2D games using Silverlight - implementing the game loop

This article shows how to create a games loop in a Microsoft Silverlight application, which once created on Windows Phone 7, should be able to run with minimal modification on Windows Phone 8 and Windows 8 devices. It is the first article in a series which will show how to create a complete working game (a clone of the classic Arkanoid game).

Introduction

The recommendation for Windows 8 and Windows Phone 8 is that new apps be written in the new Metro style app so they can run largely unchanged across both platforms, while games should be written using DirectX and C++ to minimize the porting effort. However if you want also to target Windows Phone 7 with your application then using DirectX and C++ is not an option.

This article outlines an option to allow you to create games that are capable to run on all three platforms using Microsoft Silverlight. Silverlight is not a game framework but it is suitable for game creation since it is Rich Internet Application framework. This might be appropriate for simple games where you need to target all three platforms and investing in both XNA and DirectX/C++ is more costly than maintaining a single codeline.

Microsoft Silverlight does not guarantee the frame rate (in the same way that XNA does) and hence for some games and on some platforms the performance may differ; there may also be other limitations. However for the game used in this example we haven't run into any issues on any current Windows Phone 7 devices, including those with reduced memory and CPU.

For any game we need :

1. Game loop
2. Sprite animation
3. Collision detection

This article explores the first part of the problem - creating a game project application skeleton with a game loop. The second article in the series will extend the application with more objects and explain object collision detection.

JailBreaker (Arkanoid clone) running on a PC

Game project

The game project contains source code and some related information or this article. Unfortunately the project does not contain installation package for a quick demo because Windows Phone does not allow application side loading. Please check supplementary application project to how to build and run the application on your device(readme.txt).

If you don't have a device and Microsoft developer id you can still test the code using the device emulator in the SDK. Note however that the emulator cannot simulate accelerometer changing for users to play the game. You can emulate device shaking only, but this is not a good mechanism for game control.

Note: Good news for students - you can get developer id for free of charge!

Please check this on how to obtain Microsoft developer id and setup development environment. Also Useful links: Windows Phone development

Implementation & design

Game rendering loop

The game rendering loop needs to be run in such a way that the app remains responsive to user input. In this example we hook into the CompositionTarget.Rendering . event inside Windows Presentation Framework's (WPF) rendering loop. This event is fired once each time WPF decides to render a frame. This is by far the simplest solution, and appropriate for this case where we have fast moving objects with little position calculation (accelerometer data).

(C) Copyright Nokia 2014. All rights reserved.

Other alternatives that were considered were to use timer tick animation or a completely separate thread. Timer tick animation would work, but results in "jerky" (non-smooth) movement (see StoryBoard class for more information on timer based animation). Using a separate thread is also a good approach, but the complexity is not justified in this case because the amount of calculation required is trivial.

When creating the game physics engine (covered in the next article) we will consider using a worker thread or `StoryBoard class`.

Target device constraints

We are targeting the Windows Phone device with the poorest specification: Lumia 610. This device lacks most of sensors and has memory limit of 256 Mb. As a result we cannot use the Motion class, but instead must use the accelerometer sensor. The accelerometer works badly in all directions except when device is in portrait mode.

Code walkthrough

Examine the following structure in the project source code:

- Main application (MainPage.xml, MainPage.xaml.cs)
  - Game board container (BoardControl.xml, BoardControl.xaml.cs. Also it contains components below)
    - Game loop (function `void gameLoopHandler(object sender, EventArgs e)`)
      - Controlled object (BoardControl.xml, XAML definition **<Rectangle Name="bouncer" ...../>**)

1. in `BoardControl` container find line that register rendering handler:

```
CompositionTarget.Rendering += new EventHandler(gameLoopHandler);
```

2. Then find the handler implementation:

```
        void gameLoopHandler(object sender, EventArgs e)
        {
            if (state != STATE.RUNNING)
                return;

            handleAccelerometer(sender, e);
        }
```

3. Note, function `handleAccelerometer(sender, e)` only reads the class member variable `Vector3 acceleration` for accelerometer reading. This variable is written in separate thread via `void accelerometerDataChanged(object sender, SensorReadingEventArgs<AccelerometerReading> e)` handler.
4. Again to subscribe for accelerometer's data reading there is line:

```
accelerometer.CurrentValueChanged += new
EventHandler<SensorReadingEventArgs<AccelerometerReading>>(accelerometerDataChanged);
```

5. all the rest of the code implementation is related to the application auxiliary supporting as suspending gameplay when control is out of the screen (note we use Pivot control as the main page) and resuming it when it is visible again.
6. the last thing that may look wierd to you is suppression possible exceptions in **onLoad** handler. That is for WPF Designer. WPF Designer panel appears on the right side of XAML document opened in Visual Studio. The panel displays live-view of your page by running virtual machine. WPF Designer can through exception while your User Control instantiation even your code will work fine on device or emulator. There is a guide from Microsoft about Troubleshooting WPF and Silverlight Designer Load Failures. The main rule is: not place anything that might throw an exception into class constructor. And that is **onLoad** handler for. But still my preview is broken on accelerometer instantiation - that is why exception handling is here.

Tip: when you type in Microsoft Visual Studio editor, along with context sensitive help there is a code generator available.

Try it. Type `accelerometer.CurrentValueChanged` + and when you start typing = sign to assign a handler, see pop-up hint allows

(C) Copyright Nokia 2014. All rights reserved.

you to generate the handler code

Profiling performance

The approach outlined in this article hooks into the `CompositionTarget.Rendering` event, which provides no guarantee on the frame refresh rate. To measure the impact of platform on refresh rate we've added a refresh rate measurement in the upper-right corner (displays milliseconds between frames).

Measurements using this approach show that the screen refresh rate on a PC desktop browser is about 16 milliseconds, while on Nokia Lumia 610 the rate is about 48 milliseconds. As result if you do not adjust object movement to current screen refresh rate, movement is considerably slow on Lumia 610.

The screen rate measurement is a part of the physics engine implementation that we explain in our next wiki article.

Running the game in a browser

Microsoft Silverlight apps can run on Windows Phone 7.X device and on a PC desktop browser with minimal modifications. However there is no sensor support in a PC desktop browser so you should implement different user input control when the application is to be run embedded in a browser.

You will also have to modify code to exclude .NET assemblies that exist on Windows Phone 7 but do not exist for Silverlight desktop applications. Fortunately .NET supports conditional compilation so you can target at least two platforms, Windows Phone 7 and desktop application with single code base. Please check supplementary application project for details(readme.txt).

Porting to Microsoft Visual Studio 2012

Microsoft Visual Studio 2012 development tool does not include Windows Phone 7 or (Zune Phone) target. However you can run your WP7 Silverlight application on Windows 8 device with minimal modification -- only renaming namespaces and the project rebuilding is required. To build your project for desktop browser you need to install Microsoft Visual Studio Express 2012 for WEB. After installation Microsoft Visual Studio Express 2012 for WEB, original Microsoft Visual Studio Express 2010 for Windows Phone may get broken but you can use Microsoft Expression Blend 4 without debugging facilities though. Microsoft Expression Blend 4 comes with Windows Phone 7 SDK for free.

Summary

This article has shown how to create a games loop by hooking into the `CompositionTarget.Rendering` event. The next wiki article in the series is 2D games using Silverlight - Collision detection implementation.

(C) Copyright Nokia 2014. All rights reserved.