Getting your CFP abstract accepted: A case study (or 2)
Helps if...

- Established
- ~100s millions users
- Thriving community
- Many high-profile users
- Booming
HarfBuzz
But if...

• Experimental
• Unused so far
• No community
• No users
• Stale
GLyphy
How to keynote LCA
Just cut the jokes out dude!
GLyphy
An experiment in GPU-accelerated text rendering
Status quo

- Hint
- Rasterize
- Upload to GPU texture
- Blit
Transformation dependent
Lets make text beautiful!

lolz
What would you do if you knew you could not fail have a high-resolution display?
Coverage-based Anti-Aliasing
SDF-based Anti-Aliasing
SDF is *linear* over uniform-scaling and translation.
Represent SDF on GPU
Vector all the glyphs!
Distance to Bézier
Ouch!
Convert to line segments
Convert to circular arc splines
Arc-spline approximation
Physics simulation
GPU time!

- Stuff it all in a texture
- Ship it!
You're insane!
Corner cases

- Overlapping contours
- Tangent arcs
- Float preceision
Random access

• Coarse grid
• Various optimizations
It's insane!
Demo time!
Limitations
Memory footprint
Speed + memory
font-dependent
Advantages
Memory footprint
Subpixel positioning
Challenges

- Shader size / complexity
- Pixel cost
- Conditionals
- Dependent texture lookups
- Variable loop iterations
- Interpolation accuracy
varying vec3 lightDir, normal;
uniform sampler2D tex, l3d;

void main()
{
    vec3 ct, cf, c;
    vec4 texel;
    float intensity, at, af, a;

    intensity = max(dot(lightDir, normalize(normal)), 0.0);
    cf = intensity * (gl_FrontMaterial.diffuse).rgb +
        gl_FrontMaterial.ambient.rgb;
    af = gl_FrontMaterial.diffuse.a;

    texel = texture2D(tex, gl_TexCoord[0].st);
    ct = texel.rgb;
    at = texel.a;

    c = cf * ct;
    a = af * at;

    float coef = smoothstep(1.0, 0.2, intensity);
    c += coef * vec3(texture2D(l3d, gl_TexCoord[0].st));

    gl_FragColor = vec4(c, a);
}
/*
 * Copyright 2012 Google, Inc. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * Google Author(s): Behdad Esfahbod, Maysum Panju
 */
#ifndef GLYPHY_TEXTURE1D_FUNC
#define GLYPHY_TEXTURE1D_FUNC glyphy_texture1D_func
#endif

#ifndef GLYPHY_TEXTURE1D_EXTRA_DECLS
#define GLYPHY_TEXTURE1D_EXTRA_DECLS
#endif

#ifndef GLYPHY_TEXTURE1D_EXTRA_ARGS
#define GLYPHY_TEXTURE1D_EXTRA_ARGS
#endif

#ifndef GLYPHY_SDF_TEXTURE1D_FUNC
#define GLYPHY_SDF_TEXTURE1D_FUNC GLYPHY_TEXTURE1D_FUNC
#endif

#ifndef GLYPHY_SDF_TEXTURE1D_EXTRA_DECLS
#define GLYPHY_SDF_TEXTURE1D_EXTRA_DECLS GLYPHY_TEXTURE1D_EXTRA_DECLS
#endif

#ifndef GLYPHY_SDF_TEXTURE1D_EXTRA_ARGS
#define GLYPHY_SDF_TEXTURE1D_EXTRA_ARGS GLYPHY_TEXTURE1D_EXTRA_ARGS
#endif

#ifndef GLYPHY_SDF_TEXTURE1D
#define GLYPHY_SDF_TEXTURE1D(offset) GLYPHY_RGBA(GLYPHY_SDF_TEXTURE1D_FUNC (offset GLYPHY_TEXTURE1D_EXTRA_ARGS))
#endif

