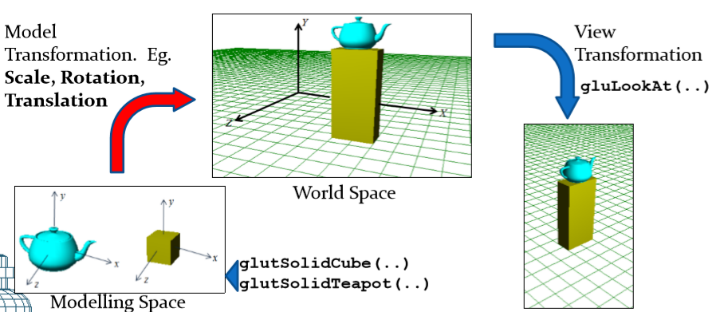
**Transformations**

**Model-View Transformation**

Transformation = changes the vertex/vector coordinates of the object

* Types:
  + Transformations of objects within the same frame: E.g. Translations, rotations and scale transformations. These transformations are called Model Transformations.
  + Projection transformations. These are based on the camera’s frustum parameters.

Objects create their own local coordinate space then are transformed into world space.

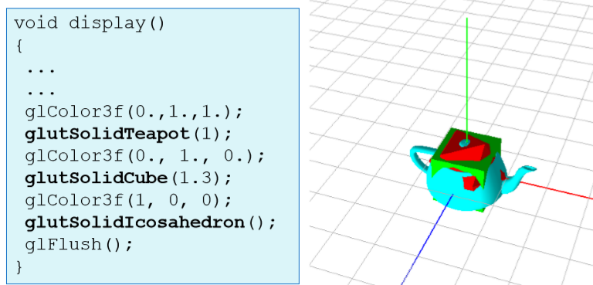
**Transformations in OpenGL**

Translations: **glTranslatef**(a, b, c);

Rotations: **glRotatef**(angle, l, m, n);

Scale Transformations: **glScalef**(sx, sy, sz);

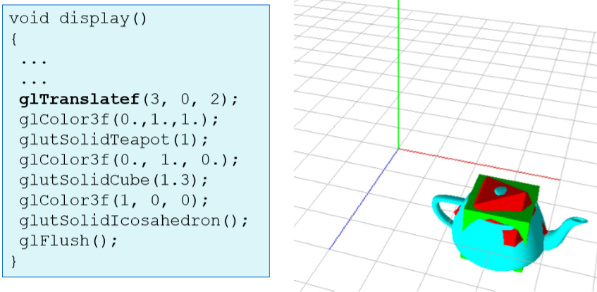
Generalized transformation: **glMultMatrixf**(mat);

Model transformations form part of the model-view matrix. **glMatrixMode**(GL\_MODELVIEW); **glLoadIdentity**();

**glTranslatef** not used ->

**Model Creation** and **Translation**

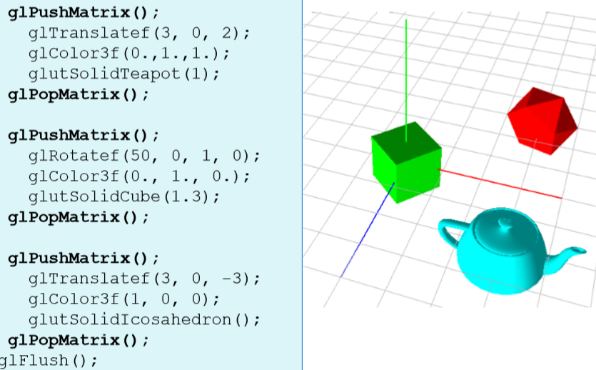
* Function = **glTranslatef**(a, b, c);
* If a model doesn’t have its transformation specified by using **glTranslatef**. Then default = origin **0** in the world space

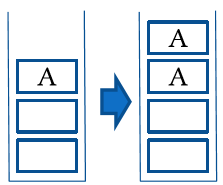


**glTranslatef(3, 0, 2);** include at the top therefore changes the translation.

**glTranslatef** used ->

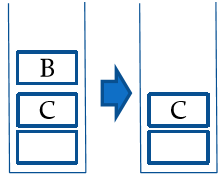
**Matrix Stack**

* shows how we can change the transformations of each object.

