COSC363 Computer Graphics
Lab04: 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. This lab also introduces the procedure for texture mapping a set of polygonal surfaces.

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 the arrays vx[], vy[], vz[] inside the display() function.

← Fig (a)

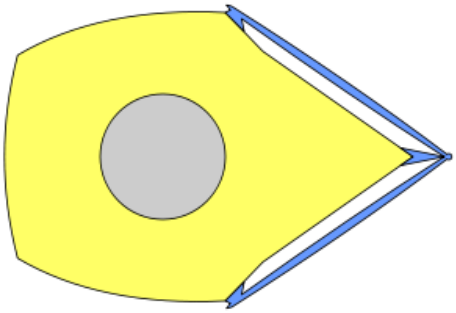


Fig (b).

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 polygon has 18 (= *N*) vertices. The camera can be moved around the scene using the left and right arrow keys.

2. Delete the code 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. Use the following equations instead:

  $w_x = v_x \cos\theta + v_z \sin\theta$, θ = -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 [6]-10.

- Replace the values in (vx[], vy[], vz[]) with the transformed values in (wx[], wy[], wz[])

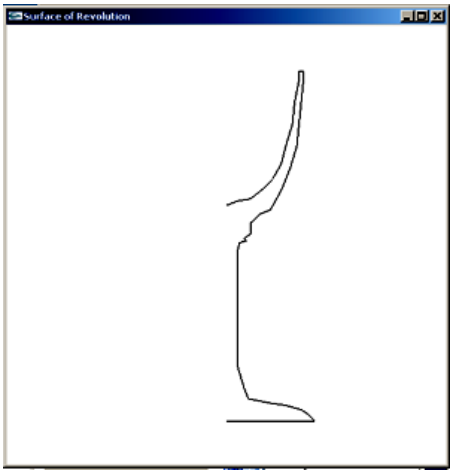
The above three steps generate the surface of one "block" of the tower. Repeat the steps 8 more times.

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. (c).

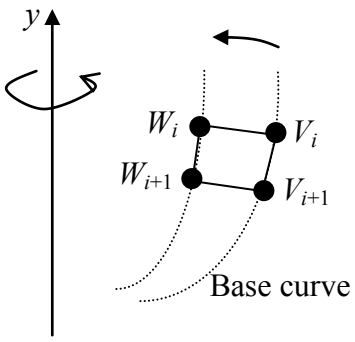
Fig. (c)

II. SurfRevln.cpp:

1. The program SurfRevln.cpp requires the input file "Wineglass.dat". This file contains the two dimensional coordinates of 35 points along a curve that we will use to generate the 3D model of a wineglass. The output of the program is shown in Fig. (d). The vertex coordinates V_i are stored in the arrays vx[*i*], vy[*i*], vz[*i*], *i* = 0..34.



(d)



(e)

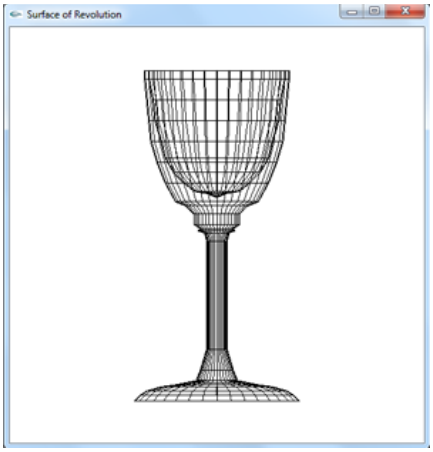
2. The base curve is drawn using the primitive type GL_LINE_STRIP inside the display function. Replace this code segment (marked –A--) with the code that performs the following tasks:

(Step 1). Each point (vx[i], vy[i], vz[i]) is rotated about the y-axis by 10 degrees. Use the following equations:
$w_x = v_x \cos\theta + v_z \sin\theta$, $\theta = 10$ Degs (Convert to radians!)
$w_y = v_y$
$w_z = -v_x \sin\theta + v_z \cos\theta$
The transformed coordinates must be stored in the arrays wx[i], wy[i], wz[i]. These points W_i define the rotated base curve.

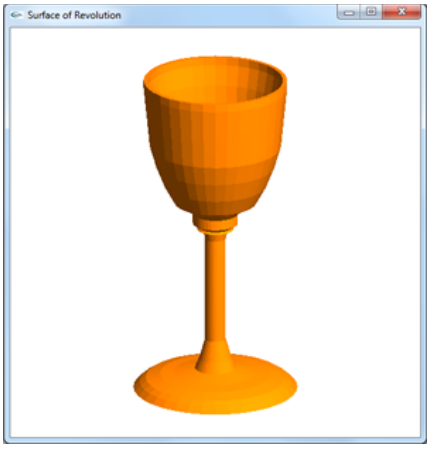
(Step 2). After obtaining the points (wx[i], wy[i], wz[i]), $i = 0..34$, we need to connect the points together to form a mesh surface (Fig. (e)). Define quads (GL_QUADS) using points $V_iV_{i+1}W_{i+1}W_i$, $i = 0..33$.

