Basic Bulletin

22ste jaargang oktober 2015 Nummer 3

HCC BASIC interessegroep

Inhoud

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Jan van der Linden	071-3413679	voorz@basic-gg.hcc.nl
Secretaris	Gordon Rahman Tobias Asserstraat 6 2037 JA Haarlem	023-5334881	secr@basic-gg.hcc.nl
Penningmeester	Piet Boere	0348-473115	penm@basic-gg.hcc.nl
Bestuurslid	Titus Krijgsman	075-6145458	t.krijgsman8@upcmail.nl
Redacteur	M.A. Kurvers Schaapsveld 46 3773 ZJ Barneveld	06-30896598	m.a.kurvers@live.nl
Ledenadministratie	Fred Luchsinger	0318-571187	f.luchsinger@kader.hcc.nl
Webmaster	Jan van der Linden	071-3413679	j.vd.linden@kader.hcc.nl

http://www.basic.hcc.nl

Redactioneel

Hoe dan ook, foutjes maken we altijd. Het hoeven niet eens zulke grote fouten te zijn, maar zelfs de simpelste foutjes worden vaak gezien als de grootste bedreigingen. Als we het slim aanpakken, dan hoeven we een mug niet als een olifant te zien.

Bent u gewend te programmeren in Windows Applicatie projecten of in een Basic dialect zonder IDE? Wilt u wel eens in een andere programmeerwereld stappen? De game wereld is een mooie uitdaging en dankzij XNA Game Studio is het nare gebruik van DirectX en OpenGL (en de API's) nu verleden tijd.

Marco Kurvers

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

Formattering en conversie

BBC BASIC omvat niet de instructie **PRINT USING** die, in veel dialecten van BASIC, een handige manier biedt voor het opmaken van numerieke uitvoer. Hoewel gelijkwaardige effecten kunnen worden verkregen met behulp van het **@%** opmaak besturingselement variabele, eventueel in combinatie met string-manipulatie functies (b.v. MID$), zijn ze minder eenvoudig te bereiken.

BBC BASIC mist ook de **UPPER$** (of **UCASE$**) en **LOWER$** (of **LCASE$**) die in sommige BASIC dialecten een handige manier bieden om tekst te converteren naar hoofdletters (kapitalen) of kleine letters.

De **FNUSING** library biedt vervangingen voor deze operaties. Het moet van uw programma geladen worden met gebruik van onderstaand commando:

```
INSTALL @lib$+"FNUSING"
```

Als alternatief, aangezien de functies vrij kort zijn, wilt u ze misschien in uw eigen programma nemen (gebruik de opdracht **Insert** in het **File** menu).

De functies zijn:

- FNusing
- FNlower
- FNupper

FNusing(format$,value)

De **FNusing** functie heeft twee parameters, een format string en een numerieke waarde. Normaal wordt het gebruikt met een **PRINT** statement, zie onderstaande context:

```
PRINT FNusing(fmt1$,val1) FNusing(fmt2$,val2) .....
```

Een significant verschil uit de conventionele PRINT USING statement is dat elk format string gerefereerd kan worden met één numerieke waarde, dus u moet **FNusing** voor elke waarde aanroepen, die u voor de uitvoer wilt hebben.

De format string is een letterlijke of variabele reeks met speciale opmaaktekens, die hieronder worden opgesomd:

Het hash-teken wordt gebruikt om de positie van een cijfer te geven. Cijfer posities zijn altijd gevuld: als het getal minder cijfers heeft dan de opgegeven posities, zal het rechts uitgelijnde voorafgaan door spaties. Een decimale punt mag in elke positie worden ingevoegd in het veld en de getallen worden afgerond wanneer dat nodig is, bijvoorbeeld:

```
PRINT FNusing("##.##",.78)
 0.78
PRINT FNusing("###.##",987.654)
987.65
```

+ Een plusteken aan het begin of eind van een format veld zorgt ervoor dat het teken van het getal (plus of min) vooraan of achteraan geprint wordt, bijvoorbeeld:

```
PRINT FNusing("+###.##",2.4)
  +2.40
PRINT FNusing("##.##+",55.678)
55.68+
PRINT FNusing("##.##+",-3)
 3.00-
```

- Een minteken aan het eind van het format veld zorgt voor negatieve getallen die met een afsluitend minteken geprint worden, bijvoorbeeld:

```
PRINT FNusing("##.##-",-68.95)
68.95-
PRINT FNusing("###.##-",-7)
  7.00-
```

** Een dubbele asterisk aan het begin van het format veld zorgt ervoor dat de overige spaties gevuld worden met asterisken. De ** zorgt ook voor twee meer cijferposities, bijvoorbeeld:

```
PRINT FNusing("**#.#",12.39)
*12.4
PRINT FNusing("**##.##",-0.9)
**-0.90
```

$$ Een dubbele dollar (of pond) teken aan het begin van het format veld zorgt ervoor dat een dollar (of pond) teken direct aan de linkerkant van het geformatteerde getal geprint wordt. De $$ zorgt ook voor twee meer cijferposities, waarvan één het huidige symbool is, bijvoorbeeld:

```
PRINT FNusing("$$###.##",45.67)
  $45.67
PRINT FNusing("ff###.##",123.45)
 £123.45
```

**$ Een **$ (of **£) aan het begin van het format veld combineert de effecten van de vorige twee formatteringen. Overige spaties worden gevuld met asterisken en een dollar (of pond) teken, die geprint worden voor het getal. **$ zorgt voor drie meer cijferposities, waarvan één het huidige symbool is, bijvoorbeeld:

```
PRINT FNusing("**$##.##",2.34)
***$2.34
```

, Een komma aan de linkerkant van de decimale punt in de format string zorgt ervoor dat een komma tussen elke derde cijfer wordt geprint voor de decimale punt, bijvoorbeeld:

```
PRINT FNusing("#,###.##",1234.5)
1,234.50
PRINT FNusing("##,###,###",1E6)
 1,000,000
```

^^^^ Vier invoegtekens mogen worden geplaatst achter de cijfer karakters om voor een exponentieel format te zorgen. De vier invoegtekens maken extra spaties voor “E-xx” die geprint wordt, bijvoorbeeld:

```
PRINT FNusing("##.##^^^^",234.56)
 2.35E2
PRINT FNusing("##.##^^^^",1E-30)
```

```
1.00E-30
```

Als de format string karakters anders zijn dan de hierboven gegeven, worden ze *letterlijk* in de uitvoer-string opgenomen, bijvoorbeeld:

```
PRINT FNusing("Price ££#.## including VAT",29.99)
Price £29.99 including VAT
```

Als het getal niet in het format ingedeeld kan worden, zullen vraagtekens worden geprint:

```
PRINT FNusing("##.##",123)
?????
```

FNlower(string$)

De **FNlower** functie neemt een gegeven string parameter, converteert de hoofdletters (inclusief A tot Z), als er enkele zijn, in kleine letters (a tot z) en retourneert deze.

```
PRINT FNlower("The Quick Brown Fox")
the quick brown fox
```

FNupper(string$)
De **FNupper** functie neemt een gegeven string parameter, converteert de kleine letters (inclusief a tot z), als er enkele zijn, in hoofdletters (A tot Z) en retourneert deze.

```
PRINT FNupper("The Quick Brown Fox")
THE QUICK BROWN FOX
```

In de volgende Bulletin komt de library *Multiple Document Interface*. Dit onderwerp zal erg uitgebreid zijn, daarom houd ik het hierbij. Ook het onderwerp over *Popup- en sub-menu's toevoegen* zal van de partij zijn. Waarschijnlijk zal het in meerdere delen komen.

XNA Game Studio 4.0, een andere programmeerwereld.

Spellen en grafische programma's zijn bijzonder. Deze programma's hebben een uitvoer die op een andere manier bestuurd worden. Maar als we een Windows programma maken, doen we toch wat grafisch. De vensters, het menu, de werkbalken en zelfs alle controls die we gebruiken bestaan uit grafische onderdelen. We kunnen ze echter niet grafisch beschouwen, omdat het onderdelen zijn die door Windows beheerd worden en niet via de grafische kaart. Ook een shape control, waar we lijnen en rechthoeken mee kunnen plaatsen (en wel *plaatsen*, dus niet tekenen), worden door Windows beheerd.

We kunnen wel tekenen met LINE en CIRCLE statements in sommige Basic dialecten, maar zonder een Windows venster hebben de statements geen vaste plek onder hun voeten. Windows vereist met gewone tekencommando's dat we een venster moeten gebruiken. Is het een Basic dialect zonder een Windows venster, dan zal toch een venster worden gebruikt: het console venster.

XNA met Visual C# en Visual Basic
Voor Visual Studio 2010 kan XNA in het project worden ingevoegd door gebruik van een sjabloon. Deze sjabloon is kant-en-klaar, althans voor Visual C#. Het XNA Game Studio Framework bestaat uit *references*, maar ook het .NET Framework bestaat uit references. Het enige verschil is dat het .NET Framework een standaard library is die altijd aanwezig is als Visual Studio gestart wordt. Bij het ope-

nen van een Windows project zal meteen gebruik worden gemaakt van de library. De references van .NET hoeven niet ingevoegd te worden. Bij XNA is dat in Visual Basic wel het geval.

De sjabloon, het verschil tussen Visual C# en Visual Basic
In Visual C# staan de sjablonen in de lijst van de templates, zie figuur 1. Figuur 1 laat de onderdelen zien die voor het .NET Framework bestemd is, samen met de onderdelen voor het XNA Game Studio Framework.

