Chapter 11

2D graphics

This final chapter has three main purposes. First, it presents in detail Mathematica’s built-in two-dimensional graphics features. You’ve been learning to use some of these by trial and example. This chapter provides the thorough knowledge of the basics necessary to acquire real facility with two-dimensional illustration. Second, this chapter extends the built-in features, presenting for your use software tools the author has employed to produce hundreds of figures for books, papers, and courseware on various mathematical subjects, especially geometry. They’re organized in the Mathematica add-on package twoDG and included on the CD-ROM enclosed with this book. Third, this chapter presents many techniques and strategies used by software engineers to construct software tool kits of many kinds, and discusses in detail the construction of the corresponding Mathematica package file twoDG.m.
This chapter describes in some detail nearly all Mathematica’s built-in two-dimensional graphics features, but only some features of its related standard add-on packages. And it contains no discussion of three-dimensional graphics. Why? First, some features of the two-dimensional add-on packages are intended for producing special kinds of graphs—for example, bar graphs. They’re complicated, but you can use the help browser to figure them out if you know the basics. Adeptness in their use results not so much from mathematical knowledge or programming skill, but from experience and judgement concerning the psychological impact of data representation. They’re beyond the scope of this book. Second, this book’s readers are expected to be familiar with plane analytic geometry, vector algebra, and single-variable calculus, but it avoids much dependence on higher-dimensional mathematics. Any discussion of three-dimensional techniques that required only this background would be misleadingly sketchy. It would considerably increase the bulk of the chapter, or else require a less thorough coverage of two-dimensional techniques. The author decided to cover two-dimensional graphics thoroughly and well, and leave three dimensions to others. With background in three-dimensional analytic geometry and calculus, and the experience this chapter provides in general graphics techniques and the use and design of software tool kits, you should be able to use the help browser and other books\(^1\) to learn quickly about three-dimensional graphics.

Mathematica’s standard add-on packages and this book’s twoDG graphics package are tool kits. Conceptual tool kits are useful in mathematical as well as programming practice. In an earlier book on geometry, the author suggested a strategy for their use and development:

Construct a tool kit for a whole field of applications, but independent of any particular one. Make your tools general, so that you can use them for every problem you encounter in that field. That way, you’ll retain familiarity with them. Employ uniform methods across the entire tool kit, so it’s easy to use and document. Write your documentation intending that your tools be used by others with different backgrounds, and by yourself later when you don’t remember details of the underlying theory. Finally, develop your tool kit ahead of time, when you’re concentrating on its theory and all its details. You won’t be distracted by the immediate need for some special case. Having spent this effort in constructing your tools, you’ll feel justified to continue with thorough documentation that could prove invaluable later. If you follow these guidelines, you’ll create an intellectual tool kit that’s easy, effective, and safe to use in a wide variety of applications.\(^2\)

\(^1\) For example, consult Glynn and Grey 1999, part 9. I should also cite a more advanced reference.

Some additional guidelines apply specifically to software tools. First, isolate mathematical and programming problems. Programming difficulties distract you, and can keep you from finding the elegant mathematical approach that is in fact easiest to program! Second, strive for generality. Each application will differ a little from previous ones. A tool that’s too particular will need modification again and again and may lose a little robustness each time. Finally, test the code! Try it in as many different situations as possible.

Written under these guidelines, this book’s twoDG add-on package should prove useful to you. But it was developed to solve the author’s problems, not necessarily yours. By following the twoDG example you should be able to construct your own tool kits for application areas you want to investigate or results you want to present using Mathematica.

Section 11.1 discusses some general object-oriented programming (OOP) techniques that underlie Mathematica’s built-in graphics features as well as the construction of the twoDG package. Section 11.2 uses them with Mathematica’s Point objects, its simplest graphics primitives, to build some of the twoDG apparatus for doing analytic geometry calculations. This introduction to twoDG imparts some of its flavor, and demonstrates Mathematica techniques for algebraic calculation. It shows what a Point object really is, and how you can do more with it. For example, it shows that Point[{0,1}] is not a function call. Sections 11.3–11.6 cover basic Mathematica drawing techniques, starting with the Point and Line primitives and related directives in 11.3. Section 11.4 is a thorough and organized presentation of the Plot functions you’ve seen and used already, haphazardly and by trial and error, to produce Cartesian, parametric, and polar graphs as needed. Sections 11.5 and 11.6 cover details of color directives and graphics options that you can use to enhance these figures.

Sections 11.7–11.9 continue development of the twoDG package. Its analytic geometry capabilities are greatly expanded in 11.7, and 11.8 demonstrates them, verifying analytically a beautiful theorem you may have met in a deductive geometry course. Section 11.9 presents some more involved features that you can use to embellish geometric figures. Mathematica’s rather meager animation features are described in 11.10. The book can’t show moving diagrams, of course, but all the code from this chapter is included on the CD-ROM enclosed with it, and you can execute that to see the demonstrations.
The final section, 11.11, is not about graphics, but about packaging. It shows how to reorganize a Mathematica notebook as an add-on package, using twoDG as an example.

In addition to the chapter introduction that you just read, this file contains material that won’t appear in the published version.

