Design Prototype

Evidence of Soundness

Theoretical Background

The Microcomputer Museum project is an effort that presents classic video games & systems as a robust experience which combines the fun, hands-on aspect that emulation offers with the cultural significance & preservation efforts that a museum exhibit would bring. While each of these elements exist separately in their own rights, our goal is to integrate them and provide the user with an interactive and authentic experience.

Prior Art

Our project in its current form offers users the opportunity to play Nintendo Famicom/NES games via the FCEUX Emulator, but there are plans to extend our efforts to other consoles [1]. While this idea has been done in the form of plug-and-play systems like Nintendo's Famicom/NES Classic mini, our project will work to replicate the authentic experience of playing a game system in the 80s/90s through an immersive environment [2]. Furthermore, video game museums do exist—however they tend not to be as interactive and instead simply serve as displays of older hardware. [3].

Empirical Evidence

Every project requires experimentation and research before implementation. Our project is no different. Our exploratory and experimental efforts include methods for launching FCEUX from the GUI and different ways to load ROMs. These actions can vary across operating systems so it was important to find a way that works seamlessly using the Linux-based Raspbian OS.

External Interfaces

Presentation

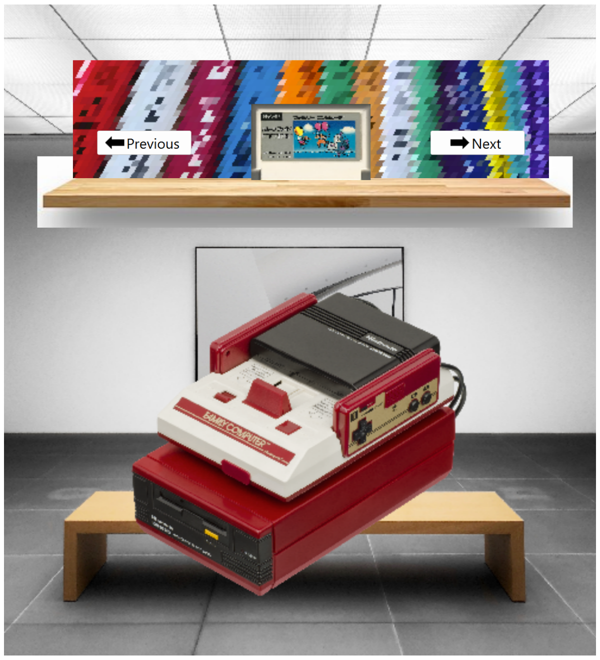

Figure 1: Screenshot of GUI(C++)

The graphical interface is relatively simple with a few background images used for aesthetic purposes(art gallery, bookshelf, cartridge background). It also contains a next and previous button used for scrolling through the loaded ROM images. There is a center image of the Famicom system which is where ROM images will be dropped to be loaded into the FCEUX emulator as it starts up. There is another image which is centered in the middle of the bookshelf and represents the current ROM which can be loaded by dragging and dropping its image onto the Famicom

Perception

ROMs that are available are loaded into a vector alphabetically with ROM images loaded the same way so their indices correspond to the related game. Each ROM image can be seen by clicking through the vector elements by pressing the next or previous button. When the user finds the game they would like to play they can click on the ROM image and drag it down to the center of the Famicom: console and when the mouse is released, the FCEUX emulator is started with that game loaded in. FCEUX emulator has its own set of user interactions allowing keyboard, mouse, and other peripheral input depending on the game.

Usability

Some of the robustness of the system occurs when the next or previous button is clicked on the last or first ROM image respectively it loops to the other end of the ROM vector. For a user looking to add new ROMs they would like to play, they need to ensure that they add the ROM and ROM image to the same respective index in the corresponding vectors.

Internal Systems

Component Architecture

Our graphical user interface (GUI) has been developed using C++ programming language and utilizes QT's cross-platform framework to create a visually appealing and user-friendly interface [4]. This GUI is connected to FCEUX, which is an open-source emulator for Nintendo Entertainment System and Family Computer Disk System, allowing users to play their favorite games seamlessly. Our entire system runs on Raspbian, an open-source Linux operating system, which is designed to boot the GUI automatically after the system boots up on the Libre Renegade single-board computer [5].

Communication Mechanisms

Once our GUI is fully booted up, it displays a visually engaging picture of a Famicom system. Within this screen, you have the ability to click on a button to move forwards or backwards, which will cause the game cartridges to move in the corresponding direction. When you find the game you want to play, simply select it and drag it to the picture of the Famicom, as if you were inserting a real game cartridge. Each cartridge has a corresponding ROM file location, which our program utilizes to automatically open FCEUX and load the specific game associated with that ROM.

The primary method of communication between the various components of our system is through command lines, which are executed via the terminal [6]. This ensures that our system is operating efficiently and with minimal overhead. We did explore named pipe communication and may implement that in the future [7].

Resilience

Up to this point in our code development, we have not been able to implement a resilient system that includes exception handling. However, we have been able to debug our code effectively through the use of QDebug.

As we continue to optimize and improve our code, we plan to implement exception handling in order to catch any runtime errors and other potential bugs that may arise. Additionally, we plan to create a log that will allow us to track any error messages that may occur during the execution of our program. By doing so, we will be better equipped to identify and address any issues that may arise in the future, which will ultimately lead to a more robust and reliable system.

Effort (Percentage Breakdown)

	Research	**Interfaces**	**Systems**
Rudolph Civil	40%	20%	40%
Fatima Elfasi	25%	65%	10%
Christian Walk	10%	70%	20%
Team Overall	25%	51.67%	23.33%

Sources

[1] S. Porst, “The all in one NES/Famicom/Dendy Emulator,” *FCEUX*. [Online]. Available:
https://fceux.com/web/home.html. [Accessed: 19-Apr-2023].

[2] Nintendo UK,
https://www.nintendo.co.uk/Misc-/Nintendo-Classic-Mini-Nintendo-Entertainment-System/Nintendo-Classic-Mini-Nintendo-Entertainment-System-1124287.html [Accessed: 19-Apr-2023].

[3] “National Videogame Museum,” *NVM Museum*. [Online]. Available:
https://nvmusa.org/. [Accessed: 19-Apr-2023].

[4] K. Buzdar, "Compiling your first Qt Program in Ubuntu," *Vitux.* [Online]. Available:
https://vitux.com/compiling-your-first-qt-program-in-ubuntu. [Accessed: 19-Apr-2023].

[5] S. Hymel, "How to Run a Raspberry Pi Program on Startup," *Sparkfun.* [Online]. Available:
https://learn.sparkfun.com/tutorials/how-to-run-a-raspberry-pi-program-on-startup. [Accessed: 19-Apr-2023].

[6] FCEUX. (n.d.). Command Line Options. *FCEUX*. [Online]. Available:
https://fceux.com/web/help/CommandLineOptions.html. [Accessed: 19-Apr-2023].

[7] "Inter Process Communication Named Pipes," *Tutorialspoint.* [Online]. Available:
https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_named_pipes.htm#:~:text=Named%20pipe%20is%20meant%20for,the%20client%20to%20the%20server. [Accessed: 19-Apr-2023].