Realtime Lighting Manipulation in Color via Lookup Tablee

K. B. Evans

National Research Council of Canada

Ottawa, Ontario

ABSTRACT

Encoding surface normal vectors and storing them in a color frame buffer permits realtime modification of scene lighting by updating only the lookup table contents. This paper extends earlier work [BASS81] along these lines by allowing the use of up to seven colors in the image. Also described is a simple stochastic method for removing the distracting appearance of the facets which arise in the use of methods for normal vector encoding. These two improvements allow the user to apply multi-colored surface patterns to complex free-form shapes and then interactively modify a Phong lighting model containing an arbitrary number of light sources. In this work the position and characteristics of three light sources can be manipulated in real time. The pattern generation technique used is also briefly described since it provides a simple method of enhancing the perception of shape by the selection of an appropriate tiling pattern and/or the creation of cuts or holes in the object to allow inspection of obscured details.

RÉSUMÉ

Par le codage de données concernant des surfaces et leur stockage dans le tampon d'images d'un dispositif d'affichage à trame couleur, il est possible de modifier l'éclairage d'une scène en temps réel par la simple mise à jour de la table de consultation. La présente communication traite des perfectionnements apportés aux premiers travaux dans ce domaine, grâce auxquels on peut maintenant utiliser jusqu'à sept couleurs dans l'image. Elle décrit également une méthode stochastique simple pour éliminer l'apparence gênante des facettes qui se produit lorsqu'on utilise les méthodes de codage normales. Ces deux perfectionnements permettent à l'utilisateur d'appliquer des figures de surface multicolores à des formes non imposées complexes et de modifier ensuite en mode interactif un modèle d'éclairage Phong comprenant un nombre arbitraire de sources lumineuses. Dans le présent travail, on peut manipuler en temps réel la position et les caractéristiques de trois sources lumineuses. On décrit aussi sommairement la technique de génération de figures, car elle permet d'utiliser une méthode simple pour améliorer la perception de la forme, basée soit sur le choix d'un quadrillage approprié, soit sur la création de fentes ou trous dans l'objet, soit sur les deux, de façon à permettre l'examen de détails non précis.

Keywords: realtime lighting dynamics, lighting models, lookup tables, normal vector encoding, image synthesis, parametric surfaces, B-splines.

OVERVIEW OF THIS WORK

In our work with B-splines, it proved disconcertingly easy to produce simple shaded images that were hard to interpret unambiguously. The use of a Phong lighting model [PHON75], after a number of trial iterations, went a long way towards enabling the viewer to understand a single viewpoint. The selection of suitable surface patterns or textures provided additional cues. It therefore seemed reasonable to assume that a blend of surface patterns, Phong lighting, multiple light sources and the ability to manipulate the lighting in realtime, would yield a relatively inexpensive and unambiguous rendering of complex surfaces.

This paper describes a visualization technique with the following properties:

- the object is covered with a tiling pattern using up to seven colors which, in the case of the B-splines used here, follow lines of constant parameter;

- the two sides of a surface are readily distinguished by the use of different tiling patterns.

- once the object has been rendered (as described below), the user is able to modify the lighting model parameters and position the lights in realtime. Up to three lights are used in this work, since there happen to be four buttons on the tablet puck.

ENCODING OF NORMALS

The unit normal vector of a surface at pixel (i,j) can be encoded in numerous ways. The following are three simple schemes:

1) (x,y) encoding of cartesian coordinates
2) equal stepsize encoding of (θ,ϕ) spherical coordinates
3) equal projected area encoding of (θ,ϕ) spherical coordinates.

At the time of writing,(1) and (2) have been tried and (3) remains as an expected improvement over (2). The top rings in figures 1 and 2 show the results of applying these schemes, without any stochastic effects, to a torus with monochrome, unpatterned, untextured shading using (x,y) encoding (fig. 1) and (θ,ϕ) encoding (fig. 2).

