Annika Alhokankare

Musiikin visualisointisovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

31.3.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Annika Alhokankare Musiikin visualisointisovellus 36 sivua + 2 liitettä 31.3.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Lehtori Ilkka Kylmäniemi

Insinöörityöprojektina kehitettiin ja otettiin käyttöön musiikin visualisointisovellus, joka visualisoi reaaliaikaisesti musiikista saatavaa MIDI- ja äänitietoa live-esiintymistilanteessa. Sovelluksen tilaaja on muusikko, joka soittaa kosketinsoittimia sooloartistina. Hän tarvitsi musiikkiesityksensä tueksi visuaalisen esityksen, jotta live-esiintymisistä tulisi mielenkiintoinen kokemus, jossa musiikki ja visuaaliset tehosteet toimivat yhdessä luoden mielenkiintoisen ja uuden kokemuksen yleisölle.

Visuaalinen esitys voidaan heijastaa projektorilla esiintymislavan taustakankaalle tai esittää näytöltä. Esityksen aikana sitä ohjailee visuaalisia esityksiä reaaliajassa luova VJ (video jockey), joka valitsee kuhunkin kappaleeseen sopivat visualisoinnit. Sovellusta käytettiin kaksi kertaa live-esitystilanteissa eri tiloissa. Sen todettiin toimivan halutulla tavalla, vaikka kehitysmahdollisuuksia ilmenikin.

Sovellus kehitettiin Unity-ohjelmalla OS X -käyttöjärjestelmillä toimivaksi. Unity on monipuolinen ohjelma, joka mahdollistaa erilaisten sovellusten kehityksen usealle alustalle samaan aikaan. Sovellus perustuu Unity-kohtauksiin, joita siihen koottiin kuusi erilaista. Kohtauksilla ei ole kestoa, vaan ne jatkuvat, kunnes käyttäjä avaa uuden kohtauksen. Sovellus toimii pikanäppäimillä, niin että tiettyä numeronäppäintä painamalla aukeaa uusi esitys. Jokainen kohtaus on oma kokonaisuutensa ja suunniteltu tilaajamuusikon kappaleisiin sopivaksi, mutta koska suurin osa sovelluksen tuottamista visualisoinneista koostuu abstrakteista elementeistä, voidaan sovelluksen esityksistä muokata helposti toiselle muusikolle tai isommalle yhtyeelle sopiva.

MIDI-tietona sovelluksessa visualisoidaan MIDI nuotti päälle -viestin toimintaa, jolla saadaan kosketinsoittimen koskettimen painalluksen nopeustietoa. Digitaalisesta äänitiedosta sovellus visualisoi äänen aaltomuotoa ja taajuusspektrejä. Äänitiedon havaittiin tuottavan varsin informatiivisia, jopa infografiikkaan verrattavissa olevia visualisointeja. MIDI-tieto otettiin sovelluksessa käyttöön lisäosan avulla, kun taas äänitieto vastaanotettiin käyttäen Unityn omia kirjastoja.

Avainsanat	musiikin visualisointi, Unity, MIDI, äänitieto

Author Title Number of Pages Date	Annika Alhokankare Music visualization application 36 pages + 2 appendices 31 March 2017
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Ilkka Kylmäniemi, Senior Lecturer

This project consisted of the development of a music visualization application and its deployment during a live gig. The application was developed for a musician who plays keyboards as solo artist to enable interesting and new experience for his audience. The application visualizes audio and MIDI data in real time with mainly abstract graphics and therefore it can easily be modified to work with other musicians' performances too.

The visual presentation can be projected to a canvas with a projector or streamed on a screen. During a live performance, the application is controlled by a VJ (video jockey) who chooses a fitting scene for each song. The application has been used twice at live gigs and it worked as designed. However, the development will continue because more variety in the visuals is needed.

The application was developed for the OS X operating system with Unity. The visual shows in the application are all different Unity scenes, which will go on until the user opens a new scene.

The application visualizes MIDI Note On messages, which are created when a key on the MIDI keyboard is pressed. The application also visualizes the waveform and frequency spectrum of sound, which were observed to produce almost infographic-like visualizations.

Keywords	music visualization, Unity, MIDI, audio data

Sisällys

Lyhenteet

C#	Oliopohjainen ohjelmointikieli.
DJ	*Disc jockey.* Henkilö, joka soittaa musiikkia yleensä äänilevyiltä tai tietokoneelta. Suomessa käytetään myös nimitystä tiskijukka.
FFT	*Fast Fourier transform*. Nopea Fourier'n muunnos on integraalimuunnos, jota käytetään taajuusanalyyseihin signaalikäsittelyssä.
JIT	*Just-In-Time*. Ohjelman kääntäminen ajonaikaisesti.
MIDI	*Musical Instrument Digital Interface*. Sähköisten musiikkilaitteiden välisen tiedonsiirron standardi.
.NET	Microsoftin kehittämä ohjelmistokehitysympäristö.
OS X	Applen Macintosh-tietokoneiden käyttöjärjestelmäperhe, johon kuuluvat käyttöjärjestelmät Mountain Lion, Maverics, Yosemite ja El Capitan.
USB	*Universal Serial Bus*. Asynkroniseen tiedonsiirtoon perustuva liitäntä.
VJ	*Video jockey.* Ammattinimike reaaliaikaista kuvaa tuottaville esiintyjille.

1 Johdanto

Insinöörityöprojektin tavoitteena on toteuttaa visuaalisia esityksiä tuottava sovellus, jota tullaan käyttämään live-esitystilanteessa musiikkiesityksen taustalla. Sovelluksen tuottama grafiikka elää musiikin mukana reaaliaikaisesti reagoiden muusikon soittimen tuottamaan MIDI (Musical Instrument Digital intrface)- ja äänitietoon. Sovelluksen tilaaja on muusikko, joka soittaa elektronista musiikkia kosketinsoittimilla. Sovellus on tarkoitus ottaa käyttöön joulukuussa 2016 klubikeikalla, jonne muusikko haluaa visuaalisen esityksen musiikkiesityksen tueksi. Visuaalisen esityksen on tarkoitus toimia yhdessä musiikkiesityksen kanssa niin, että ne muodostavat yhdessä yhtenäisen kokonaisuuden tehden esitystilanteesta mielenkiintoisen ja uuden kokemuksen yleisölle.

Vastaavia reaaliaikaisia visualisointeja tuottavia ohjelmia käyttävät tiskijukat esitystensä taustalla. Visualisointeja voidaan heijastaa projektorilla kankaalle tai näyttää ne esiintymislavalle tuoduilta näytöiltä. Esityksestä halutaan personoitu ja tilaajamuusikon kappaleisiin soveltuva, joten kehittämällä oma sovellus alusta alkaen saadaan esityksestä muokattua tilaajan tarpeisiin soveltuva. Sovellus tehdään kuitenkin sellaiseksi, että sitä on mahdollisuus käyttää muidenkin muusikoiden musiikkiesitysten kanssa. Mahdollisuus sovelluksen kaupallistamiseen otetaan huomioon niin, että sen voi tulevaisuudessa myydä muillekin muusikoille käytettäväksi.

Insinöörityöraportissa perehdytään sovelluksen kehitykseen yleisesti Unityllä ja tarkemmin mahdollisuuksiin käyttää musiikista ja äänestä saatavaa tietoa sekä esittelee esimerkkejä äänitiedon visualisoinneista. Musiikista saatavalla tiedolla tarkoitetaan kosketinsoittimista saatavaa MIDI-tietoa ja digitoidusta äänestä saatavilla olevaa äänitietoa. MIDI-tietoa saadaan esimerkiksi sovelluksen tilaajamuusikon käyttämästä kosketinsoittimesta, jonka koskettimen painalluksesta sovellus voidaan ohjelmoida luomaan haluttu visualisointi. MIDI-tieto on hyvin yksinkertaista verrattuna digitoidusta äänestä saatavaan tietoon. Digitaalinen äänitieto on digitoitua ääniaaltoa, josta saadaan mielenkiintoisia ja informatiivisia visualisointeja. Insinöörityöprojektissa selvitetään digitaalisen äänitiedon mahdollisuuksia visualisoinneissa. Reaaliaikainen äänen aaltomuoto tulee olemaan yksi visualisoitava elementti, ja sen lisäksi tavoitteena on saada visualisoitua äänen taajuusspektrejä.

Ohjelman käyttöliittymäksi otetaan tietokoneen näppäimistö, josta tietyillä näppäimillä saadaan näytölle haluttu kohtaus. Sovellukseen on suunnitteilla alle kymmenen komentonäppäintä, joten sen kontrollointi sujuu ilman graafista käyttöliittymää. Sovelluksen käyttäjän eli VJ:n (video jockey) tulee vain osata nämä pikanäppäimet.

Sovellus valmistetaan Unity-ohjelmalla, joka on ensisijaisesti pelimoottori, mutta monipuolisten ominaisuuksiensa vuoksi soveltuu myös tämänkaltaisen sovelluksen kehitykseen. Unity itsessään mahdollistaa äänitiedon vastaanottamisen ja lukemisen. Unityyn otetaan käyttöön lisäosa MIDI-tiedon vastaanottamisen mahdollistamiseksi. Visualisoinneissa Unityssä voidaan käyttää 2D- ja 3D-grafiikkaa, ja se sisältää valmiin editorin partikkeliefekteille, joita tullaan käyttämään visualisoinneissa. Sovellusta varten valmistetaan itsetehtyjä 3D-malleja ja tekstuureja, joiden käyttö Unityssä on myös mahdollista.

2 Musiikin visualisointi yleisesti ja sovelluskehitys Unityllä

2.1 MIDI- ja äänitiedolla kontrolloitavat visuaaliset esitykset

VJ on henkilö, joka tuottaa visuaalisia esityksiä reaaliajassa usein DJ:n eli tiskijukan tuottaman musiikkiesityksen rinnalla. VJ:eiden ja tiskijukkien käyttöön suunniteltuja valmiita musiikin visualisointiohjelmia on olemassa erilaisia, sekä MIDI- että äänitietoa käyttäviä. MusicBeam-ohjelmalla saadaan aikaiseksi projektorilla heijastettava valoesitys. Ohjelmassa on valittavana erilaisia valoesityksiä, jotka reagoivat musiikkiin. Näyttämövalot voi tämän ohjelman avulla siis korvata projektorilla. Bazik on esimerkki ohjelmasta, jolla voi muokata ja esittää reaaliajassa musiikkiin reagoivia visualisointeja. Kuvassa 1 on esimerkki yhdestä Bazik-ohjelman tuottamasta visualisoinnista, joita voi valita valmiista kuviovaihtoehdoista, ja siihen voi ladata myös omia Shader-tiedostoja.

Kuva 1. Bazik-ohjelman tuottama visualisointi (2).

Bazik-ohjelmaa voi kontrolloida myös MIDI-ohjauslaitteilla. Se on ammattimaisille VJ:eille ja tiskijukille suunniteltu ohjelma, jonka kehityksen aloitti ranskalainen DJ Maxime Dreyfus. Resolume on Bazikin kaltainen ohjelma, mutta monipuolisempi. Sillä voi luoda videota, ääntä ja kuvituksia sisältäviä esityksiä, joita pystyy kontrolloimaan MIDI-liitäntäisillä laitteilla, ja esityksestä voi tehdä yhtä aikaa usealla näytöllä esitettävän. Ohjelmalla pystyy sekoittaman videota ja kuvaa keskenään esityksen aikana. (1; 2; 3.)

MIDI-ohjauslaitteita käytetään teatteri- ja konserttiesityksissä esimerkiksi väliverhojen, pyrotekniikan ja valo- ja äänimiksereiden kontrollointiin. Tähän tarkoitukseen ovat olemassa MIDI Show Control (MSC) -viestit (4, s. 172). Monet valopöydät sisältävät MIDI-

liitännät, joiden avulla voidaan ohjailla näyttämövaloja johtamalla esimerkiksi viesti bassorummuniskusta muuntimen kautta valopöytään (4, s. 130).

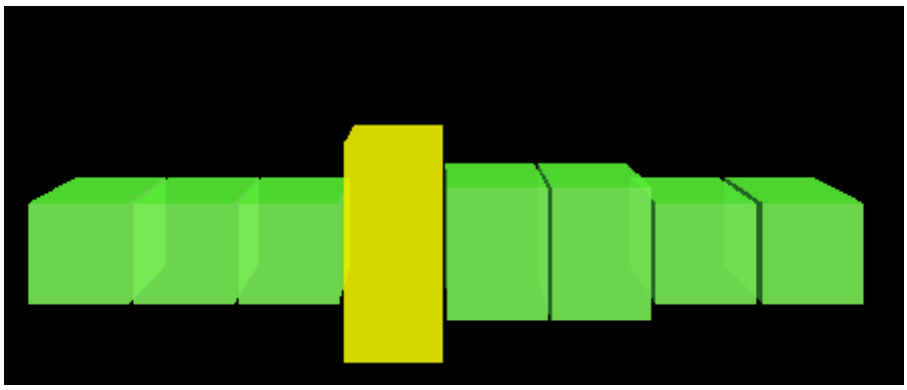
2.2 Sovelluksen visualisointien perusteet

Ihmisen kuuloalue, äänen voimakkuuden eli paineen vaihtelujen havainnointialue ja taajuuksien havainnointiskaala, on huomattavan laaja. Ihmisen korva kykenee havainnoimaan kuuloaistimuksia yhdeksän kertaa isommalta taajuusalueelta kuin silmä näköaistimuksia. (5, s. 79.) Näköaisti päihittää kuitenkin kuuloaistin tiedon tuottamistiheydessä, sillä sen on arvioitu tuottavan joka hetki aivoihin kahdeksan kertaa enemmän tietoa kuin kaikki muut aistit yhteensä (6, s. 17). Sanaa ”visualisointi” käytetään kuvaamaan sekä prosessia, jossa tietoa muutetaan visuaaliseen muotoon, että tämän prosessin lopputulosta. Sanalle ”visualisointi” on visualisointitutkija Robert Kosara esittänyt seuraavan määritelmän:

- Visualisointi perustuu (ei-visuaaliseen) dataan.
- Visualisointi tuottaa lopputuloksenaan kuvan.
- Lopputuloksen tulee olla tulkittavissa ja tunnistettavissa.

