Rendering

The Future of rendering in GNOME

Owen Taylor
otaylor@redhat.com

GUADEC 5
Kristiansand, Norway
June 28-30, 2004

Outline

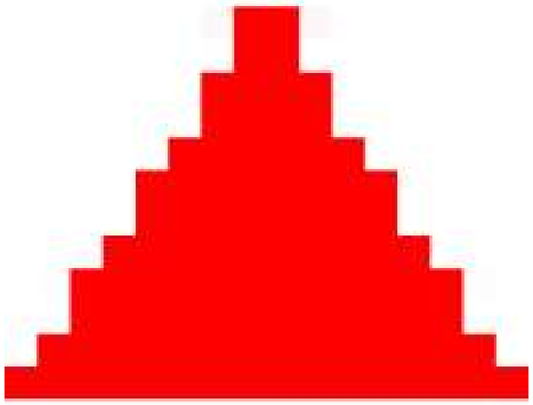
- Current issues
- A new rendering system
- Text
- Alpha Channels
- Printing
- Theming
- Animation

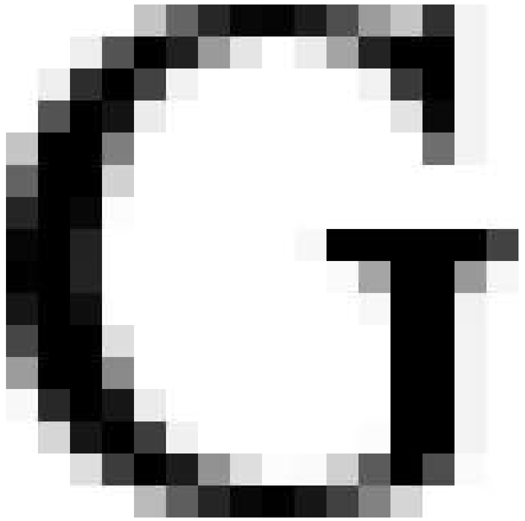
Trends in user interface

- Movement away from strict overlapping windows
  - Popups
  - Alpha transparency
- Proritization of information
  - Computer is actively seeking information
- Explanation
  - Computer is acting on behalf of user

Diverse Rendering

$$\begin{tikzcd}
\text{GDK} \arrow[d] \\
\text{Core X}
\end{tikzcd}$$



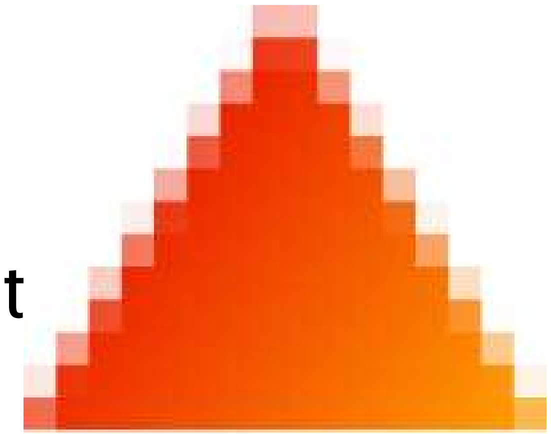


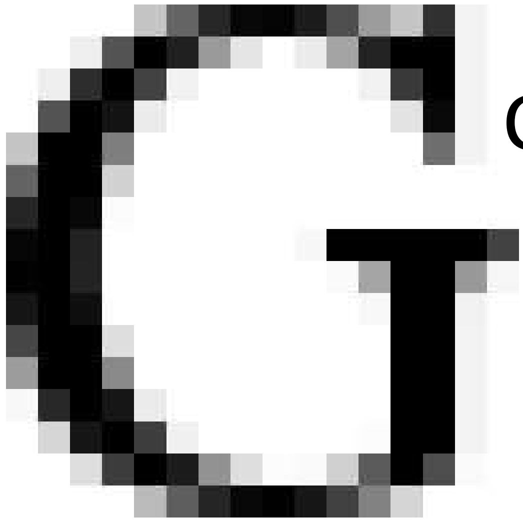
$$\begin{tikzcd}
\text{GDK} \arrow[dd] \arrow[dr] & \\
& \text{Pango} \arrow[dl] \\
\text{Xft} &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{GDK} \arrow[d] \\
\text{RENDER}
\end{tikzcd}$$

