National College of Ireland

Subtitilizer

Software Development

2022/2023

Lloyd Portes

x19509219

BSHCSD4

x19509219@student.ncirl.ie

Technical Report

Contents

Executive Summary 2

1.0 Introduction 2

1.1. Background 2

1.2. Aims ... 2
1.3. Technology ... 3
1.4. Structure ... 3
2.0 System ... 3
2.1. Requirements ... 3
2.1.1. Functional Requirements ... 3
2.1.1.1. Use Case Diagram ... 4
Requirement 1: Input Video ... 4
Description & Priority ... 4
Use Case ... 4
Requirement 2: Input Language ... 6
Description & Priority ... 6
Use Case ... 6
Requirement 3: Output Language ... 7
Description & Priority ... 7
Use Case ... 7
Requirement 4: Export Video ... 8
Description & Priority ... 8
Use Case ... 8
2.1.2. Data Requirements ... 10
2.1.3. User Requirements ... 10
2.1.4. Environmental Requirements ... 10
2.1.5. Usability Requirements ... 10
2.2. Design & Architecture ... 10
2.3. Implementation ... 15
2.3.1 Code with GUI ... 15
2.3.2 Code without GUI ... 16
2.4. Graphical User Interface (GUI) ... 39
2.5. Testing ... 42
3.0 Conclusion ... 45
4.0 Further Development or Research ... 46
Appendices ... 47
Objectives ... 47
Background ... 47
State of the Art ... 47
Data ... 47
Methodology & Analysis ... 48
Technical Details ... 48

Project Plan 48
Testing 49
Github repo: https://github.com/LloydPortes2/FYP-Subtitilizer 49
Project proposal 50
3.0 Objectives 51
4.0 Background 52
5.0 State of the Art 52
6.0 Data 52
7.0 Methodology & Analysis 52
8.0 Technical Details 53
9.0 Project Plan 53
10.0 Testing 53
Reflective Journal 55

Executive Summary

In my project I will introduce how I have planned how to create Subtitilizer. Creating the project from scratch was difficult but was manageable due to the documentation that is online. I will state the processes for my use case diagrams, requirements, my objectives and my planned work flow to overcome the obstacles of learning C++ and learning FFmpeg's extensive library.

As of now during the mid-point presentation date, learning C++ and FFmpeg while being restricted with time is challenging but very rewarding as you will see throughout the report of how creating implementing a GUI's and making sure everything works coherently.

The design of my software will explain how the algorithms complement each other with the creation of the videos. All videos get dissected into smaller parts which will then be manipulated by the software and then placed back together.

1.0 Introduction

1.1. Background

The reason why I undertook this project was because there were videos online that I was looking for but never had the subtitles. Most of the time, they were speaking in a language I did not understand. I wanted to solve my problem.

1.2. Aims

The project aims to achieve when a video is processed through the software, there will be subtitles on top of the video you have chosen. The user will be able to have the subtitles in a given language they choose.

1.3. Technology

I have chosen to use Google's API for audio to transcription and translation. I will use them by implementing their online code and attach it to my software. The program will first take the audio, use Google's transcription, then translate the file. Afterwards, my program will place the transcription file on top of the video.

For encoding and decoding the video, I have chosen an open software called FFMPEG. I will write code based on the library they provide. From the video size to audio. I will omit features that change the video such as colour correction as the main point of the software is to add subtitles only.

1.4. Structure

Provide a brief overview of the structure of the document and what is addressed in each section.

I will be talking about the headings Systems and Proposal. In My Systems heading, I address all the requirements I have to enable the functionality and logic with how they work along with their use case. Then I address the data requirements, usability and everything else about how the inner workings of my code operate.

In my Proposal, I discuss why I have come about this project along with research and how I would like to achieve my program with abstract headings. I do not talk about code here but how I would tackle on creating the project.

2.0 System

2.1. Requirements

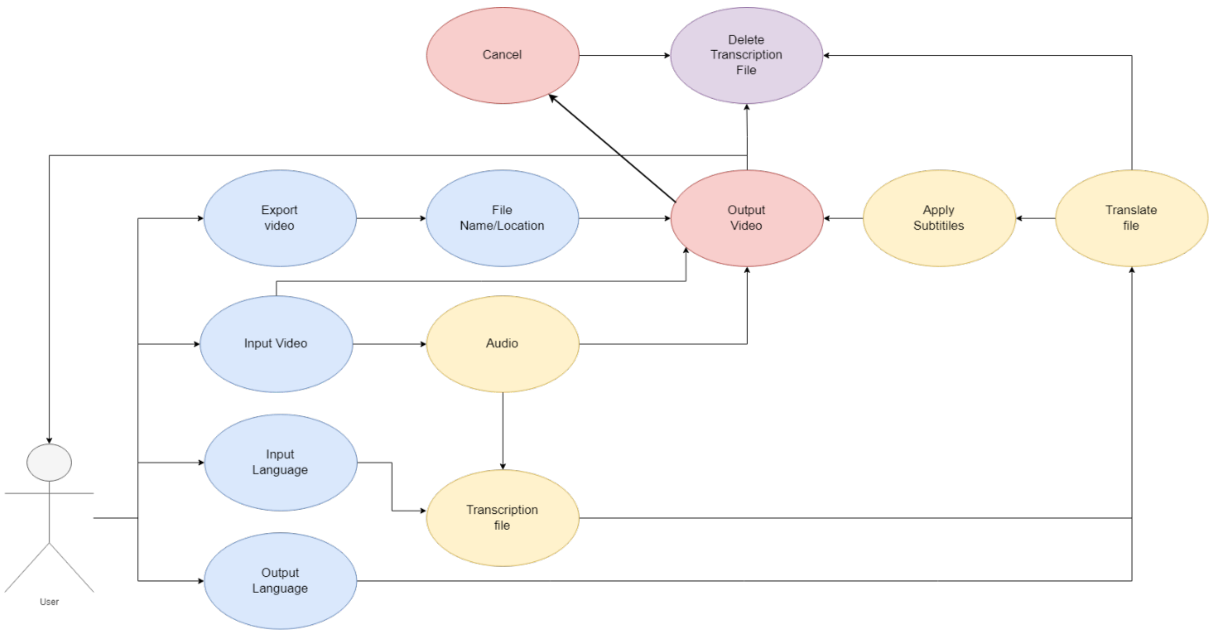
As of right now, my application is in the prototype stage, where there are no functional buttons. The user should be able to pick a video where it will open a windows explorer and choose a file from their storage. The user should then choose the language it originally is then chose an output language the user wants. Example, English to German. The user will then export the video to a mp4 file.

2.1.1. Functional Requirements

1. Software must run on x64 arch of Windows
2. Button that enables the user to select video from their storage
3. User must be able to choose original language to desired language with buttons
4. Button for export
5. Googles API should start
6. Taking audio and transcribing it to file
7. Taking transcription file and translate
8. SRT file to be merged with video
9. Start encoding and decoding video
10. Uses FFMPEG library to export
11. Delete transcribing and SRT file to reduce space usage.

12. Output video is completed

2.1.1.1. Use Case Diagram



Requirement 1: Input Video

Description & Priority

Choosing a video is important for the software is important otherwise the software will not have an input to use. By selecting a video, the software knows what audio to take and transcribe. This requirement is priority level 1.

Use Case

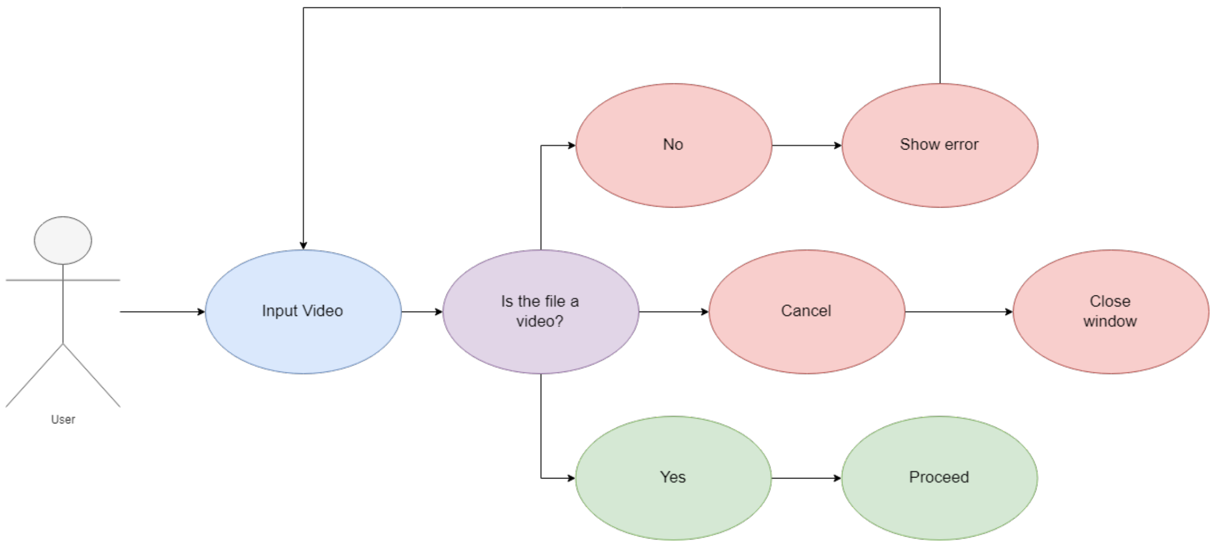
Scope

The scope of this use case is controlling files that are of video type extension. Anything that is not a video will not be accepted.

Description

The use case describes that users are only allowed to input a video and no other file. If a user were to input another file, an error would occur.

Use Case Diagram



Flow Description

Precondition

The software must be running and the User must have a video to be processed

Activation

This use case starts when a User attempts to input a video

Main flow

1. The system identifies the file
2. If User inputs a file that is not a video (See A1)
3. If the User cancels the selection (See E1)
4. The User can change the language input and output
5. Export is conditional if user came from A1 or E1

Alternate flow

A1: File checker
1. The system gives a pop-up message
2. Pop up message says file is not an accepted format
3. Open windows explorer again
4. If file type is not accepted format, repeat A1 position 1
5. If file type is acceptable continue to position 4 of the main flow

Exceptional flow

E1: Cancel
1. The User presses the cancel button
2. The windows explorer closes
3. The use case continues at position 1 of the main flow

Termination

The User cancelled the selection or the software is closed

Post condition

The software is waiting for a user action.

Requirement 2: Input Language

Description & Priority

Choosing an input language is important for the software otherwise the software will input the wrong translation. Words can sound similar in different languages; therefore, it is priority level 2.

Use Case

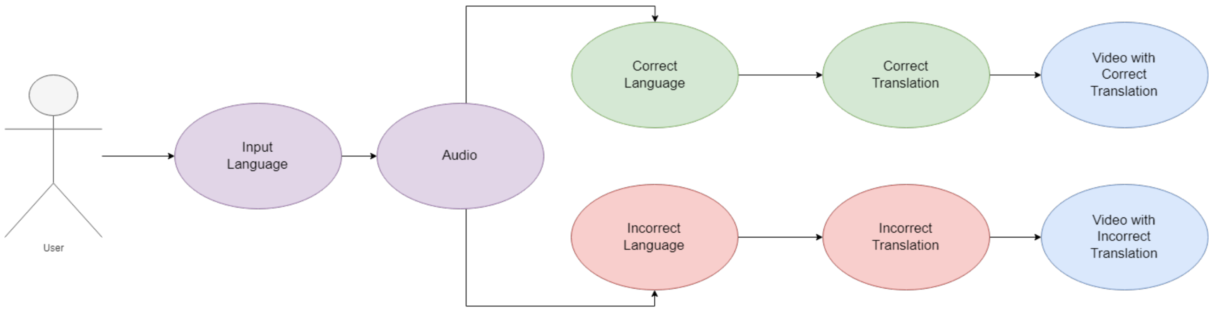
Scope

The scope of this use case is limited to the language that will be used in the video.

Description

Having the wrong language inputted will result in a wrong translation. It is very important.

Use Case Diagram



Flow Description

Precondition

The software must be running

Activation

This use case starts when a User chooses the input language of the video

Main flow

1. The system sets the language in the video
2. If User inputs a language that is spoken in the video (See A1)
3. If User inputs a language that is not spoken in the video (See A1)
4. The software waits for output language
5. The User can now export the video

Alternate flow

A1: Language detector

1. The system sets the input language
2. The use case continues at position 4 of the main flow

Termination

The User cancelled the selection or the software is closed

Post condition

The software is waiting for a user action.

Requirement 3: Output Language

Description & Priority

Choosing an output language is important for the software is important otherwise the software will output the wrong translation. Therefore, this is a priority level 3

Use Case

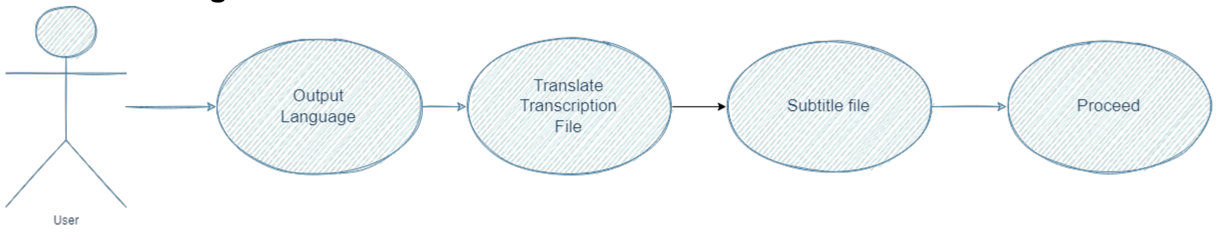
Scope

The scope of this use case is limited to the output of the subtitles of the video

Description

This will output the desired language by the user

Use Case Diagram



Flow Description

Precondition

The file along with the input and output languages must be selected before the video is able to export.

Activation

This use case starts when a User chooses the output language of the subtitle

Main flow

1. The system sets the language in the video
2. If User inputs a language that is spoken in the video (See A1)
3. If User inputs a language that is not spoken in the video (See A1)
4. The software waits for output language
5. The User can now export the video

Alternate flow

A1: Language detector

1. The system sets the input language
2. The use case continues at position 4 of the main flow

Termination

The User cancelled the selection or the software is closed

Post condition

The software is waiting for a user action.

Requirement 4: Export Video

Description & Priority

Exporting is the least important but is still important since it will take time processing the video because it uses computer resources. The reason why it’s last because the other components are needed before you are able to export the video. Therefore, this is priority level 4.

Use Case

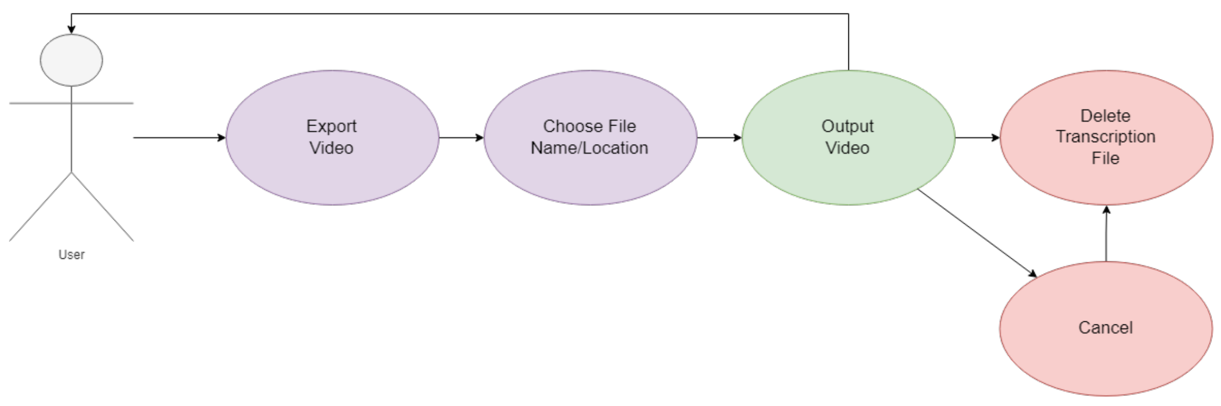
Scope

The scope of exporting the video file will set the name and the location of the file.

Description

The User will have a windows explorer open and will be asked to name and choose the location of the file.

Use Case Diagram



Flow Description

Precondition

The software must be running.

Activation

This use case starts when a User attempts to export the video.

Main flow

1. The software waits for the user action
2. The User selects the export button (See A1)
3. The audio is being transcribed
4. The transcribed file is being translated
5. The software waits for the user action (See E1)
6. The file is named and saved in the location set by the user
7. The subtitles file is deleted

Alternate flow

A1: Language injector
The software is processing the video
The use case continues at position 4 of the main flow

E1: Cancel
If the User presses the cancel button
The software stops the operation
The subtitles file is deleted
The use case continues at position 1 of the main flow

Termination

The User cancelled the process and will stop the operation.

Post condition

The video has been created. The name of the file and their location has been set. The software is waiting for a user action.

2.1.2. Data Requirements

Video

The user is required to input a video. The software's main purpose is to transcribe audio from the video and translate it. After translating the transcription, the file will be made into a SRT file, a file that holds subtitles. The subtitles will then be applied to the video.

Input and Output Languages

The input language is very important because having the wrong language can lead to multiple errors in translation. Words can sound very similar to other languages. It is important before the user exports the video; they have confirmed the language contained in the video.

The output language is what the subtitles will be on-top of the video. It is also important the user is sure what language they want the output to be as. Depending on the length of the video, it can take a long time processing the video.

2.1.3. User Requirements

The user requirements describe the necessary functionality and features that the system must provide in order to meet the needs of the user.

- User must be able to select a video
- User must be able to select an input language
- User must be able to select an output language
- User must be able to save the file name and location
- User must be able to view the saved file

2.1.4. Environmental Requirements

A pc with strong capabilities as the process is very intensive. A strong internet connection would be needed as the software utilizes the Google's API services.

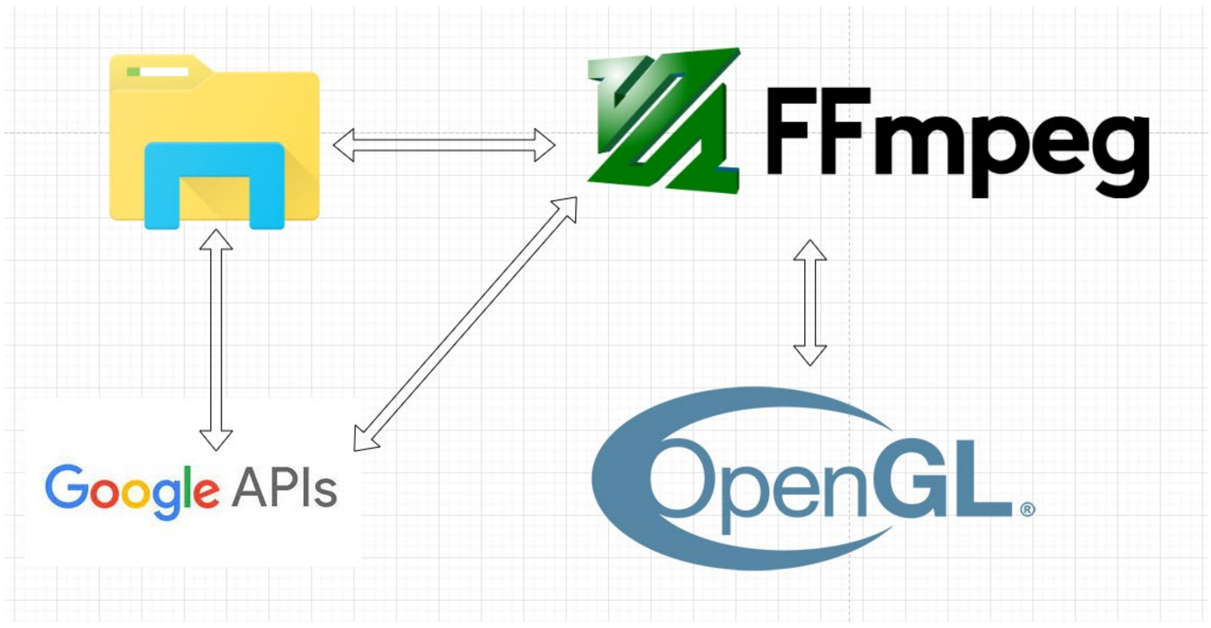
2.1.5. Usability Requirements

The software will work on most recent computers, although it is recommended that the user has a medium to high specs for faster processing of the video. The software takes advantage of the CPU as of right now. It would be most optimal to render with a dedicated graphics card but to keep it simple for now, it uses the CPU as everyone has a CPU in a computer.

A stable and strong internet is also preferred since the software takes advantage of Google's API services. The software will be sending and receiving packets to and from Google.

2.2. Design & Architecture

Architecture Diagram



There are too many algorithms for me to list when using FFMPEG. The few that I will be using when rendering a video using H.264 will be Video buffering verifier (VBV), Intra prediction and Inter prediction. For encoding and decoding audio. I have chosen mp3 for simplicity. AAC has more advanced features that I will not use for this project. The video will then as a mp4 file.

My project will work as follows:

The first step in this software's user interface is letting the user select a video file to add subtitles to. The user also has the option of choosing a language for both the input and output subtitles. The user starts the export procedure by pushing a button after choosing the file and languages.

The intricate operation of manipulating the video and subtitles is handled by the program invisibly using the powerful FFmpeg library. When a user chooses a video file FFmpeg automatically demultiplexes the tracks and divides them into separate streams. These streams include a variety of elements, including video frames, audio samples, subtitles, and more related data.

The program uses an encoder module once the video frames have been processed and the subtitles have been incorporated flawlessly. The processed data is intelligently compressed by this encoder, thereby lowering its size while making an effort to preserve the best possible video and audio quality. When all the streams have been carefully combined into one final file, a muxer assumes center stage.

Algorithms:

Video buffering verifier (VBV) is a mechanism in video encoding that controls the flow of data from the encoder to the decoder. It helps to ensure that the decoder has enough data to decode the video stream smoothly, without running out of data or stalling.

Intra prediction is a technique used in video encoding to predict the values of blocks of pixels within a single frame based on the pixels in the surrounding area. Intra prediction is used to reduce the amount of data that needs to be transmitted by encoding only the difference between the predicted values and the actual values.

