Orbital Simulator Game

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

[1]Georgia Institute of Technology

March 7, 2025

1 Research Topic & Question

This project will fall under the astrophysics sub-field. The study will answer if an orbital simulator can be created with fast enough computational methods to be able to be interacted with as a video game.

2 Simulation Description

The simulator will model an n-body astrophysics problem with pre-decided positions and masses chosen for each level. The user will have control over the initial conditions of a rocket, modeled as a point mass, that will navigate through the spheres of influence of the bodies. We will be using GitHub to work collaboratively on our code: `https://github.com/mshakula/ORB.POND` (ORBital.PONDerer).

3 Literature Review

Our video game-style project is a highly simplified version of an N-body simulation, focusing on reducing computation time while implementing graphics in a way that this can work as a real-time video game.

N-body simulations model gravitational interactions in systems like galaxies and star clusters. Historically, computational power has been a major limiting factor, since simulating these systems is expensive. However, modern computers have far higher processing power that allows for these to take place. Direct N-body simulations solve Newton's equations for each particle, but this brute-force approach is $O(N^2)$ in complexity. Specific programs have been introduced to account for this, such as tree codes and grid-based methods [1], but in our case we should be able to get by with direct computation since we will be limiting the case to two-dimensional systems with fewer bodies. However, should the computation prove slower than expected, we can easily refer to Dehnen and Read's paper [1] and to Aarseth's paper [2] for help with this issue. Should we encounter issues with singularities in force calculations, the techniques introduced for force softening [2] can also allow us to stabilize the simulation.

Rather than computational complexity from direct force calculations, our main concern is finding a time-integration technique that works for our particular case. Here we benefit again from the simplified nature of our game. Generally the recommended techniques are a leapfrog integrator [1, 2] as we learned about in class, but because we are realistically only performing calculations over the course of a few seconds (in real time), we are less concerned with a loss of energy, and will be instead using the classic RK4 method (see section 5) [3].

4 Required Input Physics

Physically, we will consider Newtonian gravitational force and acceleration, governed by:

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

$$\mathbf{a} = -G\frac{M}{r^3}\mathbf{r} \tag{1}$$

Where G is the gravitational constant, M is the mass of a planetary body, $\mathbf{r}$ is the position vector from the planetary body to the moving rocket, and $r = \|\mathbf{r}\|$ is the distance between the two. In our simulation, the planetary bodies are treated as fixed (or predetermined) massive points that exert gravitational forces on a single moving rocket. The rocket's motion is influenced by the cumulative effect of all the gravitational forces from the planets. For each planet i, the gravitational force acting on the rocket is given by

$$\mathbf{F}_i = -G\frac{M_i m_r}{r_i^3}(\mathbf{r}_r - \mathbf{r}_i) \tag{2}$$

Where M_i is the mass of the ith planet, m_r is the mass of the rocket, $\mathbf{r}_r$ and $\mathbf{r}_i$ are the position vectors of the rocket and the ith planet, respectively, and $r_i = \|\mathbf{r}_r - \mathbf{r}_i\|$. Since the rocket's mass cancels out when calculating acceleration, its net acceleration becomes

$$\mathbf{a}_r = -G\sum_i \frac{M_i}{r_i^3}(\mathbf{r}_r - \mathbf{r}_i) \tag{3}$$

5 Computational Methods

The goal of our simulation is for it to be both accurate and interactive. The latter part implies that modeling must occur fast enough that a user perceives everything as happening in real-time and without delay. For this reason, our simulation's numerical methods must be accurate, but more importantly they must also be fast.

General Case

For most of the problems we have discussed simulating, the system can be adequately evolved through time using a 4th-Order Runge-Kutta (RK4) method [3]. RK4 is a numerical technique for solving ordinary differential equations (ODEs) of the form $\frac{dx}{dt} = f(x,t)$. Provided an initial value $x(t_0) = x_0$, the RK4 method advances the solution through time in steps of size h using a weighted average of slopes. Specifically:

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4}$$

where the k values are the following step-slope products:

$$\begin{aligned}
k_1 &= h \cdot f(x_n, t_n) \\
k_2 &= h \cdot f\left(x_n + \frac{1}{2}k_1, t_n + \frac{1}{2}h\right) \\
k_3 &= h \cdot f\left(x_n + \frac{1}{2}k_2, t_n + \frac{1}{2}h\right) \\
k_4 &= h \cdot f(x_n + k_3, t_n + h)
\end{aligned}$$

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

Highly Variable Force Fields

For cases where the rocket is exposed to quickly varying forces, and thus will have a rapidly varying velocity, a more refined method than RK4 is needed in order to avoid overshooting the actual solution. This will be accomplished with an adaptive RK4 (ARK4) method using the varying step size method [3].

