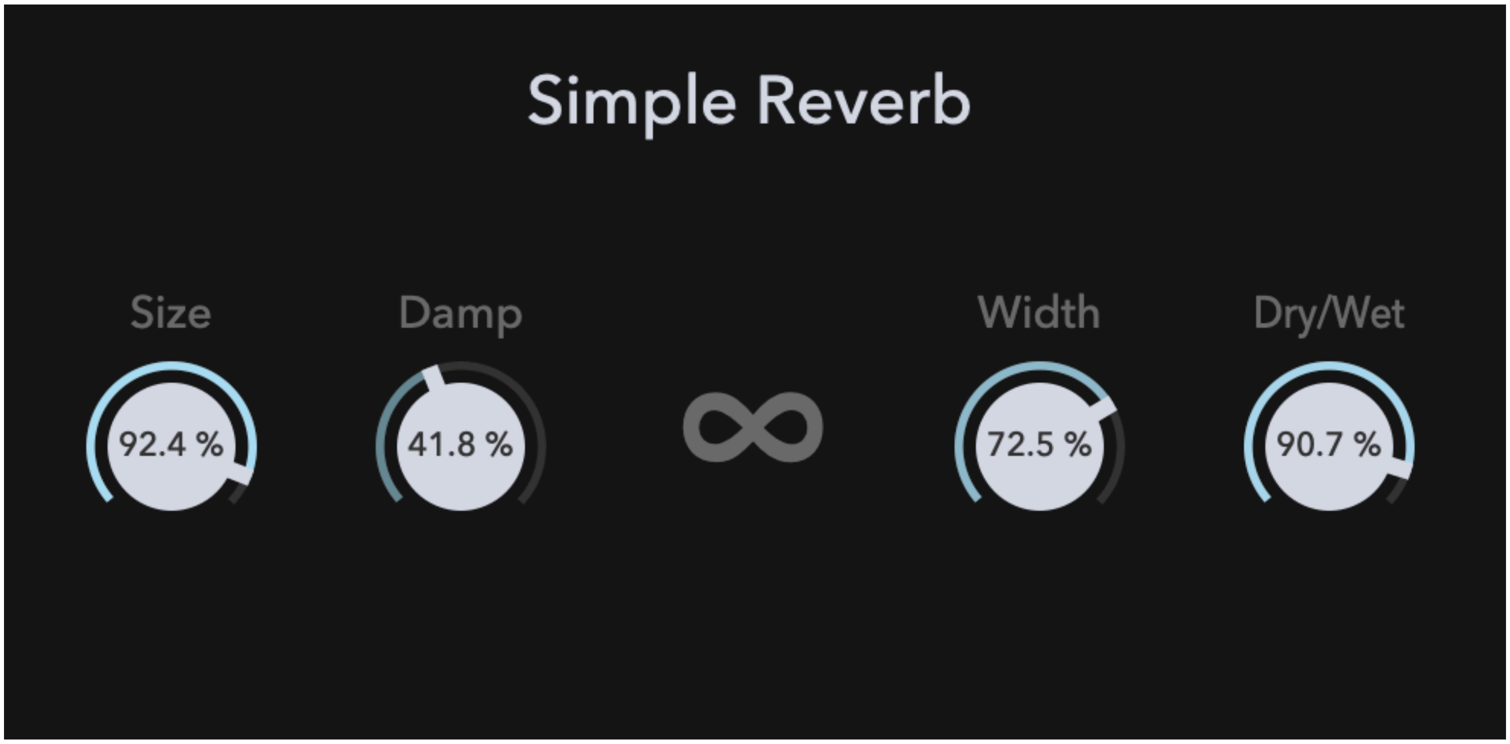
