Getting Started with Unity 3D on Windows Phone

This code example demonstrates how to get started with Unity 3D on Windows Phone. It has been created from original blog content kindly donated by roguecode .

14 Apr 2013

Introduction

Unity 3D is an engine and tools which make it very easy to create cross-platform 3D games. It consists of an IDE (which has free and paid versions) and an exporter for each platform (desktop is free, but all the mobile platforms require purchase of a license).

Note: Unity 3D is available for Windows Phone 8 in beta form (27 March 2013), and is accessible through the beta program as described in the Unity 4 Beta program for Windows Phone 8 apps (Unity Blog). At time of writing it is not yet product standard - apps are watermarked and cannot be submitted to the store.

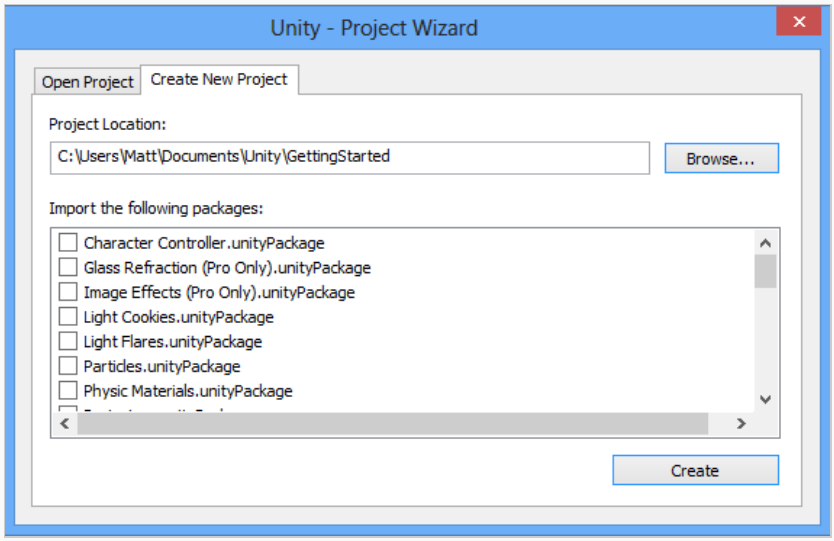
Unity 3D on Windows Phone has one great advantage when compared to other Windows platforms - you can add Windows Phone specific code on top of your Unity 3D game if needed (at the cost of breaking cross-platform compatibility). This opens up a lot of great possibilities (for example, our current game at RogueCode would be impossible without it). Note, this works because instead of generating a XAP file ready to be uploaded to the store, the exporter creates a normal Windows Phone solution that resembles the Direct3D and XAML template.

The code example discussed in the article creates a simple scene where a ball bounces around (as shown in the video below). We first shows you the basics of setting up the project, and then demonstrate how to fire events in a Unity3D C# script, and react to that in your Windows Phone project code.

The media player is loading...

