A Tool for Demonstrating the Interaction among Lighting/Material Parameters and Potential Problems in Polygon-Based Rendering

Tin-Tin Yu, John Lowther and Ching-Kuang Shene[1]

Department of Computer Science

Michigan Technological University

Houghton, MI 49931

Email: {tiyu,john,shene}@mtu.edu

ABSTRACT

This paper presents a tool **PBR-Tutor** for demonstrating the interaction among lighting and material parameters and revealing commonly seen polygon-based rendering problems. **PBR-Tutor** is simple and easy of use, and can be used by instructors in a classroom setting and by students for self study.

1. MOTIVATION

While each illumination model provides an explicit relationship of the lighting and material parameters, the interaction among these parameters is so complex that cannot be described with simple discussions. Moreover, strange rendering problems can occur due to the nature of polygon-based rendering algorithms. Students may be confused, puzzled, or even surprised when strange results appear in their images. The well-known computer scientist Tony Hoare once said: "***You can't teach beginning programmers top-down design because they don't know which way is up.***" This also applies to computer graphics because there are so many factors involved in the generation of an image and any mistake would cause the image to become unusable. To address this issue, we must tell students which

[1] Communicating author.

way is up. More precisely, before asking students to write their programs, we have to ensure they know the relationship and the interaction among rendering parameters, and point out potential problems of polygon-based rendering algorithms. While many introductory books discuss illumination models to some degree [1, 4, 5, 6], only a few more advanced ones present the potential rendering problems [2, 8]. Hence, the authors developed **PBR-Tutor** (**P**olygon-**B**ased **R**endering **Tutor**) to demonstrate the interaction among lighting and material parameters, and reveal some commonly seen polygon-based rendering problems. With **PBR-Tutor**, students will be well-equipped and more confident before they start writing their programs, and be able to verify their understanding by doing and visualization.

2. USER INTERFACE

PBR-Tutor is designed to run on Windows, and its user interface is divided into three regions (Figure 1). The top region has 11 icons that permit the user to select a demonstration scene and various transformations; the left region has a number of slides to change rendering parameters; and, the remaining region is the drawing canvas that displays the rendered result. The drawing canvas always shows an object and a spotlight. The scene, the object, or spotlight can be translated and rotated.

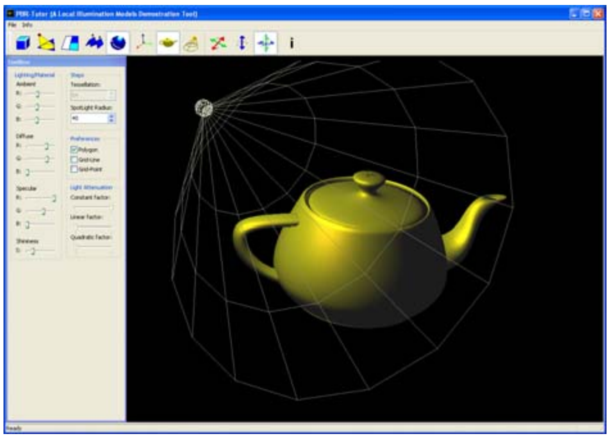


Figure 1: User Interface

The top region contains 11 icons which are divided into three groups (Figure 2). Group A has five icons for selecting a particular scene to demonstrate a rendering effect. Group B has three icons for selecting the scene, the object, or the spotlight to be transformed. Group C has three icons which are used to specify the type of transformations. Group C uses two icons to translate the selected item in the *xz*-direction and *y*-direction, and one icon to rotate the selected item. Selected icons are highlighted. Figure 2 shows that the selected item is the object and the selected transformation is rotation.

