

COMP 371 -- Winter 2012 (Quiz #1 Review)

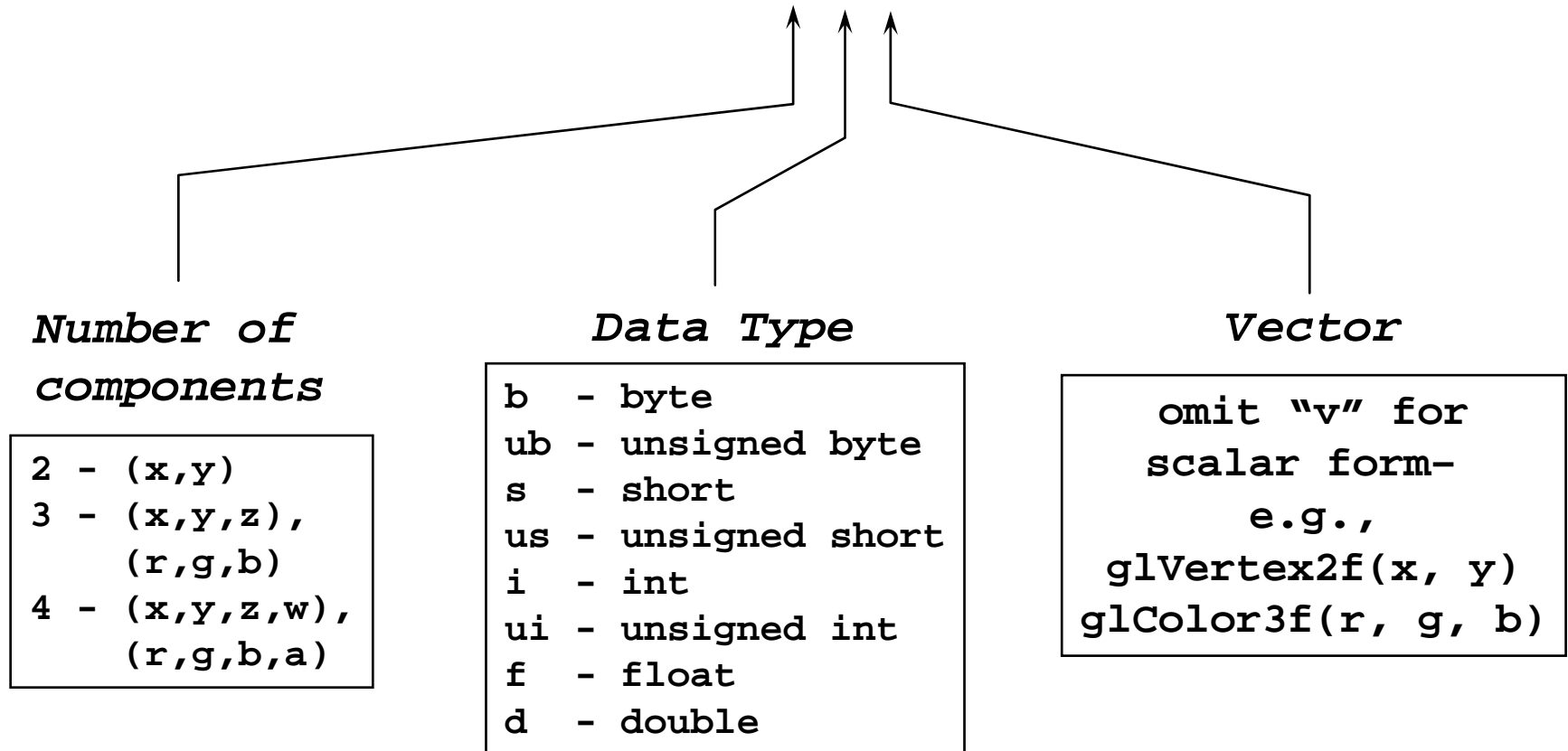
Computer Graphics

- OpenGL Basics
- 2D Graphics Algorithms
- 2D/3D Graphics Concepts
- 2D Transformations
- 3D Transformations

OpenGL Command Formats

`glVertex3fv(v)`

`glColor3fv(v)`



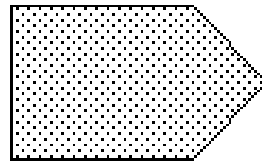
Specifying Object Vertices

- Every object is specified by vertices: `glVertex3f (2.0, 4.1, 6.0);`
 // specifies a vertex at the x, y, z coordinate (2.0, 4.1, 6.0).
 // The “3f” means 3 floating point coordinates.
 - Other examples:
`glVertex2i (4, 5);` // 2 integers for x and y. z = 0.
`glVertex3fv (vector);` // float vector[3] = {5.0, 3.2, 5.0};
- Current color affects any vertices
 - `glColor3f (0.0, 0.5, 1.0);`
 // no Red, half-intensity Green, full-intensity Blue
- Vertices are specified only between `glBegin(mode)` and `glEnd()`, usually in a counter-clockwise order for polygons.

