Orbital Simulator Game - Progress Report

Alice Heranval[†,1], Jami Milliken[⋆,1], Matvey Shakula[‡,1], Garrett Youngblood[§,1]

[1]Georgia Institute of Technology

March 28, 2025

1 Research Topic & Question

This project falls under the astrophysics sub-field, seeking to answer whether an orbital simulator can be created with fast enough computational methods to be interacted with in real time as a video game.

2 Current Setup

The simulator models an n-body astrophysics problem with predetermined positions and masses chosen for each level. The user has control over the initial conditions of a rocket, modeled as a point mass, that navigates through the spheres of influence of the bodies.

Our initial conditions are partially set by the user, who can aim a cannon using the mouse in order to determine the starting angle of the trajectory. Velocities and gravitational constant (currently G = 2000), however, are hardcoded. Our domain is the entirety of the screen; as long as the satellite is on the screen, the simulation continues. Once it has exited the boundaries of the screen, however, the simulation ends and one of the ending conditions is established. For now, our simulation uses RK4 for trajectory calculations.

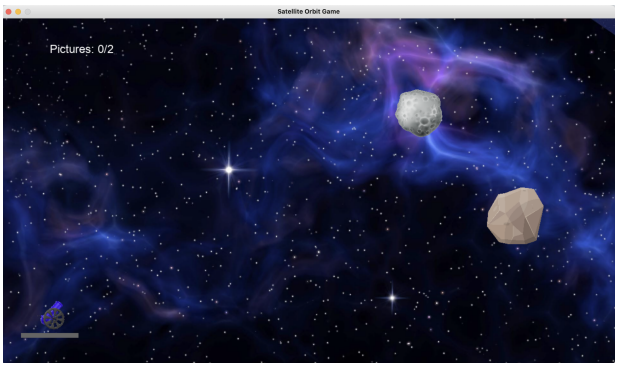
3 Current Status

We have begun putting together our code and testing elements of our GUI, settling on a working version using `Pygame`. The main portion of the game has a loading screen, with buttons that can lead to starting the game itself. In the meantime, we have been working in other folders to add the game elements, and we have one working level at the moment which shows that we have a good idea of how to use the GUI and connect it to the physics.

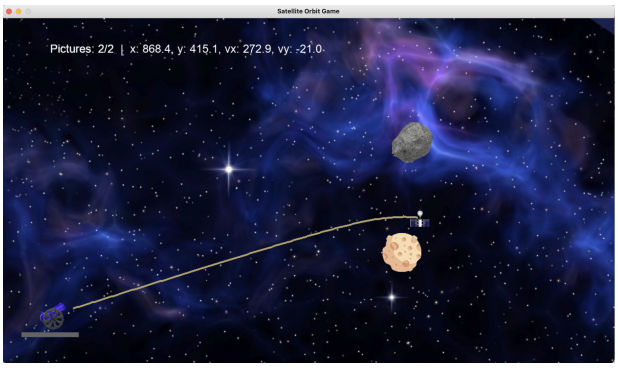
Each time the game is loaded, two asteroids are selected at random, with different masses corresponding to their size on the screen, and the trajectory of the rocket is then calculated. The position of the rocket is updated in the simulation and on the screen as it moves. We use our gravitational constant to calculate the effects of the masses on the path of the satellite. Currently, the satellite must take two "pictures" of the surfaces–one for each asteroid–which are defined in our code as being within a certain distance of the asteroid's surface. The simulation continues to run until the satellite has exited the screen, or until it collides with one of the masses. If the satellite successfully takes two pictures, the game is considered "won"–in the future, more levels would then be accessible, but in this case the user can press any key to restart. If the satellite takes one picture or none, or else collides with an asteroid, then the game is lost and the user must restart. See Fig.1 and Fig.2 for visuals.

After touching up the current simulation, we have some more ideas for levels that we would like to implement. Though we would like to limit these to three to ensure enough time to make the simulations and game look good, we also want to showcase a versatility of methods learned in class by choosing from a list of ideas with diverse needs for numerical methods

[†]aheranval3@gatech.edu, [⋆]jmilliken3@gatech.edu, [‡]mshakula3@gatech.edu, [§]gyoungblood8@gatech.edu

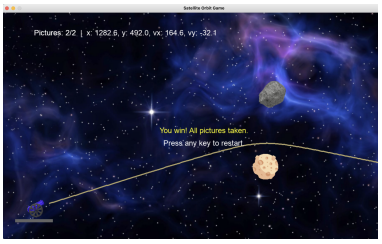


(a) Initial setup, aiming the cannon.

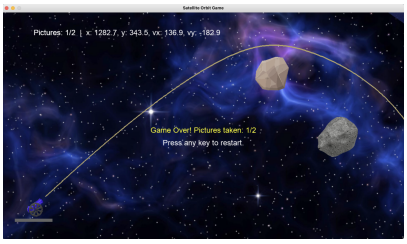


(b) Rocket passing by our objects.

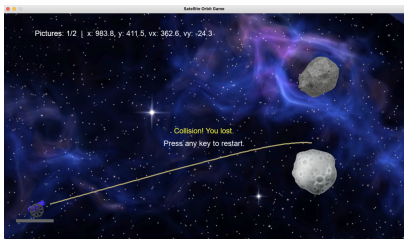
Figure 1: Initial setup, and beginning of the game with rocket in motion.



(a) Winning condition. All pictures taken.



(b) Losing condition: insufficient pictures.



(c) Losing condition: collision with massive bodies.

Figure 2: Ending conditions for the game.

(such as Verlet, or an adaptive method for high accelerations). The following are some ideas we have:

- Add more asteroids
- Effects of going through the atmosphere
  - Could be difficult with the gravitational constant and the size of our screen
- Landing on another spot, such as orbiting twice and landing after
- Adding a black hole
- Letting masses move around, even with respect to one another perhaps
- Working with Lagrange points - for example, a little space telescope that the user must direct to the Lagrange points highlighted on the screen, which are then calculated by our code in advance

4 GitHub

We are using the following GitHub repository: `https://github.com/mshakula/ORB.POND` (ORBital.PONDerer).

†aheranval3@gatech.edu, ⋆jmilliken3@gatech.edu, ‡mshakula3@gatech.edu, §gyoungblood8@gatech.edu

5 Difficulties and Change of Plans

Since we are using RK4, we do not take energy conservation into account. This is fine for the level we have implemented, but could be an interesting next step–for example, with the implementation of a level that requires the user to enter a stable orbit, or to orbit a certain number of times. We also have not taken acceleration into account, besides the change in trajectory due to passing near masses.

We were given the suggestion to add moving bodies, which would resemble more of an n-body simulation. This is something we are considering implementing as a level, but could also work well with the level currently added. This would also be tricky as it would require some more fine-tuning of the gravitational constant to see what a good value would be.

We were concerned that using Python may slow down our game, but so far this has not been an issue with the current environment. The game is however not entirely fluid, so this may still become an issue in the future and is something we should look out for, particularly with optimizing our code.

6 List of Figures

Since this project takes the form of a simulator game, we will likely not include scientific figures in our presentation but rather images of the game itself. As seen above, users playing the game will have access to the position and the velocity in x and y directions (since we are limiting this to two dimensions), which will be displayed on the screen. The path taken by the object launched into space will also be shown.

In our final presentation, we would like to include images similar to those in this report (Fig.1, Fig.2), where we show ending conditions of various levels and the initial setup, and perhaps even show a short demo since the goal of this project is to have a quick-running simulation.

[†]aheranval3@gatech.edu, [⋆]jmilliken3@gatech.edu, [‡]mshakula3@gatech.edu, [§]gyoungblood8@gatech.edu