View-Dependent Simplification for Glyph-Based Visualizations of Large Datasets

Stephen R. Saucier <stephen.saucier@umbc.edu>
University of Maryland Baltimore County

Abstract—***View-dependent simplification and similar dynamic simplification algorithms are relatively recent developments for the simplification of polygonal environments. Such algorithms operate by classifying the vertices in a hierarchical fashion and dynamically generating polygons based not only on the original vertices but also the current viewpoint. Unimportant areas can be greatly simplified while those that are of immediate interest are rendered with a high degree of detail.***

Traditionally, little consideration has been given to the idea of applying view-dependent simplification to glyph-based visualizations. Often many glyphs overlap, resulting in very complex areas that do not provide much additional information. Here it is the outlier glyphs that are of primary interest; as the glyphs are an arbitrary representation of the data points, it is not necessary to preserve the geometry of each glyph exactly as long as the spatial relationships of the data points are preserved—a perfect application for view-dependent simplification.

Preliminary results applying view-dependent simplification techniques to glyph-based visualization of document data produced compelling results, providing a large reduction in polygons while still maintaining enough detail to preserve the important areas of the original.

Keywords—***View-Dependent Simplification, Level of Detail, Glyph.***

I. Introduction

Polygons have been used as a basis for representing geometry in the field of computer graphics for some time. The popularity of this representation is the result of many factors; a few of the more important reasons are that they are easy to describe (only three points are required to draw a triangle) and that intersections with triangles are easy to calculate. As a result, rendering polygons is fast and inexpensive. Polygons serve as a good way of representing data; almost any other type of geometric description can be converted to polygons and maintain a high degree of accuracy. However, the tendency with polygons is to try to include as much detail as possible, and as a result, the number of polygons that need to be rendered often is greater than the available video hardware can support.

In the field of data visualization, polygons serve as the underlying representation of the geometry, but it is not the polygons themselves that are of immediate concern; it is the higher-level abstractions (which happen to generate the required polygons transparently for efficient rendering of the representative geometry). One such abstraction which is commonly used for representing data in this way is with *glyphs*. Glyphs are simple geometric objects (usually a primitive shape such as a sphere or cube) that are used to represent the data. Glyphs have been used successfully in a number of different ways, including information visualization [13] as well as an efficient mechanism for rendering volumes [14].

One of the recent methods of using glyphs is to visualize document data obtained through information retrieval techniques. In this sceme, the various attributes of the glyph such as the color, size, or location can be given values based on properties of the document. For instance, similarities generated by the information retrieval engine for a variety of queries can be used to assign a location to the glyph along each of the primary coordinate axes [1]. Assuming that the document pool is sufficiently large, the amount of glyphs becomes large enough that the number of polygons becomes an issue for the display hardware. Even though each glyph has a simple geometry, the volume of glyphs alone is such that the overall geometry is very complex. However, as a high-degree of interactivity is needed to effectively navigate and gain an understanding of the data, it is clear that some sort of simplification is necessary.

While the quickest and easiest method to solve this problem is to simply apply a threshold to the data before generating the glyphs, thus masking away the documents that are of lesser interest, this is not a desirable solution because it is necessary to understand the overall distribution of the data, and simply deleting less important glyphs makes this impossible.

A simple improvement is to display co-located points only once; however this does not provide enough of a gain as even documents with a low relevance to a given query will often not be completely irrelevant. This results in a tight distribution of glyphs within a relatively small range of values.

As a result, it is clear that some more advanced simplification mechanism in necessary. It is necessary to preserve the general structure of the glyphs as this information is crucial to understanding the data; however, maintaining the original amount of detail is not required since the glyph is an arbitrary representation (assuming that some glyph attributes such as size and orientation were not originally meaningful).

II. Relevant Simplification Concepts

In order to forge a compromise between the detail and complexity present in many datasets and providing smooth frame rates and a high degree of interactivity to the user, there has been a desire to simplify the polygonal data into a form which provides as much detail as possible in the current view while making simplifications where possible.

