Chapter 9

Graphical User Interfaces

At a Glance

Instructor’s Manual Table of Contents

Lecture Notes

Overview

Chapter 9 provides an introduction to graphical user interfaces. Students learn about the structure of a GUI-based program and using the MVC pattern. Through several examples, students learn to instantiate and lay out different types of window objects, including labels, entry fields, and command buttons, in a window's frame. Events and event-driven programming are also introduced, and students learn to define methods that handle events associated with window objects. Achieving different layouts using grids and nested frames (panes) is also covered.

Objectives

After completing this chapter, students will be able to:

- Structure a GUI-based program using the model/view/controller pattern
- Instantiate and lay out different types of window objects such as labels, entry fields, and command buttons
- Define methods that handle events associated with window objects
- Organize sets of window objects in nested frames

Teaching Tips

The Behavior of Terminal-Based Programs and GUI-Based Programs

1. Explain to students that this section presents two different versions of the `bouncy` program from a user's point of view: a terminal-based user interface and a *graphical user interface*.

The Terminal-Based Version

1. Use Figure 9.1 to describe how the user interacts with the terminal-based version of the program.

2. Explain the different effects of this interface on users, as listed in the bullet points on Page 351 of the text.

The GUI-Based Version

1. Use Figure 9.2 to describe how the user interacts with the GUI-based version of the program. Introduce the following terms: *window object*, *widget*, *label*, *entry field*, *command button*, and *title bar*.

2. Explain how this interface solves the user interaction problems introduced with the terminal-based version.

Event-Driven Programming

1. Describe the role of *event-driven programming* in GUI-based programs, and identify the steps involved in such programming.

2. Explain that GUI based programs adhere to the model/view pattern, which separates the resources and responsibilities for managing the data model from those concerned with displaying it and interacting with users.

3. Use Figure 9.3 to introduce the *model/view/controller* (*MVC*) pattern.

Teaching Tip	The PyjamasDesktop Python toolkit uses the PyWebkitGtk toolkit and provides the "V" in the MVC pattern. For more information, visit: http://wiki.python.org/moin/PyjamasDesktop.

4. Provide some insight on how to code a GUI-based event-driven program (see the steps listed on pages 354-355).

Quick Quiz 1

1. What is a GUI?
   Answer: Most modern computer software employs a graphical user interface or GUI. A GUI displays text as well as small images (called icons) that represent objects such as directories, files of different types, command buttons, and drop-down menus. In addition to entering text at the keyboard, the user of a GUI can select an icon with a pointing device, such as a mouse, and move that icon around on the display. Commands can be activated by pressing the Enter key or Control keys, by pressing a command button, or by selecting a drop-down menu item with the mouse. Put more simply, a GUI displays all information, including text, graphically to its users, and allows them to manipulate this information directly with a pointing device.

2. GUI-based programs display windows that contain various components, also called window objects or ____________________.
   Answer: widgets

3. True or False: In a GUI-based program, a status bar contains the title of the program.
   Answer: False

4. What are entry fields?
   Answer: Entry fields are boxes within which the program can output text or receive it as input from the user.

Coding Simple GUI-Based Programs

1. Briefly introduce the role of the `tkinter` and `tkinter.messagebox` standard Python modules.

Teaching Tip	Python has a huge number of GUI frameworks (or toolkits) available for it. You can find a list of several of these toolkits at: http://wiki.python.org/moin/GuiProgramming.

Teaching Tip	`tkinter` is based on the `Tk` widget set (a package to implement GUI programs under a windowing system). For more information on `tkinter`, visit: http://infohost.nmt.edu/tcc/help/lang/python/tkinter.html.

Windows and Labels

1. Use the `LabelDemo` example provided in the book and Figure 9.4 to explain how to create windows and labels using `tkinter`. Introduce the term *grid layout* and remind students of the meaning of the term *parent component*.

Displaying Images

1. Use the `ImageDemo` example provided in the book and Figure 9.5 to show how to create a label with an image using `tkinter`.

Command Buttons and Responding to Events

1. Use the `ButtonDemo` example provided in the book and Figure 9.6 to show how to create a button and how to enable it to respond to clicks.

Viewing the Images of Playing Cards

1. Use Figure 9.7 and the code provided in the book to show how to view the images of playing cards in Python. Stress that the `Card` and `Deck` classes created in Chapter 8 are reused for this purpose, and explain that the program uses existing image files to display the cards.

Entry Fields for the Input and Output of Text

1. Introduce the terms: *form filler* and *entry field*.

2. Use Table 9.1 to describe the three types of data container objects that can be used with `Entry` fields.

3. Use the `CircleArea` example and Figure 9.8 to show how to use `Entry` fields.

Using Pop-up Dialog Boxes

1. Use Table 9.2 to introduce the role of some of the most important `tkinter.messagebox` functions.

2. Use Figure 9.9 and the sample code provided in the book to show how to create an event handler that uses a pop-up dialog box that shows an error message.

Teaching Tip	For more information on the `tkinter.messagebox` module, visit: http://epydoc.sourceforge.net/stdlib/tkMessageBox-module.html.

Case Study: A GUI-Based ATM

1. Guide students as they step through this section.

Request

1. Ask students to replace the terminal-based interface of the ATM program with a GUI.

Analysis

1. Use Figure 9.10 to describe the GUI interface that will be implemented in this program.

Design

1. Point out that the `_login` and `_logout` methods of the ATM class are used for the even-driven programming.

2. Explain to students what needs to happen in each function. Refer to the bullet points on Pages 368-369.

Implementation (Coding)

1. Point out that no new classes are implemented. The `ATM` class is modified so that it now extends the `Frame` class.

2. Walk students through the code, explaining how it accomplishes the desired tasks.

Quick Quiz 2

1. What is `tkinter`?
   Answer: `tkinter` is a Python standard module that includes classes for windows and numerous types of window objects.

