Nanyang Technological University

Lab 1 Report:

CZ2003 Computer Graphics and Visualization

Virtualization using polygons

Sim Jun Kai (U1721530F) SSP1

## 2D – Hexagon

We used the given polygons source file as a reference to change into a 6-sided hexagon. The requirement is to view it as 2D and change the displayed polygon mesh into hexagon. Through the source file, we added two vertices to the original square base, then we connect the six vertices together with the functionality; geometry IndexedFaceSet: coordIndex. The order of the vertices in coordIndex affects how the point will be connected. It gives us the option of either going clockwise and anti-clock wise, but we need to set the coordinate points at geometry IndexedFaceSet: coord Coordinate beforehand. We plot the points in anti-clockwise direction, e.g.

(-0.5 1 0) is vertex 0 and top left point

(1 1 0) is vertex 1 and top right point

(1.5 0 0) is vertex 2 and medium right point

(1 -1 0) is vertex 3 and bottom right point

(-0.5 -1 0) is vertex 4 and bottom left point

(-1 0 0) is vertex 5 and medium left point

We connect the point from 0 -> 5 -> 4 -> 3 -> 2 -> 1. Hence a single sided hexagon is formed.

## 3D Cube

The concept of creating 3D cube is the same as creating hexagon. We used the given polygons source file as reference. Instead of adding two vertices, we duplicated the square base and plot it on a bigger (on top of the square base) coordinate point. Just to note, you can also put the duplicated square base below your original base. We will have 8 vertices in total and we can start connecting the side of the original square base and the duplicated square base. In total, there are two solutions in making 3D cube. The first solution is we will have 12 sets of geometry IndexedFaceSet: coordIndex. We have catered for the observation the 3D cube from all direction even the inside of the cube. Another solution is 10 sets of geometry IndexedFaceSet: coordIndex. Personally, there does not have any wrong with the solution itself, because we do not cater to audience who are inside the cube. We will just view the 3D cube from the exterior. However, if there is an observer viewing from the interior of the cube, there will be empty space at the top and bottom of the cube. We will plot the cube, e.g.

Bottom of the Cube (base)

(-1 -1 1) is vertex 0 and base top left point

(1 -1 1) is vertex 1 and base top right point

(1 -1 -1) is vertex 2 and base bottom right point

(-1 -1 -1) is vertex 3 and base bottom left point

Top of the Cube (duplicated base **DB**)

(-1 1 1) is vertex 4 and **DB** top left point

( 1 1 1) is vertex 5 and **DB** top right point

( 1 1 -1) is vertex 6 and **DB** bottom right point

(-1 1 -1) is vertex 7 and **DB** bottom left point

We connect the point of the two bases first. 0 -> 3 -> 2 -> 1 and 4 -> 5 -> 6 -> 7. Afterwards, we create the side of the cube by linking the base top left to **DB** top left 0 -> 4 -> 5 -> 1 and 1 -> 5 -> 4 -> 0 to prevent empty spaces when rotating. Afterwards the right side of the cube 1 -> 5 -> 6 -> 2 and 2 -> 6 -> 5 -> 1. The bottom left the **DB** left 3 -> 7 -> 6 -> 2 and 2-> 6 -> 7 -> 3. Lastly, 0 -> 4 -> 7 ->3 and 3 ->7 -> 4 -> 0 to cover the cube. In total 10 sets of coordIndex. The additional two coordIndex which I never include because I personally feel that we are viewing from the exterior and not interior is 1 -> 2 -> 3 -> 0 and 7 -> 6 -> 5 -> 4.

## Polygons

DEF \_1 Background set the background colour to what was defined in the blanket. The colour order goes in red, green and blue. We also pre-set the appearance of the hexagon to be the same of the polygons.