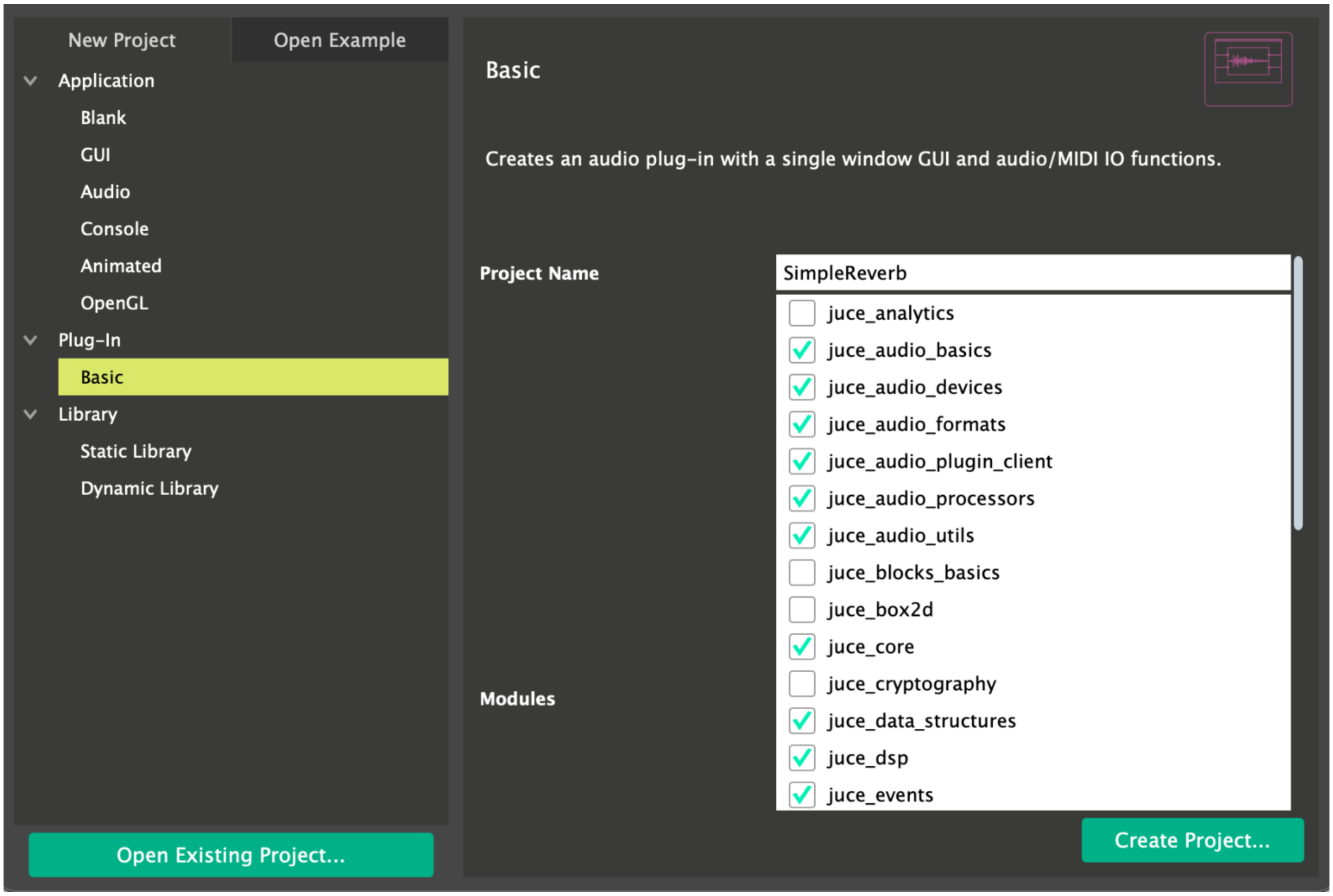
前言



