Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第三篇 XNA游戏篇

# XNA 3D模型展示

《周易》：“是以立天之道曰阴与阳，立地之道曰柔与刚，立人之道曰仁与义。”我以为立Windows Phone应用程序之道为Silverlight与XNA 。Silverlight应用程序嵌入XNA模型，XNA应用程序可以嵌入Silverlight元素，二者相得益彰，互为补充。本章以Silverlight应用程序嵌入XNA 3D模型为例说明。

## 概述

本章参考和引用APP HUB（<http://create.msdn.com/en-US>）的Game Development Tutorial，以及MSDN Windows Phone开发文档。

《周易》：“是以立天之道曰阴与阳，立地之道曰柔与刚，立人之道曰仁与义。”我以为，立Windows Phone应用程序之道曰Silverlight与XNA 。Silverlight应用程序嵌入XNA模型，XNA应用程序可以嵌入Silverlight元素，二者相得益彰，互为补充。本章以Silverlight应用程序嵌入XNA 3D模型为例说明。

在本例中使用XNA的**UIElementRenderer**渲染Silverlight的 **TextBlock**控件，并呈现Silverlight的高级排版和矢量图功能。即Silverlight应用程序嵌入到XNA的3D模型。

Windows Phone Mango还支持后台传输服务，当应用程序处于非激活状态时传输服务仍可在后台独立运行。 本例说明如何使用后台传输服务。

## 动手实践——XNA 3D模型应用程序

### 应用后台传输服务

本节中实现在Silverlight框架中远程下载XNA模型。

首先在Silverlight的MainPage.xaml中显示下载列表，使用Visual Studio 2010打开解决方案ModelViewer\ModelViewer.sln， MainPage.xaml在设计视图中显示内容如图18-1 MainPage 。列表分为两个部分，本地模型列表和远程模型列表。本例中使用Windows Phone Mango提供的**BackgroundTransferService**类创建和管理下载功能。当用户在远程XNA模型列表中选择XNA模型，点击下载按钮后。应用程序将XNA模型下载至手机中，并将其从远程模型列表移动到本地模型列表。

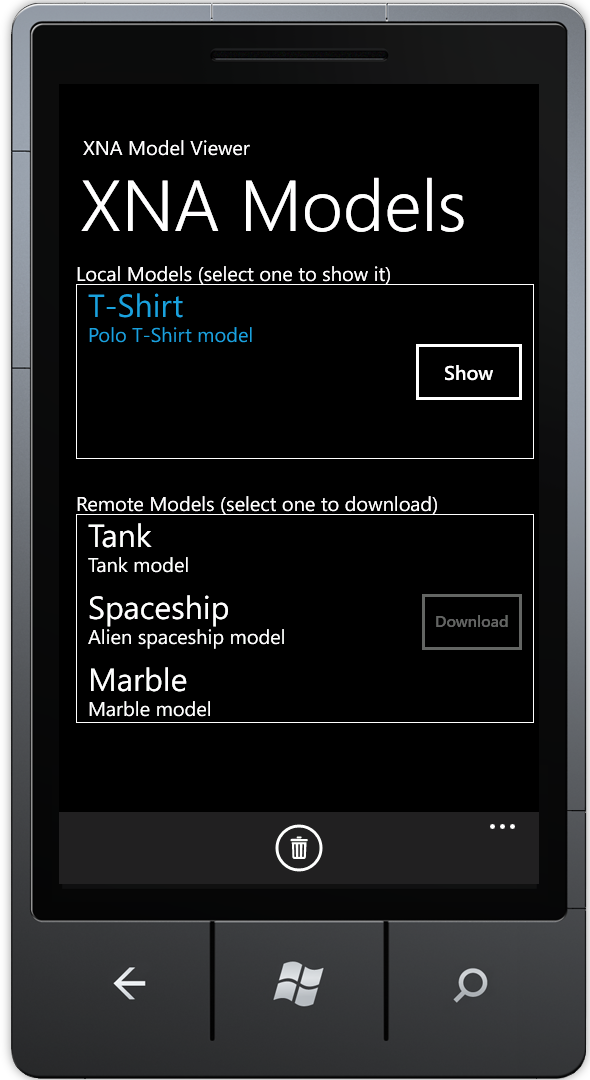


图18-1 MainPage

后台传输服务的实现方法：创建后台传输请求**BackgroundTransferRequest**，添加至后台传输服务**BackgroundTransferService**队列。

