Nieuwsbrief

18de jaargang december 2011 Nummer 4

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	*Blz*
Gordon Rahman	29

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	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.hcc.nl

Redactioneel

Na veel drukte werd helaas de nieuwsbrief wat vertraagd, maar ik heb het toch af kunnen krijgen.

In deze nieuwsbrief kunt u veel variatie vinden in verschillende BASIC versies, PowerBASIC, Liberty BASIC, Just BASIC, GW-BASIC en Visual Basic 2008. Van deze laatste twee versies presenteren we alweer deel 9, waarin u uitgelegd wordt hoe we vlakken in de ruimte kunnen tekenen. Vooral Programma 28 zal u een prachtig voorbeeld geven.

Voor Liberty BASIC gebruikers heb ik leuke onderwerpen; stylebits en statictext mogelijkheden en meer uitleg over het commando NOMAINWIN. Wat doet dat commando NOMAINWIN eigenlijk?

Mocht de nieuwsbrief na de Kerstdagen verschijnen dan wens ik u alsnog goede feestdagen en een goed uiteinde, ook namens het bestuur van de hcc! BASIC IG.

Marco Kurvers

We hebben bestuurlijk een turbulent jaar achter de rug. Onze voorzitter Willem Gobel is eind 2010 teruggetreden. Willem is niet erg actief meer voor de HCC of de BASIC IG. Hij heeft het te druk met zijn overige hobby's en bezigheden. Gelukkig is het overgebleven bestuur tijdelijk opnieuw gerangschikt en aangevuld en heeft zich prima erdoorheen geslagen. Jan van der Linden is de nieuwe voorzitter (a.i). Piet Boere heeft te kennen gegeven nog penningmeester te zullen blijven, terwijl de overige leden op hun posten zijn gebleven. Marco Kurvers is redacteur van de Nieuwsbrieven gebleven.

Voorspoedig 2012

Het bestuur.

BASIC leren – PowerBASIC (2).

In de vorige nieuwsbrief besprak ik als laatste het symbooltype ‘Binary coded decimal (BCD) floating point (@@). Het ‘BASIC leren deel 1’ was al gelijk een grote eerste les over PowerBASIC, waarbij het meeste alleen maar over de speciale symbolen achter de variabelen (de klasse) ging, en over het weergavebereik van de types. Ik zal u gerust stellen; het waren ze echter nog niet allemaal. Er zijn nog veel meer soorten in PowerBASIC. Iedereen die in een andere Basic versie werkt moet hieraan wennen, want alleen PowerBASIC beschikt over zoveel symbolen en types.

Natuurlijk, u zou de types na kunnen maken in uw Basic versie, maar dan nog kunnen we niet spreken van een gereserveerd sleutelwoordtype.

Variable-length strings ($)
Variable-length strings zijn variabelen die tekens bevatten van verschillende lengtes. Elke stringvariabele gebruikt twee bytes, het locatieadres, waar een handle naar verwijst. De handle verwijst dus niet naar de hele string van de variabele, maar alleen naar het locatieadres van die twee bytes.

Onthoud dat uw programma het hele programmageheugen als stringruimte mag gebruiken. Er bestaat geen grens aan een maximum van 64K zoals we dat wel in de “interpretive” BASIC of Turbo Basic 1.x hebben.

Meer over dit string geheugen en over het $STRING metastatement kunt u vinden in de PowerBASIC help, in (hand)boeken of via Google door het trefwoord ‘metastatement’ in te toetsen.

Deze strings worden benoemd met het dollarteken ($) of door gebruik van het DEFSTR type definitie. U kunt ook de variable-length variabelen declareren door gebruik van het STRING sleutelwoord met het DIM statement, bijvoorbeeld:

```
DIM I AS STRING
```

Flex strings ($$)
Flex strings is een nieuwtje in PowerBASIC. Deze stringklasses worden benoemd met het dubbele dollarteken ($$) of door gebruik van het DEFFLX type definitie. U kunt ook flex string variabelen declareren door gebruik van het FLEX sleutelwoord met het DIM statement, bijvoorbeeld:

```
DIM I AS FLEX
```

Flex strings kunt u zien als karakter arrays. Deze worden gedeclareerd bij gebruik van het MAP statement:

```
MAP FlexStringName$$ * FlexStringLength      ' enkele flex string
DIM ArrayName$$(subscripts)                  ' array van flex
MAP ArrayName$$() * FlexStringLength         ' strings
DEFFLX A-C          ' Variabelen waarvan de namen beginnen met a, b
                    ' of c. Deze zijn nu standaard flex strings.
```

Net als een reguliere string, neemt een flex string variabele ook twee bytes in beslag dat een handle heeft naar een locatieadres. Al het beschikbare geheugen mag worden gebruikt om flex strings in op te slaan, zoals ook bij reguliere strings, en de maximumlengte van een flex string is standaard 32750 tekens, de huidige string segmentlengte.

Hieronder is een voorbeeld dat laat zien wat de flex strings voor mogelijkheden hebben:

```
INPUT #1, NumCustomers%
INPUT #1, CustNameLen%, CustAddrLen%, CustRestLen%
DIM CustInfo$$(1:NumCustomers%), CustName$$(1:NumCustomers%), _
    CustAddr$$(1:NumCustomers%), CustRest$$(1:NumCustomers%)
MAP CustInfo$$() * CustNameLen% + CustAddrLen% + CustRestLen%, _
    CustNameLen% AS CustName$$(), CustAddrLen% AS CustAddr$$(), _
    CustRestLen% AS CustRest$$()
PRINT  "Hier zijn de namen van de personen: "
FOR Record% = 1 TO NumCustomers%
  INPUT #1, CustInfo$$(Record%)
  PRINT CustName$$(Record%)
NEXT Record%
CustName$$(NumCustomers%)= "END"
```

Dit leest de aantal personen (*NumCustomers%*) van een database uit een bestand #1. Daarna leest het de lengtes van de naam (*CustNameLen%*), het adres (*CustAddrLen%*) en de plaats (*CustRestLen %*) velden, dat elk persoon “record” opmaakt in de database.

Vier flex string arrays zijn gedimensioneerd om informatie vast te houden voor elke persoon.

- *CustInfo$$()* houdt de gehele set van informatie bij van elke persoon (naam, adres en de plaats).
- *CustName$$()* houdt de naam van elke persoon, gedefinieerd met de eerste lengte van *CustNameLen%* aantal karakters uit het aangegeven element van *CustInfo$$()*.
- *CustAddr$$()* houdt het adres van elke persoon, gedefinieerd met de volgende lengte van *CustAddrLen%* aantal karakters uit het aangegeven element van *CustInfo$$()*.
- *CustRest$$()* houdt de plaats van elke persoon, gedefinieerd met de rest van *CustRestLen%* aantal karakters uit het aangegeven element van *CustInfo$$()*.

Nadat elk record gelezen is in de *CustInfo$$()* array zal de persoonsnaam geprint worden uit het aangegeven element van de *CustName$$()* array. De laatste regel in het voorbeeld kent het woord “END” toe aan het persoonsnaamveld van het laatste record plus genoeg ruimte die overblijft met het verschil van het aantal karakters uit *CustNameLen%*. Onthoud dat dit niet alleen de waarde in *CustName$$(NumCustomers%)* veranderd, maar ook de waarde veranderd van het eerste aantal *CustNameLen%* karakters van *CustInfo$$(NumCustomers%)*.

Flex strings zijn zeer flexibel: De lengtes van alle strings en arrays, betrokken in het voorbeeld, worden dynamisch bepaald in *run-time*, alles uit de vier gelezen integers vanuit het gegevensbestand.

Een andere string is de fixed length string. Het verschil met de flex string is dat de fixed length string een vaste lengte heeft die niet in run-time gewijzigd kan worden.

U kunt een fixed length string variabele declareren met gebruik van het STRING*x sleutelwoord met het DIM statement, bijvoorbeeld:

```
DIM I AS STRING*10
```

De fixed length strings worden ook gebruikt als velden in gebruiker gedefinieerde types of unions.

Constanten

PowerBASIC programma's verwerken twee onderscheiden gegevenssoorten: *constanten* en *variabelen*. Een variabele kan telkens een andere waarde hebben als een programma draait. Een constante waarde is vast (fixed) tijdens de compileertijd en kan niet worden gewijzigd tijdens de programma-uitvoer. PowerBASIC kent drie constant-types: stringconstanten, numerieke constanten en equates (gelijkstellers).

Stringconstanten
Dit zijn eenvoudige groepen van tekens tussen aanhalingstekens, bijvoorbeeld:

```
"Dit is een string"
"3.14159"
"Marco Kurvers, 3773 ZJ  Barneveld"
```

Als een stringconstante het laatste onderdeel is in een regel, zijn de gesloten aanhalingstekens optioneel:

```
PRINT "Dit is slordig maar wel toegestaan.
```

Numerieke constanten
Dit zijn constanten die alleen numerieke waarden hebben en alleen cijfers van 0 tot en met 9 en een decimale punt hebben. Negatieve constanten hebben een minus teken (-) nodig; een plus teken (+) is optioneel voor positieve constanten. De precieze waardengrens wordt bepaald uit welke type de constante zal moeten zijn (integer, long integer, kwart integer, enkele precisie, dubbele precisie, externe precisie, binair gecodeerde decimale vaste punt of binair gecodeerde decimale drijvende punt) en door PowerBASIC als constante gebruikt wordt.

U kunt ook een grens opgeven met welke precisie de constante opgeslagen wordt door één van de typespecificaties (%, &, &&, ?, ??, ???, !, #, ##, @, @@). Dit vermogen wordt zeer belangrijk bij het werken met BCD vaste en drijvende punt getallen. Bijvoorbeeld `AVAR@@ = 1.1` slaat de uitgebreide precisie vertegenwoordiging van 1.1 op, wat dus 1.100000023841858 is in de BCD drijvende punt variabele `AVAR@@`. Om de exacte waarde 1.1 op te slaan, moet u met het BCD drijvende punt type aanduiding (@@) opgeven:

```
avar@@ = 1.1@@
```

Als een numerieke constante niet gevolgd wordt door een typeaanduiding, worden de volgende regels gebruikt om te bepalen welke precisie nodig is om de waarde in op te slaan:

1. Als de waarde geen decimale punt bevat en het bereik 0 tot en met 255 is, slaat PowerBASIC de waarde op als een byte.
2. Als de waarde een geheel getal is van -32768 tot en met 32767 en buiten het bereik ligt van byte constanten, slaat PowerBASIC de waarde op als een integer.
3. Als de waarde een geheel getal is van 32768 tot en met 65535, slaat PowerBASIC de waarde op als een word.
4. Als de waarde een geheel getal is van -2^{31} tot en met $2^{31}-1$ inclusief (over -2 biljoen tot en met +2 biljoen) en buiten het bereik ligt van word constanten, slaat PowerBASIC de waarde op als een long integer.
   Langere integers worden bewaard als quad integers.
5. Tot slot als een integer-waarde positief is, de maximumwaarde voor een long integer overschrijdt en nog steeds binnen het bereik voor een double word valt, slaat PowerBASIC de waarde op als een double word.
6. Als de waarde een decimaalteken bevat en maximaal zes cijfers heeft, slaat PowerBASIC het als een enkele precisie floating point op.

7. Een numerieke constante met een decimaalteken en meer dan zes cijfers, maar minder dan 17 cijfers, of het gehele getal te groot is voor een quad integer maar klein genoeg om binnen het bereik van een dubbele precisie floating point te vallen, wordt opgeslagen in een dubbele precisie indeling. Grotere waarden (met maximaal 18 significante cijfers) worden opgeslagen in een uitgebreide precisie indeling. Bijvoorbeeld:

```
345.1               ' Een enkele precisie constante
1.10321             ' Een enkele precisie constante
1.103213            ' Een dubbele precisie constante
3453212.1234        ' Een dubbele precisie constante
1112223.4445556667  ' Een uitgebreide precisie constante
```

Wanneer het teken van een integer-constante niet is herkend (er is geen type-id), gebruikt PowerBASIC de volgende regels om te bepalen of de waarde met of zonder voorteken opgeslagen moet worden:

1. Als het getal een type-ID bevat, zal het getal met teken of zonder teken specifiek worden gezien als: byte, word en double word zonder teken; integer, long integer, quad integer met teken.
2. Als er geen type-ID is, zal het getal standaard zonder teken zijn tenzij de meest significante bit van het getal (de 7de, 15de, 31ste of 63ste) een één-bit is en het toonaangevende cijfer van het getal niet een nul is. Enkele voorbeelden zijn in volgorde:

```
32767??             ' Een word constante (zonder teken)
-40000              ' Een long integer constante (met teken)
32                  ' Een byte constante (zonder teken)
-32                 ' Een integer constante (met teken)
```

Equates

In PowerBASIC kunt u constanten gebruiken door te verwijzen met namen. Let wel, dat komt neer dat zij een globaal effect hebben; dat wil zeggen, ze zijn beschikbaar in uw programma. In tegenstelling tot variabelen kunt u een equate aan de linkerzijde van een toewijzingsinstructie slechts eenmaal gebruiken, en alleen een constante waarde (niet een variabele expressie) kan worden toegewezen. U kunt een expressie gebruiken, zolang alle delen van de expressie constant zijn. Bijvoorbeeld, de volgende equates zijn toegestaan:

```
%X = 1
%Y = 1 + 1
%Z = %X * %Y
```

Een waarde moet worden toegewezen aan elke equate voordat die verwezen kan worden, zelfs als die waarde nul is. Als u geen equate definieert, wordt er een fout gegenereerd tijdens het compileren.

