LANGMINE MEDIA PLAYER APPLICATION

[1]Shubham Beerh, [2]Taru Jain, [3]Ritik Gupta, [4]Gunjan Chugh
[1, 2, 3] Students, [4]Assistant Professor
Department of Information Technology
GGSIPU, New Delhi, India

Abstract
With the rise in advancements in the entertainment technologies, the demand for user friendly, lightweight and platform independent media players has drastically increased. Traditional media players are often platform dependent, heavy weight and not very intuitive to use. In this work, we present a media player application, which works well on almost all operating systems, is light weight and user friendly. We also equip our media player with a translator to aid language learners. In this paper, we detail the purpose, design and complete technology stack for our media player.

1 INTRODUCTION

The rise in entertainment and multimedia sector has increased the demand of media player applications. However, there is still room for improvement of such applications. Current applications are either platform dependent [Rowe and Smith(1992)], or without any translation support [VideoLan(2006), Daniel(2018)].

In this work, we present a media player application that can be easily rendered on any kind of desktop system without any dependency issues. We have focused on making it more user interactive and friendly as compared to the solutions already available in the market. We integrate a translator in the application to ease the process of translation for users especially language learners. Specifically, we deliver a lightweight application, with minimal memory requirements and high efficiency. The installation is non-tedious and user interface (UI) is kept simple so as to make it user friendly and target users of all age groups and computer literacy levels. It is a fully deployed, ready to go software

2 PRIOR WORK

The existing media players such as VLC [VideoLan(2006)] [1]media player and windows media player [2] serve a great purpose but are arduous to install in Unix based operating systems. Some are completely based on UNIX platforms

[Rowe and Smith(1992)]. Systems such as [Mahmoud et al.(2003)Mahmoud, Yu, and Long] are LAN based and might be inconvenient to users who are in remote location and do not have sufficient internet bandwidth. Other systems such as KODI [Daniel(2018)] 3 though platform independent, are quite heavy as compared to the solution we provide. Moreover, to the best of our knowledge, there exists no media player application with a built-in translator feature.

3. SYSTEM DESCRIPTION

Java is known to be go-to language for building systems with as few dependencies issues as possible. Since we also wanted our system to be user friendly, we used Java FX 4

3.1 System Design

The Java FX includes an FXML document. The purpose of this FXML document is to create an online stage which contains a scene, which further contains the User Interface of the software.

The stage and the scene are being controlled by the main JAVA file, whereas, the FXML elements are controlled by the controller file and their designs will be designed through an external Cascading Style Sheet (CSS).

3.2 Functionalities

Like other media players, ours too provides most or all of the following features. It allows the users to play any mp4 file with ease and a lot of additional features. The user interface will be simple and handy for anyone to use it easily. It will have a play button, pause button, forward and backward buttons, volume and seek sliders and much more.

Specifically, it will have the functionalities of Play, Pause, Start, Stop, Fast Forward, Faster Forward, Slow, Slower, Volume Slider, Seek Slider, Exit, and Translate.

We would like to elaborate the working of the translator feature. In order to ensure that the free availability of this feature, we did not make use of any paid service for instance Google API. Instead, we created a script on Google Scripts, which we then published and deployed as a web app. This generates a link which can then be called as a function argument in the document controller class in FXML. This function will also contain the source and target languages which are to be translated be-tween. This complete function can be given as an input to the web app we created which will then generate corresponding response and pass this response as the function's output. The user can call this function by clicking on the translate but-ton. The input fields such as the word to be translated, language selection tools (source and target language codes), the translate button and the output field (the translated

version of the source word), are all integrated together by the FXML document. This translator feature being free also have some quota limitations and the Google scripts might al-low a set number of translation requests per day.

3.3 System Requirements

LangMine Media Player will be compatible with the following processors:

Core2Duo Pentium

Intel Core Processors - i3, i5 and i7

Memory Requirements: LangMine Media Player requires a minimal space of approximately 1 MB depending upon the platform on which the player is being compiled and then getting executed.

Operating Systems Support: Our media player supports Windows, Linux and Mac

System Setup: Following are the requirements that need to be setup in the system in order to ensure a smooth run of the application:

Java Development Kit (JDK) should be pre-installed.

