1 Computer Graphics, Cont.

We left off last lectures talking about...

Rotation Matrices

We can rotate something counter clockwise with

$$R = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{\text{counterclockwise}}$$

If we wanted to switch to clockwise rotation, we just need to flip the sign of the sin functions.

$$R = \underbrace{\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}}_{\text{clockwise}}$$

In computer graphics, we usually don't just want to rotate something along one axis. We'd normally want to transform, skew, rotate, etc. all at once...

Composite Transform

Composite transforms are as simple as taking multiple transformation matrices and multiplying them together.

Example

Scale a square by 3 and rotate it by $\frac{\pi}{4}$ $(45°)$.

Scaling the square is easy:

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

For rotating, we'll use a rotation matrix

$$R\left(\frac{\pi}{4}\right) \approx \begin{bmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \end{bmatrix}$$

Now we can multiply to get a composite matrix

$$SR = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \end{bmatrix} = \begin{bmatrix} 2.1 & -2.1 \\ 2.1 & 2.1 \end{bmatrix}$$

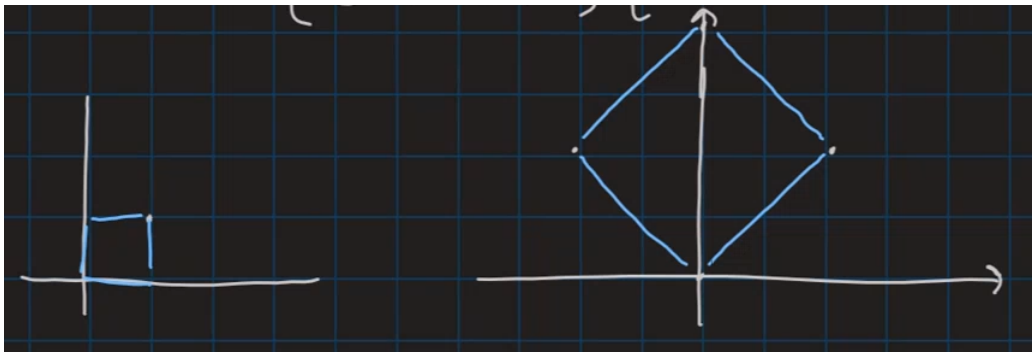
Here's our square D

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

So now we can transform

$$SRD = \begin{bmatrix} 2.1 & -2.1 \\ 2.1 & 2.1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2.1 & 0 & -2.1 \\ 0 & 2.1 & 4.2 & 2.1 \end{bmatrix}$$

Here's how that works out graphically



Composite Transform

Translation

Translation is not necessarily linear. We should ask

- Does the $\vec{0}$ vector map to the $\vec{0}$ vector in target space?
- is this transform closed under addition?

Something to think about (on the homework): is

$$\vec{x} \mapsto \vec{x} + 1$$

linear? (from what he said, I'm pretty sure it's not, but I'm not yet sure why)

Homogeneous Coordinates

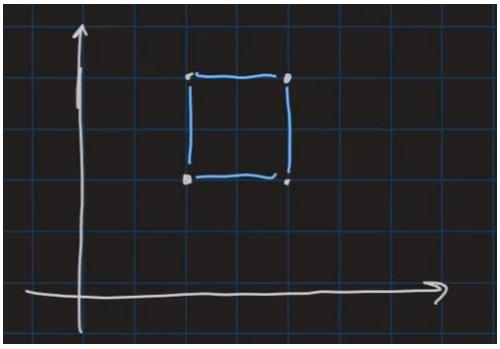
Let (x, y) be a 2D coordinate. Then $(x, y, 1)$ is a homogeneous 2D coordinate. This is equivalent to an x, y point being projected onto a plane in 3D space at $z = 1$. This allows us to use a 3×3 matrix.

$$T = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+h \\ y+k \\ 1 \end{bmatrix}$$

Example

Consider a square defined by

$$D = \begin{bmatrix} 2 & 4 & 4 & 2 \\ 2 & 2 & 4 & 4 \end{bmatrix}$$



Transform the square such that it is rotated 45° clockwise about the point (2, 2) and scaled by 0.5.

If we were to rotate the way we learned above, it would rotate with respect to the origin (0, 0). That would rotate the square towards the 4^{th} quadrant. Instead, we need to translate to the origin first, then move it back once we're done.

