Nieuwsbrief

16de jaargang juni 2009 Nummer 2

HCC BASIC interessegroep

Inhoud

Onderwerp	blz.
BASIC snufjes 1: **- Het verschil tussen numeriek en alfanumeriek.** **- Do ... Loop nesten in een For ... Next lus**	**4**
Verder met XHTML en Scripts.	**6**
De module Main.	**10**
BASIC snufjes 2: **- Wat is een member?** **- Het Collection object.**	**15**
Mijn BASIC keuze: PowerBASIC. (Deel 1)	**18**
Basic controls: de tabbladen. (Deel 1)	**20**
Liberty BASIC: Slepen en neerzetten.	**23**

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Gordon Rahman kwam met een	
bijdrage.	23

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Willem Gobel	0118-850837	voorz@basic-gg.hcc.nl
Secretaris	Gordon Rahman Tobias Asserstraat 6 2037 JA Haarlem	023-5334881	secr@basic-gg.hcc.nl
Penningmeester	Piet Boere	0348-473115	penm@basic-gg.hcc.nl
Bestuurslid	Titus Krijgsman	075-6145458	t.krijgsman8@upcmail.nl
Redacteur	M.A. Kurvers Schaapsveld 46 3773 ZJ Barneveld	0342-424452	m.a.kurvers@hccnet.nl
Ledenadministratie	Fred Luchsinger	0318-571187	f.luchsinger@kader.hcc.nl
Webmaster	Jan van der Linden	071-3413679	j.vd.linden@kader.hcc.nl

http://www.basic.hccnet.nl

Redactioneel

In nieuwsbrief nr 2 heb ik wat BASIC snufjes, vooral de tweede die over het nesten van de lusstructuren gaat. Ook ga ik het over interessante snufjes hebben: de collecties. En al eens geprobeerd te werken met tabbladen?

Gordon Rahman heeft een heel mooi onderwerp in Liberty BASIC, hoe u drag and drop - dus slepen en neerzetten - kunt programmeren. U kunt complete listings in de bijdrage vinden.

Marco Kurvers

Het verschil tussen numeriek en alfanumeriek.

In computercode werken we met twee belangrijke onderdelen, numerieke en alfanumerieke code. Ook wel getallen en tekst genoemd, maar getallen kunnen ook nummers zijn en tekst kan bestaan uit symbolen en nummers. Daarom heet het (alfa)numeriek.
We weten dat we niet met tekst kunnen rekenen, dus ook niet op kunnen tellen. Toch gebruiken we in de meeste BASIC versies en dialecten dezelfde operator, de plus '+'. Andere versies ondersteunen niet de '+' operator, maar de '&' operator (ook wel de ampersand genoemd). In plaats van tekst optellen noemen we dat in computertaal: samenvoegen. Samenvoegen lijkt op invoegen en toevoegen, en eigenlijk gebeurt dat ook. In BASIC hebben we al vaak deze gebeurtenissen gezien. Zodra we een expressie gekoppeld zien aan een andere expressie dan spreken we van samenvoegen. Is het echter niet gekoppeld, maar uitgerekend dan spreken we van berekenen.

Tegenwoordig gebruiken we geen symbolen meer om met numerieke en alfanumerieke variabelen te kunnen werken, zoals variabelen met aan het eind een dollar-teken om alfanumerieke waarden toe te kennen.

Een waarde met een spatie.
Sommige waarden kunnen langer zijn dan de lengte van de waarde die toegekend is. Dat heeft te maken met de numerieke variabelen, die met teken (signed) of zonder teken (unsigned) kunnen zijn. BASIC weet van te voren niet wat voor een numerieke waarde wordt toegekend, zodat altijd één symbool, de spatie, wordt gereserveerd om het min-teken kwijt te kunnen.

Als er staat: `X = 2`
En daarna: `A = STR(X)  'of een andere converteerfunctie`
En daarna: `L = Len(A)  'of een andere lengtefunctie`
Dan is lengte `L` gelijk aan 2 en niet 1. Dat komt doordat er een spatie wordt gereserveerd.
Als er staat: `A = "25"` dan kunnen we niet spreken van een numeriek getal 25, maar van een alfanumeriek getal 25. Denk er dus aan dat dit geen normaal getal is en hiermee niet gerekend kan worden. Het is het beste om ook telefoonnummers nooit aan een numerieke variabele toe te kennen, omdat anders een lengte van elf tekens wordt gereserveerd en niet een lengte van tien tekens. Declareer een variabele voor een telefoonnummer zoals hieronder staat:

```
DIM TELNR AS STRING * 10   'de meeste BASIC versies ondersteunen wel
                           'het sterretje
```

Wordt het sterretje echter niet door BASIC begrepen, declareer de variabele dan gewoon met type `STRING`.

Er zijn ook BASIC versies waar variabelen niet gedeclareerd hoeven te worden. Denk maar aan Power BASIC en Liberty BASIC. Het verschil of een variabele nou numeriek of alfa-numeriek is, kan men zien aan het symbool dat een variabele heeft.

Het dollarteken '$', zoals eerder uitgelegd, wordt gebruikt om alfanumerieke waarden te kunnen gebruiken.
Gewone variabelen zonder symbool zijn standaard variabelen en kunnen dus elk getal of tekst aan. Het maakt niet uit of ze signed of unsigned zijn, singel of double zijn, enzovoort. Wat de laatste twee betekenen komt later aan bod.
Andere drie symbolen, de `! & %`, worden ook in verschillende BASIC versies gebruikt, maar er zullen versies zijn die dit niet accepteren. VBA beschouwt zelfs het uitroepteken alsof men een variabele zonder symbool gebruikt. Er zullen ook versies zijn die de variabelen zonder symbolen alleen numerieke waarden ondersteunen, en ook VBA heeft dat probleem. U kunt wel een string getal toekennen, maar VBA zal de waarde automatisch converteren in een numeriek getal.

Als het declareren kan, in de BASIC versie die u gebruikt, doe dat dan liever dan het niet te doen. U voorkomt op die manier fouten waar BASIC niets aan kan doen.

Do ... Loop nesten in een For ... Next lus.

Wat vaak vreemd uitkomt zijn de herhalingsstructuren. BASIC kent er een heleboel van. Het begon in de jaren '70 met een FOR ... NEXT, maar algauw kwamen er nieuwe lussen bij. Eén van de bekendste lusstructuur is de DO ... LOOP. De syntaxis van de lussen zien er als volgt uit.

```
FOR <tel> = <start> TO <eind> [STEP <stap>] ... NEXT [<tel>]
DO ... LOOP
DO ... LOOP UNTIL <conditie>
DO ... LOOP WHILE <conditie>
DO UNTIL <conditie> ... LOOP
DO WHILE <conditie> ... LOOP
```

Wat ik met bovenstaande titel bedoel is dat de lussen genest kunnen worden, of anders gezegd: een lus in een hoofdlus geprogrammeerd kan worden. Men kan daar zoveel lussen voor gebruiken, maar meestal is één binnenlus bij een hoofdlus al voldoende.
Bij bovenstaande lusstructuren is de tweede lus, de `DO ... LOOP`, de gevaarlijkste. Het is een onbeperkte lus, omdat er geen conditie bij staat. Er moet een keer uit de lus worden gesprongen om oneindig herhalen te voorkomen.

Onderstaande voorbeelden laten geneste lussen zien.

```
FOR I = 1 TO 10
   FOR J = 20 TO 50
      ...
   NEXT J
NEXT I

FOR X = 1 TO 10
   Y = 0
   S(X) = 0
   DO
      Y = Y + 1
      IF MID$(R$(X), Y, 1) = CHR$(32) THEN S(X) = S(X) + 1
   LOOP UNTIL S(X) = 10 OR Y = LEN(R$(X)
NEXT X
```

Uit het eerste voorbeeld kunnen er echter fouten worden gemaakt, namelijk om de lussen kruislings te nesten. Onderstaand voorbeeld laat de fout zien.

```
FOR I = 1 TO 10
   FOR J = 20 TO 50
      ...
   NEXT I
NEXT J
```

Om zo min mogelijk fouten te maken is er één oplossing: laat de lusvariabelen achter de `NEXT` statements weg.

Lussen nesten mag altijd. Toch is het beter om de binnenlus apart in een subroutine of functie te houden en die in de buitenlus aan te roepen.

Marco Kurvers

Verder met XHTML en Scripts.

In nieuwsbrief nr. 1 hebben we kennis gemaakt met de Crimson Editor. U hebt toen kunnen zien hoe we VBScript in XHTML kunnen gebruiken. De Crimson Editor biedt vele mogelijkheden, beter dan de code in een normaal kladblok te schrijven. De editor houdt alle sleutelwoorden, getallen, symbolen en identifiers apart met kleuren. Die kleuren kunt u zelf veranderen.

