Basic Bulletin

22ste jaargang, augustus 2015 Nummer 2

Inhoud

Onderwerp	blz.
BBC BASIC for Windows – De library's (5).	4
Ik kan programmeren, makkelijker gezegd dan gedaan.	10
Liberty BASIC API Reference.	12
Informatie: een nieuwe Programmeer Interesse Groep	16
Visual Basic 2010 – (Dialoog)vensters.	17
Appendix – De drie grootste library's (.NET, VCL, XNA)	20

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Jan van der Linden	071-3413679	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	06-30896598	m.a.kurvers@live.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.hcc.nl

Redactioneel

Programmeren is handig om te leren. Je leert de programmeertaal, je leert hoe je programma's schrijft en je leert de technieken die erbij horen, zoals de structuren. Maar als we dit gericht gaan bekijken, komen we in een hele andere programmeerwereld. De kennis die je hebt in programmeren is gewoonweg niet te vergelijken in het bedrijfsleven, of in de hobbytijd. Wie de gezegde '***Er zijn meerdere wegen naar Rome!'*** kent, zal heus niet na de cursus alvast in zijn handen gaan wrijven!

In de Appendix gaat het om de drie grootste library's die door verschillende programmeertalen worden gebruikt. Het .NET kennen we in Visual Basic, maar de andere twee zijn voor u misschien niet bekent. VCL is een library die gebruikt wordt door Borland programmeertalen en XNA is een sjabloon library speciaal geschikt om spellen te kunnen maken.

Marco Kurvers

BBC BASIC for Windows – De library's (5).

Vakken en knoppen.

Windows ™ programma's met drukknoppen, invoervakken, enzovoort, zijn algemeen te integreren binnen de dialoogvensters. *BBC BASIC voor Windows* ondersteunt dit gebruik door middel van de WINLIB2 bibliotheek. Nochtans is het perfect mogelijk om dergelijke items rechtstreeks op te nemen in het uitvoervenster van uw programma.

De **WINLIB5** bibliotheek bevat een reeks procedures en functies voor het opnemen van drukknoppen, invoervakken etc. in uw programma zonder de noodzaak om een dialoog vak om hen heen te maken. De bibliotheek moet worden geladen vanuit uw programma met behulp van de opdracht:

```
INSTALL @lib$+"WINLIB5"
```

De procedures en functies zijn:

- FN_button
- FN_combobox
- FN_editbox
- FN_listbox
- FN_staticbox
- FN_createwindow
- PROC_closewindow
- PROC_setfocus
- FN_setproc

De **WINLIB5A** bibliotheek bevat een identieke set van functies, maar elk heeft een extra eerste parameter: de ingang voor het bovenliggende venster, of te wel de ***parent window handle***. Deze bibliotheek kunt u gebruiken in plaats van WINLIB5, wanneer u de knoppen, vakken of besturingselementen op een venster wilt plaatsen, anders dan op uw belangrijkste uitvoervenster. Misschien wilt u bijvoorbeeld een keuzelijst met invoervak op een werkbalk plaatsen.

De **WINLIB5U** bibliotheek bevat een identieke set van functies naar WINLIB5A, behalve dat die een tekenreeks in Unicode (UTF-8)-indeling ontvangen, waardoor niet-ANSI (bijvoorbeeld de buitenlandse taal) tekens kunnen worden opgenomen.

FN_button(tekst$,x%,y%,cx%,cy%,id%,style%)

Deze functie maakt een rechthoekige drukknop. De tekenreeks **tekst$** geeft de tekst die binnen in de knop moet worden weergegeven, de waarden **x%** en **y%** geven de positie van de knop en de waarden **cx%** en **cy%** geven de grootte van de knop (in pixels, waar **0,0** de linkerbovenhoek van het venster is).

De waarde **id%** is een unieke identifier van de drukknop en kan elke constante hebben die u kiest (met reden) of een waarde geretourneerd van FN_setproc. De waarde **style%** kan nul zijn, maar andere waarden wijzigen het uiterlijk of gedrag van de knop, bijvoorbeeld de waarde &100 (BS_LEFT) zorgt ervoor dat de tekst links-uitgelijnd wordt in plaats van gecentreerd, &80 (BS_BITMAP) maakt een **bitmap knop**, 3 (BS_AUTOCHECKBOX) creëert een **keuzevak** en 9 (BS_AUTORADIOBUTTON) creëert een **optieknop**.

Bij klikken op de knop, treedt er een ON SYS gebeurtenis op, op dezelfde manier als bij een menu selectie, met **@wparam%** is dat hetzelfde als bij de waarde van **id%**.

De functie retourneert de vensteringang van de knop, die nodig is wanneer de knop verwijderd wordt met PROC_closewindow.

Om een image te laden in een afbeeldingsknop, doe als volgt:

```
LR_LOADFROMFILE = 16
BM_SETIMAGE = 247
SYS "LoadImage", 0, bmpfile$, 0, cx%, cy%, LR_LOADFROMFILE TO hbitmap%
SYS "SendMessage", hbutton%, BM_SETIMAGE, 0, hbitmap%
```

De waarde **bmpfile$** is de naam van een Windows™ afbeeldingsbestand waar de image in zit, **cx%** en **cy%** zijn de dimensies van de image in pixels en **hbutton%** is de geretourneerde waarde vanuit FN_button. Zodra u klaar bent met de knop (maar niet eerder), verwijder dan de bitmap handle:

```
SYS "DeleteObject", hbitmap%
```

FN_combobox(“”,x%,y%,cx%,cy%,id%,style%)

Deze functie maakt een combobox. De tekst tekenreeks is niet in gebruik en moet als een lege tekenreeks worden gegeven, de waarden **x%** en **y%** geven de positie van de box en de waarden **cx%** en **cy%** geven de grootte van de box (in pixels, waar **0,0** de linkerbovenhoek van het venster is).

De waarde **id%** is een unieke identifier van de combobox en kan elke constante hebben die u kiest (met reden) of een waarde geretourneerd van FN_setproc. De waarde **style%** kan nul zijn, maar andere waarden wijzigen het uiterlijk of gedrag van de combobox, bijvoorbeeld de waarde 3 (CBS_DROPDOWNLIST) maakt een combobox met *uitschuifbare lijst*. De functie retourneert de vensteringang van de combobox.

Om een lijst van tekenreeksen in een combobox aan te maken, doet u het volgende:

```
CB_ADDSTRING = 323
SYS "SendMessage", hbox%, CB_ADDSTRING, 0, "Combobox item 0"
SYS "SendMessage", hbox%, CB_ADDSTRING, 0, "Combobox item 1"
enz.
```

waarvan **hbox%** de geretourneerde waarde is vanuit FN_combobox. Een geïnitialiseerde (standaard) selectie kan als volgt gemaakt worden:

```
CB_SETCURSEL = 334
SYS "SendMessage", hbox%, CB_SETCURSEL, index%, 0
```

Om te bepalen welk item in een combobox geselecteerd is, doet u het volgende:

```
CB_GETCURSEL = 327
SYS "SendMessage", hbox%, CB_GETCURSEL, 0, 0 TO sel%
```

FN_editbox(tekst$,x%,y%,cx%,cy%,id%,style%)

Deze functie maakt een invoervak. De tekenreeks **tekst$** geeft de oorspronkelijke inhoud van het vak, de waarden **x%** en **y%** geven de positie van het vak en de waarden **cx%** en **cy%** geven de grootte van het vak (in pixels, waar **0,0** de linkerbovenhoek van het venster is).

De waarde **id%** is een unieke identifier van het invoervak en kan elke constante hebben die u kiest (met reden) of een waarde geretourneerd van FN_setproc. De waarde **style%** kan nul zijn, maar andere waarden wijzigen het uiterlijk of gedrag van het vak, bijvoorbeeld de waarde &80 (ES_AUTOHS-

CROLL) laat de inhoud van het vak horizontaal scrollen. De functie retourneert de vensteringang van het vak.