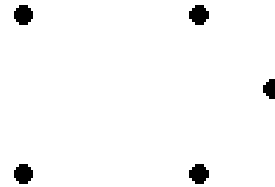

```
glBegin (GL_TRIANGLES);
  glVertex2i (0, 0);
  glVertex2i (2, 0);
  glVertex2i (1, 1);
glEnd();
```

Drawing Filled Polygons

```
glBegin( GL_POLYGON );  
    glVertex2f( 0.0, 0.0 );  
    glVertex2f( 0.0, 3.0 );  
    glVertex2f( 3.0, 3.0 );  
    glVertex2f( 4.0, 1.5 );  
    glVertex2f( 3.0, 0.0 );  
glEnd( );
```



GL_POLYGON

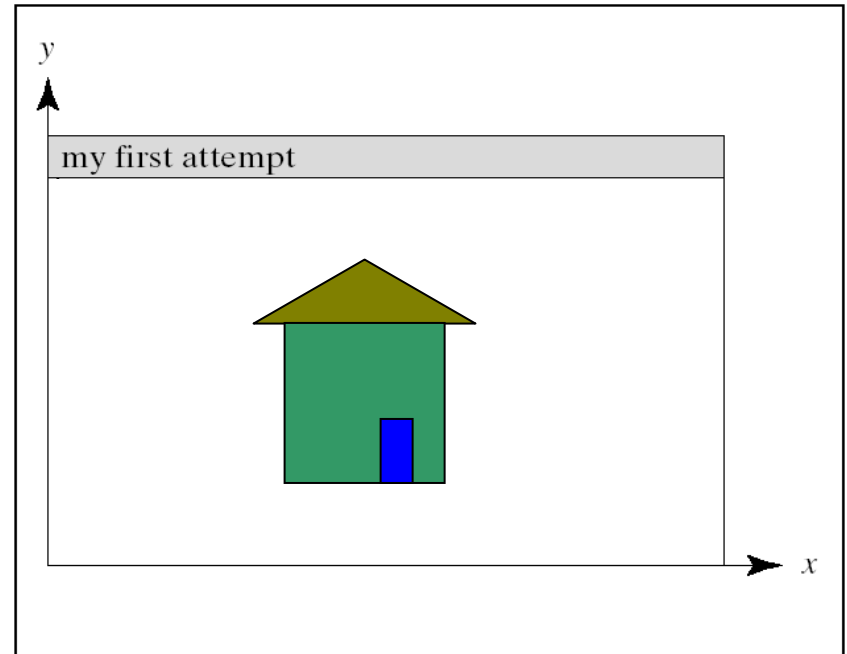


GL_POINTS

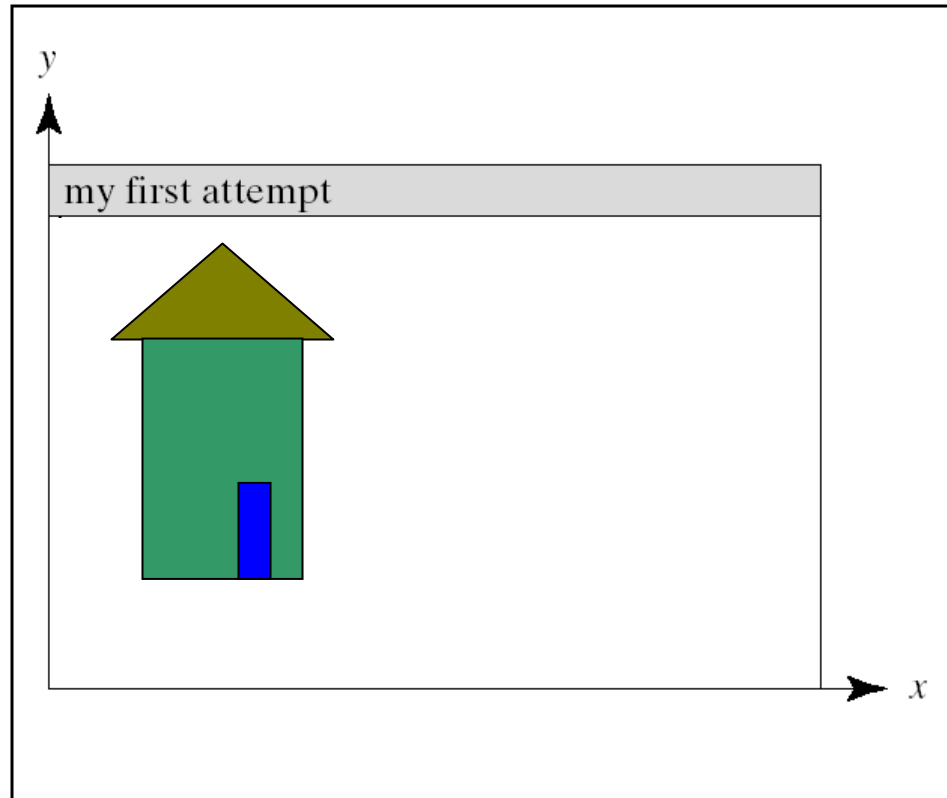
Drawing Example:

- `glColor()`: Range is $[0, 1]$ for each color channel
- `glRect(x1, y1, x2, y2)`
specifying opposite corners of rectangle is equivalent to `GL_POLYGON` with four vertices listed (i.e., filled)
- `glRect(x1, y1, x2, y2)` is equivalent to:

