CodePool

a system for collaborative network music

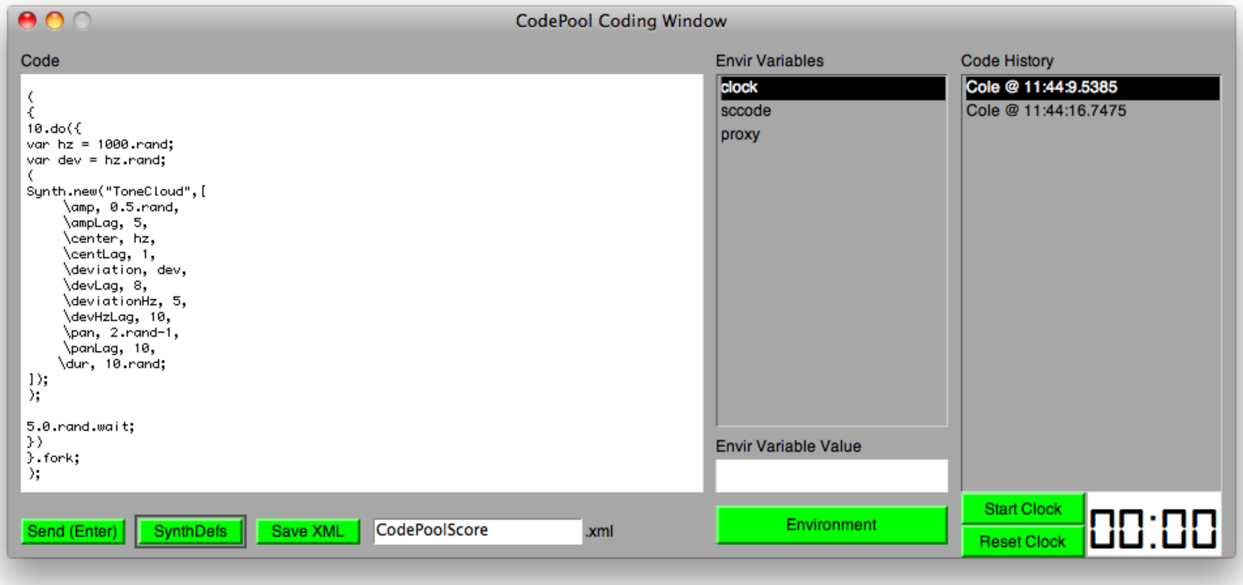
developed by Cole Ingraham

Introduction

CodePool is a work in progress designed by Cole Ingraham to facilitate collaborative coding based network music. Its features are designed to 1) synchronize a group of users over a network, 2) keep a record of all events executed with the system, and 3) provide various utilities to enhance performances involving live coding. The system is currently designed in and for the SuperCollider audio language and is stable on Mac OSX and somewhat supported on Linux and Windows. CodePool was premiered as a live structured improvisation by the LAG laptop quartet (Cole Ingraham, Benjamin O'Brien, Chad McKinney, and Curtis McKinney) as part of the 10/29/2009 All Music Alliance concert at University of the Pacific in Stockton, CA.

