

## YASC

<u>GitHub</u>

By Yan Zhuang

#### Introduction

 The goal of the project is to develop an interactive recorder performance tool, incorporating Joy-Con controllers via a web interface for gesture-based inputs and SuperCollider for sound synthesis.



#### SuperCollider

- It's an engine for sound synthesis and algorithmic music composition.
- It's a real-time sound-based
   OOP language.
- https://supercollider.github.io

```
recorder.scd (D:/work/YASC/supercollider) - SuperCollider IDE
                                                                                      File Session Edit View Language Server Help
                                                                                                           YASC
                                                                    🗶 🖪 Post window Auto Scroll
                                                                       Device options:
 3 s.clear;
                                                                         - MME : Microsoft 声音
                                                                                                          AMPLITUDE ENVELOPE
 5 Pdef.all.stop;
 6 Window.closeAll;
                                                                                                               Release (s)
10 SynthDef(\waveguideFlute, {
                                                                                                                0.200
       arg scl = 0.2, freq = 440, ipress = 0.9, ibreath = 0.09,
  ifeedbk1 = 0.4, ifeedbk2 = 0.4, dur = 20000, gate = 1, amp = 1, out = 0, atk = 0.1, rel = 0.2, vdep = 0.3, vfr = 5;
                                                                                                           VIBRATO CONTROLS
       var kenv1, kenv2, kenvibr, kvibr, sr, cr, block;
       var poly, signalOut, ifqc;
       var aflow1, asum1, asum2, afgc, atemp1, ax, apoly, asum3,
                                                                          - Windows WDM-KS
   avalue, atemp2, aflute1;
                                                                          - Windows WDM-KS
       var fdbckArray:
                                                                          - Windows WDM-KS : He
       sr = SampleRate.ir;
                                                                         - Windows WDM-KS : He
      cr = ControlRate.ir;
                                                                          - Windows WDM-KS :
      block = cr.reciprocal;
                                                                                                           REVERB CONTROLS
                                                                        Requested devices:
       kenv1 = EnvGen.kr(Env.new([ 0.0, 1.1 * ipress, ipress,
   ipress, 0.0], [ 0.06, 0.2, dur - 0.46, 0.2], 'linear'),
                                                                         - (default)
                                                                         Out:
       kenv2 = EnvGen.kr(Env.new([ 0.0, 1.0, 1.0, 0.0 ], [ atk,
                                                                         - (default)
  dur - 0.02, rel ], 'linear' ), gate, doneAction: 2);
                                                                                                               Time (s)
       kenvibr = EnvGen.kr(Env.new( [ 0.0, 0.0, 1.0, 1.0, 0.0 ],
                                                                        Selecting default syste
   [ 0.5, 0.5, dur - 1.5, 0.5 ], 'linear'), gate);
       aflow1 = LFClipNoise.ar( sr, kenv1 );
                                                                        Booting with:
       kvibr = SinOsc.ar(vfr, 0, 0.1 * kenvibr * vdep );
                                                                         In: MME: 麦克风阵列 (
       asum1 = ( ibreath * aflow1 ) + kenv1 + kvibr;
                                                                         Out: MME: 扬声器 (Rea
       afgc = ifgc.reciprocal - ( asum1/20000 ) - ( 9/sr ) + (
                                                                         Sample rate: 44100.00
  ifqc/12000000 ) - block;
                                                                         Latency (in/out): 0.6
       fdbckArray = LocalIn.ar( 1 );
                                                                        SC AudioDriver: sample
       aflute1 = fdbckArray;
                                                                        SuperCollider 3 server
       asum2 = asum1 + ( aflute1 * ifeedbk1 );
                                                                       Requested notification
      ax = DelayC.ar( asum2, ifqc.reciprocal - block * 0.5, afqc
                                                                        localhost: server proce
  * 0.5 - (asum1/ifqc/cr) + 0.001);
                                                                        localhost: keeping clie
      apoly = ax - (ax.cubed);
                                                                        Shared memory server ir
       asum3 = apoly + ( aflute1 * ifeedbk2 );
                                                                        server 'localhost' disc
       avalue = LPF.ar( asum3, 2000 );
                                                                       Shared memory server in
      aflute1 = DelayC.ar( avalue, ifqc.reciprocal - block, afqc
       fdbckArray = [ aflute1 ];
       LocalOut.ar( fdbckArray );
       signalOut = avalue
                  Interpreter: Active Server: 1.078
```

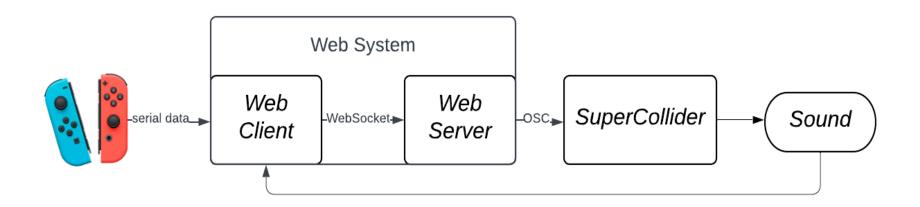
### Joy-Con controller

- Primary controllers for the Nintendo Switch gaming console.
- We could use that to play music.



