GTK Toolkit v1 Evaluation
(final version)

Eero Tamminen
<eero.tamminen@movial.fi>

2001-10-24

Abstract

Description of the GTK toolkit and summary of its features compared to embedded platform requirements.

Contents

List of Figures

Part I

GTK toolkit

1 Summary

The combination of GNOME / GTK is one of the two most popular desktop environments / widget sets used on Linux. (KDE / Qt is another.) On commercial UNIX implementations, GNOME / GTK is replacing proprietary user interfaces such as CDE / Motif.

1.1 Advantages

- Modular.
- Has lots of widgets.
- Nice user interface builder.
- Gtk v1 supports internationalization; Gtk v2 also supports Unicode and bi-directional text.
- As it's one of the two main Open Source widget sets, and traditional UNIX companies are moving to use it, GNOME's development should be guaranteed.

1.2 Disadvantages

- No embedded version or build options for smaller configurations are ready. Existing versions depend on X server.
- Very large and complex, especially v2. When tracking development versions, dependency handling is a problem.
- Hard to cross-compile.

1.3 Conclusions

For embedded use on low-end and middle-range devices, the library would need to be seriously pruned. It's uncertain whether it can be made small enough, as there are no building options for this application. The widgets do much more than is required of PDA widgets, and several of them should be rewritten for smaller screen sizes.

If Unicode level 3 support is needed, the only real Open Source alternatives are Qt v3 (just released) and Pango / GTK v2.

1.4 Missing

Some things are missing from this evaluation:

- Locale handling and Unicode-input applications for GTK were not checked.
- Memory use of GtkFB v2 and normal GTK v2 applications were not tested. However, the values listed for GtkFB and DirectFB development versions in the sections 6 and 7 should give some indication of memory requirements.

2 Components

2.1 GTK components

The GTK toolkit[1] is composed of several components that are *separate* projects, but released together (because of the version interdependencies).

2.1.1 GLib

Utility library for GTK. Provides list, tree, hash, thread, and allocation handling; datatypes; type conversions; portability macros and functions; dynamic loading; string, debugging, logging, and profiling utilities; and a lexical scanner.

2.1.2 GDK

Graphics drawing/event library for GTK. Works on both X and Win32 APIs. There are two partial ports for the Linux framebuffer and a (seemingly abandoned) partial Nano-X port.

The GDK library is included into the GTK package.

2.1.3 GTK

User-interface widget library.

2.2 GTK v2

The GTK v2 release schedule is compatible with that of GNOME v2: they were released in the beginning of 2002.

GTK v2 will add accessibility and enhanced internationalization (Unicode / UTF-8 text layout with Pango) frameworks. This will add the following library dependencies.

[1]http://www.gtk.org/

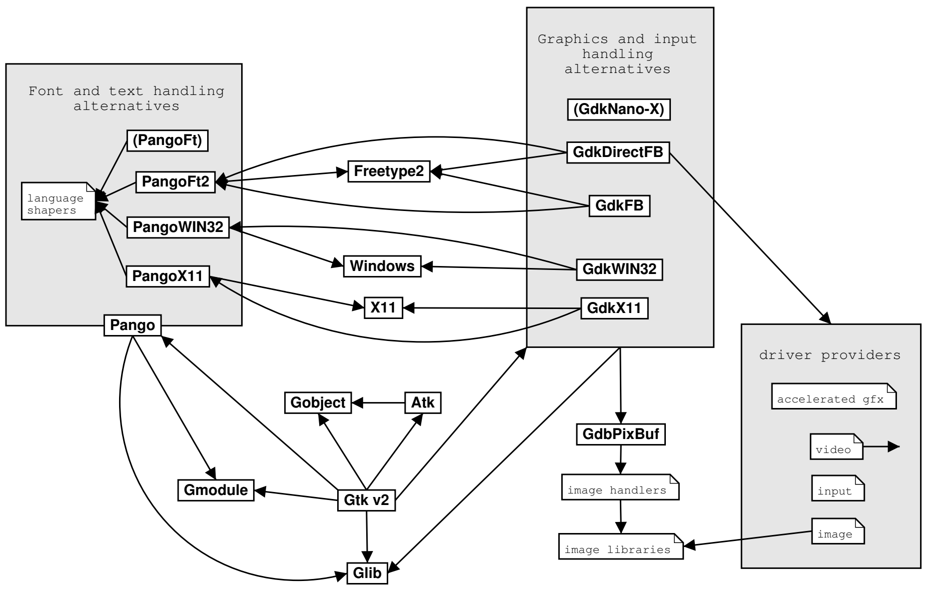


Figure 1: GTK v2 library dependencies

2.2.1 GdkPixbuf

Image-handling library for GDK v2, with dynamically loaded image type-handlers.

2.2.2 ATK

The ATK library provides a set of interfaces for accessibility. By supporting the ATK interfaces, an application or toolkit can be used with tools such as screen readers, magnifiers, and alternative input devices.

2.2.3 Pango[2]

Unicode text layout and rendering library with bi-directional text routines. Replaces all GTK text-output routines. Very modular; there's a module structure for both layout engines and input methods.

