


CS380: Introduction to Computer Graphics
Affine Transform
Chapter 3

Min H. Kim
KAIST School of Computing


Min H. Kim (KAIST) *Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012*



Announcement

- A new homework will be released every Friday night (mostly). Please check the course website frequently.


Min H. Kim (KAIST) *Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012* 2



Linear Transformation

SUMMARY

Min H. Kim (KAIST) *Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012* 3



Geometric data types

- Point: \tilde{p}
 - represents place
- Vector: \vec{v}
 - represents motion/offset between points
- Coordinate vector: \mathbf{c}
- Coordinate system: \vec{s}^t
 - “basis” for vectors
 - “frame” for points

Min H. Kim (KAIST) *Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012* 4

Vector space

- A vector space V is some set of elements \vec{v}
- Coordinate vector:
- if $\vec{b}_1 \dots \vec{b}_n$ are linearly independent, all vectors \vec{v} of V can be expressed with coordinates c_i of a basis of V (a set of \vec{b}_i).
- n is the dimension of the basis/space

$$\vec{v} = \vec{b}^t \mathbf{c}. \quad \vec{v} = \sum_i c_i \vec{b}_i = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 5

Linear transformation

- 3-by-3 matrix

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}.$$

- Linear transform of a vector
 $\vec{v} = \vec{b}^t \mathbf{c} \Rightarrow \vec{b}^t M \mathbf{c}$
- Linear transform of a basis
 $\vec{v} = \vec{b}^t \mathbf{c} = \vec{a}^t M^{-1} \mathbf{c}.$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 6

Product of vectors; rotation

- Dot product: $\vec{v} \cdot \vec{w} := \|\vec{v}\| \|\vec{w}\| \cos \theta.$
- Cross product: $\vec{v} \times \vec{w} := \|\vec{v}\| \|\vec{w}\| \sin \theta \vec{n},$
- 3D rotation matrices:

around z axis,	x axis,	y axis
$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$	$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 7

Chapter 3

AFFINE TRANSFORMATION

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 8

Points vs. vectors

- Vector \vec{v} := motion between points in space
 - lives in a space we call R^3
 - has the structure of a linear/vector space.
 - addition and scalar multiplication have meaning
 - zero vector is no motion
 - cannot really translate motion

Points vs. vectors

- Point \tilde{p} := a position in space
 - lives in a space we might call A^3
 - has the structure of a so-called affine space.
 - addition and scalar multiplication don't make sense
 - zero doesn't make sense
 - subtraction does make sense, gives us a vector

Points and frames

- Subtraction of points:

$$\tilde{p} - \tilde{q} = \vec{v}$$
- Moving a point with a vector:

$$\tilde{q} + \vec{v} = \tilde{p}$$
- Basis is three vectors

$$\vec{v} = \sum_i c_i \vec{b}_i$$
- How could we present translations?
 - Affine transform (4-by-4 matrix)
 - usage: transformation of objects and camera projection (3D \rightarrow 2D)

Points and frames

- for affine space we will use a frame
 - start with a chosen origin point \tilde{o}
 - add to it a linear combination combination of vectors using coordinates c_i to get to any desired point \tilde{p}

Affine frame

- Movement of a point (original $\tilde{o} \rightarrow$ a point \tilde{p})

$$\tilde{p} = \tilde{o} + \vec{v}.$$

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}.$$
- Affine frame** (made of three vectors and a point): $\tilde{p} = \vec{f}' \mathbf{c}.$

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} = \vec{f}'$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 13

Defining an affine matrix

- point is specified with a 4-coordinates vector
 - four numbers
 - last one is always 1 $\begin{bmatrix} c_1 & c_2 & c_3 & 1 \end{bmatrix}'$
 - ... or 0. (and we get a vector)
- let's define an affine matrix as 4-by-4 matrix

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- we are transforming a point to another with an affine frame:

$$\tilde{p} = \vec{f}' \mathbf{c}.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 14

Transforming a point

- affine transform

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 15

Transforming a point

- for short

$$\tilde{p} = \vec{f}' \mathbf{c} \Rightarrow \vec{f}' \mathbf{A} \mathbf{c}.$$
 where $\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} = \vec{f}'$
- transforming **coordinate vectors** (4 with a one as the fourth entry)

$$\begin{bmatrix} c'_1 \\ c'_2 \\ c'_3 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 16

transforming a point

- Alternatively, transforming the basis vectors

$$\begin{bmatrix} \bar{b}'_1 & \bar{b}'_2 & \bar{b}'_3 & \tilde{o}' \end{bmatrix} = \begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This transformation is to apply the affine transform to a frame as

$$\begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \Rightarrow \begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 17

Linear transformation

- 3-by-3 transform matrix → 4-by-4 affine transform

$$\begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} a & b & c & 0 \\ e & f & g & 0 \\ i & j & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

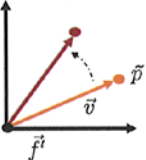
Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 18

Linear transformation

- affine transformation: $L = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$

where, L is a 4-by-4 matrix; I is a 3-by-3 matrix.

- A linear transform is applied to a point. This is accomplished by applying the linear transform to its offset vector.



Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 19

Translation transform

- translation transformation to points

$$\begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 20

Translation transform

- translation transformation to points

$$c_1 \Rightarrow c_1 + t_x$$

$$c_2 \Rightarrow c_2 + t_y$$

$$c_3 \Rightarrow c_3 + t_z$$

- translation matrix

$$T = \begin{bmatrix} i & t \\ 0 & 1 \end{bmatrix}$$

- where, T is a 4-by-4 matrix; i is a 3-by-3 identity matrix, t is 3-by-1 matrix for translation.

Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

21

Affine transform matrix

- An affine matrix can be factored into a linear part and a translational part:

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c & 0 \\ e & f & g & 0 \\ i & j & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} l & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} i & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} l & 0 \\ 0 & 1 \end{bmatrix} \quad A = TL$$

Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

22

Affine transform matrix

- NB as **matrix multiplication is not commutative**, the order of the multiplication TL matters!!!

$$TL \neq LT$$

- Since these matrices have the same size (4-by-4), it is difficult to debug when you messed up the order. Pay extra attention on it while you are coding...

Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

23

Rigid body transformation

- When the linear transform is a rotation, we call this as **rigid body transformation (rotation + translation only)**.

$$A = TR$$

- A rigid body transformation preserves dot product between vectors, handedness of a basis, and distance between points.
- Its geometric topology is maintained while transforming it.

Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

24

Affine transform acting on vector

- If fourth coordinate of \mathbf{c} is zero, this just transforms a vector to a vector.
 - note that the fourth column is irrelevant
 - a vector cannot be translated

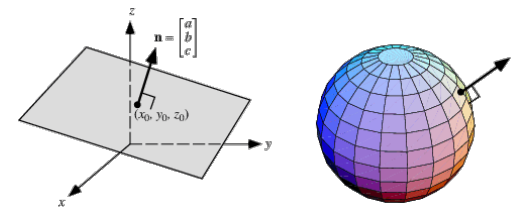
Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

25

Normals

- Normal: a vector that is orthogonal to the **tangent plane** of the surfaces at that point.
 - the tangent plane is a plane of vectors that are defined by subtracting (infinitesimally) nearby surface points: $\vec{n} \cdot (\tilde{p}_1 - \tilde{p}_2) = 0$



Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

26

Normals

- We use normals for shading
- how do they transform
- suppose i rotate forward
 - normal gets rotated forward
- suppose squash in the y direction

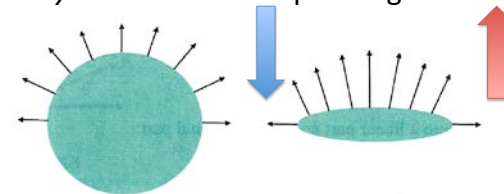
Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

27

Changing a shape

- Squashing a sphere makes its normals stretch along the y axis instead of squashing.



- normal gets higher in the y direction
- what is the rule?

$$\begin{bmatrix} nx \\ ny \\ nz \end{bmatrix} \neq \begin{bmatrix} nx' \\ ny' \\ nz' \end{bmatrix}$$

Min H. Kim (KAIST)

Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012

28

Transforming normals

- Since the normal \vec{n} and very close points \tilde{p}_1 and \tilde{p}_2 are on a surface:

$$\vec{n} \cdot (\tilde{p}_1 - \tilde{p}_2) = 0$$

$$\begin{bmatrix} nx & ny & nz & * \end{bmatrix} \begin{pmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} - \begin{bmatrix} x0 \\ y0 \\ z0 \\ 1 \end{bmatrix} \end{pmatrix} = 0.$$
- After applying an affine transform A ,

the normal of the transformed geometry

$$\begin{pmatrix} \begin{bmatrix} nx & ny & nz & * \end{bmatrix} A^{-1} \end{pmatrix} \begin{pmatrix} A \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix} - A \begin{bmatrix} x0 \\ y0 \\ z0 \\ 1 \end{bmatrix} \end{pmatrix} = 0.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 29

Transforming normals

- Transformed normals:

$$\begin{bmatrix} nx' & ny' & nz' \end{bmatrix} = \begin{bmatrix} nx & ny & nz \end{bmatrix} L^{-1}.$$
- Transposing this expression:

$$\begin{bmatrix} nx' \\ ny' \\ nz' \end{bmatrix} = L^{-t} \begin{bmatrix} nx \\ ny \\ nz \end{bmatrix}.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 30

Transforming normals

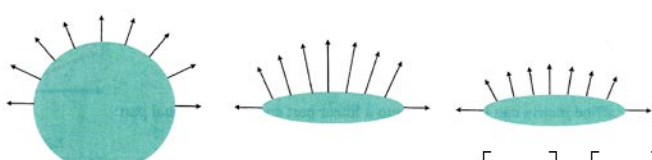
- Remember L is a rotation matrix (orthonormal), thus its inverse transpose is the same as the original: $L^{-t} = L$.

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$LL^t = I \quad (L^t = L^{-1}), \det L = 1$$
- inverse transpose
 - so inverse transpose/transpose inverse is the rule
 - for rotation, transpose = inverse
 - for scale, transpose = nothing
 - in the code next week, we will send A and L^t to the vertex shader.

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 31

Transforming normals

- Renormalize to correct unit normals of squashed shape:
 

$$L^{-t} \begin{bmatrix} nx' \\ ny' \\ nz' \end{bmatrix} \Rightarrow \begin{bmatrix} nx \\ ny \\ nz \end{bmatrix}.$$

Min H. Kim (KAIST) Foundations of 3D Computer Graphics, S. Gortler, MIT Press, 2012 32

Quizzes



- We will have quizzes about what we have learned so far next time.