GENERATIONS
SIGGRAPH2018

Digital Typography

25 years of text rendering in computer graphics

SIGGRAPH '18 Course - August 12-16, 2018 - Vancouver, BC, Canada - 10.1145/3214834.3214837

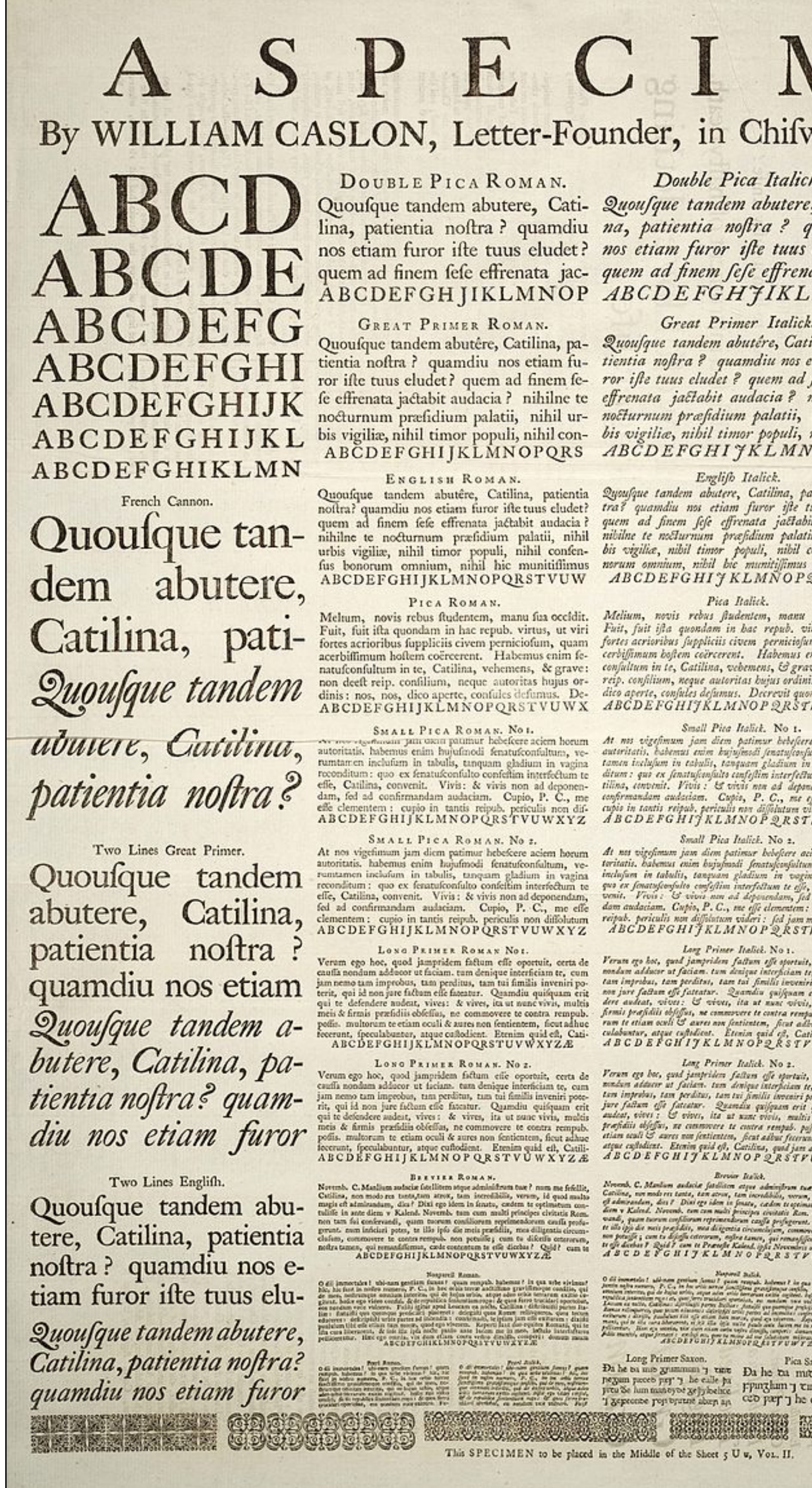
GENERATIONS
SIGGRAPH2018

INTRODUCTION

Typography is *the art of arranging type to make written language legible, readable, and appealing when displayed*.

However, for the neophyte, typography is mostly apprehended as the juxtaposition of characters displayed on the screen while for the expert, typography means typeface, scripts, unicode, glyphs, ascender, descender, tracking, hinting, kerning, shaping, weight, slant, etc.

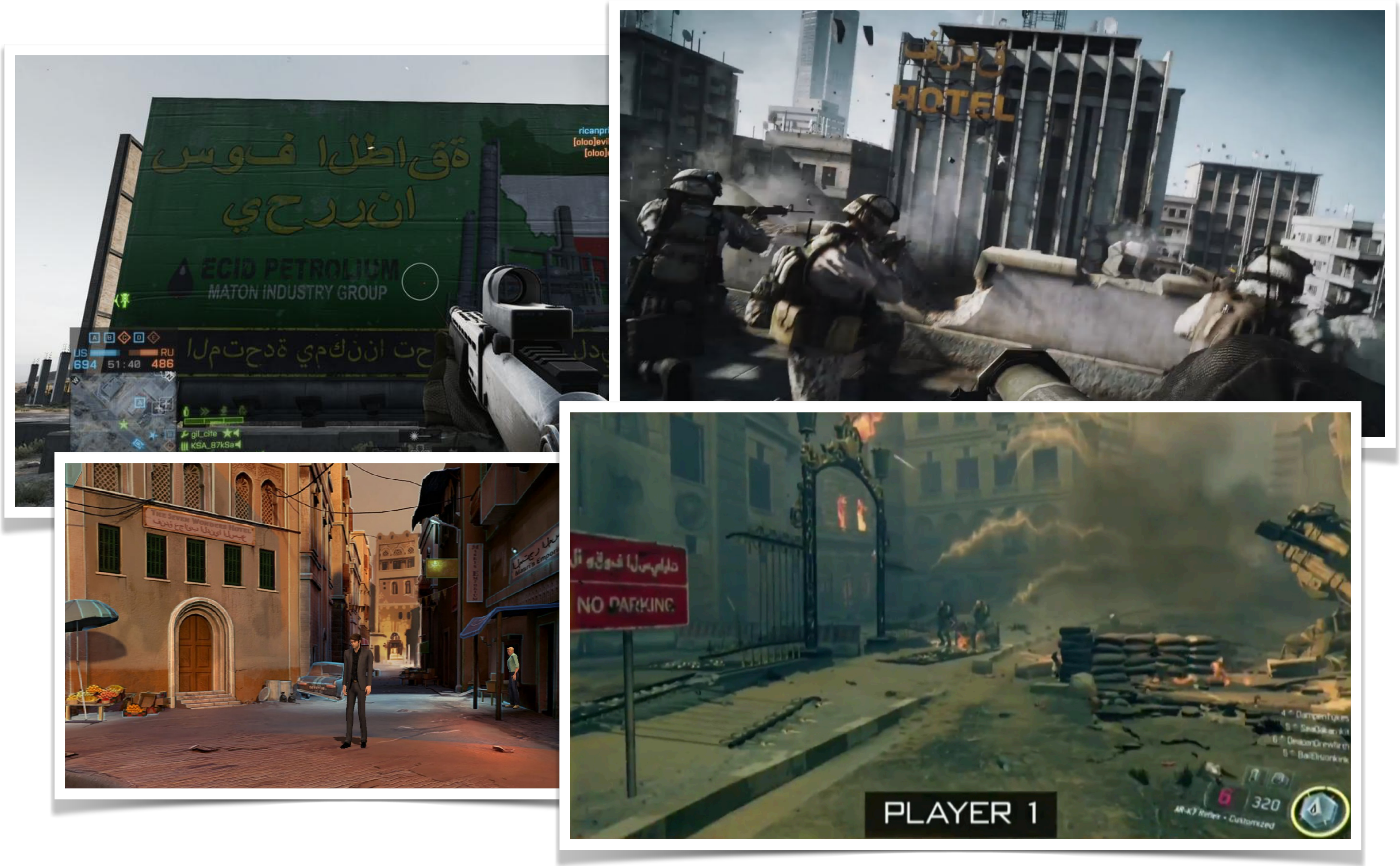
Typography is actually much more than the mere rendering of glyphs and involves many different concepts.



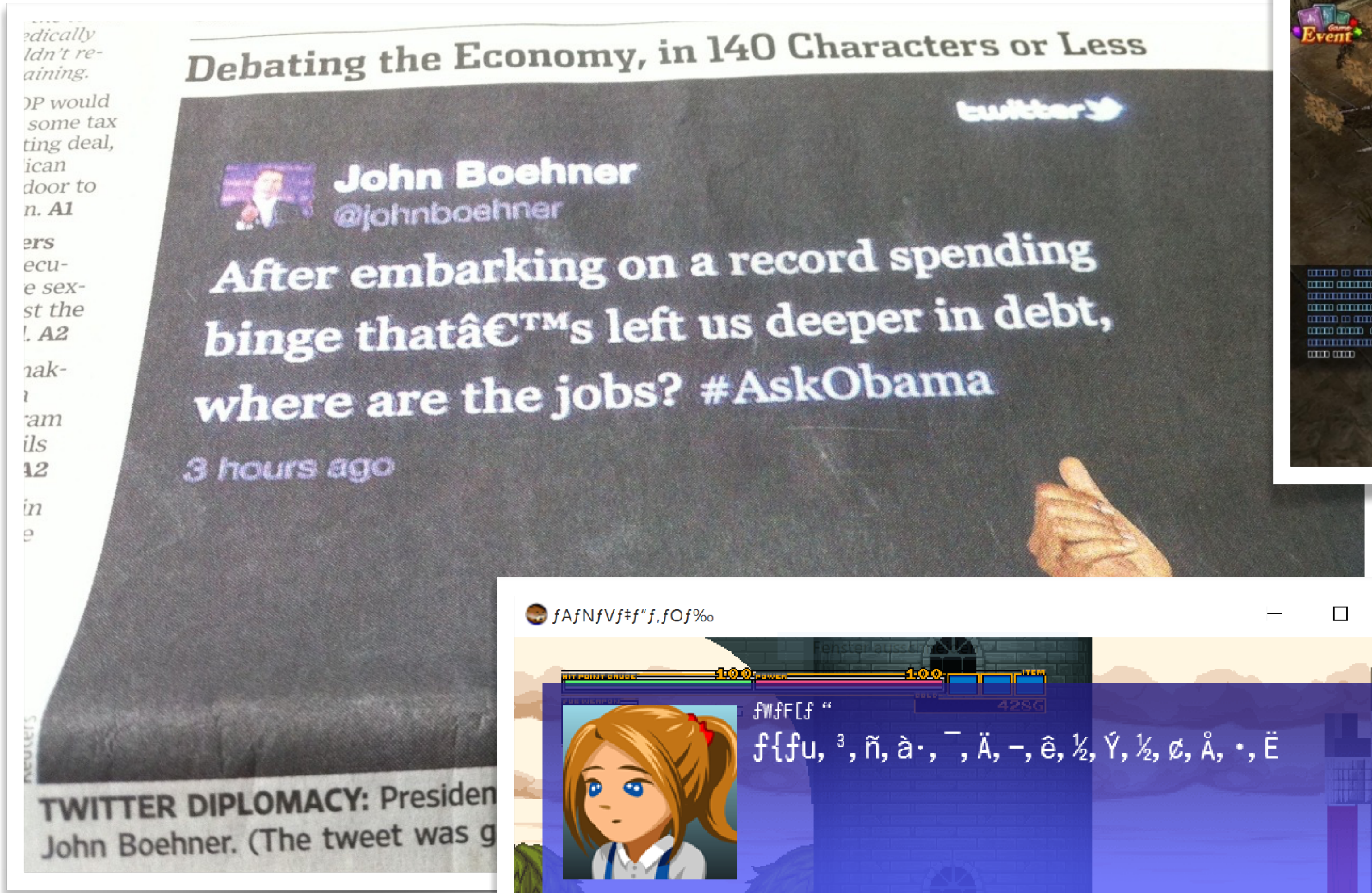
A specimen sheet of typefaces and languages, by William Caslon I, letter founder; dated 1734.

NOPE, NOT ARABIC!

From “Nope, not Arabic” tumbler (https://nopenotarabic.tumblr.com/)



I ❓ Unicode


Debating the Economy, in 140 Characters or Less
twitter
John Boehner
@johnboehner
After embarking on a record spending binge thatâ€™s left us deeper in debt, where are the jobs? #AskObama
3 hours ago
TWITTER DIPLOMACY: Presiden
John Boehner. (The tweet was g


U+1F4A9
HAPPENS

N. ROUGIER (INRIA) & B. ESFAHBOD (GOOGLE)

BAD KERNING (keming)

Disaster is a space away… (FLICK, click, CLINT, FINAL)

Rockstar game removed the extra space

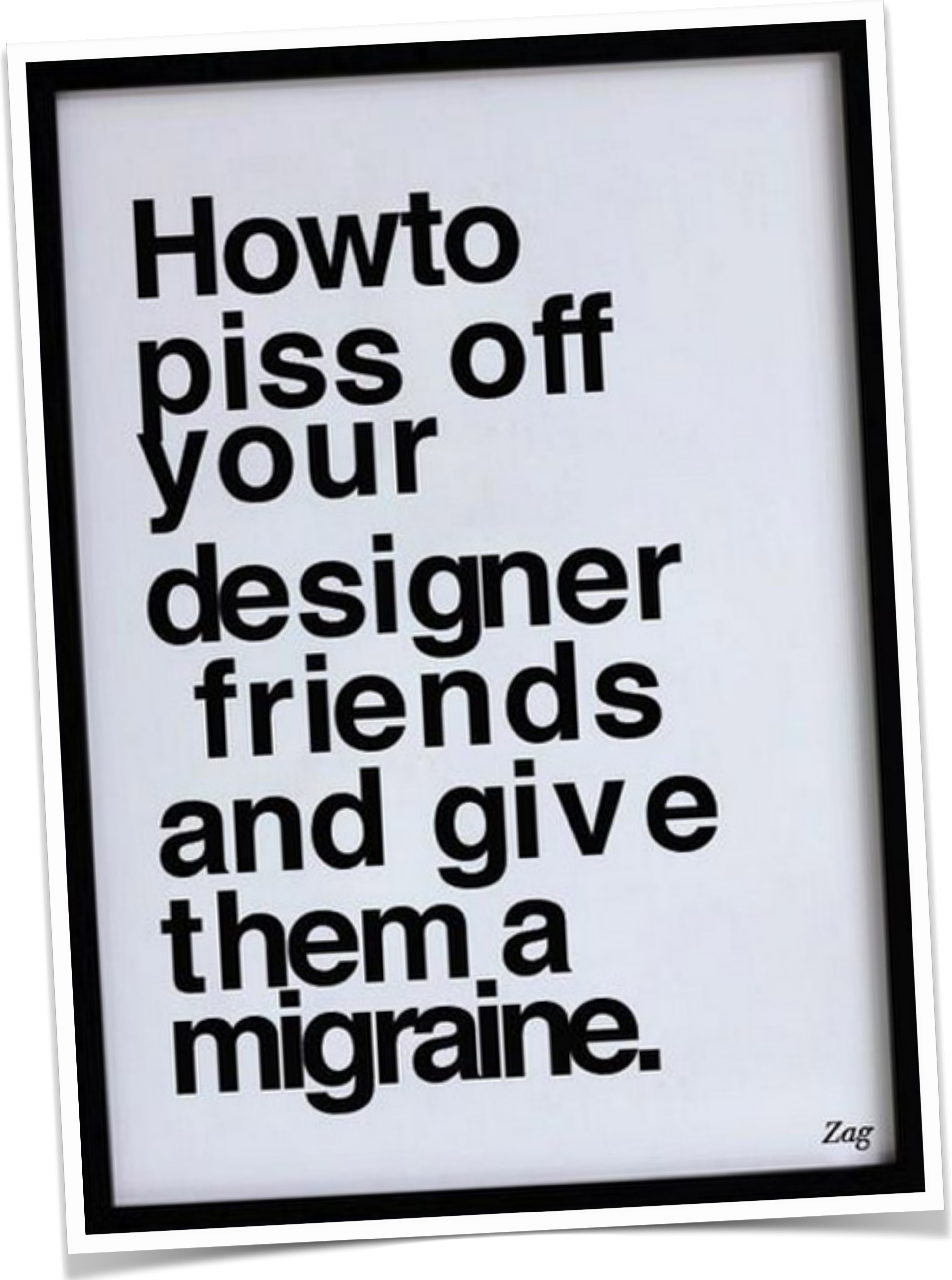
click versus dick → Apple added an extra space

GIBBERISH CAN BE PRETTY...

The Matrix code is actually a sushi recipe...

DIGITAL TYPOGRAPHY

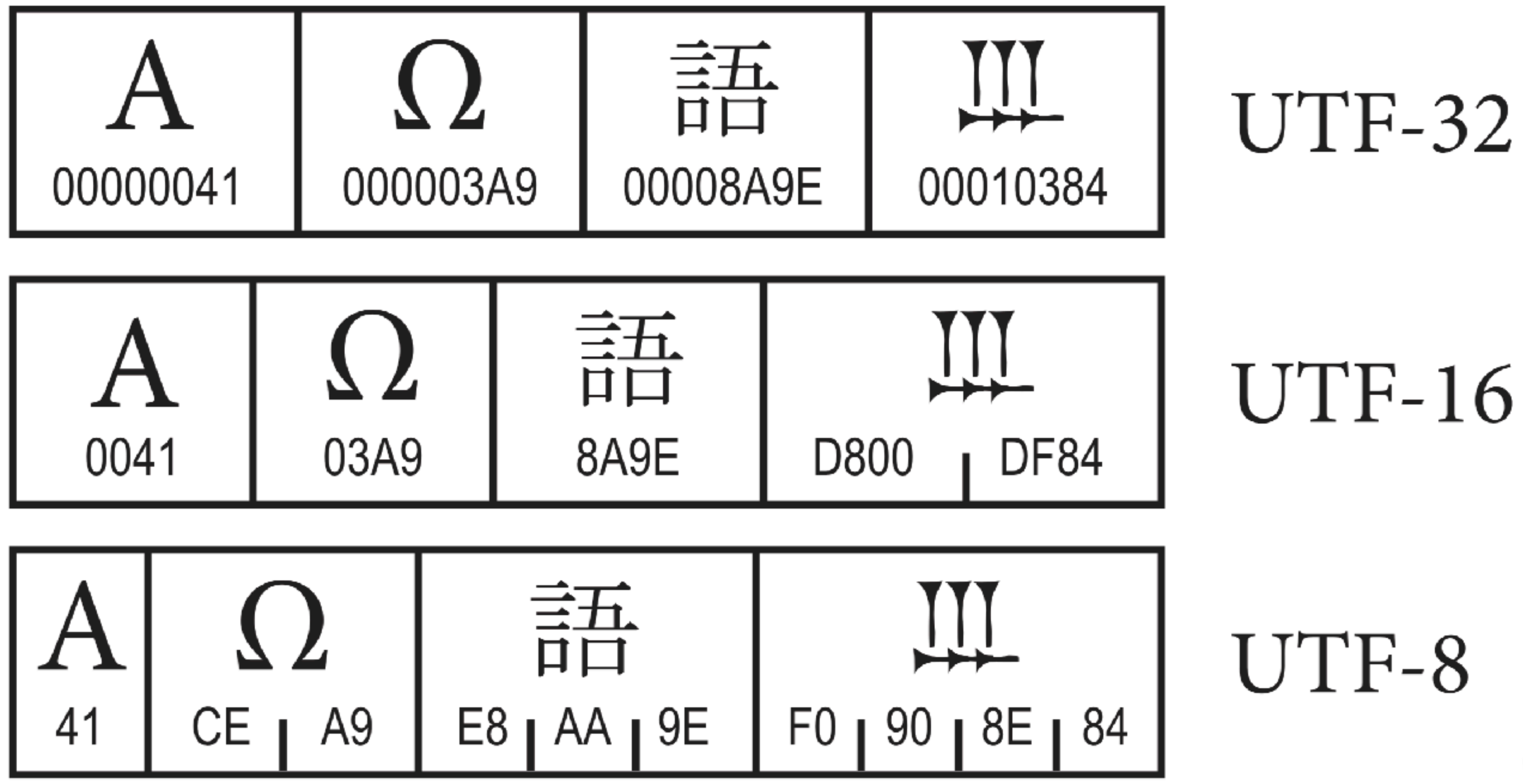
Typography is a vast and complex domain with many rules.
You might consider to enforce some basic rules.

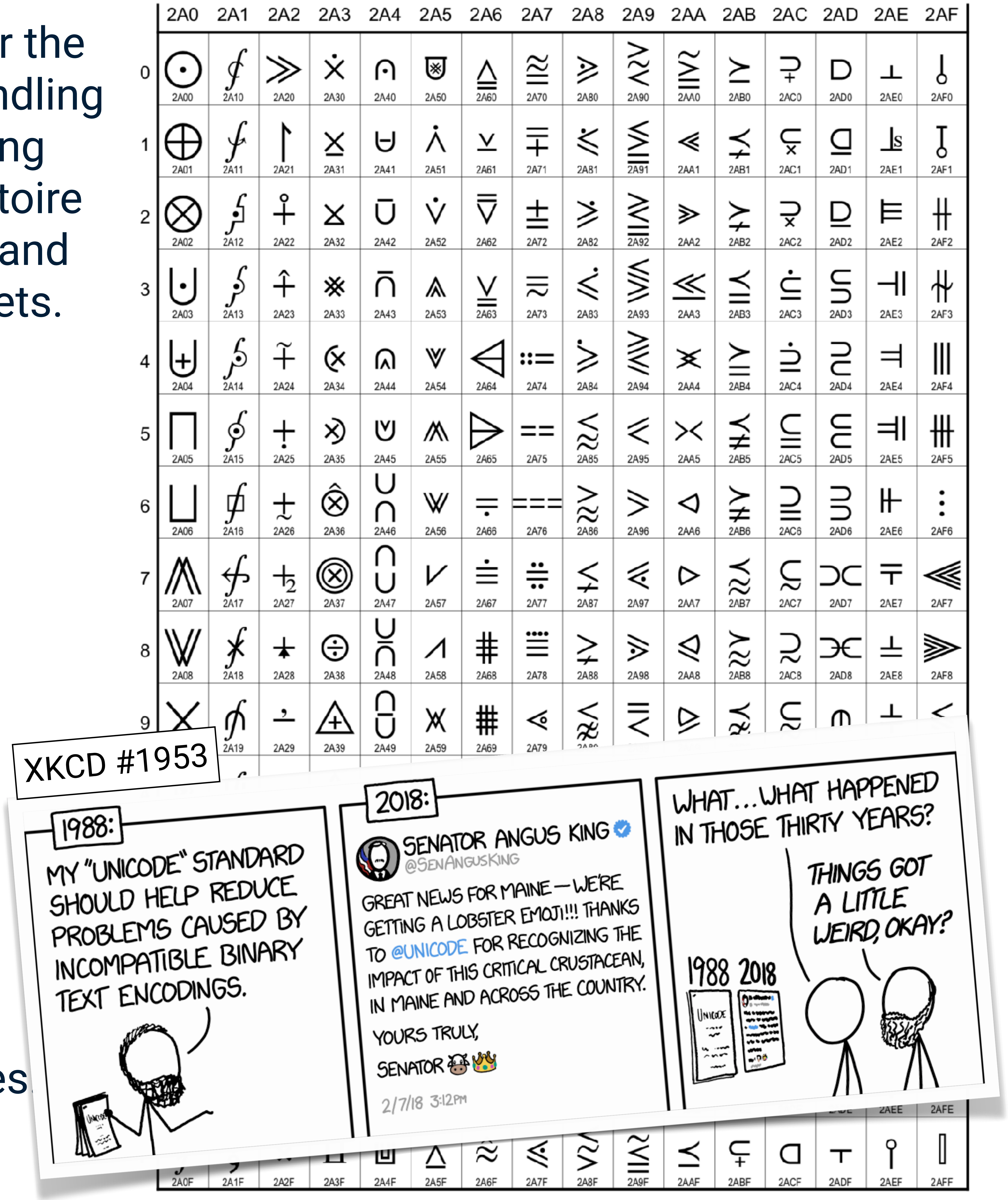


The Anatomy of Typography - Janie Kliever

Unicode

Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems. The latest version contains a repertoire of 136,755 characters covering 139 modern and historic scripts, as well as multiple symbol sets.





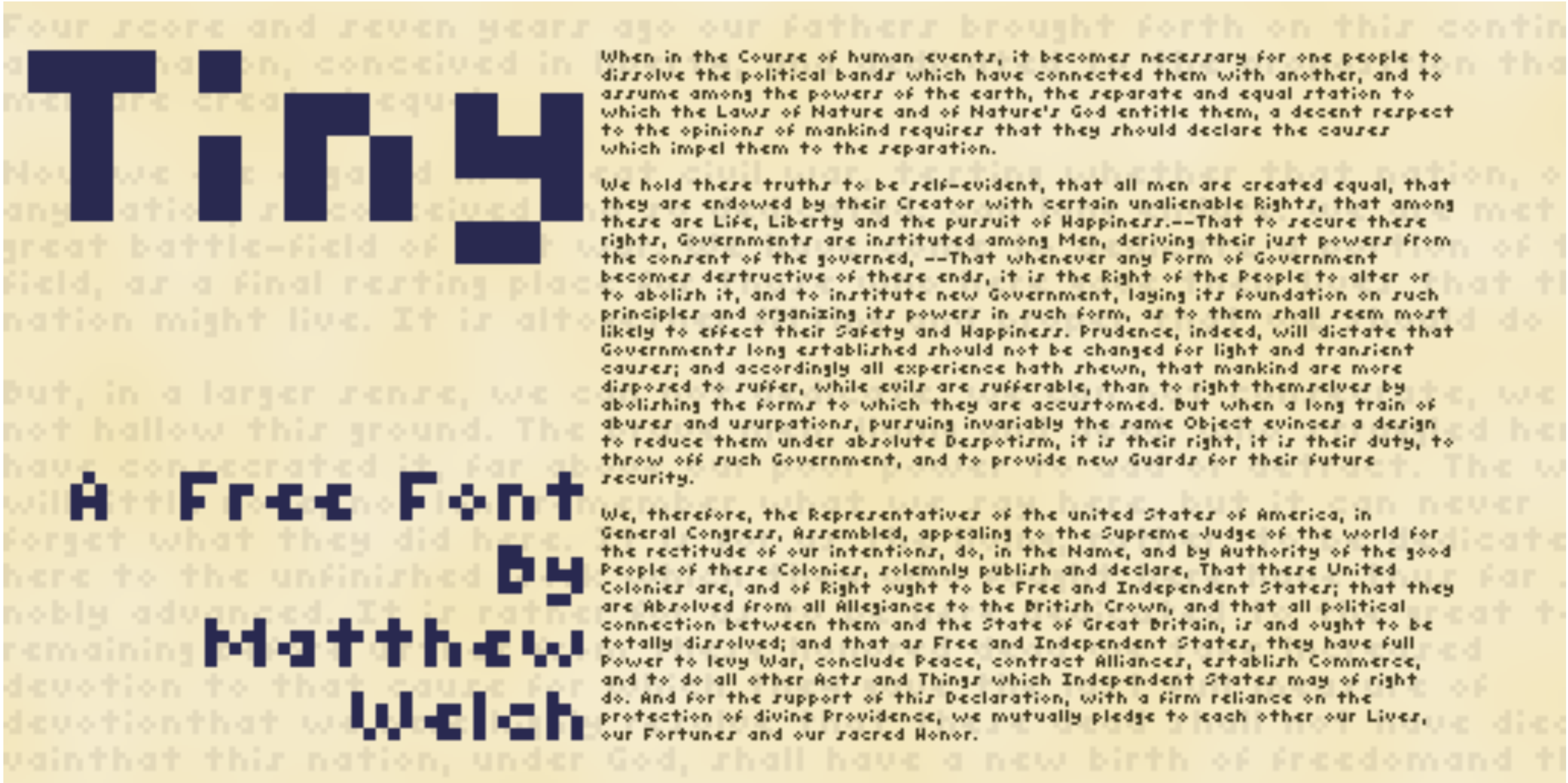
UTF-8 (Unicode Transformation Format, RFC 3629) is a variable width character encoding capable of encoding all 1,112,064 valid code points in Unicode using one to four 8-bit bytes.