Om de inhoud van een invoervak te kunnen lezen, doe het volgende:

```
DEF FNgettext(hbox%)
LOCAL tekst%
DIM tekst% LOCAL 65535
SYS "GetWindowText", hbox%, tekst%, 65535
= $$tekst%
```

waarvan de parameter **hbox%** de geretourneerde waarde is vanuit FN_editbox.

FN_listbox(“”,x%,y%,cx%,cy%,id%,style%)

Deze functie maakt een lijstvak. De tekenreeks is ongebruikt en moet worden ingesteld als een lege tekenreeks, de waarden **x%** en **y%** geven de positie van het vak en de waarden **cx%** en **cy%** geven de grootte van het vak (in pixels, waar **0,0** de linkerbovenhoek van het venster is).

De waarde **id%** is een unieke identifier van het lijstvak en kan elke constante hebben die u kiest (met reden) of een waarde geretourneerd van FN_setproc. De waarde **style%** kan nul zijn, maar andere waarden wijzigen het uiterlijk of gedrag van het vak, bijvoorbeeld de waarde 2 (LBS_SORT) zorgt ervoor dat de inhoud van het vak gesorteerd wordt. De functie retourneert de vensteringang van het vak.

Om een lijst van tekenreeksen in een lijstvak in te voeren, doe het volgende:

```
LB_ADDSTRING = 384
SYS "SendMessage", hbox%, LB_ADDSTRING, 0, "Listbox item 0"
SYS "SendMessage", hbox%, LB_ADDSTRING, 0, "Listbox item 1"
enz.
```

waarvan **hbox%** de geretourneerde waarde is vanuit FN_listbox. Om te bepalen welk item geselecteerd is, doe u het volgende:

```
LB_GETCURSEL = 392
SYS "SendMessage", hbox%, LB_GETCURSEL, 0, 0 TO sel%
```

De waarde **sel%** geeft de index (beginnend bij 0) van het huidig geselecteerde item of -1 als geen item geselecteerd is.

FN_staticbox(tekst$,x%,y%,cx%,cy%,id%,style%)

Deze functie maakt een statisch vak. De tekenreeks **tekst$** geeft de inhoud (als het een tekstvak is), de waarden **x%** en **y%** geeft de positie van het vak en de waarden **cx%** en **cy%** geven de grootte van het vak (in pixels, waarvan **0,0** de linkerbovenhoek van het venster is).

De waarde **id%** is een unieke identifier van het statische vak en kan elke constante hebben die u kiest (met reden) of een waarde geretourneerd van FN_setproc. De waarde **style%** kan nul zijn, maar andere waarden wijzigen het uiterlijk of gedrag van het vak, bijvoorbeeld de waarde &E (SS_BITMAP) geeft aan dat het vak een bitmap afbeelding kan bevatten. De functie retourneert de vensteringang van het vak.

Om een bitrmap afbeelding in een statisch vak te laden, doe het volgende:

```
LR_LOADFROMFILE = 16
STM_SETIMAGE = 370
SYS "LoadImage", 0, bmpfile$, 0, cx%, cy%, LR_LOADFROMFILE TO hbitmap%
SYS "SendMessage", hbox%, STM_SETIMAGE, 0, hbitmap%
```

De waarde **bmpfile$** is de naam van een Windows™ afbeeldingsbestand waar de image in zit, **cx%** en **cy%** zijn de dimensies van de image in pixels en **hbox%** is de geretourneerde waarde vanuit FN_staticbox. Zodra u klaar bent met het vak (maar niet eerder), verwijder dan de bitmap handle:

```
SYS "DeleteObject", hbitmap%
```

FN_createwindow(class$,tekst$,x%,y%,cx%,cy%,id%,style%,exst%)

Deze functie kan worden gebruikt om onderliggende venstertypes te maken, op een andere manier dan de hierboven beschreven **createwindow**. De parameters hebben hetzelfde doel als voor andere types, behalve dat **class$** de vensterklasse naam is en **exst%** een uitgebreid venster stijl is. De functie retourneert de vensteringang.

Merk op dat de **style%** parameter is exclusief-ORed met **&50000000**, die de numerieke vergelijking is van `WS_CHILD + WS_VISIBLE`. Als u liever niet het venster binnen de grenzen van het ouder wil hebben, voeg dan **&C0000000** `(WS_CHILD + WS_POPUP)` toe aan de opgegeven parameter.

PROC_closewindow(hwnd%)

Deze procedure kan worden gebruikt om elk venster, die gemaakt is met de bovenstaande functies, te sluiten (dat wil zeggen: verwijderen). De waarde **hwnd%** is de vensteringang geretourneerd voor de juiste functie.

PROC_setfocus(hwnd%)

Deze procedure kan worden gebruikt om te bepalen welk venster de input focus heeft (dat wil zeggen: toetsenbordinvoer ontvangt). De waarde **hwnd%** is de handle van het venster dat de focus ontvangt die de geretourneerde waarde is van één van bovenstaande functies of **@hwnd%** in volgorde voor uw BASIC programma om telkens toetsenbordinvoer te kunnen ontvangen.

FN_setproc(PROCnaam)

Deze functie (aanwezig in versie 1.4 of later van WINLIB5) neemt als een parameter de naam van een procedure en retourneert een ID nummer die u kunt gebruiken wanneer een control of een menu item gemaakt wordt. Met deze functie automatiseert het proces van het uitvoeren van een procedure als (bijvoorbeeld) een knop of menu item wordt geklikt, zonder dat ON SYS gebruikt moet worden. Hier zijn enkele voorbeelden van het gebruik ervan:

Na een menu gemaakt is en wanneer in het menu de **Openen** optie geselecteerd wordt, veroorzaakt dat **PROCopenen** die uitgevoerd wordt en bij de optie **Afsluiten** veroorzaakt dat **PROCafsluiten** die uitgevoerd wordt:

```
SYS "CreatePopupMenu" TO hfile%
SYS "AppendMenu", hfile%, 0, FN_setproc(PROCopenen), "&Openen"
SYS "AppendMenu", hfile%, 0, FN_setproc(PROCafsluiten), "&Afsluiten"
```

Na de knoppen in een dialoogvenster gemaakt zijn en wanneer geklikt wordt op **Button 1**, veroorzaakt dat **PROCbutton1** die uitgevoerd wordt en klikken op **Button 2** veroorzaakt dat **PROCbutton2** die uitgevoerd wordt:

```
INSTALL @lib$+"WINLIB2"
dlg% = FN_newdialog("Button test", 200, 100, 100, 100, 8, 1000)
PROC_pushbutton(dlg%,"Button 1",FN_setproc(PROCbutton1),20,10,64,16,0)
PROC_pushbutton(dlg%,"Button 2",FN_setproc(PROCbutton2),20,32,64,16,0)
```

Na de knoppen gemaakt zijn op uw hoofdvenster en wanneer geklikt wordt op **Button 3**, veroorzaakt dat **PROCbutton3** die uitgevoerd wordt en klikken op **Button 4** veroorzaakt dat **PROCbutton4** die uitgevoerd wordt:

```
hbutt3% = FN_button("Button 3",300,20,100,24,FN_setproc(PROCbutton3),0)
hbutt4% = FN_button("Button 4",300,90,100,24,FN_setproc(PROCbutton4),0)
```

Als u de waarden wilt weten van **@wparam%** en/of **@lparam%** (onwaarschijnlijk in het geval van menuselecties of drukken op de knoppen, maar mogelijk met andere controls) kunt u dit regelen bij het doorgeven aan de procedure door een paar haakjes toe te voegen wanneer u FN_setproc aanroept, zie hieronder:

```
hedit% = FN_editbox("",300,20,100,24,FN_setproc(PROCeditbox()),0)
```

Vervolgens moet u de procedure met twee parameters definiëren:

```
DEF PROCeditbox(W%, L%)
```

waarvan **W%** en **L%** de waarden van @wparam% en @lparam% respectievelijk ontvangen.