XHTML heeft een tag `SCRIPT` om andere scripttalen samen met XHTML te kunnen gebruiken, maar houd er rekening mee dat u niet alle talen met de tag `SCRIPT` kunt gebruiken.

Een voorbeeld is de script CSS die met de tag STYLE ingeschakeld moet worden.

TIP! Het is mogelijk om alle CSS opmaakcode in aparte documenten te houden, dat geldt ook voor de VBScript code. Door in de hoofdpagina, de index genoemd, deze documenten te koppelen, kunt u de hoofdpagina klein houden en van daaruit de opmaakcode CSS met eventueel wat scriptcode aanroepen. Bedenk goed dat CSS geen aanroepende statements kan hebben en alleen bedoeld is om uw pagina('s) keurig op te maken.

Echter ondersteunen niet alle browsers het attribuut LANGUAGE, en niet alle browsers ondersteunen het attribuut TYPE om de juiste scripttaal te kunnen kiezen. Daarom heeft men bepaald om zowel het TYPE attribuut als het LANGUAGE attribuut op te geven.

Hieronder ziet u de syntaxis van de script. Verder ziet u hoe u de controls, die u normaal op Windows formulieren plaatst, op een pagina kunt plaatsen via formulieren.

```
<script type="text/vbscript language="vbscript">
   ...
</script>
```

Forms en controls
Met VBScript kunt u regels code uitvoeren die te maken hebben met de events (gebeurtenissen) van de controls. Voordat u de controls op de pagina kunt plaatsen, moeten er formulieren worden aangemaakt. Voer de onderstaande FORM tag regel nog niet in bovenstaande script! Die tags zijn namelijk geen VBScript statements en moeten in de BODY staan.

```
<FORM NAME="Form1" [ACTION="ASPages/Register.asp" METHOD="GET"]>
   <FONT SIZE=grootte lettertype>
   ...
</FORM>
```

Als het om controls gaat, die met gegevens te maken hebben, moet u beginnen met de tag <INPUT>. Echter, niet bij alle web-controls moet dat, die met invoer te maken hebben. Bijvoorbeeld bij de control TEXTAREA zal de tag <INPUT> niet werken, omdat de control TEXTAREA met meerdere invoerregels werkt.

Hieronder kunt u meerdere controls vinden met uitleg. Natuurlijk zijn er nog meer controls en ook ActiveX controls kunt u gebruiken, maar die kunnen niet direct op het form worden geplaatst. Die moeten eerst naar de client gekopieerd worden en veel gebruikers willen dat niet.

Hieronder is een voorbeeld van een intrinsieke control die direct op het form geplaatst kan worden. Geef de control op zonder aanhalingstekens, maar de naam control en de waarde

moeten wel met aanhalingstekens. Vergeet niet om de intrinsieke controls, die met gegevensinvoer te maken hebben, te beginnen met de tag <INPUT>.

```
<INPUT TYPE=control NAME="naam control" VALUE="waarde">
```

TEXT	Met de control TEXT kan een invoerkader worden geplaatst om gegevens in te voeren, zoals een naam of een adres, enzovoort. Gebruik onderstaande regel om een invoervak te plaatsen: `<INPUT TYPE=TEXT NAME="Text1" VALUE="Test">` U kunt de regel nog uitbreiden door achter VALUE="" de SIZE en MAXLENGTH kenmerken op te geven. Hiermee kan de breedte van de control en de invoerlengte opgegeven worden.
PASSWORD	Met de control PASSWORD kunt u hetzelfde doen als dat u doet met de control TEXT. Echter zullen de ingevoerde tekens niet te zien zijn maar alleen sterretjes.

TEXTAREA	De control TEXTAREA komt overeen met de control TEXT, maar hierin kunt u meerdere regels tekst opgeven. De gebruikelijke navigatietoetsen werken hier ook. Onderstaande regel laat zien hoe u de tag <TEXTAREA> gebruikt:
	`<TEXTAREA NAME="Commentaar" ROWS=10 COLS=30>`
	`  De mooiste editor die ik ken!`
	`</TEXTAREA>`

Command Button	Klik op een Command Button om een handeling te activeren. Zonder VBScript zijn er echter maar twee handelingen mogelijk:
	- stuur de informatie door naar de server;
	- zet alle waarden terug naar de standaard waarden.
	De twee controls heten SUBMIT en RESET en werken net zoals een gewone button.
	`<INPUT TYPE=SUBMIT VALUE="Verzend Gegevens">`
	`<INPUT TYPE=RESET VALUE="Waarden Terugzetten">`
	
	Wilt u echter eigen knoppen maken, dan hebt u VBScript code nodig om de knoppen te besturen, zie hieronder. Het is dan niet mogelijk om de twee bovenstaande controls te gebruiken.

De event-handlers en VBScript.
Nu is natuurlijk de vraag: waar kunnen de ingevoerde gegevens gecontroleerd en verwerkt worden? Met gebruik van VBScript zullen ook meerdere controls gebruikt kunnen worden. De beperking van de Command Button, zoals dat in de laatste rij uitgelegd werd, zal niet meer nodig zijn. Nu kunt u de Button control gebruiken om zelf knoppen te programmeren. Vraag twee is dan: hoe werkt het verband tussen HTML controls en script code?

Een script kunt u vergelijken met een module die u maakt in Visual Basic, maar voor web-

site programmeren zal ook een event-handler in de script komen te staan.
Hieronder ziet u een voorbeeld met HTML en VBScript.

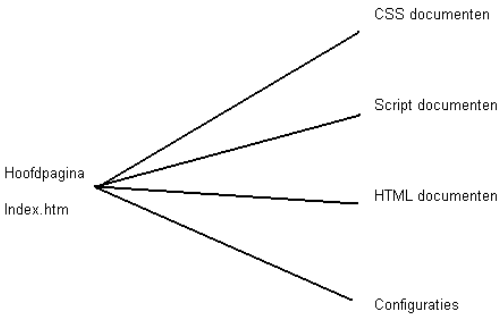
```
<HTML>
   <HEAD>
      <SCRIPT TYPE="TEXT/VBScript" LANGUAGE="VBScript">
         Sub btnOK_OnClick()
            MsgBox "Dit is de OK knop." & vbCrLf & _
                   "De datum is: " & Date
         End Sub
      </SCRIPT>
      <TITLE>Script voorbeeld.</TITLE>
   </HEAD>
   <BODY>
      <FORM NAME="Form1">
         <FONT SIZE=4>
         Dit is een voorbeeld met VBScript.<BR/>
         Klik op de knop voor de datum.
         <INPUT TYPE=BUTTON NAME="btnOK" VALUE="&OK">
      </FORM>
   </BODY>
</HTML>
```

De gegevens die zijn verzameld worden gewoonlijk naar het bestand "Register.asp" gestuurd, maar in bovenstaande voorbeeld is het nu niet opgegeven. De URL "ASPages/Register.asp" hebt u nodig wanneer u een "Register now" knop wilt gebruiken. U moet dan het type `BUTTON` vervangen door het type `SUBMIT`. De scriptcode zal dan ook niet worden aangeroepen.

TIP! Het is ook mogelijk een eigen registerknop te programmeren met de normale BUTTON en VBScript code. Zorg er dan wel voor dat de methode `SubMit` van de formulierinstantie RegistrationForm in de script staat om ervoor te zorgen dat de gegevens verzameld worden, zie meer daarover in het boek Visual Basic 6.0 van Sybex of bekijk de website: http://www.sybex.nl.

Documenten en pagina's.
In bovenstaand voorbeeld ziet u dat twee programmeertalen zijn gebruikt, HTML en VBScript. Doordat de naam van de button `btnOK` precies overeenkomt met de event subroutine `btnOK_OnClick()`, zal de actie goed worden uitgevoerd. U mag ook niet-event subroutines en functies schrijven in de scripts, als u maar ervoor zorgt dat ze uitgevoerd worden en niet in de documenten onuitgevoerd blijven staan en schrijf ook niet teveel scriptcode in een document terwijl de acties pas in andere pagina's ondernomen mogen worden. Hoe dat allemaal in elkaar zit kunt u zien bij figuur 1.

Figuur 1.



Om een website te programmeren is het verstandig aan de opbouw van figuur 1 te houden. We kunnen zeggen dat een website gewoon een library is die uit allerlei soorten documenten bestaat, maar dan is wel de vraag: waar zijn dan de pagina's? Is dan een document een pagina? Het antwoord daarop is: ja, alleen wanneer het document als een html bestand wordt opgeslagen; nee, als het document met een andere extensie wordt opgeslagen, zoals een CSS of een Script.
Het eerste antwoord kan ook 'nee' zijn, want ook een html bestand kan dienen als koppel-document voor één pagina.