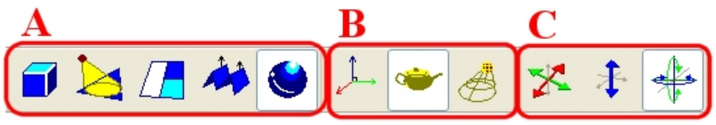
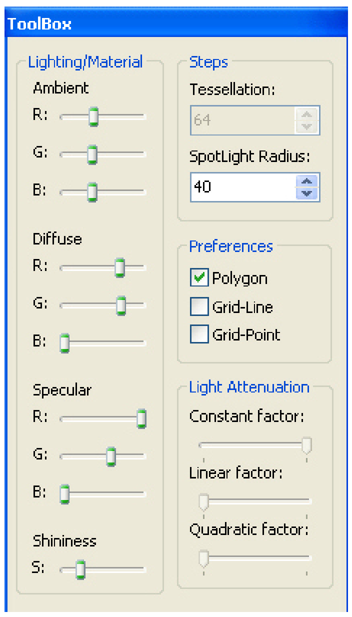


Figure 2: Tool Bar



ToolBox

Lighting/Material

Ambient
R:
G:
B:

Diffuse
R:
G:
B:

Specular
R:
G:
B:

Shininess
S:

Steps

Tessellation: 64

SpotLight Radius: 40

Preferences

[✓ Polygon]
Grid-Line
Grid-Point

Light Attenuation

Constant factor:

Linear factor:

Quadratic factor:

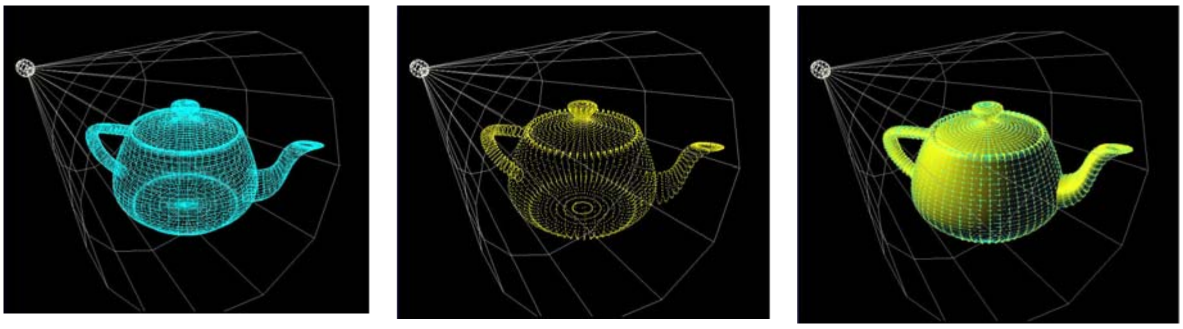
Figure 3

The slides in Figure 3 allow the user to change various rendering parameters. The Lighting/Material slides on the left column are used to modify the ambient, diffuse, specular and shininess parameters. Except for shininess, each term has three slides for the red, green and blue color components. The Light Attenuation factor f_{att} is defined as follows,

$$f_{att} = \min\left(1, \frac{1}{c_0 + c_1 d + c_2 d^2}\right)$$

Here, d is the distance between the light source and object, and c_0, c_1 and c_2 are the constant, linear, and quadratic terms, respectively.

The remaining items are for displaying purposes. The Preferences buttons allow the object on the drawing canvas to be rendered with polygons (*i.e.*, smooth shading), grid lines, grid points, or the combination of any selections. Figure 1 shows the well-known Utah teapot with smooth shading, while Figure 4 renders the same teapot with grid lines, grid points, and all three rendering selections. Permitting a user to select from polygons, grid lines and grid points is important, because many rendering problems require the knowledge of the edges and vertices of a polygon.



(a) Grid Lines (b) Grid Points (c) All

Figure 4

The upper right corner has two items (Figure 3), the number of polygons (*i.e.*, Tessellation) and the light cone size (*i.e.*, Spotlight Radius). The number of polygons plays an important role in rendering polygons, not only because of reducing angular silhouettes but also because of its impact on various odd rendered results. The light cone size (*i.e.*, the radius of the shown finite wireframe cone) also helps the user understand the interaction between light source and polygon rendering.

3. GOURAUD SHADING AND MACH BANDS

Gouraud shading is the most commonly used method for polygon rendering. It requires vertex normals. If the normal of a vertex is not available, it may be estimated by averaging the face normals of all adjacent faces of that vertex. Gouraud shading uses a vertex and its normal vector to determine the vertex intensity with the help of an illumination model. Then, each polygon is shaded using linear interpolation of vertex intensities along each edge and then between edges along each scan line.