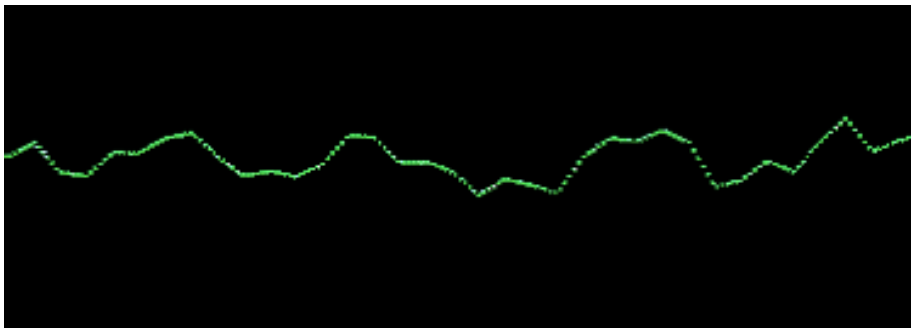
Infografiikka on tietoa välittävää, selittävää, viestintää tukevaa grafiikkaa, joka kuvaa usein numerotietoa tai sellaiseksi muutettavissa olevaa tietoa. (6, s. 20, 23.)

Visualisointityyliltään osa sovelluksen elementeistä on määriteltävissä infografiikaksi, jossa käytetään tilastografiikan kuvitustyyppejä, tosin vähän tyyliteltyinä. Kuvan 2 kuvakaappauksessa sovelluksesta on spektrianalyysiin perustuva pylväskokoelma, jota kutsutaan tilastografiikassa logaritmista asteikkoa käyttäväksi pystypylväskuvioksi. Logaritmisia asteikkoja ei tulkinnan vaikeuden vuoksi yleensä käytetä infografiikassa. Tässä asteikossa vaaka- tai pystyakselilla olevia arvoja ei ole järjestetty lineaarisen tasaisesti vaan logaritmisesti. Visualisoinnissa on lisäksi korostettuna suurimman arvon kulloisellakin hetkellä omaava pylväs keltaisella värillä.



Kuva 2. Taajuusspektrin visualisointi.

Kuvassa 3 on äänen aaltomuotoa kuvaava visualisointi, joka on tilastokuvioiden perustyypeistä ehkä käytetyin eli viivakuvio. Se soveltuu hyvin etenkin jatkuva-arvoisen tiedon esittämiseen. Tavallaan nämä visualisoinnit ovat reaaliaikaisesti päivittyvää infografiikkaa. (6, s. 186, 217, 190.)



Kuva 3. Äänen aaltomuodon visualisointi.

Sovellus visualisoi ei-visuaalista dataa; se ei pyri välittämään tietoa, vaikka sen luomissa visuaalisissa esityksissä käytetään ääni- ja MIDI-tietoa. Sovelluksessa visualisoitu äänenaaltomuoto on informaatiota sisältävä visualisointi, mutta katsojan silmissä abstrakti elementti. Toki musiikin teoriaa tuntevat saattavat ymmärtää, mitä tietoa äänestä kulloinkin on visualisoitu. Tavoitteena on tuoda esitystilanteeseen lisäelementtejä ja tunnelmaa. Esitystilanteisiin, joissa sovellusta käytetään, odotetaan yleisön tulevan katsomaan ensisijaisesti muusikkoa eikä visuaalista esitystä, joten se ei saa toisaalta olla liian hallitseva elementti kokonaisesityksessä. Muutamia poikkeuksia lukuun ottamatta visuaalinen esitys sisältää vain abstrakteja muotoja, värejä ja valoja. Poikkeuksina ovat tikkuukkohahmo, joka kulkee toistuvasti ruudun halki, laatikoista muodostuva kaupunkimaisema ja Pac-Man -hahmon näköinen olio. Musiikkivideomaisia kerronnallisia kohtauksia ei esityksissä ole.

Sovelluksen kehitykseen käytetyt ohjelmat

Sovellus tehtiin Unity-ohjelman versiolla 5.4. Siinä on monipuoliset mahdollisuudet luoda 3D- ja 2D-grafiikkaelementtejä ja antaa niille toimintoja. Sovelluksessa eri objektit liikkuvat ja muuttavat kokoaan ja värejään MIDI- ja äänitiedon mukaan. Unityssä eri elementtejä voi liikuttaa myös fysiikan sääntöjen mukaan voimia käyttäen tai ohjelmoimalla niille liikeratoja. Unityllä voi tehdä monipuolisesti sovelluksia eri alustoille.

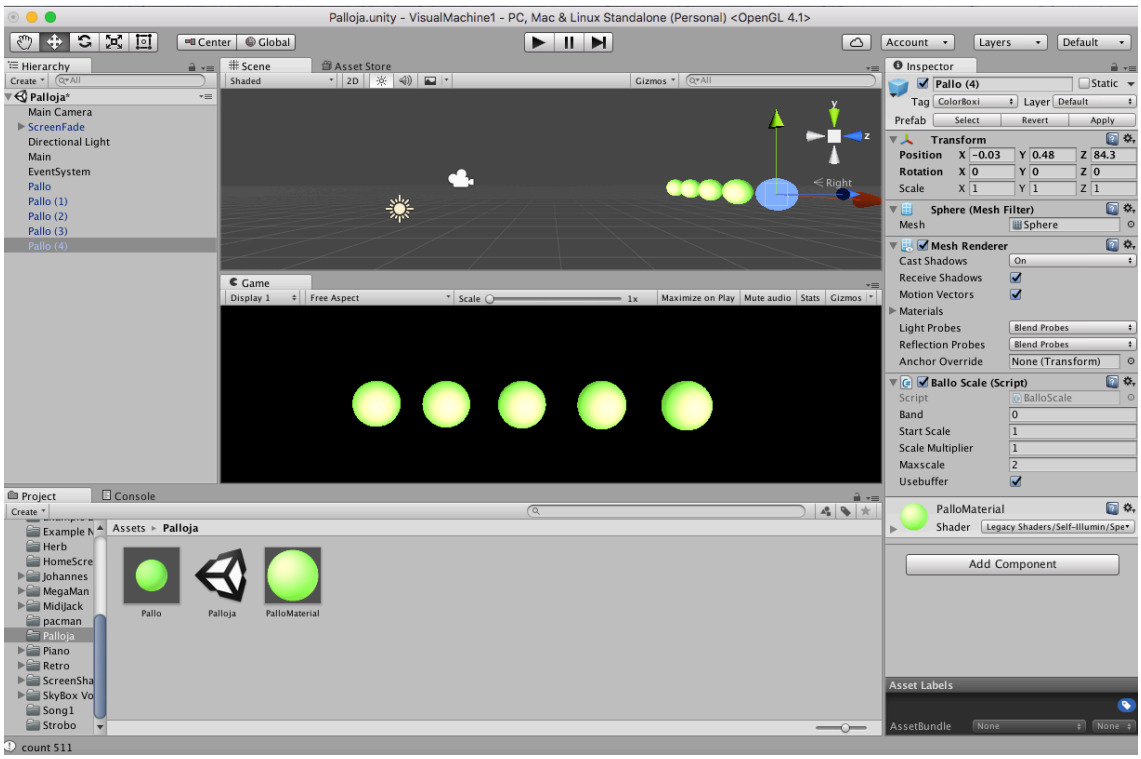
Sovelluksen valmistukseen käytettiin Unityn lisäksi grafiikkaohjelmia. Blender-ohjelmaa käytettiin 3D-mallien valmistukseen. Yksinkertaisimmat 3D-mallit, kuten kuutiot ja pallot, tehtiin Unityssä. 2D-grafiikkaa ja tekstuureja 3D-malleille tehtiin Photoshop CC- ja Illustrator CC -ohjelmilla.

2.3 Unityn käyttö sovelluskehityksessä

Unity on pelinkehitystä varten suunniteltu pelimoottori, mutta se soveltuu myös muunlaisten sovellusten tekemiseen. Ohjelmasta on olemassa ilmaisversio, jonka toiminnot riittävät hyvin laajojenkin sovellusten kehittämiseen. Ohjelmalla voi tehdä 2D- ja 3D-objekteja, ladata siihen valmiita lisäosia ja objekteja, animoida sekä yhdistää osia koodilla tai graafisessa käyttöliittymässä. Unityllä voi kehittää sovelluksia monelle eri alustalle samanaikaisesti. Työpöytäsovelluksia on mahdollista Unityllä kehittää PC-, Mac- ja Linux-alustoille, ja WebGL-sovelluksia selainympäristöihin. Sovelluksia voi kehittää älypuhelimien ja taulutietokoneiden IOS-, Android-, Tizen- ja Windows-käyttöjärjestelmille, ja näiden lisäksi ohjelmia voi tehdä tvOS-, Xbox-, SamsungTV- ja Playstation-laitteisiin. (7, s. 3.)

Kuva 4 on kuvakaappaus tavanomaisesta Unity-projektin kehitysnäkymästä. Kuvan yläosassa on avoinna kohtaus (Scene) -ikkuna, jossa editoitavana on Palloja-niminen kohtaus kolmiulotteisessa tilassa. Se sisältää muun muassa kameran, valonlähteen ja viisi samanlaista pallo-objektia. Alempana peli (Game) -ikkunassa näkyy kameranäkymä, eli se näkymä, minkä valmiin sovelluksen käyttäjä tulee näkemään. Sinisellä korostettu pallo-objekti on valittuna, ja sen tiedot näkyvät oikean reunan katselmus (Inspector) -ikkunassa, josta näkyy, että objektilla on polygoniverkkona (Mesh) pallon muotoinen kappale, siihen on liitetty BalloSace.cs-niminen skriptitiedosto ja sillä on materiaalina PalloMaterial-niminen materiaali. Objektin sijaintia, kokoa ja suuntaa pystyy muuttamaan

yläreunan siirtymä (Transform) -kohdasta, ja sen sijaintia voi muuttaa myös kohtausikkunassa liikuttelemalla. Kaikkia katselmusikkunan kohtia pystyy myös muokkaamaan objektiin liitettävien skriptitiedostojen kautta. Näin on mahdollista vaihtaa esimerkiksi materiaalin väriä tai objektin sijaintia.



Kuva 4. Unity-editorin ikkuna, jossa on avoinna projekti.

Kuvan 4 tilanteessa kehitteillä on kolmiulotteista tilaa käyttävä kohtaus, mutta Unityllä voi tehdä myös 2D-pelejä.

Unityssä objekteista voi tehdä uudelleen käytettäviä malliobjekteja (prefab). Peliin tarvitaan usein paljon samanlaisia objekteja, joiden pitää olla samaan aikaan käytössä. Tällöin käytetään uudelleen käytettävää malliobjektia. Malliobjektista luodaan ensin yksi mallikappale, joka muutetaan Unity-editorin valikoista prefab-muotoiseksi. Kun yhteen malliobjektiin tehdyt muutokset tallennetaan, tulevat muutokset käyttöön myös kaikkiin muihin samanlaisiin malliobjekteihin. Kuvassa 4 on viisi kopiota samasta Pallo-nimisestä malliobjektista. Objekteja voi lisätä kohtaukseen koodin kautta tai lisäämällä niitä editoriikkunassa kohtaukseen. (7, s. 429.)

Unityssä mittayksikkönä käytetään kohtausikkunassa, objektien luonnissa ja ohjelmoinnissa tunnuksettomia arvoja. Esimerkiksi uusi 3D-peruskappale kuutio on luontihetkellä

jokaiselta sivultaan yhden yksikön pituinen. Jos tarkoitus on luoda reaalimaailman kaltainen tila, on mittakaava hyvä pitää muuttumattomana, niin että yhden metrin korkuinen kappale on myös Unityssä yhden yksikön korkuinen. Tällöin fysiikkaan pohjautuvat toiminnot näyttävät myös luonnollisilta. (7, s. 20.)

Unityssä on erittäin helppo liittää skriptejä eli ohjelmointikielellä kirjoitettuja komentosarjoja editorissa oleviin objekteihin. Käytettävissä on kolme ohjelmointikieltä: JavaScript, C#-kieli ja Boo-kieli, joka on poistettu Unityn ohjelmointidokumentaatiosta Unity 5.0 -version julkaisun jälkeen sen vähäisen käytön vuoksi (8). C# on oliopohjainen vahvasti tyypitetty korkeamman tason ohjelmointikieli. Unityn toteutus JavaScriptistä on erittäin tehokas ajonaikaisesti käännettävä JIT (Just-In-Time) -versio kielestä. Tämä JavaScriptin versio, jota kutsutaan myös UnityScriptiksi, on nopeampi kuin internetympäristössä käytetty dynaaminen komentosarjoihin tukeutuva JavaScript. Unity käyttää Mono-ohjelmistokehitysympäristöä, joka on alusta- ja käyttöjärjestelmäriippumaton pohja .NET-arkkitehtuurille. Käytössä olevat ohjelmointikielet pystyvät käyttämään siihen kuuluvia luokkakirjastoja. .NET-arkkitehtuuri on kiinteästi yhteydessä C#-kieleen, jolla suurin osa sen luokkakirjastoista on toteutettu. Yhdessä Unity-projektissa voi käyttää myös useaa ohjelmointikieltä, ja jopa kutsujen lähettäminen erikielisten tiedostojen välillä on mahdollista. (7, s. 143–143.)

Unityn mukana tulee integroitu ohjelmointiympäristö MonoDevelop-Unity, jossa on tekstieditori ohjelmointikielisen tekstin kirjoittamista varten ja vianetsintämahdollisuudet. Sitä ei ole kuitenkaan pakko käyttää, vaan Unity-projektiin liittyvät ohjelmointitiedostot voi kirjoittaa myös ulkopuolisella ohjelmointieditorilla (7, s. 145). Menardin ja Wagstaffin mukaan (7, s. 144) ohjelmoija voi valita Unityssä käytettävän ohjelmointikielen omien mieltymyksiensä mukaan, sillä kaikki kielet ovat yhtä tehokkaita. Sovelluksen kehityksessä käytettiin ohjelmointikielenä C#-kieltä. C# oli käytännöllinen valinta koko sovelluksen kehitykseen, koska sovelluksessa käytetyn MIDI Jack -ohjelmointirajapinnan komennot ovat käytettävissä vain C#-kielellä (9).

3 Sovelluksen vaatimukset, rakenne ja käyttö live-esitystilanteessa

3.1 Sovelluksen sisältö, käyttöliittymä, toiminnot ja vaatimukset

Valmistettu sovellus visualisoi musiikkia reaaliajassa live-esitystilanteessa näytöltä tai projektorilla heijastettuna. Näytön tai projektorin käyttö riippuu esiintymistilasta ja sen mahdollisuuksista. Sovellus tuottaa erilaisia visualisointeja sen mukaan, mitä informaatiota se saa MIDI-tietoa tuottavasta soittimesta ja äänitiedosta. Vaikka muusikko esittäisi saman kappaleen kaksi kertaa, eivät visuaaliset esitykset ole koskaan samanlaisia keskenään, koska musiikkitiedon lisäksi joidenkin objektien liikkeisiin vaikuttavat myös satunnaislukugeneraattorin antamat arvot. Ohjelmaan tehtiin erilaisia visuaalisia esityksiä, joista valitaan kuhunkin esitettävään musiikkikappaleeseen sopiva kohtaus.

