2D Transformations

- Represent **linear** transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.

[H&B chapter 5]

2D Transformations

- Represent **linear** transformations by matrices
- To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
- Recall the definition of matrix times vector:

$$\begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- A linear function f(x) satisfies (by definition):

$$f(ax+by) = af(x) + bf(y)$$

- Note that “x” can be an abstract entity (e.g. a vector)–as long as addition and multiplication by a scalar are defined.
- Algebra reveals that matrix multiplication satisfies the above condition

- In particular., if we define f(**x**)=M • **x**, where M is a matrix and **x** is a vector, then

$$\begin{aligned} f(a\mathbf{x}+b\mathbf{y}) &= M(a\mathbf{x}+b\mathbf{y}) \\ &= aM\mathbf{x}+bM\mathbf{y} \\ &= af(\mathbf{x})+bf(\mathbf{y}) \end{aligned}$$

- Where the middle step can be verified using algebra (next slide)

Proof that matrix multiplication is linear

$$M(a\mathbf{x}+b\mathbf{y}) = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\begin{pmatrix} ax_1+by_1 \\ ax_2+by_2 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}ax_1 + a_{11}by_1 + a_{12}ax_2 + a_{12}by_2 \\ a_{21}ax_1 + a_{21}by_1 + a_{22}ax_2 + a_{22}by_2 \end{pmatrix}$$

$$= a\begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{pmatrix} + b\begin{pmatrix} a_{11}y_1 + a_{12}y_2 \\ a_{21}y_1 + a_{22}y_2 \end{pmatrix}$$

$$= aM\mathbf{x} + bM\mathbf{y}$$

- Now consider the linear transformation of a point on a line segment connecting two points, **x** and **y**.
- Recall that in parametric form, that point is: $t\mathbf{x}+(1-t)\mathbf{y}$
- The transformed point is: $f(t\mathbf{x}+(1-t)\mathbf{y}) = tf(\mathbf{x})+(1-t)f(\mathbf{y})$
- Notice that is a point on the line segment from the point $f(\mathbf{x})$ to the point $f(\mathbf{y})$,
- This shows that a linear transformation maps line segments to line segments

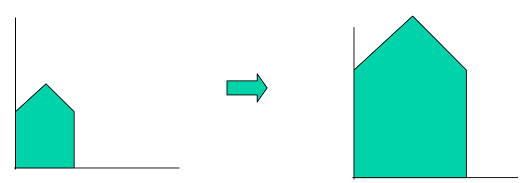
[H&B chapter 5]

2D Transformations of objects

- To transform line segments, transform endpoints
- To transform polygons, transform vertices

2D Transformations

- Scale (stretch) by a factor of k

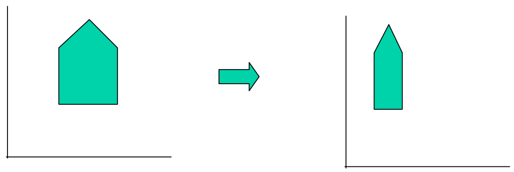


$$M = \begin{vmatrix} k & 0 \\ 0 & k \end{vmatrix}$$

(k = 2 in the example)

2D Transformations

- Scale by a factor of (S_x, S_y)

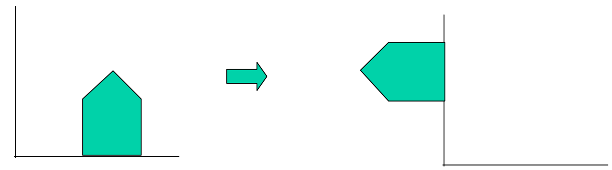


$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$

(Above, $S_x = 1/2$, $S_y = 1$)

2D Transformations

- Rotate around origin by θ (Orthogonal)

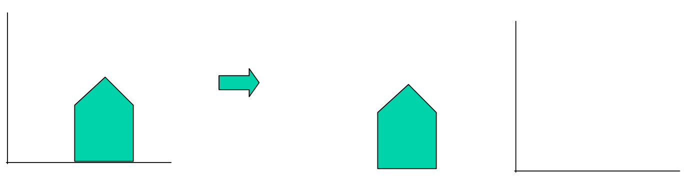


$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$

(Above, $\theta = 90^o$)

2D Transformations

- Flip over y axis (Orthogonal)

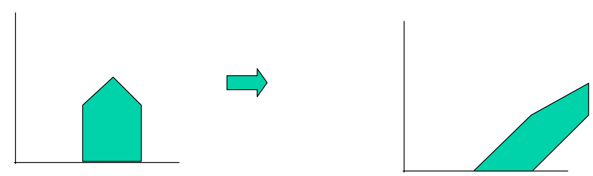


$$M = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

Flip over x axis is ?

