# Metafont

Natalia Slowikowska, Cedrick Giese

February 24, 2023

# Table of contents

# Table of contents

# What are TEX and Metafont?

- both developed by Donald E. Knuth (1978 and 1979 respectively)
- TEX: typesetting system
  - aims to create visually appealing books
  - used for technical and mathematical manuscripts with focus on high quality and low effort
- Metafont: companion to TEX; aims to create refined and portable fonts
- together they describe how a page will look, i.e. where a character will go (TEX) and how it will look (Metafont)
- LATEX: format for TEX; defines macros to make TEX more user-friendly, readable

# Table of contents

# Importance of Metafont

- high flexibility and portability through "meta-design"
  - introduced mathematical font design
- Imitation of "pen strokes"
- Developed with a focus on typesetting
- integrated into TEX

# Table of contents

# Functions

- Metafont is algebraic, declarative
  - variables are defined through equations
  - often simplifies equations automatically
- equations are relations, not assignments!
- algebra enables the "meta" aspect of Metafont

## The Fundamental Building Blocks of Metafont

- points defined by using Cartesian coordinates
- paths between points
- different pens

# Algebra: Example

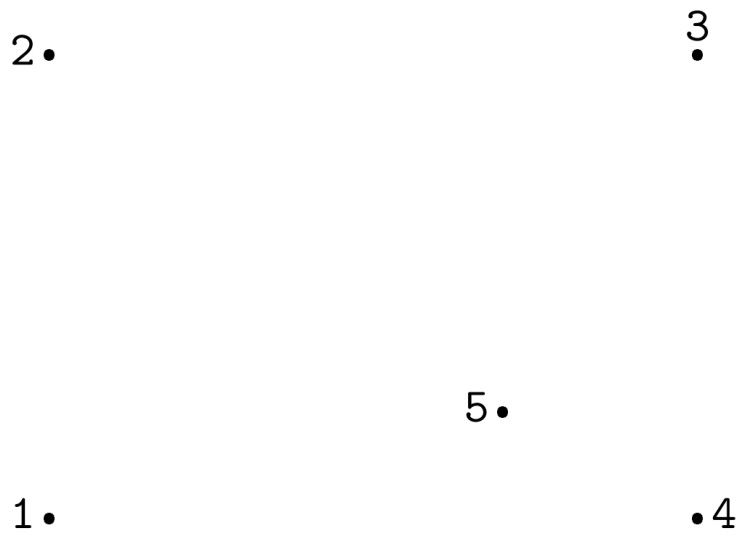
```
*tracingequations:=tracingonline:=1;
*a+b-c=0;
## c=b+a
*c=2a;
## b=a
*a=5;
## a=5
#### b=5
#### c=10
*c=0;
!  Inconsistent equation (off by -10).
<to be read again>
;
<*> c=0;
?
```

# Points

- Points are:
  - pairs of Cartesian coordinates on two dimensional plane
- numbered using an index, e.g. $n$-th point: $(x_n, y_n)$ or simply as $z_n$
- basic vector arithmetic for points: addition, subtraction, scaling, rotation
- mediation operator between two points:
  - "t-of-the-way" point
  - described by $t[z_n, z_m] = z_n + t(z_m - z_n)$

# Points: Example

```
z1=(10,10);
x3=15x1;
y3=y1+100;
z2=(x1,y3);
z4=(x3,y1);
z5 = 0.7[z2,z4];
labels(range 1 thru 5);
shipit;
end
```



# Paths

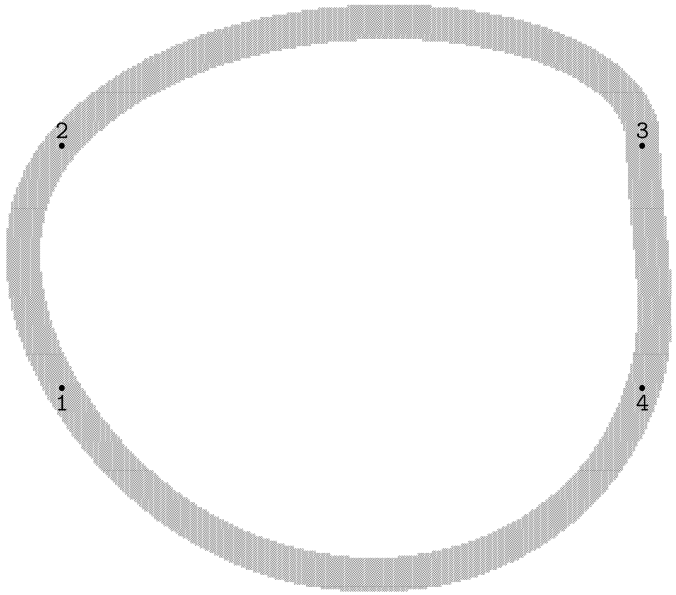
- can be curves or straight lines
- *draw* connects points using De Casteljau's algorithm

$$z(t) = (1-t)^3 z_1 + 3(1-t)^2 t\, z_2 + 3(1-t) t^2 z_3 + t^3 z_4, \text{ with } t = \tfrac{1}{2}$$

- granular path control by optional operators: tension, direction, cycle
- Metafont does not need all four key points of a curve
  - can determine missing key points by itself

# Paths: Example

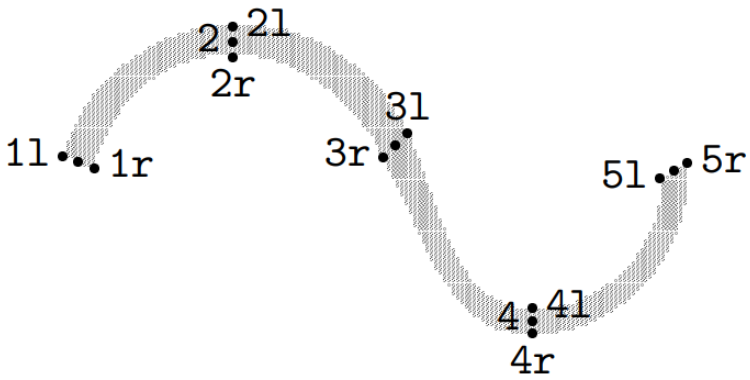
```
z1=(10,10);
x3=25x1;
y3=y1+100;
z2=(x1,y3);
z4=(x3,y1);
labels(range 1 thru 4);
draw z1..z2..tension1.5..z3{down}..z4..cycle;
shipit;
end
```



# Pens

- different pens allow for more interesting fonts
- *pickup* command to choose pen
- same scaling and rotation operators as for points
- alternative way of drawing letters: draw outline and fill in shape (analogous command: *unfill*)
- definition of penpositions makes filling in more intuitive
  - define pen breadth and angle at a specific point
  - e.g. $penpos_k(b, d)$: pen has breadth b and angle d at point $z_k$
  - *penstroke* command used to fill in shape

