Amy Wang

GUI Tetris

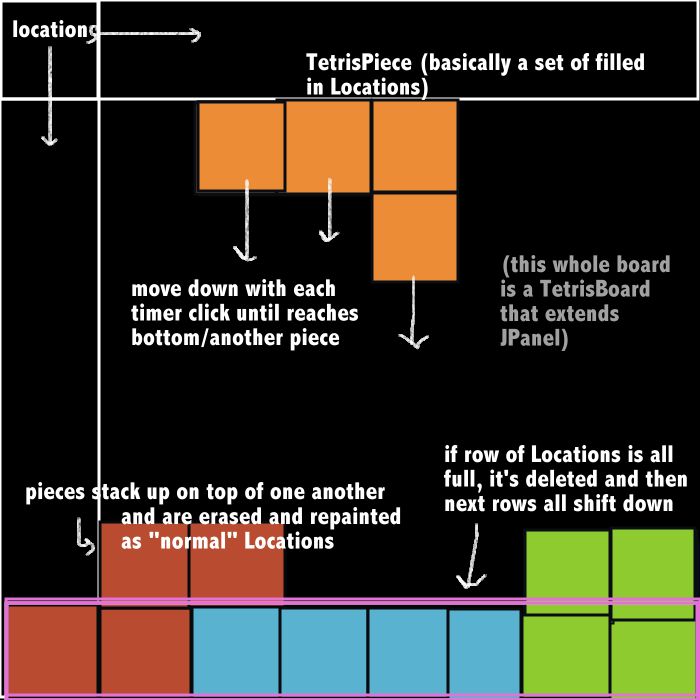
5/4/15

## **Functional Overview**

My project will be a GUI Tetris game, potentially attached to a GUI “arcade” interface that will include several options for games to play. In terms of behavior, I aim for it to behave as a generic Tetris game would—I want to the user to be able to rotate and direct the TetrisPieces in order to make the game last as long as possible, as well as earn points and see their final score upon reaching “game over.” I intend for anyone to be able to use it (provided, of course, that they are capable of using a computer), and in particular, I am aiming to make it a very easily accessible product for those *not* too familiar with computer programming. Hopefully if I have enough time, I will be able to export and run it entirely separate from JGrasp. Also if I have time, I’m hoping to collaborate with Kristy and combine our games into a virtual “arcade”—but that remains to be seen.

## **Design Overview**

The program will utilize three different classes: Location, TetrisPiece, and TetrisMain. Below is a fairly simplified diagram of how I want it to look & act visually. Pseudocode is in the Design Details section, and elaborates further on how exactly I plan on implementing the program.



## **Design Details**

Before I can actually begin coding the program, I will need to figure out how GUIs work…this will probably involve a lot of reading, Youtube videos, messing around with JGrasp, and obsessively analyzing the code for GridWorld and TileMain. As of right now I have yet to learn all I need to know about this aspect of Java, so parts of my pseudocode are still lacking. But here is my plan thus far:

**Location.java:**

Essentially quite similar to GridWorld’s Location class.

INSTANCE VARIABLES

Private int size //the length/width of single Location units

Private Point top //the top left point of the Location

Private Boolean full //tells whether or not the Location is being “occupied” by a TetrisPiece

METHODS

Public void Location(int size, int top) //specified constructor

This.size = size

This.top = top

Public Point getTop()

Return top

Public Point getSize()

Return size

Public void setTop(Point p)

Top = p

Public void setSize(int size)

This.size = size

Public Boolean isfull()

Return full

Public void fill(Color c) //paints in the Location w/ the signified color and sets it to full

If (full is not true)

Draw a rectangle of size this.size and color c starting at Point top

Full = true

Return true

Else

Return false

Public void empty()

Draw a rectangle of size this.size and color Color.BLACK starting at Point top

Full = false

Public Boolean equals(Location other)

If(other.getTop and other.getSize are equal to this one’s)

Return true

Return false

**TetrisPiece.java:**

INSTANCE VARIABLES

Int orientation // must be either 0, 1, 2, 3

Char tiletype //must be either “I” “L” “O” “Z”

Color tilecolor

Location [ ] locations //list of all the locations filled in by the TetrisPiece

METHODS

Public Tile (char type)

Tiletype = type

If(tiletype = I)

Add a row of length 4 of Locations (starting at the first row of Locations of the panel) to the locations array.

Else if (tiletype = L)

Add an “L” of Locations (starting at the first row of Locations of the panel) to the locations array.

Else if (tiletype = O)

Add a square of Locations (starting at the first row of Locations of the panel) to the locations array.

Else if (tiletype = Z)

Add a “Z” shape of Locations (starting at the first row of Locations of the panel) to the locations array.

Else

Throw exception (print message “Invalid Piece type”)

Call drawSelf to draw the Tile.

getLocations()

return locations

rotate()

(orientation + 1)%4

Erase each Location and moving it to whichever Location is dictated by the specific orientation, as well as the tiletype? Then calls drawSelf again. This part needs some more fleshing out…

drawSelf()—colors in each of the Locations in the locations array by calling the Location.paint() method for each Location in the locations array

move(Grid 1) – “moves” the piece by erasing each location and moving it 1 block to the left, right, or down, then redrawing itself in the new position. If the down key is pressed, moves down 2 units. If the left key is pressed, moves down and to the left a unit. If the right key is pressed, moves down and right a unit. Otherwise, moves downwards one unit, provided all spaces are unoccupied. Returns true if it moved successfully, false otherwise.

eraseSelf()

empty out all Locations in the locations array

**TetrisBoard.java extends JPanel implements KeyListener //NOT SUPER WELL LAID OUT B/C I HAVEN’T READ UP ENOUGH ON GUIS**

Private int width

Private int height

Private int unitsize

TetrisPiece activepiece

Public TetrisBoard()

Create new Timer

Don’t know syntax, BUT the timer will: Set a length of time between “ticks” and call all listeners/actions with each tick.

Make a new int, “score”, and set it equal to 0

Create a new JButton “scorebox” displaying score

Create new EventListener trackactive–will track the active piece and note whether or not it can move with each tick of the timer.

If(the piece can’t move)

Erase the piece from the board and re-fill in all its Locations with the Color of the piece

Call checkiffull (see below)

Generate new piece at top of Board and set it as the new activepiece

Create new KeyListener keylisten—will track the user’s keyboard actions with each timer tick, and note if the up, down, left or right arrow buttons are pressed

If(up arrow is pressed)

Rotate the piece w/ “rotate” method

Else if(any of the other arrow keys are pressed)

Call the “move” method on the piece, passing in the specific key pressed as a parameter

Create new EventListener checkiffull—checks each time tick to make sure that the row of Locations in which the TetrisPiece has fallen/solidified is not completely full.

If(the row is full)

Empty all the Locations in the row and shift down all the others on top to fill the blank spaces (by erasing and redrawing 1 below)

Add 10 to score

Create new EventListener gameend—checks to see if the game has ended (i.e. a block is touching the top of the panel)

If(the top row of Locations is NOT completely empty)

Stop everything and display a new dialog box that says “GAME OVER” + score

**TetrisMain.java:**

CLASS CONSTANTS

* Int screenwidth = 500
* Int screenheight = 700
* Int unitsize = 50 //size of one Tetris “unit”

MAIN METHOD

Create a new JFrame called “Tetris”

Set default close operation to “exit on close”

Set size to be (screenwidth, screenheight)

Make a new TetrisBoard and add it to the JFrame

Set “Tetris” to be visible

## **Testing**

Since there isn’t really a lot of user input involved in this game, apart from key-pressing, for the most part, I’m just planning on pressing different buttons to test alternate input. The four “base cases,” of course, will consist of the up, down, right and left keys. Apart from that, I will basically be testing other keyboard keys, as well as different combinations of keyboard keys. For example, I will note what happens when I press the space bar, the space bar with a letter, a space bar with an arrow key, etc. I will also be testing different combinations of the arrow keys themselves, in order to decide an “order of precedence” in which they will execute their actions (for example, the right key will dominate the left key—if the user presses both at once, the Tetris piece will move right, not left.)

