

Graphical Problem Solving and Visual Communication in the Beginning Computer Graphics Course

Steve Cunningham
California State University Stanislaus
Turlock, CA 95382 USA
rsc@cs.csustan.edu

Abstract

The beginning computer graphics course can teach much more than just computer graphics. It can also provide an excellent introduction to graphical problem solving and visual communication, and in so doing can be an excellent complement to other computer science courses that teach more analytical problem solving. This paper describes the graphical problem-solving and visual communication contexts and discusses how they can be supported by the beginning computer graphics course.

1 Problem solving in the computer science curriculum

A graduate of our program recently visited campus to recruit other students, and told us “In my group, we want to hire people with problem solving abilities. We have found that task-oriented people just want the answer and seem to stumble when they run into a problem, while problem-solving people tend to want to know what's causing the problem so they can come up with a solution. The problem solving I learned as a student gave me the skills to tackle increasingly complex problems and move up the ladder in my company.”

Those of us who teach computer science take great pride in the fact that our students learn the kind of problem solving described above by our graduate. We use textbooks for our beginning courses with titles such as *Programming and Problem Solving in C++* (or C, or Java, or once upon a time Pascal, or ...) and we put significant emphasis on analyzing problems, breaking them down into component parts, and designing solutions with various kinds of programming tools. We then rely on the problem solving skills that students learn in our early courses as we teach more advanced courses, and our students rely on these skills when they graduate and go into advanced studies or professional employment. In many ways, one might say that problem solving in a computational setting is perhaps the most important value-added part of a computer science education.

Problem solving is a much more extensive subject than just the approach we cover in our curriculum, however. In [13]. Robert Root-Bernstein, of the Center for Integrative Studies at Michigan State University, describes twelve tools of thought that help us use our imagination to observe, analyze, and explore as we solve problems. As quoted and somewhat expanded in [12], these tools are:

1. *Pattern Seeking*. The artist as well as the scientist places great value in seeing patterns and hidden structure where others do not see any. We search for patterns to discover principles and reveal information.
2. *Pattern Forming*. By creating patterns and by bringing ideas and images together we see our thinking more clearly. Scientific visualization is based on forming patterns with the data to be able to see new relationships and new patterns.
3. *Analogizing*. The ability to see how an idea in one domain or subject area can be related or transferred to another domain or subject can help us understand the concept or a new relationship. Thinking of DNA as “unzipping” helps related common objects to invisible structures.
4. *Abstracting*. To abstract is to eliminate all the unnecessary details from our perception and reveal the basic principle or structure.
5. *Mental Visualization*. Seeing with the mind's eye is our most direct way to create images of what we are thinking about. Trying to picture something helps us to work out its structure. Some people are able to animate their visualizations and see how they move.
6. *Physical Modeling*. Our ideas need a three dimensional form because it is so hard to think in three dimensions. Sculptors do it better than anyone else. Architects and molecular biologists must design with space in mind for new buildings or new molecules.
7. *Kinesthetic thinking*. Some people simply need to move when they think or are learning. Many learning skills are imbedded in muscle memory. Driving a car, using hand tools, throwing pots on a wheel, and computer interactivity build on the motion memory that is being built up in our bodies.
8. *Aesthetics*. We are often caught off guard by the sudden recognition of the beauty of an image, view, or experience. Beauty has a wholeness and an emotional feel that commands all of our attention. What do we see when we are moved by beauty?

9. *Playacting or Internalizing* (imagining oneself as an object). Identifying with an object or concept helps us to know it better and begin to imagine interacting with it.
10. *Manipulative Skill* (hand-eye coordination). By working with materials, by sketching, and by building models we derive knowledge through our senses.
11. *Playing* (experimenting: trial and error). Sometimes, as a way of learning, we must give ourselves permission to try something at which we may fail. Play and humor help us feel more comfortable with what we don't know so we can explore what we may find.
12. *Transformational Thinking*. Change is a part of us and we transform as we grow and so does the world. The dynamics and forms of change are important to understand. What is the sequence? Where does it begin? What are the critical points along it?