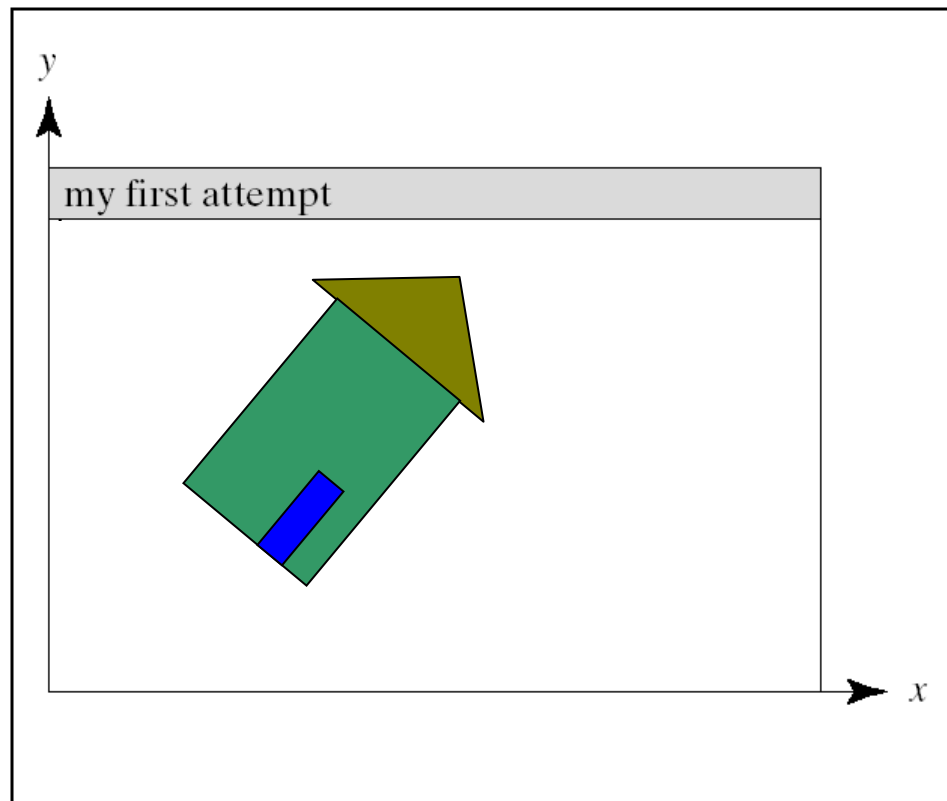
```
glBegin(GL_POLYGON)
  glVertex2(x1, y1);
  glVertex2(x2, y1);
  glVertex2(x2, y2);
  glVertex2(x1, y2);
glEnd();
```



Drawing Example: Translation and Scaling



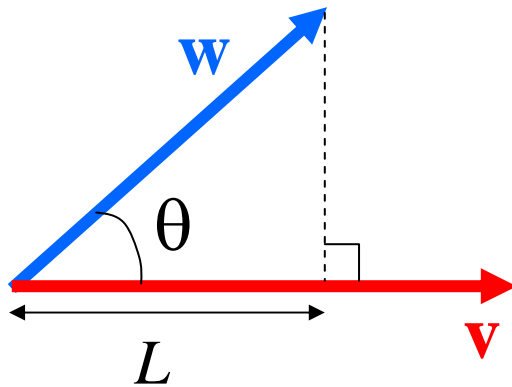
Drawing Example: Rotation



Inner (dot) product

- Defined for vectors:

$$\langle \mathbf{v}, \mathbf{w} \rangle = \|\mathbf{v}\| \cdot \|\mathbf{w}\| \cdot \cos \theta$$



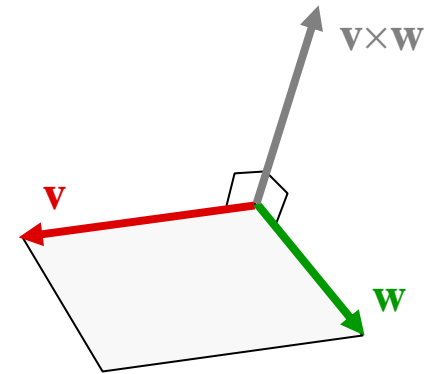
$$\cos \theta = \frac{L}{\|\mathbf{w}\|}$$

$$L = \frac{\langle \mathbf{v}, \mathbf{w} \rangle}{\|\mathbf{v}\|}$$

Projection of \mathbf{W} onto \mathbf{V}

Cross product

- $\mathbf{v} \times \mathbf{w}$ is a new vector:
 - $\|\mathbf{v} \times \mathbf{w}\| = \|\mathbf{v}\| \|\mathbf{w}\| |\sin \angle(\mathbf{v}, \mathbf{w})|$
 - $(\mathbf{v} \times \mathbf{w}) \perp \mathbf{v}$, $(\mathbf{v} \times \mathbf{w}) \perp \mathbf{w}$
 - \mathbf{v} , \mathbf{w} , $\mathbf{v} \times \mathbf{w}$ is a right-hand system



- In matrix form in 3D:

$$\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)^T, \quad \mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)^T$$

$$\mathbf{v} \times \mathbf{w} = \det \begin{pmatrix} \hat{i} & \hat{j} & \hat{k} \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 \end{pmatrix} = (\mathbf{v}_2 \mathbf{w}_3 - \mathbf{v}_3 \mathbf{w}_2, \mathbf{w}_1 \mathbf{v}_3 - \mathbf{w}_3 \mathbf{v}_1, \mathbf{v}_1 \mathbf{w}_2 - \mathbf{v}_2 \mathbf{w}_1)$$

2-D Transformations: Summary

2-D Translation
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

2-D Rotation
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2-D Scaling
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2-D Shear (horizontal)
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2-D Reflection (vertical)
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Homogeneous coordinates

- Add an extra coordinate, W , to a point.
 - $P(x,y) \rightarrow P(x_h, y_h, W) = P(xW, yW, W)$.
- Two sets of homogeneous coordinates represent the same point if they are a multiple of each other.
 - $(2,5,3)$ and $(4,10,6)$ represent the same point.
- At least one component must be non-zero $\Rightarrow (0,0,0)$ is not defined.
- If $W \neq 0$, divide by it to get Cartesian coordinates of homogeneous point $(x_h/W, y_h/W, 1)$.
- If $W=0$, point is said to be at infinity.