Level-of-Detail

Historically, simplification has been achieved by creating several versions of the data with varying degrees of resolution. These simplifications were most often made by hand, and though time consuming, at run time the geometry used could be selected based upon desired frame rates, desired amount of detail, or other factors [5]. As a result, this method has become known as *level-of-detail*. Because the number of available resolutions was finite and all computations were done beforehand, everything could be done quickly at run time, when the processor cycles were at a premium.

However, as the computational abilities of computers increased, it was not always necessary to simplify the geometry beforehand. In addition, it was desirable to have smoother transitions between levels of detail, but it also infeasible to create additional simplifications beforehand (due to time, money, or disk space constraints), and as a result, simplification methods that could operate at run-time became more desirable.

View-Dependent Simplification

Without a lot of specific information about an arbitrary dataset, it is difficult to determine what aspects of the geometry can be simplified. In addition, using level-of-detail approaches, discrete levels of resolution have to be created beforehand, and with particularly large datasets, this can create a number of problems. For example, with large, contiguous regions, it is possible that certain areas could be simplified, but others may be the area of focus, and full detail may be desirable. However, traditional methods of selecting a precomputed simplification of the geometry does not provide an adequate solution. With simplification methods that work at run-time, it is possible to gradually and increasingly simplify data as it becomes further away from the area of interest. This results in visualizations that can be quickly rendered while avoiding the problems with sudden changes in resolution that plagued the level-of-detail approaches. In addition, the view-dependent approach provides automated, adaptive and efficient simplification, all without the need of human intervention [2,12,15].

The most notable incarnation of view-dependent simplification is known as *hierarchical dynamic simplification* (HDS). This general mechanism of this method has become more or less the standard way of implementing view-dependent simplification, and as a result, view-dependent simplification has almost become synonymous with heirarchical dynamic simplification.

HDS operates by storing the vertices of the entire scene in a data structure known as a *vertex tree* [10,12]. The scene is simplified by collapsing multiple vertices into one representative vertex. As a result, many triangles become degenerate (in the form of lines or points), and as a result they can be removed. Conversely, if additional levels of detail are required, nodes in the tree will unfold creating additional triangles which will then be displayed [9,12].

This method works well in general because very few regions change at any particular instant whileinteracting with a scene. The small number of polygons that *do* need to be updated are called *boundary nodes* because they exist in the vertex tree between the active and inactive nodes. As the scene changes, nodes along the boundary unfold in areas that require additional detail, and collapse in areas where greater simplification is possible.

A similar approach to hieratchical dynamic simplification is known as progressive meshes. This method builds upon automatic triangle mesh simplification [8] by providing operation in a view-dependent fashion. The general concepts are, for the most part, the same as HDS—each vertex is stored in a tree—and an algorithm exists to determine which vertices should be simplified [4,7,11].

In order to provide a generalized framework for run-time simplification of polygons, implementations of view-dependent simplification techniques predefine the fundamental operations; however, it is possible to provide fast and efficient operation tailored to individual needs because it is possible to specify the method by which simplification decisions are made for when nodes are to fold and unfold. There are a few general methods that are generally used: a screen-space error threshold, a silhouette test, and a triangle budget [12].

Screen-Space Error Threshold: with view-dependent simplification, the goal is to reduce geometric complexity as much as possible without a visible reduction in quality. As a result, it makes sense to reduce the number of polygons in areas that do not occupy much screen space. This is the exact strategy of simplification that screen-space error threshold uses. Polygons that can be removed without causing a change in screen of more than a user-defined number of pixels are collapsed.

Silhouette Preservation: one of the ways that the human visual system is able to detect and recognize objects is through their borders and contours. As a result, if object edges can be detected and given a higher resolution than the rest of the object, the perceived amount of quality in the scene is increased. This method works nicely with the screen-space error threshold, allowing a higher degree of screen-space error in interior regions, while specifying a much lower tolerance for error in the border regions.