Getting Started

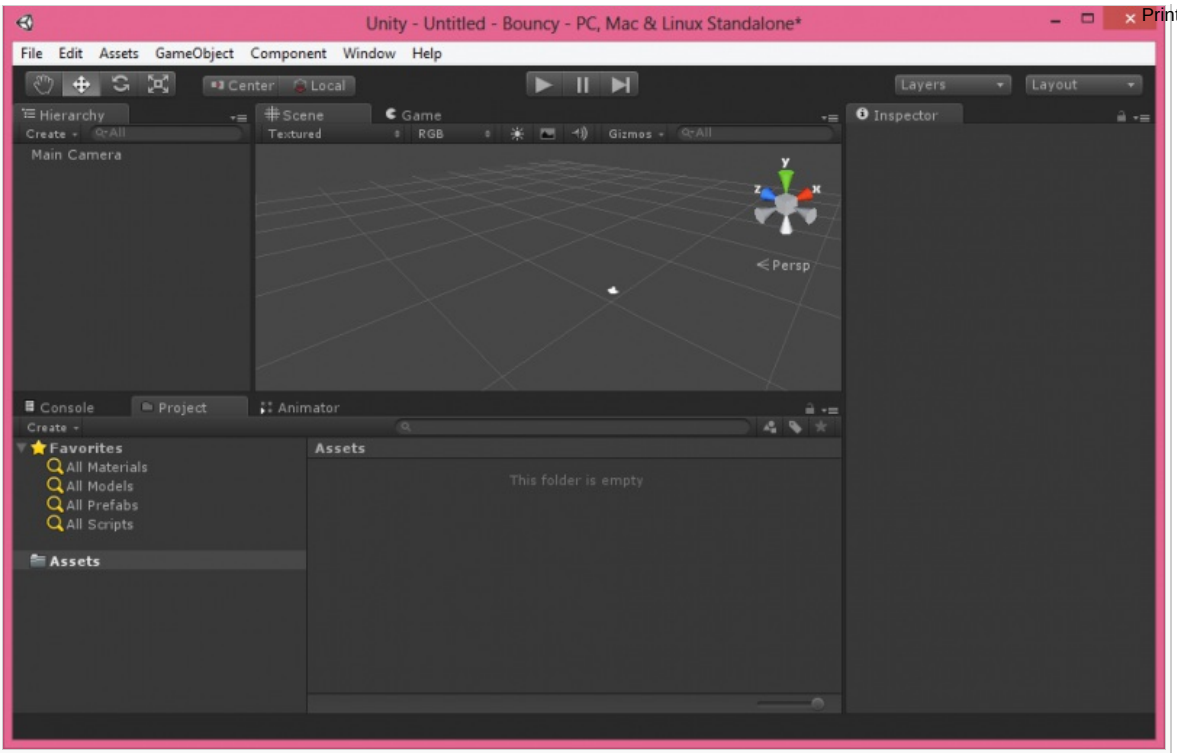
1. First create a new project in Unity: **File -> New Project**:



Create new project file in Unity

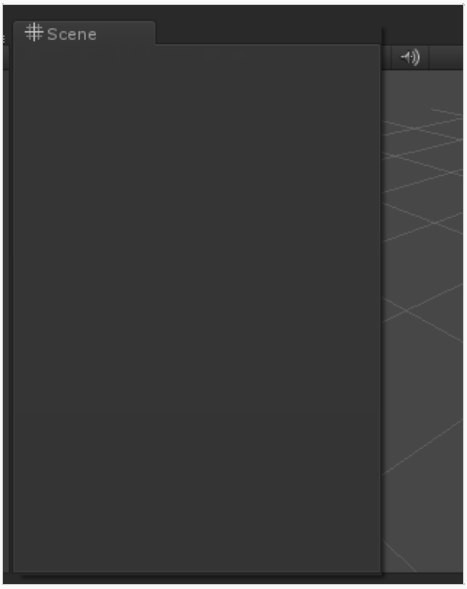
Unity will then display a new empty environment as shown below:

(C) Copyright Nokia 2013. All rights reserved.



New empty project in Unity

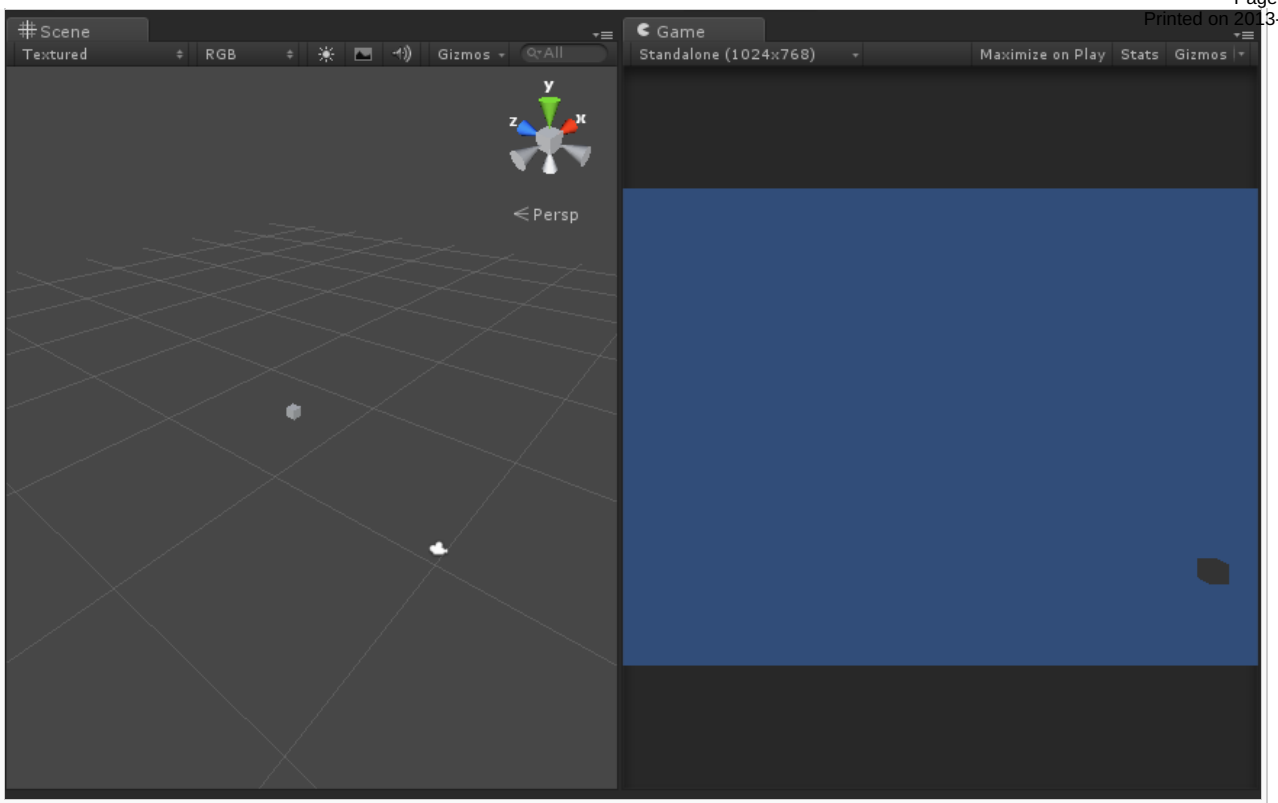
2. Then select **File –> Save** and save your current scene in the *Assets* folder. That folder is where you will keep all your scripts, models, materials, etc.
3. Re-arrange the Scene pane and the Game pane side-by-side so you can see what your camera sees while designing your scene (this is optional, but useful). Click and drag the Scene tab directly down about an inch until your see this:



Arranging scene and game panels next to each other

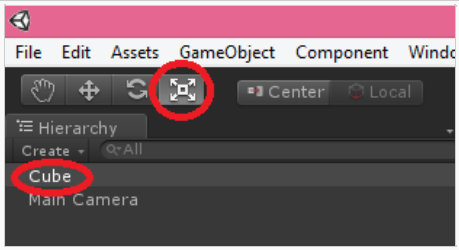
After releasing the mouse you will get this side-by-side view:

(C) Copyright Nokia 2013. All rights reserved.



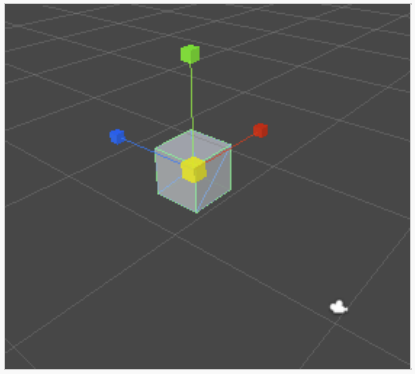
Side-by-side view

4. The Hierarchy pane on the left is where all the objects of the scene will be. Currently there is just the default camera. Let's go ahead and create a cube by clicking **[top menus] GameObject –> Create Other –> Cube**. Now with the cube selected in the Hierarchy pane, click the resize button shown above.



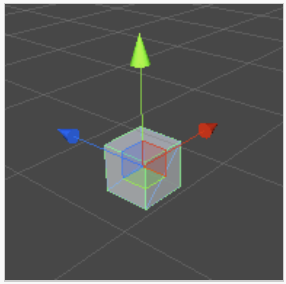
Resize cube

5. If you've ever used a 3D modelling app you will feel right at home with the manipulation tools. In the Scene pane, click and drag the centre block (which will resize the cube equally on X, Y, and Z) until the cube is a bit bigger. Notice how it grows in the Game pane too.



