Streaming Terrains

Francis Chang, Wu-chi Feng
Portland State University
Department of Computer Science
{francis, wuchi}@cs.pdx.edu

ABSTRACT

Streaming computer graphics data is challenging because of the need to retrieve large triangle meshes before any display can begin. This paper proposes and analyzes the benefits of borrowing techniques from lossy image compression to implement a novel technique of progressive terrain rendering for streaming over a network. The goal of the work is to provide a quality-aware framework for 3-D rendering of heightfields.

Keywords: View-dependent progressive mesh, heightfield compression, graphics, streaming

1. INTRODUCTION

Virtual reality systems [active, croquet, gearth, wwind, sl] have risen in popularity with readily available high-speed networking and affordable consumer computer graphics processing hardware. However, the deployment of networking hardware has not kept pace with the increasing quality and complexity of visualization data. Even with such advances, the resolution of such visualization can easily consume any additional gains in bandwidth.

There is a need for techniques and algorithms that are aware of both network and rendering constraints because data and viewers are often not co-located. Models should be transmitted in a quality-aware manner that allows data to be sent continuously in a compact form that allows clients to view data with progressively increasing quality. To maximize the user's experience, the order in which data is transmitted should be dictated by the viewer's local perception.

In this paper, we focus on the streaming delivery of terrain data for fly-overs. The goals of the system are two-fold. First, distant terrain details and data that are outside the viewer's frustum should be transmitted with a low priority. Second, terrain that is near the viewer should be downloaded to the viewing client with high priority. We propose the application of lossy image compression techniques to represent the height fields for terrain data, allowing the fly-overs to start with low latency, while capturing the essence of the terrain being represented. Using the Grand Canyon terrain data set, we compare the efficacy of our approach with several basic graphics streaming engines. Our results show that we can present the user with a high quality interactive experience with smaller delay.

In the following section, we describe some of the related work. In Section 3, we describe our network model and propose a novel height-field representation and compression algorithm to address the problem of terrain streaming.

NOSSDAV, '07 Urbana, Illinois USA.

2. RELATED WORK

Several systems have been implemented for the streaming of computer graphics data. The networked computer game *Second Life* [sl] is a massively multiplayer online dynamic virtual world that allows users to explore a large 3 dimensional space, where players can create and exchange virtual items. Objects are described using a primitive constructive solid geometry model. Terrain and map information are sent in 256x256 patches of non-progressive JPEG data. Using the JPEG patches, rendering is performed with a triangle-splitting algorithm based on an exponential distance metric.

From the computer graphics field, a significant amount of work has been done in the field of progressive meshing. Most of the work in this area focuses on arbitrary 3-dimensional meshes, as opposed to specific optimizations for heightfields, which we explore in this paper. Moreover, the viewer's perspective is not taken into account, resulting in suboptimal viewer-independent streaming algorithms [chen, isenburg, allies].

A network-aware transport protocol has been shown to significantly improve speed and quality of progressive streaming in image data by explicitly modelling packet loss and performing out-of-order data processing [raman]. This approach improves the latency of progressive refinement; however it does not consider prioritization of regions of interest nor 3 dimensional geometry.

For streaming terrain, we can use multi-resolution bitmaps for progressive rendering. [reddy] organizes data in a quad-tree structure, with each child node representing a refinement of one-quarter of the space. This approach takes only the viewers location into account when streaming, without considering the visual importance of existing geological features in the data. [tsai] extends this approach by considering terrain complexity and culling patches outside the viewer's frustum, but does not prioritize information based on viewer distance.

A similar approach streaming terrain approach divides the terrain into square tiles, attempting to pre-cache visible areas around the viewer [pouderoux]. This

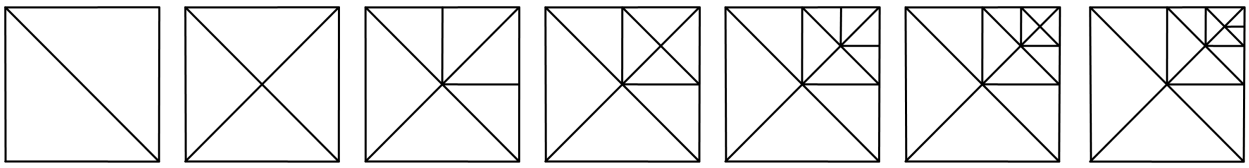

Figure 1: The recursive splitting of triangles in a ROAM patch. This example illustrates progressive refinement to add detail to the upper right-hand corner of the tile. Each vertex represents a rendered height post.

approach tries to minimize computational complexity for CPU-constrained devices by compiling patches into display lists which can be quickly re-rendered by graphics hardware on successive frames.

