

**NANYANG**  
**TECHNOLOGICAL**  
**UNIVERSITY**

CZ2003 Computer Graphics and  
Visualization

Lab 1 Report: Visualization using polygons

ACHARYA ATUL

U1923502C

SSP1

## Explore different Graphics Modes of the VRML browser

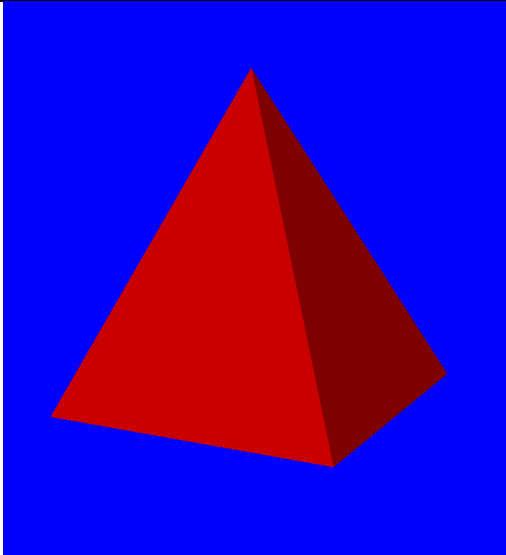
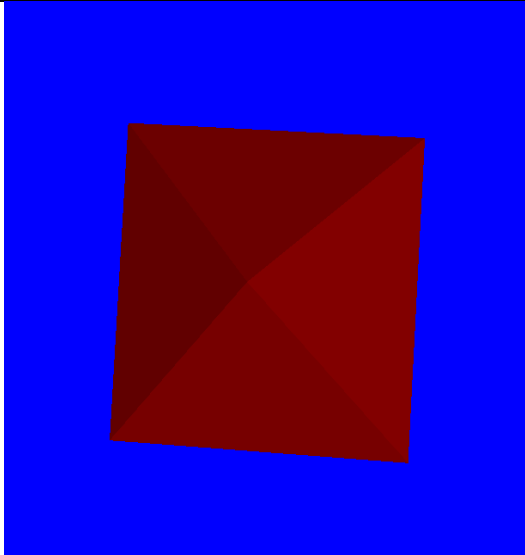
The rendering of a simple polygon mesh (pyramid) is displayed below in different graphic modes of the VRML browser. The file “**Pyramid.wrl**” contains the definition of the simple polygon mesh. The polygon is defined by the following vertices :

