Image Panel & Image Navigator

Review

- Graphics2D class for drawing shapes in a component
  - `paintComponent(Graphics g)`
  - `Graphics2D g2 = (Graphics2D)g;`

- (0, 0) is the component's upper-left corner
- Shapes: Line2D, Rectangle2D, . . .

- `draw(Shape s)`
- `fill(Shape s)`
- `drawImage(Image img,`
  - `int dx1, int dy1, int dx2, int dy2,`
  - `int sx1, int sy1, int sx2, int sy2)`
- `drawString(String str, int x, int y)`

A Similar Problem: Double Sponge Bob

- Let's solve a problem that is similar to the Record Indexer project
  - Create two frame windows, each containing an instance of the "Sponge Bob" component
  - Allow each window to be scaled and translated
  - Keep the origins of the two windows in synch so that translating one window will perform an equivalent translation in the other window
  - Demo: Double Sponge Bob (Scaling, Translation, Synch)

The Hard Part: Coordinate Transformations

- Shape dimensions and locations are defined in terms of unscaled, untranslated coordinates
  - We call these "world coordinates"

- Each window has its own scaling and translation settings
- This complicates three things:
  - Drawing shapes in each window at the appropriate scale and location
  - Doing "hit testing" on the shapes when mouse events occur
  - Keeping the translation settings of the two windows in synch

- When drawing shapes in a window, we must convert the shape dimensions and locations defined in "world coordinates" to "device coordinates" that can be used for drawing
- This conversion must account for the current scale and translation settings of the window

The Hard Part: Coordinate Transformations

- Example
  - Suppose that a window currently has a scaling factor of 0.5, and a translation of 10 pixels in the X direction and 20 pixels in the Y direction (i.e., Point (10, 20) is in the top-left corner of the window)
  - A shape that is 400 pixels wide in “world coordinates” will be drawn only 200 pixels wide in “device coordinates”
  - A shape that is located at position (250, 300) in “world coordinates” will be drawn at position (240, 280) in “device coordinates”

Scaling

- Start with code that implements neither scaling nor translation nor window synchronization
  - Double Sponge Bob (No Transforms)
- Now, add code to implement scaling
  - Add `scale` field to `DrawingComponent` and initialize to `1.0` in constructor
  - Add `setScale` method to `DrawingComponent`
  - Add slider change listener to `DrawingFrame` that calls `DrawingComponent.setScale`
  - In `DrawingComponent.paintComponent` add a call to Graphics2D.scale
  - Continue to draw shapes with world coordinates, and let Java automatically scale them as they are drawn
  - Double Sponge Bob (Scaling)

Translation

- Start with code that implements scaling but not translation or window synchronization
  - Double Sponge Bob (Scaling)
- Now, add code to implement translation
  - Add fields to `DrawingComponent` for tracking how far the user has translated the drawing (in world coordinates), and initialize them to `(0,0)` in the constructor
  - In `DrawingComponent.paintComponent` add a call to Graphics2D.translate
  - Add listeners for implementing mouse-based translation of shapes. Use `AffineTransform` class to convert from device coordinates to world coordinates. Update fields that track how far the user has translated the drawing.
  - Double Sponge Bob (Scaling & Translation)

Window Synchronization

- Start with code that implements scaling and translation but not window synchronization
  - Double Sponge Bob (Scaling & Translation)
- Now, add code to implement translation
  - Add `DrawingListener` interface
  - Add list of listeners and `addDrawingListener` method to `DrawingComponent`
  - Add `notifyTranslationChanged` method to `DrawingComponent` and call it from `mouseDragged`
  - Implement two listeners on the `Drawing` class (one for each window), add `addDrawingListener` method to `DrawingFrame`, and add a listener to each frame in `Drawing`
  - Add `setTranslation` methods to `DrawingComponent` and `DrawingFrame`
  - Double Sponge Bob (Scaling & Translation & Synch)

Image Panel

- How can we apply these ideas to implement the Image Panel in the Record Indexer project?

Image Panel

- How can we apply these ideas to implement the Image Panel in the Record Indexer project?
  - Instead of scaling relative to its top-left corner like DoubleSB, the Image Panel scales relative to its center point (panel.width / 2, panel.height / 2)
  - Therefore, the transform required to convert points from world to device coordinates is a little different than in DoubleSB
    - translate(this.getWidth() / 2.0, this.getHeight() / 2.0);
    - scale(scale, scale);
    - translate(-image.getWidth() / 2.0 + w_translateX,
      -image.getHeight() / 2.0 + w_translateY);

Image Navigator

- How can we apply these ideas to keep the Image Panel and Image Navigator in synch in the Record Indexer project?

Image Navigator

- How can we apply these same ideas to keep the Image Panel and Image Navigator in synch in the Record Indexer project?
  - The Image Panel has its own scale and translation settings
  - The Image Navigator has its own scale setting (based on its current size)
  - Conversions between device and world coordinates are similar to the "Double Sponge Bob" example
  - Zooming in Image Panel preserves the "center point" (i.e., the point at the center of the window should stay fixed during zooming)
  - Image Panel keeps track of how far the user has translated the image (in world coordinates)
  - The Image Navigator can query the Image Panel for the currently visible rectangle (so it can draw the navigator rectangle)
  - The Image Navigator can adjust the Image Panel's translation as the navigator rectangle is dragged