Coding workshop: the basic WaveformDisplay class

Introduction

Waveform displays are often found in audio software. In DJ applications, they allow the DJ to see the overall structure of the musical piece and to identify cue points, or points where the DJ might want to start the playback of a given track. For example, they might want to drop the track when the drums start.

In this worksheet, we will implement a waveform display Component in our DJ application.

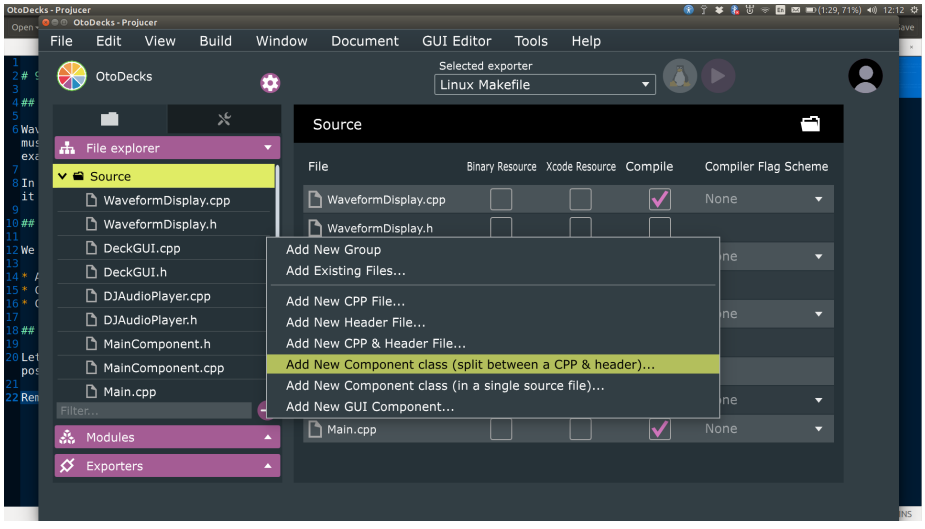
JUCE library components we will use

We will use the following JUCE library components in this worksheet:

- AudioThumbnail
- Component
- ChangeListener

Create a new Component class called WaveformDisplay

Let's start by creating the new class. Remembering that we need to use Projucer to add files to our projects, open the project in Projucer and add a new Component class with separate header and CPP files, as shown in the diagram below:



width="50%">

Call the class WaveformDisplay.

Now *save the project* from Projucer and open it in your IDE. Check out the WaveformDisplay.h header file. The class definition should look something like this:

```
class WaveformDisplay    : public Component
{
public:
    WaveformDisplay();
    ~WaveformDisplay();

    void paint (Graphics&) override;
    void resized() override;

private:
    JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (WaveformDisplay)
};
```

You will note that it inherits from Component. The WaveformDisplay will behave like a Component.

Write some simple paint code and integrate the WaveformDisplay into the DeckGUI

The next step is to write some basic code in the paint function, then to add this Component to our DeckGUI to verify that it works. In paint, you'll see that Projucer has already generated some starter code:

```
void WaveformDisplay::paint (Graphics& g)
{

    g.fillAll (getLookAndFeel().findColour (ResizableWindow::backgroundColourId));   // clea

    g.setColour (Colours::grey);
    g.drawRect (getLocalBounds(), 1);   // draw an outline around the component

    g.setColour (Colours::white);
    g.setFont (14.0f);
    g.drawText ("WaveformDisplay", getLocalBounds(),
                Justification::centred, true);   // draw some placeholder text
}
```

Let's adjust that a bit (change the colours, change the font size, change the text)

```
void WaveformDisplay::paint (Graphics& g)
{

    g.fillAll (getLookAndFeel().findColour (ResizableWindow::backgroundColourId));   // clea
```

```
    g.setColour (Colours::black);
    g.drawRect (getLocalBounds(), 1);   // draw an outline around the component

    g.setColour (Colours::green);
    g.setFont (24.0f);
    g.drawText ("Waveform here!", getLocalBounds(),
                Justification::centred, true);   // draw some placeholder text
}
```

Now let's add this Component to the DeckGUI class. In DeckGUI.h:

```
#include "WaveformDisplay.h"
```

Then in the private section of the DeckGUI class:

```
WaveformDisplay waveformDisplay;
```

