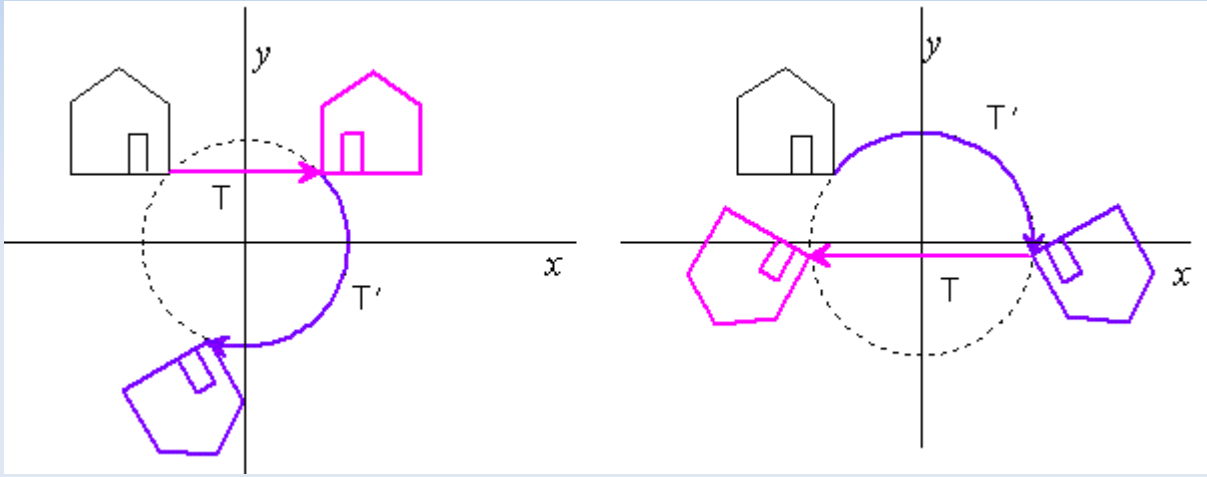
Transformaciones geométricas en 2D y 3D (Parte II)

Contenido

- Conmutatividad en transformaciones geométricas
- Tranformaciones básicas en 3D: rotación, traslación y escalamiento
- Otras tranformaciones 3D: reflexión y shearing
- Rotación con respecto a un eje arbitrario
- ¿Qué provee OpenGl?

¿Serán conmutativas las transformaciones?

- ¿ P' = T' T P = T T' P ?



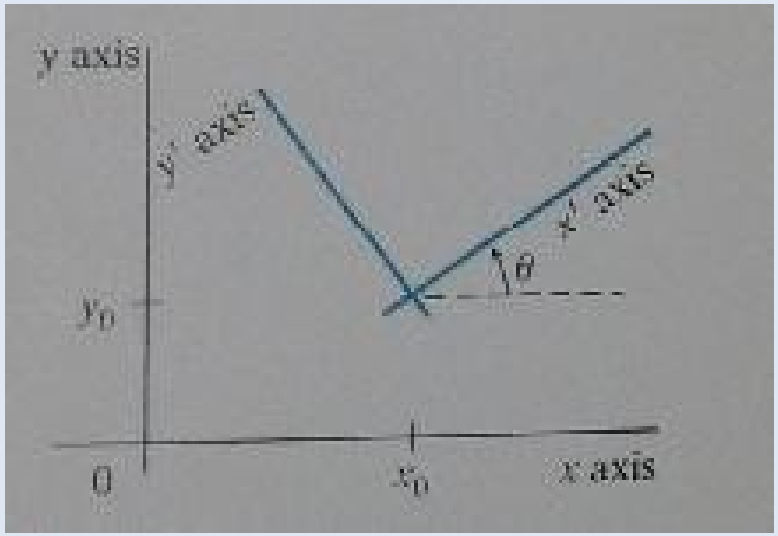
- T = reflexion y T' = rotacion no son conmutativas

¿Serán conmutativas las transformaciones?

- Son conmutativas en dos dimensiones;
  - Traslaciones entre si
  - Escalamiento entre si
  - Rotaciones entre si
  - Escalamiento ($s_x = s_y$) y rotación
  - Propuesto (demostrarlo)
- Composición arbitraria de transformaciones (escalamiento, rotación y traslación)
  - Rotaciones preservan ángulos y longitudes
  - Una secuencia arbitraria de transformaciones preserva el paralelismo de las líneas, pero no longitudes ni ángulos

