1

KPL 编程入门

作者：**Jon SchWartz**

翻译：敏捷培训大巴

2006

目录

简介：什么是编程？

电脑编程其实很简单：就是向电脑下指令。

电脑可以很好地执行指令。我们告诉它什么，它就执行什么。但是他们没有想象力！因此当我们编写电脑指令，必须把我们想做的事情正确无误的传递给它。

为什么我要学习 KPL 编程？

不同的电脑编程语言，对于编程者而言，只是意味着用不同的方式告诉电脑工作的方式。Kids Programming Language(儿童编程语言)，简称 KPL，对初学者有以下优势：

- KPL 专门设计以使初学者尽可能容易地学习
- KPL 专门设计以使在学习中感到乐趣
- KPL，与其它的初学者语言不同，专门设计成跟现今的专业编程语言尽可能地相似

人们总是重复着一句古老的谚语，这个谚语也同样适合 KPL：

在学会跑之前，先学会走路

学习 KPL 编程就是学会“走路”的编程。学完 KPL 编程后，你就可以非常容易的学**跑**了-无论你决定开始学习 Java，Python，Visual Basic 或是 C#.

如何使用本教程？

没有编程经验的初学者，最好的学习办法就是从头至尾按顺序的阅读和学习本教程。在开始新内容前，确信已经理解了前面每节的内容。这可能是相对有效的方法，尤其是你的机器上还没有安装有 KPL 的时候，这样保持到你开始进入“用 KPL 编程”的学习章节。这种研读和学习 KPL 的方法，会有助于在学习的过程中集中注意力，避免复杂性。

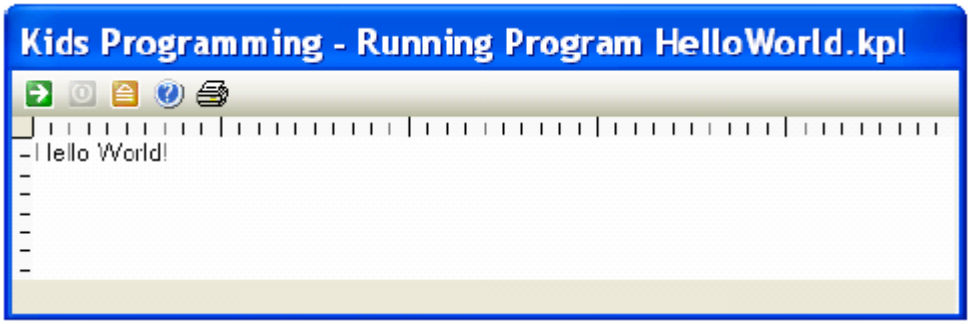
电脑编程意味着学习跟人的正常思维情况不同的思维方式。电脑要求我们比平时更加逻辑，顺序化和精确。刚开始的时候，可能会很难，但是你能做到！一旦你可以开始驾驭他，它就真正的为你所用。

学习新东西的最好的方法之一是向有编程经验的人提问并获得解答。那么，现在你想想看，谁可以在学习 KPL 的过程中帮助你呢？

好了，让我们来看程序了 ！

```
1  Program HelloWorld
2      Method Main()
3          Print("Hello World!")
4      End Method
5  End Program
```

就是它了！如果你运行这段 KPL 代码，你将会看到：



你看见这 5 行代码就是 KPL 中代码的截屏，跟你在实际的 KPL 开发过程中看到的 KPL 代码完全一样。

首先要注意的是，最左边的行号：对 KPL 自身来说，没有什么用途，只是协助便于阅读和定位 KPL 代码。旧的编程语言，比如 GWBASIC 用代码前的行标来定义代码的执行顺序，但 KPL 不是这样定义的。KPL 程序通常按一次执行一个结构片段的方式，从头到尾地执行代码。关于这一点，超出了教程的范围，我们在这里暂不讨论。

KPL 编程重要的规则之一是每一行 KPL 代码在程序文件中都必须是单独的一行。比如，下面的 KPL 代码跟上面的代码内容一样，但是没有单独成行，因此程序无法运行：

```
1  Program HelloWorld  Method Main()  Print("Hello World!")  End Method  End Program
```

所有的电脑编程语言都有一些规则，电脑会按照编程的指令执行代码。人与人的交流也有很多规则-我们如此多得使用这些规则，只是我们时常并没有意识到我们在使用这些规则。比如，我们不会拿起

电话就说“再见!”，也不会挂机的时候说“你好!”，当然，这只是简单的举例，但非常类似。因此 KPL 也要求一个特殊的方式表明开始和结束。所有的 KPL 代码都必须象下面第 1 行：Program Helloworld 开头的，所有的 KPL 代码都必须以象下面第 5 行 End Program 结束：

```
1 Program HelloWorld
2     Method Main()
3         Print("Hello World!")
4     End Method
5 End Program
```


你可以任意命名程序名称，但最好的命名应该描述程序的用途。在这个例子中，我选择 HelloWorld 作为程序名称。你也可以使用一个完全不同的名字，比如程序命名为：MyFirstProgram。

然后，要重点了解 KPL 编程的 Method Main()方法。所有的 KPL 程序从这个方法开始执行。你会看到这个结构里面的执行语句是：Print(“Hello World!”)

Method Main()用来定义成程序开始执行的起始方法-这一点与现代编程语言如出一辙。跟预料的一样，End Method 与 Method Main()匹配结对出现。

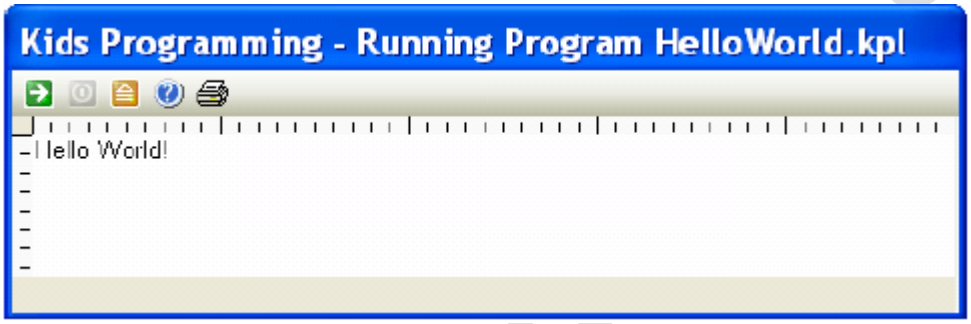
本教程不深入地解释什么是方法，但是，你会从包含 print(“Hello World!”)代码的 Method main()这样的结构中感受到。

以上就是这段仅包含一句有效指令的特殊程序的全部解释。该指令告诉电脑执行 Print(“Hello World!”).

好的，总结一下：代码第 1 行和第 5 行告诉电脑开始和结束。第 2 行和第 4 行告诉电脑 Method Main()的开始和结束。

```
1  Program HelloWorld
2      Method Main()
3          Print("Hello World!")
4      End Method
5  End Program
```

现在让我们重新运行程序，看看窗口会出现什么。注意，在窗口的“内部”，电脑只做了一件事情，我们通过 KPL 告诉它该做的一件事情。电脑输出了：Hello World ！



什么是电脑图形？

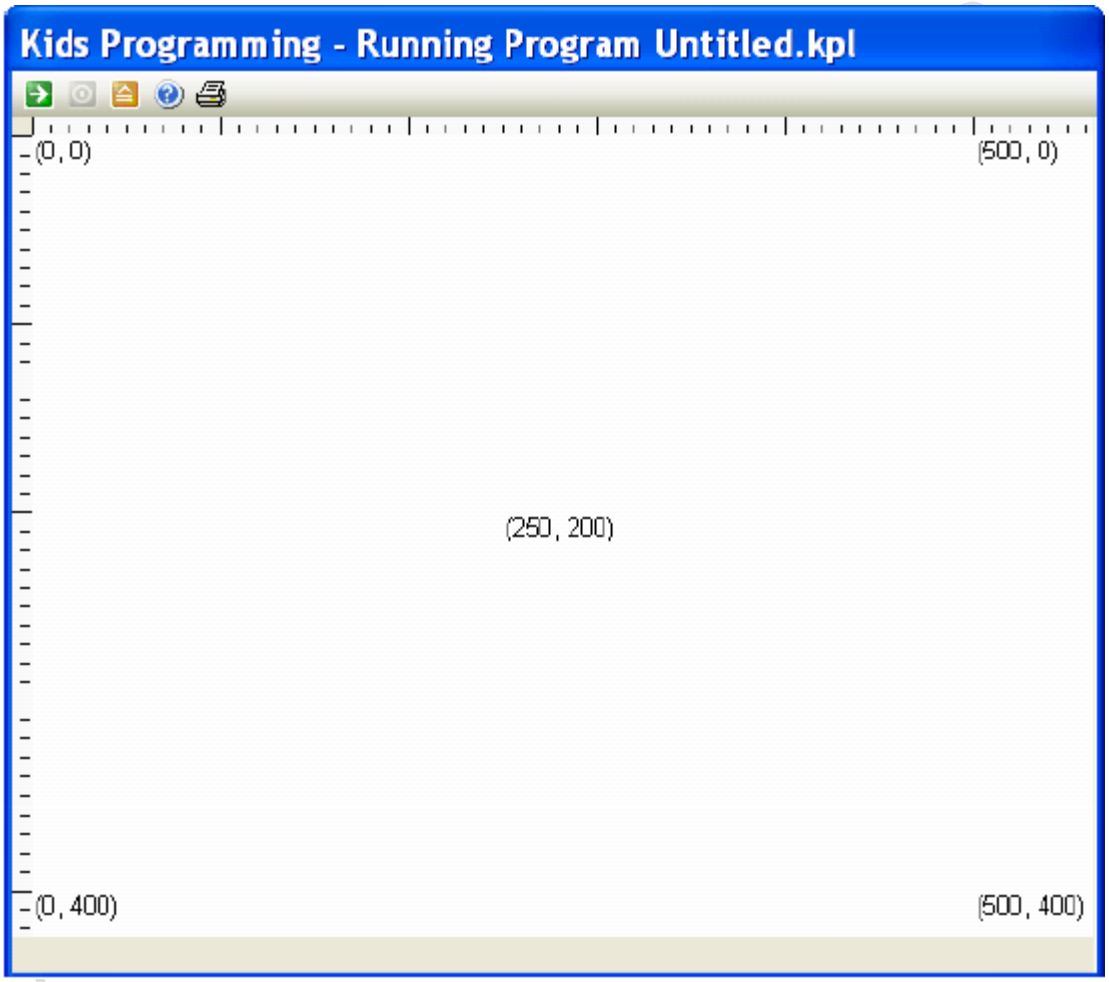
Hello World 是一个经典程序-但是，它一点都不让人激动！好吧，我们来聊聊有趣的图形编程吧！

我们从如何在屏幕上放置图形为例来说明电脑图形。电脑上使用的坐标体系跟我们在学校里学的坐标体系有所不同。不过，好在电脑坐标也非常容易使用，特别是因为它非常方便于在屏幕上定位。

电脑使用（x，y）的坐标体系定位屏幕，屏幕左边（Left）定义

为 X=0,顶部（Top）定义为 Y=0。这意味着这样的 x=0 和 y=0 这样的位置，就是屏幕的左上角。向右移动，X 值增加，向下移动，Y 值增加。

如果你还不能理解，这有一个详细的图示，表明了在 KPL 程序中几个（x，y）坐标的显示定位。



请留意上面的数字。在屏幕上方的第一个坐标的 X 值，正如你看到的一样，从左到右移动，X 的值会增加。在屏幕下方的第二个坐标的 Y 值，也如你所见，随着从顶部到底部的移动，Y 值也会增加。

接下来，我们开始写第一个 KPL 的图形程序。在这个过程中，我

们会展示如何在图形坐标系统中应用 KPL 在屏幕上做出很酷的图形。首先，我们运行这个完整的 KPL 程序实例：程序运行后，会创建一个图形。然后，我们逐步地详解 KPL 代码的细节，说明它是如何工作的。

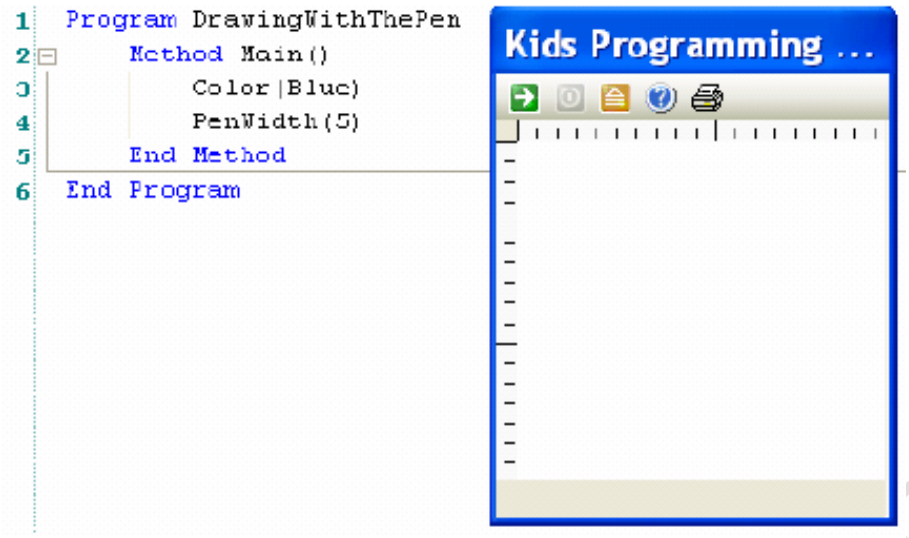
```
1  Program DrawingWithThePen
2      Method Main()
3          Color(Blue)
4          PenWidth(5)
5          MoveTo(100, 0)
6          MoveTo(50, 150)
7          MoveTo(180, 50)
8          MoveTo(20, 50)
9          MoveTo(150, 150)
10         MoveTo(100, 0)
11     End Method
12 End Program
```

注意，在 Method Main（）方法里，从第 3 行到第 10 行，一共有 8 行 KPL 指令。指令要比我们学的第一个程序略为多一点，但仅仅通过 8 行指令，我们可以在电脑屏幕上画出一个蓝色的五星来。这真是很酷的事情！

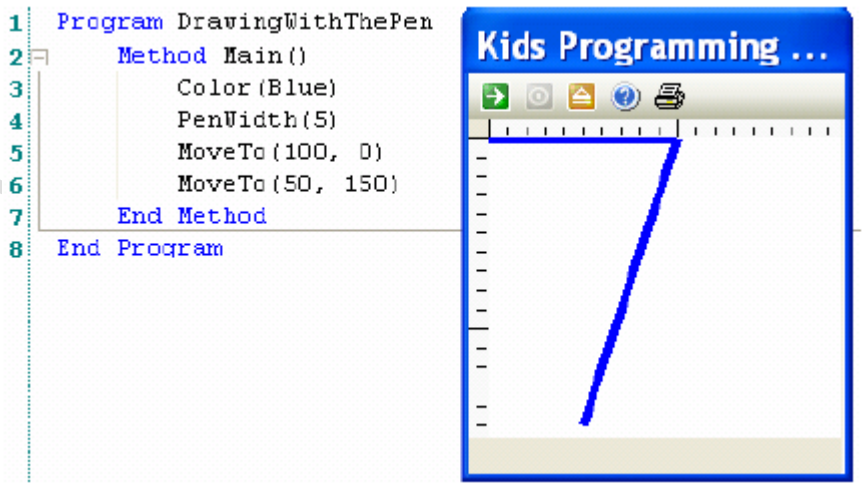
注意，这段程序的开始和结束方式，跟我们前面提到的第一个程序是一样的。除了程序的名称命名成 DrawingWithThePen：

```
1  Program DrawingWithThePen
2      Method Main()
3
4      End Method
5  End Program
```

