HUMAN COMPUTER INTERACTION

Lecture 10: Output

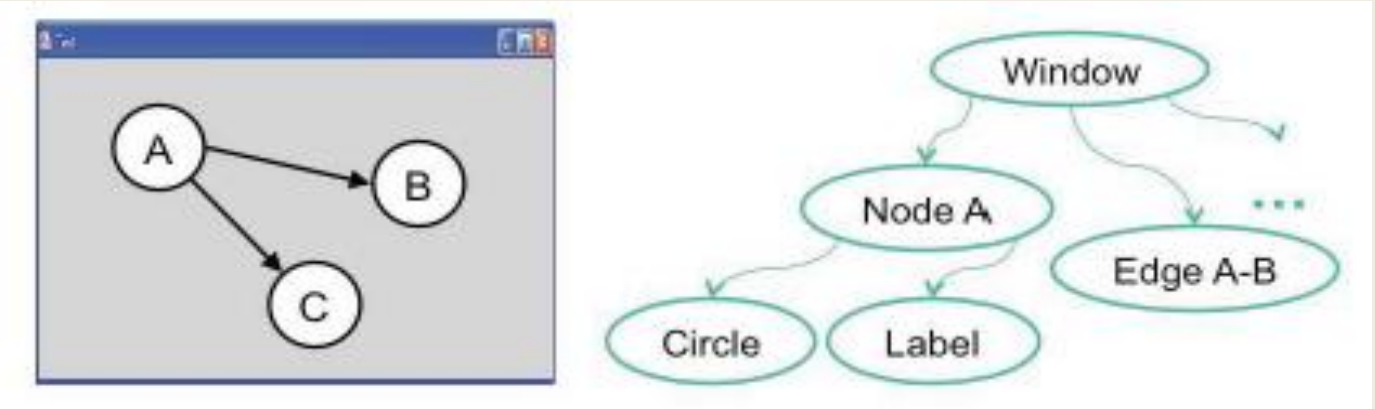
Today’s Topic

- Output approaches
- Drawing
- Rasterization
- Declarative programming

Three Output Approaches

- Objects
  - *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*
  - *Examples: 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

