Microsoft Small Basic

مقدمة فى البرمجة

الفصل 1

مقدمــة

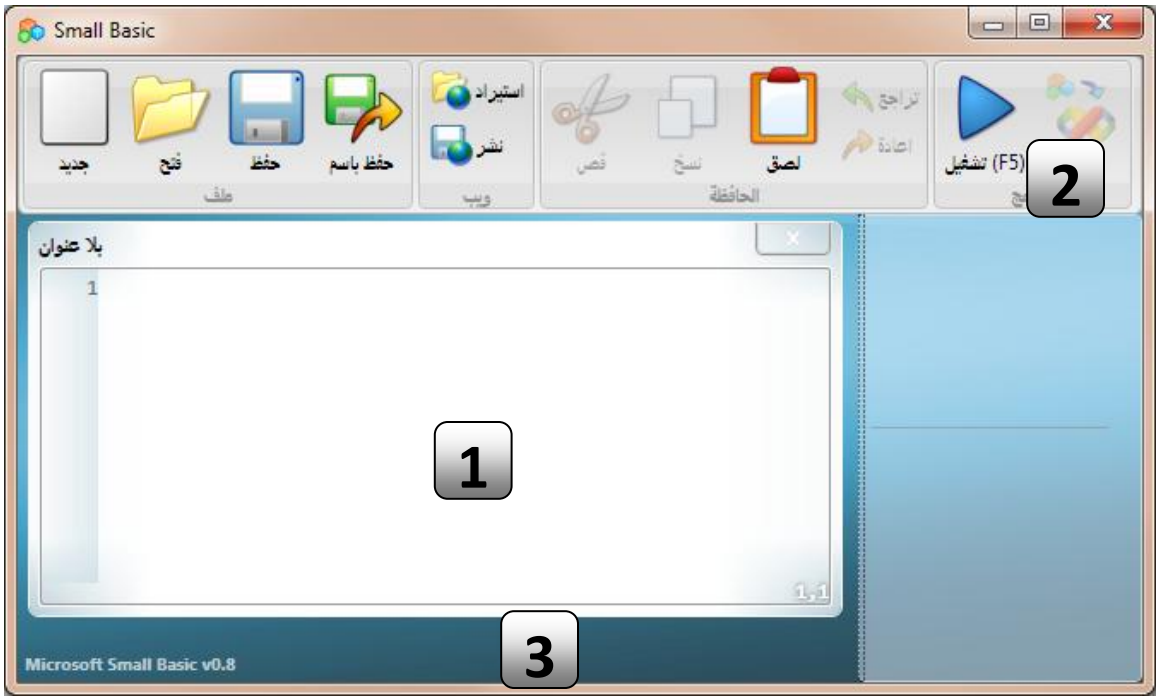
Small Basic والبــرمجـــة

البرمجة هى عملية تكوين برنامج باستخدام لغة البرمجة. كما نفهم ونتحدث الانجليزية او الاسبانية او الفرنسية، يستطيع اكمبيوتر ان يفهم البرامج المكتوبة بلغات معينة. تسمى هذه اللغات لغات البرمجة. فى البداية كان هناك عدد قليل من لغات البرمجة وكان من السهل ان تفهم وتتعلم هذه اللغات. ولكن مع زيادة تعقيد اجهزة الكمبيوتر والبرامج، تطورت لغات البرمجة بسرعة لتشمل مفاهيم اكثر تعقيدا على مدار الوقت. بالتالى اصبحت غالبية لغات البرمجة الحديثة والمفاهيم الخاصة بها صعبة الفهم بالنسبة للمبتدىء. وقد بدأت هذه الحقيقة تثبيط الناس من التعلم أو محاولة برمجة الكمبيوتر.

Small Basic هو لغة برمجة مصممة خصيصا لجعل البرمجة سهلة للغاية وممتعة بالنسبة للمبتدئين. الهدف من Small Basic كسر الحاجز وتوفير الخطوات الاولى الى عالم البرمجة الرائع.

البيئة الخاصة بالـ Small Basic

لنبدأ بمقدمة سريعة فى البيئة الخاصة بـ Small Basic. عندما تبدأ تشغيل Small Basic سوف ترى اطار مماثل للرسم التوضيحى التالى.



الشكل 1 البيئة الخاصة بـ Small Basic

هذه هى البيئة الخاصة بالــ Small Basic التى سوف تقوم فيها بكتابة وتشغيل برامج الــ Small Basic. هذه البيئة تشمل العديد من العناصر التى تم تعريفها بالارقام.

المحرر، معرف بــ (1) هو المكان الذى سنقوم فيه بكتابة برامج الــ Small Basic. عندما تقوم بفتح نموذج لبرنامج (Template) او برنامج تم حفظه مسبقا، سوف يظهر على هذا المحرر. يمكنك بعد ذلك تعديله وحفظه للاستخدام فيما بعد.

يمكنك ايضا ان تقوم بفتح وتشغيل اكثر من برنامج فى وقت واحد. كل برنامج تقوم بتشغيله سيتم عرضه فى محرر منفصل. المحرر الذى يحتوى على البرنامج الذى تقوم بتشغيله حاليا يسمى بــ المحرر النشط.

شريط الادوات، المعرف ب (2) يستخدم فى اصدار اوامر اما الى المحرر النشط او الى البيئة. سوف نتعلم الاوامر المختلفة فى شريط الادوات لاحقا.

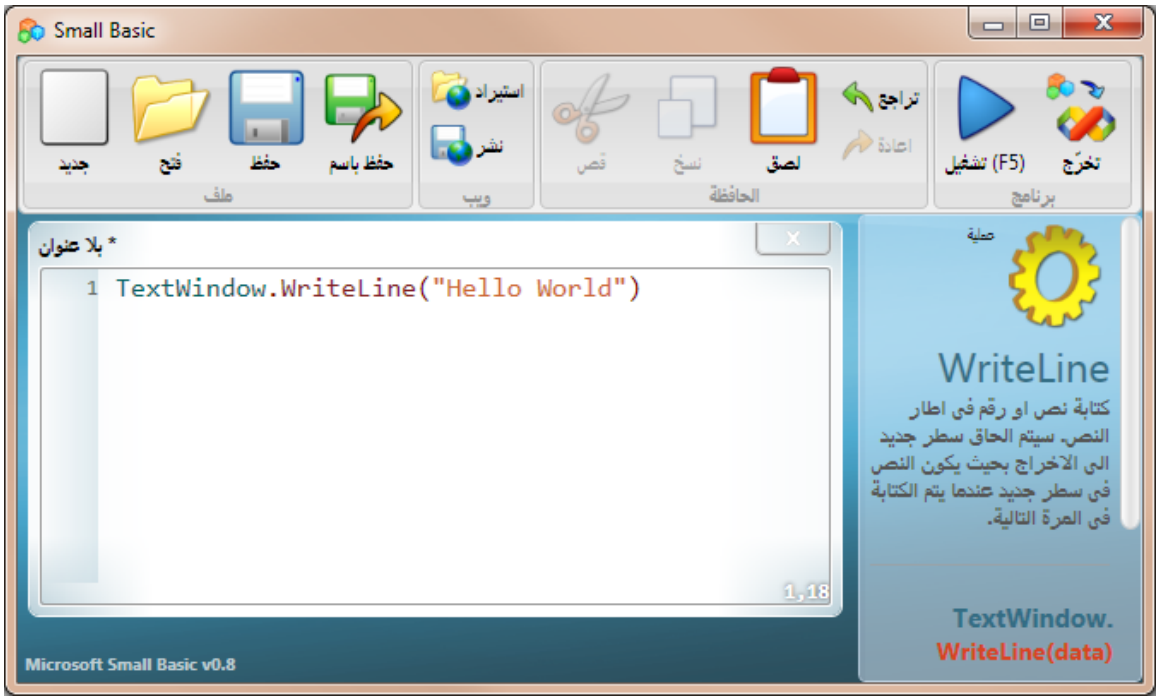
السطح، المعرف بـ (3) هو المكان الذى تظهر فيه اطارات المحرر.

برنامجنا الاول

بما انك اصبحت مالوفا بالبيئة الخاصة بــ Small Basic، سوف نبدا البرمجة فيها. كما هو موضح اعلاه، المحرر هو المكان الذى نكتب فيه برامجنا. فهيا بنا نبدأ بكتابة السطر التالى فى المحرر.

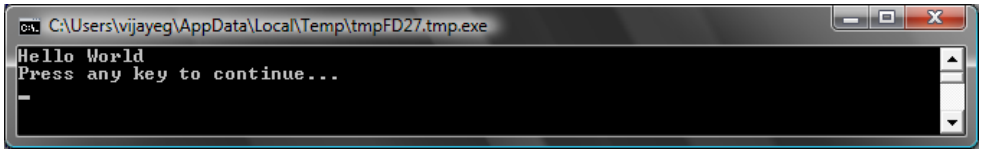
```
TextWindow.WriteLine("Hello World")
```

هذا هو برنامجنا الاول فى الـ Small Basic. ولو قمت بكتابته صحيحا، سوف ترى شيئا مماثلا للرسم التوضيحى اسفل.



الشكل 2 - اول برنامج

بما اننا قمنا بكتابة برنامجنا الجديد، هيا نقوم بتشغيله لنرى ماذا سيحدث. يمكننا ان نقوم بتشغيل برنامجنا اما بالنقر على زر التشغيل الموجود على شريط الادوات او باستخدام مفتاح الاختصار F5 الموجود على لوحة المفاتيح. لو كل شيء تم على ما يرام، سيتم تشغيل برنامجنا وستكون النتائج كما هو موضح اسفل.



الشكل 3 – اول اخراج لبرنامج

مبروك! لقد قمت بكتابة وتشغيل اول برنامج . هو برنامج صغير جدا وسهل، ولكن فى نفس الوقت خطوة كبيرة لكى تصبح مبرمج حقيقى! الآن، يجب ان نغطى تفصيلة واحدة اضافية قبل ان نقوم بعمل برامج اكبر. يجب ان نفهم ماذا حدث – ماذا قلنا لجهاز الكبيوتر وكيف عرف جهاز الكمبيوتر ماذا يفعل؟ فى الفصل التالى، سنقوم بتحليل البرنامج الذى قمنا بكتابته، لكى نفهم ماذا يحدث.

الشكل 4 - Intellisense

حفظ برنامجنا

اذا اردت ان تغلق الــ Small Basic وتقوم بتشغيل البرنامج الذى انتهيت من كتابته فى وقت لاحق، يمكنك ان تحفظ البرنامج. ايضا حفظ البرامج من ان لآخر عادة مفيدة حتى لا تفقد المعلومات فى حالة ايقاف التشغيل الطارىء او فى حالة انقطاع الطاقة. يمكنك ان تقوم بحفظ برنامجك الحالى بالنقر على رمز "حفظ" الموجود بشريط الادوات او باستخدام الاختصار "Ctrl+S" (اضغط على مفتاح S مع استمرارية الضغط على مفتاح Ctrl).

الفصل 2

فهم برنامجنا الاول

ما هو برنامج الكمبيوتر؟

البرنامج هو مجموعة من الاوامر الموجهة الى الكمبيوتر. هذه الاوامر تحدد للكمبيوتر ماذا يفعل بالتفصيل، وغالبا ما يتبع الكمبيوتر هذه الاوامر. الكمبيوتر مثل الاشخاص، يقوم فقط باتباع الاوامر اذا كانت بلغة يفهمها. تسمى هذه الاوامر بلغات البرمجة. هناك العديد من لغات البرمجة التى يفهمها الكمبيوتر والـ Small Basic واحدة منها.

تخيل مناقشة بينك وبين صديقك. انت واصدقاؤك ستستخدمون كلمات فى هيئة جمل لنقل المعلومات بينكما. ايضا، لغات البرمجة تحتوى على مجموعة من الكلمات التى يمكن تنظيمها فى هيئة جمل لنقل المعلومات الى الكمبيوتر. والبرامج فى الاساس هى مجموعة من الجمل (احيانا تكون قليلة واحيانا اخرى يصل عددها الى الاف) التى معا تكون ذات معنى للمبرمج والكمبيوتر فحسب.

برامج الـ Small Basic

برنامج الـ Small Basic الامثل يحتوى على مجموعة من العبارات. كل سطر بالبرنامج يمثل عبارة وكل عبارة هى امر موجه للكمبيوتر. عندما نسال الكمبيوتر ان يقوم بتنفيذ برنامج سيأخذ الكمبيوتر البرنامج ويقرأ العبارة الاولى. يفهم ماذا نحاول ان نقول ثم يقوم بتنفيذ الامر. عندما ينتهى الكمبيوتر من تنفيذ العبارة الاولى، سوف يعود مرة اخرى الى البرنامج ويقرأ وينفذ السطر التالى. تستمر هذه العملية حتى نهاية البرنامج.

عودة الى برنامجنا الاول

هذا هو اول برنامج كتبناه.

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

هذا برنامج بسيط يحتوى على عبارة واحدة. هذه العبارة تقول للكمبيوتر ان يكتب سطر واحد من النص Hello World فى اطار النص.

```
Write Hello World
```

قد تكون لاحظت ان هذه العبارة يمكن ان تقسم الى اجزاء اصغر كما تقسم العبارات الى كلمات. العبارة الاولى الاولى مقسمة الى 3 اجزاء:

أ) TextWindow
ب) WriteLine
ج) "Hello World"

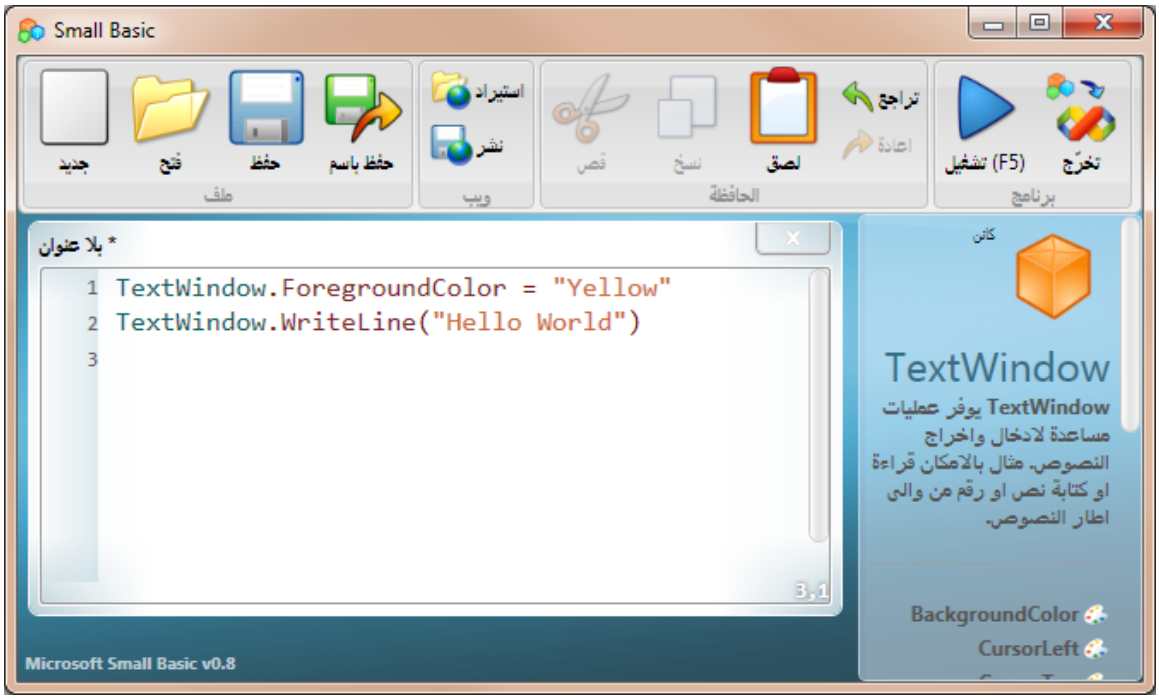
النقطة، الاقواس وعلامات الاقتباس كلها علامات ترقيم يجب وضعها فى المكان الملائم بالعبارة حتى يتفهم الكمبيوتر هدفنا.

قد تتذكر الاطار الاسود الذى ظهر عندما قمنا بتشغيل برنامجنا الاول. هذا الاطار الاسود يسمى بـــ Text Window وفى بعض الاحيان يسمى بـــ Console. هذا هو المكان الذى يظهر فيه نتيجة البرنامج.Text Window ،فى برنامجنا، تسمى بالكائن. هناك العديد من هذه الكائنات متاحة لنا لكى نستخدمها فى برامجنا. يمكن ان نقوم بالعديد من العمليات المختلفة على هذه الكائنات. لقد استخدمنا بالفعل العملية *WriteLine* فى برنامجنا. قد تكون لاحظت ايضا ان العملية *WriteLine* يليها **Hello World** بين علام اقتباس.تم تمرير هذا النص كمدخلات الى العملية WriteLine ، الذى يتم طباعته للمستخدم. يسمى هذا بمدخلات العملية. بعض العمليات تاخذ واحد او اكثر من المدخلات بينما لا تاخذ العمليات الاخرى اى مدخلات.

برنامجنا الثانى

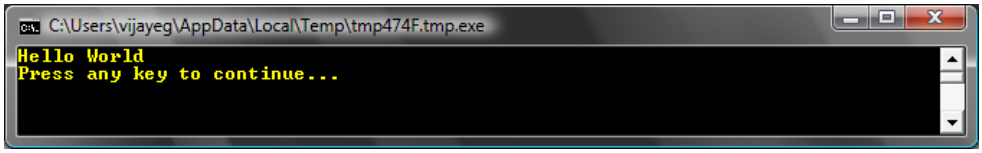
بما انك استوعبت برنامجنا الاول، هيا نجعله اجمل عن طريق اضافة بعض الالوان.

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



الشكل 5 – اضافة الالوان

عندما تقوم بتنفيذ البرنامج الموضح اعلاه، ستلاحظ انه يتم طباعة نفس النص"Hello World" داخل الـ TextWindow، ولكن هذه المرة يتم طباعته باللون الاصفر بدلا من اللون الرمادى مسبقا.



الشكل 6 - Hello World باللون الاصفر

لاحظ العبارة الجديدة التى قمنا باضافتها الى البرنامج الاصلى. يستخدم البرنامج هنا كلمة جديدة، *ForegroundColor* والتى تم معادلتها الى القيمة *"Yellow"*. هذا يعنى اننا قمنا بتعيين "Yellow" الى *ForegroundColor* . الآن، الفرق بين ForegroundColor والعملية WriteLine هو ان ForegroundColor لم تستخدم اى مدخلات او اى اقواس. بدلا من ذلك تم اتباعها بعلامة يساوى ثم كلمة. نحن نعرف ForegroundColor بانها خاصية لـ TextWindow . هذه قائمة بالقيم الصالحة لخاصية ForegroundColor. حاول استبدال "Yellow" بواحد من هذه القيم وراقب النتيجة – لا تنسى علامات الاقتباس، هى احد علامات الترقيم المطلوبة.

```
Black
Blue
Cyan
Gray
Green
Magenta
Red
```

```
White
Yellow
DarkBlue
DarkCyan
DarkGray
DarkGreen
DarkMagenta
DarkRed
DarkYellow
```

الفصل 3

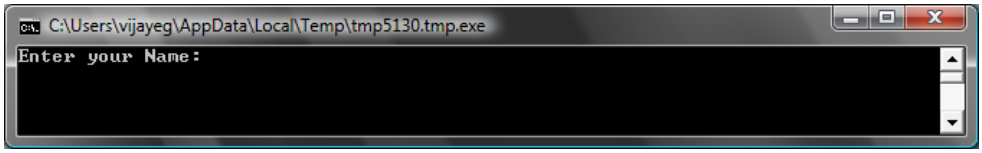
مقدمة فى المتغيرات

استخدام المتغيرات فى برنامجنا

اليس لطفيا لو ان برنامجنا يستطيع ان يقول "Hello" مقترنا باسم المستخدم بدلا من قوله "Hello" بصفة عام. حتى يمكننا فعل ذلك يجب ان نسال المستخدم عن اسمه/اسمها ثم نقوم بحفظه فى مكان ما ثم نقوم بطباعة "Hello" مقترنا باسم المستخدم. هيا نرى كيف نفعل ذلك:

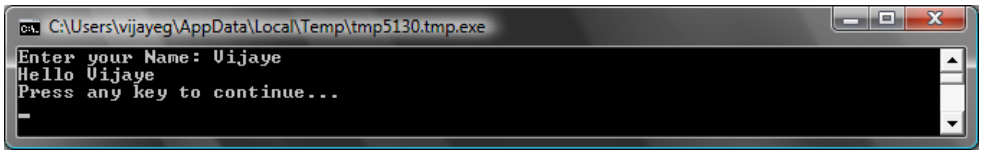
```
TextWindow.Write("Enter your Name: ")
name = TextWindow.Read()
TextWindow.WriteLine("Hello " + name)
```

عندما تقوم بكتابة وتنفيذ هذا البرنامج، سترى هذا الاخراج:



الشكل 7 – السؤال عن اسم المستخدم

و عندما تقوم بكتابة اسمك والضغط على زر ENTER، سترى هذا الاخراج:



الشكل 8 - ترحيب دافئ

اذا قمت بتشغيل البرنامج مرة اخرى، سيوجهلك نفس السؤال مرة اخرى. يمكنك كتابة اسم مختلف وفى هذه الحالة سيقول الكمبيوتر Hello مقترنا بهذا الاسم.

تحليل البرنامج

فى البرنامج الذى تم تشغيله، هذا هو السطر الذى قد يكون جذب انتباهك:

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

Read() مماثل لــ *WriteLine()*، ولكن بدون مدخلات. انها العملية التى تقول للكمبيوتر ان ينتظر المتسخدم حتى يقوم بكتابة شئ ما والضغط على مفتاح *Enter*. بمجرد ان يقوم المستخدم بالضغط على مفتاح *Enter*، تقوم هذه العملية باخذ ما كتبه المستخدم واعادته للبرنامج. النقطة المثيرة للانتباه هنا ان اى شيء كتبه المستخدم تم حفظه فى متغير يسمى *name*. يمكن تعريف المتغير بانه مكان لحفظ القيم بصورة مؤقتة لاستخدامها فيما بعد. فى السطر الموضح اعلاه، تم استخدام *name* لحفظ اسم المستخدم.

السطر التالى ايضا مثير للاهتمام:

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

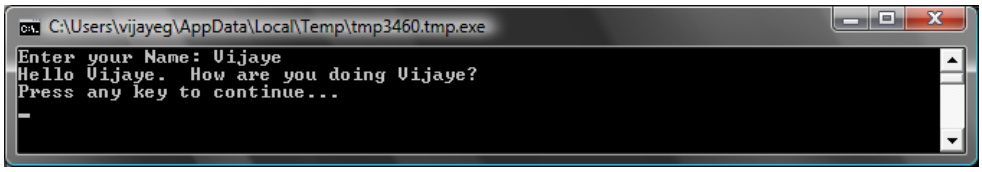
هذا هو المكان الذى ستقوم فيه باستخدام القيمة التى تم حفظها فى المتغير name. نأخذ القيمة الموجودة بــ name ونلحقها بــ "Hello" ونكتبها فى اطار النص.

Write، تماما كما WriteLine هى عملية اخرى فى ConsoleWindow. Write تتيح لك كتابة شئ ما الى ConsoleWindow ولكنه يسمح للنص التالى ان يكون على نفس سطر النص الحالي.

بمجرد ان يتم تعيين المتغير، يمكنك استخدامه اى عدد من المرات. على سبيل المثال، يمكنك عمل التالى:

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

و سترى هذا الاخراج:



الشكل 9 - اعادة استخدام المتغير

قواعد تسمية المتغيرات

المتغيرات لها اسماء مقترنة بها حتى يمكنك التعرف عليهم. هناك قواعد معينة سهلة وارشادات جيدة خاصة بتسمية المتغيرات كتالى:

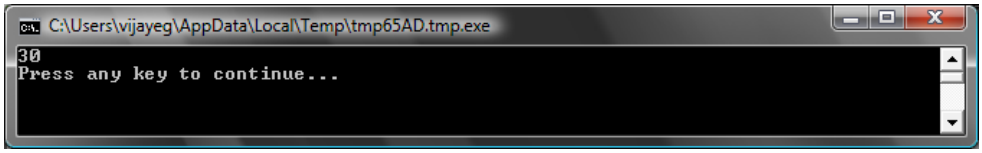
1. الاسم يجب ان يبدأ بحرف ويجب الا يكون أى من الكلمات الاساسية مثل if, for, then, etc.
2. الاسم يمكن ان يحتوى من اى مزيج من الحروف، الأرقام والشرط السفلية.
3. المتغيرات يمكن أن تكون طويلة كما تريد ، لذا من المفيد أن يكون اسم المتغير ذات معنى معبر
4. اسم المتغير يعبر عن طريقة استخدامه

اللعب بالارقام

لقد رأينا كيف يمكنك استخدام المتغيرات لحفظ اسم المستخدم. فى البرامج القليلة القادمة، سنرى كيف يتم حفظ والتعامل مع الارقام فى المتغيرات. هيا نبدأ ببرنامج بسيط:

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

عندما تقوم بتشغيل هذا البرنامج ستجد هذا الاخراج:



الشكل 10 – جمع رقمين

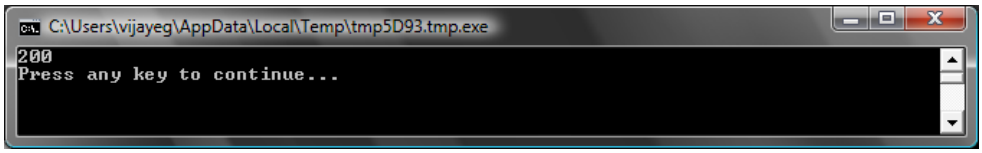
فى السطر الاول من البرنامج، ستقوم بتعيين القيمة 10 الى المتغير number1. وفى السطر الثانى، ستقوم بتعيين القيمة 20 الى المتغير number2. فى السطر الثالث، ستقوم بأضافة المتغير number1 وnumber2 وتعيين النتيجة الى المتغير number3. بالتالى، فى هذه الحالة المتغير number3 ستكون قيمته 30. وهذا ما تم طباعته فى اطار النص.

الآن، هيا نقوم بتعديل البرنامج قليلا وملاحظة النتائج:

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

سيقوم البرنامج الموضح اعلاه بضرب number1 فى number2 وحفظ النتيجة فى number3. ويمكنك ان ترى نتيجة هذا البرنامج ادناه:

لاحظ أنه لا يوجد علامات اقتباس حول الأرقام. وذلك لأن علامات الاقتباس ليست ضرورية للأرقام. تحتاج فقط الى علامات الاقتباس عندما تستخدم النص.



الشكل 11 – ضرب رقمين

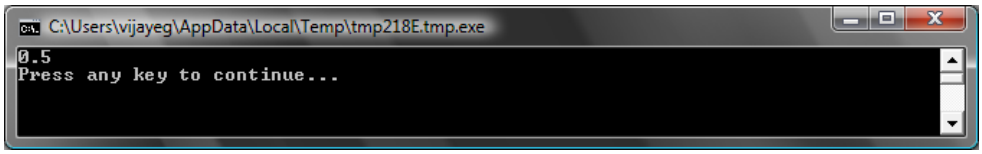
يمكنك ان تقوم بطرح او قسمة الارقام بنفس الطريقة. عملية الطرح ستكون كتالى:

```
number3 = number1 - number2
```

و علامة القسمة هى '/'. سيكون البرنامج كالتالى:

```
number3 = number1 / number2
```

و ستكون نتيجة البرنامج كالتالى:



الشكل 12 – قسمة رقمين

محول بسيط لدرجات الحرارة

سنقوم باستخدام الصيغة $°C = \frac{5(°F-32)}{9}$ فى البرنامج التالى لتحويل درجة حرارة فهرنهايت الى درجة حرارة مئوية.

أولا، سنقوم بالحصول على درجة الحرارة بالفهرنهايت من المستخدم وحفظها فى متغير. هناك عملية خاصة تسمح لنا بقراءة الارقام من المستخدم وهى **TextWindow.ReadNumber**.

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

بمجرد ان يتم حفظ درجة الحرارة بالفهرنهايت فى متغير، يمكننا تحويلها الى درجة حرارة مؤية كالتالى:

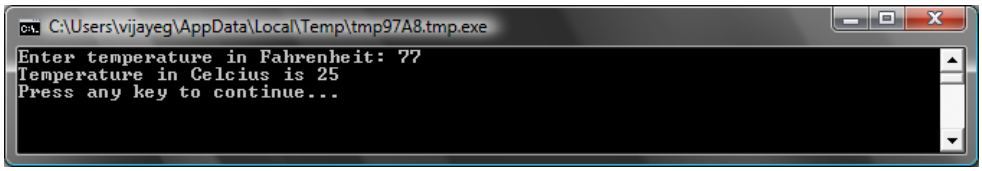
```
celsius = 5 * (fahr - 32) / 9
```

الاقواس تقول للكمبيوتر ان يقوم بحساب جزء fahr – 32 اولا ثم يقوم ببقية الحسابات. ما يجب علينا فعله الآن هو طباعة النتيجة للمستخدم. بوضع كل ذلك معا، يكمننا الحصول على البرنامج التالى:

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

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

و نتيجة هذا البرنامج ستكون:



الشكل 13 – محول درجة الحرارة

الفصل 4

الشروط والتفرعات

بالرجوع مرة اخرى الى برنامجنا الاول، اليس ظريفا لو قلنا Good Morning World او Good Evening World على حسب التوقيت اليومى بدلا من Hello World بصفة عامة؟ فى البرنامج التالى، سنجعل الكمبيوتر يقول Good Morning World اذا كان التوقيت قبل الساعة 12 ظهرا وGood Evening اذا كان التوقيت بعد الساعة 12 ظهرا.

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

سترى احدى الاخراجين على حسب التوقيت الذى تقوم فيه بتشغيل البرنامج:

الشكل 14 - Good Morning World

الشكل 15 - Good Evening World

هيا بنا نقوم بتحليل اول ثلاثة سطور من البرنامج. هذا السطر يقول للكمبيوتر اذا كانت الساعة اقل من 12 ظهرا اذن قم بطباعة "Good Morning World". الكلمات If, Then and Endif هى كلمات مخصوصة يفهمها الكمبيوتر عندما يتم تشغيل البرنامج. الكلمة If دائما يتبعها شرط الذى يعد فى هذه الحالة **(Clock.Hour < 12)**. تذكر ان الاقواس مهمة حتى يتفهم الكمبيوتر ماذا تريد. الشرط يتبعه **then** والعملية المراد تنفيذها. وبعد العملية يأتى **Endif** ليقول للكمبيوتر ان التنفيذ المشروط للعملية قد انتهى.

