Nebulous: Networked Space Flight Engine

Dylan Rush, Jon Schiavo, Hector Muñoz-Avila

Computer Science & Engineering, Lehigh University, Bethlehem, PA 18015

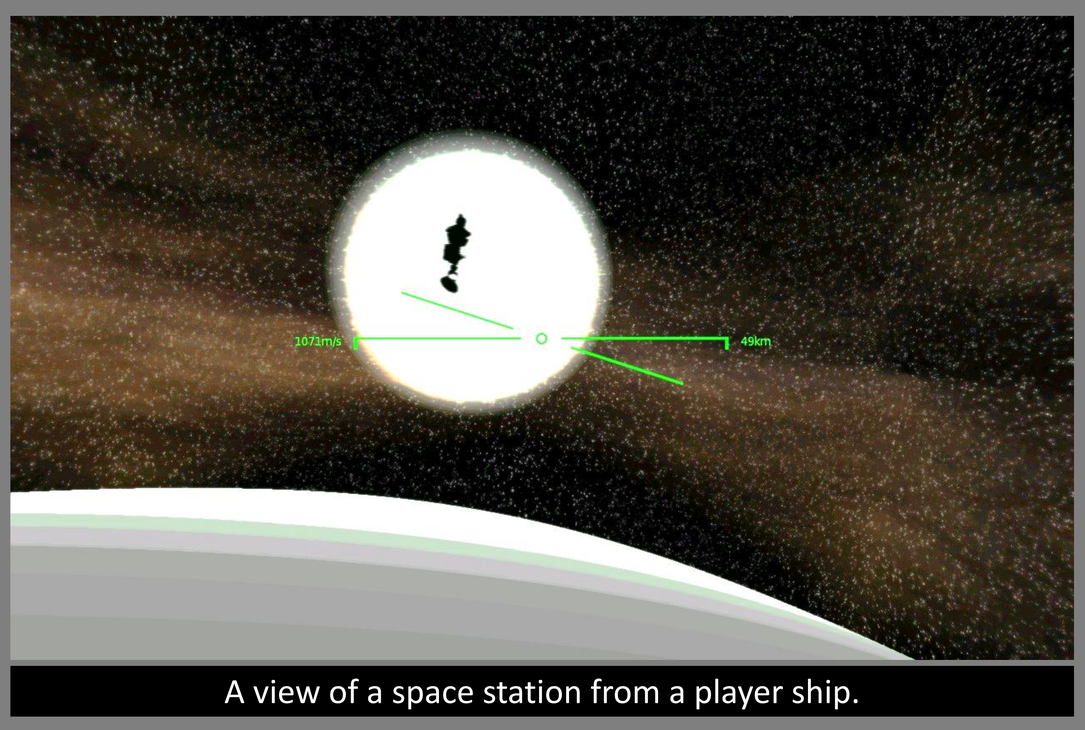
Abstract

Nebulous is a real-time game engine which allows a player to move around a procedurally generated universe which scales seamlessly from light-years to centimeters. Expanding upon the existing code, we created a networked version of the engine which supports a shared simulation for multiple players.

Divergence From Common Practice

Most modern multiplayer games limit dynamic objects to player avatars. Few, if any, objects in the world move on their own.

Our solution aimed to be able to synchronize any number of dynamic objects with realistic physics, anywhere in the system.



A view of a space station from a player ship.

Synchronization

The physics engine we used is highly deterministic for independent dynamic objects. However, some objects (such as the players' ships), do not act deterministically.

To ensure all clients are running an accurate simulation the server periodically re-synchronizes the clients to bring them up the speed with the world state.

Dead Reckoning (aka Client Prediction)

In order for the client to maintain interactive framerates, it cannot always wait for the server's updates.

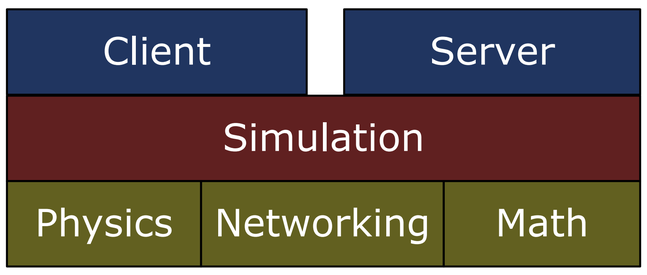
Instead, smooth gameplay is accomplished through Dead Reckoning: the client presents the player with its best guess of what the world state should look like, until the server re-synchronizes it. By reading the player input packets which are relayed through the server, the client can keep a semi-accurate model of the universe which will suffice until an update.

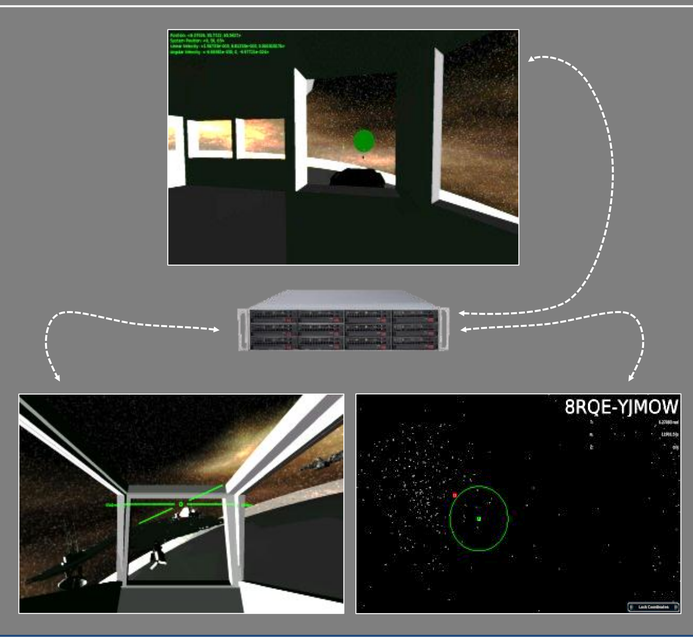
The Client

- Local simulation of physics
- Resynchronizes with server periodically
- Sends player input to server
- Renders the world

Layered API

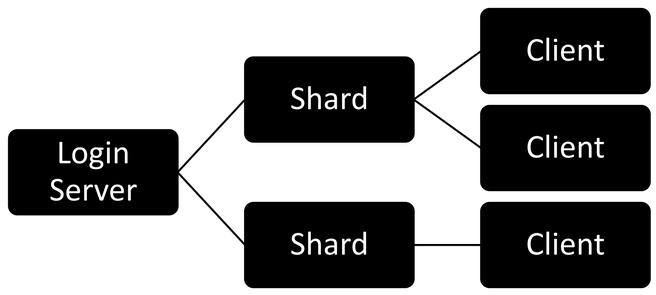
Client and server code share libraries for consistency, reliability and good design.





The Server

- Authority on the world state.
- Divided into Login Server and Shards (Game Servers).
- Login Server maintains player characters and accounts. Players choose character on connection
- Each Shard represents a star system with players in it. Systems without players are not simulated to save resources.
  - Independent threads.
  - Actions on one Shard do not alter the state of others.



Acknowledgements

The following libraries were used:
Open Dynamics Engine (ODE) – Created by Russell Smith [www.ode.org]
CEGUI – Created by Paul Turner [www.cegui.co.uk]
DirectX 9.0 – Created by Microsoft Corporation

LEHIGH UNIVERSITY