

GTK+ version 2.0

**Owen Taylor
Red Hat, Inc.**

otaylor@redhat.com

Copyright © 2000 Red Hat, Inc. All rights reserved.
version 1.00
October 4, 2000

Abstract

GTK+ is emerging as a standard for both open-source and commercial software on Linux. Soon to be released GTK+ version 2.0 provides advantages for the user and programmer as well as for people deploying the resulting applications. Users will appreciate the enhanced functionality of existing and new widgets as well as the improvements to the look and feel of the user interface. Programmers will find the powerful new widgets easier to use and more functional. Markets will grow with the ports to additional windowing systems and the enhanced internationalization.

Introduction

GTK started as a toolkit for the GIMP¹ project. It provided the widgets² that the GUI required and derived its name from the name of the project and the word ToolKit. The + was added to signify an enhanced version of GTK with object oriented features. Although it is written as GTK+ it is spoken as GTK (without the +).

Developers choose GTK+ for a number of reasons:

- It provides a modern look and feel and a convenient programming interface. It is flexible and can be easily extended with new types of graphical controls.
- It works well with many different programming languages. GTK+ is written in C, the lingua-franca of the Linux and UNIX environment, and you can use it with C. But it can also be used from C++, Perl, Python, Ada, Eiffel, Ruby, and many other languages.
- It is licensed under the GNU Lesser General Public License (LGPL). This open-source license requires the source for changes to GTK+ itself to be released, but does not impose restrictions on applications that link against GTK+. The LGPL is distinguished from the full GPL which restricts the ability to link proprietary code to the licensed code.

GTK+ has been used in hundreds of applications (www.gtk.org/apps) including many open-source applications such as the GNOME desktop (www.gnome.org), AbiWord, Dia, and, of course, the GIMP. It has also been used in a number of proprietary applications such as Applixware and AOL Instant Messenger for Linux.

GTK+ has been revised many times. GTK+ 1.0 was released in April 1998. Version 1.2 was released about a year later and added a number of new widgets and features including the capability to theme the appearance of the toolkit.³ Version 2.0 is nearing completion and introduces a number of exciting features that are discussed in this paper.

Internationalization

As with most open-source software, contributions to GTK+ have been made by developers around the world, and one of the things that these developers have contributed is improvements to GTK+ for working with different human languages. Already in version 1.2, GTK+ worked with a wide range of languages.

1. GIMP stands for the GNU Image Manipulation Program (www.gimp.org), a program similar to Adobe® Photoshop.®

2. A *widget* is an element of a graphical user interface, such as a push button or text entry box. Widget libraries such as GTK+ contain routines for creating dozens of different types of widgets. Complex widgets can be entire dialogs for selecting a file or color.

3. *Theming* is the ability to change the appearance of the toolkit on the fly by replacing the drawing code and providing alternate sets of colors and images.

The support for internationalization in GTK+ 1.2 was based on the support for internationalization in the X Window System, providing support for European and East Asian languages. However, the support was limited. First, there were languages that it did not support—in particular languages written from right to left such as Arabic and Hebrew, and the languages of South Asia which require sophisticated rendering routines that combine characters based on context. (These languages are often called *complex-text languages*.) In addition, the way that internationalization works using the X internationalization support is not particularly convenient—each locale (country/language) gets its own encoding so it is difficult to write code that works properly in all locales.

Another area that has been improved in GTK+ 2.0 is the handling of *input methods*. An input method provides facilities for converting keystrokes into text. For some languages, this may simply be a matter of handling a few extra keys for inputting accents, but for other languages it may be a much more complex process. In Japanese the user types the pronunciation of a word and then chooses between several words with that pronunciation. The support in GTK+ 1.2 for input methods was tied closely to the support in Xlib. GTK+ 2.0 provides a new set of APIs that abstract away from the low-level Xlib APIs. These new APIs are simpler and have more features than the old ones: GTK+ 2.0's input method support is better for cross-platform usage, can support alternate input method frameworks on X (such as Sun's recently released IIIMF), supports switching input methods on the fly, and is better integrated into GTK+'s widgets.

Pango

It was clear that GTK+ 2.0 needed a considerably more advanced way of handling text than what was provided by X. Support was needed for Unicode,⁴ and for complex-text and right-to-left languages. Rather than adding these features to GTK+, which would have prevented them from being used in other contexts such as printing, a library, named Pango, was created (www.pango.org).

The goal of the Pango project is to provide an open-source framework for the layout and rendering of text that can support many languages. (The name Pango is derived from the Greek word *pan*, meaning *all*, and the Japanese word *go* meaning *language*.) Pango is an offshoot of the GTK+ and GNOME projects and the initial focus is operation in these environments, although there is nothing limiting the use of Pango to these environments. The Pango library provides many features including:

4. Similar to ASCII, Unicode is a standard way to represent characters as integers. While ASCII uses seven bits to represent a character, Unicode uses up to 21 bits, allowing over one million possible different characters. The seven bits of ASCII can be used to hold the characters for English plus punctuation (128 characters); historically this has been extended to eight bits (256 characters) to hold the ASCII characters, plus the characters necessary for one other language, for example, French, Greek, or Russian. But with Unicode, there is enough space available to represent the characters of all of these languages *at the same time*, along with languages such as Japanese and Chinese which each use thousands of characters.

GTK+ version 2.0

- A high-level font API. Instead of referring to fonts by opaque X Logical Font Descriptors (-adobe-helvetica-bold-r-normal--12-120-75-75-p-70-iso8859-1), it uses simple, descriptive names such as “Helvetica Bold 12.”
- A framework for plugging in modules that handle layout for different languages. This allows Pango to be extended to handle different languages without altering its core.
- Unicode support and support for right-to-left and complex-text languages.
- High-level APIs that abstract many of the difficult tasks incurred when handling internationalized text. (For instance, handling insertion point motion in a string of mixed right-to-left Arabic and left-to-right English.)

Pango development has focused on displaying text using traditional X fonts. Support for using true-type fonts directly and for using the system fonts when running on Microsoft Windows has also been written.

The only portions of Pango that need to know about the font technology are the language modules. For X fonts, the language coverage includes (in addition to European Languages and East Asian languages) languages such as Arabic, Hebrew, Hindi, Tamil, and Thai. Soon Pango will be able to handle any language in the world.

New Widgets

GTK+ 2.0 introduces widgets that give a programmer significantly more power and make it easier to accomplish the task at hand. The most important additions to the new version of GTK+ are the *text* and *tree/list* widgets, described in the following paragraphs.

The new text widget is derived from the text widget in the Tk toolkit (Tk is a popular GUI library used with the Tcl language). The Tk text widget has been widely used and is known for its power and flexibility. The new text widget inherits the ability to handle efficiently large buffers of text tagged with many different styles from the original code base and extends the Tk widget in a number of ways:

- The new widget supports multiple views of the same text buffer, as might be found in a text editor that allows split-screen operation.
- The same separation of the back end buffer and front end buffer allows a program to reuse the editing and text-storage facilities of a widget in other contexts where a full editing widget would not be desirable. For example, it is possible to use the back end of the text widget to create an editable text item for a graphics-editing program.

GTK+ version 2.0

- The new widget uses pixel-based scrolling rather than the line-based scrolling of the original widget. In the Tk widget, the increments on the scrollbar represented lines, not pixels, so moving the scrollbar by a small amount could produce a large jump in the position of the displayed text. Pixel-based scrolling yields a more intuitive operation for users.
- The new text widget uses Pango which greatly increases the GTK+ internationalization capabilities. The text widget supports bidirectional text, localized line-break algorithms, and displays feedback from input methods directly inside the widget instead of in a separate window.

