Strengthening Visual Skills by Recognising Rendering Algorithms

Rosalee Wolfe
DePaul University
Chicago, Illinois
wolfe@cs.depaul.edu

Abstract

Computer science students contemplating a career in graphics need to develop a visual sense, but traditional course topics do not meet this need. Visual analysis is a teaching technique developed for computer science instructors that helps impart this ability. Through the use of a few visual cues, students learn to visually identify surface algorithms, shaders and lighting techniques. An interactive software package called TERA (Tool for Exploring Rendering Algorithms) provides nearly a million image combinations that students can use to practice their visual identification skills.

Keywords: *algorithm visualisation, computer graphics education.*

1. Background

For a successful career in computer graphics, computer science majors need to develop a visual sense in addition to their technical knowledge. Computer science graduates hired by the graphics industry will work side-by-side with artists, and it is essential for effective communication that they have an appreciation and enthusiasm for the visual aspects of the field. In addition to programming skills and knowledge of operating systems, recent labor market analyses list "strong visual style sense" and "understanding how artistic elements work with technical elements," as desirable traits for programmers and technical directors [1,2]. An increasing number of graphics houses want to see a demo reel or a portfolio from computer science graduates who count Unix shell scripting, C++ programming and the mathematical derivation of NURBS in their skill set.

There are many exemplary resources [3, 4] and learning tools that allow students to examine aspects of geometry [5, 6, 7, 8, 9, 10], rasterisation [6, 11], illumination [6, 12, 13, 14] and the entire 3D rendering pipeline [13]. These present visualisations that a student can use to explore and study an algorithm in isolation. This is a sound pedagogical approach for a first introduction, but eventually a student will need to understand an algorithm's behavior in co-operation with other algorithms and in the context of a finished image.

Unlike other specialities in computer science, the choice of a graphics algorithm usually cannot be based solely upon space, speed and implementation considerations. Equal in importance to time and memory requirements is the visual effect that a graphics algorithm produces. In fact, visual appearance will be the overriding factor for some applications. Visual understanding is essential to developing and debugging new algorithms. For example, when writing a shader, a programmer looks at a rendered sample, analyses how its appearance differs from the ideal and uses the information thus gained to refine the shader. An essential aspect to developing a visual sensibility is the ability to identify and compare rendering algorithms. Visual knowledge can prepare students to answer such real-life questions as, "Algorithm X gives us exactly the effect we want, but it is prohibitively expensive. Can we get a similar effect by tweaking our implementation of Algorithm Y?"

Although the graphics industry desires a developed visual sensibility in new computer science graduates, there is no place in the traditional curriculum where students can gain this knowledge. The discipline of computer science draws on mathematics and emphasises algorithm study and development, which are almost entirely text-based and involve little visualisation. Computer science instructors often view the prospect of discussing the visual aspects of computer graphics as a daunting, if not overwhelming endeavor. Typical reactions include

I'm not an artist! How am I supposed to gain the requisite background to teach this?

I don't have time for this in class – I have too many algorithms to cover as it is.

Both reactions are understandable and justifiable. Computer science instructors are not artists and their goal does not include producing artists. Further, there is an enormous amount of technical knowledge in the computer graphics discipline, and attempting to decide what topics belong in an introductory course has been the topic of many papers. Any approach to the problem of inculcating a visual sensibility has to be one that does not take up much lecture time and constitutes a teaching method that computer science instructors are willing to try. This paper proposes that *visual analysis* is one approach that fits these constraints.

2. Visual Analysis

Visual analysis is a teaching technique developed specifically for computer science instructors that imparts this knowledge [15]. The technique stems from critical analysis, which establishes a structure for examining works of art [16]. Students in the visual arts learn to describe and compare works in terms of design, concept and media.

Instead of design, concept and media, visual analysis teaches students to recognise a small number of visual cues, including:

- visibility of polygon faces or polygon edges
- transition from light to dark on diffuse-reflecting surfaces
- color and shape of specular highlights
- presence of transparency, reflection, refraction, patterns and textures

1. First, determine the surface algorithm.	
Appearance	*Algorithm*
Outlined polygons. Object's far sides are visible. Uninterrupted horizon lines.	Wireframe
Outlined polygons. An object's far sides are not visible. Occluded objects are invisible.	Hidden-line removal
Filled-in polygons. Presence of refraction in transparent objects. (Presence of reflection, shadows are also helpful)	Ray tracing
Filled-in polygons. No refraction, reflection or shadows.	Z-buffer
2. Determine the shader.	
For Z-buffer	
One color per object. Objects appear flat, as if cut from paper.	Constant
One color per polygon. Objects now appear to have a shaded contour.	Faceted
Smooth transitions from light to dark. Specular highlights follow polygon edges.	Gouraud
Smooth transitions from light to dark. Specular highlights are white, compact and elliptical. "Shiny plastic" look.	Phong
For ray tracing	
Opaque, colored surface. No highlight.	Diffuse
Opaque, colored surface with highlight.	Diffuse and Specular
Transparent object that appears to bend light.	Transparency
Part of scene is visible in the surface of the object.	Reflection
3. Determine additional surface interest (both z-buffer and ray tracing)	
Image appears to have been pasted or glued onto object.	2-D texture mapping
Object appears to have been carved from a solid substance like stone or wood.	3-D texture mapping
Object surface appears rough or wrinkled, but its profile is smooth.	Bump mapping
4. Light sources	
Harsh shadow / lighter shadow.	Low/high ambient
Sharply defined shadow.	Point light
Presence of a penumbra, but degree of shade is constant within penumbra.	Area light
Color bleeding; deepening of shadow in corners, under furniture.	Radiosity

Table 1: Visual cues and algorithms (visual equivalencies omitted).

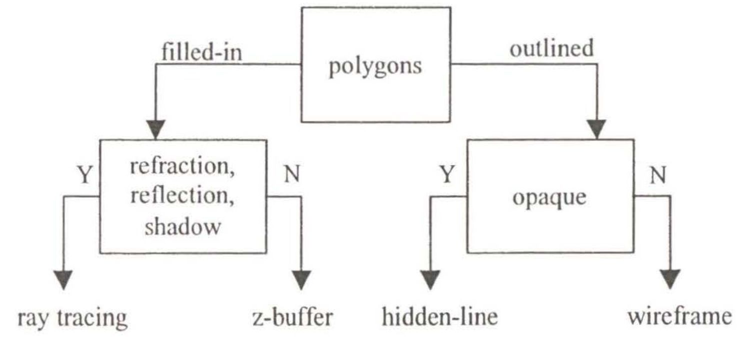


Figure 1: Three visual cues identify four surface algorithms.

- sharpness of shadow
- interactions between adjacent diffuse reflectors (color bleeding)

These cues are usually sufficient to identify a rendering algorithm as Table 1 demonstrates. Informal studies have shown that both instructors and students find this list of cues non-threatening and easy to spot in an image. By learning to observe and describe these cues, students are able to identify rendering algorithms.

3. Classroom Presentation

In the first lecture, the teacher discusses images portraying three or four algorithms that produce starkly different effects. For example, the four surface algorithms of wireframe, hidden-line, ray tracing and z-buffer create distinctive effects and can be distinguished by the use of only three visual cues. See Figure 1. Each week, the teacher adds more algorithms, and by the midterm, the class has examined all commonly used rendering algorithms. Until the midterm, the teacher emphasises the characteristic visual behavior of the algorithms. After the midterm, the teacher shows how multiple algorithms can achieve equivalent effects. By the end of the course, students learn to identify the surface algorithm, shaders and types of light sources.

While discussing a visual cue, the teacher presents two or three images that demonstrate it. After explaining the visual cue, the teacher gives the name of the algorithm that creates the cue. The teacher then shows a short series of images and invites students to identify the visual cues and then suggest a rendering algorithm. Students should specify the cues first so that they learn to spot them in the context of an image. Any premature guesses as to the identity of the rendering algorithm are met with the response, "Cues first!".

After the first class meeting, discussions on visual analysis begin with a short series of images that review the rendering algorithms from the previous meeting. Students first name the visual cues and then suggest a rendering algorithm.

The methodology requires only five minutes of an hour-long lecture. Such a small amount of time will not significantly impact the presentation of other topics, especially if the teacher covers visual analysis during the last five minutes, when student focus is beginning to wander. In fact, there is a way of presenting this material without using any lecture time at all, as will be discussed in the next section.

