Problem Set 10: VST Plugin for Synthesizer and Effects

Please send back via NYU Brightspace
- A zip archive named as
  PS10_Last_First.zip
  This should contain:
  - Source code for Synthesizer plugin:
    - PluginProcessor.cpp
    - PluginProcessor.h
    - PluginEditor.cpp
    - PluginEditor.h
  - A screenshot of your Synthesizer plugin running with AudioPluginHost. You could also include a screenshot of your plugin running with your favorite DAW software.

Total points: 100
Points are awarded as follows:

Pass Test 1
- 20 Points – JUCE is correctly set up. Synth and DelayFx code is present.

Pass Test 2
- 20 Points – signal processing is working.

Pass Test 3
- 40 Points – all components are working.

General
- 20 Points - clear code, good comments, sensible formatting of code.

Task Overview
In this problem set you will take the C++ Synth class that you created in Problem Set 9 and use the JUCE framework to create a Synthesizer plugin. You will add to that delay effects code supplied by instructor. The parameters for both synthesizer and effects will be controlled by a Graphical User Interface (GUI) that you will create.

See the following JUCE tutorials, especially for how to program the GUI:
- Tutorial: Create a basic Audio/MIDI plugin, Part 1: Setting up
- Tutorial: Create a basic Audio/MIDI plugin, Part 2: Coding your plug-in
- Tutorial: The Slider Class
- Tutorial: Listeners and Broadcasters
- Tutorial: Radio buttons and checkboxes - JUCE: Class Index
- Tutorial: Plugin examples

Review how to use the JUCE tools

Review lecture L4-7-JUCE_VST_Plugin and the additional notes L4-7-JUCE_VST_plugin_howto.

Setup JUCE and create a new Plugin "Synth"

- Click on JUCE/projucer

This presents the "Create New Project" window

- In left panel, under Plug-In, select **Basic**
- In right panel, set Project Name to **Synth**
- Click on Create Project. Browse to where you want project to be created, e.g. C-Programming/PS10 and click the Open button.

This presents the "Synth-Projucer" window

- At the top of the left-hand panel, click on the "gear" icon
- In right panel, set Company Name to **NYU**
- In right panel, scroll down and under Plugin Formats, select VST3
- In right panel, under Plugin Characteristics, select **Plugin MIDI Input**
- Save the project parameters by selecting, at the top menu, File->Save

This results in the new folder "Synth" under C-Programming/PS10

Add PS09 Synthesizer code to PS10 project

Find the folder with the project source code. This should be Synth/Source. You will see these code files:

PluginEditor.cpp
PluginEditor.h
PluginProcessor.cpp
PluginProcessor.h

~~Find the folder with your PS09 source code, and copy these files to the Synth/Source folder:~~

~~synth.cpp~~
~~synth.h~~

We are not doing this because I had to change synth.h and synth.cpp to work in the plugin context. **Instead add the supplied synth.cpp and synth.h files**.

In the zip archive for this problem set, copy these files to the Synth/Source folder:

synth.cpp
synth.h
delay_fx.cpp
delay_fx.h

In the Synth window

- Select the File Explorer tab
- Select the "+" icon at the bottom of the File Explorer window
- Select "Add existing files"

Select the files

synth.cpp
synth.h
delay_fx.cpp

delay_fx.h

Note: you may have to specify the "Exporters". You should have one (or both) of:

- Xcode (maxOS)
- Visual Studio 2022

Formatted: Font: Not Bold

Formatted: Font: Not Bold

Formatted: Font: (Default) +Body (Cambria)

Test 1:

- Click on the XCode icon (or VS icon)

This will bring up XCode (or your VS)

- Click on the "Build" icon ("play" icon in XCode) to compile code

You may get warnings:

- Validate project settings

If so, then accept the recommendation to validate. Then Build again.
This should result in a "Succeeded" result (no errors). You have successfully compiled the Synth VST plugin. However, at this point the code doesn't do anything.

Create Synth VST Plugin
PluginProcessor.h
At top of this file, after
```
#include <JuceHeader.h>
```
Add:
```
#include "synth.h"
#include "delay_fx.h"
```

At the bottom of this file, just above the "private:" keyword, add these lines:
```
Synth synth;     /* declares Synth object */
DelayFx delay_fx;     /* declares DelayFx object */
```

PluginProcessor.cpp
At top of this file, after
```
#include "PluginProcessor.h"
#include "PluginEditor.h"
```
Add:
```
#include "synth.h"
#include "delay_fx.h"
```

In function
```
void SynthAudioProcessor::prepareToPlay (double sampleRate,
int samplesPerBlock)
```
Add code to initialize Class Synth:
```
     synth.init_synth(sampleRate);
```
and Class DelayFx:
```
     delay_fx.init_delay_effects(sampleRate);
```
This will set the sampling rate and other parameters.

