Windows 8 开发 31 日

第 02 日

Orientation 和 snap

译者：BeyondVincent(破船)

时间：2013.4.23

版本： 2.0

关于破船

程序猿砌墙于云南昆明!

长期扎根移动软件开发!

爱跑步爱打篮球爱运动!

命中无大富大贵之面相!

愿健康与平淡相随一生!

你可以发邮件与破船取得联系: BeyondVincent@gmail.com

还可以关注破船的微博: 腾讯微博和新浪微博。

这里是破船的个人博客，欢迎光临：破船之家

关于 Windows 8 开发 31 日翻译

Windows 8 开发 31 日是由 Jeff Blankenburg 和 Clark Sell 原创的。

官方站点：http://31daysofwindows8.com/

涉及到两个版本：

XAML/C#（由 Jeff Blankenburg 撰写）

HTML5/JS （由 Clark Sell 撰写）

其中涉及到的资源和相关代码请到这里下载：

https://github.com/csell5/31DaysOfWindows8

在这里，由于破船对 HTML5/JS 不熟悉，所以只翻译 XAML/C#相关主题。

建议大家前往看原创内容，如果看不明白，再来这里看我翻译的相关内容。

如果翻译不正确的地方，可以通过上面的联系方式告诉破船。

破船祝你阅读愉快！

目录

关于破船 2

关于 Windows 8 开发 31 日翻译 3

目录 4

第 02 日 Orientation 和 snap. 5

1.0. 介绍 5
1.1. Supporting Rotation 6
1.2. 识别 Orientation 改变 7
1.3. 远程调试 10
1.4. 回到代码 13
1.5. 总结 17

第 02 日 Orientation 和 snap

1.0. 介绍

今天我们来谈谈屏幕尺寸，以及为什么在 Windows 8 开发中，它很重要。本文的第一部分，我将讨论 orientation，基于用户手持设备的方法，使用一些简单的方法，我可以让我的程序更加有用。第二部分，我将介绍在“snapped”状态下的应用程序，以及如何修改界面以适应更小的屏幕尺寸。

Orientation 和 snap 非常重要：如果在程序中，你不考虑它们，那么你的程序不会被提交到 Windows Store 中。

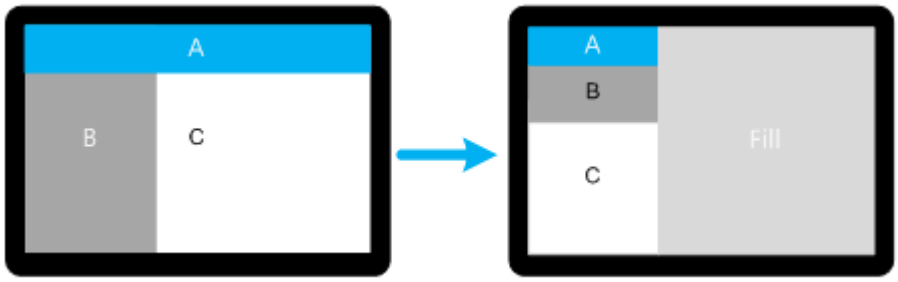
下面是 Windows 8 app certification requirements 中的 section3.6：

Your app must support a snapped layout. In landscape orientation, your app’s functions must be fully accessible when the app’s display size is 1024 x 768. Your app must remain functional when the customer snaps and unsnaps the app.

