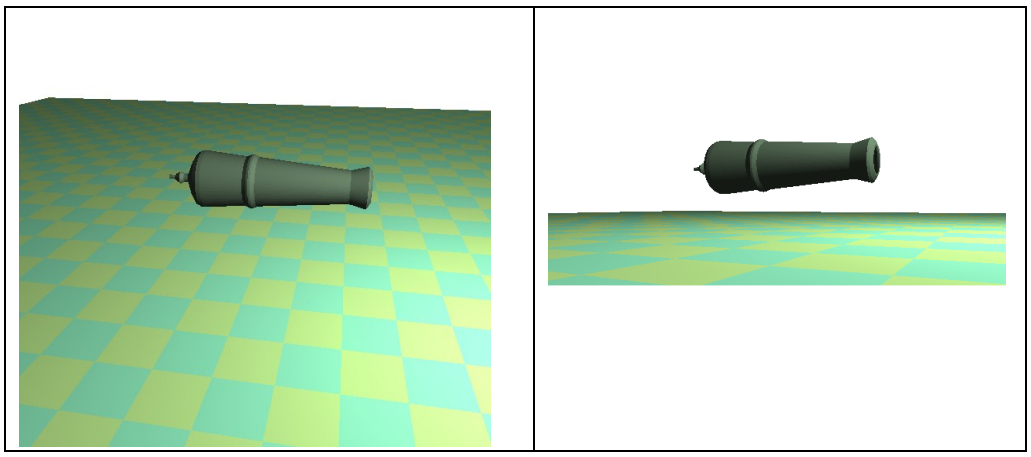
COSC363 Computer Graphics
Lab02: Transformations

Aim:

This lab provides an introduction to transformations required for modeling and animation of objects in a three-dimensional scene.

I. Cannon.cpp:

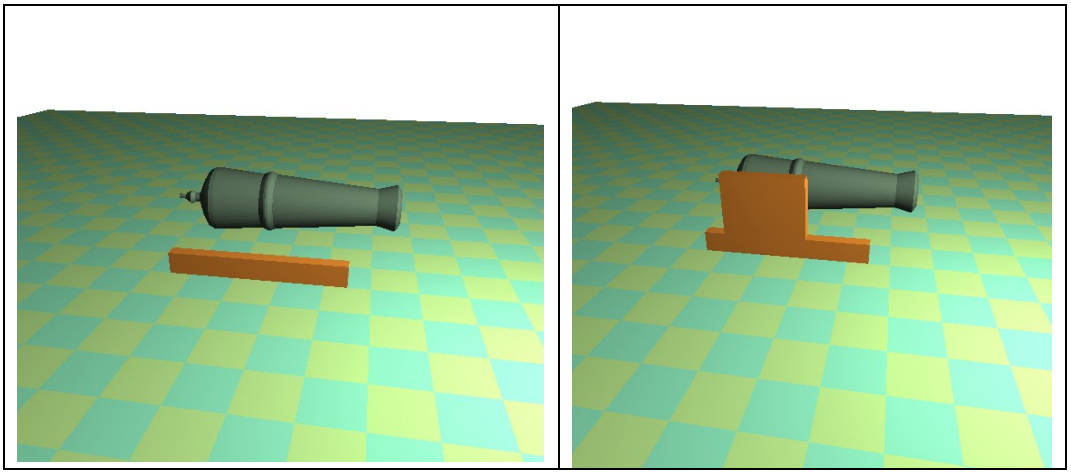
1. The program **`Cannon.cpp`** displays a scene consisting of a model of a cannon (Fig. (a)). The program loads the mesh file for the model from "Cannon.off". The scene can be rotated using left/right arrow keys, and the camera's height can be raised or lowered using up/down keys. As can be seen from Fig. (b), the model's centre is not the origin, but a point above the floor plane.



(a) (b)

We will now design a mounting bracket for the cannon using a set of GLUT cubes. Inside the display() function, start adding the code for shapes and their transformations as detailed below, at the point marked by the comment "`--start here`".

Create a solid cube of size 1 unit. Use scale factors (80, 10, 6) to scale the cube first. Then translate it to the point (–10, 5, 17). Please write the transformation functions in the correct order (see example on Slide [2]-11). You should get the display in Fig. (c). Now create a second cube, scale it by using factors (40, 30, 6) and translate it to the point (–20, 25, 17). The required output is shown in Fig. (d) below; however, you may get a completely different output where only a part of the second cube is visible at the top-left corner of the window. This is because the transformations applied to the first cube were also applied to the second cube. We can separate the transformations meant for one object from those applied to another using `glPushMatrix()...glPopMatrix()` blocks as shown below. These blocks allow us to apply a set of transformations to an object and then discard them before creating the transformations for the next object.

