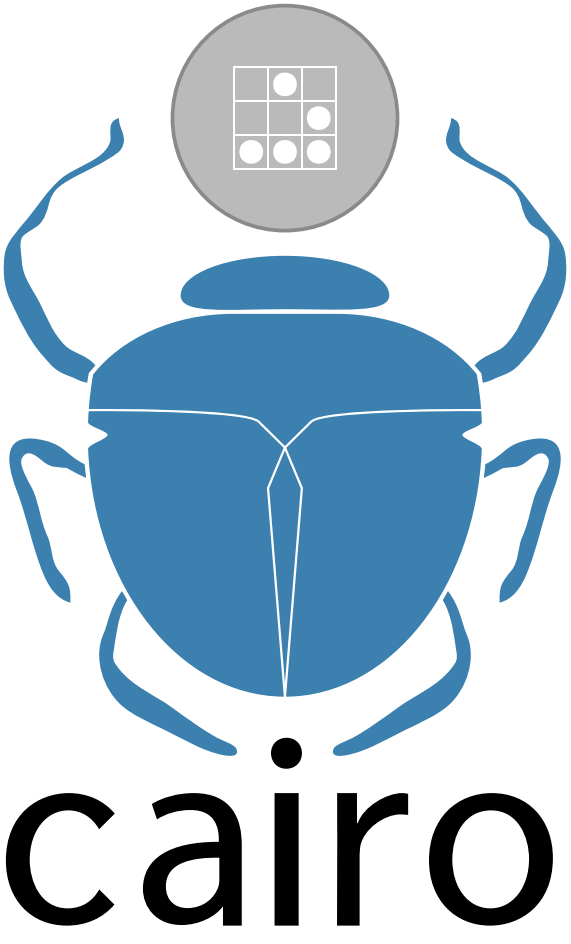
cairo status



Keith Packard

cairographics.org

keithp@keithp.com

2005-06-20

cairo status report

- API stabilization
- Backend Capabilities
- Internal work

API stabilization

- Big transition from 0.4 to 0.5
- Last chance to make things right
- No more changes for 1.0
- Still adding a few things

API Changes

- cairo_create now takes surface
- cairo_matrix_t exposed structure
- added user_data to objects
- callback-based file access
- consistent rendering model
- yet another font API rework
- non-callback cairo_copy_path
- No more PNG surfaces

rendering model

- Target, Source, Mask
- Source and Mask are Patterns
- Target is a Surface
- Patterns are Surfaces, Solids and Gradients
- Target = (Source IN Mask) OP Target

Font API

- cairo_font_face_t - unscaled face
- cairo_scaled_font_t - face at device size
- Separate font matrix from CTM
- Measure text without cairo_t
- Fonts are output-independent

Backends

- X backend mature
- New “real” PDF backend
- Meta surfaces
- “Real” PostScript backend
- Windows backend uses windows fonts
- Mac OS X backend currently broken

Internal work

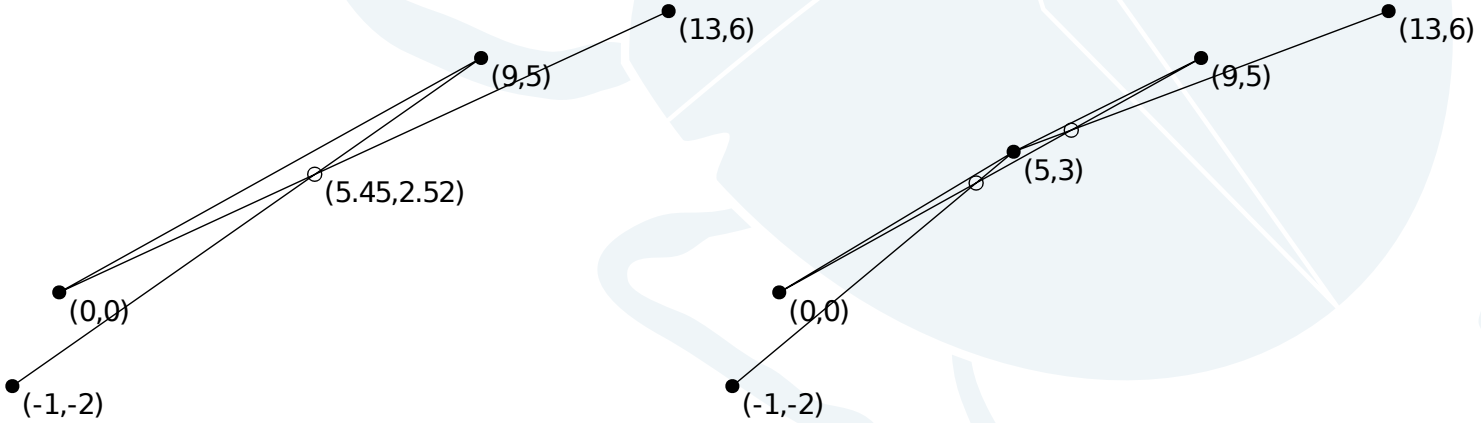
- Tesselation algorithm
- Trapezoid fills
- Compositing code
- Glyph management
- Meta surfaces

