

Iniziare a programmare con Phrogram

di Jon Schwartz

Documento aggiornato il 5 Settembre 2006
Traduzione di meo aggiornata al 6 Gennaio 2007
Sito web: www.phrogram.com

Iniziare a programmare con Phrogram

Indice

Introduzione: cos'e' la programmazione?.....	3
Perche' imparare a programmare con Phrogram?	3
Come utilizzare questo Tutorial?.....	3
OK, vediamo il primo programma!.....	4
E la grafica?!.....	6
Grafica per i giochi.....	11
Utilizzare variabili e cicli con Phrogram.....	14
Utilizzare Phrogram sul proprio Computer.....	17
I passi successivi dopo il Tutorial.....	24

Introduzione: cos'e' la programmazione?

La programmazione su un computer e' semplicemente: **dare comandi ad un computer**.

I computer sono bravissimi nell'eseguire le istruzioni. Fanno esattamente quello che si dice loro di fare. Ma non hanno alcuna **immaginazione!** Percio', quando si scrivono programmi per dare istruzioni ad un computer, e' necessario specificare in modo molto preciso quello che si vuole far eseguire.

Perche' imparare a programmare con Phrogram?

Differenti linguaggi di programmazione danno al programmatore (che sei tu!) modi diversi di indicare al computer i compiti da svolgere. Phrogram ha diversi importanti vantaggi per chi inizia a programmare:

- Phrogram e' stato progettato per far imparare ai principianti **nel modo piu' semplice possibile**
- Phrogram e' stato progettato per rendere l'apprendimento **il piu' divertente possibile**
- Phrogram, a differenza di altri linguaggi per l'insegnamento, e' stato progettato per essere **il piu' simile possibile** ai linguaggi di programmazione utilizzati attualmente dai programmatori professionisti

Iniziare a programmare con Phrogram e' il modo piu' semplice di imparare la vera programmazione e coloro che vogliono programmare per divertimento scopriranno che Phrogram fornisce tutto cio' che serve, anche piu' di altri linguaggi o ambienti. Ma volendo passare ad altri tipi di linguaggi di programmazione, come gli ambienti di sviluppo "enterprise", si trovera' che Phrogram fornisce un'ottima preparazione per linguaggi come Java, C# o VB. Phrogram e' stato progettato per essere il piu' simile possibile a tali linguaggi e ai loro ambienti di programmazioni. Eccetto, ovviamente, che Phrogram e' molto piu' semplice e molto piu' divertente!

Come utilizzare questo Tutorial?

Per chi e' un vero principiante che non ha mai programmato il modo migliore di utilizzare questo Tutorial e' quello di leggere e studiarlo una sezione alla volta nello stesso ordine, proseguendo solo se si sono ben compresi i concetti presentati in una sezione prima di passare alla successiva. E' probabilmente meglio studiare il Tutorial senza utilizzare un computer prima di aver raggiunto la sezione "**Utilizzare Phrogram sul proprio Computer**". Studiando Phrogram in questo modo si evita di essere distratti da dettagli che vengono appresi in sezioni successive del Tutorial.

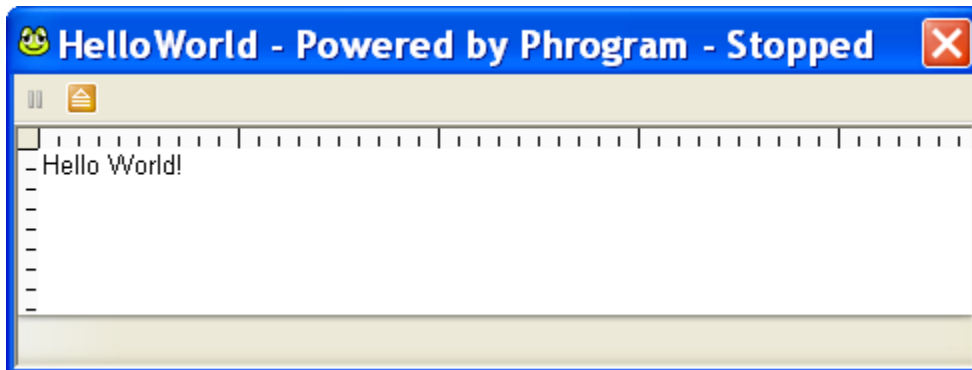
La programmazione richiede di imparare a pensare in un modo diverso da quello normalmente utilizzato dalla gente. I Computer costringono ad essere molto piu' logici, ordinati e precisi di quanto sia altrimenti necessario! Imparare puo' sembrare difficile all'inizio, ma tutti possono farcela! Ed una volta capito il meccanismo diventa tutto molto piu' facile.

Uno dei modi migliori per imparare questo nuovo modo di pensare e' fare domande ed avere spiegazioni da chi gia' conosce la programmazione. Riuscite a pensare a qualcuno che possa rispondere alle vostre domande mentre studiate questo Tutorial su Phrogram?

OK, vediamo il primo programma!

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

Eccolo! Quando viene eseguito questo programma Phrogram, si vedra':



Le 5 linee di codice Phrogram sopra sono l'immagine di come appare il codice in Phrogram, così si può leggere esattamente come appare il programma sul computer.

La prima cosa da dire è che i numeri di linea che si vedono sulla sinistra di ogni linea di codice non sono utilizzati da Phrogram ma servono solo come informazione. Alcuni vecchi linguaggi, come il GWBASIC, utilizzavano effettivamente i numeri di linea per eseguire il codice. Phrogram non lavora in questo modo. I programmi Phrogram vengono normalmente eseguiti un'istruzione alla volta partendo dall'inizio e proseguendo per tutto il programma. Ci sono alcune eccezioni ma non sono presentate in questo Tutorial per principianti.

Una importante regola di programmazione in Phrogram è che ogni istruzione deve essere posta su una linea separata. Ad esempio questo programma Phrogram, che contiene le stesse istruzioni ma sulla stessa riga, non funziona affatto:

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

Tutti i linguaggi di programmazione hanno delle regole che il programmatore deve seguire in modo che le istruzioni siano comprensibili al computer. Anche la comunicazione tra persone ha molte regole che servono allo stesso scopo, non ce ne accorgiamo solo perché siamo così abituati che le applichiamo senza pensare. Ad esempio non diciamo "A presto!" quando solleviamo la cornetta e non diciamo "Pronto?" quando mettiamo giù il telefono! È un esempio semplice ma appropriato perché anche Phrogram richiede un modo preciso per iniziare e terminare un programma. Tutti i programmi Phrogram devono iniziare con una linea come **Program HelloWorld** nella linea 1. E tutti i programmi Phrogram debbono finire con **End Program** come nella linea 5:

```

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

```

Si puo' chiamare il programma come si vuole, ma e' meglio se il nome descrive quello che fa effettivamente il programma. In questo caso **HelloWorld** e' il nome scelto. Si puo' scegliere un nome differente come **Program MioPrimoProgramma**.

Method Main() e' la seconda cosa importante da sapere sulla programmazione con Phrogram. Tutti i programmi Phrogram iniziano ad eseguire la prima istruzione contenuta in **Method Main()**. Proseguendo a leggere si passa all'istruzione **Print("Hello World!")**. Iniziare con il **Method Main()** sembra un modo arbitrario di definire dove inizia l'esecuzione di un programma - ma e', in effetti, il modo utilizzato da tutti i piu' moderni linguaggi di programmazione. Come ci si puo' logicamente aspettare **End Method** corrisponde e dichiara la fine di **Method Main()**.

Non si vedranno altri dettagli sui metodi in questo Tutorial quindi, per adesso, sara' sufficiente vedere il codice che segue contenuto "dentro" il **Method Main()**. E' una sola istruzione: **Print("Hello World!")**.

