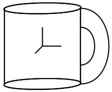
3D Transformations

CMSC 435/634

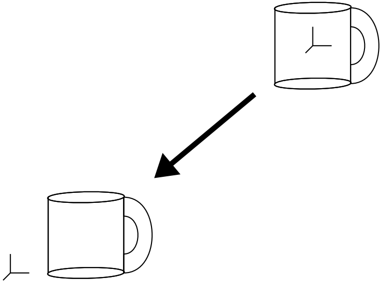
Transformation

Webster: The operation of changing one configuration or expression into another in accordance with a mathematical rule



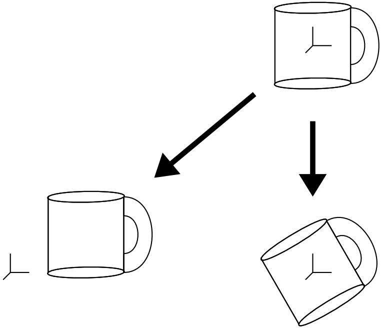
Transformation

Webster: The operation of changing one configuration or expression into another in accordance with a mathematical rule



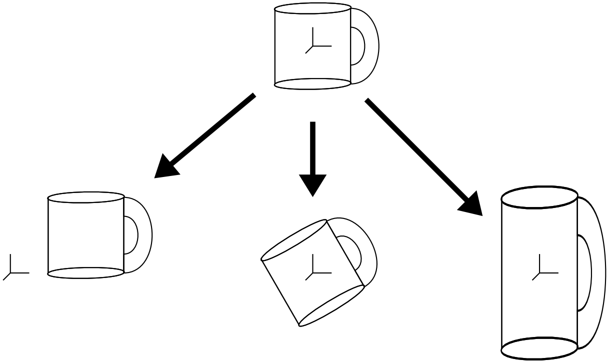
Transformation

Webster: The operation of changing one configuration or expression into another in accordance with a mathematical rule



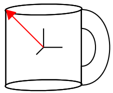
Transformation

Webster: The operation of changing one configuration or expression into another in accordance with a mathematical rule



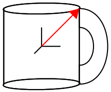
Using Transformation

- Points on object represented as vector offset from origin



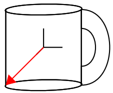
Using Transformation

- Points on object represented as vector offset from origin



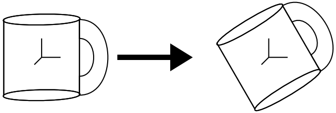
Using Transformation

- Points on object represented as vector offset from origin



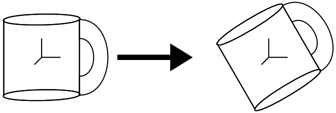
Using Transformation

- Points on object represented as vector offset from origin
- Transform is a vector to vector function
  - $\vec{p'} = f(\vec{p})$



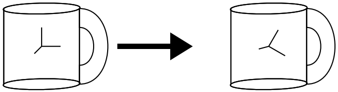
Using Transformation

- Points on object represented as vector offset from origin
- Transform is a vector to vector function
  - $\vec{p'} = f(\vec{p})$
- Relativity:
  - From $\vec{p'}$ point of view, object is transformed



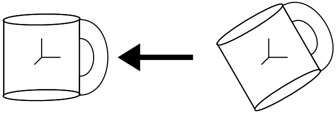
Using Transformation

- Points on object represented as vector offset from origin
- Transform is a vector to vector function
  - $\vec{p'} = f(\vec{p})$
- Relativity:
  - From $\vec{p'}$ point of view, object is transformed
  - From $\vec{p}$ point of view, coordinate system changes



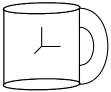
Using Transformation

- Points on object represented as vector offset from origin
- Transform is a vector to vector function
  - $\vec{p'} = f(\vec{p})$
- Relativity:
  - From $\vec{p'}$ point of view, object is transformed
  - From $\vec{p}$ point of view, coordinate system changes
- Inverse transform, $\vec{p} = f^{-1}(\vec{p'})$