قد يكون هناك اكثر من عملية بين then وEnd if وسيقوم الكمبيوتر بتنفيذهم جميعا اذا كان الشرط صالح. على سبيل المثال، يمكنك كتابة التالى:

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

Else

قد تكون لاحظت ان الشرط مكرر بالبرنامج الموجود فى بداية هذا الفصل. قيمة Clock.Hour قد تكون اقل من 12 ام لا. لم نكن بحاجة لأستخدام الشرط الثانى. فى هذه الحالة يمكنك اختصار العبارتين if..then..endif فى عبارة واحدة عن طريق استخدام كلمة جديدة else. لو قمنا بأعادة كتابة البرنامج بأستخدام else، سيكون كالتالى:

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

هذا البرنامج سيفعل نفس الشىء الذى يفعله البرنامج الاخر، وهو ما يقودنا الى درس مهم جدا فى البرمجة:

هناك طرق متعددة لعمل نفس الشيء في البرمجة. احيانا قد تكون طريقة اكثر منطقية من الطرق الاخرى. الخيار متروك للمبرمج. سوف تتعلم هذه التقنيات المختلفة ومساوىء وعيوب كل منها مع كتابة برامج اكثر وزيادة خبرتك.

المسافة البادئة

في جميع الأمثلة يمكنك ان تلاحظ المسافة البادئة فى البيانات *If* و *Else* و *EndIf*. هذه السافات البادئة ليست ضرورية. وسيتمكن الكمبيوتر من فهم البرنامج على ما يرام بدونهم. ومع ذلك ، فإنها تساعدنا على رؤية وفهم هيكل البرنامج بطريقة أسهل. ومن ثم، تعتبر اضافة مسافة بادئة للبيانات هذا القبيل عادة جيدة.

فردى أم زوجى

هيا بنا نقوم بكتابة برنامج يبلغنا اذا كان الرقم الذى تم ادخاله فردى ام زوجى.

```
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
```

و عندما تقوم بتشغيل البرنامج سترى الاخراج التالى:

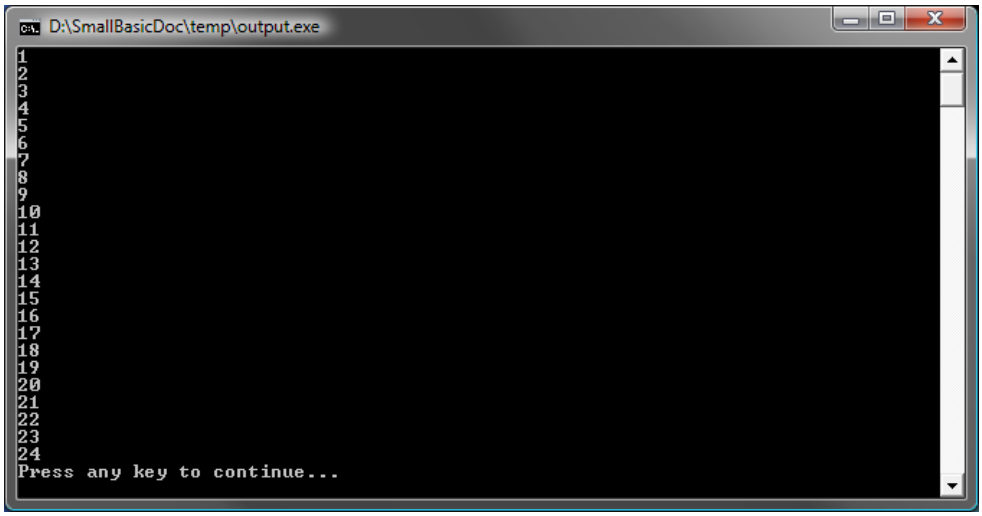
الشكل 16 - فردى ام زوجى

فى هذا البرنامج قدمنا لكم عملية اخرى جديدة ومفيدة **Math.Remainder**. و بالفعل كما قد تكون فهمت بالفعل ستقوم **Math.Remainder** بقسمة الرقم الاول على الرقم الثانى واعادة البقية.

التفريعات

تذكر لقد تعلمت فى الفصل الثانى ان الكمبيوتر يقوم بتنفيذ كل عبارة بالبرنامج على حدة بالترتيب من اعلى لاسفل. لكن هناك عبارة مخصوصة تجعل الكمبيوتر ينتقل لعبارة اخرى خارج الترتيب. هيا نلقى نظرة على البرنامج التالى:

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



الشكل 17 – استخدام Goto

فى البرنامج الموضح اعلاه تم تعيين القيمة 1 الى المتغير i . ثم قمنا بأضافة عبارة جديدة تنتهى بنقطتان (:)

```
start:
```

يسمى هذا بـ label. تعد Labels كالاشارات المرجعية التى يفهمها الكمبيوتر. يمكنك ان تسمى الاشارة المرجعية بأى اسم وان تضيف اى عدد من الاشارات المرجعية فى برنامجك طالما تم تسميتهم باسماء فريدة.as long as they are all uniquely named

عبارة اخرى مثيرة للاهتمام هنا هى:

```
i = i + 1
```

هذه العبارة تقول للكمبيوتر ان يقوم بأضافة 1 الى المتغير i وتعيين النتيجة الى المتغير i. بالتالى لو كانت قيمة i تساوى 1 قبل هذه العبارة، ستساوى 2 بعد تنفيذ هذه العبارة.

و اخيرا

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

هذا الجزء يقول للكمبيوتر اذا كانت قيمة i اقل من 25 يجب ان يبدأ بتنفيذ العبارات من الاشارة المرجعية start.

التشغيل اللانهائى

باستخدام عبارة **Goto** يمكن ان يقوم الكمبيوتر باعادة اى شيء اى عدد من المرات. على سبيل المثال يمكنك ان تأخذ البرنامج الذى قمنا بكتابته لابلاغنا اذا كان الرقم فردى او زوجى و تعديله كما هو موضح ادناه و سيتم تشغيل البرنامج الى ما لا نهاية. يمكنك ايقاف البرنامج بالنقر على زر اغلاق (X) الموجود فى الركن الايمن العلوى من الاطار.

```
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
```

الشكل 18 – فردى او زوجى بلا نهاية

الفصل 5

التكرار الحلقى

التكرار الحلقى For

لنأخذ البرنامج الذى قمنا بكتبته فى الفصل السابق.

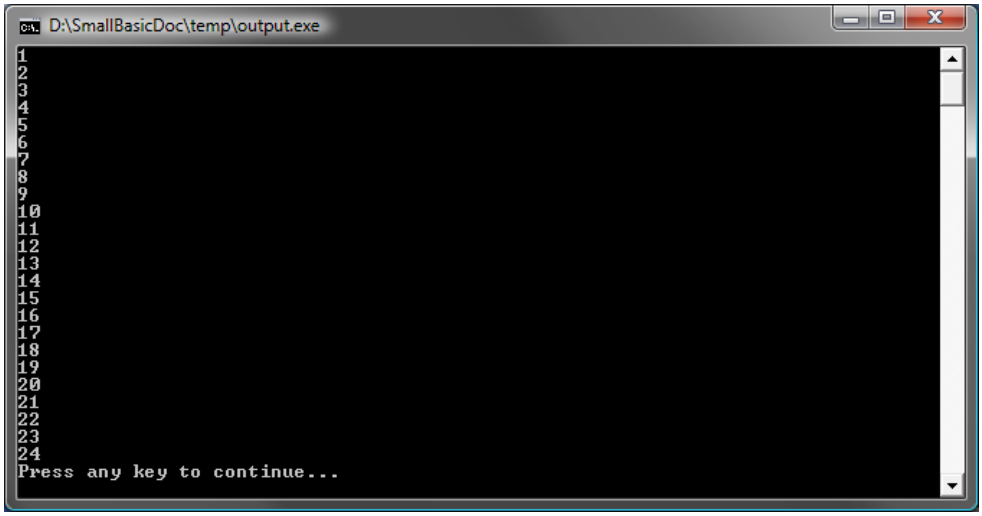
```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
  Goto start
EndIf
```

هذا البرنامج يطبع الارقام مرتبة من 1 الى 24. عملية تزايد المتغير شائعة جدا في مجال البرمجة لذا تقدم لغات البرمجة عادة طريقة أسهل للقيام بذلك.

البرنامج بالاعلى معادل لهذا البرنامج:

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

والاخراج هو:



الشكل 19 – استعمال التكرار الحلقى For

لاحظ اننا خفضنا عد سطور البرنامج من 8 الى 4، ومع ذلك يظل الاخراج مماثلا تماما للبرنامج ذو الـ 8 اسطر!
هل تذكر اننا قلنا سابقا انه توجد عدة طرق للقيام بنفس الشىء؟ هذا مثال جيد لذلك.

For..EndFor فى لغة البرمجة يسمى تكرار حلقى حيث انه يسمح باعطاء قيمة بدء وقيمة انهاء، ثم جعل الكمبيوتر يزيد المتغير لك. كل مرة يزيد فيها الكمبيوتر قيمة المتغير، يقوم ايضا بتشغيل البيانات الموجودة بين **For** و**EndFor**.

ولكن ان اردت ان يزاد المتغير بـ 2 بدلا من 1، مثلا لطباعة الارقام المفردة من 1 الى 24، عندها يمكنك استخدام التكرار الحلقى لعمل هذا ايضا.

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

الشكل 20 – الاعداد الفردية فقط

جزء الـ **Step 2** من بيان**For** يبلغ الكمبيوتر بزيادة قيمة المتغير بـ2 بدلا من 1 كالعادة. باستخدام **Step** بامكانك تحديد اى زيادة تريدها. حتى بامكانك تحديد رقم سلبى وجعل عد الكمبيوتر الى الوراء، كما فى المثال التالى:

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

الشكل 21 – العد الى الوراء

التكرار الحلقى While

التكرار الحلقى **While** هو وسيلة اخرى للتكرار، وتكون عادة مفيدة عندما لا يعرف عد مرات التكرارمسبقا. في حين أن تكرار **For** يعمل لعدد محدد مسبقا ، التكرار **While** يعمل حتى يصبح شرطا معينا هو الصحيح. فى المثال التالى، نقوم بقسمة عدد على 2 حتى يصير الناتج اصغر من 1.

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

الشكل 22 – تكرار القسمة الى النصف

فى البرنامج باعلى نقوم بتعيين القيمة 100 الى*number* ومن ثم نقوم بتشغيل التكرار الحلقى While ما دامت قيمة الرقم اكبر من 1. داخل التكرار الحلقى نقوم بطباع الرقم ثم قسمته على اثنان. وكما هو متوقع، احراج البرنامج هو عدة ارقام، كل منها هو نصف ما قبله.

يكون صعبا كتابة نفس البرنامج باستخدام التكرار الحلقى **For** لاننا لا نعرف كم مرة تتكرر الحلقة. باستخدام **While** يكون سهلا التحقيق من شرط ما ومن ثم سؤال الكمبيوتر اما التكملة او التكرار.

يكون من المثير للاهتمام أن نلاحظ ان كل تكرار حلقى **While** من الممكن ترجمته الى بيان **If..Then**. على سبيل المثال ، يمكننا إعادة كتابة البرنامج أعلاه على النحو التالي دون التأثير على النتيجة النهائية.

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

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

داخليا الكمبيوتر يعيد كتابة كل تكرار حلقى While كبيانات تستخدم If..Then مع واحد أو أكثر من بيانات Goto.

الفصل 6

بدء الرسومات

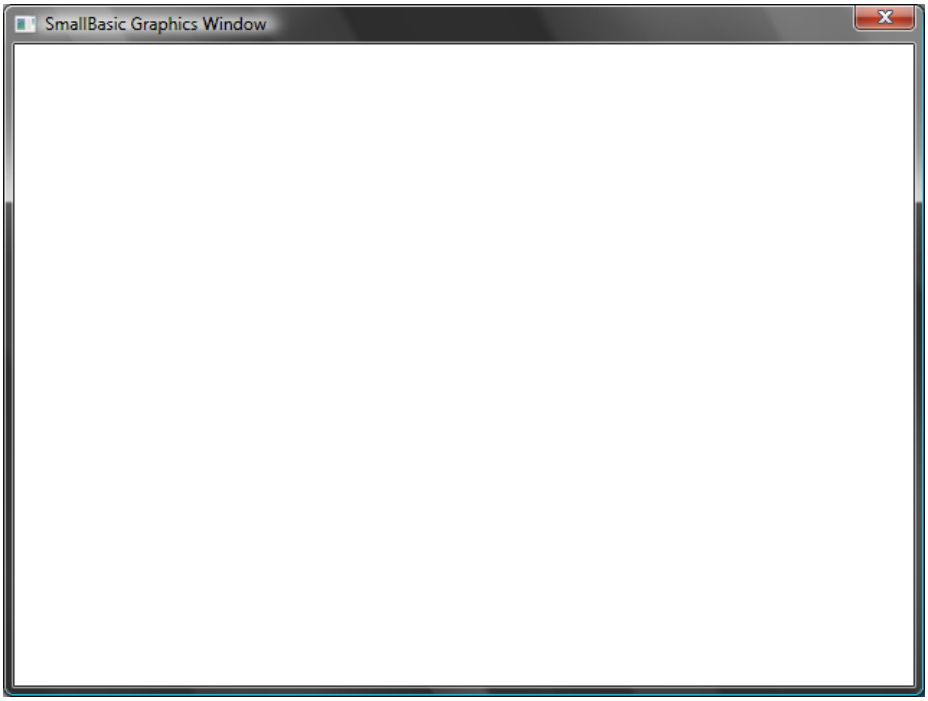
حتى الآن استخدمنا TextWindow فى جميع امثلتنا لشرح اسس لغة برمجة Small Basic . ولكن لغة Small Basic تحتوى ايضا على مجموعة قوية من قدرات التعامل مع الرسومات والتى سيتم شرحها فى هذا الفصل.

تقديـم GraphicsWindow

كما سمح لنا TextWindow بالتعامل مع الارقام والحروف، يوفر لنا Small Basic اطار GraphicsWindow والذى بالمكان استعماله للرسم. لنبدأ باظهار اطار الرسومات GraphicsWindow .

```
GraphicsWindow.Show()
```

عندما يتم تشغيل هذا البرنامج، ستلاحظ انه بدلا من الاطار الاسود المعتاد، ستحصل هذه المرة على اطار ابيض اللون كما هو مبين فى الاسفل. لا يمكن عمل الكثير فى هذا الاطار الآن، ولكن هذا هو الاطار الاساسى الذى سنعمل معه فى هذا الفصل. بامكانك اغلاق الاطار عن طريق الضغط على علامة "X" فى أعلى الزاوية اليمنى.



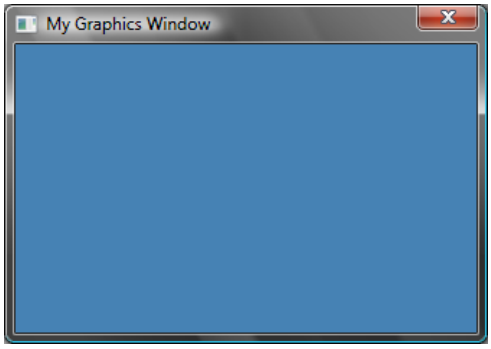
الشكل 23 – اطار رسومات فارغ

إنشاء اطار الرسومات

اطار الرسومات يسمح لك بتخصيص مظهره. يمكنك تغيير العنوان، والخلفية وحجمها. دعونا نمضي قدما وتعديله قليلا للتعرف على الاطار.

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

ترى بالاسفل كيف يبدو الاطار المخصص. يمكنك تغيير لون الخلفية إلى واحدة من القيم المدرجة في الملحق "ب". جرب هذه الخصائص لترى كيف يمكنك تغيير مظهر النافذة.

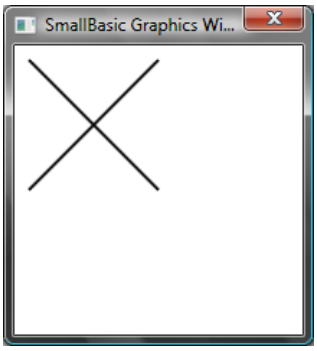


الشكل 24 - اطار رسومات مخصص

رسم الخطوط

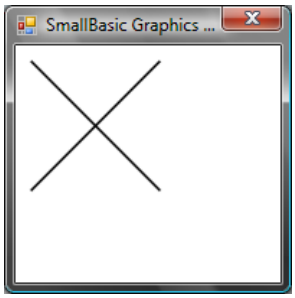
بعدما يبدأ GraphicsWindow، يمكننا رسم الأشكال والنصوص وحتى الصور عليه. لنبدأ برسم بعض الأشكال البسيطة. وفيما يلي برنامج يرسم خطان متقاطعان على اطار الرسومات.

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



الشكل 25 - تقاطع

أول سطرين من البرنامج يقومان باعداد الاطار والسطرين التاليان يرسما الخطوط المتقاطعة. اول رقمين بعد *DrawLine* يحددا احداثيات x و y للبدء والرقمين التاليين يحددا احداثيات نهاية الخط .والشيء المثير للاهتمام مع رسومات الكمبيوتر هو أن تنسيق (0 ، 0) يبدأ في أعلى الزاوية اليسرى من النافذة. فعليا، احداثيات الاطار فى الربع الثانى.

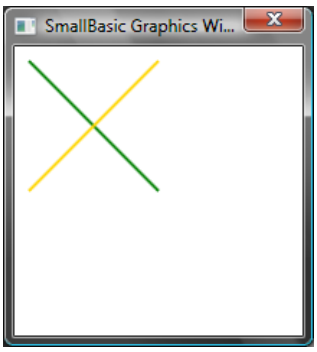


الشكل 26 - The co-ordinate map

وإذا عدنا إلى برنامج رسم الخطوط ، فمن المثير للاهتمام أن نلاحظ أن Small Basic يسمح لك تعديل خصائص الخط، مثل اللون وسمكه. أولا ، دعنا نغير لون الأسطر كما هو موضح في البرنامج أدناه.

بدلا من استخدام اسماء الالوان، بالامكان استخدام الطريقة المستعملة على الويب (#RRGGBB). مثال #FF0000 للون الاسود و#FF0000 يعنى الاحمر، و #FFFF00 للاصفر. سنتعلم المزيد عن الالوان فى الملحق "ب".

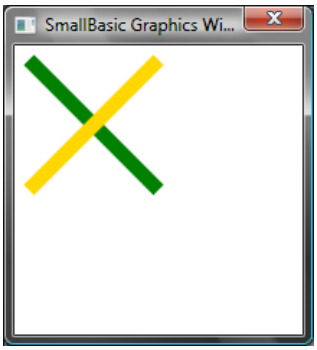
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



الشكل 27 - Changing Line Color

والآن ، لنقوم بتعديل حجم الخط ايضا. في البرنامج أدناه ، سنغير عرض الخط إلى 10 ، بدلا من العرض الافتراضي 1.

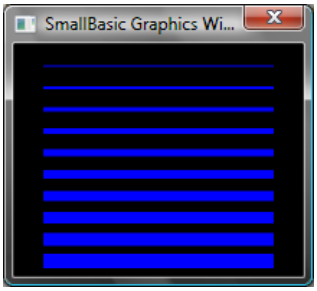
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenWidth = 10
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



الشكل 28 - Thick Colorful Lines

PenWidth و *PenColor* معا يعدلا خصائص القلم الذى يرسم هذه الخطوط. وهي لا تؤثر فقط على الخطوط ولكن أيضا أي شكل يتم رسمه بعدها.

```
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
```



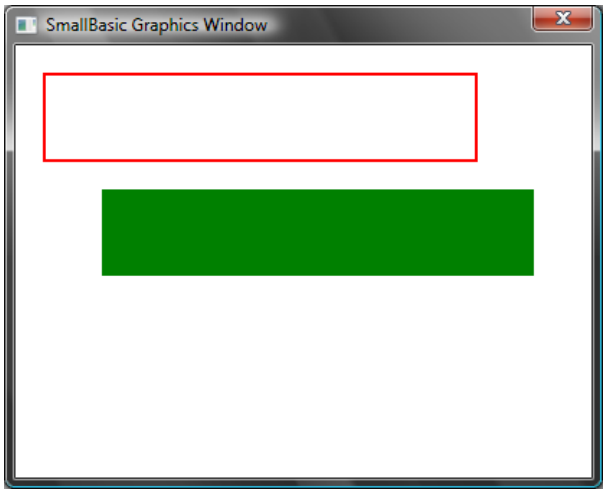
الشكل 29 - احجام متعددة للقلم

الجزء المثير للاهتمام فى هذا البرنامج هو التكرار الحلقى، فى كل تكرار نقوم بزيادة *PenWidth* (عرض القلم) وباستخدامه نقوم برسم خط جديد تحت آخر خط قمنا برسمه.

رسم وتعبئة الاشكال

عندما يتعلق الأمر برسم الأشكال، عادة يكون هناك نوعان من العمليات لكل شكل. عملية *Draw* (الرسم) وعملية *Fill* (التعبئة). عمليات الرسم تقوم برسم مخطط الشكل باستخدام القلم، وعمليات التعبئة تقوم بطلاء الشكل باستخدام فرشاة. على سبيل المثال في البرنامج أدناه ، هناك مستطيلان، واحد يتم رسمه باستخدام القلم الأحمر والآخر يتم طلائه باستخدام فرشاة خضراء.

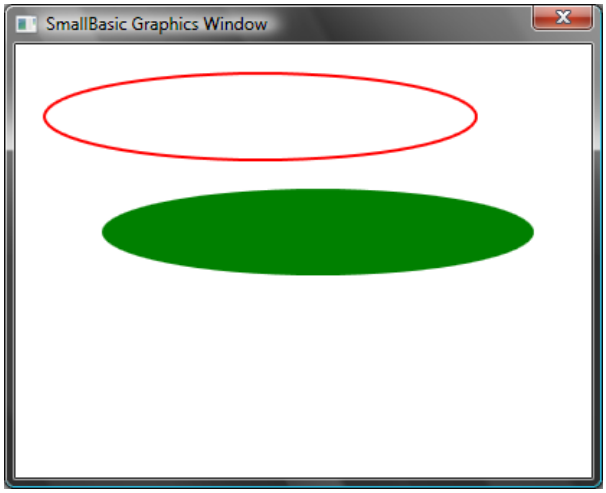
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

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

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



الشكل30 - الرسم والتعبئة

لرسم مستطيل أو لتعبئته، تحتاج إلى أربعة أرقام. أول رقمين يمثلا الاحداثيات x و y لأعلى الزاوية اليسرى من المستطيل. العدد الثالث يحدد عرض المستطيل بينما يحدد الرابع ارتفاعها. في الواقع ، ينطبق الشيء نفسه على رسم وتعبئة القطع الناقص كما هو موضح في البرنامج أدناه.

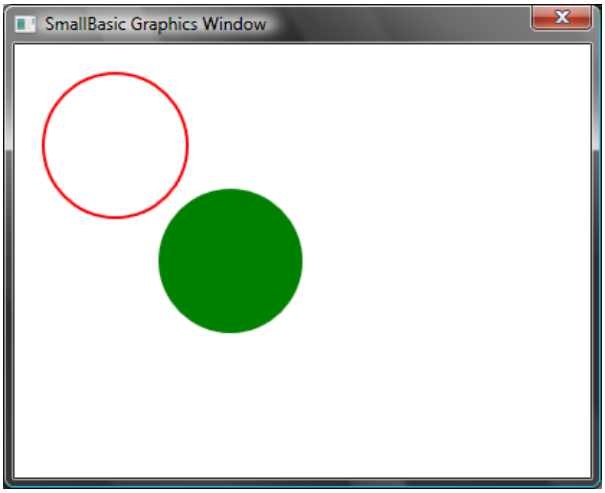
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

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

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



الشكل 31 - رسم وتعبئة Ellipses

القطع الناقص هو ايضا احد الدوائر. إذا كنت تريد أن ترسم دوائر، سيكون عليك تحديد نفس العرض والارتفاع.

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

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

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



الشكل 32 - دوائر

الفصل السابع

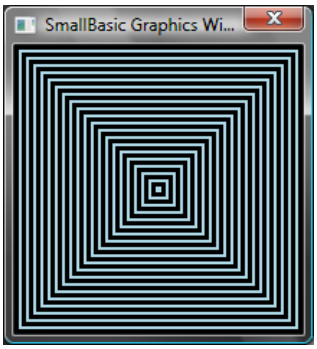
المرح مع الأشكال

فى هذا الفصل سنمرح قليلا باستخدام ما تعلمناه سابقا. يحتوي هذا الفصل على امثلة تجمع بين كل ما تعلمته حتى الآن لإنشاء بعض البرامج الشيقة.

Rectangalore

نقوم هنا برسم عدة مستطيلات فى تكرار حلقى مع تزايد الحجم.

```
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
```

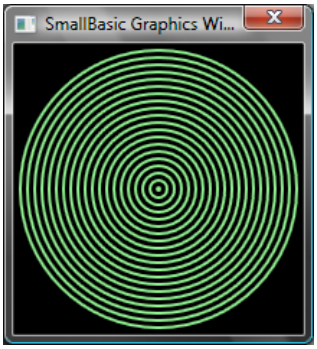


الشكل 33 - Rectangalore

Circtacular

هذا البرنامج بديل للبرنامج السابق، يقوم برسم دوائر بدلا من مربعات.

```
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
```



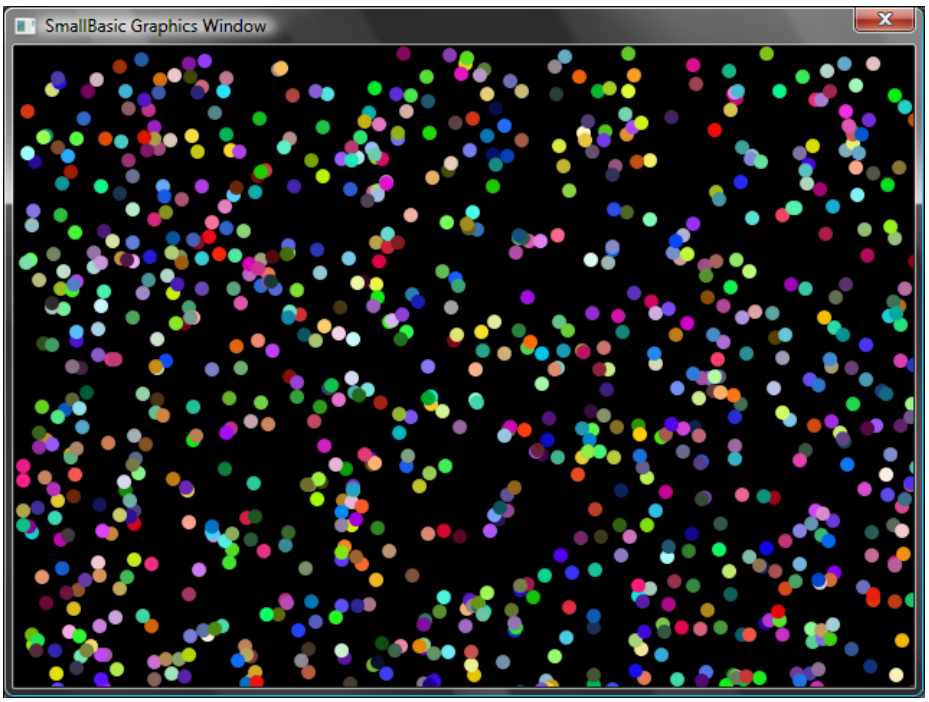
الشكل 34 - Circtacular

Randomize

هذا البرنامج يستخدم العملية GraphicsWindow.*GetRandomColor* لتعيين الوان عشوائية للفرشاة ثم يستخدم العملية Math.*GetRandomNumber* لتعيين الاحداثى *X* و *y* الخاص بالدوائر. يمكن دمج هاتين العمليتين بطرق شيقة من اجل خلق برامج شيقة تعطى نتائج مختلفة فى كل مرة يتم تشغيلها.

```
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
```



الشكل 35 - Randomize

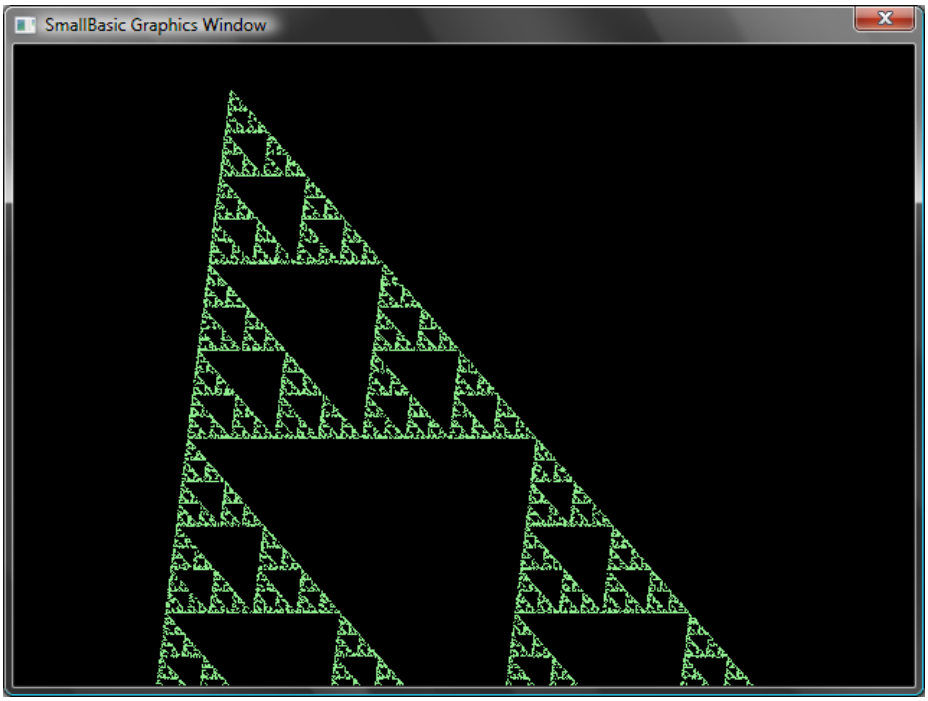
Fractals

يقوم البرنامج التالى برسم مثلث fractal مستخدما ارقام عشوائية.الـ fractal هو رسم هندسى يمكن تقسيمه الى عدة اجزاء، كل منها يمثل الشكل الام بدقة. فى هذه الحالة يقوم البرنامج برسم مئات المثلثات تمثل كل منها المثلث الام. و بما انه يتم تشغيل البرنامج لبضعة ثوان يمكنك ان ترى المثلثات و هى تتشكل ببطء من مجرد نقاط.

