

# C# WPF Video Player

Zachary Rose, Logan Walsh

## Abstract

**Our project is a custom video player as a C# WPF application. It will allow you to browse to a file, load up a video, and control the playback as needed. AVI, MWV, and MP4 will be supported file types. Those who want more control over their video playback beyond what Windows comes with will find this application useful.**

## 1. Introduction

For this project, we plan to make a media player program that offers some features that are unavailable in Windows Media Player. It will include a file browser to select a media file, and it will include the controls for playback such as: play/pause, fast forward/slow down, and skip to different parts of the video, as well as volume controls.

This project is intended for those who want some finer control over the playback of their video files. Windows Media Player does have the basic features for controlling playback of media, but it leaves much to be desired in customization and more niche functionality.

Windows Media Player lacks features such as moving forward and backwards a frame at a time, enlarging the video at a key press, and muting the video with a key press. In fact, Windows Media Player lacks keyboard controls entirely. VLC Media Player has these features and more, but it can sometimes feel daunting and over bloated for an average use case. This media player will fit comfortably in the middle between the most casual of users and super users.

### 1.1. Background

Some useful terminology:

- media: in this case, a file containing video and/or sound
- media player: a program that can playback different media, and it usually allows for some control over different aspects of the playback
- seeker: something that enables a user to skip to an arbitrary part of a media's duration with a single click, often in the form of a horizontal bar
- playback: media is in playback when it is currently being shown to the user
- file browser: a user interface to make selecting a file from the system convenient
- Windows Media Player: a media player that comes with Windows OS by default. Very bare-bones
- Fullscreen: refers to when the video is scaled up to cover an entire monitor

We decided to make a media player because we thought it would be interesting to make a tool that we might actually use, rather than some project for the sake of displaying some unusual programming concept. It will be good practice towards developing a user interface for a user-oriented application.

### 1.2. Impacts

Our hope is that this project will become a useful little tool for those who wish to have a light-weight and customizable media player.

### 1.3. Challenges

By far the hardest thing will likely be designing an interface to dynamically change key bindings. It will need a file to store them, the ability to parse and edit said file, and an interface with which the user will change the bindings from within the app.

## 2. Scope

Basic goals for getting started: You can load up a video, play and pause it using buttons, and it can be replayed.

The project will be done once the program runs without any issues and has enough features to differentiate it from Windows Media Player. In other words, the ability to control the video playback with the keyboard and to create custom keybindings.

We think that this is a worthwhile project because it has practical use, and it will allow us to make further use of the WPF framework. It will require an attractive interface and different form elements working together behind the scenes.

Basic Goals:

- Can load a video file to prepare for playback.
- Can control playback through both mouse and keyboard.
- Supports AVI, MKV, and MP4 filetypes.

Stretch Goals:

- Permanent customization for key bindings, with the ability to reset to defaults.
- Combining multiple files into a playlist with associated skip forward/backwards buttons.
- WPF window can be resized, with all the elements keeping the same size ratio.

## 2.1. Requirements

The functional requirements for a media player are straight forward for anyone who has used one before. We considered what we want to be able to do when watching a video, as well as what is absolutely necessary to do so. Loading a file is the absolute base requirement, but we've come to expect things like pausing and fast forwarding.

### 2.1.1. Functional.

- User can play video files
- User can use video timeline to skip to desired part of video directly
- Application supports fullscreen playback
- User can pause and unpause the playback at will
- Application supports different playback speeds, between .5x and 2x
- User can change the volume as desired

### 2.1.2. Non-Functional.

- Application must be able to edit and reload keybindings between different sessions.
- Application must be able to load and play several videos sequentially without performance loss.

## 2.2. Use Cases

Use Case Number: 1

Use Case Name: Loading up a video

Description: The user wants to load a video from their hard drive to prepare for playback.

- 1) User opens up file browsing menu by left-clicking "File," followed by "Load File" (See Figure 1).
- 2) User navigates to desired video and selects it.

Termination Outcome: Video is loaded and begins playback.

Alternative: User wishes to load a video using a file path

- 1) User enters file path into search bar (See Figure 1).
- 2) User left-clicks "OK" button.

Termination Outcome: Video is paused and prepared for playback.

Note: File selected doesn't exist.

- 1) User selects a file that doesn't exist, doesn't have permissions to play, or isn't a valid video format.

