Affine Transformations & Homogeneous Coordinates

Wolfgang Heidrich

Course News

Assignment 1
- Due March 31
- More at end of lecture

Homework 1
- Exercise problems for transformations
- Discussed in labs next week

Reading
- Chapter 5

The Rendering Pipeline

Recap: Modeling and Viewing Transformation

Affine transformations
- Linear transformations + translations
- Can be expressed as a 3x3 matrix + 3 vector

\[ x' = M \cdot x + t \]

Recap: Compositing of Affine Transformations

In general:
- Transformation of geometry into coordinate system
  where operation becomes simpler
- Perform operation
- Transform geometry back to original coordinate system

Recap: Compositing of Affine Transformations

Example: 2D rotation around arbitrary center

Consider this transformation

\[ x' = \text{Id} \cdot \left( R(\phi) \cdot \left( \text{Id} \cdot x - t \right) \right) + t \]

i.e.: 

\[
\begin{pmatrix}
    x' \\
    y'
\end{pmatrix} = 
\begin{pmatrix}
    1 & 0 & 0 \\
    0 & 1 & 0 \\
\end{pmatrix} \cdot 
\begin{pmatrix}
    \cos \phi & -\sin \phi \\
    \sin \phi & \cos \phi \\
\end{pmatrix} \cdot 
\begin{pmatrix}
    1 & 0 & x - a \\
    0 & 1 & y - b \\
\end{pmatrix} + 
\begin{pmatrix}
    a \\
    b
\end{pmatrix}
\]
Our composite example is a rotation around an arbitrary 2D point with position t!
Recap: Compositing of Affine Transformations

Second Interpretation:
Step 1: translate frame (move origin to object)

Recap: Compositing of Affine Transformations

Second Interpretation:
Step 2: rotate frame by -Φ (inverse of rot. by Φ)

Recap: Compositing of Affine Transformations

Second Interpretation:
Step 3: translate frame back (vector t in new frame!)

Recap: Compositing of Affine Transformations

NOTES:
- All transformations are always with respect to the current coordinate frame
- The results of both interpretations are identical
  - Note that the object has the same relative position and orientation with respect to the coordinate frame!

Compositing of Affine Transformations

Another Example: 3D rotation around arbitrary axis

Rotate axis to z-axis
Rotate by Φ around z-axis
Rotate z-axis back to original axis

Composite transformation:
\[ R(v, \phi) = R_z(\alpha) \cdot R_y(\beta) \cdot R_z(\phi) \cdot R_y(\beta) \cdot R_z(\alpha) \]
\[ = (R_z(\beta) \cdot R_z(\alpha))^{-1} \cdot R_z(\phi) \cdot (R_y(\beta) \cdot R_z(\alpha)) \]

Compositing of Affine Transformations

Yet another example (on whiteboard):

\[ \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} \cos \frac{\pi}{4} & -\sin \frac{\pi}{4} \\ \sin \frac{\pi}{4} & \cos \frac{\pi}{4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \]
**Properties of Affine Transformations**

**Definition:**
A linear combination of points or vectors is given as

\[ x = \sum_{i=1}^{n} a_i \cdot x_i, \text{ for } a_i \in \mathbb{R} \]

An affine combination of points or vectors is given as

\[ x = \sum_{i=1}^{n} a_i \cdot x_i, \text{ with } \sum_{i=1}^{n} a_i = 1 \]

**Example:**

- Affine combination of 2 points
  
  \[ x = a_1 \cdot x_1 + a_2 \cdot x_2 \]
  
  with \( a_1 + a_2 = 1 \)
  
  \[ = (1 - a_2) \cdot x_1 + a_2 \cdot x_2 \]
  
  \[ = x_1 + a_2 \cdot (x_2 - x_1) \]

**Definition:**
A convex combination is an affine combination where all the weights \( a_i \) are positive

Note: this implies \( 0 \leq a_i \leq 1, \text{ i=1...n} \)

**Example:**

- Convex combination of 3 points
  
  \[ x = \alpha \cdot x_1 + \beta \cdot x_2 + \gamma \cdot x_3 \]
  
  with \( \alpha + \beta + \gamma = 1 \), \( 0 \leq \alpha, \beta, \gamma \leq 1 \)

\( \alpha, \beta, \gamma \) are called Barycentric coordinates

**Theorem:**
The following statements are synonymous

- A transformation \( T(x) \) is affine, i.e.:
  
  \[ x' = T(x) := M \cdot x + t, \]
  
  for some matrix \( M \) and vector \( t \)

- \( T(x) \) preserves affine combinations, i.e.:
  
  \[ T(\sum_{i=1}^{n} a_i \cdot x_i) = \sum_{i=1}^{n} a_i \cdot T(x_i), \text{ for } \sum_{i=1}^{n} a_i = 1 \]

- \( T(x) \) maps parallel lines to parallel lines

**Preservation of affine combinations:**
- Can compute transformation of every point on line or triangle by simply transforming the control points
Homogeneous Coordinates

*Homogeneous representation of points:* Add an additional component \( w = 1 \) to all points. All multiples of this vector are considered to represent the same 3D point.

- Use square brackets (rather than round ones) to denote homogeneous coordinates (different from text book)

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
= \begin{pmatrix}
  x' \\
  y' \\
  z' \\
  w
\end{pmatrix} \quad \forall w \neq 0
\]

Geometrically In 2D

*Cartesian Coordinates:*

\[
x
\]

\[
y
\]

Homogeneous Matrices

*Combining the two matrices into one:*

\[
\begin{pmatrix}
  m_{11} & m_{12} & m_{13} & 0 \\
  m_{21} & m_{22} & m_{23} & 0 \\
  m_{31} & m_{32} & m_{33} & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
= \begin{pmatrix}
  m_{11} & m_{12} & m_{13} & t_x \\
  m_{21} & m_{22} & m_{23} & t_y \\
  m_{31} & m_{32} & m_{33} & t_z \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

Geometrically In 2D

*Homogeneous Coordinates:*

\[
\begin{pmatrix}
  x' \\
  y' \\
  z' \\
  w
\end{pmatrix}
= \begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]

Homogeneous Matrices

*Affine Transformations:*

\[
\begin{pmatrix}
  m_{11} & m_{12} & m_{13} & 0 \\
  m_{21} & m_{22} & m_{23} & 0 \\
  m_{31} & m_{32} & m_{33} & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
= \begin{pmatrix}
  m_{11} & m_{12} & m_{13} & t_x \\
  m_{21} & m_{22} & m_{23} & t_y \\
  m_{31} & m_{32} & m_{33} & t_z \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

Homogeneous Coordinates – Composite Transformations

*Example: 2D rotation around arbitrary center*

This:

\[
x' = \text{Id} \cdot (R(\phi) \cdot (\text{Id} \cdot x - t)) + t
\]

translates by \( t \)

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix}
= \begin{pmatrix}
  \cos \phi & -\sin \phi \\
  \sin \phi & \cos \phi
\end{pmatrix}
\begin{pmatrix}
  x \\
  y
\end{pmatrix}
= \begin{pmatrix}
  \cos \phi & -\sin \phi \\
  \sin \phi & \cos \phi
\end{pmatrix}
\begin{pmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix}
+ \begin{pmatrix}
  t_x \\
  t_y
\end{pmatrix}
\]

Corresponds to this in full expansion:

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix}
= \begin{pmatrix}
  \cos \phi & -\sin \phi \\
  \sin \phi & \cos \phi
\end{pmatrix}
\begin{pmatrix}
  1 & 0 & (a) \\
  0 & 1 & (b)
\end{pmatrix}
+ \begin{pmatrix}
  (a) \\
  (b)
\end{pmatrix}
\]
Homogeneous Coordinates – Composite Transformations

Example: 2D rotation around arbitrary center

Euclidean coordinates:
\[
\begin{align*}
(x') &= \begin{pmatrix} 1 & 0 \\ -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \\
(y') &= \begin{pmatrix} 0 & 1 \\ \cos \phi & \sin \phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}
\end{align*}
\]

Homogeneous coordinates:
\[
\begin{align*}
(x') &= \begin{pmatrix} 1 & 0 & a \\ -\sin \phi & \cos \phi & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \\
(y') &= \begin{pmatrix} 0 & 1 & 0 \\ \cos \phi & \sin \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}
\end{align*}
\]

Homogeneous Transformations

Notes:
- A composite transformation is now just the product of a few matrices
- Rather than multiply each point sequentially with 3 matrices, first multiply the matrices, then multiply each point with only one (composite) matrix
  - Much faster for large \# of points!
- The composite matrix describing the affine transformation always has the bottom row 0,0,1 (2D), or 0,0,0,1 (3D)

Homogeneous Matrices

Note:
- Multiplication of the matrix with a constant does not change the transformation!
\[
\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & t_x & k \\ m_{21} & m_{22} & m_{23} & t_y & k \\ m_{31} & m_{32} & m_{33} & t_z & k \\ 0 & 0 & 0 & 1 & k \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
\]

Homogeneous Vectors

Earlier discussion describes points only
- What about vectors (directions)?
- What is the affine transformation of a vector?
  - Rotation
  - Scaling
  - Translation

Vectors are invariant under translation!

Homogeneous Vectors

Representing vectors in homogeneous coordinates
- Need representation that is only affected by linear transformations, but not by translations
- This is achieved by setting \( w = 0 \)
\[
\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & t_x & 0 \\ m_{21} & m_{22} & m_{23} & t_y & 0 \\ m_{31} & m_{32} & m_{33} & t_z & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
\]

Homogeneous Coordinates

Properties
- Unified representation as 4-vector (in 3D) for
  - Points
  - Vectors / directions
- Affine transformations become 4x4 matrices
- Composing multiple affine transformations involves simply multiplying the matrices
- 3D affine transformations have 12 degrees of freedom
- Need mapping of 4 points to uniquely define transformation
The Rendering Pipeline

Modeling Transformation

- Map geometry from local object coordinate system into a global world coordinate system
- Same as placing objects

Transformations:
- Arbitrary affine transformations are possible
- Even more complex transformations may be desirable, but are not available in hardware
- Freeform deformations

Viewing Transformation

- Map geometry from world coordinate system into camera coordinate system
- Camera coordinate system is right-handed, viewing direction is negative z-axis
- Same as placing camera

Transformations:
- Usually only rigid body transformations
  - Rotations and translations
  - Objects have same size and shape in camera and world coordinates

Model/View Transformation

- Combine modeling and viewing transform.
- Combine both into a single matrix
- Saves computation time if many points are to be transformed
- Possible because the viewing transformation directly follows the modeling transformation without intermediate operations

Coming Up

Next time:
- Transformation hierarchies
- OpenGL commands for transformations/drawing

Next week:
- Perspective transformations