# Geometric Transformations



## Linear Algebra Review

- Matrices
  - notation
  - basic operations
  - matrix-vector multiplication



## Matrices

- Notation for matrices and vectors
  - use column form for vectors

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} m_{ij} \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T$$



## Matrix Operations

- Addition

$$T = M + N \Rightarrow [t_{ij}] = [m_{ij} + n_{ij}]$$

- Scalar Multiply

$$T = \alpha M \Rightarrow [t_{ij}] = [\alpha m_{ij}]$$

- Transpose

## Matrix Operations

- Matrix-Matrix Multiply
  - row-column multiplication
  - not commutative
  - associative

$$T = MN \Rightarrow [t_{ij}] = \left[\sum_k m_{ik} n_{kj}\right]$$

$$\begin{bmatrix} \boxed{t_{11}} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} = \begin{bmatrix} \boxed{m_{11} \quad m_{12}} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} \boxed{n_{11}} & n_{12} \\ \boxed{n_{21}} & n_{22} \end{bmatrix}$$



## Matrix Operations

- Matrix-Vector Multiply
  - row-column multiplication

$$\mathbf{u} = M\mathbf{v} \Rightarrow [u_i] = \left[\sum_k m_{ik} v_k\right]$$

$$\begin{bmatrix} \boxed{u_1} \\ u_2 \end{bmatrix} = \begin{bmatrix} \boxed{m_{11} \quad m_{12}} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} \boxed{v_1} \\ \boxed{v_2} \end{bmatrix}$$



## Special Matrices

- Identity

$$I = [i_{ij}] = \begin{cases} 1 & \text{for i = j} \\ 0 & \text{for i} \neq \text{j} \end{cases} \qquad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$MI = IM = M$$

- Zero

$$O = [o_{ij}] = 0 \qquad O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$M + O = M$$



## Matrix Operations

- Transpose
  - flip along the diagonal

$$T = M^T \Rightarrow [t_{ij}] = [m_{ji}]$$

- Inverse
  - will not compute explicitly in this course

$$T = M^{-1} \Rightarrow TM = MM^{-1} = M^{-1}M = I$$



## Matrix Operations Properties

- Linearity of multiplication

  $\alpha(A+B) = \alpha A + \alpha B$

  $M(A+B) = MA + MB$
- Associativity of multiplication

  $A(BC) = (AB)C$
- Transpose and Inverse of Matrix Multiply

  $(AB)^T = B^T A^T$

  $(AB)^{-1} = B^{-1} A^{-1}$



## Geometric Transformation

- Function that maps points to points

$$\mathbf{p} \rightarrow \mathbf{p}' = X(\mathbf{p})$$

- Different transformations have restriction on the form of *M*
  - we will look at linear, affine and projections

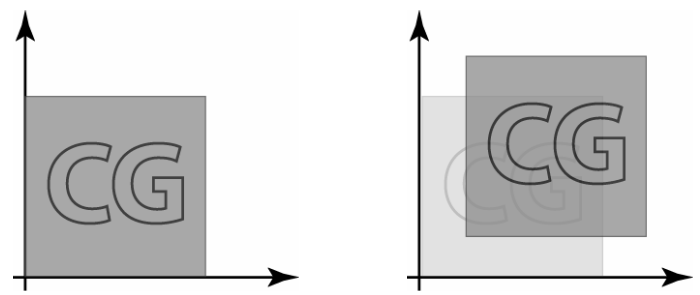


## 2D Transformations



## Translation

- Simplest form $T_{\mathbf{t}}(\mathbf{p}) = \mathbf{p} + \mathbf{t}$
- Inverse $T_{\mathbf{t}}^{-1}(\mathbf{p}) = T_{-\mathbf{t}}(\mathbf{p}) = \mathbf{p} - \mathbf{t}$



## Linear Transformation

- fundamental property

  $X(\alpha\mathbf{p}+\beta\mathbf{q})=\alpha X(\mathbf{p})+\beta X(\mathbf{q})$
- can be represented in matrix form

  $X(\mathbf{p})=M\mathbf{p}$
- other properties
  - maps origin to origin
  - maps lines to lines
  - parallel lines remain parallel
  - length ratios are preserved
  - closed under composition

