Visualization of continuous data using shape in glyph-based visualization

Pradyut Panda

Department of Computer Science,

University of Maryland, Baltimore County.

Advisor: Dr. David Ebert

15th July 1998

Abstract

This paper looks at the use of shape to represent or visualize a single dimension of continuously varying data in a glyph-based rendering system that is used for the visualization of multi-variate datasets. We use procedurally generated shapes to generate the shape of the glyph. Three algorithms were used on this paper to generate the shape of the glyph. The first method was the distortion of a basic shape such as a cube using a fractional Brownian motion type of turbulence function. At low values the distortion is minimal and the shape closely resemble it's basic shape whereas at high values the shape is highly distorted. However it is possible to see the underlying basic shape. The second technique to generate shape that we looked at, was the gradual warping from one pre-defined shape to another. This was achieved by performing a linear interpolation between corresponding points, while preserving polygon faces and normals. The third technique used was the refinement of a polygonal approximation of a smooth surface. Here the number of polygons is increased as the data value increases. Initial results show, that for a small number (three to four) of distinct shapes, the turbulated cube and the refinement of a circle allow very good discrimination of shape. A detailed study of the information space would allow up to ten distinct shapes. However, this approach to representing data is limited by the perceptual limitations of the human visual system.

1.0 Introduction

Glyph-based visualization is a powerful technique to visualize multi-variate data. A glyph is an icon or graphical object that has certain properties such as x, y, z location in three dimensional space, color, opacity, orientation, shape and size. These properties may be thought of as the visual dimensions of a glyph. Multiple values can then be encoded in the parameters of an icon. A single dimension of data is mapped to each of these visual dimensions of the glyph. In the current implementation, upto seven data dimensions may be mapped to these glyph attributes. An array of glyphs is created whose length is equal to the length of the data dimensions being mapped. This array of glyphs is then rendered to visualize the data. One such system that is based on glyph-based volume-rendering is the Stereoscopic Field Analyzer (SFA)[9]. The Stereoscopic Field Analyzer allows the visualization of three-dimensional volumetric data or multi-variate, time-varying data. In this paper, we look at ways to extend the use of shape to visualize data in the Stereoscopic Field Analyzer.

Previous work has been done in automatic generation of shape for continuous data using procedural techniques[1]. The primary approach to generating shape was to use superellipsoid surfaces. Superquadrics are used to represent continuously varying data by using the data value being mapped as an input to the function generating the superquadric. Superquadrics are an extension to simple quadratic surfaces, where the exponents for the trigonometric functions are different from one. This theoretically allows us to represent infinitely small variations in the data using shape. However, it is widely believed that it is difficult for our perceptual system to discriminate between very small variations in shape.

Work has also being done in using implict surfaces for the purpose of text visualization[8]. Implict surfaces are surfaces that are not modeled explicitly using polygons or spline patches or any other techniques but are implicitly defined by a function. Using implicit surfaces, document content was mapped to shape. A scheme was also defined to view a document corpus as a collection of shapes, thereby providing both local and global information.

The rest of this paper is organized as follows. In the next section, this paper talks about glyph-based visualization and the base system we have used in our work which is the Stereoscopic Field Analyzer. Section three contains the main concept discussed in this paper, which is the procedural generation of shape for the purpose of visualization of data. In section four, we discuss some of the implementation details of the system and section five follows with a discussion about the results. Lastly we talk about the future direction this research could take.

2.0 Glyph-based visualization and SFA

2.1 Glyphs and the representation of data

A glyph is a graphical object or icon which has certain attributes or properties. For three dimensional glyphs these attributes are its x, y, z location in three dimensional space, its color, opacity, size, shape and direction, if the glyph has any particular orientation. As mentioned before, these attributes may be thought of as the visual dimensions of the glyph. We may then map a single data dimension, that we wish to represent, to each of these visual dimensions of the glyph. The location of a glyph in three dimensional space probably represents the most easily understood of all the glyph attributes and perhaps conveys the most information. The color of a glyph which could be any interpolation scheme on some pre-defined colors, could represent importance of the data item with more important data being represented by brighter colors and less important data being represented by dull colors. Color may also be used in other ways to represent multiple dimensions of data. Similarly, an example of the use of the transparency of a glyph, could be to show the age of the information with older data having lower opacity values and thus fading into time. The size of the glyph can be used to stress importance with larger glyphs being more important. We would like to use shape to represent a single dimension of continuous data. To do this we would need to vary the shape smoothly as the value of the data changes. The orientation of a glyph is used when we need to represent vector data, which has both a location in three dimensional space and an orientation.