The user and system paths should be set to JDK directory.

JAVA HOME creation and its path should be set to JDK directory.

JAVA TOOL OPTIONS creation and its value set to Dfile.encoding=UTF8

WiFi is required.

4 TECHNOLOGY STACK

JavaFX: It is a java library for designing, creating, testing and deploying cross platform GUI applications. It comes with a plethora of built-in UI controls like TableView, ListView etc. All the UI components can be styled using CSS. We discuss its main components required to build the application as follows:

FXML: It is an XML based language used to create the user interface of an application. This enables us to compose JavaFX GUIs in a fashion similar to how web GUIs are composed in HTML, therefore enabling us to separate JavaFX layout code from the rest of application code and thus ensuring clean up of code. The way FXML documents are controlled is by setting up a controller class for an FXML file. This class can bind the GUI components declared in the FXML file together, making the controller object act as a mediator.

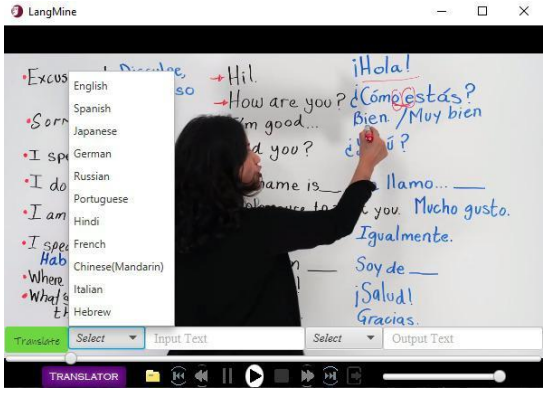

Figure 1: Source language feature in translator of LangMine

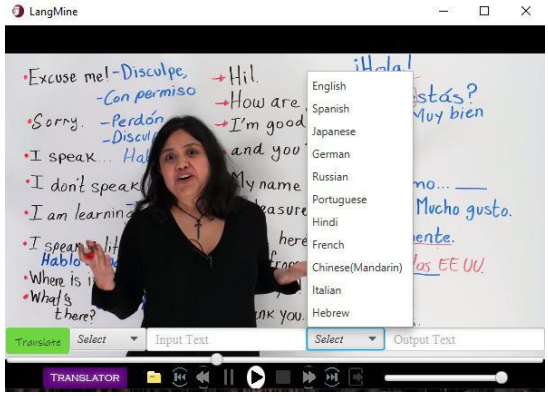

Figure 2: Target language feature in translator of LangMine

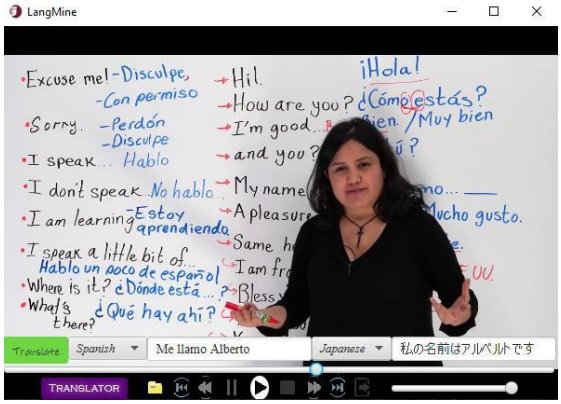

Figure 3: Translation process in LangMine. Video Source

JavaFX's Scene Builder: It is a tool allowing us to design user interfaces without the need to write code. It can be integrated into all the major Integrated Development Environment (IDEs) like Netbeans, Intellij, and Eclipse. We can drag and drop UI components to the work area, modify their properties, and apply style sheets

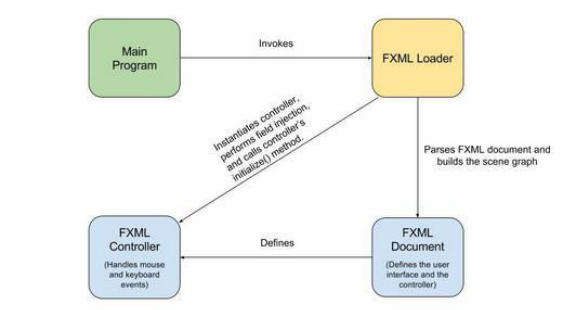

Figure 4: Application structure for JavaFX. Source

Styling Sheets in JavaFX: Style Sheets con-tain style definitions that control the look of user interface elements. Using CSS in JavaFX applications is similar to using CSS in HTML. The default CSS for all JavaFX applications is written in a file called mod-ena.css. This default CSS is always applied to every JavaFX application.

