Lecture 5: Viewing

CSE 40166 Computer Graphics (Fall 2010)

Review: from 3D world to 2D pixels

1. Transformations are represented by matrix multiplication.

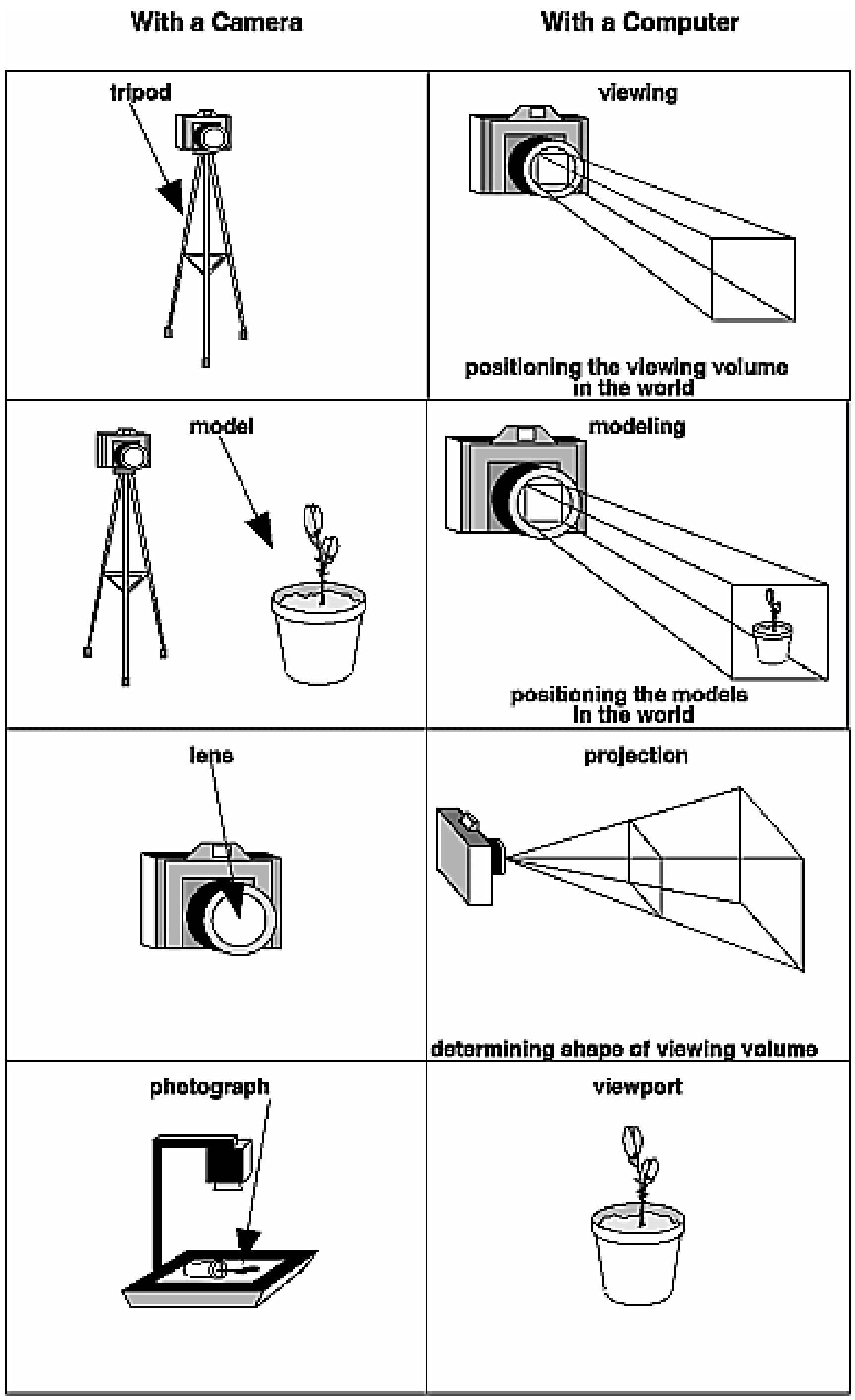
   - **Modeling**
   - **Viewing**
   - **Projection**