Termination Outcome: Application requests a valid file and allows for re-entry.

Use Case Number: 2

Use Case Name: Editing keybindings

Description: The user wants to change the default keyboard bindings for different aspects of playback control and expects these changes to persist through different sessions of using the application.

- 1) User opens up the options by left-clicking the "Options" tab with the mouse.
- 2) User selects "Hotkeys" by left-clicking with the mouse.
- 3) User selects the binding they want to change, and presses the desired keyboard key to change it to.

Termination Outcome: New keybindings are set for any available functionality, and the new bindings will be saved to a file for automatic loading upon opening the application.

Use Case Number: 3

Use Case Name: Controlling playback

Description: During playback, the user wants to control different aspects of the playback. See Figure 2 for the buttons available to left-click.

- 1) User pauses and unpauses playback by left-clicking the pause button.
- 2) User increases and decreases playback speed using the corresponding buttons.
- 3) Alternative: User enters a playback speed directly into the text box and presses the enter key.
- 4) User increases and decreases volume by dragging the volume slider using the mouse.
- 5) User enters fullscreen mode by left-clicking the fullscreen button.

Termination Outcome: Video is in playback in the way desired by the user.

Alternative: User wishes to use keybindings to control video playback

- 1) User pauses and unpauses playback by pressing the bound buttons on the keyboard.
- 2) User increases and decreases playback speed by pressing the bound buttons on the keyboard.
- 3) User increases and decreases volume by pressing the bound buttons on the keyboard.
- 4) User enters fullscreen mode by pressing the bound button on the keyboard.

Termination Outcome: Video is in playback in the way desired by the user.