In function
```
void SynthAudioProcessor::processBlock (AudioBuffer<float>&
buffer, MidiBuffer& midiMessages)
```

After the comment block
```
   // This is the place where you'd normally do the guts of your plugin's
   // audio processing...
   // Make sure to reset the state if your inner loop is processing
   // the samples and the outer loop is handling the channels.
   // Alternatively, you can process the samples with the channels
   // interleaved by keeping the same state.
```

Add the code from the provided file
`process_midi.cpp`
The code checks if there is a MIDI key input, converts the MIDI key code to a frequency, writes it to `synth.new_freq` and sets CMD to `CMD_ADD_KEY`. This is similar to what is done in the `main.cpp` Ncurses loop in PS09.

Add the code to synthesize a block of output samples:
```
/* synthesize a block of outputs */
float *synth_out = synth.synth_block(buffer.getNumSamples());
```

Since `Class Synth` has internal storage for its output
`float output[FRAMES_PER_BUFFER]);`
we only need to declare a pointer (`float *synth_out`) that will hold the pointer returned by `synth_block().`

Next add the code to pass synth output through the effects processor:
```
/* apply effects to a block of outputs */
float *fx_out = delay_fx.delay_effects(synth_out,
buffer.getNumSamples());
```
Similar to Class Synth, Class DelayFx has internal storage for its output.

Note: In a VST3 plugin, the input and output data are NOT interleaved. This is in contrast to PortAudio input and output data, which IS interleaved. Hence, for your VST code, synth_block() is called once and then copied to each channel's output buffer.

Next is the default code provided by ProJucer:
```
for (int channel = 0; channel < totalNumInputChannels; ++channel)
    {
        auto* channelData = buffer.getWritePointer (channel);

        // ..do something to the data...
    }
```

Replace the line

```
        // ..do something to the data...
```

With

- Another for() loop over all samples:

```
for (int sample = 0; sample < buffer.getNumSamples(); ++sample)
```

- In this loop body, copy every sample from the `fx_out[]` buffer to the (non-interleaved) `channelData[]` buffer. The for loop uses `buffer.getNumSamples()` to determine the number of samples in this loop. Multiply fx_out[] by synth.ampl in the loop:
  `channelData[sample] = synth.ampl * fx_out[sample];`

Test 2:
- Click on the XCode icon (or VS icon)

This will bring up XCode
- Click on the “Build” icon (“play” icon) to compile code

This should result in a “Succeeded” result (no errors)
- Don’t worry about “Deprecated” warnings.

Open JUCE Plug-in Host
Clear and re-load plugins: **NOTE: you must do this after every compilation of the synth plug-in!**
- Options->Edit list of available plug-ins…
- In the Available Plugins panel,
  - select Options->Clear List
  - select Options->Scan for new or updates VST3 plug-ins
- Close Available Plugins panel

In JUCE Plug-in Hose panel
- Right-click and load:
  - NYU->Synth
- Connect output red dot of Midi Input to input red dot of Synth
- Connect L and R output green dots of Synth to L and R input green dots of Audio Output.
- Click on any key of the piano keyboard graphic and note should play!

The basic Synth plug-in is working, but the audio effects are bypassed.

Programming the VST GUI

The GUI Panel will have
- A title “NYU Synthesizer”
- 2 sliders
- 5 radio buttons (i.e., only one of the five can be enabled at any one time)

The GUI panel could have (but does not need to have) a layout as shown at the end of this document.

GUI elements are declared in PluginEditor.h and defined in PluginEditor.cpp. All of the elements described in this section can be in the "Private" section of PluginEditor.h.

PluginEditor.h
Create two sliders for Synth

- Volume (which will be output level after all processing)
- Synth Sustain (less sustain is more rapid decay of the note)

This results in declarations in PluginEditor.h for slider changes and sliders like:

```
    void sliderValueChanged (juce::Slider* slider)
override;
    juce::Slider sliderName;
    juce::Label sliderNameValue;
```

Where sliderName (sliderNameValue) is replaced by a suitable name for each slider and its label.

The Volume slider should adjust the output signal level between float 0.0 and float 1.0 with steps of 0.1. The slider listener function should call the Synth member function `set_ampl(value)`, where value is the slider value.

The Synth Decay slider should adjust the Synth Class member data `decay_factor` to be one of 10 values, where the slider range (from 0.0 to 1.0 in 0.1 steps) is converted into an index of 0 to 9 and used as a table look-up. The retrieved value is used in the Synth member function `set_decay(value)`.