Sprites.

De **SPRITELIB** bibliotheek bevat een set van procedures en functies voor het maken en beheren van sprites. Een sprite is een grafisch object dat wordt weergegeven vóór en onafhankelijk van, wat anders geplot is in het BBC BASIC-venster. Een sprite kan elke vorm en patroon hebben: wanneer het ondoorzichtig is, zal het volledig verdwijnen als het achter zit (bijvoorbeeld een ander object of venster) en wanneer het transparant is, zal de normale vensterinhoud doorschijnend worden weergegeven. U kunt een sprite verplaatsen zonder te hoeven bekommeren om het opnieuw tekenen van de achtergrond die *geopenbaard* wordt.

De library moet worden geladen vanuit uw programma met gebruik van dit commando:

```
INSTALL @lib$+"SPRITELIB"
```

De procedures en functies die daar vandaan komen zijn:

- FN_initsprites
- FN_createsprite
- FN_createspritefrombmp
- PROC_movesprite
- PROC_updatesprite
- PROC_exitsprites

FN_initsprites(nsprite%)

Deze functie initialiseert het sprite systeem. Het heeft één parameter nodig: het maximum aantal sprites die zal moeten worden weergegeven. Deze parameter moet zo klein mogelijk worden ingesteld, want hoe groter het aantal sprites des te meer het geheugen gebruikt en hoe groter het vertragende effect op het scherm.

```
IF FN_initsprites(2) = 0 STOP
```

De geretourneerde waarde is TRUE als het sprite systeem met succes geïnitialiseerd is en FALSE als het niet zo is. De functie zal falen als de opgegeven parameter nul is, of als er onvoldoende geheugen beschikbaar is.

FN_createsprite(sprite%,file$,width%,height%)

Deze functie maakt een sprite van een bepaalde grootte en met een opgegeven vorm en uiterlijk. Het vereist vier parameters: het sprite nummer (die tussen nul en één minder dan de parameter, geleverd door **FN_initsprites**, moet liggen), de naam van het afbeeldingsbestand die de sprite bevat (en de transparantie), de breedte van de sprite en de hoogte van de sprite (beide in pixels).

Het bestand moet een Icon formaat bestand zijn (meestal met de extensie .ICO) gegenereerd met behulp van een geschikte icon editor. *BBC BASIC for Windows* wordt geleverd met een eenvoudige icon editor (ICONEDIT.BBC), die zal volstaan als er geen andere editor beschikbaar is. Onder normale omstandigheden moet u dezelfde dimensies opgeven in de **FN_createsprite** aanroep als toen het pictogram werd gemaakt, maar als u dat niet doet, zal de sprite worden geschaald naar de opgegeven grootte (met daarmee enkel gepaard verlies van kwaliteit).

```
ok% = FN_createsprite(0, "bbcmicro.ico", 64, 64)
```

De geretourneerde waarde is niet-nul als de sprite succesvol gemaakt is en nul als dat niet het geval is. De meest waarschijnlijke reden voor de functie te mislukken is als het bestand niet bestaat of niet een geschikt pictogram-formaat (icon) bestand is.

De sprite wordt niet weergegeven totdat **PROC_move** of **PROC_updatesprite** uitgevoerd is.

FN_createspritefrombmp(sprite%,bmpfile$)

Deze functie is vergelijkbaar met **FN_createsprite**, behalve dat in plaats van een icon (.ICO) bestand dat wordt vereist een bitmap (.BMP) bestand moet worden opgegeven, en er zijn geen parameters van de grootte nodig (de sprite kan worden geschaald door de juiste breedte en hoogte waarden doorgegeven aan **PROC_updatesprite**). De functie maakt het eenvoudig om de sprite te maken op runtime, aangezien een geschikt BMP-bestand kan worden gegenereerd met behulp van de ***GSAVE** opdracht. De sprite kan elke grootte (binnen reden) en is niet beperkt tot 256 x 256 pixels zoals het geval voor een pictogrambestand zou zijn.

Het BMP bestand moet een enkele bitmap hebben die **twee keer de hoogte** van de sprite is. De bovenste helft is het transparante sprite masker en de onderste helft is de sprite afbeelding. Hier links ziet u een voorbeeld van een geschikte bitmap.

Het masker is wit waar de sprite volledig transparant is en zwart waar de sprite volledig ondoorzichtig is (tussentijdse transparantie waarden worden niet ondersteund). In gebieden waar de sprite doorzichtig is moet de afbeelding zwart zijn.

PROC_movesprite(sprite%,xpos%,ypos%,show%)

Deze procedure toont (of verbergt) een sprite en verplaatst het naar een opgegeven positie op het scherm. Het vereist vier parameters: het sprite nummer, de horizontale en verticale coördinaten van de sprite (in BBC BASIC grafische units) en een waarde die bepaald of de sprite weergegeven wordt (1) of niet (0). De coördinaten verwijzen naar de linkerbovenhoek van de sprite (die strookt met tekst-tekens die zijn uitgezet in de **VDU 5**-modus).

```
PROC_movesprite(0, 200, 200, 1)
```

In het geval van een geanimeerde sprite (. ANI bestand) is de show% parameter het vereiste frame-nummer (+1), dus 1 zorgt ervoor dat het eerste frame moet worden weergegeven, 2 zorgt ervoor dat het tweede frame moet worden weergegeven, enzovoort.

Sprites zijn onaangetast door het huidige venster (indien aanwezig) en altijd weer te geven voor elke andere afbeeldingen of tekst. Sprites hebben een vooraf gedefinieerde prioriteitsvolgorde, zodanig dat als ze elkaar overlappen, een hoger genummerde sprite altijd boven op een lager genummerde sprite wordt weergegeven.

PROC_updatesprite(sprite%,xpos%,ypos%,show%,wide%,high%)

Deze procedure is vergelijkbaar met **PROC_movesprite**, behalve dat het extra parameters heeft, **wide%** en **high%**. Deze parameters mogen ingesteld worden als gewenste breedte en hoogte waar-den van de sprite, respectievelijk gegeven in pixels. Zo kan bijvoorbeeld een sprite kleiner of groter worden gemaakt als het wordt verplaatst.

Daarnaast, door negatieve waarden op te geven voor **wide%** en/of **high%** is het mogelijk de sprite te spiegelen over zijn horizontale of verticale as (of beide). Merk op dat deze functie alleen voor sprites werkt met een 1-bit masker (niet een lineaire alpha masker) en is gebaseerd op een ongedocumen-teerde functie van Windows.

PROC_exitsprites

Wanneer u klaar bent met de sprites, kunt u ze uitschakelen.

```
PROC_exitsprites
```

Omdat de sprite routines ruimte gebruiken op de *heap*, is het essentieel dat u **PROC_exitsprites** aan-roept voordat u een **CLEAR**, **CHAIN** of een **RUN** statement uitvoert. Doet u het niet, dan zal zeer waarschijnlijk *BBC BASIC for Windows* crashen.

Ik kan programmeren, makkelijker gezegd dan gedaan.

Voorwoord
Tijdens de leuke - **Ik kan programmeren.** – stukjes, werd veel verteld over hoe leuk het is om te kun-nen programmeren, als je het geleerd hebt. Ja, dat klopt. Je kunt het als je het geleerd hebt…. of niet!

