

CS 160: UI Implementation

Professor John Canny

3/6/2006

1

Outline

Output

- * Basic 2-D computer graphics
- * Color models

Input

- * Event overview
- * Windowing systems
- * Window events
- * Event dispatching

Development platforms

2-D Computer Graphics

- Models for images
 - * Strokes, pixels, regions
- Coordinate systems
 - * Device, physical
- Canvas
- Drawing
 - * Paths, shapes, text

Stroke Model



Describe image as strokes (w/ color/thickness)

+ Line ((10, 4), (17,4), thick 2, red)

+ Circle ((19, 13), radius 3, thick 3, white)

Maps to early vector displays & plotters

Most UI toolkits have stroked objects

* arcs, ellipses, rounded rectangles, etc.

Problems with Stroke Model?



- How would you represent with strokes?
- Solution?

Pixel Model


- ☞ Break-up complex images into discrete “pixels” & store color for each
- ☞ Resolution
 - * Spatial: number of rows by columns
 - * e.g., 1280 x 1024 is a good monitor display
 - * Quality laser printer: 10200 x 13200 (1200 dpi)
 - * Image depth (i.e., number of bits per pixel)
 - * Several styles... 8-bit, 24-bit, 32-bit


Image Depth



- 📄 Bit map - 1 bit/pixel (on/off)
 - * B&W screens or print-outs

Image Depth (cont.)

 Gray scale - 2-8 bits/pixel

 Full color - 24 bits/pixel

* 8 bits per primary color (Red, Green, Blue)

Image Depth (cont.)

Full color - 32 bits/pixel

- * Usually just 24-bit color (used for efficiency)
- * Extra 8-bits are optional - can be used for "alpha" (transparency)

Color mapped - 8 bits/pixel

- * Store index @ pixel - map into table w/ 24 bits
- * Cuts space & computation
- * Problem????

Image Depth (cont.)

 Jpeg image of blue sky

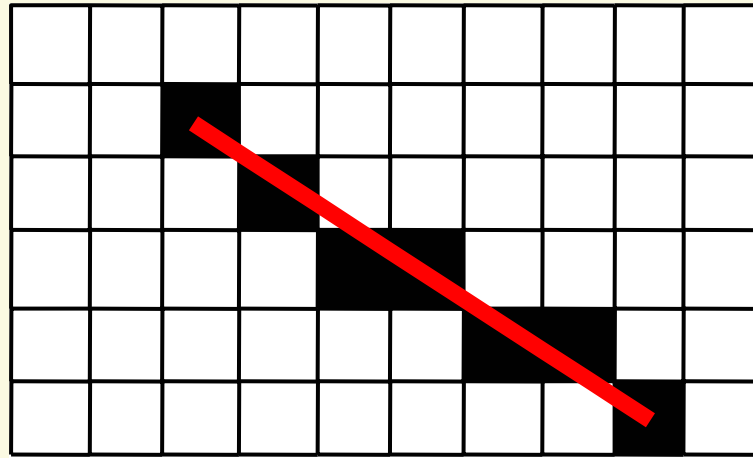


Image Depth (cont.)

 Blue sky with limited image depth

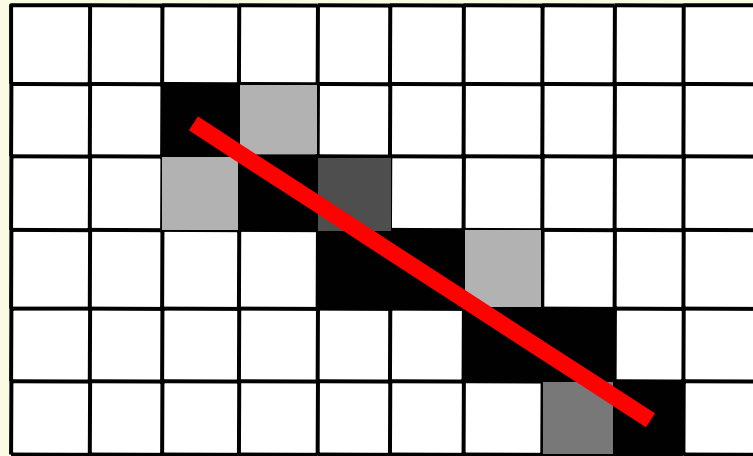


Aliasing



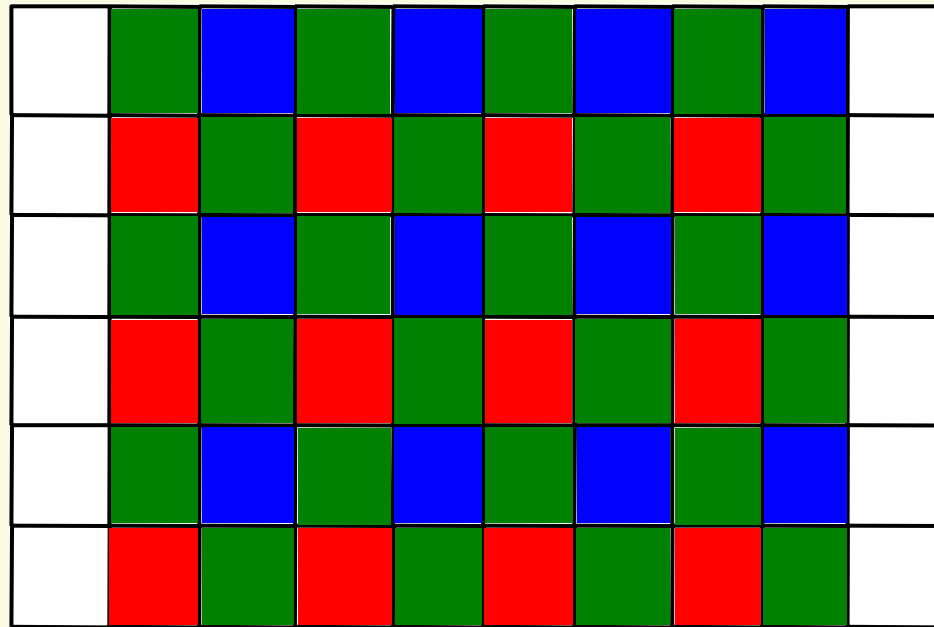
- Smooth objects (e.g., lines) appear jagged since resolution is too low
- Antialiasing - fill-in some jagged places w/ gray scale or primary colors