Triangle Budget Simplification: Whereas screen-space error threshold and silhouette preservation hope to maintain a level of visual quality, often it is the case where there is a limited amount of rendering resources, and it is more important to sacrifice quality for the sake of frame-rates. This can be achieved by specifying the maximum number of triangles that are desired. The geometry is reduced to the appropriate levels, and the algorithm does its best to minimize the error that is introduced.

III. Approach

The major similarity with the existing implementations of view-dependent simplification techniques is that they are designed primarily to simplify large, complex objects such as terrains [6] or *computer-aided design* (CAD) [3] models.

The idea of simplifying glyphs has been overlooked because glyphs are intended to be simple geometric shapes that represent data points, and as a result the glyphs themselves should not require simplification. However, as discussed previously, when large datasets are visualized using glyph-based methods, it is often the case that there are areas where large numbers of glyphs are clustered, and the overall number of polygons is very large [1,14]. Due to the fact that many glyphs are located in clusters, a great deal of polygons are located on interior surfaces, and serve no descriptive purpose while resulting in a lengthy rendering process.

View-dependent simplification methods are a perfect way of handling these situations, however, as they are designed to reduce polygons dramatically in areas where they are of less perceptual importance. Additionally, glyphs can be simplified considerably since the shape is of secondary importance, and it is the location which is informative. As a result, screen-space error can be considerably higher than in most other polygonal models, and large amounts of simplification should be possible.

Using view-dependent simplification techniques, there are a number of reasons that we should be able to achieve this goal. One of the side-effects of view-dependent simplification techniques is that they do not preserve topology; that is, it is possible that polygons that are not originally connected may become connected through the simplification process, as demonstrated in figure 1. This side-effect

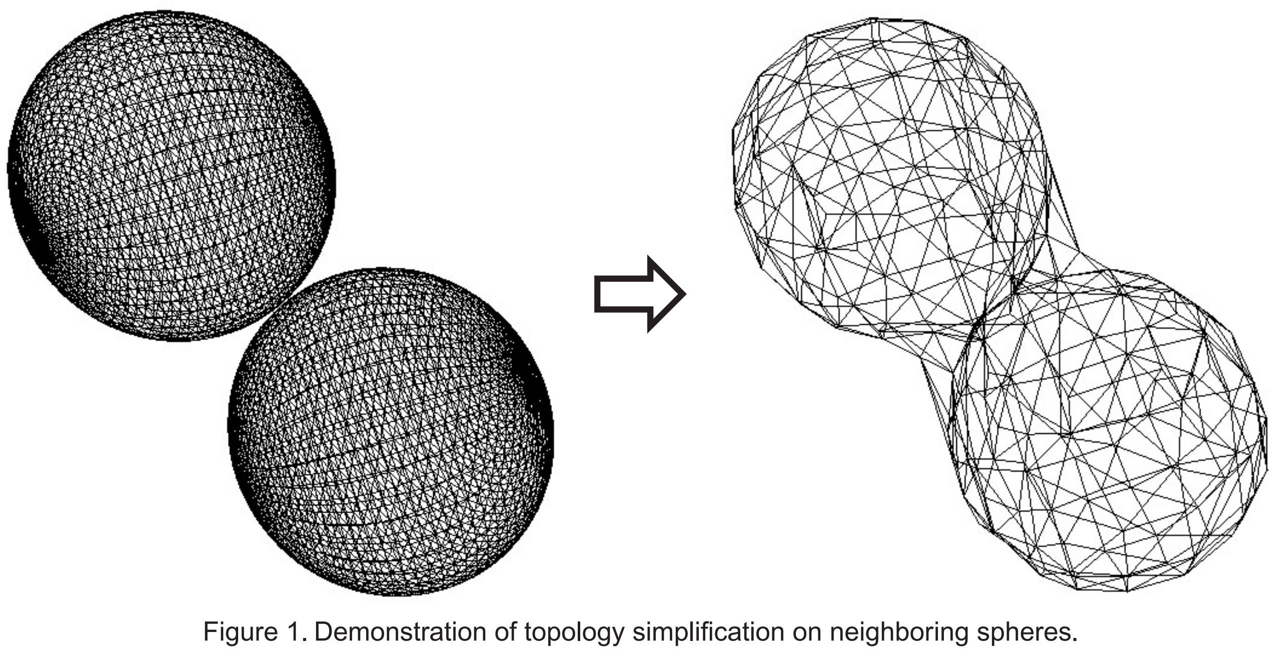
Figure 1. Demonstration of topology simplification on neighboring spheres.

