Mathematics 308—January 11, 1996

Drawing in PostScript

In these notes, I will explain more about the mechanism of drawing in PostScript. I begin with a few short remarks which will be largely review. In the second section I will introduce the transformation between **user coordinates** and **page coordinates**. This part of PostScript is essentially mathematical, and also rather elegant.

The drawing environment

When you start a path, you put the command `newpath` into your program. Then you set a current point with the `moveto` command and add a number of commands adding pieces to the path you want to draw, such as lines or arcs of circles. Finally, you actually place the path on the current page with one of the three commands `stroke`, `fill` or `clip`.

There are several different parameters involved in how the path is drawn. The most important are **colour** (for us, usually just a shade of grey), the **width of lines**, the **transformation** from the user coordinates to page coordinates, and the **clipping path**.

We already know about most of these. A few extra remarks:

Colour You always start off with colour black. If you are a Canadian nationalist you can put a line

```
/setgrey setgray def
```

and spell things the way you like. If you want to put the current shade on the stack you write

```
currentgray
```

so that if the current colour is either black or white the sequence

```
1 currentgray sub setgray
```

toggles it to the opposite shade.

Clipping The clipping path sets a path outside which nothing is actually drawn. It gives you a kind of window which you will see your drawings through. It is built like nay other path, but replaces `stroke` or `fill` by `clip`. These are all of the basic drawing operators.

Refinements There are a number of other things you can specify—dash pattern, the way paths are closed up, the way corners are rounded, etc.

Coordinates The most important thing to be dealt with is the transformation from the plane where you draw things to the plane where they are rendered.

When you start up, one unit of drawing is equal to 1/72 of an inch. This is for historical reasons—this unit is almost the same as a printer's point, the unit used since 1500 to specify the size of letters on a printed page. Also, the origin of your coordinate system gets mapped onto the lower left of the page, and the x and y axes are perpendicular, going along the sides of the page.

You can change these by using commands `scale`, which changes the units in the x and y axes; `translate`, which moves the origin; and `rotate`, which rotates the axes. You can combine these in any order to get more complicated changes, and you can also you can change the way in which your paths are rendered in a more general away.

Some mathematics is necessary in order to understand how this works.

Affine transformations

Suppose given two planes, each with rectangular coordinates. If A is a 2×2 matrix and (τ_x, τ_y) a vector in the second plane, then the map taking

$$(x \quad y) \mapsto (x \quad y) A + (\tau_x \quad \tau_y)$$

is called an **affine transformation**. In short, an affine transformation is a linear transformation followed by a vector translation. Note that I am considering vectors as **row** vectors rather than columns. This means matrices are applied on the right to them. I am doing this because it is a convention of PostScript.

Explicitly, if

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

then this map takes

$$(x, y) \mapsto (u, v) = (ax + cy, bx + dy) + (\tau_x, \tau_y) = (ax + cy + \tau_x, bx + dy + \tau_y) \ .$$

Lemma. *If A is an invertible matrix, then this transformation is invertible.*

Because we can solve

$$\begin{aligned} u &= ax + cy + \tau_x \\ v &= bx + dy + \tau_y \end{aligned}$$

or

$$\begin{aligned} ax + cy &= u - \tau_x \\ bx + dy &= v - \tau_y \end{aligned}$$

for (x, y) if we are given (u, v). In terms of matrices if

$$(u \quad v) = (x \quad y) A + (\tau_x \quad \tau_y)$$

then

$$(x \quad y) = (u \quad v) A^{-1} - (\tau_x \quad \tau_y) A^{-1}$$

so that the inverse is also an affine transformation.

Lemma. *An affine transformation takes lines to lines.*

Let ℓ be a parametrized line of the form

$$(x_0 \quad y_0) + t (p \quad q)$$

that is to say going through $(x_0 \quad y_0)$, in the direction $(p \quad q)$. Then $T\ell$ goes through $(x_0 \quad y_0) A + (\tau_x \quad \tau_y)$ in the direction $(p \quad q) A$.

