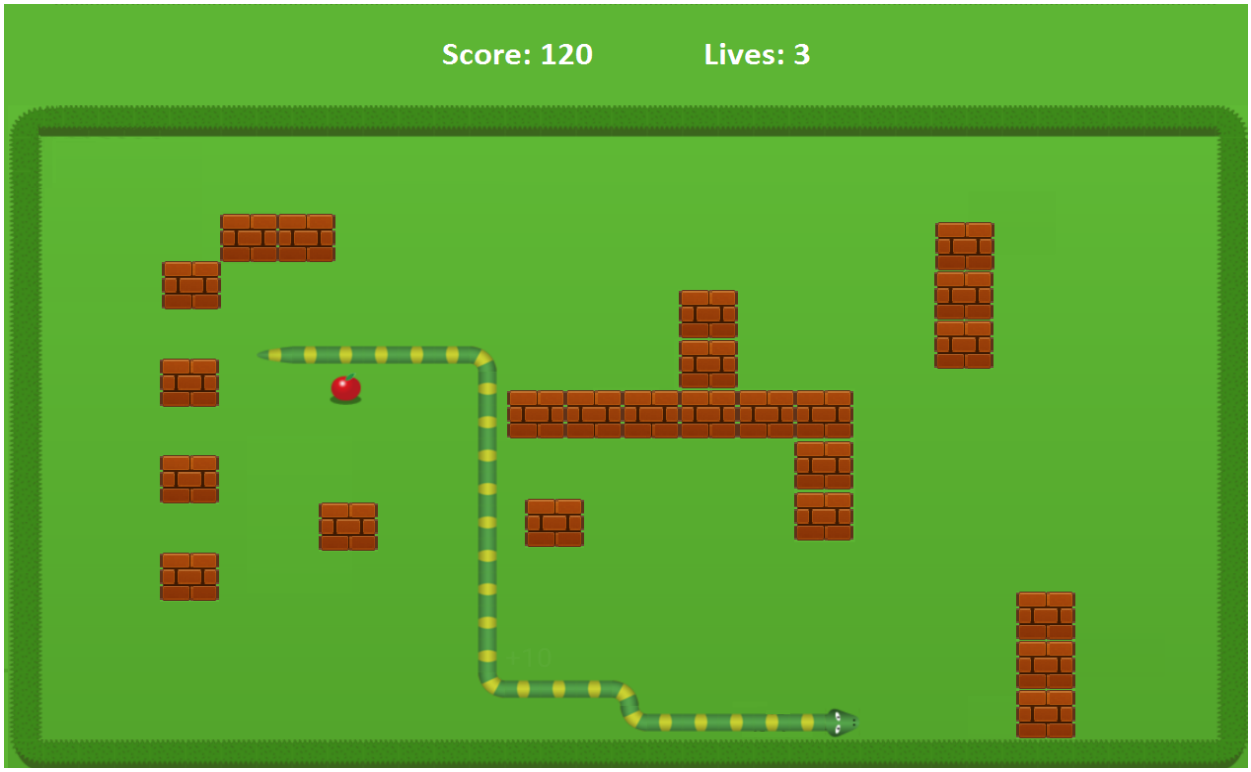
CPSC 359 – Winter 2016
Assignment 4
Raspberry Pi Video Game
70 points (Weight: 14%)
Due April 1st @11:59 PM

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

Game Objective: A snake moves across a maze-like screen avoiding walls and feeding on apples. Winning the game requires feeding the snake at least 20 apples after which the exit door is unlocked. The snake may not hit a wall nor can it collide with itself.



The background is a screenshot of the game from
https://lh3.googleusercontent.com/5Xd9_cskAmixNr6md6ajln9HshDBl2Iu2XQWshlro2iQ_I20yeN9IlcmlIZf3aOktQ=h900.

We modified it and added the walls.

Game Logic

The ***game environment*** is a finite 2D *n* x *m* grid (not necessarily a square).

