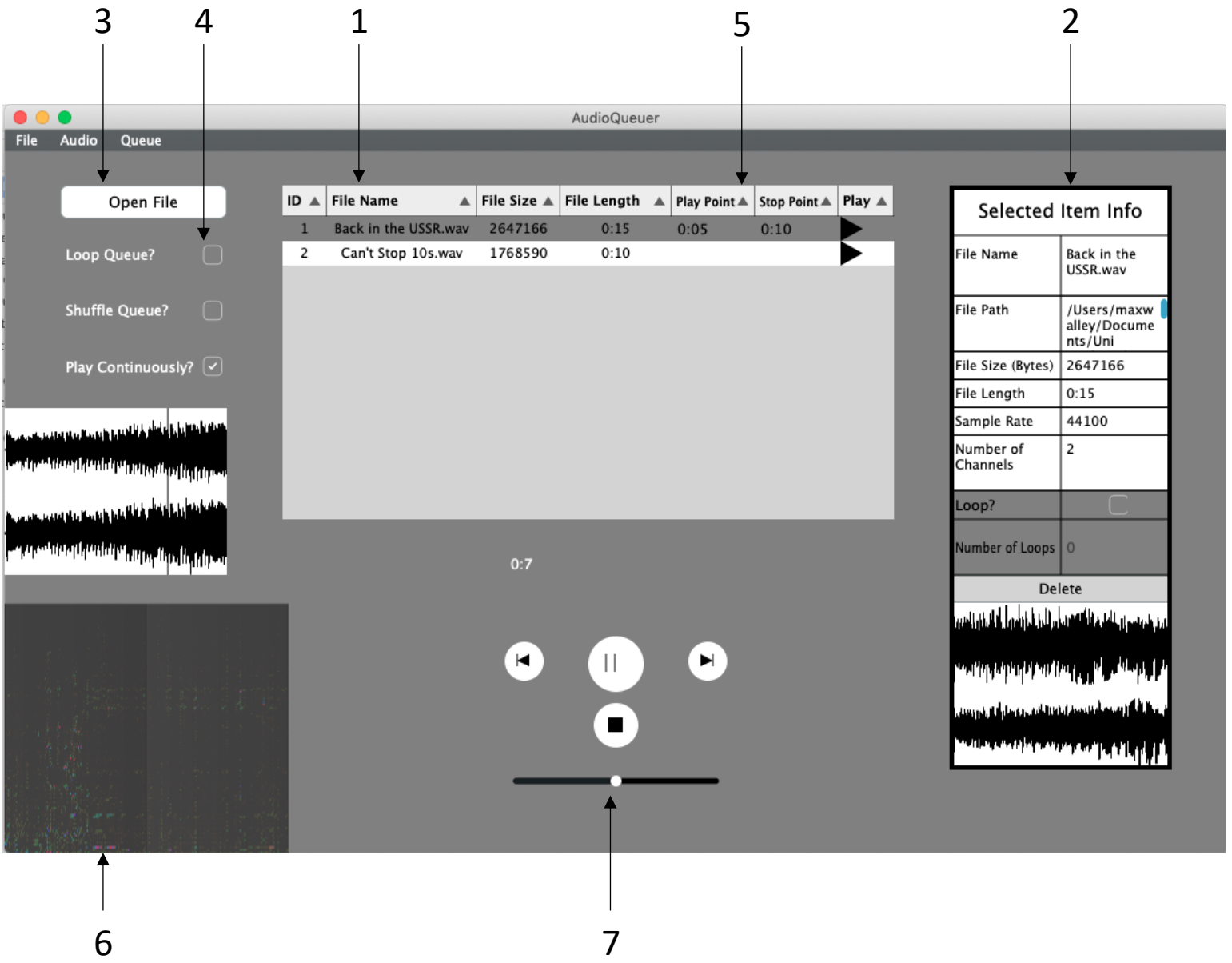
SDA Assignment Written Report - AudioQueuer

Table of Contents

1 - Introduction

The software that has been created is an audio queuing application. This application allows a user to play back audio files in a number of different ways. This allows the application to be useful in several scenarios such as queuing and triggering sound effects for a stage show or to control and playthrough a playlist of audio tracks.

2 - User Manual



1) The Queue Table – This displays every file in the queue. If a file is selected the background colour is changed to grey.

2) Selected Item Info Box – This displays the data of the file that is selected in the table. It also contains controls to manipulate this files playback. If Loop is toggled the files playback will loop however many times are specified in the Number of Loops section. Use the Delete button to delete the file from the queue.

3) Open File Button – This allows files to be added to the queue.

4) Queue Controls – Use these to control and manipulate the playback of the queue. The Loop Queue button toggles whether the queue loops back to the first item once it has played through all items. The Shuffle Queue button toggles whether the queue moves through items in a random or linear order. The Play Continuously button toggles whether the queue stops after every item. If this is toggled the play button will have to be pressed to move onto the next item in the queue.

5) Individual File Controls – These controls allow file playback to be manipulated on an individual level. Use Play Points and Stop Points to specify times when the file starts and stops. The Play button is for playing the file separately from the queue.

6) Playback Data Displays – These two displays show information about the file currently playing. The top display shows a waveform with a playhead running through it. To navigate through the file click on the waveform where you would like playback to move to. The lower display shows a frequency spectrum.

7) Playback Controls – These controls control the playback of the queue. The play/pause button will play and pause the queue. The skip button will move playback to the next file in the queue. The back button will move playback to the previous file in the queue. The stop button will stop the queue and the slider is for controlling volume.

3 - System Documentation

See appendices A and B for related diagrams.

The MainComponent class is Component class that holds the entire GUI for the application. This class also handles sending and receiving data and triggers to other parts of the system.

MainComponent holds an instance of the AudioPlayer class. This inherits from AudioSource and handles the audio playback of the system through inherited methods.

The AudioTable class is a Component class that handles both the table and the selection of the file to play next from the table. Each file on the table has its data held in a QueueItem class. These QueueItems are held in an OwnedArray<> called items. The class also holds an int variable called 'nextIndexToPlay' which is used to allocate which instance from items should be played next. This is done through the Arrays '[]' operator. The file from this instance can then be sent to MainComponent when requested. MainComponent can then send this on to AudioPlayer. This is done using 'items[nextIndexToPlay]->getFile()'. When AudioTable needs to find a new file to play the 'moveNextIndexToPlayOn()' method is called. This changes 'nextIndexToPlay' appropriately.

A similar method is used to fill the infoBox with data. When QueueItem is instantiated, its allocated files data is arranged into an ItemInfo struct. When an item in the queue is selected this struct is sent to the infoBox through its 'changeData()' method. This struct is retrieved by AudioTable in a similar way to a file to play: 'items[currentIndexSelected]->getItemData()'.

4 - Conclusions

Throughout this project I have developed my knowledge in C++ and general programming techniques. This includes areas such as threading and using memory allocated on the heap. Similarly, this project has allowed me to gain experience using object orientated programming and programming using a class based approach.

Towards the end of development, I gained experience documenting code for the first time. This included creating UML diagrams and writing Doxygen comments. These documentation skills will be useful in future projects.

A key problem that was faced during development of this application was adding components to cells in the TableListBox held in the AudioTable class. The TableListBox attempted to take over ownership of these components causing the application to attempt to delete them twice. For an ideal solution a large amount of basic system architecture would have needed to be changed. Due to this error occurring late in development a relatively complex system used this basic architecture as its foundation and thus the ideal solution was not implemented. Instead, a solution using heap memory was implemented.

During the last tests of the application an error arose where a user cannot add a file after previously deleting a file that was not the last in the queue. Due to this error occurring late in development a quick fix was installed that does not allow the user to delete items that are not at the end of the queue.

5 - Future Developments

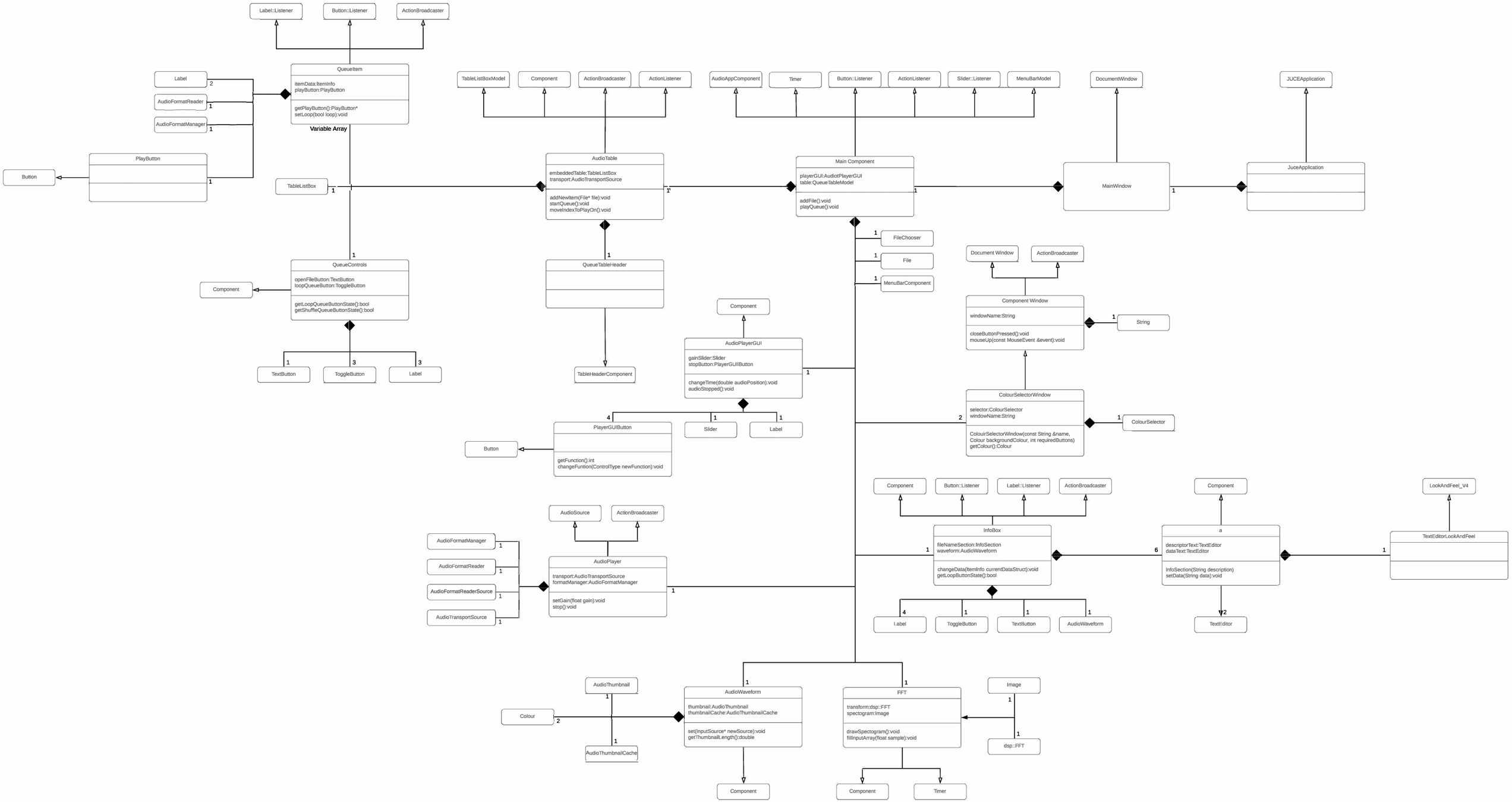
The primary future objective of development will be to allow the user to delete items that are not at the end of the queue.

A key development for the application would be the inclusion of presets and saving. This would allow users to close and reopen the application without losing their queue. This could be achieved using XML which Juce contains a lot of support for.

Finally, a potential development of the system could be to edit the metadata for the files in the queue. This would allow users to add in missing data or change current information.

