**Project Outline: Simulating 2D Diffusion-Limited Aggregation and Brownian Trees**

*Document requirements:*

* *Outlines the physical system I will model (including key properties of the system: mass, charge, fields etc), any Python packages I will use, any equations describing the system I will model, any approximation methods, and any programming techniques I will use*
  + *Abstraction and inheritance to build more sophisticated classes, to model objects with more complicated properties, or chaotic systems with a wide range of possible motion*
  + *What sorts of random number generators will I use? How will I store the data and analyse it?*
* *How will I test functionality of the simulation, and what is a simple version of the system that can be used to validate the simulation? Need thorough testing of all individual components*

===================================================================================

In this project, I aim to model two-dimensional diffusion-limited aggregation, illustrated by the so-called ‘Brownian trees’ generated as a result of this phenomenon. Diffusion-limited aggregation (DLA) is a process involving the aggregation, or clustering, of particles undergoing Brownian motion. When such particles are allowed to adhere to a central seed point, line or any other chosen shape, clusters (Brownian trees) are formed with distinct tree-like shapes. This is a demonstration of fractal geometry at play in the natural world, finding many practical applications from electrochemical deposition to snowflake formation.

To begin, I intend to model simple random walks, wherein randomly moving particles wander from a starting point. Assuming a constant step size, I will model a series of small, discrete particles in 1D, 2D and then 3D, using *numpy.random.choice* to choose a unit step in a given direction. In all dimensions, the average displacement after N steps should be zero, since a particle has an equal probability of travelling in any one direction. Meanwhile, the rms displacement should be. I will test these probabilistic characteristics to ensure the model is as random as possible, and hence not biased.

Since random walks act as a discretisation of Brownian motion, taking a smaller time step will allow convergence towards a Brownian motion simulation, which is a time continuous stochastic (random) process. This will act as a simple model upon which to build a more complex DLA simulation. I will use a (pseudo) random number generator to ..

Based on this Brownian motion class, I will extend the model to DLA by introducing a ‘seed’ particle around which

In order to make this simulation as efficient and accurate as possible,

How will I structure things - classes

I will explore the variation of different parameters, such as:

* number of particles (‘walkers’)
* step size (and whether or not it is constant -> biased)
* boundaries of the system
* speed of the particles ???
* ‘sticking coefficient’ (changes density of tree)
* ‘attractor geometries’

In addition to the regular modules (e.g. *numpy, matplotlib, scipy*), I intend to use the following:

* *random* and *numpy.random.randn* to generate pseudo-random numbers
* *numpy.random.choice* to randomly choose given values
* *pandas* to store and save data in dataframes
* *matplotlib.animation* to create simple animations of random walks/Brownian motion
* *pygame* to host the application for running the final DLA simulation
* *pytest* to test functions and classes

If time allows, I intend to extend the 2D DLA simulation to three dimensions (including visualisations), and analyse the corresponding physical properties/change in parameters accordingly.