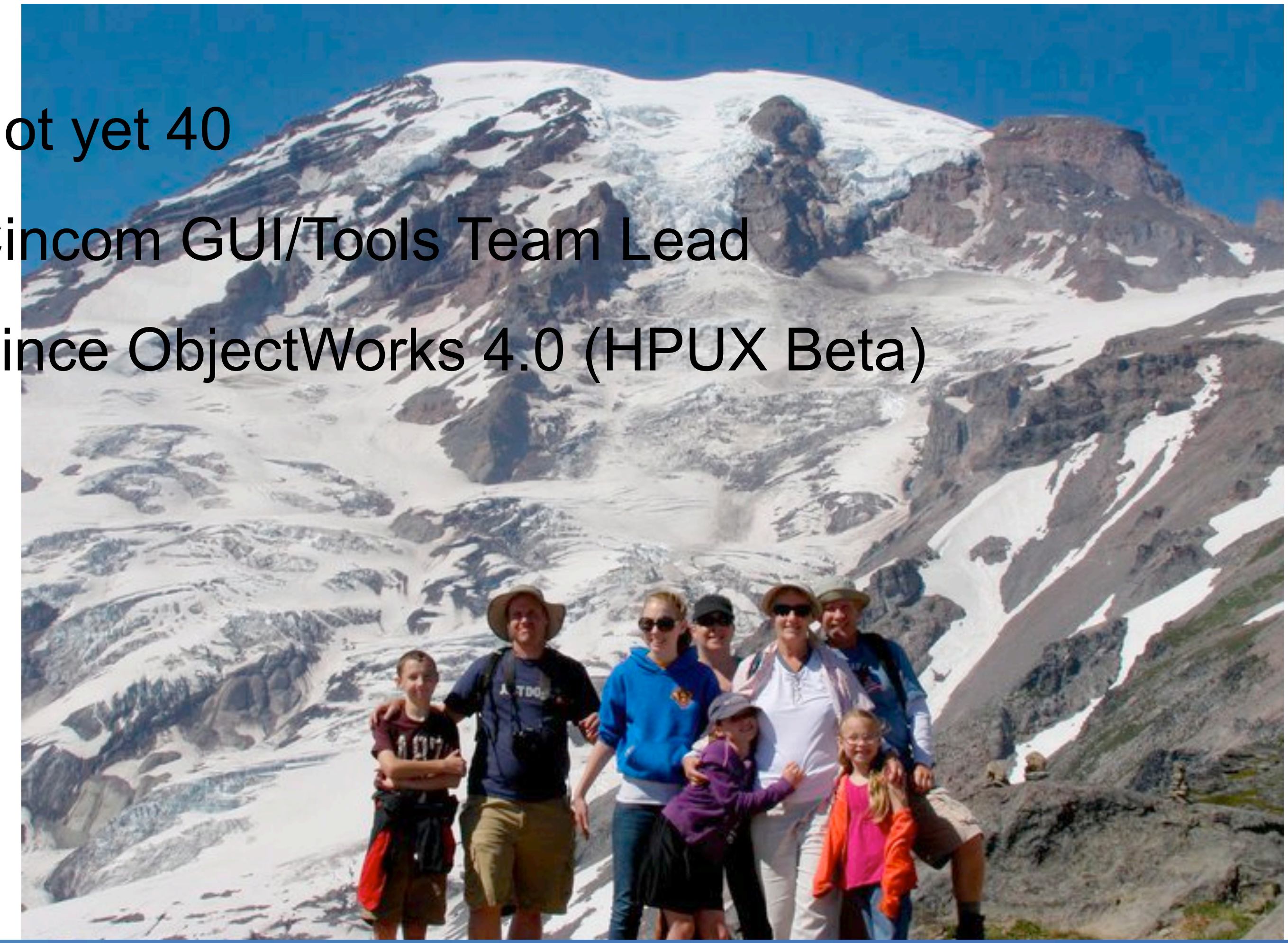
Bonding with Pango

Travis Griggs

GUI/Tools Lead

15 September 2010

Who Am I?



- Not yet 40
- Cincom GUI/Tools Team Lead
- Since ObjectWorks 4.0 (HPUX Beta)

What’s This About?

- Not a Tutorial
- Experience Report
- I’m no expert, I just know more than I used to

What’s Pango?

- Παν語
- “Pango is a library for laying out and rendering of text, with an emphasis on internationalization. Pango can be used anywhere that text layout is needed... ...The integration of Pango with Cairo provides a complete solution with high quality text handling and graphics rendering.”
- Owen Taylor
- Behdad Esfahbod

Our Baseline

- Classic Roman Text Rendering
  - 1:1 code point to grapheme mapping
  - Left to Right
- ComposedText (Paragraph)
  - indenting & tabs (left)
  - word wrapping
  - alignment
  - justification
  - font resolution (underline, bold, italic, color, couple others)

Challenge 1: Glyph Resolution

- Want to show strings of mixed charsets (despite any base font)
- Will it show everything? Yes... if you have enough fonts
- First rite of passage: “The Character Map Viewer”

Challenge 2: Right to Left Rendering

- wef a eman to ,werbeH ,naisreP ,cibarA
- ...sdrow dnasuoht a htrow s’erutcip A

Challenge 3: BIDI

- Bi-Directional text support
- If we mix texts of different directions, we’d like that to look right too
- Yeesh. Yet another “Hello World” example...
- Followed by a more “timely” example...