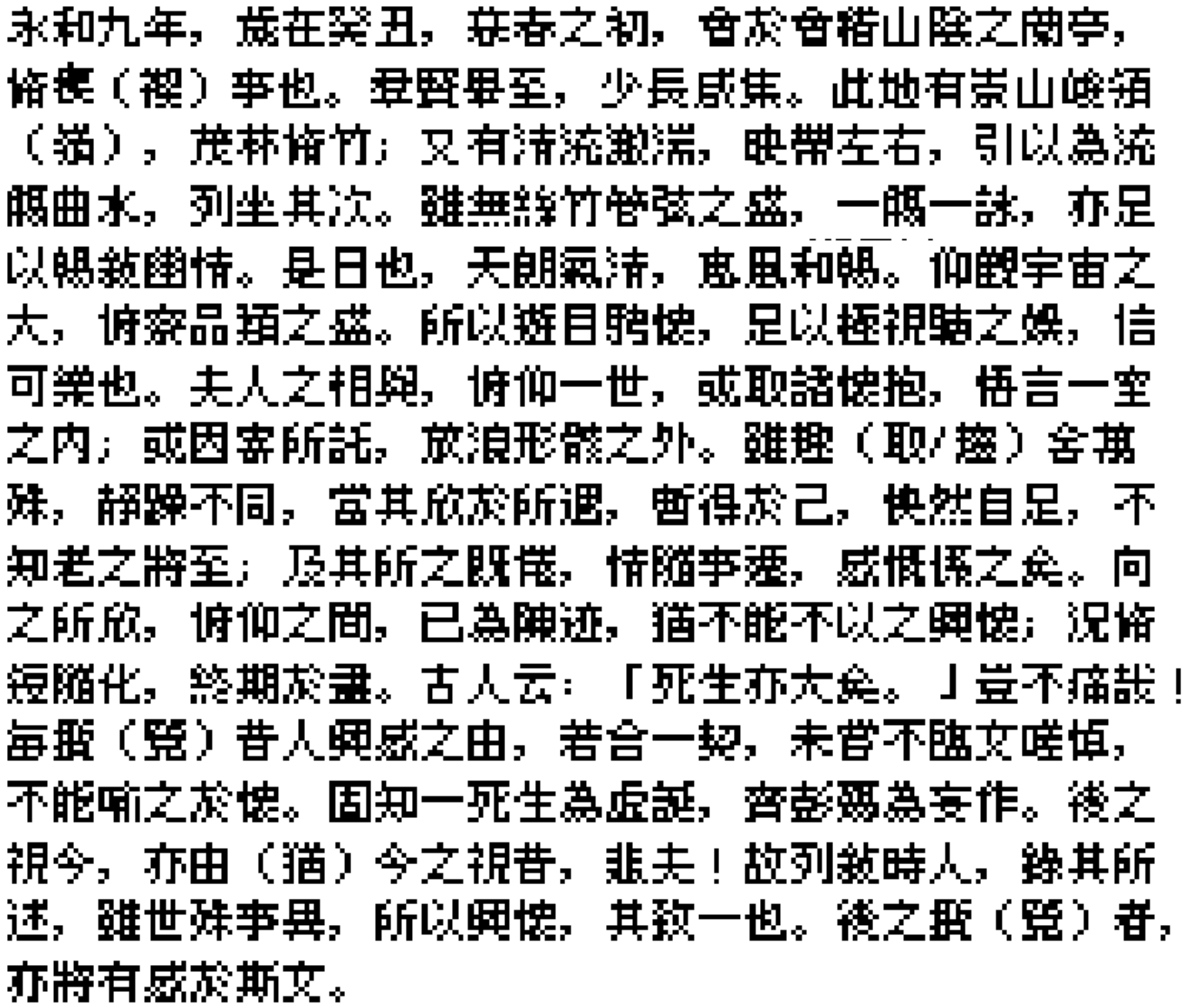
Famous fonts

One font to rule'em all

When text is rendered by a computer, sometimes characters are displayed as "tofu". They are little boxes to indicate your device doesn't have a font to display the text. Google has been developing a font family called Noto, which aims to support all languages with a harmonious look and feel. Noto is Google's answer to tofu.

ITSY BITSY BITMAP FONTS

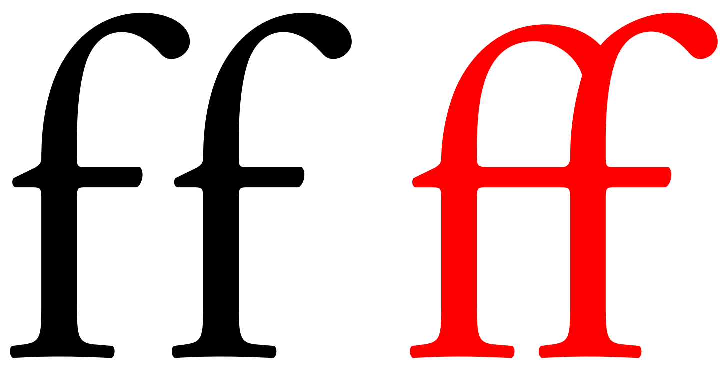




GENERATIONS
SIGGRAPH2018

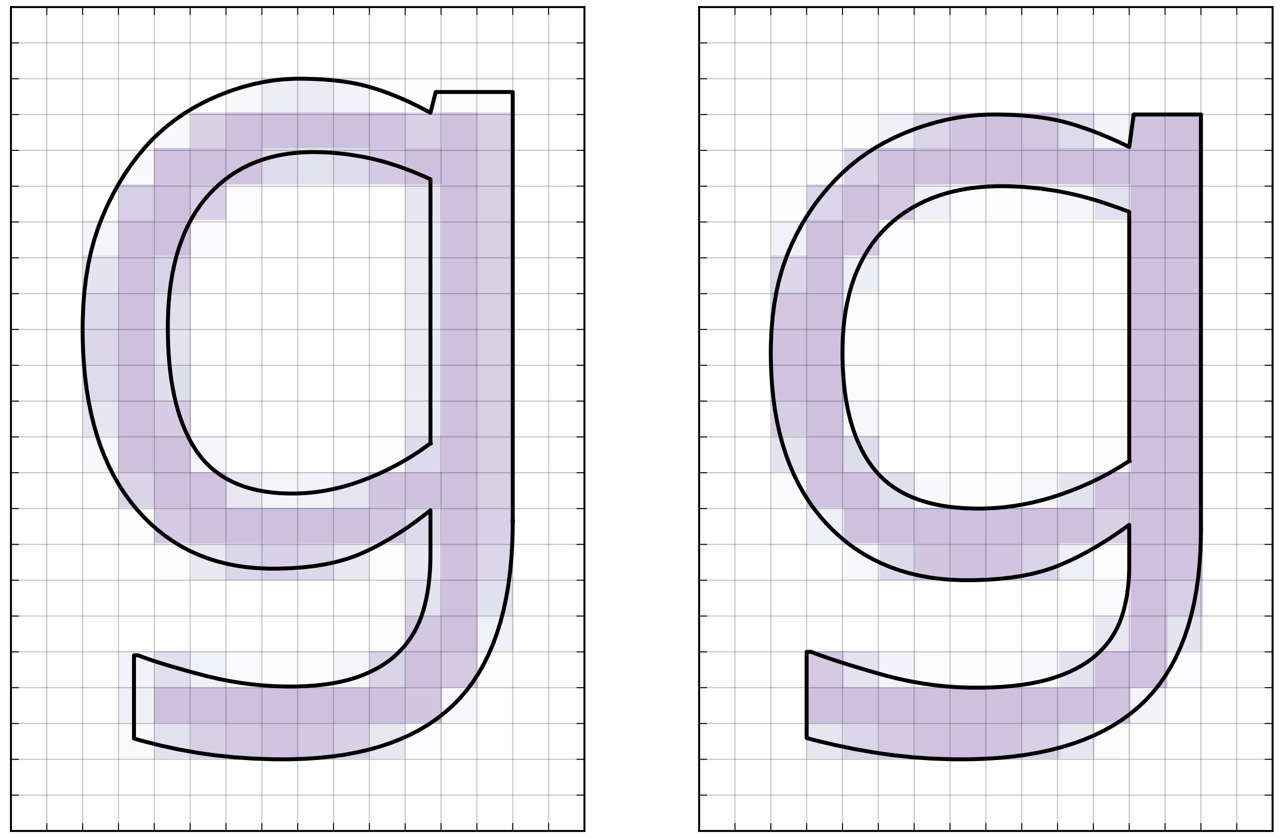
BASIC TYPOGRAPHY

Kerning is the process of adjusting the spacing between characters in a proportional font.



A ligature occurs where two or more graphemes or letters are joined as a single glyph.

A diacritic mark is a glyph added to a letter, or basic glyph.



Hinting is the use of instructions to adjust the display of a font so that it lines up with a rasterized grid.

A various number of face metrics are defined for all glyphs in a given font.

ADVANCED TYPOGRAPHY

العربية العربية

Text shaping is the process of converting Unicode text to glyph indices and positions.

द ् ध ् र ् य

द्ध्र्य

Complex text layout (CTL) refers to the typesetting of writing systems in which the shape or positioning of a grapheme depends on its relation to other graphemes (wikipedia).

Zapfino Zapfino

Stylistic alternate allows to replace a glyph by some variant.

The title is مفتاح معايير الويب in Arabic.

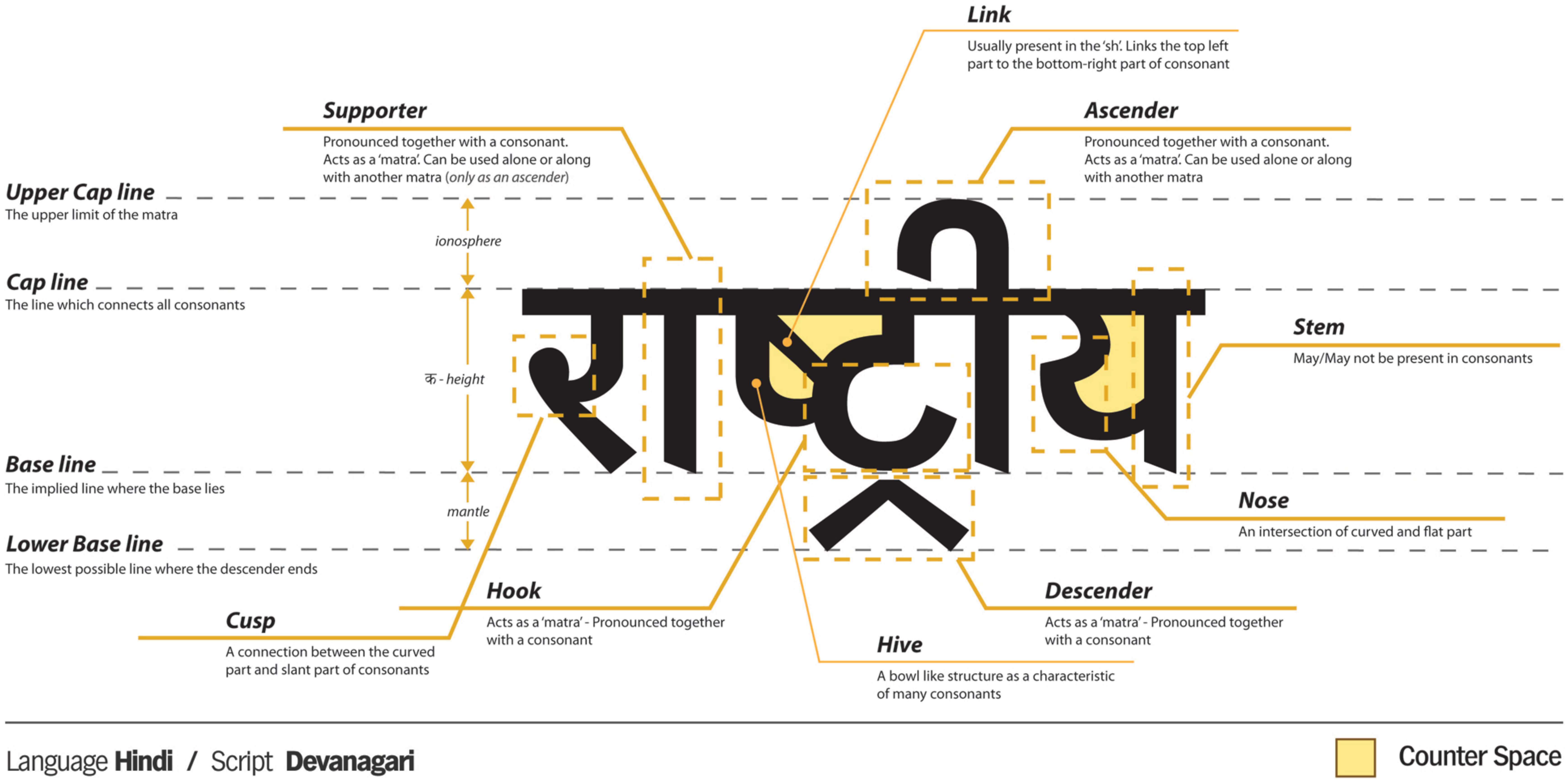
The set of rules applied to produce the correct order at the time of display are described by the Unicode Bidirectional Algorithm.

Etc.

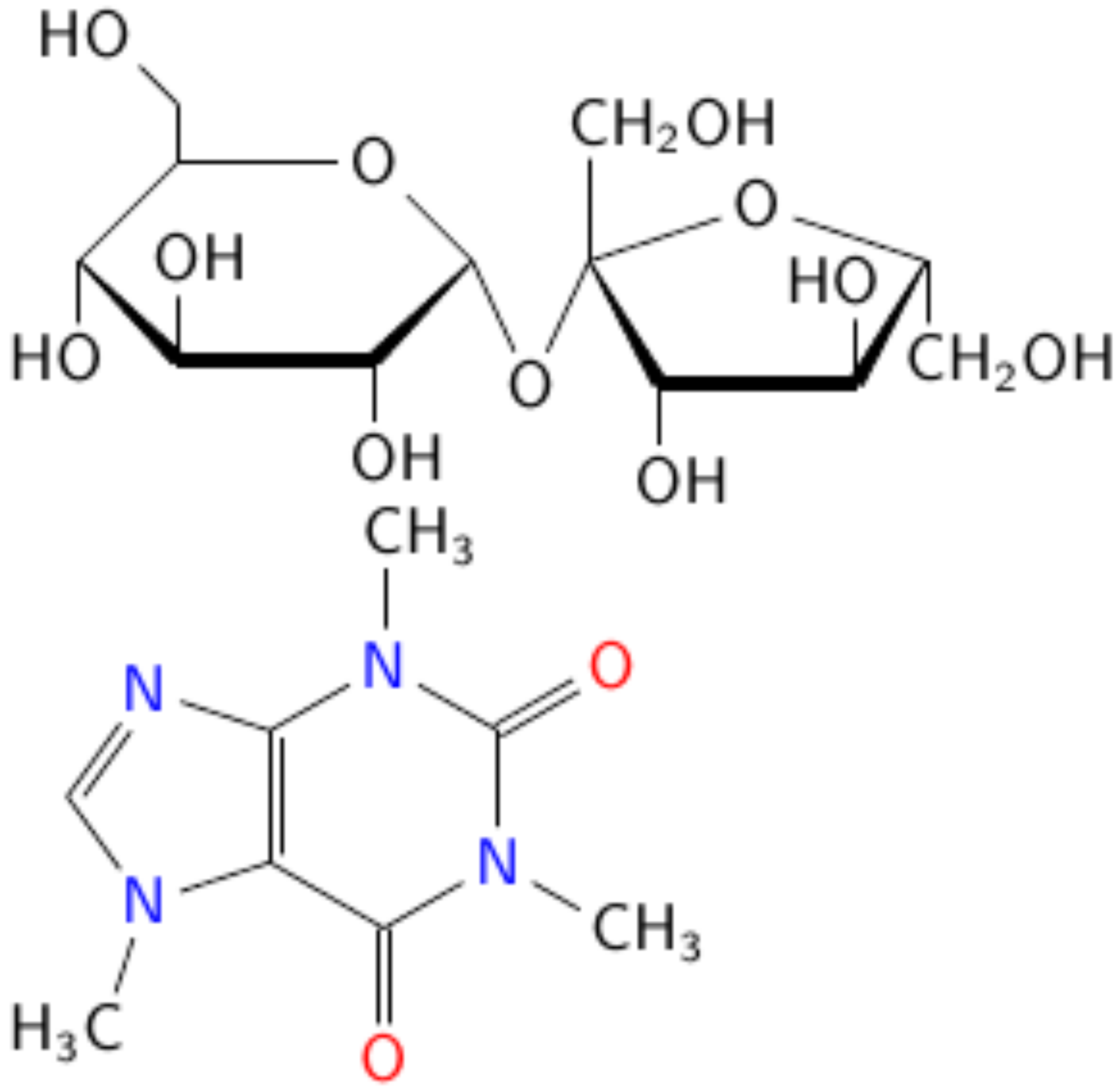
GENERATIONS
SIGGRAPH2018

Advanced Typography

Anatomy of the Hindi Font by Aditya Dipankar.



Other forms of complex layout



$$\sum_{n=1}^{k} \frac{1}{n} > \int_{1}^{k+1} \frac{1}{x} dx = \ln(k+1)$$

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4}$$

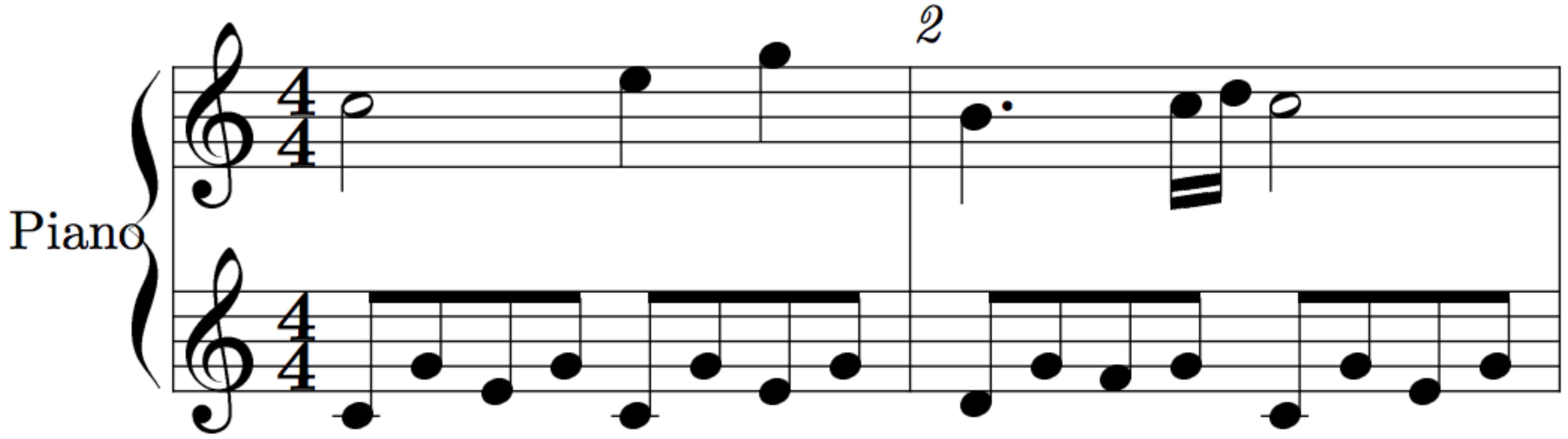
$$\int_{a}^{b} f(x)dx = F(b) - F(a)$$

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

$$\overline{(z/w)} = \bar{z}/\bar{w}$$

$$r = |z| = \sqrt{x^2 + y^2}$$

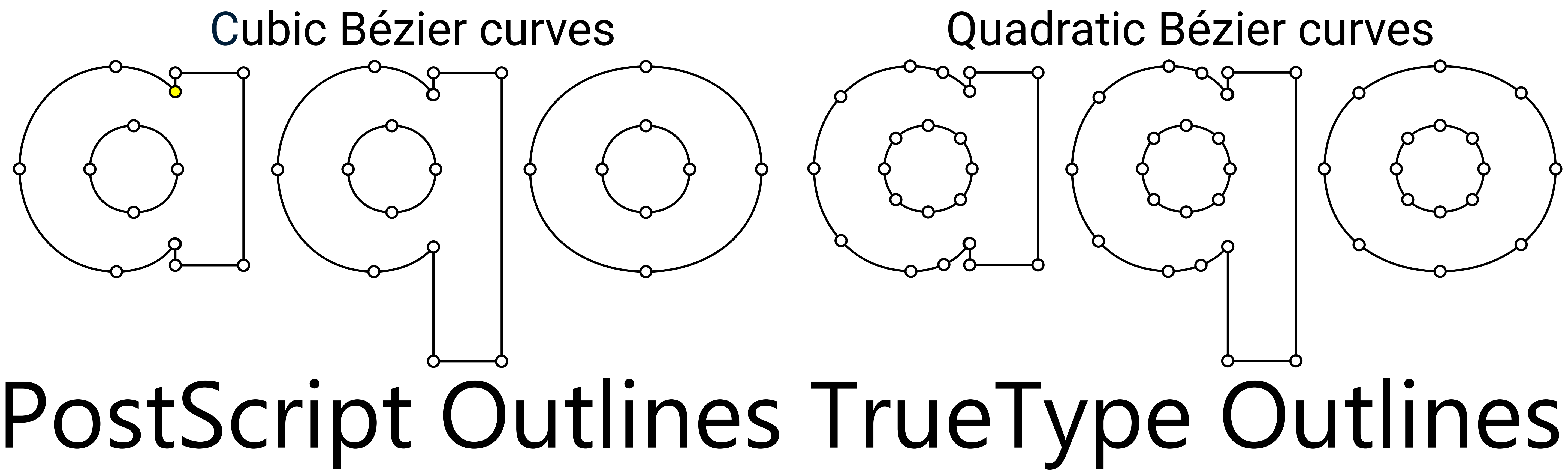
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 3 \end{bmatrix}$$



Made with Latex (what else?)

FONT FORMATS

Types
- Type 1(.pfm, .pfb): glyphs are described with cubic Bézier curves
- True Type (.ttf): glyphs are described with quadratic Bézier curves
- Open Type (.otf): glyphs are described with quadratic or cubic Bézier curves
- Web Open Font (.woff): compressed TrueType or Open Type
- And many more actually…



The Missing Guide to Font Formats

A PRIMER ON BÉZIER CURVES

Pomax (2017) A primer on Bézier curves

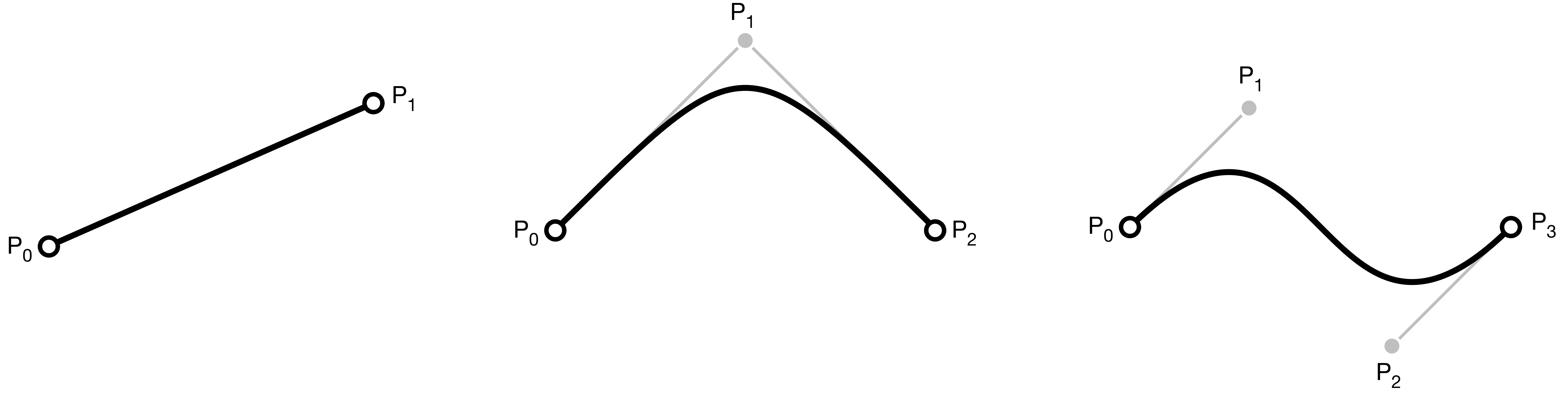
Sergei Natanovich Bernstein (1912) / Paul de Casteljau (1959) / Pierre Bézier (1962)

A Bézier curve is defined by a set of control points P_0 through P_n, where n is called its order. The first and last control points are always the end points of the curve.

