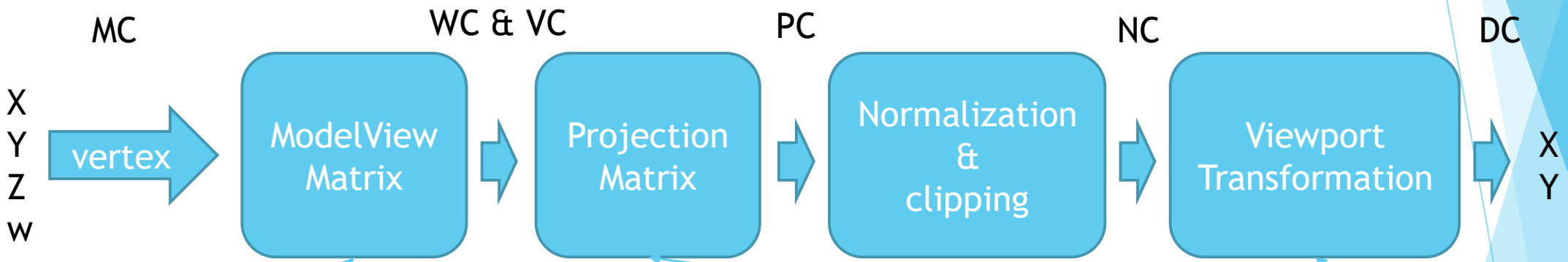
OpenGL – Viewing and Transformation

Overview



TODO:

1. Switch matrix mode to GL_MODELVIEW and call glLoadIdentity().
2. Call gluLookAt().
3. Your own modeling transformations.

TODO:

1. Switch matrix mode to GL_PROJECTION and call glLoadIdentity().
2. Call gluPerspective() if you want perspective projection.
3. Call gluOrtho2D() if you want orthogonal projection.

TODO:

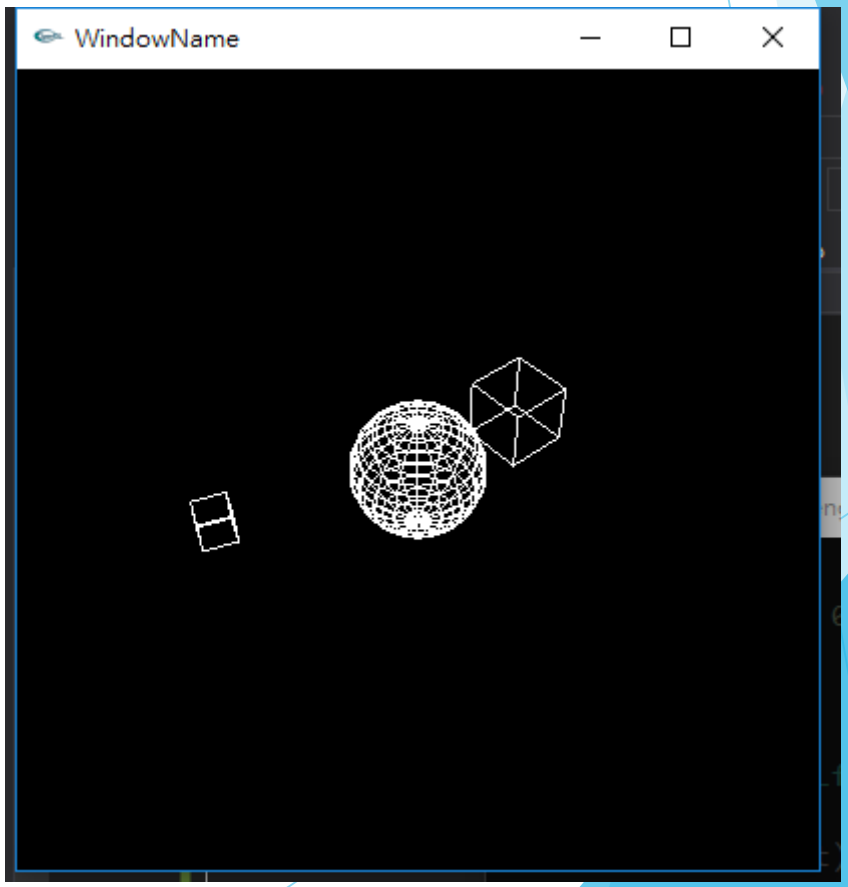
1. Call glViewPort()

