CODE NINJAS®

Silver Belt Ninja Guide
Activity 06: Evil Fortress of Doctor Worm

Activity 6

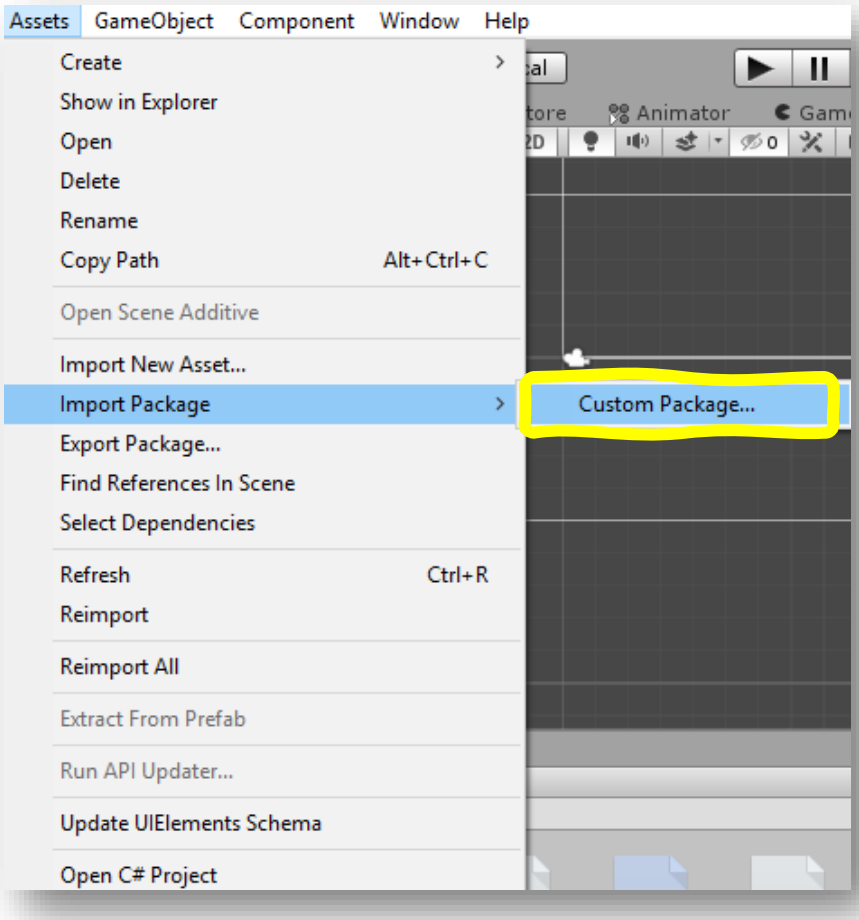
Evil Fortress of Doctor Worm

Welcome to the Evil Fortress of Doctor Worm! When you playtest the game, you can see that you can move the character. However, the doors and switches aren't working yet.

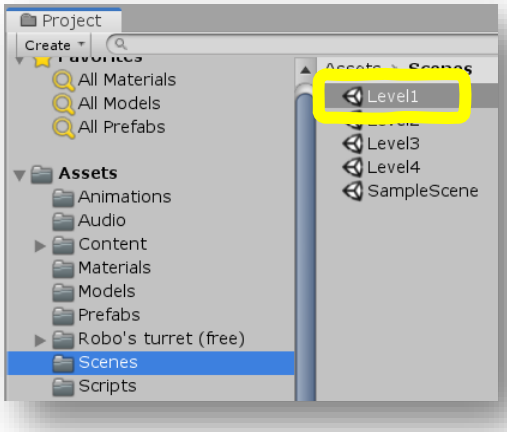
The mission: Codey is trapped in the evil fortress, and it's up to you to set him free! You'll need to create scripts to make doors in the fortress and add triggers so Codey can open them to escape.

1 Start a new Unity Project and name it *YOUR INITIALS – Doctor Worm*. Select **3D core**.

2 Import the **Activity 06 – Evil Fortress.unitypackage** file by going to **Assets > Import Package > Custom Package > All > Import**.



3 Double-click on the **Level1** scene. You can find this in the **Project** tab under **Assets > Scenes**.

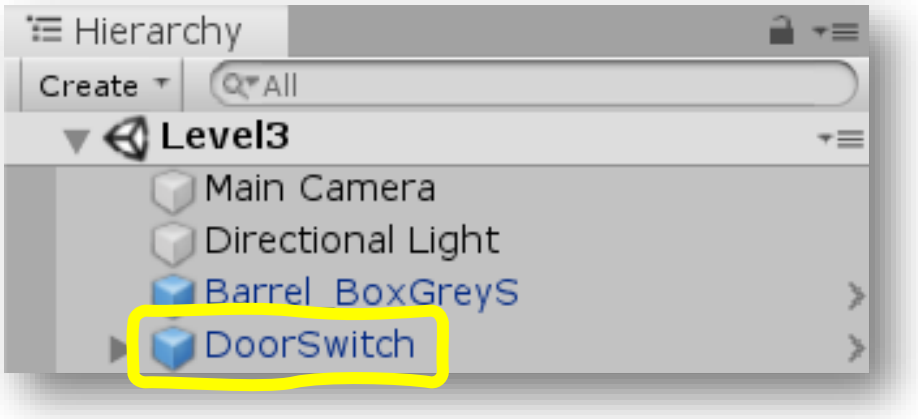


4 **Playtest** the game.

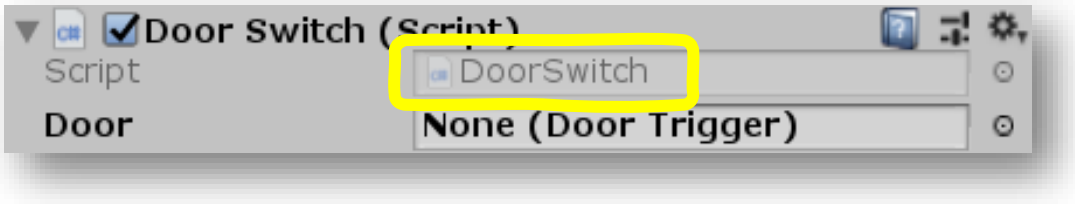
When you play, you'll notice the trigger to open the door doesn't work yet. You'll fix that in the next steps.

5 Let's connect the trigger to the door.

First, select the **DoorSwitch** game object in the Hierarchy.



6 In the Inspector, go to the **DoorSwitch (Script)** component and open the script.



7 The script has three functions and three variables.

The functions are designed to make the door switch change its color depending on whether the player has triggered the switch or not.

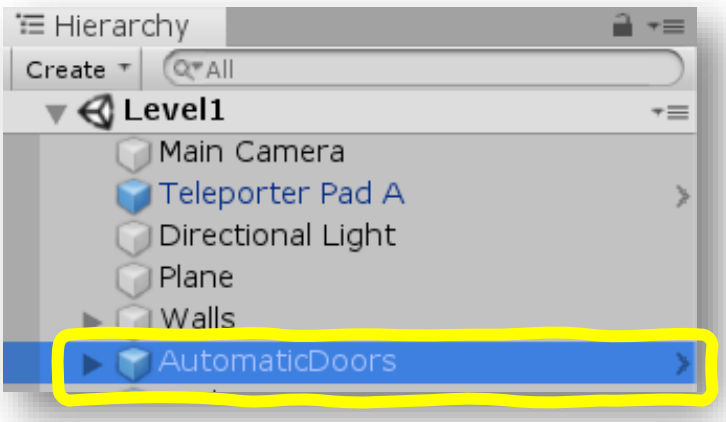
8 We are using `OnTriggerStay` for when the player is on the switch and doesn't leave. OnTriggerStay continues to run as long as something is colliding with it. The door switch will change its color from red to green.

```
//When any object enters the switch and stays, the switch icon is Green and the Red part is hidden.
Unity Message | – references
public void OnTriggerStay(Collider other)
{
    StopAllCoroutines();
    switchIcon = this.transform.Find("green").gameObject;
    switchIcon.GetComponent<SpriteRenderer>().enabled = true;
    switchIcon = this.transform.Find("red").gameObject;
    switchIcon.GetComponent<SpriteRenderer>().enabled = false;

    //Have the door check if all of the switches are green
    door.SwitchCheck();
}
```

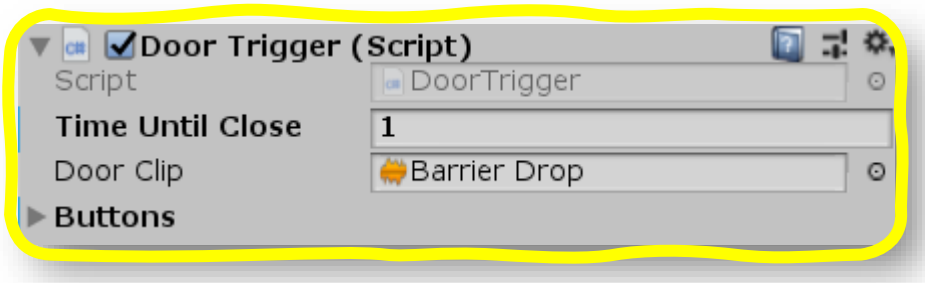
9 Return to the Unity Editor. With the **DoorSwitch** game object still selected, go to the **Door Switch** (**Script)** component in the **Inspector**.

10 On the **Door Switch (Script)** object, click the **Door** property's tiny circle to open the Selector window. Find and attach the *"AutomaticDoors"* object.

11 In the **Hierarchy**, select the **AutomaticDoors** game object.



12 In the **Inspector**, find the **DoorTrigger** script.



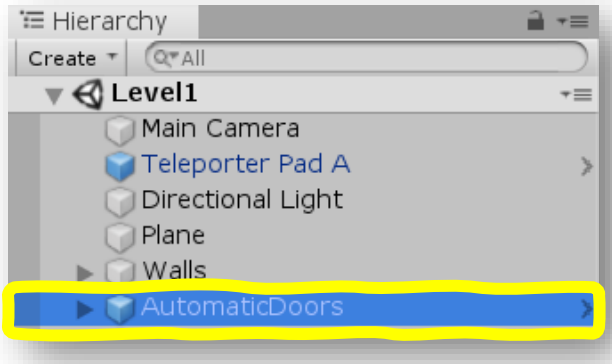
13 There is a variable called **Buttons** that is used for the door switches. Click on the triangle to the left of **Buttons** to see how many switches are in the scene.

There is only one switch in the first part of Doctor Worm.

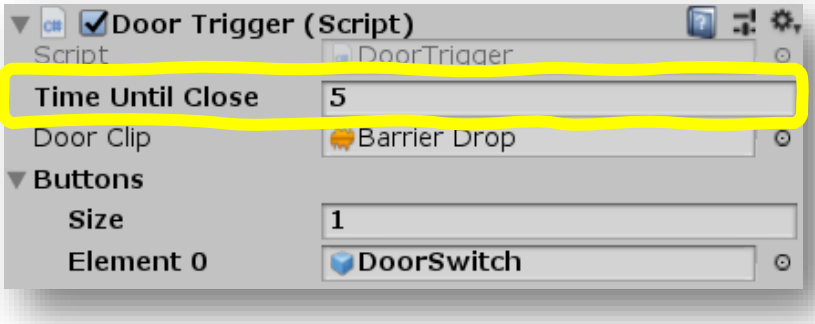
14 **Playtest** your game by clicking the **play** button. Does the switch work? What happens when the player steps off the switch?

15 Right now, the door closes 1 second after the player leaves the switch. That's not enough time to let the player leave the fortress!

16 Select the **AutomaticDoors** in the **Hierarchy**.



17 In the **Inspector**, find the **Door Trigger (Script)** component and change the value of **Time Until Close** from 1 to 5. This will leave the door open for 5 seconds giving Codey more time to escape.



18 **Playtest** the game. Now the player can make it to the exit!

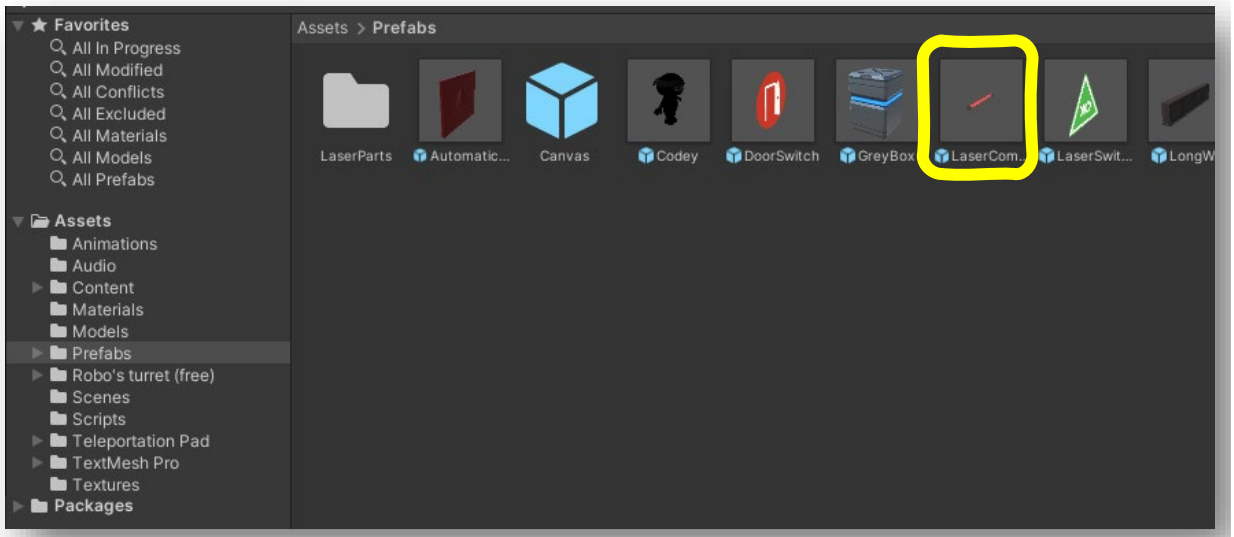
Congratulations, you have finished part 1 of the Evil Fortress of Doctor Worm! Continue your lesson with part 2 on the next page.

19 Open the Unity Project that you used for part 1 of the Evil Fortress of Doctor Worm.

20 In the **Project** tab within the **Assets > Scenes** folder, open **Level 2.**

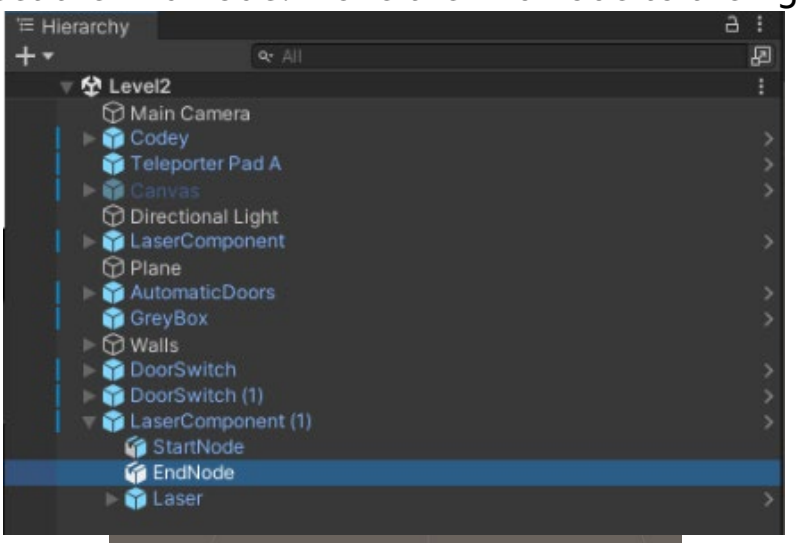
21 This maze looks a bit empty! Let's add some challenge for the player by adding in a laser.

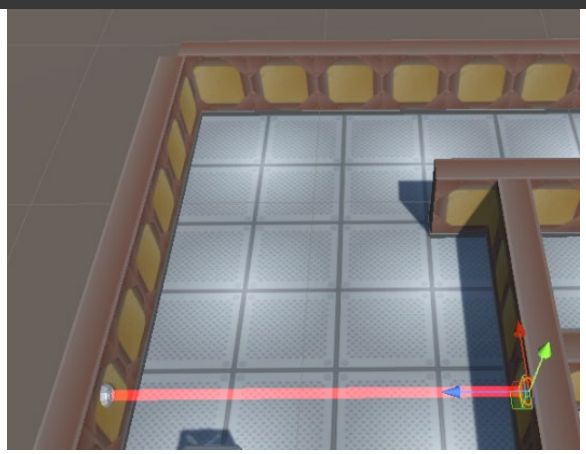
22 In the **Project** tab, in the **Assets > Prefabs** folder, drag the **LaserComponent** object into the scene.



23 Adjust the laser using the transform tool so it is against the left wall.

24 Expand the prefab by clicking on the arrow next to it in the hierarchy, and then select the **EndNode**. Move the EndNode to the right wall.



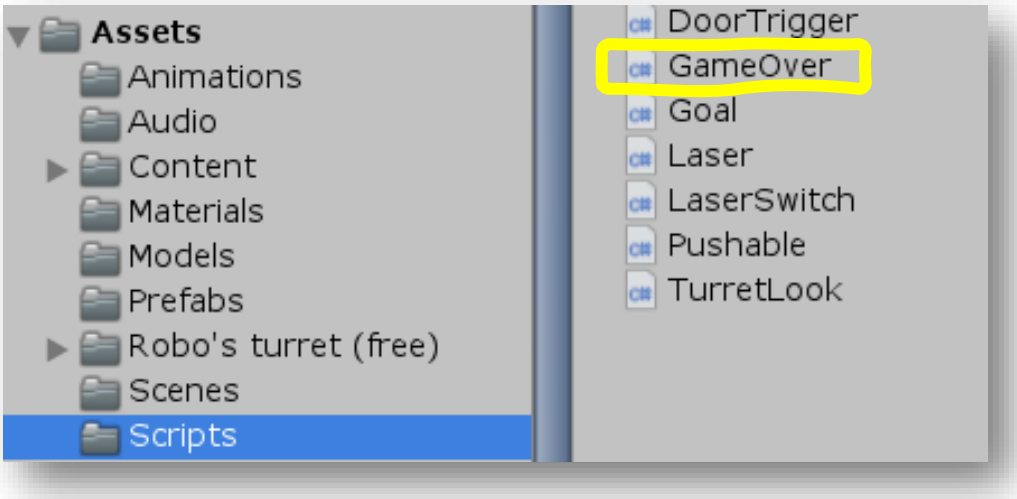


25 Try adjusting the other laser in the scene to go all the way to the other wall!

26 **Playtest** the game. What happens when Codey touches a laser?

27 An evil mad scientist like Doctor Worm needs a little more drama than just resetting the scene. We can add an **animation** that plays whenever Codey touches the laser.

In the **Project** tab under **Assets > Script**, open the **GameOver** script.



28 In the **Game Over** script, scroll down to the `PlayerHit` function.

Inside the function, there is a conditional `if (playerTouchedLaser)` that controls what happens when the player touches the laser.

```
public void PlayerHit()
{
    //Audio for hitting the player
    teleporterAudio.PlayOneShot(laserHit, 1.0f);
    StopAllAudio();

    if (playerTouchedLaser)
    {
        RestartLevel();
    }
}
```

29 We don't want to restart the level immediately; we want Codey to react to touching the laser first. Above the `RestartLevel();` line, type `animator.SetTrigger("PlayerHit");` to tell Codey's animator to play the correct animation.

```
public void PlayerHit()
{
    //Audio for hitting the player
    teleporterAudio.PlayOneShot(laserHit, 1.0f);
    StopAllAudio();

    if (playerTouchedLaser)
    {
        animator.SetTrigger("PlayerHit");
        RestartLevel();
    }
}
```

30 **Playtest** your game. What happens when Codey touches the laser now? Did anything change?

31 We are asking Codey to play his hit animation, but we aren't giving him time to do it!

We are still immediately restarting the scene.

32 In Unity, when you want to delay a function from running, use the `Invoke` function.

Change the `RestartLevel();` line to `Invoke("RestartLevel", 5);` to ask Unity to run the `RestartLevel` function 5 seconds after Codey touches the laser.

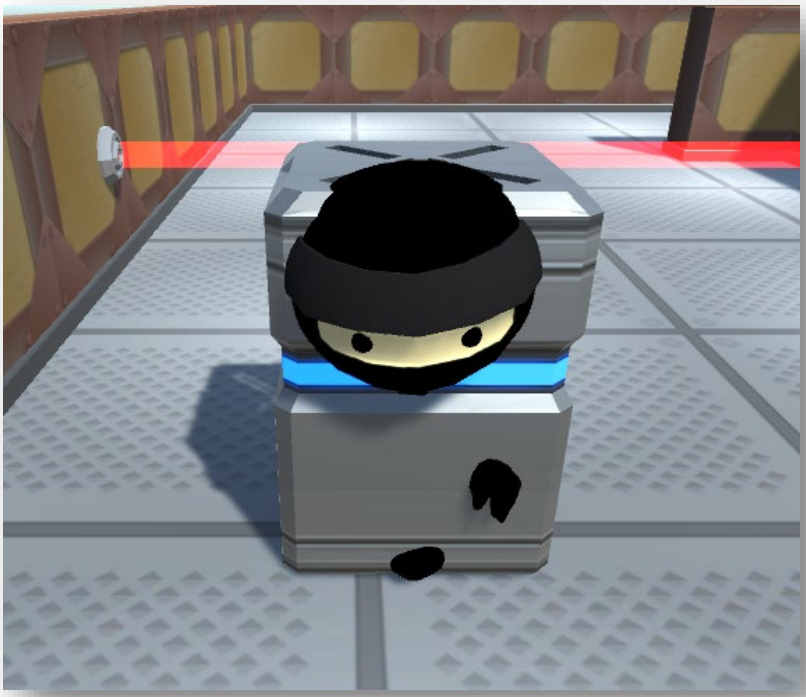
```
public void PlayerHit()
{
    //Audio for hitting the player
    teleporterAudio.PlayOneShot(laserHit, 1.0f);
    StopAllAudio();

    if (playerTouchedLaser)
    {
        animator.SetTrigger("PlayerHit");
        Invoke("RestartLevel", 5);
    }
}
```

33 **Playtest** your game.

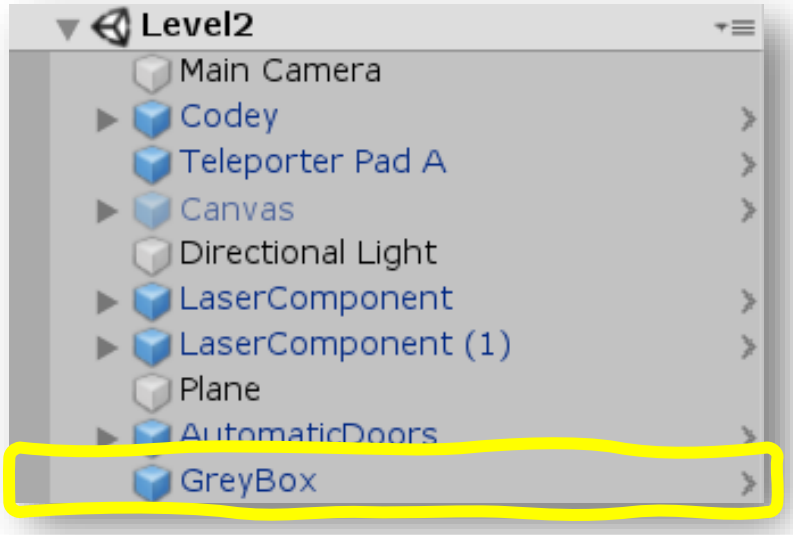
Codey should now properly react when he touches the laser.

What happens if Codey tries to push the box in front of the laser?

34 We need to code the boxes so Codey can push the boxes without running through them!

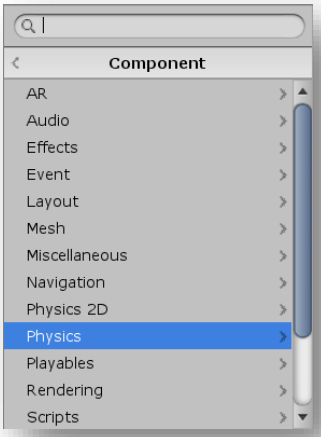


35 Select the **GreyBox** game object in the **Hierarchy**.

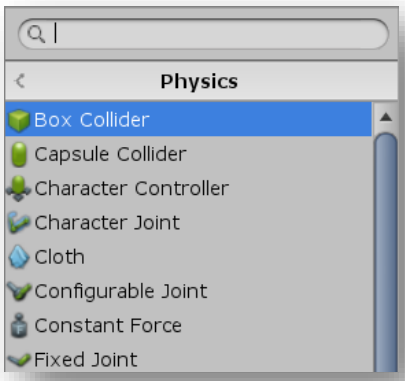


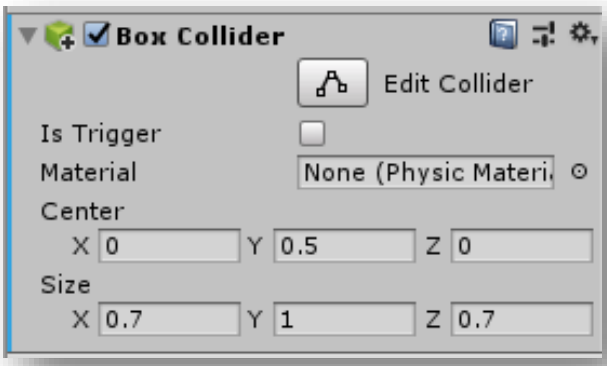
36 In the **Inspector**, click on the Add Component button.

37 Click on **Physics**.



38 Click on **Box Collider** to add a Box Collider component to the **GreyBox**.



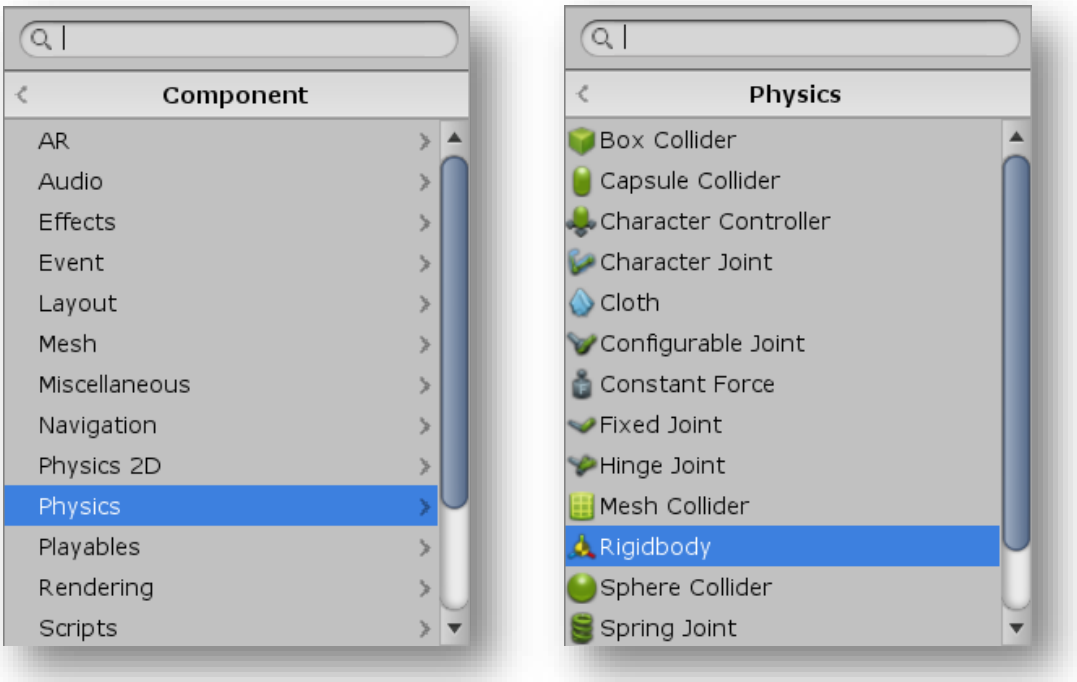


39 **Playtest** your game and try pushing the box into the laser. Codey collides with the box, but it doesn't move!

40 While we were able to make Codey collide with the box, he can't push it yet. This is because we need to add one more component to the box to make it respond to physics!

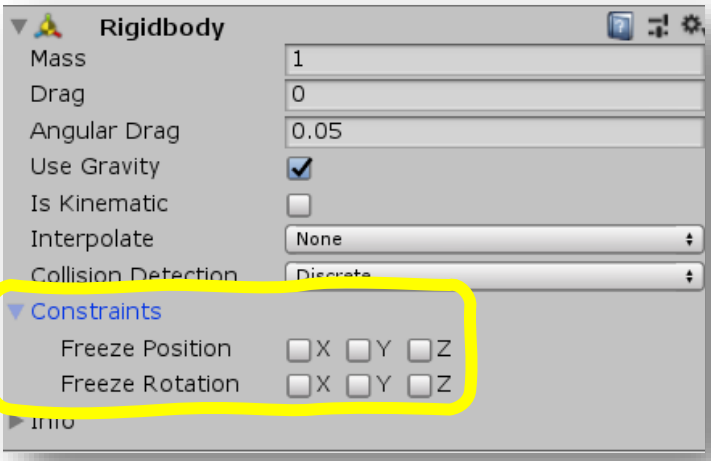
41 With the **GreyBox** selected in the **Hierarchy**, click on the Add Component button in the **Inspector**.

42 Click on **Physics** and then **Rigidbody** to add a Rigidbody component.

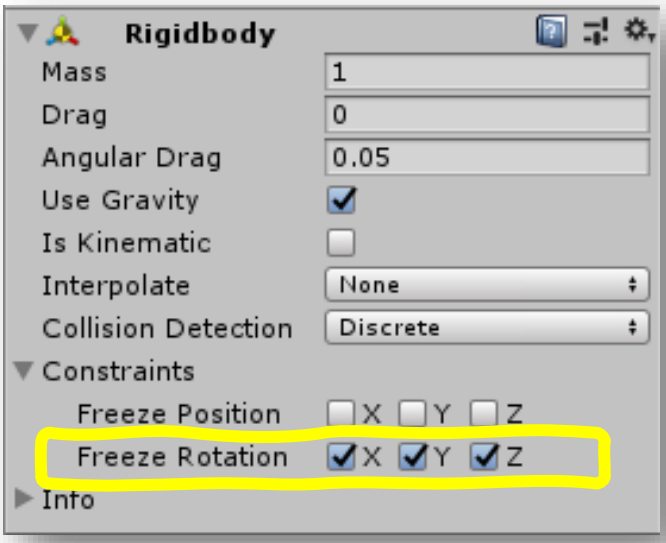


43 **Playtest** your game. What happens when Codey pushes the box? He can tip it over now!

44 On the new **Rigidbody** component, click on the triangle next to **Constraints** to expand the properties.



45 Check all three of the **Freeze Rotation** boxes to prevent the box from tipping over.



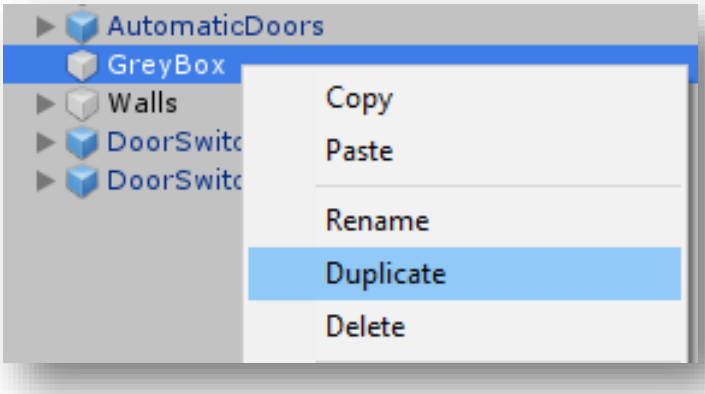
46 **Playtest** the game.

Codey can now get by the first laser!

What about the second laser?

47 We need to make a second **GreyBox** for Codey to push around!

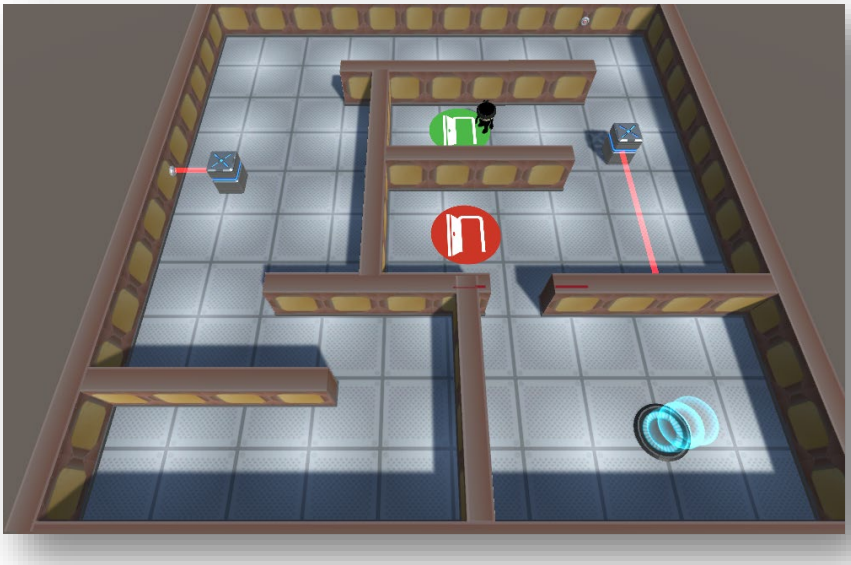
48 Right-click the **GreyBox** game object in the **Hierarchy** and select **Duplicate**.



49 Move the new box somewhere above the first laser in the Scene by dragging the blue and red arrows.

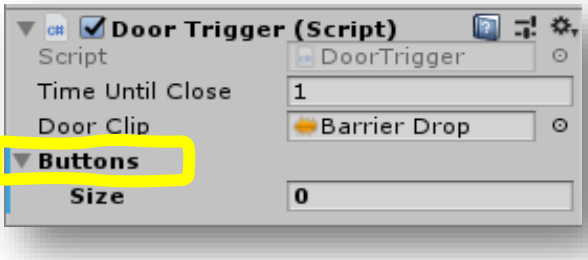
50 **Playtest** the game. Codey can now get by both sets of lasers! Can he open the door yet?

51 Codey can open the door by standing on just one of the switches. Doctor Worm needs to fix his fortress!

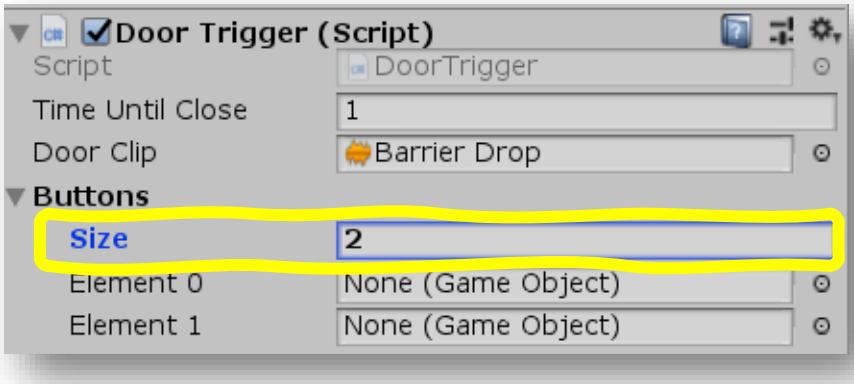


52 Select the **AutomaticDoors** object in the **Hierarchy**.

53 Find the **Door Trigger (Script)** and expand the **Buttons** property.

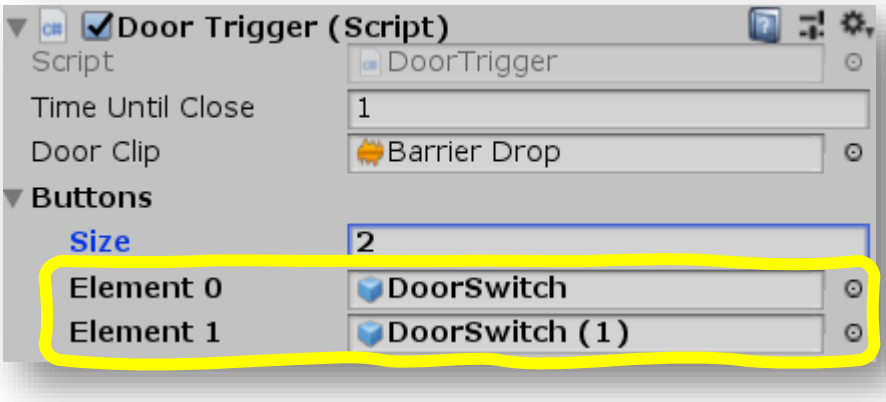


54 Change the value of **Size** from 0 to 2.

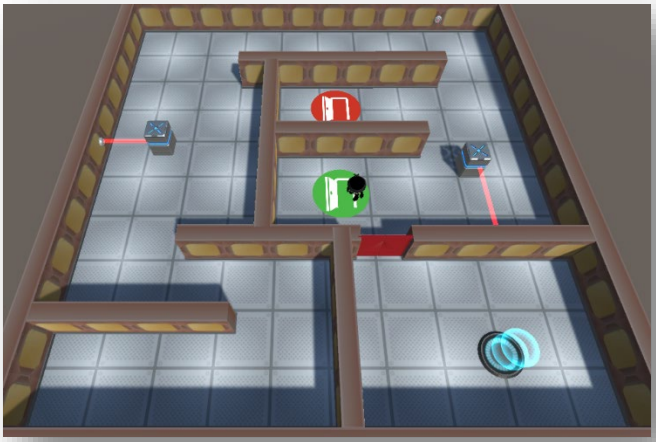


55 Attach the **DoorSwitch** object to the **Element 0** property and the **DoorSwitch (1)** object to the **Element 1** property.

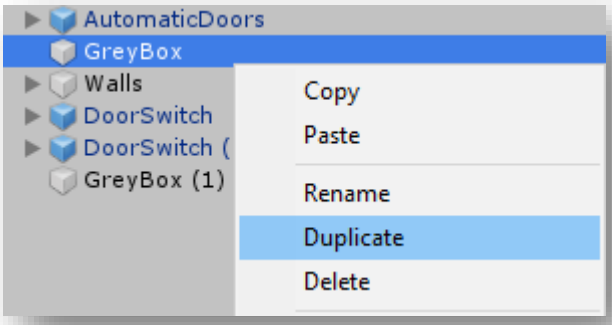
Either drag the objects from the **Hierarchy** to the property in the **Inspector** or click on the little circle to open the **Game Object Selector**.



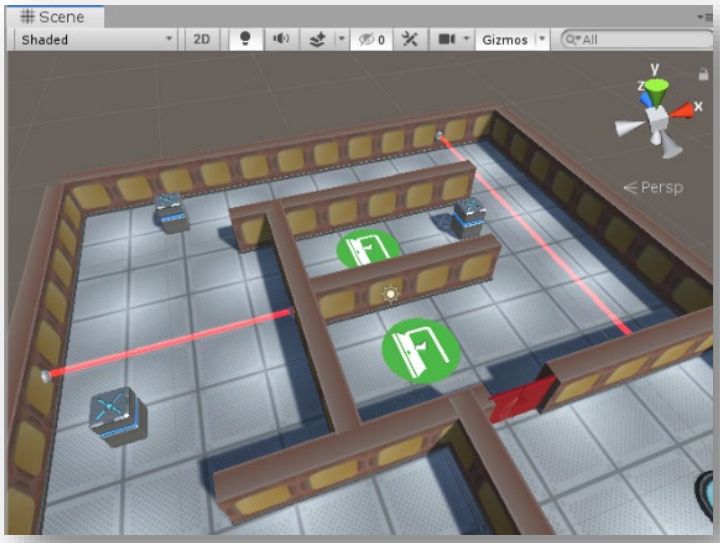
56 Playtest the game. The switches are now properly wired to the door and Codey can't escape!



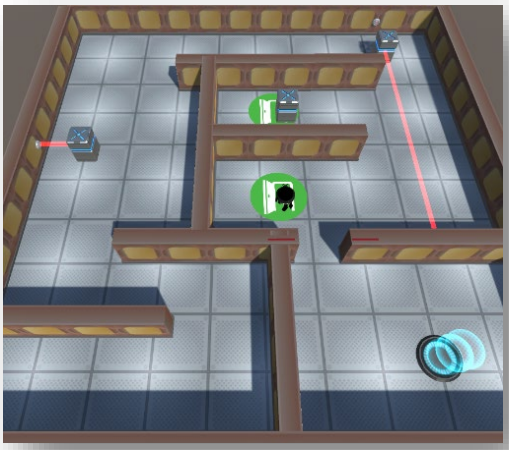
57 How can Codey get out?! We can duplicate the GreyBox one more time. Right-click the **GreyBox** object in the **Hierarchy** and select **Duplicate**.



58 Move the new **GreyBox** somewhere near the two switches using the blue and red arrows.

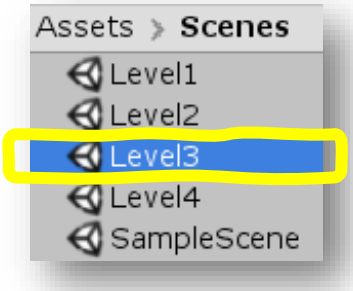


59 Playtest the game. Codey can successfully escape!

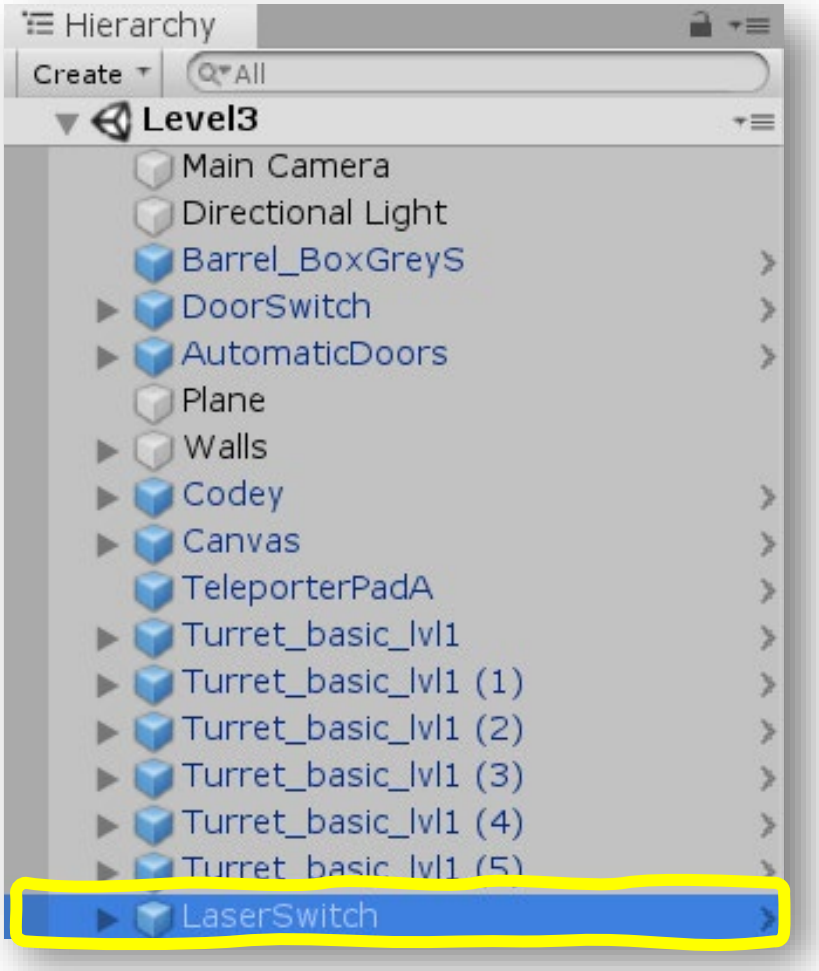


Codey's adventure continues in the Evil Fortress of Doctor Worm part 3! Begin part 3 on the next page.

