Example Tutorial: https://github.com/oddmax/unity-wave-function-collapse-3d

Our project will be to create a demonstration of wave function collapse in a unity project.

- We will use wave function collapse in our Games for Change Game for roguelike room generation, so our implementation will be 2D.
- In general 2D is also just easier to work with

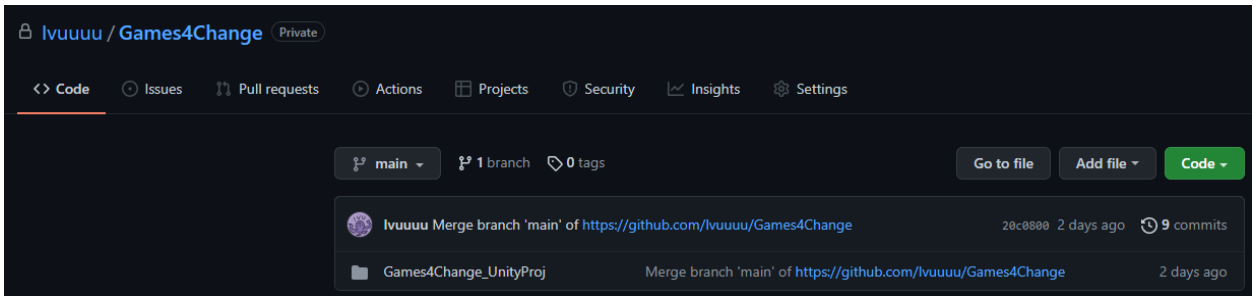
For our implementation, we will be using scriptable objects to make it easier to create specific rule sets for tiles.

- Each tile will have a section for image and a list of compatible objects for each side

The basic steps for WFC are:

1. Create a list of 6 lists that will hold all valid neighbors (one for each cube face)
    a. Whenever the algorithm collapses a cell, remove any cell types from neighboring cells that aren't in the list of valid neighbors.
2. Label each module (cube face) with a socket identifier
3. Loop over every module in the set and store the position of each vertex that sits along each edge of the 6 boundaries
4. Store and label each of these (above). This creates a dictionary of module names and profiles.
5. There is special markup needed for symmetrical and asymmetrical modules:
    a. Tag symmetrical with an s next to its socket label (ex 1s)
    b. Since symmetrical tiles will always fit with itself, we can add that as a rule for matching cube faces
    c. Asymmetrical tiles, on the other hand, will fit with a mirrored version of themselves. This means we can store an asymmetrical module as two different sockets and mark one with F (for flipped)
6. There is also special markup needed for Top and Bottom modules:
    a. For each top/bottom module store 4 versions of each socket and label them with a rotation index
    b. There is an extra check needed for these modules as well; vertical sockets will only be considered valid if they have the same socket index and rotation index

We will also be using GitHub to actively work on the project together and be documenting any specifics steps to make GitHub useful for Unity development in particular.



https://www.boristhebrave.com/2020/02/08/wave-function-collapse-tips-and-tricks/
https://www.uproomgames.com/dev-log/wave-function-collapse
https://forum.unity.com/threads/wave-function-collapse-algorithm.473423/

https://chloesun.medium.com/implementation-of-wave-function-collapse-algorithm-in-houdini-for-3d-content-generation-76f8eec573b1

https://notabug.org/selfsame/unity-wave-function-collapse
https://github.com/mxgmn/WaveFunctionCollapse/blob/master/SimpleTiledModel.cs

https://medium.com/swlh/wave-function-collapse-tutorial-with-a-basic-exmaple-implementation-in-python-152d83d5cdb1

WFC in our situation:

1. Place entry point for room
2. Check possibilities for all tiles, placing them into a list
3. Check adjacent tiles for each tile in list
    a. Update possibilities based on this
4. Whichever has lowest possibilities, generate/set a tile from the list of possible tiles (including exit - when exit is generated then generate hallway to the end of the level)
5. Repeat 3-4

In order to make sure paths are always accessible (not separated by a void), make a rule so that new paths can only be generated as long as they are connected to previously generated paths