Like the new text widget, the new list and tree widgets employ the model-view architecture which enables multiple view widgets to display data from a single data object. This type of architecture is found in many modern toolkits including Java's Swing toolkit. Along with the ability to display multiple views, the model-view architecture offers a great deal of flexibility. The data can be stored in one of the models that GTK+ provides: `GtkListModel` (a flat list) or `GtkTreeModel` (a static tree stored in memory), or to represent large data sets efficiently, a programmer can also create new types of data objects. For example a programmer could create a tree model that represents all of the files in a filesystem. This model could read directories on demand instead of reading them all in advance. There is also flexibility in the way that a data object controls what is displayed on the screen. Along with the data contained within each list row, a data object can store additional properties for rendering such as colors and fonts.

While the new text and list/tree widgets are the most noticeable new widgets, a few other widgets are planned for GTK+ 2.0 as well. In particular, GTK+ 2.0 will include improved combobox widgets, dialog and message-box widgets, and possibly wrap box widgets for arranging child widgets in a flexible manner in a resizable area.

It is not the intent of GTK+ 2.0 to provide every widget that a programmer could desire. One of the strengths of GTK+ has always been the ease with which it has been possible to add new types of widgets. Version 1.2 of GTK+ includes approximately 80 widgets in the core. At least that many add-on widgets are available from other developers. In comparison version 2.0 will add only 4 or 5 widgets. The intent of version 2.0 is to provide a small number of extremely powerful widgets that can be used in a wide range of applications.

Portability

GTK+ was developed in the UNIX and Linux environment and for the X Window System. This environment continues to be the focus for GTK+, but GTK+ 2.0 will also work with a number of other windowing systems including Microsoft Windows. Support is under development for running GTK+ directly on the Linux framebuffer without an external windowing system and for BeOS.

Making GTK+ portable was actually quite simple because GTK+ already contained a library named GDK (the Graphics Drawing Kit), an abstraction layer between the GTK+ widgets and the underlying windowing system. The original focus of GDK was to simplify programming with the X libraries by providing a higher level interface, but because this library insulated the programmer from the details of X programming, it was straightforward to adopt it to work on top of other windowing systems as well.

The ports of GTK+ that are in progress or available fall into two major categories. In the first category are ports to other desktop operating systems such as Win32, MacOS, and BeOS. The existence of these ports will make GTK+ a more attractive target for developers that need cross-platform capability.

The second class of ports are ports intended for different types of devices than desktop computers. In this category is a port that accesses the Linux framebuffer directly and runs without a separate windowing system and a port that runs on top of the Nano-X lightweight windowing system (www.microwindows.org). It is not clear which windowing system will emerge as the windowing environment of choice on these devices running Linux—people are working on reducing the footprint of X so that it can run on these devices and of course GTK+ will work there as well. But whichever windowing environment becomes dominant, GTK+ will run on it.

Base Library Improvements

Many of the new features in GTK+ 2.0 are enabled by changes not to the GTK+ library itself, but in the other libraries that make up the GTK+ programming environment. The Pango library is one of these, the changes to the GDK library that improved portability are another. These are probably the biggest changes, but other major improvements have been made as well.

Along with Pango the **gdk-pixbuf** library, which loads and manipulates images, has also been added to the GTK+ environment. This library supports the PNG, TIF, XPM, and GIF formats. Most applications need to be able to load and display graphical elements of their user interfaces. With **gdk-pixbuf** it takes just few function calls to create a **GdkPixbuf** object from a file or inline data in the program. This **GdkPixbuf** object can then be used in all of the different places that the image is needed: As the icon in a toolbar or the image in a standalone **GtkImage** widget. In addition, **gdk-pixbuf** provides fast functions for scaling and blending images.

As of version 2.0 the basic object system has been separated from the GUI/widget portion of GTK+. Although this change is largely invisible to users, it is a very important change. To understand its importance you need to look at what the object system is.