[duchaineau] proposes a technique for level of detail management to simplify terrain geometry for real-time rendering. ROAM uses triangle decimation to reduce geometrical complexity by considering the visual impact of rendering additional vertices.

3. ALGORITHMS

Before we describe our proposed approach, we first describe some of the basic assumptions we have regarding the system and network.

3.1 System and Network Assumptions

The basic assumptions we make when modelling our system are that

- Local storage and computing power are large relative to network bandwidth.
- The network is reliable and delivers all packets with minimal latency.

These assumptions are chosen to reflect the goal of our research – to construct an algorithm that can deliver a high-quality 3D reconstruction of a terrain over constrained network infrastructure.

The architectural model we follow is to construct a single server and client. The server stores all the world data and sends it to the client in a quality-aware manner. The client is responsible for rendering the scene and sending viewer updated information to the server.

3.2 Lossless Rendering

To simulate a lossless terrain-streaming method, we constructed an adaptation of the ROAM algorithm [duchaineau]. Normally, this algorithm is used for mesh simplification for real-time rendering. In this application, we repurpose the ROAM vertex creation mechanism for use in the prioritization of data for adaptive network streaming. This algorithm will serve as the baseline comparison case for our experimental work.

The ROAM algorithm divides a landscape into square patches that are represented by progressively refined triangle meshes, allowing finer details to be "aggregated" together when network adaptation becomes necessary.

In the coarsest representation, a ROAM patch is represented by two right angle isosceles triangles. As higher detail is demanded, a triangle may be split into two children triangles, introducing an additional vertex (Figure 1).

This triangle mesh is always constructed in a way to prevent the formation of T-junctions – visual cracks in the mesh, formed when two neighbouring triangles are rendered at incompatible detail levels.

In practice, each ROAM patch is represented in memory by a binary tree, with each node representing a triangular area. Each triangle is in turn represented by two smaller triangles that form the descendents of each node. This data structure is referred to as a binary triangle tree (BTT). The BTT is constructed so that travelling down the branches of the tree represents progressive refinement of the terrain mesh, and hence, additional visual detail that can be presented to the user.

In our implementation, there are two BTTs representing each ROAM patch – one on the server, and one on the client. Initially, the server's BTT will be fully populated with the full terrain geometry, while the client's BTT will contain only the coarsest representation. Thus, given infinite resources, the BTT on the client would match that of the server.

Data is sent from the server to the client to populate the client's BTT - the server constructs a vertex stream to send to the client, based on the viewer's location and orientation, using a distance-variance metric for vertex prioritization. This is similar to the way standard ROAM implements its progressive refinement.

In our implementation, the variance of all the child vertices is divided by the distance of the node from the viewer to form a score for each vertex in the terrain mesh. All nodes not yet downloaded are placed in a priority queue for streaming to the client. These scores are recalculated per frame to avoid sending late data.

It is important to note that this distance-variance metric differs from the original paper which uses bounding volumes to calculate screen-space rendering error. Our approach enables us to pre-calculate much of the per-frame node prioritization, as well as simplify the visual-weighting estimation. We believe that such changes would be necessary in a practical implementation of streaming ROAM.

3.3 Lossy Rendering

Our proposed approach to stream terrain data is to represent map geometry as a collection of 2-dimensional tiled bitmaps. In this approach, the height-fields that will be rendered are represented as "image" data and compressed using JPEG [jpeg]. Thus, the pixel intensity in the image corresponds to the height at a given location on our map. Because terrain data is fairly smooth (modulo

