**Ayesha Ali**

[**aali151299@gmail.com**](mailto:aali151299@gmail.com)

**Generate an Attractor**

**Melbourne, Australia**

The way I started this problem was to first understand what an attractor is. I first started to watch YouTube videos and online articles related to the concept. I then realized it is a very theoretical concept in the field of physics and different types of attractors such as Lorenz attractors and strange attractors. All this was so confusing until I finally understood each particle in time moves in 3 dimensions x,y and z. One practical example is let x be wind speed, y be temperature and z be humidity. The position of particle wrt to the 3 coordinates with time determines the weather like hot, cold or rain. This movement can be determined with an equation dependent on multiple variables. This model has also been used to the understand the movement of air within the atmosphere which is dependent on multiple variables such as speed, density, temperature, and altitude which Edward Lorenz had simplified with many assumptions and its visualization was that like a butterfly, hence the butterfly effect. I also noticed that how this phenomenon became multidisciplinary with not just being limited to mathematics and physics, but also art and philosophy with the idea that everything is interconnected and how small events can have large consequences. This made me understand that the CEO of Arcitecta got inspiration from art to write a software for handling data to unveil its full potential. I also started to understand the applications in simpler circumstances such as the movement of a pendulum and rolling of a marble. It will follow a particular motion in time unless an external resistance stops it.

Once I understood the problem, I had to implement it. I decided to make a simple web application using React as I am proficient with React from my professional experience with Fonterra. I decided to use dark mode for the application because I find it more aesthetic. After figuring out the basic UI the main task was to make an attractor. Initially I thought of making a 3D attractor using the library called [react-three-fiber](https://docs.pmnd.rs/react-three-fiber/getting-started/introduction). But I realized there are a lot of complexities with making a 3D visualization like projection and lighting. Then I decided to switch to 2D visualization using [react-vis](https://github.com/uber/react-vis) to keep it simple. The next step was to decide the attractor equations. I remembered from my university physics that sin and cos equations are harmonic in nature. I wanted to go with 1 variable only to make it simple. I chose x = cos(t) and y = sin(t). I decided to increase the interval from 0 by 0.01 to get a smoother as with 0.1 it was a linear curve. I decided to use setInterval so that the function is called with a delay and clearInterval which will clear the interval ID.

The tricky part for me was to implement the recording and replay attractor at a point in time. The recording feature was still easy to implement because all I had to do was store the attractor data from the point of time the recording starts till the point of time recording stops by the user. Main issue is replaying it. The main questions were, how to update the UI to show the replay has stopped and what should the attractor look like after that and how to go about cleaning up the data. I implemented replay feature with a click of button and run the recorded data by calculating the time duration from the length of recorded data. I decided to run the simulation after recording and then replay. After multiple test cases, I noticed my laptop was running slow and the simulation after replay was behaving weirdly even after using clearInterval. I think there is a cache or memory issue affecting replay.

I have included a snippet of the code:

const toggleReplaying = () => {

setReplaying((prevReplaying) => !prevReplaying);

setAttractorData([...recordedData]); // Set attractor data to recorded data

const replayDuration = recordedData.length \* 10; // Calculate duration of replay in milliseconds

console.log(replayDuration);

const replayInterval = setInterval(() => {

setReplaying(false); // Set isReplaying to false after the replay duration

clearInterval(replayInterval); // Clear interval after replay duration

}, replayDuration);

};

I came up with another approach in which I decided to restart the simulation after replaying and using a for loop to keep track of the index in recorded data. With this approach I noticed that the new simulation was not running. Then I used conditional rendering which also did not work. I decided to implement the replay in a different way that the user will enter the time stamp based to which it wants to see attractor to start again. I was able to implement but the old interval was not being cleared even though I was clearing the interval every time the time changed in a useEffect. I also tried to introduce another variable to track if the interval is running or not and clear it if it is running and the time is updated but it did not work. With limited time, I could not get the replay to work despite trying multiple approaches.

To summarize, I have implemented the attractor simulation with time intervals between 0 and 1 and recording functionality. You can see the recorded data in the console to verify that. But the recorded data cannot be shown in the simulation and when the user enters a time to reset the simulation, the previous simulation is not being cleared. I also wanted to put music (Oppenheimer OST) in the simulation background and modify it as the simulation replays but could not implement it because of time. Overall, it was a great learning experience as I got to explore a new concept in physics, implement a software from scratch and explore Arcitecta.

Links:

<https://www.youtube.com/watch?v=aAJkLh76QnM>

<https://www.studysmarter.co.uk/explanations/physics/classical-mechanics/attractor/>