Basic Bulletin

22ste jaargang maart 2015 Nummer 1

HCC BASIC interessegroep

Inhoud

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

Met PowerBASIC kom ik terug over variabelen, arrays en mappen. Dat laatste is heel uniek in BASIC, want die mogelijkheid geeft veel flexibiliteit met het werken in gegevensstructuren. Geen ander BASIC dialect heeft die mogelijkheid, hoewel dynamische arrays ook flexibiliteit geven. Er is ook een mogelijkheid een eigen databasetabel in elkaar te zetten en elke rij te vullen doormiddel van een array element.

In Excel weten we dat we vensters kunnen toevoegen aan de Excel werkbladen via macro's. We kunnen echter ook een werkblad ombouwen als een werkvenster. Misschien eens wat anders dan alleen maar steeds die kale draaitabellen.

Marco Kurvers

BBC BASIC for Windows – De library's (4).

Eigenschappenvensters en wizards.

De **WINLIB4** library bevat een aantal procedures en functies voor het maken en besturen van eigenschappenvensters en wizards. De library moet worden geladen vanuit uw programma met het volgende commando:

```
INSTALL @lib$+"WINLIB4"
```

De procedures en functies zijn:

- FN_newpropsheet
- PROC_showpropsheet
- PROC_closepropsheet

De WINLIB4U bibliotheek bevat een identieke set van functies, behalve dat die een tekenreeks ontvangen in Unicode (UTF-8)-indeling, waardoor niet-ANSI (bijvoorbeeld de buitenlandse taal) tekens kunnen worden opgenomen.

Eigenschappenvensters en **wizards** zijn zeer gelijk; beide bestaan ze uit meerdere dialoogvensterpagina's binnen een enkele raam (alleen één pagina kan per keer worden weergegeven). Ze verschillen vooral op de manier waarop die de afzonderlijke pagina's zijn geselecteerd: in een **eigenschappenvenster** kunnen de pagina's geselecteerd worden in elke volgorde (elk heeft een *tab* die aangeklikt kan worden door de gebruiker), terwijl dat in een **wizard** op volgorde geselecteerd moet worden met gebruik van de **Volgende** en **Vorige** knoppen.

Omdat de eigenschappenvensters en wizards als meerdere dialoogvensters zijn, is dit precies hoe ze worden gemaakt in *BBC BASIC for Windows*. Elke pagina moet worden gemaakt met gebruik van de FN_newdialog functie in de WINLIB2 bibiliotheek. Echter, in tegenstelling tot conventionele dialoogvensters, moet u niet **OK**, **Toepassen** of **Annuleren** knoppen in de afzonderlijke pagina's opnemen.

In plaats van het weergeven van de afzonderlijke dialoogvensters met **PROC_showdialog** zijn ze samen gegroepeerd als een eigenschappenvenster of een wizard met behulp van **FN_newpropsheet** en weergegeven met behulp van **PROC_showpropsheet**.

FN_newpropsheet(caption$,npages%,initial%,style%,dlg%())

Deze functie groepeert meerdere dialoogvensters om een enkel eigenschappenvenster of wizard om te vormen. Vijf parameters dienen te worden verstrekt: een bijschrift voor de titelbalk, het aantal dialoogvensterpagina's, de pagina die in eerste instantie moet worden weergegeven (waarbij de eerste pagina met nul is genummerd), de window style (meestal **0** voor een eigenschappenvenster of **&20** voor een wizard) en een serie dialoogvenster *pointers* (elk heeft de verwezen resultaatwaarde van FN_newdialog).

Het volgende programma fragment maakt een eigenschappenvenster met drie dialoogvensterpagina's:

```
pages% = 3
DIM page%(pages%-1)
page%(0) = FN_newdialog("First page", 32, 32, 288, 128, 8, 650)
REM create the contents of the first page here
page%(1) = FN_newdialog("Second page", 32, 32, 288, 128, 8, 1100)
REM create the contents of the second page here
```

```
page%(2) = FN_newdialog("Third page", 32, 32, 288, 128, 8, 1100)
REM create the contents of the third page here
psh% = FN_newpropsheet("Property sheet",pages%,0,0,page%())
```

Maak, na elke aanroep van FN_newdialog, de inhoud van de relevante pagina aan met gebruik van de procedures, dat aanwezig is in WINLIB2, bijvoorbeeld PROC_static, PROC_editbox, enz. Als een wizard in plaats van een eigenschappenvenster wordt gevraagd, zou de laatste regel moeten zijn:

```
psh% = FN_newpropsheet("Property sheet",pages%,0,&20,page%())
```

PROC_showpropsheet(psh%,hdlg%())

Deze procedure geeft het eigenschappenvenster of wizard weer. Twee parameters dienen te worden verstrekt: de pointer naar het eigenschappenvenster geretourneerd door FN_newpropsheet en een lege array voor de dialoogvenster handles. Die array moet worden geDIMensioneerd om minstens zoveel elementen als er pagina's in het eigenschappenvenster of wizard zijn, en als resultaat van PROC_showpropsheet zullen de handles de inhoud bevatten van elke dialoogvensterpagina. Deze zullen nodig zijn om te initialiseren en de inhoud van de afzonderlijke pagina's te ondervragen.

Het volgende programma fragment geeft het eigenschappenvenster of wizard weer, gemaakt als hierboven:

```
DIM hdlg%(pages%-1)
PROC_showpropsheet(psh%,hdlg%())
```

De inhoud van een eigenschappenvenster initialiseren

Net als met een dialoogvenster, kan de oorspronkelijke inhoud van de eigenschappenvensterpagina's worden gedefinieerd wanneer de items binnen hen worden gemaakt. Bijvoorbeeld, de inhoud van een invoervak kan worden opgegeven in de PROC_editbox procedure aanroep. Maar u kunt ook de inhoud wijzigen op andere momenten, en in het geval van keuzelijsten en keuzelijsten met invoervakken, die meerdere items bevatten, moet u andere methoden gebruiken om de inhoud te initialiseren.

