GPU Font Rendering Current State of the Art

Eric Lengyel, Ph.D. Terathon Software

About the speaker

- Working in game/graphics dev since 1994
 - Previously at Sierra, Apple, Naughty Dog

- Current projects:
 - Slug Library, C4 Engine, The 31st, FGED



About this talk

- Unicode
- Glyphs
- TrueType
- Font Rendering
- Typography

Unicode

- Defines character codes
- Originally 16-bit
- Now has range 0x000000 0x10FFFF
- Divided into 17 "planes"

Basic Multilingual Plane

- 0x0000 0xFFFF
- First 128 code points are ASCII
- Lots of other common scripts/languages
- Lots of common symbols

Basic Multilingual Plane



Supplementary Multilingual Plane

- 0x010000 0x01FFFF
- Rare characters from many languages
- Rare scripts like Cuneiform and Hieroglyphs
- Mathematical symbols and bold / italic
- Emoticons

Supplementary Multilingual Plane



Supplementary Ideographic Plane

- 0x020000 0x02FFFF
- Less common CJK ideographs

Supplementary Ideographic Plane

200	201	202	203	204	205	206	207	208	209	20A	20B	20C	20D	20E	20F
210	211	212	213	214	215	216	217	218	219	21A	21B	21C	21D	21E	21F
220	221	222	223	224	225	226	227	228	229	22A	22B	22C	22D	22E	22F
230	231	232	233	234	235	236	237	238	239	23A	23B	23C	23D	23E	23F
240	241	242	243	244	245	246	247	248	249	24A	24B	24C	24D	24E	24F
250	251	252	253	254	255	256	257	258	259	25A	25B	25C	25D	25E	25F
260	261	262	263	264	265	266	267	268	269	26A	26B	26C	26D	26E	26F
270	271	272	273	274	275	276	277	278	279	27A	27B	27C	27D	27E	27F
280	281	282	283	284	285	286	287	288	289	28A	28B	28C	28D	28E	28F
290	291	292	293	294	295	296	297	298	299	29A	29B	29C	29D	29E	29F
2A0	2A1	2A2	2A3	2A4	2A5	2A6	2A7	2A8	2A9	2AA	2AB	2AC	2AD	2AE	2AF
2B0	2B1	2B2	2B3	2B4	2B5	2B6	2B7	2B8	2B9	2BA	2BB	2BC	2BD	2BE	2BF
2C0	2C1	2C2	2C3	2C4	2C5	2C6	2C7	2C8	2C9	2CA	2CB	2CC	2CD	2CE	2CF
2D0	2D1	2D2	2D3	2D4	2D5	2D6	2D7	2D8	2D9	2DA	2DB	2DC	2DD	2DE	2DF
2E0	2E1	2E2	2E3	2E4	2E5	2E6	2E7	2E8	2E9	2EA	2EB	2EC	2ED	2EE	2EF
2F0	2F1	2F2	2F3	2F4	2F5	2F6	2F7	2F8	2F9	2FA	2FB	2FC	2FD	2FE	2FF

CJK characters
Unallocated code points

As of Unicode 10.0

Other Planes

Planes 0x03 – 0x0D unused

 Plane 0x0E contains special tags and variation selectors

Planes 0x0F and 0x10 for private use only

Character Encoding

- ASCII
- UCS-2 (Universal Coded Character Set)
 - Always 16 bits per character
- UTF-16
 - 16 bits or 32 bits per character
- UTF-32
 - Always 32 bits per character

UTF-8

- 1 4 bytes per character
 - Using variable-length encoding

Values 0x00 – 0x7F identical to ASCII

 High bit set indicates part of multi-byte sequence

UTF-8

- 1 byte: 0x00 0x7F
- 2 bytes: 0x0080 0x07FF
- 3 bytes: 0x0800 0xFFFF
- 4 bytes: 0x010000 0x10FFFF

Glyphs

Fonts contain glyphs

Glyphs have font-specific internal numbering

 Fonts contain tables that map character codes (Unicode values) to glyph indexes

Glyphs

- Fonts typically contain many more glyphs that are not directly mapped from characters
 - Type variations
 - Alternate styles
 - Ligatures, ZWJ sequences
 - Initial, medial, final forms (Arabic)
- More about these later

TrueType

 Contains resolution-independent representations of glyph outlines

Has character-to-glyph mappings

 Usually contains several other tables with typographic information (e.g., kerning)

Glyph Outline

Glyph defined by one or more closed contours

 Each contour defined by continuous sequence of quadratic Bézier curves

 Winding number determines whether a given point is inside the glyph

Winding Number

 Contours defining outer edge of glyph wound in one direction (either CW or CCW is okay)

 Contours defining a hole in the glyph wound in the opposite direction

Winding Number

- Count number of positive loops for outer contours
- Subtract number of negative loops for inner contours
- Nonzero means point inside glyph boundary

0

Glyph Outline / Winding Number



Font Rendering in Games

- Text rendered in lots of places
 - GUI: Buttons, menus, ...
 - HUD: Score, health, ammo, ...
 - In scene: Signs, labels, computer screens, ...
 - Debug info: Console, stats, timings, ...

GPU Font Rendering: Current State of the Art



Basic GPU Font Rendering

 Rasterize each glyph on CPU and store results in a texture map called an "atlas"

• Can be done for multiple font sizes at once

Packing methods can vary in sophistication

```
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[
\]^_`abcdefghijklmnopqrstuvwxyz{
|}~ i¢£¤¥¦§¨©ª«¬-®<sup>-</sup>°±<sup>23</sup>´µ¶'.]<sup>1</sup>°»¼
½¾ċÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏĐÑÒÓÔÕÖ×
ØÙÚÛÜÝÞBàáâãäåæçèéêëìíîïðñòóôõ
ö÷øùúûüýþÿ----`'´``″"•...£€™∂Δ∏Σ-
·√∞[≈≠≡≤≥
```



Image credit: freetype-gl

Render one quad for each glyph

• Texture map the glyph's image from the atlas

Very simple and stupid fast

- Very limited quality
- Only looks good at originally rendered size
- Magnification looks terrible



Minification also problematic

Mipmaps work to a degree

 Glyphs must be surrounded by empty space in atlas to prevent bleeding into neighbors

 Instead of storing glyph images in atlas, store distance to glyph outline at each point

> Image credit: Konstantin Käfer, "Drawing Text with Signed Distance Fields in Mapbox GL", 2014.



 Render linear coverage by scaling distance to pixel units and clamping

 Requires derivatives in pixel shader and extra computation

• Still very fast

- Addresses magnification problem
- Also allows good perspective rendering



Image credit: Chris Green, "Improved Alpha-Tested Magnification for Vector Textures and Special Effects", 2007.

- Need high resolution to capture glyph details
- Sharp corners always rounded off
 - Can be addressed with multiple distance channels
- Minification becomes bigger problem
 - Because one distance value can't account for multiple curves in scaled-down field



Image credit: David Rosen, "High-quality text rendering", 2013.

Resolution Independence

- Render directly from original outline data
 - Control points for quadratic Bézier curves

- No more texture atlases!
 - No resolution-dependent approximation
 - Impossible to lose detail

Loop-Blinn Method

 Creates a triangulation for each glyph using its outline control points

• Each triangle corresponds to one Bézier curve

 Simple calculation based on interpolated texture coordinates yields inside/outside state

Loop-Blinn Method



Image credit: Charles Loop and Jim Blinn, "Resolution Independent Curve Rendering using Programmable Graphics Hardware", 2005.
Needs further subdivision for interior triangles so they never border more than one curve

- Correct antialiasing also requires more triangles in the exterior
 - Consider a pixel intersecting the outline but without its center covered by a triangle



Image credit: Charles Loop and Jim Blinn, "Resolution Independent Curve Rendering using Programmable Graphics Hardware", 2005.

 Requires a large number of triangles for each glyph

More complex glyphs could require 1000s!

Calculation of triangles is complex

Produces high-quality magnification

- However, minification is poor
 - Any pixel is covered by at most one triangle
 - Each triangle corresponds to only one curve
 - Thus, it's impossible for one pixel to consider contribution from multiple nearby curves

Covers each glyph with a single quad

 Pixel shader considers subset of all Bézier curves to determine winding number

Basically ray tracing glyphs

 For a given point, shoot a ray outward and count curve intersections

 An intersection makes a positive or negative contribution based on its winding direction

Nonzero total means inside glyph boundary

Antialiasing possible along ray direction

 If intersection occurs within pixel, it makes a fractional contribution

 Test rays in multiple directions and average to get isotropic antialiasing



 Very slow to test all Bézier curves defining the glyph for each ray

 Dobbie method divides glyph's bounding box into cells

Each cell has list of intersecting curves



Image credit: Will Dobbie, "GPU text rendering with vector textures", 2016.

