CPSC 359 – Spring 2019

Assignment Project

Assignment 3 (8%)
SNES Controller Device Driver
Due: Jun 10th @ 11:59pm (midnight)

Objective: Build a simple device driver for a SNES controller. The driver will be used in the next assignment as the primary input device for your interactive game.

Deliverables:

1. Print creator name(s) at the beginning.
2. Print “Please press a button…”.
3. Wait for user input.
4. If user presses a button, print a message on the console indicating the button pressed.
5. Loop back to step 2 if any key other than START is pressed.
    a. You might assume that one key is pressed at a time.
    b. Previous key needs to be released in order to read a new one, including the same button being pressed.
6. Pressing the “START” button will end the program displaying an exit message.

Example session:

```
Created by: John Smith and Sarah Smith

Please press a button…          (User Presses Joy-pad RIGHT button)

You have pressed Joy-pad RIGHT

Please press a button…          (User Presses Y button)

You have pressed Y

Please press a button…          (User Presses START button)

Program is terminating…
```

1. Use at least the following functions:
    a. Init_GPIO: the function initializes a GPIO line, the line number and function code must be passed as parameters. The function needs to be general.
    b. Write_Latch: write a bit to the SNES latch line.
    c. Write_Clock: writes a bit to the SNES clock line.

d. Read_Data: reads a bit from the SNES data line.
e. Read_SNES: main SNES function that reads input (buttons pressed) from a SNES controller. Returns the code of a pressed button in a register.

2. Read the submission instructions on the last page.

Grading:

1.	Display creator names & messages	1
2.	Correctly reading/printing buttons	10
3.	Using functions	5
4.	Code efficiency	5
5.	Loop back (not "START")	2
6.	Well documented code	2
	Total	25 points

Assignment 4 (22%)
Raspberry Pi Video Game
Due Jun 17th @ 11:59pm (midnight)

Objective: The objective of this assignment is to expose you to video programming using C on Raspberry Pi 3.

Game Objective: Implement a video game that requires the player to fight a number of moving and attacking enemy objects. The player uses an attacking object to attack and terminate the enemy. The objective of the game is to terminate all enemy objects before they terminate the player's object.

You must implement the following minimum functionalities to get the full mark. Beyond that, you are free to add more features to your game.

Game Logic

- The ***game environment*** is a finite 2D *n* x *m* grid.
- A ***game map*** is an instance of the game environment (for a value of *n & m* ≥ *20*)
  - Each cell of the grid is either a *space tile or an obstacle tile.*
- A ***game state*** is a representation of the game as it is being played, and contains things that you need to keep track of:
  - An instance of a *game map.*
  - *Player's position* on the *game map* (initialized to a starting position).
  - *Enemies' position* on the *game map* (initialized to a starting position).

    - *Laser bullets' positions whenever exist.*
    - The *score* collected by the player (initialized to zero).
    - Life meter (initialized to a positive value).
    - A *win and lose condition* flags.

- **Enemies:**
    - 3 types of enemies must be present in the game: Pawns, Knights and Queens. Types are differentiated in hardness of killing.
    - The game must contain at least 10 pawns, 5 knights, and 2 queens. If you choose to use more objects keep a similar proportion for each type. That is pawns must be the majority with some nights and a few queens

- The ***game transitions*** into a new state:
    - The user controls a ship to move around the screen and shoot at the enemy objects; the actual movement patterns is left for you.
    - The enemy objects march back and forth left and right avoiding being shot by the user ship. They shoot back at the user ship. The way they move on the screen is up to you to decide.
    - The user collects score points by terminating (killing) enemy objects.
        - **Pawns**: must be the easiest to kill and the user gets 5 score points for terminating a pawn.
        - **Knights**: are harder to kill than pawns; the user gets 10 points for terminating a knight
        - **Queens**: are the hardest to kill but are the most rewarding. A queen earns the user 100 points
    - If the user ship is hit by an enemy bullet, the user loses from the life meter, If the life meter becomes 0, the user loses and the game ends. The initial life meter is some positive value.
- All objects may be permitted to move outside the screen, but you have to make sure that they can come back. You may also wish to limit movements to the visible screen.
- The game must have obstacles scattered around the screen. The user ship can hide behind such obstacles. However, obstacles shrink when they are hit by bullets (from either party) until they disappear.
- ***Game ends*** when:
    - The user terminates all enemy objects (user wins),
    - The enemy terminates the user object (user loses), or
    - The user decides to quit.