2. Clipping volume used to throw out objects.

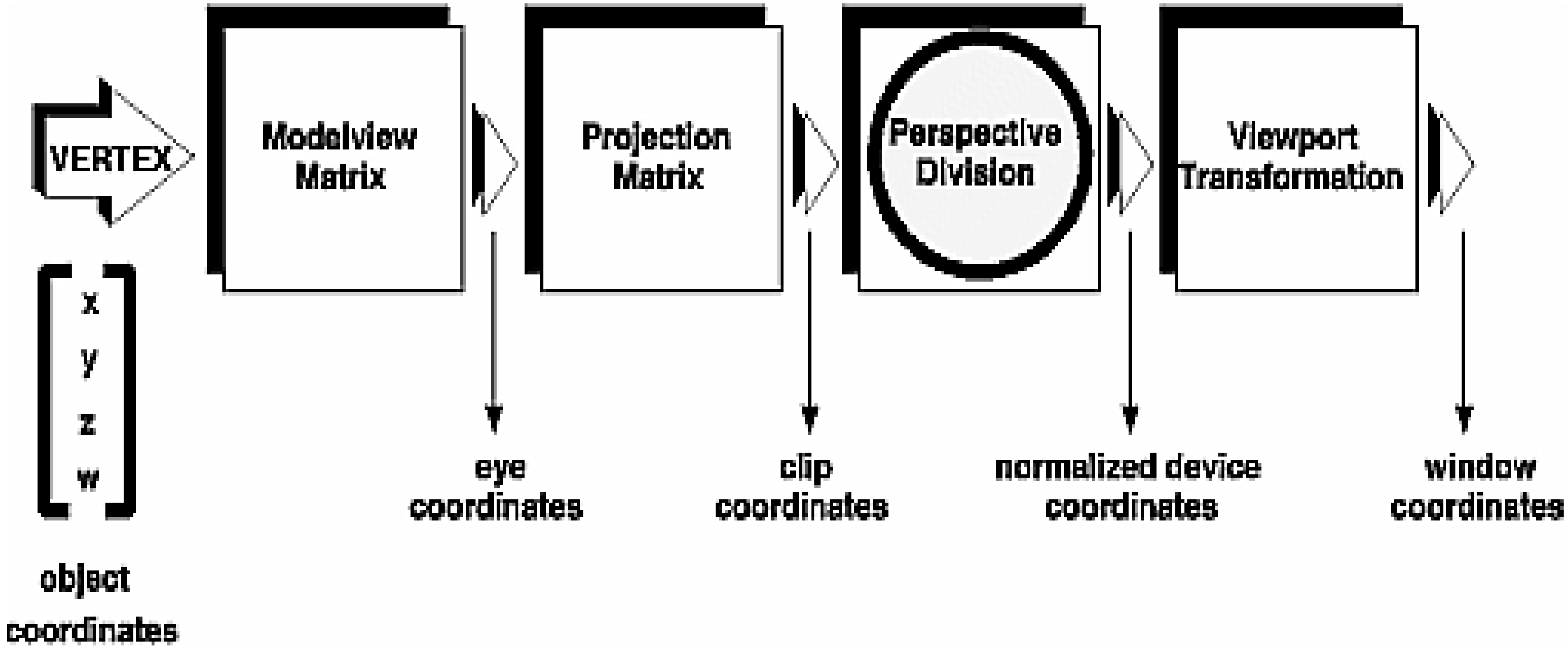
3. Correspondance between transformed coordinates and screen pixels.

Camera Analogy

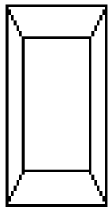
- Setup tripod and point camera at scene (**viewing transformation**)
- Arrange scene to be photograph (**modeling transformation**)
- Choose a camera lens or zoom (**projection transformation**)
- Determine how large final image should be (**viewport transformation**)



Vertex Tranformation Pipeline



Cube Example



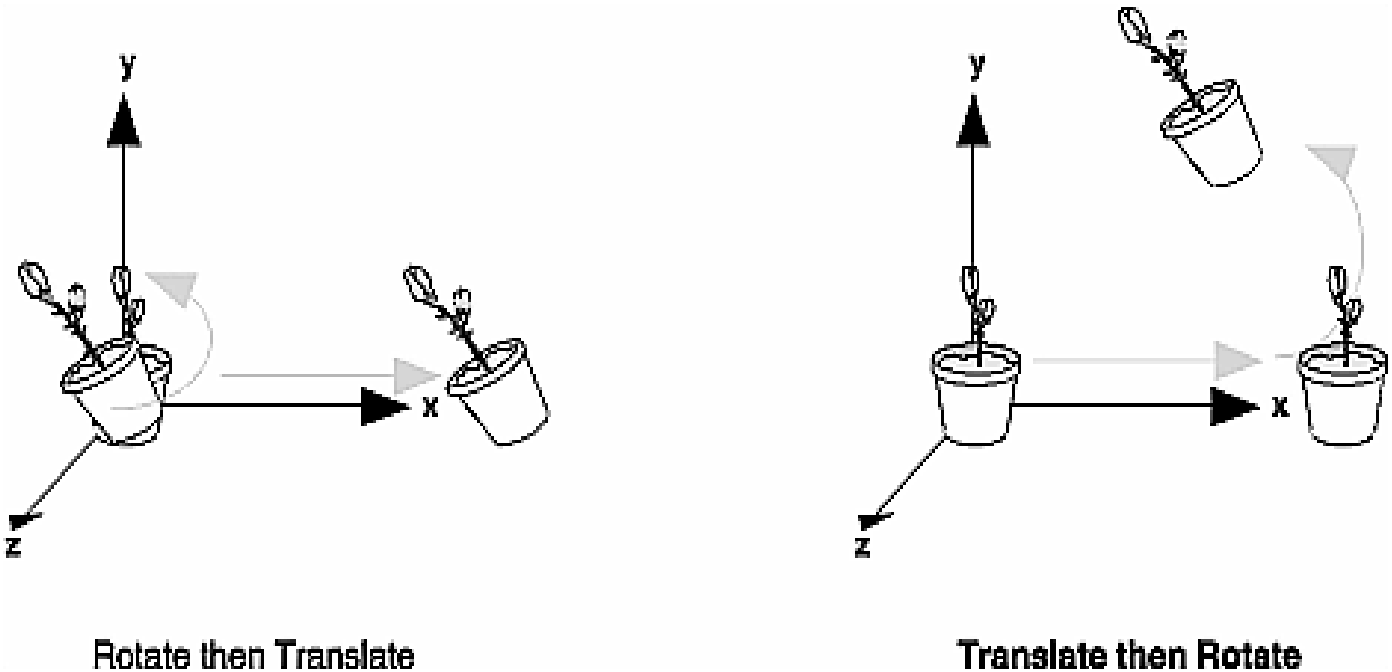
```
void display(void) {
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glLoadIdentity();        /* viewing transformation */
   gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
   glScalef(1.0, 2.0, 1.0); /* modeling transformation */
   glutWireCube (1.0);
   glFlush();
}

void reshape(int w, int h) {
   glViewport(0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
   glMatrixMode (GL_MODELVIEW);
}
```

