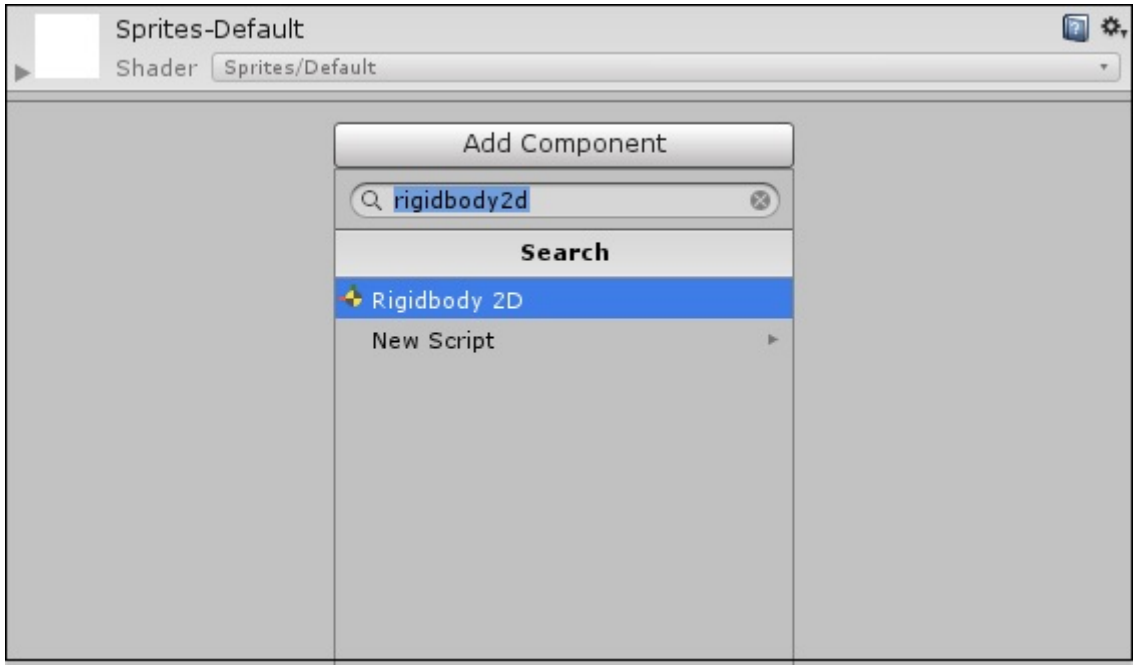
Unity - Rigidbodies and Physics

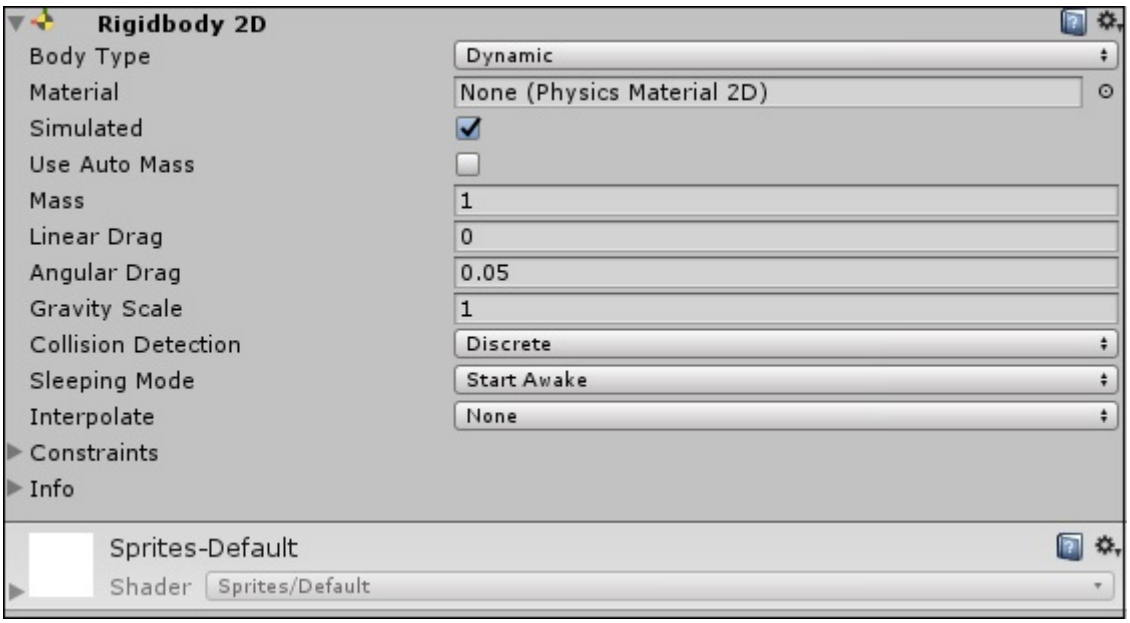
The main issue with the collisions in the last chapter was with the code. **We will now modify the values of the GameObject's position directly**. We are simply adding a value to the position, if the player is pressing a key. We need a way to make the player move in such a way that it reacts properly to boundaries and other GameObjects.

To do so, we need to understand what **rigidbodies** are. Rigidbodies are components that allow a GameObject to react to **real-time physics**. This includes reactions to forces and gravity, mass, drag and momentum.

You can attach a Rigidbody to your GameObject by simply clicking on **Add Component** and typing in Rigidbody2D in the search field.