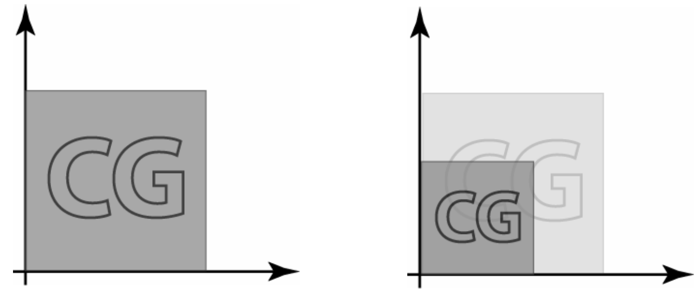




## Uniform Scale

$$S_s\mathbf{p}=\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}\begin{bmatrix} p_x \\ p_y \end{bmatrix}=\begin{bmatrix} sp_x \\ sp_y \end{bmatrix}$$

$$S_s^{-1}=S_{1/s}$$



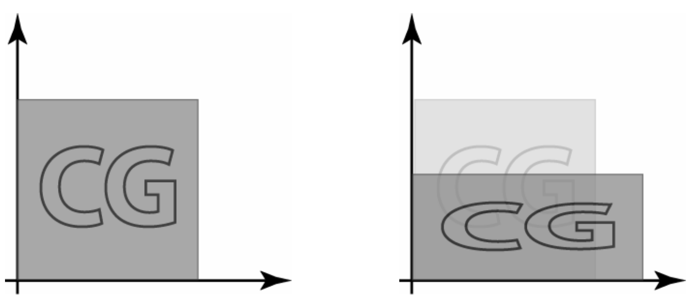





## Non-uniform Scale

$$S_\mathbf{s}\mathbf{p}=\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}\begin{bmatrix} p_x \\ p_y \end{bmatrix}=\begin{bmatrix} s_x p_x \\ s_y p_y \end{bmatrix}$$

$$S_\mathbf{s}^{-1}=S_{(1/s_x,1/s_y)}$$



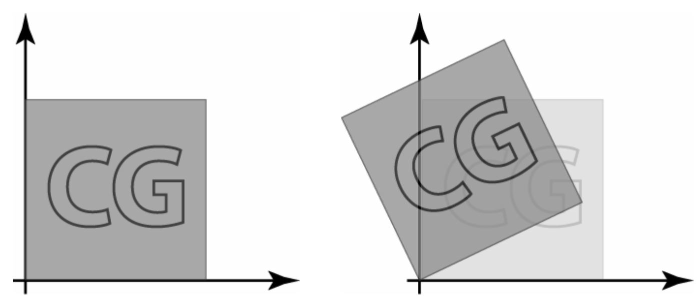





## Rotation

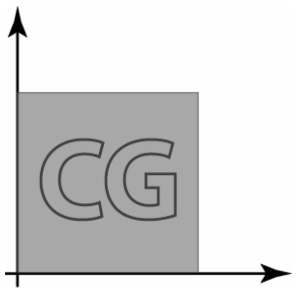
$$R_\theta\mathbf{p}=\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} p_x \\ p_y \end{bmatrix}=\begin{bmatrix} p_x\cos\theta-p_y\sin\theta \\ p_x\sin\theta+p_y\cos\theta \end{bmatrix}$$

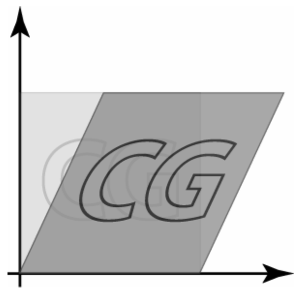
$$R_\theta^{-1}=R_{-\theta}$$



## Shear

$$Sh_{\mathbf{s}}\mathbf{p} = \begin{bmatrix} 1 & s_x \\ s_y & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} p_x + s_x p_y \\ s_y p_x + p_y \end{bmatrix}$$



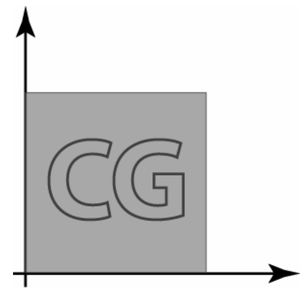



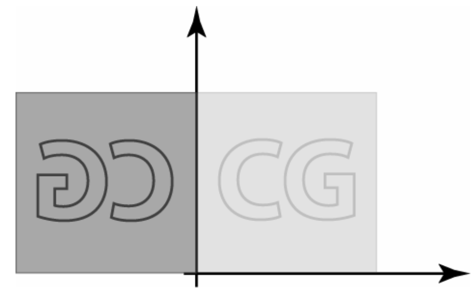



## Reflection

$$Rl_x\mathbf{p} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} -p_x \\ p_y \end{bmatrix} \qquad Rl_y\mathbf{p} = \ldots$$

$$Rl_o\mathbf{p} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} -p_x \\ -p_y \end{bmatrix}$$






## Combining translation and linear transforms

- represent linear together with translation
  - rigid body transformation are a subset of this

$$X_{M,\mathbf{t}}(\mathbf{p}) = M\mathbf{p} + \mathbf{t}$$

- goal: unified format for all transformations



## Homogeneous coordinates

- represent points with 1 additional coordinate *w*
  - set it to 1 for points

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_w \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

## Homogeneous coordinates

- represent translation with a 3x3 matrix

$$T_{\mathbf{t}}\mathbf{p} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ 1 \end{bmatrix}$$

- add one row and column to linear transforms

$$M\mathbf{p} = \begin{bmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}p_x + m_{12}p_y \\ m_{21}p_x + m_{22}p_y \\ 1 \end{bmatrix}$$



## Affine Transformations

- combining linear and translation in one matrix

$$T\mathbf{p} = M\mathbf{p} + \mathbf{t} = \begin{bmatrix} M & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

- properties
  - *does not map origin to origin*
  - maps lines to lines
  - parallel lines remain parallel
  - length ratios are preserved
  - closed under composition



## Affine Transformations

translation

$$T_{\mathbf{t}} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

scale

$$S_{\mathbf{s}} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

rotation

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



## Compositing transformations

- applying one transformation after another
  - expressed by function composition

$$\mathbf{p}' = X_2(X_1(\mathbf{p})) = (X_2 \circ X_1)(\mathbf{p})$$

- for the transforms presented before, computed by matrix multiplication

$$(X_2 \circ X_1)(\mathbf{p}) = X_2(X_1(\mathbf{p})) = M_2(M_1\mathbf{p}) = (M_2 M_1)(\mathbf{p})$$

## Compositing transformations

- translation

$$\begin{bmatrix} 0 & \mathbf{t}_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \mathbf{t}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{t}_1 + \mathbf{t}_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

- linear transformations

$$\begin{bmatrix} M_2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} M_2 M_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

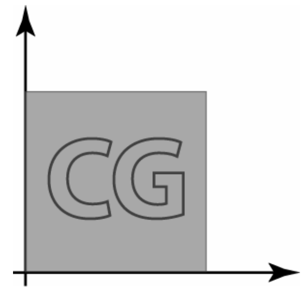
- affine transformations

$$\begin{bmatrix} M_2 & \mathbf{t}_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M_1 & \mathbf{t}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} M_2 M_1 & M_2 \mathbf{t}_1 + \mathbf{t}_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