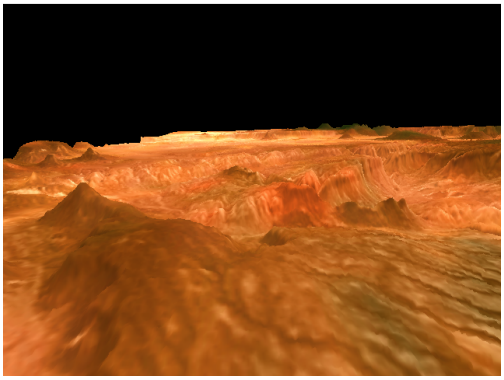

Figure 2: Screenshot of a fly-through of the data

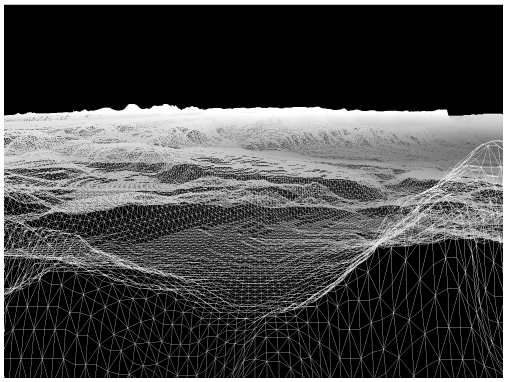

Figure 3: Underlying rendered geometry

cliffs), we expect that such a representation will efficiently represent terrain data.

In our implementation, the entire terrain is divided into 64^2 square bitmaps and compressed using JPEG encoding in progressive mode to allow progressive refinement as data is streamed to the client.

In the simple case, all visible patches are streamed with equal priority. Patches that are outside the viewer's frustum are not downloaded to the client. We refer to this approach as the *jpeg-nopri* approach.

In an extension to this algorithm, visible patches are prioritized with respect to their distance from the viewer and the size of the compressed patch (Equation 1).

$$\text{patch importance} = \frac{\text{size of patch}}{\text{distance from viewer}} \qquad (1)$$

The proximity of the patch to the viewer is used to determine its visual weight, while the size of the compressed patch is used as a coarse metric to determine the patch's geometric complexity.

The bandwidth from the server is divided among visible patches in proportion to the score yielded from Equation 1. This prioritization is very similar to that presented in [pouderoux]. However, their algorithm estimates a tile's viewer independent importance based on its height, whereas our approach approximates visual complexity by its compressed data footprint.

The server overhead for implementing this streaming solution is much smaller than the ROAM-based algorithms introduced in Section 3.1. This is because the calculations for determining priority streaming order are coarser-grained and only require a much simplified understanding of client state.

4. EXPERIMENTATION

We have implemented our system using an OpenGL renderer to simulate various fly-throughs over the terrain, in a 640 x 480 viewport. Example images are shown in Figure 2 and Figure 3. Our simulations are constructed on the framework provided by [turner].

The network streaming is completely simulated in a stand-alone program. The simulation models a network with zero latency and a bandwidth of 56kbps.

The choice of a 56kbps stems from the idea that terrain data should only consist of a portion of a true virtual simulation's network stream. In a realistic scenario the data stream would include information such as objects, buildings, textures and avatars.

This simulation deals only with terrain geometry. Texture information is not sent. In practice, texture information can be generated procedurally. In such approaches, texture is inferred from the terrain geometry and need not be sent over the network.

The output of the client simulations were captured and compared to a full-detail rendering of the walkthrough, using a PSNR metric.

We make the assumption that the viewer is capable of maintaining a constant 25fps refresh rate. Dividing the available bandwidth by the frame rate gives us an allowance of 280 bytes per frame.

4.1 Simulation Dataset

The simulation dataset used in this experiment was the Grand Canyon dataset from The U.S. Geological Survey (USGS) with processing by Chad McCabe of Microsoft Geography Product Unit [usgs]. The subset used for simulation was based on a 2048x2048 grid with 8-bit heightposts (Figure 4), an area of roughly 15000 km^2.

To test our streaming framework, we designed three representative walk-throughs to measure the performance of the various algorithms under different conditions.

The simplest terrain walk-through simulation we use simply crosses the simulated grid diagonally from corner to corner. This crossing is accomplished over 2048 rendered frames.

The second walk-through also traverses the terrain from corner to corner. We augment this walk-through by pausing in the center of the map to rotate the viewer 360 degrees. This requires the streaming system to cope with a changing client orientation.

The third walk-through traverses the grid diagonally while continually panning over the terrain. This is the most demanding of the three walk-throughs, requiring the streaming solution to adapt to a constantly changing viewer location and orientation.