### 2.3. Interface Mockups

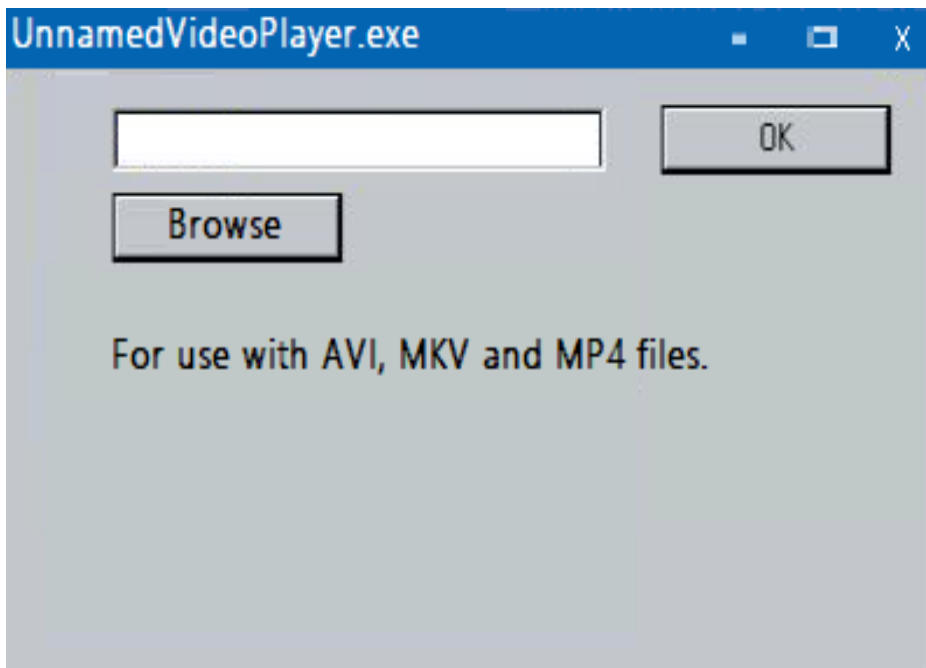


Figure 1. Sample of interface for selecting video file

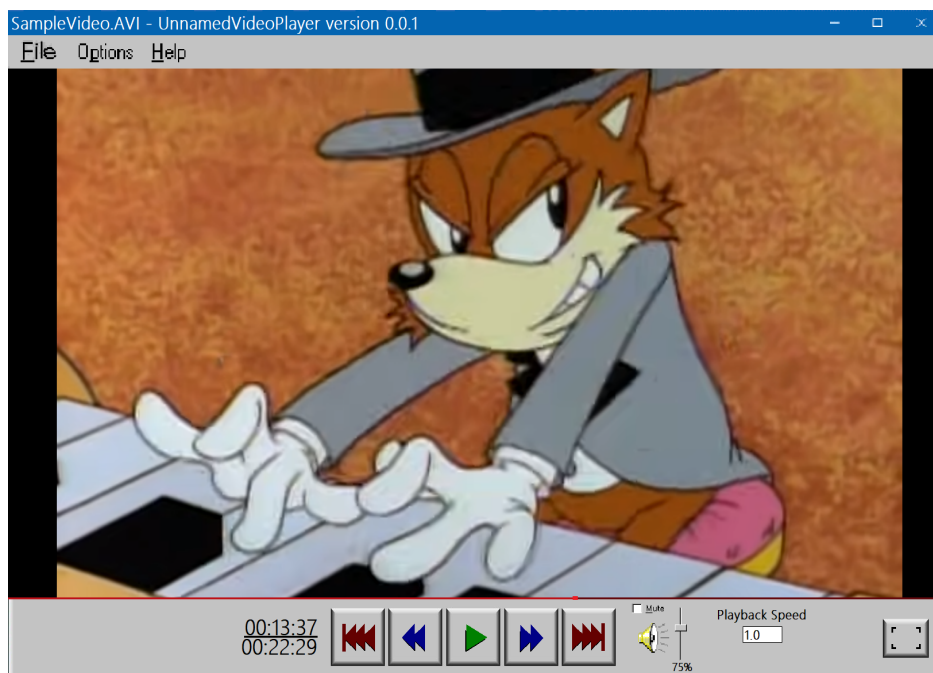


Figure 2. Sample of interface when a video is in playback

### 3. Project Timeline

See Figure 3.

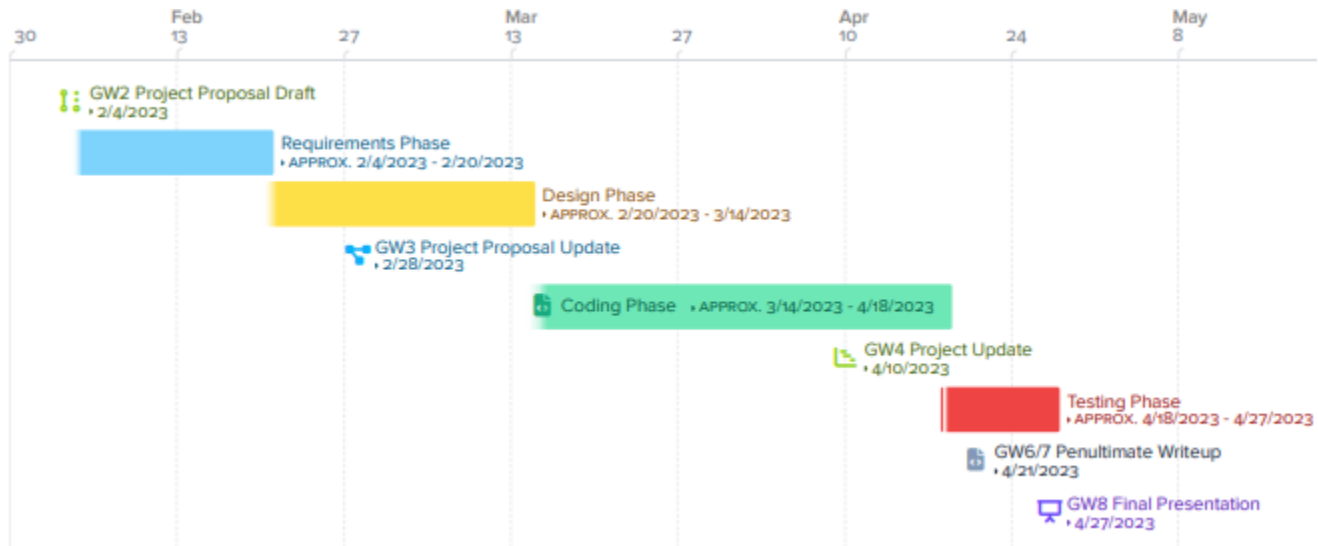


Figure 3. Proposed project timeline

### 4. Project Structure

Our main class contains several important features that work together to update the Media Element. A seeker is updated on a timer to remain updated with video playback, and in the same way the video playback is updated when the seeker is moved. A timer is also used to implement playing in reverse; every tick of the timer pushes the media backwards by an amount with respect to the video playback speed.

