II

Building artificial Neurons in three dimensional space

# Introduction

We have discussed the possibility of three dimensional mesh neural networks and the potential for artificial neurons. In doing so we have touched on some issues we may encounter in larger mesh networks and security issues whilst developing and designing such functionality.

During the course of the second iteration we will be building neurons and deploying artificial intelligence within their core to called as a callback function. The artificial intelligence will look at deploying and making use of MST, logging, reporting and providing return information for analysis.

We will deploy a three to six neuron setup with various neuron attributes and make use of a simple dictionary data set with specific output requirements. These neurons will simply be housed within a mathematical bounded three dimensional space and have all of their respective connectivities in place either hard coded or autonomous if we deploy them at random space points within the volume.

In doing so we will also look at the dormant neurons within the volume and research a best practice to randomise with a little more even distribution in a volume whilst still maintaining ‘organic’ distribution. We will make use of standardised algorithms to assess their functionality and examine efficiency of both deployment at random compared to hard coded and in comparison to existing results as if the artificial intelligence algorithm were simply a single entity.

With that in mind, we will perform those tasks to examine whether artificially, more neurons is better than one before proceeding to examine whether all neurons should have the same artificial intelligence or whether clusters of variable artificial intelligence can interact well.

They will be build in a shared folder on a guest virtual machine operating system which has had the virtual networking card stripped out after the updates and the necessary IDE installed.

# Volume

When setting the volume metrics, we have two options; We can set mathematical bounds such as ten cubed giving us 1000 dots matrices to work with or we can use three dimensional ArrayLists which tend to get a bit cumbersome when trying to find out which matrix location each neuron has. In addition, each matrix location will be housing another multi dimensional ArrayList and makes for very difficult decompression and confusion in the programmatic section.

What will be best for the volume is to set perhaps a simple mathematical representation yielding mathematical volumetrics and better precision in neural proximities. Further more, scalability will be a lot easier to manage when embedding to the hardware. What we end up with are mathematical boundaries and neural proximities pin pointing programmatic memory locations at their neural epicentre.

The issue we will encounter is programmatically indexing the mesh layers and ensuring the correct depth, though, it may be possible to simply set integer type precision for the volume and float data types for the neural proximities permitting better precision when connectivity is assessed.