3D Computer Graphics

Jared Kirschner

November 8, 2010

Abstract

We are surrounded by graphical displays—video games, cell phones, television sets, computer-aided design software, interactive touch screens, instrument panels, and countless others. In order to understand how image data can be manipulated and rendered on a screen to produce a coherent, realistic image, we need only to understand some basic concepts of linear algebra.

To produce an image, we will need a set of vectors (points) defined in the relevant dimension ($\mathbb{R}^n$ where $n \geq 2$), mathematical information defining the connections between points (in the form of lines or curves), and information concerning how to fill the region bounded by the sets of connected points. [1] For the purposes of this paper, we will be focusing on the vectors in $\mathbb{R}^3$ which define the nodes of an image. By performing linear transformations on the set of vectors through matrix multiplication, we can translate, rotate, scale, and distort an image in $\mathbb{R}^n$, among other possibilities. But ultimately, to display the graphics in two dimensions, we must perform a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^2$. Through the use of perspective projections, we can create a two-dimensional set of data which appears to the observer as an object in n-dimensional space (usually $\mathbb{R}^3$). By combining these transformations, the complex animations and visuals we have grown so accustomed to in our modern world are made possible.

1 Vector Data

To represent a three-dimensional object, it is trivial to state that we need at least three dimensions of data per vector. However, in the case of computer graphics, we actually use $(n+1)$-dimensions of data to store and manipulate n-dimensions of data. A vector $\vec{p} = (x, y, z)$ in $\mathbb{R}^3$ can be represented as a plane in $\mathbb{R}^4$ at 1 unit above the xyz-plane by the homogenous coordinates of $\vec{h} = (X, Y, Z, H)$ where $H \neq 0$ and:

$$x = \frac{X}{H}, \qquad y = \frac{Y}{H}, \qquad z = \frac{Z}{H} \tag{1}$$

Using a homogenous coordinate system provides two key advantages. First, it allows scaling and translation to be achieved by matrix multiplication. Second, it allows for perspective projections which make a two-dimensional image appear three-dimensional.

2 Data Transformations

Before processing the vector data to produce a realistic image, it is necessary to perform transformations on the object such that it has the desired properties in $\mathbb{R}^3$. Translation, rotation, shearing, and scaling are examples of transformations one might want to do before processing the data into an image. As shearing and scaling are fairly elementary operations, they will not be discussed here.

2.1 Translation

In a non-homogenous coordinate system, translation must be accomplished by matrix addition rather than matrix multiplication. This is because $T(x, y, z) \mapsto (x + t_1, y + t_2, z + t_3)$ where $\vec{t} = (t_1, t_2, t_3)$ is the translation vector in $\mathbb{R}^3$ is not a linear transformation. All matrix transformations are linear transformations, therefore, the transformation T cannot be obtained from matrix multiplication. As "the mathematics of computer graphics is intimately connected with matrix multiplication", we will instead perform translation by matrix multiplication using homogenous coordinates: [2]

$$\begin{bmatrix} X + t_1 \\ Y + t_2 \\ Z + t_3 \\ H \end{bmatrix} = \left[\begin{array}{c|c} I_3 & \vec{t} \\ \hline 0 & 1 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ H \end{bmatrix} \tag{2}$$

... where I_3 is the 3×3 identity matrix, and $\vec{t}$ is the translation vector in $\mathbb{R}^3$. The partitioned matrix given in Equation 2 can be extended to any dimensional space $\mathbb{R}^n$. However, in this particular case, we are examining how a vector in $\mathbb{R}^3$ represented by homogenous coordinates can be successfully translated using matrix multiplication.

2.2 Rotation

Let us examine a vector $\vec{v}$ in $\mathbb{R}^2$. In order to define a linear transformation T such that $T(\vec{v})$ is a rotation of $\vec{v}$ by θ radians, we must find the standard matrix A of the transformation such that $T(\vec{v}) = A\vec{v}$. The columns of the standard matrix A will be the transformations of the columns of the identity matrix, as shown below.