Create a set of Radio buttons that disables all effects or enables only one of the four effects:

- Effects off
- Enable Vibrato
- Enable Chorus
- Enable Echo
- Enable LFO

In PluginEditor.h this requires, for each slider, something like:

```
juce::ToggleButton toggleButtonName;
```

Where toggleButtonName is replaced by a suitable name for the toggle button, like "vib_fx".

You will have to add other code as well. Review the tutorials.

PluginEditor.cpp

At top of this file, after
```
#include "PluginProcessor.h"
#include "PluginEditor.h"
```
Add:
```
#include "synth.h"
#include "delay_fx.h"
```

Next, add this code to identify that the buttons should act as Radiobuttons:
```
enum RadioButtonIDs {
    FxButtons = 10001
};
```

Three functions in this file need to be modified:
```
SynthAudioProcessorEditor::SynthAudioProcessorEditor()

SynthAudioProcessorEditor::paint ()

SynthAudioProcessorEditor::resized()
```

SynthAudioProcessorEditor()
You must define the two sliders and the four togglebuttons. Read the JUCE tutorials to see how to do this.

You must declare a listener function for each of the sliders.

At the end of this function, add this code that translates each button click to an action by the DelayFx member functions. In this code, the button names (x_fs, .etc) can be anything, but have to match to the names in PluginEditor.h.
```
    x_fx.onClick = [this] { audioProcessor.delay_fx.disable_all(); };
    v_fx.onClick = [this] { audioProcessor.delay_fx.enable_vibrato(5, 0.4); };
    c_fx.onClick = [this] { audioProcessor.delay_fx.enable_chorus(15); };
    e_fx.onClick = [this] { audioProcessor.delay_fx.enable_echo(50); };
    l_fx.onClick = [this] { audioProcessor.delay_fx.enable_lfo(10); };
```

Paint()
Only thing to add is the plugin label “NYU Synthesizer” at the top of the GUI panel.

Resize()
Place the two sliders and the five radio buttons within the GUI panel.

Slider Listener Function
Finally, at the bottom of this file, add a slider listener function
```
void SynthAudioProcessorEditor::sliderValueChanged
(juce::Slider* slider)
{
     //Your code here
}
```

Note that you can use

```
if (slider == &synthAmp) {
…
}
else {
…
}
```

to determine which slider was changed, where synthAmp would be the name of the Output Level slider.

Use

```
synthAmp.getValue();
```

to get the slider value, where synthAmp could be the name of the first slider.

Test 3:

- Click on the XCode icon (or VS icon)

This will bring up XCode

- Click on the "Build" icon ("play" icon) to compile code

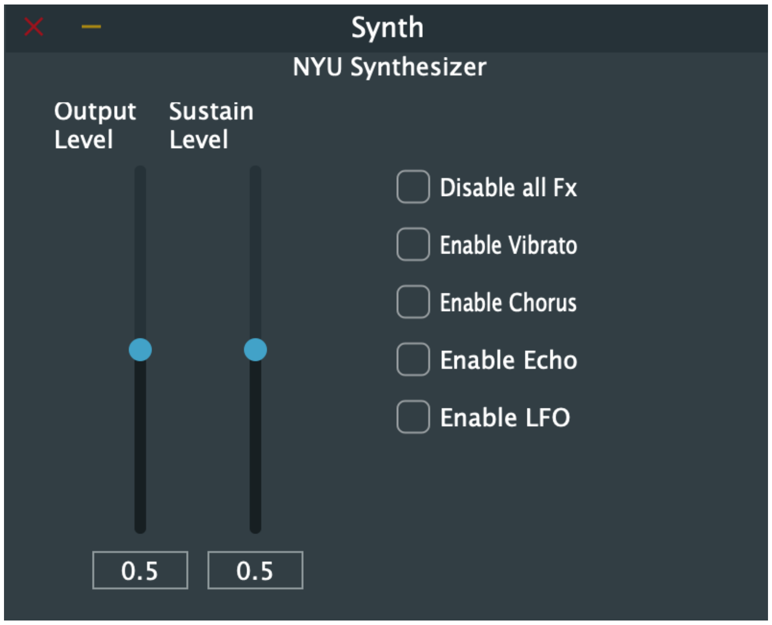
Hopefully this should result in a "Succeeded" result (no errors)

Open JUCE Plug-in Host
Scan for new VST3 plug-ins
Load

- NYU->Synth
- Connect output Midi Input to input of Synth
- Connect L and R outputs of Synth to L and R inputs of Audio Output.

Click on any key of the piano keyboard graphic and note should play.

- Double-click on Synth to show the GUI, which might be as shown here:



- Manipulate the sliders
  - The box below should change to show the new slider value
  - The audio output level should change in response to first slider
  - The tone decay time should change in response to second slider
- Click on the radio buttons
  - "Disable" should disable all effect
  - The other buttons should enable the appropriate effect

Congratulations!