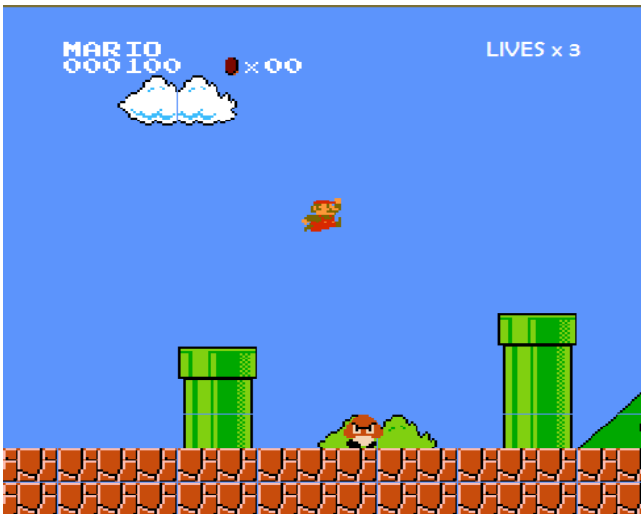
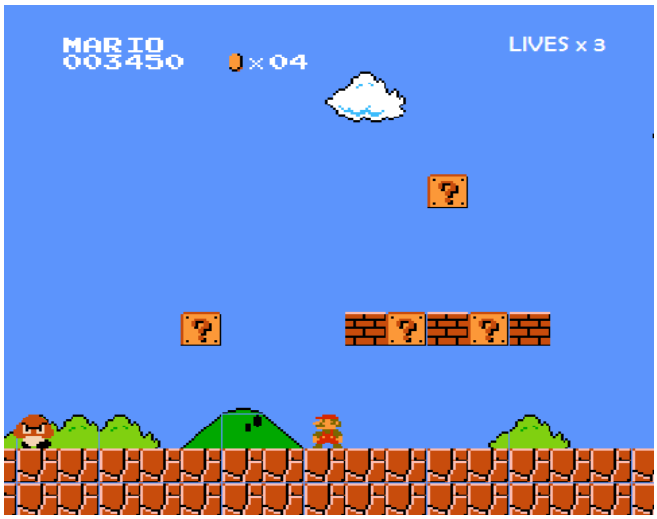
CPSC 359 – Winter 2017
Assignment 4
Raspberry Pi Video Game
75 points (Weight: 16%)
Due: March 31st @ 11:59 PM

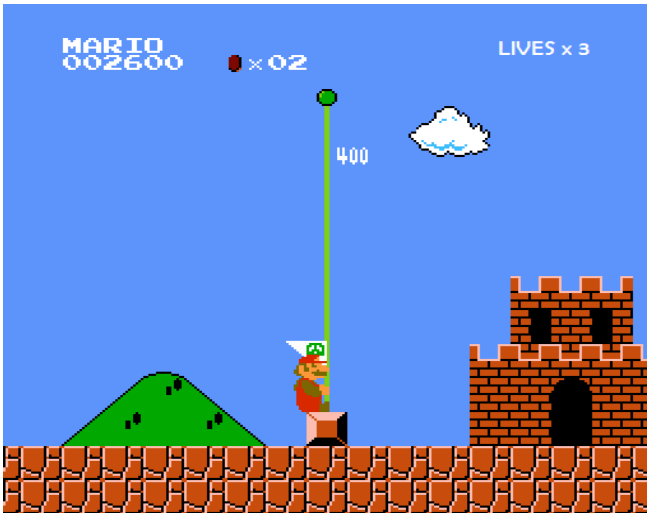
Objective: The objective of this assignment is to expose you to Video programming and interrupt handling on the Raspberry Pi.

Game Objective: To implement the famous Super Mario video game. In this game, Mario travels through a map in order to reach the palace where Princess Peach resides. On his way to the castle, Mario will be faced by several monsters and obstacles which will make the way to the palace challenging. The game is won when Mario reaches the castle overcoming these obstacles.

If you want to refresh your memory on how the original game looks like or benefit from some ideas about the different components of the game, then go to http://www.begamer.com/flash-game/242906/super-mario-original/ and play the game online!!







The ***game environment*** is a finite 2D *n* x *m* grid (not necessarily a square). As Mario moves to the right or left, the view of the map will move accordingly loading new scenes to the game. You can load the new scenes to the game as Mario moves forward piece after piece or you can do it in such way that loads the whole new scene of the game (as Mario reaches the right edge of screen).

- A ***game map*** is an instance of the game environment (for a value of $20 \leq n \leq 25$ *& m* $\geq$ *20*)
  - Specifies the score of the player in terms of coins collected and general score.
  - Specifies the number of lives left (starting with a minimum of 3 lives).
  - There should be at least 3 different stages of the map which Mario will explore moving forward.
  - The objects in the game are: Mario, monsters (min 1), coin blocks (min 3), wood blocks (min 2), green pipes (min 1), and holes (min 1).
  - Mario can move left, right, and he can also jump
  - The game should contain at least 2 different types of monsters.
  - The coin blocks are in midair, Mario has to jump to be able to get the coins. On hit, the block will remain there but with no coin value.
  - The wood blocks are usually located next to the coin blocks. If Mario jumps and hits a wood block from the bottom, then this block will be destroyed. Mario can jump on top of these blocks and walk on top of the wood and coin blocks.
  - The green pipes are just an obstacle in Mario's way; he should jump on top of them to proceed.
- A ***game state*** is a representation of the game as it is being played, and contains:
  - An instance of a *game map.*
  - *Mario and the different tiles positions* on the game map (Initialized to a starting position).
  - The *score and coins* collected by the player (initialized to zero).
  - Number of *lives* left.
  - A *win condition* flag.
  - A *lose condition* flag
- The ***game transitions*** into a new state when the player performs an action
  - Mario jumps into a coin block gaining a coin and score value
  - Mario jumps and hits the top of the monster which removes the monster and increases the score
  - Mario gets hit by a monster or falls off the map into a hole will make Mario lose an available life or set the lose flag.
  - Mario reaches the castle; this will set the win condition flag.
- ***Value-Packs***:
  - You must implement at least one *value-pack* of your choice that adds some feature to the game (such as a pack to giving life to Mario or speeding up the game etc.).
  - You may get extra marks for additional creative *value-packs*.
  - The value packs should appear after about 30 seconds into the game (to do so, you should use interrupts on the time) in a random place. Check https://en.wikipedia.org/wiki/Xorshift for a simple random number generator.

