De naslaggids voor ontwikkelaars voor Small Basic: Hoofdstuk 3: Programmaobject

Voorbeeld

In dit hoofdstuk beginnen we ons overzicht van objecten die worden gebruikt om de Small Basic-programma's te bouwen. Voor elk object vatten we de eigenschappen, methoden en gebeurtenissen samen. Vervolgens bouwen we verschillende voorbeeldprogramma's die het gebruik van het object illustreren. We beginnen met het object Small Basic **Program**.

Programma object

Het **object Program** (of beter gezegd klasse) helpt bij de uitvoering van het programma. We gebruiken het om te identificeren in welke map uw programma is opgeslagen, vertragingen te implementeren en het programma te stoppen.

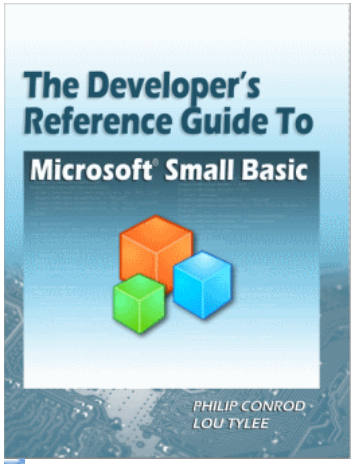
Eigenschappen van het programma:

Directory
Hiermee wordt de map van het uitvoerende programma opgehaald.

Programma methoden:

Vertraging (milliseconden)
Vertraagt de uitvoering van het programma met het opgegeven aantal **milliseconden**.

End()
Beëindigt het programma.



Dit hoofdstuk is een bewerking van het boek *The Developer's Reference Guide To Microsoft Small Basic* van Philip Conrod en Lou Tylee.

Om dit boek in zijn geheel te kopen, zie de Computer Science For Kids website.

Voorbeeld 3-1. Programma Directory

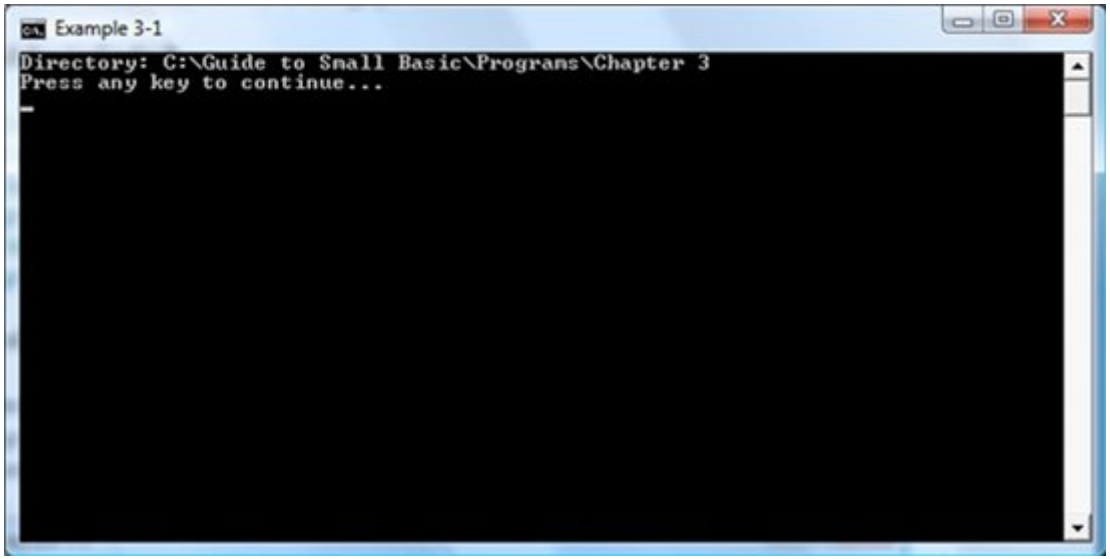
Schrijf een programma waarin de map (map) wordt weergegeven waarin uw programma is opgeslagen.

