## Code Review on Gameboard

November 26, 2016 @3:00PM – 5:00PM in SPINKS 372

### Summary

The first thing that was established in our code review was that no question or comment was stupid. We wanted to ensure that everyone’s voice was heard and to promote an open work environment as ensuring clarity was our main concern.

As lots of our system relies on how the coordinates are stored and accessed we decided that it would be best to review the Gameboard as well as classes that manipulate it. Michael had done most of the heavy work in implementing this, so the first step was to logically step through each function, explaining what each function is meant to do, as well as why they had done it that way. From there, each member opened the source code and began to read it line by line. Anytime that someone had a question or a comment we would discuss it as a group and ensure that it was appropriately addressed and recorded.

### What was discussed

* Arianne: “Why are coordinates stored in a hash instead of something like a 2D array?”
  + Michael: “A 2D array will have unused space. You would have to loop over coordinates that don’t exist and check for nullity every time. Also, if you don’t know the exact index of the coordinate you can just check if the hash contains it. “
* Arianne: “What are the benefits of setting coordinate’s (x, y) values to the physical position on the screen?”
  + Michael: “By setting the coordinate’s (x, y) position to the physical location on the screen we can just draw the hexagon at exactly where it’s supposed to be. This reduces the amount of math that needs to be done every time that the board is repainted as it is only done when the game is initialized.”
* Arianne: “Why are do you use a hashmap keyed to a pair of integers with a value of a coordinate?
  + Michael: “That’s… a good question. Now that you mention it, it’s completely pointless; the key is the exact same as the value. Thanks for pointing that out, is there any other areas that you guys notice that need patching up?”
  + Kristof: “You could map a coordinate to it’s corresponding polygon/hex thing. That way you reduce the amount of hashmaps in the system, and the key (or the coordinate) maps to the value (or the hex) that needs to be drawn”
  + Michael: “That’s a good idea. I’ll refactor the code after this meeting.”
* Kristof: “How are the values determined in getNeighbors?”
  + Michael: “Well they’re based on the absolute position of each coordinate. I can approximate locations between each variable thanks to this math (wrote stuff on whiteboard). I can only approximate things because integers automatically truncate values so I may be slightly off on each ‘guess’ of where the coordinate that I’m trying to get. This is one of the reasons why I created an inner class to the gameboard that determines if a coordinate is within a range of a circle. “
* Chris: “Why do you check if a coordinate is in a range of a circle? We’re working with hexagons – not circles.”
  + Michael: “This is done because it is less computationally expensive to check if a given value is in range of a circle.”
* Chris: “Was Pythagoreans Theorem used in calculating the distance that a coordinate is from the center of a circle?”
  + Michael: “Yes.”
* Chris: “I thought running a square root operation is fairly resource intensive, how is this better than doing the math to calculate the bounding box of a hex?”
  + Michael: “In calculating the bounding box of a hex you must use sin, cos, and tan quite a bit while working with radians. These operations combined are much more computationally complex than the single square root function that is used. Furthermore, this is a turn based game, and if a value takes a millisecond to calculate I don’t think it will really make a difference.”
* Arianne: “Why do we define a function called drawCenteredCircle and what does the math inside of it do?”
  + Sam: “This is done because when Swing draws circles their central point is not in the actual center of the circle. The math creates an offset coordinate and draws the circle so that the center of the circle visually corresponds to what the physical coordinate is.”
* Arianne: “Why is it drawCenteredCircle not used?”
  + Michael: “It was only created to debug/visually see the bounding circles of each hexagon. It allowed me, Sam and Kristof to find some of the problems occurring with bounding circles like bounding circle overlap.”
* Chris: “What exactly is going on inside of the buildGrid() function?”
  + Michael: “I don’t fully understand all of the math. I added a link to Stack Overflow where the one user wrote it. It may be best to check there.”
* Chris: “Why doesn’t the origin of the Gameboard actually equal the central coordinate once the grid is built?”
  + Sam: “It’s largely to do with the same issue presented with drawCenteredCircle. The hex’s must be drawn at an offset to ensure that they’re actually centered in the JPanel.”
* Sam: “I missed how you got the corresponding hex to a mouse click because you did it after our pair programming session. Is that found by using a circular range?”
  + Michael: “Yes, I made an keyed to a circular range, with a corresponding hex/coordinate as its value. Then what I do is loop through all the possible ranges and find one that the current selected coordinate is a part of. As this operation is done frequently is was put into its own function getKeysFromRange.”
* Sam: “Okay, so if a coordinate is not in range of any of the ranges the system loop through all possible ranges and check if the coordinate is in range only to return null in its worst case. Is that a good solution? It seems like it would be a computationally intensive algorithm that is going to be called a lot.
  + Michael: “Honestly, I couldn’t think of a better one. If anyone has any ideas, please let me know.”
  + Chris: “You could do the same math that you did in buildGrid to get the exact location of a hex if it was a 2DArray. There wouldn’t be any rounding errors due to it being an integer either then.”
  + Sam: “That would reduce the complexity quite a bit, as the x and y offsets are calculated upon initialization and are stored for the duration of the Gameboard’s life… The question is it worthwhile to change the implementation at this point?”
  + Michael: “Honestly, I won’t have time to fix it if we decide to change it. I also don’t think it’s really a necessary refactor because I will still have to check if a mouse click falls within a certain range – the user won’t click on the exact pixel coordinate of each hex.”
  + Kristof: “At this point it’s not worthwhile. The solution works, and we don’t really need to worry about that sort of thing with modern processors, but it’s a nice thing to note.”
  + Michael: “Are we all okay with not changing it?”
  + ALL: “Yes.”
* Michael: “Are there any other questions?”
  + ALL: “No.”

### What were the benefits and costs accrued?

* **Benefit & cost:** We noticed that there were unnecessary hash maps that needed to be removed. This hardly affected our current work thanks to MVC’s modular design. The change was completed in under an hour as it was just basic refactoring. The benefit of this was that less storage was being done in the system, and that there were less hashes to keep up to date (reduced coupling).
* **Cost:** Sam and Chris pointed out that the same math could be used as what was used in buildGrid() to instantly calculate the pixel coordinate of each hexagon based on their index. This would the amount that bounding circles are relied upon in getNeighbors… It wasn’t worthwhile to implement though.
* **Cost:** We spent a significant amount of time reviewing the code when other features could have been implemented. The main benefit is that all people on the team now understand what is occurring in the Gameboard/GameScreen/Coordinate classes much better now, but it wasn’t necessary for the parts of the code that they were working on. Furthermore, all pieces of code that each group member is working on functions the exact same as it did before as the interface that they work with has not changed whatsoever.

EDITED: November 27, 2016