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 Unity, OpenGL, (and math)
  - Row vectors in early computer graphics (e.g., IRIS GL), and currently in Direct3D
- We will use column vectors here
- Start with 2D, extend to 3D
Translation

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

- Defining
  $$P = [x \ y], P' = [x' \ y'], T = [d_x \ d_y]$$

- Then
  $$P' = P + T = [x' \ y'] = [x \ y] + [d_x \ d_y]$$

Scale

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

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

- Then
  $$P' = S \cdot P = [x' \ y'] = \begin{bmatrix} s_x & 0 \\ s_x & y \end{bmatrix} = [s_x & 0] \begin{bmatrix} x \\ y \end{bmatrix}.$$
Derivation of 2D Rotation

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

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

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 points and vectors (points at infinity) to be expressed as a 3 element vector
  - allow all 2D transformations to be expressed as multiplication with a 3×3 matrix

The point \( \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \) is represented as \( \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} Wx \\ Wy \\ W \end{bmatrix}, \) for nonzero \( W \)

**Homogeneous Coordinates**

Point
\[
\begin{bmatrix} X \\ Y \\ W \end{bmatrix} = \begin{bmatrix} WX \\ WY \\ W \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x/W \\ y/W \\ 1 \end{bmatrix}, \text{ for nonzero } W
\]

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 \( P_1 \)

\[
T(x_1,y_1) \cdot R(\theta) \cdot T(-x_1,-y_1) = \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
= \begin{bmatrix} \cos \theta & -\sin \theta & x_1(1-\cos \theta) + y_1 \sin \theta \\ \sin \theta & \cos \theta & y_1(1-\cos \theta) - x_1 \sin \theta \\ 0 & 0 & 1 \end{bmatrix}
\]

Rotate about origin

\( P_1 \) Original house

Rotate about \( P_1 \)

After translation of \( P_1 \) to origin

After rotation by \( \theta \)

After translation back to original \( P_1 \)
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 \textit{not} in general
  - Commutative

- However, \( M_1 \cdot M_2 \) is commutative in the following 2D cases
  
  \[
  \begin{array}{c}
  M_1 \\
  \text{Translate} \\
  \text{Scale} \\
  \text{Rotate} \\
  \text{Scale, where } s_x = s_y \\
  \text{Scale}
  \end{array}
  \begin{array}{c}
  M_2 \\
  \text{Translate} \\
  \text{Scale} \\
  \text{Rotate} \\
  \text{Rotate, where } \theta = n \cdot 180^\circ \text{ for integral } n
  \end{array}
  \]
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

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

Upper left 2×2 matrix is special orthogonal:
\[AA^T = I, \quad \det A = 1\]

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

\[
\begin{bmatrix}
rs_{11} & rs_{12} & d'_{x} \\
rs_{21} & rs_{22} & d'_{y} \\
0 & 0 & 1
\end{bmatrix}
\]
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
  - 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 vs. OpenGL and Unity

- **Direct3D**
  - Points documented as *row* vectors
  - Matrices written *conventionally*
    - Stored in *row-major order*
  - \[ \mathbf{v}' = \mathbf{v} A B C \]
  - 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)
  - \[ \mathbf{v}' = C B A \mathbf{v} \]
  - 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.

3D Points

- **OpenGL and Unity**
  - 4 element column vector
  - \[ [x \ y \ z \ 1] \]
- **Direct3D**
  - 4 element row vector
  - \[
    \begin{bmatrix}
      x \\
      y \\
      z \\
      1
    \end{bmatrix}
  \]

Note:
- Points in Unity are typically expressed as 3 element vectors
- Transformations in Unity are typically expressed as specific fields in the UI and functions in code
- Rotations are represented internally in Unity as *quaternions*
### 3D Transformations (for column vectors)

**3D Rigid Body Transformations**

- **Upper left $3 \times 3$ submatrix rows**
  - are unit vectors (length = 1)
  - are mutually perpendicular (dot product = 0)
  - rotate into the $x$, $y$, and $z$ axes
- **Upper left $3 \times 3$ submatrix columns**
  - are unit vectors (length = 1)
  - are mutually perpendicular (dot product = 0)
  - are vectors into which the $x$, $y$, and $z$ axes rotate

\[
T(d_x,d_y,d_z) = \begin{bmatrix}
1 & 0 & 0 & d_x \\
0 & 1 & 0 & d_y \\
0 & 0 & 1 & d_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
R_x(\theta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{bmatrix}
\]

\[
R_y(\theta) = \begin{bmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{bmatrix}
\]

\[
R_z(\theta) = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
S(s_x,s_y,s_z) = \begin{bmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & s_z
\end{bmatrix}
\]

\[
R_x(\theta) = \begin{bmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{bmatrix}
\]

\[
R_y(\theta) = \begin{bmatrix}
\cos \theta & 0 & -\sin \theta \\
0 & 1 & 0 \\
\sin \theta & 0 & \cos \theta
\end{bmatrix}
\]

\[
R_z(\theta) = \begin{bmatrix}
\cos \theta & \sin \theta & 0 \\
-\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]