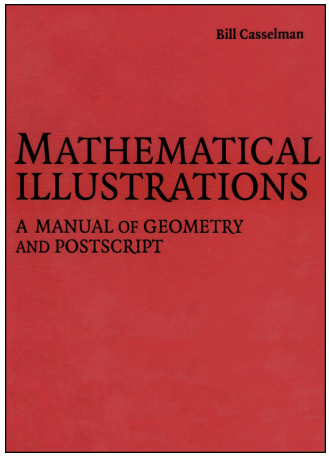
Book Review

Mathematical Illustrations: A Manual of Geometry and PostScript

Reviewed by Denis Roegel

Mathematical Illustrations:
A Manual of Geometry and PostScript
Bill Casselman
Cambridge University Press, 2005
336 pages, ISBN 0521839211
Hardcover, US$90.00; Paperback US$39.99

High quality mathematical illustration has long been a specialized craft, akin to the layout of musical scores or mathematical formulæ. Illustrating a proof, or drawing a graph, used to be difficult, and it is easy to find errors in the drawings in old books. Drawing errors are of course a problem for the understanding of a proof. Unfortunately, such cases are still common in the current scientific literature, perhaps because mouse-made drawings do not take full advantage of the power now available. The shift of power, from hand-quality drawings to computer-quality drawings, goes back to the 1970s and started with the advent of good printers. New computer languages were developed for graphical tasks. In 1981, Brian Kernighan described his high-level PIC language for typesetting graphics [3]. At about the same time, the first version of the PostScript language was made public. Unlike languages such as PIC, PostScript was designed not as a user-oriented language but rather as a page description language to serve as an interface between graphics-producing software and printers. Printers with PostScript interpreters were then manufactured, and consequently software was written to produce PostScript files. These files were normally independent of the printer, provided the printer knew PostScript. This independence is still a major asset of typesetting software such as TEX, where one often first produces a PostScript file, then prints it. The software has to know only about PostScript, not about the printer.



PostScript can also be used directly as a graphics programming language, and Bill Casselman's book *Mathematical Illustrations: A Manual of Geometry and PostScript* is devoted to the use of PostScript in the context of geometry.

A graphics language designed for the user has different requirements from one designed as a back-end for typesetting software. High-level languages such as PIC, MetaPost (created by Donald Knuth and John Hobby [4]), and others, try to achieve abstraction, flexibility, and naturalness. In MetaPost, for instance, drawing a segment from $(0, 0)$ to $(100, 40)$ is done with

```
draw (0,0)--(100,40)
```

PostScript, instead, is a language designed to be interpreted easily by a machine, and the burden of creating PostScript code normally lies not on the user, but on some software's driver. The same segment as above is drawn in PostScript with:

Denis Roegel is maître de conférences at Université Nancy. His email address is roegel@loria.fr.

`newpath 0 0 moveto 100 40 lineto stroke`

Although there is a correspondence between the two expressions, the latter is obviously less user-friendly.

PostScript, in fact, is a stack- and list-based language. The language inherits features found in HP calculators using the Reverse Polish Notation. Other examples of such languages are FORTH and the BibTEX style language for typesetting bibliographies in TEX.

In a stack-based language, operands are given first (and put on a stack), then the operations. `2 3 mul 4 add 12 sub` will, for instance, push 2 and 3 on the stack, multiply them, push 4, adding it to the previous result 6, and subtracting 12, resulting in -2 on the stack. `100 40 lineto` means "draw a line from the current point to point $(100, 40)$." Casselman's book shows how far one can go with such a language, and how it can be used for mathematical illustrations.

In fact, Casselman's book is not only an introduction to PostScript, but actually a book covering two topics: geometry and PostScript. It was used by Casselman as a text for a third-year undergraduate course in geometry. Casselman is a mathematician interested in graphics [1, 2] and the book will appeal as much to mathematicians as it will appeal to programmers.

