Arcade Zone

## CS5319 Group 03

**Install, Compile, Execute**

Arcade Zone was developed using Python version 3.12.2. The primary Python library that was used to develop Arcade Zone is Pygame version 2.5.2. Any Integrated Development Environment (IDE), such as Microsoft’s Visual Studio Code or JetBrain’s PyCharm, can be used to execute the source code as long as it supports Python execution. Arcade Zone is also executable from the command-line/terminal prompt.

First, install Python:

>>> Install Python version 3.12.2:

Windows: <https://www.python.org/downloads/windows/>

MacOS: <https://www.python.org/downloads/>

Next, to be able to install the Pygame library we install the Python package manager *pip*:

>>> Install pip:

Windows: <https://pip.pypa.io/en/stable/installation/>

MacOS: <https://pip.pypa.io/en/stable/installation/>

After a successful pip installation, the Pygame library must be installed:

>>> Install Pygame library:

Windows: <https://www.pygame.org/wiki/GettingStarted#Windows%20installation>

MacOS: <https://www.pygame.org/wiki/GettingStarted#Pygame%20Installation>

With Python, pip, and Pygame installed, an IDE is needed. If one of the two IDE’s mentioned above is desired for use and not installed, the following links are for installation:

>>> Install IDE:

Visual Studio Code: <https://code.visualstudio.com/download>

PyCharm: <https://www.jetbrains.com/help/pycharm/installation-guide.html>

After all downloads are complete, the Arcade Zone source can be downloaded from GitHub and saved into the desired directory:

>>> Download Arcade Zone source code:

<https://github.com/bat351/CS5319-FinalProjectGroup03-Shaun-Mathew_Paige-McFarlain_Isaiah-Batista>

Two primary folders are found in the Arcade Zone source code: *Selected* and *Unselected*. Selected has the implementation for the Model-Controller-View (MVC) architecture and Unselected has the implementation for the Layered architecture.

>>> Execute MVC implementation:

Open the folder named *Selected* into the IDE of choice.

Run the file named *menu.py*

*>>>* Execute Layered implementation:

Open the folder named Unselected into the IDE of choice.

Run the file named *menu\_ui.py*

**Client-Server to Layered (Virtual Machine)**

As the team began implementation, one of the drawbacks of the Client-Server architecture was dealing with cost and security. If the team was to implement Client-Server to connect two players over a network, we would need to rely on third-party services such as network connectivity and database storage to ensure our system was able to run. Having limited resources we decided to drift from the Client-Server implementation. Additionally, making our application available over a webpage instead of an executable on a local machine introduces security risks since the application would no longer be isolated to a local machine.

Furthermore, our purpose for this application to be executed on the local machine was to encourage the human-human interaction that is sometimes lost when players are connected over a network. Although most user’s today have access to a personal local machine, we decided to move the users to a single machine to further expand the human-human interaction by creating a sense of community through the act of sharing.

Considering these tradeoffs, we opted for a secondary architecture that would allow us to still implement an application that would be direct to download and execute on the local machine. In our project proposal we decided to compare MVC and Client-Server, but during implementation decided to compare MVC and Layered architectures.

**Architecture Design Choices and Differences**

MVC is our primary choice architecture since Arcade Zone can be separated into three main components: Controller for handling the user’s actions, View for displaying the current state of the application to the user, Model for all game-logic needed to have functional games available to the user. Focusing all user input to the Controller and all display features to the View, the team was able to focus on the game logic in isolation. This also allowed the games to be developed simultaneously. When a game was fully developed the objective became integration with the main user interface: main menu. We decided to make each game its own module and integrating to the menu was efficient.

Layered is our secondary choice architecture because it presents a simpler implementation since we could separate functionality into layers: User Interface, Business Logic, Data. Separation gives the system a more organized structure that is easier to follow. Implementing the Layered architecture proved to be easier to follow in our source code since there was not as much overlapping between the primary components such as in MVC.

**Final Selection**

Even after implementation, MVC remained our primary selection for Arcade Zone. Our rationale is that we can continue to scale up and add games without having to keep our main components strictly isolated from each other. MVC makes it efficient to scale up with minimal changes to the main user interface.