Patrik Johansson

Dynaamisen musiikin tuotanto peliympäristöön

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

22.11.2021

Tiivistelmä

Tekijä:	Patrik Johansson
Otsikko:	Dynaamisen musiikin tuotanto peliympäristöön
Sivumäärä:	46 sivua
Aika:	22.11.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisovellukset
Ohjaajat:	Lehtori Antti Laiho

Opinnäytetyössä oli tarkoituksena selvittää dynaamisen videopelimusiikin toteutustapoja, pohtia niiden vaikutuksia peliympäristöön ja selvittää prosessia dynaamisen videopelimusiikin luontiin niin sävellyksellisestä kuin teknisestä näkökulmasta.

Tavoitteena oli luoda yksinkertainen, pelattava pelimaailma, jonka sisällä pelaaja pystyy tarkastelemaan erityyppisiä sekä horisontaalisen uudelleen sekvensoinnin että vertikaalisen uudelleen orkesteroinnin keinoja käyttäviä dynaamisen videopelimusiikin toteutuksia. Teknisen toteuttamisen lisäksi projektin esimerkkitoteutuksia varten tuotettiin omat taustamusiikkikappaleensa, jotka tehostavat näiden tiettyjen dynaamisen pelimusiikin tilanteita myös musiikkiteorian näkökulmasta.

Pelimaailman ja pelillisyyden luomiseen sekä yksityiskohtaisempaan skriptaamisen luomiseen käytettiin Unity-pelimoottoria. Äänimoottorina käytettiin FMOD Studiota, jossa on valmiudet integroimiseen Unityn kanssa. Suurin osa dynamiikan pohjustuksen teknisestä toteutuksesta tehtiin käyttäen FMOD:n automaatioraitoja.

Dynaamisen videopelimusiikin avulla äänisuunnittelija kykenee luomaan pelikokemukseen hienovaraisempia ja tehokkaampia musiikillisia hetkiä ja lisäämään musiikin aktiivisuutta pelin äänimaailmassa.

Avainsanat: videopelimusiikki, dynamiikka, adaptiivisuus

Abstract

Author:	Patrik Johansson
Title:	Dynamic Music Production for Game Environments
Number of Pages:	46 pages
Date:	22 November 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Game Applications
Supervisors:	Antti Laiho, Senior Lecturer

The intent of this thesis is to examine the ways to execute dynamic music in video games, inspect their effects on game environments and clarify the process of creating the dynamic videogame music in both compositional and technical standpoints.

The objective was to create a simple, playable game environment where the player can examine different approaches of dynamic videogame music that uses both methods of horizontal re-sequencing and vertical re-orchestration. In addition to the technical execution, a piece of background music was created for each case study that also complemented the specific situations in the terms of music theory.

Game engine Unity was used in creating the game environment, gameplay and elaborate scripting concerning dynamic music. The project's sound engine was FMOD Studio, which has the capability of integrating with Unity. Most of the set up for the technical implementations of the dynamic music tracks was made using FMOD's parameter automation tracks.

With dynamic videogame music the sound designer is capable of creating more subtle and powerful musical moments for added energy in music inside the game's wider soundscape.

Keywords: videogame music, dynamics, adaptivity

Sisällys

Lyhenteet

1 Johdanto 1

2 Dynamiikka videopelimusiikissa 2

2.1 Dynaamisen musiikin määritelmä 2
2.2 Dynaamisen videopelimusiikin historia 3
2.3 Dynamiikan käyttö nykypeleissä 6
2.4 Musiikki tehokeinona peleissä 7
2.5 Reaktiivinen musiikki sävellystyössä 9

3 Erilaisten dynamiikan toteutustapojen selvittäminen 10

3.1 Projektin tavoite 10
3.2 Toteutustavat 11
3.3 Unity pelimusiikkiprojektin alustana 12
3.4 FMOD-äänimoottori 12

4 Esimerkkien toteutus 20

4.1 Perustan laatiminen äänitapahtumille 20
4.2 1. osa: Musiikin muovautuminen etäisyyden mukaan 21
4.3 2. osa: Pelihahmon vauhtia mukaileva arpeggio-raita 27
4.4 3. osa: Triggerillä käynnistyvä transitio toiseen kappaleeseen 32
4.5 4. osa: Pelihahmon törmäyksestä nopeutuva kappale 38

5 Dynaamisen videopelimusiikin toteutuksen analyysi 41

5.1 Projektin haastavuus 41
5.2 Havaintoja toteutuksesta 42
5.3 Dynamiikan implementointi peliprojekteissa 43
5.4 Projektin hyöty 44

6 Yhteenveto 45

Lähteet 47

Lyhenteet

DAW: *Digital Audio Workstation*. Yleinen termi digitaalisille äänenkäsittely-ohjelmille, jollaisia käytetään esimerkiksi musiikin tekemiseen ja jälkikäsittelyyn.

FMOD: Firelight Technologiesin kehittämä kaupallinen äänimoottori ensisijaisesti videopelien monipuolista äänitoteutusta varten.

MIDI: *Musical Instrument Digital Interface*. Tiedonsiirtojärjestelmä datan välitykseen musiikkilaitteiden välillä. Data sisältää informaatiota mm. nuottien sijainnista, sävelkorkeudesta ja voimakkuudesta.

1 Johdanto

Opinnäytetyön tarkoituksena on tutkia dynaamisen taustamusiikin käyttöä videopeleissä: käyttötapoja, niiden vaikutusta pelin olemukseen, dynaamisen videopelimusiikin historiaa ja tapoja implementoida tiettyjä tehokeinoja erikokoisiin videopeliprojekteihin.

Raportissa käydään aluksi läpi dynaamisen videopelimusiikin kehitysvaiheet, ja sen avulla selvitetään musiikin asemaa videopeleissä. Samalla tutkitaan teknisten rajoitteiden vaikutusta varhaiseen videopelimusiikkiin ja sitä, kuinka samaisten rajoitteiden pohjalta syntyi monta dynaamisen videopelimusiikin piirrettä, jotka ovat käytössä nykyäänkin.

Opinnäytetyötä varten tehtiin Unityssä esittelyversio eli demo, johon sisältyy neljä erilaista tapaa muovata dynaamisesti taustamusiikkia joko korostamaan pelihahmon liikettä tai painottamaan pelissä tapahtuvaa toimintaa. Raportissa tutkitaan näiden tehokeinojen vaikutusta pelattavuuteen, selitetään, miten ne on implementoitu Unity-projektiin FMOD-ääniefektimoottoria hyödyntäen, ja käydään läpi, mitä musiikkiteorian konsepteja näitä dynaamisia kappaleita säveltäessä on huomioitu. Dynaamiselle musiikille on käyttötapoja videopeleissä käytännössä rajattomasti, mutta näiden neljän esimerkin pohjalta äänisuunnittelijat voinevat keksiä useita erilaisia tapoja muuntaa musiikkikappaleita dynaamisesti omissa projekteissaan.

Raportissa oletetaan lukijan perehtyneen jonkin verran Unityn käyttöön ja C#-ohjelmointikieleen. Raportissa käydään läpi myös FMOD-ääniefektimoottorin käyttöä ja jonkin verran dynaamiseen videopelimusiikkiin liittyvää musiikkiteoriaa, mutta lukijalla ei oleteta olevan pohjatietoa näistä aiheista.

2 Dynamiikka videopelimusiikissa

2.1 Dynaamisen musiikin määritelmä

Dynaamisella musiikilla (tunnettu myös mukautuvana musiikkina) videopeleissä tarkoitetaan taustamusiikkia, joka reagoi pelaamisen aikana tapahtuviin asioihin. Konsepti on vastakkainen staattiselle musiikille, joka on etukäteen ajastettu ja nauhoitettu reagoimaan ruudulla eteneville, ennakkoon ajastetuille kohtauksille. Elokuvien ääniraidat ovat hyvä esimerkki tämänkaltaisesta staattisesta musiikista; niiden musiikki on valmiiksi editoitu ja nauhoitettu käyttäytymään tietyllä tapaa valmiiksi säveltäjän tai leikkaajan määrittämillä ajoituksilla. Kummankin tyyppinen musiikki reagoi liikkuvan kuvan tapahtumiin, mikä voi tarkoittaa mitä tahansa hienovaraista muutoksista musiikkikappaleen vaihtumiseen, mutta kriittinen ero dynaamisen ja staattisen musiikin muutoksiin tulee siitä, että dynaamisen musiikin muutoksia ei ole sidottu lineaariseen aikajanaan. (1.)

Raren vuonna 1996 julkaisema tasoloikkapeli Banjo-Kazooie on varhainen esimerkki hienovaraisemmasta dynaamisesta musiikista. Pelissä jokaisella tasolla on uniikin, alati pyörivän taustamusiikkinsa lisäksi esimerkiksi vaimeampi marimbaversio, johon taustamusiikki häivytetään pelihahmon sukeltaessa vedenpinnan alle. (2.) Pelimusiikkia ei siis ole ohjelmoitu tai nauhoitettu muuttumaan tietyssä kohdassa videota tai välianimaatiota, vaan musiikin muutoksia määrää pelaajan ohjaama toiminta aktiivisessa pelitilanteessa.

Dynaamiseen musiikkiin voi joskus sisällyttää myös tiettyjä ääniefekteiksi luokiteltavia sointitapahtumia. Pelattavuuden reagoivat, musiikkia tehostavat ääniefektit on toteutettu siten, että ne mukailevat taustamusiikin sointuharmoniaa tai melodiaa näin esimerkiksi vahvistaen tai painottaen taustamusiikin senhetkistä energiaa ja tunnelmaa.

2.2 Dynaamisen videopelimusiikin historia

Ensimmäinen videopeli, joka käytti dynaamiseksi miellettävää taustamusiikkia, oli Space Invaders (1978) (2). Tuolloin videopelimusiikki itsessään oli vielä varsin primitiivistä, ja Space Invadersin taustamusiikki koostuukin vain neljästä jatkuvasti toistuvasta yksinkertaisesta kolmioaaltoäänestä. Pelin taustamusiikista tekee dynaamista se, että tietyin intervallein tämän toistuvan kappaleen tempo nopeutuu ruutua alaspäin pelaajaa kohti laskeutuvien vihollisten nopeuden kiihtyessä samoilla intervalleilla (3). Näin taustamusiikki lisää kasvavaa painetta pelaajalle, joka tuntee vihollisten nopeuden muutoksen ja niiden lähestymisen myös musiikin kautta.

Toinen varhaisimmaksi dynaamiseksi musiikiksi mielletty taustamusiikki löytyy pelistä Frogger (1981), jossa taustamusiikki oli jo merkittävästi monipuolisempaa ja musiikkikappaleita useampia (2). Froggerissa pelaaja yrittää saada viisi sammakkoa yksi kerrallaan tien ja joen yli turvallisiin koloihin. Joka kerta, kun sammakko saapuu turvallisesti koloon, peli pysäyttää toistuvan taustamusiikkiraidan, soittaa lyhyen sävellyksen ja palaa sitten toistamaan alkuperäistä taustamusiikkia (4). Peli toistaa jokaiselle saavutetulle kololle erilaisen musiikkiraidan, mikä oli jo itsessään varsin iso määrä musiikkia videopelissä 1980-luvun vaihteessa.

Taustamusiikista tehtiin dynaamista videopelien varhaisessa vaiheessa kuitenkin olosuhteiden pakosta. Yleensä tavoitteena oli saada tärkeäksi nähdyt ääniefektit soimaan rajatuilla äänikanavilla ongelmitta. Esimerkiksi vuonna 1987 julkaistu Atari 2600 -versio Double Dragon -pelistä pyrki saamaan kaiken hyödyn TIA-äänipiirin kahdesta äänikanavasta (5). Taustamusiikki käyttää kohinaääntä rumpuina, vastavuoroisesti näppäilevää bassoa ja melodiaääntä. Rumpujen kanssa samaista kohinaääntä käyttävien ääniefektien soidessa rumpuraita väistyy, kunnes ääniefekti on soinut loppuun. Ruudunvaihtoääniefekti taas joutuu mykistämään sekä rummut että basson soidakseen selkeästi.

Säveltäjät ja ohjelmoijat joutuivat tasapainoilemaan samankaltaisten äänikanavarajoitusten kanssa pitkään, mutta esimerkiksi Commodore 64:n SID-äänipiiri

mahdollisti jo huomattavasti laajemman muokattavuuden (6). R-Type-videopelin Commodore 64 -versiossa Chris Hielsbeckin sovitus Masato Ishizakin sävellystyöstä liitti avaruusaluksen ampumisäänet suoraan musiikkiraitaan, jolloin efektiraita soi vain pelaajan ohjaaman aluksen ampuessa. Nämä ”ampumisäänet” on Hielsbeckin kappaleessa sävelletty kuin energiseksi rumpuosioksi. Lopputuloksena taustamusiikki tuntuu tekevän itse ampumisesta innostavampaa pitäen samalla musiikkia äänimaailman etualalla.

1990-luvun alkupuolella konsolipelaamisen johtavat yhtiöt Nintendo ja Sega rupesivat käyttämään soundfonteja. Soundfontit ovat sampleista eli ääniklipeistä koottuja kompressoituja kirjastoja, joita pystyy käyttämään sävellysten toistamiseen MIDI-komentojen avulla (7). MIDI (Musical Instrument Digital Interface) on tiedonsiirtojärjestelmä, joka vie nuottien, sävelkorkeuden, nuotin pituuden, vibraton ja panoroinnin digitaalista informaatiota (8). Säveltäjä pystyy käytännössä säveltämään MIDI-partituureja haluamallaan käyttöliittymällä, ja MIDI toistaa ne parametreineen eri soitinraidoille asetetuilla äänillä.

Soundfontit mahdollistivat paljon persoonallisemman ja monipuolisemman äänimaailman, sillä osaava säveltäjä saattoi täyttää soundfontin itse tuotetuilla äänillä yksinkertaisten ääniaaltojen sijaan. Käytännössä ainoa rajoitus säveltäjille oli tilanpuute, jonka vuoksi sampleja jouduttiin kompressoimaan paljon. Näistä rajoituksista huolimatta pelimusiikki alkoi viimein saavuttaa pariäänisten melodioiden sijaan jo perinteisempien sävellyksien struktuureita, harmonioita ja piirteitä. David Wise, yksi tunnetuimpia videopelisäveltäjiä 1990-luvun alussa, puski SNES-konsolin äänikapasiteetit äärimmilleen ja sai näin aikaiseksi todella tunnelmallista ja monipuolista videopelimusiikkia. Wisen säveltämässä Donkey Kong Country -pelissä hänen tuottamansa äänipankit käyttivät konsolilta niin paljon muistia, ettei sitä riittänyt MIDI-ohjaimen käyttöön. Wise ratkaisi rajoitukset ilman kompromisseja kirjoittamalla käsin tekstieditorilla halutut äänet ja niiden parametrit oikeaan järjestykseen heksamuodossa. (9.)

MIDI:n ja soundfontien käyttö mahdollisti tehokkaita ja kevyitä dynaamisen musiikin käyttötapoja jopa nauhoitettuun musiikkiin verrattuna. Aiemmin mainitun,

vuonna 1997 julkaistun Nintendo 64 -pelin Banjo-Kazooie säveltäjä Grant Kirkhope käytti pelin triggereitä häivyttääkseen ja nostaakseen tiettyjä soitinraitoja midipartituurissa, mikä mahdollisti pelimusiikin reagoimisen pelihahmon toimintaan. Jokaisen pelialueen yhdessä taustamusiikin MIDI-partituurissa soivat samanaikaisesti taustamusiikin kahden eri version soitinraidat. Pelihahmon sukeltaessa vedenpinnan alle peli laukaisee triggerin, joka aiheuttaa tavanomaisen taustamusiikin soittimien raitojen häivytyksen ja mykistettyjen, vedenalaisen version soittimien raitojen (harppu, marimba) nostamisen häivytyksellä esille. Pintaan noustessa taas laukeaa triggeri, jonka seurauksena vedenalainen versio väistyy tavanomaisen musiikin tieltä. Peli käyttää samanlaista triggerireaktiota ja MIDI-partituuritäyttöä myös aulamaailmassa. Pelihahmon lähestyessä jonkin pelimaailman sisäänkäyntialuetta peli laukaisee triggerin, jonka seurauksena aulamaailman tavanomainen taustamusiikki häivyttää esille kyseiseen kenttään assosioituja instrumentteja. Kummassakaan tapauksessa musiikki ei siis katkea eikä melodia tai harmonia muutu, vaan musiikkikappale muuttaa olemustaan pelitilanteeseen sopivammaksi eri soittimien ja parametrien avulla. (10.)

Tapa on hyvin kevyt ja yksinkertainen, sillä näin taustamusiikki ei tarvitse kuin yhden MIDI-partituurin kerrallaan eikä ole tarvetta luoda systeemiä kahden MIDI-partituurin häivytykseen toisiinsa tai ajoittaa niitä oikein; kumpikin versio soi samassa MIDI-partituurissa samaan aikaan, joten häivytyksen voi helposti toteuttaa käyttäen MIDI:n sisäänrakennettuja parametreja. Soundfontien väistyttyäkin 2000-luvun taitteessa varsinkin Nintendo on edelleen käyttänyt MIDI:ä täysin nauhoitettujen taustamusiikkikappaleiden dynamiikkaan. Sen joustavuus parametrien muokattavuudessa pelin aikana on edelleen yksinkertaisempi, helpompi ja kevyempi kuin nauhoitettua musiikkia varten luodut työkalut.

Nauhoitettua musiikkia käyttänyt, mutta samalla MIDI:ä dynamiikkaa varten soveltanut peli on erimerkiksi Super Mario Galaxy vuodelta 2007. Koji Kondon orkesterinauhoitetun taustamusiikin ohelle äänijärjestelmäohjelmoija Masafumi Kawamura rakensi äänentoistojärjestelmän, jolla MIDI:ä hyödyntäen ääniefektit voivat seurata nauhoitetun taustamusiikin tempoa ja sointua. Esimerkiksi

pelihahmon lentäessä ulos tähtilaukaisimesta peli toistaa harpulla arpeggiota (sävelasteikolla nousevasti tai laskevasti etenevää yksiäänistä säveltä) samassa tempossa ja sävellajissa kuin mitä sillä hetkellä soiva taustamusiikki toistaa. (11.)

2.3 Dynamiikan käyttö nykypeleissä

Nykyään videopeleille ja konsoleille on varattu niin paljon muistia, että äänenlaadulle ja taustamusiikin määrälle ei ole enää suurempia rajoituksia. Musiikin määrä ja laatu vaikuttaa lähinnä pelin kokoon kiintolevyllä, ja tehokkaalla äänen kompressoinnilla musiikkikappaleiden ko'oista voidaan saada häviävän pieniä. Nintendon Super Smash Bros. Ultimate -pelissä oli vuonna 2018 julkaisun aikaan yli 900 verrattain korkealaatuista musiikkikappaletta, yli 28 tuntia musiikkia yhteensä (12), joka todella tehokkaan kompressointitekniikan ansiosta vei vain 1,2 Gt tallennusmuistia (13). Musiikkimäärä nousi yli 900:sta yli 1100:aan vuoden 2021 loppuun mennessä julkaistujen DLC-pakettien myötä (14).

Nykyisin pelimusiikki tehdään säveltäjän parhaaksi näkemässään muodossa, yleensä jonkinlaisessa digitaalisessa äänenkäsittelyohjelmassa eli DAW:ssa (Digital Audio Workstation) ja soittimiksi voi nauhoittaa tai luoda minkälaisia ääniä tahansa, kuten vaikkapa levyttävät artistit tai elokuvamuusikot ovat tehneet. Äänimaailmoiden teknisten rajoitusten hälvetessä pelisäveltäjät ovat voineet emuloida paljon lähemmin esimerkiksi elokuvamusiikin tehokeinoja.

Vaikka nauhoitettua musiikkia ei olekaan helppoa muovata dynaamiseksi instrumenteittain, valmiiksi miksattuja musiikkipätkiä voi käyttää dynaamisesti erilaisilla tavoilla. Esimerkiksi Rockstarin vuonna 2010 julkaistussa pelissä Red Dead Redemption suurin osa kaikista pelin musiikkiraidoista on tarkoituksella sävelletty samassa tempossa A-molli-sävellajiin (2). Tämä tarkoittaa sitä, että pelin taustamusiikki voi vaihtua esimerkiksi rauhallisesta matkamusiikista hektiseksi tulitaistelumusiikiksi ilman, että musiikkiraitojen transitiot törmäävät epämiellyttäviin tai sopimattomiin sointusiirtymiin. Tämä antoi musiikin ohjaajille

vapaammat kädet vaihtaa musiikkia mahdollisimman sulavasti tekemättä ylimääräisiä, mahdollisesti hankalia poikkeussääntöjä.

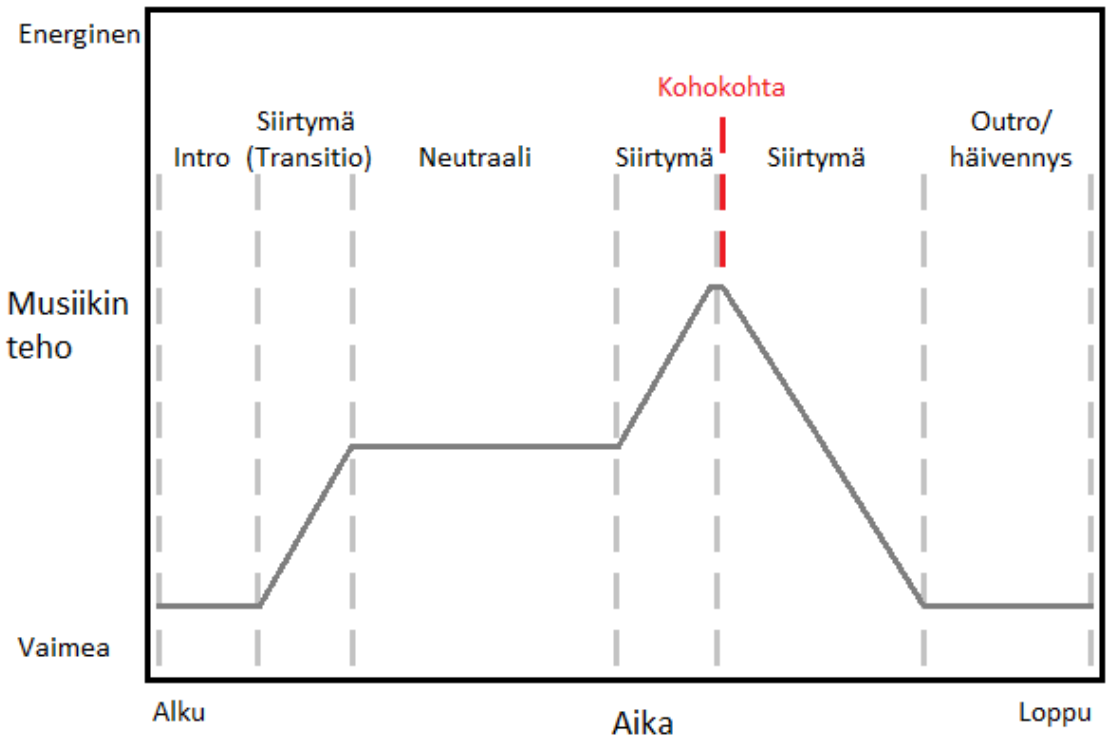
2.4 Musiikki tehokeinona peleissä

Videopelimusiikkia on aina ollut vaikeampi ohjata reagoimaan tapahtumiin yhtä hienovaraisesti kuin esimerkiksi elokuvamusiikkia. Elokuvamusiikki on yleisimmin valmiiksi sävelletty tiettyjä kohtauksia varten, ja usein säveltäjä on nähnyt elokuvan lähes valmiissa muodossaan tehdessään päätöksiä musiikkikappaleiden transitioista ja korostettavista hetkistä. Näin säveltäjällä on hyvin vapaat kädet esimerkiksi korottaa taustamusiikki riemukkaaksi fanfaariksi onnistumisen hetkellä, tuoda hillittyjä sointuja hiljaisuuden vallitessa tai korostaa hirviön pelottavaa sisääntuloa sähäkällä vibratoviululla.

Dynaamisella musiikin keinoilla videopelit pystyvät emuloimaan monia elokuvamusiikin tehokeinoja, vaikka elokuvien kaltaista lineaarista aikajanaa ei olekaan videopelisäveltäjien käytettävissä. Taustamusiikin vaihtuminen sopivammaksi kohtauksen mukaan, musiikin äkillinen mykistyminen painottamaan tilannetta tai tiettyjen soittimien häivyttäminen tarpeen mukaan onnistuu helposti niin elokuvamusiikissa kuin dynaamisessa videopelimusiikissakin.

Kaikki elokuvien tehokeinot eivät ainakaan vielä ole kuitenkaan mahdollisia dynaamisen pelimusiikin keinoin. Moni hienovaraisimmista muovautumisista elokuvamusiikissa tapahtuu tukiessa tiettyä hetkeä, esimerkiksi taustamusiikin yltäessä kohokohtaansa tietyllä hetkellä tarinaa. Kuvassa 1 on hypoteettinen esimerkki lineaarista mediaa varten tuotetun musiikkikappaleen osien vuorovaikutuksesta. Videopeleissä saman toteuttaminen ei ole niin yksinkertaista, sillä kohokohtaan nouseminen sulavasti vaatii musiikilta siirtymävaiheen, ikään kuin kuljettavan reitin neutraalivaiheesta kohokohtaan. Tämän vuoksi perinteisellä tavalla sävelletty tarinaa tukeva taustamusiikki tarvitsee toimiakseen etukäteen tiedon siitä, minkä ajan kuluttua kohokohta tulisi saavuttaa. Tätä pulmaa ovat pelien säveltäjät ja äänijärjestelmäohjelmoijat yrittäneet ratkoa eri tavoin, esimerkiksi säveltämällä kohokohtaan yltyvästä musiikista sellaista, että

hyvinkin äkillinen siirtymä kohokohtaan ei tuntuisi niin odottamattomalta. Ongelman ratkaisemiseksi on luotu lineaarisia tai lähes lineaarisia lyhyitä pelisektioita taustamusiikin neutraalivaiheen ja kohokohdan välillä, jotka toimivat ennalta tiedettynä siltana näiden välillä, tai tavallisimmin kohokohdat tulevat esiin pelin välivideoissa, joiden taustamusiikit saa sopimaan ennalta luodun videon lineaariseen aikajanaan yhtä helposti kuin elokuvamusiikin tapauksessa.



Kuva 1. Hypoteettinen esimerkki lineaarista aikajatkumoa seuraavan taustamusiikin rakenteesta. Varsinkin siirtymävaiheet ovat haastavia dynaamiselle videopelimusiikille.

Siirtymät kohokohdan jälkeen taas ovat yleensä varsin helppoja toteuttaa videopeleissäkin, sillä kohokohdan soitua tarkkaa kohdeajastusta ei yleensä tarvitse tietää. Kappale voi käytännössä noudattaa säveltäjän mielensä mukaan tuottamaa siirtymää ilman ongelmia, kunhan senhetkinen taustamusiikki ei vaadi toisenlaisia dynaamisia muutoksia samaan aikaan.

Dynaamisella musiikilla voidaan saavuttaa paljon sellaisia erilaisia, pelattavuutta tehostavia hetkiä musiikin avulla, jotka eivät olisi muuten mahdollisia

videopeleissä, joista valtaosa etenee pelaajan ohjauksen tahtiin. Pelattavuudesta laukaistavat ääniefektit voivat toimia yhdessä taustamusiikin kanssa jakamalla keskenään samaa tahtia tai melodiaa ja näin luoda kokonaisvaltaisemman multimediakokemuksen, jossa taustamusiikki ei ole vain pelattavuudesta erillinen pala, vaan tuntumaa tai liikkeen tunnetta vahvistava osa peliä.

2.5 Reaktiivinen musiikki sävellystyössä

Pelimusiikkiin liittyvän tietotekniikan kehittyminen ja dynaamisen musiikin keinojen yleistyminen on lisännyt pelisäveltäjien työvälineisiin tekniikoita, jotka ovat uniikkeja dynaamisesti muovaantuvalle musiikille. Nämä tekniikat voivat usein vaatia säveltäjältä erilaista näkemystä siitä, mitä musiikkikappale tarvitsee tai mitkä näkökohdat vaativat tarkempaa harkintaa (15).

Dynaamiselle videopelimusiikille on kaksi erilaista toteutustapaa: horisontaalinen uudelleen sekvensointi ja vertikaalinen uudelleen orkestrointi. Kumpaankin tapaan liittyy omat käyttökeinonsa ja ne täyttävät erilaisia tarpeita.

Horisontaalisella uudelleen sekvensoinnilla viitataan tekniikkaan, jossa kokonaisiksi nauhoitettuja ja miksattuja musiikkikappaleita tai lyhyempiä osia muovataan keskenään. Tavanomaisimpia käyttötapoja tälle tekniikalle on kahden musiikkiraidan tai musiikin osan häivyttäminen toisiinsa esimerkiksi kohtauksen tai pelialueen vaihtuessa. Horisontaalisella tekniikalla siis käsitellään tavanomaisesti musiikkikappaleita yksittäisinä entiteetteinä ja niiden senhetkistä suhdetta toisiinsa. (16.)

Vertikaalisella uudelleen orkestroinnilla taas tarkoitetaan tekniikkaa, jolla yksittäisen musiikkikappaleen sijaan muovataan musiikkikappaleen eri osia: soittimia, ääniä, tempoa, jne. (16). Esimerkkinä hienovaraisemmasta verikaalisen tekniikan toteutuksesta Nintendon Mario Kart 8 -ajopelissä pelaajan ajokin lähestyessä tiettyä nopeutta taustamusiikkia säestämään nostetaan hi-hat-rumpukomppi, joka antaa ylimääräistä vaikutetta ajotilanteen hektisyydestä. Vertikaalista tekniikkaa varten dynaamisesti muovautuva taustamusiikki on miksattava

osissa, jotta yksittäisiä soittimia ja ääniä pystyy muovaamaan erikseen. Mario Kart 8 -esimerkissä hi-hat-rumpukomppi on siis oma, taustamusiikista erillinen ääniraitansa, joka soi jatkuvasti taustamusiikin tahtiin. Mielikuva rumpukompin katoamisesta ajokin pysähtyessä ja kompin palaaminen ajokin kiihdyttäessä on toteutettu rumpukompin äänenvoimakkuuden muuttamisella pelin triggereiden laukaisuista. Muovaaminen siis tapahtuu pelkästään yhteen taustamusiikin osaan samalla, kun kompista erilleen miksattu musiikkiraita soi muovaamatta jatkuen normaalisti. (17.)

3 Erilaisten dynamiikan toteutustapojen selvittäminen

3.1 Projektin tavoite

Opinnäytetyön tavoitteena oli luoda yksinkertainen pelimaailma, jonka sisällä käyttäjä voi tarkastella erilaisia dynaamisen videopelimusiikin käyttötapoja. Tarkoituksena oli suunnitella interaktiiviseen pelimaailmaan neljä erityyppistä dynaamisen musiikin instanssia, säveltää näille instansseille sopiva taustamusiikkinsa niille tarpeellisine ominaisuuksineen, selvittää näiden instanssien toteutustapoja ja toteuttaa nämä instanssit pelimoottorin sisällä.

Insinöörityöraportissa käydään läpi näiden käyttötapojen ominaisuudet yksi kerrallaan, selostetaan niiden toteutuksen pelimoottorin ja äänimoottorin osalta ja selvitetään niiden käyttötarkoituksia eri videopeleissä. Projektin toivotaan antavan työkalut yksinkertaisten dynaamisen videopelimusiikin tehokeinojen tuottamiseen ja inspiroivan niiden soveltamista myös erilaisiin, uudenlaisiin ja hienovaraisempiin tilanteisiin.

Työssä perehdytään myös lähtökohtaiseen käsitykseen dynaamisen videopelimusiikin määritelmästä, sen käyttötarkoituksesta, dynaamisen musiikin aktiivisesta vaikutuksesta pelillisyyteen, sen eri käyttötavoista ja niiden kehityksestä ennen ja nyt sekä näiden käyttötapojen ja niihin liittyvien työkalujen vaikutuksesta videopelisäveltäjien tapaan luoda musiikkia.

Lopulta projektin tavoitteena on myös pohjustaa osaamista ja ymmärtämistä videopelimusiikin tarpeista ja näiden tarpeiden mahdollisista eroista ja yhtäläisyyksistä muihin ammattimusiikin aloihin nähden.

