Beginning Programming With Phrogram

By Jon Schwartz

Document updated September 5, 2006

website: www.phrogram.com

Beginning Programming With Phrogram

Table of Contents

Introduction: What is Programming?

Computer programming is, simply put: **giving instructions to a computer.**

Computers are very good at following instructions. They do exactly what we tell them to do. But they have **no imagination!** And so when we write programs to give computers instructions, we must tell them very precisely what we want them to do.

Why Should I Learn to Program With Phrogram?

Different computer languages give you, the programmer, different ways to tell your computer what you want it to do. Phrogram has several important advantages for you as a beginner:

- Phrogram was carefully designed to make it **as easy as possible** for a beginner to learn.
- Phrogram was carefully designed to make it **as fun as possible** for you to learn
- Phrogram, unlike other learning languages, is also carefully designed to make it **as similar as possible** to the languages which are used by professional programmers today

Beginning programming with Phrogram is the easiest way to learn real programming – and many of you who want to program fun things will find that Phrogram does everything you need it to do, better than any other language or environment does. But if you want to graduate to other kinds of programming, like "enterprise development", you'll find that Phrogram has prepared you well to move on to languages like Java or C# or VB. Phrogram is carefully designed to be as much as possible like those languages, and their programming environment. Except, of course, Phrogram is much easier, and much more fun!

How Should I Use This Tutorial?

If you are truly a beginner, and have never done any programming, the best way to use this tutorial is to read and study this tutorial section by section, in order, making sure you understand each section before moving on to the next. It will probably be best to study the tutorial without using Phrogram on your computer, until you get to the section of the tutorial **"Using Phrogram on Your Computer"**. Studying and learning Phrogram this way will help you avoid being distracted by details which you will get to at the end of the tutorial.

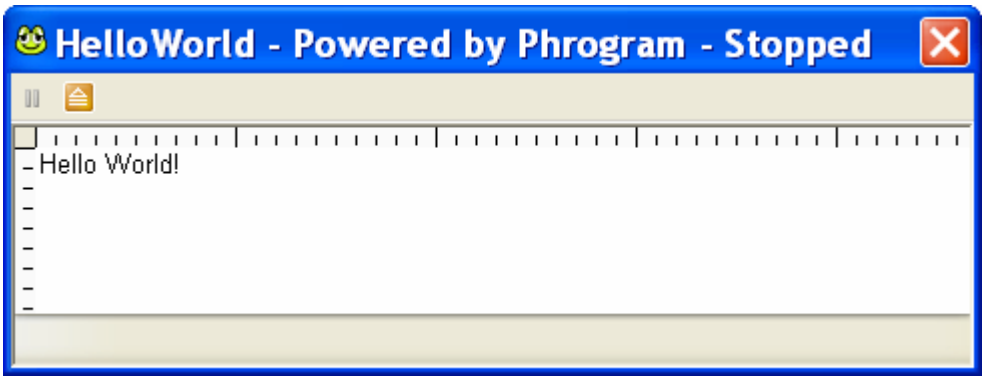
Computer programming involves learning to think in ways people do not normally think. Computers require us to be much more logical, orderly and precise than we normally need to be! Learning this may be difficult at first, but you can do it! And once you get the hang of it, it really does become fairly easy.

One of the best ways to learn that new way of thinking is to ask questions of or get explanations from someone who already understands computer programming. Can you think of anyone who could help you with questions and answers as you study programming with this Phrogram tutorial?

OK, Show Me a Program!

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

Well, that's it! When you run that Phrogram program, this is what you will see:



The 5 lines of Phrogram code you see above are a screenshot of what the code looks like in Phrogram, so you can see it here exactly as you will see it when you are programming in Phrogram.

The first thing I should mention is that the line numbers you see to the left of each line of code are not actually used by Phrogram – they are just there for your own information as you read and work with your Phrogram programs. Older languages, such as GWBASIC, actually used line numbers to process code in the order desired by the programmer. Phrogram does not do this. Phrogram programs are normally processed one instruction at a time, starting at the top and moving downward through the program. There are exceptions to this, which we won't discuss in this beginner's tutorial.

One important rule for programming in Phrogram is that each line of Phrogram code needs to be on a separate line in your program file. For instance, this Phrogram program has the same code in it, but not on separate lines, and so this program will not work:

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

All computer languages have some rules which the programmer has to follow, so that the computer will understand the programmer's instructions. Person-to-person communication has lots of rules, too, that serve the same purpose – it's just that we are so used to those rules that we follow them without having to think about them. For instance, we don't say **"Goodbye"** when we pick up the phone! And we don't say **"Hello"** when we hang up the phone! Sure, it's a silly example, but it's very relevant, since Phrogram also requires a specific way for beginning and ending a program. All Phrogram programs must begin with a line like **`Program HelloWorld`** as shown in line 1. And all Phrogram programs must end with **`End Program`** as shown in line 5:

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

You can name your program anything you want, but it's best to name it something which describes what the program is doing. In this case **HelloWorld** is the name I chose. You could just as easily use a different name, like **Program MyFirstProgram.**

Method Main() is the next important thing to know about Phrogram programming. All Phrogram programs start by processing the first instruction in **Method Main()**. Take a look below and you will see that instruction is **Print("Hello World!").**

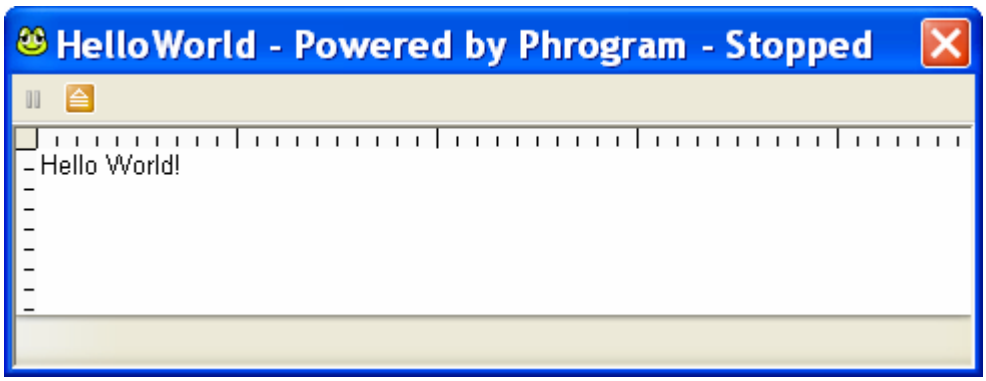
Method Main() is a bit of an arbitrary way to define where the program begins processing instructions – but it is, in fact, based on the way all modern programming languages work. As you might expect logically, **End Method** matches and declares the end of **Method Main().**

We won't explain Methods further in this tutorial, but for now, it will hopefully make sense when you look at the code below that there is only one instruction **"in" Method Main(),** and that is the line **Print("Hello World!").**

All this was a detailed explanation of why this particular program contains only one actual instruction, the one which tells the computer to **Print("Hello World!").** So, let's summarize: Lines 1 and 5 tell the computer that they are the beginning and the end of this Phrogram program. And lines 2 and 4 tell the computer that they are the beginning and end of **Method Main().**

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

Now let's look again at the window which appears when you run this program. Notice that "inside" the window, the computer has indeed done only one thing, the thing we told it to do with this Phrogram program. The computer has displayed the words **Hello World!**



The Phrogram program window adds the word "Stopped" to the title bar when the program has finished running. To close that Phrogram program window, you click the orange button on the toolbar, with the Eject icon: 

