HYDRA CODERS

Problem Number : 02

ASTEROID COLLISION SIMULATOR

PROBLEM UNDERSTANDING

INTRODUCTION:

The goal of this simulation is to model asteroid motion in a 2D space, detect collisions based on specific parameters, and output the results accordingly.

KEY COMPONENTS OF THE PROBLEM:

Each asteroid is characterized by its position (x, y), radius, and velocity (v_x, v_y). A collision between two asteroids occurs when the Euclidean distance between their centers is less than the sum of their radii.

CONSTRAINTS:

The input data format includes fields such as asteroid ID, position (x, y), radius, and velocity (v_x, v_y). The simulation is executed in fixed time steps (e.g., 0.1 seconds) over a predefined maximum period. The output will include a list of collision events, with the asteroid IDs involved in each collision.

SOLUTION APPROACH

SOLUTION

- 1. Input Parsing: Read and validate the asteroid data file.
- 2. Simulation: Calculate the position of each asteroid over discrete time steps.
- 3. Collision Detection: Compare distances between asteroids at each time step.
- 4. Output Results: Record collisions with the time of occurrence and IDs of colliding asteroids.
- 5. Visualization: Animate the trajectories and add enhancements like a space-themed background.

Algorithm Flow:

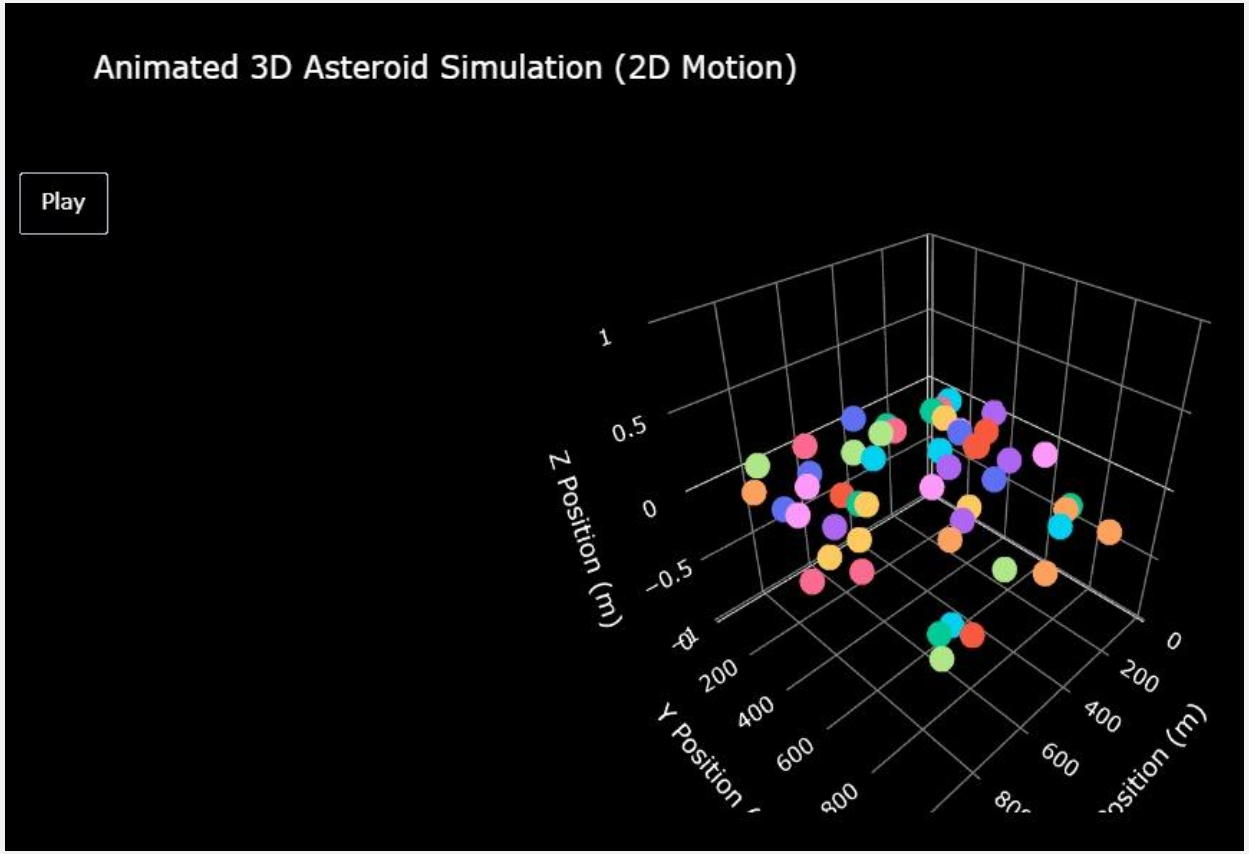
- 1. Parse asteroid data.
- 2. For each time step:
- - Update the position of all asteroids.
  - Check every pair for collisions.
  - Record collisions and move to the next time step.
- 3. Output the detected collisions.

IMPLEMENTATION

- Programming Language: Python
- Key Functions :
- 1. parse_input() – Reads the input file and converts data into asteroid objects.
- 2. detect_collisions() – Simulates motion and identifies collisions using the Euclidean distance formula: \[\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \]
- 3. write_output() – Writes collision events to a text file.
- 4. animate_simulation_3d()– Creates an interactive 3D visualization using Plotly, featuring asteroid trajectories.-
- Key Libraries: - math for distance calculations. - numpy for handling time steps and array-based computations. - plotly.graph_objects for 3D visualizations.

RESULTS

- Collision Detection Output:
- - Example from collisions.txt:
- 0.5 1 2 1.2 3 4 2.5 5 6 (Format: [time] [asteroid ID 1] [asteroid ID 2])-
- Visualization Output:
- - Include a screenshot or example of the 3D plot showing asteroid trajectories with the space.
- Performance: - Handles up to hundreds of asteroids efficiently for 10 seconds with a 0.1-second time step.



TEAM COLLABRATION

- Roles and Responsibilities:
- Member 1: Worked on input parsing and data validation.
- Member 2: Implemented collision detection and distance calculations.
- Member 3: Developed and fine-tuned the 3D animation using Plotly.
- Member 4: Coordinated the integration of all components, added background image, and ensured output compliance.
- Collaborative Effort: - Regular team discussions to refine the approach. - Integrated testing to validate all functions before final deployment.

REFERENCES

- - Key Python Libraries and Resources:
- 1. Plotly Documentation](https://plotly.com/python/) 2. Numpy Documentation](https://numpy.org/)
- 3. YouTube Tutorials for 3D Plotting in Python:
- - Plotly Tutorial: 3D Animation in Python: [YouTube Video](https://www.youtube.com/watch?v=JLsJ1CFl_9k)
- - Advanced Data Visualization with Python: [YouTube Video](https://www.youtube.com/watch?v=5P9jozRbUrE) - (Replace with your favorite links if necessary.)-
- Image Source: - [Night Sky Purple](https://www.oxplore.org/sites/default/files/inline-images/Night%20sky%20purple.jpg)

THANK YOU