Event Dispatch

Interactor Tree

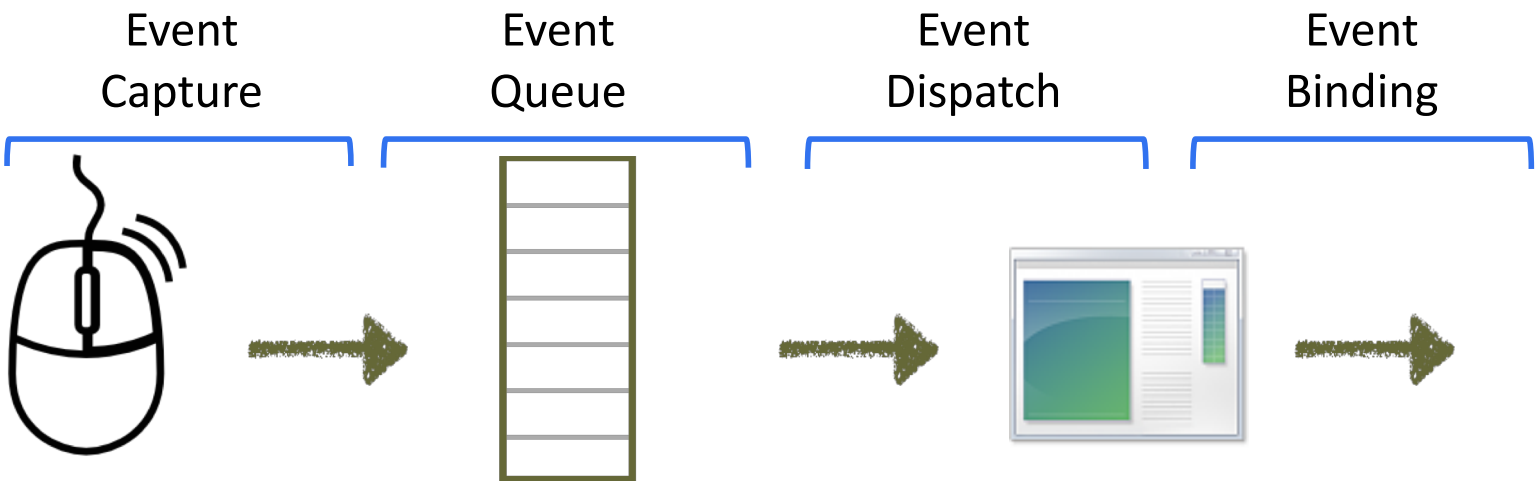
Lightweight vs. Heavyweight

Positional Dispatch

Focus Dispatch

Event Architecture

- A pipeline:
  - **Capture** and **Queue** low-level hardware events
  - **Dispatch** events to correct window and widget
  - **Bind** event with application code



Event Dispatch

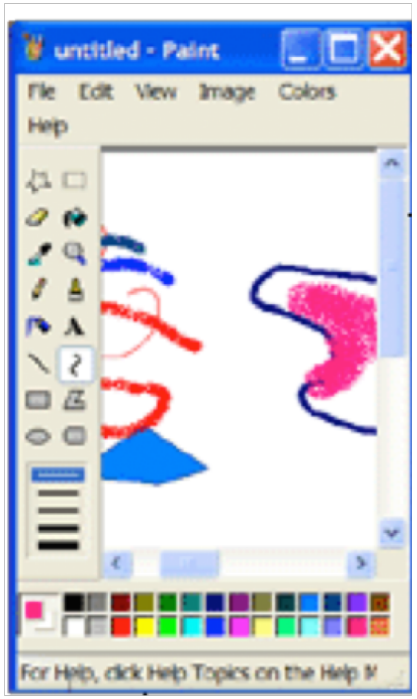
Event Loop servers to handle the initial dispatch to the window.

- Iterates through the event queue, and dispatches events
  - Serves as a low-level mechanism for event dispatch
- Dispatches to the application window, which
  - triggered the event, or
  - is in the foreground/accepting events

We want the event to eventually get dispatched to the **correct widget to handle that event**.