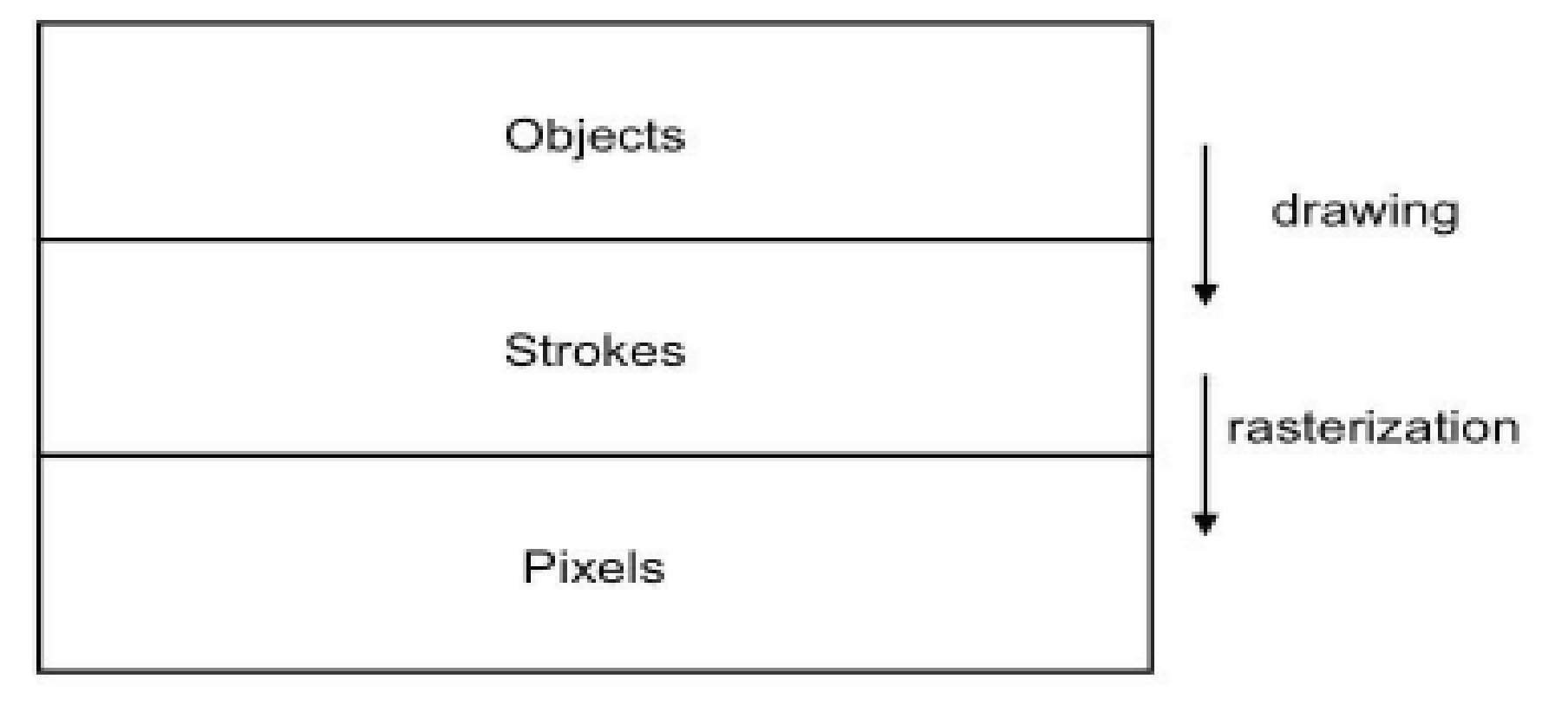


- Component model
  - *Each node and edge is a component*
  - *A node might have two subcomponents: circle and label*
- Stroke model
  - *Graph view draws line, circles and text*
- Pixel model
  - *Graph view has pixel images of the nodes*

Issues in choosing Output Models

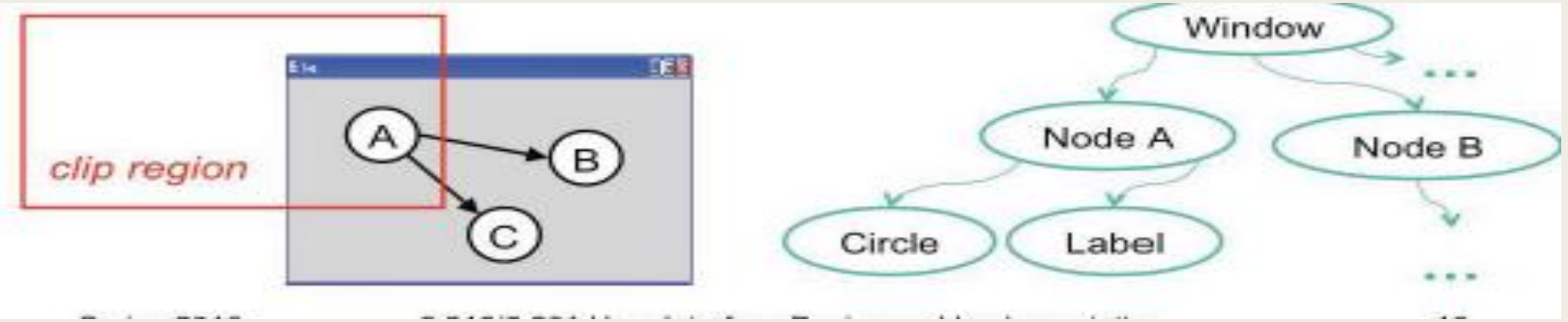
- Layout
- Input
- Redraw
- Drawing order
- Heavyweight objects
- Device dependence

