extracted from: http://www.catsinthesky.com/blog/article/2012/04/9/farseer-physics-box2d-and-unity-part-4

Farseer Physics (Box2D) and Unity (Part #4)

posted by gabs | Apr 30th, 15:45

Part #1 | Part #2 | Part #3 | Part #4 | .unitypackage | GitHub

This part of the tutorial covers **collision events**, a.k.a. **triggers**. Unlike collision filtering, triggers are always present in games. They are essential for a game's logic, from save points to deadly bullets.

Since triggers can be used for anything and in many different ways, I'll cover more the programming logic behind it (instead relying on already made components like in the previous parts). Grab the Unity package file, start a new project then import everything. The base scene is in Tutorials/Part 4/ CollisionEventsBase (Image 1).

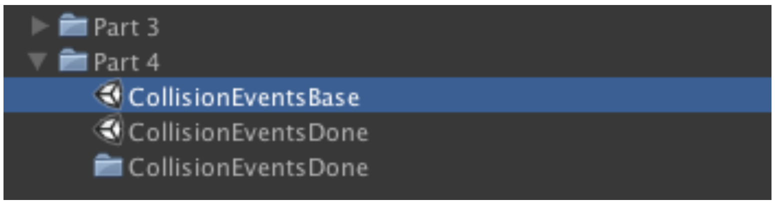


Image 1 - The tutorial starter scene

Now FarseerUnity also stores the **Tag** information inside Body and Fixture instances. You can use this to work with your trigger logic. For this tutorial, we will change the PLAYER tag to "Player" (Image 2) and the playerBOX tag to "Respawn" (Image 3).

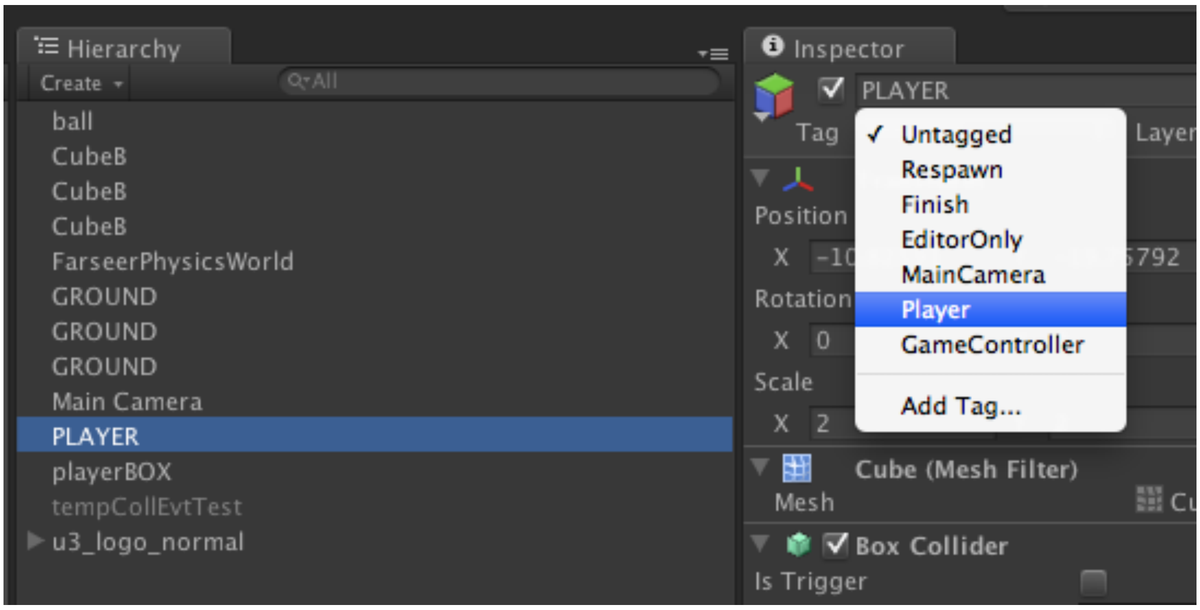


Image 2 - PLAYER tag

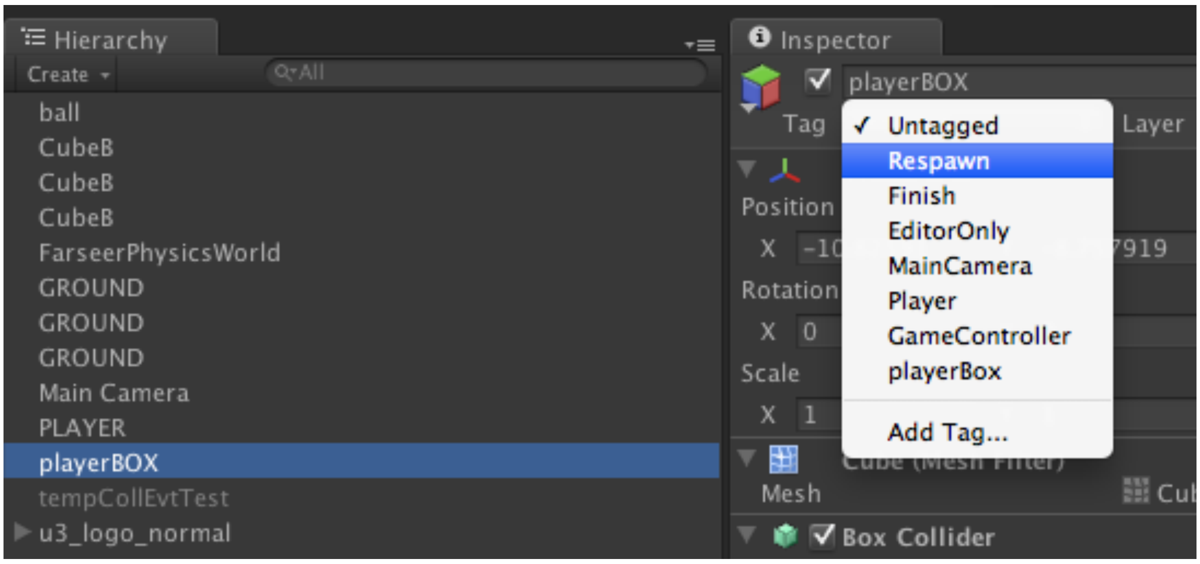


Image 3 - playerBOX tag

Now, create a new C# script at Tutorials/Part 4/CollisionEventsBase folder named "TestCollisionEventsCp". For now we will leave it with this code:

```
using UnityEngine;
using System.Collections.Generic;
using FarseerPhysics.Dynamics.Contacts;
using FarseerPhysics.Dynamics;
using FVector2 = Microsoft.Xna.Framework.FVector2;
public class TestCollisionEventsCp : MonoBehaviour
{
}
```

Add this script to the PLAYER game object (Image 4).

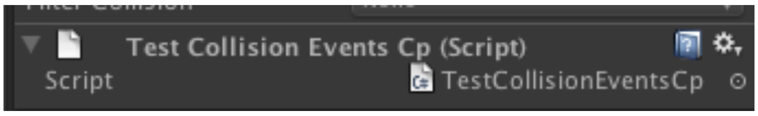


Image 4 - The new component added to the PLAYER game object

Now replace the code with this basic structure:

```
using UnityEngine;
using System.Collections.Generic;
using FarseerPhysics.Dynamics.Contacts;
using FarseerPhysics.Dynamics;
using FVector2 = Microsoft.Xna.Framework.FVector2;
public class TestCollisionEventsCp : MonoBehaviour
{
      private Body body;
```

```
	private List<Contact> lastContacts;
	void Start()
	{
		body = GetComponent<FSBodyComponent>().PhysicsBody;
		lastContacts = new List<Contact>();
	}
	void Update()
	{
	}

	private bool OnCollisionEvent(Fixture fixtureA, Fixture fixtureB, 
Contact contact)
	{
		return true;
	}
}
```

The Farseer Physics Engine uses default C# events to dispatch when a collision takes place (body.OnCollision) and when it stops (body.OnSeparation). This is similar to Physx OnCollisionEnter and OnCollisionExit, but it is per fixture (a body can have multiple fixtures/ shapes). To handle the in-between collision steps, you can store the contact instance, like it's prepared for the lastContacts variable.

After this line:

```
		lastContacts = new List<Contact>();
```

Add this line:

```
		body.OnCollision += OnCollisionEvent;
```

Now when a collision takes place, the OnCollisionEvent function will be triggered. It returns a boolean because the function tells Farseer if we want this collision to occur, or not. If you return false, the objects won't collide at all. Change the "return true;" into "return false;" and hit play on the editor to see what happens.

Now, to test a scenario when you don't want a collision between 2 specific objects (in this case, PLAYER and playerBOX), move the CubeB stack away from the player game object, and replace the OnCollisionEvent function with this:

```
	private bool OnCollisionEvent(Fixture fixtureA, Fixture fixtureB, 
Contact contact)
	{
		Body bodyB = fixtureB.Body;
		if(bodyB.UserTag == "Respawn")
			return false;
		return true;
	}
```

Since we set the tags earlier, we can use them to filter a collision just like a simple collision filtering setup would do. If you hit play now, PLAYER and playerBOX won't collide because this event is manually filtering them out.

