TacticalSpace Conceptual Design

patternjuggler galacticnorth@gmail.com

2005.04.24

Contents

1 Introduction

TacticalSpace is a 6DOF simulation of hard-science-fiction space combat with relativistic physics and modeling of speed of light delays.

2 Relativistic Physics

Objects nearing the speed of light will experience standard effects such as observation of relatively moving objects have slower clocks and show lorentz contraction.

Until a proper explanation of stellar aberration is developed, the absence of aberration will be explained as being eliminated by shipboard computers so that battlespace management is easier.

3 Light Travel Delays

The player will have to assume a particular inertial frame in which to manage their fleet (FIXME move somewhere else). Every ship has its own inertial frame and will see the battle space differently.

Since the volume of battlespace is large relative to the speed of light, any object will see distant objects in the states they were seconds or minutes in the past when the light had left the distant object. This will limit their ability to track and engage distant objects effectively since the object may have changed its state drastically in later moments. One of the primary goals of the TacticalSpace project is to develop weapons and tactics to make this playable.

4 Guidance Navigation and Control

Progressively sophisticated gnc laws will be used for controllable objects in the battlespace.

PID

$$F_x = P\Delta x + I \int \Delta x dt + Dd\Delta x/dt \tag{1}$$

5 Gravity

Massive objects will exert forces on each other and all objects, which may drastically complicate play.

Non-sphereically shaped objects may have non-uniform gravity fields and complicate matters even further, although multiple point objects dispersed in space is effectively the same thing.

Gravitation Constant

$$G = 6.67259 * 10^{-11} m^3/(kgs^2) or Nm^2/kg^2 \tag{2}$$

$$Force = G\frac{m_1 m_2}{r^2}; \tag{3}$$

6 Physics

Aside from the Einsteinian physics already discussed, normal collision and response modeling will be done for object interaction. Damage to objects will typically be the result of collisions with other objects.

6.1 Player interaction

The game should be capable of exercising all functionality automatically, that is all missions should be playable by the AI, although it is expected that the user will have much more tactical depth than the AI.

The player will be tasked with setting waypoints for fleets or individual ships and setting targets or target priorities. The per-ship/fleet player workload should decrease with time as gnc increases in sophistication. As this workload decreases, complexity of mission goals and gravity wells and fleets size should go up to keep player interest (FIXME this is process oriented, should stick to ideal final product?).

7 Architecture

God's eye view/absolute inertial frame

It is possible to generate arbitrary points-of-view for any inertial frame in a space given knowledge derived from a single inertial frame. The simulation will run at a specified rate on a single inertial frame. The simulation will hold information about simultaneous state vectors of distant objects even

though the speed of light forbids such concurrent knowledge, and even though alternate frames do not agree on what events are simultaneous.

It is expected that the greater the velocity vector between the Simulation Frame and a particular object's frame, the lower the fidelity of the simulation of that object's frame. For this reason an arbitrary speed limit slightly less than c may be needed to minimize problems.

7.1 Light Travel Delays

An algorithm to simulate the speed of light is one of the most critical and interesting parts TacticalSpace, and for the most part represents the key difference between this project and other space battle games.

In reality, space is full of photons travel at the speed of light and viewers will capture those photons and see images of objects as they were when the photons were emitted or bounced off of them. It is assumed from the start that a photon level simulation is not realizable.

The problem is simplified for the initial space configuration- small discrete objects in a 3D volume rather than large objects where light reflected from different parts of the object would arrive at a viewer with different ages.

For this problem, the small discrete objects need to maintain state history for states as old as light would take to travel across the battle space. Additional algorithmic features to compress the amount of state history that needs to be maintained are possible but less critical.

7.1.1 The Algorithm

The purpose of the algorithm is to find the point in history of a distant object where the distance from the object to the viewpoint's current position converted to time (divide by c) is equal to how old that point in history is. That is, how long ago did light leave the object so it just now arrives at the viewer?

So there could be many ways the algorithm could be implemented in order to search through the history for the right index.

Currently, on initialization the algorithm operates very simply by starting with the earliest point in the distant object's history and computes the distance to the viewer.