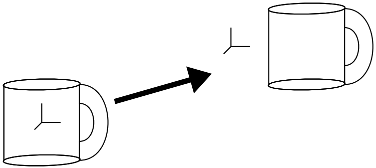
Composing Transforms

- Order matters



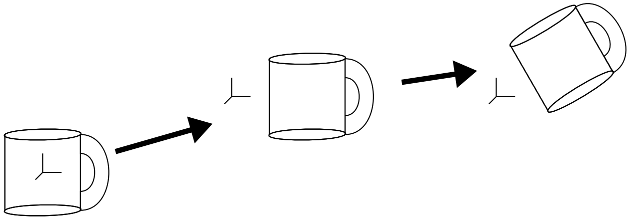
Composing Transforms

- Order matters
  - $T(\vec{p})$



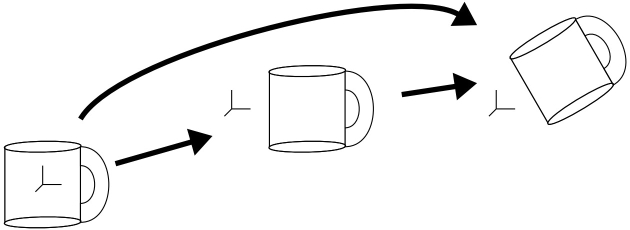
Composing Transforms

- Order matters
  - $R(T(\vec{p}))$



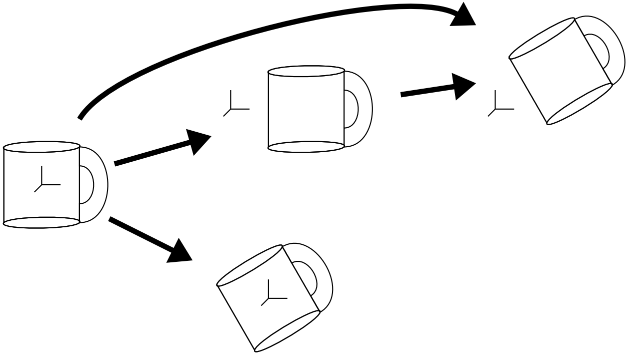
Composing Transforms

- Order matters
  - $R(T(\vec{p})) = R \circ T(\vec{p})$



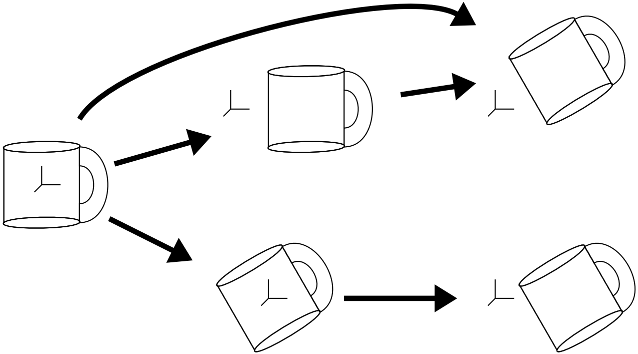
Composing Transforms

- Order matters
    - $R(T(\vec{p})) = R \circ T(\vec{p})$
    - $R(\vec{p})$



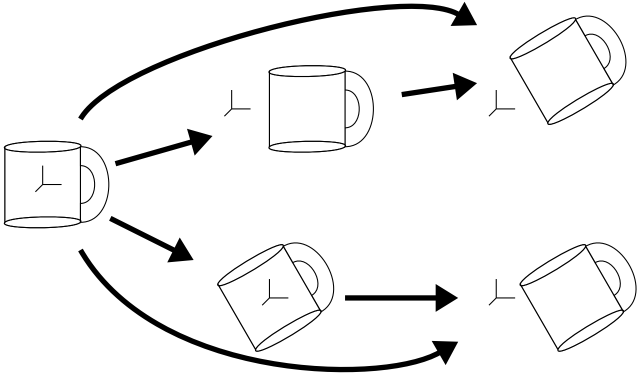
Composing Transforms

- Order matters
  - $R(T(\vec{p})) = R \circ T(\vec{p})$
  - $T(R(\vec{p}))$



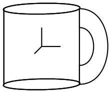
Composing Transforms

- Order matters
  - $R(T(\vec{p})) = R \circ T(\vec{p})$
  - $T(R(\vec{p})) = T \circ R(\vec{p})$



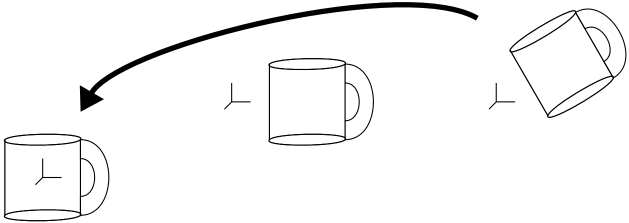
Inverting Composed Transforms

- Reverse order



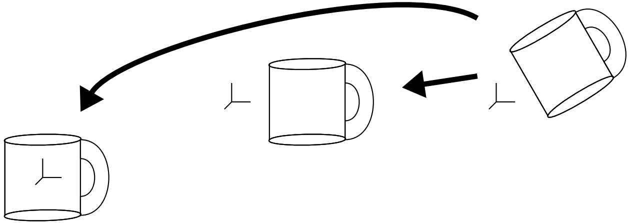
Inverting Composed Transforms

- Reverse order
  - $(R \circ T)^{-1}(\vec{p'})$



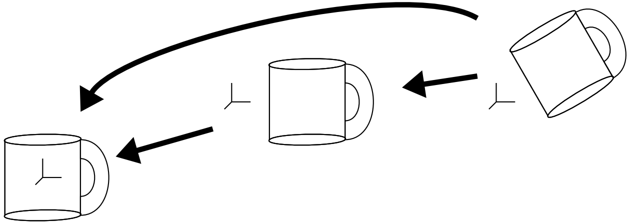
Inverting Composed Transforms

- Reverse order
  - $(R \circ T)^{-1}(\vec{p'}) = \qquad R^{-1}(\vec{p'})$



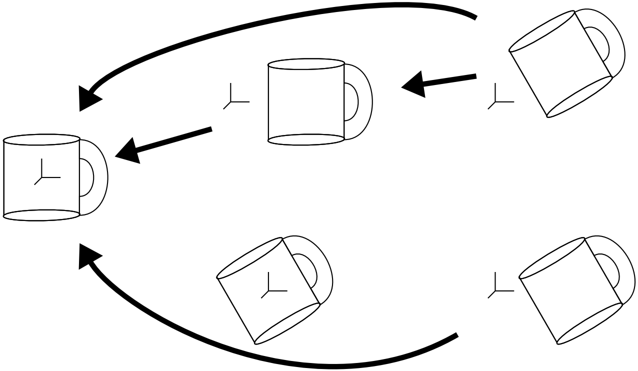
Inverting Composed Transforms

- Reverse order
  - $(R \circ T)^{-1}(\vec{p'}) = T^{-1}(R^{-1}(\vec{p'}))$



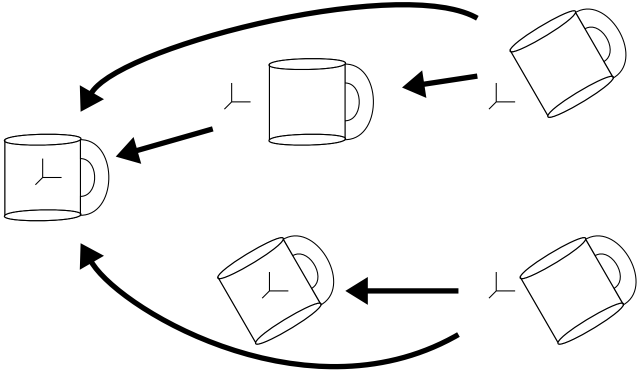
Inverting Composed Transforms

- Reverse order
  - $(R \circ T)^{-1}(\vec{p'}) = T^{-1}(R^{-1}(\vec{p'}))$
  - $(T \circ R)^{-1}(\vec{p'})$



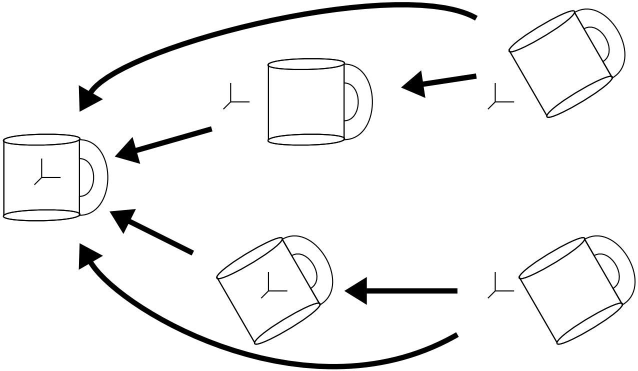
Inverting Composed Transforms

- Reverse order
  - $(R \circ T)^{-1}(\vec{p'}) = T^{-1}(R^{-1}(\vec{p'}))$
  - $(T \circ R)^{-1}(\vec{p'}) = \qquad T^{-1}(\vec{p'})$