U kunt ook gebruik maken van equates om de incidentie van “magische getallen” in uw programma’s te verlagen. Magische getallen zijn mysterieuze waarden die iets voor u betekenen wanneer u eerst een programma schrijft, maar niet wanneer u zes maanden later terugkomt. Dat is de reden waarom equates geschikt zijn en de programma’s bijzonder leesbaar maakt. Neem bijvoorbeeld een matrix uit schaakstukken die u wilt bijhouden. Als we definiëren

```
%MAXPIECES = 32
%NPARAM    = 3
%TYPE      = 1
%RANK      = 2
%FILE      = 3
%KING      = 1
%PAWN      = 2
```

dan kunnen we een matrixarray definiëren en statements maken als volgt:

```
DIM piece(1:%MAXPIECES, 1:NPARAM)
piece(1, %TYPE) = %KING
piece(1, %RANK) = 4
piece(1, %FILE) = 1
```

Dit zet een 32 x 3 matrixarray op. Het eerste element is de soort eenheid, de tweede en derde geven de huidige positie op het bord. Opmerking: hoe beter leesbaar is dit dan:

```
DIM piece(1:32, 1:3)
piece(1, 1) = 1
piece(1, 2) = 4
piece(1, 3) = 1
```

We konden een vergelijkbaar effect bereiken met behulp van commentaar, maar er is geen manier om te zorgen dat, wanneer het programma verandert, het commentaar zal worden bijgewerkt. Met behulp van equates komt het neer dat de noodzaak voor commentaar verminderd.

Ook komt het er op neer dat het niet alleen om een betere leesbaarheid gaat; het programma veranderen gaat gemakkelijker door alleen maar de definities van de equates te wijzigen. Bijvoorbeeld, stel dat u leiding geeft aan een kleuterschool en dat u de gegevens wilt bijhouden die afhangen van hoeveel kinderen u hebt. Bovendien wilt u rapporten per week uit kunnen printen. In plaats van het aantal op verschillende plaatsen te typen, kunt u beter, om voor elke week het te kunnen veranderen, het getal toewijzen aan een constante.

```
%NUMKIDS = 28
```

U kunt dan de constante *%NUMKIDS* gebruiken in uw hele programma.

```
' bereken het inkomen; het inschrijfgeld is €85 per week;
' ouders betalen wanneer hun kinderen de dagen wel of niet missen

income% = %NUMKIDS * 85
{statements}

' bereken de werkelijke aanwezigheid

attend% = %NUMKIDS - absent%
{statements}

' bereken hoeveel lunches het kost per kind; merk op het
' gebruik van een andere constante voor de kosten;
' die kan ook variëren!

perkid% = %LUNCHCOST / attend%

' bereken de nettowinst per kind na het betalen van de lunches
' (u hebt natuurlijk meer ruimte dan dit, maar we willen het
' simpel houden)

net% = (income% - perkid%) / %NUMKIDS

' en zo gaat het verder
```

Als uw beschrijving stabiel blijft zal uw programma nog steeds gemakkelijker te volgen zijn en als uw beschrijving gewijzigd wordt, moet u alleen de toewijzingsinstructies voor de constanten wijzigen, zodat u een herziend programma kunt uitvoeren. Denk aan de tijd dat u bespaart!

U kunt ook een waarde conditioneel aan een equate toewijzen met behulp van het $IF metastatement, bijvoorbeeld:

```
%BIGCLASS = 1

$IF %BIGCLASS
  %NUMKIDS = 40
$ELSE
  %NUMKIDS = 20
$ENDIF
```

Ook maken de equates de SELECT statements meer leesbaar:

```
SELECT CASE piece(x, %TYPE)
  CASE %KING
    { proces koning verplaatst }
  CASE %PAWN
    { proces pion verplaatst }
  CASE %QUEEN
    . . .
    . . .
END SELECT
```

Deze code is zinvol wanneer u na een lange tijd terugkomt.

Dit waren de twee lessen over PowerBASIC. Op internet kunt u meer vinden over de types van PowerBASIC.

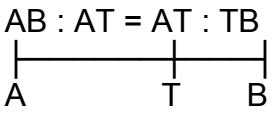
Mocht u vragen hebben, stuur ze dan naar mijn email adres die u in de vraagbakenlijst ziet staan

Marco Kurvers

Grafisch programmeren in GW-BASIC (9).

Voordat we naar het volgende onderwerp gaan, het tekenen van vlakken in de ruimte, zal nog één programma over het tekenen van driedimensionale figuren uitgelegd worden.

Programma 27 tekent de projectie van een regelmatig twintigvlak (icosaëder). Het is bekend dat er slechts vijf ‘echtregelmatige’ veelvlakken (polyeders) zijn, namelijk een tetraëder (regelmatig drievlak), een kubus (regelmatig zesvlak), een octaëder (regelmatig achtvlak), een dodecaëder (regelmatig twaalfvlak) en een icosaëder (regelmatig twintigvlak). Een icosaëder heeft 12 hoeken, 30 ribben en 20 vlakken. Een vlak is een gelijkzijdige driehoek. Als we Programma 22 willen gebruiken om een dergelijk icosaëder te tekenen dan hebben we de coördinaten van alle 12 hoekpunten nodig. Als we de icosaëder in een speciale stand zetten kunnen we de hoekpuntcoördinaten met behulp van de techniek van de ‘gulden snede’ relatief eenvoudig berekenen. Wie zich voor de hier achterliggende wiskundige theorie interesseert moet er maar eens een meetkundeboek op naslaan. De ‘gulden snede’-techniek deelt een lijnstuk AB in twee stukken AT en TB op een zodanige manier dat de volgende evenredigheid geldt:



Kiezen we voor AB de lengte 1, dan kunnen we de lengte van AT berekenen; immers

$$\frac{AB}{AT} = \frac{AT}{TB} \quad \text{dus} \quad \frac{1}{t} = \frac{t}{1-t} \quad \text{als} \quad t = AT$$

dus dan moet gelden $t^2 = 1 - t$ ofwel $t^2 + t - 1 = 0$.
Met de alom bekende abc-formule berekenen we nu

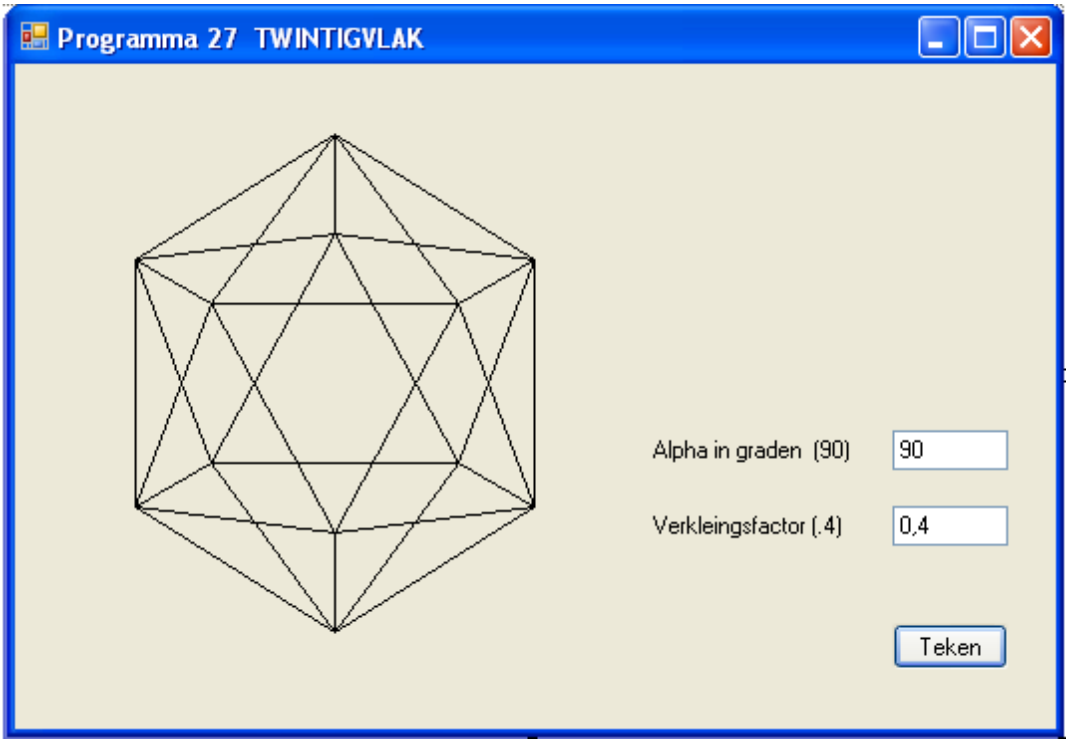
$$t_{1,2} = \frac{-1 \pm \sqrt{1+4}}{2}$$

De positieve wortel geeft: $t = \frac{-1 + \sqrt{5}}{2}$

Dit is de lengte van het grootste van de twee stukken die we krijgen als we een lijnstuk, ter lengte 1, vervolgens de ‘gulden snede’ in twee stukken verdelen. (t = 0,618; 1 - t = 0,382).
Deze waarde wordt in regel 180 berekend en in de regels 210-320 gebruikt om de hoekpuntcoördinaten van de icosaëder te berekenen. Om te zorgen dat het twintigvlak voldoende groot getekend wordt zijn alle coördinaten met 100 (F) vermenigvuldigd. Kies voor alpha (A) 90^0 en voor k (K) de waarde 0,4. Andere waarden vertekenen het beeld. Alpha = 180^0 en K = 0,6 geeft trouwens iets onverwachts!

```
100 '         programma 27       TWINTIGVLAK
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "ALPHA IN GRADEN    (90) "; A
150 INPUT "VERKLEININGSFACTOR (.4) "; K
160 U=160: V=160: H=.5: RD=4*ATN(1)/180
170 W=A*RD: C=K*COS(W): S=K*SIN(W)
180 T=(SQR(5)-1)/2 : F=100
190 CLS
200 DIM X(12), Y(12), Z(12), Z$(3)
210 X(1) = 0   : Y(1) = F*T : Z(1) = -F
220 X(2) = 0   : Y(2) =-F*T : Z(2) = -F
230 X(3) = F   : Y(3) = 0   : Z(3) = -F*T
240 X(4) =-F   : Y(4) = 0   : Z(4) = -F*T
250 X(5) = F*T : Y(5) = F   : Z(5) = 0
260 X(6) =-F*T : Y(6) = F   : Z(6) = 0
270 X(7) = F*T : Y(7) =-F   : Z(7) = 0
280 X(8) =-F*T : Y(8) =-F   : Z(8) = 0
290 X(9) = F   : Y(9) = 0   : Z(9) = F*T
300 X(10)=-F   : Y(10)= 0   : Z(10)= F*T
310 X(11)= 0   : Y(11)= F*T : Z(11)= F
320 X(12)= 0   : Y(12)=-F*T : Z(12)= F
330 FOR N=1 TO 3
340    READ Z$(N) : L=LEN(Z$(N))
350    FOR M=1 TO L-1 STEP 2
```

```
360       A$=MID$(Z$(N),M,1)   :I=ASC(A$)-64
370       B$=MID$(Z$(N),M+1,1):J=ASC(B$)-64
380       X1=INT(U+X(I)+C*Y(I)+H)
390       Y1=INT(V-S*Y(I)-Z(I)+H)
400       X2=INT(U+X(J)+C*Y(J)+H)
410       Y2=INT(V-S*Y(J)-Z(J)+H)
420       LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
430    NEXT M
440 NEXT N
450 A$=INKEY$: IF A$="" THEN 450
460 CLS: KEY ON: END
470 DATA "BCCEEFFDDBABACAEAFAD"
480 DATA "BHHDDJJFFKKEEIICCGGB"
490 DATA "GHHJJKKIIGLGLHLJLKLI"
```



De Visual Basic listing hieronder laat weer een verschil zien tussen GW-BASIC en de VB 2008 versie. De hele initialisatie voor de hoekcoördinaten hoeven niet meer regel voor regel geprogrammeerd te worden. Dankzij de accolades kunnen de vaste waarden direct in een dynamische array ingegeven worden.