Resizing the cube

6. You can move objects in a similar way by clicking on the Translate button (two to the left of Resize used earlier).



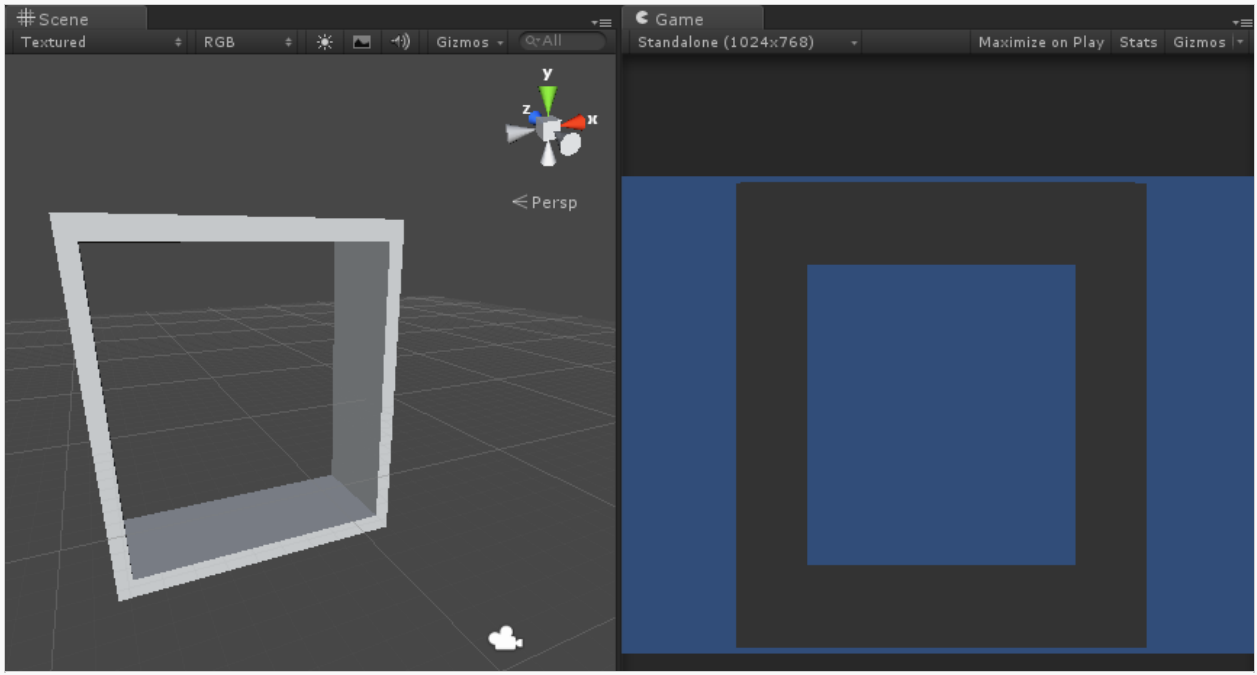
Moving the cube

7. Before building up the rest of the objects you should get used to controlling the view in the Scene pane. Remember that this is just the view that you are designing with – the final game will see objects from the perspective of the Camera object. Here are a few ways to control your view:
   - Hold down middle mouse button and drag will pan around the scene.

(C) Copyright Nokia 2013. All rights reserved.

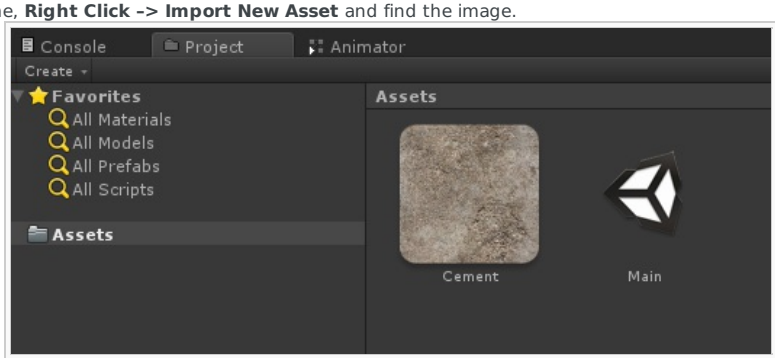
- Rolling the middle mouse button forward and back will zoom in and out.
- Holding Alt + Left mouse button and dragging will rotate around.

