Spatial Interest Management in Networked Games

Ali Anvari and Miguel Rio

Networks/Services Research Laboratory, University College London

Abstract: This paper addresses a scalability and telecommunication challenge posed by multiplayer internet gaming. It also aims to provide an overview of a solution to this challenge, termed Area of Interest, and propose how Area of Interest can be improved using Spatial Database techniques. This proposal can be regarded as a form of cross-layer optimization between the Application Layer and the Session Layer.

1. Problem and Motivation

The motivation for studying gaming is that games are an increasingly important commercial application of telecommunications networks which pose acute scalability challenges. The type of game considered here is the immersive virtual reality type game that leads to pairwise interactions between all objects in a game, leading to problems with bandwidth and latency.

Solving these challenges both forwards the networking state of the art, and allows the production of better games, which itself has commercial value. It is easy to gain an indication of how much value consulting market research (DFC Intelligence estimates online gaming to be an US$18 billion revenue business in 2011, up from US$15.6 billion in 2010[1].)

The scalability bottleneck is the number of players a single game can cope with. This challenge is illustrated by the fact that a game server may be able to host many thousands of players[2], but actually managing to fit two thousand players in the same *game* is unheard of.

2. Related work

The military have been running wargames with 10000 avatars since the nineties[3]. They achieved this by using distributed simulation, partitioning the game across many servers. However, they used special hardware, and hardware multicast and their own dedicated networks to synchronize the servers. Such facilities will not be available to a games company using cloud computing and the public internet, and so distributed simulation is problematic.

Academic projects in distributed virtual environments have been developed that use peer-to-peer game logic, and application layer multicast (with perhaps distributed hash tables)[4].

But peer-to-peer techniques are often unsuitable for the world of commercial gaming, because of the imperative to prevent dishonest players from cheating. It may be possible to lock down a closed platform such as a games console or iPhone (though even these may be 'jailbroken'), but it is utterly impossible for a games developer to control the configuration of a personal computer, creating all sorts of opportunities for cheating.

For example, a player could hack the device drivers on his graphics card so that he could see walls and so detect where others players are hiding. This attack affects neither the game software itself nor the data and so is very hard to detect. Though some academics have attempted mitigating the problem, games developers have decided that it is better not to have the problem in the first place.

Games developers regard securing the client as infeasible, making statements such as "it is a fundamentally unwinnable technical battle to make a completely cheat proof game of this type" [5]. This is not a problem for small, private "death match" type games between small groups of friends, a scenario where cheating is more tolerable.(Then again small games don't have scaling issues anyway).

For large, public gaming services hosting with many players the consensus is "Never put *anything* on the client. The client is in the hands of the enemy. Never ever ever forget this."[6]

Hence, in the general case, games developers prefer not to send the client any data beyond what it needs to display to its own user and only its own user, and will always use a central server as a trusted reference point for the true state of the game. This research will therefore focus on exclusively central server-based architecture.

3. Game Scalability

A virtual reality is based on a computer animated 3D world backed by a physics model, which at a minimum resolves colliding objects. (For example, game characters are not allowed to move through walls). The player's camera position sees through the eyes of an avatar – a game entity representing the player to the rest of the game world (a superhero soldier or perhaps a cute cartoon animal). One plays by controlling the avatar to run, jump, grab at things, shoot at things and so forth.

There are other avatars controlled by artificial intelligence, termed Non Player Characters, and constant data about inanimate objects, termed Non Destructible Scenery. There are tricks for avoiding sending data about these game objects over the network. For example, all NPC actions can be perfectly extrapolated from a single synchronisation seed[7] and all Scenery data can be pre-computed[8] as a map or level file distributed with the client and fetched from disk as needed.

But neither trick will work for avoiding Player Character data, and so the scalability of a game is determined by the number of players and by the number of interactions of their avatars with the game state. This means that a game's complexity is potentially combinatorial in the number of game objects and at least quadratic in the number of players.

Therefore, a game could potentially send millions of updates per second. However, the bandwidth feasibly available to a game may be significantly less than 1 megabit per second per client, and even less than 100 kilobits per second[9][10]. And even when the player has a high bandwidth internet link from his client to the game, contention and congestion problems may throttle his link.

Games also require real-time responsiveness and so have tight (sub-second) requirements for both frequency of state updates (termed frame rate) and total delay (termed lag). This includes the time to process the game simulation and to transmit the data over the internet (and the time taken by any "time saving" optimizers to calculate!).

Worst still, bandwidth problems exacerbate latency problems, as packets that can't be sent are queued or even dropped entirely. In practise, round trip times of half a second have been observed within congested datacentres[11].

If a game client's lag is too high overall, that client may be ejected from a game session. If a game server gets bogged down calculating thousands or millions of object interactions, the game ceases to be real-time and hangs for all players, which is even worse.

4. Interest Management and Area of Interest

The game state is updated by the physics model, based on the interactions between avatars and other game objects, and these interactions diminish with distance. For example, an avatar cannot punch another avatar if they are at greater than arm's length distance. An avatar cannot shoot another avatar if it is outside weapons range.

An avatar's perception also diminishes with distance. For example, an avatar that is far away from the camera may appear as an infinitesimal dot on the horizon smaller than a single screen pixel.

The network gaming literature has focused on improving scalability using Interest Management techniques for filtering the data to be transmitted to each client.

Because interactions between game objects are geometry limited, the Interest filter may be based on geometric distance. Pairs of game objects that are too far away to interact with slow game logic unnecessarily if the game logic still checks for interactions between them. Similarly, game objects too far away for an avatar to observe should not be sent to the avatar's client, both to prevent cheating (see

section 2) and so as not to waste bandwidth. The term Area of Interest is generally used to describe such culling though (depending on context) other terms may be used, such as Area of Effect or Causality Bubble.

(Note that Occlusion can affect the game state because game objects can occlude and therefore shield other game objects. For example, an avatar may not be able to shoot another avatar that is hiding behind a defensive wall. Occlusion also reduces perception. For example, an avatar may not be able to see another avatar hiding behind a defensive wall. So Occlusion is potentially an Interest metric.)

5. Flash Mobs and Variable Area of Interest

Area of Interest is often defined as some sort of fixed bounding volume, but this approach will fail if avatars cluster in the same area of the game[4]. There is no good reason to assume that avatars will not cluster in the same area – if it's an interesting area of the game, the converse is true. Therefore, the worst-case scalability of a game is no better with Area of Interest than without.

VELVET is a proposed Variable Area of Interest scheme[12]. VELVET can overcome crowding or sparseness of avatars and other game objects by contracting or expanding the Interest radius respectively. One can think of this as the game server always sending the game client the data about a fixed number of nearby game objects, the K Nearest Neighbours. Furthermore, the number of neighbours itself can be increased or decreased based on the server's upload capacity, the client's download capacity and the network congestion. This can provide a form of cross-layer optimization between the Application Layer and the Session Layer.

VELVET has been shown to work via simulation. However, implementation could prove tricky, because the system must first calculate the density of nearby objects in game space in order to choose the right radius for the Variable Area of Interest. But the density is found by counting the number of objects within a small radius, which is tantamount to calculating a fixed Area of interest. So this is a chicken and egg problem. We are aware of no implementation of Variable Area of Interest.

6. Spatial Interest Management

Spatial Database techniques have been used in computer graphics and geographic information systems. There are several spatial algorithms for calculating proximity and intersection of points and volumes [13]. These include nearest neighbour, reverse nearest neighbour, distances, and view frustum culling. We hypothesize that similar algorithms could be used to quickly calculate object densities.

The building block for these algorithms is the nearest point lookup. The complexity of looking up the nearest of a list of points to a given coordinate is O(N) as a one-time operation, but can be as low as O(log N) if looked up in an index structure such as a balanced tree.

(So far we have been investigating balanced 3D-trees[14]. A 3D-tree is constructed just like a balanced binary tree, except that it uses three key values - x, y and z spatial coordinates - instead of one. Each split in the tree is based on not just a key value but also a key axis. For example, a branch cannot be marked "<5", but must be marked "x<5" or "y<5" or "z<5".)

Spatial algorithms gain speed by trading the time taken to a build a spatial index against the repeated speedups gained by using that index to look up individual points (or for coarse-grained culling). Using such an index to replace a nested loop algorithm will improve the scalability by a factor of N/(log N) – and the more balanced the tree, the better. Spatial indexing has been used successfully for player partitioning (but for a different purpose) in the game EVE online[15].

This is the motivation for attempting to implement Variable Area of Interest using spatial indexing. We anticipate that the time taken to create the index will determine the effectiveness of the proposal.

7. Summary

Network gaming is increasingly commercially important. Network games face scalability challenges posed by lag and bandwidth. Conventional theoretical solutions, such as distributed simulation, multicast and peer-to-peer techniques, are problematic in an environment of cheating and cloud computing. Network gaming research has shown that a Variable Area of Interest filtering scheme could help solve the scaling problem. But so far it has not been shown how to implement Variable Area of Interest. Our thesis is that it can be implemented using Spatial Database techniques.

References

[1] Reuters Factbox: A look at the $65 billion video games industry, 6 June 2011, Liana B. Baker
http://uk.reuters.com/article/2011/06/06/us-videogames-factbox-idUKTRE7552I20110606
fetched 20 July 2011

[2] Eurogamer: World of Tanks sets MMO server record, 24 February 2011, Wesley Yin-Poole
http://www.eurogamer.net/articles/2011-02-24-world-of-tanks-sets-mmo-server-record
fetched 26 July 2011

[3] Distributed Interactive Simulation for Synthetic Forces
Sixth Heterogeneous Computing Workshop (HCW '97), 1 April 1997, Paul Messina, Sharon Brunette, Dan Davis, Tom Gottschalk Dave Curkendall, Laura Ekroot, and Herb Siege1

[4] VON: A Scalable Peer-to-Peer Network for Virtual Environments, IEEE Network ,
July/August 2006, Shun-Yun Hu, Jui-Fa Chen and Tsu-Han Chen

[5] Quake 2 Source Code .plan, 21 December 2001, John Carmack
http://www.floatingorigin.com/mirror/John%20Carmack-planFilBlog.htm
fetched 26 July 2011

[6] The Laws of Online World Design, Raph Koster
http://www.raphkoster.com/gaming/laws.shtml
fetched 26 July 2011

[7] 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond, 22 March 2001, Paul Bettner and Mark Terrano, Games Developer Conference (GDC2001)

[8] Making Crash Bandicoot, 2 February 2011, Andy Gavin
http://all-things-andy-gavin.com/2011/02/02/making-crash-bandicoot-part-1/
fetched 26 July 2011

[9] Developing Online Games: An Insider's Guide, 7 March 2003, Jessica Mulligan and Bridgette Patrovsky, ISBN: 9781592730001

[10] private email from Bridgette Patrovsky, 10 June 2010

[11] Designs, lessons and advice from building large distributed systems, October 2009, Jeff Dean,
3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware.

[12] VELVET: An Adaptive Hybrid Architecture for Very Large Virtual Environments,
published 2004, Jauvane de Oliveira and Nicolas D. Georganas,
MIT Press Journals - Presence: Teleoperators and Virtual Environments December 2003, Vol 12 No 6

[13] PostGIS 2.0.0SVN Manual SVN Revision (7680)
http://postgis.refractions.net/
fetched 26 July 2011

[14] K-d tree
http://en.wikipedia.org/wiki/K-d_tree
fetched 26 July 2011

[15] The server technology of EVE Online: How to cope with 300,000 players on one server, 2008,
Halldor Fannar Guajónsson, Austin Game Developer Conference, 2008