Merk ook op dat de N-lus van 0 tot 2 loopt en de ASC waarde afgetrokken wordt met 65 en niet 64. Het is noodzaak er op te letten dat een array begint met element 0.

```
Public Class frmProg27

    Private Const U As Integer = 160
    Private Const V As Integer = 160
    Private Const H As Double = 0.5
    Private Const F As Integer = 100
    Private T As Double = (Math.Sqrt(5) - 1) / 2
    Private X() As Double = _
    { _
        0, 0, F, -F, F * T, -F * T, F * T, -F * T, F, -F, 0, 0 _
    }
    Private Y() As Double = _
    { _
        F * T, -F * T, 0, 0, F, F, -F, -F, 0, 0, F * T, -F * T _
    }
    Private Z() As Double = _
    { _
        -F, -F, -F * T, -F * T, 0, 0, 0, 0, F * T, F * T, F, F _
```

```
    }
    Private strZ() As String = _
    { _
        "BCCEEFFDDBABACAEAFAD", _
        "BHHDDJJFFKKEEIICCGGB", _
        "GHHJJKKIIGLGLHLJLKLI" _
    }
    Private Sub btnTeken_Click(…, …) …
        Refresh()
    End Sub

    Private Sub frmProg27_Paint(…, ByVal e As …PaintEventArgs) …
        If txtA.Text() <> "" And txtK.Text() <> "" Then
            Dim A As Integer = CInt(txtA.Text())
            Dim K As Double = CDbl(txtK.Text())
            Dim RD As Double = 4 * Math.Atan(1) / 180
            Dim W As Double = A * RD
            Dim C As Double = K * Math.Cos(W)
            Dim S As Double = K * Math.Sin(W)
            For N As Integer = 0 To 2
                Dim L As Integer = strZ(N).Length()
                For M As Integer = 1 To L - 1 Step 2
                    Dim I As Integer = Asc(Mid(strZ(N), M, 1)) - 65
                    Dim J As Integer = Asc(Mid(strZ(N), M + 1, 1)) - 65
                    Dim X1 As Integer = Int(U + X(I) + C * Y(I) + H)
                    Dim Y1 As Integer = Int(V - S * Y(I) - Z(I) + H)
                    Dim X2 As Integer = Int(U + X(J) + C * Y(J) + H)
                    Dim Y2 As Integer = Int(V - S * Y(J) - Z(J) + H)
                    e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
                Next
            Next
        End If
    End Sub
End Class
```

Belangrijk! Toets niet het geraamte van de event subroutines in, maar open het codevenster door te klikken op de controls, zoals op de Teken knop. Na het klikken zal Visual Basic automatisch de subroutines aanmaken.
U hoeft ook niet op de drie-punten te letten die in de subroutines staan. Er staan normaal gesproken argumenten – de namen van de parameters – die voor u niet van belang zijn.

We hebben ons uitvoerig beziggehouden met het tekenen van driedimensionale lichamen. We herhalen hiervan nog eens de belangrijkste punten:

1. We gebruiken de parallelprojectie om een driedimensionaal lichaam met n hoekpunten P(x,y,z) in een plat vlak te tekenen. Voor de projectiehoek alpha (α) gebruiken we bij voorkeur de waarde 45^0 of 60^0. De schuin-naar-achteren lopende ribben worden korter getekend dan ze in werkelijkheid zijn. Als verkleiningsfactor kiezen we vaak 1/2 of 1/3.
2. Elk punt P(x,y,z) van het ruimtelijke lichaam wordt op een punt P'(x',y') in het projectievlak geprojecteerd. De hierbij gebruikte projectieformules zijn:

$$x' = U + x + k.y.\cos\alpha$$
$$y' = V - k.y.\sin\alpha - z$$

Zoals gebruikelijk zijn U en V de coördinaten van het midden van het beeldscherm. Tot nu toe

hebben we U = 160 en V = 160 verondersteld. We nemen nu echter U = 320 en V = 160 en gebruiken de schermresolutie 640x325. De FNX functie wordt ook nu niet gebruikt.

In de volgende BASIC programma's zien we de bovenstaande transformatievergelijkingen in de vorm:

```
XG = INT(U+XX+C*YY+H)
YG = INT(V-S*YY-Z+H)
```

De betekenis van de gebruikte variabelen is:

XG, YG	:	beeldschermcoördinaten x',y'
XX, YY	:	de lopende coördinaten x,y
C	:	K*COS(W*RD)
S	:	K*SIN(W*RD)
W	:	projectiehoek in graden
RD	:	factor π/180 voor omrekenen van graden in radialen
K	:	verkleiningsfactor

Tekenen van driedimensionale functies.
Als paradepaardje van menige demonstratie van Hoge-Resolutie-Graphics wordt vaak een programma gebruikt dat een of andere fraaie grafiek van een driedimensionale functie tekent. Als leek vraag je je af hoe ze weten welke functies ze moeten nemen, hoe het tekenen van de grafiek van zo'n functie geprogrammeerd wordt, en hoe ze het programma-technisch voor elkaar krijgen dat de 'onzichtbare lijnen' ook inderdaad niet getekend worden. Bij dergelijke demonstraties wordt doorgaans geen programmalisting getoond. Mocht dit wel het geval zijn dan is het programma vaak zo machine-afhankelijk dat het omzetten van het programma naar een andere machine lastig, zo niet ondoenlijk is.

In dit onderwerp hopen we u antwoord te kunnen geven op de volgende vragen:

1. Hoe vind ik 'mooie' driedimensionale functies?
2. Hoe ziet een algemeen programma voor het tekenen van een dergelijke functie eruit?
3. Hoe onderdruk je het tekenen van de onzichtbare lijnen (hidden lines)?

We beginnen met de definitie van een driedimensionale functie.

Elke functie van de vorm z = f(x,y) heet een driedimensionale functie en vormt een, meestal, gekromd vlak in een driedimensionaal (ruimtelijk) coördinatenstelsel. Speciaal voor niet-wiskundigen volgt nu een voorbeeld van een functie z = f(x,y) en het gekromde vlak dat als 'grafiek' bij de functie hoort.

Stel dat we voor z = f(x,y) de volgende vergelijking nemen:

$$z = e^{-(x^2+y^2)}$$ (e is het grondtal van de natuurlijke logaritme; e = 2,718284183…)

Voor elk punt (x,y) in het (platte) X-Y-vlak kunnen we nu de daarbij horende waarde van z uitrekenen. Als x = 0,5 en y = 0,1 dan is

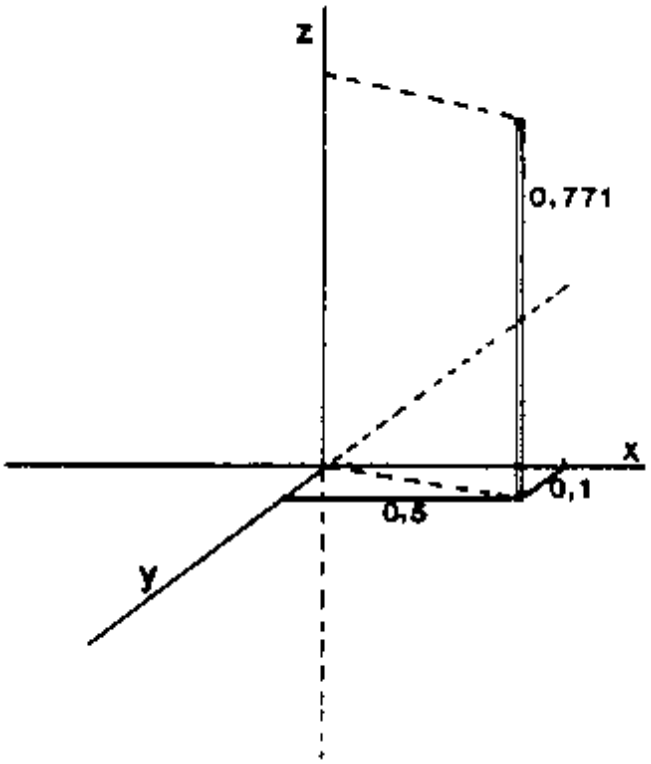
$$z = e^{-(0,25+0,01)} \rightarrow$$

$$z = e^{-0,26} \rightarrow$$

z = 0,771, want 2,71828… tot de macht -0,26 is 0,771.

Zet nu (in gedachten) in het voetpunt P(0,5; 0,1) in het X-Y-vlak een staafje met een lengte van 0,771 rechtop. Doe ditzelfde voor nog een groot aantal punten (x,y).
Leg nu over al deze staafjes een elastische folie. Dit (golvende) stuk folie vormt een goed model van het vlak met vergelijking

$$z = e^{-(x^2+y^2)}.$$



Hoe dit eruit zou zien kunt u zien bij Programma 28.

Antwoord op de eerste vraag.
Zoek met behulp van Programma 7, uit een aantal nieuwsbrieven terug, een willekeurige continufunctie die symmetrisch ten opzichte van de y-as is. De grafiek van de functie moet 'bergen en dalen' vertonen (de functie moet maxima en minima hebben). Erg geschikt zijn trigonometrische functies (sin, cos) en combinaties van deze functies. Ook geschikt zijn functies waarin alleen termen voorkomen met 'even-machten' van x en exponentiële functies.

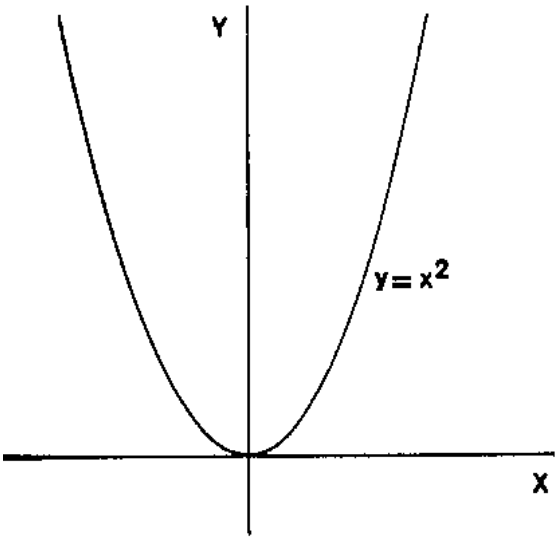
Enkele voorbeelden:

$$y = e^{-x^2} \quad ; \quad y = \sin(x)$$

$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7}$$

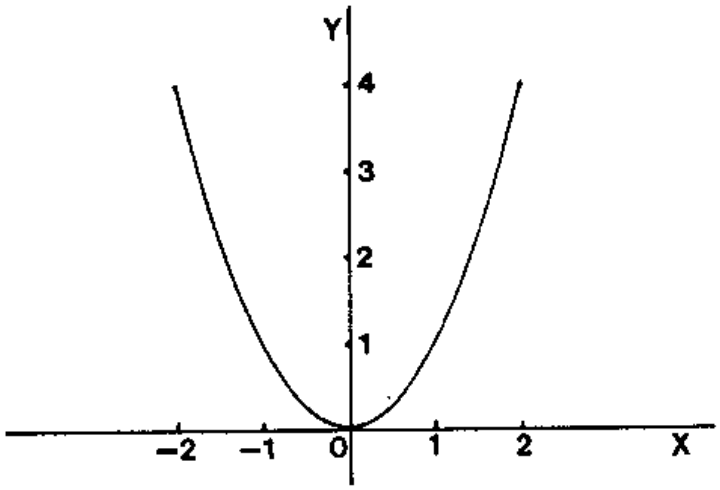
Bekijk nu de grafiek van zo'n functie in het interval $-a \le x \le a$. Dit 'stuk grafiek' gaan we draaien rond de y-as. Zo ontstaat, ten opzichte van de y-as, een draai-symmetrisch driedimensionaal vlak. Als we nu de y-as als z-as kiezen, dan wordt de vergelijking van een dergelijk draai-symmetrisch ruimtelijk vlak (rond de z-as) van de vorm

$$z = f(r) \quad \text{met} \quad r = \sqrt{x^2+y^2}$$

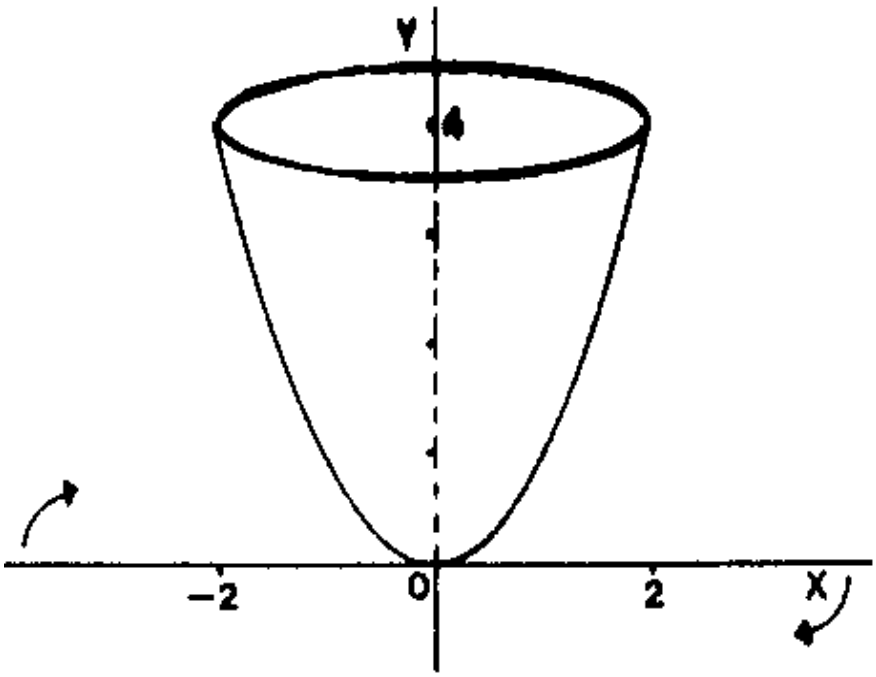


We zullen dit aan een eenvoudig voorbeeld proberen te verklaren. We kiezen als voorbeeld de eenvoudige paraboolvergelijking $y=x^2$.