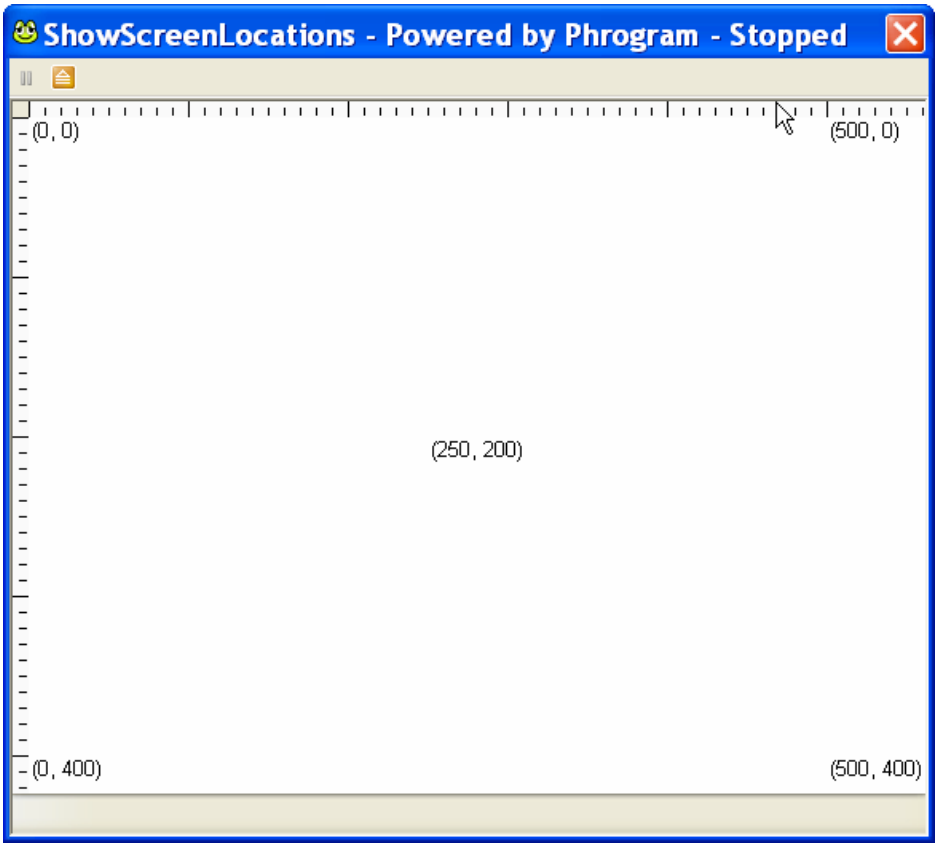
What about Computer Graphics?!

Hello World! is a classic first program – but it's not very exciting! So let's talk about graphics.

An explanation and example of computer graphics has to start with where computers place graphics on the screen. Computers use a **coordinate system** which is different from the **algebra** coordinate system which we all learn in school. But actually, the computer coordinate system can be easier to work with, particularly because it makes it simpler to work with locations on the computer screen.

Computers use an **(X, Y)** coordinate system for locations on the computer screen in which **the left edge of the screen defines X = 0**, and **the top of the screen defines Y = 0**. This means that the **origin**, where **X = 0** and **Y = 0,** is the **upper left corner** of the screen. Moving to the right across the screen increases the X value, and moving down the screen increases the Y value.

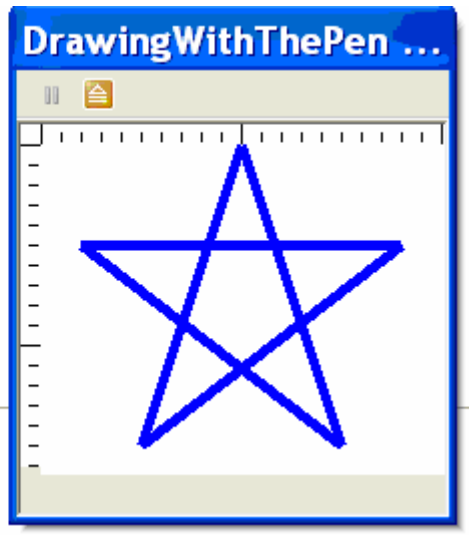
If you are not used to thinking about coordinate systems, this is a complicated explanation, so here is a picture which shows where several **(X, Y)** locations are displayed by a Phrogram program:



Please take a moment to examine the numbers in the picture above. The **first value** of each pair is the **X value** of that location on the screen – and as you can see, the X value increases as you move from the left to the right. The **second value** of each pair is the **Y value** of the location on the screen – and as you can see, the Y value increases as you move from the top to the bottom.

Let's write our first Phrogram graphics program, and in the process see exactly how you use the graphics coordinates system to make cool graphics on the screen with Phrogram. We'll start by showing the full Phrogram program, as well as the graphics it creates when you run it. And then we will step through the Phrogram program in detail, to explain exactly how it works:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen
4          myPen.LineWidth = 5
5          myPen.Color = Blue
6          myPen.MoveTo(100, 0)
7          myPen.DrawTo(50, 150)
8          myPen.DrawTo(180, 50)
9          myPen.DrawTo(20, 50)
10         myPen.DrawTo(150, 150)
11         myPen.DrawTo(100, 0)
12     End Method
13 End Program
```



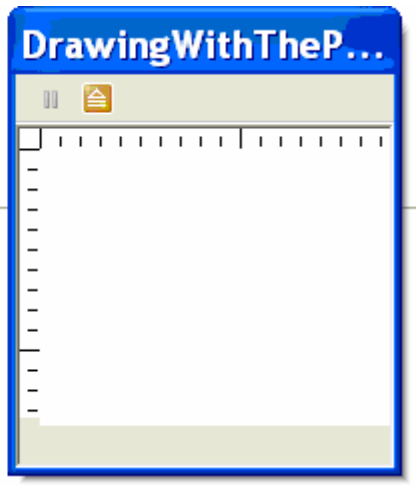
Notice there are now 9 Phrogram instructions in **`Method Main()`**, from line 3 to line 11. That's a lot more than our first example, but hopefully you agree it's cool that Phrogram can draw a blue star on the computer screen like that with only 9 instructions!

Notice that the program begins and ends in the same way our first program did, except that we have named this program **`DrawingWithThePen:`**

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

Now let's start adding our instructions to **`Method Main()`**, to examine what they each do:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen:
4          myPen.LineWidth = 5
5          myPen.Color = Blue
6      End Method
7  End Program
```



Our first three instructions have been added to our program, but as you see, the program doesn't display any graphics yet. The first instruction we have added is **`Define myPen As Pen.`** We will wait until later to explain this more carefully, but for now, just read this literally: "I am defining myPen as a Pen." Think about this logically: if I tell Phrogram that myPen is a Pen, then Phrogram will understand that, and will know the myPen can do things that Pens do. What are some things pens can do? Well, pens can Draw, and they can have different colors, and some pens draw thinner lines than other pens, right? This is as true with Phrogram's pens as it is with the pens you use to write on paper. So let's see how to do each of those things with a Pen in Phrogram,

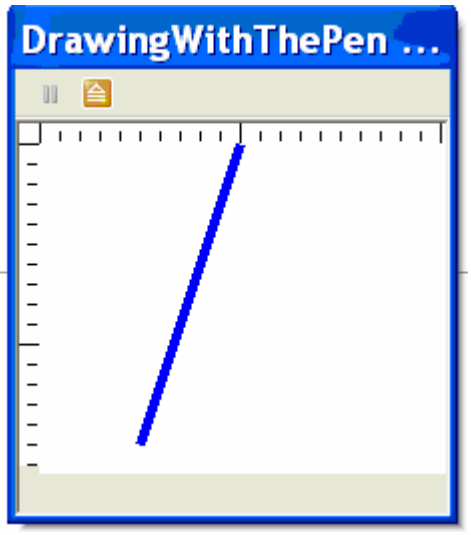
Let me remind you of the next two instructions, and then I will explain those as well:

```
Define myPen As Pen:
myPen.LineWidth = 5
myPen.Color = Blue
```

Can you guess what **`myPen.Color = Blue`** does? I bet you can. It tells Phrogram that we want it to draw using the color Blue. **`myPen.LineWidth = 5`** tells Phrogram that when we draw with myPen, we want Phrogram to draw a line which is 5 pixels wide. A pixel is one tiny dot on your computer screen, so a line which is 5 pixels wide will look a little wide, as if you drew it with a magic marker. You could, of course, use a different value for LineWidth. **`myPen.LineWidth = 2`** would draw a thinner line, like a blue ballpoint pen. And **`myPen.LineWidth = 10`** would draw a thicker line, like a crayon.

