OULUN AMMATTIKORKEAKOULU

Markku Leinonen

WEB AUDIO API -ÄÄNIGENERAATTORI

WEB AUDIO API -ÄÄNIGENERAATTORI

Markku Leinonen
Opinnäytetyö
Syksy 2015
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä(t): Markku Leinonen
Opinnäytetyön nimi: Web Audio API -Äänigeneraattori
Työn ohjaaja(t): Terhi Holappa
Työn valmistumislukukausi ja -vuosi: Syksy 2015 Sivumäärä: 32 + 3 liitettä

Tämän työn aiheena oli W3C-yhteisön tarjoama Web Audio API ja sen tarjoamat äänenmuokkausmenetelmät. Tarkoituksena oli kehittää äänigeneraattorisovellus, jossa olisi yksi tai useampi äänilähde (oscillator ja noise). Oskillaattoreiden tuottamaa ääntä tulisi olla mahdollista muokata erilaisin audio- ja elektroniikka-alan äänenmuokkausperiaattein. Näitä periaatteita ovat mm. volume, frequency, VCO, VCA, LPF, LFO, delay ja feedback. Idea aiheeseen saatiin elektronisen musiikin harrastajilta.

Tarkoituksena oli toteuttaa laitteistoriippumaton Google Chrome -verkkoselainpohjainen äänigeneraattorisovellus, käyttäen HTML5-, CSS3- ja JavaScript-ohjelmointikieliä ja Web Audio-ohjelmointirajapintaa. Tarkoituksena oli myös toteuttaa sovellus, joka ei tarvitsisi toimiakseen mitään riippuvuuksia eikä internet-yhteyttä, vaan sovelluksen tulisi toimia paikallisesti esimerkiksi PC:lla tai tablettitietokoneella. Tässä onnistuttiin varsin hyvin.

Projekti aloitettiin tutustumalla mahdollisiin olemassa oleviin Web Audio API -äänenmuokkaussovelluksiin. Tarjolla olevissa sovelluksissa huomattiin mm. Noise-ominaisuuden monipuolisen muokkauksen puute. Tässä projektissa kehitetty ja toteutettu äänigeneraattorisovellus toimii ilman kolmansien osapuolien tai muiden ohjelmoijien tekemiä apusovelluksia. Sovellus toimii Chrome-verkkoselaimessa täysin JavaScript- ja Web Audio API-pohjaisesti. Sovelluksen ulkoasu on toteutettu HTML5- ja CSS3-periaatteita käyttäen. Sovellus toimii sille määritellyssä tarkoituksessaan eli äänen tuottamisessa ja muokkaamisessa varsin hyvin. Sovellukseen jäi kehitettävää tallennusominaisuuden osalta.

Asiasanat: Web Audio API, HTML5, CSS3, JavaScript, Äänigeneraattori.

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technolgy, software development

---

Author(s): Markku Leinonen
Title of thesis: Web Audio API Sound Generator
Supervisor(s): Terhi Holappa
Term and year when the thesis was submitted: autumn 2015 Pages: 32 + 3 appendices

---

The subject of this project is Web Audio API offered by W3C. The aim of this project was to develop and implement a sound generator application using Web Audio API and HTML5, CSS3 and JavaScript programming languages. The application should be developed in a way that it could be used locally with Google Chrome web browser without any connection to internet. There should be at least two separate oscillator and Noise-sources in the application. The user should be able to modify this oscillating sound or noise according to the principles of the audio technic. The principles such as volume, frequency, VCO, VCA, LPF, LFO, delay and feedback must be included in the generator. The idea for the project was found out among electronic musicians. The application turned out to be very suitable for the purpose it was originally designed for.

Keywords: Web Audio API, HTML5, CSS3, JavaScript, Sound generator

ALKULAUSE

Kiitos kuuluu kaikille tavalla tai toisella tähän projektiin osallistuneille henkilöille, erityisesti perheelleni, joka joutui kuuntelemaan välillä toivotontakin sanailua asian tiimoilta. Pitkähkön ohjelmointitauon (n. 3 vuotta) jälkeen lähdin siis melkein alkutekijöistä liikkeelle. Projektin alussa opettelin HTML5-merkkauskielen ja CSS3-tyylikielen perusteet aivan alusta alkaen lukemalla aiheista kaksi Jukka K. Korpelan kirjoittamaa kirjaa. JavaScript-ohjelmointikieli näytti hiukan tutummalta, sillä olin aiemmin opiskellut olio-ohjelmointikieliä. Tämä projekti oli henkilökohtaisesti todella haastava, sillä tosiaan pitkän ohjelmointitauon jälkeen jouduin aloittamaan ohjelmoinnin likipitäen alusta. Web Audio API:n toimintaperiaate oli minulle etäisesti tuttu erään harjoitteluprojektin yhteydestä, mutta sen toteuttaminen käytännössä oli aivan uutta. Tämän kaiken yhteen sovittaminen tuntui aluksi liiankin haastavalta, jopa mahdottomalta. Opiskeltuani asioista enemmän ja tutustuttuani olemassa oleviin Web Audio API:a käyttäviin sovelluksiin alkoi projekti edetä.

Omasta mielestäni onnistuin ihan hyvin annetussa tehtävässä eli sovelluksen toteuttamisessa. Sen sijaan projektin hallinnassa minulla on vielä paljon opeteltavaa. Sovellus kuitenkin ajaa asiansa ja se ilahduttaa minua. Tulen käyttämään Äänigeneraattori-sovellusta soittimena omissa elektronista äänimaailmaa tarvitsevissa musiikkiesityksissä. Lisää ominaisuuksia tulen kehittämään ja toteuttamaan sovellukseen käyttökokemusten mukaan. Matkani oli pitkä, mutta kannattava.

Oulussa 2.12.2015

Markku Leinonen

SISÄLLYS

SANASTO

AudioNode = solmu, Web Audio API perustuu solmujen yhdistelyyn.

Delay = äänen kulun viivästäminen, jolla luodaan kaikuefekti.

Feedback = signaalin takaisinkytkentä, jolla tässä saadaan kaikuefekti toistumaan useita kertoja.

Frequency, freq = äänentaajuus.

LFO = Low Frequency Oscillator eli matalataajuusoskillaattori, on ääntä moduloiva efekti, joka tuottaa äänen rytmisen pulssin tai pyyhkäisyefektin esim ns. tremolo-, vibrato-, ripple- ja wobble-efektit.

Limiter = äänenvoimakkuuksien äkillisen nousun eli signaalipiikkien vaimennusmenetelmä, jolla estetään äänen leikkaantuminen eli säröytyminen.

LOWPASS - / HIGHPASS-filter = äänentaajuussuodatin. LOWPASS eli alipäästösuodatin, päästää rajataajuden alapuoliset taajuudet läpi ja HIGHPASS eli ylipäästösuodatin päästää rajataajuuden yläpuoliset taajuudet läpi. Asetetun taajuusrajan alapuoliset (HIGHPASS-filter) tai yläpuoliset (LOWPASS-filter) taajuudet siis suodatetaan eli vaimennetaan pois. Suodatus tapahtuu tietyllä jyrkkyydellä desibeli per oktaavi (Q-arvo).

Noise = kohina (Pink, White) eli äänitekniikassa käytetty kaikkia taajuuksia sisältävä ääniaalto.

Oscillator = oskillaattori = jaksollisesti värähtelevä järjestelmä. Tässä projektissa käytetään tuottamaan tietyn taajuista äänisignaalia.

Touchpad = kosketusalue eli mm. kannettavissa tietokoneissa käytettävä erillistä ohjaushiirtä vastaava komponentti, jolla ohjataan tietokoneen näytöllä hiiriosoitinta.

VCA = Voltage Controlled Amplifier, jänniteohjattu vahvistin.

VCO = Voltage Controlled Oscillator, jänniteohjattu oskillaattori.

Web Audio API = Web Audio Application Programming Interface eli Web Audio-ohjelmointirajapinta.

1 JOHDANTO

Projektin tarkoituksena on kehittää internet-selainpohjainen Äänigeneraattori-sovellus Web Audio API:n tarjoamalla äänenkäsittelyyn tarkoitetuilla ohjelmointi-rajapinnalla. Idea projektiin saatiin toimittajan omien ja muutaman elektronisen musiikin harrastajan ja ammattilaisen kokemuksista. Sovellusta vastaavien elektronisien ja elektromekaanisien laitteiden monipuolista sujuvaa käyttöä rajoittaa mm. kierrettävien säätönappuloiden likipitäen mahdoton yhtäaikainen käyttö ja kytkentäjohtojen runsaus. Erilaisten äänilähteiden ja äänenmuokkaus-laitteiden ketjuttaminen ja yhdistäminen toisiinsa voi olla jopa mahdotonta. Mekaanisiin laitteisiin verrattuna ohjelmistopohjaisessa äänigeneraattorissa etuna olisi mm. useiden äänilähteiden toisiinsa yhdistäminen, käyttäjällä olisi vähemmän laitteita siirreltävänä paikasta toiseen ja kosketusnäytöllä varustetulla tietokoneella olisi mahdollista käyttää useampia liukusäätimiä yhtäaikaa.

Vaatimusäärittelyn mukaan sovelluksessa tulisi olla äänilähdeoskillaattorin lisäksi kaiku-, kohina-, limitter- ja talletusominaisuudet (liite 2). Touchpad haluttiin mukaan kokeilumielellä, joten sen kehittäminen jätettiin lähinnä kokeilu-asteelle. Lisäksi sovellusta tulisi voida käyttää tietokoneen näppäimistöllä. Tämän projektin lopputuotteen eli WebAudioAPI_SoundGenerator-sovelluksen on tarkoitus toimia paikallisesti ilman internetyhteyttä.

Jo tarjolla olevista Web Audio APIin pohjautuvista sovelluksista huomattiin muutama seikka, joihin nimenomaan haluttiin parannusta. Ensimmäinen visuaalisesti tai mielleyhtymää häiritsevä ominaisuus oli se, että äänenmuokkausnappulat ovat ”pyöritettäviä” vaikkakin niitä käytetään vetämällä esim hiirellä joko ylös-alas- tai vasemmalta-oikealle. Käyttäjä siis ikään kuin pyörittää nappuloita vaikka käyttää vetoliikettä. Sovelluksissa tätä fyysistä vetoliikettä kuitenkin visualisoidaan nappulan pyörivällä liikkeellä. Juuri tämän visuaalisen harhan poistamiseen haluttiin käyttää liukukytkin-tekniikkaa. Kun käyttäjä vetää hiirellä tai sormella johonkin suuntaan tietokoneen näytöllä, niin myös säätökomponentti liikkuu visuaalisesti samankaltaisesti ja samaan suuntaisesti eli käyttökokemuksesta pyritään saamaan luonnollisempi. Toinen

suurehko puute tarjolla olevissa äänenmuokkaussovelluksissa oli Noise-ominaisuuden monipuolisen muokkauksen puute. Tähän ehdottomasti haluttiin parannusta.

Projektin toimittajana tavoitteeni on oppia Web Audio API- / JavaScript- ja HTML5- / CSS3-tekniikoita kehittämällä ulkonäöltään ja käytettävyydeltään käyttäjää miellyttävä äänigeneraattorisovellus, joka siten jossain määrin korvaisi vastaavia elektronisia ja elektromekaanisia laitteita. Äänigeneraattorin tulisi lisäksi kuullostaa samankaltaiselta kuin esimerkiksi Electronic Dream Plant WASP, Music From Outer Space W.S.G ja Korg Mono -tuotesarjan laitteet.

Projektin aluksi ja kuluessa kerättiin taustatietoja ja tutustuttiin jo olemassa oleviin Web Audio API:a käyttäviin web-sovelluksiin. Vaatimusmäärittelyn luomisen tueksi lähetettiin sähköpostikyselyt muutamalle elektronisen musiikin harrastajalle ja ammattilaiselle. Sovelluksen käyttöliittymä eli UI suunniteltiin ja toteutettiin vaatimusmäärittelyn pohjalta HTML-merkkauskielen uusinta versiota HTML5-kieltä ja CSS-tyylikielen uusinta versiota CSS3-kielitä käyttäen. Äänigeneraattorin ominaisuuksien testausta suoritettiin osa kerrallaan koko toteustuvaiheen aikana. Lopputestaukseen pyydettiin mukaan aiemmin vaatimusmäärittelyn yhteydessä haastateltuja henkilöitä.

2 WEB AUDIO API

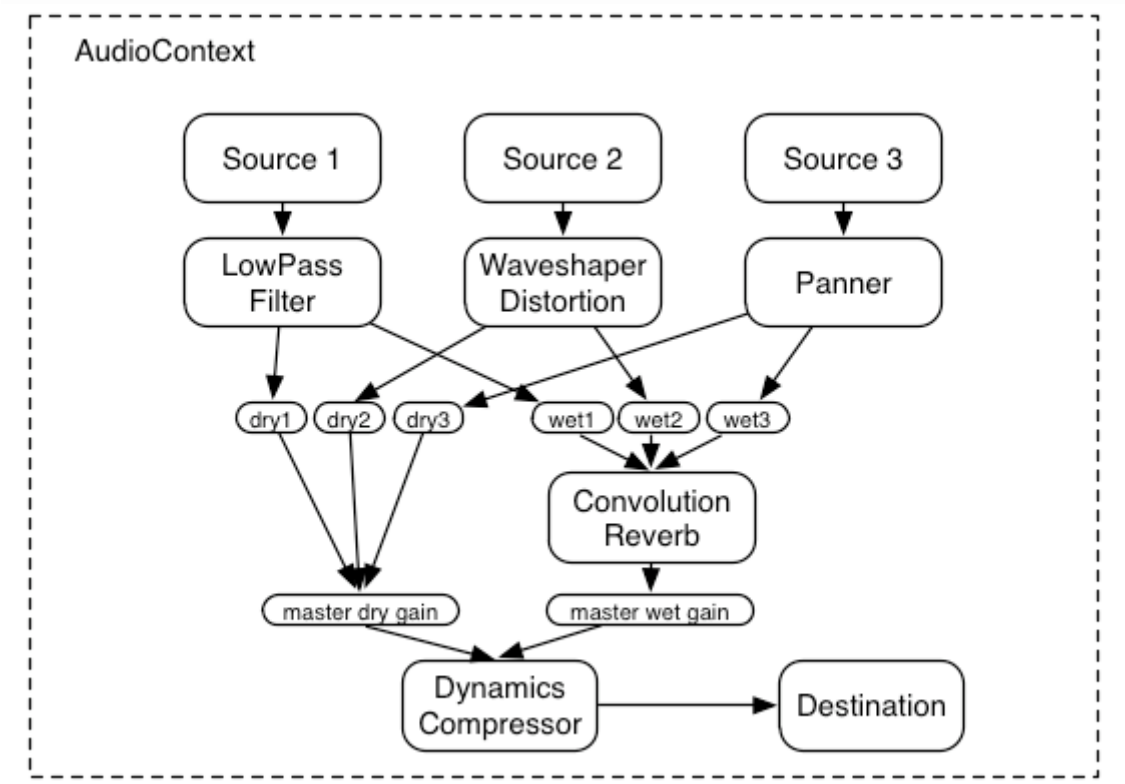
Web Audio API on eräs W3C:n (World Wide Web Consortium) kehittämistä yhteensopivuus Web-teknologioista. W3C on kansainvälinen järjestö, joka kehittää avoimia standardeja. Kehitetyt standardit määrittelevät avoimen Webin sovellusalustan (Open Web Platform), joka tarjoaa mahdollisuuden toteuttaa rikkaita ja vuorovaikutteisia sovelluksia, jotka ovat saatavilla millä laitteilla tahansa. Avoimen Webin sovellusalusta kulmakivi on HTML5. (1).

Tässä projektissa on käytetty Web Audio API -spesifikaatio versiota ”W3C Working Draft 10 October 2013” (2). Tulevaisuutta silmällä pitäen on tutkittu myös jossain vaiheessa tulevaisuudessa julkaistavaa uutta spesifikaatioversiota. Kehitteillä olevat versiot ovat luettavissa W3C web-sivuilla nimellä ”Nightly Draft”. Web Audio API ei siis ole vielä standardi. Web Audio API -spesifikaatio kuvaa ylemmän luokan JavaScript -ohjelmointirajapinnan äänen prosessointiin ja syntetisointiin web-sovelluksissa. Pääajatusmallina Web Audio API:ssa on äänen reitittäminen solmuobjekteja toisiinsa yhdistelemällä määritellä haluttu äänenmuokkauksen lopputulos. Tämän projektin lopputuote eli WebAudioAPI_SoundGenerator-sovellus on toteutettu ketjuttamalla useita solmuja toisiinsa. Lisäksi sovelluksen on tarkoitus toimia paikallisesti ilman internetyhteyttä.

2.1 AudioNode eli Audiosolmu

Web Audio API perustuu ns. AudioNode eli solmun käyttöön (kuva 1). Näitä solmuja toisiinsa yhdistelemällä ääntä muokataan halutun lopputuloksen saavuttamiseksi. Solmut voidaan jakaa neljään pääluokkaan: lähdesolmut, muuntelusolmut, analyysisolmut ja päätepistesolmut (3). AudioContext -rajapinta edustaa yhteyttä solmujen ja signaalien reititykselle päätepisteelle eli DestinationNode-solmulle. AudioNode-solmu edustaa AudioContextin objekteja ts. liityntöjä. Solmut luodaan create-motodilla esim. context.createGain(). Solmut yhdistetään toisiinsa connect-metodilla esim. gainNode.connect(delayNode). Solmut irrotetaan toisistaan disconnect-metodilla. Jokaisella solmulla on myös eri määrä sisääntuloja (input) ja

ulostuloja (output), joihin liitetään toisten solmujen sisääntuloja ja ulostuloja. Solmujen sisääntuloja ovat myös erilaiset parametri syötteet, kuten frequency, gain ja type.



KUVA 1. Modulaarinen solmujen reititys (2)

2.2 Web Audio API äänenkäsittely ja solmut tässä projektissa

AudioNode-solmut (W3C Web Audio API 4.2) ovat AudioContextin varsinaiset rakennuspalikat. Tässä projektissa edellä mainittuja solmuja ovat mm. oscillatorNode, destinationNode ja biquadfilterNode. OscillatorNode-solmu toimii äänenmuokkauksen kohteena olevan äänisignaalin äänilähteenä. DestinationNode on päätepiste, jonka käyttäjää ”kuulee”. BiquadfilterNode-solmulla muutetaan mm. äänensävyä. Jokaisella AudioNode-solmulla voi olla sisäätuloja ja ulostuloja (inputs ja outputs). Lähdesolmulla on vain ulostulo. Päätesolmulla on vain sisääntulo, ei ulostuloa. Solmuja voidaan kytkeä ympyrään eli ensimmäisestä solmusta toiseen ja toisesta takaisin ensimmäiseen. Tässä tilanteessa yhteyksien väliin on kytkettävä ainakin yksi delayNode. Tässä projektissa delay-efektiin liittyvä feedback-efekti on toteutettu

kuitenkin suoralla takaisinkytkennällä delayNode-solmuun. Solmujen liittymät luodaan AudioContext -rajapinnasta erilaisin create -metodein ja kytketään toisiinsa connect -metodilla. Tässä sovelluksessa käytetyt solmujen luontimetodit ovat createOscillator, -Gain, -BiquadFilter, -Delay, -DynamicsCompressor, -Analyser, -Buffer ja -SourceBuffer. Solmujen ominaisuuksia muutellaan AudioParam-liitynnällä.