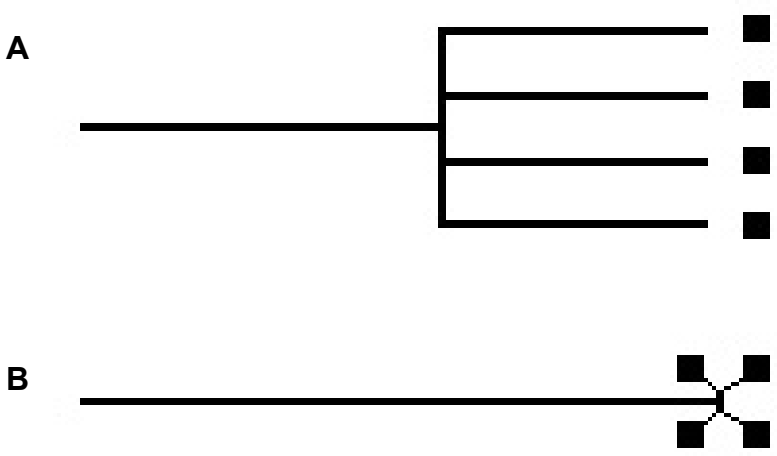
Wanneer je kunt programmeren, lijkt het zo makkelijk gezegd: ik heb het toch geleerd?! Al doende leert men, maar breng mij maar niet van de wijs dat een programma schrijven voor iemand net zo ge-makkelijk lijkt zoals je het tijdens de cursus geleerd hebt. Bij het schrijven van een programma zijn er meerdere wegen naar Rome, en dat is precies het probleem waar men veel tegenaan loopt. Tijdens een cursus leer je een programmeertaal, maar je leert alleen maar een rechte pad. Krijg je in de echte programmeerwereld een opdracht, dan moet je ook de zijwegen kennen. Een bepaalde richting die gegeven wordt moet je nemen. Dwaal je af, dan doet het programma niet naar wens van de klant.

Een structuur kiezen
Een reis plannen naar Rome kan op verschillende manieren. Iedereen heeft zijn eigen programma opbouw, of de ene nou via Frankrijk gaat en de ander via Duitsland, ze komen allebei bij hun bestemming aan. Dit gaat met programmeren net zo. Niemand kiest dezelfde manier van programmeren. Dat mag best. Een voorbeeld is het gebruik van een lusstructuur. Een While … Wend lus is een structuur waarvan het einde nog niet bekend is, maar weet de programmeur het echter wel, dan is het gebruik van een For … Next lus veel sympathieker. Weet de programmeur het einde niet (het moet met een voorwaarde worden bepaald), dan is het weer sympathieker om gebruik te maken van een While … Wend lus. Doet de programmeur dat niet, dan moet de lus afgebroken worden met een Exit For of nog erger; het gebruik van een GoTo statement.

In het bedrijfsleven moeten programma's goed en snel kunnen werken. Door functionele structuren te gebruiken, is de chef en ook de klant tevreden, maar neem je een weg naar Rome via Polen dan is de tevredenheid snel over. Zulke structuren nemen veel code, tijd en geld in beslag en dat is niet de bedoeling. Maak dus goede structuren in je programma's. Kleine verschillen in de lusstructuren en keuzestructuren geeft niet, maar als het om procedures en functies, en zelfs om klassen gaat, dan is het andere koek.

Vooral klassenobjecten moeten aan de eisen voldoen. Een programma maken zonder klassen is een schuur met honderd fietsen met maar één wiel. Maak dus niet steeds het wiel opnieuw in je programma.

Modules, klassenmodules en klassenunits
Maak je wel gebruik van klassen, dan ben je goed bezig. Een goed voorbeeld is het snel kunnen werken voor een postbode. Post bezorgen in een flat of appartementengebouw is veel fijner dan post bezorgen in rijtjeshuizen waarvan elk huis zijn eigen 'ingang' heeft naar het postvak. Kijk maar eens naar onderstaande afbeeldingen.



Zou je het fijn vinden om telkens heen en weer te moeten lopen, zoals afbeelding **A** laat zien? Afbeelding **B** laat zien dat het in een flat of appartementengebouw niet hoeft, omdat er maar één ingang is met één kast met meerdere postvakken. Post erin, klaar is kees.

Wat bedoel ik hier precies mee?
Afbeelding **A** is een afbeelding zonder een goede klassenstructuur. Door maar één programma te gebruiken, waarvan alle klassen (de huizen) niets met elkaar te maken hebben, zou telkens de code (de ingang waar de postbode steeds heen moet lopen) opnieuw gemaakt moeten worden. Bij afbeelding **B** is er een code, waarvan alle klassen wel met elkaar te maken hebben (appartementen in één gebouw). De code hoeft niet steeds opnieuw gemaakt te worden en de postbode hoeft alleen maar voor al die woningen de post erin te doen. De postbode is dan ook veel sneller klaar dan wanneer hij een hele straat af moet lopen en elke keer naar het volgende pad moet lopen om de post in het vak te kunnen doen. Natuurlijk, in het echt moet dat ook gedaan worden, maar hier wil ik alleen een 'Neder-

landstalige techniek' geven om te laten zien wat men bedoeld met 'het wiel opnieuw uit moeten vinden'.

Visual Basic 6 en andere BASIC dialecten kennen helaas alleen methode **A**. In Visual Basic 6 moeten we gebruik maken van een globale module en klassenmodules. Helaas wordt de globale module wel eens volgestopt met variabelen, subroutines en functies. Het is beter om het programma te ordenen en de juiste subroutines en functies in de juiste klassenmodules te maken. Dit is ook weer waar ik het in het begin over had, de meerdere wegen naar Rome. Door alleen gebruik te maken van één module, zal de reis naar Rome veel langer duren dan wanneer alles prima geordend wordt.

Klassenunits
Afbeelding **B** kunnen we nog uitbreiden. Bedrijven hebben niet altijd bij de hoofdingang een kast met postvakken. Zij maken gebruik van een *postbus*. Door de brief met het juiste postnummer daar in te stoppen, weet het bedrijf naar welk kantoor het moet. Al deze kantoren staan met de postbus in verbinding. Dit kunnen we een *library* of een boomstructuur noemen. In Visual Studio met het .NET Framework library, wordt gebruik gemaakt van klassenunits. Het verschil met een klassenmodule is nu, dat in een unit meerdere klassen gemaakt kunnen worden. Een klassenmodule was eerder een klasse zonder een geraamte die niet te wijzigen was. De OOP en polymorfisme geven daarbij een prima zonnetje naar Rome om uiteindelijk programma's klaar te hebben die het juiste doel hebben, zoals de klant het hebben wil.

Liberty BASIC API Reference.

IsWindowVisible
Deze functie retourneert of het venster zichtbaar is of niet, waarvan de handle "h" is. Het resultaat is niet-nul als het venster zichtbaar is. De functie kan ook niet-nul resulteren als het venster compleet is verduisterd bij andere vensters, omdat dit de zichtbare status is die bepaald wordt en niet het actuele uiterlijk van de desktop. Als het venster verborgen was met een ShowWindow aanroep, zal het resultaat van de functie nul zijn.

```
calldll #user32, "IsWindowVisible", h as ulong, result as boolean
```

IsZoomed
Deze functie retourneert niet-nul als het venster waarvan de handle is "h" gemaximaliseerd is.

```
calldll #user32, "IsZoomed", h as ulong, result as boolean
```

lstrcmp en lstrcmpi
"lstrcmp" vergelijkt twee strings. De returnwaarde is 0 als de strings gelijk zijn, kleiner dan 0 als de eerste string kleiner is en groter dan 0 als de eerste string groter is. Deze vergelijking is hoofdletter gevoelig. Gebruik lstrcmpi voor een niet hoofdletter gevoelige vergelijking.

```
calldll #user32, "lstrcmpA",_      'hoofdletter gevoelig
s1$ as ptr,_         'eerste string om te vergelijken
s2$ as ptr,_         'tweede string om te vergelijken
result as long       'vergelijkingscode

calldll #user32, "lstrcmpiA",_     'niet hoofdletter gevoelig
s1$ as ptr,_         'eerste string om te vergelijken
s2$ as ptr,_         'tweede string om te vergelijken
result as long       'vergelijkingscode
```

