OpenGL Transformations

Excerpted from

An Interactive Introduction to OpenGL Programming

Dave Shreiner

Ed Angel

Vicki Shreiner

Transformations in OpenGL

Ed Angel

Transformations in OpenGL

Pipline สำหรับแปลงในเรื่องต่อ

Modeling → แปลงโมเดล ← ย้ายตำแหน่ง (translate), ย่อ/ขยายขนาดโมเดล (scale), หมุนโมเดล (Rotate) ในทิศต่างๆ

Viewing

- orient camera จัดมุมกล้อง ตั้งกล้อง เล็งกล้องไปที่จุดที่โมเดลที่เราต้องการ

Projection → จากภาพที่กล้องเห็นให้อยู่ในระนาบ แมพบนพื้นที่ 2 มิติ ปรับขนาดเวลล์ภายกล้อง, แคบ

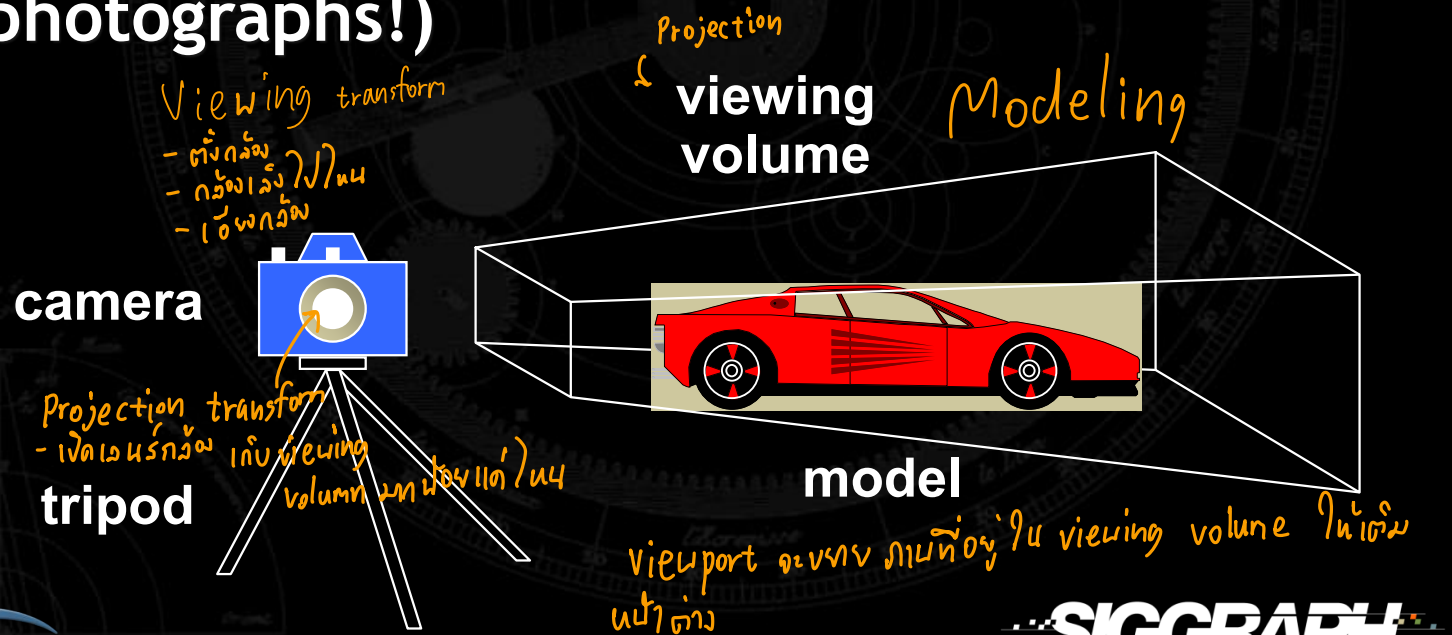
Viewport (Map to screen)

transformation เอาข้อมูลที่แปลงไปแล้วให้แมพอยู่บนจอภาพ

ขนาดเท่าหน้าต่างเรา

Camera Analogy

3D is just like taking a photograph (lots of photographs!)



