Streaming Terrain Rendering

Soumyajit Deb*
Microsoft Research

P.J. Narayanan†
CVIT, IIIT Hyderabad

Shiben Bhattacharjee‡
CVIT, IIIT Hyderabad

Abstract

Terrains and other geometric models have been traditionally stored locally. Their remote access presents the characteristics that are a combination of data serving such as files and real-time streaming like audio-visual media. In this sketch we describe a client-server system to serve and stream large terrains to heterogenous clients. This process is sensitive to both the client's capabilities as well as the available network bandwidth. Level of Detail and view prediction are used to alleviate the effects of changing latency and bandwidth. We discuss the design of a terrain streaming system and present preliminary results.

Keywords: Terrain Rendering, Virtual Environments, Collaborative Environments

1 Introduction

Traditional graphics applications typically store geometry locally on secondary storage. Geometry can also be stored remotely and be streamed on the fly when required. Serving of geometry can be beneficial and difficult if network bandwidth is low and latency unpredictable. In situations when data can't be replicated such as a dynamic battlefield visualization system, streaming is the only possible solution. With streaming, different users may read and update different portions of the virtual environment while maintaining a collaborative and consistent system. A similar situation is presented in modern massively multiplayer games. As of now, these games have generally static environments where the basic terrain does not undergo change. This sketch concentrates exclusively on streaming terrain over client-server networks in such situations. Incremental terrain rendering has been well studied lately. [Losasso and Hoppe 2004] try to apply the idea of mipmapping to terrains and uses an image pyramid to represent the various levels of detail of the terrain. [Lindstrom and Pascucci 2005] use view dependent refinement of the terrain for out of core visualization. However there have not been any attempts to explicitly address the networking issues such as latency.

2 Design and Implementation

The basic objective of a geometry server is to provide each client with data appropriate to it as quickly and efficiently as possible. The server must allow the highest quality of rendered output possible for the client and transmit geometry and assets that allow the client to maintain acceptable frame rates. Changing latencies should not cause the system to freeze or hang for long durations during the walkthrough. A block diagram of our system is indicated in Fig 1. The server module maintains state of each client and transmits an optimized representation based on available bandwidth and client capabilities. It also maintains the state of dynamic objects The client module handles interfacing with the server, visibility culling, prefetching and efficient caching. The user program is responsible for the final rendering and user interaction. The system uses visibility culling to reduce the amount of data that is needed to be transmitted. Further, the client locally caches the transmitted data to avoid retransmission when the viewer retraces the path. We utilize the Least Potentially Visible[Deb and Narayanan 2006] caching and

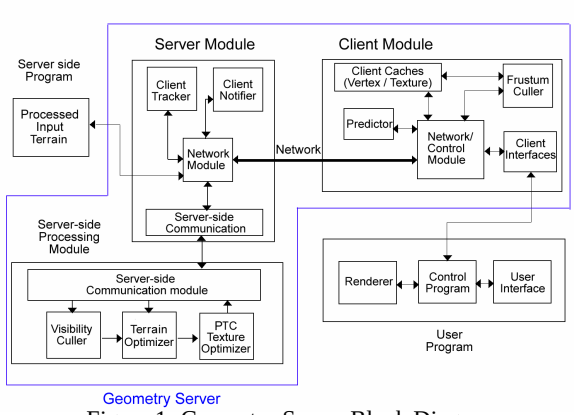

Figure 1: Geometry Server Block Diagram

culling scheme. The system also uses clientside prediction followed by prefetching of data to avoid issues with changing latencies.

We use a tile based terrain representation similar to [Losasso and Hoppe 2004]. This allows us to select only the tiles in the viewing frustum for transmission and also lets us use traditional image compression mechanisms. The client cache maintains tiles of different levels of detail. A lower level of detail tile is never transmitted to the client. The highest level of detail tile to be transmitted is fixed based upon the available client capabilities. For a low end client, we need not transmit extremely large heightmaps. We only transmit the residue between the different levels of detail to save bandwidth. An image pyramid is constructed at the server and only lower levels of the pyramid are transmitted to the client using PTC compression. For a 150MB heightmap of the terrain, we end up transmitting less than a megabyte of data for a 75 second walkthrough.

3 Conclusion

We presented a Terrain Streaming system which adapts to client characteristics and network bandwidth. The system supports dynamic entities in the environment allowing the content developer to create collaborative 3D virtual environments. The system uses a combination of visibility culling, clientside caching, speculative prefetching, motion prediction and deep compression to achieve perfomance similar to local rendering. Streaming systems that serve terrains are especially suitable for applications like Virtual Earth which must transmit large amounts of terrain information. Multiplayer games and flight simulators shall also benefit by utilizing streaming to incorporate new content.

References

DEB, S., AND NARAYANAN, P. 2006. Geometry servers and streaming. *CVIT Technical Report*.

LINDSTROM, P., AND PASCUCCI, V. 2005. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. In *IEEE Transactions on Visualization and Computer Graphics*.

LOSASSO, F., AND HOPPE, H. 2004. Geometry clipmaps: terrain rendering using nested regular grids. In *ACM Trans. Graph.*

*e-mail: sdeb@microsoft.com
†e-mail:pjn@iiit.ac.in
‡e-mail:shiben@students.iiit.ac.in