Code Coverage - Worksheet

Tasks

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

Useful Links

Code Coverage package documentation:
docs.unity3d.com/Packages/com.unity.testtools.codecoverage@latest

Unity Forum thread:
forum.unity.com/threads/code-coverage-package-preview.777542

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.

Install the Code Coverage package (2 min)

Skip this task if the package is already installed

From the Unity Package Manager

Use the Unity Package Manager to find and install the Code Coverage package.

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.

Manually from the Package Manifest

You can also install the Code Coverage package manually. To do this, add a reference to Code Coverage in your project's **Packages/manifest.json** file. There are two ways you can reference a specific version of the Code Coverage package, depending on how you use it.

Using a production version of the package

You can point the Package Manager at a publicly available version. To do this manually, add it to **manifest.json**:

```
"dependencies": {
   //...
   "com.unity.testtools.codecoverage":"<full version number>"
}
```

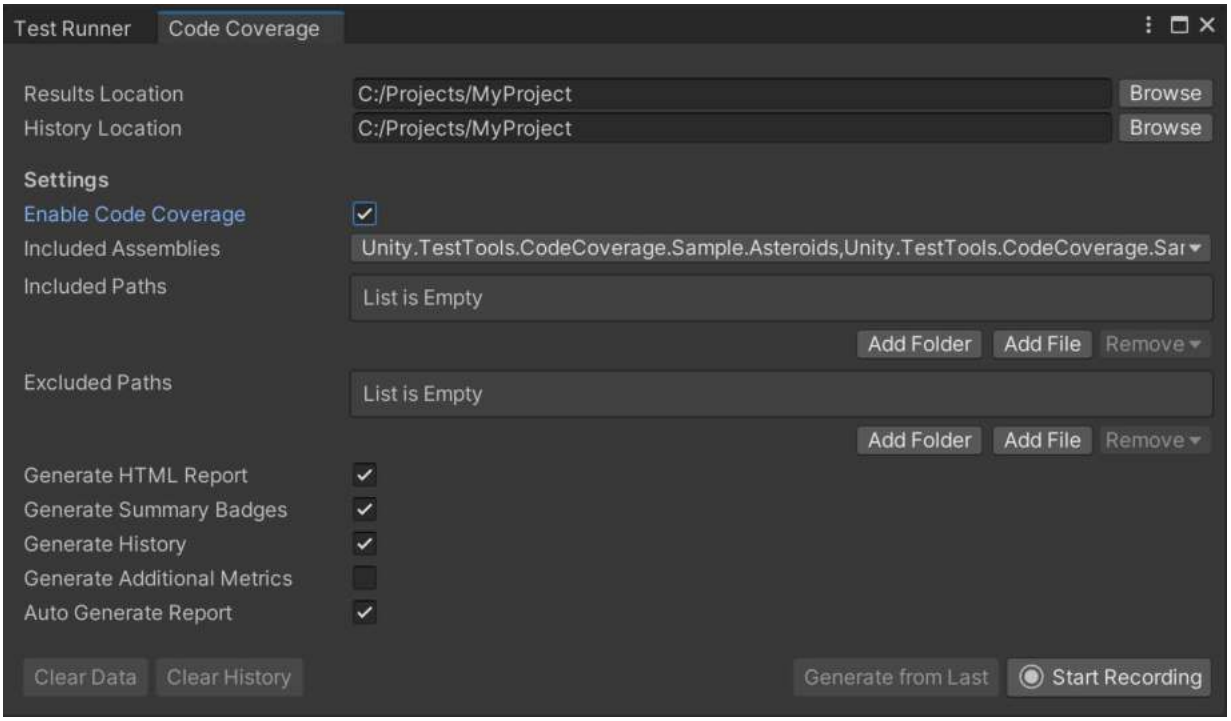
Using a local clone of the package

If you want to use a cloned version of the package, you can point the Package Manager at a local folder as the package location:

```
"dependencies": {
   //...
   "com.unity.testtools.codecoverage":"file:path/to/package/root"
}
```

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 that enabling Code Coverage adds some overhead to the editor and can affect the performance.