2.3 GNOME[3]

GNOME[4] is a desktop environment based on the GTK widget set. It adds the following to that widget set:

- Additional (top level / policy) widgets

[2]http://www.pango.org/
[3]http://www.gnome.org/
[4]GNOME = GNU Network Object Model Environment

- Component architecture (CORBA), embedding, and document object model
- Data storage using XML
- Metadata and mime types support
- Sound, image, and printing support
- Virtual filesystem access
- Database access
- X session management and a few different window managers
- Desktop and office applications
- Build environment

At least some of these are in flux; they depend on other projects that are in development right now (printing, for example). GNOME is intended for desktop computers; it's too large for PDAs.

3 Features

3.1 Generic GTK features

- Object-oriented and implemented in C.
- Offers bindings for other languages (complete support for Ada, C++, Perl, and Python).
- Drag-and-drop (both Motif and X dnd protocols).
- Input methods (X11R6 XIM standard).
- Localization (GNU gettext and locale functions).
- Internationalization (v1.2 supports UTF-8 and text layout comes in v2).
- Thread safe (by using *GLib* functions).
- Theming support.
- V2 will have working Win32 support.

3.2 Unicode and font features

All these concern the *GTK version 2*:

- Unicode 3 support (bi-directional text and languages) using *Pango*.
- Support GUI for directionality. In addition to text, widgets (such as checkboxes) can be either left-to-right or right-to-left.
- *Pango* doesn't support vertical text. The currently supported X font system doesn't provide the information required for high-quality internationalized output (accent positioning, ligature, and alternate glyph forms). Other toolkits are no better in this respect, though.
- *Pango* has a separate module for each language / font-engine combination. Each font engine offers different capabilities for implementing more advanced language-display features (for example, glyph composing for Thai).
- *Pango* reference implementation is based on X fonts, but the basic language shaper also supports Xft (X with FreeType), FreeType2 (used on GtkFB), and Win32.
- FreeType2 supports TrueType, Type1 (Postscript), the OpenType outline font, and the X11 PCF and Windows FNT bitmap font formats.
- Currently, *Pango* uses UTF-8 internally, like the rest of the GTK. Switching into UCS-4 has been discussed.
- *Pango* includes routines from the FriBidi[5] bi-directional text library. (As source for required functions, it doesn't add a dependency for the library.)

3.3 Widgets

GTK has a rich set of widgets. GTK v1.2 contains all the widgets mentioned in our internal embedded widget toolkit requirements. Here are some additional details:

Windows The following options can be toggled: borders, modality, resizability, sticking, destroy with parent. The following parameters can be changed: icon, title, size, position.

Containers Horizontal and vertical box, horizontal and vertical pane, table, notebook (tabs), fixed coordinate and layout (with custom drawing) widgets, alignment and aspect-ratio constraint widgets, frame and separator ornament-container widgets.

Buttons Normal buttons, toggle buttons, check buttons, or radio buttons. You can 'pack' other widgets (text, image) inside a button.

Label Can have multiple lines of text; the text can be justified, word-wrapped, or underlined (with an underline pattern).

[5]http://fribidi.sourceforge.net/

Choice list GTK has menu widgets; they can have tear-off items.

Combobox list GTK has this widget.

Scrollbar Separate widgets for vertical and horizontal orientations. There are also range and scale widgets (with lots of options) for selecting and showing input values.

Progress bar Can work in four different orientations and either continuous or discreet mode.

Control bar Both scale and range widgets. Widget updates can be continuous, discontinuous, or delayed.

Menu Can be created manually from menu item, menu, and menubar widgets, or you can use *ItemFactory*, which creates the menu from an array you provide.

Single-line text input Supports cut-and-paste. Editability and visibility can be toggled.

Multiline text input Supports word wrap, cut-and-paste, different colors and fonts. Doesn't support horizontal scrollbars. Editability can be toggled.

Secret text input Single-line text input with visibility off.

Pop-up dialog GTK has normal dialog and message-dialog widgets. Dialog can be set to be modal.

Image support The GTKImage widget supports all that GDK supports. GDK handles all the required color, etc. for images, and supports both client- and server-side images.

Animation No special support in v1.2.

Structured string input *No widget included for this in GTK v1.2.*

Tabs Notebook widget tabs can be on any of the four edges. Tabs can also be hidden.

Listbox Normal listbox and multicolumn widgets. The latter can have titles on the columns.

Tree Normal tree and multicolumn-tree widgets.

Color selector Color can be selected with RGB and HSV sliders or by selecting from HS-wheel / V-bar. Opacity of the color can also be selected. *This widget doesn't fit into the iPAQ screen.*

File selector dialog Dual-pane file-selection dialog (directory and file panes). *This widget doesn't fit into the iPAQ screen.*

Font selector Font selector and preview. *This widget doesn't fit into the iPAQ screen.*

Calendar This widget has many options (week starts on Monday, show week numbers, show day names, change month).

There was no **character map** widget, but it should be available as a separate application.

3.3.1 Notes

Many more features are found in the GNOME libraries. For example, most GTK 1.2 applications use GNOME libraries for image support; with GNOME and GTK v2, the image support has moved into GTK in the form of the GdkPixbuf.

Focus-change order can be specified between widgets.

3.3.2 GTK v2 widget notes

GTK v2 enhanced theming support includes setting widget borders (helps make widgets smaller for PDAs), colors, drawing functions, icons, fonts.

GTK v1.3 (development version for GTK v2) has the following additional features compared to GTK 1.2 widgets:

Text widget Can have multiple views of the same text, different font styles, text spacing and colors, wrapping, justification, Unicode, and bi-directional text (but not vertical).

Animation The GDK library in GTK v2 supports GdkPixbufs, which handles issues related to animation[6].

3.4 Modularity

The supplied build environment doesn't support smaller toolkit configurations with fewer widgets and (in the case of Pango) fewer language engines. Those configurations have to be added manually to the Gtk and Pango Autotools build files.

3.5 Event Handling

To handle events, GTK uses *signals* instead of callbacks. The difference between signals and events is that signals have nothing to do with input streams; they're just machinery for handling callbacks when something changes. For more information, see the section "Programming GTK" on page 14.

3.6 Error handling

GTK widgets use extensive GLib allocation and list functions[7]. GLib uses its own allocation routines internally. If allocation fails, these functions *log()* it with LOG_LEVEL_ERROR (each of the GTK components has its own GLib logging 'domain'), which GLib logging functions interpret as fatal and *abort()* the program. Calling *abort()* terminates the program unless the signal handler for *SIGBRT* is installed.

[6] For efficiency on a PDA, you might want to send video straight to the framebuffer instead of going through the widget set and window engine.

[7] The GDK 1.2 X input-method code seems to be only exception. It uses malloc and doesn't check the results.

GLib allocation routines have provisions for using external memory debugging and profiling tools.

GTK code seems to check other system calls normally.

4 Tool support

4.1 Compiling

GTK is built using the *configure* script and GNU *make*.

Configure scripts for GTK **v1.2** and its applications resolve their dependencies using *glib-config* and *gtk-config* shell scripts built when those packages are configured.

Configure scripts for GTK **1.3** (v2-dev) and its applications use a tool called *pkgconfig* and its configuration files (which are written out by *configure* in that package) instead.

Note that *configure* scripts are not built for cross-compiling!

Libraries are compiled using gcc. Other compilers were not tested.

4.2 User-interface builder

Glade is a RAD tool to enable quick-and-easy development of user interfaces for the GTK+ toolkit and the GNOME desktop environment. It also contains built-in support for generating the C source code needed to re-create the interfaces. All you need to do is provide the code for the UI callbacks.

The GNOME version has an additional panel for GNOME widgets and a built-in documentation browser.

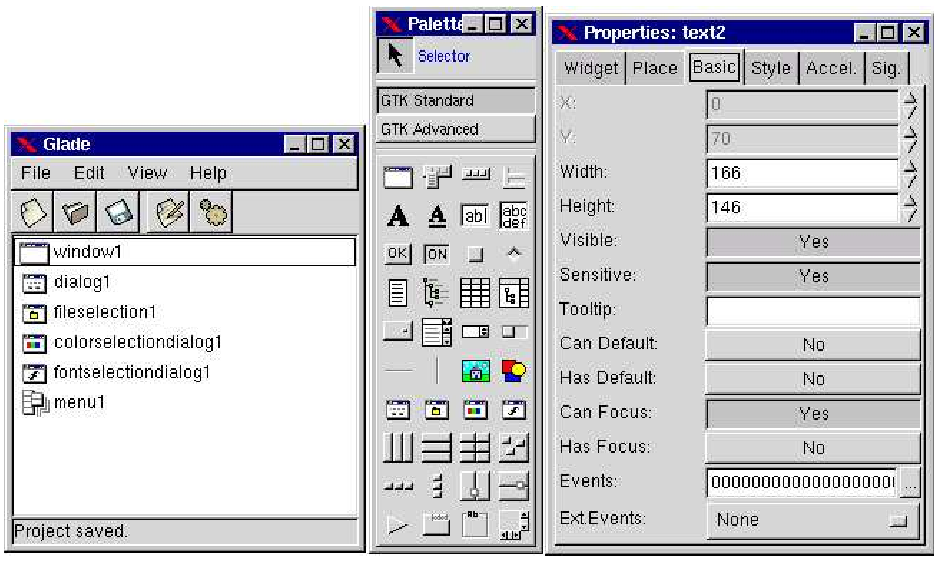
4.2.1 Dependencies and building

The user interfaces designed in Glade are stored in the well-known XML format, enabling easy integration with external tools. Several tools are already available that can turn the XML files into source code in other languages, such as C++, Perl, and Python. Other tools, such as libglade, can load the XML files and *create the interfaces at runtime*. Use of runtime interfaces will add *libglade* and *libxml* dependencies to the application; without them, Glade applications have only the additional *libXi* dependency (30 KB on x86).

To compile programs output by Glade, you need GNU autoconf, automake, make, and a full GTK installation (includes some m4 macros). Glade creates the directory structure, UI description file, gettext files, configure scripts, and UI source code for the application.

4.2.2 Internationalization

Glade supports GNU gettext message catalogs and provides 'configure' and locale code for them. You will also need message-catalog software for extracting the strings into catalogs and translating the message catalogs.



window1
dialog1
fileselection1
colorselectiondialog1
fontselectiondialog1
menu1

Project saved.

X: 0
Y: 70
Width: 166
Height: 146
Visible: Yes
Sensitive: Yes
Tooltip:
Can Default: No
Has Default: No
Can Focus: Yes
Has Focus: No
Events: 0000000000000000000
Ext.Events: None

Figure 2: Glade project, widget palette, and widget properties windows.

4.2.3 Documentation and license

Documentation includes FAQ and User Guides in several languages. In the GNOME version of Glade, these are accessible within the program. Three mailing lists are dedicated to Glade: glade-announce@ximian.com, glade-users@ximian.com, and glade-devel@ximian.com.

