Animated Illuminated Lines 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 dynamic texture map on 3D polylines to represent field streamlines is described, for the purpose of visualizing continuous fluid dynamics fields. The method combines stream lines and particle animation into one hybrid technique, and employs texture display on lines to represent full 3D lighting, motion 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: illuminated lines, 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. The ability to view a real-time animated display of surface as well as flow-field information allows the investigator to rapidly pinpoint key areas of study.

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 because the human visual system excels at 3D shape perception [Curington 2000]. However, only a small number of ribbons or tubes can be used before the display becomes cluttered. If large numbers of streamlines are used, the bundles of lines saturate the display such that perception of line shape and spatial position are lost.

This system takes advantage of graphics facilities to perform illuminated shading for lines. This is useful for accurately visualizing the 3D structure of vector field data, especially for large volumes of data. The speed afforded by hardware acceleration allows interactive visualization of large vector field data sets. The results are quite spectacular and provide dramatically more visual cues to the structure of the 3D field compared to normal lines.

Normally there is no such concept of shading for lines, which are after all infinitesimally thin objects. However, it is possible to give sensible meaning to this idea by considering the limit of a shaded cylinder as its radius goes to zero. Such operations are not supported in most standard graphics interfaces. It is possible, however, to achieve illumination of lines by using texture.

The basic technique relies on using the texture map capabilities of OpenGL graphics drivers to simulate illumination by light sources. This is done by generating an appropriate texture image and assigning the texture coordinates for each line vertex. Using the texture transformation matrix, one can employ OpenGL to evaluate the appropriate inner products and use the results to index into the texture. The result is a color returned from the textures that corresponds to the proper lighting illumination effects from an advanced shading model. The illumination is calculated in two parts: first the shading model is used to

construct the texture map ahead of time, then during display the lines are drawn with this texture map, where each element of the algorithm is implemented as a separate module in the visual development environment.

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 line geometry, while velocity variations over time are shown using an animated texture with variable transparency. The color used in the texture map directly replaces scalar color assignment, yielding high-quality color contours and avoiding artifacts introduced by RGB interpolation during graphics display [Curington 1999]. Most importantly, advanced shading models can be encoded into the texture for 3D illumination effects to reveal line curvature and spatial placement.

2. Related Systems

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 these game engines, polygons are texture-mapped using a hybrid mixture of ray tracing and Z-buffer methods. The "wobble" effect used to animate bubbling lava pits or waves in water is created by applying a sine wave offset index into a look-up table, changing the texture *u-v* offset [Isokovic 97]. The OpenGL texture display facilities are now widely available in most systems and achieve high performance. The illuminated line system uses the OpenGL API because of both raw drawing speed and versatility in object construction.

The method described here is related to a published method using "surface particles" [Stolk & van Wijk 1991], where many thousands of individual particles are integrated through the flow field, in which 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, involving a specialized research renderer.

The illuminated field line method was first introduced [Zöckler 1996, Stalling 1997] as a new and novel way to represent 3D vector fields, winning awards and reported as a general technique on a wide range of application areas. The method was implemented as subclasses to the OpenInventor software system.

Schussman recently described an application and extension of the original technique to complex toroidal geometries involved in the magnetic containment field of the Tokamak fusion experiments [Schussman 2000]. This implementation added the key concept of non-photorealistic rendering methods such as the halo effect to improve line depth ordering perception, although details of the algorithm were not described.

3. Animated Textures

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 movements of these positions are computed using 4^{th} 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.

The resulting positions are then linked together to form Polylines. Time is resolved along the path by a moving parameter of Euclidean path distance. To create particle animation displays from this data is straightforward – streamline processing causes time values to be displayed as pulses of texture attributes moving down the lines. The resulting graphics effect yields particle movement animation.

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 occurs in normalized floating-point ranges rather than byte ranges, such that the user gains greater resolution control over contour colors [Curington 1999]. By encoding a bi-directional reflection distribution

function (BRDF) into the texture, more advanced shading models can be used than normally available through OpenGL.