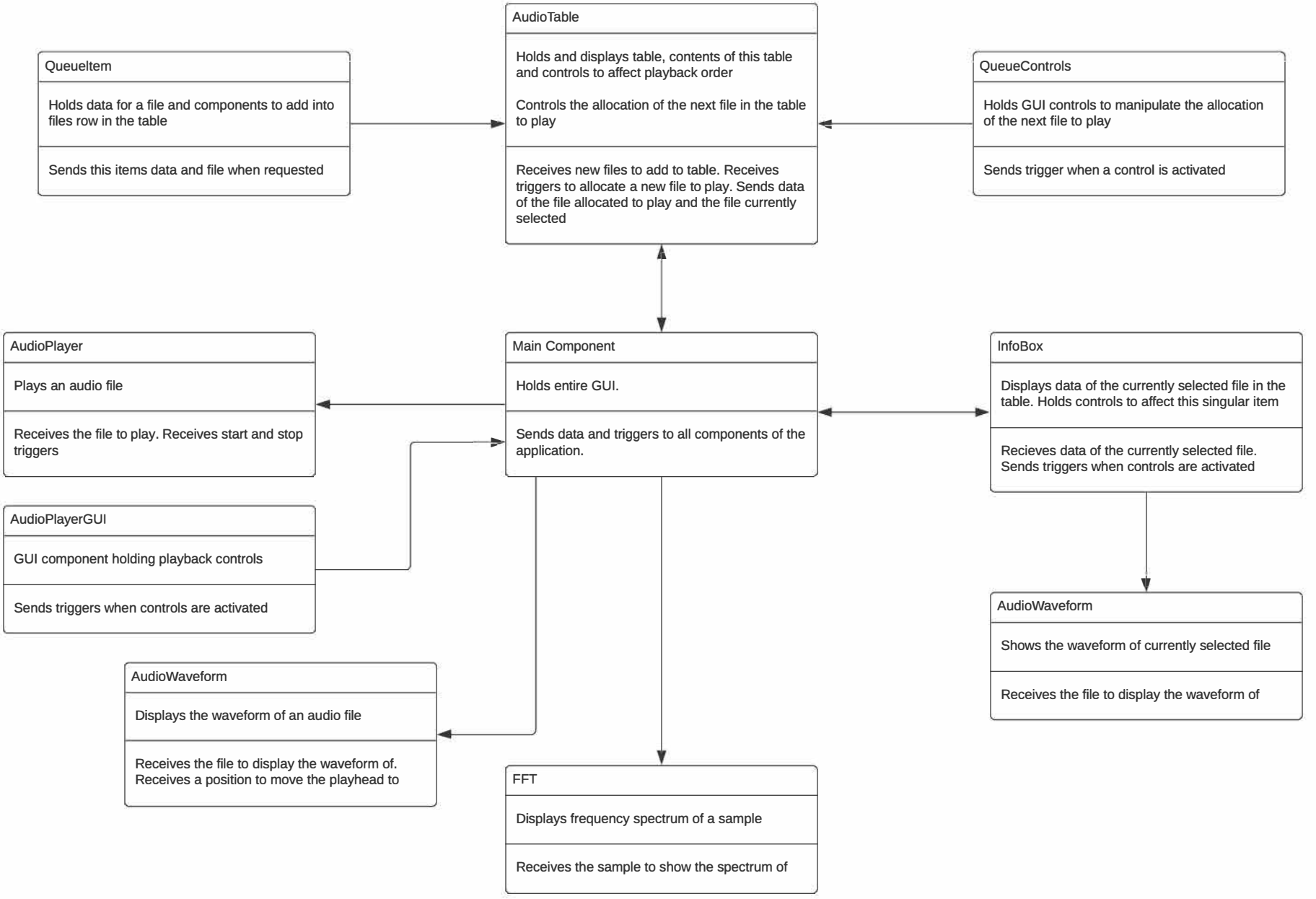
Appendix A - UML Class Relationship Diagram

Max Walley | January 28, 2020



Appendix B - Application Structure Diagram

maximillian2.walley | January 25, 2020



Appendix C - AudioQueuer Doxygen

Max Walley
Version 1
1/28/20 3:54:00 PM

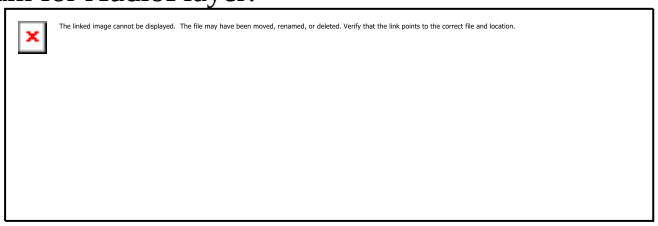
Table of Contents

Class Documentation

AudioPlayer Class Reference

`#include <AudioPlayer.h>`

Inheritance diagram for AudioPlayer:



Public Member Functions

- **AudioPlayer** ()
- **~AudioPlayer** ()
- void **prepareToPlay** (int samplesPerBlockExpected, double sampleRate) override
- void **getNextAudioBlock** (const AudioSourceChannelInfo &bufferToFill) override
- void **releaseResources** () override
- void **loadNewFile** (File *fileToLoad, int playPoint, int stopPoint, bool sendNotificationAtEnd)
- void **pause** ()
- bool **isPlaying** () const
- bool **isPaused** () const
- void **playFromPause** ()
- double **getTransportPosition** () const
- void **setGain** (float gain)
- void **setTransportPosition** (double position)
- double **getTransportLengthInSeconds** () const
- void **stop** ()
- AudioFormatManager * **getAudioFormatManager** ()
- void **setStopPoint** (int newStopPoint)

Detailed Description

A class the holds an audio player. To start playing a file call **loadNewFile()**. If a file is paused call **playFromPause()** rather than **loadNewFile()**.

Constructor & Destructor Documentation

AudioPlayer::AudioPlayer ()

Constructor

AudioPlayer::~AudioPlayer ()

Destructor

Member Function Documentation

AudioFormatManager * AudioPlayer::getAudioFormatManager ()

Returns the AudioFormatManager this player is using

Returns

the AudioFormatManager the player is using

void AudioPlayer::getNextAudioBlock (const AudioSourceChannelInfo & *bufferToFill*) `[override]`

Implementation of the JUCE AudioSource method

double AudioPlayer::getTransportLengthInSeconds () const

Returns the length of the current file loaded into the player in seconds

Returns

the length of the current file loaded into the player in seconds

double AudioPlayer::getTransportPosition () const

Returns the current position of the transport in seconds

Returns

the current position of the transport in seconds

bool AudioPlayer::isPaused () const

Returns whether the player is currently paused or not

Returns

whether the player is paused or not

bool AudioPlayer::isPlaying () const

Returns whether the player is currently playing or not

Returns

whether the player is playing or not

void AudioPlayer::loadNewFile (File * *fileToLoad*, int *playPoint*, int *stopPoint*, bool *sendNotificationAtEnd*)

Loads and plays a new file

Parameters

fileToLoad	the file to load and play. If this is nullptr no file will be loaded.
playPoint	the point in the file to start playing from in seconds. If this is blank transport will start from the start of the file.
stopPoint	the point in the file to stop playing at in seconds. If this is blank transport will end at the end of the file.
sendNotificationAtEnd	whether to send a notification in the form of an ActionMessage at the end of playback. This message is "Transport Finished".

void AudioPlayer::pause ()

Pauses the transport

See also

playFromPause

void AudioPlayer::playFromPause ()

Plays the transport from a pause. If the transport is not paused this will do nothing

void AudioPlayer::prepareToPlay (int *samplesPerBlockExpected*, double *sampleRate*) `[override]`

Implementation of the JUCE AudioSource method

void AudioPlayer::releaseResources () `[override]`

Implementation of the JUCE AudioSource Method

void AudioPlayer::setGain (float *gain*)

Sets the gain of the player

Parameters

gain	the new gain multiplier to apply to audio processed by the player

void AudioPlayer::setStopPoint (int *newStopPoint*)

Sets the stop point of the file playing

Parameters

newStopPoint	the new stop point to use in seconds

void AudioPlayer::setTransportPosition (double *position*)

Sets the transport position at a new position in the current audio file being played

Parameters

position	the new position to move the transport to, in seconds

void AudioPlayer::stop ()

Stops the player playing

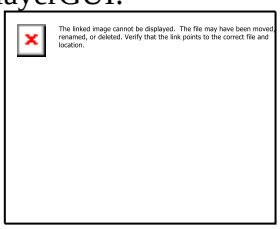
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioPlayer.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioPlayer.cpp**