Glade's license is GPL, but its documentation says that "you are free to use whatever license you like for the source code generated by Glade".

4.3 Programming GTK

These are the general steps to creating a GTK widget in C:

1. Use one of the various *gtk_*_new* functions to create a new widget.
2. Connect all signals and events you want to use to the appropriate handlers.
3. Set the attributes of the widget.
4. Pack the widget into a container using the appropriate call, such as *gtk_container_add()* or *gtk_box_pack_start()*.
5. Use *gtk_widget_show()* to show the widget.

Although GTK is written in C, it's programmed in an object-oriented manner:

- Widgets are inheritable.
- Events are connected to widgets using signals, like this:
  ```
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
  GTK_SIGNAL_FUNC(callback_function), callback_data);
  ```
- For operations such as signal connecting, objects (C structures) and functions have to be "cast" into their base type. These are the available casting operation macros:
  ```
  GTK_WIDGET(widget)
  GTK_OBJECT(object)
  GTK_SIGNAL_FUNC(function)
  GTK_CONTAINER(container)
  GTK_WINDOW(window)
  GTK_BOX(box)
  ```

The GTK v1.2 widget hierarchy is as follows:

```
GtkObject
  +GtkWidget
  | +GtkMisc
  | | +GtkLabel
  | | | +GtkAccelLabel
```

```
| | | `GtkTipsQuery
| | +GtkArrow
| | +GtkImage
| | `GtkPixmap
| +GtkContainer
| | +GtkBin
| | | +GtkAlignment
| | | +GtkFrame
| | | | `GtkAspectFrame
| | | +GtkButton
| | | | +GtkToggleButton
| | | | | `GtkCheckButton
| | | | |   `GtkRadioButton
| | | | `GtkOptionMenu
| | | +GtkItem
| | | | +GtkMenuItem
| | | | | +GtkCheckMenuItem
| | | | | | `GtkRadioMenuItem
| | | | | `GtkTearoffMenuItem
| | | | +GtkListItem
| | | | `GtkTreeItem
| | | +GtkWindow
| | | | +GtkColorSelectionDialog
| | | | +GtkDialog
| | | | | `GtkInputDialog
| | | | +GtkDrawWindow
| | | | +GtkFileSelection
| | | | +GtkFontSelectionDialog
| | | | `GtkPlug
| | | +GtkEventBox
| | | +GtkHandleBox
| | | +GtkScrolledWindow
| | | `GtkViewport
| | +GtkBox
| | | +GtkButtonBox
| | | | +GtkHButtonBox
| | | | `GtkVButtonBox
| | | +GtkVBox
| | | | +GtkColorSelection
| | | | `GtkGammaCurve
| | | `GtkHBox
| | |   +GtkCombo
| | |   `GtkStatusbar
| | +GtkCList
| | | `GtkCTree
| | +GtkFixed
| | +GtkNotebook
| | | `GtkFontSelection
| | +GtkPaned
| | | +GtkHPaned
| | | `GtkVPaned
| | +GtkLayout
| | +GtkList
| | +GtkMenuShell
| | | +GtkMenuBar
| | | `GtkMenu
| | +GtkPacker
| | +GtkSocket
| | +GtkTable
| | +GtkToolbar
| | `GtkTree
| +GtkCalendar
| +GtkDrawingArea
| | `GtkCurve
| +GtkEditable
| | +GtkEntry
| | | `GtkSpinButton
| | `GtkText
| +GtkRuler
| | +GtkHRuler
| | `GtkVRuler
| +GtkRange
| | +GtkScale
| | | +GtkHScale
| | | `GtkVScale
| | `GtkScrollbar
| |   +GtkHScrollbar
| |   `GtkVScrollbar
| +GtkSeparator
| | +GtkHSeparator
| | `GtkVSeparator
| +GtkPreview
| `GtkProgress
|   `GtkProgressBar
+GtkData
| +GtkAdjustment
| `GtkTooltips
`GtkItemFactory
```

5 GTK v1.2.10 (stable version)

Building is simpler with this version than with v1.3 (there are fewer components), but the configuring required for **GLib** library cross-compiling is still difficult, as its configure script has dozens of places that don't work with cross-compiling (see the section "Cross-compiling GLib v1.2" on page 27).

This version is available from the GTK FTP site `ftp://ftp.gtk.org/pub/gtk/v1.2/`.

5.1 Binary sizes

Everything is compiled with the ’*arm-linux-gcc*’ cross-compiler, using the ’*-Os*’ optimization option; produced binaries are stripped with ’*arm-linux-strip*’ using the ’*-R=.notes -R=.comment*’ options.

The following table lists the dynamically compiled library sizes.

Name	x86	ARM
libglib	106 KB	119 KB
libgmodule	8 KB	8 KB
libgthread	8 KB	8 KB
libgdk	174 KB	186 KB
libgtk	900 KB	1004 KB
Total	1196 KB	1325 KB

The dynamically compiled and stripped “Hello World” application binary size for ARM is 5 KB. The “Glade” (UI builder) binary size is 721 KB.

5.2 Dependencies

The following table shows the libraries that GTK libraries depend on and their binary sizes (on the *Familiar* distribution).