Kleine basiscode:

```
1. ' Guide to Small Basic, Example 3-1
2. TextWindow.Title = "Example 3-1"
3. TextWindow.WriteLine("Directory: " + Program.Directory)
```

Opgeslagen als **voorbeeld 3-1** in **de map Guide to Small Basic\Programs\Chapter 3**.

Sla het programma op en **voer** het uit. De programmamap wordt geschreven in het tekstvenster:



Uw map zal anders zijn, ervan uitgaande dat u uw programma hebt opgeslagen in een map met een andere naam.

Voorbeeld 3-2. Programma vertraging

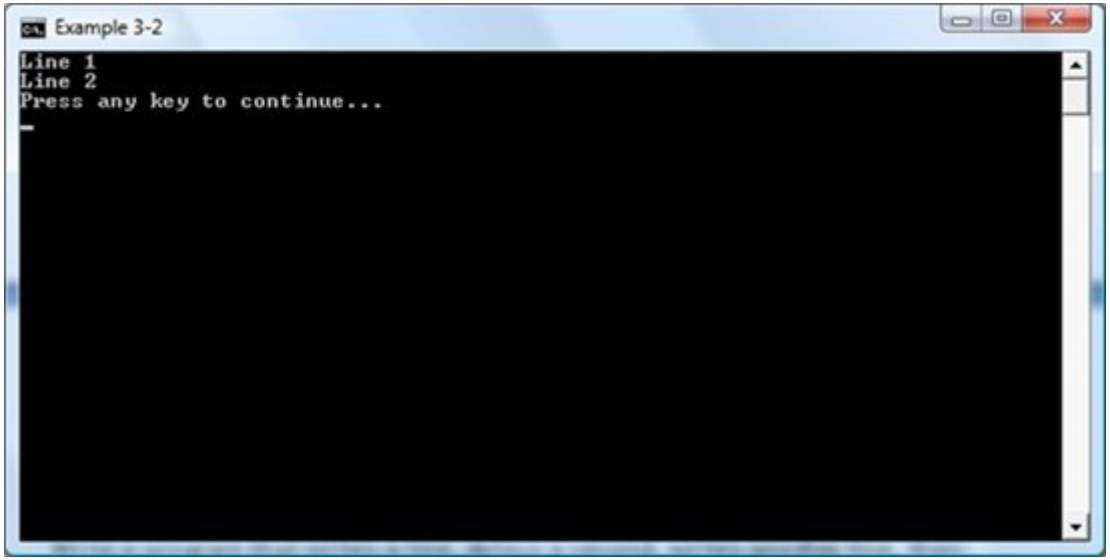
Schrijf een programma dat een regel schrijft, een seconde vertraagt, een andere regel schrijft en vervolgens twee seconden vertraagt voordat het eindigt.

Kleine basiscode:

```
1. ' Guide to Small Basic, Example 3-2
2. TextWindow.Title = "Example 3-2"
3. TextWindow.WriteLine("Line 1")
4. Program.Delay(1000)
5. TextWindow.WriteLine("Line 2")
6. Program.Delay(2000)
```

Opgeslagen als **voorbeeld 3-2** in **de map Guide to Small Basic\Programs\Chapter 3**.

Sla het programma op en **voer** het uit. In het tekstvenster ziet u **regel 1-weergave**, een vertraging van 1 seconde (1000 milliseconden), **lijn 2-weergave** en vervolgens een vertraging van 2 seconden voordat het programma eindigt:



Programma End Methode

In de korte tekstvensterprogramma's die we hebben geschreven, eindigen ze met deze verklaring:

Druk op een willekeurige toets om door te gaan...

Hierdoor kunnen we de inhoud van het venster zien voordat het programma wordt gesloten.

Als u een programma beëindigt met:

Program.End()

De "Druk op een willekeurige toets om door te gaan ..." instructie wordt niet gezien en het tekstvenster wordt gesloten.

Voorbeeld 3-3. Einde programma

Herhaal **voorbeeld 3-2**, maar voeg een **instructie End** toe. Dat wil zeggen, schrijf een programma dat een regel schrijft, een seconde vertraagt, een andere regel schrijft en vervolgens twee seconden vertraagt, voordat het eindigt met een **eindverklaring**.

Kleine basiscode:

```
1. ' Guide to Small Basic, Example 3-3
2. TextWindow.Title = "Example 3-3"
3. TextWindow.WriteLine("Line 1")
4. Program.Delay(1000)
5. TextWindow.WriteLine("Line 2")
6. Program.Delay(2000)
7. Program.End()
```

Opgeslagen als **voorbeeld 3-3** in **de map Guide to Small Basic\Programs\Chapter 3**.

Sla het programma op en **voer** het uit. In het tekstvenster ziet u **regel 1-weergave**, een vertraging van 1 seconde (1000 milliseconden), **regel 2-weergave**, vervolgens een vertraging van 2 seconden, waarna het tekstvenster verdwijnt.

Hoofdstuk Review

Na het voltooien van dit hoofdstuk moet u begrijpen:

- Gebruik van het object **Property.**
- Hoe u uw programmamap kunt identificeren.
- Hoe een programmavertraging te implementeren.
- Hoe een programma te stoppen en het tekstvenster te laten verdwijnen.

Vervolgens bekijken we in meer detail het **TextWindow-object** dat we hebben gebruikt.

Tekenreeksmethoden

Dingen voor tekstmethoden

- Small Basic biedt een krachtige set methoden om te werken met variabelen van het tekenreekstype, die erg belangrijk zijn in Small Basic. Deze methoden zijn gekoppeld aan het object **Text**.

- Om het aantal tekens in (of de lengte van) een tekenreeksvariabele te bepalen, gebruiken we de methode **GetLength**. **MyString** als voorbeeld gebruiken:

```
1. MyString = "Small Basic is fun!"
2. LenString = Text.GetLength(MyString)
```

LenString heeft een waarde van **19**. Tekens in de tekenreeksvariabele beginnen bij index 1 en eindigen bij 19.

- U kunt subtekenreeksen van tekens extraheren. Voor deze taak wordt de methode **GetSubText** gebruikt. U geeft de tekenreeks, de beginpositie en het aantal tekens op dat moet worden geëxtraheerd. Dit voorbeeld begint bij teken 2 en extraheert 6 tekens:

```
1. MyString = "Small Basic is fun!"
2. SubString = Text.GetSubText(MyString, 2, 6)
```

De **substringvariabele** is gelijk aan **"mall B"** Notice u kunt dit gebruiken om van 1 naar zoveel tekens te extraheren als u wilt.

- Misschien wil je gewoon een uiterst links deel van een string. Gebruik de methode **GetSubText** met een beginpositie van 1. In dit voorbeeld worden de 3 meest linkse tekens uit een tekenreeks gehaald:

```
1. MyString = "Small Basic is fun!"
2. LeftString = Text.GetSubText(MyString, 1, 3)
```

De variabele **LeftString** is gelijk aan **"Sma"**

- Als u het uiterst rechtse gedeelte van een tekenreeks wilt ophalen, gebruikt u de methode **GetSubTextToEnd**. Geef het teken op waarmee u wilt beginnen en het juiste gedeelte van de tekenreeks wordt geretourneerd. Om 6 tekens aan het einde van ons voorbeeld te krijgen, gebruikt u:

```
1. MyString = "Small Basic is fun!"
2. RightString = Text.GetSubTextToEnd(MyString, 13)
```

De **RightString** variabele is gelijk aan **"s fun!"**

- Vaak wilt u letters converteren naar hoofdletters of omgekeerd. Small Basic biedt hiervoor twee methoden: **ConvertToUpperCase** en **ConvertToLowerCase**. De methode ConvertToUpperCase converteert alle letters in een tekenreeksvariabele naar hoofdletters, terwijl de methode ConvertToLowerCase alle letters converteert naar kleine letters. Niet-alfabetische tekens worden genegeerd in de conversie. En als een brief al in het gewenste geval is, wordt deze ongewijzigd gelaten. Voor ons voorbeeld (een beetje aangepast):

```
1. MyString = "Read About Small Basic in 2010!”
2. A = Text.ConvertToUpperCase(MyString)
3. B = Text.ConvertToLowerCase(MyString)
```

De eerste conversie met **ConvertToUpperCase** resulteert in:

A = "LEES OVER SMALL BASIC IN 2010!"

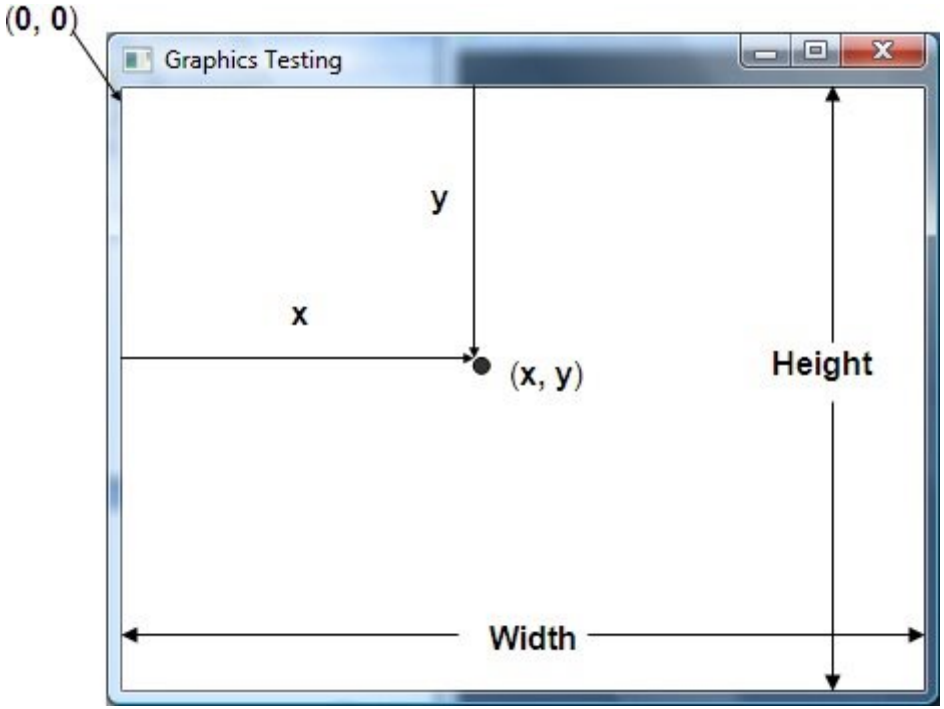
En de tweede conversie met **ConvertToLowerCase** levert op:

B = "lees over kleine basic in 2010!"

Grafische methoden

- Al onze spellen worden "gehost" door het Small Basic graphics venster. Om iets in het grafische venster te plaatsen, moet het daar worden "getekend" met behulp van een van de vele Small Basic-grafische methoden. Er moet zelfs tekst getekend worden! Laten we eens kijken naar deze methoden.

- Het coördinatensysteem dat door het grafische venster wordt gebruikt, is:



  Het venster is **Breedte** pixels breed en **Hoogte** pixels hoog. We gebruiken twee waarden (coördinaten) om een enkele pixel in het venster te identificeren. De **x** (horizontale) coördinaat neemt toe van links naar rechts, beginnend bij **0**. De **y** (verticale) coördinaat neemt toe van boven naar beneden, ook beginnend bij **0**. Punten in het gebied worden aangeduid met de twee coördinaten tussen haakjes, of **(x, y)**.

- Om lijnen en vormen te tekenen, gebruiken we een **pen**. U kunt kleur en breedte kiezen. Coderegels die deze taak uitvoeren, zijn:

```
1. GraphicsWindow.PenColor = Color
2. GraphicsWindow.PenWidth = Width
```

waarbij **Kleur** de kleur is waarin uw pen tekent en **Breedte** de gehele breedte (standaard een waarde van 1) van de getekende lijn (in pixels) is. Deze pen trekt een ononderbroken lijn. Als u een kleur wilt opgeven, gebruikt u een kleurnaam zoals "Rood", "Wit" of "Blauw". Bijlage I geeft een overzicht van de vele kleuren die beschikbaar zijn met Small Basic.