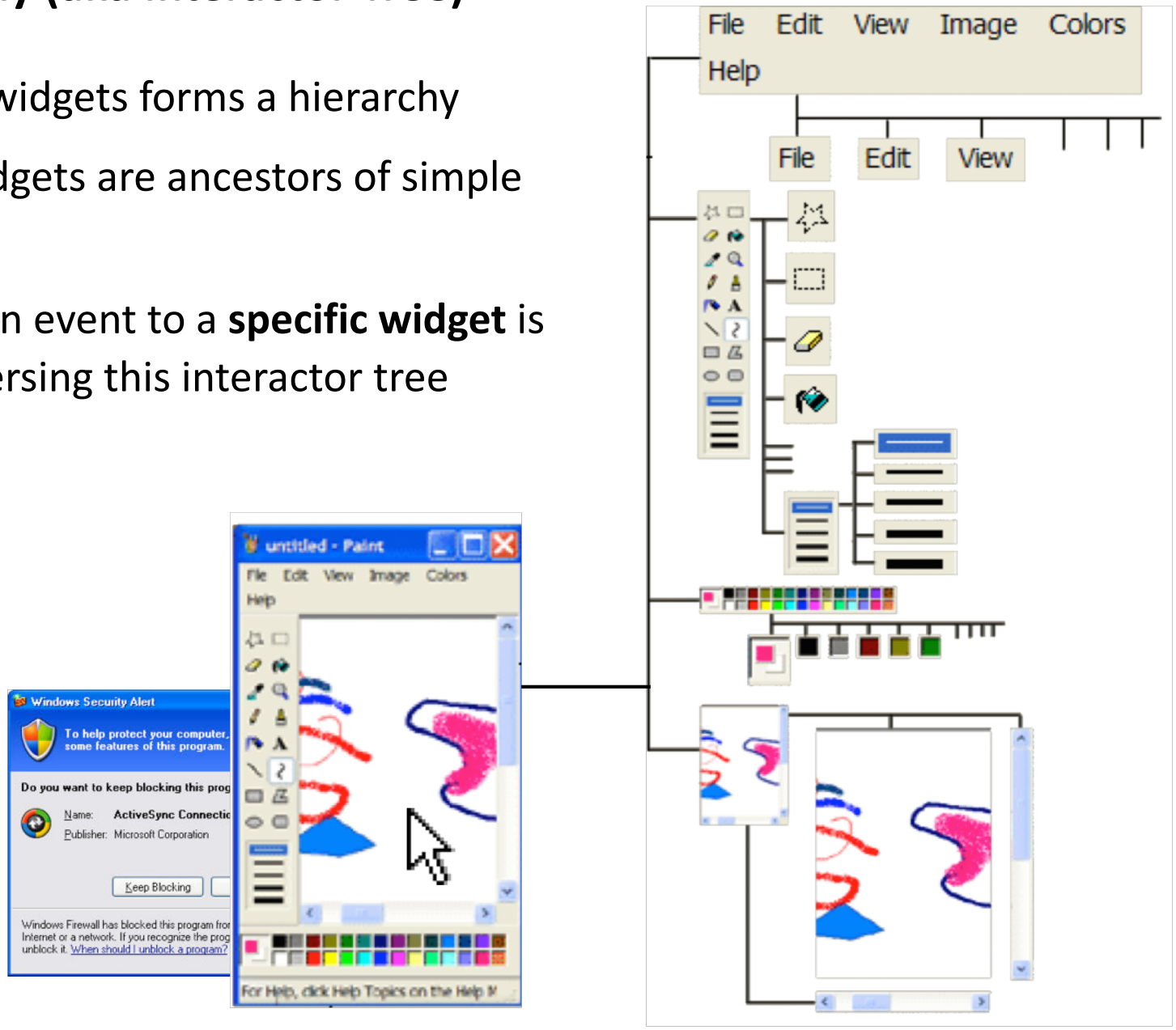
This is often the widget that generated it as well.

- e.g. if you click a button on a form, we want that button to receive and process the click event.



