Projects

Star collector

Make a minigame where you collect sparkling, spinning, stars as fast as you can

Step 1 Introduction

Make a minigame where you collect spinning stars as fast as you can.

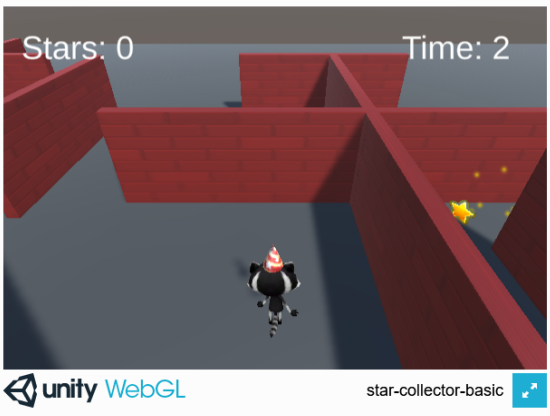
This project is brought to you with generous support from **Unity Technologies** (https://unity.com/). These **projects** (https://projects.raspberrypi.org/en/pathways/unity-intro) offer young people the opportunity to take their first steps in creating virtual worlds using Real-Time 3D.

This project follows on from **Explore a 3D world** (https://projects.raspberrypi.org/en/projects/explore-a-3d-world). You can use the Unity scene that you created in that project as the basis for this project. We've also provided a starter project that you can use.

> A **minigame** is a short computer game. Larger computer games often contain multiple minigames. Do you play any games that contain minigames?

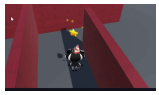
You will:

- Use **colliders** and **triggers** to control what happens when GameObjects collide
- Add **sound** and **particle effects**
- Create, set, and display score and time **variables**



Step 2 A spinning star

The collectibles in this game are stars that spin to attract attention.



Launch the Unity Hub and open the project you created for **Explore a 3D world** (https://projects.raspberrypi.org/en/projects/explore-a-3d-world).

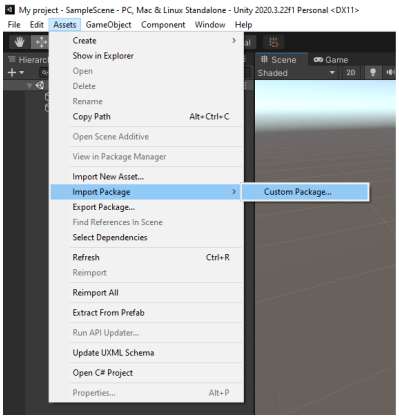
I haven't got my Explore a 3D world project

If you are not able to open your Explore a 3D world project, you can download, unzip, and import this **Star collector starter package** (https://rpf.io/p/en/star-collector-go).

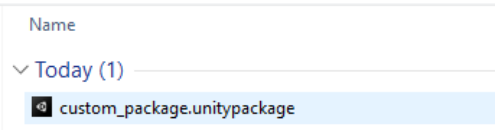
After the package has been imported, go to the Assets folder and double-click on the **3D World** scene to load it.

Import a package

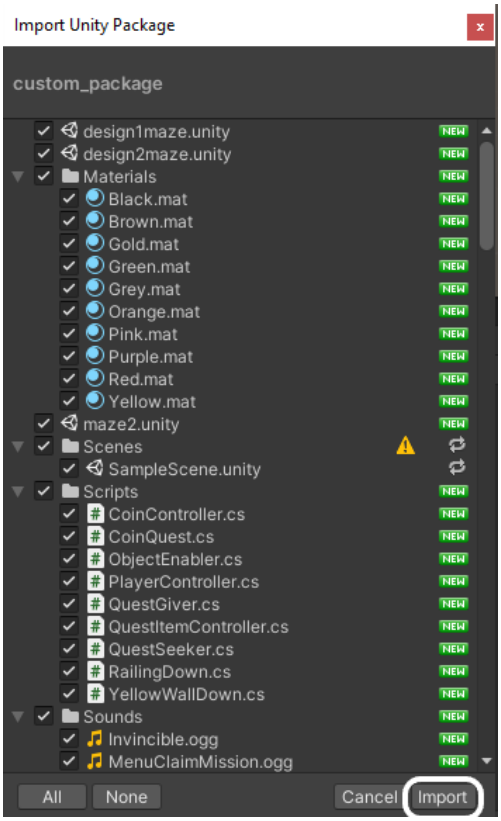
Import your downloaded package.



Select the location of your downloaded package.



Click on the **Import** button to import all of the package.



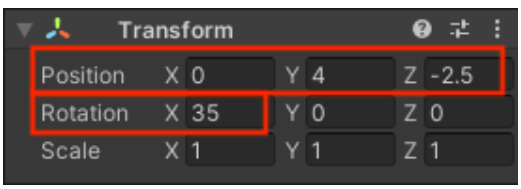
Right-click on the **3D World** scene in the Hierarchy window and **Save Scene As** `Star Collector`.

This creates a new Scene file in the Project window. Scenes in a project can share Assets including Scripts.

Your project now contains two scenes, but you will only work on one scene at a time.

The star collector minigame needs a camera view that is high enough to view the layout of some of the map but not too high or it will reveal the position of the stars.

In the Hierarchy window, click on **Player** then select **Main Camera**, and change the Position and Rotation in the Inspector window's Transform component to:



You're also going to need to add a few more walls to your scene. Click on a wall and press `Ctrl+D` to duplicate the wall.

Position the new walls using the transform and rotate tools or by changing the values in the Transform component. Repeat this several times, so that you have plenty of places to hide stars.

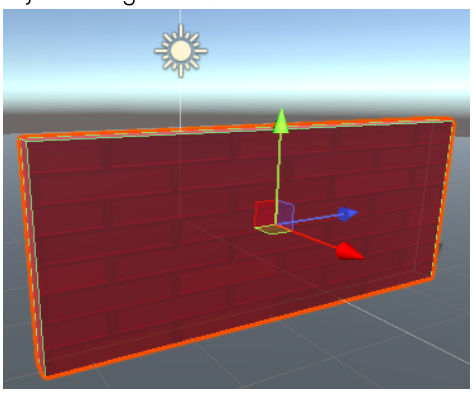
You can navigate around your scene to see it from different angles. If you get lost, click on your Player in the Hierarchy and then use `Shift+F` to focus on the Player.

Transform tools

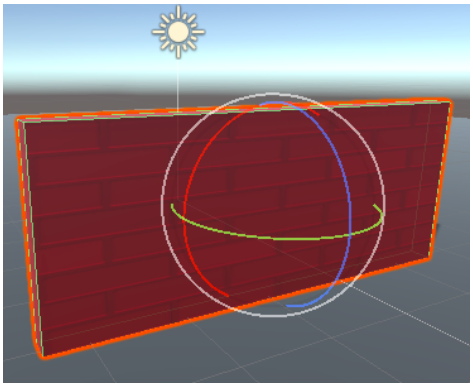
The Transform tools allow you to move around in 3D space in the Scene view and move, rotate, and scale your game objects.

You can click on a tool to start using it, or use a keyboard shortcut:

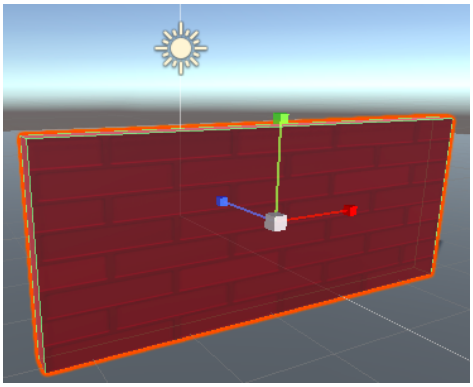
- Q, Hand – Pan around.
- W, Translate – Move a game object. Drag the coloured arrows to move in x, y, z directions.



- E, Rotate – Rotate a game object. Drag the coloured circles to rotate in x, y, z directions.



- R, Scale – Resize a game object. Drag the coloured cubes to resize an object in x, y, z directions.



- T, Rect – Change a 2D object such as text.

You can also change values in the Transform window of a Game object in the Inspector.

Tip: Sometimes it's easier to drag an object to roughly the right place using the Transform tools and then adjust the values to round numbers in the Transform for accurate positioning.

Tip: The directions are coloured-coded in the Scene view: x is red, y is green (up and down), and z is blue.

