

CMSC 132: Object-Oriented Programming II



Graphic User Interface (GUI)

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

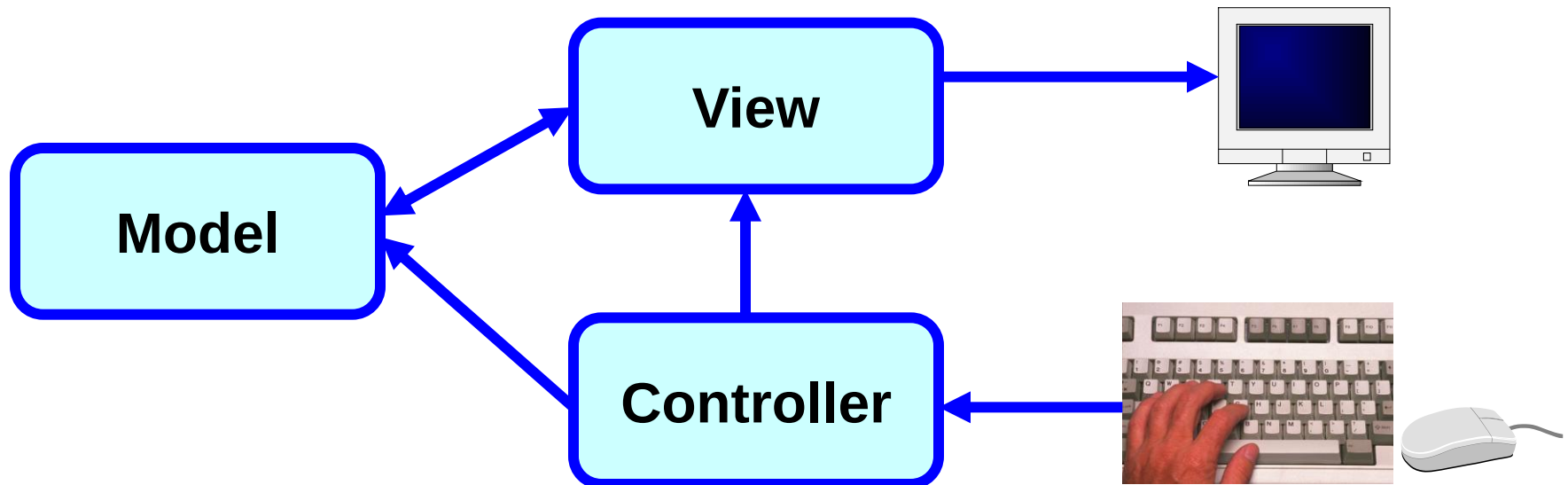
- Switches & light bulbs
- Punch cards & teletype (typewriter)
- Keyboard & black/white monitor (text)
- Mouse & color monitor (graphics)

GUI Topics

- **Model-View-Controller model**
- **Java support for GUIs**
 - **Containers**
 - **Components**
 - **Events**
- **Event-driven programming**

Model-View-Controller (MVC)

- **Model for GUI programming (Xerox PARC '78)**
- **Separates GUI into 3 components**
 1. **Model** ⇒ application data
 2. **View** ⇒ visual interface
 3. **Controller** ⇒ user interaction



MVC Model 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)

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
- **SWT (Standard Widget Toolkit; from Eclipse)**

Creating a GUI in Java

1. Define a **container** to hold components
 - Examples: JFrame, JPanel, JApplet...
2. Add GUI **components** to the container
 - Examples: JButton, JTextField, JScrollBar...
 - Use layout manager to determine positions
3. Add actions to GUI
 - Add event listeners to GUI components

GUI Elements 1 – Container

■ Definition

- Abstractions occupying space in GUI

■ Properties

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

■ Examples

- JFrame, JDialog, JPanel, JScrollPane

GUI Elements 2 – 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 3 – Events