المنطق نفسه صعب شرحه سأتركه لك كتمرين للاستكشاف.

```
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
```



الشكل 36 - Triangle Fractal

ان كنت تريد ان ترى النقاط و هى تشكل الـ fractal ببطء يمكنك احداث تأخير فى التكرار الحلقى بأستخدام العملية **Program.***Delay*.
هذه العملية تأخذ رقم يقوم بتحديد مقدار التأخير بالمللى ثانية. هذا هو البرنامج المعدل مع السطر المعدل موضحا بالخط الغامق.

```
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
```

زيادة التأخير سيجعل البرنامج ابطأ. جرب ارقام مختلفة لمعرفة ما هو الافضل لذوقك.

تعديل اخر يمكنك عمله بالبرنامج هو استبدال السطر التالى:

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

بـهذا السطر:

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

بهذا التعديل سيقوم البرنامج برسم بكسل من المثلث بأستخدام أرقام عشوائية.

الفصل *8*

Turtle Graphics

لغة Logo

فى السبعينيات من القرن الماضى كانت هناك لغة برمجة بسيطة ولكن قوية فى نفس الوقت، تسمى بالـ Logo وكانت تستخدم من قبل عدد قليل من الباحثين. واستمر استخدامها حتى قام احد الاشخاص بأضافة ما يسمى "Turtle Graphics" (رسومات السلحفاة) الى لغة البرمجة و اتاحة "Turtle" (سلحفاة) التى كانت مرئية على الشاشة و استجابت لاوامر مثل *Move Forward* (امضى قدما) ، *Turn Right* (استدر يمينا)، *Turn Left* (استدر يسارا)، الخ. باستخدام الـLogo استطاع الناس رسم اشكال شيقة على الشاشة. كل ذلك كان من شأنه وهذا جعل لغة Logo سهلة على الفور وجذابة للناس من جميع الأعمار، وكانت مسؤولة الى حد كبير عن شعبيتها الكبيرة الثمانينات.

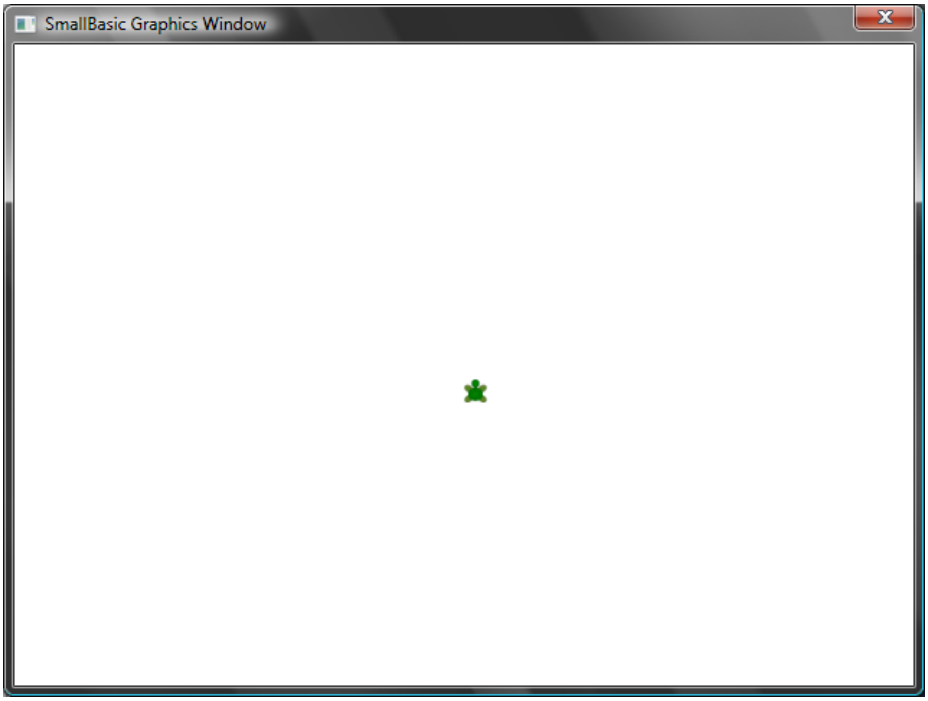
Small Basic توفر ايضا كائن **Turtle** (السلحفاة) مع العديد من الأوامر التي يمكن استدعاؤها من ضمن برامج Small Basic. في هذا الفصل ، سوف نستخدم الـ**Turtle** للرسم على الشاشة.

كائن Turtle

وبادئ ذي بدء ، نحن بحاجة إلى جعل السلحفاة (Turtle) مرئية على الشاشة. ويمكن تحقيق ذلك من خلال برنامج بسيط ذو سطر واحد.

```
Turtle.Show()
```

عند تشغيل هذا البرنامج ستلاحظ اطار ابيض، تماما مثل الذى رأيناه في الفصل السابق، باستثناء هذا الاطار لديه سلحفاة (Turtle) في المركز. هذه السلحفاة ستقوم باتباع تعليماتنا ورسم كل ما نطلبه منها.



الشكل 37 - السلحفاة ظاهرة فى الاطار

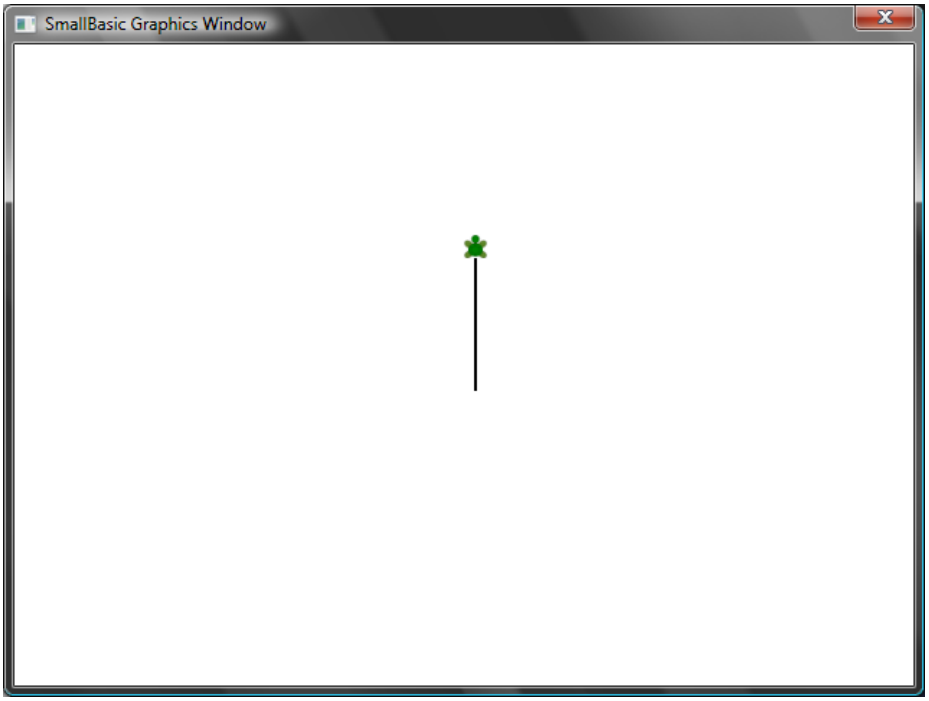
الرسم والتحريك

احد من التعليمات التى تفهما السلحفاة هو امر**Move** (تحريك).هذه العملية تأخذ رقم واحد وهذا الرقم يبلغ السلحفاة المسافة التى ستنتقلها. فى المثال التالى سنطلب من السلحفاة ان تتحرك 100 بكسل.

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

عند تشغيل هذا البرنامج ، يمكنك ان ترى في الواقع السلحفاة تتحرك ببطء 100 بكسل صعودا. واثناء تحركها يترى ايضا خط يُرسم ورائها. وعندما تتوقف السلحفاة عن الحركة سترى نتيجة مشابه للشكل التالى.

عن داستخدام عمليات السلحفاة (Turtle)، لا يجب بالضرورة استدعاء ()Show. The Turtle will السلحفاء تظهر تلقائيا عندما تقوم بتنفيذ احد العمليات.



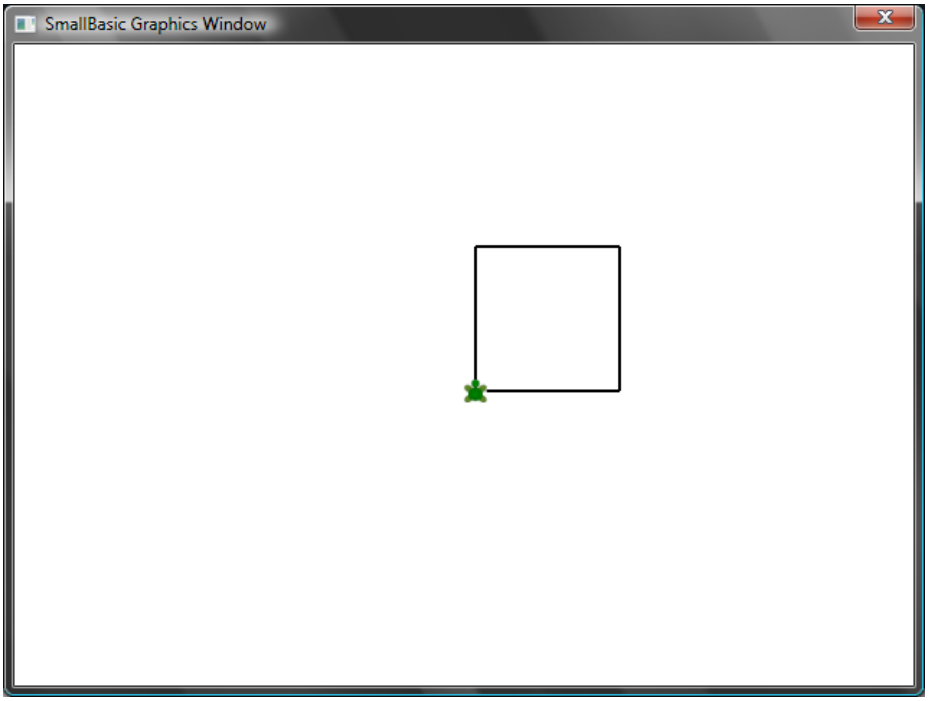
الشكل 38 - تحريك السلحفاة مئة بكسل

رسم مربع

المربع لديه أربع جوانب، وهما اثنان رأسي واثنان أفقي. من أجل رسم مربع نحن بحاجة إلى جعل السلحفاة ترسم خطا ، ثم تستدير يمينا وترسم خط آخر ، ويستمر ذلك حتى الانتهاء من جميع الجوانب الأربعة. إذا كان لنا أن يترجم هذا إلى برنامج ، سيكون كالتالى.

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

عند تشغيل هذا البرنامج ، يمكنك مشاهدة السلحفاة ترسم مربعا، سطر واحد في كل مرة، والنتيجة تبدو كالشكل أدناه.



الشكل 39 – السلحفاة ترسم مربعا

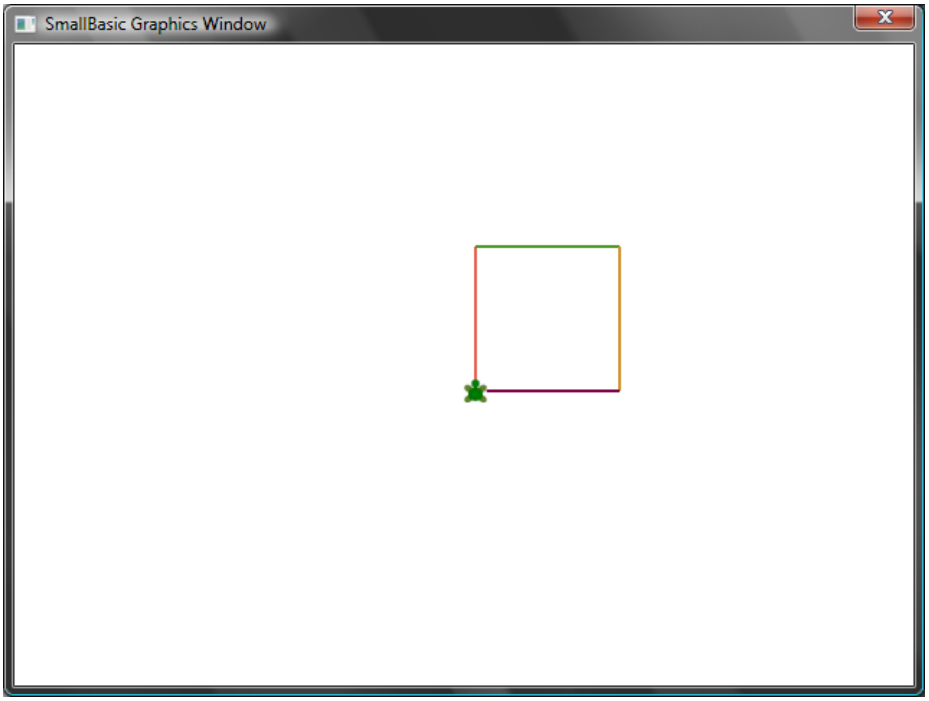
ومن المثير للاهتمام أن نلاحظ أنما نقوم بإصدار اثنين من نفس تعليمات مرارا وتكرارا، أربع مرات على وجه التحديد. لقد تعلمنا سابقا بالفعل أن هذه الأوامر المتكررة يمكن تنفيذها باستخدام التكرار الحلقى. مثلا إذا ما أخذنا البرنامج المذكور أعلاه وعدلناه ليستخدم التكرار الحلقى **For..EndFor** . في نهاية المطاف سنحصل على برنامج أبسط من ذلك بكثير.

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

تغيير الألوان

تقوم السلحفاة بالرسم فى نفس اطار الرسومات GraphicsWindow الذى استخدمناه فى الفصل السابق. وهذا يعني أن جميع العمليات التي تعلمناها في الفصل السابق ما زالت صالحة هنا. على سبيل المثال ، فإن البرنامج التالي يرسم مربعا ذو لون مختلف لكل جانب.

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



الشكل 40 – تغيير الالوان

رسم اشكال اكثر تعقيدا

بالاضافة الى عمليات **TurnRight** و **TurnLeft** ، كائن Turtle (السلحفاة) لديه عملية **Turn** (الانعطاف). باستخدام هذه العملية بالامكان رسم اى شكل مضلع. مثال، البرنامج التالى يرسم شكل مسدس (شكل ذو ستة جوانب).

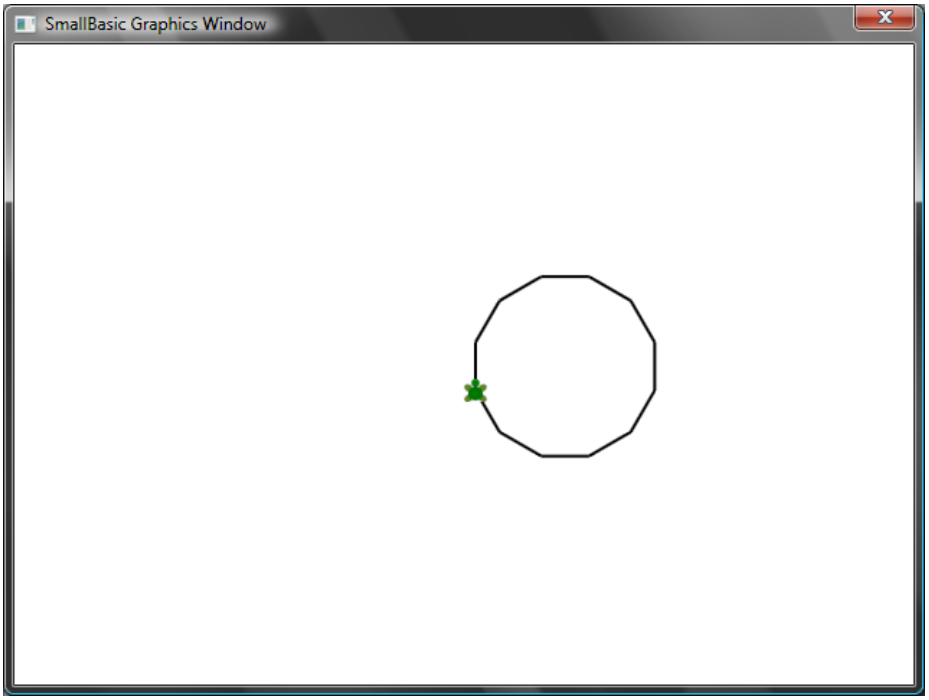
```
For i = 1 To 6
  Turtle.Move(100)
  Turtle.Turn(60)
EndFor
```

جرب هذا البرنامج ان كان يرسم حقا شكلا مسدسا. نلاحظ أنه بما ان الزاوية بين كل جانب هى 60 درجة، لذا نستخدم **Turn(60)** . هذا المضلع كل جوانبه متساوية فى الطول، لذا يمكن الحصول بسهولة على الزاوية بين كل جانب من خلال تقسيم 360 على عدد الجوانب.باستعمال هذه المعلوما وباستخدام المتغيرات، يمكن أن نكتب برنامجا عامة جدا لرسم أي شكل مضلع متساوى الجوانب.

```
sides = 12

length = 400 / sides
angle = 360 / sides

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

```
EndFor
```

باستخدام هذا البرنامج، يمكنك رسم أي مضلع فقط عن طريق تعديل المتغير **sides** . تعيين 4 لهذا المتغير سيمكننا من رسم المربع كما فى اول مثال. اما وضع قيمة كبيرة بما فيه الكفاية، مثلا 50 من شأنه أن يجعل الرسم تقريبا دائرى تماما.



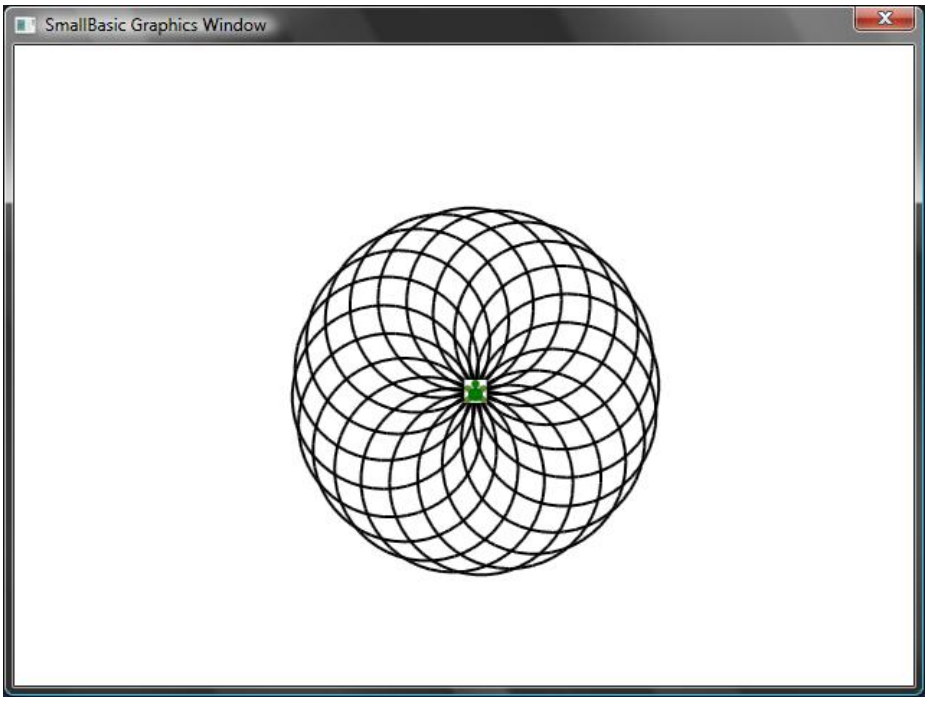
الشكل 41 – رسم مضلع ذو 12 جانب

باستخدام هذه التقنية يمكننا أن نجعل السلحفاة ترسم دوائر متعددة في كل مرة مع تحول قليل مما يؤدى إلى نتائج مثيرة للاهتمام.

```
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
```

البرنامج السابق ينفذ اثنين من التكرار الحلقى **For..EndFor**. واحد داخل الآخر. التكرار الداخلى (*i = 1 to sides*) مشابه لبرنامج المضلع ويقوم برسم دائرة. التكرار الخارجى (*j = 1 to 20*) مسئول عن ادارة السلحفاة قليلا لرسم كل دائرة. ويبلغ السلحفاة برسم 20 دائرة. عند وضع كل هذا معا، نحصل على نمط مثير للاهتمام، مثل الشكل التالى.

في البرنامج أعلاه، جعلنا السلحفاة ترسم بشكل أسرع من خلال تحديد السرعة إلى 9. يمكنك تعيين هذه الخاصية إلى أي قيمة بين 1 و 10 لجعل السلحفاة تتحرك بالسرعة التي تريدها .



الشكل 42 – التحرك فى دوائر

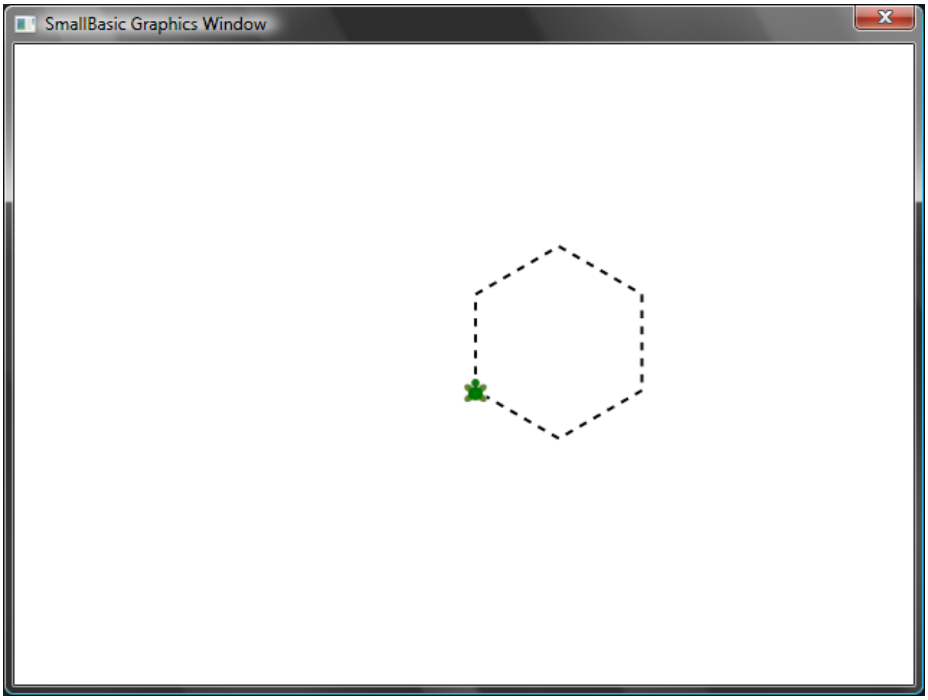
التحرك

بالامكان جعل السلحفاة تتوقف عن الرسم باستخدام العملية **PenUp** . هذا يسمح لك بنقل السلحفاة إلى أي مكان على الشاشة من دون رسم خط. استدعاء PenDown سيجعل سلحفاة ترسم مرة أخرى. ويمكن استخدام هذا للحصول على بعض المؤثرات المثيرة للاهتمام ، مثلا رسم خطوط المنقطة. وفيما يلي برنامج يستخدم هذا لرسم المضلع ذى جوانب منقطة.

```
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
```

مرة اخرى، يحتوى هذا البرنامج على تكرارين حلقيين. التكرار الداخلى يرسم خط منقط واحد، بينما يقوم التكرار الخارجى بتحديد عدد الخطوط التتى سيتم رسمها. فى هذا المثال قمنا بتحديد قيمة المتغير**sides** بـ 6، لذا قمنا بالحصول على مضلع شكله مسدس كما هو مبين بالاسفل.



الشكل 43 - استخدام PenUp و PenDown

Chapter 9

الروتينات الفرعية

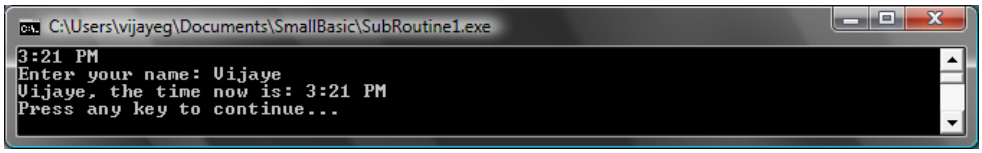
في كثير من الأحيان أثناء كتابة برامج ستظهر بعض الحالات التي سيتعين علينا فيها تشغيل نفس مجموعة من الخطوات ، مرارا وتكرارا. في هذه الحالات ، فإنه قد لا يكون له معنى لإعادة كتابة البيانات نفسها مرات عدة. وعند ذلك تكون *الروتينات الفرعية* (*Subroutines*) مفيدا.

الروتين الفرعى هو جزء من التعليات البرمجية فى برنامج أوسع عادة لفعل شيء محدد جدا ، ويمكن استدعاؤها من أي مكان في البرنامج. يتم تحديد *الروتينات* الفرعية من خلال الاسم الذي يلي الكلمة الرئيسية **Sub** وينتهى بكلمة **EndSub**. على سبيل المثال، المقتطف التالي يمثل روتين اسمه *PrintTime* ، و وظيفته طباعة في الوقت الحالي إلى TextWindow.

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

أدناه برنامج يتضمن الروتين الفرعى ويقوم باستدعائه من عدة أماكن.

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

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



الشكل 44 - استدعاء روتين فرعى بسيط

تقوم بتنفيذ روتين فرعى بواسطة استدعاء *SubroutineName()* (*SubroutineName* هو اسم الروتين الفرعى) وكما جرت العادة العلامات "()" ضرورية لتعريف الكمبيوترانك تريد تنفيذ روتين فرعى.

فوائد استخدام الروتينات الفرعية

كما رأينا للتو أعلاه ، الروتينات الفرعية تساعد على التقليل من كمية التعليمات البرمجية التى ستكتبها. بمجرد الانتهاء من كتابة روتين PrintTime ، يمكنك استدعائه من أي مكان في البرنامج وسوف يقوم بطباعة الوقت الحالي.

وبالإضافة إلى ذلك، الروتينات الفرعية يمكن أن تساعد على تحليل المشاكل المعقدة إلى قطع ابسط.مثال، ان كان لديك معادلة معقدة يمكنك كتابة عدة روتينات فرعية لحل قطع صغيرة من المعادلة المعقدة. ومن ثم يمكنك وضع النتائج معا للحصول على حل المعادلة المعقدة الأصلية.

تذكر، يمكنك استدعاء روتين SmallBasic من داخل البرنامج نفسه. لا يمكنك استدعاء روتين من ضمن برنامج آخر.

تساعد الروتينات الفرعية ايضا على تحسين سهولة قراءة البرنامج. اذا كان لديك اسماء جيدة للروتينات الفرعية للاجزاء التى تعمل كثيرا بالبرنامج، فان البرنامج يصير سهل القراءة والفهم. ويكون هذا مهما جدا إذا كنت تريد أن تفهم برنامج شخص آخر أو إذا كنت تريد برنامجك مفهوما من قبل الآخرين. وفي بعض الأحيان يكون حتى مفيدا عندما كنت تريد قراءة البرنامج الخاص بك مثلا بعد اسبوع من كتباتك له.

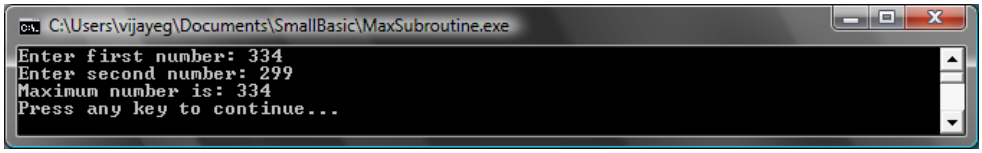
استخدام المتغيرات

يمكنك الوصول إلى واستخدام أي متغير لديك في برنامج من ضمن روتين فرعى. وكمثال على ذلك، يقبل البرنامج التالي رقمين ويطبع أكبرهما. لاحظ أن المتغير *max* يستخدم في داخل وخارج للروتين.

```
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
```

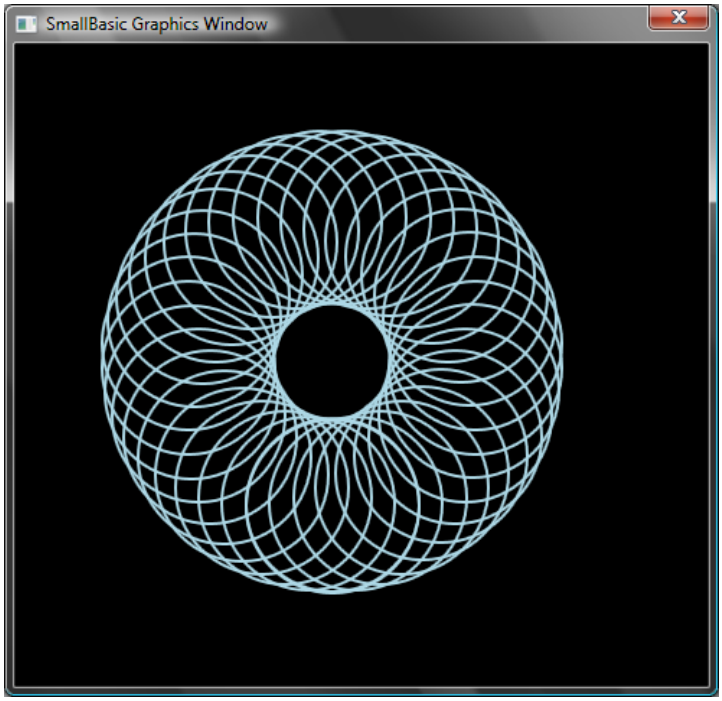
وناتج هذا البرنامج يشبه التالى.



الشكل 45 - اكبر رقمين باستخدام روتين فرعى

دعونا ننظر في مثال آخر من شأنه توضيح كيفية استخدام الروتينات الفرعية. هذه المرة سنقوم باستخدام برنامج رسومات يحسب نقاط مختلفة يتم تخزينها في متغيرات x و y . ثم يقوم باستدعاء روتين **DrawCircleUsingCenter** المسؤول عن رسم دائرة باستخدام x و y لمركز الدائرة.

```
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
```



الشكل 46 - مثال للروتينات الفرعية باستخدام الرسومات

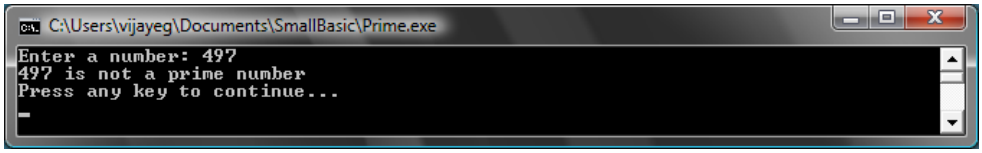
استدعاء روتين فرعى من داخل تكرار دائرى

أحيانا يتم استدعاء روتين فرعى من داخل تكرار دائرى، خلال ذلك الوقت يقومون بتنفيذ نفس البيانات ولكن مع قيم مختلفة لواحد أو أكثر من المتغيرات. على سبيل المثال، إذا كان لديك روتين يدعى **PrimeCheck** وهذا الروتين يحدد ان كان الرقم المعطى عدد أولى ام لا. يمكنك كتابة برنامج يتيح للمستخدم إدخال قيمة ويمكن القول بعد ذلك إذا كان عدد اولى ام لا ، وذلك باستخدام هذا الروتين. البرنامج أدناه يوضح ذلك.

```
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
```

روتين PrimeCheck يأخذ قيمة *i*، ويحاول ان يقسمه على أرقام اصغر. إذا قمنا بقسمة *i على رقم* ولم يوجد باقي فـ *i* ليس رقم اولى. عند هذه النقطة يقوم الروتين بتعيين قيمة *isPrime* الى *"False"* (خطأ) ويخرج. إذا كان الرقم غير قابل للقسمة على أعداد أصغر تبقى قيمة *isPrime* كما هى: *"True"* (صحيح).

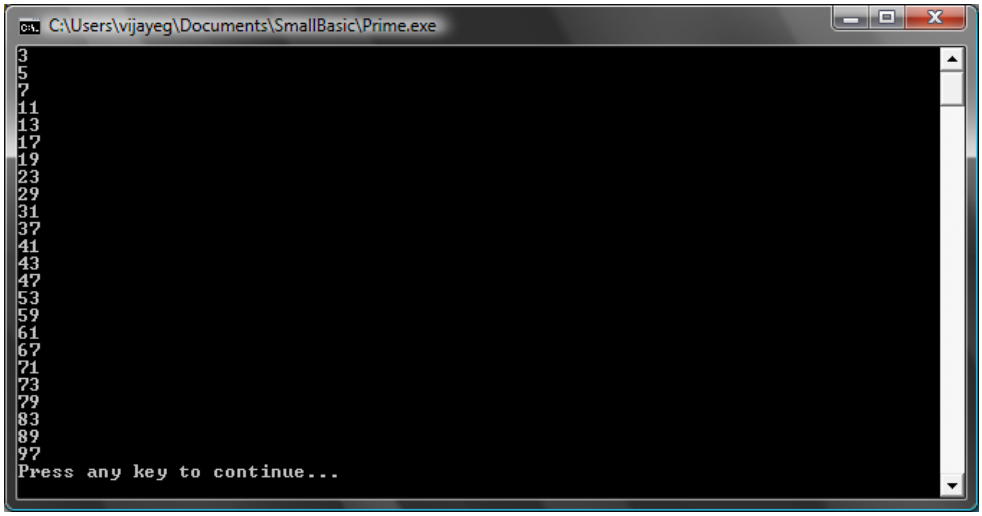


الشكل 47 - اختبار الارقام الاولية

الآن بما اننا لدينا روتين يقوم باختبار الارقام الاولية، بالامكان استخدامه لمعرفة جميع الارقام الاولية مثلا الاصغر من 100. يمكننا تعديل البرنامج السابق بسهولة لاستدعاء *PrimeCheck* داخل تكرار حلقى. عندها يقوم الروتين باستخدام قيمة مختلفة عند كل تكرار. لترى كيفية عمل ذلك فى المثال التالى.

```
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
```

فى البرنامج السابق يتم تعديل قيمة *i* مع كل تكرار. داخل التكرار الحلقى نقوم باستدعاء الروتين الفرعى *PrimeCheck*. يقوم هذا الروتين بعد ذلك بتحديد ان كانت قيمة *i* لرقم اولى ام لا. بعد ذلك تخزّن النتيجة فى المتغير *isPrime* والذى يتم الوصول اليه من خارج التكرار الحلقى. بعدها يتم طباعة قيمة *i* ان كانت لرقم اولى. وبما ان التكرار يتم من 3 الى 100، فيتم الحصول على جميع الارقام الوليم من 3 الى 100 ايضا. بالاسفل نتيجة البرنامج.



الشكل 48 - الاعداد الاولية

الفصل *10*

الصفائف

الآن يجب أن تكون على دراية جيدة باستخدام المتغيرات - بعد ان وصلنا إلى هذا الحد لا بد انك تقضى وقتا طيبا، أليس كذلك؟

دعونا نعيد النظر في أول برنامج كتبناه باستخدام المتغيرات:

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

في هذا البرنامج، تلقينا اسم المستخدم وتم تخزينه فى متغير يسمى **name**. في وقت لاحق قلنا "Hello" (مرحبا) للمستخدم. الآن ، دعونا نقول ان هناك أكثر من مستخدم واحد - مثلا 5 مستخدمين. كيف نقوم بتخزين جميع أسمائهم؟ احد الطرق للقيام بذلك هي :

```
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)
```

عند تشغيل هذا البرنامج نحصل على التالى:

الشكل 49 - بدون استخدام الصفائف

بوضوح يجب أن تكون هناك طريقة أفضل لكتابة مثل هذا البرنامج البسيط، أليس كذلك؟ وخاصة ان الكمبيوتر جيد حقا في ممارسة المهام المتكررة ، لماذا ينبغي علينا عناء مع كتابة التعليمات البرمجية نفسها مرارا وتكرارا لكل مستخدم جديد؟ الحيلة هنا هو تخزين واسترجاع أكثر من اسم مستخدم واحد بستخدام نفس المتغير. اذا كنا نستطيع القيام بذلك فيمكننا ايضا استخدام التكرار الحلقى **For** الذى تعلمناه في الفصول السابقة. هذا ما سوف تساعدنا فيه الصفائف.

ما هو الصفيف؟

الصفيف هو نوع خاص من المتغيرات التي يمكن أن تحمل أكثر من قيمة واحدة في وقت واحد. ما نعنيه هو أنه بدلا من الاضطرار إلى إنشاء **name1** و **name2** و **name3** و **name4** و **name5** لتخزين أسماء المستخدمين الخمسة ، بالامكان استخدام متغير **name** فقط لتخزين اسم اسم كافة المستخدمين الخمسة. الطريقة لتخزين قيم متعددة نقوم باستخدام ما يسمى بـ "الفهرس". مثلا، **name[1]** و **name[2]** و **name[3]** و **name[4]** و **name[5]** كل منها تستعمل لتخزين قيمة. الارقام 1 و2 و3 و4 و 5 تسمى "فهارس" الصفيف.

مع ان **name[1]** و **name[2]** و **name[3]** و **name[4]** و **name[5]** تبدو كلها كأنها تستعمل اكثر من متغير، فى الحقيقة هى كلها متغير واحد. وقد تسأل ما هى الفائدة من ذلك؟ افضل ميزة لتخزين القيم فى صفيف هو أنه يمكنك تحديد فهرس باستخدام متغير آخر بما يسمح لك بالوصول الى الصفائف بسهولة داخل تكرار دائرى.

والآن ، دعونا نرى كيف يمكننا وضع هذه المعلومات الجديدة لإعادة كتابة برنامجنا السابق باستخدام المصفوفات.

```
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("")
```

اسهل كثيرا، اليس كذلك؟ لاحظ السطرين بالخط العريض. الاول يخزن القيمة فى الصفيف والثانى يقرأ القيمة من الصفيف.
القيمة التى يتم تخزينها فى**name[2]** لن تتأثر بتلك المخزنة فى **name[1]**. لذا فى معظم الحالات يمكنك اعتبار **name[1]** و **name[2]** متغيران مختلفان ولكن بنفس الهوية.

الشكل 50 - استخدام الصفائف

البرنامج أعلاه يعطي النتيجة نفسها تقريبا واحد من دون المصفوفات ، باستثناء الفاصلة في نهاية *Mantis* . ويمكننا تصليح ذلك من خلال اعادة كتابة تكرار الطباعة:

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

فهرسة صفيف

في برنامجنا السابق رأيت كيف قمنا باستخدام أرقام وفهارس لتخزين واسترجاع القيم من الصفيف. ولكن لا تقتصر الفهارس على الأرقام فقط. يكون مفيدا جدا استخدام الفهارس النصية ايضا. على سبيل المثال ، في البرنامج التالي، نسأل ونخزن معلومات مختلفة عن المستخدم وبعد ذلك نقوم بطباعة المعلومات التى يطلبها المستخدم.

```
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])
```

الشكل 51 - استخدام فهارس غير رقمية

أكثر من بعد واحد

تريد تخزين اسم ورقم هاتف جميع اصدقائك ومن ثم تكون قادر على البحث عن أرقام الهواتف الخاصة بهم كلما كنت في حاجة - مثل دليل الهاتف. كيف تكتب مثل هذا البرنامج؟

في هذه الحالة ، هناك مجموعتين من الفهارس (المعروف أيضا باسم بعد الصفيف) المعنية. نفترض أننا نحدد كل صديق عن طريق اللقب. يصبح هذا فهرسنا الأول في الصفيف. وبمجرد استخدام الفهرس الأول للحصول على متغير صديقنا ، يصبح الفهرس الثانى هو **name** او **phone** ويساعدنا على الحصول على الاسم الكامل ورقم الهاتف لهذا الصديق.

طريقة تخزين هذه البيانات سيكون مثل هذا :

```
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"
```

لأن هناك فهرسين لنفس الصفيف **friends** ، يسمى هذا الصفيف: صفيف ثنائى الابعاد.

متى قمنا باعداد هذا البرنامج، بامكاننا استخدام اللقب فى الادخال وبعدها نقوم بطباة كل المعلومات المخزنة عنهم. وفيما يلي البرنامج الكامل الذى يفعل ذلك :

```
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"])
```

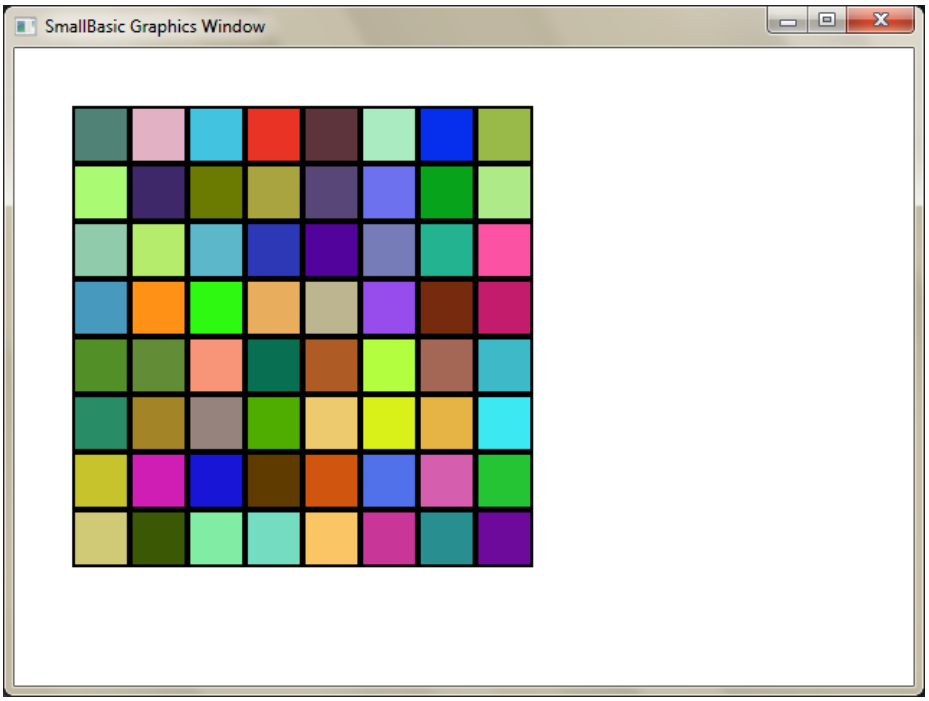
الشكل 52 - A simple phone book

استخدام الصفائف لتمثيل الشبكات

احد الاستخدانات الشائعة للصفائف المتعددة الابعاد هو تمثيل الشبكات والجداول. الجداول لها صفوف واعمدة لذا تناسب صفيف ثنائى الابعاد. وهنا برنامج بسيط يرتب صناديق داخل شبكة:

```
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
```

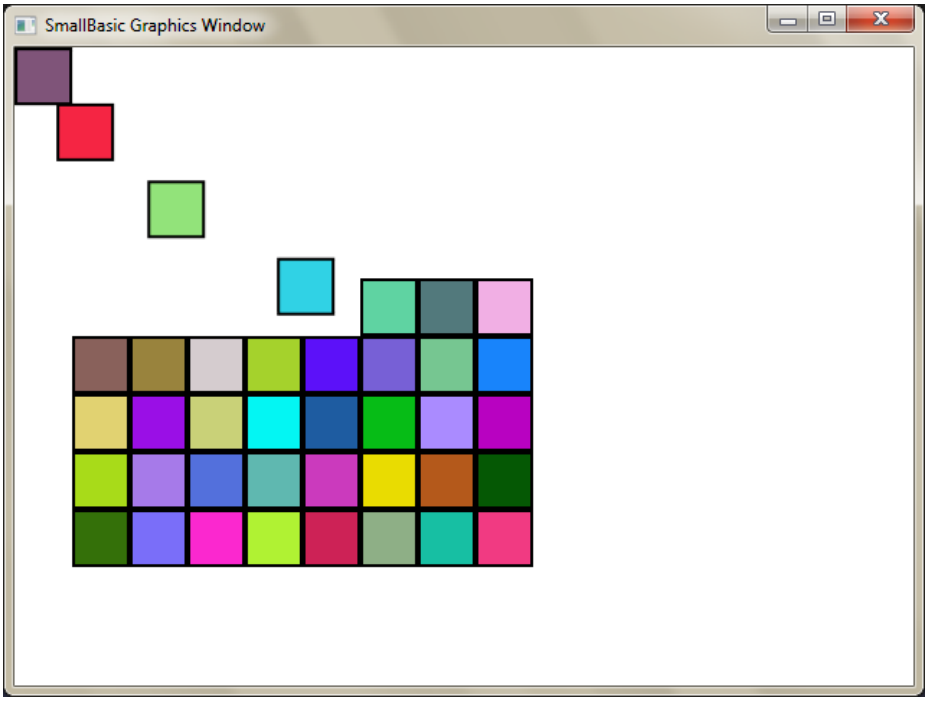
هذا البرنامج يضيف مستطيلات ويرتبها لتشكيل شبكة 8×8. بالاضافة الى وضع هذه الصناديق ، فإنه أيضا يقوم ببتخزين هذه الصناديق في صفيف. وذلك لتسهيل تتبع هذه الصناديق واستخدامها مرة أخرى عند الحاجة إليها.



الشكل 53 - ترتيب الصناديق فى شبكة

على سبيل المثال ، إضافة التعليمات البرمجية التالية لنهاية البرنامج السابق من شأنها أن تجعل هذه المربعات تتحرك لأعلى الزاوية اليسرى.

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



الشكل 54 - تتبع المربعات في الشبكة

الفصل *11*

الأحداث والتفاعل

في اول فصلين ، قدمنا الكائنات التي لديها خصائص وعمليات. بالإضافة إلى الخصائص والعمليات ، بعض الكائنات توفر ما يسمى بالأحداث (**Events**). الأحداث مثل الإشارات التي رُفعت، على سبيل المثال ، ردا على إجراءات المستخدم ، مثل تحريك الماوس أو الضغط عليه. بمعنى ما، الأحداث هى عكس العمليات. في حالة العملية، المبرمج هو يستدعيها لجعل الكمبيوتر القيام بشيء ، بينما في حالة الأحداث ، الكمبيوتر هو الذى يعرفك عندما يكون هناك شيء للاهتمام.

كيف تكون الأحداث مفيدة؟

الأحداث اساسية لإدخال التفاعل في البرنامج. إذا كنت تريد أن تتيح للمستخدم التفاعل مع البرنامج ، الأحداث هي التي ستستخدمها. مثلا لو كنت تكتب لعبة تيك تاك تو، سوف تريد ان تسمح للمستخدم أن يختار دوره، أليس كذلك؟ الاحداث تتيح لك ان تتلقى إدخال المستخدم من داخل البرنامج الخاص بك. وإذا كان هذا يبدو من الصعب فهم من اول وهلة، لا تقلق ، سنلقي نظرة على مثال بسيط جدا من شأنه أن يساعدك على فهم الأحداث، ما هي، وكيف يمكن استخدامها.

أدناه برنامج بسيط جدا لديه بيان واحد فقط وروتين فرعى واحد. الروتين يستخدم عملية ShowMessage على GraphicsWindow لكى يعرض مربع رسالة للمستخدم.

```
GraphicsWindow.MouseDown = OnMouseDown

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

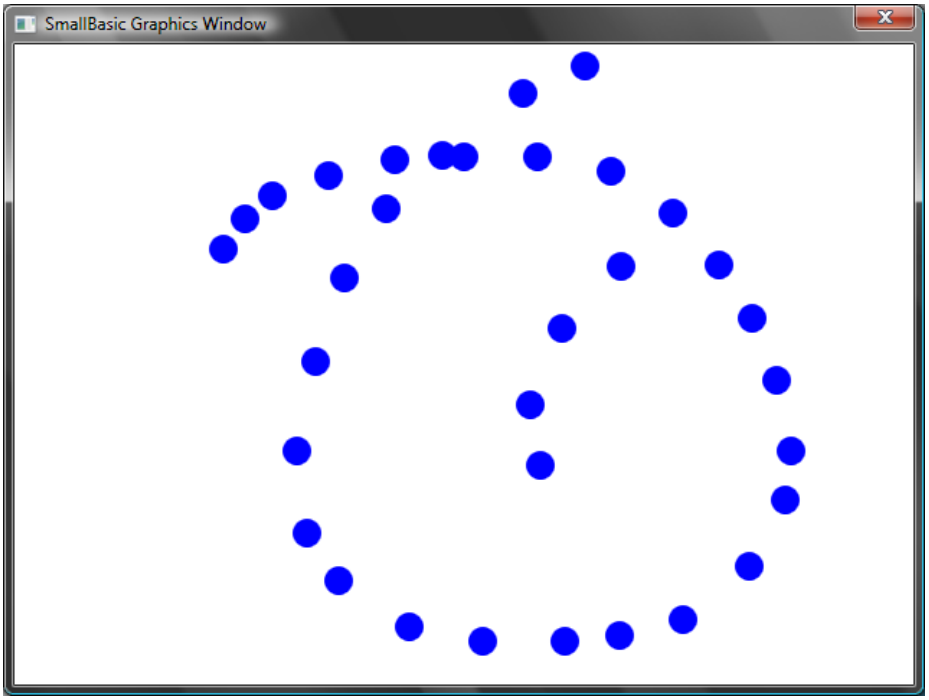
الجزء المثير للاهتمام في البرنامج أعلاه هو السطر الذي قمنا فيه بتعيين اسم روتين إلى الحدث **MouseDown** من كائن GraphicsWindow. ستلاحظ أن MouseDown يبدو كثيرا مثل خاصية - إلا أنه بدلا من تعيين قيمة ما، قمنا بتعيين روتين OnMouseDown إليها. هذا ما هو اهم ما فى الأحداث - عندما يحدث هذا الحدث ، يتم استدعاء الروتين تلقائيا. في هذه الحالة ، يتم استدعاء روتين OnMouseDown في كل مرة ينقر المستخدم باستخدام الماوس على GraphicsWindow. الآن جرب بنفسك تشغيل البرنامج. أي وقت تنقر زر الفأرة على GraphicsWindow ، سترى مربع رسالة على غرار هو مبين في الصورة ادناه.

الشكل 55 - الاستجابة لحدث

هذا الطريقة فى التعامل مع الاحداث تسمح بانشاء برامج خلاقة جدا ومثيرة للاهتمام. البرامج المكتوبة بهذه الطريقة غالبا ما تسمى برامج مستندة الى احداث (event-driven programming).

بامكانك تعديل روتين OnMouseDown لفعل أشياء أخرى غير اظهار مربع رسالة. على سبيل المثال ، كما في البرنامج أدناه ، يمكنك رسم نقاط زرقاء كبيرة فى الموقع الذى يقوم فيه المستخدم بالنقر على الماوس.

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

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



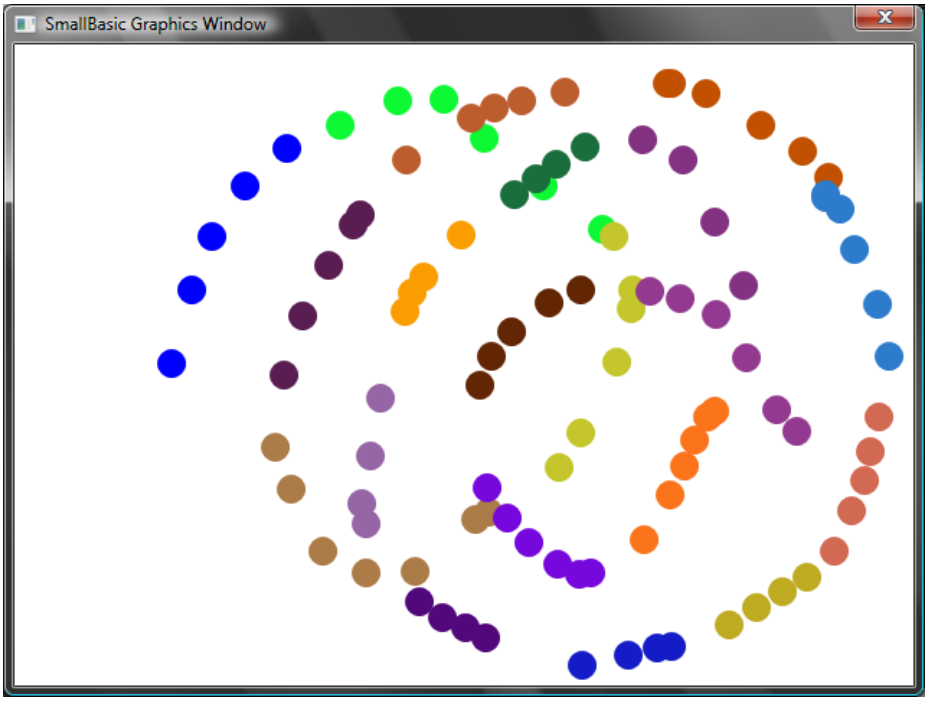
الشكل 56 – التعامل مع حدث MouseDown

لاحظ أنه في البرنامج أعلاه ،استخدمنا *MouseX* و *MouseY* للحصول على احداثيات الماوس. ثم استخدمنا هذه الاحداثيات كمركز الدائرة لرسمها.

التعامل مع أحداث متعددة

فى الحقيقة لا حدود لعدد الأحداث التي تريد التعامل معها. من الممكن حتى استخدام روتين فرعى واحد لمعالجة أحداث متعددة. ولكن يمكنك التعامل مع هذا الحدث مرة واحدة فقط. إذا حاولت تعيين اثنين من الروتينات الفرعية إلى الحدث نفسه، سيفوز الثاني.
لتوضيح هذا ، لنأخذ المثال السابق نضيف روتين فرع للتعامل مع الضغط على المفاتيح. أيضا ، لنجعل هذا الروتين الجديد يغير لون الفرشاة ، بحيث عند النقر بالماوس الخاص بك ، فإنك سوف تحصل على لون مختلف لكل نقطة.

```
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
```



الشكل 57 - التعامل مع أحداث متعددة

إذا قمت بتشغيل هذا البرنامج والنقر على الاطار ، ستحصل على نقطة زرقاء. الآن ، إذا ضغطت على أي مفتاح مرة واحدة وانقر مرة أخرى ، ستحصل على نقطة ملونة مختلفة. الذي يحدث عند الضغط على مفتاح هو أنه يتم تنفيذ روتين *OnKeyDown* الذي يغير لون الفرشاة إلى لون عشوائي. بعد ذلك عند النقر على الفأرة ، يتم رسم الدائرة باستخدام اللون الجديد الذى تم اختياره عشوائيا.

برنامج الرسم

بفهمنا للأحداث والروتينات الفرعية ، يمكننا الآن كتابة برنامج يتيح للمستخدمين الرسم على الاطار. فمن السهل جدا أن كتابة مثل هذا البرنامج اذا قمنا بتقسيم المشكلة إلى قطع اصغر. كخطوة أولى ، لنبدأ بكتابة برنامج يسمح للمستخدمين تحريك الماوس في أي مكان على اطار الرسومات، تاركا وراءه أثر أينما يتم تحريك الماوس.

```
GraphicsWindow.MouseMove = OnMouseMove

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

ولكن عند تشغيل هذا البرنامج ، الخط الأول يبدأ دائما من الحافة اليسرى العليا من الاطار عند الاحداثى (0 ، 0). يمكننا حل هذه المشكلة عن طريق التعامل مع الحدث *MouseDown* والتقاط قيمة *prevX* و *prevY* عندما يأتى الحدث.
أيضا، نحن حقا بحاجة فقط لرسم الخط عنما يكون زر الماوس مضغوطا باستمرار. من أجل الحصول على هذا، سوف نستخدم الخاصية

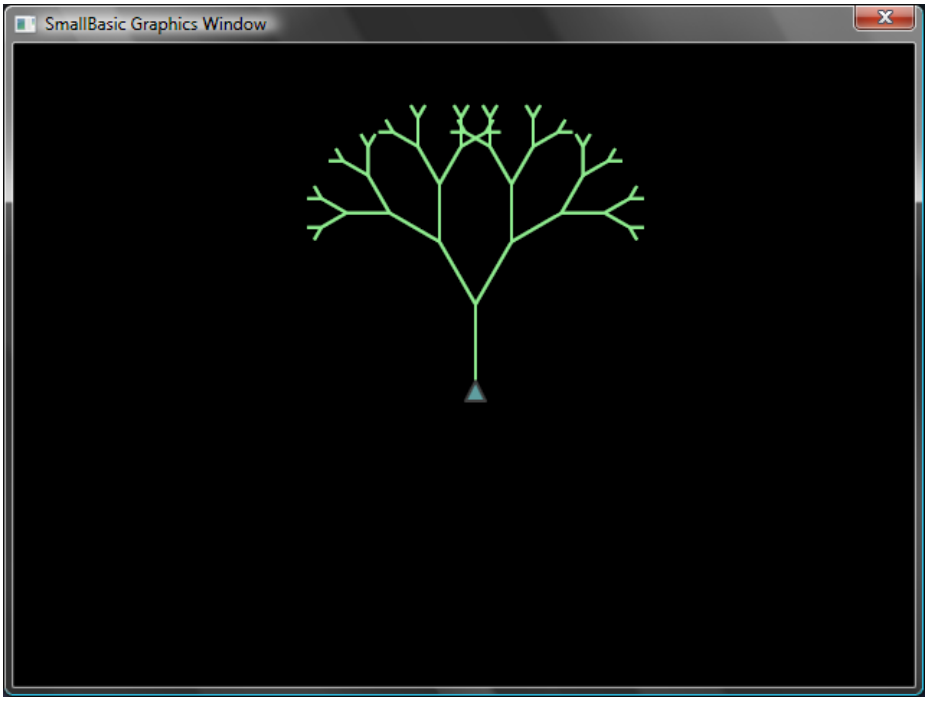
IsLeftButtonDown لكائن *Mouse*. هذه الخاصية تبلغنا ان كان الزر الايسر للماوس مضغوطا أم لا. إذا كانت هذه القيمة صحيحة سنقوم برسم خط ، إن لم يكن سنقوم بتخطي الخط.

```
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
```

ملحق "أ"

أمثلة مرحة

Fractal السلحفاة



الشكل 58 - Turtle ترسم شجرة 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
```

صور من Flickr

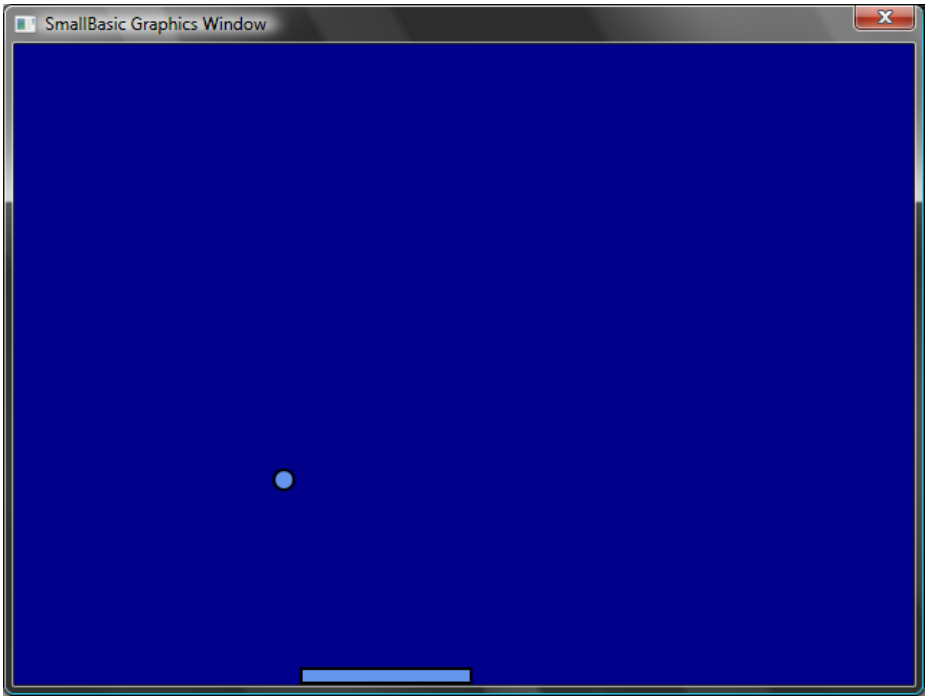
الشكل 59 - استرداد صور من Flickr

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

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

خلفية ديناميكية لسطح المكتب

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

لعبة المضرب



الشكل 60 - لعبة المضرب

```
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
```

ملحق "ب"

الألوان

هذه هى قائمة اسماء الألوان المستخدمة فى Small Basic، مصنفة حسب قاعدة اللون.

الألوان البرتقالية

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

الألوان الصفراء

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

الألوان الحمراء

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

الألوان الوردية

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

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

الألوان الزرقاء

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

Khaki	#F0E68C
DarkKhaki	#BDB76B

الألوان القرمزية

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

الألوان الخضراء

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

الألوان البيضاء

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

الألوان الرمادية

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

DeepSkyBlue	#00BFFF
DodgerBlue	#1E90FF
CornflowerBlue	#6495ED
MediumSlateBlue	#7B68EE
RoyalBlue	#4169E1
Blue	#0000FF
MediumBlue	#0000CD
DarkBlue	#00008B
Navy	#000080
MidnightBlue	#191970

الألوان البنية

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