The problem solving approach used in most computer science courses is very strongly focused on the symbolic and analytical approach that is emphasized by our programming languages. These are, in fact, a subset of the problem solving techniques described above, with computer science focusing on *abstracting* (using symbols to represent quantities or objects), *pattern seeking* (identifying parts of the problem that are familiar), and *pattern forming* (writing re-usable functions or classes). The problem solving approach called *patterns* (see, for example, [1] or [9]) that is proving its value in programming courses also takes advantage of *analogizing* by giving students examples of many patterns and having them create programs (solve problems) by applying analogous patterns. But while individual instructors may create richer problem solving environments, these typical computer science approaches usually do not include the additional problem solving techniques noted by Root-Bernstein.

In addition to listing these 12 problem-solving techniques, Root-Bernstein goes on to say that these tools are not useful unless the ideas can be communicated by yet another set of skills:

- verbal (written) communication,
- mathematical communication,
- visual communication,
- aural (sound) communication, and
- kinesthetic-tactile communication.

These methods of communication are to be considered for both receiving and expressing information.

Our students have the same communication experience as other students in the university: verbal, written, and mathematical communication. Visual communication is simply not part of their general education pattern. We use little visual communication in our teaching except for the diagrams that help students see the patterns of our processes; our students use almost no visual communication in expressing their learning. And yet we know the value of good visual communication, because the world around us is filled with it. We see carefully-presented diagrams in our textbooks and in online course materials; we see extraordinary presentations of scientific principles

and processes in popular scientific publications and on educational television; we see sophisticated graphic design all around us in commercial and entertainment areas. We swim in a sea of visual communication but we do not teach our students to create it themselves. We need to help our students learn to use effective visual communication—something much better than clip art and canned animations in PowerPoint—in their professional work.

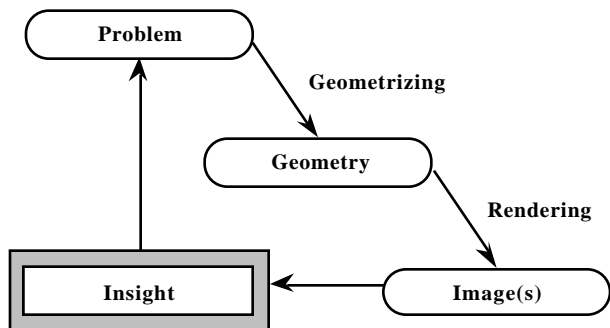
2 Graphical problem solving

These aspects of problem solving and communication are remarkably well supported by computer graphics. Computer graphics relies on traditional analysis, of course, so *pattern seeking* and *pattern forming*, *analogizing*, and *abstracting* are certainly present. The kinds of patterns, analogies, and abstractions are different in computer graphics, however, because they need geometric thinking and reasoning, and the tool domain is different, because it involves light, surface, space, and motion. In general, graphical problemsolving may be thought of as using images or graphical representations of a problem to gain insight into a problem or its solutions. The space for graphical problem solving can readily go beyond three dimensions, involving time and color as two additional dimensions that can be brought into play for modeling a problem and its solution; in this, computer graphics adds problem modeling capabilities that are not matched in any other field of study.

There are significant ways that computer graphics involves other problem solving techniques. Because we create images, students must think in terms of the images they want to design, so there is a great deal of *mental visualization* that must be learned. Because our images move—thanks to the advances in graphics capabilities that are now available to us at modest cost—students are now able to think in *kinesthetic* terms, sometimes actually using physical models to see analogies and physical motion to see how to create similar motion behaviors in their images. In communicating their problem solving, students must learn about *aesthetics* and the way it supports visual thinking in order to use the subtler aspects of an image to support their communications. And there are two other capabilities that are increasingly available to undergraduate courses: the ability to use rapid prototyping tools to create physical versions of our graphical objects allows us to use *physical modeling* to participate in the solution verification phase of problem solving (for a particularly innovative approach to this, see [3]), and the ability to include sound in animations or slide shows to support narration or data-driven sound for communication. Sound is becoming simple to create and integrate, and rapid prototyping tools show signs of becoming affordable.