Now suppose you would want to react to this collision immediately. The Contact class holds everything you need to do that. The global Manifold holds the global collision points and the collision normal. Replace OnCollisionEvent with this:

```
	private bool OnCollisionEvent(Fixture fixtureA, Fixture fixtureB, Contact contact)
	{
		Body bodyB = fixtureB.Body;
		if(bodyB.UserTag == "Respawn")
		{
			FVector2 normal;
			FarseerPhysics.Common.FixedArray2<FVector2> contactPoints;
			contact.GetWorldManifold(out normal, out contactPoints);
			bodyB.ApplyLinearImpulse(normal * -55f);
			body.ApplyLinearImpulse(normal * 55f);
		}
		return true;
	}
```

In the code above, the normal variable holds the global collision normal, so when we apply a force using it, the objects will fly at opposite directions. The global contact points could also be used for many things, such as playing 3D impact sounds at a point.

Suppose that playerBOX is a collectible item. You could do something like this:

```
	private bool OnCollisionEvent(Fixture fixtureA, Fixture fixtureB, Contact contact)
	{
		Body bodyB = fixtureB.Body;
		if(bodyB.UserTag == "Respawn")
		{
			// remove BodyB controller
			Destroy(bodyB.UserFSBodyComponent.gameObject);
			// add points, stats, anything else
			// here
			// don't return a collision
			return false;
		}
		return true;
	}
```

And for checking if the player can jump:

CODENEW

Now, let's do a more complex example to use what the Contact class can offer, like getting the total weight of an object (like scales). Since the contact also stores the resulting normal and tangent impulses, it's easy to do it. First, create a new 3D Text game object to display the weight (Image 5).

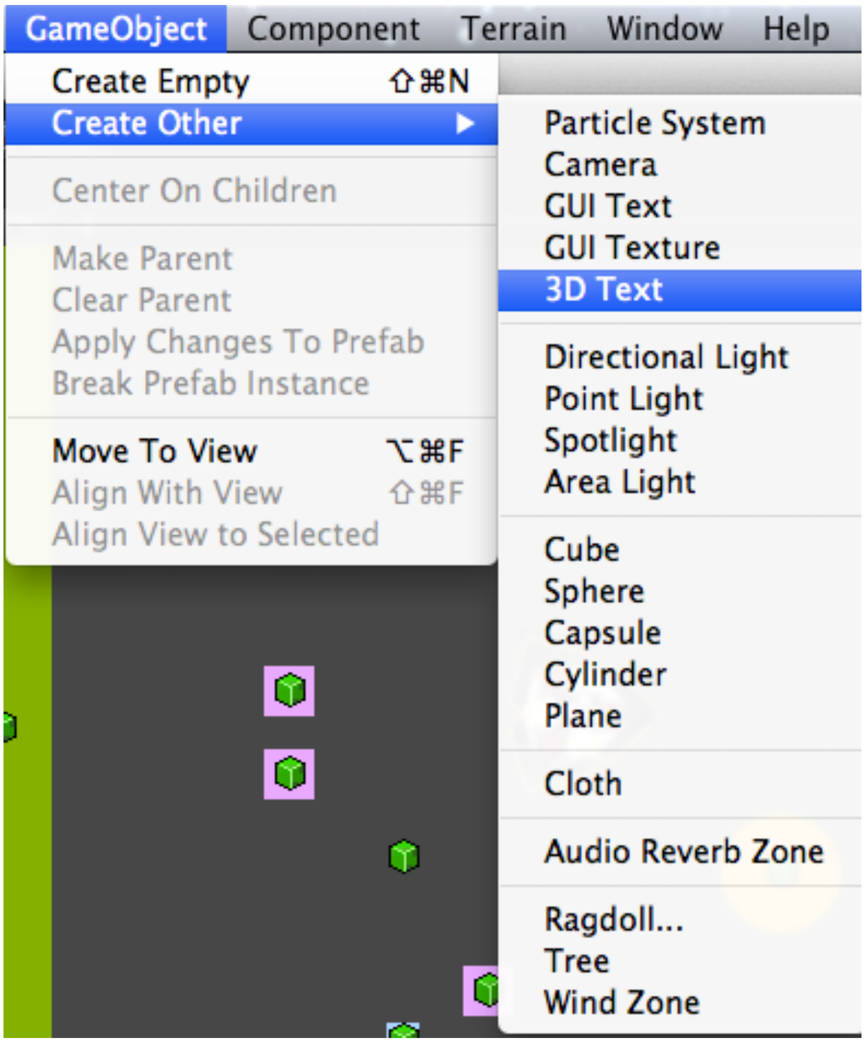


Image 5 - New 3D Text

Now make this game object a child of PLAYER (Image 6).