Name	ARM
libXi (Glade)	32 KB
libXext	70 KB
libX11	893 KB
libdl	9 KB
libm	153 KB
libc	1097 KB
ld-linux	442 KB
Total	2696 KB

Image operations are usually done with *imlib*, which is provided with GNOME, so that will be an added requirement, along with any external libraries supporting the required image formats; for example, JPG and PNG. (GIF and uncompressed image formats don’t need an additional library.)

5.3 Memory use

The following table lists the memory usage on **iPAQ** with the *Tiny-X* server that comes with the *Familiar* distribution and applications dynamically linked to GTK, X, and glibc libraries:

Name	VmSize	VmRSS	VmData	VmStk	VmExe	VmLib
Hello World	4204	1950	224	76	4	3484
Calendar	4380	2480	388	76	16	3484
Glade	5500	3608	756	76	688	3516
Xfbdev min	4268	3124	1768	16	832	1348
Xfbdev max	5140	3888	2640	16	832	1348
Xfbdev glade	3920	2700	1420	16	832	1348

The ’Xfbdev max’ setup had 27 small GTK sample applications running at the same time. ’Xfbdev glade’ had freshly started X and slightly used Glade. All of them used the BlackBox window manager and FreeType extension from the Familiar distribution.

6 GTK v1.3.9 (2.0 development version)

Building this version is very complicated and requires a lot of work and time. You must have exactly matching versions of the components for them to be compilable together; in the development branch, the components change often. Cross-compiling is even harder with this version than with GTK v1.2. See the section “Cross-compiling GTK v1.3” on page 27.

Note: *When compiling development versions, clean away other versions of the components and make sure that all the component sources are from the same release. Sometimes you also have to compile external dependencies from source. Development releases before the API freeze have source incompatibilities in every version!*

This version has lots of nice-looking widgets and the best Unicode character output so far. The development version is unstable (selection in a large text widget crashed it), and some of the widget options (such as changing widget order or direction) didn’t seem to update the screen properly afterwards.

This version is available from the GTK FTP site `ftp://ftp.gtk.org/pub/gtk/v1.3/`.

6.1 Binary sizes

The following table lists the GTK libraries on which the *gtk-demo* application depends:

Name	ARM
libgtk-x11	2599 KB
libpango	208 KB
libpango-x	49 KB
libgdk_pixbuf	82 KB
libgobject	267 KB
libgmodule	11 KB
libglib	513 KB
libatk	83 KB
Total	3812 KB

6.2 Dependencies

GTK 1.3 library dependencies are the same as those for GTK v1.2 (see the section "GTK v1.2 dependencies" on page 17).

In this version of GTK, image operations are done with the GDK-PixBuf library and its dynamically loaded extension modules, which may depend on other libraries (such as libjpeg, libpng, and libz).

7 GtkFB

GtkFB is an effort to make GTK+ easier to use on embedded platforms. It allows programs to present a graphical interface without having to make use of the X Window System and its associated overhead.

GtkFB, included with GTK v2, provides GTK with a GDK version for the framebuffer, which borrows its drawing code from X11 sources. At least one project is underway for using this on dedicated devices that need Arabic HTML output on x86[8]. The GtkFB part doesn't seem to be updated with all the GTK v1.3 release changes.

GtkFB doesn't have its own home page, but Red Hat has a "GTK+ for the Linux Framebuffer" white paper[9] on it.

7.1 GtkFB limitations

According to the Red Hat white paper:

- The main limitation is the single-process model. All code in the GTK system must be in the same binary and run in the same process. This means that you can't use processes to separate and protect different parts of the system from each other. It also makes it harder to design larger systems.
- Another problem is that some GTK+ programs make direct X11 API calls when using X11 features that are not supported in GDK. These programs cannot be used with GtkFB without change.
- The framebuffer doesn't have some of the X features, such as graphics hardware support, network transparency, DGA, multiple screen and visual support, and Xv and Xrender extensions.

7.2 Binary sizes

According to the white paper, memory and storage requirements are as follows:

- GtkFB libraries occupy about 2 MB of disk space. Additionally, FreeType is 202 KB, libjpeg is 138 KB, libpng is 126 KB, and libz (needed by libpng) is 58 KB.

[8]http://mail.gnome.org/archives/gtk-i18n-list/2001-October/msg00012.html

[9]The white paper was here but has disappeared:
`http://www.redhat.com/devnet/articles/gtkfb-whitepaper/`

- For the ’testgtk’ program (including several widgets), the RSS size was 3.4 MB. The total virtual memory size was 6.6 MB, out of which 2.3 MB was shared with other processes. Of the total virtual size, 940 KB is the mapped framebuffer, which is not actual RAM.
- Of the total, 72 KB is the mapped Arial font, 112 KB is mapped locale info, 1444 KB is mapped libc code, and 120 KB is the mapped program binary. The heap is 836 KB, and the stack is 24 KB.

It’s worth reading the whole “Memory and Storage Requirements” section in the white paper for its insights on the above data.

7.3 Dependencies

- GLib
- Pango
- FreeType
- libpng, libjpeg, libtiff (these are optional)

8 GTK / DirectFB

DirectFB is a thin library that provides developers with hardware graphics acceleration, input-device handling and abstraction, and an integrated windowing system with support for translucent windows and multiple display layers on top of the Linux framebuffer device. It is a complete hardware abstraction layer with software fallbacks for every graphics operation that is not supported by the underlying hardware.

It’s geared toward digi-TV and MHP, with support for transparency, and allows video streaming into Windows (currently, AVI and MPEG formats are supported). The latest version (as of 2001-10-24), 0.9.6, adds support for iPAQ, for example. DirectFB also uses and supports threading.

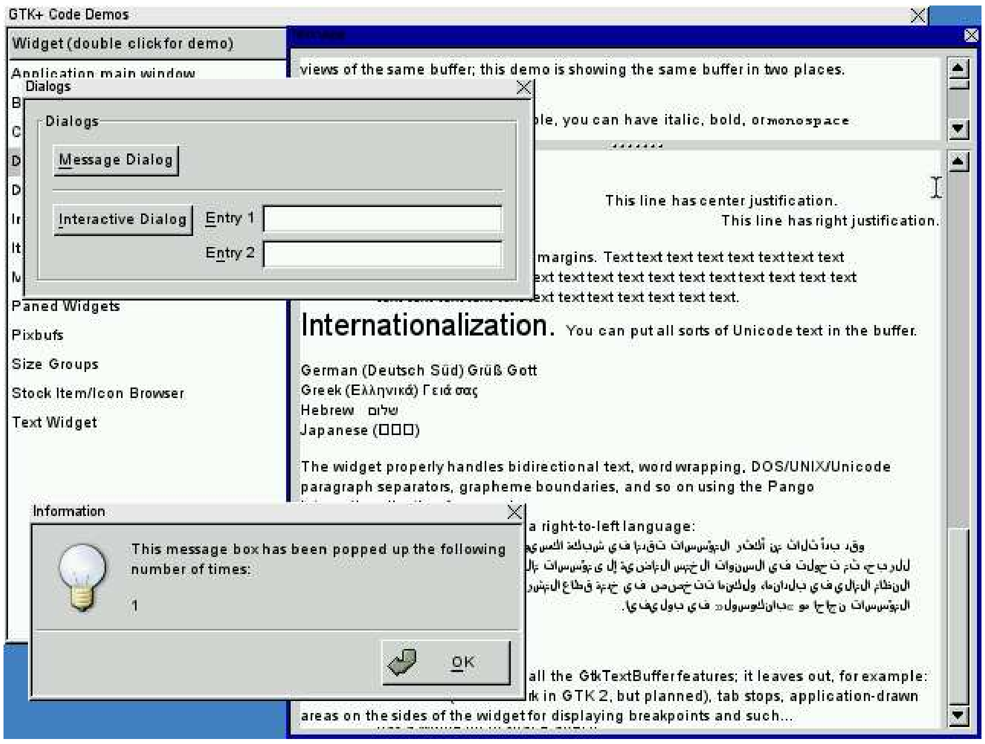
DirectFB is available from `http://www.directfb.org/`.

Video streaming support was not tested. Other testing was done on x86 Linux 2.2.18 (see the section “Building DirectFB” on page 28).

8.1 Notes

The text editing didn’t work properly (probably a GDK-DirectFB font problem) and closing the text editor widget SIGSEGVs the application (probably due to Pango’s immaturity in GTK 1.3.7).

Windows couldn’t be lowered or topped (new windows always opened on top), just moved around. This might not be problem on a PDA.


GTK+ Code Demos

Widget (double click for demo)

Application main window

Paned Widgets
Pixbufs
Size Groups
Stock Item/Icon Browser
Text Widget

Dialogs

Dialogs

Message Dialog

Interactive Dialog

Entry 1

Entry 2

views of the same buffer; this demo is showing the same buffer in two places.

ple, you can have italic, bold, or monospace

This line has center justification.

This line has right justification.

margins. Text text text text text text text text
ext text text text text text text text text text text
ext text text text text text text.

Internationalization. You can put all sorts of Unicode text in the buffer.

German (Deutsch Süd) Grüß Gott
Greek (Ελληνικά) Γειά σας
Hebrew שלום
Japanese (□□□)

The widget properly handles bidirectional text, word wrapping, DOS/UNIX/Unicode paragraph separators, grapheme boundaries, and so on using the Pango

a right-to-left language:
وقد بدأ ثلاث من أكثر المؤسسات تقدما في شبكة اكسي
للربح، ثم تحولت في السنوات الخمس الماضية إلى مؤسسات مال
النظام المالي في بلدانها، ولكنها تتخصص في خدمة قطاع المشر
المؤسسات نجاحا هو »بانكوسول« في بوليفيا.

Information

This message box has been popped up the following number of times:

1

OK

all the GtkTextBuffer features; it leaves out, for example:
rk in GTK 2, but planned), tab stops, application-drawn
areas on the sides of the widget for displaying breakpoints and such...

Figure 3: ’gtk-demo’ on top of GTK / DirectFB

Applications were quite slow on my P133 when compared to X11, which can use 2D acceleration features of my Matrox Millenium I graphics card. (DirectFB used non-accelerated fb-mode.)

DirectFB had troubles with virtual terminal switching (I assume because I used Linux 2.2.18 instead of the recommended 2.4).

8.2 Binary sizes

DirectFB is composed of the main library and different on-demand loaded accelerated graphics, input, image, and video provider driver libraries. The following table shows ones that were compiled:

Name	x86
libdirectfb	140 KB
input/keyboard	9 KB
input/serialmouse	11 KB
input/ps2mouse	8 KB
input/joystick	7 KB
gfx/matrox	39 KB
gfx/ati128	16 KB

Note: Image and video providers are also small, but they depend on additional libraries (for example, the *jpeg* image provider depends on *libjpeg*).