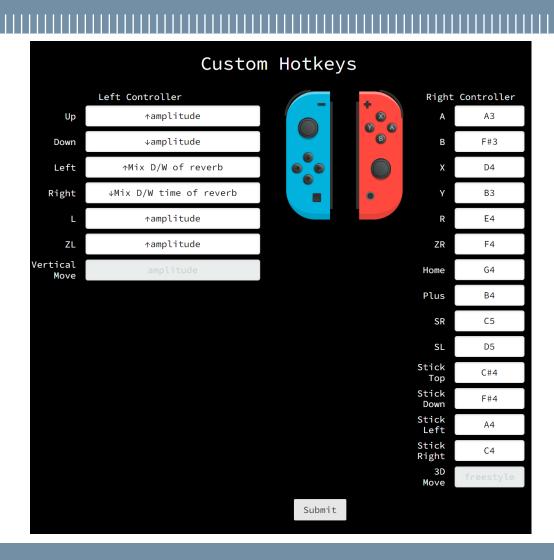
#### Structure overview

- User interaction accomplished with Joy-Con controllers.
- Web system acts as main interface.
- SuperCollider serves as sound synthesis engine, generating/editing sounds according to the web system commands sent via OSC protocol.
- Sound captured by the web system to provide visual feedback.



### Web system – Map your controller

- Through this page, Joy-Con controller can be mapped to different functions.
- The left controller is responsible for adjusting parameters in SuperCollider.
- The right controller is dedicated to play notes.



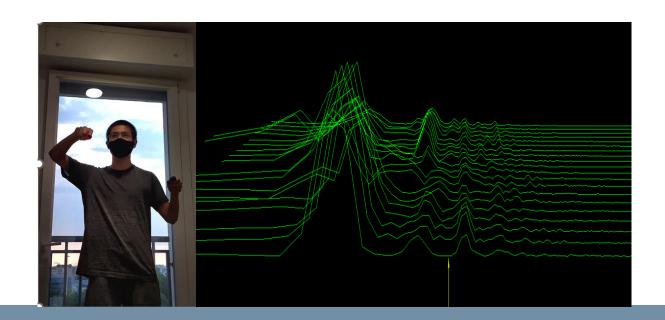
### Web system – Train your memory

- The dino game page is an interactive training tool designed to help users familiarize with their custom hotkeys.
- If you want to jump, press the corresponding hotkey.



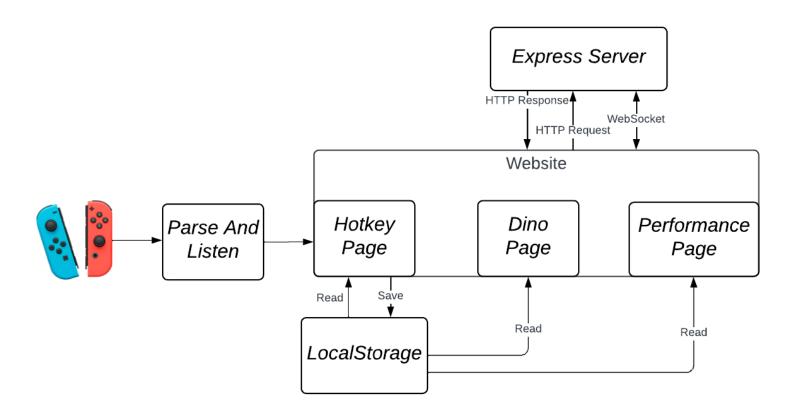
#### Web system – Play recorder

 The performance page provides an immersive audio-visual experience by showing the output sound spectrum.



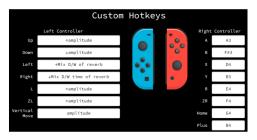
### Web system – Implementation

There are three main components which run the web system:



#### Web system – libraries

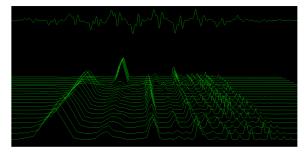
- 1. Vue3 for web interface.
- 2. joy-con-webhid for parsing joy-con controllers' input.
- 3. meyda for the sound analysis, three.js for rendering spectrum.





(a) Hotkey page detail

(b) Dino game detail



(c) Performance page detail

### Sound synthesis – GUI

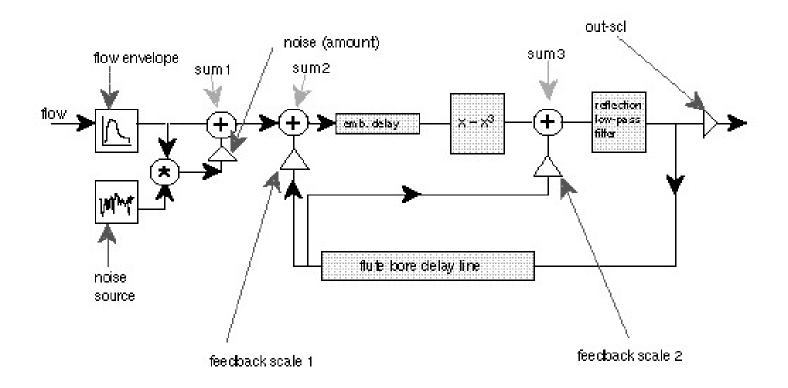
The recorder sound produced in SuperCollider. Users can see the changes made with controllers of some parameters:

- Amplitude envelope attack and release time.
- Vibrato frequency and depth.
- Reverb D/W mix, time and pre-delay.



### Sound synthesis – Implementation

#### Perry Cook's recorder model



#### Future work

- •Implement features that allow users to record, save, and replay their performances.
- •The waveguide model doesn't behave well at high frequencies. In the short future, multi-modal or multi-dimensional models could be adopted to solve the issue.
- •Offer more sophisticated sound parameters for users to control in SuperCollider.

# Thanks