**Geog5995 – Coding for Social Science: Core Skills – Supporting Document**

*Intention of the Software*

This project represents my first major coding project starting from scratch creating an agent based model using the Python coding language. The title of the project is ‘Town Planning for Drunks’ and the intention was to model the journey of customers leaving a pub back to their own home, the journey is tracked and density information for all agents is presented in a visual output and saved as a CSV file upon completion of the code running, allowing the end user to see the density (step) count for each point in the map that could be used within a town planning context.

The model first reads in raster file data made up of numbers between 0-250 on a 300x300 grid and is made up of a pub in the centre of the data and 25 houses situated around the pub with each house having an assigned value of a multiple of 10 ranging from 10-250. Agents are then added to the model which are created by importing information stored within an agent class, there are 25 agents each with a unique home that will move one at a time until each reaches their destination. The agents move based on the generation of a random number between 0 – 1 dictating which direction the x/y coordinated move, upon each move the individual step count increases by 1 step as well as the density value of the point they currently occupy. A range of visual outputs are created when the code is run show the different stages of the process, this is partly to show that the code is working and partly to show exactly what is happening, step counts for each agent are printed for the user to see along with an average step count, this information could be used a planning context to test changes such as pub opening/closing times to ensure people are not out too late or changing access routes to and from the pub.

*Issues During Development*

Model development was a learning process and a process of trial and error on my part to reach the final version, this project was chosen from the given list as I found it the most relatable to previous experience gained from the practice exercises. The basics of creating an agent based model were based on this although I did create this model from scratch trying to build on that foundation of knowledge rather than adapting previous code. The main issues faced in the project are in the following areas;

* Agents moving one at time rather than collectively.
* Assigning agents having a home and moving until they reach this rather than iteration based.
* Agent Individual step count
* Mapping Density
* Getting Agents not to retrace steps – I was unable to solve this.

Many of these issues came down to and could be solved through gaining a better understanding of the rules of coding languages and python logic. The use of ‘for loops’, while statements and if and else statements solved a lot of the issues faced, getting them in the correct order and properly indented was an issue at first, initially I could get each agent to move once before the loop finished rather than each agent moving until reaching the destination.

By far the most complicated problem I faced was assigning each agent a home destination, while getting them to stop when they got there was solved using a combination of the for, while and if commands. This was a long period of trial and error in which I tried many different approaches before finding something that worked, to get this to work in the agent class the agents have ‘self.house = number’ and ‘self.home = False’. The number that dictates the agents house and is defined when the agents are made, when the model is run and they are created using the length the of the number of agents; ‘for i in range(num\_of\_agents): number =(i+1)\*10’. As the number of agents is 25 starting at zero this ensures each agent will be assigned a multiple of 10 relating to the environment raster data. Self.home = False is used in conjunction with the while command, as long as the patch the agent is standing on is not their home number the loop will continue.

The density issue was solved with the reading in of two identical environment maps, the first is the one the agents are created on and interact with, in the agent class the agents have access to both ‘self.environment’ and ‘self.environmentdensity’ so each time they occupy a set of coordinates on the environment the +1 for density is added to the corresponding point on the environmentdensity file, finally the environmentdensity is written and saved to a new file called ‘densitymap’ so each time the model is opened the starting environments are not affected.

The issue I was not able to solve from the brief was stopping agents retracing their steps, it is something I spent a lot of time trying and believe this will be a case of working out the logic but I have not been able to do that. I tried experimenting creating another value similar to density where if the value was above 0 meaning it had been stood on the agent could not walk on this, the issue I had with this was implementing this to work with more than one agent as it could work initially but there was very quickly nowhere the subsequent agents would stand on. Another issue faced with this method was agents getting stuck with nowhere to move so the model would run indefinitely. One way around this I had in mind was to implement some kind of memory where if an agent stood on a patch they couldn’t stand on it for X amount of moves and after a while they would forget they stood on this (they are drunks after all) but I was unable to implement this, also there is a step count and average steps, these vary dramatically each time the model is run often averaging over 100,000 steps so setting the memory value needs more thought. This is something I would like to try and address as my coding skills increase and will return to.

*Sources*

For the coding part of the project I relied heavily on the lecture slides delivered in the week long taught module phase, these helped significantly in the understanding of class types, data types and the logic how to get the programme to run. For tougher elements, I did look at Stack Overflow for advice and help although I never took code directly from the site, the issue I had here was the advanced level of questions and answers available that was often far too detailed compared to the problems faced, although it did provide a useful source of information and suggestions for further reading. The python documentation also was used in understanding the basics of the code being used, numpy and matplotlib libraries were also used for the averaging part of the code and mapping parts of the code respectively.

