Nieuwsbrief

17de jaargang december 2010 **Nummer 4**

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Geen	

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

De nieuwsbrief heeft een nieuw jasje gekregen. Lees meer op BASIC nieuws.

VBA is even op vakantie. In deze nieuwsbrief ga ik het hebben over de Excel functies. Er zijn er ontzettend veel van en ook hele aparte waar ik een paar van uit haal.

In nieuwsbrief nummer 3 heeft u al wat kunnen lezen over 'Zelf gegevenstypen maken'. Er bestaan nog meer soorten gegevenstypen, maar deze horen echter niet tot Basic zelf. Het zijn gegevenstypen die onderdeel zijn van het Microsoft .NET Framework, die ik ook wel de jas van Basic noem.

Al eens geprobeerd een programmacode van een andere BASIC versie of van een andere programmeertaal te converteren naar de BASIC versie waar u mee werkt? Er zijn veel converteerprogramma's en wizards. Helaas werken ze niet allemaal honderd procent correct. Lees meer op BASIC nieuws.

Er bestaat een programma, Crimson Editor, die zeer handig is om websites te programmeren. In vorige nieuwsbrieven heb ik er al wat over geschreven. Webdesigners zijn er genoeg, maar ze veroorzaken ook problemen die we in code niet tegenkomen. Lees meer daarover waarom niet.

Marco Kurvers

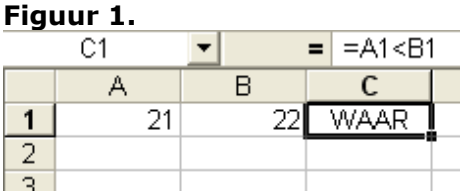
BASIC cursus: VBA met Excel (4).

In het Redactionele gedeelte heb ik verteld dat VBA op vakantie is. We gaan eens kijken wat Excel zelf voor ons te bieden heeft. In de nieuwsbrieven van het jaar 2006 heb ik geschreven dat het minder zou worden voor VBA. En inderdaad, het gebruik van expressies, condities, verwijzingen en zelfs functies is in Excel sterk toegenomen.

Het werkboek beheren.

Waarden kunnen we simpel in de cellen invoeren. Het kan ook gebeuren dat er verschillen bepaald moeten worden, bijvoorbeeld om twee totaalwaarden te vergelijken. Als voorbeeld ziet u Figuur 1. Ik heb twee getallen ingevoerd en in de derde cel heb ik Excel laten bepalen of cel A1 kleiner is dan cel B1 door in cel C1 een conditie te schrijven.

Figuur 1.

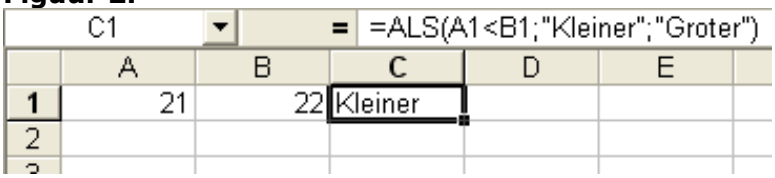


U kunt in elke cel alle BASIC operatoren gebruiken. Zoals u hier ziet zal de waarde WAAR in cel C1 verschijnen.

Zodra u de waarde in cel A1 hoger zet dan de waarde in B1, zal het resultaat in C1 direct veranderen in ONWAAR.

Dat de waarde WAAR of ONWAAR verschijnt is misschien niet wat u wilt. Misschien wilt u liever uw eigen waarden of meldingen weergeven. Om dit te doen moet u gebruik maken van beslissingen die een DAN en een ANDERS heeft. Excel verwacht dat u achter DAN en ANDERS een constante of een expressie opgeeft. De constante kan ook tekst zijn zodat u wat anders kunt weergeven dan alleen maar WAAR of ONWAAR, zie Figuur 2.

Figuur 2.



Figuur 2 laat een functie zien met twee tekstconstanten. Eén van de twee wordt weergegeven als aan de voorwaarde wel of niet wordt voldaan.

Onthoud dat u in een ALS functie de tekstconstanten tussen dubbele aanhalingstekens moet plaatsen. Direct invoeren zal ervoor zorgen dat ook de aanhalingstekens worden weergegeven. Het enkele aanhalingsteken werkt niet in de ALS functie.

U kunt in een ALS functie van alles in de expressies gebruiken. Een ALS functie nesten is zelfs mogelijk. De invoerbalk kan daardoor wel ingewikkeld worden. Het kan dan geen kwaad om een VBA functie te schrijven.

Verwijzingen.

We hoeven niet altijd het resultaat te laten bepalen door de ALS functie en zeker niet als u meerdere ALS functies hebt genest. Hebt u geen zin om een VBA functie te schrijven dan biedt Excel nog een andere mogelijkheid: een expressie laten verwijzen naar een andere cel en die de rest uit laten voeren. U moet dan de eerste ALS functie als een *binnenste* ALS functie zien. Waarom? Kijk eens naar onderstaand voorbeeld.

`=ALS(ALS(`*conditie1*`;`*waar1*`;`*onwaar1*`);`*waar2*`;`*onwaar2*`)`

Net als in normale berekeningen geldt: wat binnen de haakjes staat wordt als eerste uitgevoerd. De binnenste ALS functie zal het resultaat van de expressie 'waar1' of 'onwaar1' daarom geven aan de buitenste ALS functie die het resultaat van de expressie 'waar2' of 'onwaar2' zal bepalen.
Figuur 3 geeft een voorbeeld hoe we de voorgaande ALS functie in de nieuwe ALS functie kunnen nesten.

Figuur 3.

A6 = =ALS(ALS(A1<B1;E2;E3);"Nee";E4)

	A	B	C	D	E	F
1	21	22	Kleiner			
2	100				100	
3					200	
4					Juist	
5						
6	Nee					
7						

Omdat de binnenste ALS functie een 'waar' resultaat geeft, zal het resultaat van de buitenste ALS functie de tekst "Nee" zijn.

Om de twee ALS functies uit elkaar te houden en toch samen te laten werken, hadden we de formule ook zo kunnen schrijven: `=ALS(C1="Kleiner";"Nee";E4)`

Zouden we schrijven: `=ALS(C1="Groter";"Nee";E4)` dan zal de tekst "Juist" worden weergeven die in cel E4 bepaald wordt door een andere ALS functie. We hoeven dus niet telkens de tekst "Juist" in de cellen te herhalen. Door te verwijzen naar de cellen, kunnen de formules 'ingekapseld' uitgevoerd worden.

Let op! U moet in de conditie een waarde opgeven achter een cel. Het is niet toegestaan om alleen de cel-naam op te geven.

R1K1 verwijzingen.

Met een verwijzing in de vorm R1K1 wordt de locatie van een cel aangeduid met een 'R', gevolgd door een rijnummer en een 'K', gevolgd door een kolomnummer. Zo is de absolute celverwijzing R1K1 identiek aan de absolute celverwijzing A1 in de verwijzingsstijl A1. Als A1 de actieve cel is, verwijst de relatieve cel-verwijzing R[1]K[1] naar de cel die zich één rij onder en één rij rechts van de cel bevindt, in dit geval dus B2.

Hier volgen enkele voorbeelden van verwijzingen in de vorm van R1K1.

Verwijzing	Betekenis
R[-2]K	Een relatieve verwijzing naar de cel die zich twee rijen hoger en in dezelfde kolom bevindt.
R[2]K[2]	Een relatieve verwijzing naar de cel die zich twee rijen lager en twee kolommen naar rechts bevindt.
R2K2	Een absolute verwijzing naar de cel die zich in de tweede rij en in de tweede kolom bevindt.
R[-1]	Een relatieve verwijzing naar de hele rij die zich boven de actieve cel bevindt.
R	Een absolute verwijzing naar de huidige rij.

Wanneer u relatieve celverwijzingen gebruikt, zullen ze automatisch worden aangepast zodra u ze ver-plaatst. Bij absolute celverwijzingen niet.

Labels en namen in formules.

In werkbladen ziet u labels boven de kolommen staan en links ervan de rijen. Dit zijn de namen die de cel-len beschrijven. Deze namen kunt u in formules gebruiken om naar de gegevens te verwijzen, zoals eerder gezien. U kunt zelf ook cellen, bereiken, formules of constanten een naam geven in plaats van de labels te gebruiken.

Let op! Microsoft Excel herkent standaard geen labels in formules. Als u labels in formules wilt gebruiken moet u deze instellen bij Opties – tabblad Berekenen.

Na de optie te hebben gekozen, kunt u zelf namen gebruiken in formules. Onderstaand voorbeeld laat een som zien.

Test
10
20
30
60

In de cel waar 60 staat zal een som in de invoerbalk er als volgt uitzien:

```
=SOM(Test)
```

Door elke kolom, die waarden bevat, een naam te geven, kunt u sommen, beslissingen enzovoort, eenvoudig uitvoeren door alleen maar de kolomnaam op te geven.

Het uitroepteken '!'.

Met het uitroepteken hoeft u niet per se op één werkblad te blijven werken. U kunt ook verwijzen naar andere werkbladen, zoals onderstaand voorbeeld laat zien:

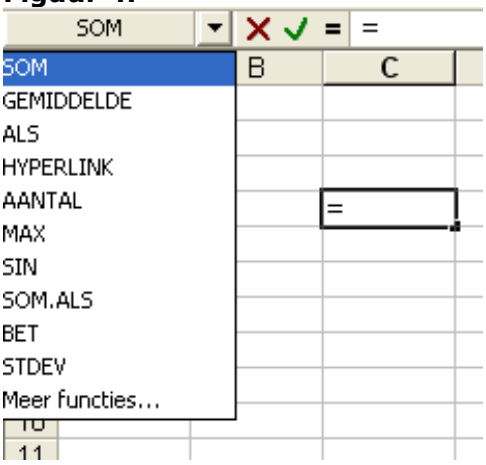
```
=Blad2!A1
```

Als u die regel typt in Blad1 cel A1 en u typt in Blad2 cel A1 de som =10*50 dan zal het resultaat van de som niet alleen in Blad2 komen te staan, maar ook in Blad1.

Meer Excel functies.

In Excel is er niet alleen de ALS functie, er zijn nog meer functies. Die kunt u vinden door in de invoerbalk het teken '=' te typen, zie Figuur 4.

Figuur 4.



Links ervan ziet u SOM verschijnen. Met het driehoekje rechts van SOM opent u de lijst met functies.

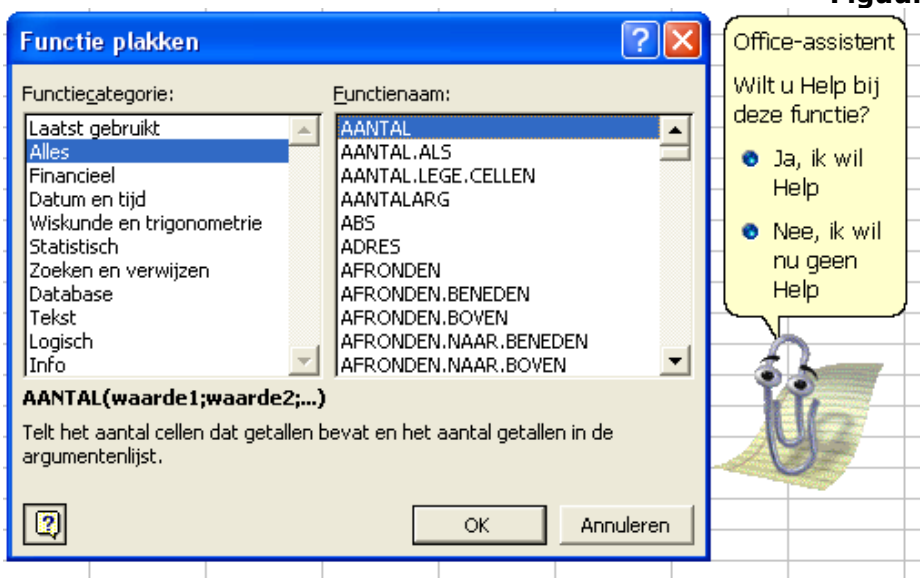
Er is een functie die goed naar voren komt. Een aparte functie, omdat die uit twee functies bestaat gescheiden met een punt, de SOM.ALS functie. Daar kom ik straks op terug.

Onderaan ziet u staan 'Meer functies...' . Als u daar op klikt, verschijnt er een dialoogformulier. Zie Figuur 5.

Figuur 5.

De Office-assistent vraagt meteen of u hulp wilt hebben bij het formulier. Alles wat u dan doet zal met extra uitleg door de Office-assistent gegeven worden.

Links in de lijst ziet u de functiecategorieën. Met al deze functies hebt u genoeg gereedschap om prima werkbladen te maken.



Aan de rechterkant ziet u de functies op alfabetische volgorde per categorie. Weer kunt u vreemde functies vinden die uit meerdere functies bestaan en gescheiden zijn door één of meerdere punten.

U kunt heel lang blijven experimenteren met deze categorieën. Bekijk de functies maar eens door de beschrijving van een functie te lezen. Als voorbeeld wordt er uitgelegd dat functie AANTAL de aantal cellen telt dat getallen bevat en ook de aantal getallen in de argumentenlijst. Bekijk ook eens wat de beschrijving zegt als u klikt op de functie AANTAL.LEGE.CELLEN.
Voor het automatiseren van de werkbladen hebt u toch VBA nodig. Voor logische en wiskundige bewerkingen zijn de functiecategorieën beschikbaar. U mag de functies ook opnemen in een macro. VBA herkent helaas niet de Excel functies. Ze worden ook als strings toegekend aan de eigenschap FormulaRC of FormulaR1C1.

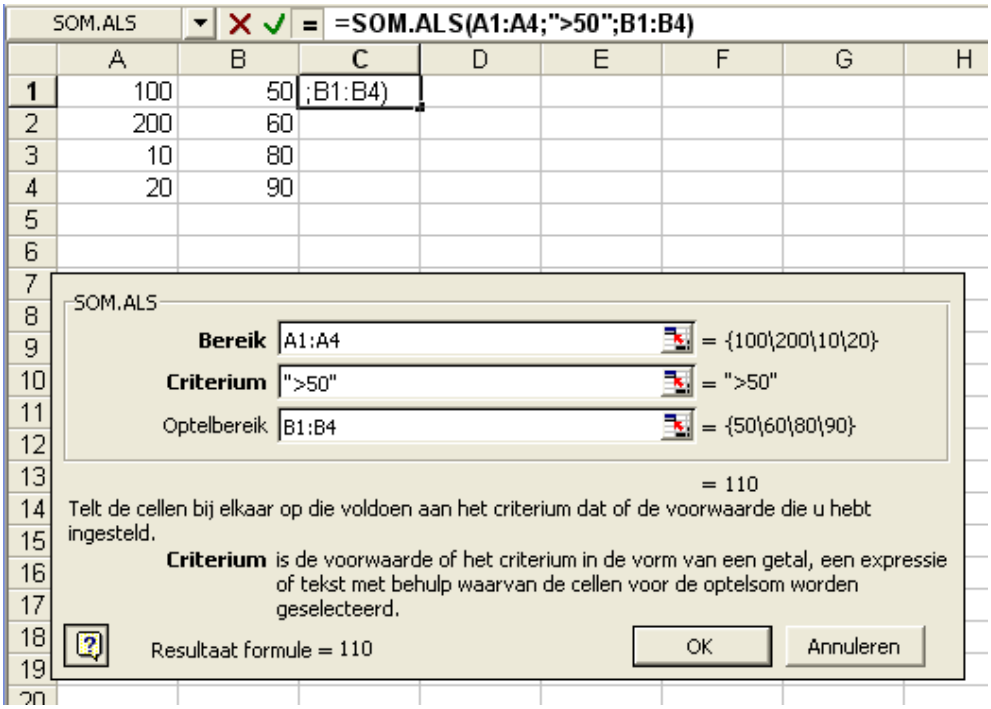
Tip! In macrovoorbeelden die in deze lessen staan kunt u de genoemde eigenschappen tegenkomen. VBA gebruikt altijd FormulaRC of FormulaR1C1 als die een formule in een cel tegenkomt. De opgenomen macro gebruikt geen Cells of Range om de formule met de gekozen functie toe te kennen.

Meerdere functies gescheiden met één of meerdere punten.

U hebt ze al gezien, de functies met punten ertussen. Maar wat houdt de punt in? Om dat uit te zoeken zal ik een paar van die functies eruit halen, onder andere de functie SOM.ALS. Die functie werkt zeer handig bij het bepalen en uitrekenen van verschillende kolommen en ook gebieden. De functie heeft de volgende syntaxis:

`=SOM.ALS(`**`bereik;criterium;optelbereik`**`)`

Als u in de invoerbalk de functie intoetst, zal aan de linkerkant de functienaam verschijnen. Klik op de functie en een dialoogformulier zal verschijnen, zie Figuur 6.



Figuur 6.

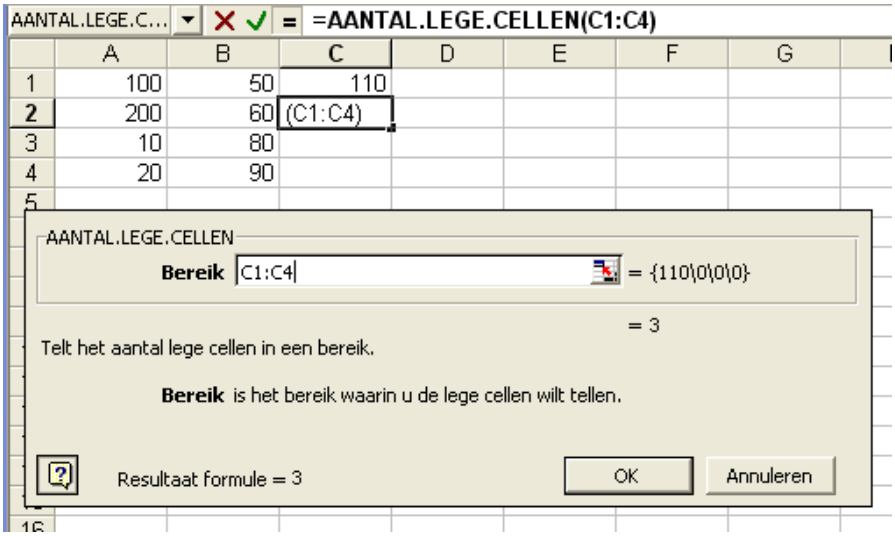
In het opgegeven bereik zal de ALS functie het criterium bepalen. Als de waarden in het bereik groter zijn dan 50, zullen de waarden in het optelbereik opgeteld worden. Nog lastig voor u om het te begrijpen? Bekijk dan eens onderstaand tabel waar ik elke rij stap voor stap laat zien.

Rij met waarde	Criterium	Resultaat
A1: 100	Groter dan 50?	Ja, tel het op. Nu eerste getal B1: 50
A2: 200	Groter dan 50?	Ja, tel het op. B2: 60 plus B1: 50 = 110
A3: 10	Groter dan 50?	Nee, tel het niet op.
A4: 20	Groter dan 50?	Nee, tel het niet op.

Dus zal het resultaat van de formule waarde 110 zijn zoals Figuur 6 laat zien.

Het is mogelijk om het optelbereik met andere rijen en/of kolommen op te geven. Een uitroepteken voor gebruik van een ander werkblad is ook toegestaan.

Een tweede functie die ik eruit haal is de AANTAL.LEGE.CELLEN functie. Toets de functie in en klik erop. Een dialoogformulier verschijnt, zie Figuur 7.



Figuur 7.

In het opgegeven bereik zal de functie bepalen hoeveel cellen er leeg zijn. Ook cel C2 wordt gezien als een lege cel, ook al staat de formule er in. Dat komt omdat de formule pas compleet is als op de knop OK wordt geklikt, want dan zal het resultaat in cel C2 worden weergegeven.

Onthoud dat waarde 0 niet als een lege cel wordt gezien. Maak eens een nieuwe formule met de functie, maar nu in cel D1. Zorg ervoor dat u in cel C3 de waarde 0 hebt ingetoetst en verwijder de formule die nog in cel C2 staat. Het bereik blijft gewoon C1:C4. Als het goed is zal het resultaat nu 2 zijn in plaats van 3.

In de volgende nieuwsbrief gaan we al deze mogelijkheden bekijken in VBA. Wat kunnen we in VBA wel en niet uitvoeren? We gaan dan ook eens kijken hoe we dialoogformulieren kunnen weergeven en hoe we nieuwe formulieren zelf ontwerpen.

Marco Kurvers

Gegevenstechnieken en opslagstructuren.

In de vorige nieuwsbrief heb ik laten zien hoe u gegevenstypen kunt maken met het TYPE statement voor recordtypen en het ENUM statement voor enumeratietypen. Er bestaan nog veel meer soorten gegevenstypen. Niet alleen nieuwe soorten die u zelf maakt, maar gegevenstypen voor databases, internet browsers, netwerken en zelfs complete systeemservers. De vraag is echter... hoe komen we daar aan en hoe gebruiken we die? Een andere vraag zal ook wel gelijk gesteld worden, namelijk of Basic ons toegang kan verlenen zodat we die techniek kunnen gebruiken in onze programma's.
Het is mogelijk, maar de ene Basic versie houdt de toegang met de objecten, die daarvoor nodig zijn, meer beperkt dan de andere Basic versie. Visual Basic 6.0 zorgt ervoor dat we API subroutines uit DLL library's kunnen aanroepen. Voordat we dat mogen doen moeten we wel eerst de subroutines declareren. Een ander probleem met beperkte toegang is de databasebeheer. De library's DAO en ADO bieden goede opslagstructuren om een programma met een werkende database te kunnen maken. Het nadeel is echter dat we altijd een databaseprogramma nodig hebben, want de library's hebben een connectie om een databasebestand te koppelen. Zonder die connectie kan de library niet werken. Besturingselementen hebben ook die connectie-eigenschappen. Gelukkig zijn ze niet meer nodig dankzij een nieuwe techniek die niet alleen voor goede gegevenstechnieken en opslagstructuren zorgt, maar ook zorgt voor een goede programmabesturing; het Microsoft .NET Framework.

XML en het DataSet object.

XML (Extensible Markup Language) kon al eerder in Basic en ook in andere programmeertalen worden gebruikt. Om de code voor de besturing te programmeren was echter niet gemakkelijk. De gegevens die tussen de XML tags staan, werden in tekstbestanden opgeslagen en moesten met API subroutines ingelezen en weggeschreven worden. De gegevens moesten op de juiste veldvolgorde geschreven worden. Een nieuw veld wegschrijven betekent dus meteen de waarde die erbij hoort. Het XML tabel moet eerst worden gemaakt voordat de rest kan worden gedaan, maar dat kwam dan wel direct erachteraan.

Onderstaand voorbeeld laat zien hoe XML werkt:

```
<?xml version="1.0" encoding="utf-8"?>
<Ledenlijst name="BASIC IG" xml:lang="en">
    <Persoon>
        <KeyID>LP00</KeyID>
        <Naam>Marco Kurvers</Naam>
        <Adres>Schaapsveld 46</Adres>
```

```
        <Postcode>3773ZJ</Postcode>
        <Plaats>Barneveld</Plaats>
        <Telefoon>0342424452</Telefoon>
        <EMail>m.a.kurvers@hccnet.nl</EMail>
    </Persoon>
    <Persoon>
        <KeyID>LP01</KeyID>
        <Naam>Karel Groen</Naam>
        <Adres>Lesliestraat 10</Adres>
        <Postcode>1289MZ</Postcode>
        <Plaats>Beterland</Plaats>
        <Telefoon>1001676888</Telefoon>
        <EMail></EMail>
    </Persoon>
</Ledenlijst>
```

Een ander XML voorbeeld is een muziekplaylist, die er als volgt uit zou kunnen zien:

```
<?xml version="1.0" encoding="utf-8"?>
<playlist name="mylist" xml:lang="en">
 <song>
  <title>Little Fluffy Clouds</title>
  <artist>the Orb</artist>
 </song>
 <song>
  <title>Goodbye mother Earth</title>
  <artist>Underworld</artist>
 </song>
</playlist>
```

Wanneer u tientallen of misschien wel honderden personen toe moet voegen bent u in het tekstbestand lang bezig en het ADO library heeft geen gereedschap om u een XML database te laten beheren.

Het DataSet object.

Vanaf Visual Basic .NET 2003 is er een extra visueel scherm in de IDE waarmee u een eigen XML database-schema kunt maken. Als het schema in het programma gebruikt wordt zullen de bestanden er ongeveer uit komen te zien zoals de bovenste twee voorbeelden er uit zien, inclusief de gegevens die aangemaakt en bewerkt worden. Dit werkt veel sneller en dankzij de IDE kunt u het schema als een boomstructuur bekijken en kunt u eventueel wijzigingen aanbrengen.

Er is ook een andere manier om XML databases te maken en wel intern, zonder eerst een databaseschema te maken in de IDE. Dankzij het .NET Framework zijn de API aanroepingen verleden tijd en hebben we nu toegang tot gegevenstypen en methoden om XML databases dynamisch te programmeren zonder enige databaseprogramma's te hoeven gebruiken voor het aanmaken van de connecties.

Het DataSet object is het basis object om XML tabellen en velden te maken en daarna deze met gegevens te vullen. Wat er allemaal in de XML bestanden gebeurt hoeven we niet meer naar om te kijken.

De instantie van het DataSet object maken we aan in de Main module in Visual Basic.

```
Module modMain
    Public ds As DataSet

    Public Sub Main()
        ds = New DataSet
        If Not (ds Is Nothing) Then
            Try
                Application.Run(New mdiForm)
            Catch ex As Exception
                'Hier foutmelding geven met de methode MessageBox.Show()
            End Try
        Else
            'Hier foutmelding geven dat de database niet kan werken
        End If
    End Sub
```

```
End Module
```

Uiteraard mag de objectinstantie ook in de Load event van het mdiForm worden gecreëerd, maar stel dat er een exception optreedt. We moeten dan weer het mdiForm sluiten zodat de applicatie wordt beëindigd. Met bovenstaande code wordt de objectinstantie gecreëerd voordat de applicatie het mdiForm runt. Daarom is het beter om zulke initialisaties en creaties eerst in de Main subroutine te laten beginnen in een module.

De DataTable, DataColumn en DataRow objecten

Met deze drie objecttypen worden de tabellen, kolommen en rijen gemaakt. Hoewel ze apart gemaakt moeten worden, hebben ze toch een samenwerking. Elke instantie van deze drie objecten komt terecht in de databoom. Zodra ze in de boom zijn toegevoegd, hebben we vanuit het basisobject, ook wel de wortel genoemd, toegang tot de gegevens die bewerkt kunnen worden. Maar let op, als we eerst een tabel hebben toegevoegd kunnen we niet meteen een nieuwe rij toevoegen, ook al hebben we al toegang tot de Rows collectie. Eerst hebben we de Columns collectie nodig om kolommen aan te maken. Dat doen we met behulp van het DataColumn object.

Met de ds objectinstantie kan nu een nieuw tabel worden aangemaakt:

```
ds.Tables.Add(tabelnaam)
```

De Add methode geeft echter het nieuwe DataTable object terug, zodat we met de instantie kunnen werken:

```
NieuwTabel = ds.Tables.Add(tabelnaam)
```

Of we maken eerst een nieuw tabel aan en geven dat door aan de Add methode:

```
ds.Tables.Add(NieuwTabel)
```

De tweede manier, om de Add methode het nieuwe DataTable object terug te laten geven, is de beste keuze. Stel dat het fout gaat met het toevoegen van een nieuw tabel; we zouden dan met de statements Try en Catch de fout moeten onderdrukken. Door de tweede keuze te nemen, kunnen we code maken zonder Try en Catch, zoals onderstaand voorbeeld laat zien:

```
Dim NieuwTabel As DataTable = ds.Tables.Add("Nieuwe Tabel")
If Not (NieuwTabel Is Nothing) Then
    'Plaats hier code om de nieuwe tabel te bewerken
Else
    'Geef hier een Message.Show() melding dat het aanmaken van een nieuwe tabel is mislukt
End If
```

Maar dit kan ook met de derde keuze zodat we de Add methode als een subroutine gaan gebruiken in plaats van een functie:

```
Dim NieuwTabel As DataTable = New DataTable
If Not (NieuwTabel Is Nothing) Then
    'Plaats hier code om de nieuwe tabel te bewerken
    ds.Tables.Add(NieuwTabel)
Else
    'Geef hier een Message.Show() melding dat het aanmaken van een nieuwe tabel is mislukt
End If
```

Op de derde manier moet altijd eerst de nieuwe tabel worden bewerkt voordat deze toegevoegd mag worden. Waarom gaat dat op de tweede manier dan andersom? Als we de nieuwe tabel definiëren en deze gelijk gaan toevoegen in hetzelfde Dim statement kan het bewerken daarna worden gedaan, omdat de NieuwTabel instantie naar ds.Tables() verwijst. Dat gebeurt niet op de derde manier en moeten we dus na het bewerken van de NieuwTabel instantie die toevoegen aan ds.Tables().

Is het aanmaken van de tabel gelukt dan kan het nieuwe record worden gemaakt dat bij de tabel hoort. Het creëren van een record gaat precies zo te werk als met het creëren van een tabel, op één verschil na. We moeten eerst de juiste kolommen hebben om rijen (records) toe te kunnen voegen.
Dat betekent niet dat we daarna geen nieuwe kolom toe zouden mogen voegen. XML werkt dynamisch dus de DataSet ook. Een nieuw kolom zorgt er automatisch voor dat de cellen in de rijen op die kolom leeg zijn.

Het aanmaken van kolommen doen we met de Columns() collectie. Die collectie zorgt gelijk voor de rijen, zodat we zowel de Columns() collectie als de Rows() collectie in de Tables() terug kunnen vinden.

Onderstaande voorbeeld laat zien hoe u een kolom en een rij toevoegt. De kolom wordt toegevoegd op de derde manier, net zoals bovenstaande derde manier met de tabel toevoegen.

```
Dim NieuwKolom As DataColumn = New DataColumn
If Not (NieuwKolom Is Nothing) Then
    With NieuwKolom
        .DataType = System.Type.GetType("System.Decimal")
        .AllowDBNull = False
        .Caption = "Prijs"
        .ColumnName = "Prijs"
        .DefaultValue = 25
    End With
    ds.Tables(0).Columns.Add(NieuwKolom)

    Dim NieuwRij As DataRow = ds.Tables(0).NewRow()
    If Not (NieuwRij Is Nothing) Then
        NieuwRij("Prijs") = 50
        ds.Tables(0).Rows.Add(NieuwRij)
    End If
End If
```

Een rij toevoegen gaat echter op een hele andere manier. Door de methode NewRow() van de juiste tabel aan te roepen, kunt u de nieuwe rij bewerken. Echter, NewRow() is niet voldoende. Nadat u de rij bewerkt heeft moet u die toevoegen aan de Rows() collectie.

Het object DataColumn heeft nog een eigenschap die niet in bovenstaand voorbeeld aanwezig is. De optionele boolean variabele `Unique`. U hebt hem alleen nodig als u relaties wilt maken met meerdere tabellen. De Tables() collectie kan deze relaties echter niet alleen voor u regelen. Daar hebt u een andere object voor nodig, het DataRelation object. Met dit object verbindt u de tabellen door aan de juiste rijen van de tweede tabel de unieke sleutelkolom mee te geven aan de rijen van de eerste tabel. Op de rijen hoeft u zich echter niet op te concentreren. Het gaat alleen maar om het koppelen van de twee tabellen. Als dat klaar is, gaat het relatief werken met de rijen automatisch. U moet wel zelf de juiste sleutels opgeven. Denk maar aan de vorige nieuwsbrief toen ik het over de enumeratietypen had.

Onderstaande code geeft een praktijkvoorbeeld uit bovenstaande theorie.

```
Dim ouderKolom As DataColumn = ds.Tables("Ouder").Columns("SleutelID")
Dim kindKolom As DataColumn = ds.Tables("Kind").Columns("SleutelID")

' Creëer relatie.
Dim relOuderKind As DataRelation = New DataRelation("OuderKind", ouderKolom, kindKolom)

' Voeg de relatie toe aan de DataSet.
ds.Relations.Add(relOuderKind)
```

U kunt zoveel relaties maken als u zelf wilt. Het is daarom mogelijk om een 1 op n relatie te ontwerpen, waardoor een tabel een relatie met meer dan één tabel relatief werkt.

Het XML schema inlezen en opslaan.

Het DataSet basis object heeft vier methoden om op twee manieren het XML schema in te lezen en op te slaan. Onderstaande methoden staan in een tabel met de beschrijvingen van de methoden.

ReadXML	Leest het XML schema in, maar leest ook de gegevens in.
ReadXMLSchema	Leest het XML schema in, zonder de gegevens.
WriteXML	Schrijft het XML schema weg met de opgeslagen gegevens.
WriteXMLSchema	Schrijft het XML schema weg, zonder de opgeslagen gegevens.

Met de opgeslagen gegevens bedoel ik dat deze bewaard zijn in de dataset, niet in het bestand.

Het lezen en schrijven van bovenstaande vier methoden kan ook nog eens op verschillende manieren zoals onderstaande tabel aangeeft.

System.Xml.XmlReader System.Xml.XmlWriter	Lezen/schrijven via een opgegeven XmlReader/XmlWriter object.
System.IO.Stream	Lezen/schrijven via een opgegeven Stream object.
System.IO.TextReader System.IO.TextWriter	Lezen/schrijven via een opgegeven TextReader/TextWriter object. U kunt deze objecten vergelijken met de verouderde Open en Close statements.
String	Leest het schema in/schrijft het schema weg met eventueel de gegevens via het opgegeven bestandsnaam, inclusief padnaam, als String type.

De System typen in de tabel kunt u kiezen als parameter in de methoden van het DataSet basis object. U kunt er maar één kiezen, want de methoden hebben maar één parameter. Met andere woorden: ze zijn 3 keer *overloaded*. In Visual Basic .NET kunnen methoden gekopieerd worden en andere parameters hebben. Eén van de methoden is altijd de originele en wordt niet meegerekend.

Het relationeel model.

Het relationele model is een datamodel dat is opgebouwd volgens de regels van de relationele algebra. In relationele databases zijn gegevens opgeslagen in tabellen die zijn ontworpen aan de hand van fysieke relationele datamodellen.
Relationele databases zijn verre van de enige manier om data op te slaan, maar zij vormen tegenwoordig wel de factor: de standaard in de industrie. Producten als Oracle en DB2 hebben de relationele datamodelleringstheorie in de jaren '80 algemeen ingang doen vinden in bedrijven en instellingen. Daarnaast hebben desktopproducten als Microsoft Access relationele databases toegankelijk gemaakt voor het grote publiek.

Heersende verwarring in terminologie.

Al te vaak wordt gedacht dat de term relationele database afkomstig is van het gebruik van relaties die tussen tabellen worden gelegd. Dit is niet het geval. het woord relationeel is van toepassing binnen één tabel en heeft niets te maken met relaties tussen tabellen onderling. De grote verwarring hier omtrent is het gevolg van diverse factoren:

1. Een relationele database bestaat in de praktijk altijd uit verschillende tabellen waartussen relaties mogelijk zijn. Intuïtief, maar ook helaas foutief, is de link tussen 'relationele database' en deze relaties zijn dan al te snel gelegd.
2. In het Nederlandstalige taalgebied komt daar nog bij dat zowel de relatietabellen als de relaties tussen die tabellen, dus eigenlijk relaties tussen wiskundige relaties, allebei aangeduid kunnen worden met het begrip 'relatie'. In het Engels is er een onderscheid 'Relation' en 'Relationship'.
3. De fout is zover doorgedrongen dat ze ook voorkomt in boeken waarvan tot miljoenen exemplaren worden verkocht, zoals boeken over 'How computers work', bij gerenommeerde uitgeverijen en zelfs op websites die claimen over databases en SQL te gaan. Het terugdraaien is dus bijna onmogelijk geworden.
4. Zoals eerder opgemerkt worden desktop versies van relationele databases gebruikt door het (zeer) grote publiek. Basiskennis van relationele databases, multi-user, isolation, transactions, locking enzovoort, is niet langer meer een vereiste, en iedereen denkt al snel weg te zijn met de relationele technologie. Hierdoor wordt al te snel uitgegaan van een kennis van de materie en in het geval van 'relationeel' zelf een eigen invulling gegeven door zelfs de grote meerderheid van de informatici.

Relaties in databaseopslag.

Een relatie in databaseopslag is een tweedimensionale tabel die wordt gebruikt om data op te slaan, zie de bovenstaande praktijk over het DataRelation object. Om een relatie genoemd te worden dient deze tabel aan een aantal voorwaarden te voldoen.

Onderdelen

Een relatie bestaat uit een heading en een body. De heading bevat de attribuutnamen, de body en de tuples. Een tuple is een verzameling van ongeordende attribuutwaarden, terwijl een attribuut op zijn beurt bestaat uit zijn naam en zijn waarde.

Omdat het gaat om een tweedimensionale tabel, kunnen we, zij het grofweg, stellen dat:

- de heading overeenkomt met de tabelhoofding;

- de body overeenkomt met de rijen daaronder;
- een tuple overeenkomt met een rij;
- een attribuutwaarde overeenkomt met een waarde die in de tabel is ingevuld;
- een attribuutnaam de naam is die wordt weergegeven in de tabelhoofding.

Tabellen en relaties.

Praktische benadering
Een tabel is een standaardmanier om gegevens op te slaan die reeds ingang vond lang voor er sprake was van databases en zelfs lang voor computers bestonden. Het is een logische en bijna intuïtieve manier van opslag die zelfs geen verdere uitleg vereist. Als het relationele model slechts hieruit bestond zou *Ted Codd*, de man die dit model voor het eerst beschreef in 1970, er niet zo een naam mee hebben gemaakt, en zouden softwareontwikkelaars genoegen nemen met het ontwikkelen van spreadsheetprogramma's, de tabelapplicaties bij uitstek.

Toch zijn deze intuïtieve tabellen, met rijen en kolommen, ook de basis van het relationele model. Niet alle tabellen echter voldoen aan de eisen om in aanmerking te komen voor dataopslag binnen relationele databases. Tabellen moeten hiervoor voldoen aan al deze criteria. Indien een tabel voldoet aan al deze criteria, noemen we deze tabel een relatievariabele. Alle relaties zijn dus tabellen, maar niet alle tabellen zijn relatievariabelen. Omdat de term relatievariabele nogal omslachtig is, wordt die in de praktijk steeds afgekort tot *relvar*, gelukkig ook in het Engels daar afgekort voor relation variable.

NOOT: verder gebruiken we voor rij ook wel de term record en kolom noemen we ook wel variabele of attribuut.

De criteria waaraan een tabel moet voldoen, om een relvar te worden genoemd, zijn:
- Elke rij beschrijft juist één entiteit.
- Binnen een rij worden nooit gegevens bijgehouden over meerdere entiteiten en één entiteit wordt binnen één tabel nooit beschreven over meerdere rijen.
- Elke rij is uniek.
- Elke kolom beschrijft precies één attribuut van een entiteit en in elke rij is de betekenis van het attribuut dezelfde.
- De volgorde van de rijen is onbelangrijk. Dit wil zeggen dat een andere volgorde van rijen nooit tot informatieverlies kan leiden.
- De volgorde van de kolommen is onbelangrijk. Dit wil zeggen dat een andere volgorde van kolommen nooit tot informatieverlies kan leiden.

Wiskundige benadering: een basis.

Inleiding
Een wiskundige relatie is het begrip waarvan de term 'relationele database' is afgeleid. Ted Codd ontwikkelde indertijd het relationele model door wiskundige logica toe te passen op gegevensopslag en was daarmee de absolute grondlegger van de relationele algebra en het relationeel model en bij uitbreiding de relationele database.

Een relatie
Een relatie kan wiskundig gezien worden als een bepaald verband tussen verschillende verzamelingen. Dit verband kan bestaan tussen twee verzamelingen (V1 en V2) of meer (V1, V2, ...Vn) verzamelingen.

Koppels
Indien het verband zich stelt tussen twee verzamelingen, bestaat de relatie uit een verzameling koppels. Alle mogelijke koppels die gevormd kunnen worden tussen twee verzamelingen noemen we het cartesisch product. In de praktijk zal een relatie niet alle mogelijke koppels behelsen, maar een subset van deze koppels.

Een voorbeeld: *(zie dit ook in de vorige nieuwsbrief over 'Zelf gegevenstypen maken.')*

```
V1 = {1, 2, 3}
V2 = {2, 4}
```

Het cartesisch product van deze relatie zijn alle mogelijke combinaties van deze twee sets:
{(1, 2), (1, 4), (2, 2), (2, 4), (3, 2), 3, 4)}

Een voorbeeldrelatie tussen deze twee sets zou kunnen zijn: *<: V1 is kleiner dan V2*. De relatie kan dan uitgedrukt worden als volgende verzameling koppels: {(1,2),(1,4),(2,4),(3,4)}

In de praktijk bestaat een relatie dus uit een subset van het cartesisch product van de verzamelingen die deel uitmaken van die relatie.

n-tupels
Indien de relatie een verband uitdrukt tussen meer dan twee verzamelingen, spreken we van een n-tupel waarbij de n staat voor de graad van de relatie (3 verzamelingen: 3-tupel, 4 verzamelingen: 4-tupel enzovoort.

Een voorbeeld van zo'n relatie is bijvoorbeeld x=y*z. Gesteld dat deze relatie plaatsvindt tussen de 3 verzamelingen V1, V2 en V3 waarbij x deel uitmaakt van V1, y van V2 en z van V3 en waarbij V1, V2 en V3 allemaal bestaan uit de natuurlijke getallen 2 tot en met 10 is de relatie de verzameling van 3-tupels: {(4,2,2),(6,2,3),(6,3,2),(8,2,4),(8,4,2),(9,3,3),(10,2,5),(10,5,2)}

De relatie: toepassing op gegevensopslag
Een relatie zoals de n-tupel van graad 3 hierboven laat zich dus schrijven door een opsomming van tupels. Wiskundig wordt dit gedaan zoals hierboven: met accolades, haakjes en komma's. Een andere mogelijke manier om hetzelfde te doen is gebruikmaken van een tabelvorm. Als hoofding worden hier dan de verzamelingen gebruikt waaruit de waarden mogelijk kunnen komen. Zo een [verzameling] heet binnen [gegevensopslag]: [het domein].

