

Microsoft Small Basic

Programlamaya Giriş

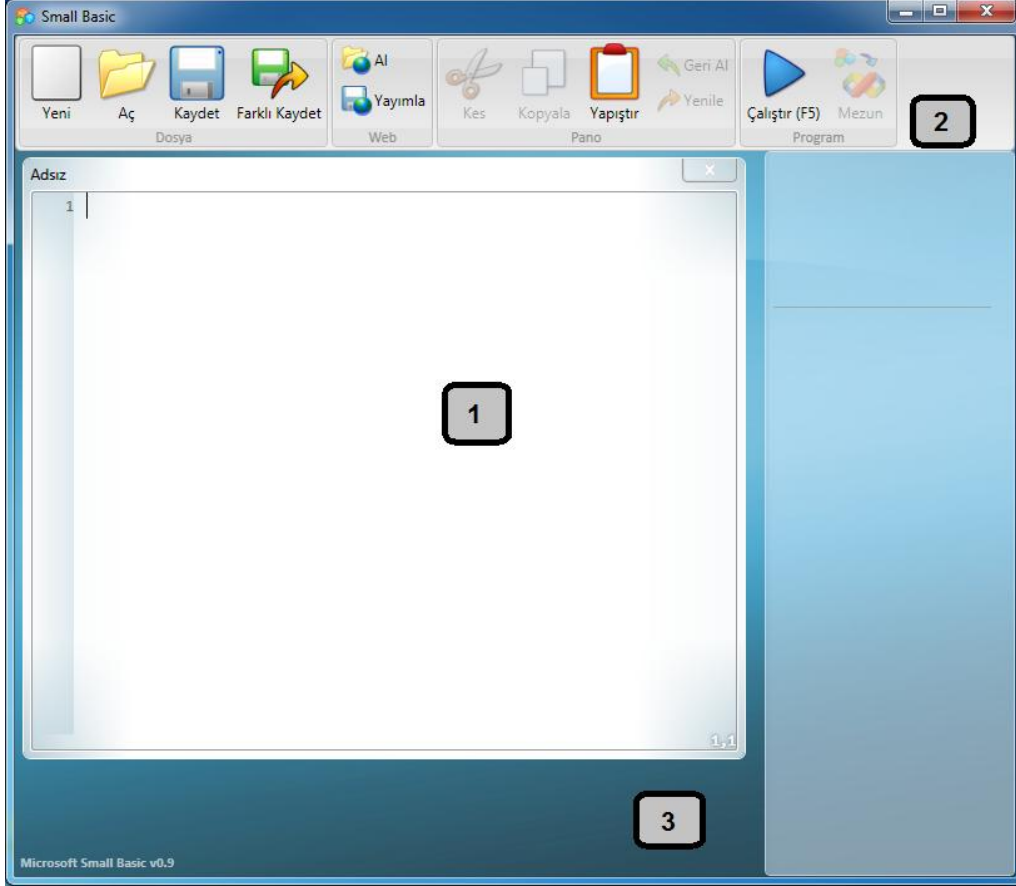
Small Basic ve Programlama

Bilgisayar Programlaması, programlama dilleri kullanılarak, bilgisayar yazılımlarının oluşturulması süreci olarak tanımlanır. Tıpkı bizim İngilizce'yi, İspanyolca'yı veya Fransızca'yı konuşup anlamamız gibi, bilgisayarlar da belirli dillerde yazılmış programları anlayabilirler. Bunlar programlama dilleri olarak adlandırılır. Başlangıçta, yalnızca birkaç tane programlama dili vardı ve bunların öğrenilmesi ve kavranması oldukça kolaydı. Ancak, bilgisayarlar ve yazılımlar giderek sofistike hale geldikçe, programlama dilleri de hızla gelişti ve daha karmaşık kavramları içerir hale geldi. Bunun sonucu olarak, şu anda çoğu programlama dilinin ve kavramlarının kavranması yeni başlayan birisi için oldukça zorlayıcı durumdadır. Bu da, insanların bilgisayar programlamasını öğrenme veya gerçekleştirmeye yönelik girişimlerinde cesaretlerini kırmaya başladı.

Small Basic, programlamayı yeni başlayanlar için son derece kolay, anlaşılır ve eğlenceli hale getirmek üzere tasarlanmış olan bir programlama dilidir. Small Basic'in amacı, engeli aşağıya çekmek ve şaşırtıcı bilgisayar programlaması dünyasına bir atlama taşı olarak görev yapmaktır.

Small Basic Ortamı

Small Basic Ortamını kısaca tanıtmakla işe başlayalım. SmallBasic'i ilk çalıştırdığınızda, aşağıdaki şekle benzeyen bir pencere göreceksiniz.



Şekil 1 – Small Basic Ortamı

Bu, Small Basic programlarımızı yazıp, çalıştıracığımız Small Basic Ortamıdır. Bu ortamın, sayılarla tanımlanmış çeşitli belirgin öğeleri vardır.

[1] ile tanımlanmış olan **Düzenleyici**, Small Basic programlarını yazacağımız yerdir. Örnek bir programı ya da daha önce kaydedilmiş olan bir programı açtığınızda, ekranda bu düzenleyici görüntülenecektir. Bu durumda, o programı değiştirebilir ve daha sonra kullanmak üzere kaydedebilirsiniz.

Ayrıca, bir kerede birden fazla programı açabilir ve üzerinde çalışabilirsiniz. Üzerinde çalıştığınız her bir program, ayrı bir düzenleyicide görüntülenecektir. O anda üzerinde çalıştığınız programı içeren düzenleyici, *aktif düzenleyici* olarak adlandırılır.

[2] ile tanımlanmış olan **Araç Çubuğu**, *aktif düzenleyiciye* veya ortama komut vermek için kullanılır. İlerledikçe, araç çubuğundaki çeşitli komutları öğreneceğiz.

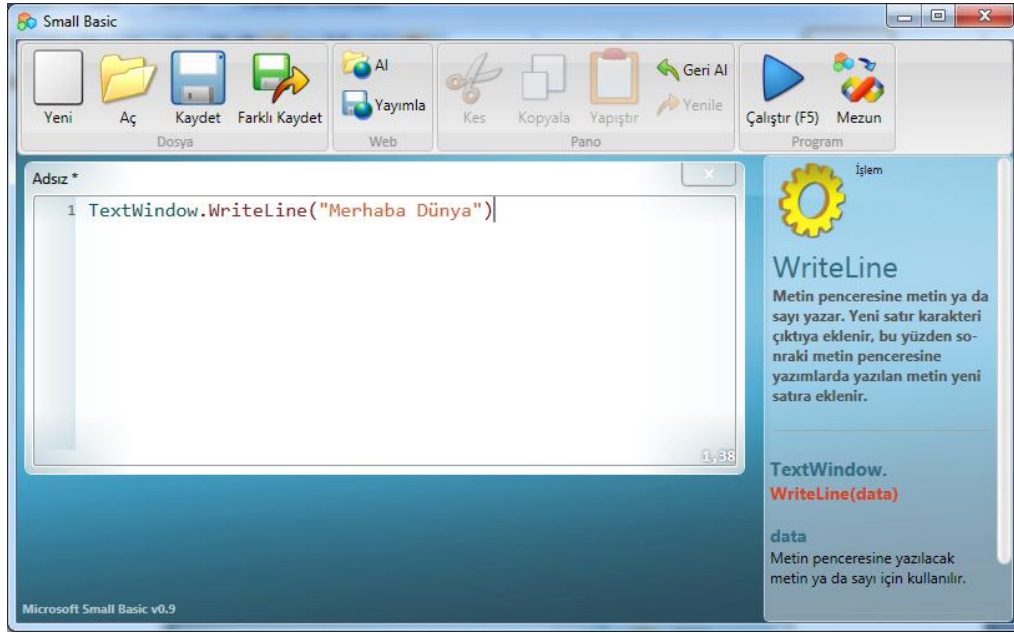
[3] ile tanımlanmış olan **Yüze**, tüm düzenleyici pencerelerinin gittiği yerdir.

İlk Programımız

Artık Small Basic Ortamı ile tanıştığınıza göre, onu kullanarak programlama yapmaya başlayacağız. Yukarıda söz ettiğimiz gibi, düzenleyici programlarımızı yazdığımız yerdir. Bunu yapmak için, önce aşağıdaki satırı düzenleyiciye yazın.

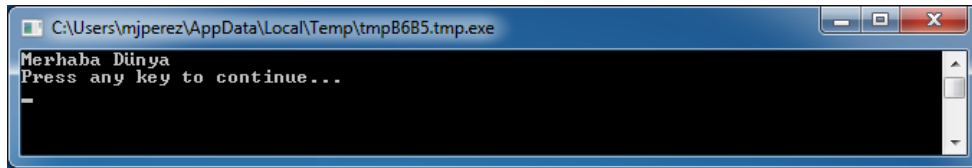
```
TextWindow.WriteLine("Merhaba Dünya")
```

Bu bizim ilk Small Basic programımız. Ve eğer bunu doğru yazdıysanız, aşağıdaki şekle benzer bir şey görüyor olmalısınız.



Şekil 2 – İlk Program

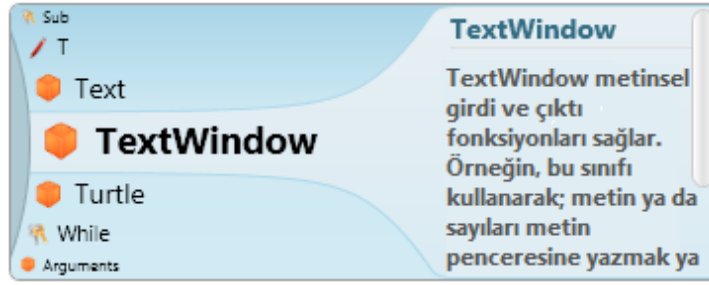
Yeni programımızı yazdığımıza göre, şimdi onu çalıştıralım ve neler olduğunu görelim. Programımızı araç çubuğu üzerindeki *Run* (Çalıştır) düğmesine basarak veya klavyenin üzerindeki F5 kısayol tuşunu kullanarak çalıştırabiliriz. Her şey yolunda giderse, programımızın aşağıdaki sonucu verecek şekilde çalışması beklenir.



Şekil 3 – İlk Program Çıktısı

Tebrikler! İlk Small Basic programınızı yazdınız ve çalıştırdınız. Bu oldukça küçük ve basit bir program, ancak yine de gerçek bir bilgisayar programcısı olma yolunda büyük bir adım! Şimdi, daha büyük programlar oluşturma konusuna geçmeden önce, söz etmemiz gereken bir detay daha var. Az önce neler olduğunu anlamamız gerekiyor – bilgisayara tam olarak ne dedik ve ne yapacağını nasıl bildi? Bir sonraki bölümde, bunu anlayabilmek için, yazdığımız programı analiz edeceğiz.

İlk programınızı yazarken, içinde bir öğeler listesi bulunan bir açılır pencerenin (Şekil 4) çıktığını fark etmiş olabilirsiniz. Bu, “akıllı algılama” olarak adlandırılır ve programınız daha hızlı yazmanıza yardımcı olur. Yukarı/Aşağı ok tuşlarına basarak listenin bir ucundan diğerine gidebilirsiniz ve istediğiniz şeyi bulduğunuzda, seçtiğiniz öğeyi programınıza eklemek için Enter tuşuna basabilirsiniz.



Şekil 4 – Akıllı Algılama

Programımızı kaydetmek

Small Basic'i kapatmak ve yazdığınız program üzerinde daha sonra tekrar çalışmak istiyorsanız, programı kaydedebilirsiniz. Aslında bu, kazara kapanma veya güç kesintisi durumunda bilgilerinizi kaybetmemeniz için, programları zaman zaman kaydetmek açısından iyi bir pratiktir. Üzerinde çalıştığınız programı, araç çubuğu üzerindeki “save” (kaydet) ikonuna basarak veya “Ctrl+S” (Ctrl tuşuna basılı tutarken, S tuşuna basın) kısayol tuşunu kullanarak kaydedebiliriz.

İlk Programımızı Anlamak

Bir bilgisayar programı aslında nedir?

Bir program, bilgisayar için bir talimatlar dizisidir. Bu talimatlar bilgisayara tam olarak ne yapacağını söyler ve bilgisayar da daima bu talimatları izler. Tıpkı insanlar gibi, bilgisayarlar da talimatları ancak anlayabilecekleri bir dilde verilirse izleyebilirler. Bunlar programlama dilleri olarak adlandırılır. Bilgisayarın anlayabileceği pek çok dil vardır ve **Small Basic** de bunlardan birisidir.

Sizinle arkadaşınız arasında bir konuşma geçtiğini hayal edin. Siz ve arkadaşlarınız, bilgileri birbirinize iletmek için, cümleler şeklinde organize olmuş kelimeler kullanırsınız. Benzer şekilde, programlama dilleri de bilgileri bilgisayara ileten cümleler şeklinde organize edilebilen kelime toplulukları içerirler. Ve, programlar temel olarak toplu halde programcıya ve bilgisayara bir anlam ifade eden cümleler topluluğudur (bunlar bazen yalnızca birkaç tane, bazen de binlerce olabilir).

Small Basic Programları

Tipik bir Small Basic programı, bir dizi *ifadeden* oluşur. Programın her bir satırı bir ifadeyi temsil eder ve her bir ifade de bilgisayar için bir talimattır. Bilgisayardan bir Small Basic programını uygulamasını istediğimizde, bilgisayar programı alır ve ilk ifadeyi okur. Söylemeye çalıştığımız şeyi anlar ve sonra da talimatımızı uygular. İlk ifademizi uygulamayı tamamladığında, programa geri dönüp ikinci satırı okur ve uygular. Programın sonuna ulaşıncaya kadar da bunu yapmaya devam eder. İşte bu noktada, programımız biter.

Bilgisayarın anlayabileceği pek çok dil vardır. Java, C++, Python, VB, vs. dillerinin tümü, basitten karmaşık yazılım programlarına kadar çeşitli programlar geliştirmek için kullanılabilir, güçlü modern bilgisayar dilleridir.

İlk Programımıza Geri Dönelim

İşte yazdığımız ilk program:

```
TextWindow.WriteLine("Merhaba Dünya")
```

Bu, tek bir *ifadeden* oluşan oldukça basit bir program. Bu ifade, bilgisayara **Merhaba Dünya** metnini Metin Penceresine yazmasını söylüyor.

Bu, gerçek anlamda bilgisayarın zihninde şöyle çevriliyor:

```
Write Merhaba Dünya
```

Cümlelerin kelimelere bölünebilmesi gibi, ifadelerin de daha küçük bölümlere ayrılabilceğini fark etmiş olabilirsiniz. İlk ifadede, 3 ayrı bölüm bulunuyor:

- TextWindow
- WriteLine
- "Merhaba Dünya"

Nokta, parantezler ve tırnak işaretlerinin tümü, bilgisayarın niyetimizi anlaması için, ifadede doğru yerlere yerleştirilmesi gereken noktalama işaretleri.

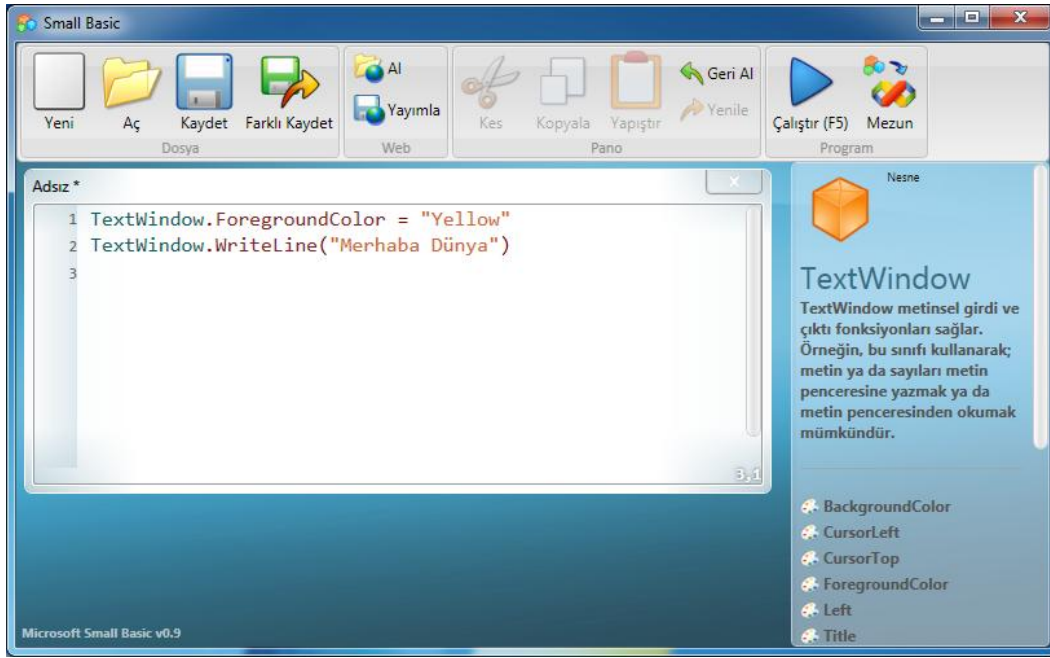
İlk programımızı çalıştırdığımızda çıkan siyah pencereyi hatırlayacaksınız. Siyah pencere, TextWindow veya bazen Konsol olarak adlandırılır. Bu, bu programın sonucunun gideceği yerdir. Bizim programımızda **TextWindow**, bir *nesne* olarak adlandırılır. Programlarımızda kullanılmak üzere, bu tip çeşitli nesnelere mevcuttur. Bu nesnelere üzerinde çeşitli farklı *işlemler* gerçekleştirebiliriz. Halihazırda programımızda *WriteLine* işlemini kullandık. Ayrıca, *WriteLine* işlemini, tırnak işareti içerisinde **Merhaba Dünya** metninin takip ettiğini de fark etmişsinizdir. Bu metin, *WriteLine* işlemine bir girdi olarak geçilmiştir, bu da daha sonra kullanıcı için yazdırılır. Bu, işlemle ilgili bir *girdi* olarak adlandırılır. Bazı işlemler bir ya da daha fazla girdi alırken, bazıları hiç almaz.