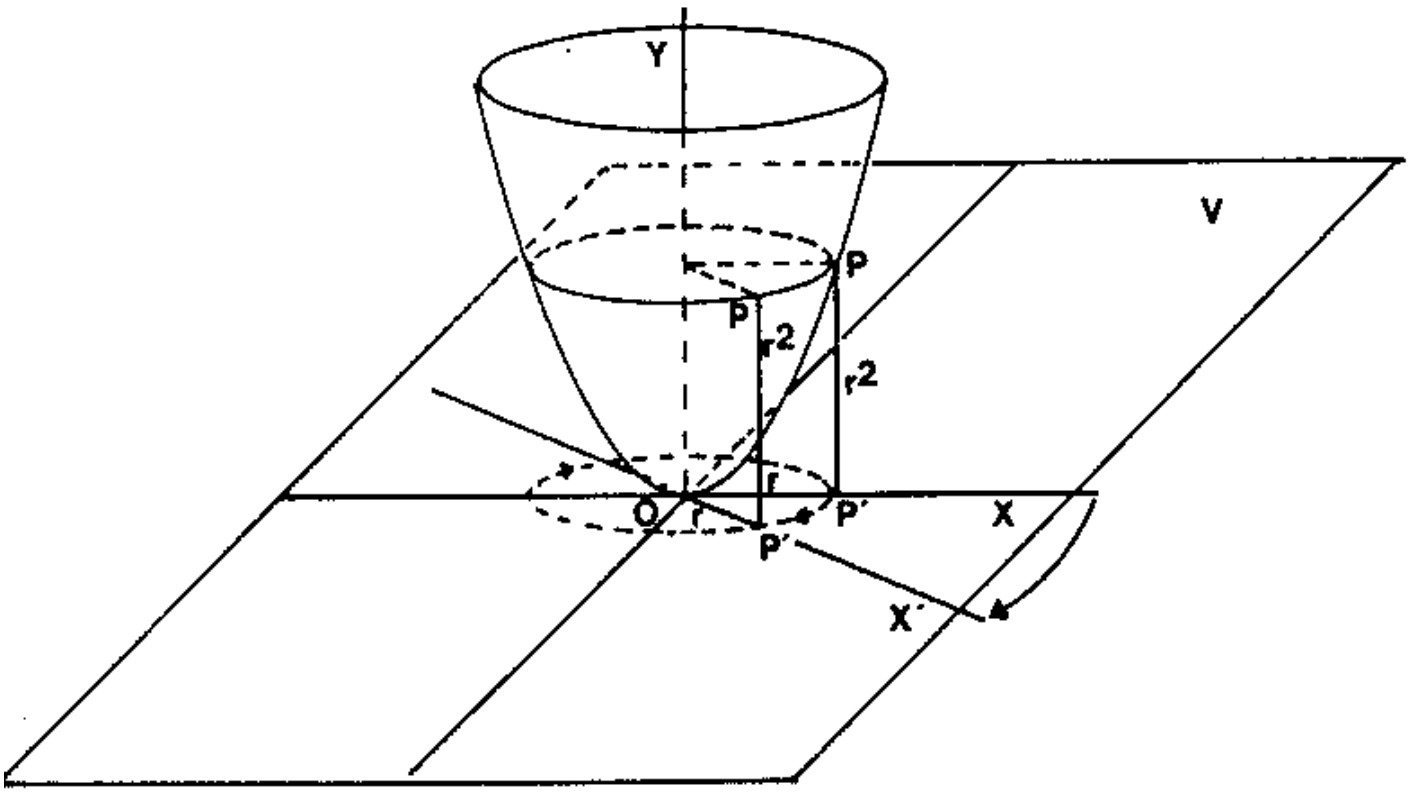
Stel we bekijken de grafiek voor $-2 \leq x \leq 2$:



Als we nu de x-as rond de y-as laten draaien (de x-as komt als het ware loodrecht het papier uit), dan ontstaat er een draai-symmetrisch vlak rond de y-as. Dit is een kegel:



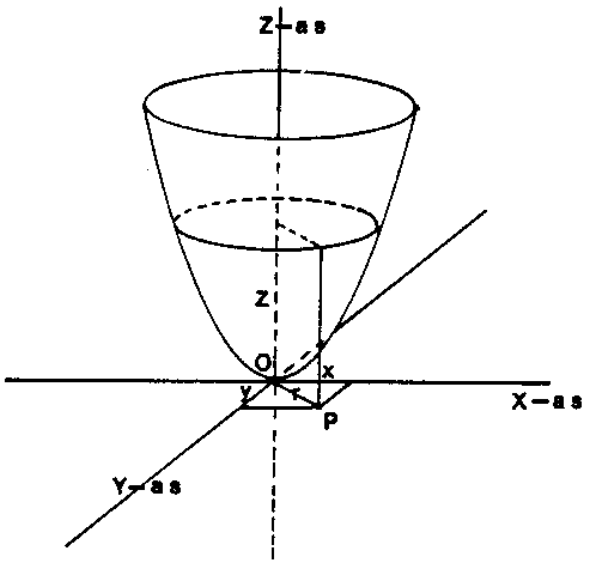
Elk punt P op de kegelmantel heeft de eigenschap $y=r^2$, waarbij r de afstand is van het geprojecteerde punt P' tot het punt O. We kunnen het vlak V dat door de ronddraaiende x-as wordt gevormd als volgt tekenen:



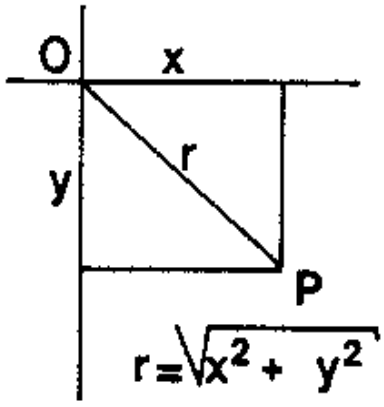
Welnu, als we nu de z-as als y-as nemen en we nemen de y-as in vlak V loodrecht op de x-as, dan zien we dat de afstand r geschreven kan worden als

r = $\sqrt{x^2+y^2}$ (stelling van Pythagoras, zie je in de volgende tekening)

en nu wordt $y=r^2$ geschreven als $z=r^2$ met $r^2 = x^2+y^2$, dus de vergelijking van de paraboloïde (kegel) wordt dan



z = x^2+y^2



Dus $y=x^2$ wordt bij draaiing om de y-as en bij een z-as die de plaats van de y-as inneemt $z = x^2+y^2$.
Voor de bovengenoemde functies geldt het volgende:

$$y = e^{-x^2} \quad \text{wordt} \quad z = e^{-(x^2+y^2)}$$

$$y = \frac{\sin(x)}{x} \quad \text{wordt} \quad z = \frac{\sin(r)}{r}$$

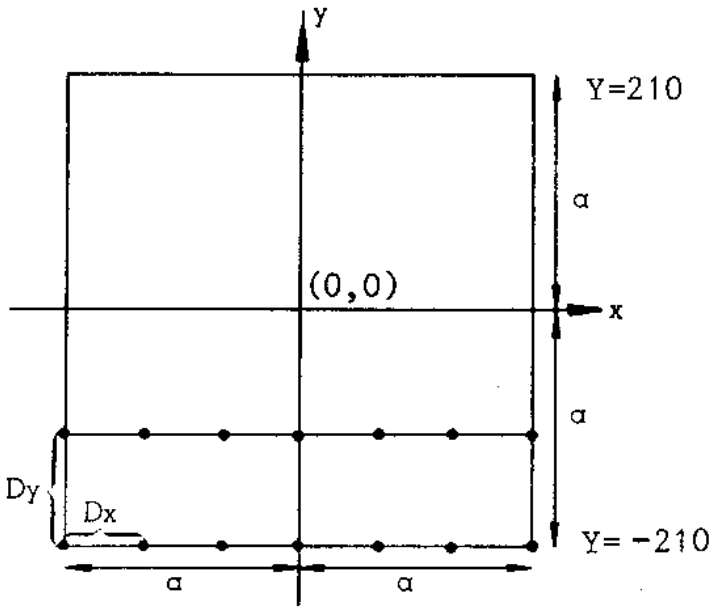
$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7} \quad \text{wordt}$$

$$y = \cos(r) - \frac{\cos(3r)}{3} + \frac{\cos(5r)}{5} - \frac{\cos(7r)}{7}$$

met $r = \sqrt{x^2+y^2}$

Voor deze klasse van draai-symmetrische figuren ontwikkelen we een algemeen tekenprogramma.

Antwoord op de tweede vraag
Bekijk de onderstaande tekening. U kijkt recht bovenop het ruimtelijke vlak van een driedimensionale functie. We willen dat het op het grondvlak geprojecteerde vlak een vierkant is met zijden van 2a. Straks zullen we zien dat, als we een mooie tekening van zo'n ruimtelijk vlak op het beeldscherm willen maken, a=210 een heel goede keus is.



Als 'doorgewinterde' programmeur ziet u natuurlijk direct dat het programma voor het tekenen van het vlak een geneste FOR-NEXT (dx,dy) zal bevatten! Als we een vlak als grafiek van z = f(x,y) willen tekenen, moeten we namelijk voor een aantal combinaties van x en y de functiewaarde z = f(x,y) berekenen. We beginnen met y = -210 en laten x met stapjes dx (bijvoorbeeld dx=3) van -210 naar +210 lopen. Voor elke combinatie x,y met y = -210 berekenen we de functiewaarde z = f(x,y). Zo ontstaan de punten (x,y,z) van het ruimtelijke vlak waarvan de geprojecteerde punten (x,y) in de tekening op de onderste zijde van het vierkant als bolletjes getekend zijn. Nu verhogen we y met dy en laten x weer van -210 tot +210 lopen. Dit geeft de tweede reep van het vlak. Op deze wijze ontstaat het hele vlak, waarvan de punten (x,y,z) natuurlijk nog getransformeerd moeten worden naar beeldschermcoördinaten (x',y').

Denk erom dat we in dit hoofdstuk uitgaan van een beeldscherm met 640 bij 325 beeldpunten en niet van 320 bij 200 zoals we tot nu toe gedaan hebben. Voor deze driedimensionale functies willen we graag het hele horizontale schermbereik (0-640) benutten. De DEF FNX opdracht zullen we in de komende drie programma's dan ook niet gebruiken.

Een eerste globale opzet voor het tekenen van een 'vierkant'-stuk vlak is:

```
FOR Y = -210 TO 210 STEP DY
      FOR X = -210 TO 210 STEP DX
            GOSUB 1000
            :
      NEXT X
NEXT Y
```

In de subroutine 1000 wordt voor een punt (x,y) de waarde van z = f(x,y) berekend. Met de bekende transformaties wordt vervolgens het punt P(x,y,z) overgebracht naar een punt P'(x',y') op het beeldscherm. Nu kan het berekende punt P(x,y,z), dat in het vlak ligt, als het punt P'(x',y') op het scherm worden getekend.

Jammer genoeg lenen niet alle driedimensionale functies zich voor de intervallen -210 ≤ x ≤ 210 en -210 ≤ y ≤ 210; vandaar dat de waarde voor a met een INPUT opdracht ingelezen wordt. Als we in het programma toch -210 ≤ XX ≤ 210 en -210 ≤ YY ≤ 210 kiezen, moet de ingetoetste waarde A als volgt gebruikt worden:

$$X = XX \cdot \frac{A}{210} \quad \text{en} \quad Y = YY \cdot \frac{A}{210}$$

dat wil zeggen er geldt: $-A \le X \le A$ en $-A \le Y \le A$

De z-waarden van met name de trigonometrische driedimensionale functies zal tussen -1 en 1 liggen. Om deze waarden wat 'op te blazen' kan een vermenigvuldigingsfactor (K1) worden ingetoetst. Goede waarden voor K1 liggen tussen 30 en 100.