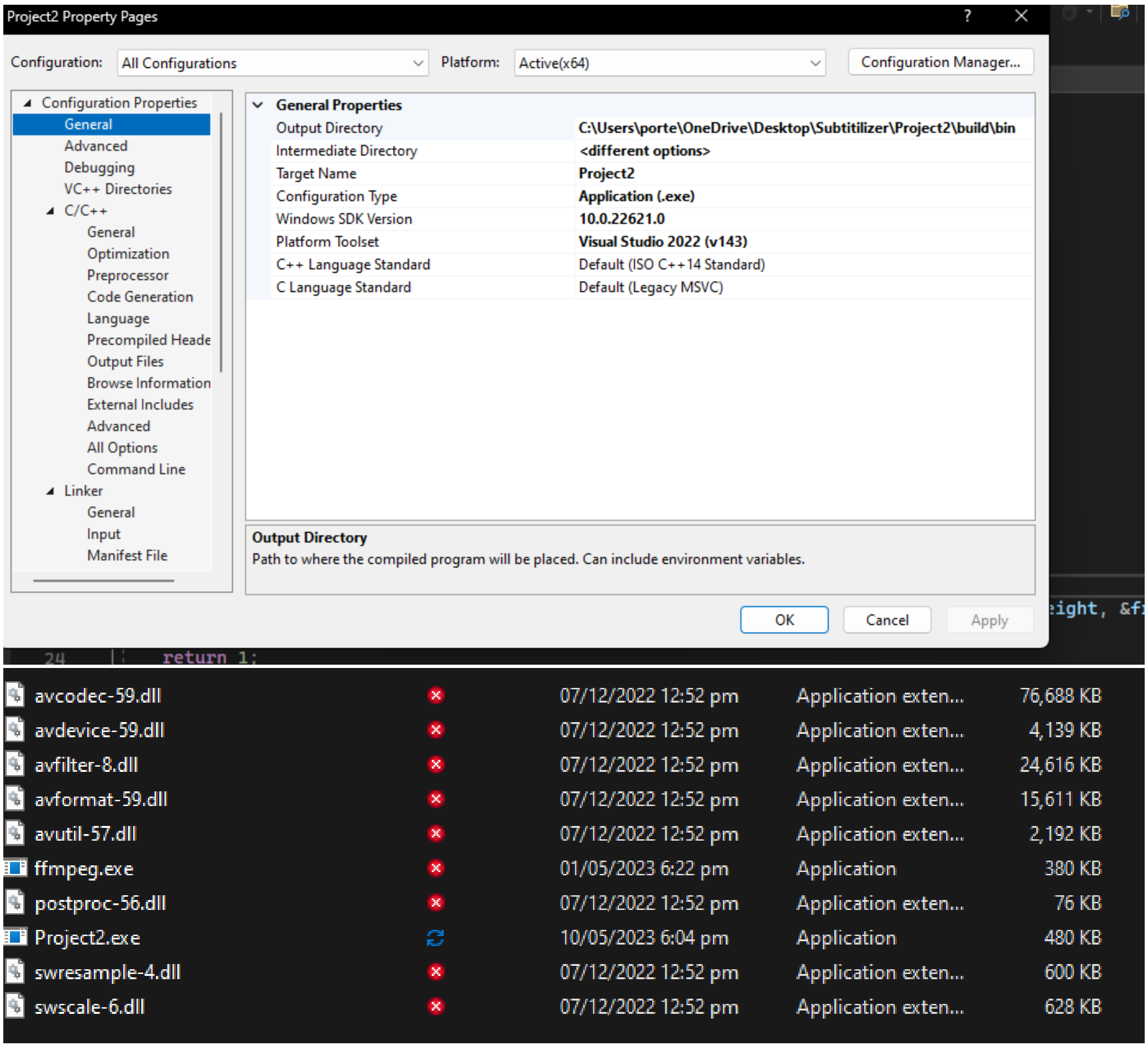
Inter prediction is a technique used in video encoding to predict the values of blocks of pixels between two consecutive frames based on the pixels in the previous frame. Inter prediction is used to reduce the amount of data that needs to be transmitted by encoding only the difference between the predicted values and the actual values.

Both intra prediction and inter prediction are used to improve the efficiency of video encoding by reducing the amount of data that needs to be transmitted. They are key components of many video codecs including those used by FFmpeg.

The user's chosen location on their computer will then receive the finished file with the subtitles. A broad variety of media players may be used to playback this file, and viewers will experience a better watching experience thanks to the subtitles' elegant overlaying of the video content. This practical approach offers a quick and easy way to translate videos and making them accessible to a broader audience and potentially serving educational purposes.

Although the program uses Google's API for speech recognition and translation, there are certain restrictions to take into account. The precision and caliber of the voice recognition and translation process may change depending on the accessibility and dependability of Google's API service. Additionally, there may be related fees and terms of service considerations when using Google's API.

Linking up FFmpeg:
For me to be able to use FFmpeg, I would download the libraries and .dll files required for the functionalities. For FFmpeg to function correctly I would then set up Visual Studio’s properties for the current project to take the libraries and link up the necessary files.



Configuration: All Configurations   Platform: Active(x64)   Configuration Manager...

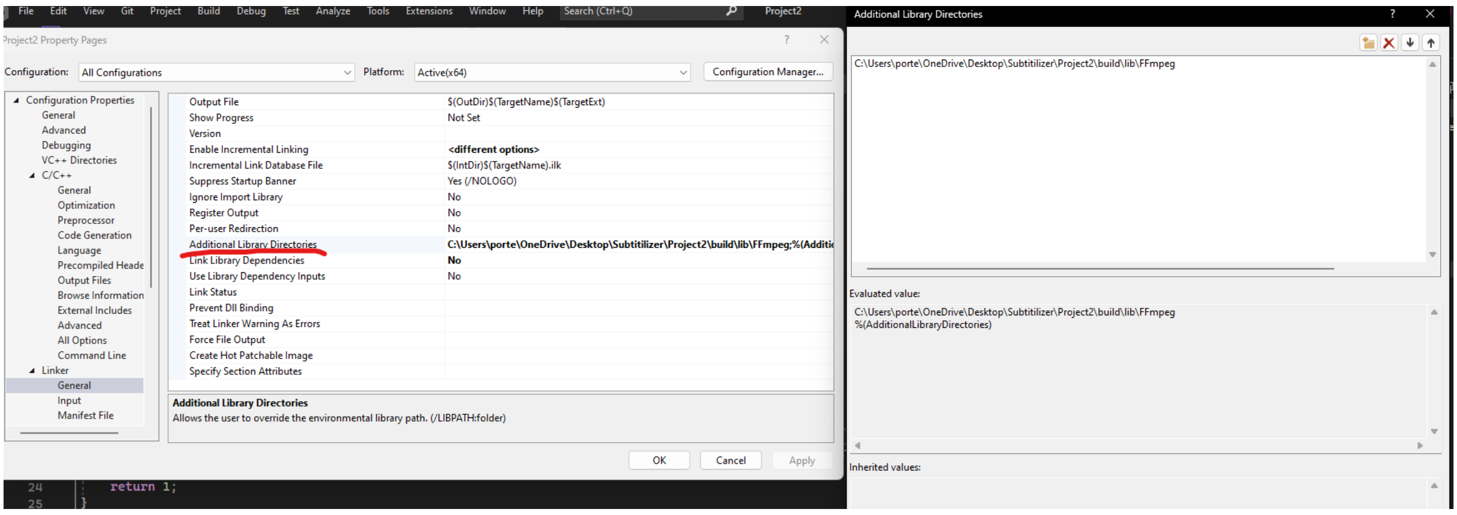
General Properties

Output Directory	**C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\build\bin**
Intermediate Directory	**<different options>**
Target Name	**Project2**
Configuration Type	**Application (.exe)**
Windows SDK Version	**10.0.22621.0**
Platform Toolset	**Visual Studio 2022 (v143)**
C++ Language Standard	Default (ISO C++14 Standard)
C Language Standard	Default (Legacy MSVC)

Output Directory
Path to where the compiled program will be placed. Can include environment variables.

```
avcodec-59.dll       07/12/2022 12:52 pm   Application exten...   76,688 KB
avdevice-59.dll      07/12/2022 12:52 pm   Application exten...    4,139 KB
avfilter-8.dll       07/12/2022 12:52 pm   Application exten...   24,616 KB
avformat-59.dll      07/12/2022 12:52 pm   Application exten...   15,611 KB
avutil-57.dll        07/12/2022 12:52 pm   Application exten...    2,192 KB
ffmpeg.exe           01/05/2023 6:22 pm    Application               380 KB
postproc-56.dll      07/12/2022 12:52 pm   Application exten...       76 KB
Project2.exe         10/05/2023 6:04 pm    Application               480 KB
swresample-4.dll     07/12/2022 12:52 pm   Application exten...      600 KB
swscale-6.dll        07/12/2022 12:52 pm   Application exten...      628 KB
```

The picture above shows the output directory while also having the .dll files to function correctly



Configuration: All Configurations   Platform: Active(x64)   Configuration Manager...

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**<different options>**
Incremental Link Database File	$(IntDir)$(TargetName).ilk
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\build\lib\FFmpeg;%(Additic**
Link Library Dependencies	**No**
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Additional Library Directories
Allows the user to override the environmental library path. (/LIBPATH:folder)

```
return 1;
}
```

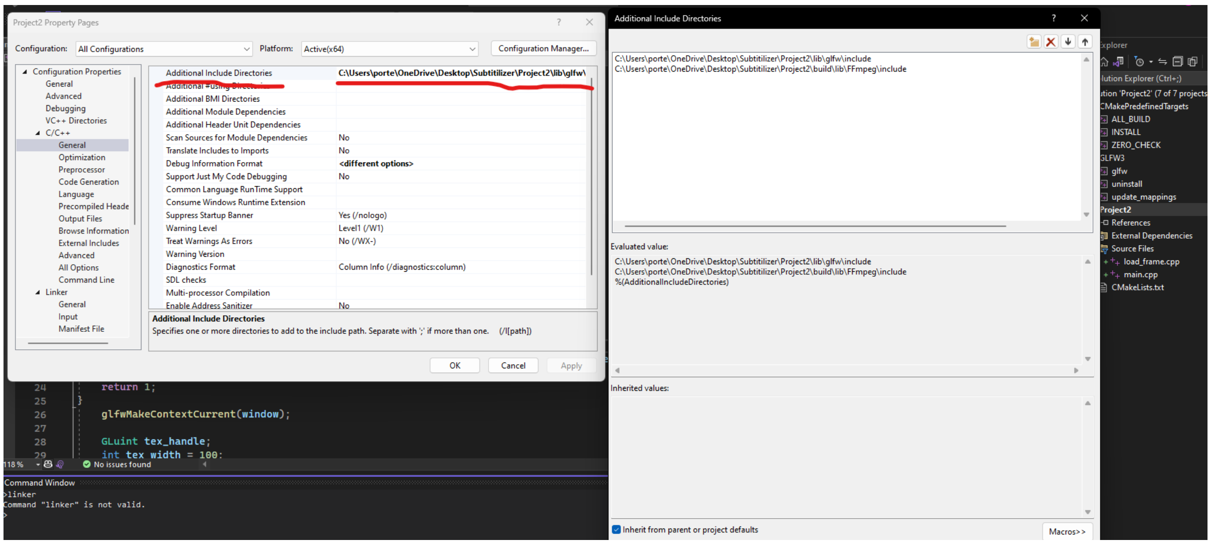
Additional Library Directories

C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\build\lib\FFmpeg

Evaluated value:
C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\build\lib\FFmpeg
%(AdditionalLibraryDirectories)

Inherited values:

The picture above then links the libraries together.



Configuration: All Configurations   Platform: Active(x64)

Additional Include Directories	C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\lib\glfw\
Additional #using Directories	
Additional BMI Directories	
Additional Module Dependencies	
Additional Header Unit Dependencies	
Scan Sources for Module Dependencies	No
Translate Includes to Imports	No
Debug Information Format	**<different options>**
Support Just My Code Debugging	No
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level1 (/W1)
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Column Info (/diagnostics:column)
SDL checks	
Multi-processor Compilation	
Enable Address Sanitizer	No

Additional Include Directories
Specifies one or more directories to add to the include path. Separate with ';' if more than one. (/I[path])

Additional Include Directories

```
C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\lib\glfw\include
C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\build\lib\FFmpeg\include
```

Evaluated value:

```
C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\lib\glfw\include
C:\Users\porte\OneDrive\Desktop\Subtitilizer\Project2\build\lib\FFmpeg\include
%(AdditionalIncludeDirectories)
```

Inherited values:

☑ Inherit from parent or project defaults

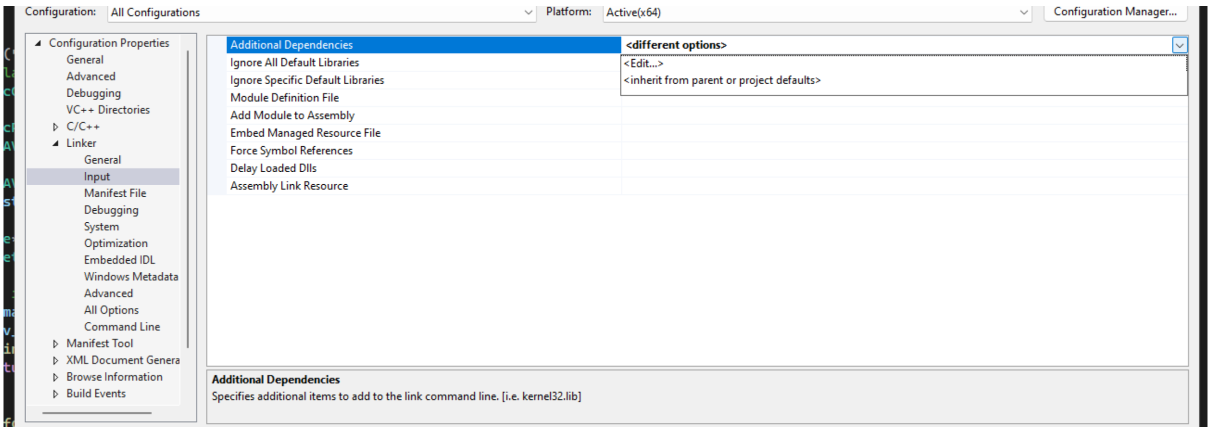
```
        return 1;
    }

    glfwMakeContextCurrent(window);

    GLuint tex_handle;
    int tex width = 100;
```

118 %   No issues found

Command Window

```
>linker
Command "linker" is not valid.
>
```

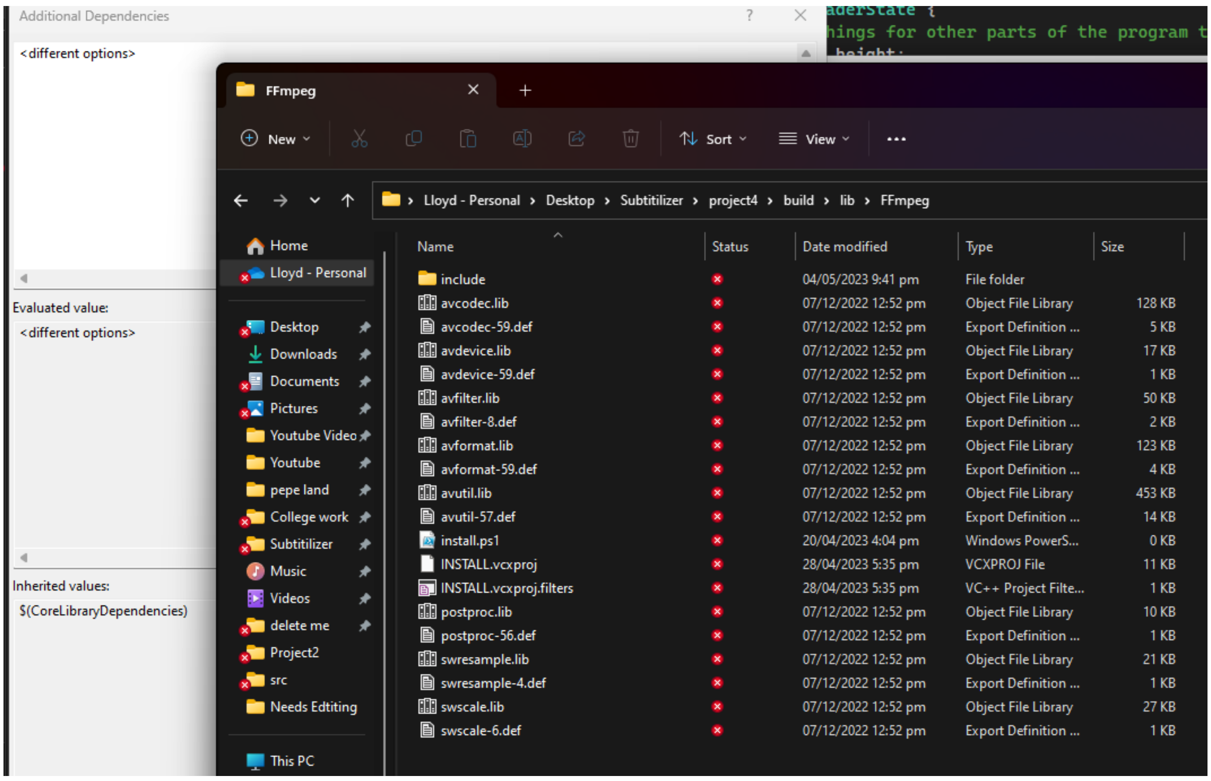
The above picture then contains the include folder for FFmpeg. This has the header files to be able to use the libraries and communicates to the program where to get certain functionalities.



Configuration: All Configurations   Platform: Active(x64)

Additional Dependencies	**<different options>**
Ignore All Default Libraries	<Edit...>
Ignore Specific Default Libraries	<inherit from parent or project defaults>
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]


Additional Dependencies

<different options>

Evaluated value:

<different options>

Inherited values:

$(CoreLibraryDependencies)

FFmpeg

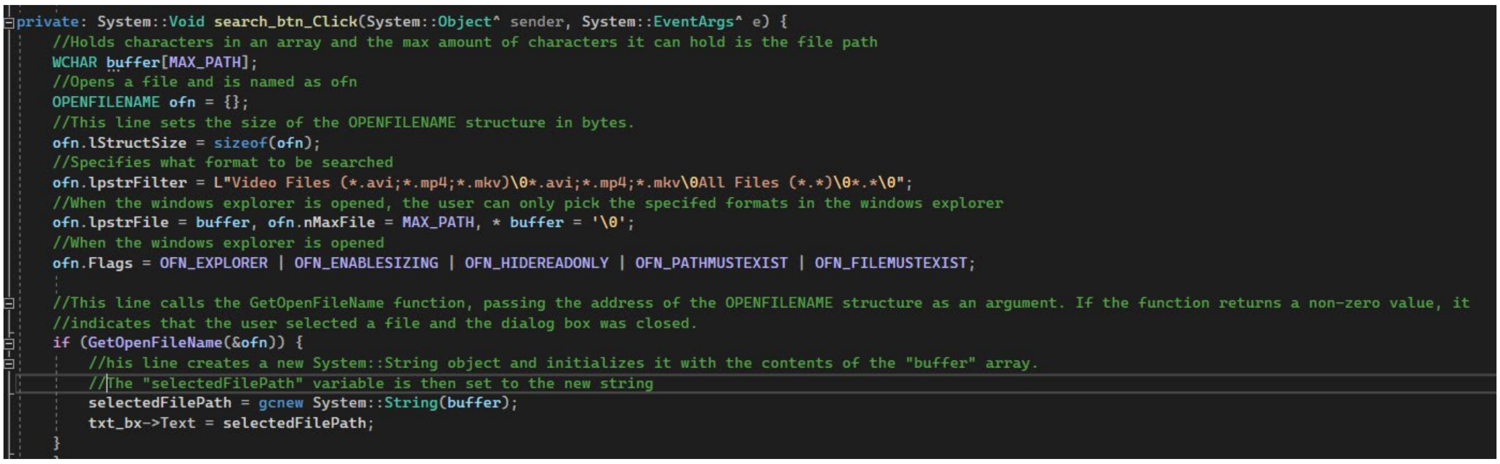
Lloyd - Personal › Desktop › Subtitilizer › project4 › build › lib › FFmpeg

Name	Status	Date modified	Type	Size
include	✖	04/05/2023 9:41 pm	File folder	
avcodec.lib	✖	07/12/2022 12:52 pm	Object File Library	128 KB
avcodec-59.def	✖	07/12/2022 12:52 pm	Export Definition ...	5 KB
avdevice.lib	✖	07/12/2022 12:52 pm	Object File Library	17 KB
avdevice-59.def	✖	07/12/2022 12:52 pm	Export Definition ...	1 KB
avfilter.lib	✖	07/12/2022 12:52 pm	Object File Library	50 KB
avfilter-8.def	✖	07/12/2022 12:52 pm	Export Definition ...	2 KB
avformat.lib	✖	07/12/2022 12:52 pm	Object File Library	123 KB
avformat-59.def	✖	07/12/2022 12:52 pm	Export Definition ...	4 KB
avutil.lib	✖	07/12/2022 12:52 pm	Object File Library	453 KB
avutil-57.def	✖	07/12/2022 12:52 pm	Export Definition ...	14 KB
install.ps1	✖	20/04/2023 4:04 pm	Windows PowerS...	0 KB
INSTALL.vcxproj	✖	28/04/2023 5:35 pm	VCXPROJ File	11 KB
INSTALL.vcxproj.filters	✖	28/04/2023 5:35 pm	VC++ Project Filte...	1 KB
postproc.lib	✖	07/12/2022 12:52 pm	Object File Library	10 KB
postproc-56.def	✖	07/12/2022 12:52 pm	Export Definition ...	1 KB
swresample.lib	✖	07/12/2022 12:52 pm	Object File Library	21 KB
swresample-4.def	✖	07/12/2022 12:52 pm	Export Definition ...	1 KB
swscale.lib	✖	07/12/2022 12:52 pm	Object File Library	27 KB
swscale-6.def	✖	07/12/2022 12:52 pm	Export Definition ...	1 KB

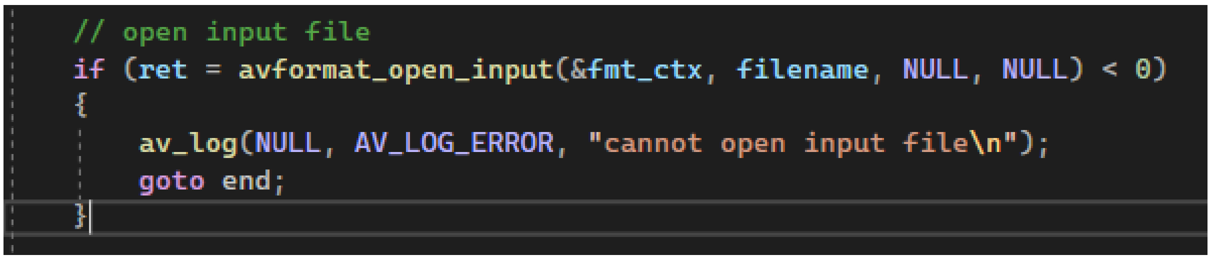
In the 2 above pictures, I would then navigate to the input section from the Linker. I would copy and paste the library file names to be able to use the libraries.