Viewing and Modeling Transformations

- Viewing and modeling transformations are inextricably related in OpenGL and are in fact combined into a single modelview matrix
  - ***Do you want to move the camera in one direction, or move the object in the opposite direction?***
  - ***Each way of thinking about transformations has advantages and disadvantages, but in some cases one way more naturally matches the effect of the intended transformation.***

Transformation Order Matters




```
glTranslatef()
glRotate()
draw_flower_pot()
```

```
glRotatef()
glTranslatef()
draw_flower_pot()
```

Grand, Fixed Coordinate System

- Matrix multiplications affect the position, orientation, and scaling of your model.

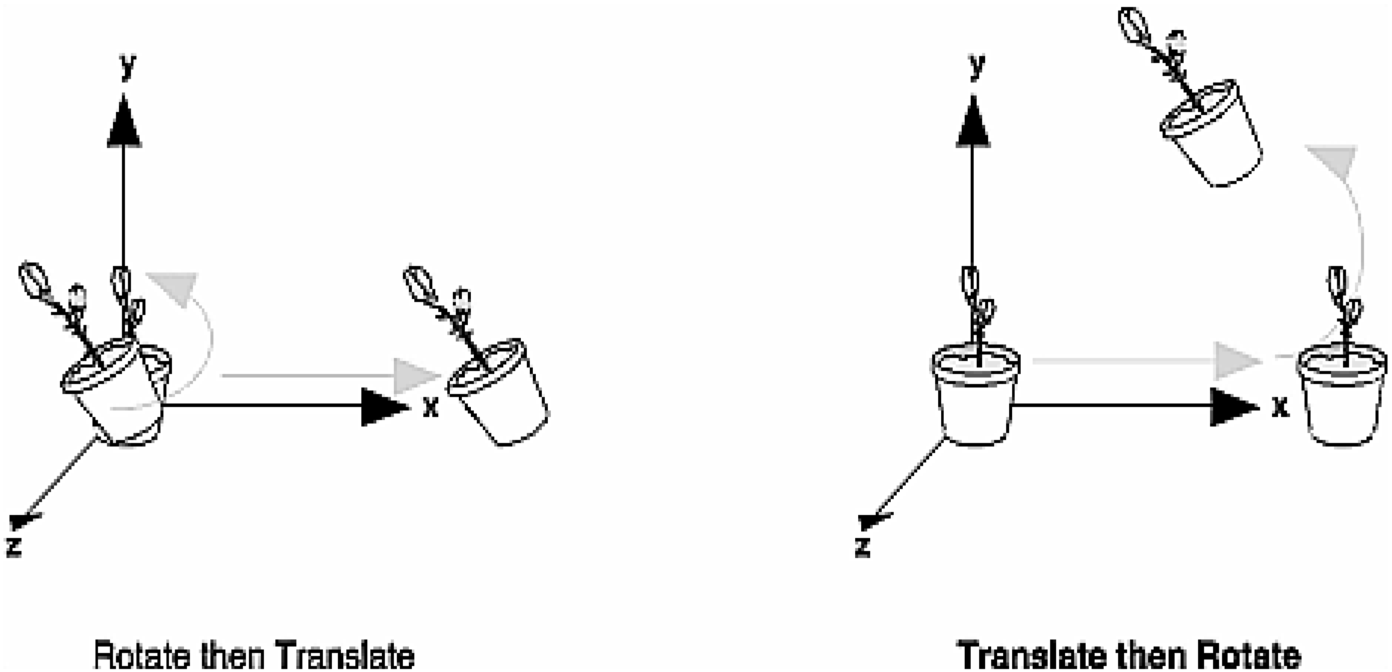
  Transformations move object in fixed global coordinate system.

- Multiplications occur in **opposite order** from how they appear in code.

Moving a Local Coordinate System

- Instead of fixed coordinate system, image a local coordinate system that is tied to the object you are drawing.
- All operations occur relative to this changing coordinate system (and thus multiplications are same order as in the code)
- Extremely useful for hierarchical objections (i.e arms, legs, joints).
- Scaling may be problematic (translations move by a multiple since they are stretch).