Example

```
16  int degree = 0;
```

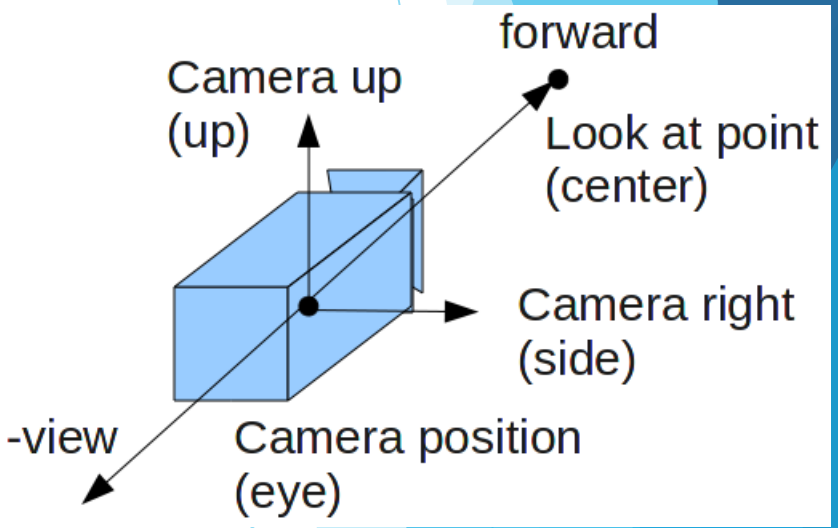
```
39  void display()
40  {
41      //ModelView Matrix
42      glMatrixMode(GL_MODELVIEW);
43      glLoadIdentity();
44      gluLookAt(0.0f, 10.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
45      //Projection Matrix
46      glMatrixMode(GL_PROJECTION);
47      glLoadIdentity();
48      gluPerspective(45, width / (GLfloat)height, 0.1, 1000);
49      //Viewport Matrix
50      glViewport(0, 0, width, height);
51
52      //
53      glMatrixMode(GL_MODELVIEW);
54      glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
55      glClear(GL_COLOR_BUFFER_BIT);
56
57      //sum
58      glutWireSphere(1, 18, 18);
59      //plant1
60      glPushMatrix();
61      glRotatef(degree, 0.0f, 1.0f, 0.0f);
62      glTranslatef(3.0, 0.0, 0.0);
63      glScalef(0.5, 0.5, 0.5);
64      glutWireCube(1);
65      glPopMatrix();
66      //plant2
67      glPushMatrix();
68      glRotatef(degree * 2, 0.0f, 1.0f, 0.0f);
69      glTranslatef(2.0, 0.0, 0.0);
70      glutWireCube(1);
71      glPopMatrix();
72
73      glutSwapBuffers();
74  }
```

```
98  void idle() {
99      Sleep(20);  //about 50 fps
100     degree = (degree + 1) % 360;
101     glutPostRedisplay();
102 }
```