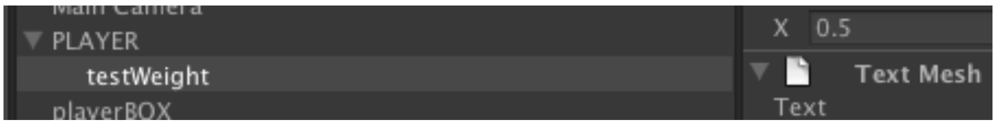
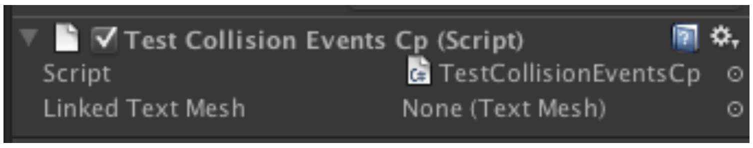


Image 6 - 3D Text GO as a child of PLAYER GO

And add a public member inside TestCollisionEventsCp class (paste the code below):

```
public TextMesh LinkedTextMesh;
```

Now drag the 3D Text game object instance to the LinkedTextMesh property (Image 7):



Test Collision Events Cp (Script)
Script: TestCollisionEventsCp
Linked Text Mesh: None (Text Mesh)

Image 7 - Linking the 3D text instance.

Now replace the TestCollisionEventsCp code with this:

```
using UnityEngine;
using System.Collections.Generic;
using FarseerPhysics.Dynamics.Contacts;
using FarseerPhysics.Dynamics;
using FVector2 = Microsoft.Xna.Framework.FVector2;

public class TestCollisionEventsCp : MonoBehaviour
{
	public TextMesh LinkedTextMesh;

	private Body body;

	private List<Contact> lastContacts;

	void Start()
	{
		body = GetComponent<FSBodyComponent>().PhysicsBody;
		lastContacts = new List<Contact>();
		body.OnCollision += OnCollisionEvent;
	}

	void Update()
	{
		float weight = body.Mass;
		GetWeight(ref weight);
		LinkedTextMesh.text = weight.ToString("#0.00") + "Kg";
	}

	private void GetWeight(ref float weight)
	{
		if(lastContacts.Count < 1)
			return;
		float ownWeight = weight;
		weight = 0f;
		foreach(Contact lastContact in lastContacts)
		{
			bool isTouching = lastContact.IsTouching();
			if(isTouching)
			{
FarseerPhysics.Common.FixedArray2<FarseerPhysics.Collision.ManifoldPoint>
localManifoldPoints =
lastContact.Manifold.Points;
				// gravity = 9.8f (hard coded here just for testing
purposes)
```

```
                    // Time.fixedDeltaTime is the FPE timeStep
                    weight +=
(1f * (localManifoldPoints[0].NormalImpulse/Time.fixedDeltaTime) / 9.8f);
                    weight +=
(1f * (localManifoldPoints[1].NormalImpulse/Time.fixedDeltaTime) / 9.8f);
               }
          }
          // remove inactive contacts
          for(int i = 0; i < lastContacts.Count; i++)
          {
               if(!lastContacts[i].IsTouching())
               {
                    lastContacts.RemoveAt(i);
                    i = Mathf.Max(0, i - 1);
               }
          }
          // calc weight
          weight -= ownWeight;
          weight *= 0.5f;
          weight += ownWeight;
     }

     private bool OnCollisionEvent(Fixture fixtureA, Fixture fixtureB,
Contact contact)
     {
          if(!lastContacts.Contains(contact))
               lastContacts.Add(contact);
          return true;
     }

}
```

As you can see, the local Manifold points are packed with impulse data. This can be used to determine the total weight, crush force, break force, etc. To view the global manifold points and normals, add this:

```
     void OnDrawGizmos()
     {
          if(lastContacts == null)
               return;
          foreach(Contact lastContact in lastContacts)
          {
               if(!lastContact.IsTouching())
                    return;

               FarseerPhysics.Common.FixedArray2<FVector2> contactPoints;
               FVector2 normal;
               lastContact.GetWorldManifold(out normal, out contactPoints);

               Vector3 p0 = FSHelper.FVector2ToVector3(contactPoints[0]);
               Vector3 p1 = FSHelper.FVector2ToVector3(contactPoints[1]);
               Vector3 pn = FSHelper.FVector2ToVector3(normal);
               Gizmos.color = Color.red;
```

```
                Gizmos.DrawWireSphere(p0, 0.15f);
                Gizmos.DrawLine(p0, p0 + pn * 2f);
                Gizmos.color = Color.yellow;
                Gizmos.DrawWireSphere(p1, 0.15f);
                Gizmos.DrawLine(p1, p1 + pn * 2f);
            }
        }
```

If everything is working properly, the weight will change when you stack objects on top (Image 8):

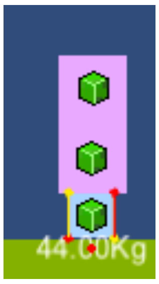


Image 8 - 44.0Kg

That's it for part 4! I decided to focus on programming because it doesn't matter how many collision events components I may add in the future, there will always be a case when your game needs something specific.