Introduction

# Computer Music Practice Examples

# SIOE

Supercut Remix Generator

## Step-by-step process

Joo Won Park
www.joowonpark.net/cmpe

## Index

**Step 0.  Goal**

**Step 1.  Make S(iAllone)**

**Step 2.  Make L(Harmony)**

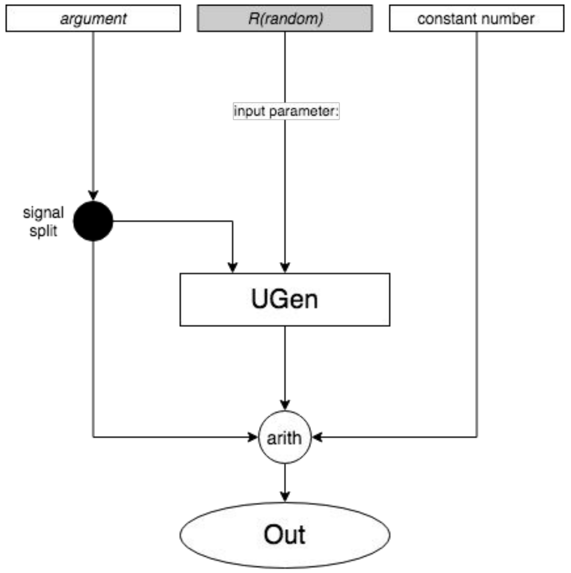
**Step 3.  Make F(~sioe)**

**Step 4.  Make GUI**

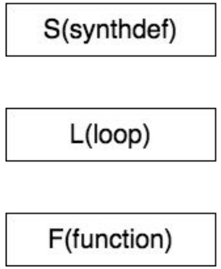
Link to Music and App

Link to .scd files

## Block Diagram Legend



UGen:
Unite Generator.
Processes audio or data
arrow:
Shows the direction of signal
The text in the line (input parameter:) shows the name of the input parameter of an input in the connected UGen
argument:
Controllerable arguments
Written in italics
Can be a list in [arg,arg,arg] format
constant number:
Discrete numeric value
signal splitter:
Used when one signal is connected to multiple inputs
Arith:
Arithmetics.
Incoming signals are added(+), subtracted(-), multiplied(x), or divided(/).
Out:
Audible audio output



S(synthdef) : SynthDef. Includes OSCFunc

L(loop): loops including do{} and Routine

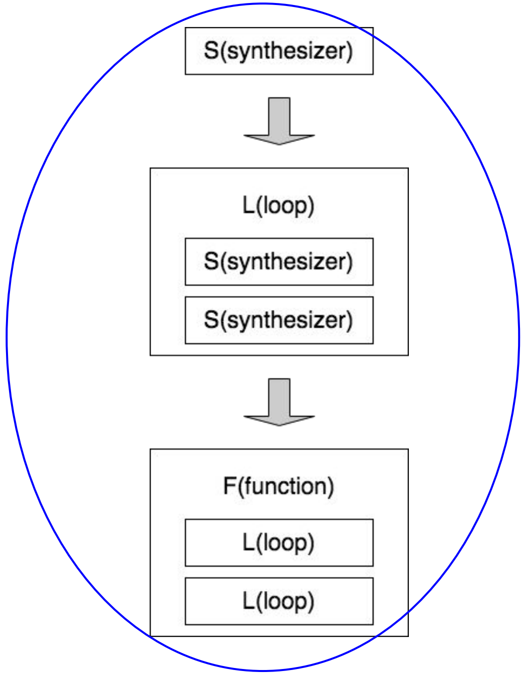
F(function): custom function.

