Coding workshop: Refactoring the program to allow for a single Thumbnail cache and format manager.

Introduction

In this worksheet, we are going to work on the WaveformDislay, so that has the components it needs to do efficient waveform plotting. To achieve this, we will need to do some refactoring so that the application can make use of a cache of audio thumbnails and a single format manager. This involves a fair bit of plumbing to get everything working how we want.

Add the library components to the WaveformDisplay class

First, we need to change the WaveformDisplay class, so it has the library classes it needs to load and plot an audio file.

In WaveformDisplay.h, add this field to the private section:

```
AudioThumbnail audioThumbnail;
```

You can read about AudioThumbnail in the JUCE documentation, but it provides functions to read an audio file then to plot it (loadFrom and drawChannel / drawChannels).

If you look up AudioThumbnail, you will note that its constructor takes several arguments:

```
AudioThumbnail (int sourceSamplesPerThumbnailSample,
                AudioFormatManager &formatManagerToUse,
                AudioThumbnailCache &cacheToUse)
```

Let's consider those arguments:

- sourceSamplesPerThumbnailSample is a measure of the resolution of the image. Let's say we wanted one-second resolution, i.e. the image plots one value for every second of audio. We would pass in the sample rate for this parameter. The lower the value, the higher the resolution.
- formatManagerToUse is an AudioFormatManager that the AudioThumbnail will use to read audio files. It is being passed by reference. At the moment, we create one of these for each DJAudioPlayer. We will do a bit of refactoring so this works more simply.
- cacheToUse - this provides some background functionality for efficiently generating and storing multiple thumbnails, and it can be shared between several thumbnails. Therefore it makes sense to create this at the application level (in MainComponent so that it can be shared across all WaveformDisplays.

Set up the WaveformDisplay constructor so it can correctly instantiate the AudioThumbnail

In WaveformDisplay.h, update the constructor to this:

```
WaveformDisplay( AudioFormatManager &   formatManagerToUse,
                 AudioThumbnailCache &  cacheToUse );
```

Then in WaveformDisplay.cpp, update the constructor to this:

```
WaveformDisplay::WaveformDisplay(AudioFormatManager &   formatManagerToUse,
                                 AudioThumbnailCache &   cacheToUse)
                                 : audioThumbnail(1000, formatManagerToUse, cacheToUse)
```

Notes:

- We receive the AudioFormatManager by reference
- We receive the AudioThumbnailCache by reference
- We use member initialisation to initialise the audioThumb data member of WaveformDisplay by passing three arguments to audioThumb's constructor: 1000, formatManagerToUse and cacheToUse.

On the last point, we have not seen this way of using member initialisation before. Previously we've sent in a variable of the same type as the data member. Here we are actually calling the constructor of the data member. Chapter 11 p394 covers member initialisation via data passing. You can read about member initialisation via constructor calling on data members that are classes here: https://www.learncpp.com/cpp-tutorial/8-5a-constructor-member-initializer-lists/ (Section Initializing member variables that are classes)

Fix the DeckGUI header

Now we need to update the DeckGUI class as it is now not constructing the WaveformDisplay object properly. Currently, in DeckGUI.h, we have this:

```
WaveformDisplay waveform;
```

That will call the WaveformDisplay constructor with no arguments, but we just set it up to receive two arguments! The question is - which bit of the program should create the single AudioFormatManager and the AudioThumbnailCache. Since we only need one of each in the whole application, let's create on at the MainComponent level and feed it through to the places where it is required. So we need to go up to the MainComponent class.

Add AudioFormatManager and AudioThumbnailCache objects to the MainComponent class

In MainComponent.h, at the top of the private section:

```
private:

    AudioFormatManager formatManager;
    // note we need to tell it how large the cache is. 20 files are ok for now.
    AudioThumbnailCache thumbnailCache{20};
```

In MainComponent.cpp's constructor call registerBasicFormats:

```
formatManager.registerBasicFormats();
```

Update the DeckGUI constructor, so it needs an AudioFormatManager and an AudioThubmnailCache

Now DeckGUI needs access to the AudioFormatManager and the AudioThubmnailCache so it can pass them onto the WaveformDisplay.

In DeckGUI.h, change the constructor so it can receive the objects WaveformDisplay needs:

```
DeckGUI(DJAudioPlayer* player,
       AudioFormatManager &     formatManagerToUse,
       AudioThumbnailCache &    cacheToUse );
```

And in DeckGUI.cpp, update the constructor so it passes these variables straight over to the WaveformDisplay:

```
DeckGUI::DeckGUI(DJAudioPlayer* _player,
                 AudioFormatManager &    formatManagerToUse,
                 AudioThumbnailCache &   cacheToUse
        )
        : player(_player),  // assign _player to our player data member
          waveformDisplay(formatManagerToUse, cacheToUse) // call the constructor on wavefor
```

Note that you may have called your player variable djAudioPlayer, so change : player to : djAudioPlayer in that case.

Modify the MainComponent.h file

Now we have changed the constructor of DeckGUI, we need to update the data members of the MainComponent class.

The private section of the MainComponent class in MainComponent.h should now look something like this:

```
private:
    //==============================================================================
    // Your private member variables go here...

    AudioFormatManager formatManager;
    AudioThumbnailCache thumbCache{100};
```

```
DJAudioPlayer player1{formatManager};
DeckGUI deckGUI1{&player1, formatManager, thumbCache};

DJAudioPlayer player2{formatManager};
DeckGUI deckGUI2{&player2, formatManager, thumbCache};

MixerAudioSource mixerSource;


JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (MainComponent)
```

Right - let's go for a compile. We should see the same GUI we saw earlier, but now all the plumbing is in place to get the waveform display to draw a waveform.

Conclusion

In this worksheet, we have prepared the data an AudioThubmnail needs to draw a waveform. Since we wanted to have more than one WaveformDisplay, we needed to share this data between all WaveformDisplays. To achieve this, we placed the data at the top-level class, MainComponent, then passed it through constructors to the AudioThumbnail. Along the way, the data went into DeckGUI, to WaveformDisplay then into WaveforDisplay's audioThumb data member.