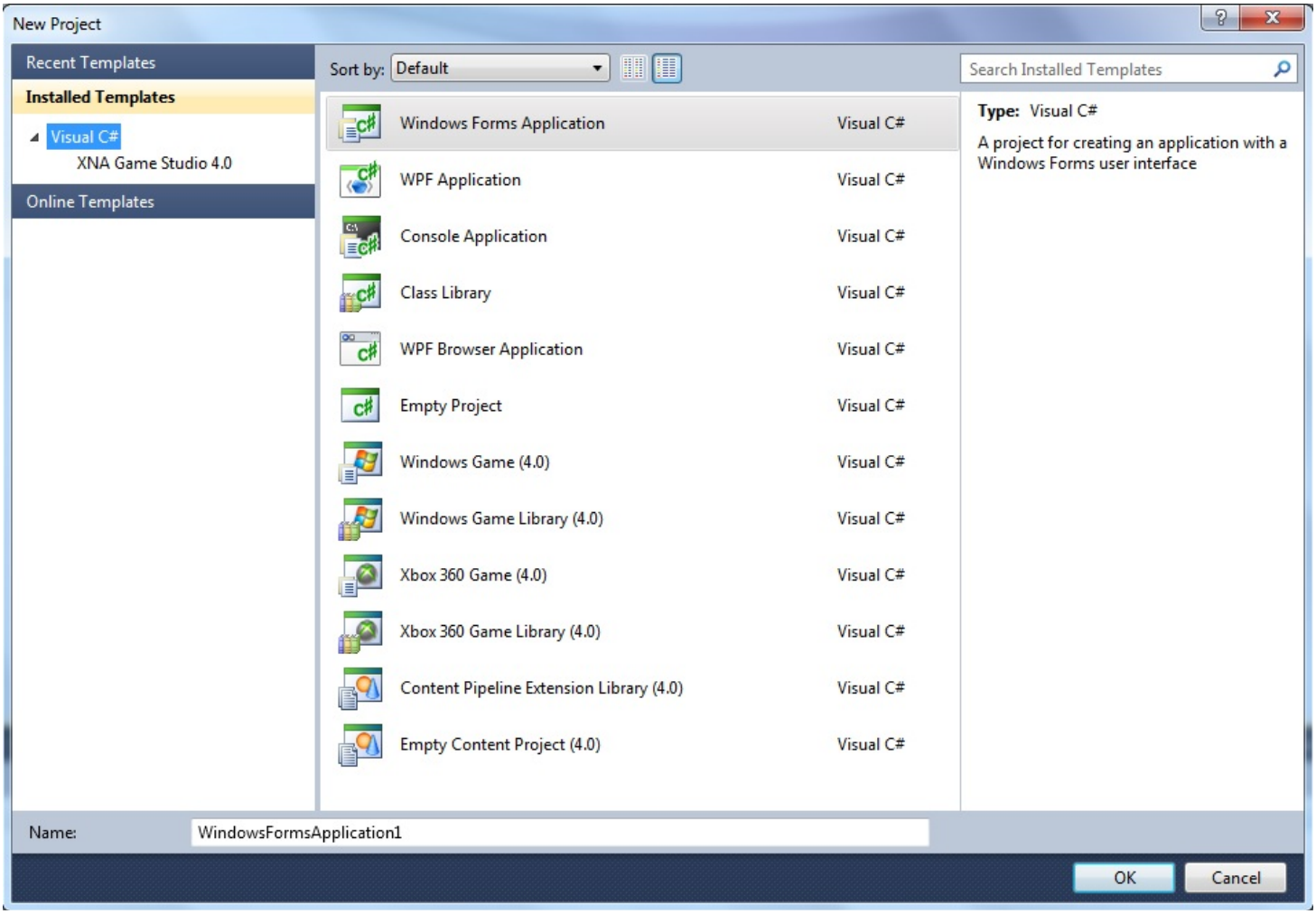


Fig.1 De onderdelen voor de standaard Windows applicatie sjabloon en voor de XNA sjabloon van Visual C#.

Ook al zijn het twee verschillende library's, toch hebben ze met elkaar gemeen. Dat heeft vooral te maken als ik het straks over de *Content Pipeline* ga hebben. Een solution onderdeel dat helaas in Visual Basic niet ondersteund wordt.

Hieronder is figuur 2 te zien met Visual Basic, maar zonder de XNA Game Studio sjabloon. Visual Basic heeft een nadeel dat er geen kant-en-klare sjabloon is ontwikkeld, maar dat betekent niet dat we XNA niet zouden kunnen gebruiken in Visual Basic. Wat we moeten doen is de sjabloon van Visual C# als voorbeeld houden en die namaken in een Visual Basic project. Zie de Appendix met de twee complete sjablonen.

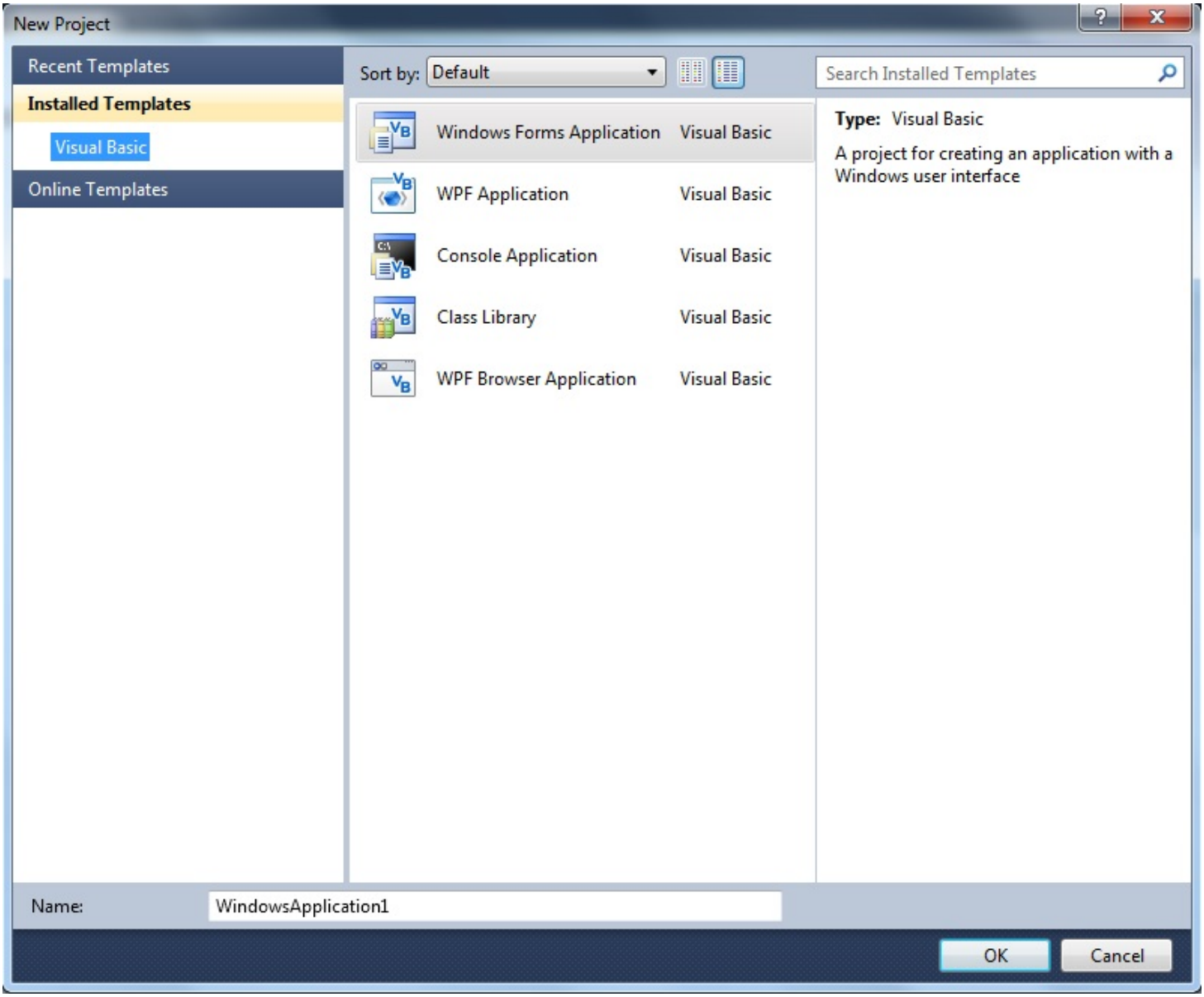
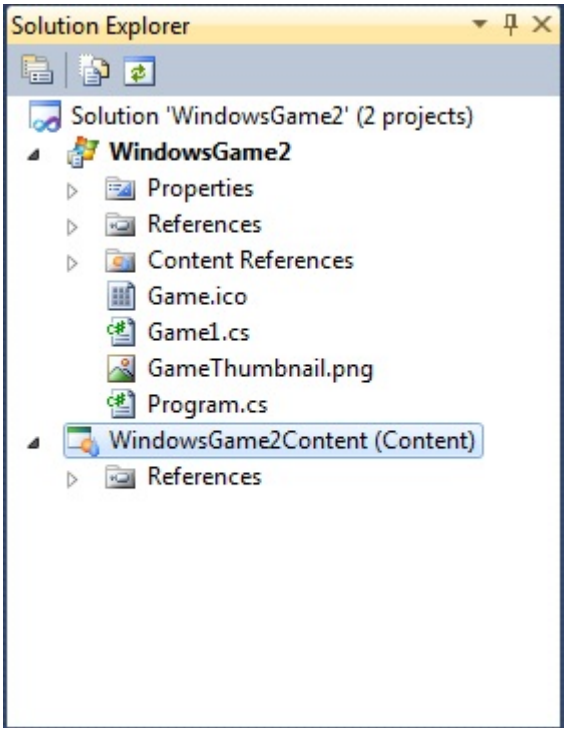


Fig. 2 Een nieuw project kiezen in Visual Basic, zonder een XNA sjabloon.