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x_h \\ y_h \\ h \end{bmatrix} = \begin{bmatrix} x = \frac{x_h}{h} \\ y = \frac{y_h}{h} \\ \frac{h}{h} \end{bmatrix} \text{ Ex. } \begin{bmatrix} 4 \\ 2 \end{bmatrix} \rightarrow \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}, \text{ Ex. } \begin{bmatrix} 3 \\ 7 \end{bmatrix} \rightarrow \begin{bmatrix} 6 \\ 14 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 = \frac{6}{2} \\ 7 = \frac{14}{2} \\ \frac{2}{2} \end{bmatrix}$$

Translation in Homogeneous coordinates

Now, we can write the translation as the multiplication by specially designed matrix:

Translation:

$$T = \begin{bmatrix} \mathbf{1} & \mathbf{0} & d_1 \\ \mathbf{0} & \mathbf{1} & d_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad T \cdot \vec{x} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & d_1 \\ \mathbf{0} & \mathbf{1} & d_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} x_1 + d_1 \\ x_2 + d_2 \\ \mathbf{1} \end{bmatrix} = \vec{x} + \vec{d}$$

$$\vec{x} \equiv \begin{bmatrix} \mathbf{1} \\ \mathbf{2} \\ \mathbf{1} \end{bmatrix} \quad T \cdot \vec{x} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \mathbf{2} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{2} \\ \mathbf{3} \\ \mathbf{1} \end{bmatrix} = \vec{x} + \vec{d}$$

Sequential Translations in Homogeneous Coordinates

We can check that the matrix representing two sequential translations can be written as the multiplication of their matrices.

Two translations

$$T_1 \cdot T_2 = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{d}_1 \\ \mathbf{0} & \mathbf{1} & \mathbf{d}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{g}_1 \\ \mathbf{0} & \mathbf{1} & \mathbf{g}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{d}_1 + \mathbf{g}_1 \\ \mathbf{0} & \mathbf{1} & \mathbf{d}_2 + \mathbf{g}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} = \vec{\mathbf{d}} + \vec{\mathbf{g}}$$

1. $T(0,0) = I$

2. $T(s_x, s_y) \cdot T(t_x, t_y) = T(s_x + t_x, s_y + t_y)$

3. $T(s_x, s_y) \cdot T(t_x, t_y) = T(t_x, t_y) \cdot T(s_x, s_y)$

4. $T^{-1}(s_x, s_y) = T(-s_x, -s_y)$

Scaling in Homogeneous coordinates

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

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

Scaling matrix looks similar to what it was for ordinary coordinates:

Scaling

$$S = \begin{bmatrix} S_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & S_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad S \cdot \vec{x} = \begin{bmatrix} S_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & S_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} S_1 x_1 \\ S_2 x_2 \\ \mathbf{1} \end{bmatrix}$$

$$S \cdot \vec{x} = \begin{bmatrix} 2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0.5 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} 2 \\ 0.5 \\ \mathbf{1} \end{bmatrix}$$

What is the Matrix for Scaling $0.1x_1$ and $10x_2$?

Several Scalings in Homogeneous coordinates

The matrix of two successful scalings
is the multiplication of two scaling matrices:

Two scalings

$$S_B \cdot S_A \cdot \vec{x} = \begin{bmatrix} S_1^B & 0 & 0 \\ 0 & S_2^B & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_1^A & 0 & 0 \\ 0 & S_2^A & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_1^B S_1^A & 0 & 0 \\ 0 & S_2^B S_2^A & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} S_1^B S_1^A x_1 \\ S_2^B S_2^A x_2 \\ 1 \end{bmatrix}$$

What is the Matrix for Scaling $0.1x_1$ and $10x_2$ and then $20x_1$ and $0.1x_2$?

Rotation in Homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Easy to check, that clock-wise rotation on angle θ is given by:

Rotation

$$R \cdot \vec{x} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta)x_1 - \sin(\theta)x_2 \\ \sin(\theta)x_1 + \cos(\theta)x_2 \\ 1 \end{bmatrix}$$

Two successful rotations can be represented by multiplication of their matrices:

Two Rotations

$$R(\theta_2) \cdot R(\theta_1) = \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 \\ \sin \theta_2 & \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R(\theta_1 + \theta_2)$$

Homogeneous form of rotation.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For rotation matrices,

$$R^{-1}(\theta) = R(-\theta).$$

Rotation matrices are orthogonal, i.e.:

$$R^{-1}(\theta) = R^T(\theta)$$

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R^T(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(-\theta) = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Final Transformations - Compare Equations

$$T(t_x, t_y) = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P = T(t_x, t_y) + P \rightarrow P = T(t_x, t_y) \bullet P$$

$$S(s_x, s_y) = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P = S(s_x, s_y) \bullet P \rightarrow P = S(s_x, s_y) \bullet P$$

$$R(\theta) = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P = R(\theta) \bullet P \rightarrow P = R(\theta) \bullet P$$

Combining Transformations

$$P' = A \bullet P, P'' = B \bullet P' \rightarrow P'' = B \bullet A \bullet P$$

$$P(3,4), T(0.4641, -2), R(60)$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 1 & 0 & t_x \\ \sin\theta & \cos\theta & 0 & 0 & 1 & t_y \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \cos\theta - t_y \sin\theta \\ \sin\theta & \cos\theta & t_x \sin\theta + t_y \cos\theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Alternatively:

$$\begin{bmatrix} x''' \\ y''' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x''' \\ y''' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Therefore, $A \bullet B \bullet P \neq B \bullet A \bullet P$