Moving in the Scene view

To use flythrough mode, hold down the **right** mouse button **and use**:

- WASD to move around
- QE for up and down
- Shift to go faster
- The mouse to rotate the camera

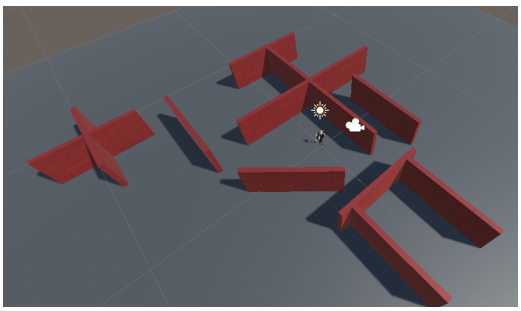
Use the scroll wheel on your mouse to zoom in and out. Your trackpad may also have a scroll motion.

To move the scene around, hold the `ALT` key and the middle mouse button and then drag to move. You can also use the arrow keys to move around.

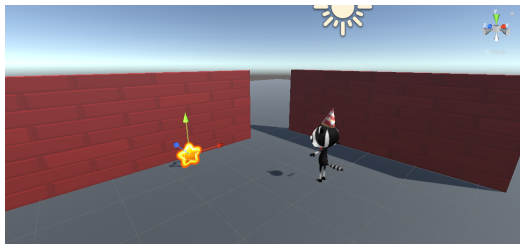
To focus on an object, click on the GameObject in the Scene view and then click `F`.

You can also click on an object in the Hierarchy window and then click `Shift`+`F` to focus on that GameObject in the Scene view.

Tip: If you get lost, click on your Player in the Hierarchy window and then `Shift`+`F` to focus on the Player. Then you can use the scroll wheel to zoom out.



In the Project window, go to the **Models** folder and drag the **Star** into the **Scene view**.

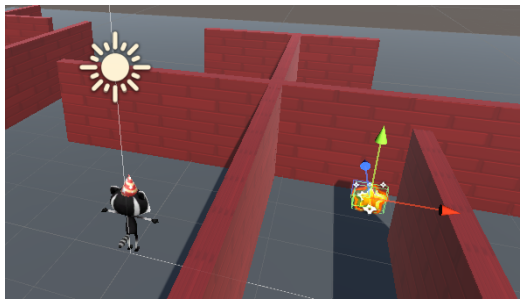


Make sure the Star GameObject is selected in the Hierarchy window and position it using either:

- The arrows from the Transform tool and the Scene view
- The coordinates from the Transform component in the Inspector window

Your star should be off the ground; position `y = 0.7` is about right.

You might want to hide the star behind a wall so it's harder for players of your game to find:



In the Inspector window, click **Add Component** and choose **New script**, then name your new script `StarController`.

Double-click on `StarController` in the script component to launch your script in the editor.

In **Explore a 3D world** (https://projects.raspberrypi.org/en/projects/explore-a-3d-world/) you used `transform.Rotate` to turn the Player. You can use the same method to spin the Star around the y-axis.

Underneath the public class code, create a variable called `spinSpeed` so you can control how fast your star spins:

StarController.cs

```
5  public class StarController : MonoBehaviour
6  {
7      float spinSpeed = 0.5f;
```

Add code to spin your star:

StarController.cs - Update()

```
16 void Update()
17 {
18     transform.Rotate(Vector3.up * spinSpeed); // Rotate about the y (up) axis
19 }
```

Save your script then return to the Unity Editor.

Test: Play your scene and check that the star is spinning:

Debug: Make sure you added the Script to the Star GameObject. If you accidentally added it to a different GameObject, then you can click the three dots next to the Script component and choose **Remove Component**.

Debug: Change the value of your `spinSpeed` variable if you want to speed up or slow down the speed at which the star spins.

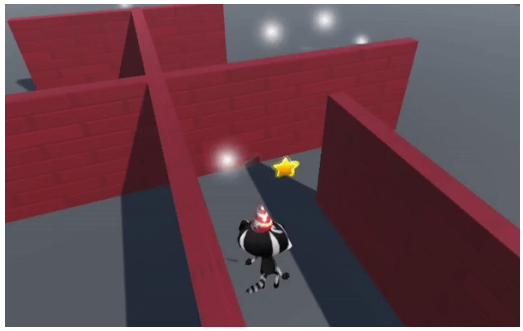
Time for a particle system.

> A **particle effect** uses lots of small images, or 'particles', to create a visual effect that adds life to a computer game. Next time you play a computer game, look out for all the places where particle effects are used.

Right-click on the **Star GameObject** in the Hierarchy window and choose **Effects** then **Particle System**. This will add a Particle System GameObject to the Star.

