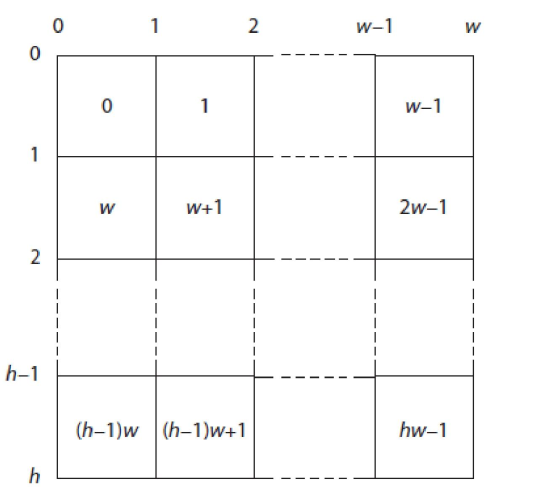
**图像**

1. 理解图像坐标空间。

每个图像都有自己的内部坐标系或图像空间。图像在图像空间中占据一个矩形，单位宽，单位高，其中w和h是样本中图像的宽度和高度。每个样本占据一个

平方单位。坐标原点（0,0）位于图像的左上角，坐标范围从左往右对应水平从0到w，垂直从上到下对应从0到h。

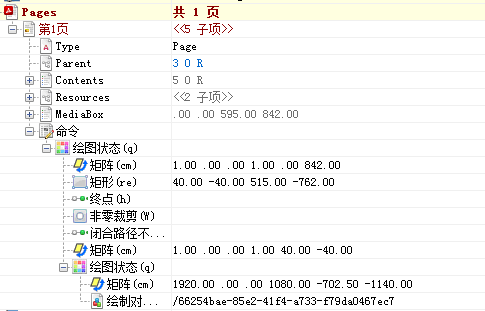


在上图中方块内的数字表示样本的顺序，从0开始计数。左上角第一个样本在坐标（0,0）处，第二个样本在（1,0）处，以此类推到最后一个。第一行的样本，其左上角位于（w – 1，0）并且其右上角位于（w，0）。之后的下一个样本位于坐标（0,1），（1,1）等，直到图像的最后一个样本，其左上角位于（w - 1，h - 1），右下角在（w，h）。

图像空间和用户空间之间的对应关系是恒定的：单位以用户坐标（0,0）和（1，1）为界的用户空间平方对应到图像空间中图像的边界。从图像空间到用户空间的转换可以用矩阵来描述[1/w 0 0 -1 / h 0 1]。

1. 基于DrawImage.pdf，理解图像坐标空间、用户坐标空间转换、理解图像数据。

在文档结构图中可以清楚的看到用户坐标系进行了怎样的转换：



1. 首先将用户坐标空间的原点沿y轴正方向平移到了(0,842)点处。
2. 指定了绘制图像矩形空间的坐标原点在点(40,-40)处，根据宽为515,高度为762个单位的矩形，对应到程序代码中的值分别为：

page.Canvas.ClientSize.Width 和page.Canvas.ClientSize.Heigh

1. 将坐标原点平移到点(40,-40)点处，准备进行绘图操作。

图像坐标空间中，首先将图像坐标系进行如下的缩放操作，缩放矩阵为[1920,0,0;0,1080,0;0,0,1],然后再将该坐标系原点平移到点（-702.5，-1140）点处。先进行缩放操作，然后再进行平移操作。对应如下代码：

doc = new PdfDocument();

page = doc.Pages.Add();

//添加图片

PdfImage image2 = PdfImage.FromFile(ImageFileName);

float width = image2.Width \* 0.75f;

float height = image2.Height \* 0.75f;

float x = (page.Canvas.ClientSize.Width - width) / 2;

page.Canvas.DrawImage(image2, x, 60);

然后对该DrawImage.pdf文件的坐标转换进行操作。

测试代码：

namespace DrawImageDemo

{

class Program

{

/// <summary>

/// 从pdf中抽取出image图片

/// </summary>

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

private static void ExcateImages(String fileName)

{

PdfDocument doc = new PdfDocument();

doc.LoadFromFile(fileName);

IList<Image> images = new List<Image>();

foreach (PdfPageBase page in doc.Pages)

{

if (page.ExtractImages() != null)

{

foreach (Image image in page.ExtractImages())

{

images.Add(image);

}

}

}

doc.Close();

int index = 0;

foreach (Image image in images)

{

String imageFileName = String.Format("Image-{0}.png", index++);

image.Save(imageFileName, ImageFormat.Png);