• this material between horizontal rules;
• the box after the chapter title, which includes the table of contents with page count;
• after the next rule, the list of references for this chapter; and
• after the final rule, a list of problems that remain concerning this chapter.

The most important of problem, though, needs to be faced here. This chapter is probably too long. If so, it can be split after 11.6. This introduction would have to be split and rewritten, which would add about two pages. What would result would be a “basics” chapter and an “advanced” chapter.

References


1. I need a reference to the filling theorem in 11.3.

2. I need a reference for 3D graphics more advanced than Glynn and Grey.
Problems remaining

General problems
a. The chapter-length problem discussed earlier in this file.

Problems with accompanying notebook files
a. I haven’t yet made Mathematica input and output look the way I want.
b. It’s possible to hide the cell bracket for the initialization groups. If you know what you’re doing you can unhide it because double-clicking where it ought to be works. This would be a way to avoid readers’ inadvertently opening initialization cell groups and getting lost. But it’s a bit big-brotherish and would annoy some.

Word processing problems
a. My code style looks bad against my Title style in headings, even if I twiddle its size and boldness. I may have to change to a different Helvetica-type font for code in headings.
b. The bold Math1 font is ok for Greeks in code and matches Mathematica on screen. They seem to look ok for Greeks in text, too. This might be a better choice than my current one, SymbolPropBT. On the other hand, it didn’t survive my last quest for Greeks. Check it out.
c. A *.jpg bitmap file worked great in 11.10 when other formats failed. I set it to the exact size needed. Maybe I should use that format generally, if a more recent Mathematica doesn’t correct the Export problems.
d. Maybe I should set the ImageSize option default to 3 inches = 216 pt and avoid resizing figures.
e. Maybe I should put figures on even numbered pages in left columns, not right.
f. I may have some left and right double quotes in code, but they should be symmetric. Check it out.
g. Some lines in 11.8 and 11.9 with multiple fonts came out ok in WordPerfect but bumpy in Acrobat. Check it out.
h. The indexer doesn’t distinguish upper- and lowercase. I may have to do that by hand.

Problems with this chapter introduction
b. I need a reference for 3D graphics more advanced than Glynn and Grey. Wagon?
Problems listed by section

1. Object-oriented programming
2. Manipulating points
   a. \texttt{midpoint[P_,Q_,t_]:=(1-t)P + t Q} would be a useful definition. Should I add it? Perhaps as an exercise? Perhaps put it in the package and have the exercise ask what it does? I used it, unnamed, in solving exercise 11.9.8 (with \( t < 0 \), I think). Is there a more appropriate name? Perhaps \texttt{tPoint}?
   b. Should I note that responses don’t keep up with overloads, but ??? responses do? And that only other overloaded \texttt{System`} symbol in the book is \texttt{Circle}?
3. Using graphics primitives and directives
   a. I need a reference for the theorem about the filling algorithm.
   b. I need to check out the dashing line bug in a more recent version.
4. \texttt{Plot} functions
5. \texttt{Color}
   a. I forgot to mention that the names in \texttt{Graphics`Colors} are actually directives, easy to use in display lists, and that I’m therefore loading that package with \texttt{twoDG}.
   b. I need to solve exercise 4. Also, I probably need to quote the algorithm in the statement of the problem, because the source won’t be accessible to all.
6. Graphics options
   a. I need exercises for this section. Since they have to do with modifications of earlier figures, I’ll provide them very late in the writing process.
7. Analytic geometry with \texttt{twoDG}
   a. The derivation of the formula for the distance from a point to a line is opaque. I should merely refer to an analytic geometry text, but I can’t find an appropriate one. (I forgot to put this formula in \texttt{Methods}!)
8. Analytic geometry demonstration
   a. Some formulas on pp. 4-5 with multiple fonts came out OK in \texttt{WordPerfect} but bumpy with \texttt{Acrobat}. Investigate.
9. Finishing touches with \texttt{twoDG}
   a. Some formulas on pp. 3, 6 came out OK in \texttt{WordPerfect} but bumpy with \texttt{Acrobat}.
   b. I forgot to implement \( A \) and \( \pi \) for \texttt{Circle} objects. Exercise?
10. Multiple graphics and animation
    a. I need to construct exercise 2 at some later time.
    b. See if I can find out how to program \texttt{animateWithPauses}.
11. Packaging
    a. See how I can avoid the insertion at the end of \texttt{twoDG.nb} that got me around a bizarre glitch.
Problems to consider concerning earlier chapters

1. I need to distinguish early between a formula that looks like a function call and one that really is. The latter is rather informal. That is, much of the time, Mathematica evaluates a formula and gets another formula via substitution rules, stopping when no more rules apply. But it may get to the point where the evaluation can perform some numerical computations, and return a computed answer. That’s surely a function call. What’s not clear is the extent to which the application of rules represents application of some function whose arguments are functions or formulas. This comes up in 11.1.

2. I have to discuss full form early.

3. Early chapters have to give sufficient examples of the use of Plot functions, loops of various sorts, modules, etc., that those won’t be new in chapter 11.

4. I have to provide in earlier chapters some figures that can be improved or mangled in the chapter 11 exercises.