See <https://audioordeal.co.uk/how-to-build-a-vst-lesson-1-intro-to-juce/>

After downloading Juce, run Projucer. You can use global search paths for modules by clicking on **Enable/disable global path for modules...** To set your global search paths, navigate to menu item **Projucer > Global Search Paths** on MacOS or [**File**](https://docs.juce.com/master/classFile.html)**> Global Search Paths** on Windows and Linux.

Open global paths and make sure they are set to the right locations. Global paths need to be set in the Projucer app, in my case they are in C:\Program Files.

Graphical user interface

Description automatically generated

Projucer will automatically generate different starter files depending on what you want to develop. We will develop an Audio Plugin. An audio plugin can be placed on a channel in your DAW.

Graphical user interface

Description automatically generated

Projucer automatically creates project files, for now select audio plugin.

Graphical user interface, text

Description automatically generated

On this screen choose the project name, location and which IDE you have installed.

This screen shows the files automatically created by JUCE. Open up the settings by clicking the Settings Icon or selecting View -> Show Project Settings and select VST3 as the plugin format. JUCE makes it very easy to change the format of your plugin, for this tutorial we will create a VST3 plugin. Now select the open in IDE button, which will open up the project in your IDE.

Graphical user interface, text, application

Description automatically generated

1. Open Settings
2. Select VST3
3. Open in IDE, in this case Visual Studio

Build a solution in your IDE to make sure everything is installed properly.

## Coding Your First Plugin

You should see 4 files in your source folder:

Graphical user interface, text, application, chat or text message

Description automatically generated

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

These files contain a bunch of auto-generated code, you may feel intimidated at this point, but you don’t need to worry about most of this code yet! To start we will focus on the PluginProcessor.cpp file, so open up this file  in your IDE. This is where you will code the bulk of your plugin.

Scroll down until you reach the process block. Delete all the pre-generated code between the brackets.

A computer screen capture

Description automatically generated with medium confidence

When JUCE auto-generated the project files, a buffer array was created. This is an array of samples that changes depending on the block size set in your DAW. So if your block size is set to 512 samples, the array will have a length of 512. We will use a for loop to change the data in this array, by iterating through the array and changing the value of the samples.

A screenshot of a computer

Description automatically generated with medium confidence

Write this code into your process block:

auto\* channeldata = buffer.getWritePointer(0); //—1

for (int i = 0; i < buffer.getNumSamples(); i++) //—2

       {

       }

1: Assigns a variable channeldata as the write pointer of the audio buffer. Selecting 0 in getWritePointer() will assign channeldata to the left input channel.

2: For loop that iterates through the audio buffer but doesn’t change anything yet.

Text

Description automatically generated

Add these three lines to the for loop.

auto input = channeldata[i]; //—1

       input = input \* 0.0f; //—2

       channeldata[i] = input; //—3

1: Sets a variable input as one sample i from the channeldata variable. This input variable will change every sample.

2: Multiplies input by 0. Muting this channel of audio data.

3: Input is written back into channel data.

So this code multiplies every sample value in the audio buffer by 0, muting the output. When this plugin is placed on a track in your DAW it will mute the left channel of audio.’

These tables should give you a better idea about what this code is doing. The top table shows the sample data in a buffer before it enters our for loop. The bottom table shows the sample data after the loop. Every sample value has been mulitplied by 0. When we write this data back into the channeldata variable we are replacing the first sample values with the sample values multiplied by 0, which mutes the output from this channel.

Let’s try this out now in your DAW, first build the project. This will create a VST3 file in the project directory under Builds->VisualStudio2017->x64->Debug->VST3. Open your DAW of choice and set this folder as a [VST](https://audioordeal.co.uk/top-free-vsts-of-2019/) location, or move the built file to your VST directory.

Add the new VST effect to an audio channel and you should see the following results:

A screenshot of a computer

Description automatically generated

The left channel of audio is muted when the plugin is turned on. To mute both channels of audio make the following modifications to the code:

Text

Description automatically generated

auto inputL = channeldataL[i];   
auto inputR = channeldataR[i]; //—1

inputL = inputL \* 0;  
inputR = inputR \* 0; //—2

channeldataL[i] = inputL;  
channeldataR[i] = inputR; //—3

With this code both channels  are affected by the code. If you build the plugin again you should see both left and right channel being muted by the plugin.

**Make sure to move the newly built plugin to your VST folder to see the results**

Try changing the value multiplied to InputL and InputR, you will see a boost or cut in volume depending on the values you choose after you have built the VST plugin. To change this value in real-time from the DAW we need to build a UI.

There are a few ways to design the UI for a JUCE plugin. One of the simplest ways is to use the generic audio processor UI. This can be used to add UI elements like sliders and boxes to your plugin.

To set this up replace the following code:



//return new DemoProjectAudioProcessorEditor (\*this);

return new GenericAudioProcessorEditor(this);

This means the plugin will use the generic UI provided with the JUCE library.

Next, we need to add a variable in the PluginProcessor.h file. Open this file, find the private section and add the following code:

Text

Description automatically generated

Adding an AudioParameterFloat to Plugin.Processor.h

AudioParameterFloat\* gain;

This variable will be changed by the UI slider we add next.

Back in PluginProcessor.cpp add this code:

Text

Description automatically generated

This adds a new parameter, the generic UI will generate a slider to control this parameter.

addParameter(gain = new AudioParameterFloat(“gain”, “Gain”, 0.0f, 1.0f, 0.0f));

“gain” references the AudioProcessorFloat we created in PluginProcessor.h.

“Gain” will be shown next to the slider in the DAW.

The float values set the minimum, maximum, and default values for the slider.

Build the plugin now and you will see a slider called Gain. This can be moved between 0 and 1 but doesn’t change anything yet.

Text

Description automatically generated

float gSlider = gain->get();

inputL = inputL \* gSlider;

inputR = inputR \* gSlider;

This code first declares a new variable gSlider, then uses a get function to find the value of the gain slider, this value will be changed by moving the slider in the DAW. This value is then multiplied to inputL and inputR, replacing the static float we were using before.

Build the plugin again and it will look like this in your DAW:

Graphical user interface, application

Description automatically generated

The gain slider will now change the volume of incoming audio data.

In this tutorial we created a very basic gain plugin, this isn’t a very exciting plugin but shows the process of developing and testing audio plugins using JUCE. More complicated plugins are made by modifying the process block and adding more UI elements like buttons and choice boxes.

In the next tutorial I’ll show you how to use these UI elements and how to code a slightly more sophisticated plugin.

Code for the plugin can be found here: <https://github.com/aRycroft/JuceTutorial1>