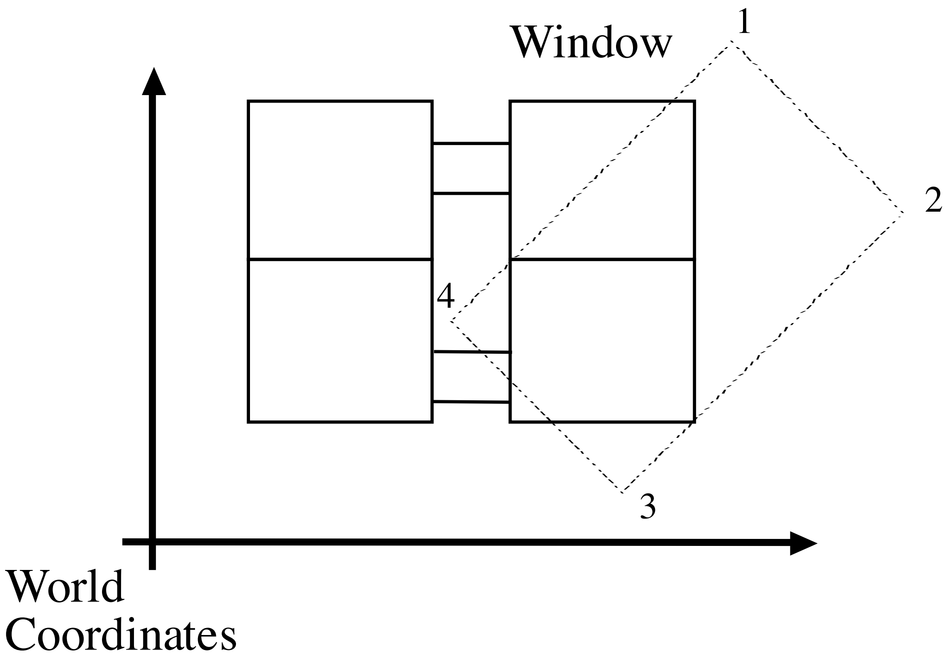
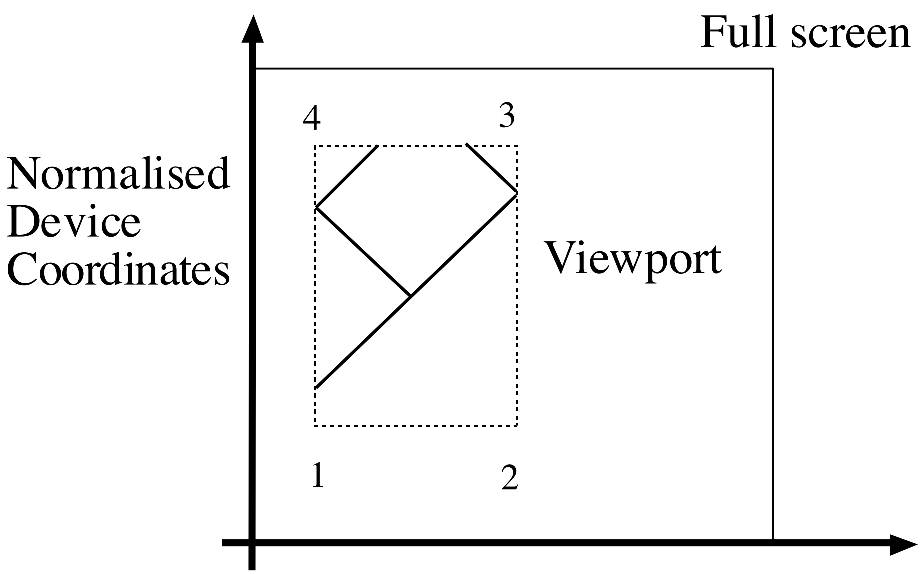
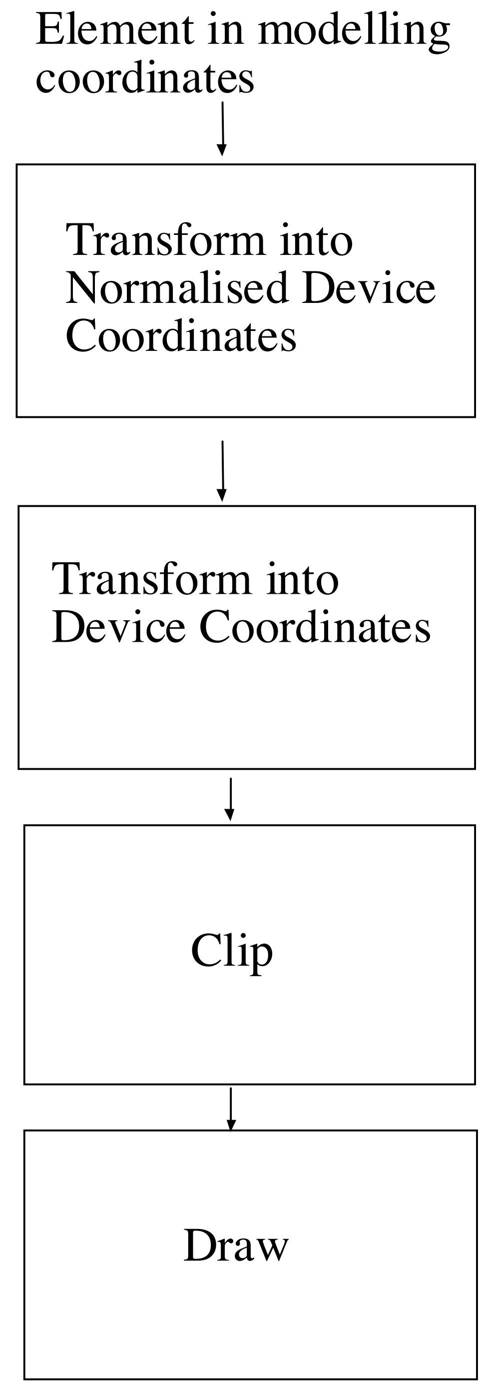
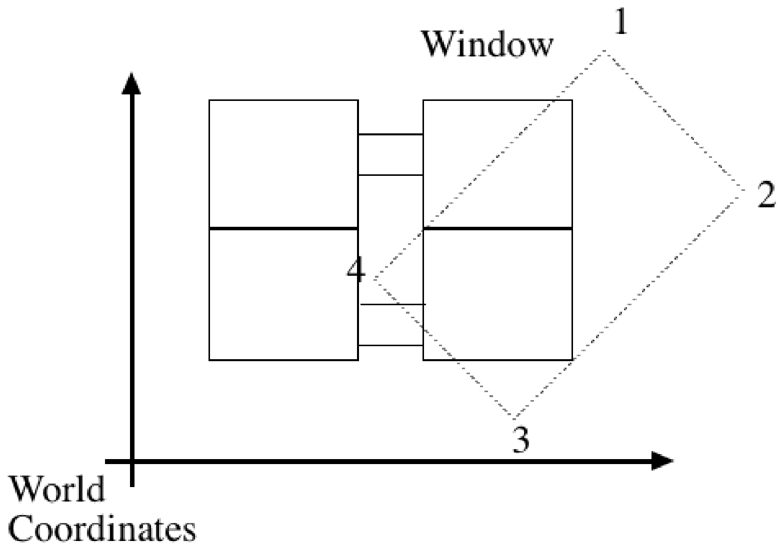
2D viewing