Linear (n=1): $B_{P0/P1}(t) = (1-t)P0 + tP1$

Quadratic (n=2): $B_{P0/P1/P2}(t) = (1-t)B_{P0/P1}(t) + tB_{P1/P2}(t)$

Cubic(n=3): $B_{P0/P1/P2/P3}(t) = (1-t)B_{P0/P1/P2}(t) + tB_{P1/P2/P3}(t)$

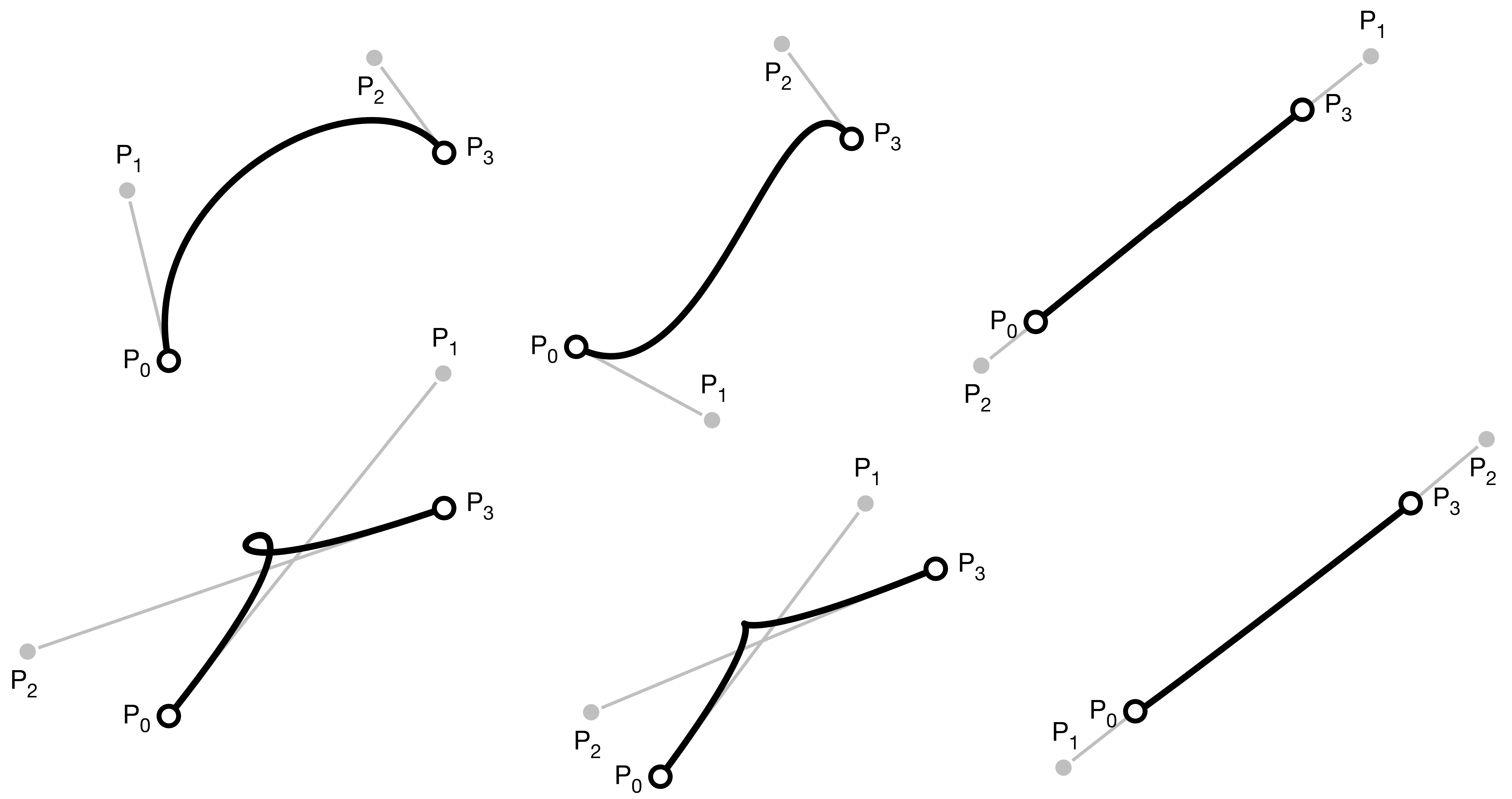


Some related problems: signed distance, thick curves, subdivision, bounding box, linear speed, self-intersection, curve splitting, arc length, approximation of a cubic with quadratics, etc.

A primer on Bézier curves

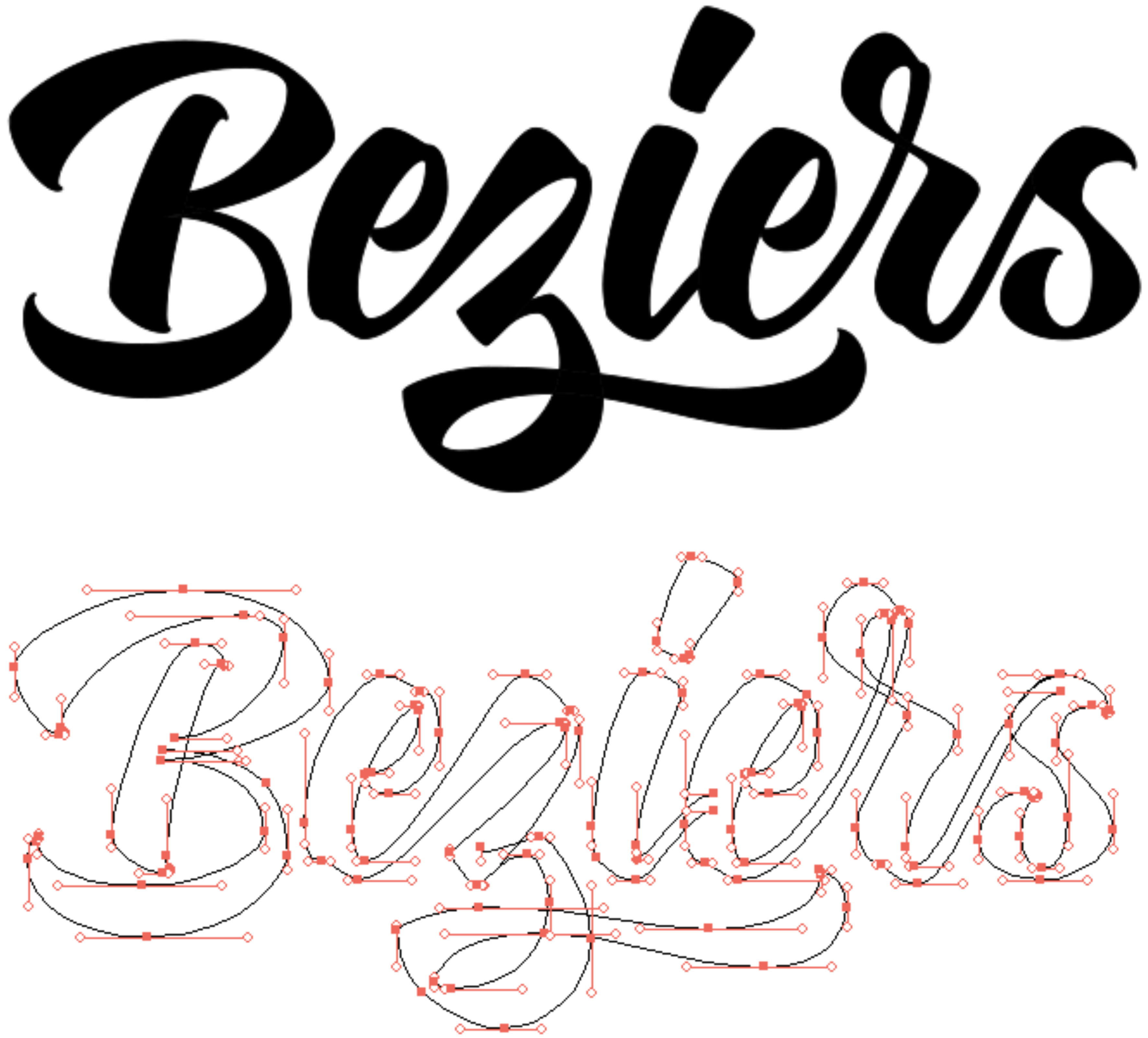
Pomax (2017) A primer on Bézier curves

Cubic Bézier curves have a lot of corner cases: inflection points, cusp, loop, overlap

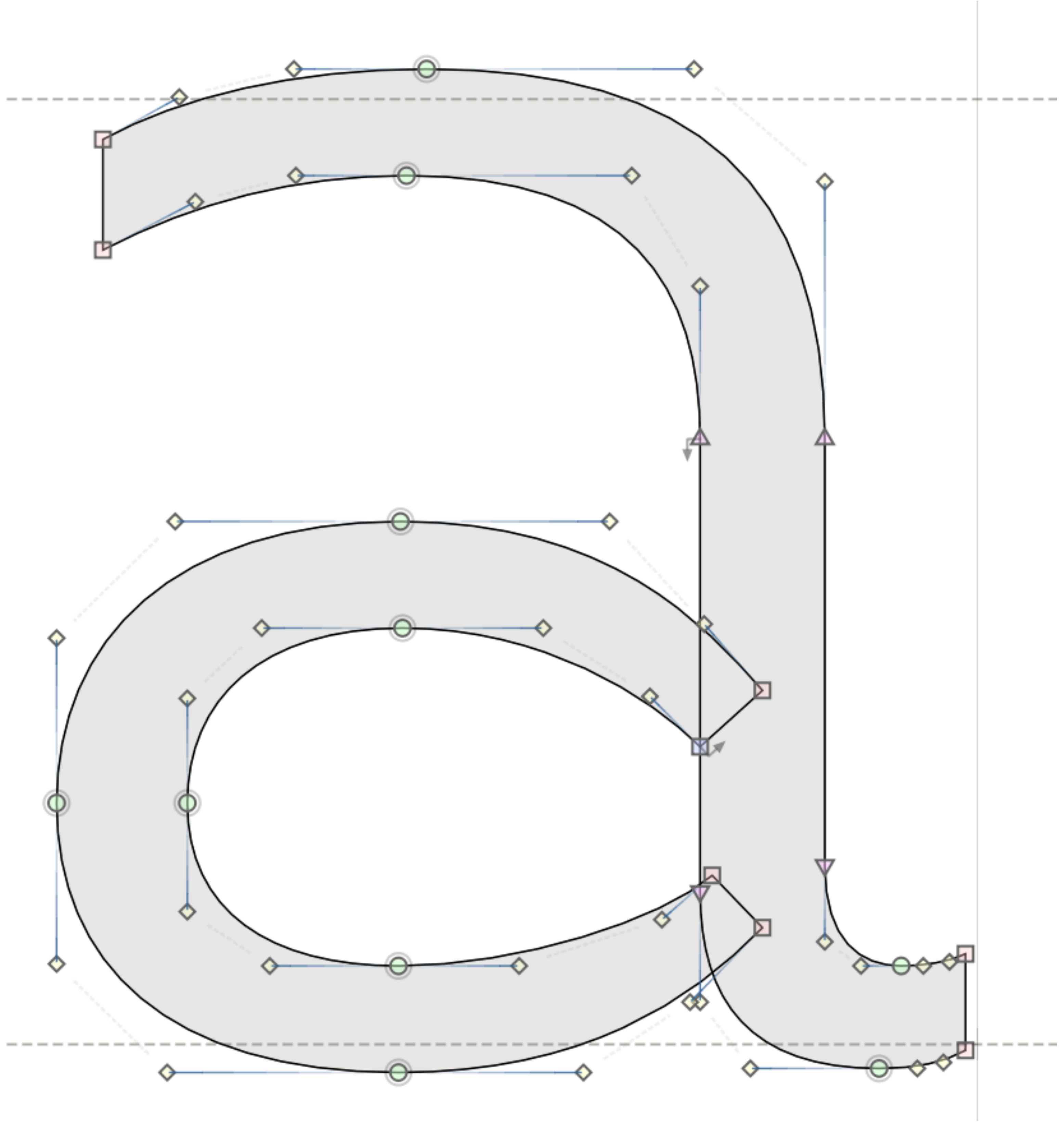


Anatomy of a glyph

Vector fonts are collections of vector images, consisting of lines and curves defining the boundary of glyphs (wikipedia). Type 1 and Type 3 Postscript fonts are described with cubic Bézier curves. Truetype fonts are described with quadratic Bézier curves.



Images by The Australian Graphic Supply Co



Images by FontLab

Font tables

Open Type Tables

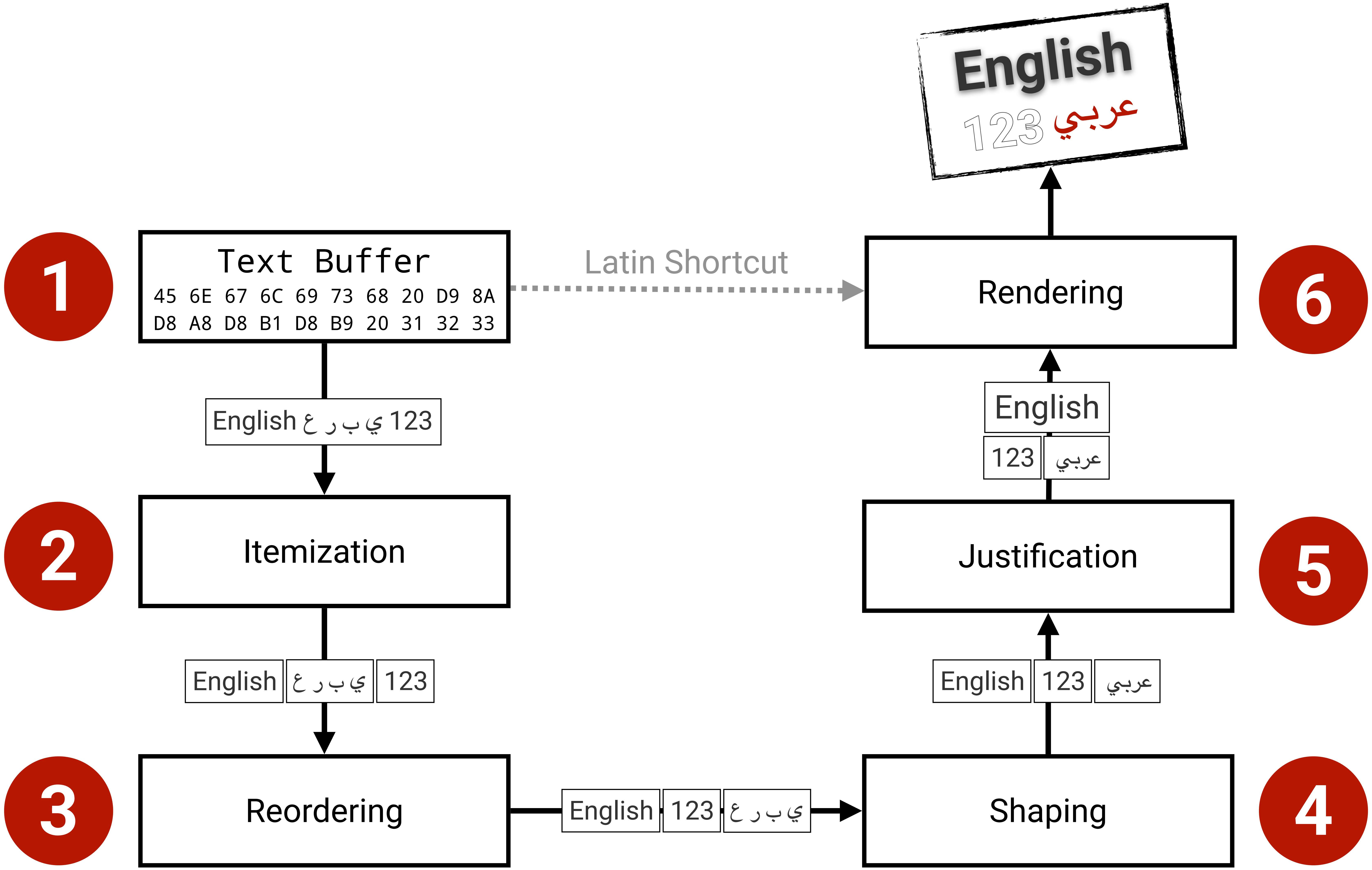
- BASE: Baseline data
- CMAP: Character to glyph mapping
- GDEF: Glyph definition data
- GSUB: Glyph substitution data
  - Single: Replaces one glyph with one glyph.
  - Multiple: Replaces one glyph with more than one glyph.
  - Alternate: Replaces one glyph with one of many glyphs.
  - Ligature: Replaces multiple glyphs with one glyph.
  - Context: Replaces one or more glyphs in context.
  - Chaining: Replaces one or more glyphs in chained context.
- GPOS: Glyph positioning data
- JSTF: Justification data

and many more…

Libraries

- STB_truetype (single header file, basic support)
  → https://github.com/nothings/stb
- FreeType (standard support without text shaping)
  → https://www.freetype.org
- HarfBuzz (advanced support with text shaping)
  → https://harfbuzz.github.io

TEXT RENDERING PIPELINE



GENERATIONS SIGGRAPH2018

Part I

Near the end of the last century (1997), Mark Kilgard introduced a simple OpenGL-based API for texture mapped text.

The method packed many rasterized glyphs into a single alpha-only texture map and used a lookup table to assign texture coordinates to a quadrilateral to extract a glyph when rendering.

Image by Daniel Ullrich
commons.wikimedia.org/wiki/User:Threedots

GENERATIONS
SIGGRAPH2018

2D RASTERIZATION

Same input, different outputs

コンピュータによる CJKV（中国語、日本語、韓国語、ベトナム語）テキスト処理は、多くの謎や複雑な仕組み（intrigue）に包まれている。そこにはたしかにたくらみ（intrigue）があることは認めるが、それよりも謎の多いことのほうが問題である。

Lorem ipsum dolor sit amet, consectetuer adipisci elit. Cras sit amet dui. Nam sapien. Fusce vestibulum ornare metus. Maecenas ligula orci, consequat vitae, dictum nec, lacinia non, elit. Aliquam iaculis molestie neque. Maecenas suscipit felis ut pede convallis malesuada. Aliquam erat volutpat. Nunc pulvinar condimentum nunc. Donec ac sem vel leo bibendum aliquam. Pellentesque habitant

FreeType Native CFF Rasterizer

コンピュータによる CJKV（中国語、日本語、韓国語、ベトナム語）テキスト処理は、多くの謎や複雑な仕組み（intrigue）に包まれている。そこにはたしかにたくらみ（intrigue）があることは認めるが、それよりも謎の多いことのほうが問題である。

Lorem ipsum dolor sit amet, consectetuer adipisci elit. Cras sit amet dui. Nam sapien. Fusce vestibulum ornare metus. Maecenas ligula orci, consequat vitae, dictum nec, lacinia non, elit. Aliquam iaculis molestie neque. Maecenas suscipit felis ut pede convallis malesuada. Aliquam erat volutpat. Nunc pulvinar condimentum nunc. Donec ac sem vel leo bibendum aliquam. Pellentesque habitant

FreeType Light Auto Hint Rasterizer

コンピュータによる CJKV（中国語、日本語、韓国語、ベトナム語）テキスト処理は、多くの謎や複雑な仕組み（intrigue）に包まれている。そこにはたしかにたくらみ（intrigue）があることは認めるが、それよりも謎の多いことのほうが問題である。

Lorem ipsum dolor sit amet, consectetuer adipisci elit. Cras sit amet dui. Nam sapien. Fusce vestibulum ornare metus. Maecenas ligula orci, consequat vitae, dictum nec, lacinia non, elit. Aliquam iaculis molestie neque. Maecenas suscipit felis ut pede convallis malesuada. Aliquam erat volutpat. Nunc pulvinar condimentum nunc. Donec ac sem vel leo bibendum aliquam. Pellentesque habitant

FreeType using the New Adobe CFF Rasterizer (2013)

Quartz Rendering

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis varius elementum. Proin cursus posuere mollis. Vivamus id justo nec massa convallis dapibus. Praesent a arcu augue, pretium commodo elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

ClearType Rendering

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis facilisis varius elementum. Proin cursus posuere mollis. Vivamus id justo nec massa convallis dapibus. Praesent a arcu augue, pretium commodo elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

GENERATIONS
SIGGRAPH2018

2D Rasterization

GDI rendering of FacitWeb (above) and Minion Pro (below) with no antialiasing.
(Source Adobe Typekit Blog)

2D RASTERIZATION

GDI rendering of FacitWeb (above) and Minion Pro (below) with standard antialiasing.
(Source Adobe Typekit Blog)

GENERATIONS SIGGRAPH2018

2D RASTERIZATION

Core text rendering of FacitWeb (above) and Minion Pro (below).
(Source Adobe Typekit Blog)

GENERATIONS SIGGRAPH2018

2D Rasterization

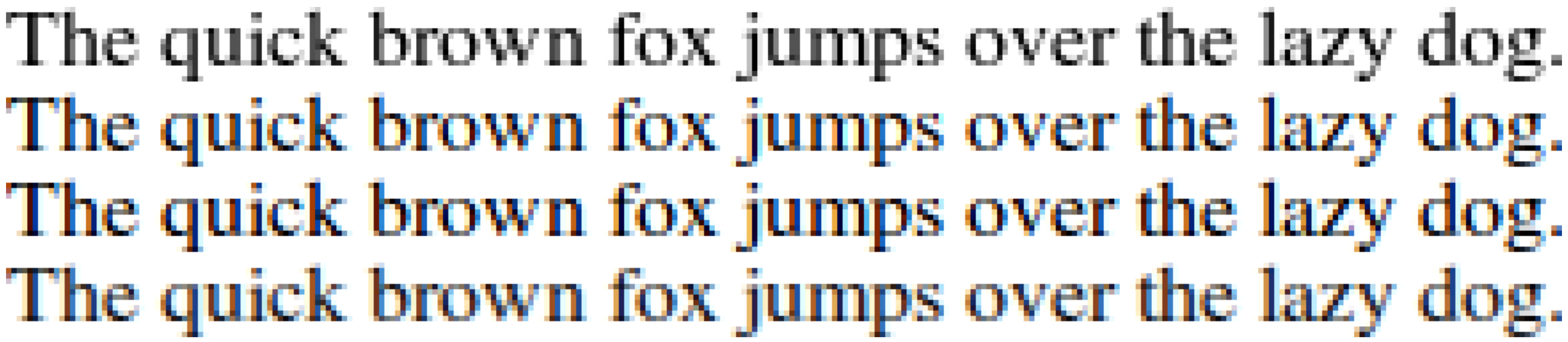
DirectWrite rendering of FacitWeb (above) and Minion Pro (below).
(Source Adobe Typekit Blog)

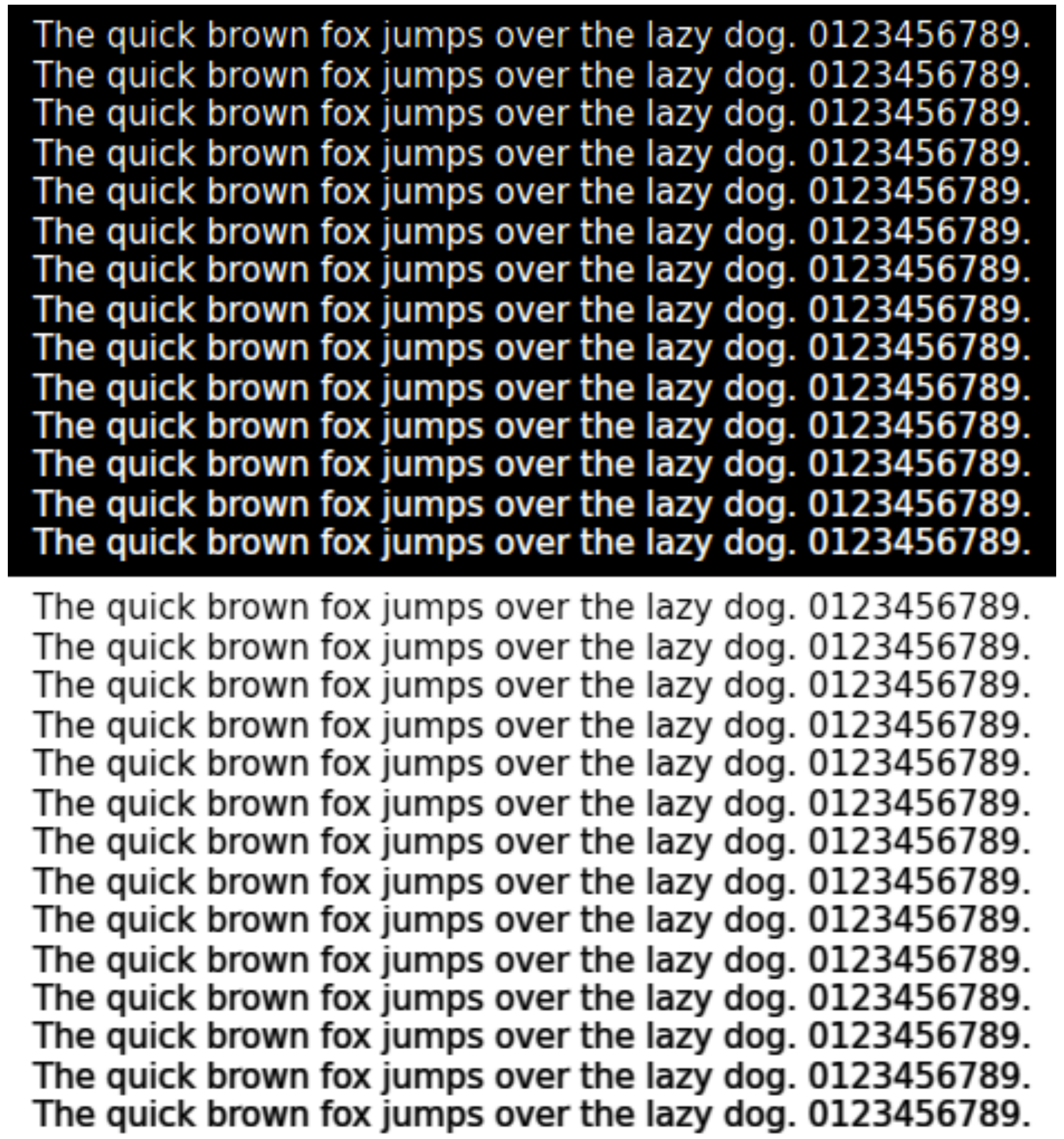
2D RASTERIZATION

GDI rendering of FacitWeb (above) and Minion Pro (below) with ClearType enabled.
(Source Adobe Typekit Blog)

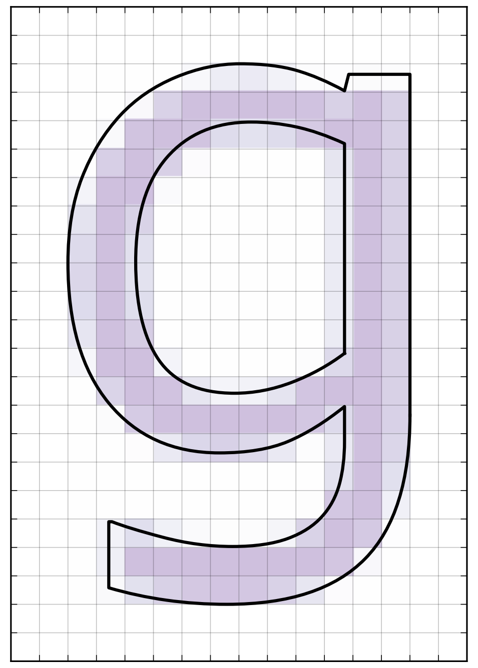
2D Rasterization

Gamma correction, hinting, energy distribution, etc.