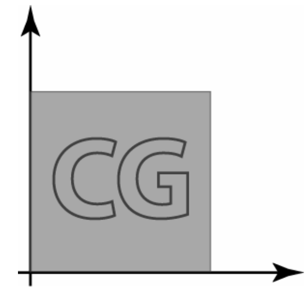


## Compositing Transformations

- composition is not commutative




rotate,
then translate

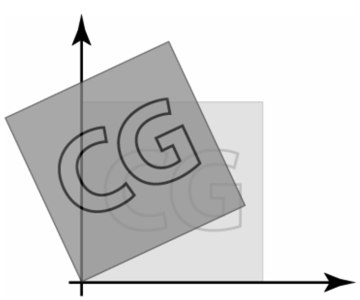



translate,
then rotate

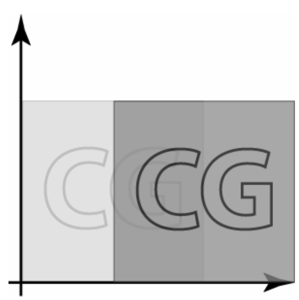


## Compositing Transformations

- composition is not commutative




rotate,
then translate

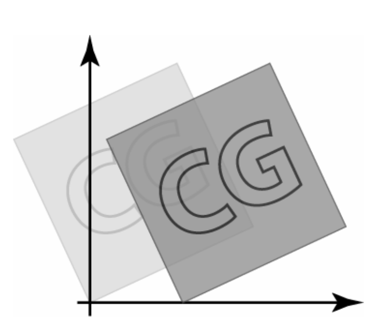



translate,
then rotate

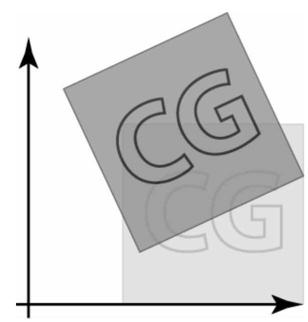


## Compositing Transformations

- composition is not commutative




rotate,
then translate




translate,
then rotate

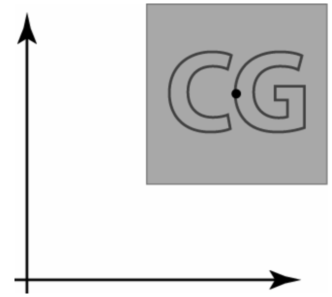
## Complex Transformation

- often represented as combination of simpler ones
  - intuitive geometric interpretation
- rotation around arbitrary axis
  - translate to axis center
  - rotate
  - translate back

$$R_{\mathbf{a},\theta} = T_{-\mathbf{a}} R_\theta T_\mathbf{a}$$

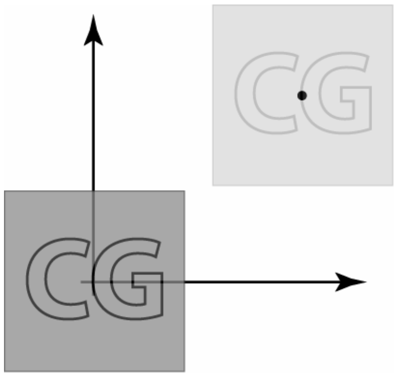


## Rotation around arbitrary axis



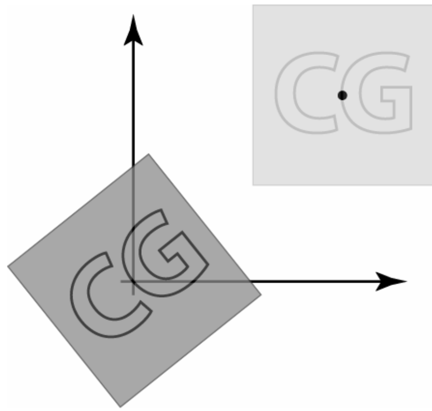



## Rotation around arbitrary axis

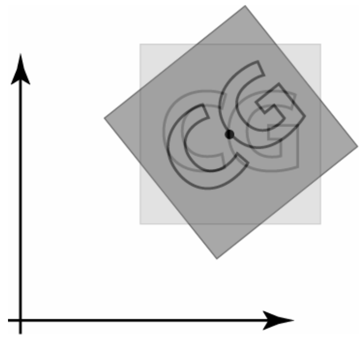





## Rotation around arbitrary axis



## Rotation around arbitrary axis





## Transforming points and vectors

- points and vectors are different entities
  - vectors: encode direction (difference of points)
  - points: encode position (origin plus a vector)
- points and vector transform differently
  - we have shown how points transforms previously
  - vectors simply ignore the translation

$$\mathbf{v} = \mathbf{p} - \mathbf{q}$$

$$X(\mathbf{p}) = M\mathbf{p} + \mathbf{t}$$

$$X(\mathbf{v}) = (M\mathbf{p} + \mathbf{t}) - (M\mathbf{q} + \mathbf{t}) = M(\mathbf{p} - \mathbf{q})$$



## Transforming points and vectors

- use homogeneous coordinates with w=0

$$\begin{bmatrix} M & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} = \begin{bmatrix} M\mathbf{v} \\ 0 \end{bmatrix}$$

- everything is consistent!
- but what is that “w” anyway?



