Graphics Pipeline

Justin Huffman, Eryn Kelsey-Adkins, Jesus Torres

November 28, 2018

Abstract

The rendering pipeline is theseries of steps a graphics system needs to follow to render a 3D scene to a 2D screen. These steps start with the Application, then the Geometry, the Rasterization, and finally the Display. This presentation provides a brief overview of each of the steps and the matrices used to transform 3D objects into 2D displays.

1 Introduction

Here is the text of your introduction.

2 Application: how 3D models are stored

2.1 Face-Vertex Mesh

2.1.1 Face List

A list of faces (triangles), with each face being defined by the three vertices that make it up.

2.1.2 Vertex List

List of vertices (points in 3D space), with each vertex being defined by its x, y, and z coordinates, as well as a list of faces that it belongs to. This is redundant, but saves having to search the face list.

2.2 Homogeneous Coordinates

Imagine a 2D projector. The X and Y values are easy to see, but what happens to the image as you move the projector closer or further away from the screen? The image is scaled up or down. This is the W value in homogeneous coordinates.[1] All vertices and vectors are in homogenous (x, y, z, w) form. At this time, $W = 1$ for vertices (points in space) and $W = 0$ for vectors (directions and

[1] https://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/

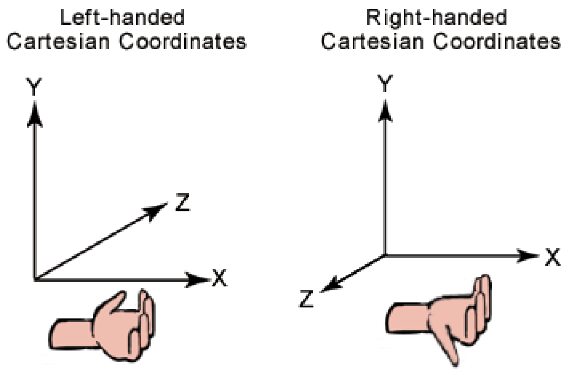

Figure 1: Left- and Right- Handed Cartesian Coordinates

facings). Other values of W are used in later steps for projection. The use of homogeneous vectors allows for standard 4x4 matrices to be used to transform both vertices and vectors.

2.3 Model/World Transformation

2.3.1 World Coordinate System

The world coordinates are the bedrock of the 3D model. This universal coordinate system is independent of the camera and objects or primatives, which each have their own coordinate system, are placed within the world coordinates. Every point is defined by three vertices (x, y, z). The cartesian coordinates can be left- or right- handed, defined by whether the z vertex points into or out of the display./footnote[1]cs.uic.edu Depending on the coordinate system being used, the models may need to be inverted when loading for veiwing.

2.3.2 Object Coordinate System

As mentioned above, each object has its own set of coordinates. Objects are most often represented by polyhedren, further broken down into triangles. Each triangle is defined, counterintuitively, by 4 vertices: the 3 edges, and a fourth vertex defining the front of the primative. When objects are moved in relation to the world coordinates, the origin of the object (in the object's coordinate system) is moved, with all other points transformed and equal amount.

O bjects can further have a hierarchal coordinate system. The University of Illinois uses the example of a hand being positioned relative to the arm, rather than to the world coordinate system.

2.4 Camera Transformation

The camera transformation can be thought of as one more coordinate system: based upon the viewpoint of the observer, changing as the point of view changes. This coordinate system is defined by:

- A scene of the world coordinates, including all primatives within those coordinates
- The camera position - $C = (x_c, y_c, z_c)$
- The point of the camera's focus - $A = (x_a, y_a, z_a)$
- The field of view angle - α
- The definition of the near and far planes, perpindicular to A with a distance of n and f from the camera

A pyramid is formed with C, A, and α which is then truncated by n and f into a frustum.

3 Geometry

3.1 Projection

Projection transforms the view volume into a unit cube with extreme points at (1, 1, 1) and (-1, -1, -1). This step is called projection even though it is a volume to volume transformation because the z-coordinate is saved for the rasterization step.

3.1.1 Types of Projection

3.1.2 Orthographic Projection

Orthographic projection is mostly used in engineering for multi-view drawings. It is to scale, and all parallel line remain parallel with no forced perspective. An orthographic projection transformation can be represented as

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Each homogeneous vector $v = (v_x, v_y, v_z, 1)$, the transformed vector would be

$$P_v = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ 0 \\ 1 \end{pmatrix}$$

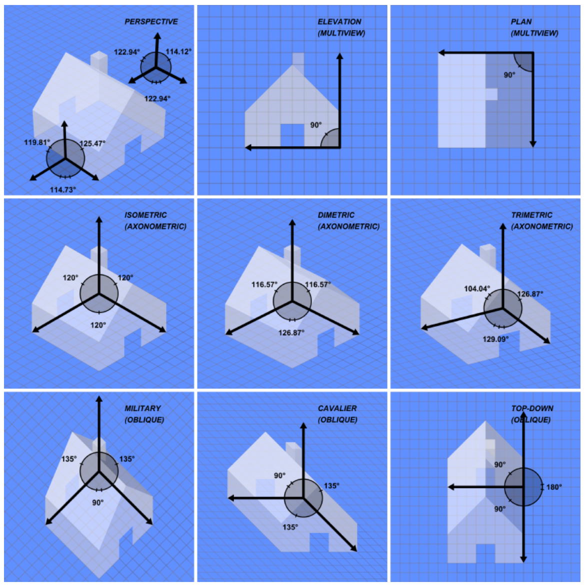

Figure 2: A comparison of projection types

3.1.3 Weak Perspective Projection

Weak perspective projection adds a scaling factor to orthographic projection. This scaling factor uses averages to apporximate perspective and is used with simple models and small fields of view.

$$P_x = \frac{X}{Z_{ave}}$$

$$P_y = \frac{Y}{Z_{ave}}$$

assuming focal length $f = 1$.

3.1.4 Perspective Projection

Perspective projection most closely replicates the human binocular vision where distant objects appear smaller. To describe the transformation, we must start with four variables:

- $a_{x,y,z}$ - the 3D position of a point A that is to be projected
- $c_{x,y,z}$ - the 3D position of a point C representing the camera
- $\theta_{x,y,z}$ - the orientation of the camera

- $e_{x,y,z}$ - the display surface's positon relative to the camera pinhole. (Although negative z values are more correct, the create an inverted image both horizontally and vertically, so most conventions use positive values.

To compute the 2D projection ($b_{x,y}$) of the point A ($a_{x,y,z}$) that is to be projected, a vector $d_{x,y,z}$ is first defined as the position of point A with respect to the camera transformation, with origin in C and rotated by θ:

$$\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{pmatrix} \begin{pmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{pmatrix} \begin{pmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \right)$$

3.2 Clipping

Clipping saves time and energy by specifying only those objects within the view volume for rendering. The vidual unit cube has been transformed into a truncated pyramid (frustrum) and objects outside of the frustrum are discarded by a method appropriately named frustrum culling. Objects that will be obscured by other primatives are also culled through backface culling. Objects inside the frustum are saved for rasterization.

O bjects that intersect the edges of the frustrum must be clipped. Line clipping simply modifies the endpoints of lines to lie within the appropriate rectangle. Polygon clipping converts a polygon that intersects the frustum into one or more polygons the form the intersection of the original with the clip window. The Sutherland-Hodgman Polygon Clipping Algorithm is commonly used[2]:

3.3 Window-Viewport Transformation

Translates camera coordinated into screen coordinated in order to match the size and location of the window

3.4 Rasterization

Converts primitives (shapes) to discrete coordinates (pixels)

3.4.1 Bresenham's line algorithm

Draws line between two points in pixels

3.4.2 Z-buffering

Solves visibility problem

[2]https://www.geeksforgeeks.org/polygon-clipping-sutherland-hodgman-algorithm-please-change-bmp-images-jpeg-png/

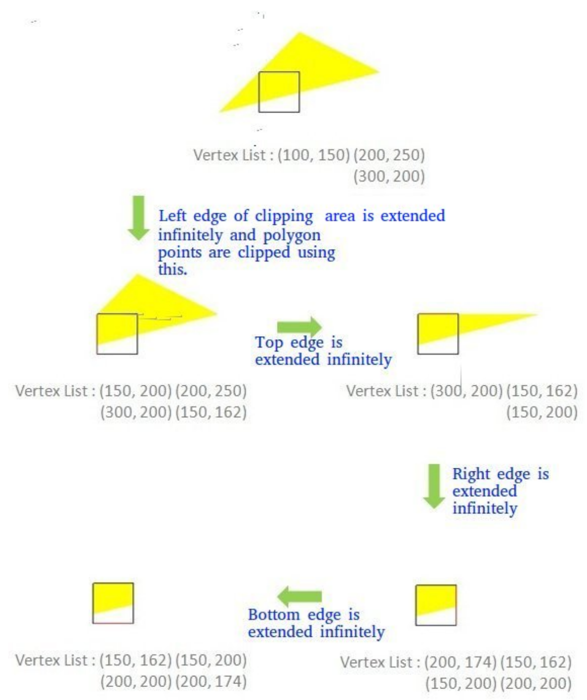
Vertex List : (100, 150) (200, 250) (300, 200)
Left edge of clipping area is extended infinitely and polygon points are clipped using this.
Vertex List : (150, 200) (200, 250) (300, 200) (150, 162)
Top edge is extended infinitely
Vertex List : (300, 200) (150, 162) (150, 200)
Right edge is extended infinitely
Vertex List : (200, 174) (150, 162) (150, 200) (200, 200)
Bottom edge is extended infinitely
Vertex List : (150, 162) (150, 200) (200, 200) (200, 174)

Figure 3: An example of the Sutherland-Hodgman Algorithm in use

4 Conclusion

Write your conclusion here.