Outline Fonts

Steven R. Bagley

Previously…

- Last term, looked at how bitmap fonts worked…
- Store the bit pattern needed for each glyph
- Blit it to the screen at the correct place
- But needs a definition for each glyph at each individual point size at every resolution…

Somebody has to create them all

Outline Fonts

- Can’t scale bitmaps and achieve a good result
- Outline fonts take a different approach
- Store a mathematical description of each glyph
- Then scale it to the required size before converting it into a bitmap

Or Vector fonts…

Bitmap can then be drawn as before…

Glyph Description

- Vector description drawn to a known size
- PostScript fonts often use a 1000x1000 unit square to represent each glyph
- Although each glyph will be a unique size
- This is scaled down to be 1pt x 1pt
- Then scaled up to the required point size

Although, in practice, you'd combine the two scaling factors

Glyph Description

- The description can then be rasterized at the required resolution
- Just as a PostScript file is…
- In fact, for PostScript Type 3 fonts the glyph descriptions are just PostScript procedures

PostScript fonts

- PostScript originally supported two types of fonts
- Type 1 fonts — encrypted, support hinting
  - Originally, only Adobe could produce them
- Type 3 fonts — open

PostScript Type 3 fonts

- Represented by a PostScript dictionary
- Presented to `definefont` to inform the interpreter of its existence
- Dictionary must contain

Font Dictionaries

Key	Type	Semantics
`FontMatrix`	array	Used to transform character coordinates into user coordinates.
`FontType`	integer	Indicates the type of the font
`FontBBox`	array	Defines the bounding box of the smallest rectangle that can contain all the characters in the font
`Encoding`	array	An array of 256 names that maps character codes to character names.

All postscript fonts (both type 1 and type 3) must contain these

Type 3 Glyphs

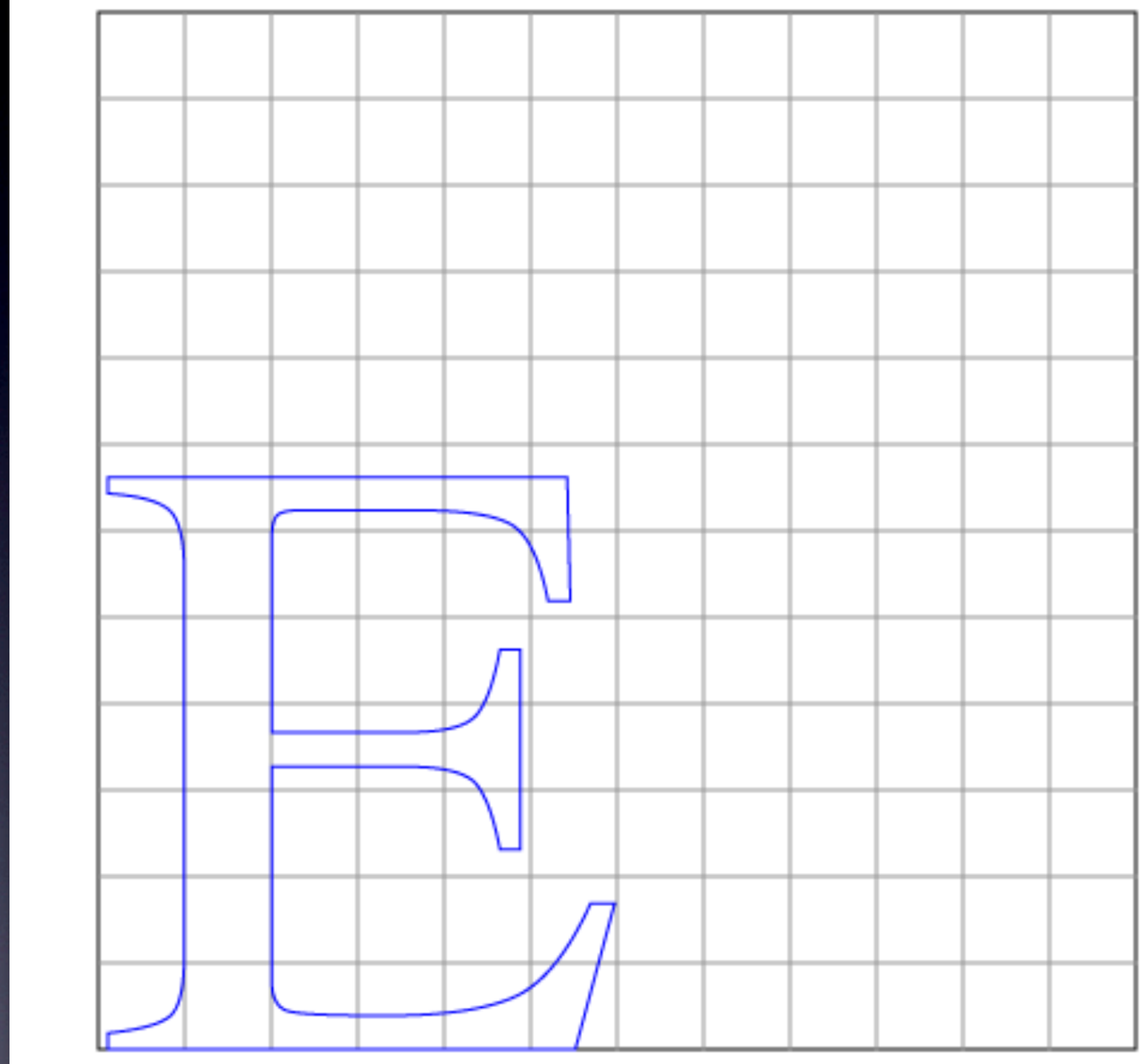
- Type 3 font dictionaries also have another entry — `BuildChar`
- A procedure that is used to draw any glyph
  - Called with the font dictionary and the character code to draw on the stack
  - CTM set up with the `FontMatrix` in place, so the procedure can just draw