- A ***game map*** is an instance of the game environment (for *n & m* $\geq$ *20*)
  - It specifies the score of the player.
  - It specifies number of lives left (starting with a minimum of 3 lives).
  - The screen is divided into cells. Each cell is one of a *floor, wall*, or a *border.*

    - The cells on the edge of the map are filled with *border tiles*; the snake cannot pass through the border or wall tiles.
    - Wooden walls must be scattered around the screen; use at least 20 such walls in different shapes.
    - An apple appears on the screen at the beginning of the game. When the snake eats the apple, then the snake grows by at least one cell and another apple will appear in a random cell. The new apple must be on a floor tile and it cannot overlap with the snake.
    - The starting snake size should be at least 3 cells.
- A ***game state*** is a representation of the game as it is being played, and it contains an instance of the *game map* with:
    - *The snake,*
    - *An Apple,*
    - *Wooden tiles,*
    - The *score* of the player (initialized to zero),
    - Number of *lives* left,
    - A *win condition* flag, and
    - A *lose condition* flag.
- The ***game transitions*** into a new state when the player performs an action.
    - The user collects score points by feeding apples to the snake. The initial score is zero.
    - The snake's contact with itself, a border, or a wooden tile will decrease the number of available lives by one.
    - The position of a new apple must be decided randomly. Check https://en.wikipedia.org/wiki/Xorshift for simple random number generator.
- ***Exit Door & Value-Packs***:
    - Use your imagination to implement at least one *value-pack* of your choice that adds some feature to the game. For example, the snake gets faster.
    - Creative *value-packs* will warrant extra marks.
    - The value packs should be employed after sometime from the start of the game (for example, 30 seconds into the game). To do so, you must use timer interrupts. You must also employ randomization as to where these value packs appear.
    - After eating all 20 apples, a door will appear randomly on the screen on an empty floor tile and when the snake enters, the *win condition* flag is set and the game is won.
- ***Action***:
    - Move by one cell in one of the four cardinal directions (up, down, left, right) if the action is valid. The move results in the *snake's position* being set to the position of the destination cell. If the user attempts to move the snake in a direction that is

opposite to the direction of the snake, the action is ignored and the snake will continue moving in its direction. For example, if the snake is moving up and the user presses down then the user's action will have no effect.
    - Moving into a floor tile marked as *value-pack* will result in:
        - The overall score being incremented,
        - The removal of the *value-pack* marking on the destination floor tile, and
        - Application of the feature effect.
- ***Game ends*** when:
    - The win condition flag is set,
    - The lose condition flag is set (including zero remaining lives), or
    - The user decides to quit.

Game Interface

- The main menu screen includes
    - CPSC 359 title,
    - Game title,
    - Creator name(s),
    - Menu options labeled "Start Game" and "Quit Game", and
    - A visual indicator of which of these 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
    - The 2D grid should be (roughly) in the center of the screen
    - Each different tile type is drawn with a different visual representation (in color and shape)
    - Score and lives-left are 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 is displayed
    - If the *lose condition* flag is set, display a "Game Lost" message is displayed
        - Both messages should be prominent (large and in the middle of screen)
- 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
    - Pressing the Start button will open a Game Menu that contains 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 are not processed when Game Menu is open
- Pressing the Start button will close the Game Menu
- Press Up or Down on the D-Pad scrolls between menu options
- Pressing the ***A*** button activates the selected 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 with Start Game selected to start game 1
      v. Press A with Quit Game selected to exit game 1
   b. Draw current game state: ***(22 marks)***
      i. All objects are drawn according to interface specifications 5
      ii. The snake moves in the available spaces and its body follows it 2
      iii. Apples disappear and appear again if eaten 3
      iv. Score/Lives are drawn as specified 4
      v. Value-packs are Implemented 4
      vi. Game Won message drawn on win condition 2
      vii. 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 Snake 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 apple and the value packs are randomly shown on 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	70

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 three students in order to complete the assignment, but you are not required to do so. **You are advised to keep the teams from Assignment 3 for this assignment.** 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, kernel.img, etc) to your TA via the appropriate dropbox on Desire2Learn. 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 **four** days
-25% for each additional day or portion of a day after the first **four** days
Hence, no submissions will be accepted after 7 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 (20 assembly language instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.