Now in DeckGUI.cpp, add this to DeckGUI::DeckGUI:

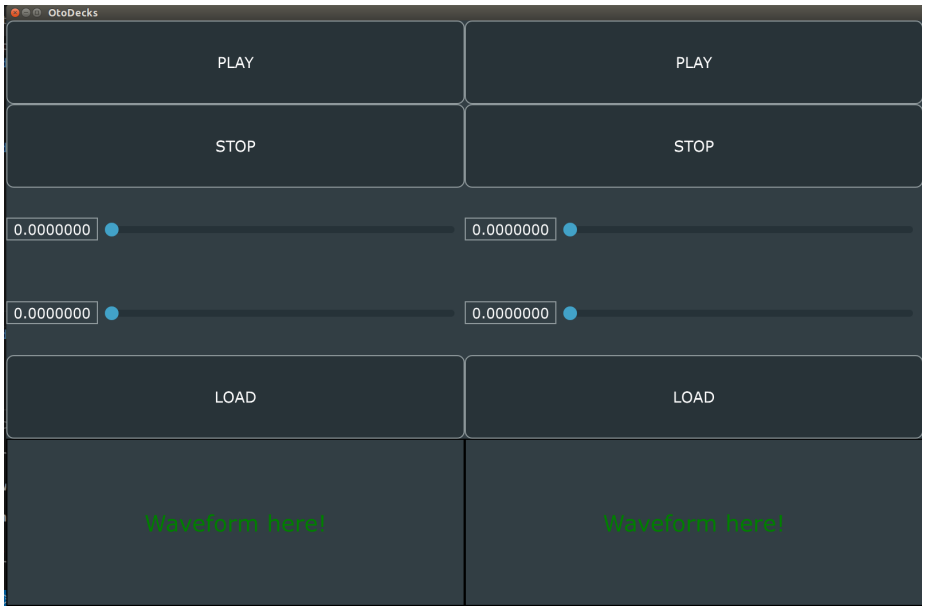
```
addAndMakeVisible(waveformDisplay);
```

You might remember that the next thing to do to add a Component is to resize it in the owning Component's resized function.

In the resized function, we'll do a bit of re-arrangement as we want the waveform to take up two rows. Let's squeeze it above the load button. In DeckGUI.cpp, add this code (You might have volumeSlider or volSlider, and posSlider or positionSlider, depending on how you go to this point.):

```
void DeckGUI::resized()
{
    double rowH = getHeight() / 8;
    playButton.setBounds(0, 0, getWidth(), rowH);
    stopButton.setBounds(0, rowH, getWidth(), rowH);
    volSlider.setBounds(0, rowH * 2, getWidth(), rowH);
    speedSlider.setBounds(0, rowH * 3, getWidth(), rowH);
    posSlider.setBounds(0, rowH * 4, getWidth(), rowH);
    waveformDisplay.setBounds(0, rowH * 5, getWidth(), rowH * 2);
    loadButton.setBounds(0, rowH * 7, getWidth(), rowH);
}
```

Now compile and run. You should see something like this:



width="50%">

Conclusion

In this worksheet, we have implemented a new Component and integrated into our application.

The WaveformDisplay.h file

```
#pragma once
#include "../JuceLibraryCode/JuceHeader.h"

class WaveformDisplay    : public Component

public:
    WaveformDisplay();
    ~WaveformDisplay();

    void paint (Graphics&) override;
    void resized() override;

private:

    JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (WaveformDisplay)
};
```

The WaveformDisplay.cpp file

```
#include "../JuceLibraryCode/JuceHeader.h"
#include "WaveformDisplay.h"

WaveformDisplay::WaveformDisplay()
{
}

WaveformDisplay::~WaveformDisplay()
{
}

void WaveformDisplay::paint (Graphics& g)
{
    g.fillAll (getLookAndFeel().findColour (ResizableWindow::backgroundColourId));   // clea

    g.setColour (Colours::grey);
    g.drawRect (getLocalBounds(), 1);   // draw an outline around the component

    g.setColour (Colours::orange);

    g.setFont (20.0f);
    g.drawText ("File not loaded...", getLocalBounds(),
                Justification::centred, true);
}

void WaveformDisplay::resized()
{
}
```