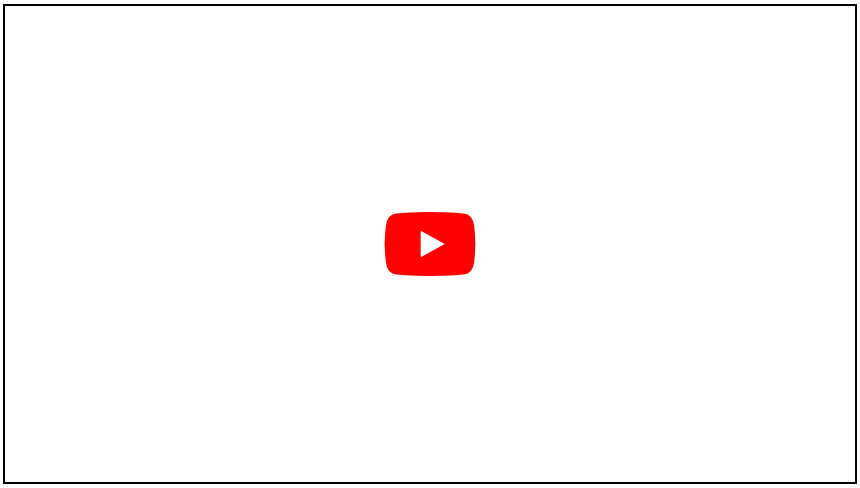
Networked Square Shootout

ECE 5725 2023SP

A Project By Andrew Cheng, Eric Sun



Demonstration Video

Introduction

Welcome to an incredible world where technology and gaming collide! In this thrilling adventure, we delve into the realm of TCP internet connection sockets, Pygame, and the mesmerizing TFT display. Prepare to witness a revolution in gaming as we unlock the secrets of creating a captivating field of vision, complete with bullet-shooting action and seamless movement. We take things a step further by designing meticulously crafted data structures that will be serialized through the very sockets that connect our players. The organization and efficiency of the Model-View-Controller (MVC) pattern takes center stage, with one Raspberry Pi proudly assuming the role of the server, while both Pi devices seamlessly transform into powerful clients. Additionally, we embrace cross-platform compatibility, ensuring that this epic gaming experience transcends limitations. With the addition of cross-platform play, the possibilities become limitless, opening doors to an immersive journey that transcends the confines of a single device. Get ready to embark on a gaming adventure like no other, where technology meets imagination and the boundaries of possibility are shattered. Welcome to the future of gaming!

Project Objective:

- Our primary objective was to synergize the fundamental principles of software engineering with the core components of hardware, namely the Raspberry Pi and the PiTFT. Our focus centered on creating an engaging multiplayer game where two players could connect and engage in thrilling battles within a captivating darkness. To ensure seamless connectivity, we employed a reliable TCP connection that kept the players united in their quest for victory. Get ready for an exciting gaming experience that will keep you on the edge of your seat!.
- A secondary objective was the incorporation of a true server-client model for modern gaming. Through the model, players could play cross-platform, and there could be multiple games hosted on a single server.

Design

Our game is multiplayer on at least two distinct devices, so we first needed our Raspberry Pi to communicate with each other. We decided to use the socket Python library to send data over a TCP connection. The socket API calls to open and configure a network socket are available to consult in the documentation. We used IPv4 and chose all clients and servers to use Port 5555. However, we needed to spend significant effort on configuring the network sockets, especially the IP address, since on different runs, DHCP may assign our our server code a different address by the network router. We also leveraged the pickle library to send over network sockets. Pickle serializes Python data types into a byte stream that can be pushed through a socket and deserialized back into a Python data structure on the other end. We designed our shooter game using the MVC pattern (model, view, controller). In the client-server interaction, the server acts as the model, keeping track of the game state. The clients act as the view component, which displays the view sent by the server. And the client also acts as the controller, sending user interactions such as touchscreen presses and button presses to the server to update the state of the game. We designed a class that would be pickled and unpickled called ShootDTO (shoot data transfer object). Ideally, there would be two separate DTOs, one that the server sends to the client to display the game state and one that the client sends to the server to inform the server of user interactions. However, we ended up using a unified DTO that contained both types of fields. To prevent a client from “hacking” and having a greater field of vision than it should, we cleared fields that the client should not have to know before sending, such as the location of other players outside its field of vision. Data that the client sends comprises user interactions such as button presses to control the movement of the player and taps on the TFT to shoot bullets. We did some computations on the client's end. Based on the coordinates of the button tap and the player, we calculated a unit 2-D directional vector to indicate the direction in which the bullet should move and send it to the server. We utilized the core component of Pygame's objects and frame generation. Each frame consisted of multiple bullets, clients, and screens being “blipped” on and off. Each frame consisted of individual moments and client actions that would be reflected across all clients. The pygame framework provided a strong backbone for implementing our game that we used to springboard into more complexities that the Raspberry Pi provided. Overall, the usage of pygame accelerated the development of our game and contributed to many of the key features that we utilized. One of the core features of our game is the ability for cross-platform capability. Whether it is multi Raspberry Pi's playing against each other, multiple laptops against each other, or even a Raspberry Pi playing against a laptop, each circumstance can play. We sought to reduce the number of restrictions for people. Each group of two players who connect to our server is delegated their own room. All players are set to the same rules regardless of the platform that they play on.

