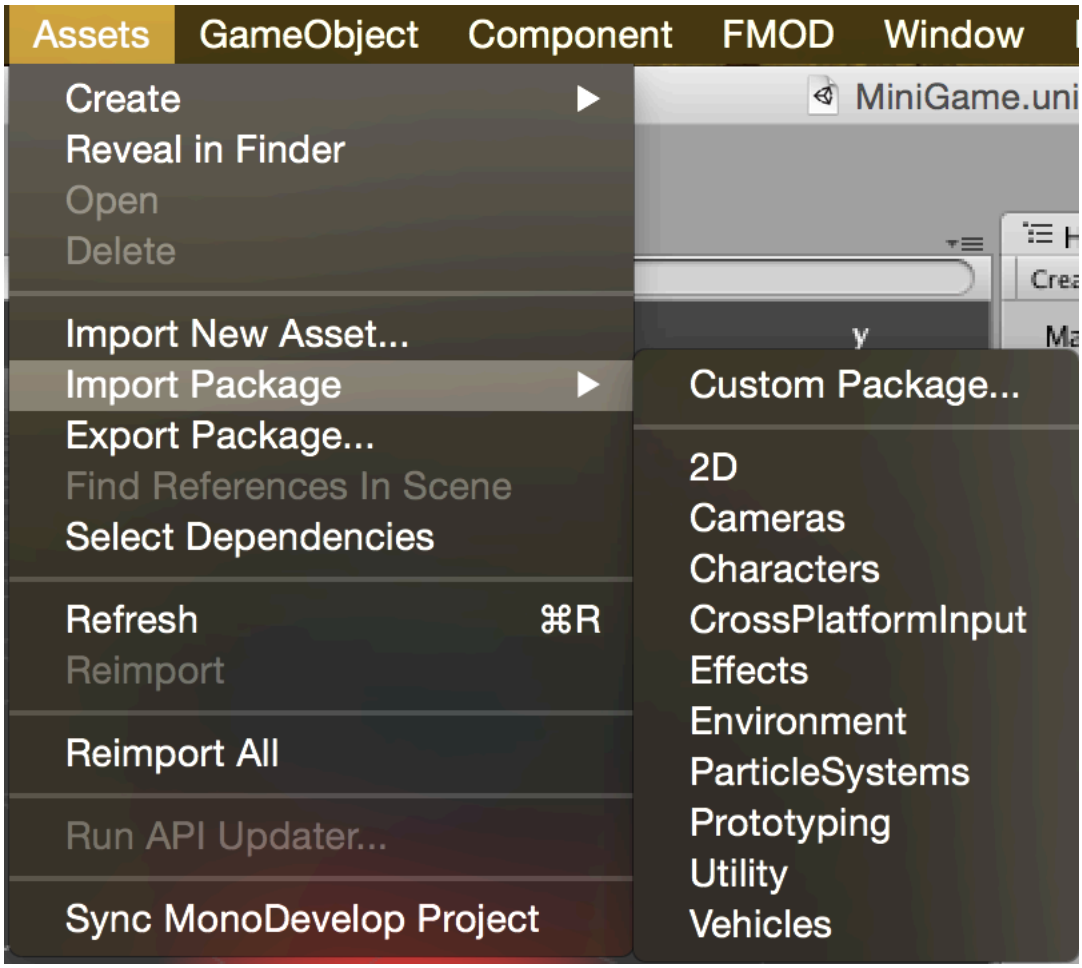
3.4 Äänimoottorin valinta

Äänimoottoreista kahdesta vahvimmasta ehdokkaasta FMOD Studiosta ja Wwisestä, hiuksenhienolla erolla ensimmäinen otettiin käyttöön sen helppokäyttöisyyden, yhteensopivuuden ja lisenssin perusteella. Wwisen käyttölisenssin 200 assetin rajoitus tuntui riittämättömältä ja toisaalta taas FMODin kaupallisen käyttölisenssin 100 000 dollarin tuotantokuluraja liian kaukaiselta saavutettavaksi. FMOD Studion käyttöliittymä tuntui heti alkuun myös enemmän kevyemmältä käyttää ja riittävältä käyttötarkoituksiin. Epäilystä ei kuitenkaan ole, ettei Wwise pystyisi pidemmällä käyttökokemuksella yhtä tehokkaaseen työskentelyyn kuin FMOD Studio. MIDI-tiedoston tuen puute FMOD Studiossa ei ole este, koska ääntä kuitenkin voidaan kompressoida pienempään kokoon äänipankkia rakentaessa.

4 FMOD STUDION YHTEISKÄYTTÖ UNITYN KANSSA

4.1 Integrointi

FMODin integrointi Unity-pelimoottoriin vaatii mukautetun **asset**-paketin (engl. custom package), joka voidaan ladata FMODin sivuilta. Mukautetut paketit ovat tiedostopäätteeltään .**unitypackage**-muotoisia, ja ne tuodaan Unityyn ohjelmassa, jonka jälkeen yleensä valitaan tuotavaksi kaikki pakettiin liittyvät tiedostot. (Kuva 8.)



Kuva 8. Mukautetun paketin tuominen Unityyn.

Paketin tuomisen jälkeen Unityn valikkoon ilmestyy oma valikkonsa FMODille, jonka kautta voidaan tuoda FMOD Studion luomat äänipankit Unity-projektiin ja

jatkossa kun äänipankkeja päivitetään, saman valikon kautta voidaan myös päivittää Unity-projekti ajan tasalle.

Ensimmäistä kertaa kun äänipankit tuodaan Unity-projektiin, tarvitsee ohjelmalle osoittaa oikea kansiopolku, johon äänipankit ovat rakennettuna. FMOD Studio rakentaa äänipankit Builds-kansioon, josta ne voidaan tuoda Unity-projektiin. Äänipankki pitää päivittää uudelleentuomisen jälkeen.

Äänitapahtumat saadaan toimintaan sijoittamalla kuuntelija eli **FMOD_Listener-**komentosarja kiinni haluttuun peliobjektiin. Yleensä kuuntelija sijoitetaan kiinni pelattavaan hahmoon tai kameraan riippuen käyttötarkoituksesta ja halutusta vaikutuksesta kuuntelijaan. Ensimmäisestä persoonasta tai sen takaa kuvattuna on tyypillistä liittää kuuntelija kiinni pelattavaan hahmoon, kun taas kolmannesta persoonasta ylhäältä kuvattuna kuuntelija liitetään kameraan. (Fmod.org 2015e.)

