MBARI Media Management

vlcj/JavaFX based video player

Proposal

Caprica Software Limited

December 13th, 2019

Table of Contents

1 Introduction

MBARI have a requirement to deliver a video player application based on vlcj and JavaFX.

This document presents an analysis of the requirements resulting in a breakdown of the work packages required to deliver a solution.

Requirements and acceptance criteria are sourced from the following document:

https://github.com/mbari-media-management/Cthulhu/blob/master/docs/requirements/C1_PHASE1.md

1.1 Scope

This is potentially a multi-phase project.

This proposal is concerned only with those requirements identified for Phase 1.

In the aforementioned Phase 1 requirements document, a number of requirements are marked as “out of scope” for Phase 1, these have been omitted from this document.

1.2 Key Requirements

There are a number of fundamental requirements:

- Target Java version is Java 11
- Software must use standard build tools (e.g. Maven or Gradle)
- Software must be developed in Java

1.3 Target Platforms

There are a number of mandatory target platforms:

- Primary platform is macOS (Catalina)
- Secondary platform is Linux

2 Major Software Components

2.1 vlcj

The vlcj project provides a Java API and framework for embedding a native VLC media player into a Java application.

The vlcj project is an OpenSource project that is redistributed according to the GNU GPL Version 3, or alternatively as a commercial product.

2.2 JavaFX

JavaFX 13 provides a new native “PixelBuffer” API that can be used as a video frame buffer that can be shared directly with a JavaFX ImageView component to render video frames as efficiently as possible.

2.3 LibVLC

LibVLC is the native library interface to the cross-platform VLC media player developed by VideoLAN.

LibVLC and VLC are mostly distributed according to the GNU LGPL Version 2.1 license, note however that some plugins may be licensed under GPL or other licenses.

2.4 Licensing Concerns

Using software licensed under GPL has serious implications that must be understood by the client.

3 Requirements

3.1 Methodology

Work packages and tasks are identified by analysing the requirements.

An estimate of the difficultly or complexity of each task is made using “story points”.

Story points are used to estimate tasks based on their comparative complexity, they are not directly analogous to a number of days.

Development sprints are planned to include a specific number of achievable story points, a sprint typically covers two weeks elapsed time.

3.2 Work Packages

3.2.1 Create a standard JavaFX application

Create a standard JavaFX application with a main frame, an embedded video player, and various menus and/or toolbars as needed.

#	Task Description	Story Points
1	Create a skeleton application, basic layout, menus and toolbars	2
2	Integrate an event bus	1
3	Integrate a vlcj media player	1
4	Implement a system to manage currently active video windows	2
5	Implement an application domain model	1
6	Implement a mechanism for persistent file-based preferences (e.g. YAML or JSON)	1
	Total	8

3.2.2 Media player user interface / video controls

Create a standard JavaFX application with a main frame, an embedded video player, and various menus and/or toolbars as needed.

#	Task Description	Story Points
1	Implement a dynamic media player controls pane, adjusting visibility based on mouse and focus events	3
2	Implement a current time indicator	1
3	Implement a toggled duration or time remaining indicator	1
4	Implement play/pause buttons/icons	2
5	Implement a video timeline (“scrubber”)	3
	Total	10

3.2.3 Shuttle controls

Implement keyboard shortcuts to provide “shuttling” functionality.

#	Task Description	Story Points
1	Implement “shuttling” controls activated by keyboard shortcuts 1. step back (approximately one “frame”) 2. step forward (approximately one “frame”) 3. small skip back 4. small skip forward 5. large skip back 6. large skip forward	2
2	Implement a throttle for skip events so as not to overwhelm the media player with many queued events	3
	Total	5

3.2.4 Menu items

Add menu items to invoke various application functions.

#	Task Description	Story Points
1	Create menu item for “open local file”, including LRU list of previously opened items	1
2	Create menu item for “open remove URL”, including LRU list of previously opened items	1
3	Create menu item for application preferences dialog	1
4	Implement preference editor for UDP listen port	0.5
5	Implement preference editor for bounding-box time-window	0.5
6	Implement preference editors for the various shuttling controls (e.g. to configure how much to skip)	1
	Implement preference editors for any other appropriate settings (like bounding box colours)	1
	Total	6

Note: LRU means “Last Recently Used”.

3.2.5 UDP remote control

Implement existing MBARI library for UDP application communication.

