Graphical User Interface (GUI)

Fawzi Emad  
Chau-Wen Tseng  

Department of Computer Science  
University of Maryland, College Park

Graphical User Interface (GUI)

- User interface  
  - Interface between user and computer  
  - Both input and output  
  - Affects usability of computer

- Interface improving with better hardware  
  - Switches & light bulbs  
  - Punch cards & teletype (typewriter)  
  - Keyboard & black/white monitor (text)  
  - Mouse & color monitor (graphics)
Graphical User Interface (GUI)

- Design issues
  - Ease of use
  - Ease of understanding
  - Ability to convey information
  - Maintainability
  - Efficiency

Graphic User Interface Overview

- GUI elements
- Java GUI classes
- Event-driven programming
- Model-View-Controller (MVC) Pattern
GUI Elements

- Component ⇒ items displayed (widgets)
- Container ⇒ region containing widgets
- Layout ⇒ arrangement of container
- Event ⇒ interactions for GUI

GUI Elements – Component

- Definition
  - Actual items (widgets) user sees in GUI

- Examples
  - Labels (fixed text)
  - Text areas (for entering text)
  - Buttons
  - Checkboxes
  - Tables
  - Menus
  - Toolbars
  - Etc...
GUI Elements – Container

Definition
- Abstractions occupying space in GUI

Properties
- Usually contain one or more widgets
- Can be nested in other containers

Example
- Window containing
  - 1 menu (component)
  - 3 buttons (component)
  - 2 windows (container)

GUI Elements – Layout

Definition
- Arrangement of widgets in container

Layout specification
- Logical terms (2nd row, 1st column, left)
  - Preferred approach
- Actual coordinates (100 pixels, 5 inches)
  - Can be too rigid, limited to certain window sizes

Layout manager
- Entity translating layout specifications into actual coordinates at runtime, depending on conditions
Example Java Layout Managers

- **GridLayout**
  - Lays out components in a grid of user specified size

- **BorderLayout**
  - Designates portions of the container as North, South, East, West, and Center

- **CardLayout**
  - Adds components one on top of another

- **GridBagLayout**
  - Customizable manager that can use rows and columns of varying lengths

GUI Elements – Events

- **Definition**
  - Action or condition occurring outside normal flow of control of program

- **Examples**
  - Mouse clicks
  - Keyboard input
  - Menu selections
  - Window actions
Event-driven Programming

- Normal (control flow-based) programming
  - Approach
    - Start at main()
    - Continue until end of program or exit()

- Event-driven programming
  - Unable to predict time & occurrence of event
  - Approach
    - Start with main()
    - Build GUI
    - Await events (& perform associated computation)

Event-driven Programming in Java

- During implementation
  - Implement event listeners for each event
  - Usually one event listener class per widget

- At run time
  - Register listener object with widget object
  - Java generates event object when events occur
  - Java then passes event object to event listener

- Example of observer design pattern
Event-driven Programming in Java

- Example listeners & actions causing event
  - ActionEvent ⇒ clicking button in GUI
  - CaretEvent ⇒ selecting portion of text in GUI
  - FocusEvent ⇒ component gains / loses focus
  - KeyEvent ⇒ pressing key
  - ItemEvent ⇒ selecting item from pull-down menu
  - MouseEvent ⇒ dragging mouse over widget
  - TextEvent ⇒ changing text within a field
  - WindowEvent ⇒ closing a window

Java GUI Classes

- AWT (Abstract Window Toolkit) (java.awt.*)
  - Old GUI framework for Java (Java 1.1)
  - Some reliance on native code counterparts
  - Platform independence problems

- Swing (javax.swing.*)
  - New GUI framework first introduced in Java 1.2
  - Includes AWT features plus many enhancements
  - Pure Java components (no reliance on native code)
  - Pluggable look and feel architecture
Java GUI Components

Examples
- JFrame
- JTextField
- JLabel
- JButton
- JList
- JComboBox
- Menu
- Combo
- Panes
- Indicators
- Dialog boxes
- JFileChooser
- Color chooser
- JTable
- JTree

JFrame
**JTextField and JLabel**

![Image of JTextField and JLabel example]

**Buttons**

![Image of Button example]

**Toolbar**

![Image of Toolbar example]
JList and JComboBox

Menu

A Menu | Another Menu
---|---
A text-only menu item | Alt-1
Both text and icon
A radio button menu item
A check box menu item
A submenu
Panels and Panes

Tabbed Pane

Split Pane

Scroll Pane

Various Indicators

Spinner

Slider

Progress Bar
A Dialog Box

![An Inane Question]

Would you like green eggs and ham?