Adding the Particle System as a child object of the Star means that if you move the star in Scene view, the particles will move with it.

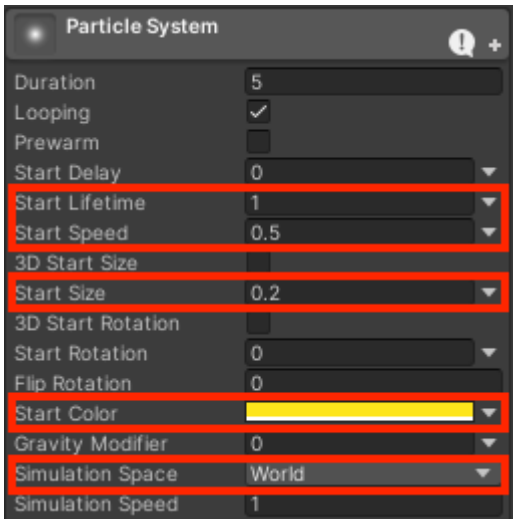
Test: Play your scene to see the default particle effect. It's spinning with the star and it's not quite right for a sparkling star:



Exit Play mode.

There are lots of settings that you can use to customise the Particle System.

Click on **Particle System** beneath the Star in the Hierarchy. Use these settings to create a sparkle effect that doesn't spin with the Star:



Tip: To close the colour picker, click on the 'X' or click elsewhere in the Unity Editor.

Test: Click **Play** to see the effect.

Adjust the settings until you are happy with the particle effect.

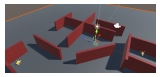
Remember, you can try things out in Play mode, but you need to exit Play mode to make changes that you want to keep:

Now that star is just asking to be collected!

Save your project

Step 3 Collecting the star

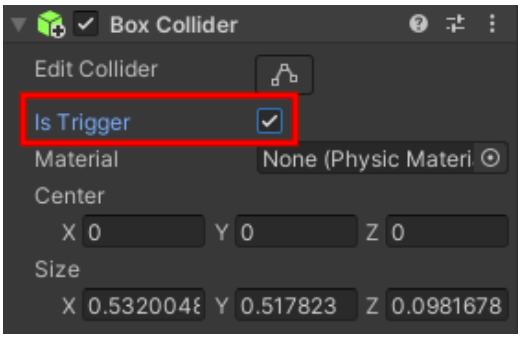
The star needs to disappear when you collect it.



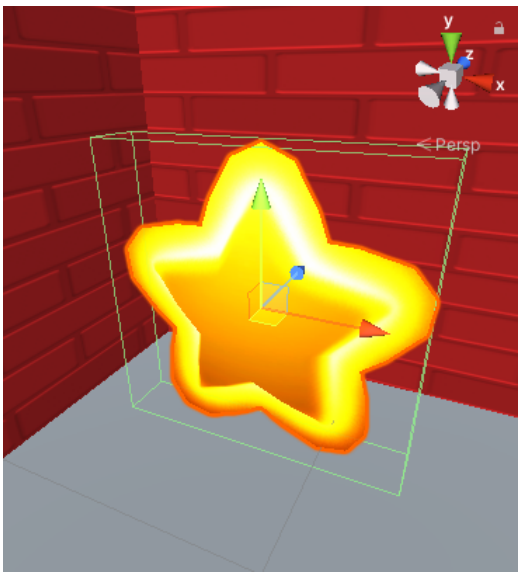
In Unity, a Collider with a **Trigger** calls the `OnTriggerEnter` method when a collision happens, but it does not prevent a Player walking into the Collider.

Select the **Star** and in the Inspector window, click **Add Component**. Start typing `box` until you see **Box Collider** and click it. A new component will be added to the Star in the Inspector window.

Check the **Is Trigger** box.

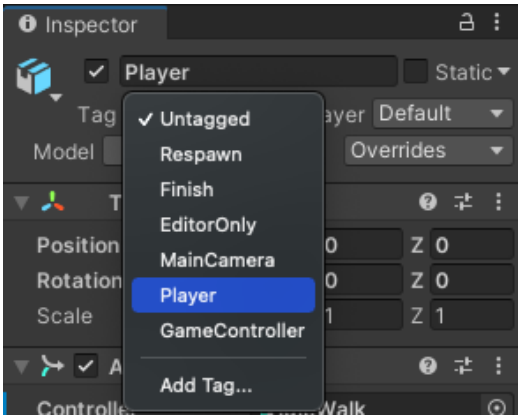


Click `Shift+F` to focus on the Star in the Scene view. You will see a green box outline around the Star: this shows the outline of the Collider. If the Player's Collider enters this area, then there will be a collision and `OnTriggerEnter` will be called:



You only want the star to be collected if the GameObject that has collided with it is the Player. Unity uses **Tags** to label GameObjects. Unity includes a Player tag.