Rotation around arbitrary point

How to write the rotation around a point? $\vec{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$

Bring the point p to the origin; make a rotation, bring it back:

$$T(\vec{p}) \cdot R \cdot T(-\vec{p}) \cdot \vec{x} = \begin{array}{c} \text{Bring } p \\ \text{back} \end{array} \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{array}{c} \text{Rotation} \end{array} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{array}{c} \text{Bring } p \text{ to} \\ \text{the origin} \end{array} \begin{bmatrix} 1 & 0 & -p_1 \\ 0 & 1 & -p_2 \\ 0 & 0 & 1 \end{bmatrix}$$

... the same procedure for scaling:

$$T(\vec{p}) \cdot S \cdot T(-\vec{p}) \cdot \vec{x} = \begin{array}{c} \text{Bring } p \\ \text{back} \end{array} \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{array}{c} \text{Scaling} \end{array} \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{array}{c} \text{Bring } p \text{ to} \\ \text{the origin} \end{array} \begin{bmatrix} 1 & 0 & -p_1 \\ 0 & 1 & -p_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Example 1. Series of transformations

Write down the matrix for: Rotation $\theta=90^\circ$ around $p=(2,5)$, translation $(-2,2)$, uniform scaling ($s=2$) around $p=(-1,1)$.

$$\begin{array}{ccc}
 \text{Translation} & \text{Scaling} & \text{Rotation} \\
 T = \begin{bmatrix} 1 & 0 & d_1 \\ 0 & 1 & d_2 \\ 0 & 0 & 1 \end{bmatrix} & S = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} & R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
 \end{array}$$

$$(T(-1,1) \cdot S(2,2) \cdot T(1,-1)) \cdot T(-2,2) \cdot (T(2,5) \cdot R(90) \cdot T(-2,-5))$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot T(-2,2)(T(2,5) \cdot R \cdot T(-2,-5)) = \dots$$

$$\dots = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 7 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformations in OpenGL

Given: a unit square centered at the origin

Write: OpenGL code that

- 1) translates by one unit in x, and then
- 2) rotates by $\theta=90^\circ$ about the z-axis

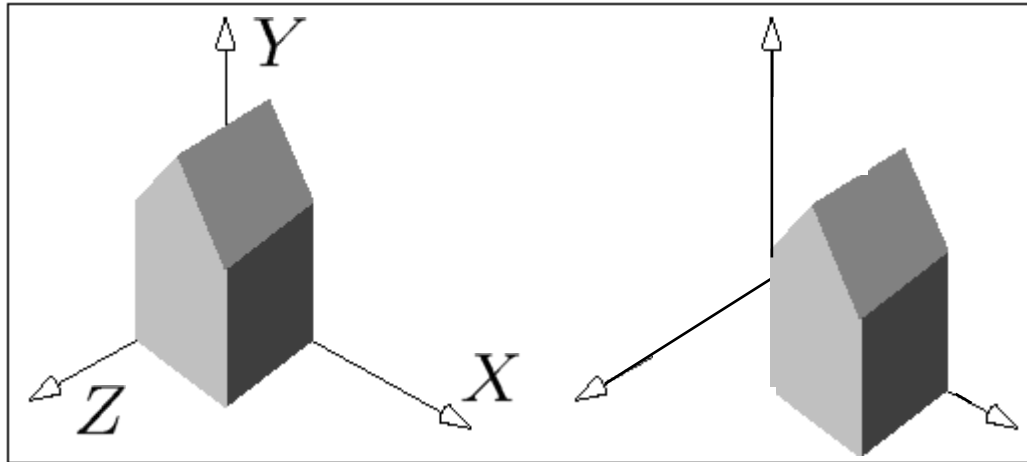
Solution:

```
glRotatef(90.0, 0.0, 0.0, 1.0);  
glTranslatef(1.0, 0.0, 0.0);  
drawsquare();  
glFlush();
```

The point: specify OpenGL commands in opposite order of occurrence.

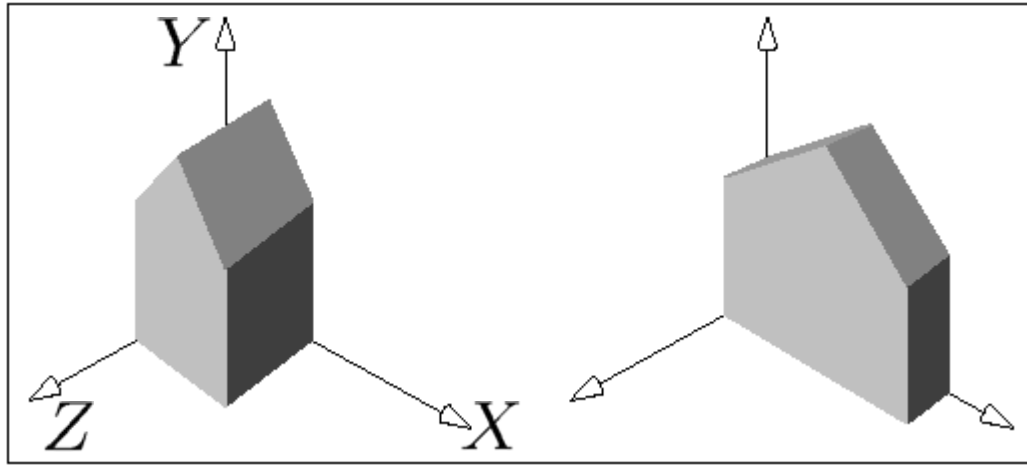
In matrix notation: $V' = I R T V$

3-D Translations



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3-D Scaling



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3-D Uniform Scaling

Note that uniform scaling can also be accomplished using homogeneous coordinate properties by manipulating scale coordinate

$$\begin{pmatrix} x \\ y \\ z \\ 1/s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/s \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

