IN "PLANE" VIEW

Peter Allison, Louis Basel, Oana Ciorbea, Shantanu Dhaka, Brooks Gordon, Rebecca Hudson, Betsy Lee, Derek Lee, John Playforth, Michael Xiao

Introduction by: Jon Choate

This article was produced as an assignment for a computer graphics course that I am currently teaching to high school seniors. The assignment was to produce an image of a cube in perspective with hidden lines removed. The article describes the mathematics the students used to produce the image. They produced the actual image using Microsoft Excel. If any of you would like a copy of the spreadsheet, send your email address to me at jchoate@groton.org and I will email it to you. We are also in the process of putting up a website dedicated to applications of secondary mathematics in computer graphics. If you have any material you would like to contribute to this site, please contact me and I would be glad to add it to the site if it is suitable. There will be more about the site in the next edition of *Consortium*. ❑

In the world of computer graphics, the essential goal is to portray a three-dimensional world on a two-dimensional surface. The task presented to programmers is to accomplish this goal in the most efficient way possible. As a class, we tackled this problem in the context of a cube. In order to portray the cube on a two-dimensional computer screen, we set up a projection scheme that maps three-dimensional points onto a single plane while retaining perspective. Essentially, we needed a point to serve as an "eye," and a plane onto which all points will be mapped. In our case, we assigned the xy plane to be our projection plane and the point $(x_{eye}, y_{eye}, z_{eye})$ to be our eye.

To properly project our cube, we first needed to find the projected image of each of the vertices. To find these projected vertices, we connected each preimage point to the eye. The intersection of that line and the xy plane was the projected point. We let $(x_{point}, y_{point}, z_{point})$ be a point in three-dimensional space on the opposite side of the xy plane as the eye $(x_{eye}, y_{eye}, z_{eye})$ as seen in **Figure 1**, and we derived the formulas for its projection as follows.

The parametric equation of the line between the eye and our point is $(x, y, z) = (1 - t) * (x_{eye}, y_{eye}, z_{eye}) + (t) * (x_{point}, y_{point}, z_{point})$. Our projected point was on this line; specifically, it existed where $z = 0$. To solve for t when $z = 0$, we solved the separate z equation, $z = (1 - t) * z_{eye} + (t) * (z_{point})$. After some algebra, this left us with the equation $t = (-z_{eye})/(z_{point} - z_{eye})$. Once we calculated t, we inserted that value back into our original equation and found the equation for the x and y coordinates of the projected image. The equations we found were $x = (1 - t) * x_{eye} + t * x_{point}$ and $y = (1 - t) * y_{eye} + t * y_{point}$. To simplify our calculations we placed our eye on the z-axis so that $x_{eye} = y_{eye} = 0$, and our final equations for the projected points were $x = (-z_{eye})/(z_{point} - z_{eye}) * x_{point}$ and $y = (-z_{eye})/(z_{point} - z_{eye}) * y_{point}$. We used these equations to calculate the projected image for a vertex of our cube, and we repeated this for each of the subsequent vertices. We plotted these points on a graph, and connected the respective vertices to create a wire-frame image of the cube.

However, a wire-frame is often not the way in which a programmer wishes to display a three-dimensional object. When looking at a solid cube we see full faces, not a wire-frame, and it is possible to see a maximum of three faces as the others are hidden behind the visible faces. To account for this in our projection, we needed to find a method for identifying hidden faces. The only faces of an object that we can see are those that are tilted towards our eye. In other words, a face is visible only if the vector perpendicular to the face (the normal vector) forms an angle less than 90 degrees with a vector from a point on the face to the eye. For example, in **Figure 2**, the vector from the eye to the face BFGC forms an angle greater than 90 degrees with the normal to this face, so this face is not visible. Conversely, the vector from the eye to the face AEHD has an angle of less than 90 degrees with the normal to this face, so this face is visible. The dot product serves as a convenient test to see if the angle between our two vectors is greater or less than 90 degrees. If the dot product of these two vectors is positive, the angle is less than 90 degrees and the face is visible. Conversely, if the dot product is negative or zero then the face is hidden.

To calculate the normal vector, we took the cross product of two edges of each face (Edge 1 x Edge 2 = Normal). In order to ensure that the normal vector was directed away from the center of the cube, we made use of the right hand rule.

Using the normal vector of each face, we took the dot product of the normal vector and the vector from a point on

that face to the eye. Using our dot product test, we determined if the face was visible. All visible faces were plotted and filled in, while the faces with non-positive dot products were not displayed. We were thus able to create an image of the cube that is both accurate in terms of perspective and realistic in its display of solid faces.

Having created our projection scheme, we wanted, finally, to develop a test that showed that it maintained perspective. The test that we created is as follows: we imagined that we placed parallel lines within our three-dimensional space and then found their projected images using our scheme. The projection of two parallel lines onto a two-dimensional viewing plane using our projection scheme should result in the convergence of the lines at a vanishing point. We used our viewing plane, which consisted of the plane $z = 0$ (the xy-plane) and points of the form $(x, y, 0)$. Two lines are parallel if they have the same directional vectors. Therefore, the set of lines, $(x, y, z) = (x_1, y_1, z_1) + r(a, b, c)$ are parallel regardless of x_1, y_1, and z_1 if a, b, and c are constant. Thus the projection of the point (x, y, z) as r goes to infinity should be the same no matter what initial point (x_1, y_1, z_1) is chosen. To demonstrate that the projections are independent of (x_1, y_1, z_1) as r goes to infinity and thus show that all parallel lines go to the same vanishing point, we first considered x' and y', which we assumed to be the x and y projections of a point along the line $(x, y, z) = (x_1, y_1, z_1) + r(a, b, c)$ for any r.

