CS 160: Lecture 12

Professor John Canny
Fall 2004

Outline

Output
- basic 2-D computer graphics
- color models

Input
- event overview
- windowing systems
- window events
- event dispatching

Widget communication & layout

2-D Computer Graphics

- Models for images
  - strokes, pixels, regions
- Coordinate systems
  - device, physical
- Canvas
- Drawing
  - paths, shapes, text
- Clipping

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**
  - group pixels (some on, some off)
- **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????

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).

Region Model

- Use the stroke model to outline region
- Fill the region with colors & blendings (i.e., patterns)
- Advantages??
  * allows representation of filled shapes w/ little memory
  * independence from display resolution
- Text represented this way & converted to bitmaps inside of the printer

Outline Fonts

- Used by both Postscript & TrueType

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

Drawing

- 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, Moiger, Times Roman, Courier
  * defines the general shape of the characters
  * some are mono-spaced ("i" gets same space as "o")
  * 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. Bitmap fonts
  * need pixels to draw on screen so may store as BM
  * problems: takes lots of space font in several sizes
  * instead store as a closed shape (e.g., outline only)
  * easy to scale to any size and convert to bitmap

Vector vs. Raster Formats

- Vector:
  * Macromedia Flash.
  * SVG (Scalable Vector Graphics), a W3C standard.
  * VML (Microsoft), Powerpoint animation.
- Raster:
  * Jpeg: Better for smooth images
  * Gif, Png: Better for line art or "South Park" animation

Color Models

- 256 levels for each primary adequate
  * \( \rightarrow \) 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 light is
  * high is pure, low means it is mixed w/ white/gray
  * value is intensity (dark vs. light)

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"(\( \alpha \)) to encode transparency (e.g. png)
- \( C = \alpha \times C_f + (1-\alpha) \times C_r \) - each color channel

Break
Sequential Programs

- Program takes control, prompts for input
- Examples include
  - command-line prompts (DOS, UNIX)
  - LISP interpreter
- The user waits on the program
  - program tells user it’s ready for more input
  - user enters more input

Sequential Programs (cont.)

- Architecture
  - Program reads in a line of text
  - Program parses the text
  - Program evaluates the result
  - Maybe some output
  - Loop back to beginning
- But how do you model the many actions a user can take?
  - for example, a word processor?
  - need to do printing, editing, inserting, whenever user wants to

Sequential Programs (cont.)

- Usually end up with lots of modes
  - lots of state variables
- Other examples of modes
  - paint programs (line, bucket-fill, rectangle, etc)
  - universal remotes with TV / VCR mode
  - vi edit mode and command mode
- Problems with modes?

Sequential Programs (cont.)

- Problems with modes?
  - gets confusing if too many modes
  - can be easy to make errors
  - need feedback as to what mode you are in
  - how to switch between modes?
- We’ll need a more advanced model to simplify windows programming

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 Example
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

Windowing Systems

- Partitioning to prevent chaos
- Infrastructure to support common services
- Two major aspects
  - Software services to applications
    - Create and organize windows
    - Implement interaction in those windows
  - Window manager
    - UI allowing user to control size & placement of windows

Interactor Tree

- Decompose interactive objects into a tree
  - Interactive objects also known as "widgets"
  - Based on screen geometry of objects
  - Nested rectangles
- Used for dispatching events
  - Events are dispatched (sent) to code in widget
  - The code then handles the event
- Variety of methods for dispatching events
  - Return to this later

Interactor Tree 1

- Display Screen
- "F:\cs160\Public" window
- Inner Window
- Title bar
- Horizontal scroll bar
- Contents area
- "CDJukebox" folder
- "Home Ent..." folder
- Size control
- "Web Newspaper" window

Interactor Tree 2

- Display Screen
  - Outer Win [black]
  - " ????? ?

Interactor Tree 2

- Display Screen
  - Outer Win [black]
  - Inner Win [green]
  - Result Win [tan]
  - Result String
  - Keypad [Teal]
  - = button
  - - button
  - + button
  - 0 button
**Windows**

- Top level windows known as *root windows*
  - provide UI abstraction for multiple apps
  - windowing system arbitrates interactive resources
- Each root window belongs to an app.
  - all descendant windows belong to same app
  - violated by ActiveX
- Windows vs. widgets/controls
  - X, NeXTStep, MS Windows
  - everything is window

**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 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
  - custom code goes there

**Main Event Loop**

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

**Platforms - PC**

- For regular PC development, the options are:
  - C++/C#/VBasic (Visual Studio)
  - Java
  - Rapid prototyping: Suede, Silk, Satin (see guir.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. (but tables are different in most browsers)

Web standards

For portability, it's 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 is going to 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 two standards for 2D graphics:
- VML (old) promoted by Microsoft - static 2D graphics, available in MS IE now.
- SVG (new) dynamic 2D graphics, the latest W3C standard. No browser support natively, but plug-in available (Adobe), and custom builds of Mozilla support it.

PDAs

Two options for native development:
- MS Embedded Visual Tools 3.0 (VB and C++) - includes emulators for all platforms (download).
- MS Visual Studio .NET (huge!) includes tools for XML exchange.
- Java: Chai VM for HP Jornadas etc.
- Usually well behind PC Java - no Jini support
- Flash: Interesting choice for small devices, better use of limited screen space, but check functionality (esp. script execution).
Cell phones - BREW

- BREW is Qualcomm's "Binary Runtime Environment for Wireless"
- Something like the WIN32 API, but smaller. BREW 2.0 includes support for GPS-one - though no providers yet
- Can get GPS info through serial port on emulator PC.

10/13/2004 49

XML services - Mark Logic CIS

- When prototyping mobile apps, the backend services play a large role.
- Since XML is the lingua franca for inter-device communication, a general-purpose XML database engine is ideal.
- Mark Logic CIS server for CS160 running at the URL given in class.

10/13/2004 50

XML services - ML CIS

- Background reading on XQuery:

10/13/2004 51

Summary

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

10/13/2004 51