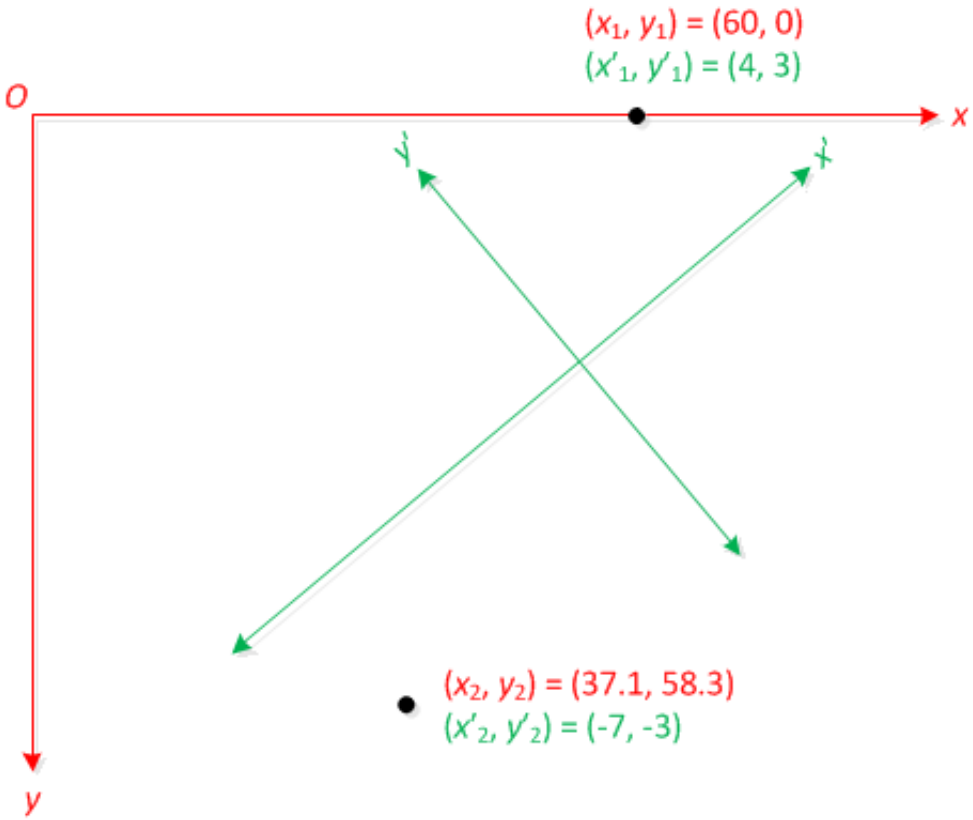
How to map points between 2D coordinate systems

This topic describes how to map points from one 2D coordinate system to another. This can be useful for graphics systems, such as canvas, that do not provide a current transformation matrix (CTM).

An affine transformation is a linear (or first-order) transformation that can relate two 2D coordinate systems, typically through a composition of rotations, translations, dilations, and shears. The following is a possible example:



Assuming a typical screen coordinate system (shown in red), affine transformations are expressed algebraically as follows:

$$x' = ax + by + c$$
$$y' = bx - ay + d$$

In other words, if we can determine *a*, *b*, *c*, and *d*, we can calculate the point (*x'*, *y'*) from the point (*x*, *y*) and vice versa.