# Pens: Example




```
1 = 1r + (-3.8,1.4)
3 = 3r + (2.8,2.8)
5 = 5l + (3.2,1.9)
```

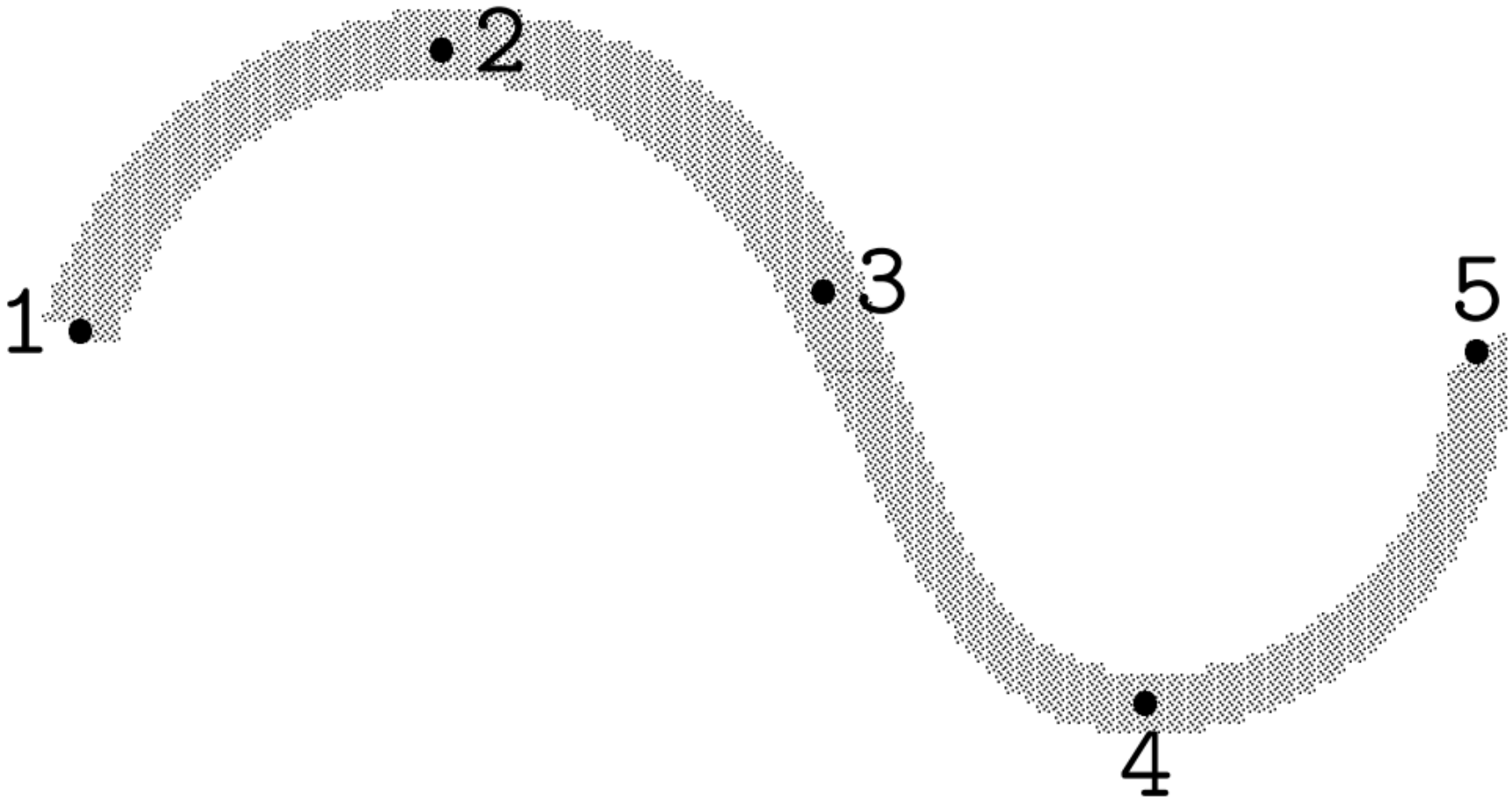
# Pens: Example Code

```
pickup pencircle xscaled 1.4pt yscaled 0.4pt;
z1=(24,49); z1r=z1l+(8,0) rotated -20; z1=0.5[z1l,z1r];
z2=(60,77); z2r=z2l+(7,0) rotated -90; z2=0.5[z2l,z2r];
z3=(98,53); z3r=z3l+(8,0) rotated -135; z3=0.5[z3r,z3l];
z4=(130,12); z4r= z4l +(6,0) rotated -90; z4=0.5[z4l,z4r];
z5=(163,47); z5r=z5l + (7.4,0) rotated 30; z5=0.5[z5l,z5r];
fill z1l..z2l..z3l..z4l{right}..z5l--z5r..{left}z4r..z3r..z2r..z1r--cycle;
labels(range 1 thru 5);
labels(1l,1r,2l,2r,3l,3r,4l,4r,5l,5r);
```

# Using penpos: Example Code

```
pickup pencircle xscaled 1.4pt yscaled 0.4pt;
z1=(24,49);
z2=(60,77);
z3=(98,53);
z4=(130,12);
z5=(163,47);
penpos1(8,-20);
penpos2(7,-90);
penpos3(8,-135);
penpos4(6,-90);
penpos5(7.4,30);
penstroke z1e..z2e..z3e..z4e{right}..z5e;
labels(range 1 thru 5);
shipit;
end
```

