VICTORIA UNIVERSITY OF WELLINGTON

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Comic-strip rendering

Peter Hall

Technical Report CS-TR-95/2
January 1995

Abstract

Photorealistic rendering methods have been at the centre of much computer graphic research and remain in that position to date. The aim of photorealism is to produce computer graphic images that cannot be distinguished from photographs. Such methods have produced impressive results, and in some cases the aim has been met. However, photorealism is not always a desirable goal, sometimes a hand-drawn quality to the image is wanted and this requires a non photorealistic renderer. Non photorealistic rendering methods have received very little attention, our work addresses this gap. We have developed a method of rendering images in a style often used in popular comic strips. Our renderer accepts three dimensional models and lighting information as input. Initially, the scene is lit using standard techniques. Next shadow effects are drawn directly on to the models and these are rendered without a lighting model. The contribution our work makes is to introduce a general mechanism by which a new, non-photorealistic, style of rendering can be achieved. We demonstrate this with a particular mechanism that produces images which resemble the drawings that appear in many popular comic-strips. This paper describes our method.

Publishing Information

This paper has been submitted to Eurographics 1995.

Author Information

Peter Hall is a lecturer at Victoria University. His principal research interests are scientific visualisation, and computer vision.

Comic-strip rendering

Peter Hall
Department of Computer Science
P.O. Box 600
Victoria University of Wellington
Wellington
New Zealand
Email: peter@comp.vuw.ac.nz

Abstract

Photorealistic rendering methods have been at the centre of much computer graphic research and remain in that position to date. The aim of photorealism is to produce computer graphic images that cannot be distinguished from photographs. Such methods have produced impressive results, and in some cases the aim has been met. However, photorealism is not always a desirable goal, sometimes a hand-drawn quality to the image is wanted and this requires a non photorealistic renderer. Non photorealistic rendering methods have received very little attention, our work addresses this gap. We have developed a method of rendering images in a style often used in popular comic strips. Our renderer accepts three dimensional models and lighting information as input. Initially, the scene is lit using standard techniques. Next shadow effects are drawn directly on to the models and these are rendered without a lighting model. The contribution our work makes is to introduce a general mechanism by which a new, non-photorealistic, style of rendering can be achieved. We demonstrate this with a particular mechanism that produces images which resemble the drawings that appear in many popular comic-strips. This paper describes our method.

Keywords and phrases: lighting models, non-photorealistic rendering, comic-strips.

1 Introduction

One of the principal aims of computer graphics research has been to synthesise images that appear to be photographs. The aim is partly manifest in scan-line methods (see a standard text such as Foley et al., 1990), but is strongly associated with ray tracing and radiosity methods. Both ray tracing and radiosity methods are special case solutions of the rendering equation (Kajiya, 1986). Early ray tracing methods produced cast shadows (see Appel, 1968). Whitted (1980) extended the method to account for multiple reflections and refractions. Objectionably sharp reflections associated with ray traced images were softened by distributed ray tracing (see Cook et al., 1984). None of these ray tracing methods simulate indirect illumination. To do so requires ray tracing from the light source to the eye, which is opposite to the usual direction of standard ray-tracers. Watt (1990) uses such a ray tracer to produce images of caustics on swimming pools. Whichever way ray tracing proceeds, it accounts most readily for light transport that involves some specular component. This stands it in contrast with radiosity methods, because they account for the diffuse transport of light. Radiosity first arose in a computer graphics context in 1984 (see Goral, 1984). Only recently have researchers been able to use radiosity methods to render glossy surfaces (Aupperle and Hanrahan, 1993).

Our aim is not to improve image quality (in the sense of synthesising images by a more accurate model of the interaction of light with matter) but to develop an altogether different style of rendering. We introduce our method via the following analogy. A scene, comprising a set of objects, is lit. We draw a pattern onto the surface of objects. The pattern we draw will leave a record of which parts of the scene were in shadow, and to what extent they were in shadow. When we view the scene, under uniform illumination, an impression of the original light distribution will be visible in the pattern we have drawn. Consequently, objects in the scene will appear to be three dimensional because the shape cues will be drawn. We can conceive of a number of possible algorithms that conform to this analogy. The primary characteristic of the method presented in this paper is that drawing is based upon a regular grid of orthogonal planes in three dimensions. We envisage that it might find use for rendering images in a comic-strip style, for example. This places the method in the class of non photorealistic renderers. Note that we do not seek to reproduce a hand-drawn comic-strip style exactly, rather we are investigating methods for drawing by computer – "artificial drawing" so to speak.

