2048 on Pi

5/8/2023

Project Summary

The goal of this project was to construct a playable version of the game 2048 in the Raspberry Pi's web browser.

Playing 2048

- The game is played on a 4x4 grid of tiles, each with their own value. The player can 'slide' the tiles in a cardinal direction.

- Every time you make a move, a new tile will appear on the board, either as a 2 or a 4, at a random position.

- If two tiles with the same number touch each other while moving, they will merge into a new tile with double the number. For example, two adjacent tiles with the number 2 will merge to form a single tile with the number 4.

- The goal of the game is to create a tile with the number 2048 by merging tiles with the same number.

- The game ends when there are no more moves to make on the board, either because all the tiles have been filled with numbers or because there are no more tiles that can be merged.

Included Features

- Joystick Controls

  Seamlessly move tiles via moving the joystick. Press in on the Z-Axis for other input.

- Keyboard Controls

  Alternatively, move tiles using a connected keyboard (WASD). Press Space for other input.

- Browser GUI

  The game is played in a browser with a CSS interface.

- High Score Display

  Watch your score climb high and try to overcome the best previous players and their score. Both scores are displayed on the external LCD.

- Randomization

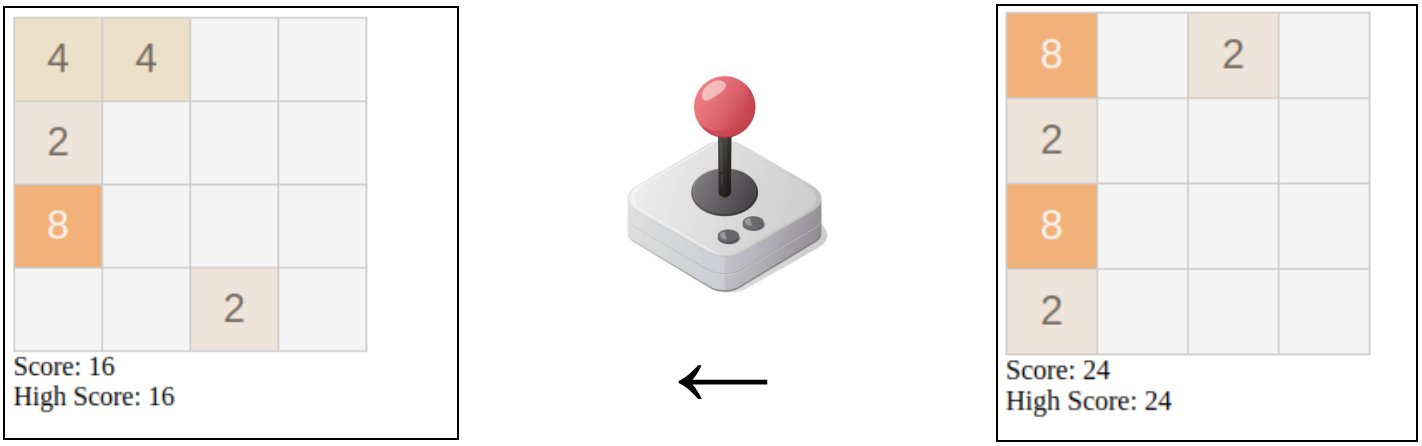
  Every game is different with randomization.

Potential Improvements and Shortcomings

- Some versions of 2048 include an undo button, but this version does not
- The game does not include any music or sound effects
- The only “multiplayer” functionality that might exist is trying to beat another player’s high score

