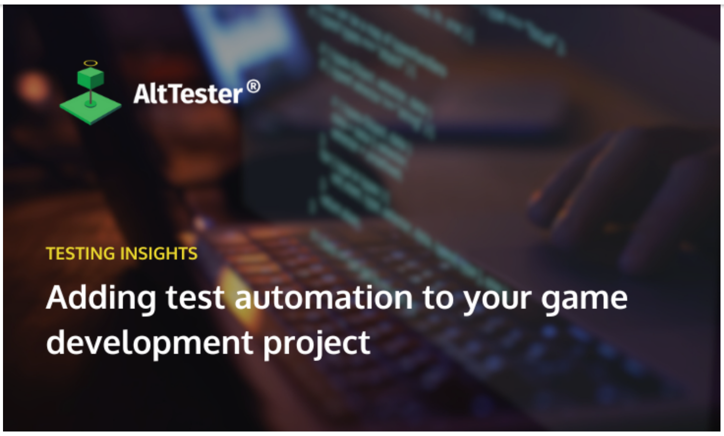


EDITORIALS TESTING INSIGHTS

Adding test automation to your game development project

by Diana Serb
Jun 10 2024

As the complexity of video games and interactive applications continues to grow, ensuring their quality and performance becomes increasingly crucial. Test automation is a powerful practice that can help teams maintain high standards while streamlining the testing process. In this article, we will explore the steps to integrate test automation into your game development project, with a special focus on using AltTester® tools to help you achieve your goals.

Assessing the Needs of Your Project

Before diving into the technical implementation, it's essential to assess the needs of your project and evaluate the available resources to ensure a smooth transition. Here are some key considerations:

1. **Project Scope and Complexity**: Determine the size and complexity of your project considering the specifics of the engine (e.g. Unity, Unreal) on which it's developed. Large projects with numerous features and frequent updates benefit more from comprehensive test automation due to the higher risk of bugs, time-consuming manual testing, and the need for consistent verification of new changes.
2. **Types of Testing Required**: Identify the types of tests that will be most beneficial for your project. Common test types include unit tests, integration tests, performance tests, and user interface (UI) tests. Each type serves a different purpose, from verifying individual components to ensuring the entire system works smoothly under various conditions.
3. **Frequency of Releases**: Consider how often you plan to release updates. Projects with frequent updates can greatly benefit from automation, as it allows for rapid regression testing to catch issues introduced by new changes.
4. **Target Platforms**: Assess the platforms your game will be released on, such as PC, consoles, or mobile devices. Different platforms may require different testing tools and strategies.

Evaluating Available Resources

Once you have a clear understanding of your project's needs, the next step is to evaluate the resources you have at your disposal. This includes both technical and human resources:

1. **Tools and Frameworks**: Research and select the appropriate tools and frameworks for test automation in Unity. AltTester® is a popular choice for Unity test automation, offering a range of features tailored for game testing. Other tools may also be required for specific types of tests or target platforms.
2. **Infrastructure**: Ensure that you have the necessary infrastructure to support automated testing. This includes setting up continuous integration/continuous deployment (CI/CD) pipelines, test environments, and hardware for running tests on different platforms.
3. **Budget**: Consider the budget available for acquiring tools, setting up infrastructure, and training your team. While some tools are free or open-source, others may require a subscription or one-time purchase.
4. **Time**: Allocate sufficient time for the initial setup and ongoing maintenance of test automation. Implementing test automation can be time-consuming initially, but in the long run, it pays off, because this approach reduces the time required for manual testing and the need for constant verification of new changes.
5. **Team Availability**: Assess the availability of your team members for setting up and maintaining

automated tests. Ensure that key team members can dedicate time to this effort without compromising other project responsibilities.

With a clear assessment of your needs and resources, you're now ready to select the right tools to implement test automation in your game development project. One such tool that stands out for its capabilities and ease of integration is AltTester®.

Using AltTester® tools in your Unity project

AltTester® is a powerful and accessible test automation tool designed specifically for games and applications developed with engines like Unity — and soon, Unreal 🎉. It allows developers to automate UI and gameplay tests, ensuring that your game functions correctly across different devices and scenarios.

The tools integrate seamlessly with popular CI/CD pipelines, enabling continuous testing and rapid feedback on code changes. By incorporating AltTester® into your project, you can achieve higher test coverage, faster testing cycles, and ultimately, a more reliable and polished product.

Preparing the project to integrate test automation

In the upcoming sections we will take Unity engine as an example. We will dive into the technical steps for integrating AltTester® into your game development project, covering installation, configuration, and creating your first automated tests.

1. Download the AltTester® tool package

The tool package contains 2 essential components of this integration: **AltTester® Unity SDK**, used for instrumenting your app and **AltTester® Desktop,** used in order to connect your application to the automated tests.

To download the package, visit our website.

> The free, open source package grants you limited access to AltTester® features. To enjoy the full range of features and benefits of AltTester® tools, start an **AltTester® Pro 14-day free trial**, or **buy a yearly or monthly membership**. More details here.

2. Import the package and build the instrumented application

This step requires access to the source code and Unity Editor, but once you have an instrumented build of your app, you can start using test automation. This step can be automated in a CI/CD for ease of use.

For instructions on how to instrument your app, visit our documentation or check out our video tutorial on How to instrument your game with AltTester® using a Windows Standalone build.

3. Create a C# project with AltTester® package

> We chose to use C# for this example, as it is one of the most popular programming languages used in game development and it is supported by the two largest game engines: Unity and Unreal.

- Start by creating a project using NUnit framework (our recommendation). To do this, you need to create an empty folder (this will be the project folder) and open it in your preferred IDE. Open a terminal and run the command below to create a basic project:

```
dotnet new nunit
```

- To be able to interact with the game through your tests, you need to add the AltDriver to your testing project by using the following command in your terminal:

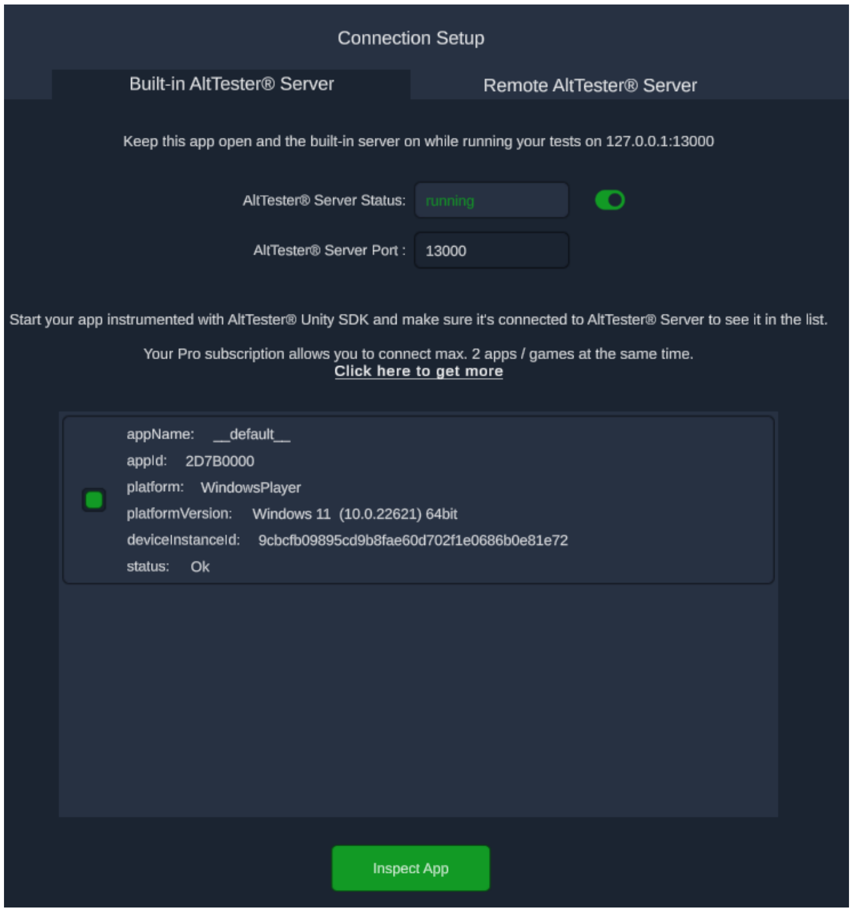
```
dotnet add package AltTester-Driver --version x.y.z
```

*Note that the **x.y.z** is the version of AltDriver you want to install.*

Check out our video tutorial that includes these steps for a more visual approach, or if you want to use other programming languages and frameworks, take a look at our documentation to see detailed test setups.

4. Setting up AltTester® Desktop

Last, but not least, the final setup step is to have AltServer up and running. AltServer is included in the AltTester® Desktop app, and it is used to create the required connection between your instrumented application and the AltDriver from your automated tests.



Keep this app open and the built-in server on while running your tests on 127.0.0.1:13000

AltTester® Server Status: running

AltTester® Server Port : 13000

Start your app instrumented with AltTester® Unity SDK and make sure it's connected to AltTester® Server to see it in the list.

Your Pro subscription allows you to connect max. 2 apps / games at the same time.
Click here to get more

appName: __default__
appId: 2D7B0000
platform: WindowsPlayer
platformVersion: Windows 11 (10.0.22621) 64bit
deviceInstanceId: 9cbcfb09895cd9b8fae60d702f1e0686b0e81e72
status: Ok

Install and open AltTester® Desktop, start your instrumented application, make sure the app is successfully connected to AltServer by checking the UI from the Desktop app and you're ready to start writing and running your tests.

Write and execute your first test with AltTester®

Once the project setup is complete, it is time to start building your test automation suite. AltTester® is very easy to use, especially if you have worked with Appium before. If you're interested in finding out the similarities between the two frameworks, check out our article on this topic.

With an **AltTester® Pro** license you have access to a wide range of features in AltTester® Desktop that can help you in writing the tests more efficiently without the need for access to the source code and the chosen game engine editor, once the build is instrumented.

To learn how to write and execute tests with AltTester®, our documentation will provide you with comprehensive step-by-step instructions. Here are some tutorials that could help you better understand and use AltTester® in this initial stage:

- Writing your first test with AltTester® – A c# guide for iOS
- How to write tests with AltTester® Robot Framework Library
- Run tests in parallel with AltTester®
- Record tests on BrowserStack devices with AltTester® Desktop

• • •

In a nutshell, here are the steps of the automation integration process:
✅ Assess the project's **needs** and specifics

✅ Evaluate the available (and required) **resources**

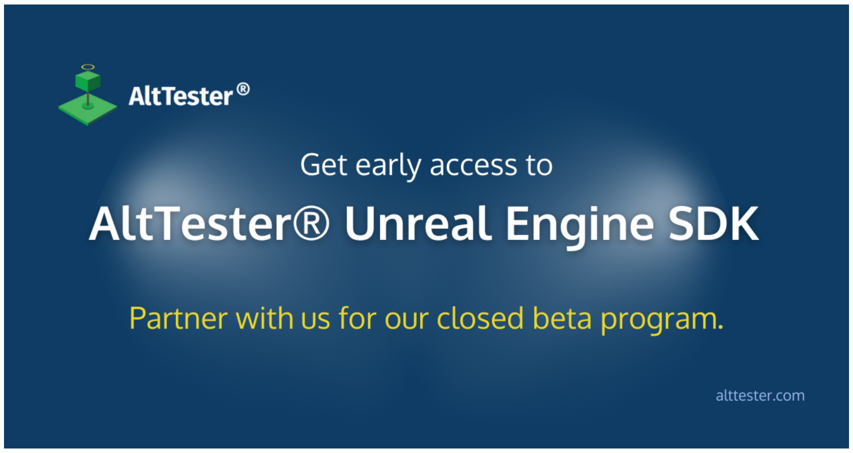
✅ Set up AltTester® as your test automation tool

✅ Write and execute your first test.

Either you use Unity or Unreal Engine for developing your game or application, the benefits and stages for adding test automation to the dev cycle are very similar. Our team at AltTester® is currently working on extending our end-to-end test automation platform for Unreal Engine as well, so all the technical steps previously described also apply to Unreal projects.

To learn more about our closed beta program for AltTester® Unreal Engine SDK, take a look at this recent post.

In conclusion, integrating test automation into your game development process with AltTester® offers numerous benefits that can significantly enhance your productivity and streamline your workflow. To learn more about what AltTester® can do to create a more efficient development process, check out this post.


AltTester®
Get early access to
AltTester® Unreal Engine SDK
Partner with us for our closed beta program.
alttester.com

AltTester | Game Development | Game Testing | Test Automation Insights | Testing Tools | Unity Test Automation | Unreal Engine Test Automation

Written by:

Diana Serb

BACK TO TOP

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment