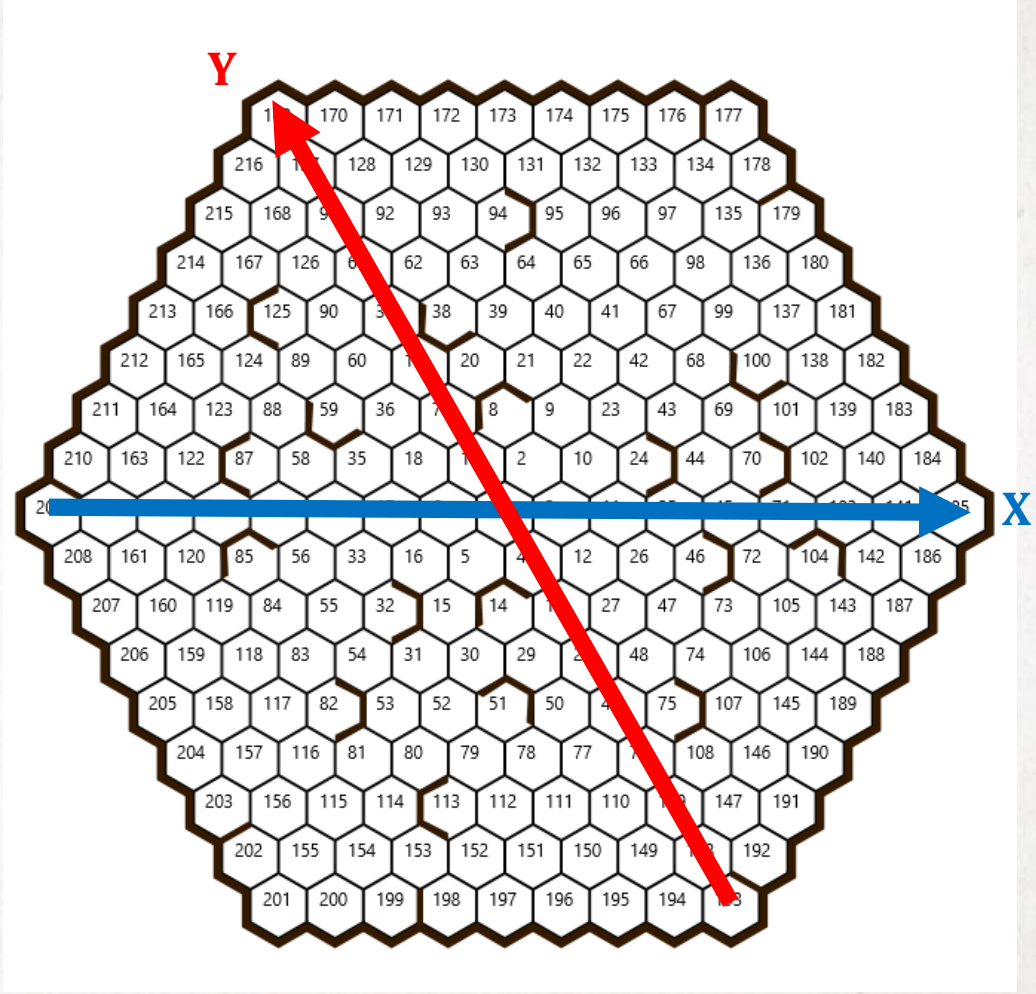
COMP1110 ASSIGNMENT 2 PRESENTATION

XAVIER O'ROURKE, QINYANG YU

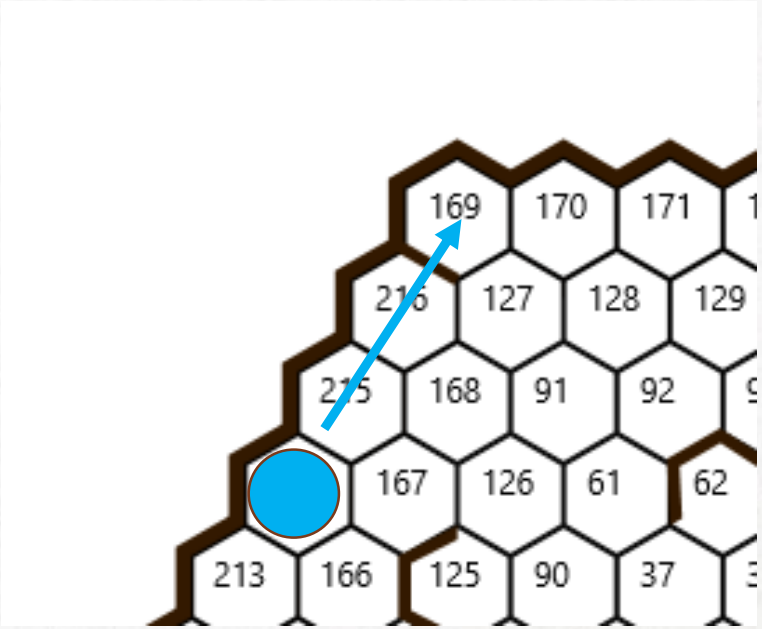
X-Y COORDINATE SYSTEM

- This was pretty much the first piece of code we wrote
- The code can convert positions (numbers 0-216) into pairs of X and Y coordinates
- So that the coordinates easily map onto tessellating hexagons, the y-axis and x-axis make a 120 degree angle with each other



hexGame.movePiece(int piece, Direction d)

- Once X-Y coordinates were implemented it was reasonably straightforward to give HexGame objects a movePiece( ) function.
- Subtle bug which escaped our notice till much later on—cranny collision didn't work when moving from position through position 216 to position 169



MINIMALPATH( )

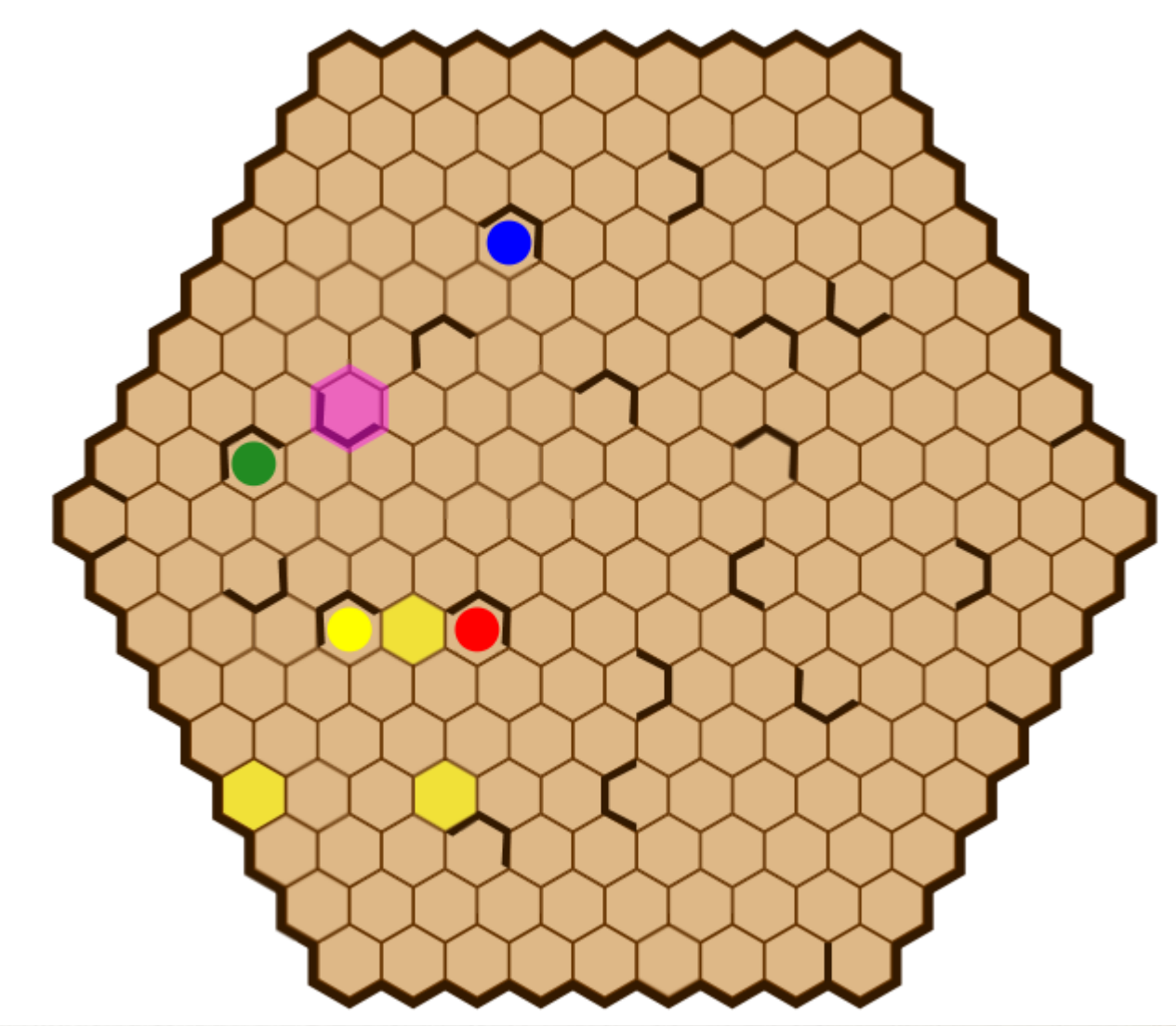
- The minimalPath( ) method we were required to write for Stage A doesn't care about where the pieces are.
- Our minimalPath( ) uses a breadth-first search
- Originally the algorithm was very slow and we had to force it to halt if the path was longer than 9 steps
- Was able to speed up the algorithm by a lot by creating an array of 217 booleans (one for each position on the board). This caps the maximum number of branches in the search tree to just 217
- Since we want the shortest path, a breadth first search never needs to try moving the piece to the same place twice!