#	Task Description	Story Points
1	Integrate MBARI library, 12 API functions to implement	1
2	Open	0.5
3	Close	0.5
4	Show	0.5
5	Request video information	0.5
6	Request all video information	0.5
7	Play	0.5
8	Pause	0.5
9	Request rate	0.5
10	Request elapsed time	0.5
11	Seek elapsed time	0.5
12	Frame advance	0.5
13	Frame capture	0.5
	Total	7

3.2.6 Bounding box annotations

Implement a system to draw dynamic bounding box rectangles directly on the video, used to create annotations.

#	Task Description	Story Points
1	Create data model (domain/value objects, UI state objects etc)	2
2	Implement rendering of bounding boxes in absolute video coordinates, scaling and repositioning correctly as the video window resizes	3
3	Correct rendering of “concepts” related to bounding boxes, associating the caption with the box, ensuring correct position and scaling, and implement a heuristic to reposition the caption to always be within the video bounds	3
4	Implement system to cue bounding boxes for rendering at the correct time within the video timeline	5
5	Implement algorithm for conversion of absolute video coordinates to screen coordinates and vice versa	2
6	Implement a mechanism to create new bounding boxes (e.g. by clicking and dragging with the mouse)	3
7	Implement a heuristic to constrain bounding boxes to the video bounds	1
	Total	19

3.2.7 Editing bounding boxes

Implement a mechanism to edit existing bounding box rectangles.

Note: not all of these items were specified in the Phase 1 requirements, nevertheless those "extra" items seem desirable.

#	Task Description	Story Points
1	Select existing bounding boxes by click, and render differently	0.5
2	Delete bounding box via selection	0.5
3	Delete bounding boxes via UDP	0.5
4	Update bounding boxes via UDP (e.g. to change the concept)	2
5	Resize bounding boxes via keyboard/mouse (e.g. grow, shrink)	2
6	Reposition bounding boxes via keyboard/mouse (e.g. dragging or cursor keys)	2
7	Send new bounding box details via UDP	0.5
	Total	8

3.2.8 Testing

Developer testing is performed as a matter of course. This work package is concerned with more structured end-to-end testing. The testing tasks here also to include bug-fixes and other small changes as a result of client feedback.

#	Task Description	Story Points
1	Testing on macOS (primary platform)	5
2	Testing on Linux (secondary platform)	3
3	Creation of test harness to simulate UDP remote control application	3
	Total	11

---

4 Resources

This section lists the development resources required to deliver the proposal.

4.1 Skills Profile

The rates shown below are Caprica Software's standard nominal consultancy rates, and are charged in £ (GBP) per day (whole days only).

#	Resource Description	Quantity	Rate
1	Java/JavaFX developer	1	400-500

4.2 Estimates

A total of 74 story points across eight work packages have been identified.

For this project, team velocity is estimated at approximately three story points per nominal working day.

The estimate for delivery of the work packages described above is therefore approximately 25 days.

4.3 Costs

Caprica Software are prepared to offer resource to be available at the bottom end of our standard consultancy rate, i.e. nominally GBP 400 per day.

We require 25 days of effort, on the understanding it may require more than 25 working days elapsed time.

Total costs to deliver this proposal would therefore be GBP 10,000.00, or a broadly equivalent amount in USD depending on exchange rate (this can be negotiated separately).

4.4 Resource Availability

Please note we are unable at this time to offer full-time dedicated resource to this project, it is more likely to be 50% availability, though this can change on a day-by-day basis.

5 Caveats

The choice of software for the solution necessitates some constraints that must be tolerated.

5.1 Limitations of vlcj/LibVLC

The vlcj project is a Java framework for the native LibVLC library. This library is part of the extremely popular VLC media player from VideoLAN. VLC is capable of playing just about any type of video without the use of external codecs.

LibVLC is an API primarily for embedding a media player in a host application – it is not a general purpose API for video processing and as such there are some inherent limitations that possibly can not be worked around.

For example:

- no reverse playback, other than a coarse skip function
- no previous frame (next frame is provided)
- seek accuracy is limited, e.g. frame-level accuracy is not possible (individual video frames can not be addressed), even milli-second accuracy is not possible and seeking to a particular position may result in a loss of precision

5.2 Limitations of the target platform

Prototype work has been carried out using JavaFX 13 on both the macOS Catalina platform and a contemporary Linux platform.

Performance, in terms of aspects like flicker-free rendering and animations, of a Java FX 13 application running on macOS was seemingly flawless.

Running that same application on a Linux platform resulted in sub-optimal performance with various visual glitches especially when resizing the application window.

These issues are inherent to the platform and this proposal can not guarantee a solution or workaround to such inherent platform limitations.

5.3 Limitations of VLC

The main concerns here are with regards to 4K and especially 8K video.

This proposal is limited by the constraints that VLC operates under.