As we shall see eventually, affine transformations can be characterized in completely geometric terms as maps from one plane to another which take lines to lines. But this is not a simple fact.

The point for us is that the map from the plane where the user draws to the drawing area on the page or the screen is always an affine transformation. It is represented in PostScript by an array of six elements `[a b c d tx ty]`, representing the matrix A and translation vector τ. It is called the **Current Transform Matrix** or **CTM** for short. It is modified by the commands `rotate`, `translate`, and `scale` as well as others.

($\bullet$) The effect of the sequence

```
sx sy translate
```

is to change the CTM to `[a b c d tx+sx ty+sy]`.

(•) The effect of

`sx sy scale`

is to change the CTM by multiplying its linear component A (the 2×2 matrix) by a diagonal matrix:

$$A \mapsto \begin{pmatrix} sx & 0 \\ 0 & sy \end{pmatrix} A$$

(•) The effect of `rotate` is

$$A \mapsto \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} A$$

To understand this, keep in mind that the matrix of a rotation by θ is

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

(see the discussion later). If this is different from what you are used to it is because here matrices act on the right.

Any affine transformation can be represented as a linear transformation in three dimensions. We do this by identifying the (x, y) plane with the plane $z = 1$ in three dimensions. The matrix

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ \tau_x & \tau_y & 1 \end{pmatrix}$$

takes this plane to itself, and has the same effect on x and y coordinates as the corresponding affine transformation does. To see this, calculate

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ \tau_x & \tau_y & 1 \end{pmatrix}$$

This representation has the virtue that the composition of transformations corresponds to the multiplication of matrices. There is a PostScript command which does exactly that for you.

(•) The sequence

`[a b c d tx ty] concat]`

replaces the CTM by the matrix product

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ \tau_x & \tau_y & 1 \end{pmatrix} \mathrm{CTM}$$

if we think of the CTM as a 3×3 matrix.

Drawing circles and ellipses

I define an ellipse to be what you get when you scale a unit circle in perpendicular directions, along axes through its centre. Accordng to this definition, a circle is to be considered as a special kind of ellipse.

The data specifying an ellipse comprise the original circle, or equivalently the centre of the circle, together with the scaling factors and axes. Unless the ellipse is a circle, these data are uniquely determined by the ellipse.

If we start with the unit circle; scale it; rotate it, and then translate it, we can obtain any given ellipse.

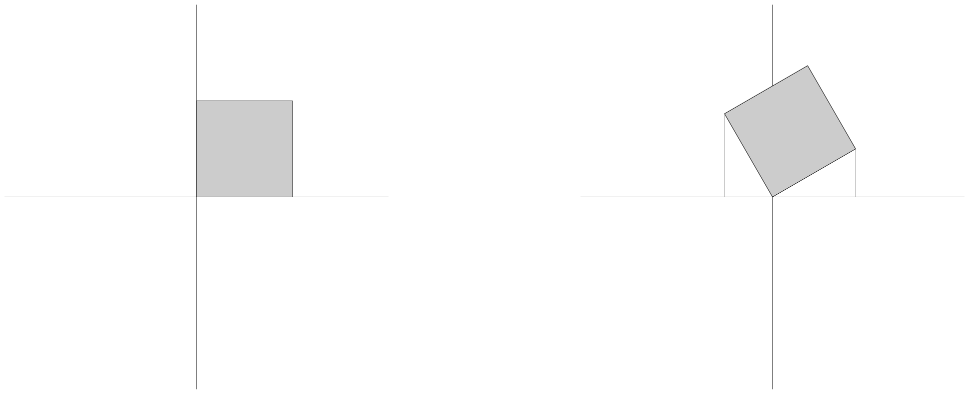
Proposition. *Every affine transformation takes ellipses into ellipses.*

This follows from these two next results.

