Nicholas Marks

3/17/2014

MAT209ia

For my final piece I decided to use three different instruments. Instruments 1, 5, and 7 corresponded to the Sine, Oscillator Tremolo, and Additive Synthesis instruments. I always liked the simplicity of the Sine Instrument. I create long chords and drones with this instrument in order to have notes enter and exit at specific times. By using this technique I am able to have notes start as an arpeggio which is held out into a chord. Another interesting technique is to have one chord have a long decay while a second has a long attack, while having both notes begin at the same time the effect we hear is a chord that morphs from one to the next. Usually an interesting dissonance can be heard causing beating during the middle of the note which resolves to the second chord. At times I also use this instrument for melodic content when I do not want tremolo.

My intentions for the OscTrem instrument were to create interesting patterns due to offset ratios for the tremolo but I could not get the sound I was expecting. The main use of this instrument is during Melodic content which has both long and short notes. The tremolo is not as noticeable during the short notes which gives the longer notes more feeling and power. I also like the sound of using chords in the tremolo instrument to show a difference between the sine chords and tremolo chords.

With the Additive Synthesis Instrument I did not have enough time to translate my melodic content into more interesting sounds. The way that I use this instrument is to create drone sounds in between themes, I also try to use this instrument to fill in chords and background bass notes. My goal was to find melodic sounds whose timbre fit the melodies that I wrote with the sine instrument, and then replace the single sine instrument with more a more colorful additive synthesis instrument.

The structure of my piece will begin with a theme consisting of a chord followed by two ascending lines offset by a small time. The second line descends halfway through leading down to a second darker chord. This melody is played with the sine instrument and repeats four times with an increasing speed. With the space between repetitions the Additive Synthesis instrument uses the fill time function to create menacing undertones. A short interlude of Additive Synthesis tones leads into the A section.

// Intro

//chord

sinWhole (s, time, 600, dt \* 5, -0.06, 0.0, dt) ; //// These Freq are also used in chord, commented to reserve space

// 650 720 780 860

time += dt \* 4 ;

// bass Chord

sinWhole (s, time, 55 \* 0.25, dt \* 5, 0, 0, dt) ;

sinWhole (s, time, 55 \* 0.5, dt \* 5, 0, 0, dt) ;

sinWhole (s, time, 55, dt \* 5, 0, 0, dt) ;

sinWhole (s, time, 110, dt \* 5, 0, 0, dt) ;

sinWhole (s, time, 55 \* 2/3, dt \* 5, 0, 0, dt) ;

sinWhole (s, time, 55 \* 4/5 , dt \* 5, 0, 0, dt) ;

//set tempo

tempo = tempo \* 2;

dt = (float)60 / tempo ;

saveTime = time;

time += dt \* 0.5;

//A section melody

for (int i=0; i<=3; i++) { sinQ (s, time, freq, dt) ; time += dt; }

for (int i=0; i<=3; i++) { sinQ (s, time, freq, dt) ; time += dt; freq = freq \* (5 \* 0.25) ; }

for (int i=0; i<=3; i++) { sinQ (s, time, freq, dt) ; time += dt; freq = freq \* (5 \* 0.25) ; }

freq = 110 \* 9/4;

time = saveTime + dt \* 6.75 ;

for (int i=0; i<=3; i++) { sinQ (s, time, freq, dt) ; time += dt; freq = freq \* (2 \* 0.667) ; }

for (int i=0; i<=5; i++) { sinQ (s, time, freq, dt) ; time += dt; freq = freq \* 0.667 ; }

time += -dt \* 0.5 ;

sinWhole (s, time, 600 \* 0.8, dt \* 8, -1.9, 3.5, dt) ; //// These are also used but commented to reserve space

/// 650 720 780 860

sinWhole (s, time, 600 \* 0.4, dt \* 8, -1.9, -0.95, dt) ;

// 650 720 780 860

//Bass

sinWhole (s, time, 70, dt \* 8, -1.5, -0.9, dt) ;

