**Pre-Alpha Build**

Contributors: Rudolph Civil, Fatima Elfasi, Zachary Thrasher, and Christian Walk

**Architectural Elements**

External Interface

Our project's external interface is designed to resemble a museum-like environment where users can interact with elements on the screen. The primary aesthetic goal of this environment is to provide an immersive and novel experience that showcases consoles and games as both pieces of history and art. If multiple consoles are implemented, users will be able to select from available consoles and switch between them using on-screen buttons. Once a console is selected, the ROM library for the respective console will load above it on a shelf, as shown in Figure 1. Users can then switch between games using buttons.

To create a more authentic feel, loading a game will involve clicking and dragging the cartridge image to the console to load the cartridge and then powering on the system with a button (which is not yet included). A game console can be powered on whether or not a game is loaded. The intention is to include the original screen of the console that shows when no cartridge is detected, such as the Famicom's system boot animation.



Figure 1: Screenshot of current GUI (using C++)

Persistent State and Internal Systems

Since we are currently reworking the GUI from Lua scripts to C++, we will have to bind the GUI to the FCEUX emulator. We don't have the specific code yet, but as described in the external interface, there will be checks to see if a cart is loaded, if power is on or off, and what libraries are available for the selected console.

We are still using FCEUX as our Famicom/NES emulator, so it is unlikely we will have to change much from the previous Emugators team's code, except for the tweaks they made for their Lua Scripts. We will likely have to make our own tweaks however, which we will find out once we get to tying the GUI with the emulator.

**Information Handling**

Communication

Our plan is to start up the GUI on boot which can load up FCEUX and load the corresponding ROM that is dragged and dropped on the GUI page. This communication between FCEUX and the GUI could occur through sockets or pipes. This approach leaves the door open for extensibility when we add SNES and potentially other console functionality, which will likely use other emulation software (since FCEUX only supports Famicom and NES emulation).

Integrity and Resilience

Regarding functionality for integrity and resilience, some measures will be put in place to ensure safety and prevent errors. e.g. If a game cartridge is inserted into the Famicom console but the power button is not activated, the ROM will be stored in a buffer, and emulation will pause until the power button is pressed. This process prevents errors caused by premature game startup. Conversely, if the power button is pressed before a ROM is inserted, it will display the Set Disk Card screen until the ROM is loaded.

The Internal Systems will ensure that FCEUX is actively emulating properly. This process ensures system resilience against program crashes by preventing faulty function execution. It acts as an extra layer of protection for the system. Additionally, other checks will be present throughout the code to ensure that all systems being accessed or modified are active before critical communication between systems occurs. These checks will prevent errors or accidental interference.