# Using penpos: Example



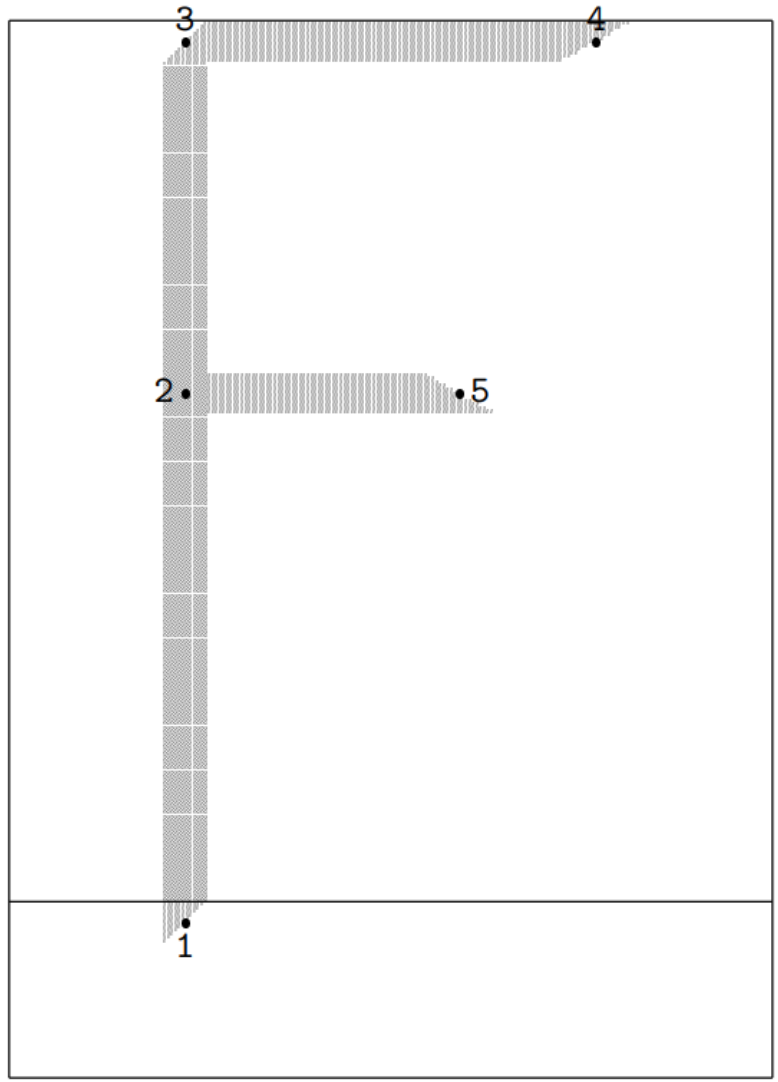
# Table of contents

# Font Creation: Introduction to Proof-Mode

- font creation involves trial-and-error-process called "proofing"
  - enlarging characters in order to make out fine details
  - visualization helps creator understand and perfect algebraic formulas and relations
- often involves comparing different version of same characters with different values
- tedious process, which proof-mode aims to simplify:
  - skips process of compiling and installing font
  - creates proof-sheet
  - key points are marked

# Font Creation: Our First Draft

METAFONT output 2023.01.27:2134 Page 1 Character 70 “Capital F”

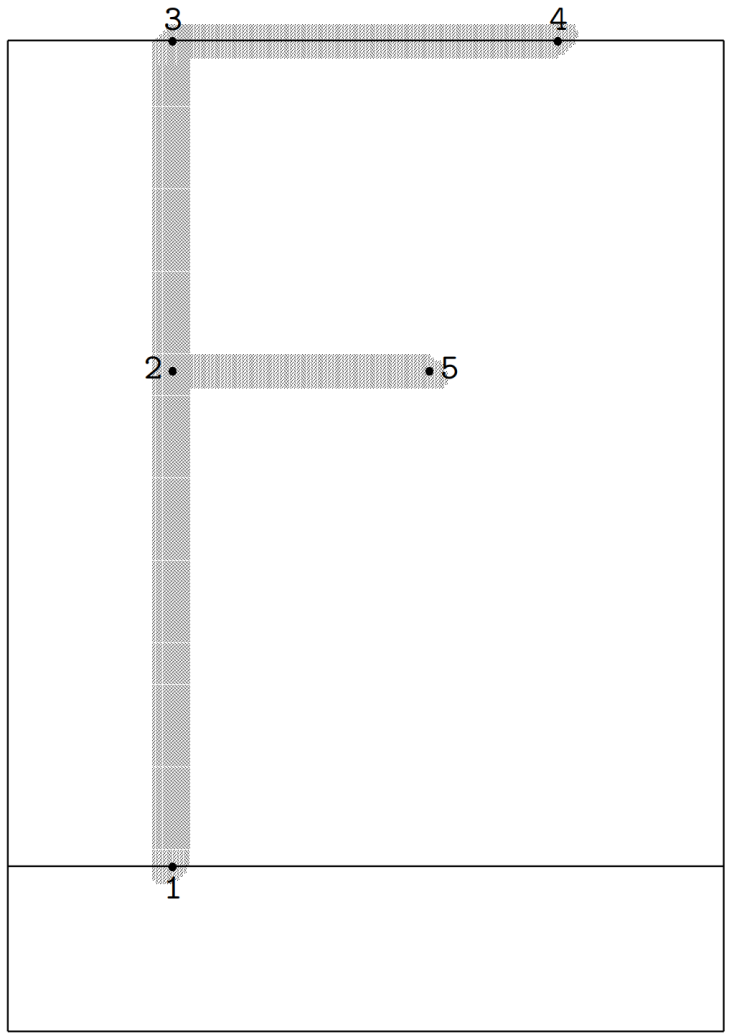


# Font Creation: Our First Draft

```
mode_setup;
u#:=4/9pt#;
define_pixels(u);
beginchar("F",13u#,15u#,3u#); "Capital F";
        x1=x3=3u;
        y1=-.3536u;
        z2=.6[z1,z3];
        y3=y4=h-.3536u;
        x4=w-x1;
        y5=y2;
        x5=2/3[x2,x4];
        penpos1(u,45);
        penpos2(u,45);
        penpos3(u,45);
        penpos4(1.4142u,30);
        penpos5(1.4142u,150);
        penstroke z1e--z3e;
        penstroke z3e--z4e;
        penstroke z2e--z5e;
        labels(range 1 thru 5);
endchar;
end
```

# Font Creation: Our Second Draft

METAFONT output 2023.01.27:2136 Page 1 Character 70 “Capital F”

