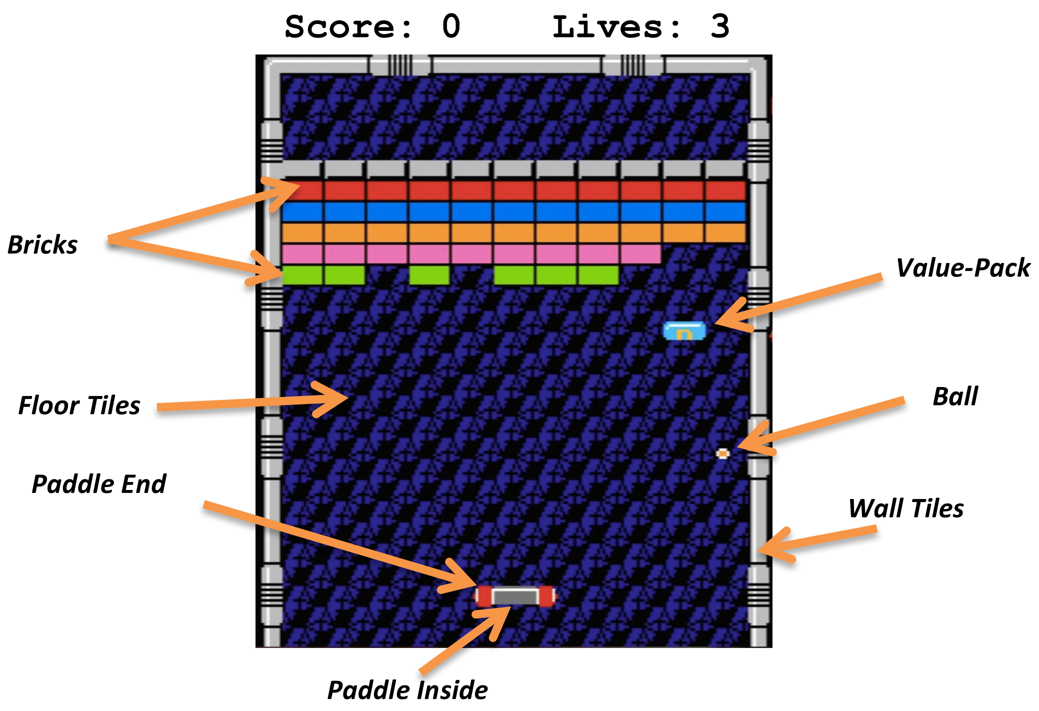
CPSC 359 – Spring 2018
Assignment 2
Raspberry Pi Video Game
55 points (Weight: 16%)
Due June 12th @11:59 PM

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

Game Objective: To implement a video game that requires a paddle at the bottom of the screen move left and right preventing a ball from falling off the playing ground. If the ball hits a brick, the brick disappears. It will bounce off if it hits a wall, a brick or a paddle. To win, all bricks must disappear.



Screen shot from: (https://www.flickr.com/photos/98859152@N04/11321061645)

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* $\geq$ *20*)
    - Each cell of the grid is either a *brick tile, wall tile*, or a *floor tile.*
    - The cells on the right, left and the upper edge of the map are filled with *wall* tiles.
    - Use at least 10 of each *brick* type.
- A ***game state*** is a representation of the game as it is being played, and contains:
    - An instance of a *game map.*
    - *Paddle's position* on the *game map* (initialized to a starting position).

    - *Ball's position & direction* on the game map, with initial *position* on top middle of the *paddle,* and initial right/upward *direction*.
    - The *score* collected by the player (initialized to zero).
    - Number of *lives* left (starting with a minimum of 3 lives).
    - A *win condition* flag.
    - A *lose condition* flag.
- **Bricks**:
    - 3 types of bricks must be present in the game. Types are differentiated in hardness of breaking.
- The ***game transitions*** into a new state when:
    - *Ball* moves on the screen (the position of the ball will change periodically according to *angle* & *direction*).
    - *Ball* Bouncing: the ball changes direction upon collision
        - *Ball* will only reverse direction if in contact with *wall* or *brick* tiles (horizontally or vertically).
        - If in contact with a *paddle*:
            - If striking left/right end, it bounces upwards and left/right at ~45º angle.
            - If striking inside left/right, it bounces in the same horizontal direction at ~45º angle.
    - The player collects points when the *ball* contact *bricks* objects. Initial score is zero.
    - If the Ball fell off the screen, player will lose an available life or set the lose flag.
    - *Paddle/Ball* cannot pass through *walls/bricks*.
- ***Action***:
    - Move in one of two cardinal directions (left, right) if action is valid. The move results in the *paddle's position* being set to the position of the destination cell.
- ***Game ends*** when:
    - The user breaks all brick tiles (win).
    - The user lives = 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

- ***Bonus if implemented (up to 3 marks): Value-Pack***
    - *A value-pack* is hidden under *brick* tiles. Breaking the brick tile will reveal it.
    - Choose one from:
        - Speed down the *ball*.
        - Catch the *ball* and shoot when you want to.
        - Expand the *paddle*.
        - Other (**Creative**) *value-pack* may be accepted (Check with your TA).
    - When revealed, a *value-pack* must travel from top to bottom.
    - Moving *paddle* 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.
        - Application of the feature effect. (The effect of a *value-pack* may be permanent till end of the game).

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).

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
    - Each different tile type is drawn with a different visual representation
        - ie: Paddle, brick, wall, floor, ball, etc. Minimally in color and shape.
        - Bricks & *(bonus)* value-packs of different types must be of different visual representation. Minimally in color.
    - Score and lives left are drawn on the screen
        - A label followed by the 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 game screen)
- The player uses the SNES controller to interact with the game
    - Pressing left or right on the D-Pad will attempt a move action
    - Pressing ***B*** button will release the ball from initial position.
    - Holding ***A*** button will increase the speed of the paddle.
    - Pressing the Start button will restart the game to its original state
    - Pressing the Select button 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 game state: ***(21 marks)***	
i. All objects are drawn according to interface specifications.	6
ii. Bricks disappear if breaks (according to its hardness).	3
iii. Ball animation/bouncing as specified.	5
iv. Score/Lives are drawn as specified.	3
v. Game Won message drawn on win condition	2
vi. Game Lost message drawn on lose condition	2
c. Interact with game: ***(9 marks)***	
i. Use D-Pad to move paddle (if move action is valid)	2
ii. Hold A button to speed up the paddle.	2
iii. Press B to release the ball from initial position.	2
iv. Press Start button to restart the game	1
v. Press Select button to return to main menu	1
vi. Press any button to return to main menu (game over).	1
d. **Bonus:** ***(3 marks)***	
i. Implement a value-pack as specified.	3
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	**55**
Max Possible Mark	**58**

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, myProg, 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.