**坐标系统**

1. 用户坐标空间、设备坐标空间？

用户坐标空间：

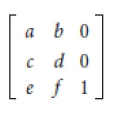
为了避免设备空间里指定对象的设备依赖影响，PDF 定义了一个始终承担着  
当前页面的相同关系的设备无关坐标系统，无论输出设备在打印或是显示。这个  
设备无关坐标系统称为用户坐标空间。

设备坐标空间：

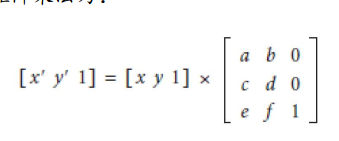
页面的内容最终显示在光栅输出设备，如显示器或者打印机。这种设备在在可成像领域用来处理像素的内置坐标系统里改变巨大。一个为特定设备的坐标系统被称为它的设备空间。如果 PDF 文件中的坐标被指定在设备空间，文件将会依赖设备，并且将会在不同设备上显示不同。

1. 坐标空间如何转换？

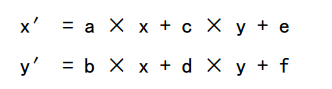
PDF 代表了一个二维空间的坐标。在这样一个空间的点(x,y)可以表示为向量形式[x y 1]。两个坐标系之间的转化时由一个 3 X 3 的转换矩阵表示如下：



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：

namespace GDI绘图

{

public partial class Form1 : Form

{

private static Graphics g;

private static GraphicsState gs;

private static Pen p;

public Form1()

{

InitializeComponent();

g = this.CreateGraphics();

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

}

private void drawCircle\_Click(object sender, EventArgs e)

{

gs = g.Save();

g.Clear(Color.WhiteSmoke);

g.TranslateTransform(20, 70);

g.DrawEllipse(p, 0, 0, 200, 200);

g.Restore(gs);

}

private void drawRec\_Click(object sender, EventArgs e)

{

gs = g.Save();

g.Clear(Color.WhiteSmoke);

int offset = 10;

//创建一个数组来存放矩阵的各个坐标

PointF[]mypointf=new PointF[] {

new PointF(0,0), new PointF(0,200) ,//

new PointF(200,200),new PointF(200,0),//

new PointF(0,0)};

for (int i = 0; i < mypointf.Length; i++)

{

mypointf[i].X+=offset;

mypointf[i].Y+=offset;

}

g.DrawLines(p, mypointf);//绘制矩形

g.DrawLines(p,new PointF[]{new PointF(300,100),new PointF(250,360)});

g.Restore(gs);

}

}

}

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：

namespace GDI绘图

{

/// <summary>

/// DrawDemo by PDF API

/// </summary>

public partial class PDFDrawDemo : Form

{

//save graphics state

private static PdfGraphicsState state;

public PDFDrawDemo()

{

InitializeComponent();

}

/// <summary>

/// open the process by fileName

/// </summary>

/// <param name="fileName"></param>

private void PDFDocumentViewer(string fileName)

{

try

{

System.Diagnostics.Process.Start(fileName);

}

catch { }

}

private void DrawCircleFunc(PdfPageBase page)

{

state = page.Canvas.Save();

PdfPen pen = new PdfPen(Color.DeepSkyBlue, 3);//边框的厚度3

PdfBrush brush1 = new PdfSolidBrush(Color.Red);//画刷，颜色为red

page.Canvas.ScaleTransform(5f, 5f);//缩放操作

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

page.Canvas.DrawEllipse(pen,brush1,0,0,20,20);//drawCircle

//restor graphics

page.Canvas.Restore(state);

}

private void DrawLineFunc(PdfPageBase page)

{

state = page.Canvas.Save();

//Draw Rectagle

PdfPen pen = PdfPens.DeepSkyBlue;

PdfBrush brush = new PdfSolidBrush(Color.Red);//画刷，颜色为red

PdfPath path = new PdfPath();

float offset = 50;

PointF[] points = new PointF[] { new PointF(0, 0), new PointF(0, 80), new PointF(80, 80), new PointF(80, 0) };

for (int i = 0; i < points.Length; i++)

{

points[i].X += offset;

points[i].Y += offset;

}

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

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

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

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

page.Canvas.DrawPath(pen, path);

page.Canvas.TranslateTransform(280, 0);

path.FillMode = PdfFillMode.Winding;

page.Canvas.DrawPath(pen, brush, path);

//drawing line

PdfPath Linepath=new PdfPath();

Linepath.AddLine(new PointF(70,270),new PointF(96,192));

page.Canvas.DrawPath(pen,Linepath);

//restor graphics

page.Canvas.Restore(state);

}

private void drawCircle\_Click(object sender, EventArgs e)

{

//Create a pdf document.

PdfDocument doc = new PdfDocument();

// Create one page

PdfPageBase page = doc.Pages.Add();

DrawCircleFunc(page);

//Save pdf file.

doc.SaveToFile("PDFDrawDemo.pdf");

doc.Close();

//Launching the Pdf file.

PDFDocumentViewer("PDFDrawDemo.pdf");

}

private void drawLine\_Click(object sender, EventArgs e)

{

//Create a pdf document.

PdfDocument doc = new PdfDocument();

// Create one page

PdfPageBase page = doc.Pages.Add();

DrawLineFunc(page);

//Save pdf file.

doc.SaveToFile("PDFDrawDemo.pdf");

doc.Close();

//Launching the Pdf file.

PDFDocumentViewer("PDFDrawDemo.pdf");

}

}

}