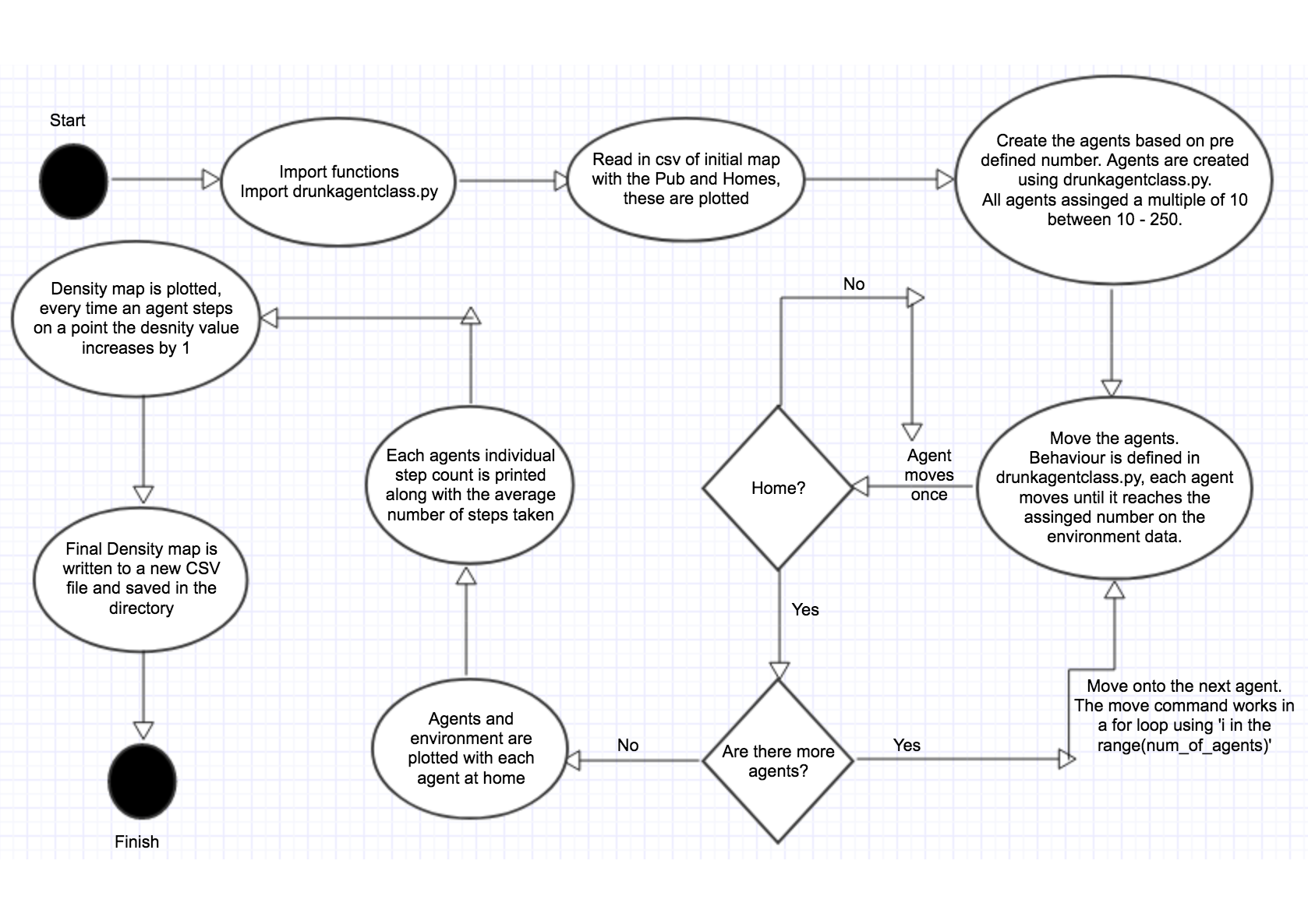
*Software Design and Software Development Process*

The software development process took on an incremental approach, I prefer to break things down into smaller subsections that fit into the big picture. As such I kept a basic algorithm in mind throughout the process which can be found in comments at the top of my model;

* Read in the data
* Model the agents interacting with the environment
* Process the data
* Write the data to a new CSV file

Keeping this mind ensures the basic requirements of the model were achieved and whilst additional elements were added and some took longer than others it was a good blueprint to stick to.

The software design followed the same approach, focusing on one element at a time such as creating the agent class, reading in the data, handling the movement code, producing agent step counts and finally the density map, the model built up slowly over time following a logical progression building to the finished version. I made a series of rough sketches of the model flow to keep things clear and ensure the model worked as planned; the final UML flow diagram demonstrating how the model works can be seen below. This was made using the Gliffy Service (https://www.gliffy.com/).



*Usability*

The usability of my programme was something I considered thought-out the design and production. My understanding of usability is that the code works with minimal errors or bugs or if they exist they will be clear for the user to rectify to run the code, as well making clear what the code is doing and how it can be used. Within usability I also consider readability such as the layout of the code in terms of spacing and comments, in this context I have presented my code in a clear and consistent way with space between each set of instructions along with detailed comments explaining what each line is doing, the layout confirms the python standard with proper indentation where required.

In terms of usability I’ve made every effort to ensure no bugs or errors will arise when the model is run, the only potential issue I can for see is potential issues reading in the data files if directories change, however I have explained what files are needed in the Readme and where to find an alternate version of the files. Using the model should be a case of running the drunkagentsclass.py followed by drunks.py within Spyder.