## Homogeneous Coordinates

- points
  - will become useful later on

$$(p_x, p_y) \equiv \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} wp_x \\ wp_y \\ w \end{bmatrix}$$

- vectors

$$(v_x, v_y) \equiv \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}$$

## Coordinate Systems Review

- points are represented wrt a coordinate system
  - cartesian coordinates in the canonical coord. system

$$\mathbf{p} = (p_x, p_y) \Leftrightarrow \mathbf{p} = \mathbf{o} + p_x\mathbf{x} + p_y\mathbf{y} = \mathbf{o} + (\mathbf{x}\cdot\mathbf{p})\mathbf{x} + (\mathbf{y}\cdot\mathbf{p})\mathbf{y}$$

- canonical coordinate system

$$\mathbf{o} = (0,0)$$
$$\mathbf{x} = (1,0)$$
$$\mathbf{y} = (0,1)$$



## Coordinate System Review

- write a point in a new coordinate system

$$\mathbf{p}' = (p'_x, p'_y) \Leftrightarrow \mathbf{p} = \mathbf{o}' + (\mathbf{x}'\cdot\mathbf{p})\mathbf{x}' + (\mathbf{y}'\cdot\mathbf{p})\mathbf{y}' = \mathbf{o}' + p'_x\mathbf{x}' + p'_y\mathbf{y}'$$

- can be represented as an affine matrix multiply

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = M \begin{bmatrix} p'_x \\ p'_y \\ 1 \end{bmatrix} = \begin{bmatrix} x'_x & y'_x & o'_x \\ x'_y & y'_y & o'_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p'_x \\ p'_y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = M \begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{o}' \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix}$$



## Coordinate System Review

- an affine transform is a change of coord. system
  - another interpretation for combination of transforms
- what is the matrix I should use to change coord?
  - just invert previous definition
    - i.e. invert combination of translation and orthonormal

$$\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = M \begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix} = M^{-1} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} x'_x & x'_y & -o'_x \\ y'_x & y'_y & -o'_y \\ 0 & 0 & 1 \end{bmatrix}$$



## 3D Transformations

## 2D to 3D transformations

- adopt homogeneous formulation in 3d
  - points have 4 coordinates
  - use 4x4 matrices for transformations
- most concept generalize very easily
  - rotation much more complex




## Affine Transformations

translation

$$T_{\mathbf{t}} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

scale

$$S_{\mathbf{s}} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$




## Affine Transformations

rotation around z

$$R_\theta^z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$




## Affine Transformations

rotation around y

$$R_\theta^y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

rotation around x

$$R_\theta^x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Rotation around arbitrary axis

- in 2d, rotation are around a point $R_{\mathbf{a},\theta} = T_{-\mathbf{a}} R_\theta T_\mathbf{a}$
  - change coordinate frame (translation)
  - rotate around the origin
  - change coordinate frame back
  - simple geometric construction
- in 3d, they are around an axis $R_{\mathbf{a},\theta} = F_\mathbf{a}^{-1} R_\theta F_\mathbf{a}$
  - change coordinate frame (align z with axis)
  - rotate around z axis
  - change coordinate frame back
  - complex geometric construction



## Rotation around arbitrary axis

- use a change of coordinate system
  - define new coordinate system with z' parallel to axis and origin on the axis
- build transform matrix as seen previously

$$F^{-1} = \begin{bmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' & \mathbf{o}' \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