Proposition. *Any invertible matrix is a product $K_1 A\, K_2$, where the K_i are rotations and A is a diagonal matrix.*

The proof will tell you explicitly how to do this. It will take a while.

Rotation through an angle θ is a linear transformation which has this effect:



The matrix of a linear transformation, using the convention that a matrix acts on row vectors, is obtained from the linear transformation by setting the rows of the matrix equal to the vectors v_1 and v_2 the transformation takes $(1, 0)$ and $(0, 1)$ to. From the picture it is clear how to compute the matrix of a rotation through θ.

An **orthogonal matrix** is one whose rows are of unit length and perpendicular to each other. Equivalently, K is orthogonal if

$$K\,{}^tK = I = {}^tK\,K$$

or yet again if the inverse of $K^{-1} = {}^tK$. Here ${}^tK$ is the **transpose** of K.

The picture or a calculation shows that rotation matrices are orthogonal. Calculation shows it has determinant 1.

The converse is also true.

Lemma. *A matrix is a rotation matrix if and only if it is an orthogonal matrix with positive determinant.*

Suppose K to be an orthogonal matrix. This means that

$$K\,{}^tK = I$$

This means no more and no less than that the rows of K have unit length and are perpendicular to each other. If v_1 is the first row, we can write it as $(\cos\theta, \sin\theta)$ for some θ, which is possible for any vector of unit length. The second row v_2 must be v_1 rotated $\pm 90°$. But the condition on the determinant implies that the sign must be positive.

Recall that a symmetric matrix S can be diagonalized as

$$S = K^{-1} D K$$

where D is a diagonal matrix whose entries are the eigenvalues of S, and that the rows of K are the eigenvectors, which can be chosen to have length 1 and be perpendicular to each other. Multiplying an eigenvector by ± 1 we may also assume K to have determinant 1. In other words K can be chosen to be a rotation.

The matrix S is said to be positive definite if the eigenvalues are positive, and this is true for a 2×2 matrix just when (a) the diagonal entries of S are positive and (b) the determinant of S is positive.

Now we are ready to begin finding the K_i and A.

Let M be an invertible matrix. If we could write

$$M = K_1 A K_2$$

where the K_i are rotations, then ${}^t A = A$ so

$$M\,{}^t M = K_1 A^2\, {}^t K_1$$

since $K_2\,{}^t K_2 = I$. Since K_1 is orthogonal, this means that its rows are the eigenvectors of $M\,{}^t M$ and that the squares of the entries of A are its eigenvalues.

Conversely, if we writre out

$$M\,{}^t M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} a^2 + b^2 & ac + bd \\ ac + bd & c^2 + d^2 \end{pmatrix}$$

we see that the matrix $M\,{}^t M$ is symmetric and positive definite, so that its eigenvalues are positive real numbers, and (Step 1) we can write

$$M\,{}^t M = K_1 D\, {}^t K_1$$

where the rows of ${}^t K_1$ or columns of K_1 are the eigenvectors of $M\,{}^t M$. We can write $D = A^2$ with A diagonal. There is a choice of sign; if $\det M < 0$, choose one sign along the diagonal of A negative. Thus $\det A = \det M$. Then (Step 2) set

$$K_2 = A^{-1} K_1^{-1} M = A^{-1}\, {}^t K_1 M$$

I claim that K_2 is a rotation matrix.

We apply the earlier result. The determinant of K_2 is positive because determinants multiply and

$$\det K_2 = \det A \det K_1^{-1} \det M = \det A \det M$$

and because of the choice of signs of A. Furthermore the equation

$$M\,{}^t M = K_1 A^2\, {}^t K_1$$

can be rewritten as

$$A^{-1} K_1^{-1} M\, {}^t M K_1 A^{-1} = I$$

or

$$K_2{}^t K_2 = I$$

Note that both Steps 1 and 2 can be carried out explicitly.

