EECS 487: Interactive Computer Graphics

Lecture 10:

- Homogeneous Coordinates
- Affine Transforms
- Transforming Normals

Points vs. Vectors

A point is a fixed location in space

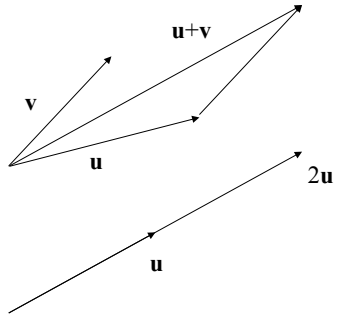
A vector can be considered a displacement or motion between points

Addition and scalar multiplication are well defined for vectors

- the addition of 2 vectors is the concatenation of 2 displacements

$$\mathbf{u}+\mathbf{v}=\begin{bmatrix}u_0\\u_1\\u_2\end{bmatrix}+\begin{bmatrix}v_0\\v_1\\v_2\end{bmatrix}=\begin{bmatrix}u_0+v_0\\u_1+v_1\\u_2+v_2\end{bmatrix}$$

- multiplying a vector by some factor scales the displacement



Points vs. Vectors

A point is a fixed location in space
- can't add two locations (Detroit + Chicago?)
- can't multiply (the lat/lon of) a location by a scale (9∗Detroit?)

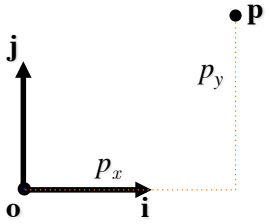
But subtracting two points gives a vector that describes the motion from one location to another: $\bar{u}=\mathbf{q}-\mathbf{p}$
(if we treat points as vectors, adding two points does give their midpoint, but that's by virtue of vector addition)

Or the vector can describe the location that is some displacement away from a given location: $\mathbf{q}=\mathbf{p}+\bar{u}$

On the other hand, since a vector consists only of magnitude and direction, it doesn't make sense to talk about moving/translating a vector

Coordinate System

Given a point $\mathbf{p}=\begin{bmatrix}p_x\\p_y\end{bmatrix}$ we usually understand it to mean a point in the Cartesian coordinate frame with origin at $\mathbf{o}$: $\mathbf{p}=\mathbf{o}+p_x\mathbf{i}+p_y\mathbf{j}$ where $\mathbf{i}$ and $\mathbf{j}$ are the standard basis



$$\mathbf{i}=\begin{bmatrix}1\\0\end{bmatrix}\text{ and }\mathbf{j}=\begin{bmatrix}0\\1\end{bmatrix}$$

In matrix form: $\mathbf{p}=\mathbf{o}+\begin{bmatrix}\mathbf{i} & \mathbf{j}\end{bmatrix}\begin{bmatrix}p_x\\p_y\end{bmatrix}=\mathbf{o}+\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}p_x\\p_y\end{bmatrix}$

Basis

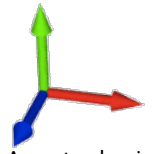
What is a basis?
A linearly independent set of vectors whose linear combinations include all vectors in the space (there are infinitely many bases for any given space)

What does it mean for a set of vectors to be linearly independent?
No vector in the set is a linear combination of the others

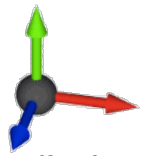
Origin and Change of Frame

Coordinates themselves (p_x, p_y) are not geometric entities
- they are just scalar coefficients
- a basis is required for a set of coordinates to represent a vector
- a frame is required for a set of coordinates to represent a point
- coordinate frame = origin + basis

A linear transform changes the basis of a coordinate system, but not the origin:
$\mathbf{p}' = \mathbf{Mp} = p_x\mathbf{u} + p_y\mathbf{v}$


A vector basis

An affine transform can change both the basis and the origin of the coordinate frame:
$\mathbf{p}' = \mathbf{Mp} + \mathbf{t} = p_x\mathbf{u} + p_y\mathbf{v} + \mathbf{t}$


An affine frame

Lozano-Perez01

Homogeneous Coordinates

Vectors live in linear space ($\mathbb{R}^3$) and can be operated by linear transforms (rotate and scale)

Points live in Euclidean space ($\mathbb{E}^3$) and can be operated by Euclidean transforms (translate and rotate)

The coordinate frame a point lives in requires the specification of a basis and an absolute origin

$$\mathbf{p} = p_x\mathbf{u} + p_y\mathbf{v} + \mathbf{o} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{o} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

