**Tetris v 0. documentation:**

Modified: 5/18/14

**void setup():**

Each piece (T piece, L piece, etc) is named X\_piece and is defined by creating a PieceType object. Each PieceType object is created by passing an svg file that includes the illustration for each block (in this case each type has a different color but all else same appearance). The object also uses a X\_pieceDesign variable, which has information on the block arrangement (what distinguishes a T from an S shape). The pieceDesign variable is an array containing 0s and 1s with 1s signifying the block location versus empty space.

The setup also has a for loop that sets empty field. In the tetris game, the field is the rectangular space where the blocks will fall. The field 2d array has values of 0s and 2s, where 0 denotes empty field (no piece occupies it) and 2 denotes a filled field (where a piece occupies it; this is a permanent piece occupying, not a transient piece)

continue to: void draw()

**void draw():**

from: void setup()

At any point of the game, one piece is falling at a time. This piece is the variable currentPiece. currentPiece can be any X\_piece and it changes everytime a piece has fallen. In each draw frame, the currentPiece.display() is called.

continue to: currentPiece.display()

**void keyPressed():**

from: when a user hits a key

Continue to: currentPiece.userInput()

**void currentPiece.display():**

from: void draw()

This is where the action happens. A Clear Space for loop clears space where piece is moving but leaves permanent space filled. This ensures that as piece moves, the empty field is redrawn. Only if the field value is 2 (permanent piece location) will it be cleared.

A Match Field for loop changes the values of the field array based on the pieceDesign. pieceDesign array has 1s or 0s where 1 denotes block presence and 0 no blocks. originX/Y is the location where piece is on the field. This loop ensures that the piece is drawn on the correct location on the field. When the piece is drawn, it means: The value of the field array element is changed to 1 where the block should be.

Call checkAllowableMoves() to detect if the piece can keep moving down.

Draw Field loop finally draws the game on the screen. It takes the field array values and draws a "block", where block is an SVG file corresponding to whatever piece is set.

Drop piece slowly If clause is ran to change the piece position based on clock speed

Reset piece If clause ensures when piece reaches bottom, another piece falls from the top

**void checkAllowableMoves():**

from: currentPiece.display()

Check Bottom Collision: if space below is already occupied ith a fallen piece, then the piece will stop moving and be a permanent fixture in the field. Where the piece has landed, the field array elements will change value to 2, where 2 indicates a permanent/fallen piece.

Call paintPiece():

To do: do collision detection for left and right

**void paintPiece():**

from: checkAllowableMoves()

Called when a piece has decided to be permanently fixed on the field. If so, the array for the field is changed to 2 to indicate that the area has been fixed by the piece.

**currentPiece.userInput()**

from: void keyPressed()

Pressing ‘x’ would rotate the piece by changing rotation\_status. Pressing left/right would change the position of the piece by changing originX/Y.

Based on the rotation action, rotation\_status will be changed. pieceDesign variable will be changed by calling **updateArray(**currentPiece, rotation\_status). This will change the pieceDesign to reflect the new rotation, where the array will be “rotated” accordingly.

**Int[][] updateArray(..)**

from: currentPiece.userInput()

Returns a new array. In currentPiece.userInput(), the user can press a key in attempt to rotate the piece. In practice, pressing the key would result in changing the pieceDesign variable, which holds information of how a piece is shaped.

Work in progress changes: