Audio Hockey Table

Chet Gnegy

November 6, 2013

1 Introduction

The purpose of this project is to create a real-time effects processor with a graphical interface that integrates the physics of real objects as well as the actions of the user into the parameter space of the signal chain. The basic design is as follows: There are a bunch of different modules that can be placed in a 2D plane, each of these modules has physical properties assigned to it and can impact the sound of some input source. The modules are represented by short cylinders, called Discs. The Discs are each associated with some unit generator, whether it be an input source or an audio effect. The proximity of the Discs to one another will determine the signal path as well as the mix level for the effects. The user, who can interact with the environment through clicking, can drag the Discs around the world, changing the parameter space. The Discs are bound to the laws of physics, however, and will bounce off of one another as well as with the walls of the environment. The physics engine will also compute friction and angular velocities, the latter of which could also be mapped to a parameter of the unit generators.

2 Audio Framework

This software utilizes the RtAudio[1] engine, which conveniently allows the system to communicate with the sound card by periodically filling buffers with audio data. To keep the system modular, the graphical display and the audio components are completely independent of each other and interact only through the parameter space of the Disc objects. The RtAudio callbacks are handled through by UGenChain class. UGenChain contains some data structure SPAGHETTI CAT!!of UnitGenerators that pass the audio signal from the input source to the next element in the chain and finally to the array designated as the output buffer.

2.1 Signal Path

The signal path, encompassed by the class UGenChain, is responsible for three main points: handling the audio stream, providing the next sample of audio data on request, and maintaining the order in which the signal chain is processed.

The first of these responsibilities involves only initializing the RtAudio engine and opening the audio stream. The provision of the next sample is fairly straightforward given that the chain of unit generators. We simply "tick", or compute one sample, of audio from each unit generator in order and pass the result from the next. We of course start with an input source, who simply returns a sample of the audio input. The effects units are similar but involve some calculation, often state-based depending on its parameters, and also on the samples that have been processed by it at previous times.

Finally, the signal chain is determined. SPAGHETTI CAT!!

2.2 Unit Generators

Several unit generators have been defined for this software. Many of them are loosely based around well known algorithms. The unit generators feature two main methods, "tick" and "set_params". As previously mentioned, "tick" processes a single sample of audio data. The method "set_params" allows for the parameters of the unit generator to be changed. This does not include the parameters that are determined by the position of the Discs, but some internal parameters to the unit generator. The "Input" unit generator is trivial and simply returns the current sample of audio from the input buffer.

2.2.1 Bit Crusher

The bit crusher effect is very handy for reproducing vintage low-fi audio effects. The audio signal, typical 16 bits in resolution, is quantized down to a specified level on the range of 1 to 16, 16 being the unquantized signal. By using a value of 8 bits, we can achieve "chip music" sounds that resemble those of older game systems. The bit crusher also features a downsampling parameter which effectively reduces the sampling rate of the signal. We can downsample by an integer number on the range 1 to 16, where we quickly experience the effects of aliasing as this parameter increases.

2.2.2 Chorus

The chorusing effect used is modeled as described in Jon Dattorro's paper on delay-line modulation[2]. The effect is achieved by summing the dry signal, $x(t)$, with a delayed copy of itself, $x(t-\tau(t, f_c, d_c))$, where $\tau(t, f_c, d_c) = d_c \sin(f_c t)$. The modulation rate, f_c is bounded on the range 0.02 - 10.0 Hz, and the depth of the effect,d is bounded by 0.0125. There is also a negative feedback path with a delay of $\tau(t, f_c, 0)$, corresponding to the average delay length of the feedforward path. The weighting of each of these paths is given by Dattorro's "white chorus".

2.2.3 Delay

This is a simple delay line. The time in seconds can be changed as well as the amount of feedback.

2.2.4 Distortion

SPAGHETTI CAT!!

2.2.5 Looper

The Looper waits for the user to trigger a start event and begins to count down from 4. It then records its input for a given number of beats at a specified tempo. Immediately after completing the recording, it begins to replay the buffer on repeat. Effects can be applied to the looper as if it is an input.

2.2.6 Reverb

This is an implementation of Freeverb$_{[3]}$ using feedback comb filters and all pass filters. The size of the simulated room and the damping can be altered in real time.

2.2.7 Tremolo

The tremolo module provides simple low frequency amplitude modulation with a sinusoidal carrier wave. The modulation rate, f_t is bounded on the range 0.02 - 10.0 Hz.
SPAGHETTI CAT!!fix the "quotes" with bold or something to signify that it is code SPAGHETTI CAT!!

[1] http://www.music.mcgill.ca/ gary/rtaudio/ [2] Jon Dattorro - Part 2: Delay-Line Modulation and Chorus https://ccrma.stanford.edu/ da
torro/EffectDesignPart2.pdf [3]https://ccrma.stanford.edu/ jos/pasp/Freeverb.html