Ook al zouden we XNA Game Studio 4.0 downloaden van Internet en installeren op de computer, toch zal deze niet in Visual Basic verschijnen. Visual C# ziet alles wel van de library. De library is van Microsoft en is gratis te downloaden. Lijkt het u moeilijk om XNA te proberen? Geen probleem, ik zal u uitleggen hoe we dezelfde solution met sjabloon moeten maken in Visual Basic. Op Internet zijn heel veel tips, documentaties en voorbeelden te vinden. Een zeer goede documentatie met alles over XNA Game Studio is **RB Withaker**. Als u die naam intoetst, zult u de website wel kunnen vinden. De meeste documentaties, en ook van RB Withaker, worden uitgelegd voor Visual C#, maar als ik u laat zien hoe we ermee om moeten gaan in Visual Basic, zal het een stuk makkelijker worden om het te converteren in Basic taal.

De solution in Visual C#



De solution in Visual C# bestaat uit 2 projecten. Het eerste project is het eigenlijke game project, maar het tweede project bevat alleen maar inhoud die te maken heeft met de *Content Pipeline*. Zoals ik hierboven al gezegd heb, ondersteunt Visual Basic het tweede project niet. Het is gewoon onmogelijk om hier inhoud, zoals textures en modellen, in een tweede Visual Basic project te stoppen.

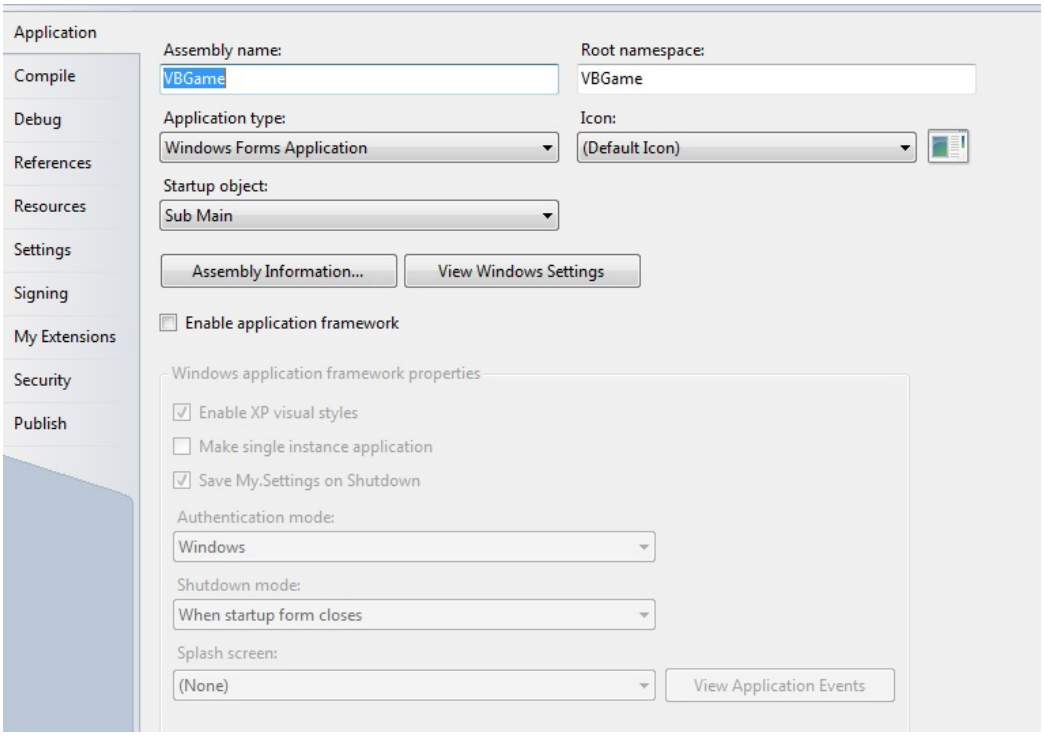
Jammer en helaas, maar er is toch een oplossing om inhoud in het game project in te laden. Daarover straks meer.

Het game project heeft twee klassen, de **Game1.cs** klasse en de **Program.cs** klasse. De Program zorgt ervoor dat de Game1 klasse gestart wordt. Daarna gebeurt alles alleen maar in de Game1 klasse als een looping die net zolang duurt totdat de speler het spel beëindigd. Dit werkt precies zo als in Visual Basic, maar de vraag is alleen nog: Waar vandaan halen we het project?

Beginnen met XNA in Visual Basic

Start Visual Basic 2010, zodat u figuur 2 voor u krijgt. Maak een nieuw project door te klikken op **Windows Forms Application**. Ook al hebben we niks te maken met de IDE van Visual Basic, het is de enige manier om met een project te beginnen. Geef het een naam, bijvoorbeeld **VBGame**. Nadat uw project gemaakt is, voeg dan een nieuwe module toe. Noem de module **mdlMain**. Maak de **Main** subroutine in de module, zoals hieronder:

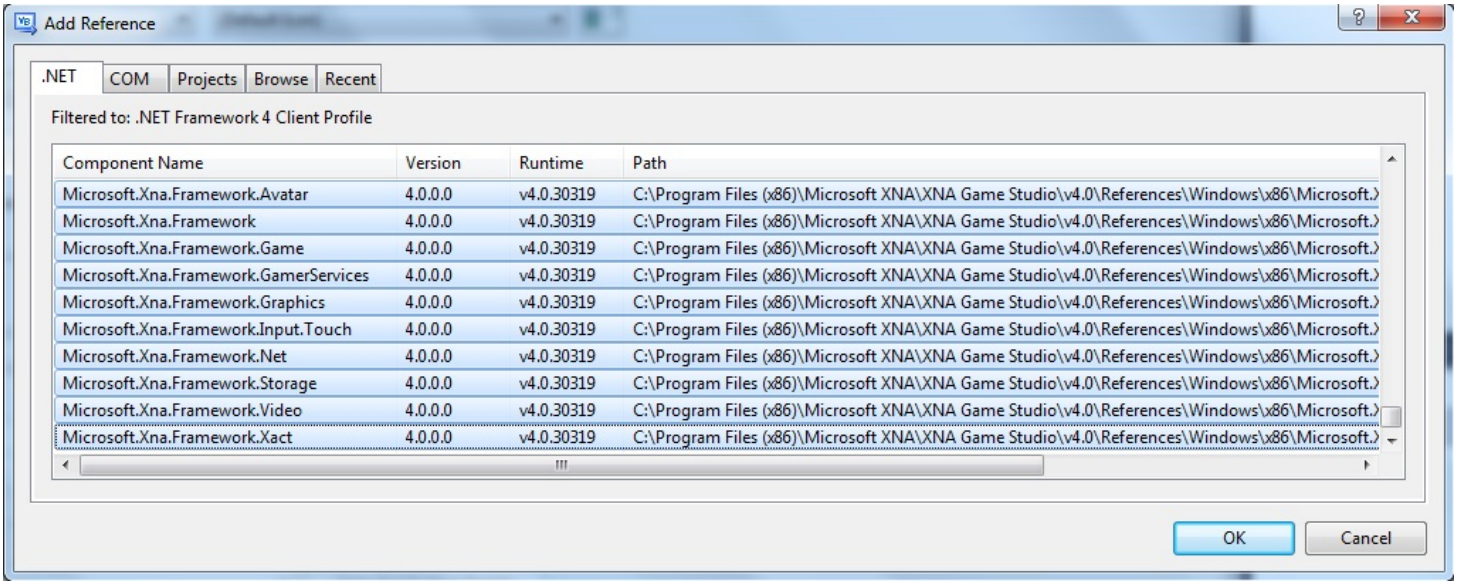
```
Module mdlMain
      Sub Main()

      End Sub
End Module
```



Nu we dit gemaakt hebben, moeten we het programma laten starten vanuit deze subroutine. Klik op het menu **Project** en kies onderaan **VBGame Properties**.

Het venster, als deze hier, verschijnt. Zorg ervoor dat u eerst het vinkje **Enable application framework** uitschakelt, zodat u als Startup object **Sub Main** kunt kiezen. Nu dit gedaan is, kunt u **Form1.vb** uit uw project verwijderen. Deze is namelijk niet nodig.

Ga nogmaals naar het Project menu en kies **Add reference…**. Een venster, zoals hieronder, verschijnt. Indien u XNA Game Studio 4.0 hebt gedownload, kunt u de namen in de lijst vinden. Selecteer ze allemaal.



Klik op **OK**.

Nu we dat gedaan hebben, zijn we daarmee klaar. Als volgende voegen we een nieuwe klasse toe. In het Project menu kiest u **Add Class…**. Noem deze **Game1**. Voeg de klasse toe en maak hem *erfbaar*, dat wil zeggen dat alles van de ouderlijke klasse (parent class) door de nieuwe klasse Game1 geërfd zal worden. Onderstaande code laat zien hoe de klasse eruit moet zien:

```
Public Class Game1
     Inherits Microsoft.Xna.Framework.Game

End Class
```

Nu moeten de verwijzingen naar XNA toegevoegd worden, zoals hieronder:

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports Microsoft.Xna.Framework
Imports Microsoft.Xna.Framework.Audio
Imports Microsoft.Xna.Framework.Content
Imports Microsoft.Xna.Framework.GamerServices
Imports Microsoft.Xna.Framework.Graphics
Imports Microsoft.Xna.Framework.Input
Imports Microsoft.Xna.Framework.Media

Public Class Game1
     Inherits Microsoft.Xna.Framework.Game

End Class
```

Nu moeten we de basis methoden, zoals Initialize, Draw…, toevoegen. Deze methoden bestaan al in de ouder klasse, ze hoeven alleen nog overgeërfd te worden met eventuele nieuwe inhoud.

Voeg onder de **Inherits** regel de twee **Dim** regels toe:

```
Dim graphics As GraphicsDeviceManager
Dim spriteBatch As SpriteBatch
```

GraphicsDeviceManager ondersteunt de grafische kaart. Het graphics object zal eigenschappen hebben, zoals het bepalen van de beeldschermresolutie en of het beeld in een venster of voluit getoond moet worden.
SpriteBatch is een belangrijke klasse waarmee we inhoud op het scherm kunnen projecteren. De manier hoe een sprite geprojecteerd wordt, heet ook wel *renderen*. Ook een model wordt gerendeerd zoals dat met een sprite wordt gedaan, maar er zijn wel verschillen tussen sprites en modellen.

Het graphics object moeten we nog creëren. Dat moet gedaan worden in de constructor van de Game1 klasse. Maak de constructor onder de Dim regels:

```
Public Sub New()
     graphics = New GraphicsDeviceManager(Me)

End Sub
```

Nu moeten de *overridable* methoden gemaakt worden. Overridable betekent dat ze *overerfbaar* zijn. Volg elke methode op de juiste manier zoals hieronder. U kunt eventueel de commentaarregels er ook bij zetten.

```
Protected Overrides Sub Initialize()
     'TODO: Voeg uw initialisatie logica hier
     MyBase.Initialize()
End Sub

Protected Overrides Sub LoadContent()
     'Creëer een nieuwe spriteBatch die gebruikt kan worden voor het
     'tekenen van de textures.
     spriteBatch = New SpriteBatch(GraphicsDevice)
     'TODO: Laad hier uw inhoud in

     MyBase.LoadContent()
End Sub

Protected Overrides Sub UnloadContent()
     'TODO: Ontlaad elke inhoud hier, maar niet van de ContentManager
     MyBase.UnloadContent()
End Sub

Protected Overrides Sub Update( _
                    ByVal gameTime As Microsoft.Xna.Framework.GameTime)
     'Staat toe het spel te beëindigen.
     If GamePad.GetState(PlayerIndex.One).Buttons.Back = _
               ButtonState.Pressed Then
          Me.Exit()
     End If
     'TODO: Voeg hier uw update logica hier

     MyBase.Update(gameTime)
End Sub
```

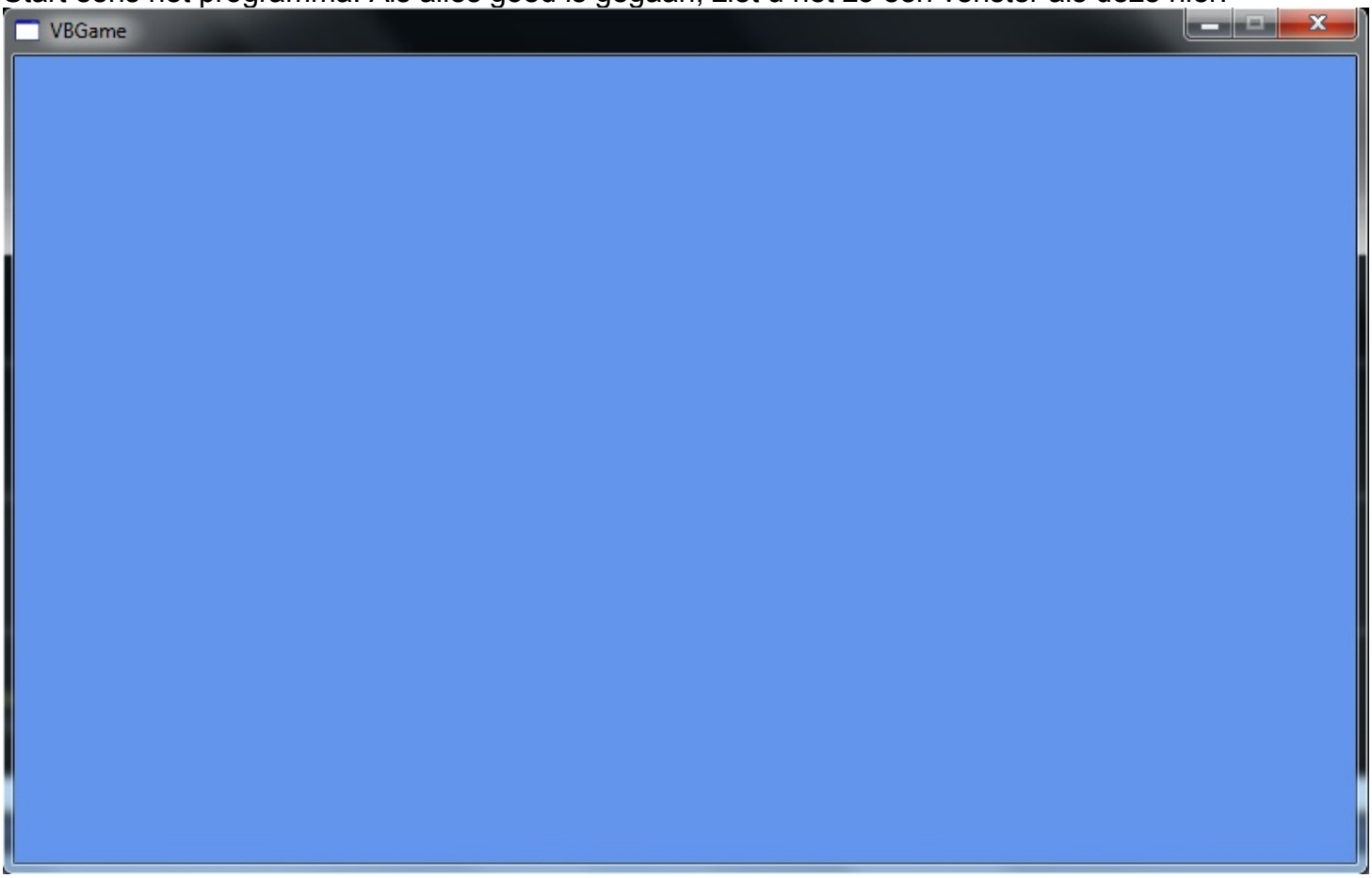
```
Protected Overrides Sub Draw( _
                    ByVal gameTime As Microsoft.Xna.Framework.GameTime)
     GraphicsDevice.Clear(Color.CornflowerBlue)
     'TODO: Voeg hier uw tekencode hier
     MyBase.Draw(gameTime)
End Sub
```

Nu bovenstaande code is gedaan, moeten we alleen nog ervoor zorgen dat we het spel kunnen starten. Daar moet de Main subroutine voor zorgen. Ga naar de Main in de module mdlMain en voeg de rest van onderstaande code toe:

```
Module mdlMain
     Sub Main()
          Using game As New Game1
               game.Run()
          End Using
     End Sub
End Module
```

Het sleutelwoord **Using** lijkt een vreemd Basic statement, althans op deze manier. Normaal kennen we deze voor de formattering in een Print statement.
U ziet dat het in deze module net zo werkt als het starten van een Windows venster, waar we echter gebruik maken van het Application object met een gedefinieerd venster object.

Start eens het programma. Als alles goed is gegaan, ziet u net zo een venster als deze hier.



Er lijkt helemaal niets te gebeuren zonder inhoud, maar dat is niet waar. Vele malen in een seconde worden de methoden Update() en Draw() uitgevoerd. In de Draw() methode wordt dus ook elke keer het scherm schoongeveegd en een blauwe bloemkleur erop geschilderd. Deze snelheid noemen we

ook wel de *rate*. Als we een spel maken, kunnen we de snelheid regelen met de doorgegeven parameter *gameTime*.

De Content Pipeline
We hebben niet veel aan een leeg scherm, daarom gaan we een sprite op het scherm tekenen. Voordat ik hiermee verder wil gaan in Visual Basic, wil ik uitleggen hoe de Content Pipeline werkt. De Content Pipeline wordt niet door Visual Basic ondersteund, maar toch heeft het er wat mee te maken als we de LoadContent() methode gaan gebruiken. Daarom laat ik u zien hoe het werkt in Visual C#.

Zoals ik eerder verteld heb, moet al het inhoud toegevoegd worden in het Content project. Dit project is de Content Pipeline, zoals u eerder hebt gezien bij de Solution afbeelding van Visual C#.

In Visual C# moeten we met de rechter muisknop klikken op **MyGameContent(Content)**. Verwar het niet met uw gamenaam VBGame, omdat het eventjes niet om Visual Basic gaat. Kies **Add Existing Item…**.
Visual C# vraagt om een bestand te kiezen, het beste is een .png bestand. Ik kies een **bal.png**. en klik op **Add**.
De sprite is toegevoegd in het Content project. Nu moeten we de bal inladen in het programma. Voordat we dat kunnen doen, moeten we een object declareren van het type **Texture2D**:

```
private Texture2D texture;    // Visual C# declaratie (private is optioneel)
```

In Visual Basic hebben we het object ook nodig, maar dan geschreven als volgt:

```
Private texture As Texture2D  ' Visual Basic declaratie
```

Neem onderstaande code over in Visual Basic, onder de spriteBatch declaratie:

```
'Texture die we willen renderen
Private texture As Texture2D
'Bepaal waar de sprite getekend wordt
Private spritePos As Vector2 = Vector2.Zero
'X en Y snelheid van de sprite
Private XSpeed As Single = 80
Private YSpeed As Single = 120
'Vector2 wordt gebruikt voor de snelheid van de sprite
Private spriteSpeed As New Vector2(XSpeed, YSpeed)
```

Nu komt het probleem dat we niet de Content kunnen gebruiken, zoals we dat wel in Visual C# kunnen doen. Dankzij de Content Pipeline kunnen we onderstaande code in de LoadContent maken:

```
protected override void LoadContent()
{
     // Create a new SpriteBatch, which can be used to draw textures.
     spriteBatch = new SpriteBatch(GraphicsDevice);

    texture = Content.Load<Texture2D>("Bal");
}
```

Eerder zei ik al dat het .NET Framework en XNA Framework wat met elkaar gemeen hebben. Daardoor is er toch een oplossing om textures in te kunnen laden. XNA heeft namelijk een **FromStream()** methode in de Texture2D klasse, een methode die afkomstig is van het .NET Framework.

Bekijk onderstaande code en voeg deze toe in de LoadContent subroutine, waar de TODO: commentaar staat.