**BackgroundTransferRequest**类需要添加引用Microsoft.Phone.BackgroundTransfer。

Silverlight/XNA Project: ModelViewer File: Downloads \Download.cs

using Microsoft.Phone.BackgroundTransfer;

在Download类的Start方法中创建后台传输服务请求。requestUri声明下载地址，downloadUri声明保存的地址。当应用程序将下载请求增加到后台下载服务BackgroundTransferService的队列中后，Windows Phone操作系统在后台任务重中执行下载。此时，即使应用程序不处于激活状态，下载过程仍在继续。

在本例中，应用程序监听*TransferStatusChanged*事件和*TransferProgressChanged*事件。*TransferStatusChanged*事件传递**BackgroundTransferEventArgs**对象，通过判断下载请求的*TransferStatus*属性，在下载完成后将下载的文件移动至本例模型列表。此时因为应用程序不再需要监听后台传输请求，所以取消*TransferStatusChanged*事件的订阅。

Silverlight/XNA Project: ModelViewer File: Downloads \Download.cs

public void Start()

{

//Create a new background transfer request

BackgroundTransferRequest request = new BackgroundTransferRequest(requestUri, downloadUri);

request.TransferPreferences = TransferPreferences.AllowCellularAndBattery;

requestId = request.RequestId;

//Subscribe to the events

request.TransferStatusChanged += request\_TransferStatusChanged;

request.TransferProgressChanged += request\_TransferProgressChanged;

//Add new request to the queue

BackgroundTransferService.Add(request);

}

private void request\_TransferStatusChanged(object sender, BackgroundTransferEventArgs e)

{

BackgroundTransferRequest request = e.Request;

if(request.TransferStatus == TransferStatus.Completed)

{

request.TransferStatusChanged -= request\_TransferStatusChanged;

request.TransferProgressChanged -= request\_TransferProgressChanged;

if(isAborted)

OnDownloadAborted();

else

OnDownloadFinished(request);

}

}

**注意：**

后台传输请求可能不会立即执行，在后台传输服务使用一个特殊的调度程序来管理下载请求。例如，当电池电量有限的情况下，操作系统可能会暂停传输，以减少能源消耗。

当下载完成时，request\_TransferStatusChanged事件处理函数移动文件，通知应用程序从队列中删除的**BackgroundTransferRequest**。Windows Phone不会自动从队列中自动删除下载请求，必须由应用程序开发人员编写代码执行。*OnDownloadAborted*方法删除下载的临时文件，并且通知下载取消的事件。

Silverlight/XNA Project: ModelViewer File: Downloads \Download.cs

public event EventHandler DownloadFinished;

public event EventHandler DownloadAborted;

private void OnDownloadFinished(BackgroundTransferRequest request)

{

//Move downloaded file to its final location

Storage.MoveFile(downloadPath, targetPath);

//Notify listeners (UI) about download complete

if(DownloadFinished != null)

DownloadFinished(this, EventArgs.Empty);

//Remove the request from the queue

BackgroundTransferService.Remove(request);

}

private void OnDownloadAborted()

{

//Delete the temporary download file

Storage.DeleteFile(downloadPath);

//Notify about the aborted download

if(DownloadAborted != null)

DownloadAborted(this, EventArgs.Empty);

}

为实现在用户界面的蓝色进度条的显示，需响应传输请求**BackgroundTransferRequest**的传输进度*TransferProgressChanged*事件。

Silverlight/XNA Project: ModelViewer File: Downloads \Download.cs

public event EventHandler<DownloadProgressEventArgs> DownloadProgress;

private void request\_TransferProgressChanged(object sender, BackgroundTransferEventArgs e)

{

BackgroundTransferRequest request = e.Request;

//While the transfer is still active

if(request.TransferStatus == TransferStatus.Transferring)

{

//Notify about progress change

if(DownloadProgress != null)

DownloadProgress(this, new DownloadProgressEventArgs(request.BytesReceived, request.TotalBytesToReceive));

}

}

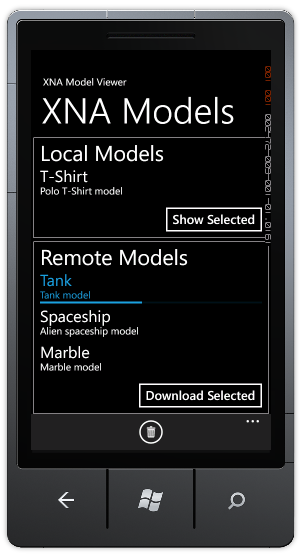


图18-2 后台传输进度条

**注意：**

当应用程序处于休眠状态或者停止运行，后台传输服务都会持续执行。在此期间，后台传输服务将所有的过程和状态事件存放在队列中，当应用程序重新激活时将接收到所有的相关更新事件。

此应用程序允许用户中止正在进行的下载。当下载开始，远程模型列表中的下载按钮显示“X”，表示停止下载。用户点击该按钮，**ModelMetadata**类将调用Download类的*Abort*方法停止下载。

在MainPage.xaml中远程模型列表ListBox，绑定SelectionChanged事件处理函数和Item模板ModelDT。

Silverlight/XNA Project: ModelViewer File: MainPage.xaml

<ListBox x:Name="lstLocalModels" ItemsSource="{Binding LocalModels}"

SelectionChanged="lstLocalModels\_SelectionChanged" ItemTemplate="{StaticResource ModelDT}" Margin="0" Grid.Row="1" BorderBrush="Red" />

在MainPage.xaml中定义列表项的数据模板，数据模板中的包含绑定Name和Description的TextBlock，以及下载进度条ProgressBar。

Silverlight/XNA Project: ModelViewer File: MainPage.xaml

<DataTemplate x:Key="ModelDT">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="280" />

<ColumnDefinition Width="60" />

</Grid.ColumnDefinitions>

<Grid Grid.Column="0" Margin="0,0,0,2" VerticalAlignment="Top">

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" MinHeight="7" />

</Grid.RowDefinitions>

<TextBlock Text="{Binding Name}" Grid.Row="0" FontSize="{StaticResource PhoneFontSizeLarge}" Margin="12,-2,12,0" HorizontalAlignment="Left" VerticalAlignment="Top" />

<TextBlock Text="{Binding Description}" Margin="12,-5,12,0" HorizontalAlignment="Left" Grid.Row="1" VerticalAlignment="Top" />

<ProgressBar Minimum="0" Maximum="100"

Value="{Binding DownloadProgress}"

Visibility="{Binding IsInProgress, Converter={StaticResource BoolToVisibilityConverter}}" Margin="0" Grid.Row="2" MinHeight="5" VerticalAlignment="Top" Width="468"

/>

</Grid>

<Button

Grid.Column="1"

Margin="0"

Padding="0"

Click="abortButton\_Click"

BorderThickness="0"

Foreground="{StaticResource PhoneAccentBrush}"

Width="60"

Height="60"

Visibility="{Binding IsInProgress, Converter={StaticResource BoolToVisibilityConverter}}"

>

<Border

Height="30"

BorderBrush="{StaticResource PhoneAccentBrush}"

BorderThickness="2"

CornerRadius="15"

Padding="10,2,10,4"

>

<TextBlock

Text="X"

FontSize="{StaticResource PhoneFontSizeSmall}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

/>

</Border>

</Button>

</Grid>

</DataTemplate>

Download类的*Abort*方法实现取消下载的功能，*Abort*方法调用后台传输服务检索定制*requestId*的后台传输服务请求，并从后台传输服务的队列中删除请求。

Silverlight/XNA Project: ModelViewer File: Downloads \Download.cs

public bool Abort()