2. ____________________ is a Python standard module that includes classes for windows and numerous types of window objects.
   Answer: `tkinter.messagebox`

3. True or False: In programs that use several buttons, each button has its own event-handling method.
   Answer: True

4. A form filler consists of labeled ____________________ fields, which allow the user to enter and edit a single line of text.
   Answer: entry

Other Useful GUI Resources

1. Briefly list some of the other useful GUI resources that will be introduced in this section.

Colors

1. Explain how the RGB color system can be used with `tkinter` (both using the hex notation and the predefined string values).

2. Use one or more examples to explain that you can set two attributes for most GUI components: a foreground color (`fg`) and a background color (`bg`).

Text Attributes

1. Use Table 9.3 and one or more examples to explain how to modify a *type font*.

Sizing and Justifying an Entry

1. Use the sample code provided in the book and Figure 9.12 to show how to set the size and justification of entry fields.

Sizing the Main Window

1. Describe the role of the `geometry` and `resizable` methods that can be run with the root window to affect its sizing.

Grid Attributes

1. Stress that, by default, a newly opened window shrink-wraps around its components and is resizable.

2. Use Table 9.4 and the examples provided in the book, to describe the `Grid` attributes and how to use them.

3. Explain the term *expansion weight* and how an expansion weight can be assigned to a row or column of cells.

Teaching Tip	For more information on the `tkinter` grid geometry manager, visit: http://effbot.org/tkinterbook/grid.htm.

Teaching Tip	"TK provides three geometry managers: `.pack()`, `.grid()` and `.place()`." For more information, visit: www.ibm.com/developerworks/linux/library/l-tkprg/.

Using Nested Frames to Organize Components

1. Use the `ComplexLayout` example and Figure 9.16 to show how to use nested frames (*panes*) to organize GUI components.

Multi-Line Text Widgets

1. Use the `TextDemo` example and Figure 9.17 to show how to create multi-line `Text` widgets.

Scrolling List Boxes

1. Introduce the term *list box*, and use Table 9.5 to describe some of the most useful `Listbox` methods.

2. Using Figure 9.18 and the sample code provided in the book, show students how to use these methods to create scrolling list boxes.

Mouse Events

1. Use Table 9.6 to introduce the mouse events available in Python.

2. Use one or more examples to show how to associate a mouse event and an event-handling method with a widget by calling the `bind` method.

Keyboard Events

1. Use Table 9.7 to introduce some of the most useful key events available in Python.

2. Use one or more examples to show how to bind a keyboard event to a handler.

Teaching Tip	For more information on events and bindings in Python, read: www.pythonware.com/library/tkinter/introduction/events-and-bindings.htm.

Quick Quiz 3

1. True or False: In Python, a hex literal begins with the `%` symbol.
   Answer: False

2. How can the user resize a window?
   Answer: The user can resize a (resizable) window by dragging its lower-right corner in any direction.

3. True or False: By default, a newly opened window shrink-wraps around its components and is resizable.
   Answer: True

4. A nested frame is sometimes called a(n) ____________________.
   Answer: pane

Class Discussion Topics

1. Ask your students to share any previous event-driven programming experience they might have had with the class.

2. Which of the widgets introduced in this chapter do students believe they will use the most? And the least? Why?

Additional Projects

1. Ask students to do some research on other Python toolkits and modules for GUI and event-driven programming. They should compile a list (including a brief description) of 5-10 commonly used toolkits.

2. Write a GUI-based program that allows the user to play the Connect Four game (note: this game is also known as “Four in a row” and “Four in a line”). For an online version of the game, visit: www.christcenteredmall.com/kids/games/connect-four/.

Additional Resources

1. GUI Programming in Python:
   http://wiki.python.org/moin/GuiProgramming

2. Tkinter: GUI programming with Python:
   http://infohost.nmt.edu/tcc/help/lang/python/tkinter.html

3. Graphic User Interface FAQ:
   www.python.org/doc/faq/gui/

4. Module `tkMessageBox`:
   http://epydoc.sourceforge.net/stdlib/tkMessageBox-module.html

5. The Tkinter Grid Geometry Manager:
   http://effbot.org/tkinterbook/grid.htm

6. Events and Bindings:
   www.pythonware.com/library/tkinter/introduction/events-and-bindings.htm

Key Terms

- **button object:** A window object that allows the user to select an action by clicking a mouse.
- **event:** An occurrence, such as a button click or a mouse motion, that can be detected and processed by a program.
- **event-driven loop:** A process, usually hidden in the operating system, that waits for an event, notifies a program that an event has occurred, and returns to wait for more events.
- **grid layout:** A Python layout class that allows the user to place window objects in a two-dimensional grid in the window.
- **GUI (graphical user interface):** A means of communication between human beings and computers that uses a pointing device for input and a bitmapped screen for output. The bitmap displays images of windows and window objects such as buttons, text fields, and drop-down menus. The user interacts with the interface by using the mouse to directly manipulate the window objects. See also window object.
- **justification:** The process of aligning text to the left, the center, or the right within a given number of columns.
- **label object:** A window object that displays text, usually to describe the roles of other window objects.
- **model/view/controller pattern (MVC):** A design plan in which the roles and responsibilities of the system are cleanly divided among data management (model), user interface display (view), and user event-handling (controller) tasks.
- **window object (widget):** A computational object that displays an image, such as a button or a text field, in a window and supports interaction with the user.