Select your **Player** GameObject and set its Tag to `Player` using the drop-down menu:



Open your StarController script by switching to your code editor or double-clicking on the script in your **My Scripts** folder from the Project window.

Add a new `OnTriggerEnter` method under the closing `}` of the `Update` method but before the closing `}` of the `StarController` class:

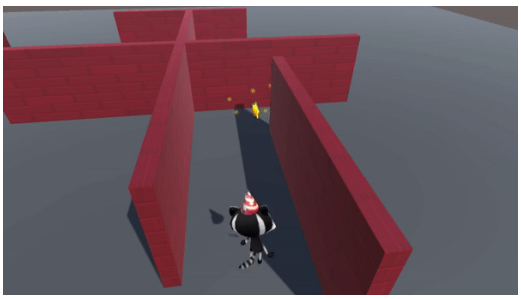
StarController.cs - OnTriggerEnter(Collider other)

```
16  void Update()
17      {
18          transform.Rotate(Vector3.up * spinSpeed); // Rotate about the y (up) axis
19      }
20      void OnTriggerEnter(Collider other)
21      {
22          // Check the tag of the colliding object
23          if (other.CompareTag("Player"))
24          {
25              gameObject.SetActive(false);
26          }
27      }
28  }
```

Save your script.

Test: Play your project. Walk into the star to see it disappear.

Debug: Make sure you have added the Player tag to your Player GameObject and not to the Star!



Adding a sound effect makes collecting a star more satisfying for the player.

Add a public `collectSound` variable to your `StarController` script to store the sound that you want to play:

StarController.cs

```
5  public class StarController : MonoBehaviour
6  {
7      float spinSpeed = 0.5f;
8      public AudioClip collectSound;
```

Making a variable `public` means you can assign it in the Inspector and access it from other GameObjects.

Add a line to the `OnTriggerEnter` method to play the sound at the location of the star. The `AudioSource.PlayClipAtPoint` method will play the sound:

StarController.cs - OnTriggerEnter(Collider other)

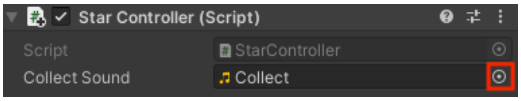
```
21 void OnTriggerEnter(Collider other)
22 {
23     // Check the tag of the colliding object
24     if (other.CompareTag("Player"))
25     {
26         AudioSource.PlayClipAtPoint(collectSound, transform.position);
27         gameObject.SetActive(false);
28     }
```

Save your code.

Switch back to the Unity Editor and click on the **Star GameObject** in the Hierarchy window.

Find the **Collect Sound** property of the Star's StarController script component in the Inspector window.

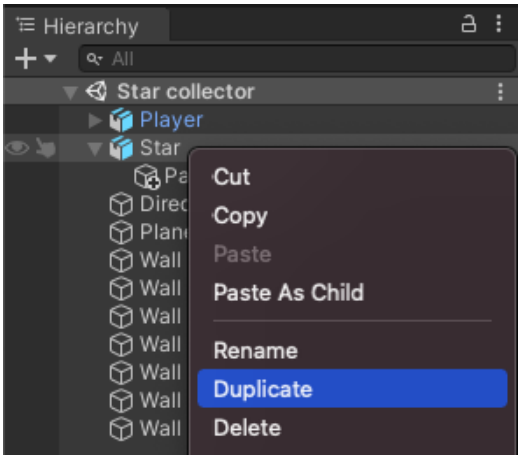
Click on the circle to the right of the Collect Sound property and choose the **Collect** sound:



Test: Play your scene and collect the star to hear the sound.

