Transformations
Translation, Rotation, Scale
Composite transformations

Homogeneous Coordinates

- Homogeneous coordinates are key to all computer graphics systems
- Hardware pipeline all work with 4 dimensional representations
- All standard transformations (rotation, translation, scaling) can be implemented by matrix multiplications with 4 x 4 matrices

A Single Representation

With these rules, we can keep track of the difference:

$$v = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 = [\alpha_1\ \alpha_2\ \alpha_3\ 0\ ]\ [v_1\ v_2\ v_3\ \ P_0]^T$$

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 = [\beta_1\ \beta_2\ \beta_3\ 1\ ]\ [v_1\ v_2\ v_3\ \ P_0]^T$$

Thus we obtain a four-dimensional representation for both:

$$\mathbf{v} = [\alpha_1\ \alpha_2\ \alpha_3\ 0\ ]^T$$

$$\mathbf{p} = [\beta_1\ \beta_2\ \beta_3\ 1\ ]^T$$

3

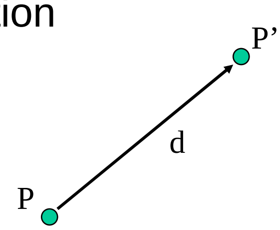
Affine Transformations

- Line preserving
- Characteristic of many physically important transformations
  - Rigid body transformations: translation, rotation
  - Non-rigid: Scaling, shear
- Importance in graphics is that we need only transform vertices (points) of line segments and polygons, then system draws between the transformed points

4

Translation

- Move (translate, displace) a point to a new location

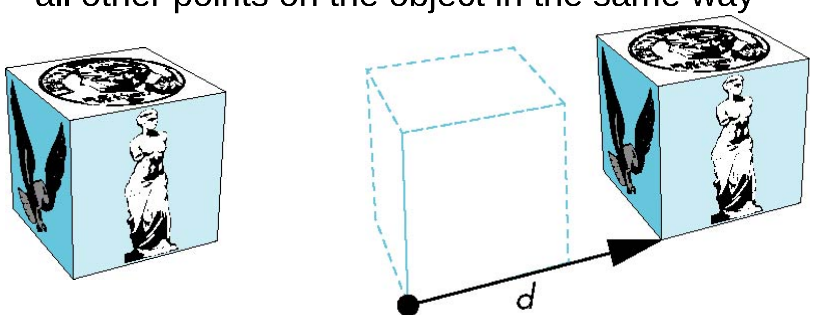


- Displacement determined by a vector d
  - Three degrees of freedom
  - P'=P+d

5

Moving objects

When we move a point on an object to a new location, to preserve the object, we must move all other points on the object in the same way



object

translation: every point displaced by the same vector, d

6

Translation Using Representations

Using the homogeneous coordinate representation in some frame

$\mathbf{p}=[\ x\ y\ z\ 1]^T$

$\mathbf{p}'=[x'\ y'\ z'\ 1]^T$

$\mathbf{d}=[dx\ dy\ dz\ 0]^T$

Hence $\mathbf{p}' = \mathbf{p} + \mathbf{d}$ or

$x'=x+d_x$

$y'=y+d_y$

$z'=z+d_z$

note that this expression is in four dimensions and expresses that point = vector + point

7

Translation Matrix

We can also express translation using a 4 x 4 matrix $\mathbf{T}$ in homogeneous coordinates $\mathbf{p}'=\mathbf{T}\mathbf{p}$ where

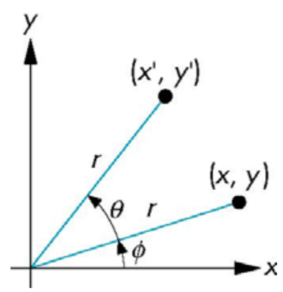
$$\mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This form is better for implementation because all affine transformations can be expressed this way and multiple transformations can be concatenated together

8

Rotation (2D)

- Consider rotation about the origin by θ degrees
  - radius stays the same, angle increases by θ



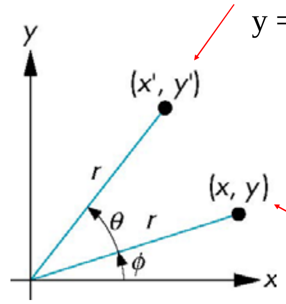
What is this rotation about the z axis?