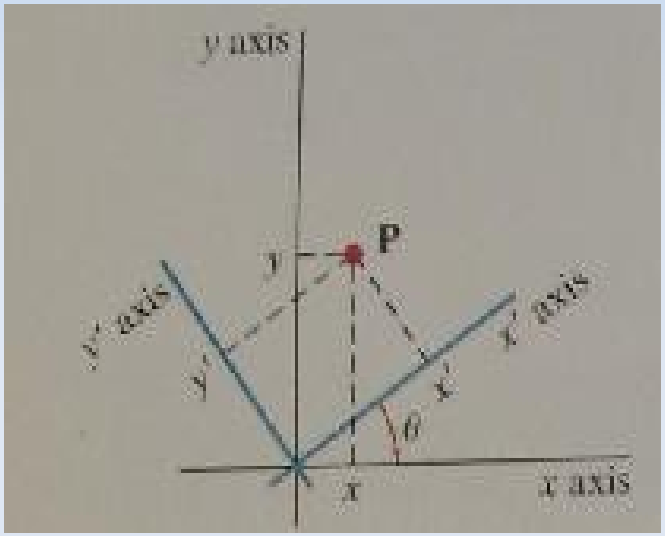
Transformaciones entre sistemas de coordenadas en 2D

- En las etapas del proceso de visualización necesitamos cambiar el sistema de coordenadas. Ejemplo: MCS-->WCS
- Ejemplo: Sistema de coordenada x'y' con origen en $(x_{0,} y_0)$ y orientación θ en el sistema de referencia xy.
  - Cómo transformar un objeto en coordenadas del sistema xy al sistema x'y'?



Transformaciones entre sistemas de coordenadas en 2D

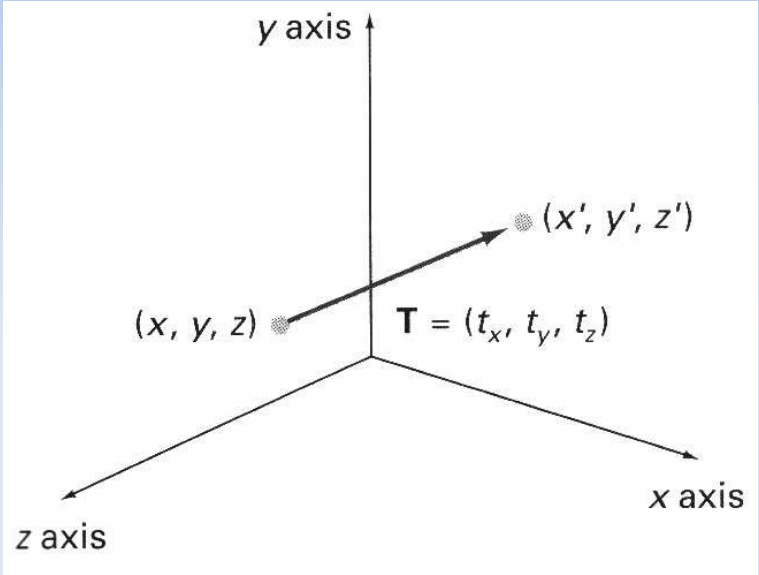
- Pasos a seguir:
  - Trasladar el origen del sistema x'y' $(x_{0,}\, y_0)$ de tal manera que coincida en (0,0) en el sistema xy
  - Rotar de tal manera que los ejes coincidan
  - Transformación resultante es:



$$M_{xy,x'\,y'} = R(-\theta)\,T(-x_{0,} - y_0)$$

Transformaciones en 3D

- Traslación

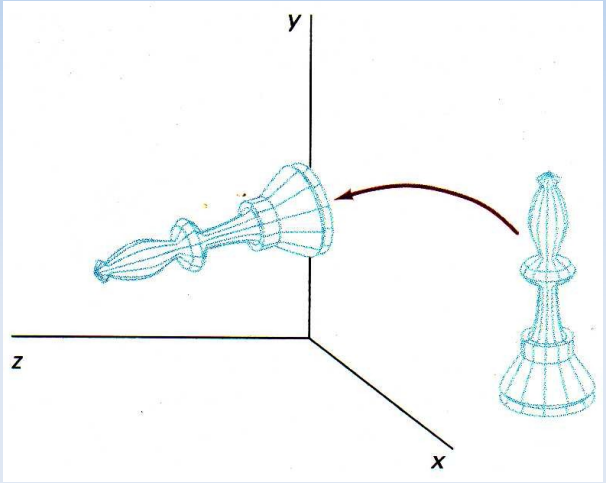


$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformaciones en 3D

- Rotación

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\operatorname{sen}\theta & 0 \\ 0 & \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

