

Animated Texture Alpha-Masks for Flow Visualization

Ian Curington

Advanced Visual Systems Ltd.

Hanworth Lane, Chertsey, Surrey KT16 9JX, U.K.

ianc@avs.com

Abstract

A method of using a moving texture alpha-mask to represent scientific data is described, for the purpose of visualizing continuous fluid dynamics fields. The method combines stream tubes and particle animation into one hybrid technique, and employs texture surface display to represent time, flow velocity and 3D flow structure in one view. The technique exploits texture graphics systems in common use for games, and achieves high graphics efficiency during animation.

Key words: interpolation, fluid dynamics, color, flow visualization, texture, animation.

1. Introduction

Computer graphics techniques have long been used as an investigative tool for computational fluid dynamics (CFD). Many techniques are in common use, including vector arrow plots, contours, isosurfaces, cut planes, streamlines and particle animation. The main problem with 3D flow visualization is the comprehension of the spatial structures of a flow field.

With particle tracing techniques, particles are released at a certain position and followed as they flow, using animation to show the particle motion. This provides good insight into the magnitude of the flow velocity in the flow field, but less insight into the precise direction and position of the particles due to 3D ambiguities on a 2D screen.

An alternative approach is to show the particle trajectory paths instead of the actual particles using streamlines. Rotational aspects of the flow are visualized using a ribbon, where the surface orientation of the ribbon is controlled using flow field vorticity [Curington 91]. A related method uses stream tube structures instead of ribbons, where local velocity variations are mapped to tube

radius. Using ribbons or tubes instead of lines gives good three dimensional insight into rotational aspects of the flow and the path position as the human visual system is good at 3D shape perception. However, much of the velocity information is lost using this representation.

The approach described here combines the particle tracing technique with the geometric stream ribbon and tube technique including the advantages of both. The spatial information is shown using geometry, while velocity variations over time are shown using an animated texture with variable transparency. The color used in the texture map is a direct replacement to scalar color assignment, yielding high quality color contours and avoids artifacts introduced by RGB interpolation during graphics display [Curington 99].

2. Related Systems

2.1 The explosive growth of home PC games and game oriented PC graphics technology has contributed to the development of a large number of 3D graphics software engines. At least 15 different graphics engines used for games support full texture mapping of 3D polygonal objects. "QUAKE" is typical of many of these. In the QUAKE engine polygons are texture mapped using a hybrid mixture of ray tracing and Z-buffer methods. The "wobble" effect used to animate moving lava pits or water is created by applying a sine wave offset from a look-up table, changing the texture x-y offset [Isokovic 97].

2.2 Advanced animation software used for film, TV and special effects (Alias/Wavefront, SoftImage, Renderman etc.) also use texture mapping. Using texture maps on simple polygons than by modeling objects directly with geometric primitives can create much more complex scenes. Allowing a special color or "alpha mask" to control object transparency is used to see through to the

background on arbitrarily shaped regions of the object using the texture map method [Wolfe 99].

2.3 Such transparency mapping is used to lay transparent or semi-transparent objects over a scene by representing transparency values in the texture image as well as color values. This technique is used in flight simulators, and shown to be an effective method to simulate atmospheric clouds [Gardner 85]. Clusters of ellipsoids are draped with variable transparent textures to create effective cloud patterns. Texture map filtering applied to transparency and color values automatically leads to soft boundaries between the clouds and the background.



Figure 1 Synthetic Cloud models by Gardiner, using textured ellipsoids. Overlapping ellipsoids create complex shapes, while the texture yields fuzzy boundaries.

2.4 A common way to distributed 3D geometric models on the internet is with VRML, with viewers available as web browser plug-ins. The VRML specification allows objects to contain references to texture sources, including dynamic textures [Web3D 97]. The VRML texture interface is designed to allow live video feeds to update textures. It has proven more difficult to closely couple such scenes or texture vertex assignment to dynamic content creation system such as flow visualization tools in such a way as to achieve interactive animation.

2.5 The texture mapping approach described here uses methods developed to improve graphics quality for scalar data and avoid artifacts introduced by more typical graphics display systems. The method assigns texture coordinates

based on a mapping to the numerical scalar field values, such that RGB interpolation is avoided, interpolation is done in normalized floating point ranges rather than byte ranges, and greater resolution control is available for contour colors [Curington 99].

