Lecture 5: Transforms

Computer Graphics
CMU 15-462/15-662, Spring 2017
Cube

(-1, -1, -1)

(1, -1, -1)

(1, -1, 1)

(-1, -1, 1)

(-1, 1, -1)

(-1, 1, 1)

(1, 1, -1)

(1, 1, 1)
Cube man

The original cube

Stretched out cube moved up

Squishy cube moved to the right
$f$ transforms $x$ to $f(x)$
And what is our favorite type of transform?
Linear Transform

But what does it mean?

\[ u = (u_1, u_2) \]

\[ f(u) = u_1 a_1 + u_2 a_2 \]
Linear transforms

Transform $f$ is linear if and only if:

\[ f(x + y) = f(x) + f(y) \]

\[ f(ax) = af(x) \]
Scale

Uniform scale:

\[ S_a(x) = ax \]

Non-uniform scale??
Is scale a linear transform?

Yes!

\[ S_2(x) = 2x \]
\[ aS_2(x) = 2ax \]
\[ S_2(ax) = 2ax \]
\[ S_2(ax) = aS_2(x) \]

\[ S_2(x + y) = 2(x + y) \]
\[ S_2(x) + S_2(y) = 2x + 2y \]
\[ S_2(x + y) = S_2(x) + S_2(y) \]
Rotation

\[ R_\theta = \text{rotate counter-clockwise by } \theta \]
Rotation as Circular Motion

\[ R_\theta = \text{rotate counter-clockwise by } \theta \]

As angle changes, points move along circular trajectories.

Hence, rotations preserve length of vectors: \[ |R_\theta(x)| = |x| \]
Is rotation linear?

Yes!
Translation

\[ T_b(x) = \text{translate by } b \]

\[ T_b(x) = x + b \]
Is translation linear?

No. Translation is affine.
Reflection

Re_y = reflection about y

Re_x = reflection about x
Shear (in $x$ direction)
Compose basic transforms to construct more complex transforms

Note: order of composition matters

Top-right: scale, then translate
Bottom-right: translate, then scale

\[ f(x) = T_{3,1}(S_{0.5}(x)) \]

\[ f(x) = S_{0.5}(T_{3,1}(x)) \]
How would you perform these transformations?

1. **Shear:**
   \[ S_{1x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

2. **Reflection:**
   \[ S_{x} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \]

3. **Scale:**
   \[ S_{2x} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \]

4. **Shear:**
   \[ S_{1x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]
Common pattern: rotation about point x

Step 1: translate by -x

Step 2: rotate

Step 3: translate by x
Summary of basic transforms

**Linear:**

\[ f(x + y) = f(x) + f(y) \]
\[ f(ax) = af(x) \]

- Scale
- Rotation
- Reflection
- Shear

**Not linear:**

Translation

**Affine:**

Composition of linear transform + translation
(all examples on previous two slides)

\[ f(x) = g(x) + b \]

Not affine: perspective projection (will discuss later)

**Euclidean: (Isometries)**

Preserve distance between points (preserves length)

\[ |f(x) - f(y)| = |x - y| \]

- Translation
- Rotation
- Reflection

“Rigid body” transforms are Euclidean transforms that also preserve “winding” (does not include reflection)
Representing Transforms
Review: representing points in a coordinate space

Consider coordinate space defined by orthogonal vectors \( \mathbf{e}_1 \) and \( \mathbf{e}_2 \)

\[
x = 2\mathbf{e}_1 + 2\mathbf{e}_2
\]

\[
x = \begin{bmatrix} 2 \\ 2 \end{bmatrix}
\]

\[
x = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}
\] in coordinate space defined by \( \mathbf{e}_1 \) and \( \mathbf{e}_2 \), with origin at \((1.5, 1)\)

\[
x = \begin{bmatrix} \sqrt{8} \\ 0 \end{bmatrix}
\] in coordinate space defined by \( \mathbf{e}_3 \) and \( \mathbf{e}_4 \), with origin at \((0, 0)\)
Review: 2D matrix multiplication

\[
\begin{bmatrix}
ax + by \\
\end{bmatrix} = \begin{bmatrix}
a & b \\
\end{bmatrix} \begin{bmatrix}
x \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
cx + dy \\
\end{bmatrix} = \begin{bmatrix}
c & d \\
\end{bmatrix} \begin{bmatrix}
y \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
a \\
c \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
\end{bmatrix} + \begin{bmatrix}
b \\
d \\
\end{bmatrix} =
\]
Linear transforms in 2D can be represented as 2x2 matrices