By comparison with the quest for photorealism the amount of research into non photorealistic rendering has been very small. Haeberli (1990) demonstrated that a suitable point-spread function can not only be used to post-process digital images, but also can result in ray-traced images that appear to be painted. A post-processing method was used by Saito and Takahashi (1990) to produce images that appeared drawn in a technical style, such as cross-hatching. Hanrahan and Haeberli (1990) produced a system that enabled users to paint directly onto the surface of three dimensional objects. Our work is similar to each of these non photorealistic methods, without being any of them. Like Saito and Takahashi (1990), our method can yield images shaded in a cross-hatch style. Unlike those authors, our mechanism operates in three dimensions. Like Hanrahan and Haeberli (1990), we paint on the surface of objects. Unlike them we draw shadows - our method is sensitive to a lighting model. We use a simple point-spread function, in contrast to Haeberli (1990), yet we are able to produce images that have a 'hand-drawn' quality to them. Our method was first developed in a volume rendering context and then in a more general scientific visualisation arena; see Hall (1992), and Hall (1993) respectively. We have developed that method and now present it in a form suitable for a more general computer graphics environment.

We present a simple algorithm in Section 2, and show examples of the images it produces in Section 3. In Section 3 we mention other algorithms that conform to the analogy above, and conclude the paper.

2 Method

As the introductory analogy suggests, there are two main procedures to the algorithm.

1. At a point, compute the light entering the eye from that point.
2. Use that light to compute a colour for the point.

In the first of these procedures a scene is lit using standard lighting models, such as that due to Phong (see Phong 1975). The second procedure is a post-process that alters the colour of the surface in some way. However, unlike Saito and Takahashi (1990) who perform post-processing on pixels in an image (and a set of buffers associated with it), our post-processing takes place before colour ever reaches the pixel, and in this sense is closer to the ray-tracing work of Haeberli

(1990). The remainder of this section describes a post-processing mechanism. This particular mechanism yields images that are rendered in a cross-hatch style in a single pass of the renderer.

Given a colour value at a point, estimated via a standard lighting model, we compute the energy it carries and use that to adapt a three-dimensional grid. The three dimensional grid is virtual. It comprises sets of slabs that extend through the entire scene. Each slab has one of the major axes as its normal so that the slabs can be arranged into sets according to their normal, with slabs from any two distinct sets being orthogonal. Slabs are of finite width, and are symmetric about the plane of infinitesimal width. The distance between theses planes is called the inter-plane gap.

Adapting the grid means specifying (1) the inter-plane gap, and (2) the slab width. This adaptation is sensitive to the amount of light leaving a surface element in the direction of the observer. For each set of slabs we make a binary decision whether the point lays in any slab in that set. Because there are three sets of slabs we make three independent decisions. An obvious way to combine these decisions is to decide the point lies in grid if it lies in any of its slabs. In such a case the three-dimensional grid will appear as lines drawn on the surface of the object. Hence the grid can be regarded as a three-dimensional texture that is defined procedurally.

If the point lies on the grid then it will be drawn with ‘shadow’ colour, otherwise it will be drawn with ‘surface’ colour, or some combination of these. Typically the ‘shadow’ colour will be just the colour computed by the standard renderer and the ‘surface’ colour will be the diffuse reflection coefficients of the surface. Using computed light as shadow colour rather than black, say, has the dual advantages of being sensitive to coloured shadows and producing more pleasing results. Moreover, the images produced would be very hard to produce by hand, and this is in line with our “artificial drawing” goal.