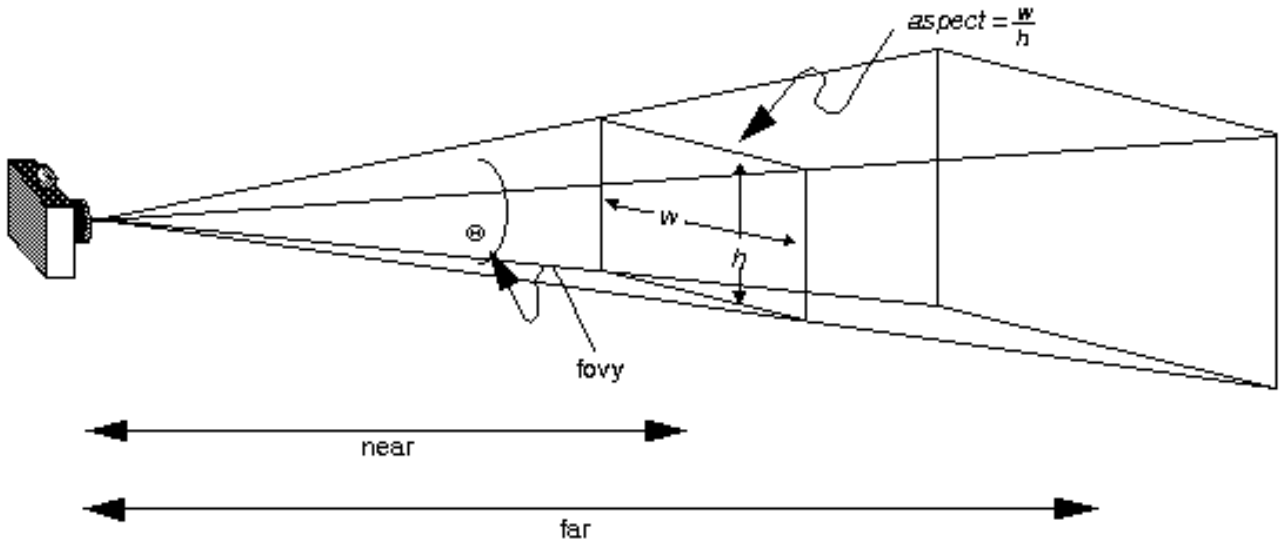
ModelView matrix

- void glMatrixMode(GLenum mode);
  - Switch between three modes
  - GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE
  - Each matrix mode has its own matrix stack
- void glLoadIdentity(void);
  - Replace the current matrix with the identity matrix
- void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdoubpe upz);
  - eyex, eyey, eyez: is where the camera is positioned
  - centerx, centery, centerz: is where the camera looks at
  - upx, upy, upz: is the up-vector of the camera
- Your own modeling transformations



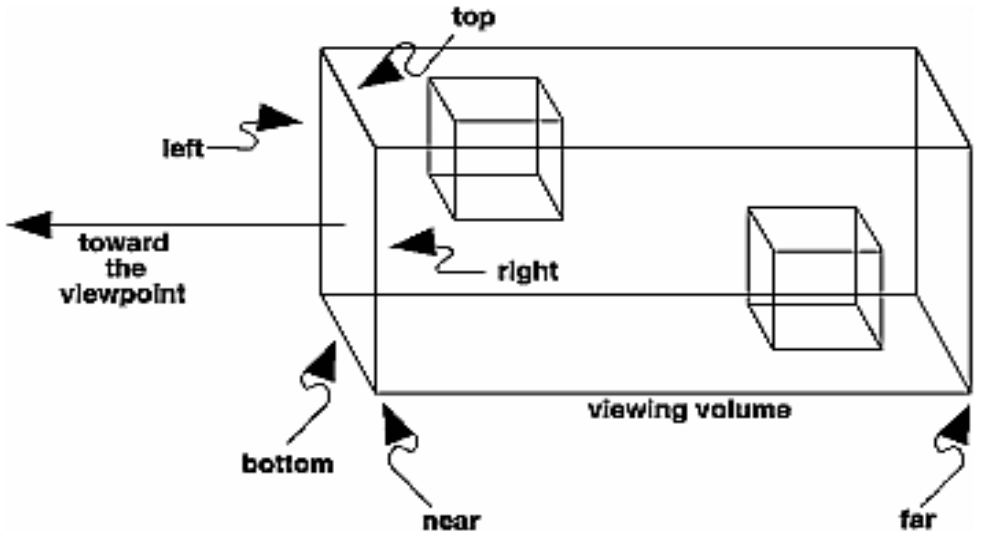
Projection matrix

- glMatrixMode(GL_PROJECTION);
- glLoadIdentity();
- void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far);
  - fovy: specifies the field of view angle, in degrees, in the y direction.
  - aspect: specifies the aspect ratio that determines the field of view in the x direction.
  - zNear: Specifies the distance from the viewer to the near clipping plane (always positive).
  - zFar: Specifies the distance from the viewer to the far clipping plane (always positive).