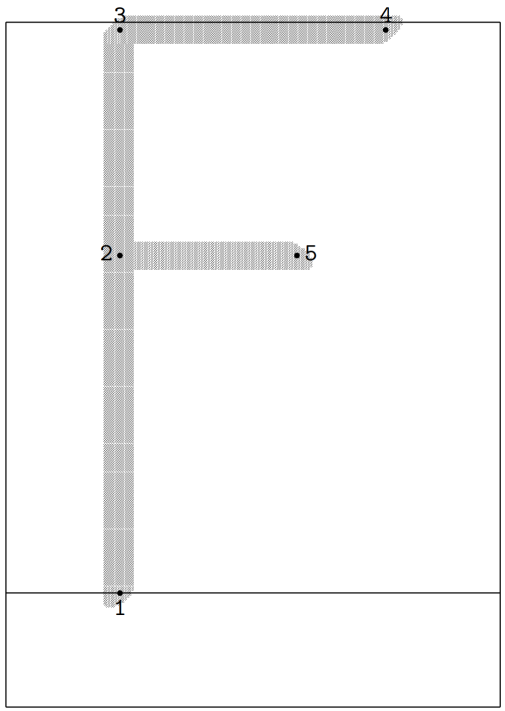


# More extensive Proofing: Code

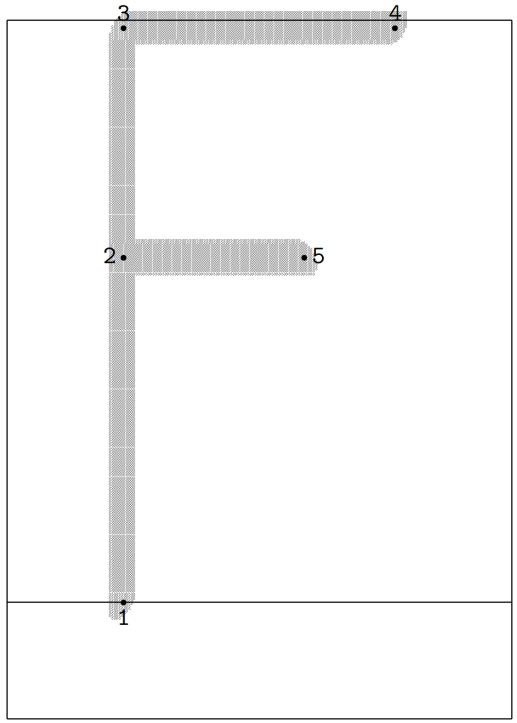
```
u:=4/9pt;
define_pixels(u);
def test_T(expr code,pAngle) =
beginchar(code,13u,15u,3u); "Capital F";
x1=x3=3u;
y1=0;
z2=.6[z1,z3];
y3=y4=h;
x4=w-x1;
y5=y2;
x5=2/3[x2,x4];
pickup pencircle scaled .5u xscaled 2 rotated pAngle;
draw z1--z3;
draw z3--z4;
pickup currentpen rotated -2pAngle;
draw z2--z5;
labels(range 1 thru 5);
endchar; enddef;
```

# More Extensive Proofing: Candidate Examples

METAFONT output 2023.01.25:0333 Page 4 Character 104 “Capital F”

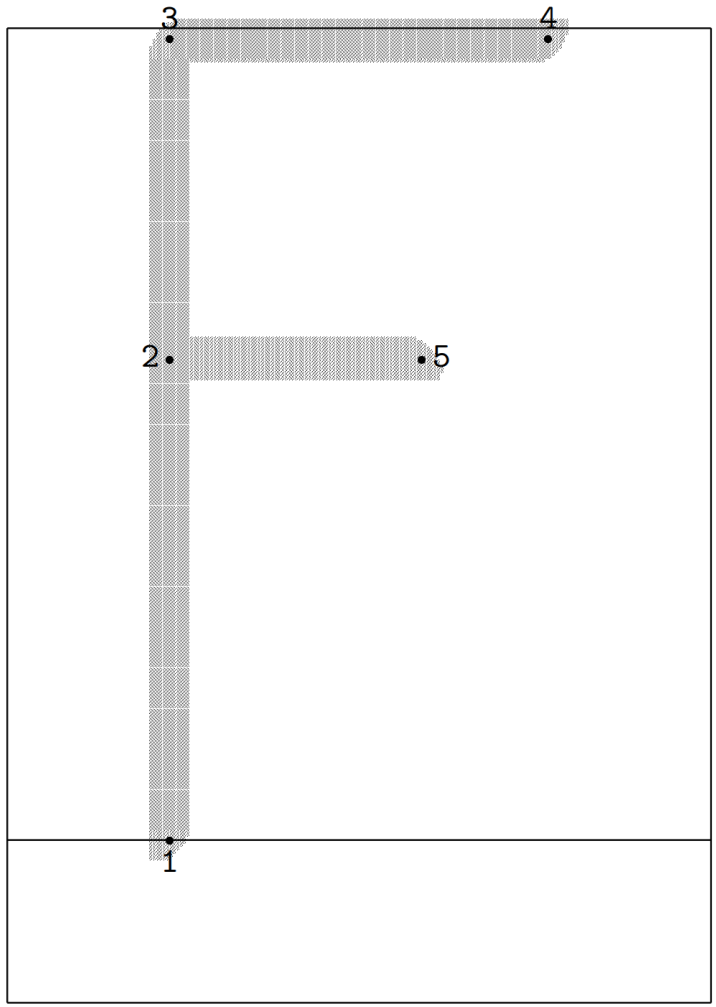



METAFONT output 2023.01.25:0333 Page 6 Character 106 “Capital F”