The algorithm is summarised in Figure 1, using a two-dimensional example for simplicity. Given the brief description above it is seen that there are four main step to the algorithm:

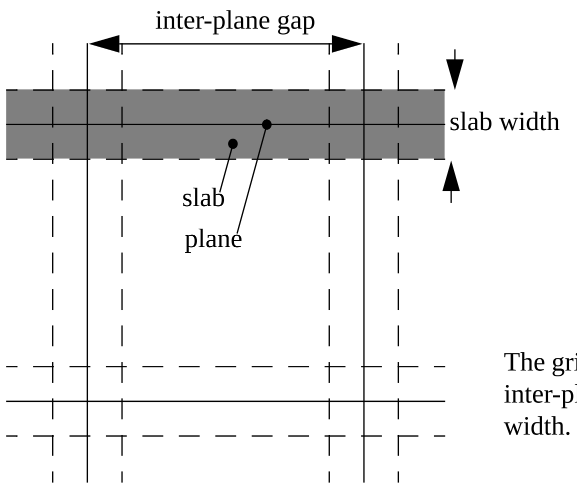
1. The grid is adapted as a function of the light entering the eye from a given point. Adaptation of the grid must be such as to ensure continuity of grid lines from lighter areas into darker areas on a surface. If improperly done, the effect can be quite misleading. The width of the grid planes is further adjusted to allow for the orientation of the elemental surface.

2. The point is transformed into grid-coordinates. For each set of slabs a binary decisions is made as to whether the point lies in any slab of that set.

3. The independent binary decisions are combined using a Boolean expression. The point lies on the grid only if the Boolean expression yields TRUE. A particular Boolean expression is used to define a particular drawing pattern. Drawing patterns include cross-hatch, or half-tone, or some other style. Each drawing pattern is defined in three-dimensions. There are 256 different drawing patterns that can be defined.

4. Finally, the colour of the surface is re-computed and stored in a pixel. Inevitably, aliasing effects arise and must be dealt with. Here the surface colour and shadow colour are blended.

Each of these steps will now be detailed in turn.



The grid to be adapted has well-defined inter-plane gap and a well defined slab width.

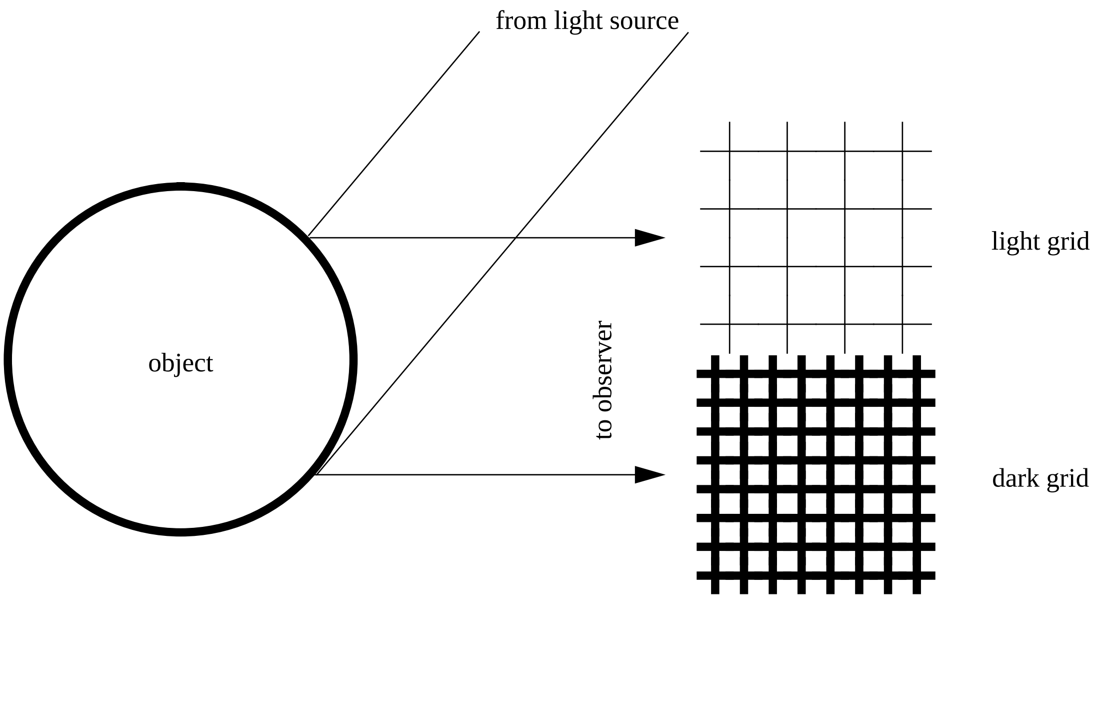


Fig 1 : The light leaving an object is used to adapt a grid.