Transformation Order Redux

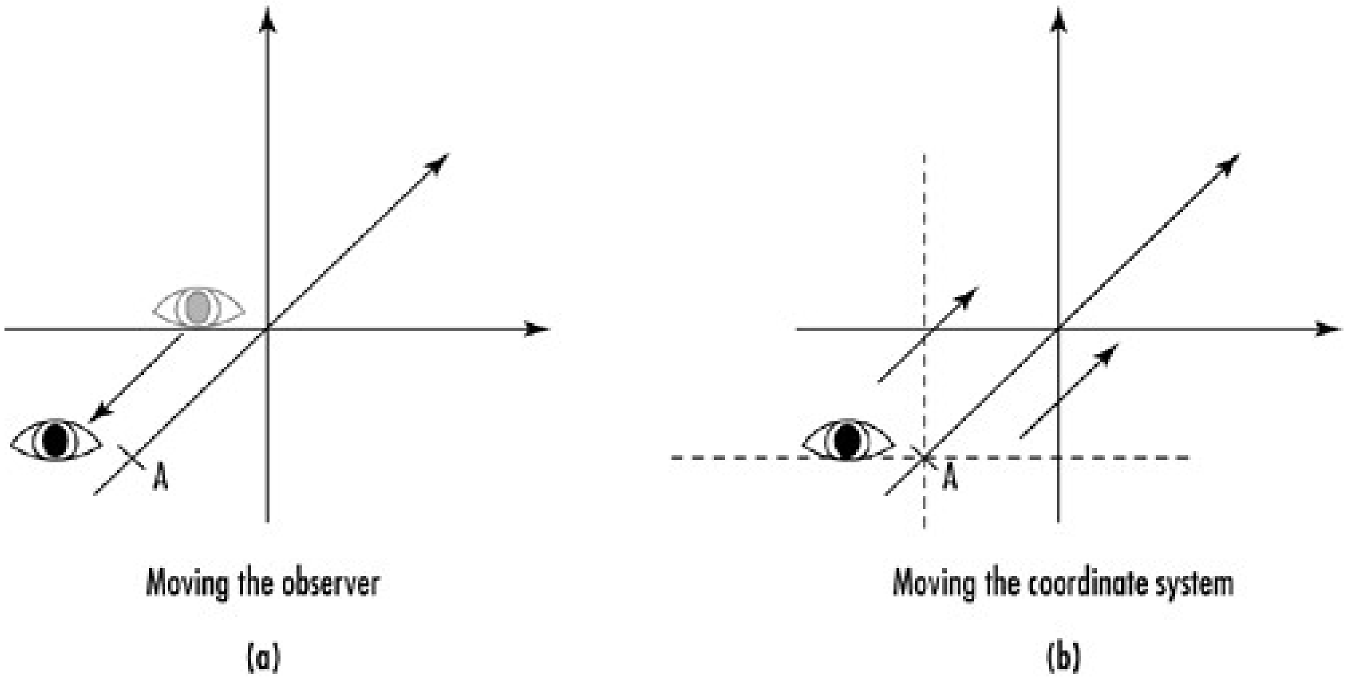



```
glTranslatef()
glRotate()
draw_flower_pot()
```

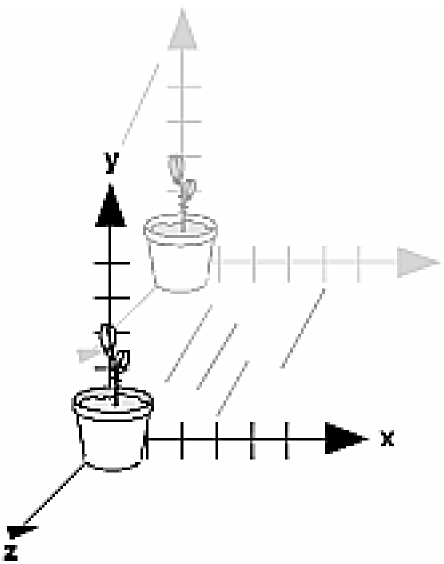
```
glRotatef()
glTranslatef()
draw_flower_pot()
```

Modelview Duality

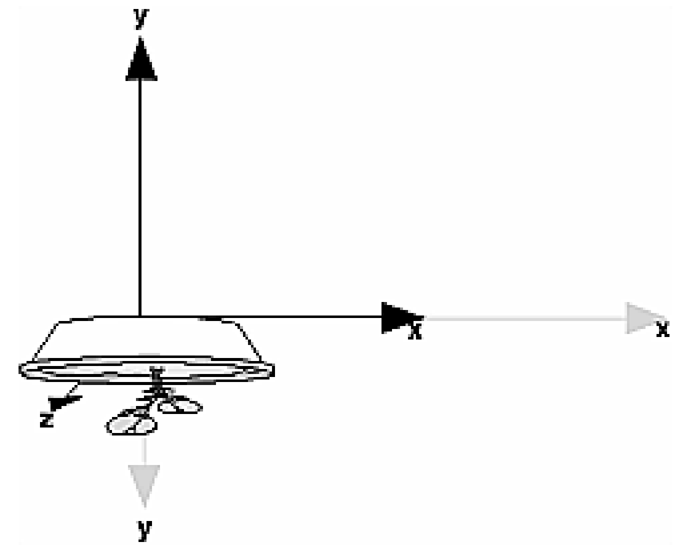
- There is no real difference between moving an object backward and moving the reference system forward.
- Viewing transformation, therefore, is essentially nothing but a modeling transformation that you apply to the viewer before drawing objects