Erilaiset esitykset tehtiin omiksi Unity-kohtauksiksi (scene), joilla on omat käsikirjoituksensa. Sovelluksen esitykset suunniteltiin toimimaan tilaajamuusikon kappaleiden mukana, mutta osa niistä on helposti, joko sellaisenaan tai pienellä muuntelulla, käytettävissä muidenkin musiikkiesitysten kanssa. Ohjelma tehtiin soolomuusikolle, mutta se toimii myös esiintymiskokoonpanolla, jossa on enemmän soittajia. Sovelluksen vastaanottama MIDI-tieto suunniteltiin toimimaan kosketinsoittimien kanssa, mutta äänitietoa käyttävät osat toimivat kaiken äänitiedon kanssa. Visuaalisten esitysten sopivuus toisen esiintyjän musiikkiesitykseen riippuu enemmän toisen esiintyjän esittämän musiikin tyylilajista kuin teknisistä seikoista. Vaikka sovellus ei saisi MIDI-tietoa kosketinsoittimista, kuva ei silti koskaan ole täysin elotonta, vaan sovelluksen visualisoinneissa on myös paljon osia, jotka liikkuvat ilman MIDI- tai äänitietoa.

Esitystilanteessa sovellusta kontrolloidaan pikanäppäimillä. Esityksen aikana sitä ohjaa VJ. Tietokoneen näppäimistön näppäimistä 1–6 avautuu jokaisesta erilainen esitys. Esimerkiksi yhdessä kohtauksessa on käsikirjoitettu ennalta ohjelmoitu kameran liike kolmiulotteisessa tilassa ja samaan aikaan kohtauksessa on reaaliajassa MIDI- ja äänitiedon mukaan kokoaan muuttavia laatikoita ja pilviä. Useassa kohtauksessa se järjestys, jossa elementit, kuten eri partikkeliefektit ja objektit, tulevat näkyviin ja poistuvat näkyvistä, on etukäteen ohjelmoitu ja käsikirjoitettu. Esityksille annettiin nimet taulukon 1 mukaisesti.

Taulukko 1. Sovelluksen kohtaukset ja niiden pikanäppäimet.

Pikanäppäin	Kohtauksen nimi
H	Alkuruutu
L	Logo, kohtaus, jossa artistin logo ja nimi
B	Black, kokonaan musta ruutu
1	Retro
2	Piano
3	PacMan
4	Johannes
5	Castle
6	Disco

Kohtauksien lisäksi sovelluksessa on kokonaan musta ruutu ja alkuruutu, joita on tarkoitus näyttää musiikkikappaleiden välissä olevien taukojen aikana. Kohtauksilla ei ole pituutta, vaan ne loppuvat vasta, kun sovelluksen käyttäjä vaihtaa kohtausta tai palaa ohjelman alkuruutuun. Kohtauksissa käytetään osittain samoja elementtejä ja koodia. Kohtausten välillä on häivytysefekti mustaan, joka tekee siirtymät kohtausten välillä sujuvan näköisiksi.

Sovellus tehtiin Unity-ohjelmalla Mac-tietokoneiden OS X -käyttöjärjestelmälle. Sovellus on mahdollista muuntaa myös esimerkiksi Windows-käyttöjärjestelmässä toimivaksi. Jotta sovelluksen kaikki toiminnot saadaan käyttöön, pitää tietokoneessa olla sisäinen mikrofoni, ulkoinen mikrofoni tai äänitiedon sisääntulo muusta laitteesta ja MIDI-tiedon vastaanottomahdollisuus. MIDI-tietoa voidaan vastaanottaa joko suoraan MIDI-tietoa tuottavasta ohjauslaitteesta tai MIDI-sovittimen kautta.

3.2 Sovelluksen käyttö live-esitystilanteessa

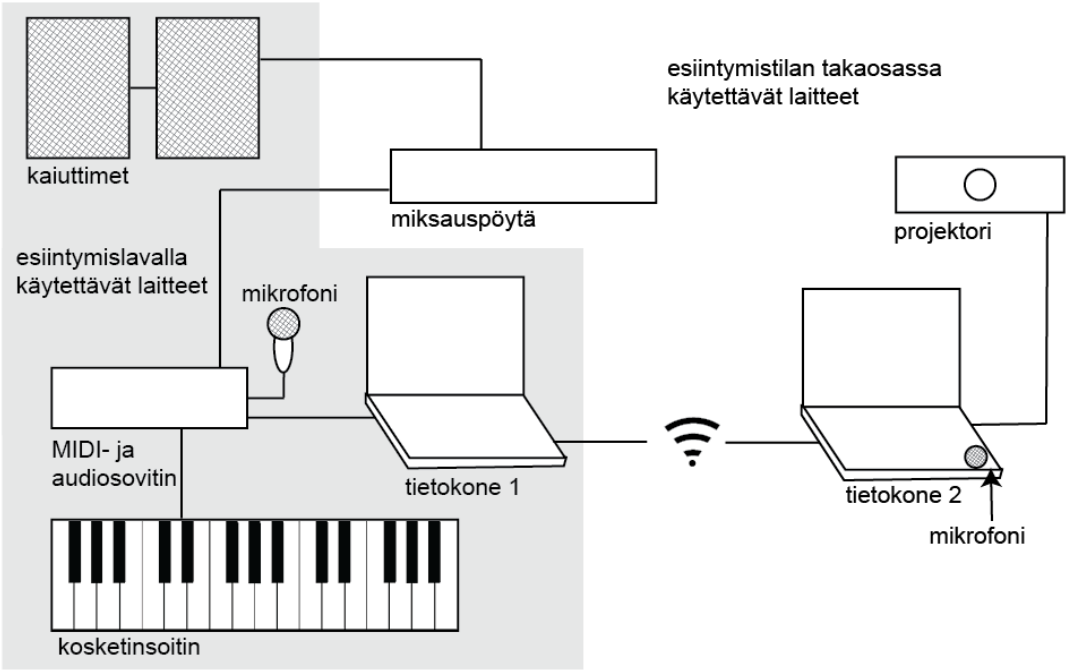
Sovellusta käytettiin kaksi kertaa keikalla. Kuva 5 on otettu toiselta keikalta Lahdessa 3.3.2017. Esiintymispaikassa oli projektori ja sen lisäksi ravintolan puolella kolme näyttöä, joilta yleensä näytetään televisio-ohjelmia. Niitä kaikkia käytettiin visualisointien esittämiseen. Esiintymislavan taustalle asetettiin valkoinen muovinen suojapeite, sillä esiintymislavan taustalla on tavallisesti mustat verhot, joista heijastettu kuva näkyi testeissä huonosti.

Kuva 5. Sovellusta käytettiin esityksessä 3.3.2017 ravintola Tirrassa Lahdessa.

Muusikko, jonka esiintymisiä varten sovellus suunniteltiin, soittaa kosketinsoitinta, laulaa ja käyttää myös taustanauhoja esityksen tukena. Sovelluksen ensimmäisellä käyttökerralla laitekokoonpano oli seuraava:

- kosketinsoitin
- kaksi MacBook pro -kannettavaa tietokonetta
- projektori
- miksauspöytä
- kaiuttimet
- mikrofoni
- audio- ja MIDI-sovitin (interface).

Toisella esiintymiskerralla laitekokoonpano oli melkein samanlainen, mutta silloin käytettiin useita mikrofoneja. Kuvassa 6 on kuvattu ensimmäisessä esitystilanteessa käytetty laitekokoonpano, yhteydet ja sijainnit tilassa. Kosketinsoittimista MIDI-tieto kulkee audio- ja MIDI-sovittimen kautta esiintyjän käytössä olevaan tietokoneeseen (kuvassa 6 tietokone 1). Muusikko ohjaa MIDI-koskettimistolla MainStage-ohjelmaan luotuja virtuaalisia soittimia. MainStage-ohjelmasta valitaan myös soitettavat taustanauhat. MainStage-ohjelmasta äänitieto lähtee audio- ja MIDI-sovittimeen, joka muuttaa digitaalisen äänitiedon analogiseksi. Tämän jälkeen analoginen signaali kulkee miksauspöydän kautta kaiuttimiin.

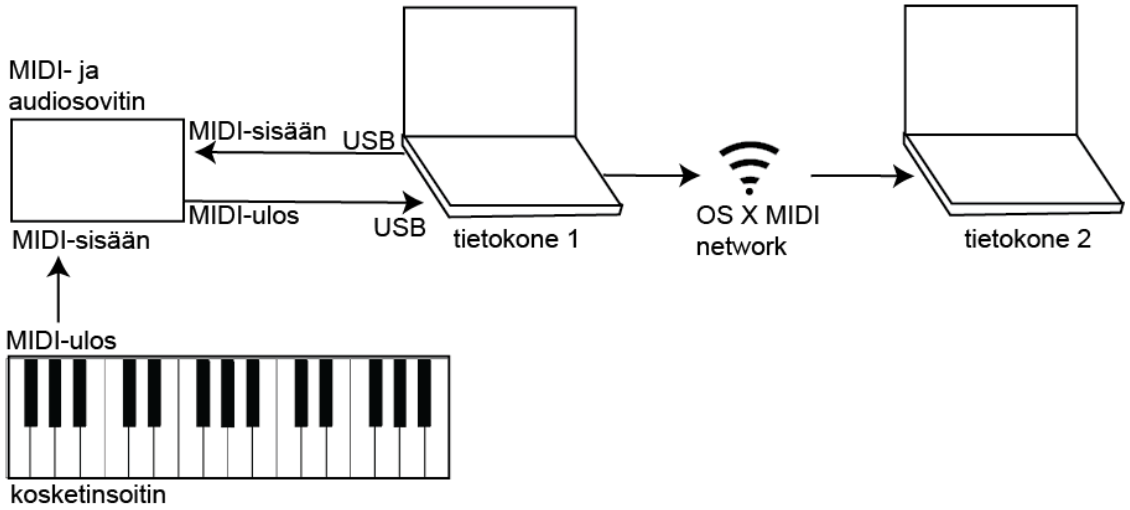


Kuva 6. Esitystilanteessa käytetty laitekokoonpano.

Muusikon tietokoneelta visuaalista esitystä ajavalle koneelle (kuvassa 6 tietokone 2) kulkee MIDI-tieto OS X MIDI Network -verkkoyhteyden kautta. MIDI-tiedon lähetys MIDI Networkin kautta vaatii verkkoyhteyden tietokoneisiin. Tietokoneen mikrofoni vastaanottaa äänisignaalin. On myös mahdollista ottaa äänisignaali miksauspöydän kautta, jolloin esitykseen liittymättömät äänet eivät vaikuta visuaaliseen esitykseen. Esitystilanteessa päädyttiin käytännön syistä käyttämään esitystietokoneen mikrofonia äänisignaalin vastaanottamiseen. Tässä esitystilanteessa esitys heijastettiin projektorilla lavan taustakankaalle.

3.3 MIDI-tiedon käyttö live-esitystilanteessa

Projektin live-esitystilanteissa muusikon koneelle tuli kosketinsoittimesta MIDI-tieto sovittimen kautta. MacBook Pro -tietokoneissa, joita molemmat käytössä olleet koneet olivat, ei ole MIDI-sisään-liitäntää. Mac OS X MIDI interface -rajapinta mahdollistaa MIDI-viestin lähettämisen toiselta OS X -käyttöjärjestelmää käyttävältä koneelta toiselle MIDI Network -verkkoyhteyden avulla. Live-esiintymistilanteessa tätä käytettiin tiedonsiirtoon muusikon tietokoneen ja visuaalista esitystä ohjaavan tietokoneen välillä. Tiedonkulku tapahtui kuvan 7 mukaisesti.



Kuva 7. Live-esitystilanteessa käytetyt MIDI-liitännät.

Kuvan 7 mukaisesti tietokone 1 asetetaan Audio MIDI Setup -ohjelmassa, joka kuuluu käyttöjärjestelmään, lähettämään MIDI-tieto tietokoneelle 2, joka taas vastaavasti asetetaan vastaanottamaan signaali OS X MIDI Network -verkon kautta. Audio MIDI Setup -ohjelmalla voi hallinnoida tietokoneeseen liitettyjä MIDI-, äänentoisto- ja nauhoituslaitteita. Sovelluksen kehitys- ja testausvaiheessa on MIDI-tieto haettu ohjauskoskettimistosta USB-liitännän kautta Mac tietokoneeseen. OS X -käyttöjärjestelmät sisältävät mahdollisuuden lukea MIDI-tietoa ilman tulkitsijaa vastaanottamalla sen MIDI-USB-kaapelin kautta. Kehitysvaiheessa käytettiin Oxygen 49 -kosketinsoitinta, joka pystyy kommunikoimaan OS X -käyttöjärjestelmien Mac OS X Core MIDI -rajapinnan kanssa ilman ulkopuolisia ajureita. Se sopii tämän vuoksi hyvin sovelluksen ohjaamiseen kehitysvaiheessa. (10.)

4 MIDI-tieto ja MIDI-viestit ja niiden käyttö sovelluksessa

4.1 MIDI-tieto

MIDI-tiedon tuottaminen on usein soittamista. MIDI on tiedonsiirtostandardi, joka mahdollistaa kommunikoinnin MIDI-laitteiden välillä. MIDI-tietoa lähettävää laitetta kutsutaan ohjauslaitteeksi ja viestin vastaanottavaa laitetta kohdelaitteeksi. Ohjauslaite voi olla esimerkiksi kosketinohjain, kielisoitintyyppinen ohjain kuten MIDI-kitara tai muu kielisoitin tai perkussio-ohjain eli lyöntialusta. Tässä sovelluksessa kosketinsoitinta käytetään ohjauslaitteena. Se lähettää MIDI-viestin sovittimen ja toisen kohdelaitetietokoneen kautta sovellukselle. Kohdelaitteen tehtävä on useimmiten tuottaa tai muokata ääntä. MIDI ei lähetä äänen aaltomuotoa, vaan viestityypistä riippuen se voi lähettää esimerkiksi sävelen alun, lopun tai sävelkorkeuden. (5, s. 25; 4, s. 13, 21, 32 ja 34.)

