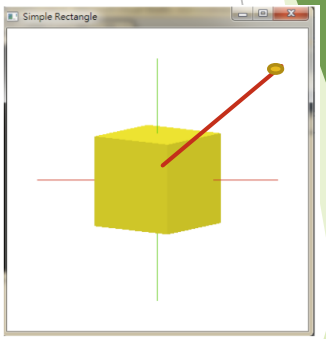
Lab 03
Transformation Matrix

Lab 03 Goal

1. Rotate along x, y, z respectively.
   - use your own key setting
2. Translate along x, y, z respectively
   - use your own key setting
3. Reset to origin
   - use your own key setting

- Write comments in your code about your key setting
- Do not use glRotate, glTranslate in your code
- Turn in your code



Note: The rotational matrix is provided in this pdf (for arbitrary rotation), use it for this Lab assignment.

Transformation Matrix

- All modeling transformations are represented as 4x4 matrices

- Identity matrix

```
GLfloat rotMatrix[] = {
        1.0, 0.0, 0.0, 0.0,
        0.0, 1.0, 0.0, 0.0,
        0.0, 0.0, 1.0, 0.0,
        0.0, 0.0, 0.0, 1.0   };
```

```
rotMatrix[0] = 1;  rotMatrix[4] = 0;   rotMatrix[8] = 0;  rotMatrix[12] = 0;
rotMatrix[1] = 0;  rotMatrix[5] = 1;   rotMatrix[9] = 0;  rotMatrix[13] = 0;
rotMatrix[2] = 0;  rotMatrix[6] = 0;  rotMatrix[10] = 1;  rotMatrix[14] = 0;
rotMatrix[3] = 0;  rotMatrix[7] = 0;  rotMatrix[11] = 0;  rotMatrix[15] = 1;
```

Translation Revisited

$$T(t_x, t_y, t_z)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \left[\begin{array}{cccc|c} 1 & 0 & 0 & t_x & x \\ 0 & 1 & 0 & t_y & y \\ 0 & 0 & 1 & t_z & z \\ \hline 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Rotation Revisited

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The call glRotate(Θ, 1, 0, 0) generates R_x as follows:

$$R_x(\theta)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation Revisited

The call glRotate(Θ, 0, 0, 1) generates R_z as follows:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Scaling Revisited

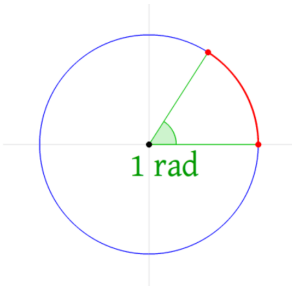
$$S(s_x, s_y, s_z)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \left[\begin{array}{cccc|c} s_x & 0 & 0 & 0 & x \\ 0 & s_y & 0 & 0 & y \\ 0 & 0 & s_z & 0 & z \\ \hline 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

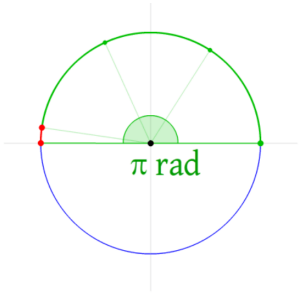
Degree to radians conversion

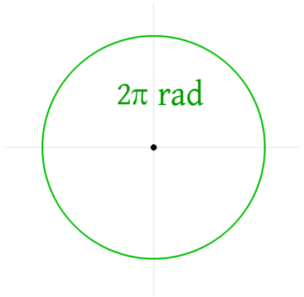
```
#define PI 3.14159265

int main ()
{
    double  degree, result;
    degree = 60.0;
    result = cos ( degree * PI / 180.0 );   //   = 2PI /360
    printf ("The cosine of %f degrees is %f.\n", degree, result );
    return 0;

}
```







360 degree = 2PI

radian: the length of a corresponding arc of a unit circle

glMultiMatrix

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity;
glMultMatrixf(rotMatrix);
glMultMatrixf(translateMatrix);

//draw_the_object
glutSolidCube(6);
```

```
GLfloat rotMatrix[] = {
        1.0, 0.0, 0.0, 0.0,
        0.0, 1.0, 0.0, 0.0,
        0.0, 0.0, 1.0, 0.0,
        0.0, 0.0, 0.0, 1.0   };
```

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(angle, 1,0,0);
glTranslatef(tx,ty,tz);
//draw the object
glutSolidCube(6);
```