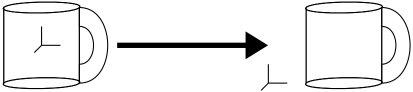
Inverting Composed Transforms

- Reverse order
  - $(R \circ T)^{-1}(\vec{p'}) = T^{-1}(R^{-1}(\vec{p'}))$
  - $(T \circ R)^{-1}(\vec{p'}) = R^{-1}(T^{-1}(\vec{p'}))$



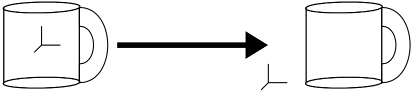
Translation

- $\vec{p'} = \vec{p} + \vec{t}$



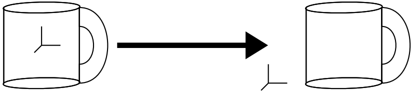
Translation

- $\vec{p'} = \vec{p} + \vec{t}$
- $\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ p_z + t_z \end{bmatrix}$



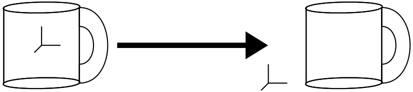
Translation

- $\vec{p'} = \vec{p} + \vec{t}$
- $\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ p_z + t_z \end{bmatrix}$
- $\vec{t}$ says where $\vec{p}$-space origin ends up ($\vec{p'} = \vec{0} + \vec{t}$)



Translation

- $\vec{p'} = \vec{p} + \vec{t}$
- $\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ p_z + t_z \end{bmatrix}$
- $\vec{t}$ says where $\vec{p}$-space origin ends up ($\vec{p'} = \vec{0} + \vec{t}$)
- Composition: $\vec{p'} = (\vec{p} + \vec{t_0}) + \vec{t_1} = \vec{p} + (\vec{t_0} + \vec{t_1})$



Linear Transforms

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Linear Transforms

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$
- Matrix says where $\vec{p}$-space axes end up

Linear Transforms

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$
- Matrix says where $\vec{p}$-space axes end up
  - $$\begin{bmatrix} a \\ d \\ g \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Linear Transforms

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$
- Matrix says where $\vec{p}$-space axes end up
  - $$\begin{bmatrix} b \\ e \\ h \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Linear Transforms

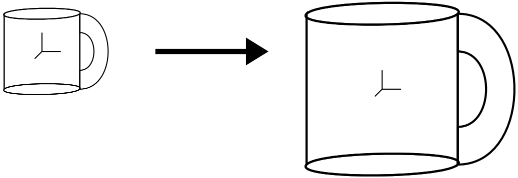
- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$
- Matrix says where $\vec{p}$-space axes end up
  - $$\begin{bmatrix} c \\ f \\ i \end{bmatrix} = \begin{bmatrix} a & b & \color{red}{c} \\ d & e & \color{red}{f} \\ g & h & \color{red}{i} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Linear Transforms

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$
- Matrix says where $\vec{p}$-space axes end up
  - $$\begin{bmatrix} c \\ f \\ i \end{bmatrix} = \begin{bmatrix} a & b & \color{red}{c} \\ d & e & \color{red}{f} \\ g & h & \color{red}{i} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$
  - Composition: $\vec{p'} = M\ (N\ \vec{p}) = (M\ N)\vec{p}$

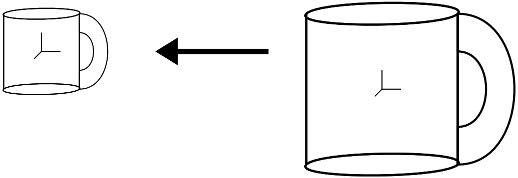
Common case: Scaling

- $\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} s_x & p_x \\ s_y & p_y \\ s_z & p_z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$



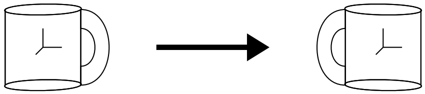
Common case: Scaling

- $\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} s_x\ p_x \\ s_y\ p_y \\ s_z\ p_z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$
- Inverse: $\begin{bmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1/s_z \end{bmatrix}$



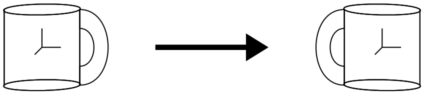
Common case: Reflection

- Negative scaling

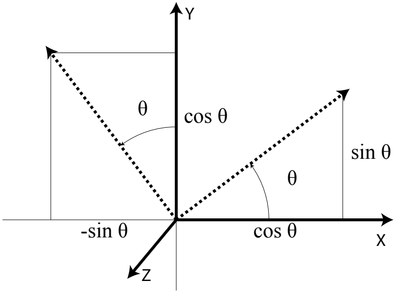


Common case: Reflection

- Negative scaling
- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} -p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

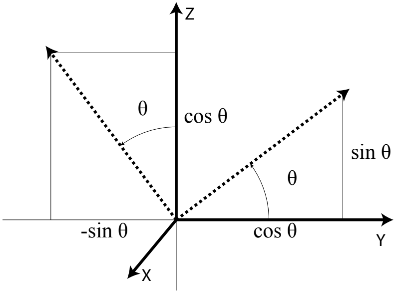


Common case: Rotation



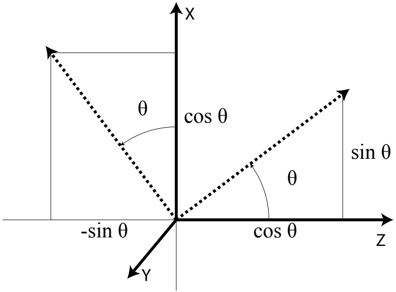
- Orthogonal, so $M^{-1} = M^T$
- Rotate around Z: $\vec{p'} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{p}$

Common case: Rotation



- Orthogonal, so $M^{-1} = M^T$
- Rotate around X: $\vec{p'} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \vec{p}$

Common case: Rotation



- Orthogonal, so $M^{-1} = M^T$
- Rotate around Y: $\vec{p'} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \vec{p}$

Composing Transforms

- Scale by s along axis $\hat{a}$

Composing Transforms

- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with Z

Composing Transforms

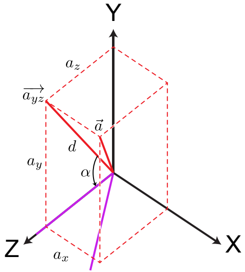
- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with Z
  - Scale along Z

Composing Transforms

- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with Z
  - Scale along Z
  - Rotate back

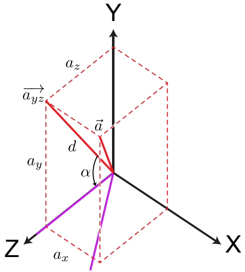
Rotate by α around X into XZ plane

- Projection of $\hat{a}$ onto YZ: $\overrightarrow{a_{yz}} = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix}$