Anti-Aliasing



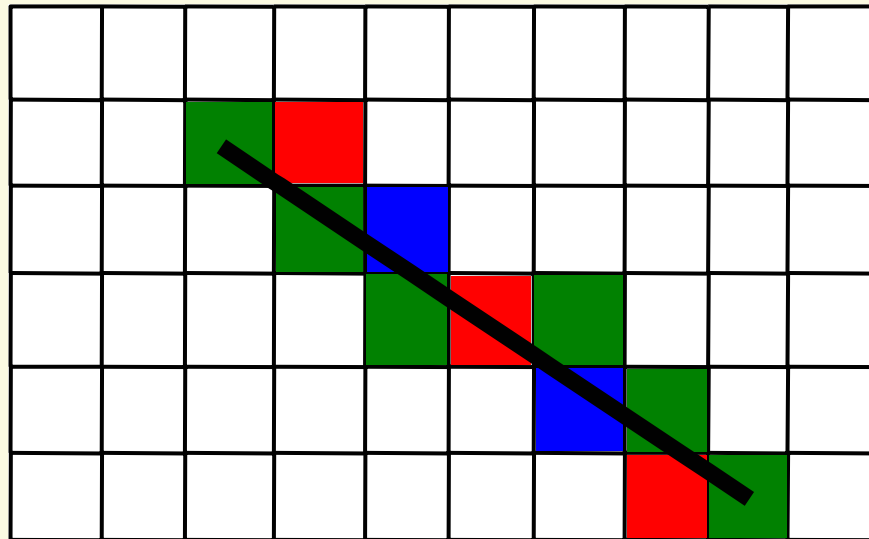
- ❏ Pixels colored in proportion to relative amount of line that crosses them.
- ❏ Equivalently, draw the line in B/W at finer resolution and then color each pixel in proportion to number of colored sub-pixels.

Cleartype



 The pixel matrix for a laptop or LCD screen.

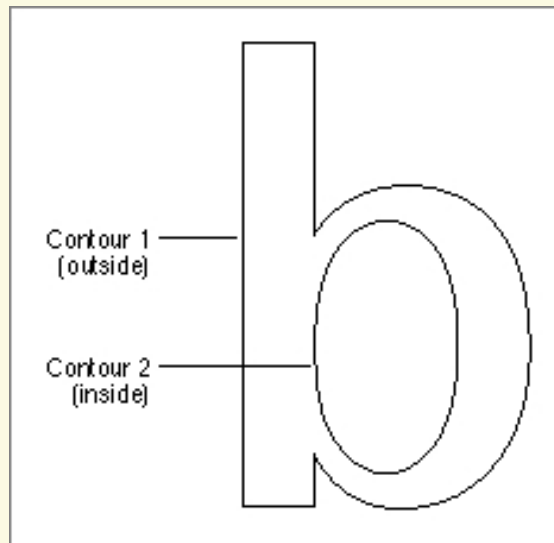
Cleartype



- Use sub-pixel color pixels as though they were gray pixels (can cause color anomalies).

Outline Fonts

Used by both Postscript & TrueType



Boundary is represented with splines, and can be scaled to any size.

Canvas

- 📄 Abstraction for the drawing surface
 - * Most toolkits support one
- 📄 Defines methods used for drawing
- 📄 Each instance has a height, width, & defines its physical units
- 📄 Use the same method interface for
 - * Windows
 - * Image in memory
 - * Printed output
- 📄 Called Graphical Device Interface (GDI) by MS

Graphics Context

Could specify with:

- * `void Canvas::Rectangle (x1, y1, x2, y2, lineWidth, lineColor, fillColor)`

Lots of parameters!

- * shapes have properties in common
 - + geometry, line/border width, line/fill color, pattern

Use current settings of canvas

- * Usually there is a "graphicscontext" or similar abstraction that defines all the parameters needed for drawing.

Text Font Selection

Font family

- * Garamond, Arial, Modern, Times Roman, Courier
- * defines the general shape of the characters
 - + Some are mono-spaced ("i" gets same space as "G")
 - + Serif (e.g., Times) vs. sans serif (e.g., Arial)
 - + Serifs have "feet" at baseline -> easier to track eye but look bad on low-resolution displays.

Style

- * normal, bold, *italic*, ***bold italic***

size in points (1 point = 1/72 inch)

Text (cont.)