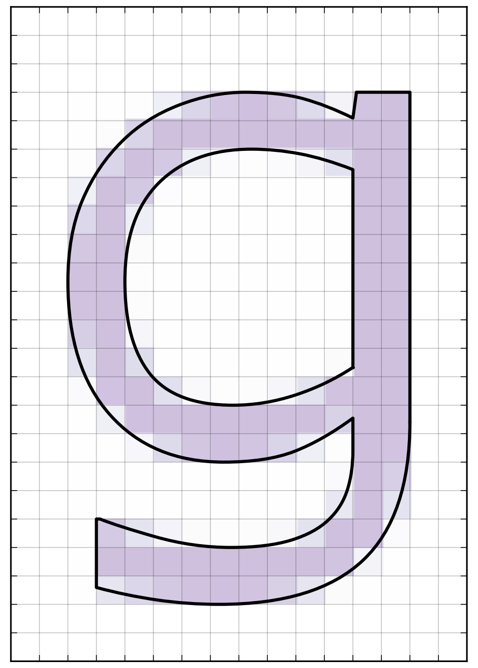


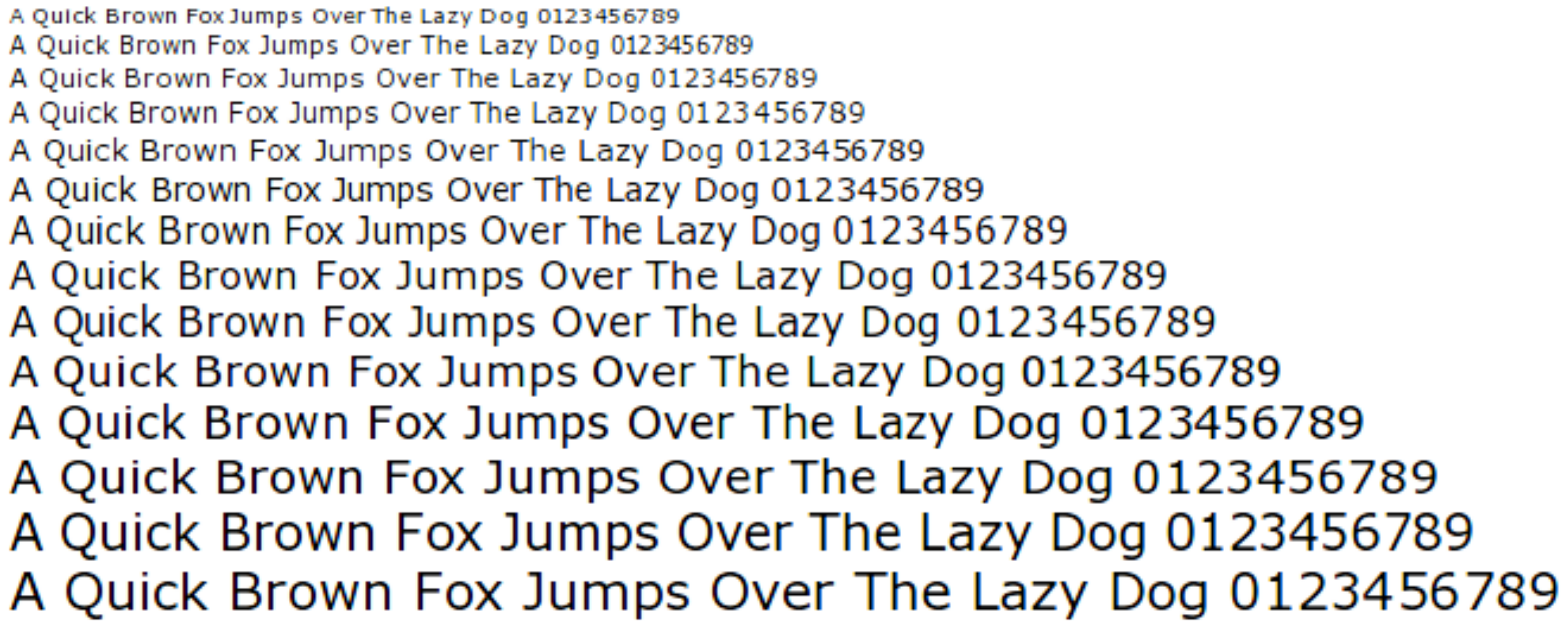


No hinting



Native hinting

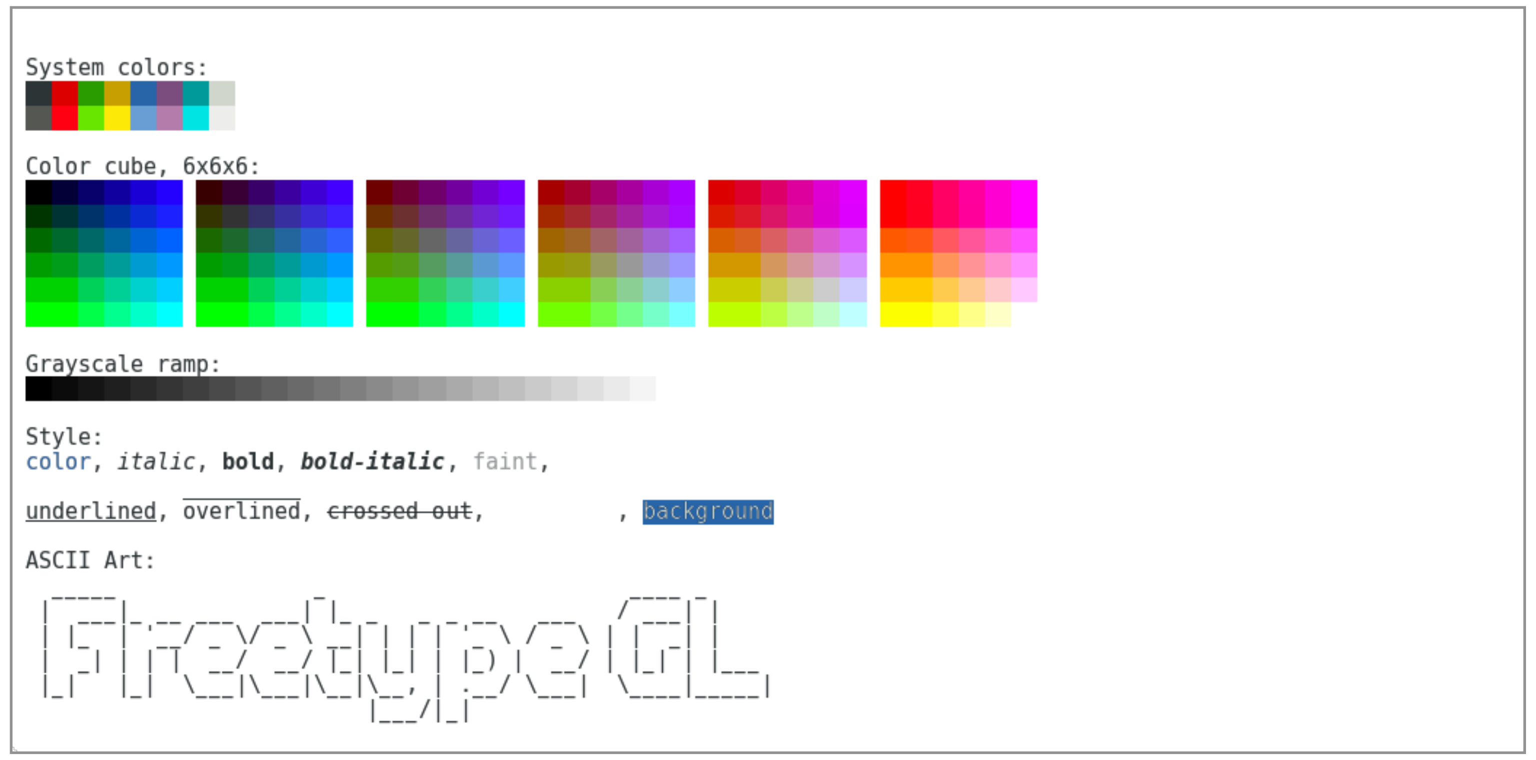




TEXTURE FONT I

Kilgard (1997) A Simple OpenGL-based API for Texture Mapped Text

Texture mapping is well suited for rendering text because textures can be rendered quickly with current 3D hardware and even via clever programming of today's fast CPUs. Textures can be stretched. rotated, scaled, and even projected (assuming the texture mapping is perspective correct) so that texture mapped text looks reasonable in 3D scenes.

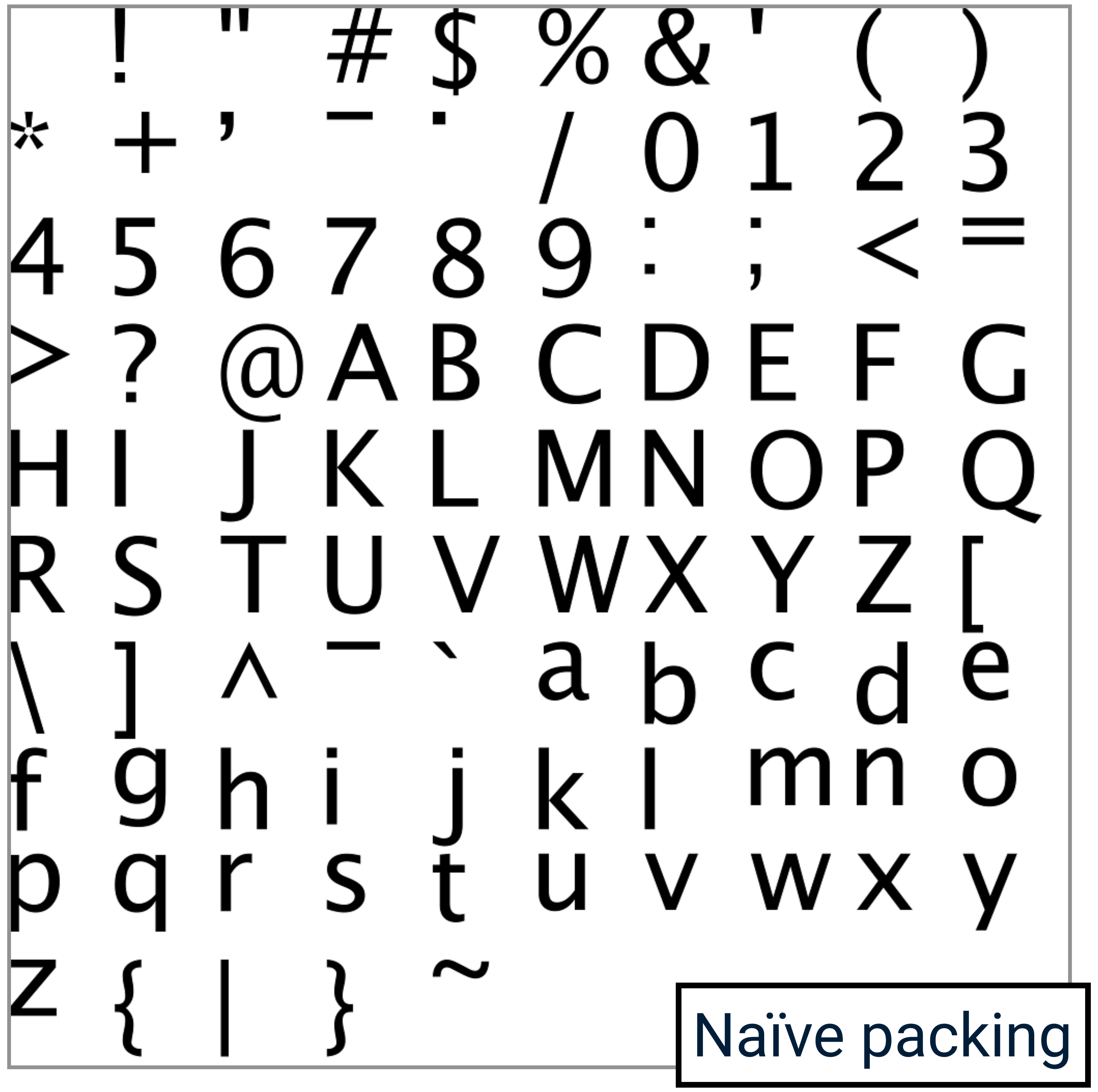


https://github.com/rougier/freetype-gl

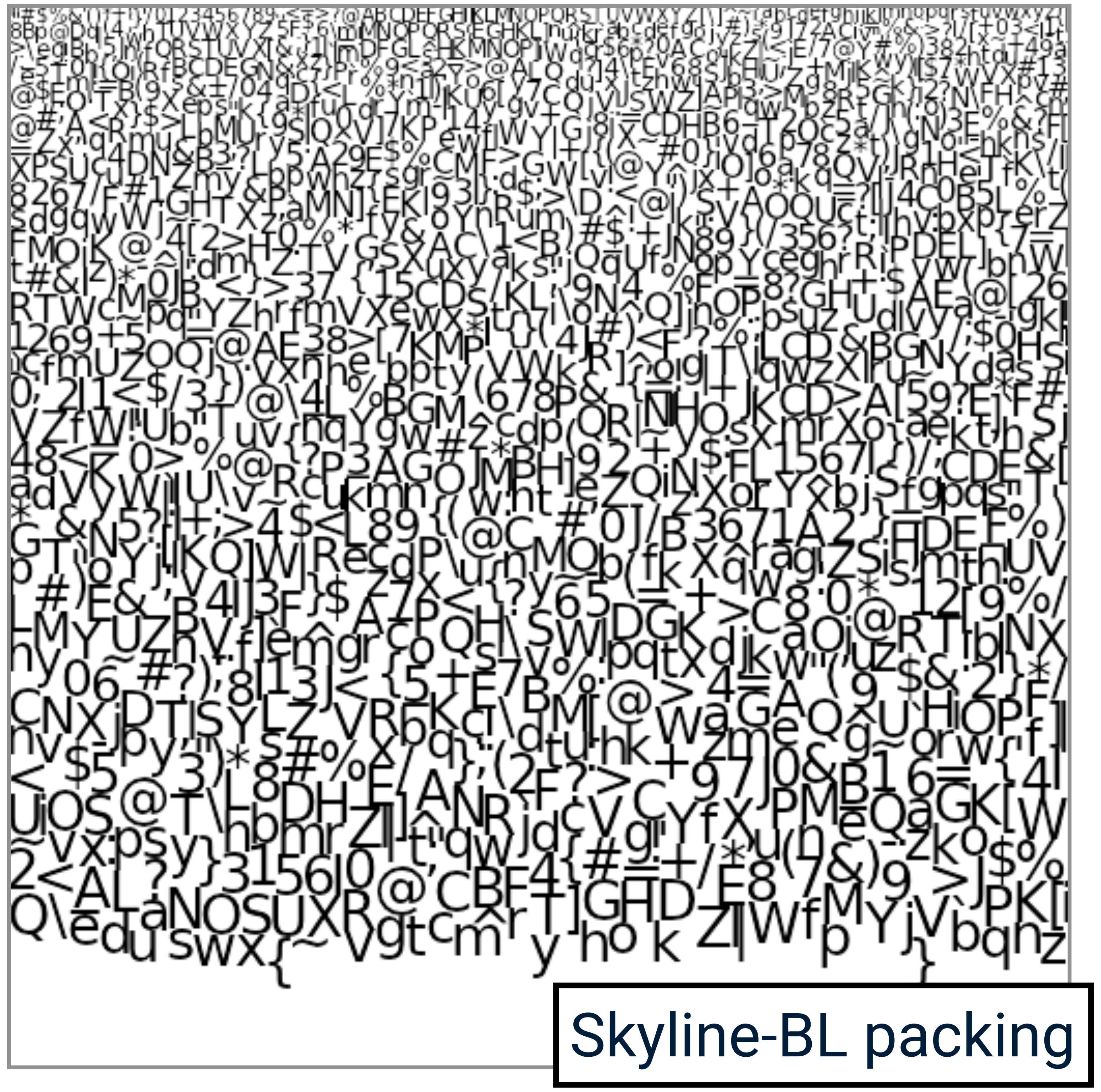
Bin packing algorithms

Jylänky (2010) A Thousand Ways to Pack the Bin

Several algorithms that can be used to solve the problem of packing rectangles into two-dimensional finite bins. Most of the algorithms have well been studied in literature, but some of the variants are less known and some are apparently regarded as "folklore" and no previous reference is known. Different variants are presented and compared.



Naïve packing

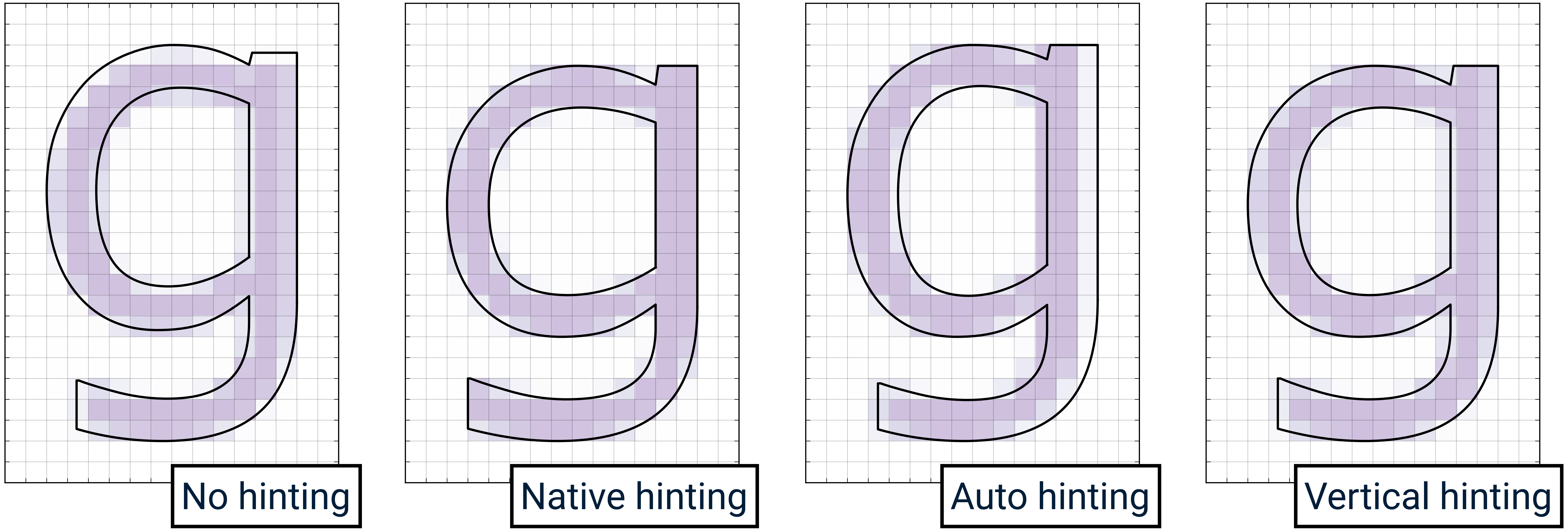


Skyline-BL packing

TEXTURE FONT II

Rougier (2013) Higher Quality 2D Text Rendering

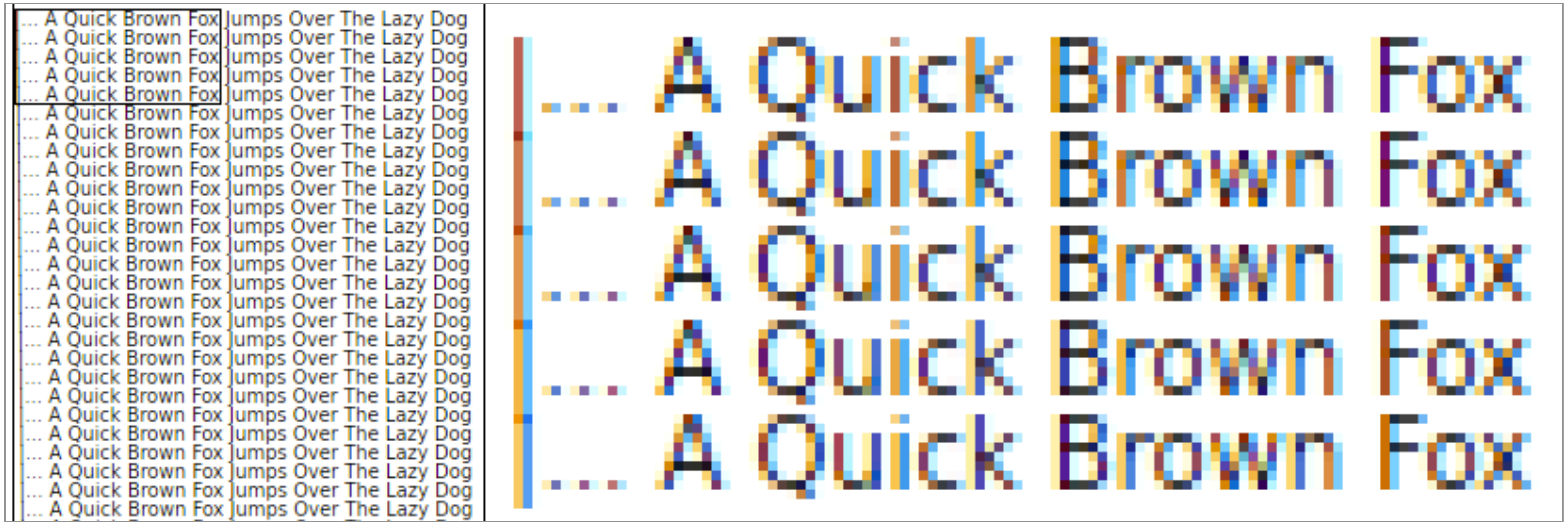
When displaying text on low-resolution devices (DPI < 150), one typically has to decide if one wants to respect the pixel grid (e.g., Cleartype technology / Microsoft / native hinting) for crisp rendering or, to privilege glyph shapes (Quartz technology / Apple / no hinting) at the cost of blurring. There is, however, a third way that may combine the best of the two technologies (vertical hinting).



TEXTURE FONT II

Rougier (2013) Higher Quality 2D Text Rendering

i. Use horizontal RGB sub-pixel anti-aliasing for LCD flat panels.
ii. Use vertical hinting only and completely discard the horizontal one.
iii. Use accurate glyph advance values from unhinted glyph
iv. Use accurate, high resolution values from the kerning table.



GENERATIONS SIGGRAPH2018

TEXTURE FONT II

Rougier (2013) Higher Quality 2D Text Rendering

Subpixel rendering on the CPU
Subpixel positioning on the GPU

$0 < s \leq 1/3,\ t = 3s$	$R_{out} = tB_{left} + (1-t)R$ $G_{out} = tR + (1-t)G$ $B_{out} = tG + (1-t)B$
$1/3 < s \leq 2/3,\ t = 3s-1$	$R_{out} = tG_{left} + (1-t)B_{left}$ $G_{out} = tB_{left} + (1-t)R$ $B_{out} = tR + (1-t)G$
$2/3 < s < 1,\ t = 3s-2$	$R_{out} = tR_{left} + (1-t)G_{left}$ $G_{out} = tG_{left} + (1-t)B_{left}$ $B_{out} = tB_{left} + (1-t)R$

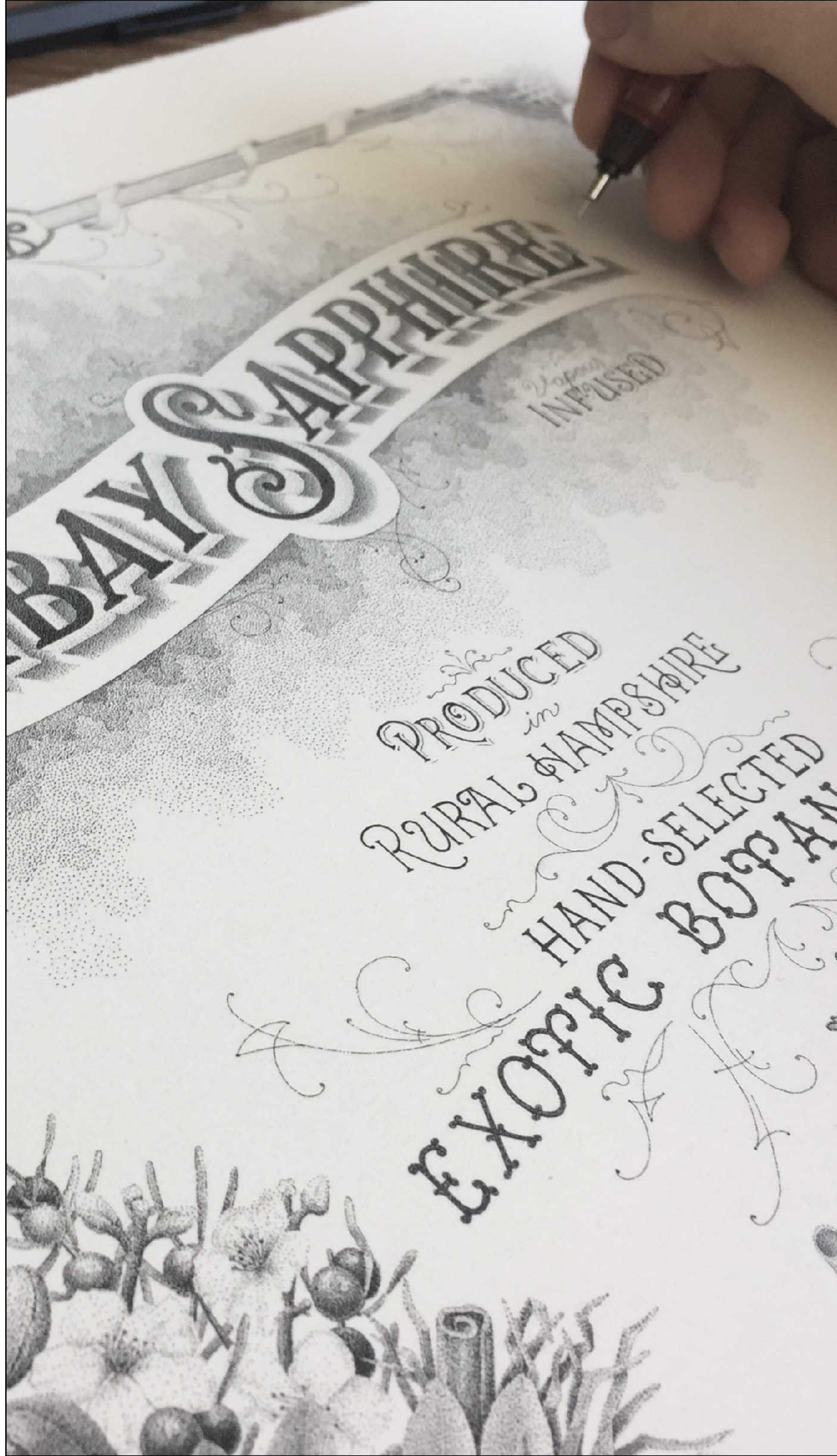
```
uniform sampler2D texture;
uniform vec2 pixel;
varying float shift;
void main()
{
    vec2 uv       = gl_TexCoord[0].xy;
    vec4 current  = texture2D(texture, uv);
    vec4 previous = texture2D(texture, uv+vec2(-1,0)*pixel);
    float r = current.r;
    float g = current.g;
    float b = current.b;
    float a = current.a;
    if( shift <= 1.0/3.0 )
    {
        float z = 3.0*shift;
        r = mix(current.r, previous.b, z);
        g = mix(current.g, current.r,  z);
        b = mix(current.b, current.g,  z);
    }
    else if( shift <= 2.0/3.0 )
    {
        float z = 3.0*shift-1.0;
        r = mix(previous.b, previous.g, z);
        g = mix(current.r,  previous.b, z);
        b = mix(current.g,  current.r,  z);
    }
    else if( shift < 1.0 )
    {
        float z = 3.0*shift-2.0;
        r = mix(previous.g, previous.r, z);
        g = mix(previous.b, previous.g, z);
        b = mix(current.r,  previous.b, z);
    }
    gl_FragColor = vec3(r,g,b,a);
}
```

N. ROUGIER (INRIA) & B. ESFAHBOD (GOOGLE)

PART II

Distance based rendering takes advantage of a signed distance function (that can be approximated) to compute the individual coverage for each pixel.