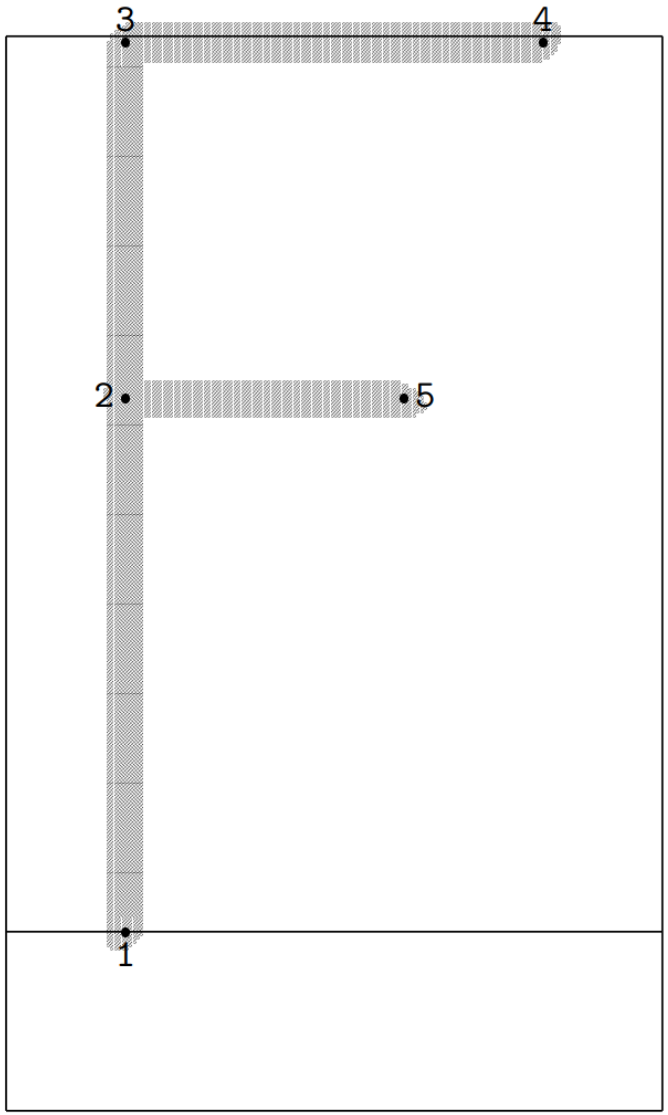


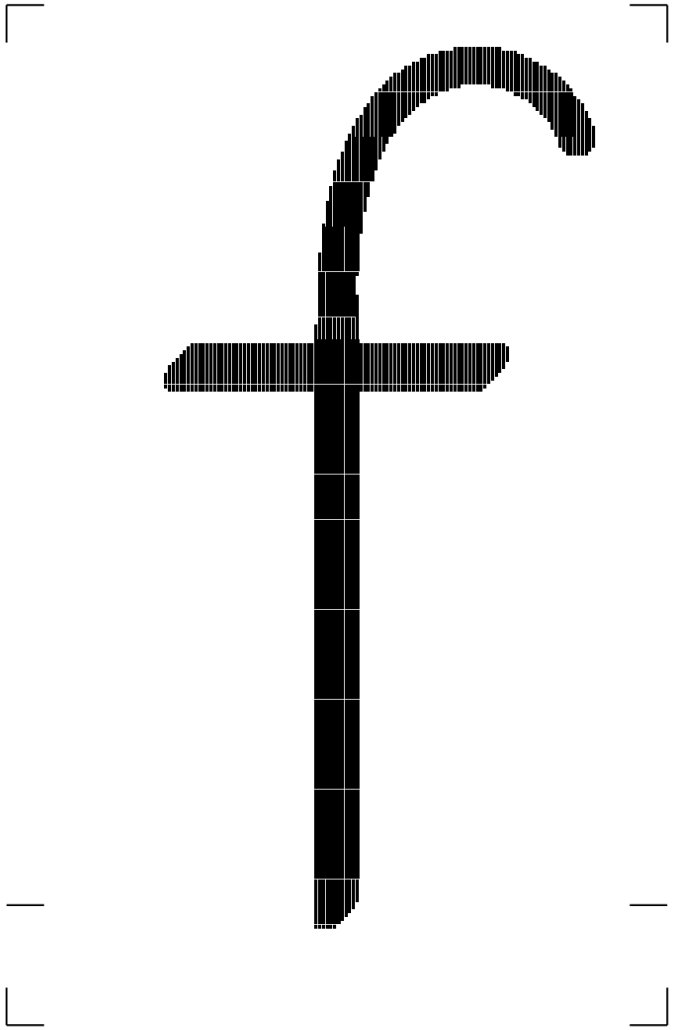
# The Candidate We Chose

METAFONT output 2023.01.25:0333 Page 5 Character 105 “Capital F”



# Font Creation: The Final Product



# Table of contents

# Reasons for differences

- Metafont invented first
- developed parallel to eachother thus independently
- Postscript was made to describe layout of a page
- very specific idea behind Metafont

# The idea

## Postscript

1. general-purpose program
2. placing points around an coordinate systems
3. stack-based

# The idea

## Postscript

1. general-purpose program
2. placing points around an coordinate systems
3. stack-based

## Metafont

1. special-purpose program for fonts
2. points described by relations to each other
3. dynamic memory allocation

# Basic differences

| | Postscript | Metafont |
|---|---|---|
| Functions | stack needs to be considered | equations |
| Points | only positive x- and y-Achsis<br>treated as any other number | cartesian coordinates<br>saved as points |
| Paths | current path<br>Graphics State Stack<br>needs all key points<br>`lineto` for all paths needed before `stroke` | no current path<br>every path directly saved<br>can determine missing key points<br>`draw` for multiple points |

# Drawing a line

PostScript:

```
newpath
100 200 moveto
200 250 lineto
100 300 lineto
2 setlinewidth
stroke
```

# Drawing a line

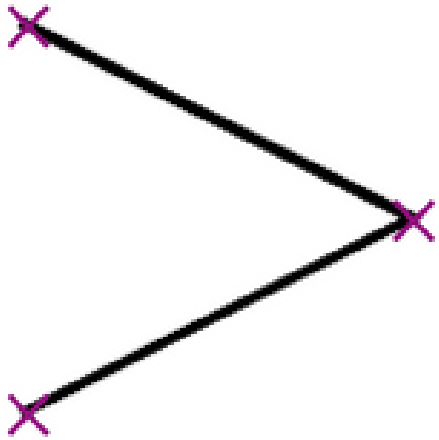
## Postscript:

```
newpath
100 200 moveto
200 250 lineto
100 300 lineto
2 setlinewidth
stroke
```

# Drawing a line

## Postscript:

```
newpath
100 200 moveto
200 250 lineto
100 300 lineto
2 setlinewidth
stroke
```



# Drawing a line

## Metafont:

```
z1=(a,b);
z2=(a+100,b+50);
z3=(a,b+100);
a = 100; b = 200;
pickup pencircle scaled 2;
draw z1--z2--z3;
```

# Drawing a line

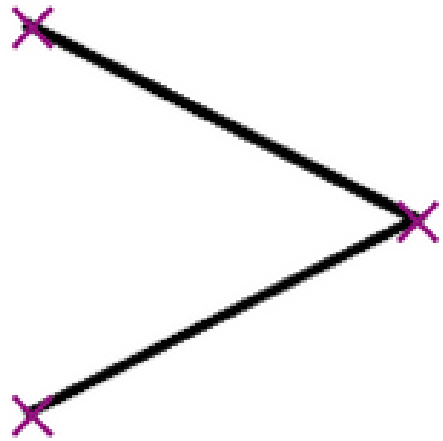
## Metafont:

```
z1=(a,b);
z2=(a+100,b+50);
z3=(a,b+100);
a = 100; b = 200;
pickup pencircle scaled 2;
draw z1--z2--z3;
```

# Drawing a line

## Metafont:

```
z1=(a,b);
z2=(a+100,b+50);
z3=(a,b+100);
a = 100; b = 200;
pickup pencircle scaled 2;
draw z1--z2--z3;
```



# Drawing a line

## Postscript

- `newpath` at beginning ending with `stroke`
- need to move from point to point
- only one type of line with different width

# Drawing a line

## Postscript

- `newpath` at beginning ending with `stroke`
- need to move from point to point
- only one type of line with different width

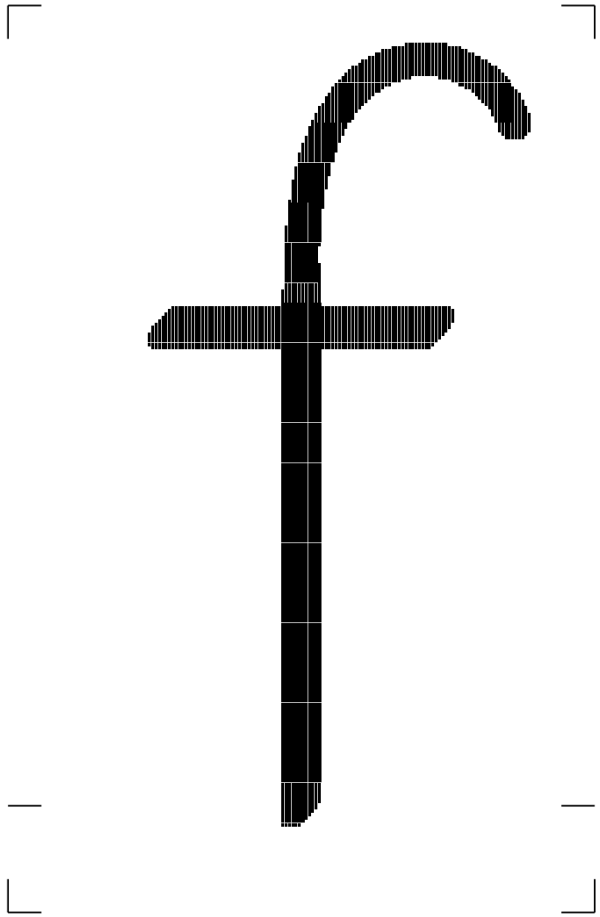
## Metafont

- points defined through relation to each other
- no need to move
- Different pens with different width
- drawing with `draw` and points to connect

