

Affine transformations

1

Reading

Required:

- ♦ Watt, Section 1.1.

Further reading:

- ♦ Foley, et al, Chapter 5.1-5.5.
- ♦ David F. Rogers and J. Alan Adams, *Mathematical Elements for Computer Graphics*, 2nd Ed., McGraw-Hill, New York, 1990, Chapter 2.

2

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...

3

Representation

We can represent a **point**, $\mathbf{p} = (x,y)$, in the plane

- ♦ as a column vector $\begin{bmatrix} x \\ y \end{bmatrix}$
- ♦ as a row vector $[x \ y]$

4

Representation, cont.

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

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

If \mathbf{p} is a column vector, M goes on the left:

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

If \mathbf{p} is a row vector, M^T goes on the right:

$$\mathbf{p}' = \mathbf{p}M^T$$
$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

We will use **column vectors**.

5

Two-dimensional transformations

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

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

So:

$$x' = ax + by$$

$$y' = cx + dy$$

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

6

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

7

Scaling

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

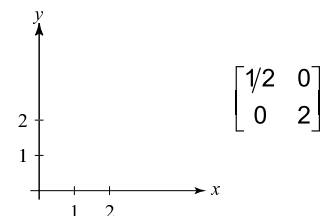
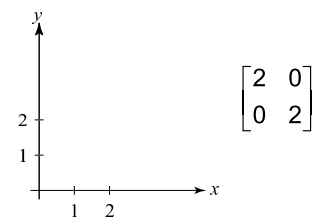
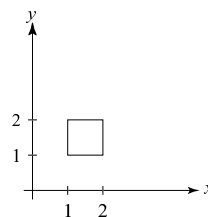
- Gives a **scaling** matrix:

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

- Provides **differential scaling** in x and y :

$$x' = ax$$

$$y' = dy$$



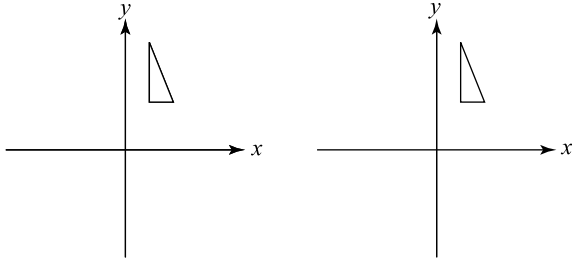
8

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

Examples:

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

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



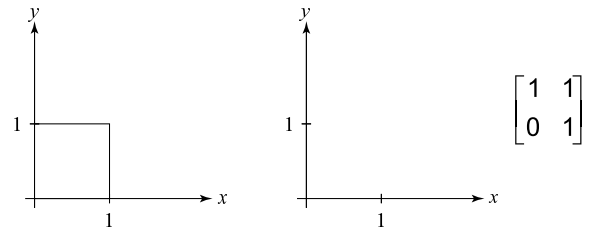
Now let's leave $a=d=1$ and experiment with b . . .

The matrix

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

gives:

$$\begin{aligned} x' &= x + by \\ y' &= y \end{aligned}$$

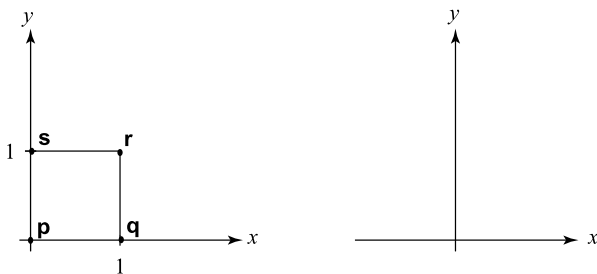


Effect on unit square

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

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} p & q & r & s \end{bmatrix} = \begin{bmatrix} p' & q' & r' & s' \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & a & a+b & b \\ 0 & c & c+d & d \end{bmatrix}$$



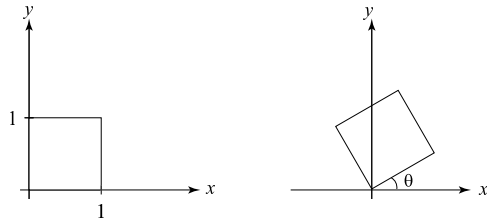
Effect on unit square, cont.

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”:



$$\diamond \begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow$$

$$\diamond \begin{bmatrix} 0 \\ 1 \end{bmatrix} \rightarrow$$

Thus,

$$M = R(\theta) = \begin{bmatrix} & \\ & \end{bmatrix}$$

13

Limitations of the 2 x 2 matrix

A 2 x 2 matrix allows

- ◆ Scaling
- ◆ Rotation
- ◆ Reflection
- ◆ Shearing

Q: What important operation does that leave out?