Bass' original results suffered from two problems: lack of controlled colors for delineating surface contours (isoparametric lines in parametric surfaces) and an objectionable facetting which causes the viewer to see a distorted approximation of the surface. The first step to overcoming these problems is the use of a true color approach. Suppose that each unit normal vector on the surface is encoded into an 8 bit value, x, representing 256 possible normal vector directions in the hemisphere facing the viewer. If this value, X, is loaded into the Red, Green and Blue banks of the framebuffer as either X or 0, then there are eight possible colors (seven excluding black) for each pixel: red, green, blue, white, yellow, purple, turquoise and black.

If all three look-up tables are loaded with identical values, corresponding to the current state of the realtime lighting model, then each pixel can have one of these eight color values in 256 shades. This allows a surface to be covered with patterns and textures. The use of tiling patterns whose edges follow the isoparametric lines of cubic B-splines [FOLE82], [BARS82], conveys useful information in ordinary intensity shading, and in the case of encoded normal images, the user is able to see the undistorted image superimposed on the facetted lghting intensity variations which correspond to the coarse 255 normal vector direction pigeonholes available to any normal encoding scheme (the third ring from the top in figures 1 and 2 and also figure 3). The ability to distinguish the two sides of a surface is shown in figures 3, 4, 5 and 6.

The lighting model is actually a variant of the Phong approach [WHIT80], [PHON75]. The Phong [PHON75] or Blinn [BLIN77] lighting models create saturated highlights which give the object a plastic appearance [COOK81]. Non-saturated specular highlights result with the technique in this paper because, using seven colors, it is not possible to saturate the specular component independently of the ambient and diffuse components. In order to saturate one color, all three banks of the lookup tables are used. Thus it is possible to create a monochrome image using one of the seven colors, with white specular highlights; it is also possible to assign red, green and blue separately to the ambient, diffuse and specular components.

The hidden surface algorithm used is a simple Z-buffer. If, in addition to specifying one of the seven colors for each tile in a surface pattern, one also indicates whether the tile is opaque or clear, then one can remove cuts or slices from the object or punch holes or windows in it and thereby see into a complex shape. As shown in figure 6, the appropriate choice of cutouts can aid in understanding certain shapes.

LIGHTING CONTROL

Any number of light sources can be manipulated: three were chosen to make natural use of the 4 button puck of the Bitpad tablet (one button for exit from anywhere on the tablet). Pressing the i-th button (i = 1,2,3) allows the user to position the light as though it were controlled by a 2D trackball [EVAN81]. By selecting other modes, the user can dynamically control: (a) the diffuse and specular intensities of the i-th light source by pressing tablet button i and using the current (x,y) tablet values (2D slider [Evan81]); (b) the ambient lighting, the coefficient of specular reflection and the coefficient of diffuse reflection (1D sliders); (c) the three specular intensity exponent values (1D sliders).

DISGUISING THE FACETS

In order to make this scheme of realtime lighting more acceptable, it is necessary to eliminate the faceted appearance and thereby stop the user from consciously having to disregard the misleading intensity reflection patterns which are contradicting the impressions created by the tiling patterns. This is done by applying a random displacement to each unit normal vector before encoding it. The work presented here uses a Gaussian probability distribution for the perturbations and allows the user to provide an input value to determine the magnitude of the mean perturbation. The rings in figures 1 and 2 show the results of encoding normals which have been subjected to the following degrees of random perturbations (counting from the top down): 1st - none; 3rd - slight; 2nd and 3rd -moderate; 4th - heavy. It is readily apparent that a slight to moderate randomizing prior to encoding has the desired effect of eliminating the objectionable facets while creating a barely perceptible illusion of texture. In effect, a large scale texture pattern, which is confusing, has been replaced by small scale texture which ranges from the barely perceptible to the aesthetically interesting. Figure 5 shows the effect of mixing the degree of randomness. Here, banks 2 and 3 (red and blue) are highly randomized, whereas bank 2 (green) is only slightly perturbed. Figure 6, in addition to showing surface cutouts, demonstrates that a seven color palette allows for a surprising variety of surface patterns even with a simple tiling scheme. The next step, if richer textures are required, is to implement full image mapping [BLIN76] using seven colors.