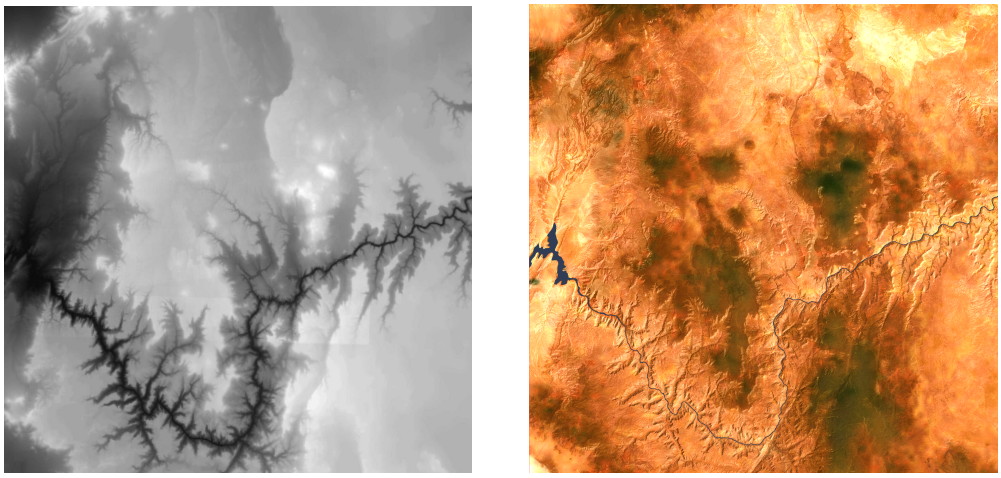

Figure 4: 2048x2048 USGS dataset of the Grand Canyon. The heightfield information is on the left, lighting information is on the right

4.2 Baseline Simulations

The initial baseline was constructed assuming the client has full knowledge of the entire map geometry. The simulations were run with a full level of detail. This represents the ideal case.

To represent the worst-case simulation, the entire terrain is represented as a 32^2 grid, (1 KB of data). This is the coarsest representation our simulation faces (denoted as *plane* in Figure 5).

The theoretical best results for the lossy rendering algorithm is given by *jpeg-full-95* and *jpeg-full-100* in Figure 5, which illustrates a client starting off with the entire jpeg representation, compressed at quality levels of 95 and 100, respectively. This study was performed to determine the quality of the data represented by using a lossy compression method (JPEG). Our experimental results show that the maximum quality that a JPEG-based technique can yield is in the 40-50db range. The difference between a jpeg compressed terrain at 95 and 100 quality is small, but measurable.

Wheon the terrain is compressed with JPEG at quality level 95, the result is a 643,414 byte compressed bitmap. Compressing it with JPEG at 100 quality yields a 1,142,924 byte output, increasing the data size by nearly a factor of two.

As expected, our progressive streaming simulation results fall between the two extremes *jpeg-95* and *plane*.

4.3 Simulation Results (Lossless)

The ROAM-based non-lossy streaming algorithm is illustrated as *roam* in Figure 5. Our simulation counts each vertex as 4-bytes of data (1 byte for height, 3 bytes for XY positional information). This simulation represents the effect of organizing data in a streaming-friendly manner, without applying any compression. Surprisingly, this yields only a marginal improvement in measured image quality.

For comparison, we have also simulated *roammax*, which is the same algorithm, but counts each vertex as only 1 byte of data. This value was chosen in accordance with the compression factors given by [alliez]. Surprisingly, this four-fold improvement in compression results in only a marginal increase in image quality. This suggests that at this level, much more bandwidth is needed to improve the quality of the experience rather than clever management of resources.

The ROAM-based streaming techniques result in "popping" artifacts – temporal discontinuities formed by the sudden introduction of a new vertex to the terrain mesh. These artifacts are not captured by our PSNR metric, but may prove distracting to the viewer. The visual impact of these artifacts can be lessened by introducing new vertices using a geomorphing technique to smooth the geometric transition between mesh refinement levels [hoppe].

4.4 Simulation Results (Lossy)

Our algorithm using progressive JPEG patches is reported by *jpeg* and *jpeg-nopri* in Figure 5. We use JPEG compression with a quality value of 95, to reflect the high end of JPEG's useful working range. *jpeg-nopri* is the case where all JPEG patches are streamed with equal priority if they are visible, while *jpeg* streams data with network priority given to patches closer to the viewer.

Both by *jpeg* and *jpeg-nopri* perform well and are bounded conservatively between the predicted best and worst case simulations. Both algorithms significantly out-perform the non-lossy examples we have implemented. Although *jpeg-nopri* can do better than *jpeg* when the view frustum mis-predicts the future importance of patches, we can see that the *jpeg* algorithm usually gives better results.