建立项目

打开Projucer，创建一个新的项目叫做SimpleReverb，记得对juce_dsp打上勾，以此来在项目中使用JUCE的DSP库。



Basic

Creates an audio plug-in with a single window GUI and audio/MIDI IO functions.

Project Name: SimpleReverb

Modules:
- [ ] juce_analytics
- [✓] juce_audio_basics
- [✓] juce_audio_devices
- [✓] juce_audio_formats
- [✓] juce_audio_plugin_client
- [✓] juce_audio_processors
- [✓] juce_audio_utils
- [ ] juce_blocks_basics
- [ ] juce_box2d
- [✓] juce_core
- [ ] juce_cryptography
- [✓] juce_data_structures
- [✓] juce_dsp
- [✓] juce_events

DSP

APVTS

我十分推荐使用APVTS(AudioProcessorValueTreeState)来管理各种参数，因为这样做会比使用传统的方法简单很多。 首先如下来创建一个APVTS对象：

```
class SimpleReverbAudioProcessor  : public juce::AudioProcessor
{
public:
 . . .
static juce::AudioProcessorValueTreeState::ParameterLayout createParameterLayout
();

juce::AudioProcessorValueTreeState apvts { *this, nullptr, "Parameters", createP
arameterLayout() };
```

在createParameter()中放置你所有想要在APVTS中管理的对象。在这个项目中我们将要设计一个混响效果器，所以我们会用到juce::dsp::Reverb::Parameters(juce::Reverb::Parameters)
用到其中的参数有：

- roomSize
- damping
- wetLevel
- dryLevel
- width
- freezeMode

这些参数可以一一对应一个旋钮，但是将Dry和Wet这两个参数分拆成两个旋钮是非常不方便的（其实就是干湿比，可以发现基本我们使用的所有插件就没有将这两个参数分开的），所以我们把这两个参数合并为一个Dry/Wet旋钮。与此同时，我个人认为以百分比的形式展示参数会更好。
APVTS中的添加具体如下：

```
juce::AudioProcessorValueTreeState::ParameterLayout SimpleReverbAudioProcessor::
createParameterLayout()
{
    juce::AudioProcessorValueTreeState::ParameterLayout layout;

    layout.add (std::make_unique<juce::AudioParameterFloat> ("Room Size",
                                                              "Room Size",
                                                              juce::NormalisableR
ange<float> (0.0f, 1.0f, 0.001f, 1.0f),
                                                              0.5f,
                                                              juce::String(),
                                                              juce::AudioProcesso
rParameter::genericParameter,
                                                              [](float value, int
) {
                                                                 if (value * 100
< 10.0f)
                                                                     return juce:
:String (value * 100, 2);
                                                                 else if (value *
 100 < 100.0f)
                                                                     return juce:
:String (value * 100, 1);
                                                                 else
                                                                     return juce:
:String (value * 100, 0); },
                                                              nullptr));

    layout.add (std::make_unique<juce::AudioParameterFloat> ("Damping",
                                                              "Damping",
                                                              juce::NormalisableR
ange<float> (0.0f, 1.0f, 0.001f, 1.0f),
                                                              0.5f,
                                                              juce::String(),
                                                              juce::AudioProcesso
rParameter::genericParameter,
                                                              [](float value, int
) {
                                                                 if (value * 100
< 10.0f)
                                                                     return juce:
:String (value * 100, 2);
                                                                 else if (value *
 100 < 100.0f)
                                                                     return juce:
:String (value * 100, 1);
                                                                 else
                                                                     return juce:
:String (value * 100, 0); },
                                                              nullptr));

    layout.add (std::make_unique<juce::AudioParameterFloat> ("Width",
                                                              "Width",
                                                              juce::NormalisableR
ange<float> (0.0f, 1.0f, 0.001f, 1.0f),
                                                              0.5f,
                                                              juce::String(),
                                                              juce::AudioProcesso
rParameter::genericParameter,
                                                              [](float value, int
) {
                                                                 if (value * 100
< 10.0f)
                                                                     return juce:
:String (value * 100, 2);
                                                                 else if (value *
 100 < 100.0f)
                                                                     return juce:
:String (value * 100, 1);
                                                                 else
```

```
                                                                    return juce:
:String (value * 100, 0); },
                                                            nullptr));

    layout.add (std::make_unique<juce::AudioParameterFloat> ("Dry/Wet",
                                                             "Dry/Wet",
                                                             juce::NormalisableR
ange<float> (0.0f, 1.0f, 0.001f, 1.0f),
                                                             0.5f,
                                                             juce::String(),
                                                             juce::AudioProcesso
rParameter::genericParameter,
                                                             [](float value, int
) {
                                                                if (value * 100
< 10.0f)
                                                                    return juce:
:String (value * 100, 2);
                                                                else if (value *
 100 < 100.0f)
                                                                    return juce:
:String (value * 100, 1);
                                                                else
                                                                    return juce:
:String (value * 100, 0); },
                                                             nullptr));

    layout.add (std::make_unique<juce::AudioParameterBool> ("Freeze", "Freeze",
false));

    return layout;
}
```

juce::dsp::Reverb

现在APVTS已经准备好了，我们将要开始设计混响部分。首先让我们创建一个juce::dsp::Reverb::Parameters对象，这个在上文稍稍介绍过。同时，创建两个juce::dsp::Reverb对象来支持双声道立体声：

```
class SimpleReverbAudioProcessor  : public juce::AudioProcessor
{
. . .
private:
    juce::dsp::Reverb::Parameters params;
    juce::dsp::Reverb leftReverb, rightReverb;
    //==========================================================================
====
    JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (SimpleReverbAudioProcessor)
};
```

