Felix Virtanen

Elektroniikka ja sähkötekniikka, 2. vuosikurssi

03.05.2021

Ohjelmoinnin peruskurssi Y2 – Projekti (Oma aihe)

Digitaalinen syntetisaattori (dokumentti)

2. Yleiskuvaus ja vaikeustaso

Luotu ohjelma on digitaalinen syntetisaattori graafisella liittymällä. Syntetisaattori sisältää kaksi oskillaattoria. Molempien oskillaattoreiden aallonmuotoa, oktaavia ja äänenvoimakkuutta pystyy säätämään erillisistä säätimistä. Oskillaattoreiden amplitudin vaihtumista ajallisesti pystyy säätämään ADSR-säätimillä. Ohjelmassa on lisäksi visuaalinen ikkuna, josta näkyy minkälainen aalto soi reaali-aikaisesti. Liittymästä löytyy myös yleisäänenvoimakkuuden säädin ja mahdollisuus tallentaa sekä ladata säätimien asennot. Syntetisaattori käyttää tietokoneen näppäimistöä, josta saa soitettua eri korkeuksien signaaleja. Ekstrana ohjelmassa on alipäästösuodatin valmiista kirjastosta, jota ei tarvitse ottaa arvostelussa huomioon.

3. Yleiskuvaus ja vaikeustaso

Ohjelma käynnistyy, kun ajetaan soundout.py. Syntetisaattori tuottaa tietynkorkeista ääntä painamalla näppäimistössä kirjaimia: a, w, s, e, d, f, t, g, y, h, u, j (pianon koskettimien mukaan). Säätimillä voi muokata ulostulevaa ääntä. Kannattaa aloittaa säätimien säätäminen äänenvoimakkuussäätimestä. Syntetisaattori toimii soittimena, joka mallintaa analogista syntetisaattoria. Kun haluat tallentaa muokkaamasi äänesi, voit tallentaa sen “Save”-napista preset- kansioon. Voi myöhemmin ladata kyseisen äänen takaisin “Load”-napista. Työ on mielestäni toteutettu keski-vaikeasti.

3. Ulkoiset kirjastot

Pyaudio: Tarvitaan signaalin toistamiseen reaali-aikaisesti sekä ulos kaiuttimista, (stream)

Threading: Tarvitaan toistamaan Pyaudion streamiä, että ohjelmassa voi tapahtua muutakin kuin yksi while-looppi.

Matplotlib: Signaalin kuvaajan piirtämiseen.

Numpy: Signaalin informaation säilyttämiseen numpyn taulukoihin (array)

Scipy.signal: käytetty square- ja saw-aallon muodostamiseen. (ekstrana alipäästösuodattimeen)

PyQt : Graafiseen ulkoasuun

Muut pythonin perusmoduulit: sys, os, math, unittest, time

5. Ohjelman rakenne

Ohjelman pääluokka on SoundOut. SoundOut sisältää jatkuva-aikaisen streamin, jonka kautta signaali muodostetaan ja soitetaan ulos kaiuttimista. Luokan päämetodi on play_sound, joka yhdistää oskillaattoreiden signaalit ja muut signaaliin vaikuttavat arvot. Play_soundin kautta muodostettu signaali myös menee kuvaajanpirtoon Plotter-luokkaan. SoundOut sisältää myös näppäimistön painamiseen liittyvää informaatiota (jonka voisi ehkä siirtää MainWindow-luokkaan).

Oscillator-luokan avulla muodostetaan oskillaattori ja sen tärkein metodi get_wave esivalmistelee SoundOut-luokalle oskillaattorin signaalin.

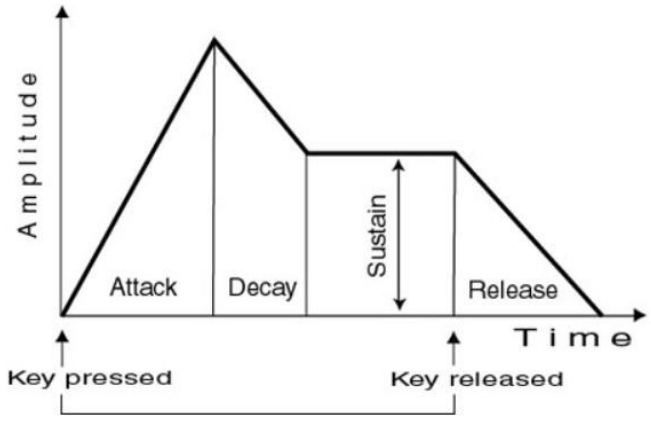
Ohjelman toinen pääluokka on MainWindow, joka on vastuussa syntetisaattorin graafisista ominaisuuksista sekä näppäimistön painalluksista. MainWindow sisältää myös syntetisaattorin lataus- ja tallennus metodit.