- Pixel footprint could overlap multiple cells
 - Have to sort that out in pixel shader

- Need to precompute whether cell center inside or outside glyph boundary
 - Then trace extra ray from pixel location to cell center to fix up winding number

• There's a serious problem:

Numerical robustness

 Floating-point round-off error causes rendering artifacts **GPU Font Rendering: Current State of the Art**

May 7, 2018 Irvine, California

Dobbie Method



Glyphy

 Similar to Dobbie method in that a glyph is covered by a single quad

 Pixel shader determines distance to nearest Bézier curve

Glyphy

Original outlines not preserved

Also has numerical robustness problems

May 7, 2018 Irvine, California



Straight lines rounded





GPU Font Rendering: Current State of the Art



Slug Library

- The result of my own research begun in 2016
- Uses one quad per glyph
- Calculates winding number in pixel shader
- Has *perfect* numerical robustness

Numerical Robustness

- Round-off errors in previous methods:
 - Generally come from determining whether roots of ray-curve intersections fall in [0,1] range
 - Problems typically occur at the endpoints
 - Especially bad when ray nearly tangent to curve
 - Hacks like using epsilons or perturbing coordinates just shift the problem cases around

Numerical Robustness

 Only way to solve is to completely eliminate the [0,1] range test

- Slug introduces an equivalence class algorithm
 - Equivalence class represents control point state
 - Same actions taken for all cases in same class

 With respect to a given ray, a particular quadratic Bézier curve is classified into one of 8 possible equivalence classes

- Based on which side of ray each of three control points falls, positive or negative
 - Exactly on ray is considered positive



For each Bézier curve, always calculate roots

$$(y_1 - 2y_2 + y_3)t^2 - 2(y_1 - y_2)t + y_1$$

$$t_1 = \frac{b - \sqrt{b^2 - ac}}{a}$$
 and $t_2 = \frac{b + \sqrt{b^2 - ac}}{a}$

 $a = y_1 - 2y_2 + y_3$ $b = y_1 - y_2$ $c = y_1$

 A 16-bit LUT tells us what to do with roots for each equivalence class (8 classes x 2 roots)

 Action taken only when x coordinate positive at a root, meaning intersection was on ray

Winding Number

- 1 in LUT for first root means add one
- 1 in LUT for second root means subtract one

- Total after considering all curves is winding number at pixel location
- Fractional values used when roots within pixel distance of ray origin

Antialiasing

 Result is coverage value with perfect one-dimensional antialiasing

Evaluate horizontal and vertical rays

Combine to produce 2D antialiasing

 For best performance, we want to minimize number of curves tested

- Cells don't work well
 - Pixel footprint can cover multiple cells
 - Pixels get larger as font size decreases

 Instead of cells, use horizontal and vertical bands that extend to infinity





- Bézier curves are sorted into the bands
 - A curve can belong to multiple bands
 - When rendering, band selected based on ray origin

- Doesn't matter how large pixel footprints get
 - Only matters in ray direction
 - Band parallel to ray extends forever

 Curves in each band are sorted to allow early exit in pixel shader

 Once right-pointing ray's origin is beyond maximum curve x coordinate, we're done

Curves sorted in both directions

 Ray points left or right depending on pixel position within a band

Reduces number of curves tested

 We want worst-case band to contain fewest curves possible

 GPU thread coherence will make shader wait for longest number of loop iterations in a group of pixels (32 or 64)

- Use large number of bands
- Merge those with equal subsets of Bézier curves



Minification

- High-quality minification achieved with adaptive supersampling
 - Based on screen-space derivatives

- Already have perfect 1D antialiasing
 - Take *n* samples in *x* and *y* directions
 - Produces better than $n \ge n$ supersampling

May 7, 2018 Irvine, California

אוני זישר ד 10-4 בירי א. בילודה 4 ו 10-10-4 (א 10-200) ישני זישריל בעליל אלעליגיע ילעעל אוני עשלע

المعرفة والمراجع والمتعادلية المعر المحرب والمتعادية المحاد والمتعادية والمحاد

weight weighter yweightingen geweng beight wij, wei rendende seiner wij weigen jaaren te

