

CZ2003

Computer Graphics and Visualization

Lab 1 Report: Visualization using polygons

SSR2

U1821610C

Liew Zhi Li (Sherna)

Contents

[Exercise 1 - Graphics Modes 2](#_Toc24734319)

[Exercise 2 – Color, Transparency & Shininess 4](#_Toc24734320)

[Exercise 3 – Change Polygon Mesh 7](#_Toc24734321)

[Exercise 4 - Extras 10](#_Toc24734322)

# Exercise 1 - Graphics Modes

Explore different Graphics Modes of the VRML browser.

|  |  |
| --- | --- |
| **Snapshot** | **Notes** |
| Pyramid (Smooth) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the smooth view of the pyramid.  We can access this by Right Click > Graphics > Smooth. |
| Pyramid (Flat) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the flat view of the pyramid.  We can access this by Right Click > Graphics > Flat.  The visual difference between flat and smooth view is that:  For flat shading, it uses the simplest shading method of applying only one illumination calculation for a single center point of each polygon.  For smooth shading, it uses [Gouraud](https://en.wikipedia.org/wiki/Gouraud_shading) shading. Illumination values for a polygon are interpolated from one vertex to another across the surface. For a polygon, illumination is calculated at its vertices and interpolating illumination values between the vertices using bilinear interpolation. |
| Pyramid (Wireframe) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the wireframe view of the pyramid.  We can access this by Right Click > Graphics > Wireframe. |

|  |  |
| --- | --- |
| Pyramid (Vertices) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the vertices view of the pyramid.  It is difficult to see as vertices are just points on the screen.  We can access this by Right Click > Graphics > Vertices. |
| Pyramid (Solid) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the solid view of the pyramid.  The edges of the pyramid are outlined in white.  We can access this by Right Click > Graphics > Solid. |
| Pyramid (Hidden Line) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the Hidden Line view of the pyramid.  The edges of the pyramid that is visible from the user’s point of view is outlined in white.  We can access this by Right Click > Graphics > Hidden Line. |
| Pyramid (Bounding Box) | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  This is a snapshot of the Bounding Box view of the pyramid.  The bounding box is a box that encloses the pyramid.  We can access this by Right Click > Graphics > Hidden Line. |

# Exercise 2 – Color, Transparency & Shininess

Examine how the color, transparency and shininess of the shape defined in diffuseColor, transparency, shininess field can be changed respectively.

|  |  |
| --- | --- |
| **Snapshot** | **Notes** |
| Red Pyramid | The polygon mesh in “1.1-polygons.wrl” displays a pyramid.  The diffuseColor field defines the color of the geometry.  It accepts 3 inputs in this order: Red, Green, Blue.  Color values must be real numbers between 0 and 1.  When we define  diffuseColor 1 0 0  The color of the pyramid is red. |
| Green Pyramid | The polygon mesh in “1.2-polygons.wrl” displays a pyramid.  To change the color of the pyramid to green, simply define  diffuseColor 0 1 0 |
| Darker Green Pyramid | The polygon mesh in “1.3-polygons.wrl” displays a pyramid.  As the color value ‘Green’ decreases from 1 to 0.5, the color of the pyramid becomes darker.  diffuseColor 0 0.5 0 |
| Black Pyramid | The polygon mesh in “1.4-polygons.wrl” displays a pyramid.  When all 3 RGB values are 0, the color of the pyramid is black.  diffuseColor 0 0 0 |

|  |  |
| --- | --- |
| Yellow Pyramid | The polygon mesh in “1.5-polygons.wrl” displays a pyramid.  To get a yellow pyramid, we can simply define  diffuseColor 1 1 0 |
| Red Pyramid | The polygon mesh in “1.6-polygons.wrl” displays a pyramid.  What happens if the color value is less than 0?  We will use the preceding yellow pyramid for this example.  diffuseColor 1 -1 0  When we set green value to -1, the software will take -1 as 0, effectively this is the same as defining 1 0 0 as we get a red pyramid. |
| Brighter Yellow Pyramid (Front view)    Brighter Yellow Pyramid  (Side view) | The polygon mesh in “1.7-polygons.wrl” displays a pyramid.  diffuseColor 1 2 0  When the color value ‘Green’ greater than 1, in this case, we set it to 2, the color of the pyramid becomes an even brighter yellow colour, compared to the yellow pyramid in “1.5-polygons.wrl”.  When we look at the pyramid from the front view, it is a bright yellow colour.  When we rotate the pyramid to view from the side, it looks neon green.  Although the color value should be between 0 and 1, the software can accept values greater than 1 and it will perform calculations to reflect this special bright yellow/neon green colour. |

|  |  |
| --- | --- |
| Transparent Red Pyramid | The polygon mesh in “1.8-polygons.wrl” displays a pyramid.  diffuseColor 1 0 0  transparency 0.75  This time, we explore what happens when we change the transparency value from 0 to 0.75.  When transparency was set to 0, it was opaque.  Now, when transparency is set to 0.75, we can see that the originally opaque red pyramid has turn slightly transparent. |
| Red Pyramid  (Shininess = 1)    Red Pyramid  (Shininess = 0.2)    Red Pyramid  (Shininess = 0) | The polygon mesh in “1.9-polygons.wrl” displays a pyramid.  diffuseColor 1 0 0  shininess 1  We explore what happens when we change the shininess value.  When shininess = 1, the surfaces of the pyramid is least shiny or less reflective.  When shininess = 0.2, the surfaces of the pyramid is much more shiny or reflective. We can see the color of the pyramid is a red gradient.  When shininess = 0, the surfaces of the pyramid is the most shiny or reflective. We can see that the color of the pyramid has turned white. It has no red at all as the surfaces of the pyramid has maximum shininess. |