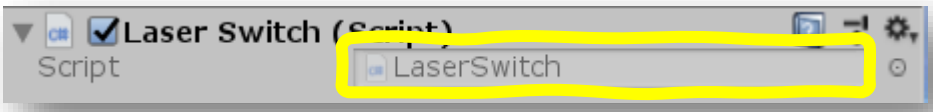
60 Open your Fortress of Doctor Worm Unity project and load the Level 3 scene.



61 Let's get the laser power switch working first. Select the **LaserSwitch** in the **Hierarchy**.



62 In the **Inspector**, find the **Laser Switch (Script)** component and open the attached script in Visual Studio.



63 We need to add a Boolean to the LaserSwitch script that we can use to inform the laser objects to turn off when Codey steps on the switch.

After `private AudioSource playBeep;` type `public bool lasersAreOff = false;` to create a Boolean with a value of false.

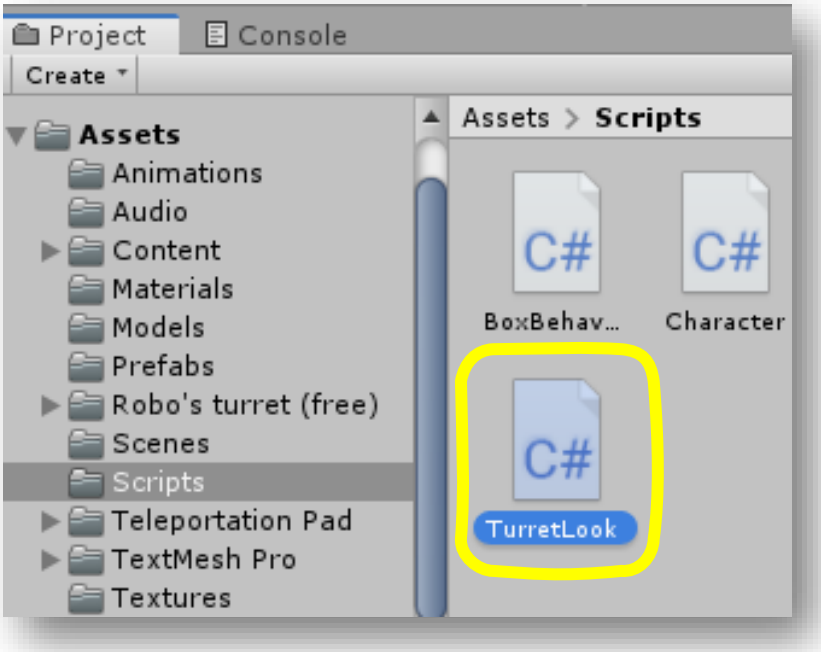
```
private GameObject switchIcon;
private AudioSource playBeep;

public bool lasersAreOff = false;
```

64 We want to turn the lasers off when Codey or the box enters the switch. In the `OnTriggerEnter` function, add `lasersAreOff = true;` after the `playBeep.Play();` line.

```
public void OnTriggerEnter(Collider other)
{
    playBeep.Play();
    lasersAreOff = true;
}
```

Save your script.

65 Each laser turret has a script called **TurretLook**. To open it, find it in the **Assets > Scripts** folder and double-click.



67 In the script's `Update` function, create an `if` statement that checks to see if the `laserSwitch`'s `lasersAreOff` Boolean is true by typing `if (laserSwitch.lasersAreOff) { }` making sure to leave an empty line in between the curly brackets.

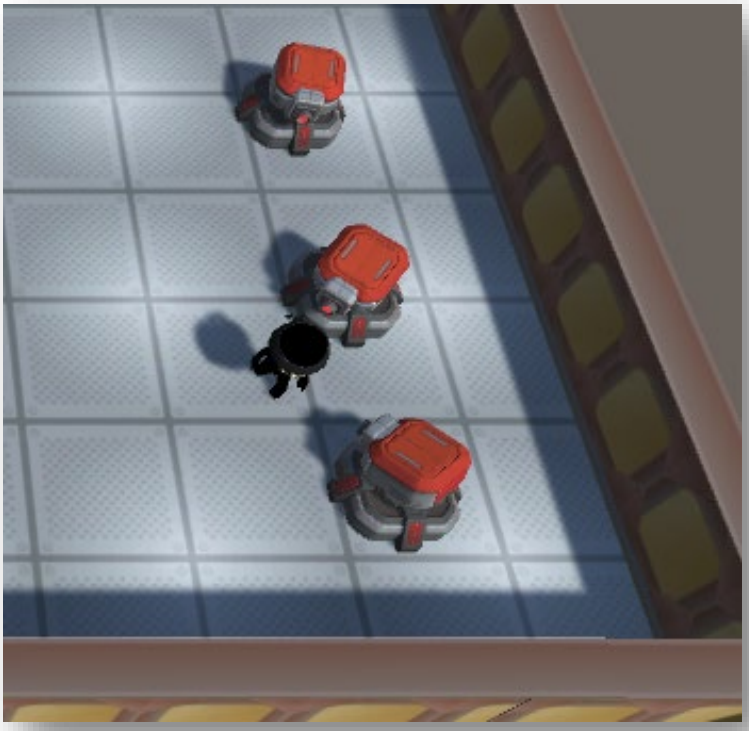
```
void Update()
{
    if (laserSwitch.lasersAreOff)
    {

    }
    transform.LookAt(player);
}
```

68 Code added within this `if` statement will execute if the switch has turned the lasers off.

We need to disable the lasers for Codey, so type `laser.SetActive(false);` inside the curly brackets.

```
void Update()
{
    if (laserSwitch.lasersAreOff)
    {
        laser.SetActive(false);
    }
    transform.LookAt(player);
}
```

69 **Play** your game. The lasers turned off, but the turrets still follow Codey!



70 We can make sure the turrets are shut down for good with one simple line of code. After the `laser.SetActive(false);` line, type `return;` This exits the Update function early and prevents the `transform.LookAt(player);` code from running.

```
void Update()
{
    if (laserSwitch.lasersAreOff)
    {
        laser.SetActive(false);
        return;
    }
    transform.LookAt(player);
}
```

71 **Playtest** your game. The turrets are now completely disabled!