Hebt u geen website editor, dan is het moeilijk om alles bij te houden. Het belangrijkste is in ieder geval: houd alles bij elkaar in dezelfde map. Gebruikt u echter Crimson Editor dan zal het bijhouden van alles wat makkelijker gaan. Crimson geeft u de mogelijkheid om met projecten te kunnen werken net zoals het in Visual Basic te werk gaat. Geef de juiste configuraties op via het menu 'Tools' en kies het menu-item 'Preferences'. Na de juiste instellingen kunt u een project maken door een nieuwe te kiezen of er een te openen, het menu 'Project' spreekt voor zich.
Door de juiste bestanden in uw project op te slaan, hoeft u uw bestanden zelf niet meer bij elkaar te houden – vergeet echter niet een standaardmap aan te maken. Crimson opent uw bestanden in tabbladen zodra u uw project opent.

In de volgende nieuwsbrief zal ik u laten zien hoe de documenten gekoppeld kunnen worden, hoe een CSS opmaak werkt en hoe object id's de web-controls als groepen kunnen werken zonder gebruik van formulieren.

Marco Kurvers

De module Main.

Verschillende BASIC versies, die visueel werken, hebben een startcode met een hoofdformulier nodig om de hele applicatie op gang te kunnen brengen. Dat op gang brengen, dus ervoor zorgen dat uw afgeronde werk goed zal functioneren, kan op verschillende manieren worden gedaan. Eén ding is wel belangrijk: de hoofdmodule, die we de startcode kunnen noemen. Helaas wordt dat door de gebruiker vaak vergeten en wordt meteen een formulier

opgestart zonder een hoofdmodule te gebruiken.

In Visual Basic 6 met een project beginnen.

Hieronder ziet u de mogelijkheden die u kunt kiezen. In een lijst staan de beschrijvingen van de soorten applicaties.

1 Standard EXE
Met een standaard EXE kunt u een normale Windows applicatie schrijven. Hoe u de omgeving van de applicatie wilt hebben komt pas later aan bod, zoals het instellen van een MDI omgeving.

2 ActiveX EXE
Met een ActiveX EXE kunt u een uitvoerbaar onderdeel maken voor componenten, designers en insertable objects. Degenen die al gemaakt zijn en gebruikt kunnen worden kunt u vinden in het menu 'Project' van Visual Basic en het menu-item 'Components'.

3 ActiveX DLL
Met een ActiveX DLL kunt u een bestuurbaar onderdeel maken voor object library's en andere referenties. Het is dan mogelijk om instanties te declareren voor die objecten. Door eerst zo'n DLL te maken hoeft u het wiel niet nog eens uit te vinden. De code kunt u dan hergebruiken.

4 ActiveX Control
Met een ActiveX Control kunt u zelf een control of besturingselement maken, net zo een als die in de toolbar van Visual Basic staat. Voordat uw control in de toolbar kan staan, moet ervoor gezorgd worden dat uw ActiveX in de menu-item 'References' komt te staan. U doet dat door op de knop 'Browse' te klikken en het juiste ActiveX bestand te kiezen, zodat de beschrijving van de control in de lijst komt te staan met daaronder de lokatie van het bestand.

5 VB Application Wizard
Met een VB Applicatie Wizard kunt u een programma schrijven met alle mogelijkheden voor de gebruiker zodat de applicatie met de juiste gebruikersinstellingen zal werken. Een wizard is niet hetzelfde als een setup programma, die heeft namelijk te maken met de installatie van de applicatie.

6 VB Wizard Manager
Voor de applicaties kunnen verschillende soorten wizards gemaakt worden. Met behulp

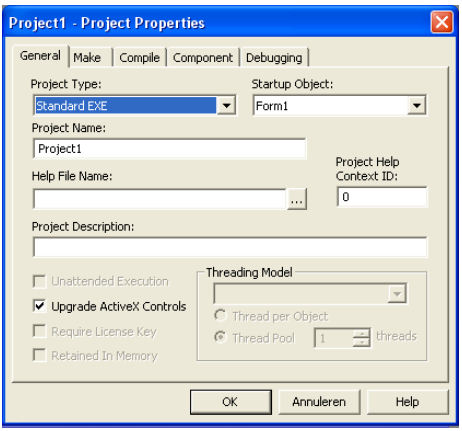
van een manager (beheerder) zal het maken van een wizard wat makkelijker gaan.

7 Data Project

Hiermee kunt u een programma schrijven met extra controls en formulieren die met data te maken hebben. Wat u er bijvoorbeeld bij krijgt is een DataEnvironment, een omgeving om het contact met de database in te kunnen stellen, en een DataReport, een pagina opgedeeld in secties om een rapport te kunnen maken.

De rest van de eventuele keuzes zijn minder belangrijk en het zou zelfs kunnen dat niet elke gebruiker hetzelfde 'New Project' opstartscherm heeft.

Meestal wordt de keuze 'Standard EXE' gekozen. Kies deze om met een nieuw project te beginnen. Zodra het nieuwe formulier 'Form1' er staat zullen we eens kijken wat de projectinstellingen zijn.



Project Type:
Hier ziet u het type die u gekozen hebt in het opstartscherm 'New Project', maar het is mogelijk om nog steeds van type te veranderen. Dat is niet aan te raden.
Project Name:
U kunt hier een eigen projectnaam opgeven.
Startup Object:
Kies hier welk object het startobject moet zijn. Hoewel een formulier gekozen mag worden, is het toch niet aan te raden.

Waarom een formulier niet geschikt is om als startobject te gebruiken.
Er zijn verschillende redenen waarom dat zo is:

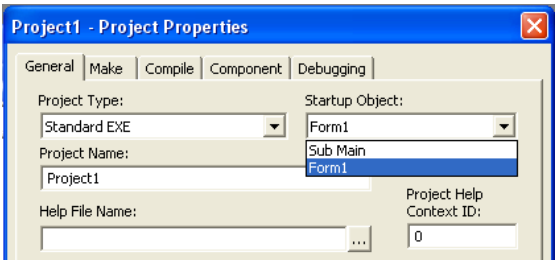
1 Een formulier is geen object, maar een sjabloon. Het is niet mogelijk om zelf een formulier in code te bewerken. In Visual Basic .NET is een formulier wel een object.
2 Belangrijke gegevens, zoals een database object, moeten in het hele project bekend zijn. Door een formulier als startobject te gebruiken zou u, wanneer u de database nodig hebt, telkens het formulier op moeten geven, zodat de compiler de database herkend.

Er is daarom één oplossing om van het probleem 'Start Object' af te komen: het aanmaken van een module. Het is mogelijk om zoveel modules aan te maken als u wilt, maar pas op: teveel modules is ook niet verstandig. Meestal zijn één of twee modules wel genoeg voor uw project. Mocht u merken dat een module erg vaak wordt gebruikt, maak er dan een klasse van. Met een klasse kunt u een object maken en ermee werken wanneer het u van pas komt en wanneer u het niet meer nodig mocht hebben kunt u het object weer vernietigen.

De subroutine 'Main' als startobject in een module.
Zodra u een module hebt toegevoegd, kunt u de naam van de module wijzigen in:
`modMain`
Type in het codescherm onderstaande subroutine.

```
Public Sub Main()

End Sub
```

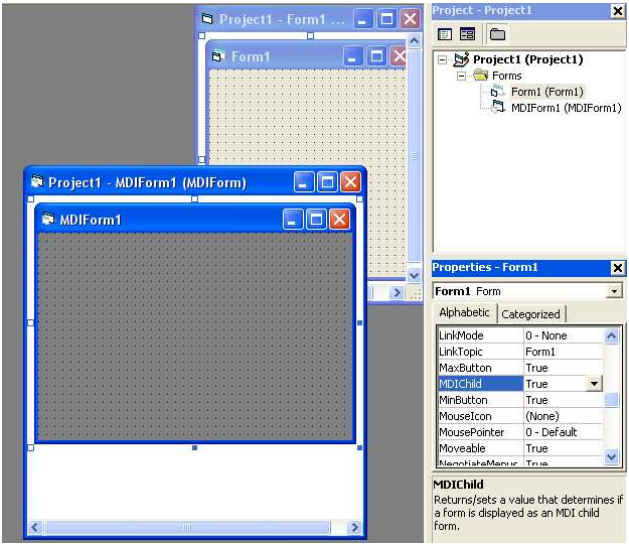
Open nogmaals het formulier 'Project Properties' en klik op het pijltje van 'Startup Object:'. Nu kunt u zien dat de Main subroutine is toegevoegd, zie figuur.