As a test program, I used ’gtk-demo’, which contains most of the GTK widgets and uses several different fonts. Its stripped binary size is 78 KB.

8.3 Dependencies

The test program ’gtk-demo’ has the following dependencies (GTK v13.7) in addition to the required DirectFB libraries:

Name	x86
libgtk-directfb	2211 KB
libgdk-directfb	333 KB
libgdk_pixbuf	73 KB
libpango	208 KB
libpangoft2	44 KB
libfreetype	230 KB
libatk	85 KB
libgobject	250 KB
libgmodule	14 KB
libglib	491 KB
libthread	86 KB
libm	132 KB
libc	1167 KB
ld-linux	91 KB
Total	5415 KB

Notes: DirectFB does its own thread handling; it doesn't use GLib thread utilities. Pango loads its language shapers (10-20 KB) on demand, like DirectFB does with its own input, image, and video providers. FreeType is a TrueType font renderer.

8.4 Memory use

Memory usage was determined with the 'top' program[10]. All the DirectFB applications seem to have three running threads (processes sharing the same memory and data).

After starting, the 'gtk-demo' application had SIZE and RSS of 8 MB and SHARE of 3.4 MB. As can be seen from the "Binary sizes" and "Dependencies" sections, most of these totals come from the libraries. After using the multiview / multifont test editor and image (animation) widgets, application memory usage grew to SIZE and RSS of 10 MB and SHARE of 4.1 MB.

A smaller 'pixbuf-demo' application had SIZE of 7.6 MB and SHARE of 2.7 MB.

Mapped 640x480x16 framebuffer accounts for 641 KB. Font data, locale information, and dynamically loaded modules also account for some of the used memory.

9 Documentation

9.1 Online documentation

GTK has comprehensive developer documentation available online:

- FAQ `http://www.gtk.org/faq/`.

[10]Simo's 'tracker' program is for ARM, not x86.

- Large and comprehensive programming tutorial `http://developer.gnome.org/arch/doc/devel-doc.html`.
- Full API references for *GLib*, *GDK*, and *GTK* in GTK v1.2. GTK v2 has API references covering all of its parts (*GLib, GObject, Pango, GdkPixbuf, GDK, and GTK*). Documentation is available here `http://www.gtk.org/api/`.

A tutorial is available in its source format here `ftp://ftp.gtk.org/pub/gtk/tutorial/` and API references are included with the package sources.

Documents are maintained in *Linuxdoc SGML*, which can be converted to HTML, LaTeX, PDF, and text formats.

9.2 GTK books

Two books have been published on GTK and GNOME application development:

- *Developing Linux Applications with GTK+ and GDK*[11] (New Riders, 1999), by Eric Harlow.
- *GTK+/GNOME Application Development*[12] (New Riders, 1999), by Havoc Pennington. The free version of the book is here `http://developer.gnome.org/doc/GGAD/`.

10 Community

10.1 Organizational support

GTK development seems mainly backed by *Red Hat*[13], as many prominent GTK developers (such as Owen Taylor and Havoc Pennington) are employed there. *Sun Microsystems* has contributed test code to GTK and helps in its localization.

GNOME development is backed by *Ximian*[14] (makes GNOME desktop applications for business use), *Sun Microsystems* (Open Office integration and UI testing[15]), and possibly by *HP*. Sun and HP both intend to offer GNOME as an alternative for CDE on their UNIX OSes.

There's also a nonprofit *GNOME Foundation*[16] that funds and helps direct the future course of GNOME. Its Advisory Board is currently composed of the following organizations and companies: Debian Project, Free Software Foundation, Hewlett-Packard, IBM, MandrakeSoft, Red Flag Linux, Red Hat, Sun Microsystems, Wipro, and Ximian[17].

Almost all Linux distributions now support both GNOME / GTK and KDE / Qt.

[11] The ISBN is 0-7357-0021-4.
[12] The ISBN is 0-7357-0078-8.
[13] http://www.redhat.com/
[14] http://www.ximian.com/
[15] http://developer.gnome.org/projects/gup/ut1_report/
[16] http://www.gnome.org/faqs/gnome-foundation-faq/
[17] At the end of 2001, the GNOME Advisory Board was composed of Borland, Compaq, HP, IBM, Sun Microsystems, Object Management Group, The Debian Project, Free Software Foundation, Gnumatic, MandrakeSoft, Red Flag Linux, Red Hat, TurboLinux, VA Linux, and Ximian.

10.2 Mailing lists

Several mailing lists are devoted to discussion of GTK:

- **gtk-list@gnome.org**: main list
- **gtk-app-devel-list@gnome.org**: application development list
- **gtk-devel-list@gnome.org**: code discussion list
- **gtk-doc-list@gnome.org**: documentation list
- **gtk-i18n-list@gnome.org**: internationalization and localization list (discussions cover Pango implementation, help requests, patches, and announcements about new Pango modules or versions)

The Pango list averages one message a day; the other mailing lists total about 40-80 new messages per day.

10.3 Bugs

Bugs reported to the mailing lists are usually addressed within the same day.

Bugs are reported to Bugzilla `http://bugzilla.gnome.org/`. There are separate categories for each of the libraries and most of the larger applications.