- Scale
- Shear
- Rotation
- Translation?
Linear transforms in 2D can be represented as 2x2 matrices

Consider non-uniform scale: \( S_s = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \)

Scaling amounts in each direction: 
\( s = \begin{bmatrix} 0.5 & 2 \end{bmatrix}^T \)

Matrix representing scale transform:
\( S_s = \begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix} \)
Shear

Shear in x:

\[ H_{xs} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \]

Shear in y:

\[ H_{ys} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix} \]
Rotation matrix (2D)

Question: what happens to $(1, 0)$ and $(0,1)$ after rotation by $\theta$?

Reminder: rotation moves points along circular trajectories.

(Recall that $\cos \theta$ and $\sin \theta$ are the coordinates of a point on the unit circle.)

Answer:

$R_\theta(1, 0) = (\cos(\theta), \sin(\theta))$

$R_\theta(0, 1) = (\cos(\theta + \pi/2), \sin(\theta + \pi/2))$

Which means the matrix must look like:

$$R_\theta = \begin{bmatrix} \cos(\theta) & \cos(\theta + \pi/2) \\ \sin(\theta) & \sin(\theta + \pi/2) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$
Rotation matrix (2D): another way...

\[
R_\theta = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]
How do we compose linear transforms?

Compose linear transforms via matrix multiplication. This example: uniform scale, followed by rotation

\[ f(x) = R_{\pi/4} S_{[1.5,1.5]} x \]

Enables simple, efficient implementation: reduce complex chain of transforms to a single matrix multiplication.
Translation?

$T_b(x) = x + b$

Recall: translation is not a linear transform
→ Output coefficients are not a linear combination of input coefficients
→ Translation operation cannot be represented by a 2x2 matrix

$x_{\text{out}x} = x_x + b_x$

$x_{\text{out}y} = x_y + b_y$

Translation math
2D homogeneous coordinates (2D-H)

Key idea: represent 2D points in 3D coordinate space

So the point \((x, y)\) is represented as the 3-vector: \([x\ y\ 1]^T\)

And transforms are represented a 3\times3 matrices that transform these vectors.

For example: here are 2D scale and rotation transforms written in 2D homogeneous form:

\[ S_s = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_\theta = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Observe:
In these examples, the last row just propagates third coordinate of input to output.
Expressing transformations in 2D-H coords

Translation expressed as 3x3 matrix multiplication:

\[
T_b = \begin{bmatrix}
1 & 0 & b_x \\
0 & 1 & b_y \\
0 & 0 & 1
\end{bmatrix}
\]

\[
T_bx = \begin{bmatrix}
1 & 0 & b_x \\
0 & 1 & b_y \\
0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x_x \\
x_y \\
1
\end{bmatrix} = \begin{bmatrix}
x_x + b_x \\
x_y + b_y \\
1
\end{bmatrix}
\]

Homogeneous representation enables composition of affine transforms!

Example: rotation about point \( b \)

\[
T_b R_\theta T_{-b}
\]
Homogeneous coordinates: some intuition

Many points in 2D-H correspond to the same point in 2D.

\( \mathbf{x} \) and \( w \mathbf{x} \) correspond to the same 2D point.