How Output Approaches Interact

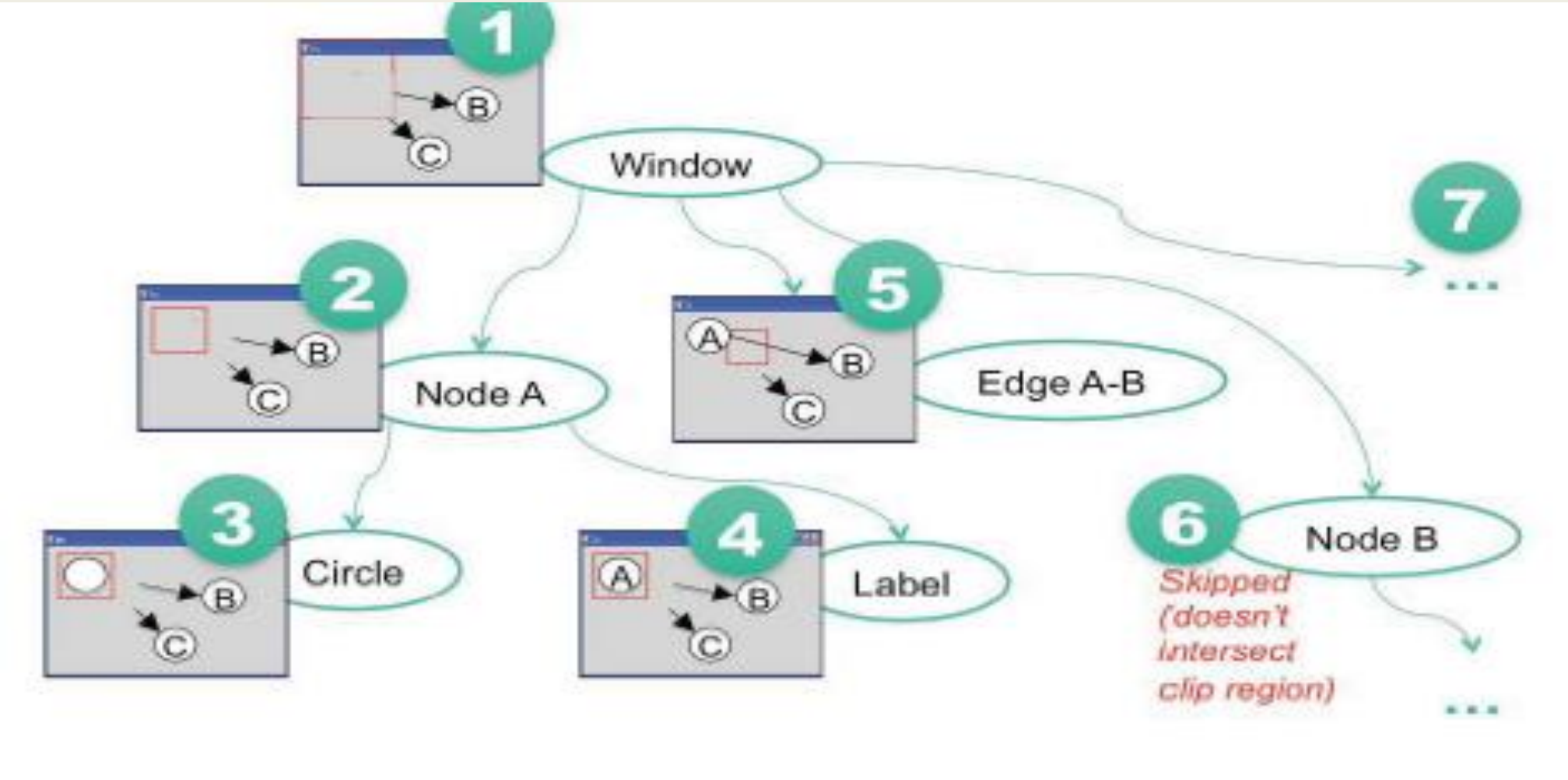


Drawing a View Tree

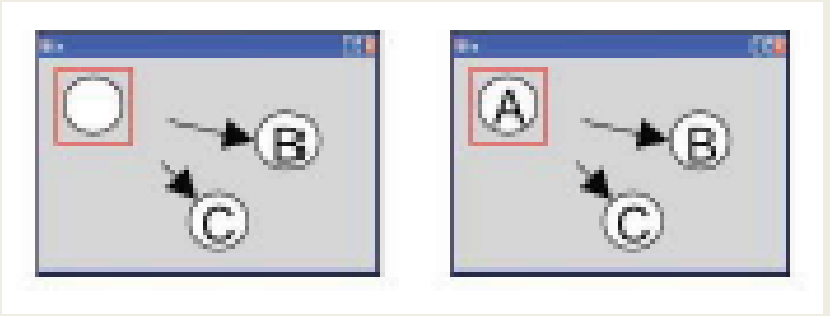
- Drawing goes top down
  - *Draw self (using strokes or pixels)*
  - *For each child component,*
    - If child intersects clipping region then
      - *Intersect clipping region with child's bounding box*
      - *Recursively draw child with clip region set to the intersection*



