

# Code Coverage - Worksheet



## Tasks

1. **What is Code Coverage** (2 min)
2. **Install the Code Coverage package** (2 min)
3. **Enable Code Coverage** (1 min)
4. **Understanding the game code: Shoot() function** (4 min)
5. **Generate a Coverage report from PlayMode tests** (3 min)
6. **Add Weapon tests to improve coverage** (3 min)
7. **Add a test for the LaserController** (4 min)
8. **Clear the coverage data** (1 min)
9. **Generate a Coverage report using Coverage Recording** (4 min)

## Useful Links

Code Coverage package documentation:

[docs.unity3d.com/Packages/com.unity.testtools.codecoverage@latest](https://docs.unity3d.com/Packages/com.unity.testtools.codecoverage@latest)

Unity Forum thread:

[forum.unity.com/threads/code-coverage-package.777542](https://forum.unity.com/threads/code-coverage-package.777542)



## 1. What is Code Coverage (2 min)

[Code Coverage](#) is a measure of how much of your code has been executed. It is normally associated with automated tests, but you can gather coverage data in Unity at any time when the Editor is running.

It is typically presented as a [report](#) that shows the percentage of the code that has been executed. For automated testing the report does not measure the quality of tests, only whether your code is executed by PlayMode and EditMode tests. It is especially useful to check that critical or high risk areas of your code are covered, because they should receive the most rigorous testing.

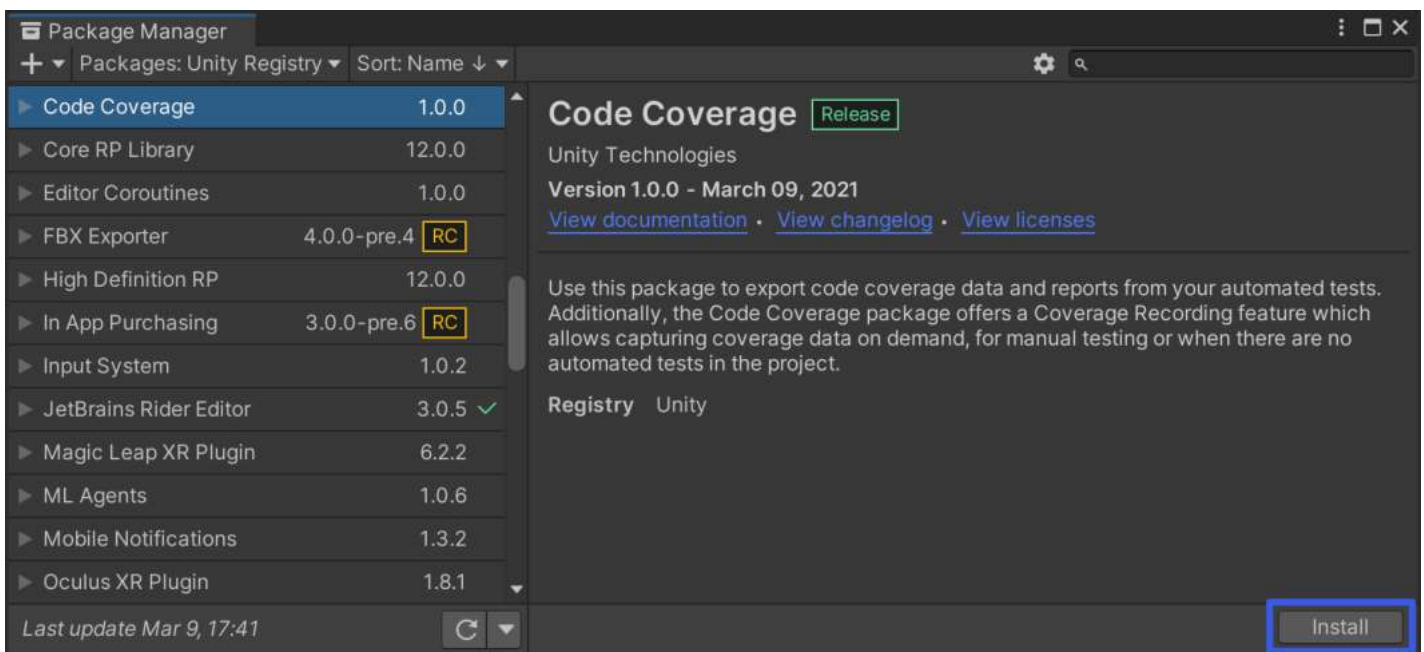
It is much easier to accidentally introduce bugs into code that is not covered by tests, because those bugs are not detected straight away by the tests and can instead cause problems later — such as after you have published your game or app.

Additionally, the Code Coverage package offers a [Coverage Recording](#) feature which allows capturing coverage data on demand, in case you do not have tests in your project or doing manual testing.

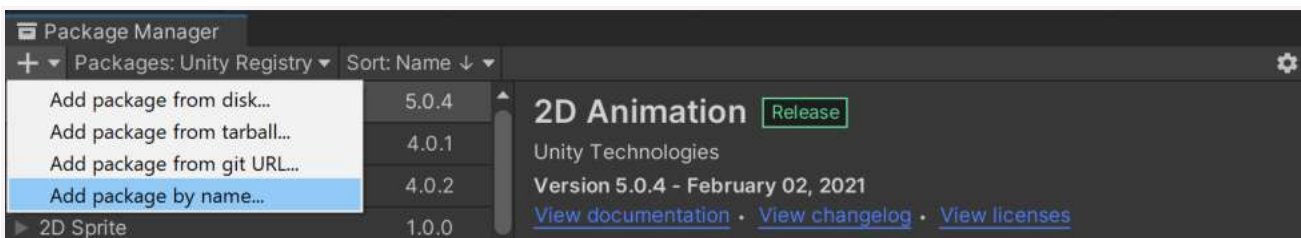
## 2. Install the Code Coverage package (2 min)

*Skip this task if the package is already installed*

Use the [Unity Package Manager](#) to find and install the **Code Coverage** package.



Alternatively, use the Add(+) dropdown and select **Add package from git URL...** or **Add package by name...** and type `com.unity.testtools.codecoverage`



To verify that Code Coverage has been installed correctly, open the Code Coverage window (go to **Window > Analysis > Code Coverage**). If you don't see the **Code Coverage** menu item, then Code Coverage did not install correctly.

### 3. Enable Code Coverage (1 min)

To enable Code Coverage open the **Code Coverage window** (go to **Window > Analysis > Code Coverage**) and select **Enable Code Coverage** if not already selected, to be able to generate Coverage data and reports.

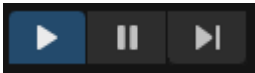


**Note:** Enabling Code Coverage adds some overhead to the editor and can affect the performance.

### 4. Understanding the game code: Shoot() function (4 min)

1. Go to `Asteroids/Scenes` in Project View and open the **Asteroids** scene.  
This is located in `Assets/Samples/Code Coverage/<version>/Code Coverage Tutorial`.

