

1.00 第21讲

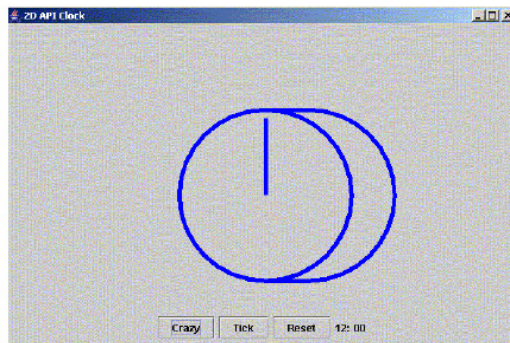
2D API

2D 转换

下次阅读：数值方法（Numerical Recipes）p32-35只阅读文字
不要担心读到C代码

Clock

- 使用model-view-controller版本的Clock，通过2D API（应用程序接口）绘图。



- 下载ClockController, ClockModel, ClockView

具有2D API的ClockView

```
import java.awt.*;
import javax.swing.*;
import java.awt.geom.*;

public class ClockView extends JPanel {
    private ClockModel model;
    private static final double CD= 200;           // Clock 直径
    private static final double X= 100;            // 到上部1h角的距离
    private static final double Y= 50;            //到上部1h角的距离
    private static final double XC= X + CD/2;      // Clock 中心 x
    private static final double YC= Y + CD/2;      // Clock 中心y
    private static final double HR= 0.3F*CD;       // 时针大小
    private static final double MI= 0.45F*CD;      //分针大小
    public ClockView(ClockModel cm) {
        model = cm;
    }
    // 待续
```

具有2D API的ClockView, p.2

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;               // 把g赋予 g2
    double minutes= model.getMinutes();
    double hourAngle = 2*Math.PI * (minutes - 3 * 60) / (12 * 60);
    double minuteAngle = 2*Math.PI * (minutes - 15) / 60;
    Ellipse2D.Double e = new Ellipse2D.Double(X, Y, CD, CD);
    Line2D hr= new Line2D.Double(XC, YC, XC+(HR*Math.cos(hourAngle)),
        YC+ (HR * Math.sin(hourAngle)) );
    Line2D mi= new Line2D.Double(XC, YC, XC+
        (MI* Math.cos(minuteAngle)), YC+ (MI * Math.sin(minuteAngle)) );
    g2.setPaint(Color.BLUE);
    BasicStroke bs= new BasicStroke(5.0F,
    BasicStroke.CAP_BUTT, BasicStroke.JOIN_BEVEL);
    g2.setStroke(bs);
    g2.draw(e);
    g2.draw(hr);
    g2.draw(mi);
}
}
```

练习

- 添加两条直线弧线到`paintComponent()`，创建如第一张幻灯片那样的图案
 - `Line2D.Double(double x0, double y0, double x1, double y1)`
 - 从 (x_0, y_0) 到 (x_1, y_1) 画一条直线
 - 使得直线长度= 时钟直径的四分之一
 - `Arc2D.Double(double x, double y, double w, double h, double start, double extent, int type)`
 - 从左上角 (x,y) 起画一条弧线，宽为 w 高为 h 。这4个参数同椭圆和圆的参数是一样的。
 - `Start`为初始角度，单位为度
 - `extent`为弧线的角度，单位为度
 - `type`是一种类型；使用`Arc2D.OPEN`
- 任选：不同颜色，不用宽度画出时针和分针。

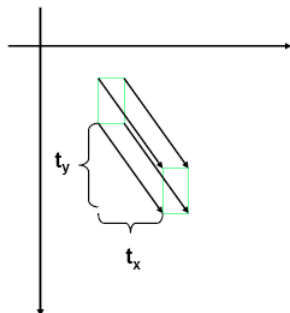
Affine转换

- 2D API支持`affine 转换`
 - Affine即为线性的（形如 $y = ax + b$ ）
- Affine转换对应2D坐标系，因此直线的笔直性和平行性得到保存。
- 所有的Affine转换都可以由一个3x3浮点矩阵来代替。
- 有一些“原始的”Affine转换，它们可以互相组合：缩放，旋转和平移。

2D API中的转换

- 转换由java.awt.geom包中的AffineTransform类实例替代。
- 可以通过无参数构造函数来创建一个新的AffineTransform对象。
 - `AffineTransform at = new AffineTransform();`
- 可在AffineTransform对象中调用下面这些方法（当然还有其他的）：
 - `at.scale(double sx, double sy)`
 - `at.translate(double tx, double ty)`
 - `at.rotate(double theta)`
 - `at.rotate(double theta, double x, double y)`
- 这些方法构成了基本转换栈：后进，先出

平移(Translation)



$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x+t_x \\ y+t_y \\ 1 \end{bmatrix}$$

Translation实例

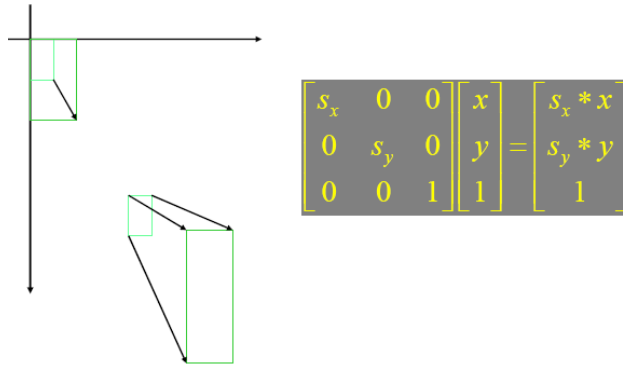
在一个JFrame里显示一个正方形面板