Casselman's book certainly does its best to address the two topics of geometry and PostScript, and in this respect it is excellent. The real problem lies in the PostScript language, which quite honestly is not the simplest graphical language, especially for beginners. Mathematicians who already have some familiarity with programming will find the book interesting and will enjoy its dual perspective. Those who are beginning in graphics might however be misled, or even deterred, depending on how far their graphical notions extend and what their aims are. If the reader is interested in an introduction to PostScript, Casselman's book certainly suits the purpose very well. But if the reader is looking for a practical language for making his/her own figures, I am doubtful that PostScript is the right choice. One should not forget that learning a language is an investment, and it may take years to take full advantage of certain languages. A better choice in this respect might be MetaPost, a language meant for users. One criterion for choosing a programming language for a given task is the number of people using that language. There are certainly more people programming in MetaPost than in PostScript (although many people manipulate PostScript files, of course).

One question raised by Casselman's book concerns the communication between its two readerships, the mathematicians and the programmers. Will the worlds of these two kinds of users really merge, as the book would like them to? Before answering this question, let us ponder a number of other issues.

Mathematical Illustrations

The book is aimed at producing mathematical illustrations, but what are mathematical illustrations? A perusal of the book will show that what is meant are geometrical constructions (for instance for the Pythagorean Theorem), curves, 3-dimensional surfaces, and 3-dimensional objects. To a great extent, these illustrations are made of straight or curved lines, or of surfaces delimited by curves and colored in some way. Labels appear only seldom, and we will come back to this topic later.

PostScript is well suited to the task of making technical illustrations such as plans, diagrams, flow charts, etc. But technical illustrations are different from mathematical illustrations, and for the latter other languages, such as MetaPost, seem better suited.

Throughout the book, the author tries to develop his methodology. For instance, he writes that "programs should reflect concepts" or that "To get good results from PostScript, first get a simple picture up on the screen that comes somewhere close to what you want and then refine it and add to it until it is exactly what you want."

I would rather say that first a drawing should be on paper, and its logical structure analyzed. It is essential to separate the design of the drawing from its implementation, because the implementation choices can cripple the design and make it more difficult to change. Afterwards, when the design has been stabilized to some extent, we know better what we want to draw and how, and we can observe the result and fine-tune it.

The Content

Let us now review some of the content of the book. The first chapter, in little more than twenty pages, gives the basics of polygonal drawing. The second chapter adds nothing about PostScript but is a refresher on coordinate geometry. Chapter 3 develops the language by presenting the means to store and reuse data. Although the chapter distinguishes between variables and procedures, both are actually data and are stored in the same way. A variable stores a value, such as an integer or a string, and a procedure stores some text, which usually is a list of commands; these commands are grouped with braces. Chapter 4 explains quite extensively and very clearly how coordinate changes are handled through matrices representing affine transformations. A 2-dimensional transformation is represented by a matrix stored as an array of 6 elements, and these matrices can easily be inverted and multiplied.

This chapter also shows how affine transformations in the plane can be represented as 3-dimensional matrices in the plane $z = 1$. The book is full of interesting problems, but sometimes the problems seem quite complex and disproportionate with the notions they purport to introduce. Conditionals, for instance, are introduced after stating a problem of line intersection taking up more than three pages. There are probably easier ways to introduce these concepts. However, the problem is interesting for other reasons, as it puts into practice the various coordinate systems and the handling of arrays.

Chapter 5 discusses programming loops and arrays and illustrates them by the drawing of regular polygons. Function graphs are a natural application of loops. Incidentally, a PostScript path can be created with a loop, somewhat like a machine leaving a trail, but the path itself is drawn only at the end of the construction. This is convenient, as there is no need to store the coordinates in an array, and we find the same useful features in MetaPost, for instance. Arrays are a natural structure to loop on, and the drawing of general polygons given by a list of pairs is a good application. Arrays also provide a very general structure and can contain lists of heterogeneous elements. The first element could be an integer, the second a pair, and so on. Readers acquainted with more recent languages such as Python will find themselves in familiar territory.

Chapter 6 extends the basic drawing facilities introduced in chapter 1. Up to now, only segments could be drawn. The main topic of this chapter is Bézier curves. The theory of Bézier curves and the more general Bernstein polynomials is sketched. The author shows for instance that points of a Bézier curve are weighted averages of its control points, that is, of the points that define the curve.