V1	V2	V3
4	2	2
6	2	3
6	3	2
8	4	2
8	2	4
9	3	3
10	2	5
10	5	2

Bij een relatie van data is dat niet anders. Evenals als de n-tupel van graad 3 hierboven (V1=V2*V3) is, heeft elke relatie een omschrijving nodig van wat er precies wordt voorgesteld. Zo een omschrijving noemen we een predicaat. Bij gegevensopslag gaat dat steeds om een volzin. Het predicaat van de netgenoemde 3-tupel wordt dus:

de waarde van V1 is de waarde van V2 vermenigvuldigd met de waarde van V3.

Een concreet voorbeeld van een relatie van graad 3 kan zijn:

KlantId	Familienaam	Voornaam
1	Verstappen	Jos
2	Vermeiren	Muriel

Het predicaat in dit geval is:

Klant met als ID 'KlantId' heeft als familienaam 'Familienaam' en als voornaam 'Voornaam'.

Marco Kurvers

Grafisch programmeren in GW-BASIC (5).

Na de mooie theorie en praktijk met voorbeelden over continue- en niet-continue tekenen in cartesische vorm, gaan we het nu over een hele andere techniek hebben. Een techniek die niet alleen mooiere tekeningen laat zien, maar ook nog eens de programma's ingewikkelder maakt. We zullen nu ook veel te maken krijgen met formules die parameters hebben. Voor Visual Basic .NET betekent dit dat de functies met parameters belangrijker worden. Een goede manier om nu dit alles op A4 formaat weer te geven.

Krommen in poolcoördinaten en in parametervorm.

In de vorige delen hebben we ons beziggehouden met het tekenen van de grafiek van een continue of niet-continue functie, waarvan de vergelijking als y = f(x) geschreven kon worden; dus in cartesische vorm. Nu gaan we grafieken tekenen van functies waarvan de vergelijking in poolcoördinaten geschreven wordt of in de zogeheten parametervorm. Dit levert zeer fraaie 'beelden' op.

Functies, waarvan de grafiek een gesloten kromme laat zien, zijn vaak eenvoudiger met poolcoördinaten te beschrijven dan in cartesische vorm. Als voorbeeld noemen we de 'hartkromme' (kardioide), waarvan de vergelijking in poolcoördinaten eenvoudig is, namelijk:

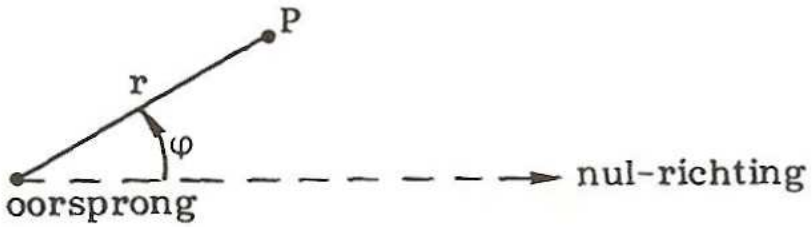
$$r = k(1 + \cos\varphi),$$

maar waarvan de cartesische vorm nogal ingewikkeld is, namelijk

$$\frac{(x^2 - y^2 - kx)^2}{k^2(x^2 + y^2)} = 1.$$

Om de volgende programma's te doorgronden (niet om ze te draaien!) is een beetje theorie nodig. Het poolcoördinatenstelsel wordt in de wiskundelessen op school niet of nauwelijks behandeld. Eigenlijk is dit jammer, want hiermee (en met de parametervorm) kunnen juist de mooiste functies (en daarmee de fraaiste grafieken) beschreven en getekend worden.

In een poolcoördinatenstelsel wordt elk punt in het platte vlak met twee coördinaten, te weten r en φ, bepaald. r is de afstand tussen het punt en de oorsprong en φ (phi, spreek uit: fie) is de hoek tussen de horizontale lijn door de oorsprong en de lijn door de oorsprong en het punt, zie onderstaande tekening.

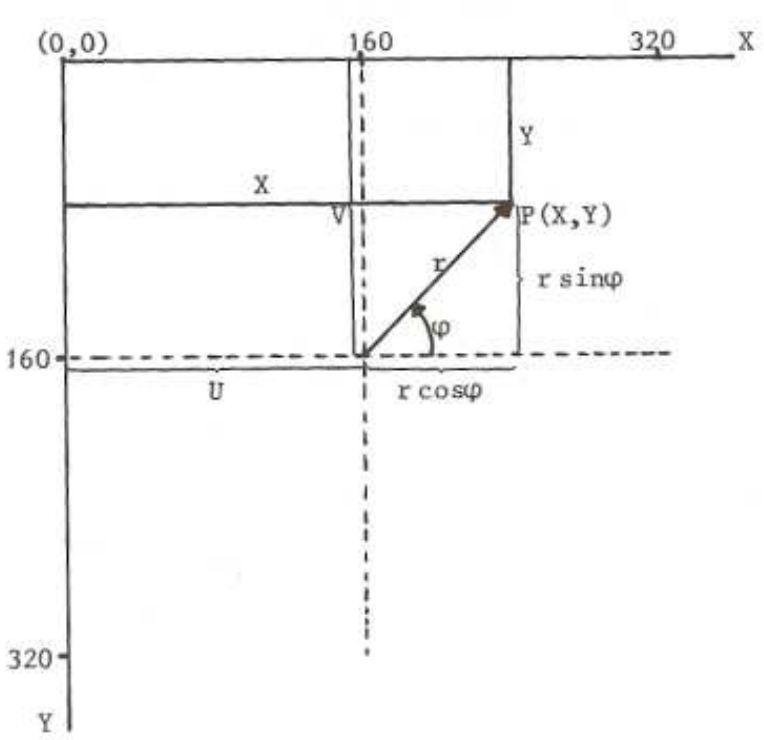


De oorsprong van het poolcoördinatenstelsel leggen we in het middelpunt van het beeldscherm (U = 160, V = 160). Zoals we weten ligt de oorsprong van het beeldschermcoördinatenstelsel in de linkerbovenhoek van het beeldscherm (HOME positie). De nul-richting in ons poolcoördinatenstelsel is horizontaal en wijst naar rechts.
$\varphi=90^0$ (π / 2 radialen) is verticaal naar boven; $\varphi=180^0$ (π radialen) is horizontaal naar links en $\varphi=270^0$ (3π radialen) is verticaal naar beneden.

We gebruiken de volgende twee transformatieformules om het, uit de functievergelijking berekende, punt P(r, φ) om te zetten in het beeldschermpunt P(X, Y):

```
X = INT(U + R*COS(P) + 0.5)     èn
Y = INT(V - R*SIN(P) + 0.5)
```

In onderstaande figuur zien we hiervoor de verklaring.



De hoek φ, in de programma's voorgesteld door de variabele P, doorloopt steeds in positieve richting (tegen de klok in) het interval van 0 tot 2π. Dit is een interval in radialen (0-360 in graden).

In alle programma's is als lusvariabele de hoek W in graden gekozen.
Met de formule P = W . (π / 180) (in de programma's P = W * RD) wordt de hoek in radialen berekend. Het voordeel van een lusvariabele die het interval $0\text{-}360^0$ doorloopt is dat de stapgrootte waarmee de lusvariabele steeds wordt opgehoogd een geheel getal kan zijn (1 bijvoorbeeld) en dat hierdoor de programma's beter leesbaar zijn. Een neveneffect is dat alle programma's bijna woordelijk in Pascal zijn over te zetten; Pascal kent immers geen gebroken stapgrootte in een lus.

TIP! In de tijd dat we nog GW-BASIC gebruikten, in de DOS tijd, bestond de programmeertaal Pascal al. Dat we geen gebroken getallen als stapgrootte in Pascal lussen kunnen gebruiken is gedeeltelijk waar. Als u van plan bent de programma's in Pascal over te zetten, moet u de FOR lussen omzetten in Repeat ... Until lussen. In Pascal werken de For lussen alleen maar met stapgrootte van 1.
Meer daarover, lees onderwerp 'Van programmeertaal naar Basic'.

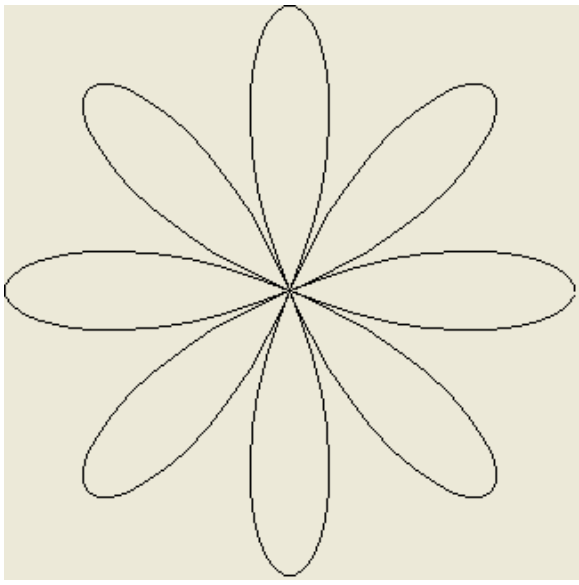
Meer theorie hebben we niet nodig. De volgende programma's spreken grotendeels voor zichzelf.

Programma 12 tekent de grafiek van de functie

$$r = k \cos(n\varphi).$$

In het programma kiezen we k = 160 (K) en $n = 4^0$.
Kies K = 110 als u in de middenresolutie (320x200) werkt.

```
100 'progr.12  GRAFIEK VAN DE FUNCTIE R*COS(4*PHI)
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160 : V=160 : H=.5 : RD=4*ATN(1)/180
150 K=160 : P=0 : GOSUB 1000
160 X1=INT(U+K*R*COS(P)+H):Y1=INT(V-K*R*SIN(P)+H)
170 FOR W=1 TO 360 STEP 2
180    P=W*RD : GOSUB 1000
190    X2=INT(U+K*R*COS(P)+H)
200    Y2=INT(V-K*R*SIN(P)+H)
210    LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
220    X1=X2 : Y1=Y2
230 NEXT W
240 A$=INKEY$: IF A$="" THEN 240
250 CLS: KEY ON: END
1000 R=COS(4*P)
1010 RETURN
```



```
Private Function PoolVorm(ByVal P As Single) As Single
    Dim R As Single = Math.Cos(4 * P)
    Return R
End Function

Private Sub frmProg12_Paint(...) Handles MyBase.Paint
    Dim U As Single = 160, V As Single = 160, H As Single = 0.5
    Dim RD As Single = 4 * Math.Atan(1) / 180
    Dim K As Single = 160, P As Single = 0
    Dim R As Single = PoolVorm(P)
    Dim X1 As Single = Int(U + K * R * Math.Cos(P) + H)
    Dim Y1 As Single = Int(V - K * R * Math.Sin(P) + H)
    For W As Integer = 1 To 360 Step 2
        P = W * RD : R = PoolVorm(P)
        Dim X2 As Single = Int(U + K * R * Math.Cos(P) + H)
        Dim Y2 As Single = Int(V - K * R * Math.Sin(P) + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
End Sub
```

Het programma tekent een achtdelige draaisymmetrische figuur; een bloem met acht blaadjes. Als u met dit programma gaat experimenteren zult u snel ontdekken dat voor even waarden van n een 2n-bladerige en voor oneven waarden van n een n-bladerige bloem ontstaat. Kunt u een tweekleurige bloem maken?

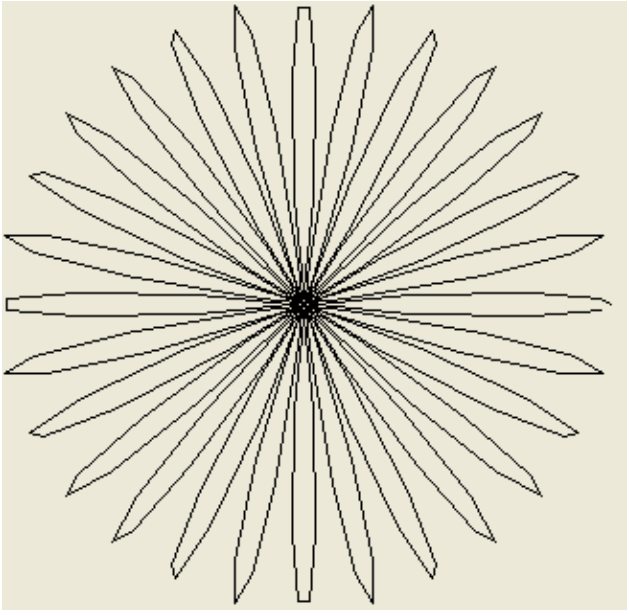
Wilt u dat uw computer sneller tekent? Neem dan een stapgrootte van 3^0 of van 5^0 in plaats van 2^0. Bedenk dan wel dat de blaadjes wat hoekiger worden.

Als u regel 1000 R = COS(4*P) vervangt door één van de onderstaande functies, krijgt u steeds een andere mooie figuur:

```
R = COS(4*SIN(2*P))   : R = ABS(R)
R = COS(4*SIN(3*P))   : R = ABS(R)
R = SIN(3*SIN(2*P))   : R = ABS(R)
R = SIN(5*COS(2*P))   : R = ABS(R)
```

U kunt naar hartelust trigonometrische functies met verschillende parameters 'mengen'. Laat uw creativiteit en nieuwsgierigheid de vrije loop.

Omdat de poolcoördinaat r per definitie niet negatief mag zijn, moeten we in regel 1000 ook de opdracht R = ABS(R) opnemen.



Deze figuur ontstaat als we in programma 12 in regel 1000 R = COS(14*P) nemen.

Gebruiken we Visual Basic.NET dan kan de afbeelding van hiernaast anders uitkomen dan wanneer we GW-BASIC gebruiken. Hoewel het resultaat van de berekening uit de functie hetzelfde is kan de tekening dus verschillend zijn. Dat kan te maken hebben met de resolutie die we gebruiken.