2D Transformations

- Shear along x axis

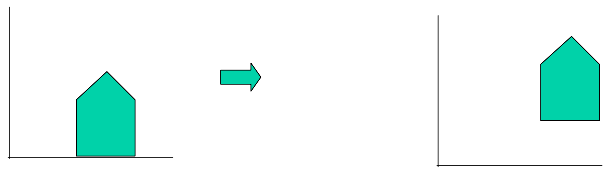


$$M = \begin{vmatrix} 1 & a \\ 0 & 1 \end{vmatrix}$$

Shear along y axis is ?

2D Transformations

- Translation $(\mathbf{P}_{new} = \mathbf{P} + \mathbf{T})$



M =?

Homogenous Coordinates

- Represent 2D points by 3D vectors
- (x,y)-->(x,y,1)
- Now a multitude of 3D points (x,y,W) represent the same 2D point, (x/W, y/W, 1)
- Represent 2D transforms with 3 by 3 matrices
- Can now do translations
- Homogenous coordinates have other uses/advantages (later)

2D Translation in H.C.

$\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$

$(x', y') = (x, y) + (t_x, t_y)$

$$M = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$

2D Scale in H.C.

$$M = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Rotation in H.C.

$$M = \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Composition of Transformations

- If we use one matrix, M_1 for one transform and another matrix, M_2 for a second transform, then the matrix for the first transform followed by the second transform is simply $M_2 M_1$
- This generalizes to any number of transforms
- Computing the combined matrix **first** and then applying it to many objects, can save **lots** of computation

Composition Example

- Matrix for rotation about a point, P
- Problem--we only know how to rotate about the origin.

Composition Example

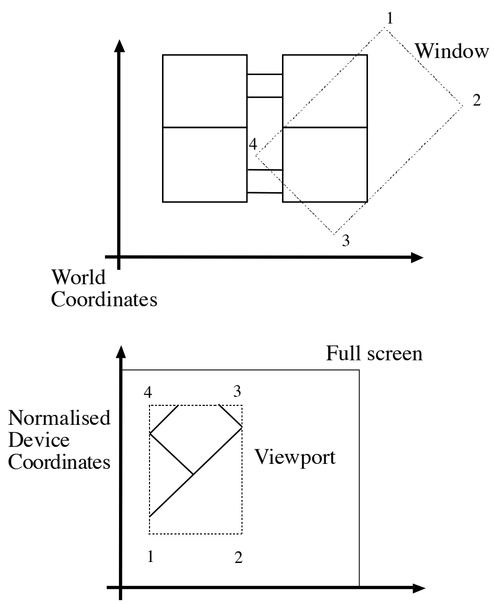
- Matrix for rotation about a point, P
- Problem--we only know how to rotate about the origin.
- Solution--translate to origin, rotate, and translate back

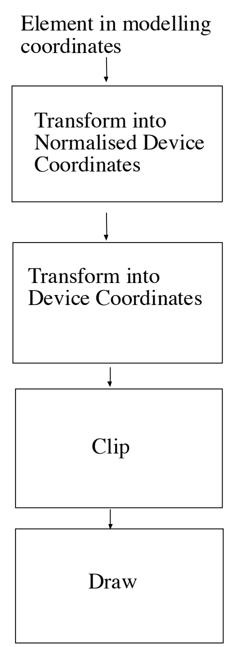
2D transformations (continued)

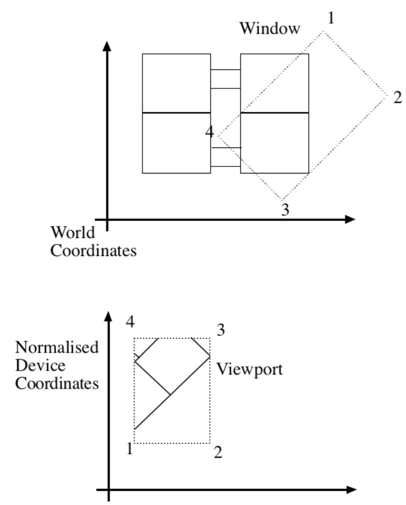
- The transformations discussed so far are invertable (why?). What are the inverses?

