Logo, company name

Description automatically generated

Computer Games Development

Project Report

Year IV

Sasa Kuzmanovic

C00249246

15/11/2022

****

Student Name: \_\_\_\_\_\_\_\_SASA KUZMANOVIC\_\_\_\_\_\_\_\_  
Student Number(s): \_\_\_\_C00249246\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature(s): \_\_\_\_\_\_\_\_\_Sasa Kuzmanovic\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_24/04/2023\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Contents**

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

[Project Introduction and/or Research Question 2](#_Toc54714375)

[Literature Review 2](#_Toc54714376)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 3](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# Acknowledgements

I would like to thank the following people who assisted in completing this project including:

Dr. Martin Harrigan who kindly provided guidelines on how my project should be progressing and what should my next step be. Being supportive in my decisions throughout the project and help me achieve my goals.

I would also like to thank my fellow class mates for testing my software and providing input on what should be improved to help increase the quality and user experience in this project.

I would also like to thank AwesomeBlade for allowing me to user their game (8BitBoy) for testing my software.

# Project Abstract

This project will introduce a way of controlling someone's input in a game through text messages, when the player wants them to. There are not many solutions to this problem which makes it a perfect project for me which will also test my skills as a programmer.

There has been attempts to create a solution for this, but nothing has gained traction to become an option that everyone uses. Sharing a screen can be done over discord, the application that I will use in this project to send the message and receive video from the other side. It is a widely used application that has always missed a feature like this. This would create a fun environment for players where they could compete in a game through text chat and would increase the fun factor of playing together. Playing with someone in a different country is difficult if the game itself does not allow it, this solution might just be the thing players needed to have fun.

# Project Introduction and/or Research Question

The project itself is quite niche, since I have been struggling to find a piece of software that does the same or a similar thing. When the player loads up a game, he may be struggling to pass a certain point in the game and may need extra help. This is where he will be able to go into a discord call to get his friend on board.

This can be an especially important piece of software as it will enable multiple people to control a single game. This could be used for many purposes like fun experience for game streamers, or only regular gameplay. This enables a Streamer to allow their viewers to provide input in the game, making it more engaging and causing chaos.

Viewers typing commands into the text box, will be able to toggle controls for moving forward, or attacking, to make it feel more like actual gameplay instead of it being very clunky and stuttery gameplay. The user will only have to type the messages and the software will handle the rest of the work. My main objective is to make sharing games together a fun and enjoyable experience for everyone involved using this piece of software.

Research Question (Networking): How will you manage multiple users sending commands into your chat, and transfer those commands into gameplay?

This project should allow players to have more fun together and play the game together when a person decides to pass the controls to the viewer.

# Literature Review

Due to a fairly limited number of projects that resemble mine I will be able to display very limited amount of data on them.

***Mixer - Microsoft***

Mixer controller sharing is a feature that was first introduced by Microsoft for their Xbox consoles which was allowing streamers to play games with their viewers as a part of their new Mixer streaming platform.

Built on top of a console like Xbox that now has declining sales and Mixer as a platform eventually failing this feature never really took off.

* ***Controlling the game***

Allowing the person to control the game makes the person feel connected to the streamer and will allow them to have more fun while playing a game. This is exactly what I wanted to achieve with my project.

***Chaos Tricks – Chaos Tricks***

Chaos Tricks is a piece of software which would allow streamers to let their viewers influence the game they are playing. The streamer would be streaming their gameplay on twitch and running the software on their computers. Viewer would then be able to donate money on this stream to trigger certain events in the game.

* ***Influencing controls***

For a viewer to be able to trigger certain events in the game it would create an environment fun for both the streamer and the viewer. This allows multiple people to participate in the game and change the outcome of the game which seems fairly interesting.

* ***Limiting users***

ChaosTricks also introduces limits to which people can influence the game with their donation system triggers. This would make sure that there isn’t incredible amounts of people trying to trigger something that would overload the system and result in problems.

