CGS Brief – C++ Networked Game

Modular Networking Engine

Complete the four C++ **Networking** tutorials up to **Dead Reckoning**, and move the Client project code into a dll project. The Server can remain an executable project.

Dead Reckoning

Take a look at different interpolation equation from the server side position to the current client side position.

Proof-of-Concept Application

Write a small game that augments the behaviour of *GameObject*, so that your game-side objects are automatically transmitted across the network.

This could be done with inheritance.

You'll have to update the base Client class in the dll project to work with collections of *GameObject* pointers rather than *GameObjects* to allow for polymorphism, and have a virtual CreateGameObject() class that the derived *GameClient* class can override.

The *GameClient* class in the test executable project is derived from *Client*, and you can have a *GameObject* derived class that adds extra functionality, such as drawing capabilities using sprites, or 3D models using your ***Computer Graphics*** engine.

If you have extra data that needs to be transmitted across the network for *GameObjects*, (eg health) you can make the Read() and Write() functions virtual and have the clients read and write this extra data.

The Server may not need to be aware of this, as it bounces messages to the clients agnostically, and could just store a list of base *GameObjects* which it manipulates for interpolation purposes. If the server needs to store extra data such as health, so that it can make the call to remove a dead object, you'll need to figure out how to do this, perhaps using a **Factory Pattern** for making *GameObjects* and putting a type ID into the base class.