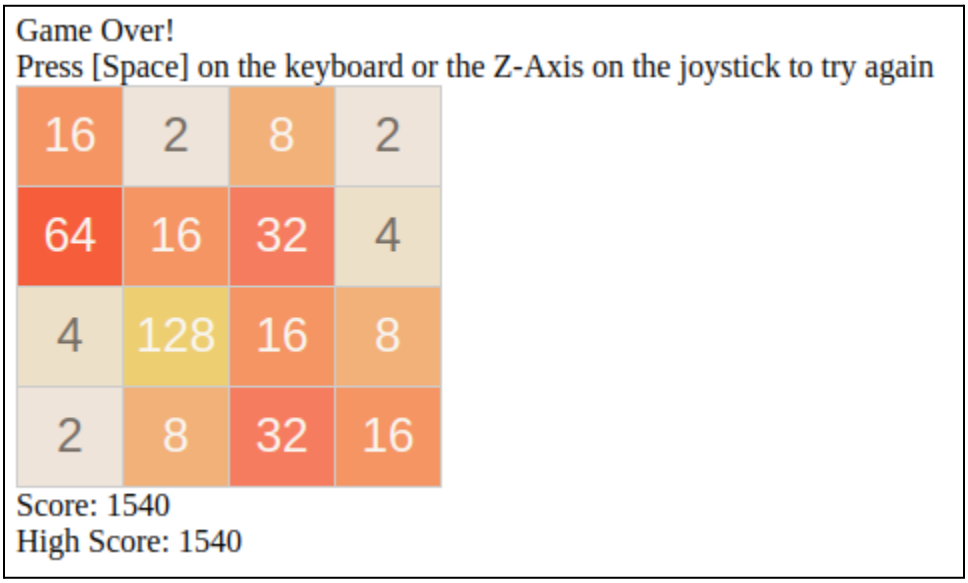
- The delay between inputs and when the display is rendered is incongruent to the otherwise fast-paced gameplay of 2048
- Polling the joystick and buffering keyboard inputs can be very slow
- The wires connected to the joystick can get loose easily, interrupting gameplay

Operation and Usage

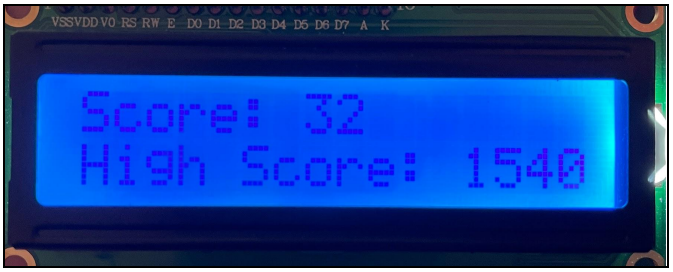


(Move Grid Left)

The diagram above illustrates the result of sliding tiles. The 2 at the bottom of the screen slides to the left, colliding with the left wall. The two 4s at the top slide to the left and collide; since they have the same value, they merge together, forming a new 8 tile. Afterwards, another tile is randomly generated (the 2 at the top of the grid).



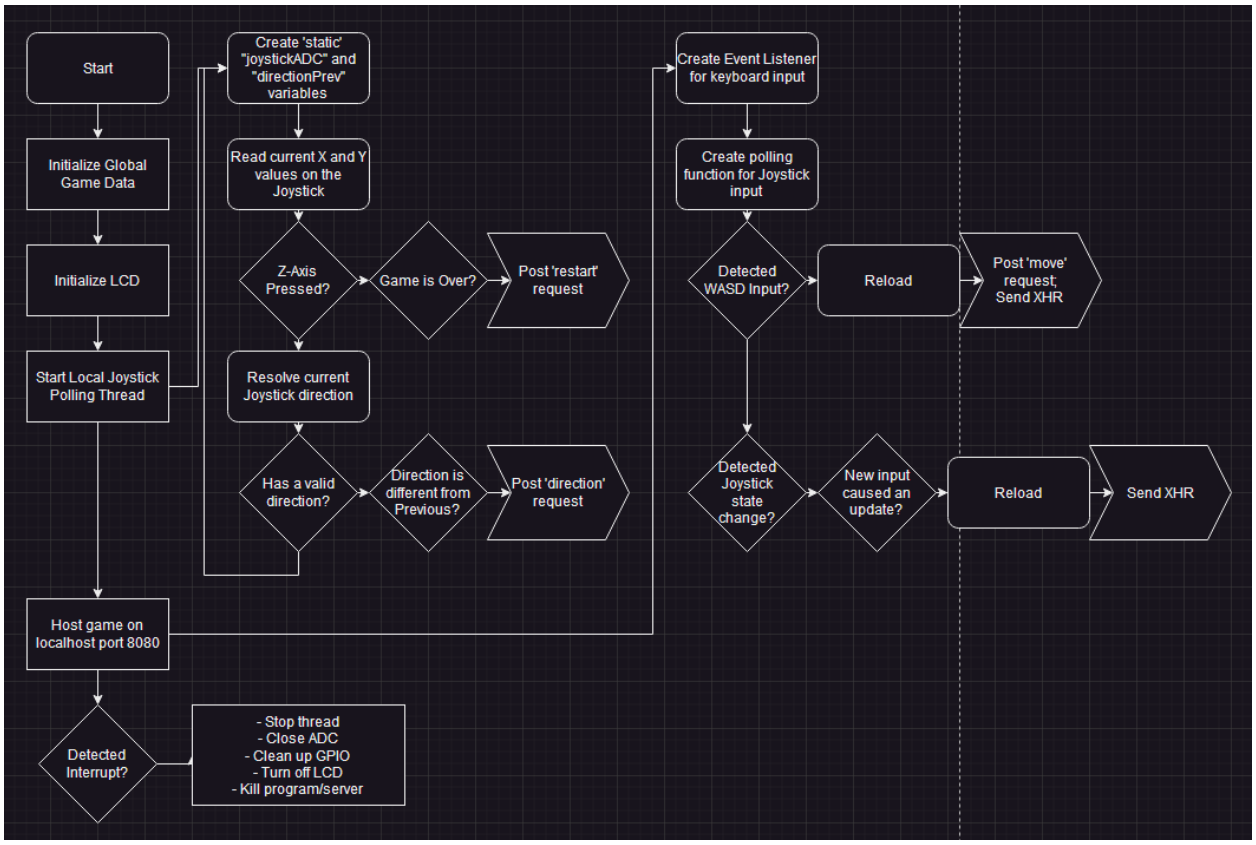
If the player runs out of valid moves, they will be prompted to press the joystick's Z-Axis or Space on the keyboard to start a new game, resetting the grid and their score.