We kunnen ons programma nu als volgt opschrijven:

```
REM Dit is een programma-opzet voor
REM het tekenen van de functie z=f(x,y)
'
INPUT "PROJECTIEHOEK IN GRADEN(45-135)";W
INPUT "VERKLEININGSFACTOR      (0.5-1)";K
INPUT "RECHTERGRENS VOOR X     ( > 0 )";A
INPUT "VERGROTINGSFACTOR       (30-80)";K1
U=320 : V=160 : H=.5 : RD=4*ATN(1)/180
C=K*COS(W*RD) : S=K*SIN(W*RD)
DX=3 : DY=5 : AF=A/210
CLS
FOR YY= -210 TO 210 STEP DY
     Y=YY*AF
     FOR XX= -210 TO 210 STEP DX
          X=XX*AF: GOSUB 1000
          XG=INT(U+XX+C*YY+H):YG=INT(V-S*YY-Z+H)
'
'         Teken het punt P(XG,YG)
'
     NEXT XX
NEXT YY
A$=INKEY$: IF A$="" THEN 340
CLS: KEY ON: END
'
' Hier volgt in subroutine 1000 de beschrijving van
' de functie z=f(x,y)
```

Zoals beloofd laten we nu zien waarom de intervallen -210 ≤ XX ≤ 210 en -210 ≤ YY ≤ 210 zo geschikt zijn om driedimensionale functies te tekenen met een hoog oplossend vermogen. We willen graag de fraaie projectie met $\alpha=45^0$ en k=0,5 gebruiken. Wat worden dan de beeldschermcoördinaten XG en YG voor de hoekpunten 'linksonder' en 'rechtsboven' van het vierkant op pagina 17? Deze punten bepalen immers of de hele grafiek op ons beeldscherm van 640 bij 325 puntjes getekend kan worden. Voor K1 nemen we 50, dat ligt zo'n beetje tussen 30 en 100! Nemen we voor de z-waarde ook 50, dan gaat het punt (x,y,z) met coördinaten (-210,-210,50) over in het beeldschermpunt (XG,YG) met

XG = INT(320-210-0,5.210.cos(45)+0,5) ➔ XG = 36
YG = INT(160+0,5.210.sin(45)-50+0,5) ➔ YG = 184

Deze waarden liggen inderdaad binnen ons 'graphic-scherm' (0 ≤ XG ≤ 640 en 0 ≤ YG ≤ 325) en de 'breedte' (640) van het scherm wordt heel goed benut, kijk maar naar de coördinaten (XG,YG) voor het punt (210,210,50) rechtsboven:

XG = INT(320+210+0,5.210.cos(45)+0,5) ➔ 604
YG = INT(160-0,5.210.sin(45)-50+0,5) ➔ 36

We benutten dus haast de hele schermbreedte (36 ≤ XG ≤ 604) voor het tekenen van de driedimensionale figuur.

Op de onder- en bovenrand (YY=-210 en YY=+210) gelden vaak heel kleine z-waarden, zodat bij $\alpha=45^0$, k=0,5 en z=0 de grafiek op het scherm zal liggen tussen

↓z-waarde
YG = INT(160+0,5.210.sin(45)-0+0,5) ➔ 234
en YG = INT(160-0,5.210.sin(45)-0+0,5) ➔ 86

Ons tekenvlak is dus ongeveer 36 ≤ XG ≤ 604 èn
86 ≤ YG ≤ 234.

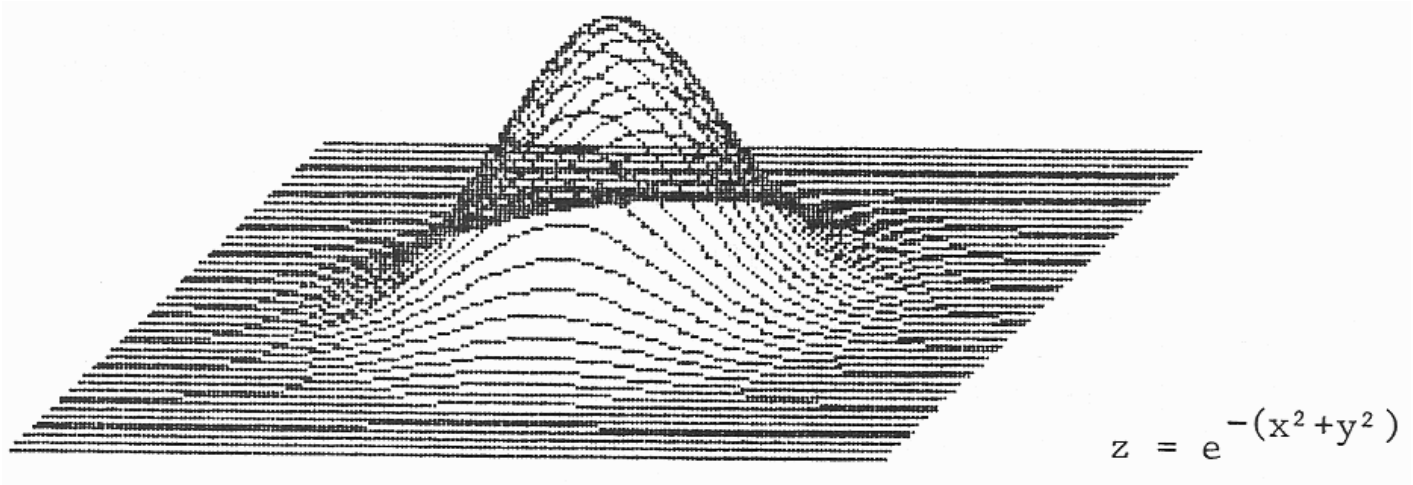
In het bovenstaande programmavoorstel wordt gebruik gemaakt van 'puntgraphics'. Hierbij worden de punten puntje voor puntje getekend. Om een enigszins acceptabele tekening te krijgen moeten erg veel 'puntjes' berekend worden (DX=1,DY=1). Dit duurt in BASIC, geneste lussen en vele trigonometrische berekeningen, erg lang. Voor de volgende tekeningen betekent dit al snel een tekentijd van meer dan 30 minuten, soms wel een uur. Dit is weinig bevredigend! Veel sneller gaat het als we de 'vectorgraphics'-methode gebruiken. Deze techniek hebben we in alle voorgaande programma's gebruikt; we verbinden steeds twee naast elkaar liggende punten door een recht lijntje. Hierbij hoeven we lang niet zoveel punten te berekenen als bij de 'puntgraphics'-techniek. Bovendien ziet de tekening er beter uit.

Technische informatie! Aan bovenstaande informatie doet het ons denken aan waarom BASIC een langzame programmeertaal was, vooral in de GW-BASIC tijd. Maar in plaats van elke keer BASIC de schuld te geven van het langzaam tekenen dat gedaan werd, moest niet alleen naar de taal worden gekeken maar ook naar het systeem. De computers zelf waren in die tijd nog helemaal niet snel. Nu de computers veel sneller zijn geworden, zien we ook dat BASIC een stuk sneller werkt en we nu ook in BASIC goed kunnen tekenen.

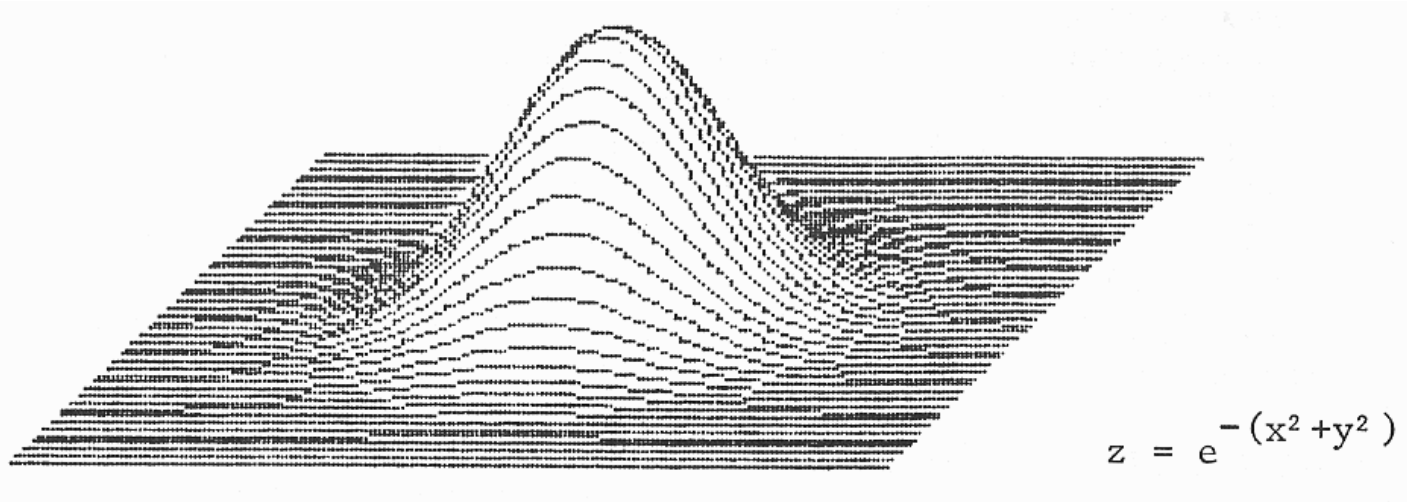
Als antwoord op de tweede vraag komen we dan met het volgende algemene programma voor het tekenen van draai-symmetrische ruimtelijke vlakken.

```
100 '    programma 28        GRAFIEK VAN Z=F(X,Y)
110 CLEAR ,19202 : SCREEN 105,,3,3
120 CLS : KEY OFF
130 INPUT "PROJECTIEHOEK IN GRADEN(45-135)";W
140 INPUT "VERKLEININGSFACTOR      (0.5-1)";K
150 INPUT "RECHTERGRENS VOOR X     ( > 0 )";A
160 INPUT "VERGROTINGSFACTOR       (30-80)";K1
170 U=320 : V=160 : H=.5 : RD=4*ATN(1)/180
180 C=K*COS(W*RD) : S=K*SIN(W*RD)
190 DX=3 : DY=5 : AF=A/210
200 CLS
210 FOR YY= -210 TO 210 STEP DY
220    Y=YY*AF
230    FOR XX= -210 TO 210 STEP DX
240       X=XX*AF: GOSUB 1000
250       XG=INT(U+XX+C*YY+H):YG=INT(V-S*YY-Z+H)
260       IF YG >= 0 AND YG <= 320 THEN 280
270         PRINT "FOUTE VERGROTINGSFACTOR":STOP
280       IF XX = -210 THEN X1=XG:Y1=YG:GOTO 320
290       X2=XG : Y2=YG
300       LINE (X1,Y1) - (X2,Y2),1
310       X1=X2 : Y1=Y2
320    NEXT XX
330 NEXT YY
340 A$=INKEY$: IF A$="" THEN 340
350 CLS: KEY ON: END
1000 Z=K1*EXP(-(X*X + Y*Y))
1010 RETURN
```

Met dit programma maakt u de tekening zoals u hieronder ziet. Voor de plotter die de auteur gebruikt heeft golden hierbij de volgende waarden: $W=75^0$, K=0,75, A=3 en K1=75.
Kies op uw computer de volgende waarden: W=45, K=0,5, A=3 en K1=80.
Wilt u zien hoe de puntgraphics er uit zien, neem dan DX en DY gelijk aan 1 en vervang de regels 260 t/m 310 door: 260 PSET(XG,YG).



U zult over de bovenstaande tekening niet tevreden zijn. De voorgrond is goed, maar in het achterste stuk zijn ook alle lijnen die voor ons onzichtbaar zijn getrokken. In onderstaande tekening zien we het resultaat met dezelfde waarden voor W, K, A en K1, maar waarin de niet-zichtbare lijnen ook niet getekend zijn. Hoe krijgen we dit voor elkaar?