2.1 Adapting the grid

Adapting the grid is necessary to give an impression of shadow. In darker regions, a grid with a narrower gap between wider planes should be used. This increases the chance that any given point will be found to lie on the grid and, hence, increases the chance that it will be drawn with shadow colour rather than surface colour. Conversely, in lighter regions a grid with a wider gap between thinner slabs should be used.

To compute the energy value, E, that controls this adaptation we use

$$E = \max(r, g, b)$$

where max is an operator that selects the maximal value of its operands, and r, g, b are the energies carried in the red, green, and blue channels respectively. We assume that E will be estimated in the range [0,1].

We adapt the inter-plane gap by restricting it to be some multiple of a minimum gap. Care should be taken when adapting the inter-plane gap because the lines drawn on the surface via the grid provide shape cues in addition to shading. We wish that any line drawn on a surface in a region that is light extends unbroken and un-deformed into regions that are darker. If this condition is not met then a cross-hatch style pattern, for example, will give misleading visual cues as to the underlying shape. To overcome this undesirable effect we ensure that the multiple is an integer power of two. That is, if g_0 is a minimum gap we compute an adapted gap, g(E), by

$$g(E) = g_0\, 2^{\mathrm{floor}(kE)}$$

where E is the light energy estimated to reach the observer, k is a parameter that scales E, and floor(x) is a function returning the greatest integer that is not larger than x. The parameter, k, controls the rate at which the gap widens. Notice that we could choose any integer base in this adaption. We use two because it provides the slowest ascent in the gap with respect to a given rise in E.

The width of grid slabs can be adapted in a very similar way. We have chosen to use the minimum width in the lightest region as the starting point for adapting slab widths. This leads to an falling off of widths according to

$$w(E) = w_0 2^{\mathrm{floor}(-kE)}$$

but is otherwise the same as for inter-plane gap. A more significant adaptation of plane width is one that depend on the orientation of the surface element. This adaptation is useful for ensuring that the width of a slab remains visually constant. This is done by observing that for any slab the width that is projected onto the surface element is in inverse proportion to the sine of the angle between them. That is, if w(E) is the width of the slab then its width on the surface will be w(E) / sin(θ), as shown in Figure 2, below.

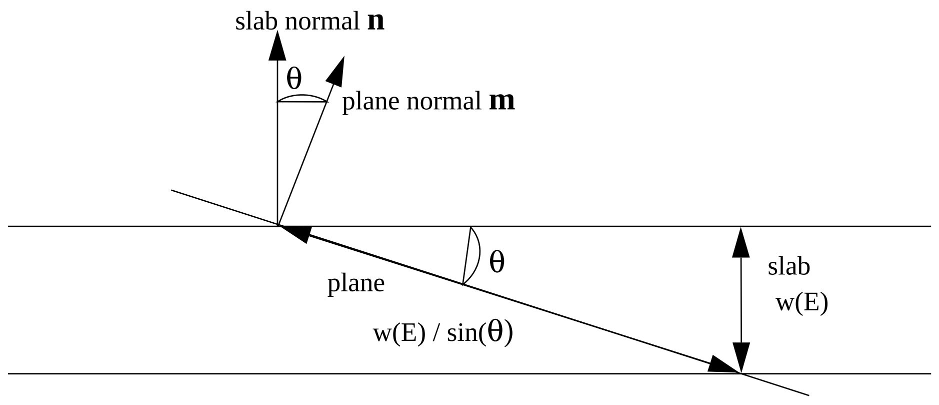


Fig 2 : The projection of a slab onto a surface element.