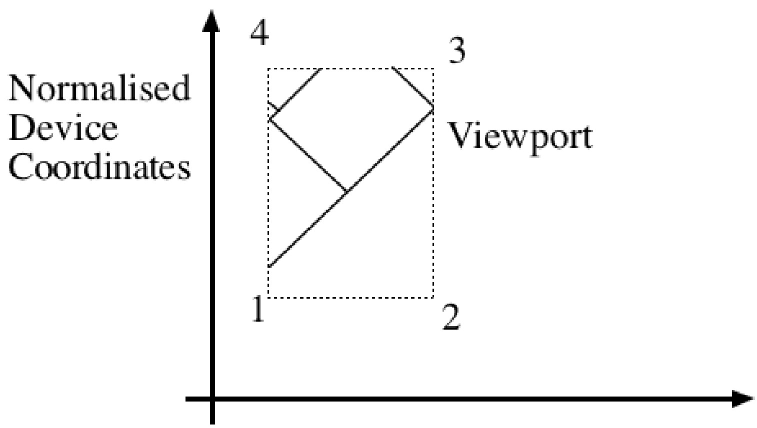
- Three coordinate systems are common in graphics
  - World coordinates or modeling coordinates - where the model is defined (meters, miles, etc.)
  - Normalized device coordinates; usually (0-1) in each variable.
  - Device coordinates: the actual coordinates of the pixels on the frame-buffer or the 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).