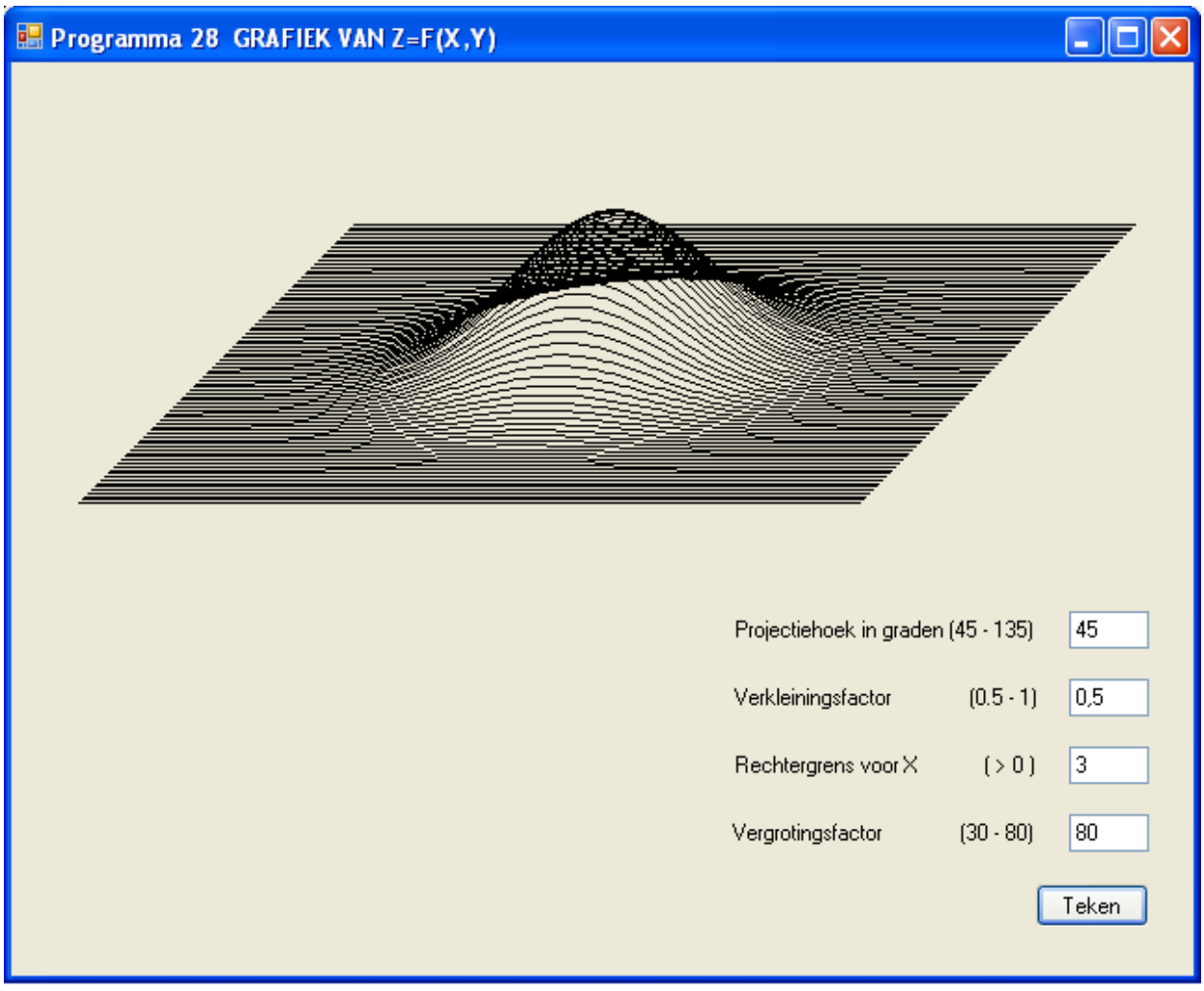


Gebruikt u de middenresolutie van 320 bij 200 puntjes, neem dan voor 210 de waarde 115 (regels 190, 210 en 280) en neem voor U de waarde 160 en voor V de waarde 100 (regel 170). Verander dan ook in regel 260 de waarde 320 in 200.

Tevens staat hieronder dezelfde listing in Visual Basic 2008 met ook de afbeelding op een formulier getekend. Merk op dat de top van de berg lager is. Dat kan komen omdat we niet meer met de resolutie werken waar vroeger mee gewerkt werd. U kunt er mee experimenteren en proberen om ook in Visual Basic de top net zo hoog te krijgen.

```
Public Class frmProg28

    Private Const U As Integer = 320
    Private Const V As Integer = 160
    Private Const H As Double = 0.5
    Private RD As Double = 4 * Math.Atan(1) / 180

    Private Function GetValueZ(ByVal intK1 As Integer, _
                               ByVal X As Double, ByVal Y As Double) As Double
        Return intK1 * Math.Exp(-(X * X + Y * Y))
    End Function

    Private Sub frmProg28_Paint(…, ByVal e As …PaintEventArgs) …
        If txtW.Text() <> "" And txtK.Text() <> "" And _
            txtA.Text() <> "" And txtK1.Text() <> "" Then
            Dim W As Integer = CInt(txtW.Text())
            Dim K As Double = CDbl(txtK.Text())
            Dim A As Integer = CInt(txtA.Text())
            Dim K1 As Integer = CInt(txtK1.Text())
            Dim C As Double = K * Math.Cos(W * RD), S As Double = K * Math.Sin(W * RD)
            Dim DX As Integer = 3, DY As Integer = 5, AF As Double = A / 210
            Dim X1, Y1, X2, Y2 As Integer
            Dim bStop As Boolean = False
            For YY As Integer = -210 To 210 Step DY
                Dim Y As Double = YY * AF
                For XX As Integer = -210 To 210 Step DX
                    Dim X As Double = XX * AF
                    Dim Z As Double = GetValueZ(K1, X, Y)
                    Dim XG As Integer = Int(U + XX + C * YY + H)
                    Dim YG As Integer = Int(V - S * YY - Z + H)
                    If YG >= 0 And YG <= 320 Then
                        If XX = -210 Then
                            X1 = XG
                            Y1 = YG
                        Else
                            X2 = XG
                            Y2 = YG
```

```
                            e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
                            X1 = X2
                            Y1 = Y2
                        End If
                    Else
                        MessageBox.Show("Foute vergrotingsfactor", "Melding")
                        bStop = True
                        Exit For
                    End If
                Next
                If bStop Then Exit For
            Next
        End If
    End Sub

    Private Sub btnTeken_Click(…, …) …
        Refresh()
    End Sub
End Class
```



In de volgende nieuwsbrief gaan we hiermee verder. Er zal dan een antwoord komen op de derde vraag hoe we de lijnen in het achterste stuk weg kunnen laten.

Bron: IBM- en GW-BASIC graphics van Academic Service
Tekst overname, tips en veranderingen: Marco Kurvers

Liberty BASIC verkennen.

In dit onderwerp laat ik u leuke mogelijkheden zien hoe we kunnen werken met StaticText en StyleBits met images. Hieronder staat de http link waar u onderstaande uitleg nog eens kunt nalezen op de site (Engelstalig):

http://lbpe.wikispaces.com/Stylebits+-+Statictext

Daarna komt er een leuk onderwerp over het commando NOMAINWIN.

Overzicht van de Stylebits Parameters.

De vier parameters voor stylebits zijn AddBit, RemoveBit, AddExtendedBit en RemoveExtendedBit. Een overzicht van deze vier parameters en een inleiding over Stylebits in het algemeen, kunt u bekijken door te klikken op de link: stylebits - windows. Stylebits is een Microsoft uitvinding om de “controls” (besturingselementen) aan te passen.

Stylebits en opgemaakte tekst (statictext).

Statictext is een besturingselement (control) dat u in een venster (window) kunt plaatsen, om teksten in het venster te tonen (display). De statement statictext #handle,”tekst”,xpos,ypos,width,height is reeds eerder in de Nieuwsbrieven uitgebreid behandeld. Met stylebits voor statictext kunt u de stijl (effecten) van deze control ook beïnvloeden.

Veel van de stylebits commando’s, die u gebruikt voor vensters en tekstvakken, kunnen ook worden gebruikt voor statictext besturingselementen. Aangezien statictext geen invoer van de gebruiker accepteert, beperken de meesten van die stylebits effecten zich tot tekstopmaak en randen. De StaticText tekst kan horizontaal worden aangepast met behulp van de volgende opdrachten:

```
_SS_LEFT
_SS_CENTER
_SS_RIGHT
```

StaticText kan ook verticaal worden gecentreerd met de opdracht:

```
_SS_CENTERIMAGE
```

Het is mogelijk om een mooi strakke look te creëren in uw tekst met behulp van de commando:

```
_WS_DISABLED
```

Wees voorzichtig met deze methode van “disable”. Want elk, via de stylebits uitgeschakeld besturingselement, vereist een API aanroep om ervoor te zorgen dat de statictext later weer ingeschakeld wordt. Hetzelfde doel kan worden bereikt met de eigen (native) Liberty BASIC commando **!Disable** en die biedt de programmeur de mogelijkheid de tekst eenvoudig zonder API te normaliseren met gebruik van de commando **!Enable**, zie later in de code.
Er zijn zeer interessante visuele effecten die bereikt kunnen worden met stylebits, zoals bij een gemaakte extra rand rond uw statictext.

Stylebits		
_WS_DLGFRAME, 0, 0, 0	.....	Verhoogde achtergrond
0, 0, _WS_EX_STATICEDGE, 0	.....	Lichtjes verzonken achtergrond

0, 0, _WS_EX_CLIENTEDGE, 0	.....	Diepere verzonken achtergrond
0, 0, _WS_EX_STATICEDGE or _WS_EX_CLIENTEDGE, 0	.....	Dubbele verzonken achtergrond
_WS_BORDER, 0, 0, 0	.....	Dunne lijnrand
0, 0, _WS_EX_DLGMODALFRAME or _WS_EX_CLIENTEDGE, 0	.....	Verhoogde frame
_WS_THICKFRAME, 0, _WS_EX_DLGMODALFRAME or _WS_EX_CLIENTEDGE, 0	.....	Dubbele verhoogde frame

Stylebits en API aanroepingen.

Stylebits staan ook aanpassingen van afbeeldingen toe in statictext (randen), maar alleen wanneer de afbeelding via een API aanroep naar user32.dll (SendMessageA) wordt weergegeven. De hoogte en breedte van de statictext zal, afhankelijk van de omschreven statictext afmetingen, de kleinere bitmap uitrekken. U kunt deze vervorming voorkomen door de hoogte en breedte van de statictext hetzelfde te houden als de geladen bitmap of met behulp van de _SS_CENTERIMAGE stylebits. Verzonken, verhoogde en platte effecten kunnen worden bereikt met de randvenster stylebits.

Ga naar de website zoals bovenaan gegeven. Klik met de rechter muisknop op deze bitmap en sla hem op als ‘boy.bmp’ Voer onderstaande code uit op dezelfde map als waar u de bitmap opgeslagen hebt. Zodra de code is uitgevoerd, zal het maximaliseren van het venster aantonen hoe de bitmap in de statictext uitstrekt om het gebied te vullen.

```
    WindowWidth=600
    WindowHeight=450

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2) - 14

    Nomainwin

    Loadbmp "boy", "boy.bmp"

    Statictext #main.st0, "", 10, 10, 54, 100
    Stylebits #main.st0, _SS_BITMAP, 0, 0, 0
    Statictext #main, "The bitmap boy.bmp 54 x 100.", 90, 30, 150, 56

    Statictext #main.st1, "", 10, 140, 80, 120
    Stylebits #main.st1, _SS_BITMAP or _WS_BORDER, 0, 0, 0
    Statictext #main, "Thin Line Border, Bitmap expands to fit
definedstatictext area (80 x 120) when window resized.", _
        100, 120, 150, 140

    Statictext #main.st2, "", 10, 300, 80, 120
    Stylebits #main.st2, _SS_BITMAP or _WS_BORDER, 0, _WS_EX_STATICEDGE, 0
    Statictext #main, "Etched look with _WS_BORDER and _WS_EX_STATEDGE,
also resizes when window resized.", _
        100, 280, 150, 140

    Statictext #main.st3, "", 480, 70, 100, 150
    Stylebits #main.st3, _SS_BITMAP or _SS_CENTERIMAGE, 0,
_WS_EX_CLIENTEDGE, 0
```

```
    Statictext #main, "_SS_ CENTERIMAGE prevents resizing when window
resized.", 320, 90, 150, 140

    Statictext #main.st4, "", 480, 250, 100, 150
    Stylebits #main.st4, _SS_BITMAP or _SS_CENTERIMAGE, 0,
_WS_EX_DLGMODALFRAME, 0
    Statictext #main, "Borders can be raised, recessed or flat.", 320, 280,
150, 140

    Open "Stylebits for Images on Statictext" for Window as #main
    #main "Trapclose [closeDemo]"
    #main "Font Times_New_Roman 14 Bold"
    hStatic0 = hWnd(#main.st0)
    hStatic1 = hWnd(#main.st1)
    hStatic2 = hWnd(#main.st2)
    hStatic3 = hWnd(#main.st3)
    hStatic4 = hWnd(#main.st4)
    hImage = hBmp("boy")
    Call setImage hStatic0, hImage
    Call setImage hStatic1, hImage
    Call setImage hStatic2, hImage
    Call setImage hStatic3, hImage
    Call setImage hStatic4, hImage
    Wait

    Sub setImage hStatic, hImage
    CallDLL #user32, "SendMessageA", _
        hStatic as Ulong, _
        _STM_SETIMAGE as Long, _
        _IMAGE_BITMAP as Long, _
        hImage as Ulong, _
        result as Long
    End Sub

[closeDemo]
    Close #main
    Unloadbmp "boy"
    End
```

Dit zijn slechts enkele voorbeelden van wat u met stylebits en statictext kunt doen. Door te gaan experimenteren, kunt u veel meer.