is a valuable resource in this situation, as this is exactly the behavior that is desired. Clusters of glyphs are too dense to provide any meaningful detail; as a result creating a generalized representation of the cluster can convey the same information while greatly reducing the number of polygons that are necessary to render. In order to better understand what is going on here, the simplification of a cluster of glyphs can be thought of as placing the glyphs within a vacuum-sealed bag. The bag would form a rigid "skin" around the somewhat tightly contained glyphs, and as a result the skin would have the same shape as the original glyphs, which could now be thrown away. However, as the bag is only a "skin," there is no interior—the simplification has served to reduce the cluster to only the boundary polygons.

The problem that presents itself when dealing with view-dependent simplification of glyph-based visualizations is that in some cases it is possible to have the outlier glyphs be simplified away. Bad choices for the simplification algorithm can result in more triangles being reserved for areas of dense geometry, since these areas are considered more important. Using a budget simplification method, this was evident. The vertex tree has a single root node, and as a result it is not possible to simplify each cluster of glyphs independently. For the purposes of glyph-based visualizations, sillhouette-preservation algorithms are probably the best choice, for these attempt to simplify the scene as much as possible which still preserving the overall shape of objects, whereas budget-based methods are concerned with meeting a particular level of simplification, and not with preserving topology. Error-space metric algorithms would also produce reasonable results; however, as error has less meaning when considering that the original geometry has no real meaning, so specifying a particular error threshold would result in less simplification than is possible in some areas, and perhaps too much in areas that would be preferable to leave in high detail (such as individual outlier glyphs or silhouette edges.

IV. Results

Initial results from the experiments proved to be promising, however, due to time constraints as well as other factors, it was not possible to meet the level of projected progress. Early problems caused the development and testing schedule to be moved back and did not leave enough time for more detailed and rigorous tests. However, a simple budget-based simplification method was tested and demonstrated a number of promising results.

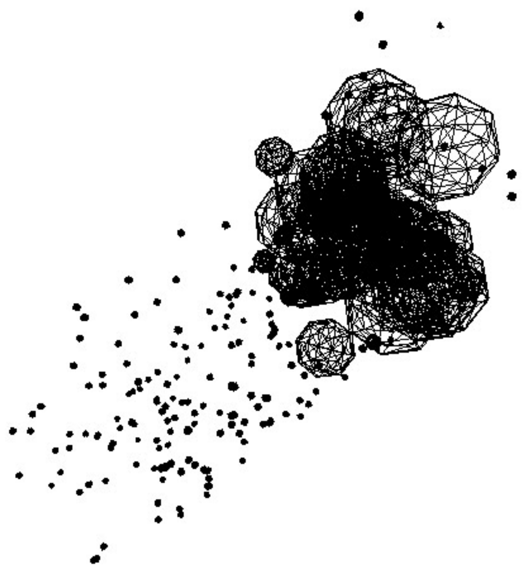
Figure 2. Original visualization using spheres (900,000 polygons).

Firstly, glyphs located in close proximity were joined when simplified, indeed to a point where two distinct spheres did simplify as a single unit over varying degrees of simplification (see figure 1).

Data that had larger amounts of polygons also showed very promising results. A generic dataset was used to create a glyph-based visualization of fair complexity. About 900,000 polygons were generated when using spheres (figure 2) for the glyphs, and about 45,000 when using cubes (figure 3).

