极客营 iGeekHome
iGeekHome
400-133-0510
www.igeekhome.com
© 极客营 版权所有 违者必究

第十一章 高级AWT

目标

- 绘图操作流程
- 形状\区域笔划
- 透明与组合
- 图像读取器和写入器
- 图像处理
- 剪贴板
- 拖放操作

绘图的基本操作

- Component类具有图形表示能力，可在屏幕上显示，并可与用户进行交互。Swing开发中我们的窗体控件大都扩展了该类。

- Component类提供了几个与绘图相关的方法：
  - **paint(Graphics g)** 绘制此组件。
  - **paintAll(Graphics g)** 绘制此组件及其所有子组件。
  - **update(Graphics g)** 更新组件。
  - **repaint()** 重绘此组件。

- 当控件加载的时候会调用paint()方法绘制控件，如果我们在程序改变了控件的大小或者颜色等等操作，可以调用repaint ()方法重绘组件，**repaint** ()方法会调用update ()方法，然后update ()方法会先清空组件，在调用paint()方法绘制组件。

Graphics类

- Graphics 对象封装了Java 支持的基本呈现操作所需的状态信息。
- 状态信息包括以下属性：
  - 要在其上绘制的 Component 对象。
  - 呈现和剪贴坐标的转换原点。
  - 当前剪贴区。
  - 当前颜色。
  - 当前字体。

Graphics类的绘制方法

- 绘制边框

draw3DRect(…)	绘制指定矩形的 3-D 高亮显示边框。
drawArc (…)	绘制一个覆盖指定矩形的圆弧或椭圆弧边框。
drawBytes (…)	绘制由指定 byte 数组给定的文本。
drawChar (…)	绘制由指定字符数组给定的文本。
drawImage (…)	绘制指定图像
drawLine (…)	画一条线。
drawOval (…)	绘制椭圆的边框。
drawPolygon (…)	绘制一个多边形。
drawPolyline (…)	绘制由 x 和 y 坐标数组定义的一系列连接线。
drawRec (…)	绘制指定矩形的边框。
drawRoundRect	绘制圆角矩形的边框。
drawString (…)	绘制由指定 string 给定的文本。

- 绘制填充的图形

fill3DRect (…)	绘制一个用当前颜色填充的 3-D 高亮显示矩形。
fillArc (…)	填充覆盖指定矩形的圆弧或椭圆弧。
fillOval (…)	使用当前颜色填充外接指定矩形框的椭圆。
fillPolygon (…)	填充由 x 和 y 坐标数组定义的闭合多边形。
fillPolygon (…)	填充多边形。

绘图示例

- 我们可以在窗体程序中重写paint()方法，使用Graphics类来绘制我们的图形

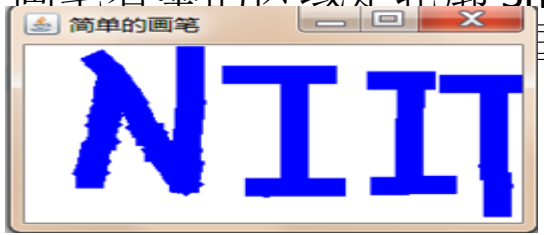
```
import java.awt.Graphics;

public class GraphicsDemo1  extends  JFrame{
      //.....

     /**
     * 重写了paint方法，利用Graphics类来绘制图形
     */
     @Override
     public void paint(Graphics g) {
     //在坐标x=30,y=50的位置开始 绘制一个 宽为100，高为100的矩形框
     g.drawRect(30, 50, 100, 100);
     }
}
```

画笔

- Graphics类是一个抽象类，我们再重新paint()方法的时候，该方法接受一个Graphics类的子类Graphics2D的实例，所以我们可以转换成Graphics2D。

- Stroke 接口允许 Graphics2D 对象获得一个 Shape，指定 Shape 的装饰轮廓，或该轮廓的风格表示形式。

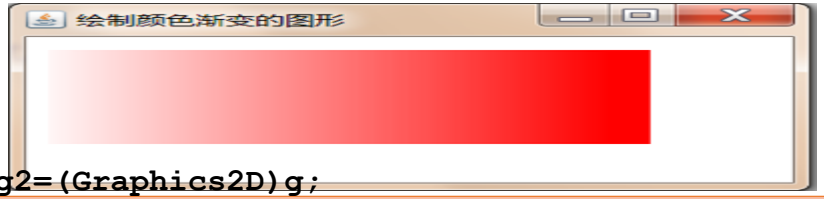
- 画笔着墨的区域是轮廓 Shape 封闭的区域，我们常用的Stroke实现类




```
Graphics2D  g2=(Graphics2D)g;
//设置基础画笔大小
g2.setStroke(new BasicStroke(15.5F));
//设置颜色
g2.setColor(Color.BLUE);
```

GradientPaint绘制颜色渐变的图形

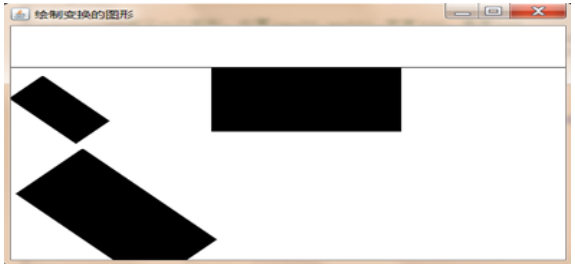
- Paint接口定义如何为 Graphics2D 操作生成颜色模式， GradientPaint 类实现了Paint接口，它提供了使用线性颜色渐变模式填充 Shape的方法。




```
Graphics2D  g2=(Graphics2D)g;
/*
* 创建一个Paint
* GradientPaint 类提供了使用线性颜色渐变模式填充 Shape 的方法。
*/
Paint  paint=new GradientPaint(10,200,Color.WHITE,300,200,Color.RED);
g2.setPaint(paint); //设置着色方式
//创建一个矩形，位置(x=20,y=50),宽度300，高度100
Rectangle2D rec=new Rectangle2D.Double(20, 50, 300, 100);
//使用填充的方式绘制
g2.fill(rec);
```

绘制旋转和缩放的图形

- java.awt.geom.AffineTransform 类表示 2D 仿射变换，它执行从 2D 坐标到其他 2D 坐标的线性映射，保留了线的“直线性”和“平行性”。
- 可以使用一系列平移 (translation)、缩放 (scale)、翻转 (flip)、旋转 (rotation) 和错切 (shear) 来构造仿射变换。




```
//缩放
g2.transform(AffineTransform.getScaleInstance(
0.5D, 0.5D));
 //旋转45度
g2.rotate(45*Math.PI/180);
//缩放50%
g2.scale(0.5, 0.5);
```

绘制透明图形

- Color 类用于封装默认标准RGB 颜色空间中的颜色
- alpha 值为 1.0 或 255 则意味着颜色完全是不透明的alpha 值为 0 或 0.0 则意味着颜色是完全透明的
- 我们可以通过使用定义Color的透明度，然后绘画笔设置颜色的方式

```
//定义一个绿色的 透明度为 0.5的颜色对象
Color  color=new Color(0,0.5f,0,.5f);
//设置画笔颜色
g2.setColor(color);
```

图像的读取和写入

- ImageIO类包含一些用来查找 ImageReader 和 ImageWriter 以及执行简单编码和解码的静态便捷方法。

- ImageIO类读取出来原始图像BufferedImage，然后通过BufferedImage对象得到Graphics2D绘图对象

```
BufferedImage bi=ImageIO.read(new File("img/java.png"));
ImageIO.write(bi,"jpeg", new File("img/java2.jpg"));
```

要想装载一个圈像，可以使用ImageI0类的静态read方法:
想要写入一个图像我们可以使用ImageIO类的静态write方法:

```
Image img=ImageIO.read(new File("img/java.png"));
```

剪贴板的使用

- 我们在平时操作的时候用到比较多的操作复制、剪切、粘贴等操作，其实都用到了一个基本的东西就是剪贴板。

- Java中AWT提供了两种类型的剪贴板：
  - 系统剪贴板和本地剪贴板。

- 与剪贴板相关的类都在java.awt.datatransfer包中。**Clipboard**类实现一种使用剪切/复制/粘贴操作传输数据的机制，代表了一个剪贴板的实例

总结

- Component类具有图形表示能力，可在屏幕上显示，并可与用户进行交互。Swing开发中我们的窗体控件大都扩展了该类
- 我们可以在窗体程序中重写paint()方法，使用Graphics类来绘制我们的图形
- java.awt.geom.AffineTransform 类表示 2D 仿射变换

400-133-0510

www.igeekhome.com

© 极客营 版权所有 违者必究