A vector space is completely defined by a relative set of coordinate basis, irrespective of origin

$$\mathbf{w} = p_x\mathbf{u} + p_y\mathbf{v} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{o} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 0 \end{bmatrix}$$

homogeneous coordinates

Lozano-Perez01

Homogeneous Coordinates

Attribute	Vector	Point
Representation	$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
Represent	magnitude and direction	location
Origin	relative	absolute
Transformations	linear scale and rotate	Euclidean translate and rotate

More generally, homogeneous coordinates increase dimensionality by adding one extra coordinate w
- recall: coordinates are not geometric entities, they are just scales of basis vectors

Homogeneous Coordinates

For now we assume w is always 1 or 0, representing, respectively, a point, preserving origin, or a vector

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Often, w is not stored, which requires different operation routines for points and vectors

To extract the Cartesian coordinates in Euclidean space out of homogeneous coordinates, divide by w

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} / w \propto \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

In perspective transformations w encodes perspective scaling

- more when we discuss perspective projection

Translation with Homogeneous Coordinates

Translation:

$$x' = ax + by + t_x$$
$$y' = dx + ey + t_y$$

in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} ax + by + t_x \\ dx + ey + t_y \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{M}\mathbf{p} + \mathbf{t}$$

with homogeneous coordinates:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ d & e & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + t_x \\ dx + ey + t_y \\ 1 \end{bmatrix}$$

$\mathbf{p}' = \mathbf{T}(\mathbf{t})\mathbf{p}$ ⇒ translation can now be combined with the other affine transforms!

Durando8

Affine Matrices

In matrix form, an affine transform always looks like:

$$\mathbf{M}_{aff} = \begin{bmatrix} m_{11} & m_{12} & t_x \\ m_{21} & m_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{lin} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

where $\mathbf{M}_{lin}$ is a linear matrix

Translation

$$\mathbf{T}(\mathbf{t}) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation

$$\mathbf{R}(\varphi, \mathbf{o}) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling

$$\mathbf{S}(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If $\mathbf{M}_{lin}$ is a rotation matrix, then $\mathbf{M}_{aff}$ consists only of rotation and translation and is therefore a rigid body transform

Curlesso8

Affine Transforms of Homogeneous Coordinates

Affine transforms, i.e., multiplication by an affine matrix, of homogeneous coordinates leaves w unchanged

$$\begin{bmatrix} m_{11} & m_{12} & t_x \\ m_{21} & m_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

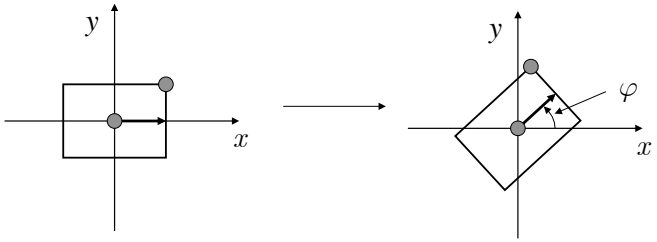
A general transform would change w

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Rotation Around Arbitrary Point

So far we have only considered rotation about the origin

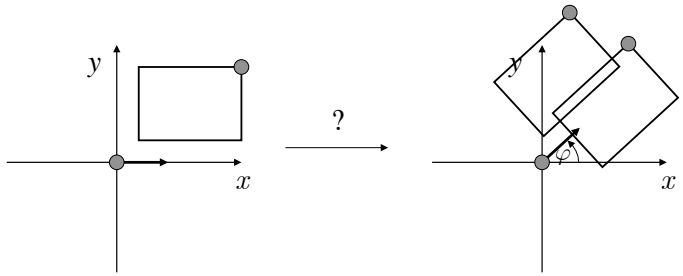
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Rotation Around Arbitrary Point

What happens if we apply the same transform to an object not centered at the origin?

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



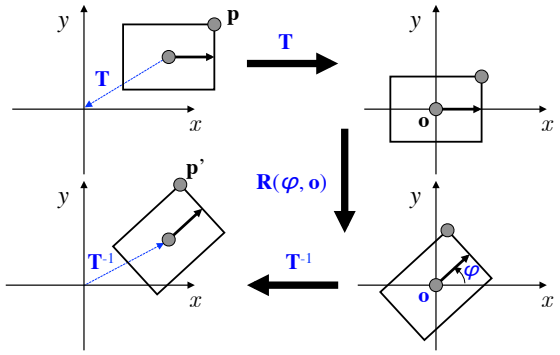
Rotation Around Arbitrary Point

What if we really want to rotate about the object center, not the origin?