这里说的是我们的应用程序需要支持 3 中可是状态，最低要求：

- 1024 x 768 (最小的屏幕分辨率& filled state)
- 320 x 768 (snapped)
- 默认的分辨率一般是 1366 x 768

下面的例子是：全屏的 app 切换到 snapped



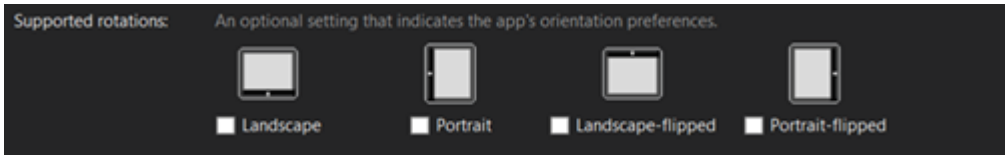
从上图可以看到，当为 snapped 是，我对内容进行了重新布局，以适应 snapped 状态。这里同样可以将程序切换到“filled”状态。

很幸运的是，这里提供了简单的方法来识别程序进入了哪种状态，本文剩下的内容将进行详细的介绍。

1.1. Supporting Rotation

首先，在 Visual Studio 2012 中用 blank template 创建一个应用程序。创建好之后，立即将程序运行在模拟器或者远程机器上。你会发现当你旋转模拟器或设备的时候，程序也会自动的旋转。为什么？如何做到的？

默认情况下，Visual Studio 中的所有 template 都支持全部旋转。还记得 package.appxmanifest 文件吗？在其中的应用程序 UI(Application UI) tab 中，你会发现这里有一项为支持的旋转。勾选一个或者多个 orientation，那么程序即支持所勾选的 orientation。不过默认情况下是全部都支持的。



跟据你的具体情况，选择支持不同的 orientation。例如，创建一个游戏应用，可能只想支持 landscape 模式。

1.2. 识别 Orientation 改变

关于 orientation，我们的第一个任务就是当 orientation 改变的时候，可以识别出来。幸运的时，Windows 8 SDK 为我们提供了一个方向传感器类 SimpleOrientationSensor，该类有相关的事件提供我们使用。

首先，我在 MainPage.xaml 文件中添加一个简单的 TextBox。下面是当前 MainPage.xaml 的完整代码：

```
<Page
x:Class="Day2_OrientationAndSnap.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="using:Day2_OrientationAndSnap"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d">
<Grid Background="{StaticResourceApplicationPageBackgroundThemeBrush}">
<TextBlock Text="No orientation reading."
x:Name="AlertBox"
FontSize="50"
TextAlignment="Center"
  Margin="0,100,0,0" />
</Grid>
</Page>
```

先在，打开 MainPage.xaml.cs 文件。我们需要添加一些代码来使用这个方向传感器。

首先，添加一个新的名称空间：using Windows.Devices.Sensors;。下一步是添加一个 SimpleOrientationSensor 类的实例对象，并添加一些事件，来识别 orientation 的改变。下面给出 MainPage.xaml.cs 文件完整的代码。之后，我会解释这些代码。

```
using System;
usingSystem.Collections.Generic;
using System.IO;
usingSystem.Linq;
usingWindows.Foundation;
usingWindows.Foundation.Collections;
usingWindows.UI.Xaml;
usingWindows.UI.Xaml.Controls;
usingWindows.UI.Xaml.Controls.Primitives;
usingWindows.UI.Xaml.Data;
usingWindows.UI.Xaml.Input;
usingWindows.UI.Xaml.Media;
usingWindows.UI.Xaml.Navigation;
usingWindows.Devices.Sensors;
namespace Day2_OrientationAndSnap
{
publicsealedpartialclassMainPage : Page
      {
privateSimpleOrientationSensororientationSensor;
publicMainPage()
            {
this.InitializeComponent();
orientationSensor = SimpleOrientationSensor.GetDefault();
            }
protectedoverridevoidOnNavigatedTo(NavigationEventArgs e)
            {
if (orientationSensor != null)
orientationSensor.OrientationChanged += newTypedEventHandler<SimpleOrientationSensor,
SimpleOrientationSensorOrientationChangedEventArgs>(orientationSensor_OrientationChanged);
            }
protectedoverridevoidOnNavigatingFrom(NavigatingCancelEventArgs e)
            {
if (orientationSensor != null)
orientationSensor.OrientationChanged -= orientationSensor_OrientationChanged;
base.OnNavigatingFrom(e);
            }
asyncprivatevoidorientationSensor_OrientationChanged(SimpleOrientationSensor sender,
```

```
SimpleOrientationSensorOrientationChangedEventArgsargs)
            {
await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, () =>
                {
ShowOrientationText(args.Orientation);
                });
            }
privatevoidShowOrientationText(SimpleOrientationsimpleOrientation)
            {
switch (simpleOrientation)
                {
caseSimpleOrientation.NotRotated:
AlertBox.Text = "Not Rotated";
break;
caseSimpleOrientation.Rotated90DegreesCounterclockwise:
AlertBox.Text = "90 Degrees CounterClockwise";
break;
caseSimpleOrientation.Rotated180DegreesCounterclockwise:
AlertBox.Text = "180 Degrees Rotated";
break;
caseSimpleOrientation.Rotated270DegreesCounterclockwise:
AlertBox.Text = "270 Degrees Rotated CounterClockwise";
break;
caseSimpleOrientation.Facedown:
AlertBox.Text = "Face Down";
break;
caseSimpleOrientation.Faceup:
AlertBox.Text = "Face Up";
break;
default:
AlertBox.Text = "Unknown";
break;
                }
            }
        }
}
```

