> 由 KiOii (*_EM_Cpper*)整理。 (KiOii (*_EM_Cpper*) makes this note.)

Playing with Colors

In this chapter

- 颜色在JavaFX中是如何表示的
- 不同的颜色模式（color patterns）
- 如何使用 图形模式（image pattern）
- 如何使用线性颜色渐变（linear color gradient）
- 如何使用径向颜色渐变 （radial color gradient）

1.Understanding Colors

> 在JavaFX中，您可以为区域(region)的文本和背景颜色指定颜色

- color : 您可以将颜色指定为统一颜色(uniform color)，图像模式或颜色渐变
  - **uniform color** : use the same color to fill the entire region
  - **image pattern** : fill a region with an **image pattern**
  - **color gradient** : 颜色渐变定义了颜色模式 (**color pattern**)，其中颜色沿着从一种颜色到另一种颜色的直线变化
    - linear or radial (线性 or 径向)
- **javafx.scene.paint** package

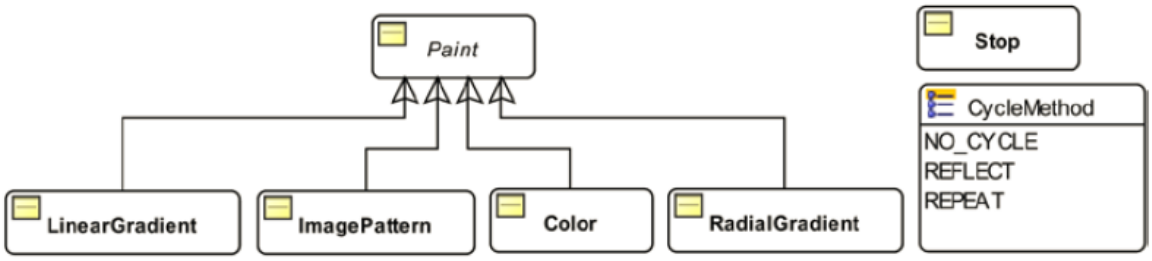

Figure 7-1. *The class diagram of color-related classes in JavaFX*

- Paint class : abstract class
  - **public static Paint valueOf(String value)** : 返回子类
    - 参数：转换为从css读取的颜色值

```
// redColor is an instance of the Color class
Paint redColor = Paint.valueOf("red");
// aLinearGradientColor is an instance of the LinearGradient class
Paint aLinearGradientColor = Paint.valueOf("linear-gradient(to bottom right, red,
black)" );
// aRadialGradientColor is an instance of the RadialGradient class
Paint aRadialGradientColor =
Paint.valueOf("radial-gradient(radius 100%, red, blue, black)");
```

> The Stop class and the CycleMethod enum are used while working with color gradients.

1.1 Using the Color Class

- **Color** class : a **solid uniform color** from the **RGB** color space
  - alpha value :
    - 0.0 ~ 1.0
    - 0 ~ 255
    - 0.0 或者 0： 意味着完全透明
    - 1.0 或者 255： 意味着完全不透明 （default 1.0）
  - 获取 实例
    - Using the **constructor**
      - 唯一:
      - `public Color(double red,double green,double blue,double opacity);`
      - exmaple

```
Color blue = new Color(0.0,0.0,1.0,1.0);
```

    - Using one of the **factory methods**
      - `Color color(double red, double green, double blue)`

        - `Color color(double red, double green, double blue, double opacity)`
        - `Color hsb(double hue, double saturation, double brightness)`
        - `Color hsb(double hue, double saturation, double brightness, double opacity)`
        - `Color rgb(int red, int green, int blue)`
        - `Color rgb(int red, int green, int blue, double opacity)`
        - **valueof** and **web** : 可以从web color value formats创建 Color对象
            - example

```
Color blue = Color.valueOf("blue");
Color blue = Color.web("blue");
Color blue = Color.web("#0000FF");
Color blue = Color.web("0X0000FF");
Color blue = Color.web("rgb(0, 0, 255)");
Color blue = Color.web("rgba(0, 0, 255, 0.5)"); // 50% transparent blue
```

    - Using one of the **color constants** declared in the Color class
        - RED
        - WHITE
        - TAN
        - BLUE
        - ...

1.2 Using the ImagePattern Class

- **image pattern** : fill a shape with an image
    - 图像可能会填满整个shape或使用平铺图案
    - 步骤:
        1. Create a `Image` object using an image form a file
        2. Define a rectangel

            > 定义一个矩形，称为锚矩形(anchor rectangle)，相对于左上角 要填充的形状的角落

    - **anchor rectangle** : image在anchor rectangle中显示，然后调整适应该anchor rectangle
    - 如果要填充的形状的边界框大于锚定矩形的边界框，则以平铺图案在形状内重复带有图像的锚定矩形（shape包裹 anchor rectangle，anchor rectangle对应一个image)
    - 获取**ImagePattern**实例
        - 构造函数
            - `ImagePattern (Image image)`
            - `ImagePattern(Image image,double x,double y,double width, double height, boolean proportional)`
                - **proportional** 参数： true代表指定相对bounding box的位置和大小

                    > flase 则不过大时默认平铺

```
//这两个效果一样
ImagePatterm ip1 = new ImagePattern(anImage);
ImagePatterm ip2 = new ImagePattern(anImage, 0.0, 0.0, 1.0, 1.0, true);
```

- 使用例子

> 根据 Image 创建 ImagePattern

```
Image img = create the image object...
ImagePattern p1 = new ImagePattern(img, 0, 0, 0.25, 0.25, true);
```

  - 使用 **ImagePattern**

```
Rectangle r1 = new Rectangle(100, 50);
r1.setFill(p1);
```

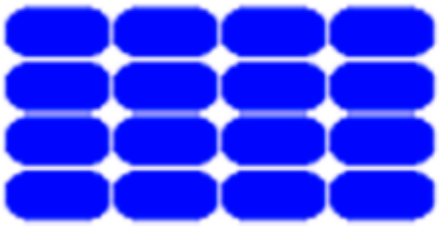

Figure 7-3. *Filling a rectangle with an image pattern*

```
Polygon triangle = new Polygon(50, 0, 0, 50, 100, 50);
triangle.setFill(p1);
```

Figure 7-4. *Filling a triangle with an image pattern*

  - 不使用平铺（tiling pattern）

```
// An image pattern to completely fill a shape with the image
ImagePatterm ip = new ImagePattern(yourImage, 0.0, 0.0, 1.0, 1.0, true);
```

  - 完整例子代码

```
import java.net.URL;
import javafx.application.Application;

import javafx.application.Platform;
```

```
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.HBox;
import javafx.scene.paint.ImagePattern;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;
public class ImagePatternApp extends Application
{
    private Image img;
    public static void main(String[] args)
    {
        Application.launch(args);
    }
    @Override
    public void init()
    {
        // Create an Image object
        final String IMAGE_PATH = "resources/picture/blue_rounded_rectangle.png";
        URL url = this.getClass().getClassLoader().getResource(IMAGE_PATH);
        if (url == null)
        {
            System.out.println(IMAGE_PATH + " file not found in CLASSPATH");
            Platform.exit();
            return;
        }
        img = new Image(url.toExternalForm());
    }
    @Override
    public void start(Stage stage)
    {
        // An anchor rectangle at (0, 0) that is 25% wide and 25% tall
        // relative to the rectangle to be filled
        ImagePattern p1 = new ImagePattern(img, 0, 0, 0.25, 0.25, true);
        Rectangle r1 = new Rectangle(100, 50);
        r1.setFill(p1);
        // An anchor rectangle at (0, 0) that is 50% wide and 50% tall
        // relative to the rectangle to be filled
        ImagePattern p2 = new ImagePattern(img, 0, 0, 0.5, 0.5, true);
        Rectangle r2 = new Rectangle(100, 50);
        r2.setFill(p2);
        // Using absolute bounds for the anchor rectangle
        ImagePattern p3 = new ImagePattern(img, 40, 15, 20, 20, false);
        Rectangle r3 = new Rectangle(100, 50);
        r3.setFill(p3);
        // Fill a circle
        ImagePattern p4 = new ImagePattern(img, 0, 0, 0.1, 0.1, true);
        Circle c = new Circle(50, 50, 25);
        c.setFill(p4);
        HBox root = new HBox();
        root.getChildren().addAll(r1, r2, r3, c);
        Scene scene = new Scene(root);

        stage.setScene(scene);
```

```
        stage.setTitle("Using Image Patterns");
        stage.show();
    }
}
```

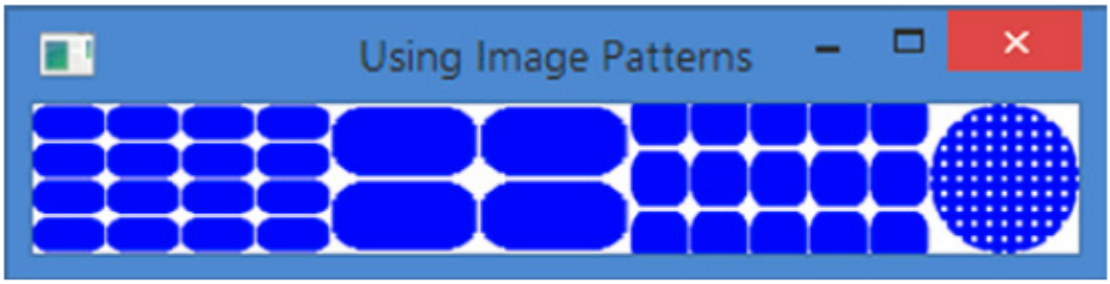

Figure 7-5. *Filling different shapes with image patterns*

2. Understanding Linear Color Gradient

- **linear color gradient** : using 渐变线性轴(gradient line axis)
  - grdient line : 每个点在 这个轴上具有不同的 color

    > 垂直这条线的垂线上的所有点则具有相同的color

    - define : **starting** point and **ending** point
  - stop-color points : 在这个轴上的点