So far, we have told Phrogram about **how** it will draw, but we haven't told it to draw anything yet. Now, let's draw our first line in the star:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen
4          myPen.LineWidth = 5
5          myPen.Color = Blue
6          myPen.MoveTo(100, 0)
7          myPen.DrawTo(50, 150)
8      End Method
9  End Program
```

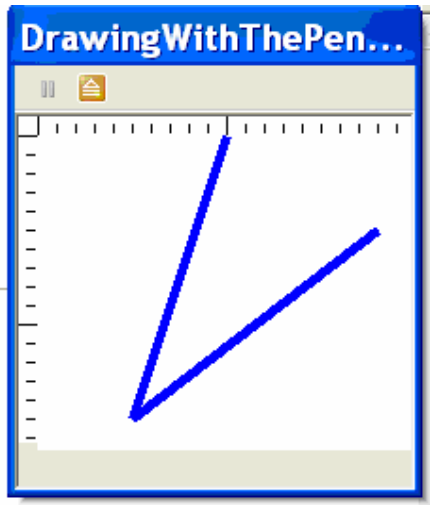


The first instruction we added is **`myPen.MoveTo(100, 0)`**. This tells Phrogram to move the pen to location **(100, 0)** on the screen, which is the top of our star. When you define a new pen in Phrogram, it always starts at location **(0, 0)** – the upper left corner. Using the MoveTo command tells Phrogram to move your pen, but NOT to draw a line as it moves.

The second instruction we added is **`myPen.DrawTo(50, 150)`**. This instruction actually moves your pen while draws the first line in our star. Consider the **(X, Y)** values for these two points for a moment. Since this instruction tells Phrogram to draw from **X = 100** to **X = 50**, the line moves to the left. And since we move from **Y = 0** to **Y = 150**, the line moves down the screen.

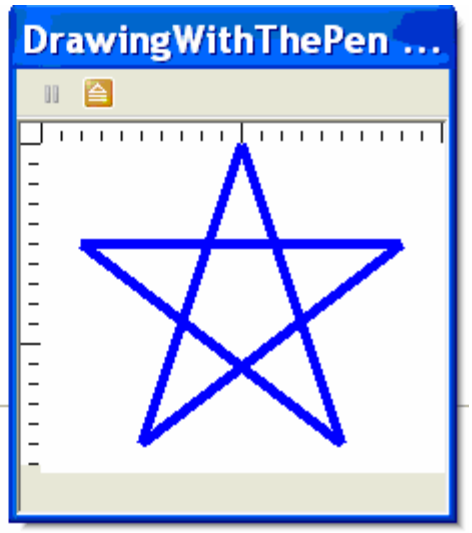
Now let's add one more instruction to Phrogram, which adds the next line to the star:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen
4          myPen.LineWidth = 5
5          myPen.Color = Blue
6          myPen.MoveTo(100, 0)
7          myPen.DrawTo(50, 150)
8          myPen.DrawTo(180, 50)
9      End Method
10 End Program
```



The instruction we added is **`myPen.DrawTo(180, 50)`**. Phrogram continues moving the pen from the previous point, which was **(50, 150)**. Have you ever seen an Etch-A-Sketch – those cool red toys which allow you to draw things on the screen by turning two knobs? Phrogram's pen is a lot like a computerized Etch-A-Sketch. Let's add the remaining lines which will finish the star now:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen
4          myPen.LineWidth = 5
5          myPen.Color = Colors.Blue
6          myPen.MoveTo(100, 0)
7          myPen.DrawTo(50, 150)
8          myPen.DrawTo(180, 50)
9          myPen.DrawTo(20, 50)
10         myPen.DrawTo(150, 150)
11         myPen.DrawTo(100, 0)
12     End Method
13 End Program
```



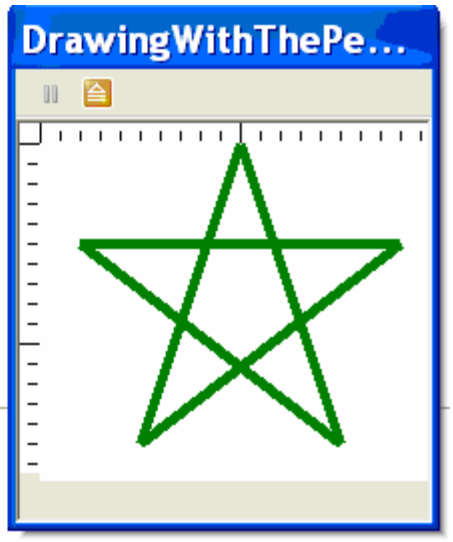
As you can see, three more Phrogram instructions cause Phrogram to draw three more lines. In total, our Phrogram instructions have caused the computer to draw five lines with **`myPen`**, and the result is a blue star.

The Phrogram program required to do this is small – only 9 Phrogram instructions. The hardest thing about this is getting used to the way the **(X, Y)** coordinate system works in Phrogram. At this point, if **(X, Y)** values aren't clear to you yet, you probably should return to the beginning of this section and read through it one more time. If after you do that **(X, Y)** coordinates still don't make sense, you may want to type your own version of this Phrogram program into Phrogram on your computer, and experiment with it by drawing lines to and from different locations on the screen. Perhaps you could practice by drawing a square, and a triangle? If you do that with Phrogram on your computer, you can skip to the section **Using Phrogram on Your Computer**, for help using Phrogram on your computer. After you are comfortable with **(X, Y)** coordinates, you can return to this point in the tutorial.

It's important to spend enough time and effort so that you are very comfortable with how Phrogram handles **(X, Y)** coordinates, since these are the basis of all graphics programming. Other languages handle graphics this same way, so when you learn this with Phrogram, you are learning the most fundamental concept needed for computer graphics programming in any language.

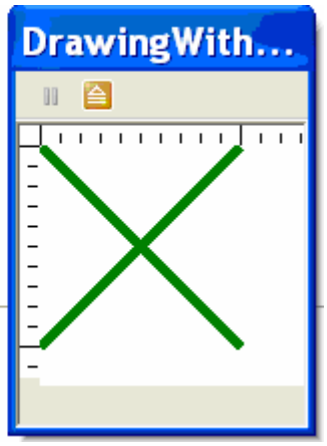
Let's add a final detail to our example, which also shows you an extra bit of control that you have using Phrogram's Pen. You can't control the color of an Etch-A-Sketch, but let's see how easy Phrogram makes that:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen
4          myPen.LineWidth = 5
5          myPen.Color = Green
6          myPen.MoveTo(100, 0)
7          myPen.DrawTo(50, 150)
8          myPen.DrawTo(180, 50)
9          myPen.DrawTo(20, 50)
10         myPen.DrawTo(150, 150)
11         myPen.DrawTo(100, 0)
12     End Method
13 End Program
```



As you can see, we just changed **`myPen.Color = Blue`** to **`myPen.Color = Green`**. Easy! Using **`MoveTo`** and **`DrawTo`** allows you to draw all kinds of objects, and to draw multiple objects – another thing you can't do with an Etch-a-Sketch. Here's a very simple example:

```
1  Program DrawingWithThePen
2      Method Main()
3          Define myPen As Pen
4          myPen.LineWidth = 5
5          myPen.Color = Green
6          myPen.DrawTo(100, 100)
7          myPen.MoveTo(100, 0)
8          myPen.DrawTo(0, 100)
9      End Method
10 End Program
```

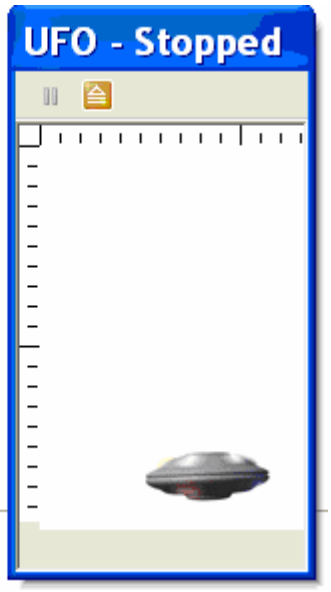


Game Graphics Using Sprites

The point of the drawing example was mostly about explaining how **(X, Y)** coordinates are used for displaying graphics. But game graphics aren't normally drawn. They are loaded from an image file, like an image of a UFO or an asteroid or an elf with a bow. Phrogram uses Sprites as a way to make this **very** easy to do!

