AsteroidX: An Asteroid Exploration Simulation and Visualisation Tool*

Tianyu Zhang, Louise A. Dennis, and Matt Webster

***Abstract*—The use of multi-agent robotics for space exploration creates the need for verification and validation using formal methods and simulation-based testing of such systems. This paper presents an asteroid exploration simulation and visualisation tool that can facilitate agent research in an approximated space setting. The software is able to simulate and visualise multiple spacecrafts navigating a customisable asteroid field environment under the control of either user or agent commands and abiding to space physics constraints. A built-in autopilot system implements Lambert's algorithm to allow autonomous orbital entry/transfer manoeuvres, and collision-free long-range path-finding if the objective is distant. A simulated scenario is described, involving two agents observing multiple asteroids during a debris strike.**

I. Introduction

Recent years have seen a proliferation of research into the use of multi-agent systems for space exploration. Such systems require verification and validation which can be done through a combination of formal methods and simulation-based testing [1, 2]. However, visualisation capabilities are lacking from many projects, hindering researchers' communication and prototyping abilities. Existing space mission visualisation projects either do not support multi-agent run-time simulation (e.g. [3, 4]), asteroid-rich environment (e.g. [5]), or have restricted access to researchers (e.g. [6]). This project aims at developing a tool for space mission simulation and visualisation in the context of an asteroid exploration mission.

II. Design and Implementation

AsteroidX consists of four basic components: a visualisation engine, a space physics engine, an external control program, and an autopilot. These are described in detail below.

A. Visualisation Engine

Visualisation was implemented using the Unity 3D graphics engine [7] (see Figure 1). Customisation of every visualisation detail, such as directional lighting, spacecraft thruster flame, and virtual orbit is allowed. Users can set up the viewpoint to follow the spacecraft or remain fixed in any place. Stabilisation of the virtual camera provides a comfortable and intuitive viewing experience. An information panel provides a wealth of information directly during the simulation.

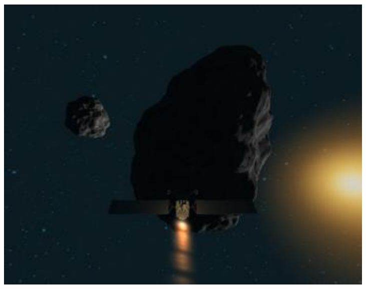


Figure 1. Visual modelling of asteroids, spacecraft, stellar sunlight, and space background.

B. Space Physics Engine

The physical constraints describing space physics, celestial mechanics, spacecraft behaviours and status are scripted in C# to interact with Unity's C/C++ core. The default scaling system is the International System of Units (m/kg/sec) but can be configured to the solar scaling system (km/10^{24} kg/years). Newton's law of universal gravitation is implemented to all objects in the scene unless specified. Detection of spacecraft collision with asteroids will trigger collision reports. Users can set orbital parameters (if in autopilot mode, discussed later) by defining longitude, semi-parameter, eccentricity, and inclination.

C. External Control Program

An external control program controlling spacecraft actions can be provided in any programming language, connecting to the simulator via TCP/IP software sockets. In most cases, external control programs transmit commands to the spacecraft, and the simulator sends back spacecraft sensor data. High-level instructions are made possible by built-in aided control programs such as autopiloting. This ability to support external control software is key to the platform's ability to serve as a research platform for agent program design. The socket interface also allows direct user control of the spacecraft.

D. Autopilot

Autopilot is a built-in aided control program included in the project with the help of software packages NBodyPhysics [8] and Polarith AI [9]. This allows the external control program to focus on high-level mission commands leaving lower-level control to the simulation. It can also assist direct users control when steering the spacecraft manually if their focus is elsewhere. Autopiloting allows automatic navigation of spacecraft to move towards targeted asteroids according to

* Work supported by EPSRC under grant EP/R026092 (FAIR-SPACE) for "Robotics and AI Hubs in Extreme and Hazardous Environments."

T. Zhang is with the Department of Computer Science, University of Liverpool, UK (e-mail: terryyy.zhang@gmail.com).

L.A. Dennis is with the Department of Computer Science, University of Manchester, UK (e-mail: louise.dennis@manchester.ac.uk).

M. Webster is with the School of Computer Science and Mathematics, Liverpool John Moores University, UK (e-mail: m.p.webster@ljmu.ac.uk).

interplanetary orbital entry/transfer manoeuvres such as Lambert Transfer, as shown in Figure 2. Simulation can be speed up to 200 times faster during automated orbital manoeuvres. Autopilot is also capable of collision-free long-range path-finding if the objective is distant, as shown in Figure 3. The sensor range can be limited to simulate the lack of global knowledge. Users/external control programs can disable the autopilot program and gain full control of the spacecraft components such as thrusters and rotation, if desired.

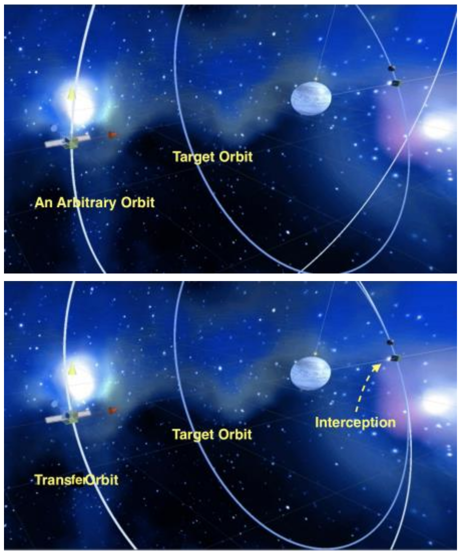


Figure 2. Autopilot orbital manoeuvre: before first burn (top) and after first burn (bottom) during an orbital transfer using Lambert's algorithm.