接着，准备一个ProcessSpec对象来存储一些必要的信息来初始化你已经创建的Reverb对象：

```
void SimpleReverbAudioProcessor::prepareToPlay (double sampleRate, int samplesPe
rBlock)
{
    juce::dsp::ProcessSpec spec;

    spec.sampleRate = sampleRate;
    spec.maximumBlockSize = samplesPerBlock;
    spec.numChannels = 1;

    leftReverb.prepare (spec);
    rightReverb.prepare (spec);
}
```

然后我们开始设计音频处理的部分。正如之前说的那样，我们要把那两个参数合并成一个Dry/Wet旋钮，所以我们会用1-wet的方式得到dry的值，具体如下：

```
void SimpleReverbAudioProcessor::processBlock (juce::AudioBuffer<float>& buffer,
 juce::MidiBuffer& midiMessages)
{
    juce::ScopedNoDenormals noDenormals;
    auto totalNumInputChannels  = getTotalNumInputChannels();
    auto totalNumOutputChannels = getTotalNumOutputChannels();

    for (auto i = totalNumInputChannels; i < totalNumOutputChannels; ++i)
        buffer.clear (i, 0, buffer.getNumSamples());

    params.roomSize   = *apvts.getRawParameterValue ("Room Size");
    params.damping    = *apvts.getRawParameterValue ("Damping");
    params.width      = *apvts.getRawParameterValue ("Width");
    params.wetLevel   = *apvts.getRawParameterValue ("Dry/Wet");
    params.dryLevel   = 1.0f - *apvts.getRawParameterValue ("Dry/Wet");
    params.freezeMode = *apvts.getRawParameterValue ("Freeze");

    leftReverb.setParameters (params);
    rightReverb.setParameters (params);
    juce::dsp::AudioBlock<float> block (buffer);

    auto leftBlock = block.getSingleChannelBlock (0);
    auto rightBlock = block.getSingleChannelBlock (1);

    juce::dsp::ProcessContextReplacing<float> leftContext (leftBlock);
    juce::dsp::ProcessContextReplacing<float> rightContext (rightBlock);

    leftReverb.process (leftContext);
    rightReverb.process (rightContext);
}
```

至此，dsp部分的设计已经完成。

UI

CustomLookAndFeel

首先，我们将要定制LookAndFeel，这是我们UI元素的基础。打开Projucer并创建一个new header and cpp file。
其中头文件的具体设计如下：

```
#pragma once

#include <JuceHeader.h>

class CustomLookAndFeel : public juce::LookAndFeel_V4
{
public:
    CustomLookAndFeel();
    ~CustomLookAndFeel();

    juce::Slider::SliderLayout getSliderLayout (juce::Slider& slider) override;

    void drawRotarySlider (juce::Graphics&, int x, int y, int width, int height,
                           float sliderPosProportional, float rotaryStartAngle,
                           float rotaryEndAngle, juce::Slider&) override;

    juce::Label* createSliderTextBox (juce::Slider& slider) override;
    juce::Font getTextButtonFont (juce::TextButton&, int buttonHeight) override;

    void drawButtonBackground (juce::Graphics& g, juce::Button& button,
                               const juce::Colour& backgroundColour,
                               bool shouldDrawButtonAsHighlighted,
                               bool shouldDrawButtonAsDown) override;

private:
    juce::Colour blue      = juce::Colour::fromFloatRGBA (0.43f, 0.83f, 1.0f,  1
.0f);
    juce::Colour offWhite  = juce::Colour::fromFloatRGBA (0.83f, 0.84f, 0.9f,  1
.0f);
    juce::Colour grey      = juce::Colour::fromFloatRGBA (0.42f, 0.42f, 0.42f, 1
.0f);
    juce::Colour blackGrey = juce::Colour::fromFloatRGBA (0.2f,  0.2f,  0.2f,  1
.0f);

    JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (CustomLookAndFeel);
};
```