{

BackgroundTransferRequest request = BackgroundTransferService.Find(requestId);

if(request != null)

{

isAborted = true;

BackgroundTransferService.Remove(request);

}

return request != null;

}

**注意：**

后台传输服务API只提供应用程序访问自己的传输服务请求队列，不能访问其他的应用程序的传输请求队列。

当后台传输服务请求**BackgroundTransferRequest**从队列中删除时，后台传输服务**BackgroundTransferRequest**触发*TransferStatusChanged*，并将传输状态属性设置为*TransferStatus.Completed*。由于下载完成的传输状态属性也为*TransferStatus.Completed*，所以在Download类的*Abort*方法中设置isAborted为True。以此来区分下载取消和下载完成。

Silverlight/XNA Project: ModelViewer File: Downloads \Download.cs

private void request\_TransferStatusChanged(object sender, BackgroundTransferEventArgs e)

{

BackgroundTransferRequest request = e.Request;

if(request.TransferStatus == TransferStatus.Completed)

{

request.TransferStatusChanged -= request\_TransferStatusChanged;

request.TransferProgressChanged -= request\_TransferProgressChanged;

if(isAborted)

OnDownloadAborted();

else

OnDownloadFinished(request);

}

}

Windows Phone Mango限制后台传输请求队列大小为5项，向队列中添加更多的请求会导致应用程序异常。

**DownloadManager**类管理下载队列，**DownloadManager**的*StartDownload*和*ProcessPendingDownloads*方法演示如何使用队列。

Silverlight/XNA Project: ModelViewer File: Downloads \DownloadManager.cs

private static readonly Collection<Download> downloads = new Collection<Download>();

private static readonly Queue<Download> pendingDownloads = new Queue<Download>();

public static void StartDownload(Download download)

{

if(BackgroundTransferService.Requests.Count() < 5)

{

download.DownloadFinished += download\_DownloadFinished;

download.DownloadAborted += download\_DownloadAborted;

download.Start();

}

else

{

pendingDownloads.Enqueue(download);

}

}

private static void ProcessPendingDownloads()

{

if(pendingDownloads.Count > 0)

{

Download download = pendingDownloads.Dequeue();

StartDownload(download);

}

}

当应用程序请求的下载数量达到极限时，下载请求将保存在*pendingDownloads*队列而不是**BackgroundTransferService**队列。本例中 *pendingDownloads*队列在应用程序逻辑删除或者关闭时将被删除。

### 加载XNA 3D模型

用户选择本地模型后，应用程序导航至GamePage显示XNA模型，本节讲解如何创建Silverlight/XNA混合应用程序。

**SharedGraphicsDeviceManager**类是其核心，此类应用XNA的渲染代替Silverlight的呈现模式。安装Windows Phone Mango开发工具集后，两个Silverlight和XNA的混合应用程序模板增加到Visual Studio：**Silverlight for Windows Phone**的模板**Windows Phone 3D Graphics Application**和**XNA Game Studio 4.0**的模板**Windows Phone Rich Graphics Application (4.0)**。

**注意：**

Windows Phone 3D Graphics Application和Windows Phone Rich Graphics Application (4.0)模板的改进：首先，SharedGraphicsDeviceManager声明在ApplicationLifetimeObjects收集App.xaml.cs。 其次， 应用程序类实现了IServiceProvider接口，以模拟XNA应用程序的行为，该方法返回的对象IServiceProvider.GetService驻留在ApplicationLifetimeObjects集合。

SharedGraphicsDeviceManager指示操作系统使用XNA渲染而不是Silverlight呈现，Windows Phone Mango的GameTimer类运行XNA引擎并为应用程序提供控制**Microsoft.Xna.Framework.Game**生命周期的方法。SpriteBatch类负责基本的视觉元素，能有效地绘制多个子画面，包括绘制背景图像和Silverlight内容。

声明UIElementRenderer。在应用程序加载时初始化，UIElementRenderer在timer\_Draw方法中绘制Silight UI。UIElementRenderer将Silverlight组件渲染为XNA 2D纹理，并触发Silverlight事件处理函数。例如，当用户触控UIElementRenderer渲染的Silverlight按钮，UIElementRenderer传递触控的信息给Silverlight按钮。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

public partial class GamePage : PhoneApplicationPage

{

//XNA Setup

GameTimer timer;

SpriteBatch spriteBatch;

//Scene background

Texture2D background;

//Model & metadata

XnaModelWrapper model;

ModelMetadata modelMetadata;

//Silverlight UI rendering

UIElementRenderer uiRenderer;

……

}

在GamePage构造函数中创建游戏定时器，初始化手势触控的支持，且必须指定支持的手势类型。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

public GamePage()

{

InitializeComponent();

// Create a timer for this page

timer = new GameTimer();

timer.UpdateInterval = TimeSpan.FromTicks(333333);

timer.Update += timer\_Update;

timer.Draw += timer\_Draw;

//Initialize gestures support - Pinch for Zoom and horizontal drag for rotate

TouchPanel.EnabledGestures = GestureType.FreeDrag | GestureType.Pinch | GestureType.PinchComplete;

this.LayoutUpdated += GamePage\_LayoutUpdated;

this.DataContext = this;

}

要在Silverlight中显示XNA 3D模型不可避免要使用XNA渲染。在**GamePage.xaml.cs**的*OnNavigatedTo*方法中，调用SharedGraphicsDeviceManager类的SetSharingMode实现XNA渲染，初始化模型并启动游戏定时器。

应用程序需要**ContentManager**在运行时加载XNA资源，XNA所需的**IServiceProvider**接口已具备，只需创建**ContentManager**加载纹理和图片。从3D模型的*modelMetadata*中重新载入默认的xRotation和yRotation。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

protected override void OnNavigatedTo(NavigationEventArgs e)

{

// Set the sharing mode of the graphics device to turn on XNA rendering

SharedGraphicsDeviceManager.Current.GraphicsDevice.SetSharingMode(true);

//Initialize SpriteBatch

spriteBatch = new SpriteBatch(SharedGraphicsDeviceManager.Current.GraphicsDevice);

App app = (App)Application.Current;

ContentManager appContentManager = new ContentManager(app, "Content");

background = appContentManager.Load<Texture2D>("background");

//Get query string parameter and initialize local metadata variable

IDictionary<string, string> data = NavigationContext.QueryString;

modelMetadata = app.ModelsStore[data["ID"]];

ModelName = modelMetadata.Name;

ModelDesc = modelMetadata.Description;

xRotation = modelMetadata.DefaultXRotation;

yRotation = modelMetadata.DefaultYRotation;

//Initialize the model

model = new XnaModelWrapper();

model.Lights = new bool[] { true, true, true };

ContentManager contentManager = modelMetadata.IsContent ? appContentManager : new CustomContentManager();

model.Load(contentManager, modelMetadata.Assets[0]);

// Start the game timer

timer.Start();

base.OnNavigatedTo(e);

}

在OnNavigatedFrom方法中关闭XNA渲染，关闭游戏定时器。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

protected override void OnNavigatedFrom(NavigationEventArgs e)

{

// Stop the game timer

timer.Stop();

// Set the sharing mode of the graphics device to turn off XNA rendering

SharedGraphicsDeviceManager.Current.GraphicsDevice.SetSharingMode(false);

base.OnNavigatedFrom(e);

}

在timer\_Draw方法中使用SpriteBatch 绘制二维图形，调用XnaModelWrapper绘制三维模型。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

private void timer\_Draw(object sender, GameTimerEventArgs e)

{

SharedGraphicsDeviceManager.Current.GraphicsDevice.Clear(Color.CornflowerBlue);

spriteBatch.Begin();

spriteBatch.Draw(background, Vector2.Zero, Color.White);

spriteBatch.End();

// Set render states.