Zoals bovenstaande listing laat zien brengt het vaak verwarring op met #main, omdat de handle eerder gebruikt wordt dan met het commando OPEN wordt aangemaakt. Ook het instellen van de breedte en hoogte van het venster wordt altijd als eerste gedaan.

Een lijst van Stylebits.
Bij MSDN Library - Static Styles is er een lijst aanwezig met alle dwStyles en dwExStyles.

Het NoMainWin commando.

Laten we eens kijken wanneer en waarom we NOMAINWIN wel of niet nodig hebben, en waar precies. Natuurlijk vraagt men vaak wat het commando inhoudt.

De opdracht NOMAINWIN is een compilerinstructie, niet een opdracht. Als Liberty BASIC dit commando in elke plaats in de code “ziet”, zal er geen mainwin in het programma aangemaakt worden. Het maakt niet uit dat als de code het commando heeft, dat die niet uitgevoerd zal worden. Liberty BASIC compileert de code in run-time en als het een NOMAINWIN ergens in de code vind zal er geen mainwin weergegeven worden.

Kijk eens in het volgende voorbeeld. Omdat variabele a gelijk is aan 2 zal de code binnen in de if… then routine niet uitgevoerd worden. Toch “ziet” Liberty BASIC het NOMAINWIN commando en onderdrukt de mainwin.

```
a = 2
if a = 4 then
    nomainwin
end if

notice "There is no mainwin!"
```

In het volgende voorbeeld wordt de code routine met de opdracht NOMAINWIN nooit uitgevoerd, maar Liberty BASIC “ziet” de opdracht NOMAINWIN en onderdrukt de mainwin.

```
notice "There is no mainwin!"

wait

[branchLabel]
nomainwin
wait
```

De MainWin sluiten.
Er is geen opdracht om de mainwin te sluiten. Er is ook geen opdracht om het te openen. Het mainwin venster, dat gebruikt wordt als het vroegere console venster, wordt standaard weergegeven, tenzij het programma een opdracht NOMAINWIN ergens in de code bevat. Het is mogelijk om de mainwin te sluiten met code. Het vereist wel het gebruik van een API aanroep. Lees verder over: De Mainwin manipuleren. Mainwin wordt automatisch gesloten als het programma met een END commando wordt beëindigd.

De Mainwin manipuleren.

Waarom zouden we de mainwin willen manipuleren?
Er zijn verscheidene nuttige commando’s die door de MainWin herkend worden zoals de MAINWIN opdracht waarmee u de grootte, vervolgens de aantal rijen en kolommen van de tekst kunt instellen, of de TITLEBAR opdracht waarmee u de tekst op de titelbalk kunt wijzigen. Sommige dingen kunnen alleen worden bereikt door API aanroepen. U kunt bijvoorbeeld de MainWin niet sluiten, gedurende het programma, nadat het is geopend.

De mainwin wordt voornamelijk gebruikt tijdens het ontwikkelen van programma's. U kunt altijd her en der variabelen laten printen in de mainwin als die open staat. Ook gebeurd het weleens dat een programma tijdens de ontwikkeling en testen vastloopt. Dergelijke programma's kunt u dan eenvoudig afsluiten door het mainwin venster te sluiten.

Als de MainWin niet met de opdracht NOMAINWIN onderdrukt is, zal aan het begin van het programma het eerste venster weergegeven worden. Als u de API aanroep naar GetActiveWindow aan het begin van uw code gebruikt, kunt u de handle van de MainWin ophalen en gebruiken om deze handle te manipuleren van het venster met API aanroepen. Dit werd voor het eerst besproken toen Liberty BASIC een 16-bits taal was. Op de website is de informatie geüpdatet voor Liberty BASIC 4, dat een 32-bits taal is. Een 32-bits taal gebruikt API aanroepen met een iets andere syntaxis dan een 16-bits taal.

De handle van de MainWin krijgen.
Zet deze code boven aan een programma voor het ophalen van de handle van de MainWin:

```
CallDLL #user32, "GetActiveWindow",_
hMainwin As uLong    'returns handle of mainwin
```

Het instellen van de grootte en de positie van de MainWin.
De MoveWindow API aanroep wordt ingesteld voor de locatie en de afmetingen van de MainWin. U zou dit het liefst willen gebruiken zodat het formaat van het venster wordt aangepast in pixels, in plaats van rijen en kolommen van tekst in de MAINWIN opdracht. U kunt ook het venster op een specifieke plek op het bureaublad vinden. Hier ziet u het instellen van de grootte en positie van de MainWin:

```
CallDLL #user32, "GetActiveWindow",_
hMainwin As uLong    'returns handle of MainWin

'can size window with MAINWIN command
'or can size AND locate with API:

x=10 : y=10 : w=230 : h = 150
CallDLL #user32, "MoveWindow",_
    hMainwin As uLong,_ 'handle
    x As Long,_         'x location
    y As Long,_         'y location
    w As Long,_         'width
    h As Long,_         'height
    1 As Boolean,_      'repaint, 1=true
    r As Boolean        'nonzero=success
```

Belangrijk! Zorg er altijd voor dat u, voordat u de opdrachten uit API aanroept, eerst de handle van de MainWin heeft. Zonder de handle kunt u namelijk niet de MainWin openen, de grootte instellen en op een locatie plaatsen.

De MainWin sluiten.
Er is geen Liberty BASIC opdracht om de MainWin te sluiten zonder het programma te beëindigen met een END commando. Als u de MainWin wilt sluiten, maar het programma niet wilt beëindigen, kunt u dat doen met de DestroyWindow API aanroep, zoals hieronder:

```
'close MainWin without ending program
Calldll #user32, "DestroyWindow",_
    hMainwin as uLong,_
```

```
    result as boolean   'nonzero=success
```

Een demo die van deze methoden gebruik maakt.
De onderstaande demo haalt de handle op van de MainWin wanneer het programma wordt gestart. Het stelt de grootte en positie in van de MainWin. De gebruiker krijgt een bericht. In dit geval is het gewoon een boodschap om op ENTER te drukken om door te gaan met het programma. Nadat de gebruiker op ENTER drukt, is de MainWin gesloten en een GUI programmavenster geopend.

```
CallDLL #user32, "GetActiveWindow",_
hMainwin As uLong    'returns handle of MainWin

'can size window with MAINWIN command
'or can size AND locate with API:

x=10 : y=10 : w=230 : h = 150
CallDLL #user32, "MoveWindow",_
    hMainwin As uLong,_ 'handle
    x As Long,_         'x location
    y As Long,_         'y location
    w As Long,_         'width
    h As Long,_         'height
    1 As Boolean,_      'repaint, 1=true
    r As Boolean        'nonzero=success


print "Hit ENTER to continue."
input aVar$

'close MainWin without ending program
Calldll #user32, "DestroyWindow",_
    hMainwin as uLong,_
    result as boolean   'nonzero=success

'the MainWin is now closed
'open a GUI window and continue program
button #1, "Close Me",[quit],UL,10,10
Open "A GUI Window" for window as #1
#1 "trapclose [quit]"
wait
[quit]  close #1:end
```

Zie nog meer voorbeelden op de website over het manipuleren van de MainWin. U kunt het vinden via de link op de pagina over de NoMainWin die u kunt vinden door te klikken op bovenstaand gegeven http link.

Kunt u het niet vinden, dan kunt u hier rechtstreeks naar de pagina over de MainWin:

http://babek.info/libertybasicfiles/lbnews/nl131/api.htm

Bekijk ook eens de leuke mogelijkheden rechts in de lijst.

Marco Kurvers

Intermezzo – een calculator maken.

Gordon Rahman gaf een huiswerkopgave aan de cursisten. De huiswerkopgave uit de Liberty BASIC workshop zal hij hieronder wat uitwerken. Voor u, veel plezier met deze intermezzo.

De opgave om een calculator te maken.

Schrijf een calculator programma in Liberty BASIC waarbij je

gebruik mag maken van bovenstaande of volgende plaatjes en een wave bestand.

Het valt op dat ik slechts drie toetsen en een achtergrond heb geleverd. De cursisten moesten zelf de tekens op de toetsen plaatsen. Dat moest met enig beleid gedaan worden, anders krijg je dergelijke toetsen als deze 

. De achtergrond van het teken 4 op de toets is wit en dat staat lelijk op de mooie toets. Gelukkig heeft Microsoft Windows daar wat op gevonden. Je kunt de achtergrond van de tekens doorzichtig maken. Daar heb je de SetBKmode functie voor nodig. Je moet ook het nummer van de DC (Device Control) kennen. Deze gegevens worden op de MSDN site goed besproken. Gevorderde Liberty BASIC programmeurs hebben dat natuurlijk reeds uitgezocht.

Ik gaf een testbestand aan de cursisten mee. Niet elke cursist is een gevorderde. Met het testbestand konden de toetsen naar eigen smaak opgemaakt worden. Hier volgt de uiteindelijke listing van een cursist. Ik zal de listing bespreken.

```
nomainwin

    WindowWidth = 267
    WindowHeight = 420
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)

    global R$

    loadbmp "calcBG", "calculatortest1.bmp"

    bmpbutton #main.button7,"CijferKnop7.bmp",cijferClick, UL, 020, 130
    bmpbutton #main.button8,"CijferKnop8.bmp",cijferClick, UL, 080, 130
    bmpbutton #main.button9,"CijferKnop9.bmp",cijferClick, UL, 140, 130

    bmpbutton #main.button4,"CijferKnop4.bmp",cijferClick, UL, 020, 180
    bmpbutton #main.button5,"CijferKnop5.bmp",cijferClick, UL, 080, 180
    bmpbutton #main.button6,"CijferKnop6.bmp",cijferClick, UL, 140, 180
```

```
    bmpbutton #main.button1,"CijferKnop1.bmp",cijferClick, UL, 020, 230
    bmpbutton #main.button2,"CijferKnop2.bmp",cijferClick, UL, 080, 230
    bmpbutton #main.button3,"CijferKnop3.bmp",cijferClick, UL, 140, 230

    bmpbutton #main.button0,"CijferKnop0.bmp",cijferClick, UL, 020, 280
    bmpbutton #main.buttonP,"CijferPunt1.bmp",cijferClick, UL, 080, 280

    bmpbutton #main.buttonD,"PowerKnop3.bmp",powerClick, UL, 200, 130
    bmpbutton #main.buttonV,"PowerKnop2.bmp",powerClick, UL, 200, 180
    bmpbutton #main.buttonA,"PowerKnop0.bmp",powerClick, UL, 200, 230
    bmpbutton #main.buttonO,"PowerKnop1.bmp",powerClick, UL, 200, 280

    bmpbutton #main.buttonI,"IsGelijkKnop1.bmp",[buttonIClick], UL, 145,
330

    bmpbutton #main.buttonC,"PowerKnop5.bmp",[buttonCClick], UL, 020, 330
    bmpbutton #main.buttonB,"PowerKnop6.bmp",[buttonBClick], UL, 140, 280
    'bmpbutton #main.button20,"PowerKnop4.bmp",[buttonPMClick], UL, 95, 80

    TextboxColor$ = "green"
    textbox #main.textbox21,  20,  40, 230,  40
    stylebits #main.textbox21, _ES_RIGHT,0,0,0

    graphicbox #main.g 0,0,267,420
    stylebits #main.g, 0,_WS_Border,0,0

    open "untitled" for window_popup as #main
    #main.g "down; drawbmp calcBG 0 0"
    #main.textbox21 "!font 15"
    #main "trapclose [quit.main]"
    #main.g "flush"
    wait


sub cijferClick handle$
        playwave "tick0.wav", async
        #main.textbox21 "!contents? Number$"
        if right$(handle$,1)= "P" then
          #main.textbox21 Number$ + "."
        else
          #main.textbox21 Number$ + right$(handle$,1)
        end if
end sub


sub powerClick handle$
    #main.textbox21 "!contents? lastR$"
    #main.textbox21 ""
    R$ = R$ + lastR$
     select case right$(handle$,1)
        case "V"
          R$ = R$ + " * "
        case "O"
          R$ = R$ + " + "
        case "A"
```

```
          R$ = R$ + " - "
        case "D"
          R$ =  R$ + " / "
    end select
end sub


[buttonCClick]
print "c"
    #main.textbox21 ""
    R$ = "" : a$ = ""
    #main.textbox21 a$
    wait

[buttonBClick]

confirm "Quit?";yn$
if yn$ = "yes" then [quit.main]
Notice "Calculator by Cursist 2011"  'Insert your Back Space
wait

[buttonIClick]
    playwave "Bounce.wav"
    #main.textbox21 "!contents? varName$"
    TR$ = R$ + varName$
print TR$
    a$  = eval$(TR$)
print a$
    #main.textbox21 ""
    #main.textbox21 a$
    R$ = ""
    wait


[buttonPMClick]
print "pm"    'Insert your plus or minus
    wait

[quit.main] 'End the program
    unloadbmp "calcBG"
    close #main
    end
```

Hier zie je het resultaat van het bovenstaand programma van een Liberty BASIC cursist uit Amstelveen.