3.2 Toteutustavat

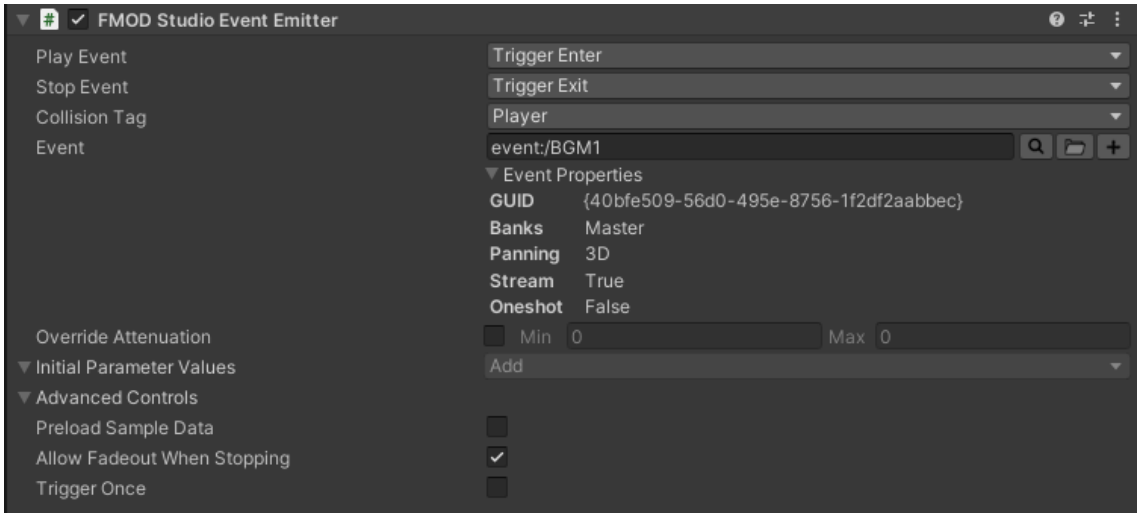
Neljän erilaisen dynaamisen musiikin toteutustapaa esittelevää projektia suunnitellessa pari avainasiaa nousee pintaan. Äänimoottorin on oltava tarpeeksi joustava ja mielellään se tukee valmiiksi niin monta projektille tarpeellisista ominaisuuksista kuin mahdollista, ja mahdollisesti puuttuvien ominaisuuksien tapauksessa niiden implementoinnin on oltava suhteellisen helppoa. Pelimoottorin osalta pelin ulkonäköön tai pelattavuuden mielekkyyteen ei tarvitse nähdä erityistä vaivaa, mutta pelimaailman luonti ja yksinkertaisen videopeliohjattavuuden toteuttaminen on tärkeä osa videopelimusiikin muovautumisen demonstrointia varten. Pelimoottorin on myös tuettava äänimoottoria.

Tässä projektissa pelimoottoriksi valittiin Unity, josta opinnäytetyön tekijällä oli eniten kokemusta entuudestaan. Unityyn päädyttiin sen nopean ja yksinkertaisen muokattavuuden ansiosta; yksinkertaisten pelimaailmoiden luominen ja toiminnan pikainen testaaminen onnistuu helposti. Tästä syystä Unityä käytetäänkin laajalti ideoiden testaukseen ja prototyyppien luomiseen, vaikka lopullinen tuote päädyttäisiinkin kehittämään eri moottorilla (18). Nopean muokattavuuden avulla musiikkiin ja äänentoistoon keskittyvä projekti voi demonstroida helposti esimerkiksi pelitriggereihin liittyviä ääni-instansseja ja pelialueen kontekstiin liittyviä musiikin muovautumisia ilman suuria ponnisteluja pelillisyyteen liittyvien ratkaisujen tai visuaalisen ilmeen kanssa.

Äänimoottoriksi valittiin FMOD (https://www.fmod.com/), joka on Firelight Technologiesin kehittämä ääniefektimoottori videopelejä ja muunlaisia sovelluksia varten. Moottori valittiin sen kokonaisvaltaisuuden takia, ja sen ominaisuudet ja miksauslaite mahdollistavat sekä vertikaalisen uudelleen orkestroinnin että horisontaalisen uudelleen sekvensoinnin keinoja noudattavat dynaamisen musiikin käyttötavat. FMOD on myös integroitu toimimaan suoraan Unityn kanssa (19).

3.3 Unity pelimusiikkiprojektin alustana

Unity käyttää pohjanaan C#-ohjelmointikieltä ja on hyvin nopeasti muokattava ja verrattain yksinkertainen pelimoottori. FMOD on integroitu toimimaan Unityn kanssa, ja näin ne kykenevät vuorovaikuttamaan keskenään joko Unityn koodin tai kuvassa 2 olevan Event Emitterin kaltaisten integroitujen käyttöliittymäkomponenttien avulla. Kokonaisuutena Unity on siis varsin kätevä pelimoottori muovautuvan pelimusiikin periaatteiden esittelyyn ja vertailuun kolmiulotteisessa ympäristössä, ja pienien muutoksien kokeilu lennosta on helppoa.



Kuva 2. FMOD:n integroitu oliokomponentti Unityn käyttöliittymässä. Event Emitterin avulla käyttäjä pystyy jopa luomaan ääni-instansseja ilman erillistä koodaamista. (20.)

Unity pitää sisällään myös esimerkiksi oman fysiikkamoottorinsa, valaistusjärjestelmänsä ja erilaisia triggereitä, joiden avulla nopea, välittömästi interaktiivisen pelitilanteen luonti onnistuu helposti ilman ulkoisia työkalujakin.

3.4 FMOD-äänimoottori

FMOD on äänimoottorituoteperhe, joka keskittyy pääosin videopelien äänentoistoratkaisuihin. Se on suosittu äänimoottori kaikenkokoisissa videopelituotannoissa, ja se tarjoaa jopa ilmaista indie-lisenssiä kaikille 200 000 dollarin

vuositulot alittavalle, alle 500 000 dollarin tuotantobudjetin tuotteelle. Äänimoottori on siis laajalti saatavilla ja mahdollistaa ammattimaisen lopputuloksen, ja sillä on kattava dokumentaatio avustamaan minkä tasoista käyttöä tahansa aloittelijasta veteraaniin.

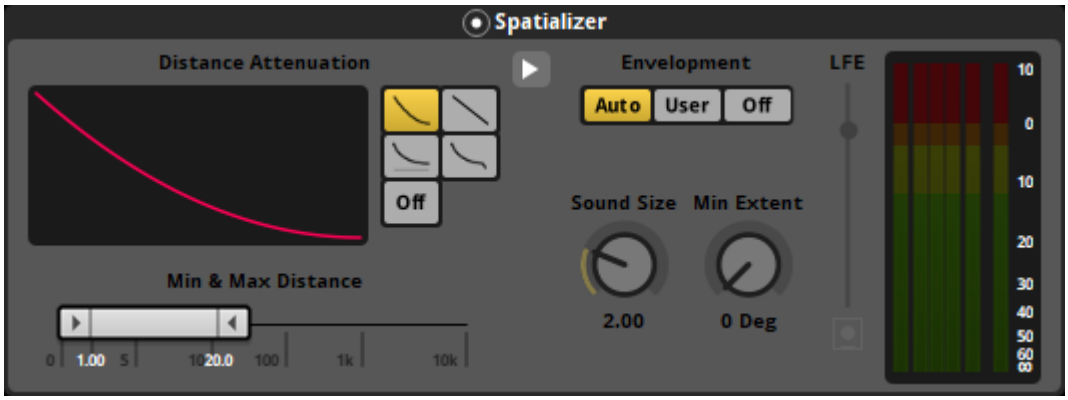
FMOD on kattavasti käytössä videopeleissä ja integroitu moniin suosituimmista pelimoottoreista. Sitä hyödyntäviä indie-pelejä ovat esimerkiksi Noita, Spelunky 2 ja Untitled Goose Game, jonka dynaamista musiikkia kekseliäästi hyödyntävä äänidesign voitti palkintoja muun muassa GDC:ssä vuonna 2020 (Independent Games Festival Awards - Excellence in Audio). FMOD:a ovat käyttäneet myös maailmanluokan julkaisijoiden avustamat peliyritykset tuottaessaan sellaisia pelejä kuin World of Warcraft, Dark Souls ja Diablo 3. (21.)

Ominaisuudet

FMOD:n suunnittelufilosofia on ollut tehdä ohjelman käyttöperiaatteista ja käyttökokemuksesta samankaltaiset kuin DAW:ien kanssa. Tämä on mahdollistanut säveltäjille kivuttoman siirtymän tavanomaisista äänenkäsittelyohjelmista FMOD:iin, sillä monet ohjelman äänenkäsittelyyn liittyvistä ominaisuuksista ja käyttöliittymän tavoista käsitellä kyseisiä ominaisuuksia joko mukailevat tai on otettu suoraan tyypillisistä DAW:n käyttöperiaatteista (22).

FMOD:n mahdollistamia äänitapahtumia on neljää eri lajia: 2D-toiminta, 2D-aikajana, 3D-toiminta ja 3D-aikajana. 2D-toimintaa käytetään ääniin, joiden on tarkoitus soida muuntumatta dynaamisesti. 2D-aikajanalla taas voidaan muokata mikserin kautta ääntä muuntumaan tietyin tavoin käyttäjän valitsemien parametrien muuttuessa. Käytännössä uuden äänitapahtuman tekeminen vain tarkoittaa uuden tapahtuman luomista valitun tapahtuman mallille, eli toimintaa ja aikajanoja voidaan käyttää samanaikaisesti saman äänitapahtuman sisällä käyttäjän tarpeiden mukaan. Äänitapahtumien 3D-variaatiot toimivat käytännössä 2D-variaatioiden tavoin sillä erolla, että kuvassa 3 näkyvän spatialisoijan (spatializer) avulla äänitapahtuman soinnille pystyy antamaan binauraalisia

ominaisuuksia, mikä antaa pelaajalle illuusion äänilähteen sijainnista suhteessa pelaajan korviin erilaisten panoroinnin ja äänenvoimakkuuden keinoin. (23.)



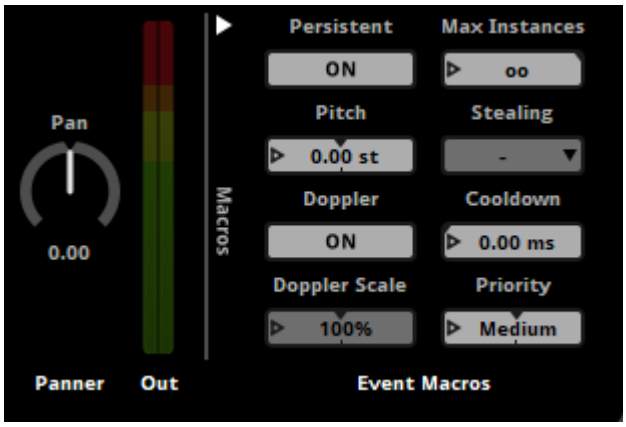
Kuva 3. 3D-äänitapahtuman spatialisoijaliitännäinen (20).

DAW:n tapaan äänisuunnittelijan pääasiallinen työskentelytila FMOD:ssa on äänimikseri, jossa voidaan käsitellä ääniraitoja ja audioväyliä (audio bus) yhteisessä aikajanassaan. Näitä raitoja pystytään muokkaamaan, leikkelemään ja tehostamaan käyttäjän mielen mukaan, ja erillisten ääniraitojen ja audioväylien äänenvoimakkuutta voi säätää erikseen vaimentimella. (22.)

Ääniraidoille voidaan antaa pre-fader- ja post-fader-efektejä ja käskyjä tavanomaisen DAW:n lailla (21). FMOD Studiossa tulee sisäänrakennettuna kokoelma tärkeimpiä efektejä (esimerkiksi kompressori, rajoitin, kaiku ja kolmikanavainen taajuuskorjain eli equalizer), mahdollisuus sivukanavoimiseen (sidechain) ja mahdollisuus lähettää ääniraitaa toiseen audioväylään (24). Ohjelman sisäänrakennetut ääniefektit ovat kevyitä ja yksinkertaisia, mikä mahdollistaa niiden käytön mahdollisesti raskaan pelin ajoaikana. Sisäänrakennetut efektit siis kuormittavat mahdollisimman vähän, mutta FMOD:iin pystyy myös tuomaan omia liitännäisefektejään, jos äänisuunnittelijalla on tarvetta käyttää jotain monimutkaisempaa dynaamista äänen muovausta pelin aikana.

Jokaisessa FMOD:n tapahtumainstanssille (event) voi myös määrätä omat makroasetuksensa, joiden avulla voidaan muokata instanssin äänentaajuutta tai määrittää esimerkiksi, kuinka monta kyseiden tapahtuman tapahtumainstanssia

voi olla samanaikaisesti käynnissä (25). Kuvassa 4 näytetään FMOD:n makrot oletusasetuksineen.

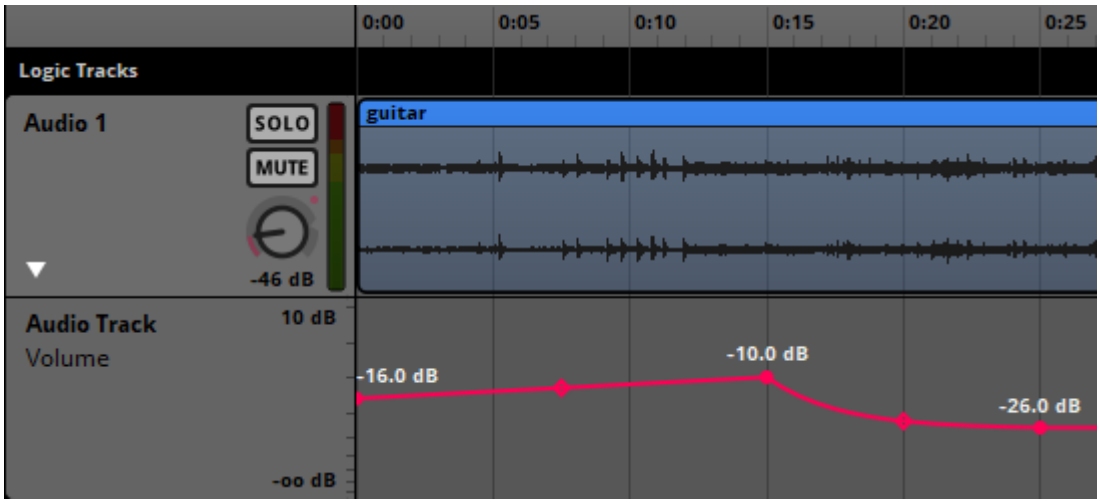


Kuva 4. FMOD:n äänitapahtuman pääaudioväylän makroikkuna oletusasetuksissaan (20).

Persistent-asetus määrittää, jääkö tapahtumainstanssi soimaan silloinkin, kun se ei muuten tuottaisi enää ääntä. Pitch eli äänentaajuus muovaa instanssin toistotaajuuden nopeutta. Doppleria voidaan käyttää, jos tapahtumainstanssin äänen halutaan imitoivan doppler-ilmiötä eli äänentaajuuden nopeuden vaihtumista ääntä aiheuttavan objektin nopeuden ja kulkusuunnan mukaan kuuntelemispisteestä. Samaan asetukseen liittyvä doppler scale määrää taajuusmuutoksen määrän. Max instances määrittää, kuinka monta tapahtumainstanssia voi olla käynnissä samanaikaisesti. Stealing taas määrittää, millä perusteella valitaan instanssit, jotka max instances -asetuksen instanssimäärän ylittyessä pysäytetään tai virtualisoidaan (tässä tapauksessa virtualisoinnilla tarkoitetaan instanssia, joka käynnistymisestä ja aktiivisuudesta huolimatta ei tuota ääntä, ennen kuin jo käynnissä oleva instanssi pysähtyy). Cooldownilla voidaan asettaa tietty aika, jonka verran uusi instanssi joutuu odottamaan aktivoitumistaan saman tapahtuman edellisen instanssin äänentoiston alkamisesta. Priority määrittää tapahtuman ääniraitojen tärkeysjärjestyksen virtualisointitilanteessa. (25.)

Yhtä dynaamisen videopelimusiikin tärkeimpiä ominaisuuksia tukee myös DAW:eista tuttu parametriautomaatiolinja. Tällaisia automaatiolinjoja pystyy liittämään kaikenlaisiin parametreihin: äänenvoimakkuuteen, efektien arvoihin tai

vaikka edellä mainittuun äänentaajuusmakroon. Linjan avulla näiden ominaisuuksien parametreja voi määritellä muuntumaan tietyllä tavalla tietyillä hetkillä kesken äänentoiston (26). Kuvassa 5 näkyy esimerkki parametriautomaatiosta, joka muuttaa ääniraidan äänenvoimakkuutta eri kohdissa aikajanaa.



Kuva 5. Ääniraidan äänenvoimakkuuden säätimeen liitetty parametriautomaatiolinja. Esimerkissä automaatio nostaa volyymia tasaisesti -10 desibeliin asti 15 sekunnin kohdalle, minkä jälkeen se laskee -26 desibeliin hiljentyen aluksi nopeammin ja lopulta tasaantuen 25 sekunnin kohdalle. (20.)

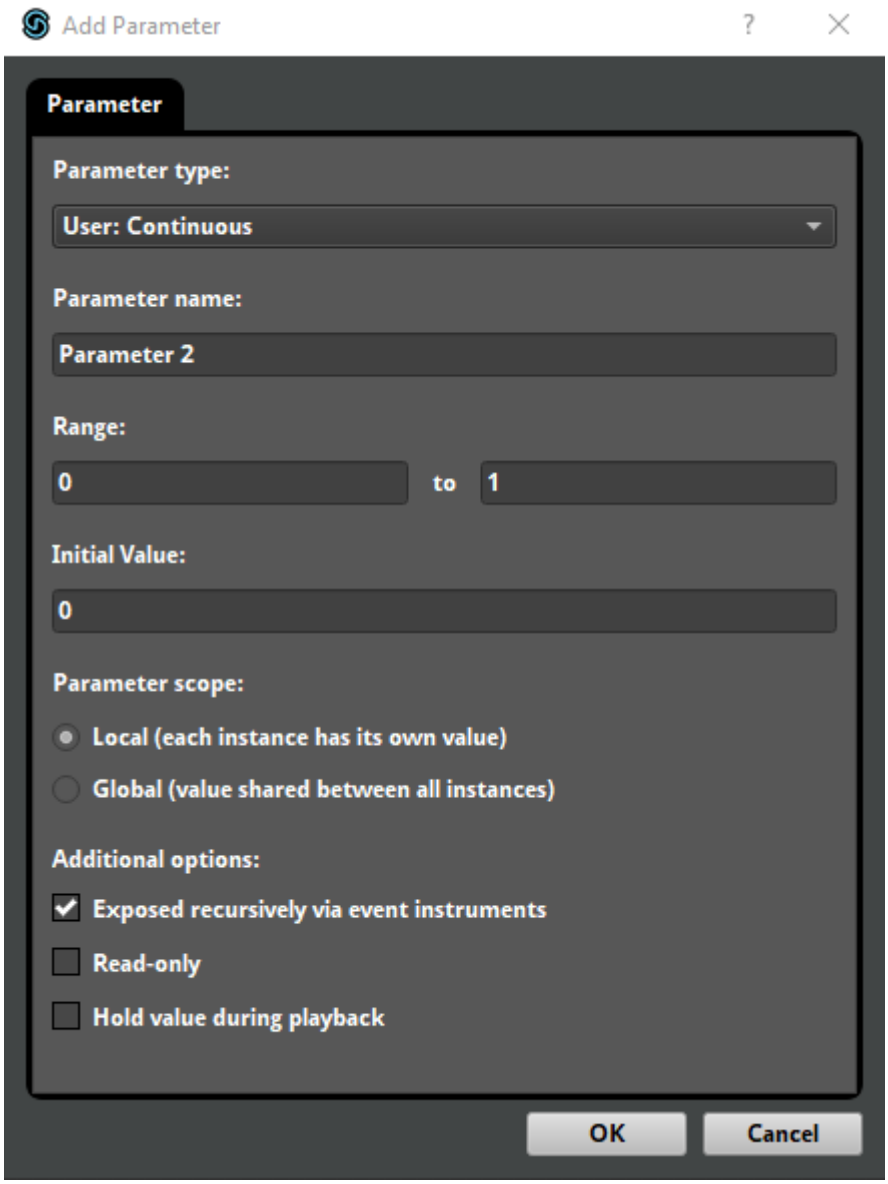
Tämä antaa äänisuunnittelijalle lisää mahdollisuuksia tapoihin muovata ääntä. FMOD:ssa automaatiota voi myös sitoa parametritriggereihin ja parametriliukusäätimiin, joita hyödyntäen esimerkiksi pelimoottorin triggerien kanssa vuorovaikuttaen voidaan luoda dynaamisesti muovautuvia ja pelitapahtumiin reagoivia peliääniä. (26.)

Äänipankit ovat eräänlaisia kirjastoja valituille äänitapahtumille. Ne alustavat ja kompressoivat äänitapahtumat videopeliprojektin käyttöön. Äänipankit mahdollistavat vain tarpeellisten äänitapahtumien lataamisen pelitilanteen mukaan, jolloin ylimääräiset ja tarpeettomat äänitapahtumat eivät varaa peliltä muistia turhaan. (22.)

FMOD:ssa voidaan hyödyntää audioväylää, joka on jälleen DAW:eista tuttu ominaisuus. Äänitapahtuman sisällä olevia eri ääniraitoja voidaan lähettää

yhdelle audioväylälle, jolloin tätä audioväylää voi käyttää jokaisen väylälle lähetetyn ääniraidan äänitulon muokkaamiseen samanaikaisesti. Näin esimerkiksi efektejä ja automaatiota pystyy implementoimaan useampaan ääniraitaan kerrallaan. (24.)

Reaktiivisen musiikin osalta yksi tärkeimmistä ominaisuuksista FMOD:ssa ovat parametrisivut (parameter sheet), joiden avulla käyttäjä voi luoda erilaisten parametrimääritteiden avulla sekä eteneviä että äkillisiä muutoksia (24). Esimerkki parametrisivun luonti-ikkunasta näkyy kuvassa 6.



Kuva 6. Uuden parametrimääritteen lisääminen (20).

Nämä parametrit voidaan liittää pelimoottorin tarjoamiin parametreihin, jolloin saadaan aikaiseksi ääniä, jotka muuttuvat reaktioina pelin tapahtumiin. Uutta parametria lisätessä käyttäjä pystyy käyttämään erilaisia FMOD:n valmiita parametrimääritteitä, jotka muuttuvat esimerkiksi olion korkeuden tai suunnan mukaan, tai käyttäjä voi tehdä mukautettuja parametrimääritteitä spesifisiin tarkoituksiin. (24.)

Mahdollisuudet

FMOD on suunniteltu pääosin kaikenlaisten videopeliäänien tarpeisiin, niin ääniefektien kuin taustamusiikin, joten ohjelma on luonnostaan rakennettu tehokkaaksi alustaksi videopelimusiikin vaatiman dynamiikan luontiin ja testailuun.

Tärkein FMOD:n tarjoama työkalu musiikin dynamiikan työstämiseen ovat parametrimääritteet, jotka mahdollistavat vuorovaikutuksen äänimoottorin ja pelimoottorin välillä. Näin mitä tahansa muuttuvaa parametria pelimoottorin sisällä voidaan käyttää reaktion käynnistämiseen äänitapahtumassa. Parametriautomaatiolinjaa käsittelemällä voidaan saada aikaan esimerkiksi taustamusiikin tai tietyn instrumenttiraidan vaimenemista pelihahmon saavuttua kohtauksen vaihtavaan triggeripisteeseen. Vaihtoehtoisesti samankaltaisen vaimenemisen voi esimerkiksi sitoa pelihahmon etäisyyteen triggeripisteestä, jolloin transitio musiikin äänekkäimmästä volyymista täyteen hiljaisuuteen on sidottu pelihahmon etenemiseen, mikä tekee transitiosta reaktiivisen ja sulavan.

Parametrimääritteillä voidaan myös käsitellä olion vauhtia eri suuntiin, mikä mahdollistaa taustamusiikin dynaamiset muutokset sidottuna suoraan pelihahmon liikkuvuuteen. Parametrimääritteelle voidaan asettaa esimerkiksi parametrivektori, jonka sisälle soveltuvat parametrit muovaavat äänitapahtuman ominaisuuksia käyttäjän asettamien arvojen mukaisesti. Käyttäjä pystyy muokkaamaan äänitapahtuman hetkellistä käyttäytymistä ääniefekteistä siirtymiin sallien lukemattomia erilaisia reaktiivisia kokoonpanoja. Käytännössä tällä voidaan esimerkiksi painottaa pelihahmon liikkuvuutta musiikin avulla, kuten

voimistamalla rumpuraidan ääntä pelihahmon kiihdyttäessä vauhtia tai siirtämällä taustamusiikkia kiertämään eri osissa pelihahmon astuessa erilaisille alustoille.

FMOD:lla pystytään myös häivyttämään musiikkia sisään ja ulos ennalta määrittelemättömällä hetkellä esimerkiksi pelihahmon koskiessa triggeriin tai tietyn ajanjakson umpeutuessa. Musiikkia pystyy myös häivyttämään toisiinsa. Näin taustamusiikin pystyy pysäyttämään tai vaihtamaan pehmeästi ilman töksähtämistä, eikä häivyttämistä varten tarvitse etukäteen tietää, missä kohtaa musiikkikappaleen aikajanaa sen kuuluisi hiljentyä.

Käyttöönotto

Pelimoottori, tässä tapauksessa Unity, tarvitsee integraatiopaketin (integration package), jotta integraatio FMOD:n kanssa on mahdollinen. Integraatiopaketin asentamiseen on kaksi tapaa.

FMOD Studio Suite on saatavilla FMOD:n lataussivulla (https://www.fmod.com/download). Lataaminen edellyttää palveluun rekisteröitymistä. Lataussivulta ladataan FMOD Studion uusin Unity Verified -versio, jonka tulisi toimia kaikissa moderneissa Unityn versioissa. Samalta sivulta löytyy myös Unityn integraatiopaketti, jolta samoin ladataan viimeisin Unity Verified -versio.

Integraatiopaketin voi vaihtoehtoisesti ladata suoraan Unityn Asset Store -kaupasta hakemalla ilmaisen paketin ”FMOD for Unity” ja klikkaamalla ”add to my assets”. Tämän jälkeen klikkaamalla ”Open in Unity” -nappia Unity aukaisee paketinhallintaikkunan (package manager). Kun klikkaa ”My Assets” -välilehden auki, FMOD for Unityn pitäisi löytyä listalta. Klikkaamalla ”download”-nappia integraatiopaketti ladataan Unityn käytettäväksi.

Kun integraatiopaketti on ladattu, se on tuotava Unityyn klikkaamalla Assets -> Import Package -> Custom Package… ja navigoimalla ladatulle integraatiopaketille. Tässä vaiheessa käyttäjä voi valita, mitkä tiedostot paketista tuodaan

Unityyn, mutta syvemmin perehtymättä toiminnallisuuden varmistamiseksi paketin koko sisällön voi tuoda Unityyn olemalla muuttamatta vaihtoehtoja ja klikkaamalla import. Tuodut tiedostot luodaan automaattisesti projektin Assets-kansion alle. Tässä vaiheessa on suositeltavaa myös tehdä Unityyn oma kansionsa FMOD-äänipankkeja varten.

Tässä vaiheessa Unity vaatii äänipankkia FMOD:lta. Tämä onnistuu FMOD Studion kautta luomalla jokin äänitapahtuma (event) ja määräämällä se jollekin äänipankille. Tämä toteutetaan klikkaamalla oikealla hiirenpainikkeella kyseistä äänitapahtumaa ja valitsemalla Assign to Bank ja sen jälkeen joko luomalla uuden pankin tai liittämällä tapahtuman valmiiseen Master-äänipankkiin. Tämän jälkeen FMOD Studion välilehdestä Edit -> Preferences… määritellään Build-välilehdessä ”Built banks output directory” -hakemisto aiemmin tehtyyn kansioon FMOD-äänipankkeja varten.

Tämän jälkeen voidaan FMOD Studion File-välilehdestä klikata Build-vaihtoehtoa, jolloin FMOD koostaa käyttäjän lisäämistä äänistä äänipankin aiemmin luotuun FMOD-äänipankkihakemistoon Unitya varten. Tässä vaiheessa Unityssa asetetaan polku projektille klikkaamalla Unityn FMOD-välilehdessä ”Edit Settings”, valitsemalla joko Project (kokonainen FMOD-projekti), Single Platform Build (yhden koontiversion sisältö) tai Multiple Platform Build (useamman alustan koontiversion sisältö) ja valitsemalla kohtaan Build Path aiemmin luodun FMOD-äänipankkikansion sisälle muodostuneen ”Desktop”-kansion polku. Jos kaikki on toiminut oikein, FMOD:n pitäisi tässä vaiheessa olla täysin integroitu Unityn kanssa.

4 Pelimusiikkisimerkkien toteutus

4.1 Perustan laatiminen äänitapahtumille

Insinöörityöprojektin alkupisteessä oli tavoite luoda vuorovaikutettava pelialue, jossa käyttäjä voi liikkua kolmiulotteisessa pelimaailmassa samalla laukaisten ja observoiden erilaisia, dynaamisesti muuttuvia musiikkiraitoja. Projektin

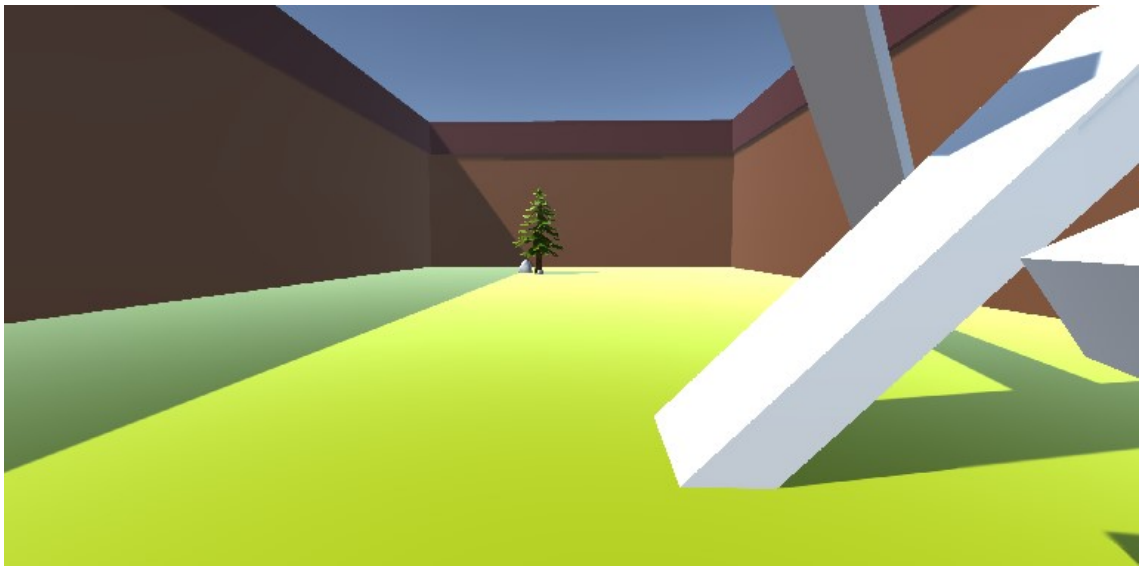
kehityksen tärkeimmät kiintopisteet olivat dynaamisesti muuntuva musiikki, kolmiulotteinen pelimaailma ja dynaamisten musiikkiraitojen muuttumisen pelitapahtumien mukaan mahdollistava pelillisyys. Unity ja siihen integroitu FMOD mahdollistavat nopeasti muokattavan ja testattavan pelimaailman ja laajamittaisen dynaamisen äänenmuokkauksen.

Projekti koostui neljän erilaisen dynaamisen pelimusiikin esittelystä, jotka rajattiin omiin huoneisiinsa. Jokainen neljästä erimerkistä sisältää dynaamisen videopelimusiikin arkkityyppejä ja esimerkkejä sekä horisontaalisesta uudelleen sekvensoinnista että vertikaalisesta uudelleen orkestroinnista toteutustapoina.

Jokaiselle neljästä dynaamisen pelimusiikin esimerkistä tarvittiin myös omat, kyseiselle dynamiikan muodolle sopivat musiikkiraitansa. Sen sijaan, että projektia varten olisi etsitty valmista musiikkia, musiikkikappaleet tuotettiin itse, jotta nämä musiikkiraidat palvelisivat esimerkkidynamiikkaa niin instrumenttiraitojen erittelyn kuin kyseisen dynamiikan tehokeinojen ja musiikkiteorian puolesta.

4.2 1. osa: Musiikin muovautuminen etäisyyden mukaan

Ensimmäinen projektia varten tuotettu dynaamisen pelimusiikin esimerkki oli muovautuva taustamusiikki pelihahmon etäisyyden mukaisesti kohdeolioon nähden. Käytännössä siis tiettyä objektia lähestyessä taustamusiikin tunnelma muuttuu dynaamisesti, mitä lähemmäksi pelihahmo kävelee, ja palautuu lähemmäs alkuperäistä muotoaan, mitä kauemmas pelihahmo peräántyy. Kuvassa 7 on kuvakaappaus huoneesta, jossa musiikin muovautuminen tapahtuu.



