METEOR MADNESS

Presented By

INFINITY EXPLORERS EG

Yousef Abdallah Moaaz Farouk Mostafa Hatem Mennatalrahman Alaa Ibrahim

Origin of the idea

The Meteor Madness project aims to simulate Near-Earth Objects (NEOs) and visualize their potential impact on the planet. The system provides two main scenarios:

Impact Simulation – modeling the destructive consequences of an asteroid collision.

Planetary Defense Simulation – demonstrating possible mitigation strategies (e.g., deflection or fragmentation).

What makes this project unique is its integration of real-time NASA API data with an interactive simulation environment. By combining authentic data with dynamic visualizations, the project creates an engaging and educational experience for users.

PROJECT OBJECTIVES

01

Collect real asteroid data using NASA's Open API. and Compute kinetic energy and damage potential using physics-based equations.

02

Visualize results on interactive maps to illustrate impact zones.

03

Provide users with an interactive experience to choose asteroid size, impact location, and defense strategies and Deliver a simple, intuitive user interface.

Project Structure

The project is structured into four main components: app.py serves as the central controller, managing API requests and coordinating data flow; calculations.py handles all physics computations related to asteroid impact parameters such as mass, velocity, energy, and environmental effects; simulation.py is responsible for visualizing and simulating impact scenarios on an interactive map; and requirements.txt lists all necessary dependencies to ensure smooth installation and reproducibility.

01.

app.py
Main controller that handles API requests, filters the data, and communicates with other modules.

02.

calculations.py
Contains physics formulas for calculating asteroid mass, velocity, impact energy, crater size, shockwaves, tsunamis, and seismic effects.

03.

simulation.py
Responsible for rendering the interactive map and simulating the impact scenarios.

04.

requirements.txt
Lists all dependencies and libraries required to run the project.

OUR Workflow

01

The user launches the application and selects:
A custom asteroid with user-defined parameters.
Or a real asteroid retrieved from NASA’s NEO API.

02

app.py fetches and processes the data.

03

calculations.py computes physical effects: energy, damage radius, crater formation, tsunamis, and seismic activity.

04

Results are sent back to app.py, which forwards them to simulation.py.

05

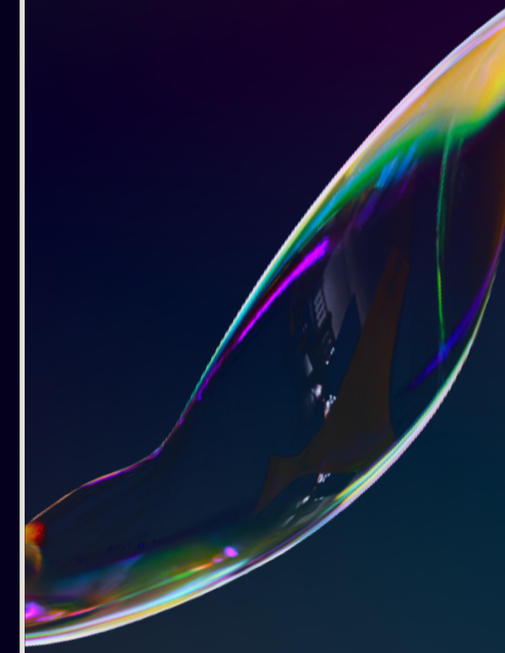
simulation.py visualizes the results on an interactive map.

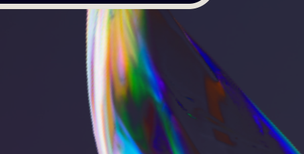
06

If the user selects a defense scenario, the simulation demonstrates asteroid deflection or fragmentation.

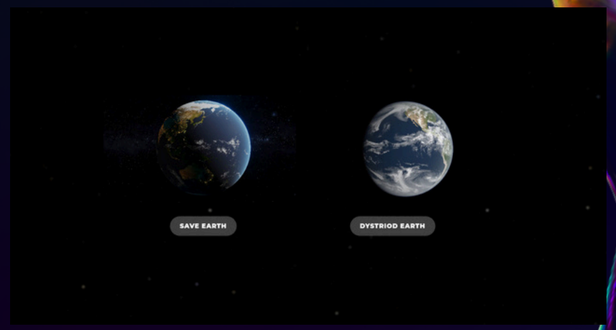
Technologies and Tools

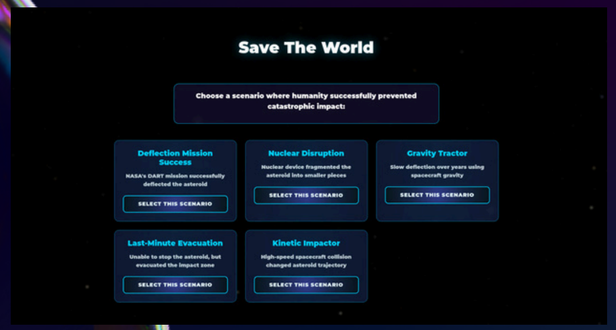
- Programming Languages: Python, TypeScript
- Data Processing & API Handling: requests, pandas, python-dotenv
- Simulation & Mapping: folium, shapely, plotly
- Application Development: Flask, PyWebView, PyQt5, PyQtWebEngine