5 RESULTS

In this work LangMine Media Player, we have achieved the following:

1. Size: We get success in making the size of our multimedia player small. The LangMine Media Player is of 51 kb (jar file). Size of the media player matters a lot. If the size is large say 50 mb (approx.) then it will take a lot of cpu time(processing time) to get loaded in the main memory to become functional. As when we open the player the operating system sends a request by the use of system calls to load the media player file from the secondary memory to main memory. There-fore the size of the media player must be small so that the user can access the features of the player efficiently and does not require extra time.
2. Platform Independency: There are so many operating systems available in the market for user; some of them are windows operating system, Linux operating system, Ubuntu operating system and many more. Different operating systems acquire different features according to the demand of the user. Some operating systems are user friendly but slow on the other hand some are fast but not user friendly. Some operating systems are good for developing applications while some are basically for gaming. Moreover more than one operating system can also work in the same system by sharing main memory. The media player is the basic need of the operating system. We also get success in making our media player application platform independent. Our operating system works efficiently on different operating systems. The system must have the java installed as we achieved this feature of platform independency by making use of javafX. The user does not need to download different media player applications for playing media in different operating systems. The user can use our proposed media player in different operating systems efficiently.
3. Features: We are successful in developing the new features in the media player along with the traditional features. Features like Play, Pause, Fast, Faster, Slow, Slower were already there in the media player. We find features like Seek Slider more useful for the user, which is also used in YouTube. The seeking event occurs when the user starts moving/skipping to a new position in the audio/video. The user finds this feature very helpful because user can skip some part or can visit that part again if somehow user missed it. Therefore we also get this feature developed in our proposed system.
4. Translator: Now days the media industry is growing exponentially. Media is avail-able in more than million languages. The users are also enthusiastic to explore the me-dia of different languages other than their native languages. But for viewing media in other language the user requires subtitles. By making only use of subtitles the user find difficulties in understanding the story. So the user has to search for different words and their meanings as well. We successfully achieved in resolving this problem by developing the translator feature in which the user can search for the words and their meaning without going on Google by making use of translator. The user just needs to pause the media and search the word in the media player application itself.

6 DISCUSSION

We would also want to discuss the limitations and future work recommendations for our work. Even though our application works well on all desktop operating systems, it is not really the case for mobile operating systems such as Android. Our application can therefore be improvised by making it feasible to run on mobile systems as well.

7 CONCLUSION

In this paper, we present a platform independent, lightweight media player application with a user friendly interface. Our novelty lies in integrating a translator feature in our application that provides convenience to users, especially language learners by resolving the hassle to switch applications in order to infer the meaning of a new word/phrase encountered. We detail the system requirements, setup and technology stack in order to help the re-search and developer community to build upon our system, We also mention future work recommendations for the same

REFERENCES

1. [Daniel(2018)] Brown Daniel. 2018. Kodi User Guide: Complete User Guide With Pictures On How To Install Kodi On Samsung Galaxy S7 And IOS Without Jailbreaking Your Ipad or Iphone. CreateSpace Independent Publishing Platform, North Charleston, SC, USA.
2. [Mahmoud et al.(2003)Mahmoud, Yu, and Long] Qusay Mahmoud, Leslie Yu, and Christopher Long. 2003. damp: A jxta-based peer-to-peer media player. page 229.
3. [Rowe and Smith(1992)] Lawrence A. Rowe and Brian C. Smith. 1992. A continuous media player. pages 376–386.
4. [VideoLan(2006)] VideoLan. 2006. Vlc media player.

FOOTNOTES

[1]https://www.videolan.org/index.html
[2]https://windows-media-player.en.softonic.com/
[3]https://kodi.tv/
[4]https://docs.oracle.com/javafx/2/

