区域设置为整个窗口。其余的调用以其他的方法限定修剪区域。拥有多种调用做几乎相同的事情是Windows的另一个特性。

GDI的完全论述超出了这里讨论的范围。对于感兴趣的读者，上面引用的参考文献提供了补充的信息。然而，关于GDI可能还值得再说几句话，因为GDI是如此之重要。GDI具有各种各样的过程调用以获取和释放设备上下文，获取关于设备上下文的信息，获取和设置设备上下文的属性（例如背景颜色），使用GDI对象（例如画笔、画刷和字体，其中每个对象都有自己的属性）。最后，当然存在许多实际在屏幕上绘图的GDI调用。

绘图过程分成四种类型：绘制直线和曲线、绘制填充区域、管理位图以及显示文本。我们在上面看到了绘制文本的例子，所以让我们快速地看看其他类型之一。调用

```
Rectangle(hdc, xleft, ytop, xright, ybottom);
```

将绘制一个填充的矩形，它的左上角和右下角分别是（xleft, ytop）和（xright, ybottom）。例如，

```
Rectangle(hdc,2,1,6,4);
```

将绘制一个如图5-41所示的矩形。线宽和颜色以及填充颜色取自设备上下文。其他的GDI调用在形式上是类似的。

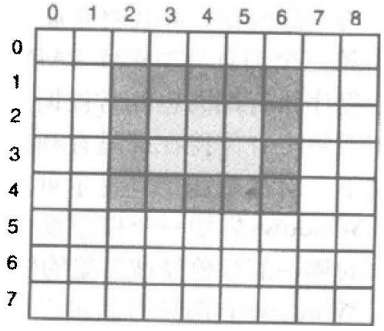

图5-41 使用Rectangle绘制矩形的例子。每个方框代表一个像素

4. 位图

GDI过程是矢量图形学的实例。它们用于在屏幕上放置几何图形和文本。它们能够十分容易地缩放到较大和较小的屏幕（如果屏幕上的像素数是相同的）。它们还是相对设备无关的。一组对GDI过程的调用可以聚集在一个文件中，描述一个复杂的图画。这样的文件称为Windows元文件（metafile），广泛地用于从一个Windows程序到另一个Windows程序传送图画。这样的文件具有扩展名.wmf。

许多Windows程序允许用户复制图画（或一部分）并且放在Windows的剪贴板上，然后用户可以转入另一个程序并且粘贴剪贴板的内容到另一个文档中。做这件事的一种方法是由第一个程序将图画表示为Windows元文件并且将其以.wmf格式放在剪贴板上。此外，还有其他的方法做这件事。

并不是计算机处理的所有图像都能够使用矢量图形学来生成。例如，照片和视频就不使用矢量图形学。反之，这些项目可以通过在图像上覆盖一层网格扫描输入。每一个网格方块的平均红、绿、蓝取值被采样并且保存为一个像素的值。这样的文件称为位图（bitmap）。Windows中有大量的工具用于处理位图。

位图的另一个用途是用于文本。在某种字体中表示一个特殊字符的一种方法是将其表示为小的位图。于是往屏幕上添加文本就变成移动位图的事情。

使用位图的一种一般方法是通过调用BitBlt过程，该过程调用如下：

```
BitBlt(dsthdc, dx, dy, wid, ht, srchdc, sx, sy, rasterop);
```

在其最简单的形式中，该过程从一个窗口中的一个矩形复制位图到另一个窗口（或同一个窗口）的一个矩形中。前三个参数设定目标窗口和位置，然后是宽度和高度，接下来是源窗口和位置。注意，每个窗口都有其自己的坐标系，（0，0）在窗口的左上角处。最后一个参数将在下面描述。

```
BitBlt(hdc2, 1, 2, 5, 7, hdcl, 2, 2, SRCCOPY);
```

的效果如图5-42所示。注意字母A的整个5×7区域被复制了，包括背景颜色。

除了复制位图外，BitBlt还可以做很多事情。最后一个参数提供了执行布尔运算的可能，从而可以将源位图与目标位图合并在一起。例如，源位图可以与目标位图执行或运算，从而融入目标位图；源位图还可以与目标位图执行异或运算，该运算保持了源位图和目标位图的特征。

位图具有的一个问题是它们不能缩放。8×12方框内的一个字符在640×480的显示器上看起来是适度的。然而，如果该位图以每英寸1200点复制到10 200位×13 200位的打印页面上，那么字符宽度（8像素）为8/1200英寸或0.17mm。此外，在具有不同彩色属性的设备之间进行复制，或者在单色设备与彩色设备之间进行复制效果并不理想。