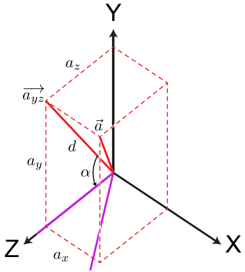
Rotate by α around X into XZ plane

- Projection of $\hat{a}$ onto YZ: $\overrightarrow{a_{yz}} = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix}$
- length $d = \sqrt{(a_y)^2 + (a_z)^2}$



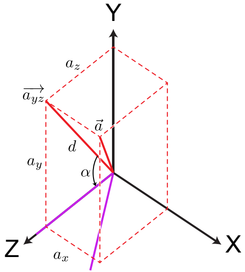
Rotate by α around X into XZ plane

- Projection of $\hat{a}$ onto YZ: $\overrightarrow{a_{yz}} = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix}$
- length $d = \sqrt{(a_y)^2 + (a_z)^2}$
- So $\cos\alpha = a_z/d$, $\sin\alpha = a_y/d$



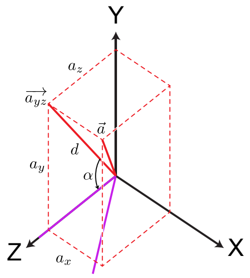
Rotate by α around X into XZ plane

- Projection of $\hat{a}$ onto YZ: $\overrightarrow{a_{yz}} = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix}$
- length $d = \sqrt{(a_y)^2 + (a_z)^2}$
- So $\cos\alpha = a_z/d$, $\sin\alpha = a_y/d$
- $R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_z/d & -a_y/d \\ 0 & a_y/d & a_z/d \end{bmatrix}$



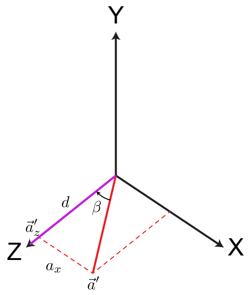
Rotate by α around X into XZ plane

- Projection of $\hat{a}$ onto YZ: $\overrightarrow{a_{yz}} = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix}$
- length $d = \sqrt{(a_y)^2 + (a_z)^2}$
- So $\cos\alpha = a_z/d$, $\sin\alpha = a_y/d$
- $R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a_z/d & -a_y/d \\ 0 & a_y/d & a_z/d \end{bmatrix}$
- Result $\hat{a}' = \begin{bmatrix} a_x \\ 0 \\ d \end{bmatrix}$



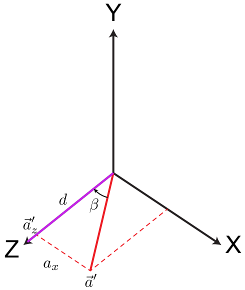
Rotate by -β around Y to Z axis

- $\hat{a}' = \begin{bmatrix} a_x \\ 0 \\ d \end{bmatrix}$



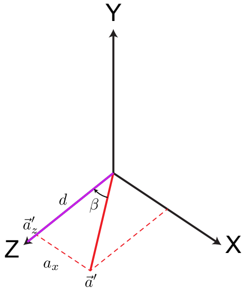
Rotate by -β around Y to Z axis

- $\hat{a}' = \begin{bmatrix} a_x \\ 0 \\ d \end{bmatrix}$
- length $= 1$



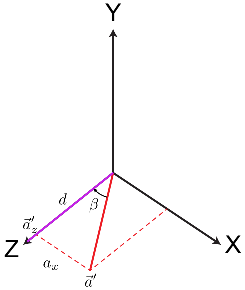
Rotate by -β around Y to Z axis

- $\hat{a}' = \begin{bmatrix} a_x \\ 0 \\ d \end{bmatrix}$
- length $= 1$
- So $\cos\beta = d$, $\sin\beta = a_x$



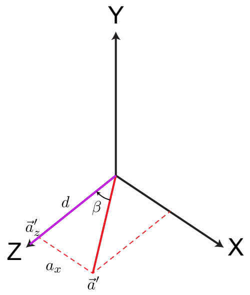
Rotate by -β around Y to Z axis

- $\hat{a}' = \begin{bmatrix} a_x \\ 0 \\ d \end{bmatrix}$
- length $= 1$
- So $\cos\beta = d$, $\sin\beta = a_x$
- $R_Y = \begin{bmatrix} d & 0 & -a_x \\ 0 & 1 & 0 \\ a_x & 0 & d \end{bmatrix}$



Rotate by -β around Y to Z axis

- $\hat{a}' = \begin{bmatrix} a_x \\ 0 \\ d \end{bmatrix}$
- length $= 1$
- So $\cos\beta = d$, $\sin\beta = a_x$
- $R_Y = \begin{bmatrix} d & 0 & -a_x \\ 0 & 1 & 0 \\ a_x & 0 & d \end{bmatrix}$
- Result $\hat{a}'' = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$



Composing Transforms

- Scale by s along Z: $S_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$

Composing Transforms

- Scale by s along Z: $S_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$
- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with XZ plane
  - $R_X \vec{p}$

Composing Transforms

- Scale by s along Z: $S_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$
- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with XZ plane
  - Rotate to align $\hat{a}$ with Z axis
  - $R_Y R_X \vec{p}$

Composing Transforms

- Scale by s along Z: $S_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$
- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with XZ plane
  - Rotate to align $\hat{a}$ with Z axis
  - Scale along Z
  - $\vec{p'} = \qquad S_Z R_Y R_X \vec{p}$

Composing Transforms

- Scale by s along Z: $S_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$
- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with XZ plane
  - Rotate to align $\hat{a}$ with Z axis
  - Scale along Z
  - Undo Z-axis alignment rotation

  - $\vec{p'} = \quad R_Y^{-1} S_Z R_Y R_X \vec{p}$

Composing Transforms

- Scale by s along Z: $S_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$
- Scale by s along axis $\hat{a}$
  - Rotate to align $\hat{a}$ with XZ plane
  - Rotate to align $\hat{a}$ with Z axis
  - Scale along Z
  - Undo Z-axis alignment rotation
  - Undo XZ-plane alignment rotation
  - $\vec{p}' = R_X^{-1} R_Y^{-1} S_Z R_Y R_X \vec{p}$

Affine Transforms

- Affine = Linear + Translation

Affine Transforms

- Affine = Linear + Translation
- Composition? $A\ (B\ \vec{p} + \vec{t_0}) + \vec{t_1} = A\ B\ \vec{p} + A\ \vec{t_0} + \vec{t_1}$
- Yuck!

Homogeneous Coordinates

- Add a '1' to each point

Homogeneous Coordinates

- Add a '1' to each point
- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

- Add a '1' to each point
- $\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$
- $\vec{p'}_x = (a\ p_x + b\ p_y + c\ p_z) + t_x$

Homogeneous Coordinates

- Add a '1' to each point
- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'}_x = (a\ p_x + b\ p_y + c\ p_z) + t_x$
- $\vec{p'}_y = (d\ p_x + e\ p_y + f\ p_z) + t_y$

Homogeneous Coordinates

- Add a '1' to each point
- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'}_x = (a\ p_x + b\ p_y + c\ p_z) + t_x$
- $\vec{p'}_y = (d\ p_x + e\ p_y + f\ p_z) + t_y$
- $\vec{p'}_z = (g\ p_x + h\ p_y + i\ p_z) + t_z$

Homogeneous Coordinates

- Add a '1' to each point
- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'}_x = (a\ p_x + b\ p_y + c\ p_z) + t_x$
- $\vec{p'}_y = (d\ p_x + e\ p_y + f\ p_z) + t_y$
- $\vec{p'}_z = (g\ p_x + h\ p_y + i\ p_z) + t_z$
- $1 = (0p_x + 0p_y + 0p_z) + 1$

Homogeneous Coordinates

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'} = \left[\begin{array}{ccc|c} \vec{x} & \vec{y} & \vec{z} & \vec{t} \end{array}\right] \vec{p}$

Homogeneous Coordinates

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'} = \left[\begin{array}{ccc|c} \vec{x} & \vec{y} & \vec{z} & \vec{t} \end{array}\right] \vec{p}$
  - $\vec{t}$ says where the $\vec{p}$-space origin ends up

Homogeneous Coordinates

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'} = \left[\begin{array}{ccc|c} \vec{x} & \vec{y} & \vec{z} & \vec{t} \end{array}\right] \vec{p}$
  - $\vec{t}$ says where the $\vec{p}$-space origin ends up
  - $\vec{x}$, $\vec{y}$, $\vec{z}$ say where the $\vec{p}$-space axes end up

Homogeneous Coordinates

- $$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
- $\vec{p'} = \left[\begin{array}{ccc|c} \vec{x} & \vec{y} & \vec{z} & \vec{t} \end{array}\right] \vec{p}$
  - $\vec{t}$ says where the $\vec{p}$-space origin ends up
  - $\vec{x}$, $\vec{y}$, $\vec{z}$ say where the $\vec{p}$-space axes end up
- Composition: Just matrix multiplies!

Composing Transforms

- Rotate by θ about line between $\vec{p}_0$ and $\vec{p}_1$:

Composing Transforms

- Rotate by θ about line between $\vec{p}_0$ and $\vec{p}_1$:
  - Translate $\vec{p}_0$ to origin
- T

Composing Transforms

- Rotate by θ about line between $\vec{p}_0$ and $\vec{p}_1$:
  - Translate $\vec{p}_0$ to origin
  - Rotate to align $\vec{p}_1 - \vec{p}_0$ with Z

- $$R_Y R_X T$$

Composing Transforms

- Rotate by θ about line between $\vec{p}_0$ and $\vec{p}_1$:
  - Translate $\vec{p}_0$ to origin
  - Rotate to align $\vec{p}_1 - \vec{p}_0$ with Z
  - Rotate by θ around Z
- $$R_Z(\theta) R_Y R_X T$$

Composing Transforms

- Rotate by θ about line between $\vec{p}_0$ and $\vec{p}_1$:
  - Translate $\vec{p}_0$ to origin
  - Rotate to align $\vec{p}_1 - \vec{p}_0$ with Z
  - Rotate by θ around Z
  - Undo $\vec{p}_1 - \vec{p}_0$ rotation
- $$R_X^{-1} R_Y^{-1} R_Z(\theta) R_Y R_X T$$

Composing Transforms

- Rotate by θ about line between $\vec{p}_0$ and $\vec{p}_1$:
  - Translate $\vec{p}_0$ to origin
  - Rotate to align $\vec{p}_1 - \vec{p}_0$ with Z
  - Rotate by θ around Z
  - Undo $\vec{p}_1 - \vec{p}_0$ rotation
  - Undo translation
- $T^{-1} R_X^{-1} R_Y^{-1} R_Z(\theta) R_Y R_X T$

Vectors

- Transform by *Jacobian Matrix*

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$
  - $J = \begin{bmatrix} \partial p'_x/\partial p_x & \partial p'_x/\partial p_y & \partial p'_x/\partial p_z \\ \partial p'_y/\partial p_x & \partial p'_y/\partial p_y & \partial p'_y/\partial p_z \\ \partial p'_z/\partial p_x & \partial p'_z/\partial p_y & \partial p'_z/\partial p_z \end{bmatrix}$

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$
  - $J = \begin{bmatrix} a & \partial p'_x / \partial p_y & \partial p'_x / \partial p_z \\ \partial p'_y / \partial p_x & \partial p'_y / \partial p_y & \partial p'_y / \partial p_z \\ \partial p'_z / \partial p_x & \partial p'_z / \partial p_y & \partial p'_z / \partial p_z \end{bmatrix}$

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$
  - $J = \begin{bmatrix} a & b & \partial p'_x / \partial p_z \\ \partial p'_y / \partial p_x & \partial p'_y / \partial p_y & \partial p'_y / \partial p_z \\ \partial p'_z / \partial p_x & \partial p'_z / \partial p_y & \partial p'_z / \partial p_z \end{bmatrix}$

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$
  - $J = \begin{bmatrix} a & b & c \\ \partial p'_y/\partial p_x & \partial p'_y/\partial p_y & \partial p'_y/\partial p_z \\ \partial p'_z/\partial p_x & \partial p'_z/\partial p_y & \partial p'_z/\partial p_z \end{bmatrix}$

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$
  - $J = \begin{bmatrix} a & b & c \\ d & e & f \\ \partial p'_z/\partial p_x & \partial p'_z/\partial p_y & \partial p'_z/\partial p_z \end{bmatrix}$

Vectors

- Transform by *Jacobian Matrix*
- Matrix of partial derivatives
  - $\begin{bmatrix} \vec{p'}_x \\ \vec{p'}_y \\ \vec{p'}_z \end{bmatrix} = \begin{bmatrix} a\ p_x + b\ p_y + c\ p_z + t_x \\ d\ p_x + e\ p_y + f\ p_z + t_y \\ g\ p_x + h\ p_y + i\ p_z + t_z \end{bmatrix}$
  - $J = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$
- *Upper-left 3x3*