// 93 113 136 168

// END Intro

The A section of the piece has a 4 chord baseline (I Know stereotypical), and unfortunately it appears my melody is basically 4/4 time. The baseline repeats four times and follows the same tempo that the melody follows, each variation is slightly different with regards to which notes are held and which ones lead to the next chord. I use an array called fluteMel to store the main A section melody and use a loop to iterate through the note list. This melody repeats 3 times at different tempos. The first iteration is at tempo 90, repeat at 80, and last at 60. During the third repetition a countermelody is added which is based off the notes in the fluteMel array. The countermelody leads from the third repetition to an improv computer solo during the fourth part. The solo usually ends with some kind of half cadence to leave a feeling of wanting more, or a resolution, that does not come. The countermelody is created using the same array as the main melody, but the order of notes and note lengths is varied differently from the main melody.

float fluteMel[] = {0, 1046.50, 1174.66, 1318.51, 880.00, 830.61, 880.00, 987.77, 739.99, 659.25, 493.88, 523.25 } ;

float fluteLen[] = {1, 2, 0.5, 0.5, 4, 3, 0.5, 0.5, 1, 1, 1, 1} ;

float bassline[] = { 261.63, 207.65, 196.00, 174.61, 164.81, 146.83} ;

// Main Theme

for (int i =0; i<12; i++) {

len = dt \* fluteLen[i];

if ( i == 5)

s.add<OscTrm>( time1 ).set(dt \* 4, fluteMel[5], amp, len \* 0.1 , len \* 0.2 , sus, 0.4,4,8,0.5, tbSqr, 0.8);

if ( i == 8)

s.add<OscTrm>( time1 ).set(dt \* 4, fluteMel[8], amp, len \* 0.1 , len \* 0.15 , sus, 0.4,4,8,0.5, tbSqr, 0.8);

if (i == 6) { }

else if (i == 7) { }

else if (i > 8) { }

else {

s.add<OscTrm>( time1 ).set(len, fluteMel[i], amp, len \* 0.1 , len \* 0.1 , sus, 0.4,4,8,0.5, tbSqr, 0.8);

}

time1 += len;

}

//Counter Melody

float temptime = time1;

for (int i=0; i<15; i++) {

freq = fluteMel[i%6] + 1 ; len = 0.5 ;

s.add<SineEnv>( temptime ).set( len, freq, 0.1, 0.03, 0.03);

temptime += len;

}

The Bass repeats four times with slight additions and variations to each chord structure. The first repetition is the simplest and shows how I outlined the pattern. Since I wrote the bass line separately from the main melody I save the time before the main melody and restore it at the beginning of the bassline. I also added the time changes after my first draft and I had to coordinate time changes between the two sections.

// START BASS

tempo = 90;

dt = (float)60 / tempo ;

for (int i = 0; i<4; i++) {

s.add<OscTrm>( time ).set(dt \* 4, bassline[i%4], 0.3, dt \* 0.05, dt \* 0.5 , 0.1, 0.4,4,8,0.5, tbSin, 0.8);

s.add<OscTrm>( time ).set(dt \* 4, bassline[i%4], 0.3, dt \* 3 , dt \* 0.05, 0.1, 0.4,4,8,0.5, tbSin, 0.8);

time += dt \* 4;

}

As the melodies of the A section trail off the Oscillator Tremolo instrument creates high pitched glissandi while the Sine instrument creates different rhythmic patterns. The first pattern consists of triplets at a tempo of 180 with a moving triplet at half the speed. The half speed triplet slows down with each repetition and does not stay in phase with the original triplet. After two triplets sound a dotted eighth - sixteenth note pattern also sounds to create a wavering triplet sound.

// Pattern based off a three pulse with the dominant amplitude on one.

for (int i = 0; i < 6; i++) {

for (int j=0; j<4; j++) {

sinQ (s, time1, 210, dt3 ) ;

time1 += dt3 ;

sinQ (s, time1, 210, dt3 , 0.1) ;

time1 += dt3 ;

sinQ (s, time1, 210, dt3 , 0.1) ;

time1 += dt3 ;

}

}

// Dotted eighth - sixteenth pattern. When overlayed on the 3 pulse above a cool effect is achieved.

for (int i = 0; i < 5; i++) {

for (int j=0; j<4; j++) {

sinQ (s, time3, 210 \* 1.25 , dt4 \* 3 ) ;

time3 += dt4 \* 3 ;

sinQ (s, time3, 210 , dt4 , 0.1) ;

time3 += dt4 ;

