2048 in PuTTY

5/3/2023

Project Summary

The goal of this project was to construct a playable version of the game 2048 in the PuTTY terminal.

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

- Toggleable Audio

  Plays immersive sound effects when merging tiles or after losing a game. Audio can be turned on or off at any time via a Switch Button. The status of the speaker is indicated via an RGB LED.

- Joystick Controls

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

- Score Display

  Watch your score climb high on the four 7-segment displays.

- High Score Display

  Try and overcome the best previous players and their score, which is displayed on the LCD.

- Game Over Indicator

  Know defeat when you see the flashing of the LEDs.

- Light and Dark Modes

  Protect your eyes with the light and dark modes. They will switch automatically based on the light, but can be overridden through a potentiometer.

- Randomization

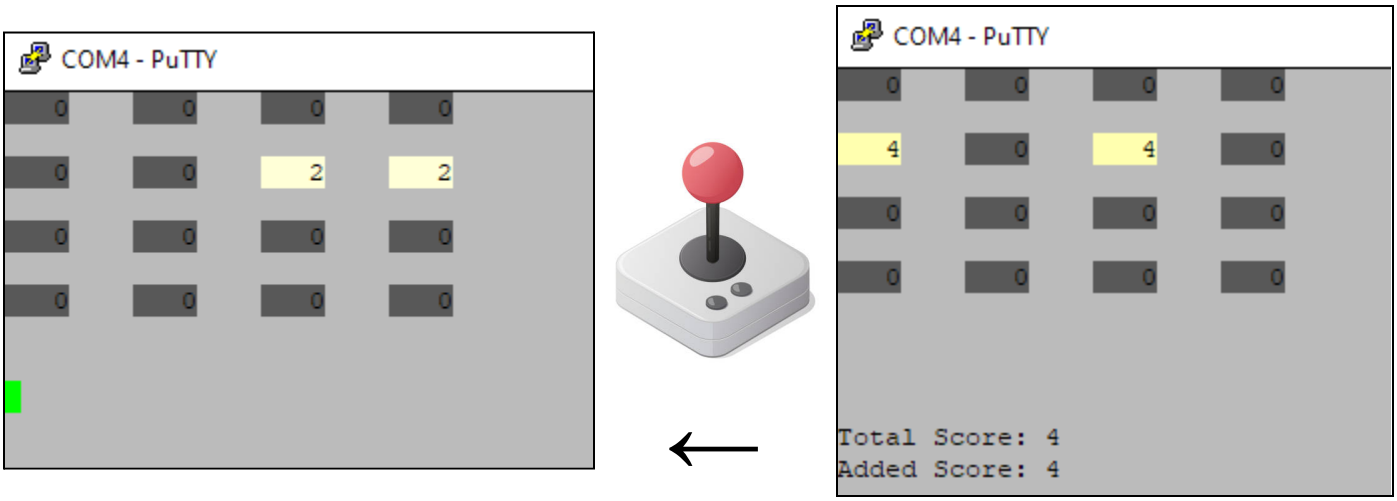
  Every game is different with randomization.

Potential Improvements and Shortcomings

- Some versions of 2048 include an undo button, but this version does not
- The game does not include any music, despite having 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
- The wires connected to the joystick can get loose easily, interrupting gameplay
- The 7-segment score display is limited to 4 digits (a maximum value of 9999)

Operation and Usage



(Move Grid Left)

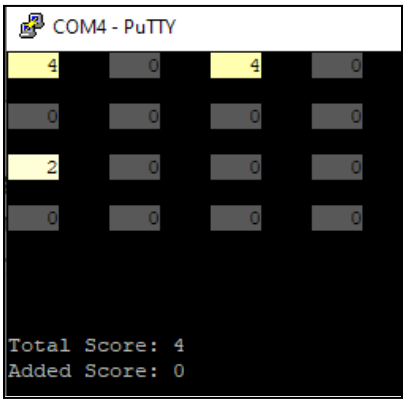
The diagram above illustrates the result of sliding and merging the first 2 randomly generated tiles. They collide at the left edge of the screen, and merge to become a new 4 tile. Additionally, a new 4 tile is randomly generated where one of the previous tiles used to be.

The 7-segments are updated to display the player's new score.

A sound effect was also played, indicating that tiles were merged successfully.