Camera Analogy and Transformations

Modeling transformations

- moving the model

Viewing transformations

- tripod–define position and orientation of the viewing volume in the world

Projection transformations

- adjust the lens of the camera ปรับตั้งเลนส์หมกล้อง

Viewport transformations

- enlarge or reduce the physical photograph

Affine Transformations

Want transformations which preserve geometry

- lines, polygons, quadrics → พื้นผิวที่เกิดจากรวมกันเส้นตรง

Affine = line preserving

- If two lines are parallel before an affine transformation then they will be parallel afterwards.
- Translation, Rotation, Scaling → เป็น Affine transform
- Concatenation (composition) → ผสม, รวมกัน ขั้นตอนชั้น
- Orthographic Projection

Perspective ไม่เป็นเพราะ เส้นไม่ขนาน เหมือนเดิม

Homogeneous Coordinates

- ใช้ vector 4 compo

- each vertex is a column vector

$$\vec{v} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

$w = 1$ → เป็นตำแหน่งเวคเตอร์ (ฟิกัส)
ถ้าเป็น 0 หมายถึง เวกเตอร์ที่ชี้ทิศทาง

- w is usually 1.0
- all operations are matrix multiplications
- directions (directed line segments) can be represented with w = 0.0

3D Transformations

เพื่อให้คูณกับ Homogeneos Coordinate [ ]4 4 [ ]4 1 กฎการคูณ matrix

A vertex is transformed by 4 x 4 matrices

- all affine operations are matrix multiplications
- all matrices are stored column-major in OpenGL
- matrices are always post-multiplied ตัวที่ถูกคูณจะอยู่ด้านหลัง
- product of matrix and vector is $\mathbf{M}\,\vec{v}$

modeling matrix

$$\mathbf{M} = \begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix}$$

เก็บข้อมูลเป็นคอลัมน์ - เรียงลงมา

Pre-multiplication and Post-multiplication

- It is important to remember that AB and BA are usually not the same
- Consequently, it is common to use the terms "pre-multiplication" and "post-multiplication"
- A is post-multiplied by B," or "B is pre-multiplied by A," we are referring to the product AB
- B is post-multiplied by A," or "A is pre-multiplied by B," we are referring to the product BA

Statpower.net. (2019). Available at: http://www.statpower.net/Content/312/Lecture Slides/Matrix 2.pdf.

Specifying Transformations

- รวม matrix เพื่อจะทำการคูณข้อมูลที่จะถูกส่งผ่านไปยัง pipeline

Programmer has two styles of specifying transformations

- specify matrices (**glLoadMatrix, glMultMatrix**)
  - เราเตรียม matrix และทำการ load เอง → state matrix
  - โหลดเข้าไปใน state ที่เก็บข้อมูล matrix
  - ที่บอกให้ไปคูณกับค่า matrix ที่อยู่ใน state เดิม
- specify operation (**glRotate, glOrtho**) 3D
  - mt 4x4 คูณค่าเดิมที่อยู่ในstate

Programmer does not have to remember the exact matrices

- check appendix of Red Book (Programming Guide)

Programming Transformations

Prior to rendering, view, locate, and orient:

- eye/camera position
- 3D geometry

Manage the matrices

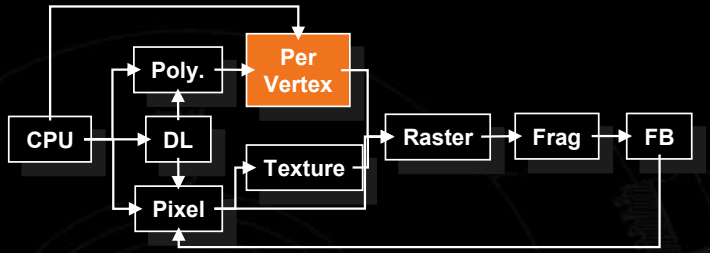
- including matrix stack

ใช้ stack ในการจัดการ matric มี push, pop

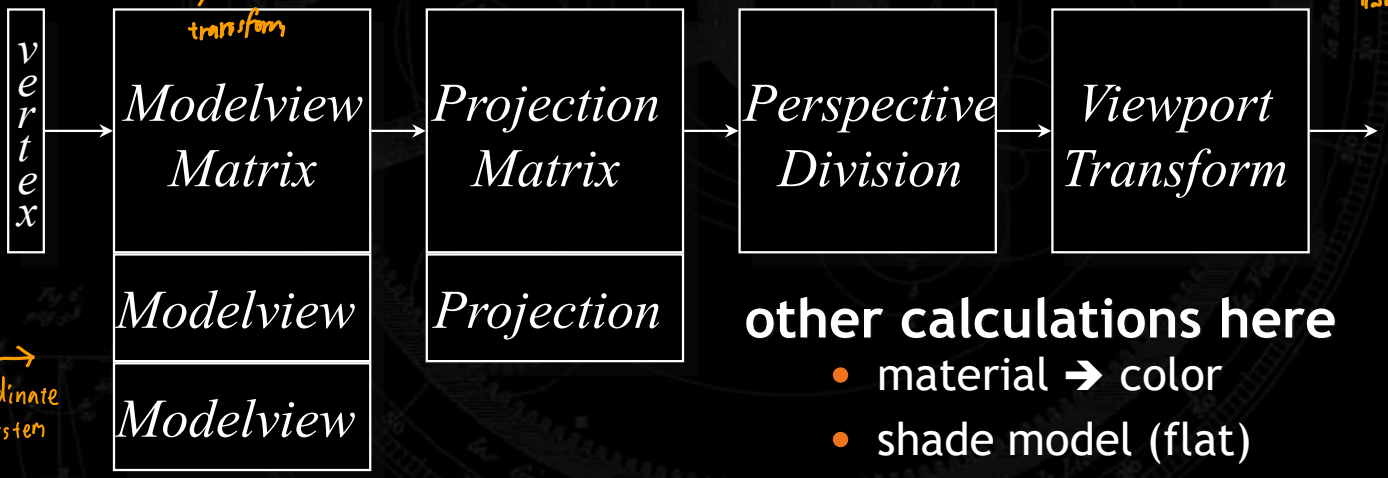
Combine (composite) transformations

scale x rotate x translate

Transformation Pipeline



hard ware



coodinate system

other calculations here

- material → color
- shade model (flat)
- polygon rendering mode
- polygon culling
- clipping

Matrix Operations

Specify Current Matrix Stack

```
glMatrixMode( GL_MODELVIEW or GL_PROJECTION )
```

Other Matrix or Stack Operations

```
glLoadIdentity()
```

วิธีสั่งโหลดค่า matrix เอกลักษณ์ I

```
glLoadMatrix{fd}()  glMultMatrix{fd}()
```

```
glPushMatrix()
```

L ค่า stage ของ matrix ที่เราset จะถูก push ลงไปใน stack

```
glPopMatrix()
```

L ค่า stage ที่เราเก็บไว้ใน stack จะย้อนกลับมา

Modeling Transformations

Move object เคลื่อนตำแหน่ง model

```
glTranslate{fd}( x, y, z )
```

Rotate object around arbitrary axis $(x \quad y \quad z)$

```
glRotate{fd}( angle, x, y, z )
```