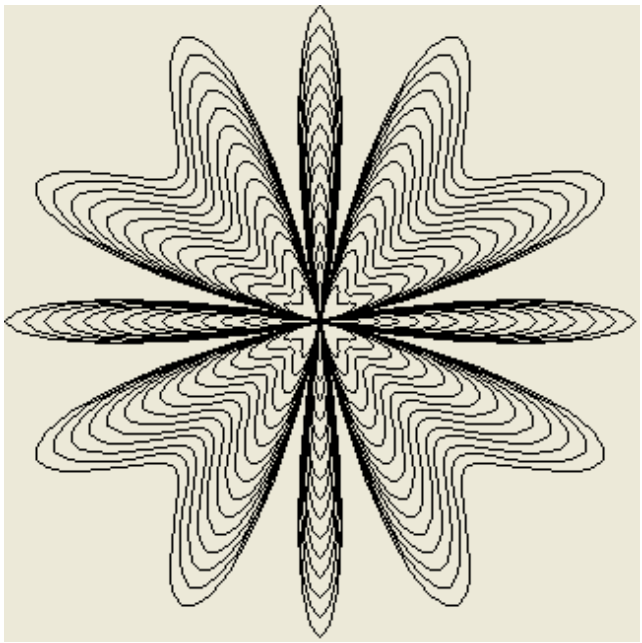
Nog veel mooiere figuren krijgen we als we niet één grafiek, maar een stelsel van gelijkvormige grafieken tekenen.

Programma 13 tekent het stelsel krommen:

R = K*COS(4*SIN(2*P)).

De parameter K loopt van 20 tot 160 in stappen van 10.
Neem K = 20 TO 100 STEP 10 als u in de middenresolutie (320x200) werkt. Voor Visual Basic zou het kunnen gebeuren dat de grafiek er weer anders uit komt te zien. Zoals u echter aan onderstaande tekening kunt zien is er nu geen verschil met de opgegeven luswaarden voor K.

```
100 'progr. 13 GRAFIEK VAN  R=K*COS(4*SIN(2*PHI))
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160: V=160: H=.5: RD=4*ATN(1)/180
150 FOR K=20 TO 160 STEP 10
160    P=0: GOSUB 1000
170    X1=INT(U+K*R*COS(P)+H)
180    Y1=INT(V-K*R*SIN(P)+H)
190    FOR W=2 TO 360 STEP 2
200       P=W*RD: GOSUB 1000
210       X2=INT(U+K*R*COS(P)+H)
220       Y2=INT(V-K*R*SIN(P)+H)
230       LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
240       X1=X2: Y1=Y2
250    NEXT W
260 NEXT K
270 A$=INKEY$: IF A$="" THEN 270
280 CLS: KEY ON: END
1000 R=COS(4*SIN(2*P))
1010 RETURN
```



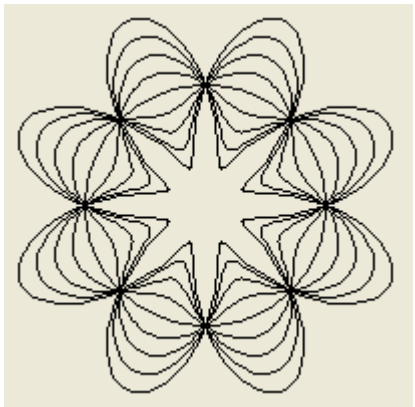
```
Private Function PoolVorm(ByVal P As Single) As Single
    Dim R As Single = Math.Cos(4 * Math.Sin(2 * P))
    Return R
End Function

'Plaats onderstaande codeblok in de Paint event.

Dim U As Single = 160, V As Single = 160, H As Single = 0.5
Dim RD As Single = 4 * Math.Atan(1) / 180
For K As Integer = 20 To 160 Step 10
    Dim P As Single = 0, R As Single = PoolVorm(P)
    Dim X1 As Single = Int(U + K * R * Math.Cos(P) + H)
    Dim Y1 As Single = Int(V - K * R * Math.Sin(P) + H)
    For W As Integer = 2 To 360 Step 2
        P = W * RD : R = PoolVorm(P)
        Dim X2 As Single = Int(U + K * R * Math.Cos(P) + H)
        Dim Y2 As Single = Int(V - K * R * Math.Sin(P) + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
Next
```

Programma 14 tekent sinuskrommen die cirkelvormig gekromd zijn.

```
100 '   programma 14    SINUSKROMMEN IN CIRKELS
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160 : V=160 : H=.5 : RD=4*ATN(1)/180
150 FOR K=-40 TO 40 STEP 10
160    X1=U+60 : Y1=V
170    FOR W=2 TO 360 STEP 2
180       P=W*RD : R=60+K*SIN(4*P)
190       X2=INT(U+R*COS(P)+H)
```



```
200       Y2=INT(V-R*SIN(P)+H)
210       LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
220       X1=X2 : Y1=Y2
230    NEXT W
240 NEXT K
250 A$=INKEY$: IF A$="" THEN 250
260 CLS: KEY ON: END
```

```
Dim U As Single = 160, V As Single = 160, H As Single = 0.5
Dim RD As Single = 4 * Math.Atan(1) / 180
For K As Integer = -40 To 40 Step 10
    Dim X1 As Single = U + 60, Y1 As Single = V
    For W As Integer = 2 To 360 Step 2
        Dim P As Single = W * RD, R As Single = 60 + K * Math.Sin(4 * P)
        Dim X2 As Single = Int(U + R * Math.Cos(P) + H)
        Dim Y2 As Single = Int(V - R * Math.Sin(P) + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
Next
```

Probeert u eens:

```
1000 R=2*TAN(2*P)+K*SIN(2*COS(SIN(6*P)))
```

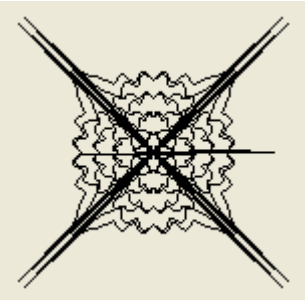
En voor Visual Basic een nieuwe PoolVorm functie met als Return code:

```
Return 2 * Math.Tan(2 * P) + K * Math.Sin(2 * Math.Cos(Math.Sin(6 * P)))
```

Verander de Dim regel voor variabele R in:

```
Dim R As Single = PoolVorm(P, K)
```

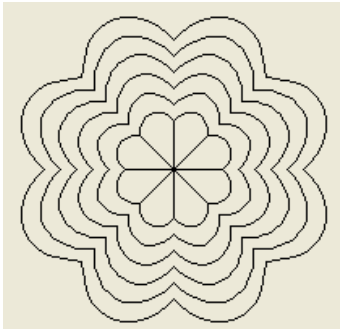
Zorg er dus voor dat u een PoolVorm functie hebt met twee parameters.



De nieuwe functie zorgt voor een schitterend spinnenweb. De Variaties zijn echt onbegrensd!

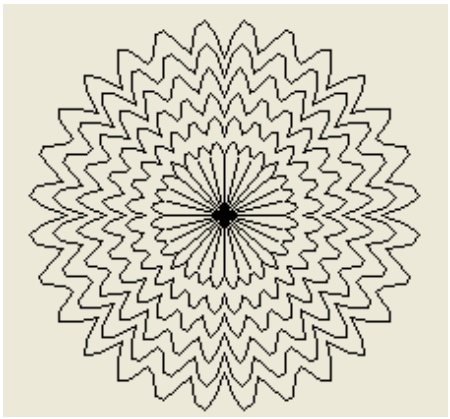
Programma 15 tekent een bloemvormig figuur met blaadjes die in het midden aan steeltjes vastzitten. In het programma is N = 4 gekozen. U kunt gerust andere waarden voor N proberen. Het effect is hetzelfde als bij programma 12.

```
100 '        programma 15              BLOEMEN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160 : V=160 : H=.5 : RD=4*ATN(1)/180
150 N=4 : C=.25
160 FOR K=30 TO 80 STEP 10
170    X1=U+K : Y1=V
180    FOR W=3 TO 360 STEP 3
190       P=W*RD: R=K*(1+C*ABS(SIN(N*P)))
200       X2=INT(U+R*COS(P)+H)
210       Y2=INT(V-R*SIN(P)+H)
220       LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
230       X1=X2 : Y1=Y2
```



```
240    NEXT W
250 NEXT K
260 R=30 : P1=(180/N)*RD
270 FOR J=1 TO N
280    P=J*P1
290    X1=INT(U+R*COS(P)+H)
300    Y1=INT(V-R*SIN(P)+H)
310    X2=INT(U+R*COS(P+4*ATN(1))+H)
320    Y2=INT(V-R*SIN(P+4*ATN(1))+H)
330    LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
340 NEXT J
350 A$=INKEY$: IF A$="" THEN 350
360 CLS: KEY ON: END
```

Kies voor deze figuur: `N = 14`



```
Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5
Dim RD As Single = 4 * Math.Atan(1) / 180
Dim N As Integer = 4, C As Single = 0.25
For K As Integer = 30 To 80 Step 10
    Dim X1 As Integer = U + K, Y1 As Integer = V
    For W As Integer = 3 To 360 Step 3
        Dim P As Single = W * RD
        Dim R As Single = K * (1 + C * Math.Abs(Math.Sin(N * P)))
        Dim X2 As Integer = Int(U + R * Math.Cos(P) + H)
        Dim Y2 As Integer = Int(V - R * Math.Sin(P) + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
Next
Dim R1 As Integer = 30, P1 As Single = (180 / N) * RD
For J As Integer = 1 To N
    Dim P As Single = J * P1
    Dim X1 As Integer = Int(U + R1 * Math.Cos(P) + H)
    Dim Y1 As Integer = Int(V - R1 * Math.Sin(P) + H)
    Dim X2 As Integer = Int(U + R1 * Math.Cos(P + 4 * Math.Atan(1)) + H)
    Dim Y2 As Integer = Int(V - R1 * Math.Sin(P + 4 * Math.Atan(1)) + H)
    e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
Next
```

In het programma, geschreven in Visual Basic.NET, lijkt de code wat verwarrend.
Ten eerste heb ik voor het eerst Integers gebruikt voor de variabelen die afgerond worden met een Int() functie, maar Single variabelen maakt geen verschil. Integer variabelen kost alleen wat minder geheugen.
Ten tweede hebben we nu een programma met twee aparte lusstructuren die gebruik maken van de variabelen X1, Y1 en X2, Y2 en ook weer een variabele P. Omdat die variabelen lokaal in de eerste lus zijn gedeclareerd, moeten ze in de tweede lus nogmaals gedeclareerd worden, anders worden ze niet herkend. Ik had ook een aparte Private declaratie kunnen maken met al die variabelen, maar dat maakt niet uit.
Ten derde zien we tussen de twee lussen een variabele R1 gedeclareerd die de waarde 30 krijgt. Waarom is die variabele gedeclareerd als naam R1 en niet gewoon als naam R? De variabele R in de eerste lus is toch lokaal? Ja, dat zouden we denken. Echter, dit werkt alleen als de variabelen in codeblokken gedeclareerd worden net zoals de variabele P, niet als de variabele buiten de codeblokken (dus alleen in de subroutine zelf) gedeclareerd wordt. Zouden we dat doen dan protesteert de compiler met de foutmelding: Variable R 'hides' a variable in enclosing block.

Bezit u een kleurenmonitor of een kleurenplotter dan kunt u na elke doorloop van de K lus de afdrukkleur veranderen. Ook de stelen van de bloem kunnen een eigen kleur krijgen. Tekent u een aantal van dergelijke bloemen naast of onder elkaar dan hebt u een begin gemaakt met 'computer art'.

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

BASIC nieuws, tips en puzzels.

Het nieuwe jasje van de nieuwsbrief.

Deze nieuwsbrief is de eerste nieuwsbrief met een nieuw jasje. Niet alleen is de tekst nu geschreven met een beter leesbaar lettertype, ook is de opmaak nu beter geworden. Zo hebben de tabellen een eigen lettertype- en kleurindeling, worden de nummering en opsommingstekens met een andere grootte en kleur getoond en wat nog het belangrijkste is; de nieuwsbrief bestaat niet meer uit A5 pagina's, maar uit A4 pagina's. Dat geeft meer ruimte voor de opmaak van afbeeldingen en tekst. Grotere alinea's die uitleg geven over een afbeelding passen nu beter bij de figuur en hoeft de afbeelding niet meer uit noodzaak verkleind te worden.
Kortom, een betere nieuwsbrief voor een leuk, leerzaam, leesplezier.

Heeft u nog wat ideeën voor de opmaak van de nieuwsbrief of hebt u wat anders dat u kwijt wil, stuur dan een bericht naar mij.

Oplossing puzzel: het juiste BASIC onderwerp.

Onderstaande puzzel bestaat uit omschrijvingen waarvan de juiste betekenissen gevonden moesten worden. Het was de bedoeling het juiste BASIC onderwerp te vinden met de eerste letters van de betekenissen.

Hebt u het op kunnen lossen? Hieronder kunt u de antwoorden vinden.

Aan deze variabele(n) kunnen alfanumerieke waarden toegekend worden.	String of Strings
Met deze functie verwijderen we de spaties in de tekst.	Trim
Deze functie zorgt ervoor dat negatieve waarden positief worden.	Abs
We moeten een beginwaarde en een eindwaarde opgeven met een FOR lus. Wat hebben we daarvoor nodig?	To
De meeste BASIC versies gebruiken dit commando om een programma te beëindigen.	End
Sommige BASIC versies ondersteunen een commando om twee teksten samen te voegen.	Merge
In QuickBASIC en latere versies kunnen we foutafhandeling routines schrijven met een commando na ON.	Error
Met dit BASIC sleutelwoord kunnen we het resultaat van een booleaanse conditie omdraaien.	Not
Met deze BASIC systeemvariabele kunnen we de tijd instellen.	Time

Het BASIC onderwerp is: **STATEMENT**

Marco Kurvers

Van programmeertaal naar Basic.