ADDITIONAL COMMENTS

It would be useful to vary the size of the spot created by the light source and thereby model studio lighting as was done at GM [WARN82]. This requires knowledge of the (x,y,z) position of each pixel in order to determine whether the spotlight falls on it. To do this in realtime requires at least 8 bits per coordinate, which implies a lookup table of 24 + 24 = 48 bits! To some extent, however, the studio floodlight with its variable-sized spot can be approximated in realtime by applying a light source of low to zero diffuse intensity and then varying the specular intensity (for brightness) and the specular cosine exponent (for spot size). The higher the exponent's value, the smaller the specular highlight. The slight peculiarity of this scheme is that spot size is not a function of

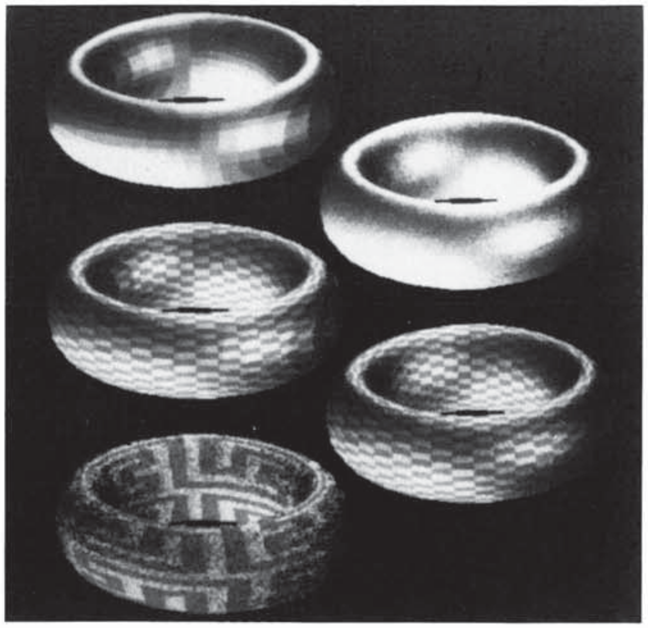

fig. 1: (x,y) encoded normals, reading from top to bottom: (a) no randomizing, (b) moderate randomizing, (c) tiling patterns and no randomizing, (d) tiling plus slight randomizing, (e) tiling plus extreme randomizing.

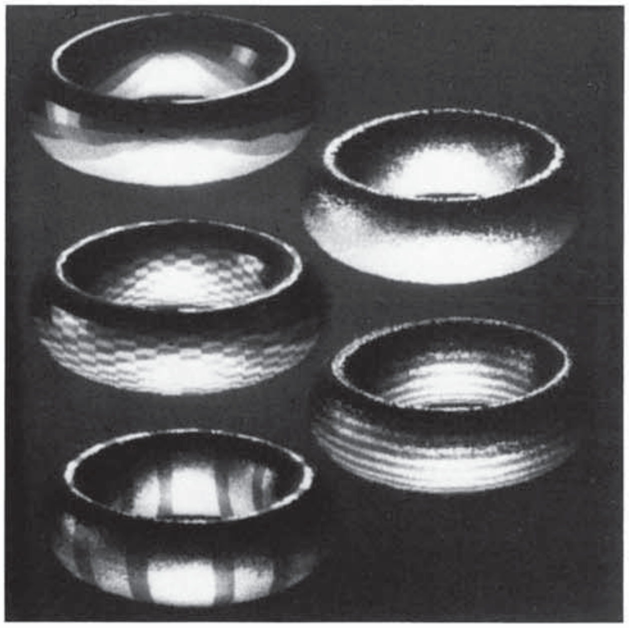

fig. 2: (θ,ϕ) encoded normals, reading from top to bottom: (a) no randomizing, (b) moderate randomizing, (c) tiling patterns and no randomizing, (d) tiling plus slight randomizing, (e) tiling plus extreme randomizing.

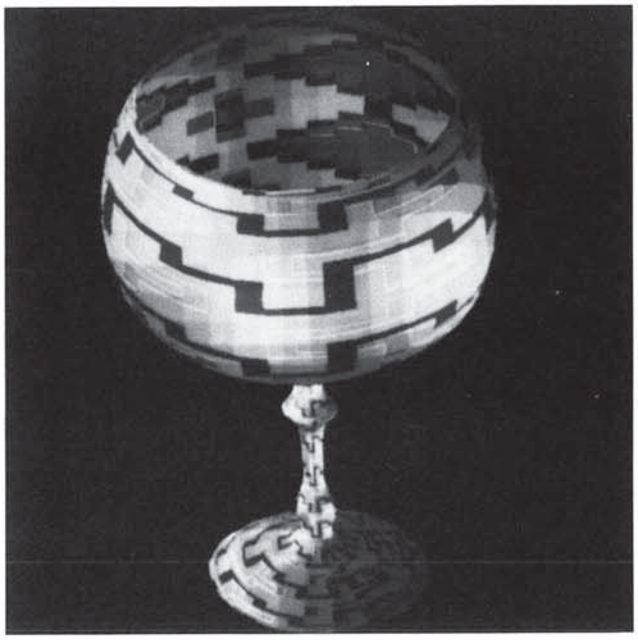

fig. 3: (x,y) encoding with no randomizing.

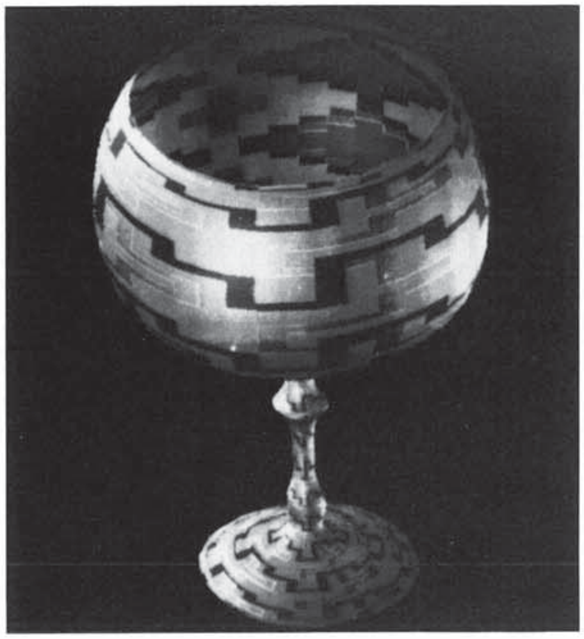

fig. 4: (x,y) encoding with slight randomizing

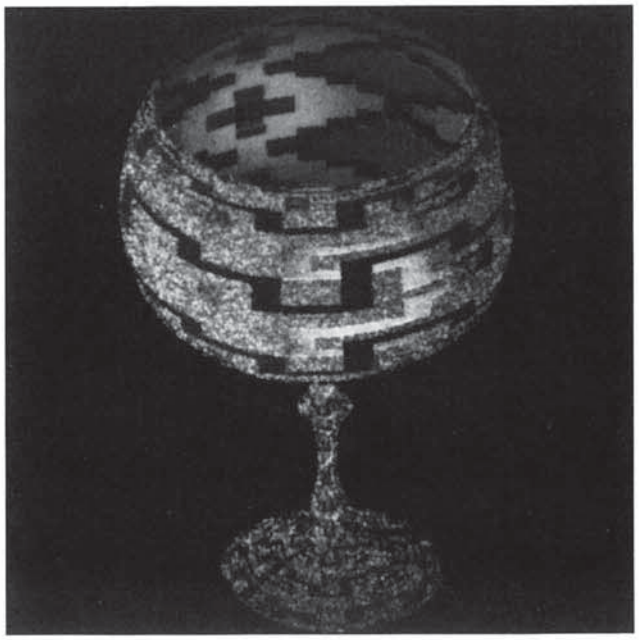

fig. 5: (x,y) encoding with slight randomizing in green, and extreme in red and blue

