Student Name	Charlie Long
Student Identification	s4917575
Programme	BSc Games Programming
Project Type	Standard Project
Primary Supervisor	Karsten Pedersen
Secondary Supervisor	Leigh McLoughlin

Distributed Generation & Simulation of Game Worlds : Progress Report

In the submitted full proposal for this project an idea was presented; distribution creation and hosting of individual sections of a game world so as to reduce server costs for game developers. The original architecture plan for this was completely decentralised. This was quickly replaced with minimal centralisation, as a central server for host and client domain distribution would have to be provided to allow players to communicate with each other without manual configuration. To further minimise the scope of the project as required, the central server could have been expanded to include game world hosting and information redistribution, but this negates the aim of generating a decentralised game instance. Instead, multiple methods of minimising required data transmission and maximising client independence will be implemented.

Post-proposal, the time has been spent both looking into academic papers on procedural generation, multithreading networking, distributed computing, peer-to-peer networks, and on the development of the engine that will be used as the basis for the final project. As an aside, testing has also been performed on building for multiple platforms, not just for Microsoft Windows, to better construct a client that will communicate between a wide range of devices. As of yet, not a lot of research has been conducted into network authoritative actions and other methods of reducing the chances of player exploitation.

As far as progress regarding the original timeline goes, the project is a little behind schedule, but techniques learnt through research performed since the timeline was created will help to reduce development time on future modules. If work is increased, as it should now that other academic pressures are reduced, the project should still succeed, even if some stretch goals are not reached such as artificially intelligent agent behaviour and complex voxel generation. A large number of the features described in the original plan are not required for appropriate demonstration of the networking architecture.

Following is the review of articles and other literature that I have found and read during this research.

Distributed Generation & Simulation of Game Worlds : Literature Review

Fragmented World Architecture :

Fragmenting game worlds is useful in both procedural generation, and in facilitating networked information transfer. When a game world is large, and cannot be pre-loaded on startup for any reason, it helps to split the world into equal sized chunks so that sections can be generated or received individually. This reduces initial load times, as chunks can be loaded when required, rather than when the client is started. The procedural voxel terrain generation in Mojang's Minecraft (17 May, 2009) is a great example because a world that is structured this way can be split easily into chunks and distributed to clients in a multiplayer instance when requested. An even more fitting example is that of Bossa's Worlds Adrift (24 May, 2017), which splits its world into fragments which are each hosted by a number of cores in a cluster though the use of a third-party tool, Improbable's SpatialOS. This allows dynamic redistribution of computing power and network bandwidth depending on the number of players in each fragment.

Central Server & Clients Vs Peer-to-Peer :

The traditional architecture for such networks is that of a centralised server providing a number of services. These include, but are not limited to world information collation and distribution, world simulation, client authoritative checks and more. *The provision of a suitable amount of game server resources is crucial for a game. If the game servers are overloaded, players experience high delays. If there are* ***fewer*** *players than expected, the game company faces costly overprovisioning [1].*

Offloading work onto clients themselves, then, is a potential solution to this problem of resource allocation. A central server essentially acts as a middleman, and this can be cut out completely if the architecture is designed correctly. With each client only transmitting changes to the fragments it controls, outbound data bandwidth can be reduced significantly. Techniques, such as world data compression and transmitting world delta from seed generated data instead of current world data can further reduce the amount of data both sent and received each frame. A potential problem with peer-to-peer networks, or networks in general, is handling loss of data in the form of packet loss; *packet loss is important because its frequency determines how often updates must be retransmitted or otherwise recovered [2].*

A number of checks can be performed to rate a client's ability to act as both a data forwarder and a traditional player. The idea of forwarding data to other clients can even be expanded upon, creating an almost hybrid architecture between centralised and peer-to-peer networks. Client specialisation could allow for a modular network of sorts, with plug and play backup, authoritative and other modules. A consistent communication API between clients could allow for further extension of this design, even allowing for user created clients which modify how the game is played for player clients.

Authoritative Clients :

In a peer-to-peer network there arises a problem of authorisation. In a traditionally centralised network the host server can provide authoritative checks to ensure that the data it receives from a client is valid and feasible. This new architecture provides no such service, and so authoritative checks must be performed between clients. Another avenue of exploitation detection is client file hashing and difference comparisons. If the game client itself has been modified significantly from what was distributed, it is likely that the player is attempting to exploit. *In the limit, a player could write* ***their*** *own version of the game client from scratch and still go undetected, providing that the behaviours it emits, as witnessed by the server, are a subset of those that the sanctioned client software could emit [3].*

Client differentiation is another important factor. Ensuring data received is from who it says it is from is an essential step in avoiding exploitation. *First, every player is uniquely identified in the game, through a mechanism such as a public-key system … allows the system to determine if any two identifiers belong to the same individual [4].*

Conclusion :

Although there are a number of large obstacles to overcome, a peer-to-peer fragmented world architecture should be feasible. Through techniques such as transmitting world information as data delta from seed generated content, chunk compression, playerless clients for decentralised backup of world data, cross-client authoritative checks, data forwarding clients and more, the reduction in data transfer should be significant enough to allow for non-trivial numbers of clients sharing a single world instance and interacting with minimal latency.

Distributed Generation & Simulation of Game Worlds : References

[1] G. Schiele, R. Suselbeck, A. Wacker, J. Hahner, C. Becker, T. Weis. Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming (Mannheim, Germany).

[2] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, X. Zhuang. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games (Pittsburgh, USA : Redmond USA).

[3] D. Bethea, R. A. Cochran, M. K. Reiter. Server-Side Verification of Client Behaviour in Online Games (North Carolina, USA).

[4] C. GauthierDickey, D. Zappala, V. Lo, J. Marr. Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games (Oregon, USA).