Lecture 5: Output Models

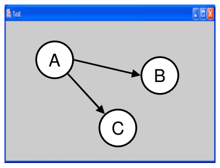
Today's Topics

- Output models
- Drawing
- Rasterization
- Color models

Three Output Models

- Components
  - Graphical objects arranged in a tree with automatic redraw
  - Example: Label object, Line object
  - Also called: views, interactors, widgets, controls, retained graphics
- Strokes
  - High-level drawing primitives: lines, shapes, curves, text
  - Example: drawText() method, drawLine() method
  - Also called: vector graphics, structured graphics
- Pixels
  - 2D array of pixels
  - Also called: raster, image, bitmap

Example: Designing a Graph View

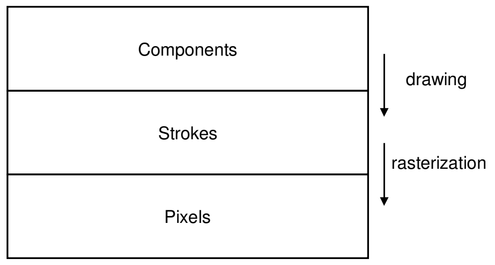


- Component model
  - Each node and edge is a component
  - A node might have two subcomponents: circle and label
- Stroke model
  - Graph view draws lines, rectangles and text
- Pixel model
  - Graph view has pixel images of the nodes

Issues in Choosing Output Models

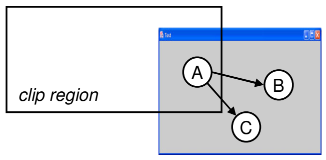
- Layout
- Input
- Redraw
- Drawing order
- Heavyweight objects
- Device dependence

How Output Models Interact

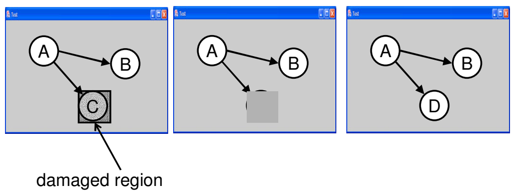


Drawing in the Component Model

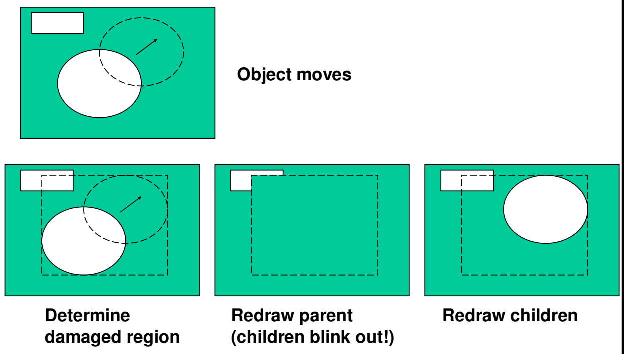
- Drawing goes top down
  - Draw self (using strokes or pixels)
  - For each child component,
    - If child intersects clipping region then
      - intersect clipping region with child's bounding box
      - recursively draw child with clip region = intersection



Damage and Automatic Redraw

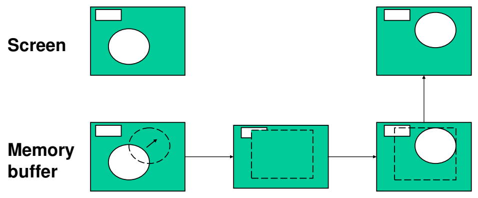


Naïve Redraw Causes Flashing Effects



Double-Buffering

- Double-buffering solves the flashing problem



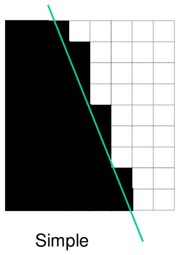
Stroke Model

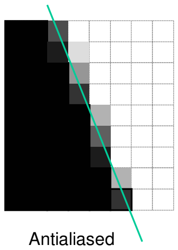
- Drawing surface
  - Also called drawable (X Windows), GDI (MS Win)
  - Screen, memory buffer, print driver, file, remote screen
- Graphics context
  - Encapsulates drawing parameters so they don't have to be passed with each call to a drawing primitive
  - Font, color, line width, fill pattern, etc.
- Coordinate system
  - Origin, scale, rotation
- Clipping region
- Drawing primitives
  - Line, circle, ellipse, arc, rectangle, text, polyline, shapes