Notice that this system of linear equations has four unknowns. Therefore, we need four equations to determine *a*, *b*, *c*, and *d*:

$$
\begin{aligned}
x'_1 &= +ax_1 + by_1 + c \\
y'_1 &= -ay_1 + bx_1 + d \\
x'_2 &= +ax_2 + by_2 + c \\
y'_2 &= -ay_2 + bx_2 + d
\end{aligned}
$$

In matrix notation, this linear system can be written as:

$$
\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ -y_1 & x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ -y_2 & x_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \Longrightarrow \mathbf{u} = \mathbf{M}\mathbf{v}
$$

To determine *a* through *d*, we invert**M** and solve as follows:

$$
\mathbf{u} = \mathbf{M}\mathbf{v} \Longrightarrow \mathbf{v} = \mathbf{M}^{-1}\mathbf{u}
$$

Note Although most modern calculators, as well as a number of mathematical websites (such as WolframAlpha), can easily invert matrices, the symbolic inverse of **M** is as follows:

$$
\begin{pmatrix}
\frac{x_1 - x_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{y_2 - y_1}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_2 - x_1}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{y_1 - y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} \\
\frac{y_1 - y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_1 - x_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{y_2 - y_1}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_2 - x_1}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} \\
\frac{x_2^2 - x_1x_2 + y_2^2 - y_1y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_2y_1 - x_1y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_1^2 - x_2x_1 + y_1^2 - y_1y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_1y_2 - x_2y_1}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} \\
\frac{x_1y_2 - x_2y_1}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_2^2 - x_1x_2 + y_2^2 - y_1y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_2y_1 - x_1y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2} & \frac{x_1^2 - x_2x_1 + y_1^2 - y_1y_2}{x_1^2 - 2x_2x_1 + x_2^2 + y_1^2 + y_2^2 - 2y_1y_2}
\end{pmatrix}
$$

Now by examining **u** and **M**, we see that we only need two known coincident points from each coordinate system:

$$
\mathbf{u} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{bmatrix}
$$

$$
\mathbf{M} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ -y_1 & x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ -y_2 & x_2 & 0 & 1 \end{bmatrix}
$$

For example, from the previous diagram, we see that:

$$(x_1, y_1) = (60, 0)$$
$$(x'_1, y'_1) = (4, 3)$$
$$(x_2, y_2) = (37.1, 58.3)$$
$$(x'_2, y'_2) = (-7, -3)$$

Plugging these values into **u** and **M** yields:

$$\begin{bmatrix} 4 \\ 3 \\ -7 \\ -3 \end{bmatrix} = \begin{bmatrix} 60 & 0 & 1 & 0 \\ -0 & 60 & 0 & 1 \\ 37.1 & 58.3 & 1 & 0 \\ -58.3 & 37.1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Taking the inverse of **M** (approximate values shown) and multiplying by **u** produces:

$$\begin{bmatrix} 0.006 & 0.015 & -0.006 & -0.015 \\ -0.015 & 0.006 & 0.015 & -0.006 \\ 0.650 & -0.892 & 0.350 & 0.892 \\ 0.892 & 0.650 & -0.892 & 0.350 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ -7 \\ -3 \end{bmatrix} \approx \begin{bmatrix} 0.1534 \\ -0.1284 \\ -5.202 \\ 10.706 \end{bmatrix}$$

That is:

$$a \approx 0.1534$$
$$b \approx -0.1284$$
$$c \approx -5.202$$
$$d \approx 10.706$$

So the transformation equations become:

$$x' = ax + by + c$$
$$y' = bx - ay + d$$

$$x' \approx (0.1534)x + (-0.1284)y + (-5.202)$$
$$y' \approx (-0.1284)x - (0.1534)y + (10.706)$$