9

Rotation (2D)

- Consider rotation about the origin by θ degrees
  - radius stays the same, angle increases by θ

$$x = r\cos(\phi + \theta)$$
$$y = r\sin(\phi + \theta)$$



$$\boxed{\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}}$$

$$x = r\cos\phi$$
$$y = r\sin\phi$$

10

Rotation about the z axis

- Rotation about z axis in three dimensions leaves all points with the same z
  - Equivalent to rotation in two dimensions in planes of constant z

$$x' = x \cos\theta - y \sin\theta$$
$$y' = x \sin\theta + y \cos\theta$$
$$z' = z$$

  - or in matrix notation (with p as a column)

$$\mathbf{p}' = \mathbf{R}_{\mathbf{z}}(\theta)\mathbf{p}$$

11

Rotation Matrix

Homogeneous Coordinates:

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

12

Rotation about x and y axes

- Same argument as for rotation about *z* axis
  - For rotation about *x* axis, *x* is unchanged
  - For rotation about *y* axis, *y* is unchanged

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

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

13

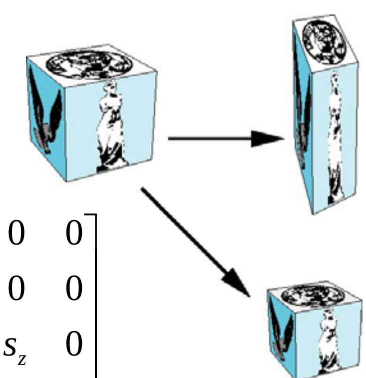
Scaling

Expand or contract along each axis (fixed point of origin)

$x' = s_x x$
$y' = s_y y$
$z' = s_z z$

$\mathbf{p}' = \mathbf{S}\mathbf{p}$

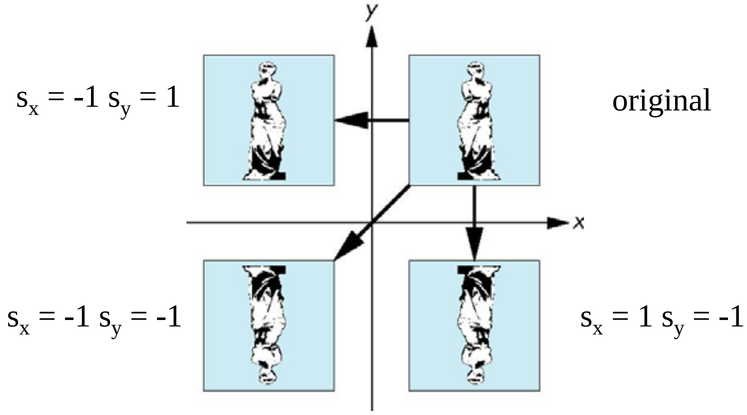
$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



14

Reflection

corresponds to negative scale factors



15

Basic transforms in OpenGL

```
glTranslatef(a,b,c);
glTranslated(a,b,c);

glScalef(a,b,c);
glScaled(a,b,c);

glRotatef(angle,x,y,z);
glRotated(angle,x,y,z);
```

16

Inverses

- Although we could compute inverse matrices by general formulas, we can also use simple geometric observations, for example:
  - Translation: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathbf{T}(-d_x, -d_y, -d_z)$
  - Rotation: $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$
    - Holds for any rotation matrix
    - Note that since $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$

    $\mathbf{R}^{-1}(\theta) = \mathbf{R}^{T}(\theta)$
  - Scaling: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$

17

Concatenation

- We can form arbitrary affine transformation matrices by multiplying together rotation, translation, and scaling matrices
- Because the same transformation is applied to many vertices, the cost of forming a matrix $\mathbf{M}=\mathbf{ABCD}$ is not significant compared to the cost of computing $\mathbf{Mp}$ for many vertices $\mathbf{p}$
- The combination of transformations must be managed with care, b/c order matters

18

Order of Transformations

- Note that matrix on the right is the first applied
- Mathematically, the following are equivalent

$$\mathbf{p}' = \mathbf{ABCp} = \mathbf{A}(\mathbf{B}(\mathbf{Cp}))$$

(but does not = **CBA p**)

- Some references use row matrices to present points. In terms of rows, we get

$$\mathbf{p}^{T\prime} = \mathbf{p}^T\mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$$

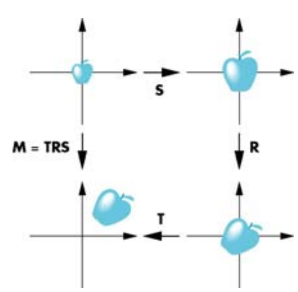
19

Order of Transformations

•In modeling, we often start with a simple object centered at the origin, oriented with the axis, and at a standard size