4.2 Äänitapahtumien toisto

Äänitapahtumien toisto tapahtuu joko FMODin pakettiin sisältyvällä komentosarjalla tai itse ohjelmoimalla. Pakettiin sisältyvä lähetin eli **FMOD_StudioEventEmitter**-komentosarja voidaan liittää haluttuun peliobjektiin kuuntelijan tavoin, mutta lähetin voidaan laittaa heti toistamaan ääntä pelin alkaessa tai halutulla hetkellä. Valmiina tuleva komentosarja on lähinnä tarkoitettu pitkäkestoisten äänitapahtumien toistoon. (Fmod.org 2015e.)

4.3 Äänityöskentely

Pelin äänimaailman rakentaminen vuorovaikutteiseksi ja mielenkiintoiseksi vaatii hieman tutustumista ohjelmointiin, jotta äänitapahtumat saadaan toimimaan halutulla tavalla muun peliympäristössä tapahtuvan kanssakäymisen myötä. Kanssakäymistä voi esimerkiksi olla adaptiivisen musiikin reagointi pelin tapahtumiin, vihollisten kanssa taistelu, vuorovaikutus esineiden ja asioiden kanssa. FMOD tarjoaa dokumentaatiota Unity-yhteiskäytön kanssa niukasti,

mutta FMOD Studion käyttäjäkunnasta löytyy paljon itsetehtyjä tutoriaaleja ja ohjemateriaalia.

Seuraavissa luvuissa on esitetty viisi esimerkkiä äänitapahtumien käytöstä yksinkertaisista haastavampiin toimintamalleihin. Esimerkit on kirjoitettu C#-ohjelmointikielellä.

4.3.1 Esimerkki 1: PlayOneShot

PlayOneShot-esimerkissä äänitapahtuma toistetaan yksinkertaisesti koodissa. Se on tarkoitettu kertaluontoisten äänien toistamiseen, joita ei tarvitse erikseen ohjelmoida lopettamaan toistoa. (Fmod.org 2015e.)

```
FMOD_StudioSystem.instance.PlayOneShot("event:/Sounds/Oneshot",transform.position)
```

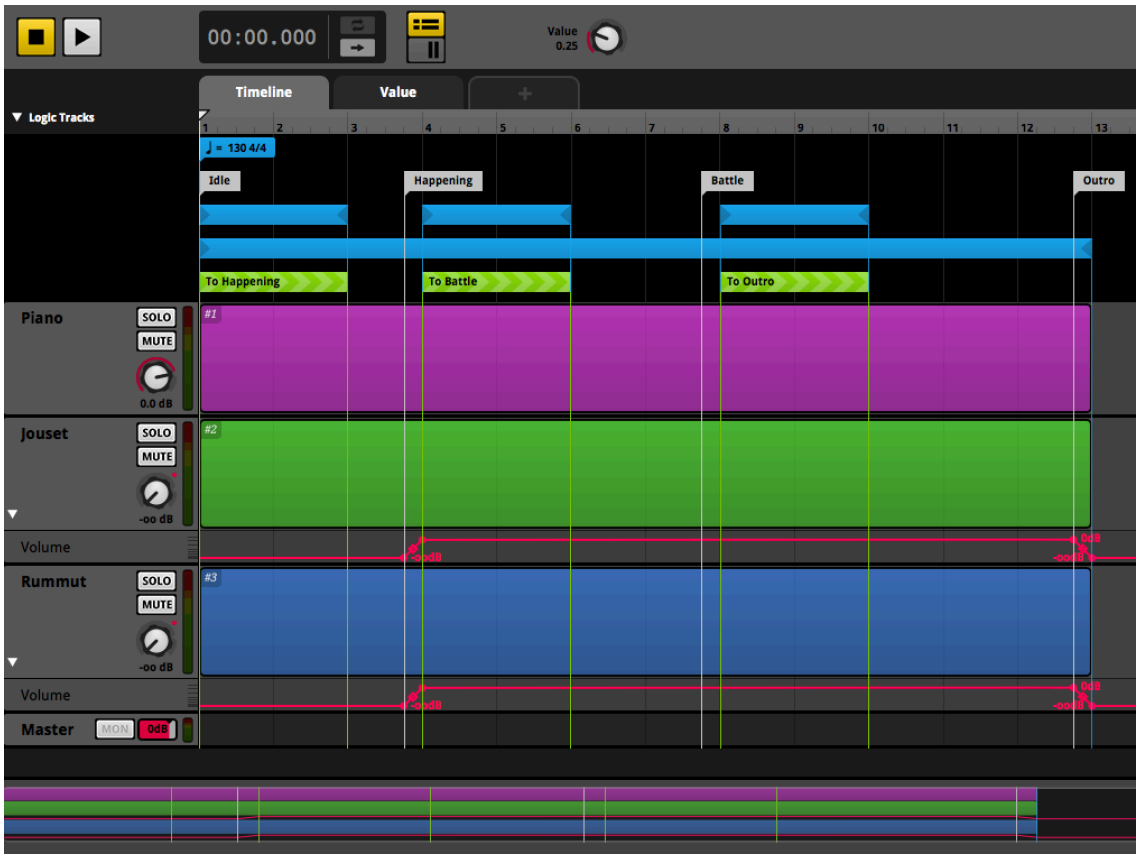
Koodipätkässä määritetään suluissa kaksi arvoa: hakemistopolku, josta äänitapahtuma löytyy, ja sijainti, jossa äänitapahtuma toistetaan. Koodissa **transform.position** toistaa äänen juuri siinä kohtaa, jossa peliobjekti ja johon komentosarja on liitetty kiinni.

4.3.2 Esimerkki 2: Adaptiivinen musiikki

Adaptiivinen eli ei-lineaarinen musiikki on vuorovaikutteista musiikkia, joka käyttäytyy esimerkiksi tilanteen tai ajan mukaan. Toisin kuin elokuvissa, joissa musiikki on valmiiksi sävelletty lineaarisesti, peleissä musiikki toistetaan usein epälineaarisesti kuten juonikin. Peleissä usein pelaaja määrittää juonen kulun, joten musiikin täytyy seurata juonen etenemistä. Adaptiivinen musiikki toteutetaan FMOD Studiossa usein siirtymäkohdilla ja silmukkaloopeilla.