EnvADSR-luokka on vastuussa signaalin amplitudin vaihtumisesta ajallisesti. Sen tärkein metodi get_adsr tunnistaa missä kohtaa näppäimen painellusta ollaan ja milloin näppäimistöstä päästetään irti. Se laskee signaalin amplitudille 0–1 kertoimen, joka muuttuu säätimien mukaan.

WaveSlider-luokka muodostaa oskillaattorin signaalinmuodon säätimen. OctaveSlider-luokka muodostaa oskillaattorin oktaavin säätimen. VolumeKnob-luokka muodostaa äänenvoimakkuus säätimen. ADSRKnob-luokka muodostaa aina jonkun ADSR-säätimistä. LowPassKnob-luokka muodostaa alipäästösuodattimen säätimen. Plotter-luokka muodostaa visuaalisen kuvan signaalista ja päivittää sitä reaaliajassa.

TestParameters-luokka toimii ohjelman testerinä. Sillä voidaan testata, että mikään äänenvoimakkuuksista ei ylitä tiettyä rajaa.

6. Algoritmit



https://making-music.com/quick-guides/envelopes/

Oskillaattorin-kaava: y(T) = A1 * WAVE_TYPE(((2*pi*f)/rate)*T)

Signaalin ulostulon kaava: Y = A*ADSR-kerroin*(y1(T) + y2(T))

A1=oskillaattorin yksilökohtainen amplitudi

A= pääamplitudi

f = taajuus

rate = näytteet per sekunti (44100)

T = aika, aikaan sisältyy kerrallaan 1024 eri näytettä

ADSR-kerroin: joku alla olevista kertoimista, riippuen ajasta ja painalluksesta

Envelope laskee kertojia välillä 0-1:

Attack-kertoja = t / At

Laskee ajanhetken t ja attack-ajan suhteen.

Decay-kertoja = ((1- ((t - At) /Dt)) * (A - S*A) + S*A) / A

Kaavan vasen puoli (1- ((t - At) /Dt)) laskee, kuinka monta prosenttia ollaan attack-ajan ja decay-ajan välillä, kun attack-ajan kohdalla ollaan 100-prosentissa. Kertomalla kaavan vasenpuoli oikealla puolella ((A - S*A) + S*A) saadaan amplitudin suuruus ajanhetkellä t. Lopuksi lasketaan saadun amplitudin ja pääamplitudin suhde. Kokeilin myös laskea suoraan suoran kaavaa, mutta kaava antoi välillä arvoja, jotka olivat suurempia kuin 1.

Sustain-kertoja = S

Antaa lukuja 0-1 välillä.

Release-kertoja = (1-(t2 / Rt))*kertoja

Laskee ajanhetken t2 ja Release-ajan suhteen. Kertoja riippuu siitä, milloin release-aika on alkanut. Esim. Jos release-aika aloitetaan decay-kertojan aikana, tulee kertojaksi decay-kerroin.

t = aika painalluksen aloittamisesta

t2 = aika painalluksen irtipäästämisestä

A = Pää-amplitudi

At = Attack-aika

Dt = Decay-aika

S = Sustain-amplitudi-suhde

R = Release-aika

7. Tietorakenteet

Käytetään numpyn taulukkoja (array) säilömään signaalin informaatio, koska ne ovat pythonin listoja tehokkaampia.

8. Tiedostot

Ohjelmalla voi tallentaa säätimien paikat csv tiedostoon ja myöhemmin lukea ne tallennetuista tiedostoista. Tarkoitus ei ole luoda csv-tiedostoa ohjelman ulkopuolella ja lukea se ohjelman avulla. Säätimien tiedot tallennetaan tiedostoon tietyssä järjestyksessä ja ne ladataan samassa järjestyksessä.

9. Testaus

Ohjelman teon edetessä testasin ohjelmaa pääosin kuuntelemalla, toimiiko kyseinen-osa oikein ja korjasin sitä mukaa, kun huomasin virheen.