As we did with the last program, we are going to start by showing you the finished example, and then explain in detail how this program works, so that you can write programs like this yourself. Here is the full Phrogram program – which if you look inside **`Method Main()`** has exactly 9 instructions, just like the last example. These 9 instructions, though, tell Phrogram to display a UFO, and move it slowly down the screen:

```
1  Program UFO
2
3      Method Main()
4
5          Define myUFO As Sprite
6          myUFO = LoadSprite( "UFO", "UFO.GIF" )
7          myUFO.MoveTo( 50, 0 )
8          myUFO.Show()
9
10         Define ufoY As Integer
11         For ufoY = 1 To 150
12             Delay(10)
13             myUFO.MoveTo( 50, ufoY)
14         Next
15     End Method
16
17 End Program
```



Notice that this program begins and ends the same way as our previous examples – and as all Phrogram programs have to – except that we have named this one **`UFO:`**

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

Let me show you some examples of image or picture files which are included with Phrogram. You can use **ANY** image file you want to use with Phrogram, including ones you add or create – these are just a few of the dozens of images which are included with Phrogram:

UFO.GIF

QUAD.GIF

PLANE.PNG

SPIDER.PNG

BLUEBALL.PNG

Now that you've seen some examples of the kinds of graphics you can use with Phrogram, let's start by adding the Phrogram instructions which cause the UFO to be displayed at the top of the screen:

```
1  Program UFO
2
3      Method Main()
4
5          Define myUFO As Sprite
6          myUFO.Load( "UFO.GIF" )
7          myUFO.MoveTo( 50, 0 )
8          myUFO.Show()
9
10     End Method
11
12 End Program
```

The first instruction we added is **`Define myUFO As Sprite`**. In computer programming, a **`Sprite`** is a graphical object which you can display and manipulate on the computer screen – like our UFO! With this first instruction, we are telling Phrogram that **`myUFO`** is one of those graphical objects.

The second instruction we added is **`myUFO.Load( "UFO.GIF" )`**. As I mentioned before, Phrogram has some rules about how we give it instructions. The rules about using **Load** are not difficult, but they are exact! **Load** requires only one **"value"** for it to work: it requires the name of the image file which Phrogram will use for this sprite – in our case, **`"UFO.GIF"`**. This value must be surrounded by double quotation marks, as shown.

Here are two examples which show **incorrect** ways of telling Phrogram to load this Sprite. Let me repeat: **these two instructions will not work** because they do not follow Phrogram's rules about calling **Load**. Can you figure out what to change to make these work?

```
myUFO.Load( 'UFO.GIF' )
myUFO.Load( UFO.GIF )
```

> **Summary:** **`myUFO.Load( "UFO.GIF" )`** tells KPL to use the picture from file **`"UFO.GIF"`** when it displays myUFO on the screen

Phrogram now needs to know where to put the Sprite on the screen, so we give it the instruction **`myUFO.MoveTo( 50, 0 )`**. **MoveTo** requires two values. First is the **X axis location** for the sprite, which we want to be **50**. Second is the **Y axis location** for the sprite, which we want to be **0**.

An important detail to notice is that **there are no quotation marks around the numeric values**. In general, Phrogram **requires** quotation marks around values that are **words**, and Phrogram **does not require** quotation marks around values that are **numbers**. This is not obvious or intuitive for beginners, but other computer languages also use this same rule. See the **Phrogram v 2 User Guide** section on Data Types for more detailed information about this.

> **Summary:** `myUFO.MoveTo( 50, 0 )` tells KPL to move our UFO sprite to the screen location **(50, 0)**.

All that remains is to tell Phrogram to display the sprite. That's the easiest inastruction yet: `myUFO.Show()`. That's it! Phrogram waits until you actually tell it to **Show** the UFO before it displays it on the screen – this allows you to set the sprite up carefully before showing it to the user.

That was a very detailed explanation, so let's get back to showing how simple it actually is in Phrogram. Here's the Phrogram program, and what it looks like when we run it. Can you look at the instructions and see which one tells Phrogram **how** to display your sprite? And can you see which instruction tells Phrogram **where** to display your sprite? How about the instruction that tells Phrogram **when** to display your sprite?

```
1  Program UFO
2
3      Method Main()
4
5          Define myUFO As Sprite
6          myUFO.Load( "UFO.GIF" )
7          myUFO.MoveTo( 50, 0 )
8          myUFO.Show()
9
10     End Method
11
12 End Program
```

Using Variables and Loops in Phrogram

Now we want our UFO to "land" by moving down the screen to the bottom of the window. In the process, we are going to define and use a **"variable,"** so that you can understand what they variables see how to use them in a Phrogram program.

The term variable is very descriptive – it means that the value can change over time. The value changing over time is the real power of using a variable, as you will see in this example. Variables should be defined with a name that makes sense given how we are going to use them. In this case, our variable is going to be used to change the Y-axis location of the UFO, to make the UFO move down the screen and "land," so we will logically call the variable **`ufoY`**.

Here is the line of code defining our variable:

```
Define ufoY As Integer
```

`Define` is a Phrogram keyword that lets Phrogram know you are defining a new variable for Phrogram to use. **`ufoY`** is the name we are going to use for the variable. **`As Integer`** tells Phrogram that **`ufoY`** is being defined as an Integer variable. Integer variables hold numeric values, like **-1** or **0** or **43**. Integer values are whole numbers only.

We know that our UFO starts at location **(50, 0)** on the screen, because that is the point where we told Phrogram to move it. How do we make it move down the screen? We do that by increasing its **Y-axis location** – to **(50, 1)** then **(50, 2)** then **(50, 3)** then **(50, 4),** etc… It's up to us how far we want it to go, so let's decide that the UFO will stop when it gets to **(50, 150)**.

Do you see the pattern there? The **X** value of the sprite's location is always **50**. The **Y** value of the sprite's location is going up by **1** pixel at a time, from **0** all the way to **150**. It's important for you to see the pattern before we show you how to use that pattern in Phrogram code – so if the pattern doesn't make sense, please read the last paragraph again.

Here is the Phrogram code which tells Phrogram to increase the value of the variable **`ufoY`** from 1 to 150, 1 step at a time:

```
For ufoY = 1 To 150
      myUFO.MoveTo( 50, ufoY )
Next
```

This is your first **"loop."** Loops are another of those new concepts which you will need to learn if you have not programmed before – but once you understand the concept, loops are easy. This loop starts with the value of **`ufoY = 1`**. Since we also said **`To 150`**, we know the loop will stop after it gets to the value of **`ufoY = 150`**. Notice the keyword **`Next`**, which defines the end of the loop. When Phrogram gets to the keyword **`Next`**, Phrogram knows that it is time **increase the value of `ufoY`** to the next value – from 1 to 2, or from 2 to 3, or from 3 to 4, etc…, all the way to from 149 to 150.

Basically, we are telling Phrogram to count from 1 to 150, and we want Phrogram to use our variable `ufoY` to keep track of the value as it counts.

And what do we want Phrogram to do while it is counting? We want it to move the UFO down the screen, right? That's what **`myUFO.MoveTo( 50, ufoY )`** tells Phrogram to do.

Since this instruction is "inside" of the **For** loop, Phrogram will perform this instruction **each time it counts** from 1 to 150. This is the real power of a loop! Just counting from 1 to 150 is not so interesting, but if you can **do something useful each time you count,** now **that** lets us do all kinds of cool things, including move our UFO down the screen! Let's examine what is happening:

```
For ufoY = 1 To 150
      myUFO.MoveTo( 50, ufoY )
Next
```

We have already seen how **MoveTo()** works: Phrogram moves **myUFO** to the **(X, Y)** location we give it. What's different this time? Before, we moved the UFO to **(50, 0)**. What happens when we do this in the loop, and we instead use our variable **ufoY**? Remember that the first time through the loop, the value of **ufoY = 1**, and the second time **ufoY = 2**, then **ufoY = 3**, then **ufoY = 4**, etc… all the way to **ufoY = 150**. So, the first time through the loop, Phrogram uses the value of **ufoY=1** to move the sprite, so that the instruction Phrogram performs is effectively:

```
myUFO.MoveTo( 50, 1 )
```

And the next time through the loop, **ufoY = 2**, so Phrogram performs:

```
myUFO.MoveTo( 50, 2 )
```

And so on like this, until Phrogram has used the loop to count all the way to 150:

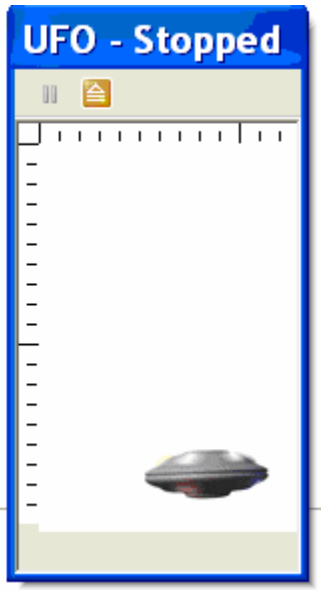
```
myUFO.MoveTo( 50, 3 )
myUFO.MoveTo( 50, 4 )
myUFO.MoveTo( 50, 5 )
…
myUFO.MoveTo( 50, 150 )
```

Each time through the **For** loop, **MoveTo** is moving **myUFO** 1 pixel down the screen – just like we want it to!

So that example explained variables and loops for the first time, and how useful it can be to perform instructions inside a loop – these are all **very important** programming concepts! If they are not clear enough yet, please go back to the beginning of this section and reread it again.

Let's add one final detail to finish our program:

```
1  Program UFO
2
3      Method Main()
4
5          Define myUFO As Sprite
6          myUFO.Load( "UFO.GIF" )
7          myUFO.MoveTo( 50, 0 )
8          myUFO.Show()
9
10         Define ufoY As Integer
11         For ufoY = 1 To 150
12             Delay(10)
13             myUFO.MoveTo( 50, ufoY)
14         Next
15     End Method
16
17 End Program
```



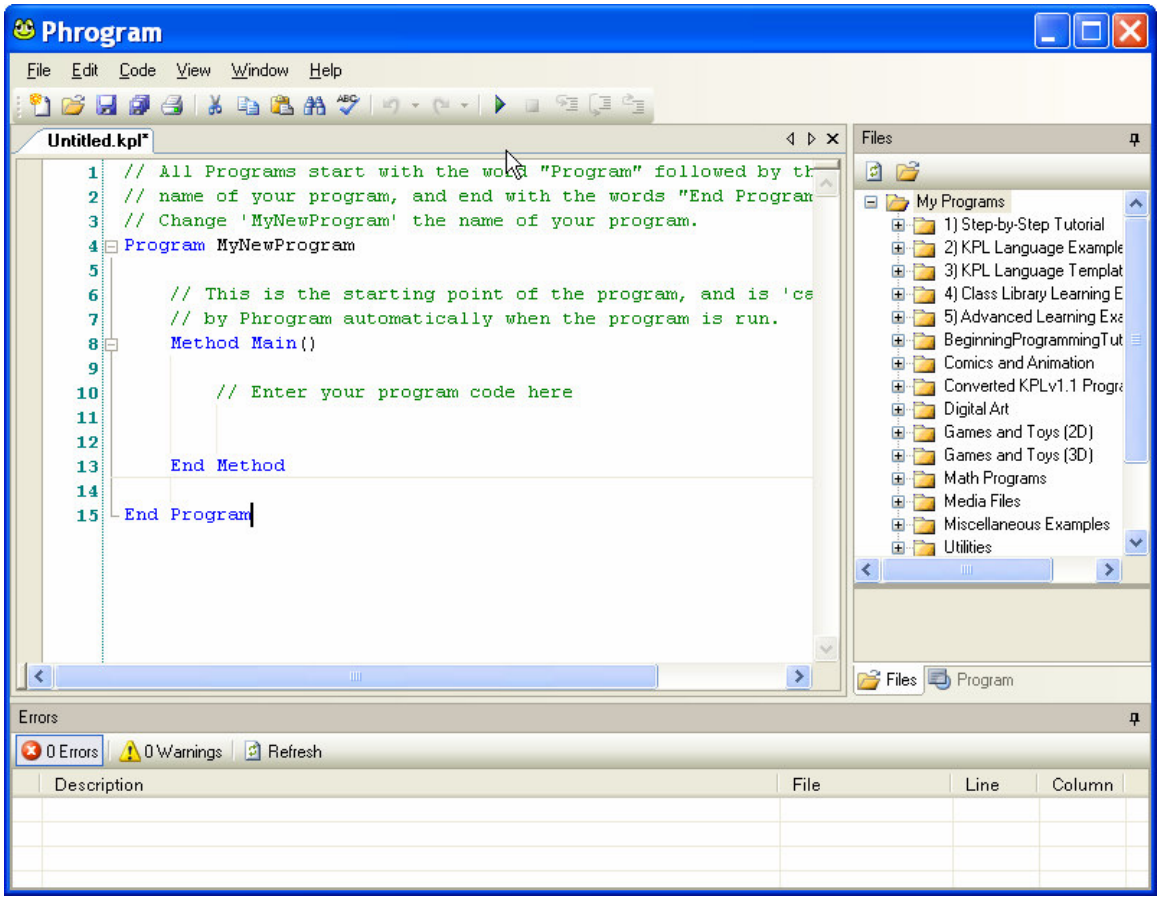
The only new instruction we have added is **`Delay ( 10 )`**, inside the **`For`** loop. Why do we have to do that? **Because computers count very very very fast!** For a computer to count from 1 to 150 takes less time that it takes you or me to blink our eyes. Really! Computers are **that** fast! If we want to actually watch our UFO move down the screen, we have to slow the computer down a little while it counts. **`Delay ( 10 )`** simply tells Phrogram to take a very short break before it moves the UFO each time.

You know how when we want to count seconds like a clock, we instead say "one thousand one, one thousand two, one thousand three" to slow our counting down a little? This is exactly like that. With **`Delay ( 10 )`** we are slowing down Phrogram's counting. When you work on this program yourself in Phrogram, try changing the **`10`** to other values and then run the program. Try **`Delay ( 2 )`** and then try **`Delay ( 50 )`**. Makes a big difference in how the UFO moves, doesn't it?

We spent several pages explaining this program in detail, which makes it seem more complicated than it is. But once you get the hang of programming in Phrogram, these 9 instructions won't take long to type – and hopefully you agree that just a few Phrogram instructions make it easy to do cool graphics with Phrogram!

Using Phrogram on Your Computer

This section assumes Phrogram is already installed, and focuses on getting you started with using Phrogram. Look for and launch Phrogram from the Kids Programming Language item on your Start menu in Windows. When you launch Phrogram, it will look something like this:



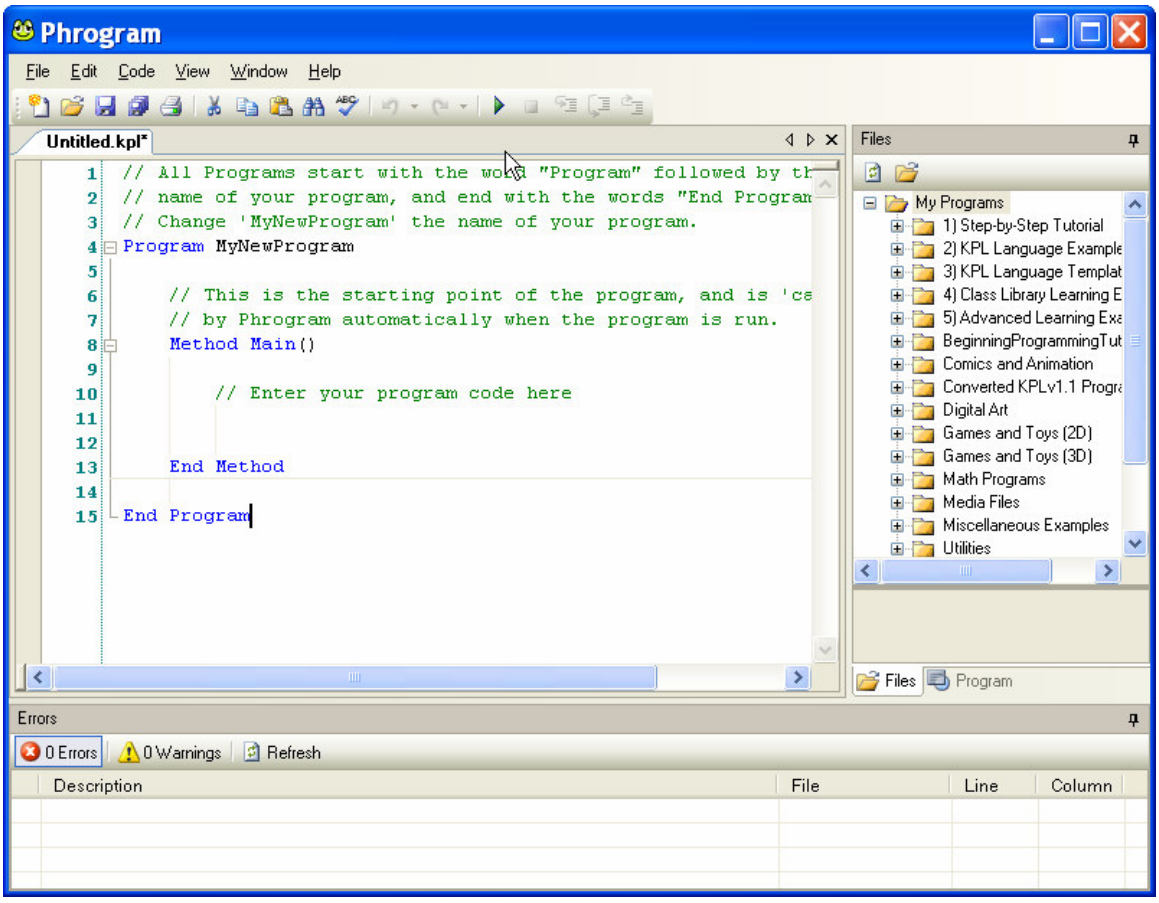
Phrogram is showing a new Phrogram program file called Untitled1.kpl in the Code Editor pane. The green lines you see in the code editor are "comments" which present information to the reader of the program, but which do not actually contain Phrogram instructions. Comments are prefixed by two forward-leaning slash marks: `//`

