15-462 Computer Graphics I
Lecture 4

Transformations

Announcement

• Guest lecture Tuesday, January 29
• From Design to Production: How a Graphics Chip is Built, Scott Whitman, nVidia

January 24, 2002
Frank Pfenning
Carnegie Mellon University
http://www.cs.cmu.edu/~fp/courses/graphics/

Vector Spaces
Affine and Euclidean Spaces
Frames
Homogeneous Coordinates
Transformation Matrices

[Angel, Ch. 4]

January 24, 2002
Frank Pfenning
Carnegie Mellon University
http://www.cs.cmu.edu/~fp/courses/graphics/

Compilation Under Windows (Answer)

• Must install GLUT
• Good source: http://www.opengl.org/
• Includes should be
#include <GL/glut.h>
#include <stdlib.h>
• Do not include <GL/gl.h> or <GL/glu.h>
• Run on lab machines before handing in!

Geometric Objects and Operations

• Primitive types: scalars, vectors, points
• Primitive operations: dot product, cross product
• Representations: coordinate systems, frames
• Implementations: matrices, homogeneous coor.
• Transformations: rotation, scaling, translation
• Composition of transformations
• OpenGL transformation matrices

Scalars

• Scalars α, β, γ from a scalar field
• Operations α+β, α·β, 0, 1, -α, ( )¹
• "Expected" laws apply
• Examples: rationals or reals with addition and multiplication

Vectors

• Vectors u, v, w from vector space
• Includes scalar field
• Vector addition u + v
• Zero vector 0
• Scalar multiplication α v

01/24/2002 15-462 Graphics I 2

01/24/2002 15-462 Graphics I 3

01/24/2002 15-462 Graphics I 4

01/24/2002 15-462 Graphics I 5

01/24/2002 15-462 Graphics I 6
Points
• Points $P$, $Q$, $R$ from affine space
• Includes vector space
• Point-point subtraction $v = P - Q$
• Define also $P = v + Q$

Euclidean Space
• Assume vector space over real number
• Dot product: $\alpha = u \cdot v$
• $0 \cdot 0 = 0$
• $u, v$ are orthogonal if $u \cdot v = 0$
• $|v|^2 = v \cdot v$ defines $|v|$, the length of $v$
• Generally work in an affine Euclidean space

Geometric Interpretations
• Lines and line segments
• Convexity
• Dot product and projections
• Cross product and normal vectors
• Planes

Lines and Line Segments
• Parametric form of line: $P(\alpha) = P_0 + \alpha d$
• Line segment between $Q$ and $R$:
  $P(\alpha) = (1 - \alpha) Q + \alpha R$ for $0 \leq \alpha \leq 1$

Convex Hull
• Convex hull defined by
  $P = \alpha_1 P_1 + \ldots + \alpha_n P_n$
  for $\alpha_1 + \ldots + \alpha_n = 1$
  and $0 \leq \alpha_i \leq 1, i = 1, \ldots, n$

Projection
• Dot product projects one vector onto other
  $u \cdot v = |u| |v| \cos(\theta)$
  [diagram correction: $x = u$]
Normal Vector

- Cross product defines normal vector
  \[ u \times v = n \]
  \[ |u \times v| = |u| |v| |\sin(\theta)| \]
- Right-hand rule

Plane

- Plane defined by point \( P_0 \) and vectors \( u \) and \( v \)
- \( u \) and \( v \) cannot be parallel
- Parametric form: \( T(\alpha, \beta) = P_0 + \alpha u + \beta v \)
- Let \( n = u \times v \) be the normal
- Then \( n \cdot (P - P_0) = 0 \) iff \( P \) lies in plane

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Coordinate Systems

- Let \( v_1, v_2, v_3 \) be three linearly independent vectors in a 3-dimensional vector space
- Can write any vector \( w \) as
  \[ w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \]
  for scalars \( \alpha_1, \alpha_2, \alpha_3 \)
- In matrix notation:
  \[
  \begin{bmatrix}
  \alpha_1 \\
  \alpha_2 \\
  \alpha_3
  \end{bmatrix}
  \]

Frames

- Frame = coordinate system + origin \( P_0 \)
- Any point \( P = P_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \)
- Useful in with homogenous coordinates

Changes of Coordinate System

- Bases \( \{u_1, u_2, u_3\} \) and \( \{v_1, v_2, v_3\} \)
- Express basis vectors \( u_i \) in terms of \( v_j \)
  \[
  u_1 = \gamma_1 v_1 + \gamma_2 v_2 + \gamma_3 v_3 \\
  u_2 = \gamma_1 v_1 + \gamma_2 v_2 + \gamma_3 v_3 \\
  u_3 = \gamma_1 v_1 + \gamma_2 v_2 + \gamma_3 v_3
  \]
- Represent in matrix form
  \[
  \begin{bmatrix}
  u_1 \\
  u_2 \\
  u_3
  \end{bmatrix} = \begin{bmatrix}
  \gamma_1 \\
  \gamma_2 \\
  \gamma_3
  \end{bmatrix} \quad \text{for} \quad M = \begin{bmatrix}
  \gamma_1 & \gamma_1 & \gamma_1 \\
  \gamma_2 & \gamma_2 & \gamma_2 \\
  \gamma_3 & \gamma_3 & \gamma_3
  \end{bmatrix}
  \]
Map to Representations