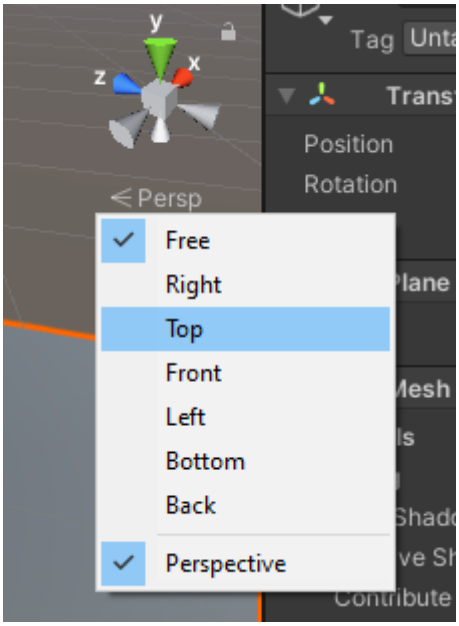
Your game needs more stars.

Select your Star in the Scene view and duplicate it with `Ctrl+D` (or `Cmd+D`). The Particle System is a child object so this will be duplicated in your new star:



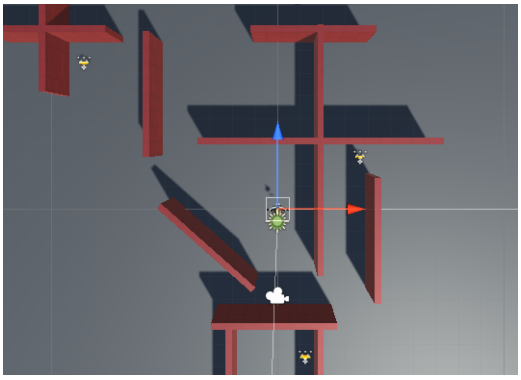
The new star will appear in the same position, so drag it to a new hiding position in the scene. The child Particle System will move with the star.

To see your map in a top-down view, right-click where it says **Persp** in the top right of the Scene view and choose **Top**. To return to the normal view, right-click on **Top** and choose **Free**:



You can use the arrow keys to move left and right and zoom. Hold the right mouse button down and drag to move and rotate.

Repeat this so you have three stars hidden on your map:

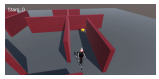


Test: Play your scene and collect all the stars make sure they all disappear and play a sound when collected.

Save your project

Step 4 Counting stars

Games often show status information such as a score. You will show the number of stars collected so far.



A Unity GameObject can have multiple scripts. You will add a new script to the Player to store and display the numbers of stars they have.

The player needs to keep track of how many stars they have collected, you can do this with a variable.

Select the **Player** and in the Inspector click **Add Component** and create a new script called `StarPlayer`. Open your new script in the code editor and create a new variable called `stars`:

StarPlayer.cs

```
5  public class StarPlayer : MonoBehaviour
6  {
7      public int stars = 0; // An integer whole number
8      // Start is called before the first frame update
9      void Start()
10     {
```

Save your script and return to the Unity Editor.

The `StarController` script needs to update the `stars` variable on the Player each time a star is collected.

Open your `StarController` script and add code to increase the number of stars the player has by 1 every time a star is collected.

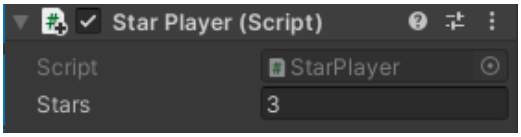
The `other` parameter of the `OnTriggerEnter` method is set to the GameObject that has collided with the Star. You can use it to access the `stars` variable from `StarPlayer`:

StarController.cs - OnTriggerEnter(Collider other)

```
21 void OnTriggerEnter(Collider other)
22 {
23     // Check the tag of the colliding object
24     if (other.CompareTag("Player"))
25     {
26         StarPlayer player = other.gameObject.GetComponent<StarPlayer>();
27         player.stars += 1; // Increase by 1
28         AudioSource.PlayClipAtPoint(collectSound, transform.position);
29         gameObject.SetActive(false);
30     }
31 }
```

Save your script and return to the Unity Editor.

Test: Run your scene and collect the three stars. Watch the public `stars` variable in the Player's Inspector window to check that the number increases by 1 every time you collect a star:



Debug: If you don't see the `stars` variable in the Inspector, make sure you have saved your `StarPlayer.cs` script.

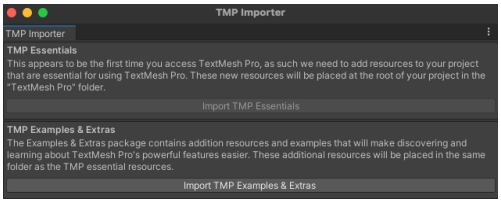
Being able to see how many stars have been collected is great for your testing but users will not be able to see that.