The solution is to re-define the slab-width by multiplying it by the absolute value of sin(θ). This is computed via the cross-product, ⊗, of the normal vectors **n** and **m**, for the slab and surface respectively. Hence the full adaptation of a slab is

$$w(E) = w_0^{\text{floor}(-kE)} \, \| \mathbf{n} \otimes \mathbf{m} \|$$

where || a || denotes the absolute value of a. Notice that this adaptation causes slab width to tend to zero as the two normals approach parallel or anti-parallel.

2.2 Testing a point

The point being rendered must be tested against the adapted grid. We compute the distance of the point to the nearest grid plane in each set of parallel slabs. Then we test the point against each nearest plane independently. To be more concrete, suppose a point at (x,y,z) in world space maps to (u,v,w) in a ‘grid’ space. Then

$$d_u = \| u - g_u \text{floor}(u / g_u) \|$$

is the minimum distance from that point to any of the planes that have (1,0,0) as their normal in grid space; these are the vw planes. Note that g_u is the adapted gap for those planes. We define a Boolean variable, b_u, as

$$b_u = \begin{cases} 1 \text{ if } d_u \le w_u \\ 0 \text{ if } d_u > w_u \end{cases}$$

where w_u is the adapted width. We can apply analogous operations to the v and w components of the point. Thus we obtain three Boolean variables b_u, b_v, and b_w. These variables are used to decide if the point lies in the grid.

2.3 Combining the Boolean variables

The three variables, b_u, b_v, and b_w are combined to generate an index, i, into a look-up table:

$$i = 4*b_u + 2*b_v + b_w$$

Only if the look up table entry has the value 1 does the point lie in the grid. This look-up table defines a Boolean expression. To see this, consider a truth table for three Boolean variables, A, B, and C. Any Boolean relation between these is defined by a final column, X, in the truth table. It is this final column that is the look-up table. Because there are three independent variables the truth table has eight rows. Hence, there are 256 different Boolean expressions that can be generated. The particular style of shading depends upon the Boolean expression we use. To create a cross-hatch shader we decide that the point is painted with shadow colour if it is in any plane. Hence the Boolean expression we want is b_u OR b_v OR b_w. To create a half-tone effect we use the Boolean expression (b_u AND b_v) OR (b_v AND b_w) OR (b_w AND b_u). These two expressions are shown in Table 1, below. Not every possible Boolean expression will be useful. Useful expression include 00001111 which is the vw plane, 00110011 which is the wu plane, and 00111111 which is either of these planes.

b_u	b_v	b_w	b_u OR b_v OR b_w	(b_u AND b_v) OR (b_v AND b_w) OR (b_w AND b_u)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Table 1: Boolean expressions for Cross-hatching and half-toning.

2.4 Computing a colour and antialiasing

Antialiasing is required to prevent the lines drawn on the surface from appearing ‘jaggy’. The correct way to do this would be by computing the fractional area of the surface element that is covered by the line being drawn and deciding on a colour by weighted sums. This expensive operation can be replaced by a less costly but faster operation. We compute the minimum distance, d, to the nearest plane and then combine surface and shadow colour using linear interpolation:

$$\mathbf{C} = d\mathbf{C}_{surf} + (1-d)\mathbf{C}_{shd}$$

where d = min(d_u, d_v, d_w). This low-cost approach produces satisfactory results when the lines are narrow.

3 Results

All the results presented below were produced with 24 bits of colour. Texture maps are pre-filtered using the ‘summed-area tables’ method (see Crow 1984). Cast shadows are created using a shadow-z-buffer (see Reeves et al. 1987); with six such buffers wrapped around each light source. The final image is post-filtered by simple area averaging of four pixels into one.

The result of using different Boolean expressions can be seen in Figure 3. From top left to bottom right the expressions are: 11111111, 01111111, 00010111, 00000000 - as defined by the result column of the truth table. These expressions could be interpreted as “always on the grid”, “on any grid slab”, “on any slab intersection”, and “never on grid” respectively. The effect of the first of these is to produce an image with whatever lighting model was used - in this case the Phong model. The effect of the last expression is to produce an image of the surface colour - its diffuse reflection coefficients including any texture map.