CMPE - Introduction

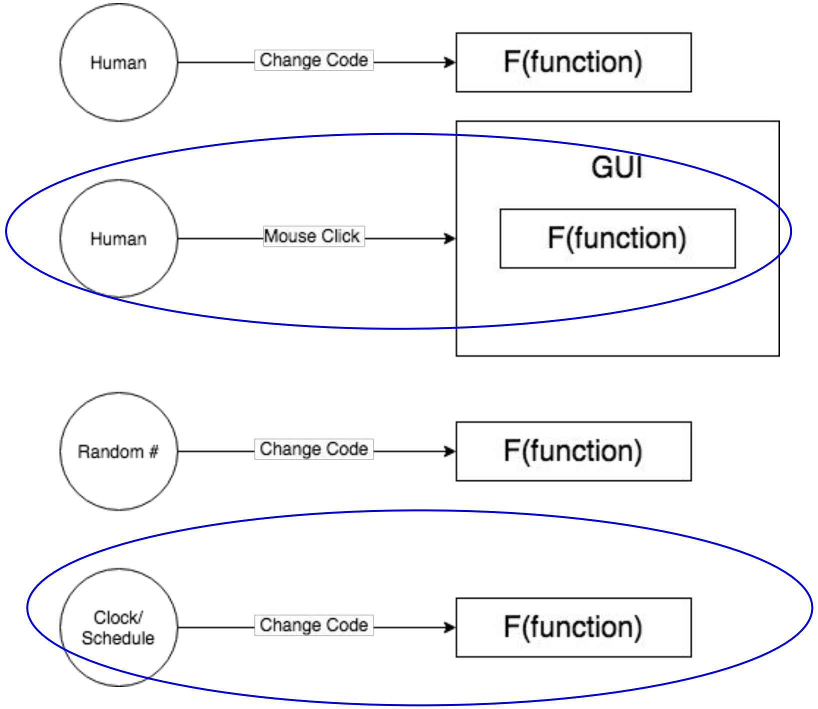
Main

# Design and Creative Process Overview

Instrument Design Process



Presentation/Performance Methods



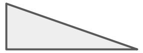
Main

## Step 0.  Goal

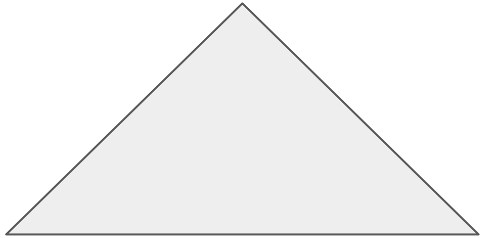
Make a program that cuts and juxtaposes a song into a single percussion sample

Original Sound

**Short Soud**



**Long Sound**



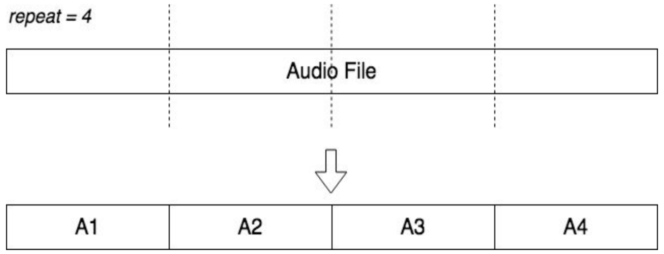
Process : automation/batch processing

1. Equally divide an audio file into [x] number of parts
2. Juxtapose the sections and put amplitude envelopes with controllable [length]
3. For the sequenced supercut, [schedule] the progress according to a controllable BPM

