Nanyang Technological University
School of Computer Science and Engineering

Laboratory Report

CZ2003
Computer Graphics and Visualization

Lab 1
Visualisation Using Polygons

By
Teo Wei Jie (U1822263C), SS2

18 April 2020

Contents

1 Experiment on diffuseColor

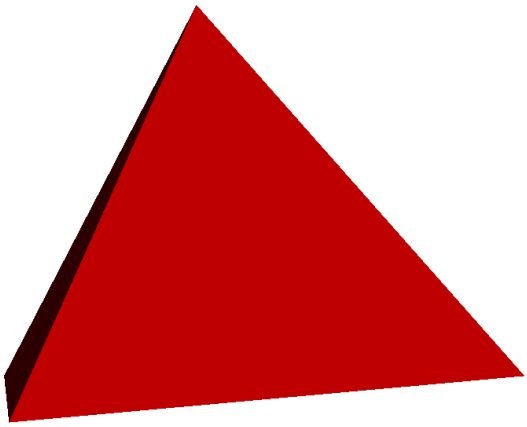

Figure 1 – Pyramid with diffuseColor = 1, 0, 0

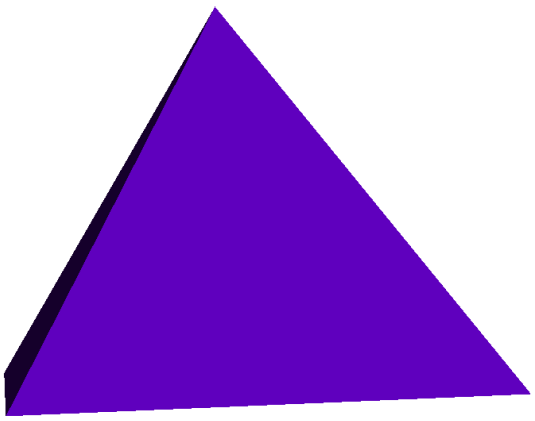

Figure 2 – Pyramid with diffuseColor = 0.498, 0, 1

Figure 3 – Color of RGB values 255, 0, 0

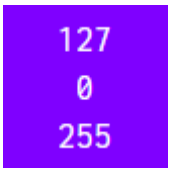

Figure 4 –Color of RGB values 127, 0, 255

Figure 1 is defined in "01_pyramid_red.wrl" and Figure 2 is defined in "02_pyramid_purple.wrl".

The diffuseColor field determines the colour of the shape in RGB with values between 0 and 1 by reflecting light from the surfaces with respect to the light source.

The values may be derived from normal RGB values of 0 to 255 by dividing the number by 255. For example, a 'R' value of 127 in normal RGB becomes 0.498 in the diffuseColor field by taking 127 / 255.

As such, the pyramids illustrated in Figures 1 and 2 corresponds to the RGB values of 255, 0, 0 and 127, 0, 255, respectively, as illustrated in Figures 3 and 4. The pyramids are duller as they are rotated slightly to the right and hence there is a slight shadow casted on the surfaces.

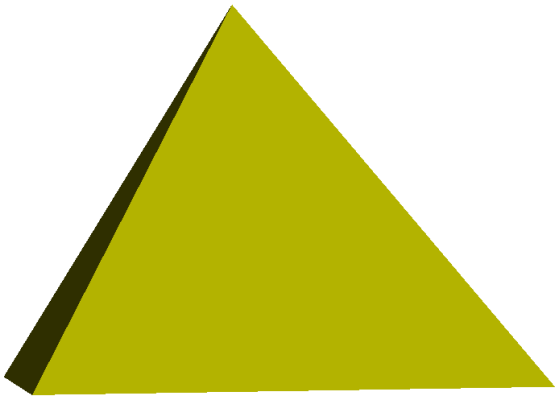

Figure 3 – Pyramid with diffuseColor = 1, 1, 0

Figure 4 – Pyramid with diffuseColor = 5, 5, 0

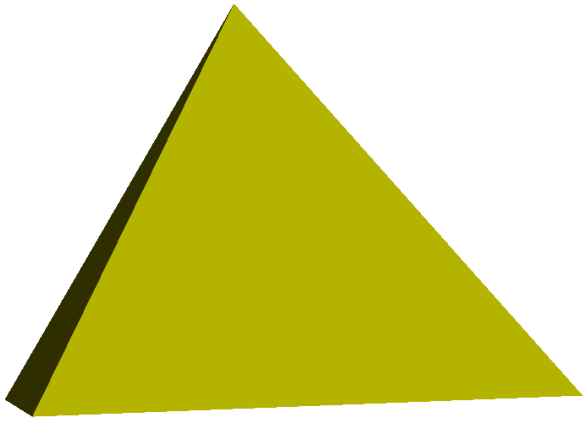

Figure 5 – Pyramid with diffuseColor = 1, 1, -5

Figure 3 is defined in "03_pyramid_yellow.wrl", Figure 4 is defined in "04_pyramid_yellow_2.wrl" and Figure 5 is defined in "05_pyramid_yellow_3.wrl".

The pyramid illustrated in Figure 4 is significantly brighter than that in Figure 3, but it also loses lighting features such as the shadow casted on the left face of the pyramid when the pyramid is rotated slightly to the right. My hypothesis is that the diffused colour is so overwhelming that the shadow casted on the pyramid becomes irrelevant, hence the pyramid in Figure 4.

Despite having a 'B' value of -5, there is no discernible difference between the two pyramids illustrated in Figures 5 and 6. My hypothesis is that a negative field does not overwhelm or affect the shadow casted on the pyramid.