2. Hit **Play** and play the game for a minute or two.



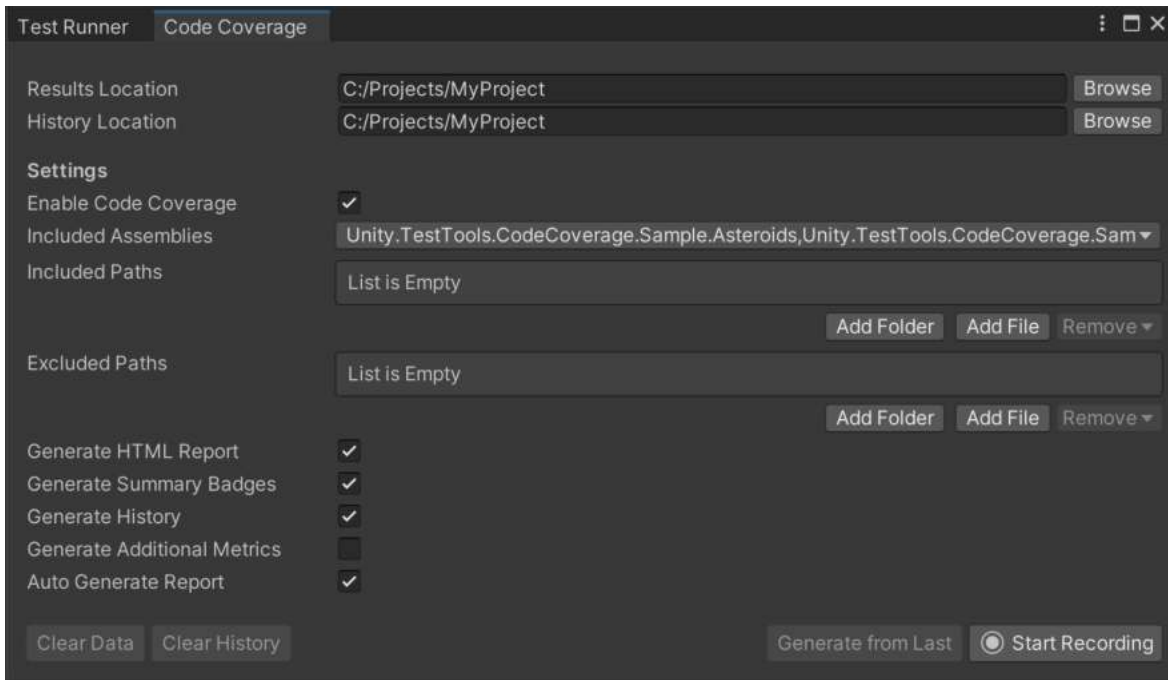
Use the arrow keys to move and spacebar to shoot.

3. Exit PlayMode.
4. Open the **Scripts/Controllers/SpaceshipController.cs** script.
5. Study the **Shoot** function.

If Weapon is Basic, the Prefabs/Weapons/Projectile prefab is instantiated  
If Weapon is Laser, the Prefabs/Weapons/Laser prefab is instantiated

## 5. Generate a Coverage report from PlayMode tests (3 min)

1. Open the **Code Coverage** window (go to **Window > Analysis > Code Coverage**).

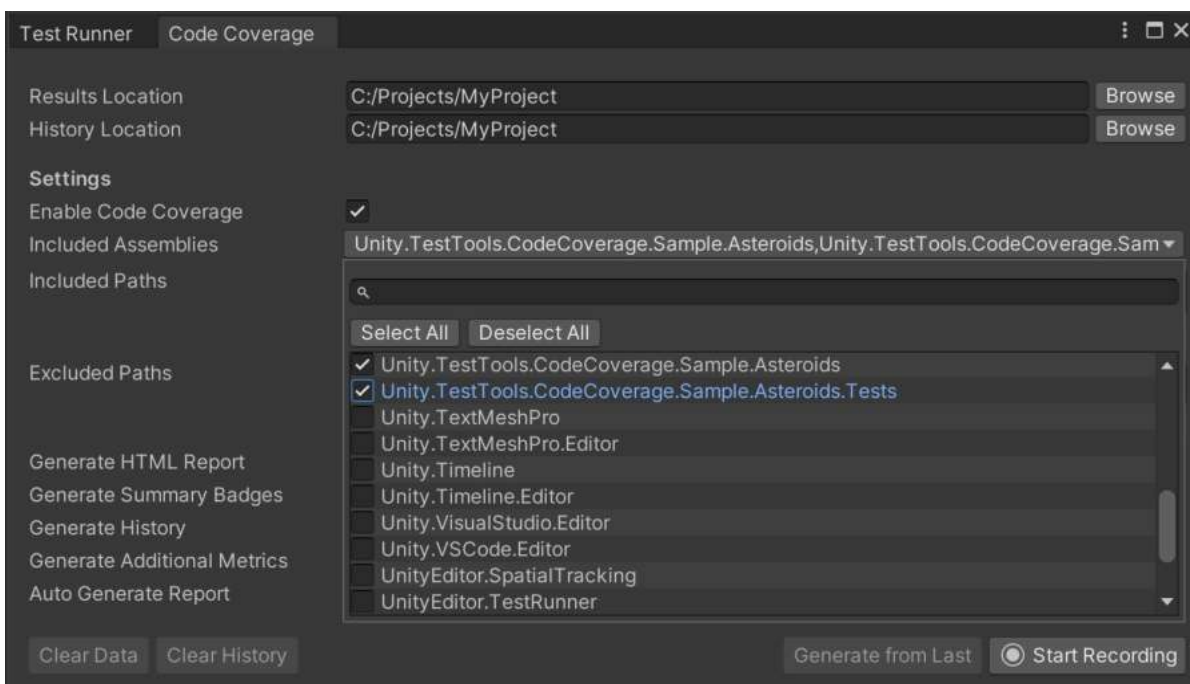


2. If you see this warning select **Switch to debug mode**.



[Code Optimization](#) was introduced in Unity 2020.1; in *Release mode* the code is optimized and therefore not directly represented by the original code. Therefore, *Debug mode* is required in order to obtain accurate code coverage information.

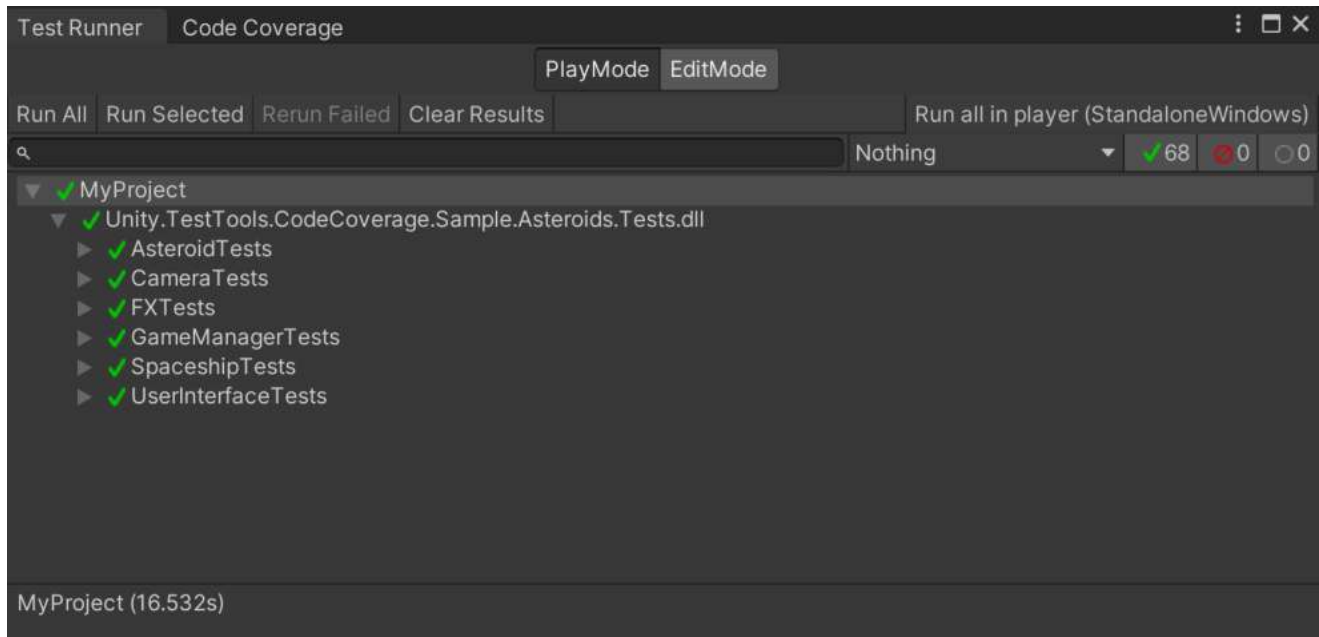
3. Click the **Included Assemblies** dropdown to make sure only `Unity.TestTools.CodeCoverage.Sample.Asteroids` and `Unity.TestTools.CodeCoverage.Sample.Asteroids.Tests` are selected.