I’ll probably also be clicking the screen in various locations, checking to make sure the program is unaffected by such clicks. And last but not least, I will certainly be having my family and friends test it out a lot for me.

## **Grading Rubric**

Write your own grading rubric (out of 40 points) that takes into account whether

Program functions as planned – 15 total

Compiles + runs without throwing exceptions 1 pt

Successful implementation of the Location class 3 pt

User interaction with GUI results in appropriate behaviors from the TetrisPiece class (rotation, downward shift, etc) 2 pt

Tetris Pieces move down screen until reaching bottom of panel 2 pt

Tetris Pieces are erased properly, Locations emptied when a row is filled out 2 pt

Proper score tallying + “game over” screen displays the proper format 2 pt

Game over screen occurs at the right time (when top row is no longer empty) 2 pt

Tiles are successfully randomly generated 1 pt

Design and efficiency – 15 total

Clear understanding and implementation of GUI 7 pt

All code has been refactored in an efficient manner 3 pt

Documentation – 5 total

All class constants, methods, parameters and return values are Javadoc’d 4 pts

Unclear statements are elaborated upon via appropriate commentation 1 pt

Test plan – 5 total

There is a test plan 1 pt

Test plan is comprehensive; alternate input throws no exception 4 pts

## **Proposed Implementation Schedule**

* Read up on GUIs and Java
* Chapter 14 in Java Programming textbook
* BlueJ website recommended by Dr. Bricker
* Youtube videos to help provide visual examples of the implementation of GUIs
* Other website reading, as necessary
* Create some simple test programs, just to experiment with GUIs
* Take another look at TileMain and GridWorld and attempt to decipher some of the code with my new knowledge of GUIs
* Revisit/rewrite pseudocode for each class in the program
* Implement Location class according to pseudocode outline
* unit test each method individually using a JPanel—make a new location, fill in the location, etc
* make changes as appropriate
* Implement TetrisPiece class

1. Make sure Pieces can generate and erase themselves—unit test and adjust as necessary
2. Implement move method, same deal as step 1
3. Make sure all methods working adequately

* With just 1 tile shape (simple square), get the TetrisBoard to work. Successfully program the Tetris piece to move down the screen until it reaches the bottom/cannot move anymore. Unit test this until it is fully functional.
* Add in code to “erase” the row of Locations, should it be full.
* Implement random TetrisPiece generator in TetrisBoard
* If time: with Kristy(?), implement GUI user interface (start game, end game, points tally, etc)

## **Potential Showstoppers**

* I’m a little afraid of not being able to adequately understand GUIs and how they work. So far I’ve done some reading and research and experimentation in JGrasp, and I think I have a better handle on them now than I did embarking on this project, but I’m still woefully inexperienced at actually implementing programs utilizing them.
* Tetris is also an unexpectedly complicated game (I “studied” some by playing it online, and was surprised by the many rules and regulations I had forgotten about.) It seems simple, but it’s actually got quite a few aspects to it that might prove fairly challenging to implement.

## **Open Questions**

* Am I on the right track?? What is the best and most efficient way to implement this program?
* How do I use GUI???
* I’m still having some trouble with the “rotate” method for the TetrisPieces, so I may need some help with that.

## **Resources**

I’m using YouTube, the Building Java Programs book, and several helpful online websites to help me figure out GUI. Hopefully Dr. Bricker can help me with some of my implementation as well, as currently I am very doubtful as to how I will be able to successfully create this program.