CODE NINJAS®

Silver Belt Ninja Guide
Activity 05: Ninja Run

Activity 5

Ninja Run

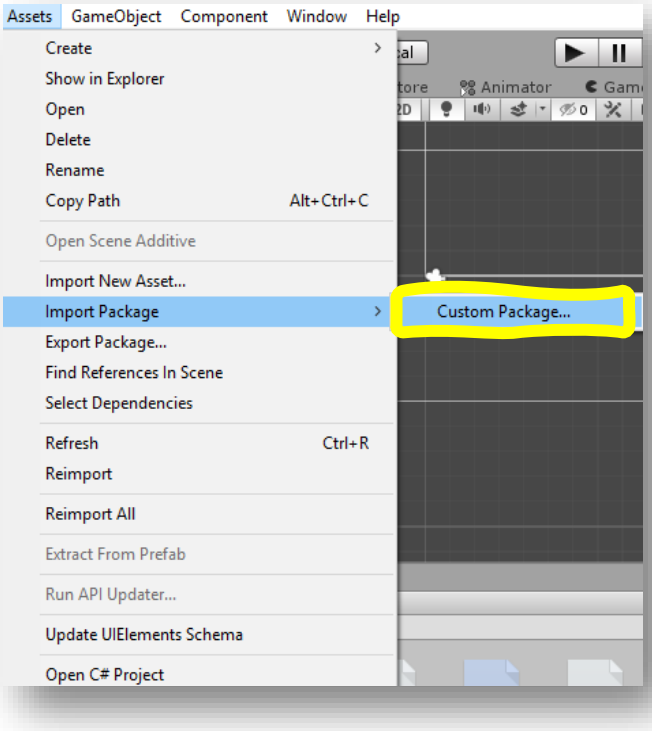
You will be able to create a pickup item with a tag and trigger collision which will be utilized to code a scoring system and control particle effects.

Your mission: With three lives, it is your goal to navigate the endless castle walls to pick up as many coins as possible. But beware – the further you get, the faster you go! Press play to test out the game.

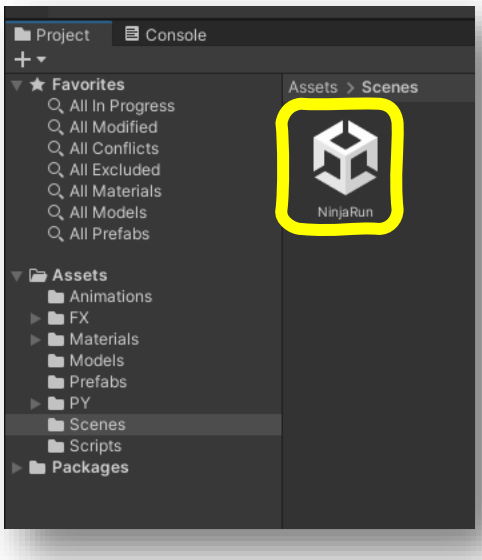
After playing, you may have noticed that Codey can't pick up any coins! Alright ninja, we've got work to do. We need to code our pickups to work, complete with score and particle effects.

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

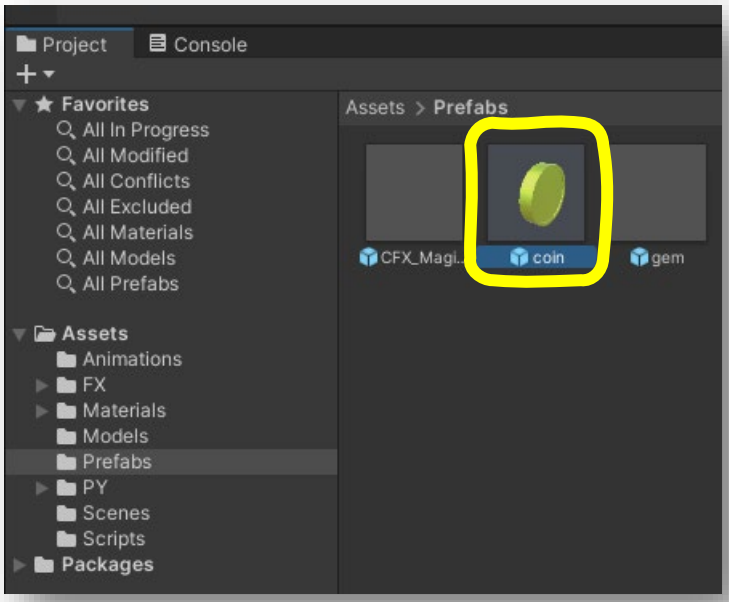
2 Import the **Activity 05 - Ninja Run** starter Unity Package by going to **Assets > Import Package > Custom Package > All > Import**.