现在，让我们开始加入指令，看看每一次都做了些什么：



我们先加入两行指令，正如你所见，程序不显示任何东西。Color(Blue)告诉 KPL 我们想用蓝色的画笔。PenWidth(5)告诉 KPL：当我们画画时，我们希望画笔画出的宽度是 5 象素的宽度。当然，你也可以设置其它的任意值。PenWidth(2)会画出细一点的线，PenWidth(10)画出的线会粗一些。然而，我们虽然告诉了 KPL 该如何画画，但我们没有告诉它要画什么。好吧，现在就让我们来画五星的第一条线。



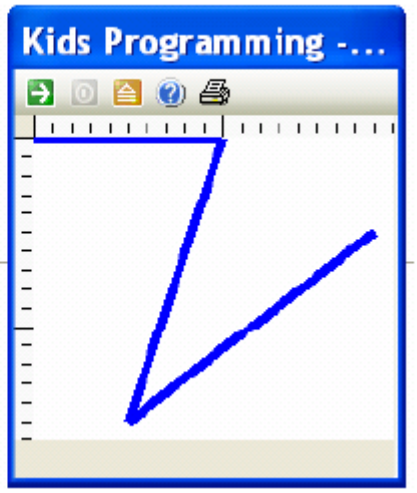
我们加的第一条指令是 Move(100,0)。告诉 KPL 移动画笔到屏幕

上的（100,0）位置，就是五星开始的顶点。KPL 的画笔位置总是从左上角（0，0）的位置开始，然后如所见，水平移动到（100，0）的位置。

新增的第二条指令是 Move(50,150)。这条指令画出了五星的第一条线，想一想这两个点的（X，Y）值。这条指令告诉 KPL 把画笔从 X=100 移到 X=50，线条向左移，然后 Y=0 到 Y=150，线向屏幕的下方移动。

现在，让我们增加更多的指令，告知 KPL 画出下一条线：

```
1  Program DrawingWithThePen
2      Method Main()
3          Color(Blue)
4          PenWidth(5)
5          MoveTo(100, 0)
6          MoveTo(50, 150)
7          MoveTo(180, 50)
8      End Method
9  End Program
```



新增的指令是：Move(180,50)。KPL 继续从先前的（50，150）移动。你曾经见过 Etch-A-Sketch-那种可以切换两个开关在屏幕上画画的红色酷玩吗？KPL 的画笔非常象是电子的 Etch-A-Sketch。让我们来完成五星新增其余的线条吧：

```
1  Program DrawingWithThePen
2      Method Main()
3          Color(Blue)
4          PenWidth(5)
5          MoveTo(100, 0)
6          MoveTo(50, 150)
7          MoveTo(180, 50)
8          MoveTo(20, 50)
9          MoveTo(150, 150)
10         MoveTo(100, 0)
11     End Method
12 End Program
```

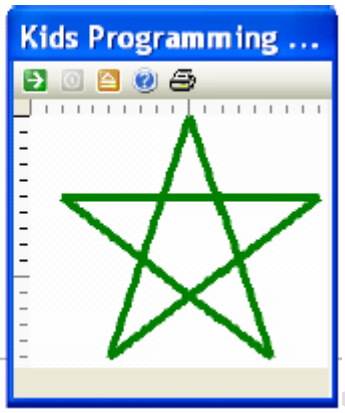
如你所见，新增的三条 KPL 指令画出了其余三条五星线。总之，按照我们的 KPL 程序指令，我们移动了 6 次画笔的位置，画出了 6 条直线，得到了一个五星。

完成上诉工作，KPL 只需做很少的工作：仅仅用了 8 条指令。在 KPL 图形编程里面最难的事可能就是要理解（X，Y）坐标系统。关于这一点，（X，Y）的位置到底代表什么，如果你还不是非常清楚的话，你可能需要回到前面的解释中去反复阅读。如果对（X，Y）坐标体系没有直接的体会，你可以在电脑上编写一些的 KPL 代码，体验一下在屏幕的不同位置画一些线条。可能你需要画一个矩形和一个三角形？如果你要马上开始 KPL 编程，你可以先跳过本节，先阅读“**用 KPL 编程**”章节的相关帮助。当你适应了（X，Y）坐标系统，你再回到这里来继续。

花足够的时间和精力在适应和了解 KPL 的（X，Y）坐标体系上是非常重要的。因为这是所有图形编程的基础。其他的图形编程也是这样。因此，当你学习 KPL 时，你也正在学习大多数电脑语言的图形编程的最基本的概念。

让我们来为程序增加一个细节：给 KPL 画笔一些额外的控制。这些事情 Etch-A-Sketch 可做不成，你无法控制 Etch-A-Sketch 画笔的颜色，但 KPL 却不费吹灰之力：

```
1  Program DrawingWithThePen
2      Method Main()
3          Color(Green)
4          PenWidth(5)
5          Pen(False)
6          MoveTo(100, 0)
7          Pen(True)
8          MoveTo(50, 150)
9          MoveTo(180, 50)
10         MoveTo(20, 50)
11         MoveTo(150, 150)
12         MoveTo(100, 0)
13     End Method
14 End Program
```



如所见，我们只是把 Color（blue）变成 Color(Green)。非常简单！还有，我们在 Move（100，0）前，新增了 Pen(false),Pen(false)告诉 KPL 在移动过程中，不要画线。所以，当画笔从（0，0）移动到（100，0）的过程中，不会再画出一条水平直线-这样五星看起来更好看！

记住，当我们说 Pen（false）的时候，电脑会严格执行该指令，当画笔移动时，KPL 停止画线。所以当我们希望 KPL 开始画五星时，我们同样需要给 KPL 一条指令：Pen（True），当它移动时，开始画线。

总结一下，Pen(False)告知 KPL 移动画笔时关闭画线，Pen(True)告知 KPL 移动画笔时开启画线。Etch-A-Sketch 可做不到这些！当你要想画各种各样的物体时，你可以随心所欲的开关画笔。这里有一个简单的例子：移动时关闭，画画时打开。

```
1  Program DrawingWithThePen
2      Method Main()
3          Color(Green)
4          PenWidth(5)
5          MoveTo(100, 100)
6          Pen(False)
7          MoveTo(100, 0)
8          Pen(True)
9          MoveTo(0, 100)
10     End Method
11 End Program
```

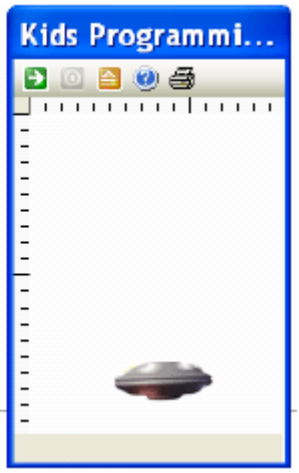
电脑精灵

画线的例子通常用于来说明显示图形的（X，Y）坐标体系。但游戏图形通常不需要画图。他们总是装入现成的图片：UFO、小行星或小飞侠。KPL 使用精灵作为图形显示的方法，这使得一切都变得非常简单！

跟上次课一样，我们先演示一个已经完成的程序，然后详细讲解程序工作的细节。然后你可以参照这个编写自己的程序。

这就是完整的 KPL 程序-跟上次的例子一样，在 Meth Main()中也只有 8 条指令。这 8 条指令，告知 KPL 在屏幕的上方显示一个 UFO，然后 UFO 再缓慢的移动到屏幕的下方：

```
1  Program UFO
2
3      Method Main()
4
5          LoadSprite( "UFO", "UFO.gif" )
6          MoveSpriteToPoint( "UFO", 50, 0 )
7          ShowSprite( "UFO" )
8
9          Var Y As Int
10         For Y = 1 To 150
11             Delay ( 10 )
12             MoveSpriteToPoint( "UFO", 50, Y )
13         Next
14
15     End Method
16
17 End Program
```



注意，程序的开始和结束跟先前的例子相同-正如所有的 KPL 程序要做的一样-但程序名命名为 UFO：

```
1  Program UFO
2
3      Method Main()
4
5      End Method
6
7  End Program
```