2D viewing

- Three coordinate systems are common in graphics
  - World coordinates or modeling coordinates - where the model is defined (meters, miles, etc.)
  - Normalized device coordinates; usually (0-1) in each variable.
  - Device coordinates: the actual coordinates of the pixels on the frame-buffer or the printer
- Need to construct transformations between coordinate systems
- Terminology:
  - window = region on drawing that will be displayed (rectangle)
  - viewport = region in NDC's/DC's where this rectangle is displayed (often simply entire screen).

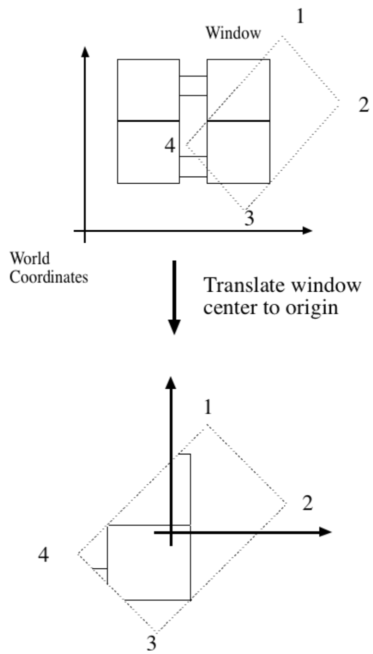






Determining the transform

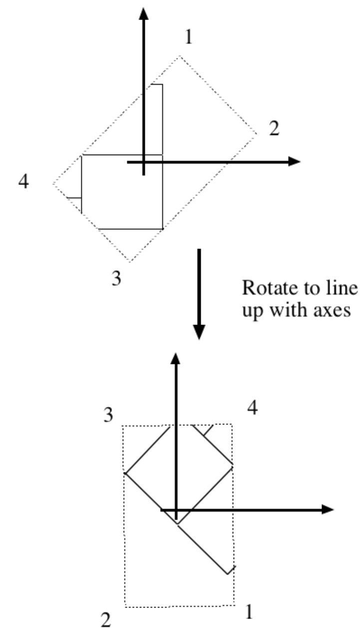
- **Plan A**: Consider this as a sequence of transformations in homogenous coords, then determine each element in closed form.
- **Plan B**: Compute numerically from point correspondences (not covered in detail in 2006)



- write (wx_i, wy_i) for coordinates of i'th point on window
- translation is:

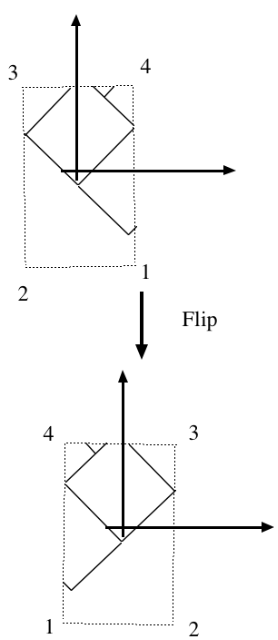
$$\begin{pmatrix}x'\\y'\\1\end{pmatrix}=\begin{pmatrix}1&0&-\overline{wx}\\0&1&-\overline{wy}\\0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}$$

(overbar denotes average over vertices, i.e., 1,2,3,4)