Testing

Our testing consisted of two main components Software and Hardware. For Software, our testing consisted of making sure our server-client connection was stable, working, and without bugs. Within our testing, we found many bugs, including overfilling packet requests, a lack of updates sent from the server, and no connection between the server and the client. For the overfilling packets, we realized that we were continuously increasing the size of the packet being sent to the server by repeatedly appending another object to the server. We found that limiting the number of objects appended and fixing the logic of the game allowed us to limit the resulting overflow of each packet. Secondly, there were many times that our connection would be unstable due to the lack of acknowledgments from the server. As a result, we switched to a more reliable library using Pickle and Socket. Through these two libraries, the connection became more secure and reliable to run our server-client interface. On the other hand, hardware testing mainly required us to manipulate the PiTFT and the various GPIO buttons on the Raspberry PI. We repeatedly went to old labs to backtest various buttons that we were using. Additionally, we chose a test-driven development route by incrementally increasing the number of buttons used. Afterward, we added the on-screen functionality by adding bullets to our characters. Each component of hardware was also tested many times through our Software. When we deduced that our Software was running properly on our hardware, we would move that software to our other instances of Raspberry Pi to test the functionality of the other Raspberry Pi. This lead to stronger back compatibility and cross-checking.

Result

For our project, everything ended up going as planned. Although we had many deviations in the overall idea of our project, the resulting pygame, PiTFT, GPIO centric game resulted as planned. Our team met the software and the hardware requirements for our project. For software, our main focus consisted of a working game that would be playable across platforms and support multiple clients at a single time. Furthermore, we wanted to make sure that the game that would be played was very smooth and provided a good user experience. For hardware, we incorporated every component of the Raspberry Pi we wanted. For the PiTFT, we had initially wanted to just be able to fire into the four quadrant directions. However, after careful engineering of the screen coordinates we were able to improve the directional capabilities of our game. Additionally, the motion control integrated very smoothly into the four buttons provided on the Raspberry Pi.

Work Distribution

Project group picture

Andrew Cheng

akc55@cornell.edu

Constructed the Software Focus of the game by building out the client facing code for players that connect to the server. Enabled the ability for multiple Raspberry Pi's to interface with the same server leading to multiple concurrent games at once.

Eric Sun

eds228@cornell.edu

Constructed the hardware connection of the game by interfacing with the Raspberry Pi, building out the connection between the servers and the clients. Made the game multi platform by allowing different forms of user input to introduce the same output.

Parts List

- 1 Raspberry Pi 4 $35.00
- 1 Raspberry Pi 3 $35.00
- 2 PiTFT TFT+Touchscreen for Raspberry Pi Adafruit $70.00

Total: $140

References

PiTFT document (https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/downloads)
Pygame Documentation (https://www.pygame.org/docs/)
R-Pi GPIO Document (https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/)
Pickle Documentation (https://docs.python.org/3/library/pickle.html)
Socket Documentation (https://docs.python.org/3/library/socket.html)
Code Github (https://github.com/andyhero12/submitECE5725)

Code Appendix

```
#IP Setup for Game Calibration
import socket

# Get the host name
hostname = socket.gethostname()

# Get the IP address associated with the host
ip_address = socket.gethostbyname(hostname)

print("Hostname:", hostname)
print("IP address:", ip_address)
```