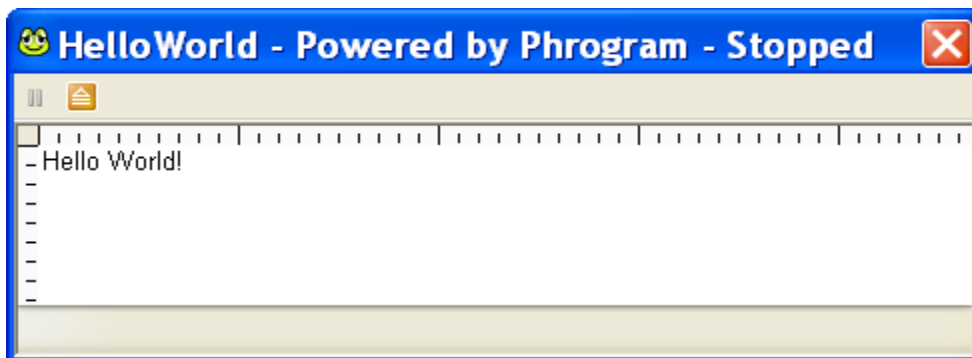
Tutto quello visto fino ad ora e' una spiegazione dettagliata di un programma che contiene una sola istruzione **Print("Hello World!")** ovvero: Computer scrivi **"Hello World!"**
 Riassumendo: le linee 1 e 5 dicono al computer dove sono l'inizio e la fine del programma Phrogram. E le linee 2 e 4 dicono al computer dov'e' l'inizio e la fine di **Method Main()**.


```

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

```

Ora controllando la finestra che appare eseguendo il programma si nota che "dentro" la finestra il computer ha fatto una sola cosa, quella che e' stata richiesta con il programma Phrogram. Il computer ha scritto le parole **Hello World!**



Phrogram aggiunge la parola "Stopped" al titolo della finestra quando il programma ha terminato l'esecuzione. Per chiudere la finestra basta premere il bottone arancione sulla barra degli strumenti che contiene l'icona: 

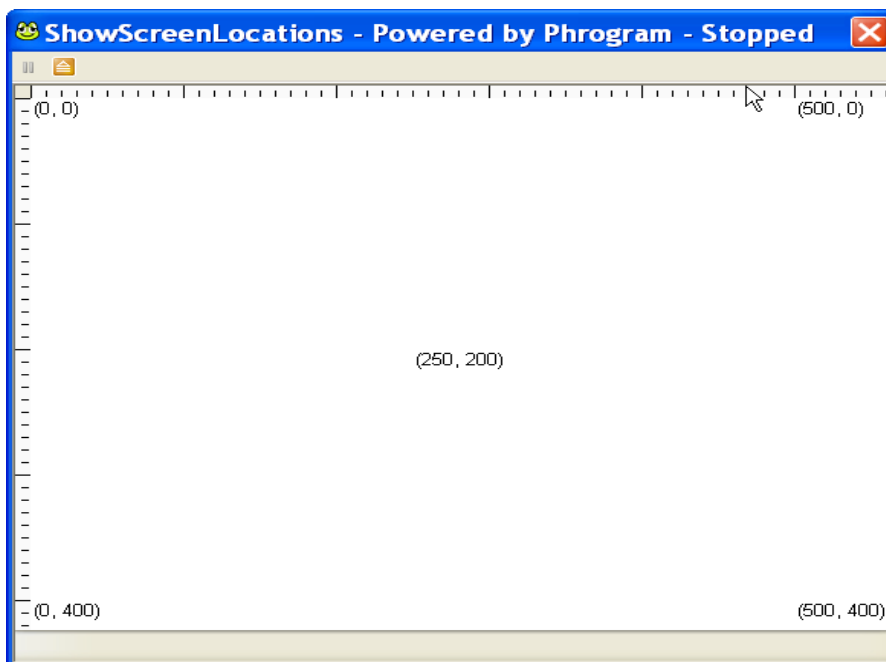
E la grafica?!

Hello World! e' il primo classico programma, ma non e' molto divertente! Quindi vediamo ora qualcosa di piu' simpatico con la grafica.

Una spiegazione della grafica su Computer deve iniziare con la descrizione di come vengono localizzate le immagini sul video. I computer utilizzano un **sistema di coordinate** che e' differente dal sistema di **coordinate cartesiane** che si impara a scuola. Ma il sistema di coordinate del computer e' piu' semplice da utilizzare per disporre gli oggetti sullo schermo.

I computer utilizzano un sistema di coordinate (**X, Y**) per individuare le posizioni sullo schermo per cui **il lato sinistro dello schermo ha coordinate X=0** e **la parte alta dello schermo ha coordinate Y=0**. Questo significa che l'**origine**, dove **X=0** e **Y=0**, e' l'**angolo in alto a sinistra** dello schermo. Spostandosi a destra si incrementa il valore di X e spostandosi verso il basso si incrementa il valore di Y.

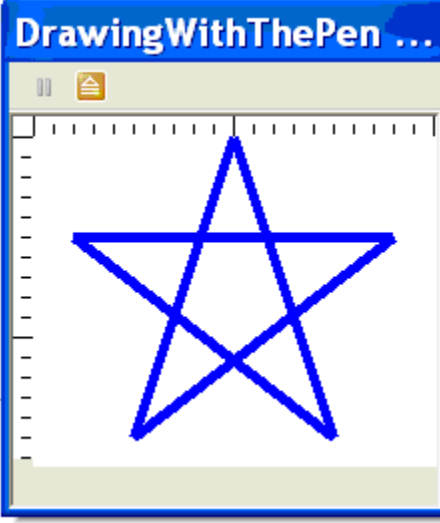
Per chi non e' abituato ad utilizzare sistemi di coordinate puo' sembrare una spiegazione complicata, ecco quindi un esempio che mostra come sono visualizzate le locazioni (**X, Y**) da un programma Phrogram:



Il **primo valore** di ogni coppia di coordinate e' il **valore di X** - come si puo' vedere il valore di X cresce spostandosi da sinistra a destra. Il **secondo valore** e' il **valore di Y** - come si puo' vedere il valore di Y cresce spostandosi verso il basso dello schermo.

Scriviamo il nostro primo programma Phrogram grafico e vediamo come si utilizzano il sistema di coordinate per realizzare alcune immagini con Phrogram. Vediamo innanzi tutto l'intero programma e la grafica prodotta eseguendolo. Quindi vedremo ogni singolo passo in dettaglio per capire come funziona ogni istruzione:

```
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
```




Ora ci sono 9 istruzioni Phrogram nel **Method Main()**, dalla linea 3 alla linea 11. Molto di piu' che nel primo esempio, ma si sara' d'accordo che e' simpatico che Phrogram possa disegnare una bella stella blu sul video con sole 9 istruzioni.

Deve essere notato che il programma inizia e termina nello stesso modo del primo programma, tranne il fatto che il programma si chiama **DrawingWithThePen** (NdT ScriviConLaPenna):

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

Ora cominciamo ad aggiungere istruzioni al **Method Main()**, per capire cosa fa ciascuna:

```
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
```



Sono state aggiunte le prime tre istruzioni al programma ma, come si vede, Phrogram non visualizza ancora nulla. La prima istruzione inserita e' **Define myPen As Pen**. Si vedra' piu' avanti una spiegazione piu' completa, ma per il momento si puo' tradurre letteralmente con "Sto definendo myPen come una Penna". Analizzando in modo logico la frase: se dico a Phrogram che myPen e' Pen, ovvero un oggetto Penna, allora Phrogram capira' l'istruzione e permettera' a myPen di fare tutte le cose che fanno le Penne. Cosa fanno le penne? Beh, le penne possono disegnare, possono essere di colori differenti, alcune penne possono tracciare linee fini altre piu' larghe, ... Tutto cio' vale con le penne di Phrogram esattamente come con le penne che si usano con la carta.

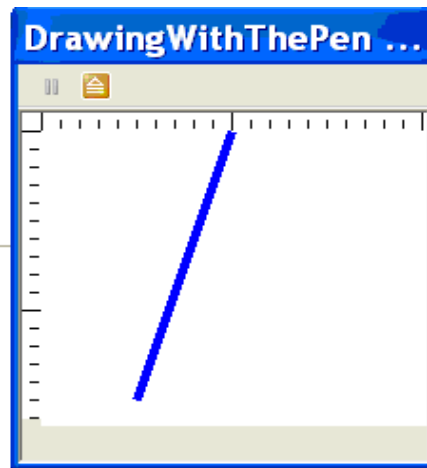
Vedendo le due istruzioni successive a **Define myPen As Pen**:

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

Cosa fara' **myPen.Color = Blue**? Elementare. Dice a Phrogram che si vuole che la penna scriva con il colore Blue. **myPen.LineWidth = 5** dice a Phrogram che quando verra' disegnata una linea con myPenna, si vuole che sia larga 5 pixel. Un pixel e' un piccolo puntino sullo schermo, cosi' una linea di 5 pixel apparira' leggermente piu' ampia come se fosse stata tracciata con un evidenziatore. Naturalmente si puo' utilizzare un valore differente come larghezza di linea Con **myPen.LineWidth = 2** si avra' una linea piu' fine, come con una penna a sfera. E con **myPen.LineWidth = 10** si traccerà una linea piu' spessa, come quella di un gesso.

Per ora e' stato detto a Phrogram **come** dovra' disegnare, ma non gli e' ancora stato ordinato di disegnare nulla. Iniziamo ora a tracciare la prima linea della stella:

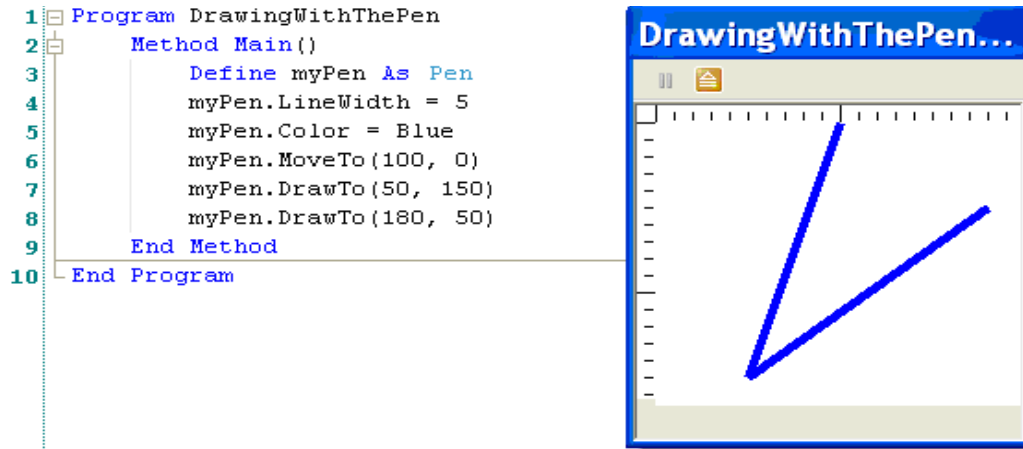
```
1 Program DrawingWithThePen  
2 Method Main()  
3   Define myPen As Pen  
4   myPen.LineWidth = 5  
5   myPen.Color = Blue  
6   myPen.MoveTo(100, 0)  
7   myPen.DrawTo(50, 150)  
8 End Method  
9 End Program
```



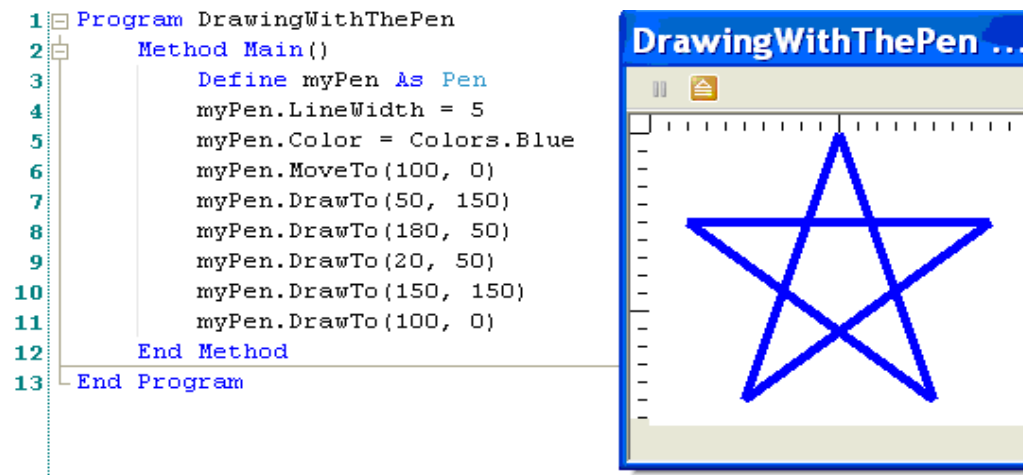
La prima istruzione aggiunta e' **myPen.MoveTo(100, 0)**. Questo dice a Phrogram di muovere la penna nella posizione (100, 0) dello schermo, che e' la punta della stella. Quando si definisce una nuova penna in Phrogram, questa parte sempre dalla locazione (0, 0) – l'angolo in alto a sinistra. Utilizzando il comando MoveTo si dice a Phrogram di muovere la penna ma di NON tracciare una linea nello spostamento.

La seconda istruzione aggiunta e' **myPen.DrawTo(50, 150)**. Questa istruzione fa muovere la penna mentre traccia la prima linea della stella. Considerando i valori (X, Y) per questi due punti si nota che questa istruzione chiede a Phrogram di tracciare di disegnare da X = 100 a X = 50, quindi la linea e' tracciata verso sinistra. E poiche' la richiesta e' da Y = 0 a Y = 150, la linea e' rivolta verso il basso.

Aggiungendo una nuova linea a Phrogram, si aggiunge una linea alla stella:



L'istruzione aggiunta e' **myPen.DrawTo(180, 50)**. Phrogram continua a spostare la penna dal punto precedente, che era (50, 150). Ricordate la "lavagna magica" che permette ai bimbi di disegnare agendo su due manopole? La penna di Phrogram e' una specie di lavagna magica computerizzata! Aggiungendo le ultime righe di codice si completa la stella:



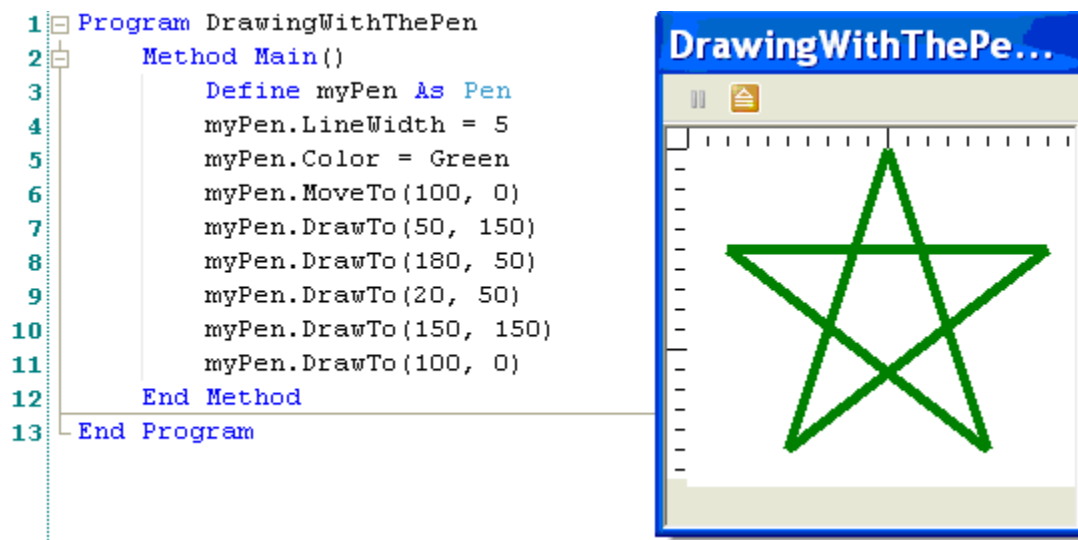
Come si puo' vedere le nuove tre linee di codice tracciano tre nuove linee sul video. In tutto le istruzioni impartite hanno fatto tracciare cinque linee con **myPen**, ed il risultato e' la stella blu.

Il programma Phrogram necessario per fare questo e' molto piccolo: solo 9 istruzioni. La cosa piu' difficile e' abituarsi al modo di operare del sistema di coordinate (X, Y) di Phrogram. A questo punto se i valori (X, Y) non sono abbastanza chiari e' meglio ritornare all'inizio della sezione e leggerla nuovamente. Se anche a questo punto le coordinate (X, Y) non sono chiare si puo'

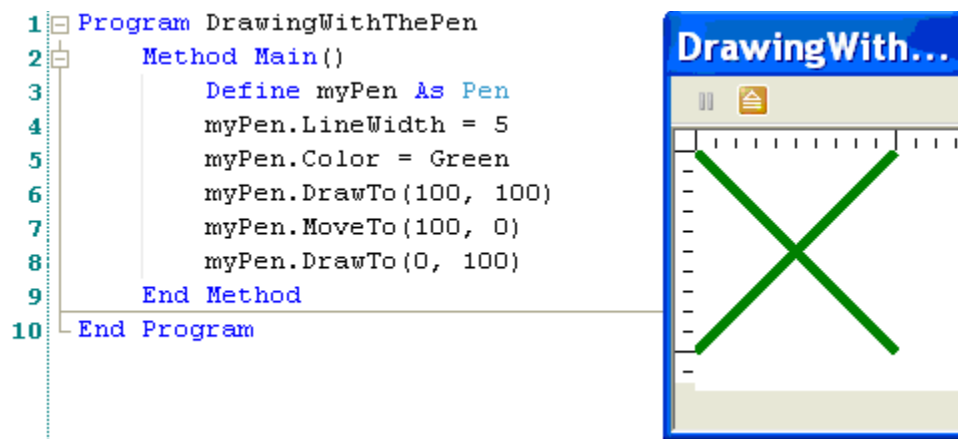
provare a scrivere una propria versione del programma e provare a tracciare linee tra posizioni differenti dello schermo. Per esempio disegnando un quadrato o un triangolo. Per imparare a farlo sul proprio computer si puo' passare alla sezione **Utilizzare Phrogram sul proprio Computer**. Una volta aquisita familiarita' con le coordinate (X, Y) si puo' riprendere da questo punto del Tutorial.

E' importante dedicare un tempo sufficiente ad essere ben confidenti sul modo di operare di Phrogram con le coordinate (X, Y) perche' sono la base di tutta la programmazione grafica. Altri linguaggi trattano la grafica nello stesso modo, percio' cosi' si imparano i concetti fondamentali della computer graphics validi programmando con ogni linguaggio.