4. Make sure **Generate HTML Report**, **Generate History** and **Auto Generate Report** are all checked.



5. Switch to the **Test Runner**, select the **PlayMode** tab and hit **Run All** tests.



6. When the tests finish running, a file viewer window will open up containing the coverage report. Select **index.htm**.

7. Look for the classes with low coverage, especially **LaserController**, **BaseProjectile** and **ProjectileController**.


You can sort the results by *Line coverage*.

Coverage									
Collapse all   Expand all									
By assembly									
Filter:									
Name	Covered	Uncovered	Coverable	Total	Line coverage	Covered	Total	Branch coverage	
Unity.TestTools.CodeCoverage.Sample.Asteroids	371	61	432	701	85.8%	0	0		
LaserController	2	21	23	36	8.6%	0	0		
BaseProjectile	1	10	11	20	9%	0	0		
ProjectileController	13	4	17	30	76.4%	0	0		
InGameMenuController	13	2	15	29	86.6%	0	0		
SpaceshipController	81	12	93	143	87%	0	0		
LifeCounter	39	3	42	63	92.8%	0	0		
GameManager	92	6	98	161	93.8%	0	0		
ScoreCounter	51	3	54	83	94.4%	0	0		
AnimatorDisabler	4	0	4	11	100%	0	0		
AsteroidController	39	0	39	62	100%	0	0		
DebrisController	19	0	19	31	100%	0	0		
EngineTrail	17	0	17	32	100%	0	0		
+									
Unity.TestTools.CodeCoverage.Sample.Asteroids.Tests	714	15	729	1439	97.9%	0	0		

See also [How to interpret the results](#).

## 6. Add Weapon tests to improve coverage (3 min)

1. Open the `Tests/WeaponTests.cs` script.
2. Uncomment all the tests (from *line 35* down to *line 237*).
3. Back in the **Test Runner**, hit **Run All** tests again.
4. When the tests finish running, a file viewer window will open up containing the coverage report. Select **index.htm**.
5. Notice that now **BaseProjectile** and **ProjectileController** coverage is considerably higher, but **LaserController** has not improved much.

Coverage									
Collapse all   Expand all									
By assembly									
Grouping 									
Compare with: <span>Date</span>									
Filter: <input type="text"/>									
Name	Covered	Uncovered	Coverable	Total	Line coverage	Covered	Total	Branch coverage	
Unity.TestTools.CodeCoverage.Sample.Asteroids	385	47	432	701	89.1%	0	0		
LaserController	2	21	23	36	8.6%	0	0		
InGameMenuController	13	2	15	29	86.6%	0	0		
SpaceshipController	81	12	93	143	87%	0	0		
LifeCounter	39	3	42	63	92.8%	0	0		
GameManager	92	6	98	161	93.8%	0	0		
ScoreCounter	51	3	54	83	94.4%	0	0		
AnimatorDisabler	4	0	4	11	100%	0	0		
AsteroidController	39	0	39	62	100%	0	0		
BaseProjectile	11	0	11	20	100%	0	0		
DebrisController	19	0	19	31	100%	0	0		
EngineTrail	17	0	17	32	100%	0	0		
ProjectileController	17	0	17	30	100%	0	0		
Unity.TestTools.CodeCoverage.Sample.Asteroids.Tests	849	0	849	1437	100%	0	0		