2 Hexagon Using VRML

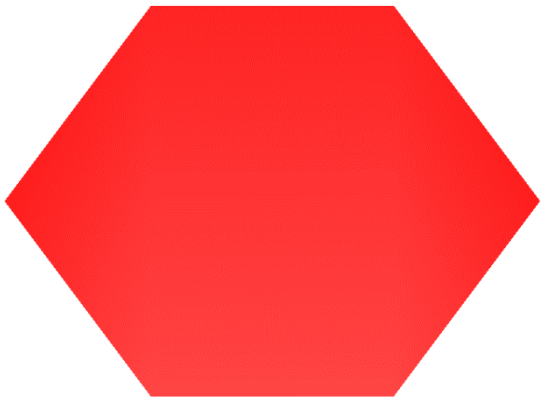

Figure 6 – Snapshot of hexagon using VRML

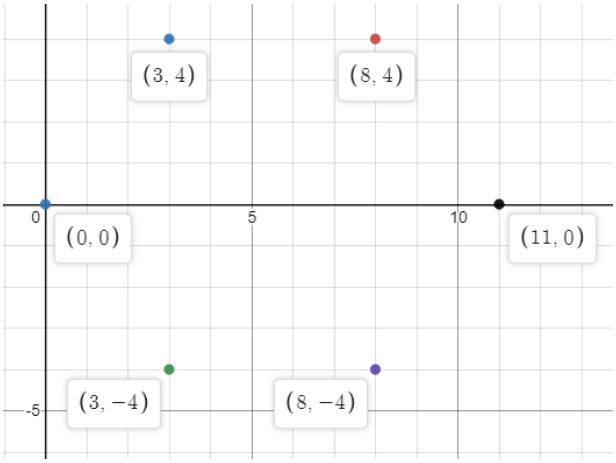

Figure 7 – Hexagon on graph paper

Figure 6 is defined in "06_hexagon.wrl" and illustrates a hexagon. To obtain the points for the vertices of the hexagon, the hexagon is first plotted on a graph as illustrated in Figure 8, using Pythagoras Theorem to determine the points with a length of 10 for each side, starting from the origin.

3 3D Cube Using VRML

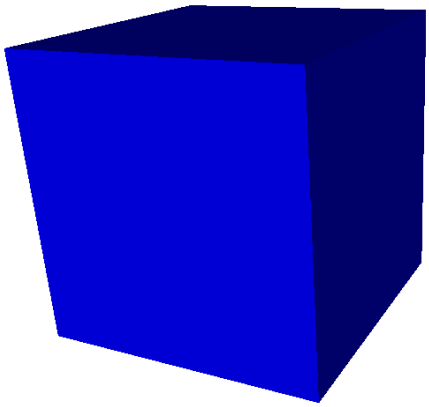

Figure 8 – Snapshot of a 3D Cube using VRML

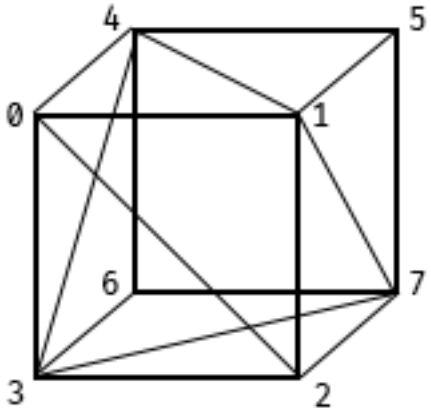

Figure 9 – Vertices of the 3D Cube

Figure 8 is defined in "08_3d_cube.wrl" and illustrates a 3D cube while Figure 9 illustrates the vertices used to form the cube. The cube is formed using triangles, 2 on each side, as I was being dumb and stubborn and wanted to try forming shapes using only triangles as most 3D models in games, etc., are formed using only triangles.

4 Experiment On Vertices Order

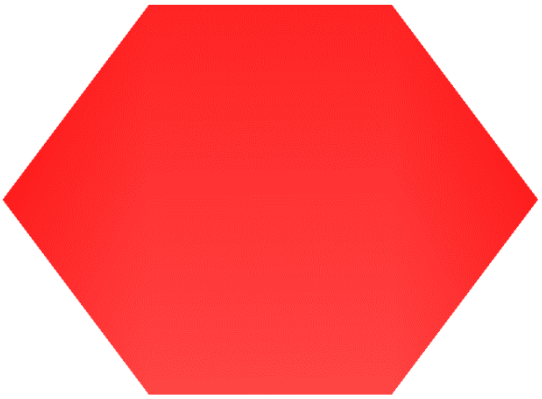

Figure 10 – Snapshot of front view of hexagon

Figure 11 – Snapshot of back view of hexagon

Figure 10 is defined in "10_hexagon_single.wrl" and illustrates a hexagon with only one visible surface. The hexagon is defined with vertices connected in an anti-clockwise manner. With reference to Figure 8, the hexagon is formed by connecting vertices in the following order:

- (0, 0), (3, -4), (8, 4), -1,
- (3, -4), (8, 4), (3, 4), -1,
- (3, -4), (8, -4), (8, 4), -1,
- (8, -4), (11, 0), (8, 4), -1

When the hexagon is rotated 180 degrees, the hexagon "disappears", as illustrated in Figure 12 (I do not know how else to illustrate it in the figure…). VRML renders the surface with vertices connected in anti-clockwise manner visible and not the opposite side of the surface. As such, the back view of the hexagon is invisible.

The difference between "10_hexagon_single.wrl" and "06_hexagon.wrl" illustrated in Figure 6 is that the latter's hexagon is visible on both sides. This is done by including the declaration "solid FALSE", which renders the surface on both sides instead of just the anti-clockwise side.