首先,创建了一个新的 SimpleOrientationSensor 对象,名称为 orientationSensor。在 MainPage() 构造函数中,使用设备中默认的方向传感器来实例化 orientationSensor。

在 OnNavigatedTo() 和 OnNavigatingFrom() 事件处理中,我分别在 orientationSensor 上添加和移除一个 OrientationChanged 事件。首先要保证该对象不

能为 null，因为在设备中，有可能没有这个传感器。

下面，是一个 orientationSensor_OrientationChanged()事件处理函数。注意，在函数名最前面有一个 async 描述符，当使用 await 关键字时，需要这个描述符。这样的目的是避免应用程序被阻塞。(可以在 MSDN 上查阅更多 async/await 相关知识)。

一旦获得了数据，就调用方法 ShowOrientationText()，并将 orientation 数据传递过去。

最后，ShowOrientationText()方法写了一个简单的 switch 语句，获取所有可能存在的 orientation：

- NotRotated
- Rotated90DegreesCounterclockwise
- Rotated180DegreesCounterclockwise
- Rotated270DegreesCounterclockwise
- Facedown
- Faceup
- Unknown.

1.3. 远程调试

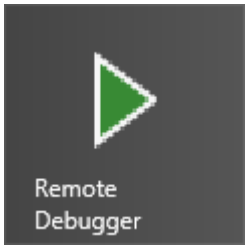
如果你的设备跟 Clark 和我的类似，用 4 核的桌面机器，8-12GB 的 RAM，27 寸显示器，鼠标，键盘。那么，不幸的时，这样的机器没有方向传感器，改变显示器的方向，不会发生方向改变的事件。我在这里也不是用模拟器来模拟方向。我在这里使用一台真实的设备来进行调试。

注意：关于模拟器，并不像 Windows Phone 中的模拟器。它只能模拟器你当

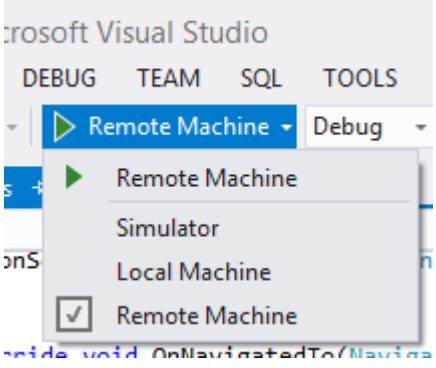
前是用的机器，并没有完整的功能。

首先，在目标设备上安装 Remote Debugging Tools，在这里我使用的是 Samsung Series 7 Slate。可以在这里下载 Remote Debugging Tools。记得根据你的设备，选择适合的版本：x84、x64 或 ARM。