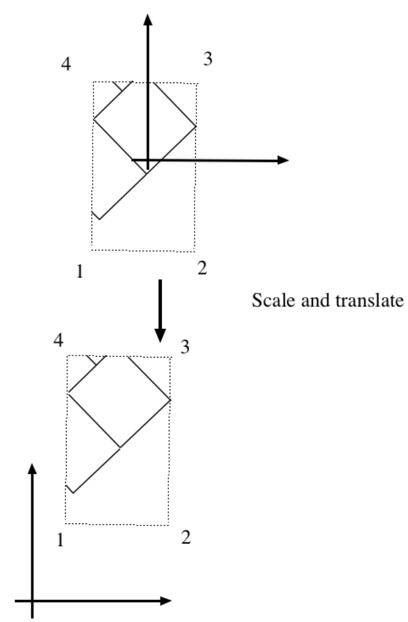
$$\begin{pmatrix}x'\\y'\\1\end{pmatrix}=\begin{pmatrix}\cos(\theta)&-\sin(\theta)&0\\\sin(\theta)&\cos(\theta)&0\\0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}$$

(Need to compute theta)



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

(Vertex order does not correspond, need to flip)



$$\begin{pmatrix}x'\\y'\\1\end{pmatrix}=\begin{pmatrix}\frac{w_{new}}{w_{old}}&0&\overline{x_{new}}\\0&\frac{h_{new}}{h_{old}}&\overline{y_{new}}\\0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}$$

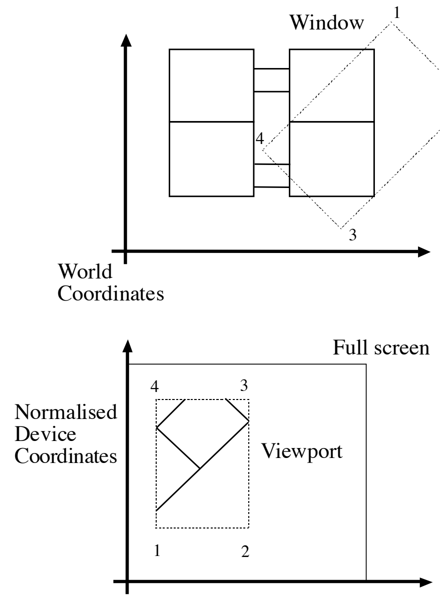
Notice that choice of new width, height, and center give translation to either normalized device coords, or to device coordinates

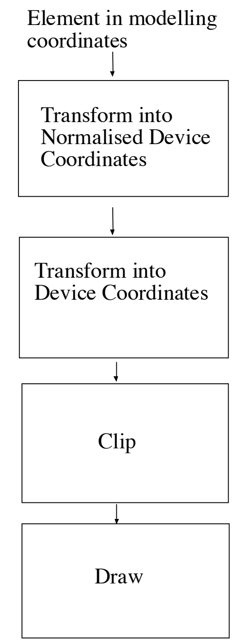
- Get overall transformation by multiplying transforms.
- This gives a single transformation matrix, whose elements are functions of window/viewport coordinates.

$$x' = M_{(\text{translate origin to viewport cog, scale})}\, M_{(\text{flip})}\, M_{(\text{rotate})} M_{(\text{translate window cog->origin})} x$$

NDC's/DC's — World coords

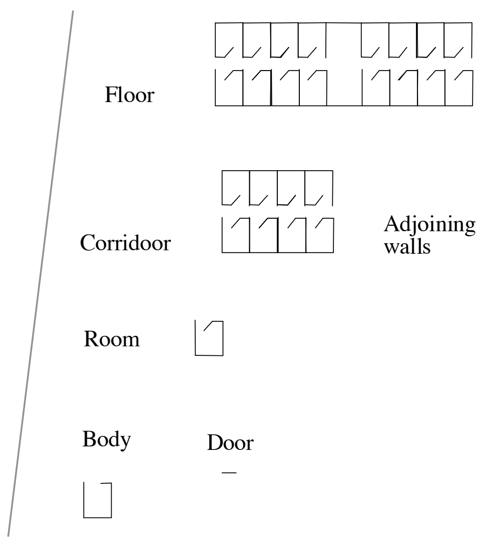
(cog==window center of gravity)





