HYDRA CODERS

PROBLEM NUMBER : 02

ASTEROID COLLISION SIMULATOR

PROBLEM UNDERSTANDING

INTRODUCTION: - THE GOAL IS TO SIMULATE ASTEROID MOTION IN 2D SPACE, DETECT COLLISIONS BASED ON GIVEN PARAMETERS, AND OUTPUT RESULTS.

KEY COMPONENTS OF THE PROBLEM: - ASTEROIDS ARE DEFINED BY THEIR POSITION \((X, Y)\), RADIUS, AND VELOCITY \((V_X, V_Y)\).

THE COLLISION CONDITION IS MET WHEN THE EUCLIDEAN DISTANCE BETWEEN ANY TWO ASTEROIDS IS LESS THAN THE SUM OF THEIR RADII.

CONSTRAINTS: - INPUT DATA FORMAT INCLUDES FIELDS LIKE ID, POSITION, RADIUS, AND VELOCITY. - SIMULATION IS EXECUTED IN FIXED TIME STEPS (E.G., 0.1 SECONDS) OVER A MAXIMUM PERIOD. - OUTPUT INCLUDES COLLISION EVENTS AND CORRESPONDING ASTEROID IDS.

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: (Optional) Animate the trajectories and add enhancements like a space-themed background.

Algorithm Flow:

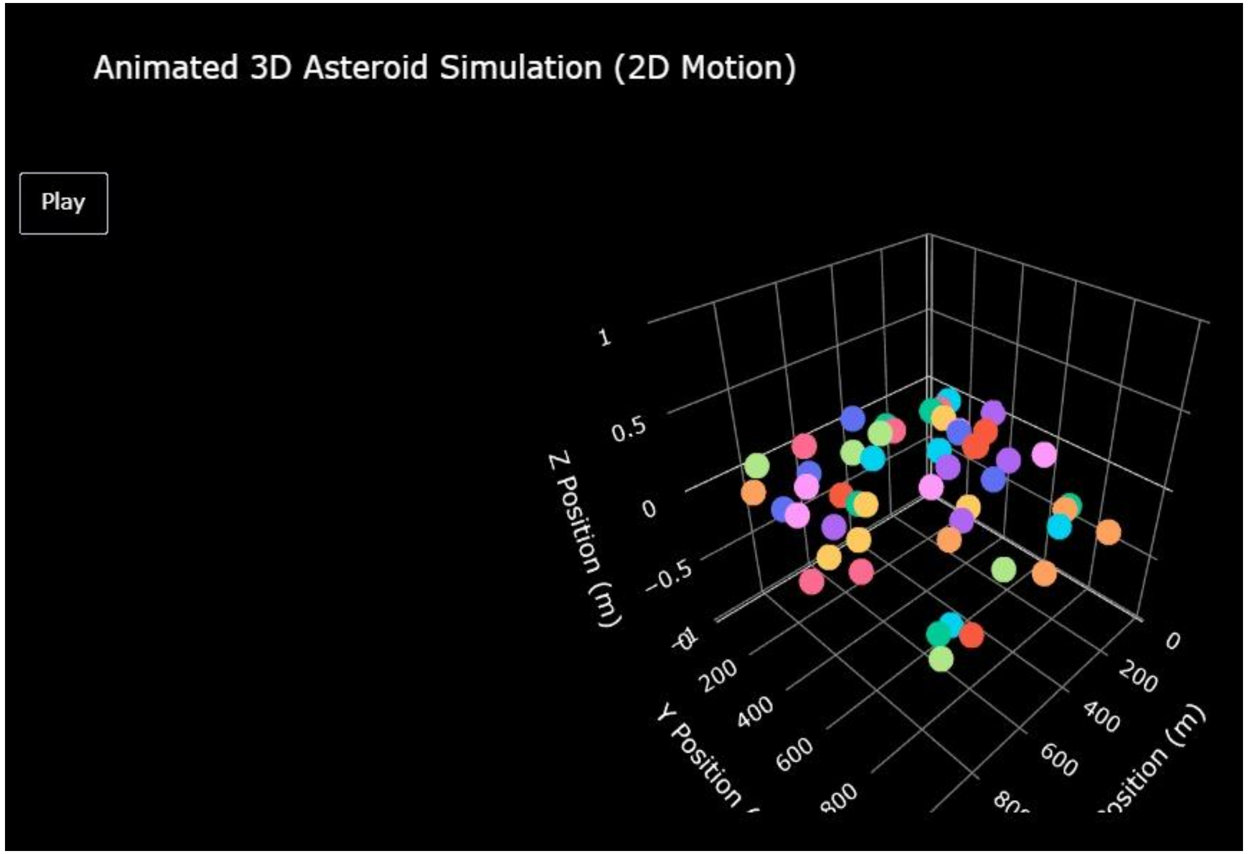
- 1. Parse asteroid data.
- 2. For each time step:
- - Update the position of all asteroids.
- - Check every pair for collisions. - Record collisions and move to the next time step.
- 3. Output the detected collisions.

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