Window
1
2
3
4
World
Coordinates

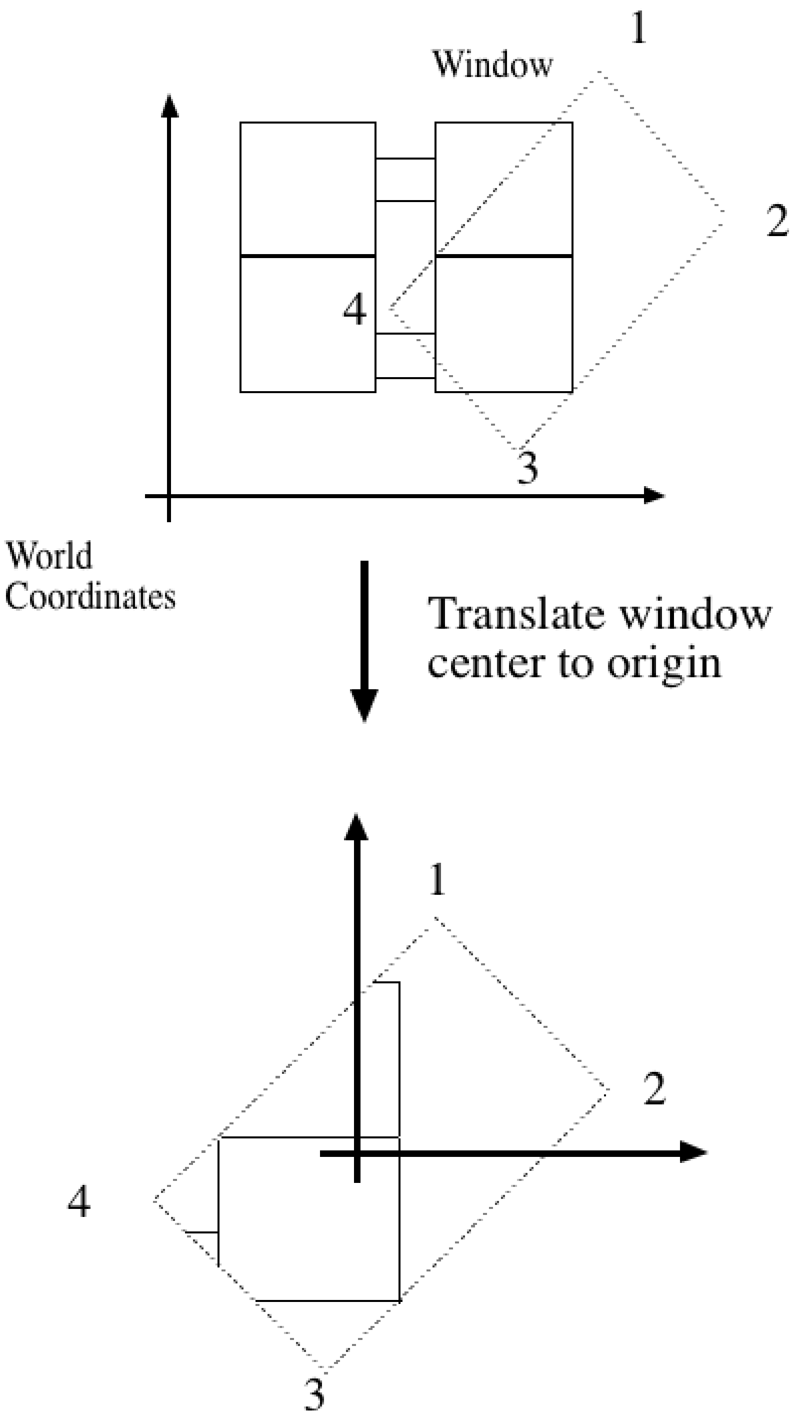

Full screen
Normalised
Device
Coordinates
4
3
Viewport
1
2


Element in modelling
coordinates
Transform into
Normalised Device
Coordinates
Transform into
Device Coordinates
Clip
Draw





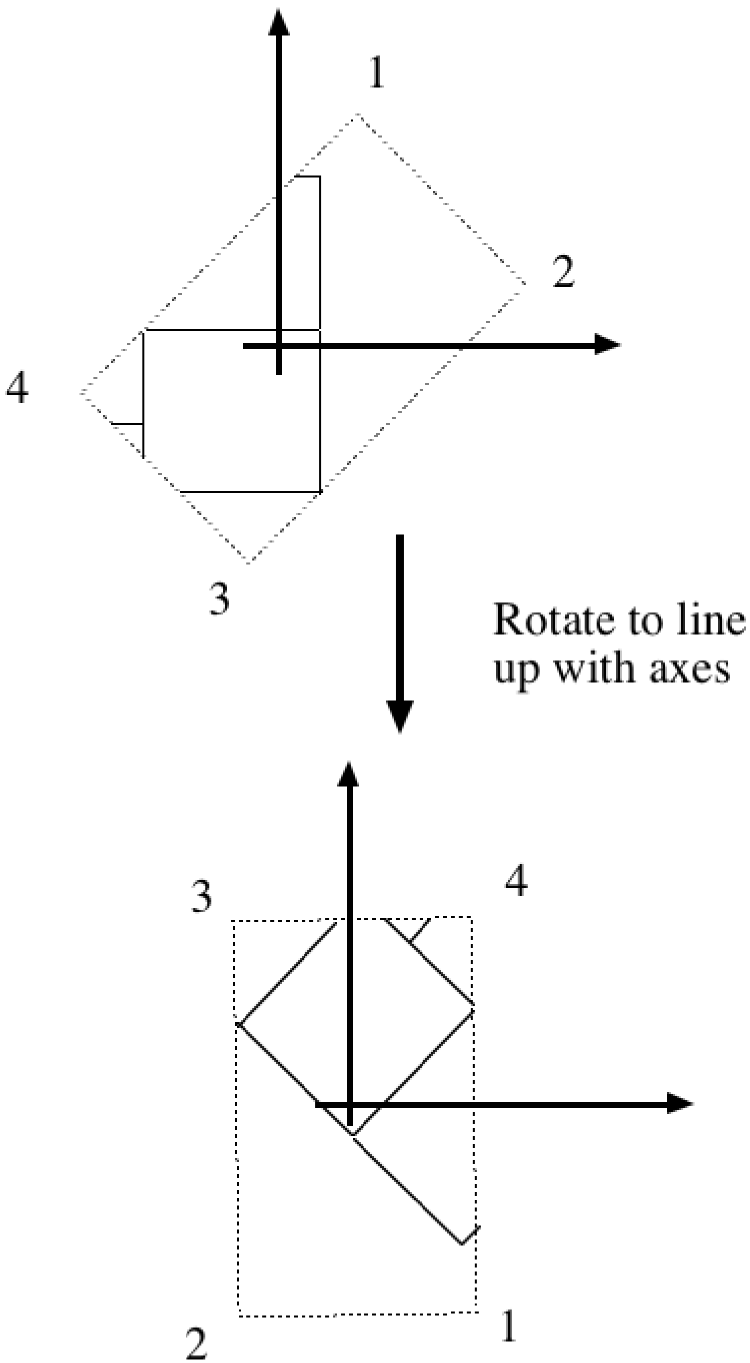
- View this as a sequence of transformations in homogenous coords, then determine each element in closed form.
- Compute numerically from point correspondences.