View Hierarchy (aka Interactor Tree)

- 2D layout of widgets forms a hierarchy
- Container widgets are ancestors of simple widgets
- Dispatching an event to a **specific widget** is done by traversing this interactor tree

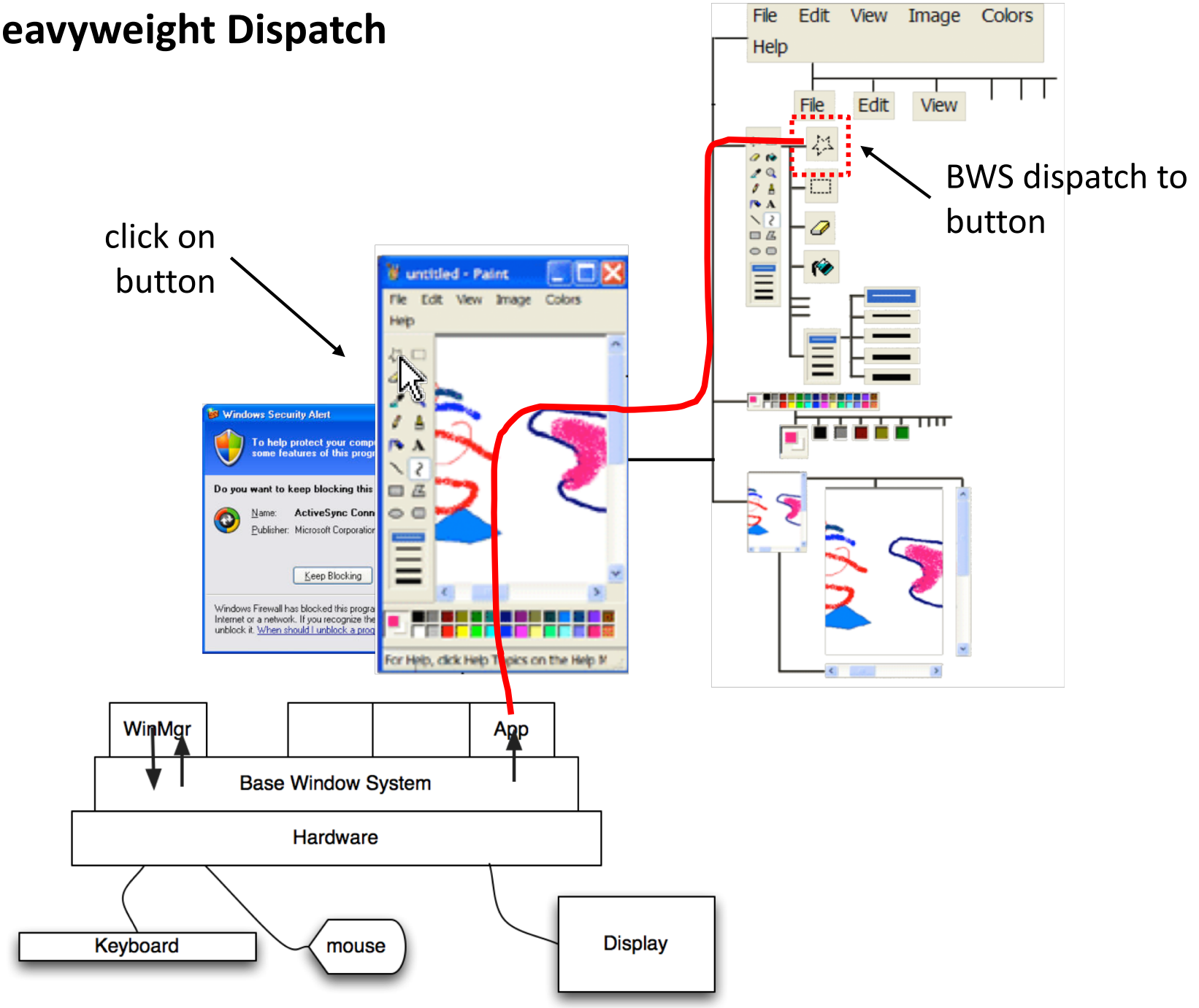


Heavyweight vs. Lightweight Widgets

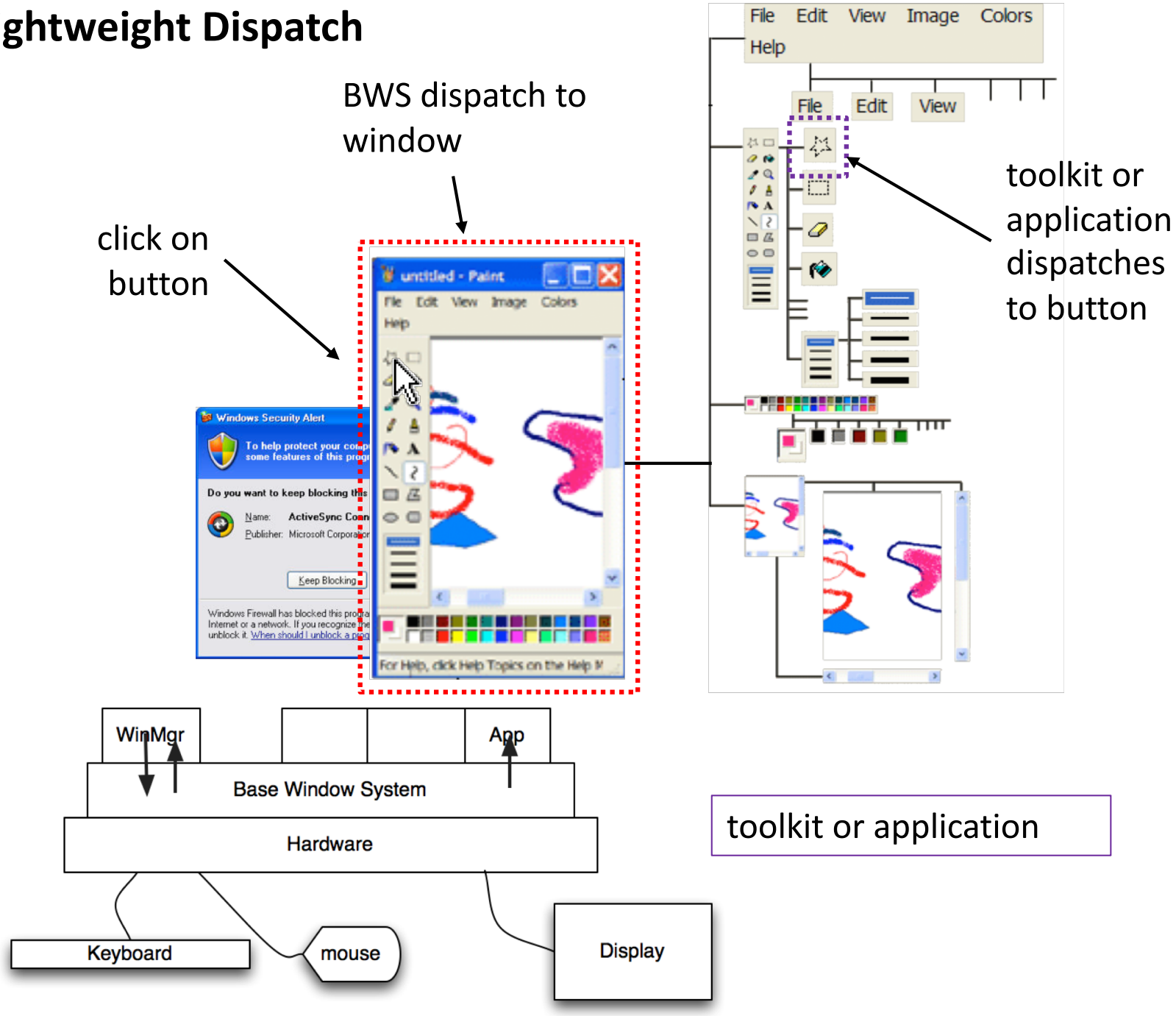
- Heavyweight widgets
  - Widget toolkit that wraps native-OS widgets
  - WS/OS provides a hierarchical “windowing” system for all widgets across all applications, and treats a widget essentially as a window
  - BWS can dispatch events to a specific widget
  - Retains platform look-and-feel by leveraging native widgets!
  - e.g. nested X Windows, Java’s AWT, HTML forms, Windows MFC

