Hermann Chong

Dr. King

Linear Algebra Applications

28 November 2001

The Importance of Matrices in the DirectX API

In the world of 3D gaming, there are two APIs (Application Program Interface) that reign supreme: OpenGL and DirectX. APIs help programmers write more efficiently by adding support in the programming language for frequently used calculations. Matrices are used a great deal in both APIs, but for the sake of brevity, this report will concentrate on the use of matrices in DirectX. DirectX uses matrices to express many objects, including 3d models, worlds, and perspectives. This report will discuss how the matrices are used, and how the API uses these calculations to make the programmer's code more efficient and powerful.

In Direct3D (a subset of the DirectX API), every object that will be sent to the monitor must be transformed through a series of matrix multiplications. Each object in DirectX is made up of triangles, lines, or points, which are called primitives. Each primitive is defined by a matrix of vertices (x, y and z coordinates). These vertices are then rotated, scaled, and translated to a two dimensional plane that can be displayed to the monitor. The matrix transformations needed to bring the vertices to the monitor is shown below in Figure 1, which explains the transformation pipeline.

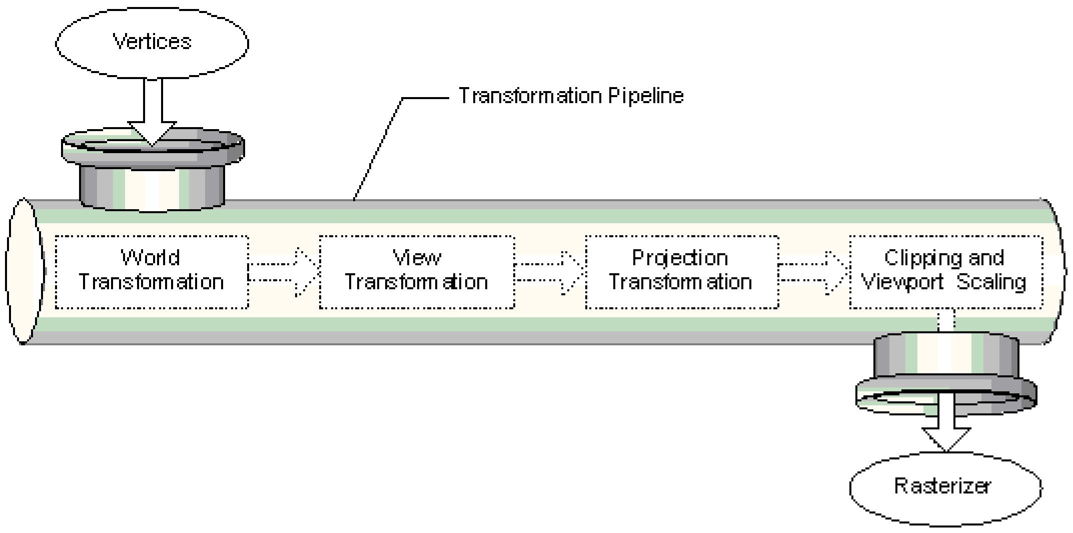

Figure 1. An example of the transformation pipeline.

World Transformation

The world transformation is a group of four matrix multiplications that move the different objects into the same ‘world’, effectively moving all the objects from their own local coordinate system to a common coordinate system. The four matrix multiplications are pitch, yaw, and roll, and a translation matrix.

In order to use matrices to handle translating for three dimensional objects, the standard 3x3 matrix is replaced with a 4x4 matrix, called a homogenous matrix, seen below in Figure 2.

$$\begin{matrix} A & \mathbf{p} \\ \mathbf{q}^T & r \end{matrix}$$

Figure 2. The structure of a homogenous matrix.

In Figure 2 above, *A* is a 3x3 matrix that is used for rotation and scaling, p is a $\sum^3$ vector used to translate points, q is a $\sum^3$ vector used to change the perspective, and r is a scalar.

By multiplying this homogenous matrix with a (*x*, *y*, *z*, 1) vector, the vector can be repositioned anywhere. These transformations are done through the API by calling functions that create the rotation and translation matrixes, multiplying together, and then multiplying the final transformation matrix to the object. At the end, all the objects are converted from their own local coordinates to a global coordinate system.

View Transformation

The view transformation requires the same types of computations as the world transformation. This transformation changes the previous coordinate system to one where the viewpoint is located at the origin. The math for this transformation is excluded because the math behind this transformation is roughly the same as for the world transformation.

Projection Transformation

The projection transformation uses the result of the world and view transformations to translate the three dimensional objects into a cuboid shape. This is done by manipulating q in Figure 2. The transformation scales and translates the matrix, acting as the ‘lens for the camera.’ As a result, objects in between the camera and the viewing plane are magnified, and the objects behind the viewing window are shrunk. This causes an illusion of perspective.

Clipping and Viewport Scaling

After the transformations have been completed, Direct3D uses two ‘clipping planes’ to determine what is seen in the window. An illustration of this is shown below in Figure 3.

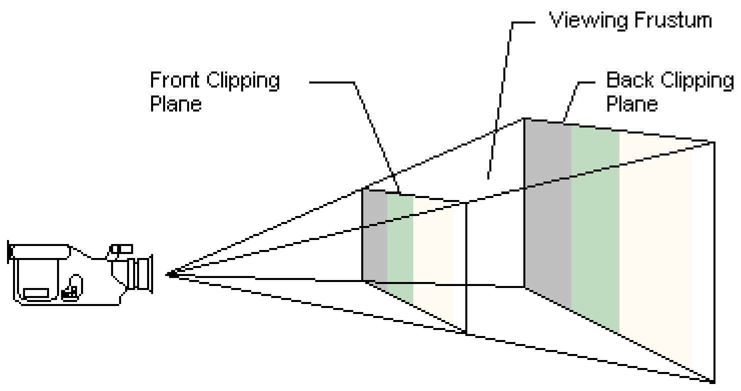

Figure 3. The clipping planes.

The volume in between the two clipping planes contains the objects that will be shown on the computer monitor. Also needed to determine what is shown on the screen is the field of view, the angle between the horizontal and the top of the line that defines the edge of the clipping planes. The API clips by filtering through the triangles in the screen, determining if they are within the volume. If the triangles are not in the volume, the API does not render the triangle to the screen. The viewport is the two dimensional plane used to display all the objects on the screen. Viewport scaling simply moves the three dimensional objects onto a two dimensional plane, determined by the screen resolution of the computer monitor. Together, clipping and viewport scaling translates the world onto the screen.

Works Cited

DirectX 8.1 SDK Documentation, Microsoft Corporation

Section 2.8, Linear Algebra and its Applications, David C. Lay