Kuva 7. Yksinkertainen huone, jonka taustamusiikin ääniraidat vaimenevat riippuen pelaajan etäisyydestä puuhun.

Projektin esimerkissä kohdeolio on puu, jota lähestyessä taustamusiikki rauhoittuu ja tekee tilaa vaimealle lintujen viserrykselle. Esimerkin toteutustapa on myös esimerkki vertikaalisesta uudelleen orkestroinnista.

Sävellys

Esimerkkiä varten tuotettu taustamusiikkikappale on yksinkertainen kahden tahdin kierto, jossa sähköbasso soittaa melodialinjaa. Hiljaiset rummut ja pitkiä sointuja soittava jousisoitinsektio tukevat melodiaa. Basso ei ole tavanomainen melodiasoitin sen korkeiden nuottien heikkouden takia, joten hillityt rummut ja helposti seurattava jousisektio auttavat tuomaan aktiivisempaa bassoa eteenpäin kappaleen taustalla. Näin rauhallinen kappale saa hieman epätavallisia piirteitä.

Puuta lähestyttäessä taustamusiikin rummut ja jousisoittimet häivytetään taka-alalle ja lopulta mykäksi jättäen jäljelle pelkän sähköbasson melodiaraidan ja lisäten äänimaailmaan lintujen viserryksen. Hiljaisella kappaleella on vähemmän väistettävää linnun viserryksen pyrkiessä äänimaailmassa etualalle. Tätä vaikutusta tehostaa vielä sähköbasson matala äänentaajuus verrattuna korkeaan

viserrykseen, jolloin kumpikaan äänistä ei kilpaile keskenään samasta äänialasta, kuten esimerkiksi korkeammat jousisoittimet kilpailisivat viserryksen kanssa. Tällä tavoin äänimaailma pysyy puhtaana ja selkeästi erotettuna, jonka ansiosta äänien erottelusta ja näin myös kuuntelemisesta helpompaa.

FMOD

DAW:ssa tuotettu valmis taustamusiikkikappale renderöitiin niin, että jokainen erillinen instrumenttiraita tuotiin omiksi äänitiedostoiksi. Nämä äänitiedostot ladattiin FMOD:iin, jossa ne voidaan yhdistää kokonaiseksi kappaleeksi. Musiikkikappale täytyy tuoda osissa, sillä muuten erillisten instrumenttiraitojen häivyttäminen dynaamisesti olisi paljon vaivalloisempi prosessi.

Aluksi FMOD-projektiin luotiin uusi kaksiulotteinen aikajanatapahtuma (2D timeline), jonka mikseriin tuotiin jokainen musiikkikappaleen instrumenttiraita omille raidoilleen. Taustamusiikin instrumenttiraitojen lisäksi aikajanan mikserissä on vielä linnun viserrykselle yksi audioraita, joka soi samanaikaisesti intrumenttiraitojen kanssa. Tässä vaiheessa alkuperäisen kappaleen intrumenttiraitojen lähtökohtaisia äänitasoja voi vielä muutella, mutta suurimmaksi osaksi tämänkaltaiset jälkiprosessoinnit on jo toteutettu ennen renderöintiä DAW:n puolella, jolla on laajemmin ominaisuuksia äänitasojen virittämiseen. Sen sijaan tiettyjen efektien käyttö lähtökohtaisestikin voi olla tärkeää toteuttaa FMOD:ssa DAW:n sijaan, sillä esimerkiksi DAW:ssa luodut kaikuefektit ovat renderöity suoraan äänitiedostoon, jonka toistamisesta seuraa kaiun töksähtävä katkeaminen äänitiedoston kiertäessä musiikkikappaleen lopusta takaisin alkuun. Sen sijaan FMOD:ssa lisätty kaikuefekti prosessoidaan pelin aikana, joten musiikkikappaleen kierto ei vaikuta sen toimintaan, mikä luo tasaisen kaikuefektin vailla keskeytyksiä.

Ääniaikajanan raitoja voidaan muovata erilaisilla triggereillä, jotka saadaan vuorovaikuttamaan Unityn skriptauksen kanssa. Etäisyyden mukaiselle muutokselle on FMOD:ssa oma sisäänrakennettu parametrisivunsa (parameter sheet), joka lisätään klikkaamalla mikserin raitojen yläpuolella olevaa plusmerkkiä,

valitsemalla ”Add parameter sheet” ja klikkaamalla tarvittua parametrityyppiä joko tarjotusta listasta tai klikkaamalla ”New parameter”. Ennen parametrisivun luontia distance-parametrille voi myös asettaa etäisyyden sen mukaan, kuinka kaukana kohdeobjektista pelihahmon pitää olla, jotta äänitapahtuma on kokonaan lähtöparametreissään. Esimerkissä etäisyys lasketaan 0:n ja 50:n välillä.

Esimerkin aikajanassa taustamusiikin tulee muuttua niin, että rummut ja jousisoittimet hiljenevät pelaajan lähestyessä kohdeoliota ja lopulta mykistyvät täysin kohdeolion kohdalla. Päinvastaisesti linnun viserrysaudion tulee häivetä sisään kohdeobjektia lähestyessä ja olla kovimmalla volyymillaan kohdeolion kohdalla. Tämä toteutetaan erillisten volyymi- eli äänenvoimakkuussäädinten (fader) automaatiolla. Etäisyysparametrin välilehdellä volyymisäädinten (ja muiden automaatiolla säädettävien efektien ja ominaisuuksien) automaatioraitaa pääsee muokkaamaan klikkaamalla automatisoitavaa kohdetta, tässä tapauksessa yksittäisen ääniraidan volyymisäädintä, oikealla hiirenpainikkeella ja valitsemalla ”Add automation”. Tämä nostaa esille ääniraidan alapuolelle automaatioraidan, jossa voi muovata kohteen toimintaa parametrin aikajanan mukaisesti. Parametrin aikajana on esimerkissä jaettu 50 osaan, jonka vasen puoli on 0 ja oikea 50. Koska parametrijanan oikeanpuolinen pääty on myös etäisyysparametrin lakipiste, rummut ja jousisoittimet alkavat siellä kovimmalla volyymillaan eikä viserrystä kuulu ollenkaan. Automaatiolla matka nollapisteestä lakipisteeseen saadaan toteutettua pehmeästi automaatiolinjalla liu’ulla, jolla saadaan aikaiseksi tasainen nousu ja lasku kahden pisteen välillä.

Lopuksi pehmentämään siirtymää pisteiden välillä vielä hieman lisää on distance-parametrin hakuvauhtia (seek speed) muutettu hitaammaksi, jolloin parametriaikajanalla kulkemisessa pisteiden välillä kestää hetki. Kuvassa 8 on esimerkki hakuvauhtiasetuksista.

Kuva 8. Seek speed määrittää viiveen määrän parametrien muutoksessa (20).

Näin muutos pelihahmon liikkuessa ei ole välitöntä, jolloin taustamusiikin muuntuminen on hienovaraisempaa ja vähentää tuntua siitä, että pelaaja muuttaa musiikkia kuin liukusäätimellä.

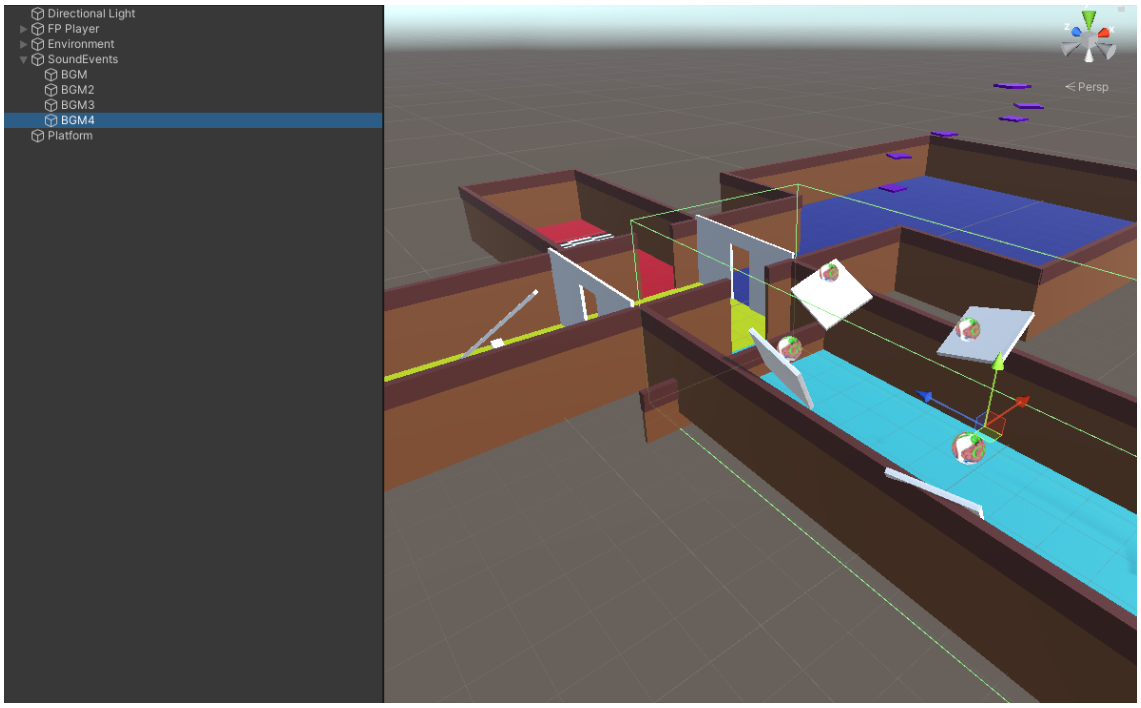
Äänitapahtuma tulee vielä lisätä äänipankkiin, joka on Unity-projektin käytössä klikkaamalla äänitapahtumaa oikealla hiiren painikkeella, klikkaamalla ”Assign to bank”, ”Browse” ja valitsemalla käytössä oleva äänipankki. Tämän jälkeen valitsemalla ”File”-välilehdeltä ”Build” FMOD rakentaa äänipankin pelimoottorin käytettäväksi.

Unity

Projektin ensimmäinen esimerkki toteutettiin täysin FMOD Studio Event Emitterin kautta, joka on Unityn olioon liitettävä yksinkertainen komponentti. Tämä tarkoittaa sitä, että äänitapahtuman äänentoisto ei vaadi ylimääräistä ohjelmointia Unityn puolella, kunhan tapahtumalle tarpeelliset ominaisuudet ovat Event Emitterin toimintojen puitteissa.

Ensiksi pelihahmo-olio vaatii komponentin FMOD Studio Listener, joka merkitsee kyseisen olion ääntä kuuntelevaksi kohteeksi. Event Emitter katsoo kohdeolioksi olion, johon se on liitettynä ja etäisyys kohdeoliosta lasketaan Listeneriä kantavan olion sijainnista.

Projektia varten eri esimerkit käyttävät eri huoneita ja eri Event Emittereitä, joten eri esimerkkien taustamusiikkia toistavat alueet on rajattu erillisiin huoneisiin. Tämä toteutettiin sisällyttämällä jokaisen huoneen alue Event Emitteriä kantavaan olioon liitettyyn box collideriin, josta on tehty trigger kuvan 9 tavoin.



Kuva 9. Pelimaailma ja sen eri dynaamisen videopelimusiikin esimerkkejä esittelevät huoneet. Kuvassa on korostettu peliolio ”BGM4” ja sen box collider, joka merkitsee neljännen esimerkin huoneen rajat, jonka sisällä kyseinen taustamusiikki soi.

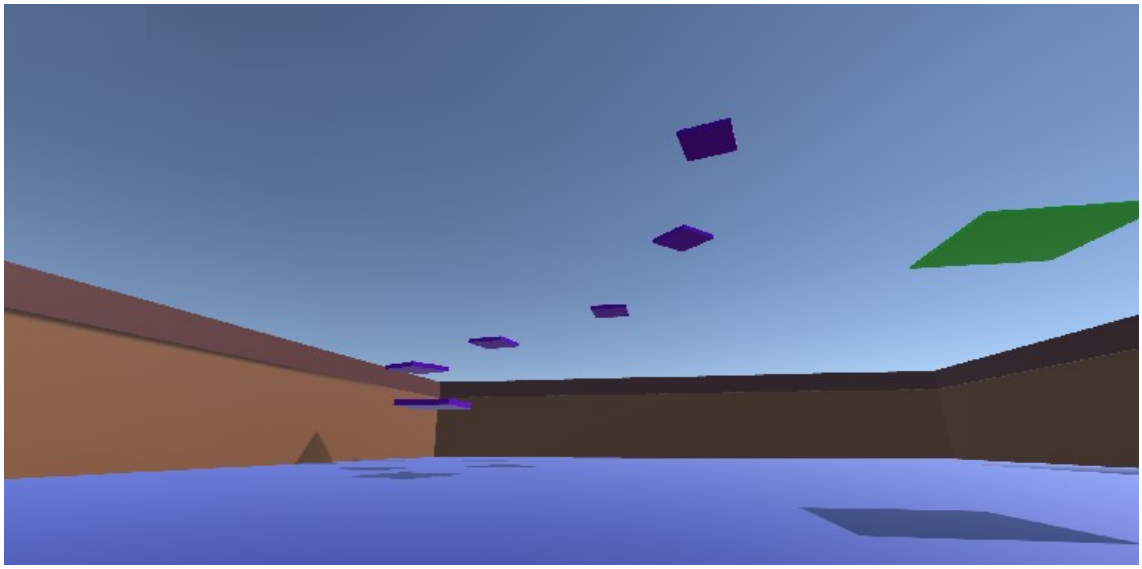
Tällaiselle peliolioon liitetylle Event Emittererille pystyy antamaan käynnistymisvalinnaksi ”Trigger Enter” ja pysäyttämisvalinnaksi ”Trigger Exit”, ja collision-tagiksi merkitään sama tag, jota pelihahmo kantaa. Tällöin olion collideriin osuva, nimettyä collision-tagia kantava olio käynnistää musiikin toiston ja colliderista poistuessaan pysäyttää sen.

Näin taustamusiikki käynnistyy pelihahmon astuessa huoneeseen. Huoneen ovella musiikki koostuu bassosta, rummuista ja jousisoittimista. Listeneriä kantava pelihahmo muuttaa FMOD:ssa asetettuja parametreja lähestymällä Event Emitteriä kantavaa oliota. Rummut ja jousisoittimet hiljenevät, mitä lähemmäs

pelihahmo lähestyy oliota, ja tilalle nousee linnun viserrystä, joka voimistuu. Lopulta pelihahmon ollessa kohdeoliossa kiinni musiikista ei kuulu kuin basso ja linnun viserrystä. Pelihahmon poistuessa ovesta musiikki lakkaa.

4.3 2. osa: Pelihahmon vauhtia mukaileva arpeggio-raita

Toinen esimerkki käyttää dynaamisesti muuntuvaa musiikkia painottamaan vauhdin tuntua. Tässä esimerkissä käytetään vertikaalista uudelleen orkestrointia muuntamaan ääntä pelihahmon senhetkisten liikkeiden mukaan. Kuvassa 10 näkyvät huoneen tasot, joiden avulla pelihahmolla voi kokeilla isompien korkeuserojen vaikutusta musiikin muutoksiin.



Kuva 10. Arpeggio-esimerkin huone. Huoneen taustamusiikki muovaantuu pelihahmon pystysuuntaisen vauhdin mukaisesti.

Kun pelihahmon vauhti on ylöspäin nouseva, taustamusiikissa etualalle nousee ylimääräinen helisevä ääniraita, joka koostuu pääosin nousevasta arpeggio-melodiasta. Kun vauhti taas on alaspäin laskevaa, myös arpeggio-melodia laskeutuva. Tasaisella maalla hypätessä pelihahmo siis kuulee peräkkäin kaksi eri arpeggio-raitaa: nouseva melodia hahmon noustessa ja laskeva melodia hahmon laskeutuessa.

Sävellys

Arpeggio-esimerkkiä varten toteutettu musiikkikappale on energinen, neljän eri soinnun kierto. Kappaleelle sävellettiin useampia sointuja, jotta olisi selkeämpää demonstroida sitä, kuinka arpeggiot seuraavat hetkellistä sointua missä kohdassa taustamusiikkia tahansa.

Instrumentteja ovat sähkö- ja pystypiano, elektroniset rummut ja sähköbasso. Lisäksi musiikissa on erilliset ääniraidat kummallekin arpeggio-melodialle, jotka soivat heleällä siniaaltoäänellä. Nämä kirkasääniset arpeggiot puskevat selkeästi läpi muusta musiikista alleviivatakseen juuri pelihahmon toimintaan liittyvää muutosta. Lisäksi arpeggiot soivat vain reaktiona pelihahmon toiminnalle, joten läpitunkeva ääni ei ehdi käydä kuulijalle epämukavaksi.

