## 3/19 - Project Started 6 pm

**Required tasks**,

* 5x5 board, playable by two players
* Implement core game loop.
  + Print board accurately
  + Allow players to take turns
  + Add lines

**Extra tasks**

* Resizable boards
* More than two players
* Game can finish
* Checks when box is complete
* Checks who completed box
* Tracks points
* Check if game is done

**For all tasks:**  
 Create a board of a size specified by the player. The board must contain dots all the dots and lines of the game. A line is between each dot.

When a specific place for a line is chosen a line will be put there. When four lines fully enclose a box, that box is considered scored. The last player to have added a line to that box gains the point for that box. If you finish a box, you get an extra turn. You can only make a box from a 1x1.

**Technical Ideas:**

There are two ways, a more functional or more object focused way.

The simple and functional way to do it, keeping classes to a minimum.

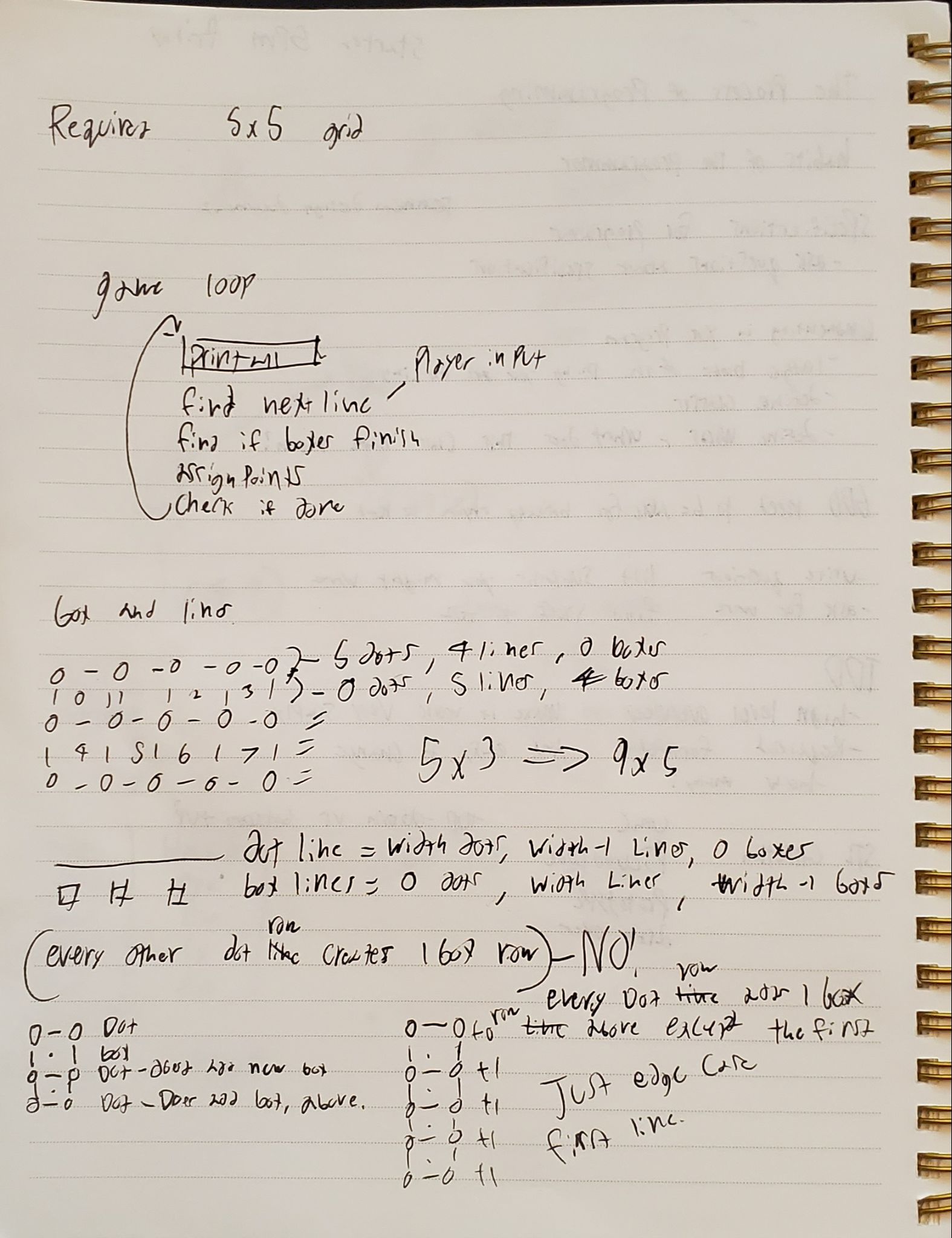
There is 1 class, for the game itself, all the logic for everything is contained within this class or just in functions around the class. This can easily just not be a class, and is only really in class in order to easily export the program for any reason. There would be functions to do all of the tasks listed above.