Yes  No

JFileChooser

![Open]

Look in: 

emacsdb  host-news  java  main
Color Chooser

JTable and JTree

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Favorite Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>Dinkins</td>
<td></td>
</tr>
<tr>
<td>Ewan</td>
<td>Dinkins</td>
<td></td>
</tr>
<tr>
<td>Amy</td>
<td>Fowler</td>
<td></td>
</tr>
<tr>
<td>Hania</td>
<td>Gajewska</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td>Geary</td>
<td></td>
</tr>
</tbody>
</table>

[]

- Music
- Classical
  - Beethoven
  - Brahms
  - Mozart
- Jazz
- Rock
Simple Java GUI Example

```java
import javax.swing.*;

public class HelloWorldApplication
{
    public static void main(String[ ] args) {
        JFrame myFrame = new ThreadExample(“Hello”);
        myFrame.setSize(300, 150);
        myFrame.setVisible(true);
    }
}
```
**JFrame Hierarchy**

- Several super classes and well as implemented interfaces
- Many, many member methods including inherited methods that allow for operations such as resizing, setting properties, adding components, etc.
- Other top level containers
  - JDialog (dialog boxes)
  - JApplet (web applets)
  - JWindow (stripped down JFrame, no title bar or window buttons)

**JFrame Structure**

- Most things go into content pane
  - getContentPane()
- Use glassPane for pop up menus, some animations
- Methods
  - getRootPane()
  - getLayeredPane()
  - getContentPane()
  - getGlassPane()
- Can set…Pane explicitly

LayeredPane manages (optional) JMenuBar
LayeredPane contains contentPane
Model-View-Controller (MVC) Pattern

- Developed at Xerox PARC in 1978
- Separates GUI into 3 components
  - Model ⇒ application data
  - View ⇒ visual interface
  - Controller ⇒ user interaction

MVC Interaction Order

1. User performs action, controller is notified
2. Controller may request changes to model
3. Controller may tell view to update
4. Model may notify view if it has been modified
5. View may need to query model for current data
6. View updates display for user
MVC Pattern – Advantages

- Separates data from its appearance
  - More robust
  - Easier to maintain
- Provides control over interface
- Easy to support multiple displays for same data

MVC Pattern – Model

- Contains application & its data
- Provide methods to access & update data
- Interface defines allowed interactions
- Fixed interface enable both model & GUIs to be easily pulled out and replaced

Examples
- Text documents
- Spreadsheets
- Web browser
- Video games
**MVC Pattern – Controller**

- Users interact with the controller
- Interprets mouse movement, keystrokes, etc.
- Communicates those activities to the model
- Interaction with model indirectly causes view(s) to update

**MVC Pattern – View**

- Provides visual representation of model
- Multiple views can display model at same time
  - Example: data represented as table and graph
- When model is updated, all its views are informed & given chance to update themselves
Principles of GUI Design

- **Model**
  - Should perform actual work
  - Should be independent of the GUI
  - But can provide access methods

- **Controller**
  - Lets user control what work the program is doing
  - Design of controller depends on model

- **View**
  - Lets user see what the program is doing
  - Should not display what controller thinks is happening (base display on model, not controller)

---

Principles of GUI Design

- **Combining controller & view**
  - Appropriate if very interdependent
  - Especially in small programs

- **Separation of concerns**
  - Never mix model code with GUI code
  - View should represent model as it really is
    - Not some remembered status
  - Controller should talk to model and view
    - Avoid manipulate them directly