The score and high score values are also displayed on the external LCD.

Programming and Design Details

The Game Loop



Program Essentials Guide
(Code snippets and explanations)

1. Reading the Joystick Values:
    - The X and Y axes are read independently from the joystickADC object defined as an attribute in the handle_joystick_input function.

```
xAxisVal = handle_joystick_input.joystickADC.analogRead(ADC_CHANNEL_X)
yAxisVal = handle_joystick_input.joystickADC.analogRead(ADC_CHANNEL_Y)
```

    - The current direction of the joystick is determined by checking if the X or Y values exceed lower or upper thresholds.
    - If the current direction is different from the recorded previous direction (and the direction is not None) then an input will be sent.

```
response = requests.get(f'http://localhost:8080/joystick_move?direction={direction}')
```

    - The function handling joystick inputs runs on a separate thread.

2. Updating the Grid and Game Data
    - The following is a snippet of the function which generates the grid (4x4 ints):

```
grid = [[0 for _ in range(GRID_WIDTH)] for _ in
range(GRID_WIDTH)]
```

    - Critical game data is stored in a global container:

```
GAME_STATE = {'grid':new_game(), 'score':0,
'high_score':0}
```

    - A function is used to spawn a tile at a random location

```
#Get random position for the new tile (target space must
be empty)
empty_tiles = [(i, j) for i in range(GRID_WIDTH) for j in
range(GRID_WIDTH) if grid[i][j] == 0]
i, j = random.choice(empty_tiles)
#Add a 2 (or rarely a 4) to the target location
grid[i][j] = (2 if random.random() < 0.9 else 4)
```

    - There are 4 functions for sliding the grid in a different direction, but only 1 function for sliding a row (merge_tiles). The move_direction functions transform the grid to work with the merge_tiles function, and then the resulting grid is transformed back to give the expected result. The following snippet is from move_right:

```
for row in grid:
    reversed_row = row[::-1]
    merged_row, row_score = merge_tiles(reversed_row)
    new_grid.append(merged_row[::-1])
    score += row_score
```

3. Scores on LCD
    - The program interfaces with the LCD with support from the LCD1602.py file.
    - The LCD calls a function whenever the scores are updated.

```
#Display LCD Score
LCD1602.write(
  LCD1602.LCD_CHAR_POSITION_1,
  LCD1602.LCD_LINE_NUM_1,
  "Score: "+str(min(DISPLAY_SCORE_MAX,
GAME_STATE['score']))
  )
```

