

# Computer Graphics

Minho Kim

September 23, 2009

# Table of contents

## Chapter 4: Geometric Objects and Transformations

Sec 4.1: Scalars, Points, and Vectors

Sec 4.2: Tree-Dimensional Primitives

Sec 4.3: Coordinate Systems and Frames

Sec 4.4: Frames in OpenGL

Sec 4.5: Modeling a Colored Cube

Sec 4.6: Affine Transformations

Sec 4.7: Translation, Rotation, and Scaling

Sec 4.8: Transformations in Homogeneous Coordinates

Sec 4.9: Concatenation of Transformations

Sec 4.10: OpenGL Transformation Matrices

Sec 4.11: Interfaces to Tree-Dimensional Applications

Sec 4.12: Quaternions

## Chapter 4: Geometric Objects and Transformations

## Sec 4.1: Scalars, Points, and Vectors

## 4.1.1: Geometric Objects

- ▶ Scales, vectors, and points
- ▶ Which of the followings make sense?  
 $v + 2u - 3w$ ,  $P + 3v$ ,  $2P - Q + 3v$ ,  $P + 3Q - v$

## 4.1.2: Coordinate-Free Geometry

- ▶ We don't need coordinate system to define a point!
- ▶ To represent a point using a coordinate system/frame is just convenient.

## 4.1.6: Lines

- ▶ *Parametric form of a line:*

All the points on a line can be expressed as  $P(\alpha) = P_0 + \alpha d$ .  
( $P(\alpha)$ ,  $P_0$  are points and  $d$  is a vector)

## 4.1.7: Affine Sums

- ▶ In an affine space,
  - ▶ the addition of two points and
  - ▶ the multiplication of a point by a scalarare not allowed.
- ▶ But a special form, called *barycentric combination*, *affine addition*, or *affine combination* is allowed;  
 $P = \alpha R + (1 - \alpha)Q$  ( $P$ ,  $R$ , and  $Q$  are points and  $\alpha$  is a scalar). Why?
- ▶ More generally, for the points  $\{P_j\}_{j=1}^n$  and the scalars  $\{\alpha_j\}_{j=1}^n$  where  $\sum_{j=1}^n \alpha_j = 1$ ,  $\sum_{j=1}^n \alpha_j P_j$  is a barycentric combination of the points  $\{P_j\}_{j=1}^n$ .

## 4.1.8: Convexity

- ▶ Special case of barycentric combination.
- ▶ For the points  $\{P_j\}_{j=1}^n$  and the scalars  $\{\alpha_j\}_{j=1}^n$  where  $\sum_{j=1}^n \alpha_j = 1$  and  $\alpha_j \geq 0, \forall \alpha_j$ ,  $\sum_{j=1}^n \alpha_j P_j$  is a *convex combination* of the points  $\{P_j\}_{j=1}^n$ .

## 4.1.9: Dot and Cross Products

- ▶ *Cross product* of two vectors  $(u, v)$  is another vector  $n = u \times v$  with
  - ▶ its length is determined as  $|n| = |u||v| \sin \theta$  ( $\theta$  is the *smaller* angle between  $u$  and  $v$ ) and
  - ▶ its direction is determined *right-handed rule*.
- ▶ Given a pair of *non-parallel* vectors  $u$  and  $v$ ,  $u$ ,  $v$  and  $u \times v$  form a right-handed coordinate system.

## 4.1.10: Planes

- ▶ All the points on a plane can be defined by
  - ▶ three *non-degenerate* points ( $P$ ,  $Q$ ,  $R$ ) on the plane as  $T(\alpha, \beta) = \beta[\alpha P + (1 - \alpha)Q] + (1 - \beta)R$ ,
  - ▶ one point ( $P_0$ ) on the plane and two (non-parallel) vectors ( $u$ ,  $v$ ) parallel to the plane as  $T(\alpha, \beta) = P_0 + \alpha u + \beta v$  or
  - ▶ one point ( $P_0$ ) on the plane and one vector orthogonal to the plane ( $n$ ) as  $n \cdot (T - P_0) = 0$ .

## Sec 4.2: Tree-Dimensional Primitives

# Three-Dimensional Primitives

- ▶ Simplification required for interactive rendering:
  - ▶ hollow objects described by surfaces
  - ▶ objects specified by vertices
  - ▶ objects approximated by flat, convex polygons
- ▶ CSG (Constructive Solid Geometry)

## Sec 4.3: Coordinate Systems and Frames

# Representation of Vectors

- ▶ Using the basis of a coordinate system
- ▶ Representation using column matrices, e.g.,  
 $w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$  is represented by

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

therefore,

$$w = \boldsymbol{\alpha}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} .$$

# Representation of Points

- ▶ Using the basis and the reference point of a frame
- ▶ Given a frame with a basis  $\{v_j\}_{j=1}^3$  and a reference point  $P_0$ , any point can be uniquely written as

$$P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3$$

## 4.3.2: Change of Coordinate Systems

- ▶ How can we convert the representation ( $\alpha$ ) of a vector in one coordinate system (with basis  $\{v_j\}_{j=1}^3$ ) to the representation ( $\beta$ ) in another coordinate system (with basis  $\{u_j\}_{j=1}^3$ )?
- ▶ In other words, for the vector

$$w = \alpha^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \beta^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix},$$

find  $\beta$  (or  $\mathbf{M}$  such that  $\beta = (\mathbf{M}^T)^{-1}\alpha$ ).

