COSC363 Computer Graphics
Lab02: Modelling and Transformations

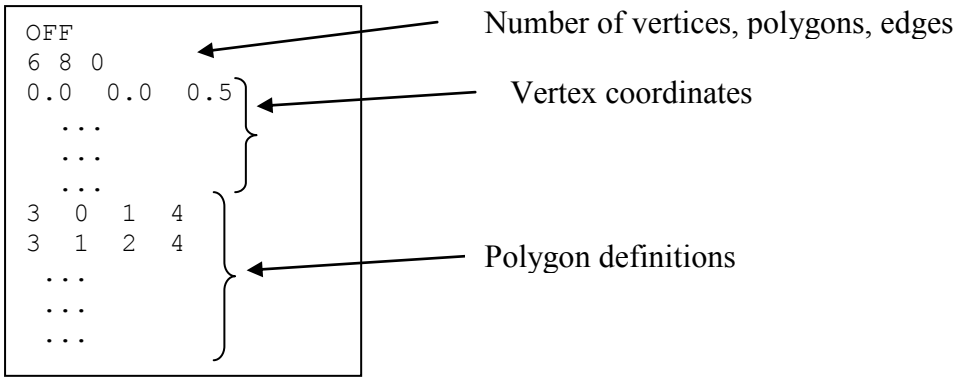
Aim:

In this lab, you will learn the following methods that are fundamental to the rendering of a scene.

- Loading 3D mesh models in OFF format and rendering them using OpenGL primitives.
- Using vector operations to compute surface normal directions so that OpenGL lighting model.
- Applying independent transformations to a set of GLUT objects to form a composite model, and animating the model using rotational transformations about pivot points.

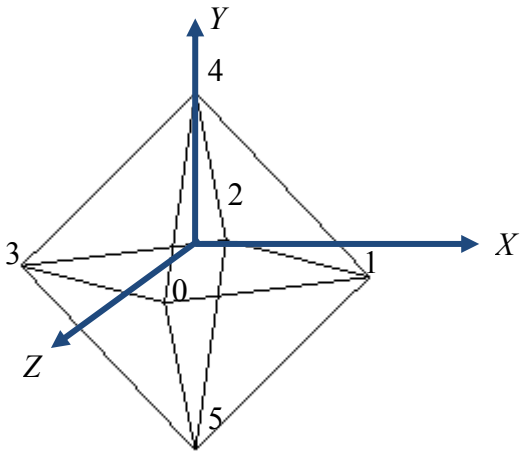
I. Object File Format (OFF):

The Object File Format (.OFF) is a convenient ASCII format for storing 3D model definitions (meshes). It uses simple vertex-list and polygon-list structures for specifying a polygonal model. Unlike other 3D mesh formats, the OFF format does not intersperse commands with values on every line, and therefore can be easily parsed to extract vertex coordinates and triangle indices. The basic structure of a mesh file in OFF format is shown in Fig. (a).



(a)

The first line contains the header keyword OFF. The second line contains the total number of vertices (n_v), total number of polygons (n_p) and edges (n_e) in the model. The number of edges is always set to 0. The second line will be followed by n_v lines containing three-dimensional coordinates of the vertices. The file will then contain n_p lines defining each polygon in terms of its vertex indices. The first number in a polygon definition indicates the number of vertices of that polygon. For a triangular mesh, this number will always be 3. The following three numbers give the vertex indices of that polygon. We will consider a simple example below.



(b)

An octahedron contains 6 vertices and 8 triangles. The octahedron in Fig. (b) intersects each of the principal axes at a distance 0.5 units from the origin. Thus the first vertex (with index 0) has coordinates (0, 0, 0.5), the second vertex has index 1 and coordinates (0.5, 0, 0) and so on. Now refer to Fig. (a), which gives the model definition for this shape. The first triangle has vertex indices 0, 1, 4. Triangles can be defined in any order, but note that the vertices must be oriented in an anti-clockwise sense when viewed from outside the model. This will ensure that the surface normal vectors are directed outwards from each triangle, as required by illumination models.

1. Create a plain text file with name "Octahedron.off" and create the model definition for the octahedron (by completing the missing entries in Fig. (a)).

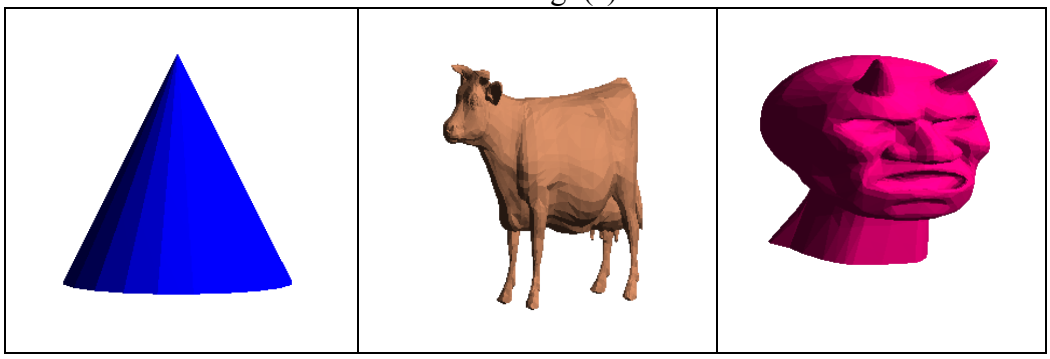
2. The program **`Model3D.cpp`** includes the function and variable declarations for reading data from an OFF file. The vertex coordinates are stored in arrays `x, y, z;` and the triangle indices in arrays `t1, t2, t3`. The variable `nvrt` stores the total number of vertices, and `ntri` the total number of triangles. Note that the `display` function contains a `glBegin()..glEnd()` block to render the triangles. If your model definition is correct, the output should look similar to that given in Fig. (b). The program also allows you to rotate the object using left and right arrow keys. Please also note the definition of the callback function for processing special keyboard events.

