OpenGL Tutorial
CISC 640/440 Computer Graphics

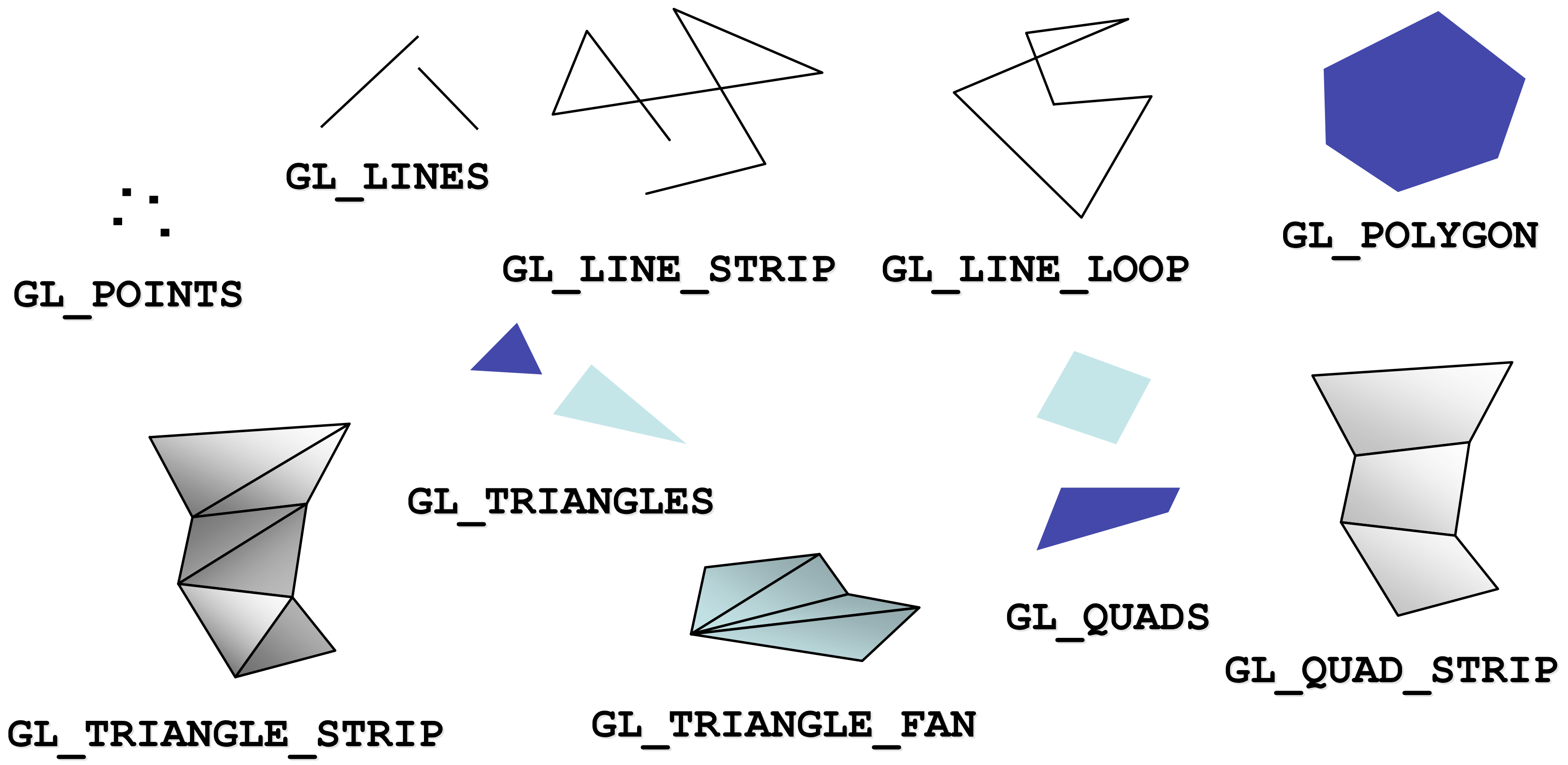
TA: Qi Li/Mani Thomas

qili@cis.udel.edu/manivt@cis.udel.edu

OpenGL: What is It?