8. Now that you are an accomplished modeller, lets create a bunch more cubes, resize them, and move them, until you have something that resembles this next image. Each wall is a cube that has been squashed and/or stretched.



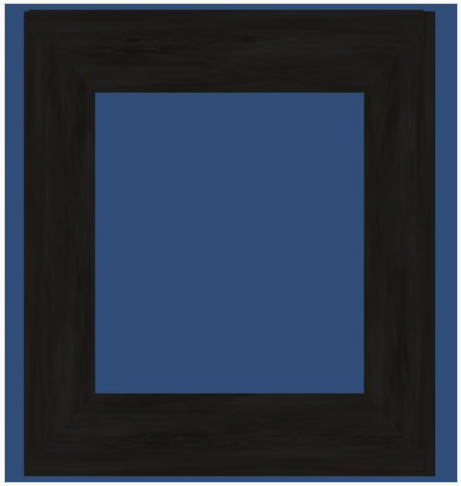
Create a new shape from a number of cube primitives

9. Notice how ugly it looks in the Game pane? That's because it has no material or pretty shader. Let's grab a random texture off the Internet to apply to these cubes. I got a free one here – but I first resized the texture to 256px X 256px. When you have a suitable image on your PC, in the bottom Project pane, **Right Click –> Import New Asset** and find the image.



Adding texture/shader

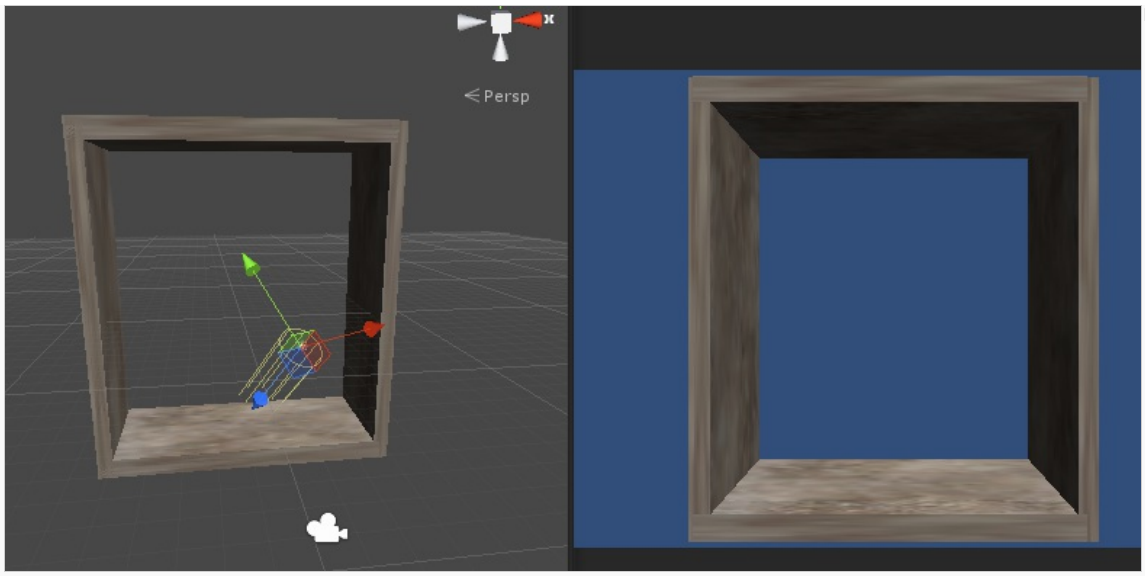
.0. Apply that texture to each cube by dragging it from the bottom onto the cubes either in the Scene pane, or their names in the Hierarchy pane.



