Wave Function Collapse Initial Overview Assignment

Names: Jasmine C, Maddox P
Date: 1/24/22

Note: This document is written with the assumption you understand (conceptually) Wave Function Collapse.

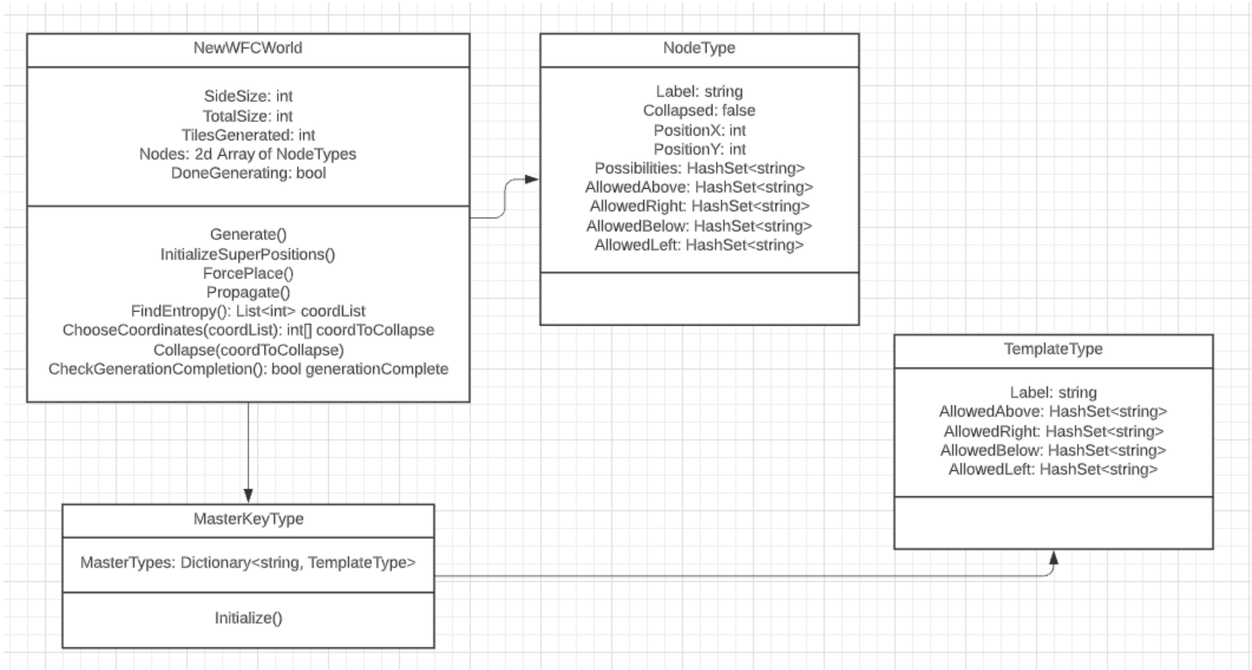
Main Goal

The main goal of this project is to create our own implementation of Wave Function Collapse (WFC) in Unity. Since we are creating the algorithm from scratch, we have been using online articles as references, specifically looking into their object modeling and pseudo code logic.

Our Progress

Initially, we jumped straight into coding each module, and failed to plan our object model effectively. This led to inefficiencies in our code, such as using a dictionary to store node information, rather than creating a custom class.

This time, we have fully planned our object model, first on paper, and then on a software called lucid chart. We have fully commented out logic on each of the scripts we plan to use, detailing what parameters we will pass in and how the main loop of the code will work. This time, we feel much more prepared which will hopefully lead to a cleaner implementation overall.


NewWFCWorld
SideSize: int
TotalSize: int
TilesGenerated: int
Nodes: 2d Array of NodeTypes
DoneGenerating: bool
Generate()
InitializeSuperPositions()
ForcePlace()
Propagate()
FindEntropy(): List<int> coordList
ChooseCoordinates(coordList): int[] coordToCollapse
Collapse(coordToCollapse)
CheckGenerationCompletion(): bool generationComplete
NodeType
Label: string
Collapsed: false
PositionX: int
PositionY: int
Possibilities: HashSet<string>
AllowedAbove: HashSet<string>
AllowedRight: HashSet<string>
AllowedBelow: HashSet<string>
AllowedLeft: HashSet<string>
TemplateType
Label: string
AllowedAbove: HashSet<string>
AllowedRight: HashSet<string>
AllowedBelow: HashSet<string>
AllowedLeft: HashSet<string>
MasterKeyType
MasterTypes: Dictionary<string, TemplateType>
Initialize()

[Object model made in LucidChart]

Important Definitions

Item	Definition
Node	Each “Tile” in the room. A good comparison is an unfilled cell in a sudoku grid.
State	Each Node will eventually only hold one state, or possible tile of information. For example, in sudoku the number 3 would be a possible cell state.

Algorithm Steps

Step #	Step Title	Step Description
1	Initialize Superpositions	Fill all nodes with all possible states. Update Node objects in 2D array.
2	Force Place	Only called if necessary:
While Not Fully Generated		
3	Propagate	Use information to cross off any possibilities that are no longer possible. Update Node objects in 2D array.
4	Find Entropy	Find the node(s) with the lowest entropy. Return a list of all possible coordinates to choose from.
5	Choose Coordinates	Take the returned list from step 4 and choose one of the coordinates from the list. Return coordinates
6	Collapse	Take the selected coords from step 6 and choose a possibility out of the possibility list.
7	Check Completion	Check a counter for collapsed tiles vs. total tiles to generate, which will tell us if the tiles are finished generating.
When Generated		
8	Spawn	Loop through each node, spawning the correct GameObject.

Useful Tools

VS Code Live Share

While trying to create the main WFC algorithm, we ran into GitHub merge conflicts since Git checks line numbers instead of content. Our solution was to use a VS code plugin called Live Share, which allows us to work on the same document at the same time, only saving the changes to one person's computer. It's essentially google docs for VS code, and helps us manage merges.

- Link for Live Share Download: https://code.visualstudio.com/learn/collaboration/live-share

Note: When you join the live share, make sure to unfollow the session leader by going to the Live Share tab and clicking on that person's name.

LucidChart

To make our UML Diagrams, we used the website LucidChart, which has a UML template already available.

- Link for LucidChart: https://www.lucidchart.com/pages/

PitFalls

Before this project our team hardly used pseudocode and UML diagrams other than when they were required for classes; however, after starting this project we realize how important these tools are when designing complex algorithms.

PLAN OUT YOUR OBJECT DIAGRAMS BEFORE YOU START CODING.

It is also a good idea to completely comment on the function logic before you start coding. This way you understand how the parameters of your functions will work together, and what each function does.

Things to Work On

- Will code data structures first and confirm they work together
- Will begin coding the WFC from the bottom up