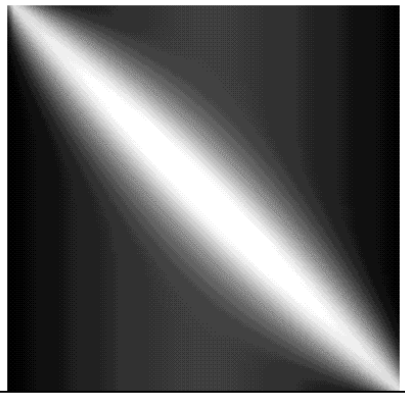

Figure 1. Texture encoded shading model. Angle variations for diffuse reflection use the left-right direction in this texture map, while specular highlights are controlled by mapping eye-reflection vector difference into the vertical image direction.

4. Texture Encoded Shading Model

Typical texture map systems assign texture to geometric shapes using a canonical 2D image coordinate space using *u-v* texture coordinates. The mapping between *u-v* and the model's *xyz* coordinates is determined by projection, such that the texture image appears to be glued to the surface or applied like wallpaper [Haeberli & Segal 93]. For our flow visualization application, we completely ignore the *xyz* model coordinates and assign *u-v* texture coordinates directly from the illumination model. However, note that lines are one-dimensional, while texture space is two-dimensional. This allows encoding of a full BRDF.

The illuminated line method uses a full Phong 3D shading model. Ambient light is added as a background constant to all shades. Diffuse illumination is computed by the cosine of the angle between the synthetic surface normal of the line and the light source. Specular shading takes into account the relationships of both the light source angle and the viewer angle differences from the surface normal vector. The shininess is varied from broad to tight highlight shape using a power of the cosine of the angle, as in classical Phong shading models. A further refinement is the Phong modifier term, which also applies an exponent term to the diffuse shading angle term.

The resulting shading function is encoded into a 2D texture map. The default texture map size is 64 x 64 pixels, but can be changed in the module's parameter block [Figure 1]. Because these values work in most circumstances, no user interface is required. Because the texture map is used to encode shading, this can be considered a piece-wise linear approximation to the full BRDF, with the piece-wise segments representing approximately 3 degrees of shading angle. The texture UV coordinate lookup then performs bi-linear interpolation from the texture within each angular segment.

5. Depth Ordering Perceptual Enhancement using Haloes

To further enhance spatial perception of field lines, a Halo model is utilized. This is a synthetic artistic presentation effect and has no basis in physical representation. The Halo mimics hand-sketched drawings in that a line breaks as it passes underneath a line that should be on top, such as in electrical circuit diagrams. When the lines and their halos are rendered, each halo obstructs the view of any line passing behind them. Simple implementations of line halo effect against a black background are made by rendering black, thicker lines first, then the true lines on top, in a depth sorted order [Schussman 2000]. The problem of this simplistic approach is that the surrounding halos are solid black and will not work if the background is any other color. The method described in this paper uses a special invisible halo

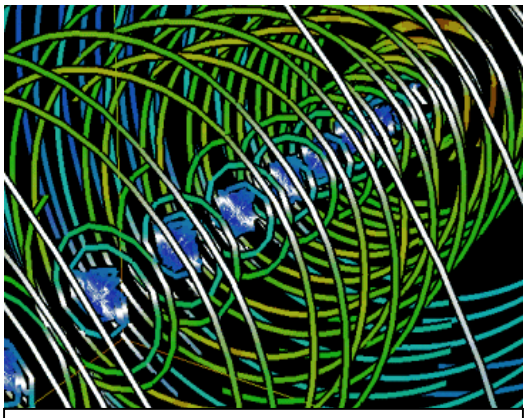

Figure 0. Shaded lines and haloes. Dark halo regions on each side of the lines help the perception of depth ordering. The background color shows through the halo region.

technique, where the background may show through the halo space around the illuminated lines. With this technique, an influence mask surrounds each line so that the depth ordering of lines becomes more visually pronounced. The width of the halo can be controlled for best visual effect [Figure 2]; a halo width of zero disables this effect.

The OpenGL attribute state mechanism is used to set up the halo rendering operation with glDepthMask() options. Normally, updates are made to both the Zbuffer and image buffer during rendering; in our case this assumption can be turned off during the halo object display update.

6. Line Transparency

The shaded field lines provide dramatically more visual cues to the structure of the field as compared to conventional solid lines. In addition, line transparency shading allows greater insight to the field structure in the interior of the field. Transparency results can be misleading since polyline rendering is not performed in a depth-sorted order. To minimize rendering artifacts, a Depth Mask toggle is provided. Active Depth Mask can also provide better results when anti-aliasing has been selected. The artifacts are normally only present at some line crossings and are normally barely visible.

