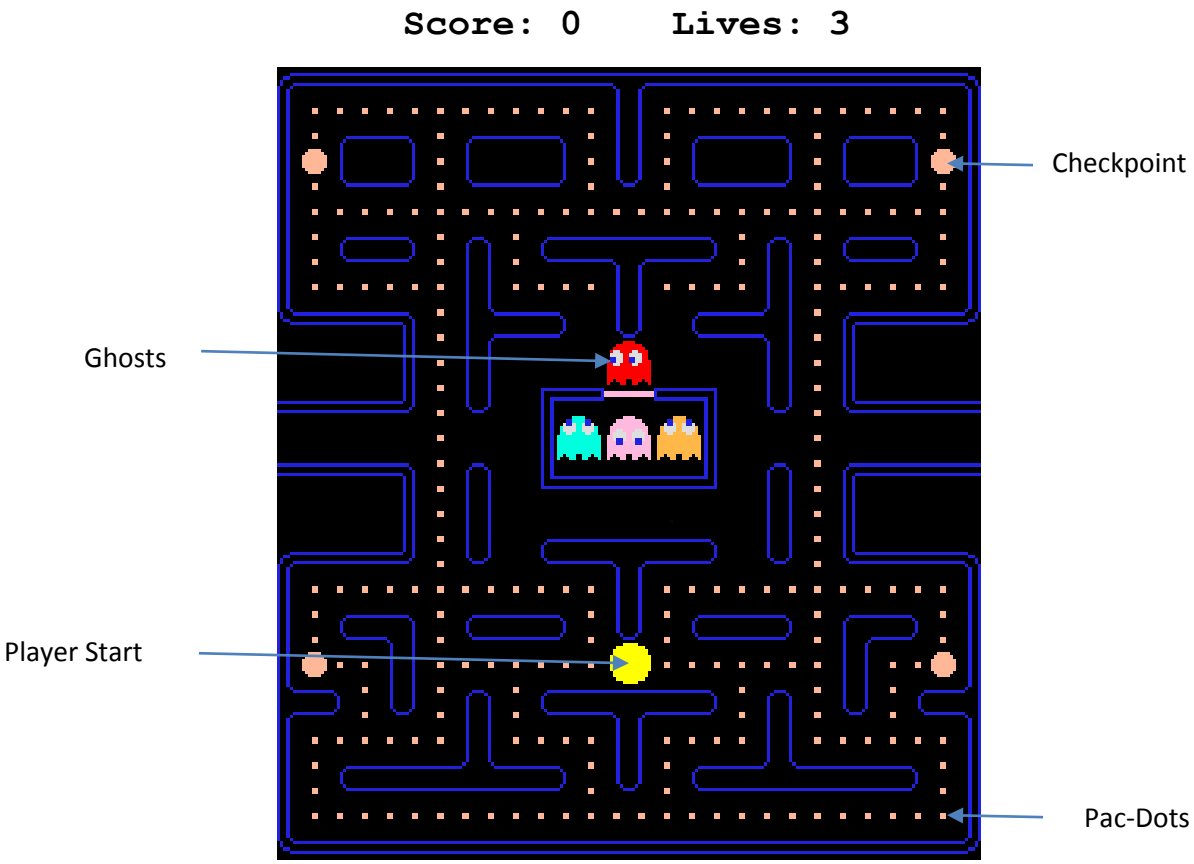
CPSC 359 – Spring 2015
Assignment 2
Raspberry Pi Video Game
40 points (18% of total mark)
Due June 26 @ 11:59PM

Objective: To implement a video game that requires a Pac-Man to travel through a maze with the goal of collecting Pac-Dots. Solving the maze will require the Pac-Man to move through the maze, collecting all Pac-Dots, and avoiding contact with enemies (Ghosts).



Source: http://upload.wikimedia.org/wikipedia/en/5/59/Pac-man.png

Game Logic

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

- Each grid cell contains either a *wall tile* or a *floor tile*
    - Floor tiles may be marked as *Pac-dot or Checkpoint.*
- A *game map* is an instance of the game environment (for a value of *n & m* $\geq$ *20*)
    - Specifies the score of the player.
    - Specifies the number of lives left (starting with a minimum of 3 lives).
    - Each cell of the grid is either a *wall* or a *floor* tile
    - The cells on the edge of the map are filled with *wall* tiles. Objects must stay within the game map.
    - One floor tile is marked *player start*.
    - Most of the floor area should be marked as Pac-dots.
- A *game state* is a representation of the game as it is being played, and contains:
    - An instance of a *game map*
    - The *player's position* on the game map
        - Initialized to the position of the floor tile marked *player start*
    - The score *collected* by the player (initially zero)
    - Number of lives left.
    - A *win condition* flag (set if and only if the player collected all Pac-dots)
    - A *lose condition* flag (set if and only if the player has zero lives left)

The game transitions into a new state by the player performing an action

- Only transitions to a new state if the player performs a *valid action*
- Player's contact with a ghost will lose an available life or set the *lose flag.*
- Ghosts are free to move in the floor area, they cannot collect Pac-dots, they cannot pass a wall tile. The decision on the pattern of movement is left to you (*No complicated AI is required*).
- Action: Move by one cell in one of the four cardinal directions (up, down, left, right)
    - Move action is *valid* only if the destination cell contains a floor tile
    - Results in the *player's position* being set to the position of the destination cell
    - Moving into a floor tile marked as *Pac-Dot* will result in:
        - The overall Score being incremented by (at least one).
        - The removal of the *Pac-Dot* marking on the destination floor tile
- Action: Moving into a floor tile marked as Checkpoint will result in:
    - Saving the State of the game at this checkpoint.
    - If a player collects another checkpoint, old checkpoint state will be deleted and the new state will be stored.

- Upon a ghost contact, if there are available lives, the state of the map will reset to the last checkpoint state, while the enemies (Ghosts) must return to their initial positions.

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

Game Interface

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 16x16 pixels
    - 2D grid should be (roughly) in the center of the screen
  - Each different tile type is drawn with a different visual representation
    - ie: wall, floor, dots, ghosts, player start and player
    - Minimally, cells are filled with different colours based on tile type
    - Game must contain at least 4 ghosts.
  - Game title and Creator name(s) drawn somewhere on the screen.
  - 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
    - 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, down, left or right on the D-Pad will attempt a move action
  - Performing a valid action will require the game state to be redrawn
  - Pressing the Start button will restart the game
  - If the win condition or lose condition flags are set
    - Pressing any button will restart to the game

Grading:

1. Game Screen
   a. Draw current game state
      i. All cells drawn according to interface specifications 5
      ii. Draw game title and creator names 2
      iii. Player Score and Lives left drawn 3
      iv. Ghosts can move in the available space. 3
      v. Checkpoints implemented as described. 4
      vi. Game Won message drawn on win condition 1
      vii. Game Lost message drawn on lose condition 1
   b. Interact with game
      i. Use D-Pad to move player (if move action is valid) 4
      ii. Press Start button to restart the game 1

	iii. Press any button to restart the game (game over)	1
2.	APCS compliant functions	3
3.	Well structured code	
	a. Use of functions to generalize repeated procedures	5
	b. Use of data structures to represent game state, etc.	5
4.	Well documented code	5
	Total	40 points

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: <u>We strongly recommend</u> that you work in teams of up to three students in order to complete the assignment, 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, 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 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 (20 assembly language instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.