**坐标系统**

1. 用户坐标空间、设备坐标空间？

用户坐标空间：

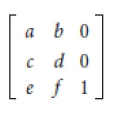
为了避免设备空间里指定对象的设备依赖影响，PDF 定义了一个始终承担着  
当前页面的相同关系的设备无关坐标系统，无论输出设备在打印或是显示。这个  
设备无关坐标系统称为用户坐标空间。

设备坐标空间：

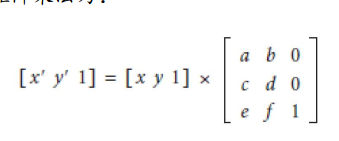
页面的内容最终显示在光栅输出设备，如显示器或者打印机。这种设备在在可成像领域用来处理像素的内置坐标系统里改变巨大。一个为特定设备的坐标系统被称为它的设备空间。如果 PDF 文件中的坐标被指定在设备空间，文件将会依赖设备，并且将会在不同设备上显示不同。

1. 坐标空间如何转换？

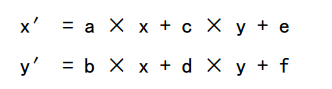
PDF 代表了一个二维空间的坐标。在这样一个空间的点(x,y)可以表示为向量形式[x y 1]。两个坐标系之间的转化时由一个 3 X 3 的转换矩阵表示如下：



由于一个转化矩阵只有 6 个元素是可以改变的，所以通常在 PDF 中它被指定  
为一个六元素的数组[a b c d e f]。坐标转换表示为矩阵乘法为



由于 PDF 转换矩阵指定了从转换坐标系到原始（未转换）坐标系的转换，在  
这个等式里 x’和 y’为未转换坐标系的坐标,x 和 y 是转换了的坐标系的坐标.  
乘法是按照以下程序进行:



上述过程也可以被表示为：



在上面的等式中，

XD 指在设备空间的坐标  
XU 指在默认用户空间的坐标  
XS 指按照一定的比例缩放用户空间的坐标

MC 表明一个状态转移矩阵

MS 表明一个指定缩放比例的矩阵

1. 使用GDI+绘制直线、矩形演示坐标空间定位、转换矩阵使用？

测试代码如下：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace ConsoleApplication1

{

class demo2

{

private static Bitmap myBitmap=null;

private static Graphics g;

public static void Main03(string[] args)

{

init();

createNewImage(g);

}

private static void pngViewer(string fileName)

{

try

{

System.Diagnostics.Process.Start(fileName);

}

catch { }

}

private static Graphics init() {

myBitmap = new Bitmap(450, 450);

//create the graphics by image

g= Graphics.FromImage(myBitmap);

return g;

}

private static void DisposeSource(Graphics g)

{

myBitmap.Save("result.png", System.Drawing.Imaging.ImageFormat.Png);

g.Dispose();

}

private static void DrawString(Graphics g) {

Font font = new Font("华为宋体", 12);

//Point一样，只是值是浮点类型

PointF point = new PointF(50, 50);

g.DrawString("我是adonai", font, Brushes.Coral, point);

DisposeSource(g);

pngViewer("result.png");

}

private static void createNewImage( Graphics g)

{

Pen p = new Pen(Color.Blue, 2);//定义了一个蓝色,宽度为的画笔

g.RotateTransform(12, System.Drawing.Drawing2D.MatrixOrder.Prepend);

g.ScaleTransform(2, 2, System.Drawing.Drawing2D.MatrixOrder.Prepend);

//g.ResetTransform();

g.DrawLine(p, 25, 25, 100, 100);//在画板上画直线,起始坐标为(10,10),终点坐标为(100,100)

g.DrawRectangle(p, 25, 10, 100, 100);//在画板上画矩形,起始坐标为(10,10),宽为,高为

// g.DrawEllipse(p, 10, 10, 100, 100);//在画板上画椭圆,起始坐标为(10,10),外接矩形的宽为,高为

DisposeSource(g);

pngViewer("result.png");

}

}

}

1. 使用Pdf绘制直线、矩形演示坐标空间定位、转换矩阵使用？

测试代码为：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using Spire.Pdf.Graphics;

using Spire.Pdf;

namespace ConsoleApplication1

{

class demo1

{

/// <summary>

/// how to draw a strightLine on page

/// </summary>

/// <param name="page"></param>

private void DrawStrightLine01(PdfPageBase page)

{

PointF[] points = new PointF[4];

//points[0] = new PointF(-1, -1);

//points[1] = new PointF(-1, 1);

//points[2] = new PointF(1, 1);

//points[3] = new PointF(1, -1);

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(95f, 95f);

// page.Canvas.RotateTransform(-11.5f);

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(-4.5f, 3.53f));

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

page.Canvas.DrawPath(pen2, path2);

}

private void DrawStrightLine(PdfPageBase page)

{

PointF[] points = new PointF[4];

int downRate = 350;

int leftRate = 150;

int baseRate = 94;

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

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

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

points[3] = new PointF(baseRate + leftRate, -baseRate + 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, 2f);

// page.Canvas.ScaleTransform(95f, 95f);

//skewTransform 斜变换

page.Canvas.SkewTransform(10, 10);

//page.Canvas.RotateTransform(-15.5f);

//translation operator

page.Canvas.TranslateTransform(0,-90);//shift shapes

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

page.Canvas.DrawPath(pen, path);

//drawing a stright line

PdfPath path2 = new PdfPath();

path2.AddLine(new PointF(0,450f), new PointF(258f,450f));

PdfPen pen2 = new PdfPen(Color.Blue, 3f);

page.Canvas.DrawPath(pen2, path2);

}

private void DrawRectangle(PdfPageBase page)

{

//notice the watermarker

//drawing a stright line

PdfPath path2 = new PdfPath();

path2.AddLine(new PointF(1f, 310), new PointF(1f, 50f));

PdfPen pen2 = new PdfPen(Color.Black, 2f);

page.Canvas.RotateTransform(-45f);

// page.Canvas.ScaleTransform(85f,85f);

page.Canvas.DrawPath(pen2, path2);

}

private void PDFDocumentViewer(string fileName)

{

try

{

System.Diagnostics.Process.Start(fileName);

}

catch { }

}

static void Main(string[] args)

{

demo1 mypro = new demo1();

//Create a pdf document.

PdfDocument doc = new PdfDocument();

// Create one page

PdfPageBase page1 = doc.Pages.Add();

PdfPageBase page2 = doc.Pages.Add();

mypro.DrawStrightLine(page1);

mypro.DrawRectangle(page2);

//Save pdf file.

doc.SaveToFile("DrawShape233.pdf");

doc.Close();

//Launching the Pdf file.

mypro.PDFDocumentViewer("DrawShape233.pdf");

}

}

}