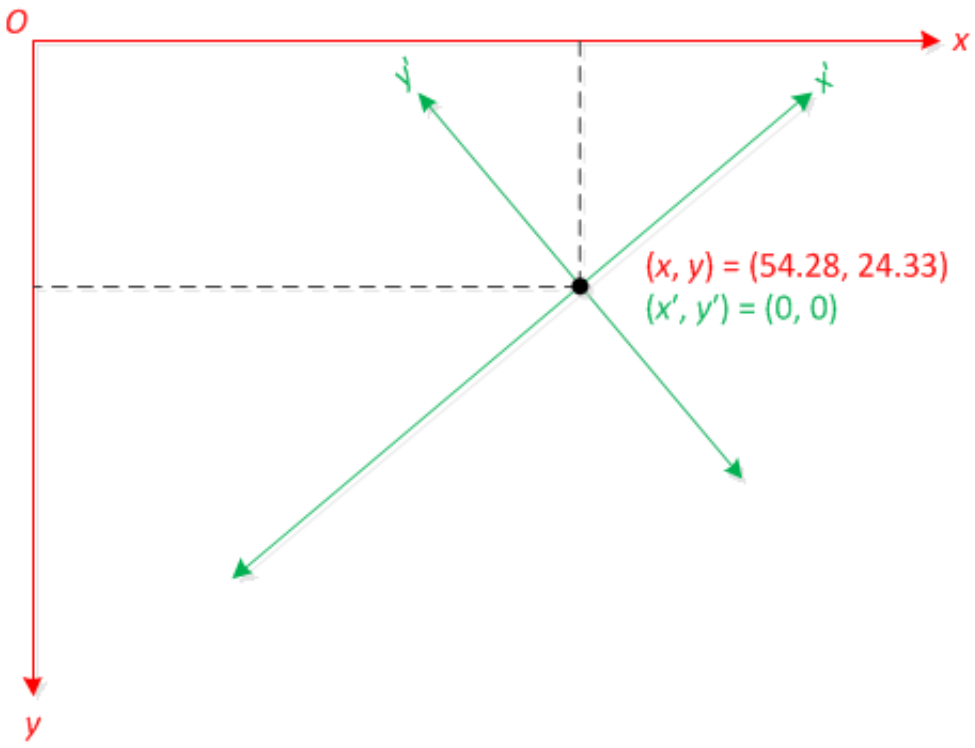
$$x' \approx 0.1534x - 0.1284y - 5.202$$
$$y' \approx -0.1284x - 0.1534y + 10.706$$

For example, (*x*, *y*) = (37.1, 58.3) maps to the point (*x*', *y*') = (-7, -3) as follows:

$$x' \approx 0.1534(37.1) - 0.1284(58.3) - 5.202 \approx -7$$
$$y' \approx -0.1284(37.1) - 0.1534(58.3) + 10.706 \approx -3$$

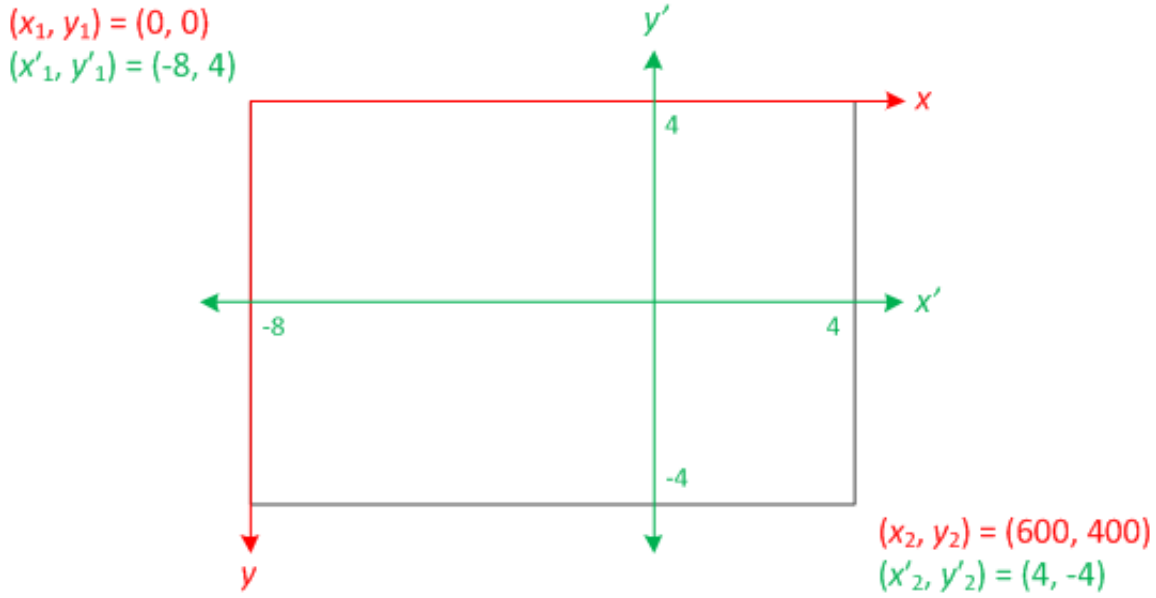
And the point (*x*, *y*) = (54.28, 24.33) maps to the *x'y'*-origin, as expected:

$$x' \approx 0.1534(54.28) - 0.1284(24.33) - 5.202 \approx 0$$
$$y' \approx -0.1284(54.28) - 0.1534(24.33) + 10.706 \approx 0$$



This transformation technique can be useful for graphics systems that do not provide a current transformation matrix (CTM). Unlike SVG, canvas (as of this writing) does not provide a CTM (that is, there is no `canvas.context.getTransform` method). Therefore, the technique we've discussed here can be used in a number of scenarios instead. For example, consider a 600 pixel by 400 pixel canvas (in red) with the following implied coordinate

system (in green):



For this relatively simple case, the required transformation equations can be derived without using linear algebra, but we do so to exemplify the above procedure. From this diagram, we see that:

$$
\begin{aligned}
(x_1, y_1) &= (0, 0) \\
(x'_1, y'_1) &= (-8, 4) \\
(x_2, y_2) &= (600, 400) \\
(x'_2, y'_2) &= (4, -4)
\end{aligned}
$$

Thus,

$$
\mathbf{M} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ -y_1 & x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ -y_2 & x_2 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 600 & 400 & 1 & 0 \\ -400 & 600 & 0 & 1 \end{bmatrix}
$$

And,

$$\mathbf{M}^{-1} \approx \begin{bmatrix} -0.001154 & 0.00769 & 0.001154 & -0.000769 \\ -0.000769 & -0.001154 & 0.000769 & 0.001154 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Therefore,

$$\mathbf{v} = \mathbf{M}^{-1}\mathbf{u}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \approx \begin{bmatrix} -0.001154 & 0.00769 & 0.001154 & -0.000769 \\ -0.000769 & -0.001154 & 0.000769 & 0.001154 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -8 \\ 4 \\ 4 \\ -4 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0 \\ -8 \\ 4 \end{bmatrix}$$

So the transformation equations become:

$$x' = ax + by + c = 0.02x + 0y - 8 = 0.02x - 8$$
$$y' = bx - ay + d = 0x - 0.02y + 4 = 4 - 0.02y$$

That is, (0, 0) maps to (-8, 4) as follows:

$$x' = (0.02)(0) - 8 = -8$$
$$y' = 4 - (0.02)(0) = 4$$

And (600, 400) maps to (4, -4) in the same way:

$$x' = (0.02)(600) - 8 = 12 - 8 = 12 - 8 = 4$$
$$y' = 4 - 0.02(400) = 4 - 8 = -4$$

As a final validation, we confirm that (400, 200) maps to the origin of the green coordinate system:

$$x' = (0.02)(400) - 8 = 8 - 8 = 0$$
$$y' = 4 - 0.02(200) = 4 - 4 = 0$$

When the transformation equation coefficients are known, you can calculate (*x*, *y*) from (*x*', *y*') as follows:

$$x = \frac{ax' + by' - bd - ac}{a^2 + b^2}$$

$$y = \frac{bx' - ay' - bc + ad}{a^2 + b^2}$$

To conclude, when a graphical system (such as canvas) does not provide a CTM, the above technique can be used to generate useful transformation equations given two coincident points from both 2D coordinate systems.

Related topics

How To Create 3D Graphics Using Canvas
HTML5 Canvas
SVG Coordinate Transformations