2.3. Implementation

2.3.1 Code with GUI


```
private: System::Void search_btn_Click(System::Object^ sender, System::EventArgs^ e) {
    //Holds characters in an array and the max amount of characters it can hold is the file path
    WCHAR buffer[MAX_PATH];
    //Opens a file and is named as ofn
    OPENFILENAME ofn = {};
    //This line sets the size of the OPENFILENAME structure in bytes.
    ofn.lStructSize = sizeof(ofn);
    //Specifies what format to be searched
    ofn.lpstrFilter = L"Video Files (*.avi;*.mp4;*.mkv)\0*.avi;*.mp4;*.mkv\0All Files (*.*)\0*.*\0";
    //When the windows explorer is opened, the user can only pick the specifed formats in the windows explorer
    ofn.lpstrFile = buffer, ofn.nMaxFile = MAX_PATH, * buffer = '\0';
    //When the windows explorer is opened
    ofn.Flags = OFN_EXPLORER | OFN_ENABLESIZING | OFN_HIDEREADONLY | OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

    //This line calls the GetOpenFileName function, passing the address of the OPENFILENAME structure as an argument. If the function returns a non-zero value, it
    //indicates that the user selected a file and the dialog box was closed.
    if (GetOpenFileName(&ofn)) {
        //his line creates a new System::String object and initializes it with the contents of the "buffer" array.
        //The "selectedFilePath" variable is then set to the new string
        selectedFilePath = gcnew System::String(buffer);
        txt_bx->Text = selectedFilePath;
    }
```

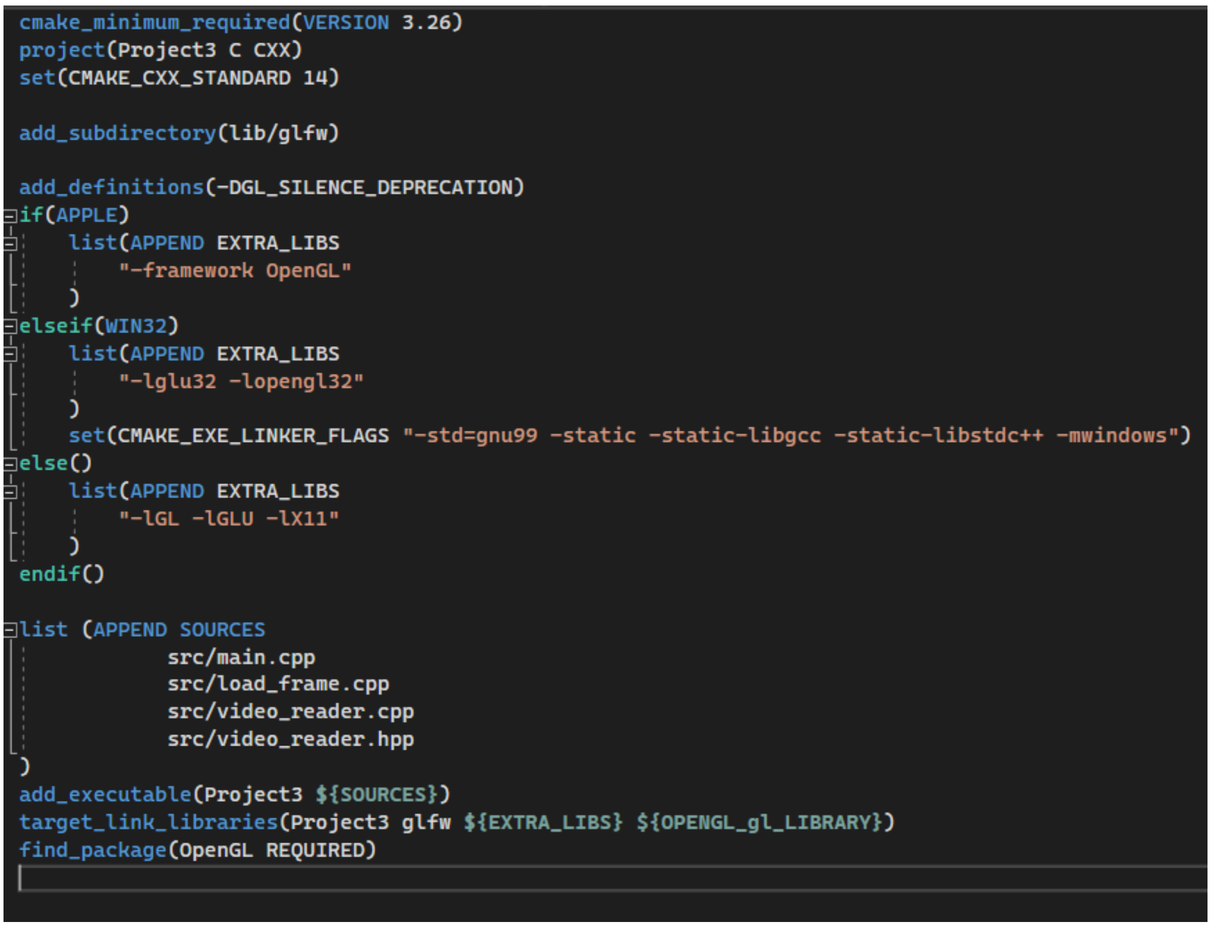
This is an event where it waits for the User action pressing a button. The function displays a dialog box using the GetOpenFileName function, which allows the user to select a file to open. The function specifies that the dialog box should allow the user to select video files with the extensions .avi, .mp4, or .mkv, or any file with the extension .*.The function stores the selected file path in the buffer array and creates a new System::String object with the contents of the array. The selectedFilePath variable is then set to the new string. Finally, the function sets the text of the txt_bx text box to the selected file path.



```c
// open input file
if (ret = avformat_open_input(&fmt_ctx, filename, NULL, NULL) < 0)
{
    av_log(NULL, AV_LOG_ERROR, "cannot open input file\n");
    goto end;
}
```

This code is checking if an error occurred while opening the input file by calling the avformat_open_input function and assigning the return value to the ret variable. If the value of ret is less than zero, it means that an error occurred, and the av_log function is called to print an error message to the standard error output. The program then jumps to the end label (which is defined later in the code).

2.3.2 Code without GUI



```cmake
cmake_minimum_required(VERSION 3.26)
project(Project3 C CXX)
set(CMAKE_CXX_STANDARD 14)

add_subdirectory(lib/glfw)

add_definitions(-DGL_SILENCE_DEPRECATION)
if(APPLE)
    list(APPEND EXTRA_LIBS
        "-framework OpenGL"
    )
elseif(WIN32)
    list(APPEND EXTRA_LIBS
        "-lglu32 -lopengl32"
    )
    set(CMAKE_EXE_LINKER_FLAGS "-std=gnu99 -static -static-libgcc -static-libstdc++ -mwindows")
else()
    list(APPEND EXTRA_LIBS
        "-lGL -lGLU -lX11"
    )
endif()

list (APPEND SOURCES
            src/main.cpp
            src/load_frame.cpp
            src/video_reader.cpp
            src/video_reader.hpp
)
add_executable(Project3 ${SOURCES})
target_link_libraries(Project3 glfw ${EXTRA_LIBS} ${OPENGL_gl_LIBRARY})
find_package(OpenGL REQUIRED)
```

The first file I made was a cmake file that auto generates a project for me that combines. I will explain from

top to bottom what the code does. The cmake file at the very minium will be version 3.26.
project(Project3 C CXX): This line specifies that the project is composed of C and C++ code and sets the project name to Project3.

Sets the C++ standard to version 14 with the command set(CMAKE_CXX_STANDARD 14).

Adds the subdirectory lib/glfw to the build process with the command add_subdirectory(lib/glfw). It signifies that the directory's CMakeLists.txt file will be treated separately.

add_definitions(-DGL_SILENCE_DEPRECATION): In order to define the macro GL_SILENCE_DEPRECATION, this line adds a compiler flag. I disable OpenGL deprecation warnings by using this macro.

if(APPLE) elseif(WIN32) else() endif: The operating system is examined in this block of code, and the EXTRA_LIBS variable is set appropriately. It adds the -framework OpenGL switch to EXTRA_LIBS on Apple platforms. It adds the -lglu32 and -lopengl32 flags to the linker on Windows systems and sets other linker flags relevant to static linking and Windows-specific options. It adds the -lGL, -lGLU, and -lX11 settings for various operating systems.

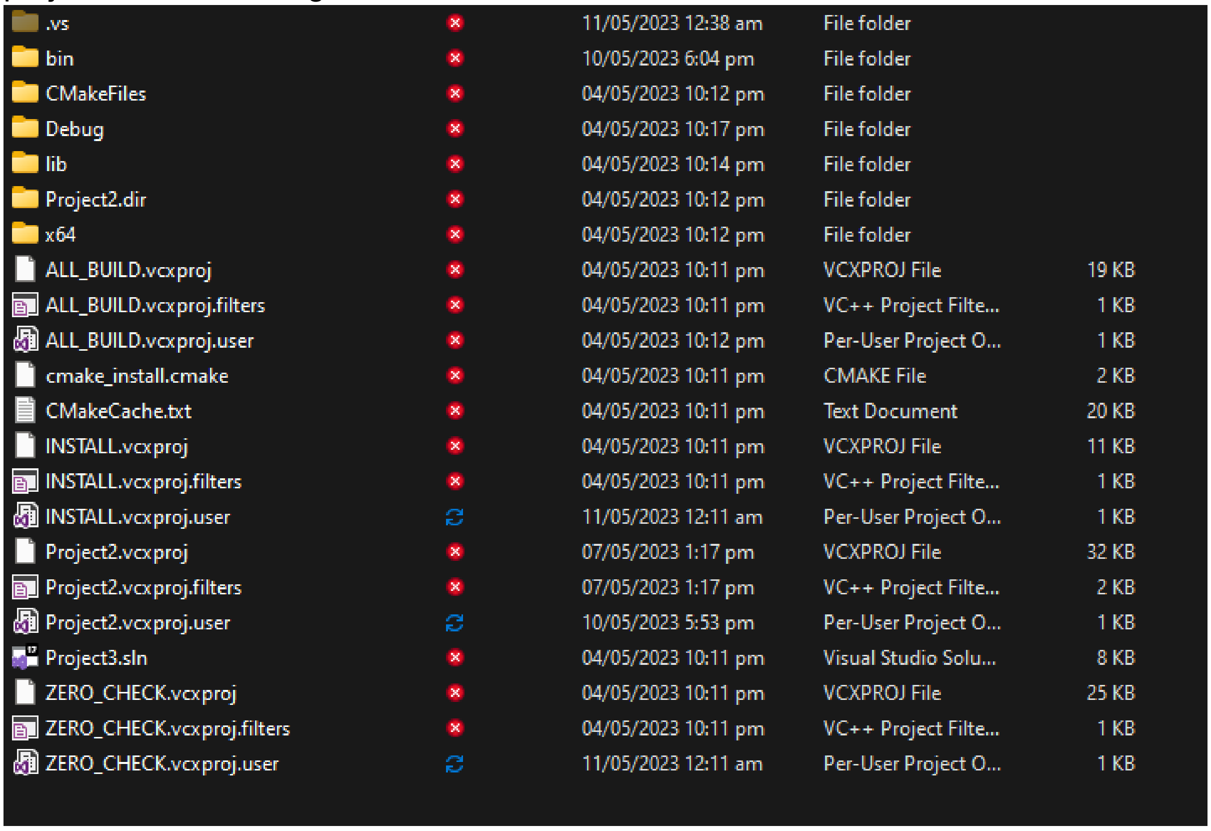
(APPEND SOURCES) list A list of source files is added to the SOURCES variable by this line. src/main.cpp, src/load_frame.cpp, src/video_reader.cpp, and src/video_reader.hpp are among the source files.

Using the source files specified in the SOURCES variable add_executable(Project3 $SOURCES) generates an executable with the name Project3.

Project 3: target_link_libraries glfw ($EXTRA_LIBS $OPENGL_gl_LIBRARY) The libraries to link with the Project3 executable are specified on this line. It contains $OPENGL_gl_LIBRARY, EXTRA_LIBS, and the glfw library.

find_package (REQUIRES OpenGL) This line looks for the OpenGL package and makes it accessible for the project's linking.

After creating the cmakelists.txt I would run this script through gitbash to create the project. Which would generate most of these files as shown below.

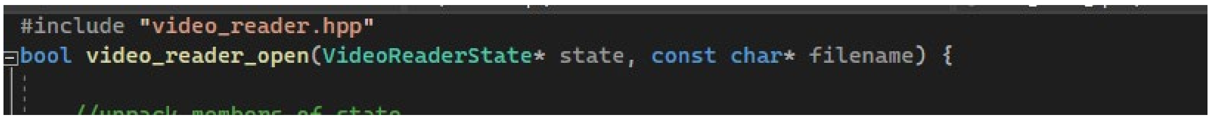


.vs	✖	11/05/2023 12:38 am	File folder	
bin	✖	10/05/2023 6:04 pm	File folder	
CMakeFiles	✖	04/05/2023 10:12 pm	File folder	
Debug	✖	04/05/2023 10:17 pm	File folder	
lib	✖	04/05/2023 10:14 pm	File folder	
Project2.dir	✖	04/05/2023 10:12 pm	File folder	
x64	✖	04/05/2023 10:12 pm	File folder	
ALL_BUILD.vcxproj	✖	04/05/2023 10:11 pm	VCXPROJ File	19 KB
ALL_BUILD.vcxproj.filters	✖	04/05/2023 10:11 pm	VC++ Project Filte...	1 KB
ALL_BUILD.vcxproj.user	✖	04/05/2023 10:12 pm	Per-User Project O...	1 KB
cmake_install.cmake	✖	04/05/2023 10:11 pm	CMAKE File	2 KB
CMakeCache.txt	✖	04/05/2023 10:11 pm	Text Document	20 KB
INSTALL.vcxproj	✖	04/05/2023 10:11 pm	VCXPROJ File	11 KB
INSTALL.vcxproj.filters	✖	04/05/2023 10:11 pm	VC++ Project Filte...	1 KB
INSTALL.vcxproj.user	🔄	11/05/2023 12:11 am	Per-User Project O...	1 KB
Project2.vcxproj	✖	07/05/2023 1:17 pm	VCXPROJ File	32 KB
Project2.vcxproj.filters	✖	07/05/2023 1:17 pm	VC++ Project Filte...	2 KB
Project2.vcxproj.user	🔄	10/05/2023 5:53 pm	Per-User Project O...	1 KB
Project3.sln	✖	04/05/2023 10:11 pm	Visual Studio Solu...	8 KB
ZERO_CHECK.vcxproj	✖	04/05/2023 10:11 pm	VCXPROJ File	25 KB
ZERO_CHECK.vcxproj.filters	✖	04/05/2023 10:11 pm	VC++ Project Filte...	1 KB
ZERO_CHECK.vcxproj.user	🔄	11/05/2023 12:11 am	Per-User Project O...	1 KB

Then to get the GLFW files, I ran this code to get the GLFW to install into my project: git submodule add https://github.com/glfw/glfw glfw. This would automatically create a GLFW folder directly from the creators of GLFW on Github.

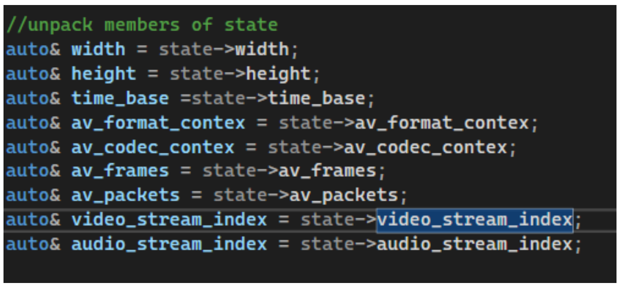
2.3.2.1 video_reader.cpp

A C++ method named video_reader_open is present in this code and is in charge of opening a video file so that it could be decoded. The function accepts two inputs: a pointer to the structure VideoReaderState, which stores various video-related data, and a reference to a string that denotes the filename of the video file that will be opened.



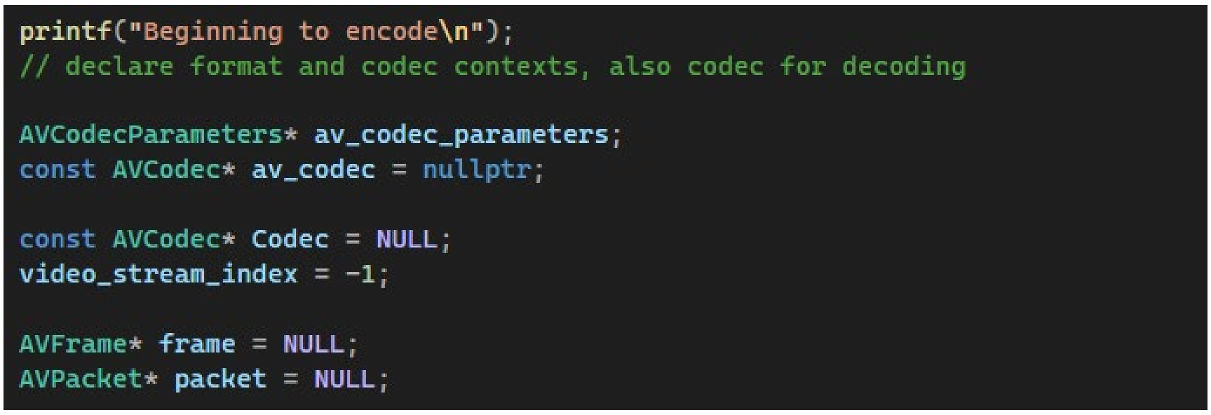
```
#include "video_reader.hpp"
bool video_reader_open(VideoReaderState* state, const char* filename) {

    //unpack members of state
```

Using the auto& keyword, which supports automated type deduction and reference variables, the method first extracts the components of the VideoReaderState structure.



```
//unpack members of state
auto& width = state->width;
auto& height = state->height;
auto& time_base =state->time_base;
auto& av_format_contex = state->av_format_contex;
auto& av_codec_contex = state->av_codec_contex;
auto& av_frames = state->av_frames;
auto& av_packets = state->av_packets;
auto& video_stream_index = state->video_stream_index;
auto& audio_stream_index = state->audio_stream_index;
```

The AVCodecContext, AVCodecParameters and AVCodec structures' variables are then declared in the code. The FFmpeg library, which is commonly used for video decoding, depends on these structures in order to function.



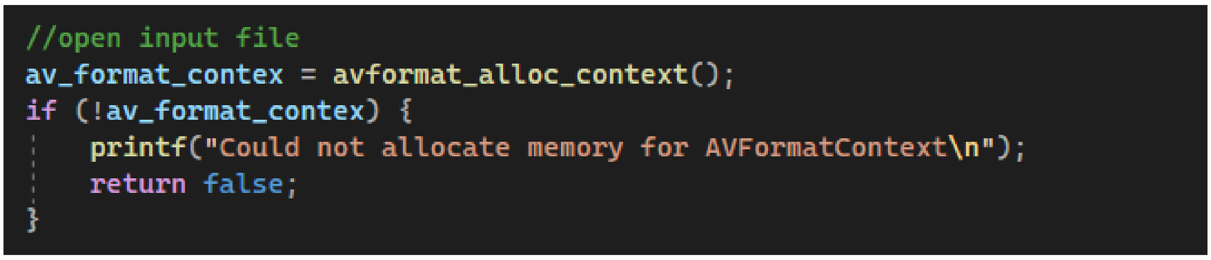
```
printf("Beginning to encode\n");
// declare format and codec contexts, also codec for decoding

AVCodecParameters* av_codec_parameters;
const AVCodec* av_codec = nullptr;

const AVCodec* Codec = NULL;
video_stream_index = -1;

AVFrame* frame = NULL;
AVPacket* packet = NULL;
```

This Code block is responsible for opening an input file and initializing an AVFormatContext structure for further processing.

The first line, av_format_contex = avformat_alloc_context();, calls the avformat_alloc_context function to allocate memory for the AVFormatContext structure. The AVFormatContext object, which is used to hold the format-specific information about the input file, is initialized by this function and memory is dynamically allocated.

The line printf(Could not allocate memory for AVFormatContextn); in the if statement produces a message stating that the memory allocation for the AVFormatContext was unsuccessful.

Finally, the line return false; exits the function and returns a boolean value of false indicating that the opening of the input file was not successful.



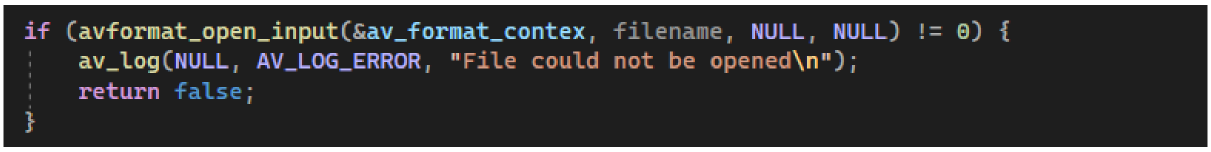
```
//open input file
av_format_contex = avformat_alloc_context();
if (!av_format_contex) {
    printf("Could not allocate memory for AVFormatContext\n");
    return false;
}
```

The provided block of code is in charge of using the AVFormatContext structure to open an input file.

The if statement verifies the avformat_open_input function's return value. By initializing the AVFormatContext structure with the supplied filename, this method makes an attempt to open the specified file. If avformat_open_input returns a value other than 0, the file opening failed.

Inside the if statement, the line av_log(NULL, AV_LOG_ERROR, File could not be opened\n); reports an error message using the av_log function from the FFmpeg library. This error message indicates that the file could not be opened.

The line return false; exits the function and returns a boolean value of false, indicating that the opening of the file was not successful.



```
if (avformat_open_input(&av_format_contex, filename, NULL, NULL) != 0) {
    av_log(NULL, AV_LOG_ERROR, "File could not be opened\n");
    return false;
}
```

By iterating through the streams in the AVFormatContext and extracting details about each stream, such as codec parameters and type this provided block of code is in charge of handling the task.

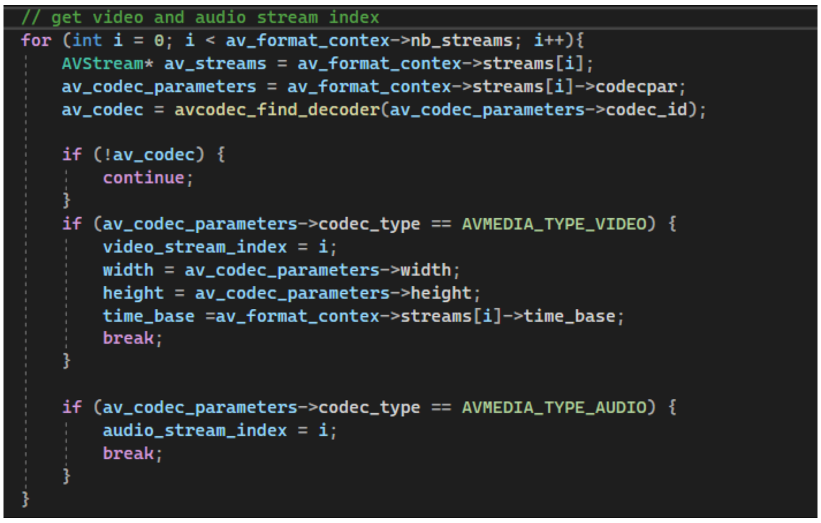
The for loop sets the value of the integer variable i to 0 and iterates as long as i is lower than the value of the AVFormatContext's nb_streams property. The function uses av_format_contex->streams[i] in each iteration to obtain the AVStream pointer of the active stream.

By gaining access to the codecpar property of the AVStream, the AVCodecParameters structure is added to the av_codec_parameters variable. This structure accomodates the information referring to the codec that was used in the stream.

The avcodec_find_decoder function from the FFmpeg library is used to try to find the decoder for the codec ID supplied in av_codec_parameters->codec_id in the following line. The code uses the continue command to go on to the next iteration if the decoder cannot be located.

The function updates the video_stream_index variable to the current index i if the codec type of the current stream is AVMEDIA_TYPE_VIDEO. Additionally, av_codec_parameters->width and av_codec_parameters->height are used to get the video's width and height, respectively. Additionally it gives the time_base variable access to the AVStream's time_base property. Since the video stream has been located, it finally uses the break command to exit the loop.

The code updates the audio_stream_index variable to the current index i and exits the loop using the break command if the codec type of the current stream is AVMEDIA_TYPE_AUDIO. This demonstrates the discovery of the audio stream.

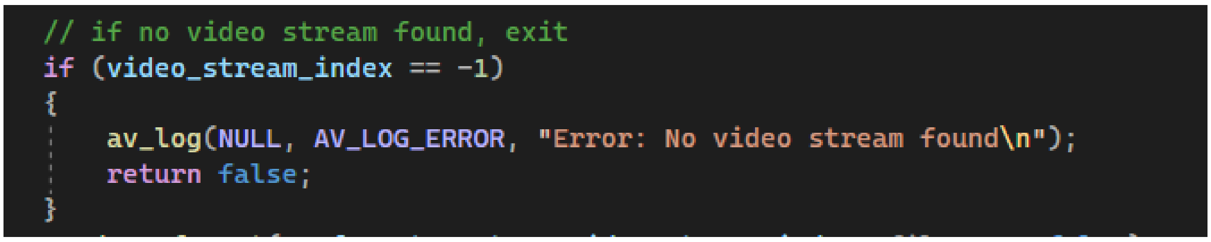


```
// get video and audio stream index
for (int i = 0; i < av_format_contex->nb_streams; i++){
    AVStream* av_streams = av_format_contex->streams[i];
    av_codec_parameters = av_format_contex->streams[i]->codecpar;
    av_codec = avcodec_find_decoder(av_codec_parameters->codec_id);

    if (!av_codec) {
        continue;
    }
    if (av_codec_parameters->codec_type == AVMEDIA_TYPE_VIDEO) {
        video_stream_index = i;
        width = av_codec_parameters->width;
        height = av_codec_parameters->height;
        time_base =av_format_contex->streams[i]->time_base;
        break;
    }

    if (av_codec_parameters->codec_type == AVMEDIA_TYPE_AUDIO) {
        audio_stream_index = i;
        break;
    }
}
```

The activities carried out by this block of code pertain to the video stream found in the one before it.

The first line determines whether the value of the variable video_stream_index is zero. If it is, it indicates that the input file did not contain a video stream. Using the av_log function, an error message is then recorded with the severity set to AV_LOG_ERROR. No video stream was identified according to the error notice, and the method returns false to indicate failure.

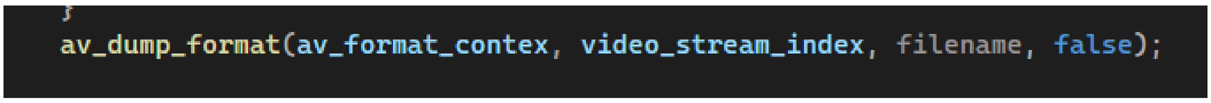
The code moves on to the following line if the video stream index is not zero. The following arguments are sent to the av_dump_format function when it is called: av_format_contex (the AVFormatContext pointer), video_stream_index (the index of the video stream), filename (the name of the input file), and false (to indicate whether or not to publish specific information about the format). The codec, duration and other metadata about the specified video stream are printed by this function along with information on its format.



```
// if no video stream found, exit
if (video_stream_index == -1)
{
    av_log(NULL, AV_LOG_ERROR, "Error: No video stream found\n");
    return false;
}
```

The FFmpeg library provides a function called av_dump_format, which is used to show details about the streams and the AVFormatContext. The filename of the input file, the AVFormatContext pointer (av_format_contex), the video stream index (video_stream_index), and a flag indicating whether to show the output in a human-readable format (false in this case) are the other arguments.

When this code block is performed, it prints facts about the input file's format including the container format, information about the codec, and several stream attributes. It offers useful details about the streams in the input file, including the codec type bitrate, resolution, duration, and other metadata.

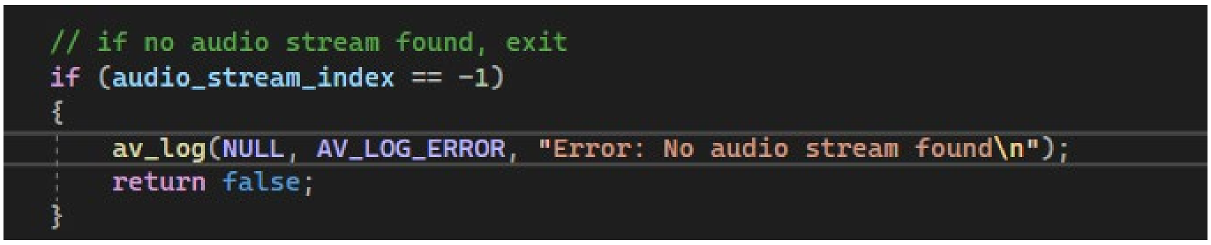


```
av_dump_format(av_format_contex, video_stream_index, filename, false);
```

The first line of code determines whether the value of the variable audio_stream_index is zero. If it is it means that the input file did not contain an audio stream. The av_log function is used to log an error in this situation. The error message notes the lack of an audio stream and the severity level is set to AV_LOG_ERROR. After that, the method returns false to denote failure.

This section of code checks to see if an audio stream is present before doing any more audio-related activities. The software responds appropriately by reporting an error message and returning false if no audio stream is discovered.

This code block's function is to offer diagnostic data for troubleshooting and analysis. Understanding the qualities of the input file and its streams is helpful since it may be applied to subsequent processing or troubleshooting.



```
// if no audio stream found, exit
if (audio_stream_index == -1)
{
    av_log(NULL, AV_LOG_ERROR, "Error: No audio stream found\n");
    return false;
}
```

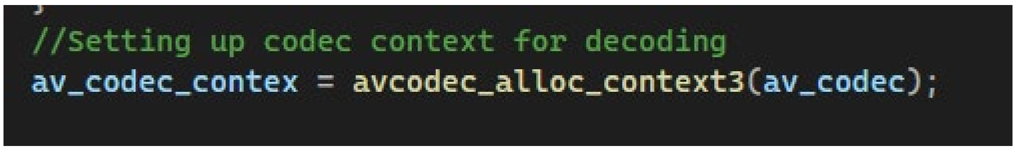
The block of codes below is responsible for initializing and opening the codec context using the codec parameters.

Allocating and initializing an AVCodecContext object for the chosen codec falls under the purview of the provided line of code.

The function avcodec_alloc_context3 is first called in the code, along with the input av_codec. The FFmpeg library provides this method, which is used to allocate a new AVCodecContext structure. The codec for which the context is being allocated is specified by the av_codec argument.

The AVCodecContext structure is allocated memory by the function, and its fields are initialized with default values appropriate for the selected codec. The AVCodecContext structure contains a number of codec-related settings and parameters, including bitrate frame rate, and encoding or decoding choices.

The variable av_codec_context will store a reference to the allocated and initialized AVCodecContext structure linked to the specified codec following the execution of this line of code. The codec may then be configured, data can be encoded or decoded, and codec-specific parameters can be managed using this context.



```
//Setting up codec context for decoding
av_codec_contex = avcodec_alloc_context3(av_codec);
```

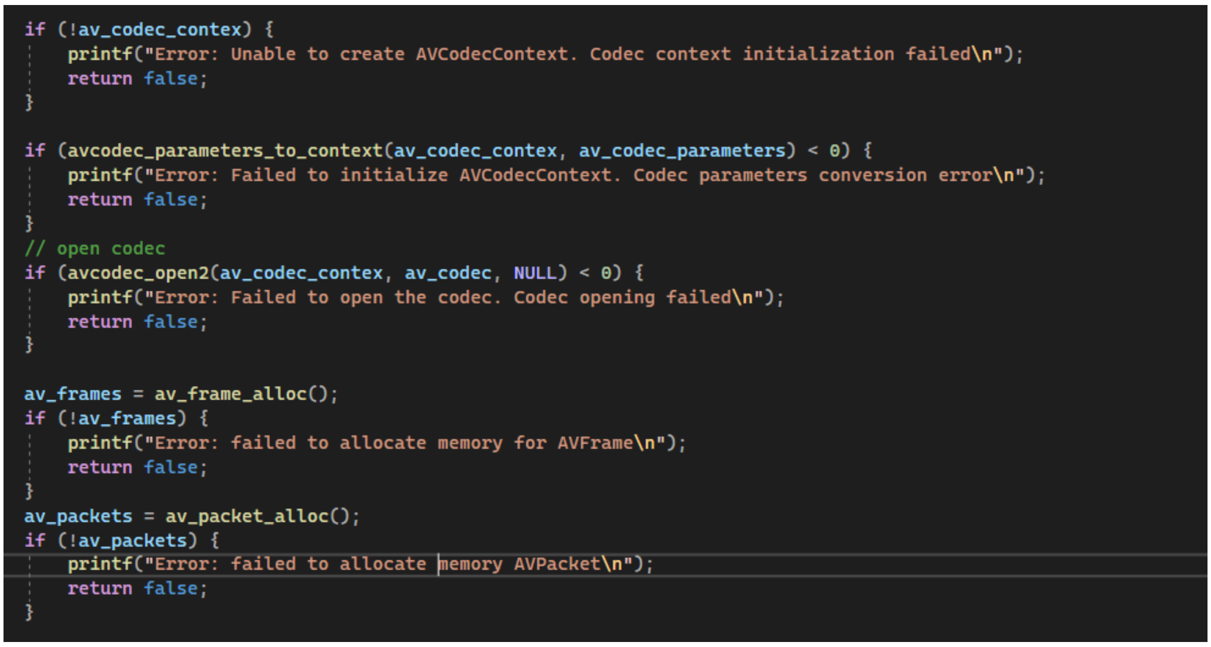
The first if statement determines whether the creation of the AVCodecContext av_codec_contex was successful. If it is NULL, this indicates a failure in the codec context's initialization. In this instance, an error message is written with the help of printf to show the failure and the associated error message. The function then returns false to show that an error happened.

The second if statement initializes the AVCodecContext av_codec_contex with the parameters taken from the AVCodecParameters av_codec_parameters using the function avcodec_parameters_to_context. If the conversion fails (< 0), it indicates that there was a conversion error and a message detailing the conversion fault is written with printf. The function also returns false in this case.

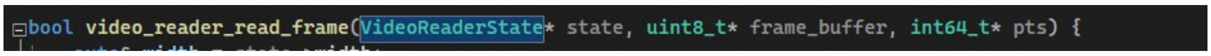
The AVCodecContext av_codec_contex is used in the following code block to open the codec given by av_codec. This is accomplished by use the avcodec_open2 function. The function returns false if the codec opening fails (< 0), at which point a message describing the failure to open the codec is written with printf.

Following that, av_frame_alloc() is used by the code to allocate memory for an AVFrame. An issue occurred while allocating memory, and a message is written using printf if the allocation fails or the resultant frame is NULL. After that the function returns false.

Finally, the code uses av_packet_alloc() to allocate memory for an AVPacket. An error occurred while allocating memory and a message is sent using printf if the allocation fails or the resultant packet is NULL. After that, the function returns false.

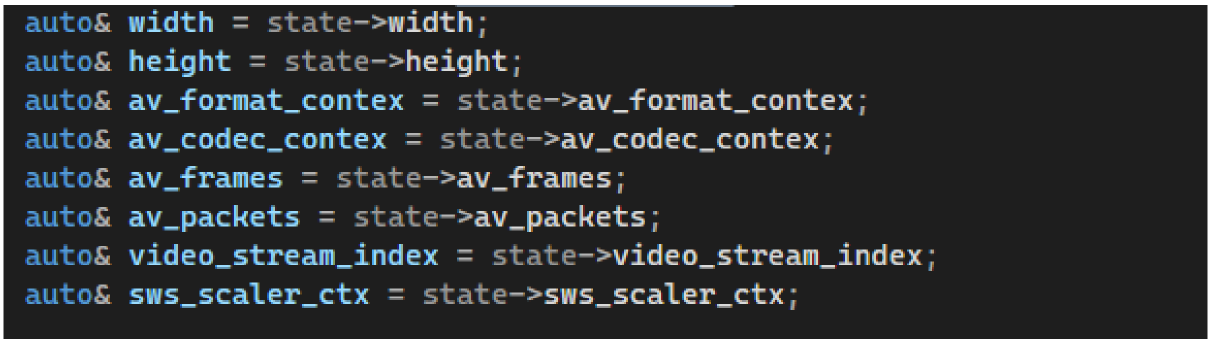


```
if (!av_codec_contex) {
    printf("Error: Unable to create AVCodecContext. Codec context initialization failed\n");
    return false;
}

if (avcodec_parameters_to_context(av_codec_contex, av_codec_parameters) < 0) {
    printf("Error: Failed to initialize AVCodecContext. Codec parameters conversion error\n");
    return false;
}
// open codec
if (avcodec_open2(av_codec_contex, av_codec, NULL) < 0) {
    printf("Error: Failed to open the codec. Codec opening failed\n");
    return false;
}

av_frames = av_frame_alloc();
if (!av_frames) {
    printf("Error: failed to allocate memory for AVFrame\n");
    return false;
}
av_packets = av_packet_alloc();
if (!av_packets) {
    printf("Error: failed to allocate memory AVPacket\n");
    return false;
}
```



```
bool video_reader_read_frame(VideoReaderState* state, uint8_t* frame_buffer, int64_t* pts) {
```

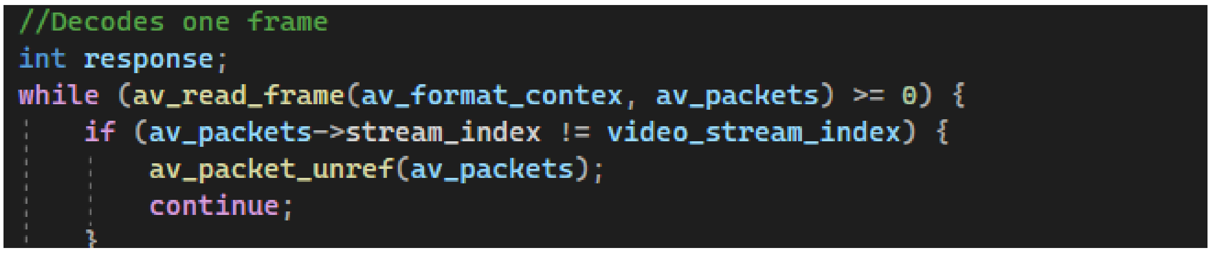
To proceed in my function, I extract the necessary members from the VideoReaderState structure to gain access during processing. These members include the width, height, format context codec context, frame, packet, video stream index, and scaler context.



```
auto& width = state->width;
auto& height = state->height;
auto& av_format_contex = state->av_format_contex;
auto& av_codec_contex = state->av_codec_contex;
auto& av_frames = state->av_frames;
auto& av_packets = state->av_packets;
auto& video_stream_index = state->video_stream_index;
auto& sws_scaler_ctx = state->sws_scaler_ctx;
```

Each packet from the video file is read and processed by the first code block: while (av_read_frame(av_format_contex, av_packets) >= 0). To read the next available packet from the video file linked to the av_format_contex (AVFormatContext), the method av_read_frame() is called inside the loop. The loop moves on to the following iteration if the function returns a non-negative value signifying successful packet reading, or zero, signifying the end of the file or stream. By doing this, the video file's whole packet collection is processed. The loop will end if the function returns a negative value, which indicates a problem with the packet reading or the fact that no further packets are available.

The if statement: if (av_packets->stream_index!= video_stream_index) determines if the current packet's stream index (av_packets->stream_index) corresponds to the anticipated video stream index (video_stream_index). We are particularly interested in processing video packets since a video file may contain several streams, including video, audio subtitles, and others. Therefore, to distinguish video packets, we compare the stream index. We go on to the next iteration of the loop if the condition evaluates to true, which indicates that the current packet does not belong to the video stream. We release the resources linked to the packet using av_packet_unref (av_packets) before going on to the next iteration. The packet is now ready for the following iteration or for being released after all packets have been processed, which ensures correct memory management.

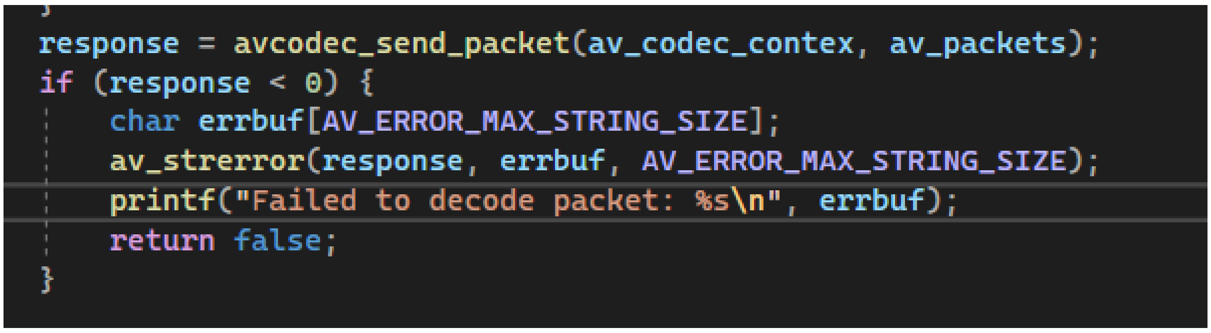


```
//Decodes one frame
int response;
while (av_read_frame(av_format_contex, av_packets) >= 0) {
    if (av_packets->stream_index != video_stream_index) {
        av_packet_unref(av_packets);
        continue;
    }
```

The packet is sent to the decoder for decoding via the code block response = avcodec_send_packet(av_codec_contex, av_packets). The AVCodecContext av_codec_contex and the packet av_packets are input parameters for the function avcodec_send_packet(). In order to decode the packet it sends it to the decoder connected to the context of the codec. The return value of this function, which denotes whether the operation was successful or unsuccessful, is stored in the response variable.

A failure to deliver the packet to the decoder is indicated by the next if line if (response 0), which checks if the response value is less than 0. If the condition is met, the packet could not be correctly decoded. In this instance, an error message is created using the av_strerror() function, which transforms the error code into a human-readable error message saved in the errbuf character array. The inability to decode the packet is then communicated via the error message which is displayed using the printf() function. When the decoding attempt is unsuccessful, the function finally returns false.

This code block is crucial for handling the case where packet decoding fails. It enables the application to recognize and deal with any problems that take place when decoding. The code can tell if a packet was properly transmitted for decoding or whether an error occurred by examining the return value. This makes it possible to handle errors correctly and assures that the program's decoding capability is resilient.

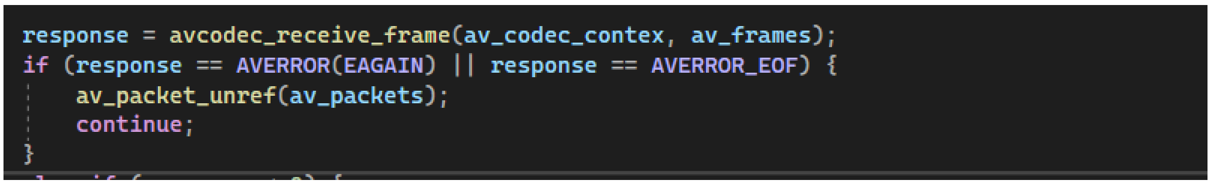


```
response = avcodec_send_packet(av_codec_contex, av_packets);
if (response < 0) {
    char errbuf[AV_ERROR_MAX_STRING_SIZE];
    av_strerror(response, errbuf, AV_ERROR_MAX_STRING_SIZE);
    printf("Failed to decode packet: %s\n", errbuf);
    return false;
}
```

A decoded frame is received from the decoder via the code block response = avcodec_receive_frame(av_codec_contex, av_frames). The AVCodecContext av_codec_contex and the AVFrame av_frames are input parameters for the method avcodec_receive_frame(). It makes an attempt to locate a decoded frame in the decoder connected to the codec context. The return value of this function, which denotes whether or not receiving a frame was successful, is stored in the response variable.

Following that the if condition if (response == AVERROR(EAGAIN) || response == AVERROR_EOF) determines if the response value is equivalent to AVERROR(EAGAIN) or AVERROR_EOF. While AVERROR_EOF indicates that the input stream has ended, AVERROR(EAGAIN) indicates that further input data is required in order to generate an output frame. The frame is released using av_packet_unref() to free up resources if one of these conditions holds true, which indicates that the frame couldn't be received at the time. The loop then moves on to the next iteration.

For managing the flow of receiving frames from the decoder, this code block is crucial. It enables the program to respond to circumstances such as when a frame is not immediately accessible or when the input stream reaches its end. The code can evaluate whether more action is necessary such as continuing to receive frames or addressing the stream's termination by comparing the response value against particular error circumstances. This guarantees correct decoding flow management and error handling.



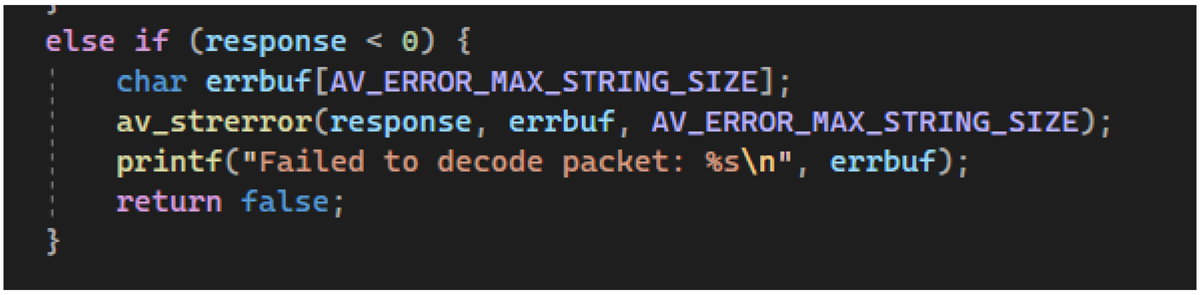
```
response = avcodec_receive_frame(av_codec_contex, av_frames);
if (response == AVERROR(EAGAIN) || response == AVERROR_EOF) {
    av_packet_unref(av_packets);
    continue;
}
```

If the preceding condition in which the response value is not equal to AVERROR(EAGAIN) or AVERROR_EOF is false then the code block else if (response 0) is examined. It deals with the situation when the response value is less than zero, signifying a failure to correctly decode the packet and receive a frame.

This block declares a character array named errbuf with the size AV_ERROR_MAX_STRING_SIZE to hold the error message related to the response value. The response value, the errbuf array, and the array's maximum size are then sent as parameters to the av_strerror() method. This method writes the error code's string representation into the errbuf array.

The error message is then displayed using the printf() function, resulting in the formatted text Failed to decode packet:%sn where %s stands in for the error message contained in errbuf.

The function then returns false to show that there was a problem with the decoding operation.



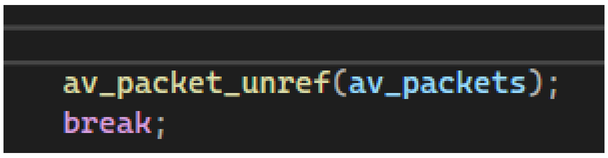
```
else if (response < 0) {
    char errbuf[AV_ERROR_MAX_STRING_SIZE];
    av_strerror(response, errbuf, AV_ERROR_MAX_STRING_SIZE);
    printf("Failed to decode packet: %s\n", errbuf);
    return false;
}
```

After successfully receiving a frame during the decoding process the code block av_packet_unref(av_packets); break; is performed.

The av_packets parameter is sent to the method av_packet_unref() in this block. This function releases the packet's associated resources including any internal data structures and allocated memory so they can be utilized by succeeding packets. When a packet is unreferenced it means that it is no longer required and may be securely discarded away.

When the break statement is met after releasing the packet, the while loop is exited by the application. After successfully decoding a frame, the program might proceed to additional processing or actions because of this. The requested frame has been received, thus there is no need to read or process any further packets. The break statement is used to break out of the loop.

In general, this code block is in charge of releasing the resources connected to the packet and notifying the end of the loop following successful frame decoding. It makes sure that memory is managed effectively and enables the software to continue using the decoded frame for additional operations or processing.



```
av_packet_unref(av_packets);
break;
```

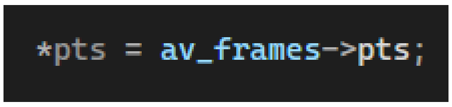
The line of code sets the variable referred to by pts to the value of the pts field of the av_frames structure.

The term av_frames in this instance refers to the AVFrame structure which stands for a decoded video frame. The presentation timestamp (PTS) of the frame is stored in the pts field of the AVFrame structure. The desired display time of each frame in the series of frames is indicated by the PTS.

The value of the PTS is given to the variable denoted by the pointer pts by assigning av_frames->pts to *pts. This enables the PTS value to be accessed and used by the caller code or function for additional processing or analysis.

This line of code's objective is to grant outside access to the decoded frame's PTS. The value can be accessed and utilized outside of the current function or scope where the decoding operation is taking place by assigning the PTS to the variable pointed to by pts. It gives the caller access to the timestamp data for the frame which is useful for synchronization editing, and other procedures that need precise timing inside a video stream.

This line of code's objective is to grant outside access to the decoded frame's PTS. The value can be accessed and utilized outside of the current function or scope where the decoding operation is taking place by assigning the PTS to the variable pointed to by pts. It gives the caller access to the timestamp data for the frame which is useful for synchronization, editing, and other procedures that need precise timing inside a video stream.



```
*pts = av_frames->pts;
```

The code block printf(Frame %c (%d) pts %d dts %d key_frame %d [coded_picture_number %d, display_picture_number %d\n], av_get_picture_type_char(av_frames->pict_type), av_codec_contex->frame_number, av_frames->pts av_frames->pkt_dts, av_frames->key_frame, av_frames->coded_picture_number, av_frames->display_picture_number); is responsible for printing information about a frame that has been decoded.

The provided message is formatted and printed to the console or standard output using the printf() function. A formatted string containing placeholders marked by %c, %d, and %s is present in the message. These placeholders are replaced with the equivalent values given as arguments to printf().

%c: This placeholder is replaced by the character representation of the picture type obtained from av_get_picture_type_char(av_frames->pict_type). It represents the type of the picture, such as I-frame, P-frame, or B-frame.

%d: These placeholders are replaced by the integer values obtained from various fields. Specifically:

av_codec_contex->frame_number: It represents the frame number or index of the current frame.

av_frames->pts: It represents the presentation timestamp of the frame.

av_frames->pkt_dts: It represents the decoding timestamp of the frame.

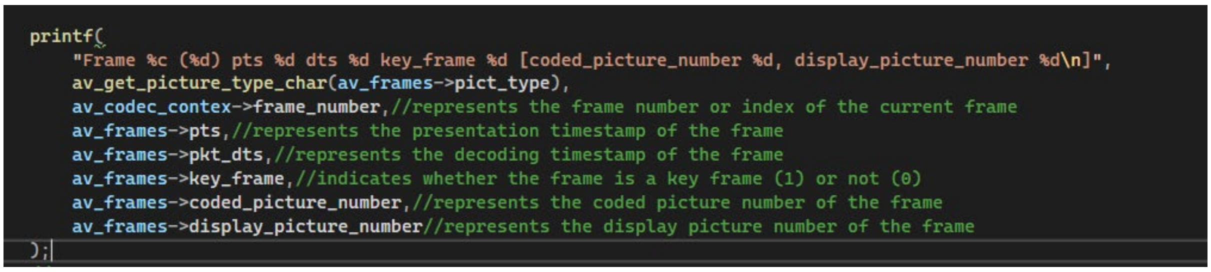
av_frames->key_frame: It indicates whether the frame is a key frame (1) or not (0).

av_frames->coded_picture_number: It represents the coded picture number of the frame.

av_frames->display_picture_number: It represents the display picture number of the frame.

The code block displays detailed information about the decoded frame, including its type, timestamps, key frame status and image numbers, by using printf() with the formatted text and the relevant values.

This code block is essential for providing details about the attributes and traits of the decoded frames, assisting with additional video data analysis, debugging, or visualization.



```
printf(
    "Frame %c (%d) pts %d dts %d key_frame %d [coded_picture_number %d, display_picture_number %d\n]",
    av_get_picture_type_char(av_frames->pict_type),
    av_codec_contex->frame_number,//represents the frame number or index of the current frame
    av_frames->pts,//represents the presentation timestamp of the frame
    av_frames->pkt_dts,//represents the decoding timestamp of the frame
    av_frames->key_frame,//indicates whether the frame is a key frame (1) or not (0)
    av_frames->coded_picture_number,//represents the coded picture number of the frame
    av_frames->display_picture_number//represents the display picture number of the frame
);
```

The code block determines if the sws_scaler_context variable has not yet been initialized, which means it is currently set to null. The !sws_scaler_context condition determines if sws_scaler_context is false or null.

In the event that the sws_scaler_context is not initialized (that is if it is null) the code moves into the if statement's body. To construct and initialize the sws scaler context the sws_getContext() method is used in the body.

The arguments are the parameters provided to sws_getContext():

width and height: These variables represent the dimensions of the input video frames.

av_codec_contex->pix_fmt:This accesses the pix_fmt property of the av_codec_contex structure which represents the pixel format of the input video frames.

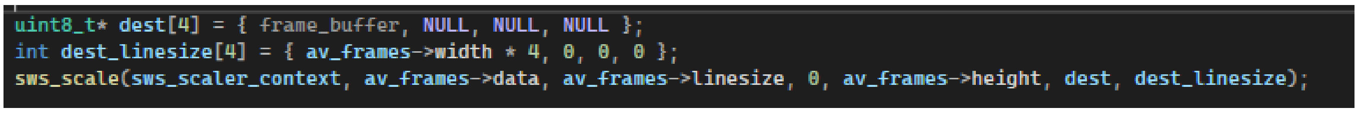
AV_PIX_FMT_RGB0: The constant AV_PIX_FMT_RGB0 designates the preferred pixel format for the output video frames. This code sample has it set to RGB which has 32 bits per pixel and 8 bits each component.

SWS_BILINEAR: The constant SWS_BILINEAR designates the scaling method the sws scaler will employ. It scales in this instance via bilinear interpolation.

NULL NULL NULL: These are extra parameters that can be used for gamma correction or other additional scaling settings. These arguments are set to null in this section of code suggesting default settings.

A pointer to the freshly constructed sws scaler context is returned by the sws_getContext() method and is then assigned to the sws_scaler_context variable.

This code block's goal is to determine whether the sws scaler context has been initialized. If not, it uses the input and output parameters to establish and initialize the sws scaler context. This guarantees that the SWS scaler is correctly configured and prepared to carry out pixel format conversions as necessary. The efficiency of successive frame conversions can be increased by avoiding needless reinitializations and only initializing the sws scaler context when necessary.

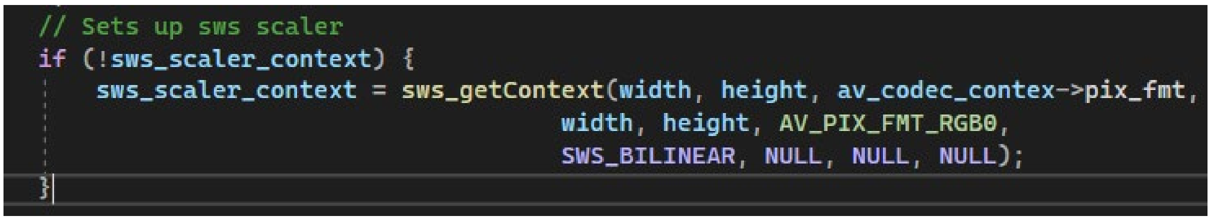


```
uint8_t* dest[4] = { frame_buffer, NULL, NULL, NULL };
int dest_linesize[4] = { av_frames->width * 4, 0, 0, 0 };
sws_scale(sws_scaler_context, av_frames->data, av_frames->linesize, 0, av_frames->height, dest, dest_linesize);
```

The destination buffer and its corresponding line widths are set up in the first line of code. Four members make up the dest array, which is created by the uint8_t* dest[4] declaration. A pointer to an 8-bit unsigned integer is contained in each element. The value of the variable frame_buffer, which stands for the destination buffer for the scaled picture, is set to the first element, frame_buffer. Indicating that they are not utilized in this particular code piece, the final three elements are initialized with NULL. The pointers to the output image planes will be stored in this array.

The int dest_linesize[4] declaration on the second line establishes a four-element array with the name dest_linesize. The line size of each element's matching picture plane is represented by a number for each element. The width of the av_frames picture multiplied by four is the value that is supplied to the first element, av_frames->width * 4. In order to estimate the stride of each picture plane, this computation establishes the number of bytes per line in the destination buffer. The three extra components are not utilized in this particular code piece since they are started with 0.

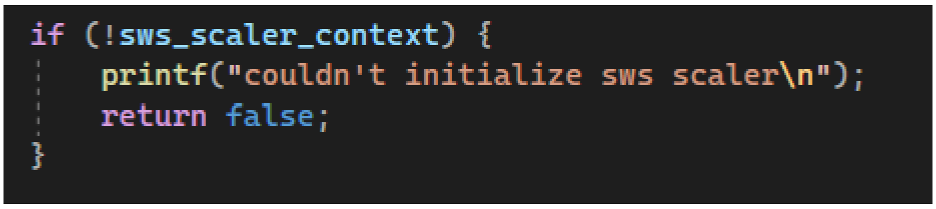
The third line then invokes the sws_scale function to scale and convert the picture. This function, which is a component of the FFmpeg library, is used for operations like format conversion and resizing. The input and output pictures, scale parameters, and other variables are specified via a number of options. A pointer to a SwsContext struct, which contains the configuration and state details for the scaling and conversion process, is used in this function as the sws_scaler_context argument. An array of pointers to the input picture planes that represent the color channels is included in the av_frames->data argument. The line sizes of each input image plane are specified by the array of integers in the av_frames->linesize argument. The source slice height is indicated by the 0 parameter, and a value of 0 indicates that the full input frame will be processed. The height of the input frame is represented by the av_frames->height parameter. The destination buffer (frame_buffer) is the first element in an array of pointers to the output picture planes that makes up the dest parameter. The line size of the destination buffer is the first member of the array of numbers that make up the dest_linesize parameter, which represents the line sizes of each output image plane. These variables work together to allow the sws_scale function to carry out the required picture scaling and format conversion operations, resulting in the intended output in the target buffer.



```
// Sets up sws scaler
if (!sws_scaler_context) {
    sws_scaler_context = sws_getContext(width, height, av_codec_contex->pix_fmt,
                                        width, height, AV_PIX_FMT_RGB0,
                                        SWS_BILINEAR, NULL, NULL, NULL);
}
```

The logical NOT operator (!) is used in the code block to determine whether the variable sws_scaler_context is null. If the sws_scaler_context is not initialized or allocated, this condition evaluates to true. The sws scaler context has not yet been generated if the condition is true. The code executes the if statement in this situation. There is another if (!sws_scaler_context) to make sure if the first code of the same function failed, it gives back an error.

The printf function, which is called inside the if statement, prints the error couldn't initialize sws scaler to the console. This error indicates that the sws scaler context could not be initialized.



```
if (!sws_scaler_context) {
    printf("couldn't initialize sws scaler\n");
    return false;
}
```

A pointer to a structure of the type VideoReaderState serves as the parameter for the method video_reader_close represented by the code block. The resources connected to the video reader state need to be cleared out using this function.

In the first line of code sws_freeContext(state->sws_scaler_context) the scaler context pointer that is held in the sws_scaler_context field of the VideoReaderState structure is sent to the sws_freeContext function. By doing this, it is guaranteed that all memory and resources allocated for the scaler context are appropriately released

The following line, avformat_close_input(&state->av_format_contex), calls the avformat_close_input function and passes the address of the av_format_contex field of the VideoReaderState structure to close the input format context. This function carries out the appropriate input format cleanup procedures, such as closing the input file that has been opened and freeing any relevant resources.

The third line avformat_free_context(state->av_format_contex) calls the avformat_free_context function and passes the av_format_contex field of the VideoReaderState structure. This releases the format context. The RAM assigned for the format context is released together with any remaining resources by this function.

By calling the av_packet_free function and giving the address of the av_packets field of the VideoReaderState structure, the fifth line, av_packet_free(&state->av_packets), releases the packet. The memory utilized by the packet structure and any related data is released by this function.

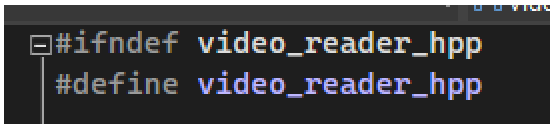
By calling the avcodec_free_context function and giving the address of the av_codec_contex field of the VideoReaderState structure, the final line, avcodec_free_context(&state->av_codec_contex), frees the codec context. The memory allotted for the codec context and any associated resources are released by this function.

The video_reader_close function guarantees that all allocated resources are correctly removed by carrying out these cleaning actions preventing memory leaks and freeing up system resources for other activities. To ensure efficient use of resources it is essential to call this function when the video reader is no longer required or when the application is closing.

2.3.2.2 video_reader.cpp

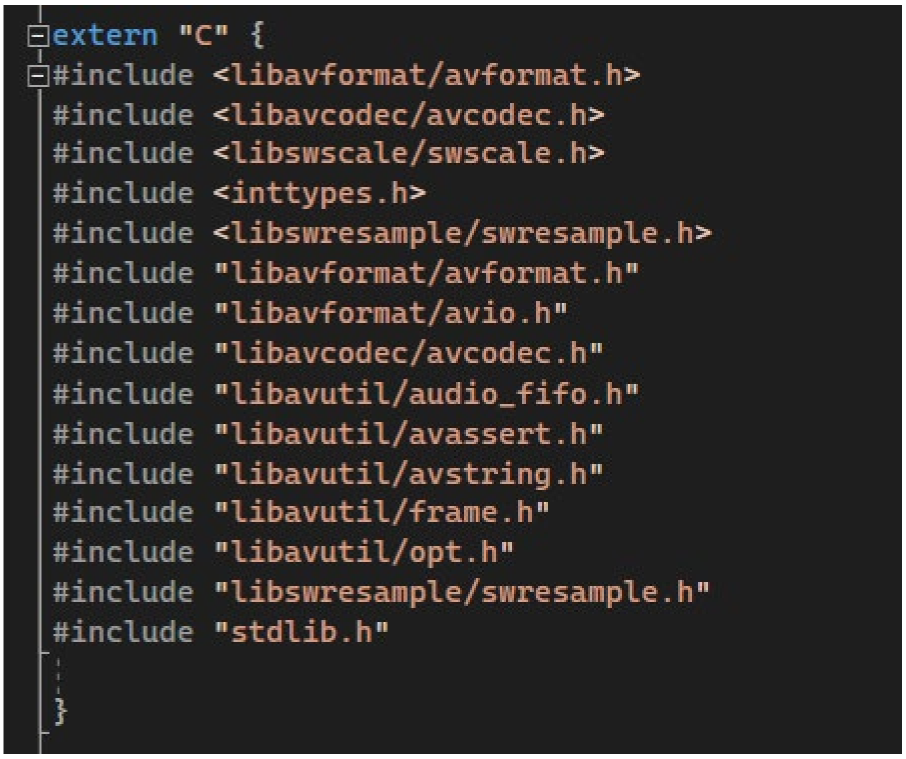
The video_reader.hpp header file, which contains the required libraries and specifies a struct and a number of function prototypes relevant to video reading using the FFmpeg library, is shown below.

The preprocessor command #ifndef determines if the name video_reader_hpp has already been declared. The succeeding lines of code will be included if it hasn't been specified. This design feature makes ensuring the header file's content is only included once, avoiding duplicate inclusions.



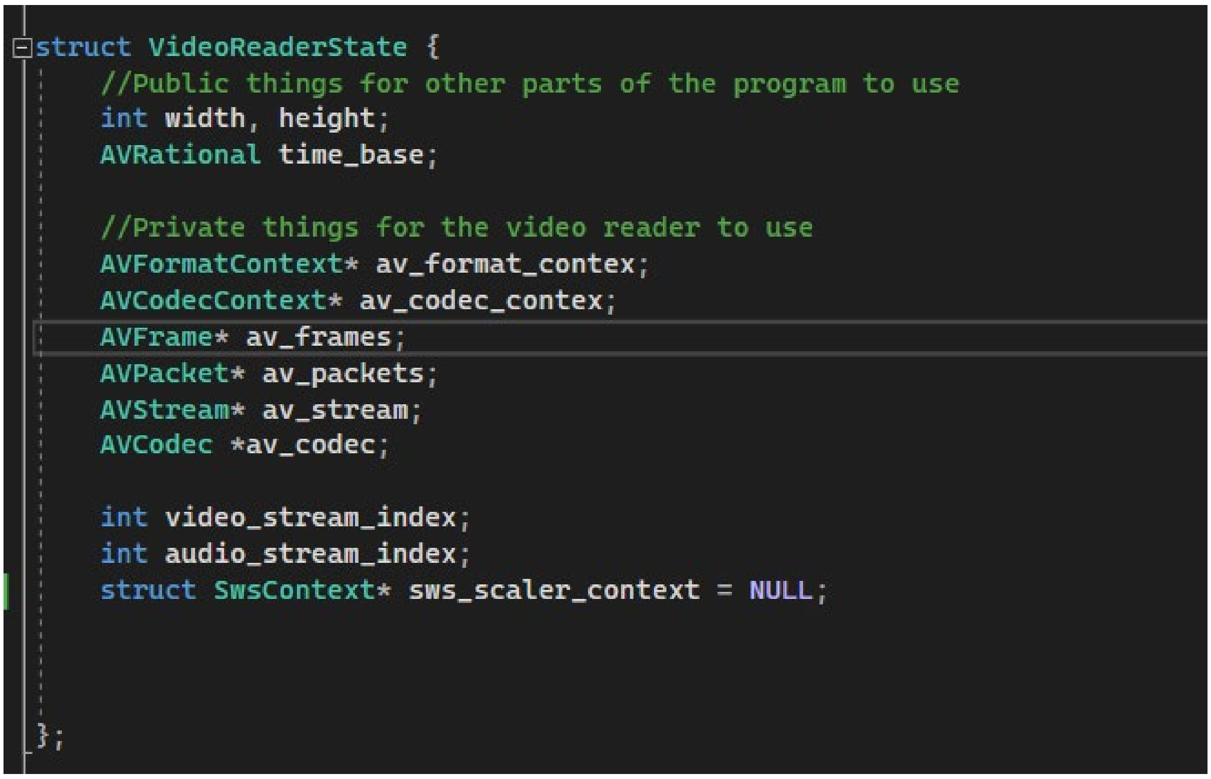
```
#ifndef video_reader_hpp
#define video_reader_hpp
```

The extern C block is used to specify that C code, rather than C++, should be generated from the included C header files (libavformat/avformat.h, libavcodec/avcodec.h, etc.). The FFmpeg library was created in C, hence this is required.



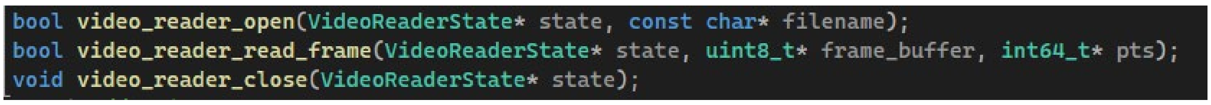
```
extern "C" {
#include <libavformat/avformat.h>
#include <libavcodec/avcodec.h>
#include <libswscale/swscale.h>
#include <inttypes.h>
#include <libswresample/swresample.h>
#include "libavformat/avformat.h"
#include "libavformat/avio.h"
#include "libavcodec/avcodec.h"
#include "libavutil/audio_fifo.h"
#include "libavutil/avassert.h"
#include "libavutil/avstring.h"
#include "libavutil/frame.h"
#include "libavutil/opt.h"
#include "libswresample/swresample.h"
#include "stdlib.h"

}
```

A struct called VideoReaderState that acts as a container for the state and data pertaining to video reading is defined in the header file. It features open components like width, height, and time_base that are used to hold the video's width, height, and time base, respectively. These public members are meant to be used by other software components. It also includes references to other FFmpeg data structures including AVFormatContext AVCodecContext, AVFrame, AVPacket, AVStream, and AVCodec in its private members. These structures are utilized by the video reader internally and are not intended to be directly accessible from the outside.



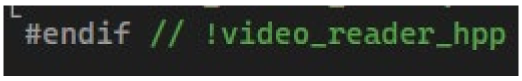
```
struct VideoReaderState {
    //Public things for other parts of the program to use
    int width, height;
    AVRational time_base;

    //Private things for the video reader to use
    AVFormatContext* av_format_contex;
    AVCodecContext* av_codec_contex;
    AVFrame* av_frames;
    AVPacket* av_packets;
    AVStream* av_stream;
    AVCodec *av_codec;

    int video_stream_index;
    int audio_stream_index;
    struct SwsContext* sws_scaler_context = NULL;



};
```

Three function prototypes are also declared in the header file: video_reader_open video_reader_read_frame and video_reader_close. These procedures are in charge of starting a video file reading video frames and shutting off the video reader, in that order. They accept a reference to VideoReaderState and other required parameters as input and output a boolean value that indicates whether the relevant operation was successful or unsuccessful.



```
bool video_reader_open(VideoReaderState* state, const char* filename);
bool video_reader_read_frame(VideoReaderState* state, uint8_t* frame_buffer, int64_t* pts);
void video_reader_close(VideoReaderState* state);
```

The conditional inclusion block comes to a conclusion using the #endif directive. It guarantees that the material in between is only included once and matches the opening #ifndef directive.



```
#endif // !video_reader_hpp
```

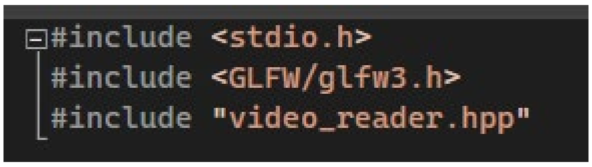
2.3.2.3 main.cpp

The provided code is a C++ source file that includes several header files and a custom header file named video_reader.hpp.

The #include directive is used to include external header files in the source file. In this case, it includes two standard header files: <stdio.h>, which provides input/output functions and <GLFW/glfw3.h> which is a header file for the GLFW library used for creating windows and handling OpenGL contexts. The inclusion of these headers allows the source file to access the functions, types, and constants defined in those libraries.

Additionally, the source file includes the custom header file video_reader.hpp using double quotes () instead of angle brackets (<>). The use of double quotes is typically used for including local project-specific header files. This header file likely contains the declarations and definitions related to video reading using the FFmpeg library, as explained in the previous explanation.

By including these header files, the source file gains access to the functionalities provided by the included libraries and the custom video reader module allowing it to utilize functions types and constants defined in those headers in its own code.



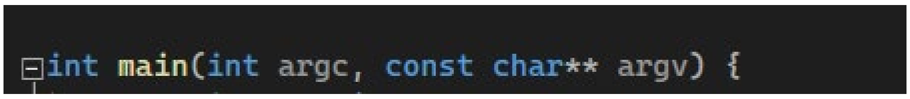
```
#include <stdio.h>
#include <GLFW/glfw3.h>
#include "video_reader.hpp"
```

It defines the main function, which is the starting point of execution for the program. The main function takes two parameters: argc, an integer representing the number of command-line arguments passed to the program, and argv, a pointer to an array of strings containing the command-line arguments.

The int before main indicates that the function returns an integer value that represents the program's exit status. By convention, a return value of 0 usually indicates successful execution while non-zero values indicate some form of error or abnormal termination.

The use of const char** argv means that the elements of the argv array are pointers to constant strings. These strings represent the command-line arguments passed to the program, with argv[0] typically being the program's name or path.

Inside the main function, you can write the program's logic, which will be executed when the program is run. The specific implementation of the main function will depend on the purpose of the program. It could involve processing the command-line arguments, calling other functions or modules performing computations or interacting with external resources.



```
int main(int argc, const char** argv) {
```

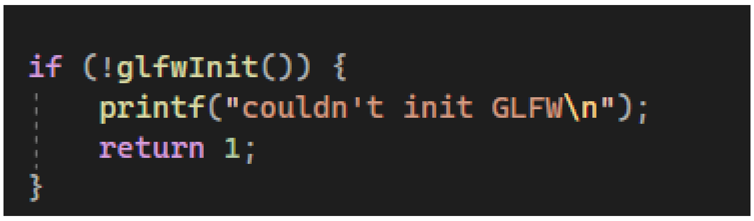
This code snippet is a conditional statement that checks if the GLFW library initialization was successful.

The glfwInit() function is a part of the GLFW library and is used to initialize the GLFW library and its associated resources. It sets up the necessary internal structures and prepares the library for further use. The function returns a boolean value indicating the success or failure of the initialization process.

The conditional statement if (!glfwInit()) checks if the return value of glfwInit() is false, which means the initialization failed. The exclamation mark ! is a logical negation operato, so !glfwInit() evaluates to true if glfwInit() returns false.

If the initialization fails the code inside the if block is executed. In this case it prints the error message couldn't init GLFW using printf() function and then returns 1. The return value of 1 is commonly used to indicate an error or abnormal termination of the program.

The purpose of this code is to handle the case where GLFW initialization fails. It provides an error message and terminates the program if GLFW initialization is unsuccessful ensuring that further execution of the program does not occur without the required initialization.



```c
if (!glfwInit()) {
    printf("couldn't init GLFW\n");
    return 1;
}
```

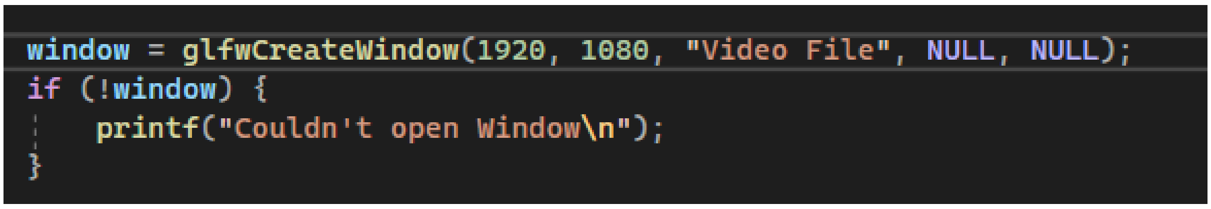
The provided code creates a GLFW window with a resolution of 1920x1080 pixels and a title of Video File.

The function glfwCreateWindow() is called to create the window. It takes several parameters: the width and height of the window (1920 and 1080, respectively), the window title (Video File), and optional parameters for monitor and shared context. In this case, the last two parameters are set to NULL indicating that they are not used.

The return value of glfwCreateWindow() is a pointer to the created window. The conditional statement if (!window) checks if the returned pointer is NULL which would indicate that the window creation failed.

If the window creation fails, the code inside the if block is executed. It uses the printf() function to print the error message Couldn't open Window to the console. This error message informs the user that the window creation was unsuccessful.

The purpose of this code is to handle the case where the GLFW window creation fails. It provides an error message to inform the user about the failure, allowing them to take appropriate action or terminate the program gracefully.



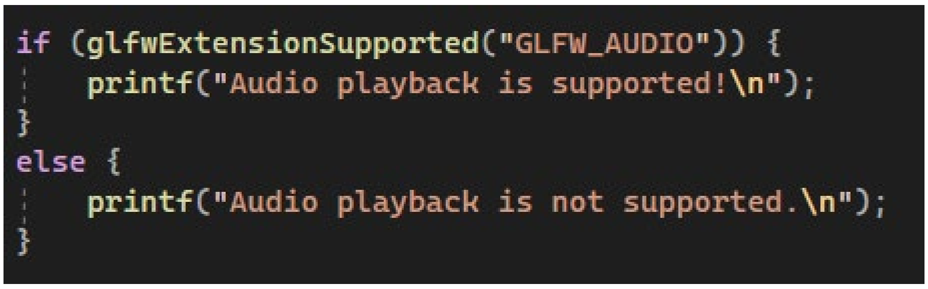
```c
window = glfwCreateWindow(1920, 1080, "Video File", NULL, NULL);
if (!window) {
    printf("Couldn't open Window\n");
}
```

The provided code checks if the GLFW library supports audio playback by using the function glfwExtensionSupported().

The conditional statement if (glfwExtensionSupported(GLFW_AUDIO)) checks if the extension named GLFW_AUDIO is supported by GLFW. If the extension is supported, the code inside the if block is executed. It uses the printf() function to print the message Audio playback is supported! to the console.

If the extension is not supported the code inside the else block is executed. It uses the printf() function to print the message Audio playback is not supported. to the console.

The purpose of this code is to check whether the GLFW library has the necessary extensions to support audio playback. It provides a way to inform the user about the audio playback support status of the GLFW library.



```
if (glfwExtensionSupported("GLFW_AUDIO")) {
    printf("Audio playback is supported!\n");
}
else {
    printf("Audio playback is not supported.\n");
}
```

The code below initializes a VideoReaderState object named vr_state and calls the video_reader_open() function to open a video file.

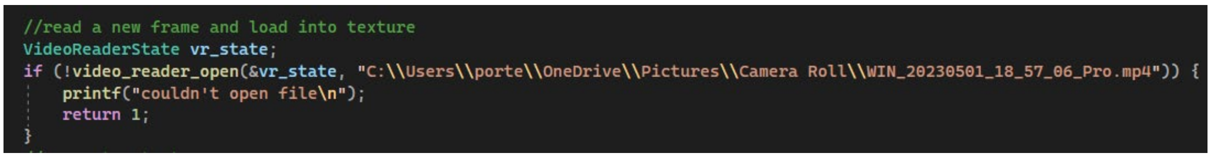
The line VideoReaderState vr_state; declares a variable vr_state of type VideoReaderState which represents the state of a video reader. This object will be used to store information about the video file being opened.

The video_reader_open() function is called with two arguments: a pointer to the vr_state object and the path of the video file. The function attempts to open the specified video file and initializes the vr_state object with the relevant information from the file.

The conditional statement if (!video_reader_open(&vr_state C:\\Users\\porte\\OneDrive\\Pictures\\Camera Roll\\WIN_20230501_18_57_06_Pro.mp4)) checks the return value of the video_reader_open() function. If the function returns false, it means that the file couldn't be opened or there was an error during the opening process. In such a case, the code inside the if block is executed.

Inside the if block, the printf() function is used to print the message couldn't open file to the console, indicating that there was an error opening the video file. The function then returns 1 to indicate an error condition.

The purpose of this code is to initialize a VideoReaderState object and open a video file for reading. It provides error handling by checking the return value of the video_reader_open() function and printing an appropriate error message if the file opening fails.



```
//read a new frame and load into texture
VideoReaderState vr_state;
if (!video_reader_open(&vr_state, "C:\\Users\\porte\\OneDrive\\Pictures\\Camera Roll\\WIN_20230501_18_57_06_Pro.mp4")) {
    printf("couldn't open file\n");
    return 1;
}
```

The next bit of code is responsible for generating a texture in OpenGL for rendering context.

glfwMakeContextCurrent(window);: This line sets the current OpenGL rendering context to the specified window. The rendering context is necessary to perform OpenGL operations.

GLuint tex_handle;: This line declares an unsigned integer variable tex_handle to store the texture handle, which is a unique identifier for the texture object.

glGenTextures(1, &tex_handle);: This line generates a texture object and assigns it to tex_handle. The 1 parameter indicates that we want to generate a single texture object.

glBindTexture(GL_TEXTURE_2D tex_handle);: This line binds the generated texture object to the target GL_TEXTURE_2D. Subsequent texture operations will affect this bound texture.

glPixelStorei(GL_UNPACK_ALIGNMENT, 1);: This line sets the pixel storage mode for unpacking pixel data. It specifies that the pixel rows in memory are aligned on byte boundaries.

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);: This line sets the texture wrapping mode for the GL_TEXTURE_WRAP_S parameter, which represents the texture coordinate axis S. It sets the wrapping mode to GL_REPEAT, which repeats the texture when the texture coordinates exceed the range of [0, 1].

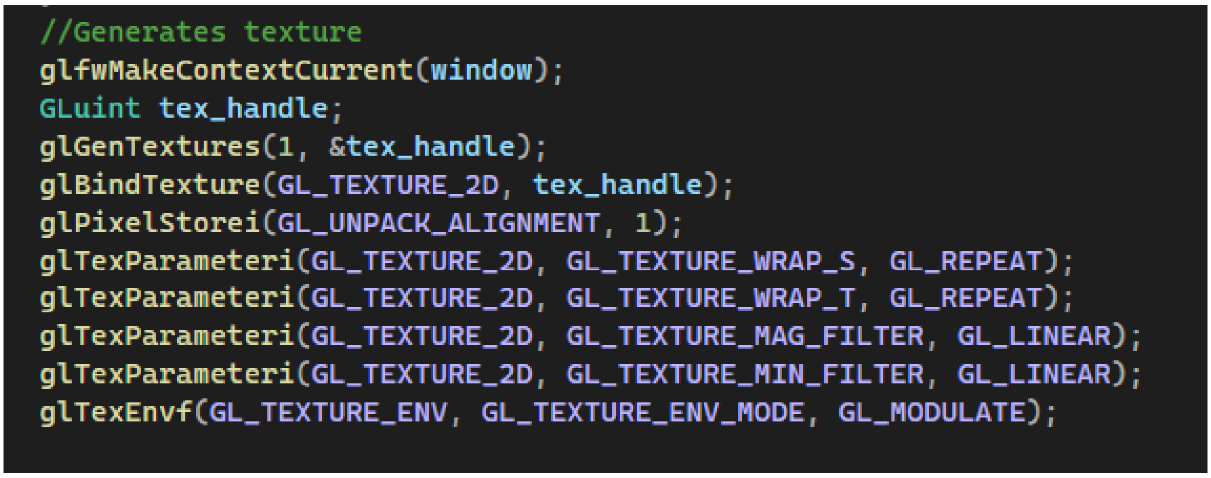
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);: This line sets the texture wrapping mode for the GL_TEXTURE_WRAP_T parameter, which represents the texture coordinate axis T. It sets the wrapping mode to GL_REPEAT, similar to the previous line.

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);: This line sets the texture magnification filter for the GL_TEXTURE_MAG_FILTER parameter. It sets the filter mode to GL_LINEAR, which performs linear interpolation to determine pixel values when the texture is magnified.

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);: This line sets the texture minification filter for the GL_TEXTURE_MIN_FILTER parameter. It sets the filter mode to GL_LINEAR, similar to the previous line.

glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);: This line sets the texture environment mode. It specifies how the texture color is combined with the underlying material color. In this case, it sets the mode to GL_MODULATE, which modulates the texture color with the material color.

These OpenGL functions collectively configure the texture parameters such as wrapping mode filtering, and environment mode, to define how the texture will be applied in the rendering pipeline.



```
//Generates texture
glfwMakeContextCurrent(window);
GLuint tex_handle;
glGenTextures(1, &tex_handle);
glBindTexture(GL_TEXTURE_2D, tex_handle);
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

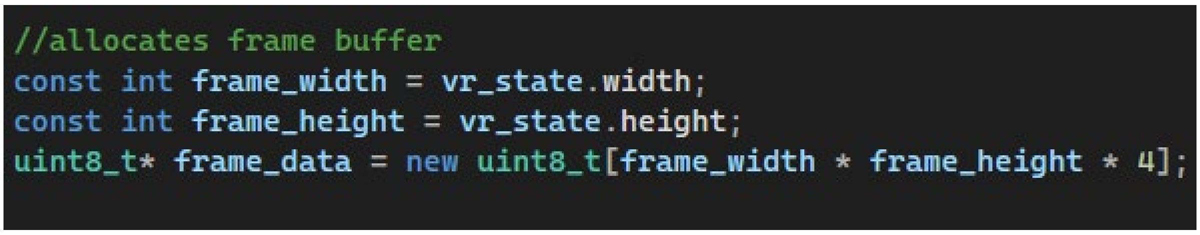
The code below is responsible for allocating a frame buffer, which is a memory buffer used to store pixel data for each frame. Here's an explanation of each line:

const int frame_width = vr_state.width;: This line declares a constant integer variable frame_width and assigns it the value of vr_state.width. It represents the width of the frame in pixels.

const int frame_height = vr_state.height;: This line declares a constant integer variable frame_height and assigns it the value of vr_state.height. It represents the height of the frame in pixels.

uint8_t* frame_data = new uint8_t[frame_width * frame_height * 4];: This line dynamically allocates memory for the frame buffer using the new operator. It creates an array of uint8_t (8-bit unsigned integers) with a size calculated as frame_width * frame_height * 4. The 4 represents the number of color components (RGBA) per pixel. The resulting memory block is assigned to the frame_data pointer variable, which can be used to access and manipulate the pixel data.

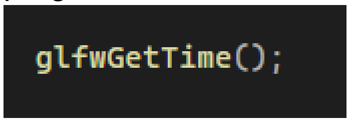
Overall these lines of code set up the necessary memory allocation for the frame buffer based on the width and height of the frame obtained from vr_state. The frame_data pointer will be used to store the pixel data of each frame in the subsequent video processing operations.



```
//allocates frame buffer
const int frame_width = vr_state.width;
const int frame_height = vr_state.height;
uint8_t* frame_data = new uint8_t[frame_width * frame_height * 4];
```

The code glfwGetTime() is a function call to retrieve the current time in seconds as a double value. It is provided by the GLFW library which is a utility library for creating and managing windows, OpenGL contexts, and handling user input.

When glfwGetTime() is called, it returns the elapsed time in seconds since GLFW was initialized or the system was started. This function is commonly used to measure time intervals or control animations and simulations based on real-time progress.



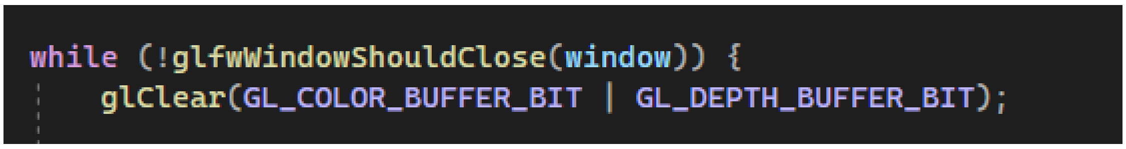
```
glfwGetTime();
```

The code represents a while loop that runs as long as the GLFW window should not be closed (!glfwWindowShouldClose(window)). This loop is responsible for rendering frames and handling events in the window.

Within the loop, the first action is to clear the color buffer and depth buffer using the glClear function. The GL_COLOR_BUFFER_BIT flag indicates that the color buffer should be cleared while the GL_DEPTH_BUFFER_BIT flag indicates that the depth buffer should be cleared. This operation ensures that the next frame starts with a clean slate removing any previous rendered content.

The color buffer stores the color values for each pixel in the window and the depth buffer stores the depth information for each pixel, allowing for accurate depth-based rendering. Clearing these buffers ensures that the rendered frame is not affected by any remnants from previous frames.

By clearing the color and depth buffers at the beginning of each iteration in the while loop, the subsequent rendering operations will start with a blank canvas, ready to display the next frame's content.



```
while (!glfwWindowShouldClose(window)) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

The provided code segment is responsible for setting up an orthographic projection for rendering in the GLFW window. Here's how it works:

First the variables window_width and window_height are declared to store the dimensions of the window. The function glfwGetFramebufferSize is then called to retrieve the size of the window's framebuffer which represents the size of the window in pixels.

