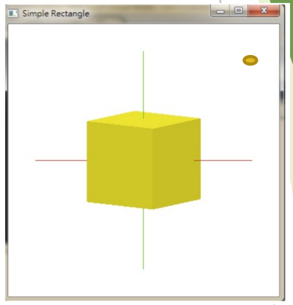
Lab 03
Transformation Matrix

Lab 04 Goal – Transformation matrix

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. Scale along x, y, z respectively
    - use your own key setting
4. Reset to origin
    - use your own key setting

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



Note: We will leave the implementation of arbitrary-axis rotation to next week (some detail haven't been discussed in class yet).

Transformation Matrix

- All modeling transformations are represented as 4x4 matrices

```
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   };
```

- Identity matrix

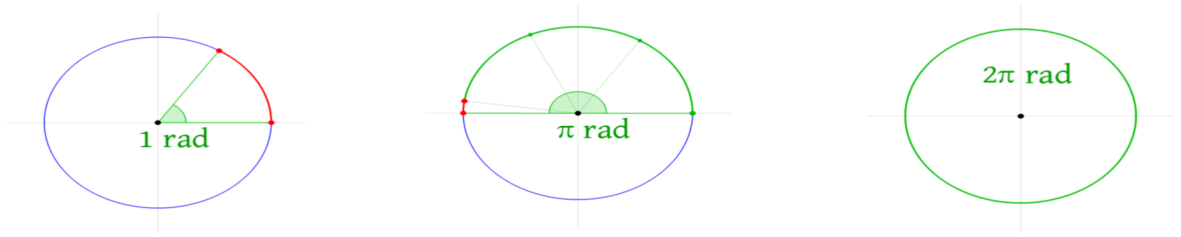
```
rotMatrix[0] = 1;   rotMatrix[4] = 0;    rotMatrix[8] = 0;   rotMatrix[12] = 0;
rotMatrix[1] = 0;   rotMatrix[5] = 1;    rotMatrix[9] = 0;   rotMatrix[13] = 0;
rotMatrix[2] = 0;   rotMatrix[6] = 0;   rotMatrix[10] = 1;   rotMatrix[14] = 0;
rotMatrix[3] = 0;   rotMatrix[7] = 0;   rotMatrix[11] = 0;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

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);
```

```
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 glRotatef(angle, 1,0,0);
 glTranslatef(tx,ty,tz);
//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   };
```