FMOD

Koska musiikkikappaleen jokaista osaa ei ole tarkoitus muovata dynaamisesti, riittää, että DAW:ssa renderöidään instrumenttiraidat yhdeksi äänitiedostoksi ja kumpikin arpeggio-raita omiksi tiedostoikseen. Ääniraidat tuodaan FMOD:iin uuden 2D-aikajanatapahtuman alle ja asetetaan alkamaan samalla hetkellä.

Esimerkkiä varten luodaan uusi parametrisivu oletussivujen sijaan. Lisättävä parametri on tyyppiä Continuous, vaihteluväli nollasta kahteen, ja aloitusarvoksi asetetaan 1. Näin aikajanatapahtumalla on kolme kiintopistettä: musiikkikappaleen neutraalitila, tila pelihahmon noustessa ylös ja tila pelihahmon laskeutuessa alas.

Parametrisivulle asetetun oletusarvon 1 on tarkoitus olla pelitilanteen neutraalitila, jolloin musiikkiraita soi, mutta kumpikin arpeggio-raita on täysin hiljaa. Arvo 2 on pelihahmon nousutilanteelle, joten nousevan arpeggio-raidan volyymiautomaatiolla äänitaso nousee mykästä nollaan desibeliin (FMOD:n ja DAW:ien tapauksessa desibeliä käytetään yleensä osoittamaan desibelimuutosta ääniraidan alkuperäisestä äänitasosta eikä suoraan äänitason desibelimäärää

itsessään). Arvo 0 on laskeutumistilanteelle, eli laskevalle arpeggiolle tehdään samankaltainen volyymiautomaatio kuin nousutilanteessa, paitsi että volyymi nousee mykästä nollaan desibeliin aikajanalla ykköspisteestä nollan suuntaan.

Parametrille annettu nopea hakuvauhti auttaa lieventämään musiikin töksähtämisen tuntua parametrien vaihtuessa toistensa välillä. Lyhyt viivästys myös pehmentää vaihtoa sen verran, että esimerkiksi arpeggion alkaminen heikolla nuotilla ei häiritse kuuntelukokemusta. Parametrin toimintaa voi soveltamisen jälkeen testailla helposti parametriaikajanan yläpuolella olevalla parametrisäätimellä, jonka avulla esimerkiksi hakijan vaikutus tai muutoksen hienovaraisuus on nopeasti testattavissa ennen äänipankin tuomista pelimoottoriin. Esimerkin pää-äänilinjalle (Master bus) on annettu myös toinen mukautettu parametri koko musiikkikappaleen häivytystä varten. Parametrin vaihteluväli on 0:n ja 1:n välillä, ja volyymi kasvaa näiden välillä tasaisesti mykästä nollaan desibeliin. Parametrille annettu hidas hakuvauhti takaa sen, että häivytysominaisuus tapahtuu, vaikka parametrin vaihteluvälillä hypättäisiinkin suoraan nollasta yhteen. Näin häivytyksen laukaisemiseen voidaan käyttää yksinkertaista triggeriä pelimoottorin puolella ja FMOD hoitaa tasaisen äänitasosäädön itsenäisesti.

Unity

Toisin kuin ensimmäisessä esimerkissä, loput esimerkit toteutettiin koodilla Event Emitter -komponentin sijaan. Esimerkki käyttää samoin Unityn box collider -komponenttia triggerinä liitettynä erilliseen olioon osoittamaan pelialue, jossa dynaamisen musiikkiraidan on tarkoitus soida. Taustamusiikin toimintalogiikkaa varten ohjelmoitu skripti on myös liitetty komponenttina samaan olioon.

FMOD:n äänitapahtuman toimintaa ohjaavaan C#-skriptiin tarvitaan ensiksi osoitus FMOD-äänimoottorin käytöstä using FMOD; -käskyllä. Luokkamäärittelyssä on viite pelihahmon GameObjectiin, jonka lisäksi määritellään FMOD:n taustamusiikki-instanssi esimerkkikoodin 1 tavalla. ”BGM2” viittaa äänitapahtuman nimeen, jolla se on tässä esimerkissä nimetty FMOD-äänipankin sisällä.

Pelaajaolion viitteen voi luoda hakemalla sen koodin avoimeen muuttujaan Unityn käyttöliittymästä tai vaikka hakemalla oliota tagin perusteella.

```
public class BGM2SetParameters : MonoBehaviour
{
    FMOD.Studio.EventInstance BGmusic2;
    public GameObject playerObject;

void Start()
    {
        BGmusic2 = FMODUnity.RuntimeManager.CreateIn-
stance("event:/BGM2");
    }
}
```

Esimerkkikoodi 1. FMOD-äänitapahtuman instanssin määrittely.

Koska FMOD:n Event Emitter -komponentti ei ole käytössä, täytyy box colliderille määritellä OnTriggerEnter- ja OnTriggerExit-komennot erikseen. Pelihahmon saapuessa collider-laatikon sisälle taustamusiikki-instanssi käynnistyy, ja pelihahmon liikkuessa laatikosta ulos musiikki häivytetään pehmeästi FMOD:ssa aiemmin toteutetun häivytysparametrisivun avulla. Instanssikutsu ja parametrikutsu toteutetaan esimerkkikoodissa 2. ”Fadeout” on esimerkissä kyseisen äänitapahtuman häivytystä ohjaavalle parametrisivulle annettu nimi, ja ” PlayerCharacter” on pelihahmon oliolle annettu nimi Unityssa. Parametrisivun oletusarvo 1 korvataan uudella arvolla 0, jonka tilaan musiikki siirtyy pehmeän verkkaisesti parametrisivulle annetun hakuvauhdin ansiosta.

```
private void OnTriggerEnter(Collider player)
{
    if (player.name == "PlayerCharacter")
    {
        BGmusic2.start();
    }
}

private void OnTriggerExit(Collider player)
{
    if (player.name == "PlayerCharacter")
    {
        BGmusic2.setParameterByName("Fadeout", 0f);
    }
}
```

Esimerkkikoodi 2. Taustamusiikki-instanssin käynnistys ja taustamusiikki-instanssin parametrisivun muutoskutsu.

Lopulta itse arpeggio-raitojen toiminnallisuus on toteutettu kutsumalla FMOD:ssa aiemmin luotua parametrisivua ja muuttamalla niiden parametreja pelihahmon liikkeen mukaan. Esimerkkikoodissa 3 on käytetty if-lauseita erittelemään tilanteet, joissa musiikin olisi tarkoitus muuttua. Koodi kutsuu myös pelihahmo-olioon ohjelmoitua liikkumisskriptiä, jonka metodi palauttaa pelihahmon senhetkisen pystysuuntaisen vauhdin. If-lauseet käsittelevät erikseen jokaisen kolmesta mahdollisesta arpeggio-äänitapahtuman tilasta ja asettaa arvoksi 2, jos olion vauhti on kohoavaa, 0 jos olion vauhti on alenevaa, ja 1, jos oliolla ei ole pystysuuntaista liikettä. Tarkistuksille annettiin pienet virherajat, jotta pelkkä epämääräinen huojunta ei käynnistäisi arpeggio-raitoja.

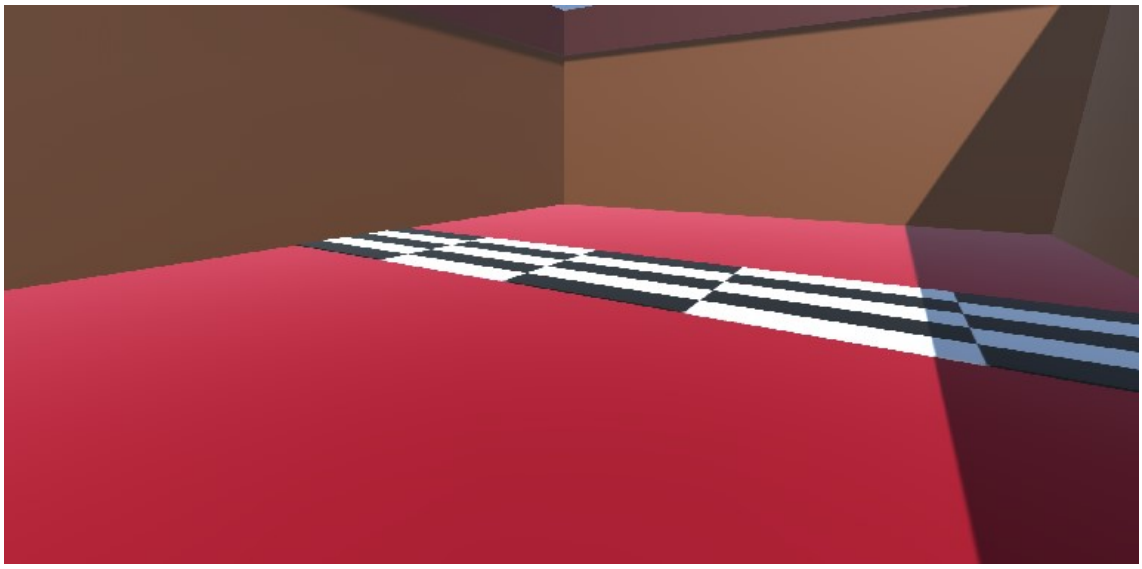
```
private void Update()
    {
        float i = playerObject.GetComponent<PlayerMovement>().Return-
Speed();

        if (i > 0.14)
        {
            BGmusic2.setParameterByName("Arpeggio", 2f);
        }
        else if (i < -0.14)
        {
            BGmusic2.setParameterByName("Arpeggio ", 0f);
        }
        else
        {
            BGmusic2.setParameterByName("Arpeggio ", 1f);

        }
    }
```

Esimerkkikoodi 3. Koodi hakee pelaajaolion liikeskriptin metodilta olion hetkellisen pystysuoraisen liikkeen vauhdin, jonka perusteella ”Arpeggio”-parametri saa uusia arvoja FMOD:ssa.

Pelihahmon tullessa huoneeseen musiikki käynnistyy räväkästi ilman häivytystä. Pelihahmon hypätessä pelaaja kuulee nousevan liikkeen aikana taustamusiikkia mukailevan, nousevan arpeggio-raidan ja pelihahmon laskeutuessa arpeggio-raita laskee. Pelihahmolla pystyy hyppimään lautoja ylöspäin ja hyppäämään korkealta alas niin, että kuulee vielä erityisen selvästi, kuinka laskeutuva arpeggio-raita seuraa taustamusiikin sointuja vaivatta. Pelihahmon poistuessa huoneesta musiikki häivytetään pehmeästi mykistymiseen asti.

4.4 3. osa: Triggerillä käynnistyvä transitio toiseen kappaleeseen

Kolmannessa esimerkissä toteutuu musiikkikappaleen vaihtuminen pelihahmon toimien mukaisesti. Kuvassa 11 näkyvässä huoneessa taustamusiikki vaihtuu lyhyen transition kautta uuteen musiikkikappaleeseen pelihahmon koskiessa ruutuviivaa.



Kuva 11. Kolmannen, dynaamisesti muuntuvat taustamusiikin huone. Ruutuviiva on triggeri, joka aktivoi FMOD-äänitapahtumassa loikan logiikkaraidalla.

Pelaajan laukaistessa triggerin Unityssa taustalla soiva musiikki pysähtyy välittömästi ja tuo tilalle ensin lyhyen transition (eräänlainen musiikillinen silta kahden musiikkikappaleen välillä) ja lopulta jättää uuden, hektisemmän musiikkikappaleen kiertämään taustalle. Kappale on esimerkki hieman monipuolisemmasta lähestymistavasta horisontaaliseen uudelleen sekventoimiseen.

Sävellys

Esimerkkiä varten tarvittiin kolme eri kappaletta: ensimmäinen kiertävä kappale, pelimoottorin laukaisema kertaalleen soiva transitiokappale ja viimeinen kiertävä kappale. Kaikki kolme musiikkikappaletta toteutettiin samassa DAW-istunnossa samoilla instrumenteilla, jotta saatiin pidettyä kappaleiden välinen johdonmukaisuus niin äänimaailmoissa kuin äänitasoissa. Instrumenttiraitoja on neljä: 70-luvun syntetisaattori, rock-rumpusetti, viulu ja elektroninen bassoääni. Transitiokappale ei käytä näistä kuin syntetisaattoria ja rumpuja, mikä luo äänellistä dynamiikkaa kahden kiertävän musiikkikappaleen välille.

Alkupisteessä soiva taustamusiikkikappale kiertää neljää sointua. Kappale on rauhallinen, rummut soittavat hieman monimutkaista mutta kevyttä komppia, ja

elektroninen bassoääni implikoi melodiaa tekemällä monimuotoisempaa liikettä viuluun verrattuna. Syntetisaattori soittaa pitkiä sointuja lyhyiden staccato-tahditusten kanssa. Kappaleen tempo on 144 iskua minuutissa.

Transitiokappaleeseen otettiin vaikutteita Mario Kart -pelisarjasta, jonka peleissä tämäntyyppiset siirtymät tapahtuvat aina pelaajan ohjastaman pelihahmon siirtyessä kilpailussa viimeiselle kierrokselle (17). Esimerkissä rummut ja syntetisaattori soivat samanaikaisesti kohtalaisen epämääräisessä rytmissä. Syntetisaattori soittaa leveää 11-sointua, joka liikkuu kromaattisesti (puolisävelaskeleittain) etenevästi korkeammalle. Transitiokappaleen tempo on 172 iskua minuutissa, joka on huomattavasti nopeampi tempo kuin ensimmäisessä kappaleessa. Transitiokappaleen kromaattisuuden luoma harmoninen hämäryys ja epämääräinen rytmi antavat kuulijalle mielikuvan siitä, että edellinen kappale on ohi, ja samalla annetaan tulevalle kappaleelle tilaa tehdä jotain irrallaan alkuperäisen kappaleen olemuksesta ja temposta.

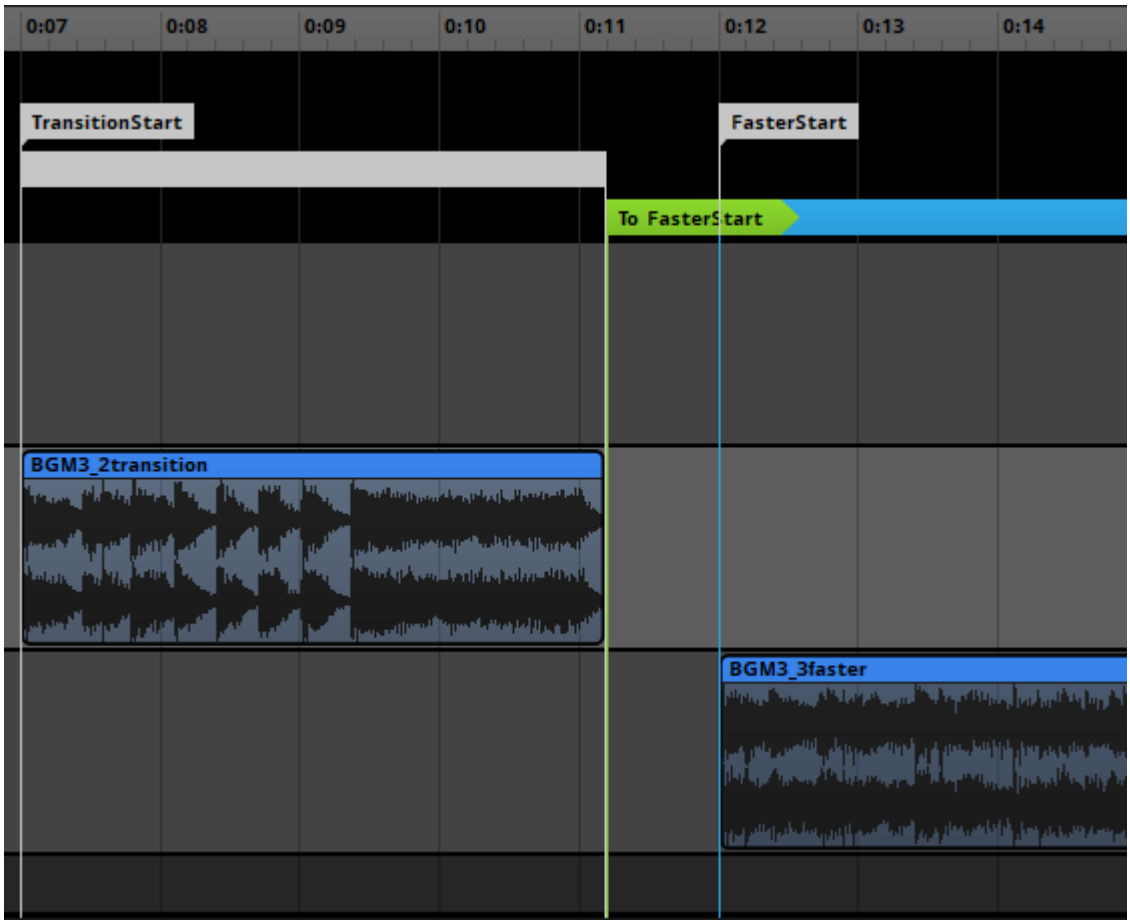
Viimeisen musiikkikappaleen tempo on transitiokappaleen lailla 172 iskua minuutissa. Kappale on kaksi kertaa pidempi kuin alkupisteessä soiva kappale, mutta käyttää pitkälti samaa sointukulkua. Ero ensimmäisen kappaleen sointuihin on toisen sointukierron loppuessa, jolloin kuulijalle totuteltuun Cmaj9-sointu kuullaankin kääntämällä ensiksi G-sävel oktaavia ylöspäin ja tämä puolet lyhyempi sointu siirtyy vielä laskemalla G- ja E-sävelet puolisävelaskelen alaspäin seuraavalle soinnulle, joka taas ratkeaa kauniisti Em9-sointuun sointukierron alussa. Näin viimeinen musiikkikappale kuulostaa pidennettynä ensimmäistä kappaletta kokonaisemmalta, mikä antaa sille lisää painoarvoa. Eroa ensimmäiseen kappaleeseen tekee myös vauhdikkaampi, yksinkertaisempi rumpukomppi sekä viulun ja bassoäänen tehtävien vaihtuminen. Viimeisessä kappaleessa bassoääni vuorostaan mukailee sointuja yksinkertaisesti ja viulu soittaa vauhdikasta melodiaa. Kappale kuulostaa näin hektisemmältä ja energisemmältä versiolta ensimmäisestä kappaleesta.