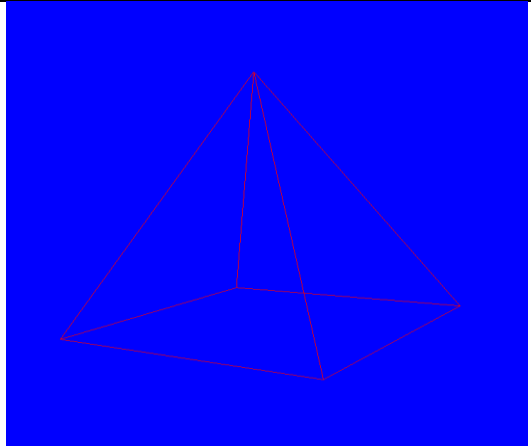
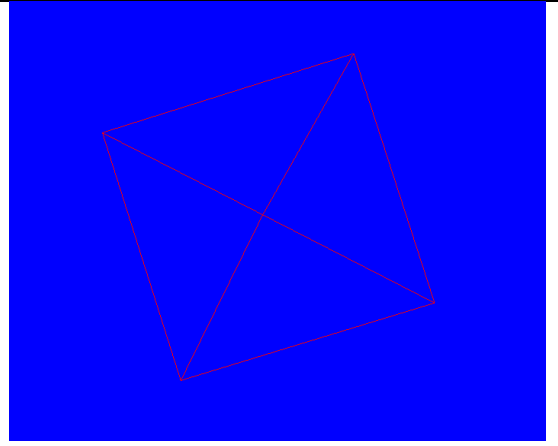
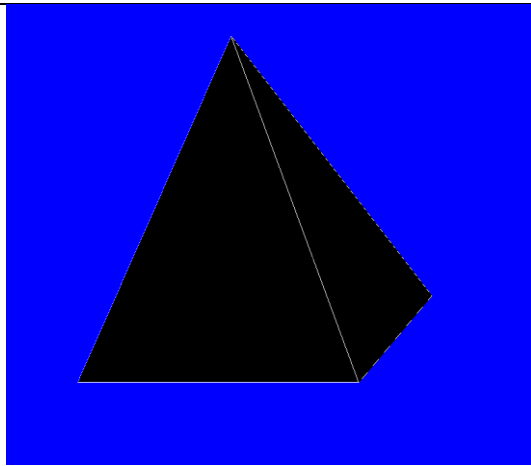
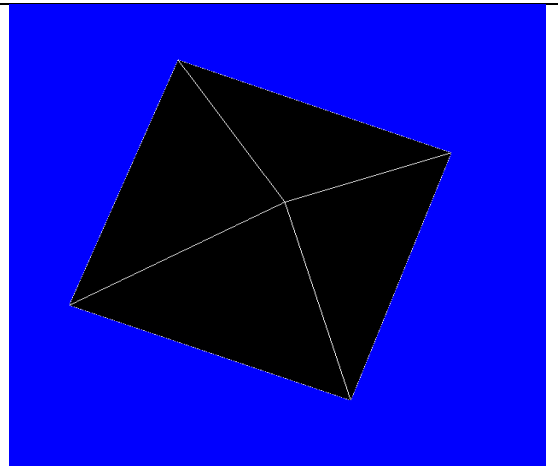
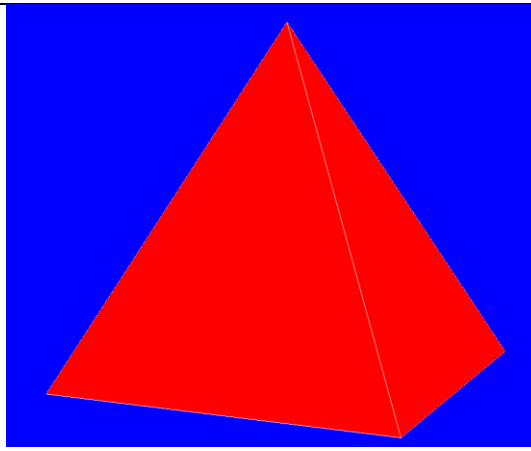
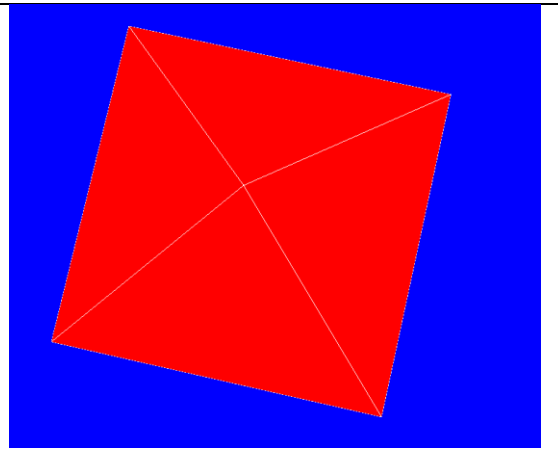
Bottom vertices:

- I.        -1.0 , -1.0, 1.0,    #vertex 0
- II.       1.0, -1.0, 1.0,    #vertex 1
- III.      1.0, -1.0, -1.0,   #vertex 2
- IV.      -1.0, -1.0, -1.0,   #vertex 3

Top vertex:

- V.       0.0   1.0   0.0    #vertex 4

MODE	FRONT VIEW	TOP VIEW
Simple Polygon Mesh		

Wireframe		
Hidden Line		
Solid		

## Examine how the color of the shape defined in diffuseColor field can be changed.

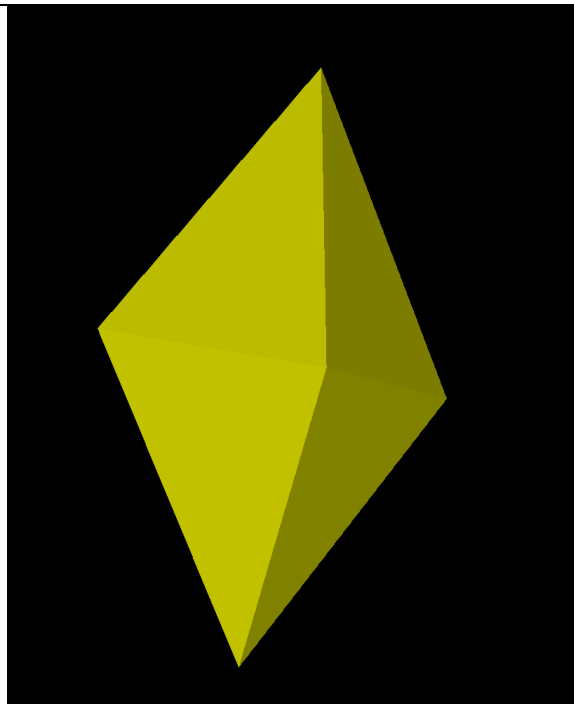
VRML uses RGB colour model (red + green + blue = white). Diffusecolor has 3 parameters whose values can take any real number between 0 and 1.

Parameter 1 defines the quantity of red.

Parameter 2 defines the quantity of green.

Parameter 3 defines the quantity of blue.

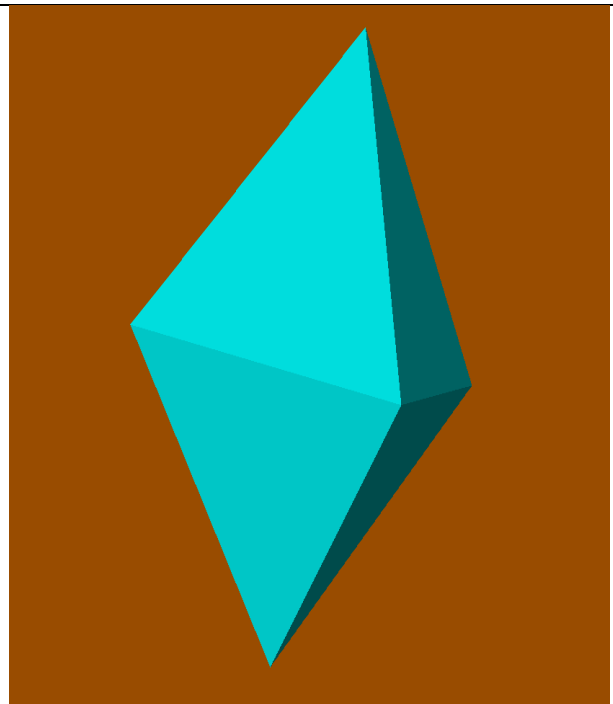
Below is an example of how the values in the diffusecolor field affect the colour of the shape that we see.



Above is a snapshot of  
**"Double\_Pyramid\_yellow.wrl"**

Setting the values of red=1, green=1, blue =0,  
We get yellow.

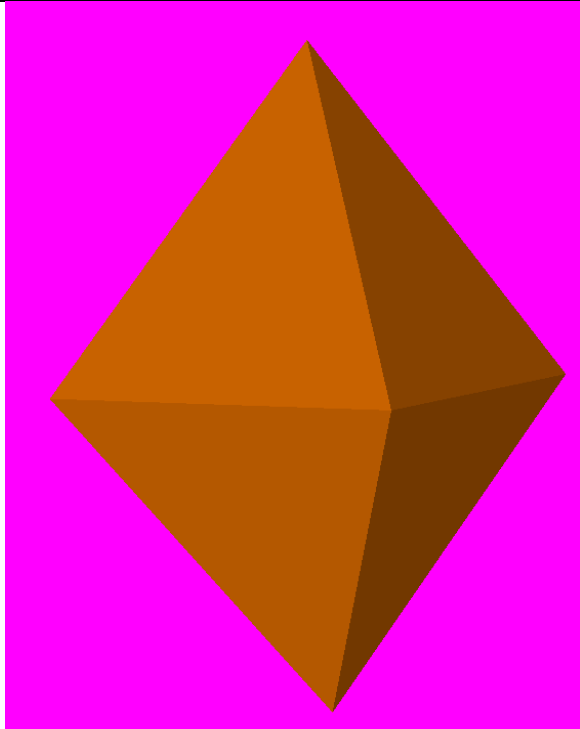
Setting the values of red=0, green=0, blue=0,  
We get black.