我们来看看 KPL 包含的一些图片文件。在 KPL 中提供了 65 张图片，但你可以使用任意的图片文件：添加现成的或自己创建的图片。


UFO.GIF


QUAD.GIF


PLANE.PNG


SPIDER.PNG


BLUEBALL.PNG

现在你已经看了 KPL 提供的图片了，我们开始新增 KPL 指令，在屏幕的上方显示 UFO 图形：

```
1  Program UFO
2
3      Method Main()
4
5          LoadSprite( "UFO", "UFO.gif" )
6          MoveSpriteToPoint( "UFO", 50, 0 )
7          ShowSprite( "UFO" )
8
9      End Method
10
11 End Program
```

我们加的第一条指令是：LoadSprite（“UFO”，”UFO.gif”）。

如前所说，KPL 有一些保留指令的关键词。LoadSprite 这个关键词，用法不算复杂，但要求明确！LoadSprite 要求两个确定的变量值才能正确运行。第一个变量值必须是我们所指的精灵的名字-在这个例子中，我们命名为 UFO。第二个变量值必须是指定精灵的图片文件所在的位置。变量值必须用双引号引用，变量间用逗号分隔。

下面有几个用 KPL 装入精灵的不正确的示例。重复一次：**下面的四个指令都不能正确运行，因为 KPL 不能理解不正确的 LoadSprite 指令**。你能修正其中的错误，使之能正确运行吗？

LoadSprite(‘UFO’ ,’ UFO.gif’)

LoadSprite(UFO,UFO.gif)

LoadSprite(“UFO” .” UFO.gif”)

LoadSprite(“UFO”)

总结：

LoadSprite (“UFO”,“UFO.gif”) 告知 KPL 用 UFO.gif 图片建立一个新的精灵，并把这个精灵命名为：UFO

现在 KPL 要知道在屏幕的什么位置显示该精灵。我们给出下一条指令：MoveSpriteToPoint(“UFO” ,50,0)。MoveSpriteToPoint 要求输入三个变量。第一个变量是精灵的名称，就是我们在 LoadSprite 中命名的精灵名称。第二个变量是精灵的 x 坐标的位置，我们设为 50，第三个变量是精灵的 Y 坐标位置，我们设为 0 。

一个重要的细节要提醒注意：数字变量不需要用括号引用。通常，KPL 对字符变量要求用括号引用，而对数字变量则不需要。关于这一点，在 **KPL 用户手册** 中有更多详细的信息。

总结：

MoveSpriteToPoint(“UFO”,50,0)告知KPL移动UFO精灵到屏幕上的（50，0）的点位置。

剩下的工作就是告诉 KPL 显示该精灵。这也是最容易的指令：ShowSprite（“UFO”）。这就好了，屏幕上 UFO，就是我们告知 KPL 放置精灵。

以上是非常详细的解释。让我们回过头来看看 KPL 是如何简单地完成这些的。下面是 KPL 程序，和运行时的效果。

```
1  Program UFO
2
3      Method Main()
4
5          LoadSprite| "UFO", "UFO.gif" )
6          MoveSpriteToPoint( "UFO", 50, 0 )
7          ShowSprite| "UFO" )
8
9      End Method
10
11 End Program
```

在 KPL 中使用变量和循环

现在我们想要 UFO 从屏幕的上方“降落”到窗口的底部。为此，我们将定义程序的第一个变量，你会看到这个变量在程序中是如何工作的。

变量这个说法表明-这个值可以随时都可以改变。正如将在这个例子中所见，变量可以随时改变它的值。

变量的命名原则，一般为其使用用途的关联名称（注：这样便于阅读程序的人理解）。在本例中，我们的变量用于改变 UFO 的 Y 坐标，使 UFO 从屏幕中“降落”，因此我们按理称该变量名为：Y。

下面是定义变量的代码：

```
Define  Y  As  Int
```

Define 是 KPL 关键词，KPL 用这个来标明你要使用一个新的变量。Y 就是要使用的变量的名称。As Int 告知 KPL：变量 Y 定义为整型变量。Int 是 Integer（整数）的简写。整型变量容纳整数值，如-1 或 0 或 43。整型变量只容纳整数。

我们知道 UFO 从屏幕上的（50，0）处开始，因为我们已告知 KPL 从这里开始移动精灵。但我们如何使 UFO“降落”到屏幕下方位置呢？我们可以不断地增加 Y 坐标的值—先到（50，1）然后是（50，2），然后是（50，3），是（50，4），一直这样…当到达（50，150）处的时候停止移动。

看到这个移动变化的过程了吗？精灵的 X 坐标位置一直是 50，精灵的 Y 坐标位置每次增加 1 个象素长度，从 1 直到 150。在我们编

写代码前，我们理解这个移动变化过程是非常重要的，如果你还没有体会是怎么的一回事情，请反复阅读体会。

以下 KPL 代码告知 KPL：增加 Y 变量值从 1 到 150，每次增量为 1：

```
For Y=1 To 150
    MoveSpriteToPoint(“UFO”,50,Y)
Next
```


这是你的第一个“**循环**”程序。**循环**是需要学习的新的编程概念，尤其是对于以前没有过编程经验的人而言。但循环的原理非常容易理解，因此也非常简单。

上述的循环从 Y=1 开始，如我们所说到 150 止，就是当 Y=150 时，循环停止。注意，在循环尾部定义的关键词 **Next**，当 KPL 程序执行到这个 Next 时，KPL 知道此时 Y 值会增加到下一个值：从 1 到 2，或从 2 到 3，从 3 到 4，依此类推...，从 149 到 150。

基本上，我们告知 KPL 从 1 到 150 开始计数。我们使用变量 Y 跟踪计数的值。

我们希望 KPL 在计数的同时做什么呢？我们想要移动 UFO 到屏幕右方？这正是 MoveSpriteToPoint(“UFO”,50,y)告知 KPL 所做的事情。

既然这条指令在 For 循环的“内部”，那么 KPL 会在从 1 到 150 每次计数的同时执行该指令。当然，这是的循环，仅仅是从 1 到 150 计数，没有什么乐趣。但是，**你可以在计数时，完成一些非常有用的事情**。现在我们来做点很酷的事情，移动 UFO 到屏幕下方，让我们来练习该如何做：


```
For Y=1 To 150
    MoveSpriteToPoint(“UFO”,50,Y)
Next
```

我们已经了解了 MoveSpriteToPoint 如何工作了：KPL 移动 UFO 精灵到给定的（X，Y）处。这次有何不同呢？以前，我们移动 UFO 到（50，0），但在循环中，我们用变量 Y 来代替位置，会发生什么情况呢？记住第一次循环的值是 Y=1，第二次是 Y=2，然后是 Y=3 ，是 Y=4，如此…同理到 Y=150。因此，第一次循环，KPL 使用 Y=1 的值移动精灵，等闲于下面的指令：

```
MoveSpriteToPoint(“UFO”,50,1)
```

然后，下次循环，Y=2 ，KPL 执行：

```
MoveSpriteToPoint(“UFO”,50,2)
```

如上，直到 KPL 使用循环计数直到 150：

```
MoveSpriteToPoint(“UFO”,50,3)
MoveSpriteToPoint(“UFO”,50,4)
MoveSpriteToPoint(“UFO”,50,5)
…
```

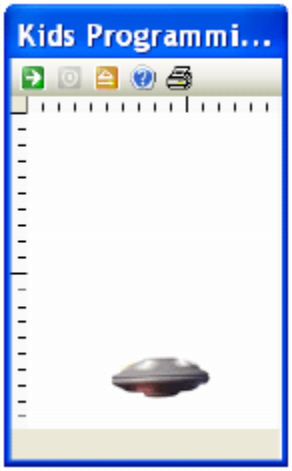
MoveSpriteToPoint(“UFO”,50,150)