Usually simple to draw

```
+ Canvas Cnv;  
+ Cnv.SetFont ("Times", Bold, 10);  
+ Cnv.Text (10, 20, "This is the text");
```

Outline vs. Bitmapped fonts

- * Precomputed bitmap fonts faster to draw
- * But separate maps needed for each font size
- * Outlines are fixed size, and can be scaled

Vector vs. Raster Image Formats

Vector:

- * Macromedia/Adobe Flash.
- * SVG (Scalable Vector Graphics), a W3C standard.
- * VML (Microsoft), Powerpoint animation.
- * XAML - the basis for Windows Vista

Raster/Bitmap:

- * Jpeg: Better for smooth images
- * Gif, PNG: Better for line art or "South Park" characters

Color Models

☞ 256 levels for each primary color

* -> 24 bits / pixel

☞ RGB model

* Specify color by **red**, **green**, & **blue** components

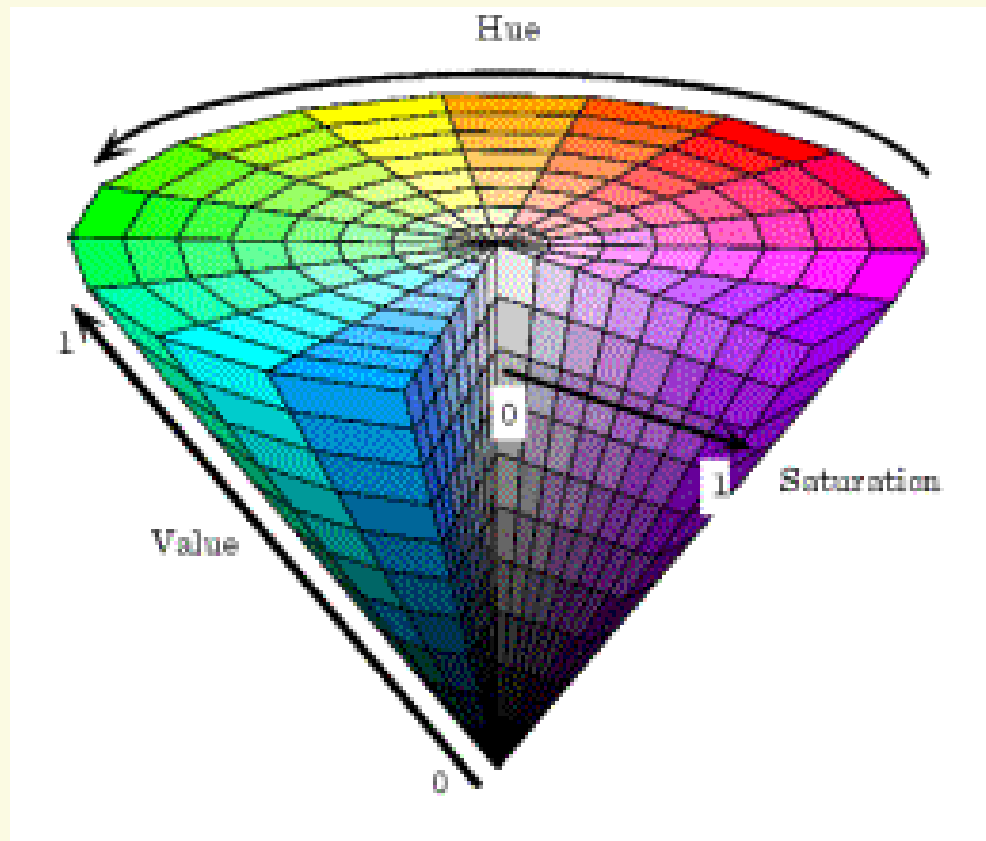
☞ HSV model - hue, saturation, & value

* Hue is primary wavelength (i.e., basic color)

* Saturation is a measure of how pure color is

* Value is intensity (dark vs. light)

HSV



Color Models (cont.)

📄 HSV is easier for people to use

- * There is a direct conversion to RGB

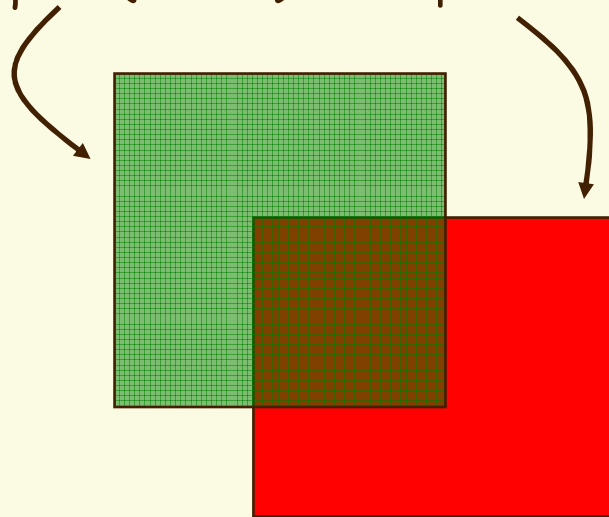
📄 CMY model

- * In terms of mixtures of pigments
- * Pigment gets color from light it absorbs and does not reflect
- * Mix Cyan, Magenta, Yellow
 - + subtractive primaries
- * Used by printers and artists

Alpha Channel

Images sometimes have a 4th channel called "alpha" (α) to encode transparency (e.g. png)

$C = \alpha \times C_f + (1-\alpha) \times C_r$ - each color channel



Break

Command-line Interaction

- ☞ Program takes control, prompts for input
- ☞ Examples include
 - * Command-line prompts (DOS, UNIX)
 - * SCHEME interpreter
- ☞ The user waits on the program
 - * Program tells user it's ready for more input
 - * User enters more input
- ☞ But what do you do for a graphical interface with many widgets?