Add texture to each of the cubes

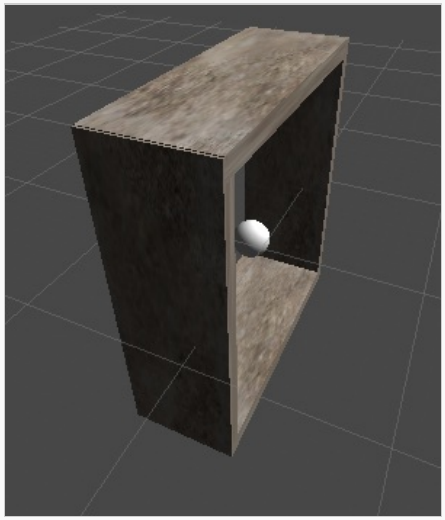
.1. Now add lighting. At the top click **GameObject –> Create Other –> Directional Light**. The default angle of the light works fine for me, but you can rotate it to get what you want. You can think of a Directional Light as the sun, so position doesn't matter, only the angle at which it shines.

(C) Copyright Nokia 2013. All rights reserved.



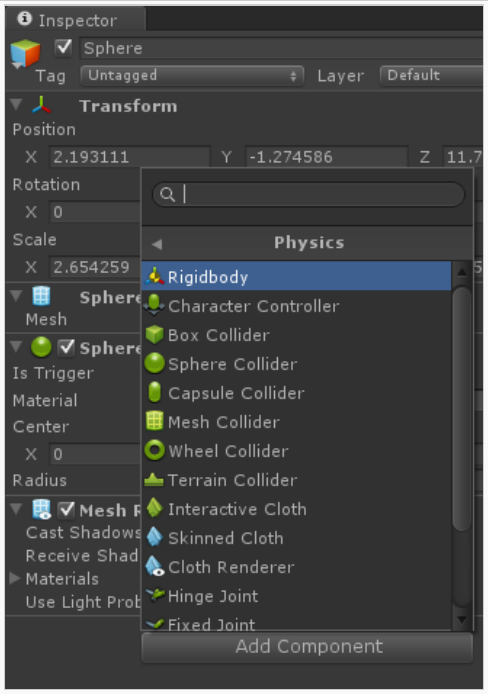
Controlling directional light

.2. Now create the ball using **GameObject –> Create Other –> Sphere**. Make it big enough to see easily, and position it roughly in the middle of the walls – most importantly make sure that it is in the right place depth-wise (as in distance to the camera).



Create and position a sphere

.3. We want this ball to bounce off the walls. This would usually be a rather complicated affair, except of course, with Unity3D. Select the sphere, then in the Inspector pane on the right click **Add Component –> Physics –> Rigid Body**.



Set rigid body physics on the ball

.4. Clicking Play (at the top) at this point will have the ball just dropping down and resting on the bottom wall. Firstly, we don't want gravity, so deselect the Gravity checkbox in the *Inspector*. Next, we want the sphere to be bouncy, so we need to create a new Physic Material for it. Right click in the Project pane at the bottom: *Create –> Physic Material*' and name it whatever you want. Now drag it from the bottom onto your sphere.

(C) Copyright Nokia 2013. All rights reserved.

.5. To create some initial movement for the sphere lets give it some code. Right click in Project pane: **Create –> C# Script**. Double-click on the new script file which will open up the code in your default editor. Stick this line of code into the `Start()` method then save:

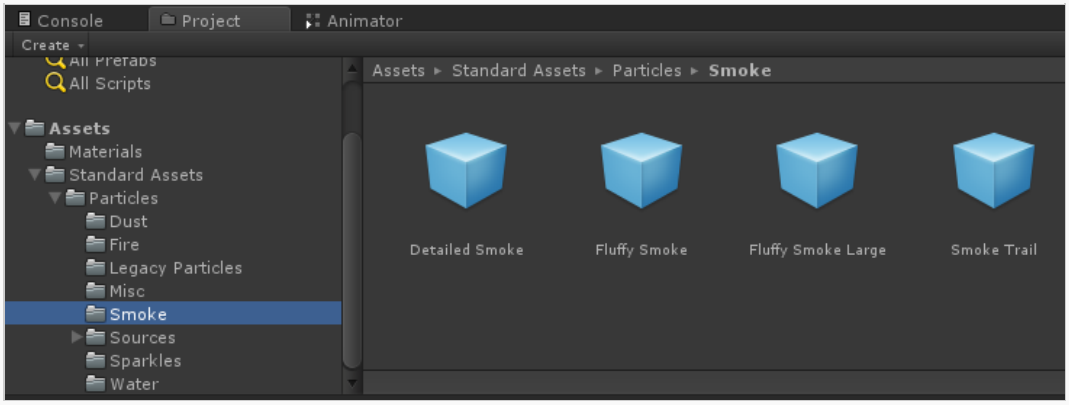
```
transform.rigidbody.AddRelativeForce(-2000f, -2000f, 0);
```