De methoden die dit doen zijn identiek aan die worden opgesomd in de **Initialisatie van een dialoogvenster** (zie de Bulletin over Dialoogvensters), behalve dat wanneer het dialoogvenster handle nodig is, moet u een element van de **hdlg%()** array opgeven, die geretourneerd is van PROC_showpropsheet, in plaats van de waarde die verwezen wordt naar FN_newdialog. Bijvoorbeeld, om de geassocieerde tekst met een item in de eerste pagina van een eigenschappenvenster of wizard te wijzigen:

```
SYS "SetDlgItemText", hdlg%(0), id%, text$
```

Normaal gesproken moet u ervoor zorgen dat het item **id%** uniek is, in plaats van in twee of meer verschillende pagina's te gebruiken. Als het opgegeven item niet in de pagina overeenkomt met de opgegeven dialoogvenster handle, mislukt de aanroep.

Het aanklikken van een eigenschappenvenster knop bepalen

U moet kunnen bepalen wanneer de gebruiker geklikt heeft op de **OK** of **Voltooien** knop, zodat de inhoud van het eigenschappenvenster of wizard gelezen kan worden en het venster van het scherm verwijderd kan worden. Wanneer een item in een eigenschappenvenster aangeklikt is, zal een bericht worden verzonden (hetzelfde als van een menu of een werkbalk), die gedetecteerd kan worden met ON SYS.

Het volgende codefragment wordt gebruikt om te wachten voor het klikken op de **OK**, **Voltooien** of **Annuleren** knop, om daarna bepaalde actie te ondernemen.

```
Click% = 0
ON SYS Click% = @wparam%  : RETURN
REPEAT WAIT 1
  click% = 0
  SWAP click%,Click%
UNTIL click% = 1 OR click% = 2
ON SYS OFF
IF click% = 1 THEN
  PRINT "OK or Finish pressed"
  REM process contents of property sheet here
ELSE
  PRINT "Cancel pressed"
ENDIF
PROC_closepropsheet(psh%)
```

De wijzigingen van een eigenschappenvenster pagina volgen

Onder bepaalde omstandigheden moet u wellicht te weten komen wanneer de gebruiker het eigenschappenvenster of wizard pagina heeft gewijzigd. Dit geldt vooral voor een wizard, want het is de verantwoordelijkheid van het gebruikersprogramma die de **Volgende** knop moet vervangen in een **Voltooien** knop, wanneer de laatste pagina van de wizard is weergegeven (en net zo als de **Vorige** knop die afgeremd wordt als de eerste pagina weergegeven wordt).

U kunt dit doen door de huidige pagina *handle* te volgen, die door een API aanroep bepaald kan worden. Bijvoorbeeld, de loop voor het volgen om te controleren welke knoppen zijn aangeklikt kan worden veranderd, ook voor pagina wijzigingen werkt dit als volgt:

```
PSM_SETWIZBUTTONS = 1136
PSM_GETCURRENTPAGEHWND = 1142
Click% = 0
ON SYS Click% = @wparam% : RETURN
oldhdlg% = 0
REPEAT
WAIT 1
  click% = 0
  SWAP click%,Click%
  SYS "SendMessage", !psh%, PSM_GETCURRENTPAGEHWND, 0, 0 TO hdlg%
  IF hdlg%<>oldhdlg% THEN
    oldhdlg% = hdlg%
    CASE hdlg% OF
      WHEN hdlg%(0):
        SYS "SendMessage", !psh%, PSM_SETWIZBUTTONS, 0, 2 : REM Next only
      WHEN hdlg%(1):
        SYS "SendMessage", !psh%, PSM_SETWIZBUTTONS, 0, 3 : REM Back and Next
      WHEN hdlg%(2):
        SYS "SendMessage", !psh%, PSM_SETWIZBUTTONS, 0, 5 : REM Back and Finish
    ENDCASE
  ENDIF
  CASE click% OF
    WHEN 1: PRINT "Finish pressed"
    REM Process contents of wizard here
    WHEN 2: PRINT "Cancel pressed"
  ENDCASE
UNTIL hdlg% = 0
PROC_closepropsheet(psh%)
```

De inhoud van een eigenschappenvenster lezen

Aangezien het hele doel van een eigenschappenvenster of wizard om gebruikersinvoer te ontvangen, is het essentieel dat de huidige inhoud kan worden bepaald, voornamelijk wanneer op de knop OK of voltooien wordt geklikt.

De methoden die dit doen zijn identiek aan degene die opgesomd worden in **De inhoud lezen van een dialoogvenster**, behalve dat wanneer het dialoogvenster handle nodig is, moet u een element opgeven van de geretourneerde **hdlg%()** array van PROC_showpropsheet in plaats van de verwezen waarde van FN_newdialog. Bijvoorbeeld, om de geassocieerde tekst met een item in een eigenschappenvenster of wizard te lezen:

```
DEF FNgetproptext(hdlg%(), page%, id%)
LOCAL text%
DIM text% LOCAL 255
SYS "GetDlgItemText", hdlg%(page%), id%, text%, 255
= $$text%
```

PROC_closepropsheet(psh%)

Wanneer de inhoud van het eigenschappenvenster of wizard is verwerkt, moet het worden verwijderd van het scherm:

```
PROC_closepropsheet(psh%)
```

Het eigenschappenvenster sjablonen blijven in het geheugen, zodat u het op elk gewenst moment kunt weergeven door PROC_showpropsheet aan te roepen.

De wizard of eigenschappenvenster moet ook worden verwijderd wanneer uw programma terugkeert naar de directe modus (bijvoorbeeld als er een fout optreedt of het einde statement wordt uitgevoerd) of wanneer uw programma venster door de gebruiker wordt gesloten. Dit kan worden bereikt door het uitvoeren van de volgende instructies onmiddellijk na de aanroep van PROC_showpropsheet:

```
ON CLOSE PROC_closepropsheet(psh%):QUIT
ON ERROR PROC_closepropsheet(psh%):PRINT'REPORT$:END
```

Omdat het eigenschappenvenster ruimte gebruikt op de *heap*, is het esentieel dat u het eerst verwijderd voordat u een CLEAR, CHAIN of RUN statement uitvoert. Een zeer waarschijnlijke crash van *BBC BASIC for Windows* is beter niet te doen.

Volgende Bulletin: Balken en knoppen

De Windows API in actie.

Verdragen:
In deze sectie worden de functies hoofdzakelijk in een referentielijst opgesomd met tussendoor verschillende routines ingevoegd om sommige API aanroepen die er worden getoond te illustreren.

