DStrips: Dynamic Triangle Strips for Real-Time Mesh Simplification and Rendering

Michael Shafae and Renato Pajarola
Computer Graphics Lab
School of Information & Computer Science
University of California, Irvine
mshafae@ics.uci.edu, pajarola@acm.org

1 Motivation

Multiresolution modelling techniques are important to cope with the increasingly complex polygonal models available today such as high-resolution isosurfaces, large terrains, and complex digitized shapes [10]. Large triangle meshes are difficult to render at interactive frame rates due to the large number of vertices to be processed by the graphics hardware. Level-of-detail (LOD) based visualization techniques [7] allow rendering the same object using triangle meshes of variable complexity. Thus, the number of processed vertices is adjusted according to the object's relative position and importance in the rendered scene. Many mesh simplification and multiresolution triangulation methods [5], [8], [4], [11], [12] have been developed to create different LODs, sequence of LOD-meshes, and hierarchical triangulations for LOD based rendering. Although reducing the amount of geometry sent to the graphics pipeline elicits a performance gain, a further optimization can be achieved by the use of optimized rendering primitives, such as triangle strips.

Triangle strips have been used extensively for static mesh representations since their widespread availability through tools such as the classic *tomesh.c* program [1], Stripe [6] and the more recent NVidia NVTriStrip tools [3] [2]. However, using such triangle strip representations and generation techniques is not practical for a multiresolution triangle mesh. The problem of representing the stripped mesh and maintaining the coherency of the triangle strips is compounded when used with LOD-meshes. In view-dependent meshing methods the underlying mesh is in a constant state of flux between view positions. This poses a significant hurdle to surmount for current triangle strip generation techniques for two core reasons. First, triangle strip generation techniques tend to require too much CPU time and memory space to be practical for interactive view-dependent triangle mesh visualization. Secondly, most triangle strip generation techniques focus on producing optimized strips, but not managing the strips in light of continuous changes to the mesh. That is, for each new view position a new stripification must be computed. Our approach, on the other hand, manages triangle strips in such a way that reconstructing the entire stripification never has to be done. Instead, it either grows triangle strips, shrinks triangle strips, or recomputes triangle strips only for small patches when necessary.

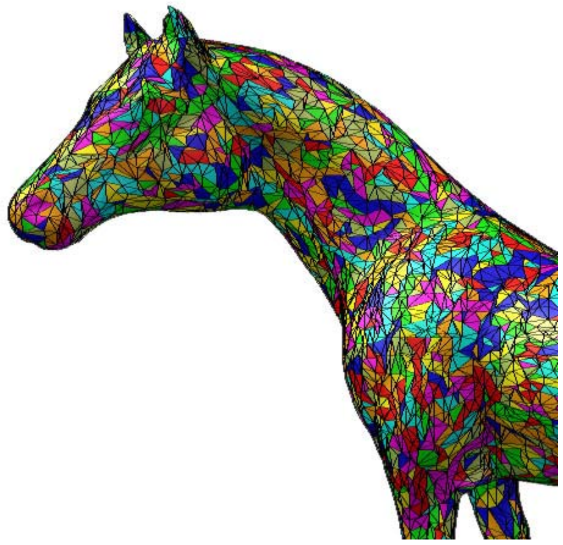

Figure 1. Example of dynamically generated triangle strips of a view-dependently simplified LOD-mesh. Individual triangle strips are pseudo-colored for better distinction (15548 triangles represented by 3432 triangle strips).

In this short paper and poster, *DStrips* is presented. DStrips is a simple yet effective triangle strip generation algorithm and data structure for real-time triangle strip generation, representation, and rendering of LOD-meshes. The implementation presented in this paper is built on a LOD-mesh using progressive edge collapse and vertex split operations [9] based on a half-edge representation of the triangle mesh connectivity [13]. However, DStrips is not tightly coupled to any one particular LOD-mesh. DStrips is easily adapted to any LOD-mesh so long as the mesh provides a mapping from an edge to its associated faces and vice-versa. Also, the edges of a face must maintain a consistent ordering and orientation in the LOD-mesh. Figure 1 presents an example screenshot of pseudo-colored triangle strips of a view-dependently simplified LOD-mesh that were generated by DStrips.