- Lightweight widgets
  - The widget toolkit that draws its own widgets and is responsible for mapping incoming events to widgets
  - BWS/OS dispatches to the window (NOT the widget)
  - Common implementation so we can guarantee consistency, completeness
  - e.g. Java Swing, JQuery UI, Windows WPF

Heavyweight Dispatch

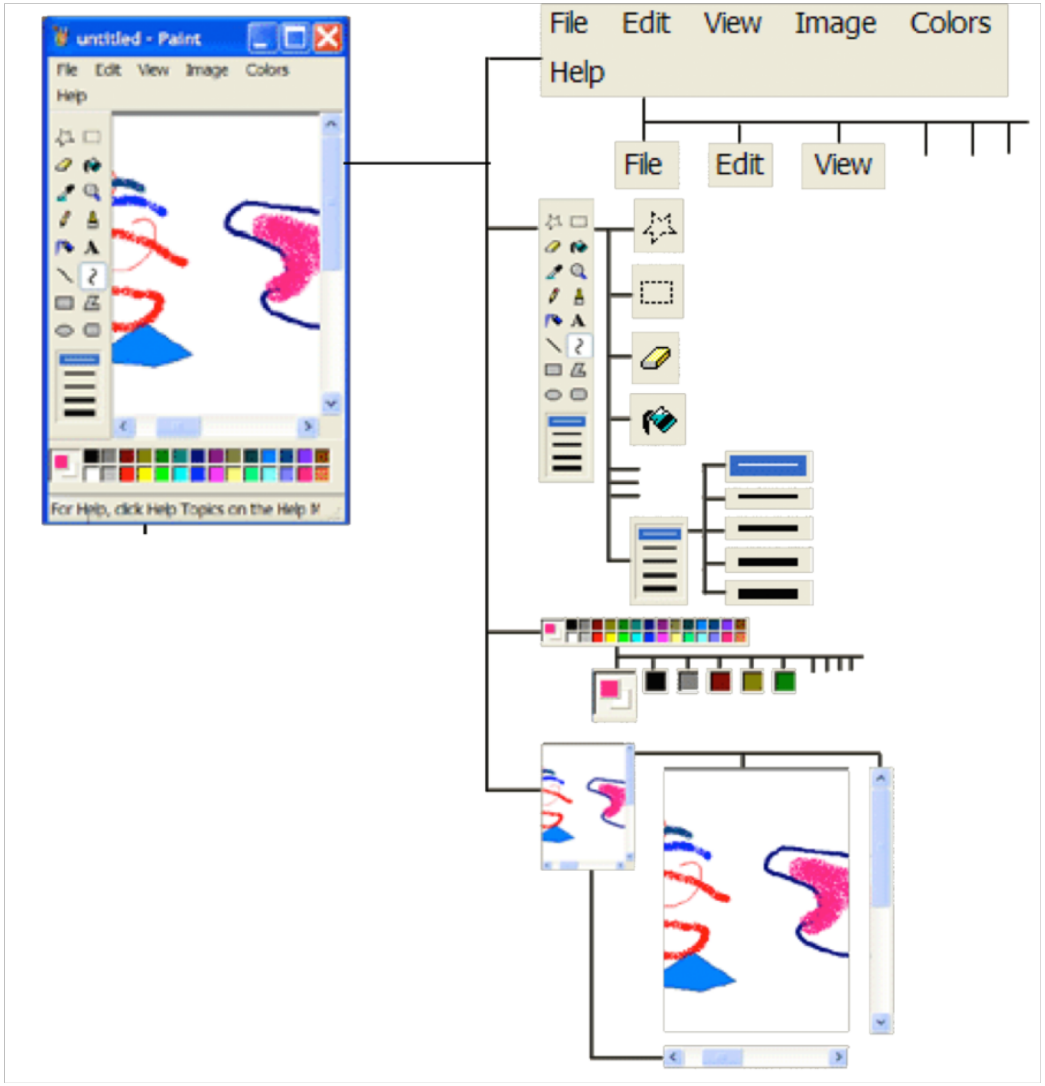


Lightweight Dispatch



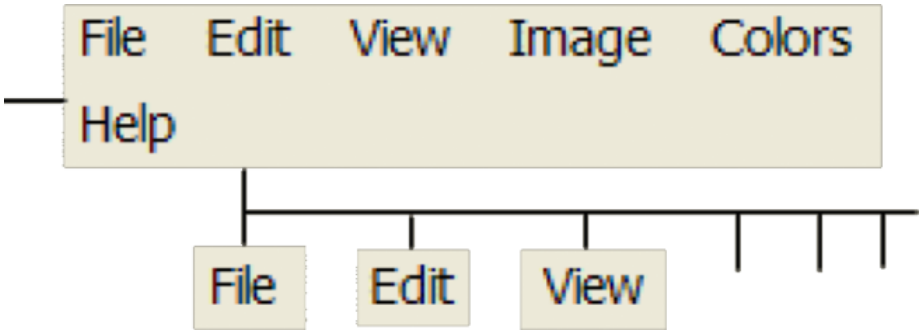
Positional Dispatch

- *Strategy:* send input to widget "under" mouse cursor
- *Issue:* many widgets overlap, so which widget receives the event first?
- Two methods:
  - Bottom-up
  - Top-down