Overall, the problem-solving process is often described with a feedback loop, where successive refinement of trial solutions lead to better and better solutions. For problem solving involving computer graphics, this loop is described in the figure below. Traditional computer graphics courses have focused on the technology of rendering an image, so

they build the Geometry \rightarrow Image(s) link; other courses have added more emphasis on modeling, which is the Geometrizing activity. But overall problem solving requires that we start farther back in the process, with the problem itself, and that we take the images we create and ask them to provide insight that informs us about the problem.



The graphical problems-solving loop

This feedback loop is very similar to the loop of theory, experiments, results, and analysis of the scientific method, and is often associated with scientific visualization [11] because of this similarity. This loop is implicit when the goal of the computer graphics is to study scientific questions, but we should realize that we can use it when we apply computer graphics to a many different kinds of questions that can be chosen to fit the nature of the particular university, instructor, class, or individual student. Thus this allows the beginning computer graphics course to serve a wide variety of programs.

3 Visual communication

As Root-Bernstein notes, problem-solving tools must be supplemented by communication skills in order to be useful. It is one thing to be able to analyze a problem and find a graphical approach to its solution; it is another to be able to communicate that approach to another person. Students need to learn the fundamentals of visual communication in the computer graphics context in order to express their graphical problem solutions. This means learning to choose an effective viewpoint for a scene; to choose how to express the modeling for a problem; to choose whether to use naturalistic expression through modeling realistic geometry and using lighting models, or an abstracted expression through geometric symbolism and a synthetic color scheme; to lay out controls for any interaction in a way that seems natural to the audience; to present and express motion to the audience in ways that make it clear what the student is communicating.

The fundamentals of sound visual communication are not difficult. Students can be led toward learning sound visual communication by considering examples of good (and of not so good) practice in graphic presentations, using tools such as the annual SIGGRAPH Video Review [14]. We can nurture good communication by providing positive critiques of their work and by allowing them to re-work their images to a certain extent to improve their

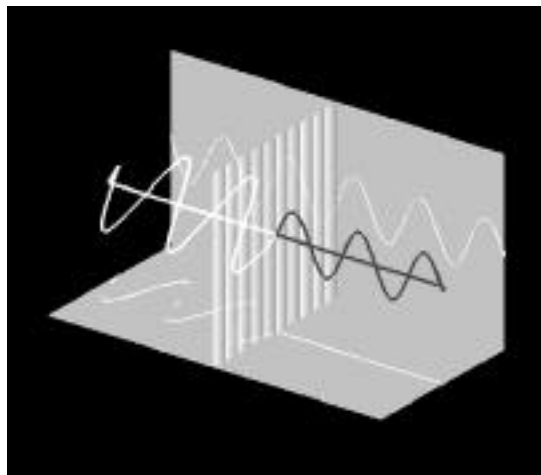
communication. We can also develop experience of group critiques in class or laboratory environments that can help improve the work of the student being critiqued, while at the same time improving the critical facilities and aesthetic eye of those students doing the critiques. In short, this can be integrated into the class discussion and into student work from the beginning without taking a great deal of extra classroom time.

The importance of visual communication in the computing field has been one of the key results of the rise of scientific and information visualization. We have seen that the results of computation or a set of information can overwhelm a researcher without visual tools to help it make sense [2], [11]. These fields are opening the door to effective ways to examine large-scale problems and even to communicate smaller problems better, and students need to have an understanding of how they can use similar approaches to present the results of work across the spectrum of computing applications.

It is important to note, however, that we are not thinking of this visual communication being the equivalent of a study of sound HCI techniques. We are trying to develop experience in seeing and creating effective images, not in understanding the depth of the user task modeling and cognitive processing that are important to sound HCI work. We believe that this could be a very useful background for doing more studies in HCI, however, if that should fit the direction of the computer science program where the course should live.

4 An example

In a recent class, a student was thinking about things she had learned in physics that interested her, and she chose to focus on the polarization of light. Her final work is shown in the figure below, which has been made into a grayscale image, though the original is in color.



A figure illustrating polarized light, from a student-written interactive animation

The basic question she chose to study was the way polarization modifies light waves. In order to do this, she

had to model light waves and a polarizing filter, and develop a way to show the light waves before and after they passed through the filter. She also had to decide how a user should interact with the results of her work and how the image could communicate the polarization process she was interested in.

Most of the modeling for this project was straightforward. Light waves can be modeled adequately as cosine waves, a polarizing filter can be modeled as a set of vertical lines, the before and after display can be accomplished by showing waves on both sides of the filter, and the effect of the polarization can be shown by projecting the waves on two orthogonal planes. For effectiveness, one of the planes is parallel to the polarizing lines and the other is perpendicular to them, allowing the viewer to see that the wave is unchanged in one plane but completely suppressed in the other.

In addition to modeling these objects, the student animated the model by parametrizing the light wave and changing the parameter in the `idle()` function, and set up an interactive capability by defining keystrokes to rotate the entire display and to rotate the angle of the wave. This lets a user explore the effect of polarization on waves of different angles and to look at the display from various viewpoints. However, it added an extra wrinkle to the presentation because the waves now seem to separate at the polarizing filter, so the center lines of the waves were added to maintain continuity between the two sides of the display. Several people have commented that the display is effective in showing the polarization process.

The image shown here, and the model as defined above, were not the student's first attempts. The original approach used much simpler models and little color, and we discussed how to show different kinds of effects and how to use color more effectively, and worked through several approaches to the display. In the process we used increasingly complex ways to do the graphics and increasingly subtle kinds of color. For example, the polarizing filter is not simply a set of vertical lines but a plane that is defined to have a shutter-like texture and be largely transparent. The final version illustrated above was completed about half way through a one-semester course.

5 Implications for the beginning computer graphics course

In recent papers [6] [7] [8] and a developing set of course materials [5], we describe a beginning computer graphics course that is based on a current computer graphics API and has some novel aspects: it emphasizes modeling through the use of a scene graph, graphical interaction through event-driven programming, animation through the API `idle` event, problem solving through a focus on projects based in the sciences, and visual communication through a study of techniques for relating aspects of computer graphics to the way an audience would be comfortable getting information or managing program control. The papers cited earlier in this paragraph include discussions of how to develop graphical problem solving experience while focusing the

primary course attention on creating images through the graphics API.

There is an interesting dialog going on in the computer graphics community at this time about the nature of the beginning computer graphics course. There are two ways to look at this course. It can be thought of as being an inward-looking course, focused on the graphics itself with the intent of developing specific knowledge and skills in the technology or process of creating images; this is the traditional focus of most computer science courses, with an assumption that the student can somehow learn how to use the course content in an application setting. The course could alternatively be an outward-looking that creates a solid graphics knowledge while at the same time helping the student to develop a set of solid computer graphics skills plus experience in using those skills in a problem-solving setting that a student can apply to the rest of his or her professional life. This dual nature of the course allows us to include students from outside computer science and create an environment in which groups of students can bring their varying experience together to solve problems and work on projects.

There are, of course, many approaches to the course that are described in the growing literature in the field [4]. A very good paper describing a course with an inward-looking, image-creating emphasis is [10]; the authors argue that the goals of the beginning course should be rendering, modeling, animation, and postprocessing. This course includes some quite advanced concepts but the paper suggests that most of the emphasis is on mastering various image-creation techniques. Most of the literature in the last ten years has described courses with a similarly inward-looking focus, but the changes in capabilities for inexpensive desktop graphics systems and the growing presence of computer graphics well outside its traditional technical scope are raising questions about whether this inward looking approach is the best way to begin computer graphics instruction. I believe that in many institutions the outward-looking beginning computer graphics course would help develop a broader student background, and for this the content of the beginning course would include a mix of computer graphics content and problem-solving content. In particular, the areas of modeling and rendering through an API, interaction, animation, graphical problem-solving, and visual communication are all possible in a single term course.