Goal

$$\begin{tikzcd}\text{GDK} \arrow[d] \\ \text{Mystery Guest}\end{tikzcd}$$

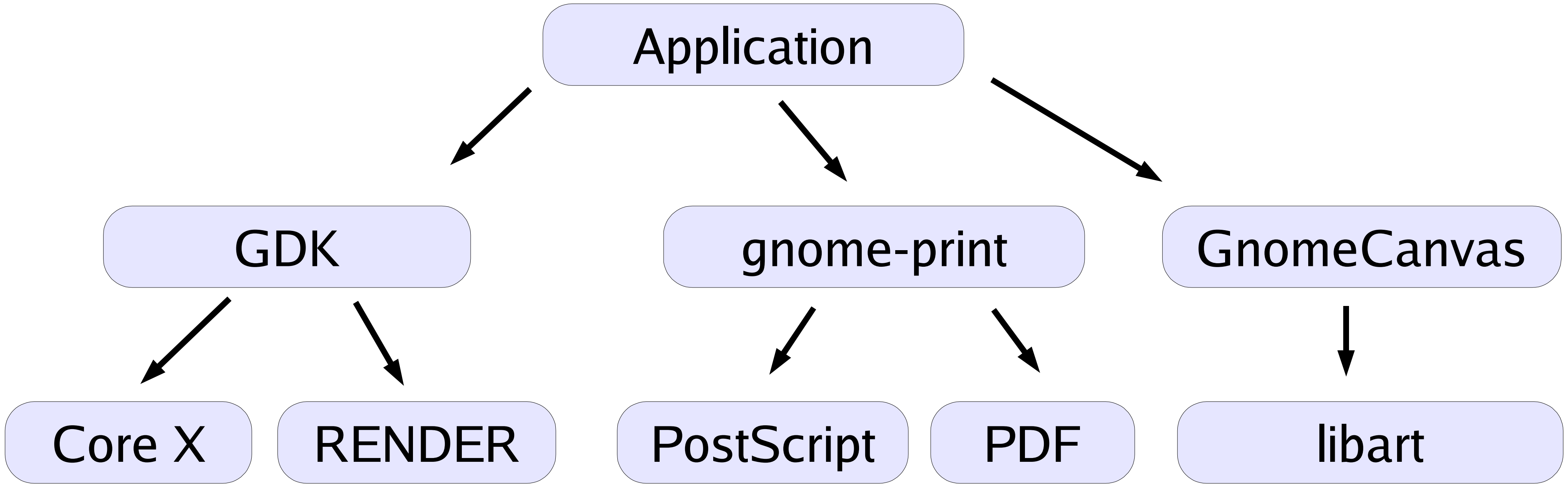




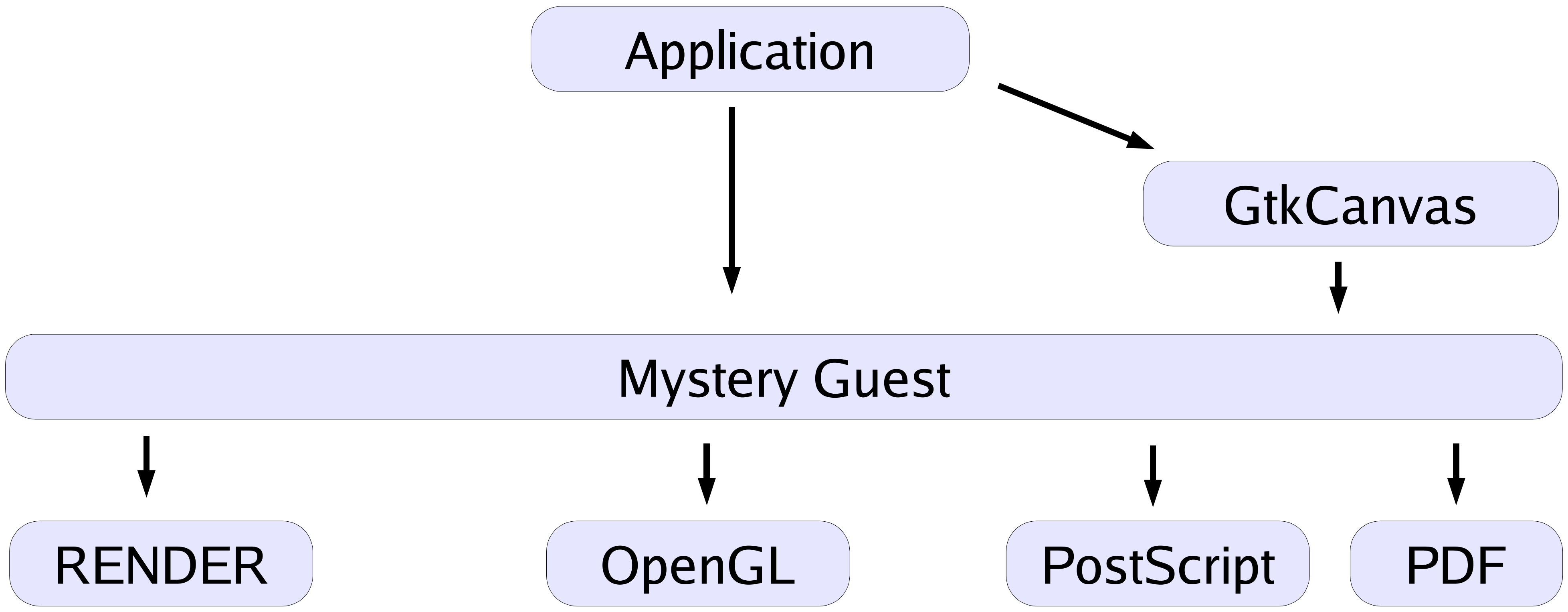
$$\begin{tikzcd}\text{GDK} \arrow[dd] \arrow[dr] & \\ & \text{Pango} \arrow[dl] \\ \text{Mystery Guest} & \end{tikzcd}$$

$$\begin{tikzcd}\text{GDK} \arrow[d] \\ \text{Mystery Guest}\end{tikzcd}$$

Diverse Interfaces



Goal


Application
GtkCanvas
Mystery Guest
RENDER
OpenGL
PostScript
PDF

Better Rendering

- GDK
  - 1987-style rendering + antialiased text, images
- gnome-print, libart
  - alpha-compositing, antialiasing
- Add gradients
- Add different compositing modes
- Hardware acceleration
  - Fast drawing needed for good animation

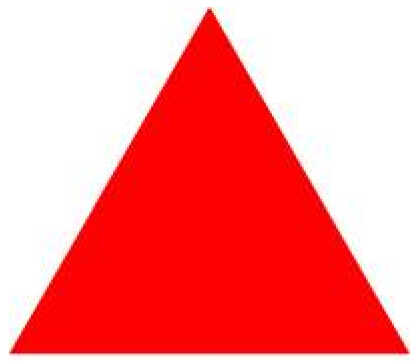
Cairo

- Mostly designed by Carl Worth
- Design goals:
  - Easy to use
  - Rendering model similar to PDF-1.4: alpha-compositing, layers, patterns, gradients,
  - Multiple backends
- Postscript-like programming interface