Bottom-up Positional Dispatch

- Lowest-level node has the first chance to handle the event
- Event is dispatched to **leaf node widget in the UI tree** that contains the mouse cursor
- The leaf node widget can either:
  1. handle the event itself
  2. pass the event to its parent
     (who can process it or send to its parent...)



Passing to Parent

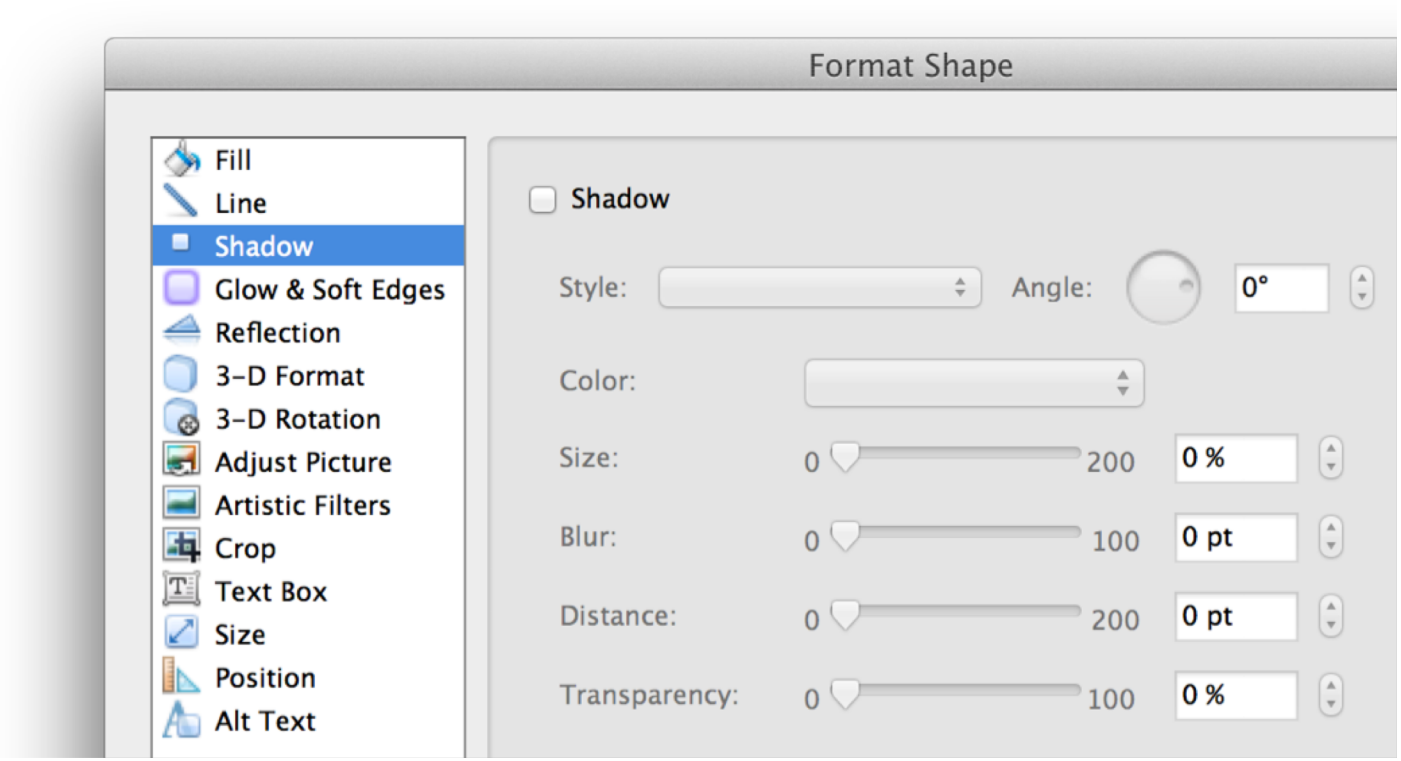
- Why would a widget pass an event to its parent?
  - Example: A palette of colour swatches may implement the colours as buttons. But palette needs to track the currently selected colour. Easiest if the palette deals with the events.