(divide by \( w \) to convert 2D-H back to 2D)

Translation is a shear in \( x \) and \( y \) in 2D-H space:

\[
T_b \mathbf{x} = \begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w x_x \\ w x_y \\ w \end{bmatrix} = \begin{bmatrix} w x_x + w b_x \\ w x_y + w b_y \\ w \end{bmatrix}
\]
**Homogeneous coordinates: points vs. vectors**

2D-H points with $w = 0$ represent 2D vectors (think: directions are points at infinity)

Unlike 2D, points and directions are distinguishable by their representation in 2D-H.

Note: translation does not modify directions:

$$T_b v = \begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}$$
Visualizing 2D transformations in 2D-H

Original shape in 2D can be viewed as many copies, uniformly scaled by w.

2D scale ↔ scale x and y; preserve w (Question: what happens to 2D shape if you scale x, y, and w)

2D rotation ↔ rotate around w

2D translate ↔ shear in xy
Moving to 3D (and 3D-H)

Represent 3D transforms as 3x3 matrices and 3D-H transforms as 4x4 matrices

Scale:

$$S_s = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \quad S_s = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shear (in x, based on y,z position):

$$H_{x,d} = \begin{bmatrix} 1 & d_y & d_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad H_{x,d} = \begin{bmatrix} 1 & d_y & d_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translate:

$$T_b = \begin{bmatrix} 1 & 0 & 0 & b_x \\ 0 & 1 & 0 & b_y \\ 0 & 0 & 1 & b_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
Rotations in 3D

Rotation about x axis:

\[
R_{x,\theta} = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{bmatrix}
\]

Rotation about y axis:

\[
R_{y,\theta} = \begin{bmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{bmatrix}
\]

Rotation about z axis:

\[
R_{z,\theta} = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
Rotation about an arbitrary axis

To rotate by $\theta$ about $w$:

1. Form orthonormal basis around $w$ (see $u$ and $v$ in figure)
2. Rotate to map $w$ to $[0 0 1]$ (change in coordinate space)

$$R_{uvw} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix}$$

$$R_{uvw}u = [1 \ 0 \ 0]$$
$$R_{uvw}v = [0 \ 1 \ 0]$$
$$R_{uvw}w = [0 \ 0 \ 1]$$

3. Perform rotation about $z$: $R_{z, \theta}$

4. Rotate back to original coordinate space: $R_{uvw}^T$

$$R_{uvw}^{-1} = R_{uvw}^T = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}$$

$$R_{w, \theta} = R_{uvw}^T R_{z, \theta} R_{uvw}$$
Transformations summary

- Transformations can be interpreted as operations that move points in space
  - e.g., for modeling, animation

- Or as a change of coordinate system
  - e.g., screen and view transforms

- Construct complex transformations as compositions of basic transforms

- Homogeneous coordinate representation allows for expression of non-linear transforms (e.g., affine, perspective projection) as matrix operations (linear transforms) in higher-dimensional space
  - Matrix representation affords simple implementation and efficient composition
Further Reading

- Basic transforms are nicely covered here (*Real Time Rendering* -- Chapter 4. by T. Akenine Moller, E. Haines, N. Hoffman)
What you should know (Part 1 of 2)

1. Which of the following operations are linear transforms: scale, rotation, shear, translation, reflection, rotation about a point that is not the origin?
2. Express scale as a linear transform
3. Express rotation as a linear transform
4. Express rotation about a point as a linear transform
5. Express shear as a linear transform
6. Express reflection as a linear transform
7. Express translation as an affine transform
8. Know what makes a transform linear vs. affine
9. Know how to build transformation matrices from start and end configurations of your object
What you should know (Part 2 of 2)

- Create 2D and 3D transformation matrices to perform specific scale, shear, rotation, reflection, and translation operations
- Compose transformations to achieve compound effects
- Rotate an object about a fixed point
- Rotate an object about a given axis
- Create an orthonormal basis given a single vector
- Understand the equivalence of $[x \ y \ 1]$ and $[wx \ wy \ w]$ vectors
- Explain/illustrate how translations in 2D ($x, y$) are a shear operation in the homogeneous coordinate space ($x, y, w$)