CPSC 359 – Winter 2015
Assignment 2
Raspberry Pi Video Game
18% of final mark
Due March 9th, 2015 @ 11:59pm (midnight)

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.

Game Logic

- The *game environment* is the full available monitor buffer
- The user collects points by terminating (killing) enemy objects. If the score becomes 0 the user loses and the game ends. The initial score is some positive value
- The enemy objects are of three types:
    - Pawns: must be the easiest to kill and the user gets 5 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
- The user controls an avatar to move around the screen and shoot at the enemy objects; the actual movement patterns is left for you
- 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 enemy objects move as well avoiding being shot by the user avatar. They shoot back at the user avatar. The way they move on the screen is up to you to decide
- If the user avatar is hit by an enemy bullet, the user loses at least 10 points from its score
- 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 avatar can hide behind such obstacles. However, obstacles shrink when they are hit by bullets (from either party) until they disappear
- The game ends:
    - When the user terminates all enemy objects (user wins),

- When the enemy terminates the user object (user's score becomes 0 and user loses), or
- When the user decides to quit

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

- 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 avatar
  - Pressing the A button will shoot a bullet
  - All buttons need to be released before a new action can be performed
  - Performing a valid action will require the game state to be redrawn
  - Pressing the Start button will show a menu with 3 options:
    - 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. Draw current game state:

Draw game title and creator names somewhere on screen.	2
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
Losing points decreases the score.	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

b. Draw game menu:

Filled box with border in center of screen	1
Draw menu options and option selector	2
Erase game menu from screen when closed	2

c. Interact with game

Use D-Pad to move Avatar (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

d. Interact with game menu:

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.	***APCS compliant functions***	3
3.	***Well structured code:***	
	Use of functions to generalize repeated procedures	5
	Use of data structures to represent game state, etc.	5
4.	***Well documented code***	5
	Total:	**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 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.