If the distance/c is greater than the age of the historical point, then that means there is no point in the history that is appropriate for showing so the object is considered invisible to the viewer (other options would include extrapolating backwards in time or just using the earliest history point).

If the distance/c is smaller than the age of the historical point, this means that this light has already traveled by the viewer, so iterate to the next younger point in history until the distance/c is greater or exactly equal to the age. If not exactly equal, the younger or older point in history can be chosen or perhaps interpolation of the states of the two can be done. The algorithm stores the history point.

On the next update, this process must be repeated. However, since it is known that the viewer will never see an earlier point in a distant object's history after seeing a later one (i.e. see an object move backwards in time), the algorithm need not start from the beginning, it can take the stored history point and move forward. The amount of iteration then needed is proportional to the relative velocities of the objects. Relatively stationary object and viewer need only iterate once, while converging object and viewer will have to iterate many times- this is analogous to the doppler effect, where an object converging on the viewer appears to have a fast-running clock.

There is probably some room for faster search algorithms especially as the numbers of objects goes up, but for the present this is adequate.

7.1.2 Wrinkles

There are many other effects to eventually consider but are for the most part unimportant.

Lighting history. Each object as shown in the past needs to be lit with light that was shining on it in the past. For the most part the scene filling light is a central non-moving star whose light remains constant at all times, if one ignores shadows. All other sources of light are probably localized and should not interact with distant objects anyhow, though this should be examined later on.

8 Third Party Tools

8.1 Libraries

OSG Open SceneGraph

OSG provides:

- a scenegraph to minimize cpu overhead.
- model loaders for a variety of 3d formats.
- lights management (?)

OSG suffers from a lack of traditional documentation, but there is a very active newsgroup where the primary creator of OSG will respond to legitimate requests. There is also a wiki, which has its uses but is much less effective than a document that has been created with the reader reading from start to finish in mind. Hopefully there will be a proper manual or print book available in the future.

OSG also suffers from a 'do everything' kind of approach that seems to default to throwing more stuff into the library rather than judiciously deciding to concentrate on core functionality and leave certain things up to users. This makes it very difficult to feel like one has a handle on any more than a small fraction of the library.

ODE Open Dynamics Engine

- Collision detection
- Collision response

  Much functionality possible with ODE is not necessary for the baseline TacticalSpace implementation, there is very little need for joints of any kind, modeling of friction is not necessary, and many other complicated tasks are not necessary for a simple mostly empty space full of small rigid objects.

Responses for all but the lowest speed collision will have to be highly custom to maintain stability. Cases:

- Asymmetric large-body to small-body collision
- High energy: both objects are destroyed, create debris objects with same energy of input system plus energy derived from explosive reactions.
- Medium energy: Large object takes 'damage' from small but is not physically affected, small object is destroyed.
- Low energy: Large object unaffected, small object bounces off.
- Symmetric collisions
- High energy: both objects destroyed creating debris objects.
- Low energy: both object bounce away, may take damage.

A very good online manual exists for ODE, as well as an active newsgroup. The ODE concept is very well defined so that suggestions and patches are rejected on the grounds of doing work that is too specialized, which keeps the library small and clean.

9 Implementation Process

The project will be implemented in a breadth-first fashion. Simplistic and minimal graphics, control, and user-interface will be in place in a playable form as early as possible. Usage of third party tools will accelerate insertion of high-quality 3d models and graphical effects once the project is mature.

9.1 First Phase

- No weapons, only kamikaze attacks (not much of a game that way unless multiple ships can be taken out at a time).
- No gravity.
- 3DOF controls only, no attitude adjustment. Collisions may cause rotations but the controls will stay in world space and ignore those rotations. All thrusters in all directions will have same maximum thrust.
- Simple PID (Proportional Integral Derivative) GNC.
- Perfect knowledge of positions of all other ships, even speed of light delays will be ignored. Actually the engine may be more suited to skipping that step and using the known positions as input to gnc laws.
- Default ODE collision detection and response, some dampening to improve stability.

A Acronyms and Definitions

PID	Proportional Integral Derivative
GNC	Guidance Navigation and Control
DOF	Degrees of Freedom
OSG	Open Scene Graph
ODE	Open Dynamics Engine