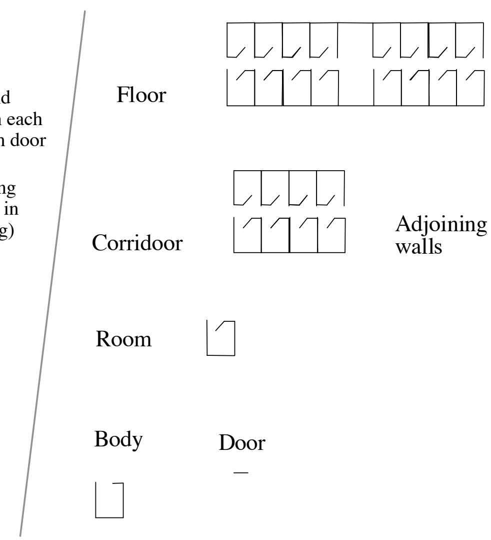
Hierarchical modeling

- Consider constructing a complex 2d drawing: e.g. an animation showing the plan view of a building, where the doors swing open and shut.

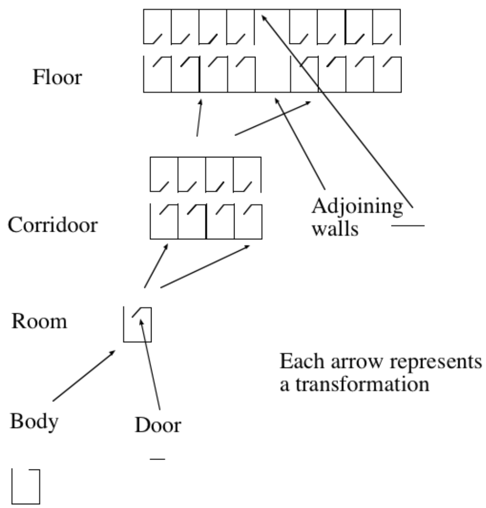


Hierarchical modeling

- Options:
  - specify everything in world coordinate frame; but then each room is different, and each door moves differently.
  - Exploit similarities by using repeated copies of models in different places (instancing)



Hierarchical modeling



Hierarchical modeling

- Model form
  - Directed acyclic graph.
  - Each node consists of 0 or more objects (lines, polygons, etc).
  - Each edge is a transformation
- There can be many edges joining two nodes (e.g. in the case of the corridor - many copies of the same room model, each transformed differently).
- Every graphics API supports hierarchies - some directly (meaning you have to learn a language to express the model) some indirectly with a matrix stack

Hierarchical modeling

Write the transformation from door coordinates to room coordinates as:

$$T_{room}^{door}$$

Then to render a door, use the transformation:

$$T_{device}^{world} T_{floor}^{corridoor} T_{corridoor}^{room} T_{room}^{door}$$

To render a body, use the transformation:

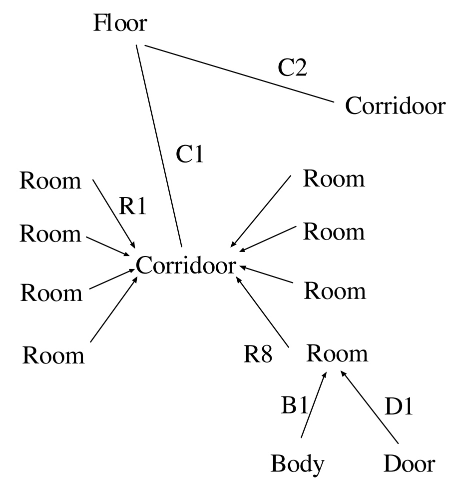
$$T_{device}^{world} T_{floor}^{corridoor} T_{corridoor}^{room} T_{room}^{body}$$

Matrix stacks and rendering

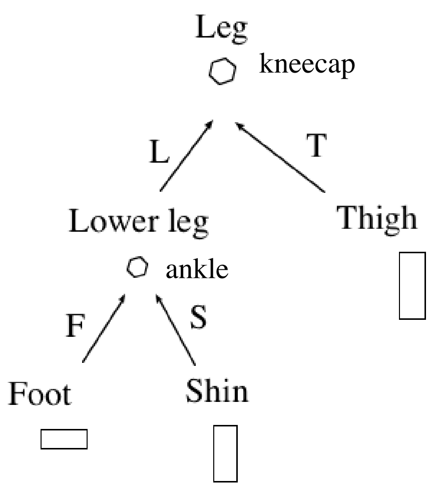
- Matrix stack:
  - Stack of matrices used for rendering
  - Applied in sequence.
  - Pop=remove last matrix
  - Push=append a new matrix
  - In previous example, body-device transformation comes from door-device transformation by popping door-room and pushing body-room

