2D transformation matrices for translation, scaling, and rotation

14 Mar 2020

(A demo application is available on Github (https://github.com/ronnieholm/GraphicsEngine).)

This post develops the foundational matrices for translation, scaling, and rotation of control points. In computer graphics, these matrices provide an efficient and uniform way of manipulating points.

Orthogonal coordinate system

We start by formalizing a traditional 2D coordinate system. It may be defined in terms of (1) orthogonal vectors $\vec{u}$ and $\vec{v}$ for the x and y axis and (2) origin O for the point of origin. Origin is the point of intersection between $\vec{u}$ and $\vec{v}$ and is commonly $(0,0)$. The difference (https://youtu.be/01YSK5gIEYQ?list=PL_w_qWAQZtAZhtzPI5pkAtcUVgmzdAP8g&t=1469) between point and vector is best illustrated by a few examples: $Q = P + \vec{v}$ and $\vec{v} = Q - P$.

A point P may be defined in terms of the vectors $\vec{u}$ and $\vec{v}$. To arrive at P we start at origin O, then move some distance along $\vec{u}$ followed by a move some distance along $\vec{v}$:

$$P = x\vec{u} + y\vec{v} + O$$
$$P = 0.5\vec{u} + 0.25\vec{v} + O$$
$$P = 0.5\begin{bmatrix}1\\0\end{bmatrix} + 0.25\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}0.5\\0.25\end{bmatrix}$$

Convention dictates that $\vec{u}$ and $\vec{v}$ are perpendicular unit vectors, that O is the zero vector, and that P is commonly written $(x,y) = (0.5, 0.25)$ with x and y being the coordinates of P. Going forward, we'll also write points in matrix form:

$$P = \begin{bmatrix} x \; y \; 1 \end{bmatrix} \begin{bmatrix}\vec{u}\\\vec{v}\\O\end{bmatrix} = x\vec{u} + y\vec{v} + O$$

For brevity, the 3x1 matrix is mostly left out, but it's still there. P in matrix form is then $\begin{bmatrix} x \; y \; 1 \end{bmatrix}$ with 1 added to carry over origin O. Applying translation, scaling, and rotation, O must remain unaffected.

Translation

Translating a point by a in x direction and by b in y direction may be expressed by the following relationship between source and destination vectors:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x+a & y+b & 1 \end{bmatrix}$$

Transforming any left matrix to a right matrix is akin to applying a postfix operator to the left side. With respect to translation, the $T_{a,b}$ operator is matrix multiplication of the left side with a special translation matrix parameterized by a and b. Expanding the multiplication allows us to verify the correctness of $T_{a,b}$:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{bmatrix} = \begin{bmatrix} 1x+0y+1a & 0x+1y+1b & 0x+0y+1 \end{bmatrix} = \begin{bmatrix} x+a & y+b & 1 \end{bmatrix}$$

Translation, scaling, and rotation matrices are characterized by their sparseness and that the last column is always the same to carry over origin O.

Scaling

Scaling a point by a in x direction and b in y direction happens with respect to origin O. The relationship between source and destination vectors is

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \rightarrow \begin{bmatrix} ax & by & 1 \end{bmatrix}$$

To satisfy this relationship requires a scaling matrix $S_{a,b}$ as follows:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \rightarrow \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} ax & by & 1 \end{bmatrix}$$

Scaling works as intended when for a set of points their centroid $C = O$. A unit square with control points $\{(-1,1),(1,1),(1,-1)\}$ satisfies this condition. Applying $S_{2,2}$ to each point has the effect of scaling from a 2x2 to a 4x4 square.

On the other hand, a unit square with control points $\{(1,3),(3,3),(3,1)\}$ has a centroid $C = (2,2)$. Applying $S_{2,2}$ to each point has the effect of scaling from a 2x2 to a 4x4 square *and* move the centroid C. To prevent this movement, each point must be first translated such that $C = O$, then scaled, and finally translated back:

$$T_{-a,-b}S_{c,d}T_{a,b}$$

The centroid C of a set of points, assuming each has equal weight, is defined as

$$(a = \overline{x}, b = \overline{y}) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i\right)$$

Rotation

Rotating a point by θ around origin implies that radius from origin to (x, y) and radius from origin to (x', y') remain unaffected:

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

With radius r and angle ϕ between x axis and point, the left side becomes

$$\begin{aligned} x &= r\cos\phi \\ y &= r\sin\phi \end{aligned}$$

Right side is then x and y with an additional rotation θ added:

$$\begin{aligned} x' &= r\cos(\phi + \theta) \\ x' &= r\cos\phi\cos\theta - r\sin\phi\sin\theta \; \{\text{apply identity sum: } \cos(u+v) = \cos u\cos v - \sin u\sin v\} \\ x' &= x\cos\phi - y\sin\theta \; \{\text{substitute in definitions of } x \text{ and } y\} \end{aligned}$$

$$\begin{aligned} y' &= r\sin(\phi + \theta) \\ y' &= r\sin\phi\cos\theta + r\cos\phi\sin\theta \; \{\text{apply identity sum: } \sin(u+v) = \sin u\cos v + \cos u\sin v\} \\ y' &= y\cos\theta + x\sin\theta \; \{\text{substitute in definitions of } x \text{ and } y\} \end{aligned}$$

In matrix form, the rotation matrix R_θ becomes

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

As with scaling, rotation must happen around origin, possibly requiring two translations:

$$T_{-a,-b}R_\theta T_{a,b}$$

If both scaling and rotation is needed, the translation back and forth need happen only once.

Summary

The translation, scaling, and rotation matrices are collectively referred to as the elementary matrices in computer graphics. Rather than apply transformations to a point one at a time, transformation matrices may be combined into a single matrix M for reuse across points:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} M \begin{bmatrix} \vec{u} \\ \vec{v} \\ O \end{bmatrix} = \begin{bmatrix} x' & y' & 1 \end{bmatrix}$$

Refer to Lecture 3: Moving Objects in Space (https://www.youtube.com/watch?v=wArGifkRD2A) of this lecture series (https://www.youtube.com/watch?v=01YSK5gIEYQ&list=PL_w_qWAQZtAZhtzPI5pkAtcUVgmzdAP8g&index=1) for a run-down of each transformation. Around 31m30s 3D versions of matrices are introduced.