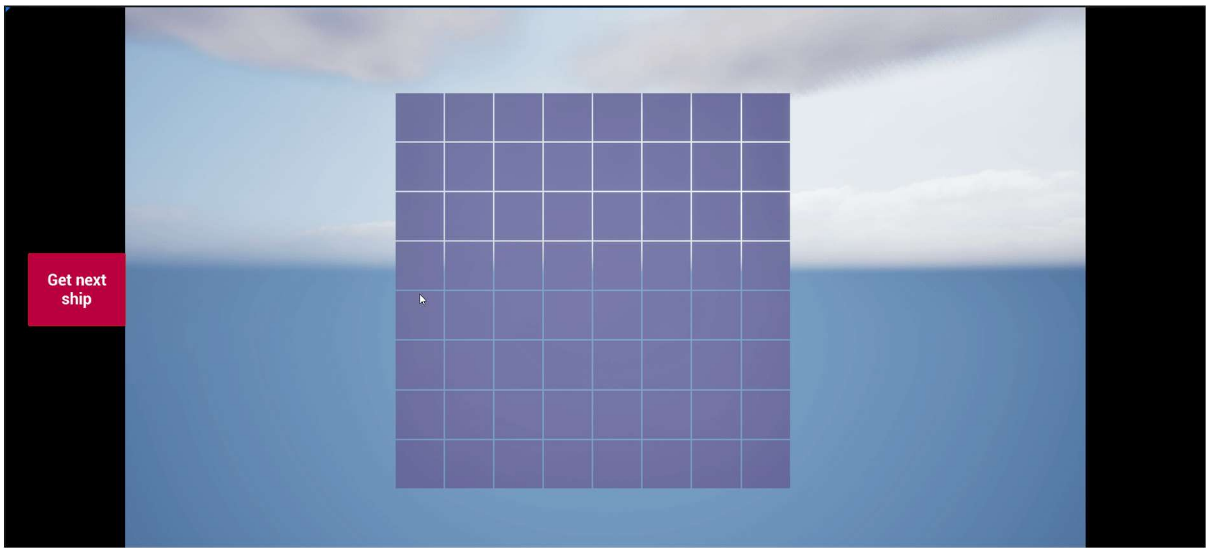
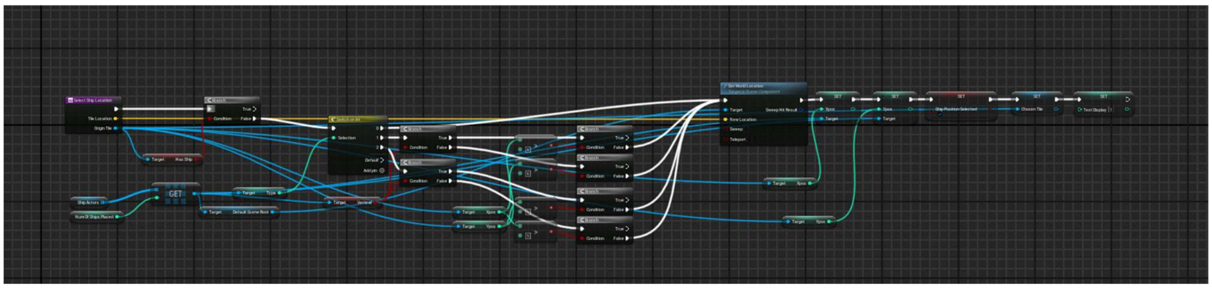
The other key component of my game was the ability to place down ships onto the grid. I initially had some difficulty when trying to implement this, but eventually came up with a system that would iterate over each cell the ship covers.

