LAB 02

TRANSFORMATION AND HIERARICHY REPRESENTATION

Objective:

To understand and apply OpenGL transformation function.

OPENGL TRANSFORMATION FUNCTION

OpenGL maintains a 4x4 transformation matrix that is multiplied by every (x,y,z) value sent to glVertex(). This is sometimes referred to as the CT (current transform). The CT can be modified by the following functions:

```
glLoadIdentity();

glTranslate( dx, dy, dz );

glScale( sx, sy, sz);

glRotate ( angleInDegrees, axisXvalue, axisYvalue, axisZvalue);
```

Note the following:

1) The arguments to these functions can be **float** or **double** data types. The last letter of the function name determines the data type of the arguments. For example:

```
glTranslatef(3.0, 4.0, 1.2); // floating point data

glTranslated(3.0, 4.0, 1.2); // double precision data
```

2) The **glLoadIdentity()** function is unique. It **replaces** the CT (current transform) with an identity matrix. This has the effect of starting transformations "from scratch". After the glLoadIdentity() function call, the CT has no effect on the points sent to glVertex() because P = **I**P (a point times an identity matrix does not change).

3) For the glTranslate, glRotate, and glScale functions, each builds a 4x4 transformation matrix using the supplied arguments and then multiplies it with points sent to glVertex(). For example, the glTranslate function might look something like the following. (You will not write this code - it is already in the OpenGL library.)

```
void glTranslatef( float dx, float dy, float dz)
{
  GLfloat m[4][4] = { { 1.0, 0.0, 0.0, dx },
                      { 0.0, 1.0, 0.0, dy },
                      { 0.0, 0.0, 1.0, dz },
                      { 0.0, 0.0, 0.0, 1.0} };
CT = CT*m;
}
```

4) The multiplication of the matrices is always (CT*m), not (m*CT), so the order of the function calls must be in reverse order from the "conceptual" ordering of the transformations. This is best shown by an example. If you want to translate a point by (-2, 5, 0), THEN rotate it by 20 degrees about the Z axis, and THEN scale it by a factor of 2 along the X axis, the function calls would be:

```
glLoadIdentity(); // always start with the identity matrix
glScalef(2.0, 1.0, 1.0);// Conceptually, this happens 3rd
glRotatef(20.0, 0.0, 0.0, 1.0);// Conceptually, this happens 2nd
glTranslatef(-2.0, 5.0, 0.0);   // Conceptually, this happens 1st
drawObject();
```

HOW TO USE OPENGL TRANSFORMATION FUNCTION

1) Working with the correct matrix:
There are two different 4x4 matrices used by OpenGL to produce an image. They are:

- Projection matrix - used to "project" a 3D image onto a 2D plane to create an image.
- ModelView matrix - used to transform objects and the camera to create the desired scene and view.

Commands like glTranslate, glRotate, and glScale will modify the **active matrix**. It is the programmer's responsibility to make sure the appropriate matrix is "active." The active matrix is determined by calls to glMatrixMode(), such as:

```
glMatrixMode(GL_PROJECTION); // make the projection matrix active

glMatrixMode(GL_MODELVIEW); // make the modelview matrix active
```

2) The transformations caused by glRotate, glTranslate, and glScale are applied to the CT (current transform). They will accumulate continually unless you reset the matrix back to the Identity matrix. If you want to start a new set of transformations, then call glLoadIdentity(), as in the following example:

```
glMatrixMode(GL_MODELVIEW);

glLoadIdentity();// start a new sequence of transforms
glTranslatef(-3.0, -4.0, 0.0);
glRotatef(90.0, 1.0, 0.0, 0.0);
drawObject1();

glLoadIdentity();// start a new sequence of transforms
glTranslatef(4.0, 2.0, 1.0);
glScalef(2.0, 1.0, 1.0);
drawObject2();
```

HIERARCHICAL REPRESENTATION

glPushMatrix ()
Push the current matrix stack
glPopMatrix ()
Pops the current matrix stack, replacing the current matrix with the one below it on the stack.

```
glMatrixMode(GL_MODELVIEW);
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
translate(0, 10);
drawTorso();
            pushmatrix();// push a copy of transform onto stack
              translate(0, 5); multiply onto current transform
              rotate(rightShoulderRotation);  multiply onto current transform
              drawHead();
            popmatrix();pop current transform off stack
            pushmatrix();
             translate(-2, 3);
             rotate(rightShoulderRotation);
             drawUpperArm();
            popmatrix();
```

Exercise:

1) Draw a skeleton as mentioned in Figure 1 using transformation function and hierarchy representation.

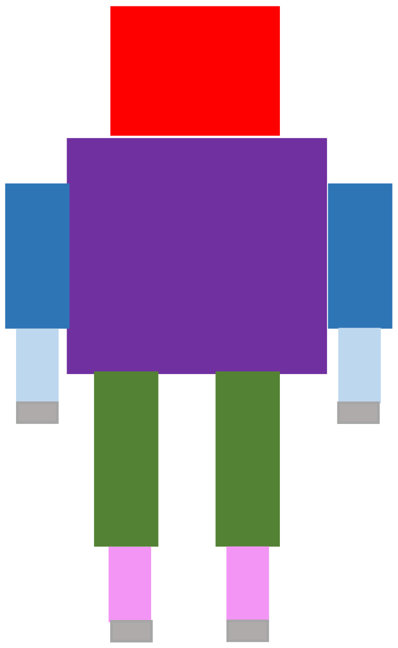

Figure 1