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

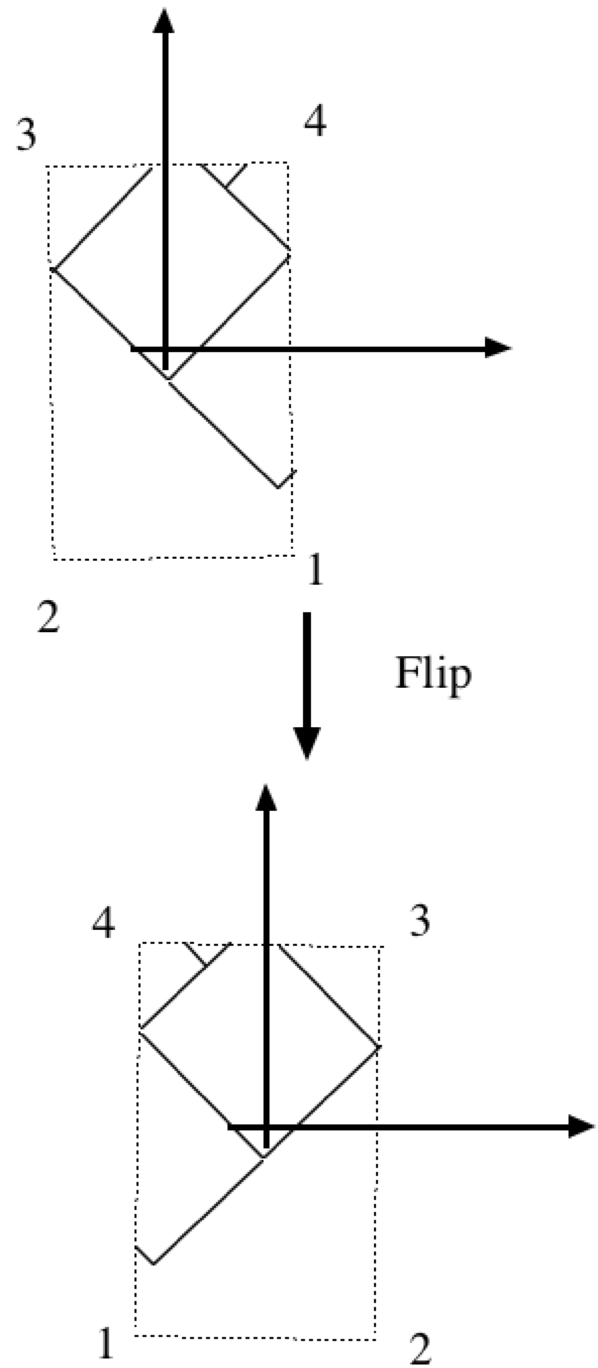
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\overline{wx} \\ 0 & 1 & -\overline{wy} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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



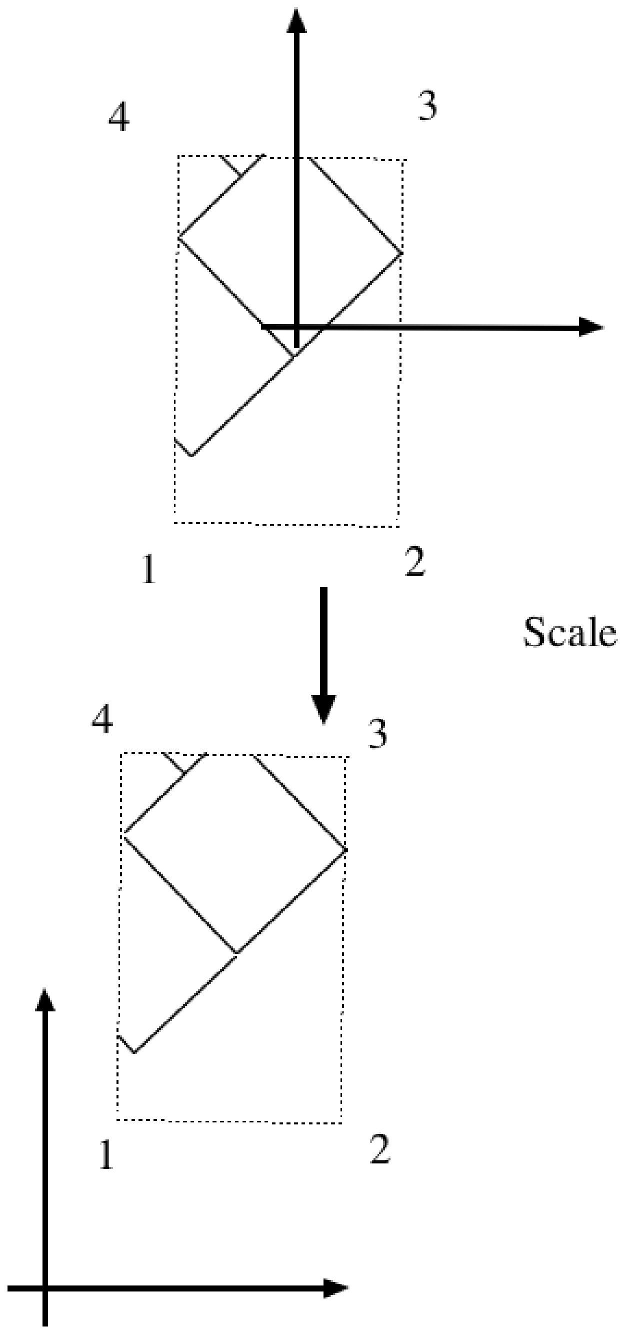
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

(Need to compute theta)



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

(Vertex order does not correspond, need to flip)



Scale and translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w_{new}}{w_{old}} & 0 & \overline{x}_{new} \\ 0 & \frac{h_{new}}{h_{old}} & \overline{y}_{new} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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)