Cairo Example

- Drawing a triangle

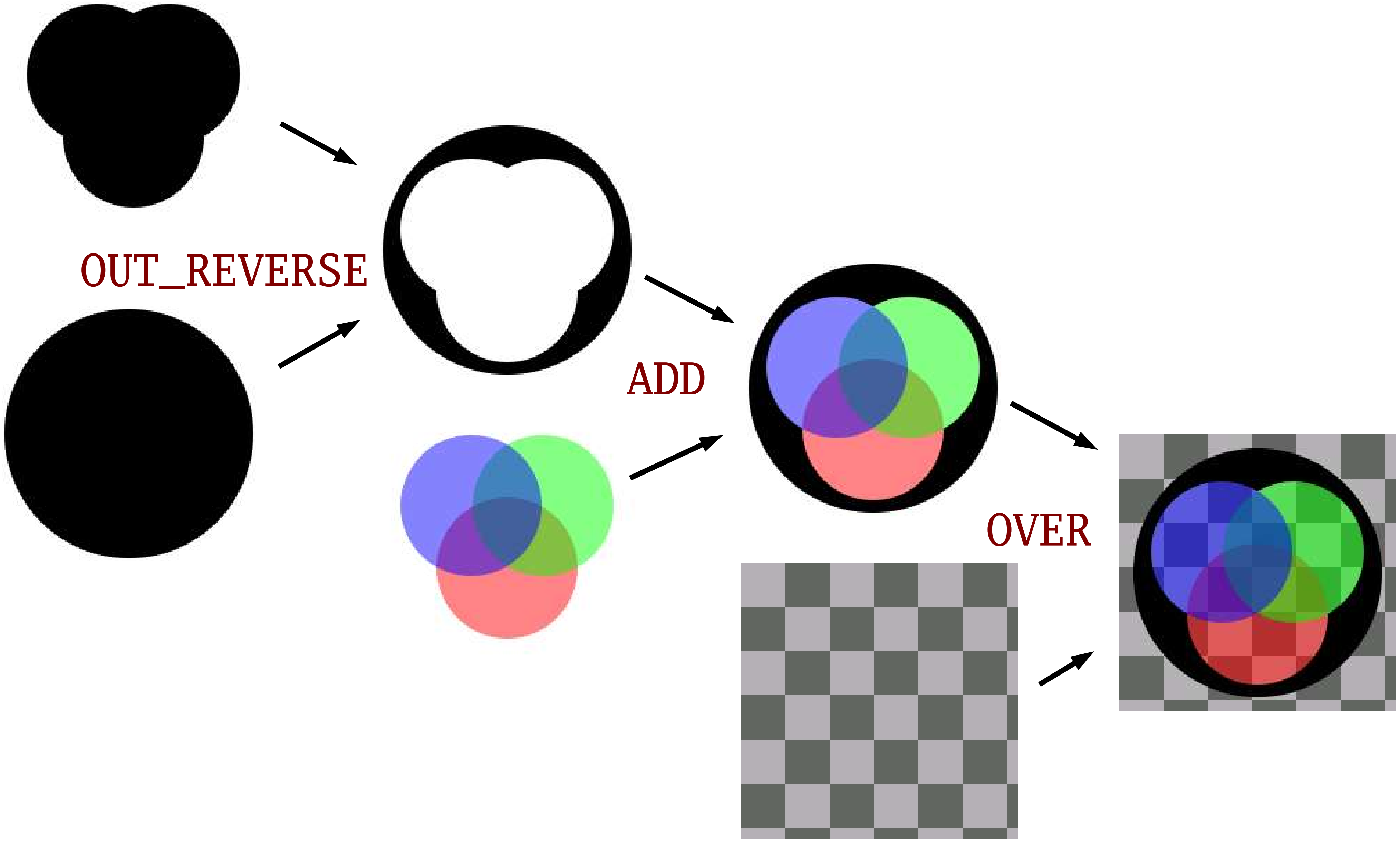
```
void draw_triangle (cairo_t *cr)
{
  cairo_set_rgb_color (cr, 1.0, 0.0, 0.0);
  cairo_move_to (cr, 50,  0);
  cairo_line_to (cr, 100, 87);
  cairo_line_to (cr, 0,   87);
  cairo_close_path (cr);
  cairo_fill (cr);
}
```