- angle is in degrees องศา

แกนที่จะหมุน หมุนแกนใดให้แกนนั้นเป็น 1 และที่เหลือเป็น 0

Dilate (stretch or shrink) or mirror object

```
glScale{fd}( x, y, z )
```

2 1 1 → y, z scale เท่าเดิม

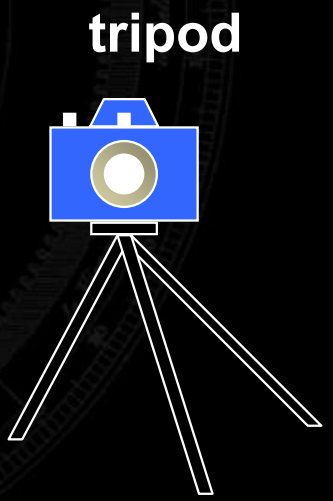
Viewing Transformations

Position the camera/eye in the scene

- place the tripod down; aim camera

To “fly through” a scene

- change viewing transformation and redraw scene

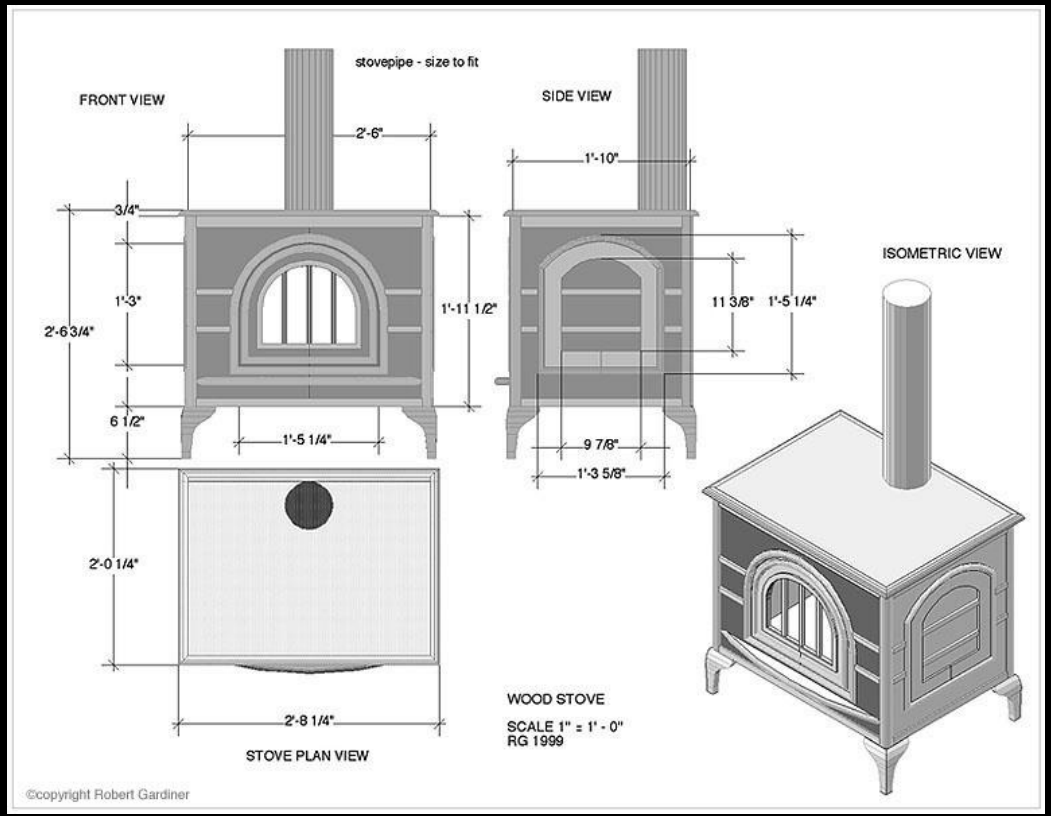



```
gluLookAt( eyex, eyey, eyez,
           aimx, aimy, aimz,
           upx, upy, upz )
```

- up vector determines unique orientation
- careful of degenerate positions

inserted slide

Orthographic Projection



http://www2.arts.ubc.ca/TheatreDesign/crslib/drft_1/cad/wdstv.htm

Perspective Projection

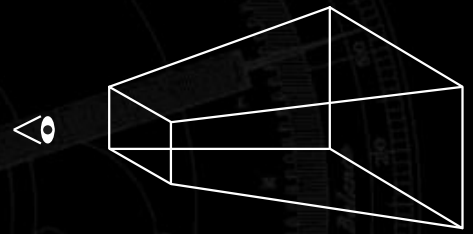
inserted slide

Orthographic

Perspective

Projection Transformation

Shape of viewing frustum



Perspective projection

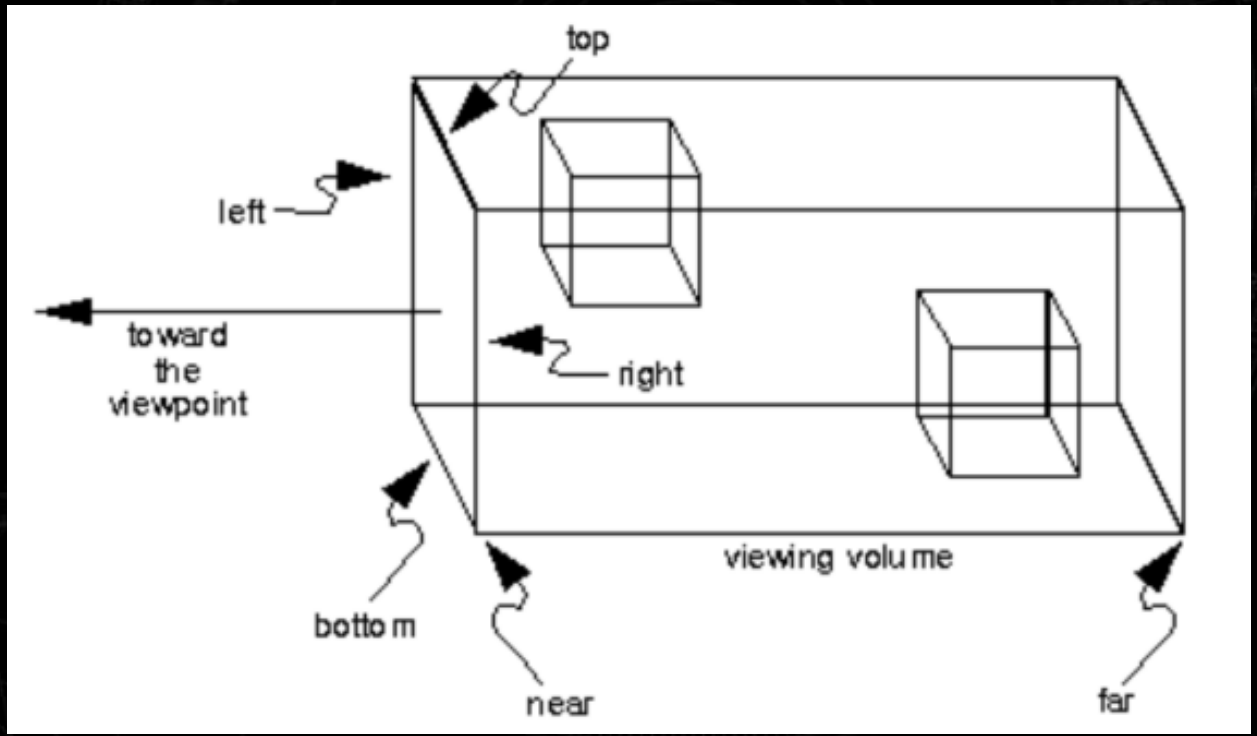
```
gluPerspective( fovy, aspect, zNear, zFar )
glFrustum( left, right, bottom, top, zNear, zFar )
```

Orthographic parallel projection

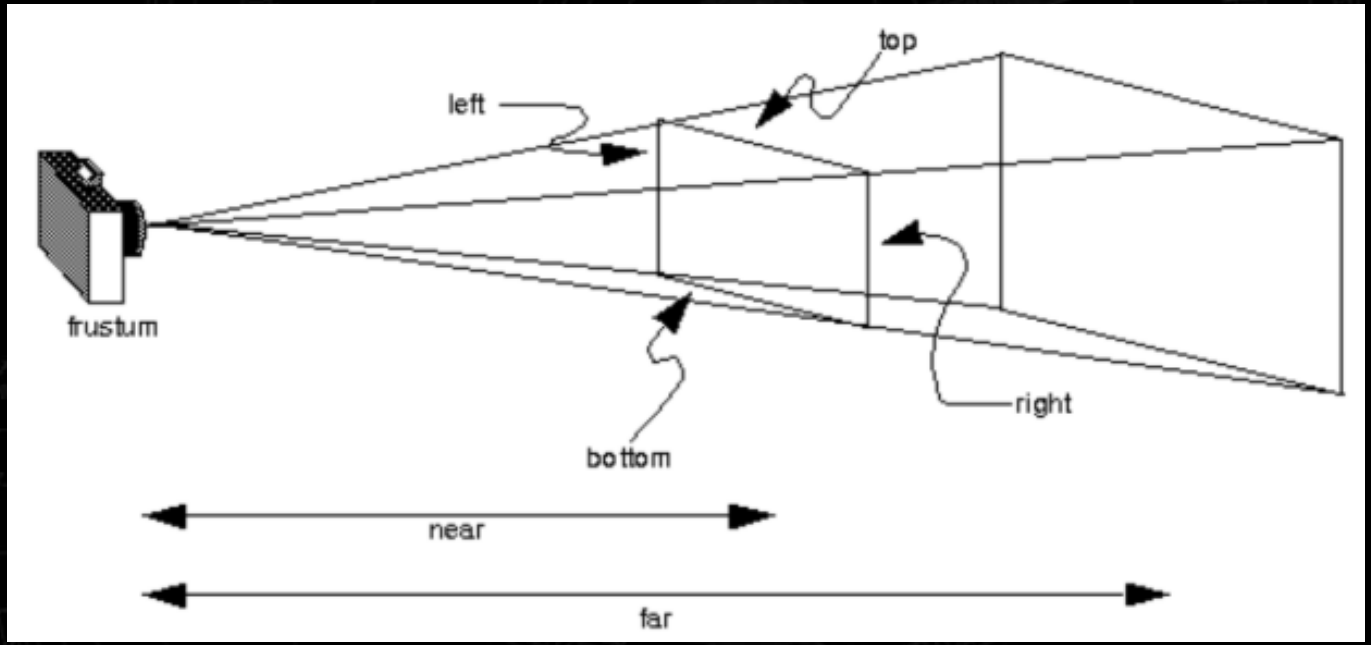
```
glOrtho( left, right, bottom, top, zNear, zFar )
gluOrtho2D( left, right, bottom, top )
```

- calls `glOrtho` with z values near zero

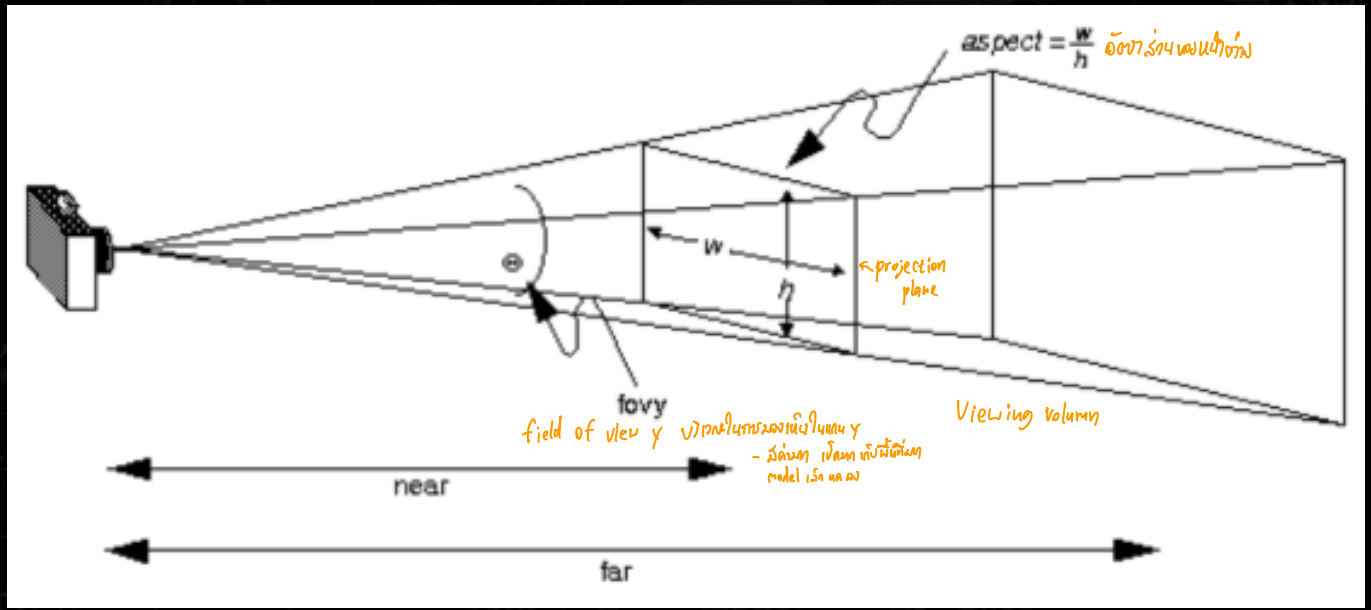
Orthographic Projection



Perspective Projection glFrustum()



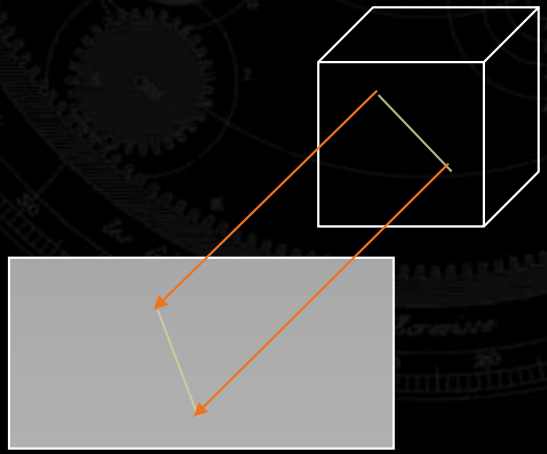
Perspective Projection gluPerspective()



Applying Projection Transformations

Typical use (orthographic projection)

```
glMatrixMode( GL_PROJECTION )
glLoadIdentity()
glOrtho( left, right, bottom, top, zNear, zFar )
```

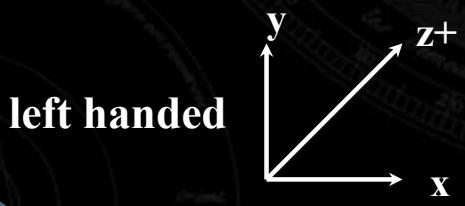


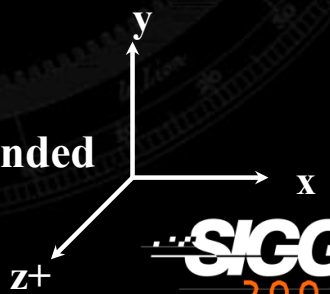
Connection: Viewing and Modeling

- Moving camera is equivalent to moving every object in the world towards a stationary camera
- Viewing transformations are equivalent to several modeling transformations
  - `gluLookAt()` has its own command
  - can make your own *polar view* or *pilot view*

Projection is left handed

- Projection transformations (`gluPerspective`, `glOrtho`) are left handed
  - think of ***zNear*** and ***zFar*** as distance from view point
- Everything else is right handed, including the vertexes to be rendered





Common Transformation Usage

2 examples of `resize()` routine

- restate projection & viewing transformations

Usually called when window resized

Registered as callback for `glutReshapeFunc()`

Viewport Transformation

Viewport

- usually same as window size
- viewport aspect ratio should be same as projection transformation or resulting image may be distorted

```
glViewport( x, y, width, height )
```

0 0 ปกติ แล้วแต่

reshape(): Perspective & LookAt

```
def reshape( w, h ):
    glViewport( 0, 0, w, h )
    glMatrixMode( GL_PROJECTION )
    glLoadIdentity()
    gluPerspective( 65, w / h, 1.0, 100.0 )

def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode( GL_MODELVIEW )
    glLoadIdentity()
    gluLookAt( 0.0, 0.0, 5.0,
               0.0, 0.0, 0.0,
               0.0, 1.0, 0.0 )
```

aspec near far

On-Line Resources

- `http://www.opengl.org`
  - start here; up to date specification and lots of sample code
- `news:comp.graphics.api.opengl`
- `http://www.sgi.com/software/opengl`
- `http://www.mesa3d.org/`
  - Brian Paul's Mesa 3D
- `http://www.cs.utah.edu/~narobins/opengl.html`
  - very special thanks to Nate Robins for the OpenGL Tutors
  - source code for tutors available here!

Books

OpenGL Programming Guide, 3rd Edition

OpenGL Reference Manual, 3rd Edition

OpenGL Programming for the X Window System

- includes many GLUT examples

Interactive Computer Graphics: A top-down approach with OpenGL, 2nd Edition

Authors

Dave Shreiner	shreiner@sgi.com
Ed Angel	angel@cs.unm.edu
Vicki Shreiner	vshreiner@sgi.com