Affine transformations

- Another approach to determining the whole transform for the pipeline; this is an affine transform.
- Matrix form:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

- Now assume that we know that $Mp_1=q_1$, $Mp_2=q_2$, $Mp_3=q_3$
- Quick way to determine transform, because this is the same as six linear equations, in six variables, which are the entries in the matrix:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix}$$

Details

- $Mp_1 = q_1$ gives first two rows
- $p_1 = (x_1, y_1, 1)^T$, $q_1 = (u_1, v_1, 1)^T$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$$

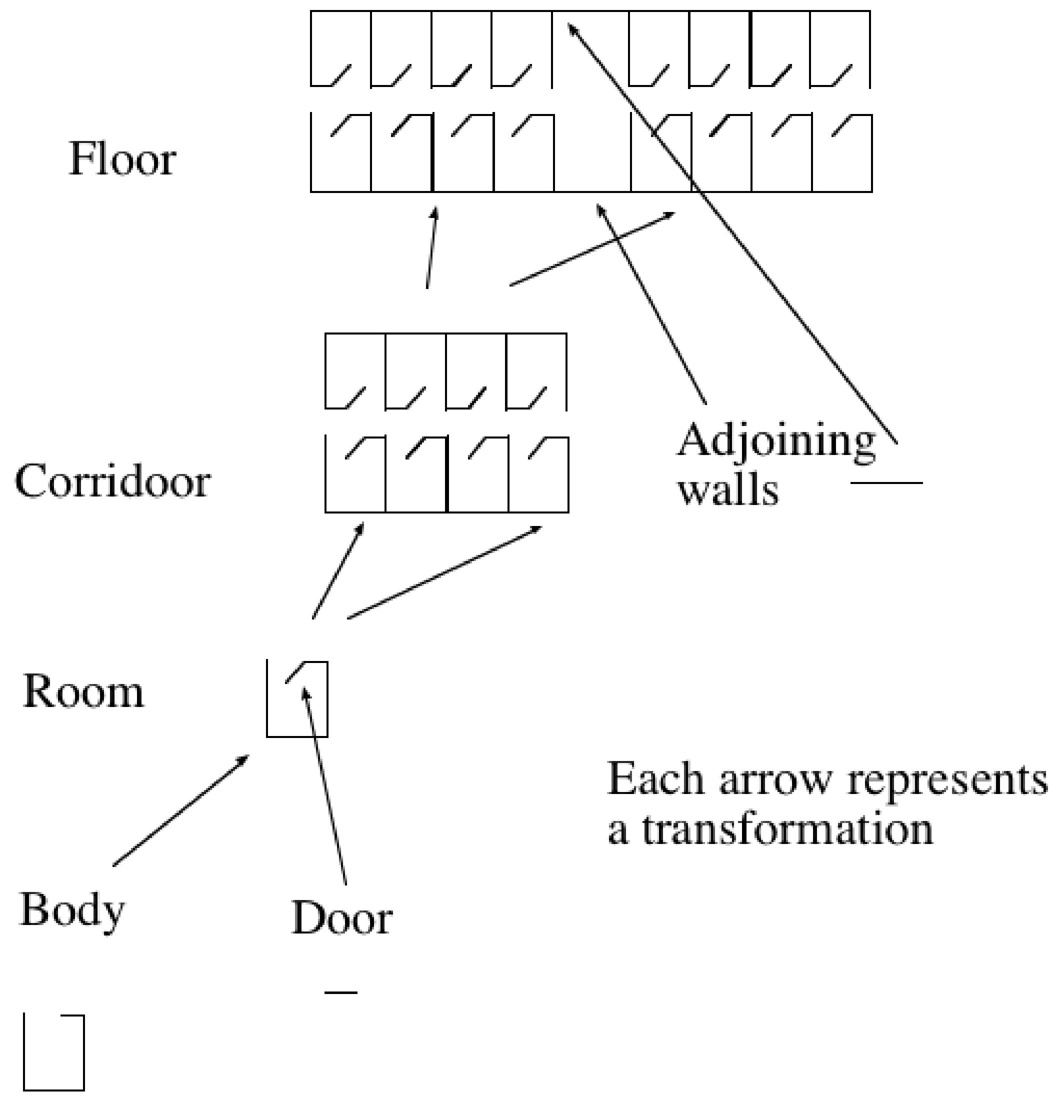
$$\boxed{\begin{aligned} ax_1 + by_1 + c &= u_1 \\ dx_1 + ey_1 + f &= v_1 \end{aligned}}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix}$$

$Mp_2 = q_2$, $Mp_3 = q_3$ give other rows

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



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

- Write the transformation from door coordinates to room coordinates as:

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

Then to render a door, use the transformation:

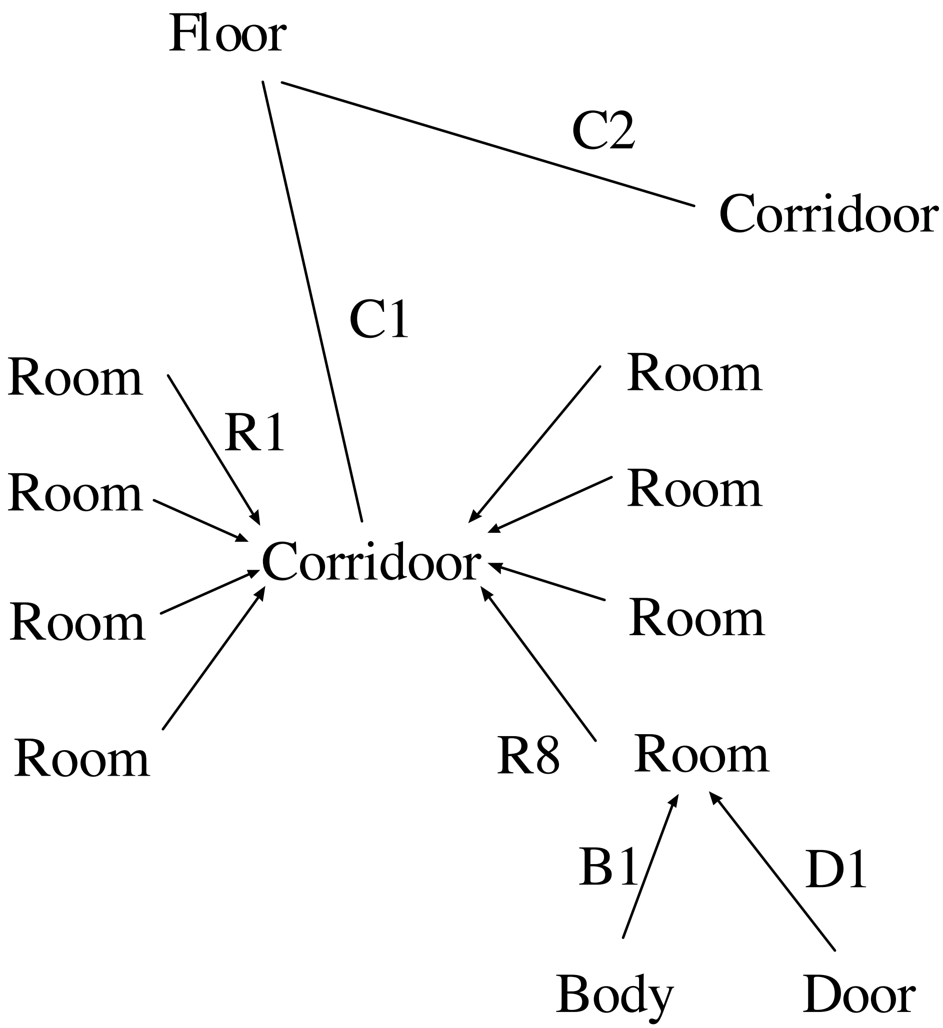
$$T_{device}^{world} T_{floor}^{corridoor} T_{corridoor}^{room} T_{room}^{door}$$

To render a body, use the transformation:

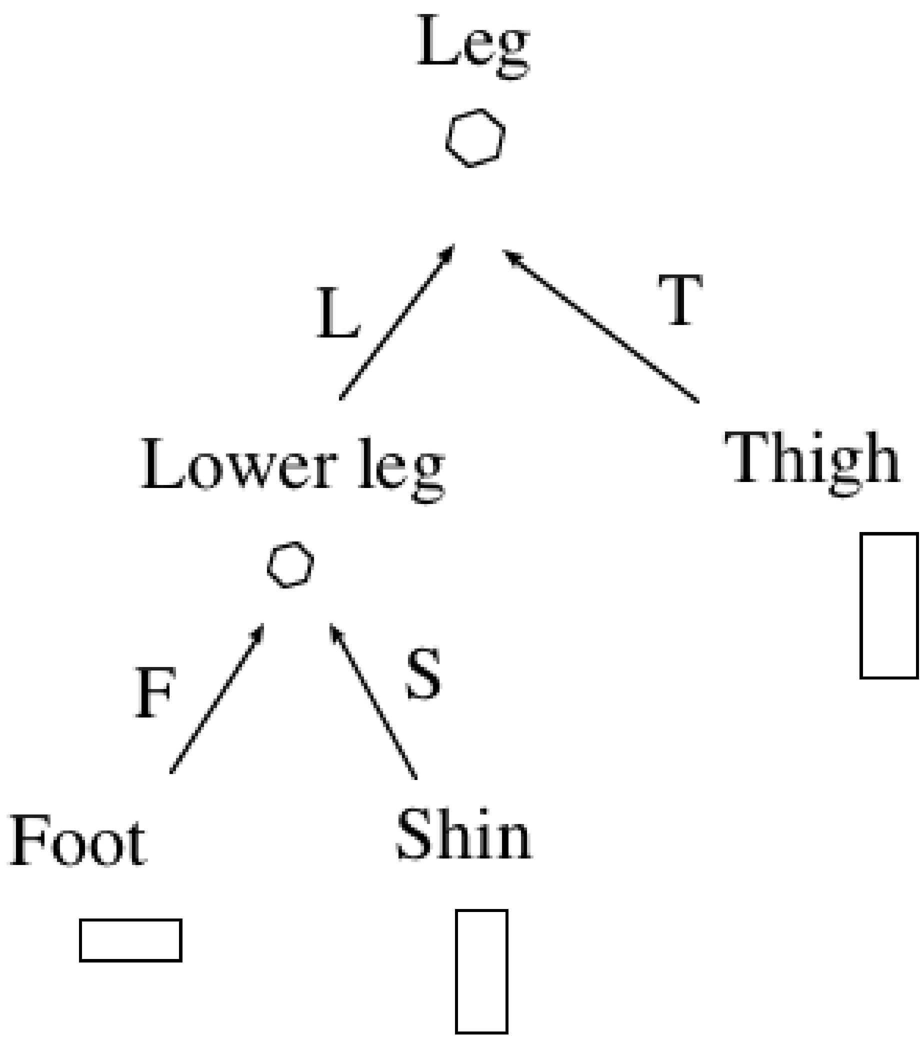
$$T_{device}^{world} T_{floor}^{corridoor} T_{corridoor}^{room} T_{room}^{body}$$