FMOD

Koska kolmannen esimerkin dynamiikan toteutustavassa on kyse horisontaalisesta uudelleen sekvensoinnista, FMOD ei tarvitse kappaleiden erillisiä instrumenttiraitoja lainkaan. DAW:ssa siis renderöidään nämä kolme kappaletta erillisinä, kokonaisina äänitiedostoina, joihin on kaikki tarvittavat instrumentit sisällytetty.

Nämä kolme äänitiedostoa tuotiin uuteen 2D-aikajanaan ja sen mikseriin omille ääniraidoilleen. Kappaleet ovat toisistaan täysin erillisiä ja soivat eri aikoihin, joten tässä esimerkissä ääniraidat on aseteltu eri kohtiin aikajanaa: alkukappale aikajanan alkuun, transitiokappale hieman alkukappaleen jälkeen 7 sekunnin kohdalle ja viimeinen kappale hieman transitiokappaleen jälkeen 12 sekunnin kohdalle.

Kappale tarvitsee kaksi kiertoaluetta (loop region), yksi alkukappaleelle ja toinen loppukappaleelle. Transitiokappaleen tulee soida vain kerran ja hypätä suoraan viimeiseen kappaleeseen soituaan kerran kokonaan. Tämän hypyn transitiokappaleesta viimeiseen kappaleeseen saa toteutettua suoraan FMOD:n tapahtuman mikserissä logiikkaraitojen (logic tracks) avulla. Logiikkaraidassa pystytään merkitsemään lähtöpisteitä, joihin saapuessaan musiikki loikkaa automaattisesti käyttäjän määräämään saapumispisteeseen. Klikatessa logiikkaraidalla oikeanpuolisella hiiren painikkeella kohdekappaleen alkukohtaan ja valitsemalla ”Add destination marker” FMOD merkitsee kohdan saapumispisteeksi. Tämän jälkeen transitioraidan loppuun klikatessa logiikkaraitaa ja valitsemalla ”Add transition to”-vaihtoehto ”Markers”-lista tarjoaa suoraan jokaista äänitapahtumaan merkittyä saapumispistettä. Näin transitiokappaleen päätyttyä FMOD löytää lähtöpisteen ja automaattisesti aloittaa toistamisen saapumispisteestä, eli viimeisen kappaleen alusta. Tämän toteutus logiikkaraidassa näkyy kuvassa 12.



Kuva 12. Mustalla logiikkaraidalla sijaitseva lähtöpiste ”To FasterStart” siirtää äänentoiston saapumispisteeseen ”FasterStart” automaattisesti, kun äänentoisto on sen kohdalla.

Logiikkaraitojen avulla äänitapahtumaan saa järjesteltyä useita eri kiintopisteitä, joiden avulla varsinkin monimutkaisemman dynaamisesti reaktiivisen taustamusiikin saa toteutettua selkeästi jäsenneltynä ja helposti muokattavana.

Unity

Kuten edellisessä esimerkissä, uusi skriptikomponentti liitettiin Unityssa tyhjään peliolioon, johon liitetty box collider trigger kattaa huoneen, jossa taustamusiikin on tarkoitus soida. Ensimmäisessä esimerkissä musiikkikappale pysäytettiin Event Emitterin kautta ja toisessa skriptin käynnistämän fadeout-parametrisivun avulla. Tässä esimerkissä musiikki lakkaa pysäyttämällä pää-äänilinjan määrittelemällä se ensin koodissa ja kutsumalla äänilinjan pysäytysmetodia

OnTriggerExit-metodissa. Esimerkkikoodissa 4 määritellään taustamusiikin äänitapahtuma, luodaan sille instanssi ja käynnistetään kyseinen instanssi edellisen esimerkin tavoin, minkä lisäksi määritellään edellä mainittu pää-äänilinja ja sen hiljentäminen OnTriggerExit-metodissa, jolloin kaikki äänitapahtumat lakkaavat. Lisäksi määritellään totuusarvomuuttujalla ”isBoost”, joka seuraa, onko musiikkia vaihtava triggeri laukaistu.

```
 public class BGM3 : MonoBehaviour
{
    FMOD.Studio.EventInstance BGmusic3;
    FMOD.Studio.Bus MasterBus;
    private bool isBoost = false;

    void Start()
    {
        MasterBus = FMODUnity.RuntimeManager.GetBus("Bus:/");
        BGmusic3 = FMODUnity.RuntimeManager.CreateIn-
stance("event:/BGM3");
    }

    private void OnTriggerEnter(Collider musicRoom)
    {
        if (musicRoom.name == "PlayerCharacter")
        {
            BGmusic3.start();
        }
    }

    private void OnTriggerExit(Collider other)
    {
        isBoost = false;
        MasterBus.stopAllEvents(FMOD.Studio.STOP_MODE.ALLOWFADEOUT);
    }
```

Esimerkkikoodi 4. Äänitapahtumainstanssin luonti, pää-äänilinjan määrittely ja instanssien pysäyttäminen sen kautta sekä musiikkikappaleen vaihtumistriggeriä seuraava totuusarvomuuttuja ja sen nollaaminen pelihahmon poistuessa tilasta.

Esimerkissä transitiokappaleen trigger on peliolioon liitetty skripti. Olion colliderin törmätessä pelihahmon collideriin skriptissä kutsutaan edellä mainitun BGM3-skriptin metodia ”Boosted()”. Esimerkkikoodissa 5 tämä metodi asettaa taustamusiikkitapahtuman uuden saapumispisteen aikajanan 7 sekunnin kohdalle, joka on transitiokappaleen lähtöpiste. Transitiokappaleen loputtua kappale siirtyy FMOD:ssa säädettyjen logiikkaraitojen avulla automaattisesti viimeiseen kappaleeseen.

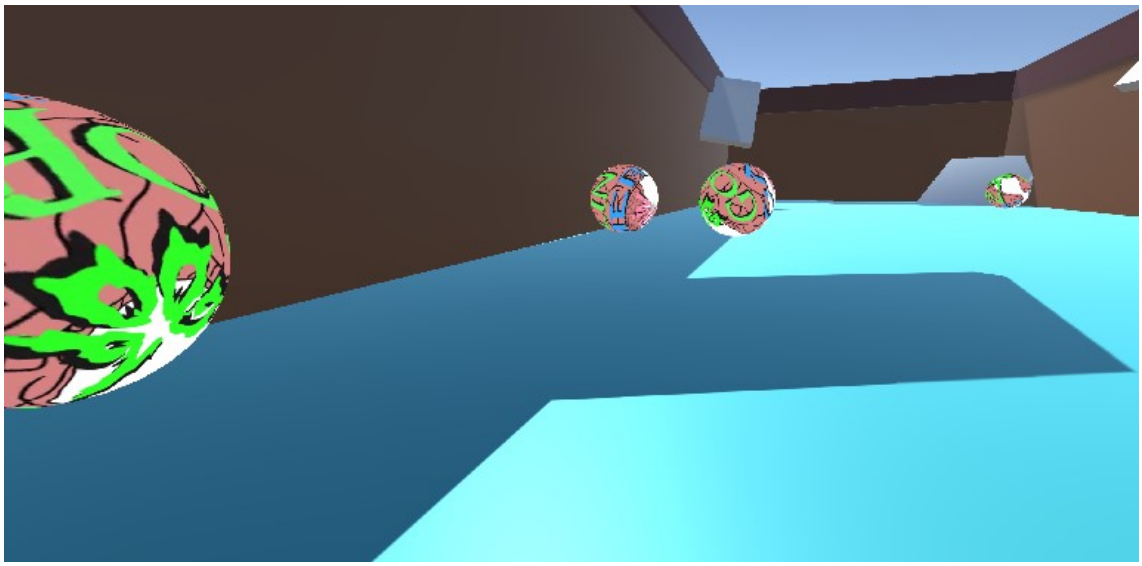
```
public void Boosted()
{
    if (isBoost == false)
    {
        isBoost = true;
        BGmusic3.setTimelinePosition(7000);
    }
}
```

Esimerkkikoodi 5. Metodi kutsuu FMOD:n metodia ”setTimelinePosition”, jonka avulla äänitapahtuman toistokohta vaihtuu aikajanalla transitiokappaleen alkuun. Alussa määritelty ”isBoost”-totuusarvomuuttuja varmistaa, että transitiokappaleeseen siirrytään vain ensimmäisestä triggerin laukaisusta, kunnes pelihahmo poistuu huoneesta.

Nyt pelihahmon tullessa huoneeseen taustamusiikki alkaa soida. Lattialla olevan laatan päälle astuessa taustamusiikki vaihtuu välittömästi nopeutuvaan, pari sekuntia kestävään transitiokappaleeseen, josta se ponnistaa hetken hengähdettyään uuteen, hektisempään versioon alkuperäisestä taustamusiikista. Pelihahmon poistuessa ovesta taustamusiikki lakkaa ja äänitapahtuma palaa alkuperäiseen lähtöasetelmaansa.

4.5 4. osa: Pelihahmon törmäyksestä nopeutuva kappale

Neljännessä esimerkissä taustamusiikki nopeutuu pelihahmon koskiessa palloon. Musiikki nopeutuu saman määrän jokaisesta kosketuksesta senhetkisestä musiikin nopeudesta lähtien. Tempon sijaan musiikin nopeutta vaihdetaan sävelkorkeuden mukaisesti, eli lähtöäänen taajuutta nopeutetaan. Taajuuden aallonpituuksien välien lyhetessä myös sävelkorkeus muuttuu jokaisella nopeutuksella korkeammaksi. Kuvassa 13 näkyy esimerkin testihuone musiikin taajuutta nopeuttavine palloineen.



Kuva 13. Neljännen esimerkin huone. Taustamusiikki nopeutuu joka kerta, kun pelihahmo koskettaa palloa.

Ideana on pelitilanne, jossa pelaajan tarkoitus on ohjastaa pelihahmoa väistämään vieriviä palloja. Pallon osuessa pelaajaan musiikki nopeutuu ja samalla kasvattaa levottomuuden ja painostuksen tunnetta.

Sävellys

Neljättä esimerkkiä varten toteutettu taustamusiikkikappale koostuu kahdesta instrumenttiraidasta: elektronisesta bassoäänestä ja lukuisia eri ääniä käyttävästä elektronisesta rumpusetistä. Hienovaraisempia harmoniaa luovia instrumentteja ei käytetä, sillä yksityiskohtaiset äänet päätyvät helposti kuulostamaan karkeilta ja litistetyiltä jälkiprosessoitujen taajuusmuutoksien seurauksena. Yksinkertainen bassoääni, jolla ei ole nousu- tai laskuvaihetta sävelen alussa ja lopussa, kuulostaa pitkälti samalta jälkiprosessoidulla taajuusmuutoksella kuin tavallisesti samalta korkeudelta soitettuna. Sama koskee elektronisia rumpuja, jotka eivät lyhyillä iskuilla ehdi tavanomaisesti luoda sellaista ääntä, joka kuulijan korvassa kuulostaisi nopeutettuna litistyneeltä.

Kappale on lyhyt neljän tahdin kierto. Koska moniäänisiä sointuja ei käytetä, kaikki sointuharmoniaa implikoiva syntyy bassoinstrumentin melodiaraidasta.

Musiikkikappaleen voi mieltää yhden soinnun sointukuluksi. Kappaleen tempo on kohtalaisen hidas, jotta musiikilla on myöhemmin tilaa nopeutua useaan kertaan, ilman että kokonaisuus alkaa kuulostaa sekavalta.

FMOD

Taustamusiikkikappaletta ei tarvitse jakaa erillisiin raitoihin, joten riittää, että DAW:ssa kappale renderöidään yhdeksi musiikkitiedostoksi. Tämä äänitiedosto tuodaan FMOD:iin uuteen ääniaikajana-tapahtumaan.

FMOD:n puolella muita muutoksia ei tehdä. Laajemmissa projekteissa ja mahdollisesti laajemmissa käyttötarpeissa äänitapahtumalle voi tehdä esimerkiksi uuden parametrisivun, jonka parametrimuutokset sidotaan pää-äänilinjan taajuuden parametriautomaatiolinjaan. Projektin esimerkissä käytetään kuitenkin FMOD:n valmiita metodeja taajuuden muuttamiseen Unityn puolella.

Unity

Unityssa äänitapahtuman toiminta alustettiin pitkälti edellisen esimerkin tavoin. Pelialueelle tippuu palloja joiden prefabiin on liitetty skripti, joka kollisiossa pelaajan kanssa kutsuu ”SpeedUp()”-metodia taustamusiikkiobjektin skriptiltä.

Taustamusiikkiobjekti on jälleen box collider -triggerin avulla määritelty alue, jossa taustamusiikin kuuluisi soida. Prefab-pallot ovat myös huoneen sisällä. Musiikki pysäytetään pää-äänilinjan katkaisemisella kuten edellisessä esimerkissä, ja ääni-instanssin äänentoistometodia kutsutaan pelihahmon törmätessä huoneen box collideriin. Määrittelyvaiheessa määritellään myös float-parametri ”currentPitch”, jonka avulla koodi pystyy tallentamaan nykyisen arvon ja korvaamaan sen korkeammalla arvolla musiikkikappaleen nopeutuessa.

Esimerkkikoodissa 6 on yksinkertainen metodi ”SpeedUp()”, joka kutsuttaessa lisää nykyistä taajuutta merkitsevään float-arvoon 0,1. Taajuus ei suoraan käänny tempomuutokseksi, vaan nopeus on sovellettava taajuuden mukaan.

Arvo 1.0 vastaa taajuuden frekvenssin oletusarvoa, ja 2.0 vastaa frekvenssin kaksinkertaistumista.

```
 public void SpeedUp()
{
    currentPitch += 0.1f;
    BGmusic4.setPitch(currentPitch);
}
```

Esimerkkikoodi 6. ”SpeedUp()”-metodi, joka kutsuttaessaan nopeuttaa taustamusiikin taajuuden frekvenssiä oletusarvon kymmeneksellä.

Pelihahmon astuessa huoneeseen taustamusiikki alkaa soida. Pelialueella pyörii palloja, joiden törmätessä pelihahmoon taustamusiikki nopeutuu ja luo enenevästi paniikinomaista vaaran tuntua. Valmiissa pelissä osuminen palloon voisi aiheuttaa visuaalisia efektejä ja ääniä painottamaan osumista ja näin myös selventämään pelaajalle pelihahmon suhdetta väistettäviin palloihin.