```
import java.awt.*;
import javax.swing.*;
public class RectangleTest {
    public static void main(String args[]) {
        JFrame frame = new JFrame("Rectangle transform");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500,500);
        Container contentPane= frame.getContentPane();
        RectanglePanel panel = new RectanglePanel();
        contentPane.add(panel);
        frame.setVisible(true);
    }
}
```

Translation实例

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*; // 对于2D类
public class RectanglePanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2= (Graphics2D) g;
        Rectangle2D rect= new Rectangle2D.Double(0,0,50,100);
        g2.setPaint(Color.BLUE);
        AffineTransform baseXf= new AffineTransform();
        // Shift to the right 50 pixels, down 50 pixels
        baseXf.translate(50,50);
        g2.transform(baseXf);
        g2.draw(rect);
    }
} // 下载并运行RectangleTest, RectanglePanel
```

缩放(Scaling)



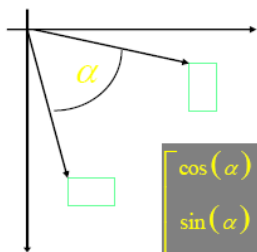
Scaling实例

- 修改RectangleTest, RectanglePanel:
- 首先, 使用RectanglePanel作为基础编写代码缩放位于原点的rect
 - 以translation实例中一样的步骤来进行。
 - 以scale方法来取代translate方法。
 - scale有两个double型参数: 一个是x坐标的缩放, 一个是y坐标的缩放。
- 下一步, 修改rect使得不在原点处。scale对不在原点的形状会产生什么样的作用?
 - 修改刚刚的两个参数, 改变矩形左上角的(x,y)的值。

Scaling说明

- 基本的缩放操作着眼于原点。若形状位于原点，则产生缩放。若位于其他地方，产生缩放的同时也产生了移动。
- s_x , x方向的缩放不需要同 s_y , y方向的缩放相一致。
- 例如，关于x轴垂直翻转，则 $s_x=1$, $s_y=-1$

旋转(Rotation)


$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cos(\alpha) - y \sin(\alpha) \\ x \sin(\alpha) + y \cos(\alpha) \\ 1 \end{bmatrix}$$

Rotation实例

- 再一次修改RectangleTest, RectanglePanel:
- 使用RectanglePanel作为基础编写代码旋转rect。
- 遵从缩放练习中相同的步骤。
 - 调用`basexf.rotate()`，取一个参数：角度，单位为弧度。
 - 相对于原点这个方法将不仅旋转而且移动矩形。
 - 你将会发现`Math.PI`或者`Math.toRadians(double degrees)`很有用
- 为了避免把rect旋转出视野，取一个小点的量旋转（10或者20度）
- 当rect位于原点的时候旋转rect是如何改变的？什么时候不是呢？

组合转换

假设我们需要2倍的缩放(x,y)，然后旋转90度

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \right)$$

rotate

scale

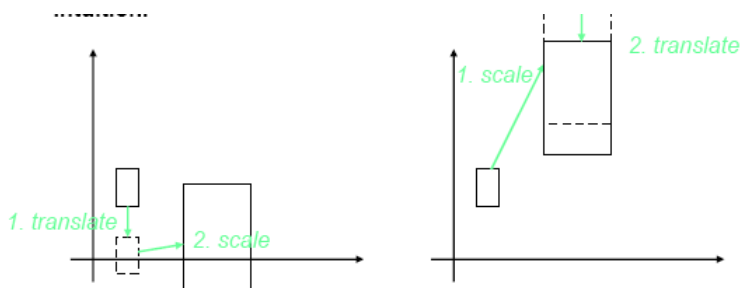
组合转换, 2

因为矩阵的乘法是结合的，所以我们可以这样来写：

$$\begin{pmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{pmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ = \begin{bmatrix} 0 & -2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

组合转换, 3

- 因为矩阵的乘法不能简单的交换，所以转换的顺序便很重要。这同我们的几何直觉相一致。

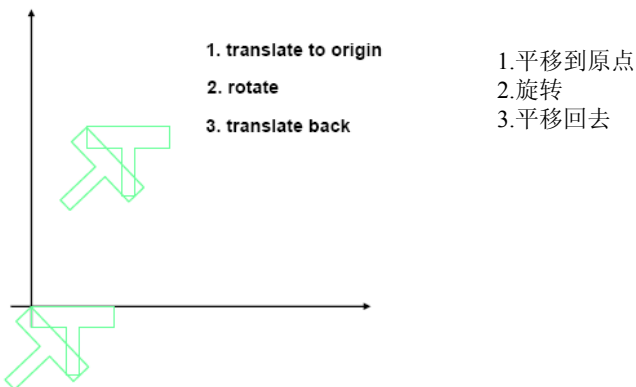


- 如果颠倒了矩阵次序，转换顺序也就相反了。

转换和原点

- 当我们转换一个形状时，转换了该形状的每一个定义点，然后重新把它们画出来。
- 若缩放或旋转一个不在原点的形状，同时也发生了平移。
- 若只需要缩放或旋转，我们需要先平移到原点，缩放或旋转，然后再平移回来。

转换和原点, 2



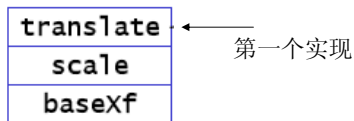
2D API中的转换

- 转换由java.awt.geom包中的AffineTransform类实例替代。
- 通过下面方法来进行混合的转换
 1. 创建一个新的AffineTransform实例
 2. 调用方法来形成一些基本的转换：后进，先实现
 - translate(double tx, double ty)
 - scale(double sx, double sy)
 - rotate(double theta)
 - rotate(double theta, double x, double y)
关于(x,y)旋转

转换实例

```
baseXf = new AffineTransform();  
baseXf.scale( scale, -scale );  
baseXf.translate( -x, -y );
```

若应用basicXF,将先进行平移，然后缩放。
记得在Java中转换就像栈一样，后进，先出



(TransformTest 和 TransformPanel 作为例子给出)

转化Swing GUI为Applet

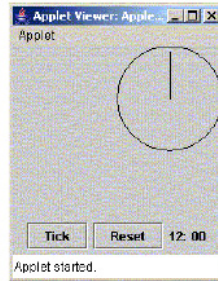
- 用适当的代码创建一个HTML页面，载入程序片（37讲介绍）
- 声明程序片名，扩展JApplet
- 不予考虑main()方法：
- 除去以下调用：
 - setSize();在HTML文件中完成
 - setDefaultCloseOperation();浏览结束程序片即结束
 - setTitle();不允许标题
 - setVisible();由浏览器完成
- 不用构造一个JFrame（不予考虑它的构造函数）
 - 程序片使用浏览器来替代
- 把main()或JFrame构造函数中剩余的代码移动到程序片的init()方法中：
 - 通常在原始的main()中没有语句了
 - 通常JFrame构造函数按照原状转移

Clock Applet

```
import javax.swing.*;
import java.awt.*;
public class AppletTest extends JApplet{
    public void init() {
        Container contentPane= getContentPane();
        ClockPanel clock= new ClockPanel();
        contentPane.add(clock);
    }
}
```

```
//事件那讲的ClockPanel类按照原状
//不需要ClockFrame类
//不予考虑main()，构造函数中的语句
//把剩余语句移到init()中
```

Clock Applet练习



下载ClockPanel
编写AppletTest（与原main(), 构造函数比较）
在Eclipse中Run -> Run As Applet

检验Swing知识

- 一个类本身能成为其自己的event handler吗?
 - Yes
 - No
- 一个类要监听事件需要执行什么接口?
 - _____
- 若是这样, 该接口有什么方法?
 - _____
- 如何绘出JButton具有圆形的边缘?
 - _____

检验Swing知识, p.2

- 为什么不直接在JFrame绘出JComponent?
— _____
- 我们在什么对象上绘出JComponent?
— _____
- 需要把System.exit(0)放到创建Swing对象的方法结尾吗?
— Yes No 为什么? _____
- repaint()掉用什么方法?
— _____
- 为什么不直接调用该方法?
— _____

检验Swing知识, p.2

- 一个匿名内部类构造函数可拥有参数
— 对 错
- 可用 'inner' 关键词来代表匿名内部类
— 对 错
- 一个匿名内部类只能有执行Listener接口的方法
— 对 错
- 匿名内部类的名称为 'this'
— 对 错
- 匿名内部类有权访问自身封装的类数据和方法
— 对 错