**glPushMatrix**();

* pushes matrix to the bottom of the stack

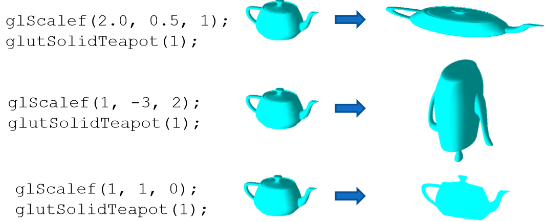
… code for object …

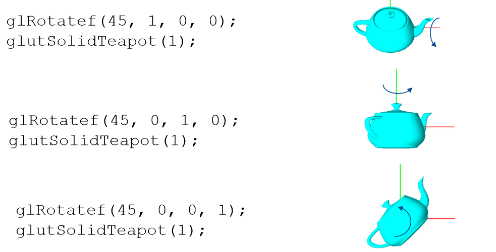


**glPopMatrix**();

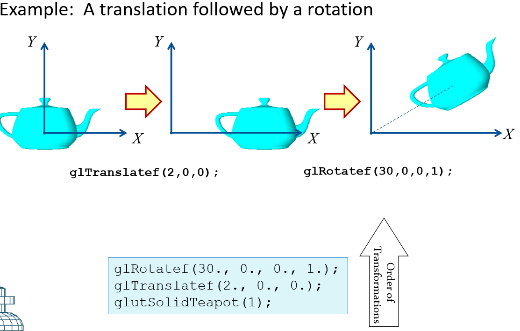
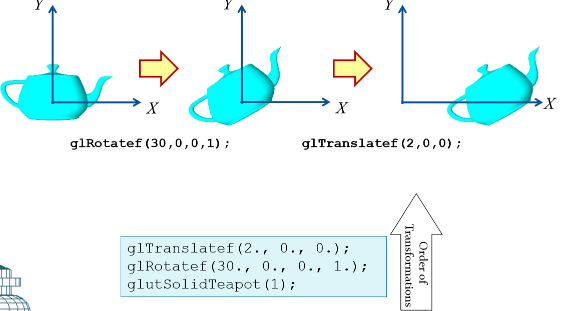
* pops the matrix at the top of the stack

**Scaling**

* Function = **glScalef** (a, b, c);
* Negative scale factor = reflection
* Zero scale factor = projection

**Rotations**

* Function = **glRotatef**(theta, l, m, n)
* Positive angle = anti-clockwise rotation
* Rotate positive = anti-clockwise

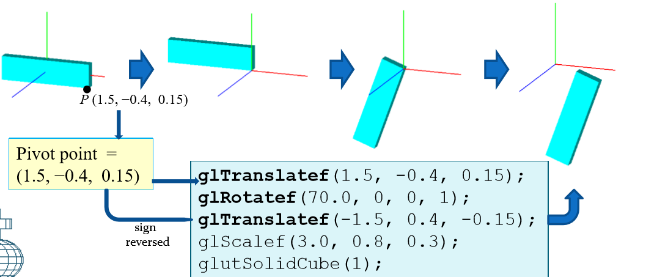
**Composite Transformations**

Example 1: A rotation followed by a translation

Example 2: A translation followed by a rotation (**This is the one we use**)

Example 1

Example 2

**Rotations about a pivot point**

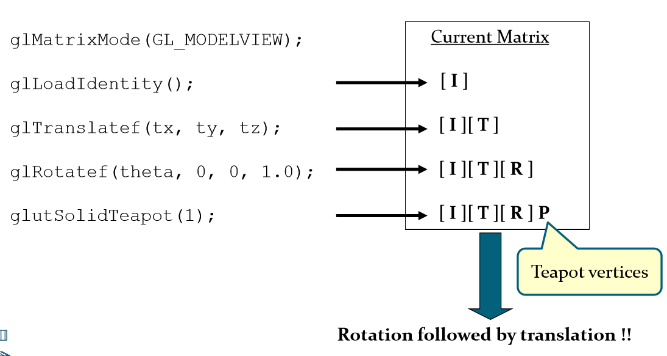
Question: How can we perform rotations about an arbitrary pivot point?

1. Reverse translate the object **glTranslatef**(-px, -py, -pz)

**glTranslatef**(-1.5, 0.4, -0.15);

1. Required rotation **glRotatef**( theta, l, m, n)
2. Translate the object **glTranslatef**(px, py, pz) to original position

**glTranslatef**(1.5, -0.4, 0.15);

def

**OpenGL Transformations**

current transformation matrix [P] = OpenGL post- multiplies

**gluLookAt(…)** = generates a transformation matrix