Cairo backends

- Local images
- X RENDER extension
- OpenGL (HW accelerated)
- Postscript
  - Just creates big bitmaps currently
  - Needs to be redone to generate text, paths, etc, where possible

Cairo Layer Modes



GTK+ integration

- Xlib wrapped by GTK+

```
void XDrawPoint (Display *display, Drawable d, GC gc,
                 int x, int y);
void gdk_draw_point (GdkDrawable *drawable, GdkGC *gc,
                     int x, int y);
```

  - Hide hard-to-use API
  - Provide cross-platform abstraction

- Not needed for Cairo
  - Application uses Cairo directly

Raw GTK+ integration

```
void
my_widget_expose (GtkWidget      *widget,
                  GdkEventExpose *event)
{
  cairo_t *cr = cairo_create ();
  gdk_drawable_update_cairo (event->window, cr);

  cairo_set_rgb_color (cr, 1.0, 1.0, 0);
  cairo_rectangle (widget->allocation.x,
                   widget->allocation.y,
                   widget->allocation.width,
                   widget->allocation.height);
  cairo_fill (cr);

  cairo_destroy (cr);
}
```

Better GTK+ integration

```
void
my_widget_paint (GtkWidget      *widget,
                 GdkEventExpose *event,
                 cairo_t        *cr)
{
  cairo_set_rgb_color (cr, 1.0, 1.0, 0);
  cairo_rectangle (widget->allocation.x,
                   widget->allocation.y,
                   widget->allocation.width,
                   widget->allocation.height);
  cairo_fill (cr);
}
```

Text Drawing

- Cairo - “Toy API”

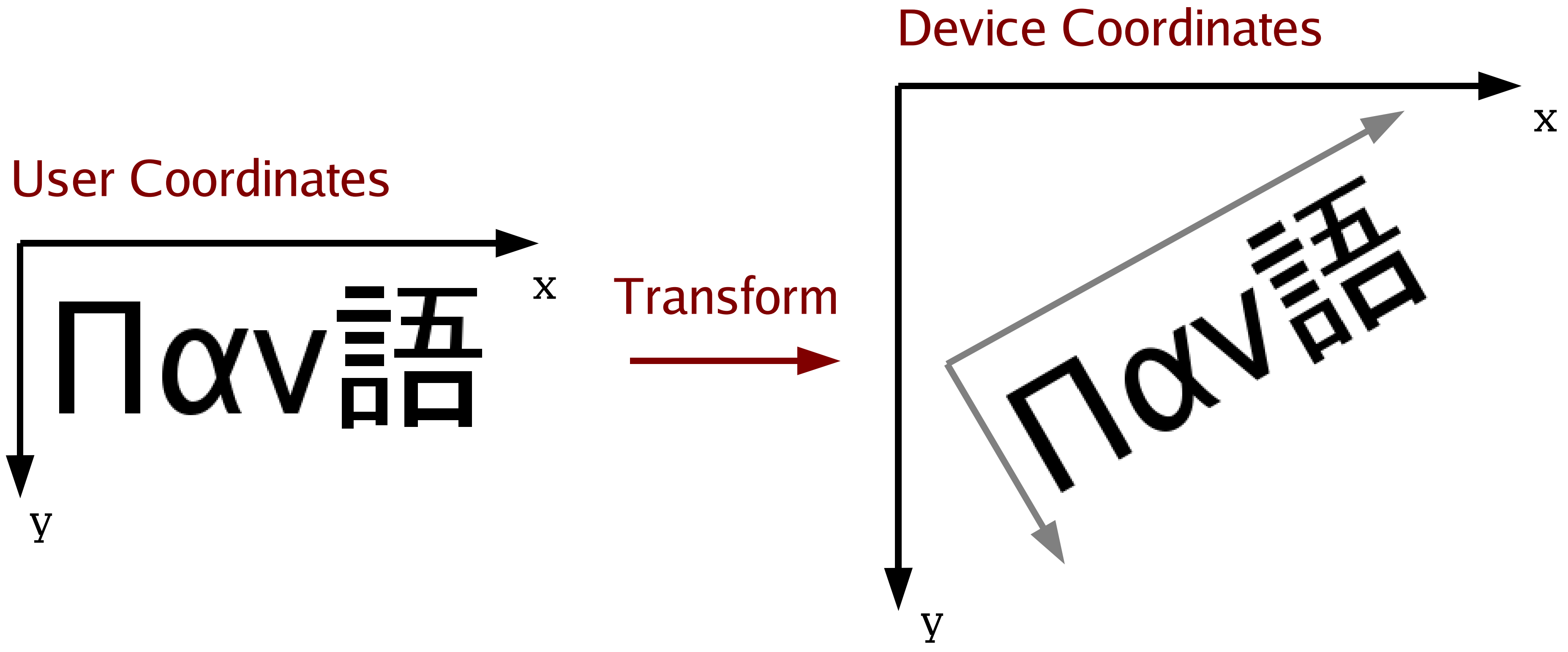
```
cairo_show_text (cr, "Hello Word");
```

