

Final Project Notes

Conor Holden

November 15, 2020

Contents

1	General Information	1
1.1	Description	1
1.2	Tools	1
1.3	Videos about Iplug2	2
1.4	Tutorials about Juce	2
2	Digital Signal Processing	2
2.1	Fast Fourier Transform Algorithm	2
2.2	Waves	2
2.3	Wave Shaping	2
2.4	Convolution	3
2.5	Keyboard State	3
3	Instruments	3
3.1	Instrument Ideas	3

1 General Information

[Project Link](#)

1.1 Description

Fiddle about with trying to develop a realistic simulation for uilleann pipes or violins, or some unique instrumental sound of your own imagination.

1.2 Tools

- [iPlug2 GitHub](#) ⇒ For Creating both plug-ins and stand-alone
- [iPlug2 Wiki](#)
- [Juce](#) ⇒ More mature and has more tutorials

- [ACM Digital Library](#)
- [Physical Modelling](#)

1.3 Videos about Iplug2

Abandoned

- [Oliver Larkin: Faust in iPlug 2](#)
- [iPlug2: Desktop Plug-in Framework Meets Web Audio Modules by Oliver Larkin](#)

1.4 Tutorials about Juce

- [Juce String Model](#)
- [Maximilian Sound Library](#)

2 Digital Signal Processing

- [Juce DSP](#)
- [Digital Signal Processing \(DSP\) Tutorial](#)

2.1 Fast Fourier Transform Algorithm

Faster version of the Discrete Fourier transform.

- Transforms waves into its components or formula
- The inverse can be used to create sound waves from

2.2 Waves

- Sin Wave \Rightarrow `std::sin (x)`
- Saw Tooth \Rightarrow map $-\pi - \pi$ to $-1 - 1$ (`juce::MathConstants<double>::pi`)
- Triangle \Rightarrow map $-\pi - 0$ to $-1 - 1$ and $0 - \pi$ to $1 - -1$

2.3 Wave Shaping

- [dsp::WaveShaper](#)

Transforming one signal into another using a transfer function.

- $\sin(x)$ can be converted to a square wave using signum transfer function $\text{sgn}(\sin(x))$
- This creates a too perfect function and thus we use a hyperbolic tangent transfer function $\tanh(\sin(x))$

- To create a square, boost the signal into clipping before using the function $\tanh(n * \sin(x))$

2.4 Convolution

- [dsp::Convolution](#)

Simulating the reverberation characteristics of a certain space by using a **pre-recorded impulse** response that describes the properties of the space in question. This process allows us to apply any type of acoustic profile to an incoming signal by convolving, **essentially multiplying every sample of data against the impulse response samples** to create the combined output.

2.5 Keyboard State

- [MidiKeyboardState](#)

Used by the keyboard component to get midi input. Hosted in the plugin processor.

```
// gets midi buffer from keyboard state and inserts it into current
// buffer
keyboardState.processNextMidiBuffer(midiMessages, 0,
    buffer.getNumSamples(), true);
```

3 Instruments

3.1 Instrument Ideas

- Uilleann Pipes
- Bodhrán
- tin whistle

There is a tonne of information on string instruments as a back up.