Tırnak işaretleri, boşluklar ve parantezler gibi noktalama işaretleri, bir bilgisayar programında son derece önemlidirler. Yerlerine ve adetlerine bağlı olarak, ifade edilen anlamı değiştirebilirler.

İkinci Programımız

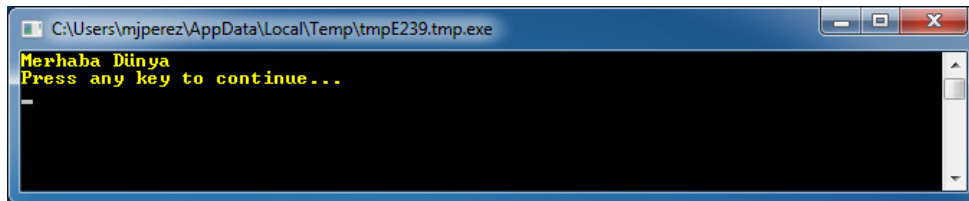
Artık ilk programımızı anladığınıza göre, buna bazı renkler ekleyerek daha süslü hale getirelim.

```
TextWindow.ForegroundColor = "Yellow"  
TextWindow.WriteLine("Merhaba Dünya")
```



Şekil 5 – Renkler Ekleme

Yukarıdaki programı çalıştırdığınızda, TextWindow aynı "Merhaba Dünya" sözcük grubunu yazdığını göreceksiniz, ancak bu kez daha önceki gibi gri yazmak yerine, sarı renkte yazacaktır.



Şekil 6 – Sarı Renkte Merhaba Dünya

Orijinal programımıza eklediğimiz yeni ifadeye dikkat edin. Bu ifadeye “Yellow” (Sarı) değerine eşitlediğimiz yeni bir kelime (ForegroundColor) kullanılıyor. Bu, ForegroundColor’a “Yellow”u atadığımız anlamına geliyor. Şimdi, ForegroundColor ile WriteLine işlemi arasındaki fark, ForegroundColor’ın herhangi bir girdi almamış veya herhangi bir paranteze ihtiyaç duymamış olmasıdır. Onun yerine, bunu bir eşittir sembolü ve bir kelime takip ediyordu. ForegroundColor’ı, TextWindow’ın bir Özelliği olarak tanımlıyoruz. İşte, ForegroundColor özelliği için geçerli olan değerlerin bir listesi. “Yellow”u bunlardan birisiyle değiştirmeyi deneyin ve sonuçları görün – tırnak işaretlerini unutmayın, bunlar gerekli noktalama işaretleridir.

Black
Blue
Cyan
Gray
Green
Magenta
Red
White
Yellow
DarkBlue
DarkCyan
DarkGray
DarkGreen
DarkMagenta
DarkRed
DarkYellow

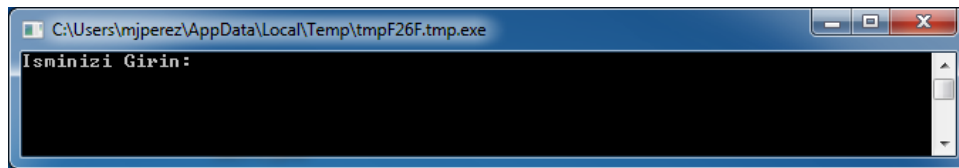
Değişkenlerin Eklenmesi

Programımızda Değişkenlerin kullanılması

Programımız genel “Merhaba Dünya?” yerine kullanıcının ismiyle birlikte “Merhaba” deseydi, daha hoş olmaz mıydı? Bunu yapmak için, önce kullanıcıya ismini sormamız ve sonra da bunu bir yerde saklayarak, kullanıcının ismiyle birlikte “Merhaba” metnini yazdırmamız gerekir. Bunu nasıl yapabileceğimizi görelim:

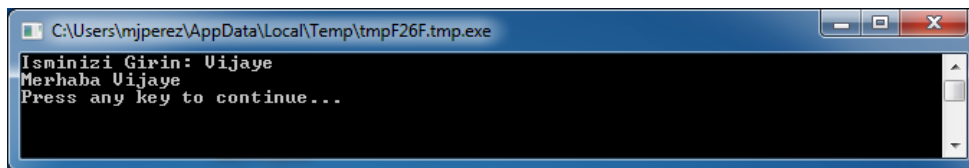
```
TextWindow.Write("İsminizi Girin: ")  
name = TextWindow.Read()  
TextWindow.WriteLine("Merhaba " + name)
```

Bu programı yazıp çalıştırdığınızda, aşağıdaki çıktıyı göreceksiniz:



Şekil 7 – Kullanıcının ismini sorun

Ve, isminizi girip ENTER tuşuna bastığınızda, aşağıdaki çıktıyı göreceksiniz:



Şekil 8 – Sıcak Bir Merhaba

Şimdi, programı tekrar çalıştırırsanız, size aynı soru tekrar sorulacaktır. Farklı bir isim girebilirsiniz ve bilgisayar size o isimle Merhaba diyecektir.

Programın analizi

Biraz önce çalıştırdığınız programda, dikkatinizi çekmiş olabilecek satır şudur:

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

Read() tıpkı *WriteLine()* gibi görünür, ancak içinde herhangi bir girdi yoktur. Bu bir işlemdir ve temel olarak bilgisayara kullanıcının bir metin girmesini ve ENTER tuşuna basmasını beklemesini söyler. Kullanıcı ENTER tuşuna bastığında, kullanıcının girdiği metni alır ve programa geri döner. İlginç olan nokta şudur ki; kullanıcının girdiği metin şimdi **isimli** bir *değişkende* saklanır. Bir *değişken*, değerleri geçici olarak saklayabildiğiniz ve sonra kullanabildiğiniz bir yer olarak tanımlanır. Yukarıdaki satırda, kullanıcının ismini saklamak için, **name** kullanılmıştır.

Bir sonraki satır da ilginçtir:

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

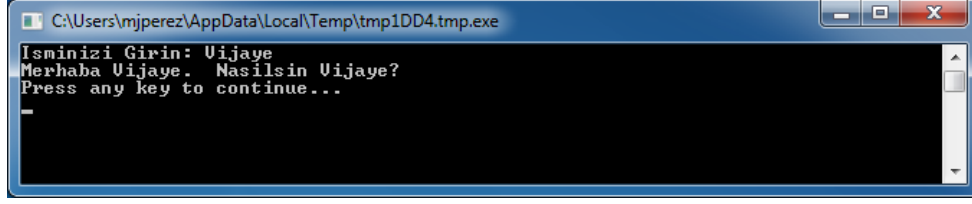
Bu, **name** (isim) değişkenimizde sakladığımız değeri kullandığımız yerdir. **name** bölümünde yazan değeri alıyoruz, bunu "Merhaba"ya ekliyoruz ve TextWindow'a yazıyoruz.

Bir değişken girildiğinde, bunu istediğiniz zaman tekrar kullanabilirsiniz. Örneğin; şunları yapabilirsiniz:

Tıpkı WriteLine gibi, Write da ConsoleWindow'da (Konsol Penceresi) bir diğer işlemdir. Write, ConsoleWindow'a bir şey yazmanıza izin verir, ancak bundan sonra gelen metnin mevcut metinle aynı satırda olmasını sağlar.

```
TextWindow.Write("İsminizi Girin: ")
name = TextWindow.Read()
TextWindow.Write("Merhaba " + name + ". ")
TextWindow.WriteLine("Nasılsın " + name + "?")
```

Ve Őu çıktıyı greceksiniz:



```
C:\Users\mjiperez\AppData\Local\Temp\tmp1DD4.tmp.exe
Isminizi Girin: Uijaye
Merhaba Uijaye. Nasilsin Uijaye?
Press any key to continue...
_
```

Őekil 9 – Bir DeęiŐkenin Tekrar Kullanılması

DeęiŐkenlerin isimlendirilmesi ile ilgili kurallar

DeęiŐkenlerin onlarla baęlantılı isimleri vardır ve onları bu Őekilde tanırırsınız. Bu deęiŐkenlerin isimlendirilmeleriyle ilgili belirli basit kurallar ve gerçekten iyi kılavuz bilgiler vardır. Bunlar:

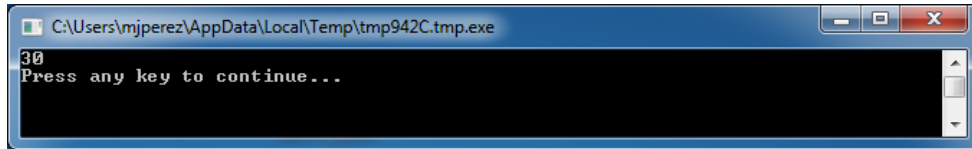
1. İsim bir harfle baŐlamalı ve **if**, **for**, **then**, vs. gibi kelimelerle çakıŐmamalıdır.
2. Bir isim, harflerin, sayıların ve altçizgilerin herhangi bir kombinasyonundan oluŐabilir.
3. DeęiŐkenleri anlamlı bir Őekilde isimlendirmek faydalıdır – deęiŐkenler istendięi kadar uzun olabileceęi iin, amalarını aıklayan deęiŐken isimleri kullanın.

Sayılarla Oynamak

Biraz nce kullanıcının ismini saklamak iin deęiŐkenleri nasıl kullanabileceęinizi grdük. Bundan sonraki birkaç programda, deęiŐkenlerde sayıları nasıl saklayabileceęimizi ve iŐleyebileceęimizi greceęiz. Gerçekten basit bir programla baŐlayalım:

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

Bu programı çalıŐtırdıęınızda, aŐaęıdaki çıktıyı greceksiniz:



```
C:\Users\mjiperez\AppData\Local\Temp\tmp942C.tmp.exe
30
Press any key to continue...
_
```

Őekil 10 – İki Sayının Toplanması

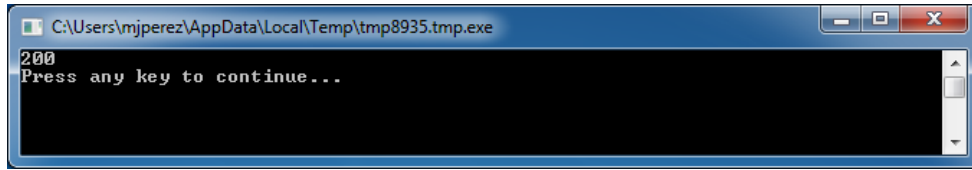
Programın birinci satırında, **number1** değişkenine 10 sayısını atıyorsunuz. Ve ikinci satırda, **number2** değişkenine 20 sayısını atıyorsunuz. Üçüncü satırda, **number1** ve **number2**'yi topluyor ve sonra çıkan sonucu **number3**'e atıyorsunuz. Böylece bu örnekte, **number3**'ün değeri 30 olacaktır. Ve TextWindow'da görüntülediğimiz değer de budur.

Sayıların başında ve sonunda tırnak işareti olmadığına dikkat edin. Sayılar için, tırnak işaretine gerek yoktur. Tırnak işaretlerine yalnızca metin kullanırken ihtiyacınız vardır.

Şimdi, programı biraz değiştirelim ve sonuçları görelim:

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

Yukarıdaki program, **number1** ile **number2**'yi çarpacak ve çıkan sonucu **number3**'te saklayacaktır. Ve bu programın sonucunda şunu görebilirsiniz:



Şekil 11 – İki Sayının Çarpılması

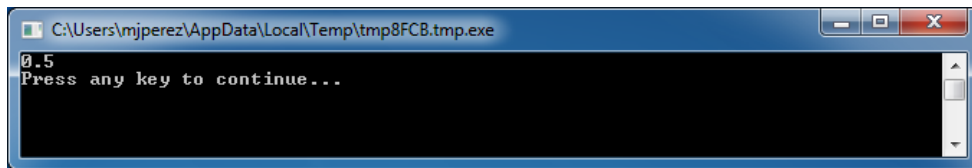
Benzer şekilde, sayıları çıkarabilir ya da bölebilirsiniz. İşte bir çıkarma işlemi:

```
number3 = number1 - number2
```

Ve bölme sembolü '/'. Program şu şekilde görünecektir:

```
number3 = number1 / number2
```

Ve bu bölmenin sonucu şu olacaktır:



Şekil 12 – İki Sayının Bölünmesi

Basit Bir Sıcaklık Dönüştürücüsü

Bir sonraki programda, Fahrenheit cinsinden sıcaklıkları Santigrada çevirmek için, $^{\circ}\text{C} = \frac{5(^{\circ}\text{F}-32)}{9}$ formülünü kullanacağız.

İlk olarak, kullanıcıdan sıcaklığı Fahrenheit cinsinden alacak ve bunu bir değişkende saklayacağız. Kullanıcıdan gelen sayıları okumamızı sağlayan özel bir işlem vardır ve bu da; **TextWindow.ReadNumber**.

```
TextWindow.Write("Sıcaklığı Fahrenheit cinsinden girin: ")
fahr = TextWindow.ReadNumber()
```

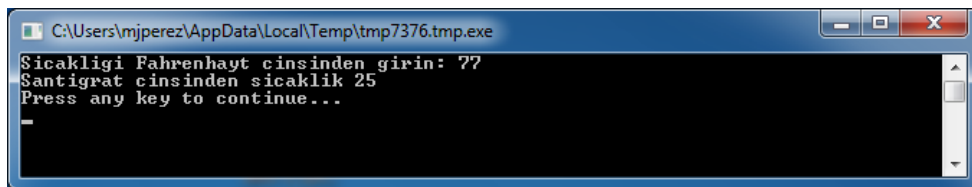
Fahrenheit cinsinden sıcaklığı bir değişkende sakladıktan sonra, bunu şu şekilde Santigrada çevirebiliriz:

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

Parantezler bilgisayara ilk önce **fahr - 32** değerini hesaplamasını ve sonra kalan işlemlere devam etmesini söyler. Şimdi, tüm yapmamız gereken, sonucu kullanıcıya göstermektir. Bunların hepsini bir araya getirdiğimizde, şu programı elde ederiz:

```
TextWindow.Write("Sıcaklığı Fahrenheit cinsinden girin: ")
fahr = TextWindow.ReadNumber()
celsius = 5 * (fahr - 32) / 9
TextWindow.WriteLine("Santigrat cinsinden sıcaklık " + celsius)
```

Ve bu programın sonucu şu olacaktır:



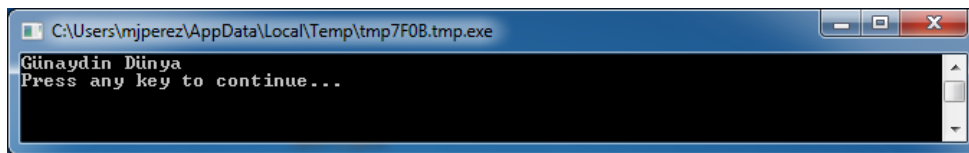
Şekil 13 – Sıcaklık Dönüşümü

Koşullar ve Dallanma

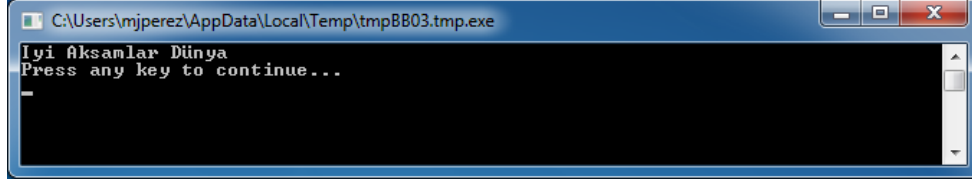
İlk programımıza geri dönecek olursak, genel bir *Merhaba Dünya* sözü yerine, günün saatine bağlı olarak, *Günaydın Dünya* ya da *İyi Akşamlar Dünya* desek daha hoş olmaz mıydı? Bir sonraki programımız için, bilgisayara saat öğlen 12:00'den önceyse *Günaydın Dünya* ve 12:00'den sonraysa *İyi Akşamlar Dünya* dedirteceğiz.

```
If (Clock.Hour < 12) Then
    TextWindow.WriteLine("Günaydın Dünya")
EndIf
If (Clock.Hour >= 12) Then
    TextWindow.WriteLine("İyi Akşamlar Dünya")
EndIf
```

Programı ne zaman çalıştırdığınıza bağlı olarak, aşağıdaki çıktılarından birisini göreceksiniz:



Şekil 14 – Günaydın Dünya



Şekil 15 – İyi Akşamlar Dünya

Programın ilk üç satırını analiz edelim. Anlamışsınızdır ki; Clock.Hour değerinin 12'den az olması durumunda, "Günaydın Dünya" yazısı yazdırılacaktır. **If**, **Then** ve **EndIf** kelimeleri, program çalışırken bilgisayar tarafından anlaşılan özel kelimelerdir. **If**

kelimesinin ardından daima bir koşul gelir, bu durumda bu koşul (**Clock.Hour < 12**)'dir. Unutmayın ki; parantezler bilgisayarın sizin niyetlerinizi anlaması için gereklidir. Koşulu **then** ve yürütülecek gerçek işlem izler. Ve işlemden sonra, **EndIf** gelir. Bu, bilgisayara koşulun uygulanmasının bittiğini söyler.

Small Basic'de, o andaki tarihe ve saate erişmek için, Saat nesnesini kullanabilirsiniz. Bu ayrıca size, o andaki Günü, Ayı, Yılı, Dakikayı, Saniyeyi ayrı ayrı alabilmenizi sağlayan bir grup özellik sağlar.

then ile **EndIf** arasında, birden fazla işlem olabilir ve koşulun geçerli olması durumunda bilgisayar bunların tümünü uygulayacaktır. Örneğin; söyle bir şey yazabilirsiniz:

```
If (Clock.Hour < 12) Then
    TextWindow.Write("Günaydın. ")
    TextWindow.WriteLine("Kahvaltı nasıldı?")
EndIf
```

Else

Bu bölümün başındaki programda, ikinci koşulun biraz gereksiz olduğunu fark etmiş olabilirsiniz. **Clock.Hour** değeri, 12'den az olabilir ya da olmayabilirdi. Gerçekten ikinci bir kontrol yapmamız gerekmedi. Bu gibi zamanlarda, iki **if..then..endif** ifadesini, yeni bir kelime olan **else**'i kullanarak, tek bir kelimeye kısaltabiliriz.

Eğer bu programı **else**'i kullanarak yeniden yazsaydık, şöyle görünecekti:

```
If (Clock.Hour < 12) Then
    TextWindow.WriteLine("Günaydın Dünya")
Else
    TextWindow.WriteLine("İyi Akşamlar Dünya")
EndIf
```


Ve bu program diğeriyle aynı işi yapacaktır, bu da bizi bilgisayar programlamasında çok önemli bir derse getirir:

“ Programlamada, genellikle aynı şeyi yapmanın pek çok yolu vardır. Bazen bir yöntem diğerinden daha anlamlı gelir. Seçim, programcıya bırakılmıştır. Daha çok program yazdıkça ve daha deneyimli hale geldikçe, bu farklı teknikleri ve avantajları ile dezavantajlarını fark etmeye başlayacaksınız.

Girintili Yazmak

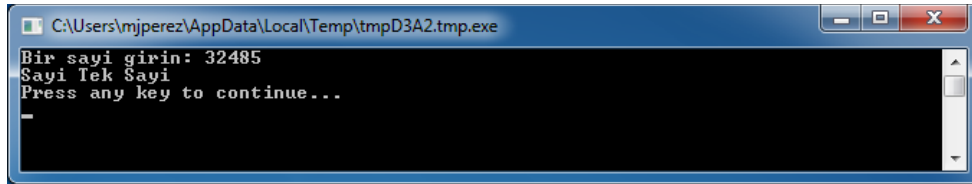
Tüm örneklerde, *If*, *Else* ve *EndIf* arasındaki ifadelerin girintili yazıldığını görebilirsiniz. Bu girintili yazma işlemi gerekli değildir. Bilgisayar programları bunlar olmadan da gayet iyi anlayacaktır. Ancak, bunlar programın yapısını daha kolay anlamamızı sağlarlar. Bu yüzden, bu tip bloklar arasındaki ifadeleri girintili yazmak genellikle iyi bir uygulama olarak değerlendirilir.

Çift ya da Tek

Şimdi artık elimizde **If..Then..Else..EndIf** ifadesi olduğuna göre, verilen bir sayının tek mi, çift mi olduğunu söyleyecek bir program yazalım.

```
TextWindow.Write("Bir sayı girin: ")
num = TextWindow.ReadNumber()
remainder = Math.Remainder(num, 2)
If (remainder = 0) Then
    TextWindow.WriteLine("Sayı Çift Sayı")
Else
    TextWindow.WriteLine("Sayı Tek Sayı")
EndIf
```

Ve bu programı çalıştırdığınızda, şöyle bir çıktı göreceksiniz:



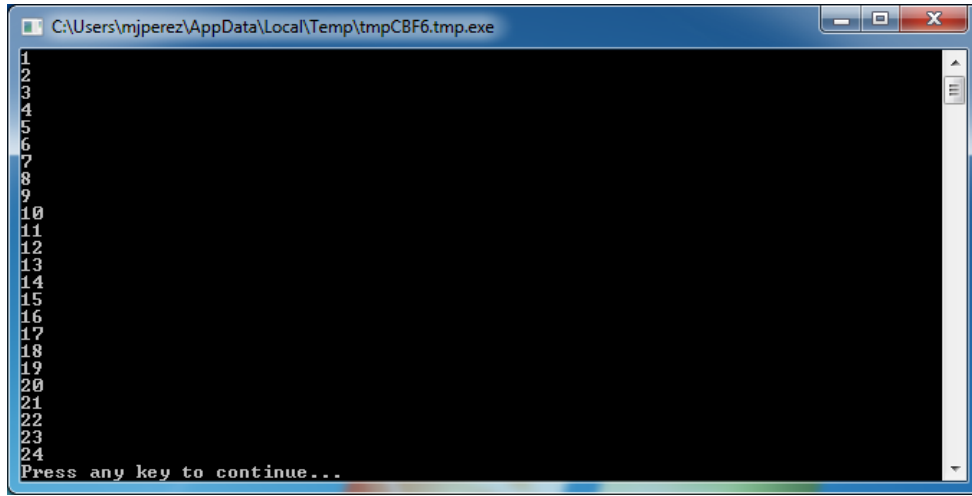
Şekil 16 – Çift ya da Tek

Bu programda, yeni bir faydalı işlem olan **Math.Remainder** işlemi tanıttık. Ve evet, belki de anlamış olduğunuz gibi, **Math.Remainder** ilk sayıyı ikinci sayıya bölecek ve sonra kalanını verecektir.

Dallanma

Hatırlarsanız, ikinci bölümde, bilgisayarın bir programı yukarıdan aşağıya doğru her defasında bir ifadeyi işlemden geçirecek şekilde çalıştığını öğrenmiştiniz. Bununla birlikte, bilgisayarın sıranın dışına çıkarak bir başka ifadeye atlamasını sağlayan özel bir ifade vardır. Bir sonraki programa bir göz atalım.

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



Şekil 17 – Goto Komutunun Kullanılması

Yukarıdaki programda, *i* değişkenine 1 değerini atadık. Ve sonra, iki nokta üst üste (:) ile biten yeni bir ifade ekledik.

```
start:
```

Bu, bir *etiket* olarak adlandırılır. Etiketler, bilgisayarın anlayabildiği yer imleri gibidir. Yer imini istediğiniz gibi adlandırabilirsiniz ve her biri farklı adlandırılmış olması koşuluyla, programınıza istediğiniz kadar etiket ekleyebilirsiniz.

Buradaki bir diğer ilginç ifade de şudur:

```
i = i + 1
```

Bu yalnızca bilgisayara *i* değişkenine 1 eklemesini ve onu tekrar *i* değişkenine atmasını söyler. Yani, *i*'nin değeri bu ifadeden önce 1 ise, ifade çalıştırdıktan sonra 2 olacaktır.

Ve son olarak da,

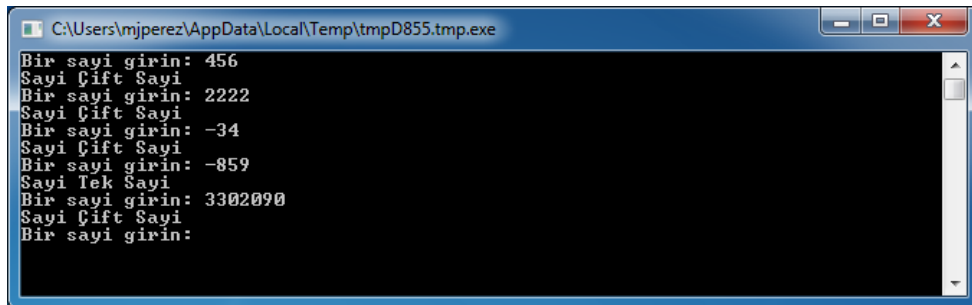
```
If (i < 25) Then
    Goto start
EndIf
```

Bu, bilgisayara *i*'nin değeri 25'den küçükse, ifadeleri **start** yer iminden başlayarak uygulamasını söyleyen bölümdür.

Sonsuz uygulama

Goto ifadesini kullanarak, bilgisayarın bir şeyi istediğiniz defa tekrarlamasını sağlayabilirsiniz. Örneğin; Çift ya da Tek programını alıp, aşağıdaki gibi değiştirdiğinizde, program sonsuza kadar çalışacaktır. Pencerenin üst sağ köşesindeki Kapatma (X) düğmesine basarak programı durdurabilirsiniz.

```
begin:
TextWindow.Write("Bir sayı girin: ")
num = TextWindow.ReadNumber()
remainder = Math.Remainder(num, 2)
If (remainder = 0) Then
    TextWindow.WriteLine("Sayı Çift Sayı")
Else
    TextWindow.WriteLine("Sayı Tek Sayı")
EndIf
Goto begin
```



Şekil 18 – Çift ya da Tek sonsuza kadar çalışan

Döngü İçin

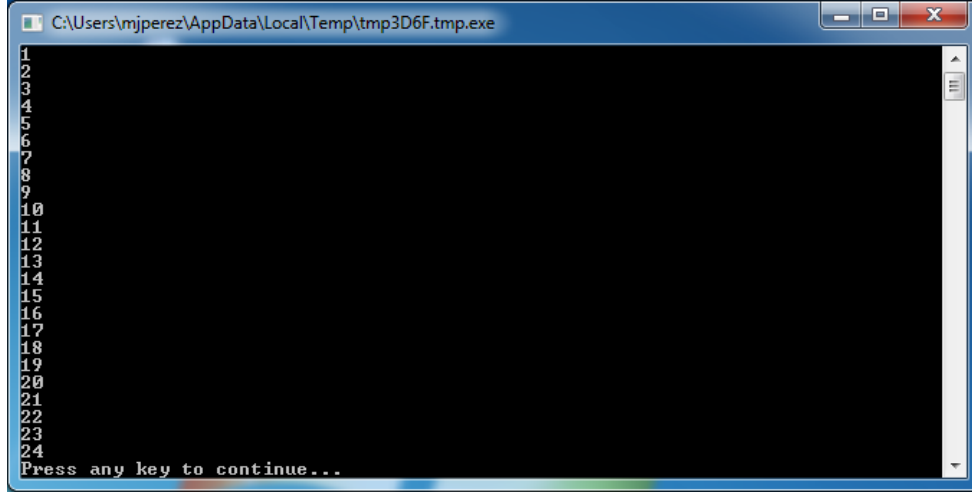
Daha önceki bölümde yazdığımız bir programı ele alalım.

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

Bu program, 1'den 24'e kadar sayıları sırayla yazdırıyor. Bu bir değişkeni artırma süreci programlamada oldukça yaygın olduğundan, programlama dilleri genellikle bunu yapmak için daha kolay bir yöntem sunarlar. Yukarıdaki program, aşağıdaki programa eşdeğerdir:

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

Ve çıktısı da şudur:



```
C:\Users\mjperrez\AppData\Local\Temp\tmp3D6F.tmp.exe
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
Press any key to continue...
```

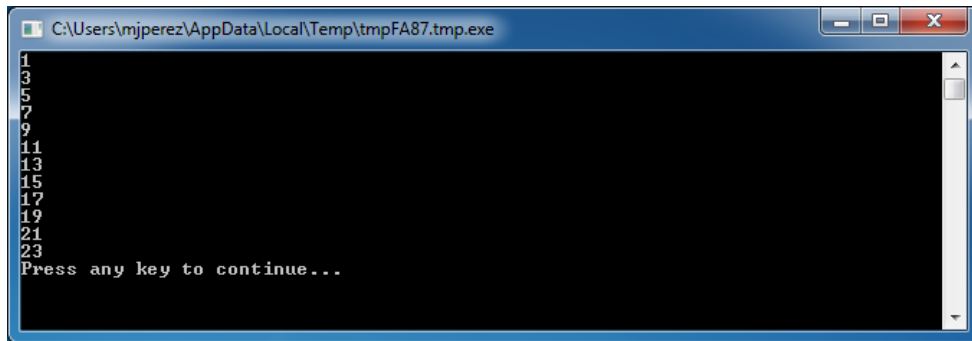
Şekil 19 – For Döngüsünün Kullanılması

Gördüğünüz gibi, 8 satırlık bir programı 4 satırlık bir programa düşürdük ve yine de 8 satırlık programla aynı işi yapıyor! Daha önce, genellikle aynı şeyi yapmanın çeşitli yolları olduğunu söylediğimizi hatırlayın. İşte bu, harika bir örnek.

For..EndFor, programlama dilinde bir *döngü* olarak adlandırılır. Bu size, bir değişkeni alıp, ona bir başlangıç ve bitiş noktası vermenizi ve bilgisayarın değişkeni sizin için artırmasını sağlar. Bilgisayar değişkenin değerini her artırdığında, **For** ve **EndFor** arasındaki ifadeleri çalıştırır.

Ancak, eğer değişkenin birer birer yerine diyelim ki ikişer ikişer artmasını isteseydiniz, 1 ile 24 arasındaki tüm tek sayıları yazdırmak isteyecektiniz, döngüyü bunu yapmak için de kullanabilirsiniz.

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

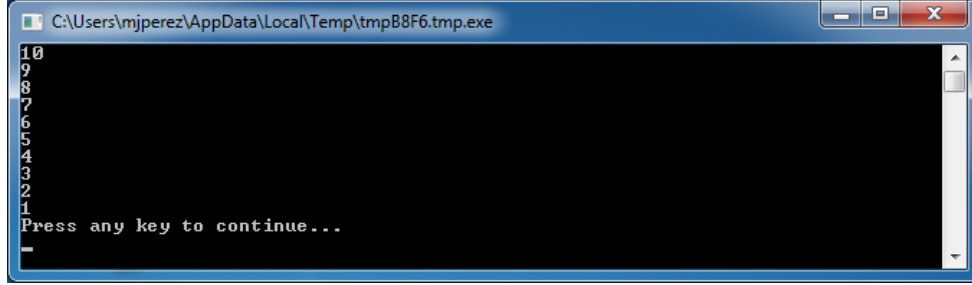


```
C:\Users\mjperrez\AppData\Local\Temp\tmpFA87.tmp.exe
1
3
5
7
9
11
13
15
17
19
21
23
Press any key to continue...
```

Şekil 20 – Yalnızca Tek Sayılar

İfadenin **Step 2** bölümü, **For** bilgisayara **i**'nin değerini 1 yerine 2 artırmasını söyler. **Step**'ı kullanarak, istediğiniz aralıklarla artırma yapabilirsiniz. Step için negatif bir değer bile belirleyebilirsiniz ve bu durumda bilgisayar, aşağıdaki örnekte olduğu gibi geriye doğru sayar:

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

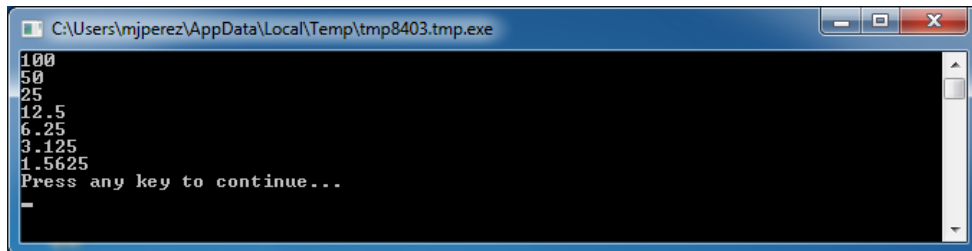


Şekil 21 – Geriye Doğru Saymak

While Döngüsü

While döngüsü, bir diğer döngü yöntemidir; bu yöntem, özellikle döngü sayısı önceden bilinmediği zaman faydalıdır. Bir For döngüsü önceden tanımlandığı kadar çalışırken, While döngüsü verilen bir koşul doğru hale gelinceye kadar çalışır. Aşağıdaki örnekte, sonuç 1'den büyük olduğu sürece bir sayıyı ikiye bölüyoruz.

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



Şekil 22 – İkiye Bölme Döngüsü

Yukarıdaki programda, *number*'a 100 değerini atıyoruz ve sayı 1'den büyük olduğu sürece While döngüsünü çalıştırıyoruz. Döngünün içinde, sayıyı yazdırıyoruz ve sonra da ikiye bölüp, yarısını buluyoruz. Ve beklendiği şekilde, programın çıktısı birbiri ardına yarıya inen sayılar oluyor.

Bu programı For döngüsünü kullanarak yazmak gerçekten zor olurdu, çünkü döngünün kaç kez çalışması gerektiğini bilemezdik. Bir While döngüsü ile, bir koşulu kontrol etmek ve bilgisayara döngüyü sürdürmesini ya da bırakmasını söylemek kolaydır.

Tüm While döngülerinin bir If..Then ifadesine dönüştürülebileceğini bilmek ilginçtir. Örneğin; yukarıdaki program, sonucu etkilemeden aşağıdaki şekilde de yazılabilir.

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

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

Aslında, bilgisayar dahili olarak, her bir While döngüsünü bir ya da daha fazla Goto ifadesi ile birlikte If..Then kullanan ifadeler şeklinde yeniden yazar.

