Preface

These notes constitute part of a course in what might be called applied geometry. On the one hand, they amount to a self-contained introduction to elementary computer graphics—a very practical introduction since the reader will learn how to use the graphics programming language PostScript to produce sophisticated mathematical illustrations. On the other, they show how useful the geometry seen in more conventional mathematics courses can be in realistic circumstances. A third theme of the course is a mildly recursive one—it aims to show at least implicitly that computers can be used to produce illustrations to accompany mathematical arguments, and hopes to convince the reader to take the art of such illustrations more seriously than is commonly done. In short, it uses computer graphics as a tool for understanding geometry, and uses geometry as a tool to produce interesting computer graphics.

These notes make up a bit more than half of the original course notes, the half dealing with two-dimensional geometry. The other half, on three-dimensional geometry, are not included because that part is in a much less satisfactory state, and I am currently revising them. The point is that PostScript has no capabilities built into it to handle three dimensions, which means that I must develop and explain my own libraries for this purpose.

These notes are aimed at several audiences. On the one hand, they make up the latest version of notes that have been distributed for several years to students in a third year university course on elementary geometry. Many of the students in this course were Computer Science majors, often simultaneously taking a professional course in computer graphics. For them this course was an opportunity to see the mathematics underlying much of what they passed over rather rapidly in other courses. Many of the remainder were majors in Mathematics whose long range intention was to teach in secondary school. For them the course was an opportunity to learn how the computers could help explain mathematical principles in new ways, and also how the pleasures of doing even simple computer graphics could motivate much of the mathematics they had only learned in an abstract context previously.

But these notes should also be suitable for introducing useful graphics tools to a large number of practicing scientists who can use them to enhance their professional work.

The prerequisites are small amounts of linear algebra and calculus—so small, in fact, that a well motivated secondary school student might profitably read them.

Of course, it would be impossible to expect all parts of these notes to be equally interesting to all the groups mentioned above. With that in mind, I have tried to make it possible to skim quickly over any part of the notes already known. The text itself includes fragments of PostScript one can perhaps use immediately in one's own tool box. More elaborate reusable procedures are available through the Internet.

The geometrical topics covered are mostly elaborations of secondary school geometry and trigonometry, although this does not mean that the mathematical problems encountered are trivial for all readers. They include, among others, Pythagoras' theorem, vector dot-products, linear and affine transformations in two and three dimensions, Bezier curves, perspective, regular solids, and spherical geometry. Keep in mind when following mathematical demonstrations that you should be critical of the use of graphics in them.

The most common question I am asked about this course is why I chose PostScript rather than some more standard language such as C or Java. The principal reason for my choice is that PostScript is designed from the start as a graphics language, and even someone who has never done any previous programming can usually produce interesting and rewarding pictures after a few sessions of work. Another question asked is why I don't use a program like Geometer's Sketchpad, which is a popular choice for many similar courses. PostScript is admittedly somewhat clumsy to teach with, but its technical difficulties have never been a serious impediment, and the rewards of using it are very great. PostScript is a computer language which is both accessible to low level access and capable of very high quality output. It is far more versatile than all the computer geometry programs I have seen, and students seem actually to like its professional quality. I shall not, however, try to hide that there is indeed a serious drawback to using it in a course like this one. Although it is easy to use at the beginning, as one tries to make more and more complicated figures with it one has also to make up for oneself the structures that other languages provide for free. This handicap has not prevented my students from producing stunning illustrations.

The main point of mixing geometry and graphics programming in one course is that a reasonable well designed programming language forces one, without too much pain, to think carefully about aspects of geometry that might otherwise seem pointless. In my experience, the reaction among students to this hybrid course is on the whole extremely favourable, although even now there are every year at least a few students who find the course extremely heavy going. At the other end of the spectrum are a larger group who tell me that the course was by far the most interesting mathematics course they have ever taken.