# Font creation

## Our font:

```
u:=4/9pt;
define_pixels(u);
pickup pencircle scaled .5u xscaled 2 rotated 50;
x1=x2=.5w;
y1=0;
y2=0.7[y1,y3];
y4=y2+3u;
x3=0.5[x2,x4];
y3=h-u;
x4=x2+4u;
y5=y6= .6[0, h-.1u];
x1=.5[x5,x6];
x5=x1-2.5u;
draw z1-z2;
a := cosd 50;
pickup pencircle scaled u;
pickup currentpen scaled a;
draw z2up..z3..z4;
pickup pencircle scaled .5u xscaled 2 rotated
50;
draw z5-z6;
labels(range 1 thru 6);
```



# Font creation

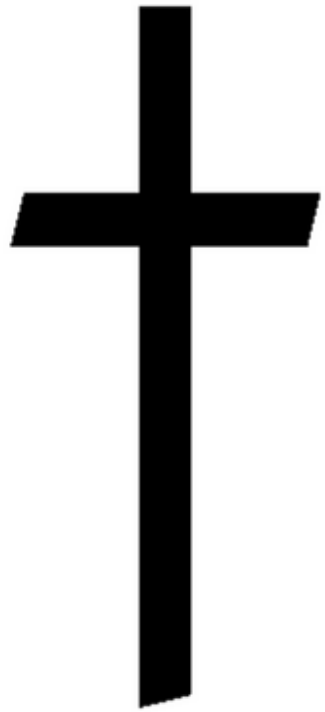
## Realization in postscript:

```
newpath
100 350 moveto
100 530 lineto
50 530 lineto
55 550 lineto
100 550 lineto
100 623 lineto
120 623 lineto
120 550 lineto
170 550 lineto
165 530 lineto
120 530 lineto
120 355 lineto
100 350 lineto
stroke
 fill
newpath
145 623 45 0 180 arc
145 623 25 180 360 arcn
180 623 10 180 0 arc
closepath
fill
stroke
```

# Font creation

## Realization in postscript:

```
newpath
100 350 moveto
100 530 lineto
50 530 lineto
55 550 lineto
100 550 lineto
100 623 lineto
120 623 lineto
120 550 lineto
170 550 lineto
165 530 lineto
120 530 lineto
120 355 lineto
100 350 lineto
stroke
fill
 newpath
145 623 45 0 180 arc
145 623 25 180 360 arcn
180 623 10 180 0 arc
closepath
fill
stroke
```



# Font creation

## Realization in postscript:

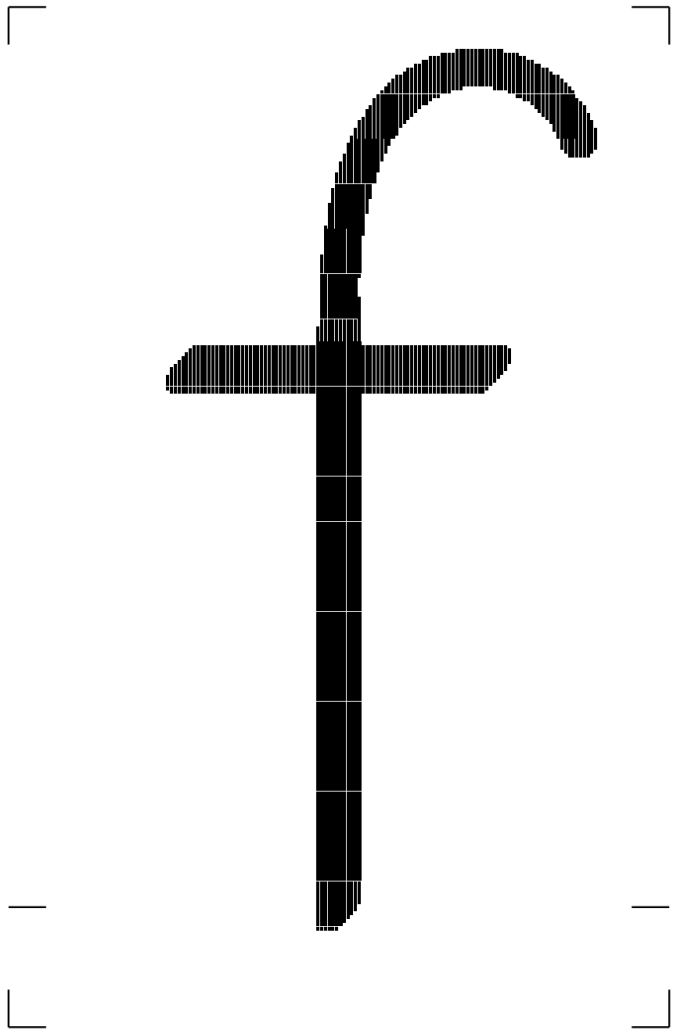
```
newpath
100 350 moveto
100 530 lineto
50 530 lineto
55 550 lineto
100 550 lineto
100 623 lineto
120 623 lineto
120 550 lineto
170 550 lineto
165 530 lineto
120 530 lineto
120 355 lineto
100 350 lineto
stroke
fill
newpath
145 623 45 0 180 arc
145 623 25 180 360 arcn
180 623 10 180 0 arc
closepath
 fill
stroke
```

# Font creation

## Realization in postscript:

```
newpath
100 350 moveto
100 530 lineto
50 530 lineto
55 550 lineto
100 550 lineto
100 623 lineto
120 623 lineto
120 550 lineto
170 550 lineto
165 530 lineto
120 530 lineto
120 355 lineto
100 350 lineto
stroke
fill
newpath
145 623 45 0 180 arc
145 623 25 180 360 arcn
180 623 10 180 0 arc
closepath
fill
stroke
```

# Font creation



# Font creation

## Postscript

1. linstall FontForge, RoboFont or FontLab
2. design your font
3. set metrics like character space, kerning etc.
4. test your font
5. export your font as OpenType or TrueType

# Font creation

## Postscript

1. linstall FontForge, RoboFont or FontLab
2. design your font
3. set metrics like character space, kerning etc.
4. test your font
5. export your font as OpenType or TrueType

## Metafont

1. learn Metafont
2. define basic parameters like size and character spacing
3. design the characters, define outlines and kerning
4. test your font
5. generate your font by running it through TEX
6. install your font

# Scaling fonts

## Postscript

- uses verctor-based output based on mathematical descriptions
- easily scaled for the resolution and used mostly for graphic design