Redraw Example

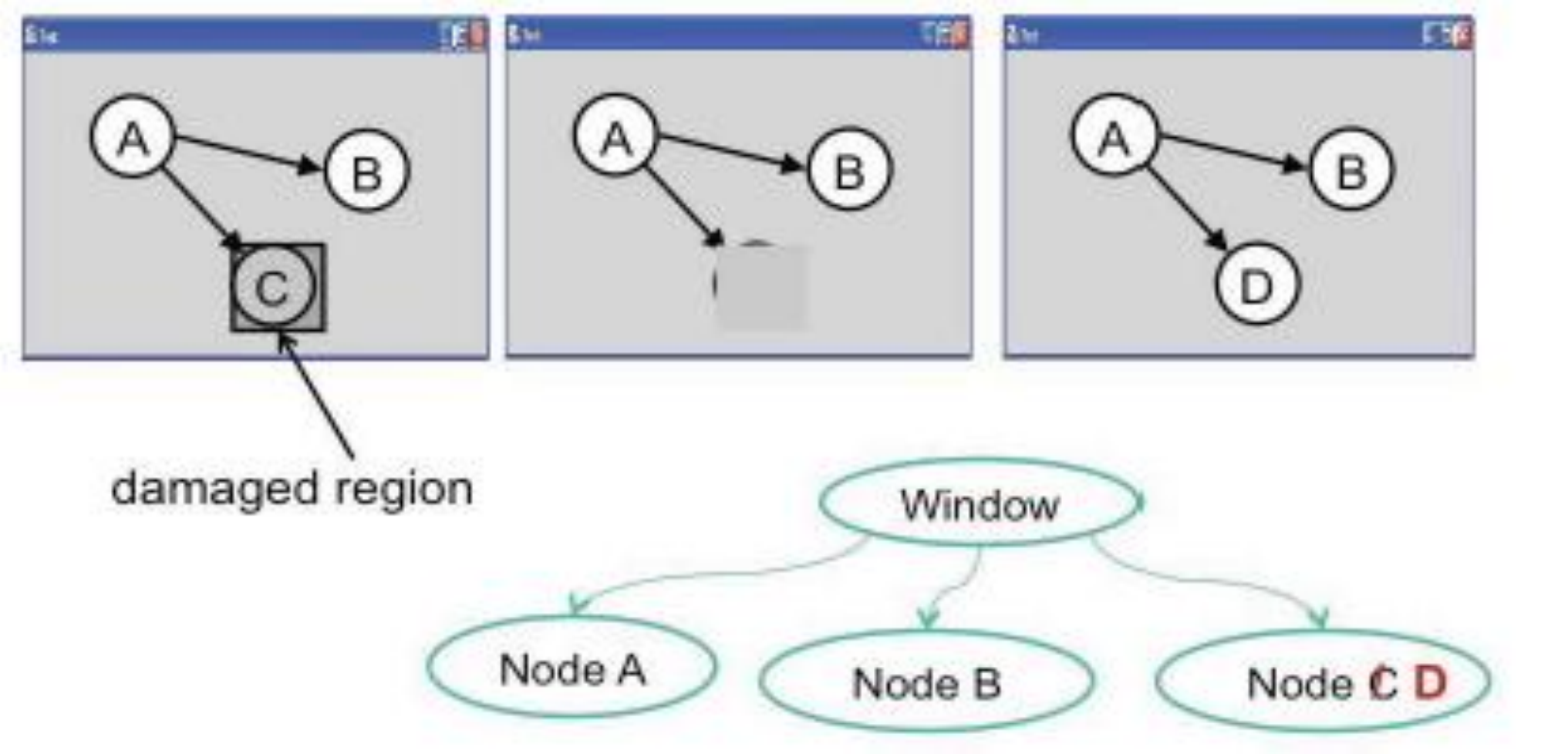


Z Order

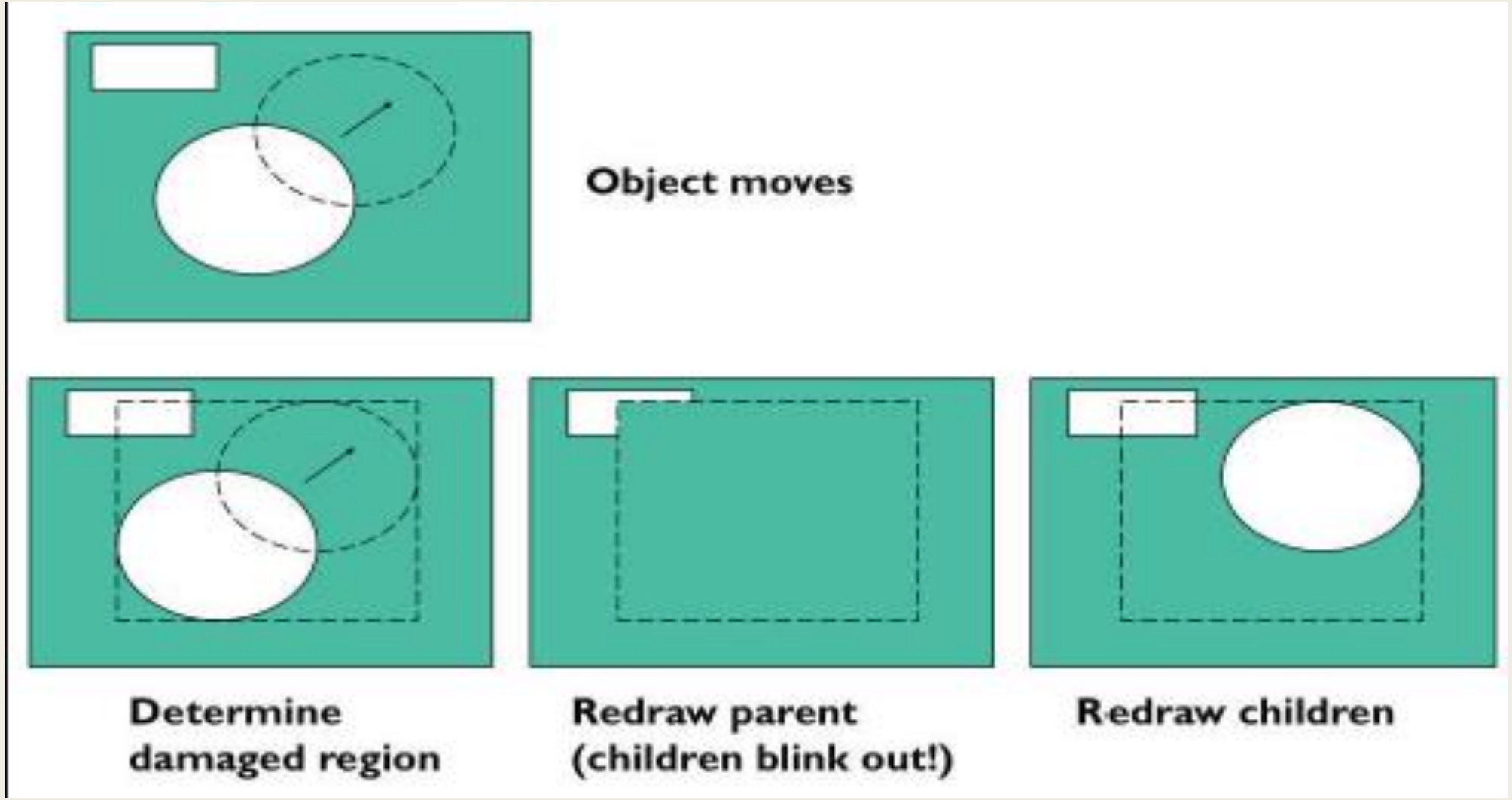


- 2D GUIs are really “2 ½ D”
  - *Drawing order produces layers*
  - *Not a true z coordinate for each object, but merely an ordering in the z dimension*
- View tree and redraw algorithm dictate z order
  - *Parents are drawn first, underneath children*
  - *Older siblings are drawn under younger ones*
    - Flex, HTML, most Gui toolkits and drawing programs behave this way
    - Java swing is backward: last component added (highest index) is drawn first