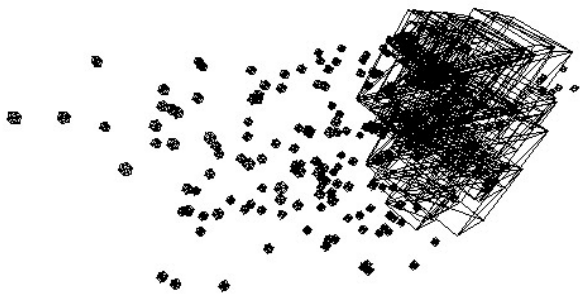
Figure 3. Original visualization using cubes (45,000 polygons).

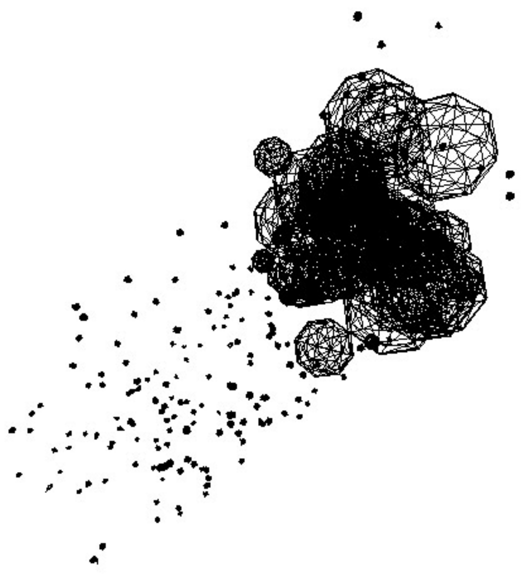

Figure 4. Simplified sphere visualization (10,000 polygons).

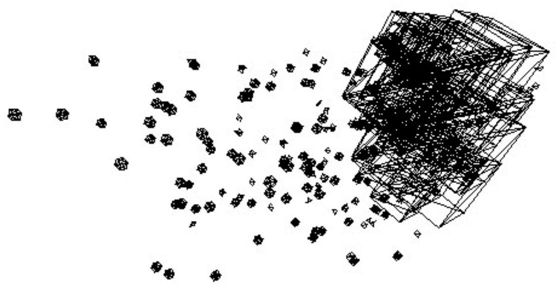

Figure 5. Simplified cube visualization (1,000 polygons).

The resulting data was simplified, using a simple visual metric to determine when a noticable difference in the visual quality became apparent. Using spheres (figure 4), the level of simplification was far more impressive than using cubes (figure 5), however, this was to be expected as cubes are a much simpler geometry, and any reduction of vertices results in a far more noticable change to the image. In addition, under these rather modest amounts of simplification, acceptable levels of detail were maintained while allowing for improved fluidity in navigation and interaction.

For these initial results, a simple budget-based simplification algorithm was used, and a number of limitations were found as a result. Some of these limitations were minor and to be expected, such as loss of detail along sillhouette edges, but others were much more major, such as lost outlier glyphs as well as areas that were not simplified as much as could theoretically be achieved. This generally occurred in interior regions (as can be expected from an algorithm that tries to do its best to maintain the shape of the original polygonal mesh while staying within its hard upper-limit polygon budget.

V. Future Directions

There are certainly a number of ways that this path could be further investigated. Certainly, an algorithm that simplifies using a silhouette preservation approach should be developed and used, as this should produce better results, both visually and in terms of simplification that should be achieved. More rigorous tests should also be derived, as current metrics for acceptable simplification are based entirely on non-scientific methods. Examples of possible metrics could include the requirement that all glyphs must not be simplified to a point in which they disappear, or a more complicatied screen-space variation tolerance between the initial and final image. Alternatively, a scalar value for each point could be used to determine the level of simplification for each glyph, and in this way, glyphs generated from points of lesser scalar values would appear less prominent in the final visualization.

There are a number of other directions that would also be desirable to persue in a longer-term. One very desirable step is to build an improved interface for interacting with the simplified visualizations, as currently only wireframe views are possible. Finally, an implementation of view-dependent simplification for the freely available visualization toolkit (vtk) would be desirable, as these techniqes could then be applied in a number of other ways and by a diverse number of users.