Aggiungiamo un dettaglio finale all'esempio, che presenta un altro elemento di controllo che si ha utilizzando la Pen di Phrogram. Con una lavagna magica non si puo' controllare il colore, ma con Phrogram e' molto facile:



Come si vede basta cambiare `myPen.Color = Blue` in `myPen.Color = Green`. Facile! Utilizzando `MoveTo` and `DrawTo` si puo' creare qualsiasi tipo di oggetto o piu' oggetti. Ecco un altro semplice esempio:

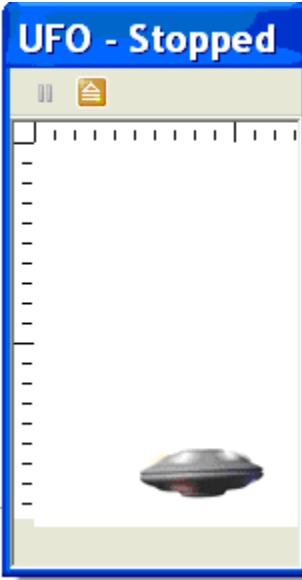


Grafica per i giochi

L'argomento principale dell'esempio precedente era la spiegazione di come funzionano le coordinate (X, Y) per disegnare la grafica. Ma le immagini nei giochi normalmente non vengono disegnate ogni volta. Vengono raccolte da un file che contiene l'immagine di un UFO, di un asteroide oppure di un Elfo con una spada. Phrogram utilizza gli Spite (immagini bidimensionali) per rendere tutto questo **molto** facile!

Come e' stato fatto in precedenza si partira' dall'esempio completo e quindi si vedra' in dettaglio come funziona il programma in modo da insegnare a programmare da soli. Ecco il programma intero - all'interno di **Method Main()** ci sono esattamente 9 istruzioni come nell'esempio precedente. Queste 9 istruzioni sono sufficienti per visualizzare un UFO e farlo scendere lentamente lungo lo schermo:

```
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
```



Questo programma inizia e finisce nello stesso modo dell'esempio precedente - tutti i programmi Phrogram sono cosi' - eccetto che il programma si chiama **UFO**:

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

Vediamo ora alcuni esempi di immagini che sono inclusi con Phrogram. Si puo' utilizzare **QUALSIASI** immagine con Phrogram, comprese nuove immagini create apposta o copiate - quelle che seguono sono alcune delle molte immagini fornite con Phrogram:



UFO.GIF



QUAD.GIF



PLANE.PNG



SPIDER.PNG



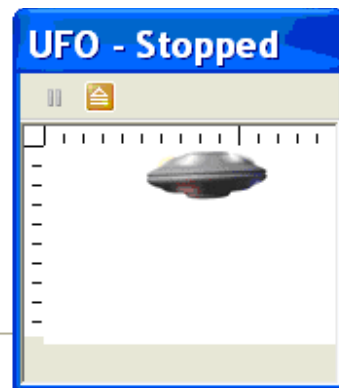
BLUEBALL.PNG

Visti alcuni esempi di immagini che si possono utilizzare con Phrogram, incominciamo ad inserire le istruzioni che visualizzano l'UFO in alto nello schermo:

```

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

```



La prima istruzione inserita e' **Define myUFO As Sprite**. Nella programmazione uno **Sprite** e' un oggetto grafico che puo' essere visualizzato e manipolato sul video -come il nostro UFO! Con questa prima istruzione si sta dicendo a Phrogram che **myUFO** e' uno di questi oggetti.

La seconda istruzione e' **myUFO.Load("UFO.GIF")**. Come gia' detto Phrogram ha delle regole su come debbono essere scritte le istruzioni. Le regole per utilizzare **Load** non sono difficili, ma sono precise! **Load** richiede un solo "valore" per funzionare: richiede il nome del file che contiene l'immagine da visualizzare nello sprite - in questo caso **"UFO.GIF"**. Il valore deve essere messo tra apici doppi.

Ecco due esempi **sbagliati** per caricare le immagini in Phrogram. Attenzione: **le due istruzioni seguenti non funzionano** perche' non seguono le regole di Phrogram. Riuscite a capire perche'?

```

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

```

Sommario: **myUFO.Load("UFO.GIF")** dice a Phrogram di utilizzare l'immagine contenuta nel file **"UFO.GIF"** quando visualizza **myUFO**

Phrogram deve sapere dove mettere l'immagine sullo schermo, perciò viene data l'istruzione **myUFO.MoveTo(50, 0)**. **MoveTo** richiede due valori. Il primo e' la posizione sull'asse **X** per lo Sprite, che si vuole sia 50. La seconda e' la posizione sull'asse **Y**, che si vuole sia 0.

Un dettaglio importante da notare e' che **non si usano gli apici per i valori numerici**. In generale Phrogram **richiede** gli apici doppi intorno a valori che sono "parole" e Phrogram **non vuole** apici

per i valori che sono **numeri**. Questo puo' non essere ovvio o intuitivo per chi inizia a programmare, ma e' cosi' nella maggioranza dei linguaggi di programmazione. Si veda la sezione sui tipi di dati nella **Phrogram v 2 User Guide** (NdT la guida utente non e' ancora stata tradotta al momento) per maggiori informazioni.

Sommario: `myUFO.MoveTo (50, 0)` dice a Phrogram di muovere lo sprite `myUFO` alla posizione **(50, 0)**.

Tutto quello che resta da fare e' visualizzare lo sprite. Questa e' l'istruzione piu' semplice: `myUFO.Show ()`. Tutto qui! Phrogram aspetta a visualizzare l'immagine dell'UFO fino a quando non viene data l'istruzione **Show** – questo consente di definire le caratteristiche dello sprite in modo completo prima di visualizzarlo all'utente.

E' stata una spiegazione molto dettagliata, ma in realta' e' semplice con Phrogram. Ecco il programma completo e cosa si vede quando e' eseguito. Le istruzioni dicono a Phrogram **come** visualizzare lo sprite, **dove** visualizzarlo e **quando** visualizzarlo. Vedete? Sapete riconoscerle ora?

```
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
```



Utilizzare variabili e cicli con Phrogram

Si puo' far "atterrare" l'UFO facendolo scendere lungo lo schermo fino al fondo della finestra? Certo! Per farlo si definira' ed utilizzerà una "variabile" in modo da spiegare cosa siano e come si utilizzano le variabili in un programma Phrogram.

Il termine variabile e' molto descrittivo, significa che il valore puo' cambiare nel tempo. Il valore che cambia e' la cosa fondamentale nell'utilizzo di una variabile, come si vedra' nell'esempio. Le variabili debbono essere definite con un nome che abbia un significato con l'uso che se ne fa. In questo caso la variabile e' utilizzata per cambiare la posizione dell'ufo sull'asse Y per farlo scendere ed "atterrare", quindi logicamente la variabile sara' chiamata **ufoY**.

Ecco la linea di codice che definisce la variabile:

```
Define ufoY As Integer
```

Define e' la parola chiave che fa sapere a Phrogram che si sta definendo una nuova variabile. **ufoY** e' il nome dato per l'utilizzo della variabile. **As Integer** dice a Phrogram che **ufoY** e' definita come una variabile Integer. Le variabili Integer possono avere valori numerici interi come **-1** o **0** o **43**. Quindi semplicemente tutti i valori interi.

La posizione iniziale dell'UFO e' **(50, 0)**, perche' questo e' il punto in cui si e' detto a Phrogram di spostarlo. Come puo' muoversi sullo schermo? Lo fara' aumentando la sua **posizione sull'asse Y** - da **(50, 1)** a **(50, 2)** quindi a **(50, 3)** poi a **(50, 4)**, ecc... Dipende dal programmatore decidere dove arrivera', ad esempio si puo' decidere di farlo arrivare al punto **(50, 150)**.

Capita la sequenza? Il valore della **X** della posizione dello spite e' sempre **50**. Il valore della **Y** aumenta di un **1** pixel alla volta, da **0** fino a **150**. E' importante capire il concetto prima di vedere come si realizza nel codice Phrogram - se non e' chiaro... meglio rileggere il paragrafo precedente.

Ecco il codice Phrogram che ordina di incrementare il valore della variabile **ufoY** da 1 a 150, un passo alla volta:

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

Questo e' il primo "ciclo". I cicli e' uno dei nuovi concetti che debbono essere appresi se non si ha mai programmato prima ma, una volta capito il concetto, i cicli sono semplici.

Questo ciclo parte con un valore di **ufoY = 1**. Poiche' e' stato scritto **To 150**, il ciclo si interrompera' dopo aver raggiunto il valore di **ufoY = 150**. E' importante notare la parola chiave **Next**, che definisce la fine delle istruzioni eseguite nel ciclo. Quando Phrogram raggiunge il **Next** sa che deve **incrementare il valore di ufoY** al valore successivo - da 1 a 2, da 2 a 3, ... fino ad arrivare da 149 a 150.