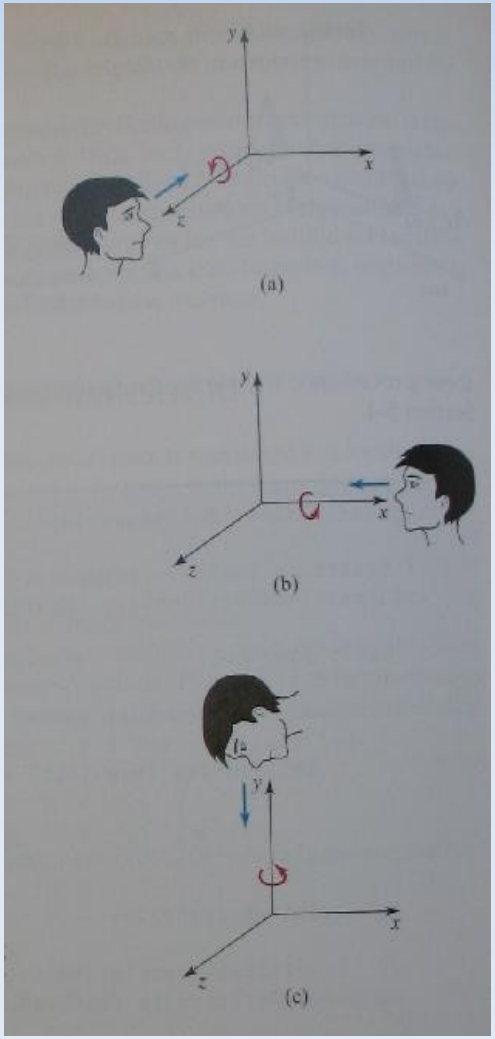


$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \operatorname{sen}\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & 0 & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformaciones en 3D

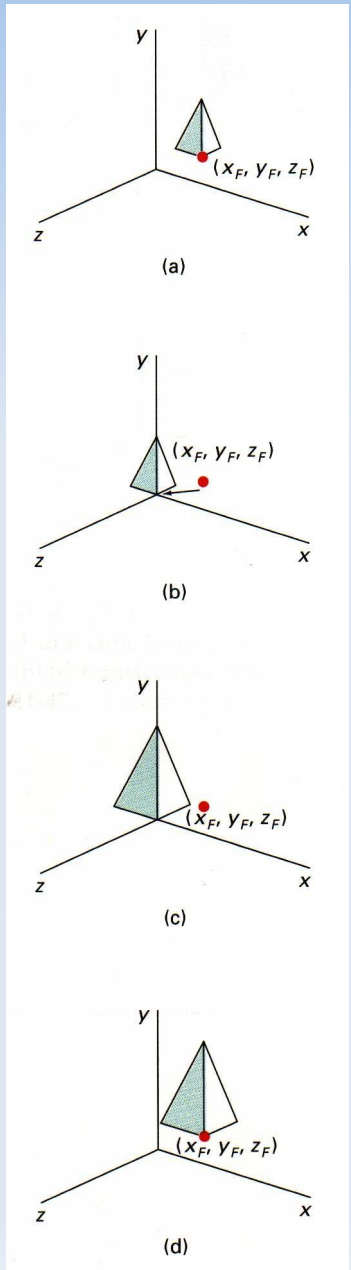
- Rotación positiva es ccw (counter clock wise) mirando desde la parte positiva del eje hacia el origen



Transformaciones en 3D

- Escalamiento

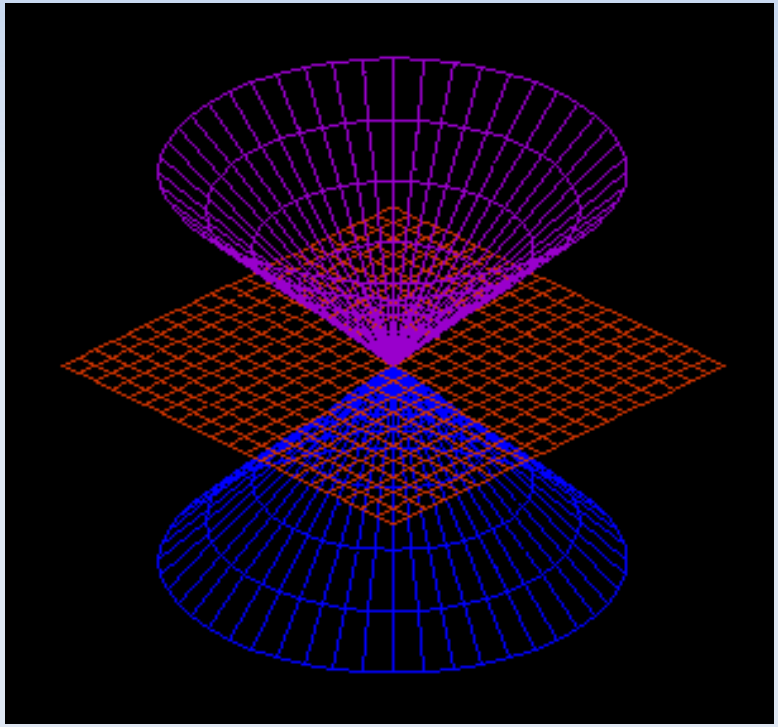
$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Otras transformaciones 3D

- Reflexión

$$RF_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- Propuesto: RF_{yz} y RF_{zx}