MIDI on nopeaa tiedonsiirtoa. Tiedon välittämisestä tulee noin millisekunnin viive, jota ei voi korvalla havaita. MIDI on myös sarjallista, asynkronista tiedonsiirtoa. Sarjallisuus tarkoittaa sitä, että biteistä koostuvat tietopaketit ja viestit kulkevat laitteesta toiseen järjestyksessä, tavu kerrallaan. Asynkronisuus eli tahdistamattomuus tarkoittaa sitä, että tiedon kohdelaite ei ole tahdistettu ohjauslaitteeseen. Kohdelaite lukee ja vastaanottaa saapuvaa tietoa oman kellotaajuutensa tahdistamana. Ohjauslaite lähettää viestin silloin, kun esimerkiksi kosketinsoittimen kosketinta painetaan. (4, s. 22.)

4.2 MIDI-viestit

MIDI-viestityypit

MIDI-viestit luokitellaan kanava- ja laitteistoviesteihin taulukon 2 mukaisesti. Kanavaviesteihin kuuluvat ääni- ja moodiviestit, joita voidaan lähettää eri MIDI-kanavilla halutuille kohdelaitteille. Yleisimpiä MIDI-viestejä ovat ääniviestit, joihin kuuluvat nuotti päällä -viesti (Note On), nuotti sammutettu -viesti (Note Off), ohjainviestit (Control Change), jälkipainoviesti (After Touch) ja soundinvaihtoviesti (Program Change). Ohjainviesti seuraa modulaatiopyörän liikautuksesta, jälkipainoviesti seuraa koskettimen painamisesta sen pohjassa ollessa ja äänen muutosviesti äänen vaihto -näppäimen (Program Change) painamisesta. Nuotti päällä -viesti seuraa esimerkiksi ohjainlaitteena olevan kosketinsoittimen koskettimen painalluksesta ja nuotti pois -viesti seuraa koskettimen vapauttamisesta. (4, s. 22, 24 ja 74.)

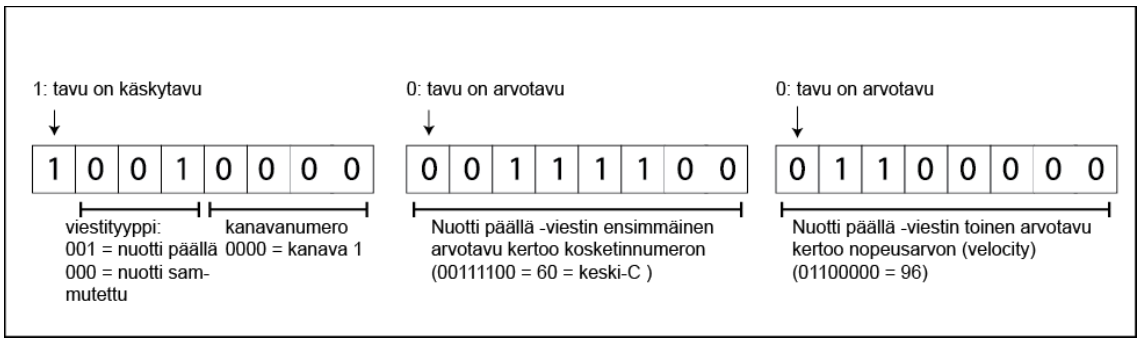
Taulukko 2. MIDI-viestien jaottelu kanava- ja laitteistoviesteihin (4, s. 74).

MIDI-viestit				
Kanavaviestit **Channel Messages**		**Laitteistoviestit** **System Messages**		
Ääniviestit Voice Messages	**Moodiviestit** Voice Messages	**Yleisviestit** System Common Messages	**Tahdistusviestit** System Real-Time Messages	**Erikoisviestit** System Exclusive Messages
Nuottiviestit Note On/Off Ohjainviestit Control Change Jälkipaino Poly Key Pressure/ Channel Pressure Soundinvaihto Program Change Taivutus Pitch Bend	Local On/Off All Notes Off Omni On/Off Mono On/Off	Quarter Frame Song Pos. Pointer Song Select Virityspyyntö Tune Request Erikoisviestin lopetus End Of System Exclusive	MIDI-kello MIDI clock Start Continue Stop Active Sensing System Reset	Manufacturer´s Sys Ex Universal Non-Comercial Sys Ex Universal Non-Real Time Sys Ex Universal Real Time Sys Ex

Laitteistoviesteillä ei ole kanavatunnusta, joten laite ottaa ne vastaan siitä riippumatta, mille MIDI-kanavalle se on asetettu. Laitteistoviesteihin kuuluvat esimerkiksi tahdistukseen ja muistitiedon siirtoon liittyvät viestit. (4, s. 74–75.)

MIDI-viestin rakenne

MIDI-viesti sisältää 1–3 tavua. Tavu on kahdeksan bitin mittainen. Kahdeksalla bitillä voidaan ilmaista lukuarvoja väliltä 0–255. MIDI-viesteissä tavun ensimmäinen bitti ilmoittaa kohdelaitteelle viestin luonteen ja loput seitsemän bittiä riittävät 128 lukuarvon ilmaisemiseen. Esimerkkinä MIDI-viestistä on kuvassa 8 nuotti päälle -viesti, joka sisältää kolme tavua. Kolmesta tavusta ensimmäinen on käskytavu, joka sisältää tiedot viestityypistä ja kanavanumeron. Kaksi muuta tavua ovat arvotavuja, joista ensimmäinen kertoo nuotin numeron ja toinen painalluksen nopeuden (velocity). (4, s. 22–23.) Kaikki koskettimistot eivät lähetä voimakkuusarvoviestiä. Nuotti päälle -viesti, jossa nopeus on 0, sammuttaa nuotin kuten nuotti sammutettu -viestikin. (4, s. 76.)



Kuva 8. Nuotti päällä -viestin rakenne (4, s. 23).

Nuottinumero eli kosketinsoittimen sävelkorkeus on välillä 0–127, jonka skaala on kymmenen oktaavia. Esimerkiksi keski-C:n kosketinnumero on 60 (00111100 eli 3CH) (4, s. 76). M-Audio Oxygen 49 -mallisessa kosketinsoittimessa, jota voi käyttää sovelluksen ohjauskoskettimistona, on neljä oktaavia, joista alin kosketin on nuottinumeroltaan 48 ja ylin 96. Oktaavi on taajuuden kaksinkertaistumista vastaava sävelkorkeuden intervalli, jonka avulla voidaan ilmaista soittimen sävelskaalan pituutta. Jokaisesta pianon koskettimesta saatava sävel on taajuudeltaan noin 6 % suurempi kuin sitä edeltävän koskettimen sävelen taajuus. Sävelasteikko C, D, E, F, G, A, H on yhden oktaavin mittainen ja tätä seuraa oktaavia korkeampi sävelasteikko. Näin samat toisistaan oktaavin päässä olevat äänet kuulostavat harmonisilta. (5, s. 175.)

MIDI-kanavia on 16, ja ne mahdollistavat tiedon lähettämisen erikseen 16 eri laitteelle yhdeltä ohjauslaitteelta. Jos kohdelaite on asetettu Omni On -tilaan, se reagoi kaikilta 16 kanavalta tuleviin viesteihin (4, s. 25–26). Käytännössä sovelluksella olisi mahdollista visualisoida eri soittimia omilla ennalta suunnitelluilla efekteillä lukemalla eri kanavilta eri soittimien omia viestejä yhtä aikaa. MIDI-viestit kulkevat MIDI-ulos-liitännän kautta ohjauslaitteelta vastaanottavalle laitteelle. Mahdollisia liitäntöjä on kolme: MIDI-sisään (MIDI In) viestien vastaanottamiseksi muilta laitteilta, MIDI-ulos (MIDI Out) viestien lähettämiseen ja MIDI-läpi (MIDI Thru), joka kopioi eteenpäin MIDI-sisään-liitännän kautta tulleet viestit (4, s. 29).

4.3 MIDI Jack -ohjelmointirajapinta

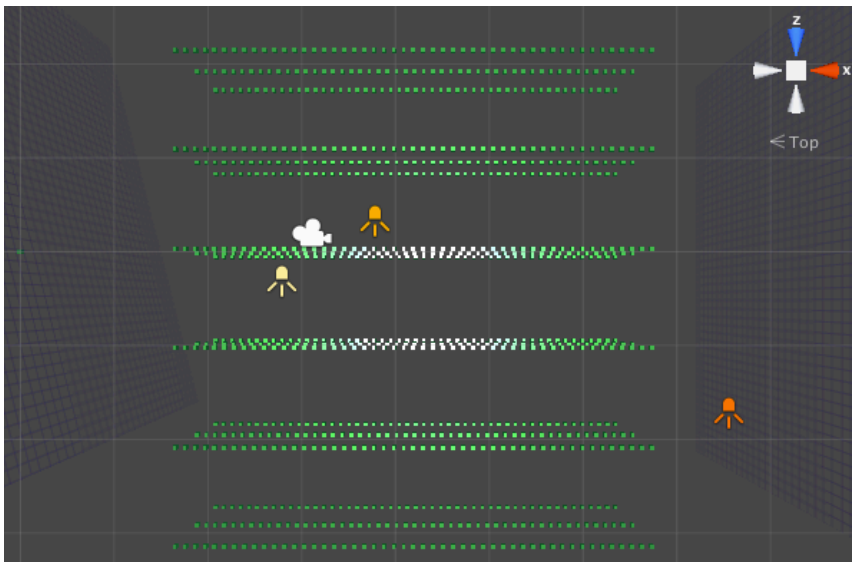
Sovelluksessa MIDI-viestien vastaanoton mahdollistaa MIDI Jack niminen ohjelmointirajapinta. MIDI Jack on Keijiro Takahashin Unityyn kehittämä avoimen lähdekoodin ohjelmointirajapinta, joka on ladattavissa GitHub-ohjelmakehitysverkkosivuston kautta. MIDI

Jack toimii Windows- ja OS X -käyttöjärjestelmille valmistettujen sovellusten kanssa. Sen avulla pystytään vastaanottamaan nuotti- ja ohjainviestejä kaikilta MIDI-kanavilta ja kaikilta koneeseen liitetyiltä MIDI-laitteilta. MIDI Jack on kokoelma C#-kielellä kirjoitettuja tiedostoja, jotka otetaan mukaan Unity-projektiin. MIDI Jackin toiminnot saa käyttöön luomalla Unityyn uuden C#-tiedoston ja lisäämällä siihen MIDI Master -luokan. Tämä tehdään lisäämällä C#-tiedoston alkuun koodi: using MidiJack;. (9.)

Liitteessä 1 on kuvattu ohjelmointirajapinnan kutsut, joiden avulla saadaan MIDI-tieto Unity-sovelluksen. Kutsu MidiMaster.GetKeyDown (kanava, nuottinumero) palauttaa tiedon tosi tai epätosi riippuen siitä, onko nuottinumeroa ja kanavaa vastaava ohjainlaitteen kosketin painettu alas. Kanava- ja nuottinumero-muuttujien tilalle sijoitetaan haluttuja arvoja vastaavat numerot. Kanavatietokohta on kaikissa MIDI Jack -kutsuissa vapaaehtoinen. Jos kohta jätetään tyhjäksi, se palauttaa kaikista kanavista saadun yhdistetyn keskiarvon. MidiMaster.GetKey (kanava, nuottinumero) palautta nopeusarvon väliltä 0–1. Jos kosketin ei ole painettuna, on arvo nolla. MidiMaster.GetKeyUp (kanava, nuottinumero) palauttaa tiedon ”tosi”, jos kosketinta ei paineta. Tehty sovellus käyttää edellä mainittuja komentoja. Niiden lisäksi MIDI Jack -lisäosalla voidaan saada tiedot modulaatiosäätimien liikautuksista. (9.)

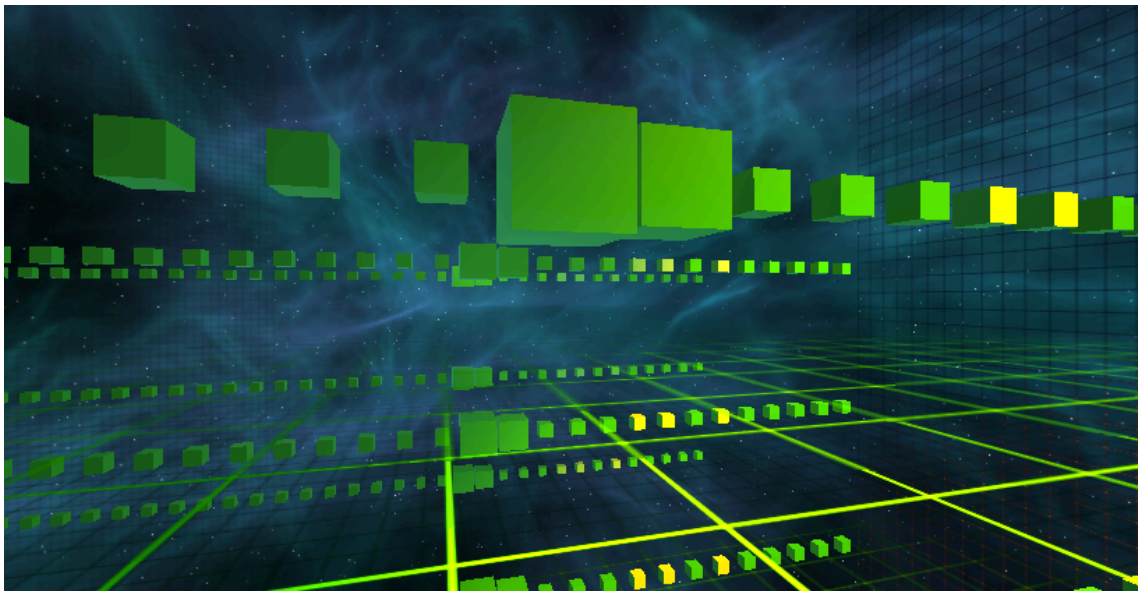
4.4 MIDI-nuottiviestin käyttö sovelluksessa

Sovelluksessa käytetään nuottiviestejä nuotti päällä ja nuotti sammutettu esimerkiksi sovelluksen Retro-nimisessä kohtauksessa. Tässä kohtauksessa kamera liikkuu kolmiulotteisessa tilassa ennalta ohjelmoitua reittiä pitkin eteenpäin välillä suuntaa vaihtaen kaartamalla oikealle tai vasemmalle. Yksi tilan elementeistä on 3 x 6 x 49 -kokoinen ruudukko kuutioita, jotka näkyvät kuvan 9 kuvakaappauksessa Unity-editorista.



Kuva 9. Retro-kohtaus Unity-editorissa ylhäältäpäin kuvattuna.

Kuutiorivejä on kuusi kolmessa kerroksessa, joissa jokaisessa rivissä on 49 kuutiota, jotka vastaavat ohjainkoskettimiston koskettimia 48–96. Kun ohjauskoskettimistossa painetaan yhtä kosketinta, sitä vastaavat kuutiot muuttavat kokoaan niin, että kuution koon muutos riippuu painalluksen nopeudesta. Kuva 10 on kuvakaappaus tilanteesta, jossa on painettu kaksi vierekkäistä kosketinta alas. Koskettimista vasemmanpuoleista on painettu nopeammin, joten vasen kuutio on oikeanpuoleista isompi.



Kuva 10. Tilanne, jossa kaksi vierekkäistä kosketinta on painettuna alas.

Tämä ohjelmiston osa on suunniteltu kosketinsoittimelle, jossa on 49 kosketinta, mutta se toimii muillakin ohjauslaitteilla. Jos kosketinsoittimessa on enemmän koskettimia,

kaikki painallukset eivät visualisoidu, ja jos käytössä on vähemmän oktaaveja sisältävä kosketinsoitin, visualisoinneissa jää näkyviin kuutioita, jotka eivät muutu ollenkaan esityksen aikana.

Esimerkkikoodi 1 on liitetty jokaiseen kuutio-objektiin. Se muuttaa kuution kokoa koskettimen painalluksen nopeuden mukaan.

```
***RetroKuutio.cs
using UnityEngine;
using MidiJack;

public class RetroKuutio : MonoBehaviour
{
public int nuottiNumero;

    void Update()
    {
    transform.localScale = Vector3.one * (0.4f +
    MidiMaster.GetKey(nuottiNumero));
    }
}
```

Esimerkkikoodi 1. Kuution kokoa muuttava koodi.

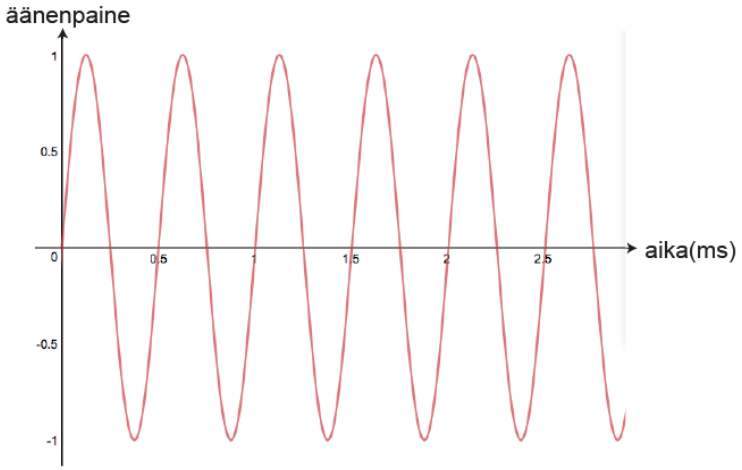
Kuutio on Unityyn luotu malliobjekti, joista jokaiselle on annettu nuottinumero väliltä 48–96 luontikoodin kautta sen luontivaiheessa nuottiNumero-muuttujaan. Update (päivitys) -komennon sisällä oleva koodi käydään läpi jokaisella ruudunpäivityskerralla. MidiMaster.GetKey-komento palauttaa senhetkisen nuottiNumero-muuttujaa vastaavan koskettimen painalluksen nopeuden. Funktio hakee MIDI-viesteistä nuotti päälle -viestin toisen arvotavun arvon. 0,4 yksikköä on kuution normaalikoko, johon lisätään saatu arvo väliltä 0–1. transform.localScale-komento kuuluu UnityEngine-luokkaan ja kykenee muokkaamaan kuution kokoa. Kuution koko vaihtelee välillä 0,4–1,4 yksikköä. Kun kosketinta ei paineta, nopeus on nolla ja kuution koko on 0,4 yksikköä. Kuution jokaisen sivun pituutta muutetaan samassa suhteessa, joten voidaan käyttää Vector3.one(koko)-komentoa, joka on lyhennelmä Vector3 (x-koko, y-koko, z-koko) -komennosta. (11.)

5 Ääni ja digitaalisen äänitiedon ominaisuudet ja mahdollisuudet Unityssä

5.1 Ääni ja ääniaallot

Sanalla ääni kuvataan kahta asiaa, joista ensimmäinen on kuuloaistimus korvassa ja toinen on väliaineen mekaaninen värähtely, joka saa aikaan kuuloaistimuksen. Fysikaalisilta ominaisuuksiltaan ääni on ilmassa olevaa värähtelyä eli aaltoliikettä. Käytännössä kaikki muukin kommunikaatio perustuu aaltoihin. Ääniaaltojen lisäksi ympärillämme on esimerkiksi valo- ja radioaaltoja. Näitä eri aaltoja yhdistää kyky kuljettaa informaatiota ja energiaa. Korvassa ilmanpaineen vaihtelujen seurauksena aaltoliikkeestä tulee hermoimpulsseja, joita aivomme tulkitsevat. Ääni on siis aaltoliikettä, jolle voidaan määrittää esimerkiksi taajuus, aallonpituus ja nopeus. (5, s. 3 ja 14.)

Kuvaajaa, joka kuvaa äänenpaineen muutosta ajan kuluessa kutsutaan äänen aaltomuodoksi. Yksinkertainen tapa kuvata äänen aaltomuotoa on piirtää kuvan 11 mukainen kuvaaja, jossa pystyakselilla on äänenpaine ja vaaka-akselilla aika. Vaakatasossa keskellä on nollapaineen viiva, joka vastaa hiljaisuutta. Kuvaajan viiva on lähellä alareunaa, kun ilmanpaine on matala, ja yläreunassa taas suuri. Jokainen piste kuvan 11 mukaisessa analogisessa signaalissa on sen taajuusarvo tietyllä ajanhetkellä. (5, s. 14–15.)



Kuva 11. Aaltomuoto, joka kuvaa perusäänestä eli ääntä, joka sisältää vain yhtä taajuutta

Taajuus tarkoittaa jonkin toistuvan ilmiön tapahtumien määrää tietyssä aikayksikössä. Sen voi laskea kuvan 11 mukaisesta vain yhtä taajuutta sisältävästä perusäänestä. Taajuuden laskemista varten on ensin määritettävä jokin aikaväli ja tutkittava sen jälkeen,

kuinka monta kertaa sama aallonharja toistuu tämän aikavälin aikana. Kun kertojen lukumäärä jaetaan aikavälin pituudella eli jaksoajalla (T), saadaan taajuus eli värähdysaika:

$$f = \frac{1}{T} \tag{1}$$

Kuvan 11 aallon taajuus lasketaan tarkastelemalla kaaviota. Koska käyrän aaltofunktio ei ole tiedossa, siitä nähdään, että kahden aallon huippukohdan ero on 0,5 ms (= 0,0005 s), joten taajuus on 2 000 Hz.

$$f = \frac{1}{0{,}0005\ s} = 2000\ Hz \tag{2}$$

Taajuuden tunnus on f (frequency), ja sen SI-järjestelmän mukainen yksikkö on hertsi (Hz). Yhden hertsin taajuudella tapahtuva tapahtuma toistuu kerran sekunnissa. Hertsi voidaan esittää muodossa

$$[f] = \frac{1}{s} = Hz, \tag{3}$$

jossa s on sekunti. (5, s. 23.)

Ihmisen kuuloalue sijoittuu välille 20–20 000 Hz, joka vastaa noin yhdeksää oktaavia. Harva ihminen pystyy havaitsemaan koko taajuusalueen, ja kyky havaita etenkin korkeita ääniä heikkenee vanhetessa. (5, s. 79–80.) Äänenpainetasolla (L_p) ilmaistaan äänen voimakkuutta mittayksiköllä desibeli [dB]. Hiljaisin ääni, minkä ihmisen korva voi havaita, vastaa äänenpainetasoltaan 2 * 10^{-5} N/m^2 1000 Hz:n taajuudella. (5, s. 104–106.)

Näytteenottotaajuus on tiheys, joka ilmoittaa jatkuvasta analogisesta signaalista otettavien näytteiden määrän sekunnissa. Sen yksikkö on hertsi. Digitaalisessa äänen käsittelyssä näytteenottotaajuus on usein 48 000 Hz. Myös pienempiä ja suurempia taajuuksia käytetään. Tämän vuoksi usein käytetään yksikköä kHz eli kilohertsi, joka tarkoittaa tuhatta hertsiä. 50 kHz:n näytteenottotaajuus tuottaa suuren määrän tietovirtaa tarkasteltavaksi lyhyessä ajassa. Tällä näytteenottotaajuudella saadaan 6 000 000 näytettä minuutissa. (12, s. 26.)

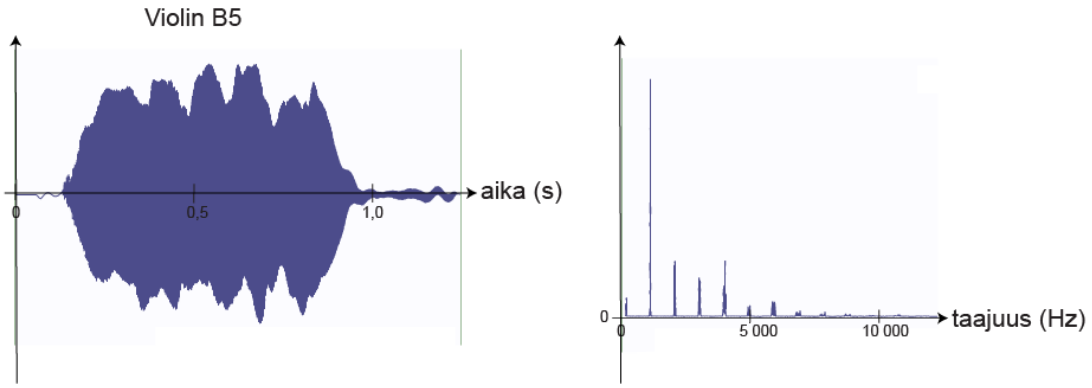
Kun näytteenottotaajuus on riittävän suuri, voidaan äänestä muodostaa uudelleen alkuperäisen kaltainen signaali (12, s. 26). Nyqvistin teoreeman mukaan tarvitaan ainakin R

määrä näytteitä sekunnissa, jos halutaan toistaa R/2 näytteitä sekunnissa sisältävä signaali R:n ollessa näytteenottotaajuus. 20 000 Hz:n taajuisen näytteen toisto vaatii 40 000 Hz:n näytteenottotaajuuden. Ylinäytteistettäessä näytteenottotaajuus on enemmän kuin signaalin, joka halutaan tallentaa, taajuus kerrottuna kahdella. Ylinäytteistäminen vähentää häiriöitä. Käytännön sovelluksista esimerkiksi CD-levyillä äänisovelluksissa näytteenottotaajuus on 44 100 Hz. (5, s. 483, 487 ja 517.)

Oskilloskooppi on laite, jolla voidaan saada aikaiseksi kuvaaja, joka kuvaa äänen aaltomuodon. Kun oskilloskoopilla tutkitaan ääniaaltoja, siihen liitetään mikrofoni, joka muuttaa ääniaallot jännitteeksi, niin että voimakkaammasta äänestä seuraa suurempi jännite. Oskilloskooppi luo kuvaajan, joka esittää mikrofonista tulevaa jännitettä ajan funktiona. (13, s. 128–129.)

5.2 Spektrianalyysi ja Fourier’n muunnos

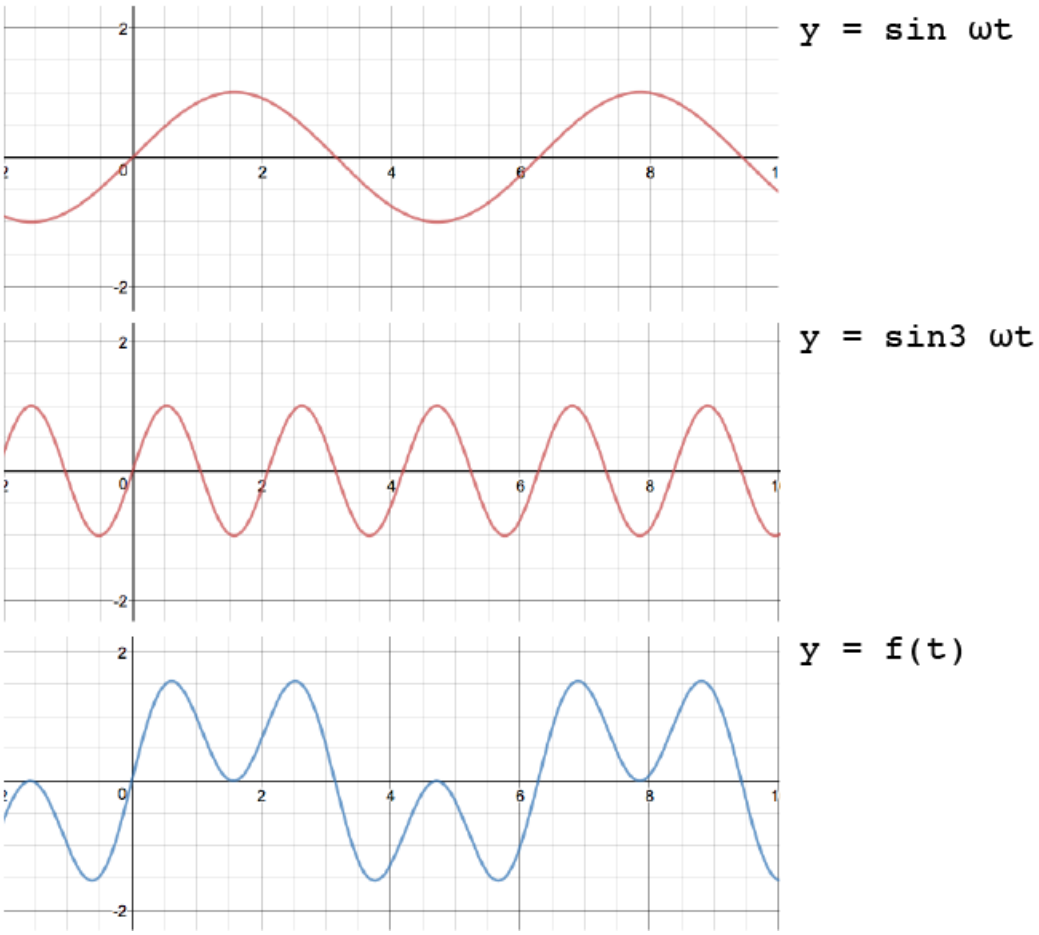
Ääniaallot sisältävät harvoin vain yhtä harmonista sinimuotoista aaltoa. Harmonisten osa-aaltojen määrittämistä jaksollisesta aallosta kutsutaan Fourier’n muunnokseksi. Joseph Fourier määritteli hänen mukaansa nimetyn Fourier’n muunnoksen seuraavasti: Mikä tahansa jaksollinen värähtely, miten tahansa monimutkainen voidaan rakentaa sarjasta yksinkertaisia värähtelyjä, joiden taajuudet ovat harmonioita perustaajuuksista, valitsemalla oikeat taajuudet ja vaiheet näistä harmonioista. Spektrianalyysi-sanaa käytetään joskus myös kuvaamaan Fourier’n muunnosta, joka on tehty äänelle. Kuvan 12 oikeanpuoleisen kuvion mukaista erittelyä, jossa on määriteltynä eri taajuuksien vahvuudet, kutsutaan taajuusspektriksi. (5, s. 136–137; 13, s. 116.)