Normals

- Normal should remain perpendicular to tangent vectors

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n_x & n_y & n_z \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = 0$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n_x & n_y & n_z \end{bmatrix} I \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = 0$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n_x & n_y & n_z \end{bmatrix} (J^{-1} J) \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = 0$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\left(\begin{bmatrix} n_x & n_y & n_z \end{bmatrix} J^{-1} \right) \left(J \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \right) = 0$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\left(\begin{bmatrix} n_x & n_y & n_z \end{bmatrix} J^{-1} \right) \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = 0$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n'_x & n'_y & n'_z \end{bmatrix} \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = 0$
- $\vec{n'} = \vec{n} J^{-1}$

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n'_x & n'_y & n'_z \end{bmatrix} \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = 0$
- $\vec{n'} = \vec{n} J^{-1}$
- Multiply by inverse on right

Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n'_x & n'_y & n'_z \end{bmatrix} \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = 0$
- $\vec{n'} = \vec{n} J^{-1}$
- Multiply by inverse on right
- OR multiply *column* normal by inverse transpose

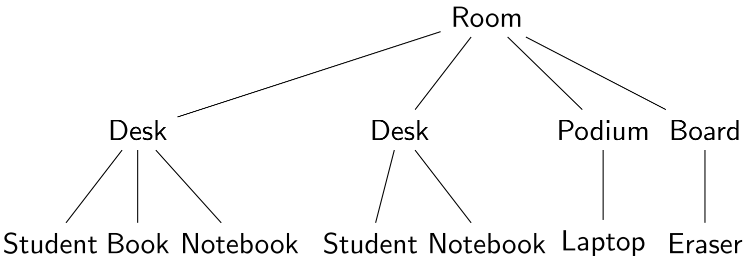
Normals

- Normal should remain perpendicular to tangent vectors
- $\vec{n} \cdot \vec{v} = \vec{n'} \cdot \vec{v'} = 0$
- $\begin{bmatrix} n'_x & n'_y & n'_z \end{bmatrix} \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = 0$
- $\vec{n'} = \vec{n} J^{-1}$
- Multiply by inverse on right
- OR multiply *column* normal by inverse transpose
  - $(J^{-1})^T = J$ if J is orthogonal (only rotations)

Coordinate System / Space

- Origin + Axes
- Reference frame
- Convert by matrix
- OpenGL convention (we use this!): Points are columns
  - $\vec{p}_{table} = TableFromPencil\ \vec{p}_{pencil}$
  - $\vec{p}_{room} = RoomFromTable\ TableFromPencil\ \vec{p}_{pencil}$
  - $\vec{p}_{room} = RoomFromPencil\ \vec{p}_{pencil}$
- Same thing in D3D convention (Points are rows, everything transposed)
  - $\vec{p}_{table} = \vec{p}_{pencil}\ PencilToTable$
  - $\vec{p}_{room} = \vec{p}_{pencil}\ PencilToTable\ TableToRoom$
  - $\vec{p}_{room} = \vec{p}_{pencil}\ PencilToRoom$

Nesting


Room
Desk
Desk
Podium
Board
Student
Book
Notebook
Student
Notebook
Laptop
Eraser

Matrix Stack

- Remember transformation, return to it later
- Push a copy, modify the copy, pop
- Keep matrix and update matrix and inverse
- Push and pop both matrix and inverse together

code	**stack** (start with Identity)
`transform(WorldFromRoom);`	WfR
`push;`	WfR WfR
`transform(RoomFromDesk);`	WfD WfR
`push;`	WfD WfD WfR
`transform(DeskFromStudent);`	WfS WfD WfR
`pop;`	WfD WfR
`push;`	WfD WfD WfR
`transform(DeskFromBook);`	WfB WfD WfR
...	

Common Spaces

- Object / Model
  - Logical coordinates for modeling
  - May have several more levels
- World
  - Common coordinates for everything
- View / Camera / Eye
  - eye/camera at (0,0,0), looking down Z (or -Z) axis
  - planes: left, right, top, bottom, near/hither, far/yon
- Normalized Device Coordinates (NDC) / Clip
  - Visible portion of scene from (-1,-1,-1) to (1,1,1)
  - Sometimes one or more components 0 to 1 instead of -1 to 1
- Raster / Pixel / Viewport
  - 0,0 to x-resolution, y-resolution
- Device / Screen
  - May translate to fit actual screen

Model→World / Model→View

- Model→World
  - All shading and rendering in World space
  - Transform all objects and lights
- Ray tracing implicitly does World→Raster
- Model→View
  - World can be any common space, might as well use View space
  - Serves just as well for single view
- World centered on viewer, but aligned with world
  - UE4: *TranslatedWorld*
  - Floating point precision better near origin
  - Helpful for huge worlds

World→View

- Also called Viewing or Camera transform
- LookAt
  - $\overrightarrow{from}$, $\overrightarrow{to}$, $\overrightarrow{up}$
  - $\left[\begin{array}{c|c|c|c} \vec{u} & \vec{v} & \vec{w} & \overrightarrow{from} \end{array} \right]$
- Roll / Pitch / Yaw
  - Translate to camera center, rotate around camera
  - $R_z \; R_x \; R_y \; T$
  - Can have gimbal lock when first and last axes align
- Orbit
  - Rotate around object center, translate out
  - $T \; R_z \; R_x \; R_y$
  - Also can have gimbal lock

View→NDC

- Also called *Projection* transform
- Orthographic / Parallel
  - Translate & Scale to view volume
  - $$\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Perspective
  - More complicated...

NDC→Raster

- Also called *Viewport* transform
- $[-1,1],[-1,1],[-1,1] \to [0,n_x],[0,n_y],[0,n_z]$
- or $\to [-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$

NDC→Raster

- Also called *Viewport* transform
- $[-1, 1], [-1, 1], [-1, 1] \rightarrow [0, n_x], [0, n_y], [0, n_z]$
- or $\rightarrow [-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$
  - Translate to $[0, 2], [0, 2], [0, 2]$

NDC→Raster

- Also called *Viewport* transform
- $[-1,1],[-1,1],[-1,1] \rightarrow [0,n_x],[0,n_y],[0,n_z]$
- or $\rightarrow [-\frac{1}{2}, n_x - \frac{1}{2}],[-\frac{1}{2}, n_y - \frac{1}{2}],[-\frac{1}{2}, n_z - \frac{1}{2}]$
  - Translate to $[0,2],[0,2],[0,2]$
  - Scale to $[0,n_x],[0,n_y],[0,n_z]$