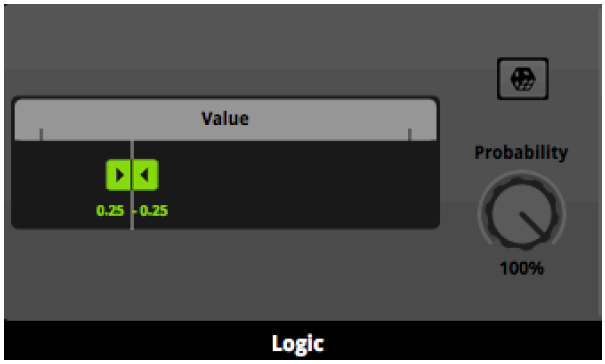
Esimerkissä adaptiivinen musiikki on toteutettu kolmella neljän tahdin kestävällä saumattomasti uudestaan toistuvalla musiikkileikkeellä. Ensimmäinen äänileike on rauhallinen pianolla toistettu tunnelmointi. Toinen äänileike sisältää nopeasti soitettuja jousisoittimia ja kolmas äänileike tahdikkaita patarumpuja. Musiikki on tarkoitus toistaa siten, että ensimmäinen äänileike soi, kun pelattava hahmo on

normaalissa tilassa. Toinen äänileike tulee mukaan, kun pelin alkaa tapahtua ja odotettavissa on esimerkiksi taistelu tai pakotilanne. Taistelun ja pakotilanteen syntyessä mukaan tulee myös kolmas äänileike. Tilanteen purkautuessa musiikki palautuu takaisin ensimmäisen äänileikkeen varaan. (Kuva 9.)



Kuva 9. Adaptiivinen musiikki FMOD Studiossa.

Kuvassa 9 olevat vihreät palkit ovat siirtymäkohtia, joista siirrytään niille asetetuille merkitsijöille, kunnes siirtymäkohdat saavat tietyn arvon pelistä. Siirtymäkohtien arvoa säätelee kuvassa näkyvä Value-muuttuja. Ensimmäisen siirtymäkohdan **To Happening** siirtymäarvoksi on määritelty logiikassa 0.25, jolloin aikakursori siirtyy **Happening**-merkitsijän alkuun ja aloittaa seuraavan siirtymäkohdan toiston silmukkaloopin sisällä. Näin toimitaan jokaisella siirtymäkohdalla, kunnes saavutetaan loppu tai FMOD saa käskyn lopettaa toisto. (Kuva 10.)



Kuva 10. Logiikan määrittely.

Unityssä äänitapahtuma haetaan ja **Value**-muuttujaa kutsutaan sekä annetaan arvo muuttujalle tilanteen muuttuessa.

```
FMOD.Studio.EventInstance Music;

FMOD.Studio.ParameterInstance Value;

void Start()

{

//äänitapahtuma haetaan

            Music =
FMOD_StudioSystem.instance.GetEvent("event:/Music/Music");

//äänitapahtuma aloitetaan

            Music.start():

//kutsutaan äänitapahtumasta Value-muuttuja

            Music.getParameter("Value", out Value);

}
```

Pelin tunnelman tiivistyessä **Value**-muuttujalle annetaan arvo, joka käskee FMODin siirtyä seuraavaan **Happening**-merkitsijän alkuun.

```
//äänitapahtuman Value-muuttujalle annetaan arvo 0.25

Value.setValue(0.25f);
```

Samalla tavalla edetään pelissä eteenpäin **Battle**- ja **Outro**-merkitsijöiden alkuun. **To Battle**-siirtymäalueelle on annettu arvo 0.5 ja **To Outro**-siirtymäalueelle 0.75. Äänitapahtuman toisto voidaan halutessa lopettaa kokonaan, ja on hyvä varmistaa, että äänitapahtuma ei jää muistiin viemään tilaa.

```
//äänitapahtuma lopetetaan
```

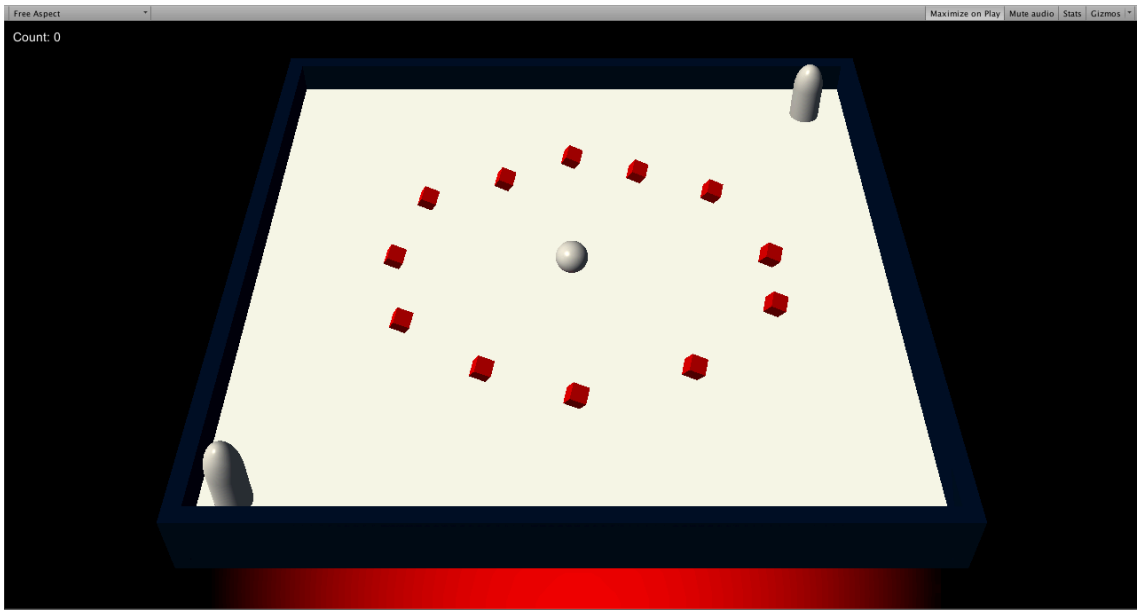
```
Music.stop():

//äänitapahtuma vapautetaan muistista

Music.release():
```