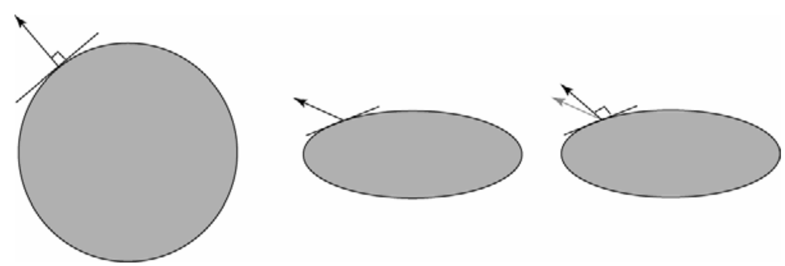
## Construct 3d frame from vectors

- given two non-parallel vectors **a** and **b**
  - i.e. a plane
  - set **x, y** parallel to **a, b**
  - **x = a / |a|**
    **z = x × b; z = z / |z|**
    **y = z × x**
- given the vector **a**
  - set **z** parallel to **a**, choose arbitrary **x, y**
  - continue as above



## Transforming normals

- points and vectors works
- tangents, i.e. differences of points, works too
- normals works differently
  - defined as orthogonal to the transformed surface
  - i.e. orthogonal to all tangents



## Transforming normals

- by definition $\mathbf{t} \cdot \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0$
- after transform $(M\mathbf{t})^T (X\mathbf{n}) = 0$
- for all **t**, we have $\mathbf{t}^T M^T X \mathbf{n} = 0$
- which gives $\mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = 0$

- normals are transformed by the inverse transpose



## Transformation Hierarchies

- often need to transform an object wrt another
  - e.g. the computer on the table
  - when the table moves, the computer moves
- naturally build a hierarchy of transformation
  - to transform the table, apply its transform
  - to transform the computer, apply the table and the computer transform

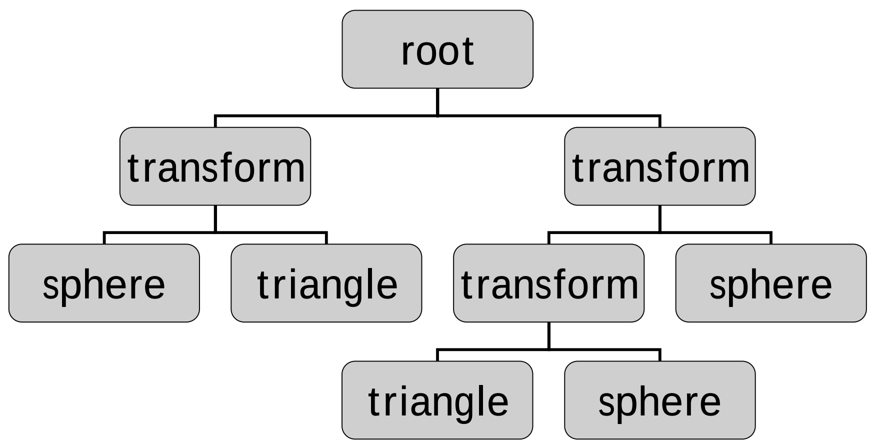


## Transformation Hierarchies

- represented as a tree data structure
  - transformation nodes
  - object nodes - leaves
- walk the tree when drawing
- very convenient representation for objects
  - all objects can be defined in their simplest form
  - e.g. every sphere can be represented by a transformation applied to the unit sphere



## Transformation Hierarchies



## Implementing Transformation Hierarchies

- transformation function for each node
  - get the parent function
  - apply the transform
  - pass the combined transform when calling children
- stack of transforms
  - push/pop when walking down/up
  - used by graphics libraries (OpenGL)
  - more flexible
  - generalized mechanism for all attributes



## Raytracing and Transformations

- transform the object
  - simple for triangles
    - since they transforms to triangles
  - but most objects require complex intersection tests
    - spheres do not transforms to spheres, but ellipsoids
- transform the ray
  - much more elegant
  - works on any surface
  - allow for much simpler intersection tests
    - only worry about unit sphere, all others are transformed



## Raytracing and Transformations

- transforming rays
  - transform origin/direction as point/vector
  - note that direction is not normalized now
    - i.e. ray parameter is not the distance
- intersect a transformed object
  - transform the ray by matrix inverse
  - intersect surface
  - transform hit point and normal by matrix