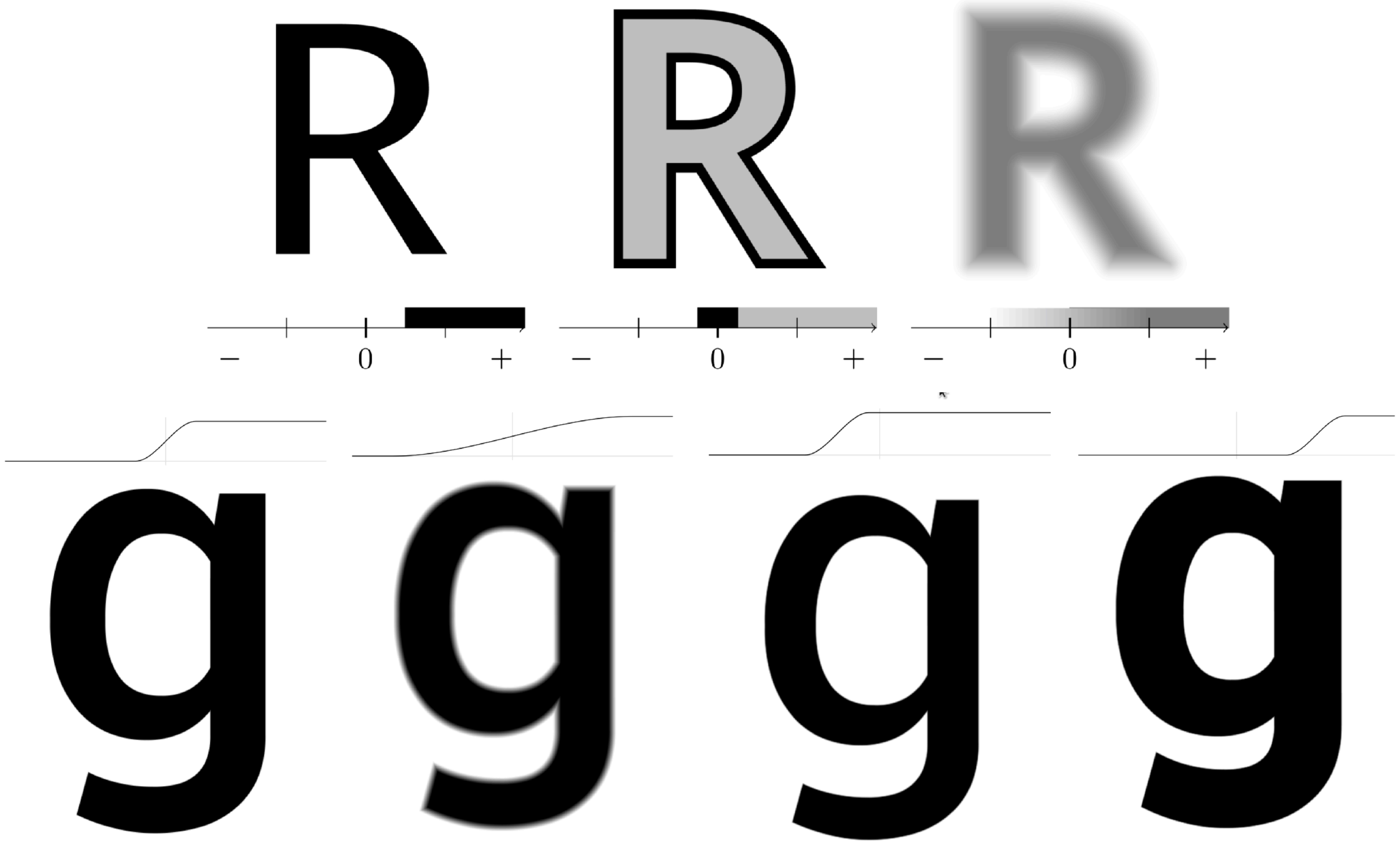
The coverage can be selected to implement various effects (stroke, thinner, thicker, shadow, etc)



Work & image by Xavier Casalta
casaltaxavier.com

Distance based rendering

Distance based rendering takes advantage of a signed distance function (that can be approximated) to compute the individual coverage for each pixel. The coverage can be selected to implement various effects (stroke, thinner, thicker, shadow, etc)



COMPUTING SDF

Gustavson & Strand (2011) Anti-Aliased Euclidean distance transform

We present a modified distance measure for use with distance transforms of anti-aliased, area sampled grayscale images of arbitrary binary contours. The modified measure can be used in any vector-propagation Euclidean distance transform. Our test implementation in the traditional SSED8 algorithm shows a considerable improvement in accuracy and homogeneity of the distance field compared to a traditional binary image transform.

See also http://contourtextures.wikidot.com

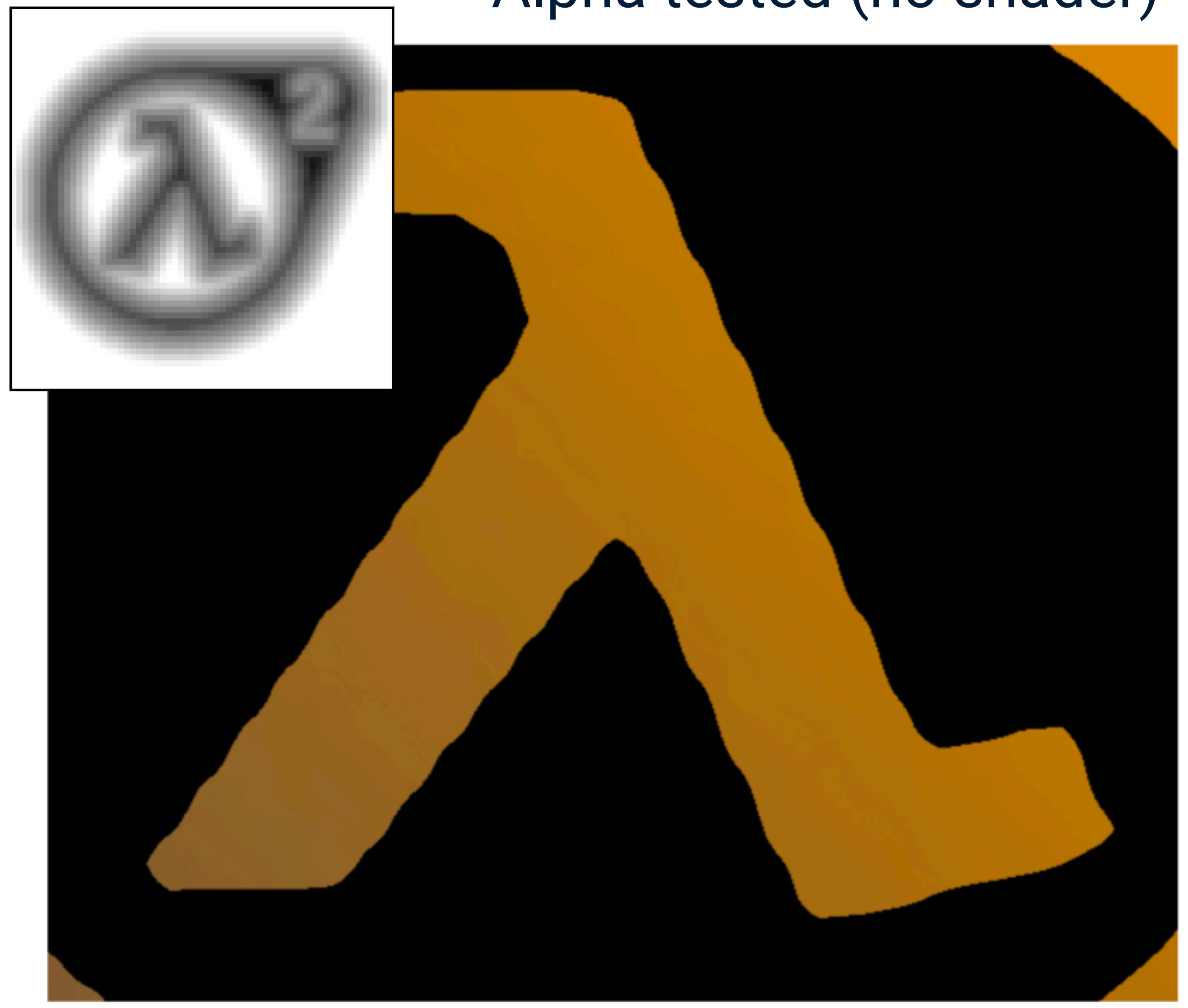
SINGLE CHANNEL SDF

Green (2007) Improved Alpha-Tested Magnification for Vector Textures

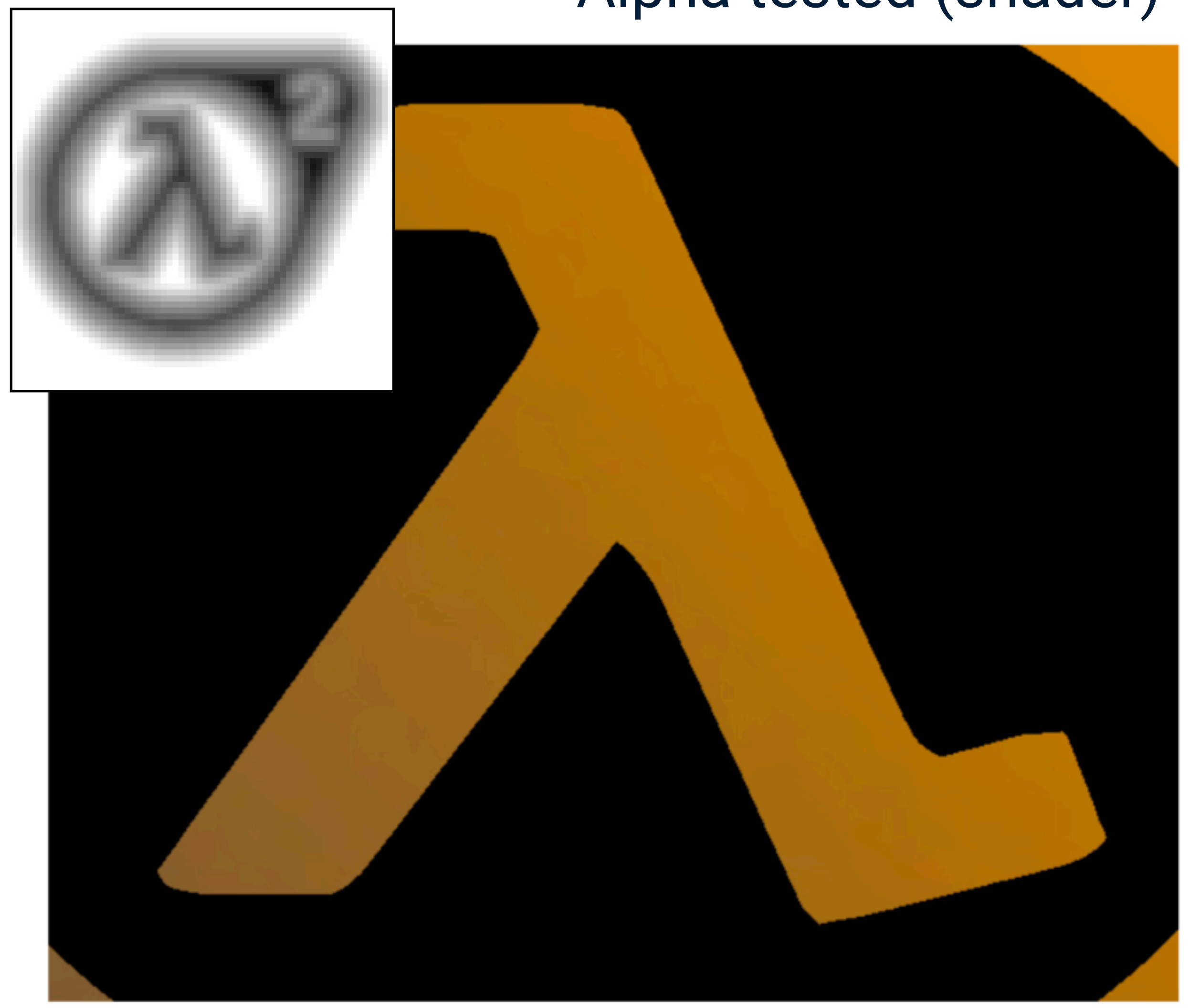
A distance field is generated from a high resolution image, and then stored into a channel of a lower-resolution texture. In the simplest case, this texture can then be rendered simply by using the alpha- testing and alpha-thresholding feature of modern GPUs, without a custom shader. This allows the technique to be used on even the lowest-end 3D graphics hardware. With the use of programmable shading, the technique is extended to perform various special effect renderings, including soft edges, outlining, drop shadows, multi-colored images, and sharp corners.

64x64 Alpha blended

64x64 Alpha tested (no shader)



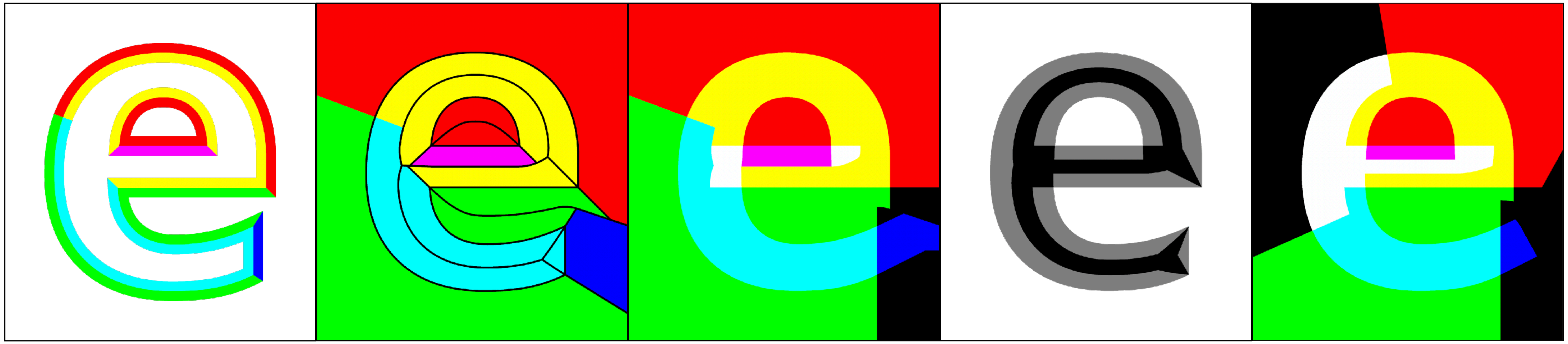
64x64 Alpha tested (shader)



SDF: MULTIPLE CHANNELS

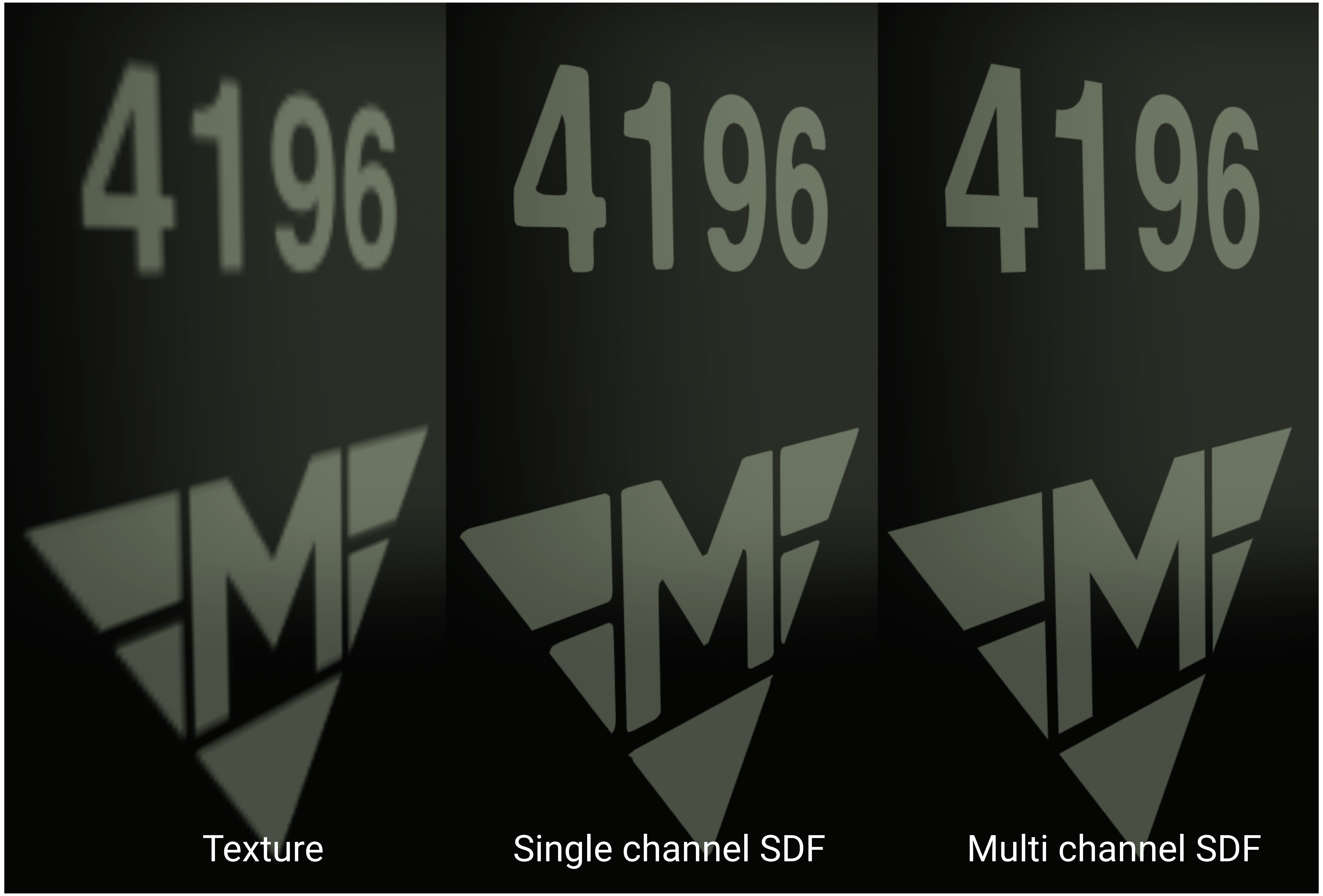
Chlumsky (2015) Shape Decomposition for Multi-channel Distance Fields

Edge Coloring | Plane partioning | Decomposition | Inner padding | Padded Decomposition



SDF: Multiple Channels

Chlumsky (2015) Shape Decomposition for Multi-channel Distance Fields

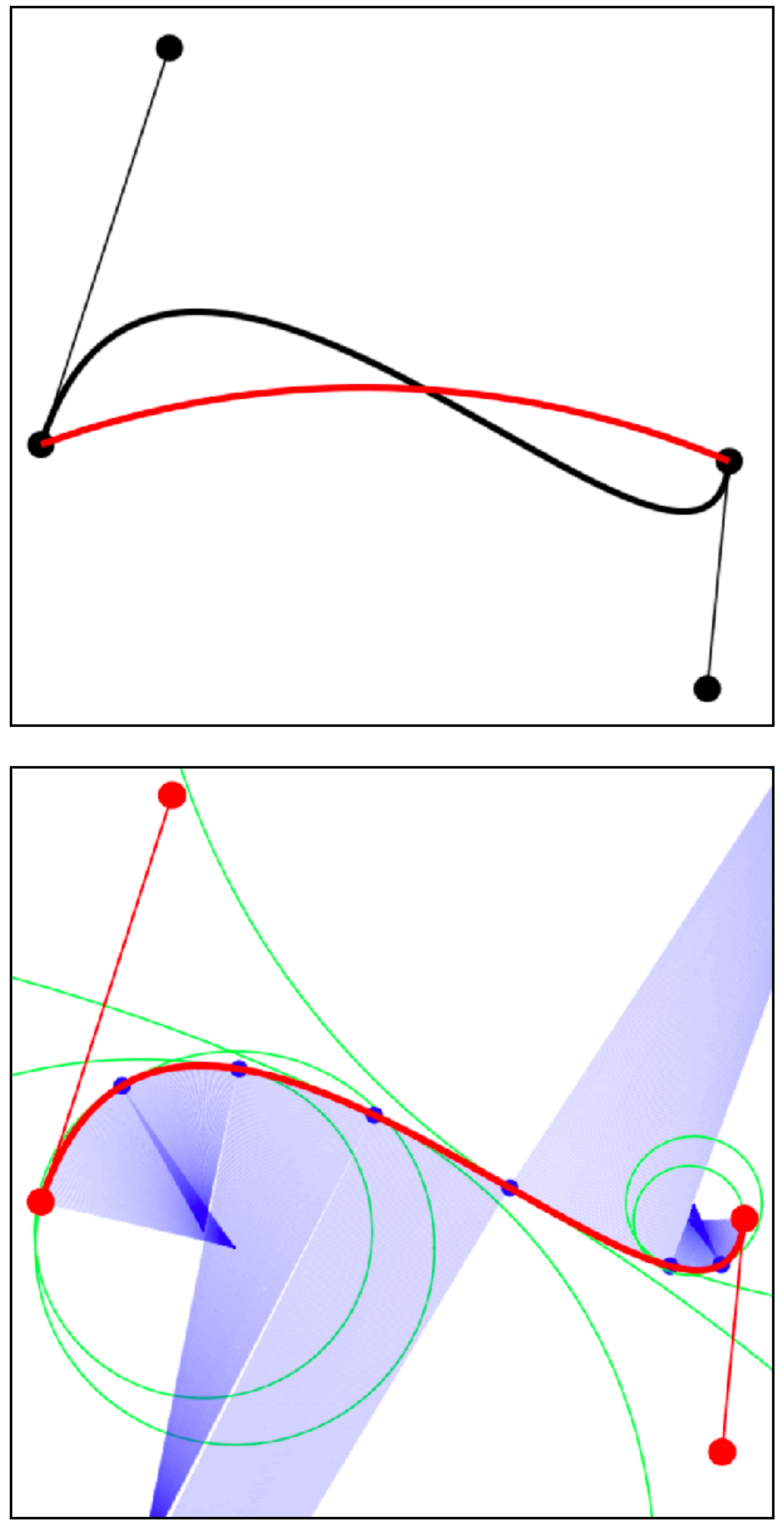


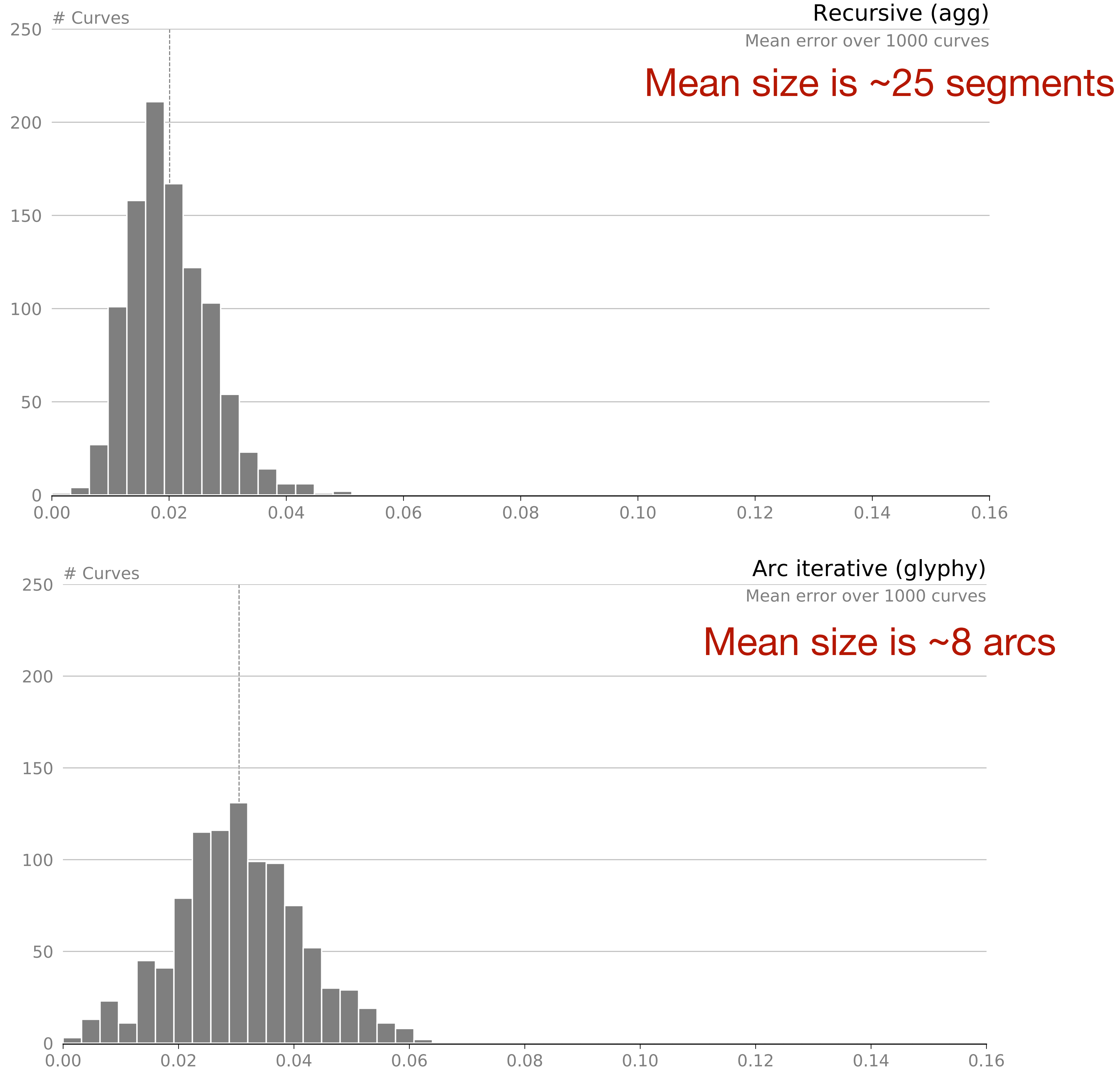
Texture | Single channel SDF | Multi channel SDF

ARC APPROXIMATION

Esfahbod (2012) Glyphy

Since distance to arbitrary Bézier curves is hard, we can instead first convert a glyph into (approximated) circular arc splines and upload this converted vector glyph to the GPU. Distance are then computed onto the GPU. Corner cases are overlapping contours, tangent arcs and float precision.