2 Innovative Aspects

Unlike other LOD-mesh with some sort of triangle stripping support, DStrips does not merely shorten the initially computed triangle strips. Rather, DStrips dynamically shrinks, grows, merges and partially recomputes strips. Table 1 briefly compares other approaches which couple triangle strips with an LOD-mesh.

Name	Algorithm	Stripification	Strip Management
Dstrips	Online	Dynamic	Shorten, Grow, Merge, Partial Re-Strip
Tunneling	Online	Dynamic	Repair & Merge (Tunneling Operation)
Stripe	Offline	Static	Not Applicable
Skip Strips (Stripe)	Pre-Process	Static	Resize Pre-Computed Strips
Mulltiresolution △ Strips	Pre-Process	Static	Resize Pre-Computed Strips

Table 1. A comparison of triangle stripification techniques. Note that a clear distinction can be drawn between the techniques which dynamically manage the triangle strips and those which shorten pre-computed triangle strips.

To illustrate the novelty of our approach, experiments were performed on a Sun Microsystems Ultra 60 workstation with dual 450MHz UltraSparc II CPUs and an Expert3D graphics card. Table 2 shows the sizes of the different models we used for testing DStrips.

Table 2 shows the average number of faces, LOD-updates and triangle strips encountered each frame. The time to perform the edge collapse and vertex split updates each frame is also recorded here since it is independent of the rendering mode. The average number of triangle strips per frame is given for the three stripping configurations: adjacency stripping, greedy stripping allowing swap operations and greedy stripping without swap operations (strictly left-right). One can see from Table 2 that adjacency stripping generates fewer strips than greedy stripping, in particular if strict left-right alternation is enforced.

Model	#	#	# △	#	Update	# Strips		
	Faces	Vertices	Drawn	Updates	Time	ADJ	GS	GNS
happy	100,000	49,794	54,784	358	3ms	7,006	8,127	12,143
horse	96,966	48,485	39,584	519	4ms	5,008	5,428	7,808
phone	165,963	83,044	60,291	498	5ms	7,272	7,904	11,382

Table 2. The model's name, total number of triangle faces, total number of vertices, per frame average numbers of rendered triangles, LOD-mesh updates, and time to perform mesh updates. The average number of triangle strips is divided into adjacency stripping (ADJ) as well as greedy stripping with swap (GS) and without swap operations (GNS).

3 Conclusion

DStrips, a simple and efficient method to dynamically generate triangle strips for real-time level-of-detail (LOD) meshing and rendering. Built on top of a widely used LOD-mesh framework using a half-edge data structure based hierarchical multiresolution triangulation framework, DStrips has shown efficient data structures and algorithms to compute a mesh stripification and to manage it dynamically through strip grow and shrink operations, strip savvy mesh updates, and partial re-stripping of the LOD-mesh.

References

[1] K. Akeley, P. Haeberli, and D. Burns. The tomesh.c program. Technical Report SGI Developer's Toolbox CD, Silicon Graphics, 1990.

[2] Curtis Beeson and Joe Demer. Nvtristrip v1.1. Software available via Internet web site., November 2000. http://developer.nvidia.com/view.asp?IO=nvtristrip_v1_1.

[3] Curtis Beeson and Joe Demer. Nvtristrip, library version. Software available via Internet web site., January 2002. http://developer.nvidia.com/view.asp?IO=nvtristrip_library.

[4] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[5] Leila De Floriani and Enrico Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, 1995.

[6] F. Evans, S. Skiena, and A. Varshney. Optimizing triangle strips for fast rendering. In *Proceedings IEEE Visualization 96*, pages 319–326. Computer Society Press, 1996.

[7] T. Funkhouser and C. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings SIGGRAPH 93*, pages 247–254. ACM SIGGRAPH, 1993.

[8] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. SIGGRAPH 97 Course Notes 25, 1997.

[9] Hugues Hoppe. Progressive meshes. In *Proceedings SIGGRAPH 96*, pages 99–108. ACM SIGGRAPH, 1996.

[10] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings SIGGRAPH 2000*, pages 131–144. ACM SIGGRAPH, 2000.

[11] Peter Lindstrom and Greg Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):98–115, April-June 1999.

[12] David P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics & Applications*, 21(3):24–35, May/June 2001.

[13] Kevin Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics & Applications*, 5(1):21–40, January 1985.