> UI or User Interface elements allow a Unity project to use objects including text, buttons, or sliders to communicate and interact with the user or player. UI elements are often used for game start screens and settings and for giving information to the user and allowing the user to make choices.

Right-click in the Hierarchy window and go to **UI** then select **Text - TextMeshPro**. This creates a canvas with a child text object; you can see the text in the **Game view**:

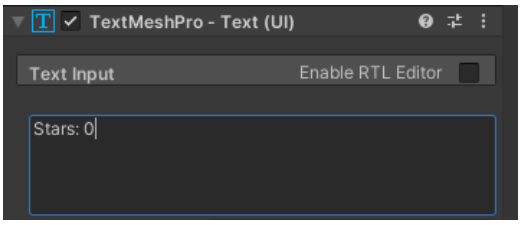
First time using TextMeshPro message

You might see a pop-up window asking you to import TextMeshPro essentials, examples, and extras to your project. If this is the case, click on the two **Import** buttons in turn, then close the window:



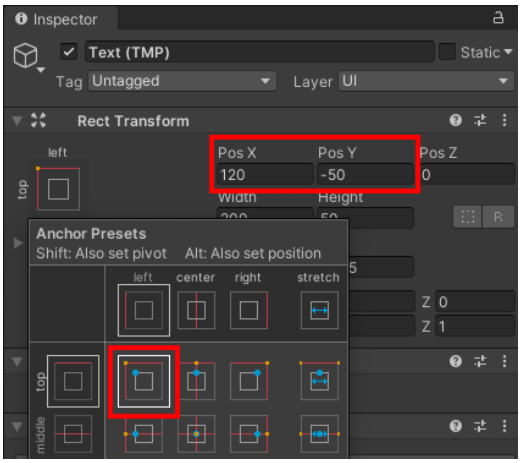
Right-click on the new **Text - (TMP) GameObject** and select `rename`. Call it `Stars Text` to easily identify it:

In the Inspector window, for the TextMeshPro GameObject, go to the **Text Input** component. Change `New Text` to `Stars: 0`:



In the **Rect Transform** component, click and change the alignment to **Top Left**. And change the position to `x = 120`, `y = -50`.

This will position the centre of your text 120 pixels from the left and -50 pixels from the top. The text will stay in position if you resize the Game view:



Tip: You can view the position of the text in the Game view even when you are not in Play mode.

Now you need to update the text that is displayed so that it shows the current number of stars collected by the player.

Open your `StarPlayer` script and add `using TMPro` at the top so that your script can use `TMP_Text`:

StarPlayer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
```

Add code to create a TMP_Text Object called `starText`:

StarPlayer.cs

```
public class StarPlayer : MonoBehaviour
{
    public int stars = 0; // An integer whole number
    public TMP_Text starText;
    // Start is called before the first frame update
```

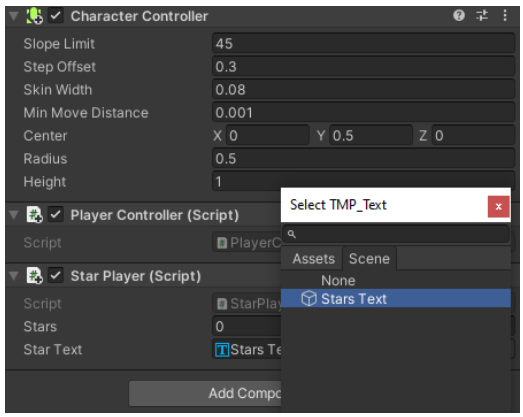
Use the `SetText` method from the `TMP_Text` class to display the number of stars collected on each update:

StarPlayer.cs - Update()

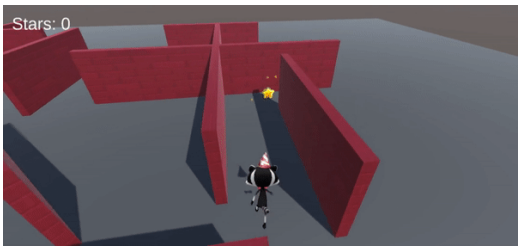
```
public class StarPlayer : MonoBehaviour
{
    // Update is called once per frame
    void Update()
    {
        starText.SetText("Stars: " + stars);
    }
```

Save your code and switch back to the Unity Editor.

In the Player's Inspector window for the `StarPlayer` script, click on the circle next to the Star Text property and choose **Stars Text** to link your text object.



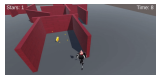
Play your scene and check that the number in the UI text increases each time you collect a star:



Save your project

Step 5 Keeping time

Now that a player can collect stars, add a timer to show the amount of time taken to collect all three stars.

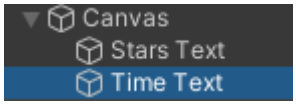


> **Game mechanics** are a key part of game design. They are the rules that control a player's actions. A **timer** is a game mechanic that adds a challenge to video games — in fact, there are many Guinness World Records based on how quickly players can complete challenges in games!

The player needs to keep track of how long they are taking to complete the minigame, you can do this with another variable.

In the Hierarchy window, right-click on your **Canvas** and from UI create another **Text - TextMeshPro GameObject**. You will see 'New text' written on your screen in Game view:

Right-click on the new **Text (TMP) GameObject** and select **rename**. Call it `Time Text` to easily identify it:



From the Inspector window, in the Text Input propery for the new TextMeshPro GameObject, change `New Text` to `Time: 0`.

Use the **Rect Transform** component to change the alignment to **Top Right**. Also change the position to `x = -60`, `y = -50`:

The text that is displayed needs to update so that it continuously shows the number of seconds since the game started.

Open your `StarPlayer` script and add code to create a TMP_Text Object called `timeText`:

StarPlayer.cs