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = v_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + v_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = v_1 e_1 + v_2 e_2$$

Therefore,

$$T(\vec{v}) = T(v_1 e_1 + v_2 e_2) = v_1 T(e_1) + v_2 T(e_2) = [T(e_1) \quad T(e_2)]\vec{v} = A\vec{v}$$

If we rotate the columns of the identity matrix by an angle θ, it can be seen that $T(e_1) = (\cos(\theta), \sin(\theta))$ and $T(e_2) = (-\sin(\theta), \cos(\theta))$ (refer to Figure 1). Therefore:

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

.

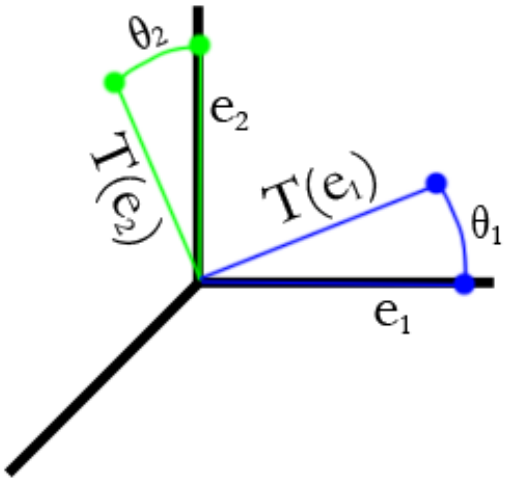

Figure 1: Rotation of the columns of the identity of the xy-plane about the z-axis.

A similar analysis can be used to determine the standard matrix A for a rotation in $\mathbb{R}^3$ along any axis.

$$T(\vec{h}) = \left[\begin{array}{c|c} A & 0 \\ \hline 0 & 1 \end{array}\right] \vec{h}$$

Unfortunately, such a transformation only rotates the vector about the origin. In computer graphics, the ability to rotate about a specific point is generally more useful. To accomplish this, we can combine translational and rotational transformations. For a given homogenous coordinate $\vec{h}$ and a point of rotation $\vec{p}$, we will translate $\vec{h}$ by $-\vec{p}$, thus moving the point of rotation $\vec{p}$ to the origin. After performing the rotation about the origin, we can move the point of rotation from the origin back to its original position

by translating by $\vec{p}$. Therefore, the composite transformation $S(\vec{h})$ is equal to $T(\vec{h} - \vec{p}) + \vec{p}$ (see Equation 3).

$$S(\vec{h}) = \left(\left[\begin{array}{c|c} I_3 & \vec{p} \\ \hline 0 & 1 \end{array} \right] \left[\begin{array}{c|c} A & 0 \\ \hline 0 & 1 \end{array} \right] \left[\begin{array}{c|c} I_3 & -\vec{p} \\ \hline 0 & 1 \end{array} \right] \right) \vec{h} \tag{3}$$

3 Image Transformations

After an object has the desired properties in $\mathbb{R}^3$, it must be transformed into $\mathbb{R}^2$ for display. The easiest way to do this would be to perform a simple projection of each vector onto a two-dimensional plane such as the xy-axis. Unfortunately, this would not produce a very realistic image. To create a realistic image, we must take into consideration factors such as the position of the observer (perspective) and of light sources (shadows), among others.

3.1 Perspective Projection

Let us define a point of observation $\vec{p}_c = (b, c, d)$ in $\mathbb{R}^3$ from which the three-dimensional object is being viewed. This point is referred to as the *center of projection*. If we project each point of the image $\vec{p}$ in $\mathbb{R}^3$ onto the xy-plane from the center of projection, we can create a two-dimensional representation of the three-dimensional image. To project a point onto a plane, we must find the location at which the line through $\vec{p}$ and $\vec{p}_c$ intersects the desired plane (see Figure 2).

