Documentation:

GEOG5995 Planning for Drunks project

**General sources used**

The code generated for this assignment was developed using skills I learnt through lectures and practicals on this module, online training courses including the CDRCs Introduction to Python module lead by Fran Pontin, and websites providing advice and suggestions in Python (typically in regard to error messages and how to solve them).

The environment data which was read into the model at the start was provided on the VLE within the materials for the GEOG5995 module for the town planning for drunks assignment. The algorithm for the code was also provided here, the idea of the project is not my own but was offered as an assignment option.

My code is also available within my Github repository from the following address:

<https://github.com/elliemarfleet/PFSS-Python2>

**The intention of the software**

The intention of the software is to build a model documenting drunk people leaving a pub and returning to their home. The model first reads in a raster file made up of numbers on a 300x300 grid. Each of the 25 drunks is given a unique ID which matches their house number (a number between 10 and 250). The drunks move in different ways dependent on how drunk they are; i.e. their *alcohol\_levels.* Upon each move the individual step count increases by a value of 1, as does the density of the point they currently occupy.

This loop continues until all of the drunks have left the pub and returned to their home. Upon reaching their home, the drunk remains there and does not move again. Once all of the drunks are home, the model stops running and a density map is produced recording how many drunks have passed through a point.

The files *drunkmodel.py*, *drunkframework.py* and *drunk.plan.txt* are all needed for the model to run successfully. It is recommended these files are saved in the same folder in your directory prior to running any code, see the final subheading ‘*How to run and what is expected’* on page 3 for exact details on running the code.

**Issues during development and how these were overcome**

The development of the model was a learning process and very much a process of trial and error to reach the final version. The basis of the model was inspired by the code produced for the first assignment for this module - the general trajectory of the steps taken and the definition of the drunk class in particular were aided by the code I produced for Assignment 1 (although largely adapted).

Initial problems I encountered related to the paths the drunks would take. I was unsure whether to have multiple scenarios in which the drunks could either;

* Walk directly home
* Move around randomly in the environment
* Inflict some conditional argument
  + i.e. the further away the drunk’s house is from the pub, the more random their walk home would be
  + the more drunk they are, the more random their walk would be

Originally my code only defined the drunk class and the move function, which I then expanded upon to include some conditional arguments based on how drunk the agents were when they began their journey.

I was also unsure how to prevent the drunks from retracing their steps (if they hadn’t returned home yet), although I tried 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, but I struggled to implement this to work with more than one agent.

**Software design and software development process**

The software development process abided to the algorithm provided on the VLE, as outlined at the top of the model:

*1. Pull in the data file and finds out the pub point and the home points.*

*2. Draws the pub and homes on the screen.*

*3. Models the drunks leaving their pub and reaching their homes, and stores how many drunks pass through each point on the map.*

*4. Draws the density of drunks passing through each point on a map.*

*5. Saves the density map to a file as text.*

I kept these steps in mind during the design stages, having sections of code generally completing each task in methodological steps. I built upon issues outlined previously during the development stages to inform my software development, with particular reference to the way in which my drunks would travel home. This was done through the use of if/else statements, i.e. if a person is drunk (their starting alcohol level is 100), they will stagger home moving at integers between 1 and 3. The staggering stops when the ‘drunk’s’ alcohol level reaches 0, demonstrating their new capability to walk home with ease. I felt this was the most efficient walk trajectory I trialled, although I’m aware other (potentially more complex) movements could have been made!

I made some rough sketches of the model flow to ensure my model worked as planned during the early design stages, and the final UML was built using the Gliffy service (<https://www.gliffy.com/>) and is available on my Github or below:  
A picture containing diagram

Description automatically generated

**Testing**

Testing of how the environment looked was provided within the code and was often commented out as it was not necessary for the model to function correctly, but rather ensured the environment had been set up and ran in the right way. For example:

﻿﻿plt.xlim(0,300)

plt.ylim(0,300)

plt.imshow(environment)

for i in range(num\_of\_drunks):

plt.scatter(drunks[i].x,drunks[i].y)

plt.show()

Testing of names such as ﻿*print(drunk\_ID)* was also provided in the code to ensure variables had been assigned correctly.

**How to run and what is expected**

The model has been produced in a Python script, so an appropriate Python software must be used when it is run (e.g. Spyder, part of Ananconda). Ensure matplotlib is also installed prior to running the model if this has not been done in the past.

To run the model, the Python scripts must exist in the same directory and the *drunkframework.py* file must be ran first, as this defines the class code which will be used to inform the model. Open both files, run the *drunkframework.py,* then run the *drunkmodel.py* file and the code will be produce an output of a *density.txt* file in the same directory as your Python files. A figure (“Figure 1”) will also be produced as part of the output in a new pop-up window, showing the movement of the drunks through the environment before they reach their homes. Lighter colours show the most walked areas of the environment.