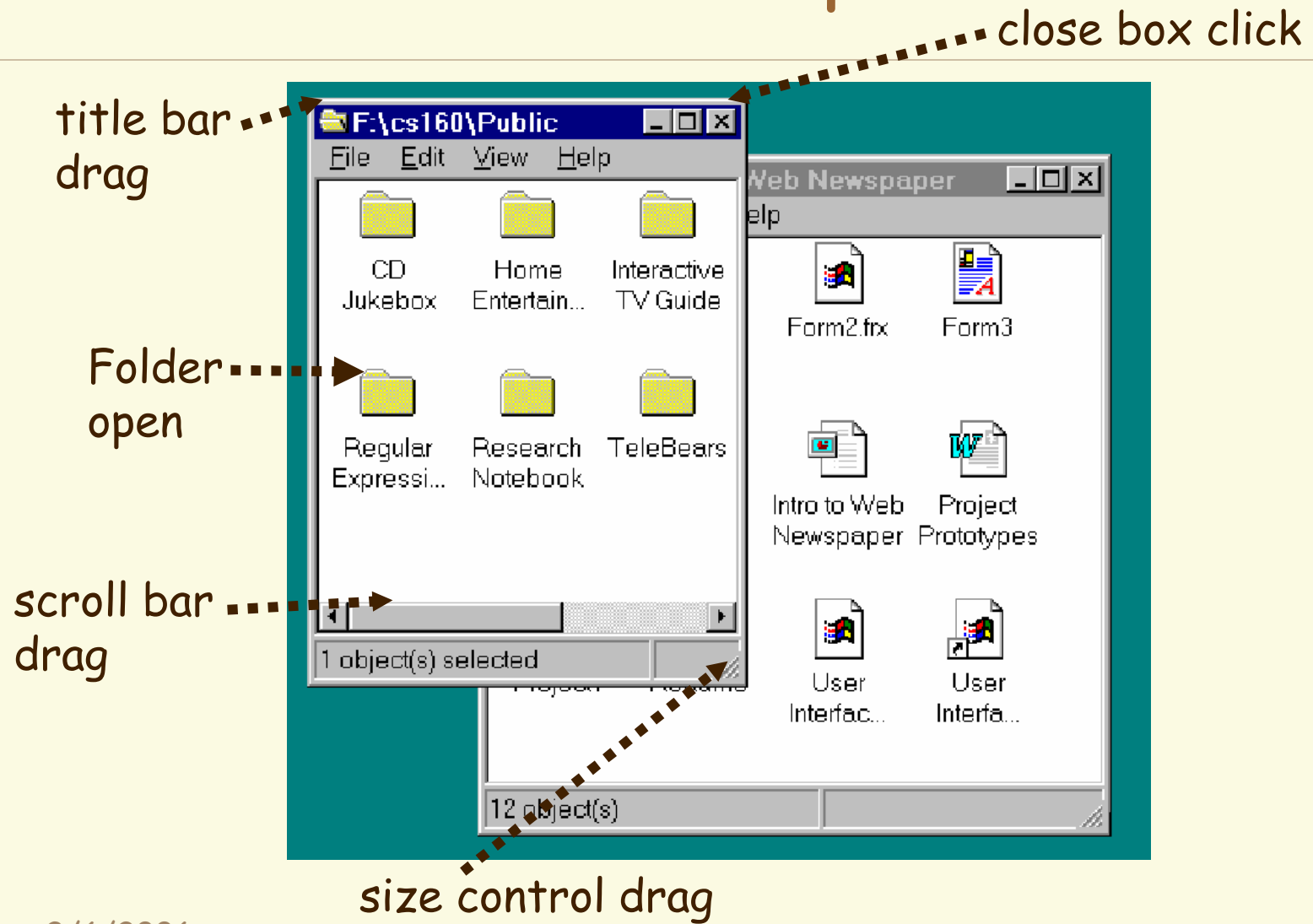
Modal Input

- ☞ You can try to limit what the user can do:
- ☞ Usually end up with lots of *modes*
 - * Only one dialog is active in the current mode
- ☞ Other examples of modes
 - * Paint programs (one tool is active)
 - * Universal remotes with TV / VCR / DVD mode
- ☞ Problems with modes?

Event-Driven Programming

- ☞ Instead of the user waiting on program, have the program wait on the user
- ☞ All communication from user to computer is done via "events"
- ☞ An *event* is something "interesting" that happens in the system
 - * Mouse button goes down
 - * Item is being dragged
 - * Keyboard button was hit

