Alex Ries

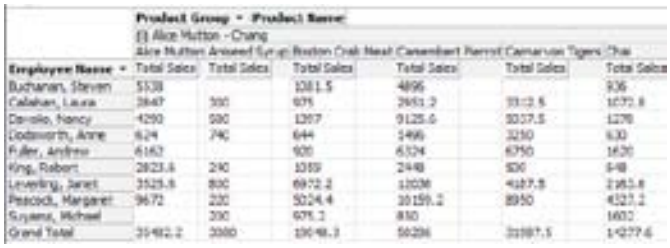
XNA voor bedrijfsapplicaties

MET XNA MAAK JE GAMES EN MEER

Door het XNA Framework te bundelen met .NET kun je op een snelle en simpele manier driedimensionale grafische onderdelen toevoegen aan een applicatie. In dit artikel gaat de auteur nader in op het XNA-framework en laat hij zien hoe je het kunt gebruiken om 3D toe te voegen aan Windows-applicaties.

Bedrijven maken veel gebruik van historische gegevens om een koers voor het bedrijf te bepalen. Dit houdt in dat er veel gegevens moeten worden opgeslagen en dit kan dan worden gedaan in een zogenaamd datawarehouse. Voor datawarehouses geldt vaak een andere structuur en organisatie van gegevens dan bij operationele databases, om het opvragen van gegevens sneller te maken. Om bij de gegevens in een datawarehouse te komen en er bijvoorbeeld rapportages van te maken, wordt gebruikgemaakt van Online Analytical Processing (OLAP). De techniek van OLAP is gebaseerd op cubes, verzamelingen van bedrijfsgegevens. Met Microsoft SQL 2005 (Enterprise Editie) worden de analysis-services meegeleverd om ondersteuning voor OLAP te bieden. Via deze services kunnen cubes gemaakt en bekeken worden. Een tool om cubes mee te ontwerpen en te bekijken, is Business Intelligence Development Studio (BIDS). Dit is Visual Studio 2005 met een verzameling projecttypen specifiek voor Microsoft analysis-services en business intelligence. BIDS wordt meegeleverd met SQL 2005 Enterprise Editie.
Om gegevens uit een cube te laten zien, kun je gebruikmaken van grafieken en tabellen die in twee dimensies zijn getekend. Om meer informatie toe te voegen, wordt gebruikgemaakt van verschillende figuren of kleuren. Dit is niet altijd even duidelijk en overzichtelijk, zeker met een grote hoeveelheid gegevens zoals de verkoopcijfers over een periode van twee of drie jaar. Het toevoegen van een extra dimensie kan een uitkomst zijn om gegevens inzichtelijker te maken. Een driedimensionale weergave kan er ook nog eens gaver en dynamischer uit zien dan een simpele lijndiagram. Een framework waarin het tekenen van driedimensionale omgevingen gemakkelijk gedaan kan worden, is XNA. Met XNA hebben ontwikkelaars de beschikking over alles wat nodig is om een driedimensionaal spel te maken. Waarom dan geen driedimensionale grafieken?

Het XNA Framework werd samen met GameStudio Express gelanceerd in augustus 2006. Het doel van XNA is om het ontwikkelen van games toegankelijker en daarmee sneller te maken. Voordat XNA bestond, gebruikten ontwikkelaars veelal eigen specialistische tools en processen. Meestal werd C of C++ als ontwikkeltaal gebruikt, in combinatie met Direct3D of OpenGL om een game-engine te bouwen. Direct3D is een grafische API die in DirectX zit en wordt ondersteund door Windows en Xbox 360. Voor OpenGL, dat ook een grafische API is, zijn er implementaties voor onder andere Windows, Linux, Playstation en de WII. Met deze API's moet nog alle game-logica ontwikkeld worden, en dat is een langdurig proces. Voor het verschijnen van XNA, bestond er ook een .NET-oplossing voor DirectX, namelijk Managed DirectX, een verbinding om met managed code DirectX aan te spreken. XNA biedt de ontwikkelaar veel meer dan alleen grafische aansturing. Er kan direct gebruik worden gemaakt van het .NET Framework, waar veel features in zitten, en het bevat het XNA Framework, dat veel games-gerelateerde patterns en practices bevat. Er bestaat zelfs een project om het XNA Framework op te nemen in Mono, het .NET Framework in Linux.



Afbeelding 1. Uitvoer van de BIDS Cube-browser

XNA samen met ADOMD

Om een applicatie te maken die queries kan uitvoeren op een OLAP-database in SQL Server 2005, gebruik je ActiveX Data Objects Multi Dimensional (ADOMD). Dit werkt op een manier die vergelijkbaar is met ADO.NET. Je maakt een command aan met een query en je voert deze uit via een connectie; zie codevoorbeeld 1. ADOMD wordt geleverd met de Enterprise Edition van SQL Server 2005, speciaal om cubes uit te lezen. Uit een query met ADOMD komt een cellset, met daarin een aantal cellen. Iedere cel is een resultaat en staat op de assen die uit de query komen. Zo kan een specifieke cel uit een cellset worden opgevraagd uit een query met drie assen door CellSet.Cells[x, y, z] aan te roepen.

Met gegevens uit de CellSet kun je een grafiek maken. Het tekenen van de grafiek doe je met behulp van XNA. XNA reikt je een draw-methode aan voor het tekenen, die automatisch door de game-class wordt uitgevoerd. Binnen de draw-methode kan een stuk code, zoals in codevoorbeeld 2, gezet worden, die door ieder punt heen loopt en vervolgens het bijbehorende model eerst schaalt en naar de juiste positie verplaatst.

Het resultaat is een driedimensionale ruimte gevuld met kegels; voor iedere cel in de CellSet een kegel. Zo'n lading kegels zegt niet

Afbeelding 2. Ontwikkeling van XNA

```
// Execute the current query
private void Execute()
{
   // Create an ADOMD command using the MDX query
   AdomdCommand queryCommand = new AdomdCommand(this.query);
   queryCommand.Connection = this.connection;
   // Insert the resulting CellSet into the conversion engine
   conversionEngine.InsertCellSet(queryCommand.ExecuteCellSet());
}
```

Codevoorbeeld 1. Een query versturen met ADOMD

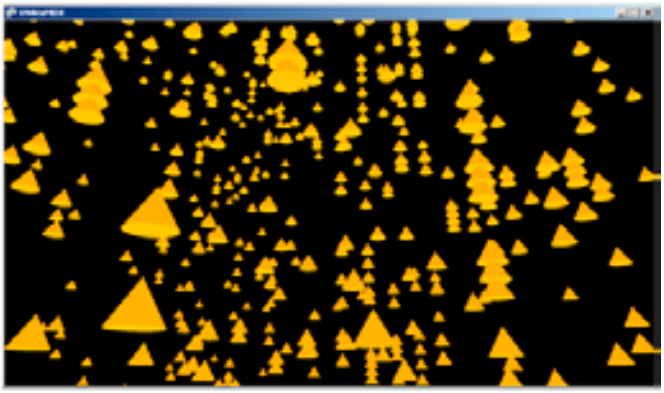
erg veel, maar maakt duidelijk wat mogelijk is. Alleen al door het soort model te veranderen in bijvoorbeeld een driedimensionaal model van het product en deze te schalen naar de waarde van de cel, ontstaat een heel ander beeld. Het gebruik van kleuren om bijvoorbeeld hot en cold spots aan te geven, zou dit ook inzichtelijker maken. Met een bewegende camera kan er door de driedimensionale ruimte 'gevlogen' worden. Stel je eens voor: een driedimensionale tour door de resultaten uit het verleden en trends voor de toekomst.

Het XNA Framework is zoals gezegd een geweldig framework voor games, je boekt er snel mooie resultaten mee. Alleen de meeste mensen die zich bezighouden met business intelligence zijn niet bezig met XNA-games. Zij gebruiken BIDS en Excel. Gelukkig is er een oplossing om XNA te verwerken in BIDS. Het is namelijk mogelijk een XNA-game naar een zelfgemaakte control te laten renderen in plaats van het scherm via een Visual Studio add-in.

XNA-control

De basis van het XNA-control is een leeg user-control. XNA kan er direct op tekenen en het begint als een leeg panel. Het deel van het control dat gebruikt wordt door de XNA-game is de handle. De volgende stap is om de XNA-game te vertellen dat hij naar deze control moet renderen. Dat kun je doen door XNA te vertellen dat hij een andere handle moet gebruiken om naar te renderen. Deze handle wordt opgeslagen in de DeviceWindowHandle. Wanneer deze handle wordt veranderd naar de handle van het user-control, zal XNA de output naar het control renderen. Het veranderen van de handle moet gebeuren binnen het PreparingDeviceSettings-event; zie codevoorbeeld 3. Dit event wordt aangeroepen wanneer XNA de voorbereidingen treft voor de GraphicsDevice. Het event PreparingDeviceSettings komt uit de GraphicsDeviceManager van de game-class.

XNA gebruikt nu het control in plaats van het eigen scherm om op te renderen. Bij het starten van een XNA-game-class wordt een scherm aangemaakt. Met dit scherm wordt de GraphicsDevice aangemaakt, die de class gebruikt voor al het renderen. Dit scherm heeft dezelfde functionaliteit als een scherm van een Windows. Forms-applicatie. Dus als het scherm afgesloten wordt, wordt de XNA-instantie ook afgesloten. Je kunt het scherm wel verbergen. Als eerste haal je de form-instantie uit de game door middel van de methode Control.FromHandle. De handle van het gamescherm



Afbeelding 3. De visualisatie van een CellSet in 3-D in een XNA-applicatie

```
// Draw each of the scaled visual points
foreach (VisualPoint point in drawPoints)
{
   // Scale the model and put it in the right place
   this.Scene.ShaderParams.World =
      Matrix.CreateScale(coneScale) *
      Matrix.CreateTranslation(new Vector3(point.X, point.Y, point.Z));
   // Draw the model
   this.DrawModel(coneModel);
}
```

Codevoorbeeld 2. Tekenen van punten met een 3D-model

```
private void OnPreparingDeviceSettings(object sender,
PreparingDeviceSettingsEventArgs args)
{
   PresentationParameters presentationParamaters =
      args.GraphicsDeviceInformation.PresentationParameters;

   // Set the device handle to the handle of our control
   presentationParamaters.DeviceWindowHandle = panel.GetSafeHandle();
}
```

Codevoorbeeld 3. Verander de DeviceWindowHandle in het event PreparingDeviceSettings

bevindt zich in de Game.Window.Handle-property. Het verbergen van het scherm doe je in het event DeviceResetting van de GraphicsDeviceService (zoals zichtbaar in codevoorbeeld 5), omdat anders het scherm weer terugkomt met de standaard waarden. Nu heb je een user-control om op te tekenen en een XNA-game-instantie die ingesteld is om dit te doen. Een XNA-game wordt normaal via een XNA-executable gestart, maar in dit geval wordt een XNA-instantie gestart vanuit de Visual Studio-applicatie. Een Windows Forms-applicatie wordt gestart met de methode Application.Run, een XNA-applicatie wordt gestart met de methode Game.Run. Die run-methode kun je aanroepen vanuit een andere thread, zie codevoorbeeld 6. Met het starten van die thread wordt er een instantie van de XNA-game-class gestart, maar het tekent nu op de user-control. Om de instantie weer te stoppen, kan de gebruikelijke Exit-methode gebruikt worden.

De XNA-control als add-in

Het control kan vervolgens gebruikt worden door een add-in met behulp van een nieuw Visual Studio-add-in-project. Met een nieuw add-in-project worden twee bestanden aangemaakt, een connect-class en een .Addin-bestand. De connect-class zorgt voor

```
// Get the XNA Game Window
windowForm = Control.FromHandle(this.Window.Handle) as Form;
```

Codevoorbeeld 4. Een form maken uit de achterliggende GameWindow

```
void OnDeviceResetting()
{
   // Set the backbuffer to the panel width and height
   graphicsDevice.PresentationParameters.BackBufferWidth =
      this.panel.Width;
   graphicsDevice.PresentationParameters.BackBufferHeight =
      this.panel.Height;

   // Hide the XNA game form
   windowForm.Hide();
   windowForm.FormBorderStyle = FormBorderStyle.None;
   windowForm.Enabled = false;
   windowForm.Visible = false;
   // Remove the taskbar button
   windowForm.ShowInTaskbar = false;
   // Put the old game form outside of the visible screen
   windowForm.SetDesktopLocation(-windowForm.Width, -windowForm.Height);
```

Codevoorbeeld 5. Het formulier verbergen in het event DeviceResetting

```
// Constructor
public GameControlPanel()
{
  InitializeComponent();
  // Create a new gamethread
  gameThread = new Thread(new ThreadStart(startGame));
}

/// <summary>
/// Start the game thread
/// </summary>
public void Start()
{
  gameThread.Start();
}

/// <summary>
/// Game thread starter method
/// </summary>
private void startGame()
{
  // Create a new game passing along this control
  game = new AddInGame(this);
  // Run the game as done normally
  game.Run();
}

/// <summary>
/// Close the game if it's running
/// </summary>
public void CloseGame()
{
  // Check if the game is running
  if (gameThread.IsAlive)
  {
    game.Exit();
  }
}
```

Codevoorbeeld 6. De game wordt gestart en gestopt vanuit het user-control met een eigen thread

de verbinding met Visual Studio en is het startpunt voor jouw add-in. Het .Addin-bestand is een XML-bestand met daarin de configuratiegegevens van de add-in zoals de naam en locatie van de assembly. Dit XML-bestand wordt ook gekopieerd naar de Addin-map van Visual Studio 2005, zodat Visual Studio de add-in kan vinden en starten. Het is nu de bedoeling dat het control waar XNA op rendert zichtbaar wordt in BIDS. Dat BIDS is gestart, kun je aflezen uit het type solution of project dat in Visual Studio 2005 is geladen. Via events kan Visual Studio doorgeven dat een solution is geopend, zie codevoorbeeld 7.
Om er voor te zorgen dat de add-in alleen maar geladen wordt bij een BIDS-solution, moeten we zien te achterhalen of er een BIDS-solution of project is geopend. Maar hoe? Ieder type solution heeft een eigen Global Unique Identifier (GUID). De plek waarop deze GUID kan worden gevonden, is niet altijd hetzelfde. Een manier om er achter te komen is om de XML van solution- of projectbe-

```
// Get the solution events from the Visual Studio Application
this.solutionEvents = this.application.Events.SolutionEvents;
// Register Events
this.solutionEvents.Opened +=
  new _dispSolutionEvents_OpenedEventHandler(SolutionOpened);
this.solutionEvents.ProjectAdded +=
  new _dispSolutionEvents_ProjectAddedEventHandler
(ProjectAddedToSolution);
```

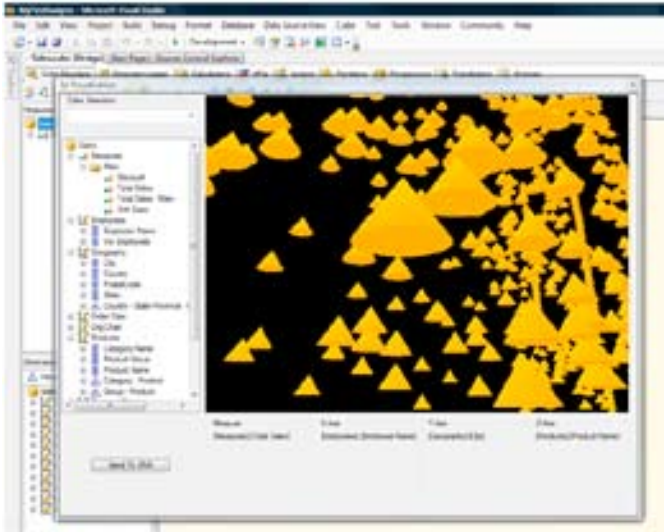
Codevoorbeeld 7. De Visual Studio-solution-events registreren

```
private void SolutionOpened()
{
  // Get the current open solution
  Solution2 solution =
    (Solution2)application.Solution;

  Projects projects = solution.Projects;

  Windows2 windows2Object =
    (Windows2)this.application.Windows;

  AddIn addinObject = this.application.AddIns.Item(1);

  // Save the type of control we want to use
  Type controlType = typeof(AddinTestControl.AddInControl);

  string assemblypath = Assembly.GetAssembly(controlType).Location;
  string classname = controlType.FullName;
  string guidpos = Guid.NewGuid().ToString("B");

  string caption = "3d Visualisation";

  // Loop through all projects in the curent solution
  foreach (Project project in projects)
  {
    // If it's a BIDS Project we can create our ToolWindow
    if (project.Kind == BidsProjectGUID)
    {
      PluginSettings.Settings.Project = project;

      Object ctlObject = new Object();

      // Create a tool window
      Window newWindowObject = windows2Object.CreateToolWindow2(
        addinObject, assemblypath, classname,
        caption, guidpos, ref ctlObject);

      // Make the new ToolWindow Visible
      newWindowObject.Visible = true;

      PluginSettings.Settings.Control.InitializeInterface();

      break;
    }
  }
}
```

Codevoorbeeld 8. Een ToolWindow aanmaken voor in Visual Studio

standen te bekijken. In het geval van BIDS stond de GUID alleen in de Project.Kind-property die je uit kunt lezen via bijvoorbeeld een quickwatch. Op het moment dat je weet dat er een BIDS-solution is ingeladen, maak je een ToolWindow aan met een XNA-user-control. Je maakt zo'n ToolWindow met de methode Create-ToolWindow2. Voor deze methode is er een GUID nodig om de ToolWindow te identificeren en een link naar de assembly met een control. Een add-in maakt gebruik van EnvDTE- en EnvDTE80-namespaces, waar onder andere de Windows2- en Solution2-classes in zitten. Deze classes kun je gebruiken om met de Visual Studio-omgeving te communiceren.

Met de code uit codevoorbeeld 8 heb je een BIDS-add-in met een scherm en een XNA-control. Om gebruik te kunnen maken van de XNA ContentPipeline, waarmee de 3D-modellen, shaders en textures worden ingeladen, moet de Content.RootDirectory van de game-class worden veranderd naar het pad van de add-in-assembly. Deze directory staat bij het laden van de game-class op de Global Application Cache (GAC) directory van XNA, doordat



Afbeelding 4. Vanuit BIDS jouw DataCube in 3D bekijken

de root directory standaard op de locatie staat waaruit de game wordt gestart. In dit geval start je de game via een thread in plaats van direct uit een game-executable.

Het uiteindelijke resultaat is de mogelijkheid om vanuit BIDS een DataCube-resultaat in drie dimensies te bekijken, zoals zichtbaar in afbeelding 4. De kracht van XNA en 3D verwerkt in een business intelligence-applicatie.

3D-mogelijkheden

Het XNA Framework is .NET en heeft dezelfde kracht en flexibiliteit. Zo kun je het inzetten als tool om 3D toe te voegen aan gewone Windows-applicaties. Het is zelfs mogelijk om dit ook als add-in te implementeren in een omgeving zoals BIDS. De krachtige 3D-mogelijkheden van XNA zijn niet beperkt tot alleen spellen, ze kunnen ook geïntegreerd worden om multidimensionale gegevens te laten zien.

Alex Ries is stagiair bij Avanade (http://www.avanade.com/), een samenwerkingsverband van Microsoft en Accenture. Voor vragen of opmerkingen is Alex te bereiken op alexri@avanade.com.

Referenties

XNA Creators Website (met XNA Game Studio download): http://creators.xna.com/
Mono.XNA: http://code.google.com/p/monoxna/Informatie voor het maken van een add-in: http://www.mztools.com/resources_vsnet_addins.aspx