2.6 The method described here is related to a published method using “surface particles” [Stolk & van Wijk 91]. Many thousands of individual particles are integrated through the flow field, where each particle is a point position with a surface normal and a texture assigned. The texture is used to create interval structures and flow animation. The flow integration, normal calculations, shading and display were coupled in such a way as to make image generation non-interactive. This technique is currently used in a commercial product from Flomerics Ltd.

3. Animated Textures

3.1 At the heart of both streamlines and particle animation for flow fields is point path integration. The streamlines technique, for instance, generates a Polyline path in 3D space based on a continuous 3D gridded or unstructured mesh containing velocity vector data at each point. A set of probe positions is established and field values are sampled at these points by tri-linear interpolation. The small scale movement of these positions are computed using 4th order Runge-Kutta integration of a mass-less particle under influence of the local velocity field. Within each computational cell, the integration step size is adaptively modified to achieve best numerical results.

3.2 The resulting positions are then linked together to form Polylines, with locally sampled attribute information along the lines added to include velocity. Time is resolved along the path by further processing – a moving window function averages path velocity and converts this to time values sampled at each path. At each position along the path, the value of time is computed as:

$$dt = \text{segment\_length} / \text{segment\_velocity} \\ t_n = t_{n-1} + dt$$

3.3 Hence, the Polyline geometry now contains time as an independent scalar value array. Extending this concept to stream ribbons or stream tubes allows shaded 3D paths to be displayed with time as an attribute variable. To create particle animation displays from this data is straight forward – by processing stream tubes using iso-line

contours, time values are displayed as small rings at the point where the tube is sliced at that particular time. By animating the contour thresholds, the rings follow the path of the tube through the flow field, producing particle movement animation. This method can be called “iso-time” contouring.

4. Time to Texture

4.1 Typical texture map system assign texture to geometric shapes using a canonical 2D image coordinate space using $u-v$ texture coordinates. The mapping between $u-v$ and the models xyz coordinates is determined by projection, such that the texture image appears to be glued to the surface or applied like wall-paper [Haeberli & Segal 93]. For our flow visualization application, we completely ignore the xyz model coordinates and assign $u-v$ texture coordinates directly from the *time* variable. The longest time of any Polyline is used to normalize the texture coordinates so they remain in the interval $(0.0 - 1.0)$. However, note that time is one-dimensional, while texture space is two-dimensional.

A solution to this situation is to use a one dimensional texture mapping technique, with time assigned to the diagonal cross-section of the image.

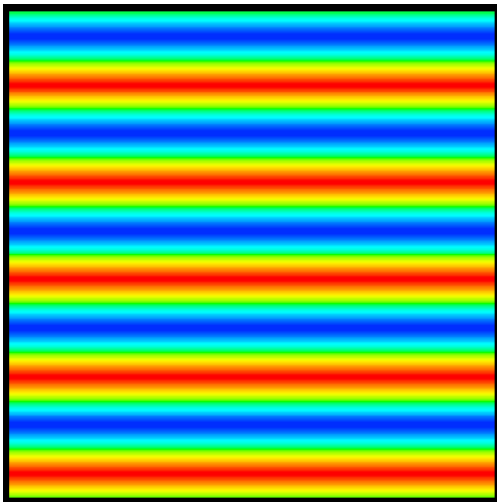


Figure 3 Vertical Texture Wave.
A 1-dimensional Cosine pulse function is placed in the image alpha channel.

Since many texture display systems need a square image structure, a 1-dimensional color-map is placed down the diagonal of the image. A color map pattern is easily sketched using a paint program to create a color map image. As so many image-handling tools exist, our prototype uses standard TIFF images. Interactive performance of this technique has been verified on systems ranging from a small laptop PC to a Silicon Graphics immersive virtual reality environment.

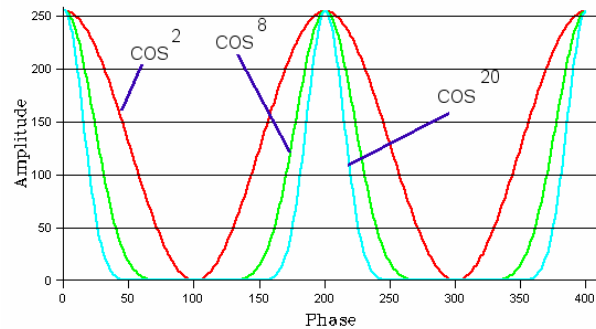


Figure 2 Pulse Waveforms. Pulses are formed by raising a cosine function to an integer power. Cosine squared yields a smooth wide pulse, whereas higher powers create shorter pulses which preserve continuity and soft edges.