There would be an array of characters for all of the dots and lines in the game, this array would always be dynamically allocated. The memory should probably be allocated once when the game starts, then should be reallocated any time the board wants to change size.

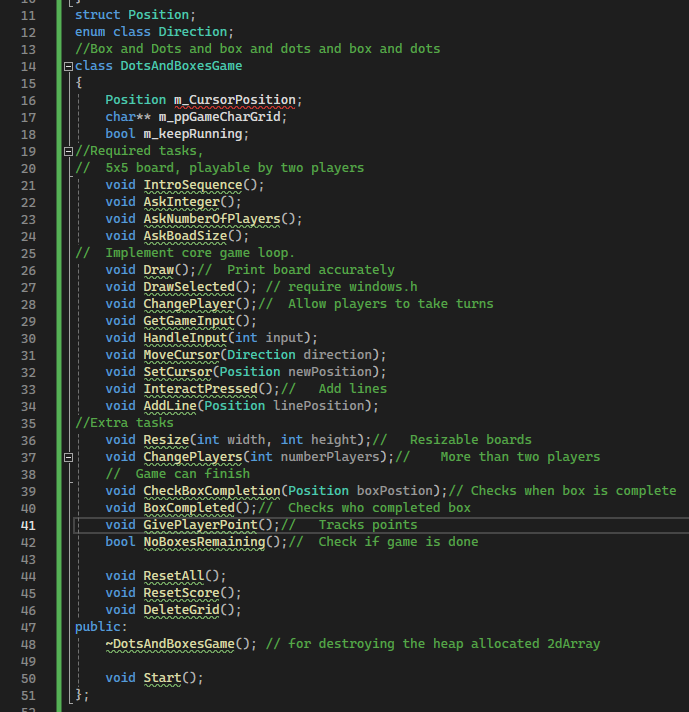
The size of the actual array needs include space for

* Every Dot Character
* Every Line Character
* Every Box ‘Character’ (this being the character surrounded by lines

The more object focused way would be, creating a class for each dot, each line, and each box. The box would keep track of every line around it, and when a new line is added, it is added to the tracker of every dot and box it can connect to. I do not see any immediate advantages of creating this project with this amount of classes, so I will start with a functional approach.



**Step 1: constructing the idea of the class(in main)**

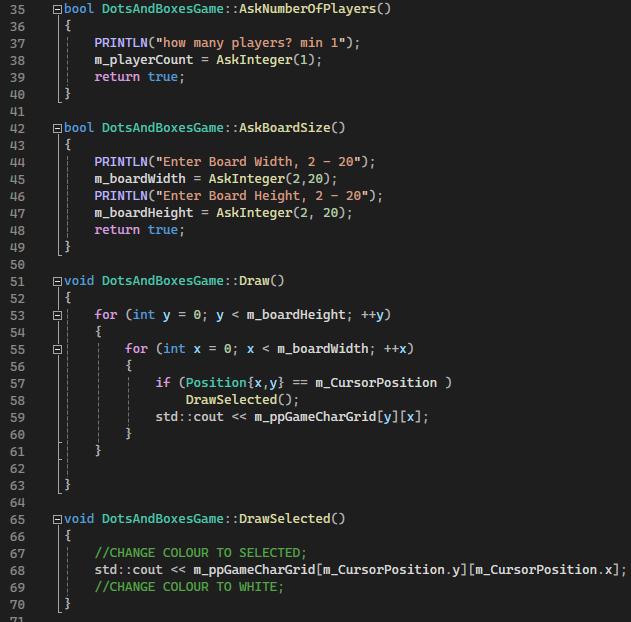


I started by listing out everything I needed from the tasks, then I made each task its own function. There were many times I decided to add extra functions because I know that in the future I will need at least 1 extra function to perform the task originally listed. For instance, “print board correctly” will require at least 1. Knowing what is correct and 2. Printing depending on the board state. To do this, I added DrawSelected to highlight the character currently selected by the player.