}

}

// Melodic triplet that changes pitch and slows down in tempo

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

tempo += -3 ;

dt = (float)60 / tempo ;

dt3 = dt \* 0.3333333333333 ;

if (i == 2) { df[0] = 1.3334; df[1] = 1.083335; df[2] = 0.87; }

else if (i == 5) { df[0] = 1.3334; df[1] = 1.083335; df[2] = 0.87; }

else if (i == 8) { df[0] = 1.3334; df[1] = 1.083335; df[2] = 0.87; }

else {

df[0] = 1.5; df[1] = 1.25; df[2] = 1;

}

// for (int j=0; j<4; j++) {

sinQ (s, time2, 210 \* df[0] \* 2, dt3 , 0.3) ;

time2 += dt3 ;

sinQ (s, time2, 210 \* df[1] \* 2, dt3 , 0.15) ;

time2 += dt3 ;

sinQ (s, time2, 210 \* df[2] \* 2, dt3 , 0.07) ;

time2 += dt3 ;

// }

}

There is a brief glimpse the introduction material which repeats multiple times quickly and then abruptly stops. These rhythmic patterns are interspersed with Additive Synthesis drones and OscTrem spacey notes. The final rhythmic pattern has a quiet five-let and moving triplet which also decreases in tempo until the piece ends with basic additive synthesis and sine tones.

// Pulse in 5

for (int i=0; i<10; i++) {

len = rand(1.5, 2.4); atk = 0.5; dcy = 0.98; amp = 0.1;

s.add<SineEnv>( time2 ).set( len, freq, amp, atk, dcy);

time2 += dt5 ; amp = 0.1;

s.add<SineEnv>( time2 ).set( len, freq, amp, atk, dcy);

time2 += dt5 ;

s.add<SineEnv>( time2 ).set( len, freq, amp, atk, dcy);

time2 += dt5 ;

s.add<SineEnv>( time2 ).set( len, freq, amp, atk, dcy);

time2 += dt5 ;

s.add<SineEnv>( time2 ).set( len, freq, amp, atk, dcy);

time2 += dt5 ;

}

// Pulse in 3 that occurs at the same time as the one above in 5.

time1 = saveTime2 + dt5 \* 10;

for (int i = 0; i < 1; i++) {

for (int j=0; j<4; j++) {

sinQ (s, time1, 210 \* 2 , dt3 , 0.05) ;

time1 += dt3 ;

sinQ (s, time1, 210 \* 2 , dt3 , 0.2) ;

time1 += dt3 ;

sinQ (s, time1, 210 \* 2 , dt3 , 0.1) ;

time1 += dt3 ;

}

}

In order to help myself I created some helper functions that I thought were useful. Some of these I created before starting while others I created along the way. The first two I created were rand and randint. Rand takes in two floats and returns a random float between the inputs, while randint takes in two ints and returns an integer. These functions are useful to make attacks and decays a little more natural during layered notes. When using arrays, randint will choose an index corresponding to notes and note lengths.

float rand(float a, float b) {

float x = al::rnd::uniform(a,b) ;

return x ;

}

One of my focuses was to be able to structure my code in a way that melodies would be able to repeat at different speeds. My goal was to create an equation that relates length to attack and decay dynamically based on a changing tempo. The two solutions that I use are sinQ and sinWhole. My idea here was that by passing in a length which could be a quarter note, half note, etc., the attack and decay would set itself automatically. The main difference between sinQ and sinWhole is the ability to have more control with sinWhole. By passing in small attack and decay values I am able to vary notes within large chords to prevent a large attack at the same time as well as change the character within a long held note. The main use of these functions occurs in the beginning of the piece where the first melody is broken up by chords. I wanted these chords to have a changing timbre and character. These functions are also useful because the tempo increase in between repetitions.

void sinWhole (Scheduler &s, float time, float freq,

float len, float a=0, float d=-0.5, float dt=1) {

float atk = len \* 0.25 + a;

float dcy = 1 + d;

s.add<SineEnv>( time ).set( len, freq, 0.1, atk \* dt, dcy \* dt);

}