5. Time-Wave Pulse Generation

In previous work, the $u-v$ texture coordinates had an offset applied, creating an animation to make the texture crawl along the stream path [Curington 91]. Here, the $u-v$ texture coordinates remain static, so the color and flow variables remain referenced to each other for analytical flow visualization, while the content of the texture alpha-mask is dynamically changed to produce an animation. In this way a continuous periodic function can be applied, rather than a direct positional shift. The center of highest opacity in the alpha-mask is used to create a time-wave pulse function. As phase is adjusted, the positions of the pulse function peaks move along the time path.

5.2 The time-wave pulse generator function is designed to generate smooth edges to particles, symmetry achieve a uniform representation of the

flow field, and periodic such that the animation can loop in a continuous cycle. The pulse generator is based on:

$$\alpha(t) = \cos^{\beta}(t+\phi)$$

where α is the alpha-mask value, β is between 2 and 20, and ϕ is the animated phase variable. Raising β to a higher value creates shorter pulses, while retaining a soft edge, while lower values create larger and wider pulses.

5.3 Most texture applications use a true-color image with Red-Green-Blue triplet values for each pixel. Advanced display systems also support a fourth value to control foreground-background mixing, or alpha-mask. This can be used to create stencils, masks, or variable transparency within the texture image. Hardware support for alpha-blending is often available to texture applications, so high-speed display is possible. The Time-Wave Pulse Generator described here is implemented as a filter that replaces the alpha-mask data in an input image by a synthesized pulse function. In this way hardware acceleration is directly exploited by the technique.

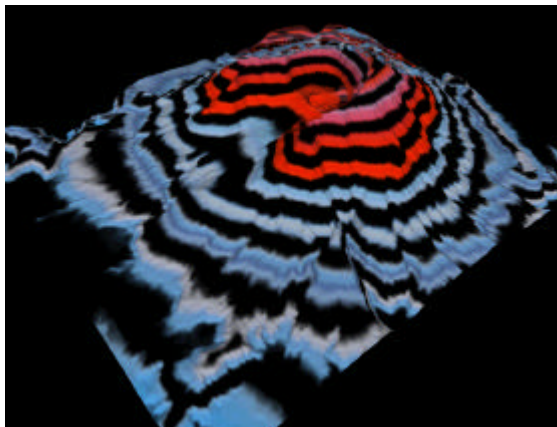


Figure 5 Height Field Contours on Mt. St. Helens using Texture Waves

6. Applications of Animated Texture Alpha-Masks

6.1 This texture-based technique has been applied to terrain data, where relative slope is highlighted using animation. By mapping the texture display to geographic height rather than time, moving contour lines of constant height are made to crawl over the

terrain. Where the slope is high, near mountain peaks, the animation progresses more slowly, while over large flat areas the animation progresses more quickly. Since color and u - v offset are not changed, color and height relationships remain unchanged. The data represents a regular gridded height field over Mt. St. Helens in Washington State, USA. The height field clearly shows the volcanic crater near the top and the large valley caused by the eruption in 1980.

6.2 The second application shows a complex 3D-flow field within a simulated room. Within the room a heat source creates convective air movement. The airflow recirculates and moves through the room from the table, up and across the ceiling, then back down along the far right-hand wall. Over 150 streamlines are displayed, with variable color and transparency using the technique described. The principle effect of particle movement is shown in an animation of this scene. The CFD solver used for this data is Phoenix, and the data provided by Adam-Net Ltd., Japan.

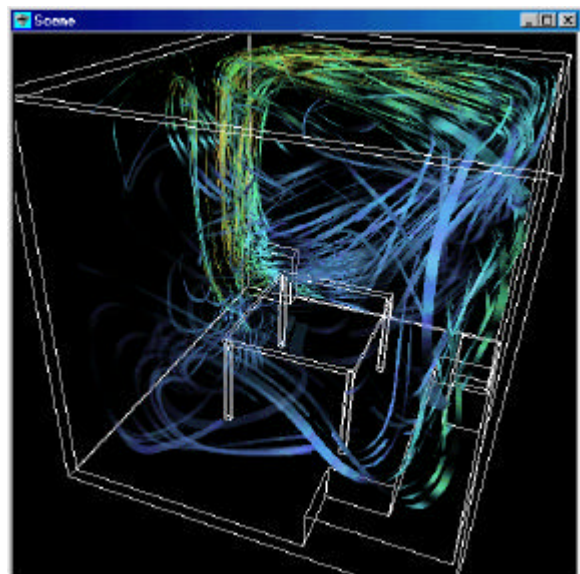


Figure 4 Convection Heat Flow in Room using Texture Wave Ribbons

7. Implementation

The development of this technique used the visualization development system AVS/Express from Advanced Visual Systems Inc. [AVS 96]. The high level macros were constructed using the visual network editor. The principle development was made in the "stream time resolver" and the "texture

