Announcements

- Assignment 1 is due Thursday at midnight
- Turnin space:
  - /afs/andrew/scs/cs/15-462/turnin
  - You should be able to create files there but not read or remove
  - Use a directory name of the format: assignment1_jkh
- TA hours Tuesday 3-5 (Joel—cluster) and Thursday 3-5 (Michael—Wean 8121)
- Or send any of us email

- Questions on Assignment 1?

Transformations

Translation, rotation, scaling

- 2D
- 3D

Homogeneous coordinates

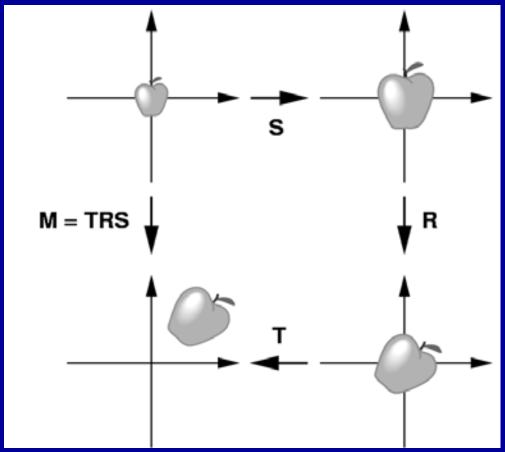
Transforming normals

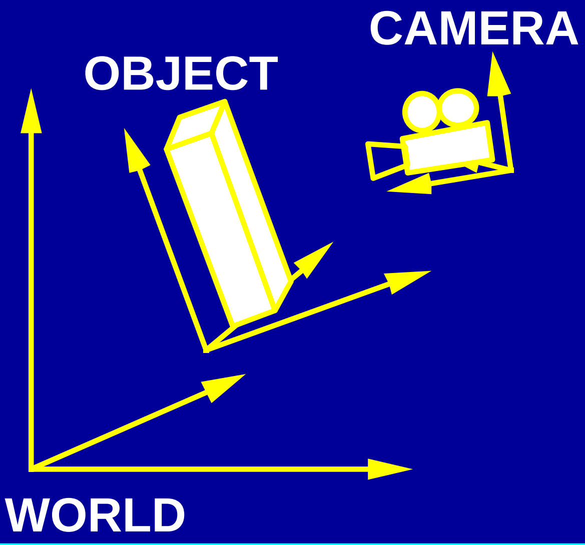
Examples

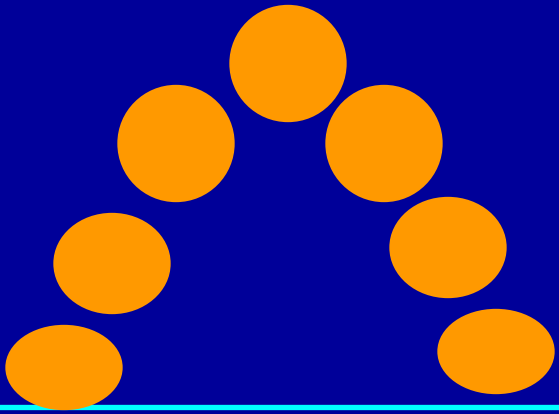
Shirley Chapter 6

Uses of Transformations

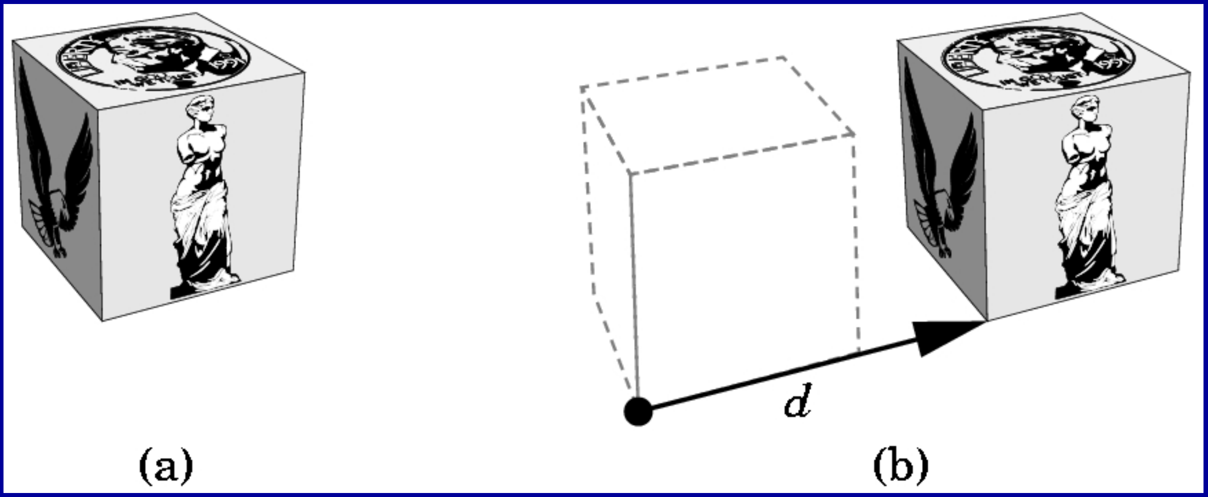
- Modeling
  - – build complex models by positioning simple components
  - – transform from object coordinates to world coordinates
- Viewing
  - – placing the virtual camera in the world
  - – specifying transformation from world coordinates to camera coordinates
- Animation
  - – vary transformations over time to create motion

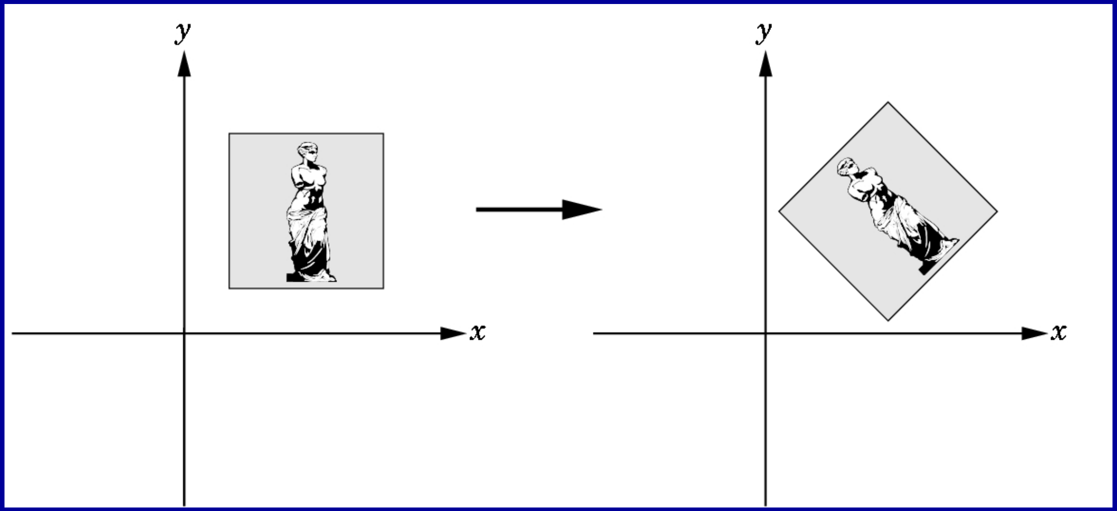


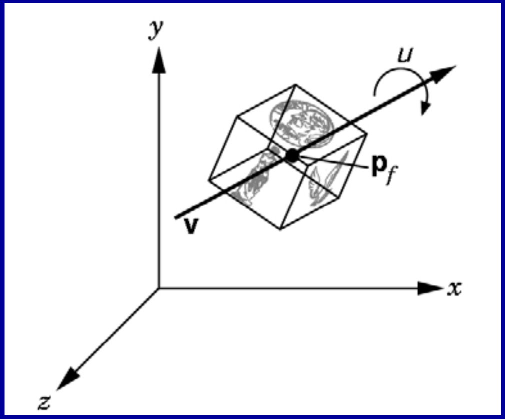




Rigid Body Transformations

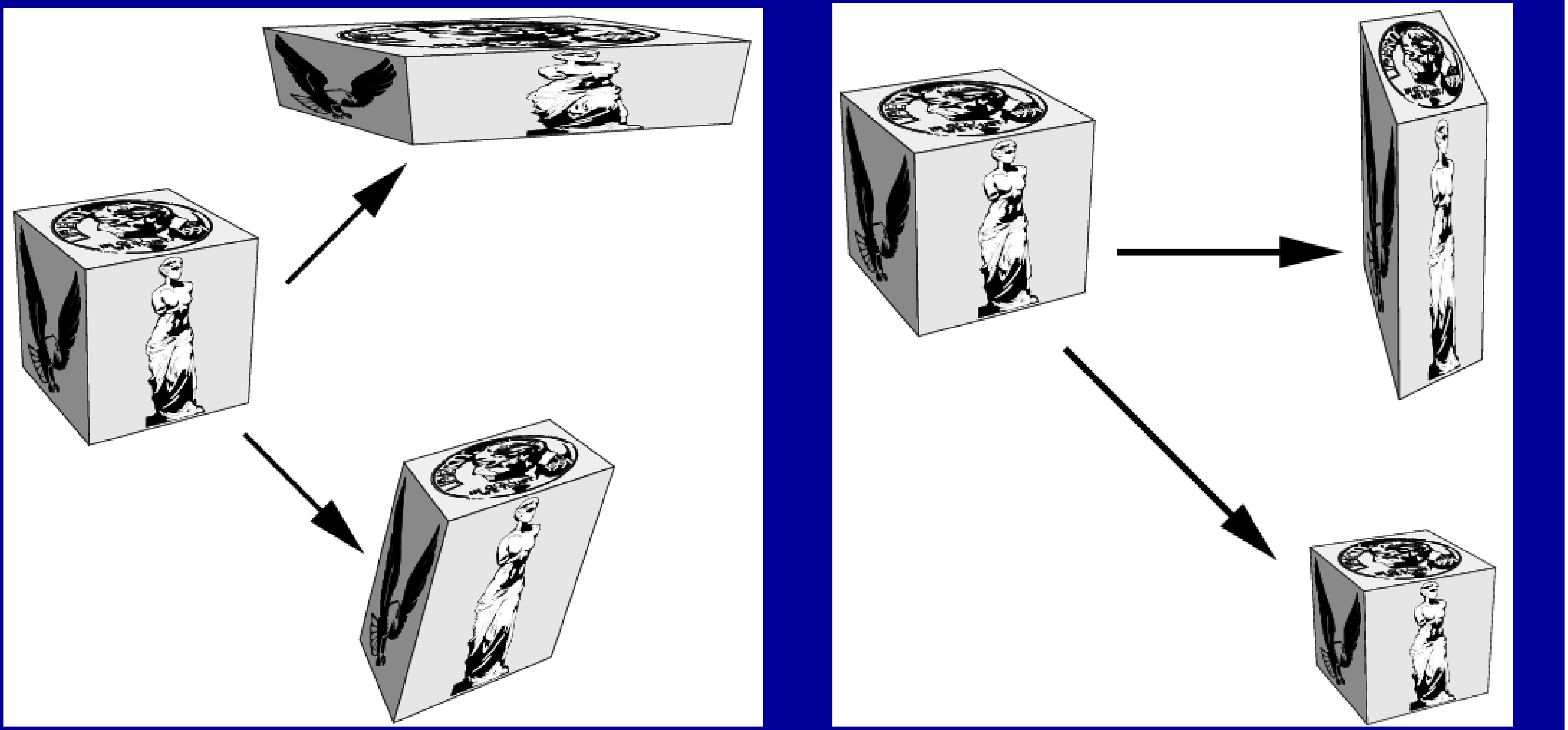






Rotation angle and line about which to rotate

Non-rigid Body Transformations



Distance between points on object do not remain constant

Basic 2D Transformations

Scale

Shear

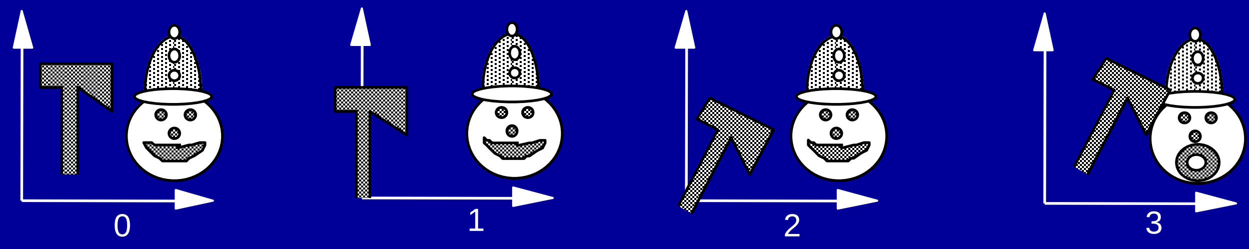
Rotate

chalkboard

Composition of Transformations

- Created by stringing basic ones together, e.g.
  - *“translate p to the origin, rotate, then translate back”*