Fondamentalmente si dice a Phrogram di contare da 1 a 150 e si vuole che Phrogram usi la variabile ufoY per mantenere il numero che e' stato contato.

E cosa si vuole che avvenga mentre Phrogram sta contando? Si vuole che l'UFO scenda lungo lo schermo... Ecco quello che `myUFO.MoveTo (50, ufoY)` dice a Phrogram di fare.

Poiche' questa istruzione e' "dentro" al ciclo `For` Phrogram eseguirà tale istruzione ogni **volta contando** da 1 a 150. Questa e' la vera importanza di un ciclo! Contare semplicemente da 1 a 160 non e' così interessante, ma se si **fa qualcosa di utile ad ogni passo** ecco che **questo** permette di fare moltissime cose simpatiche, come muovere l'UFO giu' lungo lo schermo. Vediamo come succede:

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

Come funziona `MoveTo ()` e' già stato visto: Phrogram muove `myUFO` alla posizione `(X, Y)` indicata. Cosa c'è di differente ora? Prima l'UFO e' stato spostato alla posizione `(50, 0)`. Cosa succede quando questo viene effettuato in un ciclo ed utilizziamo la variabile `ufoY`? La prima volta nel ciclo il valore di `ufoY = 1`, la seconda `ufoY = 2`, poi `ufoY = 3`, quindi `ufoY = 4`, ecc... fino a `ufoY = 150`. La prima volta che Phrogram esegue il ciclo utilizza il valore `ufoY=1` per muovere lo sprite, quindi l'istruzione effettivamente eseguita e':

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

Ed al passaggio successivo il valore e' `ufoY = 2`, quindi Phrogram esegue:

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

E così via fino a che Phrogram ha effettuato il ciclo fino ad arrivare a 150:

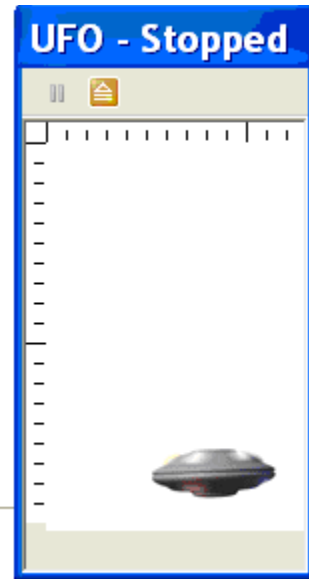
```
myUFO.MoveTo ( 50, 3 )
myUFO.MoveTo ( 50, 4 )
myUFO.MoveTo ( 50, 5 )
...
myUFO.MoveTo ( 50, 150 )
```

Ogni volta che si esegue il ciclo `For MoveTo` muove `myUFO` di 1 pixel lungo lo schermo - esattamente quello che si voleva!

In questo esempio si sono viste le variabili ed i cicli per la prima volta, e come sia utile eseguire istruzioni in un ciclo - si tratta di concetti di programmazione **molto importanti!** Se non e' ancora tutto chiaro e' meglio ritornare all'inizio della sezione e leggerla un'altra volta.

Aggiungiamo il dettaglio finale per concludere il programma:

```
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
```



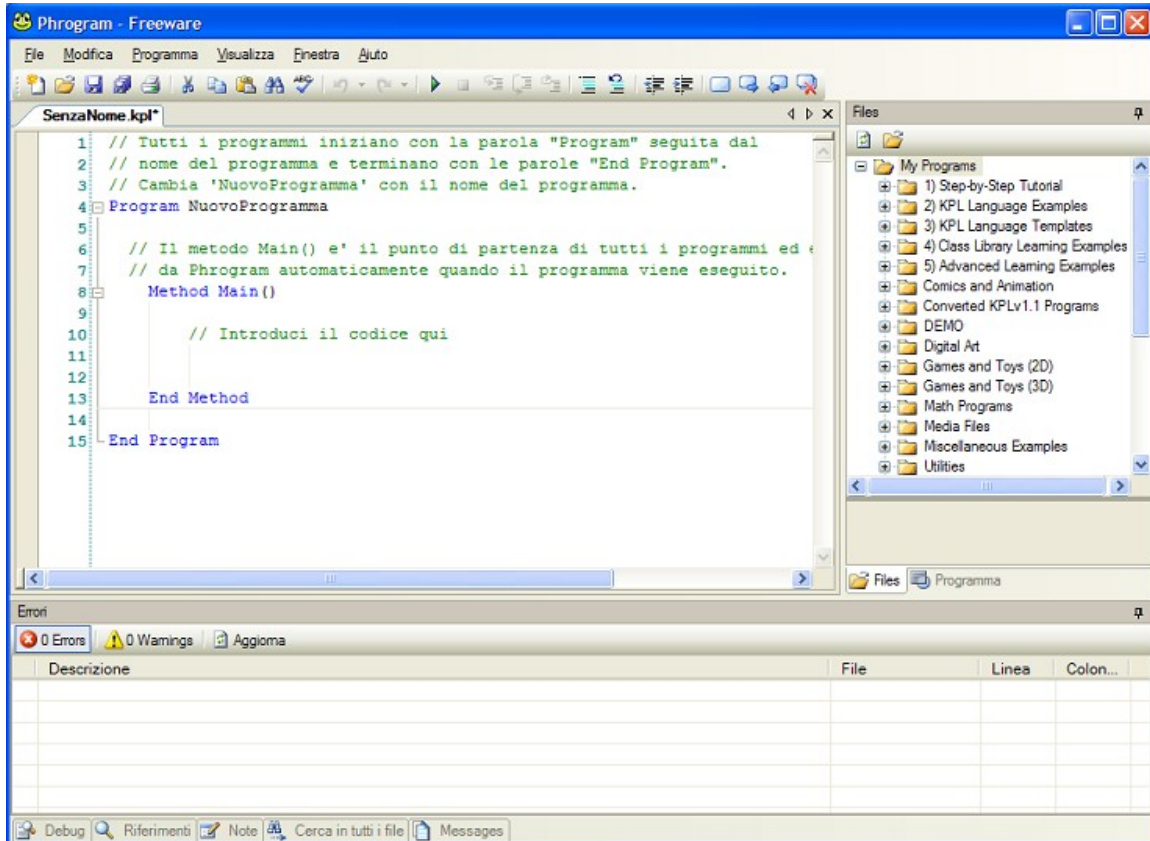
L'unica nuova istruzione che e' stata aggiunta e' **Delay (10)**, all'interno del ciclo **For**. Perche'? **Perche' i computer fanno i conti molto molto molto in fretta!** Per un computer contare da 1 a 150 richiede meno tempo di un battito di ciglia per un umano. Veramente! I computers sono **cosi'** veloci! Se si vuole vedere davvero l'UFO scendere nello schermo e' necessario rallentare il computer mentre esegue il ciclo. **Delay (10)** dice semplicemente a Phrogram di fermarsi un poco prima di muovere l'UFO ogni volta.

Per contare i secondi come un orologio si dice "MilleUNO", "MilleDUE", "MilleTRE" in modo da rallentare un poco? E' la stessa cosa. Con **Delay (10)** si rallenta il conteggio di Phrogram. Scrivendo questo programma sul computer si puo' provare a cambiare il **10** con altri valori e quindi eseguire il programma. Si puo' provare con **Delay (2)** e quindi con **(50)**. Fara' davvero una grande differenza nella velocita'!

Sono state usate molte pagine per spiegare il programma in dettaglio che ora sembra piu' complesso di quanto non sia veramente. Una volta fatta un po' di pratica, inserire queste 9 istruzioni in Phrogram non richiede molto tempo - basta molto poco per realizzare divertenti programmi grafici con Phrogram!