Top-down Positional Dispatch

- Highest-level node has the first chance to handle the event
- Event is dispatched to widget in the **highest level node** in the UI tree that contains the mouse cursor.
- The top node widget can either:
  1. handle the event itself
  2. pass the event to the child “under” the mouse (who can process it or send to its child...)

Top-down vs. Bottom-up Dispatch

- When do these behave the same way?
- Advantages of top-down?



disabled container widget policy

Positional Dispatch Limitations

- Positional dispatch can lead to odd behaviour:
  - Mouse drag starts in a scrollbar, but then moves outside the scrollbar: send the events to the adjacent widget?
  - Mouse press event in one button widget but release is in another: each button gets one of the events?
- Sometimes position isn't enough, also need to consider which widget is "in focus"

Focus Dispatch

- Events dispatched to widget regardless of mouse cursor position
- Needed for all keyboard and some mouse events:
  - Keyboard focus: Click on text field, move cursor off, start typing
  - Mouse focus: Mouse down on button, move off, mouse up … also called "mouse capture"
- Maximum one keyboard focus and one mouse focus
  - why?
- Need to gain and lose focus at appropriate times
  - Transfer focus on mouse down ("capture")
  - Transfer focus when TAB key is pressed

Focus Dispatch *Needs* Positional Dispatch

- But if a widget has focus, it should not receive every event:
  - mouse down on another suitable widget should change focus
- Often helpful to have an explicit focus manager in a container widget to manage which widget has the focus.

Accelerator Key Dispatch

- Keyboard events dispatched based on **which keys** are pressed
- Register special keyboard accelerators with specific commands
  - commands are often the target of menu item events
- The GUI toolkit intercepts accelerators and forwards to the appropriate command handler

Dispatch Summary

- Dispatch depends on what manages application-level events:
  - BWS could dispatch to widget level (heavyweight)
  - application could manage dispatch (lightweight)
  - toolkit could dispatch to widgets (lightweight, JVM/Swing)
- Positional + Focus dispatch are required for most events
  - Mouse-down events are almost always positional:
    dispatched to widget under cursor (top-down or bottom-up)
  - Other mouse and keyboard events go to widget in focus
- Non-input events may go elsewhere
  - Paint/damage events not necessarily associated with widget which receives input events
    (e.g. slider events ultimately effect other widgets to be repainted)