- **GL** (**G**raphics **L**ibrary): Library of 2-D, 3-D drawing primitives and operations
  - API for 3-D hardware acceleration
- **GLU** (**GL U**tilities): Miscellaneous functions dealing with camera set-up and higher-level shape descriptions
- **GLUT** (**GL U**tility **T**oolkit): Window-system independent toolkit with numerous utility functions, mostly dealing with user interface

OpenGL Geometric Primitives



Specifying Geometric Primitives

- Primitives are specified using

```
glBegin(primType);
...
glEnd();
```

  – *primType* determines how vertices are combined

```
GLfloat red, green, blue;
GLfloat x, y;

glBegin(primType);
for (i = 0; i < nVerts; i++) {
  glColor3f(red, green, blue);
  glVertex2f(x, y);
  ... // change coord. values
}
glEnd();
```

OpenGL Vertex/Color Command Formats

`glVertex3fv( `*`v`*` )`

`glColor3fv( `*`v`*` )`

Number of components

2 -	(x,y)
3 -	(x,y,z), (r,g,b)
4 -	(x,y,z,w), (r,g,b,a)

Data Type

b	- byte
ub	- unsigned byte
s	- short
us	- unsigned short
i	- int
ui	- unsigned int
f	- float
d	- double

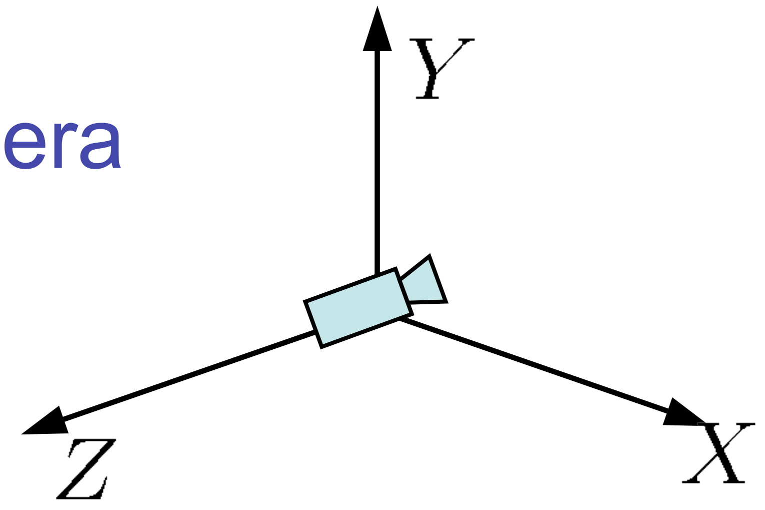
Vector

omit "v" for scalar form– e.g.,

```
glVertex2f(x, y)
glColor3f(r, g, b)
```

OpenGL 3-D coordinates

- Right-handed system
- From point of view of camera looking out into scene:
  - +X right, -X left
  - +Y up, -Y down
  - +Z **behind** camera, -Z in front
- Positive rotations are counterclockwise around axis of rotation



Transformations in OpenGl

- Modeling transformation
- Viewing transformation
- Projection transformation

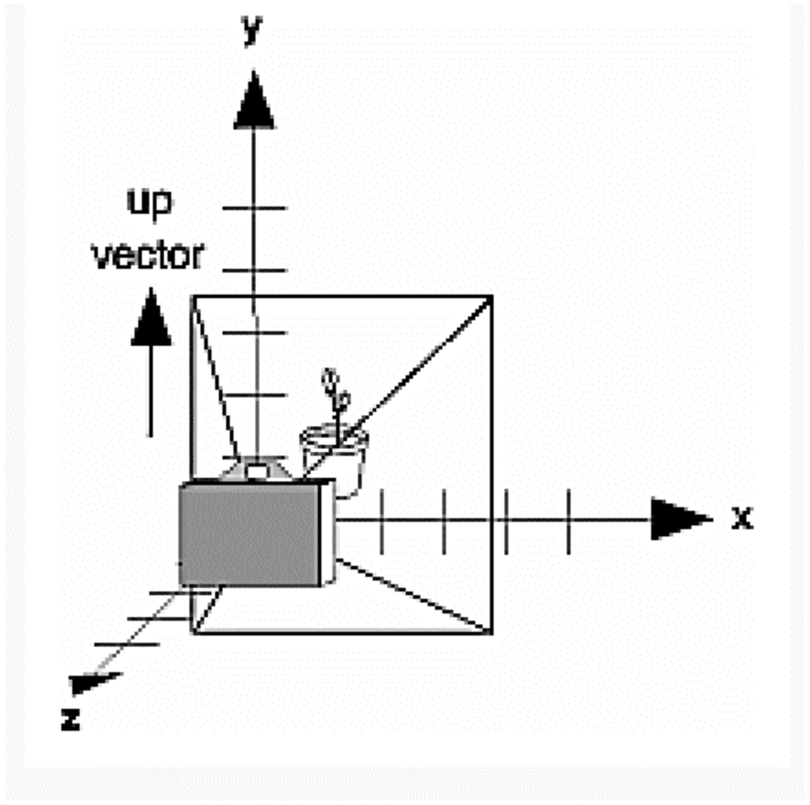
Modeling Transformation

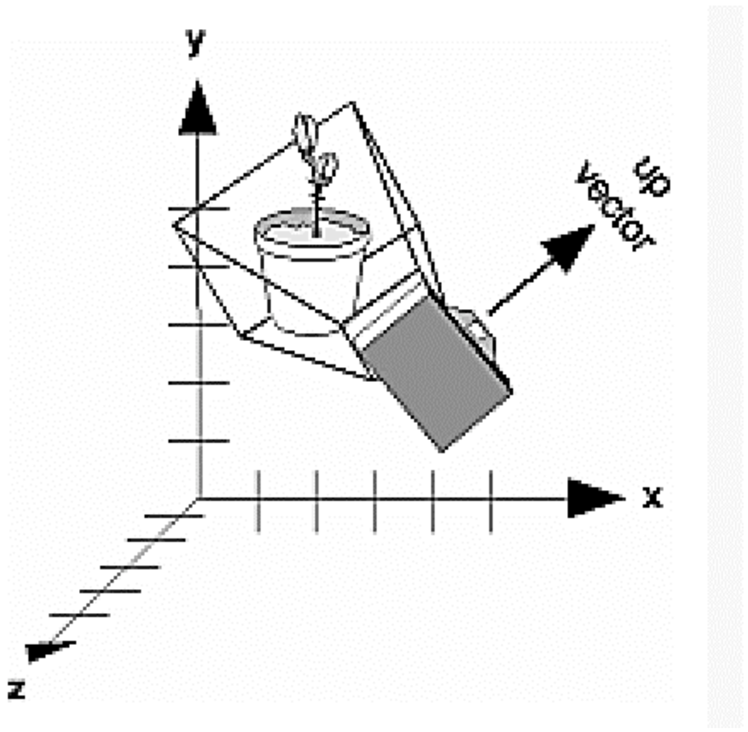
- Refer to the transformation of models (i.e., the scenes, or objects)
- Generally,
  - glMultMatrixf(M_i)
- Some simple transformations
  - Translation: glTranslate(x,y,z)
  - Scale: glScale(sx,sy,sz)
  - Rotation: glRotate(theta, x,y,z)
    - x,y,z are components of vector defining axis of rotation
    - Angle in degrees; direction is counterclockwise

Viewing Transformation

- Refer to the transformation on the camera
- Using glTranslate*() and glRotate*()
- Using gluLookAt()
  - gluLookAt (eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)
    - **eye** = (eyeX, eyeY, eyeZ)^T: Desired camera position
    - **center** = (centerX, centerY, centerZ)^T: Where camera is looking
    - **up** = (upX, upY, upZ)^T: Camera's "up" vector

