Unity Test Framework Guide

By Karina Permann, Gary Banks, Jordan Higgins

Table of Contents

Background

This section introduces the Unity Test Framework (UTF) and the mode options for testing in it.

Unity Test Framework

As a Unity package, the UTF serves as a standard test framework with two testing mode options: Play mode and Edit mode. This package facilitates structured test automation in Unity. The UTF integrates NUnit library, an open-source unit testing library for .Net languages. For further information, refer to https://docs.unity3d.com/Packages/com.unity.test-framework@1.3/manual/. With the UTF, unit tests, including unit boundary tests, may be developed.

Play Mode vs. Edit Mode Tests

The two test modes, Play mode and Edit mode, offered for the UTF are designed to accommodate different testing requirements. **Edit mode** tests are best suited for automated tests, independent of user input. These tests exclusively run in the Unity Editor and have access to both Editor and game code.

Play mode tests, conversely, are designed for input or game state dependent tests. For example, object movement and animations are generally tested through Play mode tests. These tests run as coroutines.

Unity Test Framework Installation

This section introduces the Unity Test Framework (UTF) and the mode options for testing in it.

Prerequisites

Prior to installation of the UTF, a version of 2019.2 or later of the Unity Editor must be installed.

Walkthrough

1. Open a project in the Unity Editor version 2019.2 or after.
2. Select **Window** > **Package Manager** to open the package manager where packages may be installed or upgraded.

3. Select **Packages: Unity Registry** from the dropdown at the top of the **Package Manager** window. A list of available packages should display.
4. Locate **Test Framework** by either scrolling or entering the term in the search field.
5. Assuming this package is not already installed, select **Install** at the bottom right of the window.

Boundary Test Creation

Definition.

Unit tests are meant to test the functioning of functions by providing hard coded inputs and comparing the function outputs to hardcoded, expected outputs. As functions are meant to handle acceptable inputs, borderline acceptable inputs, and unacceptable inputs with defined behaviour, all such inputs must be used.

To create a unit test, assembly for the test scripts and the test functions of such must be defined.

Prerequisites

The following instructions assume that a project is already open in the Unity Editor.

Walkthrough

6. Select **Window** > **General** > **Test Runner** to open the **Test Runner** from which tests may be created and controlled.
7. Select **EditMode** as unit tests are automated.
8. Select **Create EditMode Test Assembly Folder**.
9. Select **Create Test Script in current folder**.
10. Exit or deselect the **Test Runner** window.
11. Rename the created folder (e.g., EditMode_Tests) and test script. All Edit mode tests need to be kept separate from Play mode tests. Remember to follow standard naming conventions and use intuitive names.

Stress Test Creation

Definition

A stress test is a test that puts the application to its limits in order to see what the engine or hardware can support. A number of kinds of tests can be done that fall under the category of stress tests. Creating a bunch of items in a scene and seeing when the FPS is affected is a form of a stress test. Another form of a stress test would be seeing how performance is impacted when many inputs are given rapidly or if an actor interacts with a number of other actors all at once.

Prerequisites

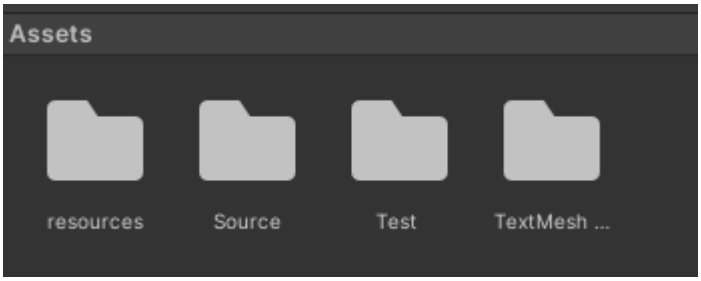
In order to start, you must know which parts of your application you are trying to test. If there is a potential for a number of enemies to be on the screen at once all trying to pathfind, this would be a good thing to test. If there is a chance that the player may pick up a massive amount of items all at once this would also be good to test. It is also important that you choose a metric for your test. A very good common metric for a stress test is FPS. Memory usage, CPU usage, and GPU usage are all also very good metrics to measure. The goal of your stress test will determine what metric you want to measure. If you are trying to see how many enemies the game can reasonably handle, FPS is a good metric. If you are trying to find recommended specifications for the game, CPU and GPU usage would be good metrics.