AudioParam (W3C Web Audio API 4.5) on parametriliityntä AudioNode-solmuihin. Parametriliitynnällä annetaan arvot lähteen ominaisuuksille esim. gainNode.gain.value -parametri, jolla annetaan gain-solmulle gain -arvo. Muita parametriliittymiä tässä projektissa ovat detune ja frequency.

Äänilähteenä voi olla myös erityyppisiä äänitiedostoja, kuten wav, mp3, ogg. Tässä tapauksessa käytetään ns. SourceBuffer -äänilähdettä. Äänilähteen sisältö voidaan ladata verkkoyhteydellä internetistä, jolloin on käytettävänä XMLHttpRequest -metodi. Tämä tapa sopii tilanteeseen, jossa sovellus sijaitsee jollain verkkopalvelimella internetyhteyden takana. Tässä projektissa kehitettävässä sovelluksessa ei kuitenkaan tulla käyttämään verkkoyhteyttä eikä erillistä äänitiedostoa, vaan äänilähde luodaan paikallisesti omalla oskillaattorilla. Äänigeneraattori-sovelluksessa SourceBuffer-solmua käytetään paikallisesti ainoastaan Noise-efektin luomiseen tarvittavan kohina-äänen tuottamiseen, joten verkkoyhteyttä ei siis tarvita.

2.2.1 OscillatorNode Interface

OscillatorNode Interface (W3C Web Audio API 4.23) on tämän sovelluksen sydän eli äänilähde. Tällä solmulla luodaan ääntä tuottava lähde. OscillatoNode-solmulla on input ja output ja kolme parametriarvoa: type, frequency ja detune. Tuotettavan äänen taajuutta kontrolloidaan oskillaattorin frequency- ja detune-parametriarvoilla. Oskillaattorin aaltomuoto valitaan type-parametriarvolla. Oskillaattorin äänenvoimakkuutta säädetään siihen liitetyllä gainNode-solmulla. Sovellus sisältää useamman oskillaattorin, joista jokaisella on oma tarkoituksensa. OscillatorNode tuottaa alkuäänen, jota muokataan

halutun laiseksi. ModulatorNode (LFO) tuottaa FM-modulaatioon tarvittavan matalataajuussignaalin (0–15 Hz), jolla ohjataan oskillaattorin moduloitumista. Oskillaatorin taajuuttaa ohjataan oscillator.freqeuncy.value ja oscillator.detune.value -sisääntuloilla. Oskillaattorin ulostulotaajuus määräytyy kaavan 1 mukaan. Esimerkiksi detune -arvolla 0 saadaan oskillaattorin taajuudeksi oskillaattorille annettu frequency-arvo ja detune-arvolla 1200 saadaan ulostuloon annettu frequency-arvo kaksinkertaisena. Tätä käytetään sovelluksen scale-valitsin ominaisuudessa. Oskillaattorissa on valittavana äänitekniikastakin tutut neljä perusaaltomuotoa: sini-, kantti-, kolmio-, saha-aaltomuodot (sine, square, triangle ja sawtooth). OscillatorNode-solmussa on tarjolla viideskin ns. custom-aaltomuoto, mutta sitä ei tässä sovelluksessa käytetä. Oskillaattorin modulointi tapahtuu kytkemällä LFO-oskillaattori lähtö moduloitavan oskillaattorin frequency-parametrisisääntuloon. AudioParam on parametriliityntä AudioNode-solmuihin ja antaa arvot lähteen ominaisuuksille. Esimerkiksi modulatorGain.connect(oscillatorNode.frequency) -liitynnässä kytketään gain-parametri oskillaattorin taajuusparametriin.

*computedFrequency(t) = frequency(t) * pow(2, detune(t) / 1200)* *KAAVA 1*

frequency = äänentaajuus (Hz)
detune = numeroarvo –1200–1200

2.2.2 AudioParam Interface

AudioParam (W3C Web Audio API 4.5) on parametriliityntä AudioNode-solmuun, millä annetaan arvot lähteen ominaisuuksille. Esimerkiksi gainNode.gain.value -parametri, jolla annetaan gain-solmulle gain-arvo. Muita parametriliittymiä tässä projektissa ovat detune ja frequency.

2.2.3 DestinationNode

DestinationNode (W3C Web Audio API 4.4) on eräänlainen AudioNode, joka edustaa päätepistettä, jonka käyttäjä ”kuulee”. Tästä solmusta kytkeydytään käytettävän laitteiston ääniulostuloon esim. kaiuttimiin.

2.2.4 GainNode

GainNode (W3C Web Audio API 4.7) tarvitaan signaalin voimakkuuden säätämiseen. GainNode säätää mm. OscillatorNode-output- ja SourceBuffer-output-signaalien voimakkuuksia. Tässä sovelluksessa GainNodea käytetään myös moduloivana solmuna. ModGainNode kytketään moduloitavan oskillaattorin taajuussisääntuloon ja se säätää oskillaattoria moduloivan signaalin voimakkuutta VCO-periaatteen mukaan. GainNodella on input ja output ja gain -parametri.

2.2.5 BiQuadFilterNode

BiQuadFilterNode (W3C Web Audio API 4.21) tarvitaan äänensävyn säätämiseen. Tässä sovelluksessa solmu on LOWPASS- ja HIGHPASS-suodattimien ja MasterFilter-solmun käytössä (6).

2.2.6 DelayNode

DelaySolmua (W3C Web Audio API 4.8) käytetään äänen viivästämiseen. Delay-efektin luominen tapahtuu kytkemällä suoran signaalin kulun rinnalle DelayNode. Tässä sovelluksessa DelayNode kytketään Filter- ja Limiter-solmujen rinnalle. Ääni kulkee ilman delay-efektiä Filter-solmusta Limiter-solmuun. Kun kytketään Delay- ja Feedback-solmut näiden rinnalle, voidaan tätä rinnakkaisreittiä kulkevaa äänen kulkua viivästyttää verrattuna Filter-solmusta Limiter-solmuun kulkevaan signaaliin, jolloin saadaan kuuluviin kaiku-efekti.

2.2.7 DynamicCompressorNode

DynamicCompression-solmu (W3C Web Audio API 4.20) tasapainottaa hiljaisten ja voimakkaiden äänien suhdetta. Ihminen aistii matalat ja korkeat taajuudet eri äänenvoimakkuuksina. Etukäteen asetetun äänenvoimakkuuden ylärajan ylittävät äänenvoimakkuudet vaimennetaan ja asetetun alarajan alittavat signaalit voimistetaan, jotta solmun ulostuloon saadaan tasainen äänenvoimakkuuden vaihtelu. Solmu vaimentaa limiter-ominaisuudellaan äänenvoimakkuuksien ylittymiset. Kaikki äänenvoimakkuudet, jotka ylittävät

asetetun rajan, leikataan annettuun rajaan. Tässä sovelluksessa käytetään limiter-ominaisuutta leikkaamaan mahdolliset mm. kaiku- ja suodatinsolmuissa syntyvät äänen kertautumisesta johtuvat liian suuret äänenvoimakkuudet eli signaalipiikit.

2.2.8 AnalyserNode

AnalyserNode (W3C Web Audio API 4.17) toimii signaalin kuvaajana taajuus/aika- ja voimakkuus/aika -näytöllä. Solmu ei muokkaa signaalia, mutta saattaa aiheuttaa signaalin kulkuun viivettä. SoundGenerator-sovelluksen Osc1:n näytöltä nähdään masterFilterNode-solmun läpi kulkevan äänen kuvaaja. AnalyserNode riittää tämän projektin äänen visualisointi tarpeisiin. Tässä projektissa AnalyserNode palvelee käyttäjää näyttämällä ylivoimakkaiksi kasvavat signaalipiikit. Seuraamalla kuvaajia käyttäjä huomaa kasvavat piikit ja voi siis varautua äänenvoimakkuudeltaan kasvavien äänientaajuuksien vaikutuksiin kokonaisäänikuvassa. Ylivoimakkaat signaalipiikit vaimennetaan säätämällä esim. LO- tai HI-PassFilter-solmujen taajuusarvoja. Tässä projektissa on käytetty Boris Smus'n esittelemää Web Audio API mukaista äänen visualisointia (3). Äänen visualisointi -ominaisuus on Web Audio APIssa vielä kokeiluasteella, mutta tämän projektin tarkoitukseen se soveltuu oikein hyvin

2.2.9 AudioBuffer Interface

AudioBuffer-liityntä (W3C Web Audio API 4.9) edustaa muistia, johon lyhyt ääninäyte eli sample talletetaan. Tässä sovelluksessa AudioBufferia käytetään Noise-äänen tuottamisessa. AudioBufferin koko on n. 1 MB, joten kovin pitkiä ääninäytteitä siihen ei voi tallettaa, maksimissaan vain noin minuutin mittaisen. AudioBuffer määritetään PCM-äänen pituuden (length = sample-frames), näytteitä sekunnissa (samplerate) ja keston (audio data in seconds) mukaan. AudioBufferille annetaan myös tieto kanavien määrästä. Tässä sovelluksessa käytetään mono-kanavaa eli yhtä kanavaa, jolloin kanavamäärä arvoksi annetaan 1. Jos halutaan käyttää äänilähteenä pitempiä ääninäytteitä, tarvitaan käyttöön äänen ns. streamaus. Streamaus tapahtuu MediaElementAudioSourceNode:n avulla. Tässä sovelluksessa ei tarvita pitkiä

ääninäyteitä, vaan vain lyhyt kaikkia taajuuksia sisältävä sample, joten AudioBuffer riittää oikein hyvin. AudioBuffer luodaan createBuffer -metodilla. Tässä sovelluksessa käytetään Boris Smus'n esittelemää Whitenoise -muodostustekniikkaa (3). Boris Smus'n tarjoamaa mallia, joka tuottaa vain jatkuvaa kohinaa, on haluttu tässä sovelluksessa muuttaa, jotta noise-efektistä saataisiin monikäyttöisempi. Tässä sovelluksessa efektin kestoa ja esiintymistaajuutta voidaan säätää noiseTime- ja noiseFreq -muuttujilla. Nämä muuttujat vastaavat lenghtInSamples -arvoa createBuffer -metodissa (kaava 2), siten, että noiseTime on kytketty liukukytkimeen ja NoiseFreq -muuttujalle on etsitty kokeilemalla sellainen sopiva vakioarvo, jotta noiseTime -arvoa muuttelemalla saadaan aikaan kohina-efekti jatkuvasta kohinasta vain lyhyeen alle sekunnin mittaiseen noise-ääneen. NoiseFreq -arvo annetaan bufferille Noise-solmun luontivaiheessa. Noise-oskillaattorilla on myös detune -parametri, jolla säädetään noise-efektin pituutta.

var buffer = context.createBuffer(1, lengthInSamples, context.sampleRate) *KAAVA 2*

1 = numberOfChannels (1= mono, 2 = stereo)
lengthInSamples = pituus näytteinä
context.sampleRate = näytteitä sekunnissa

2.2.10 AudioBufferSource

AudioBufferSource-solmu (W3C Web Audio API 4.10) on verrattavissa OscillatorNode-solmuun. Tässä sovelluksessa AudioBufferSource-solmu on Noise-äänilähde. AudioBufferiSource-solmua käytetään, kun halutaan soittaa ääninäytteistä erimittaisia pätkiä. Tässä sovelluksessa tuotetaan AudioBuffer-solmusta muodostetusta lyhyestä kaikkia taajuuksia sisältävästä ääninäytteestä loppumaton silmukka eli loop, jolloin lopputuloksena on kohina eli Noise-efekti. AudioBufferSource-solmulle annetaan start-, stop- tai loop-komennot. Start- ja stop-komentoja käytetään silloin, kun halutaan soittaa lyhyitä pätkiä AudioBufferin sisällöstä, jolloin komennoille annetaan myös aikamääreet. Tämän sovelluksen kohina-ääntä soitetaan loppumattomassa silmukassa,

jolloin Noise-lähteelle tarvitsee antaa vain start- tai stop-komennot ilman aikamäärettä.

3 HTML5, CSS3 JA JAVASCRIPT

Sovelluksessa haluttiin käyttää viimeisimpiä versiota HTML5-, CSS3- ja JavaScript-ohjelmointikielistä. Näitä kieliä haluttiin käyttää sen vuoksi, että tämän hetkinen Web Audio API tukee parhaiten juuri näitä kieliä puhtaassa internet-sovellus ohjelmoinnissa. Web Audio API itsessään on toteutettu C++ -ohjelmointikieltä käyttäen. JavaScript -ohjelmoinnissa käytettiin kielen uusinta prototype -ohjelmointisuuntausta eli ns. luokatonta ohjelmointikieltä. Sovelluksen HTML5-, CSS3- ja JavaScript-ohjelmakoodit (liite 3).

3.1 HTML5 JA CSS3

Tämän projektin lopputuloksen eli äänigeneraattorisovelluksen haluttiin näyttävän ja toimivan kuten normaali internet-sivu, sillä erotuksella, että sovellus toimii paikallisesti ilman internetyhteyttä. Sovelluksen internet-sivu-tyyppinen runko on toteutettu HTML5 -periaatteita käyttäen. Sovelluksen ulkoasu käyttöliittymälle on luotu CSS3-tyylikielellä. HTML5-merkkauskielellä luodaan web-sivu ja sille näkymä. Luotuun sivuun määritellään HTML5-ohjelmakoodissa tekstit, ruudut, liukukytkimet ja painonapit ja näille toiminnot. Toiminnot liitetään JavaScript-ohjelmakoodiin funktioihin (liite 3 s. 1–12). CSS3-tyylikielellä saadaan aikaan tämän internet-sivun lopullinen ulkoasu. CSS3-ohjelmakoodissa määritellään sovelluksen taustavärit, ruutujen muodot, liukukytkinten värit ja muodot.

Uusina ominaisuuksina CSS3-tyylikielessä on mm. neliön kulmien pyöristys ja ulkoasun skaalautuminen erikokoisissa näytöissä. Jo näillä kahdella uudella ominaisuudella saadaan kehitettyä ulkonäöltään monimuotoisempia internet-sivustoja. Sovelluksen tyylitiedot annetaan ohjelmakoodin alkupuolella (liite 3 s.1-3). Esimerkiksi liukusäätimien nappulan eli thumb ulkoasu määritellään CSS3-osion kohdassa ”.eq::-webkit-slider-thumb” (liite 3 s.3). Ulkoasun määrittelyssä on pyritty ottamaan huomioon eri selainten vaatimuksia, mutta testauksessa on havaittu, että nuo vaatimukset muuttuvat ja ovat muuttuneet uusien selain versioiden myötä, joten on kaiketi mahdotonta kehittää kaikkien eri selainten kanssa täysin yhteensopiva internet-sivu.

3.2 JavaScript

Sovelluksen painonappeihin ja liukukytkimiin perustuvissa generaattoreissa käytettiin pääasiassa JavaScript.prototype-ohjelmointia. Hiiren liikkeisiin perustuvien canvas-ruutujen toiminta toteutettii perinteisellä JavaScript-ohjelmointikielellä, koska niiden toiminta ei tässä vaiheessa vaadi monimutkaisempia luokkien ja objektien perintämekanismeja. Ajatus käyttää Canvas-ruutuja äänenmuokkaamiseen perustuu elektronisen musiikin Kaoss Pad -soittimeen. Kaoss Pad on Korgin kehittämä elektroninen touchpad-tekniikkaan perustuva elektroninen soitin. Tässä projektissa touchpad:stä eli kosketusalueesta käytetään nimitystä KaaosPad. Sovelluksen KaaosPad -osiossa on käytössä vain muutama solmu: oskillaattori ja sille gain-solmu ja Noise-lähde eli SourceBuffer ja sille gain-solmu. Käytössä oleville neljälle aaltomuodolle on oma erivärinen ruutu. Normaaleihin oskillaattoreihin liitetyillä ruuduilla valitaan oskillaattroin tyyppi ja säädetään sen taajuutta ja siihen liitetyn gain-solmun gain-parametriarvoa eli oskillaattorin äänenvoimakkuutta. Noise-ruudulla säädetään noise-solmun kohinan päällä- ja pois -tilojen vaihtumisetiheyttä eli kohinataajuutta ja siihen liitetyn gain-solmun gain-parametriarvoa.

Prototype-based tai classless eli luokaton ohjelmointi on alkanut saada suurta suosiota mm. juuri JavaScript-ohjelmoijien keskuudessa. Tällä prototype -ohjelmoinnilla saadaan mm. web-pohjaisissa peleissä säästettyä selaimen käyttämää muistia. Perinteisessä luokka-ohjelmoinnissa jokainen objektin ilmentymä varaa selainmuistista tilaa perityille ominaisuuksille (mm. muuttujat ja funktiot), kun taas prototype-ohjelmoinnissa objektien ilmentymät käyttävät suoraan samoja objekti-muodostajan muuttujia ja funktioita, jolloin siis muistia säästyy. Prototype-objekti periyttää siis suoraan ominaisuutensa objektiensa ilmentymille. Tässä projektissa haluttiin käyttää Prototype-ohjelmointia mm. sen vuoksi, että tulevaisuuden kehitysideana sovelluksella on monipuolisemmat äänentuottamisen ja -käsittelyn mahdollisuudet, kuten näppäimistö-pianokoskettimisto ja modulaarinen muokkaustyökalujen käyttö. Modulaarisuus tarkoittaa mm. eri äänenkäsittelytyökalujen uudelleen järjestämistä reaaliajassa. Esimerkiksi LOW-/HIGH-PassFilter-solmu voidaan siirtää johonkin muuhun

kohtaan äänenmuokkausketjua tai lisätä useampia samoja suodattimia eri kohtiin äänenmuokkausketjua. Tällöin samojen BiQuadFilter -funktioiden käyttö olisi eduksi muistin säästämiseksi. Tietokoneen prosessorin tehoa säästyy prototype-ohjelmoinnissa myös, koska prosessorin ei tarvitse pitää yllä ylimääräisiä muisteja, vaan laskentatehoa voi käyttää mm. äänen muokkaamiseen, joka jo itsessään on runsaasti tietokoneen resursseja varaava toiminta.

4 SOUNGENERATOR-SOVELLUS

SoundGenerator-sovelluksen sydän on oskillaattori. Kun sovellus käynnistetään ja painetaan jostain neljästä käynnistysnapista (Osc1- tai Osc2-Play/Pause tai Noise1- tai Noise2-Play/Pause) tai painetaan hiiren vasemmalla napilla jostain viidestä erivärisestä canvas-ruudusta, käynnistyy kyseessä oleva oskillaattori aina ensin ennalta asetetulla äänentaajuudella ja -voimakkuudella. Oskillaattoreiden käynnistyttyä niiden tuottamaa ääntä voi alkaa muokkaamaan. Liukukytkimillä ja painonapeilla muokkaaminen sisältää enemmän variaatioita kuin canvas-ruuduilla muokkaaminen. Lisäksi molemmissa Osc-oskillaattoreissa ja Noise-oskillaattoreissa on hiukan erilaiset äänenmuokkausmahdollisuudet. Osc1 ja Noise1 ovat monipuolisempia toiminnoiltaan verrattuna Osc2:een ja Noise2:een. Osc2 ja Noise2 soveltuvat lähinnä perus- tai pohjaäänteen tuottamiseen. Näiden pohjaäänten päälle voi sitten tuottaa monipuolisempia ääniä Osc1:llä ja Noise1:llä.