- De Small Basic **DrawLine-methode** wordt gebruikt om twee punten te verbinden met een rechtlijnig segment. Als we het punt (**x1**, **y1**) willen verbinden met (**x2**, **y2**), is de verklaring:

  GraphicsWindow.DrawLine(x1, y1, x2, y2)

  De lijn tekent in de huidige pen **Kleur** en **breedte**. Voorbeeld dat een blauwe lijn van breedte 1 tekent:

```
1. GraphicsWindow.PenColor = "Blue"
2. GraphicsWindow.PenWidth = 1
3. GraphicsWindowGraphicsWindow.DrawLine(20, 50, 380, 280)
```

- De methode Small Basic **DrawRectangle** wordt gebruikt om een rechthoek te tekenen. Om een rechthoek te tekenen, specificeren we de coördinaat van de linkerbovenhoek (**x**, **y**) en de **breedte** en **hoogte** van de rechthoek. Ga als volgt te werk om een dergelijke rechthoek in het afbeeldingsvenster te tekenen:

  GraphicsWindow.DrawRectangle (x, y, breedte, hoogte)

  De rechthoek tekent met de huidige pen. Als u een blauwe rechthoek (penbreedte 2) wilt tekenen met de linkerbovenhoek op (20, 50), breedte 150 en hoogte 100, gebruikt u deze code:

```
1. GraphicsWindow.PenColor = "Blue"
2. GraphicsWindow.PenWidth = 2
3. GraphicsWindow.DrawRectangle(20, 50, 150, 100)
```

- De Small Basic **DrawEllipse-methode** wordt gebruikt om een ellips te tekenen. Om een ellips te tekenen, specificeren we de coördinaat van de linkerbovenhoek (**x**, **y**) en de **breedte** en **hoogte** van de ellips. Ga als volgt te werk om een dergelijke ellips in het grafische venster te tekenen:

  GraphicsWindow.DrawEllipse (x, y, breedte, hoogte)

  De ellips zal tekenen met de huidige pen. Als u een groene ellips (penbreedte 3) wilt tekenen met de linkerbovenhoek op (20, 50), breedte 150 en hoogte 100, gebruikt u deze code:

```
1. GraphicsWindow.PenColor = "Green"
2. GraphicsWindow.PenWidth = 3
3. GraphicsWindow.DrawEllipse(20, 50, 150, 100)
```

- Een **penseel** is als een "brede" pen. Het wordt gebruikt om gebieden met een kleur te vullen. Het heeft een enkele eigenschap (**Kleur**). Als u de penseelkleur wilt instellen, gebruikt u het volgende:

  GraphicsWindow.BrushColor = Kleur

  Een penseel is 'effen' – gebieden volledig vullen met de opgegeven kleur

- Als u rechthoeken en weglatingstekens wilt vullen met de huidige penseelkleur, gebruikt u de methoden **FillRectangle** en **FillEllipse**. Ze hebben dezelfde argumenten als respectievelijk **DrawRectangle** en **DrawEllipse**.

- Om tekstinformatie aan het grafische venster toe te voegen, gebruiken we de Small Basic **DrawText-methode** - ja, tekst wordt naar het venster "getekend". De methode **DrawText** is:

  GraphicsWindow.DrawText(x, y, tekst)

  In deze instructie vertegenwoordigt **tekst** de tekenreeks die in het venster moet worden afgedrukt en het punt (**x, y**) is waar de tekenreeks zich bevindt. De tekenreeks tekent in het afbeeldingsvenster met de huidige penseelkleur met het standaardlettertype. Merk op dat deze methode de penseelkleur gebruikt, niet de penkleur - tekst wordt echt getekend zoals andere grafische objecten.

- Hier is een voorbeeld met **DrawText**:

```
1. GraphicsWindow.BrushColor = "Blue"
2. GraphicsWindow.DrawText(40, 100, "Isn't Small Basic fun?")
```

Dit zet de regel "Is Small Basic fun?" op (40, 100) in het grafische venster. De tekst is blauw van kleur. Door het (x- en y)-punt in te stellen, kunt u de tekst naar links of rechts uitvullen of horizontaal en/of verticaal centreren door de vensterafmetingen te kennen. De tekengrootte kan worden gewijzigd door de eigenschap **FontSize in te** stellen en **FontBold** bepaalt of het lettertype vet is of niet.