- GTK+ apps use Pango instead

```
PangoLayout *layout = pango_cairo_create_layout (cr);
pango_layout_set_text (layout, "Hello world");
pango_cairo_show_layout (cr);
g_object_unref (layout);
```

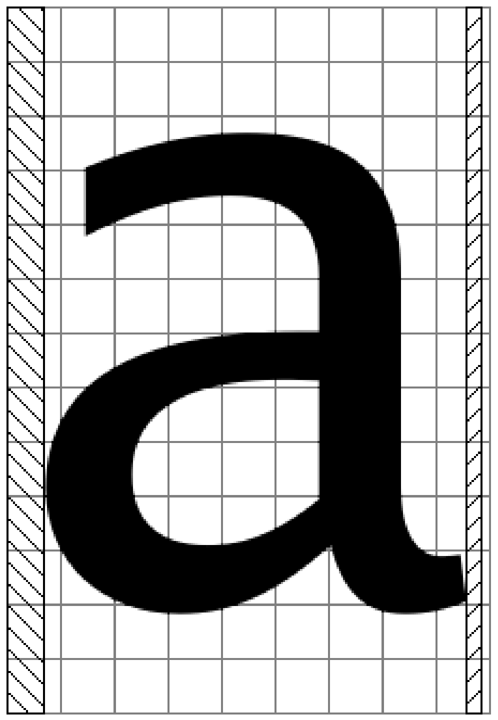
- Full capabilities of Pango
  - internationalization
  - styled text
  - typographic features

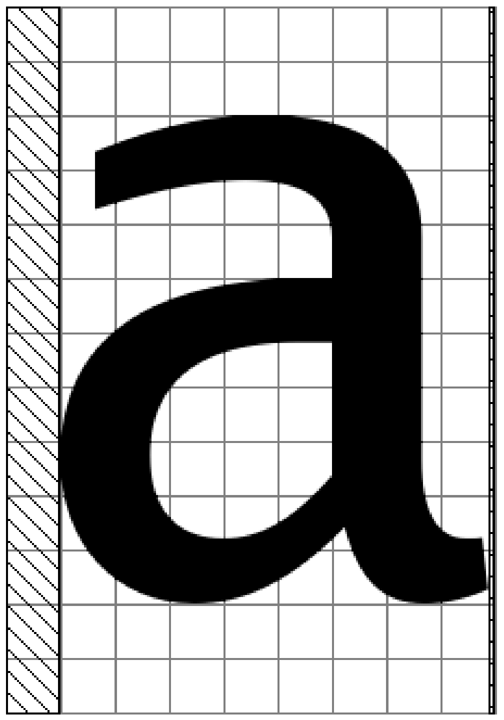
Transforms



- Layout done in user coordinates

Hinting


Linearly Scaled


Fit to Pixel Grid

- *Layout dependent on transform*

Text Details

- PangoContext independent of cairo_t

```
font_map = pango_cairo_get_default_font_map ();
context = pango_cairo_font_map_create_context (font_map);
```

