Intention of the software

This Agent-based Model follows the 5+1 requirement of “Planning for drunks” assignment. It reads in data from drunk.plan, locates Euclidian centroids of raster representing the pub and homes, as well as visualizes then on map. It then processes the movement of drunkens from the pub to homes, one by one. Drunkens are prevented from retracting previous locations by a Judge() function with command info. It also visualizes their steps to homes, stores their stay in any raster as values of “3” on map. It eventually saves the “stepped” map as a csv file.

From a user perspective, users are expected to config the number of iterations in this ABM, confirm then run the model through buttons in interface. Afterwards, users should identify the movements of drunks from pub to homes. Users should be informed by differences between drunks’ moves that the movement is factorized by the alcohol in-take: the numbers of drunk homes.

From a programming perspective, this ABM practices basic concepts of object-orientated programming. It builds different attributes of a class: drunk, in an independent module. It import this class for interaction in the ABM main script. Within this process it encapsulates class attributes by hiding private variables. It shows the advantage of object-orientated programming in terms of efficiency: In a case all drunks use the same map instance: environment as their internal attribute. They change the values in their internal environment as their presences in locations. Because their internal environments just different “tags” of the same “instance” environment, all calculations of steps are automatically stored in the environment.

Thought processes going into the software design

This ABM consists of the map, locations, drunken class, and drunken class’s methods, and a file output. The drunken.plan is a raster file thus it naturally expects a raster representation in ABM: a raster image. It is realized as reading in the .plan file as a 2-D list via the “csv” Python package. It consists of rows, with columned values inside, locations are identified by serial numbers in rows/columns and given nominal numbers. Drunken class in essence should be point data on map, with their locations gradually move from pub to homes. These coordinates are grabbed from the index of map rows and columns. The methods of drunken class are changes in their current coordinates, based on Euclidian-Boolean conditions with their homes. For example, if a drunken is in a radius to home, move this drunk arrive home, else move it randomly in this iteration. There could be another raster-based interaction between drunks and locations. For example, if a drunk is in raster of its home, then define it as arrived. However, this approach needs more coding while its display on animation is similar to the Euclidian approach. This ABM elaborates movements of drunks into 2: a conscious movement and another unconscious movement. The conscious movement is change drunk’s location to the next quarter numbers on the way home. This approach secures the general efficiency of ABM that it avoids meaningless random movements of drunks, which in most instances are opposite the way to home. This approach is adjusted from average numbers to quarter numbers which reduce the “teleports” of drunks. The unconscious movement is simplely random movements on 4 directions. This ABM use the coordinates of drunks as indexes, when changing map values as the proof of drunks’ presence in any locations. The eventual export of “stepped” map is also based on csv package.

Software development process and Issues

The development follows what in module slide: refine requirement (as discussed in previous section), divide code (into 5 parts in source code; ABM environment, variables & parameters, functions, interfaces, and export), develop, test, and refactorize.

The develop, test, and refactorize are based on 3 phases: version 0.1, 1.0, and 1.1. Version 0.1 organize basic components. Version 1.0 provides primary debugs for an operatable ABM, it also solves an advanced issue that the writing-in of steps in map causes “bleach” of colours, by reducing the nominal value of step to 3. Version 1.1 corrects the iteration between drunks movements, one by one. It also provides better visualization of drunken status of red (in-moving) and green (at home). This version has issue of unobvious visualization of steps and mismatch between drunken and map coordinates. Version 1.2. is the final version, it corrects coordinate mismatch through exchanging the inputs of x/y coordinates of drunken class. It better visualizes steps through a similar scatter plotting of step coordinates like drunken coordinates. In this version it also realizes basic config of ABM in interface.

General sources used

This ABM is structured from the material of module GEOG5990, especially the practices and assessment 1. One additional source is about reading in .plan as list: <https://www.programiz.com/python-programming/reading-csv-files>. The other is the solution of changing text string to numbers in 2-D list, from: <https://stackoverflow.com/questions/28376538/removing-quotation-marks-from-list-items>

Potential uses

This ABM demonstrates the traces of drunks going back homes on the periphery from a pub on centre. It could be helpful to other ABMs as an example, for example some tourists have a vogue direction to a landmark, thus experience different routes.

In terms of the result trace patterns, it is a net pattern radiating a whole map. This pattern is the same with the layout if Napoleonian Paris, by Baron Haussmann. From an urban planning perspective this pattern maximizes the accessibility between any 2 locations in a city. In this ABM’s regard, the efficiency of tracing back home is decided on alcohol in-take, more drunk individuals tend to have more inefficient accessibility. Therefore, this ABM might indicate an analysis in the research of accessibility. For example, introduce and replace alcohol with other factors like slopes, traffic jams, in analysing different levels of accessibility in a system where expected to be homogeneous.