Tesselation

- Complex polygons to trapezoids
- Intersections are hard
- Minimizing total edge length
- O(n log n) is possible
- Current code is $O(n^3)$

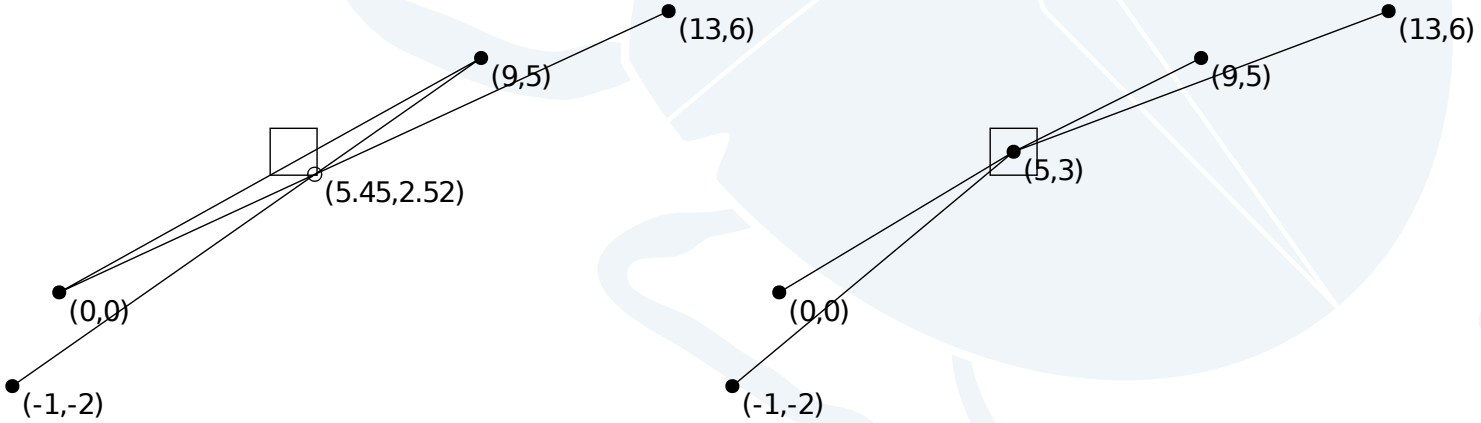
Intersections

- Intersections rounded to coordinate grid
- Moving these creates more intersections
- Number of new intersections unbounded



Solving Intersections

- John Hobby has clever trick
- Box every snapped intersection
- Move every line in the box to the same point



Optimal Tesselation

- Curves take lots of trapezoids
- Many trapezoids in one pixel is slow
- Optimal tesselation minimizes traps per pixel
- A measure of this is total edge length
- Pixel-aligned Rectangles are even faster
- Need to explore many tesselations

Filling Trapezoids

- Render requires point sampled traps
- Exact point sampling seems hard
- Current code is simple, but slow

Compositing Code

- libpixman taken from fb/miregion code
- Should eventually be shared code base
- Hard to add new optimized case
  - Hand-coded case selection
  - Should be machine generated
- General case code really slow
  - Operates one pixel at a time
  - Should operate in larger pieces
  - Perhaps 8×8 “patches”

Glyph Management

- FreeType rasterizes glyphs, computes metrics
- Metrics used in user-space
- Glyph images passed to Render extension
- Image backends re-use glyphs many times
- Goal: Cache reused data, discard temp data

Meta Surfaces

- PostScript surface can't do all of cairo
- Doing everything as pixels is bad
- Solution: MetaSurfaces
- Capture rendering commands in memory
- Replay many times while producing output
- Useful for procedural patterns too
- MetaSurface patch exists, will land soon

Wrap-up

- cairo API now stable
- Only new functions from now on
- Backends coming along
- Internal performance work remains
- 1.0 will be soon (early fall)