Programmeurs die Basic gebruiken om te programmeren, denken alleen maar aan Basic code en gebruiken eventueel Basic controls als ze een IDE venster met formulieren hebben. De vraag is echter: zijn het wel Basic controls, zoals een TextBox, ListBox, CommandButton enzovoort? Controls kunnen we zelf in Basic maken, maar als we de bestaande controls op een formulier plaatsen dan hoeft die control niet speciaal in Basic gemaakt te zijn. Dat geldt niet alleen voor de controls. Ook DLL klassen library's kunnen ergens anders vandaan komen terwijl we ze in Basic gebruiken.

En onderdelen? Wat nu als we een onderdeel in Basic willen gebruiken dat geen control of een DLL is? Met andere woorden, code die we willen overnemen uit een andere programmeertaal.
Microsoft heeft daar ook aan gedacht en ontwierp converteer programma's en wizards om makkelijker code van de ene programmeertaal te kunnen importeren in een andere programmeertaal. Zelf heb ik dat al uitgezocht en daarom kom ik met dit onderwerp in Basic nieuws om u te laten zien of die converteer pro-

gramma's en wizards wel te vertrouwen zijn en dat we ook op een andere manier code kunnen vertalen, bijvoorbeeld handmatig. Ook al zouden we een dergelijke programmeertaal als Pascal en C niet 'goed' kennen, het belangrijkste is om eerst het geraamte van de code te begrijpen. U zult snel zien dat veel statements met de Basic statements overeenkomen, maar iets anders geschreven moeten worden. Het doel van elke statement zal toch hetzelfde blijven.

Converteersoftware en wizards
Microsoft en andere Internet sites bieden vele converteer programma's en wizards om code te kunnen vertalen zonder het zelf te moeten doen. Dit lijkt een koud kunstje, maar helaas werken die programma's niet 100% correct. Er zitten haken en ogen aan en ook al werken de programma's naar behoren zonder fouten; we zouden altijd na het omzetten wat code handmatig aan moeten passen, die niet door de programma's gezien konden worden of niet vertaald konden worden. Een ander probleem is de structuur in de code. Het wizard programma in Visual Basic.NET converteert Visual Basic 6 programma's, maar de versie 6 structuur zal echter geen .NET structuur worden. Althans, niet altijd. De wizard zal geen moeite hebben met kleine projecten, maar met grotere projecten met klassen enzovoort kunnen er problemen ontstaan. Denk maar eens aan de formulieren die geen objecten zijn maar sjablonen, zwarte dozen die de gegevens van de formulieren verborgen houden zodat we die niet kunnen wijzigen. Formulierinstanties maken zal door de wizard niet worden gedaan. Dat moeten we zelf doen.
De wizard zal een klassenmodule niet als een Public Class converteren, maar als een Friend Class. Dit is ook wat we niet altijd willen, tenzij de klasse 'vriend' moet blijven voor klassen uit andere projecten, bijvoorbeeld een project dat aan uw hoofdproject is toegevoegd. Dat kan tegenwoordig in een .NET versie.
Wat de wizard ook doet is de standaard variabele aanmaken in een property en de property naam als returnwaarde handhaven, zoals onderstaande code laat zien:

```
'Visual Basic 6.0
Private pNaam As String

Public Property Get Naam() As String
    Naam = pNaam
End Property

Public Property Let Naam(ByVal strWaarde As String)
    pNaam = strWaarde
End Property

'Visual Basic .NET
Private pNaam As String

Public Property Naam() As String
    Get
        Naam = pNaam
    End Get
    Set(ByVal Value As String)
        pNaam = Value
    End Set
End Property
```

Deze code is niet goed geconverteerd door de wizard. De twee property delen zijn wel goed ingepakt in één property, maar toch is er een versie 6 gedeelte achtergebleven. Wat we missen is het Return statement om in de Get codeblok de waarde van pNaam terug te geven. De wizard heeft de oude methode laten staan door gewoon de property naam te gebruiken. Als de programmeur wil dat de code echt goed geconverteerd moet worden naar .NET code dan heeft Microsoft nog veel te doen aan de wizard.

Pascal en Delphi
Programma's uit andere programmeertalen kunnen we helaas niet door de wizard laten converteren. Dat moeten we zelf doen. Pascal en Object Pascal (Delphi) zijn geen moeilijke programmeertalen en lijken ook wel op Basic. Pascal heeft het karakter van Basic en Delphi heeft het karakter van Visual Basic. Welke verschillen zijn er wel en welke niet? Bekijk eens onderstaand tabel.

Bewerking	Omschrijving
Variabelen declaraties	Niet met Dim, maar met Var. Voorbeeld: var Naam: string;
Toekenningen	Met operator := Verwar dit niet met := als de parametervariabele opgegeven wordt. Voorbeeld: functie(par:=10) in Basic. Voorbeeld in Pascal: Naam := 'Jan';
Codeblokken	Deze worden geplaatst tussen de statements begin en end.
Functie resultaten	In Pascal: zelfde manier als in basic, gewoon de functienaam. In Delphi: gebruik result voor een functie resultaat. Voorbeeld: result := waarde;
Gegevensstructuren	Pascal en Delphi hebben allerlei soorten gegevensstructuren onder meerdere sleutelwoorden. Basic kent deze mogelijkheden alleen met het Type sleutelwoord of het Structure sleutelwoord in .NET. Voorbeeld in Pascal en Delphi: Type typenaam = record <*velden*> end;

Wanneer we dus een recordtype willen converteren naar Basic code, hoeven we alleen maar '= record' te verwijderen en achter 'end' het Type statement te typen. U ziet ook de puntkomma. Dit teken lijkt sterk op de dubbelepunt in Basic, dat voor de helft juist is. In Pascal, Delphi en ook in C/C++ zijn de puntkomma's aan het eind van een statement verplicht, maar we hoeven in Basic geen dubbelepunt achter een statement te plaatsen als we met een volgende statement op de volgende regel verder willen.
Het voordeel van de puntkomma is dat code na bijvoorbeeld een IF ... THEN makkelijk op de volgende regel geplaatst kan worden en dan pas te eindigen met een puntkomma. In Basic kan dat ook wel, maar moeten we na IF ... THEN, of een andere statement, een underscore (onderstreping) plaatsen om Basic te vertellen dat de volgende regel bij de vorige regel hoort.

Bijvoorbeeld:

```
If A > 10 Then _
   A = 0
```

En in Pascal:

```
If (A > 10) Then
   A := 0;
```

Tip! Onthoud dat het meestal om het geraamte gaat van de code, niet om wat zich erbuiten afspeelt zoals de controls. Enkele voorbeelden hiervan zijn: de sleutelwoorden, de operatoren, de keuzestructuren, de lusstructuren en de declaraties.

Marco Kurvers

Verder kijken in Game Maker.

We weten nu dat er verschillende soorten bestanden zijn waar we de sprites in op kunnen bergen. Maar voor gebruik hebben we de objecten van de sprites nodig. Die objecten kunt u vergelijken met de objecten van klassenmodules in Basic. In een object zit de besturing, de events (gebeurtenissen) met daarin de acties. Hebben we dan ook acties in Basic? Eigenlijk niet, want wat wij gebruiken in een event is gewoon de code, de statements. De statements zijn in Game Maker de acties in een object event, maar niet alle acties kunnen we vergelijken als Basic statements. Een voorbeeld is de Move actie, die voor meer uitvoer zorgt om een object te laten bewegen. Dus is er meer Basic code nodig. Hebben we al een methode die dat voor ons doet, dan hebben we geluk. Maar meer code is nog steeds nodig, ook al zouden we die code niet zien. Waarom niet alle acties Basic statements kunnen zijn komt doordat zo'n actie zelf als een Basic sleutelwoord werkt. Hebt u Game Maker open dan eens de Control tab (zie Figuur 5 in de vorige nieuwsbrief), u zal dan knoppen zien als: codeblok openen, codeblok sluiten, knop Else enzovoort. De knoppen codeblok openen en codeblok sluiten kennen we als operatoren alleen in Pascal en C, maar niet in Basic. Een If ... Then statement met een codeblok wordt altijd afgesloten met een End If.

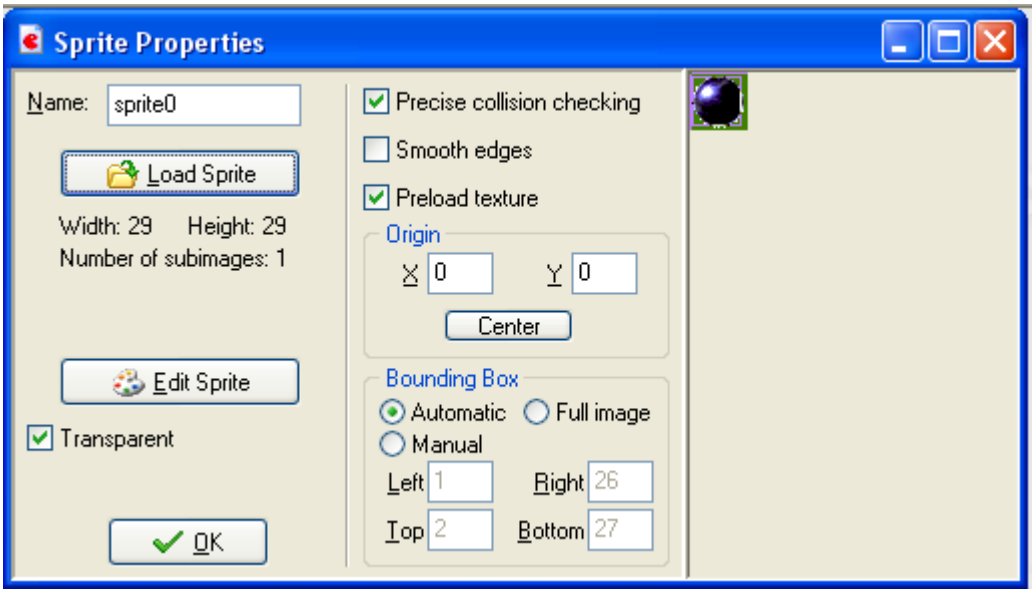
! Onthoud wat ik uitgelegd heb over 'Van programmeertaal naar Basic', alleen het geraamte is het belangrijkste dat nodig is om te vergelijken. Als u daar aan denkt zal de GML scripttaal niet moeilijk zijn.

De sprite eigenschappen instellen.

We gaan eens verder kijken hoe we bovenstaande theorie kunnen gebruiken en of we het wel snappen als we programmeertaal Basic 'ernaast' leggen. Wat ik hieronder allemaal laat zien kan ook in een Basic versie, zoals Visual Basic .NET – de Paint event met de parameter e.

Start Game Maker en creëer een nieuwe sprite via menu Resources of kies met de rechter muisknop op Sprites: Create Sprite.

Figuur 1 laat de instellingen zien van een sprite.



Figuur 1.

Na het laden van een sprite bestand moet eerst het een en ander ingesteld worden voordat het object gemaakt kan worden. Het formulier geeft de breedte en hoogte weer van de sprite, inclusief de lege ruimte die eventueel om de afbeelding aanwezig is. Ballen, poppetjes, ruimteschepen enzovoort, hebben altijd een ruimte waardoor de instelling Transparant aangevinkt moet worden. Een plaatje die de hele rechthoek in beslag zou nemen is niet transparant. Niet transparante sprites zijn altijd solid (gevuld).

Is de 'Precise collision checking' aangevinkt dan zal precies op de botsingen gecontroleerd worden. Dit is een goede keuze bij sprites die niet solid zijn, maar de optie mag ook worden aangevinkt bij solid sprites.

De optie 'Smooth edges' is zinvol bij transparante sprites. De object instanties zullen daardoor gladder en vlotter over het scherm bewegen.

Vinkt u 'Preload texture' aan dan zullen de afbeeldingen van de sprite in het textuurgeheugen herladen worden zodra het spel wordt gestart. Het zal alleen om meer afbeeldingen gaan als de sprite een GIF sprite is.

In het frame Origin kunt u bepalen waar het snijpunt moet beginnen en of u dat snijpunt gecentreerd wilt hebben. De relatieve waarden die in de acties opgegeven worden zullen altijd vanaf het snijpunt berekend worden.

In het frame 'Bounding Box' kunt u iets soortgelijks doen als met het vorige frame. Hier geeft u echter op hoe het raam van de sprite bepaald moet worden. Hebt u al een snijpunt opgegeven en verandert u een optie in de 'Bounding Box' dan kan het zijn dat u nogmaals de Origin waarden op moet geven, bijvoorbeeld dat het gecentreerde snijpunt niet meer klopt.

Automatic Kies deze optie als u een raam wilt hebben die automatisch om de sprite afbeelding ligt. U hoeft het dan niet zelf in te stellen.

Full image Kies deze optie als u geen raam om de afbeelding wilt hebben. De eventuele ruimte wordt dan ook meegerekend.

