Zeyang Li (Linus)
Carnegie Mellon University

Transformations in OpenGL

Overview

- Event-Driven Programming
- Stages of Vertex Transformation
- Basic Transformations
  - glTranslate
  - glRotate
  - glScale
- Order of Transformations
- Viewing Transformation
  - gluLookAt
- Projection Transformation
  - gluPrespective/glFrustum
  - glOrtho
- Camera

Frank's office hour will be
2:00PM-4:00PM
on Thursday,
for this week only

Event-Driven Programming

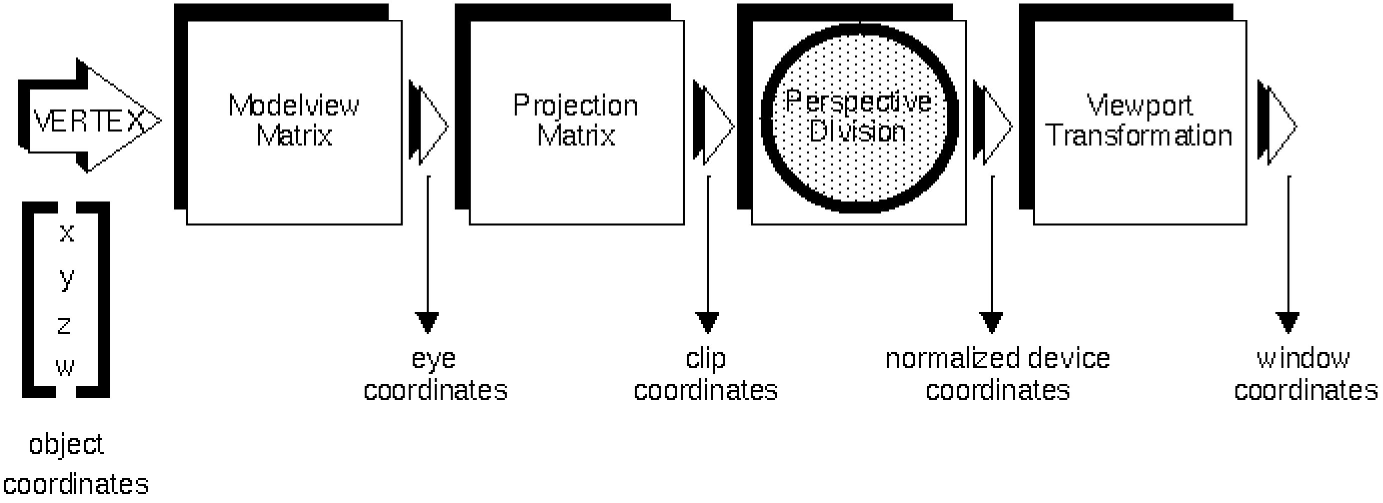
- A main loop and a bunch of callbacks.
- Each iteration, the loop processes some event, and invokes a callback function defined by the programmer.
- Ex. glut

```
// main.h
// declare event callbacks
void display();
void reshape(int width, int
height);
void keyboard(unsigned char
key, int x, int y);
```

```
// main.cpp
int main(){
 glutInitDisplayMode(GLUT_RGBA|
GLUT_DOUBLE|GLUT_DEPTH);
 glutInitWindowSize(500, 500);
 glutCreateWindow(“window”);
 glutDisplayFunc(display);
 glutReshapeFunc(reshape);
 glutKeyboardFunc(keyboard);
 glutMainLoop();
}
```

Stages of Vertex Transformation

- We will talk about Modelview Matrix and Projection Matrix



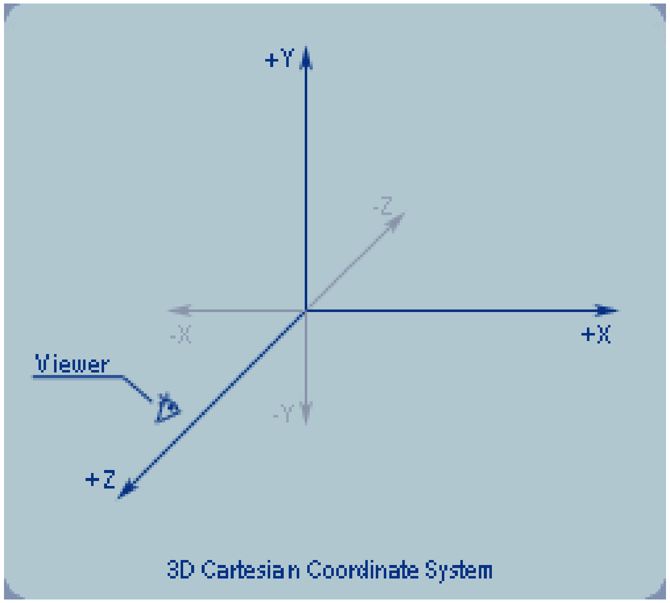
Matrix Modes

- Matrix Mode(glMatrixMode)
  - ModelView Matrix (GL_MODELVIEW)
    - Model related operations: glBegin, glEnd, glTranslate, glRotate, glScale, gluLookAt…
  - Projection Matrix (GL_PROJECTION)
    - Setup projection matrix: glViewport, gluPerspective/glOrtho/glFrustum…
  - Screen coordinates is computed by
    - Projection * ModelView * object coordinates
    - Then normalized for viewport size

Basic Transformations

- Some sample code

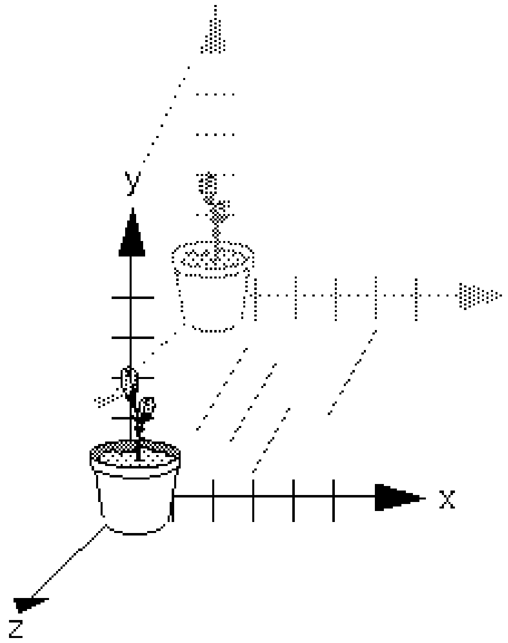
```
Display(){
        …
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glTranslatef(0.0, 0.0, -6.0);
        glRotatef(45.0, 0.0, 0.0);
        glScalef(2.0, 2.0, 2.0);
        DrawCube();
        …
}
```



3D Cartesian Coordinate System

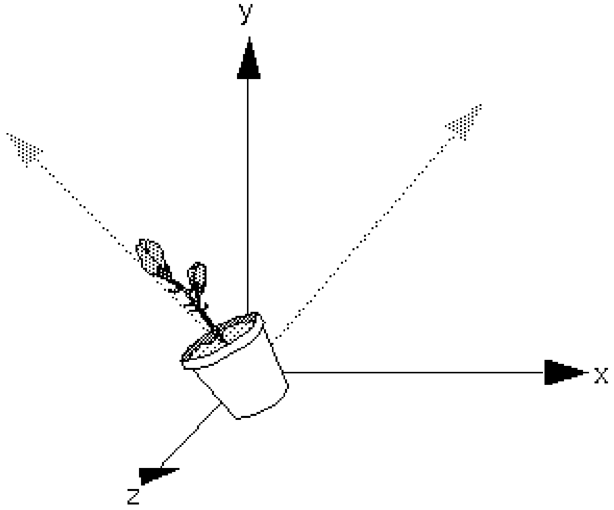
Basic Transformations

- glTranslate{fd}(TYPE x, TYPE y, TYPE z);
  - Move an object by the given x-, y-, z-values.



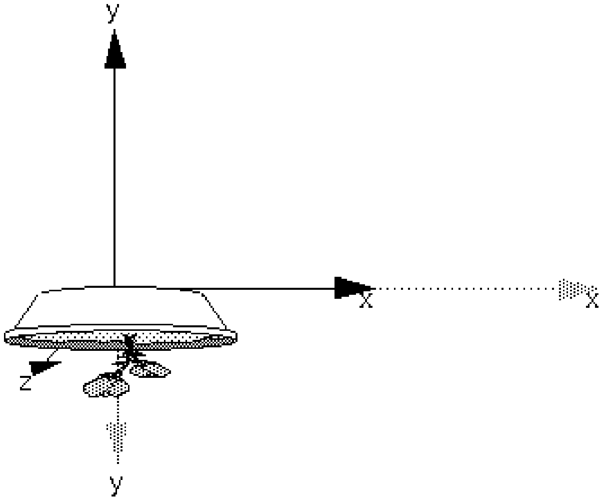
Basic Transformations

- glRotate{fd}(TYPE angle, TYPE x, TYPE y, TYPE z);
  - Rotates an object in a counterclockwise direction about the vector (x,y,z).
  - Ex. glRotatef(45.0, 0.0, 0.0, 1.0);



Basic Transformations

- glScale{fd}(TYPE, x, TYPE y, TYPE z);
  - ▫ Multiply the x-, y-, z-coordinate of every point in the object by the corresponding argument x, y, or z.
  - ▫ Ex. glScalef(2.0, -0.5, 1.0);



Basic Transformations

- glPushMatrix() / glPopMatrix()
  - Save/Load current modelview matrix to/from a stack
  - Useful when different parts of an object transform in different ways.

Basic Transformations

- glPushMatrix() / glPopMatrix()
- Ex. simple robot with a head, a body, two arms

```
transform robot
glPushMatrix()
        transform head
        draw head
glPopMatrix()

glPushMatrix()
        transform body
        glPushMatrix()
                transform left_arm
                draw left_arm
        glPopMatrix()

        glPushMatrix()
                transform right_arm
                draw right_arm
        glPopMatrix()
        draw body
glPopMatrix()
```

Order of Transformations

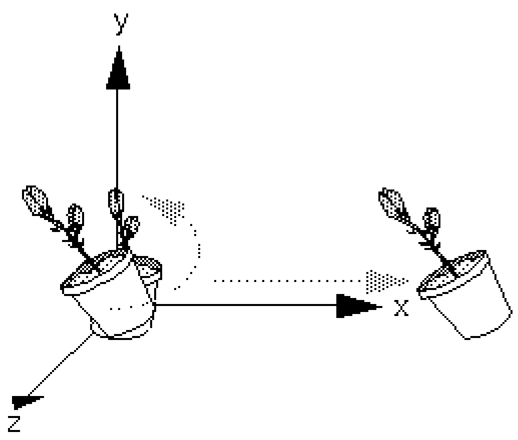
- Call order is the reverse of the order the transforms are applied.
- Different call orders result in different transforms!

```
// Example 1
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0,0.0,-6.0);
glRotatef(45.0,0.0,1.0,0.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
```

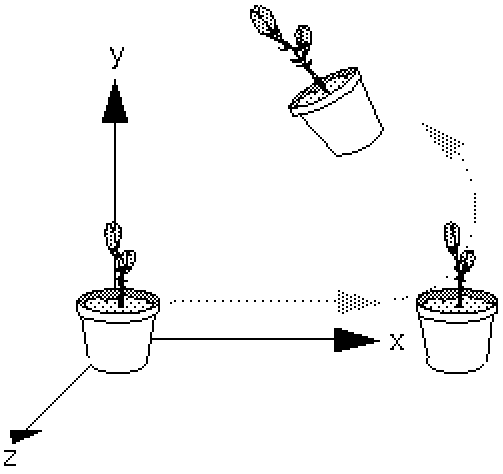
```
// Example II
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(45.0,0.0,1.0,0.0);
glTranslatef(0.0,0.0,-6.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
```

Order of Transformations

- Each transform multiplies the object by a matrix that does the corresponding transformation.
- The transform closest to the object gets multiplied first.



Rotation first



Translation first

Order of Transformations

- Let
  - glTranslate = Mat Trans
  - glRotate = Mat Rot
  - glScale = Mat Scale
  - DrawCube = v

```
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 0.0, -6.0);
glRotatef(45.0,0.0,1.0, 0.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
```

- Modelview matrix:
  - Identity -> Trans -> Trans*Rot -> Trans*Rot*Scale -> Trans*Rot*Scale*v
  - Or, Trans(Rot(Scale*v))).
  - So Scale is applied first, then Rot, then Trans

Order of Transformations

```
// Example 1
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0.0, 0.0, -6.0);
glRotatef(45.0, 0.0, 1.0, 0.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
= Trans * Rot * Scale * v
```

```
// Example II
Display(){
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(45.0, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, -6.0);
glScalef(2.0, 2.0, 2.0);
DrawCube();
…}
= Rot * Trans * Scale * v
```

- Generally, do not expect different orders of transforms to produce the same result, because matrix multiplication is not commutative.

Order of Transformations

- Another way to think about transforms.
  - Move a local coordinate system.
    - Each object has a local coordinate system.
    - Transforms happen relative to this coordinate system.
    - Unfortunately , breaks down when scale is involved.
  - P.119, OpenGL Programming Guide (5th edition.

Viewing Transformations

- How to position your camera
  - ▫ Method I. Use transform functions to position all objects in correct positions.
  - ▫ Method II.

```
gluLookAt( eye_x, eye_y, eye_z
            center_x, center_y, center_z,
            up_x, up_y, up_z)
```

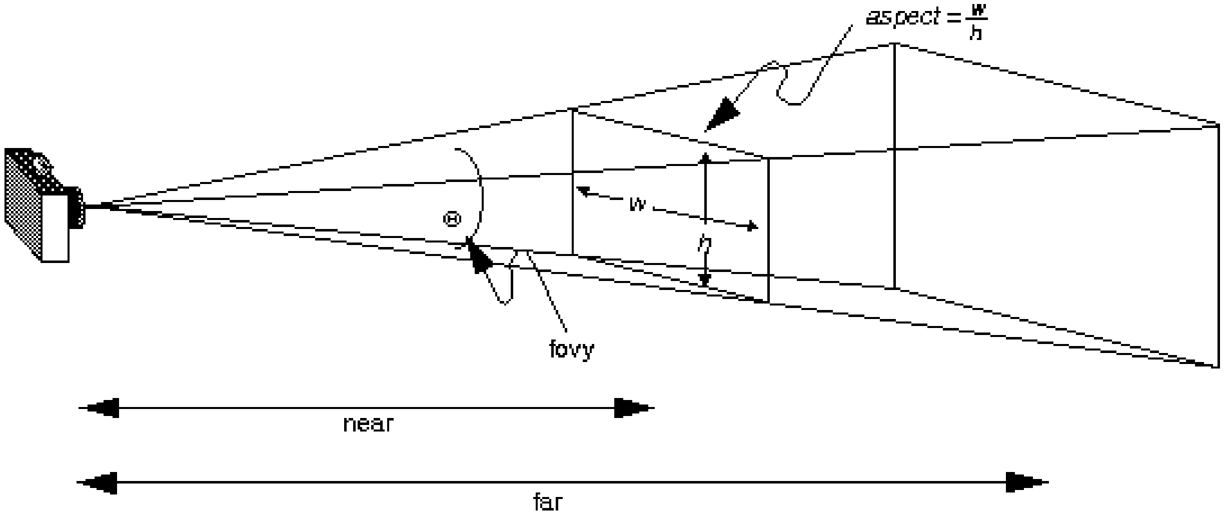
    - Which is a just a bunch of GL transformations
    - Should be used after
      glMatirxMode(GL_MODELVIEW)
      glLoadIdentity();

Projection Transformations

- glOrtho
  - Orthographic projection(objects appear the same size, no matter the distance)
- gluPerspective/glFrustum
  - Perspective projection
  - Both do the same thing, but take different set of arguments
  - gluPerspective is rumored to be more intuitive to use…

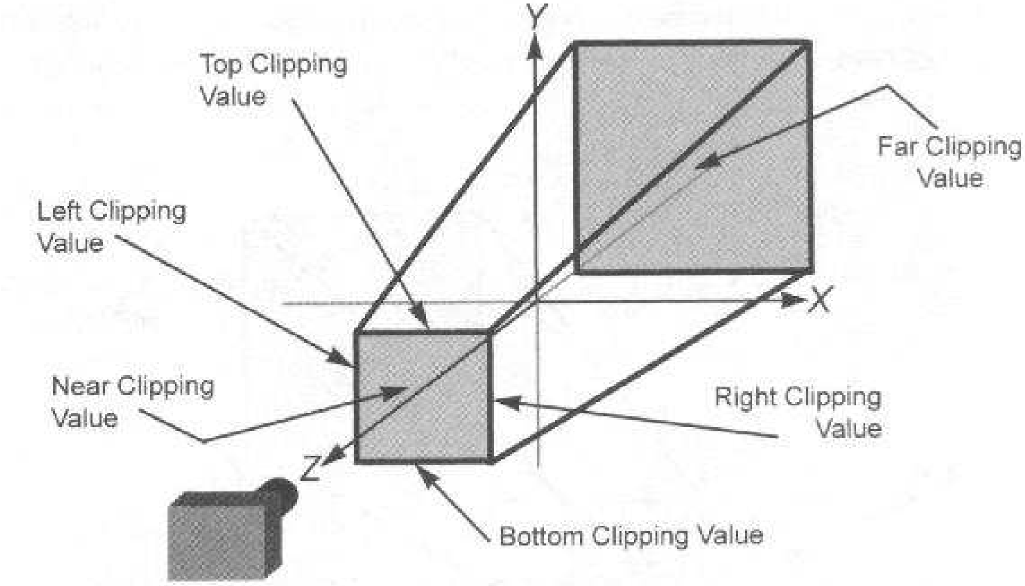
Projection Transformations

- gluPerspective(fovy, aspect, near, far)
  - Field of view is in angle(bigger, objects smaller)
  - Aspect ratio is usually set to width/height
  - Near clipping plane must > 0



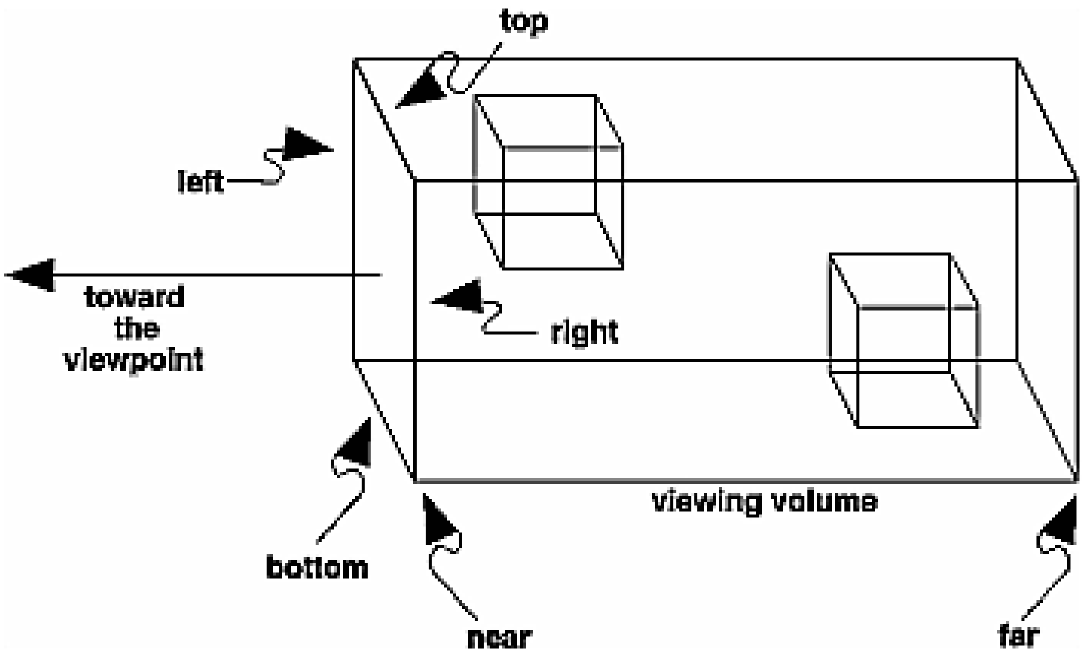
Projection Transformations

- glFrustum(left, right, bottom, top, near, far)
  - More general than gluPerspective
  - gluPerspective only produces symmetrical projections



Projection Transformations

- glOrtho(left, right, bottom, top, near, far)
  - Specify clipping coordinates in six directions.



Camera

- Camera class
  - Uses quaternions to avoid inconvenience in matrix and angle axis representation
  - Has methods to help you get arguments for gluPerspective and gluLookAt easily
- Quaternion class has “to_angle_axis” function.

Resources

- OpenGL Programming Guide
- opengl.org FAQ
    - http://www.opengl.org/resources/faq/technical/transformations.htm
- Nate Robin's Tutorials
    - Really good, have demos that allow you to dynamically tune function parameters
- Some Reading on Quaternions
    - http://www.gamedev.net/reference/articles/article1095.asp
- Google
    - OpenGL/glut tutorials are plenty
    - Help you code, but not so much for understanding