MessageBeep

Deze functie speelt een waveform geluid gecorrespondeerd bij een gegeven systeem waarschuwingshoogte. Het geluid voor elke waarschuwingshoogte is voorgeprogrammeerd bij een ingang in het [geluids]deel van de WIN.INI initialisatiebestand. Als het niet het opgegeven waarschuwingsgeluid kan afspelen, zal MessageBeep overgaan om het standaard systeemgeluid af te spelen. Kan het ook niet het standaard systeemgeluid afspelen, dan zal de functie een standaard geluidtoon produceren bij gebruik van de computerspeaker.

```
flag = 1  'Produceert een standaard geluidtoon bij gebruik van de computer-
speaker.

flag = _MB_ICONASTERISK  'Speelt het voorgeprogrammeerd geluid bij de Syste-
mAsterisk ingang in het [geluids]deel van WIN.INI.

flag = _MB_ICONEXCLAMATION    'Speelt het voorgeprogrammeerd geluid bij de
SystemExclamation ingang in het [geluids]deel van WIN.INI.

flag = _MB_ICONHAND 'Speelt het voorgeprogrammeerd geluid bij de SystemHand
ingang in het [geluids]deel van WIN.INI.

flag = _MB_ICONQUESTION  'Speelt het voorgeprogrammeerd geluid bij de Sys-
temQuestion ingang in het [geluids]deel van WIN.INI.

flag = _MB_OK  'Speelt het voorgeprogrammeerd geluid bij de SystemDefault
ingang in het [geluids]deel van WIN.INI.

flag = _MB_OK
calldll #user32, "MessageBeep", flag as long, result as boolean
```

MessageBox

Het berichtvenster MessageBox functie maakt, geeft weer en zorgt voor een berichtbalkvenster. De berichtbalk heeft een applicatie-gedefinieerd bericht aanwezig in de string Message$ en een titel, aanwezig in de string Title$, plus enkele combinaties van voorgedefinieerde iconen en drukknoppen hieronder beschreven. De “h” parameter is een venster handle of het kan worden gezet als 0. De returnwaarde bepaald welke knop ingedrukt was door de gebruiker. Kies één icoontype en één of meer drukknoppen, met extra optionele parameters hieronder weergegeven:

Iconen:

- _MB_ICONASTERISK (zelfde als _MB_ICONINFORMATION) ‘een kleine letter “i”
- _MB_ICONEXCLAMATION ‘een uitroepteken
- _MB_ICONHAND (zelfde als _MB_ICONSTOP) ‘cirkel met X [Win95/98]
- _MB_ICONQUESTION ‘een vraagteken

Drukknop sets:

- _MB_OK
- _MB_OKCANCEL
- _MB_YESNO
- _MB_YESNOCANCEL
- _MB_RETRYCANCEL
- _MB_ABORTRETRYIGNORE

Optionele parameters die toegevoegd mogen worden aan wtype met OR:

- _MB_APPLMODAL ‘ de standaard, vak is applicatie modaal
- _MB_TASKMODAL ‘taak modaal
- _MB_SYSTEMMODAL ‘systeem modaal, bevriest het systeem totdat het beantwoord is

_MB_DEFBUTTON1 ‘de standaard, eerste knop geeft de invoer terug als gebruiker op ENTER drukt
_MB_DEFBUTTON2 ‘tweede knop geeft de invoer terug als gebruiker op ENTER drukt
_MB_DEFBUTTON3 ‘derde knop geeft de invoer terug als gebruiker op ENTER drukt

```
Title$ = "Error in Processing!"

Message$ = "The operation was not completed."

wtype = _MB_ICONSTOP or _MB_RETRYCANCEL or _MB_DEFBUTTON2

calldll #user32, "MessageBoxExA", _
     h as ulong, _        'venster handle
     Message$ as ptr, _   'gewenste tekstbericht
     Title$ as ptr, _     'gewenste titelbalk bijschrift
     wtype as long, _     'flag voor icoon en knoppen
     language as word, _ 'taal identifier
     result as long       'retourneert de actiecode
```

result:
1 = OK was geklikt
2 = Annuleren was geklikt
3 = Afbreken was geklikt
4 = Opnieuw was geklikt
5 = Overslaan was geklikt
6 = Ja was geklikt
7 = Nee was geklikt

MoveWindow
De MoveWindow functie verandert de positie en dimensies van een venster of control. Voor vensters zijn de posities en dimensies relatief vanaf de linker bovenhoek van het scherm. Voor controls zijn ze relatief vanaf de linker bovenhoek van het cliëntgebied van het oudervenster. De functie retourneert niet-nul als het succesvol is verlopen.

```
calldll #user32, "MoveWindow", _
     h as ulong, _             'venster of control handle
     xOrg as long, _           'nieuwe x positie linksboven
     yOrg as long, _           'nieuwe y positie linksboven
     width as long, _          'nieuwe breedte
     height as long, _         'nieuwe hoogte
     1 as boolean, _           '1 = vernieuw nu het venster
     result as boolean         'niet-nul (true) wanneer succesvol
```

PaintDesktop
De PaintDesktop functie vult het geknipte gebied in het opgegeven device context met het bureaubladpatroon of behang. De functie is voornamelijk opgenomen ten behoeve van shell desktops.

```
calldll #user32, "PaintDesktop", hDC as ulong, re as boolean
```

ReleaseDC
Deze functie geeft het gegeven device context vrij, (hDC) maakt het vrij voor gebruik bij andere applicaties. De applicatie moet de functie ReleaseDC aanroepen voor elke aanroep van de GetWindowDC functie en voor elke aanroep van de GetDC functie die de gemeenschappelijke device context retourneert.

```
calldll #user32, "ReleaseDC", _
     h as ulong, _        'venster handle
     hDC as ulong, _      'device context handle
     result as long       '1=succes
```

ScreenToClient
Deze functie leest de x, y schermwaardes die je vult met de struct “point” en vult de struct met het correspondeerde cliëntgebied coördinaten voor het venster waarvan de handle is “h”.

```
struct point, x as long, y as long
point.x.struct = 20      'scherm x positie om te vertalen
point.y.struct = 55      'scherm y positie om te vertalen
calldll #user32, "ScreenToClient", _
     h as ulong, _        'venster handle
     point as struct, _   'naam van de struct
     r as boolean         'niet-nul=succes
clientX = point.x.struct
clientY = point.y.struct
```

SetCursorPos
Gebruik deze functie om de positie van de muiscursor om te zetten in xLoc, yLoc schermcoördinaten. Vertaal de coördinaten op de juiste manier met ScreenToClient of ClientToScreen. Merk op dat CURSOR naar de muispointer verwijst en niet naar de knipperende tekstcursor in een tekstvenster.

```
calldll #user32, "SetCursorPos", _
     xLoc as long, _      'x locatie
     yLoc as long, _      'y locatie
     result as boolean    'niet-nul=succes
```

SetFocus
Deze functie plaatst de invoerfocus in het gegeven venster of control. Alle latere toetsenbord invoer is direct op dit venster. Het venster dat daarvoor de invoerfocus had, zal het verliezen. De geretourneerde waarde identificeert het venster dat daarvoor de invoerfocus had, maar dan wel alleen succesvol mocht het venster dat hebben gehad.

```
calldll #user32, "SetFocus", h as ulong, result as ulong
```

Wordt vervolgd

Informatie: een nieuwe Programmeer Interesse Groep.

In dit korte stukje wil ik vertellen over een mogelijkheid dat wel eens een nuttig idee zou kunnen zijn. In maart was ik in De Bilt met de CompUsersDag. Het blijkt dat twee programmeertaalgroepen, C en Pascal, samen zijn als één Interesse Groep. De voorzitter van de programmeergroep vroeg aan mij of het misschien een idee is om ook Basic bij de programmeergroep aan te sluiten.

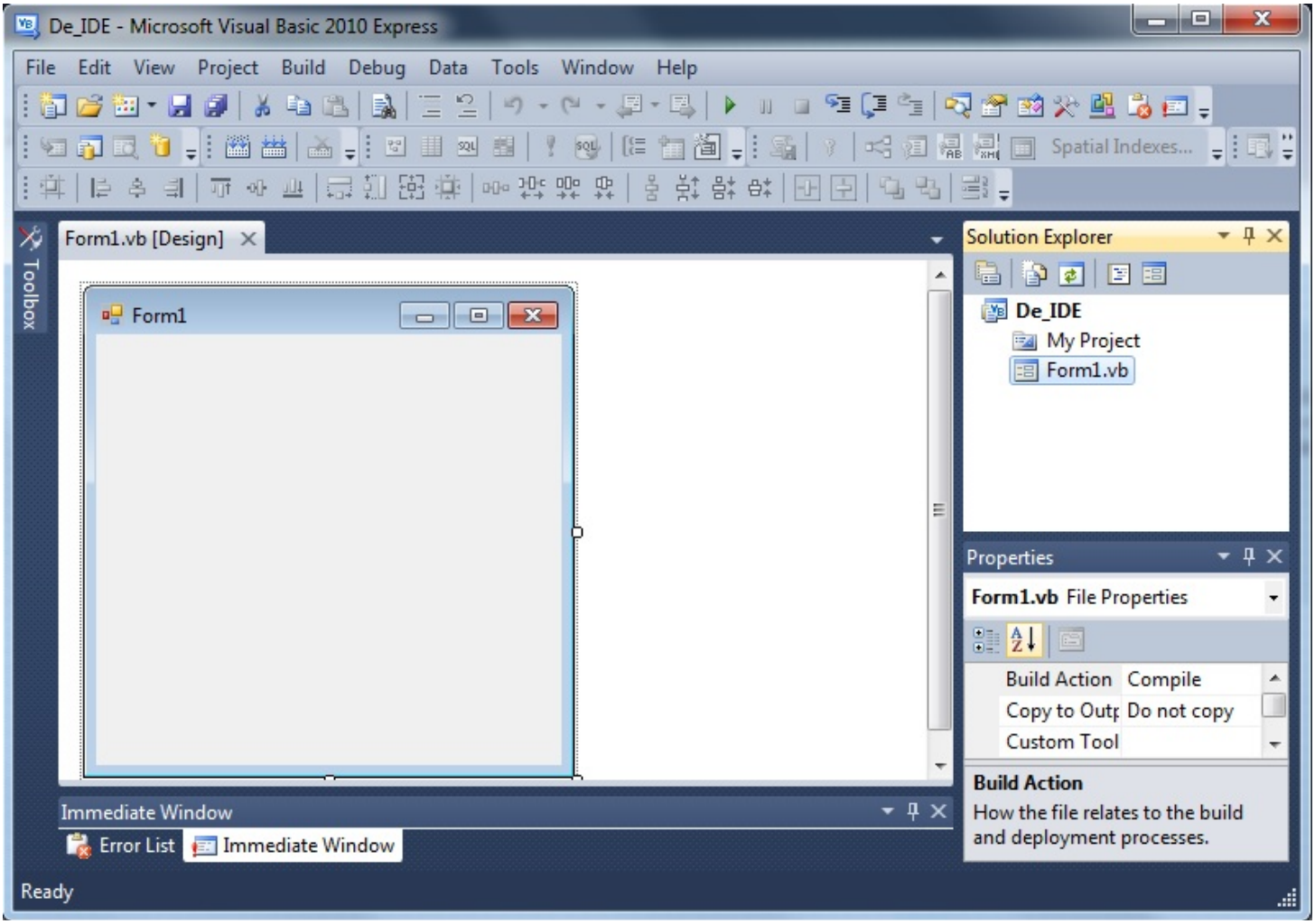
Nu is mijn vraag: “Wat vindt u daarvan?” Laat het ons weten. Stuur uw reactie naar de voorzitter of redacteur. Ik kan in ieder geval zeggen dat ik er wel wat voor voel, vooral omdat ik meer weet dan ik in deze nieuwsbrieven schrijf. Ik ben niet alleen een Basic man, maar ik programmeer ook in Delphi, C# en scripttalen (waaronder GML die bekend is in het programma Game Maker).

Een voorproefje over andere programmeertalen kunt u vinden in de Appendix. Daar ga ik het vooral hebben over de XNA library.

Marco Kurvers
Redacteur

Visual Basic 2010 – (Dialoog)vensters.

Hieronder zien we nog eens het volledige scherm van Visual Basic 2010. Hier zijn alle gereedschappen die nodig zijn om vensters te kunnen bouwen en te besturen. In het volgende Bulletin nummer gaan we eens kijken hoe de besturing in elkaar zit. De Visual Basic 6 gebruikers zullen een totaal ander soort Basic code zien, dat erg op C++ lijkt. Helaas wel, maar de statements niet waardoor het maar eventjes wennen is.

