

San Francisco State University
Mathematics Club

Illustrating *Methods of Geometry*

James T. Smith

Friday, 5 May 2000
12:10-13:00, TH935

David Meredith's computational software $X(Plore)$ was exactly what I needed to produce the figures for my new book *Methods of geometry*. Typical examples appear below. $X(Plore)$ includes only the most basic graphics software tools, but Meredith implemented them just right. I'll show how his design let me use elementary coordinate geometry and vector algebra to extend $X(Plore)$ to a more powerful graphics system. Now I can easily construct virtually any 2D line graphic, such as figure 5.8.3. With some difficulty I produced selected 3D perspective graphics, such as figure 8.4.18. I'll discuss what 3D features I had to omit, and how I provided crude substitutes that worked.

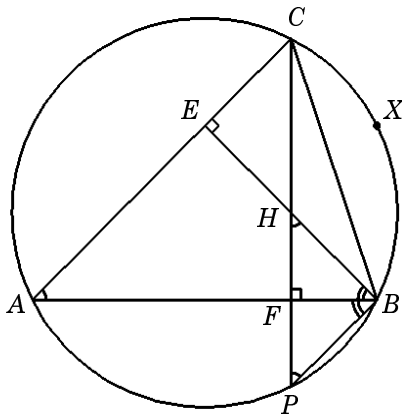


Figure 5.8.3

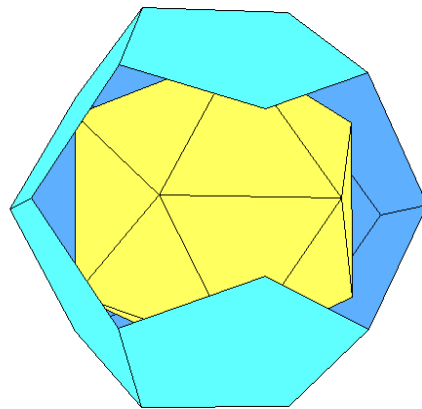


Figure 8.4.18

1. Show the flyer with *WordPerfect* and circulate copies of *Methods of geometry*.
 - a. Show how to write the labels *over* the graphics: 1 min.
 - b. Nothing beats a good word processor for producing text.
 - c. How are the graphics produced?
2. *X(Plore)*'s built-in graphics features
 - a. Execute and explain [2a.xpl](#): 10 min.
3. How I extended that to produce 2D line drawings
 - a. Execute and explain [2b.xpl](#): 15 min.
 - b. With the overhead projector, put a large fig. 5.8.3 beside the screen.
 - c. Explain [050803.xpl](#): 6 min. Chalk the features on the figure as they're produced.
4. Display capture and importation into *WordPerfect*.
 - a. Execute [050803.xpl](#).
 - b. <Alt-PrintScreen> it to the clipboard, then <Ctrl-V> that to a blank *WordPerfect* page. Invert the colors, and trim it: 2 min.
 - c. This is 640x480 dots, smooth enough when reduced to 1/2-page width, but somewhat jaggy at full-page, as in fig. 5.10.10. I decided to live with that.
 - d. Meredith's latest *X(Plore)* produces smoother graphics but won't execute old *X(Plore)* programs. I must convert my routines to *Mathematica*.
5. How I extended the 2D package to produce 3D drawings
 - a. [Click here](#) for expanded outline: 10 min.
6. How I computed 3D coordinates of regular polyhedron vertices: 0 min.
 - a. The 6 unit vectors on the axes determine the vertices of a *regular octahedron* (with 12 edges).
 - b. On each edge are two points that divide it in golden ratio. Appropriately selecting one of these on each edge (this is tricky) gives you the vertices of a *regular icosahedron* \mathfrak{I} (with 20 faces).
 - c. The centers of \mathfrak{I} 's faces form the vertices of a regular dodecahedron \mathfrak{D} (with 12 faces).
 - d. The centers of \mathfrak{D} 's faces form the vertices of a regular icosahedron \mathfrak{I}^* . Fig. 8.4.18 shows \mathfrak{D} and \mathfrak{I}^* .
7. How I selected which segments to draw in fig. 8.4.18 and which regions to shade: 4 min.
 - a. I made no general routines. I eyeballed.
 - b. Only parts of some edges are drawn. These are determined by intersecting lines that join display points \mathbf{P}^* corresponding to selected 3D points \mathbf{P} .
 - c. Most 3D figures required much trial and error to locate the eye, display origin, and rightward vector properly. Figs. 8.4.18 and 8.4.19 were by far the hardest figures to draw.
 - d. Keeping track of which edges belong to which faces was very hard. I often changed colors in the program until I identified an eyeballed segment that I needed to manipulate, then changed back after fixing the figure.

5. How I extended the 2D package to produce 3D drawings
- a. 3D data structures
 - i. A point is a column of three coordinates.
 - ii. A line is represented by a row of two points. (Not very good for computation, but I don't need much, and this works.)
 - iii. The plane with equation $ax + by + cz + d = 0$ is represented by the row $\{a, b, c, d\}$.
 - b. Renaissance Italians learned to incorporate the display as a window in the 3D environment, and Dürer published their method in 1525. Go to <http://www.newcastle.edu.au/departments/fad/fi/woodrow/durer-a.htm> and look at the lower figure where the line of sight is perpendicular to the display window.
 - c. The hardest mathematical task is to construct the desired object, so I locate it first in the 3D coordinate system. (I get the model situated first.) To set up a display coordinate system I specify

- (1) 3D coordinates of the eye,
- (2) 3D coordinates of the origin of the display coordinates.

These determine the line of sight g . I assume display plane $\mathcal{E} \perp g$.
And I specify

- (3) 3D coordinates of a vector in \mathcal{E} pointing rightward.

I use a cross-product to get a vector in the display plane pointing upward. The two vectors give me display x, y coordinate axes. Their scales are determined by the 3D environment.

- d. A 3D point $\mathbf{P} = \{x; y; z\}$ corresponds to the intersection \mathbf{P}^* of the display plane with the line joining \mathbf{P} to the eye. The display coordinates of \mathbf{P} are simply the coordinates $\{x^*, y^*\}$ of \mathbf{P}^* in the display coordinate system. It's an easy analytic geometry exercise to derive formulas for x^* and y^* in terms of x , y , and z .
- e. Regimen:
 - i. construct the object in 3D,
 - ii. locate the eye to see appropriate details,
 - iii. locate display window and the rightward direction as Dürer would,
 - iv. adjust the *X(Plore)* **Window** to show all of the object,
 - v. draw all desired segments between display points \mathbf{P}^* corresponding to appropriate 3D points \mathbf{P} .
- f. I made a package of *X(Plore)* routines to simplify this, but it's not well developed. It's targeted at the specific 3D illustrations in the book. And it severely taxes the programming and memory-management features of *X(Plore)*. I suspect the *Mathematica* version will be cleaner.

[Click here](#) to return to the main outline.

Props

1. Copies of the flyer for everyone
2. Several copies of *Methods of geometry*
3. Overhead slide of Figure 5.8.3

Place here the imported copy of fig. 5.8.3: