**Python Software Development Guide – Bacterial Bomb**

**Introduction**

The python software developed to model the Bacterial Bomb scenario outlined by University of Leeds has four basic components, read in the data, process the data, display the output, and write the output to file. This guide provides detail of the development process based around the outlined model components.

* A brief description of what the software can achieve from the end user perspective, based on the scenario.
* The theory behind the design including good practices and Unified Modelling Language (UML) diagram.
* The software development process in chronological order, a diary-like format.
* Any issues encountered in chronological order, and how they were rectified.

**Intention of the Software**

**Thought Process of Software Design**

**Software Development Process**

To ensure the model had a set location in order to map out the spread of the bacteria, a raster dataset was provided called wind.raster. This raster file was read into the model using the csv.writer function, and a 2D list was created from the raster values. This list was used to create the area for the model, and contains almost all pixel values of 0 and a single pixel of 255 to represent the epicentre of the bomb.

Chart, histogram

Description automatically generated

*Fig 1. Created Area from wind.raster*

Set up the basic framework for the animation including creating the figure and making the update function. Configured the animation parameters and update rules then show the animation.

Using functions from the numpy module found the bomb epicentre from the 2D list ‘area’, as this has a unique value of 255. The returned index values are used as the starting coordinates for the bacterial spread, however it is important to remember it returns in a YX order, as opposed to the traditional XY. Afterwards replaced the value with 0 to level off the area plain.

**Issues During Development**

Whilst attempting to retrieve the 255 index position from the 2D list ‘area’, the initial attempt was made using the enumerate function, which is ideal for indexing 1D list values. However this proved too challenging for the 2D list, so the alterative workaround was to convert the list to an array and use the functions available in numpy. Fortunately the array variables created were not required beyond the specific job mentioned, which reduced the risk of potential data compatibility issue in later code.