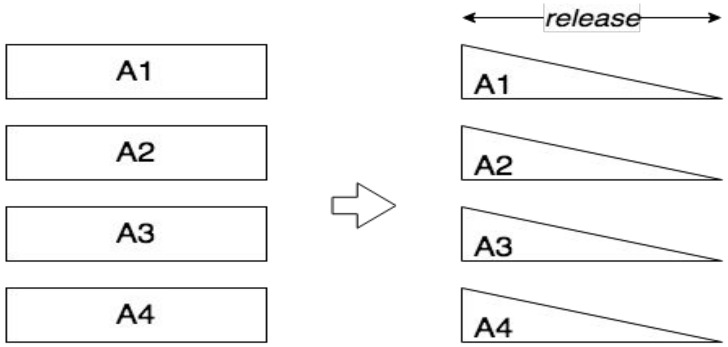
Main

## Step 0. Goal

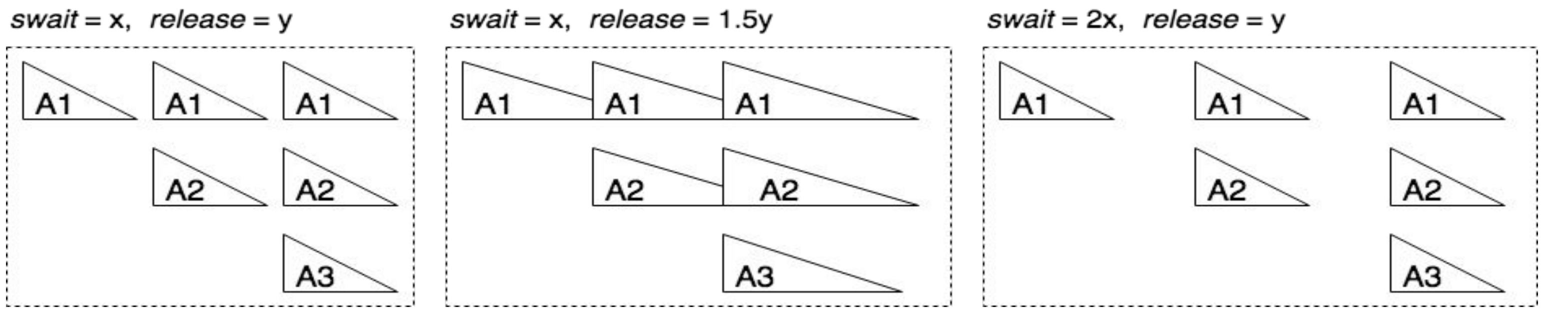
1. Equally divide an audio file into [*repeat*] parts



