LED Wall

Eli Lewis and Yigit Turan

Professor Christopher Wierenga, Project Advisor

Spring 2025 Senior Project

Github: https://github.com/LEDwallSeniorProject

Problem Description

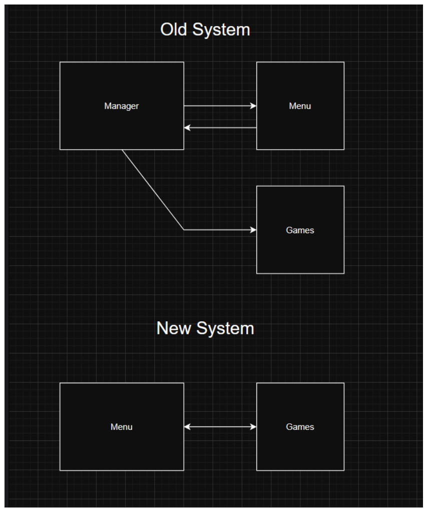
The problem we are attempting to solve with our iteration on the LED Wall senior project is twofold: Firstly, we are attempting to address performance issues that make the user experience frustrating and tiresome. Secondly, we are trying to ease the required development effort for CS-108 students who are developing projects for the board.

Background

In the previous two semesters, the senior project group before us developed two things: Firstly, a hardware LED Matrix connected to a Raspberry PI which allowed the display of images. Secondly, they also developed a Python management program and main menu program which allowed the launch of other subprograms. However, these programs and the way they were implemented had some associated performance and development concerns.

Solution

In order to solve these issues, we did three main things. First, we rewrote the main menu program and management program and combined them into one. Now, instead of the manager program launching subprograms, such as the main menu, games, or demos, the main menu simply instantiates a class containing the subprograms code, passing the keyboard and graphics handling objects as parameters to that class. See the following diagram:



Secondly, we also worked to remove extraneous libraries and to optimize a graphics library in the internal code called Shapes.py. Finally, we moved all code running on the LED Wall into a unified template class.

Design Norms

We think that the following design norms apply to our work:

1) Transparency and Aesthetics
2) Trust
3) Stewardship

The design norms of transparency and aesthetics apply to our work because we made a visually pleasing and usable interface and we enabled easier development for the programs meant to run on the LED Wall. We also think that the design norm of trust applies to our work because the wall is now significantly less prone to crashing and has better error management and logging. Finally, the norm of stewardship applies to our work because the wall runs more efficiently.

Development Approach

Similarly to our development approach last semester, we utilized an AGILE like project management approach to developing this program. Every week, we met with our advisor and were assigned specific “tasks” (AKA stories) which we would then complete in the next week (sprint) before meeting again and either continuing or pivoting based on the progress made and discoveries uncovered. We utilized Git and GitHub as version control. We also employed a type of code review system between ourselves, and would often look over the others code to ensure that it met our expected standards.