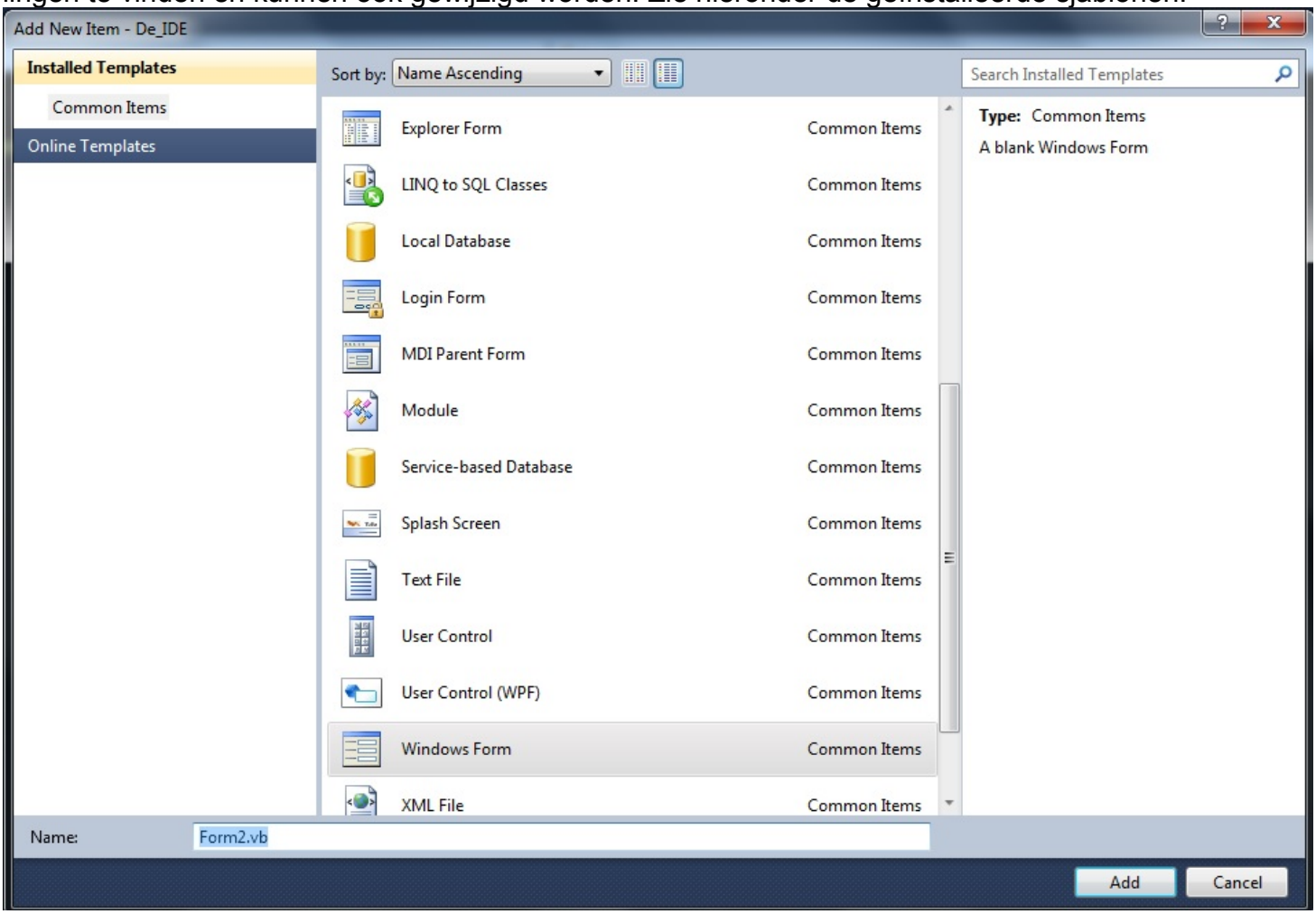


Waar het nu om gaat is het venster Form1 waar we besturingselementen, of ook genoemd: *controls*, op kunnen plaatsen. Wat kunnen we er zowel op plaatsen? Om daar achter te komen, moeten we weten wat voor *karakter* het venster heeft. Dit kan bij de Properties, aan de rechterkant, ingesteld worden. Hieronder zijn er mogelijkheden wat voor karakter we een venster kunnen geven:

- Hoofdvenster, of ook wel genoemd: MDI formulier. Dit is het applicatievenster van het hele project dat gemaakt wordt. Elke MDI kan meerdere ander soort vensters hebben, zoals een toolvenster en zoonvensters. Een project kan maar één MDI formulier hebben.
- Zoonvenster, of ook wel genoemd: child formulier. Een zoonvenster behoort altijd tot het hoofdvenster. Er kunnen meerdere zoonvensters in het project voorkomen.
- Dialoogvenster. Meestal ook wel gegevensvenster genoemd, omdat deze gemaakt wordt speciaal geschikt voor het invoeren van gegevens. In het meeste geval heeft een dergelijk venster de knoppen OK en Annuleren, maar ook de knop Toepassen kan erbij staan. De knoppen staan standaard altijd rechtsonder. Een dialoogvenster heeft alleen de sluitknop rechtsboven en kan niet standaard de focus verliezen voordat het venster gesloten wordt.

Er zijn nog meer soorten vensters, zoals het toolvenster. Deze vensters hebben een kleinere sluitknop, maar ze behoren echter tot het dialoogvenster. Deze vensters kunnen wel hun focus verliezen.

Sjabloonvensters
Kiezen we bij het menu Project -> Add Windows Form, dan zal niet meteen een lege Form verschijnen. Visual Basic heeft sjabloonvensters. Zo is direct een dialoogvenster te kiezen waar de twee standaardknoppen OK en Annuleren op staan, of een About venster met een picturebox en labels met een sluitknop. Het About venster heeft ook alvast voorgeprogrammeerde code die er voor zorgt dat de projectgegevens zullen verschijnen als het project uitgevoerd wordt. De gegevens zijn in de projectinstellingen te vinden en kunnen ook gewijzigd worden. Zie hieronder de geïnstalleerde sjablonen:



Elk sjabloon heeft het volgende doel:

- **Explorer Form**
  Dit venster lijkt veel op de Windows Verkenner. Alleen dan het geraamte van het venster. Het is niet zo dat het venster ook daadwerkelijk als een verkenner zal gedragen. Dat bepaalt u. Het venster wordt gemaakt en in tweeën gedeeld met een TreeView control en een ListView control met daartussen een splitter, die ervoor zorgt dat de twee controls groter en kleiner verschoven kunnen worden. De Windows Verkenner is een heel goed voorbeeld daarvan.
- **LINQ to SQL Classes**
  Zorgt voor het converteren van buitenstaande databases naar SQL klassenobjecten.
- **Local Database**
  Maakt een interne database aan alleen bedoeld voor het huidige project.
- **Login Form**
  Maakt een login dialoogvenster met standaard controls die het meest geschikt zijn voor het inloggen met gebruikersgegevens. Het gaat standaard altijd om *Gebruikersnaam* en *Wachtwoord* met de twee knoppen *OK* en *Annuleren*. U mag het loginvenster natuurlijk altijd aanpassen naar uw behoefte.

- **MDI Parent Form**
  Deze sjabloon maakt het hoofdvenster van het project. Alles wordt meteen klaargemaakt, zoals het hoofdmenu met een werkbalk en de daarbij behorende besturingscontrols. Helaas zal het hoofdmenu helemaal in het Engels worden gemaakt. De menu items kunnen gewijzigd worden om toch het menu in het Nederlands om te kunnen zetten. Wilt u de poespas helemaal niet, dan is het beter om gewoon een leeg formulier aan te maken en het formulier in te stellen als een MDI Parent Form.
- **Module**
  Hiermee wordt een extra codepagina aangemaakt. Met een module kunnen initialisaties en klassen worden gedefinieerd. Het is verstandig om minstens één module in een project op te nemen, die voor het aanmaken en starten van het MDI formulier zorgt.
- **Service-based Database**
  Standaard service database voor het beheren van netwerken en hardware. Zelf heb ik deze nog nooit gebruikt. Ik kan dus hier niet veel over vertellen.
- **Splash Screen**
  Hiermee kunnen startvensters worden gemaakt met een afbeelding van het programma en eventueel wat tips en hulp voor de gebruiker. Elke keer als het programma wordt gestart, worden de tips willekeurig bepaald.
- **Tekst File**
  Hiermee wordt een tekstbestand aangemaakt waar initialisatiegegevens ingezet kunnen worden. Deze gegevens zijn altijd constant, maar het bestand mag ook in code bijgewerkt worden.
- **User Control**
  Dit opent een nieuw venster die als een nieuwe control kan functioneren. U kunt van alles op het venster plaatsen. Wordt deze gecompileerd en gebouwd als een control bestand, dan zal alles samen werken als één object. Later wil ik hier op terugkomen, maar eerst wat andere onderwerpen over Visual Basic 2010.
- **User Control (WPF)**
  Zelfde manier als de vorige, maar nu is de nieuwe control alleen geschikt voor de Internet Browser onderdelen.
- **Windows Form**
  Dit is het standaard Windows Formulier die helemaal leeg is. U moet zelf de eigenschappen instellen op de manier zoals het formulier moet werken.
- **XML File**
  Naast een tekstbestand kan er ook een bestand worden aangemaakt voor databases en internetpagina's. Niet om echte html pagina's te kunnen maken, maar om gegevensstructuren te kunnen definiëren. Visual Basic 2010 heeft een interne XML editor. Voor gebruik van een database raad ik echter aan om gebruik te maken van het DataSet object. XML maakt ook gebruik van tags, net zoals HTML. Echter, een XML tag kan elke veldnaam zijn.