    > 两点之间的颜色通过 interpolation计算

  - 线性渐变具有 direction（方向）
    - 从 starting point到 ending point

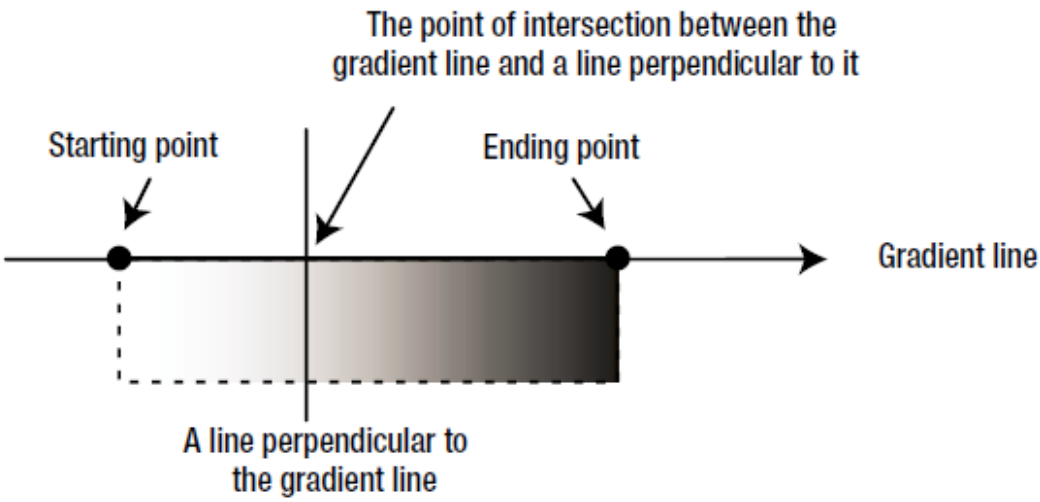

Figure 7-6. *The details of a linear color gradient*

2.1 Using the LinearGradient Class

- A instance of**LinearGradient** : 代表 a linear color gradient
    - 构造函数

```
LinearGradient(
    double startX,
    double startY,
    double endX,
    double endY,
    boolean proprotional,
    CycleMethod cycleMethod,
    List<Stop> stops
);
```

```
LinearGradient(
    double startX,
    double startY,
    double endX,
    double endY,
    boolean proportional,
    CycleMethod cycleMethod,
    Stop... stops
);
```

- **startX and startY** : 线性渐变轴起点 x和y

- **endX and endY** : 线性渐变轴的终点 x和y

- **proportional** : 如何应对 起点和终点

  - true : 相对起点和终点方框来处理 (0.0 ~ 1.0)
  - false: 相对于 Local坐标系统的数值 (例如: 0.0 ~ 200.0 等绝对数值)

- **cycleMethod** : cycleMethod参数定义应如何填充由起点和终点定义的颜色渐变边界外的区域。 假设您将比例参数设置为true的开始点和结束点分别定义为 (0.0, 0.0) 和 (0.5, 0.0) 。这只涵盖该地区的左半边。该区域的右半区应该如何填充? 您可以使用cycleMethod参数指定此行为，枚举常量:

  - `CycleMethod.NO_CYVLE` : terminal color (终点stop point颜色填充剩下的)
    - **start to end** and **end**
  - `CycleMethod.REFLECT` : **start to end** and **end to start**
  - `CycleMethod.REPEAT` : **start to end** and **start to end**

- stops : 定义了渐变线上的stop point

  - An instance of **Stop** class

  - **`Stop (double offset, Color color)`**

    - offset : 0.0 ~ 1.0 , 设置stop point相对位置

```
      > 它定义了沿着点的停止点的相对距离
      >
      > 0.0是start point
      >
      > 1.0是end point

    - color ： 设置 stop point颜色

  - 虽然没有指定可以定义的点的数量，但是至少应该是两个（颜色点）

- example

![7-6](F:\个人资料\JavaFX\7-6.png)

![7-7](F:\个人资料\JavaFX\7-7.png)

![7-8](F:\个人资料\JavaFX\7-8.png)

![7-9](F:\个人资料\JavaFX\7-9.png)

![7-10](F:\个人资料\JavaFX\7-10.png)

![7-11](F:\个人资料\JavaFX\7-11.png)

![7-12](F:\个人资料\JavaFX\7-12.png)
```

2.2Defining Linear Color Gradients Using a String Format

- **valueOf(String colorString)**
- 语法：

```
linear-gradient([gradient-line],[cycle-method],color-stops-list)
```

> [] 包裹起来

  - **gradient-line** : 默认是 "to buttom"
    - Using two point : **from** *point-1* **to** *point-2*

```
from 0% 0% to point 0% 0%   // 相对矩形 类似 proportional设置为true
or
from 0px 0px to 200px 0px   // 绝对
```

    - Using a side or s corner : **to** *side-to-corner*

> 相当于只指定了 ending point
>
> 起点被推断
>
> 例如：”to top“ 可以推断出，起点是在 “from bottom”

```
/*
*   top, left, buttom, right,
*   top left, buttom left, bottom right, top right
*/
```

- **cycle-method** : 默认是 NO_CYCLE

  > 如果缺失，则默认为NO_CYCLE。 将cycle-method参数的值指定为NO_CYCLE是一个运行时错误。 如果你想要它 NO_CYCLE，只需从语法中省略循环方法参数即可