In de volgende voorbeelden moet u er vanuit gaan dat het venster geopend is en het venster handle “h” is, zoals teruggegeven door de HWND functie:

```
h = hwnd(#main)
```

Ga er ook vanuit dat de handles van de controls met een “h” beginnen en beschrijvend zijn van de control. Bijvoorbeeld, een teksteditor gemaakt met het texteditor commando van Liberty BASIC zal een handle teruggeven na een #main.text met de HWND functie van “hText” en de handle van een graphicbox gemaakt als #main.gbox zal een handle hebben van “hGraphicBox”:

```
hText          =    hwnd(#main.text)
hGraphicBox    =    hwnd(#main.gbox)
```

Om dingen zo goed mogelijk te maken zonder telkens geduplicerde uitleg, geef dan variabelen en parameters beschrijfbare namen, zoals dit:

```
xOrg      =    originele X/locatie
yOrg      =    originele Y/locatie
xExtent   =    X einde punt
yExtent   =    Y einde punt
width     =    breedte
height    =    hoogte
```

Overeenkomsten van controls en vensters
Controls, zoals statische teksten en knoppen, zijn eigenlijk zoonvensters van het venster die hun vasthoud. Functies, zoals MoveWindow, kan werken in een venster of in een control, afhankelijk van de handle die doorgegeven is in de functie. Als de functie de handle van het venster retourneert, werkt het in het venster. Is de handle gegeven vanuit een control, dan werkt het in die control.

Speciale overwegingen voor tekst en grafische vensters
Tekstvensters zijn eigenlijk teksteditors binnen normale vensters. Grafische vensters zijn eigenlijk grafische balken binnen normale vensters. Het krijgen van een venster handle met de HWND() functie retourneert in principe de handle van de teksteditor of grafische balk, niet van het venster zelf. Als de API functie gebruikmaakt van het venster handle die bedoeld is om de tekst of grafische eigenschappen te manipuleren, dan is de HWND(#window) de juiste handle om te krijgen. Voor API functies, waarvan de bedoeling is om het venster zelf te manipuleren, is het nodig om het **ouder** venster handle van het venster te krijgen. Gebruik de GetParent API functie (later meer daarover uitgelegd) en gebruik de teruggekregen handle voor functies als MoveWindow of SetWindowTextA.

GetDC
Deze functie retourneert een handle van een device context voor het venster waarvan de handle is “h”. De DC, of device context, is nodig voor sommige andere API aanroepen. De DEVICE CONTEXT komt later ter sprake met het onderdeel over het maken van API aanroepen naar GDI.DLL. Alleen Windows accepteert een programma die vijf device contexten in een keer heeft. Zorg ervoor dat ReleaseDC elke hDC teruggeeft met deze aanroep. Gebruik een device context geretourneerd van deze API aanroep voor de functies, hier getoond, die een “hDC” parameter hebben.

```
Calldll #user32, "GetDC", h as long, hDC as long
```

Liberty BASIC API Reference.

BringWindowToTop
De BringWindowToTop functie brengt het bepaalde venster tot de bovenkant van een stapel overlappende vensters.

```
calldll #user32, "BringWindowToTop",_
h as ulong,_          'venster handle
result as boolean     'geen resultaatwaarde
```

CloseWindow
De CloseWindow functie minimaliseert een venster tot de taakbalk, maar vernietigd het venster niet.

```
calldll #user32, "CloseWindow",_
h as ulong,_          'venster handle
result as boolean
```

ClientToScreen
Deze functie zet de vensterclïënt coördinaten om in scherm coördinaten. De functie zal de linker bovenhoek van het vensterclïënt gebied in scherm coördinaten resulteren. Is de linker bovenhoek van een venster 10 pixels van de linkerzijde en 50 pixels van de top, dan zal clientXorigin 10 zijn en clientYorigin zal 50 zijn in het volgende voorbeeld.

```
struct point,_        'struct om de coördinaten te krijgen
x as long,_           'x coördinaat
y as long             'y coördinaat

point.x.struct = 0 : point.y.struct = 0

calldll #user32, "ClientToScreen",_
h as long,_           'venster handle
point as struct,_     'structure
r as long

clientXorigin = point.x.struct
clientYorigin = point.y.struct
```

CopyRect
De CopyRect functie kopieert de coördinaten van de ene rechthoek naar de andere.

```
struct rect1, left as long, top as long, right as long, bottom as long
struct rect2, left as long, top as long, right as long, bottom as long
rect1.left.struct = 10 : rect1.top.struct = 50
rect1.right.struct = 240 : rect1.bottom.struct = 200

calldll #user32, "CopyRect",_
     rect2 as struct,_    'bestemming rect struct
     rect1 as struct,_    'bron rect struct
     re as boolean        'niet-nul zijnde=succes

print "Rect 2 members zijn nu: ";
print rect2.left.struct
print rect2.top.struct
print rect2.right.struct
print rect2.bottom.struct
```

DrawAnimatedRects
De DrawAnimatedRects functie tekent een draadframe (wire-frame) rechthoek en bepaald of het om een opening vanuit een icoon of om het minimaliseren of maximaliseren van een venster gaat. De eerste rechthoek struct moet de coördinaten inhoud hebben van het geminimaliseerde venster. De tweede rechthoek struct moet de coördinaten inhoud hebben van het venster uit de herstelde positie. De functie animeert een reeks van rechthoeken zodat het lijkt alsof dat het venster is die naar de laatste locatie verplaatst. Het venster zichzelf moet dan verplaatst worden met een MoveWindow aanroep.

```
struct rect1, left as long, top as long, right as long, bottom as long
struct rect2, left as long, top as long, right as long, bottom as long

     nomainwin
     'open venster in uiterst kleine grootte in rechter benedenhoek van
     'het venster:
     UpperLeftX=DisplayWidth : UpperLeftY=DisplayHeight
     WindowWidth=1:WindowHeight=1
     open "Een Venster" for window as #1
     #1 "trapclose [quit]"
     h = hwnd(#1)
     flags=_IDANI_OPEN Or _IDANI_CAPTION

'vul rechthoek structs zo snel mogelijk:
calldll #user32, "SetRect",rect1 as struct,_
     DisplayWidth as long, DisplayHeight as long,_
     DisplayWidth as long, DisplayHeight as long,_
     ret as boolean

calldll #user32, "SetRect",rect2 as struct,_
     0 as long, 0 as long, 200 as long, 200 as long, ret as boolean

calldll #user32, "DrawAnimatedRects",_
     h as ulong,_         'venster handle
     flags as ulong,_     'vlaggen voor animatietype
     rect1 as struct,_    'geminimaliseerde positie rechthoek
     rect2 as struct,_    'herstelde positie rechthoek
     ret as boolean

'na de animatie, verplaats het actuele venster tot waar de animatie
'is gestopt
calldll #user32, "MoveWindow",_
     h as ulong,_         'venster handle
     0 as long,_          'links x
     0 as long,_          'top y
     200 as long,_        'breedte
     200 as long,_        'hoogte
     1 as boolean, ret as boolean
wait
[quit]    close #1:end
```

