COSC363 Computer Graphics
Lab05: Sweep Surfaces

Aim:

This lab aims to provide an understanding of the modeling aspects of sweep surfaces such as surfaces of revolution and extruded shapes.

I. Tower.cpp:

The "Turning Torso" (Fig. a), a twisted tower in Malmo, Sweden is an excellent architectural example of a sweep surface/structure (For more information on the tower, see: http://en.wikipedia.org/wiki/Turning_Torso). We can generate a model of the tower using the shape shown in Fig (b) as the base polygon. The program `Tower.cpp` contains a representation of the polygonal shape using vertex coordinates stored in arrays vx[], vy[], vz[] inside the `display()` function. The polygon has 18 vertices as shown in Fig. (c).

Fig. (a)

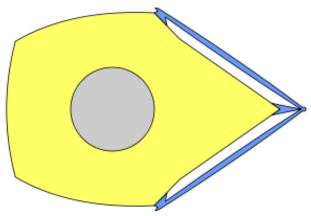


Fig. (b)

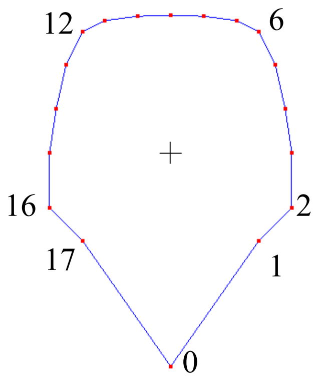


Fig. (c)

The tower consists of 9 "blocks", each block having a height of approx. 20 meters, and turned *clockwise* about the vertical axis by 10 degrees relative to the lower block.

1. The program displays just the base polygon. The camera can be moved around the scene using the left and right arrow keys.

2. Delete the code segment that draws the base polygon inside the `display()` function and implement the following algorithm:

- Generate a new set of transformed coordinates wx[i], wy[i], wz[i], i = 0.. N–1 (already declared in the program) by rotating the points (vx[i], vy[i], vz[i]) by -10 degs (clockwise) about the y-axis and translating along y-axis by 20 units. Since we require the transformed points, we cannot use the OpenGL function `glRotatef()` to perform the rotation. Instead, we use the following equations:

  $w_x = v_x \cos\theta + v_z \sin\theta$, $\theta = -10$ Degs (Convert to radians!)

  $w_y = v_y + 20$

  $w_z = - v_x \sin\theta + v_z \cos\theta$

- Join the points V_i = (vx[i], vy[i], vz[i]) and the transformed points W_i = (wx[i], wy[i], wz[i]) using a quad strip to generate the surface of the extruded shape. Refer to the code given on Slide [5]-20.

- After drawing the quad-strip, replace the values in (vx[], vy[], vz[]) with the transformed values in (wx[], wy[], wz[]).

The above three steps generate the surface of one "block" of the tower. Repeat the steps 8 more times.

The light and camera positions have already been defined in the program. Change the value of the global variable `viewAngle` to -160 degs. The program should generate an output similar to the one given in Fig. (d).

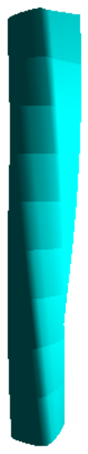


Fig. (d)

3. The program includes the necessary functions for loading a bitmap texture "TowerTexture.bmp". Please uncomment the three lines inside the `initialise()` function to enable texture mapping. The texture has 5 parts corresponding to 5 sides of the tower (Fig (e)).