The game is over when either the *win* or *lose condition* flags is set in the game state.

Game Interface

- Main Menu Screen
  - The Main Menu interface is drawn to the screen
  - Game title is drawn somewhere on the screen
  - Creator name(s) drawn somewhere on the screen
  - Menu options labeled “Start Game” and “Quit Game”
  - A visual indicator of which option is currently selected

- The player uses the SNES controller to interact with the menu
  - Select between options using Up and Down on the D-Pad
  - Activate a menu item using the ***A*** button
  - Activating Start Game will transition to the Game Screen
  - Activating Quit Game will clear the screen and exit the game (Black screen is acceptable).

- The player uses the SNES controller to interact with the game
  - Pressing up, down, left or right on the D-Pad will attempt a move action for the user ship
  - Pressing the A button will shoot a laser bullet
  - Pressing the Start button will show a menu with 3 options:
    - Pause/Resume game
    - New game
    - Quit
  - If the win condition or lose condition flags are set
    - Pressing any button will restart to the game

Grading:

1. **Game Screen:**

a. ***Main Menu Screen* (5 marks)**

Draw game title and creator names	1
Draw menu options and option selector	1
Select between menu options using up / down on D-Pad	1
Press A button with Start Game selected to start game	1
Press A button with Quit Game selected to exit game	1

b. Draw current game state (23 marks)

All objects are drawn according to interface specifications.	5
Enemies can move & fire.	4
Objects return if got outside the screen (or limit movement)	2
Enemies disappear if hit (according to its hardness).	3
Obstacles shrink if hit by bullets and disappear.	3
Score is drawn somewhere on the screen.	1
Life meter is drawn somewhere on the screen	1
Number of points collected is as specified.	2
Game Won message drawn on win condition	1
Game Lost message drawn on lose condition	1

c. Draw game menu (5 marks)

Filled box with border in center of screen	1
Draw menu options and option selector	2
Erase game menu from screen when closed	2
d. ***<u>Interact with game</u> (8 marks)***	
Use D-Pad to move the ship (if move action is valid)	3
Press A button to fire a bullet.	3
Press Start button to open game menu	1
Press any button to restart the game when game over.	1
e. ***<u>Interact with game menu:</u> (4 marks)***	
Use up / down on D-Pad to change menu selection	1
Press A button on Restart Game; resets the game	1
Press A button on Quit; terminate the game	1
Press A button on Resume; resume the game	1
2. ***<u>Well structured code:</u> (10 marks)***	
Use of functions to generalize repeated procedures	5
Use of structures to represent game state, etc.	5
3. ***<u>Well documented code</u>***	5
<u>Total:</u>	**60**

Programs that do not compile cannot receive more than **5 points**. Programs that compile, but do not implement any of the functionality described above can receive a maximum of **7 points**.

Teams: You may work in teams of up to **<u>four students</u>** in order to complete this project, but you are not required to do so. Peer evaluation in teams may be conducted.

Demonstration & Submission: Submit a .tar.gz file of your entire project directory (including source code, make file, build objects, myProg) to your TA via the appropriate dropbox on Desire2Learn. **<u>Only one submission per group</u>**. You will also need to be available to demonstrate your assignment during the tutorial.

Late Submission Policy: Late submissions will be penalized as follows:
-12.5% for each late day or portion of a day for the first two days
-25% for each additional day or portion of a day after the first two days
Hence, no submissions will be accepted after 5 days (including weekend days) of the announced deadline.

Academic Misconduct: Any similarities between assignments will be further investigated for academic misconduct. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your own original work. Any re-used code of excess of 20 lines must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.