Widths and Bounding Box

- Your `BuildChar` must tell the system how wide the character is
- And the glyphs bounding box
- Done using `setcachedevice` operator
- `BuildChar` can do pretty much anything to draw a character
- Except colour…

Outline Fonts

- Outline fonts have a problem
- When the text get small, and the resolution is low
- The mathematical description may not fall exactly on pixel boundaries
- In fact, in some cases whole parts of the letter may fall between pixel boundaries

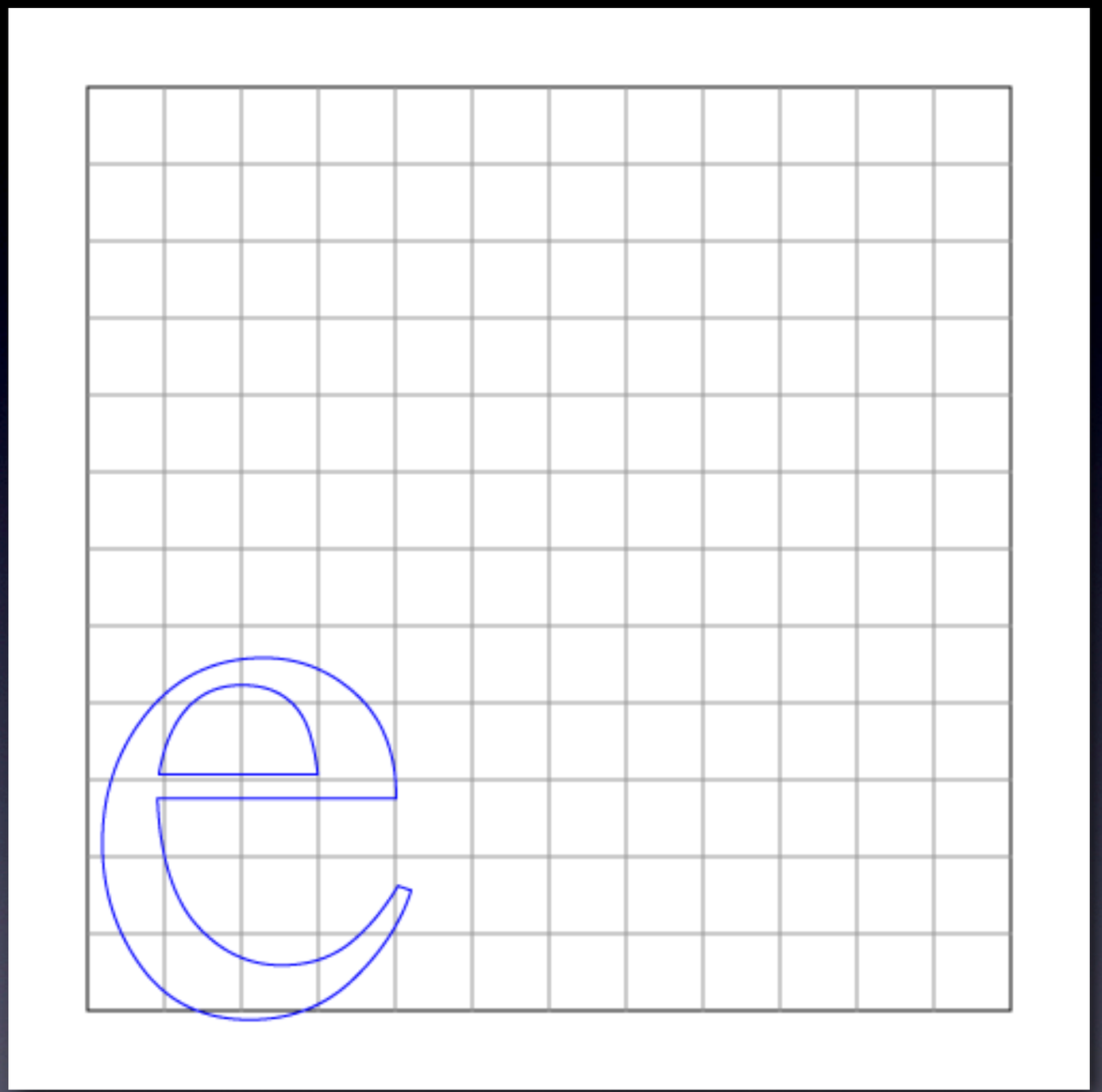


Times-Roman 10pt capital E at 72dpi
Note how the horizontal lines take up less than half a pixel…