Viewing Transformation

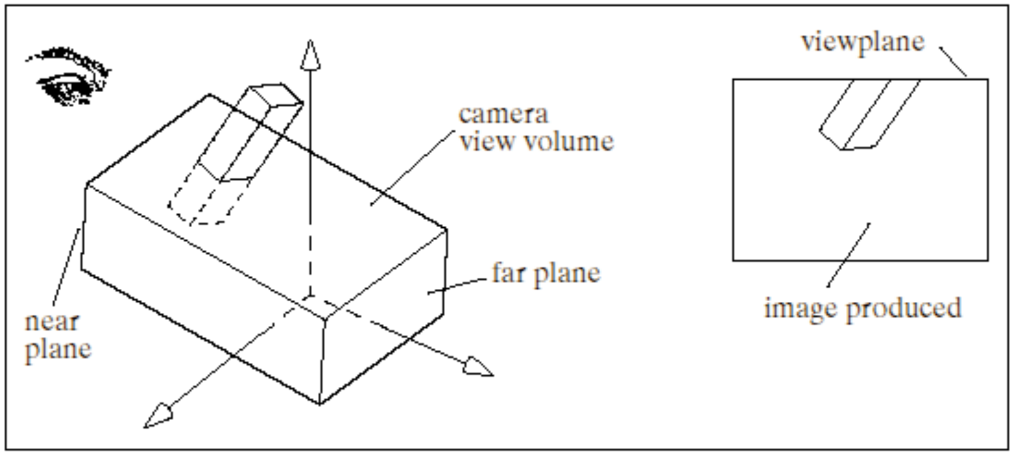




from Woo et al

Projection Transformation

- Refer to the transformation from scene to image
- Orthographic projection
  - glOrtho (left, right, bottom, top, near, far)



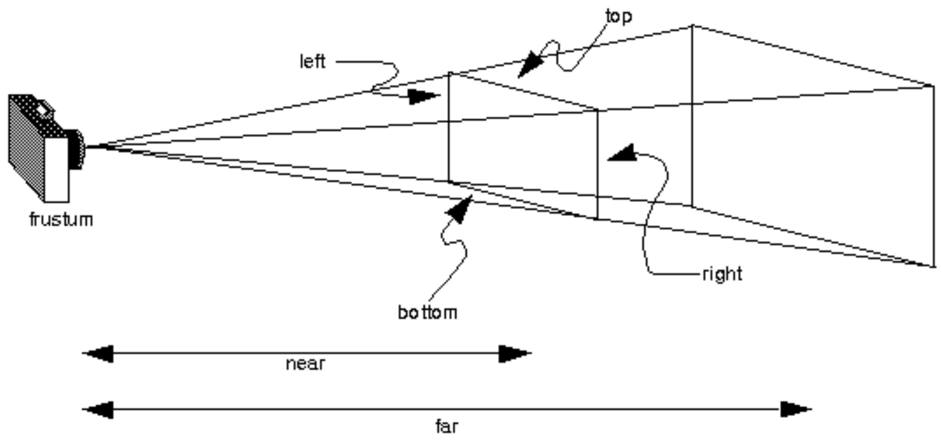
from Hill

Projection Transformation

- Refer to the transformation from scene to image
- Orthographic projection
  - glOrtho (left, right, bottom, top, near, far)
- Perspective projection
  - glFrustum (left, right, bottom, top, near, far)

Projection Transformation

- Refer to the transformation from scene to image
- Orthographic projection
  - glOrtho (left, right, bottom, top, near, far)
- Perspective projection
  - glFrustum (left, right, bottom, top, near, far)



Notes on openGl transformations

- Before applying modeling or viewing transformations, need to set **`glMatrixMode(GL_MODELVIEW)`**
- Before applying projection transformations, need to set **`glMatrixMode(GL_Projection)`**

Notes on openGl transformations

- Before applying modeling or viewing transformations, need to set `glMatrixMode(GL_MODELVIEW)`
- Before applying projection transformations, need to set `glMatrixMode(GL_Projection)`
- Replacement by either following commands

```
glLoadIdentity();
glLoadMatrix(M);
```

- Multiple transformations (either in modeling or viewing) are applied in **reverse** order