In de Main subroutine kunt u allerlei belangrijke onderdelen instellen die over het hele project bekend moeten zijn, behalve gegevens die te maken hebben met dialoogformulieren. Houd die altijd lokaal en gebruik ze alleen zodra een dialoogformulier geopend wordt. Voor die gegevens is het beter om klassen te ontwerpen en niet in modules te declareren.

We weten dat formulieren als zwarte dozen in Visual Basic 6 werken. Dit betekent dat ze minder functioneel zijn, omdat de meeste functies - in die zwarte doos - verborgen zijn. Het voordeel is dat u een formulier alleen maar hoeft te laden en te ontladen met twee makkelijke statements. Het nadeel is dat u, mocht u proberen een instantie van uw formulier aan te maken, die niet weergegeven kan worden. Een instantie moet toch ergens vandaan komen en aangezien een formulier geen object is, kan er dus geen geheugen voor die instantie gecreëerd worden. In Visual Basic .NET is het juist verplicht om een instantie voor het formulier te creëren. Visual Basic .NET kent ook niet die twee statements om formulieren te laden en te ontladen, u moet instanties aanmaken en ermee werken alsof het doodgewone klassenobjecten zijn.

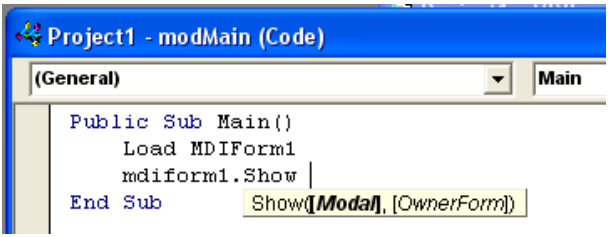
Twee verschillende projecten: met of zonder MDI?
In Visual Basic 6 is het eenvoudig om uw project voor een goede applicatie in te kunnen stellen voordat u met de code aan de gang gaat. Bedenk eerst hoe u de formulieren wilt hebben: los of vaderkind. Wilt u de laatste keuze dan hebt u twee formulieren nodig, een hoofdformulier en een formulier dat in het hoofdformulier vastzit. Hieronder ziet u een voorbeeld met twee formulieren.



Voordat u de eigenschap MDIChild voor Form1 instelt op True, wat niet helemaal erg is, kiest u in het menu Project het menu-item 'Add MDI Form'. Gelijk zal een donkere MDIForm1 verschijnen en zal ook het menu-item automatisch uit worden gezet. U kunt meerdere kindformulieren maken, maar nooit meer dan één vaderformulier.

Wanneer u de belangrijke instellingen hebt gemaakt kunt u beginnen met het startobject. De subroutine Main moet nu de juiste code hebben om het formulier MDIForm1 te kunnen tonen.
Hieronder een voorbeeld, maar ook een voorbeeld van het codescherm.

```
Public Sub Main()
   Load MDIForm1
   MDIForm1.Show
End Sub
```



De methode `Show` heeft parameters die optioneel zijn. Ik heb zelf niets opgegeven, het MDI formulier zal dan ook modeless worden geladen. Wat u op de tooltip ziet staan is de optionele keuze 'Modal'. Die keuze kan alleen voor gewone formulieren worden gemaakt. De tweede optionele keuze 'OwnerForm' is even wat lastiger. U kunt namelijk bij het weergeven van een modal formulier opgeven waar dit formulier aan moet toebehoren. Zo is het mogelijk dat de OwnerForm het MDI formulier zelf is. Kijk eens naar onderstaand voorbeeld.

```
Private Sub cmdShowResults_Click()
   ' Toont een modal form genaamd frmResults.
   frmResults.Show vbModal, Me    ' Me kan MDIForm1 zijn.
End Sub
```

U weet nu wat u kunt instellen als startobject en hoe u een vader- en kindformulier gebruikt. Later ga ik hiermee verder, want dan zal ik het ook eens over dialoogformulieren hebben en hoe we die samen met het MDI formulier kunnen gebruiken, zoals het voorbeeld laat zien met het formulier frmResults.

Marco Kurvers

Wat is een member?

Een member kunnen we vergelijken met een element, zoals een element in een array. Echter worden de elementen van een array geen members genoemd, doordat members alleen te maken hebben als elementen die we tegenkomen in:

- collecties;
- objecten;
- zelf gedefinieerde typen.

Members komen we vooral tegen als we het over collecties en objecten hebben en omdat het volgende hoofdstuk over collecties gaat, beantwoord ik alvast de vraag wat een member is zodat het volgende hoofdstuk voor u makkelijker te begrijpen is.

Het collection object.

Als u gewend bent met arrays te werken dan kunt u best wel een eind komen om met lijsten te werken. Maar arrays hebben ook nadelen. Elke array kan maar met één type werken en bovendien heeft het een omvang van zoveel elementen nodig die na de declaratie niet meer te wijzigen zijn.

```
Dim ArrayNaam([20]) As String   'string array van 20 elementen
ReDim ArrayNaam(30) As String   'string array van 30 nieuwe
                                 elementen
```

Houd er rekening mee dat u arrays, die al met een omvang van elementen zijn gedeclareerd, niet nogmaals kunt herdimensioneren. Als u bovenstaande twee regels samen gebruikt, zal Visual Basic een foutmelding geven als u probeert het aantal elementen in de `Dim` regel op te geven. Daarom ziet u ook twee rechte haken om het getal 20 staan. Als u zeker weet dat u een dynamische array wilt gebruiken vergeet dan niet om eerst de array te declareren met `Dim` en alleen de ronde haakjes. Pas daarna kunt u met `ReDim` de array herdimensioneren met een nieuwe omvang van elementen. Zorg er wel voor dat u eerst een kopie maakt van de array voordat u de array gaat herdimensioneren, want telkens zullen, na het herdimensioneren van de array, de oude gegevens verdwijnen.

Werken met collecties.
Toch kunnen we in Visual Basic dynamisch werken zonder telkens een nieuwe omvang van elementen op te moeten geven. Dat kan worden gedaan met behulp van het `Collection` object.
Declareer eerst een collection instantie:

```
Dim objCollection As New Collection
```

Met de instantie `objCollection` kunt u de methoden gebruiken die u hieronder ziet staan.

Methode	Beschrijving
`Add`	Parameters: Item, Sleutel, Ervoor, Daarna Met de methode kunt u items toevoegen van elk soort type. - Item: vereist. Een expressie van elk soort type die aan de collectie zal worden toegevoegd. - Sleutel: optioneel. Een unieke string expressie dat specificeert een sleutel string voor het aangeven van een positieve index. - Ervoor: optioneel. Een expressie dat een relatieve positie in de collectie specificeert. De toegevoegde member is in de collectie geplaatst voordat de expressie in het Ervoor argument begrepen is. Is de expressie numeriek dan moet Ervoor van 1 tot de waarde van de `Count` eigenschap, maar is de expressie een string dan moet Ervoor corresponderen met de opgegeven sleutel waarmee de member in de collectie is toegevoegd. U kunt een Ervoor positie of een Daarna positie opgeven, maar nooit beide. - Daarna: optioneel. Een expressie dat een relatieve positie in de collectie specificeert. De toegevoegde member zal pas in de collectie worden geplaatst nadat het Daarna argument begrepen is. Is de expressie numeriek dan moet Daarna van 1 tot de waarde van de `Count` eigenschap, maar is de expressie een string dan moet Daarna corresponderen met de opgegeven sleutel waarmee de member in de collectie is toegevoegd. U kunt een Ervoor positie of een Daarna positie opgeven, maar nooit beide. Onthoud, een Ervoor positie of een Daarna positie moet overeen komen met een bestaande member, anders zal een foutmelding verschijnen. Er zal ook een foutmelding verschijnen wanneer de opgegeven sleutel dupliceert met de sleutel van een bestaande member in de collectie.
`Count`	Parameters: geen Dit is geen methode maar een eigenschap. Geeft de aantal toegevoegde members in de collectie terug als een long-integer, alleen-lezen.

`Item`	Parameters: Index
	Geeft de juiste member terug via opgegeven Index van een Collection object. De Index is vereist.
	
	De `Item` methode is de standaard methode van een collectie. De volgende regels zijn daarom hetzelfde:
	
	`Print objCollection(1)`
	`Print objCollection.Item(1)`

`Remove`	Parameters: Index Verwijderd de member van de collectie met opgegeven Index. De Index is vereist.