Kaaospadejä on viisi. Neljällä kaaospadillä säädetään neljän eri oskillaattorin tuottaman signaalin taajuutta ja voimakkuutta. Nämä neljä oskillaattoria ovat harmaa eli siniaaltomuoto-, punainen eli kanttiaalto-, vihreä eli saha-aalto- ja sininen eli kolmioaaltomuoto-oskillaattori. Keltainen kaaospad on kohina-generaattori, jolla säädetään kohinataajuutta ja -voimakkuutta. Näissä ruuduissa parametrien antaminen perustuu hiiren liikkeen tunnistamiseen. Hiiren vasemman napin alhaalla pitäminen antaa asianomaisen ruudun määräämän tyyppi-parametrin kyseessä olevalle oskillaattorille. Hiiren liikkeen x-arvolla annetaan oskillaattorille taajuus-parametri ja hiiren liikkeen y-arvolla gain-parametri. Oskillaattori sammuu, kun hiiren vasen nappi vapautetaan.

Sovelluksen käyttöopasta ei olekaan tarkoitus kirjoittaa, vaan tarkoitus on nimenomaan se, että käyttäjä itse kokeilemalla ja säätämällä löytää miellyttäviä ääniä. Tässä selostuksessa käydään läpi ainoastaan ohjelman toiminta, ei siis ohjelman varsinaista yksityiskohtaista käyttöä.

4.1 Rakenne

Tyypillisessä internet-sovellustapauksessa tyylitieto-, JavaScript- ja HTML-tiedostot olisivat erillisiä tiedostoja ja HTML-tiedosto toimisi isäntänä, josta kutsuttaisiin muita tiedostoja. SoundGenerator -sovellus on kuitenkin toteutettu HTML5-ohjelmakoodirungon sisään. Kaikki ohjelmakoodi on kirjoitettu siis yhteen HTML-tiedostoon. Tarkoitus olikin kehittää sovellus ilman mitään ulkoisia tiedostoja ja internetyhteyttä. Tämän sovelluksen ohjelmakoodin voi kirjoittaa esim. Notepad++-sovelluksella, jolloin sen muokkaaminen onnistuu myös ilman ulkoisia yhteyksiä.

Ohjelmakoodin alussa kerrotaan, että dokumentin tyyppi on html. Dokumentin muuta informaationta annetaan head-elementin <head> ja </head> välissä. Sivun ulkoasun sisäinen tyylitiedosto annetaan myös head-elementin sisällä. Lisäksi se erotetaan muusta informaatiosta style-elementin <style> ja </style> väliin kirjoitettuna. Seuraavaksi luodaan body-elemetin <body></body> sisään käyttönapit (button), liukukytkimet (slider), jotka siis saavat ulkoasunsa edellä luodusta CSS-ohjelmakoodista. Canvas-elementit eli KaaosPadit saavat tyylitietonsa HTML-koodissa, niiden luonnin yhteydessä. Näin siksi, että ruutuja on vain viisi ja niiden tarvitsemat tyylitiedot ovat niin vähäiset, että on järkevää antaa canvas-ruutujen tyylitiedot jo niiden luonti vaiheessa. Javascript-ohjelmakoodi erotetaan muusta koodista script-elementillä <script></script>.

4.2 Käyttöliittymä

Käyttöliittymä oli tarkoitus kehittää sellaiseksi, että sovellusta tulisi voida käyttää sormin painelemalla kosketusnäytöllä varustetuilla kannettavilla tietokoneilla ja erityisesti tableteilla (kuva 2). Vastaavissa elektronisissa ja mekaanisissa laitteissa säätönappulat ovat useimmiten pyöritettäviä potentiometrejä, joita on vaikeaa, jopa mahdotonta, käyttää useampia samanaikaisesti. Tässä sovelluksessa pyöritettävät säätönappulat toteutettiin liukusäätimillä, jotta niiden käyttö samanaikaisesti useammalla sormella olisi mahdollista. UI on jaettu kahteen pääosaan, ruskea pohjaiseen Generator1:een ja violetti pohjaiseen Generator2:een. Lisäksi on erillinen touchpad-osa eli ns. KaaosPad-osio.

Molemmat kahdesta pääosasta sisältävät kaksi oskillaattoria Osc ja Noise ja näille äänenmuokkauseen liukusäätimet ja painonapit.

Generator1:

Osc1-osio sisältää liukusäätimet LIMITER, VOLUME, FREQUENCY ja painonapit SaveOsc1, LoadOsc1, Sine, Square, Trian, Saw, Play/Pause ja Scale-valinnat 128, 64, 32, 16, 8, 4, 2 ja 0.

Osc1 VCO LFO -osio sisältää liukusäätimet FREQUENCY, GAIN ja painonapit Sine, Square, Trian, Saw.

Osc1 LO/HI-PassFilter -osio sisältää liukusäätimet CUTOFF ja Q-VALUE ja painonapit LPF ja HPF.

Osc1 Delay/FeedBack -osio sisältää vain liukusäätimet DELAY ja FEEDBACK.

Noise1-osio sisältää liukusäätimet LIMITER, VOLUME, RATE, TIME ja painonapit Save Noise1, Load Noise1, Play/Pause, Infinit ja Chirp.

Noise1 Filter LO/HI -osio sisältää liukusäätimet CUTOFF ja Q-value ja painonapit LPF ja HPF.

Noise1 VCA -osio sisältää liukusäätimet RATE, GAIN, GRATE ja painonapit Sine, Square, Trian ja Saw.

Noise1 Delay/FeedBack -osiossa ovat vain liukusäätimet: DELAY ja FEEDBACK.

Generator2:

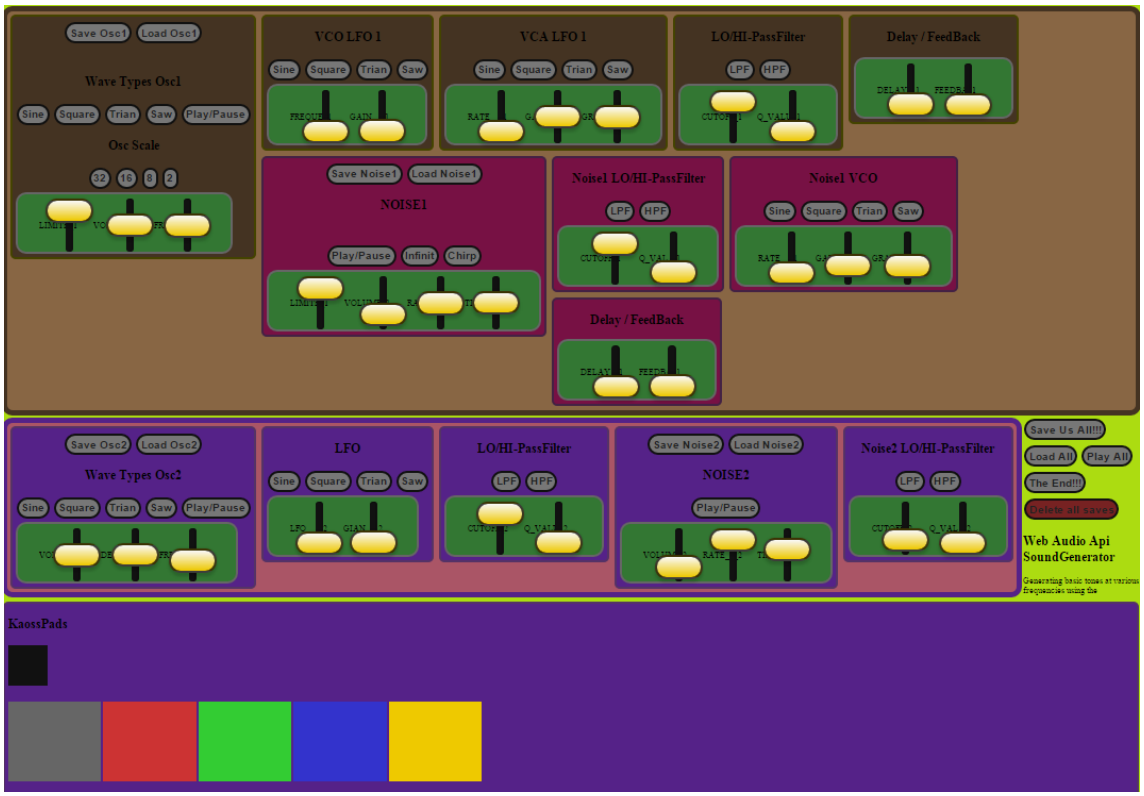
Osc2: sisältää säätimet: VOLUME, DETUNE, FREQUENCY ja painonapit: Save Osc2, Load Osc2, Sine, Square, Trian, Saw ja Play/Pause.

Osc2 LFO: FREQUENCY- ja GAIN-liukusäätimet ja Sine-, Square-, Trian- ja Saw-painonapit.

Osc2 LO/HI-PassFilter: CUTOFF- ja Q_VALUE-liukusäätimet ja LPF- ja HPF-painonapit.

Noise2: VOLUME-, RATE-, TIME -liukusäätimet ja Save Noise2-, Load Noise2- ja Play/Pause -napit.

Noise2 LO/HI-PassFilter: liukusäätimet CUTOFF ja Q-VALUE ja LPF- ja HPF-napit.



KUVA 2. Web Audio API SoundGenerator käyttöliittymä

4.3 SoundGenerator-sovellus

WebAudioAPI_SoundGenerator -sovellus käynnistetään käytössä olevan laitteen käyttöjärjestelmän sallimasta ohjelmakansiosta. Sovellus aukeaa oletusselaimeen. On kuitenkin hyvin suositeltavaa käyttää Google Chrome- tai Opera -verkkoselainta, koska esim. Firefox -verkkoselain ei ole täysin yhteen sopiva ja IE11-verkkoselaimella sovellus ei toimi laisinkaan. Safari -verkkoselainta ei ole voitu testata ollenkaan. Sovellus avautuu verkkoselaimeen ja luo käyttöliittymän ja ulkoasun niille laadituin parametrein. Parametrit löytyvät

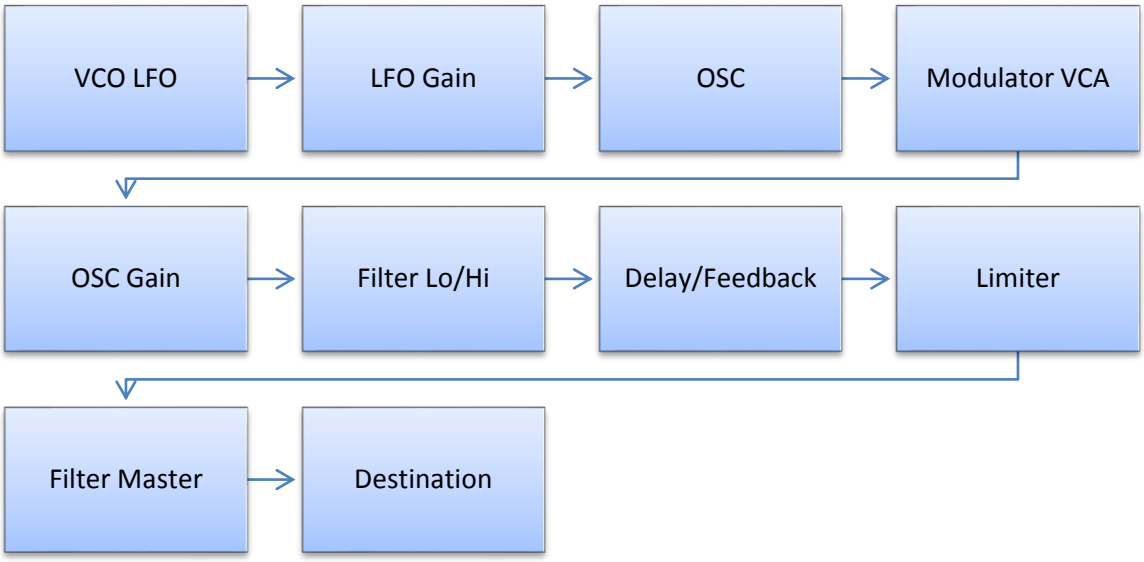
sovelluksen HTML5-merkkauskieli ja CCS3-tyylitieto-osuuksista. Liukusäätimet, painonapit ja sivunvärit muodostetaan edellä mainituista osuuksista löytyvin säännöin (liite 3 s.1-15). Esimerkiksi liukusäätimien ”thumb” ulkoasu määritellään CSS3 osion kohdassa ”.eq::-webkit-slider-thumb” (liite 3 s.4).

Varsinainen toiminnallinen osuus eli äänigeneraattori -osio alkaa ohjelma ”context=new (AudioContext || webkitAudioContext || function() { throw ”Browser does not support Web Audio API”; })();” (liite 3 s.15). Aluksi yritetään luoda AudioContext -liittymäsolmu, josta tarjotaan liittymät muille solmuille ja samalla tarkistetaan selaimen Web Audio API -yhteensopivuus ja ilmoitetaan käyttäjälle, jos käytetty selain ei tue Web Audio APIa. Seuraavaksi luodaan SoundGenerator -prototyyppi ja sille toimintofunktiot, jotka perivät SoundGenerator muodostajan ominaisuudet. SoundGeneratorilla on vain muutama muuttuja. SoundGenerator -prototyyppiobjektin ilmentymät soundGen1 ja soundGen2 luodaan aivan ohjelman lopuksi, jotta muut protoyyppiobjektin funktiot latautuvat ja konfiguroituvat ensin ja niiden jälkeen SoundGenerator -objekti on valmis, jotta siitä voi muodostaa ilmentymiä, jotka siis perivät prototyyppifunktiot. NoiseGenerator -prototyyppi luodaan omassa osassaan samalla tavalla kuin SoundGenerator.

SoundGenerator-sovelluksen sydän on siis oskillaattori. Oskillaattori käynnistyy ennalta asetetuilla oletus parametri asetuksilla (liite 3 s.15), kun painetaan esim. Osc1 Play/Pause nappia. Äänenkäsittelyketjut muodostetaan Web Audio APIn periaatteiden mukaan.

Äänenkäsittelysolmut kytketään toisiinsa seuraavassa järjestyksessä (kuva 3): OSCLFOVCO – GAINLFOVCO - freq.OSCVCO – GAINMODULATORVCAVCO – GAINOSC – Filter - Delay- GAINFeedback – Limiter – MasterFilter - Destination. OSCLFOVCA – GAINMODVCA-solmuketju kytketään GAINMODULATORVCAVCOn gain-parametriin. Filter-solmusta kytkeydytään myös suoraan Limiter-solmuun, jotta delay-efekti olisi mahdollista. Takaisinkytkentä toteutetaan Delay- ja Feedback-solmujen välillä. Feedback on gain-solmu, jolla säädetään Delay-solmuun takaisinkytkentäsignaalin voimakkuutta kaiku-efektissä. MasterFilter-solmu kytkeytyy lisäksi Analyser-

solmuun, joka näyttää äänen visuaalisena. VCA -moduloinnin voimakkuutta säädetään GAINMODVCAlla.



KUVA 3. Äänigeneraattorin äänenkäsittely solmuketju

Noise -äänenkäsittelyketju on periaatteeltaan samanlainen kuin normaalin oskillaattori solmuketju. Noise-ketju: NoiseSourceBuffer – GAINMODULATORVCANoise – GAINNoise – Filter – Delay – GAINFeedback – Limiter – MasterFilter - Destination. Filter-solmusta kytkeydytään myös suoraan Limiter-solmuun delay/feedback-efektin luomiseksi. OSCVCALFONoise - GAINVCANoise-solmuketju kytketään GAINMODULATORVCANoisen gain-parametriin. MasterFilter-solmusta kytkeydytään Analyser-solmuun ja takaisinkytkentä tehdään tässäkin solmuketjussa Delay- ja Feedback-solmujen välillä.

Limiter-solmulla estetään tarpeettomat voimakkaat signaalien leikkautumiset. Äänen leikkautuminen voi olla myös toivottua, jolloin limitteriä ei tule käyttää. MasterFilter-solmun rajataajuus on asetettu keskitaajuuteen 950, jonka molemmin puolin sekä alataajuuksilla että ylätaajuuksilla suoritetaan vaimennus. Lisäksi sovelluksen äänentaajusalue on rajattu n. 13 Hz – 10000

Hz. Tällä tajuusalueella taajuudet vaimenevat jyrkästi lähestyttäessä taajuusalueen ääripäitä. Taajuusalueen ulkopuolella signaali on vaimentunut lähestulkoon kokonaan. Näin turvataan suurilla äänenvoimakkuuksilla soitettaessa äänentoistolaitteiston toimivuus. Matalat taajuudet, jotka ovat alle laitteistojen suorituskyvyn, saattavat rikkoa mm. kaiuttimia ja saattavat olla suurilla äänenvoimakkuuksilla vaarallisia jopa terveydelle. Ylikorkeat taajuudet (suuremmat kuin 15 KHz) saattavat aiheuttaa epämiellyttävää tunnetta kuulijassa ja aiheuttaa hämmennystä eläimillä. Varsinkin ylemmät keskitaajuudet ovat vaarallisimpia ihmisen kuuloalueella ja voivat suurilla voimakkuuksilla aiheuttaa kuulovaurioita. Äkilliset ja voimakkaat kesitaajuudet voidaan vaimentaa sovelluksen limitteri-ominaisuudella. Feedback -ominaisuus toteutettiin takaisinkytkennällä Delay- ja Feedback-solmujen välillä (4).

Sovelluksen säätöjen talletus toteutettiin HTML5 Local Storage -menetelmää käyttäen (7). Tällä tallennusmenetelmällä voidaan paikallisesti tallettaa suuriakin määriä tietoa vaikuttamatta negatiivisesti internet-sivuston toimintaan. Tallennusominaisuus ei ole käytössä Äänigeneraattorin KaaosPad -osiossa. Sovellus tallettaa muistiin oskillaattorin asetukset, kun painetaan käynnissä olevan oskillaattorin Pause-nappia. Painettaessa kyseessä olevan oskillaattorin Play-nappia oskillaattori käynnistyy ja jatkaa tallentamillaan asetuksilla. Oskillaattorin asetukset voidaan tallettaa myös kovalevylle painamalla Save-nappia. Kun käyttäjä esim. sulkee internet-selaimen ja haluaa jatkaa myöhemmin edellistä äänenmuokkausistuntoa, voi tallentamiensa oskillaattoreiden asetukset ladata Load All-napilla tai yksittäisten oskillaattorien asetukset kyseessä olevan oskillaattorin Load -toiminnolla. Play All käynnistää kaikki oskillaattorit. Kaikkien oskillaattoreiden tilanteen voi tallettaa yhdellä kertaa Save Us All!!!-napilla ja ladata Load All -toiminnolla. The End!!! -toiminto nollaa kaikki oskillaattorit esiasetuksiinsa. Delete all saves -toiminto poistaa local storage -tallennukset kovalevyltä.