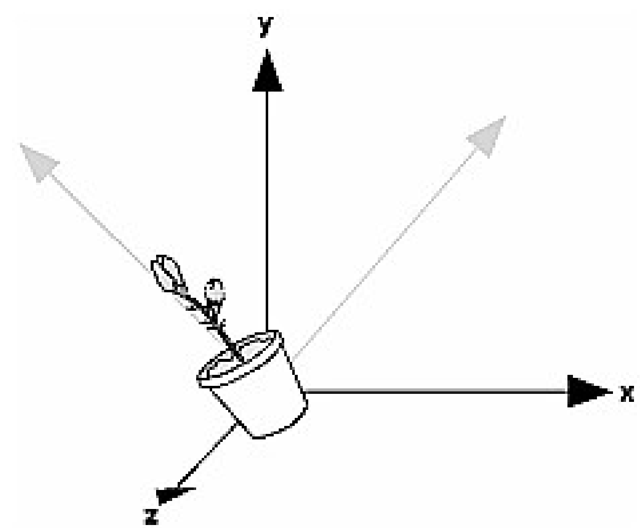
Modeling Transformations



glTranslatef



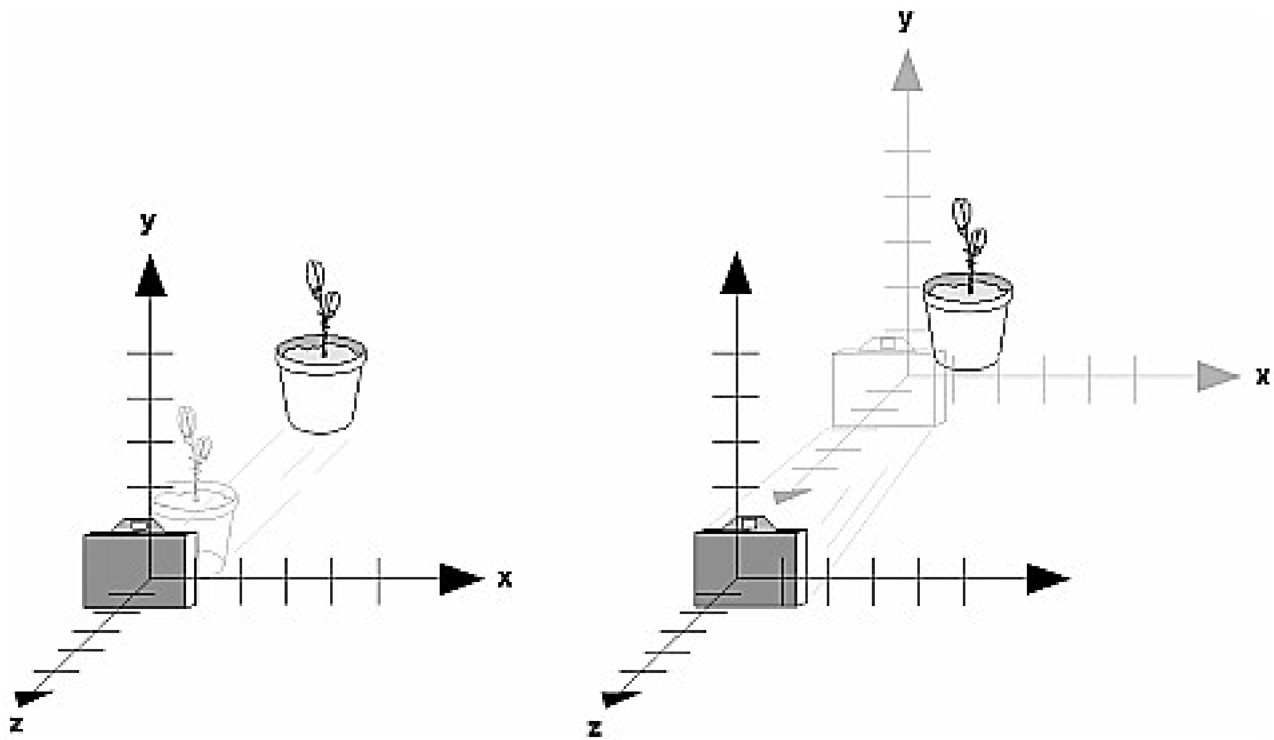
glScalef



glRotatef

Viewing Transformations

- Changes position and orientation of viewport.
- Camera moves in opposite direction as objects.

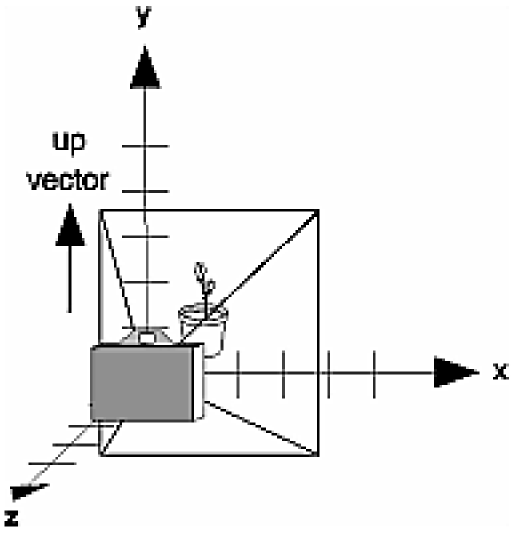


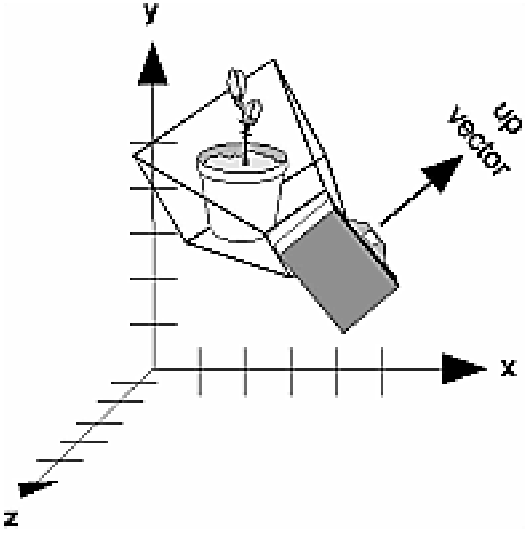
- To set viewing transformation:
  - **glTranslate, glRotate**
  - **GluLookAt**
- Viewing transformations must be called before any modeling transformations are performed.

gluLookAt

- Construct a scene around origion or some convenient location, and then want to look at it from an arbitrary point.

- **gluLookAt**: let's you specify location of viewpoint, a reference point toward which a camera is aimed, and which direction is up.





Projection Transformations

- The purpose of the projection transformation is to define a ***viewing volume***.

- Determine how an object is projected onto the screen.

- Defines which objects or portions of objects are clipped out of final image.

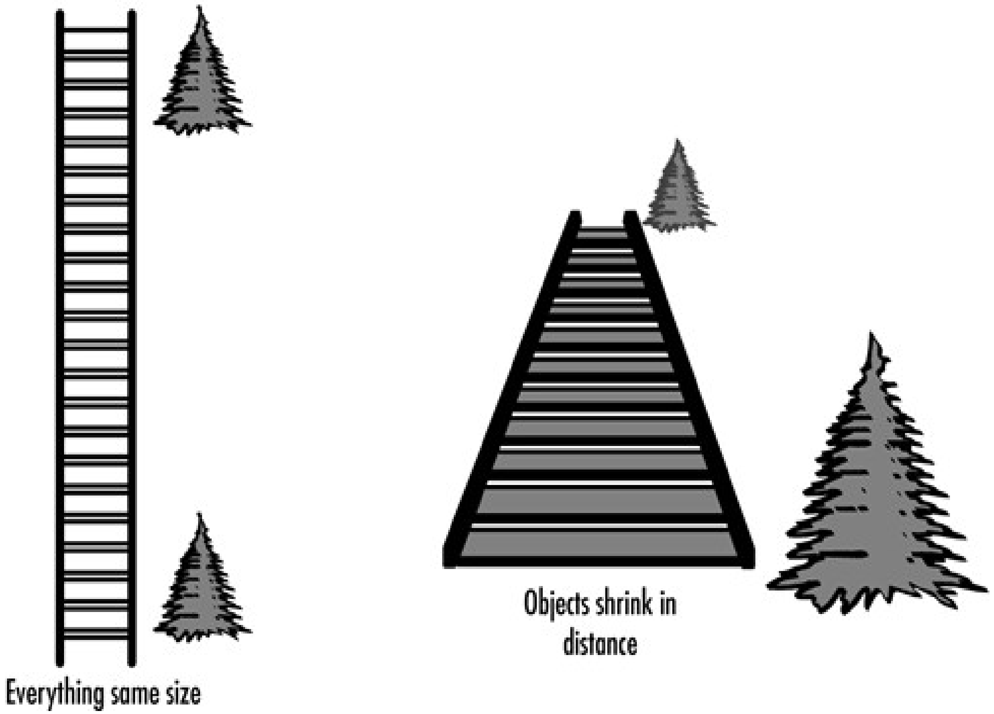
- Viewpoint exists at one end of viewing volume.

- Two main types of projections:
  - **Perspective**
  - **Orthogonal**

Perspective Projection

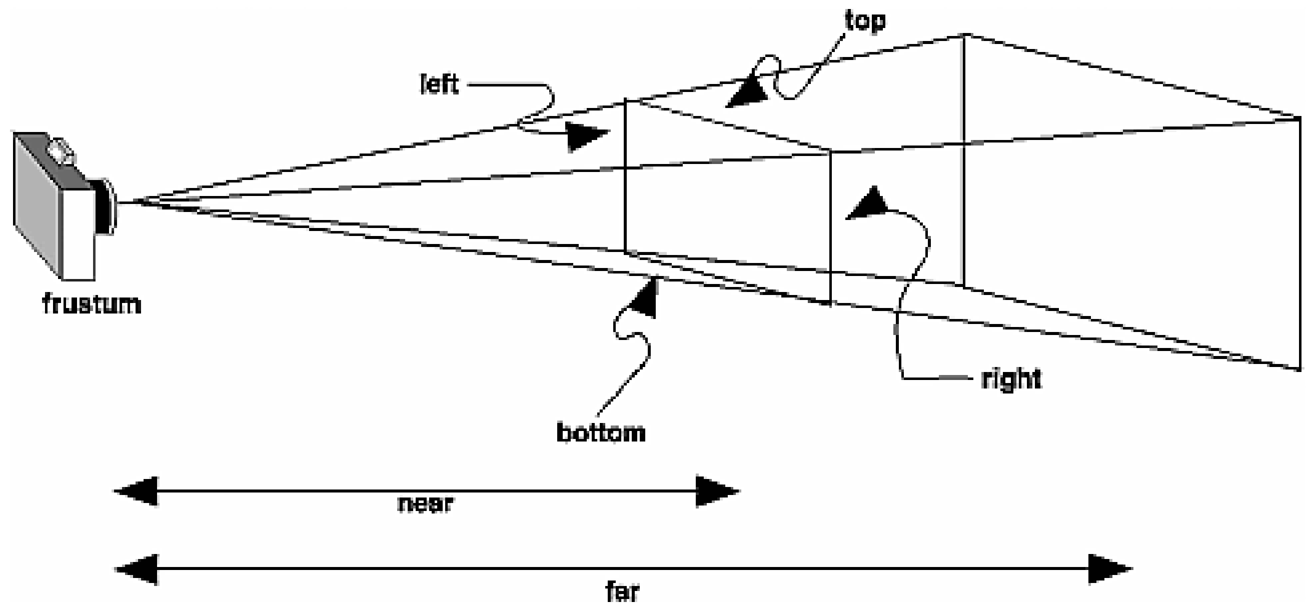
- Major characteristic is **foreshortening**:

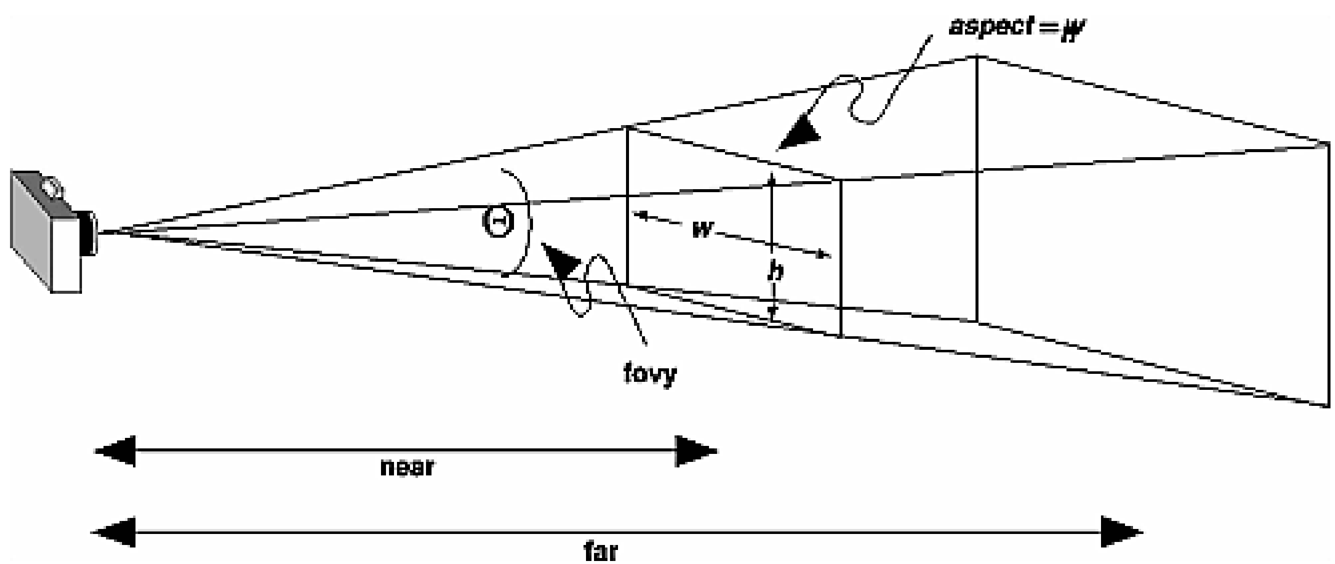
  Farther an object is from the camera, the smaller it appears in the final image.



- Viewing volume is a frustrum of a pyramid.
  - Objects inside volume are projected toward apex of pyramid.
  - Objects closer to viewpoint appear larger.

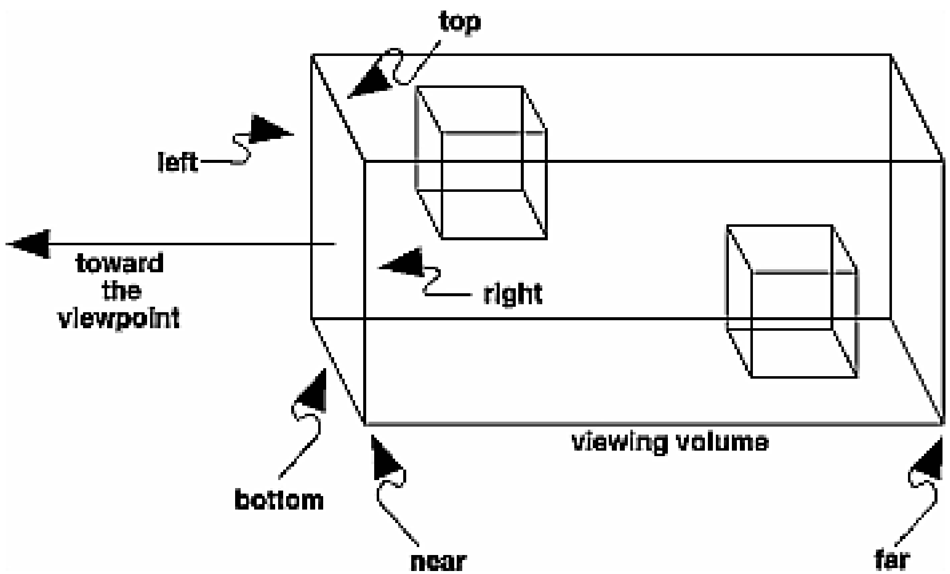
glFrustrum, gluPerspective





Orthographic Projection

- Parallel projection, all the polygons are drawn onscreen with exactly the relative dimensions specified.