***Influence on Project***

These pieces of software have changed the way I wanted to execute my project. From Mixer I took the idea of sharing a controller with someone to be able to overtake someone’s game and help them out or even play a co-op game together. Since Discord provided a way of sharing gameplay footage in real time in a voice call and allows multiple people to watch it I knew I wanted to create something multiple people can enjoy at a time. This is where ChaosTricks came into play. I took the idea of having multiple viewers at the time to play the game. I did not introduce a limit on how many people can play the game at the time. I was also thinking about a clever way of limiting who can play the game and who can establish a peer to peer connection with the streamer just like ChaosTricks. I allowed the Streamer to assign Roles on the Discord server they are currently streaming their game to, to moderate who can control their game and who is only able to watch. This way I will prevent malicious people from doing harm during the gameplay of two or more people.

.

# Evaluation and Discussion

My evaluation is slightly different and is based on testing with multiple people since there was no way of recording the latency from 2 computers at once.

While the project was using chat to detect words that are being sent into the text channel, there was a limitation straight away that started popping out at me. Latency at which the bot will read the messages and interpret them into keypresses and that showing on the screen. At the start this was solution that will allow for further development of the project, but would later have to be revised. With reading the chat, I had to implement a way of actually sending messages into the chat that the discord bot had to read. To do that I had to introduce keyboard presses on the viewer side as well, which would result in having to have chat in focus and ready to type. This resulted in a not-so-great user experience.

Graphical user interface, text, application

Description automatically generated

Limitations from using the discord did not stop there. Discord has a strict policy of the rate at which you are sending messages into chat. I came up with a way of combating this by timing how many times in half a second a button is being pressed, and only sending a single message. When you go over the discord limit there is a pop up on your screen which brings the chat out of focus and the controls stop.

Graphical user interface, application

Description automatically generated

I came up with a solution to battle all these issues. This would be a peer-to-peer connection with the streamer to remove all of these issues with latency, chat highlighting and rate limiting. A p2p connection allows for a direct connection with the streamer which would result in delivering data to the streamer in the fastest time possible. This resulted in having a lower latency for delivering data to the streamer which improved complaints from testers about a gap from when the input was sent and when the action appeared on their screen. There was no longer a rate limit at which input can be sent across and this could be done with having any window in focus on the viewers machine.

Going from a barely playable experience to almost a perfect experience was a massive jump for this project going towards the end goal of enabling friends to play games together. I still believe this is something that I could improve on by adding a layer of protection to the connection.

**Project Milestones**

1. Setting up the discord bot (10/12/2022)

* Milestone was successfully reached on (2/12/2022) hence being ahead of schedule

1. Reading text messages from chat (14/12/2022)

* Milestone was successfully reached on (2/12/2022) hence being ahead of schedule

1. Reading controller input and printing to the console on button press (1.1.2023)

* Milestone was reached on (14/12/2022) hence being ahead of schedule

1. Adding key presses support which can be used in game. (14/1/2023)

* Milestone was not reached on (14/1/2023) hence being behind the schedule by 8 days. Completed (22/1/2023)

1. Accepting the messages from the chat by the discord bot (1/2/2023)

* Milestone was not reached on (1/2/2023) hence being behind schedule 39 days. Completed (9/3/2023)

1. Adding presets to controller binds (1/4/2023)

* Milestone was not reached on (1/4/2023) hence being behind schedule 15 days.

Completed (16/4/2023)

1. Adding a peer to peer connection (25/04/2023)

* Milestone was reached on (22/4/2023) hence being ahead of schedule

Planning for the project was not done correctly and weighing on certain milestones was not assigned correctly. This resulted on being behind the schedule in the middle of the project, but it was caught up in the end.

**Major Technical Achievements**

I have put a lot of effort into establishing a peer-to-peer connection between the streamer and the viewer and I believe I achieved a very impressive result. With a language I know very little of, I have been learning the entire way through the project. Discord bot, pygame, sockets are all libraries I had to figure out from scratch to implement into my project.

