NPC AI Requirements and Design

Document version: 0.1
Status: Draft
Last Modification:2003.06.10

Created by: Keith Fulton

Other authors:

Table of Contents

Document History

<In this section you can add information about the last revision of the doc.>

Date	Ver.	Author/Comments
2003/6/10	0.1	Keith Fulton

- Initial description of how required features of NPC AI will be implemented and how other implementations and capabilities will be supported.

Introduction

The purpose of this document is to list all the requirements for two fundamental pieces of the Planeshift Engine architecture. First is the explanation and design of the "superclient" concept, and how any/all superclients will communicate with the server. Second is how the first superclient will embody this design to fulfill the requirements for our initial npcs.

"NPCs" in this document means any Non Player Character, or any computer-driven character, whether it is a merchant, a dragon, an animated sword, or a bird flying overhead providing atmosphere.

NPC AI Superclient Requirements

Since we want to support as many players as possible on a single server instance, we must use as much of the CPU as possible for the actual game server. This means it is crucial to offload other CPU intensive functions such as Artificial Intelligence to auxiliary servers and processes rather than trying to merge them into the core server. This design decision provides for several key benefits:

- Making the NPC Management area a separate process from the server, connected through a sockets interface, means the NPC AI can be done on a separate machine (and CPU) from the core game server.

- By moving the NPC Manager to a separate machine, that machine will have the capacity to control more than one NPC at a time—possibly hundreds. This requires a new network protocol designed for highly optimized updates to multiple NPCs at once without the overhead of per message headers and packet headers.

- Having it be connected via the network means it should be trivial to support multiple connections to multiple NPC clients, which means more machines can be added if it is desired to raise the number of NPCs or sophistication of them, with minimal additional CPU load on the core server.

- Allowing for multiple superclient instances connected through the network means that each superclient can use different methodologies, algorithms, tools and even languages to implement their AI functions. Since AI is such a hotbed area of research, we should anticipate much activity in this area and even encourage competitions between superclient developers, etc. As long as every superclient is compatible with the networking protocol, the AI engine can be whatever we want or need.

SuperClient Protocol

Because we anticipate that the typical superclient will oversee dozens or even hundreds of NPCs, networking setups and message structures uniquely optimized for multiple updates at once will be employed. As long as a superclient follows this protocol, it should be able to handle its NPCs in the game world successfully.

Connecting

When the superclient is started, the server must already be in a "ready" state. It sends a NPCAUTHENTICATE message to the server. The server validates the superclient connection info, and if valid, sends back two messages.

First, it sends back a message with a list of the world maps to load. The superclient should load each map specified, then process the second message.

The second message is the CEL Persistence message with all entities (players and in-game objects) currently active in the world. For most clients, only a subset of the entities within a certain range are sent to the new client. For a SuperClient, ALL entities are sent. When this message is received, the superclient should do whatever processing it deems necessary to instantiate NPC AI info in the superclient.

The third message is a list which maps NPCs currently "alive" and active between player id's and entity id's. Entity id's are assigned on the fly as the world progresses, while Player id's are permanent records for each npc in the database. Each player id is unique to that npc and permanent. Superclients should a) use the player id as their key to their setup information, and b) map from that setup information to the CEL entity list received in msg #2 after it is received.

NPC Updates

The Superclient is expected to send NPCCOMMANDLIST messages to the server on a periodic basis. These messages each contain a variable number of commands and are unraveled and executed on the server in the order they were put into the list. Overhead on this command list is 1 byte per command, instead of 14 bytes per command + 8 bytes per packet on normal client/server messaging.

The Superclient Network Manager has a series of member functions called QueueNPCCommand (with multiple overrides, depending on what command is being queued) which handles the actual message packing for this message, due to its complexity.

World Updates

The server also does not send normal world update information to the SuperClient. Since Superclients are not rendering scenes, it is less important for them to be exact with regard to DR (Dead Reckoning) positions and so forth.

Instead, the server sends updated position and velocity information for all non-stationary entities every 3 seconds. This is in the NPCWORLDUPDATE message, which packs all this information as tightly as possible. Superclients may use the velocity information if they wish, but I expect most

will employ only the position information. Since a player can only move less than 6-10 meters in 3 seconds, this should be accurate enough for realistic NPC behavior, which is all we really are going for.

Additionally, the server will send all new entities and all disconnecting entities to the superclient using the normal CEL messaging, which is already fairly byte packed. Since these messages are relatively infrequent, it is of little benefit to try to pack them or optimize them further.

Perceptions

In addition to the positions and existence of in-world entities, other game events need to be notified to the superclient, such as if one of its NPCs is attacked or if someone dies. These events, from the superclient's perspective are called "perceptions". These will be sent every 1/4th of a second to the superclient in batches. This should be accurate enough to allow NPCs to react in a timely fashion while slow enough to benefit from batching.

In addition to server-generated perceptions, which are sent unbidden to the superclient, the superclient can also employ "Requested Perceptions". This involves an actual query from the superclient for more information about a particular entity or player, and will result in an answer back asap from the server. Right now, none of these are implemented but they are anticipated.

Planeshift Reference Superclient

This section will document the superclient I am currently working on, but I haven’t written this yet.

Conclusion

Hopefully this document will help explain some large but attainable design decisions made around the NPC AI architecture and design in the Planeshift Engine. It is all subject to change, as usual. :-)

Good luck.