Intro to Unity

Hands-On Challenges

Introduction to Unity Hands-On Challenges

Challenge H: Diving Deeper

Hopefully you are feeling better with scripting. It's at this point that the difficult curve gets a little steeper. This is the point where you will need to have a somewhat firm understanding of C# (or UnityScript or Boo).

If you have no C# experience at all, then check out this video series:

http://channel9.msdn.com/Series/C-Sharp-Fundamentals-Development-for-Absolute-Beginners

There's also a great tutorial series over a csharp-station.com:

http://csharp-station.com/Tutorial/CSharp/Lesson01

In this challenge, you will be doing three things: responding to GameObject events, creating your own custom events, and finally, creating some coroutines.

Getting Started

Open the starter challenge project. If you've been following this series from the beginning, open your last saved project.

> **Note:** Unity will open to empty scene. Find the scene that you saved, or if you are using the Starter Project, open Main.scene.

GameObject Events

Currently, the breakout game is somewhat flat. The ball just hits objects and bounces off of them. You want those objects to react to being hit by disappearing.

There are many ways to do this, but the easiest is to implement collision events.

Look up this page:

http://docs.unity3d.com/Manual/EventFunctions.html

and find the event that you need to use.

Once you find that event, create a new script, call it **BrickScript** and attach it to a brick. Make sure this attachment is replicated to all the bricks in your scene.

When a collision occurs, you will want to make it disappear. To do so, you simply deactive the renderer. Once the renderer is disabled, it won't be rendered in your game.

To do so, you will simply add this bit of code inside the event method:

```
renderer.enabled = false;
```

Just note, even though the renderer is disabled, the collider is still active. Look up and implement the procedure to disable the collider as well.

Once you have it place, run your game. If you have everything in place, you should now see the bricks disappear on collision events, giving the ball more space to pass through.

You'll be adding some fancy animations later in this series.

Another way to implement this would be to destroy the objects when they were hit by the ball, then re-instantiate the bricks at the start of each new level or game.

By keeping the objects in the scene, you avoid unnecessary garbage collection or the CPU expense of creating and deleting objects. For desktop versions of the game, this would not be a problem but for mobile versions, this saves both power and battery life.

Keeping Score

While breaking blocks is certainly fun, players ultimately will want a way to keep track of their score. In a later video, you will present the actual score to the player, but first you'll need to create a way to keep track of it.

With your project open, create an Empty GameObject by selecting **GameObject \ Create Empty.**

Call this new object: **GameManager**.

Create a new **C# sharp** script, call it **GameManager**, and attach it to the **GameManager**.

Create your first instance variables inside of it.

```
public int brickPoint = 10;
private int score = 0;
```

Now when you select the GameManager, you can change the point value inside of the inspector as well as keep trackof the player's score.

Now comes the real challenge. You will create a new event that will occur everytime a ball touches a brick. Once this happen, you will want the GameManager to be notified and increase the score.

Inside of BallScript, you should declare an event and a delegate method like so:

```
public delegate void BallEventHandler(GameObject ball, GameObject
collision);
public static event BallEventHandler onBallCollideWithBrick;
```

You must call the collision event inside of the BallScript. So first, implement a collision event, then call the actual event handler when a collision occurs with a brick.

Next, you need your GameManager to subscribe to these events, so in your GameManager object, add the following code:

```
public void OnEnable() {
  BallScript.onBallCollideWithBrick += this.CollideWithBrick;
}

public void OnDisable() {
  BallScript.onBallCollideWithBrick -= this.CollideWithBrick;
}
```

By subscribing to these events during `OnEnable()` and `OnDisable()`, you can safely disable the GameManager when necessary, and not worry about any adverse side effects.

`CollideWithBlock()` is what you will need to add to actually do the score tracking. Once you are keeping score, use a `Debug.Log()` statement to output the score to the console.

You can learn more about `Debug.Log()` over here:

http://docs.unity3d.com/ScriptReference/Debug.Log.html

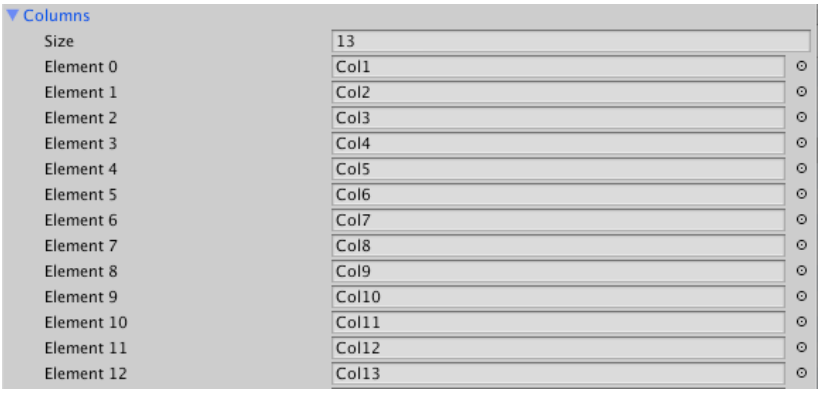
Fancy Routines

The final aspect of your challenge, is to have each column appear on their own. Later you will be adding animating each of the bricks, but for now, the columns will appear one after another.

The best way to do this is to use a co-routine!

First, you need to create an array to hold all the columns. Call this instance variable: `Columns`.

Make it public and place it in the GameManager's instance variables. Now, drag each colum into the array so that the inspector looks like this:



Now, create a new method inside of your GameManager. Call it `enableColumns()`. It should be a private method and it must return the type `IEnumerator` in order to function as co-routine.

In this method, loop through each of these columns and enabling each one of them. Before the closing brace of the for-loop, add the following bit of code:

```
yield return new WaitForSeconds(.1f);
```

This is the heart of the co-routine.

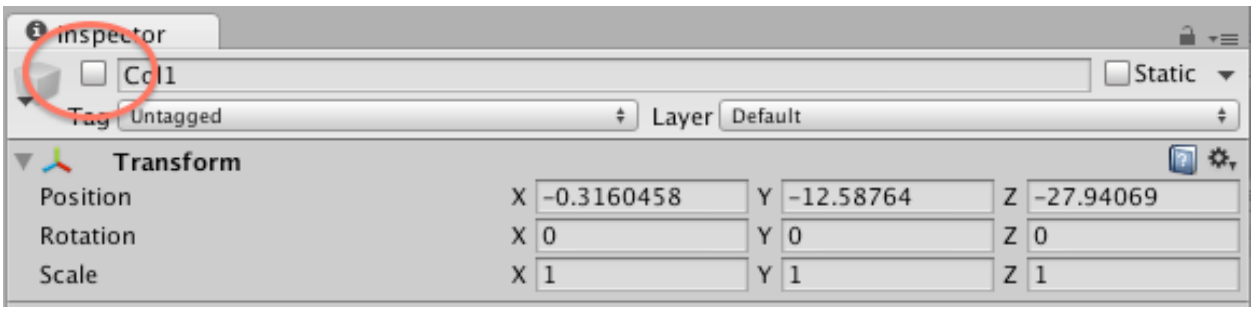
Now, to make sure the co-routine only fires once, before the end of the method, add the following bit of code:

```
StopCoroutine("enableColumns");
```

You'll need to find the place to start the co-routine. Once you determine an event that you will use, add the following code:

```
StartCoroutine("enableColumns");
```

Finally, in Unity's inspector, disable each column by clicking the checkbox in the inspector like so:



Now your columns should appear one by one.