```
'Laad de texture.
'Maak gebruik van een Stream om de inhoud in te kunnen laden.
Dim textureStream As IO.Stream = New _
     IO.StreamReader(Application.StartupPath & _
     "\Textures\Bal.png").BaseStream
'Laad de texture in.
texture = Texture2D.FromStream(GraphicsDevice, textureStream)
```

Zonder de methode FromStream zouden we niets kunnen doen met de inhoud.

Merk op dat **Application.StartupPath** wordt gebruikt. De map **Textures** bestaat niet. Maak de map zelf aan en zorg ervoor dat al de inhoud in die map staat. De StartupPath verwijst naar de map waar het uitvoerbare programma in staat. Dat is in de Debug map. Zorg er dus voor dat daar ook de Textures map in staat. De path zal dan zijn: “\VBGame\bin\Debug\Textures”

In Visual C# is al deze rompslomp niet nodig. De Content Pipeline neemt alles voor zijn rekening en daarom is enkel het bestand opgeven voldoende. Merk ook op dat in de Load methode geen extensie van het Bal bestand is opgegeven. De Content Pipeline vereist dat niet, omdat ook de extensie van het bestand voor zijn rekening wordt genomen.

Toch kent Visual Basic de Load methode. We zouden onderstaande regel kunnen nemen i.p.v. bovenstaande Stream code:

```
texture = Content.Load(Of Texture2D)("Bal.png")
```

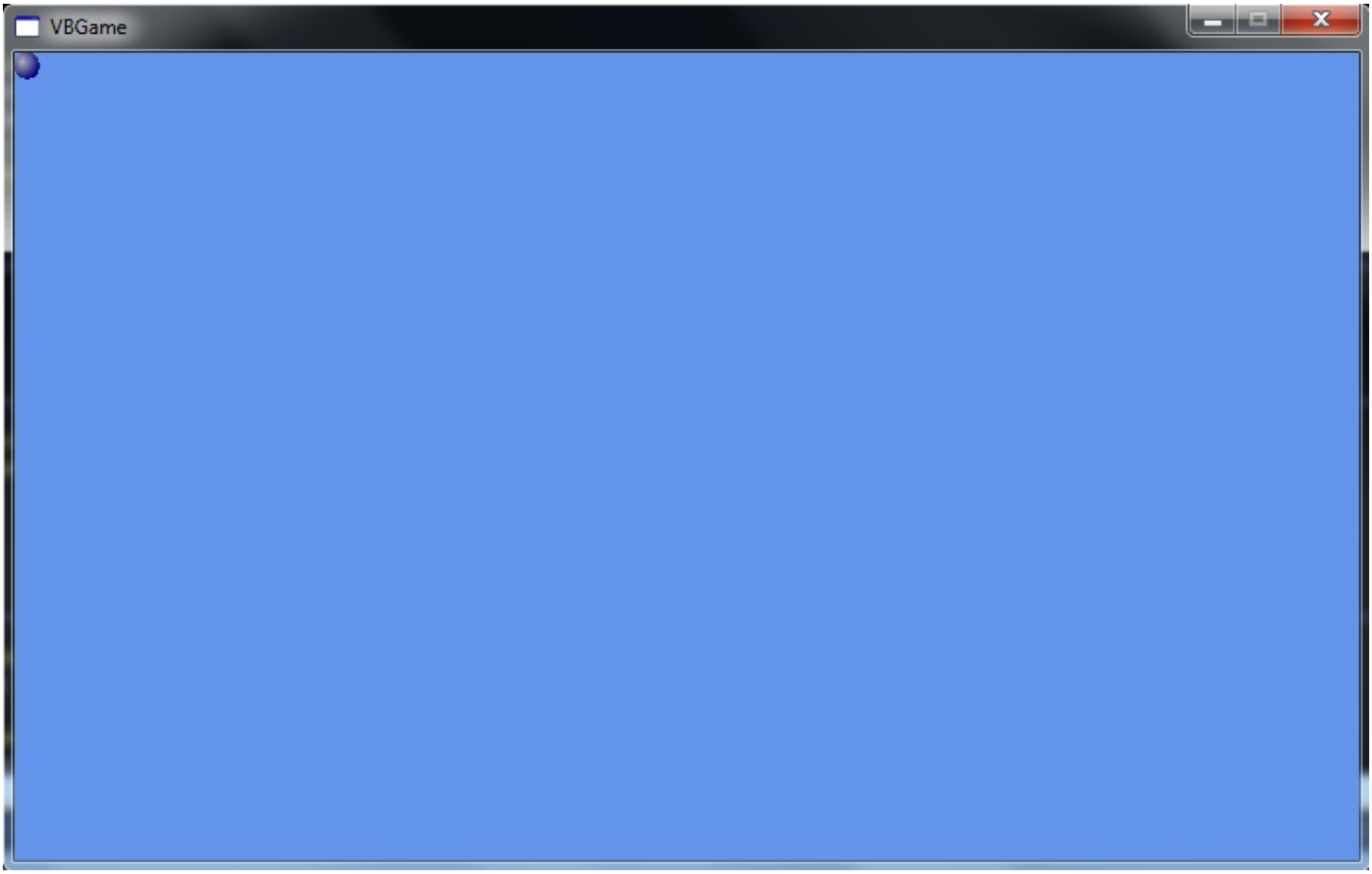
De compiler gaat hier niet over klagen, maar XNA wel. Zonder het Content project werkt bovenstaande regel niet. Als u probeert het programma te starten, zal er een foutmelding verschijnen met de melding dat het bestand niet geopend kan worden. Ook niet als u Application.StartupPath met de Textures map probeert. XNA heeft voor het Content object de Content Pipeline gewoon nodig.

Nu de texture geladen is, kunnen we het tekenen. Hieronder is de code die de texture tekent in de Draw() methode. Plaats dit waar de TODO commentaar staat:

```
'Teken de sprite
spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend)
spriteBatch.Draw(texture, spritePos, Color.White)
spriteBatch.End()
```

Als we het programma starten, zal onderstaand venster weer verschijnen, ditmaal met een sprite linksboven. Waarom de bal linksboven staat, heeft te maken met de instelling van **Vector2.Zero**. De instelling Zero zorgt ervoor dat de sprite op positie (0, 0) gerendeerd wordt.

Het venster komt niet van het sjabloon project. Ik heb namelijk eerst een kopie van het sjabloon project gemaakt, voordat ik inhoud in het project ging programmeren. Het is beter om altijd het sjabloon leeg te laten, zodat u niet telkens het sjabloon opnieuw hoeft te maken.



Net zoals het lege venster, zal ook de bal constant opnieuw worden getekend. Dit gaat zo ontzettend snel dat het niet te volgen is. In de volgende Bulletin zal ik u meer laten zien over de Update methode en de Draw methode en hoe we de bal kunnen laten bewegen en tegen de randen kunnen laten botsen.

Liberty BASIC API Reference.

SetParent
Deze functie stelt de afstamming van het venster of control waarvan de handle is “hChild” als venster handle “hParent” in, waar de afstammelingen vandaan komen. Probeer het volgende programma. Zie ook dat het venster, dat een zoonvenster is, beperkt binnen de grenzen van haar nieuwe bovenliggende venster moet worden weergegeven.

```
nomainwin
WindowWidth=200 : WindowHeight=100
open "Maak me een zoonvenster" for window as #hc

WindowWidth=400 : WindowHeight=400
open "Maak me de ouder" for window as #hp
print #hp, "trapclose [quit]"

hChild=hwnd(#hc)
hParent=hwnd(#hp)

calldll #user32, "SetParent",_
```

```
     hChild as ulong,_        'handle van venster
     hParent as ulong,_       'handle van venster dat de ouder is
     result as ulong          'handle van vorige ouder
wait
[quit] close #hp : close #hc : END
```

SetRect

De SetRect functie stelt de coördinaten van de gespecificeerde rechthoek in. Dit is hetzelfde als het toekennen van de left, top, right en bottom argumenten aan de opgegeven members van de RECT structuur. Merk op dat dit kan met de Liberty BASIC code en dat de API aanroep geschreven kan worden zonder commentaar om het kleiner te maken.

```
struct rectangle, left as long, top as long, right as long, bottom as long

calldll #user32, "SetRect",_
     rectangle as struct,_    'naam van rect struct
     0 as long,_              'left X
     0 as long,_              'top Y
     200 as long,_            'right X
     200 as long,_            'bottom Y
     ret as boolean           'niet-nul=succes

'de LB inheemse manier om een struct te vullen:
rectangle.left.struct = 0
rectangle.top.struct = 0
rectangle.right.struct = 200
rectangle.bottom.struct = 200

'minder typen met API functie:
calldll #user32, "SetRect",_
     rectangle as struct,_
     0 as long,_
     0 as long,_
     200 as long,_
     200 as long,_
     ret as boolean
```

SetSysColors

Gebruik deze functie om de systeemkleuren te wijzigen. “colorpoint” is een pointer naar een array (struct) van systeemkleuren om te wijzigen – zie [system.values]. “colornum” is een pointer naar een array (struct) van de kleuren om te wijzigen. “nChanges” is het aantal elementen in de array. Omdat deze wijziging van invloed is op het hele besturingssysteem, en niet automatisch terug kan wanneer het programma wordt gesloten, gebruik dan GetSysColors om de kleuren te krijgen die ingesteld zijn door de gebruiker voor alle kleuren die worden gewijzigd. Herstel deze kleuren wanneer het programma beëindigd wordt.

Voorbeeld:

```
struct colorpoint,_
     c1 as long,_
     c2 as long,_
     c3 as long

'drie kleuren zullen worden gewijzigd
colorpoint.c1.struct = 2      'wijzig titelbalk kleur
```