DrawIcon

Deze functie tekent de icoon van welke handle, hIcon, is teruggegeven met een aanroep van ExtractIcon of LoadImageA.

```
calldll #user32, "DrawIcon",_
     hDC as ulong,_       'DC van het venster waar de icoon zal staan
     xOrg as long,_       'x locatie op het venster om de icoon te tekenen
     yOrg as long,_       'y locatie op het venster om de icoon te tekenen
     hIcon as ulong,_     'handle van icoon die getekend moet worden
     Result as boolean    'wanneer geslaagd, Result is niet-nul
```

EnableWindow

Deze functie schakelt de muis of toetsenbord invoer aan of uit van het gegeven venster of control. Wanneer de invoer is uitgeschakeld, negeert het venster de invoer zoals muisklikken en toetsenbord drukken en toont het een grijze weergave. Wanneer de invoer is ingeschakeld, verwerkt het venster alle invoer. Liberty BASIC heeft nu native “enable/disable” mogelijkheden voor vensters en controls.

```
calldll #user32, "EnableWindow",_
     h as ulong,_         'handle van venster of control
     1 as boolean,_       '1 = ingeschakeld   0 = uitgeschakeld
     result as boolean    'resulteert niet-nul als het venster daarvoor
                          'uitgeschakeld was
```

ExitWindows
De ExitWindows functie kan Windows herstarten zodra de _EWX_REBOOT vlag gebruikt wordt, of het kan het systeem afsluiten tot het moment als het veilig is de power uit te schakelen zodra de _EWX_SHUTDOWN vlag gebruikt wordt. Gebruik met VOORZICHTIGHEID! Bij gebruik van dit commando kan leiden tot het verliezen van niet opgeslagen informatie als de gebruiker andere applicaties nog open heeft staan. Om zeker te weten of deze operatie uitgevoerd kan worden, is het beter om eerst met de gebruiker te overleggen of het kan. Windows XP werkt anders. Het is mogelijk om deze functie te gebruiken om in XP uit te loggen, maar niet om de computer af te sluiten.

```
'mogelijke afsluit vlaggen
'_EWX_LOGOFF        logt de gebruiker uit, maar laat de computer aan
'_EWX_POWEROFF      schakelt de power uit
'_EWX_REBOOT        verlaat windows, dan herstarten (heet ook: warme start)
'_EWX_SHUTDOWN      verlaat windows, laat de power aan

flag = _EWX_LOGOFF

calldll #user32, "ExitWindowsEx",_
     flag as ulong,_      'afsluittype
     0 as ulong,_         'moet nul zijn
     re as boolean        'succes=niet-nul

'Beëindig het programma tijdens het afsluiten van Windows
end
```

GetActiveWindow
Geeft de handle van het actieve venster. Als het uitgevoerd is vanuit de start van het programma, en het programma heeft niet het commando nomainwin gebruikt, dan retourneert het de handle van het hoofdvenster. Een programma kan dan het hoofdvenster manipuleren met API aanroepingen.

```
calldll #user32, "GetActiveWindow",_
     h as ulong     'retourneert de handle van het actieve venster
```

Dit voorbeeld laat zien hoe Liberty BASIC een hoofdvenster in volle grootte weergeeft:

```
calldll #user32, "GetActiveWindow",_
     h as ulong     'h is de handle van het hoofdvenster

calldll #user32, "MoveWindow",_
     h as ulong,_             'venster handle
     0 as long,_              'linkerbovenhoek x
     0 as long,_              'linkerbovenhoek y
     DisplayWidth as long,_   'breedte
     DisplayHeight as long,_  'hoogte
```

```
     1 as boolean,_           'vernieuw flag = true
     result as boolean
```

GetAsyncKeyState
Deze functie bepaalt wanneer een toets omhoog of omlaag is in de tijd dat de functie werd aangeroepen. Een negatieve waarde (kleiner dan 0) betekent dat de gespecificeerde toets omlaag is. Controleer voor VK waarden elke aanwezige lijst van Virtuele Toets Codes (Virtual Key Codes). Begrijp wel dat de VK constante moet beginnen met een underscore teken!

```
calldll #user32, "GetAsyncKeyState",_
     _VK_LEFT as long,_  'virtuele toets om te controleren
     result as long      'negatieve waarde betekent de toets is omlaag
```

GetClassName
Alle vensters en controls zijn gemaakt als deel van een KLASSE. Met deze functie krijgt u de klasse naam terug. Als de doorgegeven handle binnen de functie van een knop is, dan is bijvoorbeeld de buffer class$ gevuld bij de functie met als inhoud de klasse “Button”.

```
class$=space$(255)+chr$(0) : length = len(class$)

calldll #user32, "GetClassNameA",_
     hButton as ulong,_       'handle van venster of control
     class$ as ptr,_          'buffer om de naam te krijgen
     length as long,_         'lengte van de buffer
     returnLength as long     'lengte van de geresulteerde naam

class$=left$(class$, returnLength)
'lees linker tekens in buffer tot de opgegeven lengte
```

GetClientRect
Deze functie vult een STRUCT met de punten die de werkruimte definiëren, of een cliëntgebied van een venster. Dit is de ruimte inhoudelijk binnen de framegrootte, schuifbalken, menubalk of titelbalk van een venster. De leftX en upperY zijn altijd 0, 0. De rightX member bevat de waarde voor de breedte van het cliëntgebied, nadat het gevuld is bij de GetClientRect functie. De lowerY member bevat de hoogte.

```
h=hwnd(#main)
struct Rect,_             'naam van de struct is Rect
     leftX as long,_      'linksboven x
     upperY as long,_     'linksboven y
     rightX as long,_     'rechtsonder x
     lowerY as long       'rechtsonder y

calldll #user32, "GetClientRect",_
     h as ulong,_         'venster handle
     Rect as struct,_     'naam van de struct
     r as boolean         'niet-nul zijnde=succes

ClientWidth  = Rect.rightX.struct
ClientHeight = Rect.lowerY.struct
```

GetCursorPos
De GetCursorPos functie geeft de schermcoördinaten terug van de huidige muiscursor positie. Roep de functie ScreenToClient aan om deze coördinaten om te zetten in venstercoördinaten.

```
struct point,_ 'struct om de coördinaten te krijgen
     x as long, y as long

calldll #user32, "GetCursorPos",_
     point as struct,_   'naam van de struct
     r as boolean        'niet-nul zijnde=succes

cursorX = point.x.struct      'muis X positie
cursorY = point.y.struct      'muis Y positie
```

GetDesktopWindow
Deze functie retourneert de handle van het venster die de desktop beslaat.

```
calldll #user32, "GetDesktopWindow", hDesktop as ulong
```

GetNextWindow
Gebruik een bestaand venster handle om de handle van het volgende venster of van het vorige venster te krijgen, in de huidige geopende lijst.

```
Flag = _GW_HWNDNEXT      'Retourneert een handle van het volgende venster.
Flag = _GW_HWNDPREV      'Retourneert een handle van het vorige venster.

calldll #user32, "GetNextWindow",_
     handle as ulong,_        'bestaand venster handle
     Flag as long,_           'volgende of vorige flag
     nextWinHandle as ulong   'retourneert venster handle
```

GetParent
Deze functie haalt de handle van het oudervenster voor het venster van wie de handle “h” is. Alle controls, zoals knoppen en lijstvakken, zijn zoonvensters.

```
calldll #user32, "GetParent",_
     hButton as ulong,_       'geeft de ouder van dit venster
     hParent as ulong         retourneert de handle van het oudervenster
```

GetSysColor
Deze functie krijgt de systeemkleur (een long integer kleurwaarde) voor het Windows gebied door de aangegeven “nIndex”. Het retourneert de kleurwaarde van dat gebied. Gebruik deze functie om de systeemkleuren te krijgen voor elk gebied waarvan de kleuren worden gewijzigd door een programma, zodat ze kunnen worden hersteld naar de gebruiker standaardwaarden aan het einde van het programma. Zie SetSysColor voor meer over de nIndex waarden.

```
calldll #user32, "GetSysColor",_
     nIndex as long,_    'index van gebied om te zien
     sysColor as ulong   'kleurwaarde van gebied
```

GetWindowDC
Deze functie haalt de device context van het hele venster, vergelijkbaar met de GetDC aanroep, die de DC van het cliëntgebied van het venster haalt. Wanneer het niet langer nodig is, moet de DC worden vrijgegeven met de ReleaseDC aanroep, aangezien alleen 5 DC’s tegelijk open mogen zijn. De variabele h is de handle van het venster, verkregen met LB’s hwnd() functie.

```
calldll #user32, "GetWindowDC", h as ulong, hDC as ulong
```

GetWindowLong
Gebruik deze functie om de long waarde te halen uit de opgegeven flag argument. Dit voorbeeld haalt de instantie handle die nodig is voor de ExtractIcon functie, en voor andere functies.

```
calldll #user32, "GetWindowLongA",_
h as ulong,_             'handle van venster die als inhoud controls heeft
_GWL_HINSTANCE as long,_ 'flag voor gewenste word waarde=instantie handle
hInstance as ulong       'instantie handle van venster is geretourneerd


'mogelijke flaggen:

_GWL_EXSTYLE        Haalt de externe vensterstijlen.
_GWL_STYLE          Haalt de vensterstijlen.
_GWL_WNDPROC        Haalt het adres van het venster procedure of een handle
die het adres van het venster procedure vertegenwoordigt. U moet de Call-
WindowProc functie gebruiken om het venster procedure aan te roepen.
_GWL_HINSTANCE      Haalt de handle van de applicatie instantie.
_GWL_HWNDPARENT     Haalt de handle van het oudervenster, als er één is.
_GWL_ID             Haalt de identifier van het venster.
_GWL_USERDATA       Haalt de 32-bit waarde geassocieerd met het venster.
Elk venster heeft een corresponderende 32-bit waarde bestemd voor gebruik
bij de applicatie die het venster heeft gemaakt.
```

GetWindowRect
Deze functie haalt de dimensies van de rechthoek van een gegeven venster. De dimensies zijn gegeven in schermcoördinaten, relatief aan de linkerbovenhoek van de schermweergave, inclusief de titelbalk, border en schuifbalken, indien aanwezig. Na de aanroep moeten de dimensies gehaald worden vanuit de STRUCT.

```
struct winRect,_
     orgX as long, orgY as long, cornerX as long, cornerY as long
open "test mij" for window as #main
h = hwnd(#main)
calldll #user32, "GetWindowRect",_
     h as ulong,_              'venster handle
     winRect as struct,_       'naam van de struct
     result as boolean

print "Linkerbovenhoek van 'test mij' is: ";
print winRect.orgX.struct; ", ";winRect.orgY.struct
print "Rechterbenedenhoek van 'test mij' is: ";
print winRect.cornerX.struct; ", "; winRect.cornerY.struct
close #main : end
```

GetWindowText
Deze functie kopieert tekst van de titelbalk (als het er een heeft) van een gegeven venster in een buffer. Als het gegeven venster een control is, zal de tekst in de control gekopieerd worden. De geretourneerde waarde geeft de lengte van de gekopieerde tekenreeks aan in bytes, niet met inbegrip van het afsluitende null-teken. Het is nul als het venster geen titelbalk heeft, de titelbalk is leeg of de handle parameter is ongeldig. De allereerste lengte van Title$ (de buffer om de tekst te kunnen halen) moet groter zijn dan de benodigde tekstlengte. Zie GetWindowTextLength hieronder.

```
Title$=space$(255)+Chr$(0) : length = Len(Title$)
calldll #user32, "GetWindowTextA",_
     h as ulong,_         'venster of control handle
     Title$ as ptr,_      'buffer om informatie te halen
     length as long,_     'lengte van de buffer
     result as long       'resulteert de lengte van de info tekenreeks

Title$ = Left$(Title$, result)
```