在目标机器上运行 Remote Debugging Tools。可以看到类似如下的一个图标：

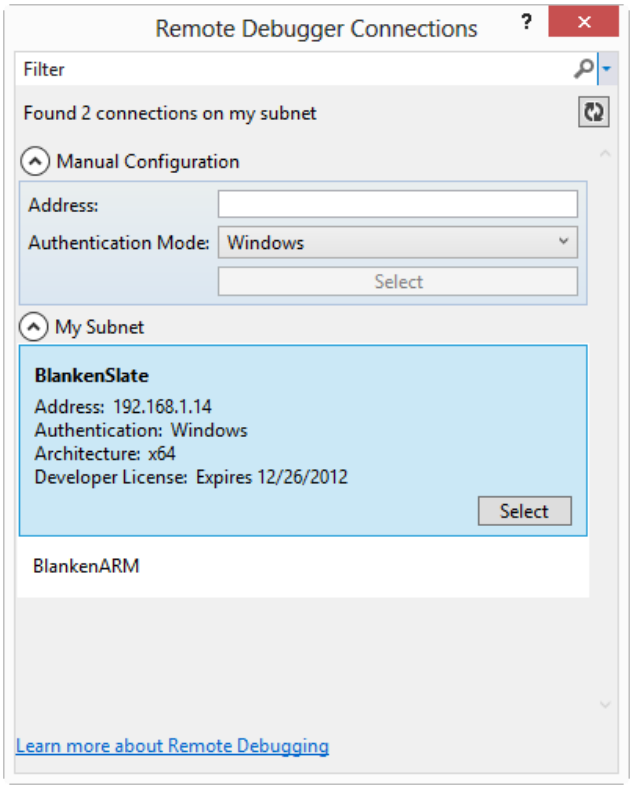


在目标设备上运行起 Remote Debugging Tools 之后，回到主机器上，选择“Remote Machine”作为部署的目标。

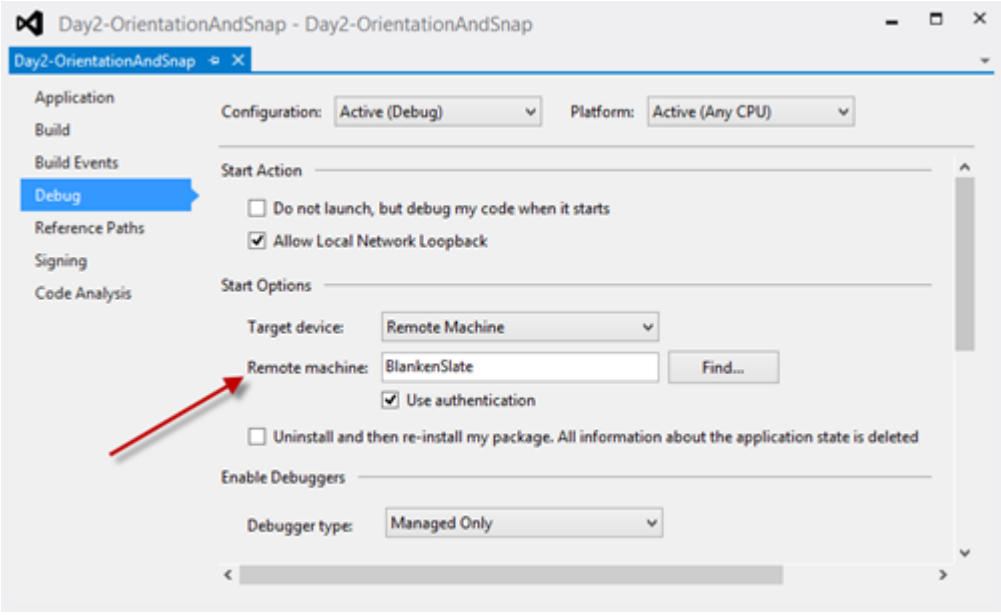


当你首次选择“Remote Machine”时，会弹出如下一个对话框。记住，只有当目标机中安装并运行了 Remote Debugger Tools，才会在子网(subnet)中找到目标设