- Used often for 2D drawing purposes where you want an exact correspondence between pixels and drawing units.
  - CAD, blueprints, text, on-screen menus

Hidden Surface Removal

- Remove surfaces that should not be visible to viewer.
- OpenGL provides **z-buffer algorithm** (depth buffer).

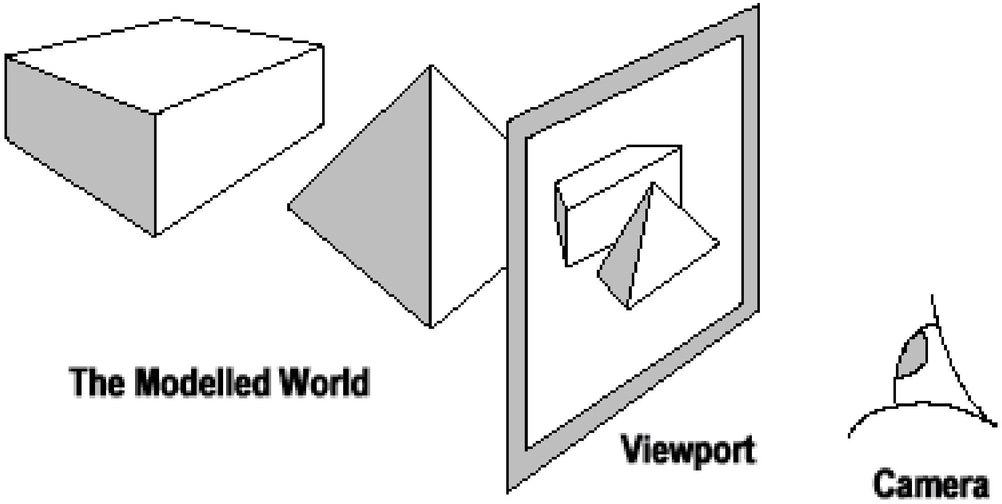


Figure 1. The Basics of 3D Graphical Processing

Z Buffer Algorithm

- As polygons are rasterized, hardware keeps track of **depth** or **z buffer**:
  - Initially, depth value is registered to far side of viewing volume
  - For each fragment, we compute the depth (distance from viewer).
    - If this depth is closer to viewer than current value, then we update color value and depth
    - Otherwise, we disregard it.

Code:

```
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Reversing the Pipeline

- ***How do we map from a mouse coordinate to object coordinates?***
  - Have to reverse the transformation process to map from window coordinates back to object space.
  - **gluUnProject** performs this reversal
    - Works best with orthographic projections
    - Requires a **wz** argument, which specifies the depth:
      - **0.0:** near clipping plane
      - **1.0:** far clipping plane

gluUnProject Example

```
GLdouble modelview[16];
GLdouble projection[16];
GLint viewport[4];
double wx, wy, wz, ox, oy, oz;

glGetDoublev(GL_MODELVIEW_MATRIX, modelview);
glGetDoublev(GL_PROJECTION_MATRIX, projection);
glGetIntegerv(GL_VIEWPORT, viewport);

wx = MouseX;
wy = viewport[3] - MouseY - 1;
glReadPixels((int)wx, (int)wy, 1, 1,
GL_DEPTH_COMPONENT, GL_FLOAT, &wz);

gluUnProject(wx, wy, wz, modelview, projection,
viewport, &ox, &oy, &oz);
```

Picking

- Logical operation that allows user to identify object on the display.

- OpenGL provides a mechanism called **selection**:
    - Adjust clipping region and viewport.
    - Keep track of primitives rendered into region near the cursor.
    - Possible selected primitives stored in a **hit list**.

Start picking

```
GLint viewport[4];
glGetIntegerv(GL_VIEWPORT, viewport);  // Get viewport information

glSelectBuffer(SBSIZE, SelectBuffer);  // Setup hit buffer
glRenderMode(GL_SELECT);               // Switch to selection mode
glInitNames();                         // Setup name stack

glMatrixMode(GL_PROJECTION);               // Adjust projection to limit
glPushMatrix();                        // area we are interested in
glLoadIdentity();                      // (5x5 area around mouse position)
gluPickMatrix(MouseX, viewport[3] – MouseY, 5, 5, viewport);
gluPerspective(45.0, (GLdouble)(WindowWidth)/(GLdouble)(WindowHeight),
               0.1, 1000.0);

glMatrixMode(GL_MODELVIEW);

// draw scene
```

Stop picking

```
GLint hits;

glMatrixMode(GL_PROJECTION);
glPopMatrix();
glMatrixMode(GL_MODELVIEW);

hits = glRenderMode(GL_RENDER);
if (hits > 0)
  process_hits(hits, SelectBuffer);
```

Process Hits

```
void process_hits(GLint hits, GLuint *buffer){
  GLuint names;
  GLuint *bp = buffer;

  for (GLint i = 0; i < hits; i++) {
    names = bp; // # names, min, max, name0, ...
    bp += 3;

    // process hits
    bp += names;
  }
```

Resources/Credits

- **OpenGL Programming Guide:**

  http://glprogramming.com/red/chapter03.html

- **OpenGL Super Bible:**

  http://www.opengl-doc.com/Sams-OpenGL.SuperBible.Third/0672326019/ch04lev1sec2.html