[CTP431]Project 3
Soya Park soya@kaist.ac.kr
November 24, 2015

Code Documentation
FM Synthesizer

#1. Making the lowpass filter parameters tunable

This project is adding two elements for each preset named filter frequency(FilterFreq), filter frequency target(FiterFregTarget) and filter frequency decay time(FilterFreqDecayTime). This is shown at line 12 to 15.

```
1    // preset parameters
2    this.modulationIndex = parameters.modulationIndex;
3    this.modulationFrequency = frequency /
     parameters.carrierModulationRatio;
4    this.IndexAttackTime = parameters.IndexAttackTime;
5    this.IndexDecayTime = parameters.IndexDecayTime;
6    this.IndexSustainLevel = parameters.IndexSustainLevel;
7    this.IndexReleaseTime = parameters.IndexReleaseTime;
8    this.AmpEnvAttackTime = parameters.AmpEnvAttackTime;
9    this.AmpEnvDecayTime = parameters.AmpEnvDecayTime;
10   this.AmpEnvSustainLevel = parameters.AmpEnvSustainLevel;
11   this.AmpEnvReleaseTime = parameters.AmpEnvReleaseTime;
12   this.FilterFreq = parameters.FilterFreq;
13   this.FilterFreqTarget = parameters.FilterFreqTarget;
14   this.FilterFreqDecayTime = parameters.FilterFreqDecayTime;

     //In order to determine wet ratio
15   this.wet = reverbRatio.val.value;
```

#2. Adding a new preset

I add a new preset named bell as following.

```
var bell_params = {
      presetName: "Bell",
      carrierModulationRatio: 1/1.4,
      modulationIndex: 2,
      IndexAttackTime: 0,
      IndexDecayTime: 10,
      IndexSustainLevel: 0.001,
      IndexReleaseTime: 3,
      AmpEnvAttackTime: 0,
      AmpEnvDecayTime: 10,
      AmpEnvSustainLevel: 0.001,
      AmpEnvReleaseTime: 3,
      FilterFreq: 5000,
      FilterFreqTarget: 2000,
      FilterFreqDecayTime: 5
};

// add presets
 . . .
presets.push(bell_params);
```

#3. Adding reverberation effect unit

I reuse the function from mini-project2 in order to add a reverberation unit and also make new gain for dry and wet ratio respectively.

```
//load impulse response
Voice.prototype.loadImpulseResponse  = function(type) {
      var request = new XMLHttpRequest();
      var url = type; //"memchu_ir2.wav";
      var conv = this.convolver;

      request.open('GET', url, true);
      request.responseType = 'arraybuffer';
      request.onload = function() {
          context.decodeAudioData(request.response, function(buffer) {
                  conv.buffer = buffer;
          });
      }

      request.send();
};
```

#4. Designing your own sound

I made organ sound with very low modulation and long amplitude release time as following.

```
var organ_params = {
        presetName: "organ",
        carrierModulationRatio: 0.5,
        modulationIndex: 0.01,
        IndexAttackTime: 0,
        IndexDecayTime: 10,
        IndexSustainLevel: 0.001,
        IndexReleaseTime: 3,
        AmpEnvAttackTime: 0.1,
        AmpEnvDecayTime: 0.001,
        AmpEnvSustainLevel: 0.001,
        AmpEnvReleaseTime: 5,
        FilterFreq: 4000,
        FilterFreqTarget: 2000,
        FilterFreqDecayTime: 5
};
```

I add five knobs - modulation index, filter frequency, filter frequency target, filter frequency decay time and wet/dry ratio - and amplitude envelope so that users are able to control parameter. I used NexusUI library(http://nexusosc.com/) for knob design.

```
for (var key in nx.widgets) {
        with (nx.widgets[key]) {
                on('*', function(data) {
                        // code that will be executed
                        synth.setParameter(canvasID, data.value)
                })
        }
}
```

```
FmSynth.prototype.setParameter = function(inParam, inValue) {
        this.parameters[inParam] = inValue * scale_param[inParam];
}
```

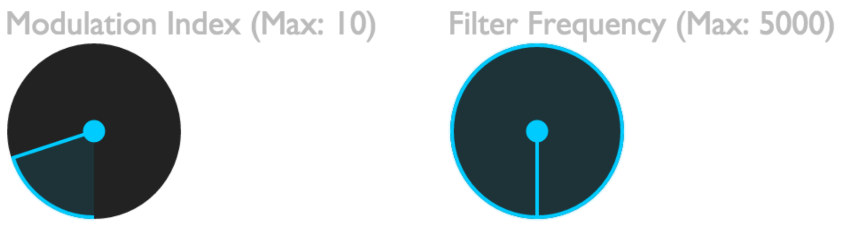

Figure 1. Example of knobs