Projection matrix

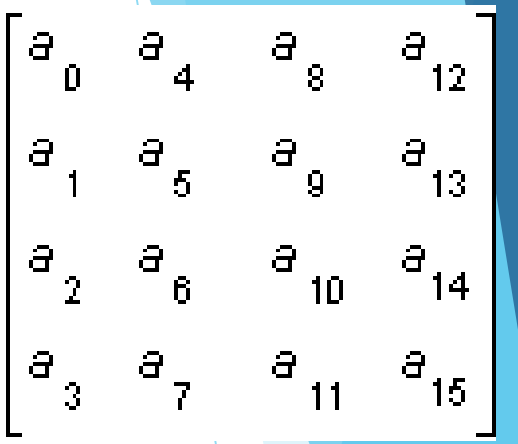
- Orthographic projection
- void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);
  - left, right: specify the coordinates for the left and right vertical clipping planes.
  - bottom, top: specify the coordinates for the bottom and top horizontal clipping planes.
  - near, far: specify the distances to the nearer and farther depth clipping planes. (can be negative)
- void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);
  - Equal to calling glOrtho with near = 1 and far = 1



Viewport transformation

- void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
  - Transform the final image into some region of the window
  - x, y: the lower-left corner of the viewport rectangle, in pixels. The default is (0, 0)
  - width, height: the width and height of the viewport. The default is set to the dimensions of that window

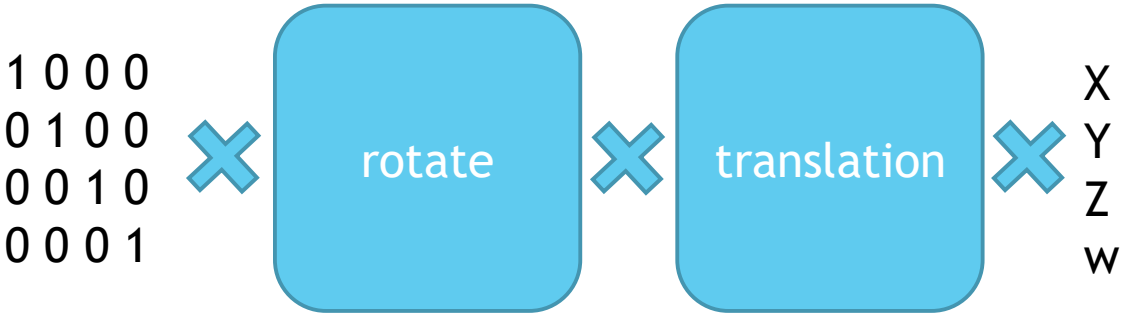
OpenGL matrix



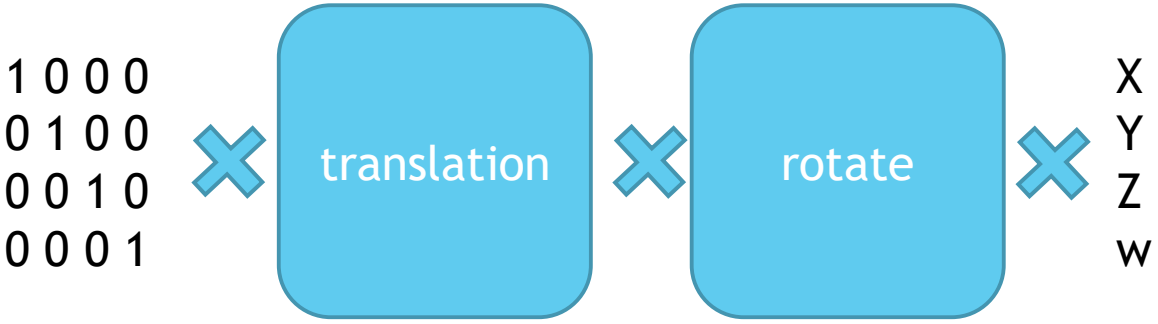
- void glMatrixMode(GLenum mode);
- void glLoadIdentity(void);
- void glLoadMatrix{f,d}(const TYPE* m);
  - replace the current matrix with the specified matrix
  - m: specifies a pointer to 16 consecutive values, which are used as the elements of a 4×4 column-major matrix
- void glMultMatrix{f,d}(const TYPE* m);
  - multiply the current matrix with the specified matrix
  - M: Points to 16 consecutive values that are used as the elements of a 4 × 4 column-major matrix.
- void glPushMatrix();
  - Push current matrix into matrix stack
- void glPopMatrix();
  - Pop matrix from matrix stack
- These stack operations of matrix is very useful for constructing a hierarchical structure