# Scaling fonts

## Postscript

- uses verctor-based output based on mathematical descriptions
- easily scaled for the resolution and used mostly for graphic design

## Metafont

- either General Format(gf) or Packed(pk)
- gf as the standard output based on mathematical equations being easily scaled but big in size
- pk: compressed bitmap version of gf format smaller in size but bad scalability

# Table of contents

# MetaPost

- created by John Hobby
- combination of Metafont and PostScript
- Metafont syntax with PostScript output

## Examples of problems fixed by MetaPost

- fonts in form of EPS (Encapsulated PostScript), SVG (Scalable Vector Graphic) or PNG
- uses the RGB or CMYK colors

# MetaPost features

## From Metafont:

- numbers
- cubic splines
- affine transformations
- text strings
- boolean quantities

# MetaPost features

## From Metafont:

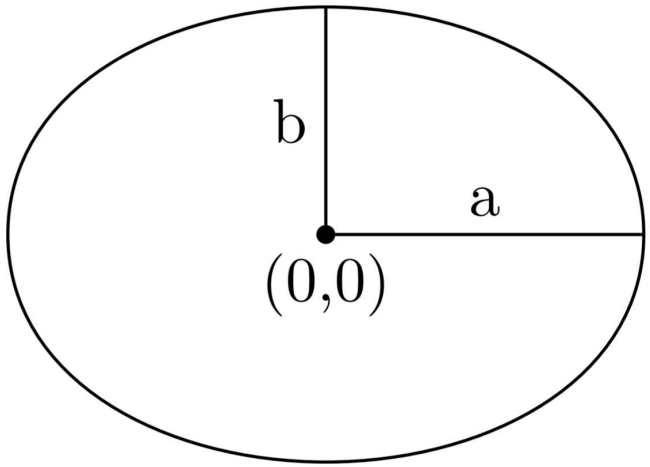
- numbers
- cubic splines
- affine transformations
- text strings
- boolean quantities

## New features:

- integrating text and graphics
- clipping
- shading
- sashed lines

# Drawing in Metapost

```
beginfig(17);
a=.7in; b=.5in;
z0=(0,0);
z1=-z3=(a,0);
z2=-z4=(0,b);
draw z1..z2..z3..z4..cycle;
draw z1-z0-z2;
label.top("a", .5[z0,z1]);
label.lft("b", .5[z0,z2]);
dotlabel.bot("(0,0)", z0);
endfig;
```



# Table of contents

# Converting Metafont to Type 1 fonts

## What are Type 1 fonts?

- standarization for PostScript's font formats used by Adobe
- line segments and Bézier curves of the third degree
- outdated today and already replaced by the OpenType format

Examples of original Type 1 fonts:

- Helvetica
- Times

# Converting Metafont to Type 1 fonts

## Why Type 1 fonts and not OpenType?

- not as much information avaliable
- conversion Metafont to OpenType is very similar
- conversions still relevant

# Converting Metafont to Type 1 fonts

## Autotracing Bitmaps

1. reads original Metafont sources
2. generates high-resolution pk bitmaps
3. calls an autotracing program
4. generates a file in Type 1 format

# Converting Metafont to Type 1 fonts

## Autotracing Bitmaps

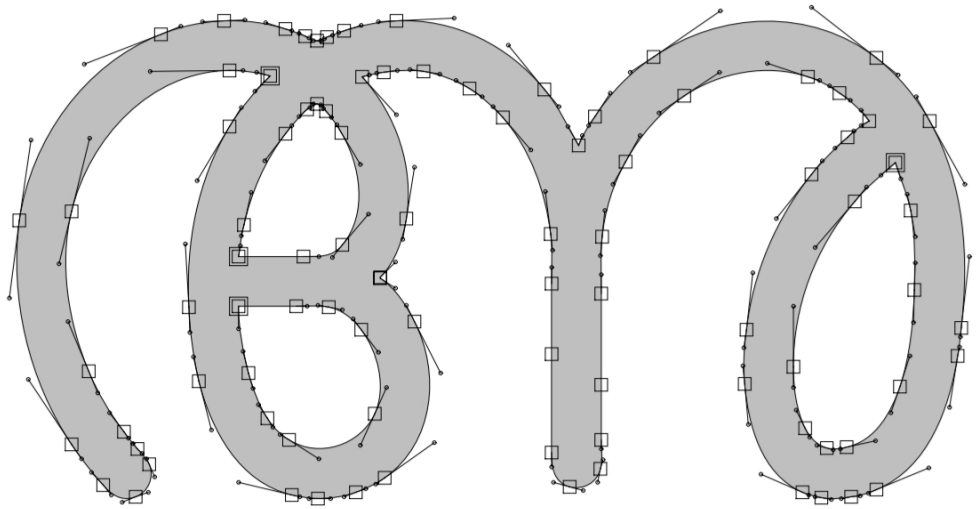
1. reads original Metafont sources
2. generates high-resolution pk bitmaps
3. calls an autotracing program
4. generates a file in Type 1 format

## Analytic conversion

1. conversion from Metafont to MetaPost
2. analyses MetaPost files
3. creates PostScript files with only outlines
4. generates a file in Type 1 format

# Autotracing Bitmaps

- examples: *TEX*trace, mftrace
- either AutoTrace or potrace as an autotracing program
- focus on points of extremes, target continuity, conciseness and consistency during tracing
- after determining those points, traces letter based on them