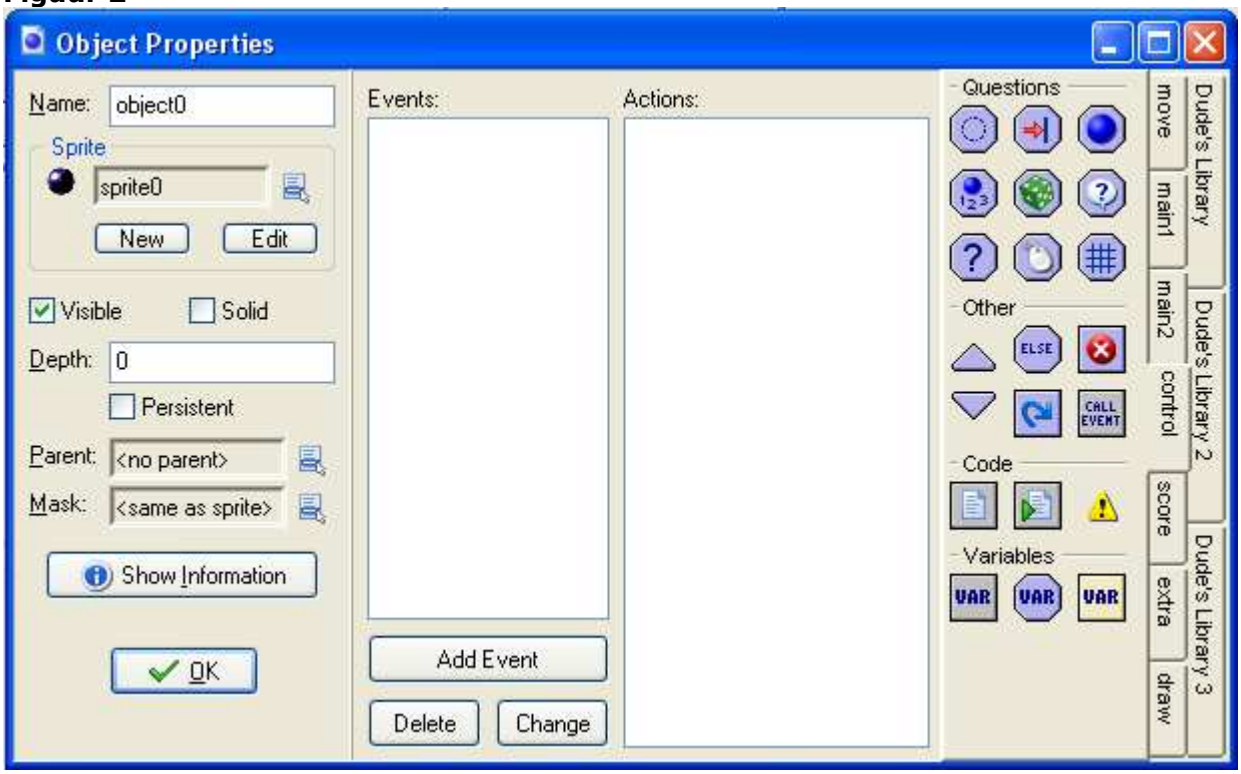
Manual Kies deze optie als u geen van de bovenste twee opties wilt. U kunt dan zelf een raam om de afbeelding instellen met de invoervakken Left, Right, Top en Bottom.

De events en acties van de objecten.

In welke Basic versie of Game editor we ook zijn, de objecten hebben altijd events waar statements in Basic en acties in Game Maker uitgevoerd moeten worden.

In Basic hoeven we niet alle events zelf te schrijven. Helaas is er een event die we graag hadden gewild, de collision event. Deze treedt op zodra twee object instanties, waarvan één in de event opgegeven wordt, elkaar raken. Als we in Basic de pictureboxen gebruiken of we maken de sprites direct, zoals dat kan in .NET met e.Graphics, moeten we zelf ervoor zorgen dat er een botsing subroutine aangeroepen wordt als twee instanties van het sprite object elkaar raken. Dit is nou het voordeel met Game Maker. Het kost minder tijd om spellen te maken en we hoeven ons niet te bekommeren om de event code.

Figuur 2



Figuur 2 laat de instellingen zien van het bal object. De sprite is gekozen.
We zien in het Sprite frame twee knoppen: New en Edit. We kunnen namelijk ook eerst een object aanmaken zonder een sprite. Door dan op New te klikken zal automatisch een nieuwe sprite behorend aan het object aangemaakt worden. Met Edit kunnen we rechtstreeks de sprite bewerken.

Als de optie Visible is aangevinkt, zal het object op het scherm zichtbaar zijn. Anders niet, maar dat betekent niet dat dan de events niet zouden werken.

Met de optie Solid bepaalt u hoe de events van het object moeten reageren. Vinkt u Solid aan, maar is de sprite echter transparant dan kan tijdens het spelen van het spel rare situaties ontstaan. Als voorbeeld: de bal kaatst na het raken van een muur niet terug. Ook transparante muren kunnen solid zijn en wel omdat muren niet altijd bewegen.

In het vak Depth kunt u de diepte aangeven van het object. Op die manier kunt u objecten met andere objecten elkaar overlappen. Het opgeven van een diepte wil niet zeggen dat u in 3D werkt. Hier gaat het alleen maar om dat u de objecten onder andere lagen kunt brengen.

Als Persistent is aangevinkt, zal het object voortbestaan tussen de rooms. Voor een balspel als break-out is deze optie niet nodig.

Met de keuzelijst Parent kunt u kiezen wat het ouderlijke object is van het object waar u mee bezig bent. In Basic kunnen we dat ook doen door een control de Parent van een andere control op te geven.
Het voordeel is dat alle objecten die de Parent hebben ook de events erven. U hoeft dus niet elke keer de events te maken. Door deze in de ouder te plaatsen kunt u elk object het laten erven.

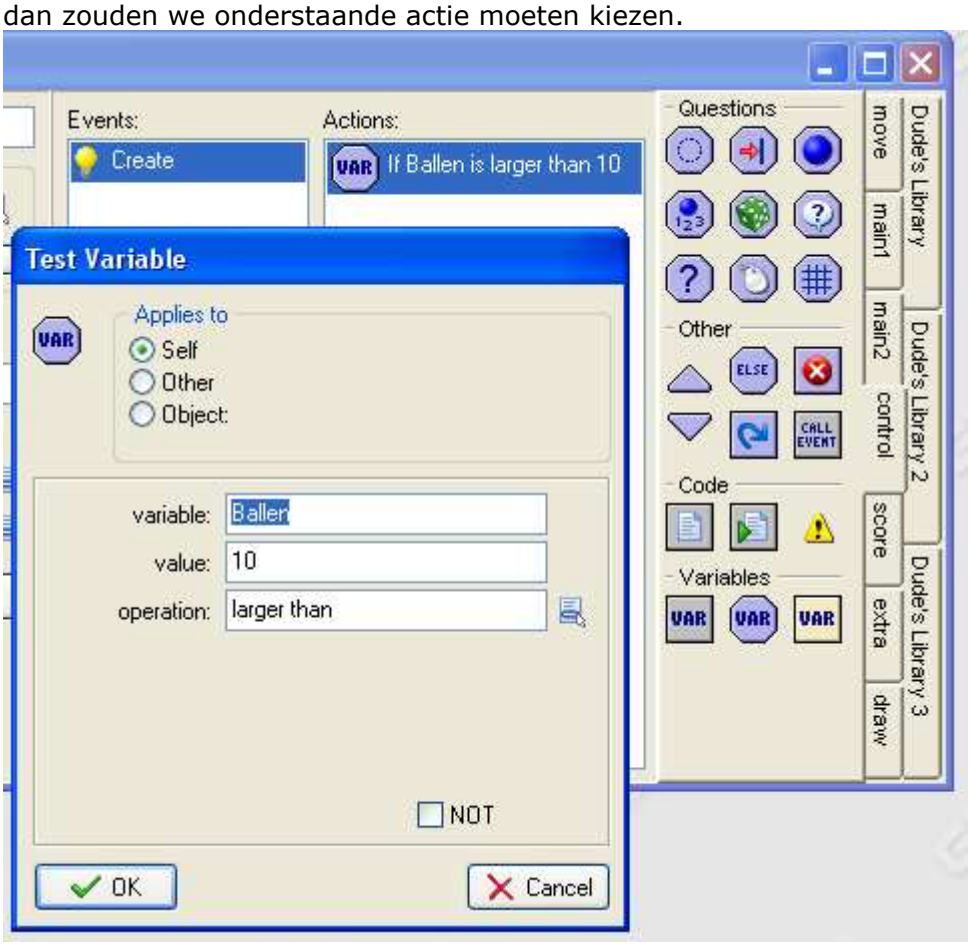
Soms kan het gebeuren dat een transparant object niet naar behoren functioneert. In veel situaties is dat het geval als het moet botsen met een solid object. Om een soortgelijke situatie te voorkomen, kunnen we het transparante object een masker geven zodat het object als een transparant-solid object functioneert.

De speler ziet het masker niet, maar de bal zal wel als een vlakje op het scherm bewegen. Er zal dan ook gereageerd worden op de botsing ter grootte van het masker, niet van de bal zelf.

In het midden kunt u events toevoegen, bewerken en verwijderen.

Aan de rechterkant ziet u de tabs. Elke tab heeft bepaalde acties. Figuur 2 laat de Control tab zien met acties waarvan er een aantal in Basic bekend voorkomen. Door een actie naar het lege vlak onder Actions te slepen, kunt u de event helemaal zonder code programmeren.

Hebben we een Basic regel als: IF ballen > 10 THEN
dan zouden we onderstaande actie moeten kiezen.



Figuur 3

De event waar de actie is ingezet is de Create event. We kunnen dat zien als het Basic sleutelwoord New. Deze treedt op zodra de instantie van het object gecreeerd wordt.
In de event staat nu één actie die hetzelfde doet als de Basic regel IF ... THEN. De acties werken onder knoppen en soms werken de knoppen ook gezamenlijk zoals bij de knoppen onder Other. Zijn er niet meer dan 10 ballen dan gebeurt er niets. We zouden dan eventueel de Else actieknop erbij kunnen plaatsen voor acties die gelden als er minder of gelijk aan 10 ballen zijn.

In het venster 'Test Variable' staan er bovenaan drie opties. De optie Self zorgt ervoor dat de eventuele opgegeven eigenschappen bij het object horen waar u mee bezig bent.
De optie Other zal alleen kunnen werken indien in de event een ander object opgegeven moest worden. Een voorbeeld is de collision event. De opgegeven eigenschappen zullen dan werken op het andere object.
De optie Object kan gekozen worden indien er in een event niks anders op zit dan een actie te laten reageren op een ander object. Ook dan gelden weer de eigenschappen voor het andere object.

Tip! Wilt u het bewerkte object samen laten reageren op het andere object, gebruik dan in de expressie het sleutelwoord Self.

Het sleutelwoord en optie Self kunnen we vergelijken met het Basic sleutelwoord Me.

Wat u verder voor opties in het venster ziet spreken voor zich. Ze werken precies zo als een IF ... THEN structuur in Basic.

U kunt ook een andere actie kiezen die ook als een IF ... THEN zal werken: het vraagteken, waar u een booleaanse conditie in kunt geven. De andere achthoekige actieknoppen werken ook als een soort IF, want die knoppen kunnen samenwerken met een Else knop.
De 1,2,3 knop is bijzonder handig om te bepalen hoeveel object instanties er nog zijn. In de actie geeft u op om welk object het gaat en geeft u op hoeveel ervan bepaald moet worden. Deze actie is zeer aan te bevelen om er voor te zorgen dat Game Maker weet wanneer de room leeg is als de bal alles vernietigd heeft. U kunt dan daarop laten reageren door het spel door te laten gaan naar de volgende room.

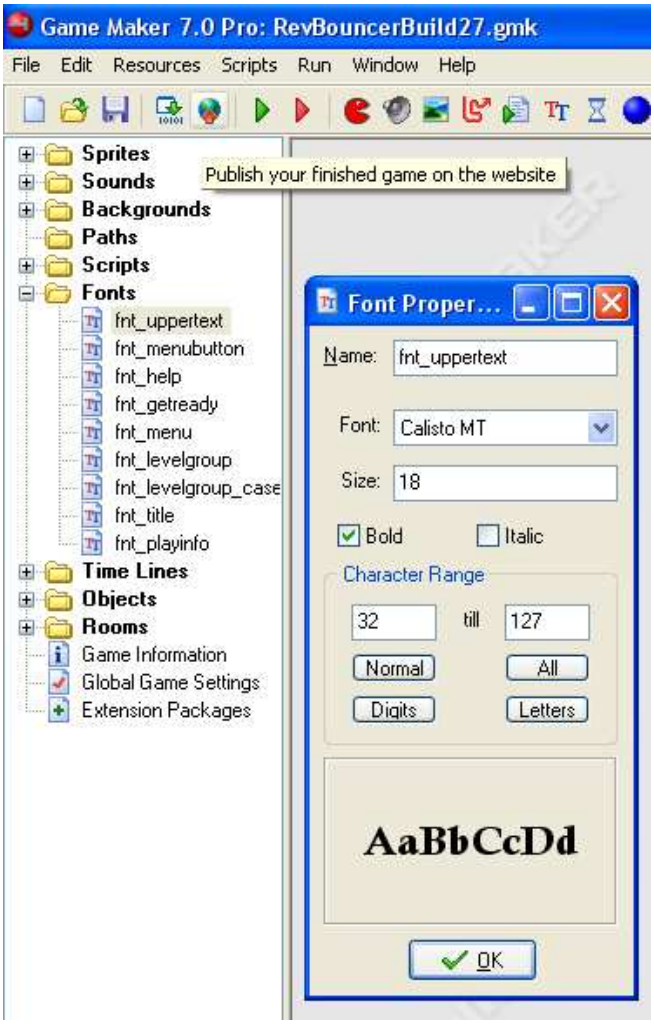
Onthoud dus dat de vierkante knoppen geen IF acties, maar instellingsacties of aanroepacties (voorgeprogrammeerde methoden) zijn.

Samenvatting.
Met Game Maker kunt u makkelijk games maken zonder gebruik van programmacode. Door middel van events en acties programmeert u de objecten door de knoppen naar het Actions venster te slepen. Elke event heeft zijn eigen Action venster
Het instellen van de sprites en de objecten moet nauwkeurig worden gedaan. De sprites moeten juist overeenkomen met de objecten. Een gekozen sprite die transparant is zal niet perfect werken in een object waarvan de optie Solid is aangevinkt.

Meer weten over Game Maker?

Op de website http://www.game-maker.nl kunt u meer vinden over Game Maker. U kunt daar voorbeelden vinden, spelonderdelen en ook de nieuwste versie van Game Maker die u gratis kunt downloaden en gebruiken.
Mark Overmars heeft ook een forum op zijn website. U kunt dan vragen stellen aan iedereen die het forum gebruikt. Zo leert u gezamenlijk meer over Game Maker. Om het forum te gebruiken, moet u eerst een account aanmaken zodat u op de site bekend bent. Het inloggen via een account is beschermt, dus uw gegevens zullen niet bekend worden. Anderen die antwoorden op uw vraag sturen zullen alleen uw inlognaam zien, niet uw normale naam.

Programmeer scripttaal GML
Zonder programmeren kunt u in Game Maker spellen maken. Toch is het niet verkeerd om wat code te gebruiken voor bijvoorbeeld muisbesturing en het weergeven van een score en een highscore. Daar is ook een actieknop van, maar die geeft geen mooie score weer voor in het spel. De GML code heeft veel methoden om tekst weer te geven.



