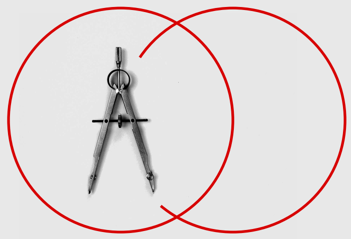
Preface



```
1 0 0 setrgbcolor
newpath
0 0 1 0 360 arc
stroke
newpath
1 0 1 0 360 arc
stroke
```

This book will show how to use PostScript for producing mathematical graphics, at several levels of sophistication. It includes also some discussion of the mathematics involved in computer graphics as well as a few remarks about good style in mathematical illustration.

To explain mathematics well often requires good illustrations, and computers in our age have changed drastically the potential of graphical output for this purpose. There are many aspects to this change. The most apparent is that computers allow one to produce graphics output of sheer volume never before imagined. A less obvious one is that they have made it possible for amateurs to produce their own illustrations of professional quality. Possible, but not easy, and certainly not as easy as it is to produce their own mathematical writing with Donald Knuth's program TEX. In spite of the advances in technology over the past 50 years, it is still not a trivial matter to come up routinely with figures that show exactly what you want them to show, exactly where you want them to show it. This is to some extent inevitable—pictures at their best contain a lot of information, and almost by definition this means that they are capable of wide variety. It is surely not possible to come up with a really simple tool that will let you create easily all the graphics you want to create—the range of possibilities is just too large. All you can hope for is that the amount of work involved in producing an illustration is in proportion to the intrinsic difficulty of what you want to do. And the intrinsic difficulty of producing a good mathematical illustration inevitably means that you should expect to do some interesting mathematics as well as solve interesting computational problems along the way. Mathematical illustrations are a special breed—*a good mathematical illustration almost always requires mathematics in the process of making it.*

Nowadays there are many tools to help one produce mathematical graphics, of course. A partial list would include the free packages `xfig`, `pictex`, `PSTricks`, `MetaFont` and `MetaPost`, as well as commercial mathematical programs such as `Maple` and `Mathematica` and professional graphics design tools such as `Illustrator`. Which one to choose apparently involves a trade-off between simplicity and quality, in which most go for what they perceive to be simplicity. The truth is that the trade-off is unnecessary—once one has made a small initial investment of effort, by far the best thing to do in most situations is to write a program in the graphics programming language PostScript. There is practically no limit to the quality of the output of a PostScript program, and as one acquires experience the difficulties of using the language decrease rapidly. The apparent complexity involved in producing simple figures by programming in PostScript, as I hope this book will demonstrate, is largely an illusion. And the amount of work involved in producing more complicated figures will usually be neither more nor less than what is necessary.

The principal advantage of PostScript is that it allows essentially complete control over the final product, something impossible with all of the graphics packages I listed above. Having such fine control over your figures means that once your code is in place, it is often quite easy to modify it. This makes it a great tool for discovering, not only explaining, mathematics.

The advantage of control is very evident to those who have used, say, `xfig` or `pictex`, less so perhaps for `Maple` and `Mathematica`. What becomes apparent in the course of heavy usage, however, is that a program like `Maple` is designed for graphics only incidentally. It produces *huge* files—really, unnecessarily huge—and in practice seems reluctant to draw exactly what you want to draw. It does do all sorts of interesting computations, but normally the best way to use this talent is to have it output data files which a PostScript program can then access. The program `Mathematica` seems to be better adapted for graphics, but there are still many simple tasks it has trouble with. In addition, all Mathematica figures somehow preserve in them the rather strong flavour of Mathematica. Whereas PostScript is far more neutral, and is better able to take on a characteristic quality of your own devising. This can be a source of great satisfaction.

The nearest competitor to PostScript is perhaps the programming language MetaPost. It is an extension of Donald Knuth's MetaFont written by John Hobby, once a graduate student of Knuth's and the developer of the great utility dvips. There are a few well known mathematicians who love it. It has many fine features, but I for one find it much less intuitive to use than PostScript.

Another advantage of using PostScript is that there exists a PostScript interpreter of good quality that costs nothing and runs on nearly all platforms—the program ghostscript, which I'll refer to throughout this book. One of the appendices explains how to install it on your computer if it is not already there, and how to configure it for a pleasant working environment.

The principal disadvantage of using PostScript is that for complicated computations it is not as convenient or efficient as the more standard languages such as Java or C. This problem has an easy solution: with just a little knowledge of PostScript you can write a program in one of those languages that will produce a PostScript file.

There are some other disadvantages of PostScript, however. Anyone who takes up this book seriously will realize this quite quickly, so I had better stave off disappointment ahead of time.

For one thing, PostScript is not a language with much structure. This, however, turns out to be both a plus and a minus—a plus because you can program in PostScript by 'cut and paste' without worrying too much about the consequences. For another, the complete language possesses an intimidating number of commands, but in practice the basic drawing language is rather simple. Almost anyone familiar with coordinate geometry can learn how to do some interesting drawing with the language, and in a remarkably short time. For more complicated work one will want to build up a library of useful procedures. This book will explain a few, and in particular one that extends PostScript to three dimensions.