Utilizzare Phrogram sul proprio Computer

Questa sezione prevede che Phrogram sia già stato installato e si concentra su come iniziare ad utilizzarlo. Basta cercare e lanciare Phrogram dal menu di Start di Windows. Lanciato Phrogram apparirà qualcosa come:



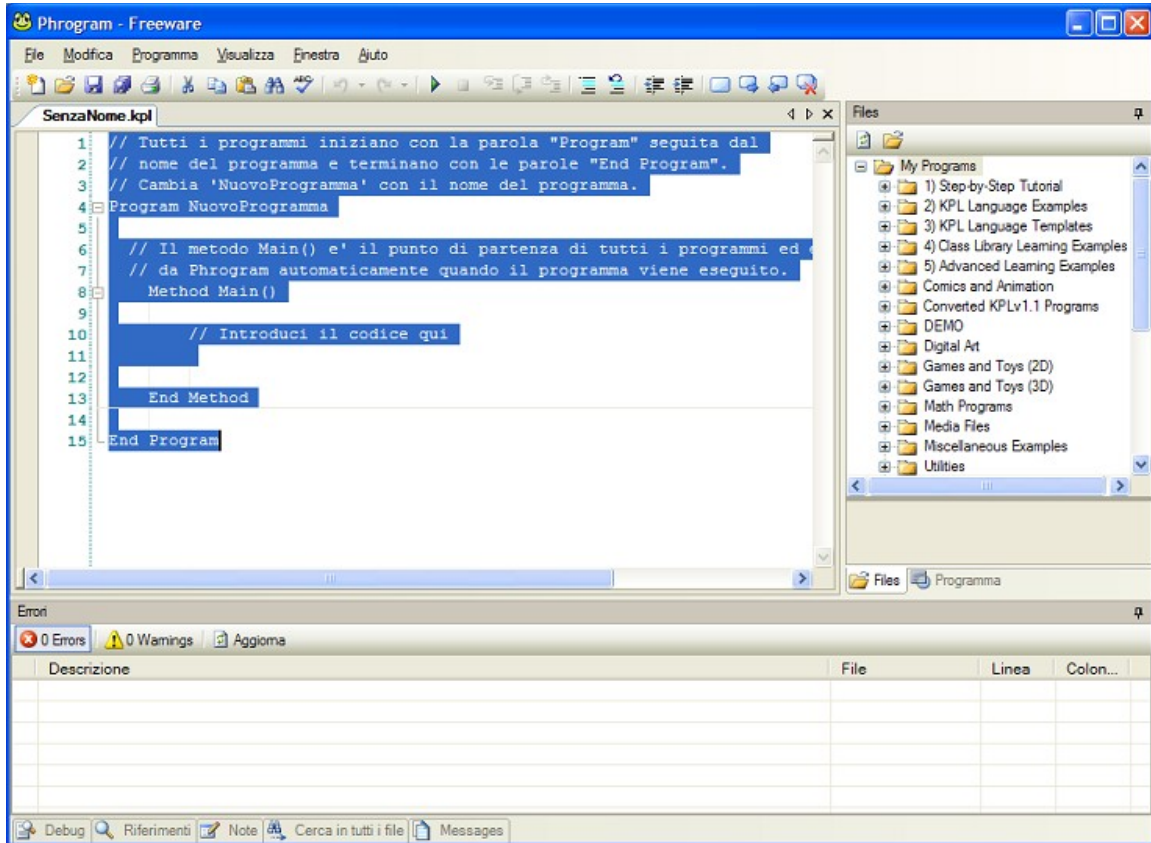
Phrogram sta mostrando un nuovo programma Phrogram chiamato SenzaNome.kpl nel pannello di modifica del codice. Le righe in verde sono “commenti” che riportano informazioni a chi legge il programma, ma che non contengono alcuna istruzione a Phrogram. I commenti sono indicati con due barre iniziali: //

Ci sono molti commenti nei programmi di esempio di Phrogram, ed è importante imparare ad usarli spesso quando si scrivono i propri programmi. I commenti aiutano gli altri a capire cosa fa il programma - ma servono anche a chi ha scritto i programmi a ricordare quando si rileggono dopo un po' di tempo!

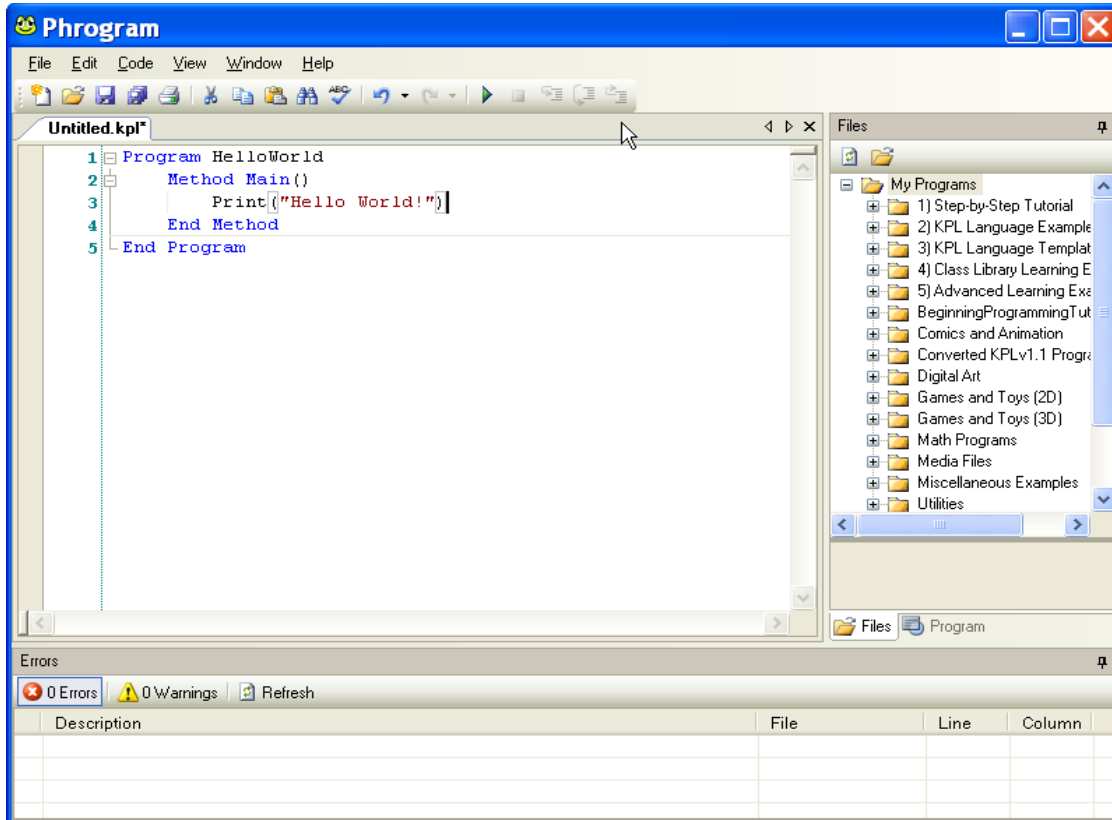
L'editor di Phrogram ha molte caratteristiche simili agli editor dei word processor o dei programmi di email. Analizziamo le voci del menu items e della toolbar. Spostando il cursore sulle icone della toolbar vengono visualizzati dei suggerimenti che li identificano.

Come esempio si introdurranno gli stessi tre semplici programmi visti all'inizio del Tutorial, quindi bisogna iniziare cancellando tutto il codice che è visualizzato nel file SenzaNome.kpl. Per


farlo clicca sul menu **Modifica** e quindi sulla voce **Seleziona tutto**. Si vedrà che tutto il codice è evidenziato come mostrato nella figura sotto. Con tutto il codice evidenziato si deve premere sul tasto "Canc", oppure selezionare il menu **Modifica** e quindi la voce **Taglia**, tutto il codice sarà cancellato dall'editor di programma di Phrogram.

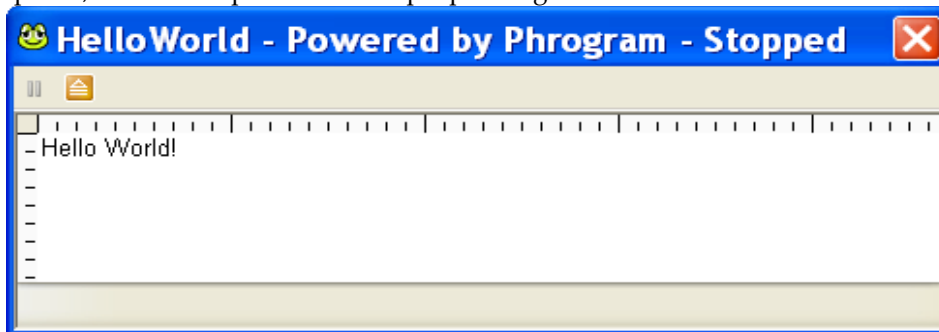


Una volta cancellato si deve digitare il primo programma di esempio in Phrogram, come si vede nella figura seguente:



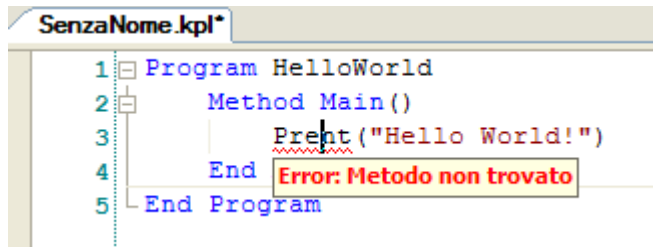
Bisogna ricordarsi che i computer sono molto precisi - quindi bisogna scrivere il programma Phrogram esattamente come visualizzato sopra. Se si scrive in modo differente potrebbe non funzionare. Si puo' utilizzare il tasto **TAB** per indentare il codice. E si deve utilizzare il tasto **INVIO** per lasciare andare a capo o lasciare una linea bianca. Indentare il codice e lasciare linee bianche non e' necessario, ma rende il programma Phrogram piu' semplice da leggere.