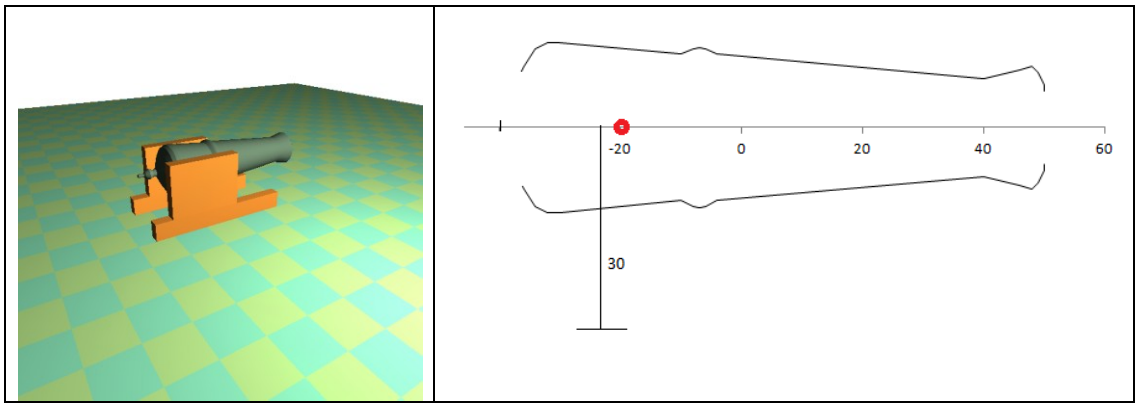


(c) (d)

```
glPushMatrix();
      ....    //Transformations for object 1
     drawObject1();
glPopMatrix();

glPushMatrix();
      ....    //Transformations for object 2
     drawObject2();
glPopMatrix();
```

Create two more cubes with the same scale factors as above, but translate them to positions (–10, 5, –17) and (–20, 25, –17) respectively. This completes the mounting bracket for the cannon (Fig. (e)).

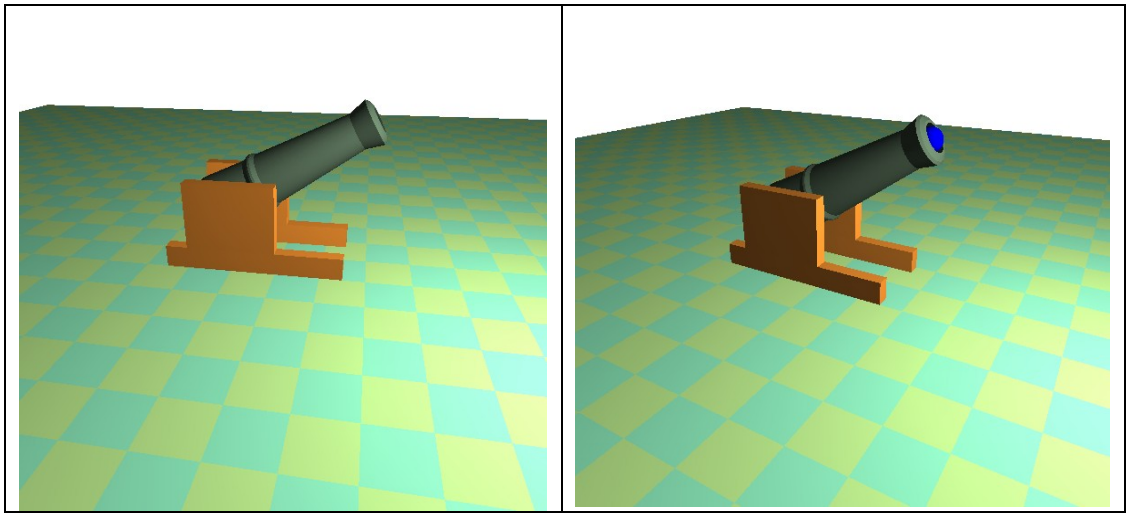


(e) (f)

We will now rotate the cannon about the pivot point shown in Fig. (f). Apply a series of transformations to the cannon (remember to use the pushMatrix-popMatrix blocks) to rotate it by 30 degs about the z-axis, with (–20, 30, 0) as the pivot point (see example on Slide [2]-15).

```
glTranslatef(px, py, pz);       //Pivot point coordinates
glRotatef(angle, axisX, axisY, axisZ);  //Rotation
glTranslatef(–px, –py, –pz);
drawObject();
```

The result of the transformation is shown in Fig. (g).