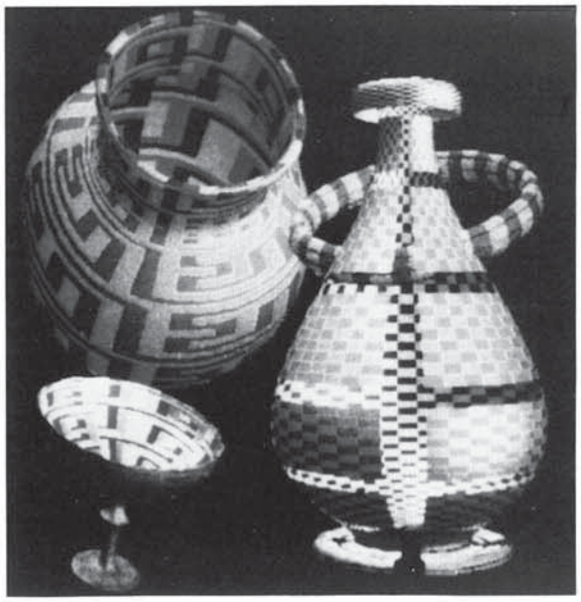

fig. 6: (x,y) encoding with various degrees of randomizing, transparent row and column cut outs in the near side of the vase and a variety of tiling patterns using three light sources

distance (all light also passes undiminished through intervening surfaces!); this does not detract from the techniques' usefulness in helping the user to rapidly comprehend a complex shape/scene.

CONCLUSIONS

Given a standard 24 bit deep RGB frame buffer structured with eight bits per bank and three 256 entry lookup tables, it is possible to encode surface normal vectors, the directions of which have been perturbed to remove the intensity facetting , and store these encoded values in the R, G, B banks so that a realtime, seven colored, multiple light source lighting model with ambient, diffuse and specular components can be run. The effect is the next best thing to a physical model generated by N/C methods or a realtime 3D shaded image generation capability, and requires less time than N/C machining and less expense than 3D hardware. Even with 3D hardware, until an entire complex scene can be generated in nearly realtime, this technique can be used to set up the optimum lighting model parameters before making a final run.

ACKNOWLEDGMENTS

Thanks are due to C.M. Penner who, in her capacity as a University of Waterloo Co-op Student worker, provided valuable software support in the early stages of this work, and to G. Bechthold and M. Duggan of the Computer Graphics Section for improvements made to our ailing prototype frame buffer.

REFERENCES

[BARS82] - B. Barsky, A. Fournier, "Computational Techniques for Parametric Curves and Surfaces", Graphics Interface'82.

[BASS81] - Daniel H. Bass, "Using the Video Lookup Table for Reflectivity Calculations: Specific Techniques and Graphics Results", Computer Graphics and Image Processing 17, 249-261 (1981)

[BLIN76] - James F. Blinn, "Texture and Reflection in Computer Generated Images", Comm. of the ACM, October 1976, Vol.19, No.10.

[BLIN77] - James F. Blinn, "Models of Light Reflection for Computer Synthesized Pictures", Siggraph '77, Vol.11, No.2, Summer '77.

[COOK81] - R.L. Cook, K.E. Torrence, "A Reflectance Model for Computer Graphics", Computer Graphics Vol.15, No.3, August 1981.

[EVAN81] - K.B. Evans, P.P. Tanner, M. Wein, "Tablet-Based Valuators That Provide One, Two or Three Degrees of Freedom", Siggraph '81.

[FOLE82] - J.D. Foley, A. VanDam, "Fundamentals of Interactive Graphics", 1982 Addison-Wesley, Chapter 13.

[FORR79] - A.R. Forrest, "On the Rendering of Surfaces", Computer Graphics Vol.13, No.2, August 1979.

[PHON75] - Bui Tuong Phong, "Illumination for Computer Generated Pictures", Comm. of the ACM, June 1975 Vol.18, No.6.

[SCHW83] - Dins Schweitzer, "Artificial Texturing: An Aid to Surface Visualization", Computer Graphics, Vol.17, No.3, July 1983.

[WARN83] - David R Warn, "Lighting Controls for Synthetic Images", Computer Graphics , Vol.17, No.3, July 1983 pp13-21

[WHIT80] - Turned Whitted, "An Improved Illumination Model for Shaded Display", Comm. of the ACM, June 1980, Vol.23, No.6.