7. Animated Pulse Generation Effects

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]. In the animated texture alpha-mask system, 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 [Curington 2000]. However, in the illuminated line method, the geometry and the texture remain static (pre-computed and cached) while the *u-v* texture coordinates or other line attributes are dynamically changed.

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. Animation effects are also produced using variable line width attributes.

This method has been equipped with simple animation effects, including the polyline width and transparency modulation along the line. It is not possible to change the width within a single segment of a polyline, thus both effects work per segment basis - this is the unit of measure of the animation effect using linear interpolation within each unit. The line region that an effect is defined over is made out of two parts - the first is the active part followed by filling (spacing) which delimits the active zones. Both zone's lengths are user settable by means of sliders (Length and Spacing). The active zone of an effect can be of type 'stair' or 'dash.' For the dash effect, the active zone makes the polyline segments that it modifies either of width 1 or totally transparent (depending on selected modulation). Following spacing segments are left with original parameters.

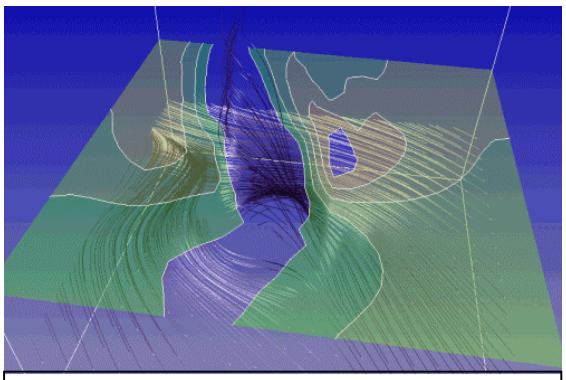

Figure 3. Wind velocity from weather model shown with illuminated streamlines.

The stair effect has three modes — up, down or triangle wave — that define stair steps with linearly increasing, decreasing or both 'steps.' This creates an illusion of a moving pulse of increasing transparency or width . Because the effects are applied per polyline segment, their rendering has to be done as segments instead of per polyline. This of course introduces a performance penalty compared to display speed without animation effects. The animation effect is computed using simple Euclidean distance along the line. The animation effects can be usefully applied to create visualizations even without creating animation.

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. Animation effects described here are implemented as a filter that replaces the alpha-mask data in an input image by a synthesized pulse function. In this way OpenGL hardware acceleration is directly exploited by the technique.

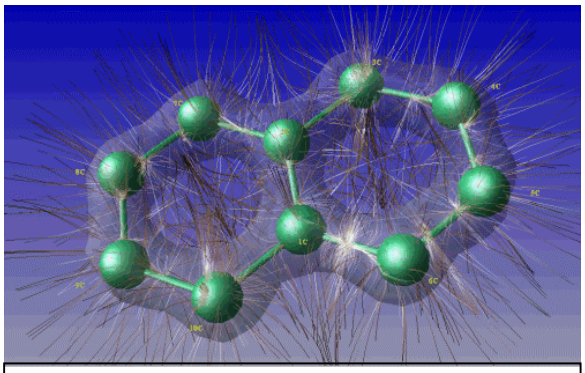

Figure 4. Naphthalene charge field. Illuminated lines show electron charge directions between atoms.

8. Application Results of the Illuminated Lines Method

IlineSpiralHalo [Figure 2]. This example shows a zoomed-in view of the spiral streamline visualization, with a nested set of cylindrical stream line paths. The line color is blended with velocity magnitude and texture mapped shading attributes. The halo effect can be seen by the dark edges of the lines, making the depth order in the perspective scene stand out clearly.

IlineWind [Figure 3]. This example shows how the illuminated line module can be used to show upward wind velocity in a small weather data set. As the view is rotated or the light direction is rotated, dramatic changes of light source shading move across the line clusters. The scene also contains a transparent contour plane, and the shading of the lines can be seen through the transparent areas.