.6. Go back to Unity3D, and drag the script onto the sphere. Clicking **Play** will have the ball bouncing a bit but slowing down. So select the Physic Material you created earlier in the Project pane, and then in the Inspector pane set these values:

```
Dynamic Friction to 0
Static Friction to 0
Bounciness to 1
Bounce Combine to Maximum
```

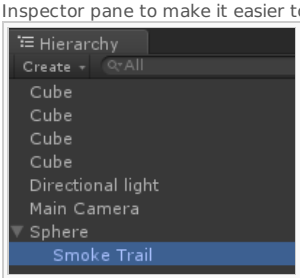
This will make it quite a bit bouncier, but it will still slow down. So drag the Physic Material over each wall, making them "bouncy" too. Now hit Play and the sphere should continually bounce around contently.

.7. Now add a smoke trail to the sphere.
- In the top menu hit **Assets –> Import New Package –> Particles** then click **Import** on the dialog that comes up. This will import the whole package – which you don't need – but it's fine for now. If you open up a different program when Unity3D has a dialog open it disappears – you will need to **Alt+Tab** back to get the dialog back.
- In the Project pane at the bottom navigate to the Smoke folder.



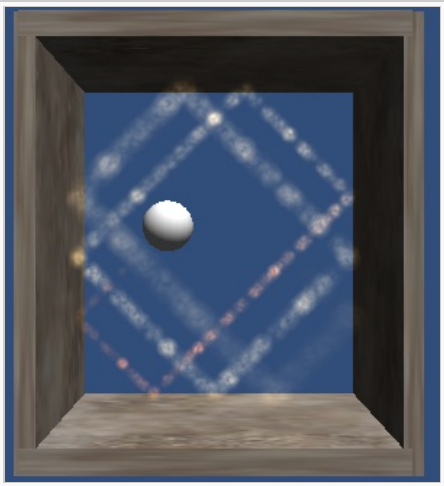
Smoke folder

- Drag the Smoke Trail onto the sphere which will make it a child. Then with it selected, change the value of Min Emission and Max Emission to 200 in the Inspector pane to make it easier to see.



Smoke trail as child of the ball

- Hit Play and you should have a smoke trail.



Sphere with smoke trail

Interacting with Windows Phone

A rather cool side-effect of how Unity3D creates a Windows Phone project is that your game ends up as a class library rendering to a XAML panel. This means that you have direct access in code to all your Unity3D script classes. So let's assume that there was no way in Unity3D to make Windows Phone vibrate – we can create an event in a script, and then sign up to that event in the Windows Phone project code and call the vibrate API.

You're a Unity3D expert by now, so I will be moving faster.

1. We need a static class that the event will be in. Create a new C# script in the Assets folder and open it up. I called mine StaticInterop. Clear the code

(C) Copyright Nokia 2013. All rights reserved.

that was generated so you just have this:

```
using UnityEngine;
using System.Collections;
using System;
public class StaticInterop { }
```