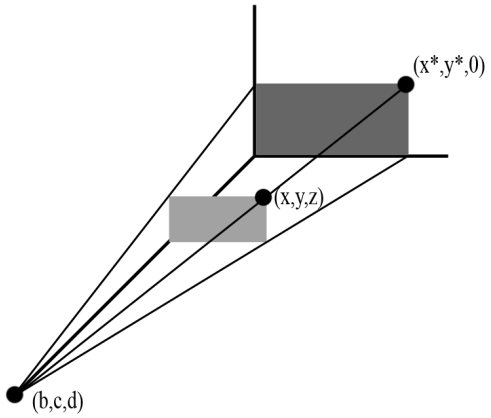

Figure 2: Perspective projection of a point (x, y, z) in $\mathbb{R}^3$ onto the xy-plane from the center of projection (b, c, d).

The line $\vec{l}(t)$ through any two vectors $\vec{v}_1$ and $\vec{v}_2$ can be defined as $\vec{l}(t) = \vec{v}_1 + t(\vec{v}_2 - \vec{v}_1)$. As such, we can define a line through the center of projection $\vec{p}_c$ and a point $\vec{p}$ in $\mathbb{R}^3$ as:

$$\vec{l}(t) = \vec{p}_c + t(\vec{p} - \vec{p}_c) = \begin{bmatrix} b + t(x-b) \\ c + t(y-c) \\ d + t(z-d) \end{bmatrix} \tag{4}$$

We can find the point $\vec{p^*} = (x^*, y^*, z^*)$ where $\vec{l}(t)$ intersects with the xy-plane by defining $z^* = 0$. This allows us to solve for t:

$$0 = l(t)_3 = d + t(z-d) \quad \therefore \quad t = \frac{d}{d-z}$$

Now that we have solved for t, from Equation 4 we can define the point $\vec{p^*}$:

$$\vec{p^*} = \begin{bmatrix} b + \frac{d}{d-z}(x-b) \\ c + \frac{d}{d-z}(y-c) \\ d + \frac{d}{d-z}(z-d) \end{bmatrix} = \begin{bmatrix} \frac{x - \frac{b}{d}z}{1 - \frac{1}{d}z} \\ \frac{y - \frac{c}{d}z}{1 - \frac{1}{d}z} \\ 0 \end{bmatrix} \tag{5}$$

For the homogenous coordinate $\vec{h} = (X, Y, Z, H)$, we can express its projection onto the xy-plane as $\vec{h}^* = \left(\frac{x - \frac{b}{d}z}{1 - \frac{1}{d}z}, \frac{y - \frac{c}{d}z}{1 - \frac{1}{d}z}, 0, H\right)$. In a homogenous coordinate system, scalar multiplication does not affect the vector (x, y, z) in $\mathbb{R}^3$ (refer to Equation 1). As such, we can multiply $\vec{h}^*$ by the scalar $\left(1 - \frac{1}{d}z\right)$ to eliminate the fractions. We can express the perspective transformation U_p as $U_p(X, Y, Z, H) \mapsto \left(x - \frac{b}{d}z, y - \frac{c}{d}z, 0, H(1 - \frac{1}{d}z)\right)$. Let G_p be the standard matrix of transformation U_p; the matrix equation $G_p\vec{h} = \vec{h}^*$ can be expressed as follows:

$$\begin{bmatrix} 1 & 0 & -\frac{b}{d} & 0 \\ 0 & 1 & -\frac{c}{d} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ H \end{bmatrix} = \begin{bmatrix} X - \frac{b}{d}z \\ Y - \frac{c}{d}z \\ 0 \\ H(1 - \frac{1}{d}z) \end{bmatrix} \tag{6}$$

To display the image in $\mathbb{R}^2$, we simply divide the vector by H^* (the fourth element of the vector) as per Equation 1 to force the fourth entry to 1. When $H = 1$, $(X, Y, Z) = (x, y, z)$, allowing for direct mapping. As the z-component of this vector is zero because we have projected it onto the xy-plane, it can readily be displayed as $\mathbb{R}^2$ by dropping the last two elements (Z and H). To accomplish this with matrix multiplication, we can left-multiply the vector by the following matrix to produce a projection onto $\mathbb{R}^2$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

3.2 Realistic Shadows

To create realistic shadows, we need to define at least one source of lighting $\vec{p_s} = (b, c, d)$ and a surface of projection. In most cases, this surface of projection is the ground. For our purposes, we will assume a flat surface at $y = 0$, but the methods discussed below can be used to determine the projection of a shadow onto any surface.