AudioPlayerGUI Class Reference

`#include <AudioPlayerGUI.h>`

Inheritance diagram for AudioPlayerGUI:



Public Member Functions

- **AudioPlayerGUI** ()
- **~AudioPlayerGUI** ()
- void **paint** (Graphics &) override
- void **resized** () override
- void **audioStopped** ()
- void **audioPaused** ()
- void **audioPlayed** ()
- void **changeTime** (double audioPosition)
- void **setButtonEnabled** (Button *button, bool enabled)

Public Attributes

- Slider **gainSlider**
- **PlayerGUIButton playPauseButton**
- **PlayerGUIButton stopButton**
- **PlayerGUIButton nextButton**
- **PlayerGUIButton lastButton**

Detailed Description

A component class that holds a basic GUI for an audio player. Holds a play/pause button, a skip button, a go back button, a stop button, a gain slider and a label showing a time. To receive callbacks when the buttons or slider are changed create listeners to them. Change the time shown on the label with **changeTime()**.

Constructor & Destructor Documentation

AudioPlayerGUI::AudioPlayerGUI ()

Constructor

AudioPlayerGUI::~AudioPlayerGUI ()

Destructor

Member Function Documentation

void AudioPlayerGUI::audioPaused ()

Changes components of this class to show a state where audio is paused

void AudioPlayerGUI::audioPlayed ()

Changes components of this class to show a state where audio is playing

void AudioPlayerGUI::audioStopped ()

Changes components of this class to show a state where audio is stopped

void AudioPlayerGUI::changeTime (double *audioPosition*)

Changes the time shown by the time label

Parameters

audioPosition	current audio position in seconds

void AudioPlayerGUI::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void AudioPlayerGUI::resized () `[override]`

Implementation of the JUCE Component method

void AudioPlayerGUI::setButtonEnabled (Button * *button*, bool *enabled*)

Changes whether a button on the component is enabled

Parameters

button	the button to change whether its enabled
enabled	whether its enabled

Member Data Documentation

Slider AudioPlayerGUI::gainSlider

PlayerGUIButton AudioPlayerGUI::lastButton

PlayerGUIButton AudioPlayerGUI::nextButton

PlayerGUIButton AudioPlayerGUI::playPauseButton

PlayerGUIButton AudioPlayerGUI::stopButton

The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioPlayerGUI.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioPlayerGUI.cpp**

AudioTable Class Reference

`#include <AudioTable.h>`

Inheritance diagram for AudioTable:

Public Member Functions

- **AudioTable** ()
- **~AudioTable** ()
- void **paint** (Graphics &g) override
- void **resized** () override

- void **addNewItem** (File *file)
- void **deleteSelectedItem** ()
- int **getSelectedRow** ()
- void **moveIndexToPlayOn** ()
- void **moveIndexToPlayBack** ()
- void **startQueue** ()
- int **getCurrentPlayPoint** () const
- int **getCurrentStopPoint** () const
- File * **getFileToPlay** () const
- **ItemInfo getCurrentPlayingDataStruct** () const
- **ItemInfo getCurrentSelectedDataStruct** () const
- void **updateSelectedItemLoopToggle** (bool newLoopToggle)
- void **updateSelectedItemNumLoops** (int newNumLoops)
- void **changeQueueControlToggle** (int control)
- void **reset** ()
- bool **isEmpty** ()
- bool **isRowSelected** ()

Public Attributes

- **QueueControls queueControls**
- OwnedArray< **QueueItem**, CriticalSection > **items**

Detailed Description

A component class that displays a table made for holding audio files and a number of controls. This class also works out which item of the table should be being played and returns a pointer to the file from that item with **getFileToPlay()**. To initilize it to start running through the table call **startQueue()**. This will cause **getFileToPlay()** to return a pointer to the first file. After playback of a file has completed call **moveIndexToPlayOn()**. This will set **getFileToPlay()** to return a pointer to the next file to play.

Constructor & Destructor Documentation

AudioTable::AudioTable ()

Constructor

AudioTable::~AudioTable ()

Destructor

Member Function Documentation

void AudioTable::addNewItem (File * *file*)

Adds a new item to the array of items and displays it in the table

Parameters

file	the file to add

void AudioTable::changeQueueControlToggle (int *control*)

Changes the value of a toggle control in the Queue Controls

Parameters

control	- the control to toggle. Set using ControlToggleButtons enum in **QueueControls**

See also

ControlToggleButtons

void AudioTable::deleteSelectedItem ()

Deletes the currently selected item from the array of items and from the table

ItemInfo AudioTable::getCurrentPlayingDataStruct () const

Returns the info of the current **QueueItem** to play in an **ItemInfo** struct

Returns

the info of the current **QueueItem** to play in an **ItemInfo** struct

int AudioTable::getCurrentPlayPoint () const

Returns the play point of the current **QueueItem** to play

Returns

the play point of the current **QueueItem** to play

ItemInfo AudioTable::getCurrentSelectedDataStruct () const

Returns the info of the current **QueueItem** selected in the table in an **ItemInfo** struct

Returns

the info of the current **QueueItem** selected in an **ItemInfo** struct

int AudioTable::getCurrentStopPoint () const

Returns the stop point of the current **QueueItem** to play

Returns

the stop point of the current **QueueItem** to play

File * AudioTable::getFileToPlay () const

Returns the file to play from specific **QueueItem** that should play next. If there is no item to play next this will return nullptr

Returns

the file to play from specific **QueueItem** that should play next

int AudioTable::getSelectedRow ()

Returns the row number of the current selected row

Returns

the row number of the current selected row

bool AudioTable::isEmpty ()

Returns whether the table is empty

Returns

whether the table is empty

bool AudioTable::isRowSelected ()

Returns whether a row is currently selected

Returns

whether a row is selected

void AudioTable::moveIndexToPlayBack ()

Moves the index to play back to the last **QueueItem** in the table. If the current playing index is the first in the table this will reset the index to play to nothing

void AudioTable::moveIndexToPlayOn ()

Moves the index to play onto the next **QueueItem** in the table, taking into account the toggle buttons on **QueueControls**

void AudioTable::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void AudioTable::reset ()

Sets the index to play back to nothing

void AudioTable::resized () `[override]`

Implementation of the JUCE Component method

void AudioTable::startQueue ()

Initilises next index to play in a variety of ways depending on **QueueControls**

void AudioTable::updateSelectedItemLoopToggle (bool *newLoopToggle*)

Changes the loop toggle of the **ItemInfo** struct of the currently selected item. Essentially changing whether the selected **QueueItem** loops or not

Parameters

newLoopToggle	the new loop toggle selection

void AudioTable::updateSelectedItemNumLoops (int *newNumLoops*)

Changes the amount of loops the currently selected **QueueItem** does by updating the number of loops in the items **ItemInfo** struct