- Need to copy transformation to PangoContext before rendering

```
pango_cairo_context_update (context, cr);
```

- Layout done for particular transformation

```
layout = pango_layout_new (context);
pango_layout_set_text (layout, “Hello World”, -1);
pango_cairo_show_layout (layout);
```

Alpha channels

- COMPOSITE extension
  - replaces fixed window handling with “composite manager”
  - Uses RENDER, OpenGL, etc to draw windows
- Adds visual with an alpha channel
  - Need corresponding GDK extensions

```
GdkVisual *gdk_screen_get_rgba_visual (GdkScreen *screen);
GdkColormap *gdk_screen_get_rgb_colormap (GdkScreen *screen);
void gdk_window_set_rgba_background (GdkWindow *window,
                                     GdkColor  *color,
                                     guint16    alpha);
```

Printing

- Cairo provides backends
- Still need
  - Print selection, page setup dialogs
  - Way to get information about selected printer (Page Size, Color vs. Monochrome)
  - Create Cairo context
- Currently: libgnomeprint, libgnomeprintui
- Belongs in GTK+
  - ~15,000 lines of code
  - Cross-platform abstraction

GTK+ Printing API

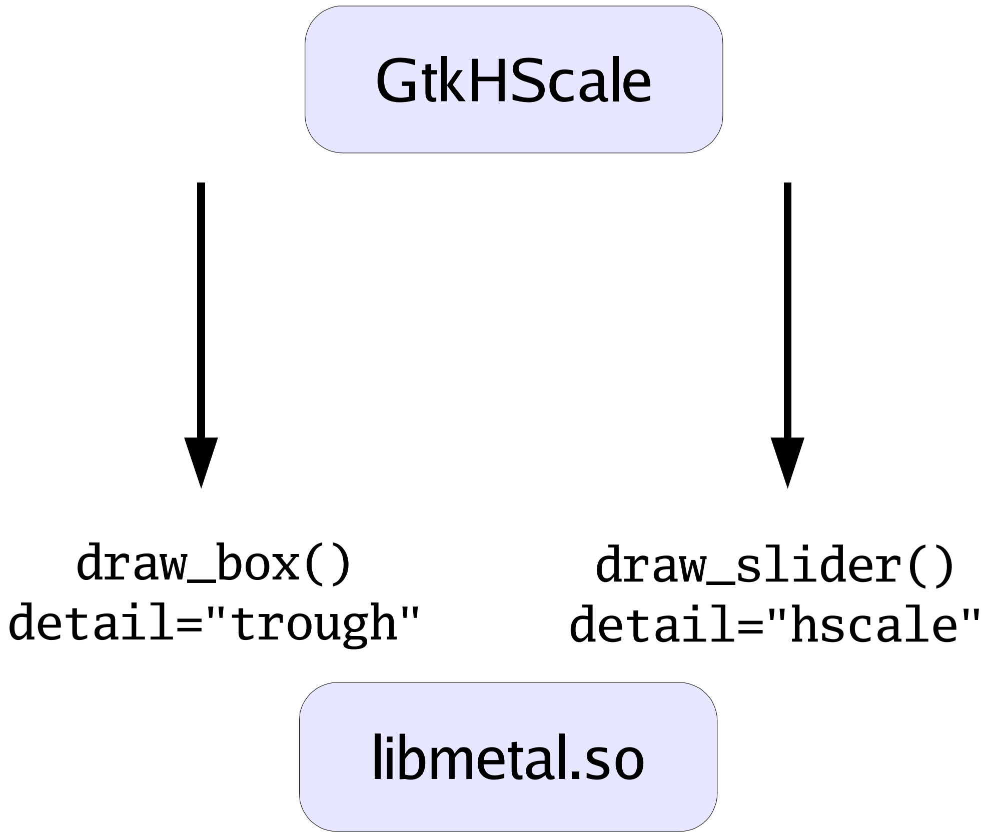
- GtkPrintChooser (...Dialog, ...Widget)
- GtkPrintJob object

```
gtk_print_job_get_page_size (job, &width, &height);
cairo_t *cr = gtk_print_job_get_cairo (job);
```