3. Each model will have its own range of values for the coordinates. The camera frustum dimensions must be calculated based on the min, max values of the coordinates so that the entire model is contained within the view frustum. Note how this computation is done inside the `initialize()` function.

4. Three models "Cone.off", "Cow.off", "Oni.off" are provided in the file Models_OFF.zip. Change the mesh file name in the program to get the wireframe displays of these models.

5. Change the polygon display mode from `GL_LINE` to `GL_FILL` in the function call `glPolygonMode(…)`. The display now gives the appearance of a flat surface with uniform colour across all polygons. For the lighting

model to work correctly, OpenGL requires the definition of normal vectors for each polygon. Un-comment the "`normal(..)`" statement in `display`, and also complete the function definition (Slide [2]-14). The output of the program should now be similar to that shown in Fig. (c).

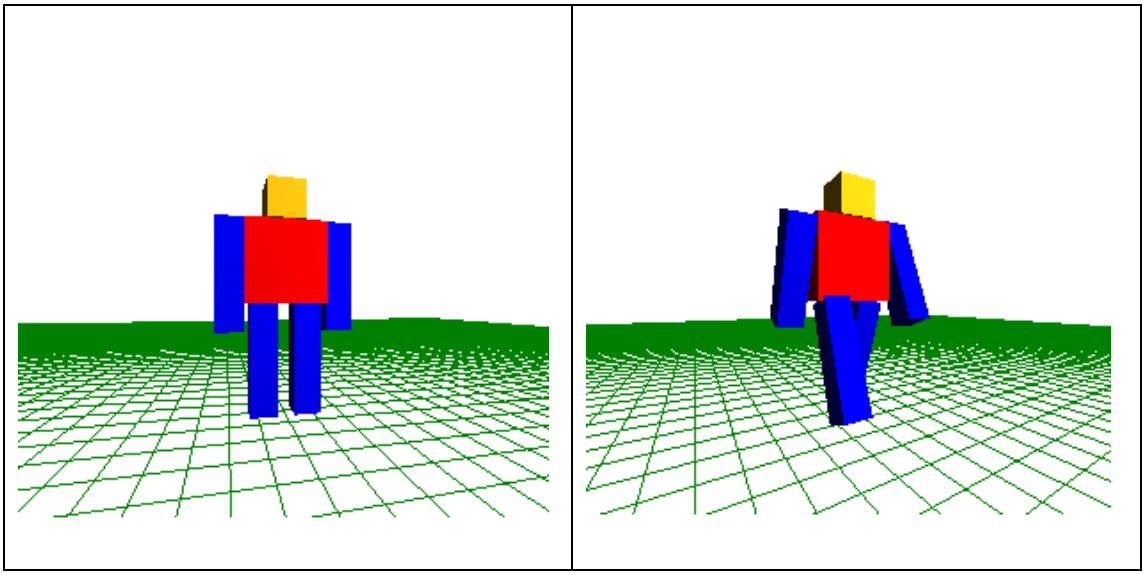


(c)

6. The "Shape Repository" http://shapes.aimatshape.net/ contains a huge collection of downloadable 3D models in OFF format.

II. Transformations:

The program **`Humanoid.cpp`** displays a simple "humanoid" model constructed using six GLUT cubes (Fig. (d)). The structure of the program is similar to "Teapot.cpp" (Lab 1). Please observe how each cube is transformed independently of others to generate the composite model.



(d) (e)

1. Define a global variable `theta` (θ) and initialize it to 20 degs. We perform the following rotational transformations of the arms and the legs of the model.

    - Rotate the left leg and right arm by θ about the *x*-axis. For the left leg, the pivot point is (0, 4.5, 0), and for the right arm the pivot point is (0, 6.5, 0). Refer to Slide [3]-15 for details on rotations about pivot

points. The transformation for the left leg is shown below as an example.

```
glPushMatrix();
  glTranslatef(0, 4.5, 0);
  glRotatef(theta, 1, 0, 0);
  glTranslatef(0, -4.5, 0);
  glTranslatef(0.8, 2, 0);
  glScalef(1, 4, 1);
  glutSolidCube(1);
glPopMatrix();
```

Rotation by θ about the pivot point (0, 4.5, 0)

Note that two successive translations can be combined into a single translation.

- Rotate the right leg and left arm by −θ about the *x*-axis. For the right leg, the pivot point is (0, 4.5, 0), and for the left arm the pivot point is (0, 6.5, 0). Fig(e) given above shows the result of the rotational transforms.

2. The variation of θ from 20 degs to −20 degs and back to 20 degs corresponds to a single "walk cycle". Define a timer function similar to what was used in Lab-1 to implement a continuous walk cycle of the humanoid model.

III. Quiz-02

The quiz will remain open until **5pm, 21-March-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.