4. Study Materials

Students need to practice visual analysis outside of the classroom. While posting images on the Web or pointing out examples in textbooks is a start, students say that they benefit more from the question-and-answer sessions held at the end of a lecture. Simply looking at images does not promote active learning.

Having students learn a rendering package helps a bit because students can choose parameters and view the resulting images. However it takes a significant amount of time to learn a package, and since the students have a priori knowledge of the surface and shading algorithms, this approach does not provide a student with a means of self assessment. A better approach is to provide students with an easy-to-use interactive tool that can demonstrate any rendering or shading algorithm while encouraging active learning.

TERA (Tool for Exploring Rendering Algorithms) is an interactive program that facilitates comparative study of the visual effects of rendering algorithms [17]. See Figure 2 for an annotated screen dump of TERA.

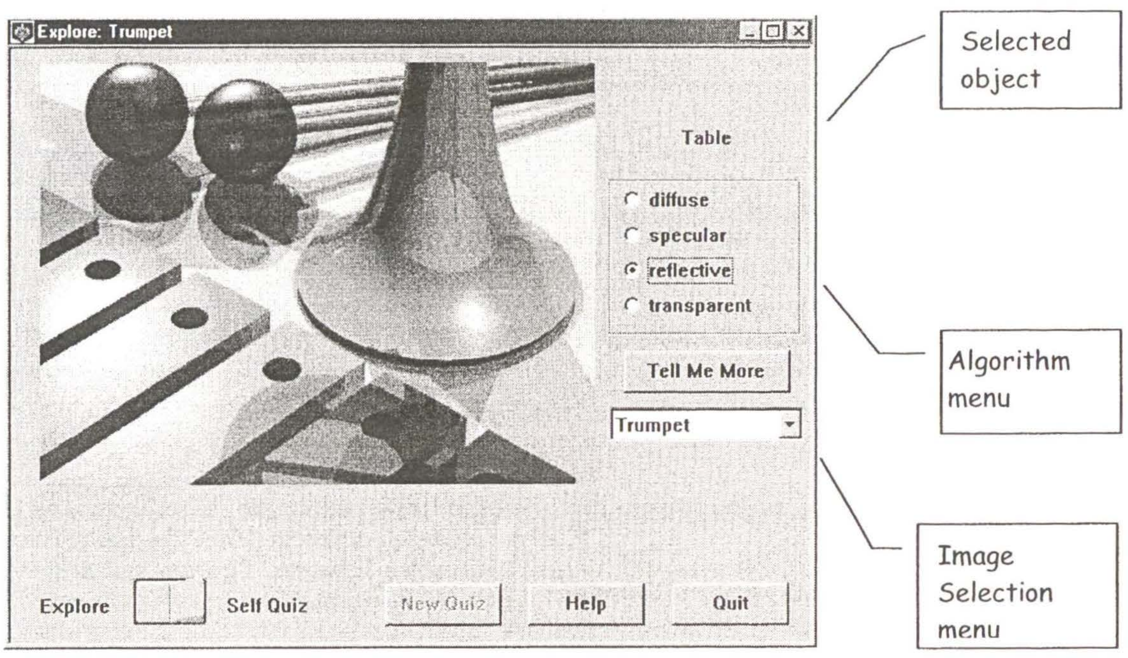


Figure 2: A sample TERA session.

A student can choose a scene and select a rendering algorithm for any object in the scene. Students can practice visual analysis using TERA. In "Self Quiz" mode, students select a scene, and TERA presents it with each object rendered by a random algorithm. Students then guess the rendering algorithm for each object in the scene. TERA responds with "Correct", "Try Again" or "Close Enough".

The "Close Enough" response is for those cases where multiple algorithms produce similar visual effects. For instance, Gouraud and Phong shading produce similar effects when no highlight is present.

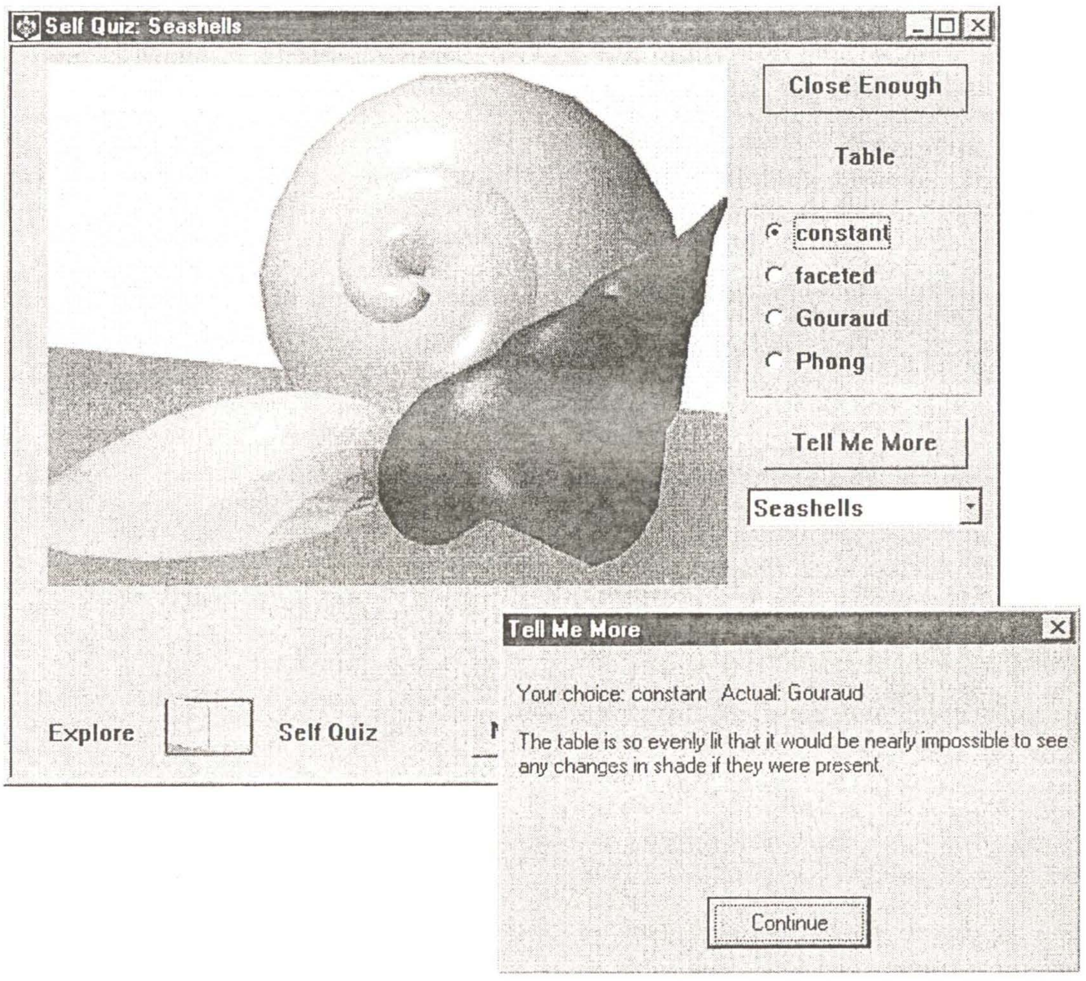
Always available is the "Tell me more" button, which provides specific feedback about a student's last algorithm selection. When a student is in "Explore" mode, the "Tell Me More" button will activate a popup window describing the relevant visual cues. In "Self Quiz" mode, when students get a response of "Close Enough", they can click on "Tell Me More" for an explanation. A pop-up window will list the algorithm they picked, the actual algorithm and a specific explanation of why the two algorithms produced effects that are visually equivalent. For example, if a flat surface is very evenly lit, then Gouraud shading may produce variations in shade that are so subtle that the result looks like constant shading. In this situation, if a student picks "constant" and receives the "Close Enough" response, the "Tell Me More" button will activate a pop-up window containing the relevant explanation. See Figure 3.

The new version of TERA is capable of creating nearly a million images for students to analyse. The images cover surface algorithms, z-buffer shaders, ray tracing shaders, texture mapping, bump mapping, lighting and visual equivalencies.

Some teachers do not discuss visual analysis in the classroom, but give a short demonstration of TERA and tell students that TERA is available in the lab. TERA has enough appeal that students are drawn to it, and they spend enough time with it that they can score reasonably well on a visual identification test [18].