```
colorpoint.c2.struct = 9      'wijzig bijschrift kleur
colorpoint.c3.struct = 15     'wijzig knop kleur

struct colornum,_
     cn1 as long,_
     cn2 as long,_
     cn3 as long

'struct voor de inhoud van de nieuwe kleurwaarden
colornum.cn1.struct = hexdec("&H00AAFF")
colornum.cn2.struct = hexdec("&HFF6600")
colornum.cn3.struct = hexdec("&HFF8833")

nChanges = 3   'aantal items in struct

calldll #user32, "SetSysColors",_
     nChanges as long,_       'aantal kleuren om te wijzigen
     colorpoint as struct,_   'naam van struct
     colornum as struct,_     'naam van struct
     r as boolean             'niet-nul=succes

[system.values]
SCROLLBAR = 0
BACKGROUND = 1
ACTIVECAPTION = 2
INACTIVECAPTION = 3
MENU = 4
WINDOW = 5
WINDOWFRAME = 6
MENUTEXT = 7
WINDOWTEXT = 8
CAPTIONTEXT = 9
ACTIVEBORDER = 10
INACTIVEBORDER = 11
APPWORKSPACE = 12
HIGHLIGHT = 13
HIGHLIGHTTEXT = 14
BTNFACE = 15
BTNSHADOW = 16
GRAYTEXT = 17
BTNTEXT = 18
ENDCOLORS = COLOR_BTNTEXT
```

SetWindowPos

Deze functie wijzigt de grootte, positie, en Z-volgorde van zoon-, pop-up en top-level vensters. Deze vensters worden gerangschikt volgens hun verschijning op het scherm; het bovenste venster krijgt de hoogste rang en is het eerste venster in de Z-volgorde.

Mogelijke vlaggen:

_SWP_DRAWFRAME	Tekent een kader (gedefinieerd in de beschrijving van de window's klasse) om het venster.
_SWP_HIDEWINDOW	Verbergt het venster.
_SWP_NOACTIVATE	Activeert niet het venster. Als deze vlag niet ingesteld is, is het venster geactiveerd en verplaatst tot de bovenkant van de bovenste of niet-bovenste groep (afhankelijk van de instelling van de WndInser-

tAfter parameter).

_SWP_NOMOVE	Behoudt de huidige positie (negeert de X en Y parameters).
_SWP_NOSIZE	Behoudt de huidige grootte (negeert de CX en CY parameters).
_SWP_NOREDRAW	Tekent niet de wijzigingen opnieuw. Als deze vlag ingesteld is, zal geen enkele soort opnieuw getekend worden. Dit geldt voor de cliënt area, het gebied van de niet-cliënt (met inbegrip van de titel en schuifbalken), en een deel van het bovenliggende venster ontdekt als gevolg van het verplaatste venster. Wanneer deze vlag is ingesteld, moet de toepassing alle delen van het venster en het bovenliggende venster expliciet ongeldig opnieuw tekenen.
_SWP_NOZORDER	Behoudt de huidige volgorde (negeert de WndInsertAfter parameter).
_SWP_SHOWWINDOW	Toont het venster.

Mogelijke waarden voor hWndInsertAfter:
Geeft het venster voorafgaan aan het geplaatste venster in de Z-volgorde. Deze parameter moet een venster handle of één van de volgende waarden zijn:

_HWND_BOTTOM	Plaatst het venster aan de onderkant van de Z-volgorde. Als de hWnd parameter het bovenste venster geeft, zal het venster zijn hoogste status verliezen en zal aan de onderste van alle vensters worden geplaatst.
_HWND_NOTOPMOST	Plaatst het venster voor al de niet-bovenste vensters (dat wil zeggen, achter alle bovenste vensters). Deze vlag heeft geen effect als het venster al een niet-bovenst venster is.
_HWND_TOP	Plaatst het venster aan de top van de Z-order.
_HWND_TOPMOST	Plaatst het venster voor al de niet-bovenste vensters. Het venster handhaaft de bovenste positie, zelfs wanneer het wordt gedeactiveerd.

```
flags = _SWP_NOMOVE or _SWP_NOSIZE

calldll #user32, "SetWindowPos",_
     hWin as ulong,_               'venster handle
     hWndInsertAfter as ulong,_    'plaatsing-volgorde handle
     xPos as long,_                'nieuwe x locatie
     yPos as long,_                'nieuwe y locatie
     width as long,_               'nieuwe breedte
     height as long,_              'nieuwe hoogte
     flags as long,_               'SetWindowPosition attributen
     result as boolean             'niet-nul=succes
```

SetWindowText
Deze functie wijzigt de bijschrift die opgenomen is in de titelbalk van een venster of de tekst die opgenomen is in een control waarvan de handle wordt doorgegeven in de aanroep. Als het venster van het type afbeeldingen of tekst is, roep dan GetParent aan en neem het oudervenster handle op in de aanroep. De HWND functie haalt de handle van de afbeelding vak of de tekst vak eerder op, wanneer daaraan toegepast, die opgenomen is in het venster, dan de handle van het venster zelf.

```
NewText$ = "Deze tekst was gewijzigd!"
calldll #user32, "SetWindowTextA",_
     h as ulong,_         'handle van venster of control
     NewText$ as ptr,_    'nieuwe tekst tekenreeks
     result as boolean    'niet-nul=succes
```

ShowCursor
Deze functie toont of verbergt de muiscursor. Een cursor weergave-niveautelling wordt intern bewaard. Het is verhoogd door een toon operatie en verlaagd door een verberg operatie. De cursor is alleen zichtbaar als de telling groter of gelijk is aan nul. Als een muis bestaat, is de geïnitialiseerde instelling van de cursor weergave-niveau nul; anders is het -1. Dit betekent dat TWEE aanroepingen om de cursor te laten verbergen, het nodig maakt om TWEE aanroepingen de cursor weer te laten geven. De cursor is een gedeelde resource. Een venster dat de cursor verbergt moet het laten zien voordat de cursor het cliëntgebied verlaat of het venster ziet daardoor af van control naar een ander venster. De retourneerde waarde geeft de nieuwe weergavetelling als het succesvol is.

```
showFlag = 0 'verbergt de cursor
showFlag = 1 'toont de cursor
calldll #user32, "ShowCursor",_
     showFlag as boolean,_    'toon of verberg vlag
     displayCount as long     'retourneert huidige weergavetelling
```

ShowScrollBar
Deze functie toont of verbergt de schuifbalk. Gebruik deze functie om de schuifbalk te tonen of te verbergen in een grafisch venster of eventueel in een grafisch vak.

```
'_SB_HORZ
'_SB_VERT
'_SB_BOTH

'visible=1,hidden=0

calldll #user32, "ShowScrollBar",_
     h as ulong,_         'handle van grafisch vak of venster
     bar as long,_        'welke balk om te tonen/te verbergen
     show as boolean,_    '1=show, 0=hide
     result as boolean    'niet-nul=succes
```

ShowWindow
Deze functie toont of verbergt vensters en controls. Het retourneert niet-nul als het venster daarvoor zichtbaar was en een nul als het venster daarvoor verborgen was. ShowWindow kent een aantal mogelijke SW_ stijlen. De “h” kan de handle van een venster of een control zijn.

```
ShowFlag = _SW_SHOW
calldll #user32, "ShowWindow",_
     h as ulong,_         'handle van venster of control
     ShowFlag as long,_   'hoe het venster weergegeven moet worden
     result as boolean
```

SwapMouseButton
Deze functie wisselt de functies van de muisknoppen. De linker muisknop wordt de rechter en visa versa. Deze wissel is alleen aanwezig bij de gebruiker in het Configuratiescherm. Onthoud dat alle applicaties de muis delen en de wissel van invloed is in het hele systeem. Roep niet deze functie aan zonder toestemming te vragen aan de gebruiker voor deze handeling. De functie geeft 0 als de muisknoppen in normale modus waren en niet-nul als ze omgewisseld waren.

```
Switch=0       '0=Normale muisknoppen 1=Wissel muisknoppen
calldll #user32, "SwapMouseButton",_
     Switch as boolean,_      'stel muisknop vlag in
     result as boolean
```

SystemParametersInfo
Deze functie stelt de query's of systeem-wijd parameters in. Deze functie kan ook het WIN.INI bestand bijwerken tijdens het instellen van een parameter, als de _SPIF_UPDATEINIFILE vlag ingesteld is. In het volgende voorbeeld kan de gebruiker de bureauachtergrond van Windows wijizgen.

```
filename$ = "C:\Windows\Carved Stone.bmp"   'of een ander gegeven bitmap
Action = _SPI_SETDESKWALLPAPER
flags = _SPIF_UPDATEINIFILE or _SPIF_SENDWININICHANGE

calldll #user32, "SystemParametersInfoA",_
     Action as long,_    'actievlag
     0 as long,_         'null
     filename$ as ptr,_  'naam van het bitmap bestand
     flags as long,_     'waar de wijzigingen gemaakt moeten worden
     result as boolean   'niet-nul=succes
```

WinHelpA
De WinHelpA functie start de Windows Help (winhlp32.exe) en vult de gegevens die wijzen op de aard van de toepassing die de opgevraagde help doorgeeft. De toepassing geeft de naam en, waar nodig, de pad naar de map van het helpbestand om weer te geven. Als de uCommand argument 3 is, dan is het onderwerp weergegeven wanneer het helpbestand geopend wordt. Als het argument 2 is, dan is Windows geïnformeerd dat de help niet langer meer nodig is. Als de andere applicaties niet langer meer het helpbestand gebruiken, sluit Windows het af.