It is important to note that in our system we do not define any particular mapping of the data to the glyphs visual dimensions. This mapping process is dynamic and it is left completely to the user, to decide which data dimensions he or she wants to map to the which visual dimensions.

2.2 Two-handed minimally-immersive interaction

The system used in this paper (the Stereoscopic Field Analyzer) uses a two-handed interaction metaphor[9] and uses minimally-immersive virtual reality techniques. The two-handed interaction is achieved with the use of a pair of 3D magnetic trackers, each of which has three buttons. The user uses his non-dominant left hand to manipulate the viewing volume and uses his dominant right-hand to perform higher-frequency picking operations. The user uses hierarchical Sundial 3D menus to change rendering and other parameters. These menus have been incorporated into the two-handed interaction paradigm.

The user wears a pair of stereoscopic glasses for three dimensional vision with the flexibility to change stereoscopic parameters such as eye separation and convergence. Unlike head-mounted displays, stereoscopic glasses allows the wearer to view his normal work environment in addition to the virtual word which he can see. This kind of minimally-immersive virtual reality allows the user to remain in his normal working environment while viewing the virtual world. This has the advantage of not disorienting the user.

The Stereoscopic Field Analyzer has been used for the purpose of scientific and information visualization[5,6]. Its flexibility allows it to be used for many different kinds of visualization applications.

3.0 The shape of the glyph as a visual dimension

The primary use of shape is to assign the same shape to every glyph. For scalar data, this shape may be a point, cube, or sphere and for vector data, the shape maybe a line, cone or an arrow. Non-continuous or discrete data may be represented by having a different shape for each discrete interval or bucket of data. The shape is also dependent on the real-time rendering constraints of the system. As the shapes are being represented by polygonal model, shapes with more faces (and therefore more polygons) will take longer to render. Another point of interest is that glyphs that are far away will appear as a dot or point. It is pointless to render these glyphs in their full complexity. For these reasons, we are using a multi-resolution model which is covered in greater detail in the next section.

As mentioned before, the idea behind using shape to represent a single dimension of continuous data is to vary the shape depending on the data value of the data dimension that has been mapped to shape. Essentially, we need a procedure that takes a single data value as its input and generates a shape based on this value.

3.1 The turbulated cube

One way to generate procedural shapes that are dependent on a data value, is to generate some distorted version of a basic shape, such as a cube, where the amount of deviation from the basic shape is proportional to the data dimension being mapped. If data values are normalized between 0.0 and 1.0, this will result in a normal cube for data with value zero and a “fuzzy” cube for data with value closer to one.

The idea then, is to apply a turbulence type function and use displacement mapping to modify some base shape. The base shape may be thought of as the low frequency component of the overall shape and the distortion as the high frequency component of the overall shape. The relative frequency of the displacement maps is also important. Too low a frequency with a high amplitude will change the basic shape and too high a frequency will mask the change in amplitude of the displacement. The use of a very high frequency will also result in a large increase in the number of polygons, and thus an increase in the rendering time. The idea is that the user should be able to see the basic shape in the overall structure of the glyph.

As we are using a polygon model to represent the shape, we may use polygon sub-division along with displacement maps to create the new shapes. The actual turbulence function used is a Fractional brownian motion (FBM)[7] type of function. Fractional brownian motion is the motion exhibited by small particles when immersed in a gas or other fluid. This is an erratic, zigzag movement which is also a form of a random walk. Mathematically, it may be expressed as the integral of white noise. For more information on fractals, please see [2,3].This function is then used to generate a height field which is used to displace the points on the surface of the basic shape. To control the distortion on the basic shape, we could scale the height field according to the data value being mapped. However, this would be the same as if we were increasing the size of the glyph. Another possibility is to vary the fractal dimension of the underlying fractal model. For values of the data which are close to zero the fractal dimension of the overall surface could be set to 2. This will result in a plane, smooth surface. For data values closer to one (assuming normalized data), the fractal dimension could be set at 2.3. We could then use a linear interpolation or a cubic-spline interpolation between the two end values. Object complexity is important for any shape as practical constraints such as the rendering time may limit the number of polygons per object.

