Microsoft Small Basic

Εισαγωγή στον Προγραμματισμό

Κεφάλαιο 1

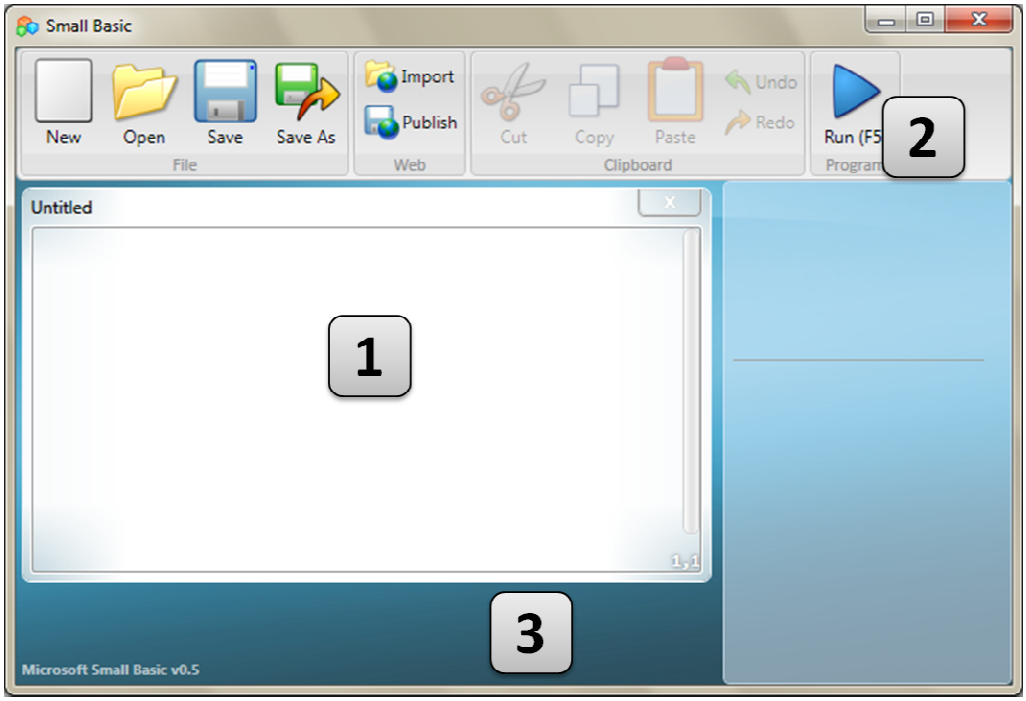
Εισαγωγή

Η Small Basic και ο Προγραμματισμός

Ο Προγραμματισμός των υπολογιστών (computers) ορίζεται ως η διαδικασία δημιουργίας λογισμικού (προγραμμάτων) ηλεκτρονικών υπολογιστών χρησιμοποιώντας γλώσσες προγραμματισμού. Ακριβώς όπως οι άνθρωποι μιλάμε και καταλαβαίνουμε αγγλικά ή ισπανικά ή γαλλικά, έτσι και οι υπολογιστές μπορούν να κατανοήσουν τα προγράμματα γραμμένα σε ορισμένες γλώσσες. Αυτές ονομάζονται γλώσσες προγραμματισμού. Στην αρχή υπήρχαν λίγες μόνο γλώσσες προγραμματισμού και ήταν πραγματικά εύκολο να τις μάθει κανείς και να τις κατανοήσει. Όμως, όπως τα computers και τα προγράμματα έγιναν όλο και πιο πολύπλοκα, έτσι και οι γλώσσες προγραμματισμού εξελίχθηκαν γρήγορα, συμπεριλαμβάνοντας όλο και πιο περίπλοκες έννοιες στην πορεία. Σαν αποτέλεσμα, οι πιο σύγχρονες γλώσσες προγραμματισμού και οι έννοιές τους είναι αρκετά δύσκολο να κατανοηθούν από έναν αρχάριο. Το γεγονός αυτό άρχισε να αποθαρρύνει τους ανθρώπους από τη μάθηση του προγραμματισμού των computers.
Η Small Basic είναι μια γλώσσα προγραμματισμού που έχει σχεδιαστεί για να κάνει τον προγραμματισμό εξαιρετικά εύκολο, προσιτό και διασκεδαστικό για τους αρχάριους.

Το Περιβάλλον της Small Basic

Ας αρχίσουμε με μια σύντομη εισαγωγή στο Περιβάλλον της Small Basic. Όταν ξεκινήσουμε τη Small Basic, θα δούμε ένα παράθυρο που μοιάζει με την παρακάτω εικόνα:



Εικόνα 1 – Το Περιβάλλον της Small Basic

Αυτό είναι το Περιβάλλον της Small Basic, στο οποίο θα γράψετε και θα εκτελέσετε(«τρέξετε», run) τα προγράμματα σε Small Basic. Το περιβάλλον αυτό έχει πολλά διαφορετικά στοιχεία τα οποία προσδιορίζουμε με αριθμούς.

Έτσι, ο **Επεξεργαστής** (**Editor**), που εμφανίζεται στο [1], είναι εκεί που θα γράψουμε τα προγράμματά μας σε Small Basic. Όταν ανοίγετε ένα παράδειγμα προγράμματος ή ένα προηγούμενο αποθηκευμένο πρόγραμμα, αυτό θα εμφανιστεί στον Επεξεργαστή. Εκεί θα μπορείτε να το τροποποιήσετε και να το αποθηκεύσετε για μελλοντική χρήση. Μπορείτε επίσης να ανοίξετε και να δουλέψετε με περισσότερα από ένα προγράμματα κάθε φορά. Το κάθε πρόγραμμα με το οποίο δουλεύετε θα εμφανίζεται σε διαφορετικό Επεξεργαστή. Ο Επεξεργαστής που θα περιέχει το πρόγραμμα με το οποίο τη συγκεκριμένη στιγμή δουλεύετε θα ονομάζεται *ενεργός επεξεργαστής*.

Η **Γραμμή Εργαλείων** (**Toolbar**), που εμφανίζεται στο [2], χρησιμοποιείται για να θέτουμε εντολές είτε στον ενεργό επεξεργαστή είτε στο περιβάλλον. Θα μάθουμε για τις διάφορες εντολές της γραμμής εργαλείων στη συνέχεια

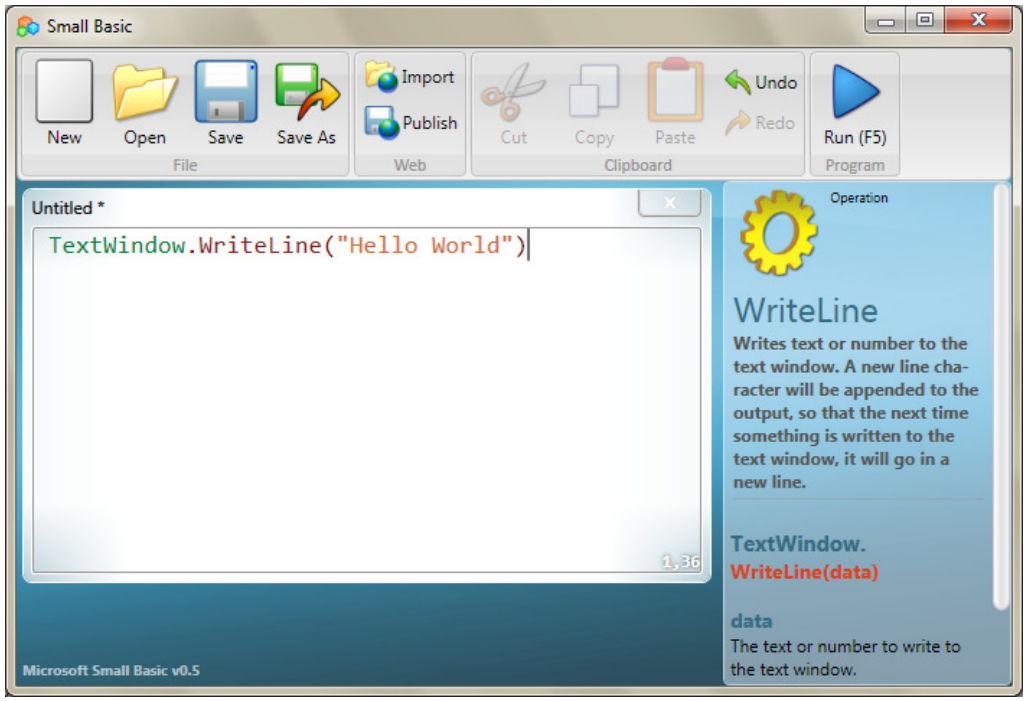
Η Επιφάνεια (**Surface**), που εμφανίζεται στο [3] είναι το μέρος όπου εμφανίζονται όλα τα παράθυρα των επεξεργαστών.

Το πρώτο μας πρόγραμμα

Τώρα που εξοικειωθήκατε με το περιβάλλον της Small Basic Environment, θα προχωρήσουμε και θα αρχίσουμε να προγραμματίζουμε. Όπως ήδη αναφέραμε παραπάνω, τα προγράμματά μας θα τα γράψουμε στον επεξεργαστή (editor). Ας γράψουμε λοιπόν την παρακάτω γραμμή στον επεξεργαστή:

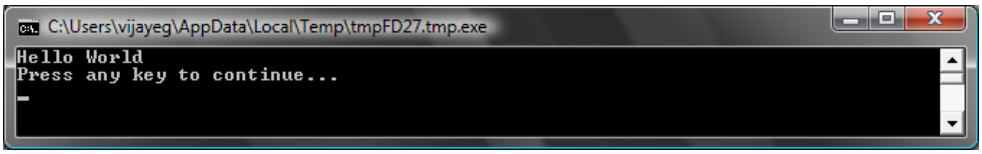
```
TextWindow.WriteLine("Hello World")
```

Αυτό είναι το πρώτο μας πρόγραμμα σε Small Basic. Αν το έχετε γράψει σωστά, θα πρέπει να βλέπετε κάτι παρόμοιο με την παρακάτω εικόνα:



Εικόνα 2 – Το πρώτο πρόγραμμα

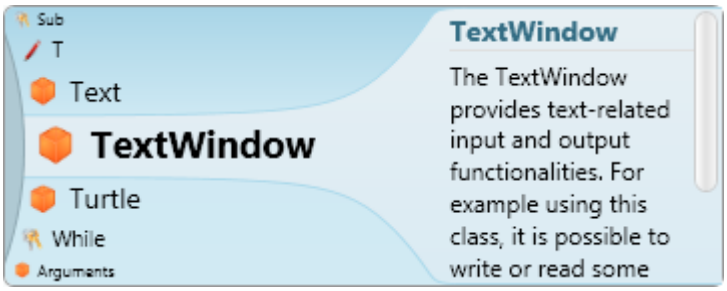
Τώρα που γράψαμε το πρώτο μας πρόγραμμα, ας προχωρήσουμε να το εκτελέσουμε και να δούμε τι θα γίνει. Μπορούμε να εκτελέσουμε τα προγράμματά μας πατώντας στο κουμπί *Run* της γραμμής εργαλείων ή πατώντας το πλήκτρο F5 στο πληκτρολόγιο. Αν όλα πάνε καλά, το πρόγραμμά μας θα εκτελεστεί και τα αποτελέσματα θα είναι όπως παρακάτω:



Εικόνα 3 – Αποτέλεσμα του πρώτου προγράμματος

Συγχαρητήρια! Μόλις γράψατε και εκτελέσατε το πρώτο σας πρόγραμμα σε Small Basic. Ένα πολύ μικρό και απλό πρόγραμμα, αλλά παρόλα αυτά ένα μεγάλο βήμα για να γίνετε ένας αληθινός προγραμματιστής ηλεκτρονικών υπολογιστών! Έχουμε όμως ακόμα μια λεπτομέρεια να καλύψουμε πριν συνεχίσουμε με μεγαλύτερα προγράμματα. Πρέπει να καταλάβουμε τι ακριβώς έγινε – τι ακριβώς είπαμε στο computer να κάνει και πως ακριβώς το computer ήξερε τι να κάνει? Στο επόμενο κεφάλαιο θα αναλύσουμε το πρόγραμμα που μόλις γράψαμε ώστε να κατανοήσουμε τι ακριβώς συνέβη.

*Καθώς θα πληκτρολογούσατε το πρώτο σας πρόγραμμα, θα προσέξατε ίσως ότι εμφανίστηκε ένα παράθυρο με μία λίστα από θέματα (*Εικόνα 4*). Αυτό λέγεται "intellisense" και μας βοηθά να πληκτρολογούμε ταχύτερα. Μπορείτε να διατρέξετε τη λίστα με τα βελάκια πάνω/κάτω και μόλις βρείτε κάτι που θέλετε, πατάτε το πλήκτρο Enter για να εισάγετε τη γραμμή που επιλέξατε μέσα στο πρόγραμμά σας.*



Εικόνα 4 - Intellisense

Αποθηκεύοντας το πρόγραμμά μας

Αν θέλετε να κλείσετε τη Small Basic και να επιστρέψετε αργότερα για να συνεχίσετε στο πρόγραμμα που μόλις γράψατε, μπορείτε να αποθηκεύσετε το πρόγραμμα. Στην πραγματικότητα, είναι πολύ καλή πρακτική να αποθηκεύετε τα προγράμματά σας από καιρό σε καιρό, έτσι ώστε να μην χάνετε πληροφορίες σε περίπτωση ατυχήματος ή διακοπής ρεύματος. Μπορούμε να αποθηκεύσουμε το τρέχον πρόγραμμα, είτε κάνοντας κλικ στο εικονίδιο "save" στη γραμμή εργαλείων ή με τη χρήση της συντόμευσης «Ctrl + S" (πατήστε το πλήκτρο S, ενώ κρατάτε πατημένο το πλήκτρο Ctrl).

Κεφάλαιο 2

Κατανοώντας το Πρώτο μας Πρόγραμμα

Τι πραγματικά είναι ένα πρόγραμμα ηλεκτρονικού υπολογιστή?

Ένα πρόγραμμα είναι ένα σύνολο οδηγιών για τον υπολογιστή. Αυτές οι οδηγίες λένε στον υπολογιστή τι ακριβώς πρέπει να κάνει, και ο υπολογιστής ακολουθεί πάντα τις οδηγίες αυτές. Ακριβώς όπως και οι άνθρωποι, οι υπολογιστές μπορούν να ακολουθήσουν τις οδηγίες μόνο αν αυτές περιγράφονται σε μια γλώσσα που κατανοούν. Αυτές ονομάζονται γλώσσες προγραμματισμού. Υπάρχουν πάρα πολλές γλώσσες που ο υπολογιστής μπορεί να κατανοήσει και η Small Basic είναι μία από αυτές.

Φανταστείτε μια συνομιλία που συμβαίνει μεταξύ εσένα και του φίλου σου. Εσύ και ο φίλος σου θα χρησιμοποιήσετε λέξεις οι οποίες θα έχουν οργανωθεί σε προτάσεις για να μεταφέρουν πληροφορίες πέρα δώθε. Ομοίως, οι γλώσσες προγραμματισμού περιλαμβάνουν συλλογές από λέξεις που μπορούν να οργανωθούν σε προτάσεις που μεταδίδουν πληροφορίες στον υπολογιστή. Και τα προγράμματα είναι βασικά σύνολα προτάσεων (μερικές φορές λίγα και μερικές φορές πολλές χιλιάδες), που μαζί κάνουν νόημα τόσο για τον προγραμματιστή όσο και τον υπολογιστή.

Υπάρχουν πολλές γλώσσες προγραμματισμού που καταλαβαίνουν οι υπολογιστές. Οι Java, C++, Python, VB, κτλ. είναι πανίσχυρες σύγχρονες γλώσσες που χρησιμοποιούνται για την ανάπτυξη περίπλοκων προγραμάτων.

Προγράμματα σε Small Basic

Ένα τυπικό πρόγραμμα σε Small Basic αποτελείται από ένα σύνολο *δηλώσεων* (*statements*). Κάθε γραμμή του προγράμματος αποτελεί μια δήλωση και κάθε δήλωση είναι μία εντολή για τον υπολογιστή. Όταν ζητάμε από τον υπολογιστή να εκτελέσει ένα πρόγραμμα Small Basic, ο υπολογιστής παίρνει το πρόγραμμα και διαβάζει την πρώτη δήλωση. Καταλαβαίνει τι προσπαθούμε να πούμε και στη συνέχεια εκτελεί τις οδηγίες μας. Μόλις εκτελέσει την πρώτη μας δήλωση, επιστρέφει πίσω στο πρόγραμμα για να διαβάσει και να εκτελέσει τη δεύτερη γραμμή. Συνεχίζει κατ' αυτόν τον τρόπο έως ότου φθάσει το τέλος του προγράμματος. Εκεί είναι που τελειώνει η εκτέλεση του προγράμματός μας.

Πίσω στο πρώτο μας πρόγραμμα

Το πρώτο μας πρόγραμμα που γράψαμε:

```
TextWindow.WriteLine("Hello World")
```

Πρόκειται για ένα πολύ απλό πρόγραμμα που αποτελείται από μια μόνο *δήλωση* (*statement*). Η δήλωση αυτή λέει στον υπολογιστή να γράψει μία γραμμή κειμένου "**Hello World**", μέσα στο Παράθυρο Κειμένου (Text Window).

Αυτό ουσιαστικά στο μυαλό του υπολογιστή μεταφράζεται σε:

```
Γράψε Hello World
```

Ίσως να έχετε ήδη προσέξει ότι η δήλωση μπορεί με τη σειρά της να διασπαστεί σε μικρότερα τμήματα (segments), όπως περίπου οι προτάσεις μπορούν να χωριστούν σε λέξεις. Στην πρώτη δήλωση έχουμε 3 διαφορετικά τμήματα:

a) TextWindow
b) WriteLine
c) "Hello World"

Η τελεία, οι παρενθέσεις και τα εισαγωγικά είναι όλα σημεία στίξης που πρέπει να τοποθετηθούν στις κατάλληλες θέσεις μέσα στη δήλωση ώστε να καταλάβει ο υπολογιστής μας τι θέλουμε να του πούμε.

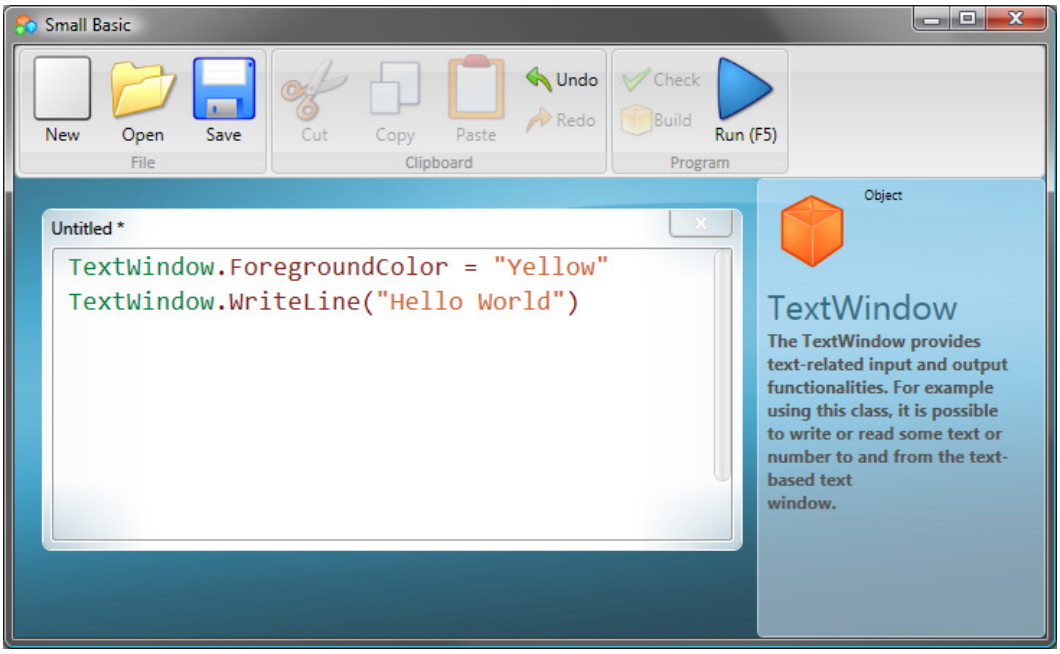
Ίσως να θυμάστε το μαύρο παράθυρο που εμφανίστηκε όταν τρέξαμε το πρώτο μας πρόγραμμα. Αυτό το μαύρο παράθυρο ονομάζεται Παράθυρό Κειμένου (TextWindow) ή μερικές φορές αναφέρεται και ως κονσόλα (Console). Εκεί καταλήγει το αποτέλεσμα της εκτέλεσης του προγράμματος. Το TextWindow στο πρόγραμμά μας ονομάζεται *αντικείμενο* (*object*). Υπάρχει ένας αριθμός τέτοιων αντικειμένων που μας είναι διαθέσιμα να χρησιμοποιήσουμε στα προγράμματά μας. Μπορούμε να εκτελέσουμε πολλές διαφορετικές *λειτουργίες* (*operations*) σε αυτά τα αντικείμενα. Έχουμε ήδη χρησιμοποιήσει τη λειτουργία WriteLine στο πρόγραμμά μας. Μπορεί επίσης να έχετε παρατηρήσει ότι η λειτουργία WriteLine ακολουθείται από το Hello World μέσα σε εισαγωγικά. Το κείμενο αυτό έχει δοθεί σαν στοιχείο στη λειτουργία WriteLine, η οποία στη συνέχεια μας το εμφανίζει. Αυτό ονομάζεται *εισαγωγή στοιχείων* (input) στη λειτουργία. Ορισμένες λειτουργίες λαμβάνουν ένα ή πολλά στοιχεία ενώ άλλες λειτουργίες δεν λαμβάνουν κανένα.

