SuperStreamer: Enabling Progressive Content Streaming in a Game Engine

Yong Xue Eu, Jermyn Tanu, Justin Jieting Law, Muhammad Hanif B Ghazali
Shuan Siang Tay, Wei Tsang Ooi, Anand Bhojan
School of Computing, National University of Singapore
bhojan@comp.nus.edu.sg

ABSTRACT

This technical demonstration presents the SuperStreamer project, which enables progressive game assets streaming to players while games are played, reducing the startup time required to download and start playing a cloud-based game. SuperStreamer modifies a popular game engine, Unreal Engine 4, to support developing and playing games with progressive game asset streaming. With SuperStreamer, developers can mark the minimal set of files, containing only the game content essential to start playing the game. When a player plays the game, these minimal set of files will be downloaded to the player's device. SuperStreamer also generates low resolution textures automatically when a developer publishes a game, and these low resolution textures are transmitted first into the game client. As players move through a game level, high quality textures required for the game will be downloaded. In our demo game, we are able to decrease the time taken to startup and load the first game level by around 30%.

1. INTRODUCTION

Cloud gaming is a form of on-demand service that allows players to run a game of their choice as quickly as possible, with the game content being streamed to the player from a remote server rather than the player having the entire game already in their local drive.

Most game streaming services are now offering game streaming through video streaming (or pixel streaming). The player uses a client provided by the service to access the game. The game runs on on a remote server and a video stream depicting the rendered content of the game is streamed to the player. The client runs on the player's device, takes in his or her input, and sends it back to the server where it is relayed into the game, effectively behaving as if the player is in control of the game. This form of game streaming allows the player to run high- end games on low-end devices, and removes the need to download the game content onto the player's device before being able to play.

Real-time interactive games, however, requires low interaction latency for the game to be playable. Various research recommends a round trip latency of 50 ms for optimal player experience in first person shooter games, a game genre that requires quick reflexes and fast responses. This is a challenge faced by game streaming services as they have to satisfy their customers by providing low latency video streaming of games. To achieve low latency, companies set up servers in various locations globally to cater to geographically distributed customers. The lack of an Internet infrastructure that is good enough to entice gamers to try out cloud gaming is speculated to be one of the reasons behind cloud gaming pioneer OnLive's recent demise.

MM '16 October 15-19, 2016, Amsterdam, Netherlands

ACM ISBN 978-1-4503-3603-1/16/10.

DOI: http://dx.doi.org/10.1145/2964284.2973827

1.1 Cloud Gaming through File Streaming

An alternative form of cloud game streaming has emerged since, focusing on streaming the game content rather than the rendered content [2, 3]. This form of streaming allows players to begin playing the game as soon as a small subset of the game content has been downloaded. Since the game runs locally on the player's device, this approach reduces the need for a low latency Internet connection, at the cost of requiring capable devices to run the game, since the game's processing is no longer offloaded to a remote server.

File streaming game services such as Spawnapps usually determine the minimal set of files required for the game to function to create a small initial download package for the player, before proceeding to stream the remaining files on demand. The methods to achieve file streaming in games are, however, proprietary, and thus the benefits of this form of streaming cannot be shared with the general public.

1.2 Our Solution: SuperStreamer

We set out to modify an existing open source game engine so that developers can build games that can stream its files progressively, giving players a relatively seamless gameplay experience without a long download time. Choosing an open source game engine allows others to learn from, and build on, what we have done. Integrating file streaming into a game engine opens the possibility of making it compatible for more games in the future.

Our solution, SuperStreamer, is a set of tools that will aid game developers using Unreal Engine 4 to package and deploy their games so that progressive file streaming will work off the bat for players of the games. We are targeting single player games.

SuperStreamer is intended to be the first part of a two step approach to future cloud gaming technology, Gamelets [1]. At the point of its conception, the client portion of SuperStreamer enabled games were to be stored and run on modified wireless access points or micro-servers in the network (known as Gamelet nodes), which in turn streams video (pixel streaming) to the resource limited clients few hops away. Gamelets system is a micro-cloud based hybrid system that combines the benefits of both file streaming and pixel streaming.

2. SYSTEM OVERVIEW

SuperStreamer consists of (i) a SuperStreamer server, (ii) a SuperStreamer client (not to be confused with game client), and (iii) a game builder. The SuperStreamer server is essentially a web server responsible for serving game files (referred to as assets from here onwards) to the player. The SuperStreamer client is a modified, lightweight, Unreal Engine client that is able to stream game assets from the SuperStreamer Server. The game builder contains modifications and additions to the Unreal Editor that aid in making the published game ready for distribution to the SuperStreamer Server and the player.