3.2 Refinement of a basic shape

Another idea in the use of shape, is the gradual refinement of a shape based on the data value being represented. In most polygon-based renderers, smooth shapes are approximated using polygons. An example of this is an n-sided polygon representing a circle. To generate a shape that depends on a data value, the number of polygons that are used to represent the shape may be controlled using the data value being represented. Low values of data would result in fewer polygons for the shape and higher values of data would result in more polygons. If we use our earlier example of a circle, the number of line segments defining the circle may be controlled using the value of the data with a minimum number being three. This is in turn will control polygon complexity. This can easily be extended to three dimension if we consider either three dimensional circles or a sphere. This paper used three dimensional circles which are essentially circles that have some depth component to them. With a small number of distinct shapes, the hope is that the user can simply count the number of polygons and from that derive the data value being represented. This could be done by counting the silhouette edges as the human visual system is good at edge detection. With a sphere, this task now becomes more challenging.

3.3 Morphing between basic shapes

The third technique presented in this paper for using shape to represent data is morphing between two distinct pre-defined shapes. This can be thought of as a linear interpolation between two shapes, with the two endpoint shapes being some fixed shape defined by the user. This is inherently the technique used in the Superquadrics paper[1]. Here we suggest using some of the platonic solids[4] and warping between them by defining a function that provides a mapping between the different vertices of the solid. As different solids have a different number of vertices, we can increase the number of vertices for each solid. This is done by setting the new vertices to the same location as some of the old ones. Thus, we can define each solid with the same number of vertices, and thereby maintain a one-to-one correspondence between vertices. This warping process must preserve the polygon faces and polygon normals for intermediate shapes. Simple examples of this technique are warping between a tetrahedron and a cube and warping between a octahedron and a cube.

4.0 Implementation details

The current implementation of SFA uses the OpenGL libraries to render the volume of glyphs. We are using OpenGL drawing lists due to efficiency considerations. A separate drawing list is created for each variation in shape for all of the above methods. The drawing list to be called is determined at run time; if any data dimension has been mapped to shape. In the current implementation of SFA sixteen bytes are being used to represent the data associated with a particular glyph. Out of this one byte is used to store the value of the data that is mapped to shape. This allows us to specify a maximum of 256 different shapes for a glyph. However, it is worth noting that it may be difficult for a viewer to discern the difference in shape for such a large number of shapes as the variation in shape will be very small. It was observed that for most practical purposes, the maximum number of different shapes should be kept at around ten.

4.1 Multi-resolution model

One of the key issues is the rendering time for each glyph. As the system is a real-time system, we cannot spend as much time as we would like to draw glyphs. One of the solutions to this is to use a multi-resolution model for each shape. The way that this is currently implemented is that based on the glyph screen size, there are two distinct intervals or buckets. If the glyph being drawn falls into the first bucket it means that it is sufficiently far enough from the eye to be rendered as a point on the screen. If the glyph is close by, then it is rendered as the full complex model. This helps speeds up the overall rendering.

5.0 Results

The Turbulated Cube
The turbulated cube was implemented using a summation of noise function[7]. The noise function used was the grid lattice noise function. The number of octaves of noise used were 4.0 and the lacunarity used was 2.0. The value of H, which controls the fractal dimension, was varied between 0.0 and 1.0. The number of distinct shapes used were ten. Initial experiments suggest that, it is possible to distinguish about five shapes. This method gave good results in cases where it was necessary to locate data with values close to the end points i.e. 0.0 and 1.0.

Refinement of a circle
This method produced good results with a small number of distinct shapes. This is because it is very easy to pick up circles with 3,4 or 5 sides. As the number of polygons increases, it is increasingly difficult to distinguish between two nearby shapes. The number of useful distinct shapes that can be used is five. One solution could be to increase the rate of change of the number of polygons as the number of polygons increases.

Warping between pre-defined shapes
In this method, we looked at three shape warping algorithms between different platonic solids. The first was from a tetrahedron to a cube. The second was from a tetrahedron to an octahedron and the third was from a octahedron to a cube. The number of distinct shapes used were ten. This method did not give very good results overall, as the two end value shapes did not have any relation to each other. The use of this method may be limited, especially when viewing multiple datasets.

