Tilemap grid  
  
At the start of the project I decided to work on the layout of the map. One of my friends send me a tutorial on how to implement a tilemap with conditional tiles. These conditional tiles are a designer friendly way to implement paths and corners within a tilemap as they look at their neighbour tiles and change their sprite accordingly.A quick map was created and I went on to make a functional path.

## Path generator

Each conditional tile also had the ability to spawn as a gameobject. Each conditional tile that is a path now intstantiate with the path prefab. This can be seen as a node.

On start the nodes add themselves to a list in the wavemanager containing all the nodes. Alongside that a simple checks happens to see if the node is the first, last or finish node so it doesn’t have to be checked on generation later on.

To make the actual path a raycast appears in each direction from the path, hitting each aligned node. Then it loops through the list of nodes starting from the first node and gets the aligned notes from the raycast. It should always have two nodes (unless it’s the last or first) where one of them is the previous checked node and the second one the new one. It loops through all nodes and creates a sorted list where index 0 is the start and index (whatever the amount of total nodes is) is the last node.

The tilemap grid had one problem, it creates itself using start and awake. I’m unable to generate a path on awake and start so this had to be moved to whenever the wave starts.

# Enemy

At this point I was told to think differently, instead of OOP use ECS ( Entity Component system). And that’s how I started working on the enemy. The enemy has a walker class which can be inherited from to make more and different walkers. For this project I’ve only made one which is the followpath. (In the future it could also attack towers or make de-routes). The walker only has the path, speed, rotationspeed and targetnode as this should be the case for every walker.

The followpath follows the path list as it’s sorted from 0 to max. Each frame it checks the distance between the enemy and the target node and if it’s less than 0.1 it checks if the current targetnode is either the end or finish (end is end of path, finish is where the damage to the player happens) node and if that isn’t the case it continues on to the next node in the list.

## Health

I discussed in the lab with my teacher Hans Wichman if it’s ok to use the “health bar” system bloons uses by changing the sprite based on how low the health is as I didn’t like seeing health bars on top of the ships and this was approved.

Whenever the enemy takes damage it tries to load the correct sprite based on it’s health. This is done through the scriptable object. A enemy can have any amount of sprite and the sprite is based on the difference between the current health and the default health it started with. This can be easily adjusted in the scriptable object it loads from.

# Towers

Once again I was thinking about ECS. A tower has a Range and shooter(s). Tower itself does not represent that much other than keeping everything together.

## Range

The range of a tower is displayed by a child meshrenderer

# ShopUI

# Building

# Wavemanager

# Inspecting towers

## Upgrades

# Enemy Statuseffects

# Scriptable objects

# Observer for goldBalance

## And more observers

# Code Review

# UIManager

## SFM pattern

## Strategy pattern

## And even a singleton

# UML Class diagram

Microsoft Visual studio has a Class Diagram Designer built in (as optional install) which quickly creates all the classes inside of a diagram. The associations aren’t automatically displayed in here only the inherited classes. I’ve sent a first draft as confirmation and was told to look at the latest mail which was about removing unnecessary methods so those have been removed as well.

## In the lab the discussion came up if towers had to be placed on a grid or if free placement was also allowed I originally thought I had a strategy pattern implemented, but after the code review I did with Wiebe this was not the case. I tried to refactor my abstract class from upgrades to implement this design pattern the intended way. I unfortunately came to the conclusion that this is not possible First I thought of making the GameState a SFM, from pausing the game to continue and game-over. But this wouldn’t work because I had to pause my enumerators for spawning waves. And for that I could do Time.timeScale = 0 but that removed the point of the gameState as SFM as it doesn’t bring additional value.