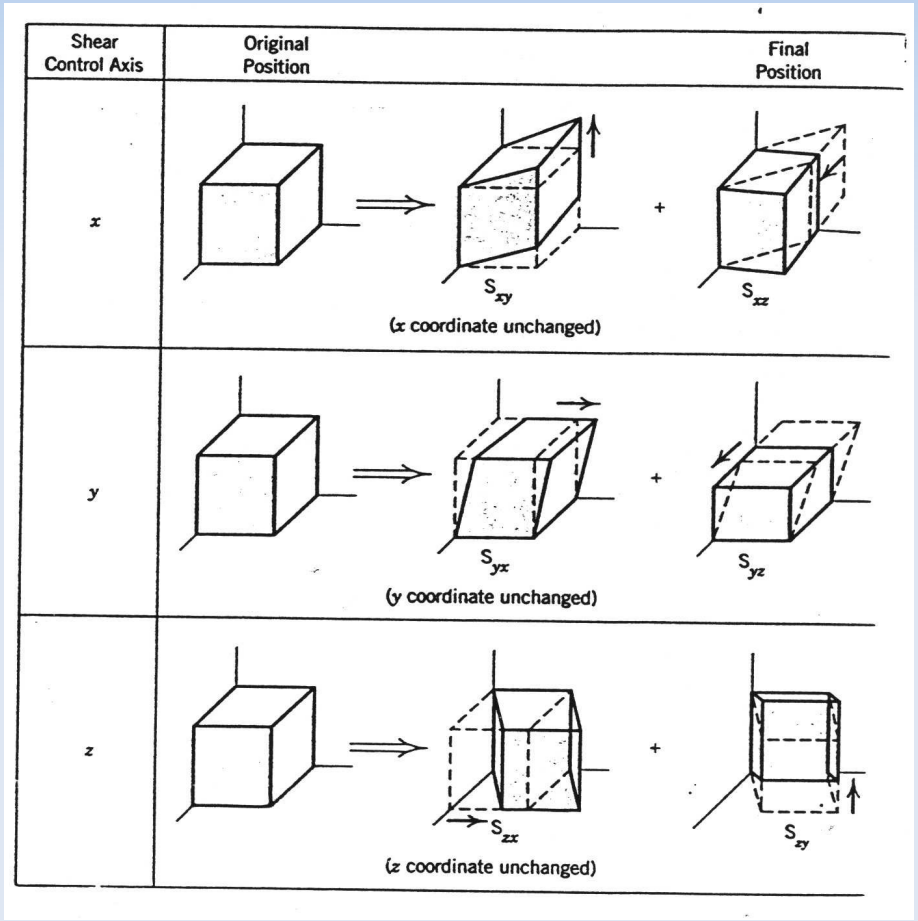
Otras transformaciones 3D

Shearing:

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

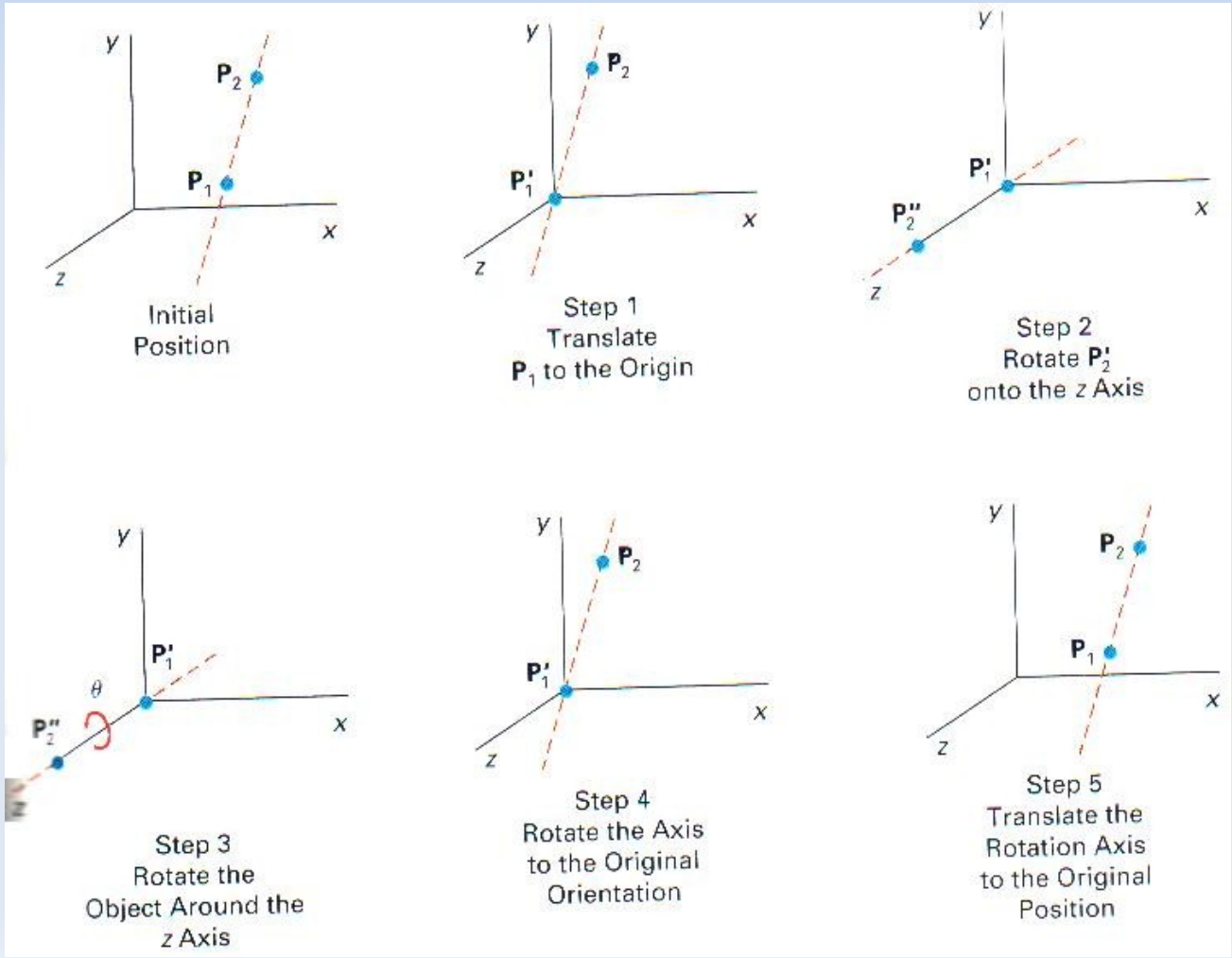
$$a = S_{zx} \quad b = S_{zy}$$

- Propuesto: SH_y y SH_x



Transformación con respecto a un eje arbitrario

Problema: dado un eje arbitrario representado por P_1P_2 determine la matriz de transformación que rota un punto R alrededor de P_1P_2 en un ángulo θ



Transformación con respecto a un eje arbitrario (2)

- Las coordenadas del eje de rotación son:

$$V = P_2 - P_1 = (x_2 - x_{1,}\, y_2 - y_{1,}\, z_2 - z_1)$$

- El vector unitario u es

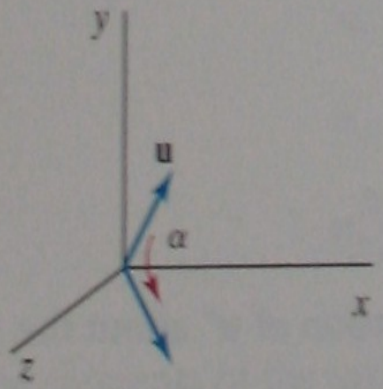
$$u = \frac{V}{|V|} = (a, b, c) = \left(\frac{x_2 - x_1}{|V|}, \frac{y_2 - y_1}{|V|}, \frac{z_2 - z_1}{|V|}\right)$$

- Paso 1: $T(-x_1, -y_1, -z_1)$

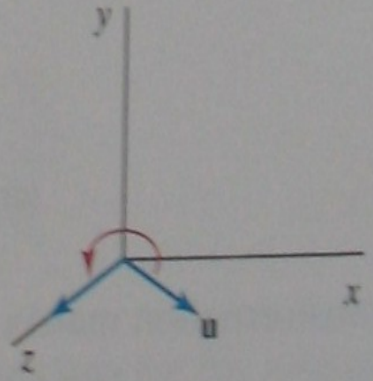
Transformación con respecto a un eje arbitrario (3)

- Paso 2: $u' = (0, b, c)$ $\cos(\alpha) = \frac{c}{d}$ $\sin(\alpha) = \frac{b}{d}$

$$d = \sqrt{b^2 + c^2}$$



(a)



(b)

FIGURE 5-45 Unit vector **u** is rotated about the x axis to bring it into the xz plane (a), then it is rotated around the y axis to align it with the z axis (b).

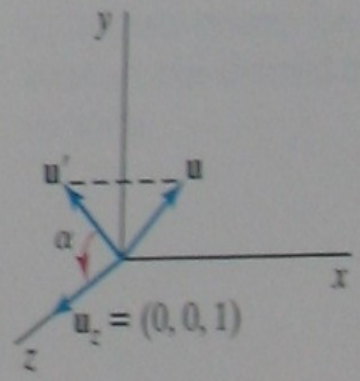


FIGURE 5-46 Rotation of **u** around the x axis into the xz plane is accomplished by rotating **u**′ (the projection of **u** in the yz plane) through angle α onto the z axis.

Transformación con respecto a un eje arbitrario (4)