```
'geeft standaard het onderwerp inhoud weer en sluit het helpbestand wanneer
'het programma beëindigd wordt

msg$="Gebruik het bestandsmenu om het helpbestand van LB4 te openen." + _
     "Laat het helpbestand open en sluit het programma." + _
     "Het helpbestand zal ook gesloten worden."
nomainwin
menu #main, "&Bestand","&Toon Help",[show],"&Afsluiten",[quit]
statictext #main.s, msg$,10,10,200,200
open "Lees Mijn Helpbestand!" for window as #main
#main "trapclose [quit]"
wait

[show]
hWndMain=hwnd(#main)     'gebruik programmavenster handle of 0
uCommand=3               'vlag om de inhoud te tonen
lpszHelp$="liberty4.hlp" : dwData=0
calldll #user32, "WinHelpA",_
     hWndMain as ulong,_      'venster handle
     lpszHelp$ as ptr,_       'naam van het helpbestand
     uCommand as long,_       'vlag voor weergavemethode
     dwData as long,_         'extra gegevens, niet gebruikt hier
     result as boolean        'niet-nul=succes
wait

[quit]
uCommand=2     'vlag om het helpbestand te sluiten
calldll #user32, "WinHelpA",_
     hWndMain as ulong,_ 'venster handle
     lpszHelp$ as ptr,_  'naam van het helpbestand
     uCommand as long,_  'vlag voor weergavemethode
```

```
     dwData as long,_    'extra gegevens
     result as boolean   'niet-nul=succes
close #main
end
```

Volgende deel
In Bulletin nummer 4 komt er een nieuw deel over functies die te maken hebben met het ophalen van systeeminformatie.

Wordt vervolgd

Tekenen in GW-BASIC – Het DRAW statement.

Wie nog het grafisch programmeren kan herinneren, weet dat er veel mogelijkheden zijn met wiskundige functies. GW-BASIC kent echter nog een andere tekenmogelijkheid dat zo uitermate krachtig is, dat we haast de andere tekenstatements niet meer zouden gebruiken. Het werkt alleen met een CGA kaart; optimale resultaten worden alleen met een kleurenscherm bereikt. Met het DRAW statement kunt u lijnen trekken in een gewenste kleur, figuren inkleuren, van richting veranderen, roteren, enzovoort. Het is mogelijk om *in* het DRAW statement een tekststring op te geven waarmee een hele reeks opdrachten verwerkt kunnen worden.

Om de werking van DRAW te begrijpen, moet u zich voorstellen dat, precies op het midden van het beeldscherm, een pen geactiveerd kan worden die met commando's naar boven, naar beneden, naar links en naar rechts gestuurd kan worden, waardoor een voorstelling op het scherm kan ontstaan. DRAW kent zoveel commando's dat het wel een programmeertaal lijkt. Eerst laat ik de eenvoudigste mogelijkheden zien en daarna de ingewikkelde.

De eenvoudigste DRAW commando's
De gewone bewegingscommando's van DRAW zijn het gemakkelijkst. Ze bestaan uit één letter (het maakt niet uit of het een hoofd- of kleine letter is), gevolgd door een getal. Dat getal geeft aan hoeveel pixels er naar boven, onder, links of rechts gegaan moet worden. De commando's bij DRAW staan tussen aanhalingstekens, net als bij een string. Tussen de commando's mogen spaties voor de leesbaarheid staan, maar dat is niet noodzakelijk. Hieronder ziet u de eerste lijst (de letter *n* staat steeds voor een getal).

Eerste serie DRAW commando's:

Naam	Betekenis	Functie
Un	Up	De pen gaat n pixels naar boven.
Dn	Down	De pen gaat n pixels naar beneden.
Ln	Left	De pen gaat n pixels naar links.
Rn	Right	De pen gaat n pixels naar rechts.
En		De pen gaat n pixels diagonaal naar rechtsboven.
Fn		De pen gaat n pixels diagonaal naar rechtsonder.
Gn		De pen gaat n pixels diagonaal naar linksonder.
Hn		De pen gaat n pixels diagonaal naar linksboven.

De DRAW commando's werken alleen op de grafische schermen, dus SCREEN 1 en SCREEN 2. Op SCREEN 0, het tekstscherm, werkt het commando niet.

Hieronder volgt een programma waarin het DRAW statement wordt gedemonstreerd. We zien daarin dat het toegestaan is in de DRAW string diverse commando's te geven na elkaar. Het programma tekent een vierkant en twee ruiten.

```
100 REM Het DRAW statement
110 SCREEN 1
120 CLS:KEY OFF
130 DRAW "U50 L50 D50 R50"
140 GOSUB 220
150 DRAW "E50 F50 G50 H50"
160 GOSUB 220
170 DRAW "U50
180 DRAW "F50 E50 H50 G50"
190 GOSUB 220
200 CLS:SCREEN 0:WIDTH 80:KEY ON
210 END
220 REM toets
230 LOCATE 23,12:PRINT"Druk op een toets!"
240 A$=INKEY$:IF LEN(A$)=0 THEN 240
250 RETURN
```

Toelichting

130 Vanuit het midden van het scherm wordt een vierkant getekend.
140 Het programma gaat steeds naar de subroutine om u de gelegenheid te geven goed op het scherm te zien wat er aan het beeld veranderd is.
150 Het eindpunt van het vierkant is ook het beginpunt van de ruit die hier getekend wordt.
170 De pen wordt op de rechterbovenhoek van het vierkant geplaatst.
180 Nu wordt vanaf dit punt weer een ruit getekend.

We kunnen de DRAW commando's ook aan stringvariabelen toekennen.

Wanneer we eerst het commando geven: VIERKANT$ = "U30 L50 D30 R50" en daarna: DRAW VIERKANT$, dan wordt er ook een figuur getrokken. Het maakt dus niet uit of we na het DRAW statement de commando's als string weergeven, tussen aanhalingstekens, of als een stringvariabele. Hier is een voorbeeldprogramma dat een huis op het scherm tekent.

```
100 REM huis
110 SCREEN 1
120 CLS:KEY OFF
130 VIERKANT$ = "U30 L50 D30 R50"
140 DRIEHOEK$ = "H30 G30 R60"
150 DRAW VIERKANT$
160 DRAW "U30 R5
170 DRAW DRIEHOEK$
180 GOSUB 210
190 CLS:SCREEN 0:WIDTH 80:KEY ON
200 END
210 REM toets
220 LOCATE 23,12:PRINT"Druk op een toets!"
230 A$=INKEY$:IF LEN(A$)=0 THEN 230
240 RETURN
```

Toelichting

130 Aan de stringvariabele VIERKANT$ wordt de waarde van vier DRAW commando's toegekend.
140 Ook aan de stringvariabele DRIEHOEK$ wordt een aantal DRAW commando's toegekend in een string.
150 Voor de uitvoering van het DRAW statement wordt de inhoud van de stringvariabele VIERKANT$ gebruikt.

160 Omdat de basis van de driehoek groter is gemaakt dan de bovenkant van de rechthoek, moet de pen even op de juiste plaats neergezet worden: eerst 30 pixels omhoog en daarna 5 pixels naar rechts.
170 De inhoud van de variabele DRIEHOEK$ wordt door het DRAW statement uitgevoerd.

Aanvullende DRAW commando's

Het DRAW statement kent heel veel commando's, zodat het op een programmeertaal lijkt. DRAW kan vergeleken worden met LOGO, die net zo werkt, maar de commando's dan wel zelfstandige statements zijn. Ook het PLAY statement werkt op dezelfde manier voor geluid en muziek. Met PLAY kunnen de noten in een stringvariabele worden verwerkt en afgespeeld worden.

Hieronder staan de aanvullende DRAW commando's.

Naam	**Betekenis**	**Functie**
Mx,y	Move	Verplaats de pen naar een x,y coördinaat. Dit verplaatsen kan absoluut of relatief gebeuren. Als de X waarde *zonder* plus- of minteken (+,-) wordt weergegeven is de adressering absoluut; als de X waarde *wel* voorzien is van plus- of minteken, dan is de adressering relatief, afhankelijk dus van de *positie* van de pen op het moment dat het commando wordt gegeven.
B		De pen kan worden verplaatst, zonder te tekenen. Dit commando wordt als een voorvoegsel aan commando's als U50 of L90 toegevoegd. Het commando luidt dan: BU50, respectievelijk BL90.
N		De pen wordt door dit commando tekenend verplaatst, maar keert daarna terug naar de oorsprong, het midden van het scherm. Ook dit commando wordt als voegvoegsel aan commando's als U50 of L90 toegevoegd. Het commando luidt dan: NU50, respectievelijk NL90.
An	Angle	Met dit commando wordt het coördinatensysteem tegen de klok in gedraaid; als *n* een negatief getal is, wordt het systeem met de klok mee gedraaid. Het getal kan variëren van 0 tot 4. De rotatie is bij 0: nul graden, bij 1: 90 graden, bij 2: 180 graden en bij 3: 270 graden.
Cn	Color	De kleur van de pen verandert, afhankelijk van de waarde van *n*. De kleuren van 0 tot en met 3 zijn afhankelijk van de paletkleur die bij het COLOR statement is gekozen.
Sn	Scale	Dit commando zet de schaal waarmee het scherm werkt, op de voor *n* ingetikte waarde. Standaard staat S op 4. De getallen die voor *n* ingetikt mogen worden, kunnen variëren van 1 tot 255.
TAn	Turn	Met dit commando wordt de geldige hoek, waarmee de pen tekent, gedraaid, zodat figuren geroteerd kunnen worden. De waarde van *n* moet tussen de -360 en de +360 liggen.
Pk,r	Paint	Dit commando kleurt een figuur in met de kleur *k* tot aan de randkleur *r*. Het is te vergelijken met het PAINT statement; de kleuren *k* zijn afhankelijk van de paletkleuren die bij het COLOR statement zijn gekozen.
Xstring;		Binnen een string kan een eerder gedefinieerde substring worden uitgevoerd. Binnen de DRAW commando's lijkt dit commando op een GOSUB statement. Voorbeeld: als de waarde van de hoekrotatie met het *TAn* commando in een variabele is vastgelegd, bijvoorbeeld X, dan wordt de rotatie bepaald door het commando DRAW "TA=X;", waarbij de puntkomma niet mag worden vergeten!

Hieronder zullen een aantal korte, overzichtelijke programma's met de mogelijkheden van deze commando's getoond worden. Vooral als we figuren roteren, ontstaan er heel aardige visuele beelden. We beginnen met een programma dat met het *N* commando werkt. Dit programma trekt vanuit de oorsprong, dat is het middelpunt van het scherm, gekleurde lijnen en keert dan weer naar de oorsprong terug om een kleine hoek te maken en opnieuw een lijn in een andere kleur te tekenen. Dit gaat zo door tot alle mogelijke lijnen zijn getrokken.

