# Geog 5995: Assignment 2

## Intention of the software:

This software is designed for the “Planning for drunks” project.

The criteria given are as follows (from the course website):

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.

## The software development process:

This section describes the software development process. Each paragraph corresponds (roughly) to a Github commit.

I started by creating the environment from the raster file provided and using a loop to figure out the location of the pub.

I then created a rudimentary agent framework and plotted the agents in the environment.

I then ensured that problems with agents hitting the environment border were prevented and animated the movement of the agents in the environment.

I then attempted to create the houses and assign one to every drinker. However this was unsuccessful.

Instead I decided not to create objects for the houses and simply assign an address to every agent instead and have the agents check if this address matched the value of their location after every step.

At this point I had a working model. The next step was to record the output of the model as a density map. Issues I had doing this are outlined in the next section.

I then added comments to explain the code and a sys argument so the file could be run from command prompt with varying numbers of iterations without having to change the code in the file (figure 1).

Finally I changed the input to be more robust to invalid entries by running the model with a default number of iterations if a value error occurred (ie: if a non-integer was inputted).

**Issues encountered:**

The main issues encounter had to do with producing the density map.

To record the density of drunks passing through each point on a map, I created a function called leave_footprint which had the value of points in the environment increase by 1 whenever an agent passed through them.

However this lead to:

- Potential false positives. For example if a point had been visited ten times it would have a value of ten. If the drunk who lived at house ten landed on that point, they would falsely believe they had arrived home and be removed from the model.
- Potential false negatives. For example if the drunk who lived at house ten landed on a point in house ten that another agent had passed through, they would falsely believe that they had not find their way home.

To resolve these problems I changed the function to **decrease** the value of the environment by 1 whenever an agent passed through them, and added an ‘if’ statement which has agents check that they aren’t on a house before changing the values of the environment (Figure 2).

However this ‘fix’ caused another issue with the way pyplot renders the changes in the environment. When the environment value is increased the relevant point on the animation becomes brighter. When the value is decreased it becomes darker making it hard to visually track how much agents pass through a given point.

A last issue that was revealed by visualising the footfall density, and having the agents not mark densities on the areas they think are houses, is that the location of the houses as plotted by pyplot and the location of houses as 'understood' by the agents were not aligned.

This is visible is figure 3 as there are 'footprints' on the location of the houses, and empty areas (some circled in red for greater visibility) where there are no footprints, these must be where the agents 'think' the houses are.

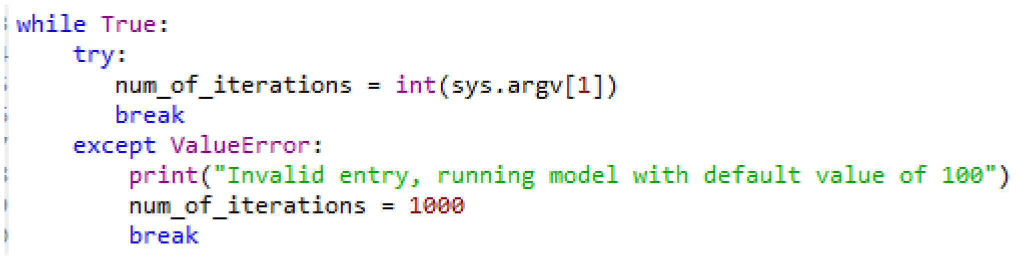
It is unclear why this is the case as the houses are not offset in any regular direction.

The second issue is that the model takes a very large number of iterations for all the agents to find their way home (the shortest I have managed in 12 000 iterations). I tried a few ways to make this faster such as playing with different movement lengths for the agents and changing the way border issues were resolved but these did not lead to noticeably improved performance.

Further work would involve finding ways to help the agents get home faster. However this would first require solving the indexing issue described above which I have been unsuccessful in doing.