Kuva 12. Vasemmalla kuvassa viulun yhden sävelen aaltomuoto ja oikealla taajuusspektri eli eri taajuuksien esiintymistiheys oikeanpuoleisesta ääninäytteestä.

Samasta soittimesta saatavaan taajuusspektriin vaikuttaa se, miten sitä soitetaan: kovaa, korkealta, matalalta vai keskialueelta. Myös äänen nauhoitustilanne vaikuttaa saatavaan spektriin. Moderneja spektrianalyysejä on kahta tyyppiä: digitaalista ja analogista. Digitaalisessa spektrianalyysissä otetaan ääniaallosta tasaisin väliajoin näytteitä, jotka syötetään tietokoneelle. Analogiset spektrianalysaattorit käyttävät suodattimia tai elektronisia piirejä erotellakseen ylä-äänet yksi toisensa jälkeen. (5, s. 138.)

Kuvassa 13 alimpana on aaltoliikkeen kuvaaja, joka on myös jaksollinen funktio. Kuvaaja voisi kuvata esimerkiksi soittimen ääntä, jossa toistuu selvästi jaksokuvio. Sen yläpuolella on sen osa-aaltofunktioiden kuvaajat.



Kuva 13. Summa-aalto ja sen osa-aallot eli harmoniset komponentit.

Ylempi osa-aalto on taajuudeltaan pienempi ja kuvasta voidaan huomata, miten toisen mutkikkaamman osa-aallon kuvio toistuu summa-aallossa yksinkertaisemman osa-aallon kuvion mukaan. Kun y_1 ja y_2 ovat osa-aaltojen amplitudit osa-aaltojen summafunktion eli alimman jaksollisen aallon kaava on

$$f(t) = \hat{y}_1 \sin \omega t + \hat{y}_2 \, sin3 \, \omega t. \tag{4}$$

Fourier'n muunnos perustuu Fourier'n sarjoihin. Jokainen jaksollinen funktio f(t) voidaan esittää sinien ja kosinien summana eli Fourier'n sarjana

$$f(x) = \sum_{n=0}^{\infty} (a_n \cos(\mathrm{n} * \omega \mathrm{t}) \; + b_n (\mathrm{n} * \omega \mathrm{t})) \tag{5}$$

Kertoimet a_n ja b_n saadaan integroimalla. Käytännön sovelluksissa Fourier'n muunnoksessa etsitään jaksollisesta aallosta amplitudikertoimet a_n ja b_n ja taajuudet n*f. Yleisimmin käytössä oleva algoritmi Fourier'n muunnokseen on nopea Fourier'n muunnos eli FFT (Fast Fourier Transform). (13, s. 116–117.)

5.3 Äänitiedon käyttö Unityssä

Unity sisältää itsessään äänitoiminnot, jotka mahdollistavat äänitiedon hakemisen sovellukseen. Unityn äänitoiminnot saa käyttöön lisäämällä kohtaukseen komponentit äänen kuuntelija (Audio Listener) ja äänilähde (Audio Source). Äänen kuuntelija vastaanottaa äänet 3D-tilassa ja soittaa ne sovelluksessa tai tulkitsee muuten halutulla tavalla. Äänilähteitä voi Unity-kohtauksessa olla useita, kun taas äänen kuuntelijoita voi olla vain yksi. Äänilähteisiin voi liittää lähteeksi äänitiedoston, tietokoneen mikrofonin tai muun tietokoneeseen liitetystä laitteesta tulevan äänen. Äänilähteen ja -kuuntelijan voi lisätä mihin tahansa objektiin kohtauksessa. Unityssä on mahdollista käyttää 3D-äänitilaa, jossa objektien etäisyys toisistaan vaikuttaa äänen voimakkuuteen. Kun äänilähde ja kuuntelija liitetään samaan objektiin, sen liikkuminen tilassa ei vaikuta äänen voimakkuuteen. Microphone-luokalla viitataan tietokoneen omaan mikrofoniin. MacBook pro -koneessa sisääntuloäänilähteen voi asettaa sen ääniasetuksista. Unityllä tehty sovellus löytää tämän äänilähteen ja pystyy käyttämään sitä. (14.)

Microphone-luokkaan voi viitata vain koodilla, koska sitä varten ei ole kohtaukseen lisättävää komponenttia, kuten esimerkiksi äänilähteellä. Komento Microphone.Start(string deviceName, bool loop, int lengthSec, int frequency); asettaa halutun mikrofonin äänilähteeksi. deviceName (laitteen nimi) -muuttujan voi jättää tyhjäksi, jolloin sovellus käyttää tietokoneen ääniasetusten mukaista oletusmikrofonia. Boolean-tyyppinen muuttuja loop (silmukka) määrittää, onko äänitiedon sisääntulo jatkuvaa vai onko se kerran soitettava leike. Kesto lengthSec (kesto sekunneissa) -muuttujan kohdalle valitaan äänileikkeen pituus sekunteina ja arvo on kokonaislukuna, ja kun soitettavan äänen sisääntulo on jatkuvaa, voi arvoksi asettaa esimerkiksi 1. Frequency (taajuus) -kohtaan sijoitetaan äänileikkeen näytteenottotaajuus hertseinä. (15.)

Tuleva äänitieto voidaan ottaa käyttöön suodattamattomana. Esimerkkikoodissa 2 GetOutputData-komennolla saadaan naytteet-nimiseen taulukkoon sillä hetkellä soivan äänilähteen ulostulotiedot. Taulukon pituuden pitää olla kahden potenssi välillä 64–8192. Kanavavaihtoehdoista valitaan channel(kanava)-muuttujaan haluttu. Vaihtoehtoja on

kolme: vasen (arvo: 0), oikea (1) ja molemmat (2). Jokainen naytteet-taulukkoon tuleva arvo kuvaa pistettä ajassa ja ääniaallossa. Taulukon pituus kuvaa kerrallaan otettavan näytteen ajallista pituutta. (16.)

```
AudioSource aanilahde = GetComponent<AudioSource>();
aanilahde.clip = Microphone.Start("Built-in Microphone", true,
1, 44100);
aanilahde.Play();
GetOutputData(float[] naytteet, int channel);
GetSpectrumData (float[] naytteet2, int channel, FFTWindow
ikkuna);
```

Esimerkkikoodi 2. Mikrofonin käyttöönotto äänilähteeksi. Tämä tulisi tehdä ohjelmatiedoston alkuvaiheessa eli esimerkiksi aloitus(Start)-funktion sisällä, joka suoritetaan ensimmäisenä.

Spektri on ote siitä, mitä tapahtuu eri taajuusalueilla tietyllä hetkellä. Esimerkkikoodissa 2 GetSpectrumData-komento hakee sillä hetkellä soivan ääninäytteen spektritiedot taulukkoon naytteet2. Kanava-muuttujaan valitaan kanavavaihtoehdoista vasen, oikea tai molemmat. Ikkuna-nimisen muuttujan paikalle valitaan haluttu spektrianalyysin apodisointifunktiotyyppi. Nopean Fourier'n muunnosfunktioita on valittavana kuusi erilaista: rectangular, triangle, Hamming, Hanning, Blackman ja BlackmanHarris (17). Taulukossa 3 on funktioiden kaavat, joista valittu suoritetaan jokaiselle naytteet2-taulukon arvolle.

Taulukko 3. Unityssä käytettävissä olevat apodisointifunktiot (17).

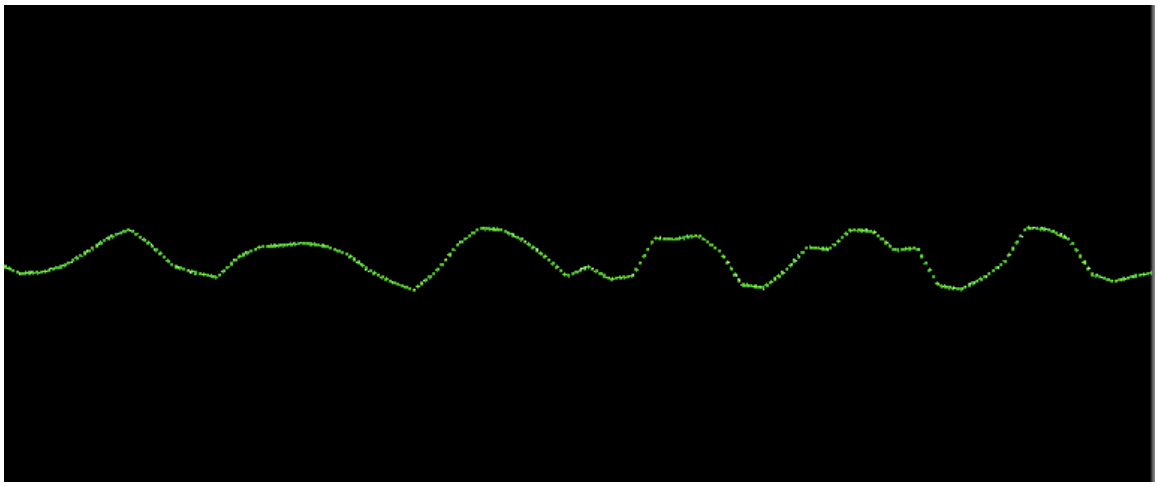
Windowing type	**Apodisointifunktion kaava**
Rectangular	W[n] = 1.0.
Triangle	W[n] = TRI(2n/N).
Hamming	W[n] = 0.54 - (0.46 * COS(n/N)).
Hanning	W[n] = 0.5 * (1.0 - COS(n/N)).
Blackman	W[n] = 0.42 - (0.5 * COS(n/N)) + (0.08 * COS(2.0 * n/N)).
BlackmanHarris	W[n] = 0.35875 - (0.48829 * COS(1.0 * n/N)) + (0.14128 * COS(2.0 * n/N)) - (0.01168 * COS(3.0 * n/N)).

Monimutkaisempi apodisointifunktio parantaa spektrin laatua vähentämällä vuotoa taajuusalueiden välillä, mutta hidastaa laskutoimituksia (17; 18).

6 Äänitiedon visualisointi sovelluksessa

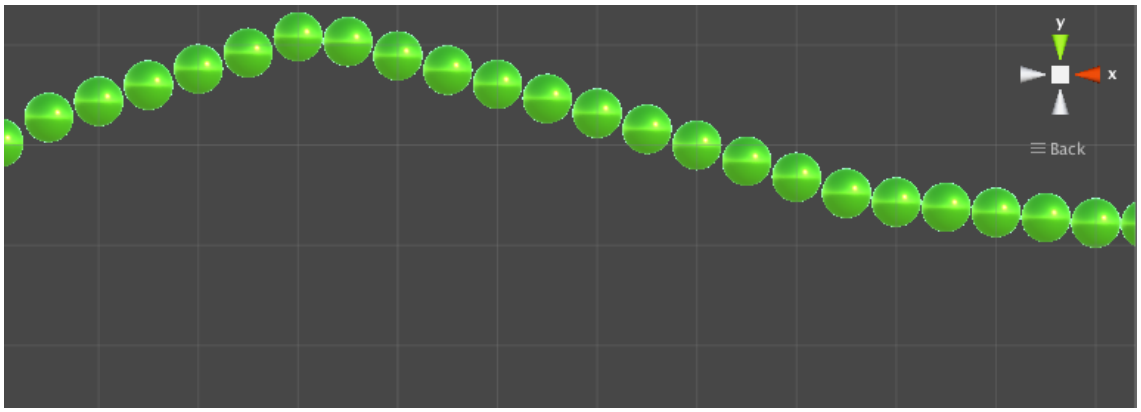
6.1 Äänen aaltomuodon visualisointi

Sovelluksessa on oskilloskooppi-tyyppinen osa, joka käyttää musiikin äänitietoa. Kuvassa 14 on kuvakaappaus visualisoidusta äänen aaltomuodosta. Aaltokuvio elää musiikin mukana ja päivittyy ruudunpäivitystiheyden mukaan.



Kuva 14. Ääniaallon visualisointi sovelluksessa.

Viivalta näyttävä kuvio on jono palloja, joita on 512. Kuvassa 15 on Unity-editori-ikkuna, jossa näkyy, että aalto muodostuu palloista, joista jokainen on kopio samasta malliobjektista.



Kuva 15. Ääniaallon muodostavat pallot Unity-editorin kohtaus-ikkunassa ohjelman ollessa käynnissä.

