Educational Platform For Creating Interactive Learning Material For Command Line Interfaces

Naman Dixit 19305R005 Abbas Ahmad 193050072 Prince Kumar 193050070

Indian Institute of Technology, Bombay

November 26, 2019

Abstract

We present an extendable platform for creating hands-on, approachable and beginner-friendly educational materials and courses to teach command line tools. We also detail some of the technical details about our approach as well as our vision for the future of the tool.

1 Introduction

In this project, we implement a platform which enables educators to create interactive learning material to teach command line interfaces and the corresponding tools. The platform provides a uniform user-friendly environment to learn CLIs for beginners. Our goal is to make learning CLIs approachable, safe and fun. To do this, we use the principles of user experience and virtualization for the sake of the student, while making sure that the platform is flexible and extendable to empower the teacher.

2 Motivation

In our experience, we have found that many students who are novices in using command line find it very obtuse and alien. They also fear learning by trial-and-error for the fear of messing up something irreversibly, due to a lack of hand-holding from such interfaces. This results in a lack of soft skills in using Linux and other UNIX systems, which then hampers the expression of the student's hard skills. An example of such phenomena would be an inability to write Makefiles due to not being proficient with the way compilers accept arguments on the command line. Thus, we found ourselves motivated to find a way to create an approachable, non-threatening teaching environment for command line interfaces, aimed at new users.

3 Prior work

While researching the problem space, we stumbled upon many projects with similar goals. SOme of the more noteworthy ones were:

War Games. Developed for teaching security concepts in Linux [7]

Terminus. Fantasy game where commands operate akin to magic spells [5]

VIM Adventures. An Zelda-like adventure game for teaching VIM keyboard bindings [4]

clmystery. A murder mystery game that tries to get user accustomed to CLIs [6]

While these works are laudable, we felt that each of them were too domain-specific and not flexible enough to cater to the diverse needs of both teachers and students. Thus, we decided to create a general platform on top of which it would be possible to create a any of such specific tutorials.

4 Implementation

We wrote the lower level code for this software in C to make sure that project was as portable as possible. We also used Lua 5.1[1] to implement a higher-level scripting layer; since Lua is itself written in ANSI C, this choice did not hurt the portability of the platform.

The overall architecture of our system consists of following components:

4.1 Platform layer

This is the lowest level layer and manages the interactions of the software with the hardware and the operating system. Such interactions include creating a window, talking the the graphics processor, playing audio, receiving user input, etc. This layer is implemented almost completely using SDL 2.0.4[3] and OpenGL 4.5[2]. This means that this layer in easily portable across a variety of systems, including Windows, Linux, Android, etc. The small part which is Linux-dependent has to do with debugging and is not critical for the software to function.

4.2 Engine layer

The next level of abstraction comes in the form of a game engine, which we implement on top of our Platform Layer. This layer implements the most basic primitives that are still recognizable as a meaning operations. These include rendering a quad, loading script code, asset management, etc. This layer is completely platform independent.

4.3 Interface layer

In this layer, we stop programming in C and move up to Lua. Using it, we implement a virtualization layer that emulates basic properties of a Linux system (shell, file system, etc.) allowing experimentation in a sandbox. We also implement a communication mechanism between the C and Lua parts of the software, as well as the lay the architectural foundations for the higher-level code.

4.4 Gameplay logic

We implement the user-interface and well as all the logic that governs user interaction with the program. We also implement a mechanism to create educational material, that can be created and deployed without having to know the internals of the program.

4.5 Tutorials

In this layer, we finally implement various commands and their instructional material. This are the actual learning material that can now be written and executed on this program, and can be used by educators to teach various command line tools.

5 Requirements

As it stands currently, this project requires the following:

- A computer with a recent Linux installed
- GPU capable of running OpenGL 4.5, with proper drivers installed
- Display with height greater than 720 pixels

6 Usage

To compile the project, one need Clang 8.0 compiler installed on a Linux computer. To compile, run the `build.linux` script from the project's root directory. To execute, run `./bin/linux/x64/game`.

The tutorial sequence can be changed in `data/scripts/tutorial.lua` while commands can be added by adding the appropriate file in the directory `data/scripts/command` and then adding an entry for the file in `data/scripts/command/command.lua`.

7 Conclusion

In this project, we created a platform to teach command line interface in a more user-friendly way so that beginners might find it intuitive and fun to learn. We succeeded in creating such a platform, and were able to create a small demonstration of a tutorial to showcase the features of the platform.

8 Future Work

While a basic platform is complete, we would like to add more features to it in order to enable a richer set of primitives to construct the educational material from. We would also like to create full sized tutorials, and implement many more commands and their usage tutorials.

References

[1] Lua. URL `https://www.lua.org/manual/5.1/`.

[2] Opengl. URL `https://www.opengl.org/`.

[3] Sdl2. URL `https://www.libsdl.org/`.

[4] Doron Linder. Vim adventures. URL `https://vim-adventures.com/`.

[5] Michele Pratusevich. Terminus. URL `http://www.mprat.org/Terminus/`.

[6] Noah Veltman. clmystery. URL `https://github.com/veltman/clmystery`.

[7] OverTheWire. Wargames. URL `https://overthewire.org/wargames/`.