As in Section 3.1, we begin by defining the line $\vec{l}(t)$ which passes through a point $\vec{p}$ and the surface of projection. In this case, $\vec{l}(t) = \vec{p_s} + t(\vec{p} - \vec{p_s})$. To determine the point $\vec{p^*} = (x^*, y^*, z^*)$ at which $\vec{l}(t)$ passes through the xz-plane ($y^* = 0$), we can solve for the parameter t:

$$0 = l(t)_3 = c + t(y - c) \quad \therefore \quad t = \frac{c}{c - y}$$

By the methods discussed in Section 3.1, we can express the perspective transformation U_s as $U_s(X, Y, Z, H) \mapsto \left(x - \frac{b}{c}y, 0, z - \frac{d}{c}y, H(1 - \frac{1}{c}y)\right)$. Let G_s be the standard matrix of transformation U_s; the matrix equation $G_s\vec{h} = \vec{h^*}$ can be expressed as follows:

$$\begin{bmatrix} 1 & -\frac{b}{c} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{d}{c} & 1 & 0 \\ 0 & \frac{1}{c} & 1 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ H \end{bmatrix} = \begin{bmatrix} X - \frac{b}{c}y \\ 0 \\ Z - \frac{d}{c}y \\ H(1 - \frac{1}{c}y) \end{bmatrix} \tag{7}$$

To display an image with a shadow, we will perform a perspective projection (refer to Section 3.1) on both the image data $\vec{h}$ and its shadow projection onto the xy-plane $\vec{h^*}$. After the perspective projection, we will once again divide by H^* as per Equation 1 and remove the z-component of the resulting vector for display in $\mathbb{R}^2$ by a simple projection.

4 The Results

Now that we have covered some of the basics of computer graphics, we will discuss how to produce an animation. We start by defining our image as a collection of vectors $\vec{h}$ in a homogenous coordinate system. These vectors will be arranged columnwise in a matrix M_h to produce our data matrix such that $M_h = [\vec{v}_1 \dots \vec{v}_m]$ where m is the number of vectors in the image. To construct

an image, we also need to define the connections between vectors. This can be done with an $m \times m$ adjacency matrix K where the position K_{ij} denotes whether or not a connection exists from $\vec{v}_i$ to $\vec{v}_j$. Though this matrix can become very large depending on the number of vectors in an image, it can be compressed by conversion to a sparse matrix. However, sparse matrices are beyond the scope of this discussion.

Figure 3 shows an animation of a wire-frame car where the point of observation pans from $\vec{p}_{c_1}$ to $\vec{p}_{c_2}$ with a light source at $\vec{p}_s$. The wire-frame car is represented as a matrix M_h with columns of the homogenous coordinates of each vector in the image. A series of transformations is performed on M_h before it is displayed as a two-dimensional image. We first perform data transformations which simply map $\mathbb{R}^4$ onto $\mathbb{R}^4$. To display the image, we perform image transformations from $\mathbb{R}^4$ to $\mathbb{R}^2$.