4.3.3 Esimerkki 4: Pallon vieritys, törmäykset ja etäisyys

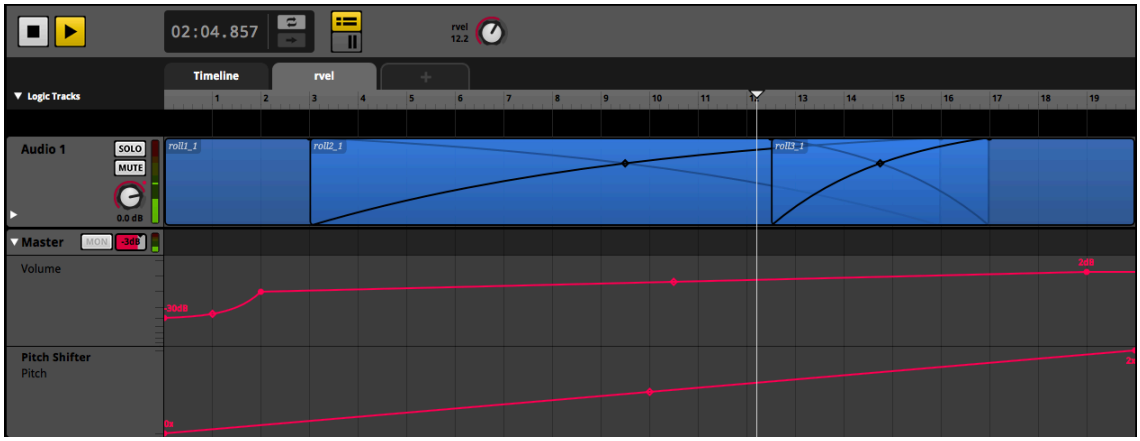
Esimerkissä 4 käydään läpi, miten pienessä pallopelissä pallon vierityksen nopeus, törmäykset seiniin ja etäisyys peliobjekteista saadaan vuorovaikutteiseksi äänen kanssa. Pelin ajatuksena on yksinkertaisesti kerätä neliön muotoiselta alueelta kerättäviä kappaleita. Mitä nopeammin pallo pyörii, sitä voimakkaampi ääni ja korkeampi sävelkorkeus pallosta lähtevät. (Kuva 11.)



Kuva 11. Kuvankaappaus pallopelistä.

FMOD Studiossa luodaan normaalisti **rolling**-äänitapahtuma pallon vieritykselle, ja tapahtumalle luodaan itsemääritetty **rvel-**muuttuja. Muuttujaa luotaessa tulee huomata Unityssä käytettävä mittayksikkö ja pallon nopeus: neliön yhden särmän pituus on 20 yksikköä, joten muuttujan maksimiarvo on suositeltavaa asettaa 20:een asti. Toisin kuin muissa esimerkeissä äänileikkeet lisätään muuttujan sisälle epälineaariselle aikajanalle eikä normaalille aikajanalle. Pallon vieritykseen käytetään esimerkissä kolme erilaista metallikuulan vieritystä muistuttavaa ääntä, jotka ristihäivytetään (engl.

crossfading) keskenään. Tapahtumalle myös lisätään äänenvoimakkuus- ja sävelkorkeusautomaatioraidat. Äänenvoimakkuus alkaa -30 dB:stä ja saavuttaa 2 dB:n tason. Sävelkorkeus alkaa nollakertaisesta puolisävelaskeleesta ja saavuttaa kaksinkertaisen puolisävelaskeleen tason arvon 20 kohdalla. (Kuva 12.)



Kuva 12. Äänileikkeiden ristihäivytys ja automaatioraidat.

Komentosarjassa, joka on liitetty ohjattavaan palloon, kutsutaan **rolling-**äänitapahtumaa ja **rvel-**muuttujaa sekä aloitetaan äänitapahtuma.

```
void Start ()
{

    //rolling-äänitapahtuma ja sen muuttuja kutsutaan
    rolling = FMOD_StudioSystem.instance.GetEvent ("event:/rolling");
    rolling.getParameter ("rvel", out rvel);
    }

    //äänitapahtuma aloitetaan
    rolling.start ();
}
```

Palloa liikutellaan nuolinäppäimillä ja sen vauhti sekä kiihtyvyys määräytyvät ennalta määritetyllä nopeudella **speed** ja **deltaTime**-arvolla, joka pyörittää palloa tasaista vauhtia ruudunpäivitysnopeudesta huolimatta **FixedUpdate-**funktiossa (Tiihonen 2013, 26). **pVel-**muuttuja saa arvonsa pallon liikkeen

voimakkuudesta, joka lähetetään kullakin kehyksellä äänitapahtuman **rvel-muuttujalle**.

```
void FixedUpdate () {

    //haetaan arvot nuolinäppäimille
    float moveHorizontal = Input.GetAxis ("Horizontal”);
    float moveVertical = Input.GetAxis ("Vertical");

    //luodaan pallolle like ja voima
    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
    GetComponent<Rigidbody>().AddForce (movement * speed * Time.deltaTime);

    //lähetetään äänitapahtumalle pallon liikeen voima
    pVel = GetComponent<Rigidbody>().velocity.magnitude;
    rvel.setValue (pVel * 2);
}
```

Lopputuloksena **rvel-**muuttuja toistaa FMOD Studiossa pallon liikkeen voimakkuuden arvolla äänitapahtumaa.

Pallon liikkeen voimakkuutta voidaan käyttää myös hyödyksi pelikentän sivuseinillä, jotka havaitsevat kuinka suurella nopeudella pallo törmää seiniin. Äänitapahtumassa **impact** on liitettynä viisi äänileikettä eri äänenkorkeuksilla ja äänenvoimakkuuksilla. Aina kun pallo törmää seinään, äänitapahtuma saa nopeudesta aiheutuneen törmäyksen arvon ja toistaa sen arvon kohdalla olevan äänileikkeen.

```
void OnTriggerEnter(Collider other) {


//tarkistetaan, saako törmäytin osuman tagilla “Wall”
        if (other.gameObject.CompareTag ("Wall"))
        {

//kutsutaan impact-äänitapahtumaa ja sen muuttujaa
            impact = FMOD_StudioSystem.instance.GetEvent("event:/impact");
            impact.getParameter ("sndVel", out impactvel);
        }
//lähetetään impact-osuman arvo äänitapahtumalle
            impactvel.setValue(pVel);
            impact.start();
        }
```

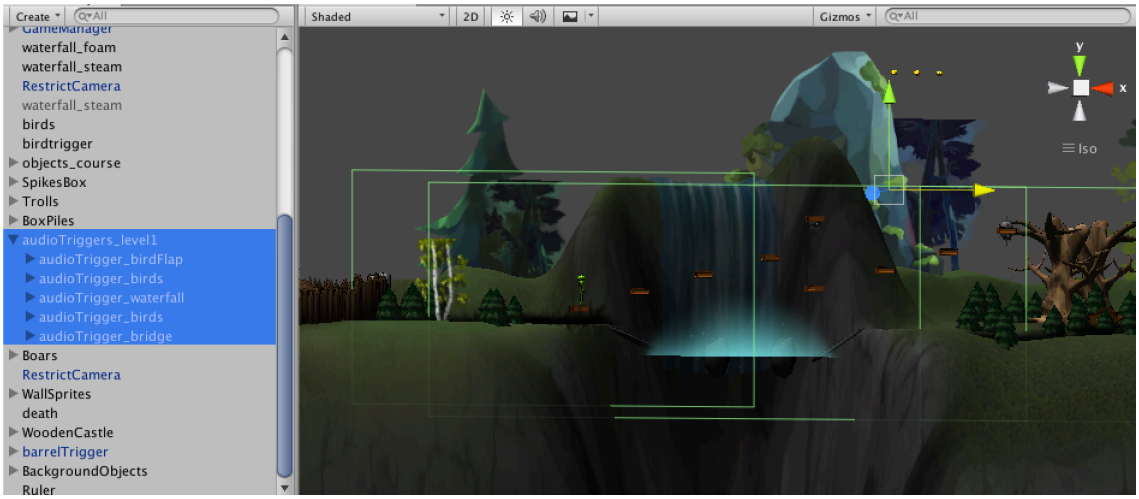
4.3.4 Esimerkki 4: Äänikytkimet

Äänikytkimillä voidaan aloittaa jokin äänitapahtuma, kun pelaaja astuu kytkimen alueen sisään. Äänitapahtuma myös lopetetaan pelaajan astuessa kytkimen alueelta ulos. Äänikytkimet ovat oivallinen tapa säästää pelin suorituskykyä 2D-peleissä, koska tarvittavat äänitapahtumat toistetaan vain silloin, kun niitä tarvitaan. Äänikytkimiä voidaan käyttää esimerkiksi ohimenevissä tilanteissa, kuten pelaajan ohittaessa vesiputouksen tai kohdatessaan esteen. (Kuva 13.)

Kytkeytyvää tapahtumaa varten Unityssä luodaan tyhjä peliobjekti, jonka sisälle luodaan laatikkotörmäytin (engl. box collider*),* joka määritetään halutun kokoiseksi. Laatikkotörmäyttimet ovat kuution tai suorakulmaisen särmiön muotoisia alueita, jonka tahkot toimivat kytkiminä *(*engl. trigger*)* kun niihin törmätään. Kytkiminä toimimista varten törmäyttimestä pitää kytkeä päälle **Is Trigger** -valinta, jotta se toimii halutulla tavalla.

Laatikkotörmäyttimen lapsiobjektiksi tyhjän peliobjektin sisään tarvitaan FMODin äänilähetin, joka voidaan sijoittaa haluttuun sijaintiin, josta äänen halutaan kuuluvan ympäristössä.

Kuvassa näkyvät ääriviivat ovat laatikkotörmäyttimiä, jotka toimivat äänikytkiminä. Pelaajan saapuessaan laatikkotörmäyttimiin äänitapahtuma alkaa ja niistä poistuessaan se loppuu.



Kuva 13. Äänikytkimiä Unityssä.

Äänikytkimen toteutettavaksi **audioTrigger**-laatikkotörmäyttimeen tarvitaan komentosarja, joka hakee **audioTrigger_position**-lapsiobjektista lähettimen äänitapahtuman toistamiseen ja samalla komentosarjassa määritellään miten äänikytkin käyttäytyy.

Äänilähetin pitää komentosarjassa kutsua erikseen. Siinä luodaan myös muuttuja peliobjektin hakemiseksi, jos äänilähetin on muualla kuin kiinni itse pääobjektissa. Boolean-muuttujaa tarvitaan tarkistamaan, halutaanko äänen loppuvan äänikytkimestä poistuessa.

```
    FMOD_StudioEventEmitter emitter;
    public GameObject audioTrigger;
    public bool dontStopOnExit;
```

Tarkistetaan onko **audioTrigger_position**-peliobjekti ja sen sisällä oleva lähetin olemassa. Jos ei ole, sille täytyy määrittää jo erikseen olemassa oleva objekti, joka sisältää lähettimen.

```
    void Start () {

//tarkistetaan, onko lähetin olemassa ja haetaan lähetin
        if (audioTrigger == null) {
     emitter = transform.Find ("audioTrigger_position").gameObject.GetComponent<FMOD_Stu
dioEventEmitter> ();
        } else {
                emitter = audioTrigger.GetComponent<FMOD_StudioEventEmitter> ();
```

```
        }
        }
```

Jotta äänitapahtuma voidaan toistaa, tarkistetaan että törmäyttimen sisään tuleva objekti on pelaaja. Pelaajaobjekti on merkitty **Player**-tagilla**,** joten kaikki peliobjektit, joilla on sama tagi, aiheuttavat äänitapahtuman toistumisen. Päinvastoin objekteilla, joilla ei ole samaa tagia, ei vaikuta äänikytkimen käyttäytymiseen. Tällä tavalla saadaan varmistettua, että äänikytkin käyttäytyy, kuten on haluttu.

```
//tarkistetaan, saako törmäytin osuman tagilla Player
    void OnTriggerEnter2D (Collider2D other ) {
        if (other.CompareTag("Player")) {

//käynnistetään lähetin
            emitter.Play ();
        }
    }
```

Kun pelaaja poistuu törmäyttimestä, äänitapahtuma lopetetaan, jos toisin ei ole määritetty

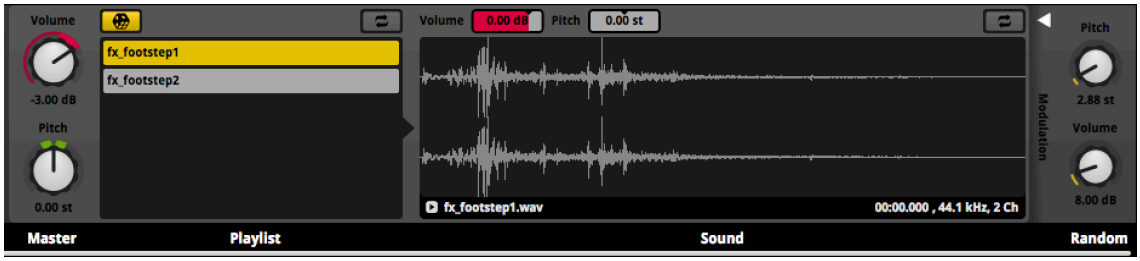
```
//törmäyttimestä poistuessa annetaan äänitapahtuman jatkua tai lähetin kytketään pois
päältä
    void OnTriggerExit2D (Collider2D other ) {
        if (dontStopOnExit == false) {
                emitter.Stop ();
            }
    }
}
```

4.3.5 Esimerkki 5: Äänien lisääminen animaatioihin funktiokutsuilla

Peli sisältää usein animoituja hahmoja ja asioita, joihin liittyvät myös äänet. Tyypillisiä hahmojen tuottamia asioita ovat kävelyäänet ja erilaisten taisteluiskujen äänet. Tässä esimerkissä käydään läpi askeläänien lisääminen pelaajahahmolle.

Pelaajahahmosta riippuen sille on tehty animaatiot usein hypylle, kävelylle ja juoksulle. Kävelyääni on tyypillisesti toteutettu yhdellä äänellä, joka toistuu aina,

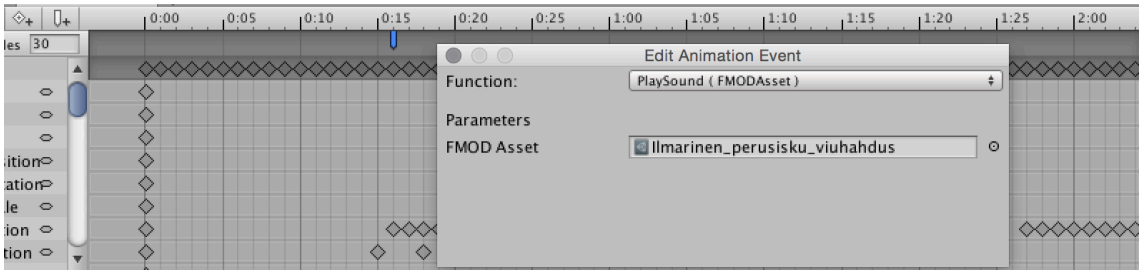
kun pelaajahahmon jalka osuu maahan. Jotta kävelyäänistä saadaan monimuotoisemman kuuloisia, tarvitaan siihen useampi ääni ja sävelkorkeuden modulointia. FMOD Studiossa tämä voidaan toteuttaa Multi Sound -äänitapahtumalla ja äänen sävelkorkeutta satunnaisuutta sekä äänenvoimakkuutta moduloimalla. (Kuva 14.)



Kuva 14. Multi Sound -äänitapahtuman muokkaus.

Multi Sound -äänitapahtumassa voidaan kytkeä satunnaisuusvalinta päälle, jolloin äänitapahtumaa valitsee satunnaisesti toistettavan äänen. Kuvassa on myös äänitapahtumassa äänenvoimakkuudelle (engl. volume) ja äänenkorkeudelle (engl. pitch) säädetty satunnaisuutta. Satunnaisuus määritetään äänenvoimakkuudelle desibeleissä ja äänenkorkeus puolisävelaskelissa (engl. semitones).

Kun askeläänien ovat valmiina, ne tarvitsee liittää pelaajahahmon animaatioon. Animaatioihin voidaan liittää Unityn animaatioeditorissa funktiokutsuja, jotka toistavat komentosarjoja. (Kuva 15.)



Kuva 15. Funktiokutsun lisääminen Unityn animaatioeditorissa.

Tapahtumaa varten tarvittava komentosarja on yksinkertainen, joka toistaa äänen. Äänitapahtumaa varten luodaan julkinen **FMODAsset**, johon voidaan valita haluttu äänitapahtuma.

```
public class SoundEvent : MonoBehaviour {
    public FMODAsset A_EventSound;

public void PlaySound (FMODAsset A_EventSound) {

//toistetaan äänitapahtuma
            FMOD_StudioSystem.instance.PlayOneShot(A_EventSound, transform.position);
    }
}
```

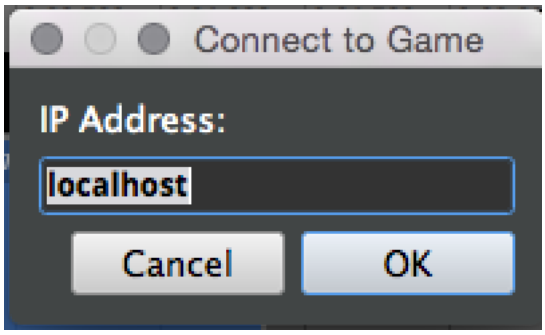
Animaatioeditorissa valitaan animaatiosta kohdat, joihin äänitapahtuma halutaan asettaa. Kävelyanimaatio on yleensä vähintään kahden askeleen kestävä, joten molemmille askelille pitää lisätä tapahtuma. Askeläänіä sisältävän äänitapahtuma lisätään animaatioeditorissa oikeisiin avainkehyksiin (*keyframe*), joissa hahmon jalka osuu maahan. Tämän jälkeen tapahtuman sisältä valitaan äänitapahtuman funktiokutsu, johon valitaan oikea äänitapahtuma hakemistosta.

4.4 Reaaliaikainen miksaus

Tasapainoisen äänimaailman toteuttamiseksi FMOD Studio tukee reaaliaikaista miksausta Unityn kanssa. Tämä säästää huomattavasti äänisuunnittelijan aikaa etsiä oikeaa tasapainoa pelissä tapahtuvien äänien välillä ja muokata eri äänien käyttäytymistä. Äänisuunnittelijan ei tarvitse ominaisuuden johdosta enää kytkeä peliä joka kerta uudelleen päälle uusilla ääniarvoilla. FMOD Studio yhdistää peliin verkon välityksellä IP-osoitteen avulla. Reaaliaikainen miksaus saadaan päälle lisäämällä parametri *FMOD_LIVEUPDATE* Unityn Player-asetuksen *Scripting Define Symbols* -tekstilaatikkoon. Toinen tapa on lisätä parametri FMOD_StudioSystem.cs-komentosarjan alkuun rivillä *#define FMOD_LIVEUPDATE*. (Fmod.org 2015e.)

Tämän jälkeen Unityssa aloitetaan peli-istunto ja FMOD Studiosta valitaan ylävalikosta *File -> Connect to Game…* ja valitaan oikea IP-osoite. Jos peli on

samassa koneessa kuin FMOD Studio, voidaan osoitteeksi valita 127.0.0.1 tai **localhost**. (Kuva 16.) (Fmod.org 2015.)

