

Effectiveness of Silhouette Rendering Algorithms in Terrain Visualisation

Ruzinoor bin Che Mat
Sekolah Teknologi Maklumat
Universiti Utara Malaysia
06010 Sintok, Kedah Darulaman.
Malaysia.
ruzinoor@uum.edu.my

Dr. Mahes Visvalingam
Reader in Digital Cartography
Department of Computer Science
The University of Hull
Hull HU6 7RX,
United Kingdom
M.Visvalingam@dcs.hull.ac.uk

Abstract: *Silhouette Rendering Algorithms have been successfully used in various applications such as communicating shape and cartoon rendering. This paper explores how effective silhouette rendering algorithms could be used in terrain visualisation. This approach has been implemented in a 3D modelling system to create a new method of displaying and viewing terrain data in an artistic style. Real terrain data has been used to test the effectiveness of the algorithm. The resulting terrain images are closer to human drawn illustration than to shaded images.*

Keywords: *Non-photorealistic animation and rendering (NPAR), visualisation, silhouette and terrain.*

1. Introduction

The aim of this paper is to evaluate the effectiveness of silhouette rendering algorithms in terrain visualisation. Currently, terrain visualisations use photorealism in 3D. But a different approach, namely Non photorealistic Rendering (NPR), is used in this paper for creating the terrain image in 3D. The NPR technique used in this paper is dependent on silhouette rendering algorithms. The evaluation of the Wireframe Method, introduced by Raskar and Cohen [16] and adapted by Liu [13], is reported in this paper.

The digital elevation models (DEM) for Port Talbot was used as real data for modelling terrain in 3D. The results produced from this data were used in the evaluation.

Previous Work in Terrain Visualisation and NPR

Terrain visualisation can be divided into two sections, namely manual representation and automated visualisation of terrain. The manual representation of terrain has been a challenge to map makers since the earliest maps were produced. Some of the earliest maps of terrain showed mountains as rounded 'molehills': the simple, uniform, side views of a regularly rounded dome [11]. The arrangement of these 'molehill' symbols evolved over time to incorporate variations in size and shape as well as being shaded to give an impression of illumination. Dowson has produced a table to categorise and provide a good review of all of these techniques [5].

The automated methods for visualising terrain in 3D include mosaics (a pseudo-colouring the cells

of the grid), profiles, contouring, vertices, wiremesh, polygon model and photorealism (see Dowson [5] p.43). He states that "The great advantage of computer visualisation of terrain data is that of producing images very quickly and effortlessly. This is especially the case with 3D diagrams which are of great value but can be time consuming and tedious to produce using conventional cartography".

This paper falls within the area of NPR, which was defined by Green et al. [8] as concerned with producing imagery that look as if an artist had drawn it. NPR has also described by Hertzmann as "any attempt to create images to convey a scene without directly rendering a physical simulation"[9]. The purpose of this technique is to produce images that appear to have been drawn by hand.

Researchers in NPR have investigated the display of 3D worlds using various display models. Recent work has focused on the modelling of artistic images and styles using silhouette [9], graphite pencil [21], computer generated watercolour [4] and pen-and-ink illustration [20].

Green et al. [8] used an interactive NPR system, which has the capability to interactively display a custom shaded model and edge lines. However, Markosian et al. [14] presented a new real time NPR technique based on an economy of line. This technique used a rendered method for determining visible lines and surfaces, based on a modification of Appel's hidden line algorithm. They demonstrated the system with several NPR styles, which all operate on complex models at interactive frame rates. The reality is that NPR software (Piranesi) is now available on the market [17], which has a capability

for rendering any 3D image as a Non-photorealistic image.

2. Silhouettes

Researchers have defined silhouettes in different ways. Hertzmann and Zorin [10] quoted that “silhouette drawing is sufficient to convey information about simple objects, it is often insufficient for depicting objects that are complex or free-form. From many points of view, a smooth object may have no visible silhouette lines, aside from the outer silhouette, and all the information inside the silhouette is lost”. Other than that, Lake et al. [12] have defined the silhouette edge as “the edge shared by a front-facing and a back-facing polygons (see Figure 1.0). The silhouette in each frame, can be found by taking the dot products of the face normal of the two faces adjacent to an edge with the viewing vector (see Equation 1.0) and comparing the product of these two dot products with zero. If the result of this computation is less than or equal to zero, the edge is a silhouette edge and it is flagged for rendering”.