Ohjelman äänenvoimakkuuksille on erikseen testit synth_test.py -tiedostossa. Testit varmistavat, että pääamplitudi ei voi kasvaa korkeammaksi kuin 0.15, oskillaattoreiden omat amplitudit eivät kasva yli yhden ja adsr-kerroin ei kasva yli yhden.

10. Ohjelman tunnetut puutteet ja viat

Presettiä tallennettaessa ja ladatessa ohjelma ei avaa suoraan preset-kansiota. (Lowpass filtterin säätimen logaritminen asteikko ei toimi täysin halutulla tavalla). Äänenvoimakkuus kasvaa ehkä hieman liian kovaksi, jos kaikki äänenvoimakkuudet ovat maksimissa ja oskillaattorit syöttävät samaa aaltoa samalla oktaavilla. Jos csv-tiedoston tekee ohjelman ulkopuolella voi se saada ohjelman jumittamaan.

Ohjelman kokoa ei voi tällä hetkellä muokata.

11. 3 parasta ja 3 heikointa kohtaa

Pidän envelopen toimintaa erityisen hyvänä ja miten osa luokista on järjestetty niin että samalla luokalla voi muodostaa monta eri toimintoa (Volumeknob). Myös näppäiden painaminen ja näppäiden pois päästäminen toimii halutulla tavalla adsr-säätimien kanssa eli näppäimiä voi painaa montaa samaan aikaan, mutta ainoastaan viimeisin soi (monofoonisuus).

Heikoimmat kohdat ovat kuvaajan piirtäminen ja alipäästö-suodatin, sillä ne hyödyntävät suurimmaksi osaksi kirjastoja (matplotlib ja scipy.signal).

12. Poikkeamat suunnitelmasta

Muodostin enemmän luokkia, kuin alkuperäisessä suunnitelmassa, sillä ymmärsin että on yksinkertaisempaa tehdä luokka tietylle säädintyypille. En myöskään ehtinyt tehdä alipäästösuodatinta. Jotkin suunnitelman aikatauluista arvioin hieman alakanttiin.

13. Toteutunut työjärjestys ja aikataulu

25.02. Projektin aloitus ja gitin tekeminen

13.03. Pythonin version vaihtaminen 3.6.8

19.03. Ensimmäisen signaalin muodostus, ensimmäinen ääni ulos kaiuttimista napin painalluksella

20–21.03. Eri taajuisten äänten aktivointi näppäimillä. WaveSlider, OctaveSlider, VolumeKnob ja AttackKnob koodit ja graafiset säätimet.

27–28.03. Loput ADSR-säätimet ja niiden graafiset osat.

02-04.04. Toisen oskillaattorin muodostus ja sen graafiset säätimet. Graafisten luokkien uudestaan-järjestelemistä. Virheen korjauksia näppäimistön kanssa.

09-10.04. Tutustuminen kuvaajan muodostamiseen graafiseen ulkoasuun ja sen tekeminen ja hienosäätö.

13.04. Tallennus ja lataus ominaisuuden tekeminen.

24–25.04. Ulkoasun muuttaminen. Säätimien ulkoasun muuttaminen. ADSR-toiminnon hienosäätö. Näppäimistön painallusten hienosäätö ADSR-kanssa. Virheen korjauksia.

02.05. Tutustuminen alipäästösuodattimeen. Koodin selkeyttäminen, turhien kommenttien poisto ja avustavien kommenttien lisääminen.

04.05. Dokumentin teko ja viimeiset siistimiset

Noudatin suunnitelmaa kohtalaisen hyvin, mutta päätin aloittaa graafisen ulkoasun tekemisen ennen ADSR-toiminnon lisäämistä.

14. Arvio lopputuloksesta

Sain ohjelmaan hyvin syntetisaattorin perustan, josta on hyvä lähteä kehittämään sitä omanlaisekseen. Ohjelma on miellyttävä silmille ja suhteellisen helppokäyttöinen ja eri säätimien toiminnot huomaavat helposti kuuntelemalla ja katsoen kuvaajaa.