- Paso 2: propuesto

$$\sin(\beta) = ?$$

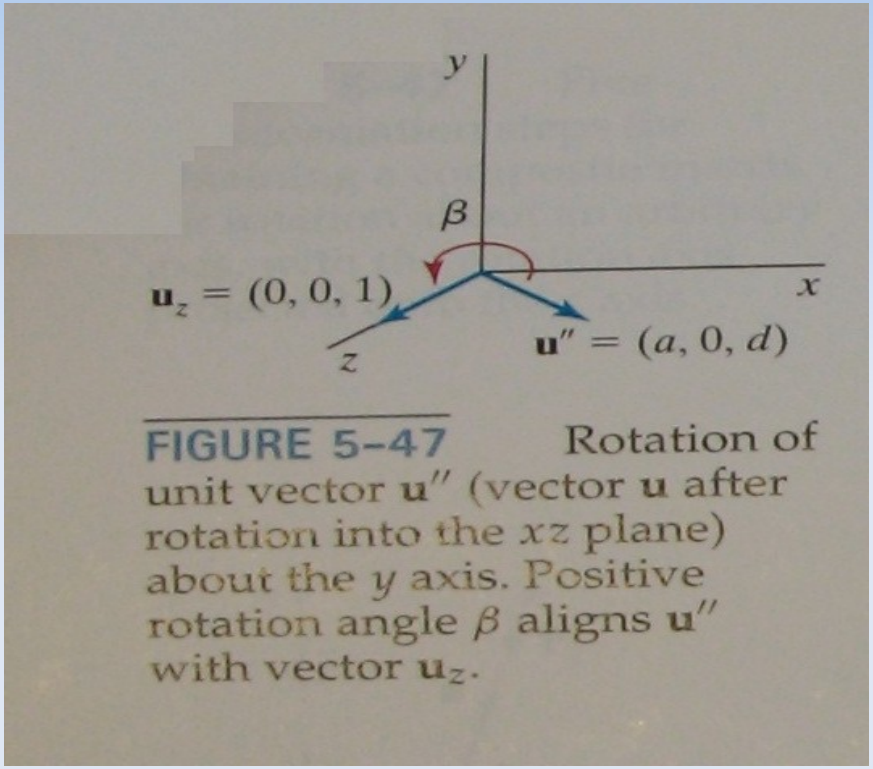


FIGURE 5-47 Rotation of unit vector $\mathbf{u}''$ (vector $\mathbf{u}$ after rotation into the xz plane) about the y axis. Positive rotation angle β aligns $\mathbf{u}''$ with vector $\mathbf{u}_z$.

- Matriz resultante

$$T(x_1, y_1, z_1) R_x(-\alpha) R_y(-\beta) R_z(\theta) R_y(\beta) R_x(\alpha) T(-x_1, -y_1, -z_1)$$

Nota: En las matrices de rotación $R^{-1} = R^T$ dado que solo la función seno cambia de signo

OpenGl .. ¿qué provee?

- glTranslate*(tx,ty,tz) (matriz de traslación 4x4)
  - glTranslatef(tx,ty,tz) (f = float)
  - glTranslated(tx,ty,tz) (f = double)
- glRotate*(θ,vx,vy,vz) (vx,vy,vz) define un eje de rotación que pasa por el origen
- glScale*(sx,sy,sz)

OpenGl .. ¿qué provee?

- Recordar que hemos visto
  - glMatrixMode(GL_PROJECTION)
- En este caso nos debemos referir a GL_MODELVIEW
  - Esto designa que la matriz actual es de tipo modelview. Esta se usa para aplicar transformaciones a los puntos de la escena
  - Cada llamada a una función de transformación genera una matrix que es multiplicada por la matriz actual
  - Para asignar valores a la matriz actual usamos:
    - glLoadIdentity();
    - glLoadMatrix*(elements16) (arreglo elements[16])

OpenGl .. ¿qué provee?

- Ejemplo:

```
glMatrixMode(GL_MODELVIEW)
GLfloat elems[16];
GLint k;
for(k=0; k<16, k++)
    elems[k] = (float) k;
glLoadMatrixf(elems);
```

OpenGl .. ¿qué provee?

- Concatenación de la matriz actual con una específica

  glMultMatrix*(otherElements16)

  - Si M es la matrix actual, la nueva M es
    - M = M * M'(otherElements16)
- Ejemplo:
  - glMatrixMode(GL_MODELVIEW);
  - glLoadIdentity();
  - glMultMatrixf(elemsM2);
  - glMultMatrixf(elemsM1);
    - M = M2*M1

OpenGl .. ¿qué provee?

