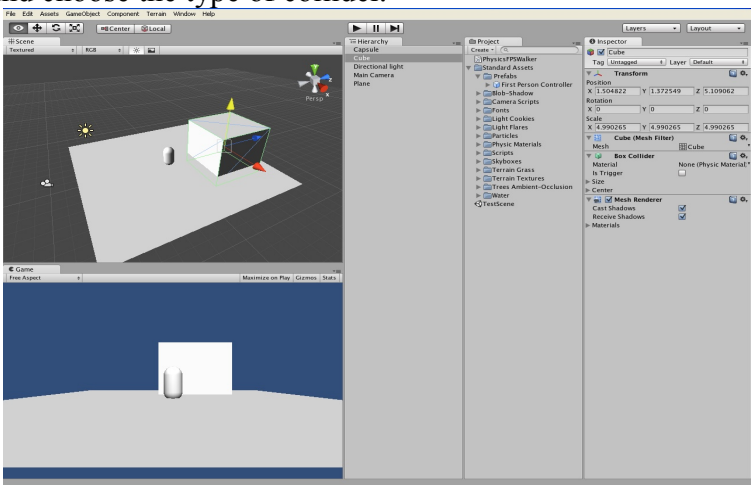
Making Trigger Zones in Unity

By Christian Barrett

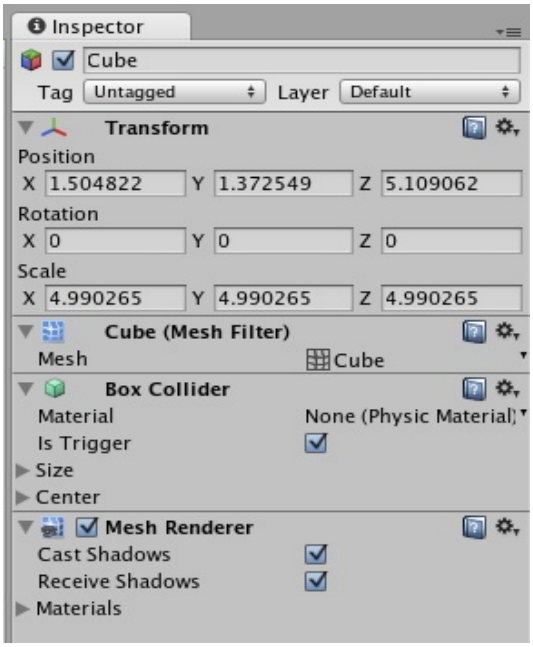
Triggers are a powerful tool in developing Unity games that every developer should know how to use. The goal of this tutorial is to teach readers how to set up an object to be a trigger and detect if it's been triggered from script.

We're going to start by making a cube for a cube-shaped trigger zone. In actuality, anything that has a collider or can get a collider can be a trigger zone, including custom meshes with custom mesh colliders. If something doesn't have a collider on it, but you still want to use it as a trigger, just go to component->physics and choose the type of collider.



So above we have our cube. If we ran the game right now, it would behave like a solid object. We don't want that - we want our cube to allow things to run through it, but still detect when something is inside its volume and trigger a script.

To do this, we need to take a look at the cube in the inspector. See the part that says box collider? Expand it, and check off the trigger box. Now if you play the game, you can walk into the cube. We now have a trigger zone.



Unfortunately, a trigger zone alone won't do anything. Which is why we need to create a script to detect when an object passes through it. This is pretty simple, as Unity provides a built in function that is called every time a collider enters it, known as **OnTriggerEnter.** What we need to do is create a script that contains this function, a variable that refers to the object we're testing for inside the trigger (for this script, it will be the player object), and some logic to determine it's the object we want that's triggering the trigger. So create a new script (assets->create->javascript), name it something logical, open it up, and write this script:

```
var target : Collider; //this is the variable that will hold the colliding object
private var triggered : boolean = false; //If we only want to detect the first time it's triggered

function OnTriggerEnter(collision : Collider)
{
	if (collision != target) //The colliding object isn't our object
	{
		return; //don't do anything if it's not our target
	}
	triggered = true; //If you want to have this only trigger once
	//Other code to do stuff on trigger goes here. You could:
	//	1. Increase a variable to keep track of score
	//	2. Transform the colliding object (send the player flying, etc)
	//	3. Play a sound (audio.Play() if you have an audio source)
	//	4. And much, much more
}
```

Now apply this script to the object that has trigger ticked off, and assign the object you want to check for collisions with to the trigger object (make sure the colliding object also has a collider of some sort on it, be it box, character, cylinder, etc). Now if you run the game, your script will detect when a character collides with it.

There's one last optional thing you can do, which is to create an invisible trigger zone. Only do this if you want to detect where the player is without the trigger zone being visible during gameplay (so this is probably not good for collectibles in the game). To remove the graphics of your trigger object, look at your object in the inspector and remove the **Mesh Renderer** component. Now your object is invisible, but still detects when something triggers it.