VI. Conclusions

View-Dependent Simplification methods proved to be helpful when used with glyph-based visualizations. Glyphs were simplified into generalized abstractions, allowing for greater understanding of the data as well as providing an increased level of interactivity and improved navigation. Glyph-based visualizations, while not traditionally the target of view-dependent simplification techniques, are the perfect candidates for this type of simplification. This is because glyphs are an abstraction to begin with, so manipulation of the individual glyphs does not reduce the understanding of the data and indeed has a number of benefits as well. While view-independent techniques would work to some extent as would preprocessing of the data, view-dependent simplification allows for the greatest degree of simplification while still providing the necessary information to make glyph-based visualizations of document data both useful, easier to interpret, and faster to render.

References

[1] Atkison, T. et al. "Case Study: Visualization and Information Retrieval Techniques for Network Intrusion Detection." VisSym '01 Joint Eurographics/IEEE TCCG Symposium on Visualization (2001).

[2] El-Sana, J. and Varshney, A. "Generalized View-Dependent Simplification." Proceedings of EUROGRAPHICS '99 (1999), C83-C94. Eurographics Association/Blackwell Publishers.

[3] Erikson, C. et al. "HLODs for Faster Display of Large Static and Dynamic Environments." Proceedings on 2001 Symposium on Interactive 3D graphics (2001), 111-120. ACM Press/ Addison-Wesley Publishing Co.

[4] Fei, G. and Wu, E. "A Real-Time Generation Algorithm for Progressive Meshes in Dynamic Environments." Proceedings of the ACM symposium on Virtual reality software and technology (1999), 178-179. ACM Press/Addison-Wesley Publishing Co.

[5] Funkhouser, T., and Séquin, C. "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. Proceedings of the 20th annual conference on computer graphics & interactive techniques (1993), 247–254. ACM Press/Addison-Wesley Publishing Co.

[6] Hoppe H. "Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering." Proceedings of IEEE Visualization '98 (1998), 35-42. IEEE Computer Society Press.

[7] Hoppe, H. "View-Dependent Refinement of Progressive Meshes." Proceedings of the 24th annual conference on computer graphics & interactive techniques (1997), 189-198, ACM Press/Addison-Wesley Publishing Co.

[8] Kobbelt, L. et al. "Interactive Multi-Resolution Modeling on Arbitrary Meshes." Proceedings of the 25th annual conference on computer Graphics (1998), 105-114. ACM Press/ Addison-Wesley Publishing Co.

[9] Lindstrom, P. and Turk, G. "Fast and Memory Efficient Polygonal Simplification." Proceedings of IEEE Visualization '98 (1998), 279-286. IEEE Computer Society Press.

[10] Luebke, D. "Hierarchical Structures for Dynamic Polygonal Simplication." Department of Computer Science, University of North Carolina at Chapel Hill, Technical Report #TR96-006 (1996).

[11] Luebke, D. "A Survey of Polygonal Simplification Algorithms." University of North Carolina at Chapel Hill Department of Computer Science Technical Report #TR97-045, (1997).

[12] Leubke, D. and Erikson, C. "View Dependent Simplification of Arbitrary Polygonal Environments." Proceedings of the 24th annual conference on computer graphics & interactive techniques, (1997) 199-208, ACM Press/Addison-Wesley Publishing Co.

[13] Rheingans, P. and desJardins, M. "Visualizing High-Dimensional Predictive Model Quality." Proceedings of IEEE Visualization '00, (2000) 493-496. IEEE Computer Society Press.

[14] Shaw, C. et al. "Interactive Volumetric Information Visualization for Document Corpus Management." International Journal on Digital Libraries 2(2/3), (1999) 144-156.

[15] Xia, J and Varshney, A. "Dynamic View-Dependent Simplification for Polygonal Models." Proceedings of IEEE Visualization '96, (1996) 327-334. IEEE Computer Society Press.