Example. Suppose we are given the matrix

$$M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Then

$$M^t M = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

Its characteristic polynomial is

$$\lambda^2 - 3\lambda + 1 = 0$$

so its eigenvalues are $3/2 \pm \sqrt{5}/2 = \lambda_1,\ \lambda_2$; numerically $\lambda_1 = 2.618033989$, $\lambda_2 = 0.381966011$. To find the eigenvectors for λ we solve

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} = \lambda \begin{pmatrix} x & y \end{pmatrix}$$

and get the eigenvector for λ to be

$$(\lambda - 1, 1)$$

or the normalized pair of eigenvectors

$$(0.850650808, 0.525731112), \quad (-0.525731112, 0.850650808)$$

Therefore

$$A = \begin{pmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \end{pmatrix} = \begin{pmatrix} 1.618033989 & 0 \\ 0 & 0.618033989 \end{pmatrix}$$

and

$$K_1 = \begin{pmatrix} 0.850650808 & -0.525731112 \\ 0.525731112 & 0.850650808 \end{pmatrix}$$

From this we get

$$\begin{aligned} K_2 &= \begin{pmatrix} 1/1.618033989 & 0 \\ 0 & 1/0.618033989 \end{pmatrix} \begin{pmatrix} 0.850650808 & 0.525731112 \\ -0.525731112 & 0.850650808 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1/1.618033989 & 0 \\ 0 & 1/0.618033989 \end{pmatrix} \begin{pmatrix} 0.850650808 & 1.376381920 \\ -0.525731112 & 0.324919696 \end{pmatrix} \\ &= \begin{pmatrix} 0.525731112 & 0.850650808 \\ -0.850650808 & 0.525731112 \end{pmatrix} \end{aligned}$$

As a consequence:

Proposition. *Any affine transformation can be represented as a composition of rotations, scalings, and translations.*

A consequence of this in turn is this:

Proposition. *A linear transformation changes areas by a factor equal to its determinant. It preserves orientations if and only if its determinant is positive.*

This is because any linear transformation can be factored as above, and determinants and volume change factors multiply under composition.

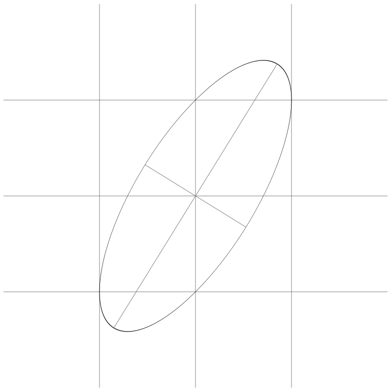
Another curious consequence is that we don't need to apply the command `concat`. The Proposition asserts that a combination of `rotate`, `scale`, and `translate` can do any job `concat` does. But the example shows that `concat` is convenient.

If C is a circle and we apply a rotation, a scale, and a rotation to it we certainly get an ellipse. This proves that affine transformations take circles to ellipses. If E is an ellipse, we can find an affine transformation which takes the unit circle C to it. Say $T_1C = E$. If T is any other affine transformation then $TE = TT_1C$ which is an ellipse. So affine transformations take ellipses to ellipses.

For example since

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 0.850650808 & -0.525731112 \\ 0.525731112 & 0.850650808 \end{pmatrix} \begin{pmatrix} 1.618033989 & 0 \\ 0 & 0.618033989 \end{pmatrix} \begin{pmatrix} 0.525731112 & 0.850650808 \\ -0.850650808 & 0.525731112 \end{pmatrix}$$

it takes the unit circle to an ellipse with major axis 1.618033989, minor axis 0.618033989, and whose major axis has an angle of about 58° with respect to the x-axis.



You draw circles in PostScript with the command `arc`. The sequence

```
1 0 moveto
0 0 10 0 360 arc
stroke
```

draws a circle of radius 10 with centre $(0, 0)$. The way to draw an ellipse is to apply `gsave`, `concat` with a suitable matrix (or a series of more elementary operations in order to change the CTM), draw a unit circle with the new CTM, and then apply `grestore`.

Exercise. *But how did I draw the axes?*