Geometric Transformations

- Geometry defined in terms of points
  - E.g., vertices of shapes (lines, polygons,…)
- Points are represented by vectors of coordinates
  - Column vectors in OpenGL, Unity (and math)
  - Row vectors in early computer graphics (e.g., IRIS GL), and currently in Direct3D and XNA
- We will use column vectors here
- Start with 2D, extend to 3D

$$P = \begin{bmatrix} x \\ y \end{bmatrix}$$
Translation

- Translation of $x$ by $d_x$, $y$ by $d_y$
  - $x' = x + d_x$, $y' = y + d_y$

- Defining
  - $P = \begin{bmatrix} x \\ y \end{bmatrix}$, $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$, $T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$

- Then
  - $P' = P + T = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$

Scale

- $x' = s_x \cdot x$, $y' = s_y \cdot y$

- Defining
  - $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$,

- Then
  - $P' = S \cdot P = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$.

- Scale is about the origin!
Derivation of 2D Rotation

- \( x = r \cos \Phi \)
- \( y = r \sin \Phi \)

- \( x' = r \cos (\theta + \Phi) = r \cos \Phi \cos \theta - r \sin \Phi \sin \theta \)
- \( y' = r \sin (\theta + \Phi) = r \cos \Phi \sin \theta + r \sin \Phi \cos \theta \)

\[ \begin{bmatrix} x' \\ y' \end{bmatrix} \]

\[ \begin{bmatrix} x \\ y \end{bmatrix} \]

Rotation

- \( x' = x \cos \theta - y \sin \theta \)
- \( y' = x \sin \theta + y \cos \theta \)
- Defining 
  \[ R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \]
  \[ P' = R \cdot P = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \]
- Positive angles are counterclockwise
- Rotation is about the origin
Homogeneous Coordinates

- Problem: Translation
  - $P' = P + T$
- is expressed differently from scale and rotation
  - $P' = S \cdot P$
  - $P' = R \cdot P$
- Solution: Homogeneous coordinates
  - allow all 2D transformations to be expressed as multiplication with a $3 \times 3$ matrix
  - allow all 2D points and vectors (points at infinity) to be expressed as a 3 element vector

The point $\begin{bmatrix} x \\ y \end{bmatrix}$ is represented as $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, for nonzero $W$

Vectors have the form $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$

Note: Vectors (points at infinity) reside in the $w = 0$ plane
Homogeneous Transformations

Given $P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, $P' = M \cdot P$, where $M$ is

Translation $T(d_x,d_y) = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$

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

Rotation $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Composition of Transformations

- $R(\theta)$ rotates about the origin
- To rotate about point $P_1 = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$:
  - Translate $P_1$ to the origin
  - Rotate by $R(\theta)$
  - Translate origin back to $P_1$

$T(x_i,y_i) \cdot R(\theta) \cdot T(-x_i,-y_i) = \begin{bmatrix} 1 & 0 & x_i \\ 0 & 1 & y_i \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_i \\ 0 & 1 & -y_i \\ 0 & 0 & 1 \end{bmatrix}$

Feiner, COMS W4172, Spring 2014
Example: Transforming a Template

- To scale and rotate a template about point $P_1$ and position at point $P_2$
  - Translate $P_1$ to the origin
  - Scale
  - Rotate
  - Translate origin to $P_2$

Matrix Multiplication

- Matrix multiplication is
  - Associative
- but not in general
  - Commutative

- However, $M_1 \cdot M_2$ is commutative in the following 2D cases

$$
\begin{array}{c|c}
M_1 & M_2 \\
\hline
\text{Translate} & \text{Translate} \\
\text{Scale} & \text{Scale} \\
\text{Rotate} & \text{Rotate} \\
\text{Scale, where } s_x = s_y & \text{Rotate} \\
\text{Scale} & \text{Rotate, where } \theta = n * 180^\circ \text{ for integral } n
\end{array}
$$
Rigid Body Transformations

- Rigid body transformations result from any sequence of rotations and translations.
- Upper left 2×2 submatrix rows:
  - are unit vectors (length = 1)
  - are perpendicular (dot product = 0)
  - rotate into the x and y axes
- Upper left 2×2 submatrix columns:
  - are unit vectors (length = 1)
  - are perpendicular (dot product = 0)
  - are vectors into which the x and y axes rotate

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & d''_x \\
\sin \theta & \cos \theta & d''_y \\
0 & 0 & 1
\end{bmatrix}
\]

Affine Transformations

- Affine transformations are defined by an arbitrary sequence of rotation, scale, and translation transformations.
- They preserve:
  - parallelism of lines
- but not:
  - length of lines
  - angle between lines
- The matrix is of the form:

\[
\begin{bmatrix}
r s_{11} & r s_{12} & d'_x \\
r s_{21} & r s_{22} & d'_y \\
0 & 0 & 1
\end{bmatrix}
\]
Shear Transformations

\[
SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} x+a \cdot y \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

\[
SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} x \\ b \cdot x+y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

Shears are affine.

3D Coordinate Systems

- **OpenGL and XNA**
  - Right-handed coordinate system
  - Looking toward origin from + axis, a 90° CCW rotation takes one + axis into another
    - \(x \rightarrow y, y \rightarrow z, z \rightarrow x\)

- **Direct3D and Unity**
  - Left-handed coordinate system
  - Looking toward origin from + axis, a 90° CW rotation takes one + axis into another
    - \(x \rightarrow y, y \rightarrow z, z \rightarrow x\)
Direct3D and XNA vs. OpenGL and Unity

- Direct3D and XNA
  - Points documented as row vectors
  - Matrices written conventionally
    - Stored in row-major order
    - $v' = vA\,B\,C$
  - RH coord sys (XNA) vs. LH coord sys (Direct3D)

- OpenGL and Unity
  - Points documented as column vectors
    - (Same vector as a point in Direct3D)
  - Matrices written conventionally (Transpose of a matrix in Direct3D)
    - But, stored in column-major order!
    - (Same array as a matrix in Direct3D)
  - $v' = C\,B\,A\,v$
  - RH coord sys (OpenGL) vs. LH coord sys (Unity)

Why? To maintain code compatibility between original IRIS GL and later OpenGL! Vectors and arrays are stored and processed identically in each. Only the documentation differs.