Nicholas Klein

11/3/2022

CSE-471

Professor Tong

Additive Synthesizer

The additive synthesizer is a component that works by adding together sinusoids to produce a sound. Here, timbre is created by adding sine waves together. Timbre is considered to consist of multiple harmonic or inharmonic partials or overtones. Each of the partials are a sine wave of different amplitude and frequency that decay over time due to modulation from an ADSR envelope. Most directly, additive synthesis generates sound by adding the output of numerous sine wave generators together. So how was this implemented in C++? The following paragraph explains.

Two classes were added to our project to allow the functioning of the additive synthesizer. The first is CAdditiveSineWave. In this class, numerous private member variables were needed. Variables for the duration, frequency, amplitude, vibrato rate and frequency, and the index, were used. This class contains a Start() method which starts audio generation by assigning “audio” to the length of wave to be generated and calling GenerateWaveTable(). GenerateWaveTable() contains a loop to iterate as long as the wave with a nested for loop to build our “sample” local variable. We then add to our sine wave radians variable by using the vibrato rate in our radians2 formula ((2\*pi\*vibratoRate)/sample rate) and the same for the vibrato wave with radians 1 ((2\*pi\*(m\_freq+m\_vibratoFreq\*sin(radians2)))). Finally we store our sample in the audio short we created in Start(). The second class we created is called CAdditiveSynth. This class has private member variables for a CAdditiveSineWave object, duration, time, release, attack, cross fade in and out, sustain, and decay. Similar to the last class, this class also has a Start() method which simply assigns values to a couple member variables. The bulk of the class here is in Generate() and SetNote(). Generate() calls the AdditiveSineWave:Generate() method on a m\_sinewave object. It then reads the component sample and makes it the resulting frame and implements attack and release. Then the time is updated, and we return true until time reaches the duration. SetNote() takes in a pointer to a note object as a parameter. The method gets a list of all attribute nodes and the length of that list. The list of attributes is looped over. Each loop, we get the value of the attribute. A CComVariant loads the attribute value as a string in UNICOE, but we then change it to an integer (VT\_I4) or double (VT\_R8) using the ChangeType function. Finally, its integer or ddouble value is read from a member variable.