I also know that I will be dynamically allocating my grid, so I need to create a constructor. And also since I want to resize my grid, I added a function that just deletes the grid for me, since I will need to use this function at least twice. I also added ResetScore and ResetAll just as some basic functions in order to control the game instead of needing to write the code every time I want to change levels.

**Step 2: fill out functions and known code**

There are a lot of functions here that are maybe only one or two lines of code, so im just going through every function in the cpp file and filling them. Any time I come across code that might take a minute to think about, I write a pseudocode comment block of what I want to be able to write in the end product.



Some things that I noticed about my functions as I am doing this:  
 Some of the functions should have a return value.

I need a list of all the characters I will use in the game as well as an interact character.

I need a variable for number of players, current player, and the size of the grid.

A few functions different parameters, like Addline should ask if the line is vert or horz, and doesn't need to know the position, as it is the cursor position,

If I were to colour things to show which player has completed a specific square or not, I would need to keep track of every boxes position and remember what color got that square.

Having colors gets tricky with more than like four players, as each player needs a unique color identity. I could instead in the prototype try to complete it with a number for the player.

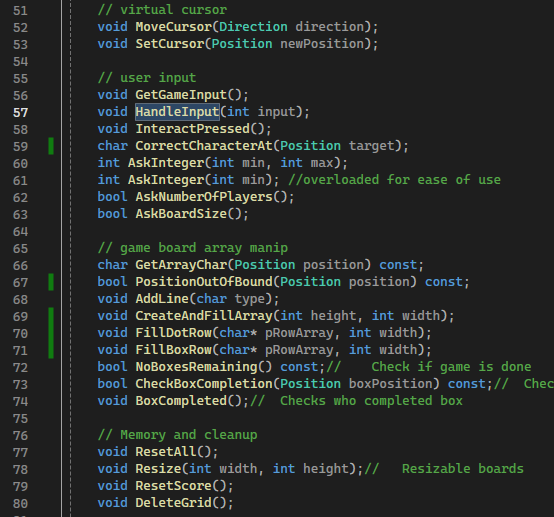
I need an array for tracking player scores

I need a boolean to track when a player has scored a box and should get an extra turn.

GivePlayerPoint() function may be redundant as I would only call it once and the function would only be a single line ++m\_pPlayerScores[m\_currentPlayer]; might as well just type instead of making a new stack frame.

## 3/20 - Project Day two

I have gotten the basic program to function in the way I believe it should. I have added a lot more functions and code than I originally hoped. There are already many places that I believe could be brought into their own classes. For instance



The functions here are somewhat sorted, but there is a lot of confusion of what is going on. I think that the board game manipulation stuff could easily be put into its own class. As well as user input could easily be put into its own class, and I could use it to do the tasks like, asking for player size and board size.

## 

## Finished Product:

The class I have for the entire game is a bit large, I wish I could trim it down to make it look a lot neater. But the intention with the project was to be much more functional with everything. In hindsight I might want every character of the game, instead of being a char literal to instead be some object that can hold data. Just to improve the ability for me to control what is going on in the game.

Perhaps I could better organize my functions from the start, and probably have had a framework in mind for how all of the program would come together. I had a good idea and framework in the class as shown earlier. However I think that If I wrote down on paper a better diagram for how I could create classes to ease the burden on a lot of this information.

Some of the specific trickiest things I worked on were,  
 1. How I determine when or if a box is completed. I thought real quick that I wanted to check every cardinal direction of where the line is placed, and check if the box at that location is then enclosed. But in practice, it just looks very strange to read what I did. There seems to be a good amount of repetition as there is no quick way to just check all the directions easily. Although I still believe that the basic idea was solid I wish I could find a better implementation of that idea.

2. User interfacing seems pretty tricky. I'm always unsure if the functions that get the input from the user should be within the class or I should find another class to do it. An easy fully static class or some free functions would easily work instead of these being in the class. This would also make the game loop look more like  
 - Draw

- GetInput

- HandleInput

Rather than the current

- Draw()

- GetInput()

The current one is theoretically less code in the main loop, but it makes it seem like I never actually handle the input from the user as the function called, get input, is actually getting input and running all the game logic.