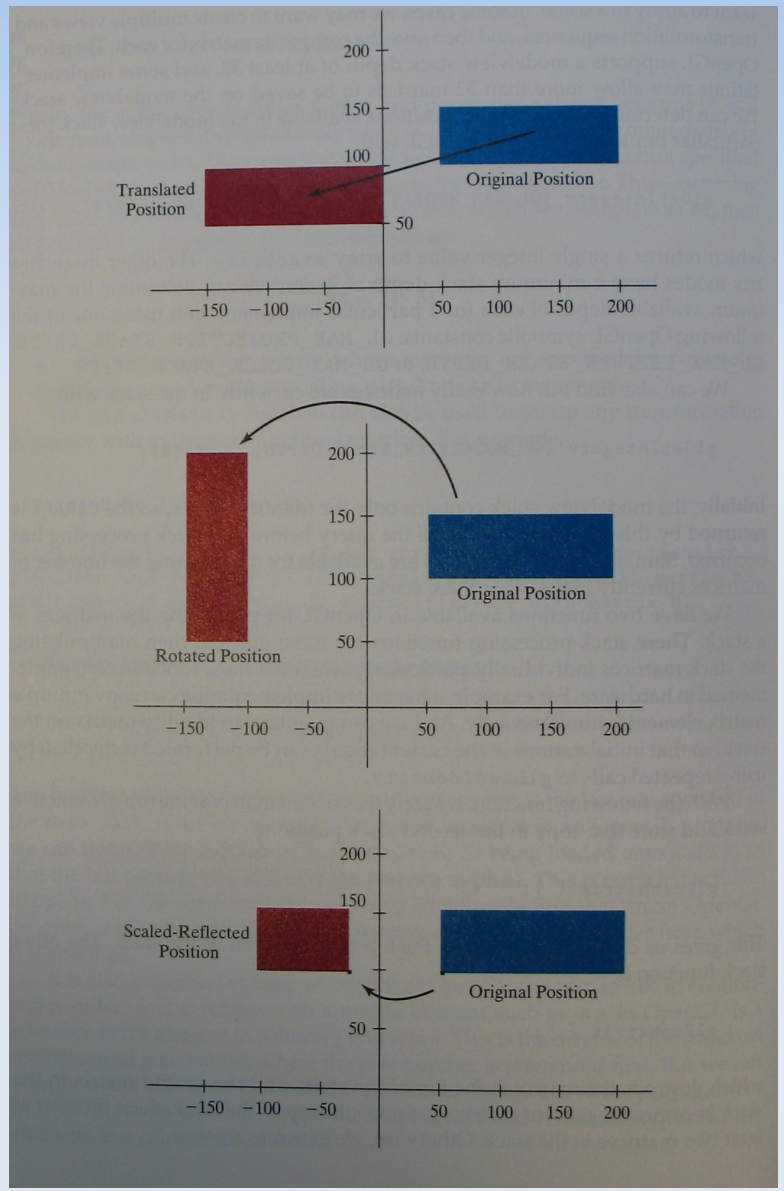
- Qué pasa si necesitamos mantener más de una matriz de transformaciones para modelar distintas escenas?
  - Por cada modo (GL_PROJECTION,GL_MODELVIEW,etc) que seleccionamos, OpenGl mantiene un “stack”. En el tope esta la matriz actual
  - El tamaño predefinido es 32
- Para determinar el tamaño máximo
  - glGetIntegerv(GL_MAX_MODELVIEW_STACK_DEPTH, stackSize);
  - glGetIntegerv(GL_MODELVIEW_STACK_DEPTH, numMats); // número de matrices

OpenGl .. ¿qué provee?

- glPushMatrix(): se copia la matriz actual en el tope del satck activo y deja la que estaba antes en el tope en la segunda posición (dos matrices iguales en las primeras dos posiciones del Stack)
- glPopMatrix(): elimina la que está en el tope. (Error si no hay menos de dos matrices)

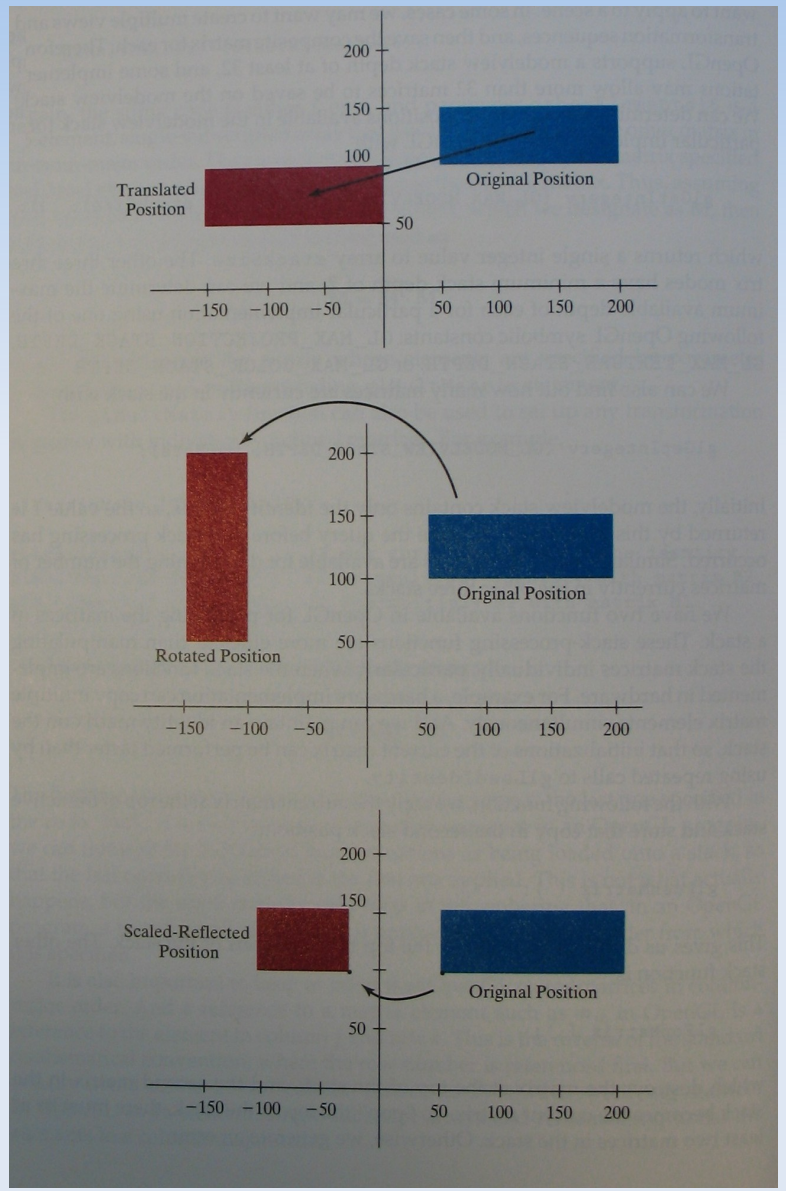
Ejemplos programación de transformaciones geométricas