Una volta scritto il programma cliccando sull'icona  o premendo il tasto **F5** il programma viene eseguito. Se il codice e' stato inserito correttamente si vedra' una finestra pop up come questa, eccetto che puo' essere un po' piu' larga:



Fatto! Ora sei un programmatore! Così' si fa! Woohoo! Ok, ok, e' un programma veramente piccolo e semplice. Ma l'hai scritto tutto tu e l'hai fatto girare!

Se Phrogram non capisce il codice che e' stato inserito, mostra cosa non ha capito. Phrogram visualizzera' una linea rossa ondulata sotto il codice che non capisce. Mettendo il cursore sulla linea viene visualizzato un popup come quello visualizzato sotto che dice **Error: Metodo non trovato**. In questo esempio si voleva scrivere **Print** ma e' stato scritto **Prent**. I computer sono molto precisi e richiedono istruzioni precise. I computer sa cosa fare se si usa il metodo **Print** – ma non sa cosa fare se si scrive **Prent**!

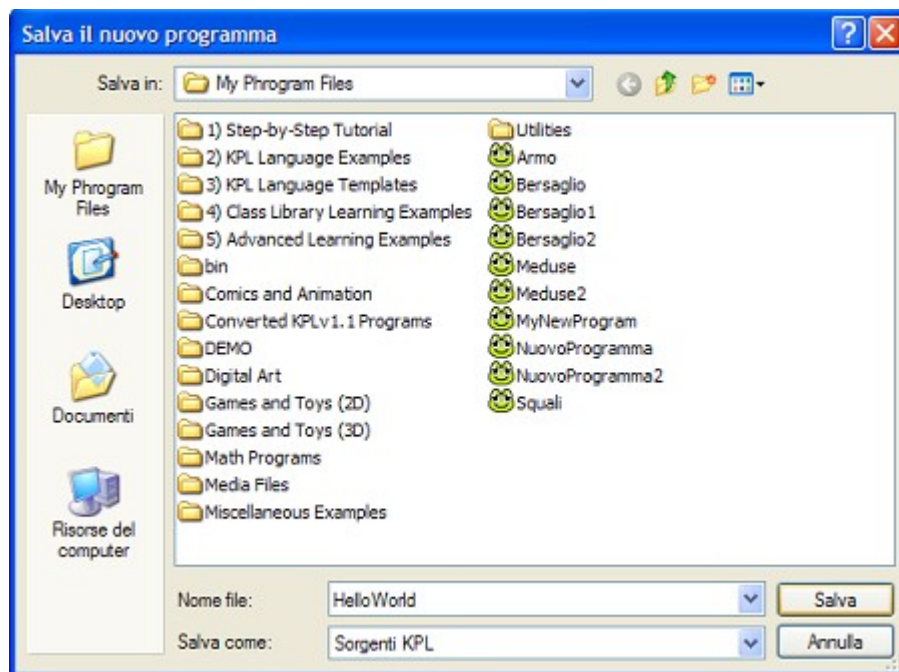


```
1 Program HelloWorld
2   Method Main()
3     Prent ("Hello World!")
4   End
5 End Program
```

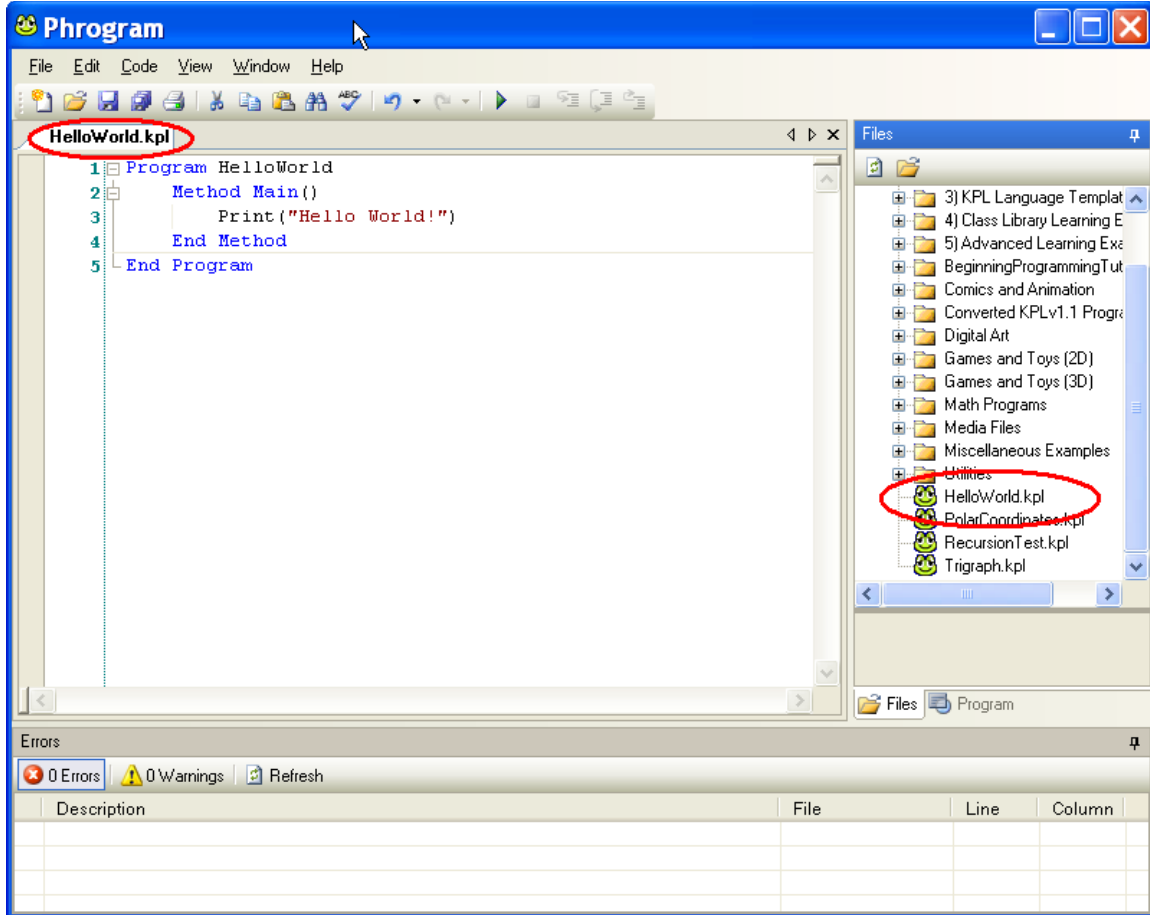
Se si incontrano errori lavorando su uno di questi esempi e' necessario confrontare con attenzione il codice scritto con quello mostrato nel Tutorial. Trovata la differenza e corretto l'errore si riuscirà ad eseguire il programma.

Tutti i programmatori qualche volta commettono errori nel loro codice, quindi non bisogna scoraggiarsi quando succede. Nessuno e' perfetto! Quando succede con il proprio codice bisogna cercare di stare calmi, mantenere la pazienza e controllare il codice attentamente - la calma e la pazienza sono il modo migliore di trovare cio' che e' sbagliato e correggerlo.

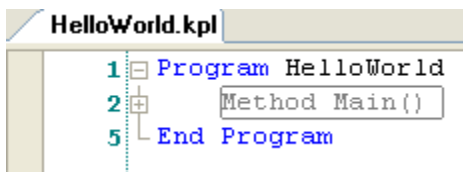
Una volta che l'esempio HelloWorld funziona clicca su **File, Salva** ed assegna il nome HelloWorld al programma come sotto:



Salvato il programma Phrogram ci sono due posti da controllare per essere sicuri:

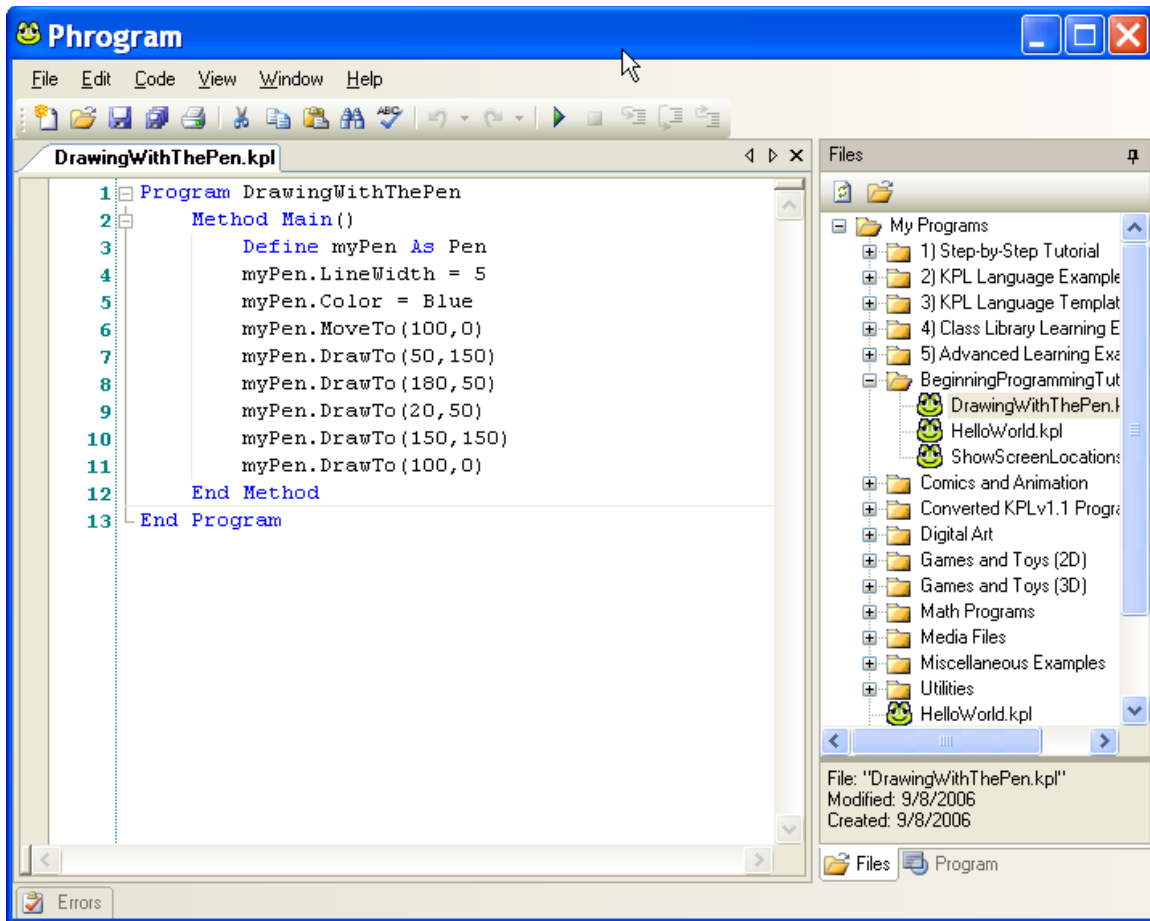


Ora che il programma e' salvato su disco puo' essere riaperto in qualsiasi momento con un doppio-click sul **Files** explorer a destra di Phrogram. Riaprendo il programma Phrogram apparira' come nella figura sotto – nessuna paura! Il codice c'e' tutto!



Cliccando sulla piccola icona + della linea 3 tutto il codice si espande in modo da vederlo tutto. Cliccando sull'icona – il codice si nasconde nuovamente. Questo non serve molto con un programma di piccole dimensioni, ma quando si scrivono programmi piu' lunghi e complessi nascondere e visualizzare il solo codice necessario e' molto utile per i programmatori.

Una volta salvato **HelloWorld**, clicca sul menu **File, Nuovo Programma**, e si vedra' un nuovo **SenzaNome.kpl** come quello iniziale. Cancella tutto il codice e scrivi il testo del secondo esempio:



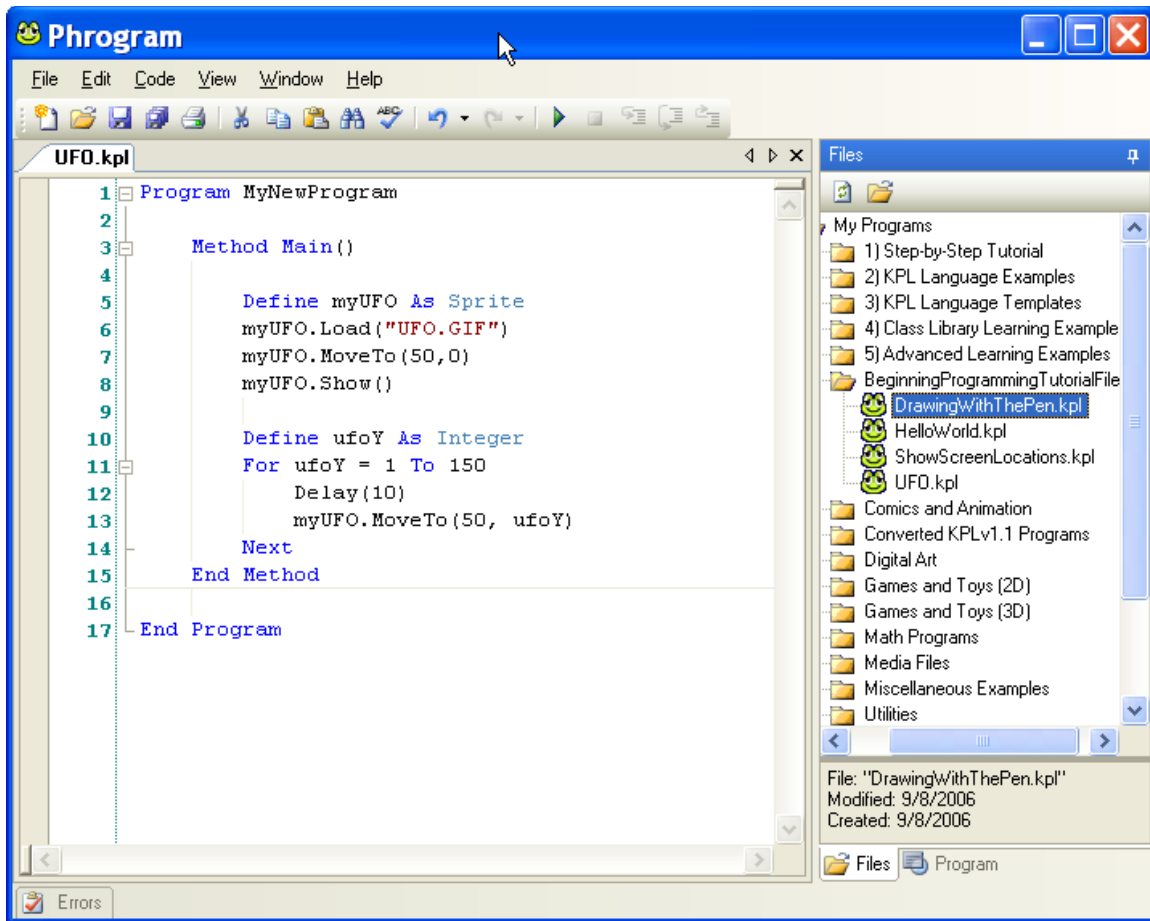
Se tutto il codice Phrogram e' stato inserito esattamente come visualizzato e quindi eseguito si vedra' una stella blu. E tu sarai diventato un programmatore di grafica! **Così si fa!**

Se trovi un errore resta calmo e paziente e cerca le differenze tra il programma e l'esempio. Quando il codice corrispondera' all'esempio funzionera' correttamente.

Non dimenticare di salvare il programma quando funziona!

E' un programma divertente per fare esperimenti. Quali altri colori stanno bene? Cosa succedera' utilizzando una penna piu' fine o piu' spessa? Come si puo' disegnare un quadrato o triangolo al posto della stella? Ed anche qualcosa di un poco piu' difficile: come si puo' rendere la stella piu' grande o piu' piccola?

Dopo aver salvato **DrawingWithThePen**, clicca su **File** menu, **Nuovo Programma**, 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:



Se il codice e' stato scritto correttamente e quindi eseguito si vedra' l'UFO volare. **Così si fa!**

Ricordate che se si ha un errore bisogna rimanere calmi e con pazienza controllare le differenze tra il programma e questo esempio. Quando si trova l'errore e lo si corregge tutti funzionera' correttamente.

Non dimenticare di salvare il programma al termine del lavoro!

Ora si possono eseguire i seguenti semplici esercizi per fare un po' di pratica di programmazione:

- Spostare piu' avanti l'UFO
- Far volare l'UFO verso destra anziche' verso il basso
- Far volare l'UFO verso il basso e verso destra nello stesso tempo

I passi successivi dopo il Tutorial

La prima cosa da fare dopo aver terminato il Tutorial e' cercare in Phrogram la directory **1) Step-by-Step Tutorial**. Ci sono 13 programmi Phrogram con i quali progredire - e' meglio studiarli nell'ordine. Infatti alcuni sono molto simili a quelli realizzati fino ad ora!

Quando si e' confidenti con i programmi in **1) Step-by-Step Tutorial**, la directory **2) Phrogram Language Examples** e' un buona guida per imparare di piu' su Phrogram.

Una volta terminati gli esempi le directory **4) Class Library Learning Examples** e **5) Advanced Learning Examples** offrono molti altri elementi.