以下为每个函数的具体内容讲解

getSliderLayout

函数getSliderLayout()用来设定slider的基本放置位置。换句话说，它定义了slider的大小和它应该出现在哪个位置，以及与它对应的文本框应该被放置在什么位置。
这个项目的RotarySlider（旋钮）的文本框被定义在旋钮的正中间。

```
juce::Slider::SliderLayout CustomLookAndFeel::getSliderLayout (juce::Slider& sli
der)
{
    auto localBounds = slider.getLocalBounds();

    juce::Slider::SliderLayout layout;

    layout.textBoxBounds = localBounds;
    layout.sliderBounds = localBounds;

    return layout;
}

drawRotarySlider()
在这个构造函数中，通过调用以下这些函数来配置细节。
```

```
RotarySlider::RotarySlider()
{
    setSliderStyle (juce::Slider::SliderStyle::RotaryVerticalDrag);
    setTextBoxStyle (juce::Slider::TextBoxBelow, true, 0, 0);
    setLookAndFeel (&customLookAndFeel);
    setColour (juce::Slider::rotarySliderFillColourId, blue);
    setColour (juce::Slider::textBoxTextColourId, blackGrey);
    setColour (juce::Slider::textBoxOutlineColourId, grey);
    setVelocityBasedMode (true);
    setVelocityModeParameters (0.5, 1, 0.09, false);
    setRange (0.0, 100.0, 0.01);
    setRotaryParameters (juce::MathConstants<float>::pi * 1.25f,
                         juce::MathConstants<float>::pi * 2.75f,
                         true);
    setWantsKeyboardFocus (true);
    setTextValueSuffix (" %");
    onValueChange = [&]()
    {
        if (getValue() < 10)
            setNumDecimalPlacesToDisplay (2);
        else if (getValue() >= 10 && getValue() < 100)
            setNumDecimalPlacesToDisplay (1);
        else
            setNumDecimalPlacesToDisplay (0);
    };
}
```

paint()

当我们将鼠标放置在旋钮上的时候，这个函数将会被调用，具体如下。

```
void RotarySlider::paint (juce::Graphics& g)
{
    juce::Slider::paint (g);

    if (hasKeyboardFocus (false))
    {
        auto length = getHeight() > 15 ? 5.0f : 4.0f;
        auto thick  = getHeight() > 15 ? 3.0f : 2.5f;

        g.setColour (findColour (juce::Slider::textBoxOutlineColourId));

        //          fromX       fromY       toX                   toY
        g.drawLine (0,          0,          0,                   length,
        thick);
        g.drawLine (0,          0,          length,              0,
        thick);
        g.drawLine (0,          getHeight(), 0,                  getHeight() -
length, thick);
        g.drawLine (0,          getHeight(), length,             getHeight(),
        thick);
        g.drawLine (getWidth(), getHeight(), getWidth() - length, getHeight(),
        thick);
        g.drawLine (getWidth(), getHeight(), getWidth(),         getHeight() -
length, thick);
        g.drawLine (getWidth(), 0,          getWidth() - length, 0,
        thick);
        g.drawLine (getWidth(), 0,          getWidth(),          length,
        thick);
    }
}
```

mouseDown()/mouseUp()

使得当你的鼠标点击并拖拽旋钮时消失，松开时再出现。

```
void RotarySlider::mouseDown (const juce::MouseEvent& event)
{
    juce::Slider::mouseDown (event);

    setMouseCursor (juce::MouseCursor::NoCursor);
}

void RotarySlider::mouseUp (const juce::MouseEvent& event)
{
    juce::Slider::mouseUp (event);

    juce::Desktop::getInstance().getMainMouseSource().setScreenPosition (event.s
ource.getLastMouseDownPosition());
    setMouseCursor (juce::MouseCursor::NormalCursor);
}
```

NameLabel

这个类被用来设定每个旋钮对应的标签。如果我们设计这个类的话，那么我们将会使用重复的代码多次，这会十分不美观。所以在Projucer中创建一个新的头文件，具体内容如下。

```cpp
#pragma once

#include <JuceHeader.h>

class NameLabel  : public juce::Label
{
public:
    NameLabel()
    {
        setFont (20.f);
        setColour (juce::Label::textColourId, grey);
        setJustificationType (juce::Justification::centred);
    }

    ~NameLabel(){}

private:
    juce::Colour grey = juce::Colour::fromFloatRGBA (0.42f, 0.42f, 0.42f, 1.0f);

};
```