2. Juxtapose the sections and put amplitude envelopes with controllable [*release*]



3. For the sequenced supercut, [*swait*] the progress according to a controllable BPM

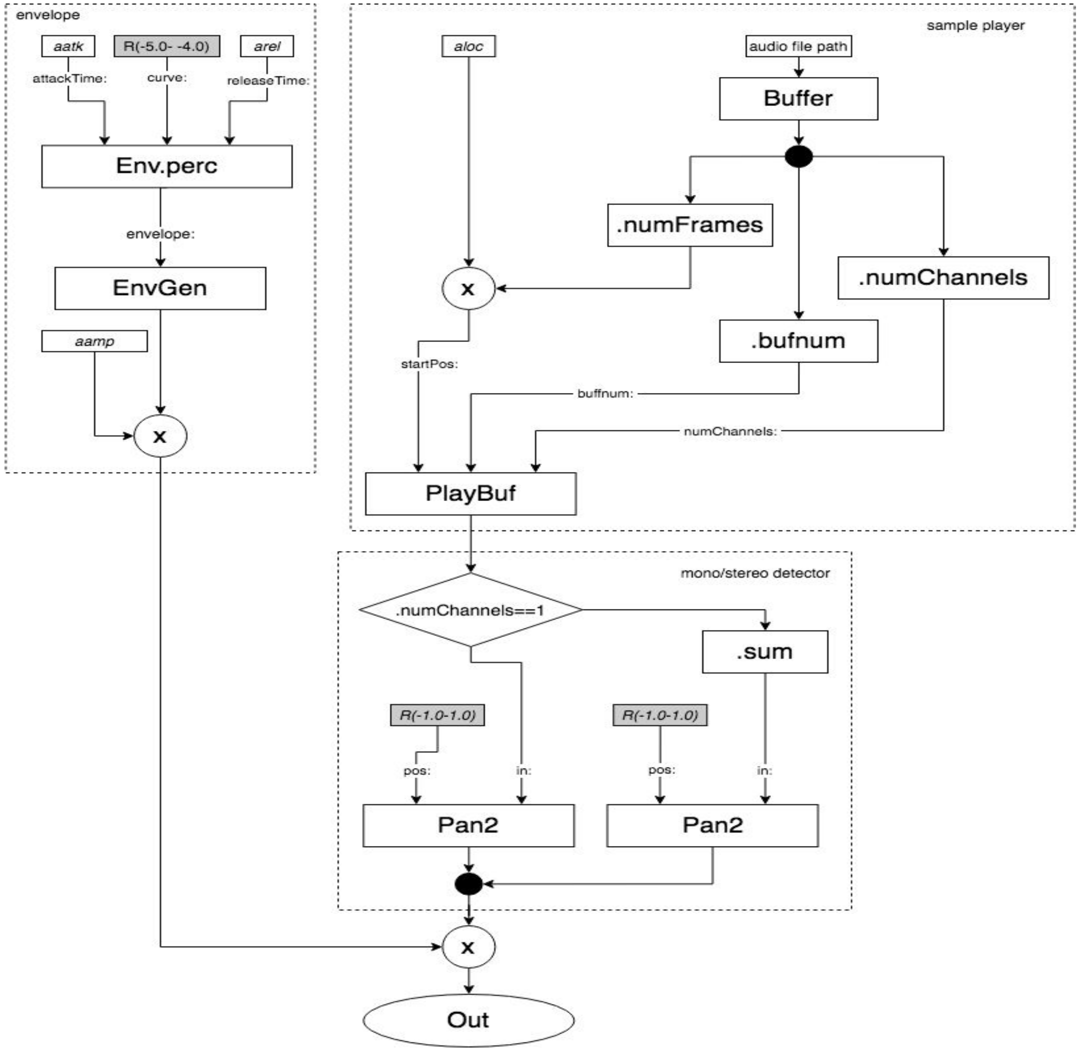


Main

## Step 1. Make S(iAllOne)

- Make a sample player with a controllable starting position
- Make an amplitude envelope
- Make a stereo/mono detector

S(iAllOne) is a simpler version of the instrument used in ISJS



## Step 1. Make S(iAllone)

- Make a sample player with a controllable starting position
- Make an amplitude envelope
- Make a stereo/mono detector

SIOE-Step1.scd

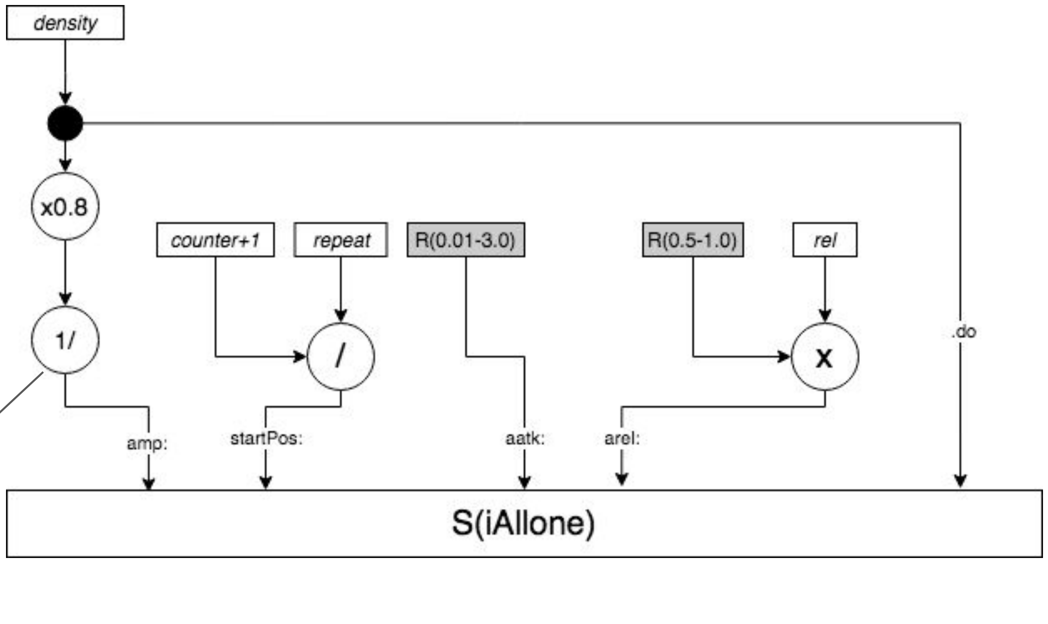
```
SynthDef("iAllone",{
	arg aamp,aatk,arel,aloc;
	var env,loc,sound,mix;

	//envelope
	env=Env.perc(aatk,arel,curve:Rand(-5.0,-4.0));
	env=EnvGen.kr(env,doneAction:2);

	//sample player
	loc=~buff.numFrames*aloc;
	sound=PlayBuf.ar(~buff.numChannels,~buff.bufnum,1,1,loc,1);

	//mono-stereo detector
	if(~buff.numChannels==1,
		{mix=Pan2.ar(sound,Rand(-1.0,1.0));},
		{mix=Pan2.ar(sound.sum,Rand(-1.0,1.0));}
	);

	Out.ar(0,mix*0.9*aamp*env)
}).add;

Synth("iAllone",[\aamp,1,\aatk,0.4,\arel,3,\aloc,0.1]);
```