Damage and Automatic Redraw

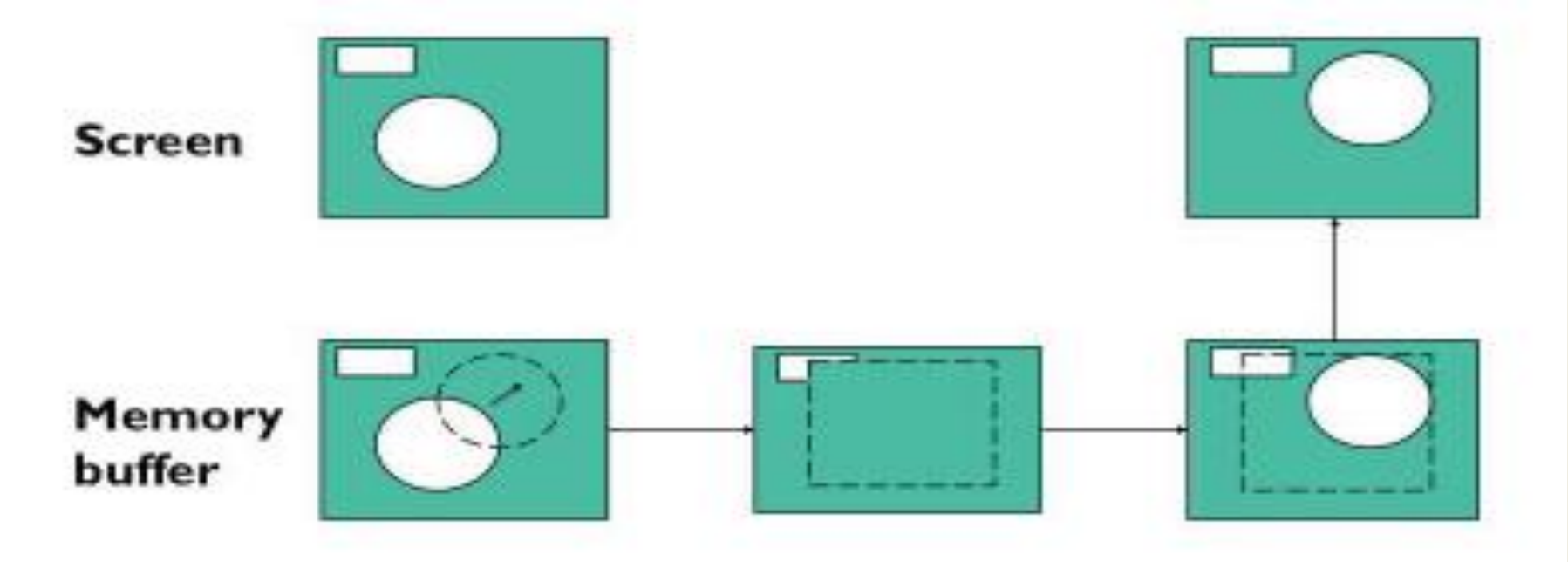


Naive Redraw causes Flashing Effects



Double Buffering

- Double-buffering solves the flashing problem



Going From Objects to Strokes

- Drawing method approach
  - *E.g. Swing paint() method*
  - *Drawing method is called directly during redraw: override it to change how component draws itself*
- Retained graphics approach
  - *E.g. Adobe Flesx*
  - *Stroke calls the recorded and played back at redraw time*
- Differences
  - *Retained graphics is less error prone*
  - *Drawing method gives more control and performance*

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 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*

HTML Canvas in One Slide

HTML element

```
<canvas width=1000
   height=1000></canvas>
```

graphics context

```
var ctx = canvas.getContext
   (“2d”)
```

coordinate system

```
ctx.translate()
ctx.rotate()
ctx.scale()
```

color, font, line style, etc.

```
ctx.strokeStyle = “rgb
   (0.5,0.5,0.5)”
ctx.fillStyle = …
ctx.font = “bold 12pt sans-
   serif”
ctx.lineWidth = 2.5
```