- translate to origin
- apply rotation matrix
- translate back to original position

$$\mathbf{p}' = \mathbf{T}^{-1}\mathbf{R}(\varphi, \mathbf{o})\mathbf{T}\mathbf{p}$$

composition possible only by use of Homogeneous Coordinates!

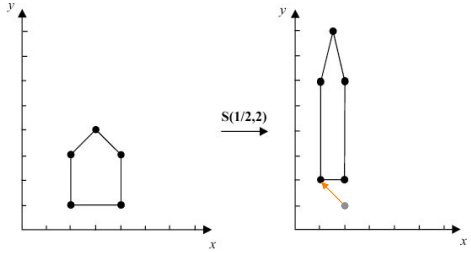


Scaling About Arbitrary Point

Similarly for scaling, we have only considered scaling object about the origin

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

Applied to an object not about the origin shows a translation side-effect that is proportional to the scaling factor

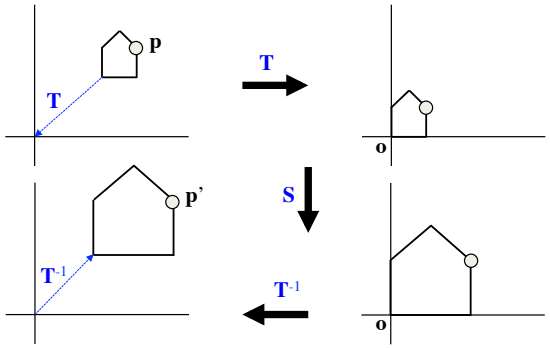


Scaling About Arbitrary Point

How do we scale an object about an arbitrary point?

- translate to origin
- apply scaling matrix
- translate back to original position

$\mathbf{p'} = \mathbf{T}^{-1}\mathbf{S}\mathbf{T}\mathbf{p}$

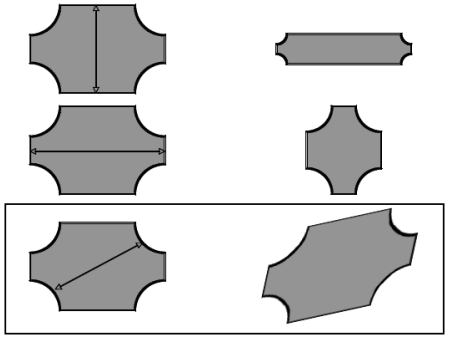


Foley et al. 94

Scaling About Arbitrary Axis

Our scaling matrix assumes scaling about coordinate axes only

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



How do we scale about arbitrary axis?

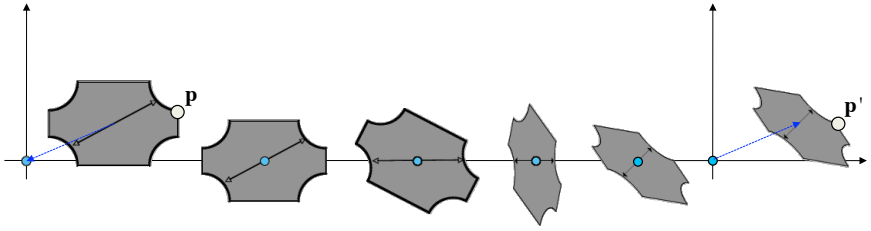
O'Brieno8

Scaling About Arbitrary Axis

Scaling about arbitrary axis:

- translate axis center to origin
- rotate axis to align with one of the coordinate axes
- scale as desired
- rotate back to original orientation
- translate back to original position

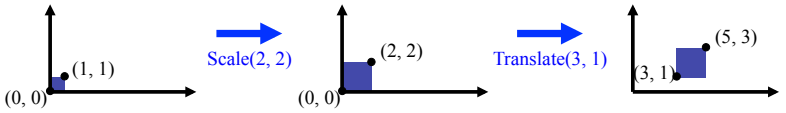
$\mathbf{p'} = \mathbf{T}^{-1}\mathbf{R}^{-1}\mathbf{S}\mathbf{R}\mathbf{T}\mathbf{p}$



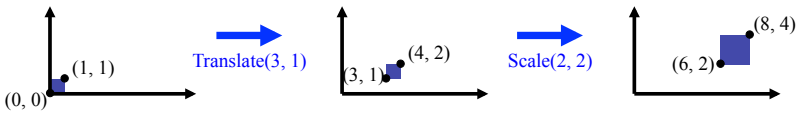
O'Brieno8

Transforms Composition is Non-commutative

Scale then translate: $\mathbf{p'} = \mathbf{T}(\mathbf{S}\mathbf{p}) = \mathbf{T}\mathbf{S}\mathbf{p}$