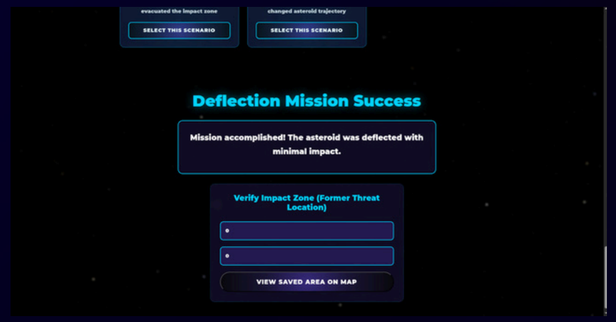


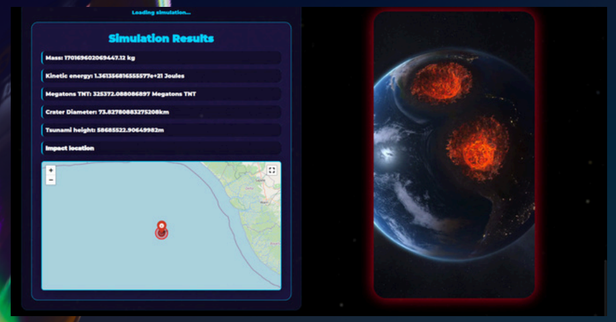


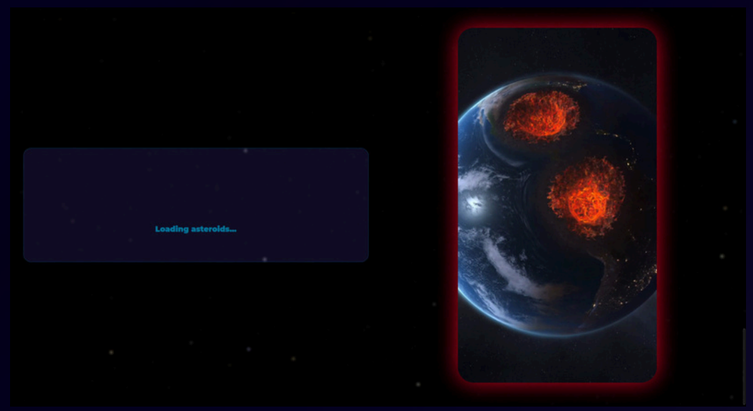
SHOTS FROM OUR WEBSITE

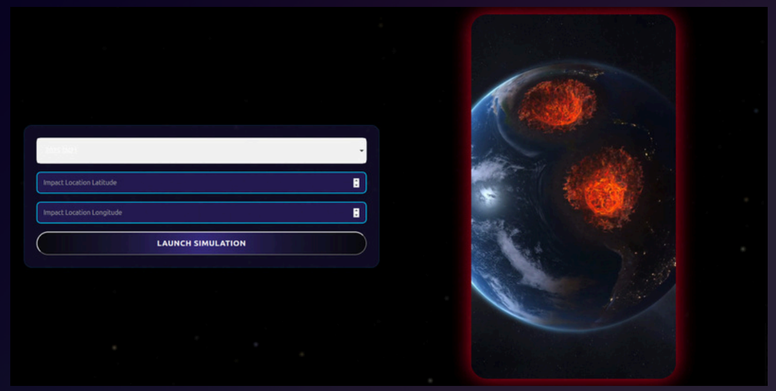


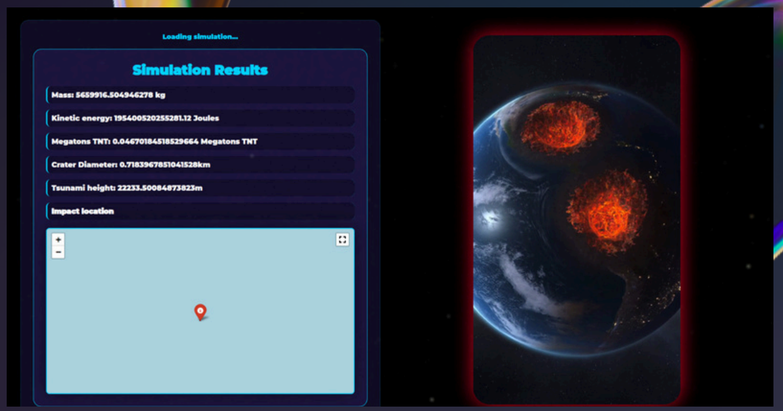












Expected Outcomes

- Accurate simulation of asteroid impacts on Earth.
- Interactive maps showing realistic destruction zones.
- Intuitive interface for exploring different scenarios.
- Educational tool that raises awareness of NEO threats and defense strategies.

Conclusion

Meteor Madness combines physics, programming, and interactive mapping to create a unique simulation tool. By merging real NASA data with engaging visualizations, the project not only demonstrates the dangers of asteroid impacts but also highlights the importance of planetary defense. This makes it a valuable educational and awareness platform.

Future Work

Integration of USGS data (terrain, seismic activity, tsunami zones): This project focuses solely on the physics aspect, without any direct integration of USGS datasets.
3D orbital visualization (space trajectory): While this project includes an interactive Earth map, it lacks a full orbital visualization component (e.g., using Three.js or similar frameworks).