IlineGauss [Figure 4]. The Naphthalene molecule computation from Gaussian "CUBE" field total density results are used as the gradient integration field for streamlines. Seeding points of the illuminated streamlines are placed at the nodes of several charge density isosurfaces, one of which is shown as a transparent shell. The animation effects

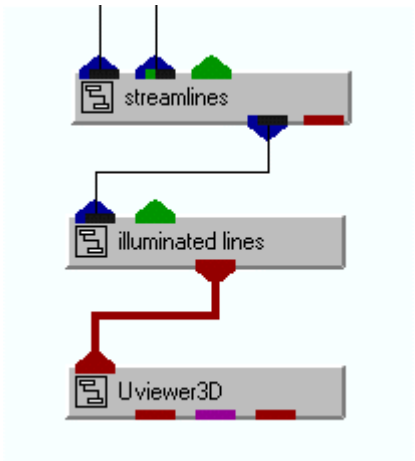

Figure 5. High level macro

are set to modulate the transparency along each streamline.

9. Implementation

The development of this technique used the visualization development environment (VDE) 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 module blocks shown in the visual network. The techniques developed for this paper were written in C++, using the AVS/Express visualization development system. The applications of this technique have been used under Microsoft Windows and various UNIX systems. The graphics display system is OpenGL-based. The animation is interactive, so parameters could be adjusted to give the best effect.

For typical users, all of the functionality of the illuminated line method is encapsulated inside one macro module, including all user interface

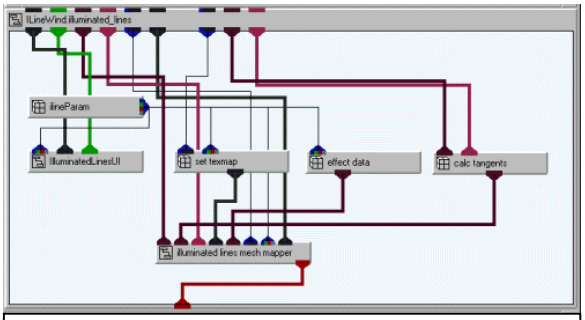

Figure 6. Expanded macro implementation modules.

components [Figure 5]. This macro can simply be placed in the visual programming network as a filter between streamlines and a 3D viewer. Inside this macro module, the method is implemented as a series of functional modules. The data preparation phase modules, such as texture illumination map, tangent vector calculation and animation effects, are each implemented as individual visual modules [Figure 6]. The data structures from these are made available to the display call-back system, which updates the display using the pre-computed data structures using a customized *render()* method inside AVS/Express. For immersive parallel graphics pipelines, such as for multi-pipe / multi-channel display systems like the CAVE™, the pre-computed data structures are passed to each renderer thread, which draws a portion of the scene with the localized coordinate view settings.

10. Conclusion

The display of illuminated field lines using texture map encoded shading model on polylines with line halo and animation effects has been described, where high-performance OpenGL based graphics features are used. The method is implemented in a modular visual development environment (VDE) and yields stunning effects for the visual perception of large numbers of 3D lines.

11. 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] Curington, I. "Animated Texture Alpha-Masks for Flow Visualization" Proceedings of IEEE IV2000, Information Visualization Conference, London, published by IEEE Computer Society, July 2000

[4] AVS - "Data Visualization Kit" AVS/Express Developer Edition Reference Manual, Advanced Visual Systems Inc., Waltham Mass., June, 1996 http://www.avs.com

[5] Haeberli and Segal "Texture Mapping as a Fundamental Drawing Primitive" Fourth Eurographics Workshop on Rendering, Cohen, Puech, Sillion Editors, Paris, France, June, 1993.

[6] Isokovic, K. "Commercial 3D Graphic Game Engines" technical report, TU Berlin, Germany, May, 1997 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] M. Zöckler, D. Stalling, H.C. Hege. "Interactive Visualization of 3D-Vector Fields Using Illuminated Stream Lines". In Proceedings of Visualization '96, pp. 107-113, published by IEEE Computer Society October 1996.

[9] Schussman, Ma, Schissel, Evans, "Visualizing DIII-D Tokamak Magnetic Field Lines", In Proceedings of Visualization '2000, pp 501-504, published by IEEE Computer Society October 2000

[10] D. Stalling, M. Zöckler, H.C. Hege. "Fast Display of Illuminated Field Lines" in IEEE Transactions on Visualization and Computer Graphics, Volume 3, Number 2, April-June 1997, (ISSN 1077-2626), a publication of the IEEE Computer Society.