Intro to Unity

Hands-On Challenges

Introduction to Unity Hands-On Challenges

Challenge F: Adding Physics to Your Game

Now that you have your game all set up, it's time to start adding some interaction to it. You will be doing this through physics. In this challenge, you will add the physics components, then write your first script to actually set things in motion.

Mind you, you will be learning about scripting in depth in later videos of this series. For now, just buckle up and hold on. The ride is going to be good! :]

Getting Started

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

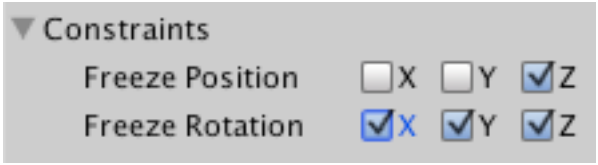
> **Note:** Unity will open to empty scene. Find the scene that you saved, or if you are using the Starter Project, open Main.scene.

The first thing you need to do is determine where to place the rigidbodies. Remember, at least one GameObject needs to be a rigidbody in order to react to collisions.

Can you guess which GameObjects should have rigidbody components attached to them? Add them to those GameObjects through the component menu.

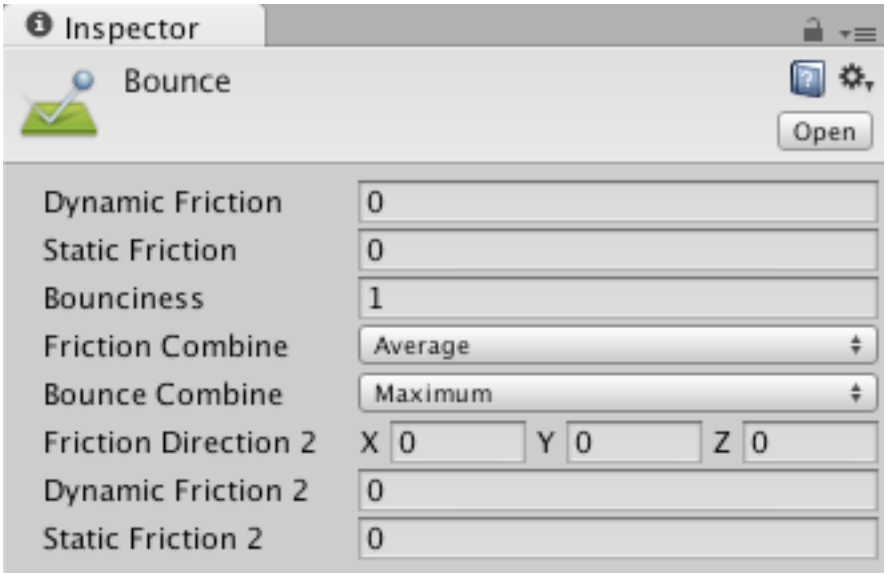
Once you add the rigidbodies, **uncheck** the **Use Gravity** checkbox since you don't want your GameObjects to fall to the ground.

Next, you need to add some contraints to one specific GameObject as you only want it to move on a 2D plane. Set the rigidbody constraints to look like the following:



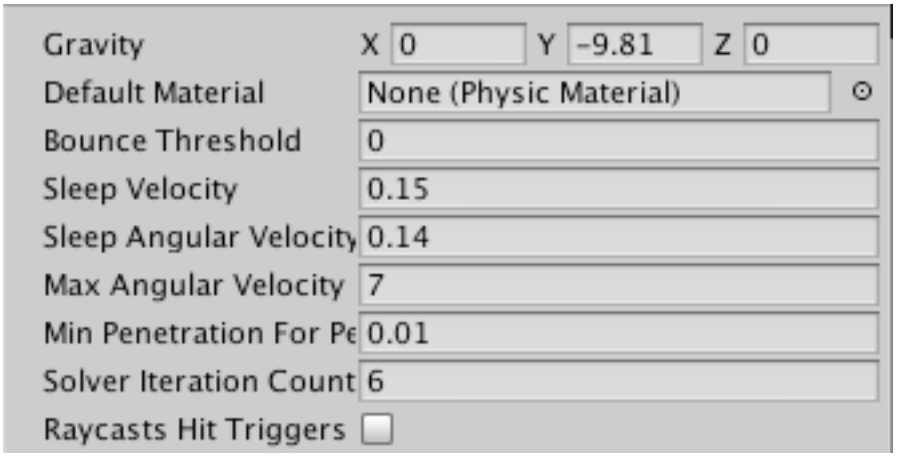
Next, you'll want to create some Physic Materials. By default, every GameObject will produce some friction. You want things to be bouncy!

Create a new folder in your Project view, and call it **Physics Material**. Inside of the folder, create a **new Physics Material** and call it **Bounce**. Set it to have the following attributes:



Now, add the **Bounce Material** to any object that you think will interact with the ball.

Next, update the project settings. To access them, select **Edit \ Project Settings \ Physics** and make the dialog look like the following:



Now to see, everything in action, create a new folder in the Project view. Call it Scripts. In the Scripts folder, create a new C# script and call it, **BallScript**. Replace the class code with the following:

```
public class BallScript : MonoBehaviour {

    public float minimumSpeed = 500; // 1
```

```
    public void LaunchBall()
    {
        if (transform.parent) {
            transform.parent = null; // 2
        }
        rigidbody.AddForce(minimumSpeed, minimumSpeed, 0); // 3
    }

    void Update () {
        if (Input.GetKeyDown(KeyCode.Space)) {
            if (transform.parent != null) {  // 4
                LaunchBall();
            }
        }
    }
}
```

Scripting will be covered in depth in later tutorials, but here's a quick breakdown of the previous code:

1. A public variable is initialized to hold the minimum speed of the ball. Since this is a public variable, you can now change the value of it in the inspector. This change can occur in realtime, as you are testing your game, and once you stop playing the game, the value will reset itself to its original value.

2. Every GameObject has an associated transform, and you can access the transform directly in code. The code checks to see if the Ball has a parent GameObject, and if so, removes it.

3. Every GameObject can also have a rigidbody so you can access the rigidbody in shorthand. If a rigidbody is not attached to a GameObject, then that property will be null. The code access the rigidbody, then applies force to it.

4. `Update()` is a standard API method that occurs once per frame. When the user presses down on the spacebar, the code checks to see if there is a parent GameObject. If this is the case, then the ball has not been launched.

Start the game and press the spacebar. You will see the ball bounce around the game area. If see your ball start to slow down after hitting a brick, rail, or paddle, check to see if you have attached the bounce Physic Material.

Congrats … your game is starting to take shape.