Nadat een collectie gecreëerd is kunnen er members worden toegevoegd met de `Add` methode, en weer worden verwijderd met de `Remove` methode. De members kunt u terugkrijgen met de standaard `Item` methode. Er bestaat zelfs een lusstructuur waarmee u door een collectie kunt bladeren, zie onderstaand codefragment:

```
Dim Member As Variant
For Each Member In objCollection
   Print Member.Value
Next Member
```

Wat dit fragment laat zien lijkt veel op een normale `For...Next` lus. Het grote voordeel is echter dat hier niet een stopwaarde opgegeven hoeft te worden, omdat een `For Each` automatisch stopt zodra de laatste member is bereikt.
Een gewone lus gebruiken kan ook, maar dan moet u gebruik maken van de `Count` eigenschap als u door de collectie wilt bladeren. Wat u zelf wilt, maar de codefragment wordt dan wel anders:

```
Dim Member As Variant, I As Long
For I = 1 To objCollection.Count()
   Member = objCollection.Item(I)
   Print Member.Value
Next I
```

Bij de fragmenten kunt u zien dat ik de member declareer als een variant type. Dat is vereist, want bladeren door een collectie met een getypeerde member werkt niet. Ook de `Item` methode geeft geen getypeerde member terug. Het probleem, dat we niet getypeerd kunnen bladeren, komt doordat elk toegevoegd element in de collectie van elk soort type kan zijn. De `Item` methode kan geen member teruggeven van elk soort type en daarom wordt er een variant member teruggegeven.

Bladeren in een array.
Ook al is een array niet hetzelfde als een collectie, toch kan een array werken in een `For...Each` lus, en nu wel getypeerd, zie voorbeeld:

```
Dim S As String, Ar(20) As String
For Each S In Ar
   Print S
Next S
```

Dat getypeerd bladeren in een array wel kan komt doordat elke array maar uit één type bestaat, dus uit bovenstaand voorbeeld zijn alle 20 elementen string elementen.
Een ander voordeel om op die manier door een array te bladeren is dat u niet meer hoeft te bepalen of de array met element 0 of met element 1 moet beginnen. Houd er wel rekening mee dat de instelling `Option Base` ook nog steeds een rol speelt.

Marco Kurvers

Mijn BASIC keuze: PowerBASIC.

PowerBASIC is heden ten dage een uiterst moderne variant van het aloude BASIC front. PowerBASIC onderscheidt zich van de meeste andere BASIC varianten door het feit dat PowerBASIC werkt met een compiler (de meeste andere BASIC versies werken met een interpreter). PowerBASIC is de opvolger van Borland's TurboBASIC. Programmeur Robert Zale kocht in 1991 de rechten van Borland en ontwikkelde de taal verder. In 1991 werd PowerBASIC 2.0 (als opvolger van TurboBASIC 1.1) uitgebracht.

Kenmerken van de taal zijn:
- zeer uitgebreide lijst van commando's en functies;
- drie versies: een DOS versie, twee 32-bit Windows versies: PB/CC en PB/DLL (sinds versie 7: PB/WIN);
- de DOS versie is 99% compatible met QBasic en QuickBASIC;
- compiler die kleine en snelle applicaties oplevert;
- geen run-time bestanden nodig.

PowerBASIC is de verdere ontwikkeling van Borland's TurboBASIC, die tezamen met TurboC en TurboPascal destijds een revolutie in PC-land betekenden: kleine compilers (op één floppydisk) die een enorme kracht en snelheid aan zelf geschreven programma's kon geven. Tot die tijd was men aangewezen op grote compilers (UCSD compiler bijvoorbeeld) die alleen op grote mainframe computers konden draaien.

Destijds al een enorme snelheid, maar aangezien de ontwikkeling van TurboBASIC tot PowerBASIC tot in 2003 is voortgezet, is PowerBASIC een uiterst modern product, met alle moderne mogelijkheden van moderne talen. Zo treft u bijvoorbeeld in PowerBASIC for DOS vanaf versie 3.2 al pointers aan. Dit was de enige Basic compiler die pointers bood.

Versies

- PowerBASIC for DOS, versies 2.0 (1991), 2.1 (1992), 3.0 (1993), 3.1 (1994), 3.2 (1995), 3.5 (1997)
- PowerBASIC for Windows
    - PB/DLL, versies 1.0 (16bit), 1.1 (16bit), 2.0 (16bit + 32bit), 5.0 (32bit), 6.0 (32bit), 6.1 (32bit), 6.2 (32bit)
    - PB/Win, versies 7.0, 8.0 (2005) en 9.0 (eind 2008)
- PowerBASIC console compiler PB/CC, versies 1.0, 1.1, 2.0, 3.0, 3.1, 3.2, 4.0 (2005), 5.0 (eind 2008)

Sinds versie 7.0 van de Windows compiler is deze van PB/DLL naar PB/Win hernoemd, omdat deze perfect geschikt was om complete programmatuur te schrijven en niet specifiek voor DLL's bedoeld was.

Sinds versie 9.0 beschikt PB/WIN over uitgebreide mogelijkheden voor Object Oriented Programming (OOP).

BLOAT

Dit is de term die Amerikanen geven aan 'log'. 'Bloatware' is dan ook grote, logge en omvangrijke software, die alleen al vanwege de grootte van de distributie bestanden veel eisen stelt aan het verspreidingsmedium (één of meerdere CD's), de computer (riante ruimte op de harde schijf) en intern geheugen (tientallen megabytes alleen al voor de toepassing).

PowerBASIC Inc is de strijd aangegaan met die omvangrijke software en wat blijkt: vrijwel alle software die u zult maken past gemakkelijk op één enkele floppydisk. Geen extra run-time bibliotheken benodigd, zoals VB wel heeft.

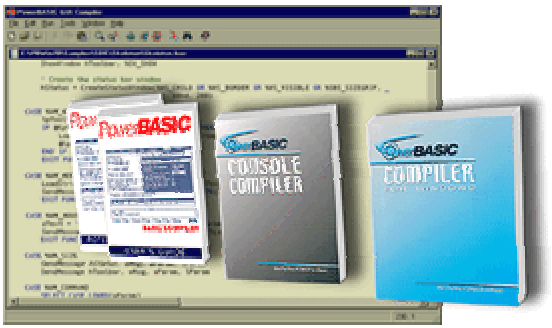
Als u op het aantal bytes wordt betaald voorziet PowerBASIC zelfs in een instelling om programma's kunstmatig groter te maken, voor diegenen die niet kunnen geloven dat een kleine Executable toch het hele werk kan doen! Nieuw in versie PB/WIN7, op verzoek van de betatesters.

Drie compiler-types.

PowerBASIC heeft drie aparte compilers voor elk van uw doel: de aloude DOS omgeving, tekstgeoriënteerde Windows applicaties of echte GUI-(grafische) Windows toepassingen. Bij alle drie compiler types heeft uw maximale kracht aan uw vingertoppen, want zelfs al zit er een traag Windows besturingssysteem tussen: PowerBASIC levert u bliksemsnelle toepassingen die u laat zien dat uw computer inderdaad snel kan zijn..... met het juiste programma en met de juiste compiler.

De Windows compilers leveren true native 32-bit Windows code af, dus geen slome interpreter, geen emulatie, maar pure machine code die met Windows meteen in het hart koppelt. De producten zijn PB/CC (voor console toepassingen) en PB/WIN (voor volledige GUI toepassingen, opvolger van PB/DLL).
De DOS compiler levert true-machine code voor uitvoer in een DOS omgeving en is ook geschikt voor near-realtime en real-time systemen.

Als u voor het eerst begint met programmeren dan is het misschien raadzaam om meteen met PB/WIN te beginnen. Hoewel programmeren in de DOS omgeving gemakkelijker is, zult u toch ooit eens naar de Windows omgeving gaan. Veel van uw opgedane kennis is dan niet meer bruikbaar door de limieten die Windows stelt aan applicaties.



Referenties

De compilers van PowerBASIC worden o.a. gebruikt door: Nationaal Lucht en Ruimtevaart laboratorium (NLR), NASA, TU-bouwkunde Delft, Vrije Universiteit van Amsterdam en diverse Nederlandse ziekenhuizen en tienduizenden semi-professionele en professionele programmeurs. Professor Hans Lauwerier was een enthousiast voorstander en gebruiker van de taal. Een overzicht van gebruikers vindt u op de site van PowerBASIC Inc (www.powerbasic.com/powerwho.asp).