A commonly seen problem with Gouraud shading is Mach bands. Mach bands, discovered by the Austrian physicist Ernst Mach (1838 – 1916), refer to an illusion that variously emphasizes edges or suggests edges in an image where the intensity change is smooth [3, page 29]. As a result, the common edge of two polygons would look more evident or brighter/darker than necessary. Mach bands can be very obvious even with Gouraud shading if the surface is approximated with a coarse tessellation. Hence, students may consider the presence of Mach bands a programming mistake rather than a human eye illusion. An instructor may use this tool to demonstrate and explain this effect.

For example, selecting the first icon in Group A activates the scene that demonstrates Mach bands. The drawing canvas shows a coarse tessellation of a sphere (Figure 5(a)). Mach bands are clearly seen in the highlight area (*i.e.*, the brighter edges from the center to the vertices of the highlight). See also Figure 5(b), where the spotlight is rotated so that only a portion of the sphere is illuminated. One may reduce Mach bands with a finer tessellation (Figure 5(c)). A finer tessellation may also reduce angular silhouettes as in Figure 5(c); but, rendering time and memory consumption will increase.

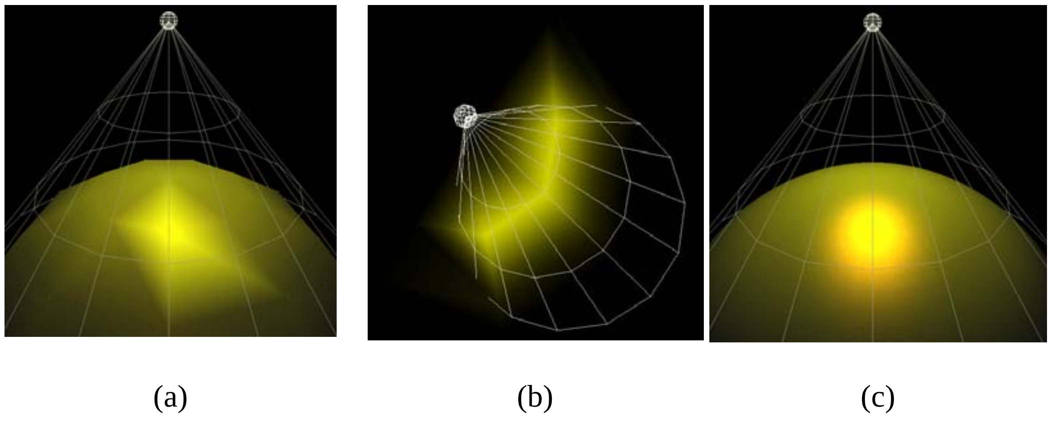
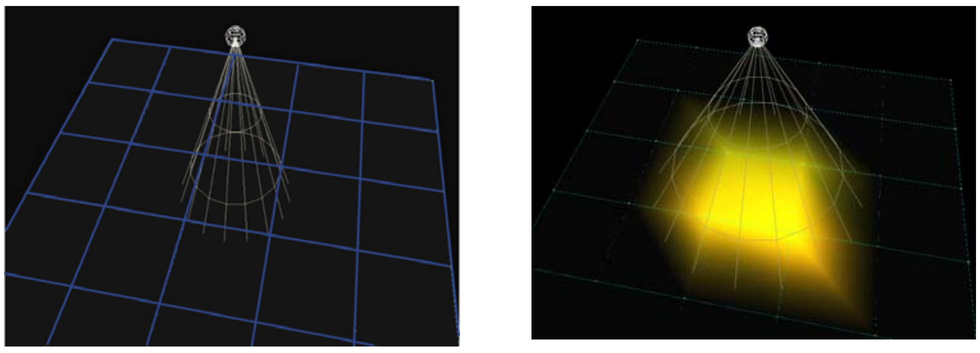

(a) (b) (c)

Figure 5

4. INTERACTION BETWEEN LIGHT SOURCES AND POLYGON VERTICES

Since Gouraud's method computes the vertex intensity and uses linear interpolation to generate colors along each edge from which the interior of a polygon is shaded, a polygon will not receive illumination if the light source misses its vertices. This may cause problems if students use spotlights.