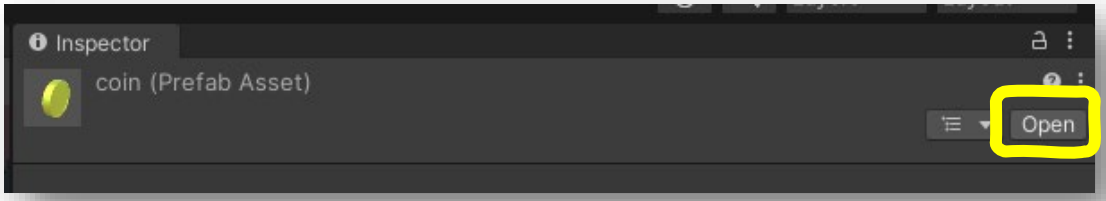
3 Double-click on the **NinjaRun** scene. You can find this in the **Project** tab under **Assets > Scenes**.



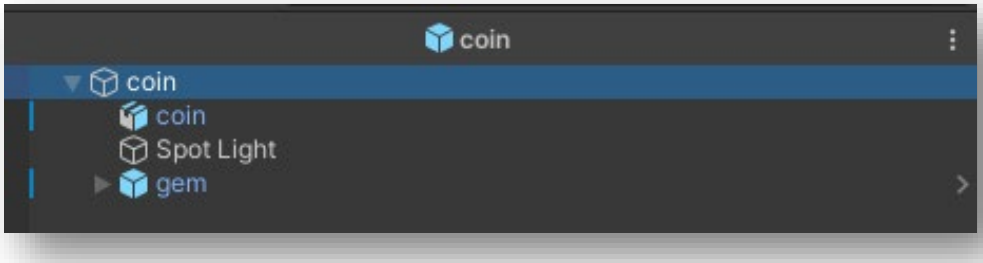
4 Let's start by creating our pickup item! Under your **Project** tab in the **Prefabs** folder, click on the **coin** prefab.



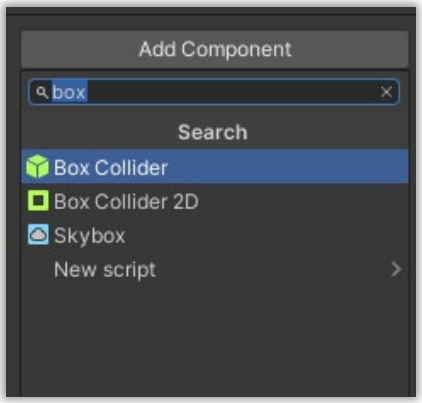
In the **Inspector**, click on the **Open Prefab** button. Here we can get a closer look and make any edits we need.