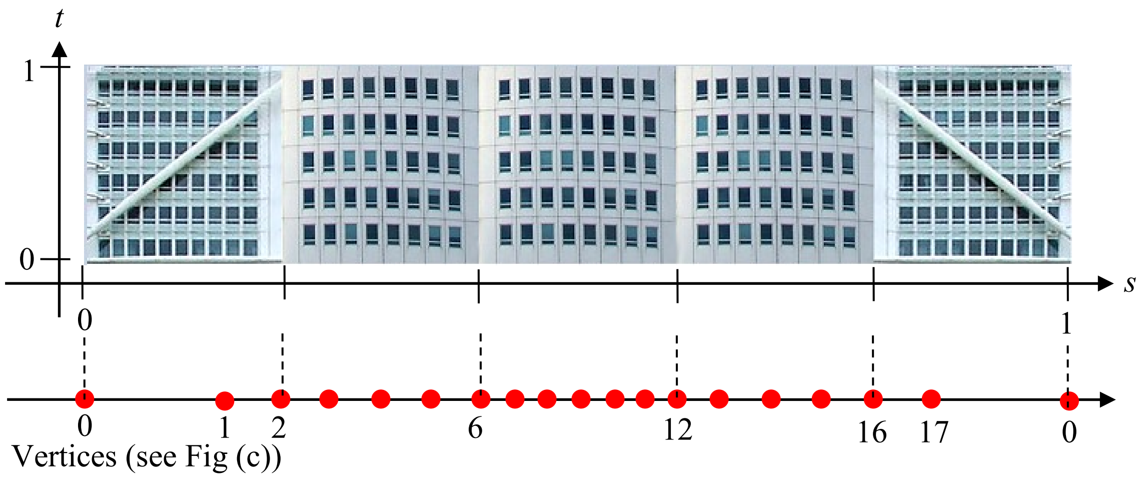
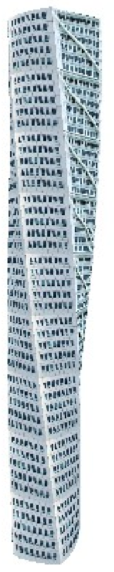


Fig. (e)

4. We need to assign texture coordinates to vertices of the quad strip. If vertex V_i has texture coordinates $(s_i, 0)$, then the corresponding vertex W_i will have texture coordinates $(s_i, 1)$. The values s_i must be carefully chosen so that the image sections in the texture are properly mapped to the 5 sides of the polygon (see Fig. (c)). Fig(e) also shows how this mapping must be done. The table below gives the texture *s*-coordinates for some of the vertices. Compute the remaining coordinates, assuming that the points are uniformly distributed within each of the 3 middle sections. Your program should then produce an output similar to the one given in Fig. (f).

Vertex index i	Tex Coord s_i	 Fig. (f). Textured tower
0	0	
1	0.15	
2	0.2	
:	:	
6	0.4	
:	:	
12	0.6	
:	:	
16	0.8	
17	0.85	
0	1	

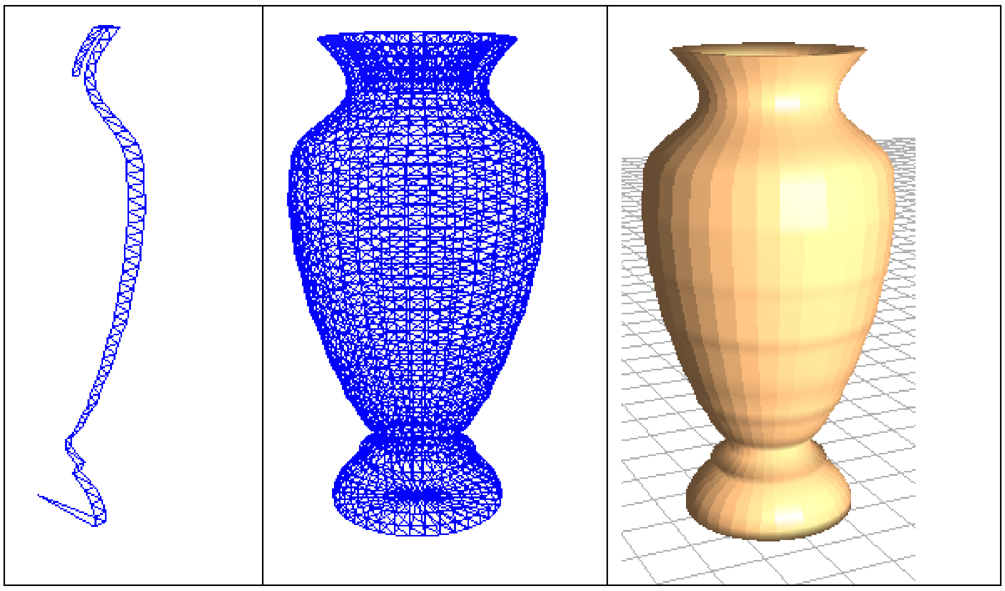
II. Vase.cpp:

The program `Vase.cpp` contains the vertex data for a base curve in arrays vx[*i*], vy[*i*], vz[*i*], $i = 0 \ldots N\text{-}1$, N=50. It displays only the floor grid of the scene. The camera can be rotated using left/right arrow keys. The up/down arrow keys change the height of the camera.

1. Create a triangle strip using the base curve, by first rotating each point (vx[*i*], vy[*i*], vz[*i*]) about the y-axis by 10 degrees to form a new set of points (wx[*i*], wy[*i*], wz[*i*]). Use the following equations:
   $w_x = v_x \cos\theta + v_z \sin\theta$, θ = 10 Degs (Convert to radians!)
   $w_y = v_y$
   $w_z = - v_x \sin\theta + v_z \cos\theta$
   These points W_i define the rotated base curve. Construct a triangle strip (Fig. (g)) using the method given on slide [5]-18.

2. Copy the coordinates W_i to V_i for the next iteration, and repeat the steps 36 times to get a 360 degree revolution of the base curve (Fig. (g)). Steps 1 and 2 can be implemented as nested loops:
```
for(int j = 0; j < 36; j++)      //36 slices in 10 deg steps
{
     for(int i = 0; i < N; i++)   //N vertices along each slice
     {
       ...
```

3. Inside the `initialise()` function, change the polygon mode from GL_LINE to GL_FILL. Inside `display()`, change the color of the vase from blue to (1.0, 0.75, 0.5, 1.0) and enable lighting. You should also perform normal computation for each triangle in the strip, as shown on slide [5]-18. Note that each triangle strip is an open strip, so there is no need to add extra triangles at the end to "close" the stip. However you will need to reverse the direction of the normal vectors (Why?) to get proper lighting on the vase (Fig. (g).


(g)

4. The program includes the function to load a bitmap texture "VaseTexture.bmp". Call this function inside `initialise()`, and enable texturing. Please note that texture mapping is disabled inside the floor function. You should therefore enable texturing of the vase by including the function call glEnable(GL_TEXTURE_2D) *after* calling floor() in the display() function. We will texture map the image to the whole surface generated above. Assign texture coordinates to the points on the triangle strip, assuming that the points are nearly equally spaced. The values of the texture *s* coordinate can be defined based on the slice index *j*, and the *t* coordinate based on the vertex index *i* in the array V_i: $s = j / 36.0$; $t = i$ / (float)N.

   The textured output is shown in Fig (h).

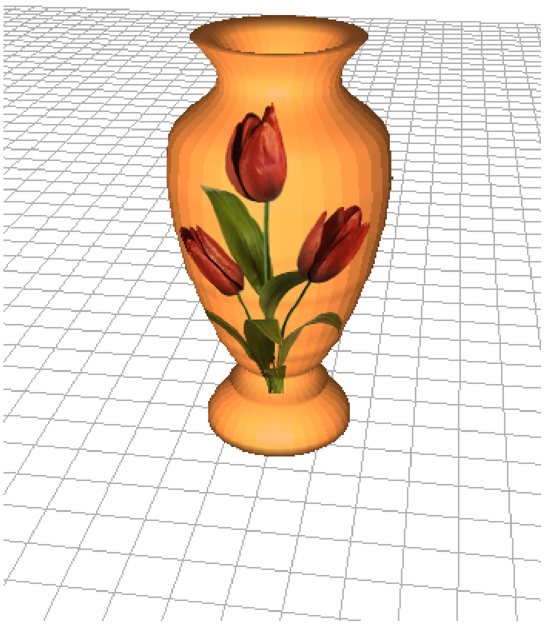


Fig (h)

Ref:

[5] Lec05_ObjectModelling.pdf (COSC363 Lecture notes)

III. Quiz-05

The quiz will remain open until **5pm, 5 April, 2019**.