5 Dynaamisen videopelimusiikin toteutuksen analyysi

5.1 Projektin haastavuus

FMOD:n kaltaisen, valmiiksi pelimoottoriin integroidun äänimoottorin käyttöönotto on hyvin nopea prosessi, ja se on valmis ideoiden nopeaan testailuun peliympäristössä välittömästi asentamisen jälkeen. Itse käyttöliittymään ja ominaisuuksiin totutteleminen voi viedä oman aikansa. Suurin osa äänenmuokkausominaisuuksista on välittömästi tuttuja osaaville äänisuunnittelijoille, mutta esimerkiksi totuttelu eri äänitapahtumien ominaisuuksiin, parametrisivujen luontiin ja muokkaukseen sekä äänipankkien käyttöön voi viedä aikansa. Kaiken kaikkiaan äänimoottorin käyttö oli kuitenkin varsin kivutonta insinöörityöprojektin laajuudessa.

Projektin suurimmat haasteet tulivat äänitapahtuman muovaamisesta Unityn skriptien kautta. Projektia toteutettaessa FMOD:n dokumentaatio oli osittain vanhentunutta, mikä johti siihen, että tietyt parametrejä koskevat metodit jouduttiin etsimään muualta. Yksinkertaisemmat ja äänimoottoreille oleellisimmat

ominaisuudet, kuten instanssien luonti tai äänitapahtumien käynnistys ja sammutus, olivat kuitenkin pitkälti vaivattomia ja selkeästi dokumentoitu skriptien kautta käytettäviksi.

Taustamusiikin luomista varten sävellystyötä oli tarkasteltava pelimusiikin dynamiikan näkökulmasta. Jokaisella esimerkillä oli omat tekniset ja musiikkiteoreettiset tarpeensa taustamusiikin puolesta, joten ennen sävellystä täytyi erikseen harkita esimerkiksi sitä, millaisia ääniraitoja taustamusiikki tarvitsee, että jokainen osa parametriautomaatiota kuulostaisi hyvältä, millaisella sointukululla saataisiin aikaan kierto, joka toimii myös musiikkikappaleen loikatessa eri sektioon tai millaiset instrumentit eivät jää taistelemaan frekvenssitilasta, vaikka dynamiikka toisikin esiin uuden instrumenttiraidan parametrimuutosten avulla. Projektin esimerkeissä taustamusiikki soi yksinkertaisissa huoneissa ilman laajemman pelin kontekstia, joten musiikkikappaleiden yleistunnelman puolesta säveltäessä oli vapaammat kädet.

5.2 Havaintoja toteutuksesta

Projektissa tutkittiin erilaisia dynaamisen videopelimusiikin toteutustapoja, mutta FMOD:n kaltaisen äänimoottorin ominaisuuksilla pystyy toteuttamaan kokonaisvaltaisesti pelin äänimaailmaa. Ääniefektejä voi toistaa samoilla ominaisuuksilla kuin musiikkiakin, joten äänimoottorin perusteet oppiessaan äänisuunnittelija pystyy jo toteuttamaan yksinkertaisia ääniefektien, musiikin ja ympäristöäänien äänimaailmakokonaisuuksia.

Äänimoottorit käyttävät yleisesti samankaltaisia ominaisuuksia kuin tavanomaiset DAW:t, joten monet käyttöliittymäelementit käyttävät samoja periaatteita. Näin esimerkiksi äänitasojen muokkaus, ääniefektien lisääminen ja automaatioraidat ovat helposti osaavan äänisuunnittelijan tai säveltäjän käytettävissä ilman erityistä opettelemista. Automaatioraitojen ymmärrys etukäteen auttaa myös äänisuunnittelijaa oppimaan dynaamisten parametrimuutoksien luomista pelillisyyden kautta.

Äänikanavien käsittelylle pelimoottorin puolella on lukuisia eri tapoja, ja näitä tapoja voi käyttää samanaikaisesti toistensa kanssa. FMOD:n valmiilla Unity-oliokomponentilla Event Emitter pystyy käynnistämään ja pysäyttämään äänitapahtumia yksinkertaisesti ja nopeasti, kun taas skriptien kautta äänitapahtumille pystyy antamaan yksityiskohtaisempia sääntöjä. Skriptien kautta voi esimerkiksi muovata äänitapahtuman instanssin sijasta pää-äänilinjaa, joka vaikuttaa jokaiseen hetkellä käynnissä olevaan instanssiin.

Pelimoottorin olioihin liitetyt, FMOD-äänitapahtuman parametreihin vaikuttavat komponentit muuntavat ääntä peliprosessin aikana äänisuunnittelijan antamien parametrisivujen tai metodien mukaisesti. Unityn puolella äänitapahtumia laukaisevia triggereitä voi olla mikä tahansa tavanomainen parametrimuutos pelimoottorissa, eli mahdollisuuksia äänen muovaamiseen pelillisyyden kautta on käytännössä rajattomasti. Äänien suunnittelu on virtaviivaista, ja muunnokset äänitapahtumassa on helppo testata sekä äänipankkien koonnin jälkeen Unityssa että FMOD-projektin sisällä käyttäen käyttöliittymän parametrisivujen liukusäätimiä.

Dynaamisen pelimusiikin säveltäminen vaatii ymmärrystä siitä, mitä haettu dynamiikka tarvitsee, millaisessa vaiheessa peliä musiikki soi ja millaista vaikutusta sillä halutaan pelaajalle tuottaa. Esimerkiksi vertikaalista uudelleen orkestrointia hyödyntäessä musiikkikappale olisi nauhoitettava ja renderöitävä niin, että muunnettavat osat ovat äänimoottoriin tuodessa omina äänitiedostoinaan. Dynamiikkaa pohtiessa ei riitä, että kappaleen miksaus on sopiva tietyssä kokoonpanossa, vaan kappaleen on kuulostettava hyvältä jokaisessa dynaamisen muunnoksen vaiheessa.

5.3 Dynamiikan implementointi peliprojekteissa

Dynaamista videopelimusiikkia pohtiessa äänisuunnittelijan täytyy nähdä pelialue ja siellä tapahtuva pelillisyys kokonaisuutena, jota taustamusiikin tulee tukea mahdollisin keinoin. Dynaamisesti muuntuva taustamusiikki on vain yksi näistä äänisuunnittelijan keinoista, joita voi käyttää sekä hienovaraisesti että

painavasti. Muuntuva musiikki ei aina tuo haluttua vaikutusta, ja musiikin yliampuva muovautuminen saattaa joskus jopa heikentää alkuperäisen tarkoitusperän vaikutusta. Siksi onkin tärkeää, että äänisuunnittelijalla on työkalut dynaamisen muutoksen vaikutuksien nopeaan ja vaivattomaan testailuun peliprojektissa ja niihin liittyvien parametrien nopeaan hienosäätöön. Tällöin tekninen toteutus on mahdollisimman vähän jatkuvan sävellystyön tiellä ja äänisuunnittelijalla on aikaa kuulla ja harkita muutoksien vivahteita.

FMOD:ssa on työkalut sekä horisontaalisen uudelleen sekvensoinnin että vertikaalisen uudelleen orkestroinnin toteutustavoille. Vertikaalista orkestrointia varten käyttöliittymästä löytyvät automaatio-ominaisuudet, ja horisontaalista sekvensointia voidaan toteuttaa eri tavoin joko hyödyntämällä mikserin lähtö- ja saapumispisteitä logiikkaraidoilla tai vaihtamalla pelimoottorin triggereillä äänitapahtumainstansseja. Tämä mahdollistaa kaikenlaisten yleisesti tunnettujen dynaamisen pelimusiikin ominaisuuksien käytön ja niiden jalostamisen huomattavasti yksinkertaisemmassa ympäristössä kuin vielä pari vuosikymmentä sitten.

5.4 Projektin hyöty

Insinöörityön tarkoitus oli osoittaa, millaisin dynaamisen pelimusiikin toteutuksen keinoin pelin taustamusiikin voi valjastaa tehokkaammin ilmaisemaan tunnelmaa, liikkeen tuntua ja toimintaa kuin pelkkien tavanomaisten sävellyskeinojen avulla. Työssä toteutettu projekti antaa erilaiset esimerkit mahdollisista toteutustavoista ja niiden käytöstä ja näin avaa aloitteleville äänisuunnittelijoille lähtöpisteen omien peliprojektien musiikkikirjon dynamisoimiseen ja mahdollisuuden pohtia parannuksia staattisen taustamusiikin vaikutukseen pelissä dynamiikan keinoin.

Projekti myös näyttää, kuinka suuri osa dynaamisen musiikin tehokeinoista on nykyään helppopääsyisiä ja myös pienen mittakaavan ja budjetin omaaville indie-kehittäjille. Dynaamisen pelimusiikin toteuttamisen tekniset rajoitteet ovat vähentyneet merkittävästi uusien käyttäjäystävällisten äänimoottoreiden

seurauksena, ja monet ammattilaiskäyttöön luoduista äänimoottoreista ylläpitävät halpoja tai ilmaisia lisenssejä rajattujen resurssien indie-kehittäjille.

Lopulta projektissa luodut esimerkit ja niiden toteutustavat toimivat äänisuunnittelijalle ponnahduslautana laajemmille ja erilaisille dynaamisen musiikin tilanteille. Näiden neljän esimerkin toteutustavat sisäistämällä äänisuunnittelija pystyy käsittelemään horisontaalisen uudelleen sekvensoinnin ja vertikaalisen uudelleen orkestroinnin periaatteita seuraavia dynamiikan toteutustapoja. Niitä kerrostamalla ja soveltamalla pelimusiikkiin voidaan luoda lukematon määrä monipuolisesti dynaamisia hetkiä.

6 Yhteenveto

Musiikin osuus videopeleissä on kokenut monenlaiset vaiheet videopelien alkuajoista lähtien. Taustamusiikin saatua progressiivisesti lisää painoarvoa videopelien äänimaailman kokonaisuudessa videopelimusiikille on hiljalleen muodostunut omat ominaisuutensa ja ominaispiirteensä. Dynaamisesti pelaajan toiminnan mukaan muovautuva musiikki on videopeleille uniikki tehokeino, jolla voidaan saada aikaan mielenkiintoisia ja pelattavuutta tehostavia hetkiä.

Dynaamisen musiikin implementointi Unityn kaltaisiin pelimoottoreihin on nykyisellään verrattain helppoa pelimoottoriin integroitujen äänimoottorien ansiosta, ja esimerkiksi FMOD:lla riittää joustavuutta kantaa niin pienien indie-projektien kuin isojen julkaisijoidenkin peliprojektien äänimaailmoja. DAW-tyyppinen mikserilinjasto tekee myös ääniraitojen kanssa työskentelystä helppoa ja monentyyppiseen dynamiikkaan tarvittava parametriautomaatioon siirtymä on ääniohjelmien kanssa työskentelevälle äänisuunnittelijalle yksinkertaista.

Pelin äänisuunnittelijan kannattaa tutustua äänimoottorien ominaisuuksiin jo varhaisessa vaiheessa pelinkehitystä ja dynaamisen musiikin keinoihin aiemmin julkaistuissa peleissä. Kun äänimoottori toimii tehokkaassa vuorovaikutuksessa valitun pelimoottorin kanssa ja äänisuunnittelija ymmärtää äänimoottorin toimintakapasiteetin, dynaamiselle musiikille voi löytää uusia käyttötapoja jo ennen

sävellysvaihetta, mikä antaa säveltäjälle aikaa harkita musiikkikappaleen suhdetta dynamiikkaan.

Lähteet

1 Clark, Andrew. 2007. Defining Adaptive Music. Verkkoaineisto. Gamasutra. <https://www.gamasutra.com/view/feature/1567/defining_adaptive_music.php?print=1>. Luettu 10.9.2021.

2 Yarwood, Jack. 2019. Playing Your Song: The Evolution of Dynamic Music in Games. Verkkoaineisto. Electronic Gaming Monthly. <https://egmnow.com/playing-your-song-the-evolution-of-dynamic-music-in-games/>. Luettu 10.9.2021.

3 Collins, Karen. 2014. 10 Inventions That Changed the History of Game Sound. Verkkoaineisto. A Sound Effect. <https://www.asoundeffect.com/10-inventions-that-changed-the-history-of-game-sound/>. Luettu 13.9.2021.

4 Collins, Karen. 2008. In the Loop: Creativity and Constraint in 8-bit Video Game Audio. Twentieth Century Music 2.4.2008, s. 212.

5 Decuir, Joe. 2015. Atari Video Computer System: Bring Entertainment Stories Home. IEEE Consumer Electronics Magazine 15.7.2015, s. 59–66.

6 Making Tracks: The Noble Art of Game Music. NEXT Generation no. 3 March 1995, s. 50.

7 Bird, Dave. 2011. What Are SoundFonts? Verkkoaineisto. Abyssmedia. <https://www.abyssmedia.com/midirenderer/soundfonts.shtml>. Luettu 13.9.2021.

8 Introduction to the MIDI Standard. Verkkoaineisto. Indiana University Bloomington. <https://cecm.indiana.edu/361/midi.html>. Luettu 15.9.2021.

9 Greening, Chris. 2010. Interview with David Wise. Verkkoaineisto. Square Enix Music Online. <https://www.squareenixmusic.com/features/interviews/davidwise.shtml>. Luettu 15.9.2021.

10 Totilo, Stephen. 2020. I Love How Banjo-Kazooie's Music Changes As I Play. Verkkoaineisto. Kotaku. <https://www.kotaku.com.au/2020/11/i-love-how-banjo-kazooies-music-changes-as-i-play/>. Luettu 15.9.2021.

11 Iwata, Satoru. Iwata Asks: Super Mario Galaxy. Verkkoaineisto. Nintendo. <https://iwataasks.nintendo.com/interviews/#/wii/super_mario_galaxy/2/0>. Luettu 17.9.2021.

12 Johnson, Astrid. 2018. Super Smash Bros Ultimate Direct: 900 Music Tracks, Music Customization, and More. Verkkoaineisto. Gamerevolution. <https://www.gamerevolution.com/news/415951 >. Luettu 20.9.2021.

13 Shibuya, Nobuaki. 2019. “Dai rantō sumasshuburazāzu SPECIAL” bōdaina asetto no birudo jikan & yōryō to notatakai: CEDEC 2019. Verkkoaineisto. IGN Japan. <https://jp.ign.com/smash-brothers-special/38416/feature/specialcedec-2019>. Luettu 20.9.2021.

14 Super Smash Bros.™ Ultimate – Battling with Sora – Nintendo Switch. 2021. Verkkoaineisto. Nintendo. <https://youtu.be/L-q6Gz_4Yqc?t=2206>. Luettu 20.9.2021.

15 Hanbali, Medhat. How to write Adaptive Music for Video Games. Verkkoaineisto. Professional Composers. <https://professionalcomposers.com/how-to-write-adaptive-music-for-video-games/>. Luettu 8.10.2021.

16 Duguay, Robby. 2016. Some thoughts on dynamic music in games. Verkkoaineisto. Robby Duguay. <http://www.robbyduguay.com/2016/07/thoughts-dynamic-music-games/>. Luettu 12.10.2021.

17 Mraz, Ronny. 2021. What makes the audio of Mario Kart 8 so fantastic? Verkkoaineisto. Splice. <https://splice.com/blog/mario-kart-8-game-audio-fantastic/>. Luettu 12.10.2021.

18 How Good is Unity for Making AAA Games? 2020. Verkkoaineisto. Juego Studios. <https://www.juegostudio.com/blog/how-good-is-unity-for-making-aaa-games>. Luettu 16.10.2021.

19 FMOD Studio Suite: The adaptive audio solution for games. Verkkoaineisto. FMOD. <https://www.fmod.com/studio>. Luettu 18.10.2021.

20 FMOD Studio. 2021. Firelight Technologies.

21 Featured games. Verkkoaineisto. FMOD. <https://www.fmod.com/games>. Luettu 18.10.2021.

22 2. FMOD Studio Concepts. Verkkoaineisto. FMOD. <https://www.fmod.com/resources/documentation-studio?version=2.01&page=fmod-studio-concepts.html>. Luettu 20.10.2021.

23 3. Organizing the Project. Verkkoaineisto. FMOD. <https://www.fmod.com/resources/documentation-studio?version=2.01&page=organizing-the-project.html>. Luettu 20.10.2021.

24 25. Glossary. Verkkoaineisto. FMOD. <https://www.fmod.com/resources/documentation-studio?version=2.01&page=glossary.html>. Luettu 20.10.2021.

25 17. Event Macros Drawer Reference. Verkkoaineisto. FMOD. <https://www.fmod.com/resources/documentation-studio?version=2.01&page=event-macro-controls-reference.html>. Luettu 20.10.2021.

26 4. Authoring Events. Verkkoaineisto. FMOD. <https://www.fmod.com/resources/documentation-studio?version=2.01&page=authoring-events.html> Luettu 20.10.2021.