Chapter 7 shows how curves can be drawn automatically. In general, the author stresses the importance of separating the construction of a curve from its drawing. For instance, drawing a hyperbola using the procedure `hyperbola` could be written as

```
newpath -2 2 4 hyperbola stroke
```

the path being drawn only when the `stroke` command is executed.

The author shows how a procedure `f` can be defined in such a way that, taking a value x, it outputs a pair $[f(x)\ f'(x)]$ on the stack. This procedure can be used to produce the control points of the Bézier curves approximating the graph of the function f. Eventually, in order to draw a function `quartic` between -1 and 1 using 8 Bézier segments, one merely writes

```
newpath
-1 1 8 /quartic mkgraph
stroke
```

The "/" here means that the name "quartic", and not the definition of the function, is pushed on the stack. This allows the `mkgraph` function to call `quartic` as needed for various values of its parameter.

One such quartic given as an example is x^4 and this is coded by

```
/quartic { 2 dict begin
 /x exch def
 [
  x x mul x mul x mul % f(x)=x^4
  x x mul x mul 4 mul % f'(x)=4x^3
 ]
end } def
```

Although the syntax is quite verbose, this gives the general idea of how function graphs can be drawn. (What follows "%" are comments.) It is, by the way, possible to define a function taking a string representing an expression such as `x^4` and producing `x x mul x mul x mul`. One could even imagine defining a function computing the formal derivative of a function. These topics are beyond the scope of this book, although they are tackled in Appendix 6 in the case of polynomials.

An extension of these ideas is the drawing of parameterized curves, where it is assumed that the array $[[x(t)\ x'(t)]\ [y(t)\ y'(t)]]$ is given. This is for instance applied to draw a circle out of 8 Bézier segments.

Chapter 8 is about the analysis of paths. Up to now, paths have been rather static. They were accumulated, and then drawn, but it is actually possible to work on a path as data. As an example, the author introduces 2-dimensional transformations and explains the role of the Jacobian derivative in the approximation of a map. If certain conditions are met, such a transformation is conformal and preserves angles. A path can then be transformed by going through its components using `pathforall` and building an array representing a new path, obtained from the transformed components of the initial one. Finally, the commands in the array are executed, and this produces the new path. This chapter then applies this technique to map transformations, such as cylindrical projections, Mercator projections, and stereographic projections.

Chapter 9 is about programming and takes as natural applications replicating structures, such as arrays or fractals. Recursion is a very natural expression of certain algorithms, but this naturalness requires some care, in particular when local variables are involved.

Sorting is an application of arrays. An array of integers can be sorted for instance by subdividing an array, and this can be done recursively. The first half can be sorted, then the second half, and then

the sorted subarrays can be merged. Two sorting algorithms are presented in this chapter, the "bubble sort" and the "quicksort".

An interesting application to sorting is a procedure finding the convex hull of a set of points in the plane. The algorithm is stated very clearly, but the procedures are not broken down enough for my taste.

Chapter 10 is an introduction to perspective and projective geometry. The author introduces the projective plane and homogeneous coordinates. This chapter is therefore a mathematical preliminary to the representation of 3-dimensional objects in two dimensions.

Chapters 11 and 12 are about transformations in three dimensions and introduce the various mathematical prerequisites and the matrix representations of the different operations, such as projections, rotations, etc.

Chapter 13 and the next one describe a 3-dimensional extension to PostScript. This extension facilitates the creation of simple 3-dimensional scenes. When it is used, the location of the eye can be defined and coordinates can be changed by translation, scaling, or rotation. Paths can be defined in 3 dimensions. The extension is provided with matrix operations that operate on 4×4 matrices.

Chapter 14 shows how convex polyhedra can be represented and how visibility and shading algorithms are used for its faces. Smooth surfaces are obtained either by a fine computation of the shading and the use of small grids, or by the use of the more elaborate shading features introduced in version 3 of PostScript. Casselman's book takes the reader to advanced topics such as Gouraud shading and illustrates them, for instance, on a sphere.

This chapter also serves as an introduction to more general 3-dimensional rendering, involving objects that are not mere convex polyhedra. We are introduced to an algorithm of binary space partition, whose code is available online.

The concluding chapter 15 is particularly interesting as it details algorithms for splitting arbitrary 2-dimensional surfaces into small triangles, for the purpose of shading and visibility control.

The book is supplemented by several useful appendices giving in particular a summary of commands, details about the editing and running environment, the inclusion of labels, and zooming.

The appendix on labels, called "Simple text display," runs only four pages, and in my opinion this treatment is too brief. It may surprise the reader to find that labels actually seldom appear in the drawings for which code is given in the book, and there are two reasons for this. First, the author claims that a good drawing should speak for itself and does not need crutches such as labels. This is certainly true for some drawings, but not everyone will agree that it applies to all drawings. In particular, the author actually uses labels in certain figures (for instance in the pentagon on page 82) when he explains their structure. A second reason is that labels go somewhat beyond graphics and represent a bridge with text typesetting. Putting a label requires both the typesetting of text (and therefore dealing with fonts, symbols, special rules of positioning for exponents, etc.), and the location of this text with respect to the graphics. Hence, text and graphics somehow must communicate, and this is of course difficult. Casselman shows several cases of graphics labeling, but they are very basic.

Two Tracks

The two tracks of Casselman's book, mathematics and programming, are interleaved, but are also to a great extent independent. This is both an advantage and a drawback. It is an advantage because it allows the mathematician to learn about the theory of Bézier curves, projective geometry, homogeneous coordinates, and other interesting matters, including some that are given as (mathematical) exercises, without being bothered by the programming. Similarly, the programmer who wants to learn about PostScript will find an excellent introduction and will be able to skip all the mathematics. The exposition in both tracks is very clear and should satisfy readers from each field.

But the independence of the tracks, or rather, the fact that a reader need not read everything in order to progress, will also be likely to discourage mathematicians from reading the PostScript parts, and programmers from reading the mathematical parts. Still this could be interpreted positively, as it allows a programmer to finish his part of the book, and then to return to it for the remaining bits.

Will the mathematicians grasp the programming part? I am somewhat doubtful. The PostScript language is very unusual and, although very interesting and informative, it can hardly be viewed as an introductory programming language. It is likely, therefore, that only those readers with an interest in graphics, and with previous programming experience, will dive into the PostScript programming.

Yet, learning about PostScript is very rewarding. To most of us who are not familiar with a stack-based language, the book shows its beauty and versatility. Moreover, it illustrates the basic features of a general programming language, such as conditionals, loops, and structures like lists and trees, and shows how these structures can be used. The language is also graphics-oriented, and therefore has features that do not have a parallel in nongraphical languages, for instance path traversal.

The book could have been improved in several ways. First, the book is two-color, and perhaps

there could have been a few four-color pages, although that would have increased the cost. Second, although the author stresses the importance of good programming, the use of comments, etc., I feel that some pieces of code (pages 157–158, for instance) are too lengthy. They will frighten the mathematician; they could have been broken into smaller pieces. A procedure occupying two pages may be acceptable with certain languages, but I do not think it is with a stack-based language. There are also some technical inaccuracies. The author writes for instance (page 47) that "A procedure in PostScript is just any sequence of commands enclosed in brackets {...}." This is not correct. A sequence of commands enclosed in brackets {...} is a group, and it is possible to have groups without naming them. A procedure is a stored group of commands with some name. So, it is not true that a procedure is assigned to a variable, but a procedure is a variable. There is also an inaccuracy concerning pixel coloring. The author writes (page 237) that "pixels are colored in the order of their depth." This is not quite true for rendering engines such as OpenGL; instead, pixels get a new color if the color corresponds to a point in space nearer than the previous one used for coloring that pixel; the result does not depend on the order in which the pixels or objects are traversed. Finally, it is too bad that the author does not offer a comparison with other 2-dimensional or 3-dimensional graphical languages, such as MetaPost or OpenGL, although MetaPost is mentioned in the preface.

References

[1] Bill Casselman, Visual explanations, *Notices of the AMS*, **46** (January 1999), 43–46.

[2] ———, Pictures and Proofs, *Notices of the AMS*, **47** (November 2000), 1257–66.

[3] Brian W. Kernighan, PIC: a language for typesetting graphics, *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, 92-98, ACM Press, 1981.

[4] John Hobby, *A User's Manual for MetaPost*, Bell Labs, 1994.