Documentation

The Java™ Tutorials

Trail: 2D Graphics
Lesson: Working with Images

Drawing an Image

As you have already learned, the `Graphics.drawImage` method draws an image at a specific location:

```
boolean Graphics.drawImage(Image img,
                   int x, int y,
                   ImageObserver observer);
```

The `x,y` location specifies the position for the top-left of the image. The `observer` parameter notifies the application of updates to an image that is loaded asynchronously. The `observer` parameter is not frequently used directly and is not needed for the `BufferedImage` class, so it usually is null.

The described method addresses only the case where the entire image is to be drawn, mapping image pixels to user space coordinates 1:1. Sometimes applications require to draw a part of the image (a sub-image), or scale the image to cover a particular area of the drawing surface, or transform or filter the image before drawing.

The overloads of the `drawImage()` method perform these operations. For example, the following overload of the `drawImage()` method enables you to draw as much of a specified area of the specified image as is currently available, scaling it to fit inside the specified area of the destination drawable surface:

```
boolean Graphics.drawImage(Image img,
       int dstx1, int dsty1, int dstx2, int dsty2,
       int srcx1, int srcy1, int srcx2, int srcy2,
       ImageObserver observer);
```

The `src` parameters represent the area of the image to copy and draw. The `dst` parameters display the area of the destination to cover by the the source area. The `dstx1, dsty1` coordinates define the location to draw the image. The width and height dimensions on the destination area are calculated by the following expressions: `(dstx2-dstx1), (dsty2-dsty1)` . If the dimensions of the source and destinations areas are different, the Java 2D API will scale up or scale down, as needed.

The following code example divides an image into four quadrants and randomly draws each quadrant of the source image into a different quadrant of the destination.

Note: If you don't see the applet running, you need to install at least the Java SE Development Kit (JDK) 7 release.

The complete code for this applet is in `JumbledImageApplet.java` .

This example uses the following code to paint the jumbled `duke_skateboard.jpg` image. It iterates over the four sub-images of the source, drawing each in turn into a randomly selected destination quadrant.

```
/* divide the image 'bi' into four rectangular
 * areas and draw each of these areas in to a
 * different part of the image, so as to jumble
 * up the image.  'cells' is an array which has
 * been populated with values which redirect
 * drawing of one subarea to another subarea.
 */
int cellWidth = bi.getWidth(null)/2;
int cellHeight = bi.getHeight(null)/2;
for (int x=0; x<2; x++) {
    int sx = x*cellWidth;
    for (int y=0; y<2; y++) {
        int sy = y*cellHeight;
        int cell = cells[x*2+y];
        int dx = (cell / 2) * cellWidth;
        int dy = (cell % 2) * cellHeight;
        g.drawImage(bi,
                    dx, dy,
                    x+cellWidth, dy+cellHeight,
                    sx, sy,
                    sx+cellWidth, sy+cellHeight,
```

```
                        null);
        }
    }
```

Filtering Images

In addition to copying and scaling images, the Java 2D API also filter an image. *Filtering* is drawing or producing a new image by applying an algorithm to the pixels of the source image. Image filters can be applied by using the following method:

```
void Graphics2D.drawImage(BufferedImage img,
                          BufferedImageOp op,
                          int x, int y)
```

The `BufferedImageOp` parameter implements the filter. The following applet represents an image drawn on top of text. Drag the slider to show more or less of the text through the image and make the image more or less transparent.

Note: If you don't see the applet running, you need to install at least the Java SE Development Kit (JDK) 7 release.

The following code shows how the filter action is done by operating on a `BufferedImage` object with an *alpha* channel and rescales that alpha channel by using the `RescaleOp` object. The alpha channel determines the translucency of each pixel. It also specifies the degree to which this image overwrites.

```
/* Create an ARGB BufferedImage */
BufferedImage img = ImageIO.read(imageSrc);
int w = img.getWidth(null);
int h = img.getHeight(null);
BufferedImage bi = new
    BufferedImage(w, h, BufferedImage.TYPE_INT_ARGB);
Graphics g = bi.getGraphics();
g.drawImage(img, 0, 0, null);

/*
 * Create a rescale filter op that makes the image
 * 50% opaque.
 */
float[] scales = { 1f, 1f, 1f, 0.5f };
float[] offsets = new float[4];
RescaleOp rop = new RescaleOp(scales, offsets, null);

/* Draw the image, applying the filter */
g2d.drawImage(bi, rop, 0, 0);
```

The complete example represented in `SeeThroughImageApplet.java` includes the code that uses the slider to adjust the transparency from the initial 50%. This example also requires the duke_skateboard.jpg image.

The `RescaleOp` object is just one of many filters that can be created. The Java 2D API has several built in filters including the following:

- `ConvolveOp`. Each output pixel is computed from surrounding pixels in the source image. It may be used to blur or sharpen images.
- `AffineTransformOp`. This filter maps pixels in the source to a different position in the destination by applying a transformation on the pixel location.
- `LookupOp`. This filter uses an application supplied lookup table to remap pixel colors.
- `RescaleOp`. This filter multiplies the colors by some factor. Can be used to lighten or darken the image, to increase or reduce its opacity, etc.

The following example uses each of the described filters as well as scaling:

Note: If you don't see the applet running, you need to install at least the Java SE Development Kit (JDK) 7 release.

The complete code for this applet is in `ImageDrawingApplet.java` and this applet requires the bld.jpg image.

Use the drop-down menu to select an image scaling or filtering operation.

Previous page: Reading/Loading an Image
Next page: Creating and Drawing to an Image