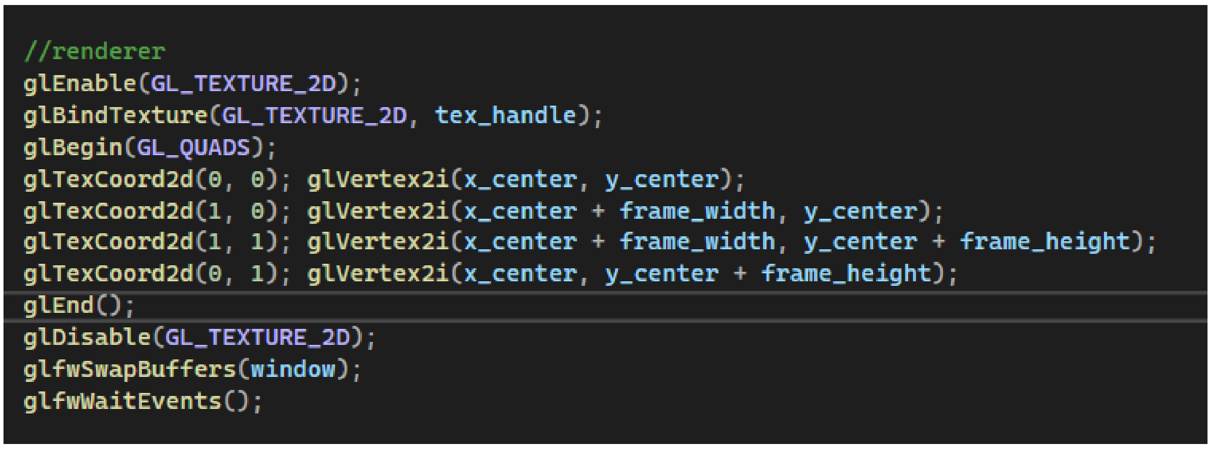
Next, the projection matrix is set as the current matrix using glMatrixMode(GL_PROJECTION). The glLoadIdentity function resets the current matrix to the identity matrix, ensuring a clean starting point for the projection transformation.

The glOrtho function defines an orthographic projection matrix specifying the coordinates of the visible area in the window. The parameters (0 window_width, window_height, 0) define the left right top and bottom boundaries of the visible area, respectively. The near and far parameters (-1 and 1) define the depth range of the visible area.

After setting up the projection matrix, the matrix mode is switched back to GL_MODELVIEW using glMatrixMode(GL_MODELVIEW). This indicates that subsequent matrix transformations will affect the modelview matrix, which is responsible for positioning and orienting objects in the scene.

Lastly, the variables x_center and y_center are calculated to determine the position at which the rendered frame will be centered within the window. These calculations ensure that the frame is aligned properly based on the window and frame dimensions.

Overall this code segment prepares the necessary projection matrix and calculates the center coordinates for rendering the frame within the window, ensuring correct positioning and alignment of the rendered content.

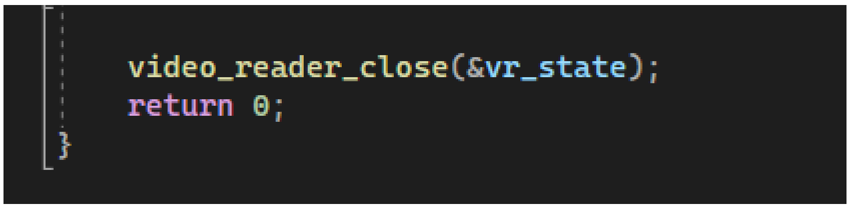


```
//renderer
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, tex_handle);
glBegin(GL_QUADS);
glTexCoord2d(0, 0); glVertex2i(x_center, y_center);
glTexCoord2d(1, 0); glVertex2i(x_center + frame_width, y_center);
glTexCoord2d(1, 1); glVertex2i(x_center + frame_width, y_center + frame_height);
glTexCoord2d(0, 1); glVertex2i(x_center, y_center + frame_height);
glEnd();
glDisable(GL_TEXTURE_2D);
glfwSwapBuffers(window);
glfwWaitEvents();
```

The code below is responsible for closing the video reader and clearing any allocoated resources used.

First, the location of the vr_state variable is sent to the function video_reader_close. This procedure is in charge of freeing up all resources connected to the video reader, including memory and by closing the file. This function is used to correctly close the video reader and release any reserved resources.

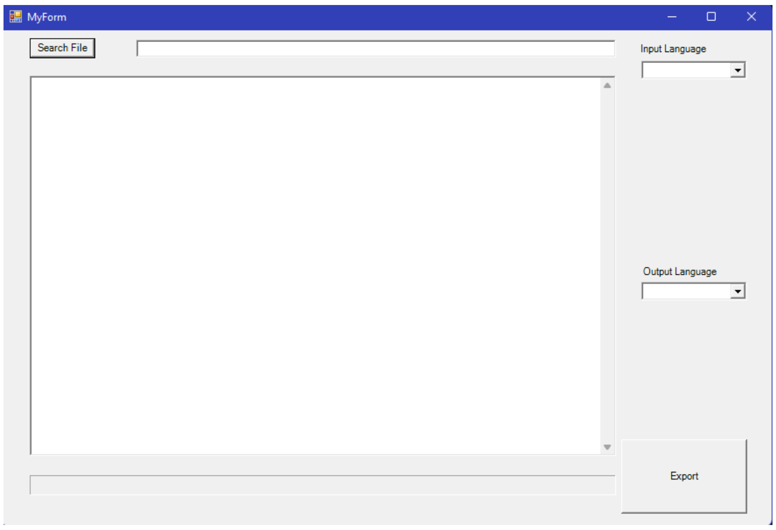
The statement return 0 is encountered after the video reader has been closed. This declaration denotes that the program execution has come to a conclusion and that the program should quit successfully. Returning 0 in this situation normally means that the application ran without any issues and successfully completed.



```
    video_reader_close(&vr_state);
    return 0;
}
```

2.4. Graphical User Interface (GUI)

My GUI has 4 user inputs. The search file button opens the windows explorer and would place the path file on the text bar to its right.



There is a private member variable selectedFilePath, which is declared as a System::String^ to hold the selected file path. It will store the path of the selected video file.

Next, we have the search_btn_Click function which is the event handler for the button click event. This function is executed when the button is clicked. Here's how it works:

The code begins by declaring a character buffer named buffer with a maximum size of MAX_PATH. This buffer will hold the file path selected by the user.

An OPENFILENAME structure named ofn is created and initialized. This structure is used to configure the file dialog options.

The ofn.lpstrFilter field is set to specify the file formats to be searched in the dialog. In this case it allows the selection of video files with extensions .avi .mp4 and .mkv, as well as all files (*.*).

The ofn.lpstrFile field is set to the address of the buffer, and ofn.nMaxFile is set to MAX_PATH, specifying the buffer size and location where the selected file path will be stored.

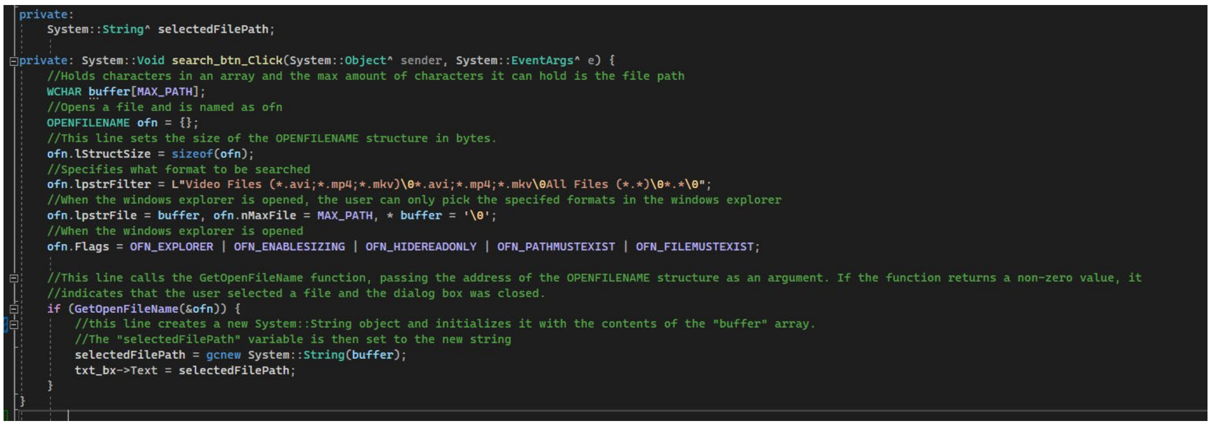
The ofn.Flags field is set to configure the behavior of the file dialog. The specified flags indicate that the dialog should use the explorer-style interface, enable resizing hide the read-only checkbox require a valid path and require an existing file.

The GetOpenFileName function is called passing the address of the ofn structure as an argument. If the function returns a non-zero value it means the user has selected a file and closed the dialog.

Inside the if statement a new System::String object is created initializing it with the contents of the buffer array. This converts the selected file path from a character array to a managed string.

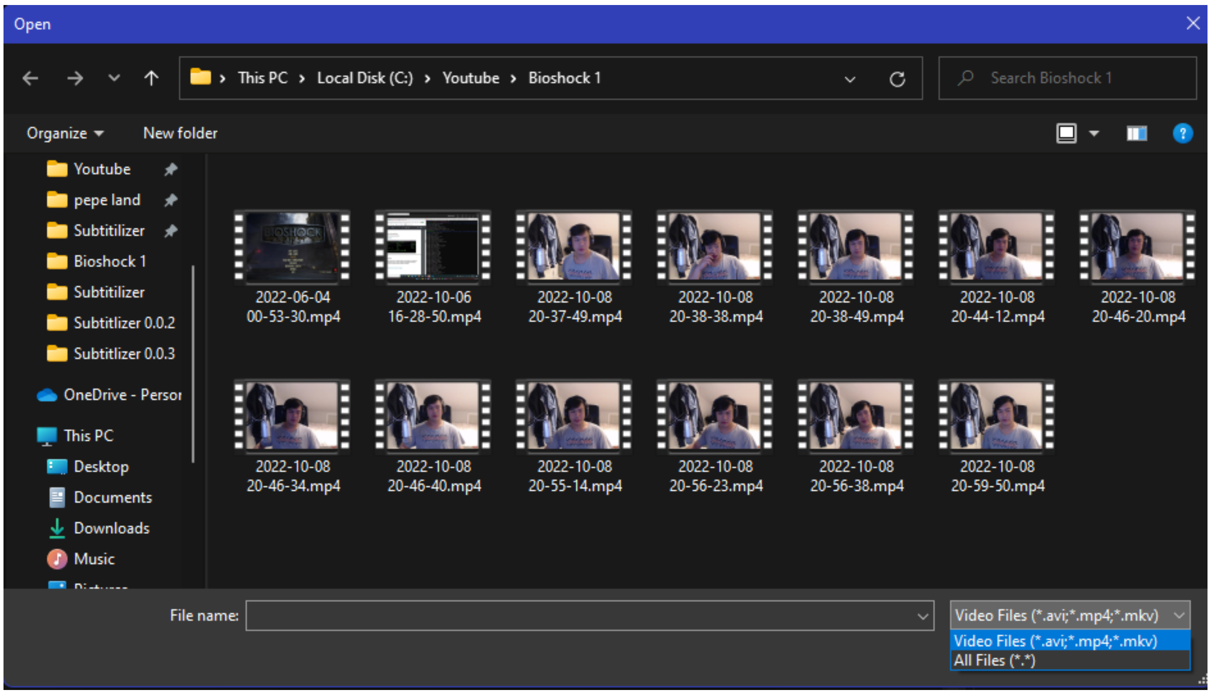
The selectedFilePath variable is assigned the value of the newly created System::String object storing the selected file path.

Finally the txt_bx control's Text property is set to selectedFilePath updating a text box in the user interface with the selected file path.



```
private:
    System::String^ selectedFilePath;

private: System::Void search_btn_Click(System::Object^ sender, System::EventArgs^ e) {
    //Holds characters in an array and the max amount of characters it can hold is the file path
    WCHAR buffer[MAX_PATH];
    //Opens a file and is named as ofn
    OPENFILENAME ofn = {};
    //This line sets the size of the OPENFILENAME structure in bytes.
    ofn.lStructSize = sizeof(ofn);
    //Specifies what format to be searched
    ofn.lpstrFilter = L"Video Files (*.avi;*.mp4;*.mkv)\0*.avi;*.mp4;*.mkv\0All Files (*.*)\0*.*\0";
    //When the windows explorer is opened, the user can only pick the specifed formats in the windows explorer
    ofn.lpstrFile = buffer, ofn.nMaxFile = MAX_PATH, * buffer = '\0';
    //When the windows explorer is opened
    ofn.Flags = OFN_EXPLORER | OFN_ENABLESIZING | OFN_HIDEREADONLY | OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

    //This line calls the GetOpenFileName function, passing the address of the OPENFILENAME structure as an argument. If the function returns a non-zero value, it
    //indicates that the user selected a file and the dialog box was closed.
    if (GetOpenFileName(&ofn)) {
        //this line creates a new System::String object and initializes it with the contents of the "buffer" array.
        //The "selectedFilePath" variable is then set to the new string
        selectedFilePath = gcnew System::String(buffer);
        txt_bx->Text = selectedFilePath;
    }
}
```

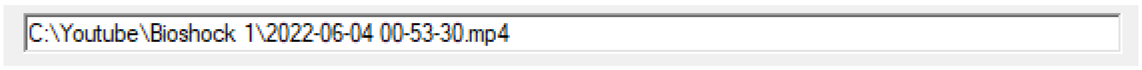
The images below shows what happens when the User presses the search file. The windows explorer filters only video files and all files. This is incomplete as I want to give an error when the User selects a file that is not a video file.



This PC › Local Disk (C:) › Youtube › Bioshock 1

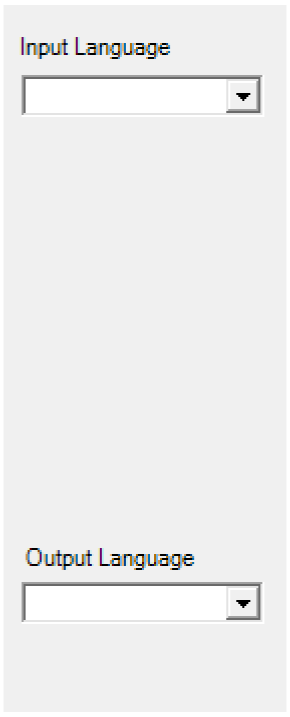
2022-06-04 00-53-30.mp4
2022-10-06 16-28-50.mp4
2022-10-08 20-37-49.mp4
2022-10-08 20-38-38.mp4
2022-10-08 20-38-49.mp4
2022-10-08 20-44-12.mp4
2022-10-08 20-46-20.mp4
2022-10-08 20-46-34.mp4
2022-10-08 20-46-40.mp4
2022-10-08 20-55-14.mp4
2022-10-08 20-56-23.mp4
2022-10-08 20-56-38.mp4
2022-10-08 20-59-50.mp4

File name:
Video Files (*.avi;*.mp4;*.mkv)
Video Files (*.avi;*.mp4;*.mkv)
All Files (*.*)

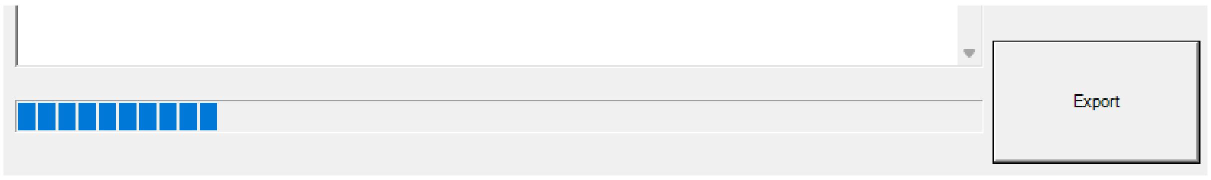


C:\Youtube\Bioshock 1\2022-06-04 00-53-30.mp4

Once the user selects a video the path file will automatically place it in the text bar.



The Input and Output Language are currently empty right now as I want to fill the boxes with Google's API. The options will be filled with the languages available with Google Translate.

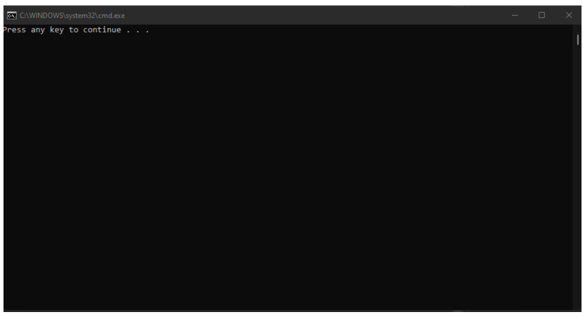


The export button will then fill up the progress bar to show the user how much time is left. For this prototype I am replicating what it would look like if a video was being processed. In the final version the speed of the progress bar will depend on the time it is processing the video.

2.5. Testing

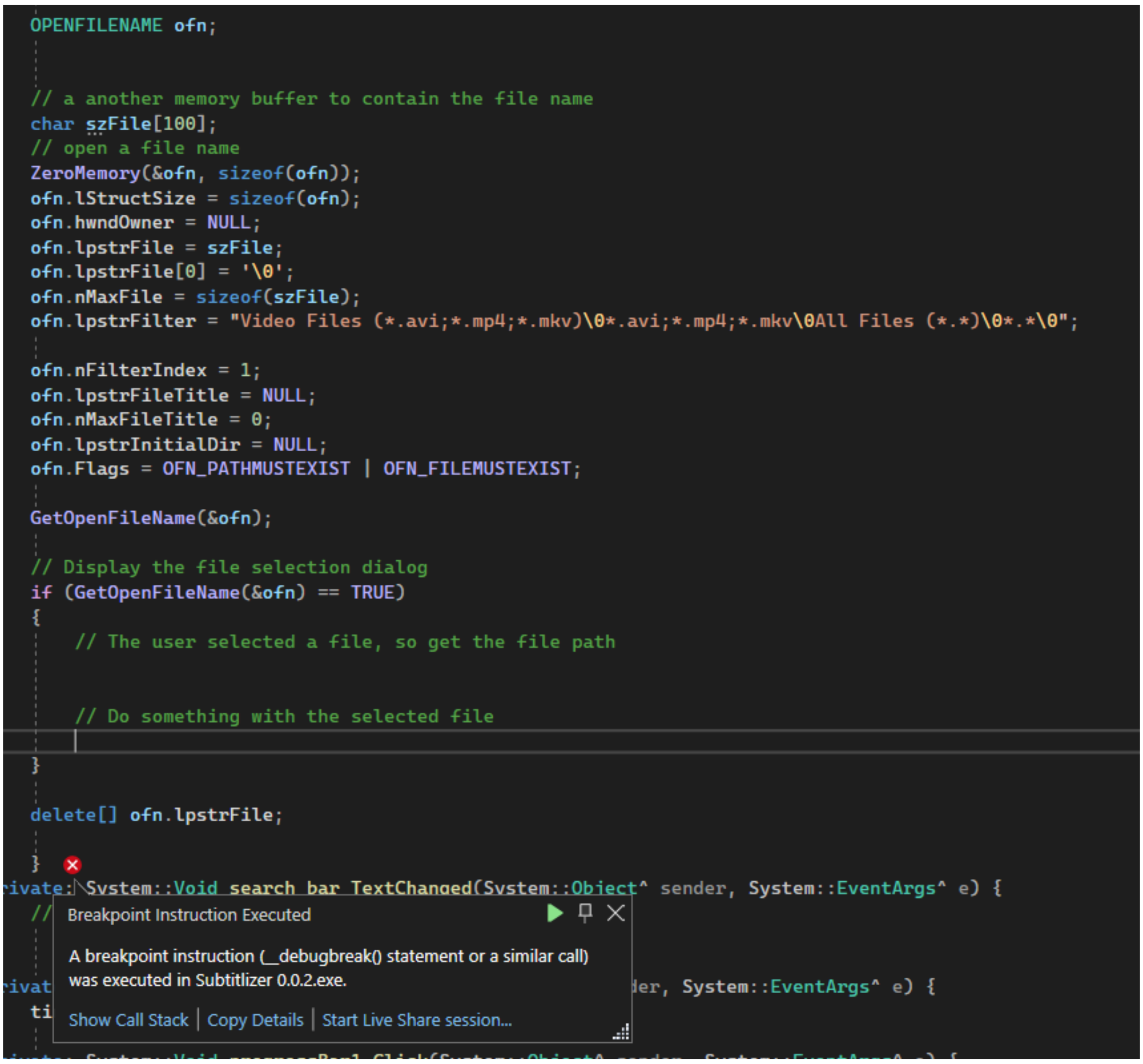
All my testing was done in Visual studio in debug mode using win64 architecture.

At first I wanted to get info but nothing was happening, to fix this, I had to link my libraries and my .dll files. What would happen is I would just get a blank screen on the console.



```
C:\WINDOWS\system32\cmd.exe
Press any key to continue . . .
```

When testing for my search function, there was multiple errors. Looking at the error, I was having memory leakage. I was getting a breakpoint instruction executed. I finally got to fix it after changing the structure of my array.



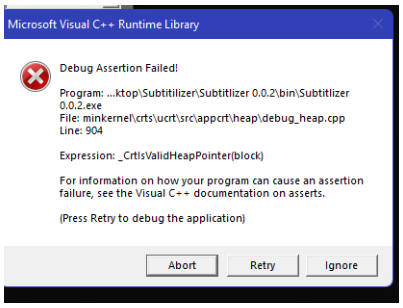
```
OPENFILENAME ofn;

// a another memory buffer to contain the file name
char szFile[100];
// open a file name
ZeroMemory(&ofn, sizeof(ofn));
ofn.lStructSize = sizeof(ofn);
ofn.hwndOwner = NULL;
ofn.lpstrFile = szFile;
ofn.lpstrFile[0] = '\0';
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFilter = "Video Files (*.avi;*.mp4;*.mkv)\0*.avi;*.mp4;*.mkv\0All Files (*.*)\0*.*\0";

ofn.nFilterIndex = 1;
ofn.lpstrFileTitle = NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = NULL;
ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

GetOpenFileName(&ofn);

// Display the file selection dialog
if (GetOpenFileName(&ofn) == TRUE)
{
    // The user selected a file, so get the file path


    // Do something with the selected file

}

delete[] ofn.lpstrFile;

}
ivate: System::Void search_bar_TextChanged(System::Object^ sender, System::EventArgs^ e) {
//
ivat                                                     der, System::EventArgs^ e) {
ti
```

Breakpoint Instruction Executed

A breakpoint instruction (__debugbreak() statement or a similar call) was executed in Subtitlizer 0.0.2.exe.