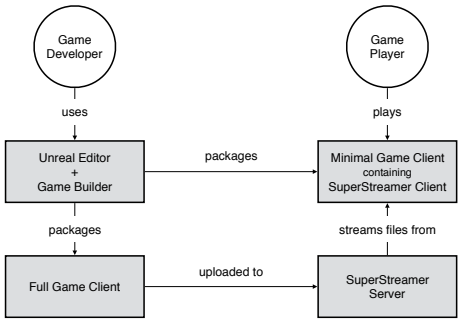

Figure 1: Users' High Level Interactions with SuperStreamer

To build a game, a game developer uses the Unreal Engine editor as per normal, except that they need to mark certain game assets as essential. These are the asset that must be present before a player can play the game. The developer then package the game for publishing using the game builder. The game builder generates lower quality textures for non-essential game assets used in the game, and include these in the game to be downloaded (initially) by the player. The high quality version of the textures are uploaded to the SuperStreamer server. It also generates asset mappings that will be used by the Server to locate the correct assets to send to the player.

To play a game, the SuperStreamer client first downloads the essential game files. Once all the files are loaded, the game is launched, and as the game is played, the SuperStreamer client predicts the high-quality textures that will be needed and prefetches them. Low quality version of the textures are rendered if the high quality version is not downloaded in time.

To prefetch the game assets, we use a combination of distance-based and frequency-based approach to prioritize which assets (i.e., high-quality textures) to download first. For distance-based approach, SuperStreamer periodically determine the distance of objects from the player current position, and assets that are closer are given higher priority than those that are further away. Assets that are referenced more times throughout the entire game are also given higher priority than those that are less frequently referenced, since the gain of downloading these assets are higher.

3. RESULTS

To evaluate the performance of the SuperStreamer Client, we measure (i) the initial download size, which is the total file size of the minimal set of files needed to run the game; (ii) the level load time, which is the time taken before a level is loaded and playable by the user.

We built a demo game using our modified version of Unreal Engine. Most assets (meshes and textures) were taken from the built-in *Starter Content* as well as the free-to-use *Infinity Blade Collection* released by Epic Games. The demo game contained 873 textures, materials, and meshes. The first level to be loaded in this game consists of 75 objects, with different textures on each object. The objects were spread out across the level.

The SuperStreamer-enabled game's initial download size is 139 MB compressed (434 MB uncompressed). At 1.5 MBps, it takes 92 secs to download the initial set of files. The level load time for the first map is 234 seconds. Giving a total of 326 seconds before the game can be played.

In constrast, the same game without using SuperStreamer has an initial download size of 688 MB compressed (1340 MB uncompressed), about 5 times larger. The level load time of the first map is only 10 secs, but it takes 459 seconds to download all the game content, giving a total of 469 seconds before the game can be played. Thus, in the case of our demo game, we achieve a 30% reduction in game loading time.

Note that, players playing the SuperStreamer-enabled game may see low-quality textures when the game starts playing and thus experience lower QoE compared to those playing the non-SuperStreamer version. SuperStreamer, however, caches high quality game assets that are downloaded. Thus, on the next launch of the game, players using the SuperStreamer-enabled game will likely have higher QoE.

4. CONCLUSION

We present SuperStreamer, a system that enabled progressive game asset streaming, with the goal of reducing game play startup time for games that are served from the cloud. Our system uses and modifies Unreal Engine 4, a popular commercial game engine, thus making the SuperStreamer approach potentially available to a large number of game developers.

Beyond this initial proof of concepts, we plan to improve upon the asset prioritization algorithms, taking into consideration multiple players, the server load, the network condition, and the QoE of players. We also plan to integrate SuperStreamer into Gamelets micro-cloud environment to support low-latency video-based cloud gaming.

5. ACKNOWLEDGMENT

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 1 T1 251RES1506 "Cloud-based Multi-user Interactive Virtual Environments".

6. REFERENCES

[1] B. Anand and A. J. H. Edwin. Gamelets – multiplayer mobile games with distributed micro-clouds. In *The 7th International Conference on Mobile Computing and Ubiquitous Networking (ICMU), 2014*, pages 14–20. IEEE, 2014.

[2] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J.-P. Laulajainen, R. Carmichael, V. Poulopoulos, A. Laikari, P. Perälä, et al. Platform for distributed 3D gaming. *International Journal of Computer Games Technology*, 2009:1, 2009.

[3] L. Lin, X. Liao, G. Tan, H. Jin, X. Yang, W. Zhang, and B. Li. LiveRender: A cloud gaming system based on compressed graphics streaming. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 347–356. ACM, 2014.