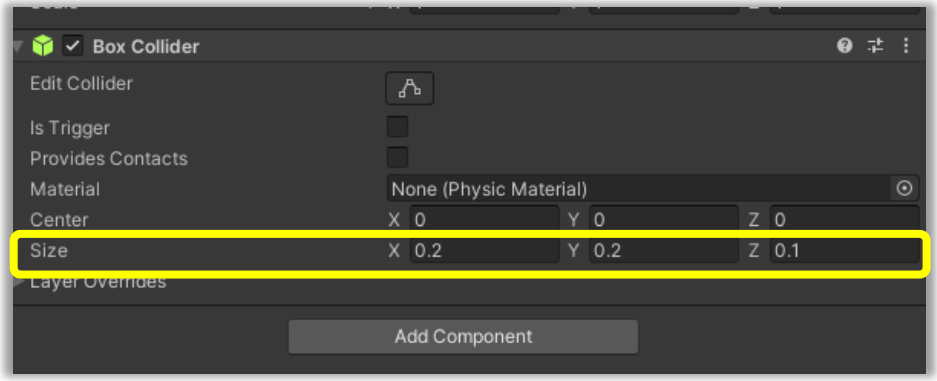


5 Notice that there is no collider component. Remember, a collider defines the space of the coin; without it, Codey can't interact with the coin. Make sure to select the *parent* of the coin prefab.



6 With the parent selected, add a **Box Collider** component and shape it to match the size of the coin. We used: (0.2, 0.2, 0.1)

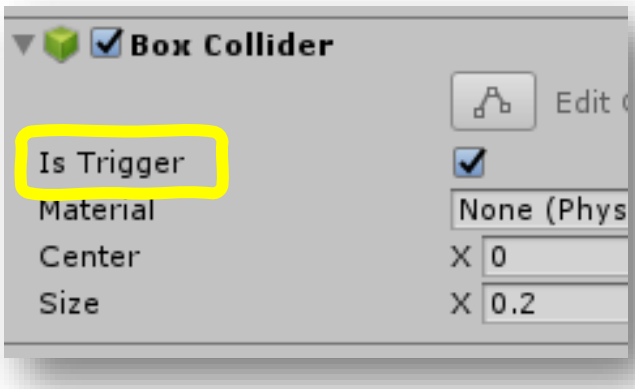




7 Press **play**. Now when Codey hits a coin, he gets stuck! The colliders are doing their job and bumping into each other. We need to know when Codey hits the coin without the physical collision.

8 To fix this, we will make the collider into a trigger. A trigger will set off when hit by another collider, but without the physical collision. This is exactly what we need!

To make a collider a trigger, check the box for **Is Trigger** in the **Box Collider** component within the **Inspector**.



9 Great job! The trigger is set up in the scene. Now, how can we use this in our code? Open the **Pickups** script – you can find it in the **Scripts** folder or on **Codey** which is a *child object* of **Player** in the **Hierarchy**. Delete the `Update()` function and replace it with:

```
void OnTriggerEnter(Collider other)
{

}
```

OnTriggerEnter()

We are using OnTriggerEnter(). This means that the function is called when the object enters the trigger.

There are also functions such as OnTriggerExit(), called when object exits trigger, and OnTriggerStay(), called while the object is inside the trigger.

10 In your `OnTriggerEnter` function, add `Debug.Log("Trigger Enter")`.

11 **Save** your work and go to your Unity window. Press **play**.

Check the console to see when the `OnTriggerEnter` function gets called.

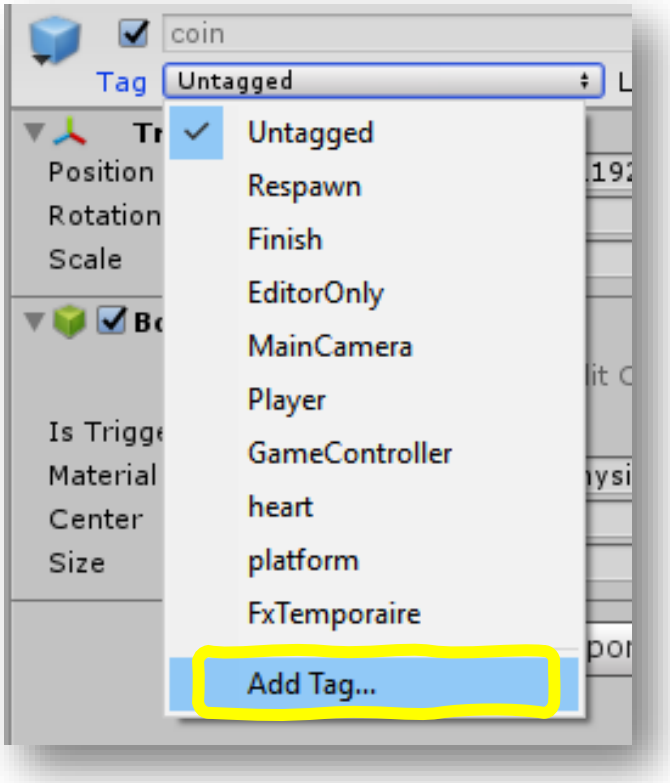
12 The coins are the only objects in the scene with a trigger, so this works perfectly.