The sensitivity of the method to cast shadows is seen by comparing Figure 4 with Figure 5. Figure 4 shows an image of a ball in a kitchen with a single, white, point light source. It has been rendered using a standard lighting model. Figure 5, shows a comic-strip style rendering of the same image. The Boolean expression used for the ball was 01011111, and was 00111111 for the wall and floor. Notice the highlight drawn on the surface of the ball in the lines.

Figure 6 shows the same sample scene, again in comic-strip style. The Boolean expressions for the ball and for the wall are 00111111 and 0101111 respectively. The wider lines in the case shadow show the effect of the poor anti-aliasing method - the short diagonal lines that cross the main lines are possibly undesirable artefacts.

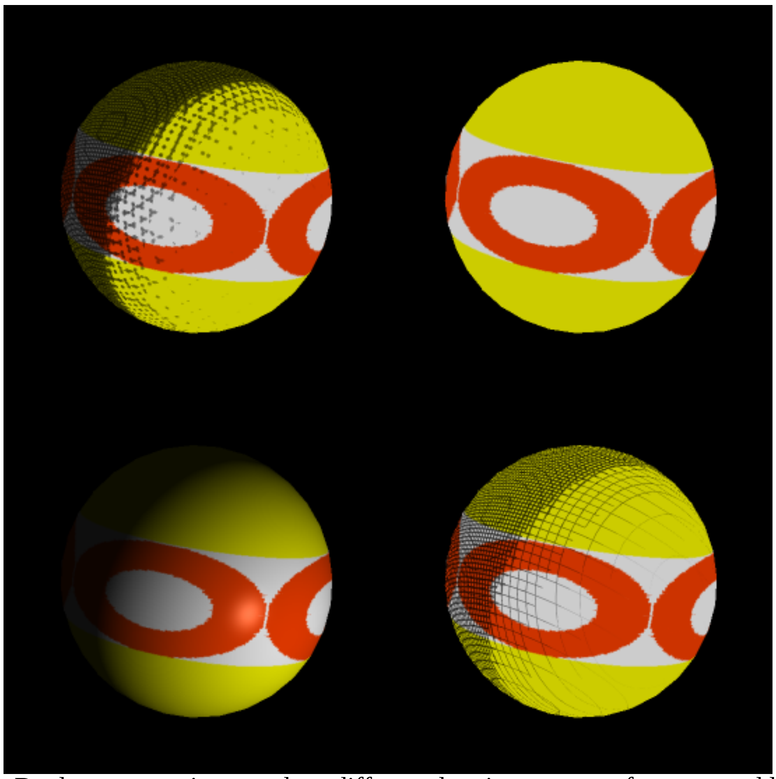


Fig 3 : Different Boolean expressions produce different drawing patterns: from top and left the grid patterns are defined by 11111111, 01111111, 00010111, 00000000.