4. Handling Incoming Inputs
    - Keyboard input is handled via an EventListener

```
document.addEventListener('keydown', function(event) {
if(['w', 'a', 's',
'd'].includes(event.key.toLowerCase())) {
    event.preventDefault();
    let xhr = new XMLHttpRequest();
    xhr.open('POST', '/move');
    ...
    xhr.send('direction=' + event.key.toLowerCase());
    }
```

5. Joystick Input
    - Joystick input is handled by polling continuously

```
    function pollJoystickInput() {
        let xhr = new XMLHttpRequest();
        xhr.open('GET', '/joystick_move');
        xhr.onreadystatechange = function() {

            ...

        var myArr = JSON.parse(xhr.responseText);
        let grid = JSON.stringify(myArr["grid"]);

            ...

        if (oldGridString != grid) {
            localStorage.setItem('oldGridString', grid);
            location.reload();
            }

       setTimeout(pollJoystickInput,
TIMEOUT_DELAY_200_MS);
       }
```

Key Wiring Connections

- The wiring for the Joystick is taken from the FNK0024 manual. A modified version including the LCD is included below. Also included is the ADS7830 ADC.

Joystick

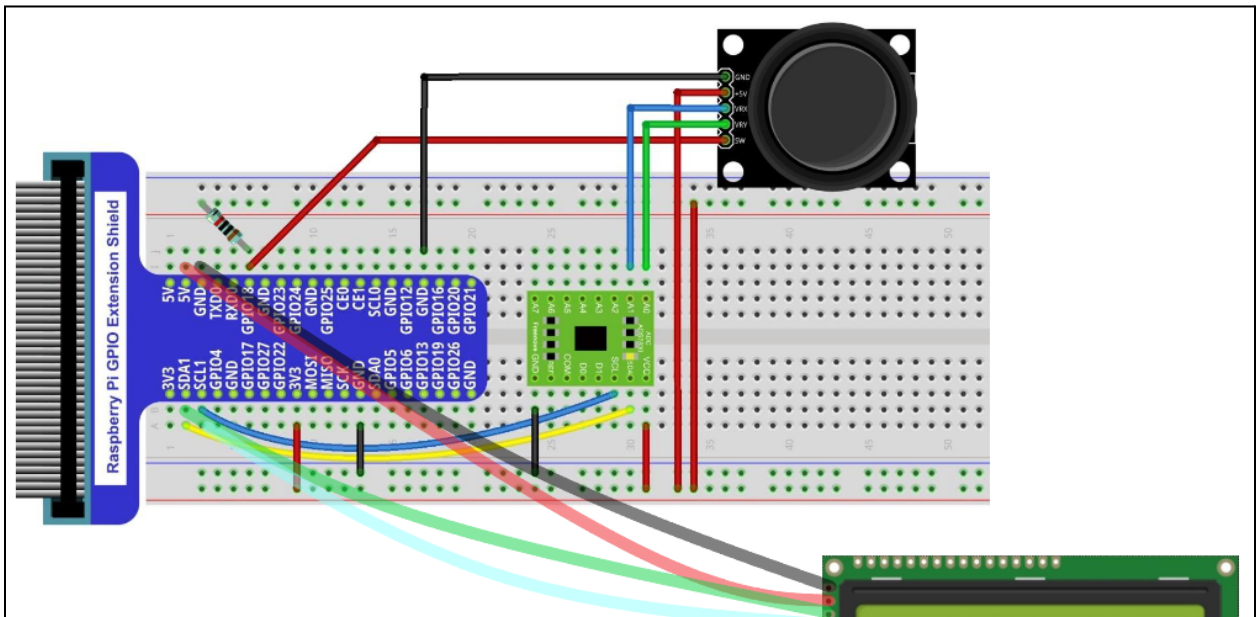
1. GND	→	GND	
2. +5V	→	+5V	(VCC)
3. VRX	→	A1	(ADS7830 ADC)
4. VRY	→	A0	(ADS7830 ADC)
5. SW	→	GPIO 18	

ADS7830 ADC

1. SDA	→	SDA1
2. SCL	→	SCL1

LCD

1. GND	→	GND
2. +5V	→	+5V
3. SDA	→	SDA1
4. SCL	→	SCL1



Citations

- OpenAI
  (Some CSS/interface code + miscellaneous auto code completion)

- LCD1602.py (from Lab 9)
  (To interface with the LCD)

- FNK0024 Manual
  (For joystick wiring)