- ***Action***:
  - Move by one cell in one of the up, left and right directions if the action is valid. The move results in Mario's *position* being set to the position of the destination cell.
  - Moving into a *value-pack* tile will result in:
    - The overall score being incremented.
    - The removal of the *value-pack* marking on the destination tile.
    - Application of the feature effect.
- ***Game ends*** when:
  - Mario gets inside the castle (win).
  - Mario's lives become zero (lose).
  - 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

Game Screen

- The current game state is drawn to the screen
  - Represented as a 2D grid of cells
    - All cells in the current game state are drawn to the screen
    - Each cell is at least 32x32 pixels
    - 2D grid should be (roughly) in the center of the screen
  - Different tile types are drawn with a different visual representation
    - ie: Mario, pipes, floor, monsters, coins, wood blocks, etc…
    - Minimally, in color and shape.
  - Score and lives left are also drawn on the screen
    - A label followed by the decimal value for each field
  - If the "Win Condition" flag is set, display a "Game Won" message
  - If the "Lose Condition" flag is set, display a "Game Lost" message
    - Both messages should be prominent (ie, large, middle of screen)
- The player uses the SNES controller to interact with the game
  - Pressing up, left or right on the D-Pad will attempt a move action
  - Pressing the Start button will open a Game Menu
    - With two menu items: Restart Game and Quit
    - Visually display menu option labels and a selector
    - Menu drawn on a filled box with a border in the center of the screen
    - Normal game controls not processed when Game Menu is open

- Pressing Start will close the Game Menu
- Press up and down on D-Pad to select between menu options
- Pressing the ***A*** button activates a menu option
- Activating Restart Game will reset the game to its original state
- Activating Quit will transition to the Main Menu screen

- If the win condition or lose condition flags are set
  - Pressing any button will return to the Main Menu screen.

Grading:

1.	Game Screen		
	a.	Main Menu Screen ***(5 marks)***	
		i. Draw game title and creator names	1
		ii. Draw menu options and option selector	1
		iii. Select between menu options using up / down on D-Pad	1
		iv. Press A button with Start Game selected to start game	1
		v. Press A button with Quit Game selected to exit game	1
	b.	Draw current game state: ***(27 marks)***	
		i. All objects are drawn according to interface specifications.	5
		ii. Mario moves in the available spaces and jumps into blocks	2
		iii. When Mario moves closer to the right (or the end of the screen) , the move will show new scenes from the right (minimum 3 different full scenes) and when he moves left, the old scene will reappear	5
		iv. Monsters disappear when being hit from top	3
		v. Score/Lives are drawn as specified.	4
		vi. Implement value-pack as specified.	4
		vii. Game Won message drawn on win condition	2
		viii. Game Lost message drawn on lose condition	2
	c.	Draw game menu: ***(4 marks)***	
		i. Filled box with border in center of screen	1
		ii. Draw menu options and option selector	1
		iii. Erase game menu from screen when closed	2
	d.	Interact with game: ***(15 marks)***	
		i. Use D-Pad to move Mario	2
		ii. The first value pack will appear after nearly 30 seconds	7
		iii. Press Start button to open game menu	1
		iv. Press any button to return to main menu (game over).	1
		v. The value packs are randomly shown in the Screen	4
	e.	Interact with game menu: ***(4 marks)***	
		i. Use up / down on D-Pad to change menu selection	1
		ii. Press A button on Restart Game; resets the game	1
		iii. Press A button on Quit; Go to Main menu	1
		iv. Press Start button to close game menu	1
2.	APCS compliant functions		3
3.	Well structured code ***(15 marks)***		
	a.	Use of functions to generalize repeated procedures	10
	b.	Use of data structures to represent game state, etc.	5
4.	Well documented code		2
	Total		75

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 are advised to work with **two other students** in class in order to complete the assignment, but you are not required to do so. You and your partner must be in tutorials taught by the same TA. Peer evaluation in teams may be conducted. **It is preferable to keep the same team from Assignment 3.**

Submission: Submit your source code (the *template* folder) to the drop box on D2L. Only one submission per team is required.

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 assignment submissions will be further investigated for potential academic misconduct.

Code sharing with a different group is prohibited. Code sharing includes looking at others' code on paper and on the computer screen. Discussions with other groups can only be carried out at the concept level. While you are encouraged to discuss the assignment with your colleagues, your final submission must be your team's original work. Any re-used code in excess of 10 lines must be acknowledged and cited. Violation of this policy may be considered academic misconduct. If unsure, always check with your instructor or TA.

D2L Marks: Any marks posted on D2L or made available using any other mean are tentative and are subject to change (after posting). They can go UP or DOWN due to necessary corrections.