SharedGraphicsDeviceManager.Current.GraphicsDevice.DepthStencilState = DepthStencilState.Default;

SharedGraphicsDeviceManager.Current.GraphicsDevice.BlendState = BlendState.Opaque;

SharedGraphicsDeviceManager.Current.GraphicsDevice.RasterizerState = RasterizerState.CullNone;

SharedGraphicsDeviceManager.Current.GraphicsDevice.SamplerStates[0] = SamplerState.LinearWrap;

// Draw the model

model.Draw();

// Update the Silverlight UI

uiRenderer.Render();

// Draw the sprite

spriteBatch.Begin();

spriteBatch.Draw(uiRenderer.Texture, Vector2.Zero, Color.White);

spriteBatch.End();

}

3D模型的更新在*timer\_Update*事件处理中实现。当GameTimer执行Draw事件后应用程序调用Update事件执行3D模型的逻辑计算。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

private void timer\_Update(object sender, GameTimerEventArgs e)

{

HandleInput();

float yaw = MathHelper.Pi + MathHelper.PiOver2 + xRotation / 100; // rotation around the y-axis

float pitch = yRotation / 100; // rotation around the x-axis

float fieldOfView = MathHelper.ToRadians(cameraFOV) / modelMetadata.FieldOfViewDivisor; // zoom

model.Rotation = modelMetadata.World \* Matrix.CreateFromYawPitchRoll(yaw, pitch, 0);

model.Projection = Matrix.CreatePerspectiveFieldOfView(fieldOfView, modelMetadata.AspectRatio, modelMetadata.NearPlaneDistance, modelMetadata.FarPlaneDistance);

model.View = modelMetadata.ViewMatrix;

model.IsTextureEnabled = true;

model.IsPerPixelLightingEnabled = true;

}

在HandleInput方法中读入手势触控的xRotation，yRotation，cameraFOV和prevLength。

Silverlight/XNA Project: ModelViewer File: GamePage.xaml.cs

private void HandleInput()

{

while (TouchPanel.IsGestureAvailable)

{

GestureSample gestureSample = TouchPanel.ReadGesture();

switch (gestureSample.GestureType)

{

case GestureType.FreeDrag:

xRotation += gestureSample.Delta.X;

yRotation -= gestureSample.Delta.Y;

break;

case GestureType.Pinch:

float gestureValue = 0;

float minFOV = 80;

float maxFOV = 20;

float gestureLengthToZoomScale = 10;

Vector2 gestureDiff = gestureSample.Position - gestureSample.Position2;

gestureValue = gestureDiff.Length() / gestureLengthToZoomScale;

if (null != prevLength) // Skip the first pinch event

cameraFOV -= gestureValue - prevLength.Value;

cameraFOV = MathHelper.Clamp(cameraFOV, maxFOV, minFOV);

prevLength = gestureValue;

break;

case GestureType.PinchComplete:

prevLength = null;

break;

default:

break;

}

}

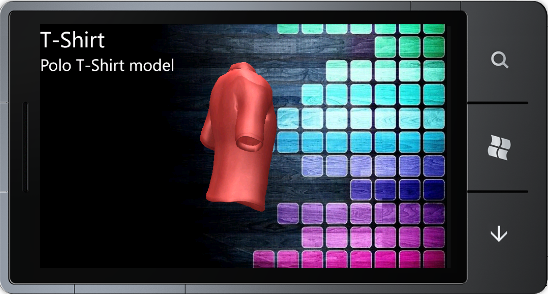
}

### 在模拟器中运行

按F5运行应用程序，或者点击Start Debugging按钮运行，如图18-3 Start Debugging。



图18-3 Start Debugging

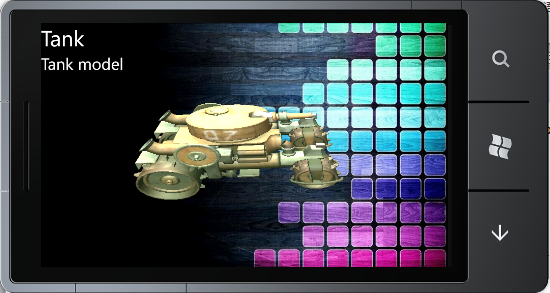
 

图18-4 XNA 3D模型展示