# Diffusion-limited aggregation simulation

Mateusz Kocot, Krzysztof Misan

23rd January 2023

## 1 Introduction

Diffusion-Limited Aggregation is a process in which particles move randomly and form clusters. Their movement resembles the Brownian motion. The whole process can be presented on a 2D or 3D canvas. In the beginning, the first particle (or a group) is attached to a given position, for instance, in the middle of the canvas. The next particles can attach to the growing aggregate when colliding with the existing structure. Eventually, various shapes can be formed in the simulation.

In this project, we created an application capable of simulating the DLA process in a configurable way. We focus only on the 2D case. We uploaded the source code with animated examples to a GitHub repository: `https://github.com/MatiXOfficial/dla-simulation`.

## 2 Application

In this section, we describe the main components of our Python application. At first, we present the technical details of the simulation. Then, we describe the implementation of the attractor feature. Finally, we show the user interface.

### 2.1 Simulation

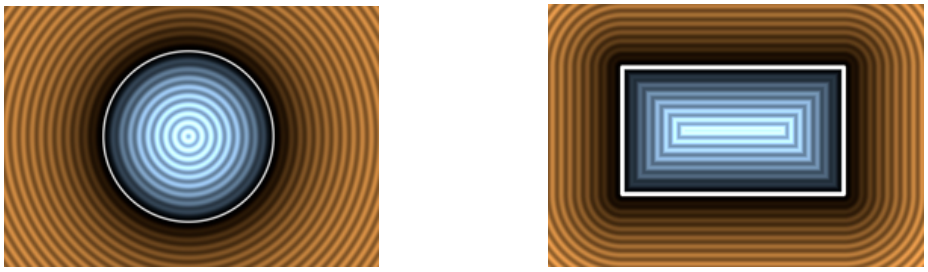
The simulation starts with initialising the grid in the shape of a square. There are different modes of initialisation. The default mode is generating a single particle in the middle of the grid. Three other modes involve starting either with the bottom edge, all edges or a circle centred in the middle.

Now that we have the first particles attached to the figure, we can begin the simulation itself. A single simulation step starts with putting a particle on a random, unoccupied field, and moving it randomly until it finds the figure. Then, it is attached to the grid in its last position before the collision. In a single move, the particle can choose one of its eight neighbours as in the Moore neighbourhood. In the default mode, the distribution is uniform. However, in the next subsection, we describe a mechanism used to modify the probabilities.

Finally, the simulation ends when the selected number of particles has been attached to the figure.

## 2.2 Attractors

In the basic DLA algorithm, the probability of a particle moving in a given direction is equally divided between all of the available spaces. We can consider a setup where this probability assumes non-equal values for different directions in euclidean space. Further, we can assume the existence of the surface within experimental space with a positive or negative attraction that is influencing the probability distribution of the particle's movement. This object, hereinafter referred to as attractor, was implemented in our work using SDF (Signed Distance Field), which is a field capable of deriving distance to the examined surface from any given point in space. In this work, we are focusing solely on the two dimensional euclidean space. Using SDF we're capable of driving the movements of a particle towards or away from the surface of the attractor. In our implementation, we are assuming that the influence of the attractor is inversely proportional to the distance (decreasing with the distance). Furthermore, the weighting formula is amplified for the negative distance which results in a significant negative force within the attractor, preventing particles from getting below the surface.



SDF - circle SDF - unoriented rectangle

**Figure 1:** SDF visualization – source: `https://iquilezles.org/articles/distfunctions2d/`

## 2.3 User Interface

Our application is shipped with the Graphical User Interface. We present it in Figure 2. It allows live visualisation of the simulation and adjusting the configuration. The parameters are:

- Refresh – how often to refresh the simulation (every simulation step, every attached particle or periodically – every 10 seconds),
- Init mode – how to initialise the simulation (middle, bottom, circle),
- Canvas size – size of the grid (canvas size x canvas size),
- Image target size – final number of particles,
- Enable attractors – if true, the attractors are included in the simulation,

- Coloring – if true, the colour of particles will change during the simulation,
- Produce GIF – if true, the app saves images during the simulation so that they can be grouped into a GIF later,
- GIF frequency – frequency of saving GIF images (in particles),
- GIF FPS – used when saving a GIF.

