Introduction to Java Graphics

October 5, 2001

Java Graphics

- Two main Graphics libraries:
  - java.awt (Abstract Window Toolkit)
    - older
    - major versions:
      - 1.0: Old original, accepted by most web browsers
      - 1.1: Accepted by Internet Explorer, but not yet entirely by Netscape
  - java.swing
    - newer
    - more portable (less sensitive to window system)
    - might not be supported on your browser, without plugins

We will use 1.0 awt

- relatively simple to use
- works on most browsers
- will generate a “deprecation” warning when compiling with Java 1.2:
  Usually this can be ignored.
- 1.1 graphics avoids the warnings, but will not run on all browsers

Step 1

- To do graphics we need somewhere to put the output
  - To create a new window, could create a specialized instance of class java.awt.Frame.
    - i.e., we must define a subclass of Frame with methods that display what we want to see.
  - To run inside a web browser, our code will be a specialized instance of class java.applet.Applet.
    - i.e., we must define a subclass of Applet with methods that display what we want to see.
Step 2

• We then need the "canvas" to draw on
  – This is represented in Java by an object of class java.awt.Graphics.
    • We create our drawing by invoking methods of the Graphics object.
  – Frame and Applet have a getGraphics() method that can be used to get the corresponding Graphics object.
    • Graphics objects can be connected to more than just displayed items: offscreen image buffers, printers, etc.

MyApplet1

```java
import java.applet.*;
import java.awt.*;

public class MyApplet1 extends Applet {

  // setup frame with title and position
  void setup(String title, int x, int y)
  {
    setBackground(Color.white);  // set the bg color
    reshape(x, y, 500, 400);  // set location and size
    setVisible(true);  // show the frame
  }

  public void init()
  {
    setup("My Frame", 50, 50);  // Do all the work in another method
  }
}
```

Desired Result

![Desired Result Image]
public void paint(Graphics g)
{
    drawStuff(g); // Do all the work in another method
}

void drawStuff(Graphics g)
{
    g.setColor(Color.black); // set drawing color
    g.drawRect(50, 50, 400, 300); // draw a rectangle
    g.fillRect(100, 100, 300, 200); // fill an oval
    g.setFont(new Font("Times", Font.BOLD, 20));
    g.setColor(Color.yellow);
    g.drawString("Harvey Mudd College", 120, 210);
}

Typical awt Drawing Methods

- void drawLine(int x1, int y1, int x2, int y2)
- void drawRect(int x, int y, int width, int height)
- void drawOval(int x, int y, int width, int height)
- void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
- void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
- void drawString(String str, int x, int y)
- void drawImage(Image img, int x, int y,
    Color bgcolor, ImageObserver observer)

(Blue bullets also have fill instead of draw.)

http://java.sun.com/j2se/1.3/docs/api/java/awt/Graphics.html
**Paint Method**

- If an event occurs, such as setting Frame visible, or repaint() called,
  - The update() method is called on Graphics of Frame or Applet, resulting in:
    1. Drawing area cleared with background color
    2. paint() is called

- Whatever is drawn by paint is displayed

**Flicker Prevention 1**

- Clearing the background in update() can create lots of flicker in the application.
- It is customary to override update() in the customized frame:
  ```java
  public void update(Graphics g) {
    paint(g);
  }
  ```
- Object-oriented languages specialize in this kind of overriding.

**Flicker Prevention 2**

- Instead of painting the background, then drawing on it, it is better to paint the complete image and then display it only when done.
- This image is known as an off-screen buffer.
- The buffer is drawn on prior to update().

**Applet without Flicker**

```java
public class MyApplet2 extends Applet {
  public void init() {
    buffer = createImage(getWidth(), getHeight());
    graphics = buffer.getGraphics();
    drawStuff(graphics); // initialize the buffer!
  }
  ...
  public void update(Graphics g) {
    paint(g);
  }
}
```
public void paint(Graphics g) {
    g.drawImage(buffer, 0, 0, null);
} // ...other code as before

Alternate Organization

Image buffer;
public void paint(Graphics g) {
    // Create buffer on demand
    if( buffer == null )
        buffer = createImage(getWidth(), getHeight());
    // overwrite existing buffer contents
    drawStuff(buffer.getGraphics());
    // write the buffer to the screen
    g.drawImage(buffer, 0, 0, null);
}

Flicker Prevention Summary

some event occurs, such as setting Frame visible, or repaint()

update() is called on Graphics of Frame, resulting in:
paint() is called on Graphics of Frame

drawing is on buffer

buffer is painted on Graphics of Frame

repaint()

• When the programmer wants to force repainting he/she calls
  repaint()
• This causes the system to schedule a call to update().
• repaint() has no arguments.
• The programmer normally does not call paint() directly, outside of update().
Applets

- “Applet” means “small application”
- Compiled with javac, but cannot run with java
  - No main(); instead: init(), run().
- Run in one of two ways:
  - In a web browser
  - Using a program appletviewer

Viewing Applets on the Web or using appletviewer

contents of MyApplet1.html:

```html
<html>
<title>MyApplet1</title>
<head>
<!-- MyApplet1 applet-->
</head>
<body>

<applet code=MyApplet1.class width=500 height=400></applet>

</body>
</html>
```

run by:

appletviewer MyApplet1.html

Web Applet Restrictions

- Can't load files on server or client
- Can only load content of URL's, and then only from the same server that contains the applet code

Examples

- appletGraphics examples on our web pages
  - and others
- Look for "Examples" on the main course page