(g) (h)

Create the model of a solid sphere of radius 5 units using the function call

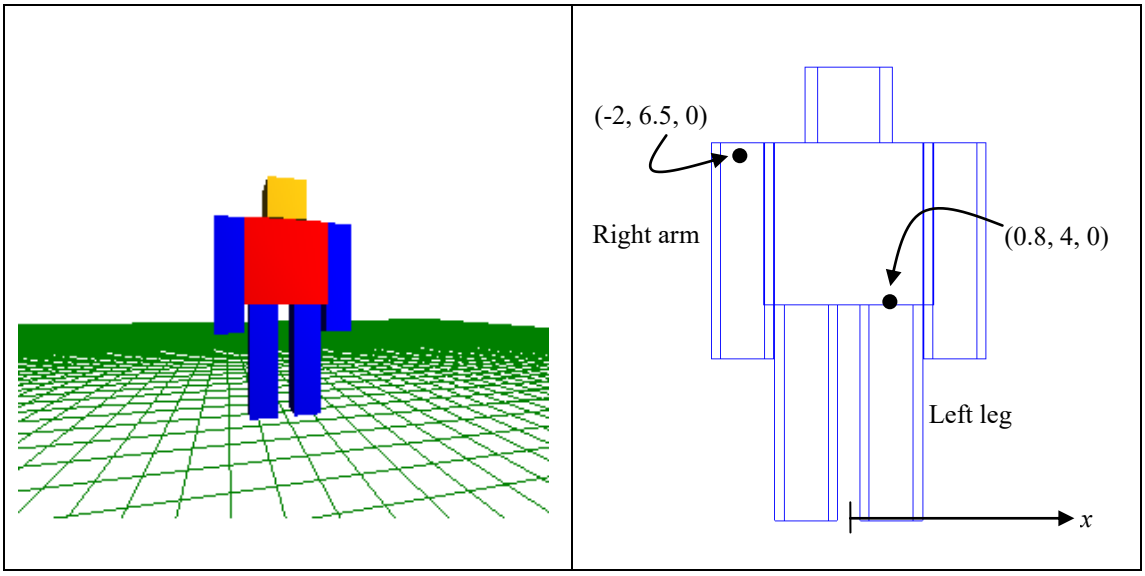
```
glutSolidSphere(5, 36, 18);
```

The second and the third parameters define the subdivisions of the sphere parallel to longitude and latitude lines. We will use this sphere as the projectile. Translate the sphere to the point (38.88, 64, 0). The output is shown in Fig. (h).

Take-home exercise: Implement keyboard and timer event call-back functions and transformations of the sphere such that when the space bar is pressed, the cannon "fires" the projectile along a trajectory with some initial velocity.

II. Humanoid.cpp:

The program **`Humanoid.cpp`** displays a simple "humanoid" model constructed using six GLUT cubes (Fig. (i)). The structure of the program is similar to the previous program. The model definition is given in the function **`drawModel()`**. Please observe how each cube is transformed independently of others to generate the composite model. You can rotate the whole scene (equivalent to moving the camera in the opposite direction!) using left/right arrow keys.



(i) (j)

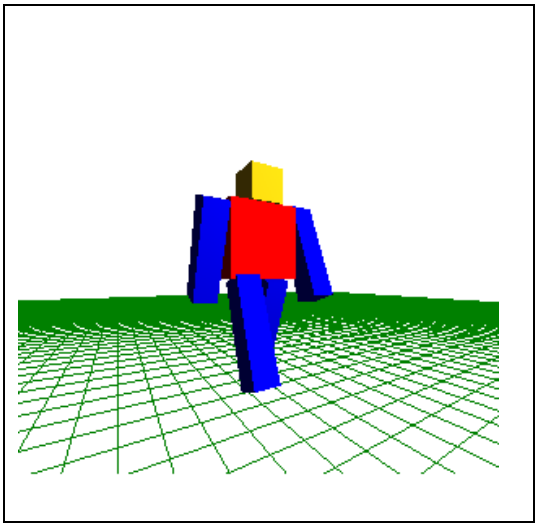
1. Define a global variable `theta` (which represents the rotation angle θ) and initialize it to 20 degs. We perform the following rotational transformations of the arms and the legs of the model.

   - Rotate the left leg and right arm by θ about the *x*-axis. Refer to fig(j) above. For the left leg, the pivot point is (0.8, 4, 0), and for the right arm the pivot point is (–2, 6.5, 0). Refer to Slide [2]:13-15 for details on rotations about pivot points. The transformation for the left leg is shown below as an example.

```
glColor3f(0., 0., 1.);                    //Left leg
glPushMatrix();
  glTranslatef(0.8, 4, 0);
  glRotatef(theta, 1, 0, 0);              Rotation by θ about the
  glTranslatef(-0.8, -4, 0);              pivot point (0.8, 4, 0)
  glTranslatef(0.8, 2.2, 0);
  glScalef(1, 4.4, 1);
  glutSolidCube(1);
glPopMatrix();
```

     Note that two successive translations may be combined into a single translation. The above change must be made inside the drawModel() function only. Similarly rotate the right arm by the same angle theta.

   - Now rotate the right leg and left arm by –θ about the *x*-axis. For the right leg, the pivot point is (–0.8, 4, 0), and for the left arm the pivot point is (2, 6.5, 0). Fig(k) below shows the result of the rotational transforms.



(k)

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 that used for the teapot, and implement a continuous walk cycle of the humanoid model.

[2] COSC363 Lecture slides "2 Transformations".

III. Quiz-02

The quiz will remain open until **5pm, 13-Mar-2020**.

A question within a quiz may be attempted multiple times. However, a fraction of the marks (25%) will be deducted for each incorrect answer.