2D Transformations

• Represent transformations by matrices
• To transform a point, represented by a vector, multiply the vector by the appropriate matrix.
• To transform line segments, transform endpoints
• To transform polygons, transform vertices
2D Transformations

- Scale (stretch) by a factor of \( k \)

\[
M = \begin{bmatrix}
k & 0 \\
0 & k
\end{bmatrix}
\]

(\( k = 2 \) in the example)
2D Transformations

• Scale by a factor of \((S_x, S_y)\)

\[
M = \begin{bmatrix}
S_x & 0 \\
0 & S_y
\end{bmatrix}
\]  

(Above, \(S_x = 1/2, S_y = 1\))
2D Transformations

- Rotate around origin by $\theta$ (Orthogonal)

\[
M = \begin{vmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta 
\end{vmatrix}
\]

(Above, $\theta=90^\circ$)
2D Transformations

- Flip over y axis (Orthogonal)

\[
M = \begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\]

Flip over x axis is ?
2D Transformations

- Shear along x axis

\[ M = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \]

Shear along y axis is?
2D Transformations

- Translation \( (\mathbf{P}_{\text{new}} = \mathbf{P} + \mathbf{T}) \)

\[ \mathbf{M} = ? \]
Homogenous Coordinates

- Represent 2D points by 3D vectors
- \((x,y)\rightarrow(x,y,1)\)
- Now a multitude of 3D points \((x,y,W)\) represent the same 2D point, \((x/W, y/W, 1)\)
- Represent 2D transforms with 3 by 3 matrices
- Can now do translations
- Homogenous coordinates have other uses/advantages (later)
2D Translation in H.C.

\[ P_{\text{new}} = P + T \]

\[ (x', y') = (x, y) + (t_x, t_y) \]

\[ M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \]
2D Scale in H.C.

\[ M = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
2D Rotation in H.C.

\[ M = \begin{bmatrix} 
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1 
\end{bmatrix} \]
Composition of Transformations (§5.4)

- If we use one matrix, $M_1$ for one transform and another matrix, $M_2$ for a second transform, then the matrix for the first transform followed by the second transform is simply $M_2M_1$
- This generalizes to any number of transforms
- Computing the combined matrix first can save lots of computation
Composition Example

- Matrix for rotation about a point, P
- Problem--we only know how to rotate about the origin.
Composition Example

- Matrix for rotation about a point, P
- Problem--we only know how to rotate about the origin.
- Solution--translate to origin, rotate, and translate back
2D transformations (continued)

- The transformations discussed so far are invertable (why?). What are the inverses?
2D viewing

- Three coordinate systems are common in graphics
  - World coordinates or modeling coordinates - where the model is defined (meters, miles, etc.)
  - Normalized device coordinates; usually (0-1) in each variable.
  - Device coordinates: the actual coordinates of the pixels on the frame-buffer or the printer
- Need to construct transformations between coordinate systems
- Terminology:
  - window = region on drawing that will be displayed (rectangle)
  - viewport = region in NDC’s/DC’s where this rectangle is displayed (often simply entire screen).
Determining the transform

- **Plan A**: Consider this as a sequence of transformations in homogenous coords, then determine each element in closed form.

- **Plan B**: Compute numerically from point correspondences.
• write \((w_{x_i}, w_{y_i})\) for coordinates of \(i\)'th point on window

• translation is:

\[
\begin{bmatrix}
  x' \\
y'
\end{bmatrix} = \overline{\begin{bmatrix} w_x & x \\
w_y & y \\
1 & 0
\end{bmatrix}} \\
\begin{bmatrix}
  1 \\
0 \\
0
\end{bmatrix}
\]

(overbar denotes average over vertices, i.e., 1,2,3,4)
Rotate to line up with axes

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

(Need to compute theta)
(Vertex order does not correspond, need to flip)
Scale and translate

\[
\begin{bmatrix}
\frac{w_{new}}{w_{old}} & 0 & x_{new} \\
0 & \frac{h_{new}}{h_{old}} & y_{new}
\end{bmatrix}
\]

Notice that choice of new width, height, and center give translation to either normalized device coords, or to device coordinates.
- Get overall transformation by multiplying transforms.
- This gives a single transformation matrix, whose elements are functions of window/viewport coordinates.

\[ x' = M_{\text{translate origin to viewport cog, scale}} \cdot M_{\text{flip}} \cdot M_{\text{rotate}} \cdot M_{\text{translate window cog->origin}} \cdot x \]

NDC’s/DC’s \hspace{1cm} World coords

\( \text{(cog==window center of gravity)} \)
Plan B: Solve for the affine transformations

• Another approach to determining the whole transform for the pipeline; this is an affine transform.
• Matrix form:

\[
\begin{bmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1
\end{bmatrix}
\]

• Now assume that we know that \( M_{p_1} = q_1, M_{p_2} = q_2, M_{p_3} = q_3 \)
• Quick way to determine transform, because this is the same as six linear equations, in six variables, which are the entries in the matrix (more details on next slide):

\[
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & a \\
0 & 0 & 0 & x_1 & y_1 & 1 & b \\
x_2 & y_2 & 1 & 0 & 0 & 0 & c \\
0 & 0 & 0 & x_2 & y_2 & 1 & d \\
x_3 & y_3 & 1 & 0 & 0 & 0 & e \\
0 & 0 & 0 & x_3 & y_3 & 1 & f
\end{bmatrix} = \begin{bmatrix}
u_1 \\
v_1 \\
u_2 \\
u_2 \\
u_3 \\
u_3
\end{bmatrix}
\]
- \( \text{Mp}_1 = q_1 \) gives first two rows
- \( p_1 = (x_1, y_1, 1)^T, \quad q_1 = (u_1, v_1, 1)^T \)

\[
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  y_1 \\
  1
\end{bmatrix}
= \begin{bmatrix}
  u_1 \\
  v_1 \\
  1
\end{bmatrix}
\]

\[
a x_1 + b y_1 + c = u_1
\]
\[
d x_1 + e y_1 + f = v_1
\]

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 & a & u_1 \\
  0 & 0 & 0 & x_1 & y_1 & 1 & b & v_1 \\
  x_2 & y_2 & 1 & 0 & 0 & 0 & c & u_2 \\
  0 & 0 & 0 & x_2 & y_2 & 1 & d & v_2 \\
  x_3 & y_3 & 1 & 0 & 0 & 0 & e & u_3 \\
  0 & 0 & 0 & x_3 & y_3 & 1 & f & v_3
\end{bmatrix}
\]

\[
\text{Mp}_2 = q_2, \text{Mp}_3 = q_3 \quad \text{give other rows}
\]