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
- Algorithm for rendering a hierarchical model:
  - stack is ________ T_{device}^{root}
  - render (root)
- Render (node)
  - for each child:
    - push transformation
    - render (child)
    - pop transformation



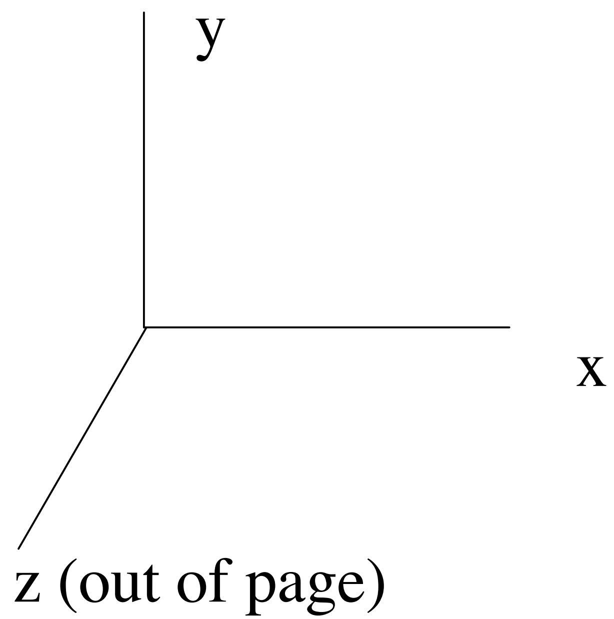
- Now to render door on first room in first corridor, stack looks like: W C1 R1 D1
- For efficiency we would store "running" products, IE, the stack contains: W, W*C1, W*C1*R1, W*C1*R1*D1.
- We do not need two copies of corridor, or 16 copies of body; we render one copy using 16 different transformations. 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.



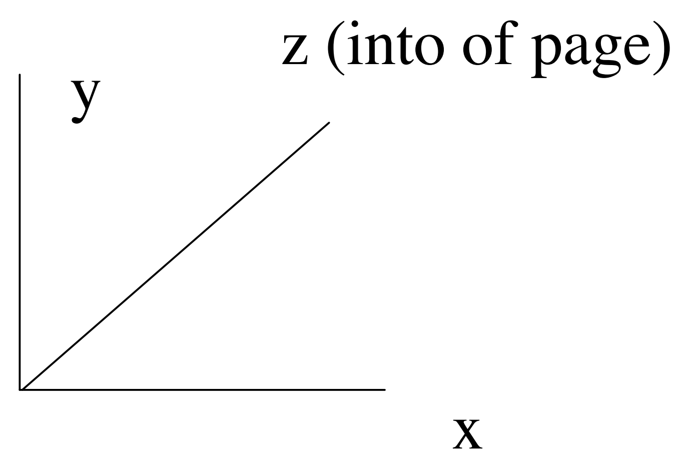
- Stack is W
- render kneecap
- Stack is W L
- render ankle
- Stack is W L F
- render foot
- Stack is W L S
- render shin
- Stack is W T
- render thigh

Transformations in 3D (Watt chapter 1)

- 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:

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

3D transformations

- Anisotropic scaling:

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

- Shear (one example):

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

Rotations in 3D

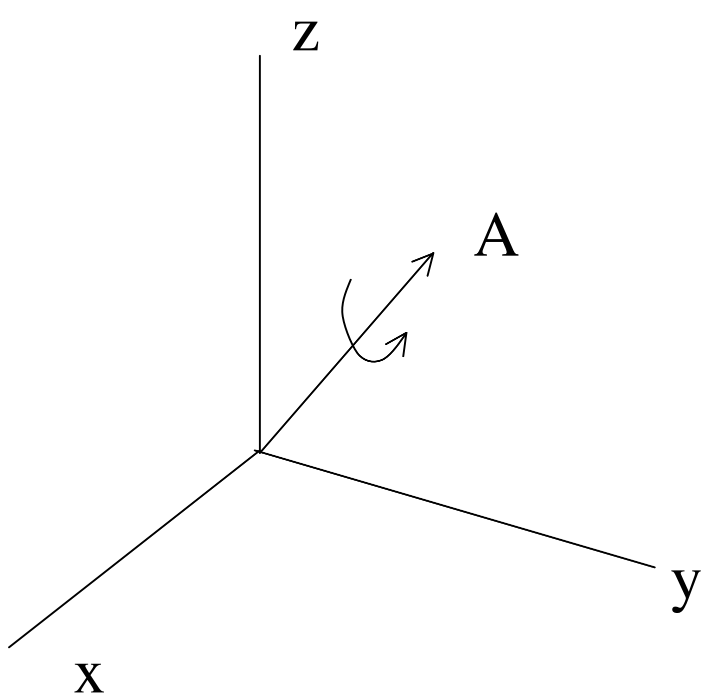
- 3 degrees of freedom
- Det(R)=1
- Orthogonal
- Many representations are possible.
- Our representation: rotate about coordinate axes in sequence.
- Sequence of axes is arbitrary, but choice does affect the angles used (cannot use same angles with different order).
- Sign of rotation follows the Right Hand Rule--point thumb along axis in direction of increasing ordinate--then fingers curl in the direction of positive rotation).