Event Examples



Major Issues

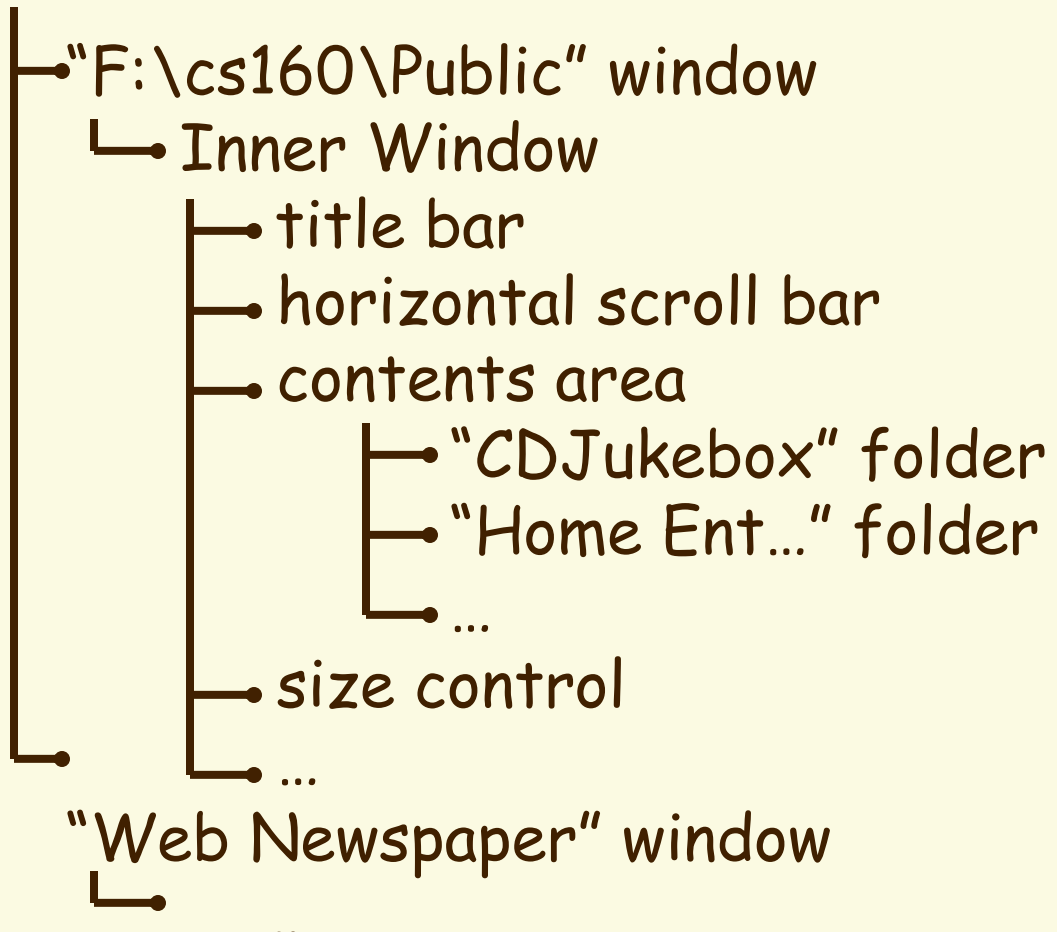
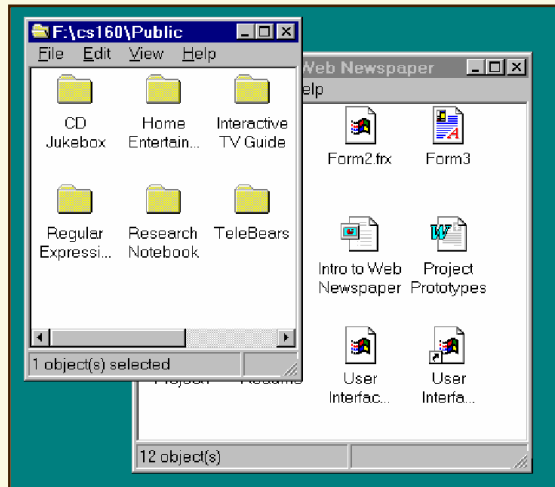
- How to decompose the UI into interactive objects?
- How to distribute input to the interactive objects
- How to partition between application & system software?
- Models for programming interactive objects
- Models for communications between objects

Interactor Tree

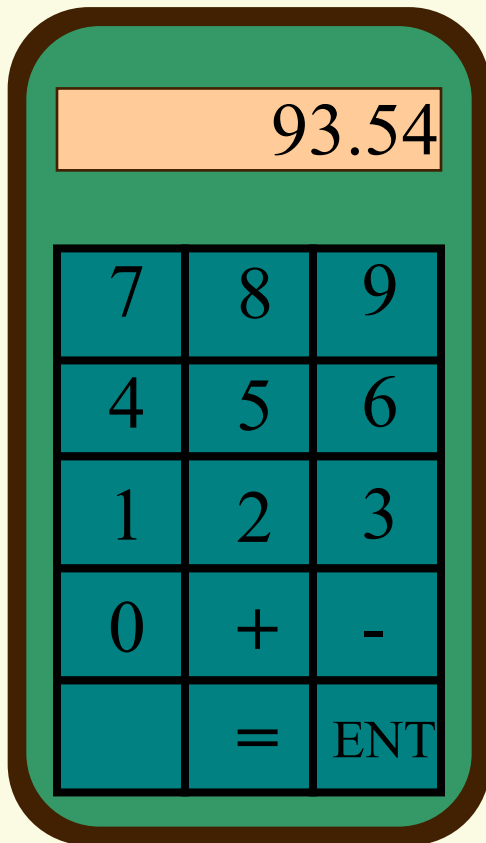
- ☞ Decompose interactive objects into a tree
 - * Interactive objects also known as "widgets"
 - * Based on screen geometry of objects
 - * Nested rectangles (except in *SVG* and some other vector languages which can handle polygons)
- ☞ Used for dispatching events
 - * Events are dispatched (sent) to code in widget
 - * The code then handles the event

