A Quantitative Analysis of Culling Techniques for Real-time Rendering of Digital Elevation Models

Michael Hesse
University of Calgary
Dept of Computer Science
Calgary, AB, Canada T2N1N4

Marina Gavrilova
University of Calgary
Dept of Computer Science
Calgary, AB, Canada T2N1N4

ABSTRACT

The paper is concerned with investigation of effects of culling techniques on quality and smoothness of terrain data visualization within a 3D interactive environment. We utilize the Real-Time Optimally Adapting Mesh (ROAM) approach, extended with a number of efficient techniques such as implicit coordinates method within the patch array representing ROAM and the viewpoint dependent triangle rendering method for dynamic level of detail (LOD) updates. The method, which allows dynamic and interactive first person view, is combined with the View Frustum culling, the Backface culling and an original Relational Position Culling. Standard error metrics are used to verify the rendering consistency. The experimentation is conducted on two data sets representing simple gradual contour changes (Susanville, California) and complex steep contour changes (Kluane National Park, USA). Based on the results, applicability of each of the culling techniques to terrain model is discussed.

Keywords

ROAM, culling techniques, error metrics, terrain data, GIS rendering.

1. INTRODUCTION

The performance of any graphical rendering system is highly dependant on the type and amount of information that is to be displayed. The greater the realism, resolution or detail of the rendered object will increase the overall scene complexity. The increase in scene complexity will decelerate the rendering of the final images. Overly complex scenes can appear sluggish or jagged by the user, which in turn will decreases the overall satisfaction with the rendered scene. Data culling techniques attempt to accelerate the rendering process by eliminating unnecessary information from the scene before it is rendered for the user. Essentially, data culling techniques will eliminate any scene information that will not directly contribute to the final image. Successful culling scene data can increase the rendering speed of a scene without the loss of consistency or realism. There are numerous data culling techniques that have been developed. However, there has been no comprehensive study comparing the performance of those techniques on a variety of data sets, varying from simple gradual contour to complex steep contours.

WSCG SHORT PAPERS proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic

In this paper, we present the results of the empirical study of three culling techniques: View Frustum culling, Backface culling and an original Relational Position culling method (quadrant based), and its effects on the visual contiguity and smoothness of the rendered terrain model. The quadrant-based *Relational Position culling* technique is a new technique that we introduce. This position based method broadly cut unnecessary data during rendering process, based on the user's view point.

In our approach, we also utilize the error metrics algorithms for increased realism of the visualization. Real-time Optimally Adapting Mesh (ROAM) method is selected as an extendable, efficient tool for internal data representation and dynamical updates of the terrain model. The method is extended with an original *implicit coordinates method* within the patch array a*nd the viewpoint dependent triangle rendering* method for dynamic level of detail (LOD) changes. The real-time level of detail reduction method based on the underlying binary triangle tree structure.

Overall, the method is characterized by the following set of features:

- Reduction in the number of triangles rendered

- Smooth, continuous changes between different surface levels of detail
- Dynamic generation of levels of detail in real-time
- Introduction of implicit coordinates method within the patch array for more efficient ROAM representation
- Introduction of the viewpoint dependent triangle rendering method for dynamic level of detail (LOD) updates
- Implementation of culling techniques, including the original Relational Position culling for more efficient terrain rendering
- Application of error metrics for increased smoothness and contiguity

The terrain data sets studied are the simple gradual contour changes (Susanville, California) and the complex steep contour changes (Kluane National Park). The data sets are represented by greyscale Digital Elevation Maps of 1024 pixels by 1024 pixels.
Each experiment set is internally represented by three quadrant detail levels, corresponding to 16, 64 and 128 nodes per side within the binary tree structure. Determining which combination of techniques work best for each dataset is accomplished by examining load time, total number of triangles per path, total number of culled triangles per path, number of frames per second and the number of triangles per frame. In addition, the resulting rendered terrain is examined for cracking and popping to confirm the visual contiguity. The occlusions culling techniques are individually and collectively combined with ROAM technique and examined with the different complexities of terrain data and representations of detail levels within the binary tree. In addition, three error metric algorithms are implemented (measuring cracking, popping and examining the rendered terrain within and outside the view frustum), guaranteeing smoothness and contiguity of rendering.

The examination of unique combinations of culling and error metric techniques allows us to derive a set of heuristics for determination of the most beneficial combination based on the characteristics of the terrain data. This, in turn, leads to the development of dynamically driven GIS system that could immediately adapt to changing conditions based the rendered data. This information is essential for designing more efficient applications that can easily be introduced into the growing Geographical Information System industry, with obvious applications in computer graphics, visualization and computer modelling areas.

2. RELATED WORK

In the past decade, a significant progress has been achieved in developing efficient GIS rendering techniques and a variety of their applications. The research usually is concerned with the data representation and efficient algorithms for performing dynamic level of detail (LOD) changes [Lindstrom, Duchaineauy, DeBerg, Turner, Zhao], implementing culling techniques[Assarsson, Coorg, Zhang], verifying data consistence and smoothness through error metrics [Carr, Lindstrom] as well as utilizing tools for more realistic displaying of data [Blow, Carr].

In 1996, P. Lindstrom [Lindstrom] proposed the real time continuous level of detail terrain rendering algorithm that focuses on a block based mesh structure. In this paper, we expand Lindstrom's error metric design to better fit the ROAM approach and the data culling methods that we introduce. The ROAM algorithm, developed by Duchaineauy [Duchaineauy], was the natural continuation of Lindstrom's algorithm. The ROAM algorithm uses split and merge functions to dynamically adjust similar triangles and construct the terrain mesh. Instead of the ROAM technique being driven by a bottom up approach, as with Lindstrom's algorithm, ROAM uses a priority queue structure to determine which of the triangle diamonds need to be split or merge. These two major enhancements enable the ROAM algorithm to be significantly faster than Lindstrom's algorithm.

In this paper, we maintain the framework of Duchaineauy's ROAM algorithms while examining ROAM's interaction with various culling and error metric techniques. We extend the approach with the implicit coordinates method within the patch array and the viewpoint dependent triangle rendering method for dynamic LOD changes. Alternative data structures (such as TIN, Delaunay triangulation) methods were studied in [Carr, Zhao] however they were not as suitable as the ROAM technique due to their complexity and heavy computations. Data culling techniques are used to determine which information needs not be rendered without loss of realism or generality. The methods chosen for this project have been studied in [Coorg, Assarsonn, Zhang]. In our paper, we utilize normal masks to determine groups of unnecessary data. This technique is easily combined within ROAM. We also introduce a novel Relational Position culling technique for increased efficiency. Quantitative analysis of culling techniques in combination with error metrics provides valuable insights on the best approach depending on the terrain model rendered.

3. TERRAIN DATA REPRESENTATION

Digital Elevation Model (DEM) can refer either to a specific elevation file format or to sources of elevation data in general. Digital elevation model data is usually stored as an array of regularly spaced elevation values, referenced horizontally either to a Universal Transverse Mercator (UTM) projection or to a geographic coordinate system [Docks]. The grid cells are spaced at regular intervals along south to north profiles that are ordered from west to east. A standard grid posting is interpolated directly from the contour files to create DEMs with 10 - 90 meter (<1 - 3 arc second) resolution (depending on the source paper map scale or contour interval). The U.S. Geological Survey (USGS) produces five primary types of elevation data: 7.5-minute DEM, 30-minute DEM, 1-degree DEM, 7.5-minute Alaska DEM, and 15-minute Alaska DEM. In this paper, we use the greyscale Digital Elevation Model (DEM) as the data source. Each shade from black (low) to white (high) represents a height level in context of the entire contour model. The height levels are not defined directly but rather as a ratio from low to high where true black would represent 0% height and true white would represent 100% height. The base dataset DEMs chosen for this project are Susanville, California (Fig. 1) and Kluane National Park (Fig. 2). The Susanville DEM represents progressive contours with gradual elevation changes where the Kluane DEM illustrates sudden steeper elevation changes.

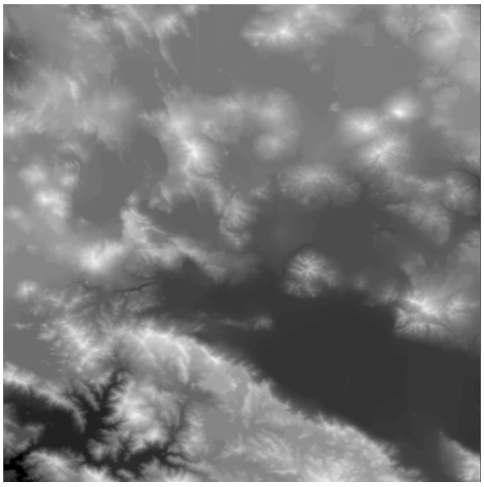

Figure 1. Susanville, California

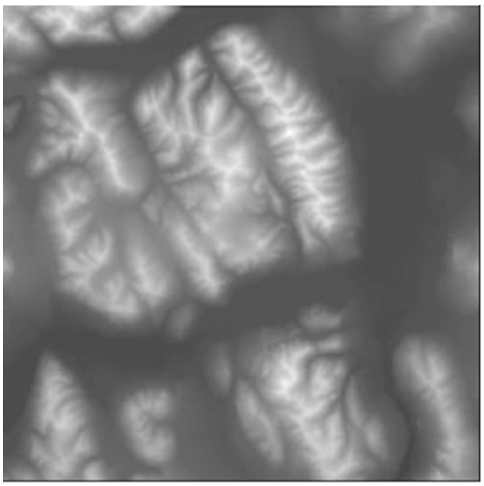

Figure 2. Kluane National Park, Canada

4. REAL-TIME TERRAIN RENDERING

Our approach to terrain data rendering is based on the Real-time Optimally Adapting Mesh (ROAM) method. The idea is to create a system for constructing triangular meshes for view dependent terrain visualization. The ROAM system constructs a consistent and dynamic detail representations of terrain data by utilization of two main priority queues. They are driven by *split and merge* operations that adjust the terrain detail level dynamically. The split and merge functions are both built and changed from the data information held within a preprocessed Binary Triangle Tree data structure. In addition, we use a specific Level of Detail procedure to reduce the total amount of computations needed to render the terrain data.

4.1 Split and Merge Function and Queues

The ROAM algorithm is built around a Binary Tree structure that supplies triangle information for the split and merge functions [Lindstrom]. These two functions are the driving force behind the dynamic view-dependant rendering method. Two triangles that share the same base and are on the same detail level are referred to as a *diamond* (Fig. 3). The split operation adds a new vertex at the diamond's center resulting in the creation of four new right-isosceles triangles, which will increase the number of triangles representing a terrain area. As the number of triangles increase, the detail level that can be represented will also increase. The merge operation works inversely to the split operation [Duchaineauy].

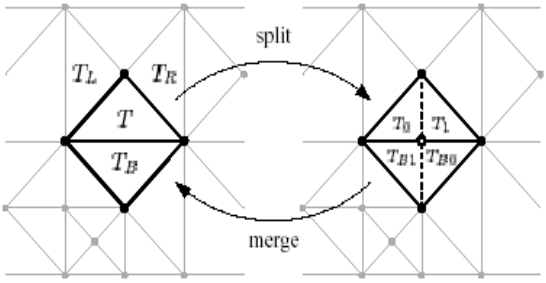

Figure 3. Split and Merge Operations

In our implementation, the split and merge operations provide a flexible framework for making detailed updates to the triangulated mesh. The basic idea of each queue is to keep the priorities for each individual triangle in the mesh triangulation. The split operations would then start with the base triangulation level in the queue and then repeatedly split the triangle until the highest priority triangle is reached. The only requirement for the split priority queue is that the child's priority level must not be more than its parent's. The merge priority queue allows the merge operation to start from the previously rendered mesh triangulation. This allows a more consistent and quicker frame-to-frame coherence. The merge operation, which is similar to the split queue, uses this queue as the starting point of approaching merging of triangles.

4.2 ROAM Implementation

The ROAM implementation used in this paper is an extrapolation of three main sources. In [Duchaineau], the basic Real-time Optimally Adapting Meshes algorithm that is based on a binary tree structure is given. Each tree node references isosceles right triangles that can be split recursively and stored as the nodes child. The recursive splitting and storing would continue until the desired detail level is reached. In [Lindstrom], a structure called a Quad tree that is used to represent a patch or quadrant with the total landscape is described. A Quad tree recursively tessellates the landscape creating an approximation of the height dataset. The Quad trees are very simple and efficient, however the atomic mesh polygon is changed from the isosceles right triangle to a patch of triangles. Combinations of the above techniques are studies in [Turner].

This paper continues on with the idea of patches and quadrants described in [Lindstrom] and [Turner], which we use to create and manage the mesh approximation within the terrain's landscape. We use the *implicit coordinates method* within an array of patch objects to ensure more efficient memory usage. Implicit coordinates, within the landscape, are stored for the isosceles right triangles that will be rendered onscreen. The advantage of our approach to data representation is that implicitly defining coordinates saves 36 bytes of RAM per triangle by not having to reference the explicit X, Y and Z coordinates. An index within the patch array references an individual Binary Triangle Tree that in turn stores the references to each triangle level of detail for that patch. The size of the patch determines the relative size of each patch within the landscape. The array size must be defined before the program can be executed. The patch objects are held within the Landscape object. The landscape object is built by combining each patch section until the entire terrain is rendered.

4.3 Level of Detail (LOD)

A conventional method to reduce the amount of computations needed to render a complex scene is to apply Level of Detail (LOD) techniques [Turner]. In this paper, we extend the LOD technique with the *viewpoint dependent triangle rendering* method, which allow more flexible information storage for dynamic and interactive first person view rendering. To accommodate this feature we allow portions of the terrain that are currently too far away to be rendered with few triangles, and also allows the same sections of terrain to be rendered with more triangles if the viewpoint moves closer. We accomplish this task by examining the field of view with the view frustum to determine which patch sections need more detail due to their proximity to the user.

4.4 Binary Tree Triangle Data Structure

To satisfy the LOD requirement a binary triangle tree structure will be used to hold the various levels of detail that is needed by the graphics-rendering engine. In the case of ROAM, a binary triangle tree structure, or a *bintree*, is a recursive structure where, at its lowest level, represents a right-isosceles triangle. Each subsequent level of the tree splits the triangle by subdividing. This is accomplished by adding an edge from the apex vertex to the midpoint of the hypotenuse of the triangle. This division produces two smaller right-isosceles triangles that can be further subdivided. The left-child representation in the bintree is the child that is on the left splitting edge when the triangle is viewed with the apex point on top and the base at the bottom. The right-child is the child to the right of the splitting edge. The base-neighbors is the neighboring tree that shares the hypotenuse edge of the tree and the right and left neighbors are those adjacent triangles that

share the hypotenuse of the right and left children respectively.

In our implementation, each patch of terrain will have an individual bintree to define the triangle detail levels. The triangle bintree structure starts with the base terrain, either the least detail representation or the detail level from the previously rendered image, in the leaf components of the structure. If a particular leaf, which holds an individual triangle of the meshed terrain image, needs to increase in detail, then the leaf will split into smaller pieces or new leafs. This transformation will also cause the triangle represented by that leaf to split. We determine the need for a change in detail level by examining the corresponding error metrics (see Section 5).

5. DATA CULLING TECHNIQUES

Data culling is a process of selecting, from the whole scene, particular information that needs to be rendered. Culling at this level is often achieved by using geometry-based methods to determine which scene information needs to be rendered. Although geometric approaches are popular, there are other algorithms that can be used to accomplish the same task by examining the relationships and positions of scene data without directly calculating geometric measures. In this section, three types of geometric culling algorithms that we implements are discussed: View Frustum Culling, Backface Culling and an original Relational Culling technique based on a pre-process point of view approach.

The view frustum is the volume of space that includes everything that is currently visible from a given viewpoint. Six planes arranged in the shape of a pyramid with the top removed, define the view frustum area. If a point or object is inside this volume then it is within the frustum area and is potentially visible. Although the point or object is within the view frustum area, it still may not be rendered due to its positional relationship with the other points and objects within the users view. If a point is outside of the frustum then it is not visible to the user and it needs not be rendered. To determine the position of the points and object, their bounded volumes are computed. Bounded volumes are illustrated by surrounding objects with a specific geometric shape that are located near to each other. The geometric shape is then referenced to determine the position of the object with respect to the frustum area. If the bounded volume lies on one of the frustum edges then that bounded volume is further subdivided into smaller bounded volumes until each object is either determined to be inside or outside the frustum area. If at the lowest detail level an object still lies on a frustum edge, the portion of the object inside the frustum area is rendered while the rest is culled.

Historically, the geometric shapes used as bounded volumes are boxes, or spheres. For this project, geometric spheres have been chosen. We store the essential information described by the bounded spheres in the hierarchies of bounded volumes as a Direct Acyclic Graph (DAG). The root node of DAG is connected to the subsequent inner bounded volume nodes, which in turn continue until the final child level of the graph. This level holds the individual object descriptions. This structure will allow for quick and easy access of object information based on their relative positions.

The second method that we implement in this project is the *Backface culling*. Based on a user's eye-space, Back-facing polygons are located on the far side of an opaque object. Once the polygons are determined to be Back-facing, they are culled before the scene is rendered. We calculate the normal of the projected polygon. Testing polygons within a scene for back-facing properties requires that each polygon be subjected to the back-facing test. This test involves calculating the polygon's normal and the vector formed from the viewing point to any point on the polygon. Although the rendered scene is displayed in three-dimensional space the normal calculation is calculated as if the scene were represented in two-dimensional space. The geometric calculation will result in either $(0,0,a)$ or $(0,0,-a)$ where $a>0$. A negative a represents a polygon that is pointing into the screen, or front facing where a positive a represents a back-facing polygon that in turn will need to be culled from the scene.

The third technique that we introduce is the original *Relational Position Culling* technique, based on pre-processing the terrain landscape into patches (see Figure 8). Each patch would contain the Binary Triangle Tree structure of its terrain data and store each triangle's detail information within its node. Additionally, a visibility flag is stored to determine which patch is seen within the view frustum. This approach is developed to quickly cut the generalized unnecessary terrain data from the terrain data set. Initially, the algorithm determines the frustum triangle corners from a two-dimensional (2D) view frustum, which gives the algorithm the user and user's viewpoint's positions. These three points are used to determine the minimal rectangle that encompasses the 2D view frustum. Any points not within this rectangle are immediately culled. Advantages of this method is its simplicity and performance, that will be discussed in the experimental section.

6. ERROR METRICS

The ROAM method described above in combination with dynamic LOD algorithm is an accurate and consistent algorithm for constructing terrain meshes that optimize a flexible, view-dependant error metrics. The use of error metrics in this fashion produces guaranteed error bounds that achieves specified triangle counts directly. We utilize three major error metric techniques in order to obtain realistic terrain rendering and validate the proposed culling techniques. They help to determine which polygons need to be rendered and to validate the terrain for realism.

Cracks are introduced in a rendered terrain mesh when two adjacent nodes are not subdivided to the same detail level. The addition of cracks will prevent the user from gaining an accurate or realistic perception by viewing the rendered terrain. To avoid cracks ensure that the two neighbours nodes have the same number of vertices before being rendered. Checking the node neighbours in all four directions while rendering the node can determine if a crack exists.

Popping occurs when the rendered terrain does not move smoothly giving the impression that the background mesh pops up and down due to the adding and removing of vertices. Popping can be rectified by producing only a limited number of levels of details based on the terrain data and can measured visually by looking at the terrain as it is being rendered.

Using *View First Frustum error metric* easily allow the terrain to be rendered to differentiate detail level by examining the rendered terrain within and outside the view frustum.

7. EXPERIMENTAL RESULTS

The algorithms were implemented in Open GL environment, in a form of a user-friendly real-time interactive software, which also allowed 'fly-through' over the rendered terrain. A variety of parameters could be selected from the drop-down menus, such as patch sizes, culling technique to be used, error metrics, rendering views or automated pre-selected paths. Numerous statistics were measured, including initialization time, number of culled triangles, number of triangles per path, number of frames per second and the number of triangles per frame (Figure 4).

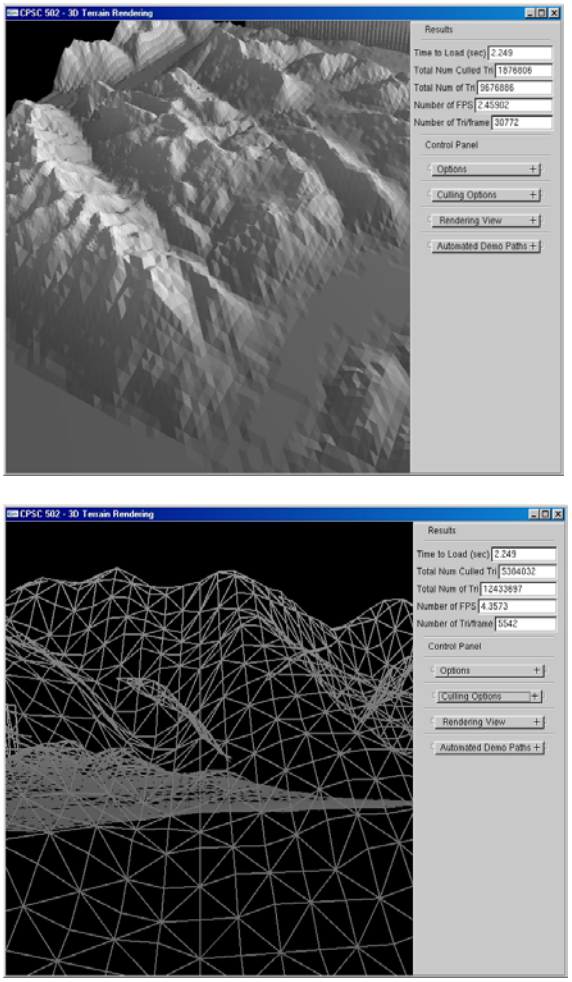

Figure 4. Sample Rendering of ROAM (above-surface, below-triangulation)

The main goal of this research was to determine more efficient methods for terrain data representation, based on the dataset attributes. This was accomplished by examining each combination of culling techniques, View frustum, Position based and Backface culling on two different terrain data sets. The first data set is a representation of the town of Susanville, California, which symbolizes a set of simple and gradual terrain height contours. The second dataset, Canada's Kluane National Park represents a more detailed and steep set of height points. Each terrain data set will have a consistent input path that will be defined prior to each map being loaded. Both terrain datasets will be examined by the size of each patch, which is referenced, by the size of its corresponding patch array. The array size number represents the size of the 2D array.

When examining the results from the aforementioned experiment, several relationships were observed throughout the entire procedure. One of the most notable observations was the change of the number of frames per second (frame rate) during each of the paths corresponding step. Figure 5 demonstrates the change in frame rate with View Frustum, Position based and Backface culling enabled with three distinct patch sizes represented by their corresponding array size. The size-16 frame rate performs as expected with the graph trend line remaining rather flat and consistent throughout the entire experimental path, except for its initial load up stage, which is completed by the 25th frame. The size-64 frame rate demonstrates some interesting qualities. The first 100 frames emulates a similar pattern as the size-16 trend line with the exception of a consistently lower frame rate due to the increase of the number of patches that need to be rendered. The size-64 trend illustrates a significant increase in frame rate from frame 100 to 170.

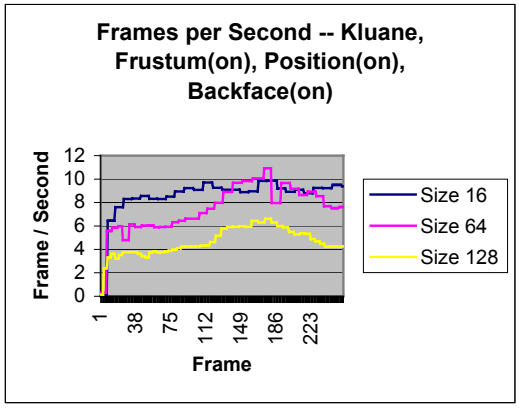

Figure 5. Frames per Second Chart

The same pattern is noticed when examining the number of triangles per frame (Fig. 6). The decrease in triangles per frame also corresponds to the first left hand turn. This is due to the quick pace of the turn and the difficultly of the rendering engine to propagate the necessary triangle detail levels before the next turn begins. As each turn is performed, the viewed landscape's true detail level is reduced.

The number of triangles per frame continues to reduce as the turn progresses. This is the result of the view frustum largely shifting out of the frame of view. The trend line flattens as the forward movement allows the rendering engine time to increase the detail level of each frame, due to the limited changes in the view frustum. As the next turn progresses, the number of triangles per frame reduces and the trend line continues to decrease. The rise of the size-64 trend line frame 180 is due to quick left and right turns which would leave the middle section of the view frustum untouched with only the frustum edges needing to be recomputed. The largest increase in the size – 64's trend line is due to the experimental path moving directly backwards.

The trend line in Figure 6 also demonstrates the experimental path's turn and increases its steep drop corresponding to the reduction of the number of triangles per frame. A similar correlation can be seen through examining the size-128 Number of triangles per Frame trend lines. As expected, due to the even smaller patch size, any view frustum change is amplified and can be seen as a more significant change in its trend line. Additionally, the size-128 Frames per Second trend line illustrates the same basic relationship as the size-64 trend line, but due to the size-128's smaller patch sizes, the change between frame rates appears more gradual.

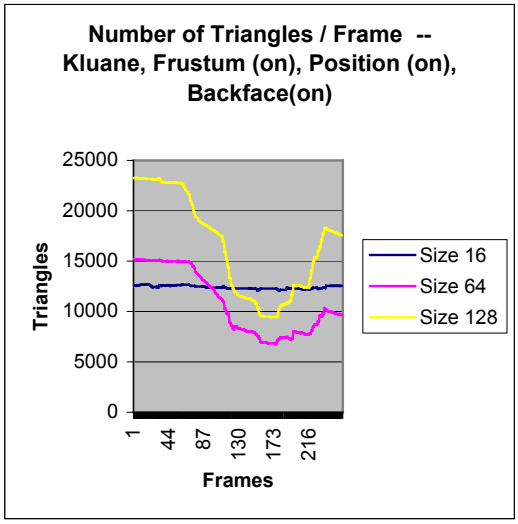

Figure 6. Number of Triangles per Frame

The experimental path using Susanville data, illustrates the same consistencies as the Kluane data. This demonstrates that the change of landscape parameters and detail levels hold little correlation of this experimental path's resulting data.