GetWindowTextLength
Deze functie haalt de lengte van de tekst in bijschrift voor een venster, of control, voor een tekstvak, knop, enz. De control of venster handle is de “h” waarde en de teruggave is het aantal tekens opgenomen in de tekst in de control of venster bijschrift. Gebruik deze functie om het aantal te krijgen om de juiste grootte in te stellen voor de buffer in GetWindowTextA.

```
calldll #user32, "GetWindowTextLengthA",_
     h as ulong,_         'venster of control handle
     numberChars as long 'retourneert de tekstlengte
```

IsIconic
Deze functie retourneert niet-nul als het venster waarvan de handle is “h” geminimaliseerd is, nul als het NIET geminimaliseerd is.

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

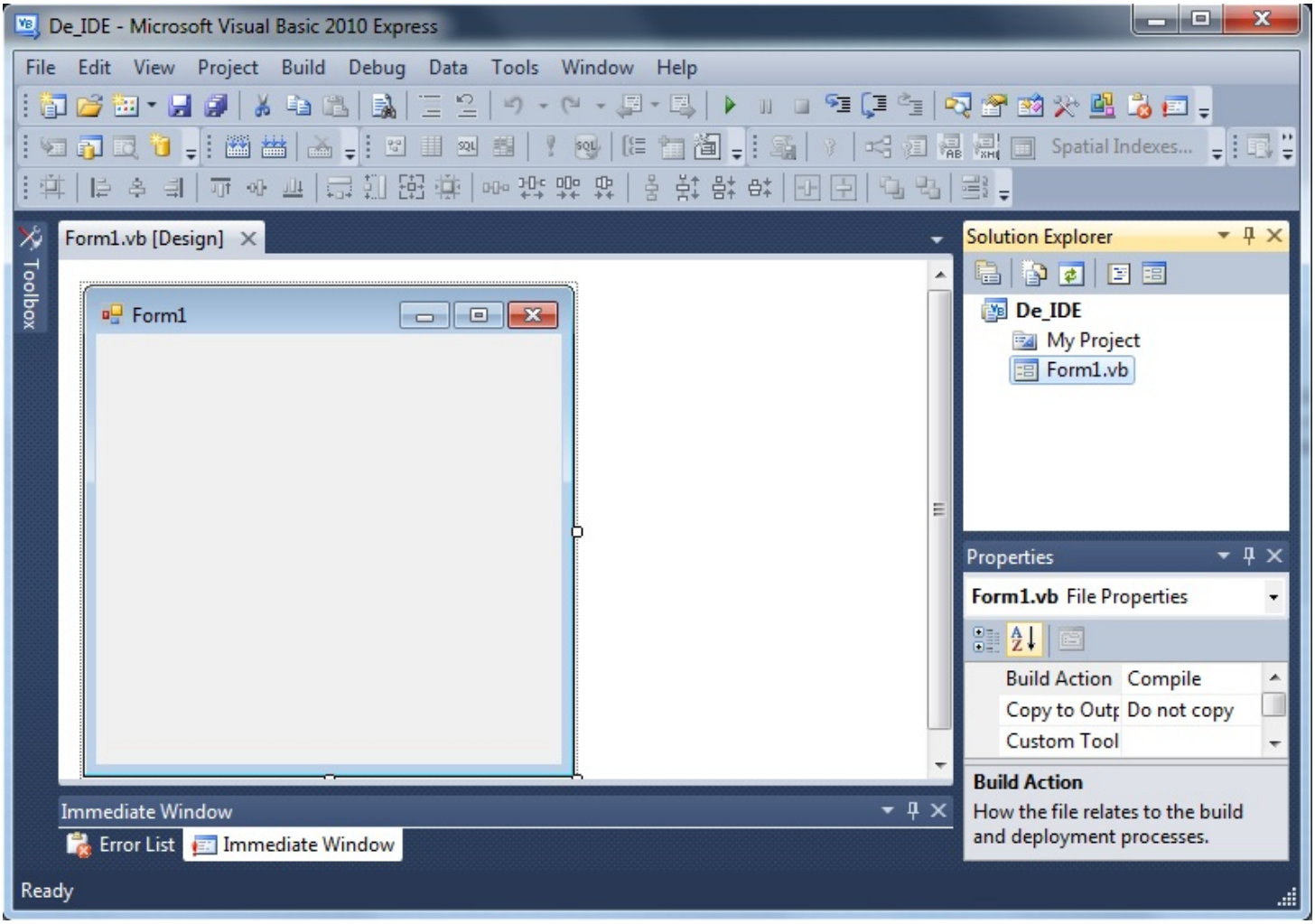
IsWindowEnabled
Deze functie retourneert niet-nul als een venster ingeschakeld is voor muis en toetsenbord invoer.

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

Wordt vervolgd

Visual Basic 2010 – De IDE.

Bekijk eens onderstaande afbeelding. Hier kunt u zien hoe het venster van Visual Basic 2010 eruit ziet.



Onder het menu ziet u drie rijen met werkbalken. Elke werkbalk begint met verticale puntjes. Door met de linker muisknop op de puntjes te klikken en te bewegen, kunt u de werkbalken op elke plaats zetten.
De eerste rij is één werkbalk en is ook de standaardbalk voor bestandsbeheer, configuratie en het uitvoeren van het project.
De tweede rij zijn de werkbalken die te maken hebben met databasebeheer. Hiermee kunt u een XML database opstellen en kunt u SQL query's maken.
De derde rij is de design werkbalk. Deze geeft extra hulpmiddelen voor het ontwerpen van de formulieren.

Het witte vlak is de kaart waar alle formulier- en codevensters in staan, ingedeeld in tabbladen. Het lege formulier Form1 is dus alleen te zien als het tabblad Form1.vb (Design) geopend wordt. In Visual Basic 6 bestond er nog geen tabblad indeling en werd alles kriskras op het scherm weergegeven.

Aan de linkerkant is de Toolbox aanwezig. Hier kunt u alle componenten in vinden die u kunt gebruiken op de formulieren. U hoeft niet meer voor extra componenten naar References te gaan, want de lijst is helemaal compleet en ingedeeld onder de componentcategorieën.

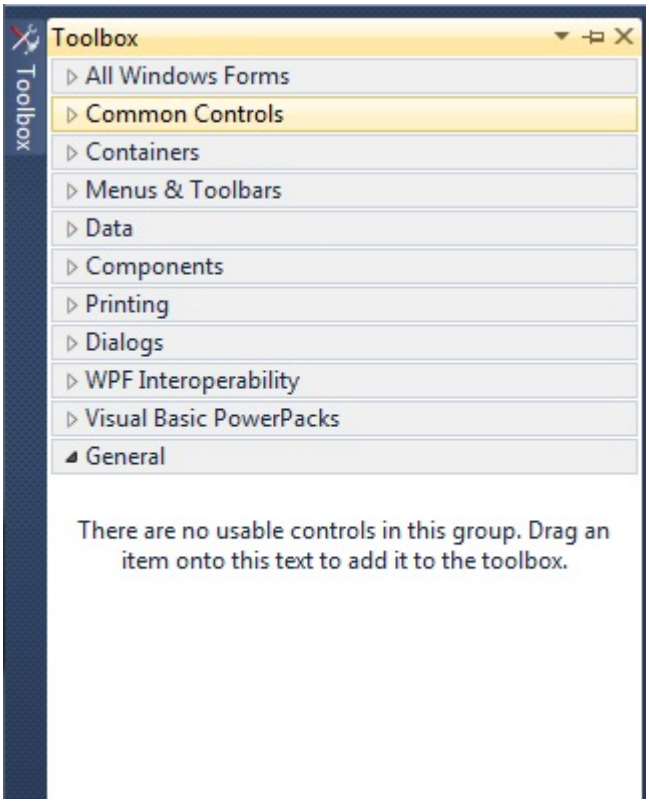
Aan de rechterkant is de Solution Explorer. Hier kunt u het project vinden. Doormiddel van een boomstructuur, wordt alles netjes ingedeeld. De Solution Exporer is echter niet alleen het project, er kunnen

meerdere projecten in worden opgenomen. Andere programmeertalen kunnen er in worden gebruikt en ook een XML project kan in de explorer worden opgenomen. Zonder problemen kan een hele solutie met van alles erin gecompileerd worden. Dat is het gemak van Visual Studio 2010.

Onder de Solution Explorer is de Properties. Hier wordt een lijst getoond met eigenschappen van de formulieren of componenten die op dat moment actief zijn. Zonder code kunnen dus al wat eigenschappen worden ingesteld. Op de knopjes kunt u ervoor kiezen of u het ingedeeld wilt hebben in categorieën of gewoon op alfabet. De knop rechts opent een eventlijst, dat is de lijst met gebeurtenissen voor het aansturen van de componenten. Soms verschijnt het 'Bliksempje' niet.

Onderaan is er een klein helpschermpje. Door op een eigenschap te klikken, krijgt u informatie over wat de eigenschap voor doel heeft.

Als we met de muis de Toolbox aanwijzen, wordt de componentenlijst geopend.
Hieronder ziet u de lijst met de categorie General actief, maar de Common Controls aangeklikt.



All Windows Forms
Hier wordt alles getoond wat normaal ingedeeld is in categorieën.

Common Controls
Dit zijn de meest gebruikte componenten, zoals de label, tekstvak en de knop.

Containers
Met deze containers kunt u uw formulieren netjes opmaken, zoals het indelen van tabbladen en groepen.

Menus & Toolbars
Maak hiermee de menu's en de werkbalken.

Data
Hier is alles te vinden over gegevensbeheer, zoals de dataset, navigator en de gegevenstabel die aan de dataset gekoppeld kan worden.

Components
Dit zijn de extra hulpmiddelen die samen met de andere componenten nodig zijn. Een voorbeeld is de ImageList. In de ImageList moeten alle afbeeldingen in worden geladen die nodig zijn voor de werkbalken. Met een werkbalk kunnen alleen de 'lege' knoppen worden toegevoegd, met eventueel de separator en nog meer werkbalkmogelijkheden.

Printing
Hier zijn alle afdrukmogelijkheden, zoals het afdrukvenster. Er kan ook direct worden afgedrukt, of eerst een afdrukvoorbeeld tonen.