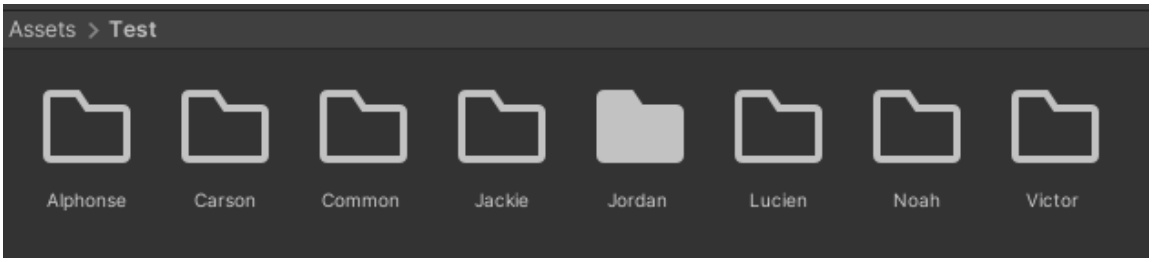
Walkthrough

For this simple example test, FPS will be the metric measured. The test will create enemies confined in a small space so that collision is forced to be calculated. The test will continue until the FPS reaches 60 for 5 frames. This test will give a good idea for how many enemies the game can handle until the FPS becomes severely impacted.

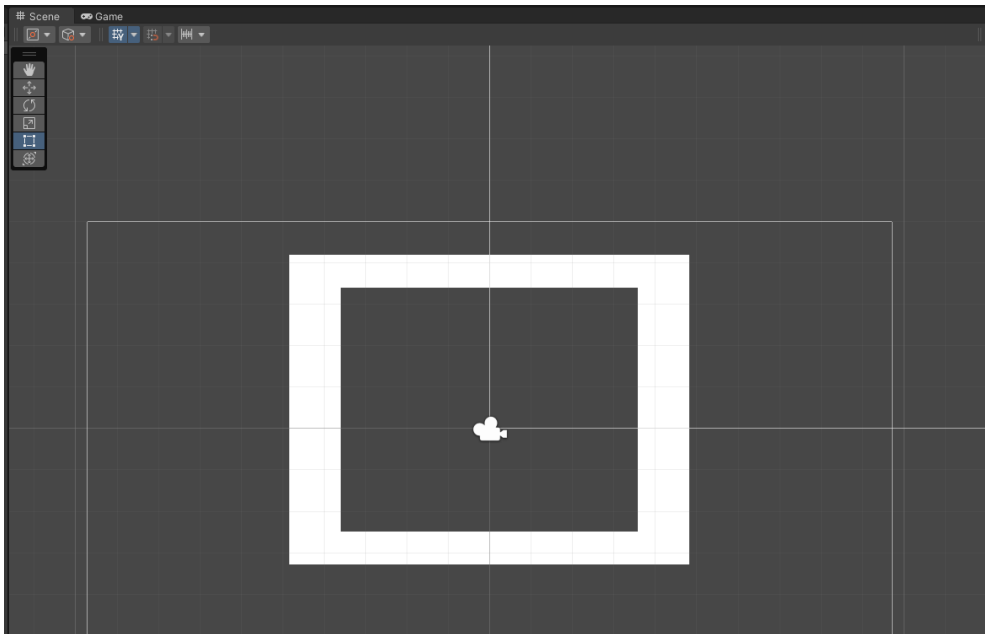
Step 1:

First, a test scene must be created to keep testing separate from production. It is assumed that the assets for the game have already been created such as enemies and the environment. The test scene and any scripts accompanied with it must be placed in their own folder. There should be a “Test” folder with multiple folders in it for each person on the team.





For the test scene, all it will contain is a simple box to spawn the enemies in. It will also need an FPS counter and an enemy counter to monitor the test.



Step 2:

Test Execution

Prerequisites

In order to run a Play mode or Edit mode test in the UTF, such a test must exist within the project.

Walkthrough

12. Open a project in the Unity Editor version 2019.2 or higher.
13. Select **Window** > **General** > **Test Runner**.
14. Select **PlayMode** or **EditMode**.
15. Select **Run All** or **Run Selected**.