Parameters

newNumLoops	the new amount of loops for this **QueueItem** to perform

---

Member Data Documentation

OwnedArray<QueueItem, CriticalSection> AudioTable::items

This array holds all the items in the table

QueueControls AudioTable::queueControls

The documentation for this class was generated from the following files:

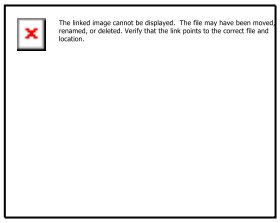
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioTable.h**

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioTable.cpp**

AudioWaveform Class Reference

`#include <AudioWaveform.h>`

Inheritance diagram for AudioWaveform:



Public Member Functions

- **AudioWaveform** (AudioFormatManager &formatManagerToUse)
- **~AudioWaveform** ()
- void **paint** (Graphics &) override
- void **set** (InputSource *newSource)
- double **getThumbnailLength** () const
- void **clear** ()
- void **moveTransportLine** (double xPixelToMoveTo)
- void **setBackgroundColour** (Colour newColour)
- void **setWaveformColour** (Colour newColour)

Detailed Description

A component class for displaying an audio waveform. Set the file for it to show using the **set()** method.

Constructor & Destructor Documentation

AudioWaveform::AudioWaveform (AudioFormatManager & *formatManagerToUse*)

AudioWaveform::~AudioWaveform ()

Member Function Documentation

void AudioWaveform::clear ()

Clears the waveform

double AudioWaveform::getThumbnailLength () const

Returns the length of the file the thumbnail is showing in seconds

Returns

the length of the file the thumbnail is showing

void AudioWaveform::moveTransportLine (double *xPixelToMoveTo*)

Moves the transport line to a new position on the waveform

Parameters

xPixelToMoveTo	the new position on the waveform to move to in pixels

void AudioWaveform::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void AudioWaveform::set (InputSource * *newSource*)

Sets the waveform to show a source

Parameters

newSource	the source to show

void AudioWaveform::setBackgroundColour (Colour *newColour*)

Changes the background colour of the waveform

Parameters

newColour	the new colour to change the background to

void AudioWaveform::setWaveformColour (Colour *newColour*)

Changes the waveform colour of the waveform

Parameters

newColour	the new colour to change the waveform to

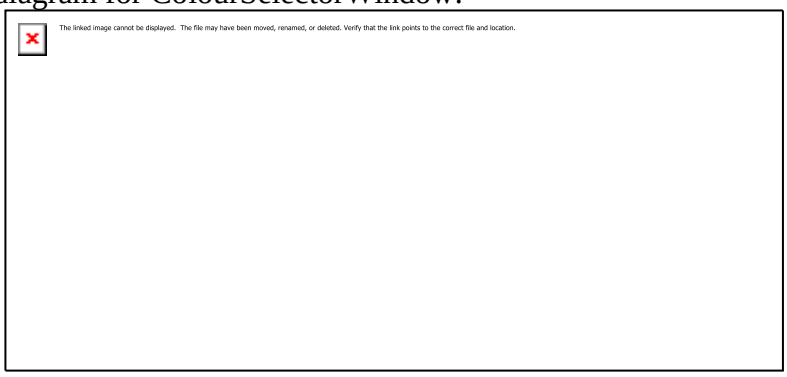
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioWaveform.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**AudioWaveform.cpp**

ColourSelectorWindow Class Reference

`#include <ColourSelectorWindow.h>`

Inheritance diagram for ColourSelectorWindow:



Public Member Functions

- **ColourSelectorWindow** (const String &name, Colour backgroundColour, int requiredButtons)
- **~ColourSelectorWindow** ()
- Colour **getColour** () const

Detailed Description

A **ComponentWindow** that holds a ColourSelector component. Sends an action message when the window is clicked or the mouse is released however to implement this the owned component must have this as a mouse listener. You can then call **getColour()** to find the colour the user selected

---

Constructor & Destructor Documentation

ColourSelectorWindow::ColourSelectorWindow (const String & *name*, Colour *backgroundColour*, int *requiredButtons*)

Constructor

Parameters

name	the name of the window
backgroundColour	the background colour of the window
requiredButtons	the buttons to put in the top left hand corner of the window

See also

DocumentWindow()

ColourSelectorWindow::~ColourSelectorWindow ()

Destructor

Member Function Documentation

Colour ColourSelectorWindow::getColour () const

Returns the colour currently selected by the user

Returns

the colour currently selected

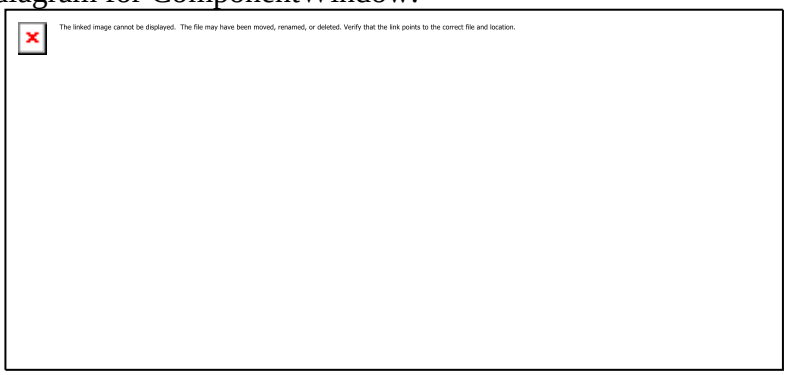
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**ColourSelectorWindow.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**ColourSelectorWindow.cpp**

ComponentWindow Class Reference

```
#include <ComponentWindow.h>
```

Inheritance diagram for ComponentWindow:



Public Member Functions

- **ComponentWindow** (const String &name, Colour backgroundColour, int requiredButtons)
- **~ComponentWindow** ()

Detailed Description

A window that holds a component. Almost exactly the same as the DocumentWindow class however, when the close button is pressed on this class the window will close. Sends an action message when the window is clicked or the mouse is released however to implement this the owned component must have this as a mouse listener

---

Constructor & Destructor Documentation

ComponentWindow::ComponentWindow (const String & *name*, Colour *backgroundColour*, int *requiredButtons*)

Constructor

Parameters

name	the name of the window
backgroundColour	the background colour of the window
requiredButtons	the buttons to put in the top left hand corner of the window

See also

DocumentWindow()

ComponentWindow::~ComponentWindow ()

Destructor

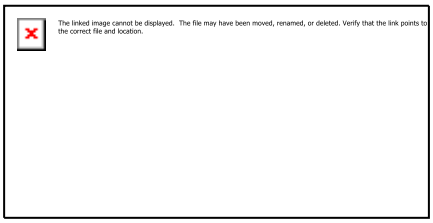
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**ComponentWindow.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**ComponentWindow.cpp**

FFT Class Reference

`#include <FFT.h>`

Inheritance diagram for FFT:



Public Member Functions

- **FFT** ()
- **~FFT** ()
- void **paint** (Graphics &) override
- void **fillInputArray** (float sample)

Detailed Description

A component class that displays a frequency spectrum. Takes input sample by sample through the fillIInputArray() method. This will start drawing as soon as it is initilised.

Constructor & Destructor Documentation

FFT::FFT ()

Constructor

FFT::~FFT ()

Destructor

Member Function Documentation

void FFT::fillInputArray (float *sample*)

Inputs data into the **FFT**

Parameters

sample	the input sample to put into the **FFT**

void FFT::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**FFT.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**FFT.cpp**

InfoBox Class Reference

`#include <InfoBox.h>`

Inheritance diagram for InfoBox:

Public Member Functions

- **InfoBox** (AudioFormatManager &manager)
- **~InfoBox** ()
- void **paint** (Graphics &g) override
- void **paintOverChildren** (Graphics &g) override
- void **resized** () override
- void **changeData** (**ItemInfo** currentDataStruct)
- bool **getLoopButtonState** () const
- int **getNewNumLoops** () const
- void **clear** ()

Detailed Description

A component class that displays a box holding data for an audio file. This will be initised to a blank display. To change the data it displays call **changeData()**.

Constructor & Destructor Documentation

InfoBox::InfoBox (AudioFormatManager & *manager*)

Constructor

Parameters

manager	the AudioFormatManager to use

InfoBox::~InfoBox ()

Destructor

Member Function Documentation

void InfoBox::changeData (ItemInfo *currentDataStruct*)

Changes the data that is shown in the info box

Parameters

currentDataStruct	the data to show

void InfoBox::clear ()

Clears all the data shown in the infobox

bool InfoBox::getLoopButtonState () const

Returns the loop button state of the current data being shown

Returns

the loop button state

int InfoBox::getNewNumLoops () const

Returns number of loops of the current data being shown

Returns

the number of loops

void InfoBox::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void InfoBox::paintOverChildren (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void InfoBox::resized () `[override]`

Implementation of the JUCE Component method

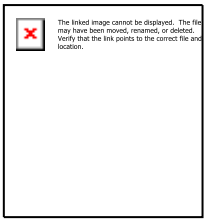
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**InfoBox.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**InfoBox.cpp**

InfoSection Class Reference

`#include <InfoSection.h>`

Inheritance diagram for InfoSection:



Public Member Functions

- **InfoSection** (String description)
- **~InfoSection** ()
- void **paint** (Graphics &g) override
- void **paintOverChildren** (Graphics &g) override
- void **resized** () override
- void **setData** (String data)
- void **clear** ()

Detailed Description

A component class that makes up a section of the **InfoBox** component class. Holds two TextEditors. One holding data and the other holding a description of the data. Set the data with **setData()**.

Constructor & Destructor Documentation

InfoSection::InfoSection (String *description*)

Constructor

Parameters

description	the text to put in the description TextEditor

InfoSection::~InfoSection ()

Destructor

Member Function Documentation

void InfoSection::clear ()

Clears both TextEditors

void InfoSection::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void InfoSection::paintOverChildren (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void InfoSection::resized () `[override]`

Implementation of the JUCE Component method

void InfoSection::setData (String *data*)

Sets the text in the data TextEditor

Parameters

data	the text to put in the data TextEditor

The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**InfoSection.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**InfoSection.cpp**

ItemInfo Struct Reference

`#include <ItemDataStruct.h>`

Public Attributes

- int **itemIndex**
- File **file**
- String **fileName**
- int64_t **size**
- int64_t **lengthInSamples**
- double **sampleRate**
- int **numChannels**
- int **lengthInSecs**
- String **lengthInTime**
- bool **loop**
- int **numLoops**

Member Data Documentation

File ItemInfo::file

String ItemInfo::fileName

int ItemInfo::itemIndex

int64_t ItemInfo::lengthInSamples

int ItemInfo::lengthInSecs

String ItemInfo::lengthInTime

bool ItemInfo::loop

int ItemInfo::numChannels

int ItemInfo::numLoops

double ItemInfo::sampleRate

int64_t ItemInfo::size

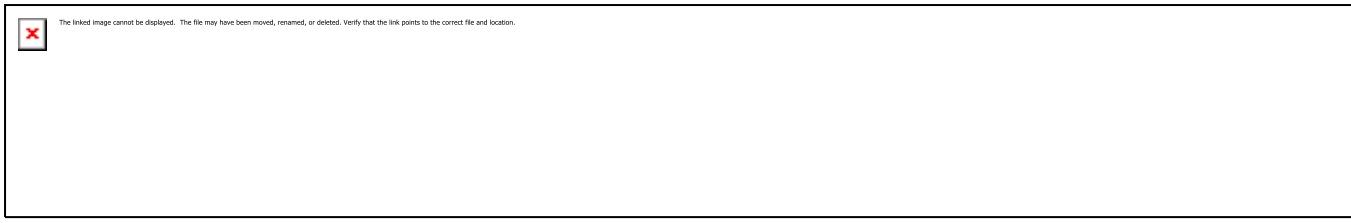
The documentation for this struct was generated from the following file:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**ItemDataStruct.h**

MainComponent Class Reference

`#include <MainComponent.h>`

Inheritance diagram for MainComponent:



Public Member Functions

- **MainComponent** ()
- **~MainComponent** ()
- void **prepareToPlay** (int samplesPerBlockExpected, double sampleRate) override
- void **getNextAudioBlock** (const AudioSourceChannelInfo &bufferToFill) override
- void **releaseResources** () override
- void **paint** (Graphics &g) override
- void **resized** () override
- StringArray **getMenuBarNames** () override
- PopupMenu **getMenuForIndex** (int topLevelMenuIndex, const String &menuName) override
- void **menuItemSelected** (int menuItemID, int topLevelMenuIndex) override

---

Detailed Description

The main component of the application. Filling all of the MainWindow. This holds every other component on the MainWindow.

Constructor & Destructor Documentation

MainComponent::MainComponent ()

Constructor

MainComponent::~MainComponent ()

Destructor

Member Function Documentation

StringArray MainComponent::getMenuBarNames () `[override]`

Implementation of the JUCE MenuBarModel method

PopupMenu MainComponent::getMenuForIndex (int *topLevelMenuIndex*, const String & *menuName*) `[override]`

Implementation of the JUCE MenuBarModel method

void MainComponent::getNextAudioBlock (const AudioSourceChannelInfo & *bufferToFill*) `[override]`

Implementation of the JUCE AudioSource method

void MainComponent::menuItemSelected (int *menuItemID*, int *topLevelMenuIndex*) `[override]`

Implementation of the JUCE MenuBarModel method

void MainComponent::paint (Graphics & *g*) `[override]`

Implementation of the JUCE Component method

void MainComponent::prepareToPlay (int *samplesPerBlockExpected*, double *sampleRate*) `[override]`

Implementation of the JUCE AudioSource method

void MainComponent::releaseResources () `[override]`

Implementation of the JUCE AudioSource method

void MainComponent::resized () `[override]`

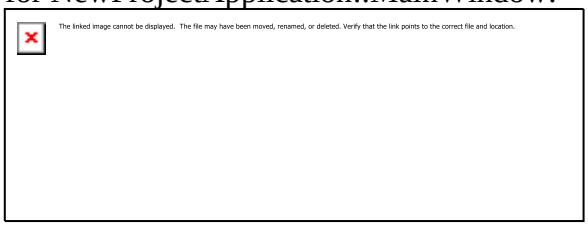
Implementation of the JUCE Component method

The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**MainComponent.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**MainComponent.cpp**

NewProjectApplication::MainWindow Class Reference

Inheritance diagram for NewProjectApplication::MainWindow:



Public Member Functions

- **MainWindow** (String name)
- void **closeButtonPressed** () override

Constructor & Destructor Documentation

NewProjectApplication::MainWindow::MainWindow (String *name*) `[inline]`

Member Function Documentation

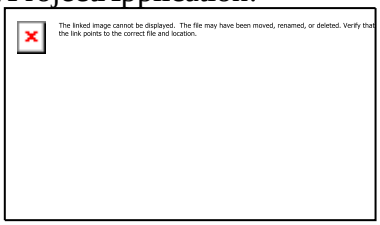
void NewProjectApplication::MainWindow::closeButtonPressed () `[inline], [override]`

The documentation for this class was generated from the following file:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**Main.cpp**

NewProjectApplication Class Reference

Inheritance diagram for NewProjectApplication:



Classes

- class **MainWindow**

Public Member Functions

- **NewProjectApplication** ()
- const String **getApplicationName** () override
- const String **getApplicationVersion** () override
- bool **moreThanOneInstanceAllowed** () override
- void **initialise** (const String &commandLine) override
- void **shutdown** () override
- void **systemRequestedQuit** () override
- void **anotherInstanceStarted** (const String &commandLine) override

---

Constructor & Destructor Documentation

NewProjectApplication::NewProjectApplication () `[inline]`

Member Function Documentation

void NewProjectApplication::anotherInstanceStarted (const String & *commandLine*) `[inline]`, `[override]`

const String NewProjectApplication::getApplicationName () `[inline]`, `[override]`

const String NewProjectApplication::getApplicationVersion () `[inline]`, `[override]`

void NewProjectApplication::initialise (const String & *commandLine*) `[inline]`, `[override]`

bool NewProjectApplication::moreThanOneInstanceAllowed () `[inline]`, `[override]`

void NewProjectApplication::shutdown () `[inline]`, `[override]`

void NewProjectApplication::systemRequestedQuit () `[inline]`, `[override]`

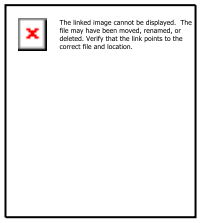
The documentation for this class was generated from the following file:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**Main.cpp**

PlayButton Class Reference

`#include <PlayButton.h>`

Inheritance diagram for PlayButton:



Public Member Functions

- **PlayButton** ()
- **~PlayButton** ()
- void **paintButton** (Graphics &g, bool shouldDrawButtonAsHighlighted, bool shouldDrawButtonAsDown) override

---

Detailed Description

A component button class that takes the form of a play triangle.

Constructor & Destructor Documentation

PlayButton::PlayButton ()

Constructor

PlayButton::~PlayButton ()

Destructor

Member Function Documentation

void PlayButton::paintButton (Graphics & *g*, bool *shouldDrawButtonAsHighlighted*, bool *shouldDrawButtonAsDown*) `[override]`

Implementation of the JUCE Button method

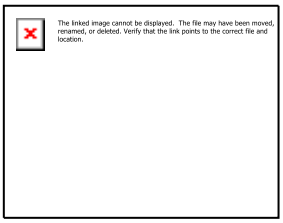
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**PlayButton.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**PlayButton.cpp**

PlayerGUIButton Class Reference

`#include <PlayerGUIButton.h>`

Inheritance diagram for PlayerGUIButton:



Public Types

- enum **ControlType** { **play**, **pause**, **stop**, **next**, **last** }

Public Member Functions

- **PlayerGUIButton** (**ControlType** function)
- **~PlayerGUIButton** ()
- void **paintButton** (Graphics &g, bool shouldDrawButtonAsHighlighted, bool shouldDrawButtonAsDown) override
- void **changeFunction** (**ControlType** newFunction)
- int **getFunction** () const

Detailed Description

A component button class for drawing multiple types of audio player buttons.

Member Enumeration Documentation

enum PlayerGUIButton::ControlType

Enumerator:

play	
pause	
stop	
next	
last	

Constructor & Destructor Documentation

PlayerGUIButton::PlayerGUIButton (ControlType *function*)

Constructor

Parameters

function	the function this button will represent

PlayerGUIButton::~PlayerGUIButton ()

Destructor

Member Function Documentation

void PlayerGUIButton::changeFunction (ControlType *newFunction*)

Changes the function this button represents

Parameters

newFunction	the new function this button should represent

int PlayerGUIButton::getFunction () const

Returns the function this button represents

Returns

the function this button represents

See also

changeFunction

void PlayerGUIButton::paintButton (Graphics & *g*, bool *shouldDrawButtonAsHighlighted*, bool *shouldDrawButtonAsDown*) `[override]`

Implementation of the JUCE Component method

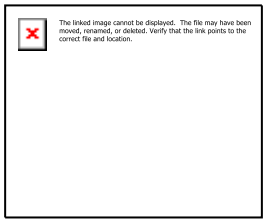
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**PlayerGUIButton.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**PlayerGUIButton.cpp**

QueueControls Class Reference

`#include <QueueControls.h>`

Inheritance diagram for QueueControls:



Public Types

- enum **ControlToggleButtons** { **loopQueue** = 1, **shuffleQueue** = 2, **playContinuously** = 3 }

Public Member Functions

- **QueueControls** ()
- **~QueueControls** ()
- void **paint** (Graphics &g) override
- void **resized** () override
- bool **getLoopQueueButtonState** ()
- bool **getShuffleQueueButtonState** ()
- bool **getContinuousButtonState** ()

- void **changeToggleState** (int control)

Public Attributes

- TextButton **openFileButton**

Detailed Description

A component class that hold GUI components to control an **AudioTable** class.