**Figures:**

Figure 1:



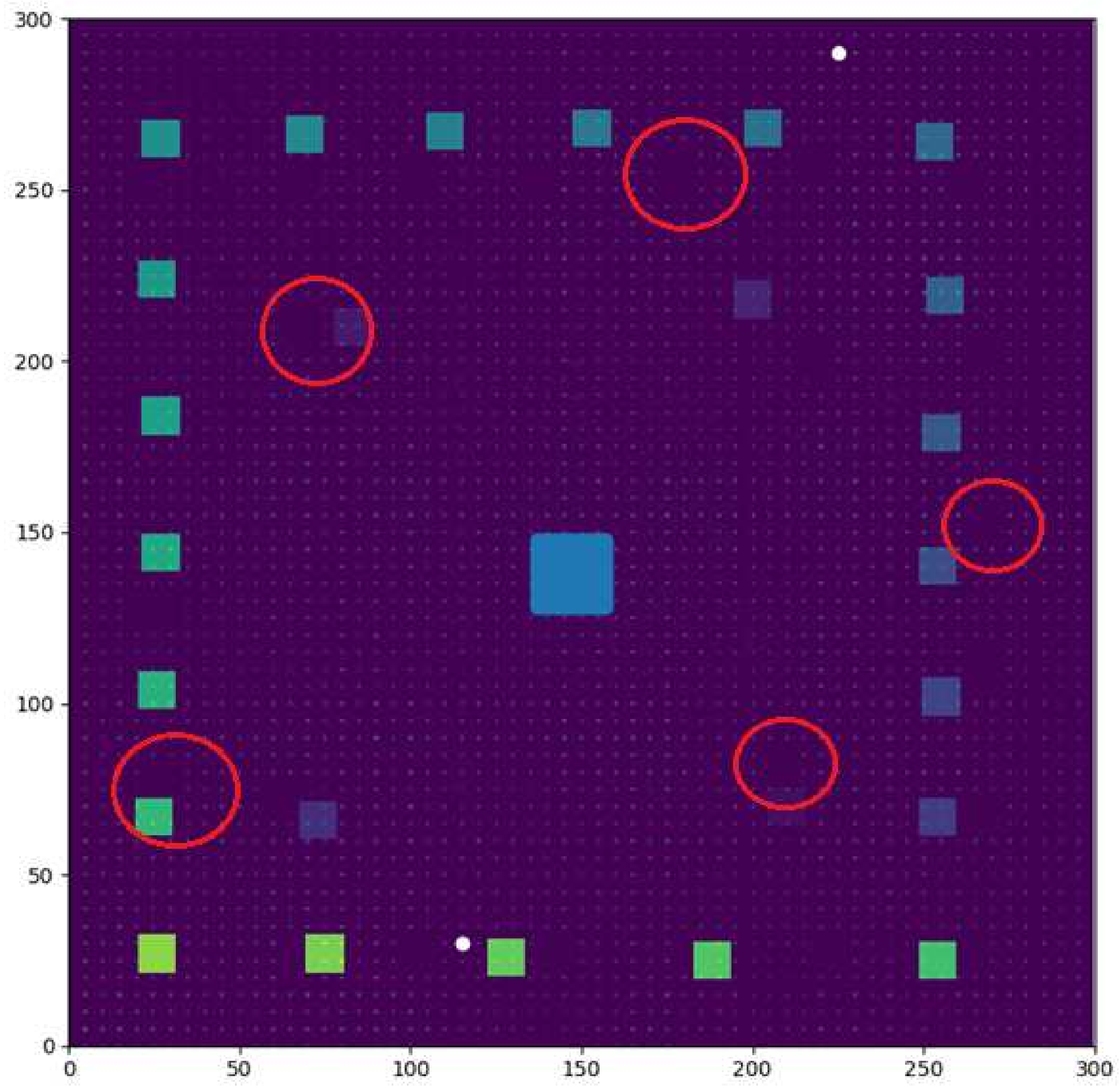
```
while True:
    try:
        num_of_iterations = int(sys.argv[1])
        break
    except ValueError:
         print("Invalid entry, running model with default value of 100")
         num_of_iterations = 1000
         break
```

Figure 2:

```python
def leave_footprint(self, pubfinder, adresses_set):
    #this ensures that the houses are not renamed if agents pass through them
    if pubfinder.iat[self.x, self.y] in adresses_set :
        pass

    else:
        self.environment[self.y][self.x] -= 1
```

Figure 3:



300
250
200
150
100
50
0
0
50
100
150
200
250
300