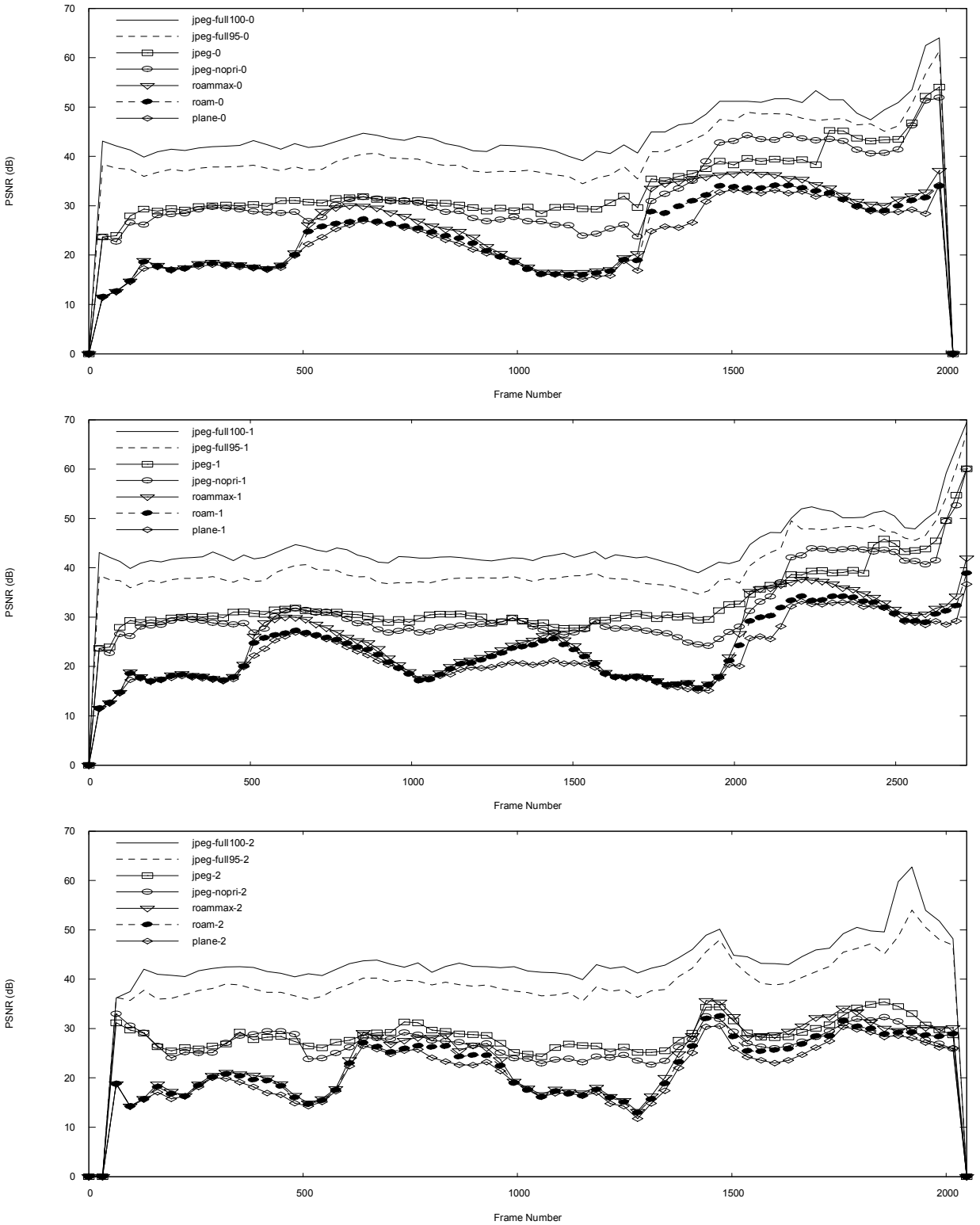

Figure 5: PSNR simulation results. Frame number is on the X axis. The PSNR for that frame (dB) is on the Y axis. The top graph represents a continuous flythrough (0). The middle graph represents a flythrough with a 360˚ pan in the midpoint (1). The bottom graph represents a flythrough with a continuous 360˚ pan (2).

During subjective examination of the rendered output, JPEG "ringing" artifacts are not easily observed – the quality increase in the streaming simulation tends to be fast enough that small inaccuracies are removed before they become too close to the viewer. However, blocking artifacts from neighbouring patches being rendered at different detail levels can be distracting.

The most surprising result is that the complexity/distance prioritized streaming technique performs only marginally better than streaming based solely on visibility. This implies that a high compression rate is more important to the visual quality of the simulation than intelligent prioritization of data. This phenomenon will become more pronounced with larger network latency, due to a less accurate prediction by the prioritization mechanism.

