CPSC 427a: Object-Oriented Programming

Michael J. Fischer
Y. Richard Yang

Lecture 22
November 19, 2009

User Interfaces

Modern computer systems support two primary general-purpose user interaces:

Command line: User input is via a command line typed at the keyboard. Output is character-based and goes to a physical or simulated typewriter-like terminal.

Graphical User Interface (GUI): User input is via a pointing device (mouse), button clicks, and keyboard. Output is graphical and goes to a window on the screen.

Interfaces for C++

The C++ standard specifies a command line interface: `iostream` and associated packages. No similar standard exists for GUIs.

De facto GUI standards in the Linux world are `GTK+` (used by the Gnome desktop) and `QT` (used by the KDE desktop).

`GTK+` is based on C; `QT` is based on an extension of `C++` and requires a special preprocessor.

`gtkmm` is a `C++` wrapper on top of `GTK+`.

Advantages: Provides type safety, polymorphism, and subclassing.
Provides a native interface to C++ code.

Disadvantages: Big and complicated.
Components not well integrated.
Documentation spread between `gtkmm` and `gtk+`.

Overall Structure of a GUI

A GUI manages one or more *windows*.

Each window displays one or more *widgets*.
These are objects that provide graphical and textual input and output to the program.

A GUI package such as gtkmm maintains a *widget tree*.

A widget controls a particular kind of user input or output.
Examples: label, text box, drawing area, button, scroll bar, etc.

Widgets can emit events, which cause user code to be executed.

Structure of gtkmm

gtkmm is built from several libraries and packages:

- gtk-2.4 is the GUI engine.
- gdk is a device layer used by gtk.
- cairo is the drawing package used to draw on the screen.
- pango is a library for laying out and rendering of text, with an emphasis on internationalization.
- libsig+ is a library for connecting events (signals) to event handlers (slots).

Compiling a gtkmm program

Many include files and libraries are needed to compile and build a gtkmm program.

A utility pkg-config is used to generate the necessary command line for the compiler.

```
> pkg-config gtkmm-2.4 --cflags
 -I/usr/include/gtkmm-2.4 -I/usr/lib64/gtkmm-2.4/include
 -I/usr/include/glibmm-2.4 -I/usr/lib64/glibmm-2.4/include
 -I/usr/include/giomm-2.4 -I/usr/lib64/giomm-2.4/include
 -I/usr/include/gdkmm-2.4 -I/usr/lib64/gdkmm-2.4/include
 -I/usr/include/pangomm-1.4 -I/usr/include/atkmm-1.6
 -I/usr/include/gtk-2.0 -I/usr/include/sigc++-2.0
 -I/usr/lib64/sigc++-2.0/include -I/usr/include/glib-2.0
 -I/usr/lib64/glib-2.0/include -I/usr/lib64/gtk-2.0/include
 -I/usr/include/cairomm-1.0 -I/usr/include/pango-1.0
 -I/usr/include/cairo -I/usr/include/pixman-1
 -I/usr/include/freetype2 -I/usr/isnclude/libpng12
 -I/usr/include/atk-1.0
```

Linking a gtkmm program

pkg-config also generates the necessary linker flags.

```
> pkg-config gtkmm-2.4 --libs
 -lgtkmm-2.4 -lgiomm-2.4 -lgdkmm-2.4 -latkmm-1.6
 -lgtk-x11-2.0 -lpangomm-1.4 -lcairomm-1.0 -lglibmm-2.4
 -lsigc-2.0 -lgdk-x11-2.0 -latk-1.0 -lgio-2.0 -lpangoft2-1.0
 -lgdk_pixbuf-2.0 -lpangocairo-1.0 -lcairo -lpango-1.0
 -lfreetype -lfontconfig -lgobject-2.0 -lgmodule-2.0
 -lglib-2.0
```

To use package config, use the backquote operator on the g++ command line:
Compiling: `g++ -c ‘pkg-config gtkmm-2.4 --cflags‘ ...`
Linking: `g++ ‘pkg-config gtkmm-2.4 --libs‘ ...`
Both: `g++ ‘pkg-config gtkmm-2.4 --cflags --libs‘ ...`

Using a GUI

The following steps are involved in creating and using gtkmm:

1. Initialize gtkmm.
2. Create a window.
3. Create and lay out widgets within the window.
4. Connect user code to events.
5. Show the widgets.
6. Enter the main event loop.

The GUI then displays the window and waits for events.
When an event occurs, the corresponding user code is run.
When the event handler returns, the GUI waits for the next event.

Example: clock

The code example 22-clock is an unfinished extension of the clock example in the gtkmm tutorial book.

It illustrates many of the features of gtkmm.

Main program

```
#include <gtkmm/main.h>
#include "clockwin.h"

int main(int argc, char** argv)
{
   Gtk::Main kit(argc, argv);
   ClockWin win;          // custom window with several widgets
   Gtk::Main::run(win); // start main event loop
   return 0;
}
```