1 Inputs

To read an axis use Input.GetAxis with one of the following default axes: “Horizontal” and “Vertical” are mapped to joystick, A, W, S, D and the arrow keys. “Mouse X” and “Mouse Y” are mapped to the mouse delta. “Fire1”, “Fire2” “Fire3” are mapped to Ctrl, Alt, Cmd keys and three mouse or joystick buttons. New input axes can be added in the Input Manager.

If you are using input for any kind of movement behaviour use Input.GetAxis. It gives you smoothed and configurable input that can be mapped to keyboard, joystick or mouse. Use Input.GetButton for action like events only. Don’t use it for movement, Input.GetAxis will make the script code smaller and simpler.

Note also that the Input flags are not reset until “Update()”, so its suggested you make all the Input Calls in the Update Loop.

```csharp
using UnityEngine;
using System.Collections;

public class Shooter : MonoBehaviour
{
    public Rigidbody projectile;
    public Transform shootPos;
    public float shotForce = 1000f;
    public float moveSpeed = 10f;

    void Update ()
    {
        float h = Input.GetAxis("Horizontal") * Time.deltaTime * moveSpeed;
        float v = Input.GetAxis("Vertical") * Time.deltaTime * moveSpeed;

        transform.Translate(new Vector3(h, v, 0f));

        if(Input.GetButtonDown("Fire1"))
        {
            Rigidbody shot = Instantiate(projectile, shootPos.position, shootPos.rotation) as Rigidbody;
            shot.AddForce(shootPos.forward * shotForce);
        }
    }
}
```

2 Colliders

Colliders are a component that allows the game object they’re attached to react to other colliders provided that one of the game objects has a rigidbody component attached.

Colliders come in various shapes and types, and are denoted in the scene view by a green outline.

They can have the following primitive shapes: a sphere, a capsule and a box. For more complex shapes you have two options: You can either combine several of these primitive shapes together by applying primitive colliders to different objects in our hierarchy.
The other option is to use a mesh collider, which will fit the exact shape of the mesh that you specify. The reason not to use a mesh collider is that it will fit the exact shape of the mesh that you specify. So if you only specify the mesh of your detailed model then it may be providing too detailed a collision mesh and effecting performance. This is the reason why it’s often better to make a compound setup instead.

When collisions occur in the game engine one collider strikes another and an event called OnCollisionEnter is called. The rigidbody provides mass and gravity. When I play the game, one falls down and strikes the other. The cube has a box collider attached to it. Also attached to our falling object is this script. This script checks for three collision events. OnCollisionEnter, OnCollisionStay and OnCollisionExit.

When each of these occurs it writes to the console using Debug.Log. It will register when Enter is called, when Stay is occurring and when Exit is called. So if we look at our console you can see that Enter is called, Stay has occurred for a while and then Exit is called.

Note that for an OnCollision message to be sent, one of the two objects colliding must have a rigidbody component.

The contact points generated by the physics engine are stored in the Collider contact. Every contact contains a contact point, normal and the two colliders that collided (see ContactPoint). From inside OnCollisionStay or OnCollisionEnter you can always be sure that contacts has at least one element.
In order to make a collider into a trigger we simply check the “Is Trigger” checkbox on the component settings in the inspector.

When a collider is a trigger things will no longer bump into it. Instead they will pass through it and this can be detected via code.

Similar to normal colliders, events are called when other colliders are overlapping with the trigger collider.

In this example we have a box collider with Is Trigger checked and a falling ball object.

This object enters the trigger, stays as it rolls forward and then exits. To check these events we attach a script to the trigger, which is registering `OnTriggerEnter` and logging to the console and likewise `OnTriggerStay` and `OnTriggerExit`.

So usually you will make a trigger and then pass a rigidbody through it. This example is no different. We’re passing our rigidbody object through the trigger zone and detecting when something enters that trigger. So the intention with the trigger collider, also known as a trigger zone, is that you can call code without the objects in your game physically colliding.
So for example, with our hoverpad, we can add a force to our ball to give the effect of it hovering so long as it stays within the trigger zone. For example, we could address the object that’s currently staying within the trigger because the object that’s staying is saved in a variable called “other” of type Collider. We could then address the rigidbody and add a force in the direction of Vector3.up, a shortcut for ‘up’ in world coordinates. We could then multiply by a number which we’ll save as a variable.

And we’ll finish out the AddForce by choosing acceleration as the mode. Now we’ll move our object over the cube and we’ll use Freeze Position to keep it in the same place in X and Z. Allowing the force of Y to create the hovering effect.

And now if I press play, as you can see as long as it stays within that area it creates the effect of hovering without having to interact without any part of the geometry.

4 RigidBodies

As a general rule, if you’re going to have moving game objects in your game you should make sure that they are rigidbody objects.

Rigidbodies are components that allow a game object to be effected by physics. They allow the object to fall under gravity, and have physics properties such as mass, drag and velocity. When we add a rigidbody component to a game object we often then refer to it as a rigidbody object.

A rigidbody component is required for any physics based interaction, and the game object must also have a collider attached in order to interact with other physics objects.