

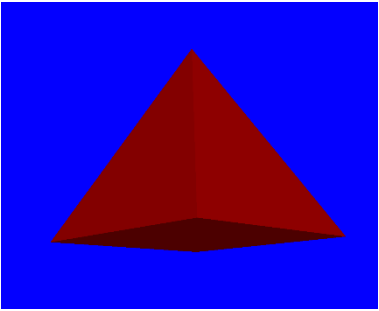
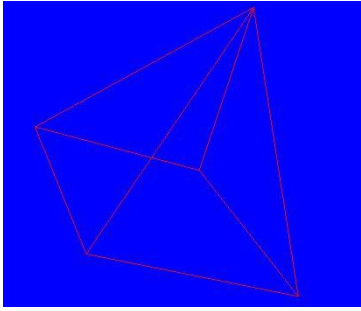
**NANYANG
TECHNOLOGICAL
UNIVERSITY**

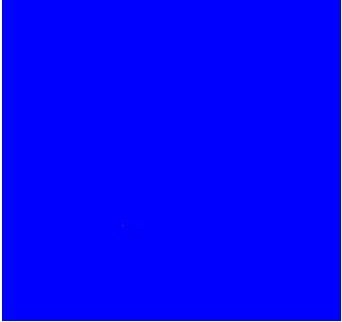
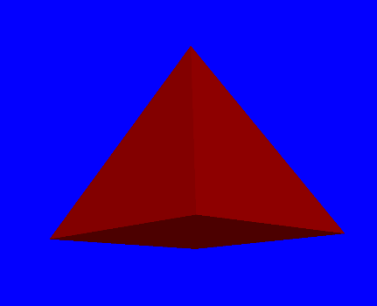
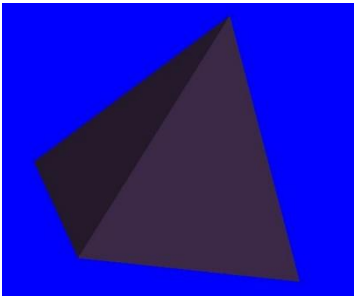
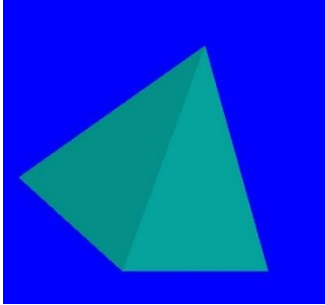
SINGAPORE

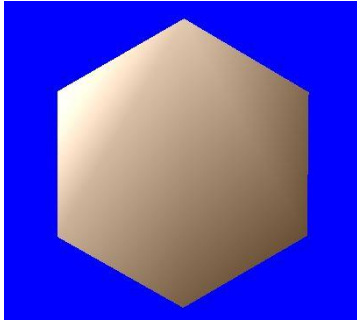
**CZ2003 COMPUTER GRAPHICS & VISUALIZATION
EXPERIMENT 1: VISUALIZATION USING POLYGONS
LAB REPORT**

Shearman Chua Wei Jie (U1820058D)

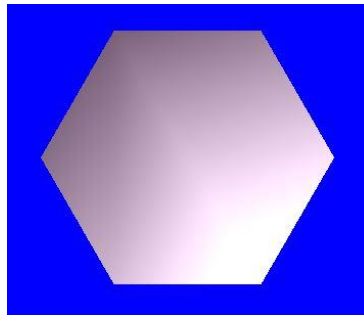
LAB GROUP: SS2

Polygon 1	Polygon 2	Note
 <p>Above is the snapshot of “polygons.wrl” which define a polygon using 5 vertices in the form (x,y,z): (-1.0 -1.0 1.0),(1.0 -1.0 1.0),(1.0 -1.0 -1.0),(-1.0 -1.0 -1.0),(0.0 1.0 0.0). and the three-dimensional pyramid is generated by defining 5 surfaces using the order of the given vertices in coordIndex [</p> <pre> #bottom square 0, 3, 2, 1, -1, #side1 0, 1, 4, -1, #side2 1, 2, 4, -1, #side3 2, 3, 4, -1, #side4 3, 0, 4, -1,]. </pre>	 <p>This is the snapshot of the same pyramid in “polygons.wrl” but in “Wireframe” mode which represents the edges that form the pyramid.</p>	<p>Note 1: To generate a solid, filled 3D polygon, the Graphics Mode must not be set to Wireframe. However, when we want to remedy and detect errors when defining our coordIndex, we can use Wireframe to identify errors easily by looking at how the vertices are connected.</p>

 <p>Above is the snapshot of polygons.wrl with Graphics Mode set to Flat. Sets BS Contact into vertices rendering mode. This shows only dots for each corner of a face.</p>	 <p>Above is the snapshot of polygons.wrl with Graphics Mode set to Flat. Sets BS Contact into flat rendering mode. This renders a scene as one would expect but disables gouraud shading.</p>	<p>Note 2:</p> <p>If we have many vertices, we can use the “Vertices” Graphic mode to render the polygon to isolate and visualize the vertices.</p>
 <p>Above is the snapshot of “pyramid-colour changed.wrl” which defines a polygon with diffuseColor 0.30, 0.2, 0.35 (#red=0.3, green=0.2, blue=0.35)</p>	 <p>Above is the snapshot of “pyramid-colour changed2.wrl” which define a polygon with diffuseColor 0.03 0.91 0.87 (#red=0.03, green=0.91, blue=0.87)</p>	<p>Note 2:</p> <p>The rendered color is composed of three primary colors in computer graphics and visualization which are Red, Green and Blue. First parameter in diffuseColor represents the Red color ratio, the second represent Green and the third for Blue. Thus, to render a polygon of a particular color we want, we can check the RGB ratio online and define the ratio of each color to obtain the rendering color desired.</p>



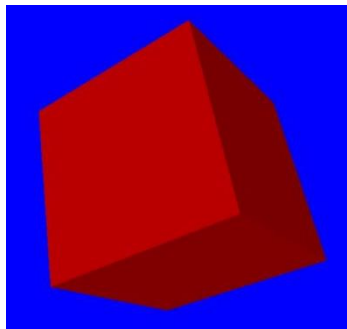
Above is the snapshot of “2D hexagon.wrl” which define a six-sided equilateral and equiangular polygon with $[0.0, 1.0, 0.0]$ as vertex 0, $[-0.8661, 0.5, 0.0]$ as vertex 1, $[-0.8661, -0.5, 0.0]$ as vertex 2, $[0.0, -1.0, 0.0]$ as vertex 3, $[0.8661, -0.5, 0.0]$ as vertex 4, $[0.8661, 0.5, 0.0]$ as vertex 5 And place order in coordIndex order 0, 1, 2, 3, 4, 5, -1, with -1 defining end of surface.



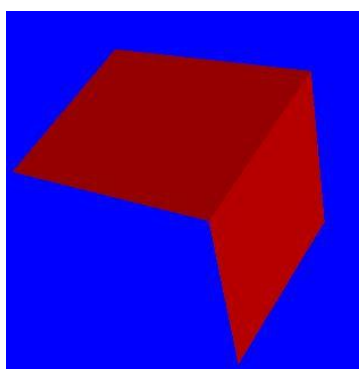
Above is the snapshot of “polygon hexagon2.wrl” which define a six-sided equilateral and equiangular polygon with $[-0.5, 0.866, 0.0]$ as vertex 0, $[-1.0, 0.0, 0.0]$ as vertex 1, $[-0.5, -0.866, 0.0]$ as vertex 2, $[0.5, -0.866, 0.0]$ as vertex 3, $[1.0, 0.0, 0.0]$ as vertex 4, $[0.5, 0.866, 0.0]$ as vertex 5 And place order in coordIndex order 0, 1, 2, 3, 4, 5, -1, with -1 defining end of surface. With a different set of coordinates from “2D hexagon.wrl”, we obtain a hexagon with the same dimensions but a different orientation.

Note 3:

The different order of vertices in coordIndex changes the direction of the visible side of the 2D surface. The VRML program uses the “right-hand” rule where if you curl the fingers of your right hand according to the order of the vertices you have defined, your thumb will indicate the direction of the visible side of the surface, or rather, the normal of the surface. To enable the surface to be seen on the computer screen, the vertices must be ordered in the anti-clockwise manner to have the normal of the surface facing out of the computer screen.



Above is the snapshot of “3D cube.wrl” which shows a 3D cube rendered based on 8 defined vertices. To ensure that the rendered surfaces are visible, all the surfaces have their vertices defined in an anti-clockwise direction which ensures that the surface direction



Above is the snapshot of “3D cube-2.wrl” which defines a 3D cube with the same 8 vertices defined in “3D cube.wrl”. However, the order of bottom surface is now “0, 1, 2, 3, -1,” instead of “0, 3, 2, 1, -1,” as in “3D cube.wrl”. Which causes the

Note 4:

The importance of the ordering the vertices in coordIndex in an anti-clockwise manner is highlighted here, otherwise, the 3D cube will not be rendered properly. In order to ensure that the polygon is rendered properly, one should always make use of the right-hand rule to check the direction of the surface. Otherwise, the surface will be not visible to the user.

points out of the computer screen. If the vertices are in a clockwise direction instead, the surfaces will face inwards into the screen of the computer.	bottom surface to be rendered to face inside the cube instead of facing outwards as the bottom surface's vertices are now declared in a clockwise manner. The order of vertices in coordIndex defines the direction where the surface is rendered.	
--	--	--