}

}

/// <summary>

/// 根据图片和文字生成方法

/// </summary>

/// <param name="ImageFileName"></param>

/// <param name="pdfFileName"></param>

private static void GenaratePDF(String ImageFileName, String pdfFileName)

{

PdfDocument doc=null;

PdfPageBase page;

try

{

doc = new PdfDocument();

page = doc.Pages.Add();

//添加图片

PdfImage image2 = PdfImage.FromFile(ImageFileName);

float width = image2.Width \* 0.75f;

float height = image2.Height \* 0.75f;

float x = (page.Canvas.ClientSize.Width - width) / 2;

//page.Canvas.ClientSize.Width 可绘制的pdf图像的区域的宽

Console.WriteLine("page.Canvas.ClientSize.Width =" + page.Canvas.ClientSize.Width);

Console.WriteLine("width=" + width+",height ="+height);

Console.WriteLine("x="+x);

Console.WriteLine("page.Canvas.ClientSize.heigh="+page.Canvas.ClientSize.Height);

page.Canvas.DrawImage(image2, x, 33);

doc.SaveToFile(pdfFileName);

PDFDocumentViewer(pdfFileName);

}

catch (Exception e)

{

throw e;

}

finally

{

if(!doc.Equals("")||doc!=null)

doc.Close();

}

}

private static void PDFDocumentViewer(string fileName)

{

try

{

System.Diagnostics.Process.Start(fileName);

}

catch { }

}

static void Main(string[] args)

{

ExcateImages("3 DrawImage.pdf");

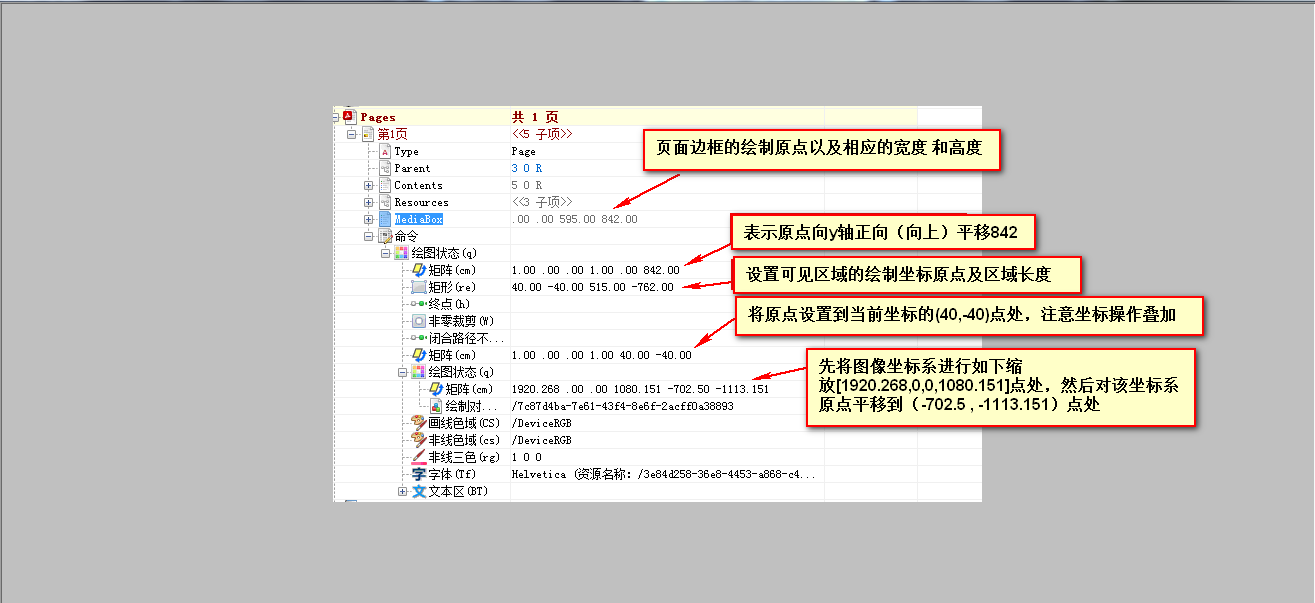
GenaratePDF("Image-0.png", "adonai111.pdf");

}

}

}

生成的文档结构图数据如下所示：



其中的绘图状态中第二个q指令所涉及到的cm转换矩阵是在图像坐标系中完成的，但是外面嵌套了一个用户坐标系的操作。这里也是一样的，先进行缩放操作，然后再进行平移操作。只不过这次调用DrawImage方法的时候传入的图像坐标原点的值从原来的(-702.5,60)修改成了(-702.5,33)