Minification

That were in the character, we try with a tendpher in the grand transf "lander" white, where the import or we trans-Records a low of the later of the Velocity of provide an equivalently second couple is the second المعاري المراجع المراجع المراجع والمعرك والمراجع والمراجع والمراجع والمراجع والمراجع والمراجع والمراجع Mandel werden. Dit seinen an Balantan, Berlanden Allen präsynanis für seitenen Brass alteration altere the Statute protects with the state with a second law of the State State of the State of the State of the State factor region is train or a factorizer for page with the training and of the factor branche programm մ 101 թե անվությունը վերջը, ներ է հայ հետո է որ տեղ ուսել, որը են տեղ են երջուց և դեմ են երջուց է են վետ af ber belande hannen bissen bekennten bekenet verstenang ande, im ynt menterant fer anteren bis e dener (A.V. and an) A. when by the strength is approximated interpreted and a solution No well prices while highly independent opposition dependence of a state of a state of a state of the lattice price of the state of the to such held many moning highworks and worked a proof monitor two, many or a strapping to high manual and that, and beignabelink hilderingstrackie antionist part salet ing, fin genational is become you becau an in the second second descent second descent and the second second second second second second second second And a second sector builty of the second ચીંગ 'અનેનાં સાગો એ, અને તેમને આ ગિવાળ ખેતું આ ગુમ્બેલ પ્રાપ્તિમાં, ઉપસ્થ મીસા ખેતમ ખેતું અનેમાં આવ્યું અને અને protections in the state of the line अग्रि प्रवेश स्वरण्डित (Yeslan & Brynnel die gebre (Kader, Britishe) (Ant and Serve (Angle-1976) का المراجع والمراجع Na deserved. Yaa andar ha majar hakayonanit oo ani sib adodha waxaa a kara a aasinaayonana fen alfreisen is ei ferstand tel eller bli eller. Televis alfreist felde ei nen fra presenty dense beneve om en densem hat den er verse disynder inden stende a verde bei oden sten fod af some with the ball with the produces (the bally of a significant sector) part and all angles, and the contribute of and constraining to \$2,5 we down below \$200 King and \$200

an de la se containeage l'he dra acteur, na spectrale de la seconda rade, na angla e la (the, inf property ("the indicated high water,) and 2011 (in the oth

Font Data

- Two texture maps, data only (no images)
- Curve texture, 4 x 16-bit float
 - Contains all Bézier curves
- Band texture, 4 x 16-bit integer
 - Contains curve subsets for all bands
Multicolor Glyphs

- Microsoft fonts use vector data for color emoji
- Layered glyphs with color palette
- Easy to handle with loop in pixel shader



Typography

 Slug algorithm can make individual glyphs look great at any scale or from any perspective

 Higher-level: Make entire lines of text look good

Metrics



Metrics



Kerning

 Some pairs of glyphs appear to the eye to have too much space in between

 Fonts usually contain kerning tables to improve overall appearance **GPU Font Rendering: Current State of the Art**

May 7, 2018 Irvine, California

Kerning

"Too Wavy." "Too Wavy."

Kerning off

Kerning on

Ligatures

 Replaces a sequence of glyphs with one new glyph that looks better

 In some languages, ligatures that change appearance are required for correctness **GPU Font Rendering: Current State of the Art**

May 7, 2018 Irvine, California

Ligatures

The firefly craft.Normal textThe firefly craft.With ligatures

ZWJ Sequences

Unicode has control character
"zero-width joiner" (ZWJ)

 Often used by fonts for combining several glyphs into single ligature **GPU Font Rendering: Current State of the Art**

May 7, 2018 Irvine, California

ZWJ Sequences



Combining Marks

 Unicode defines many accents and other symbols that are designed to combine with a preceding base character

 Fonts determine how this combination works by defining attachment points

Combining Marks



Alternate Substitution

 OpenType fonts define a large array of substitution features

Independent of Unicode

Not directly accessible through characters

Alternate Substitution

- Small caps
- Subscripts and superscripts
- Case-sensitive forms
- Stylistic alternates
- Tabular and proportional figures
- Lining and old-style figures

May 7, 2018 Irvine, California

Small Caps

Text Text

Small caps alternates

Scaled glyphs

Lining and Old-style Figures



Cursive Joining

 In languages like Arabic, letters have multiple forms depending on position in word

• Isolated, initial, medial, final forms

• Do not have separate character codes

Cursive Joining

متقدمة تقديم الخطوتخطيط النص

متقدمة تقديم الخط وتخطيط النصد

Materials

- Rendering glyphs outputs coverage value
 - (Plus color for multi-color emoji)

Can be combined with other materials in game

GPU Font Rendering: Current State of the Art

May 7, 2018 Irvine, California

Materials



Contact / More Info

lengyel@terathon.com

Twitter: @EricLengyel

sluglibrary.com