Jakub Novák

Synth Challenge 2017

Povinná skladba – Let Čmeláka

Vybral jsem si strunný nástroj – elektronickou kytaru.

Syntéza

Rozhodl jsem se pro Karplus-Strongův Algoritmus [1, 1.1], poněvadž mi to přijde jako nejvhodnější možnost ze všech syntéz pro strunné nástroje.

V komentáři funkce **synthKytary** je podrobně popsán každý řádek.

Ve zkratce jde o vytvoření vektoru nul, jeho velikost udává délka tónu.

Dále se nastaví zpoždění podle základní frekvence tónu dle rovnice zpozdeni = (Fs/freq).

Koeficienty filtru pro generování akustické kytary jsem zjistil ze stránky mathworks.com [5].

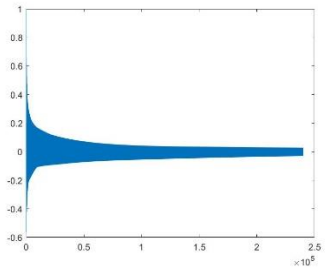
Potom je již v podstatě filtruje a získáváme výsledný signál.

Efekty

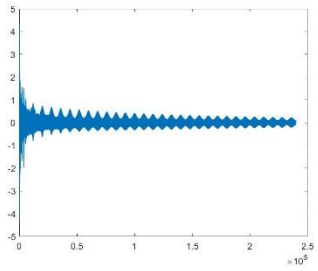
Chorus [2] a vibrato [3] jsou efekty od Jana Chvojky ze cvičení.

Distortion [4] je převzato ze stránky Steve McGoverna a je to v podstatě jen přenásobení signálu koeficientem a následná normalizace.

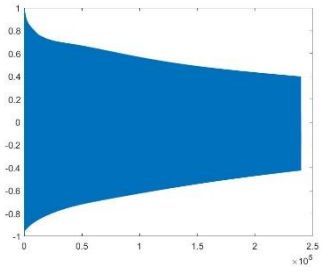
Vliv efektů na signál:



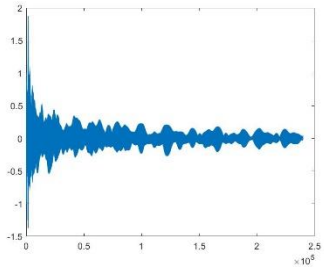
Původní signál



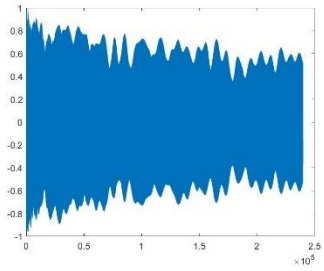
Vibrato



Distortion



Chorus



Výsledek

Stupnice

Generuje 2 oktávy Durové stupnice podle prvního tónu.

Konkrétně jsem vygeneroval a odevzdal D-dur.

Ve funkci **main** si můžete vyzkoušet jiné délky a různé Durové stupnice v rozsahu c1 až c2:

```
64      %%
65 -    Fs = 48000;
66 -    stupnice = stupniceFn('d1',2, Fs); % stupniceFn(pocatecniNota, doba, Fs);
67 -    soundsc(stupnice,Fs);
```

Číslo půltónů je generováno for cyklem.

```
18 -    klavesy = [];
19 -    k = pocatecniNota;
20 -    while length(klavesy) <= 14             % generovani stupnice
21 -        klavesy = [klavesy k];
22 -        if k==pocatecniNota+4 || k==pocatecniNota+11 || k==pocatecniNota+16 || k==p
23 -            k = k+1;
```

Na klavíru je rozdíl půltónu mezi E a F a mezi H a C, tudíž je tam +4 a +11 a je to zopakováno pro další oktávu (+12 půltónů ke každé z předchozích hodnot).

Dále je zde využita funkce nota, která podle exponenciální funkce čísla půltónu určuje jeho frekvenci.

$$f(n) = 2^{\frac{n-49}{12}} \times 440\,\mathrm{Hz}$$ [6]

Kde n je číslo půltónu.

Funkce má tvar:

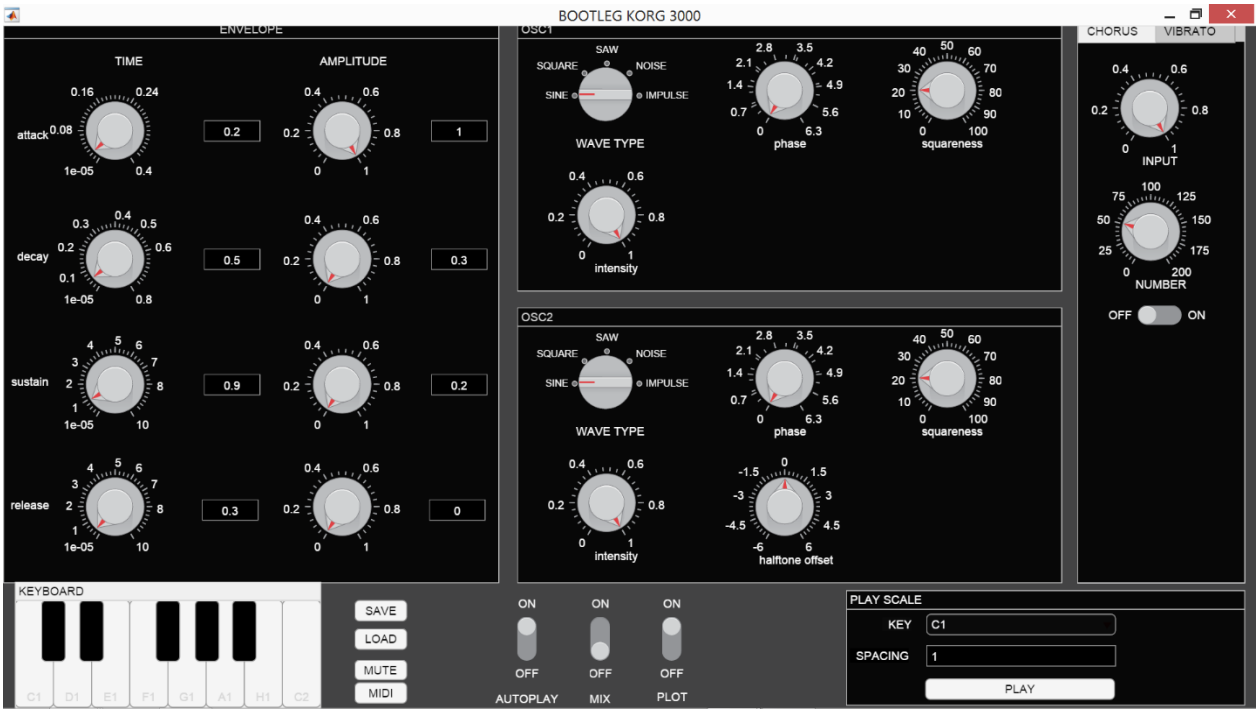
stupniceFn(pocatecniNota, doba, Fs)

kde **pocatecniNota** je ve tvaru stringu např. ‘c1’, ‘dis1’… (až po ‘c2’), doba je čas v sekundách mezi tóny a Fs je vzorkovací frekvence.

Funkce také používá funkci **convertNoteName**, která převádí stringy na čísla půltónů.

Volná tvorba

Návod s komentářem zde: https://www.youtube.com/watch?v=6Y03tzSaanQ



Ve volné tvorbě jsem se zkusil přiblížit alespoň základním funkcím syntetizéru KORG MS-20, proto název „BOOTLEG KORG 3000“. Umístění ve složce je **SynthChallenge-NovakJakub/appdesigner-UI/synthV1.mlapp** (bohužel narážím na problémy při přejmenovávání).

KORG MS-20 [7]

Jelikož jsou všechny funkce demonstrovány ve videu, nebudu se zde opakovat a rovnou zde popíšu kód.

Většina funkcí je okomentovaných (většinou Anglicky, jelikož plánuji to uploadovat na matlabová fóra a chtěl bych od lidí nějaký feedback).

Všechny parametry jsou dány na začátku v properties (AppDesigner), tyto funkce jsou v podstatě globální a ukládám do nich všechny parametry, abych je mohl používat navzájem mezi funkcemi.

signalGenerator - generování zvuku, funkce je založena na funkci sinus a např. pro průběhy obdélníkové tam jsou zase jen liché převrácené násobky sinů. Pro šum je tam generování Gaussovského šumu.

genSignal - obálka a ukládání do proměnných app.signalWave a app.signalWave2, což jsou vlastně OSC1 a OSC2 ekvivalenty v GUI.

playMix resp. **makeMix** - (rozdíl je v tom, že jeden přehrává a druhý vrací signál) potom aplikuje app.intensityVal (což je vlastně poměr mixů 2 oscilátorů) a také všechny efekty, pokud jsou zapnuté.

autoplayFcn – pokud je switch v ON, tak při každé změně se přehraje generovaný signál

soundAndPlot - soundsc + plot funkce

addEffects - mění signál, pokud je slider pro daný efekt zapnutý.

loadPreset - je mírně „špagetová“ záležitost, ale v podstatě jen ukládá všechny app.XXX proměnné do .mat souboru, který se nachází ve složce presets.

playNote – v podstatě jen podle klávesnice změní app.note resp app.note2 a znovu vygeneruje signál, který přehraje.

convertNote – konvertuje jméno půltónu na číslo půltónu např. c1 je číslo 40., zatím umí jen rozsah od c1 do c2

nota - na základě čísla půltónu generuje jeho frekvenci

Callback funkce – každá callback funkce k 90% tlačítkům zahrnuje **autoplayFcn** a také změnu dané proměnné.x

Callback funkce **PLAYButtonPushed** – je to samé co část 2. synth challenge, ale musí být nativně v AppDesigneru, kvůli tomu, že samotné generování signálu a všechny proměnné jsou v něm, to jsem ale narychlo rozřešil v tlačítku MIDI

Všechny funkce ve složce **MIDI Button** jsou přepsány z AppDesigneru do regulérních funkcí, akorát se hodí jen v některých případech.

Efekty jsem použil stejné jako v **synthKytary** z části 1.

Zdroje

[1] http://sami.fel.cvut.cz/syn/SYN10.pdf

[1.1]https://en.wikipedia.org/wiki/Karplus%E2%80%93Strong_string_synthesis

[2] http://sami.fel.cvut.cz/syn/chorus.txt

[3] http://sami.fel.cvut.cz/syn/flanger_vibrato.txt

[4] http://www.steve-m.us

[5] https://www.mathworks.com/help/signal/examples/generating-guitar-chords-using-the-karplus-strong-algorithm.html

[6] https://en.wikipedia.org/wiki/Piano_key_frequencies

[7] https://goo.gl/images/ySSLPA

[8] https://www.mathworks.com/matlabcentral/