# CPSC 359 – Fall 2016
# Assignment 4
# Raspberry Pi Video Game
# 75 points (Weight: 16%)
# Due Nov 25th @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:** The objective is to implement a Tetris video game. The game has 7 different pieces called "Tetriminos" composed of 4 blocks each. The different kinds of Tetriminos are shown in Figure 1. Each piece appears one at a time in a random sequence during the game. The player is to change the position and orientation of these shapes, by moving them sideways or by rotating them by 90 degrees, with the aim of creating a horizontal line of **ten units** without gaps. When such a line is created, it disappears, and any block above the deleted line falls down.

Each time a Tetriminos is dropped, the player's score increases by 1 point. When a line is cleared, the score will increase by 10 points. If more than one line is cleared at once, the points per line increase by 5 points. The player wins if s/he reaches a score of 150 and loses if s/he fails to reach 150 score and the last object is at the top of the game map.

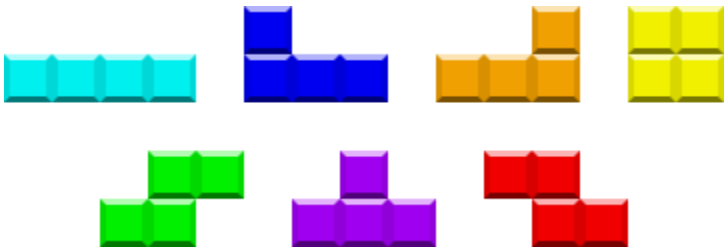

**Figure 1. The 7 Tetris shapes. Top row left to right I, J, L, and O. Bottom row left to right: S, T, and Z (Illustration is taken from Wikipedia)**

## Game Logic

The ***game environment*** is a finite 2D *n* x *m* grid (a rectangle with the height more than the width).

- A ***game map*** is an instance of the game environment ( for a value of ***n=19 & m>=*** **10**) where n is the number of rows (height) and m is the number of columns (width)
  - Specifies the score of the player.
  - The cells on the edge (top, left, right, bottom) of the map are filled with *border tiles*; the Tetriminos cannot pass through the border tiles
  - Each Tetriminos moves and rotates based on user commands

- A ***game state*** is a representation of the game as it is being played, and contains:
  - An instance of a *game map.*
  - The *score* collected by the player (initialized to zero).
  - A *win condition* flag.
  - A *lose condition* flag
- The ***game transitions*** into a new state when the player performs an action
  - A user collects score by dropping the Tetriminos on the floor or on another Tetriminos.
  - A Tetriminos making contact with the top border tile will set the lose flag.
  - The decision of which Tetriminos object will appear in the game is random. Check https://en.wikipedia.org/wiki/Xorshift for simple random number generator.
- ***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 increase the speed of the Tetriminos moving downwards)
  - You may get extra marks for additional creative *value-packs*.
  - The value packs should start appearing after some time (e.g., 20 seconds) into the game (to do so, you should use time interrupts) in a random place.
- ***Action***:
  - Move the Tetriminos downwards one block at a time until hitting another object including the bottom tile. Move the Tetriminos left or right depending on what the user specifies from the SNES controller if action is valid (not passing the border of the game). Rotating the Tetriminos by 90 degrees if it's possible depending on the user input (up or down).
  - Moving into a tile marked as *value-pack* will result in:
    - The overall score being incremented.
    - The game speed will increase or decrease depending on your choice.
    - The removal of the *value-pack* marking on the destination tile.
    - Application of the feature effect.
- ***Game ends*** when:
  - The user gets to the 150 score and wins
  - The Tetriminos hit the top tile (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
  - Each different tile type is drawn with a different visual representation
    - ie: Tetriminos, walls, value packs, etc…
    - Minimally, in color and shape.
  - Score is 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 (rotate) or right (rotate) on the D-Pad will attempt a move action (rotating or moving the Tetriminos)
  - Pressing the Start button will open a Game Menu
    - 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 button 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: ***(26 marks)***
      i. All objects are drawn according to interface specifications. 5
      ii. The Tetriminos moves in the available spaces and its body follows it 2
      iii. Score is drawn as specified. 4
      iv. Implement value-pack as specified. 4
      v. Game Won message drawn on win condition 2
      vi. Game Lost message drawn on lose condition 2
      vii. A line is cleared once a horizontal line of ten units without gaps 4
      viii. Several lines are cleared once a horizontal lines of ten units without gaps 3

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 and rotate correctly the Tetriminos 5
    ii. The first value pack will appear after sometime (e.g. 20 seconds) 6
    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 2
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 4
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 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 required to maintain the same teams from Assignment 3. Contact the instructor if this is not possible.** 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.

**D2L Marks:** Any marks posted on D2L are tentative and are subject to change (UP or DOWN).