You will see many comments in the Phrogram example programs, and over time will learn to use comments yourself, as you write your own Phrogram programs. Comments help others to understand what the Phrogram program is doing – and can even help you remember when you look back at a program which you wrote long ago!

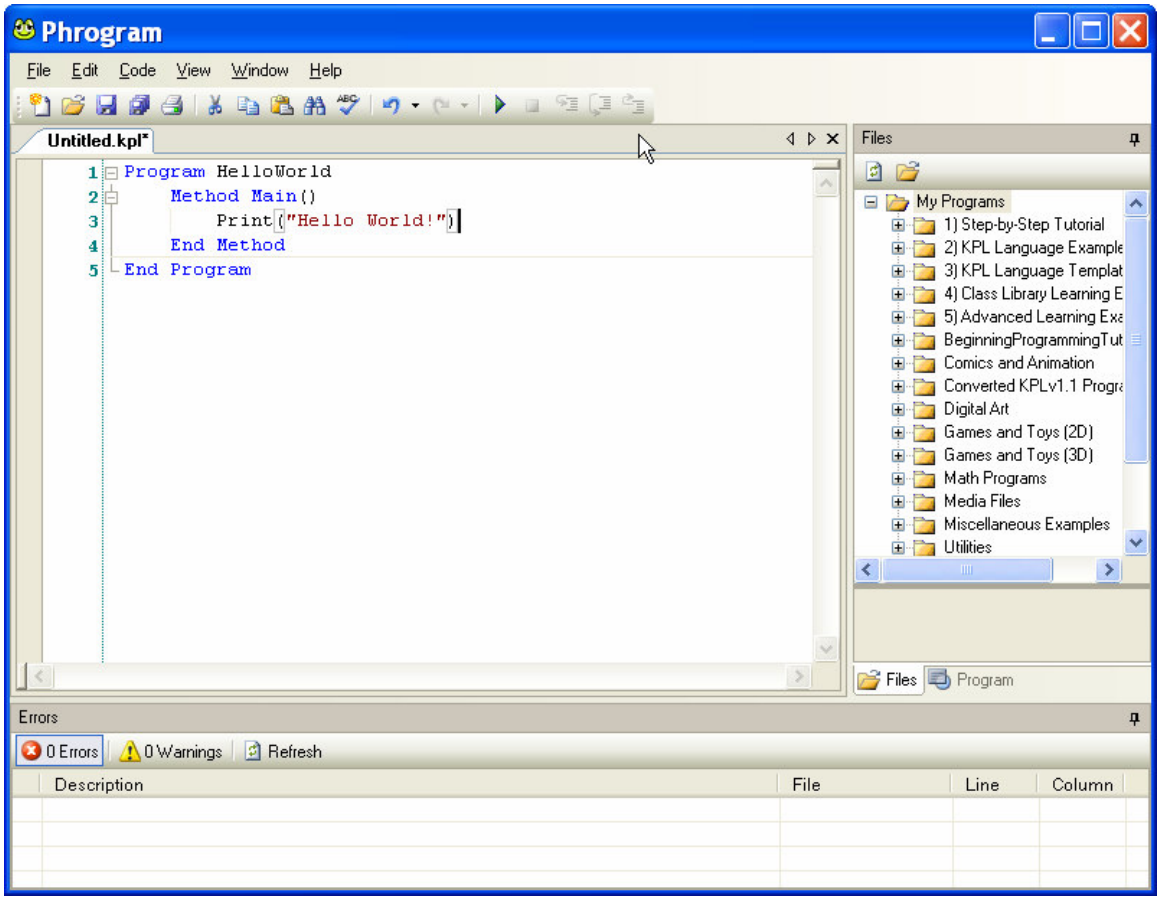
Phrogram's code editor has many of the same features as your word processor or email program. Take a moment to examine the menu items and the toolbar. If you hover the mouse pointer over the toolbar icons, you will see tooltips identifying them.

We want to focus on recreating the three example programs which we created in this tutorial, so let's start by clearing out the code which is shown in the program file Untitled1.kpl. To do this,

click the **Edit** Menu, and the **Select All** item on that menu. You will see that all code in the code editor is now highlighted as shown below. With all of the code highlighted this way, you can either press the **"Delete"** key on your keyboard, or select the **Edit** menu's **Cut** command, and all the existing code will be cleared from the Phrogram code editor.

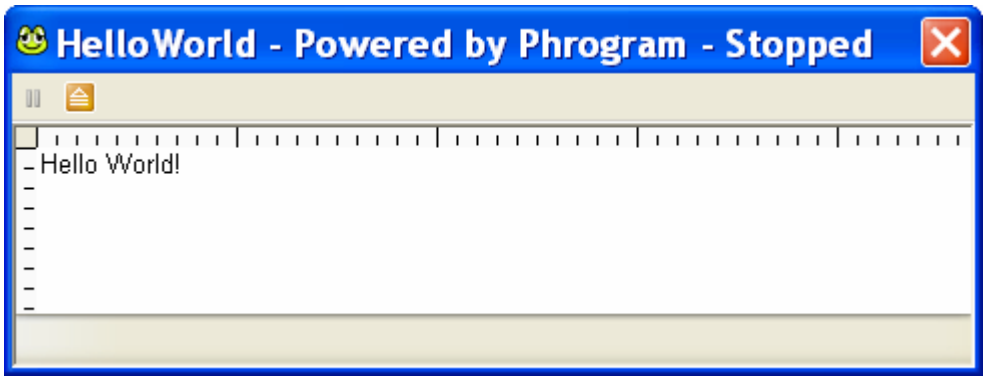


Once the previous code is deleted, go ahead and type in our first Phrogram program example, as shown here:



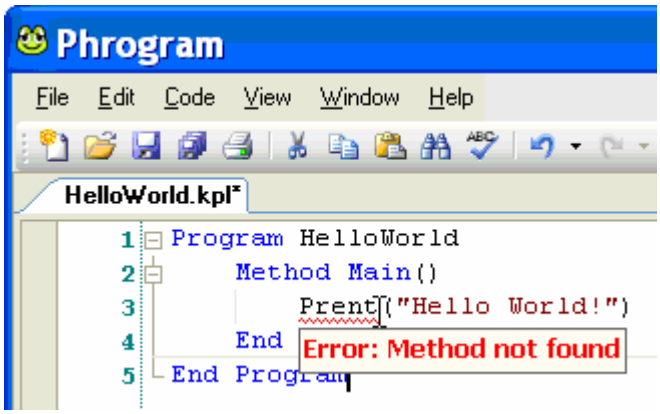
Remember that computers are very precise – so please try to type the Phrogram program exactly as shown above. It may or may not work properly if you type it differently. You can use the **TAB** key to indent your Phrogram code as shown. And you can hit **ENTER** on a line to leave it blank as shown. Using indentation and blank lines is not required, but will help you and others read and understand your Phrogram program more easily.