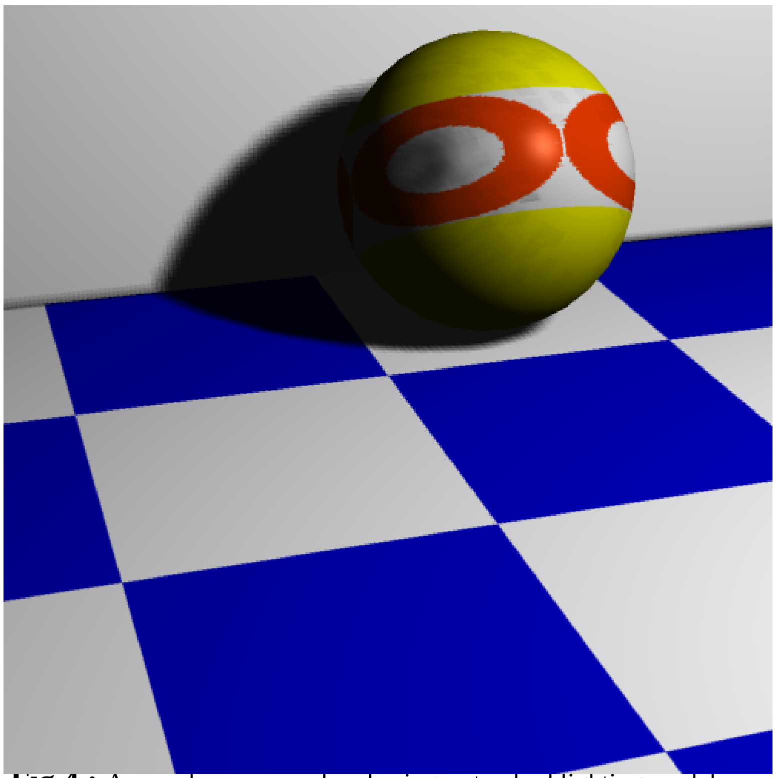


Fig 4 : A sample scene rendered using a standard lighting model.

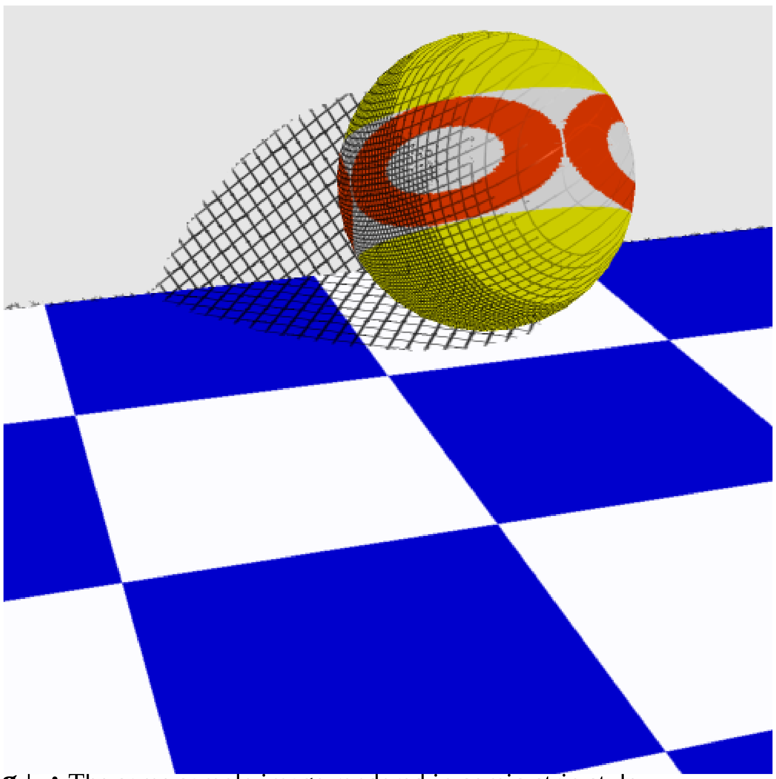


Fig 5 : The same sample image rendered in comic-strip style.