精灵每移动一次，向下移动 1 个象素-正如我们希望的一样。

这就是你的第一个变量和第一个循环，以及在循环内执行的指令-这些都是非常重要的编程概念！如果你对此还不是非常清楚，请回头过去重新阅读。

我们加一些其他的细节完善这个程序示例：

```
1  Program UFO
2
3      Method Main()
4
5          LoadSprite( "UFO", "UFO.gif" )
6          MoveSpriteToPoint( "UFO", 50, 0 )
7          ShowSprite( "UFO" )
8
9          Var Y As Int
10         For Y = 1 To 150
11             Delay ( 10 )
12             MoveSpriteToPoint( "UFO", 50, Y )
13         Next
14
15     End Method
16
17 End Program
```



在 for 循环的内部，新增的仅有的一条指令是：Delay(10)。我们为什么要这样做呢？**因为电脑计数会非常非常非常快！**从 1 到 150 计数只需要很短的时间，我们的眼睛都来不及反应过来。真的！电脑就是这样。如果我们想看见 UFO 移动的过程，我们不得不减缓电脑的计数过程。Delay(10)简单地告诉 KPL 在每次移动 UFO 时，稍微有所停留。

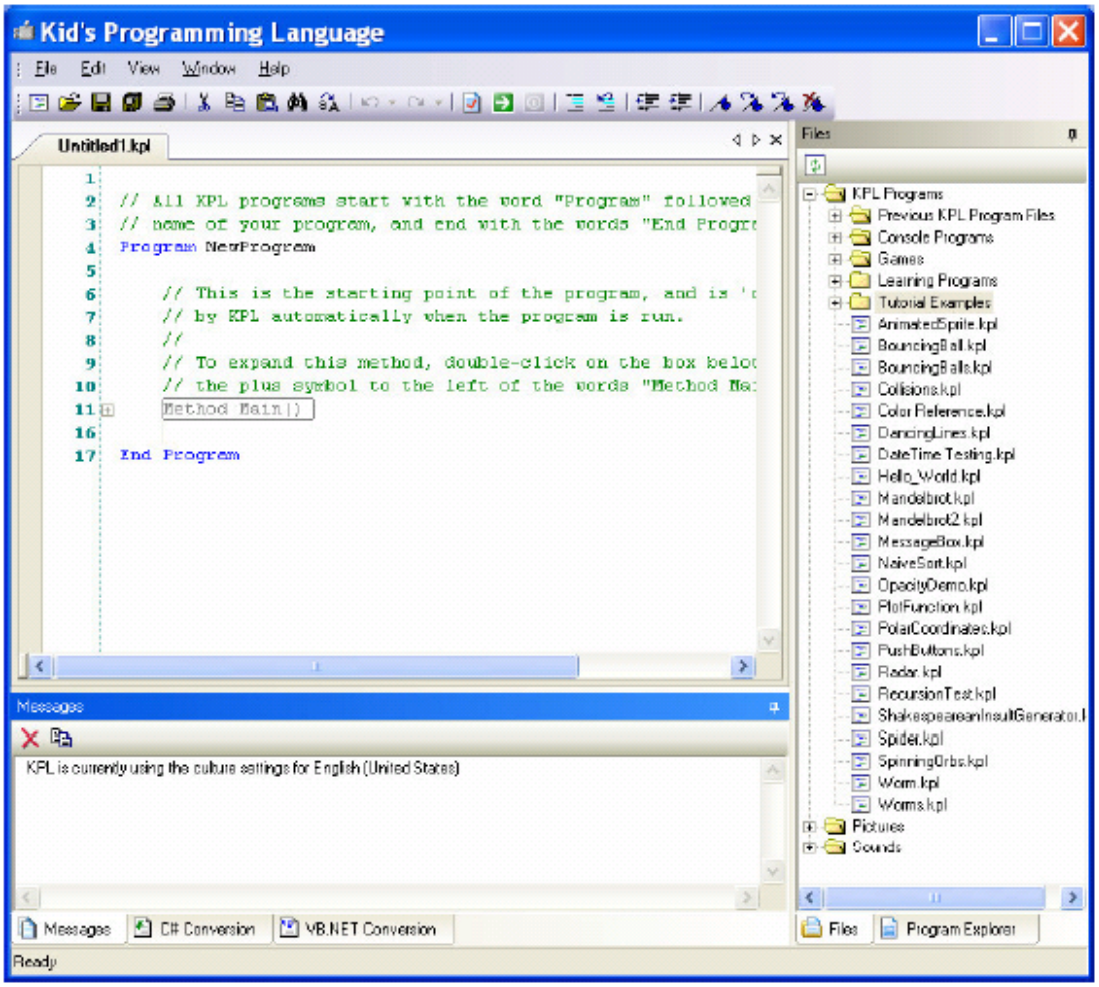
跟我们数秒一样，我们说“1001、1002、1003”那样来减缓计数速度。使用 Delay(10)会减缓 KPL 计数速度。你自己编写 KPL 程序时，

可以改变值为 10 为其他任意值。试试 Delay(2)和 Delay(100)。在 UFO 移动过程中有很大的不同，是不是？

我们花了很多页的解释到程序的细节，使之不象看起来那么复杂。一旦你热衷于 KPL 编程，这 8 条指令不需要花太多的时间输入-希望你能很好的领会这些 KPL 指令，并用它们来做一些更酷的事情！

用 KPL 编程

本章内容基于 KPL 已经安装，重点在于 KPL 的基本使用。可以从 windows 的“开始”处选择“Kids Programming Language”来启动 KPL。启动后的 KPL 界面如下：

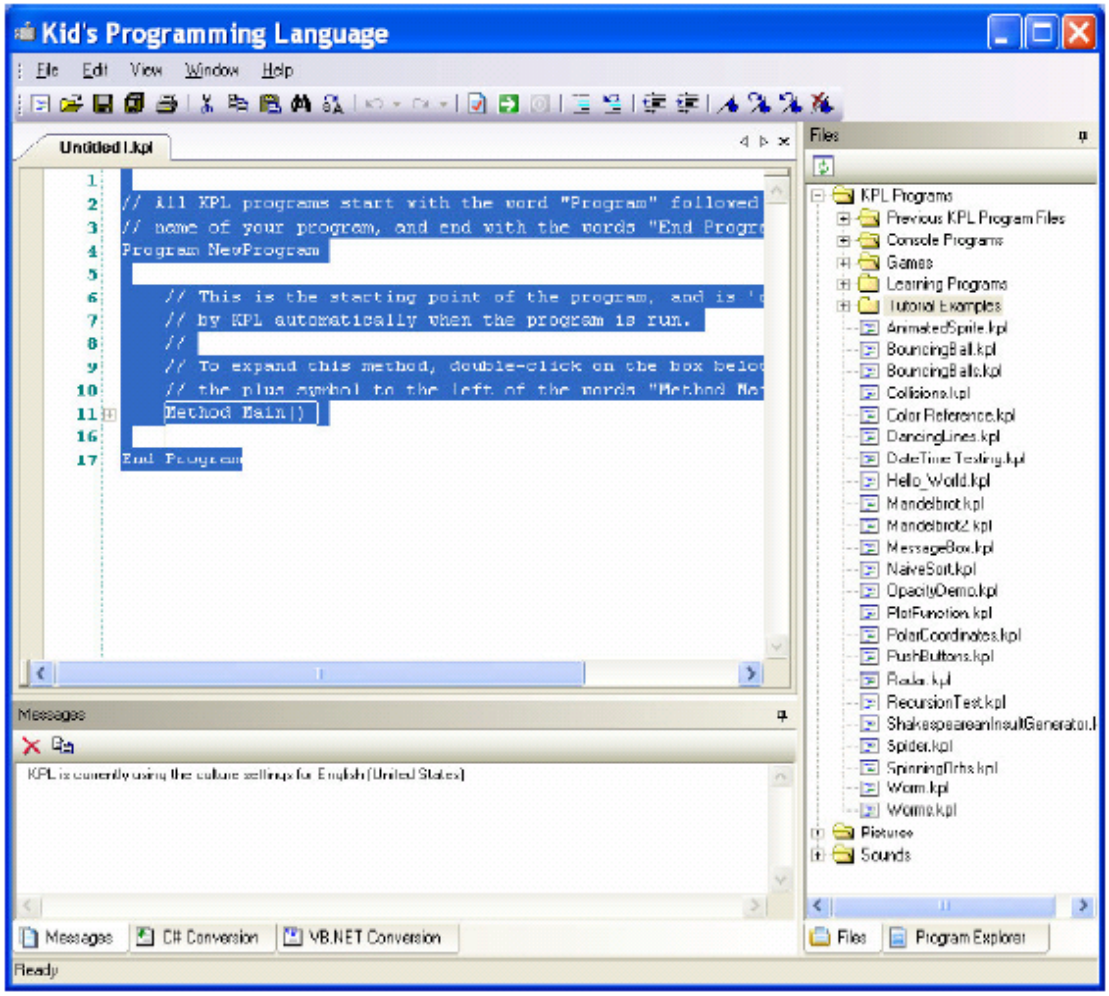


在代码编辑区域，KPL 会默认创建一个名为 Untitled1.kpl 的 KPL 程序文件。在代码编辑窗口内看见的绿色字符是程序注释-便于程序的阅读理解，但不是可以执行的 KPL 指令。注释语句以双斜杠开头：//