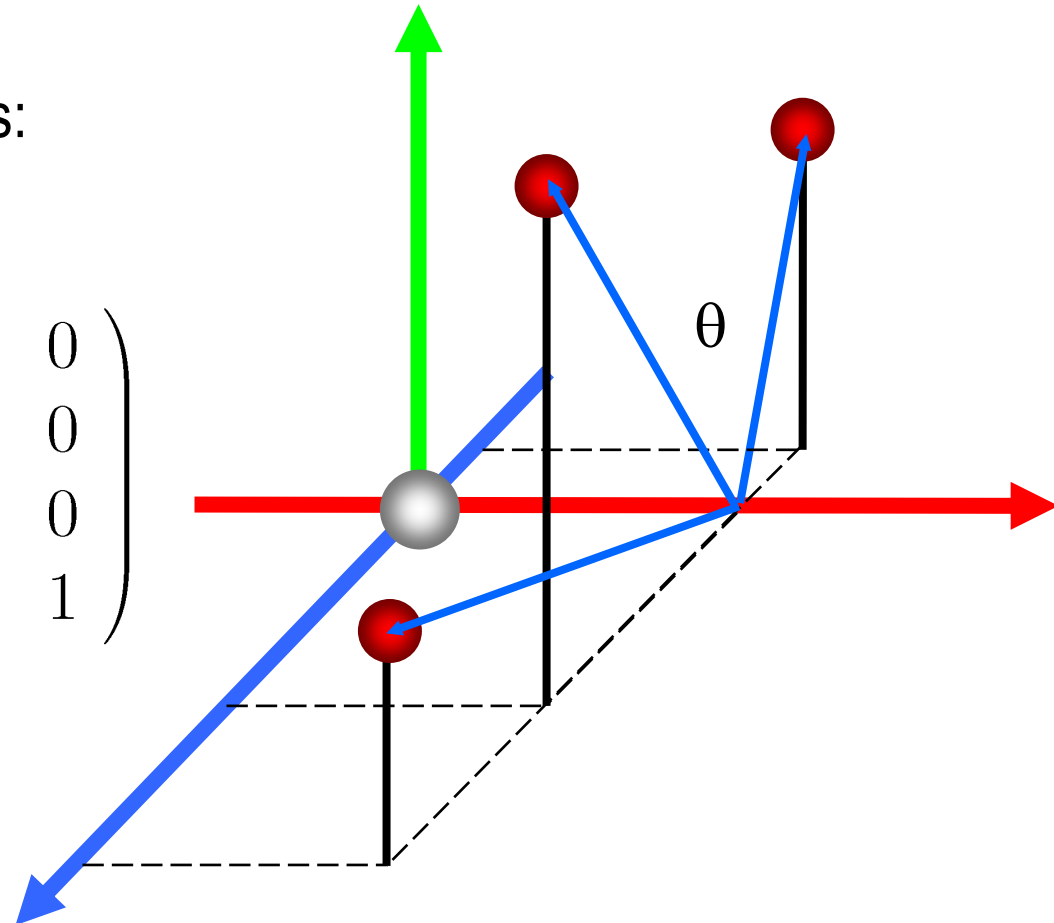
↑
must normalize to 1

“Homogeneous” scaling matrix

3-D Rotation Matrices

- Similar form to 2-D rotation matrices, but with coordinate corresponding to rotation axis held constant
- E.g., a rotation about the X axis:

$$\mathbf{R}_X(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

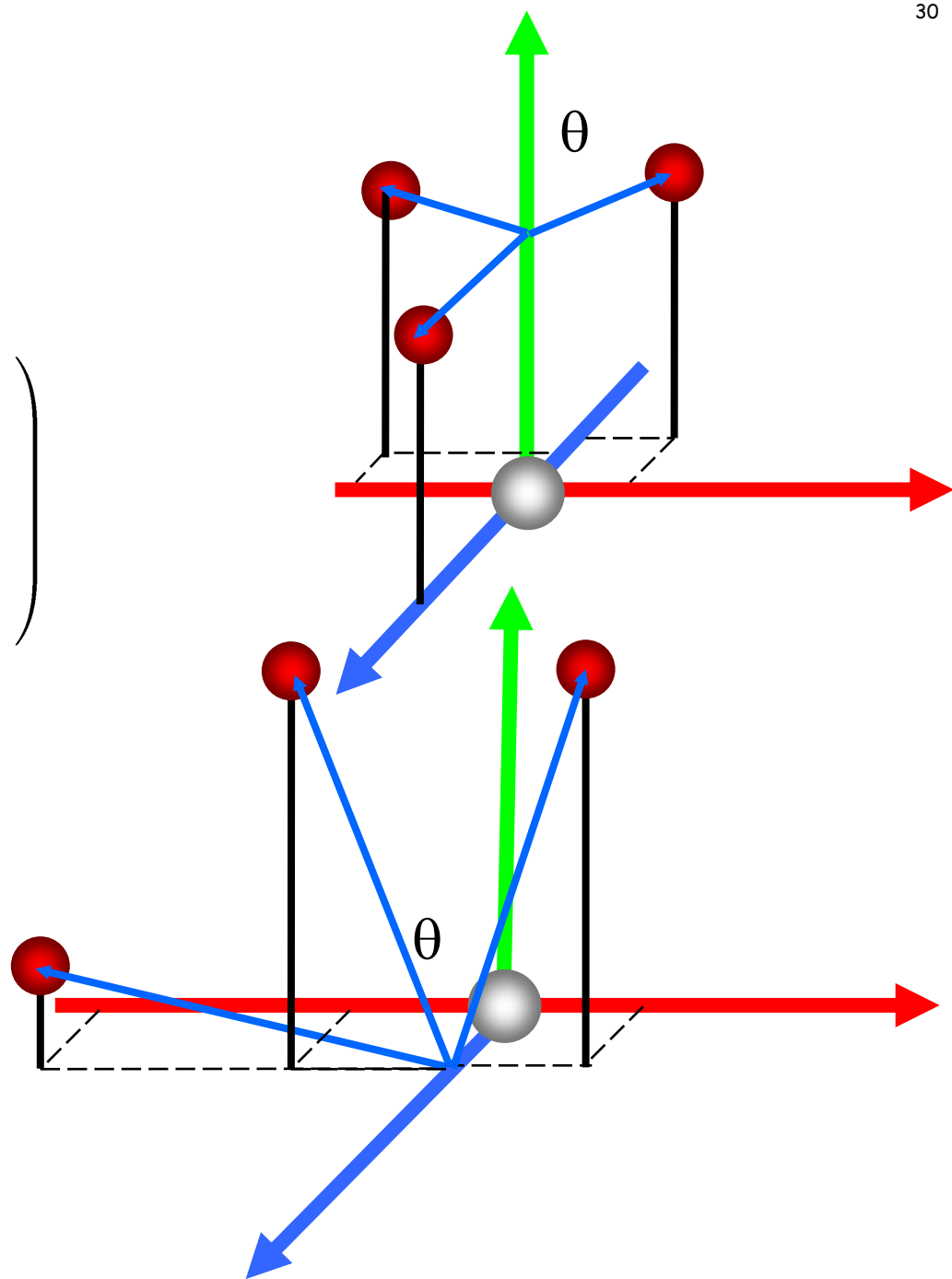


3-D Rotation Matrices

- Similarly:

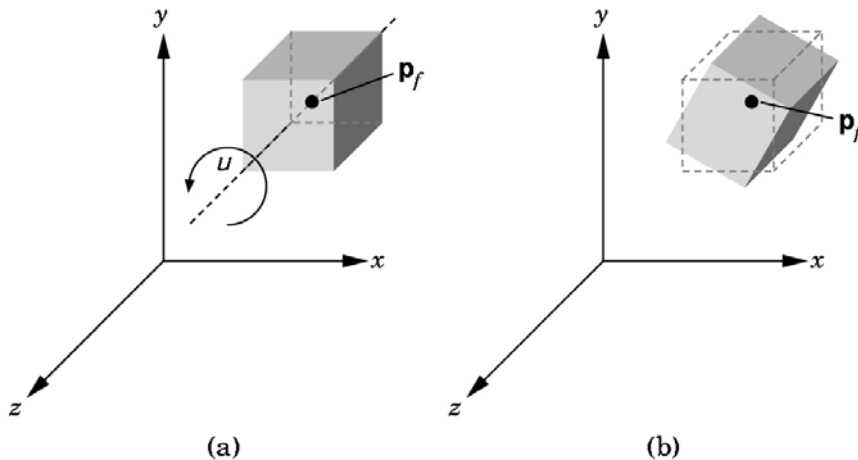
$$\mathbf{R}_Y(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_Z(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Rotation Around Vector $(0,0,1)$ About an Fixed Point (a,b,c)

- Strategy:
 - translate the cube to the origin,
 - rotate it around the z-axis about the origin,
 - translate it back again at $p_f = (a,b,c)$



```
glMatrixMode(GL_MODELVIEW)
glLoadIdentity();
glTranslatef(a,b,c);
glRotatef( $\theta$ ,0,0,1);
glTranslatef(-a,-b,-c);
DrawCube();
```

$$M = T(p_f)R_z(\theta)T(-p_f)$$

Rotating with quaternions

- Represent P as a vector $[0, P]$.
- Rotating about axis $V = (u_x, u_y, u_z)$, $\|V\| = 1$, by angle θ :

$$R_{n,\theta}(P) = q \cdot [0, P] \cdot q^{-1}$$

$$q = (\cos(\theta/2), \sin(\theta/2) * V)$$

$$= (w, a, b, c)$$

Conversion From Quaternions → Quaternion to Matrix

Write out $[0, P'] = q[0, P]q^{-1}$ and expand it like a matrix equation for the i, j , and k terms

$$\text{Matrix} = \begin{bmatrix} 1 - 2b^2 - 2c^2 & 2ab - 2cw & 2ac + 2bw \\ 2ab + 2cw & 1 - 2a^2 - 2c^2 & 2bc - 2aw \\ 2ac - 2bw & 2bc + 2aw & 1 - 2a^2 - 2b^2 \end{bmatrix}$$

Rotation About a Fixed Point in OpenGL

```
glMatrixMode(GL_MODELVIEW)  
glLoadIdentity();
```

- *Move the fixed point back again :*

```
glTranslatef(4.0, 5.0, 6.0);
```

- *Rotate about the y-axis :*

```
glRotatef(45.0, 0.0, 1.0, 0.0);
```

- *Move the fixed point to the origin :*

```
glTranslatef(-4.0, -5.0, -6.0);
```