- ▶ How to change the origin, too?  $\rightarrow$  homogeneous coordinates

## 4.3.4: Homogeneous Coordinates

- ▶ In a frame defined by  $(v_1, v_2, v_3, P_0)$ ,
  - ▶ a vector is expressed as  $w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$  and
  - ▶ a point is expressed as  $P = P_0 + \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3$ .
- ▶ Although the multiplication of a point by a scalar is not defined, if we define the multiplication by 0 and 1 as
  - ▶  $0 \cdot P = \mathbf{0}$  and
  - ▶  $1 \cdot P = P$ ,

we can express

- ▶ a vector as  $w = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$  and

- ▶ a point as  $p = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$ .

## Sec 4.4: Frames in OpenGL

# Frames in OpenGL

- ▶ Six representations
  1. Object or model coordinate
  2. World coordinate
  3. Eye (or camera) coordinate
  4. Clip coordinate
  5. Normalized device coordinate
  6. Window (or screen) coordinate
- ▶ The transformations “model coords  $\rightarrow$  world coords  $\rightarrow$  eye coords” are concatenated together into the *model-view transformation*.
- ▶ The transformation from eye coords to clip coords is done by *projection transformation*.
- ▶ By *perspective division*, the clip coords is converted to the normalized device coords.

## Sec 4.5: Modeling a Colored Cube

## Sec 4.6: Affine Transformations

# Affine Transformations

- ▶ General transformation:
  - ▶  $\mathbf{q} = f(\mathbf{p})$ , ( $p, q$  points)
  - ▶  $\mathbf{v} = f(\mathbf{u})$  ( $u, v$  vectors)
- ▶ Affine transformation:
  - ▶ Linear:  $f(\alpha p + \beta q) = \alpha f(p) + \beta f(q)$  ( $\alpha, \beta$  scalars and  $p, q$  vertices)
  - ▶ The transformation of linear combination of  $p$  and  $q$  can be obtained by the (same) linear combination of their transformations!
  - ▶ Can be represented by a  $4 \times 4$  matrix multiplication:  $\mathbf{v} = \mathbf{A}\mathbf{u}$  where

$$\mathbf{A} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- ▶ 12 DOF (degrees of freedom) for points and 9 DOF for vectors. Why?
- ▶ Composed of translation, rotation, and scaling.

## Sec 4.7: Translation, Rotation, and Scaling

## Sec 4.8: Transformations in Homogeneous Coordinates

# Affine Transformations in Homogeneous Coordinates

- ▶ Translation by  $(\alpha$

# Affine Transformations in Homogeneous Coordinates (cont'd)

- ▶ Rotation by  $\theta$  about the  $z$ -axis

$$\mathbf{R}_z = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- ▶ Rotation by  $\theta$  about the  $x$ -axis

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- ▶ Rotation by  $\theta$  about the  $y$ -axis

$$\mathbf{R}_y = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Sec 4.9: Concatenation of Transformations

# Concatenation of Transformations

- ▶  $\mathbf{q}$  obtained by applying  $\mathbf{A}$ ,  $\mathbf{B}$ , and then  $\mathbf{C}$  in the order, to  $\mathbf{p}$ .

$$\mathbf{q} = \mathbf{CBAp}$$

- ▶ **Associative but not commutative!!!**
  - ▶  $\mathbf{C(BA)} = (\mathbf{CB})\mathbf{A}$
  - ▶  $\mathbf{AB} \neq \mathbf{BA}$ .
- ▶ Rotation about an arbitrary axis
  - ▶ Can be decomposed into three rotations about the x-, y-, and z-axes.
  - ▶ *Quaternions*
- ▶ Instance transformations
  - ▶ Scaling  $\rightarrow$  rotation  $\rightarrow$  translation
  - ▶ Used for most objects modeled independently

## Sec 4.10: OpenGL Transformation Matrices

# OpenGL Transformations

- ▶ Three affine transformations
  - ▶ `glTranslate*()`: Translation
  - ▶ `glScale*()`: Scaling along each direction
  - ▶ `glRotate*()`: Rotation about arbitrary axis
- ▶ General transformations by `glMultMatrix*()`
- ▶ **Matrices in column-major format!**
- ▶ Hierarchical transformations using model-view matrix stack with push (`glPushMatrix()`) and pop (`glPopMatrix()`) operations.

## Sec 4.11: Interfaces to Tree-Dimensional Applications

## Sec 4.12: Quaternions