NDC→Raster

- Also called *Viewport* transform
- $[-1, 1], [-1, 1], [-1, 1] \rightarrow [0, n_x], [0, n_y], [0, n_z]$
- or $\rightarrow [-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$
  - Translate to $[0, 2], [0, 2], [0, 2]$
  - Scale to $[0, n_x], [0, n_y], [0, n_z]$
  - (if needed) Translate to $[-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$

NDC→Raster

- Also called *Viewport* transform
- $[-1,1],[-1,1],[-1,1] \to [0,n_x],[0,n_y],[0,n_z]$
- or $\to [-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$
  - Translate to $[0,2],[0,2],[0,2]$
  - Scale to $[0,n_x],[0,n_y],[0,n_z]$
  - (if needed) Translate to $[-\frac{1}{2}, n_x - \frac{1}{2}], [-\frac{1}{2}, n_y - \frac{1}{2}], [-\frac{1}{2}, n_z - \frac{1}{2}]$

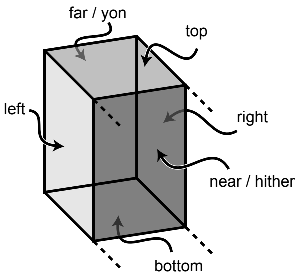
$$\begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y}{2} \\ 0 & 0 & \frac{n_z}{2} & \frac{n_z}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & \frac{n_z}{2} & \frac{n_z-1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Raster→Screen

- Usually just a translation
  - Some game consoles include scaling for performance
  - More complicated for tiled displays, domes, etc.
- Usually handled by windowing system

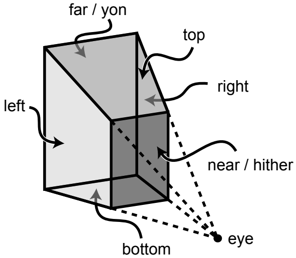
Perspective View Frustum

- Orthographic view volume is a rectangular volume



Perspective View Frustum

- Orthographic view volume is a rectangular volume
- Perspective is a truncated pyramid or *frustum*



Perspective Transform

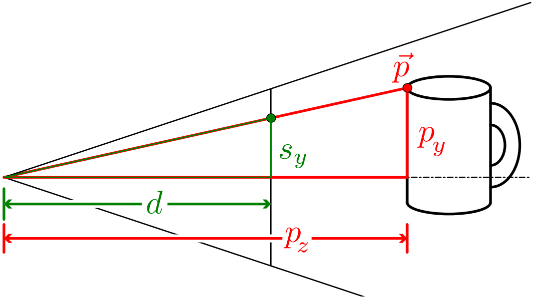
- Ray tracing
  - Given screen (s_x, s_y), parameterize all points $\vec{p}$

Perspective Transform

- Ray tracing
  - Given screen (s_x, s_y), parameterize all points $\vec{p}$
- Perspective Transform
  - Given $\vec{p}$, find (s_x, s_y)

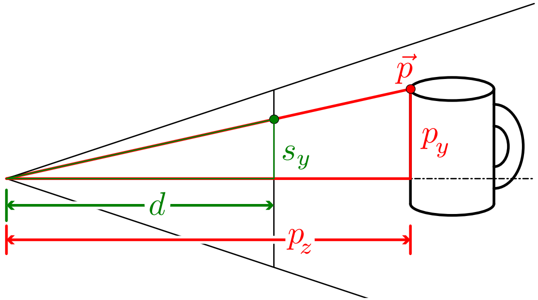
Perspective Transform

- Ray tracing
  - Given screen (s_x, s_y), parameterize all points $\vec{p}$
- Perspective Transform
  - Given $\vec{p}$, find (s_x, s_y)
  - Use similar triangles



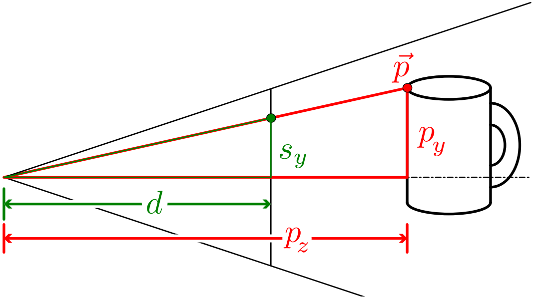
Perspective Transform

- Ray tracing
  - Given screen (s_x, s_y), parameterize all points $\vec{p}$
- Perspective Transform
  - Given $\vec{p}$, find (s_x, s_y)
  - Use similar triangles
  - $s_y / d = p_y / p_z$



Perspective Transform

- Ray tracing
  - Given screen (s_x, s_y), parameterize all points $\vec{p}$
- Perspective Transform
  - Given $\vec{p}$, find (s_x, s_y)
  - Use similar triangles
  - $s_y/d = p_y/p_z$ So $s_y = d\ p_y/p_z$



Homogeneous Equations

- Same total degree for every term

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\text{a}\,X + \text{b}\,Y + \text{c} = 0$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}X + \mathrm{b}Y + \mathrm{c} = 0$
  - $X = x/w, Y = y/w$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}\,X + \mathrm{b}\,Y + \mathrm{c} = 0$
  - $X = x/w, Y = y/w$
  - $\mathrm{a}\,x/w + \mathrm{b}\,y/w + \mathrm{c} = 0$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}\,X + \mathrm{b}\,Y + \mathrm{c} = 0$
  - $X = x/w, Y = y/w$
  - $\mathrm{a}\,x/w + \mathrm{b}\,y/w + \mathrm{c} = 0$
  - $\rightarrow \mathrm{a}\,x + \mathrm{b}\,y + \mathrm{c}\,w = 0$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}\,X + \mathrm{b}\,Y + \mathrm{c} = 0$
  - $X = x/w, Y = y/w$
  - $\mathrm{a}\,x/w + \mathrm{b}\,y/w + \mathrm{c} = 0$
  - $\rightarrow \mathrm{a}\,x + \mathrm{b}\,y + \mathrm{c}\,w = 0$
- Quadric