Interactor Tree 1

Display Screen



Interactor Tree



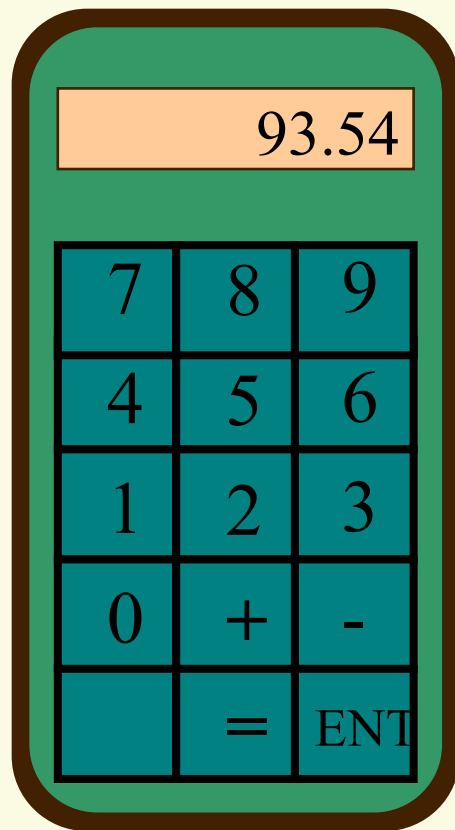
Display Screen

↳ Outer Win [*black*]

↳ ??????

Interactor Tree

Display Screen



↳ Outer Win [*black*]

↳ Inner Win [*green*]

↳ Result Win [*tan*]
↳ Result String

↳ Keypad [*Teal*]

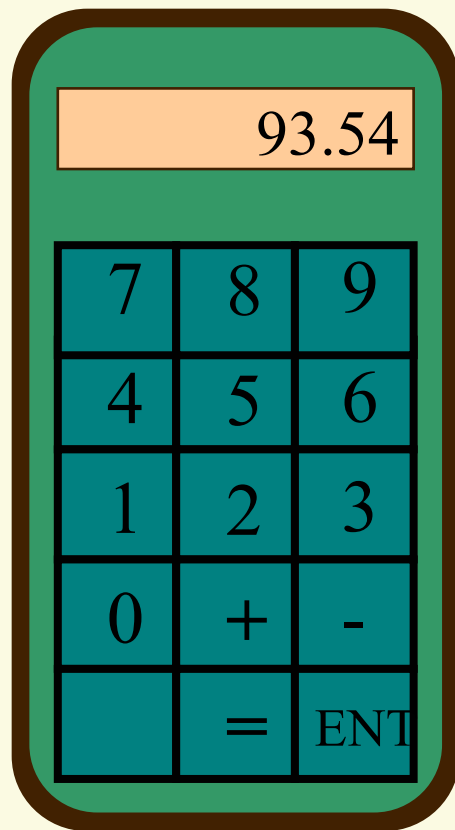
↳ = button
↳ - button
↳ + button
↳ 0 button

Event Registration

- ☞ To receive events, a widget normally needs to *register* its interest in that event with the WS
- ☞ Events are sent first to the focal widget (normally the one that's visible under the mouse)
- ☞ If that widget doesn't handle the event (not registered) the event goes to the next widget up the interactor tree that is registered.

Interactor Tree

Display Screen



↳ Outer Win [*black*]

↳ Inner Win [*green*]

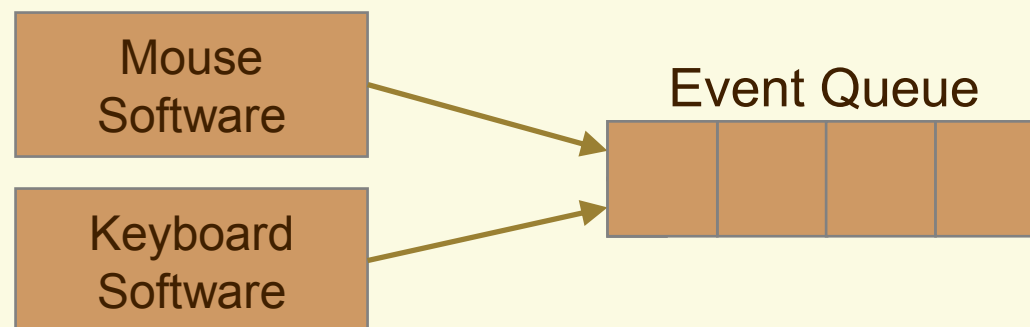
↳ Result Win [*tan*]
↳ Result String

↳ Keypad [*Teal*]

↳ = button
↳ - button
↳ + button
↳ 0 button

Event-Driven Programming

- ☞ All generated events go to a single *event queue*
 - * Provided by operating system
 - * Ensures that events are handled in the order they occurred
 - * Hides specifics of input devices from apps



Widgets

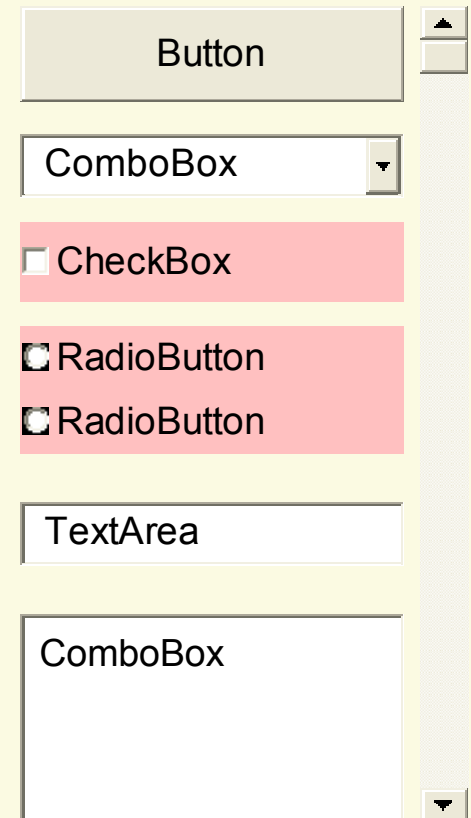
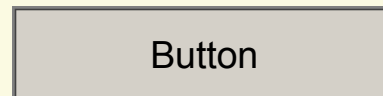
☞ Reusable interactive objects