Grafiklere Giriş

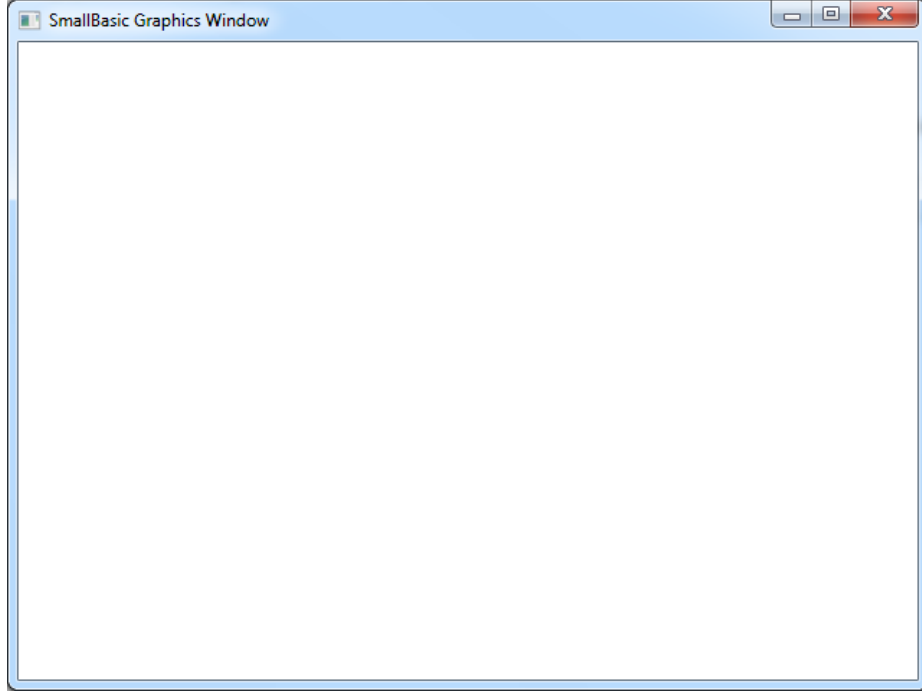
Şimdiye kadar verdiğimiz tüm örneklerimizde, Small Basic dilinin temellerini açıklamak için, TextWindow'u kullandık. Bununla birlikte, Small Basic'de bu bölümde araştırmaya başlayacağımız güçlü bir Grafik özelliği seti de bulunur.

GraphicsWindow'a Giriş

Tıpkı, Metinlerle ve Sayılarla çalışmamıza izin veren TextWindow gibi, Small Basic aynı zamanda bir şeyler çizmemizi sağlayan bir **GraphicsWindow** da sunar. GraphicsWindow'u görüntüleyerek başlayalım.

```
GraphicsWindow.Show()
```


Bu programı çalıştırdığınızda, bildiğimiz siyah metin penceresi yerine, aşağıdaki gibi beyaz bir pencerenin açıldığını göreceksiniz. Henüz bu pencerede yapacak fazla bir şey yoktur. Ancak, bu bölümde üzerinde çalışacağımız zemin pencere bu olacak. Pencerenin üst sağ köşesindeki "X" düğmesine basarak bu pencereyi kapatabilirsiniz.



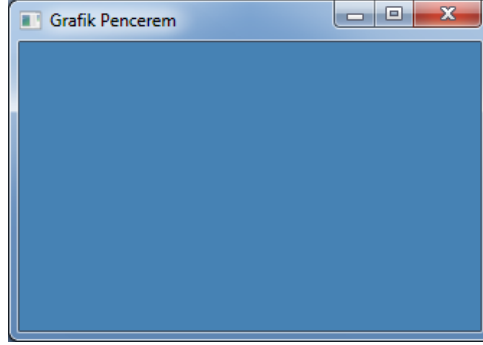
Şekil 23 – Boş Bir Grafik Penceresi

Grafik Penceresinin Kurulumu

Grafik penceresi, görünüşünü istediğiniz gibi ayarlamanıza izin verir. Bu pencerenin başlığını, arka planını ve boyutunu değiştirebilirsiniz. Şimdi devam edelim ve pencereyi daha iyi tanımak için, onu birazcık değiştirelim.

```
GraphicsWindow.BackgroundColor = "SteelBlue"  
GraphicsWindow.Title = "Grafik Pencerem"  
GraphicsWindow.Width = 320  
GraphicsWindow.Height = 200  
GraphicsWindow.Show()
```

Özelleştirilmiş bir pencere işte böyle görünür. Arka plan rengini Ek B’de listelenen pek çok renkten birisine değiştirebilirsiniz. Pencerenin görünümünü nasıl değiştirebileceğinizi görmek için, bu özelliklerle oynayın.

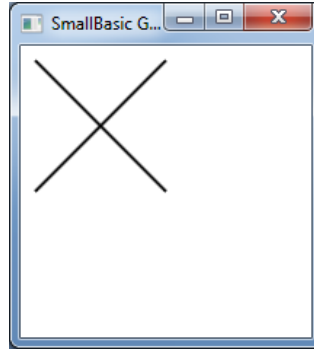


Şekil 24 – Özelleştirilmiş Bir Grafik Penceresi

Çizgiler Çizmek

GraphicsWindow’ni açtıktan sonra, üzerine şekil, metin ve hatta resim çizebiliriz. Bazı basit şekiller çizmekle başlayalım. İşte, Grafik Penceresine bir çift çizgi çizen bir program.

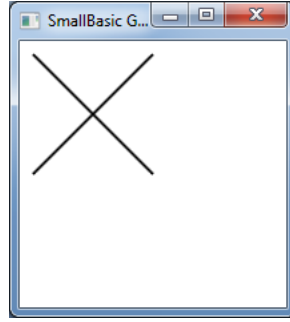
```
GraphicsWindow.Width = 200  
GraphicsWindow.Height = 200  
GraphicsWindow.DrawLine(10, 10, 100, 100)  
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Şekil 25 – Çapraz İşareti

Programın ilk iki satırı, pencereyi ayarlar ve ondan sonraki iki satır da çapraz işaretinin çizgilerini çizer. `DrawLine`'dan sonra gelen ilk iki sayı, başlangıç x ve y koordinatlarını ve diğer ikisi de, bitiş x ve y koordinatlarını belirtir. Bilgisayar grafikleri ile ilgili ilginç olan şey, koordinatların (0, 0) pencerenin üst sol köşesinden başlamasıdır. Gerçekten, koordinat uzayında, pencere 2' dördünde gibi kabul edilir.

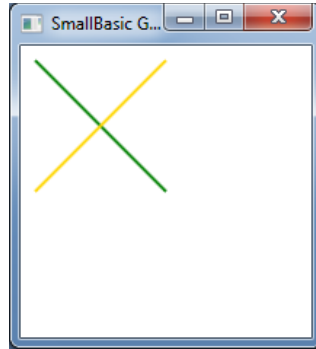
Renkler için isimler kullanmak yerine, internet renk gösterimini de (#RRGGBB) kullanabilirsiniz. Örneğin; #FF0000 Kırmızı renge karşılık gelir, #FFFF00 Sarı renge, vs. [TODO Renkler bölümünde] renkler hakkında daha fazla bilgi edineceğiz



Şekil 26 – Koordinat haritası

Çizgi programına geri dönersek, Small Basic'in çizginin renk ve kalınlık gibi özelliklerini değiştirmenize izin vermesi ilginçtir. İlk önce, aşağıdaki programda gösterildiği şekilde, çizgilerin rengini değiştirelim.

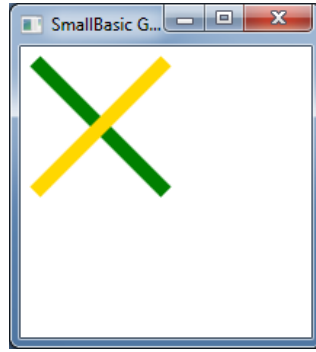
```
GraphicsWindow.Width = 200  
GraphicsWindow.Height = 200  
GraphicsWindow.PenColor = "Green"  
GraphicsWindow.DrawLine(10, 10, 100, 100)  
GraphicsWindow.PenColor = "Gold"  
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Şekil 27 – Çizgi Renginin Değiştirilmesi

Şimdi, boyutunu da değiştirelim. Aşağıdaki programda, çizgi kalınlığını varsayılan değer olan 1 yerine, 10 olarak değiştiriyoruz.

```
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)
```



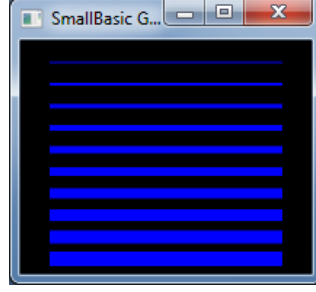
Şekil 28 – Kalın Renkli Çizgiler

PenWidth ve *PenColor*, bu çizgilerin çizildiği kalemi değiştirir. Bunlar yalnızca çizgileri değil, özellikler güncellendikten sonra çizilen tüm şekilleri etkilerler.

Daha önceki bölümlerde öğrendiğimiz döngü yapan ifadeleri kullanarak, kolayca, kalem kalınlığı gittikçe artan birden fazla çizgi çizebiliriz.

```
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
```



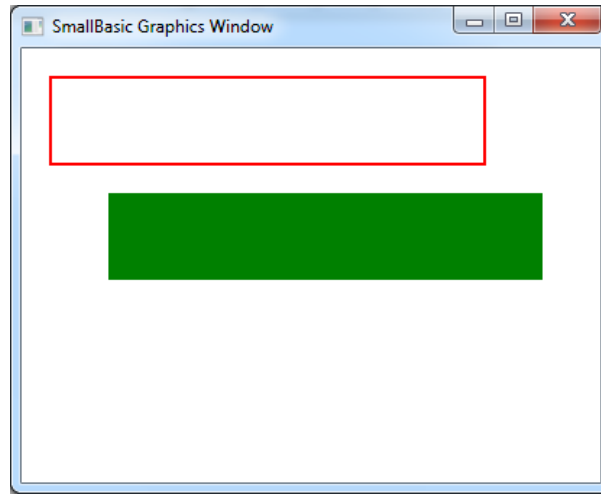
Şekil 29 – Birden Fazla Kalem Kalınlığı

Bu programın ilginç bölümü; döngü her çalıştığında PenWidth'in artması ve sonra eskisinin altına yeni bir çizgi çizmesidir.

Şekiller Çizmek ve İçlerini Doldurmak

İş şekiller çizmeye geldiğinde, her şekil için genellikle iki tip işlem vardır. Bunlar, *Çizme* ve *İçini Doldurma* işlemleridir. Çizme işlemleri, bir kalem kullanarak şeklin dış çerçevesini çizer ve İçini Doldurma işlemleri de, bir fırça kullanarak şekli boyar. Örneğin; aşağıdaki programda, iki adet dikdörtgen var, bunlardan birisi kırmızı bir kalem kullanılarak çizilmiş ve diğeri de Yeşil Fırça kullanılarak içi doldurulmuş.

```
GraphicsWindow.Width = 400  
GraphicsWindow.Height = 300  
  
GraphicsWindow.PenColor = "Red"  
GraphicsWindow.DrawRectangle(20, 20, 300, 60)  
  
GraphicsWindow.BrushColor = "Green"  
GraphicsWindow.FillRectangle(60, 100, 300, 60)
```



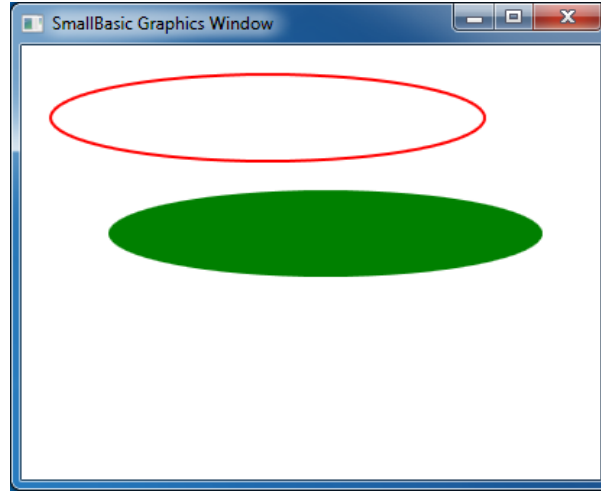
Şekil 30 – Çizmek ve İçlerini Doldurmak

Bir dikdörtgen çizmek veya içini doldurmak için, dört sayıya ihtiyacınız vardır. İlk iki sayı, dikdörtgenin üst sol köşesinin X ve Y koordinatlarını temsil eder. Üçüncü sayı, dikdörtgenin genişliğini, dördüncü ise yüksekliğini belirtir. Aslında, aynı şey aşağıdaki programdaki elipslerin çizilmesi ve içlerinin doldurulması için de geçerlidir.

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

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

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



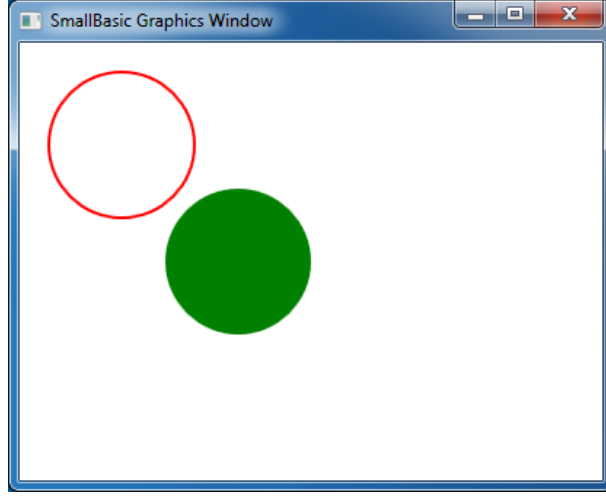
Şekil 31 – Elipsler Çizmek ve İçlerini Doldurmak

Elipsler, yalnızca genel bir daire biçimidir. Daireler çizmek isterseniz, aynı genişliği ve yüksekliği belirtmeniz gerekir.

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

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

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



Şekil 32 – Daireler

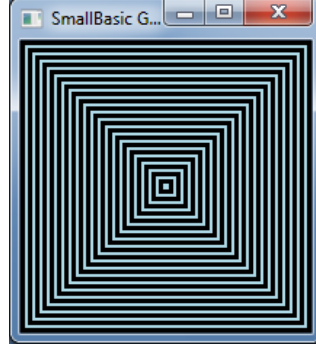
Şekillerle Eğlence

Bu bölümde, şu ana kadar öğrendiklerimizle biraz eğleneceğiz. Bu bölüm, bazı hoş görünümlü programlar yaratmak için, şu anda kadar öğrendiklerinizi birleştirmenin bazı ilginç yöntemlerini gösteren örnekler içerir.

İç İçe Dikdörtgenler

Burada, bir döngü içerisinde boyutları gittikçe artan dikdörtgenler çizeceğiz.

```
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
```

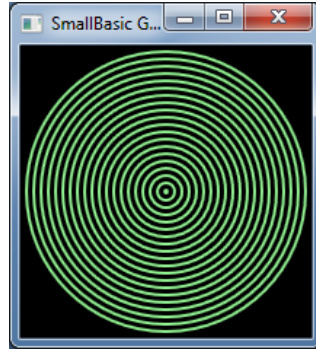



Şekil 33 – İç İçe Dikdörtgenler

İç İçe Daireler

Bir önceki programın bir varyasyonu olan bu program, kereler yerine daireler çizer.

```
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
```

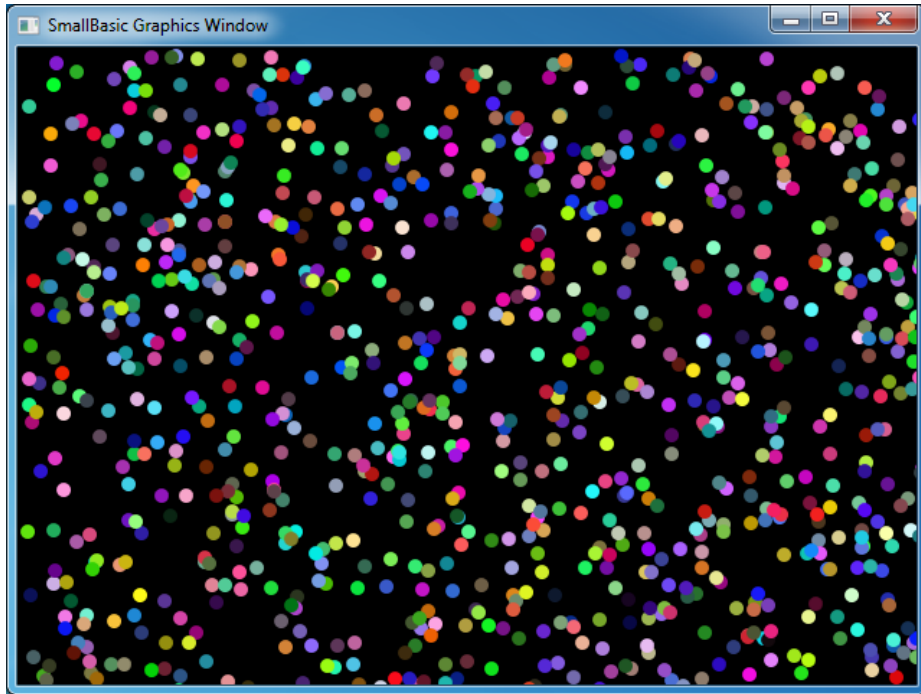


Şekil 34 – İç İçe Daireler

Rasgeleleřtirmek

Bu program, fırçanın rengini rasgele seçmek için *GraphicsWindow.GetRandomColor* işlemini ve sonra da dairelerin x ve y koordinatlarını seçmek için *Math.GetRandomNumber* işlemini kullanır. Bu iki işlem, çalıştıkları her seferde farklı sonuçlar veren ilginç programlar oluşturmak için, ilginç şekillerde birleştirilebilirler.

```
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
```



Şekil 35 – Rasgeleleřtirmek

Benzerlerin Oluşturduğu Şekiller

Aşağıdaki program, rasgele sayıları kullanarak, benzer şekillerden oluşan basit bir üçgen çizer. Benzerlerin oluşturduğu şekil, her biri tam olarak ana şekle benzeyen bölümlere ayrılabilen geometrik bir şekildir. Bu durumda program, her biri ana üçgene benzeyen yüzlerce üçgen çizer. Ve program birkaç saniye çalıştıktan sonra, üçgenlerin yalnızca noktalardan oluştuğunu görebilirsiniz. Bunun mantığını açıklamak zordur ve bunun keşfini bir alıştırmaya olarak size bırakıyorum.

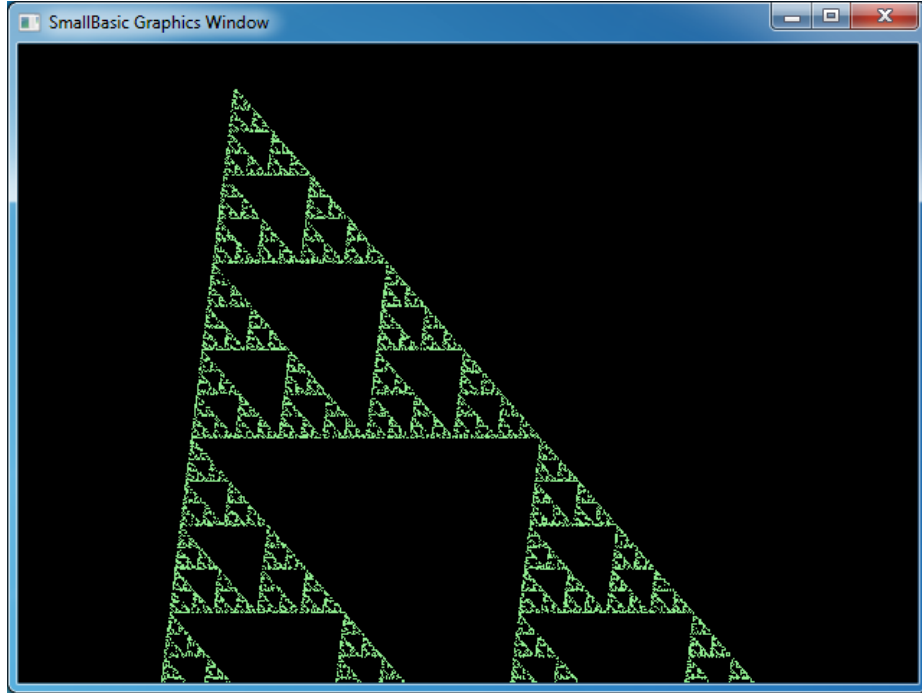
```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100

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
```



Şekil 36 – Benzer Şekillerden Oluşan Üçgen

Noktaların bu şekli oluşturmasını görmek istiyorsanız, **Program.Delay** işlemini kullanarak, döngüde bir gecikme yaratabilirsiniz. Bu işlem, gecikmeyi milisaniye olarak belirten bir sayı kullanır. İşte, programın değiştirilmiş hali, değiştirilen yer koyu renkte gösterilmiştir.

```
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
```

Gecikmenin artırılması, programı yavaşlatacaktır. İstedığınızı bulmak için sayılarla oynayın.

Bu program üzerinde yapabileceğiniz bir başka değişiklik de,

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

satırını aşağıdaki satır ile değiştirmektir:

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

Bu değişiklik, programa üçgenin piksellerini rasgele renkler kullanarak çizdirecektir.

Turtle Graphics

Logo

1970'lerde, az sayıda arařtırmacı tarafından kullanılan, oldukça basit, ancak güçlü, Logo isimli bir programlama dili vardı. Bu, birisi dile "Turtle Graphics" olarak adlandırılan şeyi ekleyinceye ve ekrana *Move Forward*, *Turn Right*, *Turn Left*, (İleri Git, Sağa Dön, Sola Dön) vs. gibi komutlara yanıt veren bir "Kurbağa" (Turtle) yerleřtirinceye kadar devam etti. Kurbağayı kullanarak, insanlar ekranda ilginç şekiller çizebiliyorlardı. Bu, dili derhal erişilebilir ve tüm yařtan insanlar açısından çekici hale getirdi ve 1980'lerde popülerliđinin oldukça artmasının en büyük nedeniydi.

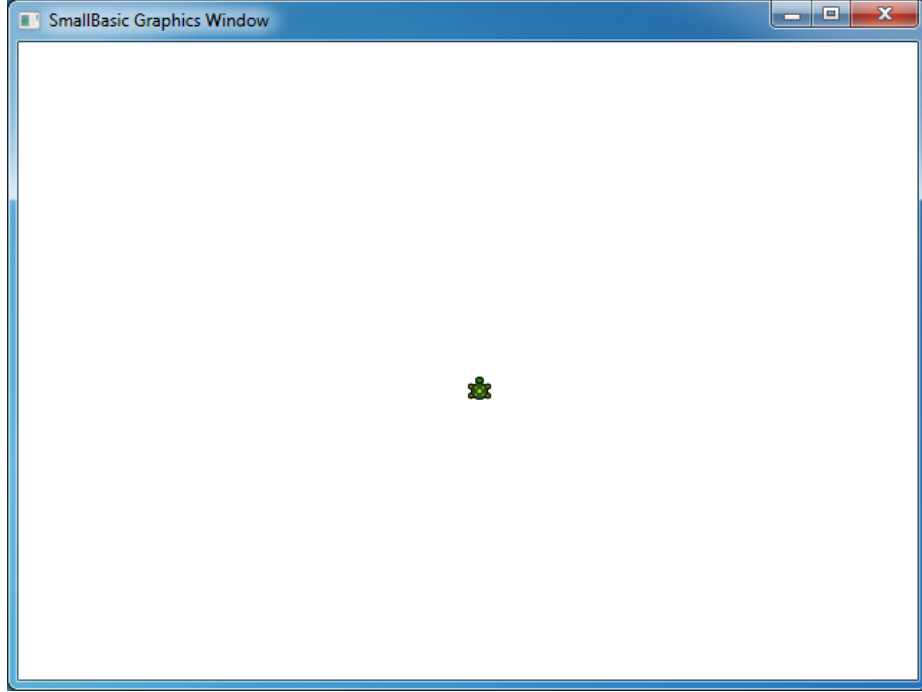
Small Basic'de, programların içerisindeki programlardan çağrılabilen pek çok komuta sahip olan bir **Turtle** nesnesi bulunur. Bu bölümde, Kurbağayı ekranda grafikler çizmek için kullanacağız.

Kurbağa

Başlangıç olarak, Kurbağanın ekranda görünür hale gelmesini sağlamamız gerekiyor. Bu tek satırlık basit bir programla yapılabilir.

```
Turtle.Show()
```

Bu programı çalıştırdığınızda, merkezinde bir Kurbağa bulunması dışında, tıpkı daha önceki bölümde gördüğümüz gibi beyaz bir pencerenin açıldığını göreceksiniz. Talimatlarımızı izleyecek ve çizmesini istediğimiz şeyi çizecek olan Kurbağa budur.



Şekil 37 – Kurbağa görünür halde

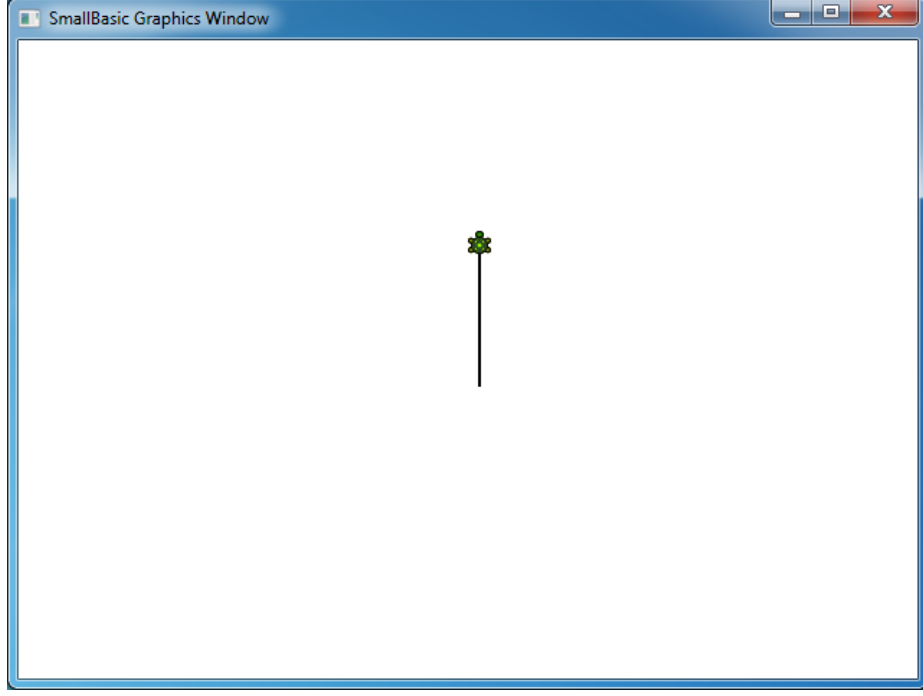
Hareket Ettirmek ve Şekiller Çizmek

Kurbağanın anladığı talimatlardan birisi **Move**'dur. Bu işlem bir sayıyı girdi olarak alır. Bu sayı, Kurbağaya ne kadar uzağa gitmesi gerektiğini söyler. Diyelim ki; aşağıdaki örnekte Kurbağaya 100 piksel hareket etmesini söyleyeceğiz.

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

Bu programı çalıştırdığınızda, kurbağanın yukarıya doğru yavaşça 100 piksel hareket ettiğini görebilirsiniz. Hareket ettikçe, arkasında bir çizgi çizdiğini de fark edeceksiniz. Kurbağa hareket etmeyi bitirdiğinde, sonuç aşağıdaki şekildeki gibi olacaktır.

Kurbağa üzerinde işlemler kullanırken, Show() komutunu kullanmaya gerek yoktur. Ne zaman bir Kurbağa işlemi gerçekleştirilirse, Kurbağa otomatik olarak görünür hale gelecektir.



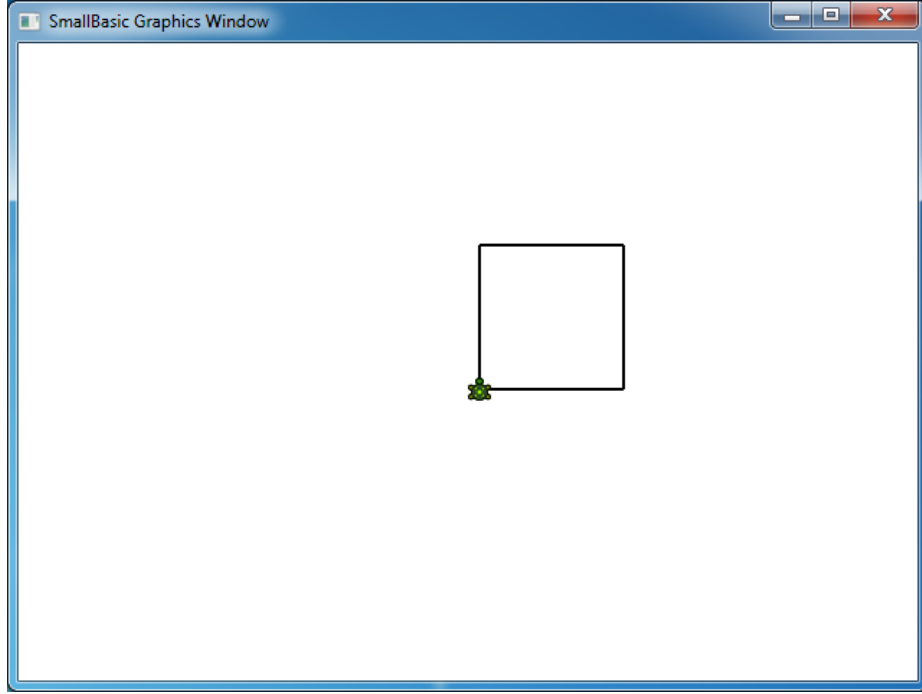
Şekil 38 – Yüz piksel hareket ettirmek

Bir Kare Çizmek

Karenin iki dikey, iki de yatay dört kenarı vardır. Bir kare çizmek için, Kurbağanın bir çizgi çizmesini, sağa dönmesini ve bir başka çizgi çizmesini ve dört kenar da tamamlanıncaya kadar bu işleme devam etmesini sağlamamız gerekir. Eğer bunu bir programa çevirseydik, şöyle görünürdü.

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


Bu programı çalıştırdığınızda, Kurbağanın her defada bir çizgi olmak üzere bir kare çizdiğini ve sonucun aşağıdaki şekildeki gibi olacağını görebilirsiniz.



Şekil 39 – Bir kare çizen Kurbağa

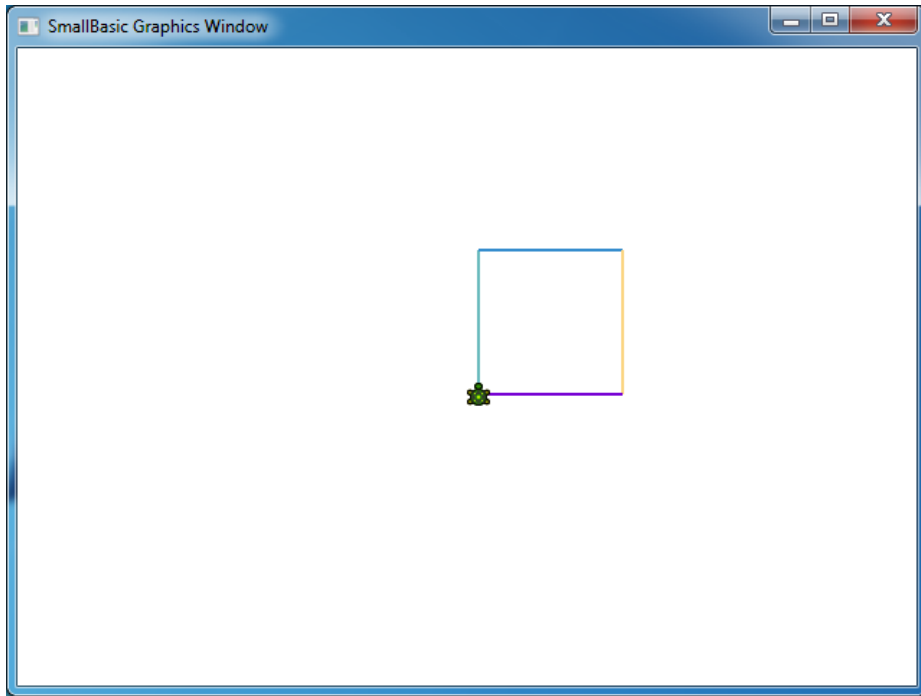
Aynı talimatları tekrar tekrar yazıyor olduğumuzu not etmek ilginç olacaktır – tam olarak dört defa. Ve bu tip tekrarlanan komutların döngü kullanılarak uygulanabileceğini daha önce öğrenmiştik. Yani, yukarıdaki programı alıp, **For..EndFor** döngüsünü kullanacak şekilde değiştirecek, sonuçta ortaya çok daha basit bir program çıkacaktır.

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

Renleri Deęiřtirmek

Kurbaęa, daha önceki bölümde gördüğümüzle aynı GraphicsWindow'da çizim yapar. Bu da, önceki bölümde öğrendiğimiz tüm işlemlerin burada da geçerli olduęu anlamına gelir. Örneęin; ařağıdaki program, karenin her bir kenarını farklı renkte çizecektir.

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



řekil 40 – Renleri Deęiřtirmek

Daha karmařık řekiller çizmek

TurnRight ve **TurnLeft** işlemlerine ek olarak, Kurbaęanın bir de **Turn** işlemleri vardır. Bu işlem, dönme açısını belirten bir girdi alır. Bu işlem kullanılarak, herhangi bir çokgeni çizmek mümkündür. Ařağıdaki program bir altıgen (altı kenarlı bir çokgen) çizer.

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

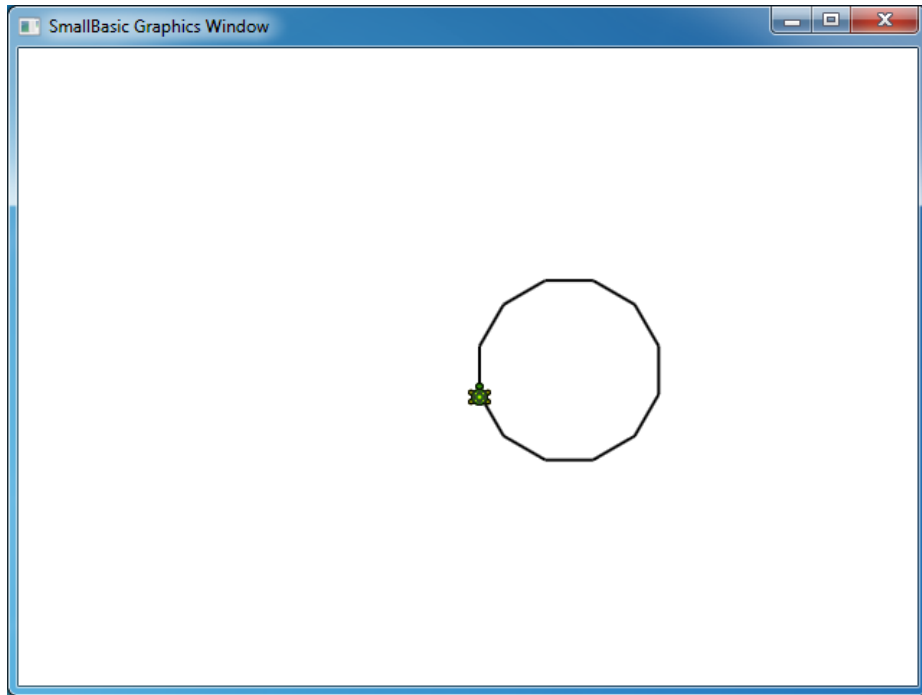
Gerçekten bir altıgen çizip çizmediğini görmek için bu programı deneyin. Kenarlar arasındaki açı 60 derece olduğu için, **Turn(60)** komutunu kullandığımıza dikkat edin. Tüm kenarları eşit olan bu tip bir çokgen için, 360 sayısı kenar adedine bölünerek, kenarlar arasındaki açı kolayca elde edilebilir. Bu bilgiyle kuşatıldıktan sonra ve değişkenleri kullanarak, artık herhangi adette kenara sahip bir çokgen çizmek için oldukça genel bir program yazabiliriz.

```
sides = 12

length = 400 / sides
angle = 360 / sides

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

Bu programı kullanarak, yalnızca **sides** değişkenini değiştirerek, herhangi bir çokgen çizebilirsiniz. Buraya 4 sayısını koymak, bize başlangıçta çizdiğimiz Kareyi verecektir. Örneğin; 50 gibi yeteri derecede büyük bir değer girmek, sonucun bir daireye çok benzemesine neden olacaktır.



Şekil 41 – 12 kenarlı bir çokgen çizmek

Biraz önce öğrendiğimiz tekniği kullanarak, ilginç bir sonuç elde edecek şekilde Kurbağanın her seferinde biraz kayarak birden fazla daire çizmesini sağlayabiliriz.

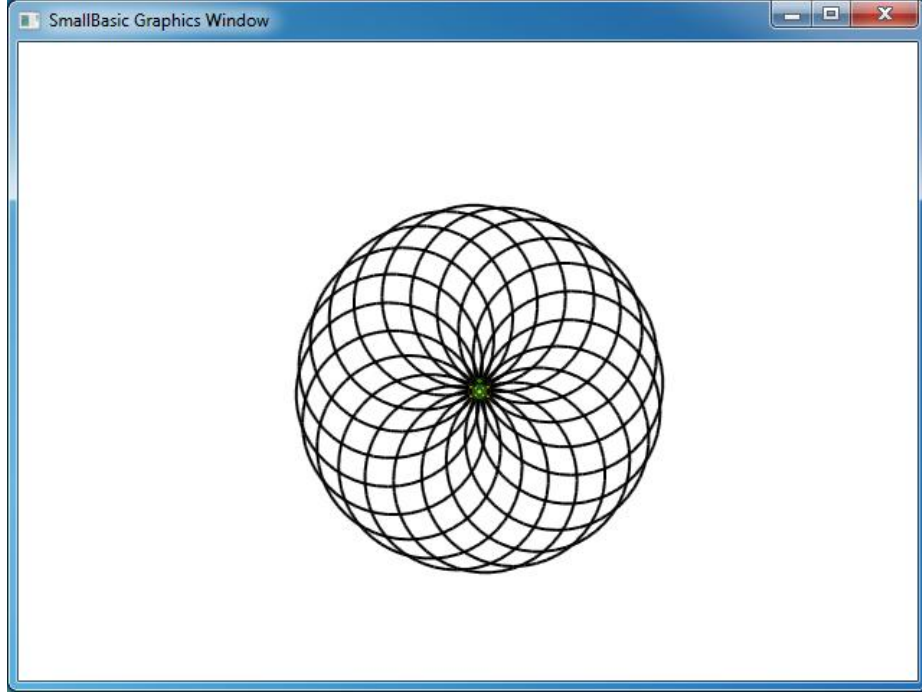
```
sides = 50
length = 400 / sides
angle = 360 / sides

Turtle.Speed = 9

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

Yukarıdaki programda, birisi diğerinin içerisinde iki adet **For..EndFor** döngü vardır. İçteki döngü ($i = 1$ to $sides$), çokgen programına benzer ve bir daire çizmekten sorumludur. Dıştaki döngü, ($j = 1$ to 20) çizilen her bir daire için, Kurbağayı biraz döndürmekten sorumludur. Bu, Kurbağaya 20 adet daire çizmesini söyler. Hepsini bir araya getirildiğinde, bu program aşağıda gösterildiği şekilde ilginç bir motifle sonuçlanır.

Yukarıdaki programda, Hızı 9 olarak girerek, Kurbağanın daha hızlı hareket etmesini sağladık. Kurbağanın istediğiniz kadar hızlı gitmesi için, bu değeri 1 ile 10 arasında herhangi bir sayı olarak girebilirsiniz.



Şekil 42 – Daireler halinde hareket etmek

Çevrede Dolaşmak

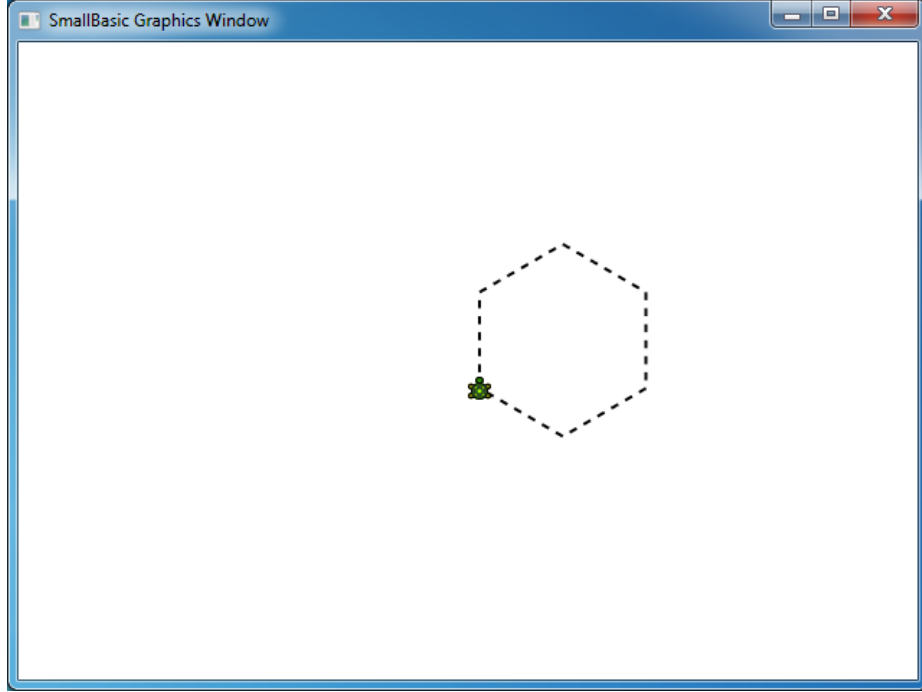
PenUp işlemini çağırarak, kurbağanın çizmemesini sağlayabilirsiniz. Bu, sizin kurbağayı çizim yapmadan ekranda herhangi bir yere hareket ettirebilmenizi sağlar. **PenDown** işleminin çağırılması, kurbağanın tekrar çizmeye başlamasını sağlayacaktır. Bu, noktalı çizgiler gibi bazı ilginç efektler elde etmek için kullanılabilir. İşte, bunu noktalı çizgili bir çokgen çizmek için kullanan bir program.

```
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
```

Yine, bu programda da iki döngü var. İçteki döngü, tek bir noktalı çizgi çizer, buna karşılık dıştaki döngü kaç çizgi çizileceğini belirtir. Örneğimizde, **sides** değişkeni için 6 sayısını kullandık ve böylece aşağıdaki şekilde, noktalı çizgili bir altıgen elde ettik.



Şekil 43 – PenUp ve PenDown işlemlerinin kullanılması

Altyordamlar

Program yazarken, sık sık aynı adım setini defalarca tekrar uygulamamız gerekecek durumlar olacaktır. Bu tip durumlarda, aynı ifadeleri birden fazla kez yazmak muhtemelen çok da anlamlı olmayacaktır. İşte burada *Altyordamlar* faydalı olurlar.

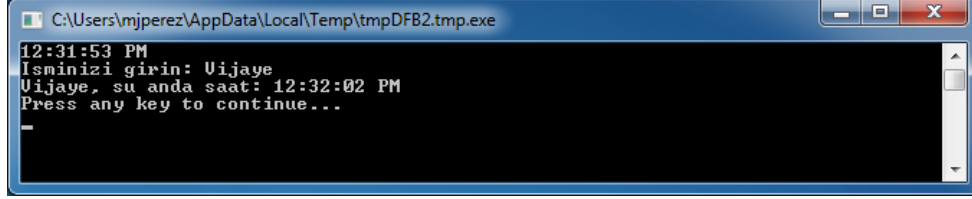
Bir altyordam, daha büyük bir program içerisindeki kodun genellikle çok özel bir şey yapan ve programdaki herhangi bir yerden çağırılabilen bir bölümdür. Altyordamlar, **Sub** anahtar kelimesinin ardından gelen bir isimle tanımlanırlar ve **EndSub** anahtar kelimesiyle sonlandırılırlar. Örneğin; aşağıdaki küçük parça ismi *PrintTime* olan bir altyordamı temsil etmektedir ve bu altyordam, o andaki zamanı *TextWindow*'a yazdırma işini yapar.

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

Aşağıda, altyordam içeren ve onu çeşitli yerlerden çağırın bir program yer almaktadır.

```
PrintTime()
TextWindow.Write("İsminizi girin: ")
name = TextWindow.Read()
TextWindow.Write(name + ", şu anda saat: ")
PrintTime()
```

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



Şekil 44 – Basit bir Altyordamın çağırılması

*SubroutineName()*ı çağırarak, bir altyordamı uygularsınız. Her zaman olduğu gibi, bilgisayara bir altyordamı uygulamak istediğinizi söylemek için, “()” noktalama işaretleri gereklidir.

Altyordamları kullanmanın avantajları

Biraz önce yukarıda gördüğümüz gibi, altyordamlar yazmanız gereken kod miktarını azaltırlar. *PrintTime* altyordamını bir kez yazdığınızda, onu programınız içerisindeki herhangi bir yerden çağırabilirsiniz ve çağırdığınızda o andaki zamanı yazdıracaktır.

Buna ek olarak, altyordamlar karmaşık problemlerin daha basit parçalara ayrılmasına da yardımcı olabilirler. Diyelim ki çözmeniz gereken karmaşık bir eşitlik var, bu karmaşık eşitliğin daha küçük parçalarını çözen çeşitli altyordamlar yazabilirsiniz. Daha sonra, orijinal karmaşık eşitliğin çözümünü elde etmek için, bunları bir araya getirebilirsiniz.

Ayrıca altyordamlar, bir programın okunabilirliğinin iyileştirilmesine de yardımcı olabilirler. Diğer bir deyişle, programınızın ortak olarak çalışan bölümleri için iyi isimlendirilmiş altyordamlarınız varsa, programınızın okunması ve kavranması daha kolay hale gelir. Bu, bir başka kişinin yazdığı programı anlamak istediğinizde veya programınızın başkaları tarafından anlaşılmasını istediğinizde çok önemlidir. Bazen bu, kendi programınızı mesela yazdıktan bir hafta sonra okumak istediğinizde de faydalıdır.

Unutmayın ki; bir SmallBasic altyordamını yalnızca aynı program içerisinde çağırabilirsiniz. Bir altyordamı başka bir program içerisinde çağıramazsınız.

Değişkenlerin kullanılması

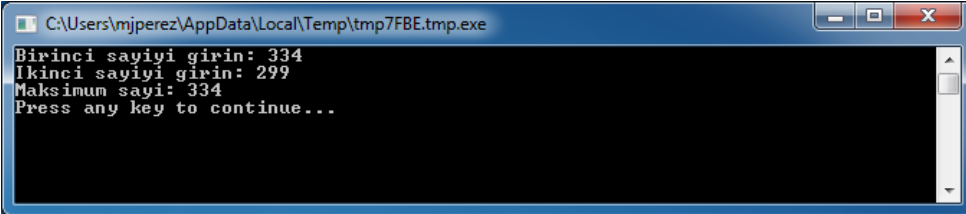
Bir programdaki herhangi bir değişkene, bir altyardam içerisinde erişebilir ve onu kullanabilirsiniz. Bir örnek olarak, aşağıdaki program iki sayıyı kabul eder ve ikisinden büyük olanı yazdırır. *max* değişkeninin altyardamın hem içinde, hem de dışında kullanıldığına dikkat edin.

```
TextWindow.Write("Birinci sayıyı girin: ")
num1 = TextWindow.ReadNumber()
TextWindow.Write("İkinci sayıyı girin: ")
num2 = TextWindow.ReadNumber()

FindMax()
TextWindow.WriteLine("Maksimum sayı: " + max)

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

Ve bu programın çıktısı şu olacaktır.

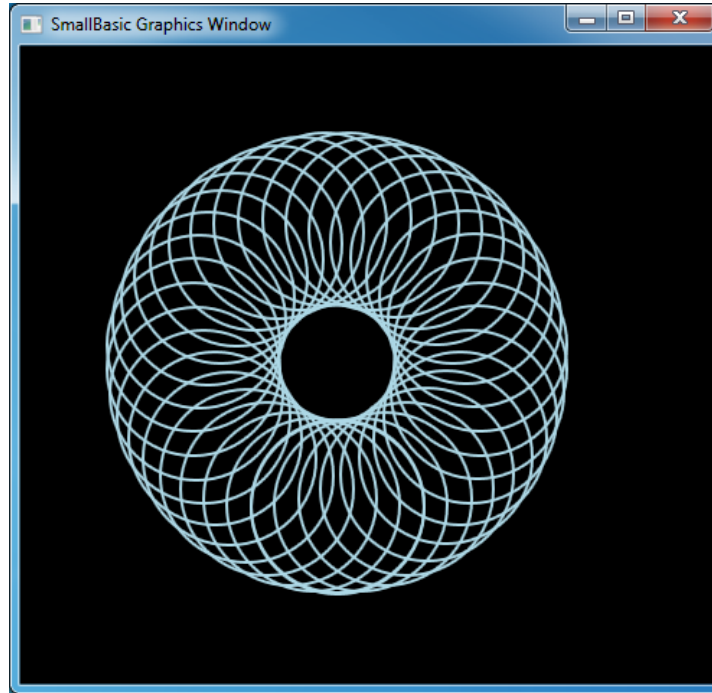


```
C:\Users\mjiperez\AppData\Local\Temp\tmp7FBE.tmp.exe
Birinci sayıyı girin: 334
İkinci sayıyı girin: 299
Maksimum sayı: 334
Press any key to continue...
```

Şekil 45 – Altyordam kullanılarak iki sayının büyüğü

Şimdi, Altyordamların kullanımını gösterecek bir başka örneğe bakalım. Bu defa, x ve y değişkenlerinde saklayacak çeşitli noktaları hesaplayan bir grafik programı kullanacağız. Program daha sonra, merkez olarak x ve y'yi kullanarak bir daire çizmekten sorumlu olan **DrawCircleUsingCenter** isimli bir altyordam kullanacak.

```
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
```



Şekil 46 – Altyordamlar İçin Grafik Örnekleri

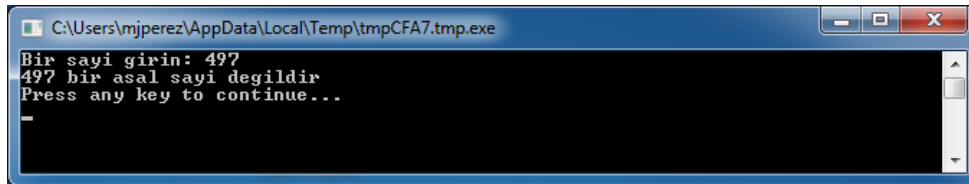
Döngüler içerisinde Altyordamların çağırılması

Bazen, altyordamlar bir döngünün içinden çağırılırlar, bu sırada aynı ifade setini uygularlar, ancak bunu bir ya da daha fazla değışkende farklı değerlerle yaparlar. Örneğin; diyelim ki, *PrimeCheck* isimli bir altyordamınız var ve bu altyordam, bir sayının asal olup olmadığını belirliyor. Kullanıcıya bir değer girmesini söyleyen bir program yazabilir ve sonra bu altyordamı kullanarak, onun asal olup olmadığını söyleyebilirsiniz. Aşağıdaki program bunu göstermektedir.

```
TextWindow.Write("Bir sayı girin: ")
i = TextWindow.ReadNumber()
isPrime = "True"
PrimeCheck()
If (isPrime = "True") Then
    TextWindow.WriteLine(i + " bir asal sayıdır")
Else
    TextWindow.WriteLine(i + " bir asal sayı değildir")
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
```

PrimeCheck altyordamı, *i* değerini alır ve bunu daha küçük sayılara bölmeye çalışır. Eğer bir sayı *i*'ye bölünür ve kalanı olmazsa, bu durumda *i* bir asal sayı değildir. Bu noktada, altyordam *isPrime*'in değerini "False" olarak belirler ve çıkar. Sayı daha küçük sayılara bölünemezse, bu durumda *isPrime*'in değeri "True" olarak kalır.



Şekil 47 – Asal Kontrolü

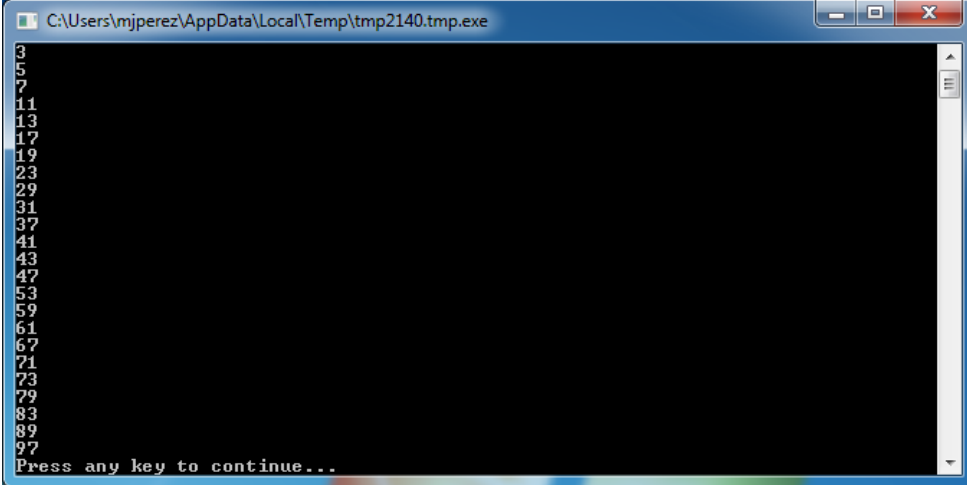
Artık bizim için Asal kontrolünü yapacak bir altyordamınız olduğuna göre, bunu diyelim ki, 100'ün altındaki tüm asal sayıları listelemek için kullanabilirsiniz. Yukarıdaki programı değiştirmek ve bir döngü

içinden *PrimeCheck*'i çağırmasını sağlamak gerçekten kolaydır. Bu, altıyordama döngü her çalıştığında hesaplaması için farklı bir değer verir. Aşağıdaki örnekle bunun nasıl yapıldığını görelim.

```
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
```

Yukarıdaki programda, *i* değeri döngü her çalıştığında güncellenir. Döngünün içinde, *PrimeCheck* altıyordamı için bir çağrı yapılır. Daha sonra, *PrimeCheck* altıyordamı *i* değerini alır ve *i*'nin bir asal sayı olup olmadığını hesaplar. Sonuç, daha sonra altıyordamın dışındaki döngü tarafından erişilen *isPrime* değişkeninde saklanır. Daha sonra, asal sayı olduğunun anlaşılması durumunda, *i*'nin değeri yazdırılır. Ve döngü 3'ten başlayıp 100'e kadar çıktığı için, 3 ile 100 arasındaki tüm asal sayıların bir listesini elde ederiz. Aşağıda programın sonucu yer almaktadır.



```
C:\Users\vmjperz\AppData\Local\Temp\tmp2140.tmp.exe
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
Press any key to continue...
```

Şekil 48 – Asal Sayılar

Şu ana kadar, size değişkenlerin kullanımı konusunda detaylı bilgiler verdik – ne de olsa bu noktaya kadar geldiniz ve hala eğleniyorsunuz, öyle değil mi?

Bir an için, değişkenlerle yazdığımız ilk programa geri dönelim:

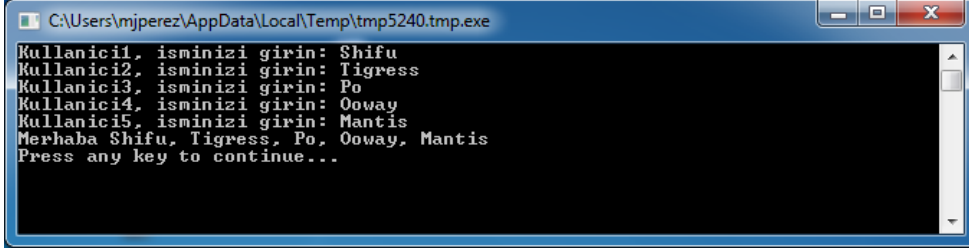
```
TextWindow.Write("İsminizi Girin: ")
name = TextWindow.Read()
TextWindow.WriteLine("Merhaba " + name)
```

Bu programda, kullanıcının ismini **name** isimli bir değişkene aldık ve orada sakladık. Daha sonra kullanıcıya “Merhaba” dedik. Şimdi, diyelim ki; birden fazla kullanıcı var – mesela 5 kullanıcı. Hepsinin ismini nasıl saklayabiliriz? Bunu yapmanın bir yolu şudur:

```
TextWindow.Write("Kullanıcı1, isminizi girin: ")
name1 = TextWindow.Read()
TextWindow.Write("Kullanıcı2, isminizi girin: ")
name2 = TextWindow.Read()
TextWindow.Write("Kullanıcı3, isminizi girin: ")
name3 = TextWindow.Read()
TextWindow.Write("Kullanıcı4, isminizi girin: ")
name4 = TextWindow.Read()
TextWindow.Write("Kullanıcı5, isminizi girin: ")
name5 = TextWindow.Read()
```

```
TextWindow.Write("Merhaba ")
TextWindow.Write(name1 + ", ")
TextWindow.Write(name2 + ", ")
TextWindow.Write(name3 + ", ")
TextWindow.Write(name4 + ", ")
TextWindow.WriteLine(name5)
```

Bu programı çalıştırdığınızda, aşağıdaki sonucu göreceksiniz:



```
C:\Users\mjpervez\AppData\Local\Temp\tmp5240.tmp.exe
Kullanici1, isminizi girin: Shifu
Kullanici2, isminizi girin: Tigress
Kullanici3, isminizi girin: Po
Kullanici4, isminizi girin: Ooway
Kullanici5, isminizi girin: Mantis
Merhaba Shifu, Tigress, Po, Ooway, Mantis
Press any key to continue...
```

Şekil 49 – Diziler kullanılmadan

Şüphesiz, bu kadar basit bir programı yazmanın daha iyi bir yolu olmalı, değil mi? Özellikle bilgisayarlar tekrarlanan görevleri yapmakta oldukça başarılı olduklarından, her bir yeni kullanıcı için aynı kodu neden tekrar tekrar yazmakla uğraşalım ki? Buradaki hile, aynı değişkeni kullanarak, birden fazla kullanıcının ismini saklamak ve geri çağırmaktır. Eğer bunu yapabilirsek, bu durumda daha önceki bölümlerde öğrendiğimiz bir **For** döngüsü kullanabiliriz. Burada diziler imdadımıza yetişir.

Bir dizi nedir?

Dizi, bir defada birden fazla değeri tutabilen özel bir değişkendir. Temel olarak anlamı şudur; beş kullanıcı adını saklamak için **name1**, **name2**, **name3**, **name4** ve **name5** yaratmak yerine, beş kullanıcı isminin tümünü saklaması için **name**'i kullanabiliriz. Birden fazla değeri saklamanın yolu, "indeks" denilen şeyi kullanmaktır. Örneğin; **name[1]**, **name[2]**, **name[3]**, **name[4]** ve **name[5]**'in her biri bir değer saklayabilir. 1, 2, 3, 4 ve 5 sayıları, dizi için *indeks* olarak adlandırılır.

name[1], **name[2]**, **name[3]**, **name[4]** ve **name[5]** farklı değişkenler gibi görünse de, gerçekte bunların hepsi yalnızca tek bir değişkendir. Ve bunun avantajı ne diye sorabilirsiniz. Değerleri bir dizide saklamanın en iyi yanı, bir başka değişkeni kullanarak indeksi belirtebilmenizdir – bu da, döngülerin içerisindeki dizilere kolayca erişebilmemizi sağlar.

Şimdi, bir önceki programımızı dizilerle yazarak, yeni öğrendiğimiz bilgiyi nasıl kullanabileceğimize bir bakalım.

```

For i = 1 To 5
    TextWindow.Write("Kullanıcı" + i + ", isminizi girin: ")
    name[i] = TextWindow.Read()
EndFor

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

```

Okuması çok daha kolay, öyle değil mi? Koyu renkte yazılmış iki satıra dikkat edin. Bunlardan birincisi; bir değeri dizide saklar ve ikicisi de onu diziden okur. **name[1]**'de sakladığınız değer, **name[2]**'de sakladığınızdan etkilenmeyecektir. Dolayısıyla, çoğu amaç için, **name[1]** ve **name[2]**'ye aynı özdeşlik içerisindeki iki farklı değişken gibi davranabilirsiniz.

```

C:\Users\mjperrez\AppData\Local\Temp\tmpA59E.tmp.exe
Kullanici1, isminizi girin: Shifu
Kullanici2, isminizi girin: Tigress
Kullanici3, isminizi girin: Po
Kullanici4, isminizi girin: Ooway
Kullanici5, isminizi girin: Mantis
Merhaba Shifu, Tigress, Po, Ooway, Mantis,
Press any key to continue...

```

Şekil 50 – Diziler kullanılarak

Yukarıdaki program, *Mantis*'in sonundaki virgül haricinde, diziler olmayan program ile neredeyse aynı sonucu verir. Yazdırma döngüsünü yeniden yazarak bunu düzeltebiliriz:

```

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

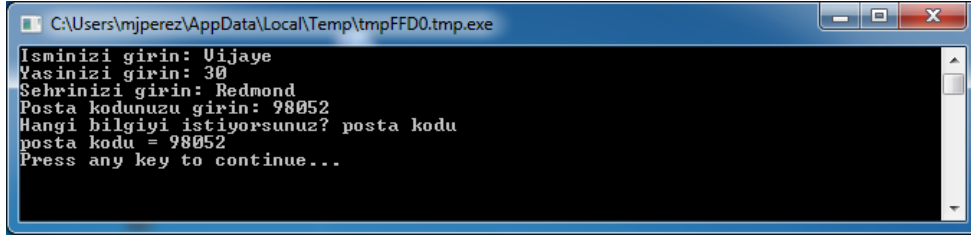
```

Bir dizinin indekslenmesi

Bir önceki programımızda, diziye değerleri saklayıp, sonra da geri çağırmak için indeks olarak sayıları kullandık. İndeksler yalnızca sayılarla sınırlı değildir ve uygulamada, metinsel indeksler kullanmak da oldukça avantajlıdır. Örneğin; aşağıdaki programda, bir kullanıcı hakkındaki çeşitli bilgi parçalarını soruyor ve saklıyoruz ve sonra da kullanıcının istediği bilgileri yazdırıyoruz.

```
TextWindow.Write("İsminizi girin: ")
user["isim"] = TextWindow.Read()
TextWindow.Write("Yaşınızı girin: ")
user["yaş"] = TextWindow.Read()
TextWindow.Write("Şehrinizi girin: ")
user["şehir"] = TextWindow.Read()
TextWindow.Write("Posta kodunuzu girin: ")
user["posta kodu"] = TextWindow.Read()

TextWindow.Write("Hangi bilgiyi istiyorsunuz? ")
index = TextWindow.Read()
TextWindow.WriteLine(index + " = " + user[index])
```



Şekil 51 – Sayısal olmayan indekslerin kullanılması

Birden fazla boyut

Diyelim ki; tüm arkadaşlarınızın isimlerini ve telefon numaralarını saklamak ve daha sonra da ihtiyaç duyduğunuzda telefon numaralarına ulaşmak istiyorsunuz – yani telefon defteri gibi bir şey. Bu tip bir programı nasıl yazabiliriz?

Bu durumda, ilgili iki indeks seti vardır (bunlar dizinin boyutları olarak da bilinirler). Her bir arkadaşınızı takma isimle tanımladığımızı varsayalım. Bu, bizim dizideki ilk indeksimiz olur. Arkadaş değişkenimizi elde etmek için ilk indeks kullandıktan sonra, indekslerin ikincisi, **isim** ve **telefon numarası** o arkadaşın gerçek ismine ve telefon numarasına ulaşmamızı sağlayacaktır.

Dizi indeksleri büyük/küçük harfe duyarlı değildir. Tıpkı normal değişkenler gibi, dizi indekslerinin de tam olarak büyük/küçük harfe uyması gerekmez.

Bu verileri saklama şeklimiz şöyle olacaktır:

```
friends["Rob"]["İsim"] = "Robert"
friends["Rob"]["Telefon"] = "555-6789"

friends["VJ"]["İsim"] = "Vijaye"
friends["VJ"]["Telefon"] = "555-4567"

friends["Ash"]["İsim"] = "Ashley"
friends["Ash"]["Telefon"] = "555-2345"
```

Aynı **friends** dizisinde iki indeksimiz olduğuna göre, bu dizi iki boyutlu dizi olarak adlandırılır.

Bu programı bir kez düzenledikten sonra, girdi olarak bir arkadaşımızın takma ismini alabilir ve sonra o kişi hakkında sakladığımız bilgileri yazdırabiliriz. İşte bunu yapan programın tümü:

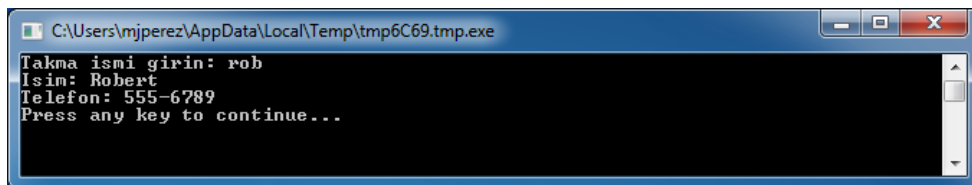
```
friends["Rob"]["İsim"] = "Robert"
friends["Rob"]["Telefon"] = "555-6789"

friends["VJ"]["İsim"] = "Vijaye"
friends["VJ"]["Telefon"] = "555-4567"

friends["Ash"]["İsim"] = "Ashley"
friends["Ash"]["Telefon"] = "555-2345"

TextWindow.Write("Takma ismi girin: ")
nickname = TextWindow.Read()

TextWindow.WriteLine("İsim: " + friends[nickname]["İsim"])
TextWindow.WriteLine("Telefon: " + friends[nickname]["Telefon"])
```



Şekil 52 – Basit bir telefon defteri

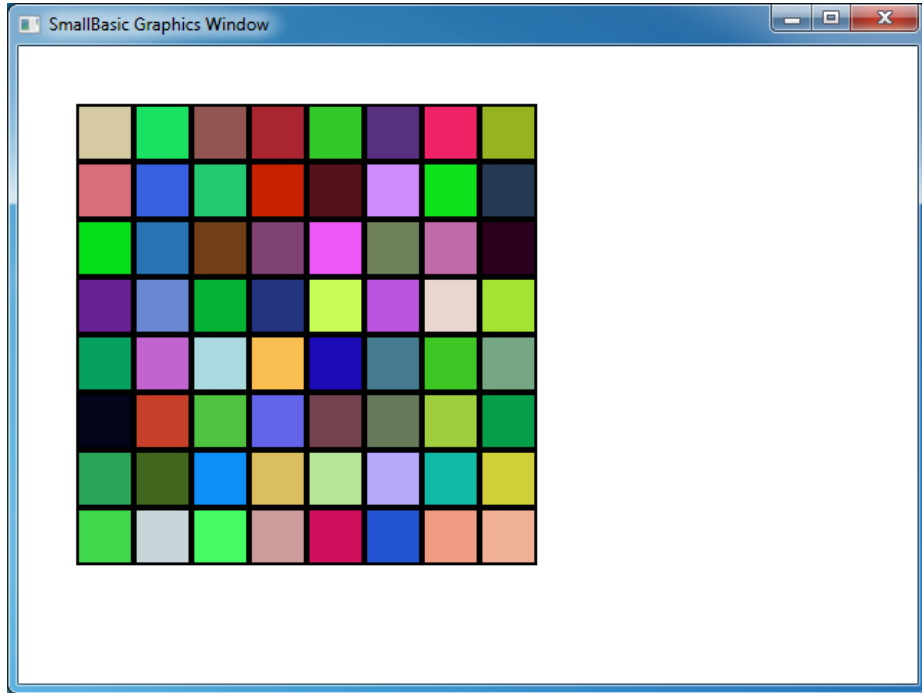
Tabloları temsil etmesi için Dizileri kullanmak

Çok boyutlu dizilerin oldukça yaygın bir kullanımı, tabloları temsil etmektir. Tabloların, iki boyutlu bir diziye çok iyi bir şekilde uyan sıraları ve kolonları vardır. Aşağıda, bir tabloya kutular yerleştiren bir program verilmiştir:

```
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
```

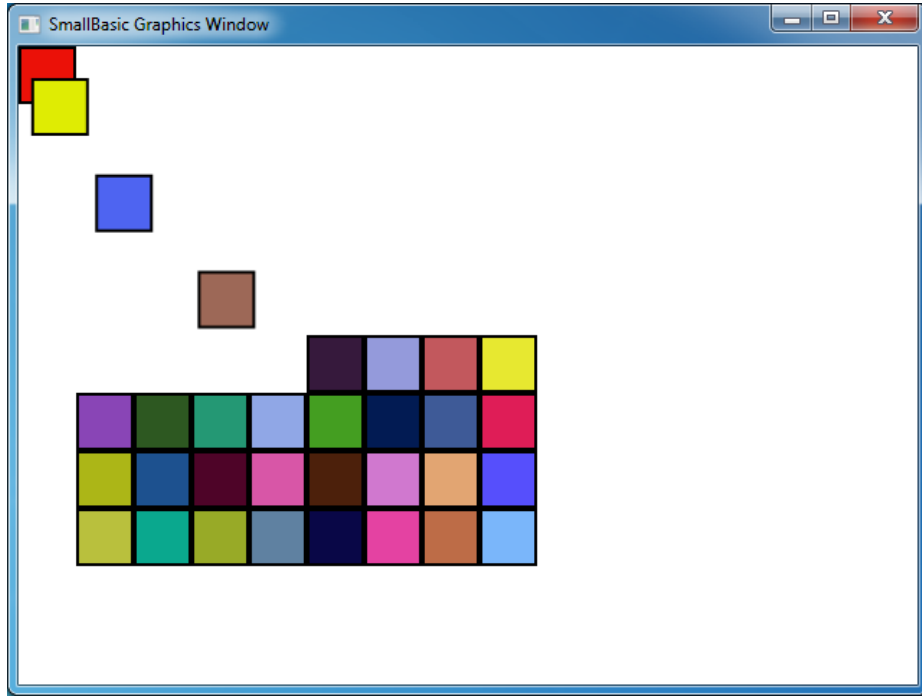
Bu program kareler ekler ve onları 8x8 bir tablo oluşturacak şekilde konumlandırır. Bu kutuları yerleştirmenin yanında, ayrıca onları bir dizi halinde de saklar. Bunu yapması, kutuları izlememizi ve gerektiğinde tekrar kullanmamızı kolaylaştırır.



Şekil 53 – Kutuları bir tabloya yerleştirmek

Örneğin; bir önceki programın sonuna aşağıdaki kodun eklenmesi, bu kutuların sol üst köşeye gitmesine neden olacaktır.

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



Şekil 54 – Tablodaki kutuların izlenmesi

Olaylar ve Etkileşim

İlk iki bölümde, *Özellikleri ve İşlemleri* olan nesnelere tanıtık. Özelliklere ve işlemlere ek olarak, bazı nesnelere *Olayları* da vardır. Olaylar, örneğin; fareyi hareket ettirmek veya üzerine tıklamak gibi kullanıcı eylemlerine karşı verilen sinyaller gibidir. Bir anlamda, olaylar işlemlerin zıddıdır. İşlemler söz konusu olduğunda, bir programcı olarak siz bilgisayara bir şeyler yaptırabilirsiniz, olaylar söz konusu olduğunda ise, bilgisayar ilginç bir şey olduğunda size haber verir.

Olaylar nasıl faydalıdır?

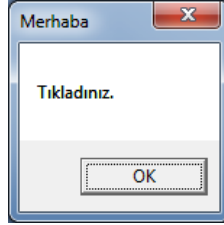
Olaylar, bir programa etkileşim katmanının merkezidir. Bir kullanıcının programınızla etkileşime girmesini isterseniz, olayları kullanırsınız. Diyelim ki; bir Üç Taş oyunu yazıyorsunuz. Kullanıcının hamlesine karar vermesine izin vermek isteyeceksiniz, değil mi? İşte burada olaylar işin içine girer – olayları kullanarak programınız içerisinden girdi alırsınız. Bunu kavraması zor gibi görünüyorsa, endişelenmeyin; olayların ne olduğunu ve nasıl kullanılabildiklerini anlamaya yardımcı olacak çok basit bir örneğe bir göz atacağız.

Aşağıda, yalnızca tek bir ifadeden ve bir altyordamdan oluşan oldukça basit bir program yer alıyor. Altyordam, kullanıcıya bir mesaj kutusu göstermek için, GraphicsWindow nesnesinde *ShowMessage* işlemini kullanır.

```
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
    GraphicsWindow.ShowMessage("Tıkladınız.", "Merhaba")
EndSub
```

Yukarıdaki programda not edilmesi gereken ilginç bölüm, GraphicsWindow nesnesinin **MouseDown** olayına altyordam ismi atadığımız satırdır. MouseDown tıpkı bir özellik gibi görünmektedir – yalnızca, bir değer atamak yerine, ona *OnMouseDown* altyordamını atıyoruz. Olaylar hakkında özel olan şey budur – olay gerçekleştiğinde, altyordam otomatik olarak çağırılır. Bu durumda, kullanıcı fareyi kullanarak her tıkladığında, GraphicsWindow üzerinde *OnMouseDown* altyordamı çağırılır. Hadi programı çalıştırın ve deneyin. Fareyle GraphicsWindow üzerine her tıkladığınızda, aşağıdaki pencerede gösterilen gibi bir mesaj kutusu göreceksiniz.

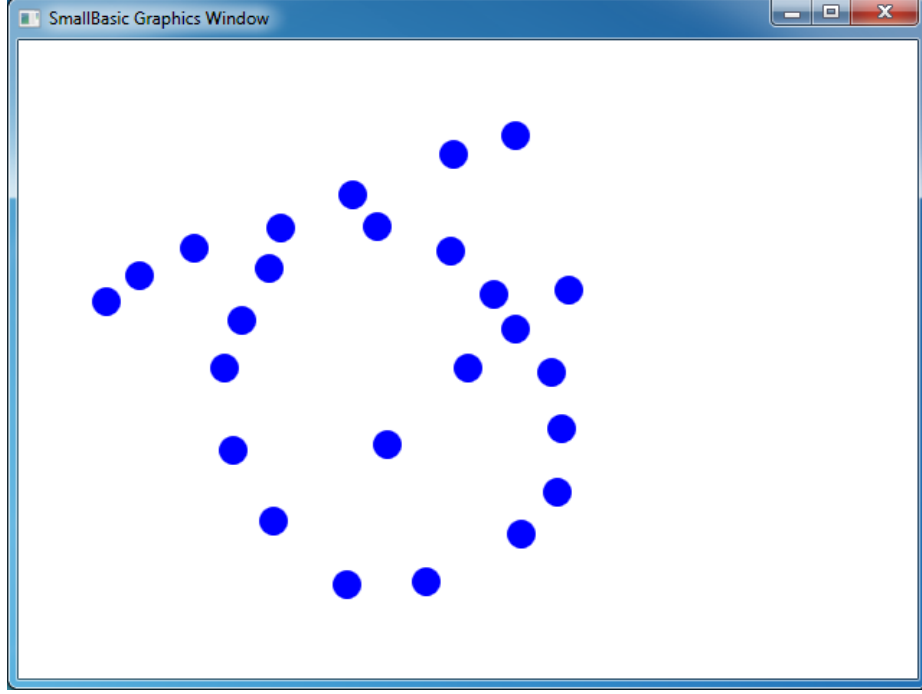


Şekil 55 – Bir olaya yanıt

Olayların bu şekilde kullanılması oldukça güçlüdür ve oldukça yaratıcı ve ilginç programların yazılabilmesini sağlar. Bu şekilde yazılmış programlar, genellikle olay güdümlü programlar olarak adlandırılırlar.

OnMouseDown altyordamını, bir mesaj kutusu açmak yerine başka şeyler yapmak üzere değiştirebilirsiniz. Örneğin; aşağıdaki programda olduğu gibi, kullanıcının fareyi tıkladığı yerlere büyük mavi noktalar çizebilirsiniz.

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



Şekil 56 – Fareye Tıklama Olayının Kullanılması

Yukarıdaki programda, farenin koordinatlarını elde etmek için *MouseX* ve *MouseY*'yi kullandığımıza dikkat edin. Daha sonra bunları, farenin koordinatlarını dairenin merkezi olarak alarak, bir daire çizmek için kullanıyoruz.

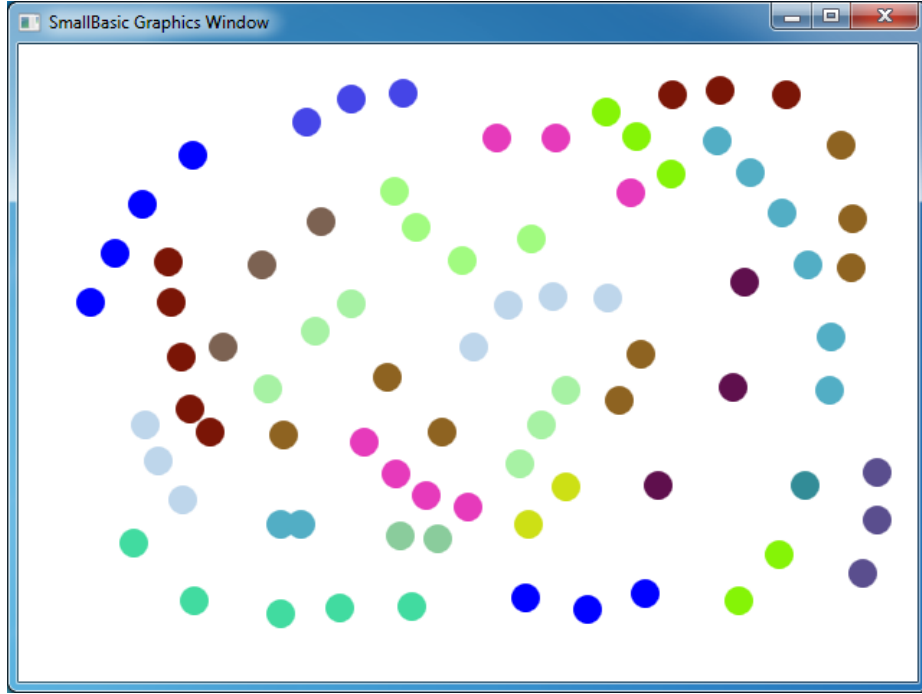
Birden fazla olayın kullanılması

Ne kadar olayı kullanmak istediğinizle ilgili gerçekte bir sınır yoktur. Hatta, birden fazla olayı kullanacak bir altyordama bile sahip olabilirsiniz. Ancak, bir olayı yalnızca bir kez kullanabilirsiniz. Aynı olaya iki altyordam atamaya çalışırsanız, ikincisi galip gelir.

Bunu göstermek için, bir önceki örneği alalım ve tuşlara basmayı kullanan bir altyordam ekleyelim. Ayrıca, bu yeni altyordama fırçanın rengini değiştirelim, böylece fareye tıkladığınızda farklı bir renkte nokta elde edin.

```
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
```



Şekil 57 – Birden fazla olayın kullanılması

Bu programı çalıştırır ve pencereye tıklarsanız, mavi bir nokta çıkacaktır. Şimdi, herhangi bir tuşa bir kez basar ve sonra yine tıklarsanız, farklı renkte bir nokta çıkacaktır. Bir tuşa bastığınızda, fırçanın rengini rasgele bir renge değiştiren *OnKeyDown* altyordamı uygulanır. Bundan sonra fareye tıkladığınızda, yeni belirlenen renk kullanılarak bir daire çizilir ve böylece ortaya rasgele renkli noktalar çıkar.

Bir boyama programı

Olaylarla ve altyordamlarla kuşatıldıktan sonra, artık kullanıcıların pencerede çizim yapmasına izin veren bir program yazabiliriz. Bu tip bir program yazmak, programı daha küçük parçalara ayırmamız koşuluyla, şaşırtıcı derecede kolaydır. İlk adım olarak, kullanıcıların fareyi grafik penceresinde herhangi bir yere hareket ettirmesine izin verecek ve fare hareket ettirildiğinde bir iz bırakacak bir program yazalım.

```
GraphicsWindow.MouseMove = OnMouseMove

Sub OnMouseMove
  x = GraphicsWindow.MouseX
  y = GraphicsWindow.MouseY
```

```
GraphicsWindow.DrawLine(prevX, prevY, x, y)
prevX = x
prevY = y
EndSub
```

Ancak, bu programı çalıştırdığınızda, ilk satır daima pencerenin sol üst kenarından (0, 0) başlar. *MouseDown* olayını kullanarak ve o olay geldiğinde *prevX* ve *prevY* değerlerini alarak bu sorunu giderebiliriz.

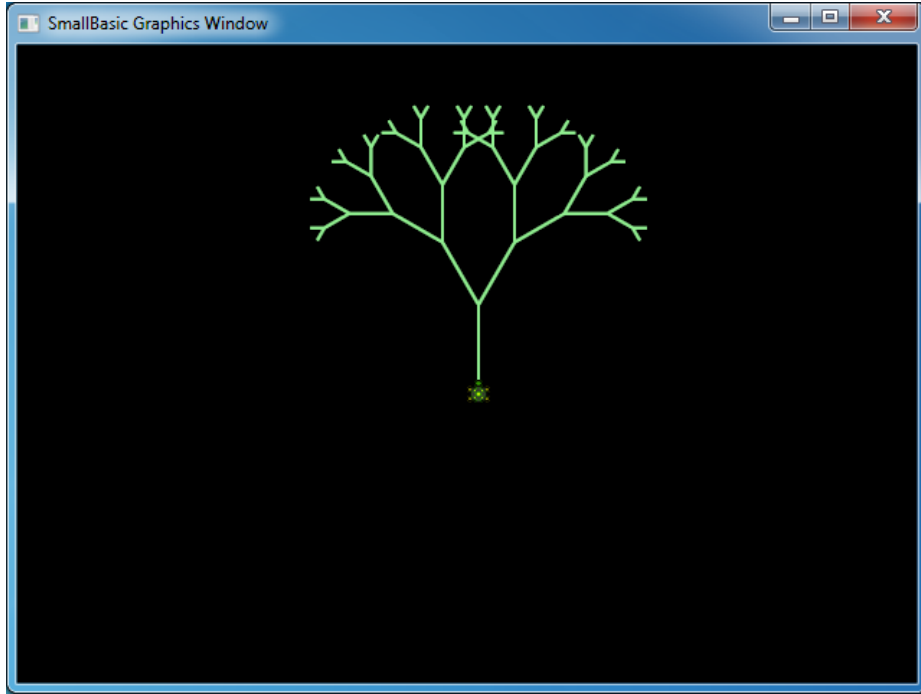
Ayrıca, ize yalnızca kullanıcı fare düğmesini basılı tuttuğunda ihtiyacımız vardır. Diğer zamanlarda, çizgiyi çizmememiz gerekir. Bu davranışı elde etmek için, **Mouse** nesnesinde *IsLeftButtonDown* özelliğini kullanacağız. Bu özellik, Sol düğmesine basılı tutulup tutulmadığını söyler. Bu değer doğruysa, çizgiyi çizeriz, değilse çizmeyiz.

```
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
```


Benzer Şekilleri Kullanan Kurbağa



Şekil 58 – Benzer şekilleri kullanarak bir ağaç çizen kurbağa

```
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("mesafe", distance)  
    distance = distance - delta  
    DrawTree()  
    Turtle.Turn(-angle * 2)  
    DrawTree()  
    Turtle.Turn(angle)  
    distance = Stack.PopValue("mesafe")  
  
    Turtle.Move(-distance)  
  EndIf  
EndSub
```

Flickr'dan Fotoğraflar



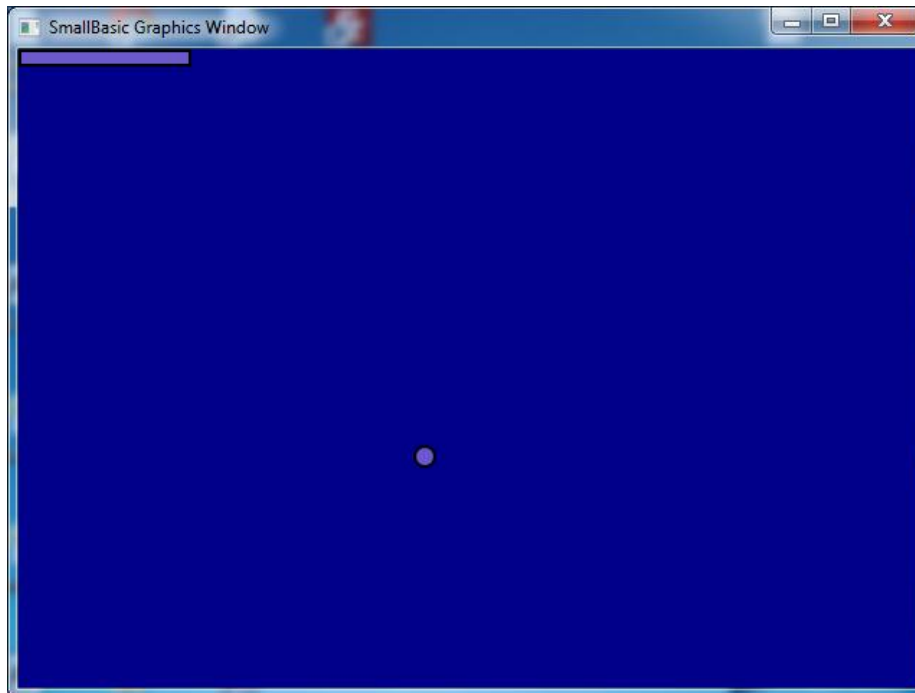
Şekil 59 – Flickr'dan fotoğraflar almak

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

Dinamik Masaüstü Duvar Kağıdı

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

Raket Oyunu



Şekil 60 – Raket Oyunu

```
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("Kaybettiniz", "Raket")

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

```

Renkler

Aşağıda Small Basic tarafından desteklenen isimlendirilmiş ve zemin rengine göre gruplandırılmış renklerin bir listesini bulacaksınız.

Kırmızı Renkler

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

Pembe Renkler

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

Turuncu Renkler

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

Sarı Renkler

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

PaleGoldenrod	#EEE8AA
Khaki	#F0E68C
DarkKhaki	#BDB76B

Mor Renkler

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

Yeşil Renkler

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

Mavi Renkler

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

Kahverengi Renkler

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

Beyaz Renkler

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

Gri Renkler

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