# Exercise 3 – Change Polygon Mesh

Change the pyramid to a 3D cube and 2D hexagon by adding new vertices and polygons.

|  |  |  |
| --- | --- | --- |
| **Snapshot** | **Notes** | |
| 3D Cube | The polygon mesh in “2.1-cube.wrl” displays a 3D cube.  In order to transform a pyramid into a 3D cube, we need to add new vertices and polygons.  A cube has 8 vertices.  We create 8 vertices, label them from vertex 0 to 7 and define the x, y, z coordinates of each vertex.  Use the Right-Hand rule to determine the order of the coordinate index for each side of the cube.  Background {skyColor 0 0 0}  Shape {      appearance Appearance{          material Material {              diffuseColor 1 1 0              specularColor 1 1 1              transparency 0              shininess 1              }          }      geometry IndexedFaceSet {          coord Coordinate {  point [                  # bottom vertices                  -1.0 -1.0 1.0,   #vertex 0                  1.0 -1.0 1.0,   #vertex 1                  1.0 -1.0 -1.0,   #vertex 2                  -1.0 -1.0 -1.0,   #vertex 3                  # top vertices                  -1.0 1.0 1.0,    #vertex 4                  1.0 1.0 1.0,    #vertex 5                  1.0 1.0 -1.0,    #vertex 6                  -1.0 1.0 -1.0,    #vertex 7  ]  }          coordIndex [              #bottom square               0, 3, 2, 1, -1,              #side1               5, 4, 0, 1, -1,              #side2               1, 2, 6, 5, -1,              #side3               7, 6, 2, 3, -1,              #side4               7, 3, 0, 4, -1,              #top square               6, 7, 4, 5, -1,              ]      }  } | |
| 2D Hexagon | | The polygon mesh in “2.2-hexagon.wrl” displays a 2D hexagon.  In order to transform a pyramid into a 2D hexagon, we need to add new vertices and polygons.  A 2D hexagon has 7 vertices.  We create 7 vertices, label them from vertex 0 to 6 and define the x, y, z coordinates of each vertex.  Background {skyColor 0 0 0}  Shape {      appearance Appearance{          material Material {              diffuseColor 1 1 0              specularColor 1 1 1              transparency 0              shininess 1              }          }      geometry IndexedFaceSet {          coord Coordinate {  point [                  # bottom vertices                  0       0       0,  #vertex 0                  0       1       0,  #vertex 1                  0.87    0.5     0,  #vertex 2                  0.87    -0.5    0,  #vertex 3                  0       -1      0, #vertex 4                  -0.87   -0.5    0,  #vertex 5                  -0.87   0.5     0,  #vertex 6  ]  }          coordIndex [              #side1               1, 6, 5, 4, 3, 2, -1,              ]      }  }  How do we obtain these x, y, z coordinate values?  Draw the diagram on a piece of paper as shown below.    Next, draw a horizontal line from vertex 6 towards the y-axis. Let this distance be x. To calculate the distance of x | |

# Exercise 4 - Extras

|  |  |
| --- | --- |
| **Snapshot** | **Notes** |
| 2D Heart | The polygon mesh in “3.1-heart.wrl” displays a 2D heart shape.  In order to transform a pyramid into a 2D heart shape, we need to add new vertices and polygons.  A 2D hexagon has 8 vertices.  We create 8 vertices, label them from vertex 0 to 7 and define the x, y, z coordinates of each vertex.  Background {skyColor 0.25 0.25 0.25}  Shape {    appearance Appearance {      material Material {diffuseColor 1 0.43 0.71}    }    geometry IndexedFaceSet {      coord Coordinate {        point [                           # define the 8 points          0 0 0,      0.3 0.3 0,      0.6 0.3 0,      1 0 0,      0 -1 0,      -1 0 0,      -0.6 0.3 0,      -0.3 0.3 0        ]      }      coordIndex [        0,7,6,5,4,3,2,1, -1      ]    }  }  How do we obtain these x, y, z coordinate values?  Draw the diagram on a piece of paper as shown below. |
| Octahedron (Smooth) | The polygon mesh in “3.2-octahedron.wrl” displays a octahedron.  An octahedron has 6 vertices.  We create 6 vertices, label them from vertex 0 to 5 and define the x, y, z coordinates of each vertex.  The Viewpoint defines a camera at 0 -5 10, orientation 1 0 0 0.39, fieldOfView 0.6.  Background {skyColor 0.25 0.25 0.25}  Viewpoint {  position 0 -5 10  orientation 1 0 0 0.39  fieldOfView 0.6  }  Shape {      appearance Appearance{          material Material {              diffuseColor 0 90.6 92.2              specularColor 1 1 1              transparency 0.5              shininess 0.1              }          }      geometry IndexedFaceSet {          coord Coordinate {  point [                  # bottom vertices                  -1.0 -1.0 1.0,   #vertex 0                  1.0 -1.0 1.0,   #vertex 1                  1.0 -1.0 -1.0,   #vertex 2                  -1.0 -1.0 -1.0,   #vertex 3                  # top vertex                  0.0 1.0 0.0    #vertex 4                  #bottom vertex                  0.0 -3.0 0.0    #vertex 5  ]  }          coordIndex [              #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,              #side5              1, 0, 5, -2,              #side6              2, 1, 5, -2,              #side7              3, 2, 5, -2,              #side8              0, 3, 5, -2,              ]      }  } |