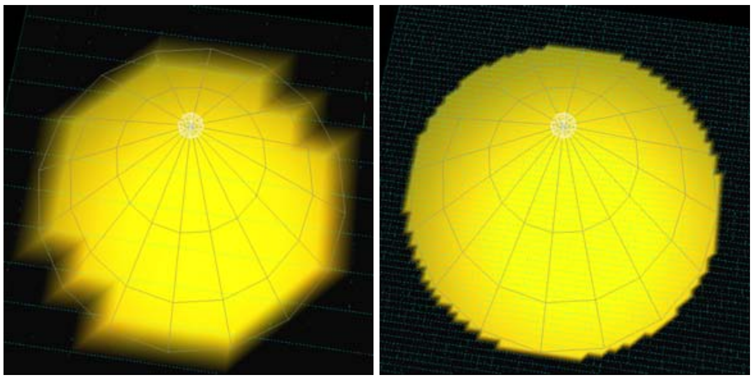
By selecting the second icon of Group A, one sees a floor and a spotlight whose small light cone misses all vertices in the scene. Hence, none of the rectangles receives illumination, and all of them are black (Figure 6(a))! If the light cone is expanded a little so that the four vertices of the center rectangle are illuminated, the rectangle is shaded properly (Figure 6(b)). Note that Mach bands are clear along the rectangle edges and some diagonals (*i.e.*, non-edges). The user may translate or rotate the light source or change the light cone size to gain more experience.



(a) (b)

Figure 6: Interaction between Vertices and Light Source

One may use this scene to demonstrate the impact of tessellation. Figure 7(a) is the same as Figure 6 but with coarse tessellation. The spotlight does not generate a smooth circular area it illuminates. By making the tessellation finer, the disk becomes smoother as in Figure 7(b). However, its zigzag boundary cannot be completely eliminated.



(a) Coarse Tessellation (b) Fine Tessellation

Figure 7: Impact of Tessellation

5. POLYGONS WITH UN-SHARED VERTICES

If adjacent polygons do not share a vertex on the common edge, shading discontinuity may occur. Figure 8 has three rectangles I, II and III. Suppose rectangle I is defined by vertices *A*, *B*, *F* and *G*, rectangle II by vertices *B*, *C*, *D* and *H*, and rectangle III by vertices *H*, *D*, *E* and *F*. Thus, vertex *H* is not shared by rectangles I and II or by I and III, even though it is on the common edge *BF*. In this case, shading discontinuity will occur at *H* and along the common edge.

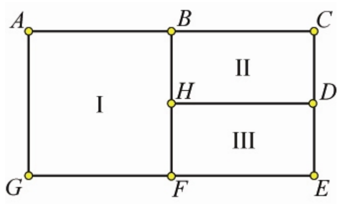
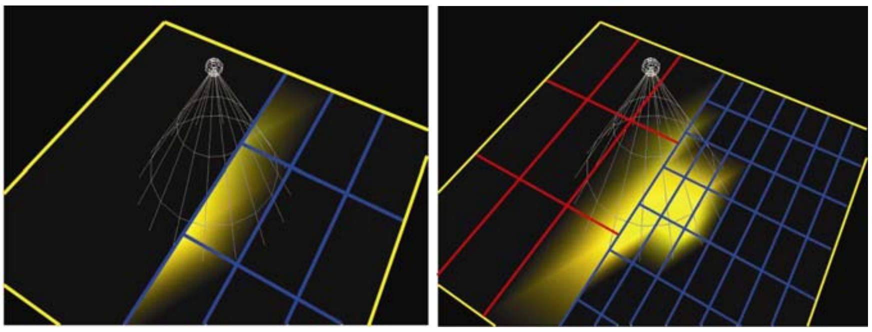


Figure 8: Polygons with Un-Shared Vertices

The cause of shading discontinuity is easy to explain. When dealing with rectangle I (Figure 8), if *B* and *F* are not illuminated, edge *BF* is black, and all scan lines between edges *AG* and *BF* are also black near edge *BF*. On the other hand, when dealing with rectangle II, if *H* is illuminated, the area surrounding *H* in rectangle II is bright. This causes the neighborhood of *H* to be shaded differently in different rectangles, and, hence, shading discontinuity occurs.

Selecting the third icon of Group A activates the demonstration of shading discontinuity. The scene contains a coarse tessellation and a spotlight whose light cone only illuminates the two "interior" vertices on the common edge (Figure 9(a)). As a result, the left rectangle is black because none of its vertices receives illumination. The user may make the tessellation finer and/or the light cone larger, or even rotate or translate the spotlight to see other discontinuity. As long as there are unshared vertices on common edges, shading discontinuity may appear. Figure 9(b) is an example of finer tessellation.



(a) (b)

Figure 9: Shading Discontinuity

6. VERTEX NORMAL INACCURACIES

Vertex normals are approximated by the average of the normals of all adjacent faces. This can lead to flat shading even though the object is not flat. For example, Figure 10 shows an M-shaped surface with face normals (solid). If the vertex normals (dashed) are computed by averaging, all would point upward and are parallel with each other. As a result, if the light source is far enough, the object will look flat without shading variation.

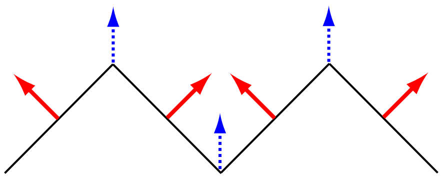


Figure 10: Averaging Normal Vectors

By selecting the fourth icon of Group A, the impact of averaging face normals may be demonstrated. The scene contains columns of "wavy" rectangles seen in Figure 10. Thus, all estimated vertex normals are parallel to each other, and the object looks very flat (Figure 11(a)). In fact, without the grid lines in Figure 11(b), the shaded result would look like a planar one as in Figure 6 and Figure 7.

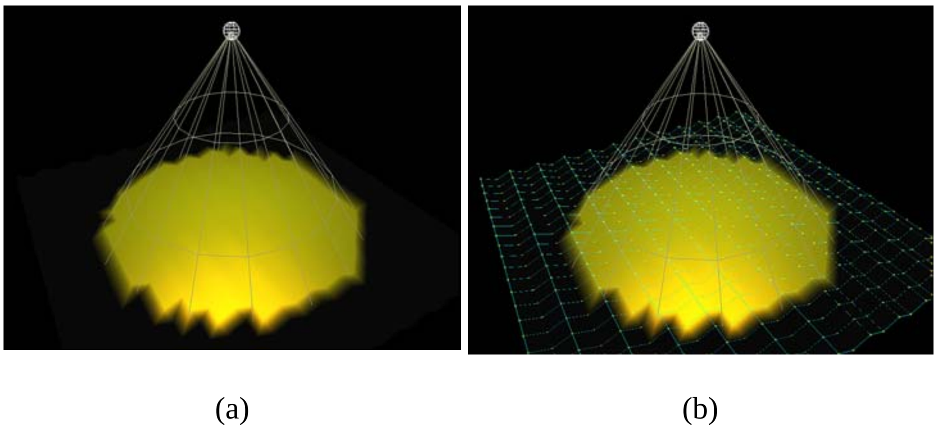
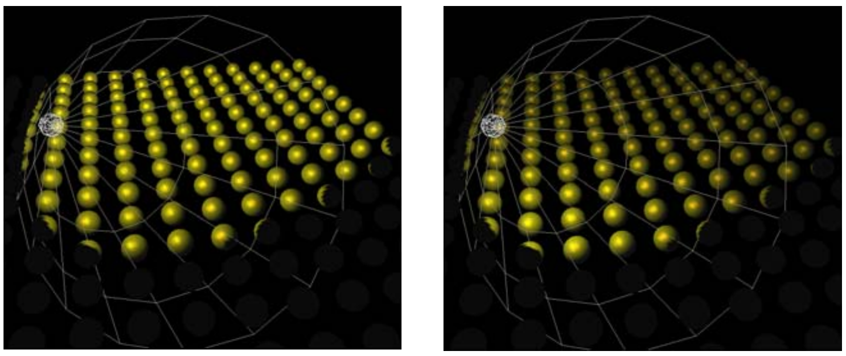


Figure 11: Shaded "Wavy" Polygons

7. LIGHTING AND MATERIAL PARAMETERS

The fifth icon of Group A activates this demonstration. The drawing canvas shows the Utah teapot and a spotlight as shown in Figure 1. If the object icon (the second icon of Group B) is selected, the slides are used to modify the Ambient, Diffuse, Specular, and Shininess *material* parameters. The Tessellation item is disabled since the Utah teapot has a fixed number of vertices, edges and faces. If the light icon (the third icon of Group B) is selected, these slides are used to modify *lighting* parameters. Additionally, one may also change the light attenuation coefficients and the radius of the spotlight; however, the Shininess slide is disabled.