☞ Handle certain events

- * Widgets say what events they are interested in
- * Event queue/interactor tree sends events to the right widget

☞ Update appearance

- * e.g. button up / button down



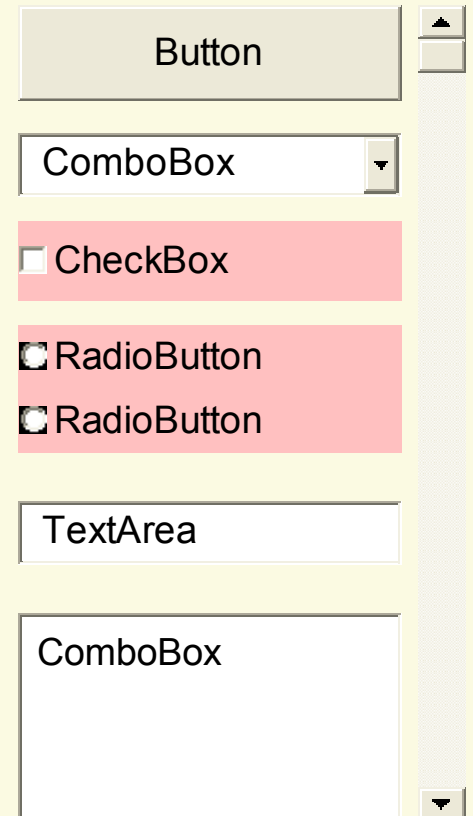
Widgets (cont.)

Generate some new events

- * "button pressed"
- * "window closing"
- * "text changed"

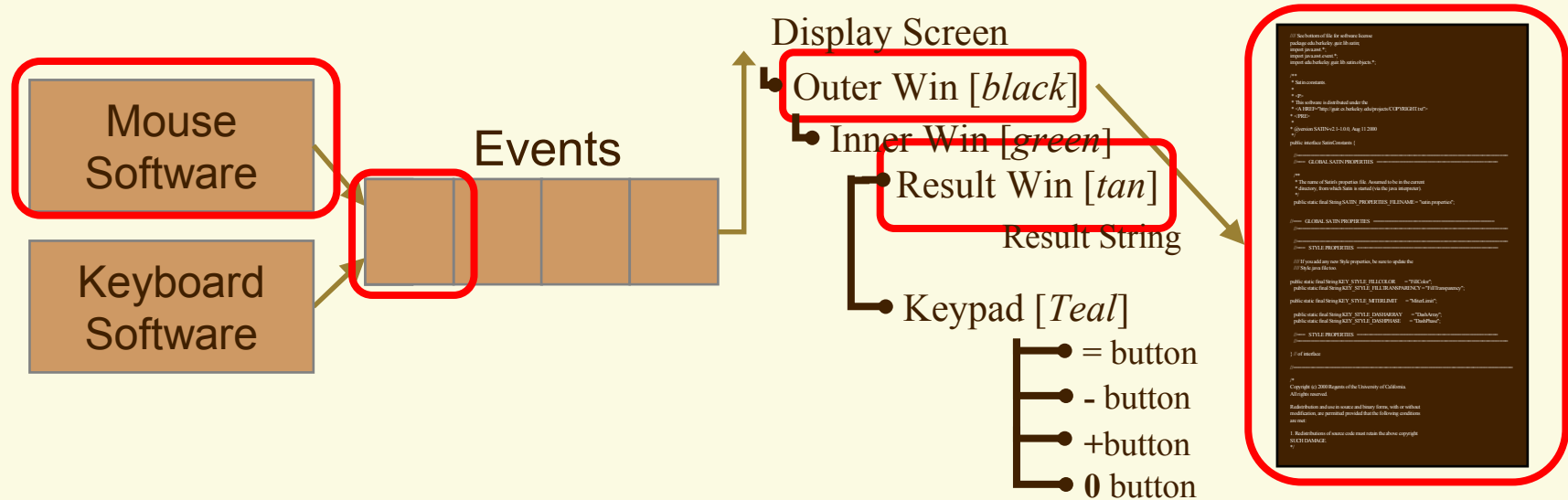
But these events are sent to interested listeners instead

- * Your code
- * Parent widgets that may need to redraw themselves



Main Event Loop

```
while (app is running) {  
  get next event  
  send event to right widget  
}
```



Platforms - PC

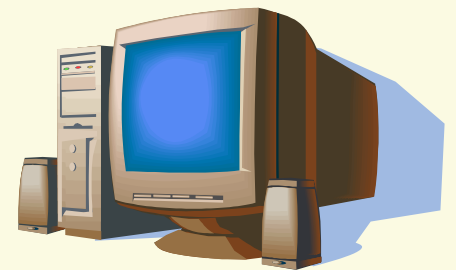
☞ For regular PC development, the options are:

☞ C#/Visual Basic/C++ (Visual Studio)

☞ Java

☞ Flash

☞ Rapid prototyping: Suede, Silk, Satin
(see quir.berkeley.edu/projects)



Platforms - Web

For web development one of the main issues is portability. Before designing your app, think about browsers for your user group.