```
100 REM De DRAW-N functie
110 SCREEN 1
120 CLS:KEY OFF
130 FOR GRAAD = 1 TO 357 STEP 3
140   FOR KLEUR = 1 TO 3
150     GRAAD = GRAAD + KLEUR - 1
160     DRAW "TA=GRAAD; C=KLEUR; NU99"
170   NEXT KLEUR
180 NEXT GRAAD
190 CIRCLE(160,100),119,2
200 PAINT (1,1),1,2
210 GOSUB 230
220 END
230 REM toets
240 A$=INKEY$:IF LEN(A$)=0 THEN 240
250 RETURN
```

Toelichting

130 Hier wordt de lus van de stralen van de cirkel gestart. STEP 3 is hier nodig om de drie contrasterende kleuren van het gekozen palet te kunnen tonen.

140 Start van de binnenste lus die de kleuren van de te tekenen stralen bepaalt.

150 De uiteindelijke GRAAD wordt nu vastgelegd, gecombineerd met de waarden van de variabele KLEUR.

160 Met het *TA* commando gevolgd door het **=** teken, wordt de waarde van de geldige GRAAD toegekend aan het rotatiemechanisme. Op vergelijkbare wijze wordt de waarde van KLEUR toegekend aan het commando C. Tenslotte wordt met het gecombineerde N en U commando een lijn van 99 pixels getrokken, waarna de pen wegens het N voorvoegsel weer teruggeplaatst wordt op de oorsprong.

190 Nu wordt een cirkel van een rode kleur om de stralen geplaatst.

200 Met het PAINT statement wordt de rest van het scherm met cyaan ingekleurd.

Een volgend programma laat 36 keer een vierkant zien op het scherm, steeds 10 graden roterend. Het gaat weer om een *TA* commando.

```
10 REM Herhaald vierkant
20 SCREEN 1
30 CLS:KEY OFF
40 DRAW "C1"
50 VIERKANT$ = "U30 L30 D30 R30"
60 FOR A = 1 TO 360 STEP 10
70   DRAW "TA=A;"
80   DRAW VIERKANT$
90 NEXT A
100 END
```

Toelichting

40 Kleur 1, cyaan, wordt gekozen.

50 Aan de stringvariabele VIERKANT$ wordt de DRAW waarde toegekend.

60 De lus wordt gestart.
70 Bij elke rondgang van de lus wordt de hoek, waarin getekend wordt, met 10 verhoogd.
80 Elke keer wordt een vierkant getekend, steeds tien graden naar links gedraaid, met de rechteronderhoek als rotatiepunt.

Het programma is met het onderstaand programma zo uitgebreid dat niet één maar negen figuren in drie verschillende kleuren op het scherm verschijnen. In het onderstaand programma speelt ook het *B* voorvoegsel een rol. Het zetten van een *B* voor een verplaatsingscommando zorgt voor de verplaatsing van de pen, zonder dat er een lijn op het scherm verschijnt.

```
100 REM Herhaald vierkant
110 SCREEN 1,1
120 CLS:KEY OFF
130 VIERKANT$ = "U30 L30 D30 R30"
140 DRAW "BL200
150 GOSUB 230:REM kleur
160 DRAW "BL300 BU70
170 GOSUB 230:REM kleur
180 DRAW "BL300 BD140
190 GOSUB 230:REM kleur
200 GOSUB 350:REM toets
210 SCREEN 0:WIDTH 80:KEY ON
220 END
230 REM Kleur
240 FOR C = 1 TO 3
250   GOSUB 280
260 NEXT C
270 RETURN
280 REM teken figuur
290 DRAW "BR100 C=C;
300 FOR A = 0 TO 360 STEP 10
310   DRAW "TA=A;
320   DRAW VIERKANT$
330 NEXT A
340 RETURN
350 REM toets
360 A$=INKEY$:IF LEN(A$)=0 THEN 360
370 RETURN
```

Toelichting

130 VIERKANT$ krijgt een waarde om een vierkant op het scherm te zetten.
140 Met het *B* commando wordt ervoor gezorgd dat de pen 200 pixels naar links wordt verplaatst, zonder dat er een lijn op het scherm verschijnt.
150 Het programma gaat naar de subroutine die in regel 240 begint. Daar wordt de kleur *C* drie keer veranderd met behulp van een lus. In die routine gaat het programma naar de volgende lus vanaf regel 300 die 36 maal het vierkant weergeeft. Daarin wordt steeds met een hoek van 10 graden geroteerd. Zie ook het vorige programma. In de regels 170 en 190 worden dezelfde subroutines aangesproken.
160 Om de bovenste rij figuren te kunnen weergeven op het scherm, moet de pen weer, zonder een lijn op het scherm te zetten, op de juiste plaats op het scherm worden gezet. Dat gebeurt in deze regel. Vanaf het punt waar de pen stond, wordt de pen 300 pixels naar links en 70 pixels naar boven gebracht.
180 Vergelijk regel 160. De pen gaat nu 300 pixels naar links en 140 pixels naar beneden, ook nu weer zonder dat er een lijn op het scherm verschijnt.

Appendix – De twee XNA sjablonen compleet, VC# en VB.

In deze Appendix laat ik u de XNA sjablonen compleet zien van Visual C# en Visual Basic. Let op dat ik nu gebruik maak van een Program module in plaats van een mdlMain module, om het verschil tussen de twee sjablonen zo klein mogelijk te houden. Hier komt eerst het complete sjabloon van Visual C# die u niet hoeft over te nemen, omdat die al direct in de sjablonenlijst gekozen kan worden.

Game1.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace WindowsGameBB201503
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting to run.
        /// This is where it can query for any required services and load any non-graphic
        /// related content.  Calling base.Initialize will enumerate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            // TODO: use this.Content to load your game content here
        }
```

```
        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// all content.
        /// </summary>
        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here

            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here

            base.Draw(gameTime);
        }
    }
}
```

Program.cs

```
using System;

namespace WindowsGameBB201503
{
#if WINDOWS || XBOX
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        static void Main(string[] args)
        {
            using (Game1 game = new Game1())
            {
                game.Run();
            }
        }
    }
#endif
}
```

De complete sjabloon die nu volgt voor Visual Basic 2010 moet volledig overgenomen worden. U kunt de module **mdlMain** dus ook **Program** noemen, hoe de naam van de module is maakt niet uit. Onthoud wel dat Visual C# geen modules kent, dus is Program.cs ook een klasse.

Game1.vb

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports Microsoft.Xna.Framework
Imports Microsoft.Xna.Framework.Audio
Imports Microsoft.Xna.Framework.Content
Imports Microsoft.Xna.Framework.GamerServices
Imports Microsoft.Xna.Framework.Graphics
Imports Microsoft.Xna.Framework.Input
Imports Microsoft.Xna.Framework.Media

Public Class Game1
    Inherits Microsoft.Xna.Framework.Game

    Dim graphics As GraphicsDeviceManager
    Dim spriteBatch As SpriteBatch

    Public Sub New()
        graphics = New GraphicsDeviceManager(Me)

    End Sub

    Protected Overrides Sub Initialize()
        'TODO: Voeg uw initialisatie hier
        MyBase.Initialize()
    End Sub

    Protected Overrides Sub LoadContent()
        spriteBatch = New SpriteBatch(GraphicsDevice)
        'TODO: Gebruik hier het Content om uw spelinhoud te laden
        MyBase.LoadContent()
    End Sub

    Protected Overrides Sub UnloadContent()
        'TODO: Ontlaad elke niet ContentManager inhoud hier
        MyBase.UnloadContent()
    End Sub

    Protected Overrides Sub Update(gameTime As Microsoft.Xna.Framework.GameTime)
        'Staat toe om het spel te sluiten
        If GamePad.GetState(PlayerIndex.One).Buttons.Back = ButtonState.Pressed Then
            Me.Exit()
        End If
        'TODO: Voeg uw update logica hier
        MyBase.Update(gameTime)
    End Sub

    Protected Overrides Sub Draw(gameTime As Microsoft.Xna.Framework.GameTime)
        GraphicsDevice.Clear(Color.CornflowerBlue)
        'TODO: Voeg hier uw tekencode hier
        MyBase.Draw(gameTime)
    End Sub
End Class
```

Program.vb

```
Module Program
    Sub Main()
        Using game As New Game1
            game.Run()
        End Using
    End Sub
End Module
```

Cursussen

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

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

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

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

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

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

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

Software

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

Hoe te bestellen

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

ABN-AMRO nummer 49.57.40.314
HCC BASIC ig
Haarlem

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

Alle prijzen zijn inclusief verzendkosten voor Nederland en België.

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

Waarover	Wie	Wanneer	Tijd	Telefoon	Email
Liberty BASIC	Gordon Rahman	ma. t/m zo.	19-23	(023) 5334881	grahman@planet.nl
MSX-Basic	Erwin Nicolai	vr. t/m zo.	18-22	(0516) 541680	basic@lordthanatos.com
PowerBasic CC	Fred Luchsinger	ma. t/m vr.	19-21		f.luchsinger@kader.hcc.nl
QBasic, QuickBasic	Jan v.d. Linden				j.vd.linden@kader.hcc.nl
Visual Basic voor Windows	Jeroen v. Hezik	ma. t/m zo.	19-21	(0346) 214131	j.a.van.hezik@kader.hcc.nl
Visual Basic .NET	Marco Kurvers	vr. t/m zo.	19-22	06 30896598	m.a.kurvers@live.nl
Basic algemeen, zoals VBA Office	Marco Kurvers	vr. t/m zo.	19-22	06 30896598	m.a.kurvers@live.nl
Web Design, met XHTML en CSS