De uitwerking en uitleg van de listing.
Het commando NOMAINWIN zorgt ervoor dat het console venster (mainwindow), dat Liberty BASIC standaard opent, nu niet getoond wordt.
De eerste vier opdrachten geven de positie van het venster (onze calculator) op het monitorscherm aan. Onderstaande regel leg ik uit:

```
UpperLeftX=int((DisplayWidth-WindowWidth)/2)
```

Hiermee positioneren we ons venster precies in het midden van ons monitorscherm (display of misschien begrijpelijker, het bureaublad).
UpperLeftX is de X coördinaat van de linkerbovenhoek van ons venster.

Daarna maken we de variabele R$ globaal. Globale variabelen zijn overal in het programma te gebruiken. Variabelen die in subroutines gebruikt worden zijn lokaal, dat wil zeggen dat de waarde van dergelijke variabelen alleen binnen de sub gelden. De waarde van R$ kan dus ook buiten de subs gebruikt worden, daarom is deze variabele hier globaal gemaakt met Global.

Daarna wordt het achtergrondplaatje in het geheugen geladen.

```
loadbmp "calcBG", "calculatortest1.bmp"
```

Daarna volgen negen “bmpbuttons”. Hier haal ik een willekeurig voorbeeld eruit.

```
bmpbutton #main.button3,"CijferKnop3.bmp",cijferClick, UL, 140, 230
```

#main.button3 is de handle (naam van de “control”). Het deel #main komt overeen met de handle-naam van het venster waarin de button staat. De naam button3 is de eigen naam van de button (bmpbutton) omdat het een plaatje is met button eigenschappen.
Let op dat de handle-namen gelijk zijn en dat alleen het laatste karakter (in dit geval 3) per button verschilt. Dat komt straks goed van pas.

De naam cijferClick is de naam van de subroutine waar het programma naartoe springt als op de bmpbutton wordt gedrukt. Het drukken op een button (muis beweging enzovoort) geeft een EVENT. Dat registreert Windows en die geeft dat EVENT door aan Liberty BASIC.
Het programma vervolgt bij:

```
sub cijferClick handle$
     playwave "tick0.wav", async
     #main.textbox21 "!contens? Number$"
     if right$(handle$,1)="P" then
          #main.textbox21 Number$ + "."
     else
          #main.textbox21 Number$ + right$(handle$,1)
     end if
end sub
```

De naam van de handle komt in de variabele handle$ te staan.
Dan wordt het geluidsfragment (wave-file) tick0.wav gespeeld (een click klank).
Daarna wordt de inhoud van textbox21 uitgelezen en die inhoud wordt in de variabele Number$ geplaatst.

Als het laatste karakter van handle$ een P is dan wordt in de tekstbox Number$ + “.” geplaatst. Zo niet, dan wordt in textbox21 Number$ + right$(handle$,1) geplaatst. Het aanwezige getal plus het laatst ingevoerde cijfer.

Terug naar de listing. Na de bmpbuttons staan de knoppen voor rekenkundige bewerkingen en daarna de knoppen voor resultaat, clear en aan/uit.

De rekenkundige knoppen doen het programma bij een door hun veroorzaakte event springen naar de sub:

```
sub powerClick handle$
     #main.textbox21 "!contens? lastR$"
     #main.textbox21 ""
     R$ = R$ + lastR$
     select case right$(handle$,1)
          case "V"
               R$ = R$ + " * "
          case "O"
               R$ = R$ + " + "
          case "A"
               R$ = R$ + " - "
          case "D"
               R$ = R$ + " / "
     end select
end sub
```

Bovenstaande sub is niet spannend. Dat kun je nu wel volgen.

De belangrijkste knop is de knop met het = teken.

```
bmpbutton #main.buttonI,"IsGelijkKnop1.bmp",[buttonIClick], UL, 145, 330
```

Deze knop veroorzaakt een EVENT naar een “label” (engels voor merkteken). Het label staat tussen rechte haakjes [buttonIClick].

```
[buttonIClick]
     playwave "Bounce.wav"
     #main.textbox21 "!contens? varName$"
     TR$ = TR$ + varName$
     a$ = eval$(TR$)
     #main.textbox21 ""
     #main.textbox21 a$
     R$ = ""
     wait
```

Eerst wordt er een wav bestand (Bounce.wav) afgespeeld.
Dan wordt de inhoud van textbox21 gelezen en in variabele varName$ geplaatst.
De variabele R$, uit de vorige sub, bestond uit een getal en een operator (+, -, \ of *), die nu aan de variabele uit de “label” subroutine gekoppeld wordt. TR$ wordt bijvoorbeeld “123 + 456” of “123 / 456” enzovoort. Deze TR$ stringvariabele wordt nu met een speciale Liberty BASIC functie geëvalueerd, Eval$(TR$).
Daarna wordt de uitkomst in het venster textbox21 geplaatst.

Nog even terug naar de listing. De tekstbox en het grafische venster zijn van speciale zogenaamde stylebits voorzien. De stylebits van de tekstbox zorgt ervoor dat de ingegeven tekst aan de rechterkant in de tekstbox wordt geplaatst tijdens het invoeren.
De stylebits van het grafische venster zorgt ervoor dat er geen rand om het venster verschijnt.

Tip! Weet u niet precies wat stylebits inhoudt? Lees dan eens het vorige onderwerp, Liberty BASIC verkennen. Daar wordt meer besproken over stylebits.

Okay. Je hebt geen Liberty BASIC maar wel Just BASIC, wat nu?
Just BASIC heeft drie nadelen:

- Just BASIC kent geen stylebits.
- Just BASIC kent de EVAL functie niet.
- Just BASIC kan de DLL's, voor het tekenen op de knoppen, niet openen.

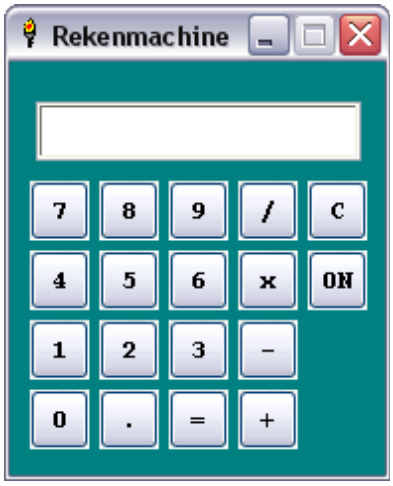
Hier volgt een voorbeeld van dezelfde opgave, maar nu in Just BASIC geprogrammeerd.

```
'** rekenmachine

     NOMAINWIN
     WindowWidth = 195 : WindowHeight = 240
     UpperLeftX = INT((DisplayWidth-WindowWidth)/2)
     UpperLeftY = INT((DisplayHeight-WindowHeight)/2)

     [Display]
          reken$="":getal=0:vgetal=0:getal$=""
          textbox   #m.textbox1, 13, 20, 164, 30
          button    #m.b1, "9", [k9], UL, 80, 60, 30, 30
          button    #m.b2, "8", [k8], UL, 45, 60, 30, 30
          button    #m.b3, "7", [k7], UL, 10, 60, 30, 30
          button    #m.b4, "6", [k6], UL, 80, 95, 30, 30
          button    #m.b5, "5", [k5], UL, 45, 95, 30, 30
          button    #m.b6, "4", [k4], UL, 10, 95, 30, 30
          button    #m.b7, "3", [k3], UL, 80, 130, 30, 30
          button    #m.b8, "2", [k2], UL, 45, 130, 30, 30
          button    #m.b9, "1", [k1], UL, 10, 130, 30, 30
          button    #m.b10, "0", [k0], UL, 10, 165, 30, 30
          button    #m.b11, ".", [kk], UL, 45, 165, 30, 30
          button    #m.b12, "x", [kx], UL, 115, 95, 30, 30
          button    #m.b13, "/", [kd], UL, 115, 60, 30, 30
          button    #m.b14, "+", [kp], UL, 115, 165, 30, 30
          button    #m.b15, "-", [km], UL, 115, 130, 30, 30
          button    #m.b16, "=", [ks], UL, 80, 165, 30, 30
          button    #m.b17, "C", [kc], UL, 150, 60, 30, 30
          button    #m.b18, "ON", [ko], UL, 150, 95, 30, 30

          BackgroundColor$ = "darkcyan"
          Open "Rekenmachine" for window_nf as #m
               #m "trapclose [einde]"
               #m "font courier new 10 bold"
          Wait

     [einde]
          close #m
          END
```

```
[ko]
     if aan = 0 then
          #m.textbox1 "!font courier new 10 bold"
          print #m.textbox1,space$(14)+"0"
          aan=1:reken$="":getal$="":getal=0:vgetal=0
     else
          print #m.textbox1,"":aan=0
     end if
     wait

[k0]
     if aan=1 then
          reken$=reken$+"0"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k1]
     if aan=1 then
          reken$=reken$+"1"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k2]
     if aan=1 then
          reken$=reken$+"2"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k3]
     if aan=1 then
          reken$=reken$+"3"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k4]
     if aan=1 then
          reken$=reken$+"4"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k5]
     if aan=1 then
          reken$=reken$+"5"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k6]
     if aan=1 then
          reken$=reken$+"6"
```

```
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k7]
     if aan=1 then
          reken$=reken$+"7"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k8]
     if aan=1 then
          reken$=reken$+"8"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[k9]
     if aan=1 then
          reken$=reken$+"9"
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[kk]
     if aan=1 then
          reken$=reken$+"."
          print #m.textbox1,space$(15-len(reken$))+reken$
     end if
     wait

[kx]
     if aan=1 then
          gosub [bereken]:maal=1
          print #m.textbox1,space$(15-len(getal$))+getal$
          reken$=""
     end if
     wait

[kd]
     if aan=1 then
          gosub [bereken]:deel=1
          print #m.textbox1,space$(15-len(getal$))+getal$
          reken$=""
     end if
     wait

[kp]
     if aan=1 then
          gosub [bereken]:plus=1
          print #m.textbox1,space$(15-len(getal$))+getal$
          reken$=""
     end if
     wait
```

```
[km]
     if aan=1 then
          gosub [bereken]:min=1
          print #m.textbox1,space$(15-len(getal$))+getal$
          reken$=""
     end if
     wait

[ks]
     if aan=1 then
          gosub [bereken]
          print #m.textbox1,space$(15-len(getal$))+getal$
          reken$=""
     end if
     wait

[bereken]
     if reken$<>"" then
          getal=val(reken$)
          if vgetal<>0 then
               if maal=1 then
                    getal=getal*vgetal:maal=0
               end if
               if deel=1 then
                    getal=vgetal/getal:deel=0
               end if
               if plus=1 then
                    getal=getal+vgetal:plus=0
               end if
               if min=1 then
                    getal=vgetal-getal:min=0
               end if
          end if
          vgetal=getal
          getal$=str$(getal)
     end if
     return
```



Dit is het resultaat van de uitvoering van de Just BASIC listing. De cursist heeft geen gebruik gemaakt van de knoppen (bmp plaatjes) die bij de opgave horen.
Bestudeer de listing zelf.

Alle listings en plaatjes staan op het Liberty BASIC forum. Om jullie kennis te laten maken met Liberty BASIC en de workshop, verklap ik dat ook de oplossingen van de andere cursisten op het forum te vinden zijn, zie hieronder de link.

http://www.libertybasic.nl/viewtopic.php?f=15&t=542

Gordon Rahman

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 in buurthuis Bronveld in Barneveld van 19:00 uur tot 21:00 uur. 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 19:00 uur tot 21:00 uur.
Meer informatie? Kijk op 'http://www.buurthuisbronveld.nl' of neem contact op met mij.

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)	j.a.van.hezik@kader.hcc.nl
Visual Basic .NET	Marco Kurvers	do. t/m zo.	19-22	214131 (0342) 424452	m.a.kurvers@hccnet.nl
Basic algemeen, zoals VBA Office Web Design, met XHTML en CSS	Marco Kurvers	do. t/m zo.	19-22	(0342) 424452	m.a.kurvers@hccnet.nl