A picture containing text

Description automatically generated

Computer Games Development CW208

Technical Design Document

Year IV

Sasa Kuzmanovic

C00249246

|  |
| --- |
| 15/12/2022 |

****

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

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

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

Contents

# 

Contents

[Introduction 4](#_Toc133413804)

[Project Goals 4](#_Toc133413805)

[System Architecture 4](#_Toc133413806)

[Technical Design 5](#_Toc133413807)

[Step 1: Reading chat 5](#_Toc133413808)

[Step 2: Establish a peer-to-peer connection 6](#_Toc133413809)

[Step 3: Capturing joystick input 7](#_Toc133413810)

[Step 4: Accept controls 7](#_Toc133413811)

[Step 5: Convert messages into Input 8](#_Toc133413812)

[References 9](#_Toc133413813)

# Introduction

The Discord Controller Sharing Project is a software solution that allows users to share their game controllers on Discord and play games together remotely. The project is developed using Python programming language and utilizes various libraries such as pygame for capturing joystick input, pynput for simulating keyboard inputs, discord.py for reading chat messages from Discord, and Python sockets for peer-to-peer communication.

# Project Goals

The primary goal of the Discord Controller Sharing Project is to enable users to share their game controllers on Discord and play games together remotely. The project aims to provide a seamless and interactive gaming experience where users can control games using their own game controllers while watching a video stream of the game on Discord. The key objectives of the project include:

* Capturing joystick input: The project needs to capture input from various game controllers, such as gamepad or joystick, connected to the user's computer and transmit the input to the game being played on the host's computer.
* Simulating keyboard inputs: The project needs to simulate keyboard inputs based on the joystick input captured, so that the game on the host's computer can receive the corresponding controls.
* Reading chat messages: The project needs to read chat messages from Discord, interpret the messages into game controller input, and transmit it to the host's computer for processing.
* Peer-to-peer communication: The project needs to establish a peer-to-peer connection between the host and clients for seamless communication of controller inputs.

# System Architecture

The Discord Controller Sharing Project consists of two main components: the host component and the client (Viewer) component. The client component is responsible for capturing the video and streaming it, simulating keyboard inputs and communicating with the client component. The client component is responsible detecting joystick presses, converting them into strings and sending them to the host (Streamer)

**The system architecture can be divided into the following major components:**

* Joystick Input: The pygame library is used to capture input from game controllers connected to the host's computer. The input is processed to determine the controls being pressed, such as buttons or axis movements.
* Keyboard Input: The pynput library is used to simulate keyboard inputs based on the joystick input captured. The simulated inputs are sent to the game on the host's computer to control the game.
* Discord Chat Reader: The discord.py library is used to read chat messages from Discord. The messages are parsed and interpreted into game controller input based on predefined commands or mappings.
* Peer-to-peer Communication: Python sockets are used to establish a peer-to-peer connection between the host and clients for communication of controller inputs. The host and clients communicate over the socket TCP connection to transmit and receive data such as controller inputs which can later be processed into keyboard inputs.

# Technical Design

The implementation of the Discord Controller Sharing Project involves several key steps:

## Step 1: Reading chat

The discord.py library is used to read chat messages from Discord. These chat messages will allow interaction with the software. The bot is able to respond to certain key messages like Hello and Play. The discord bot will check if this discord user has a role “Viewer” that will grant him the permission to establish a connection with the streamer. Not having that specific role will remove the ability to control the game and establish a connection with the streamer.

*Person without the role trying to start controlling streamers game.*

Graphical user interface, text, application, website

Description automatically generated

*Person with the correct role communicating with the bot.*

A screenshot of a computer screen

Description automatically generated with low confidence

*on\_message() method that allows reading chat messages from discord and authorises anyone with a set role.*

Graphical user interface, text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

## Step 2: Establish a peer-to-peer connection

Python sockets are used to establish a peer-to-peer connection between the host and clients for communication of controller input. The host and clients use a TCP connection to transmit and receive data such as controller inputs. The host acts as the server and the clients act as the clients in the socket communication. The socket connection is used to capture button presses of the controller and send them over to the streamers machine, which will then interpret them as keyboard presses and control the game.

*Initialising a socket for a TCP connection.*

Text

Description automatically generated

*Establishing a connection with the streamer.*

Text

Description automatically generated

*Output when a connection with a viewer is established.*



## Step 3: Capturing joystick input

The pygame library is used to capture input from game controllers connected to the host's computer. The library provides functions to detect and read input from various game controllers, such as buttons, axes, and hats. Pressing a certain button will trigger a sendInput function that will send data across a string that will later be used to recognise which button has been pressed and which button to press on a keyboard to mimic viewer’s input. On a button press (For example Right on the D-Pad) it will send a string “d”. This will be recognised by the other side as D key press. When a button is released a string “dr” will be sent over to the Streamer to indicate that the control has been released.

*Joystick D-Pad Right pressed.*

Text

Description automatically generated

*Code for checking if the button was pressed.*

Text

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidence

## Step 4: Accept controls

Accepting the controls is also a part of the peer-to-peer connection and this will allow the input that was sent from the client to the host to be interpreted and used to provide input for the game. waitForInput() function creates a socket (TCP) on the host side of the script that will be listening for any data sent over. We bind the socket to a dedicated port, so it can listen for data. When a connection is accepted, this data will then be decoded and forwarded into the checkForContents function which will decide how do use this data.

*WaitForInput method that accepts connections and data.*

*Text

Description automatically generated*

## Step 5: Convert messages into Input

When the data is received, we need to read what the data says and convert that into input. To do that we will use pynput library that will allow us to press buttons whenever we need them to be pressed. Following the example from before, when Right is pressed on the controller it sends “d” in a message, which will get passed into a checkForContents method to check for binded buttons. When a match is found inside the function, the correct button will be pressed on the keyboard. When a button is released another message is being checked and using pynput, the button that was previously pressed will now be released.

*Checking for string contents. Pressing and releasing specific button.*

*Text

Description automatically generated*

This will create input in the game and as long as the right arrow button is being held the d key on the Hosts computer will be pressed down and continue moving to the right until the key on the client side is not released and the return message is sent.

This sort of a peer-to-peer connection creates a low latency environment that improves the user experience and enables players to share their gameplay with their friends and colleagues.

**Sequence Diagram**

Diagram

Description automatically generated with medium confidence

# 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)