6.0 Conclusion

This paper has explored the use of shape in glyph based visualization. Three distinct approaches were introduced to procedurally generate shapes. We introduced the turbulated cube, which is a turbulence type function applied to a regular cube. The turbulated cube seems to holds the most promise for being an effective way to use shape. To further asses the usefulness of this technique, a formal evaluation or a user study needs to be done. The refinement of a circle is a useful method to generate shape when the number of buckets of the data can be kept small. Warping between pre-defined shapes gives very good results when we are interested in end values of the data.

It is important to note that the use of shape is limited by the human perceptual system. We are able to discriminate among only a limited number of shapes. Another important issue is the rendering time of the shapes. As we create more complex shapes, the rendering time for each shape increases, thereby limiting the size of datasets we can view in real-time. Optimizations are needed to keep the rendering time as small as possible.

7.0 Future Work

Mapping two dimensions
One way to extend this idea is to use shape to represent two dimensions of data. The shape of the glyph can be thought of as having a low frequency component and a high frequency component. The first data dimension could be used to map into a function that specifies a basic shape or the low frequency component of the overall shape. The basic shape could be a set of discretely specified shapes or a set of procedurally generated shapes such as superquadrics. The second data dimension would map to the amount of deviation from that basic shape or the distortion on the basic shape. This is on the same lines as the turbulated cube. Care will have to be taken that the amount of deviation is not so much as to hide the basic shape.

Extending the multi-resolution model
The multi-resolution model may be extended by added a third level of detail. For the third bucket we draw the shape that has the minimum number of polygons out of all the possible different shapes or reduce the number of polygons for every shape.
For the turbulated cube we could render the cube with a lower frequency component displacement map. This will reduce the number of polygons per cube. For the refinement of a circle, we could use a less refined circle if the glyph is sufficiently far away. It is harder to add this third level for the warping of shapes algorithm as it is hard to reduce the number of polygons without changing the basic shape.

Application to color
It may also be possible to apply the concept of turbulence to the color of the glyph. The idea is to use a solid texture to color the glyph. This texture would be procedurally generated by a function that depends upon some input parameter. This input parameter can be supplied by the data dimension which we want to visualize. As the data value being mapped increases, the distortion applied to some standard texture also increases. An example of this is a vertical or horizontal stripes texture. As the data value increases, the stripes become more wavy or distorted. Due to the real-time rendering constraints, the textures could be pre-computed and stored in an array of texture maps.

8.0 Acknowledgments

I would like to thank Dr. David Ebert for his helpful comments and guidance. I would also like to thank my wife Adite and my sister Pankaja for their support.

9.0 References

[1] Automatic Shape Interpolation for Glyph-based Information Visualization - David S. Ebert, James M. Kukla, Christopher D. Shaw, Amen Zwa, Ian Soboroff, and D. Aaron Roberts, Late Breaking Hot Topics Proceedings, IEEE Visualization Conference, Oct 1997.

[2] Chaos and Fractals: new frontiers of science - Peitgen H., Jurgens H. and Saupe D., Springer-Verlag, New York, 1992.

[3] Fractals and Chaos - Crilly A. J. et al, Springer-Verlag, New York, 1991.

[4] Jim Blinn's Corner - Jim Blinn, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

[5] Minimally-immersive Interactive Volumetric Information Visualization. - David Ebert, Christopher Shaw, Amen Zwa, Ethan Miller, and Aaron Roberts, Proceedings of Information Visualization Symposium 1996, IEEE Computer Society Press, Los Alamitos, CA, 1996.

[6] Multiresolution Text Analysis. - Zwa, A., Ebert, D., and Miller, E., Workshop on New Paradigms in Information Visualization and Manipulation, ACM CIKM 1996 Conference.

[7] Texturing and Modeling: A Procedural Approach - David Ebert et al, A P Professional, Florida, 1994.

[8] The shape of Shakespeare: Visualizing Text using Implicit Surfaces - Randall M. Roher, David S. Ebert, and John L. Sibert, Proceedings, IEEE Visualization Conference, 1998.

[9] Two-handed Interactive Stereoscopic Visualization - David Ebert, Christopher Shaw, Amen Zwa, Ethan Miller, and Aaron Roberts, Proceedings, IEEE Visualization Conference, 1996.