Transforms

Elements of Graphics
CS324e
Fall 2017
Student Presentation
Shapes and Hierarchies

- Shapes form complex structures via hierarchies
- Hierarchies make it easier to manipulate shapes and their structures
- Animation is a high-level form of this
- But how is this process done in practice?
Transformations

- Foundation of rendering in computer graphics
- Allows for manipulation of objects within a scene
Point Representation

- Represent a single vertex point $\mathbf{p}$ as a vector:
  $$\begin{bmatrix} x \\ y \end{bmatrix}$$

- Represent a 2-D transformation with matrix $\mathbf{M}$:
  $$\mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Multiply $\mathbf{p}$ by $\mathbf{M}$ to apply the transformation:
  $$\mathbf{p}' = \mathbf{M}\mathbf{p}$$
  $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
Multiplication

❖ How do we multiply?
\[
\begin{bmatrix}
 x' \\
 y'
\end{bmatrix} = \begin{bmatrix}
 a & b \\
 c & d
\end{bmatrix} \begin{bmatrix}
 x \\
 y
\end{bmatrix}
\]

\[
x' = ax + by \\
y' = cx + dy
\]

❖ What if we multiply by the identity matrix?
\[
\begin{bmatrix}
 1 & 0 \\
 0 & 1
\end{bmatrix}
\]

\[
x' = ax \\
y' = dy
\]
Scaling

- What happens with these two matrices applied on this square?

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}$$
Reflection

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

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

- What if we set $a = d = 1$ but then modify $b$?

\[
\begin{bmatrix}
1 & b \\
0 & 1
\end{bmatrix}
\]

\[
x' = x + by
\]

\[
y' = y
\]
Rotation

\[ M_R = R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \]
Linear Transformations

- All of these transformations are considered linear transformations
  - Scaling
  - Reflection
  - Shearing
  - Rotation
- What’s missing?
Affine Transformations

- We want objects to move, or translate, through space
- Linear space (for linear transformations) has no notion of “position”
- Therefore affine space takes linear space and adds an “origin” point
Homogenous Coordinates

- Give every point a third component:

\[ p' = Mp \]

\[
\begin{bmatrix}
a & b & t_x \\
c & d & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

\[
= [u \ v \ t]
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

\[ = x \cdot u + y \cdot v + 1 \cdot t \]
Vectors vs Points

- Note that \([a \ c \ 0]^T\) and \([b \ d \ 0]^T\) represent vectors whereas \([t_x \ t_y \ 1]^T\), \([x \ y \ 1]^T\) and \([x' \ y' \ 1]^T\) represent points.

\[
p' = Mp = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = x \cdot u + y \cdot v + 1 \cdot t
\]
Series of Transformations

- We can combine a sequence of transformations into one matrix to transform the geometric instance:

- But we can also think of this transformation as a series of simpler transformations:
Transformation Order

❖ Transformation order matters!
❖ Mathematical reason: transformation matrices do not commute under matrix multiplication
❖ Intuitive reason: what happens when we rotate then translate versus translate then rotate?
Proper Transformation Order

• To rotate an object within its current position:
  • Scale -> Rotate -> Translate

• To rotate around a specific point:
  • Scale -> Translate -> Rotate
Transformations in Processing

- \texttt{translate()}, \texttt{rotate()} and \texttt{scale()}
- \texttt{translate(x, y)} moves the objects by an \((x, y)\) offset
- \texttt{rotate(\Theta)} rotates the objects by \(\Theta\) radians
- \texttt{scale(p)} scales the objects by \(p\) percent
What is the order of transformation operations on this image (white square to black square?)

A. translate -> rotate -> scale
B. rotate -> translate -> scale
C. translate -> rotate
D. rotate -> translate
PushMatrix() and PopMatrix()

- `pushMatrix()` records the current state of the transformation matrix
- `popMatrix()` returns the transformation matrix to the previously recorded state
- These functions allow us to manipulate objects at different levels of hierarchy
- pushes and pops can be nested
Hands-on: Using Transformations

❖ Today’s activities:

1. Draw a rectangle in the upper left corner of the screen
2. Before the draw call, translate a sketch’s screen origin to the center of the screen
3. Scale then translate the sketch’s screen origin
4. Scale, rotate, translate the sketch’s screen origin
5. Scale, translate, rotate the sketch’s screen origin
6. Draw an additional object (also in the upper left corner of the screen), and apply a different set of transformations to it