Τα σημεία στίξης, όπως τα εισαγωγικά, τα κενά και οι παρενθέσεις, είναι πολύ χρήσιμα σε ένα πρόγραμμα υπολογιστή. Ανάλογα με τη θέση τους και τον αριθμό τους μέσα στον κώδικα, διαφέρει και η σημασία τους

Το Δεύτερό μας Πρόγραμμα

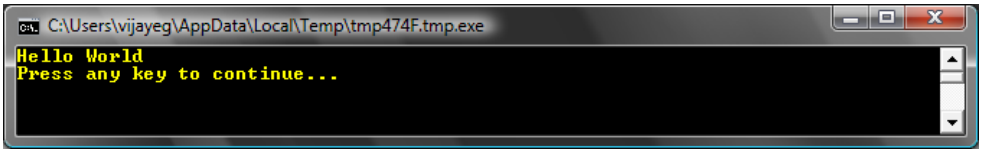
Τώρα που καταλάβαμε το πρώτο μας πρόγραμμα, ας προχωρήσουμε και να το κάνουμε λίγο φανταχτερό, προσθέτοντας λίγο χρώμα.

```
TextWindow.ForegroundColor = "Yellow"
TextWindow.WriteLine("Hello World")
```



Εικόνα 5 – Προσθέτοντας Χρώματα

Όταν εκτελέσετε το παραπάνω πρόγραμμα θα προσέξετε ότι εμφανίζει την ίδια φράση "Hello World" μέσα στο Παράθυρο Κειμένου (TextWindow), αλλά αυτή τη φορά την εμφανίζει κίτρινη αντί για γκρι που έκανε νωρίτερα.



Εικόνα 6 - Hello World σε κίτρινο

Προσέξτε τη νέα δήλωση (statement) που προσθέσαμε στο αρχικό μας πρόγραμμα. Χρησιμοποιεί μία νέα λέξη, τη *ForegroundColor* στην οποία δίνουμε την τιμή *"Yellow."* Αυτό σημαίνει ότι έχουμε *αναθέσει* (*assigned*) την τιμή "Yellow" στο *ForegroundColor.* Τώρα, η διαφορά μεταξύ της ForegroundColor και της λειτουργίας WriteLine είναι ότι στην ForegroundColor δεν δώσαμε κάποιο στοιχείο εισόδου αλλά ούτε και χρειάστηκαν καθόλου παρενθέσεις. Αντίθετα, γράψαμε το σύμβολο της *εξίσωσης* ("=") ακολουθούμενο από μια λέξη. Ορίζουμε το ForegroundColor σαν μια *Ιδιότητα* (*Property*) του TextWindow. Παρακάτω δίνεται μια λίστα με όλες τις έγκυρες τιμές για την ιδιότητα ForegroundColor. Προσπαθήστε να αντικαταστήσετε το "Yellow" με μία από αυτές τις τιμές και δείτε τα αποτελέσματα– μην ξεχάσετε να χρησιμοποιήσετε εισαγωγικά, είναι απαραίτητα σημεία στίξης.

```
Black
Blue
Cyan
Gray
```

```
Green
Magenta
Red
White
```

```
Yellow
DarkBlue
DarkCyan
DarkGray
```

```
DarkGreen
DarkMagenta
DarkRed
DarkYello
```

Κεφάλαιο 3

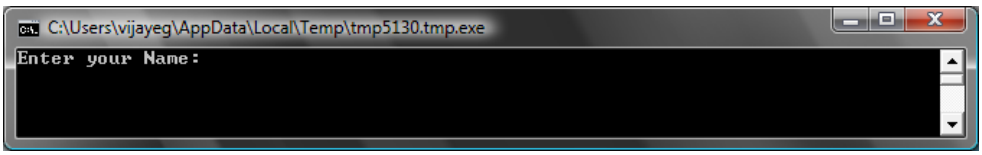
Εισαγωγή στις Μεταβλητές

Χρησιμοποιώντας Μεταβλητές στο πρόγραμμά μας

Δεν θα ήταν ωραίο αν το πρόγραμμά μας μπορούσε πραγματικά να πει "Hello" με το όνομα του χρήστη, αντί να λέει γενικά "Hello World"; Για να γίνει αυτό θα πρέπει προηγουμένως να ζητήσουμε το όνομα του χρήστη, στη συνέχεια να το αποθηκεύσουμε κάπου και στη συνέχεια να εμφανίσουμε το "Hello" ακολουθούμενο από το όνομα του χρήστη. Ας δούμε πώς μπορούμε να το κάνουμε αυτό:

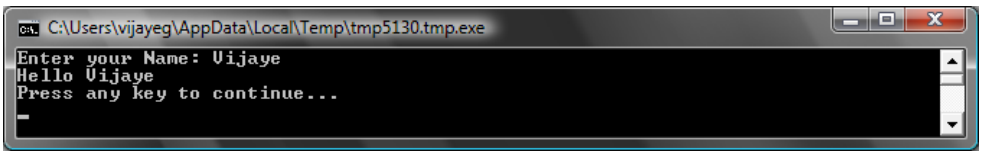
```
TextWindow.Write("Enter your Name: ")
name = TextWindow.Read()
TextWindow.WriteLine("Hello " + name)
```

Όταν πληκτρολογήσετε και εκτελέσετε αυτό το πρόγραμμα, θα δείτε ένα αποτέλεσμα σαν το παρακάτω:



Εικόνα 7 – Ρωτώντας το όνομα του χρήστη

Και όταν πληκτρολογήσετε το όνομά σας και πατήσετε το ENTER, θα δείτε το παρακάτω αποτέλεσμα:



Εικόνα 8 – Ένας θερμός χαιρετισμός...

Αν τώρα εκτελέσετε το πρόγραμμα ξανά, θα σας γίνει η ίδια ερώτηση και πάλι, Μπορείτε να δώσετε διαφορετικό όνομα αυτή τη φορά και ο υπολογιστής θα εμφανίσει το Hello με το νέο αυτό όνομα.

Ανάλυση του προγράμματος

Στο πρόγραμμα που μόλις εκτελέσατε, η γραμμή που ίσως σας κέντρισε το ενδιαφέρον είναι αυτή:

```
name = TextWindow.Read()
```

Η *Read()* μοιάζει ακριβώς σαν την *WriteLine()*, αλλά χωρίς στοιχεία εισόδου (inputs). Είναι μια λειτουργία (operation) και βασικά λέει στον υπολογιστή να περιμένει το χρήστη να πληκτρολογήσει κάτι και μετά να πατήσει το πλήκτρο ENTER. Μόλις ο χρήστης πατήσει το ENTER, παίρνει ότι είχε πληκτρολογήσει ο χρήστης και το επιστρέφει στο πρόγραμμα. Το ενδιαφέρον σημείο είναι πως ο,τιδήποτε πληκτρολόγησε ο χρήστης έχει τώρα αποθηκευθεί σε μια *μεταβλητή* (*variable*) που ονομάζεται **name**. Μια *μεταβλητή* (*variable*) είναι το μέρος όπου μπορούμε να αποθηκεύσουμε προσωρινά κάποιες τιμές και να τις χρησιμοποιήσουμε αργότερα. Στην παραπάνω γραμμή, η **name** χρησιμοποιήθηκε για να αποθηκεύσουμε το όνομα του χρήστη.

Η επόμενη γραμμή είναι επίσης ενδιαφέρουσα:

```
TextWindow.WriteLine("Hello " +
name)
```

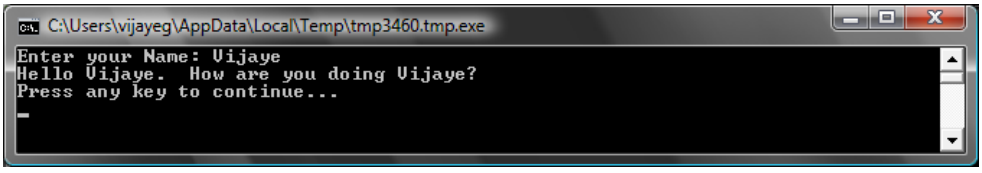
Η Write, όπως και η WriteLine, είναι άλλη μια λειτουργία (operation) για την κονσόλα. Με τη Write μπορούμε να εμφανίσουμε κείμενο στην κονσόλα αλλά ταυτόχρονα μας επιτρέπει το κείμενο που θα ακολουθήσει να είναι στην ίδια γραμμή με το κείμενο που γράψαμε.

Εδώ είναι το σημείο που χρησιμοποιούμε την τιμή που είχαμε αποθηκεύσει στη μεταβλητή μας, την **name**. Παίρνουμε την τιμή που έχει η **name** και την προσθέτουμε στο “Hello” και όλο μαζί το γράφουμε στο TextWindow.

Μόλις μία μεταβλητή πάρει μια τιμή, μπορούμε να τη χρησιμοποιήσουμε όσες φορές θέλουμε. Για παράδειγμα, μπορείς να κάνεις το εξής:

```
TextWindow.Write("Enter your Name: ")
name = TextWindow.Read()
TextWindow.Write("Hello " + name + ".  ")
TextWindow.WriteLine("How are you doing " + name + "?")
```

Και θα δεις το παρακάτω αποτέλεσμα:



Εικόνα 9 – Επαναχρησιμοποιώντας μια μεταβλητή

Κανόνες για ονομασίες Μεταβλητών

Στις μεταβλητές αναθέτουμε ονόματα και με αυτό τον τρόπο τις προσδιορίζουμε. Υπάρχουν ορισμένοι απλοί κανόνες και μερικές πολύ καλές κατευθυντήριες γραμμές για την ονομασία των μεταβλητών αυτών, όπως:

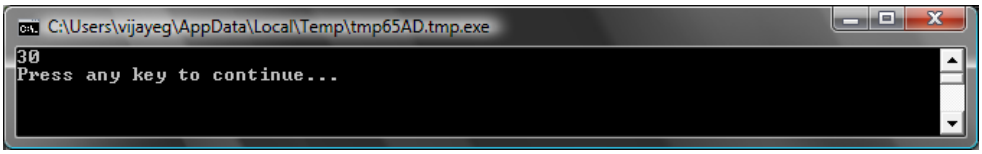
1. Η ονομασία πρέπει να ξεκινά με γράμμα και δεν θα πρέπει να είναι κάποια από τις λέξεις-κλειδιά (keywords) όπως **if**, **for**, **then**, κλπ.
2. Η ονομασία μπορεί να περιέχει οποιονδήποτε συνδυασμό από γράμματα, ψηφία και υπογραμμίσεις (underscores).
3. Είναι χρήσιμο να ονοματίζουμε τις μεταβλητές σύμφωνα με τη χρήση τους – εφόσον οι μεταβλητές μπορεί να είναι όσο μεγάλες θέλουμε, μπορούμε να χρησιμοποιήσουμε την ονομασία των μεταβλητών για να περιγράψουμε τη χρήση τους.

Παίζοντας με τους Αριθμούς

Μόλις είδαμε πώς μπορούμε να χρησιμοποιήσουμε μεταβλητές για να αποθηκεύσουμε το όνομα του χρήστη. Στα επόμενα προγράμματα θα δούμε πως μπορούμε να αποθηκεύσουμε και να διαχειριστούμε αριθμούς μέσα στις μεταβλητές. Ας αρχίσουμε λοιπόν με ένα πραγματικά απλό πρόγραμμα:

```
number1 = 10
number2 = 20
number3 = number1 + number2
TextWindow.WriteLine(number3)
```

Όταν εκτελέσετε αυτό το πρόγραμμα θα έχετε το παρακάτω αποτέλεσμα:



Εικόνα 10 – Προσθέτοντας δύο αριθμούς

Στην πρώτη γραμμή του προγράμματος αναθέτετε στη μεταβλητή **number1** την τιμή 10. Και στη δεύτερη γραμμή, αναθέτετε στη μεταβλητή **number2** την τιμή 20. Στην Τρίτη γραμμή προσθέτετε τις μεταβλητές **number1** και **number2** και στη συνέχεια το αποτέλεσμα της πρόσθεσης το αναθέτετε στη μεταβλητή **number3**. Έτσι, στην περίπτωση αυτή η μεταβλητή **number3** θα έχει την τιμή 30. Και αυτό είναι το αποτέλεσμα που εμφανίζεται στο TextWindow.

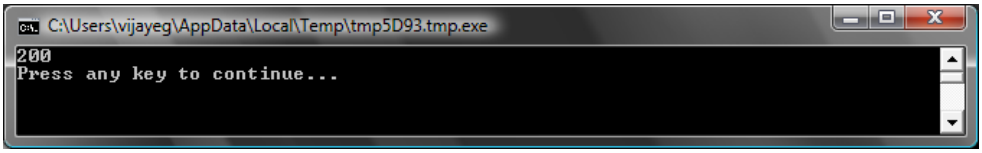
> Παρατηρείστε ότι οι αριθμοί δεν περικλείονται από εισαγωγικά. Για τους αριθμούς, τα εισαγωγικά δεν είναι απαραίτητα.Εισαγωγικά χρησιμοποιούμε μόνο όταν γράφουμε κείμενο

Τώρα, ας αλλάξουμε λίγο το πρόγραμμα και να δούμε τα αποτελέσματα:

```
number1 = 10
```

```
number2 = 20
number3 = number1 * number2
TextWindow.WriteLine(number3)
```

Το παραπάνω πρόγραμμα θα πολλαπλασιάσει την **number1** με την **number2** και θα αποθηκεύσει το αποτέλεσμα στην **number3**. Και το αποτέλεσμα του προγράμματος μπορείτε να το δείτε παρακάτω:



Εικόνα 11 – Πολλαπλασιάζοντας δύο αριθμούς

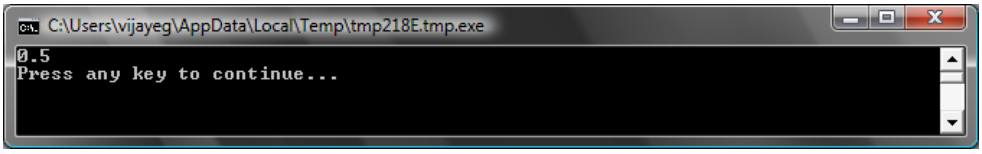
Με παρόμοιο τρόπο μπορούμε να αφαιρέσουμε ή να διαιρέσουμε αριθμούς. Για την αφαίρεση:

```
number3 = number1 - number2
```

Το σύμβολο της διαίρεσης είναι το '/'. Το πρόγραμμα θα είναι κάπως έτσι:

```
number3 = number1 / number2
```

Και το αποτέλεσμα της διαίρεσης θα είναι:



Εικόνα 12 – Διαιρώντας δύο αριθμούς

Ένας απλός μετατροπέας θερμοκρασίας

Για το επόμενό μας πρόγραμμα θα χρησιμοποιήσουμε τον τύπο $℃ = \frac{5(°F - 32)}{9}$ που μετατρέπει τη θερμοκρασία από βαθμούς Fahrenheit σε βαθμούς Celsius.

Αρχικά, θα πάρουμε τη θερμοκρασία σε Fahrenheit από το χρήστη και θα την αποθηκεύσουμε σε μια μεταβλητή. Υπάρχει μια ειδική λειτουργία που μας επιτρέπει να παίρνουμε αριθμούς από το χρήστη και αυτή είναι η **TextWindow.ReadNumber**.

```
TextWindow.Write("Enter temperature in Fahrenheit: ")
fahr = TextWindow.ReadNumber()
```

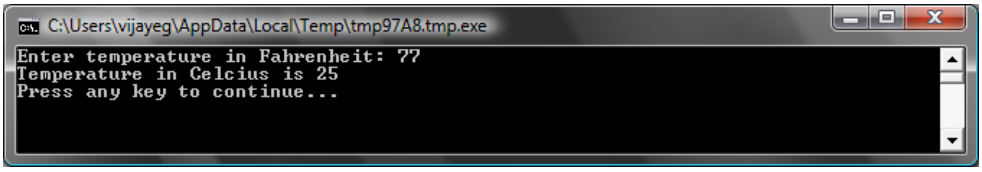
Αφού έχουμε αποθηκεύσει τη θερμοκρασία Fahrenheit σε μια μεταβλητή, μπορούμε να τη μετατρέψουμε σε Celsius ως εξής:

```
celsius = 5 * (fahr - 32) / 9
```

Οι παρενθέσεις λένε στον υπολογιστή να υπολογίσει πρώτα το κομμάτι **fahr – 32** και μετά να συνεχίσει με τους υπόλοιπους υπολογισμούς. Τώρα, το μόνο που μένει να κάνουμε είναι να εμφανίσουμε τα αποτελέσματα στο χρήστη. Βάζοντάς τα όλα μαζί έχουμε το παρακάτω πρόγραμμα:

```
TextWindow.Write("Enter temperature in Fahrenheit: ")
fahr = TextWindow.ReadNumber()
celsius = 5 * (fahr - 32) / 9
TextWindow.WriteLine("Temperature in Celsius is " + celsius)
```

Και το αποτέλεσμα του προγράμματος θα είναι:



Εικόνα 13 – Μετατροπή Θερμοκρασίας

Κεφάλαιο 4

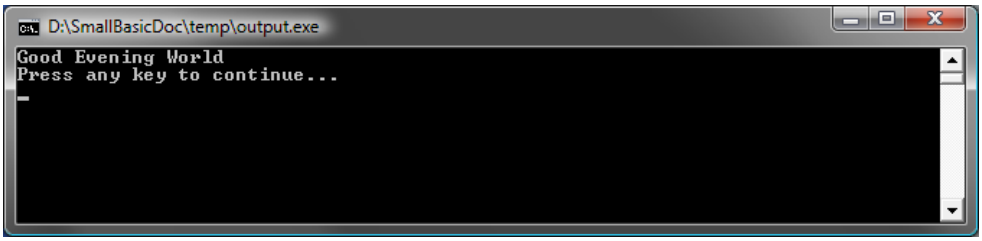
Συνθήκες και Διακλαδώσεις

Γυρνώντας στο πρώτο μας πρόγραμμα, δεν θα ήταν πιο ωραίο αν αντί να λέγαμε το γενικό «Γεια σου Κόσμε» (*Hello World*), να μπορούσαμε να πούμε «Καλημέρα Κόσμε» (*Good Morning World*) ή «Καλησπέρα Κόσμε» (*Good Evening World*) ανάλογα με το τί ώρα είναι εκείνη τη στιγμή; Για το επόμενό μας πρόγραμμα, θα κάνουμε τον υπολογιστή να λέει «Καλημέρα Κόσμε» (*Good Morning World*) αν η ώρα είναι πριν τις 12 το μεσημέρι και να λέει «Καλησπέρα Κόσμε» (*Good Evening World*) αν είναι μετά τις 12 το μεσημέρι!

```
If (Clock.Hour < 12) Then
  TextWindow.WriteLine("Good Morning World")
EndIf
If (Clock.Hour >= 12) Then
  TextWindow.WriteLine("Good Evening World")
EndIf
```

Αναλόγως με το πότε εκτελείτε το πρόγραμμα, θα δείτε ένα από τα δύο παρακάτω αποτελέσματα:

Εικόνα 14 – Καλημέρα Κόσμε



Εικόνα 15 – Καλησπέρα Κόσμε

Ας αναλύσουμε τις πρώτες τρεις γραμμές του προγράμματος. Θα έχετε ήδη καταλάβει ότι οι γραμμές αυτές λένε στον υπολογιστή ότι αν η ώρα είναι μικρότερη από τις 12 να τυπώσει το «Καλημέρα Κόσμε» (“Good Morning World”). Οι λέξεις **If**, **Then** και **EndIf** είναι ειδικές λέξεις που τις καταλαβαίνει ο υπολογιστής όταν εκτελείται το πρόγραμμα. Η λέξη **If** (**Αν**) ακολουθείται πάντα από μία συνθήκη, η οποία σε αυτή την περίπτωση είναι (**Clock.Hour < 12)**. Θυμηθείτε ότι οι παρενθέσεις είναι απαραίτητες για να καταλαβαίνει ο υπολογιστής τις προθέσεις σας. Η συνθήκη ακολουθείται από το **Then** (**Τότε**) και τη λειτουργία που πρέπει να εκτελεστεί. Και μετά τη λειτουργία ακολουθεί το **EndIf** (ΤέλοςΑν) το οποίο λέει στον υπολογιστή ότι η εκτέλεση της συνθήκης έχει τελειώσει.

Στη Small Basic, μπορείτε να χρησιμοποιήσετε το αντικείμενο Clock *για να δείτε την τρέχουσα ημερομηνία και ώρα. Έχει και μια σειρά από άλλες ιδιότητες που σας επιτρέπουν να δείτε ξεχωριστά την τρέχουσα ημέρα (Day), μήνα (Month), έτος (Year), ώρα(Hour), λεπτά (Minutes)και δευτερόλεπτα (Seconds).*

Μεταξύ του **then** και του **EndIf** θα μπορούσαν να υπάρχουν περισσότερες από μία λειτουργίες και ο υπολογιστής θα τις εκτελέσει όλες αν η συνθήκη είναι έγκυρη. Για παράδειγμα, θα μπορούσατε να γράψετε κάτι σαν αυτό:

```
If (Clock.Hour < 12) Then
  TextWindow.Write("Good Morning. ")
  TextWindow.WriteLine("How was breakfast?")
EndIf
```

Ειδάλλως (Else)

Στο πρόγραμμα στην αρχή του κεφαλαίου ίσως να προσέξατε ότι η δεύτερη συνθήκη είναι κάπως περιττή. Η ώρα (τιμή της **Clock.Hour**) θα είναι είτε πριν τις 12 είτε μετά (δεν μπορεί να είναι κάτι άλλο). Στην πραγματικότητα δηλαδή δεν χρειαζόμαστε το δεύτερο έλεγχο. Σε τέτοιες περιπτώσεις λοιπόν, μπορούμε να συντομεύσουμε τις δύο δηλώσεις **if..then..endif** σε μόνο μία χρησιμοποιώντας μία νέα λέξη, την **else** (**ειδάλλως**).