The two approaches we have discussed have a great deal of overlap, but the differences can be quite significant. Yet for students who want to specialize in computer graphics and perhaps use that as the focus of their careers, the difference is not that great, because both approaches can be the basis of solid advanced courses that will doubtless be focused on the details of various techniques for image creation. Individual instructors and departments will need to decide whether their overall, long-term needs are best served by an inward-looking beginning course or an outward-looking beginning course.

6 Conclusions

The beginning computer graphics course offers a solid opportunity to expand the problem solving and communications abilities of computer science students as well as students from other disciplines. If this course is conceived as being based on a graphics API and focusing on API-based image creation, there is time in the course to discuss more than just the API and graphics processes, so problem solving and visual communication can be included without taking anything away from the content of the course. Students who complete the course, whether they are from computer science or another field, will then come away from it with significant skills that will serve them well in their professional lives.

References

- [1] Bergin, Joseph, "Fourteen Pedagogical Patterns," online at <http://csis.pace.edu/~bergin/>
- [2] Card, Stuart K, Jock D. Mackinlay, and Ben Shneiderman, *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann, 1999
- [3] Clark, D. and M. J. Bailey, "Visualization of Height Field Data with Physical Models and Texture Photomapping," *Proceedings of IEEE Visualization '97*, October 1997.
- [4] Cunningham, Steve, "Literature Survey on Beginning Computer Graphics Courses in Computer Science," in preparation
- [5] Cunningham, Steve, *Computer Graphics: Programming, Problem Solving, and Visual Communication*, draft manuscript online at <http://www.cs.csustan.edu/~rsc/NSF/>, 2001
- [6] Cunningham, Steve and Michael J. Bailey, "Lessons from Scene Graphs: Using Scene Graphs to Teach Hierarchical Modeling," to appear in *Computers & Graphics*, 2001
- [7] Cunningham, Steve, "Projects for a Computer Graphics Programming Course," *SIGGRAPH 2000 Conference Abstracts and Applications*, July 2000, pp. 56–59
- [8] Cunningham, Steve, "Powers of 10: The Case for Changing the First Course in Computer Graphics," *Proceedings of the ACM SIGCSE 2000 Technical Symposium on Computer Science Education*, *SIGCSE Bulletin* 32(1), pp. 293–296
- [9] Gamma, Erich et. al., *Design Patterns: Elements of Reusable Object-Oriented Design*, Addison-Wesley Longman, Reading MA, 1994
- [10] Lowther, John L. and Ching-Kuang Shene, "Rendering + Modeling + Animation + Postprocessing = Computer Graphics," *Proceedings of the Seventh Annual Consortium for Computing in Small Colleges: Midwest Conference*, Valparaiso, Indiana, October 2000, reprinted in *Computer Graphics* 34(4), November 2000, pp. 17–20
- [11] McCormick, Bruce, Thomas A. DeFanti, and Maxine D. Brown, *Visualization in Scientific Computing*, *Computer Graphics* 21(6), 1987
- [12] O'Connell, Kenneth, "Visual Ways of Knowing, Thinking, and Interacting," in *Interactive Learning Through Visualization*, S. Cunningham and R. J. Hubbard, eds., Springer-Verlag Berlin Heidelberg, 1992, pp 129–136
- [13] Root-Bernstein, R. S., "Tools for Thought: Designing an Integrated Curriculum for Lifelong Learners," *Roeper Review* 10(1987), 17–21
- [14] SIGGRAPH Video Review, a publication of ACM SIGGRAPH that contains the state of the art in computer graphics animations. For information on SVR, see <http://www.siggraph.org/svr/>