Final Project: Tetris

Starter Code

You do not have to use my starter code for this assignment. Here's an overview of what you get if you do:

Tetris: The application, this class sets up the animation and key events and creates the game and board. Animation and key event messages are sent to the game object to be handled.

TetrisGame: Controls the game logic. Should control Tetris Pieces and the Board. Implement moveDown(), moveLeft(), moveRight(), etc to implement the game behavior.

TetrisBoard: "owns" squares when they have dropped and are on the board. There are several constants here that you can change if desired:

- SQUARE_SIZE: the length of a side of a tetris square
- Y_DIM_SQUARES: the y dimension measured in squares
- X_DIM_SQUARES: the x dimension measured in squares

The screen is set up to have dimensions (SQUARE_SIZE*X_DIM_SQUARES, SQUARE_SIZE*Y_DIM_SQUARES)

TetrisSquare: A single square, with a fixed height and width. Each Piece should be composed of 4 Squares. Squares have been implemented to place themselves at board coordinate locations, so that a square at board location (2, 3) would actually draw at location (SQUARE_SIZE*2, SQUARE_SIZE*3), with height and width SQUARE_SIZE. Two sample squares are drawn on the screen if you run Tetris as given to you in the assignment.

Checkpoint 1: Tetris Pieces

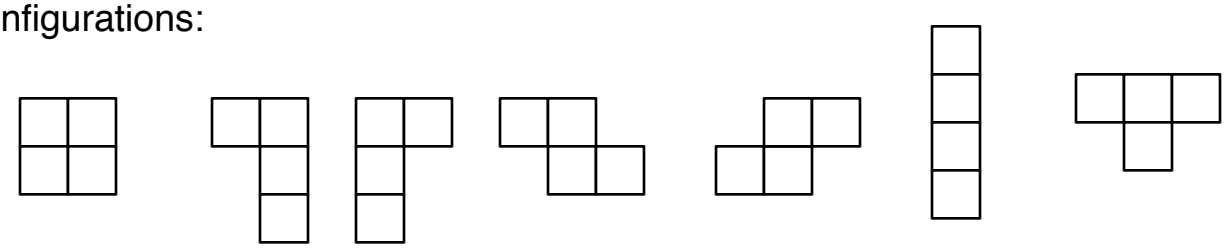
I recommend dividing this checkpoint into two parts and finishing one part early, then coming back to the second part later (in the first and second weekends available for this checkpoint, for example). If you leave the entire checkpoint for the last weekend you will most likely get behind.

In this part of the assignment, you will implement the basic Tetris Pieces, so that they move left and right and rotate. In the next assignment, you will implement the interactions between the pieces and the Tetris Board.

In this part of the assignment, you will:

- code each of the seven tetris pieces
- generate a random piece located at board location (X_DIM_SQUARES/2, 3) (on the screen near the top) each time your program starts
- code the move left and move right actions
- code the rotate left and rotate right actions

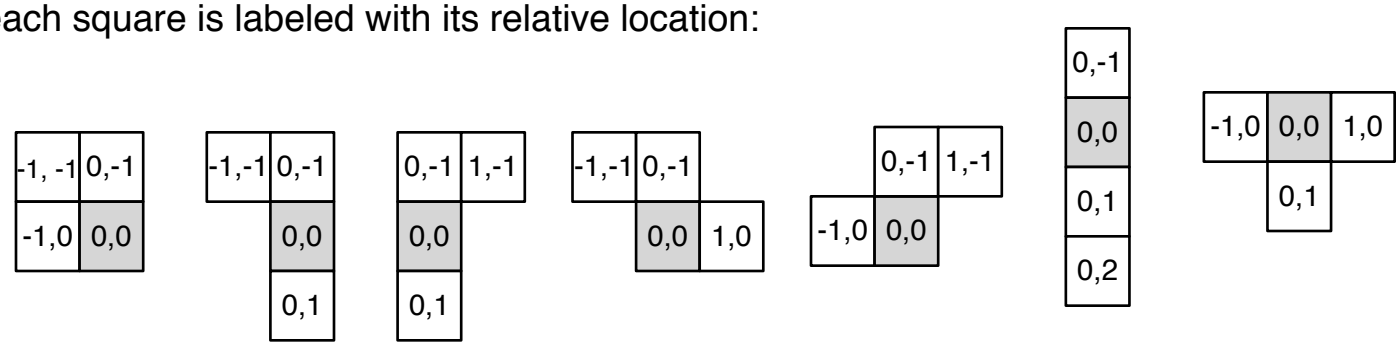
Tetris Pieces are made up of 4 Squares, arranged in one of the following seven configurations:



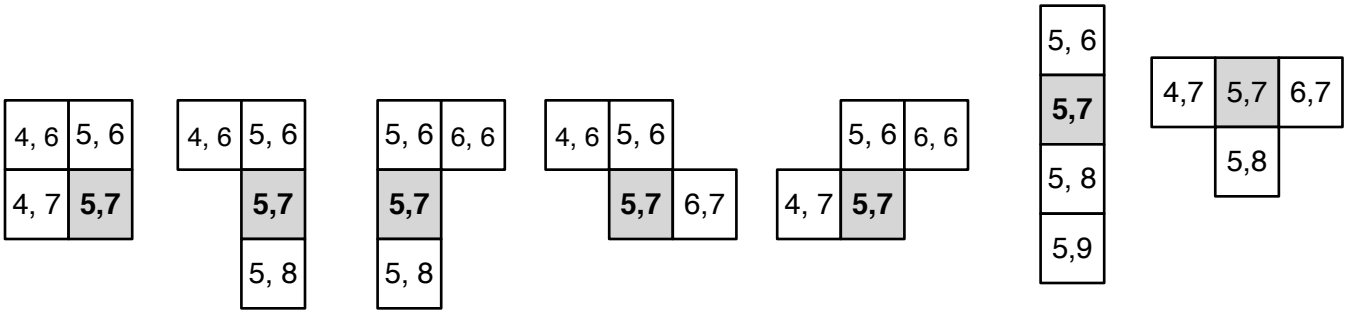
When writing the move and rotate code, it will be helpful if your Pieces keep track of two types of coordinates:

- the location of the center of the piece
- the relative locations of each block to the center

Here is what that looks like in the pieces above. The center square is shaded grey, and each square is labeled with its relative location:

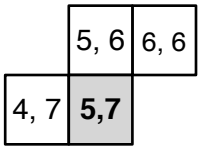


If the piece location were, for example, 5, 7, here's where each square would be located on the screen (in board locations), for each of the pieces. Notice that the each square's actual location is just the piece's location + the square's relative location:

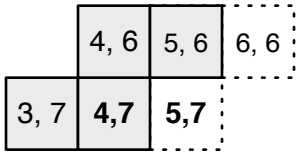


To move the piece, we can just change the piece's location. When rotating the piece, we can just change the squares' relative locations.

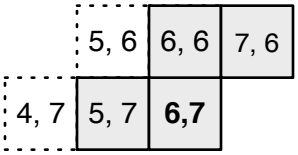
Let's take the Z shaped piece as an example. Move left and move right change the piece's center x location. The squares' relative locations stay the same, though their actual locations update with the piece:



Original location

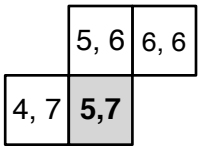


Move left

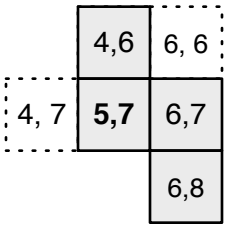


Move right

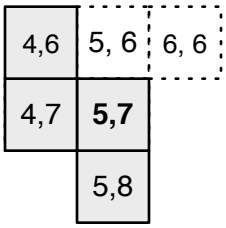
If you look at the actual locations and how they change with a rotate, its hard to see a pattern. Rotate based on actual locations would be harder to implement:



Original location

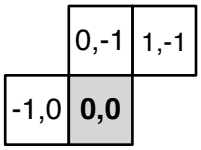


Rotate right

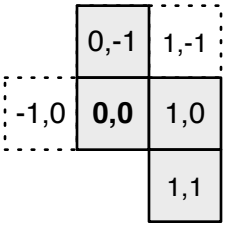


Rotate left

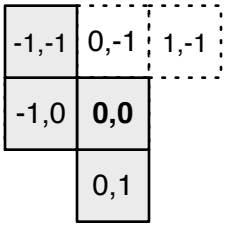
The relative location changes, however, are easier to understand and implement:



Original relative locations



Rotate right



Rotate left

For rotate right, the relative locations change as follows:

- newRelativeX = -oldRelativeY
- newRelativeY = oldRelativeX

For rotate left, the relative locations change as follows:

- newRelativeX = oldRelativeY
- newRelativeY = -oldRelativeX