$$x' = \frac{z_{eye} * (x_1 + ar)}{-z_{eye} + (z_1 + cr)}$$

$$y' = \frac{z_{eye} * (y_1 + br)}{-z_{eye} + (z_1 + cr)}$$

Here, we created two vector form equations based on the projection scheme detailed above that we derived from the point form equations:
$(x', y') = (t * x_1, t * y_1)$, with

$$t = \frac{-z_{eye}}{z - z_{eye}}.$$

We then took the limit as r goes to infinity as a way to find the projection of the vanishing point.

$$\lim_{r \to \infty} \frac{z_{eye} * (x_1 + ar)}{-z_{eye} + (z_1 + cr)} = \frac{-z_{eye} * a}{c}$$

$$\lim_{r \to \infty} \frac{z_{eye} * (y_1 + br)}{-z_{eye} + (z_1 + cr)} = \frac{-z_{eye} * b}{c}$$

The constant $-z_{eye}$ in the denominator and the constants x_1, y_1, and z_1 of each equation all become negligible as r approaches infinity. The equation for the projected vanishing point of any line $(x, y, z) = (x_1, y_1, z_1) + r(a, b, c)$ is thus:

$$(x, y) = \left(\frac{z_{eye} * a}{c}, \frac{z_{eye} * b}{c}\right).$$

We therefore found, according to the equation above, that only the direction of the vector (a, b, c) and the initial placement of the eye z_{eye} affect the location of the vanishing point. Thus, two lines of the form $(x, y, z) = (x_1, y_1, z_1) + r(a, b, c)$ have the same vanishing point so long as a, b, and c are the same for both lines. In Figure 3, we show how parallel lines, such as DH and CG, have a common vanishing point in our projection scheme as perspective demands. Thus, our test of our projection scheme was successful. ❑

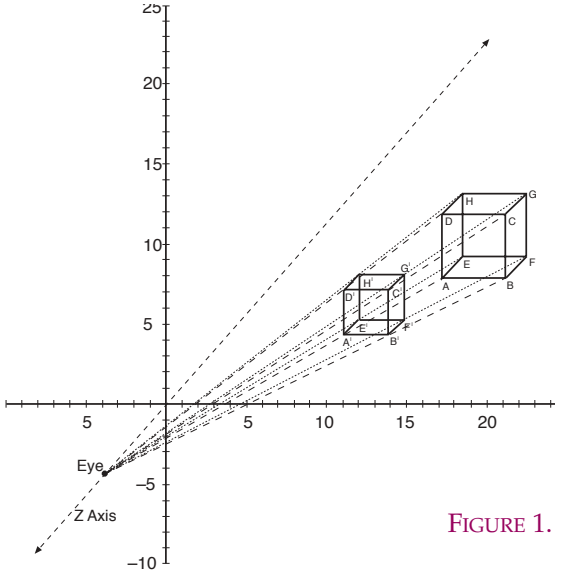

FIGURE 1.

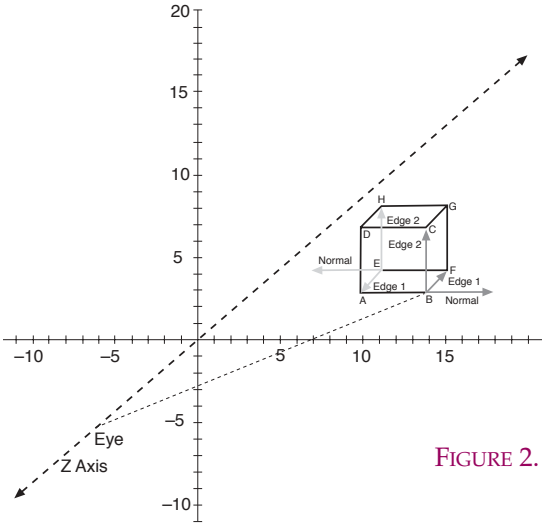

FIGURE 2.

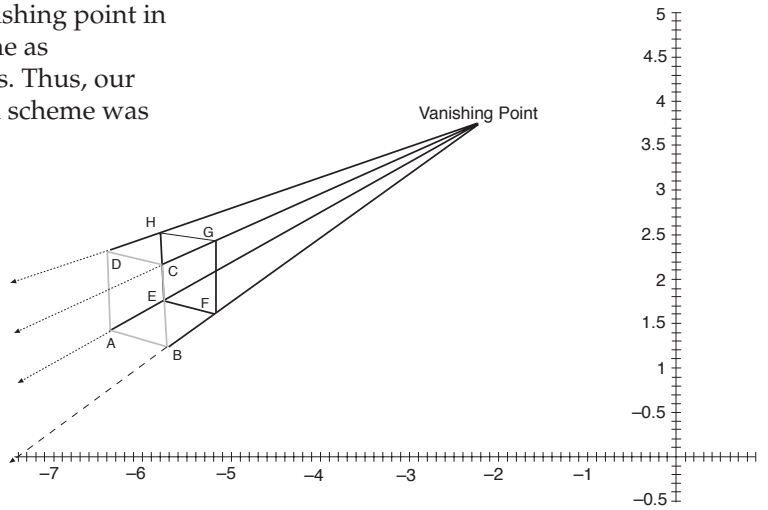

FIGURE 3.

Send solutions to old problems and any new ideas to the Modeler's Corner editor: Jonathan Choate, Groton School, Box 991, Groton, MA 01450.