The general procedure behind the varying step size method is as follows:

1. Perform two steps of size h and one step of size $2h$, starting at the same point.
2. From these calculations, compute ρ, the ratio between the ideal and actual step sizes, from the two estimates x_1 and x_2.
3. If $\rho > 1$, the actual accuracy is better than the target one. Keep the x_1 estimate (2 steps), and to avoid computational waste in the next step, increase h by the factor ρ.
4. If $\rho < 1$, the actual accuracy is worse than the target one. The calculation must be repeated with a smaller time step, decreased by the factor ρ. Note, usually the step size is not allowed to change by more than a factor of two.

This method will allow us to accurately evolve our system in time through rapid variations in gravitational fields, and actually improve the accuracy of the RK4 method via local extrapolation [3].

Stability Analysis

For cases where the simulation must cover a longer time domain, and thus undergo more iterations, the error accumulated by the RK4 method will cause it to diverge from the actual solution. Therefore, a symplectic technique is needed to ensure that the solution continues to conserve the laws of physics over a long period of time (many iterations). We will use the Verlet method [3].

The Verlet method is a special case of the Leapfrog method that can be used to solve a coupled systems of ODEs of the form: $\frac{dx}{dt} = v, \quad \frac{dv}{dt} = f(x,t)$. Provided the initial values $x(t_0) = x_0$ and $v(t_0) = v_0$, the Verlet method advances the solution through time in steps of size h using the following algorithm:

$$v\left(t + \frac{h}{2}\right) = v(t) + \frac{h}{2} \cdot f(x(t), t)$$

$$x(t+h) = x(t) + h \cdot v\left(t + \frac{h}{2}\right)$$

$$v(t+h) = v\left(t + \frac{h}{2}\right) + \frac{h}{2} \cdot f((x+h), t+h)$$

Past the first iteration,

$$v\left(t + \frac{3h}{2}\right) = v\left(t + \frac{h}{2}\right) + h \cdot f((x+h), t+h)$$

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

6 Simulation Setup

The extent of the simulator's space and time domains depends on the particular problem simulated.

Generally, the extent of the spacial domain will be simply the defined space of the simulation. Any objects which exist outside of the defined region of the simulation will be ignored, and any objects which exit the region will no longer be considered. Objects which enter the region of simulation from "off-screen" will be considered for as long as they remain within its boundaries.

The extent of the simulation's time domain will depend on the simulation's win and loss conditions. Whenever all the criteria are met for at least one of the defined win conditions, that instant will be the extent of the time domain, and the simulation will end. Likewise, whenever any of the defined lose conditions are met, that too will be the extent of the time domain, and the simulation will end.

7 Quantities to Inspect

The specific quantities inspected will vary depending on the main objective a simulation aims to present. For example, a simulation focusing on a rocket's flight path and reaching a certain destination will likely need to inspect the rocket's position, speed, and perhaps time of flight. We plan to create multiple simulations, each with their own main objective, and cumulatively plan to inspect the following quantities for the rocket:

- **Position** - The rocket's position will tell us where it is located at any given moment. This quantity will be studied as a function of time.
- **Velocity** - The rocket's velocity will tell us both how fast it is moving and where it is heading at any given moment. This quantity will be studied as a function of time.
- **Net Force** - The net force acting on the rocket will tell us how we should expect its position to change. Additionally, it can be used to identify equilibrium conditions. This quantity will be studied as a function of space, and perhaps also time.
- **Mechanical Energy** - The rocket's mechanical energy can be used to identify equilibrium conditions, as well as to check the validity of our solutions via conservation of energy. This quantity will be studied as a function of time, and perhaps also space.

Please note as the specifics of the simulations become more finalized, this list of quantities we plan to inspect will likely change and may grow or be reduced.

References

[1] Walter Dehnen and Justin I. Read. "N-body simulations of gravitational dynamics". In: *The European Physical Journal Plus* 126 (2011), p. 55.

[2] Sverre J. Aarseth. *Direct Methods for N-body Simulations*. Academic Press, 1985. ISBN: 0-12-123420-7.

[3] Mark Newman. *Computational Physics*. 1st ed. San Francisco, CA: CreateSpace Independent Publishing Platform, 2012. Chap. 8.1.3.

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