(Step 3). Copy the coordinates W_i to V_i for the next iteration.

The above steps are typically implemented using 3 separate for-loops. Repeat these steps 36 times (by adding another outer for-loop) to get a 360 degree revolution of the base curve (Fig. (f)). Use arrow keys to rotate the surface about x and y axis (already implemented in the program).



(f)



(g)

3. Converting the above model into a solid model and rendering it under a light source (Fig. (g)) requires the following steps:
   - Inside the initialize() function, change the parameter of glPolygonMode from GL_LINE (wireframe) to GL_FILL (solid).
   - Enable lighting and select a light source (Lab-03).
   - Define light and material characteristics (Lab-03).
   - Define light position inside the display function.
   - Specify normal vectors for each quad. Pass the vertex coordinates of V_i, V_{i+1}, W_{i+1} to the "normal()" function.

III. Yard.cpp:

1. The program displays five polygons (four "walls" and one "floor") forming a rectangular yard (Fig. (h)). The camera is placed at the center of the yard. The arrow keys can be used to change the view direction and position.

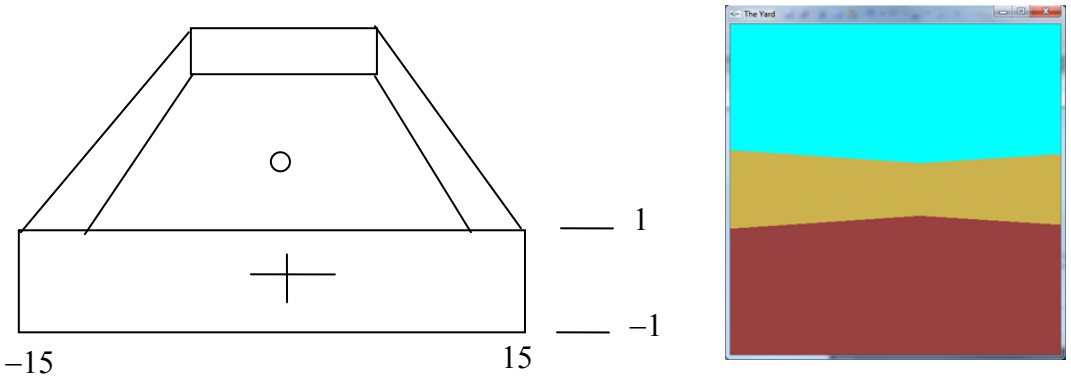
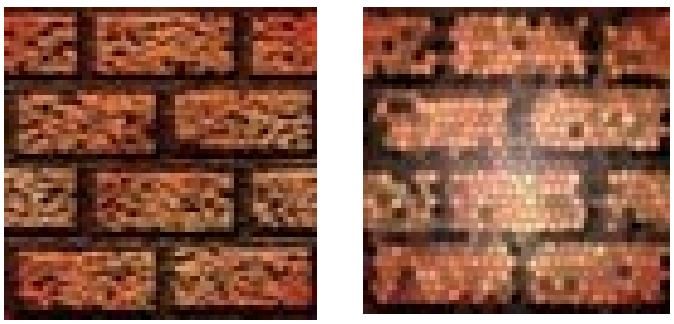


Fig. (h)

2. Two textures "Wall.tga" and "Floor.tga" are provided (Fig. (i)). The textures have a special property that they can be seamlessly tiled any number of times along both s, t directions without any visible discontinuities at boundary pixels. The program contains the necessary function definition to load both these textures. Note also that GL_REPEAT is the default wrapping mode for textures. Inside the `initialise()` function, call the function `loadTexture()`, and enable texture mapping.



Wall.tga Floor.tga

Fig. (i)

3. Using the image "Wall.tga", texture the first quad (in the function wall()) as given below. The texture is repeated 12 times in the horizontal direction, and twice along the vertical direction. Texture the remaining 3 quads also the same way.

```
glTexCoord2f(0.0,  2.0);      glVertex3f(-15, 1, -15);
glTexCoord2f(0.0,  0.0);      glVertex3f(-15, -1, -15);
glTexCoord2f(12.0, 0.0);      glVertex3f(15, -1, -15);
glTexCoord2f(12.0, 2.0);      glVertex3f(15, 1, -15);
```

4. Similarly, texture map the floor using the second texture image "Floor.tga", with a repetition count of 16 along both directions. The final output is shown in Fig. (j).

Fig. (j)

IV. Quiz-04

The quiz will remain open until **5pm, 4-April-2014**.

A quiz can be attempted only once. A question within a quiz may be attempted multiple times. However, a fraction of the marks will be deducted from the second attempt of each question.