Marco Kurvers

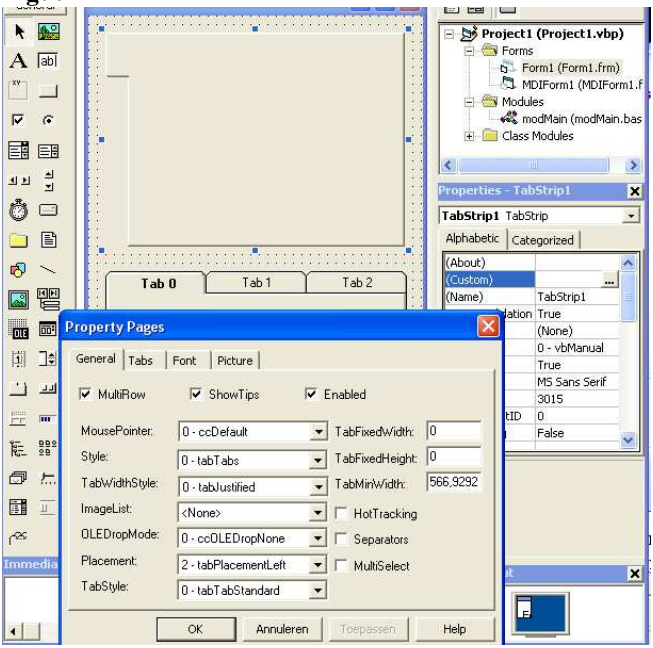
Basic controls: de tabbladen.

In veel programma's kunt u vaak zien dat bepaalde onderdelen, opties, instellingen of wat dan ook, vaak ingedeeld worden in tabbladen. Een prachtig mooi voorbeeld is het Office programma Excel waarmee de rekenbladen netjes zijn ingedeeld. De knoppen, die we onderaan kiezen, zijn eigenlijk geen knoppen. Het zijn de tabs die aan de rekenbladen vastzit-

ten. U zou zelfs zo'n programma kunnen maken door eenvoudig op het tabblad een flexgrid control te plaatsen. Toch komen de vragen natuurlijk weer tevoorschijn, zoals: hoe gebruiken we de tabbladen in Basic?

Op de volgende pagina kunt u figuur 1 zien met twee tabbladen en rechts de eigenschappenlijst met daaronder het dialoogscherm om een tabblad control op te maken.

Figuur 1



Wat u hier aan het voorbeeld ziet is eigenlijk nog steeds het project die ik ook gebruikt heb over de paragraaf: De module Main.

Hier heb ik het vaderkind formulier gevuld met twee tabblad controls, een standaard TabStrip en een uitgebreide SSTab. Het uitgebreide tabblad SSTab heeft twee tabblad-stylen en heeft een betere bladopmaak. De TabStrip kan weergegeven worden met gewone tabs, maar ook met knoppen. U kunt bij Style een andere optie kiezen, tabs of knoppen. Er is ook nog een derde style, de FlatButtons style. Daar kom ik in de volgende nieuwsbrief op terug.

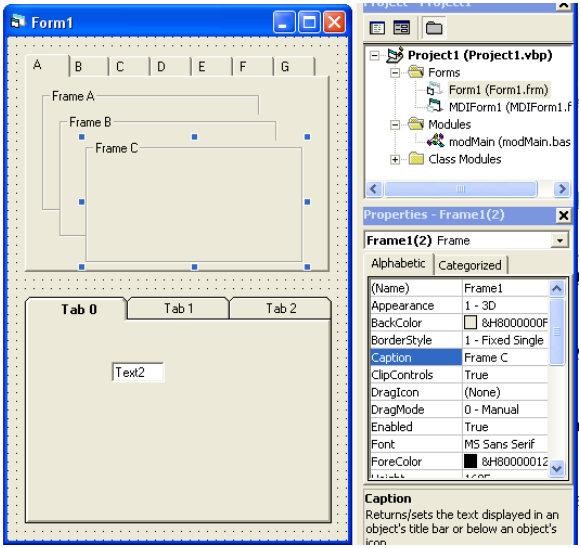
De TabStrip.

De tabcontrol die op figuur 1 actief is, is de standaard TabStrip. Het is een makkelijk te gebruiken control, vooral als u gebruik wilt maken van knoppen in plaats van tabs. Als u de TabStrip wilt gebruiken, houd er dan rekening mee dat de besturingselementen, die u op de tabbladen wilt zetten, niet op de tabbladen vast zullen staan. De TabStrip is namelijk geen containercontrol, maar de SSTab wel.

Om toch verschillende besturingselementen over de tabbladen te kunnen verdelen, moet u de TabStrip zelf beheren. Dat betekent dat u voor elke tabblad een groep moet maken met de juiste besturingselementen erop. De oplossing is door gebruik te maken van frames, maar ook de frames zullen niet op de tabbladen worden vastgehouden, zie figuur 2 op de volgende pagina.

U moet zelf de plaats bepalen waar de frames komen te staan. Dat kan echter niet op de IDE worden gedaan. Het plaatsen van de frames en het bepalen welke frame zichtbaar moet zijn moet allemaal in de code worden geprogrammeerd. Dat alles zal in de volgende nieuwsbrief komen te staan.

Figuur 2



Plaats op elke frame de juiste besturingselementen. Een frame is een containercontrol en kan daarom goed voor zulke omstandigheden worden gebruikt. Het nadeel is dat de frames veel ruimte op het formulier in beslag nemen. U kunt doen zoals de frames op figuur 2 staan, maar u kunt ook het formulier groter maken en alleen die ruimte gebruiken voor frames. Zorg er wel voor dat u de extra ruimte weg laat, door de juiste hoogte en/of breedte van het formulier in code in te stellen.

Tip! Plaats altijd het eerste frame op het eerste tabblad, want de coördinaten (Left en Top) hebt u nodig voor de andere frames. Plaats nooit alle frames met dezelfde coördinaten op elkaar.

Wat u ook kunt zien op figuur 2 is dat het frame een control-array is. Visual Basic zal, na het kopiëren en plaatsen van het tweede frame, vragen om een control-array. Bij gebruik van de tabbladen is dat best handig, want dan kunt u de actieve index van een tabblad samen laten werken met de index van een frame. U hoeft dan niet telkens het juiste frame op te zoeken en voorkomt u op die manier fouten.

De SSTab.
De SSTab heeft zeer handige tabbladen. Zoals u bij figuur 2 kunt zien staat er al een tekstvak op tabblad Tab 0 zonder gebruik van een frame. De tabbladen van de SSTab zijn containers en houden de besturingselementen vast die erop worden gezet. Zodra op een ander blad wordt geklikt, Tab 1 of Tab 2, zal het tekstvak verdwijnen. Klikt u echter weer op Tab 0 dan zal het tekstvak gewoon weer verschijnen.

Het gebruik van de SSTab biedt grote voordelen, maar maakt het programmeren een stuk moeilijker. Dat komt omdat de control veel meer eigenschappen en methoden heeft dan de simpele TabStrip. De SSTab heeft ook nog eens twee tabblad stylen: normale weergave en eigenschappen weergave. Wat de SSTab echter niet heeft zijn de knoppen. Knoppen kunnen alleen ingesteld worden bij de TabStrip.

Marco Kurvers

Liberty BASIC: Slepen en neerzetten.

Misschien zouden we de titel van deze paragraaf anders kunnen noemen. Het slepen en neerzetten, dat ook wel drag and drop wordt genoemd, is iets wat we heel vaak in programma's doen. We selecteren een figuur waarbij we de linker muisknop ingedrukt houden en slepen die figuur (die aan onze muis vastgeplakt lijkt) ergens naartoe en laten die figuur daar liggen.

Voor deze eenvoudige programmeertruc moet je in Liberty BASIC met slechts drie commando's spelen.

```
When leftButtonDown  -  Hiermee controleer je of de linker muisknop ingedrukt is.
When leftButtonMove  -  Hiermee scan je het bewegen van de muis.
When leftButtonUp    -  Hiermee controleer je of de linker muisknop vrij is.
```

Hier volgt het programma in pseudo-computertaal.

1. Controleer of de linker muisknop wordt ingedrukt en ga daarbij dan controleren of dat boven een te verslepen figuur (bijvoorbeeld een sprite) gebeurt.
2. Als 1 geldt, bepaal dan de MouseX en de MouseY coördinaten. Verberg de figuur en bepaal wat de nieuwe MouseX en MouseY zijn. Laat de figuur nu op de nieuwe locatie verschijnen. Herhaal deze handeling totdat je merkt dat de linker muisknop niet meer ingedrukt wordt.
3. Plaats de sprite en ga weer gewoon controleren of linker muisknop ingedrukt wordt.