  can also be described as a rotation about p
- Any sequence of linear transformations can be collapsed into a single matrix formed by multiplying the individual matrices together
- Order matters!
- Can apply a whole sequence of transformations at once



chalkboard

Translate to the origin, rotate, then translate back.

3D Transformations

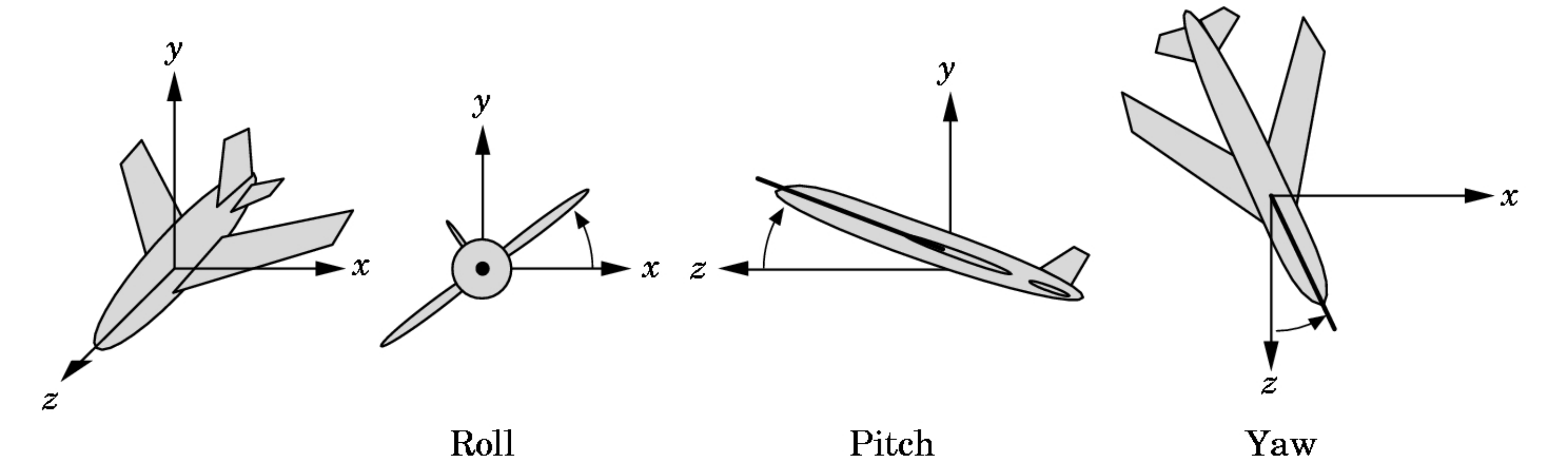
- 3-D transformations are very similar to the 2-D case
- Scale
- Shear
- Rotation is a bit more complicated in 3-D
  - different rotation axes

chalkboard

Fixed Euler Angles for 3-D Rotations

- Independent rotations about each coordinate axis
  - angle interpolation for animation generates bizarre motions
  - rotations are order-dependent, and there are no conventions about the order to use
- Widely used anyway, because they're “simple”

Euler Angles for 3-D Rotations



Roll

Pitch

Yaw

Other representations of 3D orientation

- Relative rather than fixed Euler angles
- Axis/Angle: rotate by α about axis V
- Quaternions:
  - a generalization of complex numbers
  - 3 give the rotation axis - magnitude is sin $\alpha/2$
  - 1 gives cos $\alpha/2$ (the amount of rotation)
  - unit magnitude - points on a 4-D unit sphere

But what about translation?

- Translation is not linear--how to represent as a matrix?

chalkboard

But what about translation?

- Translation is not linear--how to represent as a matrix?

chalkboard

- Trick: add extra coordinate to each vector
- This extra coordinate is the *homogeneous* coordinate, or *w*
- When extra coordinate is used, vector is said to be represented in *homogeneous coordinates*
- We call these matrices *Homogeneous* Transformations

W? Where did that come from?

- Practical answer:
  - W is a clever algebraic trick.
  - Don't worry about it too much. The w value will be 1.0 for the time being (until we get to perspective viewing transformations)

- More complete answer:
  - (x,y,w) coordinates form a 3D *projective space.*
  - All nonzero scalar multiples of (x,y,1) form an equivalence class of points that project to the same 2D Cartesian point (x,y).
  - For 3-D graphics, the 4D projective space point (x,y,z,w) maps to the 3D point (x,y,z) in the same way.

Homogeneous 2D Transformations

The basic 2D transformations become

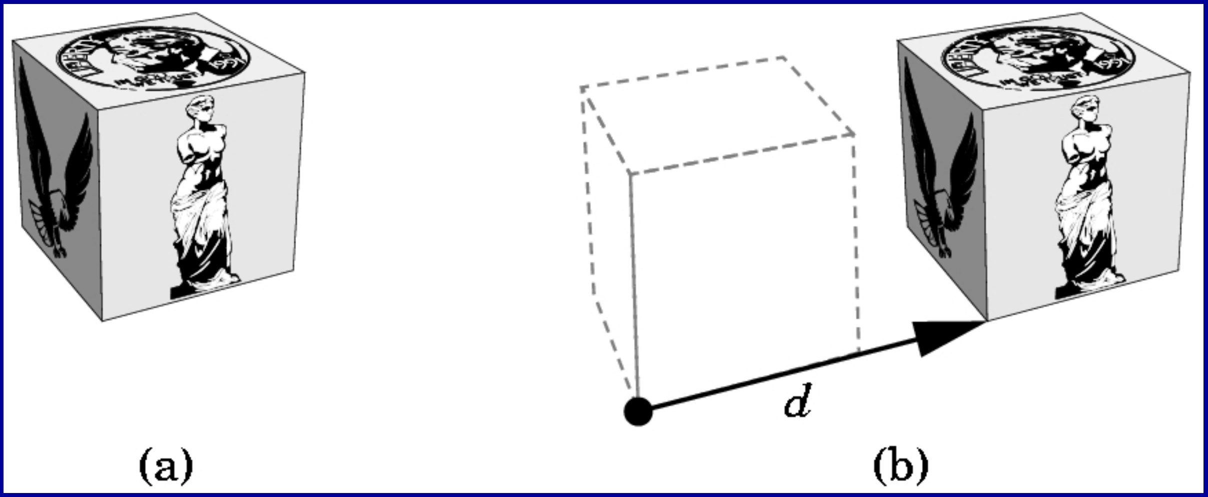
Translate: *Scale:* *Rotate:*

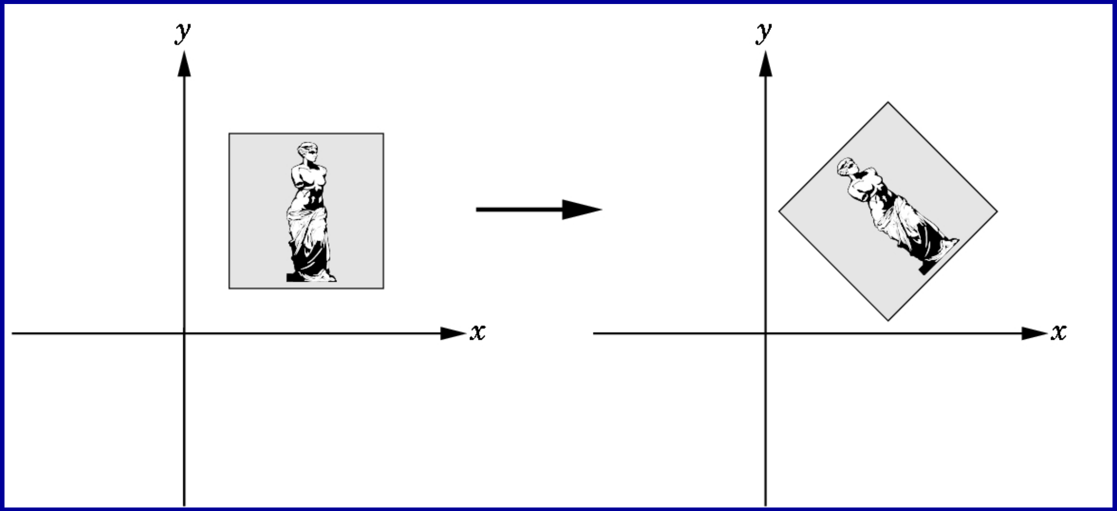
$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

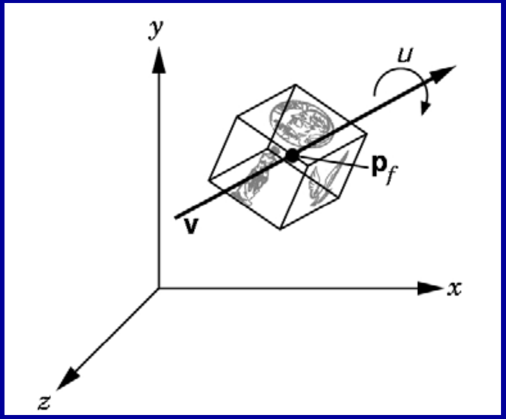
Now *any* sequence of translate/scale/rotate operations can be combined into a single homogeneous matrix by multiplication.

3D transforms are modified similarly

Rigid Body Transformations







Rotation angle and line about which to rotate

Rigid Body Transformations

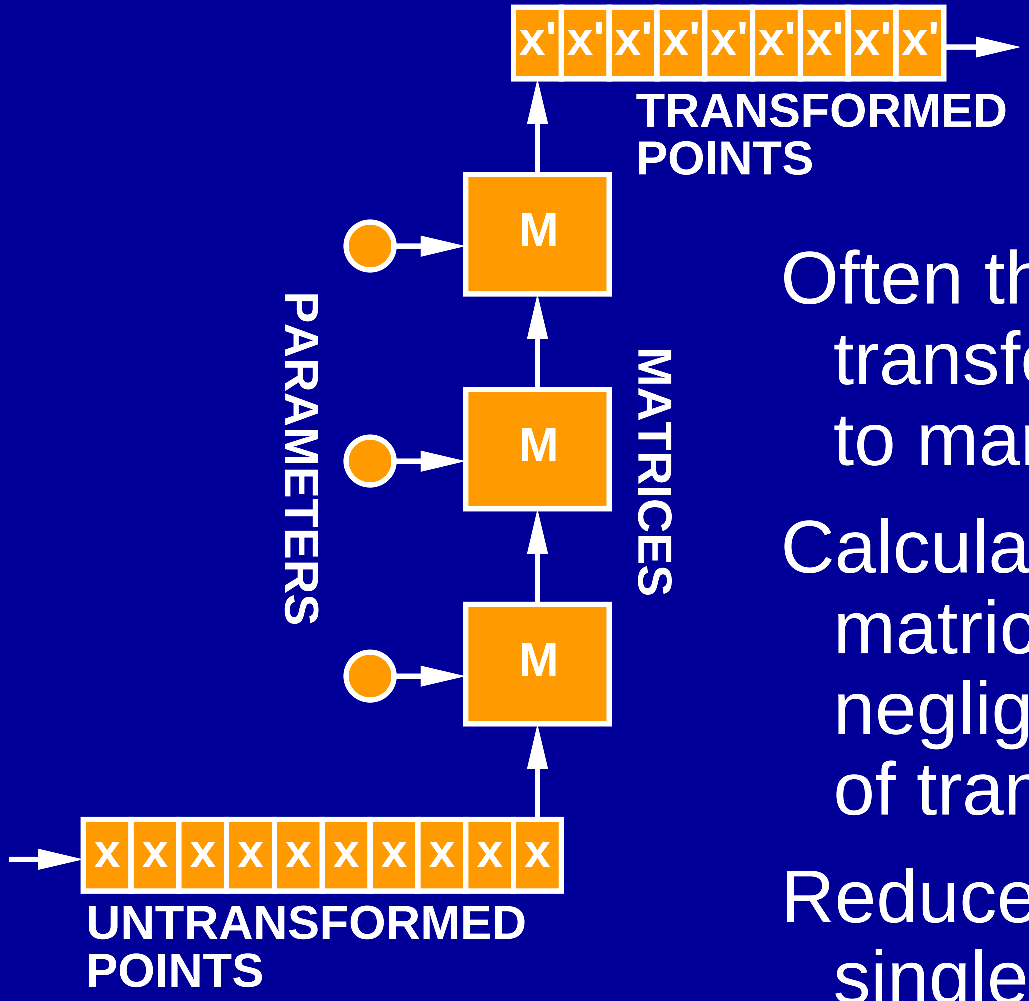
•A transformation matrix of the form

$$\begin{bmatrix} \mathbf{x_x} & \mathbf{x_y} & \mathbf{t_x} \\ \mathbf{y_x} & \mathbf{y_y} & \mathbf{t_y} \\ 0 & 0 & 1 \end{bmatrix}$$

where the upper 2x2 submatrix is a rotation matrix and column 3 is a translation vector, is a *rigid body transformation.*

•Any series of rotations and translations results in a rotation and translation of this form (and no change in the distance between vertices)

Sequences of Transformations



Often the same transformations are applied to many points

Calculation time for the matrices and combination is negligible compared to that of transforming the points

Reduce the sequence to a single matrix, then transform

Collapsing a Chain of Matrices.

- Consider the composite function ABCD, i.e. p’ = ABCDp
- Matrix multiplication isn’t commutative - the order is **important**
- But matrix multiplication is associative, so can calculate from right to left or left to right: ABCD = (((AB) C) D) = (A (B (CD))).
- Iteratively replace *either* the leading or the trailing pair by its product

$M \leftarrow D$

$M \leftarrow CM$

$M \leftarrow BM$

$M \leftarrow AM$

Premultiply

or

$M \leftarrow A$

$M \leftarrow MB$

$M \leftarrow MC$

$M \leftarrow MD$

Postmultiply

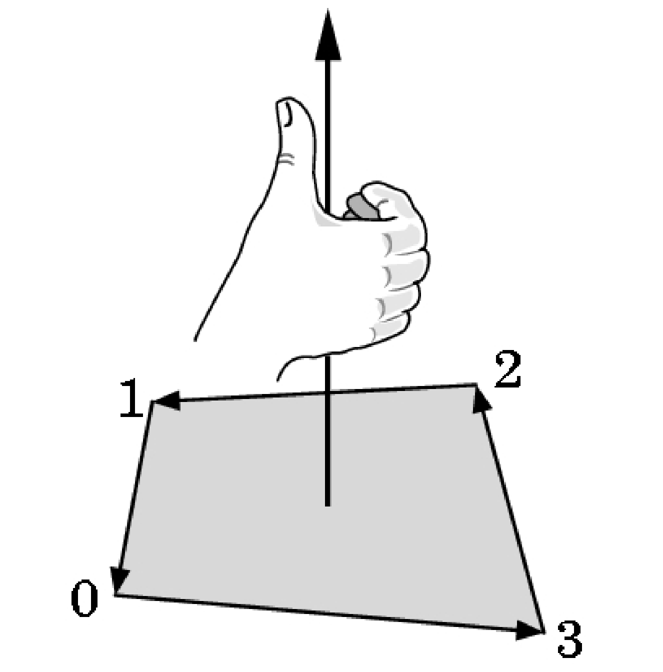
both give the same result.