5 TESTAUS JA KOKEMUKSET

Web Audio API:n toimintaperiaatteen ansiosta sovellusta on voitu kehittää osio kerrallaan. Solmujen lisääminen olemassa olevien solmujen jatkoksi onnistuu kuin Lego -rakennuspalikoiden rakentelu. Aluksi projektissa siis opeteltiin HTML5-merkkauskieltä ja sen avulla luotiin käyttöliittymän protomalli ilman varsinaista ulkoasua. Käyttöliittymään lisättiin CSS3-tyylikielellä värejä ja muotoja ja testattiin niiden toimivuutta keskenään. Web Audio API:n tarjoamat solmut otettiin käyttöön yksi kerrallaan ja yhdistettiin toisiinsa ja toimivuus testattiin antamalla parametriarvoja ohjelmallisesti. Solmujen parametriliittymät yhdistettiin toisiinsa ja käyttöliittymään JavaScript-ohjelmointikielellä ja niiden toimivuutta testattiin.

Sovellusta on kehitetty ikään kuin ketterän kehittämisen periaatteen mukaan eli on lähestytty päämäärää eli toimivaa sovellusta osio kerrallaan (5). Kehitys ja toteutus on suoritettu HTML5:stä CSS3:n kautta solmuihin ja JavaScript-ohjelmointiin. Jokaisen sprintin aikana on opeteltu uusi asia kokonaisuus ja implementoitu se edellisen vaiheen lopputulokseen. Testausta on suoritettu samanaikaisesti sovelluksen kehityksen yhteydessä. Kun on opittu ja sisäistetty uusi asia, se on liitetty osaksi jo olemassa olevaa sovellusta. Jokaista sovelluksen ominaisuutta on testattu erikseen saatavilla olevissa eri internet-selaimissa ja laitteissa. Sovellusta on kehitetty tarkoituksella Google Chrome -yhteensopivaksi, koska Chrome-selain on saatavilla suurimpaan osaan tarjolla olevista laitteista. Google myös osallistuu tiiviisti Web Audio API kehitystyöhön. Tämänkin projektin lähteenä käytetyn Web Audio API -kirjan kirjoittanut Boris Smus työskentelee Googlella. Sovellus on kaikilta osiltaan ja ominaisuuksiltaan yhteensopivin Google Chrome-verkkoselaimen kanssa.

Suurimpia ongelmia yhteensopivuudessa on Noise-ominaisuudessa. Firefox ei suostu tuottamaan Noise-ääntä, eikä sitä suostu Chrome-selainkaan tuottamaan sitä esim. Jolla puhelimessa. KaaosPad ei toimi halutulla tavalla tabletilla. Oskillaattorit voidaan käynnistää, mutta ääntä ei voi muokata. Osc1 ja Osc2 toimivat melkein kaikissa testatuissa laitteissa ja internet-selaimissa. Selainyhteensopivuuteen on ainakin yksi ratkaisu olemassa, AudioContext-

MonkeyPatch -javascript -tiedosto. Koska tässä projektissa ei haluttu käyttää ulkoisia ohjelma-scriptejä eikä lisätä muiden ohjelmoijien tekemiä monimutkaisia scriptejä sovellukseen, niin MonkeyPatch'n käyttö hylättiin.

Ulkoasu näyttää juuri halutun mukaiselta vain Google Chrome- ja Opera -verkkoselaimissa. Firefox- ja IE -verkkoselaimet eivät ymmärrä kaikilta osin tässä sovelluksessa määriteltyjä liukusäätimien "thumb"-muotoiluominaisuuksia. Sovellus ei toiminut ollenkaan IOS 9 -käyttöjärjestelmällä varustetussa iPhone 5 -älypuhelimessä.

Testauksessa käytettiin Google Chrome -verkkoselaimen omaa JavaScript-konsolia. Konsolitilaan pääse ctrl+shift+j -näppäinyhdistelmällä. Ohjelmakoodiin voidaan haluttuihin paikkoihin kirjoittaa mm. komento 'console.log()' ja sille sulkujen sisään jokin parametri esim. 'soundGen1.gain'. Sovelluksen ollessa käynnissä tämä esimerkkikomento näyttää konsolin näytöllä, onko tilanne true vai false eli tosi vai epätosi, jolloin saadaan selville, onko gain-parametri edes olemassakaan. Sovelluksen toimintaa voi seurata myös alert() -funtiolla. Alert()-funktio avaa ponnahdusikkunan verkkoselaimessa. Kirjoittamalla haluttuun kohtaan ohjelmakoodia esim. alert(soundGen1.gain.value) saadaan ponnahdusikkunaan tietoa gain-parametrista.

Testauksessa todettiin sovelluksen äänenmuokkausominaisuuden toimivan kiitettävästi Google Chrome -verkkoselaimella Win7 -käyttöjärjestelmässä ja hyvin Android -käyttöjärjestelmällä Samsung Tab 3 lite -tabletissa Google Chrome -verkkoselaimella. Android -käyttöjärjestelmällä ja Google Chrome -verkkoselaimella ei touchpad -toiminto toimi täydellisesti. Oskillaattorit voidaan käynnistää, mutta äänenmuokkaus ei ole mahdollista.

Vaatimusmäärittelyssä mainittu asetusten talletus -toiminto ei valitettavasti toimi täydellisesti. Toivottu tilanne olisi, että käyttäjä voisi itse antaa nimen talletettavalle tiedostolle ja määrätä sille myös tallennuspaikan. Tässä sovelluksessa käyttäjä ei valitettavasti voi itse valita nimeä eikä tallennuspaikkaa tiedostolle tallennuksen yhteydessä eikä siksi myöskään voi valita, mitä tallennuksia haluaa ladata sovellukseen. Tätä voitaneen pitää

harmillisena ominaisuutena sovelluksessa käytetyssä local storage -tallennuksessa.

Sovelluksen äänenmuokkaus mahdollisuudet ovat verrattain hyvät verrattuna esim. Korg Monotribe -syntetisaattoriin ja erittäin hyvät verattuna esim Weird Sound Generator -syntetisaattoriin. Äänigeneraattorin oskillaattorien tuottamaa signaalia voidaan muokata todella laaja-alaisesti puhtaasta siniaaltomuodosta kantti-aaltomuotoiseen VCO- ja VCA-modulaatio sekoitteisiin ja lisätä näihin kaiku-efekti lisättynä feedback-efektillä. Jo ainostaan VCO- ja VCA-äänenmuokkauksen perusperiaatteella saadaan tuotettua monimuotisia ja mielenkiintoisia ääniä. Noise-oskillaattorin tuottaman äänen muokkausta pidetään erittäin hyvänä ominaisuutena. Tässä sovelluksessa käyttäjän ei tarvitse tyytyä vain pelkkään yksitoikkoiseen kohinaan, vaan muutamalla lisäefektillä saadaan aikaan todella mielenkiintoisia kohina-efektejä. Kun molempien oskillaattorityyppien niin normaali- kuin noise-oskillaattorinkin tuottamaa signaalia voidaan muokata itsenäisinä komponenteina, täyttää sovellus hyvin sille annetut äänenmuokkauksen perusvaatimukset.

6 YHTEENVETO

Tämän projektin tavoitteena oli kehittää ja toteuttaa laitteistoriippumaton, lähnnä Google Chrome -verkkoselaimessa toimiva, äänigeneraattorisovellus. Sovelluksen tuli toimia ilman internetyhteyttä ja ulkoisia aputiedostoja. Kehitettävän äänigeneraattorin tuli lisäksi olla käytöltään ja äänimaailmaltaan verrattavissa samankaltaisiin mekaanisiin ja elektronisiin laitteisiin.

Tässä projektissa kehitetty sovellus on täysin yhteensopiva vain Google Chrome-verkkoselaimen kanssa. Optimaalisin tilanne olisi luonnollisesti yhteensopivuus kaikkien tarjolla olevien selainten kanssa, mutta jokaisella eri selainkehittäjällä on omat tavoitteet ja tarpeet, jotka lisäksi tuntuvat vaihtuvan ja muuttuvan milloin mitenkin. Tästä johtuen tällä hetkellä lienee mahdotonta kehittää täysin selainriippumaton selainpohjainen Web Audio API-sovellus. Sovelluksen käyttöliittymä järjestyy käyttäjäystävällisesti selaimen koon muuttuessa. Oskillaattorit pysyvät omina ryhminään, vaikka selaimen leveys tai korkeus muuttuu. Tabletilla KaaosPad -ominaisuutta ei saatu toimimaan oikein, vaan oskillaattorit saadaan vain kytkettyä päälle, muttei kytkettyä pois päältä. Myöskään äänenmuokkaus ei toimi KaaosPadilla. Muun muassa näihin puutteisiin tullaan etsimään ratkaisu seuraavissa sovelluksen kehitysvaiheissa. Sovelluksen käyttö tablet-tietokoneella onnistuu kuitenkin riittävän hyvin ilman KaaosPad -ominaisuuttakin.

Muita tulevaisuuden kehitystavoitteita sovellukselle ovat mm. asetusten tallennus nimellä, äänenmuokkaus-solmujen reitityksien ja ketjujen muuntelu reaaliajassa ja mikrofonin käyttö äänilähteenä. Lisäksi sovelluksesta saadaan visuaalisesti miellyttävämpi, kun liukusäätimien nimet implementoidaan liukusäätimien säätönuppeihin. Lopputestauksessa toivottiin liukusäätimien arvoja näytölle.

LÄHTEET

1. W3C (World Wide Web Consortium) Suomi. Saatavissa: http://www.w3c.tut.fi/. Hakupäivä 28.10.2015.

2. Web Audio API. W3C Working Draft 10 October 2013. Saatavissa: http://www.w3.org/TR/webaudio/. Hakupäivä 2.6.2015.

3. Smus, Boris 2013. Web Audio API ilmainen nettikirja. Saatavissa: http://chimera.labs.oreilly.com/books/1234000001552/index.html. Hakupäivä 2.6.2015.

4. Lowis, Chris July 23, 2014. Creating dub delay effects with the Web Audio API. Saatavissa: http://blog.chrislowis.co.uk/2014/07/23/dub-delay-web-audio-api.html. Hakupäivä 25.10.2015.

5. Tolvanen, Perttu 2013. Ketteryys haltuun: Ketterän kehityksen yleiset periaatteet. Sininen Meteoriiti. Saatavissa: http://www.meteoriitti.com/2013/06/06/ketteryys-haltuun-ketteran-kehityksen-yleiset-periaatteet/. Hakupäivä 27.5.2015.

6. Rogers, Chris with modifications by Wilson, Chris. Web Audio API Frequency response. Saatavissa: http://webaudioapi.com/samples/frequency-response/. Hakupäivä 14.9.2015.

7. Web Storage. w3schools.com. Saatavissa: http://www.w3schools.com/html/html5_webstorage.asp. Hakupäivä 28.10.2015.

LIITTEET

Liite 1 Lähtötietomuistio

Liite 2 Vaatimusmäärittely

Liite 3 WebAudioAPI_SoundGenerator.html ohjelmakoodi

LÄHTÖTIETOMUISTIO

Tekijä: Markku Leinonen

Tilaaja: Markku Leinonen

Tilaajan yhdyshenkilö ja yhteystiedot: Markku Leinonen 050-3212899
t4lema01@students.oamk.fi

Työn nimi: **Web Audio API Äänigeneraattori**

Työn kuvaus: Tarkoituksena on tehdä internetselain pohjainen Äänigeneraattori Web Audio API:n tarjoamilla äänenkäsittely rajapinnoilla. Fyysisten elektronisten ja mekaanisten vastaavien laitteiden monipuolista käyttöä rajoittaa säätönappuloiden likipitäen mahdoton yhtäaikainen käyttö ja erilaisten laitteiden yhdistäminen toisiinsa voi olla jopa mahdotonta. Kosketusnäytöllä äänensäätämiseen voidaan käyttää jopa kymmentä sormea yhtäaikaa. Fyysisiin laitteisiin verrattuna, softaware pohjaisessa äänigeneraattorissa useiden äänilähteiden yhdistäminen ja lisäetuna vähemmän laitteita siirreltävänä paikasta toiseen. Vaatimusäärittelyn avuksi toimittaja haastattelee muutamaa elektronisenmusiikin harrastajaa/ammattilaista

Työn tavoitteet: Tavoitteena oppia Web Audio APIn ja JavaScriptien ja Html5 ja CSS yhdistäminen. Yrittää saada kehitettyä ulkonäöltään ja käytettävyydeltään käyttäjää miellyttävä äänigeneraattori, joka mahdollisesti korvaisi mekaanisia laitteita. Ja äänigeneraattorin tulisi lisäksi kuullostaa uskottavalta esimerkiksi verrratuna Electronic Dream Plant WASP, Music From Outer Space W.S.G ja Korg Monotron tuotesarjan laitteisiin.

Tavoiteaikataulu: 20.5.2015 - 31.10.2015. Lähetetään email kyselyt harrastajille/ammattilaisille. Aluksi toimittaja tutustuu Html5 selainsovelluskehitys- ja CSS/CSS3-kieliin. Seuraavaksi vuorossa Web Audio API ja JavaScript. Ja lopuksi näiden kielien yhdistelmällä toteutetaan selainpohjainen Äänigeneraattori.

Päiväys ja allekirjoitukset: 25.05.2015 Markku Leinonen

KEHITETTÄVÄN SOVELLUKSEN VAATIMUSMÄÄRITTELY

Sovelluksen vaatimusmäärittelyn tueksi lähetetyt kyselyt:

Kysymykset:
1. Tarvitaanko yleensäkään elektronisen musiikin mekaanisia laitteita korvaavia sovelluksia/ohjelmia? Miksi? ja jos ei, niin Miksi ei?

2. Millaiselta sovelluksen tulisi teidän mielestänne näyttää ja "tuntua" (Voitte vaikka piirtää ja liittää kuvan mukaan)?

3. Mitä ominaisuuksia, joita olisitte mekaanisissa vastaavissa tarvinnet, sovelluksessa tulisi vähintäänkin olla (Tässä ehkä lähinnä haen liitettävyyttä, ketjutusta ja kytkettävyyttä toisiin laitteisiin)?.Ja lisäksi olisihan se käytännöllistä, jos voisi laittaa esim kolme WASPin tapaista sovellusta yhtäaikaa äänteleмään ja periaatteessa kaikkia voisi säädellä yhtäaikaa kolmesta näytöllä olevasta moduulista, vai mitä? ja Esim tarviiko välttämättä pianonkoskettimistoa?

4. Mitä lisäominaisuuksia sovelluksessa tulisi vähintäänkin olla (esim efektejä: kaiku, chorus jne)?

5. Muuta sanottavaa ja ideaa aiheeseen?

Vastaukset:

Jaakko – Jaakko Vanhala:

Wesku – Vesa Moilanen:

Kysymykset (kysymykset myös docx liitteenä):
1. Tarvitaanko yleensäkään elektronisen musiikin mekaanisia laitteita korvaavia sovelluksia/ohjelmia? Miksi? ja jos ei, niin Miksi ei?

-Varmasti joku tarvitsee. Esimerkiksi itselläni ei ole ollut tilaisuutta tutustua kovinkaan kattavasti näihin laitteisiin vaikka kiinnostusta olisi. En myöskään halua hankkia kalliita kojeita ilman testaamista. Korvaavilla sovelluksilla voi tutustua aiheeseen ja simuloida alkuperäisiä laitteita ja näin oppia tietämään mitä tarvitsee ja haluaa käyttää. En myöskään epäile etteikö sovelluksia voisi käyttää sellaisenaan esimerkiksi äänityksiin, ilman että alkuperäisiä mekaanisia laitteita hankkisi ollenkaan. Tunnepohjainen halu käyttää mekaanisia laitteita toki on sitten asia erikseen, ja siihen monet varmasti nojaavat.

2. Millaiselta sovelluksen tulisi teidän mielestänne näyttää ja "tuntua" (Voitte vaikka piirtää ja liittää kuvan mukaan)?

-Liukunapit jotka vastaavat alkuperäisiä laitteita olisi hyvä. Ehkä lukuarvot voisivat helpottaa hahmottamaan ja muistamaan jos haluaa muistaa jotain nappien asentoja. Mahdollisesti voisi olla hyvä päästä zoomaamaan lähelle nappia jotta äärimmäinen hienosäätö olisi tarvittaessa mahdollista.

3. Mitä ominaisuuksia, joita olisitte mekaanisissa vastaavissa tarvinnet, sovelluksessa tulisi vähintäänkin olla (Tässä ehkä lähinnä haen liitettävyyttä, ketjutusta ja kytkettävyyttä toisiin laitteisiin)?.Ja lisäksi olisihan se käytännöllistä, jos voisi laittaa esim kolme WASPin tapaista sovellusta yhtäaikaa ääntelemään ja periaatteessa kaikkia voisi säädellä yhtäaikaa kolmesta näytöllä olevasta moduulista, vai mitä? ja Esim tarviiko välttämättä pianonkoskettimistoa?

-Itse en tarvitse koskettimistoa lainkaan, mutta en tietenkään osaa muiden puolesta sanoa siihen mitään. Olisi tosiaan kätevää jos pystyisi käyttämään kolmea eri laitetta yhtä aikaa. Sellainen voisi olla mielenkiintoinen, että kolmesta laitteesta lähtisi kolme eri ääntä, mutta niiden nousevia ja laskevia taajuuksia voisi hidastaa ja kiihdyttää yhtä aikaa. Niin ne kaikki pelaisi samaa tarkoitusta varten yhtä aikaa. Taajuuden kiihtyminen aiheuttaa kuulijassa nousujohteista fiilistä ja laskeva päinvastoin. Jos kolme laitetta nostaisi ja kiihdyttäisi taajuuksia yhtä aikaa, ne pelaisivat samaan pussiin, aiheuttamatta ristiriitaa.

4. Mitä lisäominaisuuksia sovelluksessa tulisi vähintäänkin olla (esim efektejä: kaiku, chorus jne)?

-Kaiku, flanger, jne. Esim. meidän Hazard Wings bändissä efektimies on ajanut efektipedaalin ääniä myös wah-wahin läpi. Olen aivoin kokeilemaan ihan mitä vaan mitä on tarjolla, mutta ei tällainen sovellus sinänsä vaadi mitään.

5. Muuta sanottavaa tai ideaa aiheeseen?

-Kiinnostavaa nähdä miten tämä homma kehittyy! Olen kiinnostunut mekaanisista laitteista omissa projekteissani mutta tällaisilla sovelluksella vois hyvin simuloida ja kokeilla kuinka homma toimii. Ja kun sovellukseen tutustuu, se voi onnistuessaan jäädä tietenkin ihan korvaamattomaksi työkaluksi

Jukka – Jukka Ruohomäki:

Läjä – Veli-Matti Äijälä:

Vastaamme:
1.mielestäni tarvitaan, vaikka harrastan analogisia soittimia. Sovelluksista ja ohjelmista helppokäyttää/nopea ja muistitilaa, uudenlaista muotoilua ja myös eri sovellusten yhteenliitettävyyttä kaipaan
2. mm. "lego" palikka malli, eri värisävyillä , läpikuultavuus myös,omien mieltymysten mukaan, myös halpa mutta toimiva.
3.olen juuri samoilla mielin, yhteenliittävyyttä etc koskettimia minun mielestä ei tarvitse, esim liukusäädöt erittäin hyvä
4.kaikki ns yleiset ominaisuudet eri vaihtoehdoilla, mites myös valo/lämpö/liike ?
5.idea erittäin hyvä ja mielenkiintoinen...odotan lämmöllä

Pena – Pentti Dassum:

terve lene

sori viive, en kattele tätä gaymailii niin usein, muista käyttää pentti@umpio.com

ok täältä tulee:

1. tarvitaan sillä miks muka ei. tuskin ne rautaa korvaa, mutt uskon ett ajan kanssa opitaan rinnakkaiselämän soljuvuutta paremmin. näiden atk- sekä älykalujen tehot vaan yltyy nii tottakai niitä kuuluu hyödyntää.

2. no comments. tuo sun malli on yhtä selkee ku mikä tahansa "hienompi" grafiikaltaan... jotkut plukarit tai ohjelmat on ihan hirveen näkösii, mutta kunhan soi nii ihan sama... esim ableton on jotenki ihan kökkö, mutt sitä pidetään huippuna, taasen esim puredata on kans kökkö ja on oikeesti huippu vaiks ei olekaan mainstreamii... ja esim kun ite käytän reaper.ii nii siinä on semmone valintamhdollisuus joka plukarille erikseen, jossa voi "sammuttaa" plukarin oman GUI.n ja tilalle tulee geneeriset liu'ut, nii kyllä mä käytän sitä usein, sillä joskus on selkeempää nähdä parametrit ilman krumeluureja, kts

http://reaper.mj-s.com/SendReaControl_alone.jpg

siin yläoikealla lukee UI nii siitä pystyy vaihtaa jos tarvii.... noin niinku esimerkkinä..

tiedän kans ihmisii jotka peräänkuuluttaa "seksikästä" käyttöliittymää (arghh)

3. jos moduleja alkaa kasailee useampaa ruudulle, nii output-summan pitäis ainaki sitten toimii ilman ett ne kaataa toisensa... eli meneeks lopulta kaikki vaan "master L + R" ulos... ite oon tykänny ku voi ihan tästä kirjoitusnäppiksestä soitella, verrattuna siihen ett klikkailis pianon näköstä kiipparii hiirellä... ja sitten megabonus: jos alkaa olee useempaa kaluu ruudulla nii voiko luoda makroja, joilla sitten voi "niputtaa" toimintoja, esim yhdestä "potikasta" voi ohjailla nii monta parametrii ku haluaa, ja vielä vaikka niin, et jotain voi ohjata vastakkaiseen suuntaan... niinku et jos vaiks yhden nappulan takana on 3 parametrii niin joku niistä menis "kiinni" ku muut menee "auki"... ainaki tässä sitä pysty tekemään

http://www.vstforx.de/

mun paskakone ei vaan jaksanu sillon pari vuotta sitten pyörittää tota demoversioo kauheen hyvin.... mutt sai hiukan esimakua..

3. limitteri outputin viimiseks asiaks vakiokaluna jarruttamaan piikkejä!!!! vituttaa ku reaktorissa esim tota ei o vakiona nii saattaa tulla kauheita räjähdyksii ku rupee sohimaan... ite käytän tätä joka käänteess, masterissa aina, ja joskus "irtoraidoilla"

http://www.yohng.com/software/w1limit.html

ja vieläpä mielummin toi "old version" ku se on kevyempi...

4. anna palaa frank!!

olikohan tuossa nyt mitään järkee..... ilmottaudun beta-testaajaks näillä puheilla!!

kuulolla + terv. pentttttttttd

VAATIMUSMÄÄRITTELY:

Haastattelujen ja omien suunnitelmien perusteella Web Audio API SoundGenerator sovellukseen pyritään toteuttamaan: ainakin kaksi eri itsenäisesti toimivaa oskillaattoria, Noise-lähde, säätöjen tallennus mahdollisuus, limitteri leikkaamaan piikkejä ja kaiku. Lisäksi toimittaja haluaisi tutkia ja toteuttaa sovellukseen Korg:n kehittämän KaossPad (touchpad) tyyppisen äänimuokkaus mahdollisuuden.

```
<!DOCTYPE html>
<head>
  <title>Sound Generator | Web Audio API</title>
  <style>
body {
  background-color: #acdc11;
  padding: 0px;
  margin: 1px;
}

/*kuvia varten*/
img {
    width: 19%;
    float: left;
    border:2px solid blue;
}

/*omat sliderit ja napit*/
.osc1 {
            padding: 2px;
            }
.osc1, .intro  {
            float: left;
            margin: 2px;
            background-color: #864;
            border: 10px solid #432;
            border-radius: 20px;
            }

.group1 {
            float: left;
            margin: 5px;
            padding: 5px;
            border: 5px solid #440;
            border-radius: 10px;
            background-color: #432;
            }
.slider1 {
            padding: 40px;
            }
.slider1, .intro  {
            float: left;
            margin: 1px;
            background-color: #373;
            border: 5px solid #666;
            border-radius: 20px;
            }

.osc2 {
            padding: 2px;
            }
.osc2, .intro  {
            float: left;
            margin: 2px;
            background-color: #A56;
            border: 10px solid #528;
```

```
            border-radius: 20px;
            }
.group2 {
            float: left;
            margin: 5px;
            padding: 5px;
            border: 5px solid #536;
            border-radius: 10px;
            background-color: #528;
            }

.slider2 {
            padding: 40px;
            }
.slider2, .intro  {
            float: left;
            margin: 1px;
            background-color: #373;
            border: 5px solid #666;
            border-radius: 20px;
            }

/*Noise */
.group3 {
            float: left;
            margin: 5px;
            padding: 5px;
            border: 5px solid #424;
            border-radius: 10px;
            background-color: #714;
            }

/*kaikille buttoneille*/
button {
            float: center;
            margin: 5px 2px;
            padding: 5px 5px;
            color: #222;
            background-color: #777;
            border: 5px solid #111;
            border-radius: 20px;
            font-size: 1.3em;
            font-weight: bold;
            }

button:hover {
  background-color: #722;
  -wekit-transition: all 1s;
  -moz-transition: all 1s;}
/*omat sliderit ja napit*/

.eq {
  -moz-transform: rotate(-90deg);
  -ms-transform: rotate(-90deg);
  -webkit-transform: rotate(-90deg);
  transform: rotate(-90deg);
  -moz-border-radius: 8px;
  -webkit-border-radius: 8px;
  border-radius: 8px;
```

```
  -moz-box-shadow: -1px 0 1px rgba(0, 0, 0, 0.25) inset, 0 -3px 0 rgba(0, 0,
0, 0.1) inset;
  -webkit-box-shadow: -1px 0 1px rgba(0, 0, 0, 0.25) inset, 0 -3px 0 rgba(0,
0, 0, 0.1) inset;
  box-shadow: -1px 0 1px rgba(0, 0, 0, 0.25) inset, 0 -3px 0 rgba(0, 0, 0,
0.1) inset;
  float: center;
  position: relative;
  left: 10px;
  width: 100px;
  height: 15px;
  padding: 0;
  margin: 10px 0 0 -75px;
  background: #111111;
  color: #AAC5DB;
  border-color: #AAAAAA;
  -webkit-appearance: none !important;
  outline: none;
}

.eq::-webkit-slider-thumb {
  background: -webkit-gradient(linear, 0% 50%, 100% 50%, color-stop(0%,
#eec900), color-stop(100%, #fafafa));
  background: -moz-linear-gradient(left, #eec900, #fafafa);
  background: -webkit-linear-gradient(left, #eec900, #fafafa);
  background: linear-gradient(to right, #eec900, #fafafa);
  -moz-border-radius: 64px;
  -webkit-border-radius: 64px;
  border-radius: 64px;
  -moz-box-shadow: 1px 0 2px rgba(0, 0, 0, 0.1), 1px 0 1px rgba(0, 0, 0, 0.5)
inset, -6.66667px 0 5px rgba(0, 0, 0, 0.15);
  -webkit-box-shadow: 1px 0 2px rgba(0, 0, 0, 0.1), 1px 0 1px rgba(0, 0, 0,
0.5) inset, -6.66667px 0 5px rgba(0, 0, 0, 0.15);
  box-shadow: 1px 0 2px rgba(0, 0, 0, 0.1), 1px 0 1px rgba(0, 0, 0, 0.5)
inset, -6.66667px 0 5px rgba(0, 0, 0, 0.15);
  position: relative;
  -webkit-appearance: none !important;
  width: 45px;
  height: 90px;
  border: 5px solid rgba(0, 0, 0, 0.75);
  cursor: pointer;
}
.eq::-webkit-slider-thumb:hover {
background: -webkit-gradient(linear, 0% 50%, 100% 50%, color-stop(0%,
#eec900), color-stop(100%, #22222));
  background: -moz-linear-gradient(left, #eec900, #222222);
  background: -webkit-linear-gradient(left, #eec900, #222222);
  background: linear-gradient(to right, #eec900, #222222);
  }
  </style>
</head>
<body>
            <!--Generators Start-->
  <div>
    <!--Generator 1 Start-->
    <center>
      <div class="osc1">
        <!--Oscillator 1 Start-->
          <div id="ef" class="group1" tabindex="1">
```

```
          <div id="result"></div>
          <button onclick="soundGen1.saveOsc1()">Save Osc1</button>
          <button onclick="soundGen1.loadOsc1State()">Load Osc1</button>
          <p><canvas id="Osc2" style="background: lighgreen;" width="200"
height="10"></canvas></p>
          <h2>Wave Types Osc1</h2>
          <button onclick="soundGen1.changeType('sine')"
id="but_sine1">Sine</button>
                                                                <button
onclick="soundGen1.changeType('square')" id="but_square1">Square</button>
                                                                <button
onclick="soundGen1.changeType('triangle')" id="but_triangle1">Trian</button>
                                                                <button
onclick="soundGen1.changeType('sawtooth')" id="but_saw1">Saw</button>
                                                                <button
onclick="soundGen1.toggle()">Play/Pause</button>
                                                                <h2>Osc Scale</h2>
          <!--<button onclick="soundGen1.changeDetune('-1200')"
id="but_scale128_Osc1">128</button>
                                                                <button
onclick="soundGen1.changeDetune('-900')" id="but_scale64_Osc1">64</button>
          <button onclick="soundGen1.changeDetune('-600')"
id="but_scale32_Osc1">32</button>
                                                                <button
onclick="soundGen1.changeDetune('-300')" id="but_scale16_Osc1">16</button>
                                                                <button
onclick="soundGen1.changeDetune('0')" id="but_scale8_Osc1">8</button>
                                                                <button
onclick="soundGen1.changeDetune('300')" id="but_scale4_Osc1">4</button>
          <button onclick="soundGen1.changeDetune('600')"
id="but_scale2_Osc1">2</button>
          <button onclick="soundGen1.changeDetune('900')"
id="but_scale1_Osc1">1</button>
                                                                <button
onclick="soundGen1.changeDetune('1200')" id="but_scale0_Osc1">0</button>
          <center>
                                                                          <div
class="slider1">
               LIMITE_1<input id="limiter1" class="eq" type="range"
name="limit1" min="-50" max="0" step="5" value="0"
               onchange="soundGen1.changeLimiterTreshold(this.value);">
               VOLUME_1<input id="gain1" class="eq" type="range" name="vol1"
min="0" max="1" step="0.1" value="0.5"
               onchange="soundGen1.changeVolume(this);">
               FREQUE_1<input id="frequency1" class="eq" type="range"
name="freq1" min="66" max="5280" step="1" value="132"
               onchange="soundGen1.changeFrequency(this.value);">
                                                                          </div>
          </center>-->
          <button onclick="soundGen1.changeFrequency('27')"
id="but_scale128_Osc1">32</button>
                                                                <button
onclick="soundGen1.changeFrequency('108')" id="but_scale64_Osc1">16</button>
          <button onclick="soundGen1.changeFrequency('432')"
id="but_scale32_Osc1">8</button>
                                                                <button
onclick="soundGen1.changeFrequency('1728')" id="but_scale16_Osc1">2</button>
                                                                <!--<button
onclick="soundGen1.changeFrequency('6912')" id="but_scale8_Osc1">1</button>
```

```
                                                        <button
onclick="soundGen1.changeFrequency('300')" id="but_scale4_Osc1">4</button>
          <button onclick="soundGen1.changeFrequency('600')"
id="but_scale2_Osc1">2</button>
          <button onclick="soundGen1.changeFrequency('900')"
id="but_scale1_Osc1">1</button>
                                                        <button
onclick="soundGen1.changeFrequency('1200')" id="but_scale0_Osc1">0</button>
          -->
          <center>
                                                                    <div
class="slider1">
              LIMITE_1<input id="limiter1" class="eq" type="range"
name="limit1" min="-50" max="0" step="5" value="0"
              onchange="soundGen1.changeLimiterTreshold(this.value);">
              VOLUME_1<input id="gain1" class="eq" type="range" name="vol1"
min="0" max="1" step="0.1" value="0.5"
              onchange="soundGen1.changeVolume(this);">
              FREQUE_1<input id="frequency1" class="eq" type="range"
name="freq1" min="-1200" max="1200" step="8" value="0"
              onchange="soundGen1.changeDetune(this.value);">
                                                                    </div>
          </center>
                                                </div>
                                                <!--Oscillator 1 End-->
                                                <!--VCO LFO 1 Start-->
                                                <div id="ef" class="group1">
                                                            <h2>VCO LFO 1</h2>
                                                            <button
onclick="soundGen1.changeLFOType('sine')" id="but_LFOsine1">Sine</button>
                                                        <button
onclick="soundGen1.changeLFOType('square')"
id="but_LFOsquare1">Square</button>
                                                        <button
onclick="soundGen1.changeLFOType('triangle')"
id="but_LFOtriangle1">Trian</button>
                                                        <button
onclick="soundGen1.changeLFOType('sawtooth')" id="but_LFOsaw1">Saw</button>
                                                        <center>
                                                                    <div
class="slider1">

            FREQUE_1 <input id="LFOfrequency1" class="eq" type="range"
name="LFOfreq1" min="0" max="15" step="1" value="0"

            onchange="soundGen1.changeLFOFrequency(this.value);">

            GAIN___1 <input id="LFOgain1" class="eq" type="range"
name="volLFO1" min="0" max="100" step="1" value="0"

            onchange="soundGen1.changeLFOGain(this.value);">
                                                                    </div>
                                                        </center>
                                                </div>
                                                <!--VCO LFO 1 End-->
        <!--VCA LFO 1 Start-->
                                                <div id="ef" class="group1">
                                                            <h2>VCA LFO 1</h2>
```

```
                                                        <button
onclick="soundGen1.changeVcaLFOType('sine')"
id="but_VcaLFOsine1">Sine</button>
                                                        <button
onclick="soundGen1.changeVcaLFOType('square')"
id="but_VcaLFOsquare1">Square</button>
                                                        <button
onclick="soundGen1.changeVcaLFOType('triangle')"
id="but_VcaLFOtriangle1">Trian</button>
                                                        <button
onclick="soundGen1.changeVcaLFOType('sawtooth')"
id="but_VcaLFOsaw1">Saw</button>
                                                        <center>
                                                                    <div
class="slider1">

            RATE___1 <input id="VcaLFOfrequency1" class="eq" type="range"
name="VcaLFOfreq1" min="0" max="15" step="1" value="0"

            onchange="soundGen1.changeVcaLFOFrequency(this.value);">

            GAIN___1 <input id="VcaLFOgain1" class="eq" type="range"
name="VcavolLFO1" min="0" max="1" step="0.1" value="0.5"

            onchange="soundGen1.changeVcaLFOGain(this.value);">
              GRATE__1 <input id="VcaLFOrate1" class="eq" type="range"
name="VcarateLFO1" min="0" max="1" step="0.1" value="0.5"

            onchange="soundGen1.changeVcaLFORate(this.value);">
                                                                    </div>
                                                        </center>
                                                </div>
                                                <!--VCA LFO 1 End-->
        <!--LPF 1 Start-->
        <div id="ef" class="group1">
                                                        <h2>LO/HI-
PassFilter</h2>
                                                        <button
onclick="soundGen1.changeFilterType('lowpass')" id="but_lpf1">LPF</button>
                                                        <button
onclick="soundGen1.changeFilterType('highpass')" id="but_hpf1">HPF</button>
          <!--<button onclick="soundGen1.toggleFilter()"
id="lpf1">ON/OFF</button>-->
          <center>
                                                                    <div
class="slider1">
              CUTOFF_1<input id="lowPass1" class="eq" type="range"
name="cutoff_1" min="60" max="2000" step="20" value="2000"
              onchange="soundGen1.changeFilterFrequency(this);">
              Q_VALU_1<input id="qvalue1" class="eq" type="range"
name="qvalue_1" min="0" max="20" step="0.1" value="0"
              onchange="soundGen1.changeFilterQ(this);">
                                                                    </div>
                                                        </center>
                                                </div>
        <!--LPF 1 End-->
        <!--Delay / FeedBack 1 Start-->
        <div id="ef" class="group1">
```

```
                                                        <h2>Delay /
FeedBack</h2>
                                                        <center>
                                                                <div
class="slider1">
              DELAY__1<input id="delay1" class="eq" type="range"
name="delay_1" min="0" max="1" step="0.05" value="0"
              onchange="soundGen1.changeDelay(this.value);">
              FEEDBA_1<input id="feedBack1" class="eq" type="range"
name="feedback_1" min="0" max="0.95" step="0.05" value="0"
              onchange="soundGen1.changeFeedBack(this);">
                                                                </div>
                                                        </center>
                                                </div>
        <!--Delay / FeedBack 1 End-->
                                                <!--Noise 1 Start-->
                                                <div id="ef" class="group3">
          <button onclick="noiseGen1.saveNoise1()">Save Noise1</button>
          <button onclick="noiseGen1.loadNoise1State()">Load Noise1</button>
                                                        <h2>NOISE1</h2>
          <p><canvas id="Osc1" style="background: lighgreen;" width="200"
height="10"></canvas></p>
                                                        <button
onclick="noiseGen1.toggleNoise()" id="noise">Play/Pause</button>
          <button onclick="noiseGen1.changeNoiseT('101')"
id="but_scale32_Osc1">Infinit</button>
                                                        <button
onclick="noiseGen1.changeNoiseT('1')" id="but_scale16_Osc1">Chirp</button>
          <center>
                                                                <div
class="slider1">
              LIMITE_1<input id="Noiselimiter1" class="eq" type="range"
name="Noiselimit1" min="-50" max="0" step="5" value="0"
              onchange="noiseGen1.changeNoiseLimiterTreshold(this.value);">

            VOLUME_1 <input id="gainN1" class="eq" type="range" name="volN1"
min="0" max="1" step="0.1" value="0.1"

            oninput="noiseGen1.changeNoiseVolume(this);">

            RATE___1 <input id="noiseF1" class="eq" type="range"
name="freqN1" min="-1200" max="1200" step="10" value="0"

            oninput="noiseGen1.changeNoiseF(this.value);">

            TIME___1 <input id="noiseT1" class="eq" type="range"
name="timeN1" min="1" max="101" step="5" value="51"

            oninput="noiseGen1.changeNoiseT(this.value);">
                                                                </div>
                                                        </center>
                                                </div>
                                                <!--Noise 1 End-->
        <!--NOISE LPF 1 Start-->
        <div id="ef" class="group3">
                                                        <h2>Noise1 LO/HI-
PassFilter</h2>
```

```
                                                        <button
onclick="noiseGen1.changeNoiseFilterType('lowpass')"
id="but_lpf_nois1">LPF</button>
                                                        <button
onclick="noiseGen1.changeNoiseFilterType('highpass')"
id="but_hpf_nois1">HPF</button>
          <center>
                                                                    <div
class="slider1">
              CUTOFF_1 <input id="lowPass_nois1" class="eq" type="range"
name="cutoff1" min="60" max="10000" step="20" value="10000"
              onchange="noiseGen1.changeNoiseFilterFrequency(this);">
              Q_VAL__1 <input id="q_val_nois1" class="eq" type="range"
name="q_value1" min="0" max="20" step="0.1" value="0"
              onchange="noiseGen1.changeNoiseFilterQ(this);">
                                                                    </div>
                                                        </center>
                                                </div>
        <!--NOISE LPF 1 End-->
        <!--NoiseVCO LFO 1 Start-->
                                                <div id="ef" class="group3">
                                                        <h2>Noise1
VCO</h2>
                                                        <button
onclick="noiseGen1.changeNoiseVcaLFOType('sine')"
id="but_NoiseVcaLFOsine1">Sine</button>
                                                        <button
onclick="noiseGen1.changeNoiseVcaLFOType('square')"
id="but_NoiseVcaLFOsquare1">Square</button>
                                                        <button
onclick="noiseGen1.changeNoiseVcaLFOType('triangle')"
id="but_NoiseVcaLFOtriangle1">Trian</button>
                                                        <button
onclick="noiseGen1.changeNoiseVcaLFOType('sawtooth')"
id="but_NoiseVcaLFOsaw1">Saw</button>
                                                        <center>
                                                                    <div
class="slider1">

            RATE___1 <input id="NoiseVcaLFOfrequency1" class="eq"
type="range" name="NoiseVcaLFOfreq1" min="0" max="15" step="1" value="0"

            onchange="noiseGen1.changeNoiseVcaLFOFrequency(this.value);">

            GAIN___1 <input id="NoiseVcaLFOgain1" class="eq" type="range"
name="NoiseVcavolLFO1" min="0" max="2" step="0.1" value="0.5"

            onchange="noiseGen1.changeNoiseVcaLFOGain(this.value);">
              GRATE__1 <input id="NoiseVcaLFOrate1" class="eq" type="range"
name="NoiseVcarateLFO1" min="0" max="2" step="0.1" value="0.5"

            onchange="noiseGen1.changeNoiseVcaLFORate(this.value);">
                                                                    </div>
                                                        </center>
                                                </div>
                                                <!--Noise VCO LFO 1 End-->
        <!--Noise Delay / FeedBack 1 Start-->
        <div id="ef" class="group3">
```

```
                                                        <h2>Delay /
FeedBack</h2>
          <center>
                                                                  <div
class="slider1">
              DELAY__1<input id="Noisedelay1" class="eq" type="range"
name="Noisedelay" min="0" max="1" step="0.05" value="0"
              onchange="noiseGen1.changeNoiseDelay(this.value);">
              FEEDBA_1<input id="NoisefeedBack1" class="eq" type="range"
name="Noisefeedback" min="0" max="0.95" step="0.05" value="0"
              onchange="noiseGen1.changeNoiseFeedBack(this);">
                                                                  </div>
                                                        </center>
                                                  </div>
        <!--Noise Delay / FeedBack 1 End-->
      </div>
                        </center>
                        <!--Generator 1 End-->
    <!--Generator 2 Start-->
                        <center>
                                    <div class="osc2">
                                                  <!--Oscillator 2 Start-->
                                                  <div id="ef" class="group2">
          <button onclick="soundGen2.saveOsc2()">Save Osc2</button>
          <button onclick="soundGen2.loadOsc2State()">Load Osc2</button>
                                                        <h2>Wave Types
Osc2</h2>
                                                        <button
onclick="soundGen2.changeType('sine')" id="but_sine2">Sine</button>
                                                        <button
onclick="soundGen2.changeType('square')" id="but_square2">Square</button>
                                                        <button
onclick="soundGen2.changeType('triangle')" id="but_triangle2">Trian</button>
                                                        <button
onclick="soundGen2.changeType('sawtooth')" id="but_saw2">Saw</button>
                                                        <button
onclick="soundGen2.toggle()">Play/Pause</button>
                                                        <!--<button
onclick="noiseGen2.toggleNoise()" id="noise2">Noise</button>-->
                                                        <center>
                                                                  <div
class="slider2">

            VOLUME_2<input id="gain2" class="eq" type="range" name="vol2"
min="0" max="1" step="0.1" value="0.5"

            onchange="soundGen2.changeVolume(this);">

            DETUNE_2<input id="detune2" class="eq" type="range"
name="detune2" min="-1200" max="1200" step="8" value="0"

            onchange="soundGen2.changeDetune(this.value);">

            FREQUE_2<input id="frequency2" class="eq" type="range"
name="freq2" min="96" max="1584" step="96" value="528"

            onchange="soundGen2.changeFrequency(this.value);">
                                                                  </div>
                                                        </center>
```

```
                                                </div>
                                                <!--Oscillator 2 End-->
                                                <!--LFO 2 Start-->