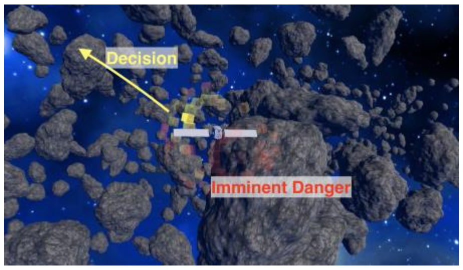


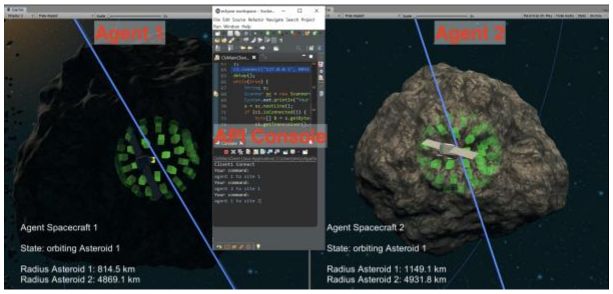
Figure 3. Autopilot path-finding: spacecraft avoiding debris strike using spatial proximity sensor.

E. Setup

The initial environment is defined by the positions, masses, and shapes (based on 3D models) of asteroids within the asteroid belt, as well as the spacecraft. Template environments are provided, and modification of those attributes can be achieved by changing these templates. Unity also provides an intuitive user interface that requires minimal effort to learn to configure those parameters in the initial environment. A detailed guide to setup can be seen on the project webpage[1].

III. Simulation Scenario

The constructed visualisation scenario in AsteroidX echoes the design of Lincoln et al. [10], in which rational agents were able to react to the environment according to sensor data in real-time. In the scenario presented here, two autonomous agents were able to negotiate and distribute responsibility while exploring a partially unknown asteroid field (see Figure 4). They could orbit the same asteroids together for close observation or orbit different asteroids. During all operations, notification of an approaching debris strike overrided all current activities, forcing the spacecraft to engage a collision avoidance mode. Figure 4 shows the screenshot for a simulated scenario where an external control program written in Java was used to instruct two spacecraft agents to orbit different asteroids. The commands sent to the agents in control of the spacecraft are shown in Table 1.



Agent Spacecraft 1
State: orbiting Asteroid 1
Radius Asteroid 1: 814.5 km
Radius Asteroid 2: 4869.1 km

Agent Spacecraft 2
State: orbiting Asteroid 1
Radius Asteroid 1: 1149.1 km
Radius Asteroid 2: 4931.8 km

Figure 4. Simulated scenario where a Java program uses software sockets to instruct two spacecraft agents to orbit different asteroids.

Table 1. Agent commands within the simulated scenario.

Command	Command Type	Agent	Additional Information
Agent 1 to Site 1	Move	1	Site 1
Agent 1 to Site 2	Move	1	Site 2
Agent 2 to Site 1	Move	2	Site 1
Agent 2 to Site 2	Move	2	Site 2
Agent 1 stand by	Abort	1	
Agent 2 stand by	Abort	2	
Agent 1 report status	Report	1	Status
Agent 2 report status	Report	2	Status

IV. Conclusion

AsteroidX is a 3D visualisation tool for simulation of asteroid exploration based on customisable models and visual effects settings. Space physics models were constructed to simulate the real-world behaviours of spacecraft and asteroids. A built-in API allows external programs running at the same time to exchange spacecraft commands and simulation information. The autopilot can take over the navigation tasks and ease the control. Setup is relatively straightforward while maintaining a high degree of customisation and flexibility. AsteroidX can be applied to more simulation-based space exploration research, development, and prototyping, and can be used to validate scenarios for verification and validation. The software could be extended by improving functionality through refinement of the template environment and the development of improved information panels and software sockets.

References

[1] M. Fisher *et al.*, "An Overview of Verification and Validation Challenges for Inspection Robots," *Robotics* , vol. 10, no. 2. 2021.

[2] M. Webster *et al.*, "A corroborative approach to verification and validation of human–robot teams," *Int. J. Rob. Res.*, vol. 39, no. 1, pp. 73–99, Nov. 2019.

[3] S. Cooley *et al.*, "General Mission Analysis Tool (GMAT)." [Online]. Available:

[1] https://github.com/TerryyyZhang/AsteroidX-Asteroid-Exploration-Simulation-and-Visualisation-Tool

https://opensource.gsfc.nasa.gov/projects/GMAT/index.php.
[4] "Mixed Reality Exploration Toolkit (MRET)." [Online]. Available: https://software.nasa.gov/software/GSC-18602-1.
[5] "42: Simulation for spacecraft attitude control system analysis and design." [Online]. Available: https://sourceforge.net/projects/fortytwospacecraftsimulation/.
[6] "Exploration Visualization Environment Version 2.15." [Online]. Available: https://software.nasa.gov/software/LAR-19859-1.
[7] "Unity3D," *v2020.1*. [Online]. Available: https://unity.com.
[8] P. Musgrave, "NBody Physics," *v8.0*, 2014. [Online]. Available: http://nbodyphysics.com/.
[9] M. Kirst and F. Pieper, "Polarith AI," *v1.7.1*, 2010. [Online]. Available: https://polarith.com/ai/.
[10] N. K. Lincoln, S. M. Veres, L. A. Dennis, M. Fisher, and A. Lisitsa, "Autonomous Asteroid Exploration by Rational Agents," *IEEE Comput. Intell. Mag.*, vol. 8, no. 4, pp. 25–38, Nov. 2013.