Elk venster heeft een besturing nodig. Het belangrijkste is de gegevensoverdracht. Gegevens moeten overgedragen worden naar het venster die geopend wordt. Zodra de gegevens klaar zijn om vanuit het venster terug te lezen, is er echter een klein probleempje. Hoe lezen we de gegevens vanuit het venster? Als op de OK knop wordt geklikt, dus zou men denken om de overdracht te laten gebeuren in de Click gebeurtenis van de OK knop. Natuurlijk, ook die weg naar Rome kan werken, maar in principe is dat niet de bedoeling. Beide overdrachten moeten *buiten* het venster plaatsvinden. Hoe kunnen we dan Basic laten weten wanneer de gegevens vanuit het venster teruggelezen kunnen worden?

In Visual Basic 6 moet een boolean variabele in het venster gedeclareerd worden. In de OK knop wordt de variabele toegekend met de waarde True en in de Annuleren knop de waarde False. In beide knoppen moet ook het statement Hide staan. Dat statement zorgt ervoor dat het venster van het scherm verdwijnt als op zo'n knop geklikt wordt. Buiten het venster, waar het openen van het venster plaatsvond, moet met die variabele gecontroleerd worden of die True of False is. Is de waarde True dan mogen de gegevens vanuit het venster gelezen worden, anders niet. Daarna, en dat mag niet vergeten worden, moet het venster gesloten worden. Ook al wordt het statement Hide gebruikt, het ven-

ster blijft aanwezig. Na het statement *vensternaam*.Close zal het venster ook niet meer verborgen zijn. Dankzij het statement Hide blijft er toegang tot het venster. Na Close is er geen bereik meer tot het venster.

Appendix – De drie grootste library's (.NET, VCL, XNA).

In de programmeerwereld bestaan er drie grote boomstructuren. Dankzij deze boomstructuren kunnen we programma's maken die in Windows gestart kunnen worden. De library's bestaan uit onderdelen die in de Windows map te vinden zijn. Echter werkt een library als een echte boom. Alles stamt af van de wortel. De wortel is de basis van alles wat er in de *takken* werkt.

Het .NET Framework library wordt gebruikt in Visual Studio. Met de programmeertalen Basic, C++, C#, F#, kan alleen het .NET Framework worden gebruikt. Zoals het Framework in elkaar zit, zo zullen de besturingselementen zich ook gedragen. Elk besturingselement stamt af, of erft alles, van de basisklasse System.Object.

Andere Basic dialecten die geen library hebben, roepen alles aan vanuit de API dll's. Liberty Basic is er een voorbeeld van. Visual Basic 6 zorgt ervoor dat de programmeur dat niet hoeft te doen. Alle controls kunnen gewoon geprogrammeerd worden en het uiteindelijke programma kan dan in Windows worden gestart. Visual Basic 6 heeft echter ook geen library, waardoor alles in zwarte dozen (sjablonen) wordt gestopt. Het nadeel is dat elke control zijn eigen eigenschappen heeft. Er wordt niets geërfd vanuit een basis control.

Visual C# ondersteunt zowel het .NET Framework als het XNA Framework. Deze library's kunnen worden gekozen bij het starten van Visual C#, tenzij XNA op het systeem aanwezig is.

VCL
VCL is een afkorting van Visual Component Library ontwikkeld door Borland. Borland bracht het uit voor C++ Builder in 2006, maar later toen CodeGear kwam, werd het ook uitgebracht voor Delphi. Borland Delphi 7 was de laatste zelfstandige IDE programmeertaal voordat CodeGear RAD Studio werd uitgebracht met Delphi 2007 en C++ Builder 2007. Beide maken gebruik van het VCL.

VCL biedt vele hulpmiddelen met programmeren. De library heeft een handige TList collectieklasse, die net zo werkt als de collectieklasse in .NET. Het verschil is dat TList een heel recordtype en zelfs een andere klasse kan inkapselen.

De structuur van VCL werkt verder op dezelfde manier, met een wortel. Net zoals bij .NET is de wortel een TObject klassentype.

XNA
Wie zou er geen spel willen maken? Dat maakt het programmeren leuker. Helaas is spellen programmeren niet makkelijk en daardoor nog moeilijker dan goede Windows applicaties programmeren. Kunt u Windows programma's maken en wilt u spellen proberen te maken, dan zult u echt in een andere programmeerwereld moeten overstappen. De techniek in de gamewereld is het leren begrijpen van sprites, vertices, views en matrices en nog veel meer. Woorden die we in de normale programmeerwereld niet tegenkomen.

Er bestaan heel veel soorten game editors om het programmeren van spellen makkelijker te maken, maar dan alsnog moeten de nieuwe woorden goed bestudeerd worden, anders is het gewoon niet te begrijpen hoe zo een programma werkt. Toch is er een game editor, speciaal geschikt voor beginners, en er kunnen spellen mee worden gemaakt zonder programmacode: Game Maker. Met Game Maker is het niet moeilijk om te begrijpen hoe sprites en objecten samenwerken en hoe de wereld en de view

met een room ingedeeld kan worden. De objecten van de sprites kunnen met de muis op de room geplaatst worden. Game Maker maakt er automatisch objectinstanties van.
Maar Game Maker mist helaas toch wat functionaliteit, zoals muisbesturing en de 3D wereld. Game Maker heeft wel muispositie variabelen waarmee de positie gelezen kan worden. Het programma is een drag en drop gamemaker om zonder programmacode spellen te kunnen maken, maar omdat er geen muisbesturingsknoppen aanwezig zijn, is men genoodzaakt toch scripts te moeten maken.

Het XNA Game Studio Framework is een gratis solution sjabloon. Net als .NET Framework bestaat XNA ook in meerdere versies. De nieuwe versie, Framework 4.0, heeft helaas geen sjabloon voor Visual Basic 2010, alleen voor Visual C#. Toch kan, door de XNA references in het project te kiezen, een soortgelijk sjabloon zelf gemaakt worden.

De volgende keer ga ik dieper op in met XNA, want ook al is het Visual C#; het is leuk om te weten hoe het sjabloon in elkaar zit en hoe we een game wereld aan kunnen sturen. Er kan ook in Google gezocht worden naar een sjabloon voor Visual Basic, want die zijn er wel, bijvoorbeeld voor Visual Basic 2005 en 2008 is er een XNA Game Studio Framework 2.0 sjabloon. Ook laat ik zien hoe we handmatig de versie 4 sjabloon in Visual Basic 2010 kunnen maken.

Cursussen

Liberty BASIC:
Cursus en naslagwerk, beide met voorbeelden op CD-ROM, € 6,00 voor leden. Niet leden € 10,00.

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.

Computercursus voor iedereen: tekstverwerking met Office en eventueel met VBA, Internet en programmeertalen, waaronder ook Basic, die u zou willen leren.
Elke dinsdag, woendag en vrijdag in buurthuis Bronveld in Barneveld van 19:00 uur tot 21:00 uur op de dinsdag en van 9:00 uur tot 11:00 uur op de woensdag en vrijdag. Kosten € 5,00 per week.
Meer informatie? Kijk op 'http://www.i-t-s.nl/rdkcomputerservice/index.php' of neem contact op met mij.

Computerworkshop voor iedereen; heeft u vragen over tekstverwerking of BASIC, dan kunt u elke 2de en 4de week per maand terecht in hetzelfde buurthuis Bronveld in Barneveld van 19:15 uur tot 21:15 uur. Kosten € 5,00.
Meer informatie? Kijk op 'http://www.buurthuisbronveld.nl' of neem contact op met mij. Voor overige informatie: http://www.tronicasoftware.nl

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.

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	Jeroen v. Hezik	ma. t/m zo.	19-21	(0346) 214131	j.a.van.hezik@kader.hcc.nl
Visual Basic .NET	Marco Kurvers	vr. t/m zo.	19-22	06 30896598	m.a.kurvers@live.nl
Basic algemeen, zoals VBA Office	Marco Kurvers	vr. t/m zo.	19-22	06 30896598	m.a.kurvers@live.nl
Web Design, met XHTML en CSS