FabDist

FabFour Group:
Amico Stefano Antonio, 10937333
Anand Abhijeet, 10859656
Pletti Manfredi, 10493741
Portentoso Alice, 10664207

April 21, 2023

1 Introduction

This project aims to create an audio distortion box for teaching and learning purpose using the SuperCollider programming language. The synthesizer is designed to produce one or two sine waves and apply various distortion techniques to the generated sound. The project also features a graphical user interface (GUI) for controlling various parameters such as input gain, output gain, distortion type, oversampling, filter cutoff and frequency.
The teaching GUI permits to explore the various types of distortion along with time/frequency graphs and transfer function definition.
The learning GUI is organized as a score quiz where the student has to recognized the distortion type from time/frequency representation.

2 Code Implementation

2.1 Creating Curves

The code defines seven different distortion curves, which are used to create various distortion effects:

- Identity curve: $f(x) = x$
- Soft clip curve: $f(x) = \tanh(2x)$
  Unlike hard clipping, soft clipping does not apply the threshold abruptly. The transition between the un-clipped and the clipped region is more smooth, which results in a less harsh sound.
- Soft clip asymmetric curve: $f(x) = \begin{cases} \tanh(2x) & x \geq 0 \\ \tanh(4x) & x < 0 \end{cases}$

Soft clipping asymmetrical apply condition to processing: if input value x is greater than or equal to zero, the function applies the hyperbolic tangent function tanh to the input value multiplied by 2. If x is less than zero, the function applies the hyperbolic tangent function to the input value multiplied by 4.

- Hard clip curve: $f(x) = \text{sign}(x) \cdot \min(2 \cdot |x|, 1)$
  Hard clipping is achieved by setting a threshold, upon which which the amplitude of the waveform is abruptly clipped. The threshold is symmetrical, since it is equally applied to the negative and positive part of the waveform. The sharp corners of the resulting waveform produce many harmonics and a very rough sound.

- Hard clip asymmetric curve: $f(x) = \begin{cases} \min(2 \cdot |x|, 1) & x \geq 0 \\ -1 \cdot \min(2.5, |x|, 1) & x < 0 \end{cases}$
  Hard clipping asymmetrical apply condition to processing: if input value x is greater than or equal to zero, the function applies the min function to the absolute value of the input value multiplied by 2 and the value 1. If x is less than zero, the function applies the min function to the absolute value of the input value multiplied by 2.5 and the value -1.

- Half rectification curve: $f(x) = \max(x, 0)$
  Half wave rectification is achieved by canceling any negative half wave, setting to zero any negative sample value. Both full wave and half wave rectification produce output signals that are no longer zero-mean.

- Full rectification curve: $f(x) = |x|$
  Full wave rectification is achieved by reflecting the sound wave over the x axis. This corresponds to making all negative sample values positive.

2.2 Synth Definitions

Two SynthDefs are defined in the code:

- sine: A sine wave generator with control parameters for frequency and input gain. The output is sent to an audio bus.

- shaper: A shaper that applies distortion to the input signal from the audio bus. It provides control parameters for curve selection, oversampling, filter cutoff, and output gain.

```
SynthDef.new("sine", {|freq1 = 440, ampin1 = 0.5, ampout1 = 0.001, bus = 0|
    Out.ar(bus, SinOsc.ar([freq1, freq1], 0, 1) * ampin1*2)
}).add;

SynthDef.new(\shaper, {|out = 0, in, curve, filterCutoff = 18000, gain = 1|

```

```
    inputSignal = In.ar(in, 2);
    distortedSignal = Shaper.ar(curve, inputSignal);
    filteredSignal = LPF.ar(distortedSignal, filterCutoff);

    Out.ar(out, filteredSignal * gain);
}).add;
```

Listing 1: Synth definition

2.3 GUI Implementation

The GUI includes several components to control the synthesizer:

- Start/Stop button: Toggles the sine wave generator on and off.
- Frequency knob: Controls the frequency of the sine wave.
- Input gain knob: Adjusts the input gain of the sine wave.
- Output gain knob: Adjusts the output gain after applying the distortion.
- Distortion type menu: Selects the distortion curve to be applied.
- Oversampling control menu: Sets the oversampling factor.
- Filter cutoff control menu: Sets the cutoff frequency of the low-pass filter applied to the distorted signal.

2.4 GUI Implementation - Learning part

The quiz GUI (learning part) tests the user's ability to identify different distortion types.

The quiz section includes:

- A static text box that displays the current question
- Six buttons that correspond to each possible answer choice
- A static text box that displays the result of the user's answer

The code also includes several functions for controlling and updating the GUI, including a function for generating a random distortion type:

```
// create random distortion type
~randomDistortionType = { |previousDistortionType|
    newDistortionType = rrand(1, 4);
  while ({newDistortionType == previousDistortionType}) {
        newDistortionType = rrand(1, 4);
    };

```

```
    curve = [softClipCurve, softClipCurveFunctionAsym, hardClipCurve,
    hardClipCurveFunctionAsym, halfRectCurve, fullRectCurve][newDistortionType - 1];
    shaper.set(\curve, curve);
    newDistortionType;
};
```

and a function for checking the user's answer and updating the score:

```
//checks if the user's answer is correct and updates the score
~score = 0;
~currentDistortionType = 0;

// display the result of the checkAnswer function
resultText = StaticText.new(window, Rect(150, 160, 300, 20));

~checkAnswer = { |answer|
    if (answer == ~currentDistortionType) {
        ~score = ~score + 1;
        resultText.string = "Correct! Your score is now " ++ ~score;
    } {
        resultText.string = "Incorrect. The correct answer was " ++ ~
    currentDistortionType;
    };
    ~nextQuestion.();
};

// move to the next question and update the question text
~nextQuestion = {
  ~currentDistortionType = ~randomDistortionType.(~currentDistortionType);
    questionText.string = "Question " ++ (~score + 1) ++ ": Which distortion type is
    this?";
};
```

3 Conclusion

This project demonstrates the creation of an audio distortion synthesizer using SuperCollider. The synthesizer generates one or two sine waves and applies various distortion techniques to achieve different sound effects. The user can control the synthesizer's parameters through a graphical user interface. The learning gui permits to understand the result of an applied distortion filter and the related transfer function. The project shows the potential for using SuperCollider to create custom audio processing tools, and future work could extend the synthesizer with more advanced features and additional distortion techniques. As an example the learning gui could be spread over an intranet class for students interaction implementing bridge codes like supercolliderjs and webRTCgui.