Example: y -axis Rotation about $p=(4,5,6)$ with $\theta = 45$

$$R_y(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -6 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 4 \\ 0 & 1 & 0 & 5 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -6 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & (4\sqrt{2} - 10)/\sqrt{2} \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & (6\sqrt{2} - 2)/\sqrt{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: y-axis Rotation about $p=(\sqrt{18},0,2)$ with $\theta = 45$

$$R_y(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

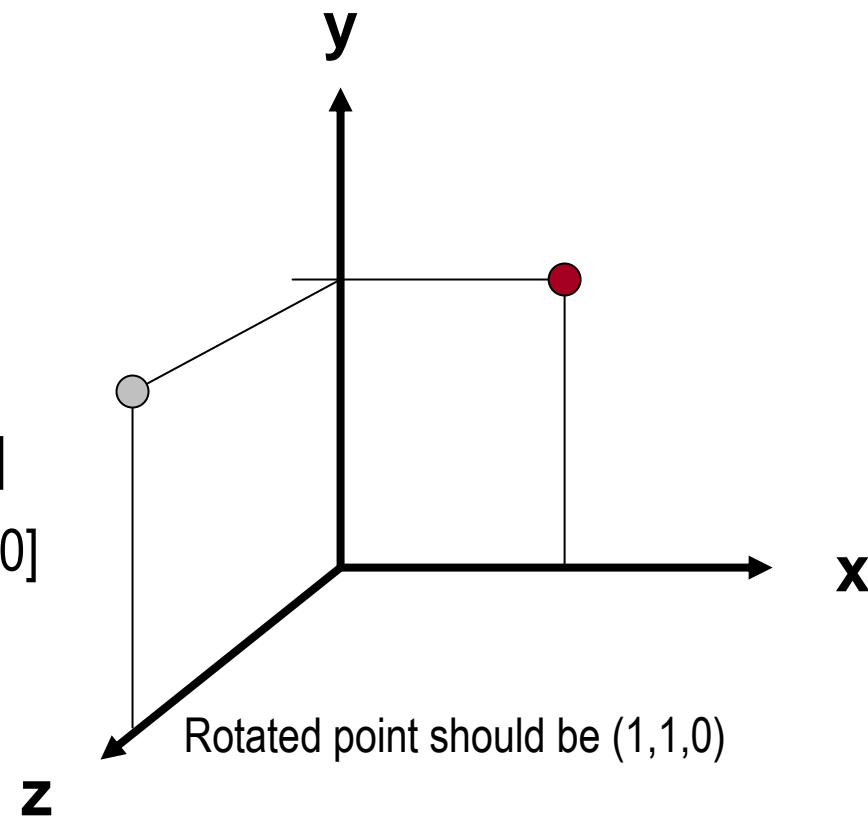
$$\begin{pmatrix} 1 & 0 & 0 & \sqrt{18} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\sqrt{18} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & \sqrt{18} \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\sqrt{18} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 2\sqrt{2} - 3 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 5 - \sqrt{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example: rotate $P = (0, 1, 1)$ 90 degrees about the vertical y -axis.

- First, compute q
 - $q = [\cos(90/2), \sin(90/2)[0, 1, 0]]$
 $= [\cos 45, 0, \sin 45, 0]$
 $= [\sqrt{2}/2, 0, \sqrt{2}/2, 0]$
- Next compute q^{-1}
 - $q^{-1} = [\sqrt{2}/2, 0, -\sqrt{2}/2, 0]$ *Why?*
- $q[0, P] = [-\sqrt{2}/2, \sqrt{2}/2, \sqrt{2}/2, \sqrt{2}/2]$
- Compute $q[0, P]q^{-1}$
 $= [\sqrt{2}/2, 0, \sqrt{2}/2, 0][0, 0, 1, 1][\sqrt{2}/2, 0, -\sqrt{2}/2, 0]$
 $= [-\sqrt{2}/2, \sqrt{2}/2, \sqrt{2}/2, \sqrt{2}/2][\sqrt{2}/2, 0, -\sqrt{2}/2, 0]$
 $= [0, 1, 1, 0]$

Transformed point is $[1, 1, 0]$.



3D Transformations: summary

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation

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

Scaling

Rotating $R(\theta) =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

About **x**-axis

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

About **y**-axis

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

About **z**-axis

OpenGL Matrix Stack Commands

- Top of stack matrix is called the *current matrix*
- **glPushMatrix ()**
 - copy the current matrix and pushes it
- **glPopMatrix ()**
 - pop the current matrix

OpenGL Matrix Stacks

- Why multiple matrix stacks?
 - certain techniques are done in different spaces
- glMatrixMode (mode)**
- choose which stack to manipulate
 - **GL_MODELVIEW**
 - **GL_PROJECTION**

Sample Questions

Choose the INCORRECT statement from the following:

- (A) The techniques used in an Optical Character Recognizer (OCR) Reader, which takes as input a scanned document and outputs an ASCII version of the document, are primarily addressed in the field of Pattern Recognition.
- (B) The techniques used in an interactive walkthrough, for synthesizing realistic visuals of virtual worlds, are primarily addressed in the field of Computer Vision.
- (C) The techniques used in finger-print identification, for identifying lines that make up the ridges in a finger print image, are primarily addressed in the field of Image Processing.
- (D) The techniques used for generating engineering drawings of objects, are primarily addressed in the field of Computer Aided Design.