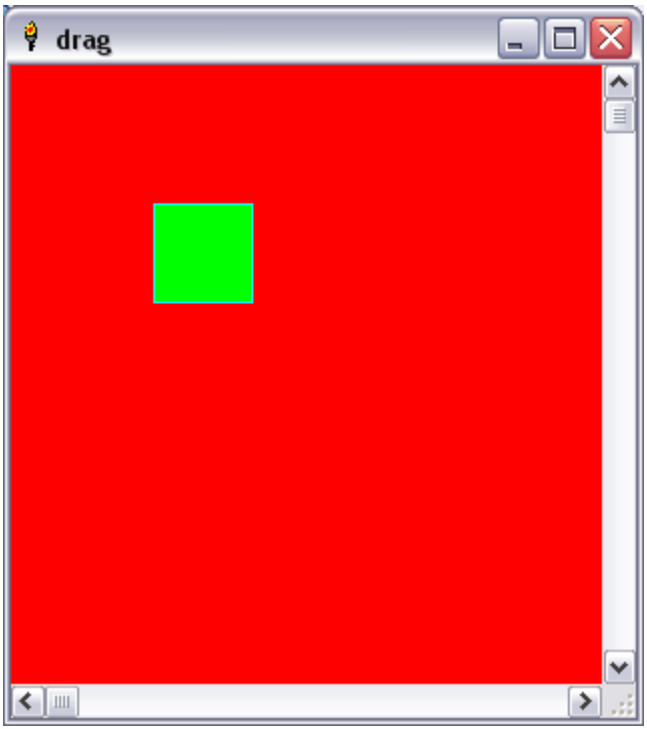
Okay, je hebt volgens het pseudo-computertaal programma dus ook een achtergrond nodig en een sprite. Verder kun je wat variëren met het thema. Ik vergat te melden dat je natuurlijk ook de Spritexy nodig hebt als je een sprite wilt slepen. Ik weet dat mijn volgende verzoek velen onder u tegen zal staan, maar ik doe het toch maar: download en installeer Liberty BASIC even of download Just BASIC tenminste. Just BASIC is de gratis versie van Liberty BASIC.

Liberty BASIC vind je op www.libertybasic.com
Just BASIC download je vanaf www.justbasic.com

Hier volgt een eenvoudig voorbeeldprogramma van een groen vierkantje in een venster met een rode achtergrond. Het vierkant kun je met de muis slepen en op een willekeurige plek in het rode venster neerleggen. De listing spreekt voor zichzelf, maar omdat u misschien voor het eerst naar een Liberty BASIC listing kijkt, zal ik bij sommige commandoregels even stil staan.

```
offsetx=0     'offset shape van muis
offsety=0
shapex=30     'shape x locatie
shapey=30     'shape y locatie
shapewide=50
shapehight=50

nomainwin     'geen uitleesvenster
open "drag" for graphics as #1  'hiermee maken we een venster (form)
   #1 "trapclose [quit]"   'controleer of het rode venster gesloten
                           wordt
   #1 "down "    'pen op papier deze instructie is nodig om te
                 tekenen
   #1 "setfocus"
   #1 "when leftButtonDown[startDrag]"   'linker muisknop ingedrukt?
                                         'spring naar label [start
                                         Drag]
   gosub [drawShape]
   wait

[quit] close #1:end

[startDrag]     'alleen procedureel als muis is in de shape
    if MouseX>shapex+shapewide then wait
    if MouseX<shapex then wait
    if MouseY>shapey+shapehight then wait
    if MouseY<shapey then wait

    offsetx=MouseX-shapex   'ok de muis staat in het groene veld
    offsety=MouseY-shapey   'ok de muis staat in het groene veld

    #1 "when leftButtonMove[doDrag]"
    #1 "when leftButtonUp[stopDrag]"
    wait

[doDrag]
    gosub [drawShape]       'verwijder huidige shape
    shapex=MouseX-offsetx
    shapey=MouseY-offsety
```

```
    gosub [drawShape]       'teken de shape op nieuwe locatie
    wait

[stopDrag]
    #1 "when leftButtonMove"
    wait

[drawShape]
    #1 "fill red"           'maak het venster (achtergrond) rood
    #1 "backcolor green"
    #1 "place ";shapex;" ";shapey
    #1 "boxfilled ";shapewide+shapex;" ";shapehight+shapey

    return
```



Zo ziet het resultaat van de listing eruit als het uitgevoerd wordt. Het groene vierkant kun je met de muis (ver)slepen. Om het gehele sleepproces een beetje beter te laten verlopen wordt naast de MouseX en MouseY calculatie ook gebruik gemaakt van een offset, waardoor het te verslepen venster niet abrupt aan de muistip blijft plakken.

In het bovenstaande voorbeeld is er geen sprite gebruikt en dus is er ook geen achtergrond plaatje aan te pas gekomen. Maar als we een sprite in het geheugen zouden hebben ge-

plaatst, dan zouden enkele commando's er even anders uitzien. Zo zou verbergen en tonen van de sprite er als volgt uit kunnen zien.

```
[noShape]
     #1 "spritevisible gcrown off"   'de sprite heet hier gcrown
     #1 "drawsprites"
     return
[drawShape]
     #1 "spritexy gcrown ";shapex;" ";shapey
     #1 "spritevisible gcrown on"
     #1 "drawsprites"
     return
```

Natuurlijk moeten we ook iets programmeren waardoor we eenvoudig kunnen ontdekken of de muis boven een sprite staat. Daartoe maken we van de muispunt een kleine sprite en zodoende kunnen we de botsingen tussen onze muis (sprite punt) en enkele andere sprite eenvoudig aflezen in de collisionlist.

Ik heb tenslotte een listing hieronder geplaatst. Het is een voorbeeld van het gebruik van de sleeptechniek. De listing is gebaseerd op het spelletje IO peg. Kun je slechts een pion overhouden?

Deze listing schreef ik naar aanleiding van een discussie op een Amerikaans forum http://libertybasic.conforums.com/index.cgi?board=game&action=display&num=1096476689. De demo maakt gebruik van een sprite die je vinden kunt in de BMP map van Liberty BASIC. Met Liberty BASIC worden standaard vele honderden listings en plaatjes meegeleverd. U moet de listing runnen vanuit de Liberty BASIC map, anders zal het programma het plaatje smiley1.bmp (de sprite) niet kunnen vinden.

Demo: Drag and Drop.
Je zou ook eigen wav bestanden kunnen gebruiken op de plekken waar dergelijke geluidsbestanden worden opgeroepen, zie bijvoorbeeld:

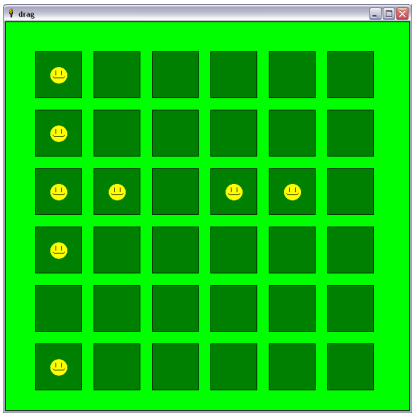
```
playwave "bloop_x.wav", async 'plaats je wave bestand hier

'drag by Gordon Rahman
'with BMP background and sprite
'special thanks to Janet and Alyce
'game seen on the Internet

    notice "HELP"+chr$(13)+_
         "Jump over a ball to clear it"+chr$(13)+_
         "Jumps can be horizontal or vertical"+chr$(13)+_
         "A cleared field takes you to the next level"

    WindowWidth = 700
    WindowHeight = 700
```

```
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)

    DIM p(6,6), p$(6,6)  'p = Playground array

    loadbmp "ssmiley",DefaultDir$+"\sprites\smiley1.bmp"

    nomainwin

    open "drag" for graphics_nsb as #1
         #1 "trapclose [quit]"
         #1 "fill green"
         #1 "backcolor darkgreen"
            FOR H=1 TO 6
                 FOR V=1 TO 6
                      X=(H-1)*100+50:Y=(V-1)*100+50
                      #1, "down; place ";X;" ";Y
                      #1, "boxfilled ";X+80;" ";Y+80
                      #1, "flush"
                 NEXT V
            NEXT H
         #1 "getbmp dragbg 0 0 700 700"
         #1 "getbmp mouser 0 0 1 2"
         #1 "addsprite mouser mouser"

         'get pattern
         DATA 1,0,0,0,0,0
         DATA 1,0,0,0,0,0
         DATA 1,1,0,1,1,0
         DATA 1,0,0,0,0,0
         DATA 0,0,0,0,0,0
         DATA 1,0,0,0,0,0

         DATA 0,0,0,1,0,0
         DATA 0,0,0,1,0,0
         DATA 0,0,0,1,0,0
         DATA 0,0,1,1,1,0
         DATA 0,0,1,0,0,0
         DATA 0,0,0,0,0,0

         'anchors for the smileys on the playboard
         for i = 1 to 6
              C(i) = i * 100 - 25
              R(i) = i * 100 - 25
         next i

[nextLevel]
     'initiate board values
     for R = 1 to 6
```

```
          for C = 1 to 6
               read datanumber
               if datanumber = 1 then
                    p(C,R) = 1
                    #1 "addsprite gcrown"+str$(q+1)+" ssmiley"
                    #1 "spritexy gcrown"+str$(q+1)+" ";C(C);" ";R(R)
                    p$(C,R) = "gcrown"+str$(q+1)
                    q = q+1
               end if
          next C
     next R

     #1 "setfocus"
     #1 "when leftButtonDown [startDrag]"
     #1 "background dragbg"
     #1 "drawsprites"
     wait

[quit] close #1:end

[startDrag]
     #1 "spritexy mouser ";MouseX;" ";MouseY
     #1 "drawsprites";
     #1 "spritecollides mouser list$"
     if len(list$) > 0 then
          #1 "spritexy? ";list$;"shapex shapey"
          ishapex = shapex : ishapey = shapey
          iX = int((shapex+25)/100) :iY = int((shapey+25)/100)
     end if

     offsetx=MouseX-shapex
     offsety=MouseY-shapey
     #1 "when leftButtonMove [doDrag]"
     #1 "when leftButtonUp [stopDrag]"
     wait

[doDrag]
     gosub [noShape] 'erase current shape
     shapex=MouseX-offsetx
     shapey=MouseY-offsety
     gosub [drawShape]  'draw shape in new spot
     wait

[stopDrag]
     nX = int((MouseX+25)/100)
     nY = int((MouseY+25)/100)
     if abs(nX-iX)=2 and nY=iY then
          if nX > iX and p(nX-1,nY)=1 and p(nX,nY)=0 then
               p(nX,nY) = 1 : p(iX,iY) = 0 :p(nX-1,nY)=0
```

```
            #1 "spritexy ";list$;" ";C(nX);" ";R(nY)
            p$(nX,nY) = list$
            #1 "removesprite ";p$(nX-1,nY)
            #1 "drawsprites"
            'voeg uw wav bestand hieronder in playwave
            playwave "bloop_x.wav", async
            gosub [testEndGame]
            if newLevel = 1 then
                 notice "NEXT LEVEL"
                 #1 "removesprite ";p$(nX,nY): p(nX,nY)=0
                 #1 "drawsprites"
                 goto [nextLevel]
            end if
            wait
       end if
       if nX < iX and p(nX+1,nY)=1 and p(nX,nY)=0 then
            p(nX,nY) = 1 : p(iX,iY) = 0 :p(nX+1,nY)=0
            #1 "spritexy ";list$;" ";C(nX);" ";R(nY)
            p$(nX,nY) = list$
            #1 "removesprite ";p$(nX+1,nY)
            #1 "drawsprites"
            playwave "bloop_x.wav", async
            gosub [testEndGame]
            if newLevel = 1 then
                 notice "NEXT LEVEL"
                 #1 "removesprite ";p$(nX,nY): p(nX,nY)=0
                 #1 "drawsprites"
                 goto [nextLevel]
            end if
            wait
       end if
end if

if abs(nY-iY)=2 and nX=iX then
       if nY > iY and p(nX,nY-1)=1 and p(nX,nY)=0 then
            p(nX,nY) = 1 : p(iX,iY) = 0 :p(nX,nY-1)=0
            #1 "spritexy ";list$;" ";C(nX);" ";R(nY)
            p$(nX,nY) = list$
            #1 "removesprite ";p$(nX,nY-1)
            #1 "drawsprites"
            playwave "bloop_x.wav", async
            gosub [testEndGame]
            if newLevel = 1 then
                 Notice "NEXT LEVEL"
                 #1 "removesprite ";p$(nX,nY): p(nX,nY)=0
                 #1 "drawsprites"
                 goto [nextLevel]
            end if
            wait
```

```
          end if
          if nY < iY and p(nX,nY+1)=1 and p(nX,nY)=0 then
               p(nX,nY) = 1 : p(iX,iY) = 0 :p(nX,nY+1)=0
               #1 "spritexy ";list$;" ";C(nX);" ";R(nY)
               p$(nX,nY) = list$
               #1 "removesprite ";p$(nX,nY+1)
               #1 "drawsprites"
               playwave "bloop_x.wav", async
               gosub [testEndGame]
               if newLevel = 1 then
                    notice "NEXT LEVEL"
                    #1 "removesprite ";p$(nX,nY): p(nX,nY)=0
                    #1 "drawsprites"
                    goto [nextLevel]
               end if
               wait
          end if
     end if

     if len(list$) > 0 then
          playwave "blurp_x.wav", async
          #1 "spritevisible ";list$;" on"
          #1 "spritexy ";list$;" ";ishapex;" ";ishapey
          #1 "drawsprites"
          #1 "drawsprites"
          #1 "when leftButtonMove"
     end if
     wait

[noShape]
     if len(list$) > 0 then
          #1 "spritevisible ";list$;"
               off"
          #1 "drawsprites"
     end if
     return

[drawShape]
     if len(list$) > 0 then
          #1 "spritexy "+list$+"
               ";shapex;" ";shapey
          #1 "spritevisible ";list$;" on"
          #1 "drawsprites"
     end if
     return

[testEndGame]
     newLevel = 0
     for R = 1 to 6
```



```
        for C = 1 to 6
             if p(C,R) = 1 then newLevel = newLevel + 1
        next C
    next R
    return
```

Gordon Rahman

Cursussen

Qbasic: Cursus, lesmateriaal en voorbeelden op CD-ROM € 6,00 voor leden. Niet leden € 10,00.
QuickBasic: Cursusboek en het lesvoorbeeld op diskette,
€ 11,00 voor leden. Niet leden € 13,50
Visual Basic 6.0: Cursus, lesmateriaal en voorbeelden op CD-ROM,
€ 6,00 voor leden. Niet leden € 10,00
Basiscursus voor senioren, Windows 95/98,
Word 97 en internet voor senioren, (geen diskette). € 11,00 voor leden. Niet leden € 13,50

Software

Catalogusdiskette,	€ 1,40 voor leden. Niet leden € 2,50
Overige diskettes,	€ 3,40 voor leden. Niet leden € 4,50
CD-ROM's,	€ 9,50 voor leden. Niet leden € 12,50

Hoe te bestellen

De cursussen, diskettes of CD-ROM kunnen worden besteld door het sturen van een e-mail naar penm@basic-gg.hcc.nl en storting van het verschuldigde bedrag op:

ABN-AMRO nummer 49.57.40.314
HCC BASIC ig
Haarlem

onder vermelding van het gewenste artikel. Vermeld in elk geval in uw e-mail ook uw adres aangezien dit bij elektronisch bankieren niet wordt meegezonden. Houd rekening met een leveringstijd van ca. 2 weken.
Teksten en broncodes van de nieuwsbrieven zijn te downloaden vanaf onze website (http://www.basic.hccnet.nl). De diskettes worden bij tijd en wijlen aangevuld met bruikbare hulp- en voorbeeldprogramma's.
Op de catalogusdiskette staat een korte maar duidelijke beschrijving van elk programma.

Alle prijzen zijn inclusief verzendkosten voor Nederland en België.

Vraagbaken

De volgende personen zijn op de aangegeven tijden beschikbaar voor vragen over programmeerproblemen. Respecteer hun privé-leven en bel alstublieft alleen op de aangegeven tijden.

Waarover	Wie	Wanneer	Tijd	Telefoon	Email
Liberty Basic	Gordon Rahman	ma. t/m zo.	19-23	(023) 5334881	grahman@planet.nl
MSX-Basic	Erwin Nicolai	vr. t/m zo.	18-22	(0516) 541680	basic@lordthanatos.com
PowerBasic CC	Fred Luchsinger	ma. t/m vr.	19-21		f.luchsinger@kader.hcc.nl
QBasic QuickBasic	Jan v.d. Linden				j.vd.linden@kader.hcc.nl
Visual Basic voor Windows Visual Basic .NET	Jeroen v. Hezik Marco Kurvers	ma. t/m zo. di. t/m zo.	19-21 19-22	(0346) 214131 (0342) 424452	j.a.van.hezik@kader.hcc.nl m.a.kurvers@hccnet.nl
Basic algemeen, zoals VBA Office Web Design, met XHTML en CSS	Marco Kurvers	di. t/m zo.	19-22	(0342) 424452	m.a.kurvers@hccnet.nl