                                                <div id="ef" class="group2">
                                                            <h2>LFO</h2>
                                                            <button
onclick="soundGen2.changeLFOType('sine')" id="but_LFOsine2">Sine</button>
                                                            <button
onclick="soundGen2.changeLFOType('square')"
id="but_LFOsquare2">Square</button>
                                                            <button
onclick="soundGen2.changeLFOType('triangle')"
id="but_LFOtriangle2">Trian</button>
                                                            <button
onclick="soundGen2.changeLFOType('sawtooth')" id="but_LFOsaw2">Saw</button>
                                                            <center>
                                                                        <div
class="slider2">

            LFO____2<input id="LFOfrequency2" class="eq" type="range"
name="LFOfreq2" min="0" max="5" step="0.1" value="0"

            onchange="soundGen2.changeLFOFrequency(this.value);">

            GIAN___2<input id="LFOgain2" class="eq" type="range"
name="volLFO2" min="0" max="100" step="1" value="0"

            onchange="soundGen2.changeLFOGain(this.value);">
                                                                        </div>
                                                            </center>
                                                </div>
                                                <!--LFO 2 End-->
        <!--LPF/HPF 2 Start-->
        <div id="ef" class="group2">
                                                            <h2>LO/HI-
PassFilter</h2>
                                                            <button
onclick="soundGen2.changeFilterType('lowpass')" id="but_lpf2">LPF</button>
                                                            <button
onclick="soundGen2.changeFilterType('highpass')" id="but_hpf2">HPF</button>
          <center>
                                                                        <div
class="slider1">
              CUTOFF_2<input id="lowHiPass2" class="eq" type="range"
name="cutoff2" min="60" max="2000" step="20" value="2000"
              onchange="soundGen2.changeFilterFrequency(this);">
              Q_VALU_2<input id="q_value_2" class="eq" type="range"
name="q_value2" min="0" max="20" step="0.1" value="0"
              onchange="soundGen2.changeFilterQ(this);">
                                                                        </div>
                                                            </center>
                                                </div>
        <!--LPF/HPF 2 End-->
                                                <!--Noise 2 Start-->
                                                <div id="ef" class="group2">
          <button onclick="noiseGen2.saveNoise2()">Save Noise2</button>
          <button onclick="noiseGen2.loadNoise2State()">Load Noise2</button>
                                                            <h2>NOISE2</h2>
```

```
                                                                <button
onclick="noiseGen2.toggleNoise()" id="noise">Play/Pause</button>
                                                                <center>
                                                                        <div
class="slider2">

            <!--VOLUME_2<input id="gainN2" class="eq" type="range"
name="volN2" min="0" max="1" step="0.1" value="0.1"

            onchange="noiseGen2.changeNoiseVolume(this);">-->
              VOLUME:2<input id="gainN2" class="eq" type="range" name="volN2"
min="0" max="1" step="0.1" value="0.1"

            onchange="noiseGen2.changeNoiseVolume(this);">

            RATE___2<input id="noiseF2" class="eq" type="range" name="freqN2"
min="1" max="100" step="1" value="100"

            onchange="noiseGen2.changeNoiseF(this.value);">

            TIME___2<input id="noiseT2" class="eq" type="range" name="timeN2"
min="1" max="100" step="1" value="70"

            onchange="noiseGen2.changeNoiseT(this.value);">
                                                                        </div>
                                                                <center>
                                                        </div>
                                                        <!--Noise 2 End-->
        <!--Noise 2 LPF/HPF start-->
        <div id="ef" class="group2">
                                                                <h2>Noise2 LO/HI-
PassFilter</h2>
                                                                <button
onclick="noiseGen2.changeNoiseFilterType('lowpass')"
id="but_lpf_nois1">LPF</button>
                                                                <button
onclick="noiseGen2.changeNoiseFilterType('highpass')"
id="but_hpf_nois1">HPF</button>
          <center>
                                                                        <div
class="slider2">
              CUTOFF_2 <input id="lowHiPass_nois2" class="eq" type="range"
name="cutoff2" min="20" max="10000" step="10" value="1000"
              onchange="noiseGen2.changeNoiseFilterFrequency(this);">
              Q_VAL__2<input id="q_val_nois2" class="eq" type="range"
name="q_value2" min="0" max="20" step="1" value="0"
              onchange="noiseGen2.changeNoiseFilterQ(this);">
                                                                        </div>
                                                                <center>
        <!--Noise 2 LPF/HPF end-->
                                        </center>
                        </div>
                        <!--Generator 2 End-->
    <!--Hiirellä leikkimis kentätä alkaa-->
    <div id="ef" class="group2">
      <!--<h2>MousePad</h2>
      <p>
```

```
      <canvas id="sine" style="background: #666666;" width="175" height="150"
draggable="false" onmousedown="playWave(event)" onmouseup="stopWave(event)"
onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="square" style="background: #CC3333;" width="175"
height="150" draggable="false" onmousedown="playWave(event)"
onmouseup="stopWave(event)" onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="triangle" style="background: #33CC33;" width="175"
height="150" draggable="false" onmousedown="playWave(event)"
onmouseup="stopWave(event)" onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="sawtooth" style="background: #3333CC;" width="175"
height="150" draggable="false" onmousedown="playWave(event)"
onmouseup="stopWave(event)" onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="noise" style="background: #EEC900;" width="175"
height="150" draggable="false" onmousedown="NoisePlay(event)"
onmouseup="stopNoise(event)" onmousemove="changeNoiseVol(event)"></canvas>
      </p>-->
      <h2>KaossPads<p><canvas id="sine" style="background: #111111;"
width="75" height="75" draggable="false" onmousedown="stopWave(event),
stopNoise(event)"></canvas>
      </h2>
      <p>
      <canvas id="sine" style="background: #666666;" width="175" height="150"
draggable="false" onmousedown="playWave(event)" onmouseup="stopWave(event)"
onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="square" style="background: #CC3333;" width="175"
height="150" draggable="false" onmousedown="playWave(event)"
onmouseup="stopWave(event)" onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="triangle" style="background: #33CC33;" width="175"
height="150" draggable="false" onmousedown="playWave(event)"
onmouseup="stopWave(event)" onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="sawtooth" style="background: #3333CC;" width="175"
height="150" draggable="false" onmousedown="playWave(event)"
onmouseup="stopWave(event)" onmousemove="changeFreqVol(event)"></canvas>
      <canvas id="noise" style="background: #EEC900;" width="175"
height="150" draggable="false" onmousedown="NoisePlay(event)"
onmouseup="stopNoise(event)" onmousemove="changeNoiseVol(event)"></canvas>
      </p>
    </div>
    <!--Hiirellä leikkimis kentät loppuu-->
           </div>
           <!--Generators End-->
<script>
//http://www.w3schools.com/js/js_object_prototypes.asp

 //Luodaan ja tarkistetaan Web Audio API tuki
//context = new AudioContext(); //riittäisi hyvin viimeisimmälle Chromelle
context = new ( AudioContext || webkitAudioContext ||
    function() { throw "Browser does not support Web Audio API"; }
    )();

/*Canvas Aaltonäyttö Boris Smus:n mukaan kirjasta Web Audio API*/
window.requestAnimFrame = (function(){
return  window.requestAnimationFrame       ||
  window.webkitRequestAnimationFrame ||
  window.mozRequestAnimationFrame    ||
  window.oRequestAnimationFrame      ||
  window.msRequestAnimationFrame     ||
  function( callback ){
  window.setTimeout(callback, 1000 / 60);
```

```
};
})();
/***************************/
/***************************/
//Normaali Oskillaattori
//OscSamplestä modattu taas 14.10.2015, nyt kokonaan omilla hässäköillä
//SOUNDGENERATOR OBJEKTI tai SoundGenerator objektin muodostin funktio
//prototyyppi ketjun pääjehu

function SoundGenerator() {
//this.SoundGenerator = function () {
  this.isOscPlaying = false;
  this.isNoise1Playing = false;

  //Aaltonäyttö
  //this.canvas1 = document.getElementById('Osc1');
            this.canvas2 = document.getElementById('Osc2');
            this.WIDTH = 300;
            this.HEIGHT = 100;

  //molemmat Oscil generaattorit aloittavat näillä arvoilla, default
asetukset
  this.currentOscilType = 'sine';
  this.currentOscilGain = 0.5;
  this.currentOscilScale = 432;
  this.currentOscilFreq = 0;
  this.currentOscilLimit = 0;
  this.currentOscilVcoLFOType = 'sine';
  this.currentOscilVcoLFOFreq = 0;
  this.currentOscilVcoLFOGain = 0;
  this.currentOscilLoHiType = 'lowpass';
  this.currentOscilLoHiCutOff = 5000;
  this.currentOscilLoHiQ = 0;
  this.currentOscilVcaLFOType = 'sine';
  this.currentOscilVcaLFOFreq = 0;
  this.currentOscilVcaLFOGain = 0.5;  //modulatorVCA
  this.currentOscilVcaLFORate = 0.5; //modGainVCA
  this.currentOscilDelayTime = 0;
  this.currentOscilDelayFeedback = 0;
  };

//context = new AudioContext();
//SOUNDGENERATOR OBJEKTI

//OscPlay funktio SoundGenerator funktio objektin prototyypiksi
SoundGenerator.prototype.OscPlay = function() {

  // Oscillator VCO
  this.oscillator = context.createOscillator(); //osc01Vco
  this.oscillator.type = this.currentOscilType; //osc01VcoType
  this.oscillator.frequency.value = this.currentOscilScale; //osc01VcoDetune
(Osc1 scale)
  this.oscillator.detune.value = this.currentOscilFreq;

  //VCO LFO eli tällä ohjataan Oscin taajuus väpätystä
  this.oscillatorLFO = context.createOscillator(); //osc02VcoLFO
  this.oscillatorLFO.frequency.value = this.currentOscilVcoLFOFreq; //0-15;
  this.oscillatorLFO.type = this.currentOscilVcoLFOType; //'sine';
  //VCO LFO gain
```

```
  this.oscillatorLFOGain = context.createGain(); //modain01Vc0
  this.oscillatorLFOGain.gain.value = this.currentOscilVcoLFOGain; //0-100;

  //VCA LFO eli tällä ohjataan Gain väpätystä
  this.oscillatorVcaLFO = context.createOscillator(); //osc03Vca
  this.oscillatorVcaLFO.frequency.value = this.currentOscilVcaLFOFreq; //0-
15;
  this.oscillatorVcaLFO.type = this.currentOscilVcaLFOType; //'sine'

  //VCA modulator gain
  this.modulatorVCA = context.createGain();  // modulatorGain03Vca Amplitude
modulator
            this.modulatorVCA.gain.value = this.currentOscilVcaLFOGain; //ns.
Syvyys (Volume)

  //VCA
            this.modGainVCA = context.createGain();
            this.modGainVCA.gain.value = this.currentOscilVcaLFORate;
            this.oscillatorVcaLFO.connect(this.modGainVCA);
            this.modGainVCA.connect(this.modulatorVCA.gain);
  //VCA

  // Create the LO/HI-paas filter.
  this.filterNode = context.createBiquadFilter();
  //filter type
  this.filterNode.type = this.currentOscilLoHiType;
  this.filterNode.frequency.value = this.currentOscilLoHiCutOff;//5000;
//perus cutOff, jotta kuulu edes jotain ON tilanteessa
  this.filterNode.Q.value = this.currentOscilLoHiQ;
  console.log(this.filterNode.Q.value);

  //Delay and Feedack http://blog.chrislowis.co.uk/2014/07/23/dub-delay-web-
audio-api.html
  //Delay
  this.delayNode = context.createDelay();
  this.delayNode.delayTime.value = this.currentOscilDelayTime;//0;
  this.delayType = this.currentOscilDelayType;

  //FeedBack
  this.feedbackNode = context.createGain();
  this.feedbackNode.gain.value = this.currentOscilDelayFeedback;//0;

  //Create Analyser aaltonäyttö
  this.analyserNode = context.createAnalyser();

  //Osc gain eli Master Volume
  this.gainNode = context.createGain();
  this.gainNode.gain.value = this.currentOscilGain;

  //Limiter
  this.limiter = context.createDynamicsCompressor();
  this.limiter.threshold.value = this.currentOscilLimit;
  this.limiter.knee.value = 30.0;
  this.limiter.ratio.value = 12.0;
  this.limiter.attack.value = 0.005;
  this.limiter.release.value = 0.250;

  // Create the Masterfilter, jotta korvat (eläimet) ja PA:t säästyy.
  this.filterMasterNode = context.createBiquadFilter();
```

```
  //filter type bandpass ja leikataan esim 432Hz ala-ja yläpuoliset f 5dB /
tuplataajuss eli 22 Hz -10dB ja 360 KHz -10dB
  this.filterMasterNode.type = 'bandpass';
  this.filterMasterNode.frequency.value = 256;
  this.filterMasterNode.Q.value = 1;

  //Connect sources to their destinations.
  this.oscillatorLFO.connect(this.oscillatorLFOGain); //osc02Vco->gain01Vco
  this.oscillatorLFOGain.connect(this.oscillator.frequency); //gain01Vco-
>osc01freq
  this.oscillator.connect(this.modulatorVCA); //osc01Vco->ModulatorVcaGain03
  this.modulatorVCA.connect(this.gainNode); //ModulatorVcaGain03->gainNode
  this.gainNode.connect(this.filterNode); //gainNode04->FilterLoHi
  this.filterNode.connect(this.limiter); //FilterLoHi->Limiter
  this.filterNode.connect(this.delayNode); //FilterLoHi->Delay
  this.delayNode.connect(this.feedbackNode); //Delay->Feedback
  this.feedbackNode.connect(this.delayNode); //Feedback->Delay
  this.delayNode.connect(this.limiter); //Delay->Limiter
  //kaikki yhteen volumeen masterFilterin ja limitterin läpi
  this.limiter.connect(this.filterMasterNode); //Limiter->MasterFilter
  this.filterMasterNode.connect(context.destination); //MasterFilter-
>Destination(output)

  //Analyser aaltonäytölle
  this.filterMasterNode.connect(this.analyserNode); //MasterFilter->Analyser

  //Delay
  this.delayNode.connect(this.limiter);
  //oscillaattorit käyntiin
  this.oscillator[this.oscillator.start ? 'start' : 'noteOn'](0);
  this.oscillatorVcaLFO[this.oscillatorLFO.start ? 'start' : 'noteOn'](0);
  this.oscillatorLFO[this.oscillatorLFO.start ? 'start' : 'noteOn'](0);

  //Canvas aaltonäyttö kutsu
  requestAnimFrame(this.visualize.bind(this));

  //Talletus play/pause nappulaa painettaessa pause tilanne talteen
  this.currentOscilType = this.oscillator.type;
  this.currentOscilGain = this.gainNode.gain.value;
  this.currentOscilScale = this.oscillator.frequency.value;
  this.currentOscilFreq = this.oscillator.detune.value;
  this.currentOscilLimit = this.limiter.threshold.value;
  this.currentOscilVcoLFOType = this.oscillatorLFO.type;
  this.currentOscilVcoLFOFreq = this.oscillatorLFO.frequency.value;
  this.currentOscilVcoLFOGain = this.oscillatorLFOGain.gain.value;
  this.currentOscilLoHiType = this.filterNode.type;
  this.currentOscilLoHiCutOff = this.filterNode.frequency.value;
  this.currentOscilLoHiQ = this.filterNode.Q.value;
  this.currentOscilVcaLFOType = this.oscillatorVcaLFO.type;
  this.currentOscilVcaLFOFreq = this.oscillatorVcaLFO.frequency.value;
  this.currentOscilVcaLFOGain = this.modulatorVCA.gain.value;
  this.currentOscilVcaLFORate = this.modGainVCA.gain.value;
  this.currentOscilDelayType = this.delayType;
  this.currentOscilDelayTime = this.delayNode.delayTime.value;
  this.currentOscilDelayFeedback = this.feedbackNode.gain.value;
};

SoundGenerator.prototype.stop = function() {
  //tilanne talletus
```

```
  this.currentOscilType = this.oscillator.type;
  this.currentOscilGain = this.gainNode.gain.value;
  this.currentOscilScale = this.oscillator.frequency.value;
  this.currentOscilFreq = this.oscillator.detune.value;
  this.currentOscilLimit = this.limiter.threshold.value;
  this.currentOscilVcoLFOType = this.oscillatorLFO.type;
  this.currentOscilVcoLFOFreq = this.oscillatorLFO.frequency.value;
  this.currentOscilVcoLFOGain = this.oscillatorLFOGain.gain.value;
  this.currentOscilLoHiType = this.filterNode.type;
  this.currentOscilLoHiCutOff = this.filterNode.frequency.value;
  this.currentOscilLoHiQ = this.filterNode.Q.value;
  this.currentOscilVcaLFOType = this.oscillatorVcaLFO.type;
  this.currentOscilVcaLFOFreq = this.oscillatorVcaLFO.frequency.value;
  this.currentOscilVcaLFOGain = this.modulatorVCA.gain.value;
  this.currentOscilVcaLFORate = this.modGainVCA.gain.value;
  this.currentOscilDelayTime = this.delayNode.delayTime.value;
  this.currentOscilDelayFeedback = this.feedbackNode.gain.value;
  //tilanne talletus loppuu
  //stoppaus
  this.oscillator.stop(0);
  this.oscillator.disconnect(); // Disconnect selaimen garbage collectoria
varten. Huom! ei tarvi enää 24.10

};
//soundGen1 asetusten talletus local storageen
SoundGenerator.prototype.saveOsc1 = function() {
  if (typeof(Storage) !== "undefined") {
    //localStorage.setPath('/');
    localStorage.setItem("osc1Type", soundGen1.oscillator.type);
    localStorage.setItem("osc1Gain", soundGen1.gainNode.gain.value);
    localStorage.setItem("osc1Scale", soundGen1.oscillator.frequency.value);
    localStorage.setItem("osc1Freq", soundGen1.oscillator.detune.value);
    localStorage.setItem("osc1Limit", soundGen1.limiter.threshold.value);
    localStorage.setItem("osc1VcoLfoType", soundGen1.oscillatorLFO.type);
    localStorage.setItem("osc1VcoLfoFreq",
soundGen1.oscillatorLFO.frequency.value);
    localStorage.setItem("osc1VcoLfoGain",
soundGen1.oscillatorLFOGain.gain.value);
    localStorage.setItem("osc1LoHiType", soundGen1.filterNode.type);
    localStorage.setItem("osc1LoHiCutOff",
soundGen1.filterNode.frequency.value);
    localStorage.setItem("osc1LoHiQ", soundGen1.filterNode.Q.value);
    localStorage.setItem("osc1VcaLfoType", soundGen1.oscillatorVcaLFO.type);
    localStorage.setItem("osc1VcaLfoFreq",
soundGen1.oscillatorVcaLFO.frequency.value);
    localStorage.setItem("osc1VcaLfoGain",
soundGen1.modulatorVCA.gain.value);
    localStorage.setItem("osc1VcaLfoRate", soundGen1.modGainVCA.gain.value);
    localStorage.setItem("osc1DelayTime",
soundGen1.delayNode.delayTime.value);
    localStorage.setItem("osc1DelayFeedback",
soundGen1.feedbackNode.gain.value);
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
```

```
//soundGene2 talletus
SoundGenerator.prototype.saveOsc2 = function() {
  if (typeof(Storage) !== "undefined") {
    localStorage.setItem("osc2Type", soundGen2.oscillator.type);
    localStorage.setItem("osc2Gain", soundGen2.gainNode.gain.value);
    localStorage.setItem("osc2Freq", soundGen2.oscillator.detune.value);
    localStorage.setItem("osc2Scale", soundGen2.oscillator.frequency.value);
    localStorage.setItem("osc2VcoLfoType", soundGen2.oscillatorLFO.type);
    localStorage.setItem("osc2VcoLfoFreq",
soundGen2.oscillatorLFO.frequency.value);
    localStorage.setItem("osc2VcoLfoGain",
soundGen2.oscillatorLFOGain.gain.value);
    localStorage.setItem("osc2LoHiType", soundGen2.filterNode.type);
    localStorage.setItem("osc2LoHiCutOff",
soundGen2.filterNode.frequency.value);
    localStorage.setItem("osc2LoHiQ", soundGen2.filterNode.Q.value);
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
//soundGen1 asetusten lataus local storagesta
SoundGenerator.prototype.loadOsc1State = function() {
  if (typeof(Storage) !== "undefined") {
    this.currentOscilType = localStorage.getItem("osc1Type");
    this.currentOscilGain = localStorage.getItem("osc1Gain");
    this.currentOscilScale = localStorage.getItem("osc1Scale");
    this.currentOscilFreq = localStorage.getItem("osc1Freq");
    this.currentOscilLimit = localStorage.getItem("osc1Limit");
    this.currentOscilVcoLFOType = localStorage.getItem("osc1VcoLfoType");
    this.currentOscilVcoLFOFreq = localStorage.getItem("osc1VcoLfoFreq");
    this.currentOscilVcoLFOGain = localStorage.getItem("osc1VcoLfoGain");
    this.currentOscilLoHiType = localStorage.getItem("osc1LoHiType");
    this.currentOscilLoHiCutOff = localStorage.getItem("osc1LoHiCutOff");
    this.currentOscilLoHiQ = localStorage.getItem("osc1LoHiQ");
    this.currentOscilVcaLFOType = localStorage.getItem("osc1VcaLfoType");
    this.currentOscilVcaLFOFreq = localStorage.getItem("osc1VcaLfoFreq");
    this.currentOscilVcaLFOGain = localStorage.getItem("osc1VcaLfoGain");
    this.currentOscilVcaLFORate = localStorage.getItem("osc1VcaLfoRate");
    this.currentOscilDelayType = localStorage.getItem("osc1DelayType");
    this.currentOscilDelayTime = localStorage.getItem("osc1DelayTime");
    this.currentOscilDelayFeedback =
localStorage.getItem("osc1DelayFeedback");
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
//soundGen2 asetusten lataus
SoundGenerator.prototype.loadOsc2State = function() {
  if (typeof(Storage) !== "undefined") {
    this.currentOscilType = localStorage.getItem("osc2Type");
    this.currentOscilGain = localStorage.getItem("osc2Gain");
    this.currentOscilScale = localStorage.getItem("osc2Scale");
    this.currentOscilFreq = localStorage.getItem("osc2Freq");
```

```
    this.currentOscilVcoLFOType = localStorage.getItem("osc2VcoLfoType");
    this.currentOscilVcoLFOFreq = localStorage.getItem("osc2VcoLfoFreq");
    this.currentOscilVcoLFOGain = localStorage.getItem("osc2VcoLfoGain");
    this.currentOscilLoHiType = localStorage.getItem("osc2LoHiType");
    this.currentOscilLoHiCutOff = localStorage.getItem("osc2LoHiCutOff");
    this.currentOscilLoHiQ = localStorage.getItem("osc2LoHiQ");
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};

SoundGenerator.prototype.toggle = function() {
  (this.isOscPlaying ? this.stop() : this.OscPlay());
  this.isOscPlaying = !this.isOscPlaying;

};
//Voimakkuus säätö toimii 15.10.2015
SoundGenerator.prototype.changeVolume = function(element) {
  this.gainNode.gain.value = element.value;
  console.log(this.gainNode.gain.value); //true, testaus Chrome konsoli
control+shift+J
};
//taajuus toimii 15.10.2015
SoundGenerator.prototype.changeFrequency = function(val) {
  this.oscillator.frequency.value = val;
  console.log(this.oscillator.frequency.value); //testaus
};
//detune toimii 15.10.2015
SoundGenerator.prototype.changeDetune = function(val) {
  this.oscillator.detune.value = val;
};
//Aaltomuodot toimii 15.10.2015
SoundGenerator.prototype.changeType = function(type) {
  this.oscillator.type = type;
  console.log(type); //testaus konsoli crtl+shift+j
};

//Limiter
SoundGenerator.prototype.changeLimiterTreshold = function(val) {
  this.limiter.threshold.value = val;
};

//LFO Low Frequency Oscillator eli moduloijan osuus
//VCO LFO tyypit
SoundGenerator.prototype.changeLFOType = function(type) {
  this.oscillatorLFO.type = type;
};

//VCO LFO Voimakkuus säätö
SoundGenerator.prototype.changeLFOGain = function(val) {
  this.oscillatorLFOGain.gain.value = val;
};
//VCO LFO taajuus
SoundGenerator.prototype.changeLFOFrequency = function(val) {
  this.oscillatorLFO.frequency.value = val;
};
```

```
//VCO

//VCA
//VCA LFO tyyppi
SoundGenerator.prototype.changeVcaLFOType = function(type) {
  this.oscillatorVcaLFO.type = type;
};
//VCA LFO Voimakkuus säätö, säädetään sisää tulevan ja modatun voluumi suhde
SoundGenerator.prototype.changeVcaLFOGain = function(val) {
  this.modulatorVCA.gain.value = val;
};
// VCA LFO grate
SoundGenerator.prototype.changeVcaLFORate = function(val) {
  this.modGainVCA.gain.value = val;
