CPSC 359 Assignment 4 Report

Abdalla ElDoumani - 30187890

Fall 2024

1 Description

The `main.c` file used the `uart_puthex` function to transmit button states as hexadecimal values. Each button press or release triggers a 16-bit integer representation of the controller's state. This integer is passed to `uart_puthex`, which sends the hexadecimal representation directly over UART.

On the host side, the Python game listens to the UART port using the `serial` library. The `read_controller_input` function retrieves the transmitted hexadecimal value, decodes it into its binary representation, and maps the bits to their respective button actions specified in the assignment. The directional buttons control tile movements, button B resets the game, Start starts a new game, button Y solves the game, and Select ends the game.

```
// Loop forever, reading from the SNES controller 30 times per second
while (1) {
    // Read data from the SNES controller
    data = get_SNES();

    // If the state of the controller has changed
    if (data != currentState) {

        // Send the button state
        uart_puthex(data);

        // Update the current state
        currentState = data;
    }

    // Delay 1/30th of a second
    microsecond_delay(33333);
}
```

```
def read_controller_input():
"""Read and decode button states from the UART."""
if ser.in_waiting > 0:
    try:
        data = ser.readline().decode().strip()
        print(f"Data: {data}")
        return int(data, 16)
    except ValueError:
        print(f"Invalid UART Data: {data}")
return None

# Read SNES controller input
button_state = read_controller_input()
if button_state is not None:
    print(f"Button State: {button_state:016b}")
    if button_state & (1 << 4):  # Pad UP
        slideTo = UP
    elif button_state & (1 << 5):  # Pad DOWN
        slideTo = DOWN
    elif button_state & (1 << 6):  # Pad LEFT
        slideTo = LEFT
    elif button_state & (1 << 7):  # Pad RIGHT
        slideTo = RIGHT
    elif button_state & (1 << 0):  # B Button
        resetAnimation(mainBoard, allMoves)
        allMoves = []
    elif button_state & (1 << 3):  # Start Button
        mainBoard, solutionSeq = generateNewPuzzle(80)
        allMoves = []
    elif button_state & (1 << 1):  # Y Button
        resetAnimation(mainBoard, solutionSeq + allMoves)
        allMoves = []
    elif button_state & (1 << 2):  # Select Button
        terminate()
```