Geometric Transformations in OpenGL

- **Transformation Matrices**

  - OpenGL maintains *three* 4 x 4 transformation matrices which are used to transform any given vertex to its screen pixel coordinates

    * *Modelview* matrix (2D or 3D geometric transformations)

    * *Projection* matrix (projection transformation)

    * *Viewport* matrix (window-to-viewport transformation)

  - The product of these matrices is the *Current Transformation Matrix (CTM)*

  - The CTM is applied to every primitive to generate its screen coordinates

- **Selecting the appropriate matrix**

  - Use *glMatrixMode(GL_MODELVIEW)* to select the *Modelview* matrix

  - Use *glMatrixMode(GL_PROJECTION)* to select the *Projection* matrix

  - The *Viewport* matrix is not directly accessible (only using *glViewport()*)

  - The current matrix mode remains the same until the next *glMatrixMode()* call !!

- **Modifying the contents of a matrix**

  - You can reset it to some matrix

    * *glLoadIdentity()* (e.g., C = I)

    * *glLoadMatrixf(pointer_to_matrix)* (e.g., C = W)
    (Important: the matrix should be stored in column-major order !!)

  - You can *postmultiply* it by a matrix

    * *glMultMatrixf(pointer_to_matrix)* (e.g., C = C W)
    (Important: the matrix should be stored in column-major order !!)

- **The Modelview matrix**

  - It can be modified by using *glLoadIdentity()*, *glLoadMatrixf()* or:

    * *glRotatef(angle, vx, vy, vz)* (e.g,. M = M R)
    (rotation about the axis specified by the vector (vx, vy, vz))

    * *glTranslate(dx, dy, dz)* (e.g,. M = M T)

    * *glScalef(sx, sy, sz)* (e.g,. M = M S)

  - Example: rotation about the axis defined by (1,2,3) and (4,5,6)

    M = T(1.0, 2.0. 3.0) R(45.0, 3.0, 3.0, 3.0) T(-1.0, -2.0, -3.0)

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity()    /* M = I */
glTranslatef(1.0, 2.0, 3.0); /* M = M  T(1.0, 2.0, 3.0)  */
glRotatef(45.0, 3.0, 4.0, 5.0); /* M = M R(45.0, 3.0, 4.0, 5.0) */
glTranslatef(-1.0, -2.0, -3.0); /* M = M  T(-1.0, -2.0, -3.0)  */
```

- **Transforming objects**

  - *Important*: First you define the transformation, then you draw the objects !!

  - Example

    Initialize Modelview matrix (M = I)
    Draw object P1
    Apply transformation T1 (M = M T1)
    Draw object P2
    Apply transformation T2 (M = M T2)
    Draw object P3 ...

  - Example

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(xangle, 1.0, 0.0, 0.0);
glRotatef(yangle, 0.0, 1.0, 0.0);
glRotatef(zangle, 0.0, 0.0, 1.0);
glutWireCube(1.0);
```

- **Matrix stack**

  - The *Modelview* and *Projection* matrices are treated internally as stacks

  - The top element of the stack is the currently "active" matrix

  - The stacks are very useful for rendering *hierarchies* of objets

  - Example

```
glMatrixMode(GL_MODELVIEW);
glLoadIdenity();
glutWireCube(1.0);
glTranslatef(0.0, 0.5, 0.0);
glRotatef(-90.0, 1.0, 0.0, 0.0);
glutWireSphere(0.5, 15, 15);
glRotatef(90.0, 1.0, 0.0, 0.0);
glTranslatef(0.0, 2.5, 0.0);
glutWireCube(1.0);
```

Rewrite the above example as follows:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdenity();
glutWireCube(1.0);
glTranslatef(0.0, 0.5, 0.0);
glPushMatrix();
glRotatef(-90.0, 1.0, 0.0, 0.0);
glutWireSphere(0.5, 15, 15);
glPopMatrix();
glTranslatef(0.0, 2.5, 0.0);
glutWireCube(1.0);
```