2. Now add this code and save:

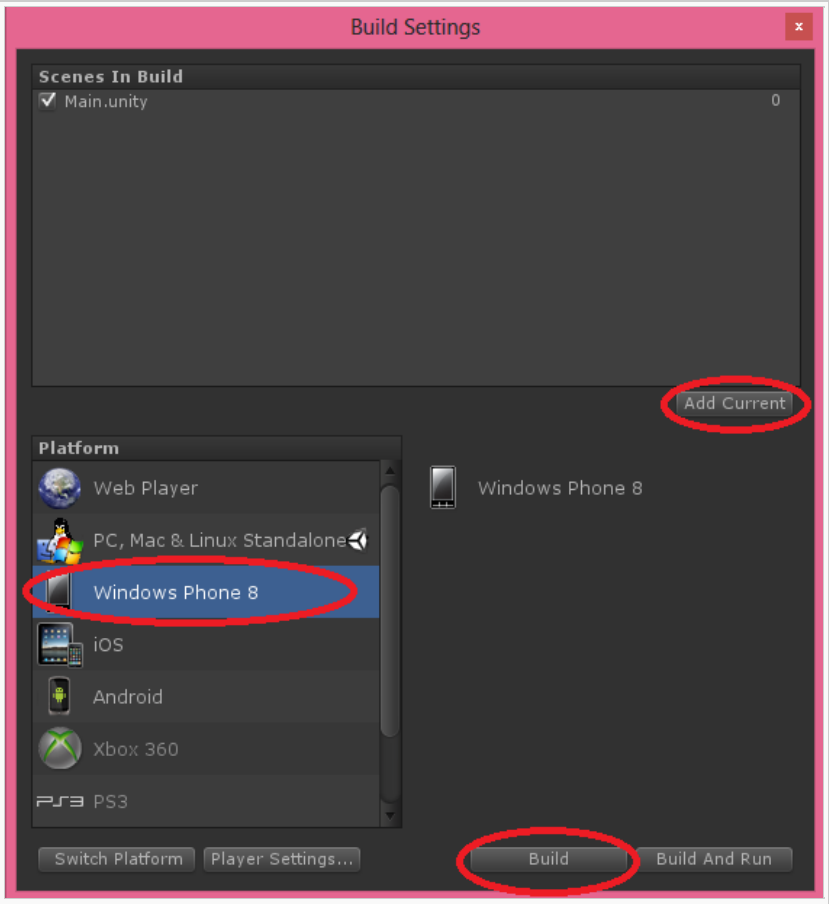
```
public class StaticInterop {
    public static event EventHandler VibrateMe;

    public static void FireVibrateMe()
    {
        if (VibrateMe != null)
        {
            VibrateMe(null, null);
        }
    }
}
```

3. So that's just a little helper to fire an event. Next we need to fire the event every time the sphere hits a wall. So open up the script you applied to the sphere earlier and add this method which will automatically get called when the body collides with something:

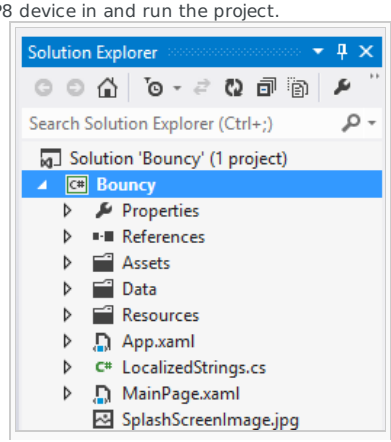
```
void OnCollisionEnter(Collision collision) {
         StaticInterop.FireVibrateMe();
    }
```

4. That's all the Unity3D stuff done, so lets export this out. Make sure your scene and all files are saved. Click **File –> Build Settings**, select Windows Phone 8, then **Add Current**. Then to export it press **Build** at the bottom. You can create a new folder for it to export to.



xxx

5. When it's finished the root folder will popup. Go into the folder that you chose to export to and open up the solution. At this stage you can plug your WP8 device in and run the project.



(C) Copyright Nokia 2013. All rights reserved.

Unity project in project explorer

6. Let's now add some vibration! In **MainPage.xaml.cs** make a class variable for the Vibration API (this will need using Microsoft.Devices):

```
VibrateController vibrator = VibrateController.Default;
```

7. Sign up to that event (we created earlier) on the last line of the `DrawingSurfaceLoaded` method:

```
StaticInterop.VibrateMe += StaticInteropVibrateMe;
```

8. And add the event handler code:

```
void StaticInterop_VibrateMe(object sender, EventArgs e)
        {
            vibrator.Start(new TimeSpan(0, 0, 0, 0, 20));
        }
```

Running the project on your Windows Phone will now have a little ball bouncing around with a smoke trail, and the phone will vibrate for 20ms every time a wall is hit.

Hopefully this has helped a few Windows Phone devs getting started in the Unity3D world.

The Llama Apocalypse

Just to show that the example is not "purely theoretical", below is the teaser for a Windows Phone 8 game

The media player is loading...

Acknowledgements

Thanks a lot to the guys at MonkeyMoon and CodingJar who helped me with all my little problems at AppCademy. Checkout their games using Unity3D on WP8!

If you've got question you can ask me here or on Twitter: @roguecode

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.