Sovellukseen tulevasta äänitiedosta otetaan ruudunpäivityskerralla 512 näytettä niiden tulojärjestyksessä. Oikeassa reunassa on ensimmäinen näyte ja viimeinen vasemmassa. Jokainen pallo kuvaa hetkeä ajassa, ja pallon korkeus määräytyy mikrofonilla saadun aallonkorkeuden mukaan. Esimerkkikoodissa 3 GetOutputData-komento antaa taulukkoon numerosarjan, josta haetaan toisessa taulukossa oleville palloille uudet korkeus eli y-arvot.

```
public GameObject palloPF;
GameObject[]palloTaulukko;

void Start(){
kuutioTaulukko = new GameObject[512];
     for(int i =0; i <512; i++)
     {
     Vector3 paikka = new Vector3(i/20f, 0, 0);
     GameObject uusiPallo = (GameObject)Instantiate(palloPF,
     paikka, Quaternion.identity);
     palloTaulukko[i] = uusiPallo;
     }
}

void Update()
{
aaniLahde.GetOutputData (naytteet, 0);

     if (palloTaulukko!= null)
     {
     for (int I = 0; I < 512; i++)
     {
     palloTaulukko[i].transform.position = transform.position =
     new Vector3(palloTaulukko[i].transform.position.x,
     naytteet[i], 0);
     }
     }
}
```

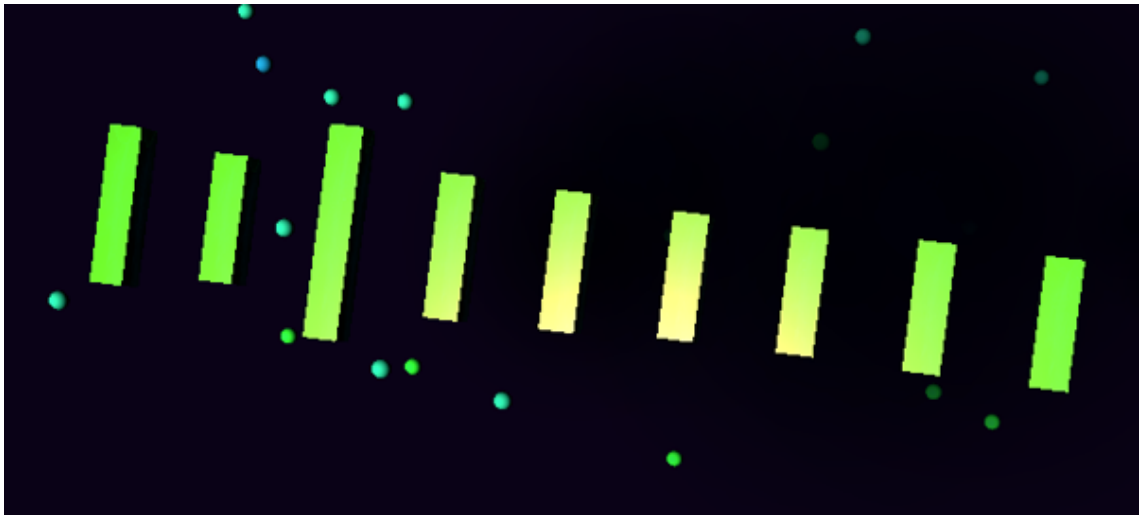
Esimerkkikoodi 3. Objektiin lisättävä koodi, jolla voidaan vaihdella objektin sijaintia y-akselilla äänitiedon mukaan.

Koodissa varmistetaan ensin, että taulukon objektit ovat olemassa. ”Naytteet” on määritelty taulukoksi, jonka pituus on 512 ja aaniLahde-muuttujaan äänilähteeksi on määrätty haluttu mikrofoni. Esimerkkikoodi 3 on osa muitakin yleisiä kohtauksen tapahtumia sisältävää koodia. GameObject (peliobjekti) -muuttujaan alussa liitetty palloPF-niminen objekti on malliobjekti, josta luodaan Start (aloitus) -funktion for-silmukassa 512 kopiota, jotka muodostavat visuaalisessa esityksessä ääniaallon. Nämä malliobjektin kopiot sijoitetaan palloTaulukko-taulukkoon, jotta niiden sijaintia y-akselilla voidaan muokata esi-

tyksen edetessä. Tämä tapahtuu Update (päivitys) -funktiossa, jossa palloTaulukko-taulukon objektit käydään läpi yksitellen ja jokainen saa uuden sijainnin y-akselilla äänitiedosta saatavan tiedon mukaan. Näin esitykseen saadaan liikkuva oikeaa äänitietoa visualisoiva käyrä, jonka äänen teoriaa tunteva katsoja tunnistaa helposti paljonkin informaatiota sisältäväksi digitoidun äänen aaltomuodoksi.

6.2 Spektritiedon käyttö sovelluksen visualisoinneissa

Spektrianalyysillä saatavaa tietoa on käytössä sovelluksessa monessakin osassa. Tässä esimerkkinä yhdeksän pylvään ryhmä, joista jokainen kuvaa eri taajuusalueen vahvuuden muutoksia. Kuvassa 16 on kuvakaappaus pylväistä, joiden korkeus muuttuu spektrianalyysillä saatavan tiedon mukaan.



Kuva 16. Spektrianalyysiä visualisoivat pylväät. Vasemmassa reunassa on pylväs, jonka indeksinumero on nolla.

Korkeampia taajuuksia esiintyy musiikista saatavassa äänessä vähemmän, joten taajuudet on jaettu yhdeksään osaan niin, että lopputuloksesta saadaan myös visuaalisesti mielenkiintoinen. Ensimmäinen tolppa sisältää matalimmasta päästä noin 43 hertsin edestä näytteitä ja viimeinen sisältää puolet tulevista näytteistä eli korkeimmat 11 050 hertsiä. Tulevan äänen näytteenottotaajuus on 44 100 hertsiä, joten käytössä on 22 050 hertsin edestä näytteitä. Saatava äänitieto sijoitetaan taulukkoon, jonka pituudeksi on valittu 512. Jokainen taulukon indeksi tulee näin sisältämään noin 43 hertsin edestä tietoa, koska

$$\frac{22\,050\ Hz}{512} = 43{,}0664\ Hz.$$

Esimerkkikoodi 4 on osa liitteen 2 SpektriInformaatio.cs-tiedoston koodista. Liitteen 2 koodin mukaisesti äänilähteeksi on asetettu tietokoneen mikrofoni aaniLahde-muuttujaan. HaeSpektriTieto-funktion alussa spektritieto haetaan 512 näytettä pitkään naytteet-taulukkoon. Saatavalle tiedolle tehdään nopea Fourier'n muunnos Blackman-kaavalla. Tämä koodi käydään läpi jokaisella ruudunpäivityskerralla.

```
***SpektriInformaatio.cs

float[] naytteet = new float[512];
float[] spektriRyhma = new float[9];

void HaeSpektriTieto()
{
aaniLahde.GetSpectrumData(naytteet,0, FFTWindow.Blackman);

int nIndeksi = 0;
int nKappaletta = 1;

     for (int i = 0; i <9; i++)
     {
     float keskiarvo = 0;

           for (int j = 0; j < nKappaletta; j++)
           {
           keskiarvo += naytteet[nIndeksi];
           nIndeksi++;
           }

           if (i == 8)
           {
           keskiarvo += naytteet[nIndeksi];
           nKappaletta ++;
           }

           spektriRyhma[i] = keskiarvo / nKappaletta;
           nKappaletta *= 2;
     }
}
```

Esimerkkikoodi 4. Koodi spektrianalyysi tiedon käyttämisestä ja taajuusnäytteistä.

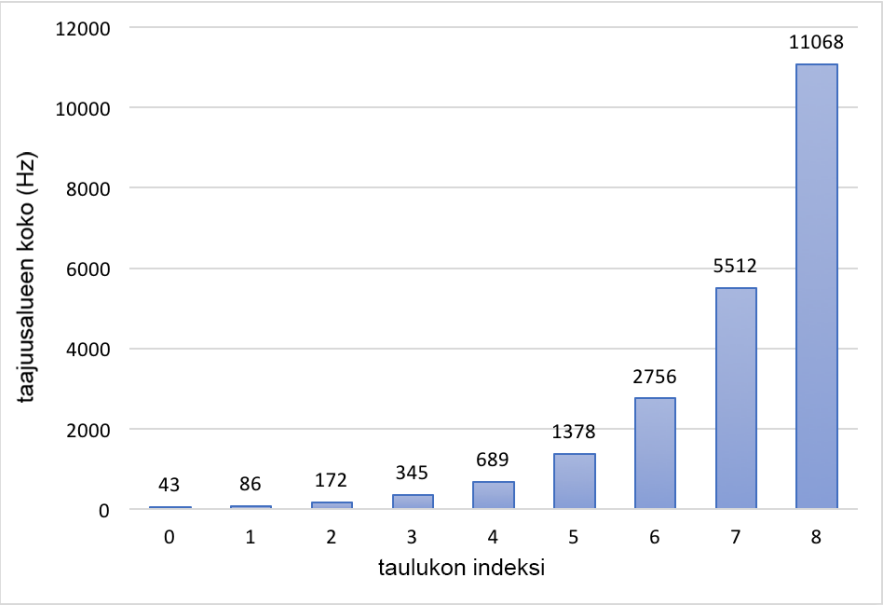
Esimerkkikoodin 4 ulommaisessa for-silmukassa käydään läpi luvut 0–8 eli uuden spektriRyhma-taulukon kaikki kohdat, joista jokainen vastaa yhtä visualisoitua spektripylvästä kuvassa 16. Taulukossa 4 on kuvattu kunkin yhdeksän spektriRyhma-taulukon kohdan laskentakaavat ja taajuussisällöt. Esimerkkikoodissa 4 sisemmällä for-silmukalla haetaan naytteet-taulukosta taulukon 4 toisen sarakkeen mukainen kappalemäärä näytteitä,

joiden sijainnit naytteet-taulukossa vastaavat taulukon 4 kolmatta saraketta ja esimerkkikoodin 4 nIndeksi-muuttujaa. Jokaisella ulomman for-silmukan kierroksella saaduista naytteet-taulukon arvoista lasketaan keskiarvo, joka sijoitetaan spektriRyhma-taulukkoon.

Taulukko 4. Spektripylväillä visualisoitavien taajuusalueiden laajuudet ja niitä koodissa vastaavien taulukon indeksikohtien laskentakaavat.

Spektri-Ryhma-taulukon indeksi	Näytteiden kappalemäärä	Indeksit naytteet-taulukossa	Spektriryhmän sisältämät taajuudet (arvot ovat noin-arvoja)	Taajuusalueen koko (näytteiden kappalemäärä * 43,0664)
0	1 = 1	0	0–43,07 Hz	43 Hz
1	1 * 2 = 2	1–2	43,08–129,20 Hz	86 Hz
2	2 * 2 = 4	3–6	129,21–301,46 Hz	172 Hz
3	4 * 2 = 8	7–14	301,47–646,00 Hz	345 Hz
4	8 * 2 = 16	15–30	646,01–1 335,06 Hz	689 Hz
5	16 * 2 = 32	31–62	1 335,07–2 713,18 Hz	1 378 Hz
6	32 * 2 = 64	63–126	2 713,19–5 469,43 Hz	2 756 Hz
7	64 * 2 = 128	127–254	5 469,44–10 981,93 Hz	5 512 Hz
8	128 * 2 + 1 = 257	255–511	10 981,94–22 050,00 Hz	11 068 Hz
Yhteensä	512			22 050 Hz

Kaikki spektriRyhma-taulukon kohdat sisältävät keskiarvon eri kappalemäärästä näytteitä. Otettavien näytteiden kappalemäärä kaksinkertaistetaan jokaisella ulomman for-silmukan kierroksella kertomalla nKappaletta-muuttuja kahdella silmukan lopussa taulukon 4 toisen sarakkeen laskukaavojen mukaisesti. Viimeiseen spektriryhmään lisätään naytteet-taulukon viimeinen näyte, joka ei laskukaavalla muuten tule mukaan. Spektripylväiden sisältämien taajuusalueiden laajuuksien kasvaminen logaritmisesti on havainnollistettu kuvan 17 kaaviossa.



Kuva 17. Spektripylväiden sisältämien taajuusalueiden laajuudet.

Koska saatavien arvojen vaihtelevuus on suuri riippuen esiintymispaikan akustiikasta ja äänen voimakkuuden vaihteluista, on hyvä saada spektripylväiden arvot skaalattua välille 0–1, jotta visualisoinnit pysyvät yhtenäisinä käyttötilanteesta riippumatta. Esimerkkikoodissa 5 spektriRyhma-taulukon arvot skaalataan käyttöön sopiviksi. Tämä suoritetaan samassa spektriInformaatio.cs-kooditiedostossa kuin esimerkkikoodi 4.

```
***SpektriInformaatio.cs

float[] spektriRyhma = new float[9];
public static float[] skaalattuSR = new float[9];
float[] spektriRyhmaKorkein = new float[9];

void SpektriRyhmienSkaalaus()
{
     for(int i=0; i < 9; i++)
     {
           if(spektriRyhma[i] > spektriRyhmaKorkein [i])
           {
           spektriRyhmaKorkein[i] = spektriRyhma[i];
           }
     skaalattuSR[i] = (spektriRyhma[i] / spektriRyhmaKorkein[i]);
     }
}
```

Esimerkkikoodi 5. Koodi spektriryhmien arvojen skaalaamisesta välille 0–1.

Esimerkkikoodissa 5 lasketaan uuteen taulukkoon arvot, joita spektripylväät tulevat käyttämään. Ensin on luotu spektriRyhma-taulukon rinnalle skaalattuSR-taulukko ja spektriRyhmaKorkein-taulukko, joka säilyttää muistissa spektriRyhma-taulukon kunkin indeksin korkeimmat arvokohdat. Jokaisella koodin suorituskerralla uusi spektriRyhma-taulukon arvo jaetaan tällä maksimiarvolla, jolloin skaalattuSR-taulukkoon sijoitettava uusi arvo on aina suurimmillaan 1. Arvot pysyvät välillä 0–1, ja niitä voidaan käyttää tämän kooditiedoston ulkopuolella, koska skaalattuSR-taulukko on asetettu julkiseksi.

Jokainen kuvan 16 palikoista on Unityyn tehty malliobjekti, johon on liitetty esimerkkikoodin 6 sisältävä C#-tiedosto.

```
***pylvasKoko.CS

using UnityEngine;
using System.Collections;

public class PylvasKoko : MonoBehaviour
{
public int spektriIndeksi;
public float alkuKoko = 1;

    void Update ()
    {
    float uusiKoko =
spektriInformaatio.skaalattuSR[spektriIndeksi] + alkuKoko;
    transform.localScale = new Vector3(0.3f, uusiKoko, 0.3f);
    }
}
```

Esimerkkikoodi 6. Pylväiden korkeuden muuttaminen spektritiedon avulla.

Esimerkkikoodissa 6 jokaiselle spektripylväälle on luontivaiheessa määritetty oma spektriryhmän indeksi numeroarvoltaan 0–8, ja niille löytyy vastaava indeksi taulukon 4 ensimmäisestä sarakkeesta. Jokaisella koodin suorituskerralla koodi tarkistaa SpektriInformaatio-tiedoston skaalattuSR-taulukosta uuden arvon spektripylvään y-korkeudeksi. Tähän arvoon lisätään pylvään alkukoko, joka on 1, joten jokaisen pylvään y-pituusarvo vaihtelee välillä 1–2. Koot x- ja z-suunnissa pysyvät ennallaan. SpektriInformaatio.cs-tiedostosta voidaan hakea muuttujiin arvoja, koska se on liitetty objektiin, joka on samassa kohtauksessa pylväs-tyyppisten malliobjektien kanssa.