## 7. Add a test for the LaserController (4 min)

1. Open the `Tests/WeaponTests.cs` script.
2. Go to the **\_18\_LaserFiresSuccessfully** test in *line 225*.
3. Uncomment and study the code.
4. Back in the **Test Runner**, hit **Run All** tests again.
5. When the tests finish running, a file viewer window will open up containing the coverage report. Select **index.htm**.
6. Notice how the coverage for **LaserController** has improved.

LaserController	15	8	23	36	65.2%	
-----------------	----	---	----	----	-------	---

7. Select the **LaserController** class to enter the class view and notice that about 2/3 (65%) of the code is now covered (green).

```
# Line Line coverage
1 1 using UnityEngine;
2
3 public class LaserController : MonoBehaviour
4 {
5     public bool isActive = true;
6     public float duration = 0.75f;
7
8     private void Update()
9     {
10         if (!GameManager.IsPaused)
11         {
12             if (isActive)
13                 Expand();
14             else
15                 Shrink();
16
17         duration += Time.deltaTime;
18         if (duration < 0.0f)
19             isActive = false;
20         }
21     }
22
23     private void Expand()
24     {
25         if (transform.localScale.y < 75.0f)
26             transform.localScale += Vector3.up * Time.deltaTime * 75.0f;
27     }
28
29     private void Shrink()
30     {
31         transform.localScale -= Vector3.up * Time.deltaTime * 75.0f;
32         transform.position += transform.up * Time.deltaTime * 75.0f;
33         if (transform.localScale.y < 0.0f)
34             Destroy(gameObject);
35     }
36 }
```

Complete the **Bonus Task** at the end of the tutorial to get 100% coverage!

## 8. Clear the coverage data (1 min)

1. Open the **Code Coverage** window (go to **Window > Analysis > Code Coverage**).
2. Select **Clear Data** and confirm.
3. Select **Clear History** and confirm.



## 9. Generate a Coverage report using Coverage Recording (4 min)

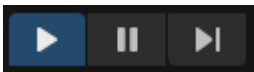
1. Go to `Asteroids/Scenes` in Project View and open the **Asteroids** scene, if not opened already.
2. Open the **Code Coverage** window. Make sure **Generate HTML Report**, **Generate History** and **Auto Generate Report** are all checked.



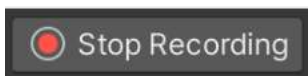
3. Select **Start Recording**.



4. Hit **Play** to play the game and **exit** PlayMode before you get **8000** points.



5. Select **Stop Recording**.



6. A file viewer window will open up containing the coverage report. Select **index.htm**.

7. Notice that **LaserController** has 0% coverage.

LaserController	0	23	23	36	0%	<div></div>
-----------------	---	----	----	----	----	-------------

8. Go back to the **Code Coverage** window.

9. Select **Start Recording**.

10. Now hit **Play** to play the game again but this time **exit** PlayMode when you get **8000** points.

11. Select **Stop Recording**.

12. Notice that **LaserController** coverage is now 100%.

LaserController	23	0	23	36	100%	<div></div>
-----------------	----	---	----	----	------	-------------

See also [How to interpret the results](#).



## 10. Bonus Task (5-8 min)

Write a new test that checks that the laser gets destroyed after 2 seconds, which will also cover the rest of the code in **LaserController**.

**Suggested name:** `_19_LaserFiresAndIsDestroyedAfterTwoSeconds`.

**Hint:** You can use `yield return new WaitForSeconds(2f);` to wait for 2 seconds.

---

**Well done for finishing the Code Coverage Tutorial!**

For questions and feedback please reach out to us in the dedicated Code Coverage package [forum thread](#).