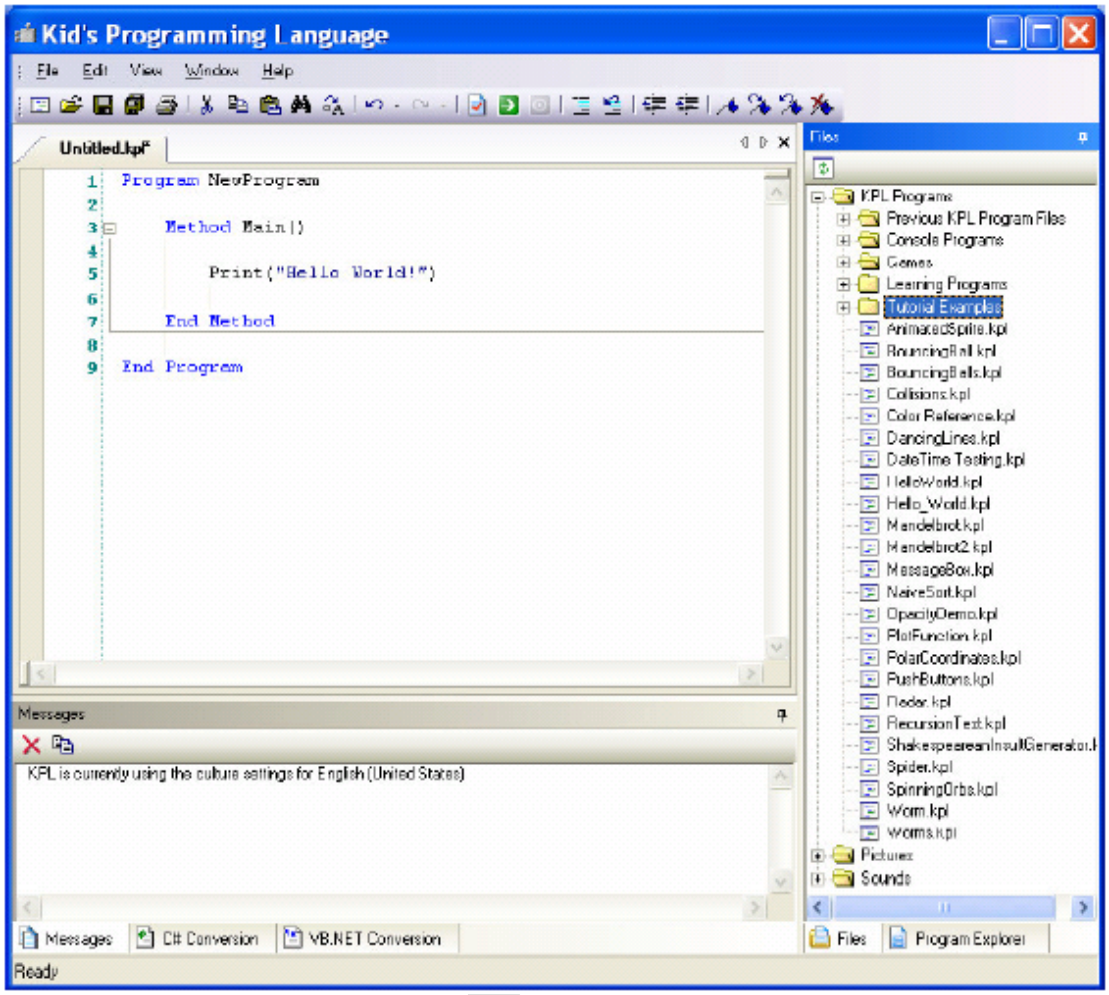
在 KPL 的例程中，有大量的程序注释，有了这些注释的帮助，你可以自学 KPL，并编写自己的 KPL 代码。注释可以帮助其他人理解程序在做什么，当你写过代码很久以后，可以帮助你恢复记忆。

KPL 的代码窗口跟平常所见的文字处理窗口或 email 程序差不多。可以练习一下菜单选项和工具栏。把鼠标指向工具栏图标，会看到相应的功能说明。

我们的重点集中在重建上述所讲的三个示例，因此我们首先要清除 untitled1.kpl 程序里的代码。要完成这个操作，单击 Edit 菜单，然后选择 Select All 菜单项。你会看到代码窗口内的代码全被选中，所有选中的代码都反相高亮显示，你可以按 Delete 键，或者用 Edit 菜单中的 Cut 选项删除 KPL 代码窗口中的所有代码。

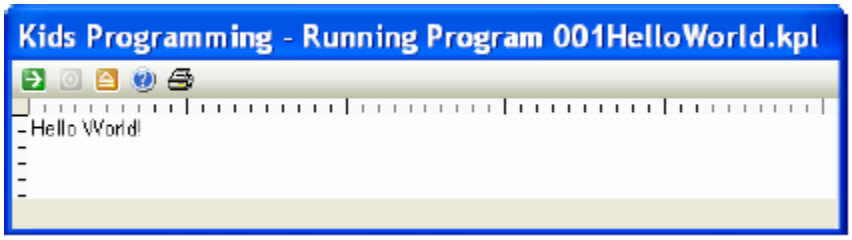


既然代码已经删除了，那么输入我们的第一个 KPL 程序示例，如下：



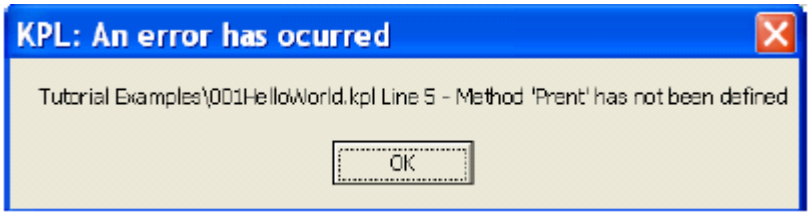
记住，电脑要求得非常严格-因此需要按上面的准确输入程序。如果你输入错误，程序有可能不会正确执行。你可以用 TAB 键来缩进你的 KPL 代码，也可以输入 Enter 键来空行。缩进和空行都不是必须的。但这样的排列有助于你和其他人更加容易地阅读和理解 KPL 程序。