drawing primitives

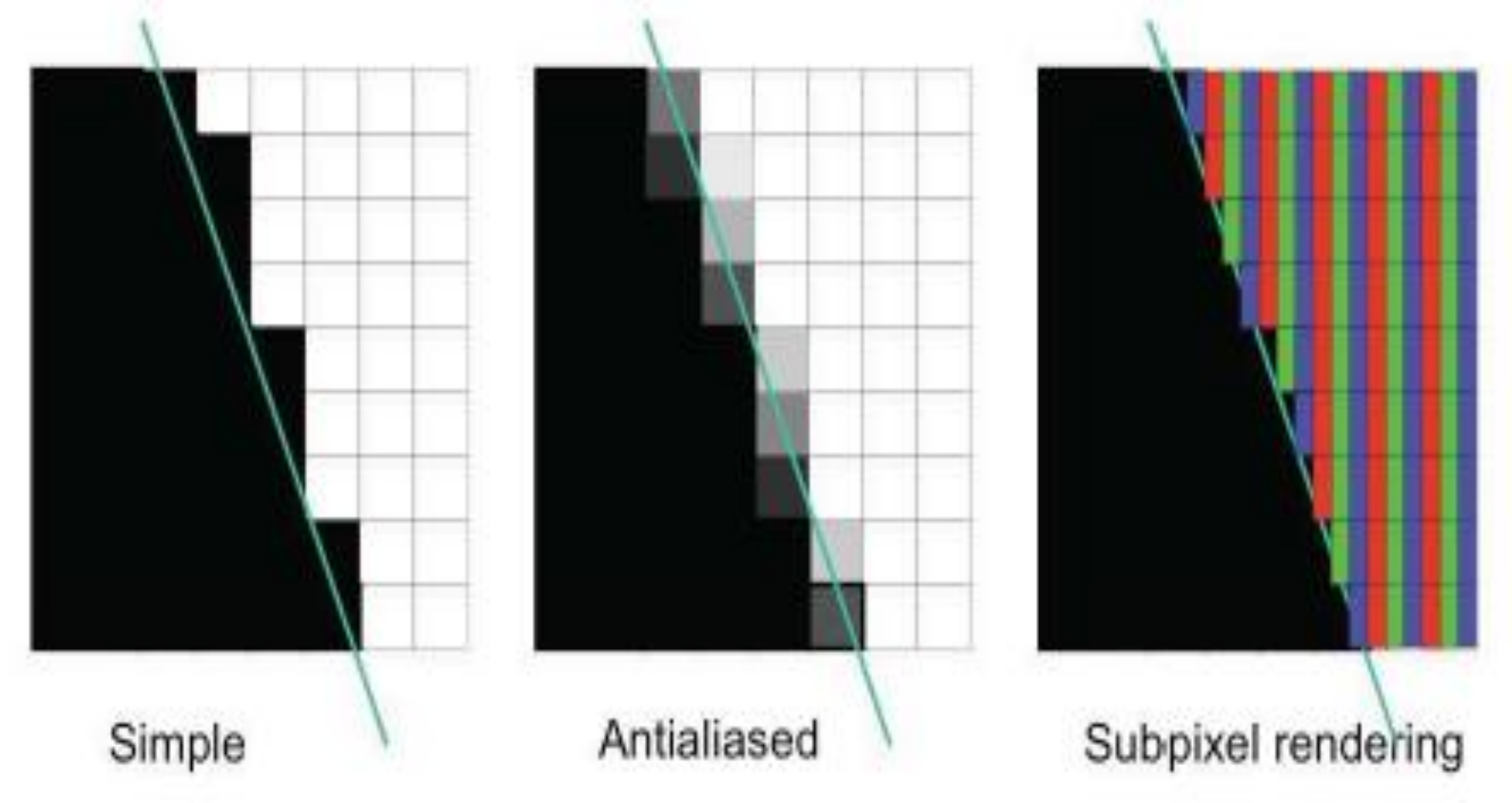
```
ctx.beginPath();
ctx.moveTo(0,0)
ctx.lineTo(500,500)
ctx.stroke()

ctx.beginPath()
ctx.arc(500,500,100,0,
   2*Math.PI,false)
ctx.fill()
```

clipping

```
ctx.beginPath()
ctx.rect(0, 0, 200, 300)
ctx.clip()
```

Antialiasing and Subpixel Rendering



Pixel Approach

- pixel approach is a rectangular array of pixels
  - *Each pixel is a vector (e.g. R,G,B components), so pixel array is really 3 dimensional*
- Bits per pixel(bpp)
  - *1 bpp: black/white, or bit mask*
  - *4-8 :each pixel is an index into color palette*
  - *24 bpp: 8 bit for each color*
- Color components (e.g. RGB) are also called channels or bands
- Pixel model can be arranged in many ways
  - *Packed into words (RGBR GBRG …) or loosely (RGB-RGB)*
  - *Separate planes *RRR…GGG…BBB…) vs interleaves (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
  - *Anti aliasing*
  - *Nonrectangular images*
  - *Translucent components*
  - *Clipping regions with antialiasing edges*

BitBlt

- BitBlt (Bit Block Transfer) copies a block of pixels from one image to another
  - *Drawing images on screen*
  - *Double-buffering*
  - *Scrolling*
  - *Clipping with nonrectangular masks*
- 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 chanel)*
  - *Losless compression*
  - *Suitable for screenshots, stroked graphics, icons*
- JPEG
  - *24 bpp, no alpha*
  - *Lossy compression: visible artifacs (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*

Hints for debugging Output

- Something you're drawing isn't appearing on the screen. Why no?
  - *Wrong place*
  - *Wrong size*
  - *Wrong color*
  - *Wrong z-order*