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}\,X + \mathrm{b}\,Y + \mathrm{c} = 0$
  - $X = x/w, Y = y/w$
  - $\mathrm{a}\,x/w + \mathrm{b}\,y/w + \mathrm{c} = 0$
  - $\rightarrow \mathrm{a}\,x + \mathrm{b}\,y + \mathrm{c}\,w = 0$
- Quadric
  - $\mathrm{a}\,X^2 + \mathrm{b}\,X\,Y + \mathrm{c}\,Y^2 + \mathrm{d}\,X + \mathrm{e}\,Y + \mathrm{f} = 0$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $a X + b Y + c = 0$
  - $X = x/w, Y = y/w$
  - $a x/w + b y/w + c = 0$
  - $\rightarrow a x + b y + c w = 0$
- Quadric
  - $a X^2 + b X Y + c Y^2 + d X + e Y + f = 0$
  - $X = x/w, Y = y/w$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}\,X + \mathrm{b}\,Y + \mathrm{c} = 0$
  - $X = x/w$, $Y = y/w$
  - $\mathrm{a}\,x/w + \mathrm{b}\,y/w + \mathrm{c} = 0$
  - $\rightarrow \mathrm{a}\,x + \mathrm{b}\,y + \mathrm{c}\,w = 0$
- Quadric
  - $\mathrm{a}\,X^2 + \mathrm{b}\,X\,Y + \mathrm{c}\,Y^2 + \mathrm{d}\,X + \mathrm{e}\,Y + \mathrm{f} = 0$
  - $X = x/w$, $Y = y/w$
  - $\mathrm{a}\,x^2/w^2 + \mathrm{b}\,x\,y/w^2 + \mathrm{c}\,y^2/w^2 + \mathrm{d}\,x/w + \mathrm{e}\,y/w + \mathrm{f} = 0$

Homogeneous Equations

- Same total degree for every term
- Introduce a new redundant variable
- Plane equation
  - $\mathrm{a}\,X + \mathrm{b}\,Y + \mathrm{c} = 0$
  - $X = x/w,\ Y = y/w$
  - $\mathrm{a}\,x/w + \mathrm{b}\,y/w + \mathrm{c} = 0$
  - $\rightarrow \mathrm{a}\,x + \mathrm{b}\,y + \mathrm{c}\,w = 0$
- Quadric
  - $\mathrm{a}\,X^2 + \mathrm{b}\,X\,Y + \mathrm{c}\,Y^2 + \mathrm{d}\,X + \mathrm{e}\,Y + \mathrm{f} = 0$
  - $X = x/w,\ Y = y/w$
  - $\mathrm{a}\,x^2/w^2 + \mathrm{b}\,x\,y/w^2 + \mathrm{c}\,y^2/w^2 + \mathrm{d}\,x/w + \mathrm{e}\,y/w + \mathrm{f} = 0$
  - $\rightarrow \mathrm{a}\,x^2 + \mathrm{b}\,x\,y + \mathrm{c}\,y^2 + \mathrm{d}\,x\,w + \mathrm{e}\,y\,w + \mathrm{f}\,w^2 = 0$

Homogeneous Coordinates

- Rather than $(x, y, z, 1)$, use (x, y, z, w)
- Real 3D point is $(X, Y, Z) = (x/w, y/w, z/w)$
- Can represent Perspective Transform as 4x4 matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ p_z/d \end{bmatrix} \rightarrow \begin{bmatrix} d\, p_x/p_z \\ d\, p_y/p_z \\ d \end{bmatrix}$$

Homogeneous Depth

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ p_z/d \end{bmatrix} \rightarrow \begin{bmatrix} d\,p_x/p_z \\ d\,p_y/p_z \\ d \end{bmatrix}$$

- Lose depth information
- Can't get $d\,p'_z/p_z = p_z$
  - Plus $x/z, y/z, z$ isn't linear
- Use *Projective Geometry*

Projective Geometry

- If x, y, z lie on a plane, $x/z, y/z, 1/z$ also lie on a plane
- $1/z$ is strictly ordered: if $z_1 < z_2$, then $1/z_1 > 1/z_2$
- New matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ 1 \\ p_z \end{bmatrix} \rightarrow \begin{bmatrix} p_x/p_z \\ p_y/p_z \\ 1/p_z \end{bmatrix}$$

Getting Fancy

- Add scale & translate
  - Field of view
  - near/far range

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & b & c \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} a\,p_x \\ a\,p_y \\ b\,p_z + c \\ -p_z \end{bmatrix} \rightarrow \begin{bmatrix} -a\,p_x/p_z \\ -a\,p_y/p_z \\ -b - c/p_z \end{bmatrix}$$

- $a = \text{cotan}(fov/2)$

Getting Fancy

- Add scale & translate
  - Field of view
  - near/far range

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & b & c \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} a\,p_x \\ a\,p_y \\ b\,p_z + c \\ -p_z \end{bmatrix} \rightarrow \begin{bmatrix} -a\,p_x/p_z \\ -a\,p_y/p_z \\ -b - c/p_z \end{bmatrix}$$

- $a = \text{cotan}(fov/2)$
- OpenGL convention: Solve for $n \rightarrow -1$ and $f \rightarrow 1$

Getting Fancy

- Add scale & translate
  - Field of view
  - near/far range

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & b & c \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} a\,p_x \\ a\,p_y \\ b\,p_z + c \\ -p_z \end{bmatrix} \rightarrow \begin{bmatrix} -a\,p_x/p_z \\ -a\,p_y/p_z \\ -b - c/p_z \end{bmatrix}$$

- $a = \text{cotan}(fov/2)$
- OpenGL convention: Solve for $n \rightarrow -1$ and $f \rightarrow 1$
  - $b = (n + f)/(n - f)$; $c = (2\,n\,f)/(f - n)$

Getting Fancy

- Add scale & translate
  - Field of view
  - near/far range

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & b & c \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} a\,p_x \\ a\,p_y \\ b\,p_z + c \\ -p_z \end{bmatrix} \rightarrow \begin{bmatrix} -a\,p_x/p_z \\ -a\,p_y/p_z \\ -b - c/p_z \end{bmatrix}$$

- $a = \text{cotan}(fov/2)$
- OpenGL convention: Solve for $n \rightarrow -1$ and $f \rightarrow 1$
  - $b = (n + f)/(n - f)$; $c = (2\,n\,f)/(f - n)$
- D3D convention: Solve for $n \rightarrow 0$ and $f \rightarrow 1$

Getting Fancy

- Add scale & translate
  - Field of view
  - near/far range

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & b & c \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} a\,p_x \\ a\,p_y \\ b\,p_z + c \\ -p_z \end{bmatrix} \rightarrow \begin{bmatrix} -a\,p_x/p_z \\ -a\,p_y/p_z \\ -b - c/p_z \end{bmatrix}$$

- $a = \text{cotan}(fov/2)$
- OpenGL convention: Solve for $n \rightarrow -1$ and $f \rightarrow 1$
  - $b = (n+f)/(n-f)$; $c = (2\,n\,f)/(f-n)$
- D3D convention: Solve for $n \rightarrow 0$ and $f \rightarrow 1$
  - $b = f/(n-f)$; $c = n\,f/(f-n)$