■ Definition

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

■ Examples

- Mouse clicks
- Keyboard input
- Menu selections
- Window actions

Programming Models

■ 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

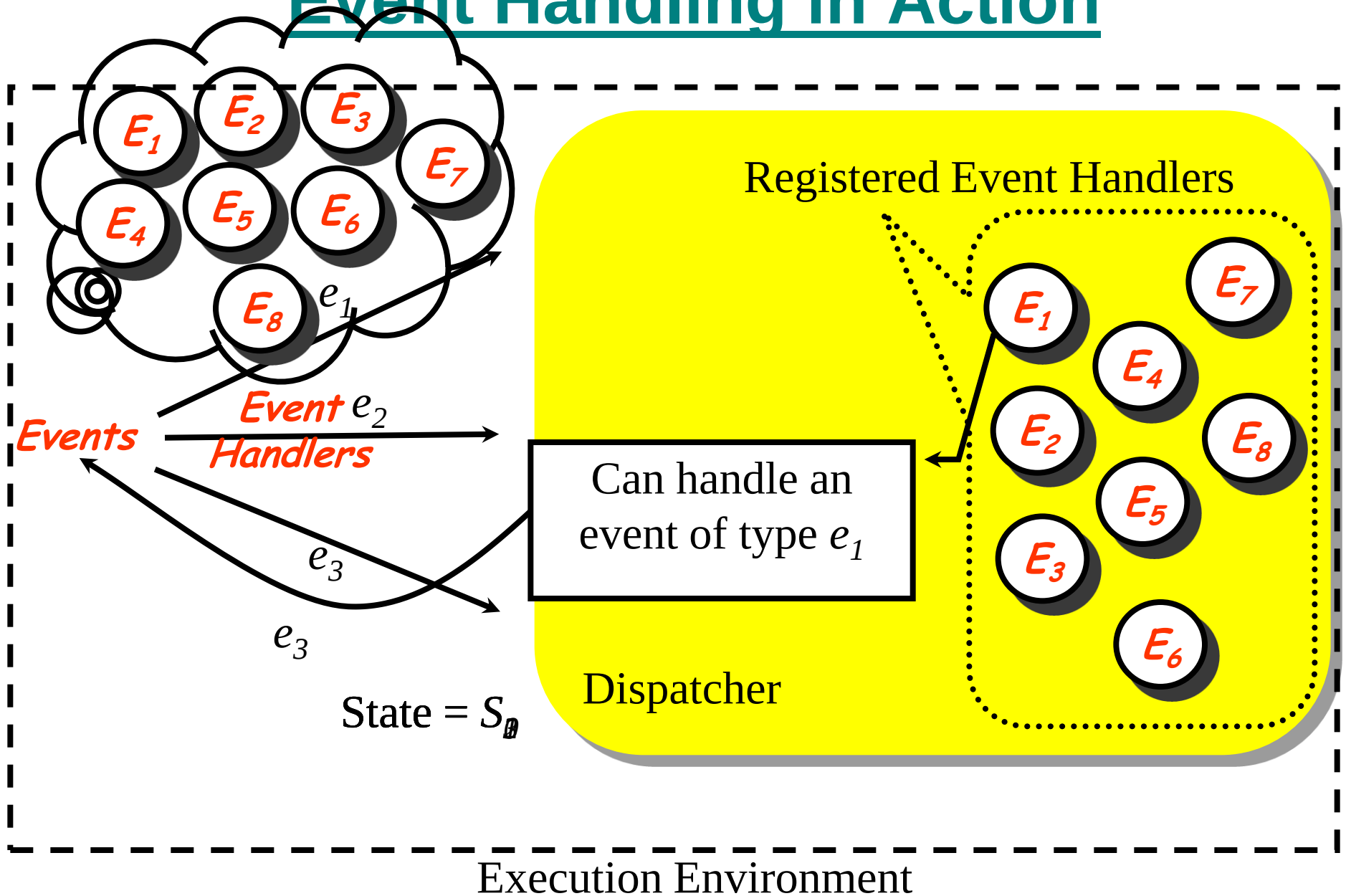
■ During implementation

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

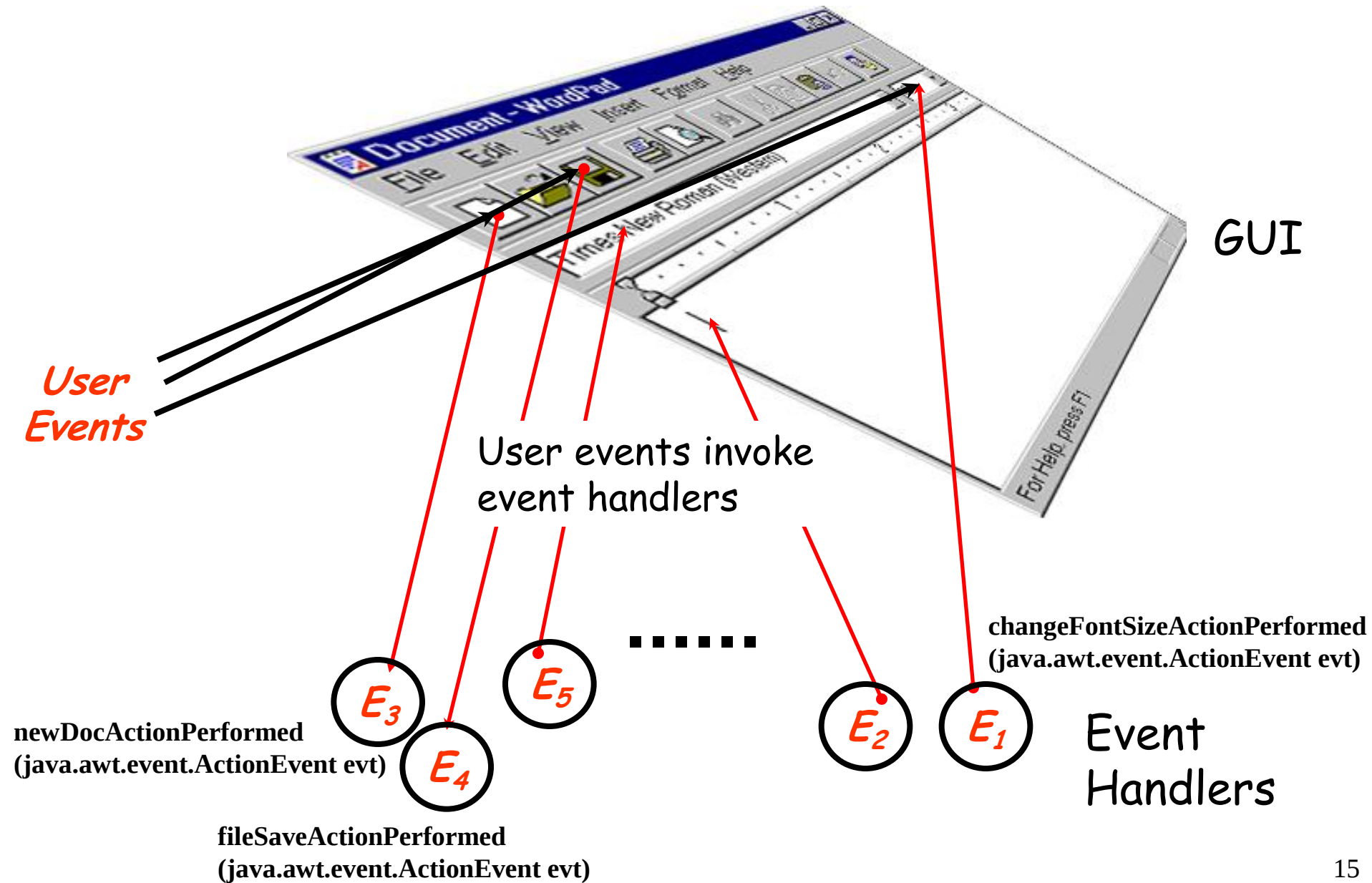
■ At run time

- Register listener object with widget object
- Java generates **event object** when events occur
- Java then passes event object to event listener

Event Handling in Action



GUIs are Event-Driven Software



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

■ In Java

- GUI events handled in **event dispatching** thread

Event Dispatching Thread

- Background thread to process events
 - From AWT graphical interface event queue
- These events are mainly **updates** that
 - Cause components to redraw themselves
 - Represent input events
- Swing uses a single-threaded painting model
 - Event Dispatching thread is the only valid thread for updating GUI components
 - Avoid updating GUI components from other threads
 - A source of common bugs

Event Dispatching Thread

■ Example code

- Allows current thread to execute GUI code in dispatching thread
- `createAndDisplayGUI`
 - Method that actually defines the GUI

```
javax.swing.SwingUtilities.invokeLater(new Runnable( ) {  
    public void run( ) {  
        createAndDisplayGUI( );  
    }  
});
```

Java Support For GUIs

■ Additional Resources

- Appendix C of textbook
- Javadoc for the JDK
- Swing tutorial
- Course slides and code handouts
- <http://filthyrichclients.org/>
- Java Ranch