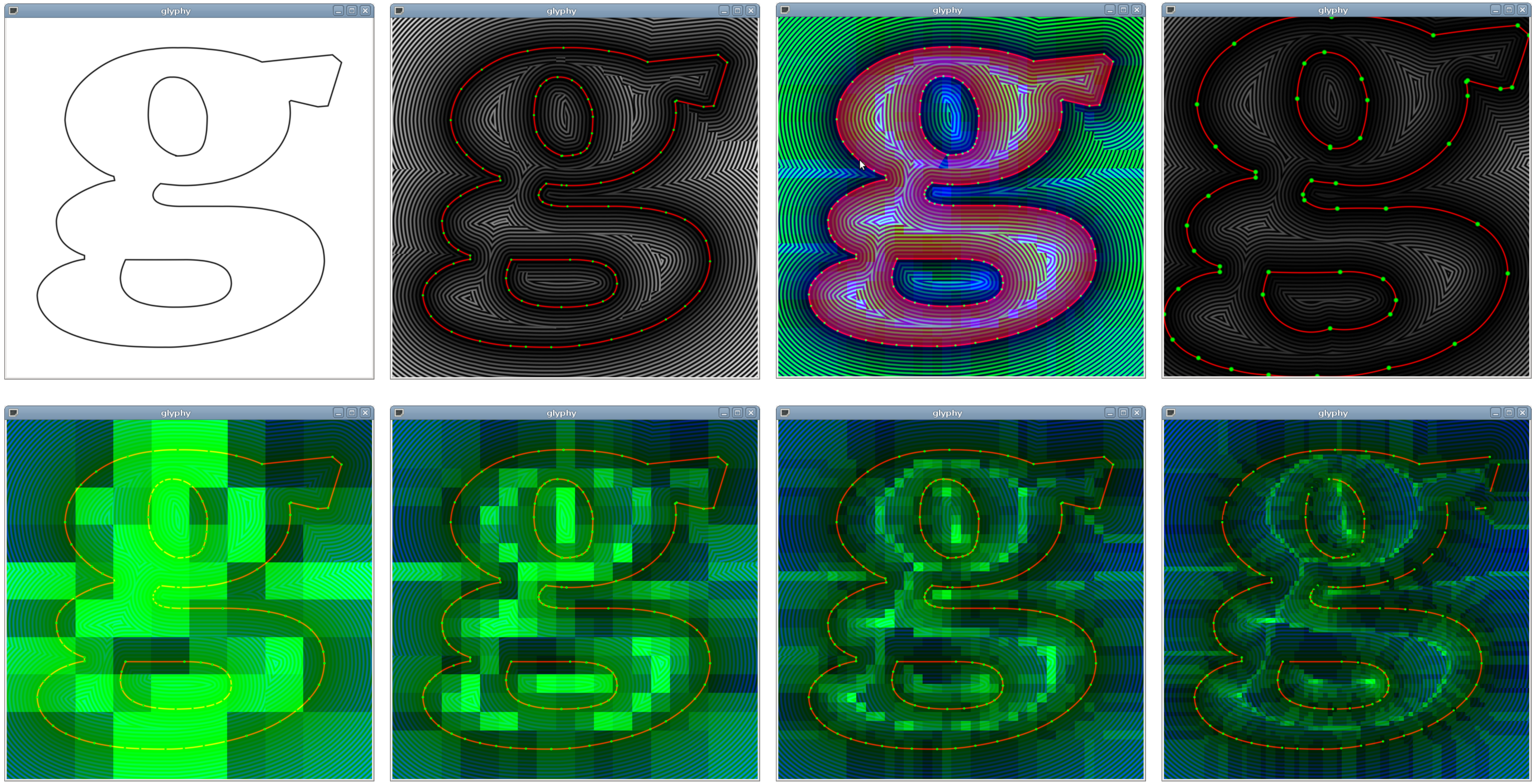




Arc approximation

Esfahbod (2012) Glyphy

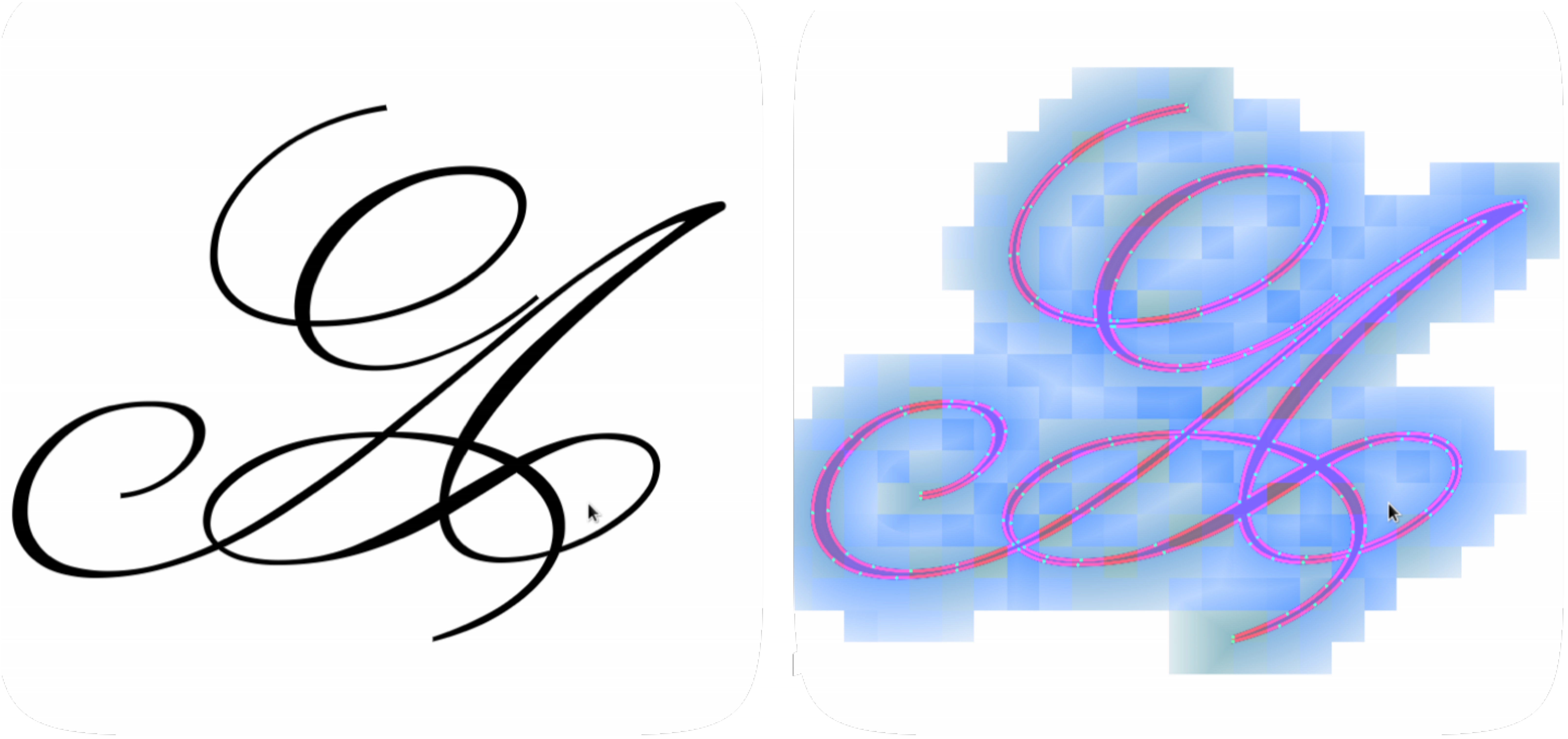
SDF-based antialiasing + subpixel positioning + random access (coarse grid).



ARC APPROXIMATION

Esfahbod (2012) Glyphy

Drawback: Memory and speed are font dependent



N. ROUGIER (INRIA) & B. ESFAHBOD (GOOGLE)

GENERATIONS
SIGGRAPH2018

PART III

Charles Loop and Jim Blinn introduced in 2005 a new approach for resolution-independent rendering of quadratic and cubic spline curves.

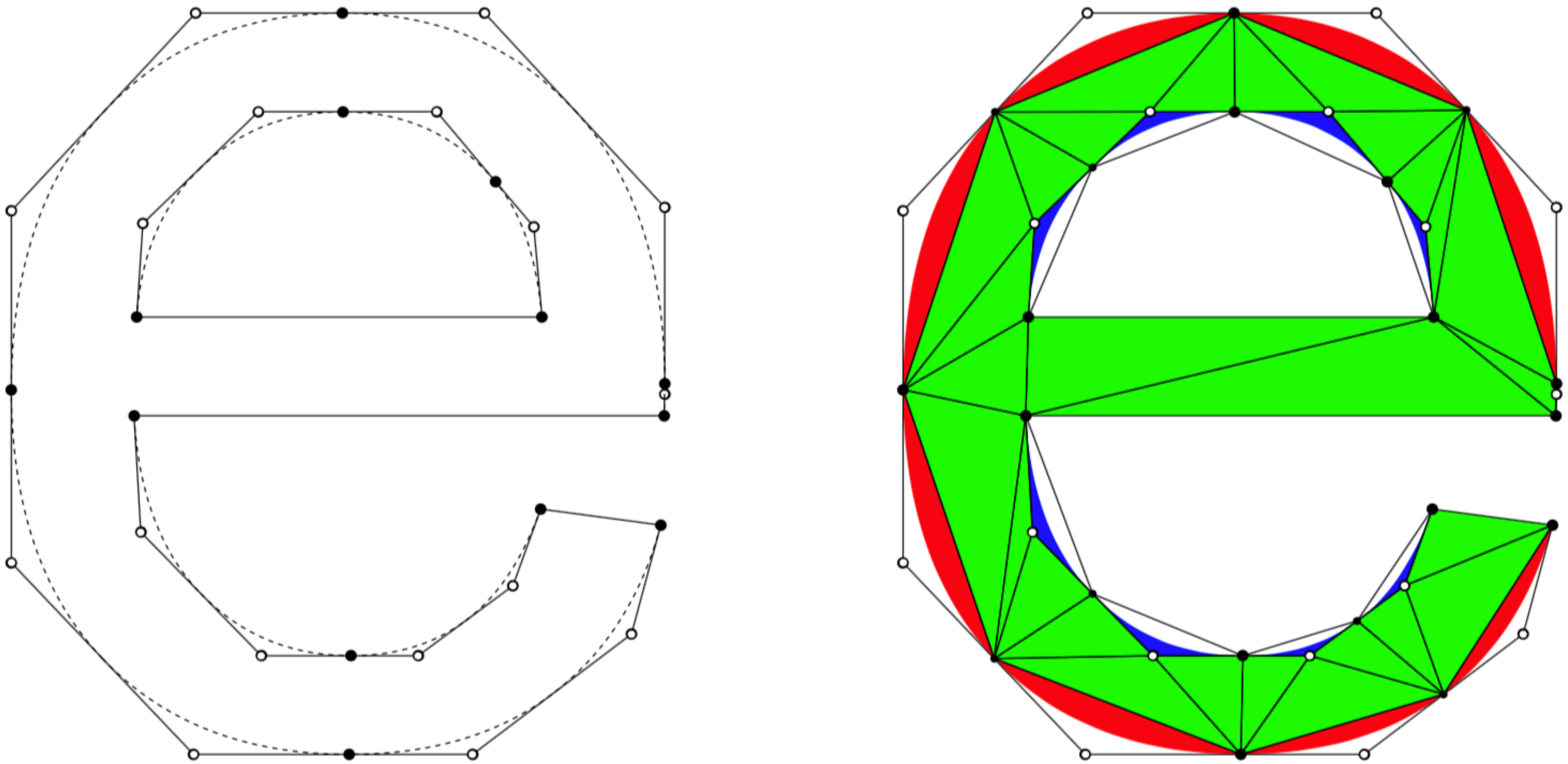
By tessellating a glyph the proper way, they offered de facto a method for resolution independent rendering of a glyph with good rendering quality.

Image by Black[Foundry]
black-foundry.com

GPU Friendly

Loop & Blinn (2005) Resolution Independent Curve Rendering

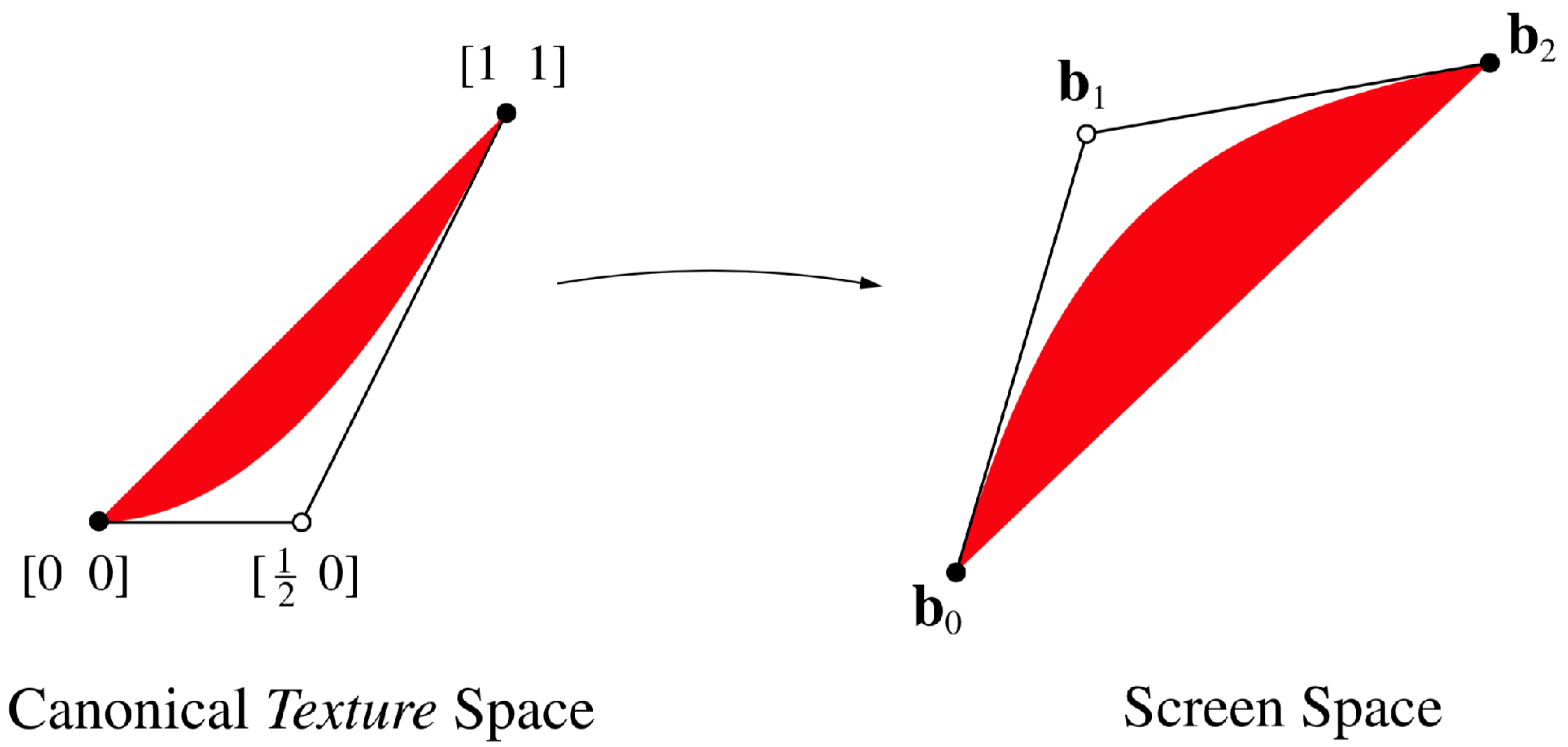
We present a method for resolution independent rendering of paths and bounded regions, defined by quadratic and cubic spline curves, that leverages the parallelism of programmable graphics hardware to achieve high performance. Our result is a mechanism for rendering vector geometry that has the following properties: resolution independence, compact geometric representation, high performance.



GPU Friendly

Loop & Blinn (2005) Resolution Independent Curve Rendering

We determine if the pixel is inside or outside the curve by evaluating $f(u, v) = u^2 - v$ *in a pixel shader program. If* $f(u, v) < 0$ *then the pixel is inside the curve, otherwise it is outside.*

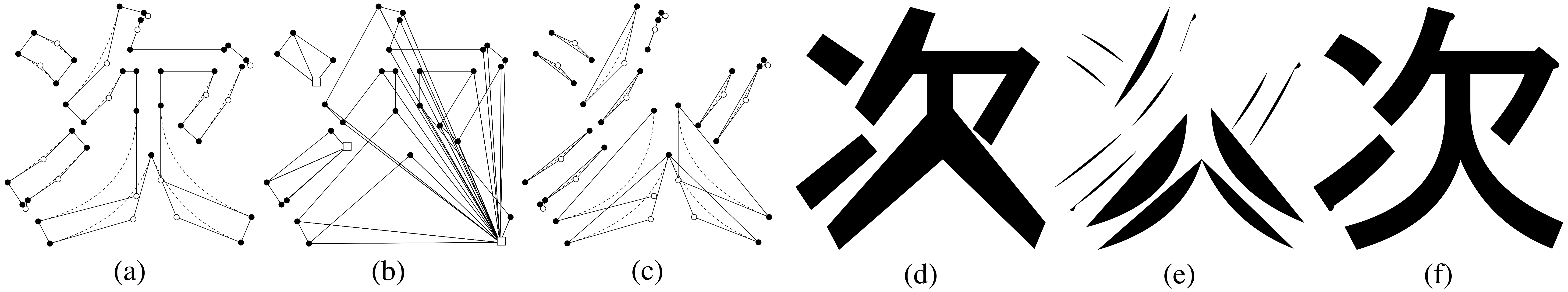


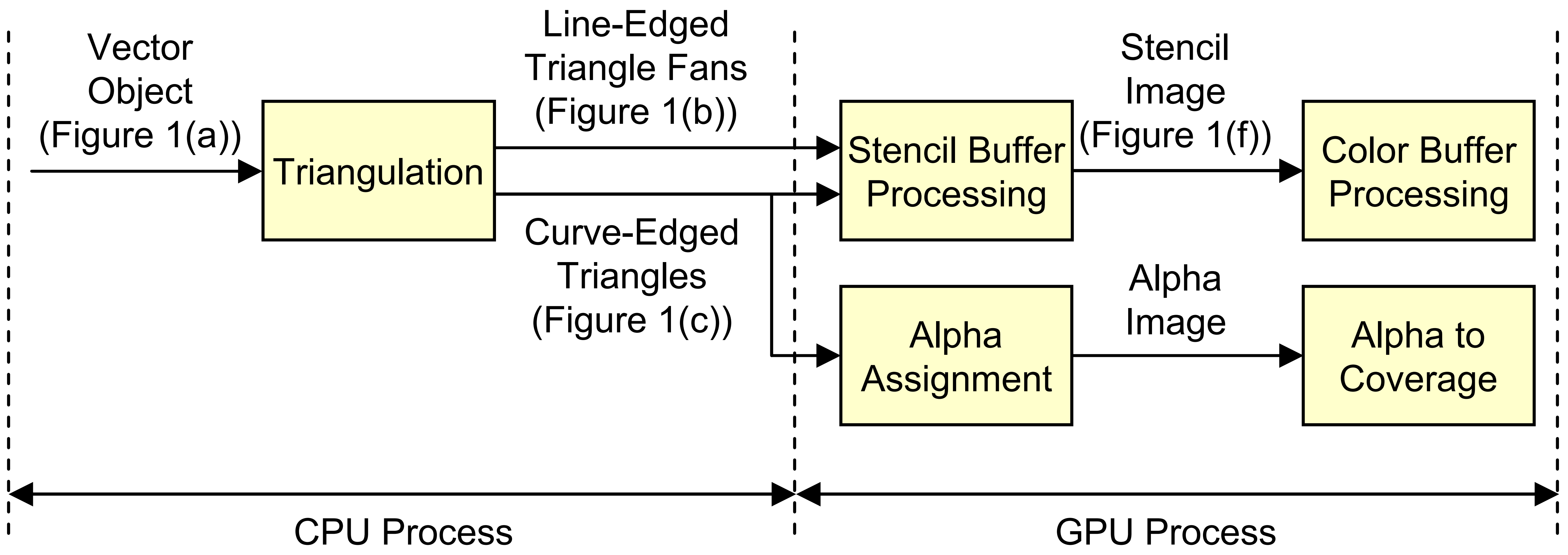
See also
"Rendering Vector Art on the GPU" (Charles Loop & Jim Blinn, GPU Gems 3, Chapter 25)
"Easy Scalable Text Rendering on the GPU" (Evan Wallace Medium)

GPU Friendly

Kokojima et al (2006) Resolution Independent Rendering of Deformable Vector Objects

This sketch presents a new method for resolution independent rendering of vector images suitable for programmable graphics hardware. We have enhanced a previous method [Loop and Blinn 2005] by using a stencil buffer and transparency multisampling.



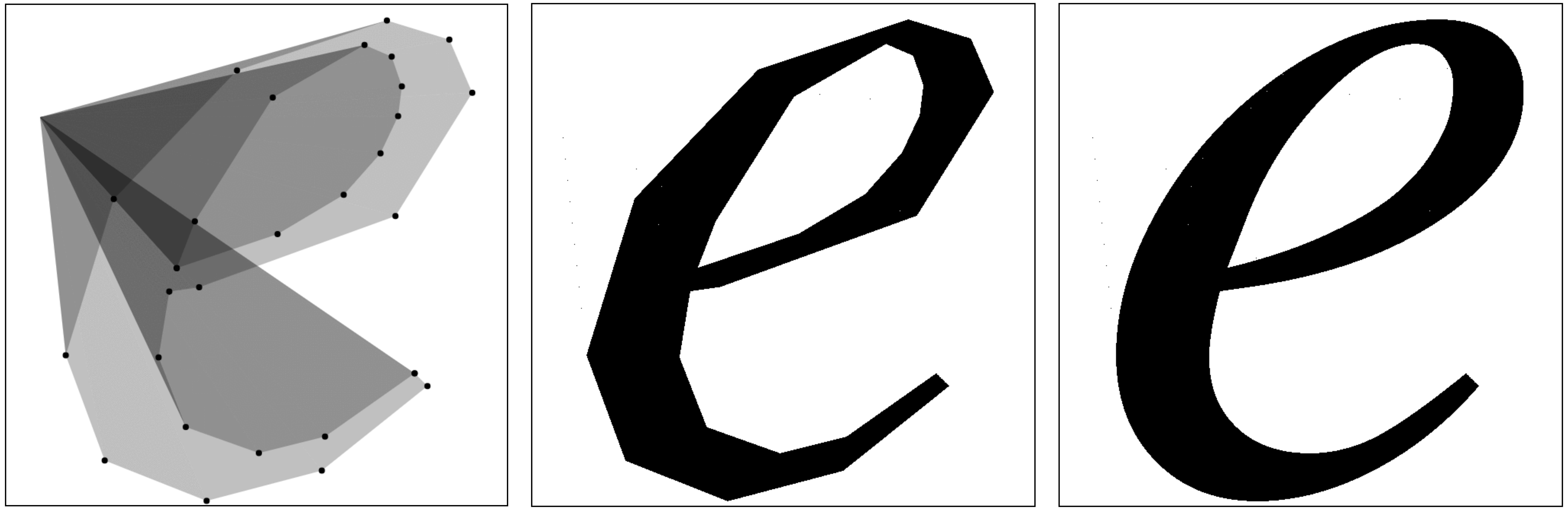


GENERATIONS
SIGGRAPH2018

GPU FRIENDLY

Kokojima et al (2006) Resolution Independent Rendering of Deformable Vector Objects

This sketch presents a new method for resolution independent rendering of vector images suitable for programmable graphics hardware. We have enhanced a previous method [Loop and Blinn 2005] by using a stencil buffer and transparency multisampling.



Animations by Evan Wallace (Easy Scalable Text Rendering on the GPU)