PluginEditor

现在我们已经完成了UI元素的设计，准备开始PluginEditor部分的设计。

```cpp
#pragma once

#include <JuceHeader.h>
#include "PluginProcessor.h"
#include "CustomLookAndFeel.h"
#include "RotarySlider.h"
#include "NameLabel.h"

//==============================================================================
/**
*/
class SimpleReverbAudioProcessorEditor  : public juce::AudioProcessorEditor
{
public:
    SimpleReverbAudioProcessorEditor (SimpleReverbAudioProcessor&);
    ~SimpleReverbAudioProcessorEditor() override;

    //==============================================================================
====
    void paint (juce::Graphics&) override;
    void resized() override;

private:
    SimpleReverbAudioProcessor& audioProcessor;

    NameLabel sizeLabel,
              dampLabel,
              widthLabel,
              dwLabel;

    RotarySlider sizeSlider,
                 dampSlider,
                 widthSlider,
                 dwSlider;

    juce::TextButton freezeButton;

    juce::AudioProcessorValueTreeState::SliderAttachment sizeSliderAttachment,
                                                         dampSliderAttachment,
                                                         widthSliderAttachment,
                                                         dwSliderAttachment;

    juce::AudioProcessorValueTreeState::ButtonAttachment freezeAttachment;

    CustomLookAndFeel customLookAndFeel;

    juce::Colour blue      = juce::Colour::fromFloatRGBA (0.43f, 0.83f, 1.0f, 1.
0f);
    juce::Colour offWhite  = juce::Colour::fromFloatRGBA (0.83f, 0.84f, 0.9f, 1.
0f);
    juce::Colour grey      = juce::Colour::fromFloatRGBA (0.42f, 0.42f, 0.42f, 1
.0f);
    juce::Colour black     = juce::Colour::fromFloatRGBA (0.08f, 0.08f, 0.08f, 1
.0f);

    JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (SimpleReverbAudioProcessorEdit
or)
};
```

SimpleReverbAudioProcessorEditor()

构造函数部分如下：

```
SimpleReverbAudioProcessorEditor::SimpleReverbAudioProcessorEditor (SimpleReverb
AudioProcessor& p)
    : AudioProcessorEditor (&p), audioProcessor (p),
      sizeSliderAttachment (audioProcessor.apvts, "Room Size", sizeSlider),
      dampSliderAttachment (audioProcessor.apvts, "Damping", dampSlider),
      widthSliderAttachment (audioProcessor.apvts, "Width", widthSlider),
      dwSliderAttachment (audioProcessor.apvts, "Dry/Wet", dwSlider),
      freezeAttachment (audioProcessor.apvts, "Freeze", freezeButton)
{
    juce::LookAndFeel::getDefaultLookAndFeel().setDefaultSansSerifTypefaceName (
"Avenir Next Medium");

    setSize (500, 250);
    setWantsKeyboardFocus (true);

    sizeLabel.setText ("Size", juce::NotificationType::dontSendNotification);
    sizeLabel.attachToComponent (&sizeSlider, false);

    dampLabel.setText ("Damp", juce::NotificationType::dontSendNotification);
    dampLabel.attachToComponent (&dampSlider, false);

    widthLabel.setText ("Width", juce::NotificationType::dontSendNotification);
    widthLabel.attachToComponent (&widthSlider, false);

    dwLabel.setText ("Dry/Wet", juce::NotificationType::dontSendNotification);
    dwLabel.attachToComponent (&dwSlider, false);

    freezeButton.setButtonText (juce::String (juce::CharPointer_UTF8 ("∞")));
    freezeButton.setClickingTogglesState (true);
    freezeButton.setLookAndFeel (&customLookAndFeel);
    freezeButton.setColour (juce::TextButton::buttonColourId, juce::Colours::tra
nsparentWhite);
    freezeButton.setColour (juce::TextButton::buttonOnColourId, juce::Colours::t
ransparentWhite);
    freezeButton.setColour (juce::TextButton::textColourOnId, blue);
    freezeButton.setColour (juce::TextButton::textColourOffId, grey);

    addAndMakeVisible (sizeSlider);
    addAndMakeVisible (dampSlider);
    addAndMakeVisible (widthSlider);
    addAndMakeVisible (dwSlider);
    addAndMakeVisible (freezeButton);
}
```

paint()