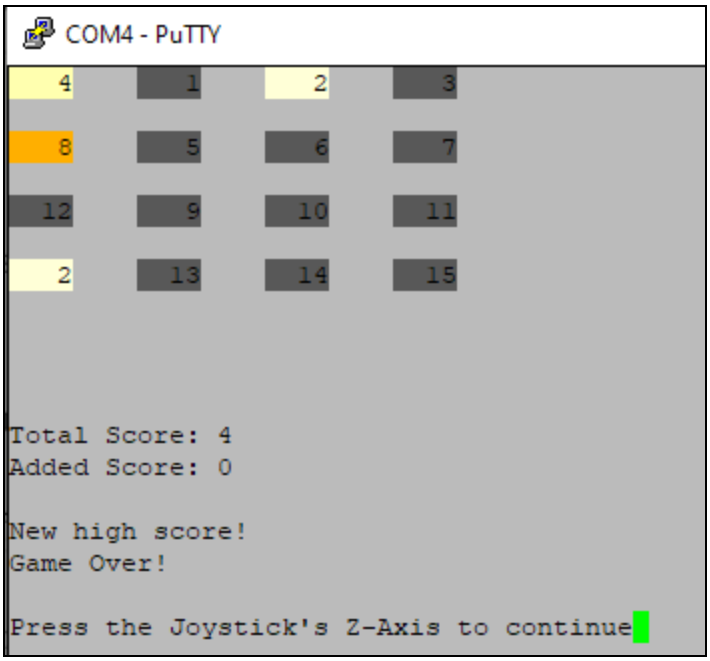
If the RGB LED displays red, it indicates that audio is enabled.

The rightmost switch button (SW5) can be used to toggle the audio on/off at any time.



If the light sensor detects low light, the game will automatically be displayed in dark mode after the next move. If high light is detected, the game will return to light mode.

The automatic settings can be overridden to force either light/dark mode by turning the potentiometer adjacent to the light sensor.



Note that the game was forcefully put in a state with invalid moves using debugging tools in this example.

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

When the game is over, the row of 8 LEDs will begin to flash, and a small tune will play.

If a new high score is reached, it will now be displayed on the LCD.

Programming and Design Details

The Game Loop

*Missing light/dark mode decision, should go before the grid is displayed

Program Essentials Guide

(Code snippets and explanations)

1. Reading the Joystick Values:
    - The X and Y axes are read independently from AD1 and AD0 respectively. Both use Channel 3.

```
int joystick_get_x_axis() {
  return ad1conv(AD_CHANNEL_JOYSTICK_X);
  }
```

    - An enum is used to store the direction (Up, Down, Left, Right, Unknown)
    - The current direction of the joystick is determined by checking if the X or Y values exceed lower or upper thresholds. If no threshold is reached, the direction will be 'Unknown'.
2. Audio
    - Audio can be toggled via a flag set within an interrupt

```
//SW5 Pressed, Set Audio Toggle Flag
if (PIFH & PORTH_SW5_BITMASK) {
  //Disable main loop
  G_SW5Pressed = _TRUE_;
  clearFlag |= PORTH_SW5_BITMASK;
  }
```

    - Sound effects are played via a function that takes a pitch value and duration. The sounds in this game use pre-defined note pitches and durations, e.g.

```
if (soundEnabled) {
  play_sound(NOTE_G, NOTE_SIXTEENTH);
  play_sound(NOTE_D, NOTE_SIXTEENTH);
  play_sound(NOTE_B, NOTE_EIGHTH);
  }
```

    - The audio indicator LED is updated via the motor functions

```
//Toggle LED to indicate sound is enabled/disabled
motor4(soundEnabled ? MOTOR_SPEED_ENABLED :
MOTOR_SPEED_DISABLED);
```

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

```
  int** grid = (int**)malloc(GRID_LENGTH * sizeof(int*));
  for (i = 0; i < GRID_LENGTH ; i++) {
    grid[i] = (int*)malloc(GRID_LENGTH * sizeof(int));
    }
```

    - A function is used to spawn a tile at a random location (if one is available).

```
    // Choose a random empty cell
    index = rand() % count;
    i = available_cells[index][CELL_INDEX_ROW];
    j = available_cells[index][CELL_INDEX_COL];
    // Spawn a new tile with a value of 2 (90% chance) or
4 (10% chance)
    value = (rand() % 10 == 0) ? 4 : 2;
    grid[i][j] = value;
```