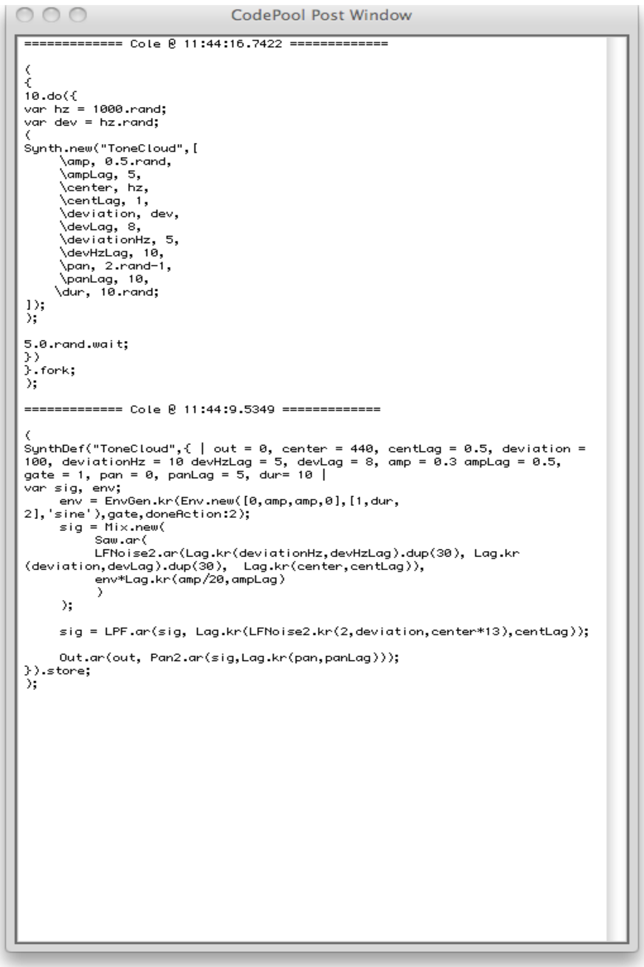
The CodePool GUI

The primary front end to CodePool is divided into two windows: the Coding window and the Post window. The Coding window has many parts as you can see below:



```
(
{
10.do({
var hz = 1000.rand;
var dev = hz.rand;
(
Synth.new("ToneCloud",[
      \amp, 0.5.rand,
      \ampLag, 5,
      \center, hz,
      \centLag, 1,
      \deviation, dev,
      \devLag, 8,
      \deviationHz, 5,
      \devHzLag, 10,
      \pan, 2.rand-1,
      \panLag, 10,
     \dur, 10.rand;
]);
);

5.0.rand.wait;
})
}.fork;
);
```

The Code area is a large text entry panel where the user enters any code they wish to send to the rest of the group by pressing the Send button. Once the code is sent a new entry is made in the Code History list on the right. This lists the time stamp and sender name of all code sent in this way. The user can then click on an entry in the list to see that block of code in the Code area where is can then be modified and resent. If the user simply wishes to resend the exact same code, they can click on the list entry and press enter. This will also create a new entry with a new time stamp. To the left of the History list is the Envir Variable list. Any time a user declares an environmental variable it will appear in this list. Selecting a variable will display its value in the area below the list. The Post window is a text area where all code sent from the Code window is posted along with the sender's name and time stamp:

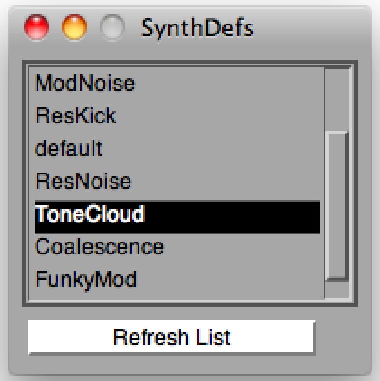


```
============= Cole @ 11:44:16.7422 =============

(
{
10.do({
var hz = 1000.rand;
var dev = hz.rand;
(
Synth.new("ToneCloud",[
     \amp, 0.5.rand,
     \ampLag, 5,
     \center, hz,
     \centLag, 1,
     \deviation, dev,
     \devLag, 8,
     \deviationHz, 5,
     \devHzLag, 10,
     \pan, 2.rand-1,
     \panLag, 10,
    \dur, 10.rand;
]);
);

5.0.rand.wait;
})
}.fork;
);

============= Cole @ 11:44:9.5349 =============

(
SynthDef("ToneCloud",{ | out = 0, center = 440, centLag = 0.5, deviation =
100, deviationHz = 10 devHzLag = 5, devLag = 8, amp = 0.3 ampLag = 0.5,
gate = 1, pan = 0, panLag = 5, dur= 10 |
var sig, env;
     env = EnvGen.kr(Env.new([0,amp,amp,0],[1,dur,
2],'sine'),gate,doneAction:2);
     sig = Mix.new(
          Saw.ar(
          LFNoise2.ar(Lag.kr(deviationHz,devHzLag).dup(30), Lag.kr
(deviation,devLag).dup(30),  Lag.kr(center,centLag)),
          env*Lag.kr(amp/20,ampLag)
           )
      );

     sig = LPF.ar(sig, Lag.kr(LFNoise2.kr(2,deviation,center*13),centLag));

     Out.ar(out, Pan2.ar(sig,Lag.kr(pan,panLag)));
}).store;
);
```

This is very similar to the History list. Its main function is to allow users to quickly see new events without having to scroll through the history list to know what happened recently.

