**路径**

1. 路径相关图形状态有哪些？对应状态设置操作是什么？

PDF 的用户应用程序维护的一个内部数据结构称之为图形状态，其保存了控

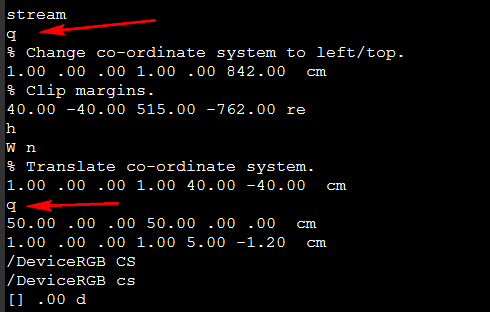
制参数的当前图形。这些参数定义了图形操作执行的整体架构。例如，f（填充）操作隐式使用了当前的颜色参数，另外 S（绘制）操作使用了图形状态的当前的

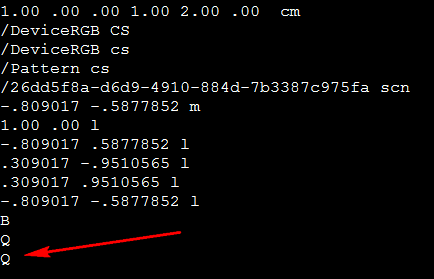
线宽参数。

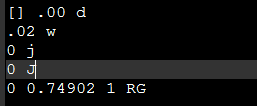
|  |  |  |
| --- | --- | --- |
| 路径相关的图形状态 | | |
| 操作数 | 操作 | 值 |
| Line width | w | 在用户空间单元上路径宽度,初始值：1.0 |
| Line cap | J | 指定被绘制的任何开放路径终点类型。初始值：0：无线头。 1：圆角线头 2：方形线头 |
| 无 | Q | 通过删除堆栈中最新存储的状态来还原图形状态，并让它成为当前状态。 |
| 无 | q | 保存图形状态堆栈中的当前图形状态 |
| a b c d e f | cm | 定义了一个状态转移矩阵 |
| lineJoin | j | 在图形状态里设置线连接样式。0：尖角连接 1：圆角连接 2：方角连接 |
| dashArray | d | 设置连接线为虚线模式 |

1. 基于DrawPath.pdf说明设置了哪些路径相关图形状态？

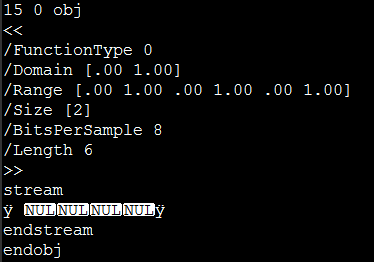
图像的状态栈：使用小写的字母q表示保存图形状态堆栈中的当前图形状态，使用大写的字母Q表示通过删除堆栈中最新存储的状态来还原图形状态，并让它成为当前状态。







使用了d定义了绘制的线为虚线，线宽(lineWidth)为0.02单位，使用j来设置连接线的样式，值为0，表示采用的连接样式为尖角连接。通过大写字母J来设置矢量线端的样式，为0表示采用的是无线头的方式。并且通过RG来设置画线（stroking）的红、绿、蓝三分量的颜色。



页面传递函数

1. 路径相关构建、绘制操作有哪些？

|  |  |  |
| --- | --- | --- |
| 路径相关构建、绘制操作 | | |
| 操作数 | 算子 | 描述 |
| x y | m | 表示将当前点移动到坐标(x,y)以开始一个新的子路径，忽略线段的连接。如果当前路径中之前的路径构造算子也是m，那么新的 m 算子将代替之前的，而之前的 m 算子操作的痕迹依然保留在路径中。 |
| x y | l | 从当前点向(x,y)点追加一条直线线段。新的当前点修改为(x,y) |
| 无 | h | 从当前点到子路径的开始点连接一条直线段，关闭当前子路径。如果当前子路径已经关闭，那么 h 算子不做任何事情。这个算子结束当前子路径。向当前路径追加入新的元素会开始一个新的子路径，即使新的路径从 h 操作的结束点开始。 |
| x y width height | re | 表示了向当前路径中加入一个矩形作为闭的子路径。矩形的左下角为(x,y)，宽为 width，高为 height。这个 x y width height re 的操作相当于：  x y m  (x + width) y l  (x + width) (y + height) l  x (y + height) l  h |
| 无 | S | 涂抹 |
| 无 | s | 涂抹并关闭路径,这个算子和h S的效果一样 |
| 无 | F | 相当于 f，为兼容性存在。尽管任何 PDF 读取程序都必须能够识别它，而 PDF 制作程序应该使用 f 代替 F |
| 无 | f | 填充路径，使用非零缠绕数规则来决定填充的区域。任何开着的子路径在填充之前都必须关闭。 |
| 无 | f\* | 填充路径，使用奇偶原则决定填充的区域 |
| 无 | B | 表示填充然后涂抹路径，使用非零缠绕数规则来决定填充区域。这个算子和使用 F 算子和 S 算子绘制的两个路径对象结合在一起的效果一样。 |
| 无 | B\* | 该算子表示填充然后并且涂抹该路径，使用奇偶原则决定填充的区域。这个算子和 B 算子得到相同的结果，只是使用 F\*算子填充的而已。 |
| 无 | b | 关闭，填充并且涂抹路径，使用非零缠绕数规则来决定填充区域。这个算子与算子序列 h B 具有相同的效果 |
| 无 | b\* | 关闭，填充并且涂抹路径，使用奇偶原则来决定填充区域。这个算子与算子序列 h B\* 具有相同的效果 |
| 无 | n | 表示不填充涂抹直接结束路径对象。这个算子是一个路径绘制空算子，在改变当前裁剪路径之前使用。 |

1. 基于DrawPath.pdf说明使用了哪些路径构建操作和路径绘制操作？



h:表示了从当前点到子路径的开始点连接一条直线段，关闭当前子路径。如果当前子路径已经关闭，那么 h 算子不做任

何事情。这个算子结束当前子路径。向当前路径追加入

新的元素会开始一个新的子路径，即使新的路径从 h 操

作的结束点开始。

re:表示了向当前路径中加入一个矩形作为闭的子路径。矩形的左

下角为(x,y)，宽为 width，高为 height。这个 x y width height re 的操作相当于：

x y m

(x + width) y l

(x + width) (y + height) l

x (y + height) l

h

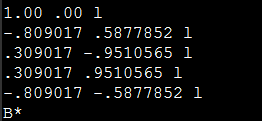


W：表示通过使当前路径与其相交来修改当前剪辑路径，使用非零绕数规则来确定哪些区域在剪辑路径内。

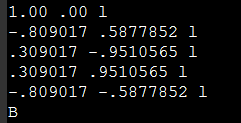
n:表示不填充涂抹直接结束路径对象。这个算子是一个路径绘制空算子，在改变当前裁剪路径之前使用。



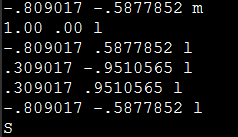
m:表示将当前点移动到坐标(x,y)以开始一个新的子路径，忽略线段的连接。如果当前路径中之前的路径构造算子也是m，那么新的 m 算子将代替之前的，而之前的 m 算子操作的痕迹依然保留在路径中。



B\*：该算子表示填充然后并且涂抹该路径，使用奇偶原则决定填充的区域。这个算子和 B 算子得到相同的结果，只是使用 F\*算子填充的而已。



B：表示填充然后涂抹路径，使用非零缠绕数规则来决定填充区域。这个算子和使用 F 算子和 S 算子绘制的两个路径对象结合在一起的效果一样。



S：该算子就表示涂抹

1. 使用Pdf API绘制直线、正方形（演示如何设置路径相关图形状态）？ 结合生成的pdf文档分析相关操作和图形状态

测试代码：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using Spire.Pdf;

using Spire.Pdf.Graphics;

namespace ConsoleApplication1

{

class Program

{

private void DrawStrightLine(PdfPageBase page)

{

PointF[] points = new PointF[4];

//clear the watermark

int downRate = 1;

int leftRate = 2;

points[0] = new PointF(-1-leftRate, -1+downRate);

points[1] = new PointF(-1-leftRate, 1+downRate);

points[2] = new PointF(1-leftRate, 1+downRate);

points[3] = new PointF(1-leftRate, -1+downRate);

PdfPath path = new PdfPath();

path.AddLine(points[0], points[1]);

path.AddLine(points[1], points[2]);

path.AddLine(points[2], points[3]);

path.AddLine(points[3], points[0]);

//save graphics state

//PdfGraphicsState state = page.Canvas.Save();

PdfPen pen = new PdfPen(Color.DeepSkyBlue, 0.02f);

page.Canvas.ScaleTransform(105f, 105f);

page.Canvas.TranslateTransform(5f, 1.2f);

page.Canvas.DrawPath(pen, path);

//drawing a stright line

PdfPath path2 = new PdfPath();

path2.AddLine(new PointF(-4.5f, 0), new PointF(-3f, 3.53f));

PdfPen pen2 = new PdfPen(Color.Red,0.046f);

page.Canvas.DrawPath(pen2, path2);

}

private void PDFDocumentViewer(string fileName)

{

try

{

System.Diagnostics.Process.Start(fileName);

}

catch { }

}

static void Main(string[] args)

{

Program mypro = new Program();

//Create a pdf document.

PdfDocument doc = new PdfDocument();

// Create one page

PdfPageBase page = doc.Pages.Add();

//mypro.DrawSpiro(page);

//mypro. DrawPath(page);

mypro.DrawStrightLine(page);

//Save pdf file.

doc.SaveToFile("DrawShape11111.pdf");

doc.Close();

//Launching the Pdf file.

mypro.PDFDocumentViewer("DrawShape11111.pdf");

}

}

}

结果如pdf所示：