};
//VCA LFO taajuus
SoundGenerator.prototype.changeVcaLFOFrequency = function(val) {
  this.oscillatorVcaLFO.frequency.value = val;
};
//VCA
//LFO Low Frequency Oscillator eli moduloijan osuus

//BiQuadFilter ja Q-arvo, LO/HI-pass filters
//Filter CutOff
SoundGenerator.prototype.changeFilterFrequency = function(element) {
  this.filterNode.frequency.value = element.value; //* 1000; // kerroin 100
20Hz- 5kHz
};
//Filter Q
SoundGenerator.prototype.changeFilterQ = function(element) {
  this.filterNode.Q.value = element.value; //* 10;  //Q-arvo 0 - 20
  console.log(this.filterNode.Q.value);
};
//Filter type
SoundGenerator.prototype.changeFilterType = function(type) {
  this.filterNode.type = type;
};
//BiQuadFilter ja Q-arvo

//Delay ja FeedBack
SoundGenerator.prototype.changeDelay = function(val) {
  this.delayNode.delayTime.value = val;
};
SoundGenerator.prototype.changeFeedBack = function(element) {
  this.feedbackNode.gain.value = element.value;
};
//Delay ja FeedBack

//Canvas http://chimera.labs.oreilly.com/books/1234000001552/ch05.html#s05_2
SoundGenerator.prototype.visualize = function() {
  this.canvas2.width = this.WIDTH;
  this.canvas2.height = this.HEIGHT;
  var drawContext = this.canvas2.getContext('2d');

  //Aika/Gain
  var times = new Uint8Array(this.analyserNode.frequencyBinCount);
  this.analyserNode.getByteTimeDomainData(times);
  for (var i = 0; i < times.length; i++) {
    var value = times[i];
```

```
    var percent = value / 256;
    var height = this.HEIGHT * percent;
    var offset = this.HEIGHT - height - 1;
    var barWidth = this.WIDTH/times.length;
    drawContext.fillStyle = 'red';
    drawContext.fillRect(i * barWidth, offset, 1, 1);
  }

  //Taajuus/Aika näyttö
  var freqDomain = new Uint8Array(this.analyserNode.frequencyBinCount);
this.analyserNode.getByteFrequencyData(freqDomain);
for (var i = 0; i < this.analyserNode.frequencyBinCount; i++) {
  var value = freqDomain[i];
  var percent = value / 256;
  var height = this.HEIGHT * percent;
  var offset = this.HEIGHT - height - 1;
  var barWidth = this.WIDTH/this.analyserNode.frequencyBinCount;
  var hue = i/this.analyserNode.frequencyBinCount * 360;
  drawContext.fillStyle = 'hsl(' + hue + ', 100%, 50%)';
  drawContext.fillRect(i * barWidth, offset, barWidth, height);
}
  requestAnimFrame(this.visualize.bind(this));
};
/*Canvas Aaltonäyttö*/
//Normi osc
/***************************/
/***************************/


/*********************************/
/*********************************/
//NOISE OBJEKTI
function NoiseGenerator() {
            this.isNoisePlaying = false;
            this.noiseFrequ = 100;

  //Aaltonäyttö
  this.canvas1 = document.getElementById('Osc1');
            this.canvas2 = document.getElementById('Osc2');
            this.WIDTH = 300;
            this.HEIGHT = 100;

  //molemmat Noise generaattorit aloittavat näillä arvoilla
  this.currentNoiseGain = 0.1;
  this.currentNoiseTime = 70;
  this.currentNoiseFreq = 0;
  this.currentNoiseLimit = 0;
  this.currentNoiseLoHiType = 'lowpass';
  this.currentNoiseLoHiCutOff = 10000;
  this.currentNoiseLoHiQ = 0;
  this.currentNoiseVcaLFOType = 'triangle';
  this.currentNoiseVcaLFOFreq = 0;
  this.currentNoiseVcaLFOGain = 0.5;  //modulatorVCA
  this.currentNoiseVcaLFORate = 0.5; //modGainVCA
  this.currentNoiseDelayType = 'norm';
  this.currentNoiseDelayTime = 0;
  this.currentNoiseDelayFeedback = 0;
}
```

```
NoiseGenerator.prototype.NoisePlay = function() {

  // Create nodes.
  this.noiseFreq = this.noiseFrequ;
            this.noiseTime = this.currentNoiseTime;
  this.buffer = context.createBuffer(1, this.noiseFreq*1000,
context.sampleRate);
  //this.noiseSourceBuffer = context.createBufferSource(); //Noise
oscillaattori
            //this.noiseSourceBuffer.buffer = this.buffer;
  this.data = this.buffer.getChannelData(0); //mono ääni
            //taajuus eli noiseFreq eli suurempi luku -> harvempi kohaus,
slider 1-100
                        //for lauseella kesto eli noiseTime eli suurempi luku
-> pitempi kohaus, slider 1-100
                        for (var i = 0; i < this.noiseTime*1000; i++) {
                                    this.data[i] = Math.random();
                                    }
  this.noiseSourceBuffer = context.createBufferSource(); //Noise
oscillaattori
            this.noiseSourceBuffer.buffer = this.buffer;
  this.noiseSourceBuffer.detune.value = this.currentNoiseFreq;
  this.noiseSourceBuffer.loop = true;
  //Noise volume säätö ja sille lähtö arvo
            this.gainNodeNoise = context.createGain();
            this.gainNodeNoise.gain.value = this.currentNoiseGain;

  //VcaLFO NOISE eli tällä ohjataan Gain väpätystä
  this.noiseVcaLFO = context.createOscillator();
  this.noiseVcaLFO.frequency.value = this.currentNoiseVcaLFOFreq;
  this.noiseVcaLFO.type = this.currentNoiseVcaLFOType;

  this.modulatorNoiseVCA = context.createGain();  //Amplitude modulaattori
            this.modulatorNoiseVCA.gain.value =
this.currentNoiseVcaLFOGain;//ns. Syvyys (Volume)

            this.modGainNoiseVCA = context.createGain();
            this.modGainNoiseVCA.gain.value = this.currentNoiseVcaLFORate;
            this.noiseVcaLFO.connect(this.modGainNoiseVCA);
            this.modGainNoiseVCA.connect(this.modulatorNoiseVCA.gain);
  //VCA_Noise

  //Noise Filter
  // Create the filter.
  this.filterNodeNoise = context.createBiquadFilter();
  //filter type
  this.filterNodeNoise.type = this.currentNoiseLoHiType;
  this.filterNodeNoise.frequency.value = this.currentNoiseLoHiCutOff; //perus
cutOff, jotta kuulu edes jotain ON tilanteessa
  this.filterNodeNoise.Q.value = this.currentNoiseLoHiQ;

  //Delay
  this.delayNoiseNode = context.createDelay();
  this.delayNoiseNode.delayTime.value = this.currentNoiseDelayTime;
  this.delayTypeNoise = this.currentNoiseDelayType;

  //FeedBack
  this.feedbackNoiseNode = context.createGain();
  this.feedbackNoiseNode.gain.value = this.currentNoiseDelayFeedback;
```

```
  //Masterfilter, jotta korvat (eläimet) ja PA:t säästyy.
  this.filterMasterNoiseNode = context.createBiquadFilter();
  //filter type bandpass ja leikataan esim 5000Hz ala-ja yläpuoliset f 5dB /
tuplataajuss eli 22 Hz -10dB ja 360 KHz -10dB
  this.filterMasterNoiseNode.type = 'bandpass';
  this.filterMasterNoiseNode.frequency.value = 950;
  this.filterMasterNoiseNode.Q.value = 7;

  //Limiter
  //var preGain = context.createGain();
  this.limiterNoise = context.createDynamicsCompressor();
  this.limiterNoise.threshold.value =
this.limiterNoise.threshold.value;//0.0; // this is the pitfall, leave some
headroom
  this.limiterNoise.knee.value = 30.0; // brute force
  this.limiterNoise.ratio.value = 12.0; // max compression
  this.limiterNoise.attack.value = 0.005; // 5ms attack
  this.limiterNoise.release.value = 0.250; // 50ms release

  //Create Analyser aaltonäyttö
  this.analyserNode = context.createAnalyser();

  this.noiseSourceBuffer.connect(this.modulatorNoiseVCA);
  this.modulatorNoiseVCA.connect(this.gainNodeNoise);//(this.gainNodeNoise);
  this.gainNodeNoise.connect(this.filterNodeNoise);

  this.filterNodeNoise.connect(this.limiterNoise);
  this.filterNodeNoise.connect(this.delayNoiseNode);
  this.delayNoiseNode.connect(this.feedbackNoiseNode);
  this.feedbackNoiseNode.connect(this.delayNoiseNode);
  this.delayNoiseNode.connect(this.limiterNoise);

  this.limiterNoise.connect(this.filterMasterNoiseNode);
  this.filterMasterNoiseNode.connect(context.destination);
  this.filterMasterNoiseNode.connect(this.analyserNode);
  this.noiseSourceBuffer[this.noiseSourceBuffer.start ? 'start' :
'noteOn'](0);
  this.noiseVcaLFO.start(0); //[this.noiseLFO.start ? 'start' : 'noteOn'](0);

  //Canvas aaltonäyttö kutsu
  requestAnimFrame(this.visualize.bind(this));

  //Tilanne talletus play/pause napilla ja talletus
  this.currentNoiseGain = this.gainNodeNoise.gain.value;
  this.currentNoiseTime = this.noiseTime;
  this.currentNoiseFreq = this.noiseSourceBuffer.detune.value;
  this.currentNoiseLimit = this.limiterNoise.threshold.value;
  this.currentNoiseLoHiType = this.filterNodeNoise.type;
  this.currentNoiseLoHiCutOff = this.filterNodeNoise.frequency.value;
  this.currentNoiseLoHiQ = this.filterNodeNoise.Q.value;
  this.currentNoiseVcaLFOType = this.noiseVcaLFO.type;
  this.currentNoiseVcaLFOFreq = this.noiseVcaLFO.frequency.value;
  this.currentNoiseVcaLFOGain = this.modulatorNoiseVCA.gain.value;
  this.currentNoiseVcaLFORate = this.modGainNoiseVCA.gain.value;
  this.currentNoiseDelayType = this.delayTypeNoise;
  this.currentNoiseDelayTime = this.delayNoiseNode.delayTime.value;
  this.currentNoiseDelayFeedback = this.feedbackNoiseNode.gain.value;
};
```

```
//Noise tilanne talletus pause tilanteessa, playllä jatkuu siitä mihin
jäätiin
NoiseGenerator.prototype.stop = function() {
  this.currentNoiseGain = this.gainNodeNoise.gain.value;
  this.currentNoiseTime = this.noiseTime;
  this.currentNoiseFreq = this.noiseSourceBuffer.detune.value;
  this.currentNoiseLimit = this.limiterNoise.threshold.value;
  this.currentNoiseLoHiType = this.filterNodeNoise.type;
  this.currentNoiseLoHiCutOff = this.filterNodeNoise.frequency.value;
  this.currentNoiseLoHiQ = this.filterNodeNoise.Q.value;
  this.currentNoiseVcaLFOType = this.noiseVcaLFO.type;
  this.currentNoiseVcaLFOFreq = this.noiseVcaLFO.frequency.value;
  this.currentNoiseVcaLFOGain = this.modulatorNoiseVCA.gain.value;
  this.currentNoiseVcaLFORate = this.modGainNoiseVCA.gain.value;
  this.currentNoiseDelayTime = this.delayNoiseNode.delayTime.value;
  this.currentNoiseDelayFeedback = this.feedbackNoiseNode.gain.value;
  this.noiseSourceBuffer.stop(0);
  this.noiseSourceBuffer.disconnect();
};
//NoiseGen1 asetusten tallennus local storageen
NoiseGenerator.prototype.saveNoise1 = function() {
//Local Storage file koneella:
C:\Users\Lennartti\AppData\Local\Google\Chrome\User Data\Default\Local
Storage
  if (typeof(Storage) !== "undefined") {
    // Save
    localStorage.setItem("noise1Gain", noiseGen1.gainNodeNoise.gain.value);
    localStorage.setItem("noise1Time", noiseGen1.noiseTime);
    localStorage.setItem("noise1Rate",
noiseGen1.noiseSourceBuffer.detune.value);
    localStorage.setItem("noise1Limit",
noiseGen1.limiterNoise.threshold.value);
    localStorage.setItem("noise1LoHiType", noiseGen1.filterNodeNoise.type);
    localStorage.setItem("noise1LoHiCutOff",
noiseGen1.filterNodeNoise.frequency.value);
    localStorage.setItem("noise1LoHiQ", noiseGen1.filterNodeNoise.Q.value);
    localStorage.setItem("noise1VcaLfoType", noiseGen1.noiseVcaLFO.type);
    localStorage.setItem("noise1VcaLfoFreq",
noiseGen1.noiseVcaLFO.frequency.value);
    localStorage.setItem("noise1VcaLfoGain",
noiseGen1.modulatorNoiseVCA.gain.value);
    localStorage.setItem("noise1VcaLfoRate",
noiseGen1.modGainNoiseVCA.gain.value);
    localStorage.setItem("noise1DelayTime",
noiseGen1.delayNoiseNode.delayTime.value);
    localStorage.setItem("noise1DelayFeedback",
noiseGen1.feedbackNoiseNode.gain.value);
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
//NoiseGen2 tallennus
NoiseGenerator.prototype.saveNoise2 = function() {
  if (typeof(Storage) !== "undefined") {
    // noise2
    localStorage.setItem("noise2Gain", noiseGen2.gainNodeNoise.gain.value);
```

```
    localStorage.setItem("noise2Time", noiseGen2.noiseTime);
    localStorage.setItem("noise2Rate",
noiseGen2.noiseSourceBuffer.detune.value);
    localStorage.setItem("noise2LoHiType", noiseGen2.filterNodeNoise.type);
    localStorage.setItem("noise2LoHiCutOff",
noiseGen2.filterNodeNoise.frequency.value);
    localStorage.setItem("noise2LoHiQ", noiseGen2.filterNodeNoise.Q.value);
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
//NoiseGen1 asetusten latats local storagesta
NoiseGenerator.prototype.loadNoise1State = function() {
  if (typeof(Storage) !== "undefined") {
    //noiseGen1
    this.currentNoiseGain = localStorage.getItem("noise1Gain");
    this.currentNoiseTime = localStorage.getItem("noise1Time");
    this.currentNoiseFreq = localStorage.getItem("noise1Rate");
    this.currentNoiseLimit = localStorage.getItem("noise1Limit");
    this.currentNoiseLoHiType = localStorage.getItem("noise1LoHiType");
    this.currentNoiseLoHiCutOff = localStorage.getItem("noise1LoHiCutOff");
    this.currentNoiseLoHiQ = localStorage.getItem("noise1LoHiQ");
    this.currentNoiseVcaLFOType = localStorage.getItem("noise1VcaLfoType");
    this.currentNoiseVcaLFOFreq = localStorage.getItem("noise1VcaLfoFreq");
    this.currentNoiseVcaLFOGain = localStorage.getItem("noise1VcaLfoGain");
    this.currentNoiseVcaLFORate = localStorage.getItem("noise1VcaLfoRate");
    this.currentNoiseDelayType = localStorage.getItem("noise1DelayType");
    this.currentNoiseDelayTime = localStorage.getItem("noise1DelayTime");
    this.currentNoiseDelayFeedback =
localStorage.getItem("noise1DelayFeedback");
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
//NoiseGen2 talleteksen lataus
NoiseGenerator.prototype.loadNoise2State = function() {
  if (typeof(Storage) !== "undefined") {
    //noiseGen2
    this.currentNoiseGain = localStorage.getItem("noise2Gain");
    this.currentNoiseTime = localStorage.getItem("noise2Time");
    this.currentNoiseFreq = localStorage.getItem("noise2Rate");
    this.currentNoiseLoHiType = localStorage.getItem("noise2LoHiType");
    this.currentNoiseLoHiCutOff = localStorage.getItem("noise2LoHiCutOff");
    this.currentNoiseLoHiQ = localStorage.getItem("noise2LoHiQ");
  }
  else {
    document.getElementById("result").innerHTML = "Sorry, your browser does
not support Web Storage...";
    alert( "Sorry, your browser does not support Web Storage...");
  }
};
//Noise pääle/pois
NoiseGenerator.prototype.toggleNoise = function() {
```

```
  (this.isNoisePlaying ? this.stop() : this.NoisePlay());
  this.isNoisePlaying = !this.isNoisePlaying;
};
// Noise volume säätö
NoiseGenerator.prototype.changeNoiseVolume = function(element) {
  this.gainNodeNoise.gain.value  = element.value;
};
//Noise Limitteri
NoiseGenerator.prototype.changeNoiseLimiterTreshold = function(val) {
  this.limiterNoise.threshold.value = val;
};
//Noise taajuus ja time
NoiseGenerator.prototype.changeNoiseF = function(val) {
  this.noiseSourceBuffer.detune.value = val;
};
NoiseGenerator.prototype.changeNoiseT = function(val) {
  this.noiseTime = val;
};
//Noise taajuus ja time
//Noise VCA
//Noise VCAtyyppi
NoiseGenerator.prototype.changeNoiseVcaLFOType = function(type) {
  this.noiseVcaLFO.type = type;
};
//NOISE VCA_LFO Voimakkuus säätö
NoiseGenerator.prototype.changeNoiseVcaLFOGain = function(val) {
  this.modulatorNoiseVCA.gain.value = val;
};
//NOISE VCA_LFO Rate säätö
NoiseGenerator.prototype.changeNoiseVcaLFORate = function(val) {
  this.modGainNoiseVCA.gain.value = val;
};
//Noise vca_LFO taajuus
NoiseGenerator.prototype.changeNoiseVcaLFOFrequency = function(val) {
  this.noiseVcaLFO.frequency.value = val;
};
//Noise VCA
//BiQuadFilter ja Q-arvo
NoiseGenerator.prototype.changeNoiseFilterFrequency = function(element) {
  this.filterNodeNoise.frequency.value = element.value;// * 1000; //kerroin
1000 -> 20Hz - 10kHz
};

NoiseGenerator.prototype.changeNoiseFilterQ = function(element) {
  this.filterNodeNoise.Q.value = element.value;// * 10 //Q-arvo 0 - 20
};

//Filter type Hi/Lo
NoiseGenerator.prototype.changeNoiseFilterType = function(type) {
  this.filterNodeNoise.type = type;
};
//BiQuadFilter ja Q-arvo
//Delay ja FeedBack
NoiseGenerator.prototype.changeNoiseDelay = function(val) {
  this.delayNoiseNode.delayTime.value = val;
  console.log(this.delayNoiseNode.delayTime.value);
};
NoiseGenerator.prototype.changeNoiseFeedBack = function(element) {
  this.feedbackNoiseNode.gain.value = element.value;
```

```
  console.log(this.feedbackNoiseNode.gain.value);
};
//Delay ja FeedBack
//Canvas Noise näyttö
NoiseGenerator.prototype.visualize = function() {
  this.canvas1.width = this.WIDTH;
  this.canvas1.height = this.HEIGHT;
  var drawContext = this.canvas1.getContext('2d');
  //var drawContext = this.canvas.getElementById('Osc1').getContext('2d');

  //Aika/Gain
  var times = new Uint8Array(this.analyserNode.frequencyBinCount);
  this.analyserNode.getByteTimeDomainData(times);
  for (var i = 0; i < times.length; i++) {
    var value = times[i];
    var percent = value / 256;
    var height = this.HEIGHT * percent;
    var offset = this.HEIGHT - height - 1;
    var barWidth = this.WIDTH/times.length;
    drawContext.fillStyle = 'red';
    drawContext.fillRect(i * barWidth, offset, 1, 1);
  }

  //Taajuus/Aika näyttö
  var freqDomain = new Uint8Array(this.analyserNode.frequencyBinCount);
this.analyserNode.getByteFrequencyData(freqDomain);
for (var i = 0; i < this.analyserNode.frequencyBinCount; i++) {
  var value = freqDomain[i];
  var percent = value / 256;
  var height = this.HEIGHT * percent;
  var offset = this.HEIGHT - height - 1;
  var barWidth = this.WIDTH/this.analyserNode.frequencyBinCount;
  var hue = i/this.analyserNode.frequencyBinCount * 360;
  drawContext.fillStyle = 'hsl(' + hue + ', 100%, 50%)';
  drawContext.fillRect(i * barWidth, offset, barWidth, height);
}
  requestAnimFrame(this.visualize.bind(this));
};
/*Canvas Aaltonäyttö*/
//NOISE
/***************************/
/***************************/
            </script>
  <div>
    <button id="SaveUsAll" onclick="saveAll()">Save Us All!!!</button>
    <button id="LoadUsAll" onclick="loadAll()">Load All</button>
    <button id="PlayAll" onclick="playAll()">Play All</button>
    <button id="TheEnd" onclick="theEnd()">The End!!!</button>
    <button id="storageClear" onclick="clearStorage()">Delete all
saves</button>
  </div>
<script>
</script>
  <footer>
                        <div id="wrapper">
                                    <div id="content">
        <div class="post">
                                                              <h2>Web Audio Api
SoundGenerator</h2>
```

```
                                                                <p>
                                                                <p>Generating
basic tones at various frequencies using the <code> OscillatorNode,
</code><code> filterNodeNode </code><code>and GainNode etc etc
...</code>.</p>
                                                                </p>
                                                                <p>All coding and
page layout by Lennart. Thanks you all Internet!</p>
                                                        </div>
                                        </div>
                                </div>
                        </footer>
<!--Hiirellä soitto alkaa-->
<script>
//muuttujat mm. talletusta varten. talletusta kuitenkaan ei käytetä tässä
hiiri/canvas yhdistelmässä
var currentOsc
, currentOscGain
, currentOscFreq
, currentOscType
, currentNoise
, currentNoiFre
, currentNoiseGain
, currentNoiseTime
, currentNoiseDetune
, isMouseDown
, Touchmove
, lastY
, lastX
, lastVolY
, lastFreX
, noiseFreq
, noiseTime
, lastNoiFreX
, lastVolNoiY

function playWave(e) {

  var osc = context.createOscillator();
            var oscGain = context.createGain();

  lastVolY = e.pageY;
            lastFreX = e.pageX;
  isMouseDown = true;
  osc.frequency.value = 268.00;
  oscGain.gain.value = 0.3;
  osc.type = e.currentTarget.id;
  //osc.start(0);
  osc.connect(oscGain);
            oscGain.connect(context.destination);
            osc[osc.start ? 'start' : 'noteOn'](0);
  //Talennusta varten
  currentOsc = osc;
  currentOscType = osc.type;
            currentOscGain = oscGain;
  currentOscFreq = osc.frequency.value;
}
```

```
function stopWave(e) {
  isMouseDown = false;
  currentOsc.stop(0);
}
function stopNoise(e) {
            isMouseDown = false;
            currentNoise.stop(0);
}
function changeFreqVol(e) {
  if (e.currentTarget.id != 'noise') {
    if (e.pageX > lastFreX) {
      currentOsc.frequency.value += 5;
      lastFreX = e.pageX;
                        }
                        else if (e.pageX < lastFreX) {
      currentOsc.frequency.value -= 5;
      lastFreX = e.pageX;
                        }
    if (e.pageY > lastVolY) {
      currentOscGain.gain.value -= 0.01;
      lastVolY = e.pageY;
                        }
                        else if (e.pageY < lastVolY) {
      currentOscGain.gain.value += 0.01;
      lastVolY = e.pageY;
                        }
  }
}
function NoisePlay (e) {
            var noiseNode = context.createBufferSource();
  lastVolNoiY = e.pageY;
  lastNoiFreX = e.pageX;
  isMouseDown = true;
            noiseFreq = 100;
            noiseTime = 70;
            var buffer = context.createBuffer(1, noiseFreq*1000,
context.sampleRate);
  noiseNode.buffer = buffer;
            var data = buffer.getChannelData(0);
            //taajuus eli noiseFreq eli suurempi luku -> harvempi kohaus,
slider 1-100
                        //for lauseella kesto eli noiseTime eli suurempi luku
-> pitempi kohaus, slider 1-100
            for (var i = 0; i < noiseTime*1000; i++) {
                        data[i] = Math.random();
            }
            noiseNode.loop = true;
            gainNoise = context.createGain();
            gainNoise.gain.value = (0.5);
            //noiseNode.start(0);
  noiseNode.connect(gainNoise);
            gainNoise.connect(context.destination);
  console.log(noiseNode.detune.value);
  noiseNode[noiseNode.start ? 'start' : 'noteOn'](0);
  currentNoise = noiseNode;
  currentNoiseGain = gainNoise;
  currentNoiseDetune = noiseNode.detune.value;
}
function changeNoiseVol (e) {
```

```
            if (isMouseDown && e.currentTarget.id == 'noise') {
                        if (e.pageX > lastNoiFreX) {
      currentNoise.detune.value += 50;
      lastNoiFreX = e.pageX;
      console.log(currentNoiFre);
                        }
                        else if (e.pageX < lastNoiFreX) {
      currentNoise.detune.value -= 50;
      lastNoiFreX = e.pageX;
                        }

    if (e.pageY > lastVolNoiY) {
      currentNoiseGain.gain.value -= 0.01;
      lastVolNoiY = e.pageY;
                        }
                        else if (e.pageY < lastVolNoiY) {
      currentNoiseGain.gain.value += 0.01;
      lastVolNoiY = e.pageY;
                        }
  }
}
</script>
<script>
  //generaattori ja noise -objektien ilmentymät
  //jotka käyttävät omien SoundGenerator ja NoiseGenerator prototyyppi
funktioita
  //Nämä käyttävät siis samoja protyyppi funktioita eivätkä varaa muistiin
joka kerta itselleen ko. funktioita
  //muistia siis säästyy, jos näitä generaattoreita haluaa luoda useammankin,
vaikka sata, prosessori vaan taitaa
  //hyytyä siinä vaiheessa ;-)
  var soundGen1 = new SoundGenerator();
  var soundGen2 = new SoundGenerator();
  var noiseGen1 = new NoiseGenerator();
  var noiseGen2 = new NoiseGenerator();
  //sivun Refresh funktio eli kaikki säädöt alkuasetuksiin
  function theEnd() {
    location.reload();
  }
  //talletetaan molempien generaattorien molempien oscillattorien tilanne
  function saveAll() {
    soundGen1.saveOsc1();
    soundGen2.saveOsc2();
    noiseGen1.saveNoise1();
    noiseGen2.saveNoise2();
  }
  //ladataan molempien generaattorien molempien oscillattorien viimeksi
talletettu tilanne
  function loadAll() {
    soundGen1.loadOsc1State();
    soundGen2.loadOsc2State();
    noiseGen1.loadNoise1State();
    noiseGen2.loadNoise2State();
  }
  //käynnistetään molempien generaattorien molemmat oscillaattori viimeksi
ladattu tai pause tilanne
  function playAll() {
    soundGen1.OscPlay();
    soundGen2.OscPlay();
```

```
    noiseGen1.NoisePlay();
    noiseGen2.NoisePlay();
  }
  //Local Storage tyhjennys: localStorage.clear();
  function clearStorage() {
    localStorage.clear();
  }
</script>
            </body>
</html>
```