Affine transformations

Geometric transformations will map points in one space to points in another: \((x',y',z') = f(x,y,z)\).

These transformations can be very simple, such as scaling each coordinate, or complex, such as non-linear twists and bends.

We'll focus on transformations that can be represented easily with matrix operations.

We'll start in 2D...

Reading

Required:
- Angel 4.6-4.10

Further reading:
- Angel, the rest of Chapter 4
- Foley, et al, Chapter 5.1-5.5.

Geometric transformations

Geometric transformations will map points in one space to points in another: \((x',y',z') = f(x,y,z)\).

These transformations can be very simple, such as scaling each coordinate, or complex, such as non-linear twists and bends.

We'll focus on transformations that can be represented easily with matrix operations.

We'll start in 2D...

Representation

We can represent a point, \(p = (x,y)\), in the plane

- as a column vector
  \[
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]
- as a row vector
  \[
  \begin{bmatrix}
  x & y
  \end{bmatrix}
  \]
Representation, cont.

We can represent a 2-D transformation $M$ by a matrix

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

If $p$ is a column vector, $M$ goes on the left:

$$p' = Mp$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

If $p$ is a row vector, $M^T$ goes on the right:

$$p' = pM^T$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

We will use column vectors.

Two-dimensional transformations

Here's all you get with a 2 x 2 transformation matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

So:

$$x' = ax + by$$

$$y' = cx + dy$$

We will develop some intimacy with the elements $a, b, c, d…$

Identity

Suppose we choose $a=d=1, b=c=0$:

- Gives the identity matrix:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Doesn't move the points at all

Scaling

Suppose we set $b=c=0$, but let $a$ and $d$ take on any positive value:

- Gives a scaling matrix:

$$\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$$

- Provides differential (non-uniform) scaling in $x$ and $y$:

$$x' = ax$$

$$y' = dy$$
Suppose we keep $b=c=0$, but let either $a$ or $d$ go negative.

Examples:

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

Now let's leave $a=d=1$ and experiment with $b$.

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

gives:
\[
x' = x + by \\
y' = y
\]

Effect on unit square

Let's see how a general 2 x 2 transformation $M$ affects the unit square:

\[
\begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
\begin{bmatrix}
p & q & r & s \\
0 & 1 & 1 & 0
\end{bmatrix}
= 
\begin{bmatrix}
0 & a & a+b & b \\
0 & c & c+d & d
\end{bmatrix}
\]

Effect on unit square, cont.

Observe:

- Origin invariant under $M$
- $M$ can be determined just by knowing how the corners $(1,0)$ and $(0,1)$ are mapped
- $a$ and $d$ give $x$- and $y$-scaling
- $b$ and $c$ give $x$- and $y$-shearing
Rotation

From our observations of the effect on the unit square, it should be easy to write down a matrix for "rotation about the origin":

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

Thus,

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

Degrees of freedom

How many degrees of freedom – free variables – does a 2X2 transformation have?

How many degrees of freedom does a 2D rotation have?

Limitations of the 2 x 2 matrix

A 2 x 2 linear transformation matrix allows

- Scaling
- Rotation
- Reflection
- Shearing

Q: What important operation does that leave out?

Homogeneous coordinates

We can loft the problem up into 3-space, adding a third component to every point:

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix}
= T(t)
\begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]

Adding the third "w" component puts us in homogenous coordinates.

Then, transform with a 3 x 3 matrix:

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 1 \\
0 & 1 & 1/2 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 1 \\
0 & 1 & 1/2 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]

… gives translation!
**Affine transformations**

The addition of translation to linear transformations gives us affine transformations.

In matrix form, 2D affine transformations always look like this:

\[
M = \begin{bmatrix}
a & b & t_x \\
c & d & t_y \\
0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
A & t \\
0 & 0 & 1
\end{bmatrix}
\]

2D affine transformations always have a bottom row of \([0 \ 0 \ 1]\).

An “affine point” is a “linear point” with an added \(w\)-coordinate which is always 1:

\[
P_{\text{aff}} = \begin{bmatrix}
x \\
y \\
1
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Applying an affine transformation gives another affine point:

\[
M P_{\text{aff}} = \begin{bmatrix}
A & t \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

**Rotation about arbitrary points**

Until now, we have only considered rotation about the origin.

With homogeneous coordinates, you can specify a rotation, \(\theta\), about any point \(q = [q_x \ q_y \ 1]^T\) with a matrix:

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 & t_x \\
\sin \theta & \cos \theta & 0 & t_y \\
0 & 0 & 1 & t_z
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

1. Translate \(q\) to origin
2. Rotate
3. Translate back

*Note:* Transformation order is important!!

**Basic 3-D transformations: scaling**

Some of the 3-D affine transformations are just like the 2-D ones.

In this case, the bottom row is always \([0 \ 0 \ 0 \ 1]\).

For example, scaling:

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
s_x & 0 & 0 & 0 \\
0 & s_y & 0 & 0 \\
0 & 0 & s_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

**Translation in 3D**

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]
Rotation in 3D

Rotation now has more possibilities in 3D:

\[
\begin{align*}
R_x(\alpha) &= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos\alpha & -\sin\alpha & 0 \\
0 & \sin\alpha & \cos\alpha & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \\
R_y(\beta) &= \begin{bmatrix}
\cos\beta & 0 & \sin\beta & 0 \\
0 & 1 & 0 & 0 \\
-\sin\beta & 0 & \cos\beta & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \\
R_z(\gamma) &= \begin{bmatrix}
\cos\gamma & -\sin\gamma & 0 & 0 \\
\sin\gamma & \cos\gamma & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{align*}
\]

Shearing in 3D

Shearing is also more complicated. Here is one example:

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
1 & b & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]

We call this a shear with respect to the x-z plane.

Properties of affine transformations

Here are some useful properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Midpoints map to midpoints (in fact, ratios are always preserved)

\[
\text{ratio} = \frac{s}{t} = \frac{\|p'q'\|}{\|qr\|}
\]
Affine transformations in OpenGL

OpenGL maintains a "modelview" matrix that holds the current transformation $M$.

The modelview matrix is applied to points (usually vertices of polygons) before drawing.

It is modified by commands including:

- $\text{glLoadIdentity()}$ \quad $M \leftarrow I$  
  - set $M$ to identity

- $\text{glTranslatef}(t_x, t_y, t_z)$ \quad $M \leftarrow MT$  
  - translate by $(t_x, t_y, t_z)$

- $\text{glRotatef}(\theta, x, y, z)$ \quad $M \leftarrow MR$  
  - rotate by angle $\theta$ about axis $(x, y, z)$

- $\text{glScalef}(s_x, s_y, s_z)$ \quad $M \leftarrow MS$  
  - scale by $(s_x, s_y, s_z)$

Note that OpenGL adds transformations by postmultiplication of the modelview matrix.

Summary

What to take away from this lecture:

- All the names in boldface.
- How points and transformations are represented.
- What all the elements of a 2 x 2 transformation matrix do and how these generalize to 3 x 3 transformations.
- What homogeneous coordinates are and how they work for affine transformations.
- How to concatenate transformations.
- The mathematical properties of affine transformations.