- There is a lot more than IE and Netscape:
 - Mozilla/Opera
 - AOL: huge community, many versions with limited browsers
 - Old versions of IE and Netscape

Web standards

- ☞ Unfortunately, HTTP is a non-standard. The current version is HTML 4 (1997), but no browsers fully support it.
- ☞ Microsoft seems to have given up on HTML 4 in 1998.
- ☞ Reasonable support for HTML 4 in Netscape 7 and Mozilla.

Web standards

-  For portability, its best to stay with HTML 3.2
-  Javascript is the most portable script. But you'll probably still need browser-specific code.

Web standards - XML

- ☞ Fortunately, the situation looks better in future. XML should become the standard for web info exchange.
- ☞ XML provides data exchange, and complementary standards control formatting - XSL and XHTML.
- ☞ Good support in Mozilla, also IE and Netscape.

XML Graphics standards

- ☞ There are several standards for 2D graphics:
- ☞ **Flash** is widely used, but a closed proprietary standard and not based on XML
- ☞ **VML** (old) promoted by Microsoft - static 2D graphics, available in MS IE and PowerPoint
- ☞ **SVG**: dynamic 2D graphics, W3C and Mobile phone standard. Hardware support in the newest phones now shipping
- ☞ **XAML** - The foundation of Windows Vista

The Cell Phone Industry

- There are 6.5 billion people on earth
 - only about 1.2 billion in "developed" countries
- They will buy 800 million mobile phones this year
 - one person in eight on the planet
- That's 4x PC or TV unit sales
- Fraction of smartphones should reach 40% by 2009
 - most common "computer"



A Typical phone

e.g. LG VX8100 (free with service contract)

- 150-200 MHz ARM processor
- 32 MB ram
- 2 GB flash (not included)

Roughly a Windows-98 PC, plus:

- Camera
- AGPS (Qualcomm/Snaptrack)
- More DSPs, OpenGL GPU
- EV-DO (300 kb/s), Bluetooth

With improvements in other phones, Windows Smart phones have moved from "PDA" to "phone" category



What's coming

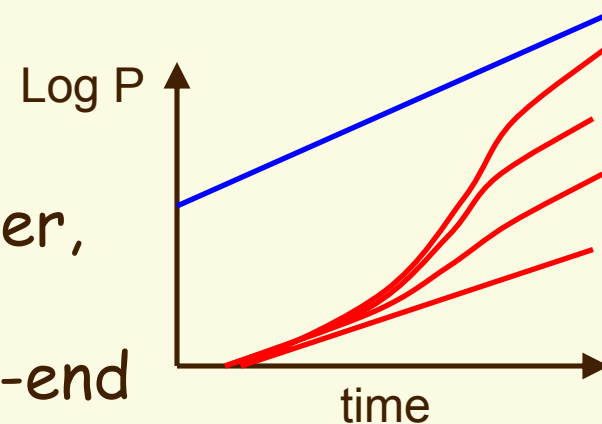
In the past, the platform was driven by **voice+messaging**

Now the high end is driven by **video, gaming, location,...**


The result is **diversification** of the platform, and **rapid catch-up** at the high end

e.g. Qualcomm is building 4 platforms:

1. Value platform (voice only)
2. ...
3. ...
4. Convergence platform (MP3 player, gamer, camera,...) several times the performance of today's high-end



The Inevitable

 In response to MIT's \$100 laptop, Microsoft last month proposed the cell phone computer for developing countries:



Microsoft Smart phones

Visual Studio 2005

- Managed code: i.e. virtual machine code

 - C#/Visual Basic: Best development support

 - C++/Native (binary) code for ARM processors



 - Best for compute-intensive apps (speech/vision)

- C# and Visual Basic support WSIWYG editing of the User Interface via Windows forms.

- Visual Studio supports "Managed C++" development for Windows but not for the Mobile Platform right now.


- Note: the SP5 phones contain the .NET Framework v1.0 - best to use those widgets.

Java

-  The i-mate SP5 phones also support Java runtime CLDC 1.1 and MIDP 1 and 2.
-  You should be able to develop J2ME apps for this configuration, but we haven't tested it.

Flash

 **Flash:** Supported already on some devices. See http://www.macromedia.com/mobile/supported_devices/handsets.html

 There is a free player available for experimentation called "Flashhack" or "Menuhack" - use at your own risk.

 Hardware support for Flash coming in phones soon, maybe this year.

Other cell phone systems - BREW

- ☞ **BREW** is Qualcomm's "Binary Runtime Environment for Wireless" aka Verizon's "Get It Now" service.
- ☞ Something like the WIN32 API, but smaller. BREW includes support for
 - ☞ GPS-ONE - much better than normal GPS
 - ☞ Streaming media and 3D graphics (OpenGL)
 - ☞ Camera, Audio, Bluetooth, Serial etc.
 - ☞ BT/serial support limited on actual phones
- ☞ Large distribution channel for apps built with BREW through over-the-air download.

Summary

- 📄 Concepts:
- 📄 2D vector graphics
- 📄 Raster graphics - color, anti-aliasing
- 📄 Interactors
- 📄 Event-driven programming
- 📄 Development platforms