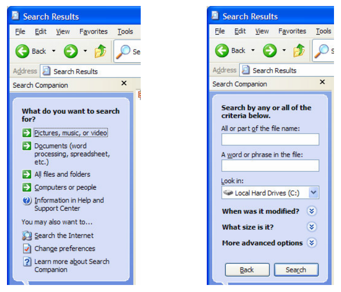
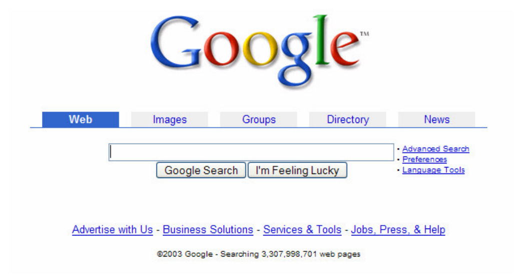
Lecture 17: Output Models 2

UI Hall of Shame or Hall of Fame?


Search Results
File Edit View Favorites Tools
Back
Address Search Results
Search Companion
What do you want to search for?
Pictures, music, or video
Documents (word processing, spreadsheet, etc.)
All files and folders
Computers or people
Information in Help and Support Center
You may also want to...
Search the Internet
Change preferences
Learn more about Search Companion
Search Results
File Edit View Favorites Tools
Back
Address Search Results
Search Companion
Search by any or all of the criteria below.
All or part of the file name:
A word or phrase in the file:
Look in:
Local Hard Drives (C:)
When was it modified?
What size is it?
More advanced options
Back
Search

UI Hall of Fame or Shame?


Google™
Web
Images
Groups
Directory
News
Advanced Search
Preferences
Language Tools
Google Search
I'm Feeling Lucky
Advertise with Us - Business Solutions - Services & Tools - Jobs, Press, & Help
©2003 Google - Searching 3,307,998,701 web pages

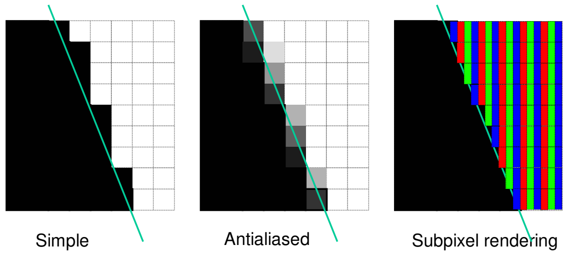
Today's Topics

- Antialiasing
- Alpha compositing
- Transforms
- Clipping
- Painting tricks

Transparency

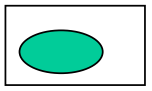
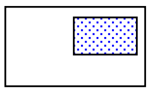
- **Alpha** is a pixel's transparency
  - from 0.0 (transparent) to 1.0 (opaque)
  - 32-bit RGBA pixels: each pixel has red, green, blue, and alpha values
- Uses for alpha
  - Antialiasing
  - Nonrectangular images
  - Translucent components

Antialiasing and Subpixel Rendering



Alpha Compositing

- Compositing rules control how source and destination pixels are combined
- Source
  - Image
  - Stroke drawing calls
- Destination
  - Drawing surface

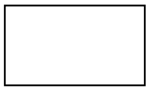
Porter-Duff Alpha Compositing Rules

Source pixel: Rs, Gs, Bs, As
Dest pixel: Rd, Gd, Bd, Ad

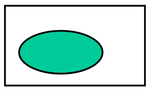
1. Premultiply RGB by A
   {RGB}s = {RGB}rs * As
   {RGB}d = {RGB}rd * Ad
2. Compute weighted combination of source and dest pixel
   {RGB}d = {RGB}s*fs + {RGB}d*fd
   Ad = As*fs + Ad*fd
   for some weights fs, fd
3. Postdivide RGB by A
   {RGB}d = {RGB}d / Ad if Ad != 0

Simple Copying

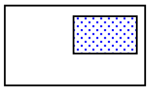
- clear (fs=0, fd=0)
  - {RGB}d = 0
  - Ad = 0



- src (fs=1, fd=0)
  - {RGB}d = {RGB}s
  - Ad = As

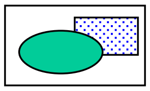


- dst (fs=0, fd=1)
  - {RGB}d = {RGB}d
  - Ad = Ad

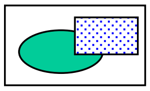


Layering

- src over dst
  - {RGBA}d = {RGBA}s + {RGBA}d*(1-As)

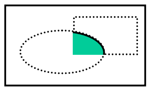


- dst over src
  - {RGBA}d = {RGBA}d + {RGBA}s*(1-Ad)

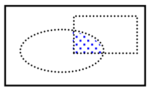


Masking

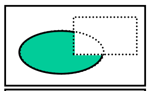
- src in dst
  - {RGBA}d = {RGBA}s*Ad



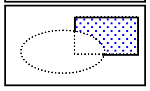
- dst in src
  - {RGBA}d = {RGBA}d*As



- src out dst
  - {RGBA}d = {RGBA}s*(1-Ad)



- dst out src
  - {RGBA}d = {RGBA}d*(1-As)



Other Masking

- src atop dst = src over dst − src out dst
  - {RGBA}d = {RGBA}s*Ad + {RGBA}d*(1-As)
- dst atop src = dst over src − dst out src
  - {RGBA}d = {RGBA}s*(1-Ad) + {RGBA}d*As
- src xor dst = src out dst + dst out src
  - {RGBA}d = {RGBA}s*(1-Ad) + {RGBA}d*(1-As)

Coordinate Transforms

- Translation
  - moves origin by dx, dy
- Scaling
  - multiplies x by sx and y by sy
- Rotation
  - rotates by theta around an axis point x, y
- Use coordinate transforms to make drawing easier

Component Model Effects

- Changing Graphics passed to children
  - Transforms: rotation, zooming
  - Clipping: setting new clipping regions
- Wrapping Graphics passed to children
  - Intercept child calls and modify or capture them
- Painting onto offscreen images and then transforming the images
  - Blur, shimmer, masking
- Using components as rubber stamps
  - Table, list, and tree cell renderers

Scene Graphs

- Traditional 2D toolkits are too limited for many graphical effects
  - View hierarchy is a tree (can't share views)
  - Parents must enclose descendents (and clip them)
  - Parents translate children, but don't otherwise transform them
- Piccolo toolkit (designed for zooming user interfaces)
  - View hierarchy is actually a **graph**
  - Components can translate, rotate, scale
  - Parents transform but **don't clip** their children by default
  - Input events and repaint requests are transformed too