$$\text{Silhouette Edge} = (\text{face normal1} \cdot \text{eyevect}) * (\text{facenormal2} \cdot \text{eyevect}) \leq 0$$

Equation 1.0

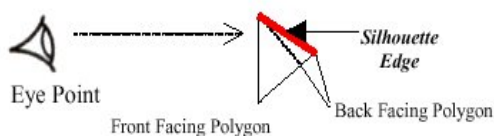


Figure 1.0: Silhouette edge detection
(Image from Lake et al. [12])

Hertzmann has provided an extensive review on silhouettes and the survey of the technological problem for finding visible silhouettes of 3D surfaces. He mentioned that "silhouette are important both perceptually and aesthetically. Efficient silhouette detection is an interesting technical problem, because silhouettes typically occur only in a few places on a 3D shape at any given moment" [9].

2.1 Techniques for Detecting Silhouettes

Researchers in NPR have produced many different techniques for detecting silhouettes. All the techniques have their advantages and disadvantages which depends on how the silhouette will be detected. The silhouette techniques can be divided into those operating on:

- The image planes (Image based)
- 3D surfaces (Model based)

- Hybrid Approach
- Highlighting Silhouettes

Image based

Saito and Takahashi [19] showed how silhouette edges could be used to enhance the display of images. The silhouette edges can be rendered by highlighting the discontinuities in the z-buffer derivative. This is similar with creases or folds in the image that can be rendered using same technique as before but for the second derivative of z-buffer. They also rendered the outlines of 3D objects by applying edge-detection filters to specially prepared depth and normal maps and compositing the results with the rest of the scene.

Curtis [3] has introduced the loose and sketchy techniques for rendering silhouettes. This technique automatically draws the visible silhouette edges of a 3-D model by using image processing and a stochastic, physically based particle system. It requires only a depth map of the model and a few simple parameters set by the user to detect the silhouette.

Model Based

Gooch et al. [7] have used model based techniques for finding the silhouette edges. The fundamental idea of this technique is that when silhouettes and other edge lines are explicitly drawn, then very low dynamic range shading is needed for the interior. Their technique was implemented by adopting a shading algorithm based on cool-to-warm tones. Warms colours are considered to advance in the image, while cool colours recede. This style of shading was adopted to ensure that black silhouettes and edge lines are clearly visible. Thus, by using silhouettes and edge lines, as well as cool-to-warm tones, they can create images that are very easy to be understood. However, this technique is complicated and difficult to implement.

Barequet et al. [1] presented an efficient tracking of perspective-accurate silhouette by reducing the problem to point location queries. They used an incremental algorithm for computing the silhouettes of polyhedral models and updating them as the viewpoint changes. This algorithm exploits the coherence in silhouettes to obtain fast updates of the silhouettes. They assumed that the silhouettes of a model from a given viewpoint is known. An edge ceases to be respectively a silhouette edge if and only if it shared by one front-facing and one back-facing polygon, and exactly one of it adjacent polygons changes orientation.

Bremer and Hughes [2] have using ray tracing technique for rendering the silhouette. The aim of

this technique is to produce a non-photorealistic rendering of an implicit surface. Silhouette and some shading near silhouettes are rendered using these techniques. Their method was directly inspired by Markosian et al. [14] that produce real time silhouette renderings of a polygonal model. This algorithm starts with finding the silhouette edges and information for shading, tests the edges for occlusion, and then draws the edges and shading information.

Hybrid Approach

This technique introduced by Northrup and Markosian [15] can be classified as the latest technique for detecting silhouette edges. It rendering silhouette outlines of 3D polygonal meshes with stylised strokes. This technique works by combining the benefits of the image-based approach with the accuracy of a geometry-based approach. It started with detecting the silhouette edges of the model and then computes the visibility and adjacency using a 2D projection of the silhouette edges. At a high level, this algorithm proceeds as follows:

- Find the silhouette edges.
- Determine visible segments of each edge.
- Apply correction for overlaps in segments.
- Link segments into smooth paths.
- Render stylised strokes along these paths.