- glMatrixMode(GL_MODELVIEW);
- glColor3f(0.0, 0.0, 1.0);
- glRecti(50, 100, 200, 150); // Dibuja en azul
- glColor3f(1.0, 0.0, 0.0);
- glTranslatef (-200.0, -50.0, -0.0); // traslada
- glRecti(50, 100, 200, 150); // Dibuja en rojo
- glLoadIdentity();
- glRotatef (90, 0.0, 0.0, 1.0); // Rota
- glRecti(50, 100, 200, 150); // Dibuja en rojo
- glLoadIdentity();
- glScalef (-0.5, 1.0, 1.0); // Escala
- glRecti(50, 100, 200, 150); // Dibuja en rojo



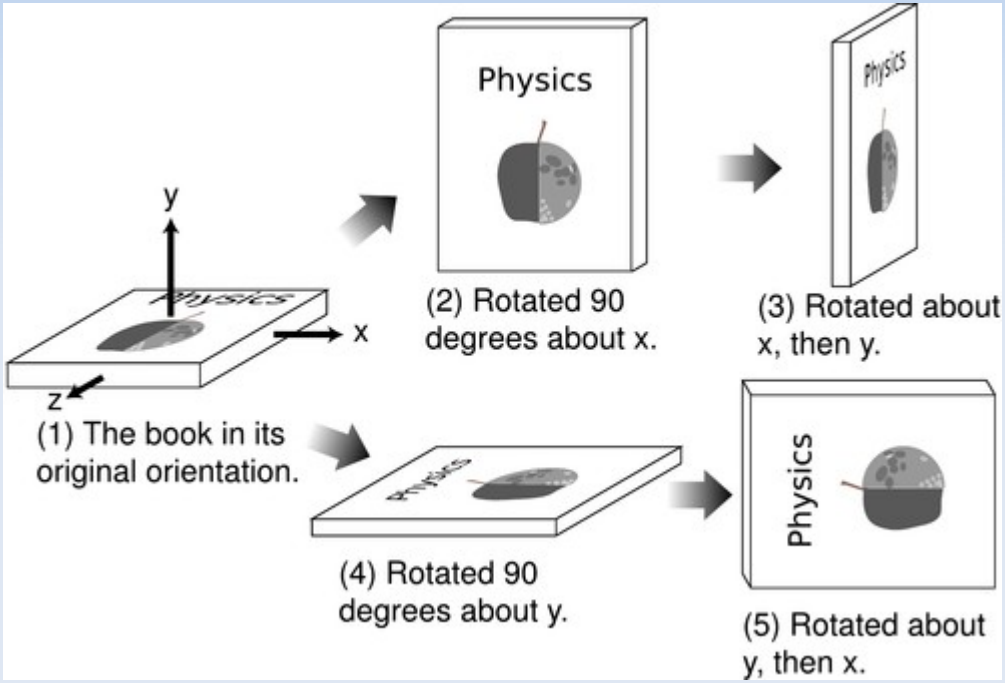
Ejemplos programación de transformaciones geométricas (con stack)

- glMatrixMode(GL_MODELVIEW);
- glColor3f(0.0, 0.0, 1.0);
- glRecti(50, 100, 200, 150); // Dibuja en azul
- GlPushMatrix(); // Copia identidad en el tope
- glColor3f(1.0, 0.0, 0.0);
- glTranslatef (-200.0, -50.0, -0.0); // traslada
- glRecti(50, 100, 200, 150); // Dibuja en rojo
- GlPopMatrix(); // elimina matriz de rotacion
- GlPushMatrix();// copia identidad en el tope
- glRotatef (90, 0.0, 0.0, 10); // Rota
- glRecti(50, 100, 200, 150); // Dibuja en rojo
- GlPopMatrix(); //elimina matriz de rotacion
- glScalef (-0.5, 1.0, 10); // Escala
- glRecti(50, 100, 200, 150); // Dibuja en rojo



OpenGl Transformaciones geométricas

- Qué transformaciones no son conmutativas en 3D que si lo eran en 2D?
  - Rotaciones



- Propuesto: Programar una rotación en torno a un eje arbitrario

Próxima clase

- 2-D y 3D viewing
  - ¿Desde donde miro la escena?
  - ¿Cómo la proyecto?