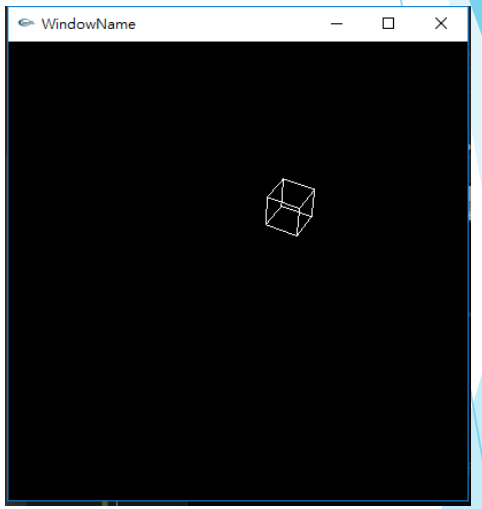
OpenGL matrix operation order

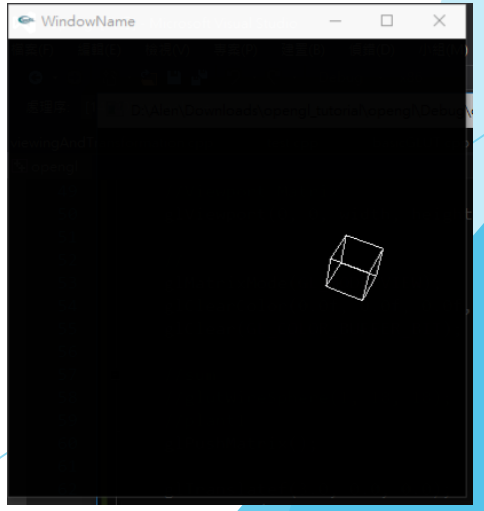
- gluLookAt(0.0f, 10.0f, 10.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
- glRotatef(60, 0.0f, 1.0f, 0.0f);
- glTranslatef(3.0, 0.0, 0.0);



- glTranslatef(3.0, 0.0, 0.0);
- glRotatef(60, 0.0f, 1.0f, 0.0f);



