OpenGL Pipeline

1 The Problem

A simplified OpenGL (sgl) engine that will take care of the Modelling, Viewing, and Projection transformations.

2 The sgl Library

The system has 4 different matrices to hold the current modelling, viewing, projection, and viewport transformations. To make matters easy, we will have different commands for each. The Simplified OpenGL library has the following functions:

• Modelling Functions: The modelling functions modify the modelling matrix mentioned before. Each of the functions right multiplies the current Modelling Matrix with a matrix corresponding to the transform. (Except, obviously, the last three.) The Modelling Matrix transforms ORC to WC.

1. sglModRotate(angle, axisx, axisy, axisz)

2. sglModTranslate(tx, ty, tz)

3. sglModScale(sx, sy, sz)

4. sglModMatrix(float mat[16])

5. sglModLoadIdentity()

6. sglModPushMatrix()

7. sglModPopMatrix()

• Viewing Functions: The viewing commands affect the camera position/parameters. The first 3 commands right multiply the current Viewing Matrix by a matrix corresponding to the command. The last one loads identity matrix. This corresponds to the VRC and WC being aligned. As with OpenGL, the camera is at the origin of VRC and looks into -Z direction. The Viewing Matrix transforms from WC to VRC.

1. sglViewRotate(angle, axisx, axisy, axisz)

2. sglViewTranslate(tx, ty, tz)

3. sglLookAt(cx, cy, cz, lx, ly, lz, upx, upy, upz)

4. sglViewMatrix(float mat[16])

5. sglViewLoadIdentity()

• Projection Functions: Projection commands define the camera: its projection model and parameters. These commands assign the Projection Matrix to that corresponding to the command. (i.e., no right multiplication!). The Ortho and Frustum commands can define asymmetric view volumes. The Projection Matrix transforms from VRC to NC. At the end, the normalized screen coordinates in the range [-1 .. 1] are obtained by dividing the first two components by the fourth one.

1. sglProjOrtho(left, right, bottom, top, near, far)

2. sglProjFrustum(left, right, bottiom, top, near, far)

• Viewport Commands: Viewport commands map the normalized view window to the actual one in use.

1. sglViewport(llx, lly, width, height)

The Viewport Matrix is defined by this command. (i.e, no right multiplication.)

2.1 Supported Primitives

We will use only the triangle primitive. The following functions are to be supported by the system with the usual meanings.

1. sglBegin(SGL TRIANGLES) // Draw a triangle for every

triplet of vertices

2. sglEnd()

3. sglColour(r, g, b)

4. sglClear(r, g, b)

5. sglVertex(x, y, z)

6. sglShow() // Makes the scene appear; write to the PLY file

7. sglClear //clears the framebuffer to the given colour.

8. sglColour defines the current colour.

OpenGL calls are used to implement the above drawing functions. The modeling, view and projection transformations are implemented using my own C++ code.

2.2 The Task

When the system starts up, all matrices are initialised to identity matrices. The above commands thereafter modify the matrices.

Geometry is really defined only by the sglVertex() commands. When the required number of vertices for a primitive is available, they are transformed from ORC to the screen co-ordinates. The primitives are formed in canonical coordinate system. Perspective division is performed after projection to get (x, y, z) co-ordinates in canonical frame. Viewport transformation is performed to get the screen coordinates. Triangles are generated in screen coordinates and added to a data structure with x, y, z, colour, etc.

The scene is built up in the memory array corresponding to the frame buffer. When a sglShow() call is received, the data structure is rendered using OpenGL as described below. SDL/GLUT commands are used to open and setup the window etc.

For rendering, OpenGL and a 3D orthographic coordinate system is used. glOrtho(scrL, scrR, scrB, scrT, -1, 1) where scrL, scrR, scrB, scrT are the left, right, bottom, top planes in screen coords is used. The z-values should provide proper visibility. Thus, clipping, rasterization, etc. is being done by the orthographic projection of OpenGL.