Adobe Graphics Model

Steven R. Bagley

Introduction

- Today’s lecture — look at the Adobe Graphics Model
- Next few lectures look at how we can drive the AGM from Postscript, PDF and SVG
- Get in the lab next Monday and have a play…

Adobe Graphics Model

- Vector-based
- But output aimed at raster devices
  - Doesn’t make any assumption about how these devices will be implemented
- Vectors *scan-converted* to the raster form
  - Need to make decisions about which pixels are inside or outside a shape

Tend to think of them as a raw bitmap in memory but not all are…
Likely that vectors won’t fall directly on pixel boundaries -- this is particularly important when converting text which needs to be legible…

Adobe Graphics Model

- Works like bitmap drawing package
- Each operation paints onto the page
- Once operation executed it cannot be removed
- Only painted over

E.g MS Paint

Paint

- Painted figures maybe letter shapes (glyphs), general filled shapes, lines or raster images
- Paint maybe in colour, black, white or gray
- Paint may take the form of a repeating pattern or gradient
- Maybe clipped to appear within other shapes as placed on the page

Painting Model

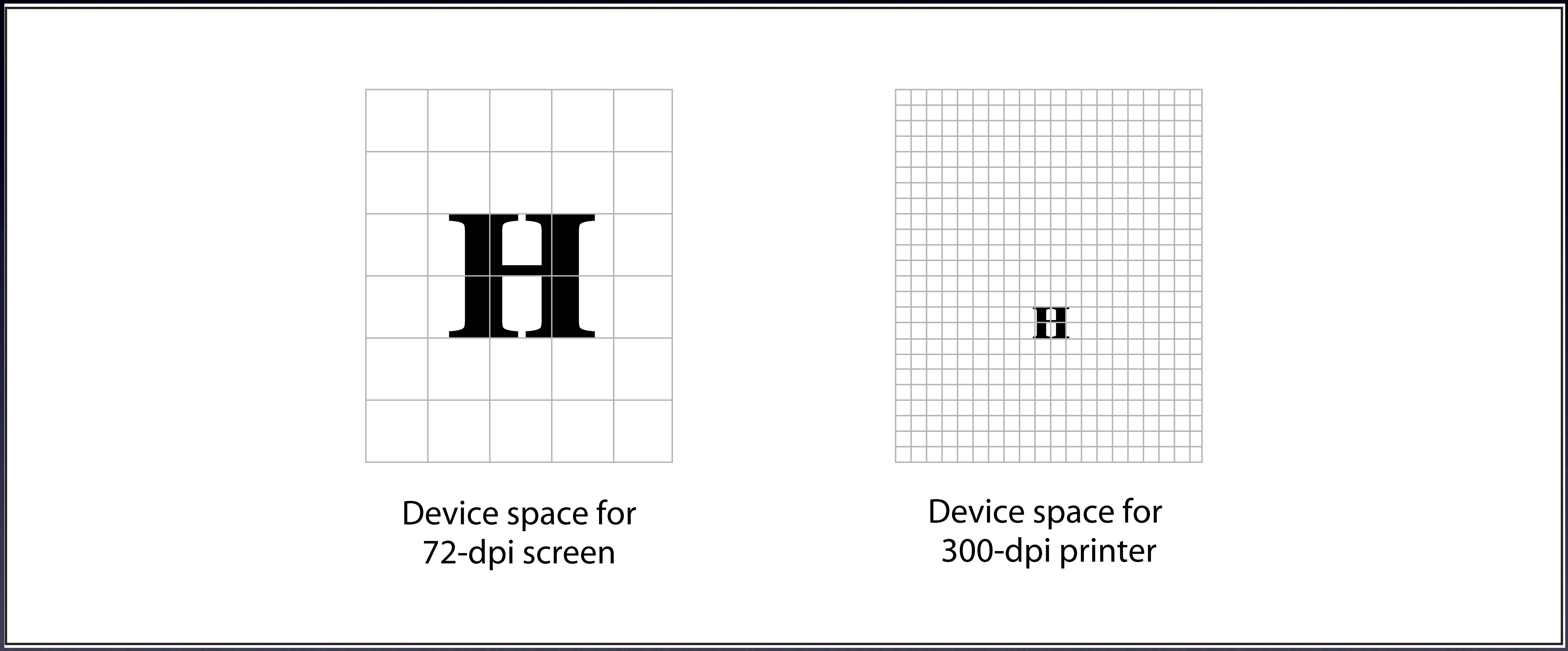
- No matter the color it is put on the page as if it were opaque
  - Although later revisions of the AGM allow for transparency
- Once page has been completely composed
- The accumulated marks are rendered and the page cleared ready for the next…

Spaces

- AGM makes use of the notion of co-ordinate *spaces*
- Main two are
  - *User Space*
  - *Device Space*
- Others include *text* space, *glyph* space, *image* space and *form* space

User Space

- User space defines a specific co-ordinate space
- Abstracts the AGM from any specific device
- Resolution, origin etc. all vary across devices
- If the AGM didn't abstract the device away then a page would look vastly different on each device…



Example of difference in apparent size of object with same dimensions on different devices

User Space

- Defines a resolution of 72dpi
- Defines a co-ordinate system
  - x positive to right
  - y positive upwards
  - Exception is SVG…
- (0,0) is usually the bottom-left corner

Roughly equivalent to the typographic dimension of a point
SVG has y positive downwards (due to being aimed at the

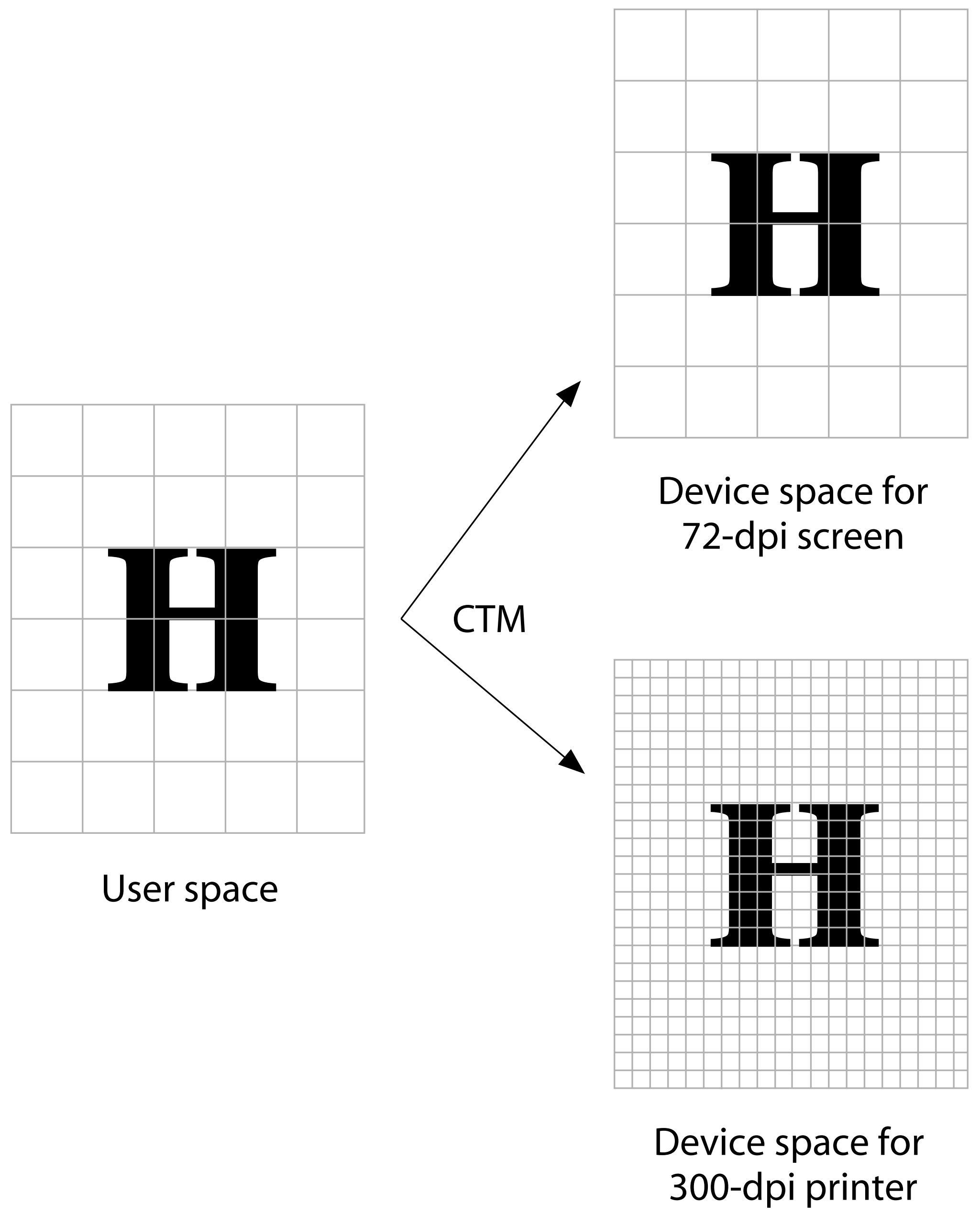
User Space

- User space is effectively unbounded
- Can happily use positive and negative co-ordinates
- You can define which rectangular section is taken as the page

Device Space

- *User Space* constant regardless of output device
- *Device space* describes how a specific device's pixels are addressed
- Varies depending how the device loads paper, addresses the screen etc.
- Necessary to transform co-ordinates from *User space* to *Device Space*

Resolution, origin, and orientation etc. all vary…


User space
CTM
Device space for 72-dpi screen
Device space for 300-dpi printer

Current Transformation Matrix

- AGM handles conversion from *User Space* to *Device Space* automatically
- Every co-ordinate is mathematically transformed to device space by multiplying it with a transformation matrix
- Referred to as the *Current Transformation Matrix*

Transformation Matrices

- Several types of geometric transformations we may need to do
  - Translate (shift) the origin to point (t_x, t_y)
  - Scale by (s_x, s_y)
  - Rotate around the origin
- All of these *affine* transformations can be captured in a single transformation matrix