Setting up the discord bot was very hard, since there was support for only limited versions of python. Documentation was fairly unclear so I had to resort to testing things for myself. When I managed to get a discord bot running, there was a few methods I was interested in. On\_message was crucial for early stages of the project which allowed me to read the messages from the discord chat that was selected. This then grew in complexity to responding to messages from the user in the chat and later interpreting those messages into key presses on the keyboard. To do this I used a pynput library which allowed me to specify which keys I wanted to press on the keyboard. This also introduced a problem with having to have chat selected while playing the game which was decreased the quality of the experience.

Decreasing the latency of reading and interpreting those messages into controls introduced latency which was not great for the project. Setting up the peer-to-peer connection was incredibly difficult with such a low-level library like socket but would greatly improve the latency issues I previously had depending on the discord bot and text chat. I had to use the documentation to come up with a way of establishing a connection that was reliable for data transfer. Changing router firewall settings was challenging but was needed in order to establish a connection since firewall would refuse the connection.

After submission I am planning to continue working on this project to make a stable piece of software that will be used by others to improve their gaming experience with their friends.

**Project Review**

The development of the Discord Controller Sharing Project faced several challenges, including technical, design, and implementation challenges. Some of the major problems during the development process were:

Discord API limitations: Discord's API has certain limitations, such as rate limits on API requests, message size limits, and limitations on message parsing. These limitations had to be considered while designing and implementing the system to ensure smooth and efficient communication between the host and clients. The initial implementation using the discord.py library for reading chat messages and interpreting them into input commands faced limitations in terms of delays in reading messages and potential issues with message parsing. 5 messages per second limit was almost always reached since platformers require a lot of button mashing.

Text-chat limitations: One of the requirements of chat messages being read by the discord bot was that the messages were actually being sent into a specific channel. To do this the viewer had to have the chat in focus and ready to type. As soon as the chat was no longer in focus the program would not be able to send messages into the channel preventing the detection of messages and stopping the controller sharing until the person highlighted the chat again to continue with messages. This was resolved by implementing a peer-to-peer connection with the streamer.

Latency and responsiveness: Achieving low latency and high responsiveness in transmitting controller inputs and updating the gameplay video was a significant challenge. The system required real-time performance to provide a smooth and seamless experience for controller sharing, and any delays or lags in transmitting inputs or updating the video could result in a poor user experience. The implementation of the peer-to-peer connection using Python sockets helped address this challenge, as it provided a more direct and efficient way to transmit inputs and reduce the delay while updating the video in real-time. Communicating to a discord bot and communicating to the streamers machine was too unreliable and the latency varied from 100ms – 12900ms. Using the discord bot was a simple way of implementing data transfer, but ultimately it was inferior to the p2p connection which provided real-time input in the game.

Video streaming; Capturing the screen of the streamer was crucial to this project, without it the project would be useless since the viewer would not be able to tell what is going on in the game. Choosing the quality and framerate was visibly impacting the latency at which video was played back, so selecting a lower resolution with lower framerate (720p30) was the most optimal solution.

Game Support: Since button presses were all on the keyboard, different mappings were required to improve the quality of the players experience. At the start simple mappings were used for testing such as Arrow Keys. These proved to be a viable option for testing and adjusting the software to get the lowest latency possible. After all of the issues were resolved, I needed to add more mappings and selecting some of the most popular presets found in games. Added support for WASD and Arrow Keys allowed some of the basic games to be played and to prove feasibility of the project as a whole.

Controller compatibility: Firstly I started using DS4 Windows to detect button presses from my PS4 controller, but I soon realised that this was not going to be an option going forward, since this would only add support for PS4 Controllers. I then found that pygame library finds the controllers that are connected to the machine and interprets all of their input in the same way to increase the number of controllers available. Due to my limited amount of controllers available I was not able to test the depth of coverage for controllers, but all three of my controllers have shown exactly the same input while using the pygame library.

Controller accessibility: Connecting the controller was an annoying problem at the start of the project especially since the connection on mine was not great. Having to have the controller connected before the program starts was something not a lot of people would think of. Disconnecting the controller by accident in the middle of the program would result in having to completely restart the program and setting it up again. This was resolved with implementing hot plugging. Using pygame to resolve this issue was fairly easy and I have not encountered any issues so far while testing it. It enables connecting and disconnecting the joystick while the program is running.

User experience and interface design: Designing a user-friendly interface and providing a seamless user experience was an important aspect of the project. Since I have not managed to create a user interface, using the console will be new to a lot of users. Separating text and clear messages were key in providing clear instructions on what is supposed to be done to start the program.

Security and privacy: Implementing a peer-to-peer connection raised concerns about security and privacy. Ensuring that the communication between the host and clients was secure and that the system did not expose any sensitive information or allow unauthorized access to the host's computer was a critical challenge. Implementing role checking using the discord bot was crucial in improving security and privacy of the users. Connection would not be established if the person does not have a role on the discord server, hence adding additional layer of safety.

# 

# Conclusions

The Discord Controller Sharing Project demonstrated the feasibility of sharing game controllers on Discord in real-time, enabling collaborative gaming experiences. The project utilized Python libraries such as pygame, pynput, and Python sockets to capture and transmit controller inputs, simulate keyboard and mouse inputs, and establish a peer-to-peer connection for seamless communication between the host and clients. Despite the challenges faced in terms of input processing, video streaming, controller compatibility, controller accessibility, Game Support, Text-chat limitations, user experience, and security, the project successfully achieved its goal of allowing users to share their game controllers on Discord and play games together remotely.

The project has potential for further enhancements and improvements in terms of expanding controller compatibility, increased security and privacy during the connection, enhancing the user interface and user experience. These enhancements could make the system more versatile, accessible and user friendly for everyone to try and enjoy. The project also highlights the power and flexibility of python as a programming language for developing innovative projects. Large number of libraries greatly expand the amount of options you have and the speed of which the project progresses.

In conclusion, this project is an innovative solution that allows users to share their gameplay on Discord and lets their friends try out the games as well. Many problems and challenges were encountered while the project was being made, but the overall it provides a good result and usable software for advanced users.

**Future Work**

Indicate what might be some next steps to try (if a student next year was going to undertake a project in this area what might be an interesting thing for him/her to examine?).

The discord controller sharing project has potential for further enhancements and improvements. Some of my recommendations would be:

Video streaming and encoding: Remove the dependency on discord to provide real-time gameplay footage since it can severely increase the latency at which video is played back. Using OpenCV could be a future solution for sending over higher quality video at a lower latency providing a much better experience and enabling the viewer to play faster paced games since the latency would be lower.

Testing equipment: In the future I would love to acquire more controllers to further test compatibility across many different platforms. This would make sure that a wider band of users can use their controller to enjoy games with their friends while not having to spend any money on a controller that is marked as compatible.

Adding additional features: Adding features like Recording and Custom key mapping would greatly improve the overall user experience since the software would provide more features without having to download additional software and improving the ease of use in general.

User interface: Since my project did not have a user interface, this would make sure that a bigger amount of people would be able to use the software even though they are not as technically literate. It would give the software more recognition and improve the success of the project.

# References

**Report**

[1] Brown M.(2018, April 24). How to use Xbox One Mixer controller sharing [Online]

(URL: <https://www.windowscentral.com/how-use-xbox-one-mixer-controller-sharing>) (Accessed 2022 December 2)

[2] Shinners P.(2000, January 1). Pygame documentation [Online]

(URL: <https://www.pygame.org/docs/>) (Accessed 2022 December 14)

**Website**

[3] ChaosTricks, (2021, July 11). Chaos Tricks [Online]

(URL: <https://chaostricks.com>) (Accessed 2022 December 2)

[4] Danny (2015 July 7). Discord.py API Reference [Online]

(URL: <https://discordpy.readthedocs.io/en/stable/api.html>) (Accessed 2022 December 2)

[5] Unknown (2022 January 1). Pynput Package Documentation [Online]

(URL: <https://pynput.readthedocs.io/en/latest/>) (Accessed 2023 January 10)

[6] Unknown (2011 May 2) socket – Low level networking interface [Online]

(URL: <https://docs.python.org/3/library/socket.html>) (Accessed 2023 April 10)

# Appendices

I believe no appendices took part in my project.