14

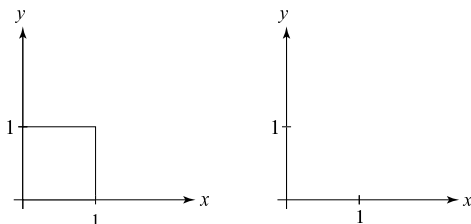
Homogeneous coordinates

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

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

And then transform with a 3 x 3 matrix:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = T(\mathbf{t}) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



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

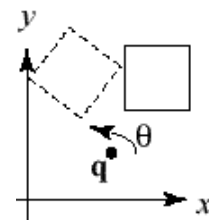
... gives **translation!**

15

Rotation about arbitrary points

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

With homogeneous coordinates, you can specify a rotation, θ , about any point $\mathbf{q} = [q_x \ q_y]^T$ with a matrix.



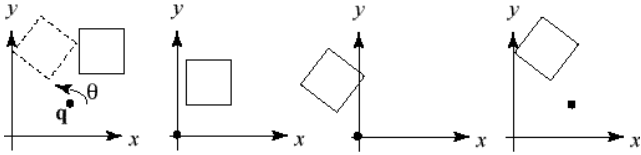
Q: how would you find the matrix for rotating about \mathbf{q} by θ ?

16

Rotation about arbitrary points

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

With homogeneous coordinates, you can specify a rotation, θ , about any point $\mathbf{q} = [q_x \ q_y]^T$ with a matrix:



1. Translate \mathbf{q} to origin
2. Rotate
3. Translate back

Note: Transformation order is important!!

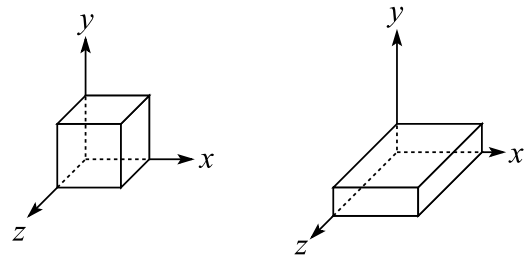
17

Basic 3-D transformations: scaling

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

For example, scaling:

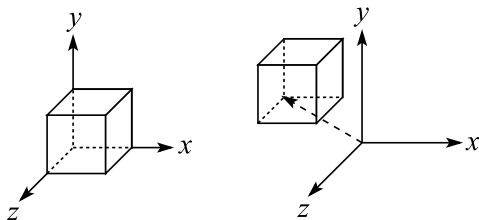
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



18

Translation in 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



19

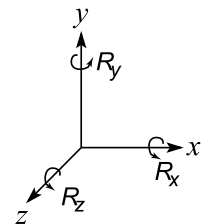
Rotation in 3D

Rotation now has more possibilities in 3D:

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

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

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



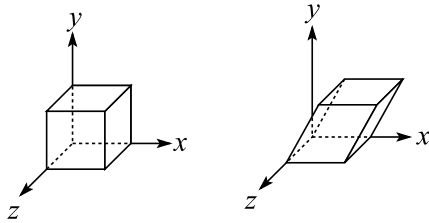
Use right hand rule

20

Shearing in 3D

Shearing is also more complicated. Here is one example:

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



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

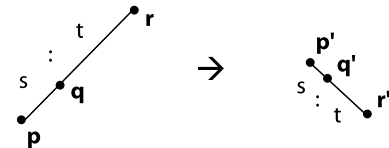
21

Properties of affine transformations

All of the transformations we've looked at so far are examples 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 along a line are always preserved)



$$\text{ratio} = \frac{\|pq\|}{\|qr\|} = \frac{s}{t} = \frac{\|p'q'\|}{\|q'r'\|}$$

22

Affine xforms 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:

- ◆ `glLoadIdentity()` **M** ← **I**
– set **M** to identity
- ◆ `glTranslatef(tx, ty, tz)` **M** ← **MT**
– translate by (*t_x*, *t_y*, *t_z*)
- ◆ `glRotatef(θ, x, y, z)` **M** ← **MR**
– rotate by angle *θ* about axis (*x*, *y*, *z*)
- ◆ `glScalef(sx, sy, sz)` **M** ← **MS**
– scale by (*s_x*, *s_y*, *s_z*)

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

23

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.

24