Imendio's Vision on GTK+

March 10, 2008 Berlin GTK+ Hackfest

Introduction

- People like GTK+; GTK+ has a large user base
- These users expect GTK+ to move forward
- Moving forwards means having vision

Imendio's vision?

- So what is our long-term vision for GTK+?
- Looks nicer
  - Animations, physics, stack UI, non-standard UI, 2.5d/3d effects
- Easier to develop for
  - Better UI builders, data abstraction layer, easier to create new widgets, easier to maintain language bindings
- Write once, deploy everywhere
  - Improved back-ends, even more integration with OS specific services

And short-term?

Short-term accomplishments

- Alpha transparency, widget stacking
- Easier layouting (sane defaults, “spreading”)
- Transitions, physics, animations
- Usage of an IDL
- Improved and more powerful theming

Sounds nice, but how?

- We believe, together with many others, that GTK+ 2.x is currently a dead-end (by policy)
- This is because of the promise of not breaking ABI
- The code base is large and contains a lot of unneeded things
- Refactoring GTK+ is very hard and in some cases impossible
- This affects all GTK+ applications

Can we change this?

- We need to have the possibility to break ABI/API in defined intervals
- Really remove deprecated code, don't accumulate it
- Avoid exposing structure fields
- Have very clear policies about this
- Ensure that the entire library stack is parallel installable

Really break ABI compatibility?

- We explored two other approaches, but there are issues with both of them.
- 1) Simply changing namespace (symbol prefix)
  - Causes dual X connections, dual main loops and unsolvable library initialization problems.
- 2) Moving to a new widget system (have two in parallel)
  - Tremendous effort to develop, maintain, integrate a parallel widget hierarchy compatibly; ABI problems remain.
  - Might be easier to write a new toolkit, but then we leave huge user-base and brand behind.

New development policy

- Break API/ABI with every major release (every 4-5 years)
- All depending libraries must be parallel installable
- Remove deprecated API and deprecated code after a defined period of time
- API is deprecated in minor releases and removed in next major release
- Configurable run-time warnings of deprecated code usage to help migration

New development policy (continued)

- Have strict guidelines for new API:
  - No public structure fields
  - Only functions are public API (IDL)
  - Property setters/getters (generated)
- Document valid ABI alterations:
  - Allowed to append functions to interfaces
  - Allowed to change void return values of functions
- Deprecated functions are disabled by default

Roadmap to 3.0 and beyond in short

- Migrating 2.x to 3.0 is not trivial and needs careful planning!
- 2.x will prepare for painless 3.0 migration
- 3.0 puts the new development policy into place, removes all deprecated code.
- 3.x and beyond bring new features

2.x Roadmap

- Provide accessors for all but deprecated public structure fields.
- Provide accessors for the current macro versions of getters/setters (GTK_WIDGET_SET_FLAGS)
- Implement private class data
- Have a diagnostic mode that warns about abuse
- Use GSEAL() to seal public structure fields
  - Rename 2.x fields with GSEAL()
  - Make sealed fields private in 3.0

3.0 Roadmap

- Remove all public structure fields
- Remove all deprecated code
- Applications that have been properly ported to the last 2.x releases will work after a simple recompile

3.x Roadmap

- Look into deprecating more API like non-multihead GDK things, gdk_draw_*, etc
- Add new cool features (short-term accomplishments)
- Incrementally deprecate stuff obsoleted by new features

In General

- This gives us an ecosystem where we can incrementally realize long-term visions
- Subcomponents can be replaced step-by-step
- API can be fixed up later, we know that we will be able to get rid of mistakes in the future

Conclusions

- Current GTK+ is a dead-end
- We all have lots of ideas and visions
- We think a new development policy will help us here and is the best way to move forward
- Our takes on a development policy and roadmap to 3.0 and beyond have been presented
- Future GTK+ will be alive and kicking

Questions / discussion