However, what would happen if there was an obstacle with a trigger? How would Unity know the difference?

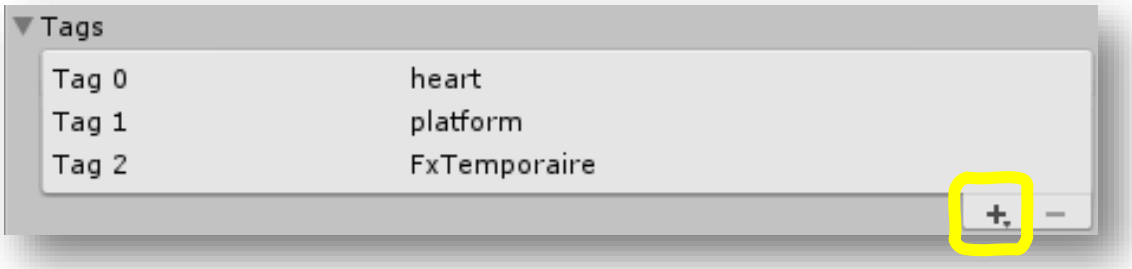
We can add a tag to objects and check the tag in the script! **Tags** are like a label that lets you easily find multiple objects with one name (or tag) in your code.

13 In your **Project** tab under **Prefabs**, click on the **coin**.

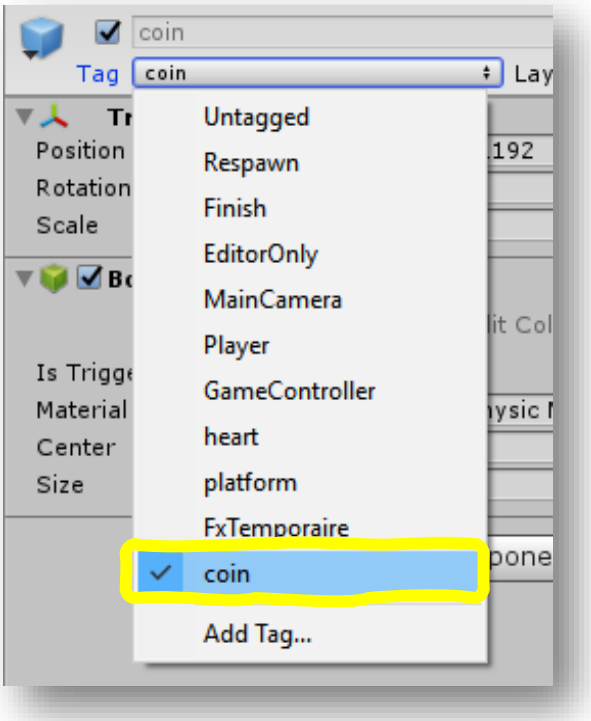
In the **Inspector** under the game object name, there is a drop-down for your tag. Right now, it is Untagged. Click on the drop-down and select **Add Tag**:



14 Click the **+**, type *coin*, and click **Enter**.



15 Click on the coin again in your **Prefabs** folder. Now when you click on the drop-down for the tag, you can select **coin**.