15. Viitteet

Äänen ulostulon ja signaalin lähteitä:

https://people.csail.mit.edu/hubert/pyaudio/docs/

https://www.wizard-notes.com/entry/dev/pyqt5-pyaudio-simple-synthesizer

https://github.com/Kurene/simple-synthesizer-with-pyqt5-pyaudio/blob/master/main.py

https://github.com/denczo/EardrumBlaster

https://www.youtube.com/watch?v=DdbYj3o3-3k

https://sidparida.com/2017/07/12/python-audio-synthesis-from-scratch-a-tutorial-series-part-2-creating-the-oscillators-and-sound-generation-module/

https://www.youtube.com/watch?v=dNJ171EsoNQ

https://www.youtube.com/watch?v=tgamhuQnOkM&t=1365s

ADSR:

https://making-music.com/quick-guides/envelopes/

https://www.youtube.com/watch?v=OSCzKOqtgcA&t=1457s

Tallennus ja lataus:

https://www.youtube.com/watch?v=fH0GQuF0FdE

Kuvaajan piirto (matplotlib):

https://matplotlib.org/stable/tutorials/intermediate/artists.html

https://www.delftstack.com/howto/matplotlib/how-to-automate-plot-updates-in-matplotlib/

https://matplotlib.org/stable/gallery/color/named_colors.html

https://matplotlib.org/3.3.4/api/_as_gen/matplotlib.figure.Figure.html

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.ion.html

Testaus:

https://pytest-qt.readthedocs.io/en/1.3.0/
http://johnnado.com/pyqt-qtest-example/
https://www.dlab.ninja/2014/02/testing-your-python-code-with-unittest.html
https://plus.cs.aalto.fi/y2/2021/materials_k03/k03_softwaretesting_pythontesting/

Alipäästö-suodatin:

https://docs.scipy.org/doc/scipy/reference/signal.html

https://stackoverflow.com/questions/12093594/how-to-implement-band-pass-butterworth-filter-with-scipy-signal-butter

Perus-kirjastot:

https://docs.python.org/3/library/time.html

https://numpy.org/doc/stable/reference/generated/numpy.arange.html

https://docs.python.org/3/library/os.path.html

Näppäimistö:

https://en.wikipedia.org/wiki/Piano_key_frequencies

https://doc.qt.io/qt-5/qkeyevent.html

https://forum.qt.io/topic/101143/problem-with-eventfilter-and-qevent-keyrelease-event/3

Qt:

https://plus.cs.aalto.fi/y2/2021/materials_k05/k05_pyqt_pyqt/

https://doc.qt.io/qt-5/qpushbutton.html

https://doc.qt.io/qt-5/qlabel.html

https://www.tutorialspoint.com/pyqt/pyqt_qlabel_widget.htm

https://doc.qt.io/qt-5/qdial.html

https://pythonbasics.org/qdial/

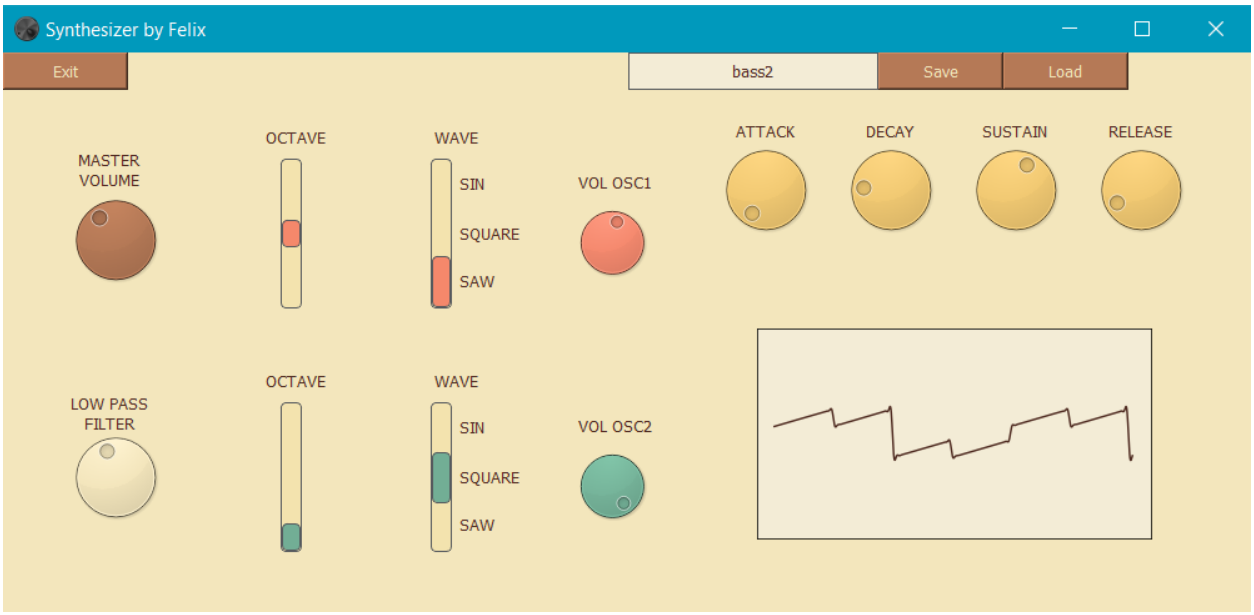
https://doc.qt.io/qt-5/qslider.html

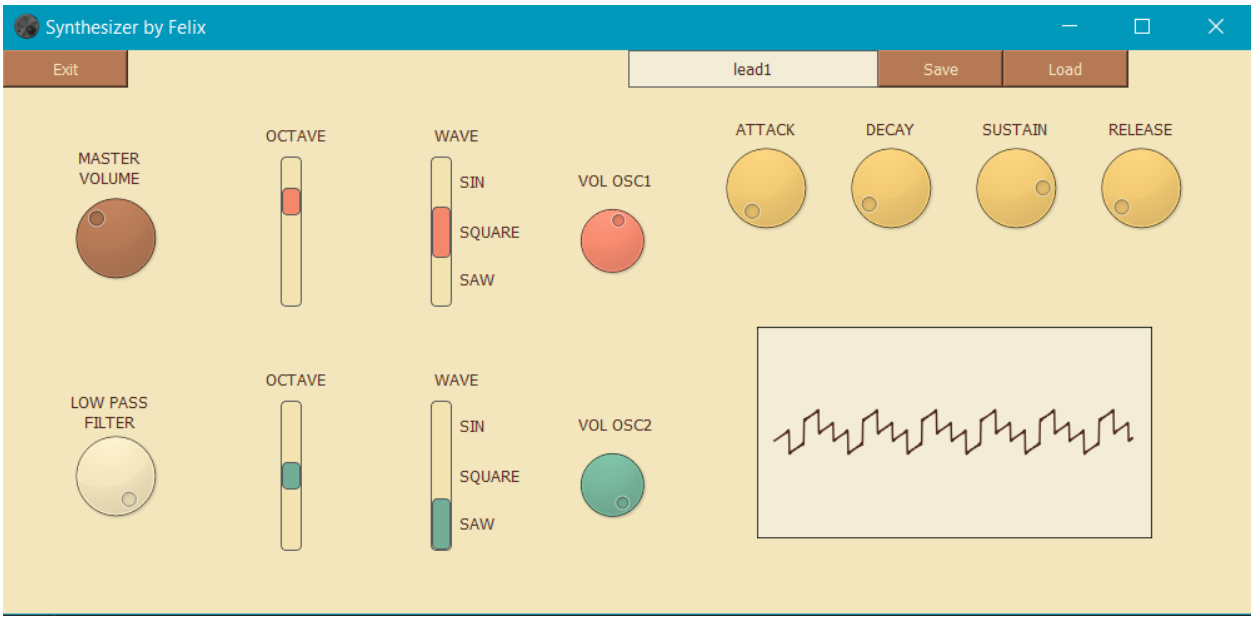
https://www.tutorialspoint.com/pyqt/pyqt_qslider_widget_signal.htm

https://forum.qt.io/topic/66256/changing-colors-of-the-qslider

https://stackoverflow.com/questions/29284643/how-to-set-an-icon-on-a-main-window-and-action-with-qt

16. Liitteet


Synthesizer by Felix
Exit
bass2
Save
Load
MASTER VOLUME
OCTAVE
WAVE
SIN
SQUARE
SAW
VOL OSC1
ATTACK
DECAY
SUSTAIN
RELEASE
LOW PASS FILTER
OCTAVE
WAVE
SIN
SQUARE
SAW
VOL OSC2


Synthesizer by Felix
Exit
lead1
Save
Load
MASTER VOLUME
OCTAVE
WAVE
SIN
SQUARE
SAW
VOL OSC1
ATTACK
DECAY
SUSTAIN
RELEASE
LOW PASS FILTER
OCTAVE
WAVE
SIN
SQUARE
SAW
VOL OSC2