RANDOM CONSTRUCTOR STRATEGY

1. Pick 8 legitimate crannies at random
2. Generate three random nooks within one of the six triangular segments of board.
3. If any of the nooks you just generated are adjacent, try again
4. Keep doing this till you've filled all 6 segments
5. If nooks aren't all connected, remove all nooks and go back to 2
6. Put between 1 and 4 pieces in random nooks
7. DONE

MAKING THE GUI

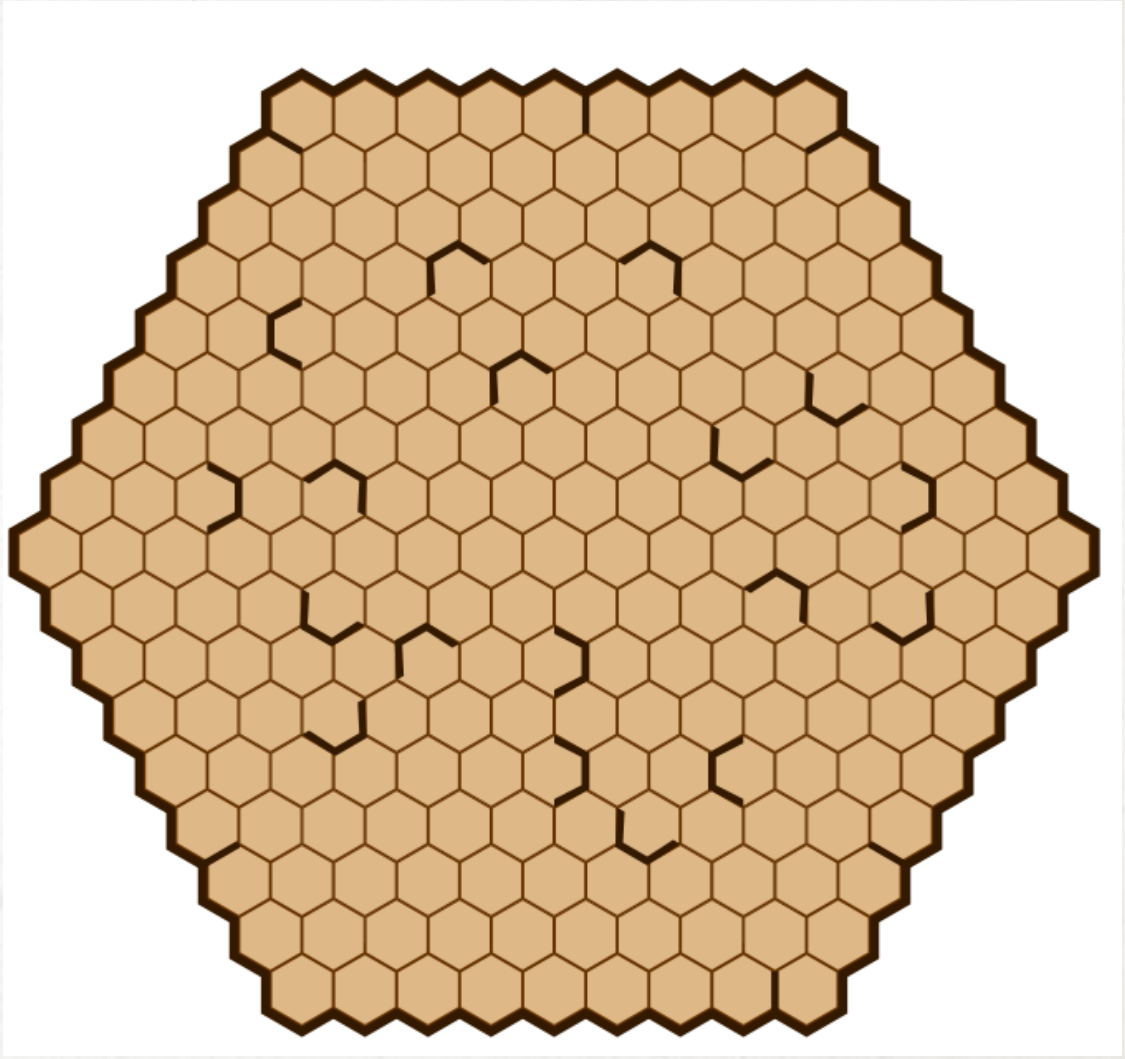
- Pieces have on-click events and animations attached to them. Game remains consistent with underlying HexGame object however, since every time a piece is moved it is destroyed after the animation, and a new one takes its place.
- Can click+drag a piece to move it. Clicking on a piece also creates flashing highlights in the positions that piece can move to. Pieces also move when the player clicks on highlights.
- The goal pulses and changes colour



MAKING THE GUI

- Began by creating a Hexagon class
- Created nook and cranny classes
- Created “DrawBoard” class whose constructor accepts a HexGame, then draws the board on the screen according to that HexGame, complete with nooks, crannies and a border

- It was a pain to get the nooks to rotate and position themselves properly!



