BRIDGES
Mathematical Connections
in Art, Music, and Science

Aesthetics, Beauty, and Computer Science

Paul S. LaFollette, Jr.
CIS Dept.
Temple University
Philadelphia, PA 19122, USA
lafollet@andante.cis.temple.edu

Undergraduate computer science majors could benefit from increased exposure to the connections that exist between their field of study and other disciplines, particularly in the arts, humanities, and social sciences. The author proposes several ways in which an increased appreciation of purportedly distant disciplines might be approached. He concludes with an argument that a good starting point would be to learn that the arts might not be so very distant after all.

> The scientist does not study nature because it is useful, he studies it because he delights in it, and he delights in it because it is beautiful. If nature were not beautiful, it would not be worth knowing, and if nature were not worth knowing, life would not be worth living. Of course, I do not here speak of that beauty that strikes the senses, the beauty of qualities and appearances, not that I undervalue such beauty, far from it, but it has nothing to do with science, I mean that profounder beauty which comes from the harmonious order of the parts, and which a pure intelligence can grasp.
>
> **Henri Poincaré**

As a computer scientist interested in undergraduate education, I came away from the 1998 Bridges Conference with an increasing sense that the curriculum we offer our majors needs to include more explicit recognition of the kinds of connections between the sciences and the arts, humanities and social sciences that Bridges celebrates. As our discipline has matured, the number of credit hours that our majors must devote to computer science courses has grown at the expense of electives. Unsurprisingly, many of our students prefer to use such few electives as remain to take additional courses in science and mathematics. Given this environment, it seems particularly important that we find ways of exposing our undergraduates to the science and mathematics that lurk in the humanities, and the humanity that hides in the sciences. This paper offers some thoughts about how I would like to see my department enhance its Computer Science curriculum to accomplish this.

There are two more or less independent and complementary paths we can follow. The first is to more explicitly demonstrate the ways in which computer science is used in other fields. This could be accomplished within our present curriculum by actively attempting to find examples from diverse problem areas to demonstrate the concepts being taught in, say, programming/algorithms/data-structure sequence. In the first programming course, for instance, I generally make use of a set of example programs for calculating and recording student test scores. The same concepts could be demonstrated with problems from other disciplines. (My more insightful colleagues may already do this, but I think it would be a good

idea to make it a goal for all of us). Similarly, the next time I teach the computer graphics course, I surely plan to discuss the potentially improper restoration of a picture by Holbein that was discussed in the 1998 Bridges proceedings.

When we are teaching service courses to non-majors we also have an opportunity to help our students see the computer science that is infusing other areas of study. Our survey course of computer science and computer applications gives a good opportunity to help students from the arts, humanities, social sciences, and the other sciences (of course) understand where our discipline can fit into their disciplines. I routinely invite colleagues from the schools of music and of communications to lecture to my sections of this course. I will undoubtedly expand these activities in the future.

All that I have proposed above concentrates on the contributions computer science can make to other areas of study. It is important in so doing that we avoid trivializing our discipline by suggesting that it has value only in application to other fields of study. It is perhaps even more important to avoid the presumptuousness of suggesting that we are an indispensable part of other disciplines. Both of these incorrect views can be avoided if we accompany the exposition of our application to the arts and humanities with the simultaneous unfolding of the importance of aesthetic considerations to our own craft.

The computer science community has long recognized that there *are*, in fact, aesthetic considerations in the development of software, and this is a part of what I am discussing. Indeed, it is an important part, for stylish, well-conceived, well designed, and beautifully written software is often more correct and more easily modified and maintained than pooly written software. This is, of course, largely because the aesthetic we have developed regards as beautiful precisely those structural elements that contribute to correctness and maintainability. The point is that we already attempt to convey to our students the elements of stylish software. We need, further, to directly convey that this is indeed an aesthetic issue, that good software is beautiful, and that the creation of beauty is not frivolous, nor limited to those in less scientific disciplines, but that it is fun, even joyful, and further is an essential element of our calling. As Donald Knuth has noted, "The process of preparing programs for a digital computer is especially attractive, not only because it can be economically and scientifically rewarding, but also because it can be an aesthetic experience much like composing poetry or music."