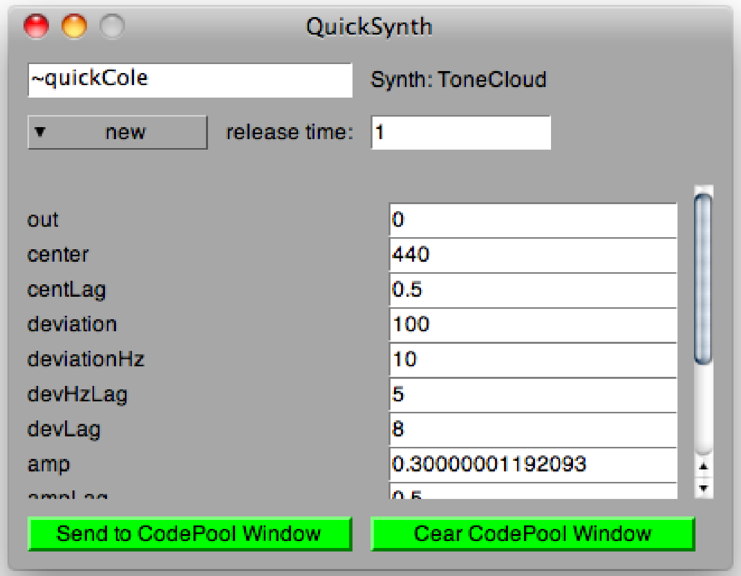
There are various components along the bottom of the Code window. On the right is a digital timer object created by Chad McKinney. Next to it are controls to start, stop and reset the clock. The control buttons have been configured to perform their actions on the clocks of everyone in the group, ensuring that they all stay synchronized. To the left of that is a large button called Environment which is used to to switch between the default environment and ProxySpace in SuperCollider. Once again clicking this button performs that action for everyone in the group. Next there is a text box and a button that say Save XML. This feature allows you to save the contents of the History list to an xml file with the name typed in the text box. By saving the History you essentially save the entire performance as everything done using CodePool appears in the History. This is useful for reviewing events and even recreating exact performances through creating an automated playback program.

The SynthDefs button brings up a list of all SynthDefs (synthesized sounds) which have been compiled through CodePool:



ModNoise
ResKick
default
ResNoise
ToneCloud
Coalescence
FunkyMod

By selecting a name from the list and pressing enter, a QuickSynth window is made which allows the user to quickly generate code to start, modify, or stop that synth:



~quickCole

Synth: ToneCloud

new

release time: 1

out	0
center	440
centLag	0.5
deviation	100
deviationHz	10
devHzLag	5
devLag	8
amp	0.30000001192093

The text box at the top specifies the variable name which the new synth will be assigned to. Next to that is the name of the synth you selected. Below that is a drop down menu where you can select the action (start, modify, or stop) the synth. If you choose to stop the synth you can specify its fade out time in the box marked release time. For starting or modifying synths, a list of all available parameters and their default values is provided. The bottom left button will generate the code based on these parameters and place it in the Code window without sending it to allow the user to make any further modifications by hand before broadcasting it to the group. The righthand button will simply clear the Code window and is just a convenience feature.

The Back End

CodePool makes use of OscGroups, an open source client-server architecture developed by Ross Bencina, to allow users to network together while behind firewalls and separate routers using NAT traversal. All network communication is then done through Open Sound Control (OSC), a UDP based network protocol developed by CNMAT at UC Berkeley, allowing for simple message formatting and flexibility. When a user starts CodePool it begins to listen for incoming OSC messages with a specific name, which is sent by all CodePool users when they click Send, and immediately runs the contents of the message on the receiver's computer. This means that not only can everyone see the code everyone else has been sending, they can also see/hear the what it does.

The broadcast and compile nature of CodePool allows for much more than simply keeping track of everyone's actions. Since anything sent from one person immediately runs on everyone's machine, groups can (and have) rehearse remotely in real time. As the system (currently) no audio signal, only code, there is practically no latency, save that of slow internet connections. In addition to rehearsing, it is also possible for the a single live performance to be to he experienced in real time from any computer anywhere that is running CodePool on the correct OscGroup without the bandwidth required by streaming live audio.

Future Goals

In order to facilitate greater cross platform compatibility, CodePool is being concurrently developed as a stand alone Java application. This version will take over the network handling and GUI components but not the audio production. It will initially still use SuperCollider as its sound engine, however it is planned to additionally interface with other audio languages such as CSound, ChucK, JSYN, and possibly graphical environments such as MAX/MSP and Pure Data. Additional utilities for rapid code generation (similar to QuickSynth) are planned for various commonly used code patterns as well.