- \( w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3, \quad a^T = [\alpha_1, \alpha_2, \alpha_3] \)
- \( w = \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3, \quad b^T = [\beta_1, \beta_2, \beta_3] \)
  \[
  \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  \end{bmatrix}
  \quad \Rightarrow \quad
  w = \begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
  \end{bmatrix}
  = b^T M
  \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  \end{bmatrix}
  
  \]

- So \( a = M^T b \) and \( b = (M^T)^{-1} a \)
- Suffices for rotation and scaling, not translation

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Linear Transformations

- 3 \times 3 matrices represent linear transformations \( a = M b \)
- Can represent rotation, scaling, and reflection
- Cannot represent translation
- \( a \) and \( b \) represent vectors, not points

\[
  w = a^T
  \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  \end{bmatrix}
  
  \]

Homogeneous Coordinates

- In affine space, \( P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0 \)
- Define \( 0 \cdot P = 0, \quad 1 \cdot P = P \)
- Then

\[
  P = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 1] 
  \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  1 \\
  \end{bmatrix}
  
  \]

- Point \( p = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 1]^T \)
- Vector \( w = \delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3 \)
- Homogeneous coords: \( a = [\delta_1 \quad \delta_2 \quad \delta_3 \quad 0]^T \)

Translation of Frame

- Express frame \((u_1, u_2, u_3, P_0)\) in \((v_1, v_2, v_3, Q_0)\)

\[
  \begin{bmatrix}
  u_1 \\
  u_2 \\
  u_3 \\
  Q_0 \\
  \end{bmatrix}
  = M
  \begin{bmatrix}
  v_1 \\
  v_2 \\
  v_3 \\
  P_0 \\
  \end{bmatrix}
  
  \]

Homogeneous Coordinates Summary

- Points \([\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 1]^T\)
- Vectors \([\delta_1 \quad \delta_2 \quad \delta_3 \quad 0]^T\)
- Change of frame

\[
  M = 
  \begin{bmatrix}
  711 & 712 & 713 & 0 \\
  721 & 722 & 723 & 0 \\
  731 & 732 & 733 & 0 \\
  741 & 742 & 743 & 1 \\
  \end{bmatrix}
  
  \]
Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Affine Transformations

- Translation
- Rotation
- Scaling
- Any composition of the above
- Express in homogeneous coordinates
- Need 4 × 4 matrices
- Later: projective transformations
- Also expressible as 4 × 4 matrices!

Translation

- \( \mathbf{p}' = \mathbf{p} + \mathbf{d} \) where \( \mathbf{d} = [\alpha_x \, \alpha_y \, \alpha_z \, 0]^T \)
- \( \mathbf{p} = [x \, y \, z \, 1]^T \)
- \( \mathbf{p}' = [x' \, y' \, z' \, 1]^T \)
- Express in matrix form \( \mathbf{p}' = \mathbf{T} \mathbf{p} \) and solve for \( \mathbf{T} \)

Scaling

- \( x' = \beta_x x \)
- \( y' = \beta_y y \)
- \( z' = \beta_z z \)
- Express as \( \mathbf{p}' = \mathbf{S} \mathbf{p} \) and solve for \( \mathbf{S} \)

Rotation in 2 Dimensions

- Rotation by \( \theta \) about the origin
- \( x' = x \cos \theta - y \sin \theta \)
- \( y' = x \sin \theta + y \cos \theta \)
- Express in matrix form
- \( \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \)
- Note determinant is 1

Rotation in 3 Dimensions

- Decompose into rotations about \( x, y, z \) axes
- \( \mathbf{R}_z = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \)
- \( \mathbf{R}_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & 0 & -\sin \theta \\ 0 & 0 & \cos \theta & \sin \theta \\ 0 & 0 & 0 & 1 \end{bmatrix} \)
Compose by Matrix Multiplication

- \( R = R_z \cdot R_y \cdot R_x \)
- Applied from right to left
- \( R \cdot p = (R_z(R_y(R_x(p)))) \)
- “Postmultiplication” in OpenGL

Rotation About a Fixed Point

- First, translate to the origin
- Second, rotate about the origin
- Third, translate back
- To rotate by \( \theta \) in about \( z \) around \( p \)
  \[ M = T(p) \cdot R_z(\theta) \cdot T(-p) = \ldots \]

Deriving Transformation Matrices

- Other examples: see [Angel, Ch. 4.8]
- See also Assignment 2 when it is out
- Hint: manipulate matrices, but remember geometric intuition

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Current Transformation Matrix

- Model-view matrix (usually affine)
- Projection matrix (usually not affine)
- Manipulated separately
  - \texttt{glMatrixMode(GL\_MODELVIEW);} 
  - \texttt{glMatrixMode(GL\_PROJECTION);} 

Manipulating the Current Matrix

- Load or postmultiply
  - \texttt{glLoadIdentity();} 
  - \texttt{glLoadMatrixf(*m);} 
  - \texttt{glMultMatrixf(*m);} 
- Library functions to compute matrices
  - \texttt{glTranslatef(dx, dy, dz);} 
  - \texttt{glRotatef(angle, vx, vy, vz);} 
  - \texttt{glScalef(sx, sy, sz);} 
- Recall: last transformation is applied first!
Summary

• Vector Spaces
• Affine and Euclidean Spaces
• Frames
• Homogeneous Coordinates
• Transformation Matrices
• OpenGL Transformation Matrices

OpenGL Tutors by Nate Robins

• Run under Windows
• Available at http://www.xmission.com/~nate/tutors.html
• Example: Transformation tutor