# Autotracing Bitmaps

## advantages

- reasonably close conversion
- simple, robust and fully automatic

# Autotracing Bitmaps

## advantages

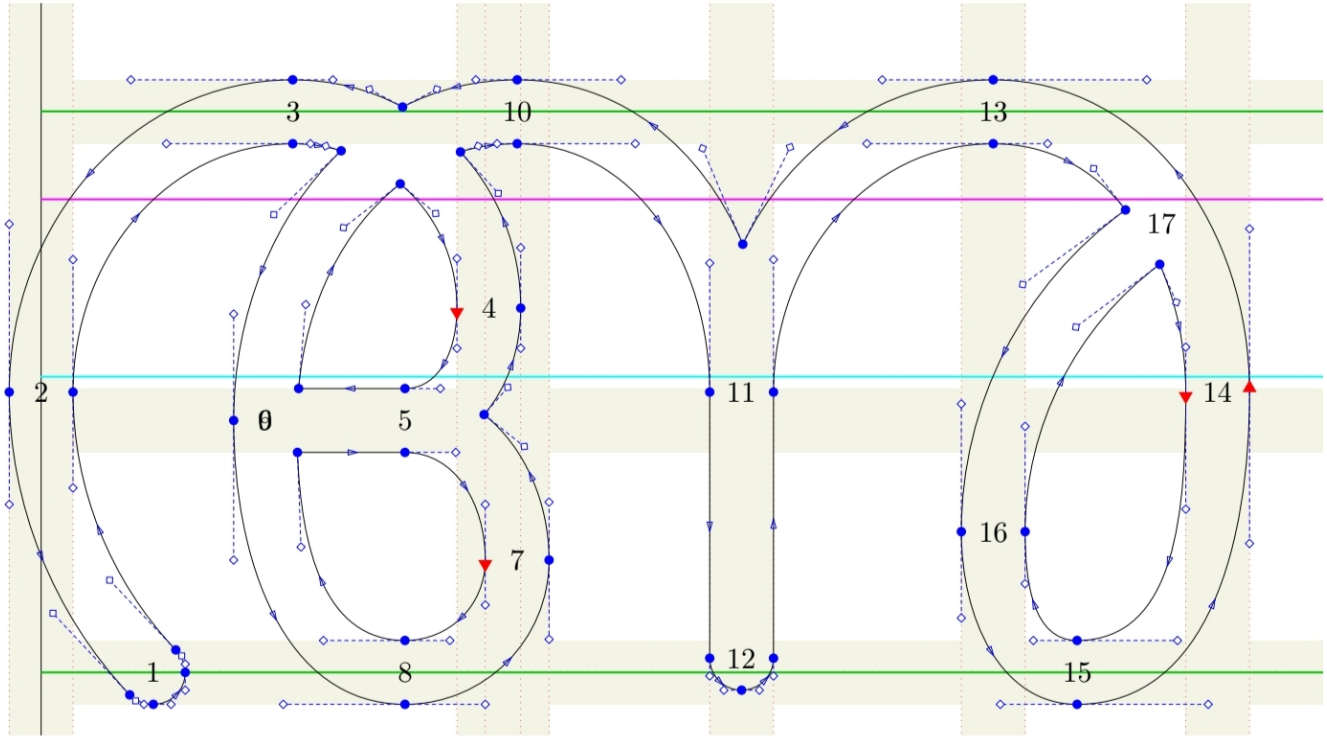
- reasonably close conversion
- simple, robust and fully automatic

## disadvantages

- only approximations
- information about nodes and other control points gets lost
- Type 1 conventions not satisfied by final fonts
- AutoTrace: problems with creating bumps/holes and recognizing corners
- potrace: problems with loss of target, violation of horizontal lines, vertical directions

# Analytic Conversion

- examples: MetaType1, MetaFog
- Type 1 fonts from Metapost sources
  $\rightarrow$ conversion of Metafont into Metapost first
- algorithm then evaluates source and creates a Type 1 font



# Analytic Conversion

## advantages

- MetaType1: complete support for Type 1, simple commands for manual insertion
- MetaFog: conversion fully automatic, for very complex fonts

# Analytic Conversion

## advantages

- MetaType1: complete support for Type 1, simple commands for manual insertion
- MetaFog: conversion fully automatic, for very complex fonts

## disadvantages

- MetaType1: manual conversion from Metafont to Metapost, bad pen stroking algorithm
- MetaFog: slow processing, cannot process Metafont-specific definitions

# Table of contents

# Why don’t we use Metafont today?

Metafont’s goals:

- automation of the typesetting process
- freedom of font creation for everyone as needed
- making typesetting "logical" and "intellectual"

# Why don’t we use Metafont today?

1st failure of Metafont:

- tried to emulate a pen drawing
  - too complicated for certain situations
  - too simplistic for other ones

# Why don’t we use Metafont today?

## 2nd failure of Metafont:

- very expandable typesetting program
  - not everyone wants to create their own font
  - turned out too complicated for non programmers

# Why don’t we use Metafont today?

## 3rd failure of Metafont:

- invented mathematical typesetting
  - user base of designers are no mathematicians
  - certain letters too complicated

**consequences**

- whole paper about the formula describing the letter ’s’
- ’s’ took an experienced programmer 3 sleepless nights

# What do you think will happen to Metafont in the future?

# Sources


[1] Nelson HF Beebe. “The design of TEX and Metafont: A retrospective”. In: *The Communications of the TEX Users Group* (2005). https://tug.org/TUGboat/tb26-1/beebe.pdf/ Accessed: 2023-02-23, p. 33.

[2] Donald E Knuth. “The concept of a meta-font”. In: *Visible language* 16.1 (1982). https://m-u-l-t-i-p-l-i-c-i-t-y.org/media/pdf/The-Concept-of-a-Meta-font.pdf/ Accessed: 2023-02-23, pp. 3–27.

[3] Donald Ervin Knuth. *The METAFONTbook*. eng. 01. [Aufl.] Reading, Mass. [u.a: Addison Wesley, 1986. ISBN: 0201134446.

[4] Donald Ervin Knuth. *The TEXbook*. eng. 6. print. Computers typesetting / Donald E. Knuth A. Reading, Mass. [u.a: Addison-Wesley, 1986. ISBN: 0201134470.

# Sources


[1] CORPORATE Adobe Systems Inc. *PostScript language reference*. https://www.adobe.com/jp/print/postscript/pdfs/PLRM.pdf/ Accessed: 2023-02-23. Addison-Wesley Longman Publishing Co., Inc., 1999.

[2] John D Hobby. *MetaPost*. https://tug.org/docs/metapost/mpman.pdf/ Accessed: 2023-02-23. 2013.

[3] Donald E Knuth. *METAFONT: A System for Alphabet Design*. Tech. rep. https://apps.dtic.mil/sti/pdfs/ADA083229.pdf/ Accessed: 2023-02-23. STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1979.

[4] Geoffrey Tobin. "METAFONT for Beginners". In: *available from public file-servers* 198 (1993). http://tug.ctan.org/info/metafont/beginners/metafont-for-beginners.pdf/ Accessed: 2023-02-23.

# Sources


[1] Christophe Grandsire. *The METAFONTtutorial*. http://metafont.tutorial.free.fr/ Accessed: 2023-02-23. 2004.

[2] Taco Hoekwater. “Generating Type 1 fonts from METAFONT sources”. In: *TUGboat* 19.3 (1998). https://tug.org/TUGboat/tb19-3/hoek1.pdf/ Accessed: 2023-02-23, pp. 256–266.

[3] Klaus Höppner. “A short introduction to MetaPost”. In: *ArsTEXnica* (2008). https://tug.org/tug2009/preprints/hoeppner.pdf/ Accessed: 2023-02-23, p. 5.

[4] Karel Pıška. “Creating Type 1 fonts from METAFONT sources: Comparison of tools, techniques and results”. In: https://tug.org/TUGboat/tb25-0/piska.pdf/ Accessed: 2023-02-23. Citeseer. 2004.

# Sources


[1] Dave Crossland. “Why didn’t Metafont catch on?” In: *TUGboat* 29.3 (2008). https://tug.org/TUGboat/tb29-3/tb93crossland.pdf/ Accessed: 2023-02-23, pp. 418–420.

[2] BIJLAGE GG. “The future of TEX and METAFONT”. In: (1990). http://www.ntg.nl/maps/05/34.pdf/ Accessed: 2023-02-23.