16 Now we will use `CompareTag()` to check the tag. Open your **Pickups** scripts, in `OnTriggerEnter()`, delete the `Debug.Log()` and add this conditional statement:

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("coin"))
    {

    }
}
```

17 Now Unity knows when Codey touches a coin, but the coin still stays in the scene!

This can be confusing because it looks like Codey didn't pick up the coin at all.

18 To remove the coin from the scene when Codey touches it, we will use the `Destroy()` function. To do this, open the **Pickups** script and add:

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("coin"))
    {
        Destroy(other.gameObject);
    }
}
```

Disappearing Game Objects

What if you wanted to destroy a game object not involved in the collision? For instance, if you wanted a door to disappear when the player picks up a key, the door is not involved in the collision between player and key. You would have to declare game object door and then use Destroy(door).

18 Remember the coins are the source of points. The User Interface (or UI for short) for the score is already in the scene. However, it doesn't update when you pick up a coin; the score just stays at 0.

19 In **Pickups**, declare `public int score` at the top.

```
public class Pickups : MonoBehaviour
{
    public int score;
    public Text scoreText;

    void Start()
    {
```

20 Right now, the score is staying at 0, but we want it to go up when we hit a coin. We'll need to include `score++` like this:

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("coin"))
    {
        score++;
        Destroy(other.gameObject);
    }
}
```

Remember ++ adds 1 to the variable.

21 Press **play**. The text score is still not updating!

Can you guess why?

The score variable is not yet connected to the text object.

In the meantime, you can test if the score is working properly by adding:

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("coin"))
    {
        score++;
        Debug.Log(score);
        Destroy(other.gameObject);
    }
}
```

Now the score will be output to the console.

22 *Scripting UI* will be explained further in a later section; for now, let's go over sending the score integer to the score text object.

The score text object is already declared in your code as `scoreText`. We will use the `ToString()` function to convert the `score` integer to a string. This way it can be displayed as text.

We will assign the scoreText.text value to the stringified version of `score`:

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("coin"))
    {
        score++;
        scoreText.text = score.ToString();
        Destroy(other.gameObject);
    }
}
```

23 Great Job! We just need to add some final details to really emphasize the pickup and add some fun. A great way to do this is with a particle system. Remember making them in Bronze Belt?

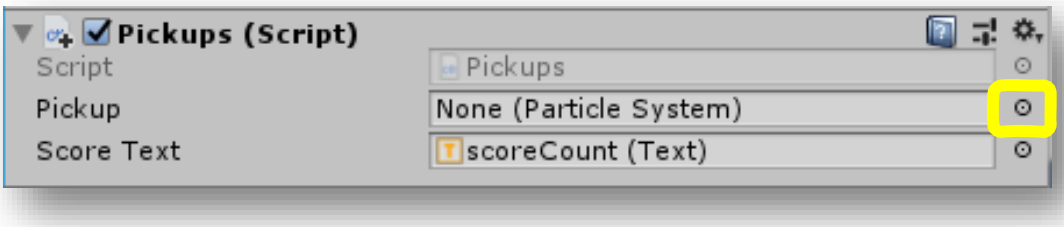
Well now you are going to learn how to control it in your code!

24 In **Pickups**, declare `public ParticleSystem Pickup`:

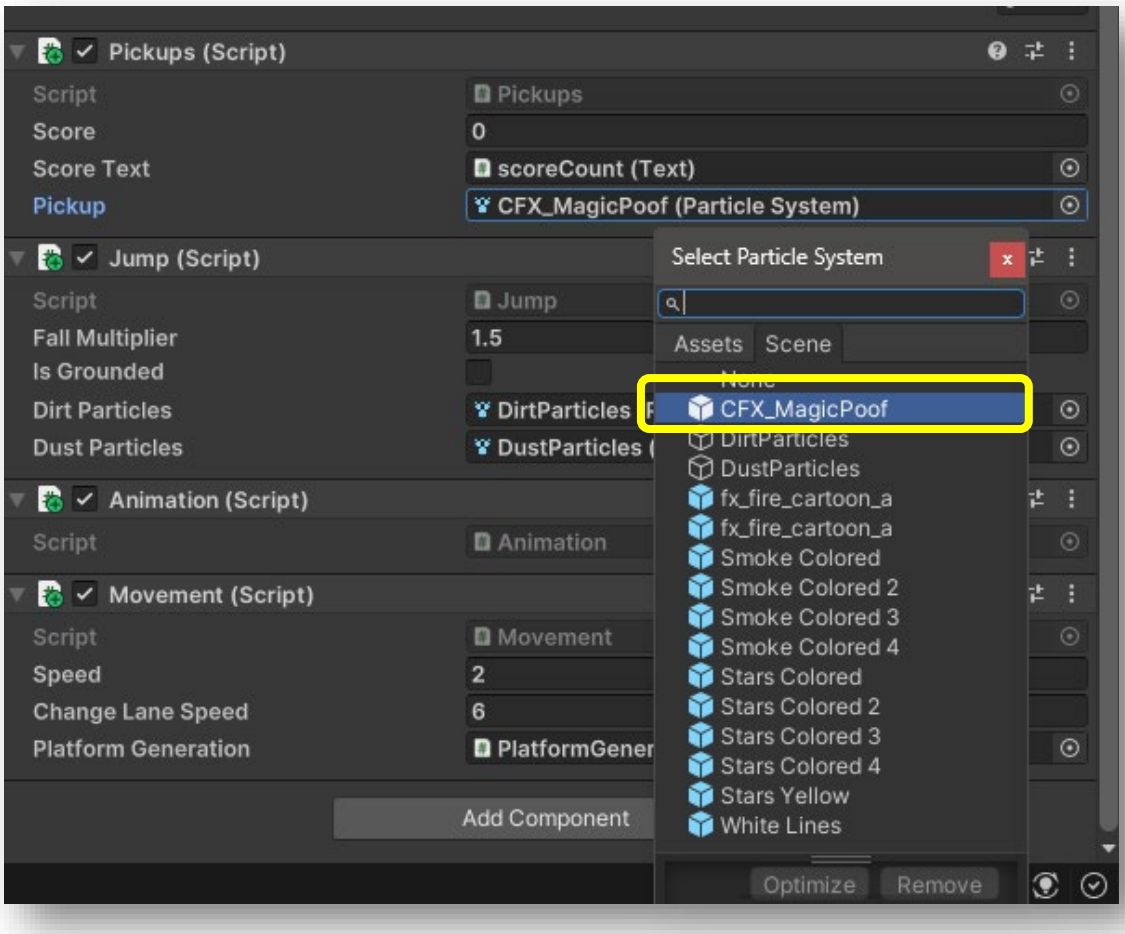
```
public class Pickups : MonoBehaviour
{
    public int score;
    public Text scoreText;

    public ParticleSystem Pickup;

    void Start()
    {
```

25 **Save** the code and switch over to your Unity window.

With **Codey** selected, look in the **Inspector** for the **Pickup** script component and click on the circle to the right of the **Pickup** slot.



In the pop-up window, select **CFX_MagicPoof**.



26 Remember this emphasizes that a coin's been picked up, so it should only play when the trigger event is called. We will use the functions `ParticleSystem.Start()` and `ParticleSystem.Stop()` which tells Unity when to play or stop a particle system.

Back in our **Pickup** script, under `Start()`, add:

```
void Start()
{
    Pickup.Stop();
}
```

In `OnTriggerEnter()`, add:

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("coin"))
    {
        score++;
        scoreText.text = score.ToString();
        Destroy(other.gameObject);

        Pickup.Play();
    }
}
```

Remember to **save**.

Yay! You did it, ninja! Codey is ready to go.

Before moving on to the Prove Yourself, challenge yourself to see how many coins you can collect.