Figuur 4
Wilt u de tekst in verschillende fonts weergeven dan moet u eerst eigen fonts aanmaken. Deze fonts zijn dan de objecten die u in de code in de **draw_set_font()** methode instelt. Het aanmaken van de fonts gaat precies zo als het aanmaken van een sprite of een sprite-object, via Resources in het menu of de rechter muisknop op de vetgedrukte tekst **Fonts**, zie Figuur 4.

Het spel dat ik zelf maak is al erg groot geworden en heeft veel objecten. Ook een aantal verschillende fonts. Meer verschillende fonts gebruiken maakt het spel leuker en gezelliger. Het is even uitzoeken voor welk doel we een bepaalde font gebruiken en wanneer we het opgeven.

In Figuur 4 is het font fnt_uppertext geopend. Hier ziet u welk font ik gekozen heb voor de bovenkant van het spel, zoals het weergeven van de score. Vandaar dat ik het object fnt_uppertext heb genoemd. Ook heb ik Bold aangevinkt zodat de weergave goed te zien is.
Als extra is er een keuze uit een karakter gebied. U kunt de knoppen Normal kiezen (letters, leestekens en cijfers), All (alle tekens uit de ascii set), Digits (alleen cijfers) of Letters (alleen letters).
Onder de instellingen ziet u de weergave van het font. U kunt dan direct zien hoe het op het scherm wordt weergegeven.

Op de volgende pagina kunt u de script bekijken die gebruik maakt van dit font.

Script: scr_draw_uppertext

```
{
    if (room == room_headwindow) then      Is de actieve room het hoofdscherm?
    {
        // teken het menu en de titel op het hoofdscherm
        draw_set_font(fnt_menu);           Instellen van het font fnt_menu.
        draw_set_color(c_yellow);          Kleur instellen voor de tekst.
        draw_text(128, 160, 'Start');
        draw_text(128, 256, 'Openen');
        draw_text(128, 352, 'Level groepen');
        draw_text(128, 448, 'Help');
        draw_text(128, 544, 'Sluiten');

        draw_set_font(fnt_title);
        draw_set_color(c_orange);
        draw_text(80, 32, 'Revence of the Bouncer');

        draw_set_font(fnt_levelgroup);
        draw_set_color(c_red);
        scr_levelgroup_cases(false);       Bepaal welke levelgroep gekozen is en geef die weer.
        draw_text(512, 224, 'Hoogste score: ' + string(global.game_hiscore));
    } else
    {
        if (room <> room_headwindow) and (room <> room_endgame) and
           (room <> room_endlevels) and (room <> room_help) then
        {
            // teken de tekst en de afbeeldingen bovenaan het spel
            draw_set_font(fnt_uppertext);
            draw_set_color(c_black);
            draw_text(8, 16, 'Score#' + string(global.game_score));
            if (global.game_score > global.game_hiscore) then      Score groter dan hiscore?
                global.game_hiscore = global.game_score;           Ja, hiscore is score.
            draw_text(174, 16, 'Hoogste score#' + string(global.game_hiscore));
            draw_text(552, 16, 'Level ' + string(global.game_level));
            draw_sprite(spr_batcounts, -1, 698, 20);
            draw_text(722, 16, string(global.game_lives));
            draw_sprite(spr_bullets, -1, 794, 20);
            draw_text(818, 16, string(global.bullets));
            draw_sprite(spr_flowers, -1, 890, 20);
            draw_text(932, 16, string(global.game_flowers));
            scr_levelgroup_cases(true);
        } else
        {
            if (room == room_endgame) or (room == room_endlevels) or
              (room == room_help) then
                scr_menubuttontext();
            if (room == room_endgame) then
              scr_playinfo('Game Over','');
            if (room == room_endlevels) then
              scr_playinfo('Gefeliciteerd!', 'Levelgroep compleet!');
        }
    }
}
```

Om van de code wat meer te begrijpen heb ik hieronder een tabel voor u met de meest opvallende onderwerpen:

Opvallende onderwerpen	**Betekenis**
Functie string()	Eigenlijk is het geen functie, maar een converteertype. Hiermee kunnen we getallen converteren naar tekst en die weergeven. Basic heeft ook zoiets als: CStr(), Str() of Str$().
Sleutelwoord global.<varnaam>	Namen die u na een global. ziet staan zijn globale variabelen. De globale variabelen zijn in alle scripts en object events bekend. U kunt ze dus ook in de actieknoppen gebruiken.

if ... [then] ... else	GML kent dezelfde operatoren en sleutelwoorden om een conditie uit te voeren met een if ... then ... else net als we in Basic kennen. GML kent echter niet de End If om een codeblok na Then of Else af te sluiten. Daar zijn in GML de accolades voor. Bovendien is het sleutelwoord then niet verplicht. Ook de operatoren and, or en not kunnen werken als taal C symbolen &&, || en !.
Extra haakjes	Het ligt eraan in welke Basic versie u werkt. Sommige versies ondersteunen meer haakjes om bijvoorbeeld parameters op te geven bij methoden. In de script ziet u ook haakjes in een if ... then. Dat mag in Basic ook, maar in GML zijn ze verplicht.
Het weergeven van tekst	In GML wordt dat niet gedaan met de "...", maar met de '...'.
Het hekje #	Als deze in de tekst wordt opgegeven zal de tekst die er volgt op de volgende regel worden geplaatst, maar dan wel op dezelfde x-coördinaat. De y-coördinaat zal door Game Maker automatisch worden bepaald. In Basic gebruiken we daarvoor de constante vbCrLf.

U kunt ook op de website meer vinden over GML. Er is een online handleiding waar u dit allemaal kunt vinden.

Marco Kurvers

Websites programmeren met Crimson (1).

Een paar keer heb ik het al over de Crimson Editor gehad. Wat voor voordelen en nadelen heeft de editor? Voor programmeurs is het een voordeel, maar voor wie liever drag en drop muisbesturing wil is het een nadeel. Crimson heeft geen IDE scherm om de controls erop te plaatsen. Het moet allemaal gedaan worden in een code editor.

Uw Crimson Editor project.

In Crimson kunt u de pagina's maken en samenbrengen in een project. Dat is handig als u uw website wilt uploaden. In het Bestand menu kunt u FTP kiezen en uw FTP gegevens instellen. U hoeft geen FTP programma te gebruiken om uw website klaar te maken.
Wanneer u de pagina's samen wilt brengen moet u er aan denken dat u de bestanden ten eerste in de website map opslaat en ten tweede alle bestanden die u nodig heeft in Crimson open hebt staan.
Toch kan het gebeuren dat de automatische FTP instelling van Crimson niet goed werkt. Raadpleeg uw provider van waar uw website vandaan komt. U kunt ook het gratis programma WS_FTP downloaden.

Een Crimson project kan uit de volgende onderdelen bestaan:

html bestanden	Dit zijn de bestanden met (x)html code. Voor goede webpagina's is html code niet voldoende. Elke pagina begint daarom ook met de DOCTYPE regel.
css bestanden	CSS code (Cascading Style Sheets) bevat meer opmaak functionaliteit dan html heeft. De code kan in een xhtml pagina worden geprogrammeerd, maar kan ook apart in een bestand worden geplaatst. Zeer handig als meerdere pagina's dezelfde opmaak nodig hebben.
BASIC	Crimson ondersteunt VBScript code. Subroutines en functies kunnen voor meer functionaliteit zorgen en verkort daardoor de code. Het is ook mogelijk event methoden te maken, bijvoorbeeld een onload event die achter een body geplaatst kan worden. Net als CSS mag VBScript code in dezelfde pagina worden geschreven, maar u kunt ook aparte BAS bestanden gebruiken.
Java	Wilt u liever BASIC gebruiken voor Windows applicaties dan kunt u ook Java gebruiken in Crimson. Voor JavaScript gelden dezelfde regels als bij VBScript. Denk erom dat u een scripttaal gebruikt die veel op C++ lijkt.
Andere scripts zoals PHP enzovoort	Via het menu Document – Syntax Type in Crimson kunt u een hele lijst vinden met meerdere programmeertalen met de bovenstaande talen erbij. Andere twee bekende talen zijn PHP en Pascal. PHP lijkt op Pascal, maar is meer gericht als een Pascal script vergelijkbaar met BASIC en VBScript. We kunnen in Crimson dus zowel PHP kiezen als Pascal, maar voor BASIC is alleen VBScript beschikbaar.
Customize...	Is er een andere programmeertaal waarvan een scriptcode bestaat? Crimson geeft een mogelijkheid om zelf uw eigen scriptcode in te voeren om het programmeren naar uw eigen hand te zetten.

index.html	Dit bestand is het hoofdbestand van elke website. Elk project heeft dus zo'n index bestand. Als we aan een project beginnen, maken we dus eerst deze pagina, ook al is het niet verplicht. Sterker nog, elke pagina waar we mee beginnen is het sjabloon dat we nodig hebben maar één sjabloon moet dan wel de index pagina zijn. Zonder sjabloon zouden we geen xhtml website kunnen bouwen als we functionaliteit willen gebruiken uit de gekozen scripts.

Het eerste sjabloon.

Het eerste sjabloon moet de index pagina zijn om een website te kunnen uploaden. Waarom hebben we een index pagina nodig en hoe maken we die?
Op internet is het wel te vinden door te Googlen naar 'Index pagina maken'. Ook in boeken staat het erin hoe de index pagina wordt gemaakt. Voor u heb ik goed nieuws en daar is dit blad ook voor bedoeld: hieronder staat het geraamte van de index. Als u onderstaande geraamte wilt overnemen, bewaar het dan eerst als 'sjabloon.htm'. Hebt u Crimson Editor gedownload kies dan het menu Bestand – Save As...
Kies het type 'html' en voer als bestandsnaam in 'sjabloon', zonder type.

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Geef hier uw titel</title>
    </head>
    <body>
        <h1>Welkom bij BASIC IG.</h1>
        <h2>Alles over ontwikkelen, ontwerpen en logica.</h2>
    </body>
</html>
```

Degenen die al eens html code hebben gebruikt zullen vooral de eerste regel vreemd vinden. Die regel, beginnend bij de tag <!DOCTYPE zorgt ervoor dat de juiste xhtml versie ingesteld wordt. De versie moet als een html documenttype openbaar opgehaald worden.

Als u bovenstaand sjabloon overneemt, laat dan de tags h1 en h2 weg. U hebt eerst een leeg sjabloon nodig om dit daarna verder op te bouwen. Elke keer wanneer u met een lege pagina wilt beginnen, opent u eerst het sjabloon.

In de tag title geeft u een duidelijke titel die bovenaan op de blauwe balk weergegeven zal worden.

Als u met uw website begint, is een welkomstregel met daaronder de omschrijving van uw website wel zo aardig. U ziet de tag <h1>. Die tag heeft een groter font dan de tag <h2>. Er zijn nog meer hn tags die van 1 tot en met 6 gaan, maar meestal is h1, h2 en h3 al voldoende. De h-tekstregels zouden anders te klein worden en daar wordt uw website niet mooier op.

Als u het voorbeeld wilt testen kunt u in Crimson naar het menu Tools gaan en klikken op View in Browser of u drukt op ALT samen met toets B.

In de volgende nieuwsbrief laat ik u meer code zien en leg ik u uit hoe we CSS opmaakcode aan de pagina's kunnen koppelen.

Marco Kurvers

De listings in de kwartaalbestanden.

Om uit de listings geen fouten te maken, staat hieronder nogmaals hoe u de listings in de kwartaalbestanden kunt gebruiken.
Als u de programmavoorbeelden, de listings genoemd, wilt gebruiken dan kunt u de tekst kopiëren die in de kwartaalbestanden staan.
Alles meteen kopiëren en in een codevenster plakken kan werken, maar houd er rekening mee dat delen van de code soms in de voorbeelden worden afgekort. De code die niet nodig is schrijf ik met drie punten '...' om de regels af te korten, zodat toch de listing netjes uitgelijnd is en goed leesbaar blijft.
Voor de Visual Basic gebruikers heb ik daarom de volgende tips:

- maak niet zelf de private event subs aan maar kies de events die voor de code nodig zijn door in de object properties op de event te dubbelklikken;
- kopieer de code tussen de regels **Private Sub** en **End Sub** en plak het in het codevenster in de juiste Private Sub event;
- er kunnen in de programmavoorbeelden ook normale subroutines staan (geen events), die kunt u helemaal kopiëren en in het codevenster plakken;
- staat er geen **Private Sub** en **End Sub** dan is het een één listing die niet uit meerdere subroutines bestaat en kunt u de code helemaal kopiëren en plakken in de juiste Private Sub event.

Let op! Maakt de code gebruik van controls, plaats dan eerst op het formulier de controls zoals u op de voorbeelden ziet. Pas dan kunt u de code kopiëren, plakken en uitvoeren. Anders zullen er fouten ontstaan, omdat de controls nog niet herkend worden.

Heeft u niet de Visual Basic .NET versie, gebruik dan uw BASIC versie. Pas de code aan voor de BASIC versie als het niet werkt. Komt u er nog niet uit, raadpleeg dan eventueel de converteerlijsten die u op de BASIC IG website kunt vinden of stuur een bericht naar mij. Geef voldoende informatie over het probleem zodat ik de juiste oplossing naar u terug kan sturen. Heeft de oplossing een voordeel voor de converteerlijsten dan pas ik die gelijk aan. In de volgende nieuwsbrief komt er een nieuwe appendix converteerlijst.

Cursussen

Qbasic: Cursus, lesmateriaal en voorbeelden op CD-ROM € 6,00 voor leden. Niet leden € 10,00.

QuickBasic: Cursusboek en het lesvoorbeeld op diskette, € 11,00 voor leden. Niet leden € 13,50

Visual Basic 6.0: Cursus, lesmateriaal en voorbeelden op CD-ROM, € 6,00 voor leden. Niet leden € 10,00

Basiscursus voor senioren, Windows 95/98,
Word 97 en internet voor senioren, (geen diskette). € 11,00 voor leden. Niet leden € 13,50

Computercursus voor iedereen: tekstverwerking met Office en eventueel met VBA, Internet en program-meertalen, 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.

Software

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

Hoe te bestellen

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

ABN-AMRO nummer 49.57.40.314
HCC BASIC ig
Haarlem

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

Alle prijzen zijn inclusief verzendkosten voor Nederland en België.

Vraagbaken

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

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