Above is a snapshot of  
**"Double\_Pyramid\_cyan.wrl"**

Setting the values of red=0, green=1, blue =1,  
We get cyan.

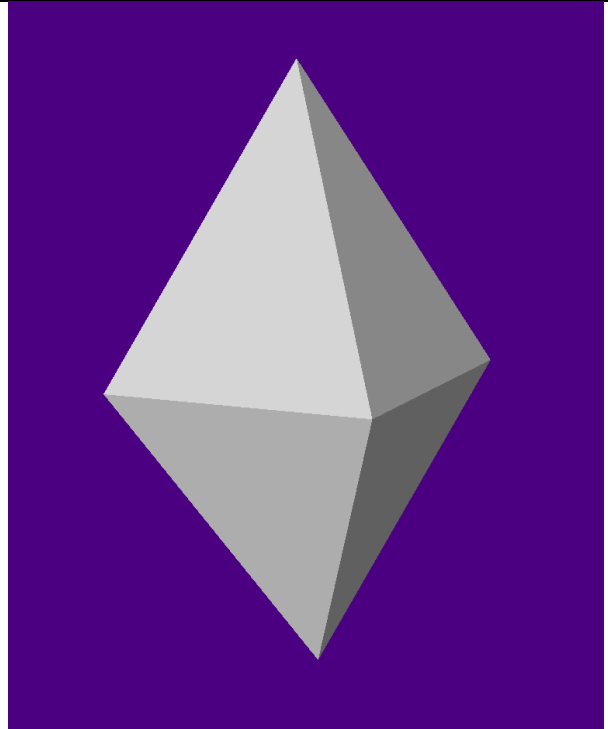
Setting the values of red=0.6, green=0.3, blue=0,  
We get brown.



Above is a snapshot of  
**"Double\_Pyramid\_orange.wrl"**

Setting the values of red=1, green=0.49, blue=0,  
We get orange.

Setting the values of red=1, green=0, blue=1,  
We get pink.




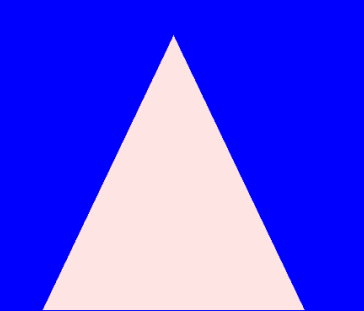
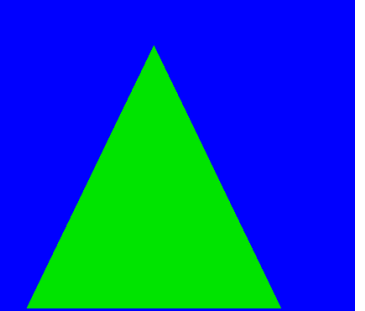
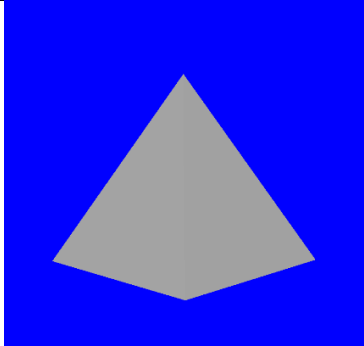
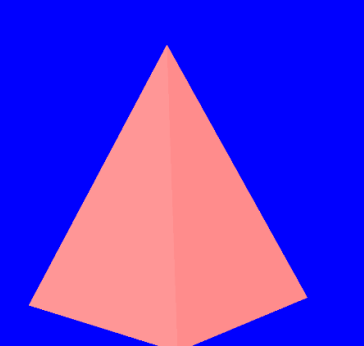
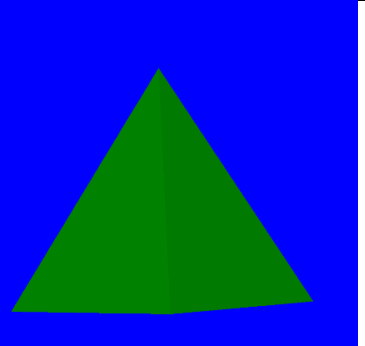
Above is a snapshot of  
**"Double\_Pyramid\_white.wrl"**