备。



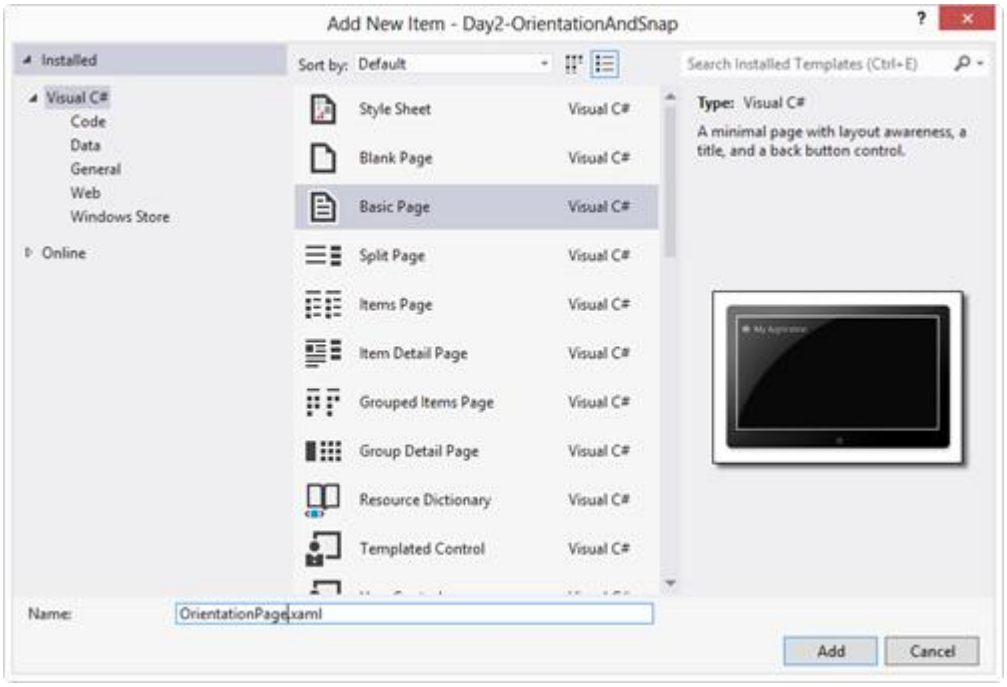
之后，如果想要切换目标设备，可以打开工程属性(Alt+Enter)，并选择 Debug tab。在这里，可以改变和移除之前选择的设备。如果移除了设备，那在下次选择 remote debug 时，同样会弹出上面的那个图。



1.4. 回到代码

现在，我的程序可以支持设备 orientation 的改变了。在方向改变的时候，我们可以写一些具体的代码，但是如果我们想要在方向改变时，重新布局 UI 界面，让用户易用，怎么办呢？这里提供了一个非常简单的方法：VisualStateManager。