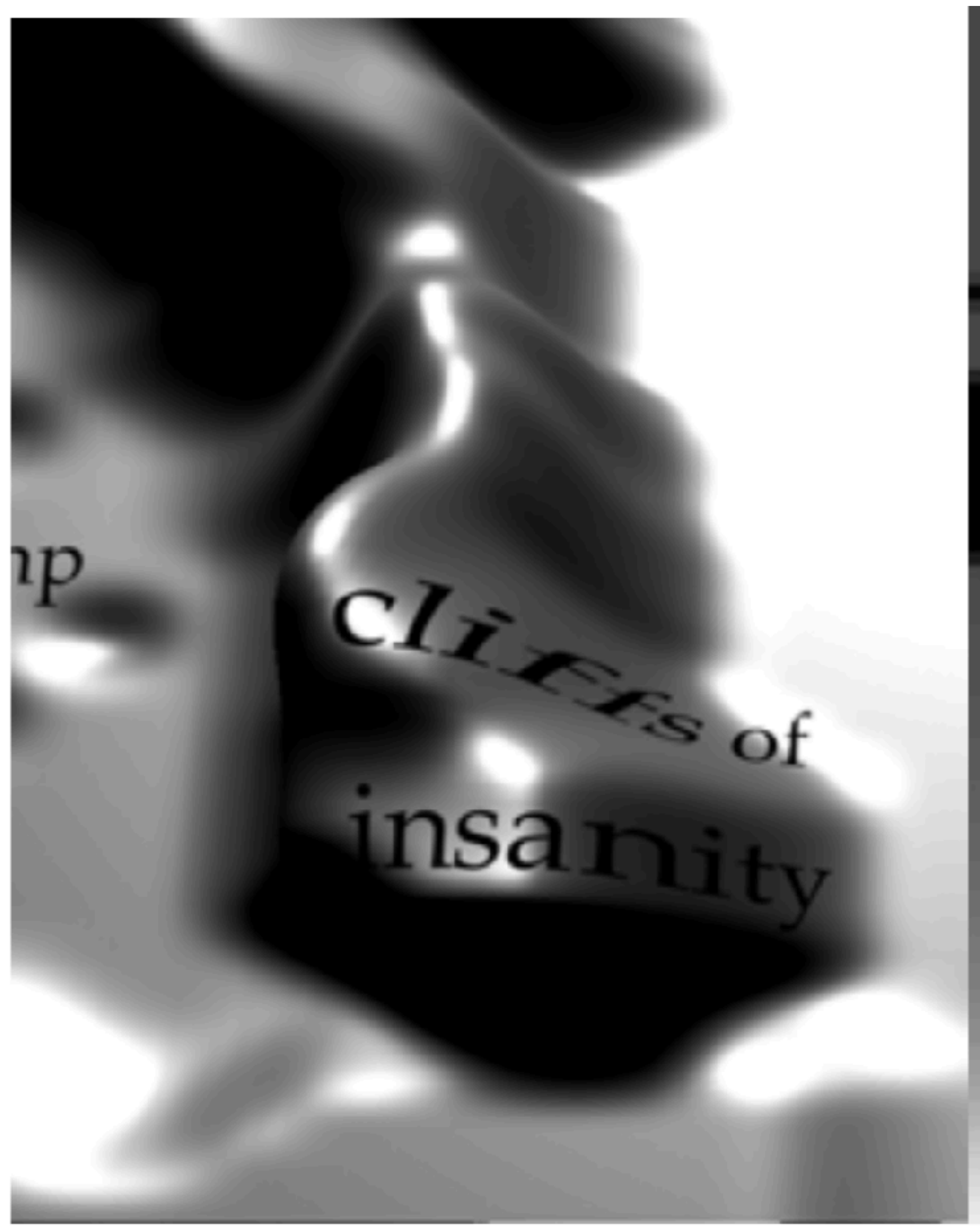
GENERATIONS SIGGRAPH2018

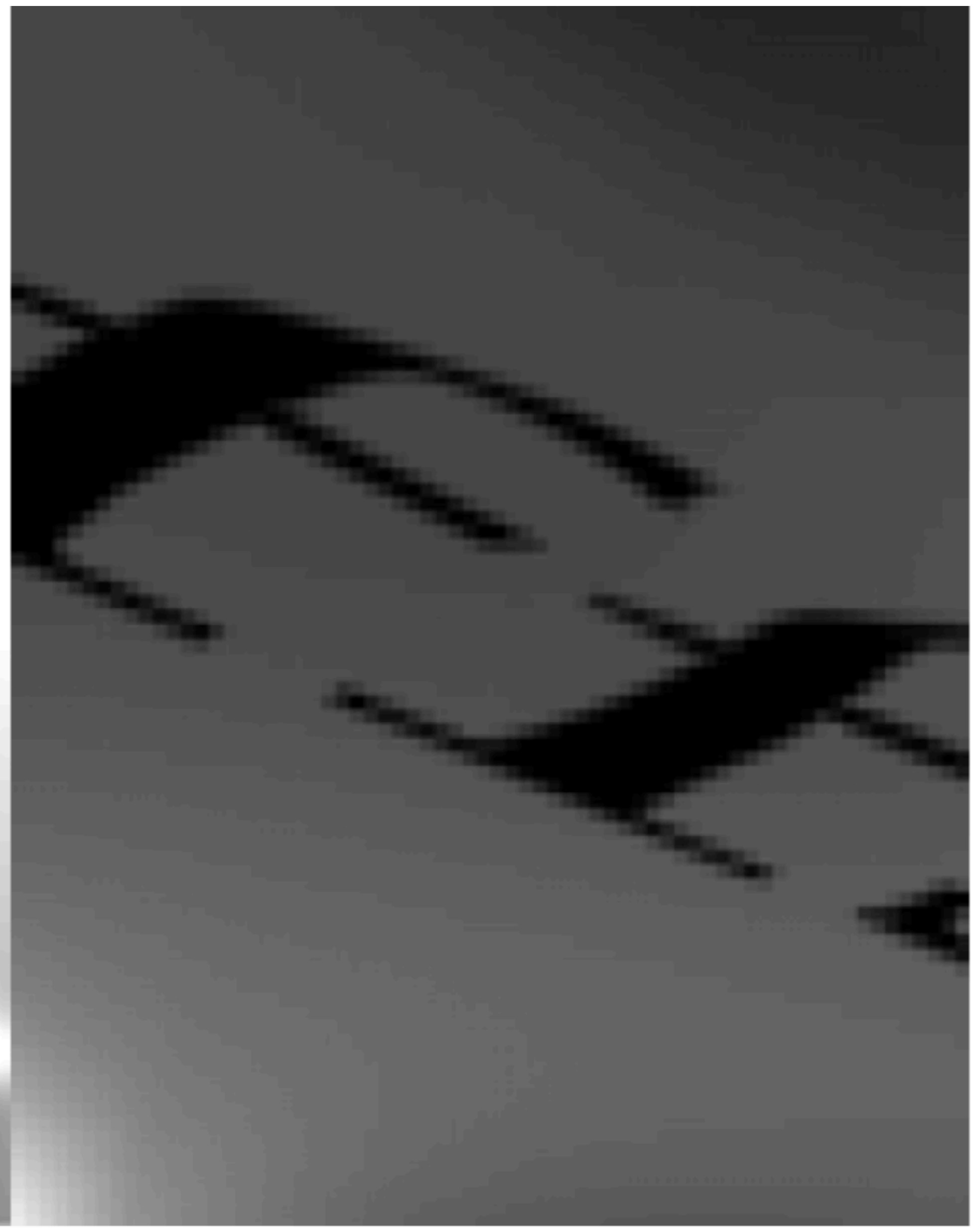
GPU FRIENDLY

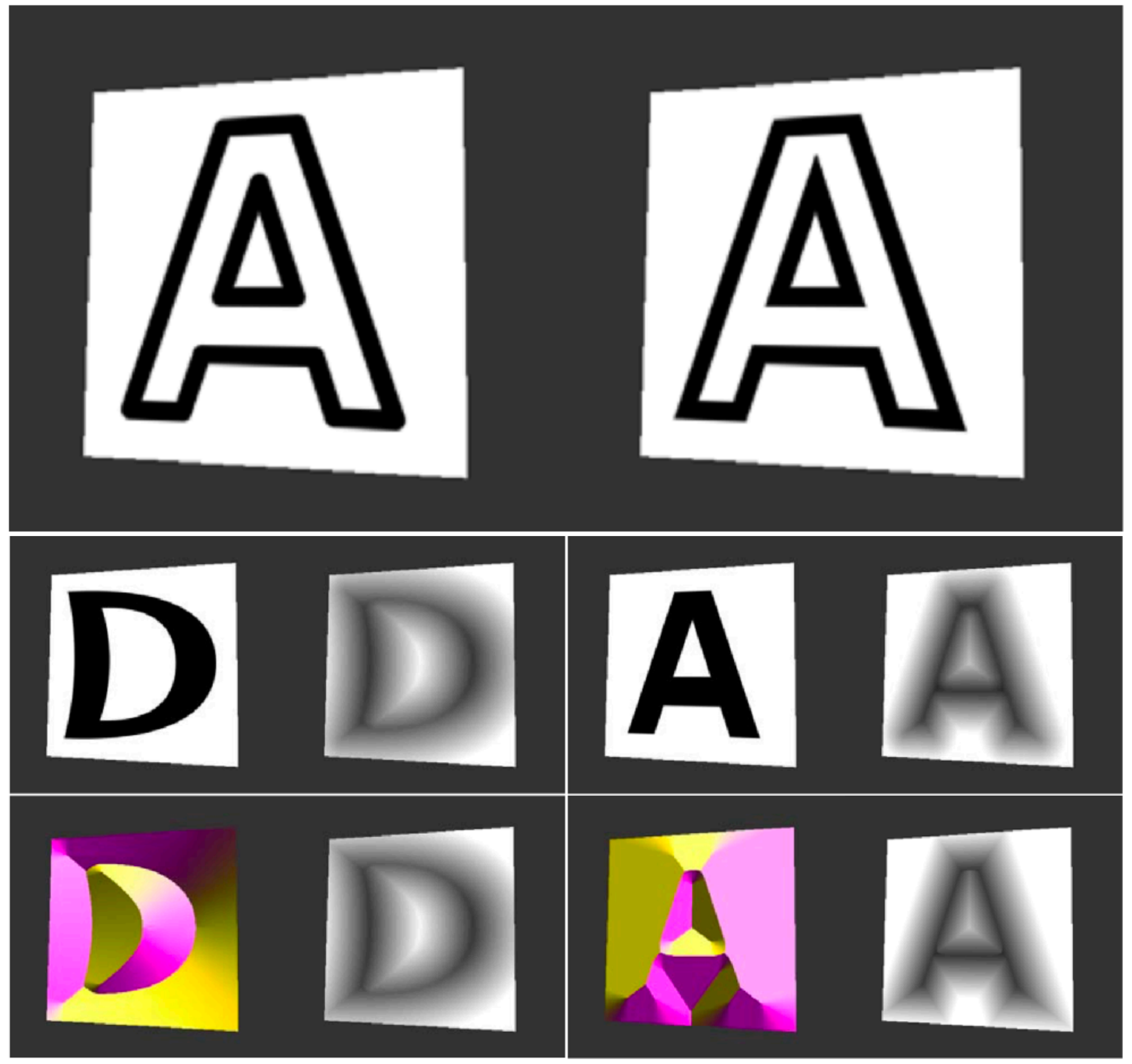
Qin et al (2006) Real-Time Texture-Mapped Vector Glyphs

We present a vector graphics representation suitable for real-time rendering on GPUs. Our representation can be used in place of a texture map, and renders precise antialiased edges at any magnification. A combination of texture data and procedural computation is used to evaluate an exact signed distance to a contour and its gradient.

Anisotropic antialiasing technique + GPU-based representation of contours + Packed grid accelerator structure based + Sprite mapping technique + Special effects



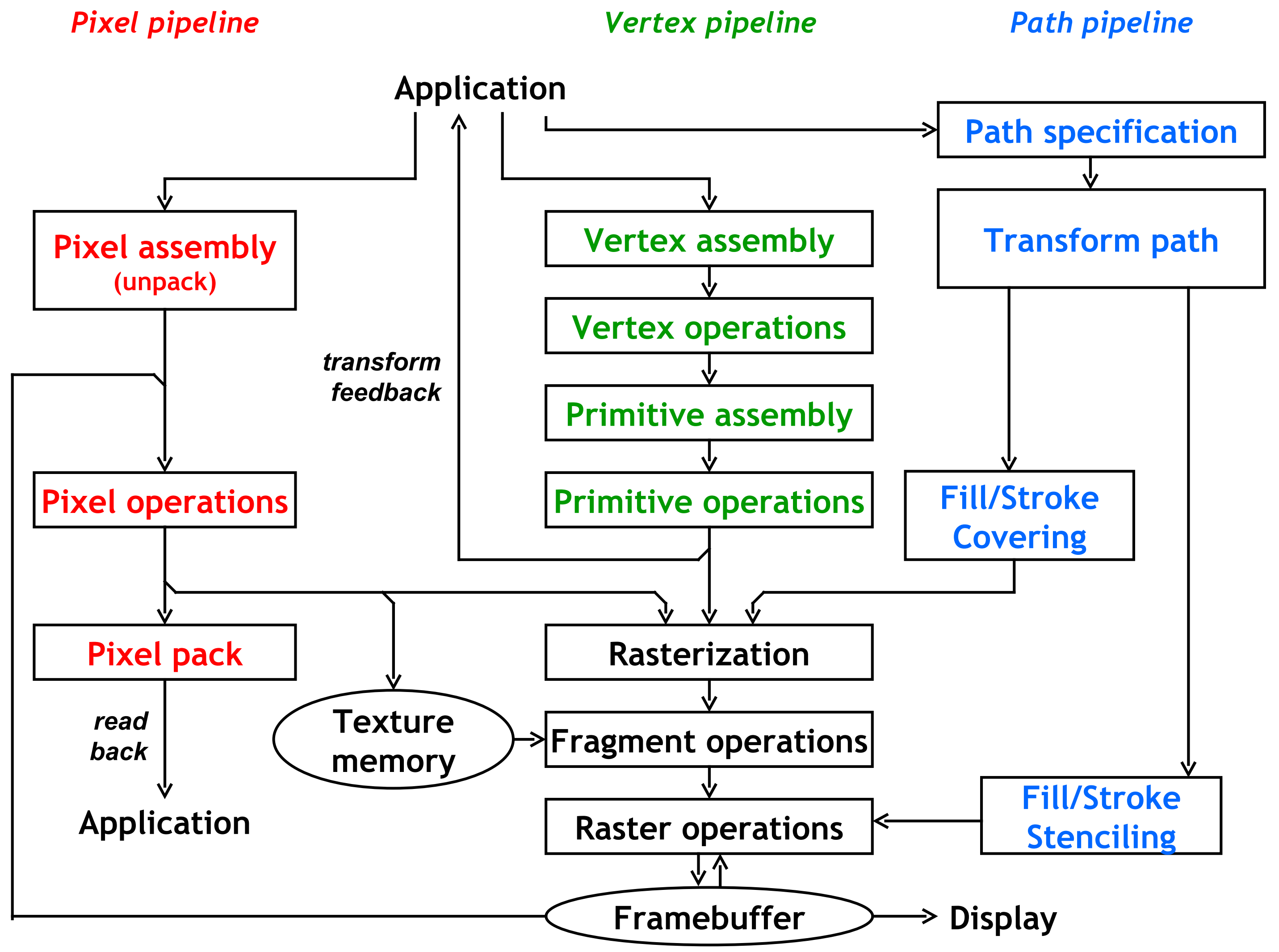




GPU FRIENDLY

Kilgard & Bolz (2012) GPU-accelerated Path Rendering

We introduce a two-step "Stencil, then Cover" (StC) programming interface. Our GPU-based approach builds upon existing techniques for curve rendering using the stencil buffer, but we explicitly decouple in our programming interface the stencil step to determine a path's filled or stroked coverage from the subsequent cover step to rasterize conservative geometry intended to test and reset the coverage determinations of the first step while shading color samples within the path.



GPU Only

Lengyel (2017) GPU-Centered Font Rendering Directly from Glyph Outlines

This paper describes a method for rendering antialiased text directly from glyph outline data on the GPU without the use of any precomputed texture images or distance fields. This capability is valuable for text displayed inside a 3D scene because, in addition to a perspective projection, the transform applied to the text is constantly changing with a dynamic camera view. Our method overcomes numerical precision problems that produced artefacts in previously published techniques and promotes high GPU utilization with an implementation that naturally avoids divergent branching.

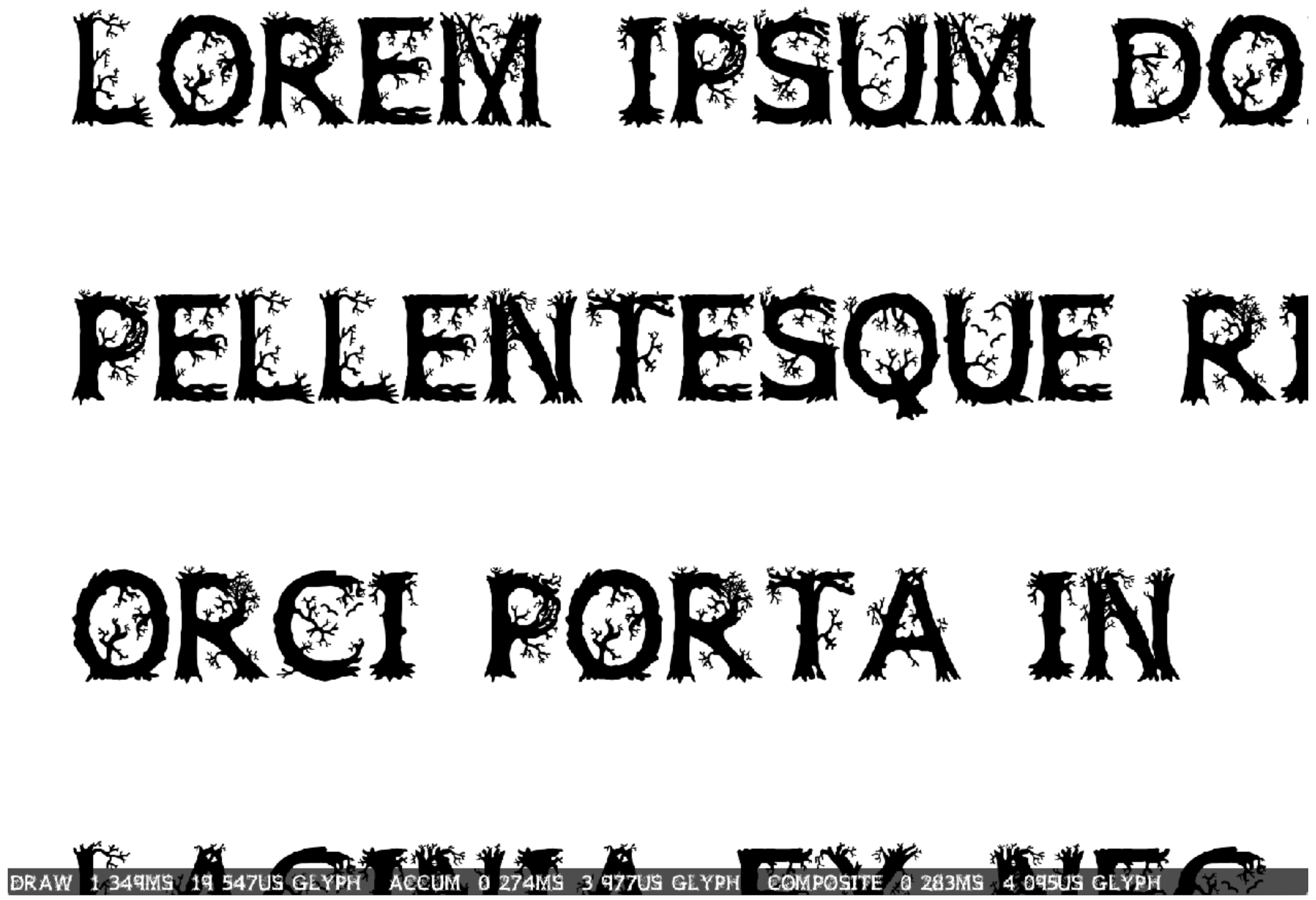
See demo at sluglibrary.com
See also GPU-Centered font rendering & GPU font rendering

Font	Sample	Complexity	Time
Arial	ABCDEFG	20	1.1 ms
Centaur	ABCDEFG	48	1.3 ms
Halloween	ABCDEFG	72	3.8 ms
Wildwood	ABCDEFG	546	13.3 ms

GPU ONLY

Walton (2017) Pathfinder, a fast GPU-based font rasterizer in Rust

“Today I’m pleased to announce Pathfinder, a Rust library for OpenType font rendering. The goal is nothing less than to be the fastest vector graphics renderer in existence, and the results so far are extremely encouraging. Not only is it very fast according to the traditional metric of raw rasterization performance, it’s practical, featuring very low setup time (end-to-end time superior to the best CPU rasterizers), best-in-class rasterization performance even at small glyph sizes, minimal memory consumption (both on CPU and GPU), compatibility with existing font formats, portability to most graphics hardware manufactured in the past five years (DirectX 10 level), and security/safety.”



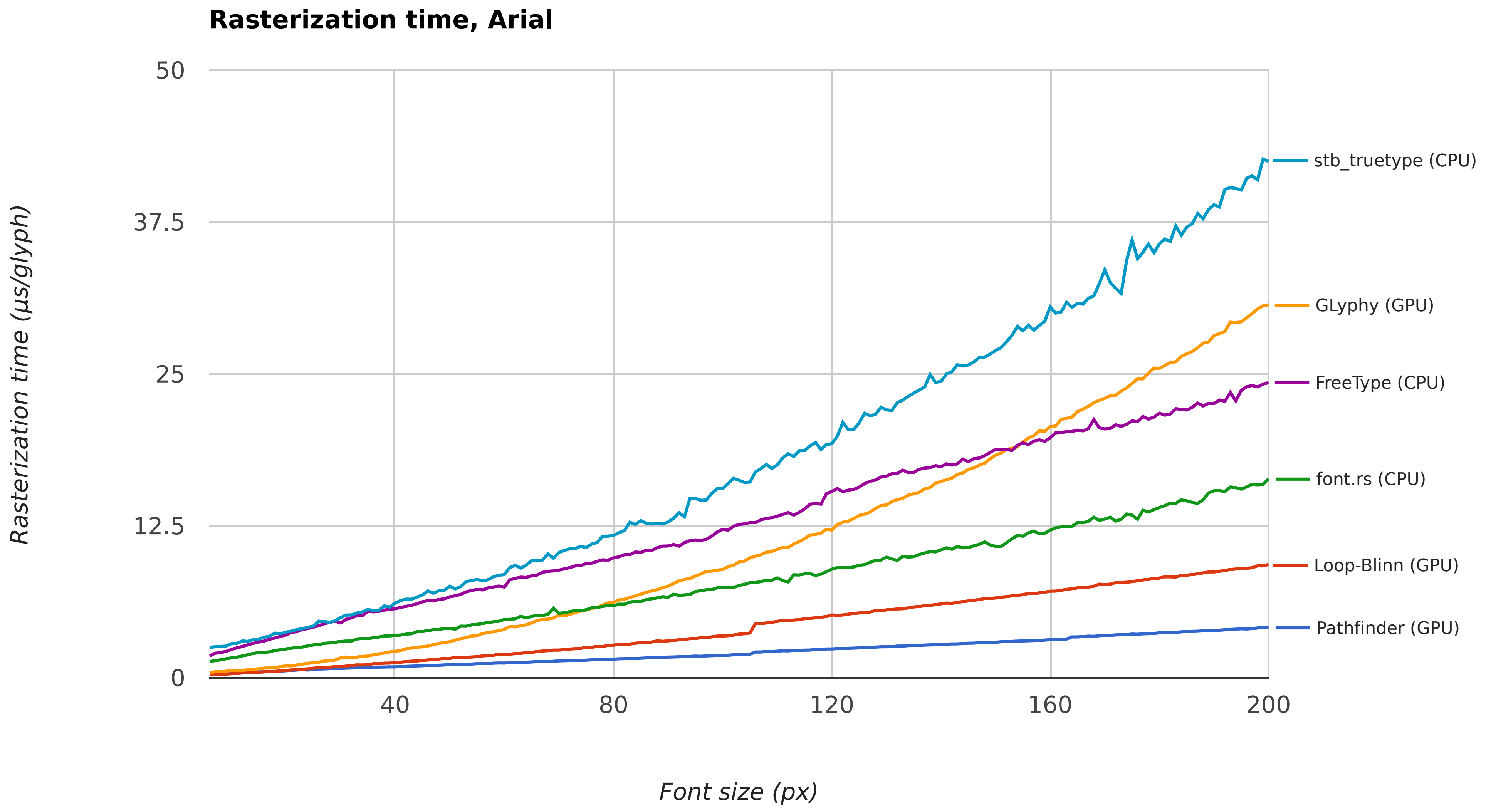
GENERATIONS SIGGRAPH2018

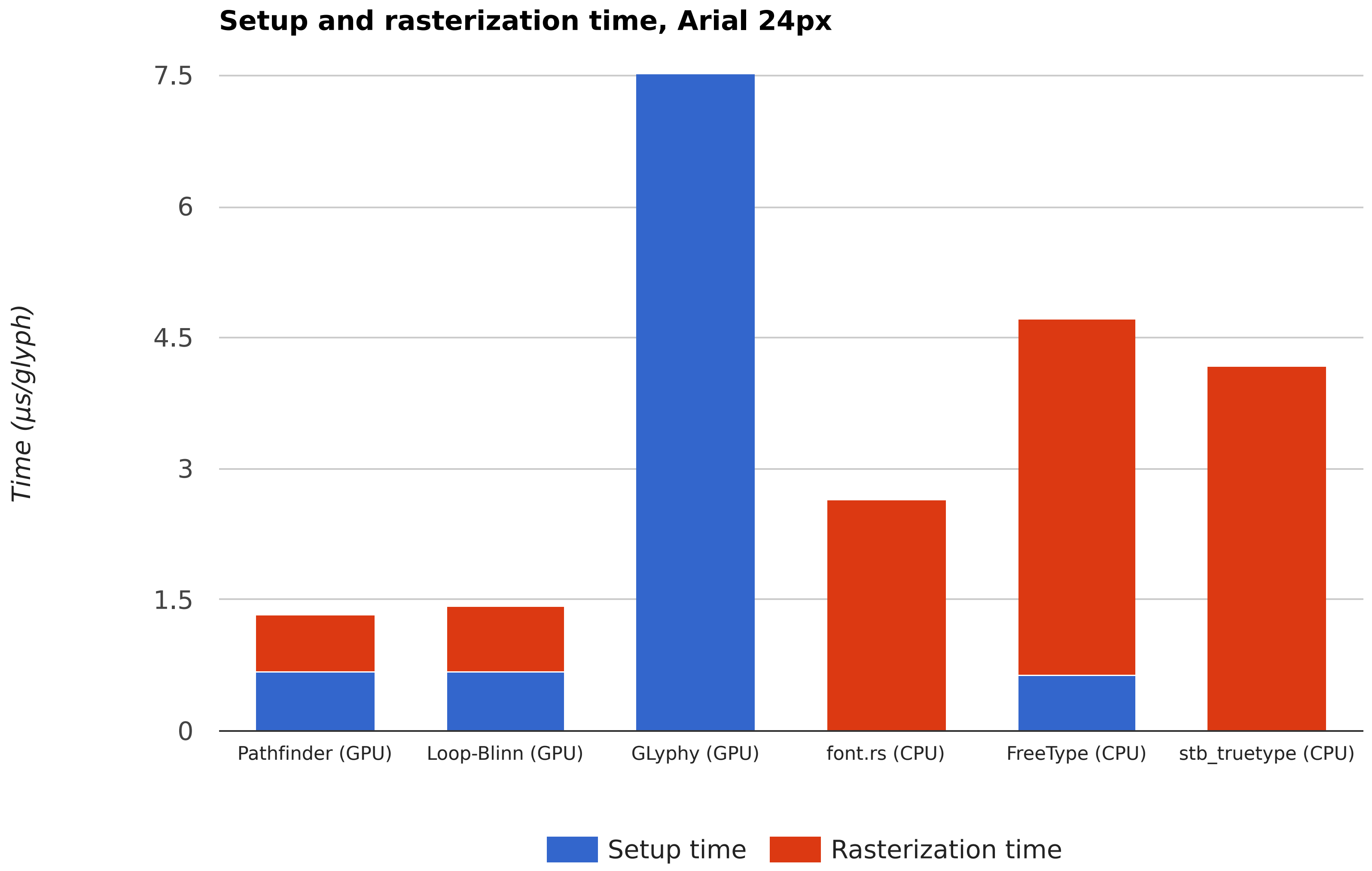
GPU Only

Walton (2017) Pathfinder, a fast GPU-based font rasterizer in Rust

PathFinder algorithm takes advantage of compute shaders (GL 4.3) & post-transform cache and rasterization occurs on the fly on the GPU. CPU setup time is minimal. Subpixel antialiasing & positionning, 2D, 3D, outline, etc.

It is mostly a (highly efficient) translation of CPU rasterization onto the GPU.





BEYOND MERE GEOMETRY

Maharik et al (2011) Digital Micrography

We present an algorithm for creating digital micrography images, or micrograms, a special type of calligrams created from minuscule text. These attractive text-art works successfully combine beautiful images with readable meaningful text.

GENERATIONS
SIGGRAPH2018

Conclusion

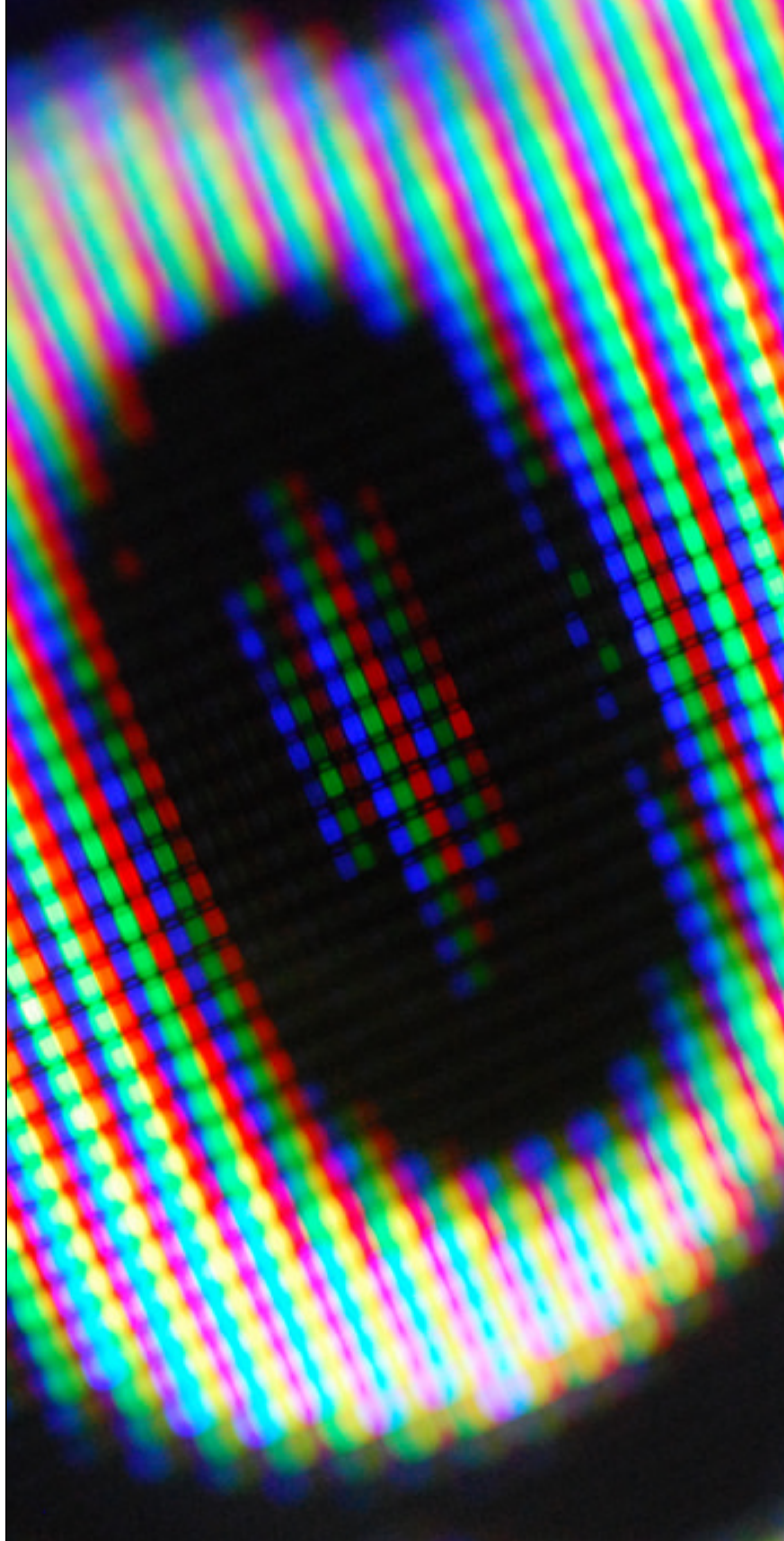
Many available techniques. Choice is dependent of usage (2D or 3D, dynamic or static, memory vs speed vs accuracy, etc).