To rotate in place:

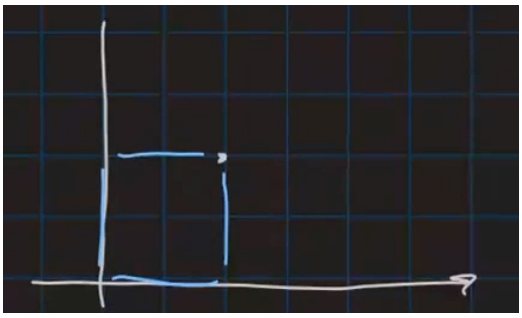
1. Translate the object to the origin
2. Apply the rotation
3. Reverse the translation

We'll start making transformation matrices. First, translate to the origin. We want the southwest corner of the square to be at the origin, so we need to move the whole think down and left by 2. We'll use the T matrix in the section above, where h and k are both -2.

$$T_1 = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

So now

$$T_1 D = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 4 & 2 \\ 2 & 2 & 4 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \end{bmatrix}$$



T_1 translated square

Now we can create our reverse translation matrix T_2 to move it back later.

$$T_2 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Next, we can define our rotation and scaling matrix. These are pretty easy, we've done them before.

$$R_c\left(\frac{\pi}{4}\right) \approx \begin{bmatrix} 0.7 & 0.7 \\ -0.7 & 0.7 \end{bmatrix}$$

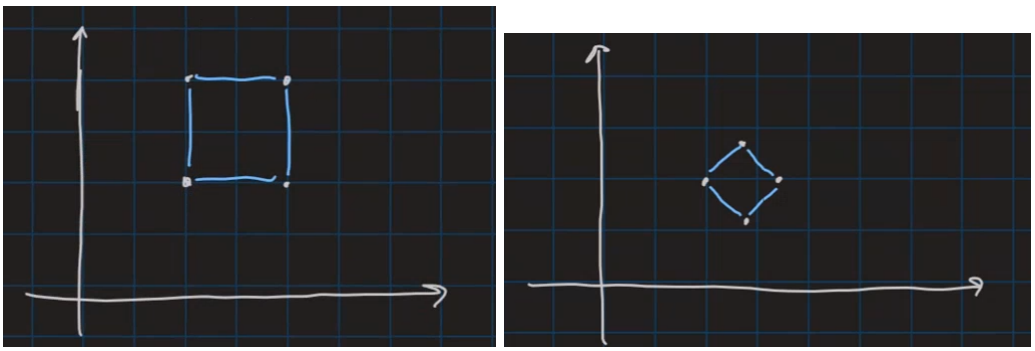
$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We need to be careful about the order we apply these in. We need to translate to the origin (T_1), apply our rotation and scaling (R_c and S), then translate back to (2, 2) (T_2). We do this from **right to left**:

$$T_2RST_1 = M = \begin{bmatrix} 0.35 & 0.35 & 0.6 \\ -0.35 & 0.35 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Now that we have our completed transformation matrix M, we can multiply by D to get the result.

$$MD = \begin{bmatrix} 2 & 2.7 & 3.4 & 2.7 \\ 2 & 1.3 & 2 & 3.7 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$



Square transformation

Homogeneous 3D Coordinates

How do these transforms change when extended to 3D?

Generally, (X, Y, Z, W) are homogeneous coordinates for (x, y, z) if $W \neq 0$ and

$$x = \frac{X}{W} \qquad y = \frac{Y}{W} \qquad z = \frac{Z}{W}$$

Scaling and translation matrices are pretty simple in 3D. They will now be 4×4:

$$S = \underbrace{\begin{bmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{scaling}} \qquad T = \underbrace{\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

Rotation matrices change more, but they're not too bad. Instead of rotating clockwise or counterclockwise, we need to rotate with respect to a certain axis. (note: this omits the last row and column of zeros to make it a homogeneous coordinate matrix. You should add a row and column of zeroes with a 1 on the corner when you actually use this.)

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

Note where the 0 and 1s are. In R_x, these would correspond with the x values.

2 Projective Geometry

Homogeneous coordinates are to projective geometry as Cartesian coordinates are to Euclidean geometry.

Perspective Projections

How do we simulate perspective?

We ran out of time, so we'll pick this up in L22 (March 26, 2021).