#ifndef GLYPHY_MAX_NUM_ENDPOINTS
#define GLYPHY_MAX_NUM_ENDPOINTS
#endif

#define GLYPHY_MAX_NUM_ENDPOINTS 32

#endif  // GLYPHY_MAX_NUM_ENDPOINTS
glyphy_arc_list_t

glyphy_arc_list (const vec2 p, const ivec2 nominal_size GLYPHY_SDF_TEXTURE1D_EXTRA_DECLS)
{
    int cell_offset = glyphy_arc_list_offset (p, nominal_size);
    vec4 arc_list_data = GLYPHY_SDF_TEXTURE1D (cell_offset);
    return glyphy_arc_list_decode (arc_list_data, nominal_size);
}
float
glyphy_sdf (const vec2 p, const ivec2 nominal_size GLYPHY_SDF_TEXTURE1D_EXTRA_DECLS)
{
  glyphy_arc_list_t arc_list = glyphy_arc_list (p, nominal_size GLYPHY_SDF_TEXTURE1D_EXTRA_ARGS);

  /* Short-circuits */
  if (arc_list.num_endpoints == 0) {
    /* far-away cell */
    return GLYPHY_INFINITY * float(arc_list.side);
  } if (arc_list.num_endpoints == -1) {
    /* single-line */
    float angle = arc_list.line_angle;
    vec2 n = vec2 (cos (angle), sin (angle));
    return dot (p - (vec2(nominal_size) * .5), n) - arc_list.line_distance;
  }
}
float side = float(arc_list.side);
float min_dist = GLYPHY_INFINITY;
glyphy_arc_t closest_arc;

glyphy_arc_endpoint_t endpoint_prev, endpoint;
endpoint_prev = glyphy_arc_endpoint_decode(GLYPHY_SDF_TEXTURE1D (arc_list.offset), nominal_size);
for (int i = 1; i < GLYPHY_MAX_NUM_ENDPOINTS; i++)
{
    if (i >= arc_list.num_endpoints) {
        break;
    }
    endpoint = glyphy_arc_endpoint_decode (GLPHY_SDF_TEXTURE1D (arc_list.offset + i), nominal_size);
    glyphy_arc_t a = glyphy_arc_t (endpoint_prev.p, endpoint.p, endpoint.d);
    endpoint_prev = endpoint;
    if (glyphy_isinf (a.d)) continue;
if (glyphy_arc_wedge_contains (a, p))
{
    float sdist = glyphy_arc_wedge_signed_dist (a, p);
    float udist = abs (sdist) * (1. - GLYPHY_EPSILON);
    if (udist <= min_dist) {
        min_dist = udist;
        side = sdist <= 0. ? -1. : +1.;
    }
}
else
{
  float udist = min (distance (p, a.p0), distance (p, a.p1));
  if (udist < min_dist) {
    min_dist = udist;
    side = 0.; /* unsure */
    closest_arc = a;
  } else if (side == 0. && udist == min_dist) {
    /* If this new distance is the same as the current minimum,
     * compare extended distances. Take the sign from the arc
     * with larger extended distance. */
    float old_ext_dist = glyphy_arc_extended_dist (closest_arc, p);
    float new_ext_dist = glyphy_arc_extended_dist (a, p);

    float ext_dist = abs (new_ext_dist) <= abs (old_ext_dist) ?
                  old_ext_dist : new_ext_dist;

    side = sign (ext_dist);
  }
}
}
if (side == 0.) {
// Technically speaking this should not happen, but it does. So try to fix it.
float ext_dist = glyphy_arc_extended_dist (closest_arc, p);
side = sign (ext_dist);
}

return min_dist * side;
Drivers
"Infinite loop detected in fragment program"
Case study
Nvidia + Mac
diff --git a/src/glyphy-common.glsl b/src/glyphy-common.glsl
index 2021d74..95f857b 100644
--- a/src/glyphy-common.glsl
+++ b/src/glyphy-common.glsl
@@ -16,17 +16,6 @@
-
-#define GLYPHY_PASTE_ARGS(prefix, name) prefix ## name
-#define GLYPHY_PASTE(prefix, name) GLYPHY_PASTE_ARGS (prefix, name)
-
-#ifndef GLYPHY_PREFIX
-#define GLYPHY_PREFIX glyphy_
-#endif
-
-#ifndef glyphy
-#define glyphy(name) name
-#endif
-
-@@ -36,13 +25,13 @@