Show Call Stack | Copy Details | Start Live Share session...

Another bug would happen if I did not run through debugger. I would select a file and the program would ask me to select a file. After the second selection, the program would crash as shown below.



Microsoft Visual C++ Runtime Library

Debug Assertion Failed!

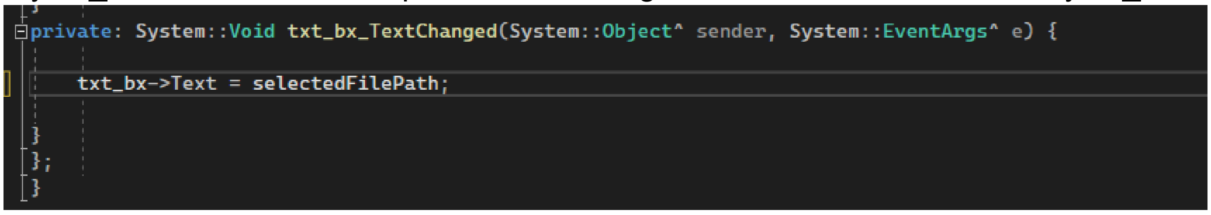
Program: ...ktop\Subtitilizer\Subtitlizer 0.0.2\bin\Subtitlizer 0.0.2.exe
File: minkernel\crts\ucrt\src\appcrt\heap\debug_heap.cpp
Line: 904

Expression: _CrtIsValidHeapPointer(block)

For information on how your program can cause an assertion failure, see the Visual C++ documentation on asserts.

(Press Retry to debug the application)

After having my search function working, my next plan was placing the file path to my txt_bx. At first I would implement the string into the event handler for my txt_bx.



```
private: System::Void txt_bx_TextChanged(System::Object^ sender, System::EventArgs^ e) {

    txt_bx->Text = selectedFilePath;

}
};
}
```

By doing it this way the file path would not show up until a user has done an action on the txt_bx. After researching online the reason why this happens is because having a text box itself is having an event. For there to be an instant update I decided to imaplament txt_bx->Text = selectedFilePath; in my search button as this seemed the most logical way since we instruct the code to place the file path directly to the text box after selecting a file.

I used Native Unit Test template from Visual Studio because it gives me an easy way to test my methods. The tests I would apply are unit testing, integration and acceptance.

In Scope: The requirements in my document were tested.

Out of scope: The features that were not included are:

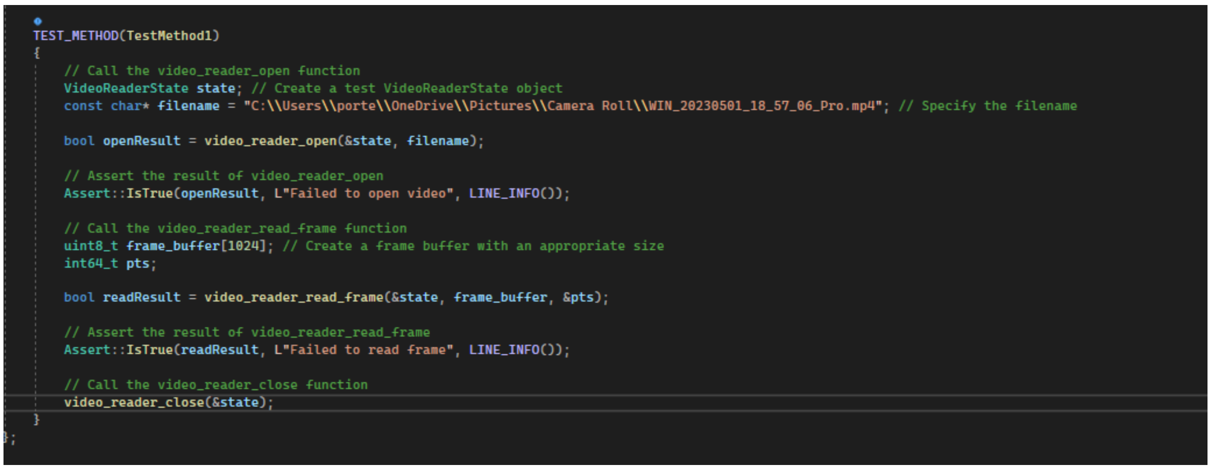
1. Computer hardware
2. Advanced video effects
3. Codec development

I want to test my methods if they were functional to its fullest capabilities. My project focuses on testing the intake of the video to be able to output and outputting.

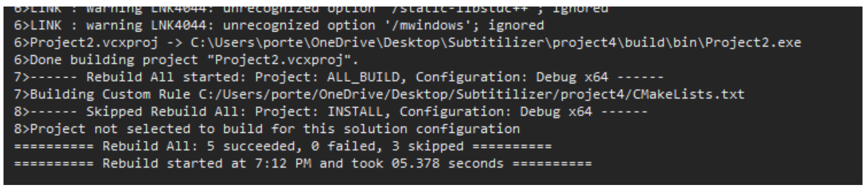
Test Case Type	Description	Use Case	Expected results	Final Result
File Input	Can select any video file	select a video in most common formats	Any video type is accepted	Pass
Language input	Can choose any language as input	Select any language	Any language is used as an input	Inconclusive
Language Output	Can choose any language as output	Select any language	Any language is used as an output	Inconclusive
Video	File is exported	File can be opened	File can be opened	Inconclusive
Error output	Error shows when a problem occurs	Problem is described	Problem is shown to user	Pass

Unit Testing

Unit testing is a crucial testing strategy that tries to confirm the operation of certain software system components, such as classes or methods. Unit testing frameworks in the context of Visual Studio offer tools for automating these tests and verifying the intended functionality of each component. A specific component that builds successfully and satisfies the requirements is the result of a successful unit test.



```
    TEST_METHOD(TestMethod1)
    {
        // Call the video_reader_open function
        VideoReaderState state; // Create a test VideoReaderState object
        const char* filename = "C:\\Users\\porte\\OneDrive\\Pictures\\Camera Roll\\WIN_20230501_18_57_06_Pro.mp4"; // Specify the filename

        bool openResult = video_reader_open(&state, filename);

        // Assert the result of video_reader_open
        Assert::IsTrue(openResult, L"Failed to open video", LINE_INFO());

        // Call the video_reader_read_frame function
        uint8_t frame_buffer[1024]; // Create a frame buffer with an appropriate size
        int64_t pts;

        bool readResult = video_reader_read_frame(&state, frame_buffer, &pts);

        // Assert the result of video_reader_read_frame
        Assert::IsTrue(readResult, L"Failed to read frame", LINE_INFO());

        // Call the video_reader_close function
        video_reader_close(&state);
    }
};
```



```
6>LINK : warning LNK4044: unrecognized option '/static-libstdc++'; ignored
6>LINK : warning LNK4044: unrecognized option '/mwindows'; ignored
6>Project2.vcxproj -> C:\Users\porte\OneDrive\Desktop\Subtitilizer\project4\build\bin\Project2.exe
6>Done building project "Project2.vcxproj".
7>------ Rebuild All started: Project: ALL_BUILD, Configuration: Debug x64 ------
7>Building Custom Rule C:/Users/porte/OneDrive/Desktop/Subtitilizer/project4/CMakeLists.txt
8>------ Skipped Rebuild All: Project: INSTALL, Configuration: Debug x64 ------
8>Project not selected to build for this solution configuration
========== Rebuild All: 5 succeeded, 0 failed, 3 skipped ==========
========== Rebuild started at 7:12 PM and took 05.378 seconds ==========
```

Integration Testing

Integration testing involves testing the interactions between different components or modules of my application to ensure they work together correctly. I dentified my integration points to know where different components would interatct with each other.

End User testing

To test my application, I used different types of containers that would use different types of encoders. I have used files such as .mov mp4 and avi. I tested this manually as there is no way to automate taking multiple videos. Having this testing would help me identify if there were any problems with usability.

3.0 Conclusion

The project seeks to give a simple yet efficient method of enhancing video accessibility by delivering a straightforward option for translating films and overlaying subtitles. Automatic transcription and translation of video footage are made possible by utilizing the strength of Google's voice recognition API making it available to a larger audience. This project has significant promise for educational uses enabling teachers to provide multilingual content and connect with a broad student population in addition to being a helpful tool for personal video translation.

It's important to remember that the project's functioning depends on Google's voice recognition API. Due to this reliance on outside services, continued use of the API may need payment or subscription plans which may impact the project's long-term viability. Additionally the voice recognition system's performance and accuracy depend on the capabilities and restrictions of Google's API.

Despite these limitations the project provides a useful method for creating subtitles and translating videos. It makes adding subtitles to videos easier with automated procedures enabling accessibility and inclusion. It creates new opportunities for cross-cultural contact and information exchange by utilizing contemporary technology.

4.0 Further Development or Research

Given with additional time there are many ways I would explore to improve the functionality and usability of my current project.

Adding new features that improve the capabilities of the program and provide users more value would be my main emphasis. This can entail connecting to external services or APIs putting in place sophisticated algorithms or data analysis methods or adding interactive visualizations.

User Experience Refinement: I would devote effort to enhancing the application's user interface (UI) and user experience (UX). This could entail rethinking the layout streamlining processes to make them clearer and more effective and doing usability tests to gather input and iteratively improve the design.

Performance Optimization: To improve the application's speed and responsiveness I would examine the performance bottlenecks and optimize the code and algorithms. This can entail analyzing the program, locating problem areas, and putting performance-improving strategies in place like caching, parallelization, or code rewriting.

Compatibility and Platform Support: I would like for most people to be able to use the software. I would like to expand this software to operating systems like most linux operating systems such as ubuntu and other versions of macOS.

Additionally, I would actively seek feedback from internet users to promote a broader software adoption. Driving ongoing development requires gathering opinions and information from a wide range of users. I would utilize a variety of techniques, including user testing sessions, feedback forms analytics analysis and user surveys to do this. I can continually improve the usability of the program solve any problems and make sure that it satisfies the changing demands of its users by incorporating actual user feedback into the development process.

Appendices

Objectives

The core feature of the software will be able to take any video and be able to place subtitles on top of the video. The user will be asked what language the video is spoken in then asked what language they would like the subtitles to be in. The user is asked for the language because some languages sound the same depending on who is speaking. This is also done to make sure the accuracy of the transcript is at its highest. The transcript will then be made into a subtilties file so the program will be able to read and place the words on top of the video.

The user will have an UI where they can change the settings of the languages. At the minimum the software will render the video in its original sizing of width and height. The user will be able to change the font, font size, colour of the font and the font

Background

I will be using Google's API where they are able to take any audio and transcribe them. Afterwards they will be put through Google's translate. After they have been translated, the software will make sure the subtitles come at the same time when a person is speaking and then end after the person speaking is done. It will repeat this process until the video is done.

I am also using an open-source software that renders the videos called FFmpeg. A free open-source software project called FFmpeg is used to stream, record, and convert audio and video. Along with a command-line tool for transcoding media files, it also comes with a collection of libraries for handling a variety of media types. FFmpeg is frequently used for many different things such as: converting music and video between different formats

I chose this project because I watch anime from different websites and only saw most websites transcribing and placing subtitles in videos usually in only in English. I thought that other users might want to be able to read in other languages such as French, Spanish, Mandarin, etc. This would also help me if there is a tutorial online that is set in a different language. There are plenty of cases when I was looking for video tutorials online on a particular subject there would be times when I could not find a video in English but they would be taught in other languages like Spanish Hindi and Russian.

State of the Art

When thinking of this idea no company has done this before. There is currently no application on the internet/market that does this feature. Youtube is the closest competitor I have but even they have restrictions on their subtitle files. Youtube will only allow subtitles to be automatically placed if the language is supported and the video is not long.

Data

The data that will be required is Google's API audio to transcription. I will be linking that to the software so that a transcription will be made. The audio will be listened to by Google's API then using google translate, another API, the user will be able to choose the language that is offered by Google.

The video that will be used for transcription can be pulled online or the user will be able to transcribe any video that is fed through the software.

The software will use the user's CPU to compute and render the video and subtitles. Having a stronger computer will render videos faster.

Methodology & Analysis

As this project is for personal use as of writing this proposal, I will use the waterfall method. If I choose to publish this software, I will then swap to the agile methodology as I will require users to pay for this service. I will look at other software that renders video and how they work so I will be able to render the video in its best quality.

My project will be broken down into tasks that would be completed in week deadlines and every month will be the milestones as I expect to have improved the software significantly as there is very little time for the project. I will be making a planner and how much time would be needed to successfully make the project. I will also give myself some leeway as problems can arise. I will be making sure the core functionalities will be included and then add additional functionalities at the end if I have the time. An example of this could be rendering the video in a lower quality to fasten the process of rendering.

Technical Details

The technical side of this project would be the rendering of the video and the synchronisation of audio and subtitles. I will have to make an algorithm that repeats of taking the transcription matching the show time of the substiles until the video has ended.

I will have to learn how to incorporate the audio to transcription and translation API. I would need to read up on the documents on how to use them effectively in the program.

I would also need to learn how to make SRT files the file that is used to put subtitles on top of videos.

I would also need to make a user interface that is easy for the user to read and use.

The closest example of replication would be Youtube's CC functionality feature with their videos.

Project Plan

January:

Week 1: Familiarize myself with the FFmpeg codebase and build system set up development environment, begin work on adding subtitles feature

Week 2: Continue work on adding subtitles feature

Week 3: Continue work on adding subtitles feature, start researching and learning about Google API for audio transcription and translation

Week 4: Continue work on adding subtitles feature, continue learning about Google API

February:

Week 1: Finish implementing the subtitles feature, start integrating the Google API for audio transcription and translation

Week 2: Continue integrating the Google API for audio transcription and translation

Week 3: Test and debug the audio transcription and translation functionality

Week 4: Continue testing and debugging the audio transcription and translation functionality

March:

Week 1: Continue testing and debugging the audio transcription and translation functionality

Week 2: Begin work on rendering the video with added subtitles and transcription/translation functionality

Week 3: Test and debug the rendering feature

Week 4: Continue testing and debugging the rendering feature, finalize project

This revised plan includes a total of 192 hours of work over the course of 3 months with 8 hours of work per day and 1 day off per week.

Testing

The test I will be doing is making a video of myself and put it through the software. I would test for the transcription file being made so I can compare what I said compared to the file it has been made.

The second test I will make is to see the video format it has produced and see if the video is the same as it was rendered through the program.

The third test I will do will be making sure the program is able to take the transcribed file and seeing if subtitles match according to the video.

The tests following the key tests I have mentioned will be refining the software to make sure it will run as smoothly as possible without and any problems.

Github repo:

https://github.com/LloydPortes2/FYP-Subtitilizer

Project proposal

National College of Ireland

Project Proposal

BSHC in Computing project

21/10/2022

Subtitlizer

Software Development

2022

Lloyd Portes

19509219

X19509219@student.ncirl.ie

Contents

1.0 Objectives 47
2.0 Background 47
3.0 State of the Art 47
4.0 Data 47
5.0 Methodology & Analysis 48
6.0 Technical Details 48
7.0 Project Plan 48

3.0 Objectives

The core feature of the software will be able to take any video and be able to place subtitles on top of the video. The user will be asked what language the video is spoken in then asked what language they would like the subtitles to be in. The user is asked for the language because some languages sound the same depending on who is speaking. This is also done to make sure the accuracy of the transcript is at its highest. The transcript will then be made into a subtilties file so the program will be able to read and place the words on top of the video.

The user will have an UI where they can change the settings of the languages. At the minimum the software will render the video in it's original sizing of width and height. The user will be able to change the font font size colour of the font and the font

4.0 Background

I will be using Google's API where they are able to take any audio and transcribe them. Afterwards they will be put through Google's translate. After they have been translated the software will make sure the subtitles come at the same time when a person is speaking and then end after the person speaking is done. It will repeat this process until the video is done.

I am also using an open-source software that renders the videos

I chose this project because I watch anime from different websites and only saw most websites transcribing and placing subtitles in videos usually in only in English. I thought that other users might want to be able to read in other languages such as French Spanish, Mandarin, etc.

5.0 State of the Art

When thinking of this idea, no company has done this before, until recently where Microsoft has released very similar functionality, where you place a video into the program and it would transcribe for you. There are also a couple of programs where they do the same thing as Microsoft. The difference that I will be doing is that I will be able to download directly from the website using the software and be able to transcribe, change the language of the transcription then place the subtitles into the video.

6.0 Data

The data that will be required is Google's api audio to transcription. I will be linking that to the software so that a transcription will be made. The audio will be listened to by Google's api, then using google translate, another api, the user will be able to choose the language that is offered by Google.

The video that will be used for transcription can be pulled online or the user will be able to transcribe any video that is fed through the software.

The software will use the user's cpu to compute and render the video and subtitles. Having a stronger computer will render videos faster.

7.0 Methodology & Analysis

As this project is for personal use as of writing this proposal, I will use the waterfall method. If I choose to publish this software, I will then swap to the agile methodology as I will require users to pay for this service. I will look at other software that renders video and how they work so I will be able to render the video in its best quality.

My project will be broken down into tasks that would be completed in week deadlines and every month will be the milestones as I expect to have improved the software significantly as there is very little time for the project. I will be making a planner and how much time would be needed to successfully make the project. I will also give myself some leeway as problems can arise. I will be making sure the core functionalities will be included and then add additional functionalities at the end if I have the time. An example of this could be rendering the video in a lower quality to fasten the process of rendering.

8.0 Technical Details

The technical side of this project would be the rendering of the video and the synchronisation of audio and subtitles. I will have to make an algorithm that repeats of taking the transcription, matching the show time of the substiles until the video has ended.

I will have to learn how to incorporate the audio to transcription and translation api. I would need to read up on the documents on how to use them effectively in the program.

I would also need to learn how to make SRT files, the file that is used to put subtitles on top of videos.

I would also need to make a user interface that is easy for the user to read and use.

The closest example of replication would be Youtube's CC functionality feature with their videos.

9.0 Project Plan

The first thing I would do is understand the APIs that I would be using and how they work and what their capabilities and limitations are. I would be giving myself a week

The next plan is giving myself two weeks to create just a UI without any functionality on FFMpeg. I would first create a wire diagram on how the software will look like. Then try to make.

I would then give myself a month on to implament. the basic functionality of adding the libraires to have the program working.

Then the API implementation from google where it will take the audio then transcribe it. I will make google translate the file to what the user wanted. Then make sure it is a file that can be read by FFMpeg.

Another month on how to have that file be synced to the video.

Then give myself a month to make sure the project is thoroughly done and make sure nothing is missing, this also gives me leeway if something from the project took longer then I anticipated to complete it.

10.0 Testing

Describe how you will evaluate the system with real technical data using system tests, integration tests etc. If applicable describe how you will evaluate the system with an end user. (be careful here re Ethics etc)

The test I will be doing is making a video of myself and put it through the software. I would test for the transcription file being made so I can compare what I said compared to the file it has been made.

The second test I will make is to see the video format it has produced and see if the video is the same as it was rendered through the program.

The third test I will do will be making sure the program is able to take the transcribed file and seeing if subtitles match according to the video.

The tests following the key tests I have mentioned will be refining the software to make sure it will run as smoothly as possible without and any problems.

Reflective Journal

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: October

What? Reflect on what has happened in your project this month? I looked at my project comment for my video project proposal. It was said that I might be doing too much by making into an extension. I have concluded that I would need to do it as a software because I feel that the cpu would not be used properly.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? I found an open source library for rendering software. I would create an UI so it would be able to utilize the important features such as rendering and implementing. This would force to use the cpu for rendering. Extensions only use ram and not cpu usage.
Now What? What can you do to address outstanding challenges? By creating a UI and using it's libraries, it would be easier to create the my project.

Student Signature	Lloyd Portes

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: November

What? Reflect on what has happened in your project this month? I have talked to Lisa my supervisor about my project on the 29th of November. I have asked what she thought of my thought process and how I want to handle it.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? I will continue with my plan of making a software but not deploy it as an extension for google chrome. I still need to create the GUI and learn how the software command functions to implement into the GUI alongside the API needed for my core features.
Now What? What can you do to address outstanding challenges? Reading the documentation available online made by google for the API and

Student Signature	Lloyd Portes

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: December

What? Reflect on what has happened in your project this month? I now have a GUI for how I want my software to work. The user can now select a video file from their windows explorer. The chosen file will now display the file location on to a text bar.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? This is a small stepping stone for me as I know have a layout. Having the user be allowed to select a file took more time than I was anticipating but have completed none the less. I now need to render a video, then attach Google's APIs
Now What?

What can you do to address outstanding challenges? I will now need to read documentation for how FFmpeg can render a video and how to call Google's API and attach it to my code.	
Student Signature	Lloyd Portes

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: Janurary

What? Reflect on what has happened in your project this month? I am currently working on rendering the video. As of this month, I'm trying to get frames from the video to output into a new video of itself. I have also installed openGL into the software so I am able to render the video	
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? I feel I am a quarter of the way to getting videos rendered. Was not a complete success but still working on making sure video and audio streams are separated to be manipulated.	
Now What? What can you do to address outstanding challenges? Need to read more documentation on how to solve these problems.	
Student Signature	Lloyd Portes

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: Feburary

What? Reflect on what has happened in your project this month? I have made tests so that I am able to see if my packets are being rendered, currently they are not.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? The videos are getting errors with the stream of my videos. My program is outputting that the video streams are giving an invalid NAL unit size.
Now What? What can you do to address outstanding challenges? I need to debug and read more documentation

Student Signature	Lloyd Portes

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: March

What? Reflect on what has happened in your project this month? I have not done anything due to other assignments
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? I have not done anything due to other assignments
Now What? What can you do to address outstanding challenges? I have not done anything due to other assignments

Student Signature	Lloyd Portes

Supervision & Reflection Template

Student Name	Lloyd Portes
Student Number	X19509219
Course	BSHCSD4
Supervisor	Lisa Murhpy

Month: April

What? Reflect on what has happened in your project this month? I am behind and trying to get my project done	
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? I am able to output one static image of colour. Everything else still remains.	
Now What? What can you do to address outstanding challenges? Do more work.	
Student Signature	Lloyd Portes