wave generator” blocks shown in the visual network. The techniques developed for this paper were written in C++, using the AVS/Express code development system. The applications of this technique have been used under Windows-98 and SGI Irix. The graphics display system used during development was OpenGL based. The animation was interactive, so parameters could be adjusted to give the best effect, then saved as MPEG movie files.

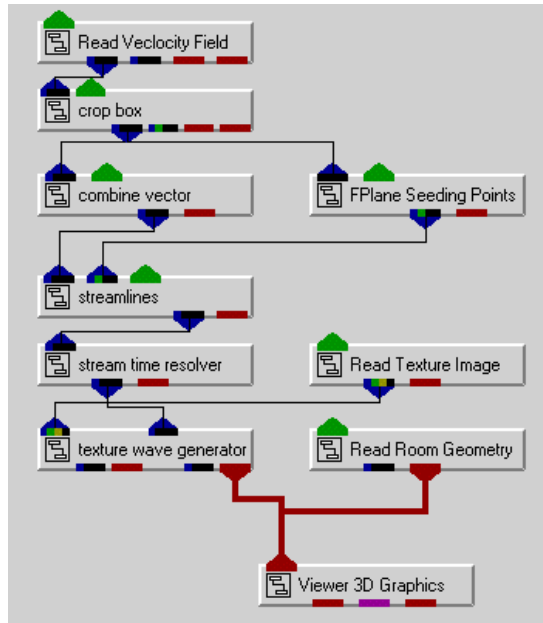


Figure 6 Implementation Architecture using Visual Program

8. Conclusion

A technique is described to reduce visual artifacts due to color interpolation using a texture map display technique. The technique exploits new texture mapping graphics hardware in widespread use for the games industry. The technique improves insight into complex fluid flow fields by combining both stream ribbons / tubes technique with particle animation. A unique time-wave pulse generator is used to create soft-edge particle visualization. Efficiency is gained by only changing the alpha-mask channel of the texture during animation over a static geometric structure. The software developed for the research presented in this paper is available at the International AVS Center site, <http://www.iavsc.org>.

9. References

- [1] Curington, I., "Visualization of Fluid Flow Data" Proceedings of Computer Graphics 91, Alexandra Palace, London, Online Publications, November 1991.
- [2] Curington, I. "Continuous Field Visualization with Multi-Resolution Textures" Proceedings of IEEE IV99, Information Visualization conference, London, July, 1999
- [3] "Data Visualization Kit" AVS/Express Developer Edition Reference Manual, Advanced Visual Systems Inc., Waltham Mass., June, 1996 <http://www.avs.com>
- [4] Gardner, G. "Visual Simulation of Clouds" Computer Graphics (SIGGRAPH '85 Proceedings), Pages 297-303, July, 1985
- [5] Haeberli and Segal "Texture Mapping as a Fundamental Drawing Primitive" Fourth Eurographics Workshop on Rendering, Cohen, Puech, Sillion Editors, Paris, France, June, 1993. <http://www.sgi.com/graphics/texmap/index.html>
- [6] Isokovic, K. "Commercial 3D Graphic Game Engines" technical report, TU Berlin, Germany, May 1997 (web only), http://cg.cs.tu-berlin.de/~ki/game_eng.html
- [7] Stolk and van Wijk, "Surface-Particles for 3D Flow Visualization" Second Eurographics Workshop on Visualization in Scientific Computing, Post and Hin Editors, Delft, Netherlands, April, 1991
- [8] "VRML 2.0 Specification" Web 3D Consortium, <http://www.web3d.org> 1997
- [9] Wolfe, R. "Teaching Texture Mapping Visually" Hypermedia and Visualization Laboratory web resources, Georgia State University and SIGGRAPH, May 1999 http://www.education.siggraph.org/materials/HyperGraph/mapping/r_wolfe/r_wolf_mapping_1.htm