Highlighting Silhouettes

The work done by Raskar and Cohen [16] renders the silhouette edges using a traditional polygonal rendering setup. The silhouette edges can be rendered by simply pulling the back-facing polygons towards the camera. This technique does not require pre-processing or adjacency information for dynamic scenes. Only two sets of polygons are needed to compute visible silhouette edges for a given viewpoint. These two sets are $P1$ the layer of visible polygons nearest viewpoint, and the second layer $P2$ of polygons from that viewpoint. It is determined by front-facing and back-facing polygons. The intersection of these two layers gives silhouette edges in object spaces. The silhouette edges also can be rendered by computing the projection of $P1$, $P2$ and $P1 \cap P2$ using a depth buffer directly in image space. The pseudo-code to render silhouette edges in black on white background are described below:-

- Draw background with white colour.
- Enable back face culling, set depth function to "Less Than"
- Render (front facing) polygons in white.
- Enable front face culling, set depth function "Equal To".

- Draw (back-facing) polygons in black wireframe.
- Repeat for a new viewpoint.

These methods of producing silhouettes are called the general method. Raskar and Cohen [16] provided three techniques for rendering the silhouette, namely a) Wireframe Rendering b) Translated back-facing polygons c) View-dependent modification of back-facing polygons. Liu [13] has adapted the first technique for highlighting silhouettes. He then rendered the polygons using stroke textures that divided into two types, which is hatching and stipple lines. All of these techniques have been proven to be very useful for 3D technical illustration fly-through and to create NPR animations.

3. Implementation

Borland C++ Builder Version 5.0 was used for adapting Raskar and Cohen [16] demo program, written in Visual C++. A successful 3D-terrain model was developed by reading the DEM data from the file. Then OpenGL component was used for rendering a silhouette and produced good sketches of terrain. The simple Graphical User Interface (GUI) was built using the Forms on Borland C++ Builder and produced a user-friendly environment for the user (see Figure 3.0).

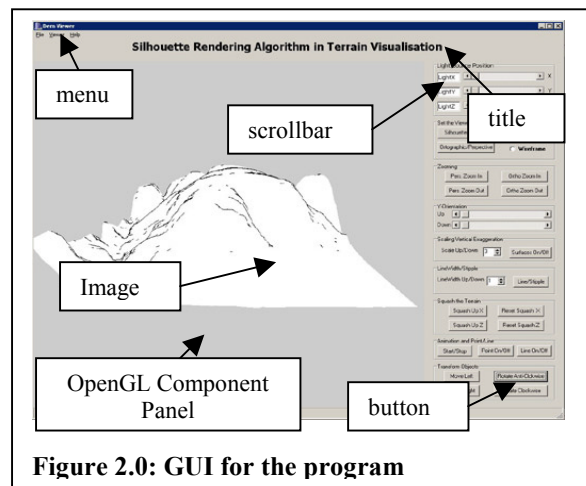


Figure 2.0: GUI for the program

The code for the silhouette rendering algorithm was written in an object oriented fashion. The advantage of using Borland C++ Builder is that it makes easier to implement the Graphical User Interface (GUI) and also creates a professional appearance. Other than that, by writing code in object oriented fashion, it makes it easier to extend this program for another purpose. It also easier for a programmer to understand the flow of this program. This program runs successfully on Windows 9x and Window NT.

4. Evaluation, Result and Discussion

Based on the literature review, one technique for silhouette rendering algorithm has been chosen for investigation, a Wireframe Method introduced by Rossignac and van Emmerick [18], and adapted by Raskar and Cohen [16] and by Liu [13]. The chosen techniques seemed appropriate for this project and were evaluated to establish whether they can produce perceptually appropriate results for terrain visualisation.

Wireframe Method

Raskar and Cohen [16] presented several simple methods to draw silhouette edges but Liu [13] only implemented one of the methods and it rendering using stroke textures. This paper investigated and implemented what has been done by Liu [13]. This method is called the Wireframe Method. The idea of this method is to render the front facing polygons as a white background and render the back facing polygons as a wireframe. It involved multi-pass rendering for highlighting silhouette edges.

Evaluation

This technique produced good 3D terrain silhouette edges. The silhouette edges produced using this method are better than those produced by Everitt's [6] One Pass Silhouette Rendering method. For investigation on this method, there are many different approaches have been used for testing. One of the approaches is by setting the line width for silhouette edges with different size. As shown by Figure 3 below, using different line width for depicting the silhouette edges, gives different results. The silhouette edges produced by line width set as 2.0 seems to give the best results. On the other hand, silhouette edges produced by line width set as 4.0 is not reasonable for depicting the silhouette edges because the line width is too thick for a silhouette edges (see Figure 3.0).