Clicking on Rigidbody2D will attach the component to your GameObject. Now that it is attached, you will notice that many new fields have opened up.



With the default settings, the GameObject will fall vertically **down** due to gravity. To avoid this, set the **Gravity Scale** to 0.

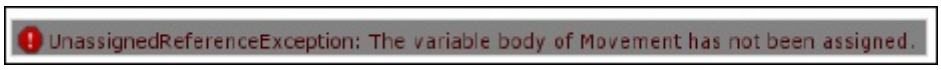
Now, playing the game will not show any visible difference, because the GameObject does not have anything to do with its physics component yet.

To solve our problem, let us open our code again, and rewrite it.

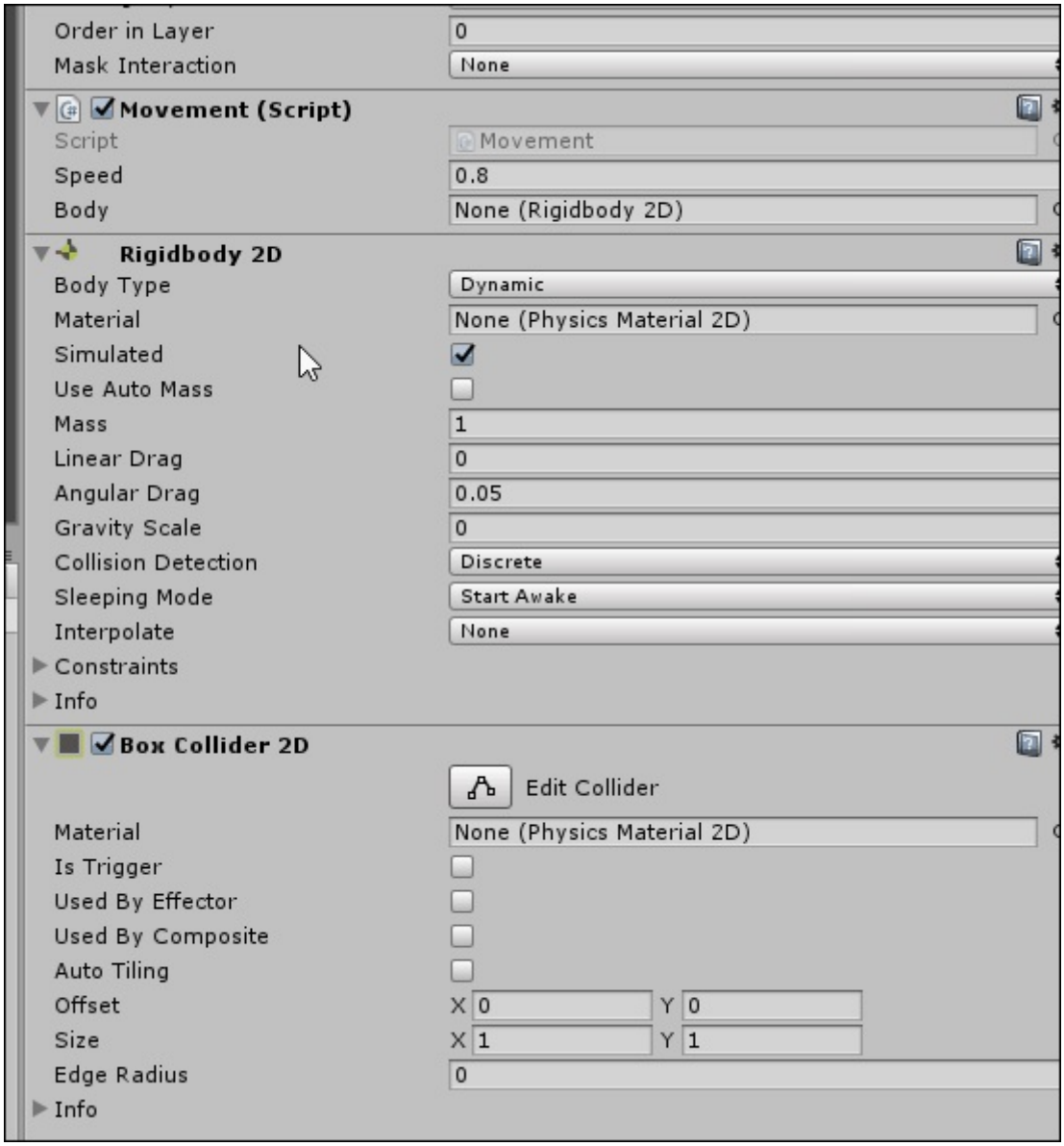
```
public class Movement : MonoBehaviour {
   public float speed;
   public Rigidbody2D body;
   // Update is called once per frame
   void Update() {
      float h = Input.GetAxisRaw("Horizontal");
      float v = Input.GetAxisRaw("Vertical");
      body.velocity = new Vector2(h * speed, v * speed);
   }
}
```

We can see that we create a **reference** to a Rigidbody2D in the declarations, and our update code works on that reference instead of the Object's transform. This means that the Rigidbody has now been given the responsibility of moving.

You may expect the **body** reference to throw NullReferenceException, since we have not assigned anything to it. If you compile and run the game as is, you will get the following error on the bottom left of the editor



To fix this, let us consider the component created by the script. Remember that public properties create their own fields in Unity, as we did with the speed variable.



Adjust the speed to a higher value, around 5, and play the game.

Your collisions will now work correctly!