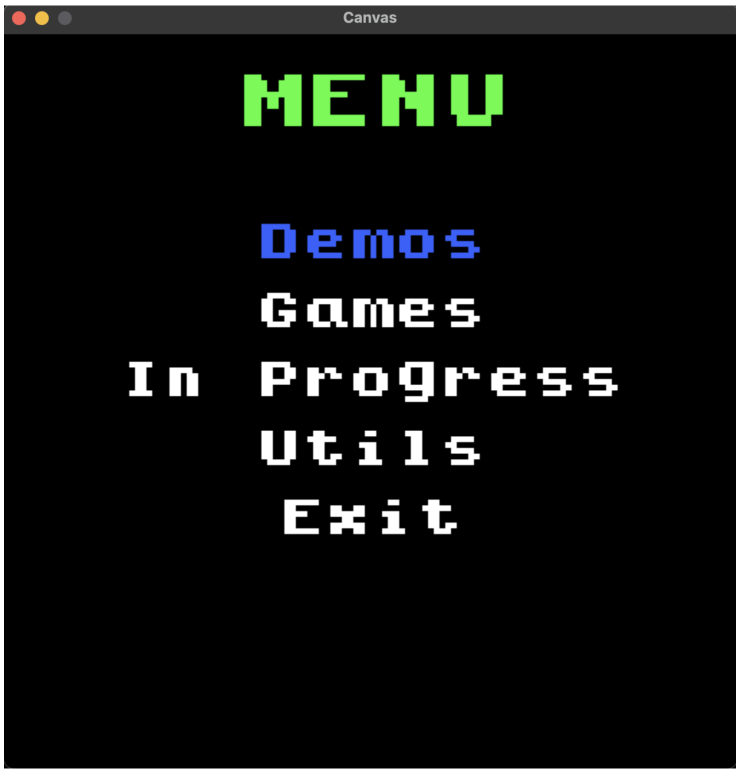
Problems Encountered and Overcome

We ran into many problems over the course of our project this semester. Firstly, it took us quite a while to understand everything that the group before us had done. Secondly, we ran into many bugs and issues due to the fact that we both had Macbooks, while the previous group had all utilized Windows laptops. Finally, we had several instances where after having optimized the code, the performance was worse than when we had begun.

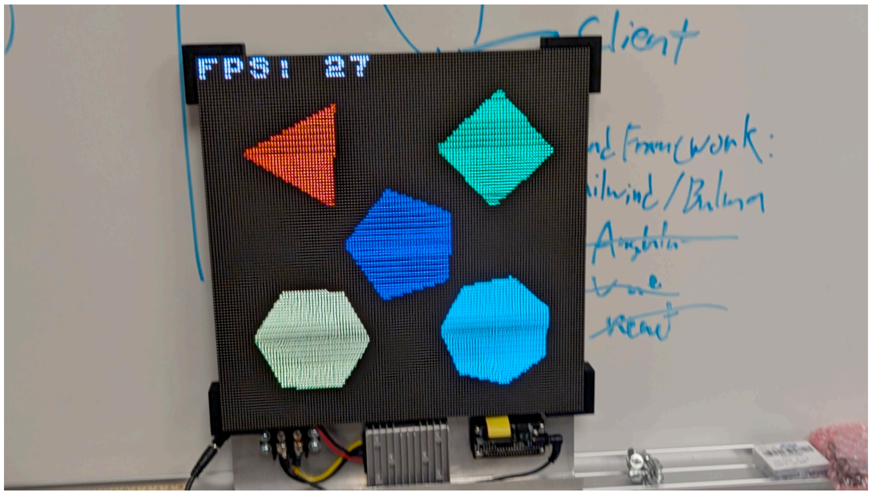
Testing Methodology

There were two methods we used to test our changes to the LED Wall. Firstly, we developed locally and tested them using the emulated version running on our computers. Then, after we were confident that our changes would work, we pushed the changes to the actual LED Wall and tested them there. On the LED Wall we tested our changes by simply using them, and also by comparing average FPS counts on a frame counter feature we implemented. This allowed us to directly compare performance before and after changes.