Outline Fonts

- The result can be that the rendered decides not to set a pixel
- And so vital parts of the glyph are no longer visible…
- Can change the meaning of letters…

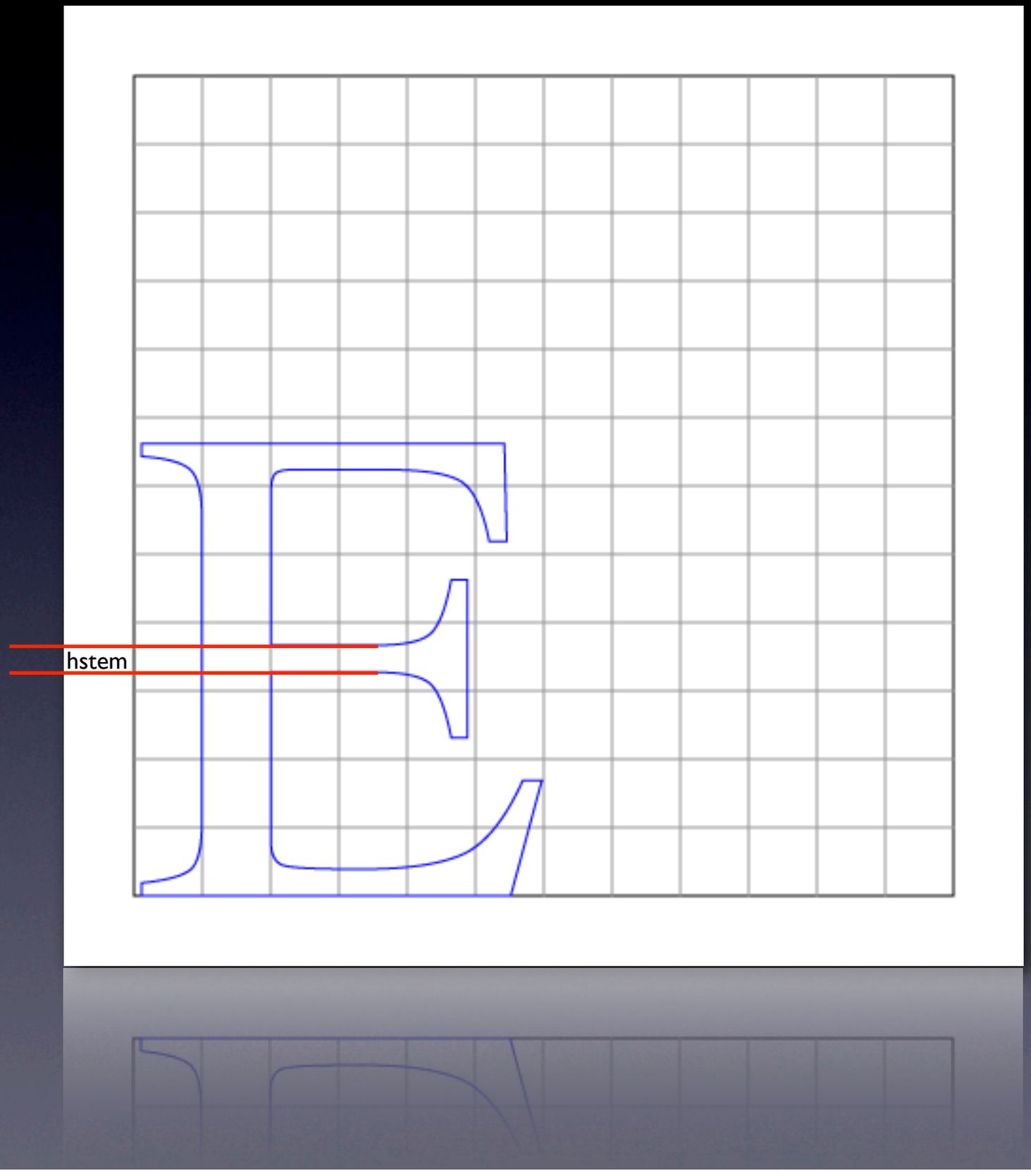
I've seen this happen…



Times-Roman 10pt capital e at 72dpi
If not careful, then middle stem will go missing and e will become c…

Hinting

- To get around this problem, Type 1 fonts allow for ‘hinting’
- These are declarative hints in the font file that put constraints on a glyph’s geometry
- E.g. the width of a stem or the location of an extremity



Times-Roman 10pt capital E at 72dpi
Note how the horizontal lines take up less than half a pixel…

Type 1 fonts

- Support hinting
- Also encrypted
- Original encryption was kept secret but was released in the early-1990s
- After the TrueType font wars…

Type 1 fonts

- Also, place some restrictions on the character definitions
- *Not* general PostScript
- Extra commands for specifying hints
- Must be drawn as filled shapes
- Uses a binary encoding for the commands