Right click to show a matrix of spheres for attenuation factor experiments. Figure 12(a) shows the result of $c_0 = 1$ and $c_1 = c_2 = 0$ (Section 2), and there is no attenuation since the linear and quadratic factors are zero. Figure 12(b) is the result of $c_0 = 0$, $c_1 \approx 0.3$ and $c_2 \approx 0$, and spheres farther away are darker than those closer to the light source. Instructors may use this demonstration to discuss the interaction between lighting and material parameters, and students may gain experience in playing with these parameters before writing their programs.



(a) $c_0 = 1, c_1 = c_2 = 0$ (b) $c_0 = 0, c_1 \approx 0.3, \ c_2 \approx 0$

Figure 12: Light Attenuation Factors

8. EXPERIENCE AND CONCLUSIONS

PBR-Tutor grew out as an exercise in the authors' computer graphics related courses. In the past decade, the authors used local (*i.e.*, OpenGL) and global (*i.e.*, ray tracing) illumination models, extensively. But, the interaction among lighting/material parameters and polygon rendering can be very complex, and, in many situations, it is difficult to explain precisely what the effect is or is not. **PBR-Tutor** is designed to avoid this confusion and to ensure the effects be precisely communicated to students via a learning-by-doing and visualization approach. **PBR-Tutor** has been used for a number of years in some graphics related courses. The main use is demonstrating the impact of lighting/material parameters and explaining some major problems in polygon rendering so that with this visual understanding students will be able to use lighting/material parameters properly and to avoid the problems in polygon rendering. The impact of **PBR-Tutor** has been very positive because many students felt that they have never thought about polygon rendering problems and in many cases considered them as programming mistakes. **PBR-Tutor** is used in the discussion of the terms in illumination models and in comparisons between local illumination using polygon-based rendering and global illumination (*e.g.*, ray tracing [7] and photon mapping [9]). Students were also asked to write a similar program to demonstrate as many effects and problems as possible. In general, no student would miss any problems discussed in this paper and some uncover more in their implementations. Some students reported that this demonstration/exercise helps them understand lighting/material parameters and recognize the problems of polygon-based rendering. In summary, the authors believe **PBR-Tutor** is a useful tool for instructors to use in classroom and for student self study. Interested readers may find **PBR-Tutor** and other useful components at the author's site `www.cs.mtu.edu/~shene/NSF-2`.

ACKNOWLEDGMENT

The work is partially supported by the National Science Foundation under grant DUE-0127401 and an IBM Eclipse Innovation Award 2003.

REFERENCES

1. Edward Angel, *Interactive Computer Graphics: A Top-Down Approach Using OpenGL*, fourth edition, Addison Wesley, 2006.
2. James D. Foley, Andries van Dam, Steven K. Feiner and John F. Hughes, *Computer Graphics: Principles and Practice*, second edition, Addison-Wesley, 1990.
3. Andrew S. Glassner, *Principles of Digital Image Synthesis*, Volume 1, Morgan Kaufmann, 1995.
4. Donald Hearn and Pauline Baker, *Computer Graphics with OpenGL*, third edition, Pearson/Prentice Hall, 2004.
5. Francis S. Hill, Jr. and Stephen M. Kelley, Computer Graphics Using OpenGL, third edition, Pearson/Prentice Hall, 2007.
6. Jeffrey J. McConnell, *Computer Graphics: Theory into Practice*, Jones & Bartlett, 2005.
7. Ching-Kuang Shene, Raytracing as a Tool for Learning Computer Graphics, *ASEE/IEEE Frontiers in Education*, Vol. III, 2002, pp. (S4G-7)-(s4G-13).
8. Alan Watt and Mark Watt, *Advanced Animation and Rendering Techniques: Theory and Practice*, The ACM Press, 1992.
9. Tin-Tin Yu, John Lowther and Ching-Kuang Shene, Photon-Mapping Made Easy, *ACM 36th Annual SIGCSE Technical Symposium*, 2005, pp. 201-205.