-struct glyphy(arc_t) {
-+struct glyphy_arc_t {
    vec2  p0;

150+fps
Macbook Air 2011
~30fps retina
Case study AMD+Mac
diff --git a/src/glyphy-common.glsl b/src/glyphy-common.glsl
index 5e969c2..c9349b7 100644
--- a/src/glyphy-common.glsl
+++ b/src/glyphy-common.glsl
@@ -151,25 +151,30 @@ glyphy_arc_wedge_contains (const glyphy_arc_t a, const vec2 p)
+float
glyphy_arc_wedge_signed_dist_shallow (const glyphy_arc_t a, const vec2 p)
+{
+    vec2 v = normalize (a.p1 - a.p0);
+    float line_d = dot (p - a.p0, glyphy_perpendicular (v));
+    ...
+    return line_d + r;
+}
+
+float
glyphy_arc_wedge_signed_dist (const glyphy_arc_t a, const vec2 p)
{
-    if (abs (a.d) <= .01)
-    {
-        vec2 v = normalize (a.p1 - a.p0);
-        float line_d = dot (p - a.p0, glyphy_perpendicular (v));
-        ...
-        return line_d + r;
-    }
+    if (abs (a.d) <= .01)
+        return glyphy_arc_wedge_signed_dist_shallow (a, p);
+    vec2 c = glyphy_arc_center (a);
+    return sign (a.d) * (distance (a.p0, c) - distance (p, c));
}
Case study
Intel + Linux
commit 137c5ece7d22bcb017e52f00273b42a191f496d
Author: Eric Anholt <eric@anholt.net>
Date: Wed Apr 11 13:24:22 2012 -0700

i965: Convert live interval computation to using live variable analysis.

Our previous live interval analysis just said that anything in a loop was live for the whole loop. If you had to spill a reg in a loop, then we would consider the unspilled value live across the loop too, so you never made progress by spilling. Eventually it would consider everything in the loop unspillable and fail out.

With the new analysis, things completely deffed and used inside the loop won't be marked live across the loop, so even if you spill/unspill something that used to be live across the loop, you reduce register pressure. But you usually don't even have to spill any more, since our intervals are smaller than before.

This fixes assertion failure trying to compile the shader for the "glyphy" text rasterier and piglit glsl-fs-unroll-explosion.

Improves Unigine Tropics performance 1.3% +/- 0.2% (n=5), by allowing more shaders to be compiled in 16-wide mode.
60+fps

i965 Thinkpad
Case study
iPod 3G
"Demo runs SLOW on my iPod 3G. ~3 FPS"
Case study
Android 4.3+LGE
int

test()
{
    float f = 1.0;
    int i = int(f);
    return i;
}
diff --git a/src/glyphy-sdf.glsl b/src/glyphy-sdf.glsl
index a46c0d4..6cc827c 100644
--- a/src/glyphy-sdf.glsl
+++ b/src/glyphy-sdf.glsl
@@ -36,7 +36,7 @@
 #endif
 #ifndef GLYPHY_SDF_TEXTURE1D
-#define GLYPHY_SDF_TEXTURE1D(offset) GLYPHY_RGBA (GLYPHY_SDF_TEXTURE1D_FUNC (...))
+#define GLYPHY_SDF_TEXTURE1D(offset) GLYPHY_RGBA(GLYPHY_SDF_TEXTURE1D_FUNC (...))
 #endif
25fps
Nexus 4/5
Code: libglyphy
- ~400 lines *.h
- ~2500 lines *.cc *.hh
- ~370 lines *.glsl
- No dependencies
Code: glyphy-demo
• ~2800 lines *.cc *.h
• ~150 lines *.glsl
• FreeType, GLUT
More work

- Subpixel-rendering
- Anisotropic-antialiasing
Gallery!