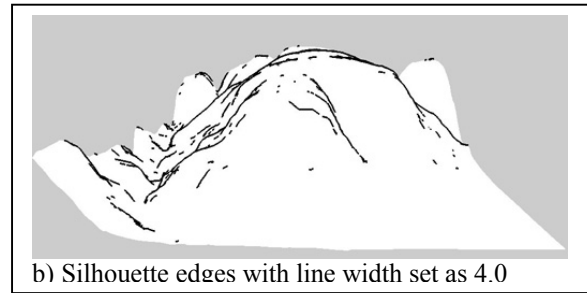
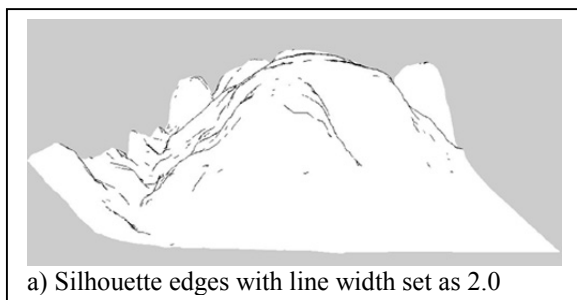


Figure 3.0: Comparison of line width.

The second approach is by setting the silhouette edges as a line and line stipple (see Figure 4.0). To set the line stipple, *glLineStipple* function has been used. The code below shows how this function is set in this program.

```
//function for setting line stipple  
glLineWidth(lineWidth);  
glLineStipple(1, 0x4444);  
glEnable(GL_LINE_STIPPLE);  
Dem.Draw();  
glDisable(GL_LINE_STIPPLE);
```

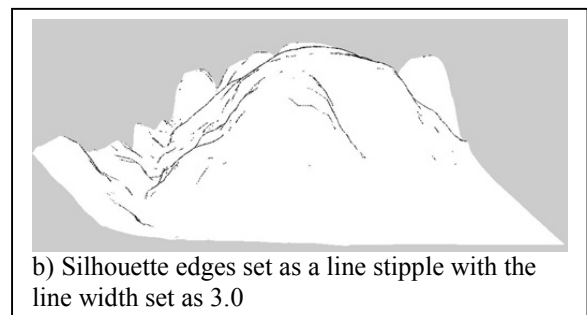
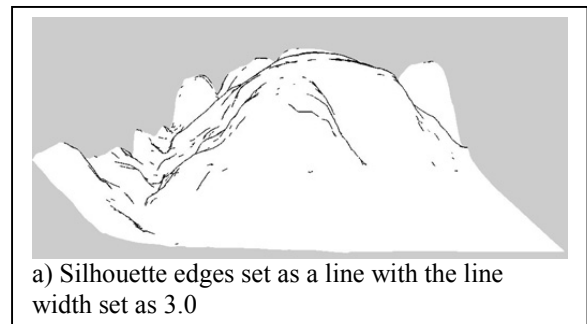


Figure 4.0: Comparison of silhouette edges between line and line stipple.

Figure 4.0 shows that the silhouette edges set by line stipple looks as if it was drawn by hand and shows more clearly the slopes and breaks of slope. The steeper slopes appear as short stipple lines and the flat areas are shown as more stipple lines. This variation makes it look hand drawn. But, people tend to draw solid lines. This means that if the people want to make their image like a hand drawn, the line stipple can be used; otherwise solid lines can be used.

Another approach for investigating this method is setting the different camera projections. One projection is set as orthographic and the other one is set as perspective (see Figure 5.0).

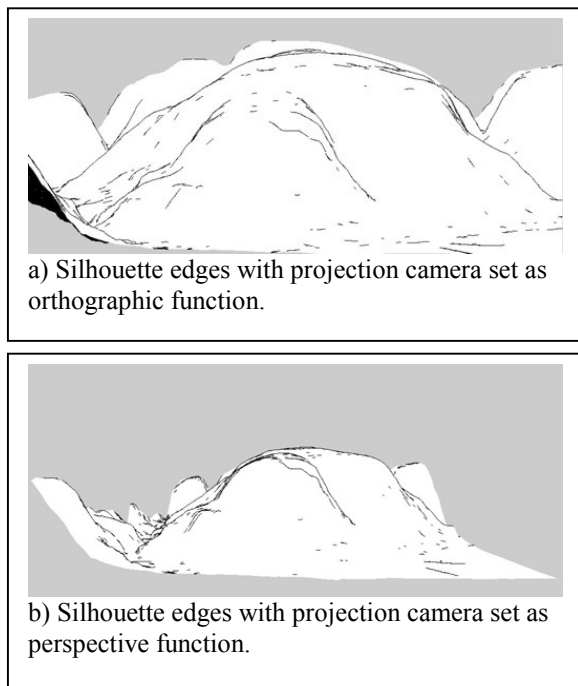


Figure 5.0: Comparison of silhouette edges between orthographic and perspective projections.