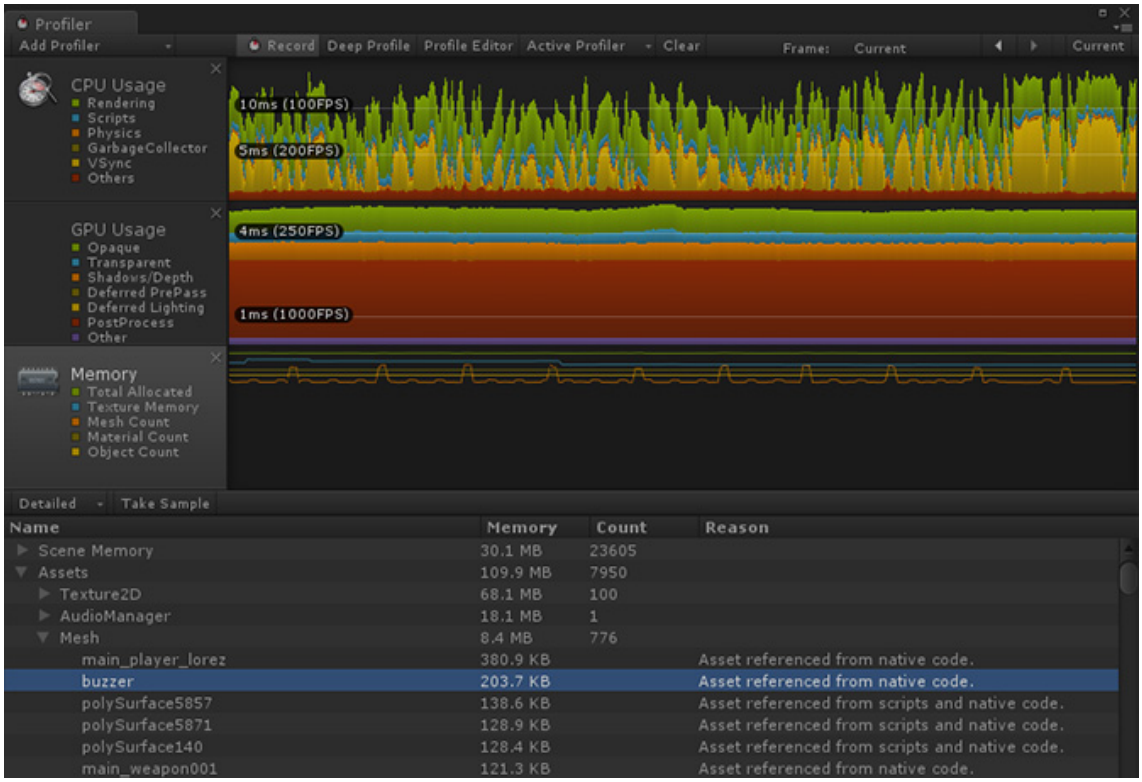


Kuva 16. FMOD Studion yhdistäminen peliin.

Yhdistämisen jälkeen käyttäjä on vapaa miksaamaan reaaliaikaisesti pelissä tapahtuvia äänitapahtumia. Kun peli-istunto lopetetaan, asetukset jäävät voimaan FMOD Studioon. Uudet asetukset tulee tallentaa ja rakentaa uudelleen Unityyn, jotta ne ovat seuraavassa peli-istunnossa voimassa.

4.5 Suorituskyvyn seuranta ja nauhoitus

FMOD Studion Unityyn pelissä tapahtuvien äänitapahtumien kuormitusta voidaan seurata esimerkiksi suorittimille ja muistille FMOD Studion Profilerin avulla reaaliaikaisesti ja nauhoittaa myöhempää käsittelyä varten. Profiler myös näyttää, kuinka paljon jokainen äänitapahtuma yksitellen vie suorituskykyä ja muistia sekä kuinka monta äänitapahtumaa päällekkäin toistuu. (Kuva 17.) (Fmod.org 2015c.)



Kuva 17. FMOD Studio Profiler.

Profilerin voidaan myös ajaa nauhoitettu peli-istunto uusilla äänillä, jolloin kuullaan ero eri äänitapahtumien välillä. Tämä nopeuttaa äänityöskentyä huomattavasti.

5 YHTEENVETO

Tehokkaan ja käytännöllisen äänimoottorien valitseminen ja vertailu muiden äänimoottoreiden kanssa auttoi löytämään sopivan äänimoottorin pelien kehittämiseen. Äänimoottorin graafinen käyttöliittymä ja käyttöystävällisyys auttoivat nopeamman pääsyn tavoitteeseen, jolta äänimaailman haluttiin pelissä kuulostavan. Lopputuloksena saadut esimerkit auttoivat ymmärtämään äänimoottorin käyttöä osana äänisuunnittelua ja oppimaan uusia työtapoja, joita voi hyvin hyödyntää tulevaisuudessa.

Esimerkkejä olisi voinut hyödyntää osana suurempaa kokonaisuutta, mutta yksinkertaistettuna omina esimerkkeinään ne toivat paremmin esille toiminnallisuuden selittäen lukijalle niiden tavoitellun lopputuloksen. Eräänä mahdollisuutena olisi ollut tarkastella esimerkkejä myös toisella äänimoottorilla tai esimerkiksi Unityn omilla äänityökaluilla.

Opinnäytetyötä tehdessäni opin paljon äänimoottoreiden perustoimintatavasta, kuten niiden eroistakin. Osana äänimoottorin implementoimista pelinkehitysympäristöön, huomasin kuinka tärkeätä on omaksua ohjelmointitaidon perusteet, joiden ansiosta äänisuunnittelusta saa monipuolisempaa. Aluksi oli hankalaa omaksua epälineaarinen työskentely-ympäristö, joka sittemmin osoittautui erittäin tehokkaaksi.

Äänimoottoreiden käyttö väliohjelmistona on erittäin suositeltavaa osana pelinkehitystä, sillä se säästää äänisuunnittelijan aikaa ja vaivaa monessa suhteessa. Väliohjelmistot myös taipuvat moneen eri käyttötarkoitukseen ja antavat äänisuunnittelijalle melkein loputtomasti tapoja, joilla äänisuunnittelua voi kehittää. Äänisuunnittelussa kannattaa ehdottomasti silti tutustua muiden äänimoottoreiden käyttöön, koska eri äänimoottorit voivat toimia eri projekteissa paremmin kuin toiset.