```
6  public class StarPlayer : MonoBehaviour
7  {
8      public int stars = 0; // An integer whole number
9      public TMP_Text starText;
10     public TMP_Text timeText;
```

`Time.time` gives the time in seconds since the Scene started. `Mathf.Round` turns a number into a whole number.

Set the text to show the number of whole seconds on each update:

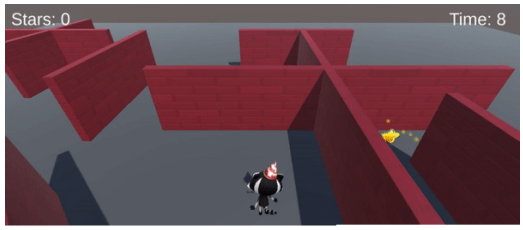
StarPlayer.cs - Update()

```
18 void Update()
19 {
20     starText.SetText("Stars: " + stars);
21     timeText.SetText("Time: " + Mathf.Round(Time.time));
22 }
```

Save your script and go back to the Unity Editor.

Select the Player in the Hierarchy window and go to the `Star Player` script component in the Inspector window. Click on the circle next to `Time Text` and choose your new 'Time Text' object.

Test: Run your minigame and check that the time updates as you play. What happens when you collect all three stars?



The time needs to stop when all three stars are collected, but currently it will keep counting up for as long as the minigame is playing.

Open the `StarPlayer` script and create an if statement around your time code to only count the seconds if the player has collected less than three stars:

StarPlayer.cs - Update()

```
18  void Update()
19  {
20      starText.SetText("Stars: " + stars);
21      if (stars < 3)
22      {
23          timeText.SetText("Time: " + Mathf.Round(Time.time));
24      }
25  }
```

Save your script and go back to the Unity Editor.

Test: Run your minigame again. The timer will stop when the player has three stars:

After the Reflection step, you can upgrade your project with the features you think are important.

Save your project

Upgrade your project

Customize your game so that it plays just the way you want it to.

Spend some time playing your game. What changes will make the user experience better for this minigame?

You could:

- Add more stars to collect and more walls to hide the stars behind
- Try changing the colour, font, and size of the text in the TextMeshPro GameObject in the Inspector window
- Adjust the settings on the Particle System to get the effect you want
- Add other GameObjects to collect; there are hearts and four-leafed clovers in the Models folder
- Make the game more difficult over time; make the Player go slower or lose some collected stars
- Store the time taken in a variable and give time penalties for collecting different objects
- Change the camera angle

Completed project

You can download the **completed project here** (https://rpf.io/p/en/star-collector-get).

Save your project

What next?

If you are following the **Introduction to Unity** (https://projects.raspberrypi.org/en/raspberrypi/unity-intro) path, you can move on to the **Non-player characters** (https://projects.raspberrypi.org/en/projects/non-player-characters) project. In this project, you will add non-player characters (NPCs) to your game to help you or make things more difficult.


Non-player characters
