

CSCE 155 - Graphical User Interfaces & Event Driven Programming

Lecture Notes

1. Introduction

- Up to now: Command Line Interface (CLI) programs, sequential with specified control flow
- Most human-interaction (HCI) is done through Graphical User Interfaces (GUI)
- Web-based (browser) or thick-clients
- GUIs are a form of event-based programming

2. History

- Windowing systems; attempts to emulate the physical desktop
- Xerox Alto (1973) – first mouse driven GUI
- Apple: Lisa (1983), Macintosh (1978, 1986)
- Windows (1.0, 1985) though DOS had some GUI support
- 90s: GUIs made home computing ubiquitous rather than just for hobbyists
- Currently: the browser has become a platform itself, mobile devices have their own GUI APIs

3. Frameworks

- C: GTK
- C++: wxWidgets, GTK+, Qt
- Java: Swing

4. GUI Components

a. Windowing System

- Handles low-level functionality: hardware (mouse, keyboard) interaction and graphics rendering

b. Window Manager

- Manages interaction of applications, windows, and widgets
- Controls the flow by registering and handling *events* and their consequences
- APIs abstract this management away so we don't need to worry about how this works

c. Misc: Windows, Menus, Icons, Tabs

d. **Widgets**: Interactive elements

- Examples: Pointer, text boxes, buttons, lists, radio/checkboxes, etc.
-

5. Control Flow

- GUI/Event Based programming is user-centric: user determines control flow through interaction of widgets

a. Modal Interaction

- At any one point, the user has a certain number of choices of what to do
- May be dependent on prior actions
- Certain, invalid choices may be restricted (open a file of a specific type)
- Certain actions may be done *asynchronously*: sequence of actions may be done in a different order by different users or differently by the same user

b. Widget hierarchy (parent-child relationship)

c. Events

- An event is an action that is initiated outside the scope of a program but that is *handled* by the program.
- Events are asynchronous (can be initiated at any time and in multiple different orders); window manager handles this signaling
- GUI APIs (window manager) handle the interaction of widgets and provide the main poll loop (the loop that listens for or polls for events)
- Application Programmer simply creates widgets (using the library) and provides Event handlers (callbacks)
- Example Default Events: OnClick, OnUpdate, OnHover, etc.
- Registering Callbacks:
 - Create a button widget (an event source that has several pre-defined events)
 - **Event Binding**: Register a callback (a function) for the OnClick event
 - When the button is clicked the specified function(s) are executed (*signaled* or *notified*)

GTK Tutorial

Lecture Notes

1. Introduction

- Several C-based windowing systems (Qt, Motif), but inadequate
- Peter Mattis introduced GTK (1998) (GTK=GIMP ToolKit, GIMP=GNU Image Manipulation Program, GNU=GNU's Not Unix)
- Version 3.2.0 (Sept 2011)
- Written in pure C with optional C++ wrappers
- <http://Gtk.org>
- Full documentation: <http://developer.gnome.org/gtk3/stable/>
- A nice tutorial: http://luv.asn.au/overheads/gtk_intro/index.html
- Official Tutorial: <http://developer.gnome.org/gtk-tutorial/2.90/>

2. Examples

- Hello world
- Keypad

3. GtkWidget

- Pointer to a general widget that can be GtkWidget, GtkButton, GtkEntry (text boxes), etc.
- Many different “factory” methods to create specific types of widgets:
 - i. `gtk_window_new`
 - ii. `gtk_label_new`
 - iii. `gtk_entry_new`
 - iv. `gtk_button_new_with_label` (and alternatives)
 - v. `gtk_vbox_new`
 - vi. `gtk_hbox_new`
- Bringing widgets together: the parent-child relationship
 - i. `gtk_container_add`
 1. Wrap first argument in `GTK_CONTAINER()` (Hierarchy of widgets: `GTK_OBJECT > GtkWidget > GTK_CONTAINER > (GTK_BUTTON, GTK_WINDOW)`)
 2. Second argument: widget to add to the first
 - ii. Layouts
 1. Horizontal Box: child widgets displayed horizontally
 2. Vertical Box: child widgets displayed vertically
 3. `gtk_box_pack_start`, `gtk_box_pack_end` (adds widgets top to bottom; order matters)
 4. Arguments? Google It!
 5. Alternative: table (`gtk_table_new`)
 - iii. Visibility

1. `gtk_widget_show` – necessary to make the widget visible (all widgets invisible by default)
- Associating callbacks
 - i. `g_signal_connect(instance, signal, handler, data)`
 - ii. <http://developer.gnome.org/gobject/stable/gobject-Signals.html#g-signal-connect>
 1. instance (button)
 2. string indicating the event/signal
 3. function pointer to the handler
 4. optional data to be passed when handler is invoked
 - iii. Example: see: <http://developer.gnome.org/gtk/2.24/GtkButton.html>
 1. Different widgets support different signals, must refer to the documentation
 2. Button: events are segmented: can activate, enter, leave, pressed, released, or clicked!

Swing Tutorial

Lecture Notes

1. Introduction

- Java 1.0 supported GUI programming through the Abstract Window Tool (AWT)
- AWT was platform-dependent (implemented using native widgets)
- Swing was introduced to increase portability
- Swing Overview: <http://docs.oracle.com/javase/tutorial/ui/overview/index.html>
- Swing Features: <http://docs.oracle.com/javase/tutorial/ui/features/index.html>
- Swing Component Overview:
<http://docs.oracle.com/javase/tutorial/ui/features/components.html>
- Oracle Swing Tutorial: <http://docs.oracle.com/javase/tutorial/uiswing/>
- Resources

2. Components

- JComponent -> {JFrame, JPanel, JLabel, JButton, JCheckBox, JRadioButton, JTextComponent -> {JTextArea, JTextField} }
- Containers:
 - i. JFrame
 1. A main, heavy-weight component
 2. Usual to extend this class
 3. Launched inside a runnable (see <http://docs.oracle.com/javase/6/docs/api/javaw/swing/package-summary.html#threading>)
 - ii. JPanel
 1. Light-weight container
- JButton(String text)
 - i. addActionListener(ActionEvent)
 - ii. setActionCommand(String)
- JTextField(String text, [int columns])

3. Layouts

- Types:
 - i. FlowLayout – new components are added to the right, if they don't fit, then they flow to the “next line”
 - ii. GridLayout – new GridLayout(rows, cols, hGap, vGap); adds left-to-right, top-to-bottom (one or the other could be made zero to allow for any number of rows or cols)
 - iii. BorderLayout – can be vertical or horizontal, adds components in a left-to-right or top-to-bottom manner
- Setting: setLayout(lm)

4. Events

- Rather than call back functions, you can define an ActionListener
 - i. May be in a class (implements ActionListener) or
 - ii. May be an anonymous class
 - iii. Single method: actionPerformed(ActionEvent)
 - iv. More info:
<http://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html>
 - v.
- You can register ActionListeners with some components: button, menu item, text inputs, etc.
 - i. addActionListener
- Multiple components can have the same listener, you can communicate which component generated the event and the “Action Command” through the ActionEvent object
 - i. getSource()
 - ii. getActionCommand() (string)

5. Examples

- Hello world
- Keypad
- Keypad Version 2