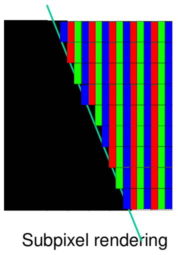
Rasterization

- Is (0,0) the center of the top-left pixel, or is it the upper left corner of the pixel?
  - MS Win: center of pixel
  - Java: upper left corner
- Where is line (0,0) – (10,0) actually drawn?
  - MS Win: endpoint pixel excluded
  - Java Graphics: pen hangs down and right
  - Java Graphics2D: antialiased pen, optional ½ pixel adjustments made for compatibility
- Where is empty rectangle (0,0) – (10,10) drawn?
  - MSWin: connecting those pixels
  - Java: extends one row below and one column right
- Where is filled rectangle (0,0) – (10,10) drawn?
  - MSWin: 121 pixels
  - Java: 100 pixels

Antialiasing and Subpixel Rendering


Simple


Antialiased


Subpixel rendering

Pixel Model

- Pixel model is a rectangular array of pixels
  - Each pixel is a vector (e.g., red, green, blue components), so pixel array is really 3 dimensional
- Bits per pixel (bpp)
  - 1 bpp: black/white, or bit mask
  - 4-8 bpp: each pixel is an index into a color palette
  - 24 bpp: 8 bits for each color
  - 32 bpp: 8 bits for each color + alpha channel
- Color components (e.g. RGB) are also called channels or bands
- Pixel model can be arranged in many ways
  - Separate planes (RRR...GGG...BBB...) vs. interleaved (RGB RGB RGB...)
  - Scanned from top to bottom vs. bottom to top

Transparency

- **Alpha** is a pixel's transparency
  - from 0.0 (transparent) to 1.0 (opaque)
  - so each pixel has red, green, blue, and alpha values
- Uses for alpha
  - Antialiasing
  - Nonrectangular images
  - Translucent components
  - Clipping regions with antialiased edges

BitBlt

- BitBlt (bit block transfer) copies a block of pixels from one image to another
  - Drawing images on screen
  - Scrolling
  - Double-buffering
  - Clipping with nonrectangular masks
- Alpha compositing rules control how pixels from source and destination are combined
  - More about this in a later lecture

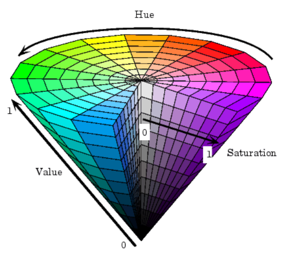
Image File Formats

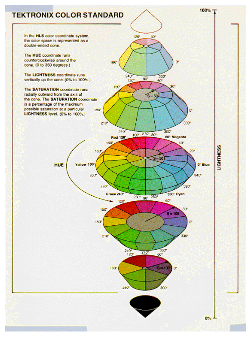
- GIF
  - 8 bpp, palette uses 24-bit colors
  - 1 color in the palette can be transparent (1-bit alpha channel)
  - lossless compression
  - suitable for screenshots, stroked graphics, icons
- JPEG
  - 24 bpp, no alpha
  - lossy compression: visible artifacts (dusty noise, moire patterns)
  - suitable for photographs
- PNG
  - lossless compression
  - 1, 2, 4, 8 bpp with palette
  - 24 or 48 bpp with true color
  - 32 or 64 bpp with true color and alpha channel
  - suitability same as GIF
  - better than GIF, but no animation
  - IE supports transparent pixels, but not full alpha transparency

Color Models

- RGB: cube
  - Red, green, blue
- HSV: hexagonal cone
  - Hue: kind of color
    - Angle around cone
  - Saturation: amount of pure color
    - 0% = gray, 100% = pure color
  - Value: brightness
    - 0% = dark, 100% = bright
- HLS: double-hexagonal cone
  - Hue, lightness, saturation
  - Pulls up center of HSV model, so that only white has lightness 1.0 and pure colors have lightness 0.5
- Cyan-Magenta-Yellow(-Black)
  - Used for printing, where pigments absorb wavelengths instead of generating them

HSV & HLS





Hints for Debugging Output

- Something you’re drawing isn’t appearing on the screen. Why not?
  - Wrong place
  - Wrong size
  - Wrong color
  - Wrong z-order