Αν ξαναγράφαμε το πρόγραμμα χρησιμοποιώντας την **else**, θα έδειχνε κάπως έτσι:

```
If (Clock.Hour < 12) Then
  TextWindow.WriteLine("Good Morning World")
Else
```

```
  TextWindow.WriteLine("Good Evening World")
EndIf
```

Το πρόγραμμα αυτό θα κάνει ακριβώς το ίδιο με το πρώτο πρόγραμμα, το οποίο μας μαθαίνει ένα πολύ σημαντικό μάθημα για τον προγραμματισμό των ηλεκτρονικών υπολογιστών:

> *Στον προγραμματισμό υπάρχουν συνήθως πολλοί τρόποι για να κάνουμε το ίδιο πράγμα. Κάποιες φορές ο ένας τρόπος έχει πιο πολύ νόημα από τον άλλο τρόπο. Η επιλογή είναι θέμα του προγραμματιστή. Καθώς θα γράφετε όλο και περισσότερα προγράμματα και θα αποκτάτε εμπειρία, θα αρχίσετε να εντοπίζετε αυτές τις διαφορετικές τεχνικές όπως και τα πλεονεκτήματα και μειονεκτήματά τους.*

Εσοχές παραγράφων (Indentation)

Σε όλα τα παραδείγματα μπορείτε να δείτε ότι οι δηλώσεις μεταξύ των *If, Else* και *EndIf* έχουν γραφτεί με εσοχή. Η στοίχιση αυτή δεν είναι απαραίτητη, ο υπολογιστής θα καταλάβει το πρόγραμμά μας και χωρίς αυτές. Παρ' όλα αυτά, οι εσοχές βοηθούν εμάς τους προγραμματιστές να καταλάβουμε ευκολότερα τη δομή του προγράμματος για αυτό και συνήθως θεωρείται καλή πρακτική να βάζουμε εσοχές στις δηλώσεις που αποτελούν υποσύνολα άλλων εντολών.

Άρτιοι ή Περιττοί

Τώρα που γνωρίζετε πλέον τις δηλώσεις **If..Then..Else..EndIf**, ας φτιάξουμε ένα πρόγραμμα που δίνοντάς του έναν αριθμό θα μας λέει αν είναι άρτιος ή περιττός!

```
TextWindow.Write("Enter a number: ")
num = TextWindow.ReadNumber()
remainder = Math.Remainder(num, 2)
If (remainder = 0) Then
  TextWindow.WriteLine("The number is Even")
Else
  TextWindow.WriteLine("The number is Odd")
EndIf
```

Όταν εκτελέσετε αυτό το πρόγραμμα, θα δείτε ένα αποτέλεσμα σαν και αυτό:

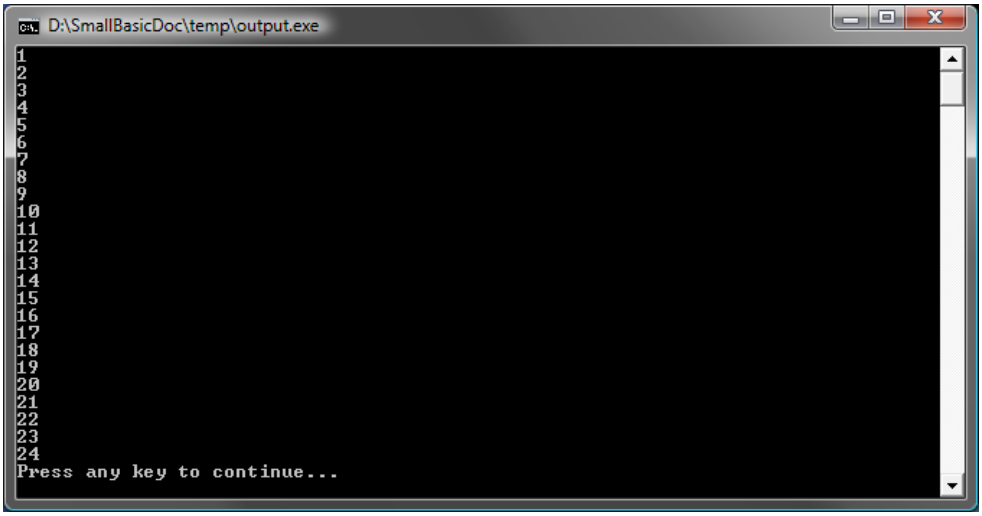
Εικόνα 16 – Άρτιος ή Περιττός

Στο πρόγραμμα αυτό, γνωρίσαμε μία άλλη χρήσιμη λειτουργία, την **Math.Remainder**. Και ναι, όπως έχετε ήδη καταλάβει, η **Math.Remainder** διαιρεί τον πρώτο αριθμό με τον δεύτερο αριθμό και μας επιστρέφει το υπόλοιπο της διαίρεσης.

Διακλάδωση

Θυμηθείτε το δεύτερο κεφάλαιο όπου μάθαμε ότι ο υπολογιστής εκτελεί μία εντολή του προγράμματος κάθε φορά και με σειρά από πάνω προς τα κάτω. Ωστόσο, υπάρχει μια ειδική δήλωση που μπορεί να κάνει τον υπολογιστή να μεταφερθεί σε μία άλλη δήλωση με διαφορετική σειρά. Ας ρίξουμε μια ματιά στο παρακάτω πρόγραμμα:

```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
  Goto start
EndIf
```



Εικόνα 17 – Χρησιμοποιώντας την Goto

Στο παραπάνω πρόγραμμα, αναθέσαμε την τιμή 1 στη μεταβλητή **i**. Και μετά προσθέσαμε μία δήλωση που στο τέλος έχει την άνω και κάτω τελεία (:)

```
start:
```

Αυτό ονομάζεται *ετικέτα* (*label*). Οι ετικέτες είναι σαν σελιδοδείκτες που καταλαβαίνει ο υπολογιστής. Μπορείς να δώσεις ότι όνομα θέλεις σε μία ετικέτα και να έχεις όσες ετικέτες θέλεις μέσα στο πρόγραμμά σου, αρκεί να έχουν διαφορετικό όνομα μεταξύ τους.

Μια άλλη ενδιαφέρουσα δήλωση είναι η εξής:

```
i = i + 1
```

Αυτό λέει στον υπολογιστή να προσθέσει την τιμή 1 στη μεταβλητή **i** και το αποτέλεσμα να το αναθέσεις πάλι στη μεταβλητή **i**. Έτσι, αν πριν την παραπάνω δήλωση η τιμή της μεταβλητής **i** ήταν 1, μετά την εκτέλεση της δήλωση αυτής θα είναι 2..

Και τελικά,

```
If (i < 25) Then
  Goto start
EndIf
```

Αυτό είναι το σημείο που λέει στον υπολογιστή ότι αν η τιμή της μεταβλητής **i** είναι μικρότερη από το 25 τότε να εκτελέσει τις εντολές που ξεκινούν από το σημείο **start**.

Ατέρμονη Εκτέλεση

Χρησιμοποιώντας τη δήλωση **Goto** μπορείτε να κάνετε τον υπολογιστή να επαναλαμβάνει κάτι πολλές φορές. Για παράδειγμα, μπορείτε να αλλάξετε το πρόγραμμα για τους Άρτιους και Περιττούς αριθμούς όπως παρακάτω και να το κάνετε να εκτελείτε για πάντα! Μπορείτε να σταματήσετε την εκτέλεση του προγράμματος πατώντας στο κουμπί Close (X) στην πάνω δεξιά γωνία του παραθύρου.

```
begin:
TextWindow.Write("Enter a number: ")
num = TextWindow.ReadNumber()
remainder = Math.Remainder(num, 2)
If (remainder = 0) Then
  TextWindow.WriteLine("The number is Even")
Else
  TextWindow.WriteLine("The number is Odd")
EndIf
Goto begin
```

Εικόνα 18 – Άρτιος ή Περιττός χωρίς τέλος

Κεφάλαιο 5

Βρόχοι (Loops)

Ο βρόχος For

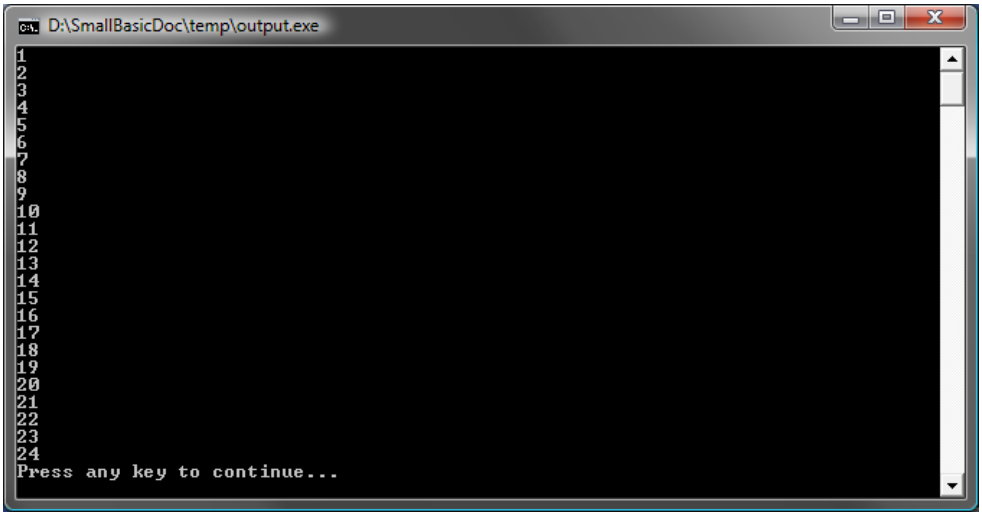
Ας δούμε το πρόγραμμα που γράψαμε στο προηγούμενο κεφάλαιο.

```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
  Goto start
EndIf
```

Το πρόγραμμα αυτό τυπώνει αριθμούς από το 1 έως το 24 με τη σειρά. Η αύξηση της τιμής μιας μεταβλητής είναι κάτι πολύ συνηθισμένο στον προγραμματισμό και για αυτό το λόγο οι γλώσσες προγραμματισμού συνήθως παρέχουν μία ευκολότερη μέθοδο για αυτή τη λειτουργία. Το παραπάνω πρόγραμμα είναι ισοδύναμο με το παρακάτω:

```
For i = 1 To 24
  TextWindow.WriteLine(i)
EndFor
```

Και το αποτέλεσμα είναι:



Εικόνα 19 – Χρησιμοποιώντας το βρόχο For

Προσέξτε ότι μειώσαμε το αρχικό πρόγραμμα των 8 γραμμών σε ένα πρόγραμμα με μόλις 4 γραμμές το οποίο συνεχίζει να κάνει ό,τι ακριβώς έκανε το πρόγραμμα των 8 γραμμών. Θυμάστε νωρίτερα που είπαμε ότι συνήθως υπάρχουν πολλοί τρόποι για να κάνουμε το ίδιο πράγμα; Ε, αυτό είναι ένα πολύ καλό παράδειγμα!

Το **For..EndFor** στον προγραμματισμό καλείται *βρόχος* (*loop*). Μας επιτρέπει να πάρουμε μια μεταβλητή, να της δώσουμε μια αρχική και μια τελική τιμή και στη συνέχεια να αφήσουμε τον υπολογιστή να αυξάνει για εμάς την τιμή της μεταβλητής. Κάθε φορά που ο υπολογιστής αυξάνει την τιμή της μεταβλητής, εκτελεί και τις δηλώσεις που υπάρχουν μεταξύ των λέξεων **For** και **EndFor**.

Αν όμως θέλατε να αυξάνεται την τιμή της μεταβλητής ανά 2 (αντί για 1) – για να τυπώνετε για παράδειγμα τους περιττούς αριθμούς μεταξύ του 1 και του 24, θα μπορούσατε επίσης να χρησιμοποιήσετε το βρόχο ως εξής:

```
For i = 1 To 24 Step 2
  TextWindow.WriteLine(i)
EndFor
```

Εικόνα 20 – Μόνο οι περιττοί αριθμοί

Το κομμάτι **Step 2** της δήλωσης **For** λέει στον υπολογιστή να αυξήσει την τιμή της μεταβλητής **i** κατά 2 αντί για το συνηθισμένο 1. Χρησιμοποιώντας την **Step** μπορείτε να ορίσετε οποιαδήποτε αύξηση εσείς θέλετε. Μπορείτε να ορίσετε ακόμα και αρνητική τιμή για τη step και να κάνετε τον υπολογιστή να μετράει αντίστροφα, όπως στο παρακάτω παράδειγμα::

```
For i = 10 To 1 Step -1
  TextWindow.WriteLine(i)
EndFor
```

Εικόνα 21 – Αντίστροφη Μέτρηση

Ο βρόχος While

Ο βρόχος While είναι άλλη μια μέθοδος επαναλήψεων που είναι ιδιαιτέρως χρήσιμη όταν δεν γνωρίζουμε εξ' αρχής τον αριθμό των επιθυμητών επαναλήψεων. Ενώ λοιπόν ο βρόχος For εκτελείτε για έναν προκαθορισμένο αριθμό επαναλήψεων, ο βρόχος While εκτελείτε έως ότου μια συγκεκριμένη συνθήκη είναι έγκυρη (αληθής). Στο παρακάτω παράδειγμα διαιρούμε έναν αριθμό στα δύο μέχρις ότου το αποτέλεσμα της διαίρεσης να είναι μεγαλύτερο του 1.

```
number = 100
While (number > 1)
  TextWindow.WriteLine(number)
  number = number / 2
EndWhile
```

Εικόνα 22 – Βρόχος Διαίρεσης στα δύο

Στο παραπάνω πρόγραμμα αναθέτουμε την τιμή 100 στη μεταβλητή *number* και εκτελούμε το βρόχο loop όσο η τιμή της μεταβλητής number είναι μεγαλύτερη του 1. Εσωτερικά στο βρόχο, τυπώνουμε την τιμή της μεταβλητής number και στη συνέχεια τη διαιρούμε δια του δύο, ουσιαστικά τη μειώνουμε στο μισό. Και όπως αναμενόταν, το αποτέλεσμα του προγράμματος είναι αριθμοί που σταδιακά μειώνονται στο μισό ο ένας μετά τον άλλον.

Θα είναι πραγματικά δύσκολο να γράψουμε το πρόγραμμα χρησιμοποιώντας το βρόχο For, επειδή δεν γνωρίζουμε πόσες φορές θα εκτελέσουμε το βρόχο. Με το βρόχο while αντιθέτως, είναι εύκολο να ελέγξουμε μια συνθήκη και να ζητήσουμε από τον υπολογιστή (με βάση τη συνθήκη) να εκτελέσει το βρόχο ή όχι.

Είναι ενδιαφέρον να προσέξουμε ότι κάθε βρόχος while μπορεί να γραφτεί χρησιμοποιώντας δηλώσεις If..Then. Για παράδειγμα, το παραπάνω πρόγραμμα μπορεί να ξαναγραφτεί όπως παρακάτω, χωρίς να αλλάξει το τελικό αποτέλεσμα.

```
number = 100
startLabel:
TextWindow.WriteLine(number)
number = number / 2

If (number > 1) Then
  Goto startLabel
EndIf
```

Στην πραγματικότητα, ο υπολογιστής εσωτερικά ξαναγράφει κάθε βρόχο While με δηλώσεις που χρησιμοποιούν If..Then μαζί με μία η παραπάνω δηλώσεις Goto.

Κεφάλαιο 6

Ξεκινώντας με τα Γραφικά

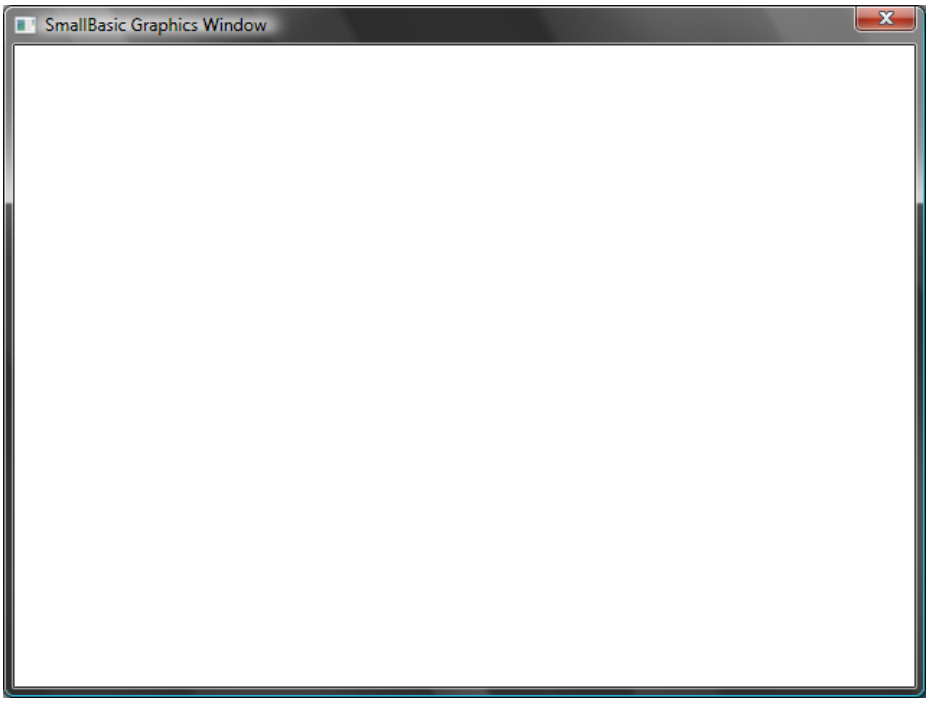
Στα παραδείγματά μας μέχρι τώρα έχουμε χρησιμοποιήσει το TextWindow για να εξοικειωθούμε με τα βασικά στοιχεία της γλώσσας Small Basic. Η Small Basic όμως έχει και μια σειρά από εξαιρετικές δυνατότητες για Γραφικά τις οποίες θα αρχίσουμε να εξερευνούμε σε αυτό το κεφάλαιο.

Εισαγωγή στο Παράθυρο Γραφικών (GraphicsWindow)

Όπως ακριβώς το TextWindow μας επέτρεπε να δουλέψουμε με κείμενο και αριθμούς, η Small Basic παρέχει και το **GraphicsWindow** που μπορούμε να χρησιμοποιήσουμε για να ζωγραφίσουμε σε αυτό. Ας αρχίσουμε με το να εμφανίσουμε το παράθυρο των γραφικών.

```
GraphicsWindow.Show()
```

Όταν εκτελέσετε αυτό το πρόγραμμα, θα προσέξετε ότι αντί του συνηθισμένου μαύρου παραθύρου κειμένου βλέπουμε ένα άσπρο παράθυρο όπως αυτό που φαίνεται παρακάτω. Δεν μπορούμε να κάνουμε και πολλά με αυτό το παράθυρο για την ώρα, αλλά θα αποτελέσει το βασικό μας παράθυρο με το οποίο θα δουλέψουμε σε αυτό το κεφάλαιο. Μπορείτε να κλείσετε το παράθυρο πατώντας στο κουμπί Close (X) στην πάνω δεξιά γωνία του παραθύρου.



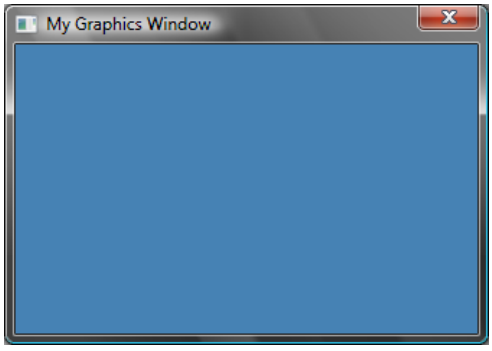
Εικόνα 23 – Ένα άδειο Παράθυρο Γραφικών

Προετοιμάζοντας το Παράθυρο Γραφικών

Το παράθυρο γραφικών σας επιτρέπει να διαμορφώσετε την εμφάνισή του όπως εσείς επιθυμείτε. Μπορείτε να αλλάξετε τον τίτλο του, το φόντο του και το μέγεθός του. Ας προχωρήσουμε λοιπόν και ας το αλλάξουμε λίγο προκειμένου εξοικειωθούμε με αυτό το παράθυρο.

```
GraphicsWindow.BackgroundColor = "SteelBlue"
GraphicsWindow.Title = "My Graphics Window"
GraphicsWindow.Width = 320
GraphicsWindow.Height = 200
GraphicsWindow.Show()
```

Εδώ λοιπόν είναι το πώς θα φαίνεται το διαμορφωμένο παράθυρό μας. Μπορείτε να αλλάξετε το φόντο του σε ένα από τα πολλά διαθέσιμα χρώματα που εμφανίζονται στο Παράρτημα Β. Πειραματιστείτε με αυτές τις ιδιότητες και δείτε πώς μπορείτε να μεταβάλετε την εμφάνιση του παραθύρου γραφικών.