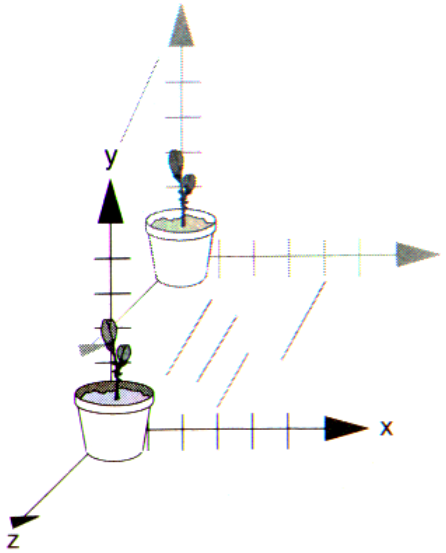




Modeling transformation

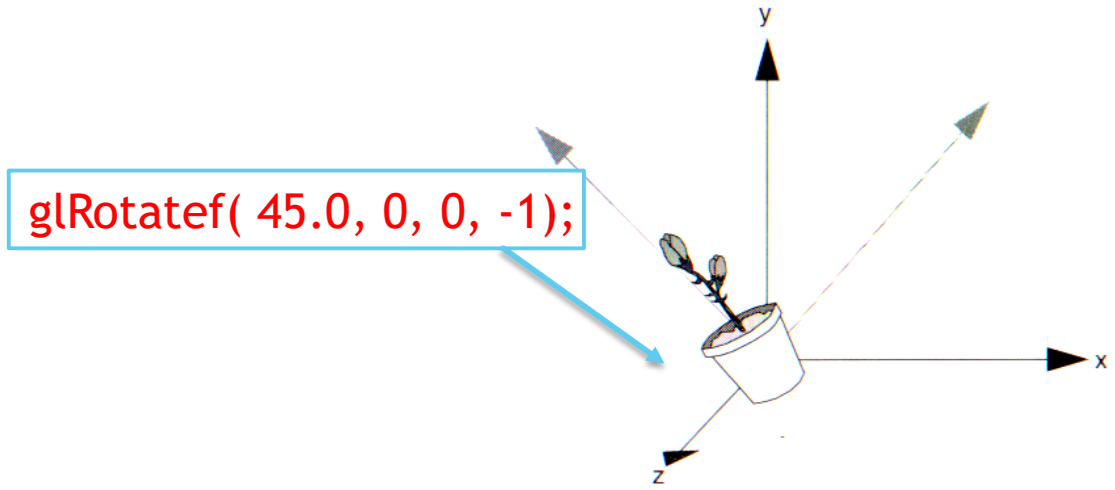
- void glTranslate{fd}(TYPE x, TYPE y, TYPE z);
  - TYPE: GLfloat or GLdouble
  - x, y, z: specify the x, y, and z coordinates of a translation vector
  - Multiplies current matrix by a matrix that moves an object by (x, y, z)

```
glTranslatef( 0, 0, -1 );
```



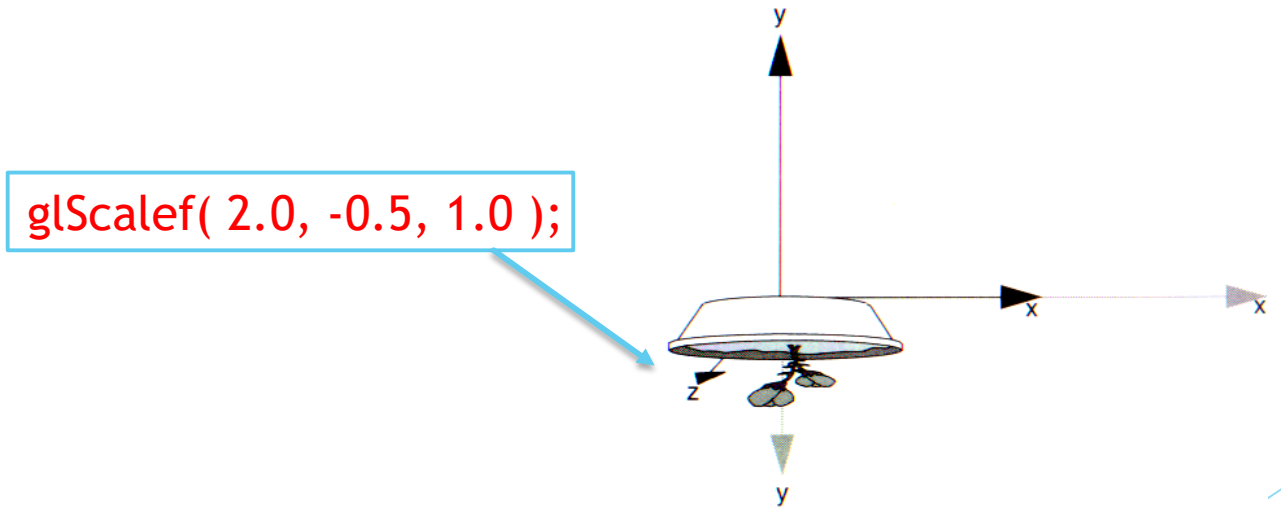
Modeling transformation

- void glRotate{fd}(TYPE angle, TYPE x, TYPE y, TYPE z);
  - TYPE: GLfloat or GLdouble
  - Rotation follows the right-hand rule.
  - Multiplies current matrix by a matrix that rotates an object in a counterclockwise direction about the ray from origin to (x, y, z) with angle as the degrees