5. Results and Feedback

In my experience, visual analysis adds excitement and a sense of the "big picture" to introductory graphics courses. Students may not be able to implement every rendering algorithm when they leave the course, but they will be able to recognise them and know their names, which provides a starting point for further investigation.



Your choice: constant Actual: Gouraud

The table is so evenly lit that it would be nearly impossible to see any changes in shade if they were present.

Figure 3: TERA in "Self Quiz" mode.

A visual sensibility, fostered by visual analysis, enhances a depth of knowledge of those rendering algorithms that students do implement. Because they are already familiar with the visual behavior of the algorithm, they have expectations of how an image should appear and can perform more of their debugging on their own without outside help. Students no longer ask, "Is it right?" They state, "There's something wrong with my highlight."

A very exciting development is the favorable feedback from digital artists. This approach promises to help art students "grasp the differences between light models, surface algorithms and shaders" [19]. Further work includes the exploration of developing the approach to expand the possibilities of better communication between digital artists and computer scientists.

References

[1] Regan and Associates. A Labor Market Analysis for the interactive Digital Media Industry: *Opportunities in Multimedia.* Sunnyvale, CA: North Valley Private Industry Council (NOVA), 1997.

[2] Public Affairs Coalition of the Alliance of Motion Picture and Television Producers, *Making Digits Dance: Visual Effects and Animation Careers in the Entertainment Industry.* Sunnyvale, CA: NOVA Private Industry Council, 1997.

[3] Owen, G. Using Hypermedia to Teach Computer Graphics. Presented at the Graphics and Visualization Workshop in *Eurographics '93* Barcelona, September, 1993.

[4] G. S. Owen. Integrating World Wide Web

Technology into Courses in Computer Graphics and Scientific Visualization. *Computer Graphics* 29(3) pages 12-14, August 1995.

[5] D. Schweitzer. Designing Interactive Visualization Tools for the Graphics Classroom. In *Proceedings of the Twenty-third SIGCSE Technical Symposium on Computer Science Education.* pages 299-303, 1992.

[6] P. Min. www.cs.princeton.edu/~min/cs426/applets.html

[7] A. Shabo, M. Guzdial and J. Stasko. Addressing Student Problems in Learning Computer Graphics. *Computer Graphics* 29 (3) pages 38-40, August 1996.

[8] R. Klein, F. Hanisch, W. Strasser. Web-Based Teaching of Computer Graphics: Concepts and Realization of an Interactive Online Course. *Computer Graphics Annual conference Series, Conference Abstracts and Applications.* pages 88-93, 1998.

[9] Y. Zhao, J. Lowther and C. Shene. A Tool for Teaching Curve Design. In *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education.* pages 97-101, 1998.

[10] Y. Zhou, Y. Zhao, J. Lowther and C. Shene. Teaching Surface Design Made Easy. In *Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education.* pages 222-226, 1999.

[11] A. C. Naiman. Interactive Teaching Modules for Computer Graphics. *Computer Graphics* 30 (3) pages 33-35, August 1996.

[12] Ansari, M. Simple Illumination Visualization. Department of Computer Science, DePaul University, Chicago, 1993.

[13] E. Wernert. A Unified Environment for Presenting, Developing and Analyzing Graphics Algorithms. *Computer Graphics* 31 (3) pages 26-28, August 1997.

[14] D. Gould, R. Simpson and A. van Dam. Granularity in the Design of Interactive Illustrations. In *Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education.* pages 306-310, 1999.

[15] A. Sears and R. Wolfe, Visual Analysis: Adding breadth to a computer graphics course. In *Proceedings of the Twenty-sixth SIGCSE Technical Symposium of Computer Science Education.* pages 195-198, March 1995.

[16] R. Arnheim. *Art and Visual Perception: A Psychology of the Creative Eye.* 2nd ed. Berkeley: University of California Press, 1974.

[17] R. Wolfe, A. Sears. TERA: an interactive Tool for Exploring Rendering Algorithms. *Journal of Computing in Small Colleges* 10 (4) pages 41-46, February 1995.

[18] R. Wolfe, S. Grissom, T. Naps and A. Sears. A Tested Tool for Teaching Computer Graphics. *Journal of Computing in Small Colleges* 12 (2) pages 70-77, November 1996.

[19] D. Eber. Personal communication, December 1997.