Rotations in 3D

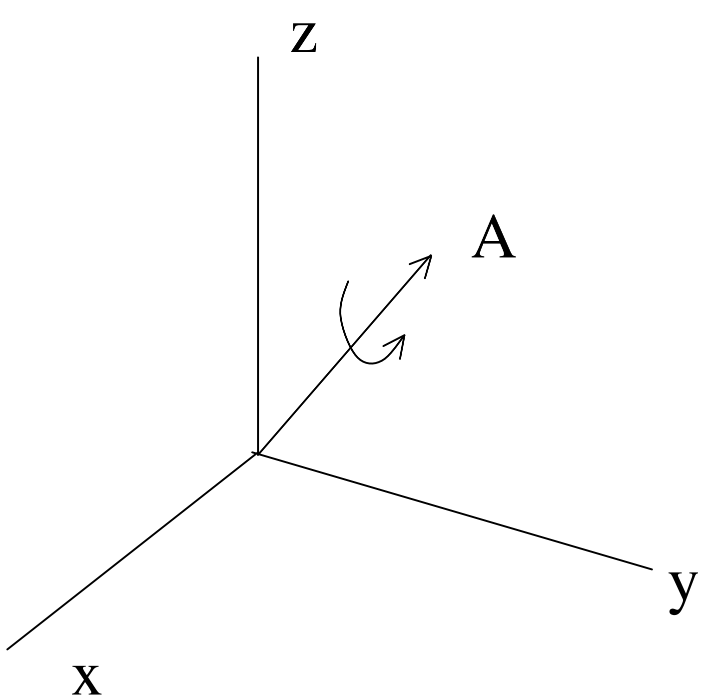
- About z-axis

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

Rotation about an arbitrary axis (likely to be needed for an assignment!)

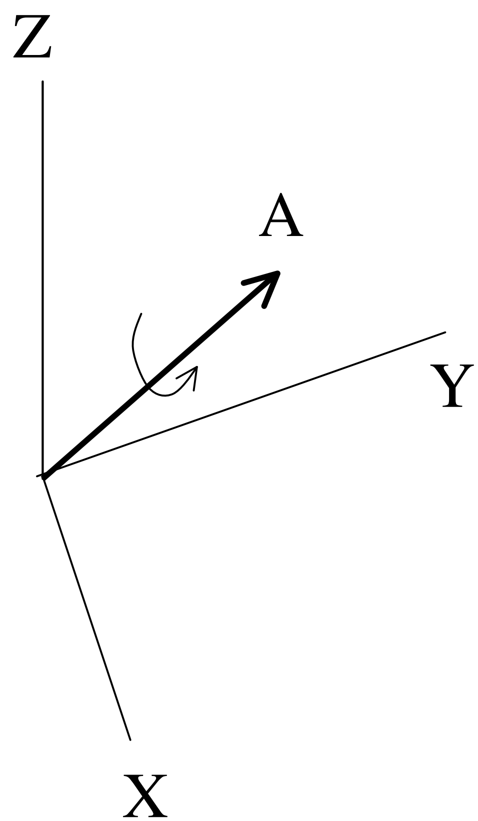


Rotation about an arbitrary axis



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

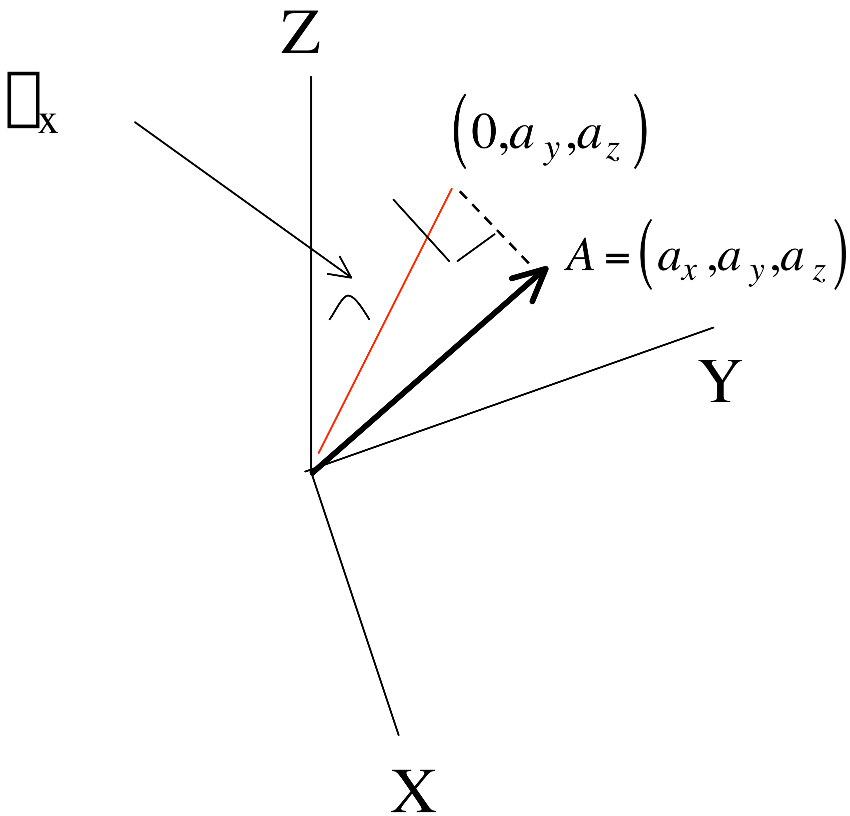
Rotation about an arbitrary axis



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 (assignment hint)



Tricky part:
rotate A to Z axis

Two steps.

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