After you have typed in the program, click on the green arrow icon  or press the **F5** key to run your Phrogram program. If your Phrogram code was entered correctly, you should see a window pop up that looks like this, except it's a bit larger:



That's it! You are now a computer programmer! Way to go! Woohoo! Okay, okay, it's a very small and simple program, but still – you typed that in and made it run!

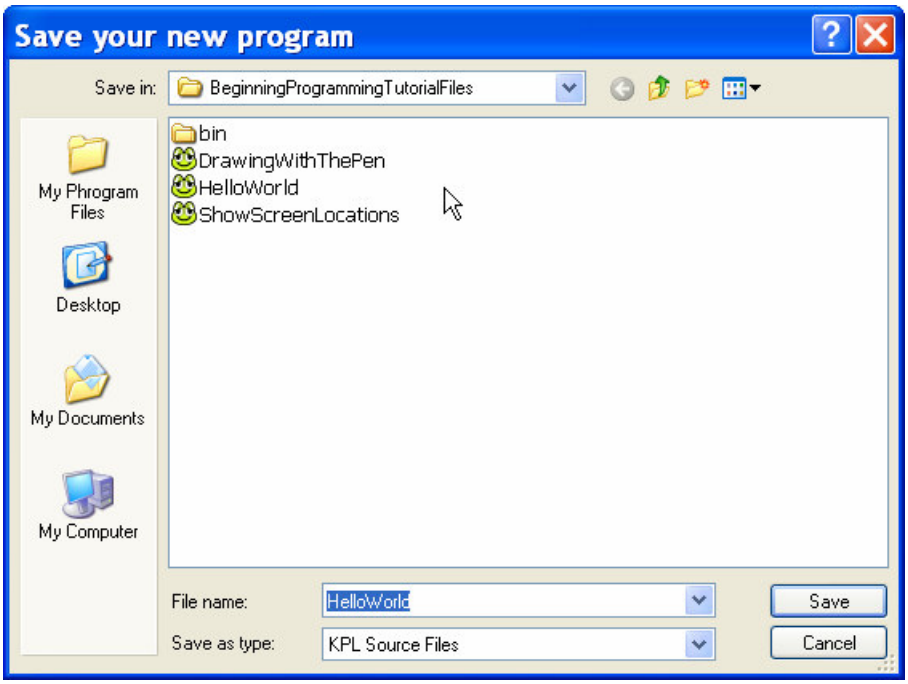
If Phrogram does not understand the code you have typed in, it will show you that it does not understand. Phrogram will put a squiggly red line underneath the code it does not understand. If you put the mouse over the squiggly line, you'll see a popup like this one shown, which says **Error: Method not found**. In this example, I was of course supposed to type **`Print`** but I typed it as **`Prent`**. As I mentioned before, computers are very precise, and they require precise instructions. The computer know what to do if I say **`Print`** – but it doesn't know what to do if I say **`Prent`**!



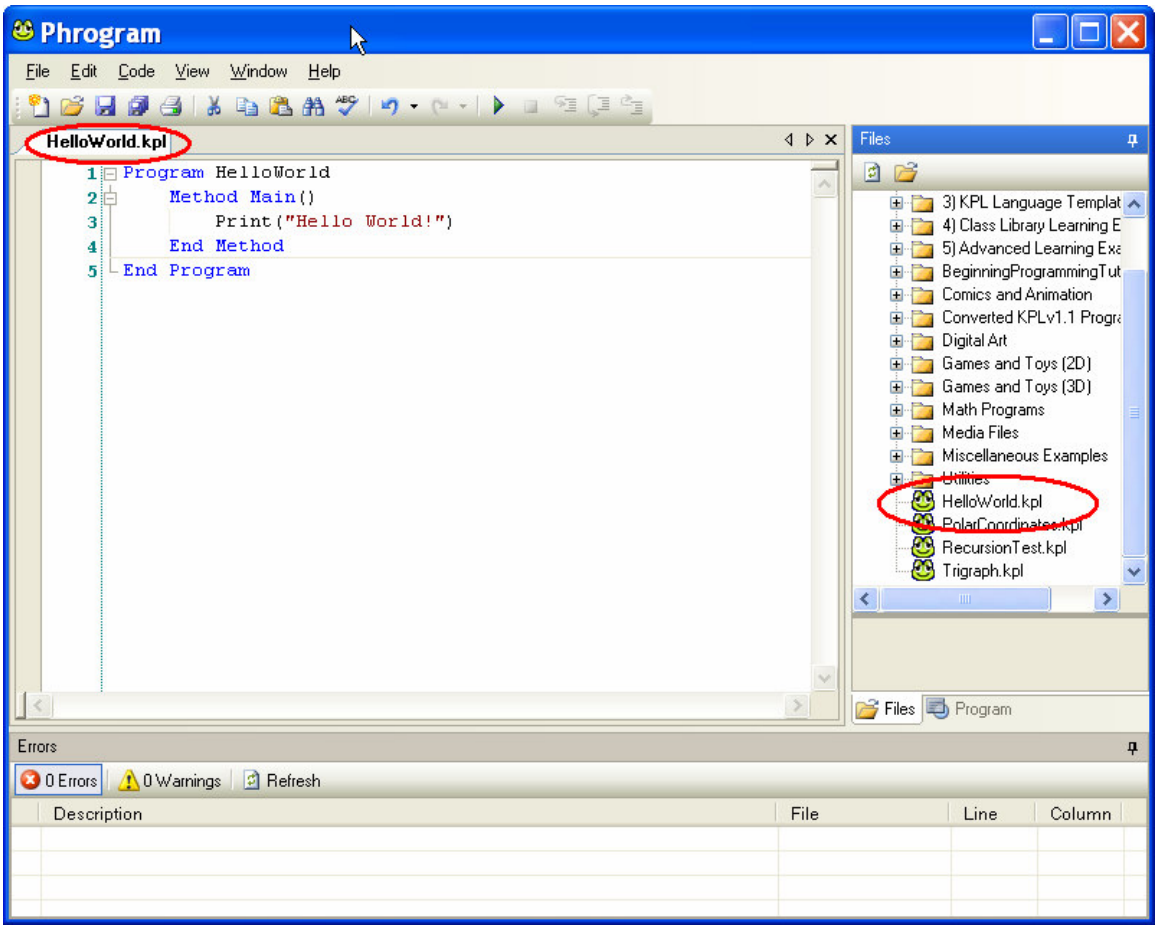
If you encounter an error when working on any of these examples, just carefully compare the code you typed against the code shown here in the tutorial. When you find the difference in your code and change it to match the example, you will be able to run the program.

All computer programmers sometimes have errors in their code, so don't feel bad about it when you do. None of us are perfect! When that happens with your code, just try to stay calm and patient and examine your code carefully – calm and patient and careful is the best way for you to find what's wrong and fix it.

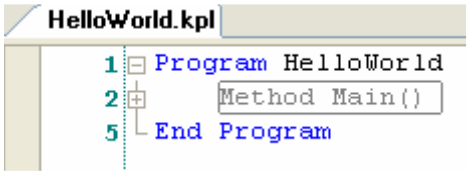
After you have the HelloWorld example working, click the **File** menu, **Save** option, and give your program the file name HelloWorld as shown:



After you save your Phrogram program you will see two places that prove that you did:

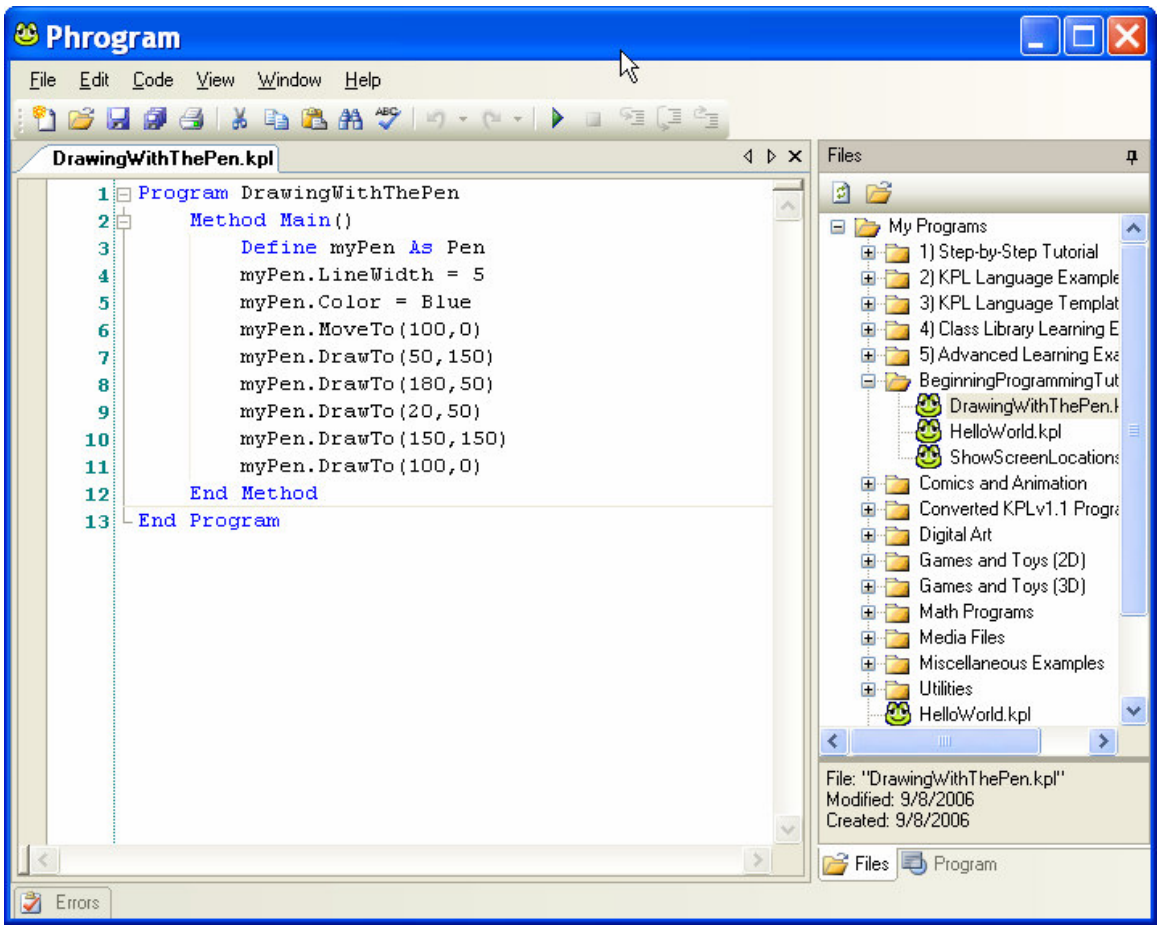


Now that your program is saved to disk, you can re-open it any time by double-clicking on it in the **Files** explorer on the right side of Phrogram. When you re-open your Phrogram program later, you will likely see it appear as shown below – don't panic, all your code is still there!



Click on the little **+** icon shown on line 3, and all the code within Method Main will "expand" so you can see it. Click the **–** icon, and the code will collapse again. This is not a useful feature for such a simple program, but when you get to writing much larger programs, you will see that hiding and showing code this way can be very convenient for you as a programmer.

After saving **HelloWorld**, click the **File** menu, **New Document Window**, and you will see a new **Untitled.kpl** program displayed, like the one you started with. Delete all of the code from it as you did before, and then enter the code for our second example:



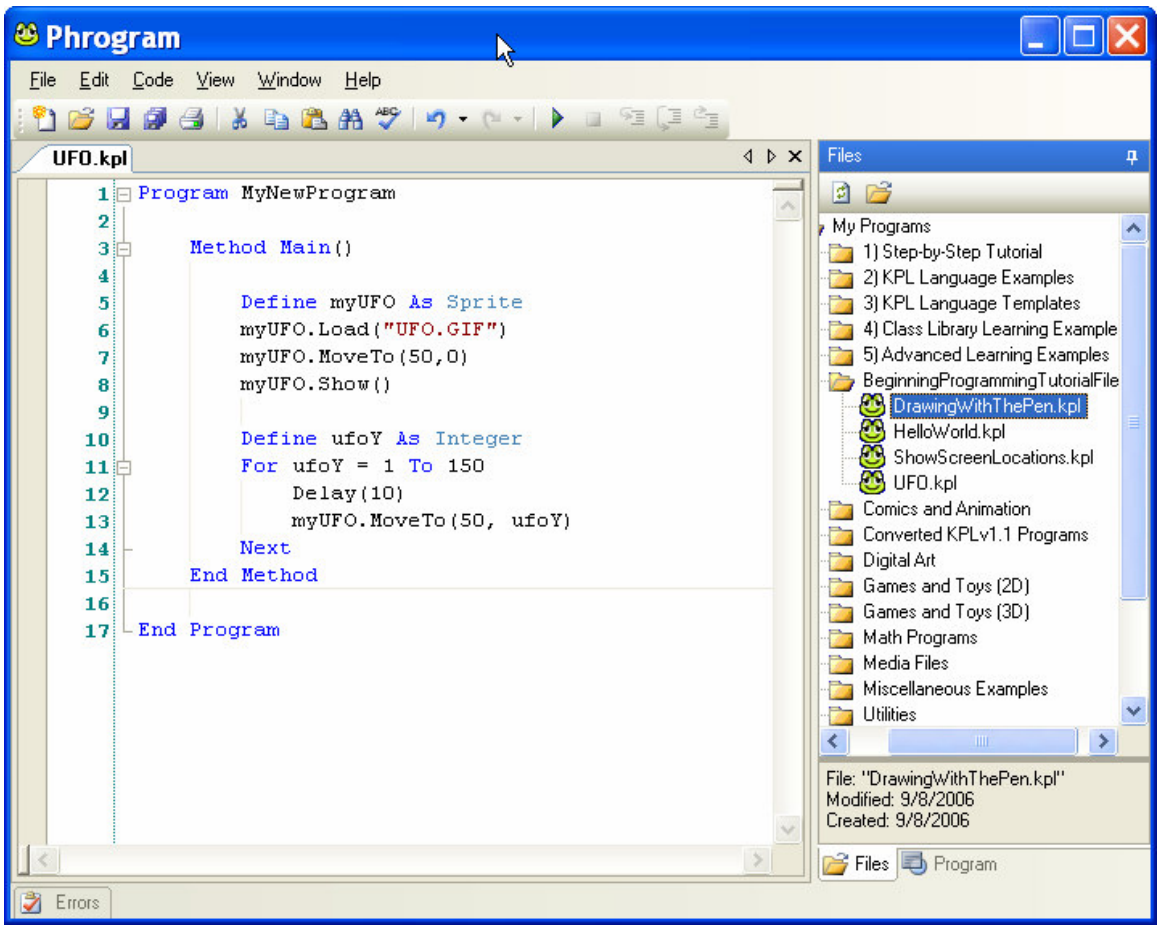
If you enter in all the Phrogram code exactly as shown here and run it, you will see the blue star, and you will be a computer graphics programmer! **Way to go!**

Remember if you encounter an error, just try to stay calm and patient and carefully look for the difference between your program and this example. When you change your code to match this exactly, it will run fine.

Don't forget to save this program after you have it working!

This is also a fun program for you to experiment with. What other colors look good? How does it look if you tell the pen to draw thicker or thinner lines? How would you draw a square or a triangle instead of a star? Here's a very difficult one: how would you make the star larger or smaller?

After saving **DrawingWithThePen**, click the **File** menu, **New Document Window**, and you will see a new **Untitled.kpl** program displayed. Delete all of the code from it as you did before, and then enter the code for our UFO example:



If you enter in all the Phrogram code exactly as shown here and run it, you will see the UFO fly. **Way to go!**

Remember if you encounter an error, just try to stay calm and patient and carefully look for the difference between your program and this example. When you change your code to match this exactly, it will run fine.

Don't forget to save this program after you have it working!

Try these fun exercises if you want some more good programming practice:

- Can you make the UFO move farther?
- How about making it fly from left to right instead of from the top down?
- Can you make the UFO fly down **and** to the right **at the same time**?

Next Steps After This Tutorial

The first thing to do after you get to this point in the tutorial is to look in Phrogram for the **1) Step-by-Step Tutorial** folder. There are 13 Phrogram programs there which you can learn from

next – it's best to study them in order. In fact, some of them will be very similar to programs you have already done in this tutorial!

After you are comfortable with the programs in the **1) Step-by-Step Tutorial**, the **2) Phrogram Language Examples** folder will be a good place to learn more about Phrogram.

And after you are done with that folder, the **4) Class Library Learning Examples** and the **5) Advanced Learning Examples** offer many more.