GTK+ Graphics

The Future! Or some ideas anyway

Very rough draft notes

Where do people seek GTK+ alternatives?

- Custom visual design of desktop applications (or parts of them)
  - WebKit
  - GtkDrawingArea
  - HippoCanvas
- Devices with custom UI (phones, etc.)
  - Hacked-up GTK+ (theming, other patches)
  - Clutter
- Compositing Manager
  - Compiz object system
  - Clutter

Stuff GTK+ makes hard

- Pixel-precise control of visual design
- Animations and transitions
- 3D effects: shader effects, rotate in/out, etc.
- Nonrectangular or overlapping elements
- Embedding in a nonstandard context (GTK+ insists all widgets are inside a GtkWindow, not some other canvas or composite overlay window, for example)

Goals

- Allow custom visual design, whether for desktop apps or embedded devices; trust people to use tool responsibly
- Support convenient retained-mode graphics, classic example is "Fifteen Game"
- Support smooth animated transitions and effects, including fast transform of images and windows
- 3D effects including shaders and rotation
- Support hardware acceleration for both 2D and 3D in the same layout
- Cleanly make use of existing widgets (Entry, Button, TextView, TreeView, etc.) and don't create a need to reinvent these

Non-goals

- API should support basic 3D effects and use of 3D APIs, but in a somewhat 2D-centric context; not designed for "virtual reality" type of UI
- Model-view should be for special cases (GtkTextBuffer, GtkTreeModel) but generic model-view support is not needed
- Not necessary to support "untrusted" API users (as with X server or Compiz objects); too hard/specialized, GtkWidget does not support this anyhow
- Canvas APIs also add some conveniences and cleanups such as border/padding/alignment for all items, height-for-width, IF_FITS packing, etc.; table these features for later work and focus on more fundamental reasons people are using a canvas rather than GTK

Proposal

Create a new "scene graph" API. Make widgets one kind of scene graph object.

- "Scene graph" seems like a less confusing way to describe this than "canvas" ?
- For this presentation, let's call the scene graph objects "actors" for short, rather than "canvas item"
- "Scene graph" and "actors" intended to emphasize that the new API will have graphics primitives, not interactive UI primitives - GTK+ will continue to provide widgets

What's in the scene graph API?

- An Actor interface
  - paint() method which can use either 2D or 3D operations
  - allocate() method allocates a 2D area
- Per-actor transform
  - Includes scale, rotate, translate
  - Includes z-axis transformation
  - Transform is post-allocate/layout but pre-paint
- Event picking and bubbling
- A Container interface defining the tree of Actor
- Layout-and-paint operation: Request, Allocate, Apply Transforms, Paint to Buffer, Wait for VSync, Swap Buffer
- Replacement for GdkWindow - clipping, scrolling, events
- Fixed-position layout manager and interface to get min and natural size request

Effect on GDK

- What stays
  - Toplevel GdkWindow and related operations
  - Event queue and dispatch of same
  - Display and Screen objects
- What becomes obsolete, replaced by scene graph
  - Child GdkWindows
  - Double-buffer approach used by GTK+ may change
- What remains obsolete
  - X11 leakage (colormaps, etc.)
  - non-Cairo drawing API

Effect on GtkWidget

- Implements the Actor interface
- GtkContainer becomes somewhat obsolete; walking hierarchy with GtkContainer "skips" all non-widget actors, as it does in HippoCanvas
- GtkContainer modified to implement the scene graph's Container interface as well
- child GdkWindow deprecated; back compat is complicated, but ideally there's some way for stock widgets to use scene graph equivalent instead (alternatively, some hack with composite redirection?)
- redraw/repaint idles and double-buffering unified with scene graph
- Widgets can be "composite" (be made up of child actors), or can just paint; maybe all widgets are containers as in Hippo?

Defining GTK+ vs. Scene Graph Layer

- GTK+ contains what we'd normally consider a widget
  - Entry, TextView, TreeView, Button
  - Widgets are rectangular and themed by standard desktop theme
- GTK+ defines input methods since those assume a lot about desktop environment
- GTK+ defines printing, file selector, color picker, etc.
- GTK+ defines toplevel window handling (GtkWindow)
- GTK+ defines keyboard navigation (but scene graph has to do focus, so non-widget actors can be focused)
- GTK+ does not contain the "graphics primitive" actors (image, line, rectangle, etc.), those are in scene graph
- Themes are a property of GTK+ widgets only, not actor in general, so e.g. a Line or Rectangle actor has no theme

What gets passed to paint()?

- Want to be able to draw with Cairo, but also GL
- GL does not support a clean paint() method since there's no equivalent of the Cairo context with a transform stack; lots of GL state is flat-out global - glEnable(FOO) - and where it has a stack, it's a global stack with limited size (glPushMatrix())
- Suggest that some possibly-thin new API defines a paint context that includes both Cairo context and GL state
- GL could also use abstraction for regular vs. ES, and for presence or absence of various extensions
- COGL does some abstraction of GL details but does not provide a context object, uses global state like plain GL
- Need a single double buffer for entire toplevel window and all drawing APIs
- Actors should deal only with their own coordinate system, not global ("window") or parent-actor-relative coordinates

Practicalities

- Evolve the scene graph API as an add-on library, a la Clutter and HippoCanvas
- The add-on should not depend on GTK since it will be a GTK dependency
- The add-on should potentially depend on GDK, though... using GdkEvent would be nice... though both Clutter and HippoCanvas chose to avoid this maybe for good reason
- There should be a "canvas widget" to embed a scene graph
- Allow a scene graph to embed widgets. Maybe just add the Actor interface to GTK_TYPE_WIDGET from the outside library? Otherwise do a HippoCanvasWidget kind of approach (actor containing a widget).

Clutter - where it stands

- Clutter is roughly along the same lines as scene graph API described here; having GL (or other hardware-oriented API) at core seems correct, with Cairo layered in
- Needs request/allocation split apart; patch in bugzilla
- Need to solve problem of what to pass to paint()
- Has ClutterEntry, etc. - should just use GtkEntry
- Currently supports "GDK-free" operation; GTK's scene graph should perhaps depend on GDK, but Clutter would not want to afaik
- Does not support multiple scene graphs in a single process, inherits single global state from GL, but should be simple to fix this
- No child GdkWindow replacement
- No damage region, always repaints everything
- Needs a pass to improve API naming, consistency

Path forward

- Get some consensus on overall approach
- Become familiar with Clutter; decide whether to start with it, if so lay out a roadmap for Clutter changes and seek approval from Clutter maintainers
- Overall if using Clutter, large and incompatible Clutter changes would be expected. Be sure this is acceptable to all.
- Begin or continue work on a Cairo backend that can draw to same double buffer as OpenGL and can employ hardware accel
- Begin work on some kind of context object to pass to paint()
- ... do all the other work

Compositing Manager Footnote

- Have been talking about apps (inside a toplevel) so far
- How should effects and features spanning toplevels work?
  - xsnow type hacks
  - dashboard widget type things
  - window manager features (transitions, drop shadows)
- X server model: object graph with untrusted clients
- Web browser model: clients provide javascript
- Would be an advantage to share scene graph API and widgets with normal apps; could have GtkEntry, etc. in CM
- What about a stack of fullscreen scene graphs as children of composite overlay; each fullscreen overlay hosts a scene graph scripted by JavaScript from some client; when a client exits, destroy JS context and its scene graph
- Proof-of-concept CM with Clutter seems to work nicely