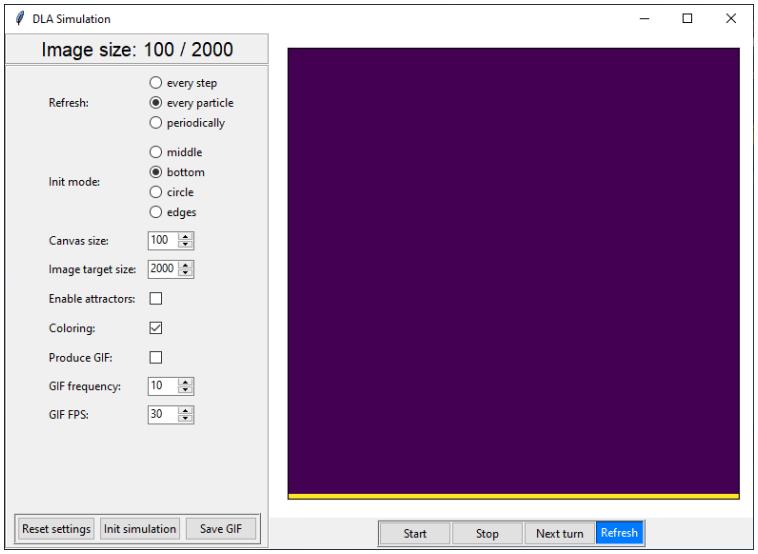



**Figure 2:** GUI

A GIF can be saved by clicking a button. It will use all the images in the *tmp_gif* folder. These images are saved only if "Produce GIF" is set to True. Note that the *tmp_gif* is cleared when the simulation is initialised with "Produce GIF" set to True.

The attractor settings can be modified in a JSON file which is loaded when the simulation is initialised. There are two types ("type" parameter) of the attractors: "sphere" and "rectangle". In the case of "sphere", one has to define "radius", whereas in the other case – "a" (height) and "b" (width). Apart from that, the user has to define "position" (e.g. [1.1, 0.5] is above the top corner, horizontally in the middle) and "force". An optional parameter is "negative". It is set to "true" by default, but if changed to "false", the attractor will attract particles instead of pushing them away.

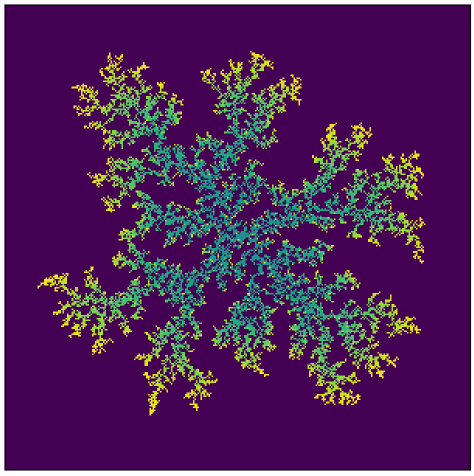
An example config is printed below.

```
{
  "attractors": [{
    "type": "sphere",
    "position": [1.1, 0.5],
    "radius": 1,
    "force": 5
  }]
}
```

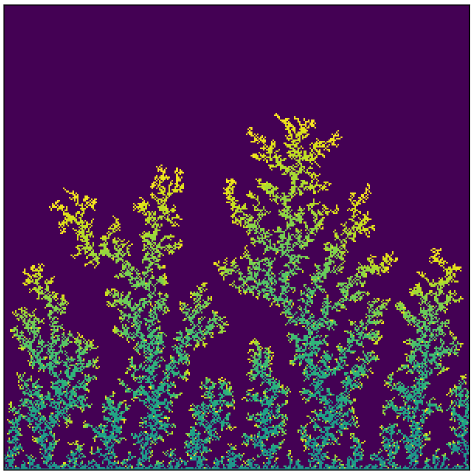
# 3 Examples

We present the final states of variously configured simulations. Respective GIFs can be found on the main page of our repository.