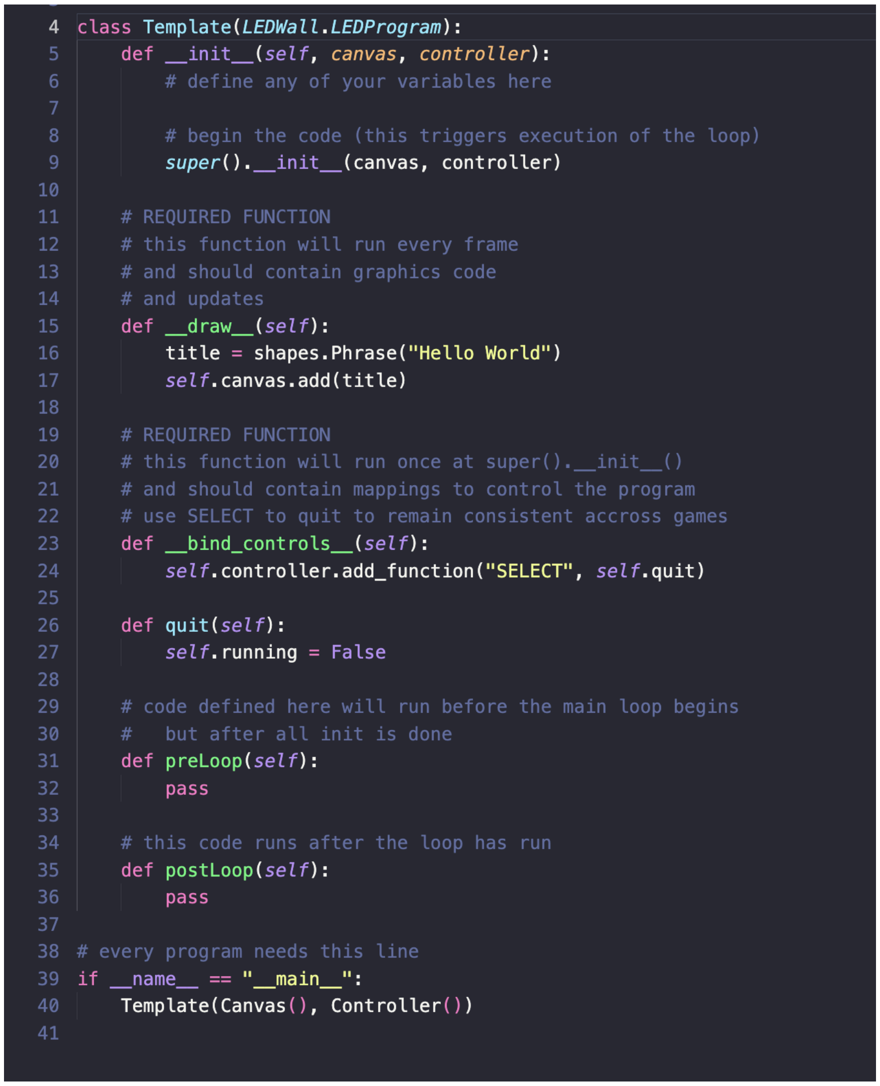
Demo



The new main menu



The board running our benchmark test at 27 fps (previously this ran at 18 fps)



```python
class Template(LEDWall.LEDProgram):
    def __init__(self, canvas, controller):
        # define any of your variables here

        # begin the code (this triggers execution of the loop)
        super().__init__(canvas, controller)

    # REQUIRED FUNCTION
    # this function will run every frame
    # and should contain graphics code
    # and updates
    def __draw__(self):
        title = shapes.Phrase("Hello World")
        self.canvas.add(title)

    # REQUIRED FUNCTION
    # this function will run once at super().__init__()
    # and should contain mappings to control the program
    # use SELECT to quit to remain consistent accross games
    def __bind_controls__(self):
        self.controller.add_function("SELECT", self.quit)

    def quit(self):
        self.running = False

    # code defined here will run before the main loop begins
    #   but after all init is done
    def preLoop(self):
        pass

    # this code runs after the loop has run
    def postLoop(self):
        pass

# every program needs this line
if __name__ == "__main__":
    Template(Canvas(), Controller())
```

The unified template class allowing easier development by 108 students (note the simplicity of the class, the actual code is available on our Github)

Future Work

We hope that future senior project groups can create more programs for the wall, improve the feature set of the menu manager, and improve the development workflow for students.

Acknowledgements

We would like to thank our project advisor, professor Christopher Wierenga, for his guidance and assistance throughout our development on this project. We would also like to thank the previous senior project group who worked on the LEDWall, and all the CS-108 students who developed projects for the wall.