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 Workshop*

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

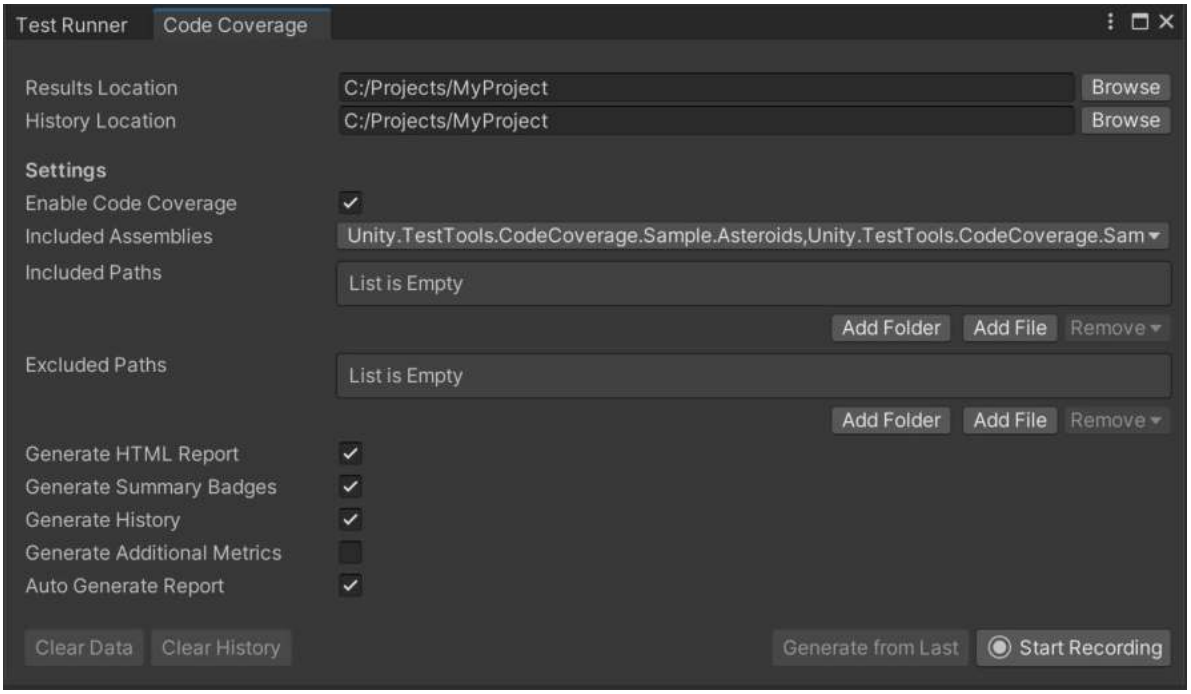
4. Open **Scripts/Controllers/SpaceshipController.cs** script