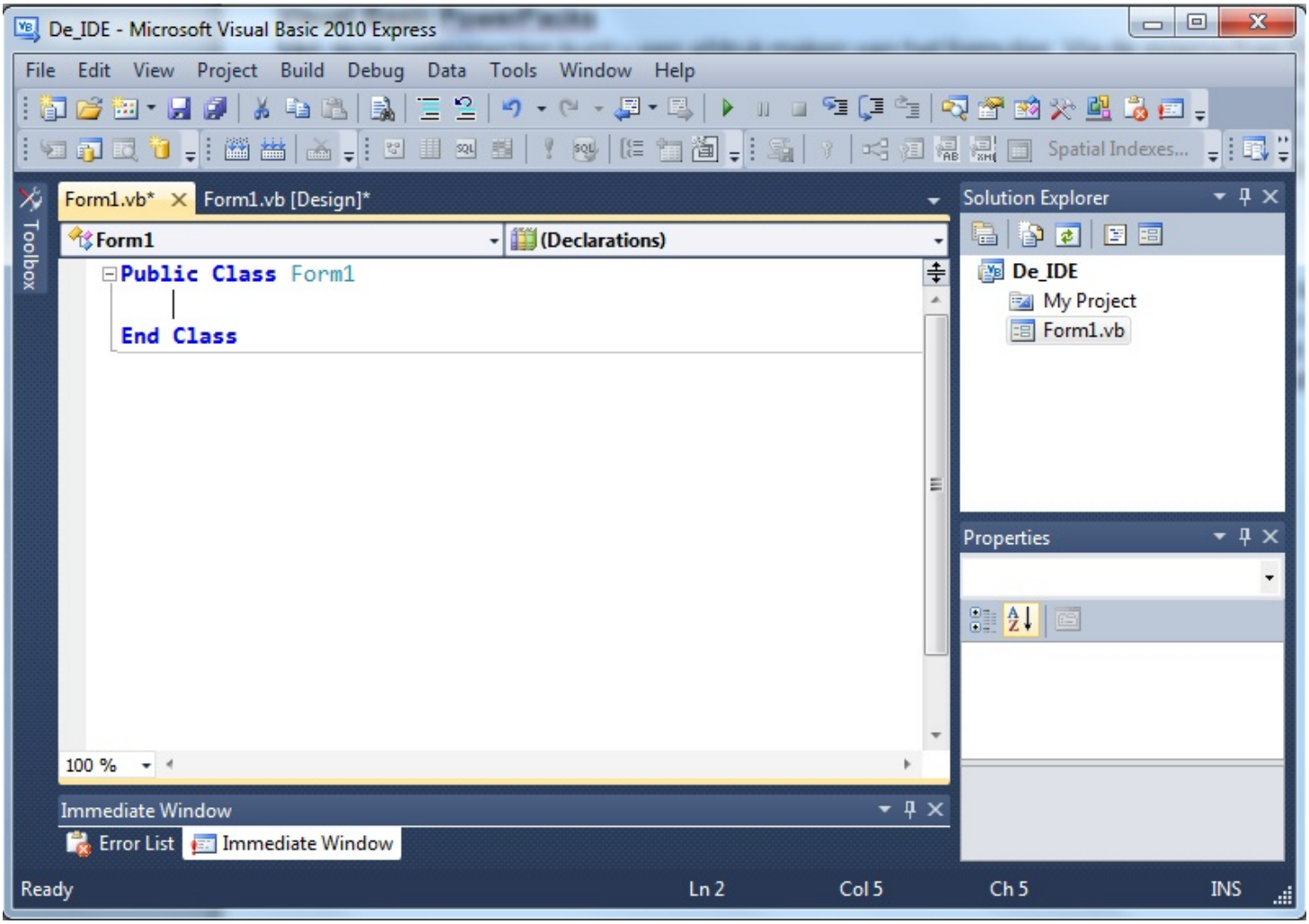
Dialogs
In Visual Basic 6 bestond de besturingscomponent CommonDialog. Hierin waren alle dialoogformulieren aanwezig die nu in deze lijst te kiezen zijn. Daarmee is de CommonDialog verleden tijd. Handig om in vervolg gewoon het FontDialog of het SaveFileDialog te kunnen kiezen.

WPF Interoperability
Dit heeft te maken met het besturen van de host. Het is namelijk mogelijk om met Visual Studio internetpagina's te maken. Echter is deze mogelijkheid sterk af te raden, omdat het code toevoegt dat nergens voor nodig is. De kans op scriptfouten en het vertragen op uw website kan namelijk toenemen. Het is beter om websites direct met HTML en CSS te schrijven of gebruik te maken van goede website editors.

Visual Basic PowerPacks
Met deze componenten kunt u een afdruk maken van het formulier. Via de eigenschap PrintAction kunt u het formulier afdrukken naar bestand. Verder zijn er ook tekencomponenten, zoals LineShape, OvalShape en RectangleShape. Het is overigens aan te raden gebruik te maken van het Graphics object om te tekenen, zie later dit jaar meer over grafisch tekenen in VB 2010.
De DataRepeater is een handige customizable gegevenslijst. Die gaan we ook later in dit jaar eens bekijken hoe dat werkt.

General
U hebt het waarschijnlijk al gezien: deze categorie is leeg. Toch is deze zeer handig om te gebruiken, want hiermee kunt u uw eigen projectonderdelen in plaatsen, zodat u niet elke keer naar de controls en componenten hoeft te zoeken. Sleep een onderdeel naar General en het staat er. Net zoals de Favorieten op internet. Bovendien kunt u ook uw eigen gemaakte componenten hierin plaatsen.

Hieronder ziet u een afbeelding van het codevenster. Nu zijn er twee tabbladen en u kunt altijd wisselen van de design en code door op een van de tabbladen te klikken.



Het verschil met Visual Basic 6 en Visual Basic .NET
Bovenstaande afbeelding laat een heel ander codevenster zien dan Visual Basic 6 altijd laat zien. Tegenwoordig zijn ook de formulieren klassen objecten geworden en geen zwarte dozen meer. Er zijn daardoor wat verschillen:

- Een formulier is nu een klasse, dus er moet een object worden aangemaakt om het formulier te kunnen gebruiken;
- Formuliergegevens moeten doorgestuurd worden naar dialoogformulieren en andersom. Er wordt nu d.m.v. de knoppen OK en Annuleren bepaald of de gegevens opgehaald mogen worden. Een extra variabele in een dialoogformulier declareren en die op True of False zetten, zodat bepaald kan worden of de gegevens opgehaald mogen worden, is niet meer nodig;
- Een control array maken van een control bestaat niet meer. Nu moet er een groep worden gemaakt van meerdere controls. Door ze dezelfde eventnaam te geven kunnen ze in één keer geprogrammeerd worden, zie de volgende bulletin meer daarover;
- Zoals u op de afbeelding ziet, zal de Load event niet meer standaard verschijnen. Bovendien zijn de events zeer uitgebreid. Ze hebben nu argumenten, zoals het object Sender. Die bepaald welk object op het formulier is geklikt. In de code kan dan het juiste object worden bewerkt. Het object kan dus van alles zijn, zoals een optieknop of een keuzevak. Dit heeft te maken met Polymorfisme, een techniek waarmee vanuit het basisobject elk ander object bepaalt kan worden;
- Doordat eerst een object aangemaakt moet worden voor het formulier, moet toch ergens het object vandaan komen. Er is één module nodig waar het hoofdprogramma in moet komen. In de subroutine Main moet het object aangemaakt worden, zodat de methode Run het formulier kan uitvoeren. Alleen het MDI formulier is nodig. Andere formulieren moeten in de events aangemaakt en uitgevoerd worden, zoals in een menuitem;
- Objecten hoeven niet meer met het sleutelwoord Set aangemaakt te worden, het kan nu op twee manieren: in de declaratie of na de declaratie en alleen in de toekenning.

In de volgende bulletins gaan we eens nader bekijken hoe we een formulier kunnen maken en aan kunnen sturen.

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