What is a Normal?– refresher

Indication of outward facing direction for lighting and shading

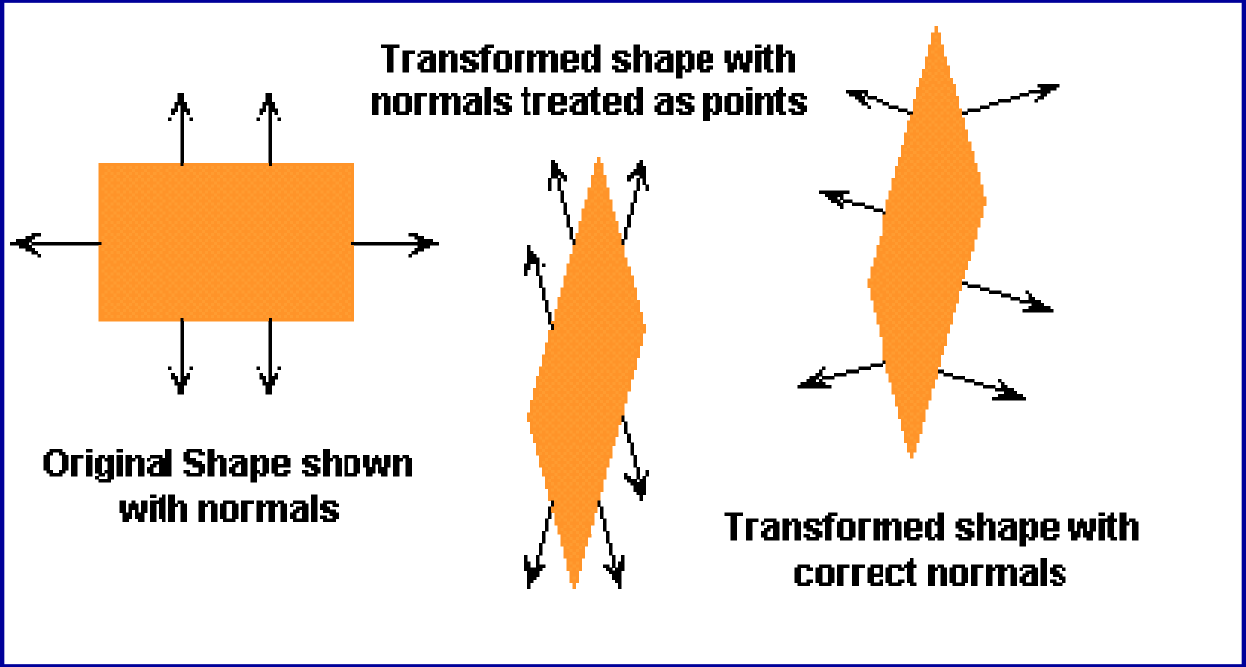
Order of definition of vertices in OpenGL

Right hand rule



Transforming Normals

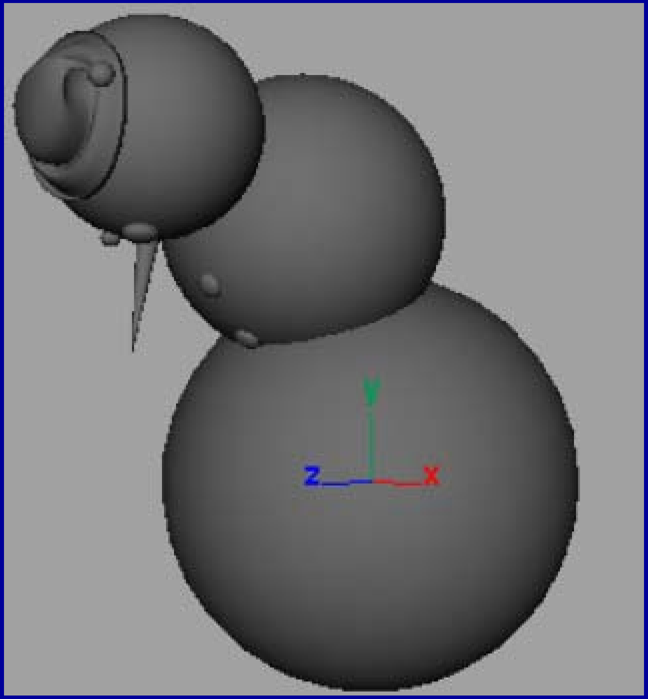
- It’s tempting to think of normal vectors as being like porcupine quills, so they would transform like points
- Alas, it’s not so.
- We need a different rule to transform normals.


Transformed shape with normals treated as points
Original Shape shown with normals
Transformed shape with correct normals

chalkboard

Examples

- Modeling with primitive shapes



Announcements

- Assignment 1 is due Thursday at midnight
- Turnin space:
  - /afs/andrew/scs/cs/15-462/turnin
  - You should be able to create files there but not read or remove
  - Use a directory name of the format: assignment1_jkh

- TA hours Tuesday 3-5 (Joel—cluster) and Thursday 3-5 (Michael—Wean 8121)
- Or send any of us email

- Questions on Assignment 1?
- Next class:
  - A bit about the details of doing this in OpenGL
  - Hierarchies of transformations