Object-oriented programming is a design philosophy; you can write an object oriented program in any language, including C, which is said, “not to

support object oriented programming.” GTK+ is written in C and was designed and implemented in a highly object-oriented manner. Instead of building this design on the features of any particular language, object orientation was implemented by creating a set of routines in C, that could be used directly from C, or could be mapped onto the features of other object-oriented languages.

Because many people were interested in using the object system of GTK+ in non-graphical programs (programs that did not use widgets) the GTK+ team has moved the object system into a separate library named **gobject**. As of GTK+ 2.0 the base class of the inheritance hierarchy is no longer GtkObject but is GObject.

By moving the object system away from the GUI/widget portion of GTK+ and into the base environment, it is possible to use the object system throughout GTK+. The object system is used for various objects in GDK, Pango, and **gdk-pixbuf** as well as in GTK+. This increased consistency reduces the amount of work necessary to write a language binding for GTK+. Previously separate glue code was needed to attach a language to widgets and other objects. With GTK+ 2.0, everything is a GObject, greatly reducing the amount of custom code that needs to be written and maintained for each language binding.

At the same time as the object system was moved away from GTK+, it was enhanced. First, support for interfaces similar to those found in Java was added. Second, support for dynamic loading and unloading of object types at run time was added, allowing new types of widgets to be defined in dynamically loaded modules.

The Programmer Interface

GTK+ 2.0 is designed to be easier for programmers to use.

- Confusing and hard-to-use interfaces from previous versions have been deprecated and replaced with easier-to-use interfaces. For example, the new text and list widgets replace widgets that people found difficult to use. A new clipboard interface provides a simple way of implementing cut-and-paste; previously this could only be done using an arcane set of routines scattered throughout GTK+.
- Code that programmers had to duplicate repeatedly has been added in new routines and convenience widgets. For example GTK+ 2.0 will contain dialog and message box widgets that simplify the process of displaying user messages.
- In some cases GTK+ has been simplified by removing (or deprecating) functions that were not useful and made it hard to figure out how to use a feature. For example the progress-bar widgets in GTK+ 1.2 included about 25 different methods; in GTK+ 2.0, there are 6 easy-to-use methods that can do almost anything required by a program.

The User Interface

In addition to features for the programmer, GTK+ 2.0 provides an improved user interface. Some of the improvements come from the changes that have already been discussed: GTK+ will work with more languages because of its bidirectional support for layout and rendering, text and list widgets will have more features because of their new implementation, and users will see a more consistent interface between GTK+ programs because of the standard routines used to create dialogs and message boxes.

An integral part of creating GTK+ 2.0 is a thorough review of the user interface that it generates which asks the question, “How can GTK+ make it easier for the user to interact with the program written with GTK+?” Because the GTK+ 2.0 development environment needs to be stabilized before the details of the user interface can be finalized, the process of designing and coding the user interface is not as far along as the changes to the programming interface. However work is well under way in improving mouse navigation through menus, improving keyboard navigation throughout the GTK+ interface, and changing the default theme of GTK+ to be more attractive, and to have clearer check and radio button indicators.

Conclusion

The most important new feature of GTK+ 2.0 is the increase in its range of applicability. Because it can be applied in many situations it will be used more frequently. GTK+ 2.0 can work with more

- **Human languages** The improvements provided by Pango and the improved input support allow users around the world to enjoy the benefits of GTK+ in their native languages.
- **Programming languages** The enhanced GObject library will make tying GTK+ to different programming languages even easier than it was before.
- **Platforms** With the new GDK framework for porting to different windowing systems, GTK+ 2.0 goes beyond GTK+'s traditional strength on X and UNIX/Linux to a broad set of systems.
- **Situations** The new widgets and enhanced programming environment in GTK+ 2.0 make it easy for programmers to create a wide variety of applications.

A release date for GTK+ 2.0 has not yet been set. In September, 2000, a preview copy was shipped with Red Hat Linux 7. The API for GTK+ 2.0 is almost finalized and a beta release that developers can begin to use in porting their applications should be available within a couple of months.