Theme System

- Needs to be specific to GTK+
  - Themes precisely customize particular widgets
  - Add new widget types to GTK+
- Needs to be general
  - Platform-native theming (GTK-WIMP)
  - Use GTK+ theme system to render other widget sets (OpenOffice, Mozilla)
- Themes have to handle custom widgets
  - Application specific widgets
  - Add-on libraries (libgnomeui, libegg, etc.)

Current Theme System



```
style "metal-scale"
{
  GtkRange::slider_width = 15

  engine "metal" {}
}

class “GtkScale” “metal-scale”
```

gtkrc file

Current Theme System

```
void gtk_paint_box (GtkStyle        *style,
                    GdkWindow       *window,
                    GtkStateType     state_type,
                    GtkShadowType    shadow_type,
                    GdkRectangle    *area,
                    GtkWidget       *widget,
                    const gchar     *detail,
                    gint             x,
                    gint             y,
                    gint             width,
                    gint             height);
```

- Implementing generic functions give “minimal rendering”
- Can special case based on widget pointer, detail string

Theme System Problems

- No specification of detail strings
- Most themes reference widget pointers
  - problem for OpenOffice, Mozilla
- Styles bound to widget classes
  - Can't create widgets that theme like, e.g, GtkEntry
- No concept of layout
  - OpenOffice, Mozilla need to copy lots of code from GTK+ internals

New theme system

- Multi-layered
  - Top layer represents widgets, has idea of layout
  - Bottom layer represents boxes, arrows, etc.
- Declarative
  - config files not code
- Careful specification
  - Multiple producers, multiple consumers
- Standard file formats
  - XML, CSS(?)

Why animate

- Improve “explanation” to user of what is going on
- Make desktop more physical
- Generally want to animate:
  - Changes that occur away from the point of interaction
  - Changes that the user doesn't expect
- Timing tricky
  - Too fast: don't see
  - Too slow: user needs to wait

Animation examples

- Current:
  - Expanders turning
  - Buttons activated through key press
  - Toolbar editing
- Future
  - Expanders opening
  - Smooth scrolling
  - GtkFileChooser pathbar
  - Desensitization

Animation additions

- Way of timing animations
  - Application creates GdkAnimation object
  - Application draws first frame
  - GdkAnimation tracks how progress on X server
  - Application receives "update" signals with new percentage when time for next frame
- Intermediate states for theme drawing?
  - E.g., partially desensitized

Conclusion

- When? GTK+-2.8 (mid-2005)
- More information:
  - These slides: http://people.redhat.com/guadec5/
  - Cairo: http://www.cairographics.org

Discussion topics (Cairo)

- Comparison with PDF/SVG
- Comparison with Avalon (Longhorn drawing)
- 3D integration
- “Pixel shader” type capabilities; expose hardware programmability

Discussion topics (GTK+)

- Usage of SVG in GTK+
- Bevel-explosion and related problems with composite widgets (E.g., GtkScrolledWindow)
- Resolution independence
  - Scaling windows on the fly
  - Padding in non-pixel units
- Changing GTK+ widget rendering to be more retained-mode