Translate then scale: $\mathbf{p'} = \mathbf{S}(\mathbf{T}\mathbf{p}) = \mathbf{S}\mathbf{T}\mathbf{p}$



Durando8

Transforms Composition is Non-commutative

Scale then translate: $\mathbf{p}' = \mathbf{T}(\mathbf{S}\mathbf{p}) = \mathbf{T}\mathbf{S}\mathbf{p}$

$$\mathbf{TS} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Translate then scale: $\mathbf{p}' = \mathbf{S}(\mathbf{T}\mathbf{p}) = \mathbf{S}\mathbf{T}\mathbf{p}$

$$\mathbf{ST} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Commutative Exceptions

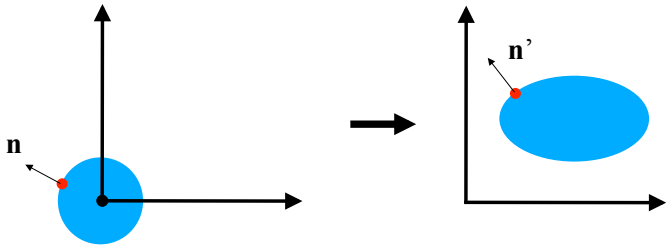
$$\mathbf{T}(\mathbf{t}_1)\mathbf{T}(\mathbf{t}_2) = \mathbf{T}(\mathbf{t}_2)\mathbf{T}(\mathbf{t}_1) = \mathbf{T}(\mathbf{t}_1 + \mathbf{t}_2)$$

$$\mathbf{S}(s_{x_1}, s_{y_1})\mathbf{S}(s_{x_2}, s_{y_2}) = \mathbf{S}(s_{x_2}, s_{y_2})\mathbf{S}(s_{x_1}, s_{y_1}) = \mathbf{S}(s_{x_1}s_{x_2}, s_{y_1}s_{y_2})$$

$$\mathbf{R}(\varphi_1, \mathbf{o})\mathbf{R}(\varphi_2, \mathbf{o}) = \mathbf{R}(\varphi_2, \mathbf{o})\mathbf{R}(\varphi_1, \mathbf{o}) = \mathbf{R}(\varphi_1 + \varphi_2, \mathbf{o})$$ 2D only!

$$\mathbf{S}(s_x, s_y)\mathbf{R}(\varphi, \mathbf{o}) = \mathbf{R}(\varphi, \mathbf{o})\mathbf{S}(s_x, s_y)$$ for isotropic **S** only!

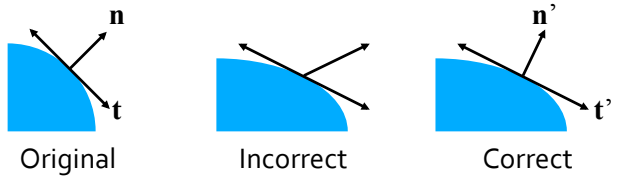
How Do we Transform Normals?



transformed normal no longer perpendicular to surface!

So How Do We Do It Right?

Observe: normals are defined by surface tangent



Pick any vector **t** in the tangent plane, how is it transformed by matrix **M**? $\mathbf{t}' = \mathbf{M}\,\mathbf{t}$

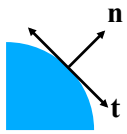
- a vector tangent (**t**) transformed (**t'**) would still be the tangent of the surface

Cause: affine transformation maps parallel lines to parallel lines, but not so for perpendicular lines

Transforming Normal

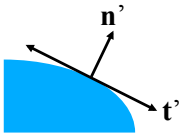
$\mathbf{t}$ is perpendicular to normal $\mathbf{n}$:

$$\mathbf{n} \cdot \mathbf{t} = 0$$
$$\mathbf{n}^T \mathbf{t} = 0$$
$$\mathbf{n}^T \mathbf{I} \mathbf{t} = \mathbf{n}^T \mathbf{M}^{-1} \mathbf{M} \mathbf{t} = 0$$
$$(\mathbf{n}^T \mathbf{M}^{-1})(\mathbf{M}\mathbf{t}) = 0$$
$$(\mathbf{n}^T \mathbf{M}^{-1})\mathbf{t}' = 0$$