当你录入代码完成后，单击工具栏上的绿色图标：  或按 F5 运行 KPL 程序。如果你的 KPL 程序录入完全正确的话，你会看到如图所示的弹出窗口：



这就是全部了！你现在已经是个电脑程序员了！继续前进，哈哈！好了，好了，这虽然只是一个简单的小程序，然而你输入后，它能运行！

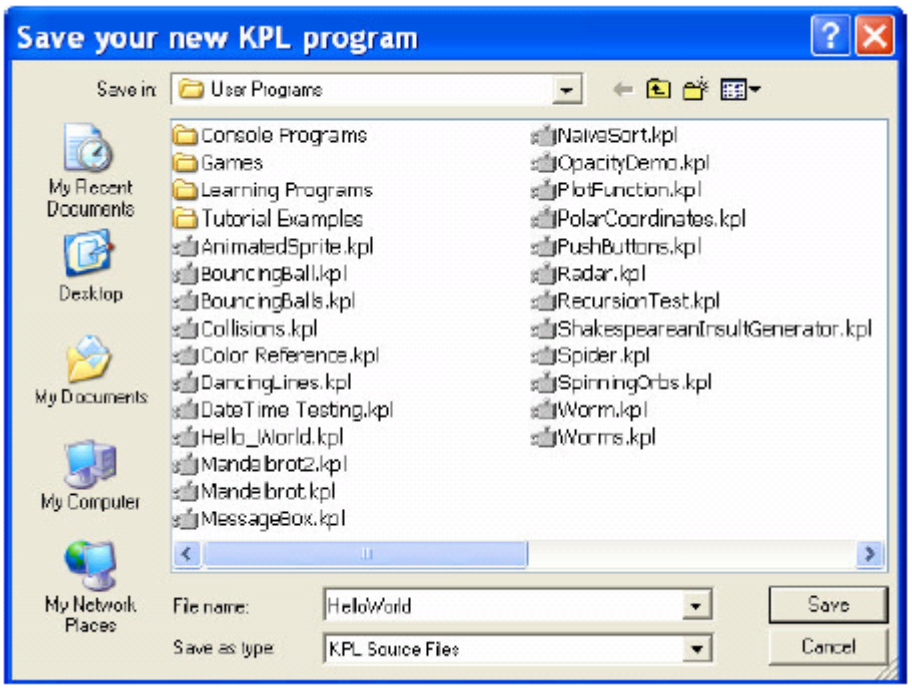
如果 KPL 不能理解你输入的代码，你会看到一个错误提示的窗口：



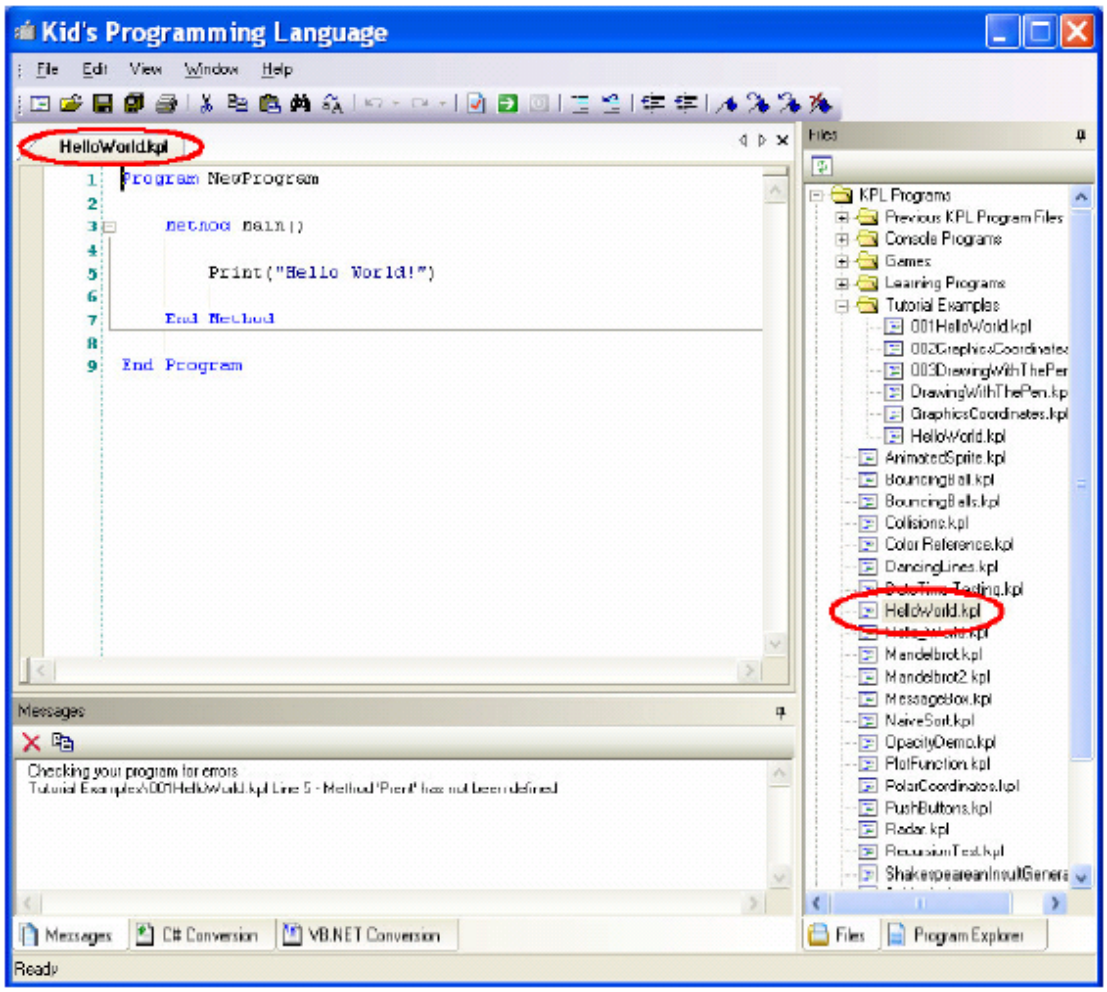
如果这样，请单击 OK 按钮，仔细地比较所输入的代码跟上面提供的屏幕截图中的代码，当你发现不同的地方，改正它，然后再重新运行程序。

所有的程序员都会写出有错的代码。所以如果你也出错，不必灰心，没有人是完美的！当代码出错时，请保持沉着、耐心，并仔细检查你的代码。沉着、耐心和仔细是发现错误和改正错误的最好方法。

完成了 HelloWorld 例程后，单击 File 菜单，选择 Save 选项，在如下的窗口内命名程序为 Helloworld:

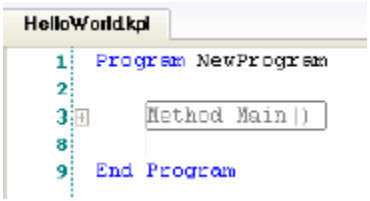


当你保存了 KPL 程序后，可以在下面两处看见所做的改变：



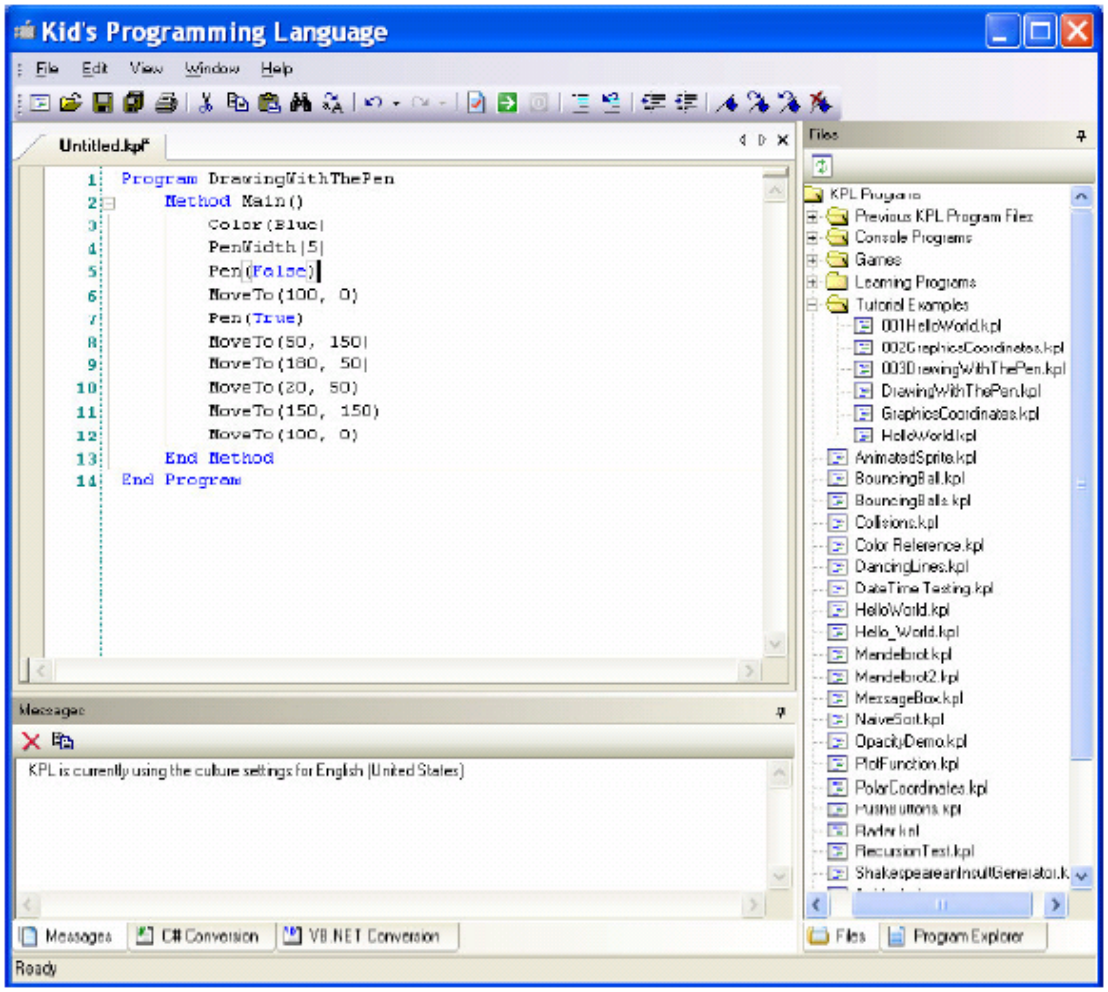
现在你的程序已经保存到磁盘了，你可以通过在 KPL 右边的 Files 文

件夹里双击该文件重新打开它。当重新打开时，你会看到如下的显示-不要担心，你的代码还在那儿！



单击第 3 行代码前面的小图标+，Method Main 里面的代码会全部展开，点击小图标-,代码又会收起来。对一个简单的程序，这个用处不大。但是当编写的代码越来越大时，你会明白用这样的方法隐藏和显示代码对编程者来说非常的方便。

保存好 HelloWorld 后，单击 File 菜单，选择 New Document Window，你会看到新的 Untitled.kpl 程序打开，跟前面的一样。先删除里面所有的代码，录入我们的第二个例程。



如果录入上述正确的代码，会看见蓝色的五星。你成为了电脑图像程序员！**继续前进！**

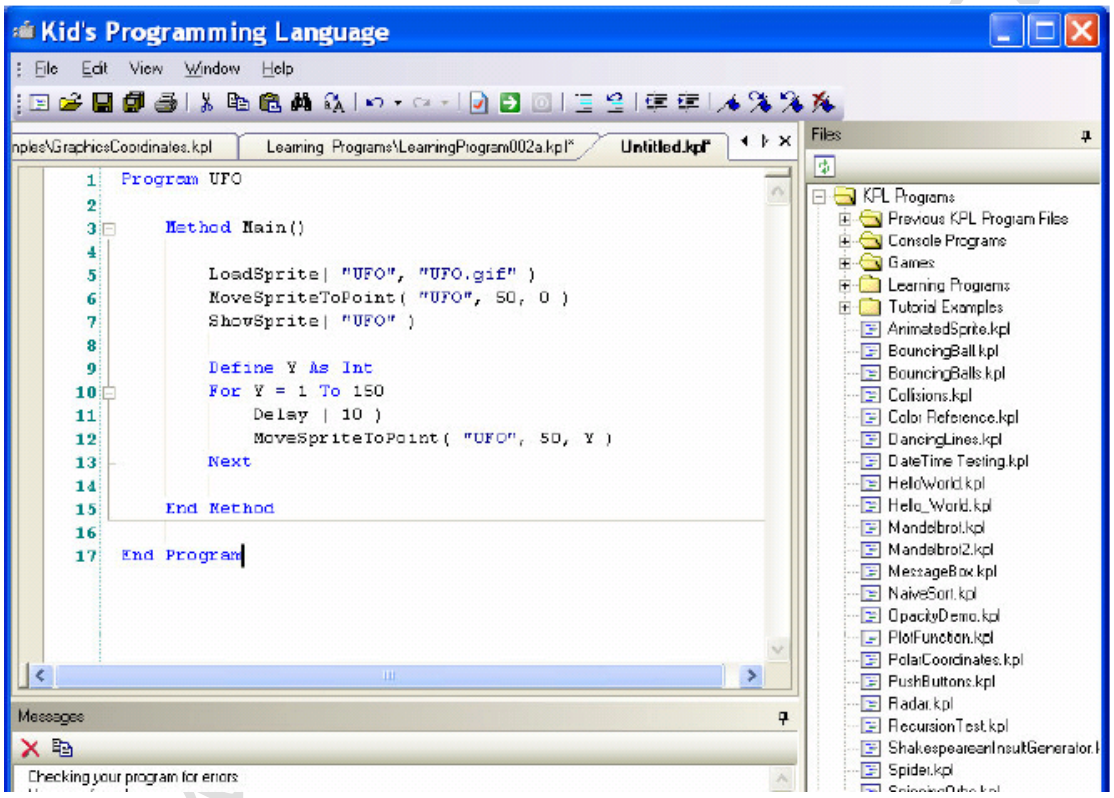
记住，如果出错的话，保持沉着和耐心，仔细检查你的代码和示例代码的不同，你修正好代码，使之正确运行。

完成工作时，千万不要忘记保存你的程序！

这个程序还可以变得更有意思一些。什么颜色看起来更好？如果使画笔变粗或变细，图形的效果又会怎么样呢？你怎么画一个正方形或三

角形，而不是一个五星？还有难一点的:如何做到使五星更大或更小？

保存好 DrawingWithThePen 程序后，单击 File 菜单，选择 New Document Window，新的 Untitled.kpl 文件打开，删除所有的代码，录入我们的 UFO 例程：



如果录入正确，会看到移动的 UFO。**继续前进！**

记住，如果出错的话，保持沉着和耐心，仔细检查你的代码和示例代码的不同，你修正好代码，使之正确运行。

完成工作时，千万不要忘记保存你的程序！

想获得更多的编程实践，让我们来做些有趣的练习：

- 能否使 UFO 移动更远一些？
- 如何使 UFO 从左到右飞，而不是从上到下飞？
- 你能使 UFO 同时向下向右飞吗？

进一步的学习

完成这个教程后，接下来就是看 KPL 里的 Learning Programs 文件夹里的内容。下一步将学习 6 个 KPL 程序-最好是按照顺序学习。事实上，这 6 个程序的前 2 个跟我们在本教程中讲的例程非常相似！当你读到此处时，可能 Learning Programs 里的程序已经更新了-这些程序作为安装的一部分存在，也可以从下载页面获得：www.kidsprogramminglanguage.com(注：中国国内用法暂时无法访问该站点)

或许你还应该下载 KPL 的**教师用户手册**，在同样的下载页面。该手册不是为初学者编排的，但可能对你同样有用，你可以把里面的例程移到自己的程序中来。

当你熟悉了 Learning Programs 的程序后，在 KPL 里面还有更多的其他的程序例程，可以供你学习和练习使用。Kplong.kpl 和 NumberGuess.kpl 是继续学习的很好的例程。这些代码在 Games 文件

夹中。都是代码较多，功能丰富的优秀程序！-但是当你理解了本教程和 Learning Programs 里的例程后，学其他的程序没有任何问题。

以下是 KPL 在线社区的链接 - 也可以从 www.kidsprogramminglanguage.com 站点上查到：

Cool KPL programs

Kids’ Discussion of KPL

Parents’ discussion of KPL

Teachers’ discussion of KPL

KPL Questions and Answers

International Language Version of KPL

如果你有很多问题或需要帮助的话，KPL Questions and Answers 社区是该首先去看的地方。在线社区也相当的活跃，而且也正变得越来越活跃！