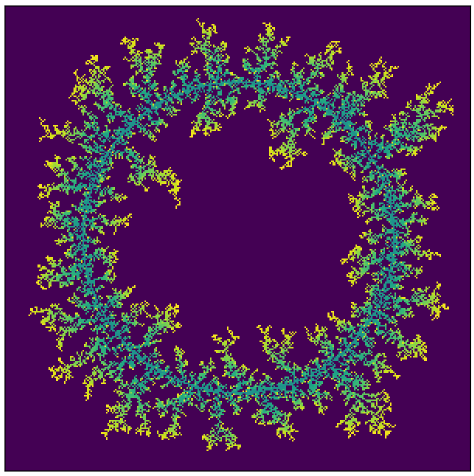
## 3.1 Examples without attractors



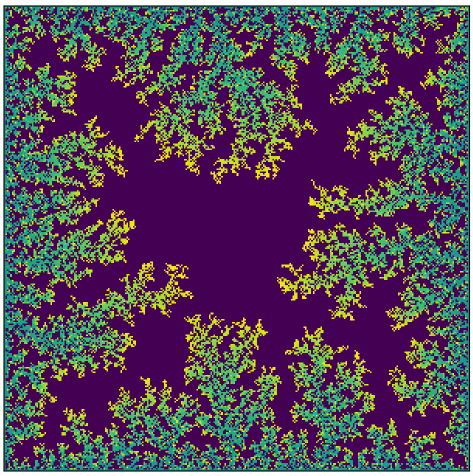
Initialisation mode: middle



Initialisation mode: bottom



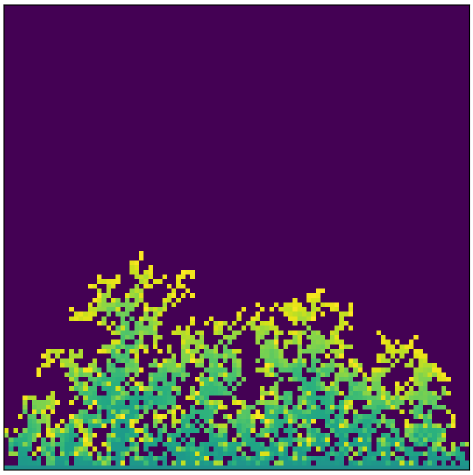
Initialisation mode: circle



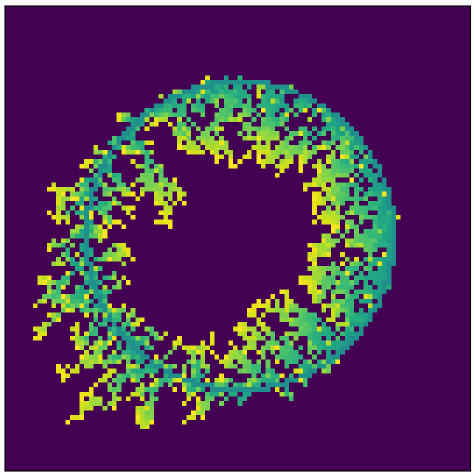
Initialisation mode: edges

## 3.2 Examples with attractors

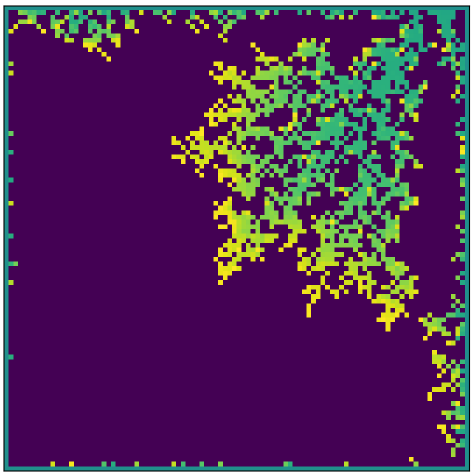
Initialization: middle, 200x200, 7500 particles, circle negative attractor at x:0.5 y:1



Initialization: bottom, 100x100, 2000 particles, 3 circle negative attractors placed in pyramid shape



Initialization: circle, 100x100, 2000 particles, circle positive attractor at x:1 y:1



Initialization: edges, 100x100, 2000 particles, circle positive attractors in 3 corners