$\mathbf{t}$' is perpendicular to normal $\mathbf{n}$':

$$\mathbf{n}' \cdot \mathbf{t}' = 0$$
$$\mathbf{n}'^T \mathbf{t}' = 0$$
$$\mathbf{n}'^T = \mathbf{n}^T \mathbf{M}^{-1}$$
$$\mathbf{n}' = (\mathbf{M}^{-1})^T \mathbf{n}$$



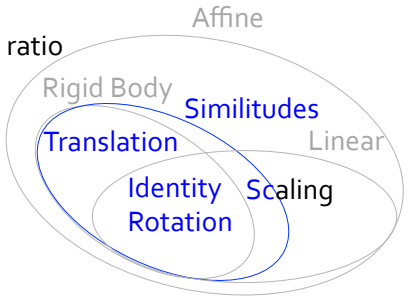
$(\mathbf{M}^{-1})^T$ exists as long as $|\mathbf{M}| \neq 0$, $\mathbf{M}$ doesn't include projection

Why Does the Normal Sometimes Transform Correctly?

Properties of similitudes:

- maintains "similar" shape (similar triangles, circles map to circles, etc.)
- angles are preserved
- distances are changed by a fixed ratio
- $(\mathbf{M}^{-1})^T = \lambda \mathbf{M}$

  $\Rightarrow \mathbf{n}' = (\mathbf{M}^{-1})^T \mathbf{n} = \lambda \mathbf{M} \mathbf{n}$



a.k.a. Orthogonal Transforms

Inverting Transformations

$\mathbf{M}\mathbf{M}^{-1} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$
In general, $\mathbf{M}^{-1}$ undoes the effect of $\mathbf{M}$
In general, compute inverse using the adjoint method, Cramer's rule, LU decomposition, Gaussian elimination, or invert SVD matrices

Special (simple) cases:

- translation: a translation in the opposite direction: $\mathbf{T}^{-1}(\mathbf{t}) = \mathbf{T}(-\mathbf{t})$
- (axis-aligned) scaling: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$
- rotation: transpose ($\mathbf{R}$ is orthogonal)

Inverting composite transforms

- swap order and invert each transform, e.g., $(\mathbf{T}(\mathbf{t})\mathbf{R}(\varphi,\mathbf{o}))^{-1} = \mathbf{R}(-\varphi,\mathbf{o})\,\mathbf{T}(-\mathbf{t})$

$$\mathbf{M} = \mathbf{M}_3\mathbf{M}_2\mathbf{M}_1$$
$$\mathbf{I} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{M}_1^{-1}(\mathbf{M}_2^{-1}(\mathbf{M}_3^{-1}\mathbf{M}_3)\mathbf{M}_2)\mathbf{M}_1$$
$$\mathbf{M}^{-1} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\mathbf{M}_3^{-1}$$

Orthogonal Matrix

$\mathbf{M}$ is an orthogonal matrix iff:

- $\mathbf{M}^{-1} = \mathbf{M}^T$
- $|\mathbf{M}| = 1$
- its columns (and rows) form an orthonormal basis:
  - columns are orthogonal: $\text{col}_1 \bullet \text{col}_2 = 0$
  - and of unit length $||\text{col}_1|| = ||\text{col}_2|| = 1$
- $\mathbf{M}$ must be a square matrix
- for $\mathbf{M}$ orthonormal, if $\mathbf{N}$ is orthogonal, so is $\mathbf{NM}$

For example, the rotation matrix is an orthogonal matrix

$$\mathbf{R}(\varphi,\mathbf{o}) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

Orthogonal Matrix

Let $\mathbf{s}$ and $\mathbf{t}$ be the two rows of $\mathbf{R}(\varphi,\mathbf{o})$:

$\mathbf{s} = [\cos\varphi \quad -\sin\varphi]^T$ and $\mathbf{t} = [\sin\varphi \quad \cos\varphi]^T$

$$\mathbf{R}(\varphi,\mathbf{o})\mathbf{s} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}\begin{bmatrix} \cos\varphi \\ -\sin\varphi \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mathbf{R}(\varphi,\mathbf{o})\mathbf{t} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}\begin{bmatrix} \sin\varphi \\ \cos\varphi \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{R}\mathbf{R}^T = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} = \mathbf{I}$$

$$\therefore \mathbf{R}\mathbf{R}^T = \mathbf{I} = \mathbf{R}\mathbf{R}^{-1} \Rightarrow \mathbf{R}^{-1} = \mathbf{R}^T$$

$\Rightarrow \mathbf{R}$ is an orthogonal matrix!