Parallel Orbital Simulation Environment (POSE)

Dylan R. Wagner

February 24, 2019

1 Context

This document will describe the aspects of the simulation software including: operation, orbital model including models of perturbing forces, concurrent parallel design, input and output, model of orbital collisions and simulation of spacecraft logic. Additional aspects related to stages of simulation development will also be provided.

2 Summary of Operation

1) The software is to take in parameters relating to the simulation environment. Read in the initial input file defining the state of objects in orbit at time 0. These objects will be added alongside entities pertaining to the orbital environment such as the Earth, Moon, Sun and other Smart objects.

2) The simulation will then calculate acceleration vectors for each object in the environment then update the corresponding velocity vector and position. For Smart objects, custom logic evaluates the environment within the simulation then updates internal state. When the simulation has completed calculations for the current time interval, collision detection and modeling is evaluated. Any new bodies generated by the collision detection and modeling algorithm will be added to the current pool of bodies in the simulation. At this point, the current state of the bodies in the simulation will be logged to a file or sent to a remote location.

3) From this point on, the simulation repeats until a stop condition is reached. This stop condition is tied to the total runtime or triggered by defined event(s) within the simulation. At the end of simulation statics will be logged to a flat file.

3 Orbital Simulation Method

The simulation software will use Cowells method. Cowells method involves adding together acceleration vectors acting on bodies in orbit. This summed acceleration vector can be separated in X, Y, Z components then integrated to find velocity with velocity being integrated to find spacial displacement.

4 Models of Perturbing Forces

- Large Body Gravity
- Earth Non-spherical Gravity
- Solar Radiation
- Drag
- Magnetic Fields
- Propulsion

5 GPU Acceleration

Most calculations on bodies within the simulation will be accelerated by one or many graphic processing units. Calculations pertaining to the orbital perturbations will be at the heart of the shader program. It is important to note that not all calculations will be performed on the GPU but the bulk will. By using GPU acceleration, massive amount of orbital debris can be included into the simulation. Using GPU acceleration the simulation can scale by using multiple GPUs.

6 Input and Output

The input and output to the simulation will consist of only text based files or streams. Below definitions for all input and output at all stages of the simulation are provided.

6.1 Input at Start up

The simulation at startup needs some information pertaining the celestial environment. This information consists of the bodies which will be simulated. The bodies are ether debris or spacecraft; each needing additional attributes which define the object. Below attributes for both debris and spacecraft are listed:

Debris

- X, Y, Z locations in the simulation space
- X, Y, Z initial velocity values
- Shape (Cube or Spheroid)
- Dimensions W, L, H
- Material (Eg: Aluminum or Aluminium oxide)

"Dumb" Spacecraft

- X, Y, Z locations in the simulation space
- X, Y, Z initial velocity values
- Spacecraft Name
- Opt01: Shape (Cube or Spheroid)
- Opt01: Dimensions W, L, H
- Opt02: Spacecraft model file

The objects read in at start up will be listed sequentially in a flat file using Javascript Object Notation (JSON).

6.2 Input at Simulation Run

6.3 Output at Simulation Stop Condition

6.4 Output at Simulation Run

7 Model of Orbital Collisions

8 Simulation of Spacecraft Logic

9 Stages of Development

10 Domain