PathFinder and Slug library seem to be the main players in 2018. Loop & Blinn still competitive.

Hinting and anti-aliasing still needed until we (all) get screen with dpi > 600.

Complex text layout is (really, really) difficult, you don't want to do it yourself: github.com/HOST-Oman/libraqm

Lot of patents around !



Television typography by Offstock
www.flickr.com/photos/126275939@N05

GENERATIONS
SIGGRAPH2018

FURTHER READING

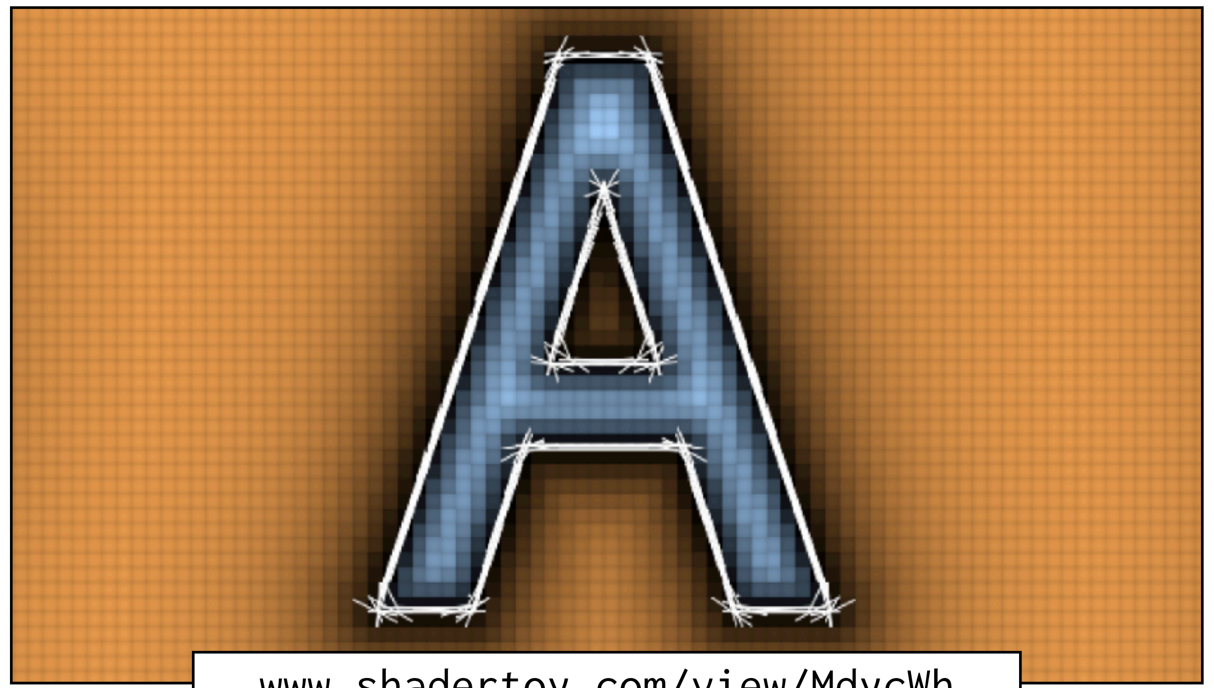
Web

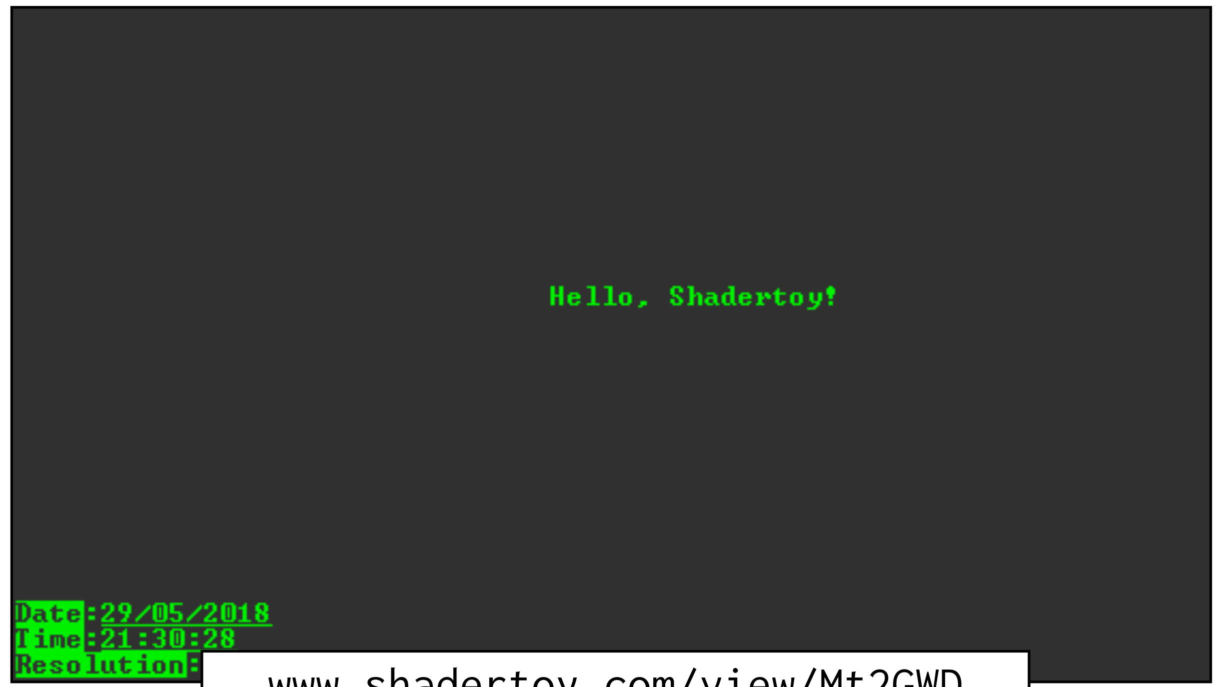
Texts Rasterization Exposures - Maxim Shemanarev (2006)
The Technology of Text - Kevin Larson (2007)
Treatise on Font Rasterisation - Freddie Witherden (2010)
State of Text Rendering - Behdad Esfahbod (2012)
CS 354 Typography - Mark Kilgard (2012)
High-DPI, Subpixel Text Positioning, Hinting - Behdad Esfahbod (2012)
Android's Font Renderer - Romain Guy (2014)
Complex Text on Simple Devices - Pedro Navarro (2016)
Introducing DirectWrite - Microsoft (2017)
A Primer on Bézier curves - Pomax (2017)

Papers

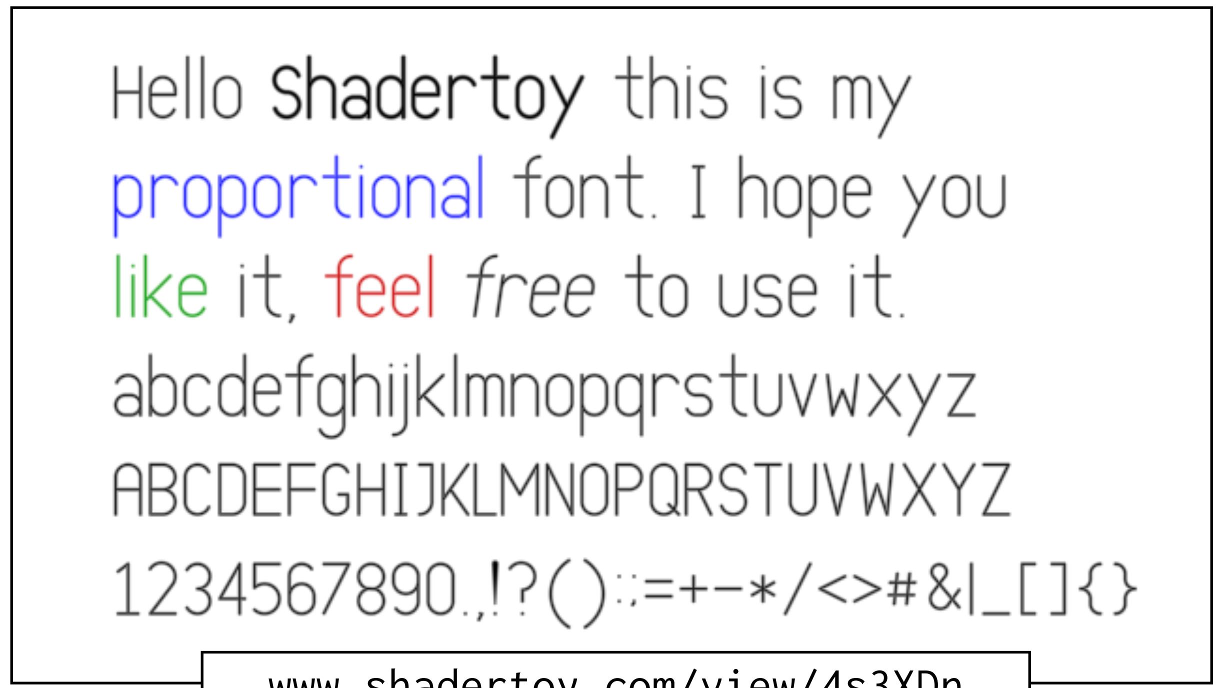
Mark Kilgard. A Simple OpenGL-based API for Texture Mapped Text (1997)
Sarah F. Frisken, et al. Adaptively sampled distance fields (2000)
Charles Loop and Jim Blinn. Resolution Independent Curve Rendering (2005)
Zhipei Qin et al, Real-Time Texture-Mapped Vector Glyphs (2006)
Chris Green. Improved Alpha-Tested Magnification for Vector Textures and Special Effects (2007)
Behdad Esfahbod. Glyphy (2011)
Ron Maharik et al. Digital Micrography (2011)
Stefan Gustavson. 2D Shape Rendering by Distance Fields (2012)
Mark Kilgard and Jeff Bolz. GPU-Accelerated Path Rendering (2012)
Nicolas P. Rougier. Higher Quality 2D Text Rendering (2013)
Viktor Chlumsky. Shape Decomposition for Multi-channel Distance Fields (2015)
Eric Lengyel. GPU-Centered Font Rendering Directly from Glyph Outlines (2017)
Patrick Walton. Path Finder (2017)

Related shadertoys

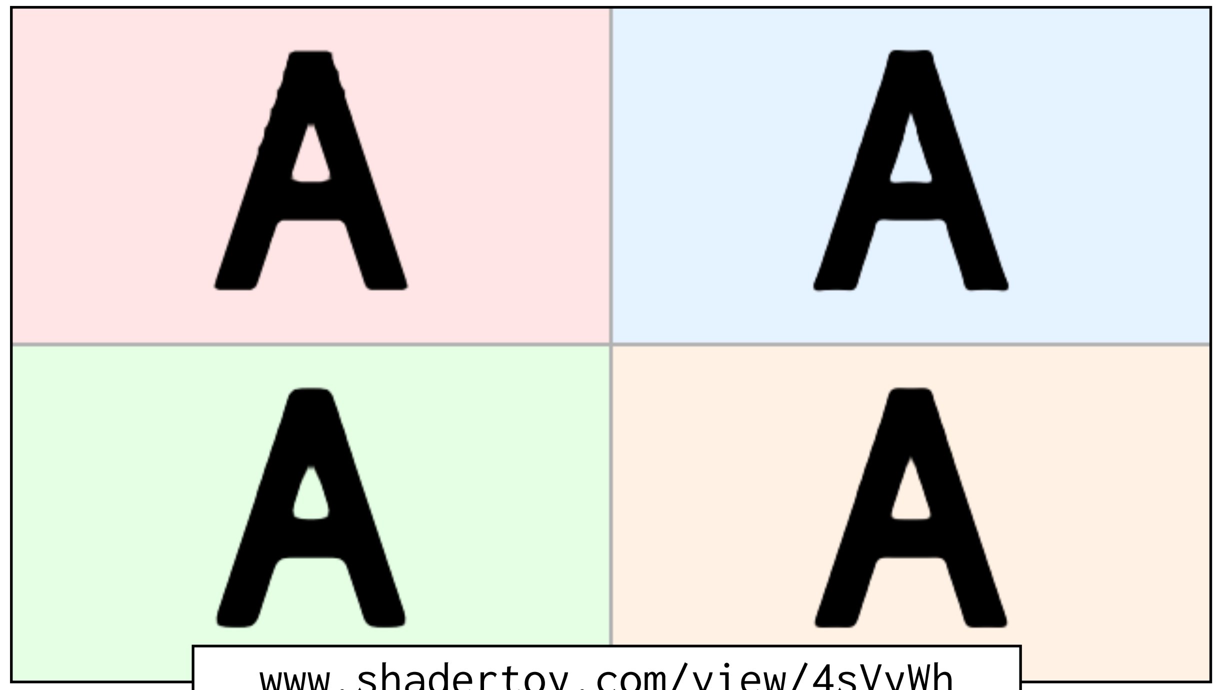

www.shadertoy.com/view/MdycWh



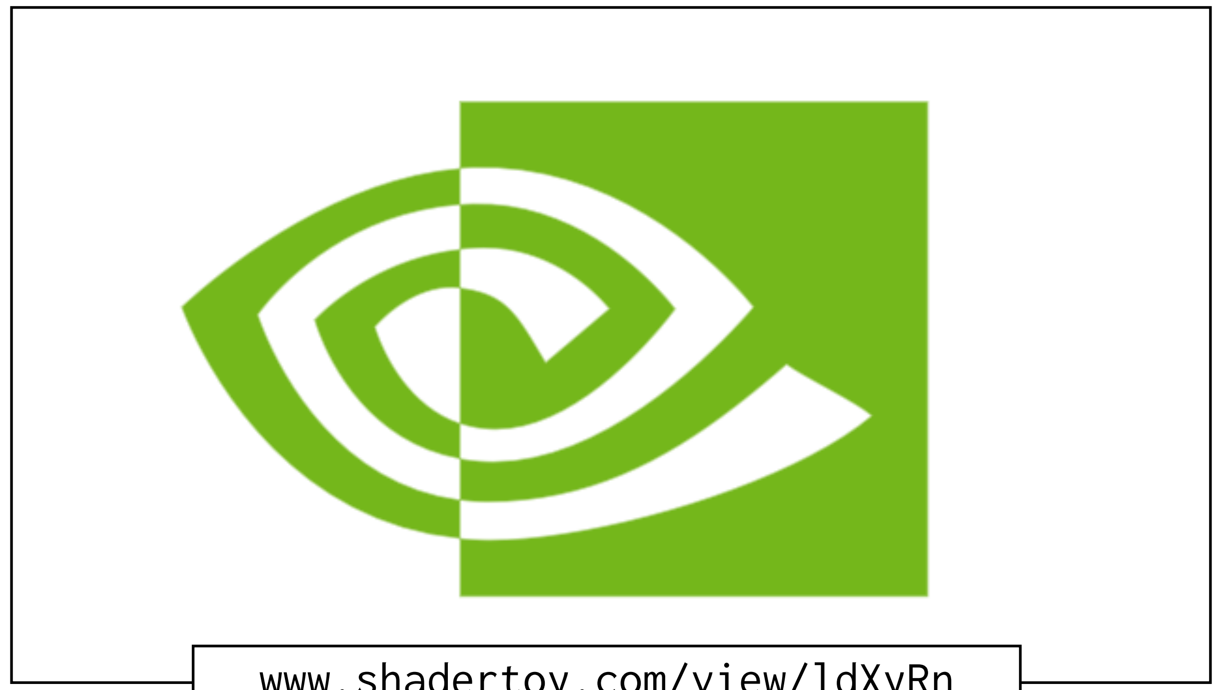
www.shadertoy.com/view/Mt2GWD

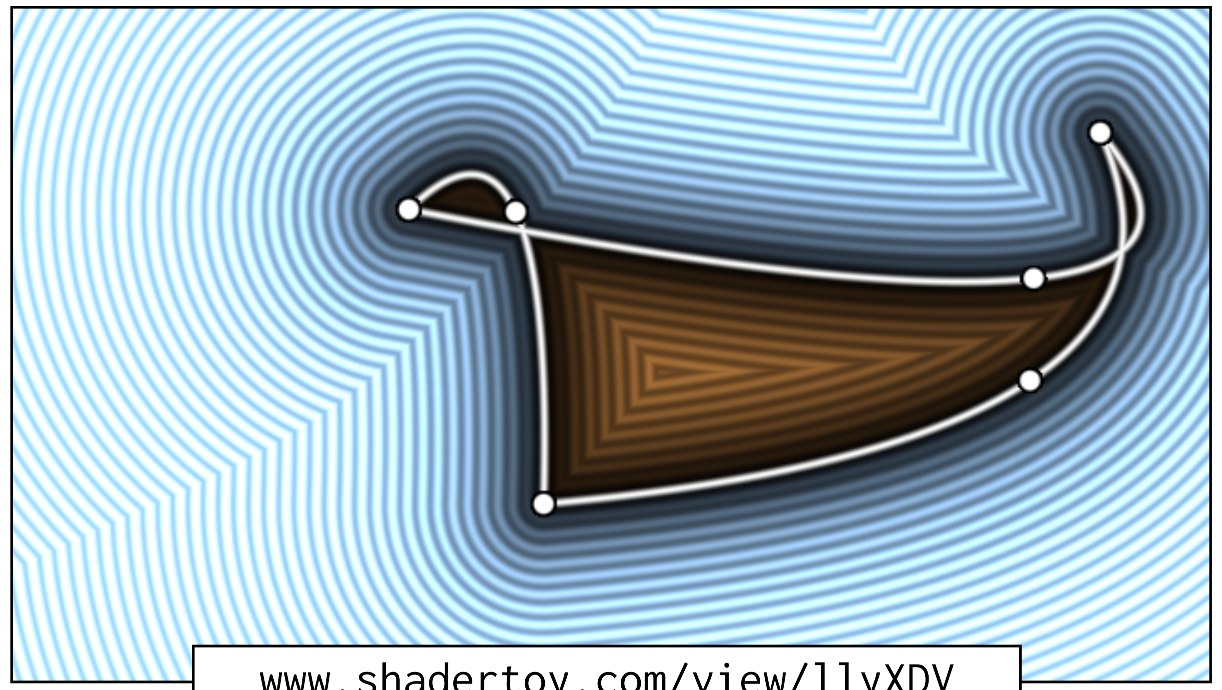


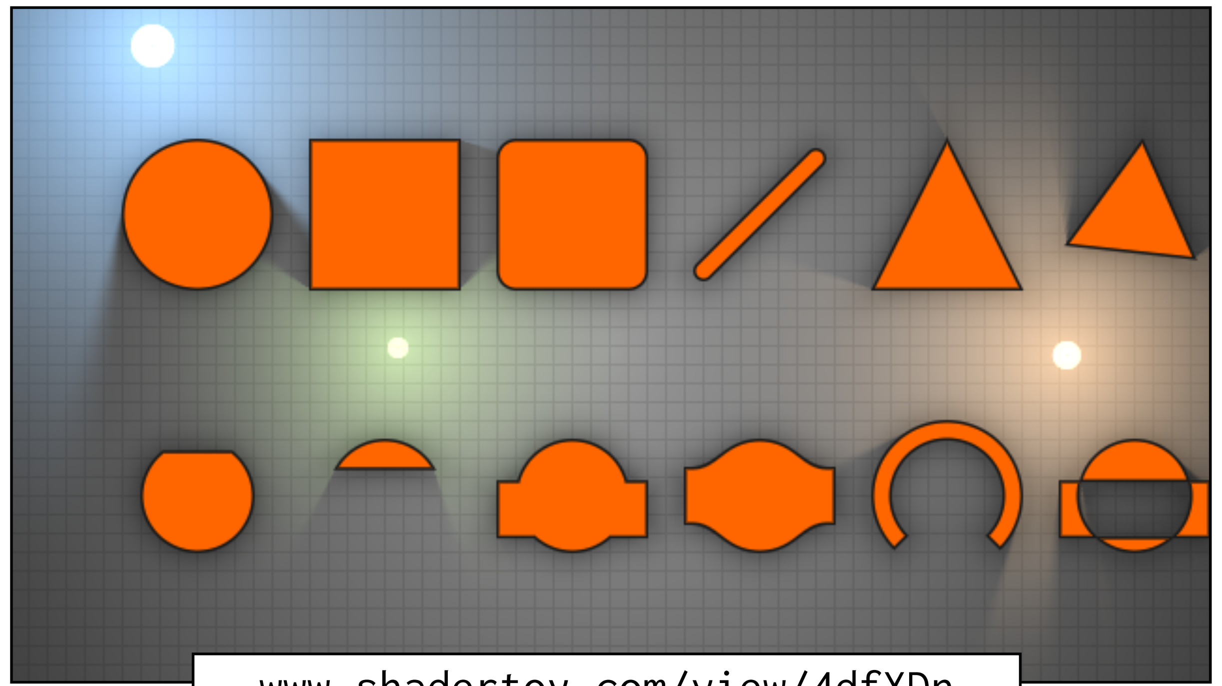
www.shadertoy.com/view/4s3XDn

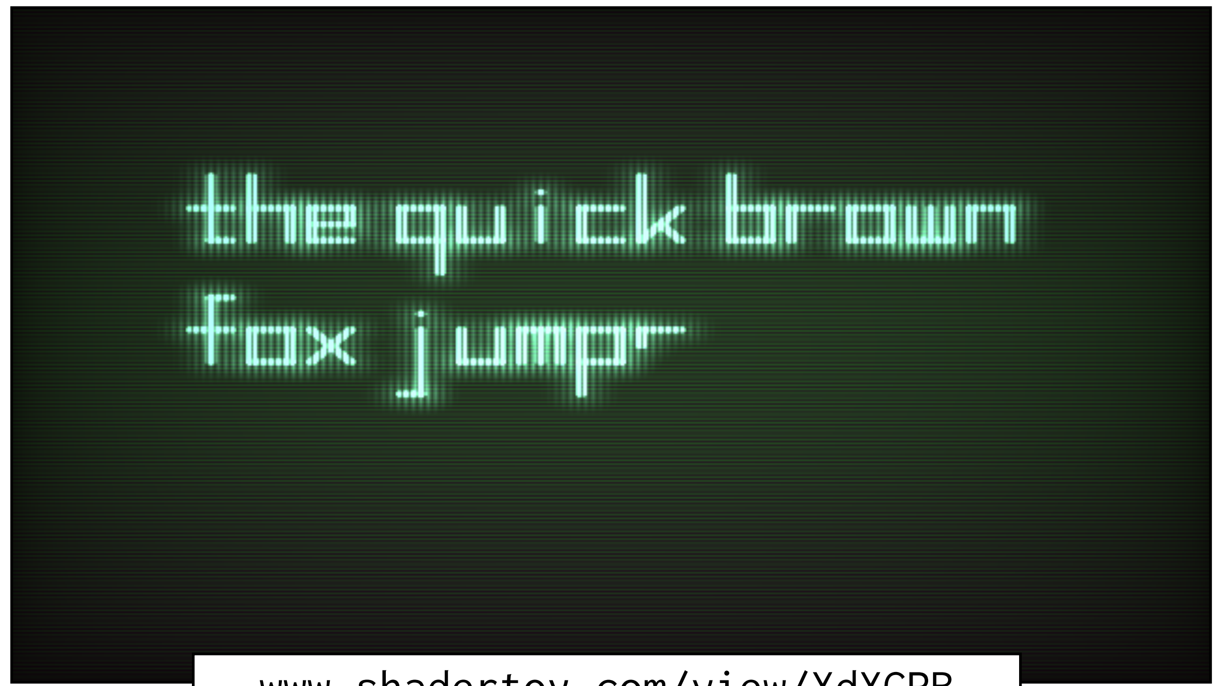


www.shadertoy.com/view/4sVyWh

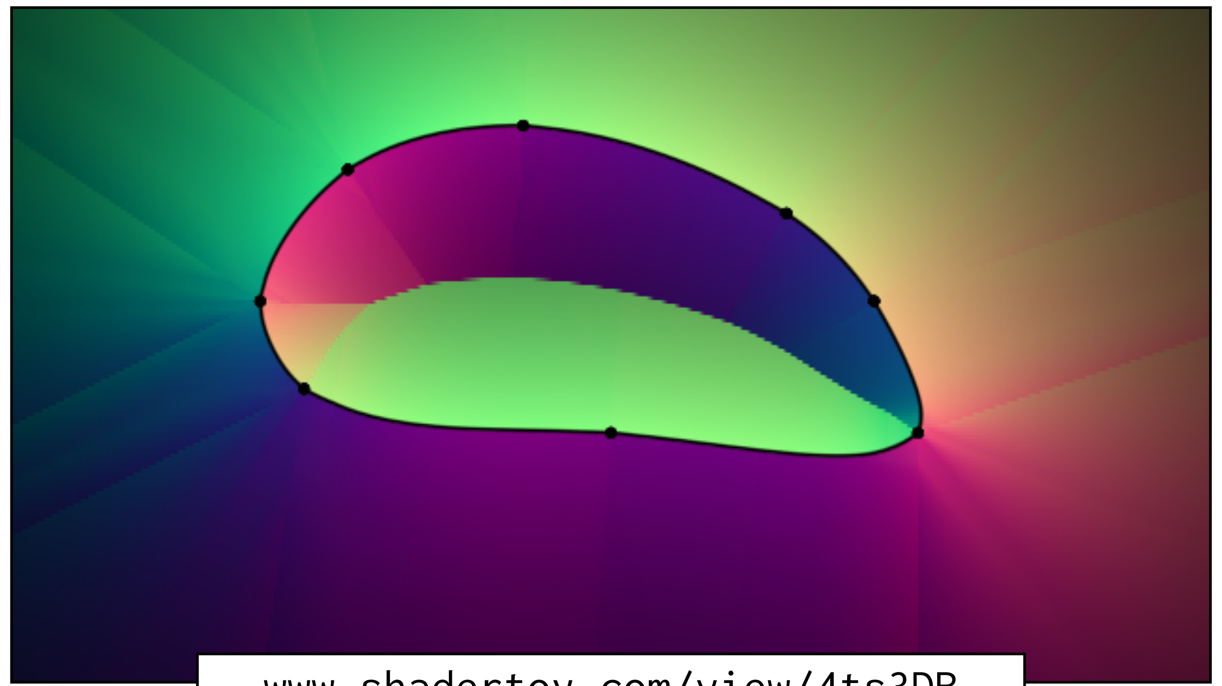

www.shadertoy.com/view/ldXyRn


www.shadertoy.com/view/llyXDV


www.shadertoy.com/view/4dfXDn



www.shadertoy.com/view/XdXGRB


www.shadertoy.com/view/4ts3DB

GENERATIONS
SIGGRAPH2018

Questions?

- Where to find good fonts? dafont.com, fontsquirrel.com
- When does cleartype patent end ? Not clear (ha ha)
- Are hinting and aliasing still needed? Yes
- What are the most complete font families? Noto, DejaVu
- How many languages in the world? Around 7000
- How many scripts in latest unicode (11.0)? 146
- How many characters in latest unicode (11.0)? 137,439
- Is there any forthcoming support for text in Vulkan ? No
- Should I tell my colleagues I love Comic Sans? No
- Are there any open source tools to design fonts? FontForge
- Why “ff” & “fi” disappear when I copy text ? Ligatures
- Should I apply AA before or after gamma correction? Before
- Should I stay or should I go? stay (a few more minutes)

Any other questions?

DIGITAL TYPOGRAPHY

Between art & design.

Gary Hustwit

Raghed Abu Hamdan

Unknown artist

GENERATIONS / VANCOUVER 12-16 AUGUST

SIGGRAPH2018