The silhouette edges in an orthographic projection are better than in perspective projection. The orthographic projection gives more information about the edges of terrain where it shows in detail the slopes and break slopes. However for perspective projection, some of the edges seem to be joining each other as a line.

The last effect that can be observed from investigating this method is that the results of terrain image produced extra lines on the silhouette edges. These effects are shown in a square in Figure 6.0.

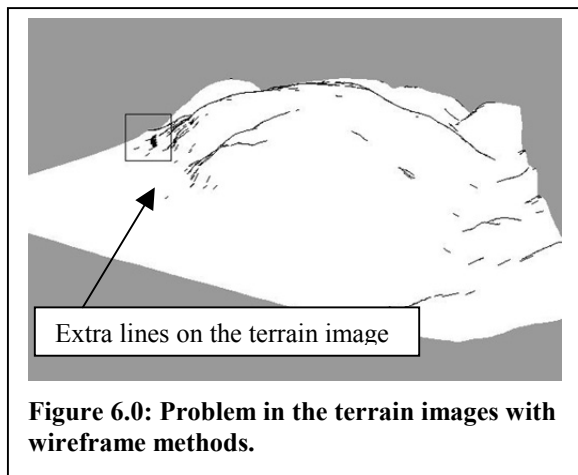


Figure 6.0: Problem in the terrain images with wireframe methods.

5. Conclusion

From the evaluation, this method seems effective for showing the silhouette edges. By setting the edges as a different type of line and setting different camera projections, the user can get different depictions of terrain models. This method works in real-time and is relatively fast for rendering the silhouette edges. The edges of the silhouettes in wireframe method are rendered for each polygon, in order to produce the terrain image. This causes problems, where some lines look like the straight lines. On the whole, this method is very effective and suitable for use with the sketching technique.

6. Acknowledgements

The digital elevation models (DEM) used for testing was provided by Ordnance Survey. The data remain their copyright.

Thanks to my wife for correcting my English in this paper.

7. References

- [1] Barequet,G.; Duncan,C.A.; Goodrich,M.T.; Kumar,S.; Pop,M.; “Efficient Perspective-Accurate Silhouette Computation”, Proceedings of the fifteenth Annual Symposium on Computational Geometry, pp. 417 – 418, 1999.
- [2] Bremer, D.J.; Hughes, J.F.; “Rapid Approximate Silhouette Rendering of Implicit Surfaces”, In Proc. The Third International Workshop on Implicit Surfaces, 1998.
<http://citeseer.nj.nec.com/cache/papers2/cs/11768/http://zSzzSzwww.cs.brown.edu/Szpeople/SzjfhzSzisSzpaper.pdf/rapid-approximate-silhouette-rendering.pdf>
[Accessed 18/5/2002]
- [3] Curtis C.; “Loose and Sketchy Animation”, SIGGRAPH 1998 Technical Sketch, 1998.
<http://www.cs.washington.edu/homes/cassidy/lose/sketch.html> [Accessed 4/05/2002]
- [4] Curtis, C.J.; Anderson, S.E.; Seims J.E.; Fleischer, K.W.; Salesin, D.H; “Computer-Generated Watercolor”, Computer Graphics (Proceedings of SIGGRAPH '97), Annual Conference Series, pp. 421 – 430, 1997.
- [5] Dowson, K.; “Towards Extracting Artistic Sketches and Maps from Digital Elevation Models”, unpublished PhD Thesis, The Department of Computer Science, The University of Hull, 203pp., 1994.