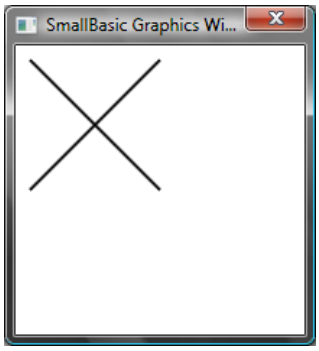


Εικόνα 24 – Ένα διαμορφωμένο Παράθυρο Γραφικών

Σχεδιάζοντας Γραμμές

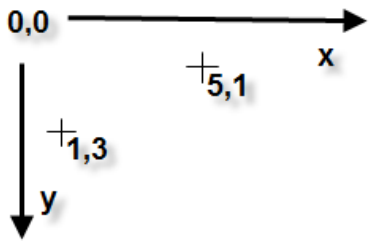
Από τη στιγμή που έχουμε εμφανίσει το Παράθυρο Γραφικών, μπορούμε να ζωγραφίσουμε σε αυτό σχήματα, κείμενο ή ακόμα και εικόνες. Ας αρχίσουμε όμως σχεδιάζοντας κάποια απλά σχήματα, όπως με το παρακάτω πρόγραμμα το οποίο σχεδιάζει δύο απλές γραμμές.

```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Εικόνα 25 – Χιαστές γραμμές

Οι πρώτες δύο εντολές του προγράμματος προετοιμάζουν το παράθυρο γραφικών και οι επόμενες δύο εντολές σχεδιάζουν τις δύο χιαστές γραμμές. Οι δύο πρώτοι αριθμοί που εμφανίζονται μετά την *DrawLine* ορίζουν τις αρχικές x και y συντεταγμένες και οι επόμενοι δύο αριθμοί ορίζουν τις τελικές x και y συντεταγμένες. Το ενδιαφέρον με τα γραφικά των υπολογιστών είναι ότι οι συντεταγμένες (0,0) αρχίζουν στο πάνω αριστερό σημείο του παραθύρου.



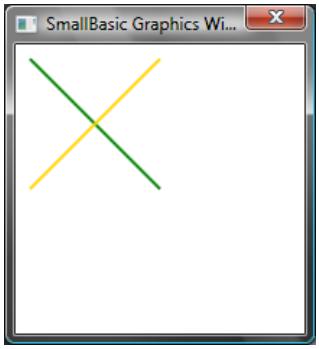
Εικόνα 26 – το Σύστημα Συντεταγμένων

Αν πάμε πίσω στο πρόγραμμα σχεδίασης γραμμών, είναι ενδιαφέρον να προσέξουμε ότι η Small Basic μας επιτρέπει να αλλάξουμε τα χαρακτηριστικά της γραμμής, όπως είναι το χρώμα

Αντί να χρησιμοποιούμε ονόματα για τα χρώματα, μπορούμε να χρησιμοποιήσουμε την ορολογία χρωμάτων του διαδυκτίου (#RRGGBB). Για παράδειγμα, το #FF0000 δηλώνει το Κόκκινο,το #FFFF00 το Κίτρινο και ούτω καθ' εξής.

και το πάχος της. Αρχικά, ας αλλάξουμε το χρώμα των γραμμών όπως φαίνεται στο παρακάτω πρόγραμμα.

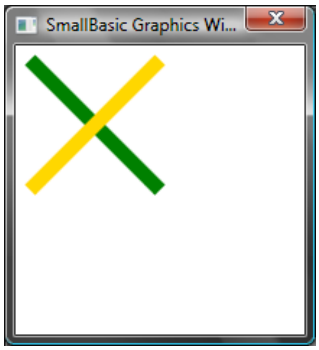
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Εικόνα 27 – Αλλάζοντας το χρώμα των γραμμών

Τώρα ας αλλάξουμε και το μέγεθός τους. Στο παρακάτω πρόγραμμα αλλάζουμε το πάχος της γραμμής στο 10 αντί της αρχικής (default) τιμής που είναι 1.

```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenWidth = 10
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```

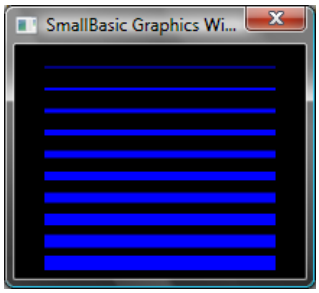


Εικόνα 28 – Χρωματιστές χοντρές γραμμές

Οι *PenWidth* και *PenColor* αλλάζουν την πένα με την οποία ζωγραφίζουμε τις γραμμές. Δεν επηρεάζουν μόνο τις γραμμές που σχεδιάζουμε αλλά και οποιοδήποτε άλλο σχήμα ζωγραφίσουμε αφού αλλάξουμε τις τιμές τους.

Χρησιμοποιώντας τους βρόχους που μάθαμε στα προηγούμενα κεφάλαιο, μπορούμε πολύ εύκολα να ζωγραφίσουμε πολλές γραμμές με αυξανόμενο πάχος.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 160
GraphicsWindow.PenColor = "Blue"

For i = 1 To 10
  GraphicsWindow.PenWidth = i
  GraphicsWindow.DrawLine(20, i * 15, 180, i * 15)
endfor
```



Εικόνα 29 – Πολλαπλά Πάχη

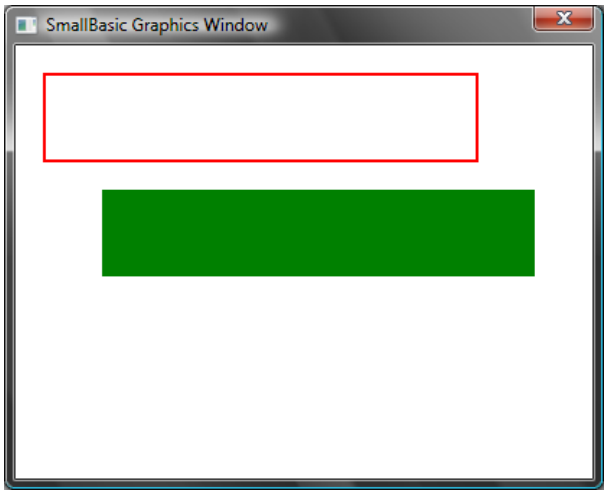
Το ενδιαφέρον κομμάτι αυτού του προγράμματος είναι ο βρόχος με τον οποίο αλλάζουμε την τιμή της *PenWidth* κάθε φορά που εκτελείτε ο βρόχος και μετά ζωγραφίζουμε και μία νέα γραμμή κάτω από την προηγούμενη.

Ζωγραφίζοντας και Χρωματίζοντας Σχήματα

Υπάρχουν συνήθως δύο ειδών λειτουργίες που μπορούμε να κάνουμε όταν ζωγραφίζουμε σχήματα. Αυτές οι λειτουργίες είναι η *Draw* (Ζωγραφίζω) και η *Fill* (Γεμίζω - Χρωματίζω). Η λειτουργία Draw ζωγραφίζει το περίγραμμα του σχεδίου χρησιμοποιώντας μια πένα (pen) ενώ η λειτουργία Fill γεμίζει το σχήμα με χρώμα (χρωματίζει) χρησιμοποιώντας την brush. Για παράδειγμα, στο παρακάτω πρόγραμμα υπάρχουν δύο παραλληλόγραμμα, το ένα έχει ζωγραφιστεί με κόκκινη πένα και το άλλο έχει γεμίσει με πράσινο χρώμα χρησιμοποιώντας πράσινη Brush.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawRectangle(20, 20, 300, 60)
```

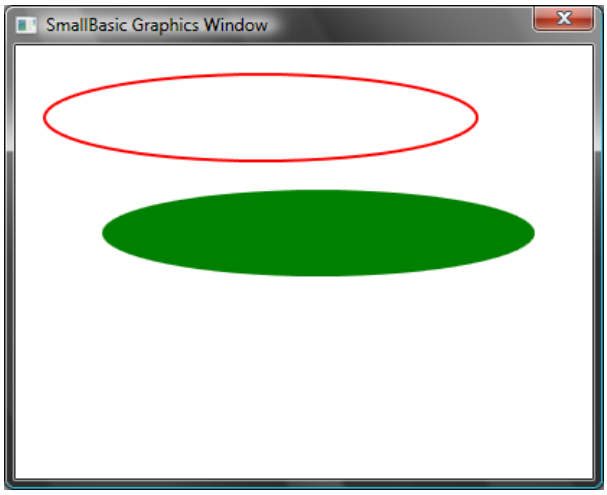
```
GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillRectangle(60, 100, 300, 60)
```



Εικόνα 30 Ζωγραφίζοντας και Χρωματίζοντας

Για να ζωγραφίσετε η να χρωματίσετε ένα παραλληλόγραμμο θα χρειαστείτε τέσσερις αριθμούς. Οι πρώτοι δύο αριθμοί αντιπροσωπεύουν τις X και Y συντεταγμένες της πάνω αριστερής γωνίας του παραλληλογράμμου. Ο τρίτος αριθμός αντιπροσωπεύει το πλάτος του παραλληλογράμμου ενώ ο τέταρτος ορίζει το ύψος του. Στην πραγματικότητα, το ίδιο ισχύει και για να ζωγραφίσετε ή να χρωματίσετε ελλείψεις, όπως φαίνεται και στο παρακάτω πρόγραμμα.

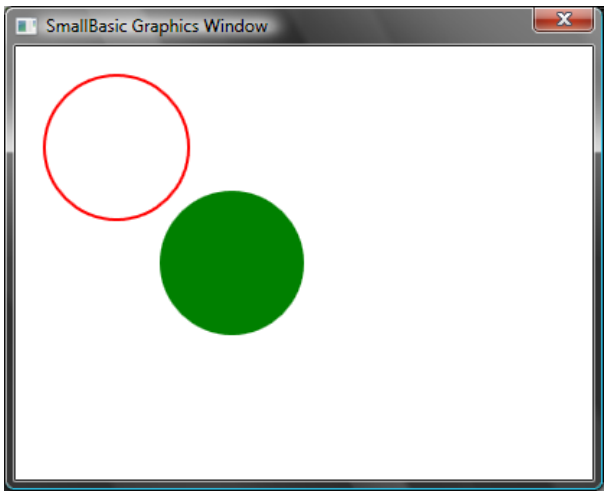
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawEllipse(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillEllipse(60, 100, 300, 60)
```



Εικόνα 31 – Ζωγραφίζοντας και Χρωματίζοντας Ελλείψεις

Οι ελλείψεις δεν είναι τίποτε άλλο παρά γενικευμένες περιπτώσεις κύκλων. Αν θέλετε λοιπόν να ζωγραφίσετε κύκλους, το μόνο που έχετε να κάνετε είναι να ορίσετε ίδια τιμή για πλάτος και ύψος.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawEllipse(20, 20, 100, 100)

GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillEllipse(100, 100, 100, 100)
```



Εικόνα 32 – Κύκλοι

Κεφάλαιο 7

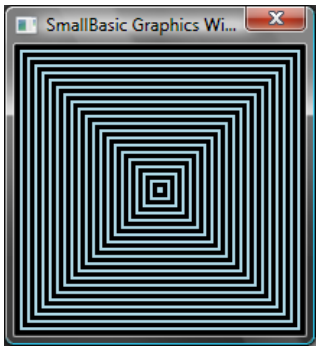
Διασκεδάζοντας με τα Σχήματα

Στο κεφάλαιο αυτό θα διασκεδάσουμε χρησιμοποιώντας όλα όσα έχουμε μάθει μέχρι τώρα. Το κεφάλαιο αυτό περιλαμβάνει κάποια παραδείγματα που παρουσιάζουν κάποιους ενδιαφέροντες τρόπους συνδυασμού όλων όσων έχετε μάθει μέχρι τώρα προκειμένου να φτιάξουμε κάποια όμορφα προγράμματα.

Διαδοχικά Τετράγωνα

Εδώ ζωγραφίζουμε τετράγωνα σε ένα βρόχο, με αυξανόμενο μέγεθος (το ένα μέσα στο άλλο).

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightBlue"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200

For i = 1 To 100 Step 5
  GraphicsWindow.DrawRectangle(100 - i, 100 - i, i * 2, i * 2)
EndFor
```

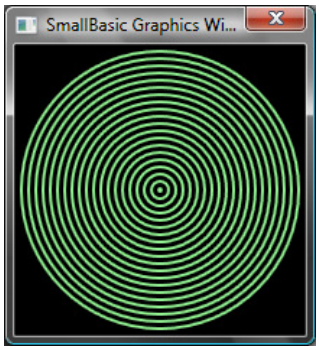


Εικόνα 33 – Διαδοχικά Τετράγωνα

Διαδοχικοί Κύκλοι

Μια παραλλαγή του προηγούμενου προγράμματος που ζωγραφίζει κύκλους αντί για τετράγωνα.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200

For i = 1 To 100 Step 5
  GraphicsWindow.DrawEllipse(100 - i, 100 - i, i * 2, i * 2)
EndFor
```

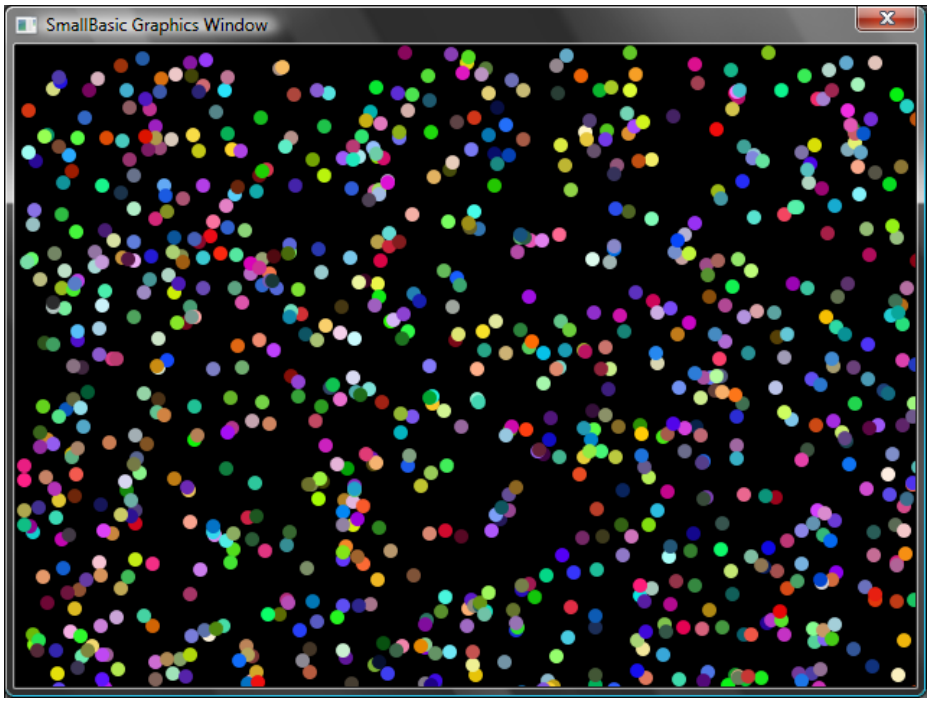


Εικόνα 34 – Διαδοχικοί Κύκλοι

Τυχαιότητα

Το πρόγραμμα αυτό χρησιμοποιεί τη λειτουργία GraphicsWindow.*GetRandomColor* για να θέσει τυχαία χρώματα για την brush και στη συνέχεια χρησιμοποιεί την Math.*GetRandomNumber* για να ορίσει τις x και y συντεταγμένες των κύκλων. Οι δύο αυτές λειτουργίες μπορούν να συνδυαστούν με ενδιαφέροντες τρόπους για να δημιουργήσουν ενδιαφέροντα προγράμματα που θα δίνουν διαφορετικά αποτελέσματα κάθε φορά που τα εκτελούμε.

```
GraphicsWindow.BackgroundColor = "Black"
For i = 1 To 1000
  GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()
  x = Math.GetRandomNumber(640)
  y = Math.GetRandomNumber(480)
  GraphicsWindow.FillEllipse(x, y, 10, 10)
EndFor
```



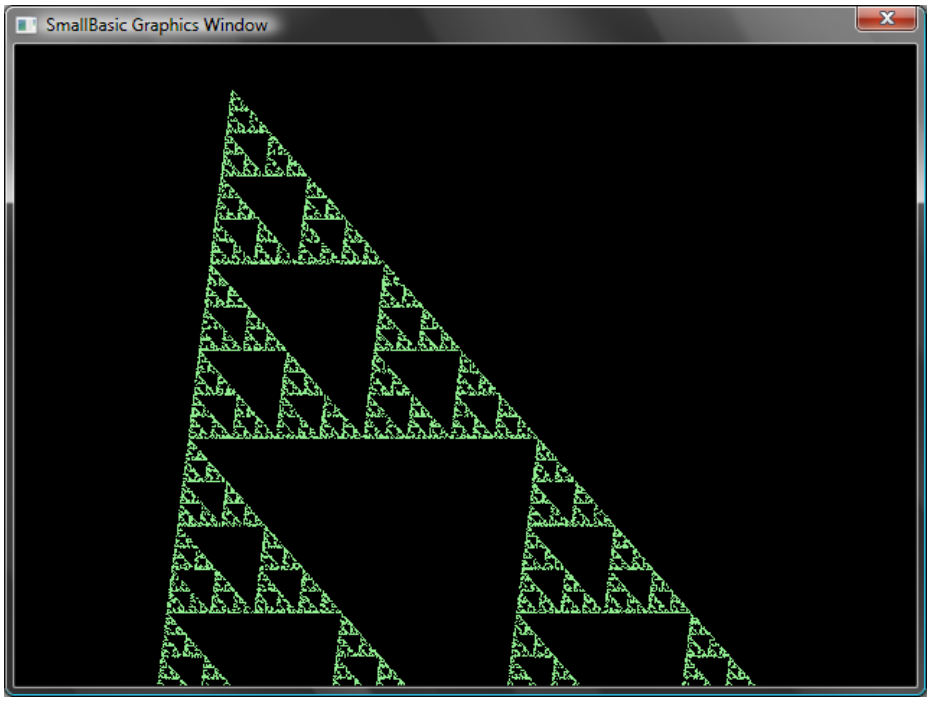
Εικόνα 35 – Τυχαιότητα

Fractals

Το παρακάτω πρόγραμμα ζωγραφίζει ένα απλό fractal τριγώνων χρησιμοποιώντας τυχαίους αριθμούς. Το fractal είναι ένα γεωμετρικό σχήμα που υποδιαιρείται σε κομμάτια, καθένα από τα οποία βασίζεται στο ίδιο ακριβώς αρχικό σχήμα. Στην περίπτωση αυτή, το πρόγραμμα ζωγραφίζει εκατοντάδες τρίγωνα καθένα από τα οποία έχει το ίδιο σχήμα με το αρχικό. Από τη στιγμή ειδικά που το πρόγραμμα εκτελείται για κάποια δευτερόλεπτα, θα μπορείτε να δείτε τα τρίγωνα να σχηματίζονται σιγά-σιγά από πολλές τελείες. Η λογική του προγράμματος είναι δύσκολο να περιγραφεί και αφήνεται ως άσκηση για εσάς για να την εξερευνήσετε.

```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100

For i = 1 To 100000
  r = Math.GetRandomNumber(3)
  ux = 150
  uy = 30
  If (r = 1) then
    ux = 30
    uy = 1000
  EndIf

  If (r = 2) Then
    ux = 1000
    uy = 1000
```

```
  EndIf

  x = (x + ux) / 2
  y = (y + uy) / 2

  GraphicsWindow.SetPixel(x, y, "LightGreen")
EndFor
```



Εικόνα 36 – Τριγωνικό Fractal

Αν θέλετε να δείτε σε αργή κίνηση τη δημιουργία του fractal μπορείτε να εισάγεται μια καθυστέρησης στο βρόχο, χρησιμοποιώντας τη λειτουργία **Program.***Delay*. Η λειτουργία αυτή λαμβάνει έναν αριθμό που ορίζει (σε χιλιοστά του δευτερολέπτου) το πόσο θα πρέπει η εκτέλεση του προγράμματος να καθυστερήσει. Παρακάτω το διαμορφωμένο πρόγραμμα στο οποίο η εντολή που εισάγει την καθυστέρηση έχει γραφεί με έντονα γράμματα.

```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100

For i = 1 To 100000
  r = Math.GetRandomNumber(3)
  ux = 150
  uy = 30
  If (r = 1) then
    ux = 30
    uy = 1000
  EndIf
```

```
  If (r = 2) Then
    ux = 1000
    uy = 1000
  EndIf

  x = (x + ux) / 2
  y = (y + uy) / 2

  GraphicsWindow.SetPixel(x, y, "LightGreen")
  Program.Delay(2)
EndFor
```

Αυξάνοντας την καθυστέρηση θα κάνουμε την εκτέλεση του προγράμματος πιο αργή. Πειραματιστείτε με τους αριθμούς για να καταλήξετε σε αυτό που σας αρέσει.

Μια άλλη παραλλαγή που μπορείτε να κάνετε στο πρόγραμμα αυτό είναι να αντικαταστήσετε την παρακάτω γραμμή:

```
GraphicsWindow.SetPixel(x, y, "LightGreen")
```

με αυτή:

```
color = GraphicsWindow.GetRandomColor()
GraphicsWindow.SetPixel(x, y, color)
```

Η αλλαγή αυτή θα κάνει το πρόγραμμα να ζωγραφίζει τα τρίγωνα χρησιμοποιώντας τυχαία χρώματα.

Κεφάλαιο 8

Γραφικά Χελώνας

Logo

Στη δεκαετία του 1970, υπήρχε μια πολύ απλή αλλά ισχυρή γλώσσα προγραμματισμού, η ονομαζόμενη Logo, που την χρησιμοποιούσαν μερικοί ερευνητές. Αυτό ίσχυες έως ότου κάποιος πρόσθεσε στη γλώσσα αυτό που ονομάζουμε «γραφικά χελώνας» (“Turtle Graphics”), με το οποίο προστέθηκε μια «χελώνα» στην οθόνη του υπολογιστή και ανταποκρινόταν σε εντολές όπως *Μετακίνηση Μπροστά*, *Στρίψε Δεξιά*, *Στρίψε Αριστερά*, (*Move Forward, Turn Right, Turn Left*) κλπ. Χρησιμοποιώντας τη χελώνα, οι άνθρωποι ήταν σε θέση να ζωγραφίσουν ενδιαφέροντα σχήματα στην οθόνη. Αυτή η δυνατότητα κατέστησε τη γλώσσα άμεσα προσιτή και ελκυστική για τους ανθρώπους όλων των ηλικιών και ήταν ο βασικός υπεύθυνος για την ευρεία δημοτικότητά της τη δεκαετία του 1980.

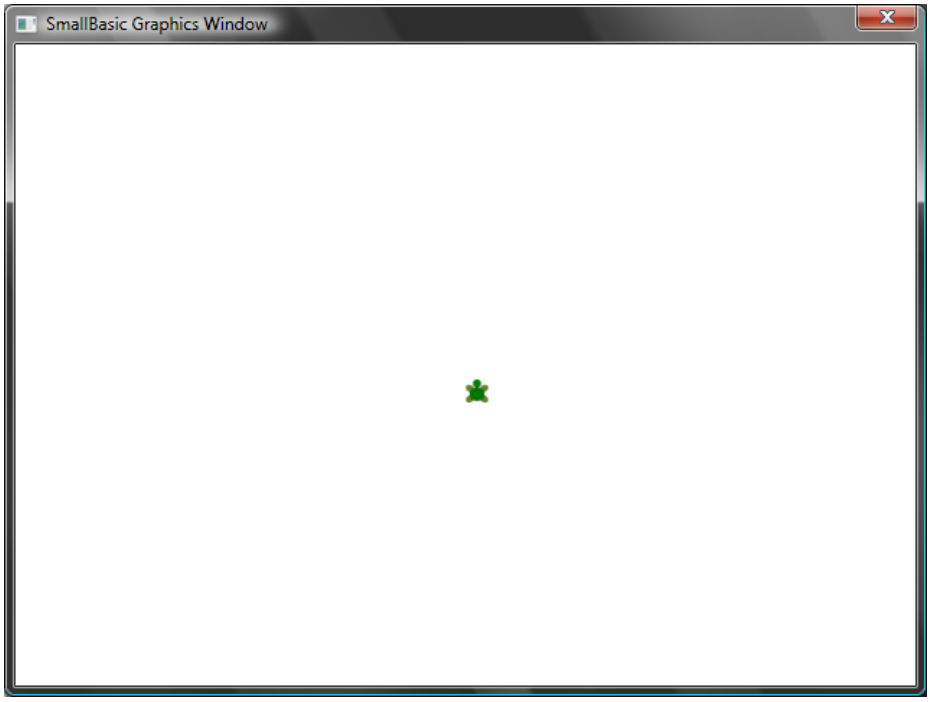
Η Small Basic παρέχει ένα αντικείμενο Χελώνα με πολλές εντολές που μπορούν να χρησιμοποιηθούν στα προγράμματα της Small Basic. Σε αυτό το κεφάλαιο, θα χρησιμοποιήσουμε τη Χελώνα για να σχεδιάσουμε γραφικά στην οθόνη μας.

Η Χελώνα

Για να ξεκινήσουμε, πρέπει να κάνουμε τη Χελώνα ορατή στην οθόνη μας. Αυτό επιτυγχάνεται με την παρακάτω απλή γραμμή.

```
Turtle.Show()
```

Όταν εκτελέσετε αυτό το πρόγραμμα θα παρατηρήσετε ένα άσπρο παράθυρο, όπως ακριβώς αυτό που είδαμε στο προηγούμενο κεφάλαιο, με τη διαφορά ότι το τωρινό έχει μια Χελώνα στο κέντρο του. Αυτή η Χελώνα θα ακολουθεί τις οδηγίες μας και θα σχεδιάζει ο,τιδήποτε της ζητήσουμε.



Εικόνα 37 – Η Χελώνα στην οθόνη μας

Μετακίνηση και Σχεδίαση

Μία από τις οδηγίες που καταλαβαίνει η Χελώνα είναι η **Move** (Μετακίνηση). Η λειτουργία αυτή δέχεται έναν αριθμό σαν στοιχείο εισόδου και αυτό ο αριθμός λέει στη Χελώνα πόσο μακριά να πάει. Στο παρακάτω παράδειγμα θα ζητήσουμε από τη Χελώνα να μετακινηθεί κατά 100 pixels.

```
Turtle.Move(100)
```

Όταν εκτελέσετε αυτό το πρόγραμμα, θα δείτε τη Χελώνα να μετακινείται αργά κατά 100 pixels προς τα πάνω. Καθώς μετακινείται, θα παρατηρήσετε ότι ζωγραφίζει μια γραμμή πίσω της. Όταν η Χελώνα σταματήσει να κινείται, το αποτέλεσμα θα είναι κάτι σαν την παρακάτω εικόνα.

Όταν ζητούμε από τη Χελώνα να εκτελέσει μια λειτουργία δεν είναι ανάγκη να καλούμε την Show(). Η Χελώνα θα εμφανίζει αυτόματα το αποτέλεσμα όποιας λειτουργίας της ζητούμε να εκτελέσει.

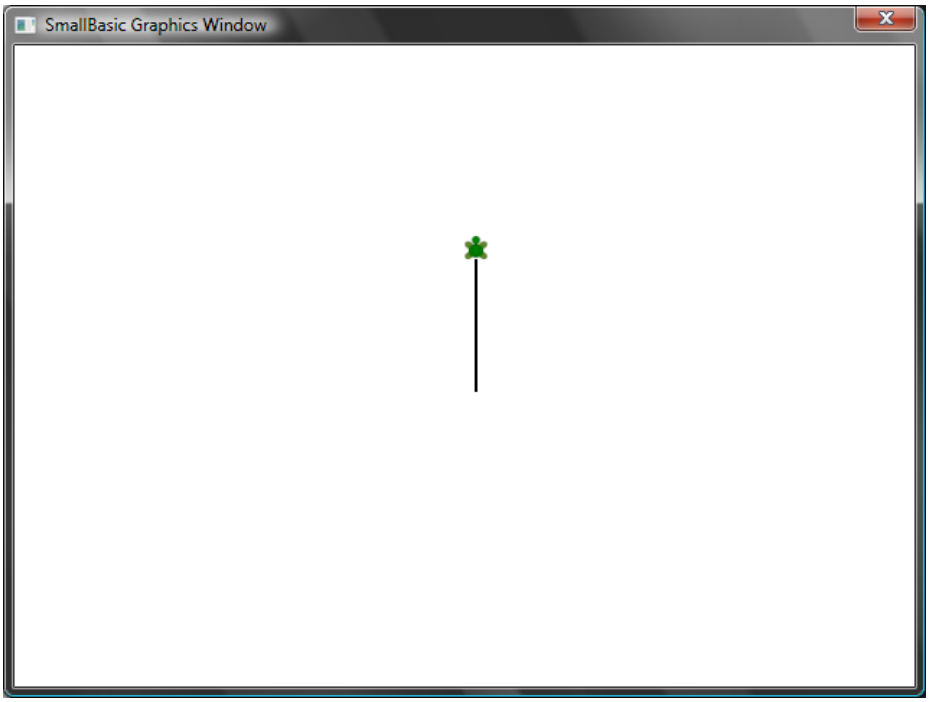


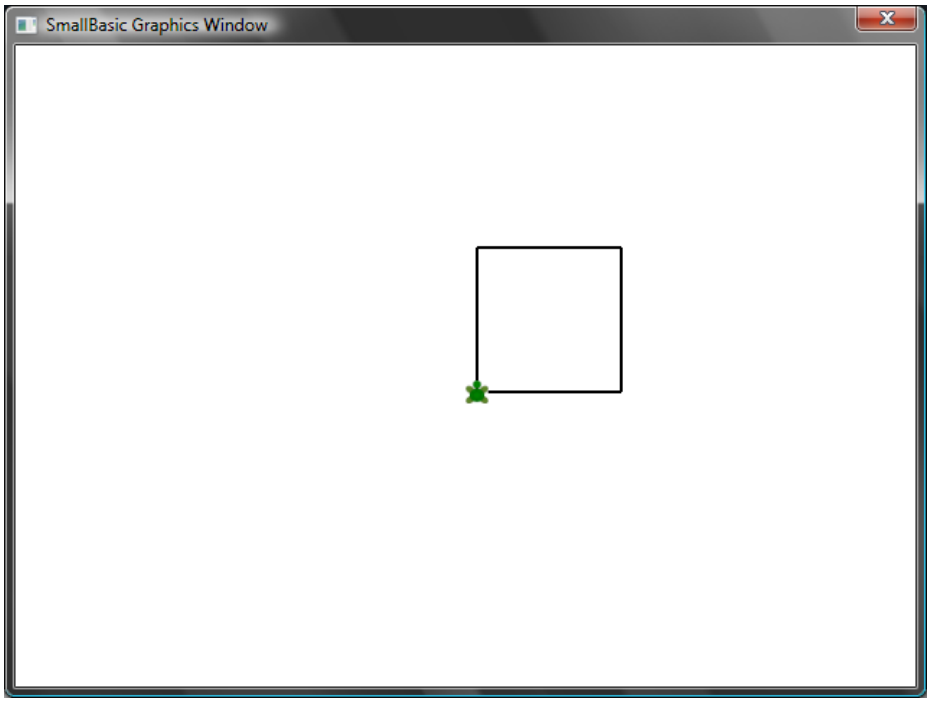
Figure 38 - Move a hundred pixels

Ζωγραφίζοντας ένα Τετράγωνο

Ένα τετράγωνο έχει τέσσερις πλευρές, δύο κάθετες και δύο οριζόντιες. Προκειμένου λοιπόν να σχεδιάσουμε ένα τετράγωνο θα πρέπει να ζητήσουμε από τη Χελώνα να σχεδιάσει μια γραμμή, να στρίψει δεξιά (**TurnRight**) και να ζωγραφίσει άλλη μια γραμμή, και να συνεχίσουμε μέχρι να σχεδιάσουμε και τις τέσσερις πλευρές. Αν τα προηγούμενα τα γράψουμε σε πρόγραμμα, θα είναι κάπως έτσι:

```
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
```

Όταν εκτελέσετε αυτό το πρόγραμμα, θα δείτε τη Χελώνα να ζωγραφίζει ένα τετράγωνο σχεδιάζοντας μία πλευρά κάθε φορά, και το αποτέλεσμα θα μοιάζει με την παρακάτω εικόνα.



Εικόνα 39 – Η Χελώνα σχεδιάζει ένα τετράγωνο

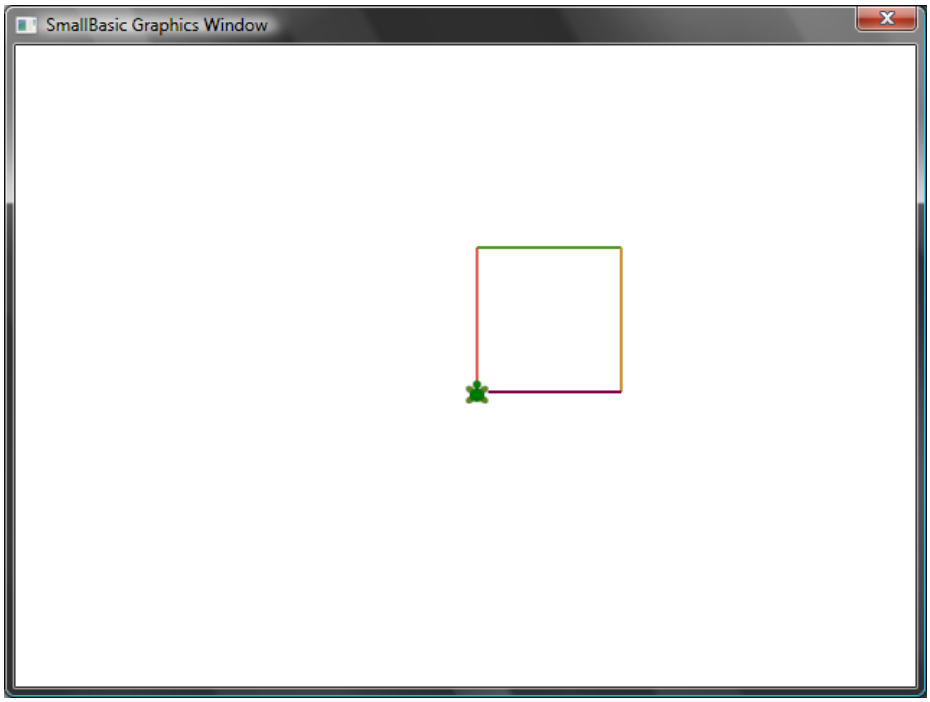
Αξίζει να παρατηρήσουμε (αν δεν το έχετε ήδη προσέξει) ότι εκτελούμε τις ίδιες δύο εντολές ξανά και ξανά – για την ακρίβεια, τέσσερις συνεχόμενες φορές. Και έχουμε ήδη μάθει ότι τέτοιες επαναλαμβανόμενες εντολές μπορούν να εκτελεστούν χρησιμοποιώντας βρόγχους. Αν αλλάξουμε λοιπόν το παραπάνω πρόγραμμα και χρησιμοποιήσουμε βρόχο **For..EndFor**, θα καταλήξουμε σε ένα πολύ πιο απλό πρόγραμμα.

```
 For i = 1 To 4
  Turtle.Move(100)
  Turtle.TurnRight()
EndFor
```

Αλλάζοντας Χρώματα

Η Χελώνα σχεδιάζει στο GraphicsWindow που γνωρίσαμε στο προηγούμενο κεφάλαιο. Αυτό σημαίνει ότι όλες οι λειτουργίες που μάθαμε στο προηγούμενο κεφάλαιο μπορούν να χρησιμοποιηθούν και εδώ. Για παράδειγμα, το παρακάτω πρόγραμμα θα ζωγραφίσει ένα τετράγωνο στο οποίο η κάθε πλευρά θα έχει διαφορετικό (τυχαίο) χρώμα.

```
For i = 1 To 4
  GraphicsWindow.PenColor = GraphicsWindow.GetRandomColor()
  Turtle.Move(100)
  Turtle.TurnRight()
EndFor
```



Εικόνα 40 – Αλλάζοντας χρώματα

Σχεδιάζοντας πιο περίπλοκα σχήματα

Η Χελώνα, εκτός από τις λειτουργίες TurnRight (Στρίψε Δεξιά) και TurnLetf (Στρίψε Αριστερά), έχει και τη λειτουργία Turn (Στρίψε). Η λειτουργία αυτή δέχεται σαν στοιχείο εισόδου έναν αριθμό που αντιπροσωπεύει τη γωνία περιστροφής. Χρησιμοποιώντας αυτή τη λειτουργία είμαστε σε θέση να ζωγραφίσουμε πολύγωνα. Το παρακάτω πρόγραμμα σχεδιάζει ένα εξάγωνο (πολύγωνο με έξι πλευρές).

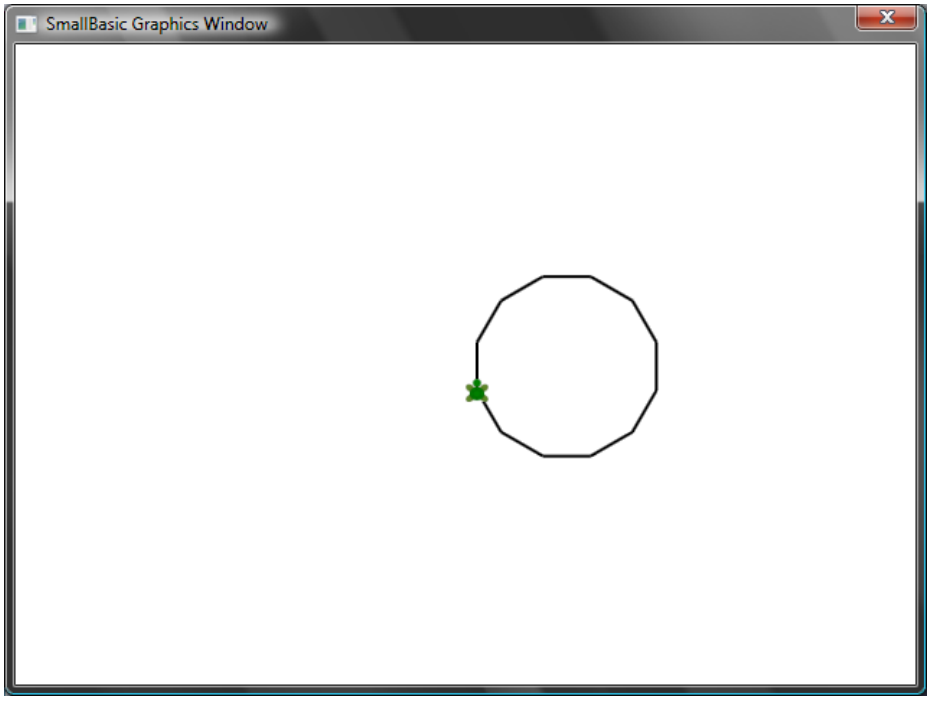
```
For i = 1 To 6
  Turtle.Move(100)
  Turtle.Turn(60)
EndFor
```

Δοκιμάστε αυτό το πρόγραμμα και δείτε αν πραγματικά σχεδιάζει ένα εξάγωνο. Παρατηρείστε ότι καθώς η γωνία μεταξύ των πλευρών είναι 60 μοίρες, χρησιμοποιούμε την εντολή **Turn(60)**. Για ένα τέτοιο πολύγωνο, στο οποίο όλες του οι πλευρές είναι ίσες, η γωνία μεταξύ των πλευρών μπορεί να υπολογιστεί εύκολα με το να διαιρέσουμε το 360 με τον αριθμό των πλευρών. Γνωρίζοντας αυτή την πληροφορία λοιπόν και χρησιμοποιώντας μεταβλητές, μπορούμε να γράψουμε ένα αρκετά γενικό πρόγραμμα που μπορεί να σχεδιάσει το πολύγωνα με όσες πλευρές θέλουμε.

```
sides = 12

length = 400 / sides
angle = 360 / sides
```

```
For i = 1 To sides
  Turtle.Move(length)
  Turtle.Turn(angle)
EndFor
```

Με αυτό το πρόγραμμα μπορείτε να σχεδιάσετε όποιο πολύγωνο θέλετε, αλλάζοντας απλά τη μεταβλητή **sides** (πλευρές). Δίνοντας τιμή 4 θα μας σχεδιάσει το τετράγωνο με το οποίο είχαμε ξεκινήσει ενώ δίνοντας μια αρκετά μεγάλη τιμή, ας πούμε 50, θα σχεδιάσει ένα πολύγωνο που δεν θα διαφέρει και πολύ από έναν κύκλο.



Εικόνα 41 – Σχεδιάζοντας ένα 12-πλευρό πολύγωνο

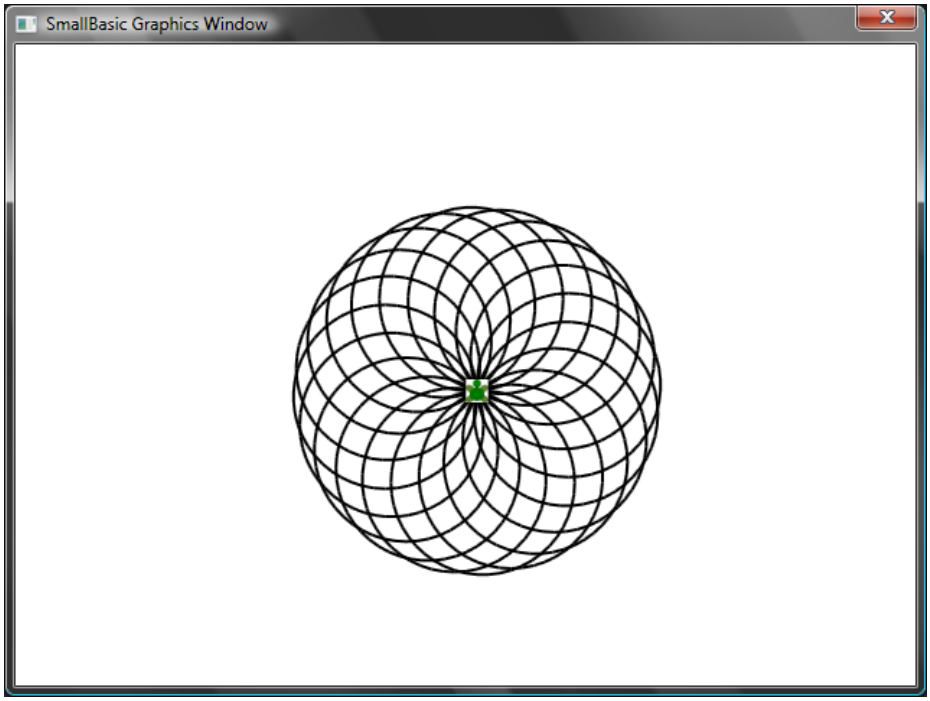
Χρησιμοποιώντας την τεχνική που μόλις μάθαμε, μπορούμε να κάνουμε τη Χελώνα να σχεδιάσει πολλαπλούς κύκλους κάθε έναν μετατοπισμένο κατά λιγάκι, δίνοντάς μας ένα πολύ ενδιαφέρον σχέδιο.

```
sides = 50
length = 400 / sides
angle = 360 / sides

Turtle.Speed = 9

For j = 1 To 20
  For i = 1 To sides
    Turtle.Move(length)
    Turtle.Turn(angle)
  EndFor
  Turtle.Turn(18)
```

```
EndFor
```

Το παραπάνω πρόγραμμα έχει δύο βρόγχους **For..EndFor**, τον έναν μέσα στον άλλον. Ο εσωτερικός βρόγχος (*i = 1 to sides*) είναι παρόμοιος με το πρόγραμμα για τα πολύγωνα και είναι αυτός που σχεδιάζει έναν κύκλο. Ο εξωτερικός βρόγχος (*j = 1 to 20*) είναι αυτός που μετατοπίζει κατά λίγο τη Χελώνα κάθε φορά που σχεδιάζεται ένας κύκλος, και με αυτό τον τρόπο σχεδιάζουμε συνολικά 20 κύκλους. Με τους δύο βρόγχους μαζί, το πρόγραμμα ζωγραφίζει μία πολύ ενδιαφέρουσα εικόνα, όπως αυτή παρακάτω.

Στο παραπάνω πρόγραμμα κάναμε τη Χελώνα να σχεδιάζει γρηγορότερα, θέτωντας στην ιδιότητα Speed (Ταχύτητα) την τιμή 9. Μπορείτε να θέσετε οποιαδήποτε τιμή από το 1 έως το 10 στην ιδιότητα αυτή για να κάνετε τη Χελώνα να πηγαίνει όσο γρήγορα θέλετε.



Εικόνα 42 – Κάνοντας κύκλους

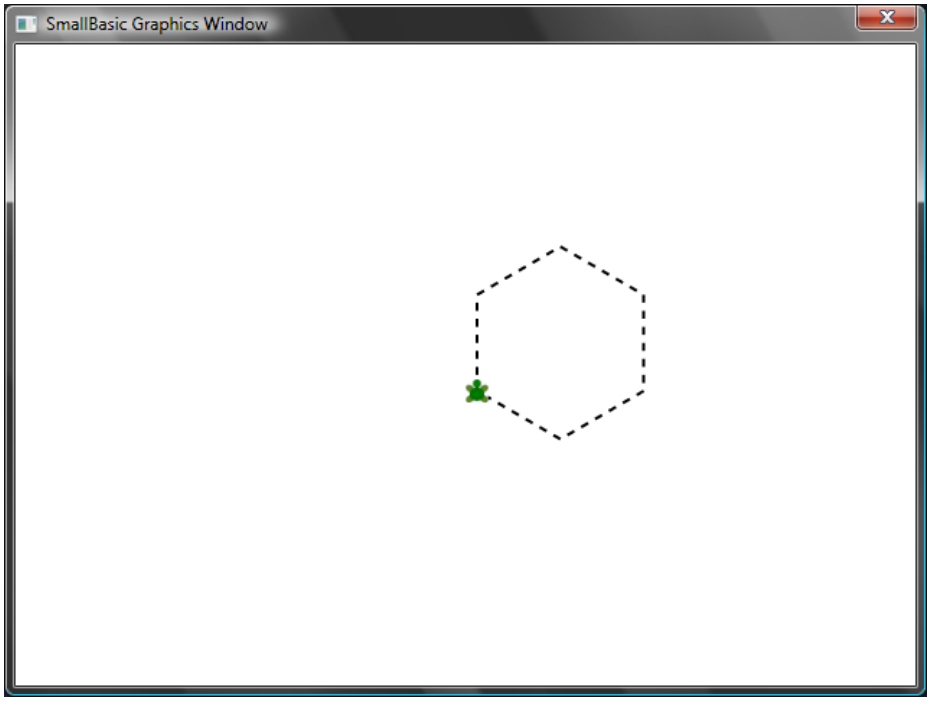
Κάνοντας Περίπατο

Μπορείτε να κάνετε τη Χελώνα να μη σχεδιάζει γραμμές, με τη λειτουργία **PenUp**. Αυτή η λειτουργία σας επιτρέπει να μεταφέρεται τη χελώνα οπουδήποτε στην οθόνη χωρίς ταυτόχρονα να ζωγραφίζει γραμμές πίσω της. Με τη λειτουργία **PenDown** επιτρέπουμε στη Χελώνα να σχεδιάζει γραμμές. Αυτές οι λειτουργίες μπορούν να χρησιμοποιηθούν για να σχεδιάσουμε ενδιαφέροντα εφέ, όπως ας πούμε διακεκομμένες γραμμές. Το παρακάτω πρόγραμμα χρησιμοποιεί αυτές τις λειτουργίες για να σχεδιάσει ένα διακεκομμένο πολύγωνο.

```
sides = 6

length = 400 / sides
```

```
angle = 360 / sides

For i = 1 To sides
  For j = 1 To 6
    Turtle.Move(length / 12)
    Turtle.PenUp()
    Turtle.Move(length / 12)
    Turtle.PenDown()
  EndFor
  Turtle.Turn(angle)
EndFor
```

Και πάλι το πρόγραμμα έχει δύο βρόγχους. Ο εσωτερικός βρόγχος σχεδιάζει μια απλή διακεκομμένη γραμμή, ενώ ο εξωτερικός καθορίζει τον αριθμό των γραμμών που θα σχεδιάσουμε (ουσιαστικά τις πλευρές του πολυγώνου). Στο παράδειγμά μας, θέσαμε την τιμή 6 στη μεταβλητή **sides** (πλευρές) οπότε σχεδιάσαμε ένα διακεκομμένο εξάγωνο, όπως παρακάτω.



Εικόνα 43 – Χρησιμοποιώντας τις PenUp και PenDown

Κεφάλαιο 9

Υπορουτίνες

Αρκετά συχνά ενώ γράφουμε προγράμματα έχουμε να αντιμετωπίσουμε περιπτώσεις κατά τις οποίες πρέπει να εκτελέσουμε το ίδιο σύνολο βημάτων ξανά και ξανά . Στις περιπτώσεις αυτές, προφανώς δεν έχει νόημα να επαναλαμβάνουμε τις ίδιες εντολές πολλές φορές. Εδώ είναι που οι *Υπορουτίνες* (*Subroutines*) αποδεικνύονται πολύ βολικές.

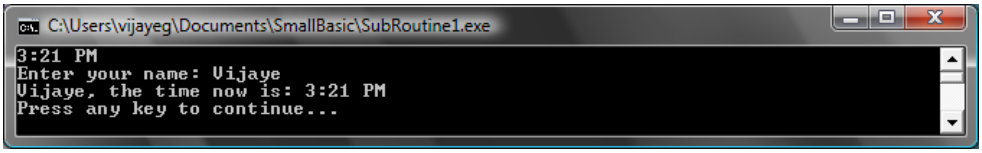
Η υπορουτίνα είναι ένα κομμάτι κώδικα, μέρος μεγάλου προγράμματος, που συνήθως κάνει κάτι πολύ συγκεκριμένο και το οποίο μπορεί να κληθεί από οπουδήποτε μέσα στο πρόγραμμα. Οι υπορουτίνες προσδιορίζονται (δηλώνονται) με ένα όνομα που ακολουθεί τη λέξη-κλειδί **Sub** και ο κώδικας της υπορουτίνας τελειώνει με τη λέξη-κλειδί **EndSub**. Για παράδειγμα, το παρακάτω απόσπασμα αντιπροσωπεύει μια υπορουτίνα το όνομα της οποίας είναι *PrintTime* και αυτό που κάνει είναι να εμφανίζει την τρέχουσα ώρα στο TextWindow.

```
Sub PrintTime
  TextWindow.WriteLine(Clock.Time)
EndSub
```

Παρακάτω εμφανίζεται το πρόγραμμα που περιλαμβάνει την υπορουτίνα και την καλεί από διάφορα σημεία.

```
PrintTime()
TextWindow.Write("Enter your name: ")
name = TextWindow.Read()
TextWindow.Write(name + ", the time now is: ")
PrintTime()

Sub PrintTime
  TextWindow.WriteLine(Clock.Time)
```

```
EndSub
```



Εικόνα 44 – Καλώντας μια απλή υπορουτίνα

Η εκτέλεση μιας υπορουτίνας γίνεται καλώντας τη με το όνομά της, *SubroutineName()*. Ως συνήθως, οι παρενθέσεις είναι απαραίτητες για να υποδείξετε στον υπολογιστή ότι θέλετε να εκτελέσετε μια υπορουτίνα.

Πλεονεκτήματα χρήσης Υπορουτινών

Όπως μόλις είδαμε παραπάνω, οι υπορουτίνες βοηθούν στη μείωση του κώδικα που πρέπει να πληκτρολογήσουμε. Από τη στιγμή που έχετε γράψει την υπορουτίνα *PrintTime*, μπορείτε να την καλέσετε από όποιο σημείο του προγράμματός σας θέλετε και αυτή θα εμφανίσει την τρέχουσα ώρα.

Να θυμάστε ότι στη SmallBasic μπορείτε να καλέσετε μια υπορουτίνα μόνο μέσα στο ίδιο πρόγραμμα που την έχετε γράψει. Δεν μπορείτε να καλέσετε μια υπορουτίνα από πρόγραμμα διαφορετικό από συτό στην οποία τη γράψατε.

Επιπλέον, οι υπορουτίνες μπορούν να βοηθήσουν στην ανάλυση σύνθετων προβλημάτων σε μικρότερα απλούστερα κομμάτια. Αν για παράδειγμα έχετε να επιλύσετε μια σύνθετη εξίσωση, μπορείτε να γράψετε πολλές απλές υπορουτίνες που η κάθε μία θα επιλύει ένα μέρος της μεγάλης σύνθετης εξίσωσης. Στη συνέχεια, συνδέετε τα επιμέρους αποτελέσματα προκειμένου να πάρετε τη λύση στην αρχική δύσκολη εξίσωση.

Οι υπορουτίνες μπορούν ακόμα να συνδράμουν στην βελτίωση της κατανόησης ενός προγράμματος. Με άλλα λόγια, αν έχετε γράψει υπορουτίνες με ονομασίες που υποδεικνύουν το τί κάνουν, τα προγράμματά σας γίνονται εύκολα αναγνώσιμα και κατανοητά. Αυτό είναι πολύ σημαντικό αν θέλετε να κατανοήσετε το πρόγραμμα κάποιου άλλου ή αν θέλετε τα προγράμματά σας να γίνονται εύκολα κατανοητά από άλλους. Μερικές φορές είναι ακόμα και για εσάς ευκολότερο να κατανοήσετε το πρόγραμμά σας όταν το δείτε, ας πούμε, μια εβδομάδα αφότου το έχετε γράψει.

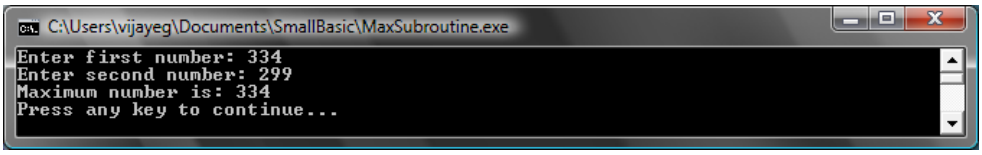
Χρησιμοποιώντας μεταβλητές

Μπορείτε να έχετε πρόσβαση και να χρησιμοποιήσετε τις μεταβλητές που έχετε δηλώσει σε ένα πρόγραμμα μέσα από μια υπορουτίνα. Για παράδειγμα, το παρακάτω πρόγραμμα δέχεται δυο αριθμούς και εμφανίζει τον μεγαλύτερο εξ αυτών. Παρατηρείστε ότι η μεταβλητή *max* χρησιμοποιείται τόσο μέσα όσο και έξω από την υπορουτίνα.

```
TextWindow.Write("Enter first number: ")
num1 = TextWindow.ReadNumber()
```

```
TextWindow.Write("Enter second number: ")
num2 = TextWindow.ReadNumber()

FindMax()
TextWindow.WriteLine("Maximum number is: " + max)

Sub FindMax
  If (num1 > num2) Then
    max = num1
  Else
    max = num2
  EndIf
EndSub
```

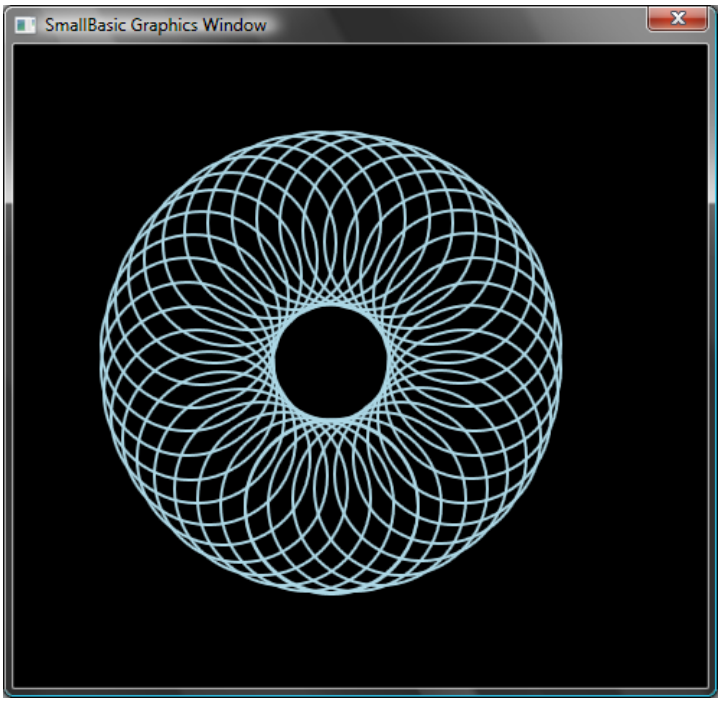
Και το αποτέλεσμα του προγράμματος είναι σαν το παρακάτω:



Εικόνα 45 – Εμφάνιση μέγιστου δύο αριθμών με χρήση υπορουτίνας

Ας δούμε ένα άλλο παράδειγμα που επεξηγεί τη χρήση των υπορουτινών. Αυτή τη φορά θα χρησιμοποιήσουμε ένα πρόγραμμα γραφικών που θα υπολογίζει διάφορα σημεία και τα οποία θα τα αποθηκεύει στις μεταβλητές *x* και *y*. Στη συνέχεια, θα καλεί την υπορουτίνα **DrawCircleUsingCenter** η οποία σχεδιάζει έναν κύκλο χρησιμοποιώντας τις τιμές των μεταβλητών x και y σαν συντεταγμένες για το κέντρο του κύκλου.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightBlue"
GraphicsWindow.Width = 480
For i = 0 To 6.4 Step 0.17
  x = Math.Sin(i) * 100 + 200
  y = Math.Cos(i) * 100 + 200

  DrawCircleUsingCenter()
EndFor

Sub DrawCircleUsingCenter
  startX = x - 40
  startY = y - 40

  GraphicsWindow.DrawEllipse(startX, startY, 120, 120)
EndSub
```



Εικόνα 46 – Παράδειγμα γραφικών με χρήση υπορουτινών

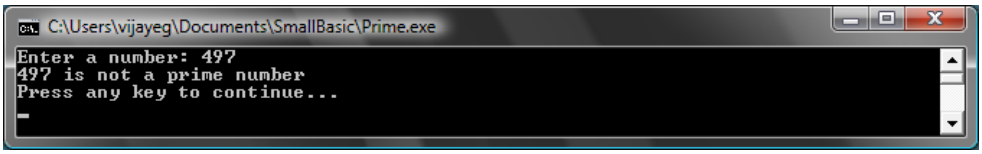
Καλώντας Υπορουτίνες μέσα σε Βρόγχους

Κάποιες φορές οι υπορουτίνες καλούνται μέσα σε βρόγχους κατά τη διάρκεια του οποίου εκτελούν τις ίδιες εντολές αλλά με διαφορετικές κάθε φορά τιμές σε μία ή περισσότερες μεταβλητές. Για παράδειγμα, ας πούμε ότι έχετε μια υπορουτίνα που ονομάζεται PrimeCheck και η υπορουτίνα αυτή ελέγχει αν ένας αριθμός είναι πρώτος (prime) η όχι. Μπορείτε να γράψετε ένα πρόγραμμα το οποίο να δέχεται μια τιμή και εν συνεχεία να μας ενημερώνει αν αυτή η τιμή είναι πρώτος αριθμός ή όχι, χρησιμοποιώντας την υπορουτίνα. (ΣτΜ: Ένας φυσικός αριθμός είναι πρώτος όταν διαιρείτε μόνο με τον εαυτό του και τη μονάδα. Π.χ.: 1, 2, 3, 5, 7, 11, 13, 17, ...). Ένα τέτοιο πρόγραμμα θα ήταν κάπως έτσι:

```
TextWindow.Write("Enter a number: ")
i = TextWindow.ReadNumber()
isPrime = "True"
PrimeCheck()
If (isPrime = "True") Then
  TextWindow.WriteLine(i + " is a prime number")
Else
  TextWindow.WriteLine(i + " is not a prime number")
EndIf

Sub PrimeCheck
  For j = 2 To Math.SquareRoot(i)
    If (Math.Remainder(i, j) = 0) Then
      isPrime = "False"
      Goto EndLoop
```

```
    EndIf
  Endfor
EndLoop:
EndSub
```

Η υπορουτίνα PrimeCheck παίρνει την τιμή της μεταβλητής *i* και προσπαθεί να τη διαιρέσει με μικρότερους αριθμούς. Αν ένας αριθμός διαιρεί την *i* και δεν αφήνει υπόλοιπο (remainder), τότε η μεταβλητή *i* δεν είναι πρώτος αριθμός. Στο σημείο αυτό η υπορουτίνα θέτει την μεταβλητή *isPrime* στην τιμή "False" και βγαίνει από τον βρόγχο. Αν ο αριθμός δεν διαιρείτε με άλλους μικρότερους του (και μεγαλύτερους της μονάδας) τότε η τιμή της μεταβλητής *isPrime* παραμένει "True".



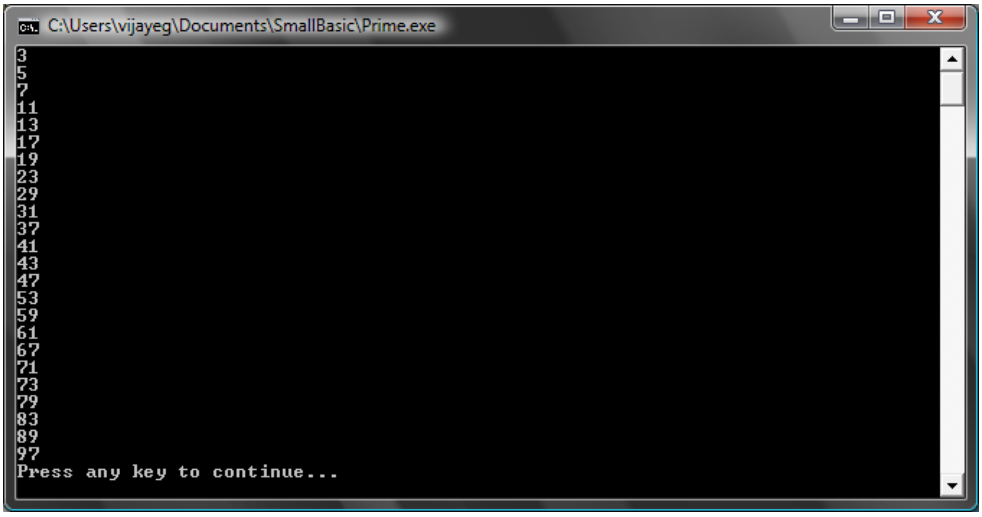
Εικόνα 47 – Έλεγχος πρώτων αριθμών

Τώρα που έχουμε στη διάθεσή μας μια υπορουτίνα που μπορεί να αποφανθεί αν ένας αριθμός είναι πρώτος, ίσως να θέλουμε να τη χρησιμοποιήσουμε για να εμφανίσουμε όλους τους πρώτους αριθμούς που είναι μικρότεροι, ας πούμε, από το 100. Είναι πραγματικά εύκολο να τροποποιήσουμε το παραπάνω πρόγραμμα και να καλούμε την υπορουτίνα *PrimeCheck* μέσα από ένα βρόγχο. Με τον τρόπο αυτό, θα δίνουμε στην υπορουτίνα ένα διαφορετικό αριθμό για έλεγχο κάθε φορά που εκτελούμε το βρόγχο. Ας δούμε πώς αυτό μπορεί να γίνει, με το παρακάτω παράδειγμα:

```
For i = 3 To 100
  isPrime = "True"
  PrimeCheck()
  If (isPrime = "True") Then
    TextWindow.WriteLine(i)
  EndIf
EndFor

Sub PrimeCheck
  For j = 2 To Math.SquareRoot(i)
    If (Math.Remainder(i, j) = 0) Then
      isPrime = "False"
      Goto EndLoop
    EndIf
  Endfor
EndLoop:
EndSub
```

Στο παραπάνω πρόγραμμα, η τιμή της μεταβλητής *i* μεταβάλετε κάθε φορά που εκτελείτε ο βρόγχος. Κάθε φορά που εκτελείτε ο βρόγχος, καλούμε την υπορουτίνα *PrimeCheck* η οποία

παίρνει τη μεταβλητή *i* και υπολογίζει αν είναι πρώτος αριθμός ή όχι. Το αποτέλεσμα αποθηκεύεται στη μεταβλητή *isPrime* η οποία στη συνέχεια ελέγχεται από το υπόλοιπο κομμάτι του βρόγχου και αν είναι “True”, τότε εμφανίζει στην οθόνη την τιμή της μεταβλητής *i*. Εφόσον ο βρόγχος ξεκινά από την τιμή 3 και φτάνει μέχρι το 100, παίρνουμε μία λίστα με όλους τους πρώτους αριθμούς που βρίσκονται μεταξύ του 3 και του 100. Το αποτέλεσμα είναι όπως παρακάτω:



Εικόνα 48 – Πρώτοι αριθμοί

Κεφάλαιο10

Πίνακες (Arrays)

Μέχρι τώρα θα πρέπει να έχετε εντρυφήσει στο πώς χρησιμοποιούμε τις μεταβλητές – εξάλλου φτάσατε μέχρι εδώ και το διασκεδάζετε ακόμα, έτσι δεν είναι;

Ας επανεξετάσουμε λοιπόν το πρώτο μας πρόγραμμα που γράψαμε με μεταβλητές:

```
TextWindow.Write("Enter your Name: ")
name = TextWindow.Read()
TextWindow.WriteLine("Hello " + name)
```

Στο πρόγραμμα αυτό, λαμβάναμε και αποθηκεύαμε το όνομα του χρήστη σε μια μεταβλητή με το όνομα **name**. Στη συνέχεια λέγαμε «Γεια σου» (“Hello”) στο χρήστη. Ας υποθέσουμε τώρα ότι έχουμε παραπάνω από έναν χρήστες – ας πούμε, 5. Πώς θα αποθηκεύαμε τα ονόματά τους; Ένας τρόπος θα ήταν αυτός:

```
TextWindow.Write("User1, enter name: ")
name1 = TextWindow.Read()
TextWindow.Write("User2, enter name: ")
name2 = TextWindow.Read()
TextWindow.Write("User3, enter name: ")
name3 = TextWindow.Read()
TextWindow.Write("User4, enter name: ")
name4 = TextWindow.Read()
TextWindow.Write("User5, enter name: ")
name5 = TextWindow.Read()

TextWindow.Write("Hello ")
TextWindow.Write(name1 + ", ")
TextWindow.Write(name2 + ", ")
```

```
TextWindow.Write(name3 + ", ")
TextWindow.Write(name4 + ", ")
TextWindow.WriteLine(name5)
```

Όταν εκτελέσετε αυτό το πρόγραμμα, τα αποτελέσματα θα είναι κάπως έτσι:

Εικόνα 49 – Χωρίς τη χρήση πινάκων

Θα πρέπει σίγουρα να υπάρχει ένας καλύτερος τρόπος να γράψουμε ένα τέτοιο πρόγραμμα, έτσι δεν είναι; Ειδικά από τη στιγμή που οι υπολογιστές είναι πολύ καλοί στο να εκτελούν επαναλαμβανόμενες λειτουργίες, γιατί να καθόμαστε να γράφουμε τον ίδιο κώδικα ξανά και ξανά για κάθε νέο χρήστη; Το κόλπο εδώ είναι να αποθηκεύουμε και να ανακτούμε το όνομα του κάθε χρήστη χρησιμοποιώντας την ίδια μεταβλητή! Αν θα μπορούσαμε να το κάνουμε αυτό, τότε θα μπορούσαμε να χρησιμοποιήσουμε ένα βρόγχο **For** που μάθαμε στα προηγούμενα κεφάλαια. Εδώ είναι που έρχονται οι Πίνακες (Arrays) να μας βοηθήσουν.

Τί είναι ο πίνακας (array);

Ο πίνακας είναι ένα ιδιαίτερο είδος μεταβλητής που μπορεί να έχει παραπάνω από μία τιμές κάθε στιγμή. Βασικά, αντί να πρέπει να ορίσουμε τις μεταβλητές **name1, name2, name3, name4** και **name5** προκειμένου να αποθηκεύσουμε τα ονόματα 5 χρηστών, θα μπορούσαμε να χρησιμοποιήσουμε μόνο τη μεταβλητή **name** και **με κάποιο τρόπο να** αποθηκεύσουμε εκεί και τα 5 ονόματα. Ο τρόπος που αποθηκεύουμε διαφορετικές τιμές σε μία μεταβλητή είναι με το να χρησιμοποιήσουμε αυτό που αποκαλούμε «δείκτης» (“index”). Για παράδειγμα, τα **name[1], name[2], name[3], name[4]** και **name[5]** μπορούν να αποθηκεύσουν από μία τιμή το κάθε ένα. Οι αριθμοί 1, 2, 3, 4 και 5 καλούνται *δείκτες* (*indeces*) για αυτό τον πίνακα.

Παρόλο που τα **name[1], name[2], name[3], name[4]** και **name[5]** μοιάζουν σαν διαφορετικές μεταβλητές, στην πραγματικότητα είναι μία μεταβλητή. Και ποιό είναι το όφελος από αυτό, μπορείτε να ρωτήσετε. Το μεγαλύτερο πλεονέκτημα στο να αποθηκεύουμε τιμές σε ένα πίνακα είναι ότι μπορείτε να ορίσετε το δείκτη χρησιμοποιώντας μια άλλη μεταβλητή – η οποία θα σας επιτρέψει να έχετε εύκολη πρόσβαση στον πίνακα μέσω της χρήσης βρόγχων.

Τώρα, ας δούμε πως μπορούμε να χρησιμοποιήσουμε τα όσα μάθαμε με το να ξαναγράψουμε το προηγούμενο πρόγραμμα χρησιμοποιώντας πίνακες:

```
For i = 1 To 5
  TextWindow.Write("User" + i + ", enter name: ")
  name[i] = TextWindow.Read()
```

```
EndFor

TextWindow.Write("Hello ")
For i = 1 To 5
  TextWindow.Write(name[i] + ", ")
EndFor
TextWindow.WriteLine("")
```

Διαβάζεται πολύ πιο εύκολα, έτσι δεν είναι; Προσέξτε τις δύο **έντονες** γραμμές παραπάνω. Η πρώτη αποθηκεύει τιμές στον πίνακα και η δεύτερη διαβάζει τις τιμές από τον πίνακα. Η τιμή που αποθηκεύσατε στην **name[1]** δεν θα επηρεαστεί από την τιμή που αποθηκεύσατε στην **name[2]**. Ως εκ τούτου, μπορείτε για τις περισσότερες των περιπτώσεων να χρησιμοποιήσετε τις **name[1]** και **name[2]** σαν δυο διαφορετικές μεταβλητές μέσα στο ίδιο πρόγραμμα.

Εικόνα 50 – Χρησιμοποιώντας Πίνακες

Το παραπάνω πρόγραμμα δίνει σχεδόν τα ίδια αποτελέσματα με αυτό χωρίς πίνακες, εκτός από το κόμμα στο τέλος του *Mantis*. Μπορούμε να το διορθώσουμε ξαναγράφοντας το βρόγχο ως εξής:

```
TextWindow.Write("Hello ")
For i = 1 To 5
  TextWindow.Write(name[i])
  If i < 5 Then
    TextWindow.Write(", ")
  EndIf
EndFor
TextWindow.WriteLine("")
```

Καταρτίζωντας ευρετήριο σε ένα πίνακα (Indexing an array)

Στο προηγούμενο πρόγραμμα είδατε ότι χρησιμοποιήσαμε αριθμούς σαν δείκτες για να αποθηκεύσουμε και να ανακτήσουμε τιμές σε ένα πίνακα. Αποδεικνύεται όμως ότι οι δείκτες δεν περιορίζονται μόνο σε αριθμούς και στην πράξη είναι πολύ χρήσιμο να χρησιμοποιούμε και κείμενο σαν δείκτες. Για παράδειγμα, στο παρακάτω πρόγραμμα ρωτάμε και αποθηκεύουμε διάφορες πληροφορίες για ένα χρήστη και στη συνέχεια εμφανίζουμε όποια πληροφορία ζητήσει ο χρήστης:

```
TextWindow.Write("Enter name: ")
user["name"] = TextWindow.Read()
TextWindow.Write("Enter age: ")
user["age"] = TextWindow.Read()
TextWindow.Write("Enter city: ")
user["city"] = TextWindow.Read()
TextWindow.Write("Enter zip: ")
user["zip"] = TextWindow.Read()

TextWindow.Write("What info do you want? ")
index = TextWindow.Read()
TextWindow.WriteLine(index + " = " + user[index])
```

Εικόνα 51 – Χρησιμοποιώντας μή-αριθμητικούς δείκτες

Περισσότερες από μία διαστάσεις

Ας υποθέσουμε ότι θέλετε να αποθηκεύσετε το όνομα και το τηλέφωνο όλων των φίλων σας και να μπορείτε οποιαδήποτε στιγμή να αναζητήσετε κάποιο τηλέφωνο – κάτι σαν έναν τηλεφωνικό κατάλογο. Πως θα μπορούσατε να γράψετε ένα τέτοιο πρόγραμμα;

> *Όπως και με τις μεταβλητές, οι δείκτες πινάκων δεν εξαρτώνται από το αν τους γράφουμε με πεζά ή κεφαλαία.*

Στην περίπτωση αυτή, υπάρχουν δύο σετ από δείκτες (γνωστές και σαν *διαστάσεις* του πίνακα). Ας υποθέσουμε ότι θα προσδιορίζουμε τον κάθε φίλο μας με βάση το χαϊδευτικό του (nickname). Αυτό θα αποτελεί τον πρώτο δείκτη στο πίνακά μας. Μόλις με βάση τον πρώτο δείκτη εντοπίσουμε το φίλο μας, ο δεύτερος δείκτης που θα χρησιμοποιήσουμε θα μας δίνει το όνομα και τον αριθμό του, με βάση τις μεταβλητές **name** και **phone**. Ο τρόπος με τον οποίο θα αποθηκεύουμε τα δεδομένα θα έχει ως εξής:

```
friends["Rob"]["Name"] = "Robert"
friends["Rob"]["Phone"] = "555-6789"

friends["VJ"]["Name"] = "Vijaye"
friends["VJ"]["Phone"] = "555-4567"

friends["Ash"]["Name"] = "Ashley"
```

```
friends["Ash"]["Phone"] = "555-2345"
```

Καθώς έχουμε δύο δείκτες στον ίδιο πίνακα **friends**, ο πίνακας αυτό θα καλείται δυσδιάστατος πίνακας.

Από τη στιγμή που έχουμε γράψει το πρόγραμμα έως εδώ, μπορούμε να πάρουμε σαν στοιχείο εισόδου το χαϊδευτικό όνομα του φίλου μας και μετά να εμφανίσουμε την πληροφορία που έχουμε αποθηκεύσει για αυτόν. Εδώ είναι το πλήρες πρόγραμμα:

```
friends["Rob"]["Name"] = "Robert"
friends["Rob"]["Phone"] = "555-6789"

friends["VJ"]["Name"] = "Vijaye"
friends["VJ"]["Phone"] = "555-4567"

friends["Ash"]["Name"] = "Ashley"
friends["Ash"]["Phone"] = "555-2345"

TextWindow.Write("Enter the nickname: ")
nickname = TextWindow.Read()

TextWindow.WriteLine("Name: " + friends[nickname]["Name"])
TextWindow.WriteLine("Phone: " + friends[nickname]["Phone"])
```

Εικόνα 52 – Ένας απλός τηλεφωνικός κατάλογος

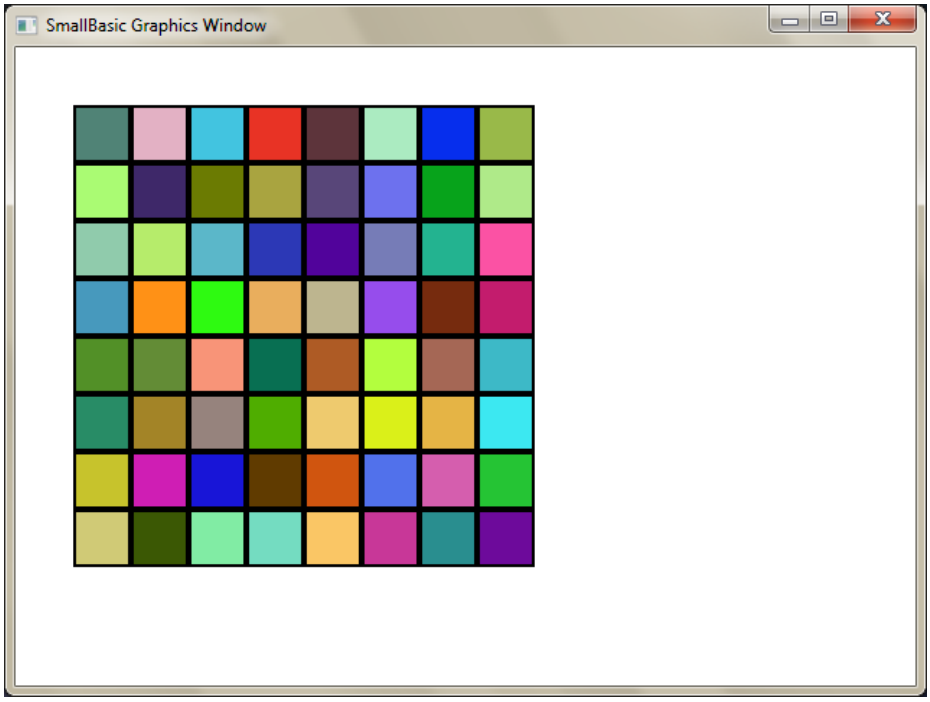
Χρησιμοποιώντας Πίνακες για να παραστήσουμε πλέγματα (grids)

Μια πολύ συχνή χρήση των πολυ-διάστατων πινάκων είναι για να παραστήσουμε πλέγματα (grids). Τα πλέγματα έχουν σειρές και στήλες, οι οποίες μπορούν πολύ όμορφα να ταιριάξουν σε ένα δυσδιάστατο πίνακα. Ένα απλό πρόγραμμα που παρατάσσει κουτιά σε ένα πλέγμα δίνεται παρακάτω:

```
rows = 8
columns = 8
size = 40

For r = 1 To rows
```

```
  For c = 1 To columns
    GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()
    boxes[r][c] = Shapes.AddRectangle(size, size)
    Shapes.Move(boxes[r][c], c * size, r * size)
  EndFor
EndFor
```

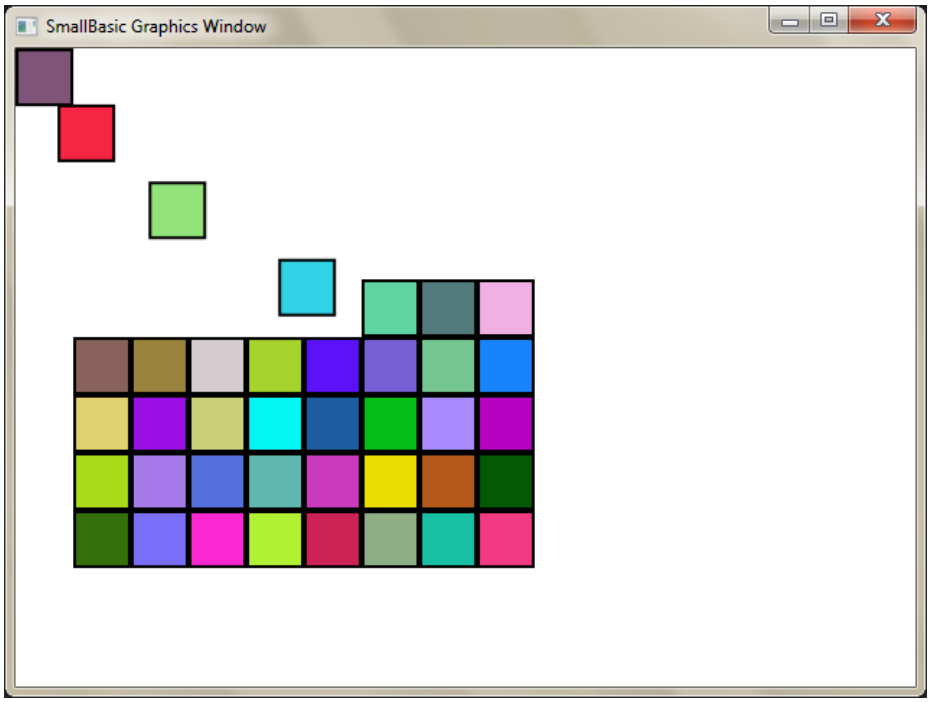
Το πρόγραμμα αυτό προσθέτει ορθογώνια παραλληλόγραμμα και τα τοποθετεί με ένα τρόπο που να σχηματίσουν ένα πλέγμα 8 επί 8. Επιπροσθέτως, το πρόγραμμα αποθηκεύει αυτά τα ορθογώνια σε ένα πίνακα. Με αυτό τον τρόπο καθιστά εύκολη δουλειά για εμάς να τα παρακολουθούμε και να τα ξαναχρησιμοποιήσουμε όποτε το θελήσουμε:



Εικόνα 53 – Παρατάσσοντας ορθογώνια σε ένα πλέγμα

Για παράδειγμα, προσθέτοντας τον παρακάτω κώδικα στο τέλος του προηγούμενου προγράμματος θα έχουμε τα ορθογώνια να μετακινούνται στην πάνω αριστερή γωνία:

```
For r = 1 To rows
  For c = 1 To columns
    Shapes.Animate(boxes[r][c], 0, 0, 1000)
    Program.Delay(300)
  EndFor
EndFor
```



Εικόνα 54 – Παρακολουθώντας τα ορθογώνια στο πλέγμα

Κεφάλαιο 11

Γεγονότα και Αλληλεπίδραση (Events and Interactivity)

Στα πρώτα δύο κεφάλαια μιλήσαμε για αντικείμενα τα οποία έχουν *Ιδιότητες* (*Properties*) και *Λειτουργίες* (*Operations*). Επιπλέον των Ιδιοτήτων και των Λειτουργιών, κάποια αντικείμενα έχουν αυτό που ονομάζουμε ***Γεγονότα*** (***Events***). Τα Γεγονότα είναι σαν σήματα που προκαλούνται, για παράδειγμα, σαν απάντηση σε μια πράξη του χρήστη, όπως το να μετακινήσει το ποντίκι ή να κάνει κλικ. Στην περίπτωση των λειτουργιών, εσείς σαν προγραμματιστής τις καλείτε προκειμένου να ενεργοποιήσετε τον υπολογιστή να κάνει κάτι, ενώ στην περίπτωση των γεγονότων ο υπολογιστής σας ενημερώνει όταν κάτι ενδιαφέρον έχει συμβεί.

Πώς τα γεγονότα μας είναι χρήσιμα ;

Τα γεγονότα έχουν ουσιαστική σημασία στην εισαγωγή της αλληλεπίδρασης (interactivity) σε ένα πρόγραμμα. Αν θέλετε να επιτρέπετε στους χρήστες να αλληλεπιδρούν με το πρόγραμμά σας αρκεί να χρησιμοποιήσετε τα γεγονότα. Ας πούμε, γράφετε το παιχνίδι Tic-Tac-Toe και θέλετε να επιτρέπετε στο χρήστη να επιλέξει το πώς θα παίξει, ωραία; Εδώ είναι που εμπλέκονται τα γεγονότα– λαμβάνετε τις επιλογές του χρήστη στο πρόγραμμά σας χρησιμοποιώντας τα γεγονότα. Αν σας φαίνεται δυσκολονόητο, μην ανησυχείτε, θα δούμε ένα πολύ απλό πρόγραμμα που θα μας βοηθήσει να κατανοήσουμε το τι είναι τα γεγονότα και πώς μπορούμε να τα χρησιμοποιήσουμε.

Παρακάτω έχουμε ένα πολύ απλό πρόγραμμα που έχει μόνο μία δήλωση και μία υπορουτίνα. Η υπορουτίνα χρησιμοποιεί τη λειτουργία *ShowMessage* στο αντικείμενο GaphicsWindow προκειμένου να εμφανίσει ένα μήνυμα στο χρήστη.

```
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  GraphicsWindow.ShowMessage("You Clicked.", "Hello")
```

```
EndSub
```

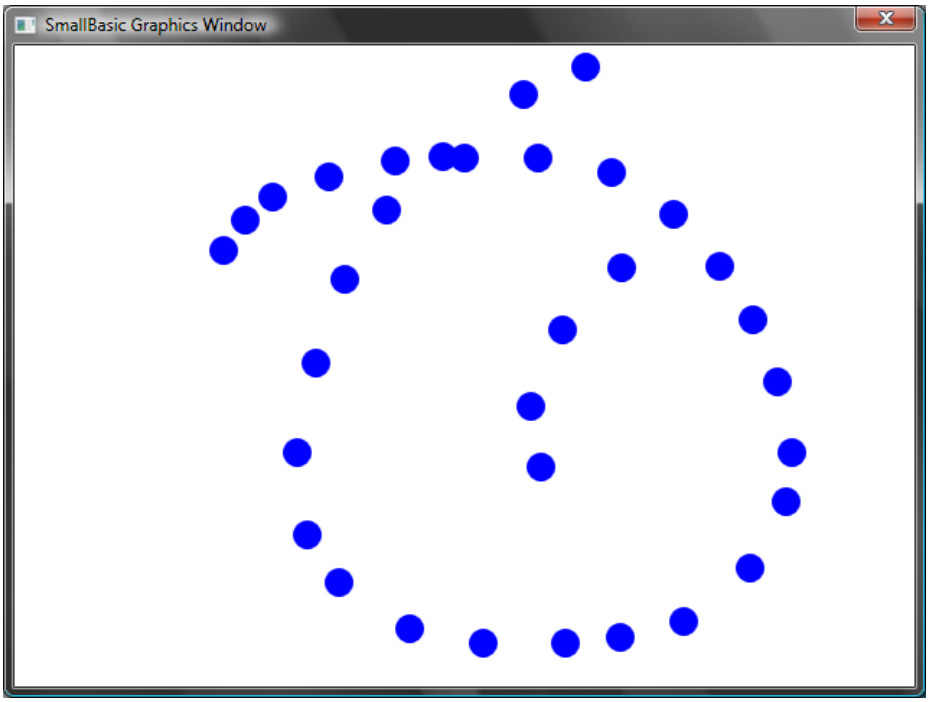
Το ενδιαφέρον κομμάτι σε αυτό το πρόγραμμα είναι στο σημείο που αναθέτουμε το όνομα της υπορουτίνας *OnMouseDown* στο γεγονός **MouseDown** του αντικειμένου GraphicsWindow. Θα έχετε μάλλον προσέξει ότι το γεγονός MouseDown μοιάζει αρκετά με μια ιδιότητα (property) –με τη διαφορά ότι αντί να της αναθέτουμε κάποια τιμή, της αναθέτουμε την υπορουτίνα *OnMouseDown*. Αυτό είναι και το ιδιαίτερο χαρακτηριστικό των γεγονότων (events) – όταν συμβεί το γεγονός, η υπορουτίνα καλείται αυτόματα. Στην περίπτωσή μας, η υπορουτίνα *OnMouseDown* καλείται κάθε φορά που ο χρήστης κάνει κλικ με το ποντίκι πάνω στο παράθυρο γραφικών (GraphicWindow). Εμπρός λοιπόν, τρέξτε το πρόγραμμα και δοκιμάστε το. Κάθε φορά που κάνετε κλικ με το ποντίκι σας στο παράθυρο γραφικών, θα βλέπετε ένα παράθυρο μηνύματος (message box) όπως αυτό στην παρακάτω εικόνα:

Εικόνα 55 – Απόκριση σε ένα γεγονός

Αυτού του είδους η διαχείριση των γεγονότων είναι πολύ αποτελεσματική και μας επιτρέπει να υλοποιούμε πολύ δημιουργικά και ενδιαφέροντα προγράμματα. Τα προγράμματα που φτιάχνονται με αυτό τον τρόπο συχνά αποκαλούνται «γεγονοτο-οδηγούμενα» (“event-driven”) προγράμματα.

Μπορείτε να τροποποιήσετε την υπορουτίνα *OnMouseDown* προκειμένου να κάνετε και άλλα πράγματα εκτός από το να εμφανίζετε το παράθυρο μηνύματος. Για παράδειγμα, όπως φαίνεται και στο παρακάτω πρόγραμμα, μπορείτε να σχεδιάσετε μεγάλες μπλε τελείες στο σημείο του παραθύρου που ο χρήστης έκανε κλικ με το ποντίκι.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  x = GraphicsWindow.MouseX - 10
  y = GraphicsWindow.MouseY - 10
  GraphicsWindow.FillEllipse(x, y, 20, 20)
EndSub
```



Εικόνα 56 – Διαχειρίζοντας το γεγονός MouseDown

Παρατηρείστε ότι στο ανωτέρω πρόγραμμα, χρησιμοποιούμε τα *MouseX* και *MouseY* για να πάρουμε τις συντεταγμένες του ποντικιού. Εν συνεχεία, χρησιμοποιούμε αυτές τις τιμές σαν το κέντρο του κύκλου που στη συνέχεια ζωγραφίζουμε.

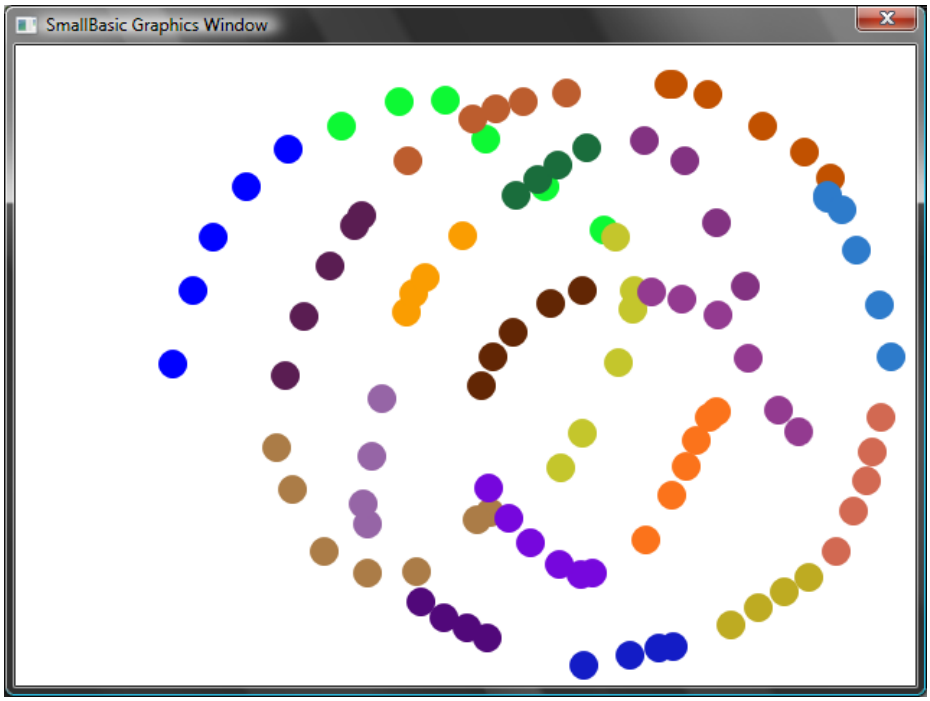
Χειρισμός πολλαπλών γεγονότων

Ουσιαστικά δεν υπάρχει όριο στον αριθμό των γεγονότων που μπορείτε να χειριστείτε στα προγράμματά σας. Μπορείτε ακόμα και να φτιάξετε μια υπορουτίνα που να χειρίζεται πολλαπλά γεγονότα. Παρόλα αυτά, θα μπορείτε να χειριστείτε ένα μόνο γεγονός κάθε φορά. Αν επιχειρήσετε να αναθέσετε δύο υπορουτίνες στο ίδιο γεγονότος, η δεύτερη υπορουτίνα θα κερδίσει.

Για να το εξηγήσουμε αυτό, ας πάρουμε το προηγούμενο παράδειγμα και να προσθέσουμε μια υπορουτίνα που θα χειρίζεται το πάτημα των πλήκτρων. Θα κάνουμε επίσης την υπορουτίνα να αλλάζει το χρώμα του πινέλου έτσι ώστε κάθε φορά που θα κάνετε κλικ με το ποντίκι σας, να ζωγραφίζουμε μια τελεία διαφορετικού χρώματος.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown
GraphicsWindow.KeyDown = OnKeyDown

Sub OnKeyDown
  GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()
EndSub

Sub OnMouseDown
  x = GraphicsWindow.MouseX - 10
  y = GraphicsWindow.MouseY - 10
```

```
  GraphicsWindow.FillEllipse(x, y, 20, 20)
EndSub
```



Εικόνα 57 – Χειρισμός πολλαπλών γεγονότων

Αν εκτελούσατε αυτό το πρόγραμμα και κάνατε κλικ στο παράθυρο, θα βλέπατε μία μπλε τελεία. Αν μετά πατούσατε ένα οποιοδήποτε πλήκτρο και μετά κάνατε πάλι κλικ, θα βλέπατε μια τελεία διαφορετικού χρώματος. Αυτό που συμβαίνει είναι ότι όταν πατάτε ένα πλήκτρο εκτελείται η υπορουτίνα *OnKeyDown* και η οποία αλλάζει το χρώμα του πινέλου σε ένα τυχαίο. Μετά από αυτό, όταν κάνετε κλικ με το ποντίκι, η τελεία που θα ζωγραφιστεί θα έχει το χρώμα που μόλις αλλάξαμε, ζωγραφίζοντας έτσι πολλές πολύχρωμες τελείες.

Ένα πρόγραμμα ζωγραφικής

Εξοπλισμένοι λοιπόν τώρα με γεγονότα και υπορουτίνες, μπορούμε να γράψουμε ένα πρόγραμμα που θα επιτρέπει στους χρήστες να ζωγραφίζουν. Είναι απίστευτα εύκολο το να γράψουμε ένα τέτοιο πρόγραμμα, αρκεί να αναλύσουμε το πρόβλημά μας σε μικρότερα επιμέρους τμήματα. Σαν πρώτο βήμα, θα γράψουμε ένα πρόγραμμα που θα επιτρέπει στο χρήστη να μετακινήσει το ποντίκι οπουδήποτε μέσα στο παράθυρο γραφικών, αφήνοντας πίσω του ένα ίχνος από τα σημεία που πέρασε.

```
GraphicsWindow.MouseMove = OnMouseMove

Sub OnMouseMove
  x = GraphicsWindow.MouseX
  y = GraphicsWindow.MouseY
  GraphicsWindow.DrawLine(prevX, prevY, x, y)
```

```
  prevX = x
  prevY = y
EndSub
```

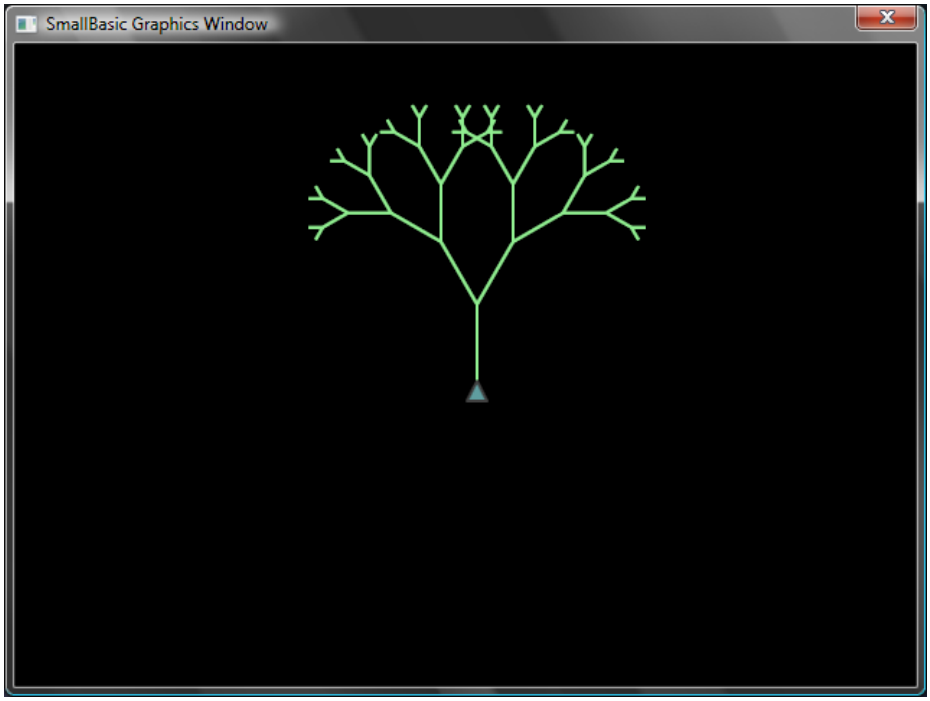
Ωστόσο, κατά την εκτέλεση αυτού του προγράμματος, η πρώτη γραμμή ξεκινά πάντα από την πάνω αριστερή γωνία του παραθύρου (συντεταγμένες 0,0). Θα διορθώσουμε αυτό το πρόβλημα με κατάλληλο χειρισμό του γεγονότος *MoudeDown* όπου οι μεταβλητές prevX και prevY (προηγούμενη X και προηγούμενη Y συντεταγμένη) θα καταγράφονται κάθε φορά που θα συμβαίνει το γεγονός *MouseDown*.

Επίσης, θα καταγράφουμε το ίχνος της μετακίνησης μόνο όσο ο χρήστης κρατάει πατημένο το αριστερό πλήκτρο του ποντικιού. Τις άλλες φορές που το πλήκτρο δεν είναι πατημένο δεν θα ζωγραφίζουμε τίποτα. Προκείμενου να προγραμματίσουμε αυτή τη συμπεριφορά, θα χρησιμοποιήσουμε την ιδιότητα *IsLeftButtonDown* του αντικειμένου **Mouse**. Η ιδιότητα αυτή μας λέει αν το αριστερό πλήκτρο του ποντικιού είναι πατημένο ή όχι. Αν αυτό συμβαίνει (η τιμή αυτής της ιδιότητας είναι αληθής) τότε θα ζωγραφίζουμε το ίχνος, διαφορετικά θα παρακάμπτουμε το κομμάτι αυτό του κώδικα.

```
GraphicsWindow.MouseMove = OnMouseMove
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  prevX = GraphicsWindow.MouseX
  prevY = GraphicsWindow.MouseY
EndSub

Sub OnMouseMove
  x = GraphicsWindow.MouseX
  y = GraphicsWindow.MouseY
  If (Mouse.IsLeftButtonDown) Then
    GraphicsWindow.DrawLine(prevX, prevY, x, y)
  EndIf
  prevX = x
  prevY = y
EndSub
```

Διασκεδαστικά Παραδείγματα

Fractal Χελώνας



Εικόνα 58 – Η χελώνα ζωγραφίζει ένα δέντρο fractal

```
angle = 30
delta = 10
distance = 60
Turtle.Speed = 9
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
DrawTree()
```

```
Sub DrawTree
  If (distance > 0) Then
    Turtle.Move(distance)
    Turtle.Turn(angle)

    Stack.PushValue("distance", distance)
    distance = distance - delta
    DrawTree()
    Turtle.Turn(-angle * 2)
    DrawTree()
    Turtle.Turn(angle)
    distance = Stack.PopValue("distance")

    Turtle.Move(-distance)
  EndIf
EndSub
```

Φωτογραφίες από το Flickr

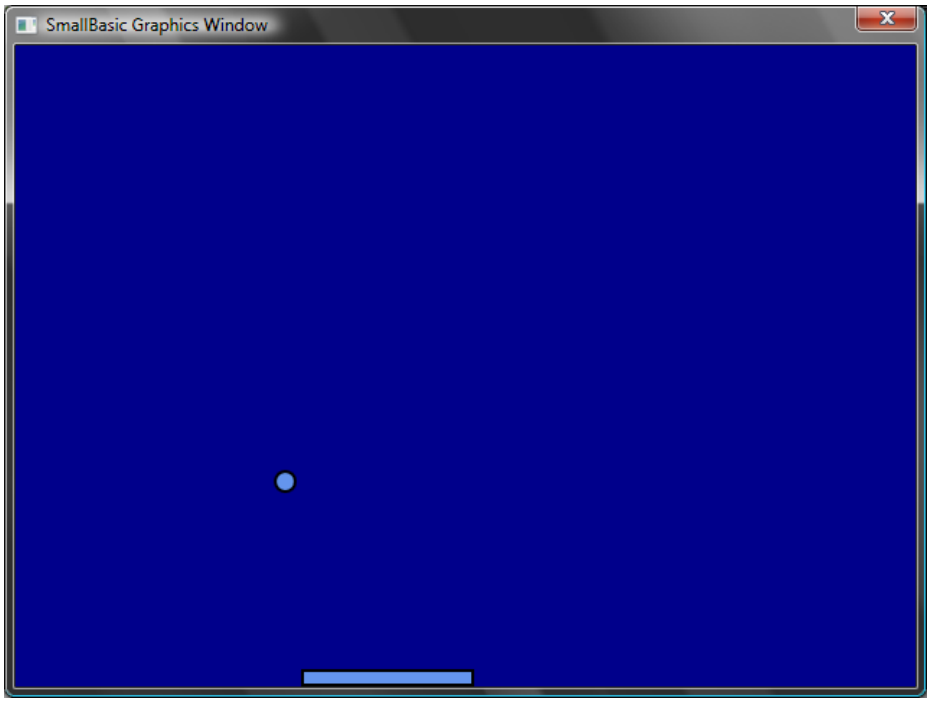
Εικόνα 59 – Διαβάζοντας εικόνες από το Flickr

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
    pic = Flickr.GetRandomPicture("mountains, river")
    GraphicsWindow.DrawResizedImage(pic, 0, 0, 640, 480)
EndSub
```

Δυναμικό Φόντο του Desktop (Desktop Wallpaper)

```
For i = 1 To 10
  pic = Flickr.GetRandomPicture("mountains")
  Desktop.SetWallPaper(pic)
  Program.Delay(10000)
EndFor
```

Παιχνίδι με Ρακέτα (Paddle)



Εικόνα 60 – Παιχνίδι Ρακέτας

```
GraphicsWindow.BackgroundColor = "DarkBlue"
paddle = Shapes.AddRectangle(120, 12)
ball = Shapes.AddEllipse(16, 16)
GraphicsWindow.MouseMove = OnMouseMove

x = 0
y = 0
deltaX = 1
deltaY = 1

RunLoop:
  x = x + deltaX
  y = y + deltaY

  gw = GraphicsWindow.Width
```

```
  gh = GraphicsWindow.Height
  If (x >= gw - 16 or x <= 0) Then
    deltaX = -deltaX
  EndIf
  If (y <= 0) Then
    deltaY = -deltaY
  EndIf

  padX = Shapes.GetLeft (paddle)
  If (y = gh - 28 and x >= padX and x <= padX + 120) Then
    deltaY = -deltaY
  EndIf

  Shapes.Move(ball, x, y)
  Program.Delay(5)

  If (y < gh) Then
    Goto RunLoop
  EndIf

GraphicsWindow.ShowMessage("You Lose", "Paddle")

Sub OnMouseMove
  paddleX = GraphicsWindow.MouseX
  Shapes.Move(paddle, paddleX - 60, GraphicsWindow.Height - 12)
EndSub
```

Παράρτημα Β

Χρώματα

Εδώ παραθέτουμε μία λίστα με τους κωδικούς των χρωμάτων που υποστηρίζονται από την Small Basic, κατηγοριοποιημένη ως προς τις βασικές αποχρώσεις.

Red Colors

IndianRed	#CD5C5C
LightCoral	#F08080
Salmon	#FA8072
DarkSalmon	#E9967A
LightSalmon	#FFA07A
Crimson	#DC143C
Red	#FF0000
FireBrick	#B22222
DarkRed	#8B0000

Pink Colors

Pink	#FFC0CB
LightPink	#FFB6C1
HotPink	#FF69B4
DeepPink	#FF1493
MediumVioletRed	#C71585
PaleVioletRed	#DB7093

Orange Colors

LightSalmon	#FFA07A
Coral	#FF7F50
Tomato	#FF6347
OrangeRed	#FF4500
DarkOrange	#FF8C00
Orange	#FFA500

Yellow Colors

Gold	#FFD700
Yellow	#FFFF00
LightYellow	#FFFFE0
LemonChiffon	#FFFACD
LightGoldenrodYellow	#FAFAD2
PapayaWhip	#FFEFD5
Moccasin	#FFE4B5
PeachPuff	#FFDAB9
PaleGoldenrod	#EEE8AA
Khaki	#F0E68C
DarkKhaki	#BDB76B

Purple Colors

Lavender	#E6E6FA
Thistle	#D8BFD8
Plum	#DDA0DD
Violet	#EE82EE
Orchid	#DA70D6
Fuchsia	#FF00FF
Magenta	#FF00FF
MediumOrchid	#BA55D3
MediumPurple	#9370DB
BlueViolet	#8A2BE2
DarkViolet	#9400D3
DarkOrchid	#9932CC
DarkMagenta	#8B008B
Purple	#800080
Indigo	#4B0082
SlateBlue	#6A5ACD
DarkSlateBlue	#483D8B
MediumSlateBlue	#7B68EE

Green Colors

GreenYellow	#ADFF2F
Chartreuse	#7FFF00
LawnGreen	#7CFC00
Lime	#00FF00
LimeGreen	#32CD32
PaleGreen	#98FB98
LightGreen	#90EE90
MediumSpringGreen	#00FA9A
SpringGreen	#00FF7F

MediumSeaGreen	#3CB371
SeaGreen	#2E8B57
ForestGreen	#228B22
Green	#008000
DarkGreen	#006400
YellowGreen	#9ACD32
OliveDrab	#6B8E23
Olive	#808000
DarkOliveGreen	#556B2F
MediumAquamarine	#66CDAA
DarkSeaGreen	#8FBC8F
LightSeaGreen	#20B2AA
DarkCyan	#008B8B
Teal	#008080

Blue Colors

Aqua	#00FFFF
Cyan	#00FFFF
LightCyan	#E0FFFF
PaleTurquoise	#AFEEEE
Aquamarine	#7FFFD4
Turquoise	#40E0D0
MediumTurquoise	#48D1CC
DarkTurquoise	#00CED1
CadetBlue	#5F9EA0
SteelBlue	#4682B4
LightSteelBlue	#B0C4DE
PowderBlue	#B0E0E6
LightBlue	#ADD8E6
SkyBlue	#87CEEB
LightSkyBlue	#87CEFA
DeepSkyBlue	#00BFFF
DodgerBlue	#1E90FF
CornflowerBlue	#6495ED
MediumSlateBlue	#7B68EE
RoyalBlue	#4169E1
Blue	#0000FF
MediumBlue	#0000CD
DarkBlue	#00008B
Navy	#000080
MidnightBlue	#191970

Brown Colors

Cornsilk	#FFF8DC
BlanchedAlmond	#FFEBCD
Bisque	#FFE4C4
NavajoWhite	#FFDEAD
Wheat	#F5DEB3
BurlyWood	#DEB887
Tan	#D2B48C
RosyBrown	#BC8F8F
SandyBrown	#F4A460
Goldenrod	#DAA520
DarkGoldenrod	#B8860B
Peru	#CD853F
Chocolate	#D2691E
SaddleBrown	#8B4513
Sienna	#A0522D
Brown	#A52A2A
Maroon	#800000

White Colors

White	#FFFFFF
Snow	#FFFAFA
Honeydew	#F0FFF0
MintCream	#F5FFFA
Azure	#F0FFFF
AliceBlue	#F0F8FF
GhostWhite	#F8F8FF

WhiteSmoke	#F5F5F5
Seashell	#FFF5EE
Beige	#F5F5DC
OldLace	#FDF5E6
FloralWhite	#FFFAF0
Ivory	#FFFFF0
AntiqueWhite	#FAEBD7
Linen	#FAF0E6
LavenderBlush	#FFF0F5
MistyRose	#FFE4E1

Gray Colors

Gainsboro	#DCDCDC
LightGray	#D3D3D3
Silver	#C0C0C0
DarkGray	#A9A9A9
Gray	#808080
DimGray	#696969
LightSlateGray	#778899
SlateGray	#708090
DarkSlateGray	#2F4F4F
Black	#000000