Hoofdstuk 11

Gebeurtenissen en interactiviteit

In de eerste twee hoofdstukken hebben we objecten met *eigenschappen* en *bewerkingen* geïntroduceerd. Sommige objecten hebben naast eigenschappen en bewerkingen ook ***gebeurtenissen***. Gebeurtenissen kunnen je vergelijken met seintjes die bijvoorbeeld worden gegeven als een reactie op gebruikersacties zoals het verplaatsen van of klikken op de muis. Gebeurtenissen zijn in zekere zin het tegenovergestelde van bewerkingen. In het geval van een bewerking vraag jij als programmeur de computer iets te doen, terwijl in het geval van gebeurtenissen de computer jou laten weten wanneer er iets interessants is gebeurd.

Waarom zijn gebeurtenissen nuttig?

Gebeurtenissen zijn belangrijke als je interactiviteit aan een programma wilt toevoegen. Je moet gebeurtenissen gebruiken als je een gebruiker wilt laten deelnemen aan je programma. Als je bijvoorbeeld een boter-kaas-en-eieren-spelletje aan het schrijven bent, wil je natuurlijk dat de gebruiker een keuze kan maken als het zijn of haar beurt is. En hier gaan we gebeurtenissen gebruiken. Met gebeurtenissen ontvang je gebruikersinvoer binnen je programma. Dit lijkt nogal ingewikkeld, maar maak je geen zorgen, we zullen in een eenvoudig voorbeeld uitleggen wat gebeurtenissen zijn en hoe ze worden gebruikt.

Hieronder volgt een zeer eenvoudig programma met slechts één instructie en één subroutine. De subroutine gebruikt de bewerking *ShowMessage* op het object GraphicsWindow om zo een berichtvenster aan de gebruiker te tonen.

```
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  GraphicsWindow.ShowMessage("Je hebt geklikt.", "Hallo")
EndSub
```

Het interessante gedeelte in het programma hierboven is de regel waar we de naam van de subroutine toewijzen aan de **MouseDown** -gebeurtenis van het GraphicsWindow-object. Je zult zien dat MouseDown erg veel lijkt op een eigenschap, maar in plaats van dat we waarden toewijzen, wijzen we de subroutine *OnMouseDown* toe. Dit maakt gebeurtenissen zo speciaal. Wanneer de gebeurtenis plaatsvindt, wordt de subroutine automatisch opgeroepen. In dit geval wordt de subroutine *OnMouseDown* elke keer opgeroepen als de gebruiker met de muis op het GraphicsWindow klikt. Voer het programma uit en probeer het zelf. Telkens wanneer je met de muis op het GraphicsWindow klikt, zie je een berichtvenster net zoals hieronder wordt weergegeven.

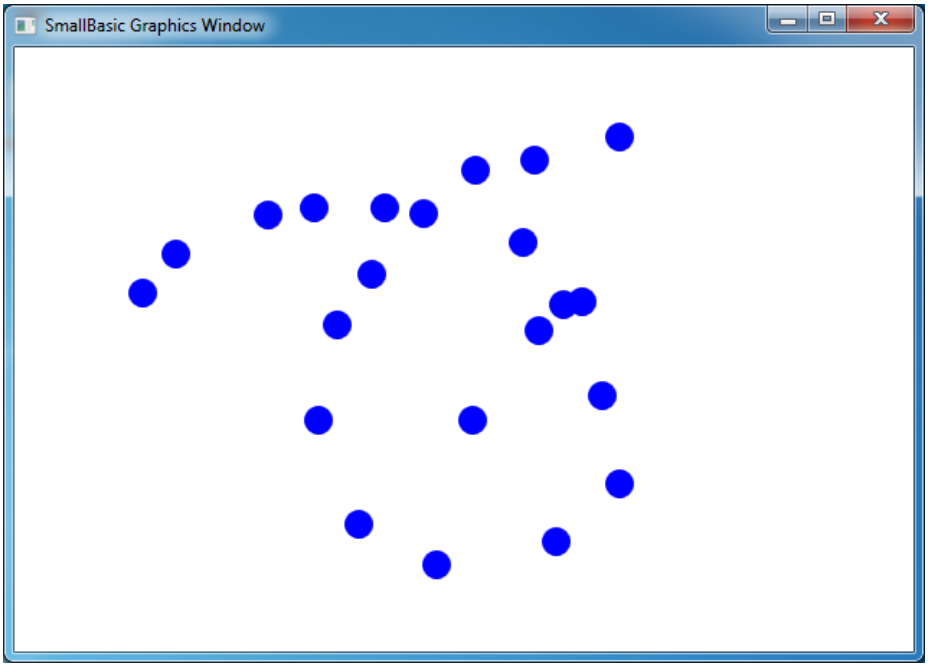
Afbeelding 55 – Reageren op een gebeurtenis

Deze manier van gebeurtenisverwerking is krachtig en stelt je in staat creatieve en interessante programma's te maken. Programma's die op deze manier worden geschreven, worden vaak gebeurtenisafhankelijke programma's genoemd.

Je kunt de subroutine *OnMouseDown* wijzigen om andere dingen te doen dan berichtvensters weer te geven. In het programma hieronder kun je bijvoorbeeld grote blauwe stippen tekenen daar waar de gebruiker met de muis klikt.