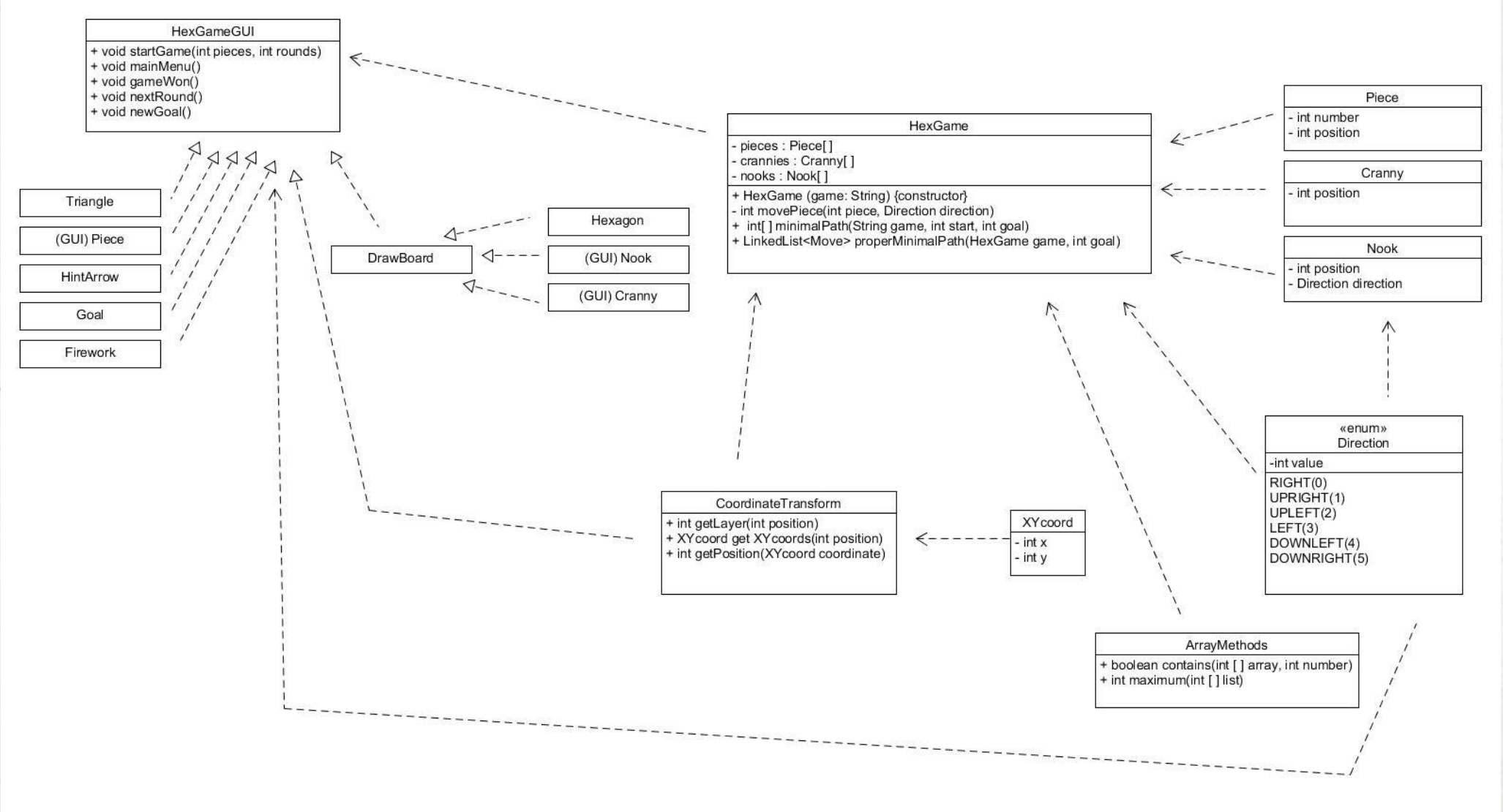
SCORE, ROUNDS, MOVES, TIME AND HINTS

- Finished game includes a timer, move counter, rounds and score
- Score is calculated by counting how many more moves you took than were absolutely necessary (lower score is better)
- Also added hint feature
- In order to implement scoring and hints needed to write another minimalPath method which could cope with lots of pieces
- Unfortunately this second minimalPath method is pretty slow and can cause the game to lag when the path to the goal is very long.

END-GAME SCREEN

MAIN MENU (STILL A WORK IN PROGRESS)

UML DIAGRAM


HexGameGUI
+ void startGame(int pieces, int rounds)
+ void mainMenu()
+ void gameWon()
+ void nextRound()
+ void newGoal()
HexGame
- pieces : Piece[ ]
- crannies : Cranny[ ]
- nooks : Nook[ ]
+ HexGame (game: String) {constructor}
- int movePiece(int piece, Direction direction)
+ int[ ] minimalPath(String game, int start, int goal)
+ LinkedList<Move> properMinimalPath(HexGame game, int goal)
Piece
- int number
- int position
Cranny
- int position
Nook
- int position
- Direction direction
Triangle
(GUI) Piece
HintArrow
Goal
Firework
DrawBoard
Hexagon
(GUI) Nook
(GUI) Cranny
«enum»
Direction
-int value
RIGHT(0)
UPRIGHT(1)
UPLEFT(2)
LEFT(3)
DOWNLEFT(4)
DOWNRIGHT(5)
CoordinateTransform
+ int getLayer(int position)
+ XYcoord get XYcoords(int position)
+ int getPosition(XYcoord coordinate)
XYcoord
- int x
- int y
ArrayMethods
+ boolean contains(int [ ] array, int number)
+ int maximum(int [ ] list)