Setting the values of red=1, green=1, blue=1,  
We get white.

Setting the values of red=0.29, green=0, blue=0.5,  
We get indigo.

When colour values are less than 0 or greater than 1, VRML treats it as if it were 0 or 1 respectively.

Below is an example when colour values are -1 and 0.

		
<p>Above is a snapshot of <b>"Pyramid 2.1.wrl"</b> It has colour definition of <math>r=1, g=1, b=1</math>. (This is used for comparison)</p>	<p>Above is a snapshot of <b>"Pyramid 2.2.wrl"</b> It has colour definition of <math>r=2, g=1, b=1</math>. When viewed along one of its face, it appears white and increasing the value beyond 1 doesn't make a difference.</p>	<p>Above is a snapshot of <b>"Pyramid 2.3.wrl"</b> It has colour definition of <math>r=-2, g=1, b=0</math>. When viewed along one of its face, it appears green and decreasing the value below 0 doesn't make a difference.</p>
		
<p>Above is a snapshot of <b>"Pyramid 2.1.wrl"</b> It has colour definition of <math>r=1, g=1, b=1</math>. (This is used for comparison)</p>	<p>Above is a snapshot of <b>"Pyramid 2.2.wrl"</b> It has colour definition of <math>r=2, g=1, b=1</math>. When viewed along one of its edges there appears to be hint of red and it tells us increasing the colour value beyond 1 does have an effect.</p>	<p>Above is a snapshot of <b>"Pyramid 2.3.wrl"</b> It has colour definition of <math>r=-2, g=1, b=0</math>. When viewed along one of its edges there seems to be no difference and it appears that decreasing the colour value below 0 doesn't seem to have any effect.</p>

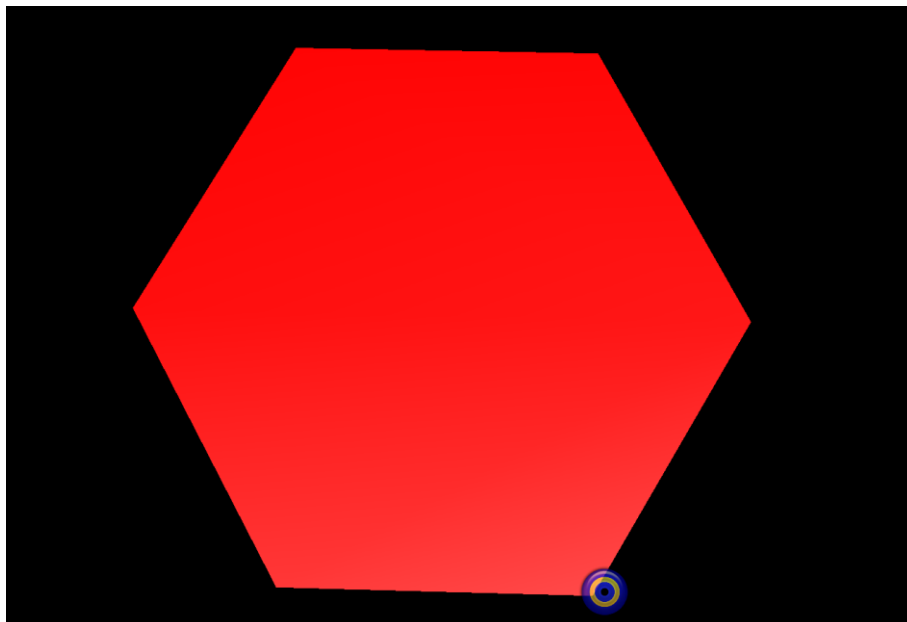
# MAKING A 2D REGULAR HEXAGON

The rendering of a 2D regular hexagon defined in the file “**regular hexagon.wrl**” is displayed below .  
The Hexagon is defined by the following vertices :