Main

## L(Harmony)

### Step 2. Make L(Harmony)

- Using S(iAllone), make a loop that creates a one-shot supercut sample
- Adjust the overall volume according to the number of repeats



| Total Number of Notes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Amplitude of Each Note | 1.25 | 0.63 | 0.42 | 0.31 | 0.25 | 0.21 | 0.18 | 0.16 |
| Resulting Total Amplitude | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 |

## SIOE-Step2.scd

### Step 2. Make L(Harmony)

- Using S(iAllone), make a loop that creates a one-shot supercut sample
- Adjust the overall volume according to the number of repeats

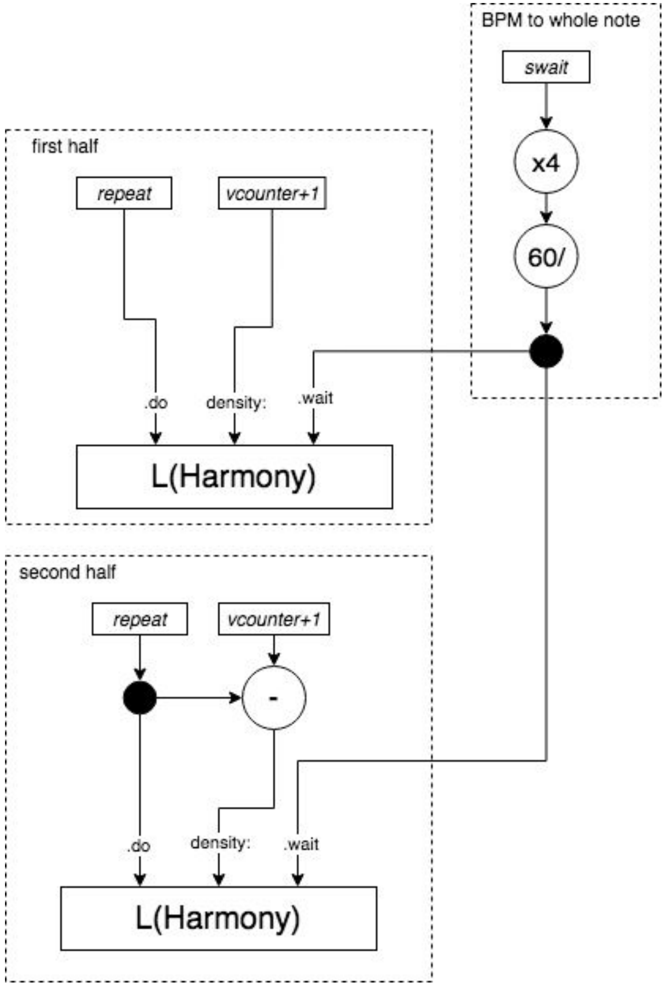
```
~oneShot={
        arg density,repeat,release;
        density.do{
                arg counter;
                var startpos;
                startpos=(counter+1)/repeat;
                //automatic volume control & automatic "splicer"
                Synth("iAllone",[\aamp,1/(density*0.8),\aatk,rrand(0.01,0.3),\arel,rrand(release*
                0.5,release),\aloc,startpos]);
        };
};
//arg density, repeat, release
~oneShot.value(10,5,2);

~oneShotTimed={
        arg density,repeat,release;
        Routine({
                density.do{
                        arg counter;
                        var sartpos;
                        startpos=(counter+1)/repeat;
                        Synth("iAllone",[\aamp,1/(density*0.8),\aatk,rrand(0.01,0.3),\arel,rrand(
                        release*0.5,release),\aloc,startpos]);
                        1.wait;
                };
        }).play;
};

~oneShotTimed.value(10,10,2);
```

Main

## Step 3. Make F(~sioe)

- Make two copies of L(Harmony) with different counter to make a symmetric form
- Make a BPM to wait time converter

F(~sioe)