Matrix stacks and rendering

- Algorithm for rendering a hierarchical model:
  - stack is T_{device}^{root}
  - render (root)
- Recursive definition of render (node)
  - if node has object, render it
  - for each child:
    - push transformation
    - render (child)
    - pop transformation



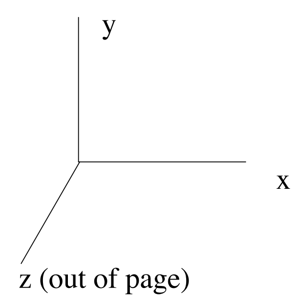
- Now to render door on first room in first corridor, stack looks like: W C1 R1 D1
- For efficiency we would store "running" products, IE, the stack contains: W, W*C1, W*C1*R1, W*C1*R1*D1.
- We do not need two copies of corridor, or 16 copies of body; we render one copy using 16 different transformations. This is known as instancing
- Animation requires care: if D1 is a single function of time, all doors will swing open and closed at the same time.

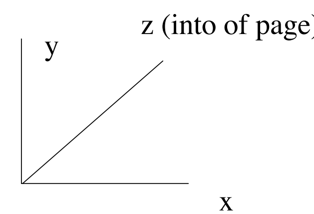


- Stack is W
- render kneecap
- Stack is W L
- render ankle
- Stack is W L F
- render foot
- Stack is W L S
- render shin
- Stack is W T
- render thigh

Transformations in 3D

- Right hand coordinate system (conventional, i.e., in math)
- In graphics a LHS is sometimes also convenient (Easy to switch between them--later).





Transformations in 3D

- Homogeneous coordinates now have four components - traditionally, (x, y, z, w)
  - ordinary to homogeneous: (x, y, z) -> (x, y, z, 1)
  - homogeneous to ordinary: (x, y, z, w) -> (x/w, y/w, z/w)
- Again, translation can be expressed as a multiplication.

Transformations in 3D

- Translation:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D transformations

- Anisotropic scaling:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Shear (one example):

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

Rotations in 3D

- 3 degrees of freedom
- Orthogonal, det(R)=1
- We can easily determine formulas for rotations about each of the axes
- For general rotations, there are many possible representations—we will use a **sequence** of rotations about coordinate axes.
- Sign of rotation follows the Right Hand Rule--point thumb along axis in direction of increasing ordinate--then fingers curl in the direction of positive rotation).

Rotations in 3D

- About x-axis

$$M = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotations in 3D

- About y-axis

$$M = \begin{vmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotations in 3D

- About z-axis

$$M = \begin{vmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Commuting transformations

- If A and B are matrices, does AB=BA? Always? Ever?
- What if A and B are restricted to particular transformations?
- What about the 2D transformations that we have studied?
- How about if A and B are restricted to be on of the three specific 3D rotations just introduced, such as rotation about the Z axis?

Demo

Commuting transformations

- If A and B are matrices, does AB=BA? Always? Ever?
- What if A and B are restricted to particular transformations?
- What about the 2D transformations that we have studied?
- How about if A and B are restricted to be on of the three specific 3D rotations just introduced, such as rotation about the Z axis?

Answer: In general AB != BA (matrix multiplication is not commutative). But if A and B are either translations or scalings, then multiplication is commutative. The same applies to rotations restricted to be about one of the 3 axis in 3D.

Rotations in 3D

- About X axis $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

- 90 degrees about X axis?

Rotations in 3D

- About X axis $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

- 90 degrees about X axis $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

Rotations in 3D

- About Y axis

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

- 90 degrees about Y-axis?