Objecten Shapes

- Gerelateerd aan grafische methoden zijn **shapes**objecten. Een **object Shapes** is een rechthoekig gebied dat we kunnen toevoegen, verplaatsen en verwijderen binnen het grafische venster. Zo'n object maakt animatie (bewegende objecten) heel eenvoudig. We kunnen vormen hebben die rechthoeken, ellipsen en zelfs afbeeldingen zijn! Laten we naar elk kijken.

- Als u een rechthoekige vorm (**MyRectangle**) wilt maken die **W-pixels** breed en **H-pixels**hoog is, gebruikt u de methode **AddRectangle**:

MyRectangle = Shapes.AddRectangle(B, H)

Hierdoor ontstaat een 'omzoomde' rechthoek. Met de huidige penkleur en penbreedte wordt de randkleur van de rechthoek vastgesteld, terwijl de huidige penseelkleur de vulkleur bepaalt. Standaard wordt het in de linkerbovenhoek van het grafische venster geplaatst.

- Als u een elliptische vorm (**MyEllipse**) wilt maken die **W-pixels** breed en **H-pixels** hoog is, gebruikt u de methode **AddEllipse**:

  MyEllipse = Shapes.AddEllipse(B, H)

  Hierdoor ontstaat een 'begrensde' ellips. Met de huidige penkleur en penbreedte wordt de ellipsrandkleur vastgesteld, terwijl de huidige penseelkleur de vulkleur bepaalt. Standaard wordt het in de linkerbovenhoek van het grafische venster geplaatst.

- Om een vorm met een afbeelding te maken, hebben we twee stappen nodig. Eerst moeten we de afbeelding laden en vervolgens de vorm maken. Stel dat u **een** jpg-afbeeldingsbestand (een digitale foto) hebt met de naam **MyImage.jpg**. (U kunt ook andere typen afbeeldingsbestanden gebruiken). Het bestand met deze afbeelding moet zich in dezelfde map bevinden als uw Small Basic-programma. De afbeelding wordt geladen met de methode **LoadImage** van het object **ImageList**:

  MyImage = ImageList.LoadImage(Program.Directory + "\MyImage.jpg")

  Vervolgens wordt de afbeeldingsvorm (**MyImageShape**) gemaakt met behulp van:

  MyImageShape = Shapes.AddImage(Mijnimage)

  De vorm wordt in de linkerbovenhoek van het grafische venster geplaatst.

- Het verplaatsen **van shapes-objecten** in een grafisch venster is eenvoudig te doen. Het is een eenvoudig proces in twee stappen: gebruik een regel om een nieuwe positie te bepalen en teken deze vervolgens opnieuw in deze nieuwe positie met behulp van de methode **Shapes** object **Verplaatsen**. Als u een vormobject met de naam **MyShape** hebt en u dit wilt verplaatsen naar (**NewX**, **NewY**), is de code:

  Shapes.Move(MyShape, NewX, NewY)

  Deze code 'wist' **MyShape** op de huidige positie en 'hertekent' het vervolgens op de nieuw opgegeven positie. Opeenvolgende overdrachten (of bewegingen) geven de indruk van beweging of animatie.

- We zullen veel voorbeelden zien van het gebruik van **Shapes-objecten**in de spelprogramma's die we bouwen. We leveren de afbeeldingsbestanden voor **alle shapes** waarvoor afbeeldingen nodig zijn.

Kleine basismethoden

- Small Basic biedt een rijk assortiment ingebouwde **methoden** die verschillende hoeveelheden berekenen of leveren. De algemene vorm van een methode is:

  ReturnedValue = ObjectNaam.MethodName(Arguments)

  waarbij **Arguments** een door komma's gescheiden lijst vertegenwoordigt met informatie die **MethodName** nodig heeft om de berekening uit te voeren. Zodra de argumenten zijn geleverd aan de methode, retourneert deze een waarde (**ReturnedValue**) voor gebruik in een toepassing.

- Some methods do not return a value, but only perform a task. To use these methods, just type:

  ObjectName.MethodName(Arguments)

Math Methods

- One set of methods we need are mathematical methods (yes, programming involves math!) Small Basic provides a set of methods that perform tasks such as square roots, trigonometric relationships, and exponential functions.

- Each of the Small Basic math functions comes from the **Math** class. All this means is that each method name must be preceded by **Math.** (say Math-dot) to work properly. The methods and the returned values are:

Math Method	**Value Returned**
Math.Abs	Returns the absolute value of a specified number.
Math.Ceiling	Gets an integer that is greater than or equal to the specified decimal number. For example, 32.233 will return 33.
Math.Cos	Returns a value containing the cosine of the specified angle in radians.
Math.Floor	Hiermee wordt een geheel getal opgehaald dat kleiner is dan of gelijk is aan het opgegeven decimale getal. 32.233 retourneert bijvoorbeeld 32.
Math.GetDegrees	Converteert een gegeven hoek in radialen naar graden.
Math.GetRadians	Converteert een gegeven hoek in graden naar radialen.
Wiskunde.log	Hiermee wordt de logaritmewaarde (basis 10) van het gegeven getal opgehaald.
Math.Max	Berekent de grootste van twee getallen.
Wiskunde.min	Berekent de kleinste van twee getallen.
Math.NaturalLog	Hiermee wordt de natuurlijke logaritmewaarde van het gegeven getal opgehaald.
Math.Pi	Een constante die de verhouding aangeeft van de omtrek van een cirkel tot de diameter (3.14159265359...).
Wiskunde.Power	Hiermee verhoogt u een getal tot een opgegeven vermogen.

Math.Restant	Deelt het eerste getal door het tweede en retourneert het resterende getal.
Wiskunde.Ronde	Geeft als resultaat het getal dat het dichtst bij de opgegeven waarde ligt.
Math.Sin	Retourneert een waarde die de sinus van de opgegeven hoek in radialen bevat.
Math.SquareRoot	Retourneert een waarde die de vierkantswortel van een getal opgeeft.
Math.Tan	Geeft als resultaat een waarde die de raaklijn van een hoek in radialen bevat.

- **Voorbeelden**:

```
1. Math.Abs(-5.4) returns the absolute value of –5.4 (returns 5.4)
2. Math.Cos(2.3) returns the cosine of an angle of 2.3 radians
3. Math.Max(7, 10) returns the larger of the two numbers (returns 10)
4. Math.Power(2, 4) returns 2 raised to the fourth power (16)
5. Math.SquareRoot(4.5) returns the square root of 4.5
```

Willekeurige getallen

- We kiezen één wiskundige methode uit vanwege het belang ervan. Bij het schrijven van games en leersoftware gebruiken we een random number generator om onvoorspelbaarheid te introduceren. De methode **Math.GetRandomNumber** wordt in Small Basic gebruikt voor willekeurige getallen.

- Wanneer u een willekeurige geheel getalwaarde (geheel getal) nodig hebt, gebruikt u deze methode:

  Math.GetRandomNumber(Limiet)

  Deze instructie genereert een willekeurige waarde die tussen 1 en **Limiet ligt**. Bijvoorbeeld de methode:

  Math.GetRandomNumber(5)

  genereert willekeurige getallen van 1 tot 5. De mogelijke waarden zijn 1, 2, 3, 4 en 5.

- Een rol van een matrijs kan een getal van 1 tot 6 produceren. Om **GetRandomNumber** te gebruiken om een dobbelsteen te rollen, zouden we schrijven:

  DieNumber = Math.GetRandomNumber(6)

- Voor een kaartspel zouden de willekeurige gehele getallen variëren van 1 tot 52, omdat er 52 kaarten in een standaard speelspel zijn. Code om dit te doen:

  CardNumber = Math.GetRandomNumber(52)

- Als we een getal tussen -100 en 100 willen, gebruiken we:

  YourNumber = 101 -Math.GetRandomNumber(201)

- Bekijk de bovenstaande voorbeelden om er zeker van te zijn dat u ziet hoe de willekeurige getalgenerator het gewenste bereik van gehele getallen produceert.

Volgende hoofdstuk > >