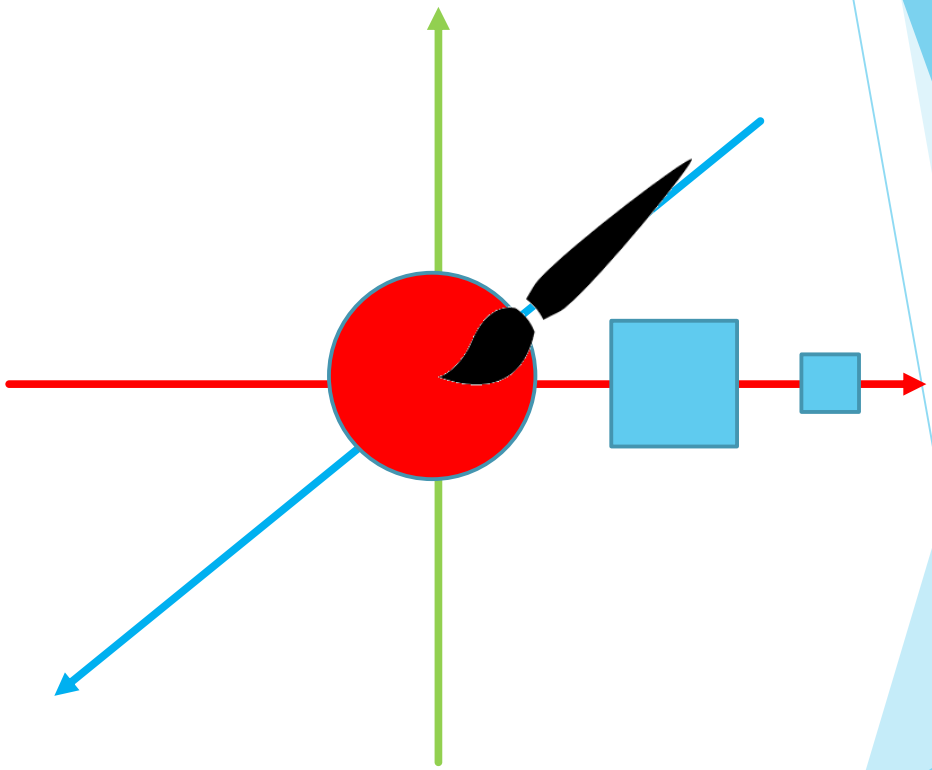
Modeling transformation

- void glScale{fd}(TYPE x, TYPE y, TYPE z);
  - TYPE: GLfloat or GLdouble
  - Multiplies current matrix by a matrix that scales an object along axes.
  - X, y , z: specify scale factors along the *x*, *y*, and *z* axes, respectively.



Push & pop matrix

```
57      //sum
58      glutWireSphere(1, 18, 18);
59      //plant1
60      glPushMatrix();
61      glRotatef(degree, 0.0f, 1.0f, 0.0f);
62      glTranslatef(3.0, 0.0, 0.0);
63      glScalef(0.5, 0.5, 0.5);
64      glutWireCube(1);
65      glPopMatrix();
66      //plant2
67      glPushMatrix();
68      glRotatef(degree * 2, 0.0f, 1.0f, 0.0f);
69      glTranslatef(2.0, 0.0, 0.0);
70      glutWireCube(1);
71      glPopMatrix();
72
```



Push & pop matrix

- If not use push and pop, it will be a trouble as your program getting more complex

```
57      //sum
58      glutWireSphere(1, 18, 18);
59      //plant1
60      glRotatef(degree, 0.0f, 1.0f, 0.0f);
61      glTranslatef(3.0, 0.0, 0.0);
62      glScalef(0.5, 0.5, 0.5);
63      glutWireCube(1);
64      glScalef(2, 2, 2);
65      glTranslatef(-3.0, 0.0, 0.0);
66      glRotatef(-degree, 0.0f, 1.0f, 0.0f);
67      //plant2
68      glRotatef(degree * 2, 0.0f, 1.0f, 0.0f);
69      glTranslatef(2.0, 0.0, 0.0);
70      glutWireCube(1);
71      glTranslatef(-2.0, 0.0, 0.0);
72      glRotatef(-degree * 2, 0.0f, 1.0f, 0.0f);
```