Additional experiments illustrate that having either the Position based culling or the View Frustum culling techniques correlate in the same fashion as Figure 6, both Position and View Frustum techniques active. However, the position based culling is more susceptible to the frustum change than the frustum technique.

The time management relationship of the split and merge algorithms was also explored. The experimentation demonstrated that patch size has a correlation between the times being spent in each algorithm. Larger patch sizes effectively provide frustum buffer that allow the merge algorithm time to reduce the triangle count of the patch before it is eliminated. As the patch size becomes smaller the merge algorithm becomes less effective as the split function is now more in demand.

8. CONCLUSIONS

The main contribution of this paper is in the development of adaptive real-time rendering algorithm based on ROAM technique, combined with culling and error metric techniques for increased rendering speed, smoothness and realism. A novel Quadrant Based Position culling technique is introduced.

Quantitative analysis of culling techniques in combination with error metrics is performed and provides insights on the best approach depending on the terrain model rendered. Examining both the number of frames per second and the number of triangles per frame suggests a number of correlations. When investigating the experimental path's frame rates in respect to culling techniques, it was shown that the patch sizes within the landscapes are significantly related to the change of frustum position. This correlation is confirmed by examining the number of triangle rendered per frame. Within the experimental culling technique sets, the results indicate that Backface culling was a less dominate method of View frustum or the Position based culling techniques. Another conclusion that can be drawn is that the Position based culling, while working in broad cuts, is significantly related to the change in movement of the frustum. View frustum culling had similar effects as the Position based culling but was not as heavily related to the motion of the frustum.

9. REFERENCES

[Assarsson] Assarsson, U. and Moller, T. Optimized View Frustum Culling Algorithms for Bounding Boxes, Journals of Graphics Tools, 2000.

[Blow] Blow, J Terrain Rendering at High Levels of Detail, in Proceedings of the Game Developers' Conference, San Jose, California, USA, 2000.

[Burtch] Burtch, B Geographic Information Systems. 2000.

[Carr] Carr, J. Data Visualization in the Geosciences, Prentice Hall 2002.

[Coorg] Coorg, S. and Teller, S. Real-Time Occlusion Culling for Models with Large Occludes, ACM Symposium on Interactive 3D Graphics, 1997.

[DeBerg] De Berg, M. and Dobrint, K. On levels of detail in terrains, Proc. 11th ACM Symposium on Computational Geometry June 1995.

[Docks] Docks, J. Another Introduction to GIS, New York: Bogus Press, 1997.

[Duchaineauy] Duchaineauy, M, Wollinshy, M. ROAMing Terrain: Real-Time Optimally Adapting [Meshes] Meshes, IEEE Visualization '97 Proceeding, 1997.

[Garland] Garland, M.and Heckbert, P. Surface Simplification Using Quadric Error Metrics, SIGGRAPH 99, 1999.

[Lindstrom] Lindstrom, P, Koller, D. Real-time continuous level of detail rendering of height fields, Computer Graphics, SIGGRAPH 1996 Proceedings, pp 109-118, 1996.

[NMD] National Mapping Division, U.S. Geological Survey, US GeoData Digital Elevation Models http://geog.hkbu.edu.hk/QZone/Teaching/.

[Turner] Turner, B. Turner. Real-Time Dynamic Level of Detail Terrain Rendering with ROAM, Gamasutra 2000.

[Zhang] Zhang, H. and Hoff, K. Fast Backface Using Normal Masks, SIGGRAPH 00, 2000.

[Zhao] Zhao, Y. Ji, Zhou et al. A Fast Algorithm For Large Scale Terrain Walkthrough, CAD/Graphics '2001, 2001.