Rotations in 3D

- About Y axis

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

- 90 degrees about Y axis

$$\begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotations in 3D

- 90 degrees about X then Y

$$\underbrace{\begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}}_{\text{Y rot}} \underbrace{\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}}_{\text{X rot}} = ?$$

Rotations in 3D

- 90 degrees about X then Y

$$\underbrace{\begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}}_{\text{Y rot}} \underbrace{\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}}_{\text{X rot}} = \begin{vmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotations in 3D

- 90 degrees about X then Y

$$\begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Y rot X rot

- 90 degrees about Y then X

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = ?$$

X rot Y rot

Rotations in 3D

- 90 degrees about X then Y

$$\begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Y rot X rot

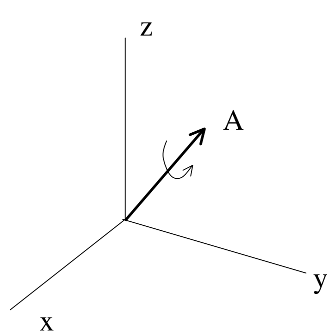
$\neq$

- 90 degrees about Y then X

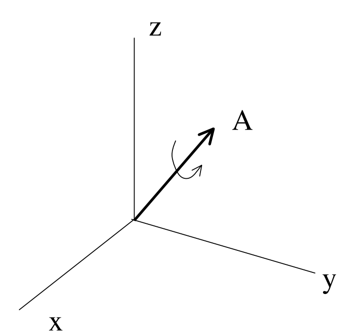
$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

X rot Y rot

Rotation about an arbitrary axis

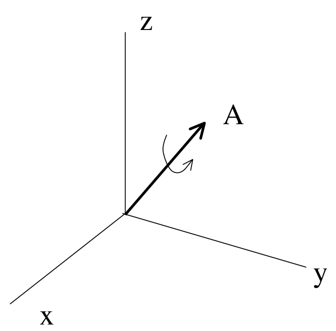


Rotation about an arbitrary axis



Strategy--rotate A to Z axis, rotate about Z axis, rotate Z back to A.

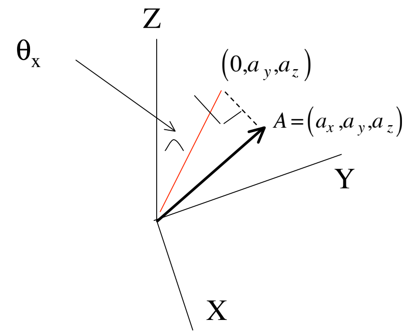
Rotation about an arbitrary axis



Tricky part:
rotate A to Z axis

Two steps.
1) Rotate about x to xz plane
2) Rotate about y to Z axis.

Rotation about an arbitrary axis

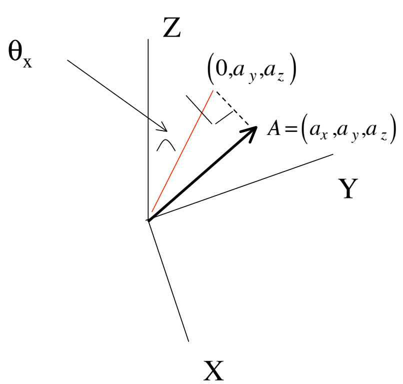


Tricky part:
rotate A to Z axis

Two steps.
1) Rotate about X to xz plane
2) Rotate about Y to Z axis.

As A rotates into the xz plane, its projection (shadow) onto the YZ plane (red line) rotates through the same angle which is easily calculated.

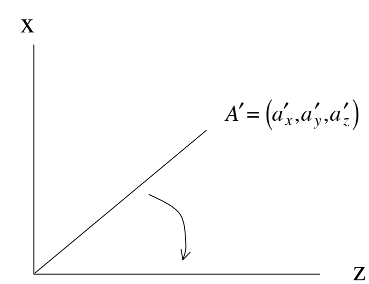
Rotation about an arbitrary axis



$$d = \sqrt{a_y^2 + a_z^2}$$
$$\sin\theta_x = a_y / d$$
$$\cos\theta_x = a_z / d$$

No need to compute angles, just put sines and cosines into rotation matrices

Rotation about an arbitrary axis



Apply $R_x(\theta_x)$ to A and renormalize to get A’

$R_y(\theta_y)$ should be easy, but note that it is clockwise.

Rotation about an arbitrary axis

Final form is

$$R_x(-\theta_x)R_y(-\theta_y)R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)$$