Transformation Matrix

- Co-ordinate is expressed as a 3x1 matrix

```
[ x y 1 ]
```

- Multiplied by a 3x3 transformation matrix

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

- Gives a new 3x1 matrix containing the transformed co-ordinates

Transformation Matrix

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Multiply this out

Transformation Matrix

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

$$\begin{aligned} x' &= ax + cy + t_x \\ y' &= bx + dy + t_y \end{aligned}$$

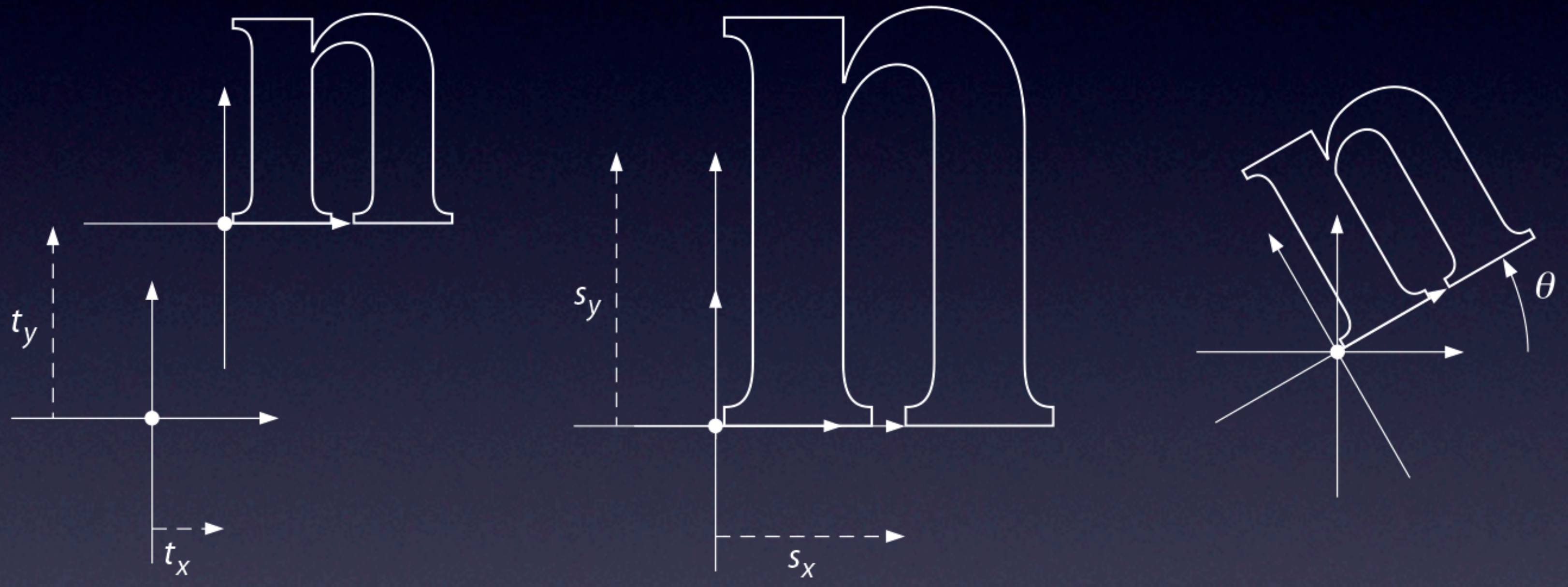
Multiply this out

Transformation Matrix

- By specifying the right values for `a`, `b`, `c`, `d`, `e`, `f`, t_x and t_y
- We can specify a translation, scale or rotation transformation

Give some examples

Transformation Matrices



Identity Translation Scaling Rotation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combining TMs

- Can combine two Transformation Matrices into a single one with the same effect
- Multiply the two Transformation Matrices together
- However order matters…
- Matrix on right-hand of multiplication happens first

Matrix mutliplication is not commutative

Scale and Translate

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scale then translate

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Translate then scale

Show multiplication out…

User to Device Space

- By combining separate affine transformations into a single matrix
- We can develop a single TM that can convert from user space to device space
- Usually a combination of translates and scales

US to DS

- Suppose we have 300dpi printer which has an origin at the top of the page going down
- Need to:
  - Scale user space co-ordinates by 300/72
  - Invert the y-axis (can do this by scaling the y-axis alone by -1)
  - Translate the origin so it’s at the top of the page

Assume page height is A4 so 595x842
Work through example
Show the three matrices, then combine them in the right order

US to DS example

$$\begin{bmatrix} \frac{300}{72} & 0 & 0 \\ 0 & \frac{300}{72} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 842 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{300}{72} & 0 & 0 \\ 0 & -\frac{300}{72} & 0 \\ 0 & 842 & 1 \end{bmatrix}$$

Start by moving the origin to the top-left (in user space)
Then scale y by -1 (now origin is the top left)
Finally can scale by 300/72 to adjust for device resolution

Current Transformation Matrix

- AGM goes further
- Allows additional transformations to be specified that can transform user space
- These are concatenated with the User Space to Device Space
- Makes it very easy to create graphical effects (e.g. rotated text)

Transform the coordinate space

Graphics State

- The AGM holds a *graphics state*
- A set of parameters that define the framework which graphics are drawn
  - e.g. current color, or line width
- Need to be set before drawing something
- Persist until its changed

Graphics State Parameters

- Current Transformation Matrix
- Position
- Path
- Clipping Path
- Colour Space and Colour
- Font
- Line width
- Line cap — shape of line ends
- Line Join
- Miter limit
- Dash pattern

Colour can be split into fill and stroke in PDF and SVG

Using AGM

- Set up Graphics State
  - Construct path then fill or stroke it
  - Or Image text etc.
- Modify Graphics State
  - Draw some more stuff…
- And repeat until