用烟灰色填满背景，并打上"Simple Reverb"字样：

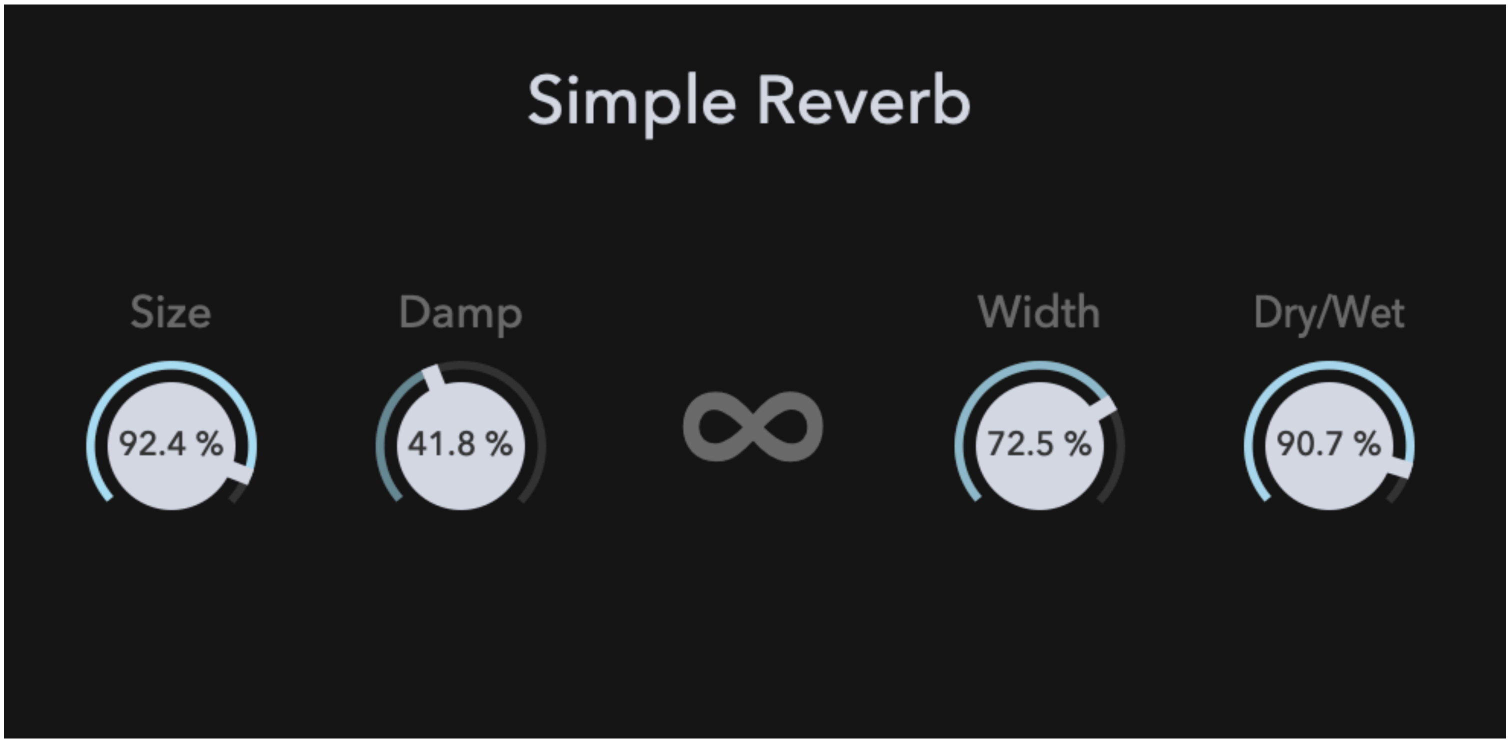
```
void SimpleReverbAudioProcessorEditor::paint (juce::Graphics& g)
{
    g.fillAll (black);

    g.setFont (30);
    g.setColour (offWhite);
    g.drawText ("Simple Reverb", 150, 0, 200, 75, juce::Justification::centred);
}
```

resized()

每个元素的放置位置和大小：

```
void SimpleReverbAudioProcessorEditor::resized()
{
    sizeSlider.setBounds (30, 120, 60, 60);
    dampSlider.setBounds (125, 120, 60, 60);
    widthSlider.setBounds (315, 120, 60, 60);
    dwSlider.setBounds (410, 120, 60, 60);
    freezeButton.setBounds (210, 120, 80, 40);
}
```

运行尝试



总结

在这篇教程中，我介绍了如何去通过JUCE DSP模块去设计一个简单的混响效果器。我十分推荐这个模块，因为它能让你很快的创建一个插件。 如果有更加高效的方法来实现这些，欢迎评论。感谢您能阅读到这里！