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
   ```

Generate a Coverage report from the PlayMode tests (3 min)

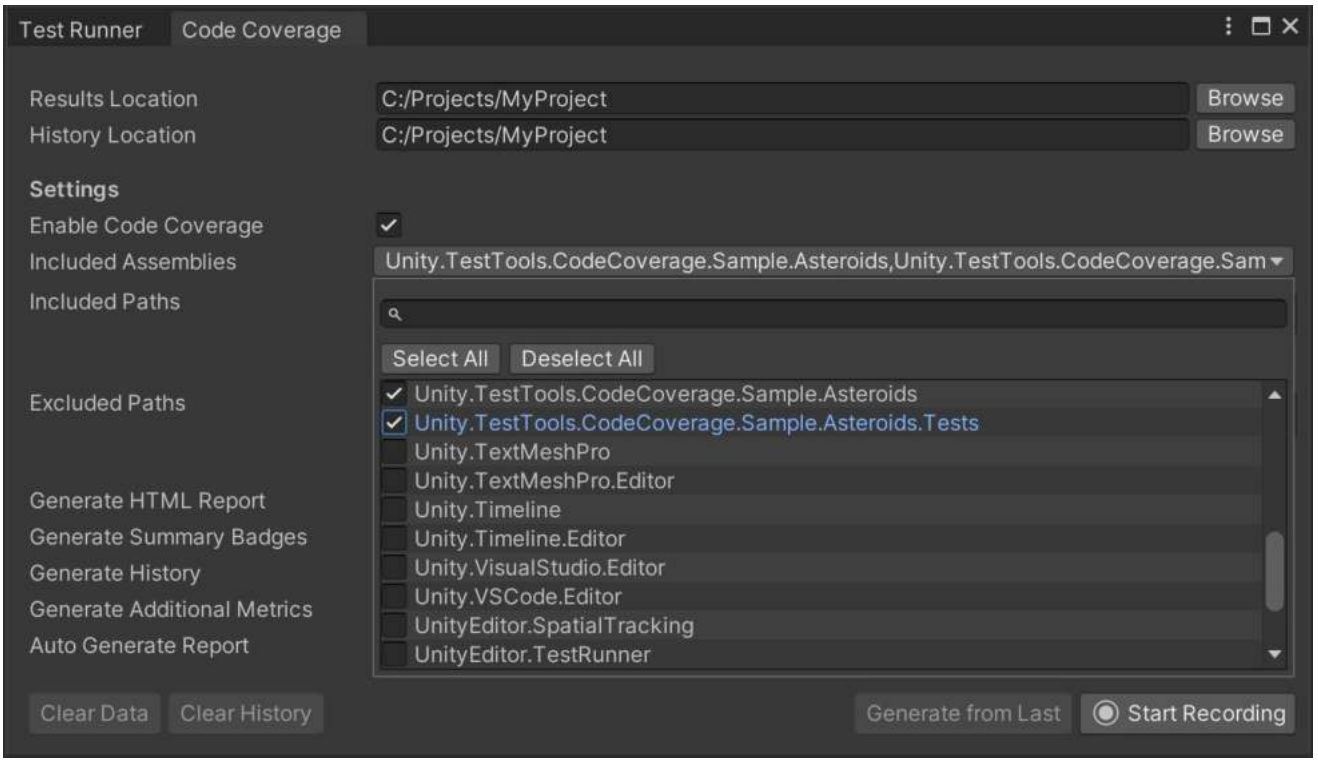
1. Open the **Code Coverage window** (go to **Window** > **Analysis** > **Code Coverage**)



2. ***Skip this step on Unity versions prior to 2020.1***.
   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.

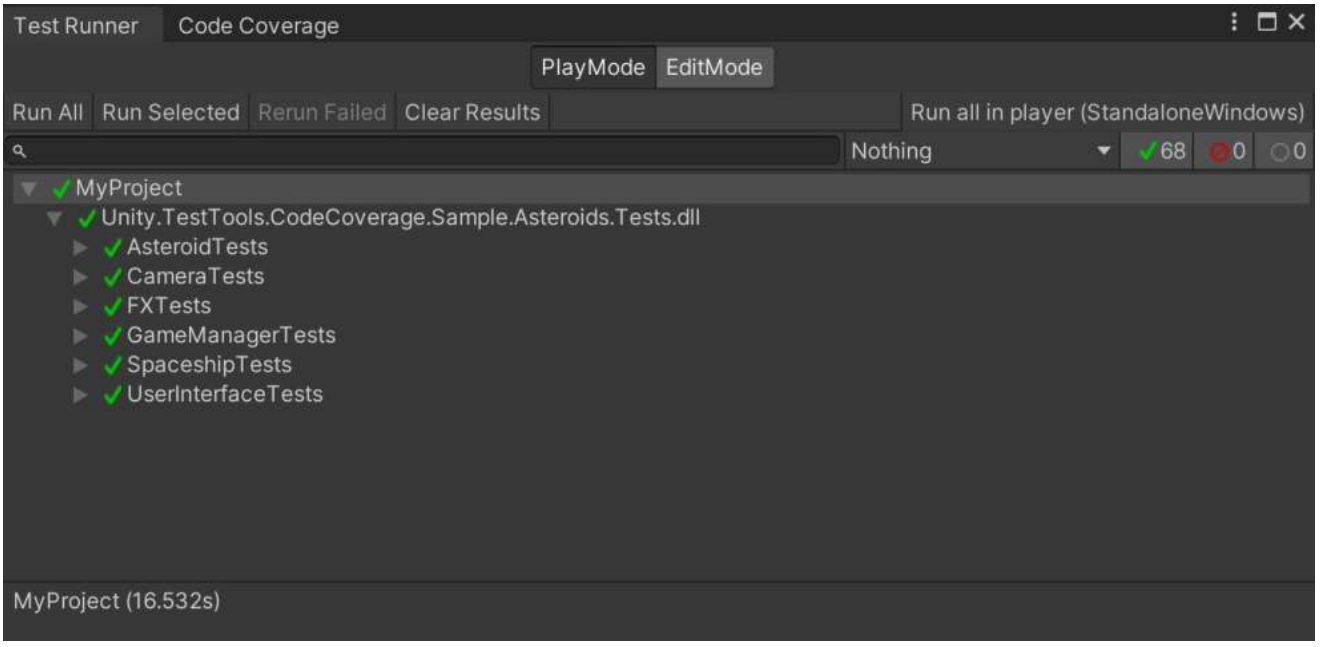
   If you see this warning select **Switch to debug mode**

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 checked

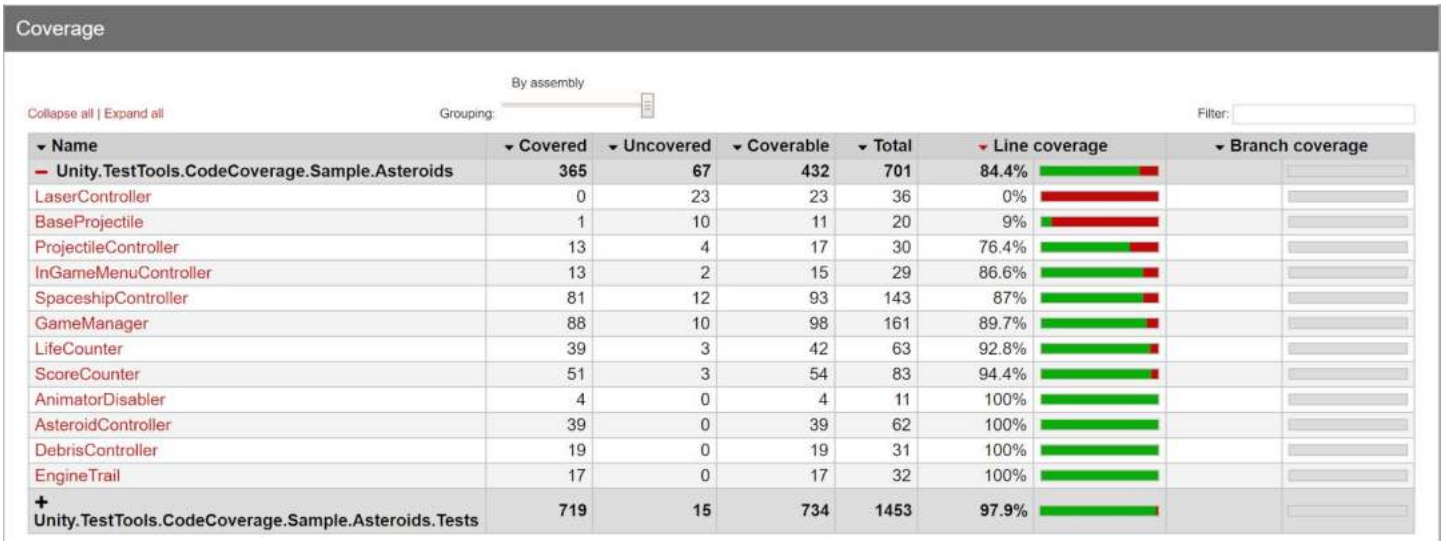
5. Switch to the **Test Runner**, select the **PlayMode** tab and hit **Run All** tests



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**

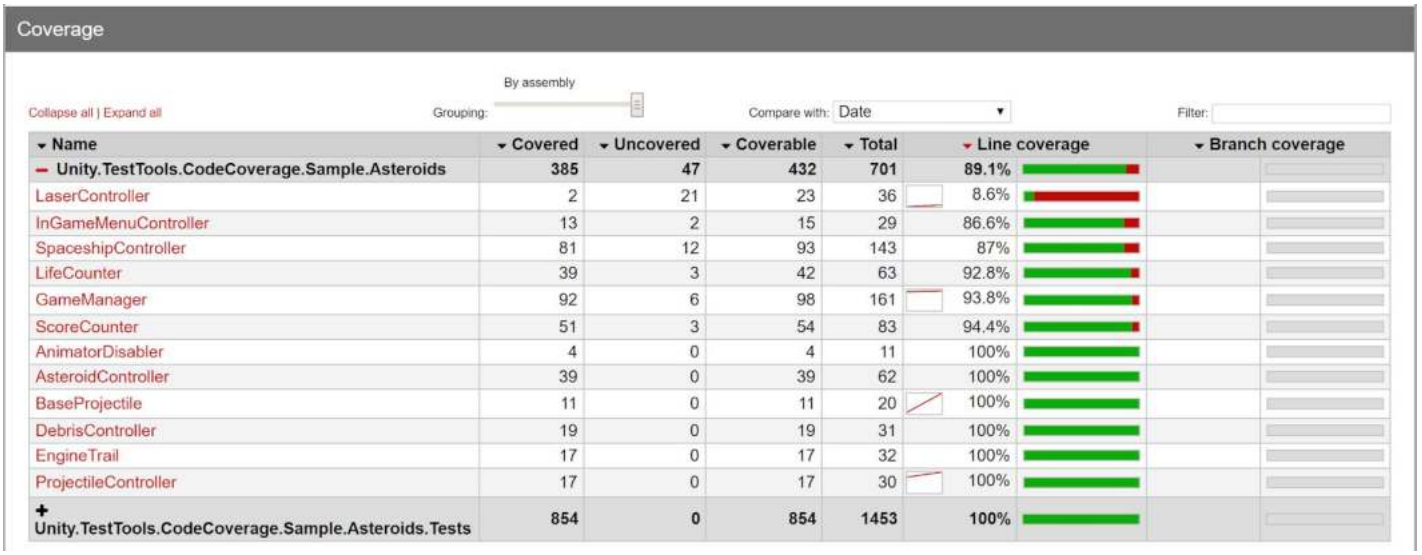
You can sort the results by *Line coverage*



Name	Covered	Uncovered	Coverable	Total	Line coverage	Branch coverage
− Unity.TestTools.CodeCoverage.Sample.Asteroids	365	67	432	701	84.4%	
LaserController	0	23	23	36	0%	
BaseProjectile	1	10	11	20	9%	
ProjectileController	13	4	17	30	76.4%	
InGameMenuController	13	2	15	29	86.6%	
SpaceshipController	81	12	93	143	87%	
GameManager	88	10	98	161	89.7%	