Beyond this, there are other issues of artistry in computer science. Specifically, there is the beauty and joy that is to be found in the uncovering of truth (in formal thinking) in the understanding and manipulation of mathematical structure. That these activities can become a form of art is a fact to which both our own students and students from other colleges need to be exposed. Toward that end, I routinely structure much of the content of our survey course around the question "What turns computer scientists on?" I try, for example, to demonstrate that I, at least, can find beauty in the analysis of simple algorithms, or that digging beneath the surface of a computer animation and uncovering the underlying structure can disclose beauty of a kind not directly related to the beauty of the animation itself. I do not necessarily expect the students to directly appreciate the joy that I find in these activities, but it seems important to me that they see that I *do* find such joy. This is, for me, a way of finding commonality of experience with those students who are more drawn to the arts, humanities, or social sciences. This engages the students, and I hope it helps them realize that scientists and mathematicians need not be insensitive to beauty, wisdom, and passion, just as non-scientists can manifest logic and reason.

For our own majors, I believe it is even more important that they be encouraged to discover the beauty as well as the technical skills of our endeavor, and I am attempting to find ways to intentionally infuse this into our curriculum. To begin, I suggest that we all, as best we can, take the opportunities that arise to reveal our own visceral responses to the beauty we encounter in our subject matter. This can be a bit

intimidating. When I allow myself to become passionate about the aesthetics of something I am teaching, it is inevitably greeted by some snickering. However, it is precisely because of the attitude which promotes such snickering that it is important that we all demonstrate to our students that it is OK, indeed desirable, to be able to respond to all manifestations of truth with emotional as well as intellectual respect.

In closing, I would like to offer one or two specific techniques that I use in presenting this material to my students. I do not generally attempt to provide definitions or guidelines for identifying beauty. Rather, early in the semester I try to begin a discussion of what it might mean to apply the words "clever," "elegant," or "beautiful" to various ideas, processes, or structures that we may encounter. Then, as the semester progresses, I freely make use of these words as occasions arise. My goal is not to impress my particular aesthetic on the students, but to provide, perhaps, a trellis on which they can begin to build their own. For our majors, this is straightforward. I end up, for instance, identifying insertion sorting as clever, the recursive solution to the towers of Hanoi problem as elegant, and a more obscure but quite lovely non-recursive algorithm for the same problem (out of which pops, rather unexpectedly, the binary Gray code) as having elements of beauty.

For the non-majors in our survey course, a significant problem is to find ideas that are non-technical enough to be accessible, at least on an intuitive level, yet still interesting enough to display elegance and beauty. One area that is somewhat intuitively accessible, especially to those students who have some minimal mathematical background, is the discussion of computational complexity and the idea of NP completeness. Possibly the most easily explained of the NP-complete problems is the subset sum problem. This can be expressed as follows:

Problem 1:

Given a set $V = \{V_1, V_2, \ldots V_n\}$ and a target value T, the problem is to determine whether there is a subset $S \subseteq V$ such that the sum of the elements of S is equal to the target value T.

It turns out that this is equivalent to the following:

Problem 2:

Let $V_{n+1} = 2T - \sum_{k=1}^{n} V_k$. Consider all of the sums of the form $\sum_{k=1}^{n+1} I_k \cdot V_k$ where each of the I's can take on the value +1 or -1. Are any of these sums equal to 0?

That the answer to problem 2 is "yes" if and only if the answer to problem 1 is "yes" is demonstrable using straightforward algebraic manipulations. Most students, I think, are willing to see this as a clever improvement. For one thing, to test a particular proposed solution, this formulation always uses all of the V's. Furthermore, the value that demonstrates success is the special value 0. (The target value T is hidden away in the V_{n+1} term.) I, at least, would argue that those facts make this a tidier representation of the problem.

There is a third equivalent statement for this problem (equivalent in the sense that it is true whenever Problem 2 is true):

Problem 3:

$$\text{Is it the case that } \int_{-\Pi/2}^{+\Pi/2} \text{Cos}(V_1X)\text{Cos}(V_2X) \ldots \text{Cos}(V_{n+1}X)dX \neq 0 \text{ ?}$$

The equivalence of this formulation is less obvious, but follows easily from repeated application of the formula for the product of two cosines. I find this to be the most elegant of the three representations we have thus far considered. This may be because I enjoy arriving at places where the continuous and discrete worlds embrace.

Best of all, though, is Problem 4:

Consider the family of matrices defined by the following rules:

$$M_1 = (V_1).$$

$$M_k = \left(\begin{array}{c|c} M_{k-1} & \text{diag}(V_k) \\ \hline \text{diag}(V_k) & M_{k-1} \end{array} \right)$$

Here, $\text{diag}(V_k)$ is a diagonal matrix with V_k as each of the diagonal elements.