  - 有效值是：**reflect， repeat**

- **color-stops-list** : a list of color stops

```
white,black             // 两个点
white 0%, black 100%    // 两个点
white 0%, yellow 50%,blue 100%     // 三个点
white 0px, yellow 100px, red 200px // 三个点
```

> 类似的，下面这两个是一样的

```
white 0%, yellow 25%, black 50%,red 75%,green 100%
white,yellow,black,red,green
```

> 在有位置之间均匀分布

```
white, yellow, black 60%, red, green
white 0%, yellow 30%, black 50%, red 80%, green 100
```

> 如果列表中的某个颜色停止位置的位置设置小于之前任何颜色停止位置所指定的位置，则其位置将设置为等于为前一个颜色停止位置设置的最大位置

```
white, yellow 50%, black 10%, green
white 0%, yellow 50%, black 50%, green 100%
```

- 下面两个一样

```
linear-gradient(white, black) // 默认的 NO_CYCLE
linear-gradient(to bottom, white, black)
```

- 例子：

```
String value = "from 0px 0px to 100px 0px, repeat, white 0%, black 100%";
LinearGradient lg2 = LinearGradient.valueOf(value);
Rectangle r2 = new Rectangle(200, 100);
r2.setFill(lg2);
```


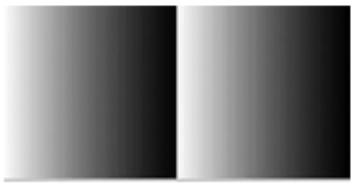

Figure 7-15. *Creating a linear color gradient using the string format*

The following string value for a linear color gradient will create a diagonal gradient from the top left corner to the bottom right corner filling the area with white and black colors:

3. Understanding Radial Color Gradient

- **radial color gradient** : 在径向颜色渐变中，颜色从单个点开始，以圆形或椭圆形向外平滑过渡

  > shape (现在讨论的是 circle): 由 center 和 radius构成
  >
  > linear gradient讨论的是矩形

  - **focus point fo the gradient** : 起点
  - 颜色从起点开始沿着一条线在所有方向上变化，直到达到形状的外围
- define： three components

  > 由三个组件来定义

  - A gradient shape (center and radius of **the gradient circle**)
  - A **focus point** that has the first color of the gradient
  - Color stops

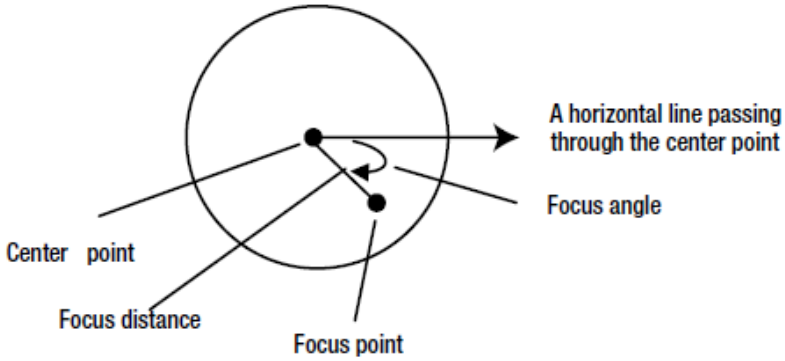

Figure 7-17. *Defining a focus point in a radial color gradient*

- > stops : 定义了渐变形状内的某处颜色值
  >
  > focus point：定义了 0% point
  >
  > 圆的边：定义了100% point
  >
  > 点处的颜色?
  >
  > 焦点和该点连线，取侧的颜色

3.1 Using the RadialGradient Class

- radial color gradient : An instance of the **RadialGradient** class
  - 构造函数

```
RadialGradient(
    double focusAngle,
    double focusDiatance,
    double centerX,
    double centerY,
    double dadius,
    boolean proportional,
    CycleMethod cycleMethod,
    List<Stop> stops
);

RadialGradient(
    double focusAngle,
    double focusDiatance,
    double centerX,
    double centerY,
    double dadius,
    boolean proportional,
    CycleMethod cycleMethod,
    Stop... stops
);
```

  - **focusAngle** : define the **focus angle** for the focus point

> 从通过中心点的水平线和连接中心点和焦点的线顺时针测量正焦距角。
>
> 负值是逆时针测量的

- **focusDistance** :根据圆的半径的百分比指定 (-1.0 ~ 1.0)

> 如果焦点距离将焦点设置在渐变圆的外围之外，则使用的焦点是圆的外围与连接中心点和设定焦点的线的交点

> The focus angle and the focus distance can have positive and negative values

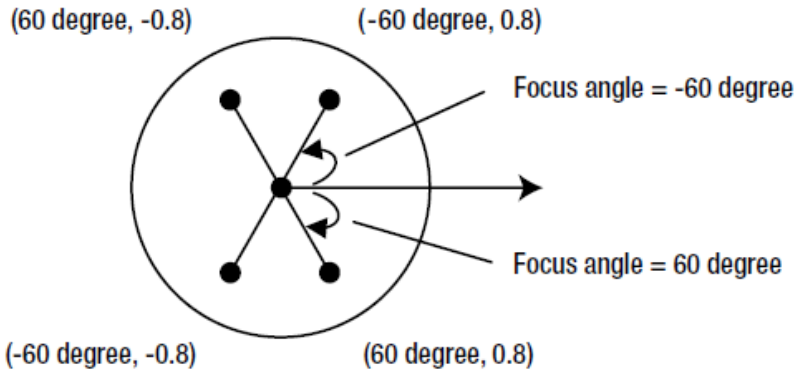

Figure 7-18. *Locating a focus point with its focus angle and focus distance*

- **centerX** and **centerY** : define the x and y of the center point
- **radius** : the radius of the gradient circle

> 可以使用绝对 的 px

- **proportional** : true, 圆心和半径都是相对整个待填充的图形
- **注意**：JavaFX让你创建一个圆形的径向渐变。 但是，当要用径向颜色渐变填充的区域具有非方形边界框（例如，矩形）时，并且指定半径 渐变圆相对于要填充的形状的大小，JavaFX将使用椭圆径向颜色渐变
- **cycleMethod** and **stops** 已经介绍过

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE), new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(0, 0, 0.5, 0.5, 0.5, true, NO_CYCLE, stops);
Circle c = new Circle(50, 50, 50);
c.setFill(rg);
```

Figure 7-19. *A radial color gradient with the same center point and focus point*

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE), new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(0, 0, 0.5, 0.5, 0.2, true, NO_CYCLE, stops);
Circle c = new Circle(50, 50, 50);
c.setFill(rg);
```

Figure 7-20. *A radial color gradient with the same center point and focus point having a gradient circle with a radius of 0.20*

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE), new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(0, 0, 0.5, 0.5, 0.2, true, REPEAT, stops);
Circle c = new Circle(50, 50, 50);
c.setFill(rg);
```

Figure 7-21. *A radial color gradient with the same center point and focus point, a gradient circle with a radius of 0.20, and the cycle method as REPEAT*

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE), new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(0, 0, 0.5, 0.5, 0.5, true, REPEAT, stops);
Rectangle r = new Rectangle(200, 100);
r.setFill(rg);
```

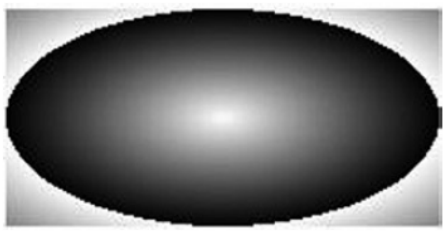

Figure 7-23. *A rectangle filled with a radial color gradient with a proportional argument value of true*

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE), new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(0, 0, 100, 50, 50, false, REPEAT, stops);
Rectangle r = new Rectangle(200, 100);
r.setFill(rg);
```


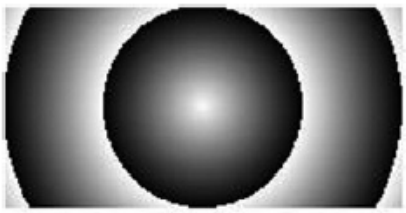

Figure 7-24. *A rectangle filled with a radial color gradient with a proportional argument value of false*

Table 7-1. *Criteria Used to Determine the Shape of a Radial Color Gradient*

Proportional Argument	Bounding Box for the Filled Region	Gradient Shape
true	Square	Circle
true	Nonsquare	Ellipse
false	Square	Circle
false	Nonsquare	Circle

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE), new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(0, 0, 0.5, 0.5, 0.2, true, REPEAT, stops);
Polygon triangle = new Polygon(0.0, 0.0, 0.0, 100.0, 100.0, 100.0);
triangle.setFill(rg);
```

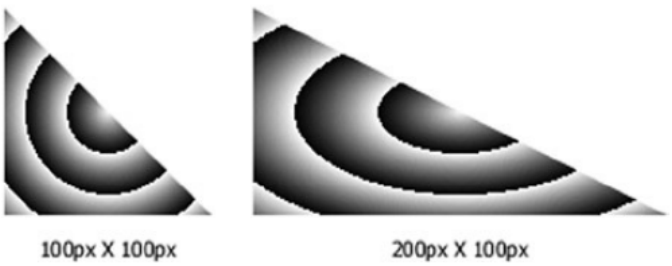

Figure 7-25. *Filling triangles with radial color gradients of circular and elliptical shapes*

```
Stop[] stops = new Stop[]{new Stop(0, Color.WHITE),
                          new Stop(0.40, Color.GRAY),
                          new Stop(0.60, Color.TAN),
                          new Stop(1, Color.BLACK)};
RadialGradient rg = new RadialGradient(-30, 1.0, 0.5, 0.5, 0.5, true, REPEAT, stops);
Circle c = new Circle(50, 50, 50);
c.setFill(rg);
```

Figure 7-26. *Using multiple color stops in a radial color gradient*

3.2Defining Radial Color Gradients in String Format

- 语法

```
radial-gradient([focus-angle],[focus-distance],[center],radius,[cycle-method],
color-stops-list)
```

  - **focus-angle** and **focus-distance** : 默认 0

    > You can specify the focus angle in degrees,radians, gradians, and turns

    ```
    focus-angle 45.0deg
    focus-angle 0.5rad
    foucs-angle 30.0grad
    focus-angle 0.125turn
    focus-distance 50%
    ```

  - **center** and **radius** :

    ```
    center 50px 50px,radius 50px
    center 50% 50%, radius 50%
    ```

  - **cycle-method** : **repeat** and **reflect**, 忽略则默认 NO_CYCLE，不能显式写NO_CYCLE
  - **color-stps-list**

    ```
    white, black
    white 0%, black 100%
    ```

```
String colorValue = "radial-gradient(focus-angle 45deg, focus-distance 50%, " +
                    "center 50% 50%, radius 50%, white 0%, black 100%)";
RadialGradient rg = RadialGradient.valueOf(colorValue);
Circle c = new Circle(50, 50, 50);
c.setFill(rg);
```

Figure 7-27. *Using string format for specifying a radial color gradient*

Summary

- **text color** and **background for region**
- uniform color
- image pattern
- linear gradient
- radial gradient
- **javafx.scene.paint**

- Paint
- Color
- ImagePattern
- LinearGradient
- RadialGradient
- Stop
- CycleMethod

> 下一章教你如何使用CSS在scene graph上面来装饰你的Node