As with the ROAM-based progressive streaming techniques, there are temporal artifacts formed by the sudden progressive refinement of a terrain patch. These problems, as with the ROAM-based algorithms, can be solved by applying a geomorphing technique on newly-refined patches to improve frame-to-frame coherence.

4.5 Future Work

In future work, we plan to examine the benefits of using the JPEG2k compression. JPEG2k has an important property that at low bit-rates, it is able to yield a superior image. We predict that this will result in an improvement in the "ramp up" time for our lossy rendering algorithm.

We will also relax the assumptions used in the design of our algorithm. Currently, we do not perform any geometric simplification between the streamed dataset and the video card. We hope to extend our work to take advantage of LOD algorithms such as ROAM, with explicit understanding of the representation of the data being streamed. At the network layer, a stronger model of packet loss and out-of-order processing can be used, borrowing from ideas in [raman] to further optimize use of the network.

Additional streaming heuristics, such as viewer velocity can also be taken into account to better predict the future relevance of data.

5. CONCLUSION

We have proposed a lossy streaming architecture for the representation of 2-dimensional terrain computer graphics data. This approach has been demonstrated to yield promising results for quality client playback of streaming terrain data. This technique has not yet reached the point of deployability, but our results show the room for potential gains in employing lossy streaming techniques in this domain.

Our experimental results demonstrate the importance of achieving a high data compression ratio in order to provide high-quality streaming terrain. This further underscores the importance of adopting lossy encoding techniques, which can yield much higher compression rates than the non-lossy approaches.

6. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. NSF CNS-0130344. We would like to thank Cory Ondrejka at Linden Lab for discussing their implementation with us, and the anonymous reviewers of an earlier version of this paper for their suggestions and feedback.

7. REFERENCES

[alliez] P. Alliez, C. Gotsman. Recent Advances in Compression of 3D Meshes. In *Proc. of the Symp. on Multiresolution in Geometric Modeling* (Sept. 2003).

[active] Active Worlds, http://www.activeworlds.com/

[chen] B. Chen, T. Nishita. Multiresolution Streaming Mesh with Shape Preserving and QoS-like Controlling. In *Proc. of ACM 2002 International Conference on 3D Web Technology* (Feb. 2002)

[croquet] Croquet Project http://www.opencroquet.org/

[duchaineau] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, M. Mineev-Weinstein. R*OAMing Terrain: Real-time Optimally Adapting Meshes*, UCRL-JC-127870

[hoppe] H. Hoppe. Progressive meshes. In *ACM SIGGRAPH 96* (Aug. 1996).

[isenburg] M. Isenburg, P. Lindstrom. Streaming Meshes. LLNL tech. report UCRL-TR-211608, April 2005.

[jpeg] Independent JPEG Group, http://www.ijg.org/

[gearth] Google Earth, http://earth.google.com/

[pouderoux] J. Pouderoux, J. Marvie. Adaptive Streaming and Rendering of Large Terrains using Strip Masks. In *Proceedings of ACM GRAPHITE 2005* (Nov. 2005)

[wwind] NASA World Wind, worldwind.arc.nasa.gov

[raman] S. Raman, H. Balakrishnan, M. Srinivasan. An Image Transport Protocol for the Internet. In *Proc. Int'l. Conf. on Network Protocols* (Nov. 2000).

[reddy] M. Reddy, Y. Leclerc, L. Iverson, N. Bletter. TerraVision II: Visualizing Massive Terrain Databases in VRML. In *IEEE Computer Graphics and Applications*, vol. 19, no. 2, pp. 30-38, 1999.

[sl] Second Life, www.secondlife.com

[tsai] F. Tsai, H.-S. Liu, J.K. Liu, K.H. Hsiao. Progressive Streaming and Rendering of 3D Terrain for Cyber City Visualization. In *Proc. 27th Asian Conference on Remote Sensing* (Oct. 2006).

[turner] B. Turner. Real-Time Dynamic Level of Detail Terrain Rendering with ROAM. http://www.gamasutra.com/features/20000403/turner_01.htm

[usgs] USGS and Chad McCabe. Grand Canyon Terrain. www.cc.gatech.edu/projects/large_models/gcanyon.html

[yang] S. Yang, C.S. Kim, C.-C. Jay Kuo, View-dependent Progressive Mesh Coding for Graphic Streaming. In *Proc. SPIE Vol. 4518, p. 154-165, Multimedia Systems and Applications IV* (Nov. 2001)