I might mention to those instructors who are contemplating using it as I do—as the basis for a course with wide appeal—that such a course inevitably requires, at least at the beginning, an enormous amount of time and patience—with both machines and students! Anyone who uses computers for course work learns quickly that they are never entirely dependable. There are always incredibly annoying details that crop up continually from time to time. There seems in fact to be a minimum 'noise level' involved in dealing with computers that may never go away as long as we are still doing new things with them. It has always been a source of relief and amazement to me that most students are already familiar with this basic truth, and have acquired an almost inexhaustible patience in dealing with them in order to find the good stuff that lies inside. The technical problems, at any rate, seem to have been only a minor nuisance. The response of students to the course has been extremely positive.

One feature of the course does not really appear in these notes. Both PostScript and Java were used to accompany mathematical arguments with a wide variety of illustrations, many animated. These were used not only to explain mathematical principles, but to try to explain how illustrations could be used in this way. Much of the discussion in class thus amounted to a kind of art criticism. I'm sure almost anyone who has tried this sort of thing has been led to see himself as a kind of art director as much as a mathematics lecturer. In the course, PostScript programs were were often written and displayed right in class, in order to experiment with various techniques. It was sobering to see that what works well and what doesn't was rarely obvious *a priori*. At the end of the course, students have been required to produce an extensive project of some kind in PostScript. I encouraged them strongly to try to illustrate mathematical proofs in their projects, and it has been a pleasant surprise to me that many students have wound up using Euclid's text as a source of ideas! In recent years Euclid has become much more accessible, thanks to David Joyce's Internet version.

The standard reference for PostScript is the *PostScript Language Tutorial and Cookbook* (usually called simply the *blue book* to distinguish it from the red and green ones) published by Addison-Wesley. It is an excellent book, and you may find it useful, but you shouldn't need it to follow these notes. As a complete technical reference for PostScript, the *red book*—the *PostScript Reference Manual*, in either of two editions—is highly recommended, but it too ought not to be necessary. There is also a huge amount help with various aspects of PostScript available on the Internet. Sites I have found especially valuable are these:

UBC documentation:	`//gamba.math.ubc.ca/localdoc/`
Internet tutorial:	`//www.cs.indiana.edu/docproject/programming/postscript/postscript.html`
Technical advice:	`//www.comlab.ox.ac.uk/internal/dow/postscript/FAQ.html`
Internet tools:	`//sgk.phast.umass.edu/computing/sw/literate/postscript.html`
Euclid:	`//140.232.1.100/~djoyce/java/elements/elements.html`
Frivolity:	`//www.math.psu.edu/ward/lego/PostScript.html`

In addition, I maintain a Web site dedicated to animated geometrical proofs in Java, which is accessible from:

Java proofs:	`//sunsite.ubc.ca/Euclid/`

This will soon include digital images of Oliver Byrne's beautifully illustrated if eccentric edition of the first six books of Euclid. Even when Byrne's visual demonstrations are not successful, they can be used as a place to start,a nd suggest further development.

Some practical matters

Anyone reading these notes will want to have available a convenient PostScript interpreter of some kind. Many UNIX machines come equipped with one, and it is installed on nearly all University computer systems. But this course would not have made it off the ground if there didn't exist a free and freely available PostScript interpreter that works on nearly all platforms, so that my students could work at home. This program is called

ghostscript. All by itself it functions as a simple and adequate PostScript interpreter, but for these notes you will want a more convenient environment for PostScript development. This is provided by one of a couple of programs called **ghostview** or **GSView**, depending on your machine. These programs provide in effect a better user interface to **ghostscript**. Take a look at **http://www.cs.wisc.edu/˜ghost/**.

Incidentally, in the very near future I hope to post these notes and related material at the UBC SunSITE **sunsite.ubc.ca**, which is partly supported by Sun Microsystems and maintained by the UBC Mathematics Dep[artment. Comments will be welcome.