Another bad feature of the language is that debugging and error handling are both atrocious, at least with the interpreters available today. It would be nice if someone would come up with an interpreter with which one could trace a PostScript program line by line while keeping an eye on its internal state. Such a tool would make development a far easier job. Making an interpreter like this doesn't look to be too difficult a task, at least if one wants only to establish a good environment for technical graphics (as opposed to commercial production). For this relatively restricted task it wouldn't be necessary to interpret the whole language, and in fact it wouldn't even be absolutely necessary to interpret PostScript itself, but merely something close to it. But in any event such an interpreter hasn't been built yet.

And, finally, the integration of text and graphics, although quite feasible, is not as easy as it might be. There seems to be no graphical problem more vexing to mathematicians than that of how to place TEX labels in a diagram. This book will explain (in one of the Appendices) how to do it, if awkwardly, in a PostScript figure. Again, what is needed is a tool that doesn't exist but that shouldn't be too hard to produce—one that creates technical text displays that one can import easily into a basic background diagram. Preferably, one with a simple mouse interface for text editing, scaling, and placement. In the end, such a program would presumably merely import a PostScript program produced by TEX into another PostScript file. There used to be a demonstration program named Import, made available by Adobe Systems when Display PostScript was first distributed on computers running X-Windows, that was better than anything else now available. It didn't do much, but it was refreshingly simple. Among its virtues was that its output was basically no more complicated than it had to be, which meant that you didn't have to stop working on a figure once you had 'imported' it. Commercial graphics programs (such as Adobe's Illustrator) are capable of importing PostScript files, but they are expensive. In addition they will take your PostScript programs up a one-way street—what they produce will be far more complicated than what you started with. They will mangle all your nice hand-written code, and the output will be impossible to adjust subsequently. Alas, Import was taken seriously by only a few, was always a bit buggy, ran on only a few platforms, and has disappeared.

Even with all its drawbacks, however, PostScript is the tool of choice for mathematical illustration. This book should therefore be of interest to a large group of people: (1) to any scientist who has to make up a fair number of technical illustrations, but who has found himself limited by what the other packages can do; (2) to any teacher, even at the level of secondary school, who would like to make mathematics a more comprehensible and better motivated subject to his students; (3) to students who want to see how even elementary mathematics that might have seemed useless when first seen can be applied to attractive real-world problems; (4) even to artists who might be interested in abstract designs with a mathematical component.

Perhaps the most surprising feature of this book is that it has been used successfully as a text for a third year undergraduate course in geometry. Students found that drawing in PostScript was a definite source of pleasure, once an initial breaking-in period had passed. From an instructor's point of view there are two great virtues to this: (1) The pleasure of producing pictures with PostScript is enough to drive students even to learning mathematics in order to do it. Drawing in PostScript forces students to become intimately familiar with coordinate systems and linear algebra. (2) Being able to draw easily on a computer makes it an intriguing process to interpret much of classical mathematics, for example Euclid's *Elements* or Newton's *Principia*, in graphical terms. In rendering mathematics in images, a person is forced to understand the essentials of an argument in the most intimate way, after all. This book does not deal directly with that possibility, but some references are given, and I hope that an interested instructor will easily figure out how to implement it. If I have not made projects of this kind available, it is because I like my own studenta to figure out things for themselves.

In spite of the difficulties of programming in PostScript, it is overall an enjoyable language to work with, largely because the final product can be truly beautiful. It is above all a complete graphics language—the only real limitation in working with it is ultimately your imagination. It is capable of extreme subtlety, brilliant colour, and infinite variety. What more does one want? But I must recall a warning, one written in an earlier time but still valid today:

Manuscripts should be decorated so that their appearance alone will induce perusal . . . beware that this art work does not become an end in itself.

From **In praise of scribes**
by Johannes Trithemius, c. 1492
(translation by Roland Behrendt)

Outline

The early chapters (1, 3–6) offer an introduction to basic features of the language. Chapters 2 and 12 offer accounts of coordinate geometry in 2D and 3D. Chapters 7–10 explore more sophisticated features of PostScript in 2D, as well as how mathematics and graphics algorithms interact in interesting ways. The remaining chapters explore three dimensions, using a library of PostScript procedures designed for the purpose. In particular, Chapter 15 is largely concerned with an exposition of the last result in Euclid's *Elements*, serving I hope as an example of how 3D pictures produced by PostScript can be used for exposition. Certain technical matters are dealt with in several appendices. I have have also included in an Epilogue a few brief remarks on the vast topic of style in mathematical graphics design.

Web site

The book refers to a great deal of PostScript code, samples as well as libraries that the reader may download and incorporate in his own programs. Links to this are to be found at

`http://www.math.ubc.ca/~cass/graphics/manual/`

along with other relevant material. In time, I will likely post an FAQ here. Errors in the text will be flagged and corrected at this site, too. Readers finding errors, or even just with suggestions for improvement, should send e-mail to `cass@math.ubc.ca`.

As an example of an undergraduate course that used this manual as a text, take a look at

`http://www.math.ubc.ca/~cass/courses/m308-02b.html/`

This includes links to student projects in PostScript, some of which are very good.

Acknowledgements

These notes have been written over a long period, and made available on the 'Net during that time, in various intermediate stages. I wish to thank above all my patient undergraduate students during those years for their help in improving them. I also wish to thank the many, many readers who have written to me from all over the world with encouragement and advice. Special thanks to the referees selected by Cambridge Press for invaluable suggestions. But above all I wish to thank Donald Knuth, not for any personal advice, but for writing all those wonderful papers and books in which he tells us by word and example that interesting mathematics can be found in nearly every corner of the computer.