A Factory class is used to change the appearance of the application during runtime. This was chosen in order to simplify adding more themes by keeping them unbound from the main class.

#### 4.1. UML Outline

See Figure 4.

The MainWindow class holds the WPF form. This form holds the button controls, and most importantly it holds the MediaElement object which is used to enable playback. Two DispatcherTimers are also used to keep the seeker updated with the video playback.

The ThemeFactory is an abstract factory interface used to change the theme of the media player. When choosing a theme a new factory is constructed, and it is used to change the appearance of certain elements.

#### 4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

### 5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

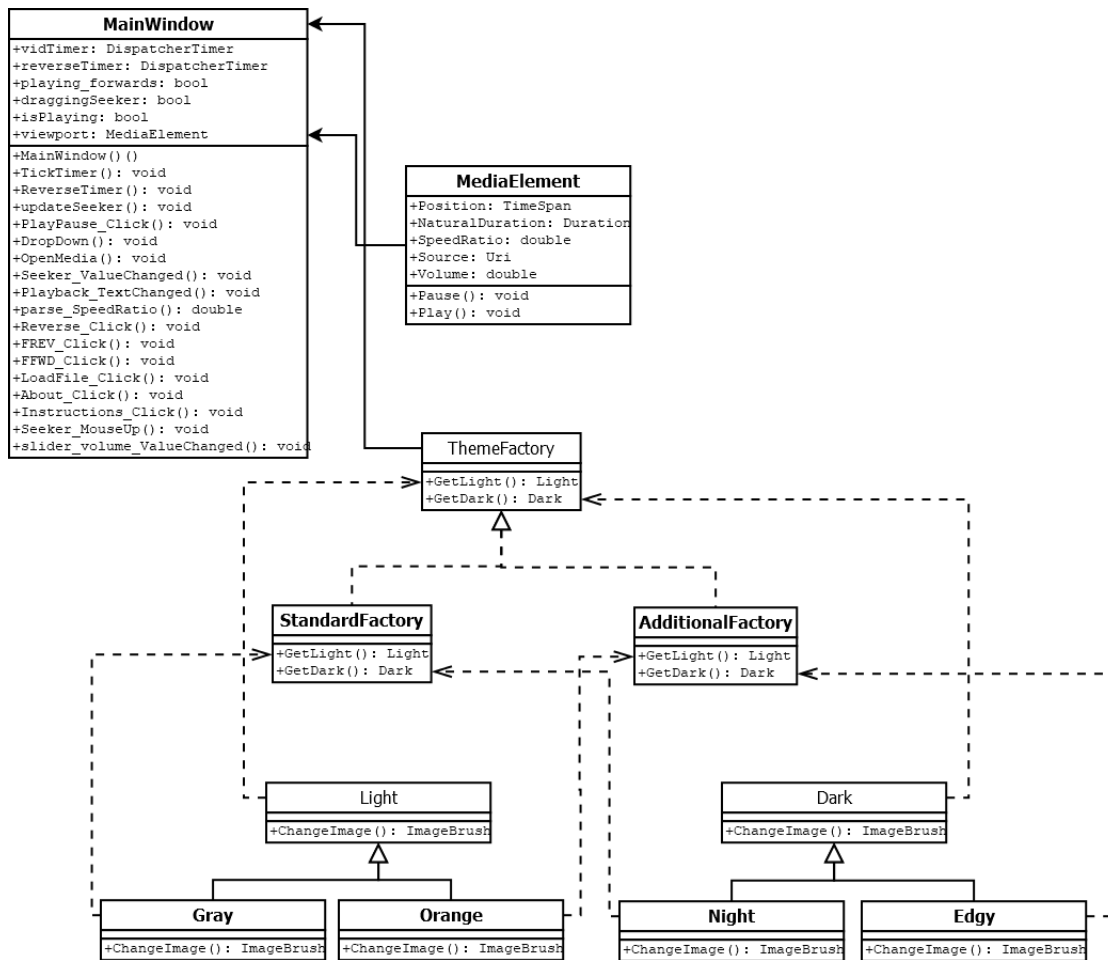


Figure 4. UML of class structure

## 5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

## References

- [1] H. Kopka and P. W. Daly, *A Guide to L<sup>A</sup>T<sub>E</sub>X*, 3rd ed. Harlow, England: Addison-Wesley, 1999.