Main

## SIOE-Step3.scd (1/2)

**Step 3. Make F(~sioe)**

- Make two copies of L(Harmony) with different counter to make a symmetric form
- Make a BPM to wait time converter

```
~sioe={arg repeat,swait,release;
        var beat;

        //BPM to waittime converter
        beat=60/swait*4;

        ~routprocess=Routine({
                "first half".postln;
                repeat.do{
                        arg vcounter;
                        var density;
                        density=vcounter+1;
                        density.do{
                                arg counter;
                                var startpos;
                                startpos=(counter+1)/repeat;
                                Synth("iAllone",[\aamp,1/(density*0.8),\aatk,rrand(0.01,0.3),\arel,r
                                rand(release*0.5,release),\aloc,startpos]);
                        };
                        ("# of voices per measure: "++density).postln;
                        beat.wait;
                };
```

Main

## SIOE-Step3.scd (2/2)

**Step 3.  Make F(~sioe)**

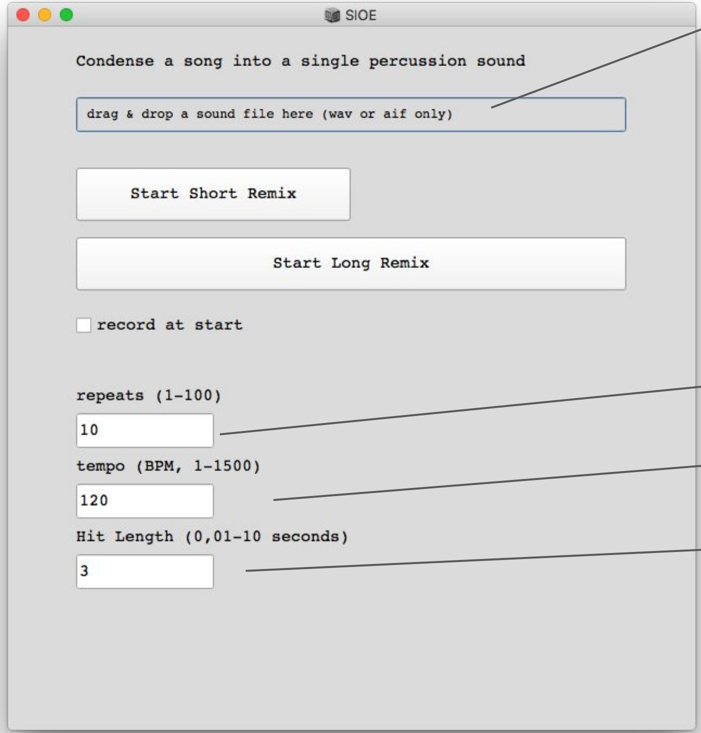
- Make two copies of L(Harmony) with different counter to make a symmetric form
- Make a BPM to wait time converter

```
			"second half".postln;
			repeat.do{
				arg vcounter;
				var density;
				//forward=vcounter+1;
				density=(repeat-(vcounter+1));
				density.do{
					arg counter;
					var startpos;
					startpos=(counter+1)/repeat;
					Synth("iAllone",[\aamp,1/(density*0.8),\aatk,rrand(0.01,0.3),\arel,r
					rand(release*0.5,release),\aloc,startpos]);
				};
				("# of voices per measure: "++density).postln;
				beat.wait;
			}
		});
		~routprocess.reset;~routprocess.play;
};

//arg repeat, swait (in BPM), release
~sioe.value(10,150,2);
```

Main

## Step 4. Make a GUI

- Make an interface to drag-drop audio file and control parameters in ~sioe
- GUI-specific feature: dynamic drag & drop



SIOE

Condense a song into a single percussion sound

drag & drop a sound file here (wav or aif only)

Start Short Remix

Start Long Remix

☐ record at start

repeats (1-100)

10

tempo (BPM, 1-1500)

120

Hit Length (0,01-10 seconds)

3

Drag and drop audio file

```
~sioe.value(
    repeat,
    swait,
    release
);
```

## Inspirations

*Barely* from Plunderphonics 69/96

By John Oswald

Lunch Doodle with Mo Willems

## What's Before & Next

ISJS: Granular Processor

RMHS: Ambient Sound Generator

Overundertone

Contact joowon@joowonpark.net if you have questions or see errors.

I S
J S

RMHS