LÄHTEET

Audiokinetic 2015a. About Audiokinetic. Viitattu 19.4.2015 https://www.audiokinetic.com/about/

Audiokinetic 2015b. Exploring the Wwise Interface. Käyttöohje. Viitattu 19.5.2015 https://www.audiokinetic.com/library/2015.1_5363/?source=Help&id=exploring_wwise_interface

Audiokinetic 2015c. Integrating Audio in Your Game: The Wwise Approach. Käyttöohje. Viitattu 5.5.2015 http://www.audiokinetic.com/download/documents/TheWwiseApproach.pdf

Audiokinetic 2014a. Wwise 2014.1 Now Available. Viitattu 1.4.2015 https://www.audiokinetic.com/about/news/wwise-2014.1-now-available/

Audiokinetic 2014b. Wwise 2014.1 Public Beta 2 is now available to all! Viitattu 23.4.2015 https://www.audiokinetic.com/about/news/wwise-2014.1-public-beta-1-is-now-available-to-all/

Audiokinetic 2015d. Wwise Product Pricing: Limited. Viitattu 10.5.2015 https://www.audiokinetic.com/licensing/pricing/#limited

Audiokinetic 20.7.2010. Wwise Tutorial 22 – Dynamic Mixing Using States. tiedosto. Viitattu 29.4.2015 https://www.youtube.com/watch?v=a77W-THkPMY

Brandon, A. 2007. Audio Middleware: The Essential Link From Studio to Game Design. Viitattu 3.3.2015 http://www.emusician.com/how-to/1334/masterclass-game-audio-middleware/48158

Collins K. & Kapralos B. & Tessler H. 2014. The Oxford Handbook of Interactive Audio. Oxford Unity Press.

Develop-Online 2012. Develop looks at the new Miles Sound System 9. Viitattu 17.5.2015 http://www.develop-online.net/analysis/key-release-miles-sound-system-9/0117301

Electronic Musician. Horowitz S. & Looney S. 2014. Masterclass: Using Game Audio Middleware. Viitattu 3.3.2015 http://www.emusician.com/how-to/1334/masterclass-game-audio-middleware/4815

FMOD 2015a. About Firelight Technologies. Viitattu 16.3.2015 http://www.fmod.org/about-us/

FMOD 2015b. FMOD Licenses. Viitattu 24.3.2015 http://www.fmod.org/sales/

FMOD 2015c. FMOD Studio. Viitattu 17.3.2015 http://www.fmod.org/products/

FMOD 14.10.2014. FMOD Studio for Real Engine 4 is now available. Viitattu 18.3.2015 http://www.fmod.org/fmod-studio-unreal-engine-4-now-available/

FMOD 2015d. FMOD Studio Plugins. Viitattu 27.3.2015 http://www.fmod.org/products/

FMOD 19.9.2013. FMOD Studio Unity Integration. Viitattu 22.3.2015 http://www.fmod.org/forum/viewtopic.php?f=30&t=16159

FMOD 2015e. FMOD Studio Unity Integration. Viitattu 19.4.2015 http://www.fmod.org/files/UnityDocumentation.pdf

FMODTV 2012. FMOD Studio Tutorial 01 – Interface Introduction. Videotiedosto. Viitattu 19.5.2015 https://www.youtube.com/watch?v=vr6pCpV9mO8&spfreload=10

Gearnuts.org 2015. Mackie Control Universal Pro. Viitattu 7.6.2015 http://www.gearnuts.com/store/detail/MCUpro

Harvey J, Sound Librarian. 2015. FMOD Studio User Manual: FMOD Studio V1.05.14.

Horowitz S. & Looney S. 5.3.2014. The Essential Guide To Game Audio: The Theory and Practice of Sound for Games. CRC Press.

Kutvonen, L. 2004. Väliohjelmistot: Middleware definition 1. Powerpoint-esitys. Viitattu 23.5.2015 http://www.cs.helsinki.fi/u/summanen/opetus/2004/middleware/luennot/1-johdanto.pdf

Meyer B. 9.3.2013. The futureo f the next-gen sound blah blah blah. Viitattu 11.5.2015 http://www.bgleymeyer.com/wp-core/the-future-of-next-gen-sound-blah-blah-blah/

Middleware N.d. What is Middleware?. Palautettu ja viitattu 23.5.2015 http://web.archive.org/web/20120629211518/http://www.middleware.org/whatis.html

Prunotto C. 15.4.2014. An Introduction to FMOD, part 1: The Interface. Viitattu 17.4.2015 https://chrisprunotto.wordpress.com/2014/04/28/an-introduction-to-fmod-part-3/

Queststudios N.d. The History Of PC Game Midi. Viitattu 19.5.2015 http://www.queststudios.com/quest/midi.html

RAD Game Tools 2015. The Miles Sound System. Viitattu 3.4.2015 http://www.radgametools.com/miles.htm

Robinson K. 2014. Ableton Live 9: Create, Produce, Perform. CRC Press.

Roos, S. 2012. Äänimoottorin suunnittelu ja toteutus. Opinnäytetyö. Turun ammattikorkeakoulu, Tietotekniikan koulutusohjelma, sulautetut ohjelmistot. Viitattu 3.3.2015 http://urn.fi/URN:NBN:fi:amk-2014060511985

Snipview.com 2012. Audiokinetic Wwise. Viitattu 7.6.2015 http://www.snipview.com/q/Audiokinetic_Wwise

-Audio 2015a. Why Fabric? Viitattu 4.4.2015 http://www.-audio.co.uk/#!fabric/c1oba

-Audio 2015b. Licensing. Viitattu. 5.4.2015 http://www.-audio.co.uk/#!licensing/c1df1

Unity3D 2015a. Frequently Asked Questions. Viitattu 25.3.2015 http://unity3d.com/unity/faq

Unity3D 2015b. The Leading Global Game Industry Software. Viitattu 25.3.2015 http://unity3d.com/public-relations

Unity3D Store 2015c. Unity Pro. Viitattu 25.3.2015 https://store.unity3d.com/

Tiihonen, P. 2013. Pelinkehitys Unity3D-pelimoottorilla aloittelijoille. Opinnäytetyö. Seinäjoen ammattikorkeakoulu, Tietotekniikan koulutusohjelma, tekniikan yksikkö. Viitattu 24.4.2015 http://urn.fi/URN:NBN:fi:amk-20130

Wikipedia.org 2014. 3D audio effect. Viitattu 23.4.2015 http://en.wikipedia.org/wiki/3D_audio_effect

Wikipedia.org 2015. Audiokinetic Wwise. Viitattu 1.4.2015 http://en.wikipedia.org/wiki/Audiokinetic_Wwise