Problem 4 asks, "Is M_{n+1} singular?"

I will leave it to the reader to ponder the various aesthetic qualities of this family of matrices, and will close by pointing out that not only is it mathematically pretty, but it is visually attractive as well. If one assigns to each of the V's a different color, and then
colors in each entry in the matrix with the color of the appropriate V, one gets the sequence of images shown in Figures 1 through 8 below. The matrix M_1 is uninteresting (it is simply a square all of one color) and is therefore not shown.

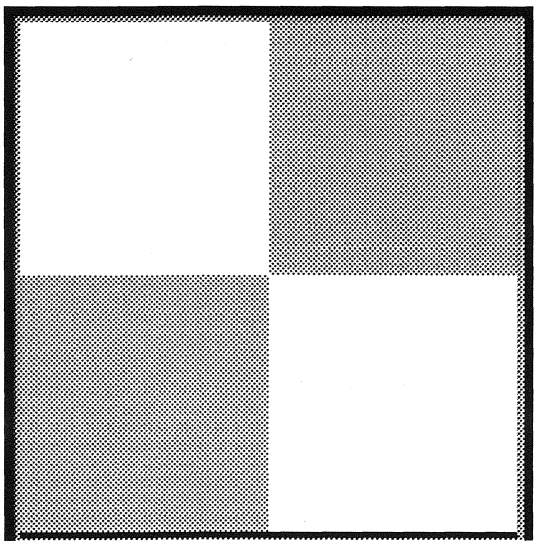

Figure 1: *The Matrix* M_2

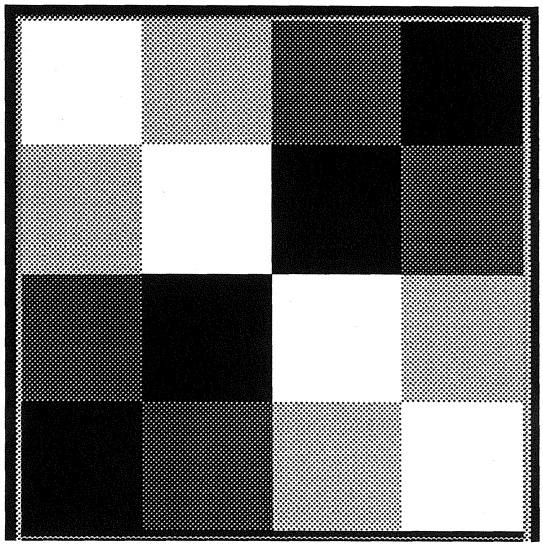

Figure 2: *The Matrix* M_3

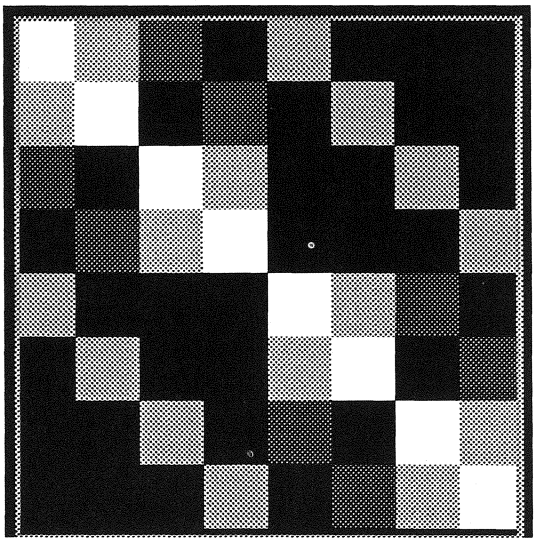

Figure 3: *The Matrix* M_4

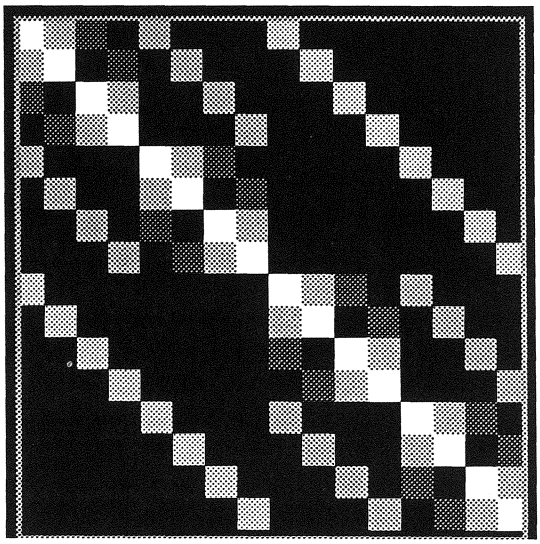

Figure 4: *The Matrix* M_5

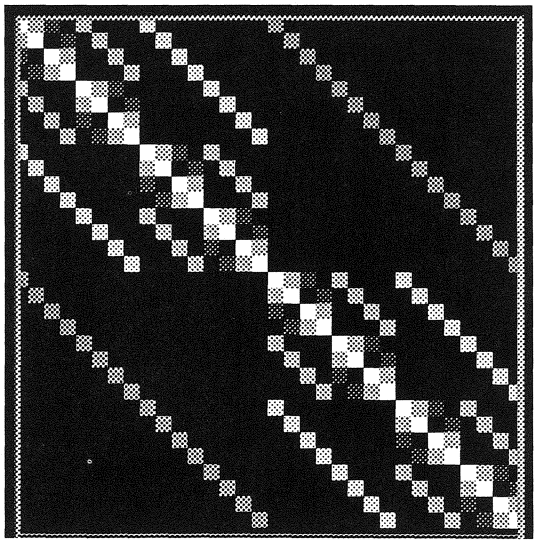

Figure 5: *The Matrix* M_6

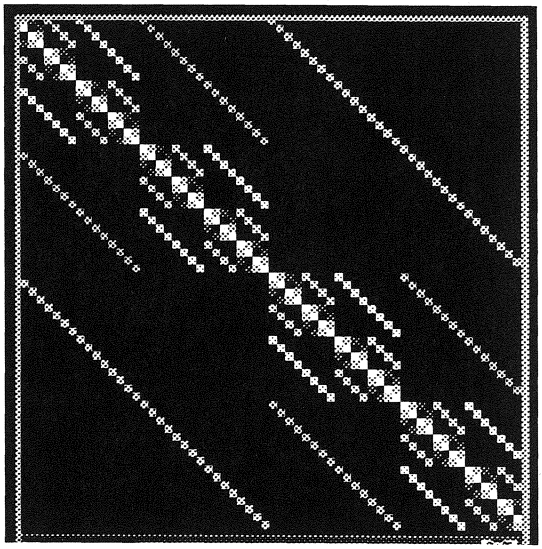

Figure 6: *The Matrix* M_7

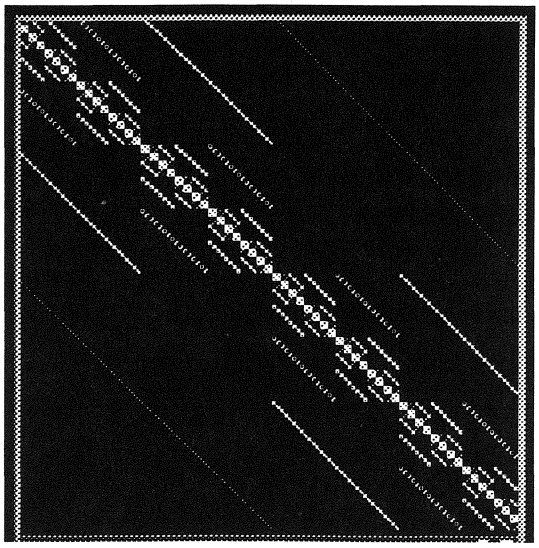

Figure 7: *The Matrix* M_8

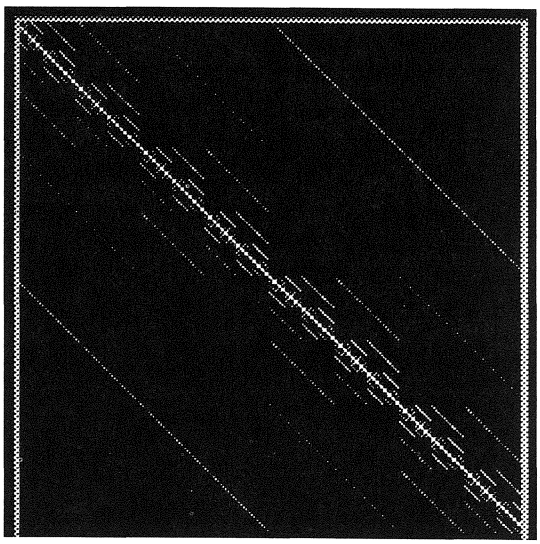

Figure 8: *The Matrix* M_9