Pylväiden visualisoinnissa päätettiin taajuusspektrin tietoja jakaa logaritmisesti pylväille visuaalisen näyttävyyden vuoksi, joten lopullinen visualisointi ei vastaa täysin sitä, mitä katsoja voisi kuvitella sen vastaavan. Palikat kuitenkin elävät näyttävästi musiikin tahtiin,

ja tässä käytetty koodi on tärkeä osa sovellusta, jossa sitä on käytetty monen muunkin osan elävöittämiseen. Toisena esimerkkinä spektrianalyysistä on sovelluksessa käytetty paljon yksittäisiä taustaelementtejä, jotka ottavat spektrianalyysistä vain yhden ryhmän spektrianalyysin tuloksen ja muuttavat kokoaan tai väriään sen tahdissa. Mielenkiintoisia visualisointeja saadaan myös, kun samaan objektiin liitetään spektritietoa useasta spektriryhmästä, siten että kappaleen x-leveys muuttuu ensimmäisen spektriryhmän tietojen mukaan ja y-korkeus muuttuu toisen spektriryhmän tietojen mukaan. Tämä osa ohjelmasta on siis käytettävissä myös osittain, ja visualisoinnit, joissa sitä käytetään ovat todella tärkeässä osassa, koska niiden yhteys ääneen on katsojan helppo havaita.

7 Yhteenveto

Insinöörityössä suunniteltiin, valmistettiin ja otettiin käyttöön sovellus, joka tuottaa reaaliaikaisesti musiikkiin reagoivan visuaalisen esityksen musiikkiesityksen taustalle live-esitystilanteessa. Valmistunut sovellus reagoi reaaliajassa MIDI- ja äänitietoon luoden lähinnä abstrakteja, mutta myös infografiikkaa lähellä olevia visualisointeja. Unity soveltui hyvin tämän sovelluksen kehitykseen, koska se sisältää itsessään äänitiedon vastaanottomahdollisuuden, ja käyttöön otetun MIDI Jack -ohjelmointirajapinnan avulla sovellus pystyy vastaanottamaan myös MIDI-tietoa.

Sovellusta on nyt käytetty kaksi kertaa live-esitystilanteessa, joissa se toimi pääosin halutulla tavalla. Sovellusta ei suunniteltu tiettyyn tilaan, joten sen jatkokehityksessä on otettava huomioon, että esiintymispaikat ovat aina erilaisia: tilan koko, akustiikka, kaiuttimista tulevan äänen voimakkuus, tilan valaistus sekä esiintymislavan taustakangas ja näytöt. Riippuen tietokoneen sijainnista esiintymistilassa yleisöstä tulevat äänet saattavat myös vaikuttaa sovelluksen vastaanottamiin äänitietoihin. Live-esitystilanteissa sovellus otti äänitiedon tietokoneen mikrofonin kautta. Mahdollinen parempi ratkaisu on ottaa äänitieto suoraan miksauspöydästä tietokoneeseen, jolloin miksauspöydästä voidaan asettaa tietokoneeseen tulevan äänen voimakkuus sopivaksi eivätkä ympäriltä tulevat häiriöäänet vaikuta visualisointeihin.

Sovelluksen tuottamat visualisoinnit on mahdollista heijastaa projektorilla taustakankaalle, tai vaihtoehtoisesti esitys on mahdollista esittää käyttäen yhtä tai useampaa näyttöä. Esiintymispaikoilta saa yleensä siellä esiintyjien käytettävissä olevien laitteiden luettelon, johon etukäteen perehtyminen on tärkeää esityksen onnistumisen kannalta. Kun esitys heijastetaan projektorilla taustakankaalle tai seinälle, taustamateriaalin väri ja ympäristön valaistus vaikuttavat esityksen väreihin ja näyttävyyteen. Esiintymistilat määrittelevät joka tapauksessa mahdollisuudet, joissa esityksestä pitäisi saada toimiva ja hyvältä näyttävä kokonaisuus. Riippumattomuutta esiintymistilasta voidaan hallita lisäämällä sovellukseen osa, jolla sen värien kirkkautta ja kontrastia voi vaihtaa esityksen aikana.

Äänen fysikaalisten ominaisuuksien tutkiminen tuotti paljon hyödyllistä tietoa äänen aaltomuotorakenteesta, taajuuksista ja spektritiedosta, minkä pohjalta visualisointeja rakennettiin. Visuaalisen esityksen informatiivisuus ei ole sovelluksen kannalta tärkeää, vaikka toisaalta on tärkeää, että tämänkaltaisissa esityksissä yleisön jäsen havaitsee, että mu-

siikilla ja kuvalla on selvä yhteys toisiinsa. Ääniaaltojen ja taajuusspektrien visualisointien havaittiin olevan melko lähellä yleensä tilastografiikassa käytettäviä kuvitustyyppejä ja infografiikkaa. MIDI-tiedon visualisointiin valittiin tietoisesti varsin yksinkertainen malli, joka toimii hyvin kappaleissa, joissa muusikko soittaa paljon kosketinsoittimia. MIDI-tieto visualisoitiin yksinkertaisesti asettamalla jonoon kuutioita, joista jokainen vastaa yhtä kosketinta ja muuttaa koskettiminen painalluksen mukaan kokoaan.

MIDI Jack -ohjelmointirajapinta mahdollistaa nuottiviestien käytön, joista sovellus käyttää vain ohjauskoskettimiston koskettimen painallusta alas ja painalluksen nopeutta. Näillä viesteillä visualisoidaan esimerkiksi kosketinsoittimen koskettimia, niin että kosketinta vastaava kuutio muuttaa kokoaan visualisoinneissa sen mukaan, onko se painettu alas. MIDI-tiedon visualisointiin käytetty ohjelmointirajapinta mahdollistaisi käytettyjen viestien lisäksi ohjainkoskettimiston modulaatiosäätimien käyttöönoton. Ne voisi olla mahdollista ohjelmoida tuottamaan visuaalinen efekti ilman, että se vaikuttaa soittimesta tulevaan ääneen. Tämä mahdollistaisi sen, että muusikko voisi kontrolloida visuaalista esitystä monipuolisemmin, mutta se vaatisi muusikolta halua ohjailla soittamisen lisäksi visuaalista esitystä. Insinöörityöprojektissa nämä esityksen osa-alueet haluttiin pitää erillään, niin että visuaalisia esityksiä kontrolloi toinen henkilö esityksen aikana. Mahdollista olisi myös ottaa MIDI-tietoa useasta eri soittimesta samaan aikaan ja ohjelmoida eri soittimille omat efektit sen mukaan, miltä kanavalta tieto tulee.

Visuaalisia esityksiä on sovelluksessa kuusi, ja niistä valitaan aina yksi soitettavaan kappaleeseen sopiva. Vaihtoehtoisesti esitys olisi myös voitu koostaa niin, että valmiista lyhyemmistä efekteistä valitaan pikanäppäimillä käytettävät, joita voi tarpeen mukaan sammuttaa ja käynnistää esityksen edetessä. Tällöin visuaalista esitystä live-esityksen aikana käyttävä henkilö saisi enemmän kontrollointimahdollisuuksia esitykseen ja pystyisi tuottamaan paljon monipuolisemmin erilaisia visualisointeja.

Ensimmäisen live-esitystilanteen kokemusten pohjalta on mahdollista parantaa sovellusta tulevaisuuden käyttöä varten. Sovellusta on mahdollista myös muokata muiden muusikoiden esityksiin käytettäväksi. Suurin osa sen sisältämästä kuvastosta on abstrakteja kuvioita, jotka soveltuvat myös muihin musiikkiesityksiin. Sovelluksen esityksestä tuli persoonallinen. Samaan lopputulokseen olisi ollut vaikeampi päästä käyttäen valmisohjelmia.

Lähteet

1 MusicBeam, readme-tiedosto. 2016. Verkkodokumentti. GitHub, Inc. <https://github.com/codingjoe/MusicBeam/blob/master/README.md>. 7.1.2016. Luettu 11.1.2017.

2 Bazik Blog: Hello world! 2014. Verkkodokumentti. Bazik. <http://bazik-vj.com/hello-world/>. 27.5.2014. Luettu 11.1.2017.

3 Resolume Avenue & Arena Manual. Verkkodokumentti. Resolume B.V. <https://resolume.com/manual/en/r5/start>. 2017. Luettu 12.1.2017.

4 Hirvi, Jussi & Tuominen, Antti Juhani. 1995. Uusi MIDI-kirja. Helsinki: Painatuskeskus.

5 Rossing, Thomas D., Moore, Richard & Wheeler, Paul. 2002. The science of sound. San Francisco: Addison Wesley.

6 Koponen, Juuso, Hildén, Jonatan & Vapaasalo, Tapio. 2016. Tieto näkyväksi. Informaatiomuotoilun perusteet. Helsinki: Aalto ARTS Books.

7 Menard, Michelle & Wagstaff, Bryan. 2015. Game Development with Unity. Second Edition. Boston: Cengage Learning.

8 Unity scripting languages and you. 2014. Verkkodokumentti. Unity Technologies. <https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/>. 3.9.2014. Luettu 12.2.2017

9 Takahashi, Keijiro. 2016. MIDI Jack. Verkkodokumentti. <https://github.com/keijiro/MidiJack>. 6.11.2016. Luettu 11.12.2016.

10 Oxygen 49, Features & Support. Verkkodokumentti. M-Audio. <http://www.m-audio.com/products/view/oxygen>. 2017. Luettu 14.1.2017.

11 Unity Documentation 5.5-001: Vector3-one. 2017. Verkkodokumentti. Unity Technologies. <https://docs.unity3d.com/ScriptReference/Vector3-one.html>. 8.3.2017. Luettu 8.3.2017.

12 Roads, Curtis et al. 1995. The Computer Music Tutorial. Cambridge, Massachusetts: The MIT Press.

13 Peltonen, Hannu, Perkkiö, Juha & Vierinen, Kari. 2004. Insinöörin (AMK) Fysiikka, osa 2. Lahden Teho-Opetus.

14 Unity User Manual 5.5: AudioOverview. 2017. Verkkodokumentti. Unity Technologies. <https://docs.unity3d.com/Manual/AudioOverview.html>.8.3.2017. Luettu 9.3.2017.

15 Unity Documentation 5.5-001: Microphone.Start. 2017. Verkkodokumentti. Unity Technologies. <https://docs.unity3d.com/ScriptReference/Microphone.Start.html>. 8.3.2017. Luettu 9.3.2017.

16 Unity Documentation 5.5-001: AudioListener.GetOutputData. 2017. Verkkodokumentti. Unity Technologies. <https://docs.unity3d.com/ScriptReference/AudioListener.GetOutputData.html>. 8.3.2017. Luettu 9.3.2017.

17 Unity Documentation 5.5-001: FFTWindow. 2017. Verkkodokumentti. Unity Technologies. <https://docs.unity3d.com/ScriptReference/FFTWindow.html>. 8.3.2017. Luettu 9.3.2017.

18 Unity Documentation 5.5-001: AudioSource.GetSpectrumData. 2017. Verkkodokumentti. Unity Technologies. <https://docs.unity3d.com/ScriptReference/AudioSource.GetSpectrumData.html>. 8.3.2017. Luettu 8.3.2017.

MIDI Jack -ohjelmointirajapinnan komennot

Seuraavat MIDI Jack -ohjelmointirajapinnan komennot ovat lainattu Keijiro Takahashin kirjoittamasta MIDI Jack readme -tiedostosta. (2.)

The basic functions of MIDI Jack are provided in the MidiMaster class.

The channel arguments in the following functions can be omitted. In that case, it returns the value in the All-Channel slot, which stores mixed status of all active channels.

- MidiMaster.GetKey (channel, noteNumber)

Returns the velocity value while the key is pressed, or zero while the key is released. The value ranges from 0.0 (note-off) to 1.0 (maximum velocity).

- MidiMaster.GetKeyDown (channel, noteNumber)

Returns true during the frame the user starts pressing down the key.

- MidiMaster.GetKeyUp (channel, noteNumber)

Returns true during the frame the user releases the key.

- MidiMaster.GetKnob (channel, knobNumber, defaultValue)

Returns the controller value (CC). The value ranges from 0.0 to 1.0.

- MidiMaster.GetKnobNumbers (channel)

Returns the list of active controllers.

There are also delegates for each type of MIDI event.

- MidiMaster.noteOnDelegate (channel, noteNumber, velocity)
- MidiMaster.noteOffDelegate (channel, noteNumber)
- MidiMaster.knobDelegate (channel, knobNumber, konbValue)

SpektriInformaatio.cs-tiedosto

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(AudioSource))]

public class SpektriInformaatio : MonoBehaviour {

AudioSource aaniLahde;
float[] naytteet = new float[512];
float[] spektriRyhma = new float[9];
public static float[] skaalattuSR = new float[9];
float[] spektriRyhmaKorkein = new float[9];

void Start ()
{
    aaniLahde = GetComponent<AudioSource> ();
    aaniLahde.clip = Microphone.Start(null, true, 1, 44100);
    aaniLahde.loop = true;
    aaniLahde.mute = true;
    while (!(Microphone.GetPosition(null) > 0))
    aaniLahde.Play();
}

void update()
{
    HaeSpektriTieto();
}

void HaeSpektriTieto()
{
    aaniLahde.GetSpectrumData(naytteet, 0, FFTWindow.Blackman);

    int nIndeksi = 0;
    int nKappaletta = 1;

        for (int i = 0; i <9; i++)
        {
        float keskiarvo = 0;

                for (int j = 0; j < nKappaletta; j++)
                {
                        keskiarvo += naytteet[nIndeksi];
                        nIndeksi++;
                }

                if (i == 8)
                {
                        keskiarvo += naytteet[nIndeksi];
                        nKappaletta ++;
                }
```

```
            spektriRyhma[i] = keskiarvo / nKappaletta;
            nKappaletta *= 2;
            }
      SpektriRyhmienSkaalaus();
}

void SpektriRyhmienSkaalaus()
{
      for(int i=0; i < 9; i++)
      {
            if(spektriRyhma[i] > spektriRyhmaKorkein [i])
            {
            spektriRyhmaKorkein[i] = spektriRyhma[i];
            }
      skaalattuSR[i] = (spektriRyhma[i] / spektriRyhmaKorkein[i]);
      }
}
}
```