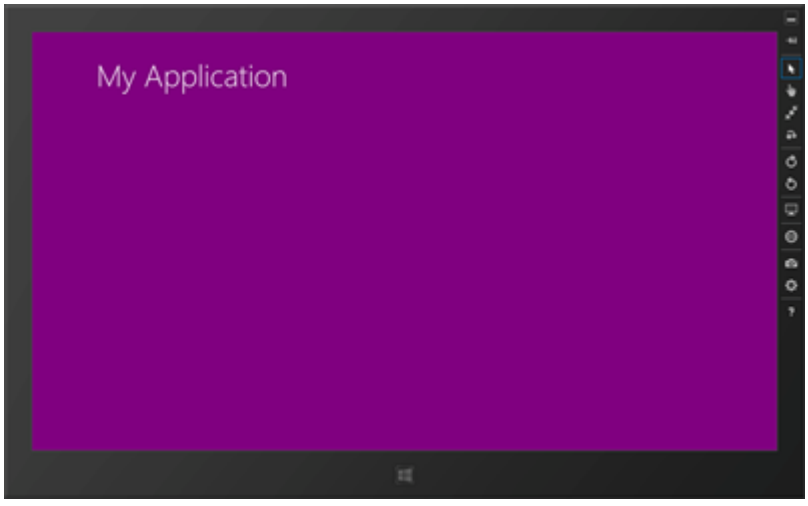
回到工程，右键单击，选择 Add…New Item…对话框。



添加一个新的 Basic Page。命名为 OrientationPage.xaml。通过查看它的 xaml 代码，会发现这是一个完全不同的 page。这是一个 LayoutAwarePage，默认情况下，它已经适配了一个 orient，并且提供了一个 VisualState 以支持 snapped。

另外，使用这种类型的 page，模拟器也会考虑 orientation 改变。

使用这种类型的 page，意味着，我们自动的获得了一个 orientation 和 snap-aware page，它们静态的设置在 visual states 中，当然，我们不一定就要这样操作。为了更加的明显，我将修改 visual states——每一个方向或状态设置不同的背景色。


My Application


My Application

下面是在 OrientationPage.xaml 中修改的 VisualState 值：

```
<VisualStateManager.VisualStateGroups>
<VisualStateGroup x:Name="ApplicationViewStates">
<VisualState x:Name="FullScreenLandscape">
<Storyboard>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="layoutRoot"Storyboard.TargetProperty="Background"
>
<DiscreteObjectKeyFrameKeyTime="0" Value="Purple"/>
</ObjectAnimationUsingKeyFrames>
</Storyboard>
</VisualState>
<VisualState x:Name="Filled">
<Storyboard>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="layoutRoot"Storyboard.TargetProperty="Background"
>
<DiscreteObjectKeyFrameKeyTime="0" Value="Orange"/>
</ObjectAnimationUsingKeyFrames>
</Storyboard>
</VisualState>
<VisualState x:Name="FullScreenPortrait">
<Storyboard>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="layoutRoot"Storyboard.TargetProperty="Background"
>
<DiscreteObjectKeyFrameKeyTime="0" Value="DarkGreen"/>
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="backButton"Storyboard.TargetProperty="Style">
<DiscreteObjectKeyFrameKeyTime="0" Value="{StaticResourcePortraitBackButtonStyle}"/>
</ObjectAnimationUsingKeyFrames>
</Storyboard>
</VisualState>
<VisualState x:Name="Snapped">
<Storyboard>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="layoutRoot"Storyboard.TargetProperty="Background"
```

```
>
<DiscreteObjectKeyFrameKeyTime="0" Value="Blue"/>
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="backButton"Storyboard.TargetProperty="Style">
<DiscreteObjectKeyFrameKeyTime="0" Value="{StaticResourceSnappedBackButtonStyle}"/>
</ObjectAnimationUsingKeyFrames>
<ObjectAnimationUsingKeyFramesStoryboard.TargetName="pageTitle"Storyboard.TargetProperty="Style">
<DiscreteObjectKeyFrameKeyTime="0" Value="{StaticResourceSnappedPageHeaderTextStyle}"/>
</ObjectAnimationUsingKeyFrames>
</Storyboard>
</VisualState>
</VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

如你说看到的，在每一个 section 中，我添加了一个新的 node，来修改 layoutRoot(Grid)的背景值。这是 LayoutAwarePage 运行的，而不用写大量的代码来对 orientation 进行管理。

1.5. 总结

今天，我们学习了如何判断用户设备的 orientation，以及如何使用 LayoutAwarePage 来管理不同的 visual states。在网上有许多 orientation 和 snap 的示例，但是在这里，你可以知道一件事：

你的程序必须处理 snapped 状态。

点击下面的图标，可以下载到本文的完整示例：

明天，我们将学习 Splash Screen。这是非常重要的一个内容。我将在明天进行介绍。

感谢你的阅读！

如果对这篇文章有什么想法想法，可以与破船联系，破船的联系方式在文章开头。

破船