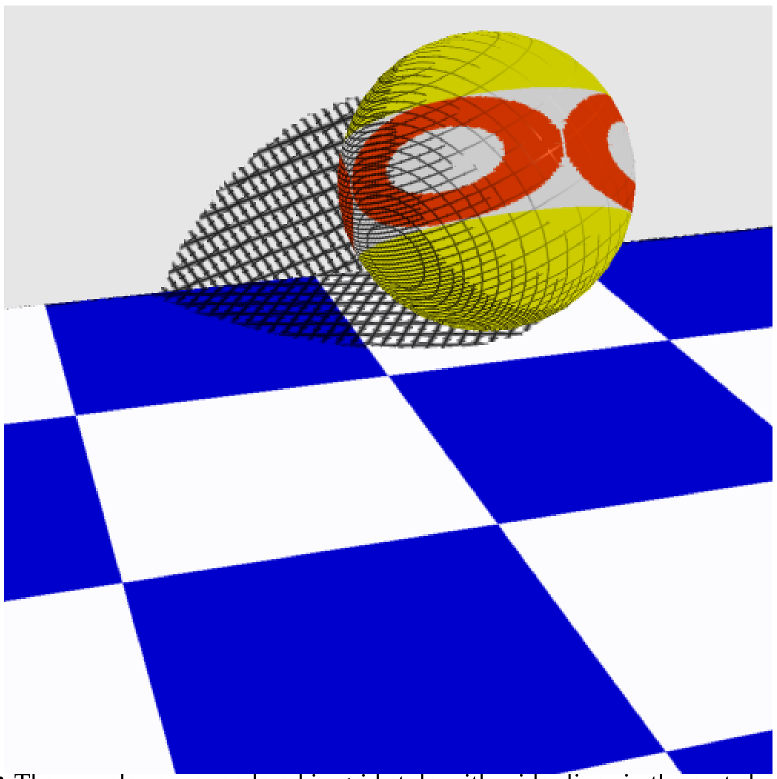


Fig 6 : The sample scene rendered in grid style with wider lines in the cast shadow.

4 Conclusion

We have provided a method that produces images with a ‘drawn’ feel to them. The method is point based, and is easily incorporated into a standard renderer. We have used a scan-line renderer here, but we could have used a ray-tracer equally well, interfering with the rendering stage of a radiosity solution would also be possible. The method works by interposing additional processing between a lighting calculation and a final rendering. This makes the mechanism sensitive to lighting conditions - one of its principle advantages. Additionally, each model can be rendered in a different style, so the method is versatile.

The novelty of our work exists in: (1) the interposition of the second procedure to the lighting calculation, and (2) the mechanism of that procedure. We have presented only a simple algorithm that produce a comic-strip style of rendering. One of its faults is that the regular grid yields images with a regular pattern. We are developing additional algorithms that address this issue and which fit with our overall scheme. This work provides a valuable step towards our goal of “artificial drawing”.

5 References

Appel, A. (1968) Some techniques for the machine rendering of solids. *AFIPS Conf. Proc. 32*: 37-45

Aupperle, L., and Hanrahan, P. (1993) A hierarchical illumination algorithm for surfaces with glossy reflection. *Proc. SIGGRAPH ‘93*: 155-162

Cook, R.L., Porter, T., and Carpenter, L. (1984) Distributed ray tracing. *Comput. Graph. 18(3) (Proc. SIGGRAPH ‘84)*: 137-144

Crow, F.C. (1984) Summed-area tables for texture mapping. *Comput. Graph. 18(3) (Proc. SIGGRAPH ‘84):* 207-212

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990) Computer graphics principles and practice, second edition. *Addison-Wesley, Reading, MA, USA.*

Goral, C., Torrance, K.E., Greenberg D.P. (1984) Modelling the interaction of light between diffuse surfaces. *Comput. Graph. 18(3) (Proc. SIGGRAPH ‘84)*: 212-222

Haeberli, P. (1990) Paint by numbers: abstract image representations. *Comput. Graph. 24(4) (Proc. SIGGRAPH ‘90)*: 207-214

Hall, P.M. (1993) Cross hatch shading for visualisation. *Tech. Report CS-93-10. Department of Computer Science, Sheffield University, England*.

Hall, P.M. (1992) Non-photoreal shape cues in the context of volume rendering. *Tech. Report CS-92-09 Department of Computer Science, Sheffield University, England.*

Haeberli (1990) Hanrahan, P., and Haeberli, P. (1990) Direct WYSIWYG painting and texturing on 3D shapes. *Comput. Graph. 24(4). (Proc. SIGGRAPH ‘90)*: 215-224

Kajiya, J. (1986) The rendering equation. *Comput. Graph. 20(4) (Proc SIGGRAPH ‘86)*: 143-150

Phong, B-T. (1975) Illumination for computer generated pictures. *Comm. ACM 18(6),* June 1975

Reeves, W., Salsin, D., and Cook, R. (1987) Rendered antialiased shadows with depth maps. *Comput. Graph. 21(4) (Proc.SIGGRAPH ‘87):* 283-291

Saito, T., and Takahashi, T. (1990) Comprehensible rendering of 3-D shapes. *Comput. Graph. 24(4). (Proc. SIGGRAPH ‘90)*: 197-206

Watt, M. (1990) Light-water interaction using backward beam tracing. *Comput. Graph. 24(4). (Proc. SIGGRAPH ‘90)*: 377-386

Whitted, T. (1980) An improved illumination model for shaded display. *Comm. ACM 7(4)*: 342-349