LifeCounter	39	3	42	63	92.8%	
ScoreCounter	51	3	54	83	94.4%	
AnimatorDisabler	4	0	4	11	100%	
AsteroidController	39	0	39	62	100%	
DebrisController	19	0	19	31	100%	
EngineTrail	17	0	17	32	100%	
+ Unity.TestTools.CodeCoverage.Sample.Asteroids.Tests	719	15	734	1453	97.9%	

Add Weapon tests to improve coverage (3 min)

1. Open **Tests/WeaponTests.cs** script
2. Uncomment all the tests (from *line 35* up 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



Name	Covered	Uncovered	Coverable	Total	Line coverage	Branch coverage
− Unity.TestTools.CodeCoverage.Sample.Asteroids	385	47	432	701	89.1%	
LaserController	2	21	23	36	8.6%	
InGameMenuController	13	2	15	29	86.6%	
SpaceshipController	81	12	93	143	87%	
LifeCounter	39	3	42	63	92.8%	
GameManager	92	6	98	161	93.8%	
ScoreCounter	51	3	54	83	94.4%	
AnimatorDisabler	4	0	4	11	100%	
AsteroidController	39	0	39	62	100%	
BaseProjectile	11	0	11	20	100%	
DebrisController	19	0	19	31	100%	
EngineTrail	17	0	17	32	100%	
ProjectileController	17	0	17	30	100%	
+ Unity.TestTools.CodeCoverage.Sample.Asteroids.Tests	854	0	854	1453	100%	

Add a test for the LaserController (4 min)