Member Enumeration Documentation

enum QueueControls::ControlToggleButtons

The toggle buttons held by this component

See also

changeToggleState

Enumerator:

loopQueue	
shuffleQueue	
playContinuously	

Constructor & Destructor Documentation

QueueControls::QueueControls ()

Constructor

QueueControls::~QueueControls ()

Destructor

Member Function Documentation

void QueueControls::changeToggleState (int *control*)

Changes the toggle state of one of the controls

Parameters

control	the control to change the toggle state of

bool QueueControls::getContinuousButtonState ()

Returns the state of the play continuously toggle button

Returns

the state of the continuous toggle button

bool QueueControls::getLoopQueueButtonState ()

Returns the state of the loop queue toggle button

Returns

the state of the loop queue toggle button

bool QueueControls::getShuffleQueueButtonState ()

Returns the state of the shuffle queue toggle button

Returns

the state of the shuffle queue toggle button

void QueueControls::paint (Graphics & *g*)`[override]`

Implementation of the JUCE Component method

void QueueControls::resized ()`[override]`

Implementation of the JUCE Component method

Member Data Documentation

TextButton QueueControls::openFileButton

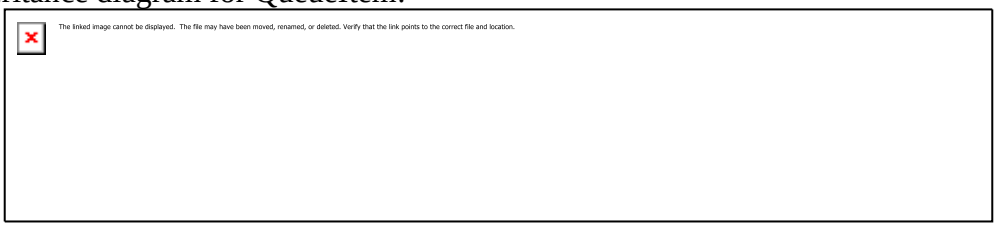
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**QueueControls.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**QueueControls.cpp**

QueueItem Class Reference

`#include <QueueItem.h>`

Inheritance diagram for QueueItem:



Public Member Functions

- **QueueItem** (int idNum, File *file)
- **~QueueItem** ()
- void **setFile** (File *file)
- File * **getFile** ()
- void **setItemIndex** (int index)
- int **getItemIndex** () const
- String **getFileName** () const
- int64_t **getFileSize** () const
- String **getLengthInTime** ()
- Label * **getPlayTimeLabel** ()

- Label * **getStopTimeLabel** ()
- **PlayButton** * **getPlayButton** ()
- int **getPlayPoint** () const
- int **getStopPoint** () const
- **ItemInfo getItemData** () const
- void **setLoop** (bool loop)
- bool **getLoop** () const
- void **setNumLoops** (int numLoops)
- int **getNumLoops** () const

Detailed Description

A class for holding data for a singular item on the table in the **AudioTable** class. This also holds the components for the tables component cells.

Constructor & Destructor Documentation

QueueItem::QueueItem (int *idNum*, File * *file*)

Constructor

Parameters

idNum	the index number that this **QueueItem** will have
file	the file this **QueueItem** will hold. This can also be set in setFile

See also

setItemIndex

setFile

QueueItem::~QueueItem ()

Destructor

Member Function Documentation

File * QueueItem::getFile ()

Returns the file this **QueueItem** holds

Returns

the file this **QueueItem** holds

See also

setFile

String QueueItem::getFileName () const

Returns the name of the file this **QueueItem** is associated with

Returns

the name of the file this **QueueItem** is associated with

int64_t QueueItem::getFileSize () const

Returns the size of the file this **QueueItem** is associated with

Returns

the size of the file this **QueueItem** is associated with

ItemInfo QueueItem::getItemData () const

Returns the item data for this **QueueItem** as an **ItemInfo** struct

Returns

the item data for this **QueueItem** as an **ItemInfo** struct

int QueueItem::getItemIndex () const

Returns the index of this **QueueItem**

Returns

the index of this **QueueItem**

See also

setItemIndex

String QueueItem::getLengthInTime ()

Returns the length of the file associated with this **QueueItem** as a string

Returns

the length of the file associated with this **QueueItem** as a string

bool QueueItem::getLoop () const

Returns whether this **QueueItem** is set to loop

Returns

whether this **QueueItem** is set to loop

See also

setLoop

int QueueItem::getNumLoops () const

Returns the amount of times this **QueueItem** is set to loop

Returns

the amount of times this **QueueItem** is set to loop

See also

setNumLoops

PlayButton * QueueItem::getPlayButton ()

Returns the play button

Returns

the play button

int QueueItem::getPlayPoint () const

Returns the play point set for this **QueueItem** in seconds

Returns

the play point for this **QueueItem** in seconds

Label * QueueItem::getPlayTimeLabel ()

Returns the play time label

Returns

the play time label

int QueueItem::getStopPoint () const

Returns the stop point for this **QueueItem** in seconds

Returns

the stop point for this **QueueItem** in seconds

Label * QueueItem::getStopTimeLabel ()

Returns the stop time label

@Return the stop time label

void QueueItem::setFile (File * *file*)

Sets the file this **QueueItem** will hold

Parameters

file	the file this **QueueItem** will hold

void QueueItem::setItemIndex (int *index*)

Sets the index of this **QueueItem**

Parameters

index	the new index to set this **QueueItem** as

void QueueItem::setLoop (bool *loop*)

Sets whether this **QueueItem** loops or not

Parameters

loop	whether to loop this **QueueItem**

void QueueItem::setNumLoops (int *numLoops*)

Sets the amount of times this **QueueItem** should loop

Parameters

numLoops	the number of loops this **QueueItem** should do

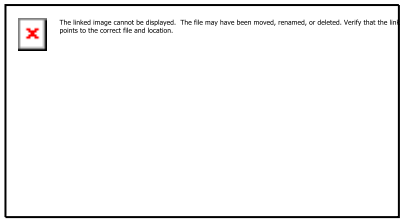
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**QueueItem.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**QueueItem.cpp**

QueueTableHeader Class Reference

`#include <QueueTableHeader.h>`

Inheritance diagram for QueueTableHeader:



Public Member Functions

- **QueueTableHeader** ()
- **~QueueTableHeader** ()

Detailed Description

Class that acts as the header for the TableListBox in the **AudioTable** class.

Constructor & Destructor Documentation

QueueTableHeader::QueueTableHeader ()

Constructor

QueueTableHeader::~QueueTableHeader ()

Destructor

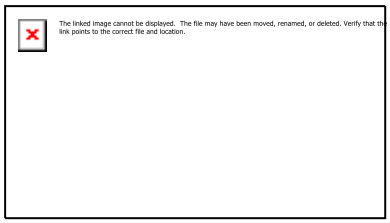
The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**QueueTableHeader.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**QueueTableHeader.cpp**

TextEditorLookAndFeel Class Reference

`#include <InfoSection.h>`

Inheritance diagram for TextEditorLookAndFeel:



Detailed Description

A class to control the look of an info section

The documentation for this class was generated from the following files:

- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**InfoSection.h**
- /Users/maxwalley/Documents/Uni Work/3rd Year/SDA-dev/Project/Source/**InfoSection.cpp**