Challenge 4: Shaping

- The story of Å and Å, the Diacritical Twins
- Back to our sandbox to play some more...

Challenge 5: Vertical Text

- Some writing systems still go right to left in vertical columns (e.g. Asian Print)
- Another example...

Challenge 6: Text Transformation

- In addition to *drawing*, Pango can emit vector information for the outlines of the glyphs it would draw
- Once we have the vectors, we can manipulate it however we want
- More Demos (2 of ‘em)...

Challenge 7: Interacting with Layouts

- For many, just rendering is enough
- If your user will “interact” with these layouts though, you need to be able to translate to/from the input devices the user uses
- Final “Rite of Passage”, building a real editor...

Binding Overview

- Similiar to CairoGraphics binding
- As Faithful as Possible to Pango API names
- Pango coordinates always in scaled integers (x1024), always converted via toPangoScale/fromPangoScale
- Only the “basic” APIs, no need to engage the low level rendering pipeline APIs (yet)
- Only mapped for the Cairo backend

Binding Overview: Layout

- Primary Object; everything ComposedText does ***plus***:
  - Variable line height
  - Forced single line mode
  - Direction control (explicit or automatic)
  - Ellipsification
  - Simple API for setting font/size
  - Spacing
  - 3 kinds of word wrap
  - Various measuring APIs
    - ink and logical extents
    - cursor locating
    - points to positions
    - positions to points

Binding Overview: FontDescription

- Describes a font request, either in full, or partial
- Simple fromString: creator (e.g. ‘Arial, 24’ ‘Mincho, 13px’)
- Can set/get:
  - family
  - gravity
  - pixel size
  - point size
  - stretch (condensed, expanded, etc)
  - style (oblique, normal, italic)
  - variant (normal, smallCaps)
  - weight (boldness)

Binding Overview: LayoutLine

- Additional querying/measurements
- Can be rendered/pathed individually

Binding Overview: Iterator

- Enumerates Layouts
  - by line...
  - or by character...
  - or by cluster...
  - or by run...
- Accesses the current one of any of those
- Other measuring/querying information
  - extents
  - current y values and baseline

Binding Overview: PangoContext

- Usually “just taken care of”
- Can be used to access/set:
  - resolution (ppi)
  - gravity
  - direction
  - default FontDescription
- Query available font families
- Query font metrics

Binding Overview: TabArray

- Manages tab information for a Layout
- Similiar to an Array interface
- APIs for setting left, right, center, and numeric tabs, but currently only actually does left

Binding Overview: Various Constants

- Each Pango ENUM type is turned into a subclass of CairoGraphics.Constant
- ENUM members are expressed as class side methods
- Example:

```
aLayout ellipsization: EllipsizeMode right
```

Binding Overview: AttributeList

- One per Layout
- Analogous to RunArray
- Array like API
- contains Attributes

Binding Overview: Attribute

- Models a Range (or if no start/stop given, does all)
- Analogous to Text emphases
  - backgroundColor
  - *family*
  - fontDescription
  - *foregroundColor*
  - gravity
  - gravityHint
  - language
  - letterSpacing
  - *pixelSize*
  - pointSize
  - rise
  - scale
  - stretch
  - *strikethrough*
  - strikethroughColor
  - *style* (oblique as well as italic)
  - *underline* (but 5 different kinds)
  - underlineColor
  - variant
  - *weight*
- Interned in Pango library, so not extensible

Binding Overview: ShapeAttribute

- Takes a “data” pointer and extents (ink and logical)
- Context’s shape render callback processes them
- block: [:cr :attribute :doPath |]
  ink: aRectangle
  logical: bRectangle
- Blocks registered in a Smalltalk registry and associated with the data value of the attribute, single universal callback finds block associated with ShapeAttribute instance and dispatches it
- Only does text replacement (for now)
- Pictures again?

Binding Overview: Markup

- Attributes can be computed from markup
- Example:

  ‘<b>Hello</b> <i>ESUG</i>’

  ‘Travis is <span size=”x-large”>feeling</span> <span color=”blue”>blue</span>’

Issues: Memory Management

- Borrows same mechanism used by Cairo binding
- Not as consistent
  - some Pango structures are ref counted (like Cairo)
  - some are not refcounted, but still need to be freed if we created them
  - others are just interfaces and need no memory management
- No fun to figure out when it goes wrong

Issues: UTF8

- Can’t Random Access
- Size has to be computed
- Not as difficult tho, when you work in pointers
- Many Pango APIs are done in 0 based byte offsets, which may jump
- No Endianness Issue
- Compact

Issues: Different Platforms

- Pango installed on any current Linux up-to-date distro
- Seems to work well on Windows, a real pain to build there though
- Not good for “non-ascii” on OSX yet (needs CoreText), not so bad to build
- Older Unix/X11 installs... mostly unknown

Cincom’s Plans

- None of what you’ve seen today is “committed” product direction
- Recognition is the first step - Modern International Text Layout is quite involved
- Some Observations:
  - Pango is the “native solution” for Linux world
  - Uniscribe is the “native solution” for Windows world
  - CoreText is the “native solution” for OSX world (>= 10.5)
  - GTK builds and uses Pango on Windows and OSX
- All in Smalltalk instead?
- Pros and Cons with using 3rd Party Libraries

Converting VisualWorks

- Could we just replace VW text rendering primitives with Pango calls?
- Let me show you...

?

Cincom