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

```
  for (i = 0; i < GRID_LENGTH; i++) {
    // Reverse the row before sliding it to the left
    int temp_row[GRID_LENGTH];
    for (j = 0; j < GRID_LENGTH; j++) {
      temp_row[j] = grid[i][GRID_INDEX_MAX - j];
    }
    // Slide the reversed row to the left
    score = slide_row_left(temp_row);
      ...
    // Reverse the row back and store it in the grid
    for (j = 0; j < GRID_LENGTH; j++) {
      grid[i][GRID_INDEX_MAX - j] = temp_row[j];
```

4. Light/Dark Mode + ANSI Colors
    - The text, background, and tile colors are set using ANSI values. Some examples include…

```
#define ANSI_TEXT_COL_BLACK "\033[30m"
#define ANSI_TEXT_COL_WHITE "\033[37m"
      ...
void print_tile_color(int value) {
  switch (value) {
  case 2:     print("\033[48;5;230m"); break; //Light
Yellow
  case 4:     print("\033[48;5;229m"); break; //Yellow
```

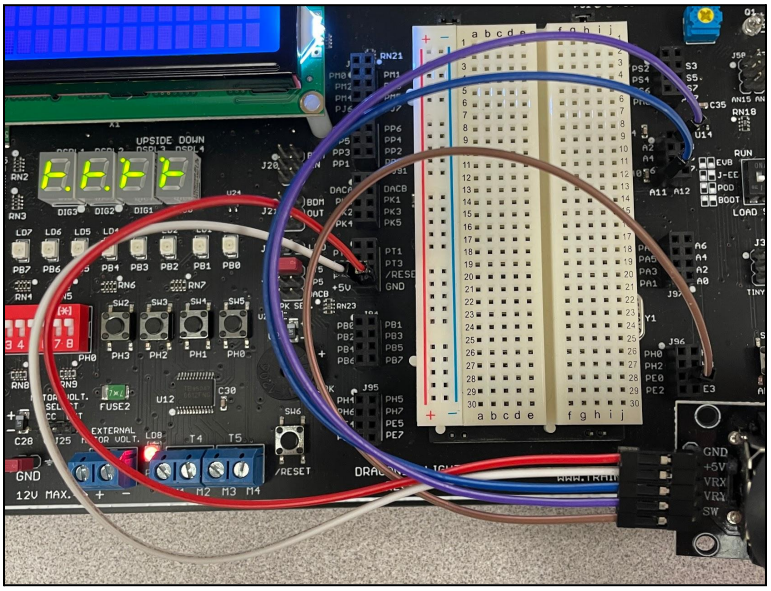
    - ANSI codes are also used for ‘resetting’ the cursor and grid, and rendering text in the correct position.
    - The function below checks the potentiometer/light sensor to print the grid with the light or dark mode.

```
  static int valPotentiometerPrev = 0;
  int valPotentiometer;
  bool use_light_mode = _FALSE_;
  //Check for potentiometer override
  valPotentiometer = ad0conv(AD_CHANNEL_POTENTIOMETER);
  //Force Dark Mode
  if (valPotentiometer < AD_POTENTIOMETER_THRESHOLD_DARK)
{
    use_light_mode = _FALSE_;
    }
  //Force Light Mode
  else if (valPotentiometer >
AD_POTENTIOMETER_THRESHOLD_LIGHT) {
    use_light_mode = _TRUE_;
    }
  //Use light sensor value
  else {
    use_light_mode = (ad0conv(AD_CHANNEL_LIGHT_SENSOR) >
AD_LIGHT_SENSOR_THRESHOLD);
    }
  if (use_light_mode) { print_grid_light(grid); }
  else { print_grid_dark(grid); }
  valPotentiometerPrev = valPotentiometer;
```

Key Wiring Connections

- The only external wiring in this design is for the Joystick, which has the following 5 connections:

1. GND	→	GND	
2. +5V	→	+5V	(VCC)
3. VRX	→	A11	(AD1 Channel 3)
4. VRY	→	A3	(AD0 Channel 3)
5. SW	→	PE1	(Interrupt 25 with PTH mask 0x08)



Citations

- https://stackoverflow.com/questions/15062967/what-are-the-max-and-min-in-c-define-function
  (For MIN and MAX functional macros)

- OpenAI
  (ANSI color codes and functions + miscellaneous auto code completion)