10.4 Testing and quality control

In Open Source projects, quality control is mostly handled by the "release early and often" strategy on the *development branch.*

With graphical user interface toolkits, the trouble is that unit and regression tests work by checking that given input produces required output. In user interfaces, the input comes from the user and the output is something visual.

This means that quality control would be more like test teams, with checklists[18] and examples of correct-looking visuals. *For underlying libraries and components, the tests can be automated:*

- GTK v1.2 includes test code for GLib (for array, list, hash, string, and type operations).
- GTK v1.3 (development version for v2) includes test code for each of the libraries, but it's not complete. There's no build target for testing and testing is not documented.

At the moment, release stability is based on the main developers' wishes and the number of outstanding bugs in the project Bugzilla database. (There are no formal test teams within the project; while there are test teams in companies outside the project, they're not public.)

[18]For example: Open file selector and resize it both larger and smaller. Does it crash? Are there any redraw errors?

11 Terminology

GTK In this document, refers to any GTK version.

GTK+ Name of the current GTK version on GTK documentation. After GTK acquired signal mechanism and widgets became inheritable, a plus (**'+'**) was added after the GTK name. This convention is not used in this documentation.

Part II

Appendixes

A Cross-compiling GLib v1.2

Configuring *GLib* for cross-compiling has proven difficult because its configure script tries to do many of its platform tests by (cross) compiling the test programs and then trying to execute them.

After editing the *glib* configure script and files (it produced a lot) and comparing them against native configuration, I determined the following procedure for “quickly” configuring packages for cross-compilation with broken configure scripts (for this example, arm-linux):

1. Set environment variables correctly for CFLAGS, LDFLAGS, etc.
2. Run ’./configure –host=arm-linux’.
3. Copy the rest of the ’config.cache’ stuff from the native configuration.
4. Run ’./configure –host=arm-linux’.
5. Diff ’config.h’ against the native version and note the differences.
6. Copy ’config.h’ from the native configuration.
7. Diff the configuration results against the native results and fix any obvious errors as in step 5.

B Cross-compiling GTK v1.3

Compilation of different GTK components was done one at a time, first natively and then for the ARM platform. Native compilations usually worked straight away, but ARM compilations had troubles. The ARM configuration (where, for example, ’configure’ tried to run cross-compiled binaries) was patched manually with the following procedure:

1. Copying “generator executables” from the native compilation.
2. Adding include files.
3. Doing directory links.
4. And so on.

One problem, for example, was that images (used internally by the library in PNG format) in the compilation process were converted into C code, for which the ARM environment lacked utilities.

Compiling was helped with home-cooked wrappers that changed PATH, LD_LIBRARY_PATH, LD_RUN_PATH, and PKG_CONFIG_PATH settings appropriately for each compilation.

This solution is not elegant, nor can it be automated.

C Building GTK / DirectFB

Building and running DirectFB v0.9.6 requires the following:

- FreeType2 library.
- Kernel with framebuffer support (preferably better than the plain VESA one) and creation of `/dev/fb*` device files. Kernel v2.4 is recommended by the documentation, although I tried it with v2.2.18.
- Changes are needed to `/dev/tty*` file rights, `/etc/fb.modes` , (the first mode has to be changed to be suitable for your framebuffer), and `.directfbrc` for VT switching. (On my machine, DirectFB couldn't open the keyboard without VT switching being disabled.)

Building and running the DirectFB version of GTK requires GTK component sources from the *same* release from which the DirectFB GDK port was taken (in my case, GTK v1.3.7). I also had to compile FreeType2 from source to get it working with Pango.

To get Pango working, I had to add TrueType fonts here:
`<buildroot>/lib/pango/ft2fonts/`

and add their names here:
`<buildroot>/etc/pango/pangoft2.aliases`

```
# File defining aliases of PangoFontDescription to FreeType2 font set
#
# family style variant weight stretch  list of face names, that should have
#                                      been found in the font files scanned
#
# Currently this file just lists some TrueType face names
#
# Following settings are usable with MicroSoft "Web Core" TrueType fonts
# available (for personal use) from: http://www.microsoft.com/typography/
#
sans normal normal normal normal "arial,verdana,comic sans ms"
serif normal normal normal normal "times new roman,georgia,trebuchet ms"
monospace normal normal normal normal "courier new"
courier normal normal normal normal "courier new"
```

D Document distribution

E More documents

More documents of this type are available from:
`http://www.movial.fi/`

F Changelog

2001-10-01 Initial ASCII version.

2001-10-05 Restructured the document completely, added summary, community, documentation, and gtk v1.2 evaluation sections and started on tools section.

2001-10-11 Added Glade, widgets, C programming, and error-handling sections and Simo's GTK 1.3 statistics. Copied framebuffer versions information from their WWW pages and white papers. Verified TODO items and corrected others.

2001-10-19 Updates for summary, Glade, quality control and testing, features, image handling, DirectFB/GTK, GTK v2 release schedule, and library dependency diagram. GTK 1.3 compilation info from Simo. Glade ARM statistics from Janne. Spell checking.

2001-10-24 Moved remaining TODOs to missing section. *Final version*.

2002-02-21 This document is now part of the Tekes study.

2002-12-19 Prepared the document for public release.

2003-04-28 Folded feedback from a professional editor into the document. The factual information in the document corresponds to the situation at the end of 2001, but that should still be valid.