- I. 4,0,0, #vertex 0
- II. 2,3.5,0, #vertex 1
- III. -2,3.5,0, #vertex 2
- IV. -4,0,0, #vertex 3
- V. -2,-3.5,0, #vertex 4
- VI. 2,-3.5,0, #vertex 5

The front face of the hexagon is given by the following ordering of the vertices: 0, 1, 2, 3, 4, 5, -1.

The rear face of the hexagon is given by the following ordering of the vertices: 5, 4, 3, 2, 1, 0, -1.



## MAKING A 3D CUBE

The rendering of a 3D cube defined in the file “**Cube.wrl**” is displayed below . The Cube is defined by the following vertices :

- I. 0,0,0, #vertex 0
- II. 1,0,0, #vertex 1
- III. 1,0,1, #vertex 2
- IV. 0,0,1, #vertex 3
- V. 1,1,0, #vertex 4
- VI. 1,1,1, #vertex 5
- VII. 0,1,1, #vertex 6
- VIII. 0,1,0, #vertex 7

The front face is given by the following ordering of the vertices: 2,5,6,3,-1

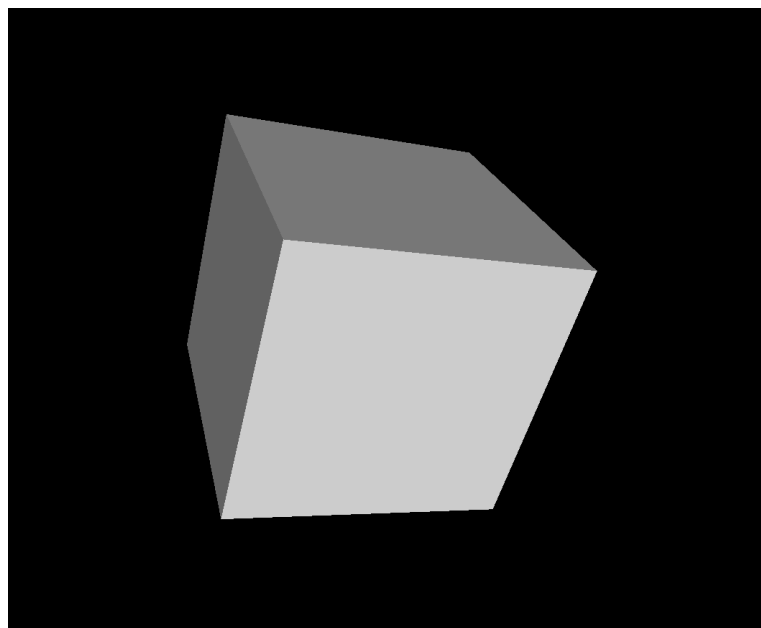
The back face is given by the following ordering of the vertices: 0,7,4,1, -1

The top face is given by the following ordering of the vertices: 5,4,7,6, -1

The bottom face is given by the following ordering of the vertices: 2,3,0,1, -1

The right face is given by the following ordering of the vertices: 1,4,5,2, -1

The left face is given by the following ordering of the vertices: 5,4,7,6, -1





## EFFECT OF ORDERING OF VERTICES ON VISIBLE SIDE OF POLYGONS

The ordering of vertices should be such that if you are looking at the face, the index values should be entered in anticlockwise order for the face to be facing you, and not away from you.

If the ordering of the vertices is changed then the image which we see is different.

Below is an example of the effect of ordering of vertices on visible sides of polygons using the 3D cube defined in the file "**Transparent Cube.wrl**":

Original ordering of vertices: 0,7,4,1, -1

New ordering of vertices: 1,4,7,0, -1