1. Open **Tests/WeaponTests.cs** script
2. Go to the **_18_LaserFiresSuccessfully** test on *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 see that most of the code is now covered (green).

   Complete the **Bonus Task** at the end of the worksheet to achieve 100% coverage!

#	Line	Line coverage
	1	`using UnityEngine;`
	2	
	3	`public class LaserController : BaseProjectile`
	4	`{`
2	5	`    public bool isActive = true;`
2	6	`    public float duration = 0.75f;`
	7	
	8	`    private void Update()`
1	9	`    {`
1	10	`        if (!GameManager.IsPaused)`
1	11	`        {`
1	12	`            if (isActive)`
1	13	`                Expand();`
	14	`            else`
0	15	`                Shrink();`
	16	
1	17	`            duration -= Time.deltaTime;`
1	18	`            if (duration <= 0.0f)`
0	19	`                isActive = false;`
1	20	`        }`
1	21	`    }`
	22	
	23	`    private void Expand()`
1	24	`    {`
1	25	`        if (transform.localScale.y <= 25.0f)`
1	26	`            transform.localScale += Vector3.up * Time.deltaTime * 75.0f;`
1	27	`    }`
	28	
	29	`    private void Shrink()`
0	30	`    {`
0	31	`        transform.localScale -= Vector3.up * Time.deltaTime * 75.0f;`
0	32	`        transform.position += transform.up * Time.deltaTime * 75.0f;`
0	33	`        if (transform.localScale.y <= 0.0f)`
0	34	`            Destroy(gameObject);`
0	35	`    }`
	36	`}`

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

Generate a Coverage report using Coverage Recording (4 min)

1. Open the **Code Coverage** window. Make sure **Generate HTML Report**, **Generate History** and **Auto Generate Report** are checked
2. Select **Start Recording**
3. Hit **Play** to play the game and **Exit** PlayMode before you get **8000** points

4. Select **Stop Recording**
5. A file viewer window will open up containing the coverage report. Select **index.htm**
6. Notice that **LaserController** has 0% coverage

7. Go back to the **Code Coverage** window
8. Select **Start Recording**
9. Now hit **Play** to play the game again but this time **Exit** PlayMode when you get **8000** points
10. Select **Stop Recording**
11. Notice that **LaserController** coverage is now 100%

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 workshop!

For questions and feedback please reach out to us in the forum thread