**CLASSES:**

**Grid**: For the background in game play and the grids

**Block**: A single block class that is inherited

**I,L,O,Z,J,S** and **T** tiles: specific tile objects that inherits the Block Class

**DETAILS:**

**GRID:**

**Attr:**

* Width/number of columns: 10 cells
* Height/ number of rows: 20 cells
* Each cell size = 30px
* Grid: a list to represent the grid in terms of numbers, ie if the cell on the grid is empty, its assigned 0 otherwise, a number that reflects the block that has occupied the cell **OR:** Its [[0 for i in range(<number of columns>)] for j in range(<number of rows>)]
* Colors: a list of colors that will present the block that has occupied the cell

**Methds:**

* **Draw(Surface/window)**: For drawing the grids on the window
* **Line\_full():** returns a Boolean to check whether there is any line on the grid that is filled with blocks, if yes, the line disappears and score is made
* **Remove\_line():** removes a line when line\_full() methods returns true
* **Cell\_is\_empty():** returns a Boolean that checks whether the cell is occupied by a block, e.g an empty cell is given a value of 0 and if its not empty a value corresponding to the color of the block occupied is assigned to the cell

**BLOCK**:

**Attr**:

* Id: an integer to distinguish the type of shapes. It must be overridden by other shapes inorder to distinguish them from others ie to know what type of shape is it
* Cells: a dictionary to hold key-value pairs of state of rotation of a block with its position i.e. {state of rotation: position}. States range from 0 to 3 but in block class it’s an empty dict. It’s must be overridden by all the shapes classes ie the I,L,S,Z,T,J and O classes to get the positions of each rotation states
* Rotation state: an integer for the state of rotation for the block
* Row\_offset: an integer for the number of rows the block moves from its origin by default = 0
* Column\_offset: an integer for the number of columns the block moves from its origin by default = 0

**Mthds**:

* **Draw(surface/window):** draw the block on grid/screen
* **Move(rows,columns):** move the block by adding the row offset and the column offset to the number of rows and number of columns provided as arguments
* **Rotate(direction):** Rotates the block in 4 directions ie Left, Right, Top and Down
* **Collided ():** returns a Boolean to check whether the block has collided with the other block, hence stopping its interactivity
* **Out-of-bounds ():** returns a Boolean value to check whether the block is still in the playing boundaries, ie not passed the game screen width
* **Get\_new\_position():** returns a list for the new position of the block when the move function is called

**Colors:** it has a static method that returns a list of colors to be used in the project

**Position:** it encapsulates a tuple of x, y coordinates for position of the block

**Game:**

Holds the Whole Game Logic and Architecture

**Attr:**

Grid: an instance of the grid class

Blocks: a list of instances of all the 7 shapes

Current block: get random block(), for the current block

Next block:get random block(), for the next block

**Mthds:**

* **Get\_random\_block():** picks a block at random from the blocks list, removes it from the list and after returns it
* **Draw(surface):** calls the draw functions for the grid and the current block

**SUMMARY FOR THE BLOCKS ROTATION STATES AND THEIR CORRESPONDING POSITIONS**

**L-Block:**

0: [(0,2), (1,0), (1,1), (1,2)]

1: [(0,1), (1,1), (2,1), (2,2)]

2: [(1,0), (1,1), (1,2), (2,0)]

3: [(0,0), (0,1), (1,1), (2,1)]

**J-Block:**

0: [(0,0), (1,0), (1,1), (1,2)]

1: [(0,1), (0,2), (1,1), (2,1)]

2: [(1,0), (1,1), (1,2), (2,2)]

3: [(0,1), (1,1), (2,0), (2,1)]

**I-Block:**

0: [(1,0), (1,1), (1,2), (1,3)]

1: [(0,2), (1,2), (2,2), (3,2)]

2: [(2,0), (2,1), (2,2), (2,3)]

3: [(0,1), (1,1), (2,1), (3,1)]

**O-Block:**

0: [(0,0), (0,1), (1,0), (1,1)]

1: [(0,0), (0,1), (1,0), (1,1)]

2: [(0,0), (0,1), (1,0), (1,1)]

3: [(0,0), (0,1), (1,0), (1,1)]

**S-Block:**

0: [(0,1), (0,2), (1,0), (1,1)]

1: [(0,1), (1,1), (1,2), (2,2)]

2: [(1,1), (1,2), (2,0), (2,1)]

3: [(0,0), (1,0), (1,1), (2,1)]

**T-Block:**

0: [(0,1), (1,0), (1,1), (1,2)]

1: [(0,1), (1,1), (1,2), (2,1)]

2: [(1,0), (1,1), (1,2), (2,1)]

3: [(0,1), (1,0), (1,1), (2,1)]

**Z-Block:**

0: [(0,0), (0,1), (1,1), (1,2)]

1: [(0,2), (1,1), (1,2), (2,1)]

2: [(1,0), (1,1), (2,1), (2,2)]

3: [(0,1), (1,0), (1,1), (2,0)]

**PROJECT WORKFLOW**

1. Install Pygame
2. Import Pygame,random,and sys packages
3. Set up the game loop
4. Create the grid
5. Create the blocks
6. Move the blocks
7. Rotate the blocks
8. Check for collisions
9. Check for completed rows
10. Check for Game Over instances
11. Create user interface
12. Add scores
13. Add Next Block
14. Add Sound