•We apply an *composite transformation* to its vertices to

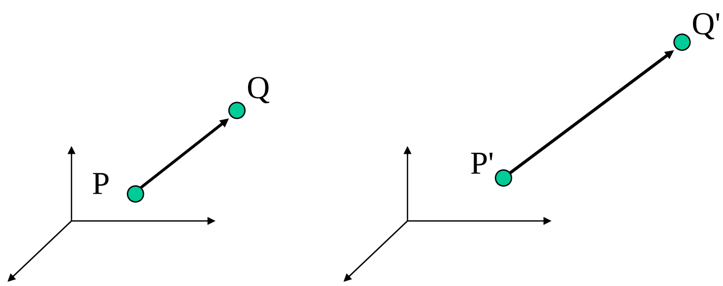
Scale
Orient
Locate



20

Composite Transformations

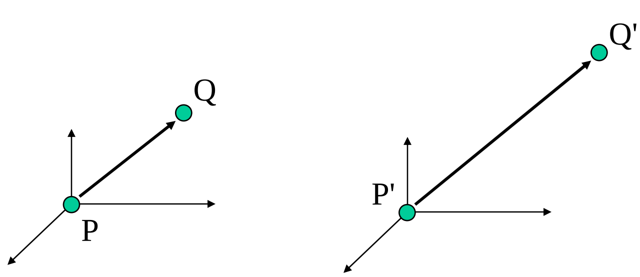
- Scaling about a fixed point
  - Applying the scale transformation also moves the object being scaled.



21

Composite Transformations

- Exception: Scaling about origin -> no movement
- Origin is a ***fixed point*** for the scale transformation
- We use composite transformations to create scale transformations with different fixed points



22

Composite Transformations

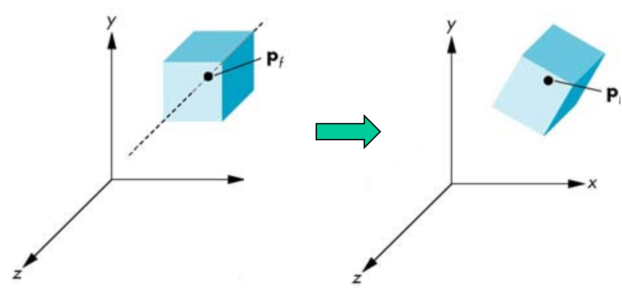
- **Fixed point scale transformation**
  - Move fixed point (px,py,pz) to origin
  - Scale by desired amount
  - Move fixed point back to original position

$$\mathbf{M} = \mathbf{T}(px, py, pz)\, \mathbf{S}(s_x, s_y, s_z)\, \mathbf{T}(-px, -py, -pz)$$

23

Composite Transformations

- Rotating about a fixed point
  - **- basic** rotation alone will rotate about origin but we want:



24

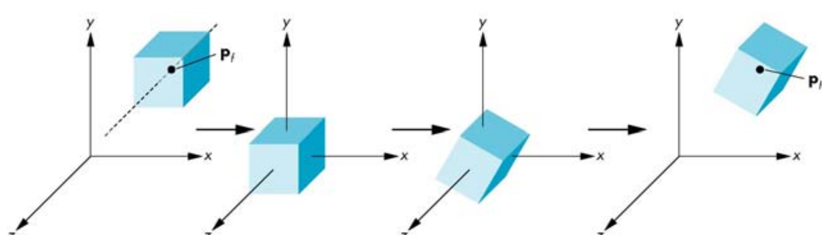
Composite Transformations

- Rotating about a fixed point
  - Move fixed point (px,py,pz) to origin
  - Rotate by desired amount
  - Move fixed point back to original position

$$\mathbf{M} = \mathbf{T}(px, py, pz)\, \mathbf{R_x}(\theta)\, \mathbf{T}(-px, -py, -pz)$$

25

Composite Transformations



26

Rotation about an arbitrary axis

Rotating about an axis by theta degrees

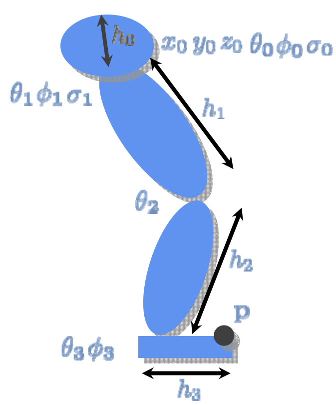
- Rotate about x to bring axis to xz plane
- Rotate about y to align axis with z -axis
- Rotate theta degrees about z
- Unrotate about y, unrotate about x

$$\mathbf{M} = \mathbf{Rx}^{-1}\ \mathbf{Ry}^{-1}\ \mathbf{Rz}(\theta)\ \mathbf{Ry}\ \mathbf{Rx}$$

- Can you determine the values of Rx and Ry?

27

Composite transformations



A series of transformations on an object can be applied as a series of matrix multiplications

$\mathbf{p}$: position in the global coordinate

$\mathbf{x}$: position in the local coordinate

$(h_3, 0, 0)$

$$\mathbf{p} = \mathbf{T}(x_0, y_0, z_0)\mathbf{R}(\theta_0)\mathbf{R}(\phi_0)\mathbf{R}(\sigma_0)\mathbf{T}(0, h_0, 0)\mathbf{R}(\theta_1)\mathbf{R}(\phi_1)\mathbf{R}(\sigma_1)\mathbf{T}(0, h_1, 0)\mathbf{R}(\theta_2)\mathbf{T}(0, h_2, 0)\mathbf{R}(\theta_3)\mathbf{R}(\phi_3)\mathbf{x}$$

28