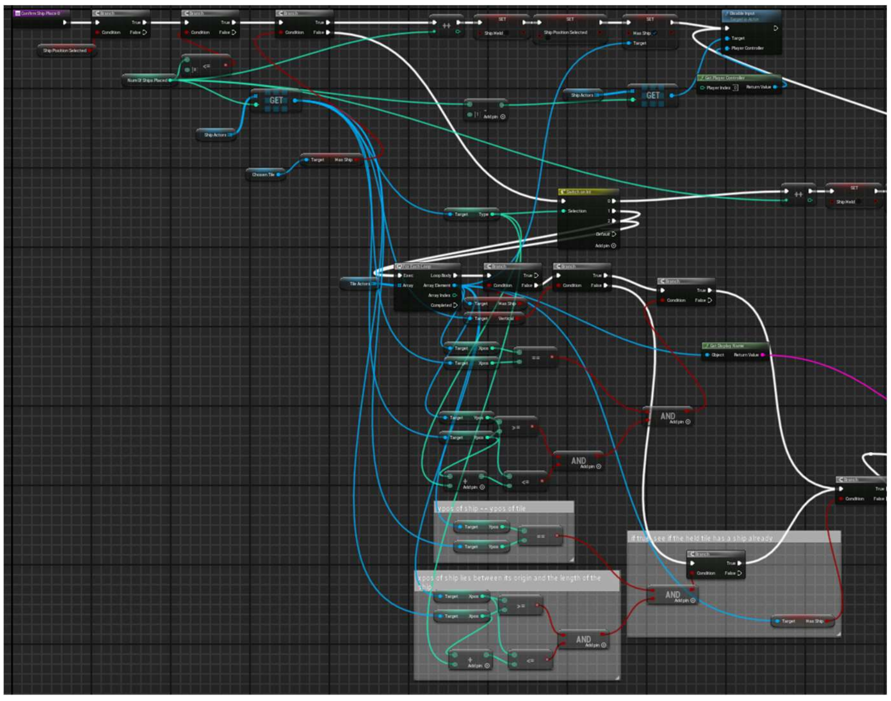




(ship placements)

Starting off, the only information I had access to when trying to place my ship, was the selected tile, the rotation of the ship, its length and the arrays which held references to the ships and tiles.

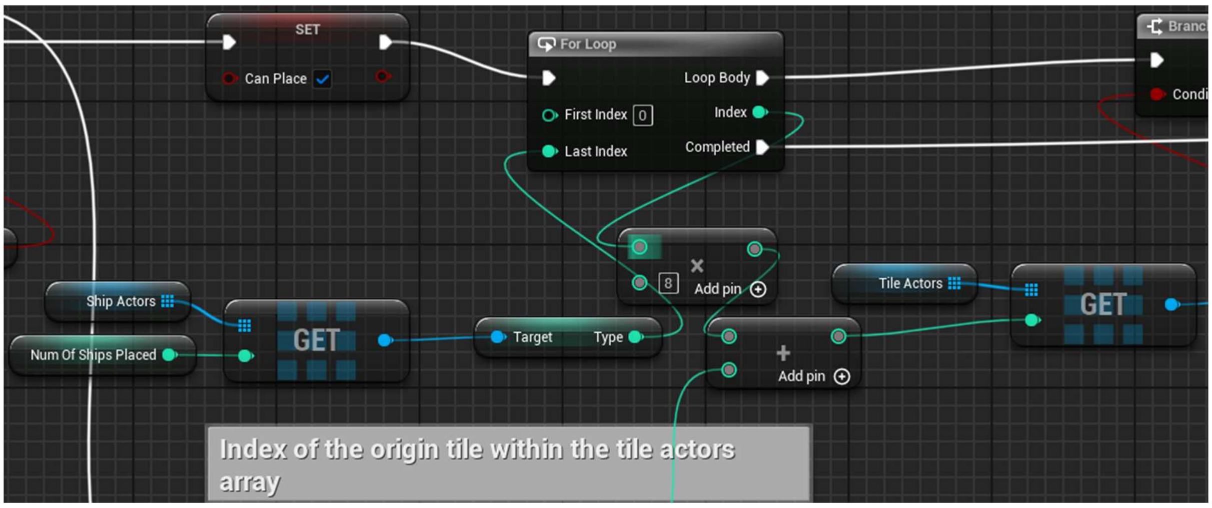
Using this, my first attempt was to iterate through each item in the tile array, then compare the ship origin X and Y position, alongside the relevant tile X and Y positions, with respect to the length and rotation of the placed ship, to set the tiles that should be occupied.



However, this very quickly turned out to be incredibly inefficient, hard to read and caused several bugs that were all very difficult to fix.

After re-considering the logic of my function, I realized that since my tiles were stored in an array, and each tile is always instantiated in order from index 0 at the bottom left, to the final index at the top right, I could use the width of the grid to manipulate the index into a set of X and Y coordinates (or vice-versa), wherein the index position = W * Y + X

This allowed me to significantly simplify my code and improve the efficiency of my function, where instead of iterating through each item in the tile array, I could get an index and target the item directly.



This improved the time complexity of the function (where the input size "n" is the number of tiles in my array) from linear time O(n) to constant time O(1), which will always be more effective. (Huang, 2020)

## *Reflections*

While my ship placement algorithms were quite frustrating to figure out at first, using a mathematical approach instead of a logical one, allowed me to greatly simplify my code. In future, I will likely continue to aim most of my functions to work on these principles, as it has proven to be more efficient, robust, and scalable.

My method of creating a tiled board worked exceedingly well and is something I will likely re-use in future projects that will require it, as I had little to no issues regarding its implementation.

## *Insight*

While this project lacked polish and visual effects, that was not the intent behind its production, as it allowed me to experiment with new techniques I had learned throughout the coding and scripting course work. Being able to apply these directly into a complete project, rather than simple demonstration programs was very satisfying, and a proof of concept for myself.

## *What I would do differently now*

If I were to re-attempt this project, I would try to formulate a more cohesive plan for how my code and game elements will pair together, as I adopted more of a “figure it out as you go” approach, which lead to my code being hard to follow and messy at times.

I would also like to re-attempt this with an artist and designer, since I feel like having a planned user experience when playing and pleasing visual effects/models could turn it into quite a fun game.