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

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



Afbeelding 56 – MouseDown-gebeurtenis verwerken

In het programma hierboven hebben we *MouseX* en *MouseY* gebruikt om de muiscoördinaten te verkrijgen. We gebruiken dit vervolgens om een cirkel te tekenen met de muiscoördinaten als het middelpunt van de cirkel.

Meerdere gebeurtenissen verwerken

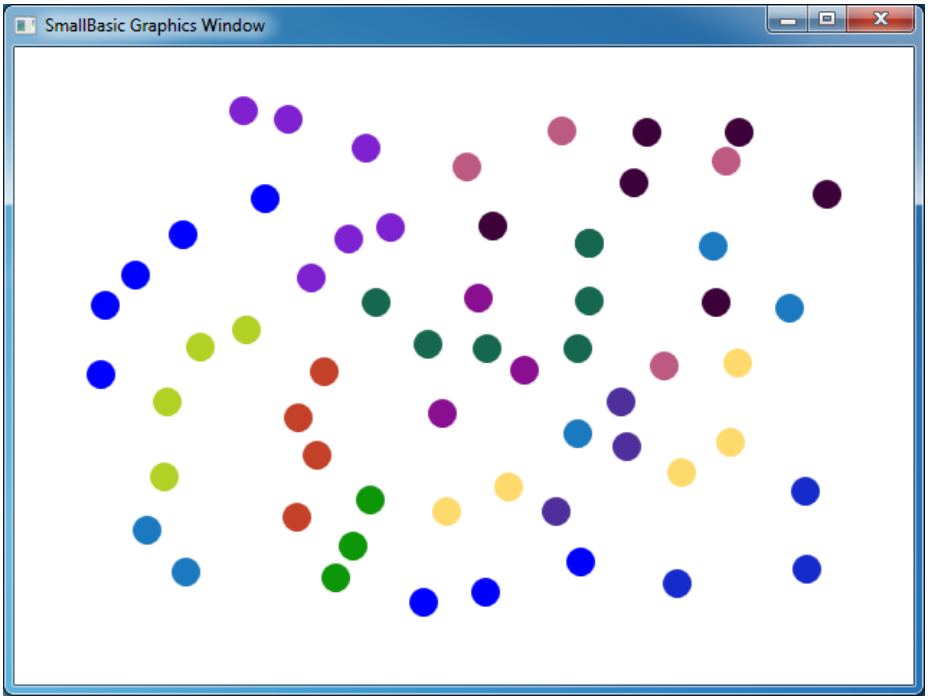
Het gebruik van gebeurtenissen is ongelimiteerd. Je kunt zelfs één subroutine meerdere gebeurtenissen laten verwerken. Je kunt een gebeurtenis echter slechts één keer verwerken. Als je twee subroutines aan dezelfde gebeurtenis probeert toe te wijzen, wordt de tweede verwerkt.

In het volgende voorbeeld wordt dit duidelijk gemaakt met een subroutine die toetsaanslagen verwerkt. Ook laten we deze nieuwe subroutine de kleur van de kwast wijzigen zodat de stip een nieuwe kleur krijgt wanneer je met de muis klikt.

```
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
```



Afbeelding 57 – Meerdere gebeurtenissen verwerken

Als je dit programma hebt uitgevoerd en op het venster hebt geklikt, wordt er een blauwe stip weergegeven. Als je nu één keer op een willekeurige toets klikt, wordt er een stip in een andere kleur weergegeven. Dit gebeurt omdat je op een toets drukt waardoor de subroutine *OnKeyDown* wordt uitgevoerd en hiermee wordt de kleur van de kwast naar een willekeurige kleur gewijzigd. Als je hierna op de muis klikt, wordt er een cirkel getekend met de nieuwe ingestelde kleur en dit geeft de stippen de willekeurige kleuren.

Een tekenprogramma

Met gebeurtenissen en subroutines kunnen we nu een programma schrijven waarmee we gebruikers op het venster kunnen laten tekenen. Het is verassend eenvoudig een dergelijk programma te schrijven als we het programma in kleinere stukjes opsplitsen. Als eerste stap gaan we een programma schrijven waarmee de gebruiker de muis overal in het venster voor grafische afbeeldingen kan verplaatsen en een spoor kan achterlaten.

```
GraphicsWindow.MouseMove = OnMouseMove

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

Als je dit programma uitvoert, begint de eerste lijn echter altijd aan de linkerbovenkant van het venster (0, 0). We kunnen dit probleem oplossen met het verwerken van de *MouseDown*-gebeurtenis en het vastleggen van *prevX*- en *prevY*-waarden wanneer die gebeurtenis wordt verwerkt.

Bovendien hoeven we eigenlijk alleen maar een spoor achter te laten wanneer de gebruiker de muisknop ingedrukt houdt. Als de gebruiker dit niet doet, moet de lijn niet worden getekend. Als we dit gedrag willen verkrijgen, gebruiken we de *IsLeftButtonDown*-eigenschap van het **Mouse**-object. Deze eigenschap vertelt ons wanneer de linkerknop wordt ingedrukt of niet. Als deze waarde waar is, wordt de lijn getekend en anders wordt de lijn niet getekend.

```
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
```