- [6] Everitt's, C.; "One-Pass Silhouette Rendering with GeForce and GeForce2", Technical Report from NVIDIA Corporation, 2000.
<http://partners.nvidia.com/marketing/developer/devrel.nsf/TechnicalDemosFrame?OpenPage>
[Accessed 4/08/2001]
- [7] Gooch A.; Gooch B.; Shirley P.; Cohen E.; "A Non-Photorealistic Lighting Model for Automatic Technical Illustration", Proceeding of SIGGRAPH 98, pp. 447 – 452, 1998.
- [8] Green, S.; Salesin, D.; Schofield S.; Hertzmann, A.; Litwinowicz, P.; Gooch, A.; Curtis, C.; Gooch. B.; "Non-Photorealistic Rendering", SIGGRAPH 99 Course Notes, 1999.
http://www.cs.utah.edu/npr/papers/npr_course_Sig99.pdf [Accessed 28/03/2002]
- [9] Hertzmann, A.; "Algorithm for Rendering in Artistic Styles", PhD Thesis Department of Computer Science, New York University, 159pp, 2001.
- [10] Hertzmann, A.; Zorin, D.; "Illustrating Smooth Surfaces", In SIGGRAPH 2000 Proceedings, 2000.
<http://citeseer.nj.nec.com/cache/papers2/cs/15894/http:zSzzSzwwww.mrl.nyu.eduzShertzmannzSziillustrationzShertzmannzorin.pdf/hertzmann00illustrating.pdf>
[Accessed 1/12/2002]
- [11] Imhof, E.; "Cartographic Relief Presentation", Steward, H.J. (ed.), Walter de Gruyter, 389pp, 1982.
- [12] Lake, A.; Marshall, C.; Harris, M.; Blackstein, M.; "Stylised Rendering Techniques For Scalable Real-Time 3D Animation", Graphics Algorithms and 3D Technologies Group (G3D) Intel Architecture Labs(IAL) University of North Carolina at Chapel Hill, 2001.
<ftp://download.intel.com/ial/3dsoftware/toon.pdf>
[Accessed 4/12/2001]
- [13] Liu, J; "Computer Generated pen-and-ink Illustration", Technical Reports Department of Computer Science SUNY at stony brook, 2001.
http://www.cs.sunysb.edu/~jliu/cse528/final_proj_report.html [Accessed 8/11/2001]
- [14] Markosian, L.; Kowalski, M.A.; Trychin, S.J.; Lubomir, Bourdev, L.D.; Goldstein, D.; Hughes, J.F.; "Real-Time Nonphotorealistic Rendering", Computer Graphics (Proceedings of SIGGRAPH '97), Annual Conference Series, pp. 415 – 420, 1997.
- [15] Northrup, J.D.; Markosian, L.; "Artistic Silhouettes: A Hybrid Approach", To appear in the proceedings of NPAR 2000, 2000.
<http://citeseer.nj.nec.com/cache/papers2/cs/14390/http:zSzzSzwwww.cs.brown.eduzSzresearchzSzgraphicszSzresearchzSzpubzSzpaperszSzsil-300dpi.pdf/northrup00artistic.pdf>
[Accessed 1/11/2001]
- [16] Raskar, R.; Cohen, M.; "Image Precision Silhouette Edges", Proceedings of the 1999 symposium on Interactive 3D graphics, Atlanta, GA USA, pp. 135 – 140, 1999.
<http://www.cs.unc.edu/~raskar/NPR/sil.html>
[Accessed 3/11/2001]
- [17] Richens, P.; "Technical Details", Informatix Software International, 2000.
http://www.informatix.co.uk/pir_ajart.htm
[Accessed 5/5/2002].
- [18] Rossignac, J.; van Emmerick, M.; "Hidden Contours on a Framebuffer", Proceedings of the 7th Eurographics Workshop on Computer Graphics Hardware, Cambridge, UK, Sept. 1992.
- [19] Saito, T.; Takahashi, T.; "Comprehensible Rendering of 3D Shapes", Computer Graphics (Proceeding of SIGGRAPH '90), 24(4), pp. 197 – 206, 1990.
- [20] Salisbury M.P.; Wong M.T.; Hughes, J.F.; Salesin, D.H.; "Orientable Textures for Image-Based Pen-and-Ink Illustration", Computer Graphics (Proceedings of SIGGRAPH '97), Annual Conference Series, pp. 401 – 406, August 1997.
- [21] Sousa, M.C.; Buchanan, J.W.; "Computer-generated graphite pencil rendering of 3D polygonal models", Computer Graphics Forum, 18(3), pp. 195 – 207, 1999.