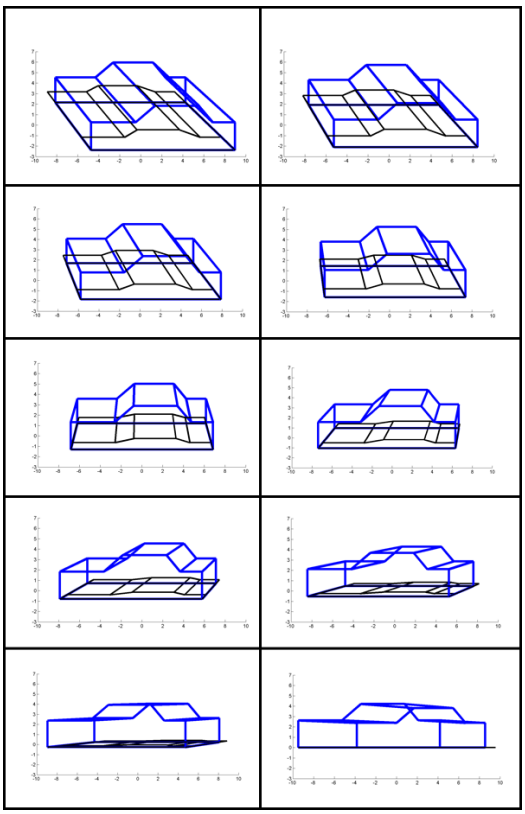

Figure 3: A series of frames showing the view of a wire-frame car as the point of observation pans from $(-50, 50, 50)$ to $(50, 0, 50)$ with a light source at $(0, 1000, 500)$. The wire-frame car itself is about 13 units wide, 4 units tall at its highest point, and 5 units wide.

To produce an animation of the type described above, we have several options. To make an object appear that it is moving, all that matters is the relative motion between the observer and the observed. As such, we can produce the same animation by translating the object with a data transformation or by moving the point of observation the same amount (thus changing the perspective projection—an image transformation). A similar argument can be used to show that a perceived rotation can be obtained with a rotation (data transformation) or a change of perspective (image transformation). In this particular implementation, I have chosen the latter option. This is because the light source is defined in the reference frame of the observer. Therefore, for the car to have a constant shadow (stationary object with a moving observer), we must use image transformations. If we wish to have a dynamic shadow (moving object with a stationary observer), we must use data transformations. Though there will be no observable difference between the method used for a light source at a significant distance on a relative scale (such as the sun), a significant difference between methods will be observed for local light sources (such as candles, light bulbs, et cetera).

To produce the image of the car in two-dimensions, the following equation was used:

$$M_{car} = \begin{bmatrix} 1 & 0 & -\frac{p_1(t)}{p_3(t)} & 0 \\ 0 & 1 & -\frac{p_1(t)}{p_3(t)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{p_3(t)} & 1 \end{bmatrix} M_h$$

... where $\vec{p}(t)$ is the parametric equation of the center of observation dependent on time t such that $\vec{p}(t) = \vec{p}_1 + \frac{\vec{p}_2 - \vec{p}_1}{t_{max}}$ for $t \leq t_{max}$ and M_{car} is the resulting matrix.

To produce the shadow of the car in two-dimensions, we perform a projection onto the xz-plane ($y = 0$), representing the ground, and then perform a perspective projection on the result:

$$M_{shadow} = \begin{bmatrix} 1 & 0 & -\frac{p_1(t)}{p_3(t)} & 0 \\ 0 & 1 & -\frac{p_1(t)}{p_3(t)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{p_3(t)} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{p_{s_1}}{p_{s_2}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{p_{s_3}}{p_{s_2}} & 1 & 0 \\ 0 & \frac{1}{p_{s_2}} & 1 & 1 \end{bmatrix} M_h$$

We then normalize each column of matrices M_{car} and M_{shadow} by their fourth entries (refer to Equation 1). To map the $\mathbb{R}^4$ onto $\mathbb{R}^2$, we can simply delete the third and fourth row of the matrices or we can left-multiply by the standard matrix $\left[\begin{array}{c|cc} I_2 & \vec{0} & \vec{0} \end{array}\right]$ of a projection transformation. These vectors

are then connected by the information contained in the adjacency matrix K. In this case, straight lines are used to connect vectors (refer to Figure 3). However, other methods can be used to connect vectors, and would simply require additional mathematical data.

References

[1] Lay, D. C. (2003). *Linear Algebra*. Retrieved October 19, 2010, from Pearson Education: http://media.pearsoncmg.com/aw/aw_lay_linearalg_updated_cw_3/cs_apps/lay03_02_cs.pdf

[2] Lay, D. C. (2006). *Linear Algebra and Its Applications*. Pearson/Addison-Wesley.