



Services ▾

Industries ▾

About us ▾

News ▾

V-Blog

Software/Game Testing Service

# Game Testing Automation 101: Basic Tips and Strategies (with Case Studies)



Chi Vo

2023-09-18 07:43:43



# Game Testing Automation 101: Basic Tips and Strategies (with Case Studies)

*Testing is an integral part of game development. It ensures that the final product is free from critical bugs, plays smoothly, and provides an enjoyable gaming experience. In recent years, game testing automation has gained traction as a powerful tool for enhancing the efficiency and effectiveness of testing processes. In this comprehensive guide, [VNEXT Global](#) will delve into the world of game testing automation, providing step-by-step tutorials, insights into essential tools and frameworks, and best practices backed by real-world case studies.*

## Tutorials and Guides of Game Testing Automation

### Step 1: Setting up Automated Testing Environments

Before diving into automated game testing, you need to set up the necessary environment. Here's a step-by-step guide:

- **Select Your Testing Tools:** Begin by choosing the right tools and frameworks for your game. Popular choices include Unity Test Framework, Unreal Automation Testing, and third-party tools like Appium for mobile games.
- **Test Environment Setup:** Create a dedicated test environment that mirrors the game's production environment as closely as possible. This environment should include the same hardware, software, and

configurations.

- **Automated Testing Scripts:** Write test scripts that automate the testing process. These scripts should encompass a range of scenarios, from basic functionality to edge cases.
- **Test Data Preparation:** Prepare test data, including inputs and expected outputs for various test cases. Ensure that you have comprehensive test scenarios that cover gameplay, graphics, performance, and compatibility.
- **Running Automated Tests:** Execute the automated test scripts in your test environment. Monitor the test execution and collect data on test outcomes, including pass/fail status, logs, and screenshots.

## Step 2: Running Automated Tests

Once you have your testing environment set up, it's time to run automated tests:

- **Continuous Integration (CI):** Integrate automated tests into your CI/CD pipeline. This ensures that tests are run automatically whenever changes are made to the game's codebase, catching issues early in the development process.
- **Test Coverage:** Ensure that your automated tests cover a wide range of game features, including core gameplay

mechanics, user interfaces, and platform-specific functionality.

- **Parallel Testing:** Run tests in parallel to speed up the testing process. This is crucial for games with multiple platforms and configurations.
- **Performance Testing:** Implement performance tests to assess how your game performs under different conditions, such as high player loads or low-end hardware.
- **Regression Testing:** Continuously perform regression testing to identify new issues introduced by code changes and updates.

### Step 3: Analyzing Test Results

After running automated tests, it's vital to analyze the results:

- **Test Result Reports:** Generate detailed test result reports that include information on test cases, pass/fail status, logs, and screenshots. These reports provide a clear picture of the game's health.
- **Defect Tracking:** Implement a defect tracking system to log and prioritize issues identified during automated testing. This helps in effective bug management and resolution.
- **Performance Metrics:** Collect performance metrics, such

as frame rates, load times, and resource usage, to identify bottlenecks and areas for optimization.

- **Feedback Loop:** Establish a feedback loop between developers, testers, and automation engineers. Encourage collaboration to resolve issues promptly.



## Tools and Frameworks for Game Testing Automation

### Unity Test Framework

Unity Test Framework is a built-in testing framework for Unity game engine. It allows developers to write and run tests directly within the Unity Editor, making it a seamless choice for Unity game developers.

### Case Study: Hollow Knight

Team Cherry, the indie studio behind Hollow Knight, employed

Unity Test Framework to ensure the game's intricate platforming mechanics were free from bugs. They created unit tests to verify character movement and abilities, reducing the chances of gameplay issues.

## Unreal Automation Testing

Unreal Engine provides a robust automation testing framework that supports both functional and performance testing. Developers can write tests using Unreal's Blueprints or C++, ensuring comprehensive coverage of game features.

### Case Study: Gears 5

The Coalition, developers of Gears 5, utilized Unreal Automation Testing extensively during development. They ran automated tests to validate gameplay mechanics, AI behavior, and multiplayer functionality, leading to a polished and bug-free release.

## Third-Party Testing Tools

While Unity and Unreal offer built-in testing frameworks, third-party tools like Appium, TestComplete, and GameDriver can be valuable additions to your testing toolkit, especially for cross-platform or mobile games.

### Case Study: Candy Crush Saga

King, the creators of Candy Crush Saga, leveraged Appium for automated mobile game testing. They ensured compatibility across a wide range of devices and operating systems, maintaining a seamless gaming experience for millions of players.



## Best Practices for Game Testing Automation

### Practice 1: Comprehensive Test Coverage

To deliver a high-quality gaming experience, ensure that your automated tests cover various aspects of the game:

- **Gameplay Testing:** Test core gameplay mechanics, character interactions, and game progression.
- **Graphics Testing:** Verify graphics rendering, resolution, and visual effects across different hardware configurations.
- **Performance Testing:** Assess frame rates, load times, and resource utilization under different conditions.



- **Compatibility Testing:** Test the game on various platforms, devices, and operating systems to ensure broad compatibility.

## **Practice 2: Continuous Integration and Deployment (CI/CD)**

Continuous Integration (CI) and Continuous Deployment (CD) are practices that integrate automated testing seamlessly into the development process. Here's how it works:

- **CI:** Developers continuously commit their code to a shared repository. Automated tests are triggered automatically whenever code changes are made. This ensures that new code additions do not break existing functionality.
- **CD:** Once code passes all automated tests in the CI pipeline, it can be automatically deployed to a staging environment for further testing. This approach allows for rapid and reliable deployment of new features and bug fixes.

By implementing CI/CD, you catch issues early in the development cycle, preventing critical bugs from reaching the final release.

## **Practice 3: Load and Stress Testing**

Load and stress testing is especially important for online multiplayer games or those with a large player base. These

tests simulate scenarios where the game experiences high player loads or other extreme conditions. Key aspects of load and stress testing include:

- **Server Load Testing:** Evaluate how the game's servers handle a massive influx of players, ensuring that they remain stable and responsive.
- **Network Stress Testing:** Simulate high-latency or packet loss scenarios to check how the game performs under less-than-ideal network conditions.
- **Player Load Testing:** Assess the game's performance when a large number of players are in a single match or session, verifying that it can handle the load without crashing or lagging.

By conducting load and stress testing, you ensure that your game can maintain a consistent and enjoyable experience even during peak usage times, minimizing player frustration.

## **Practice 4: Community Feedback Integration**

Embrace player feedback and integrate it into your automated testing process. This can help identify issues that may not be apparent through automated tests alone.

- **Feedback Collection:** Create channels for players to report issues, share suggestions, and provide feedback. This can include in-game reporting systems, forums, social media, and customer support.

- **Issue Prioritization:** Implement a system to categorize and prioritize reported issues. Critical issues affecting gameplay or stability should be addressed promptly.
- **Automated Feedback Analysis:** Utilize sentiment analysis tools and natural language processing to automate the analysis of player feedback. This can help identify recurring themes and prioritize areas for improvement.
- **Feedback-Driven Testing:** Based on player feedback, create automated test cases that specifically target reported issues. This ensures that the same problems do not reoccur in future releases.

### **Case Study: Minecraft**

Mojang Studios constantly updates Minecraft based on player feedback. They use automated testing to validate player-reported issues and prioritize bug fixes, ensuring a positive player experience.

## **Practice 5: Accessibility Testing**

Accessibility testing is a fundamental aspect of game testing that is often overlooked. It ensures that your game is inclusive and accessible to players with disabilities. Key considerations for accessibility testing include:

- **Screen Reader Compatibility:** Verify that the game's menus and interfaces are compatible with screen readers and

provide alternative text for non-text elements.

- **Colorblind Accessibility:** Test the game's visuals to ensure that colorblind players can distinguish important elements.
- **Input Options:** Ensure that the game supports various input methods, including keyboard, mouse, gamepad, and touch screen, and allow players to customize controls.
- **Subtitles and Captions:** Include subtitles and captions for dialogue, sound effects, and other audio elements to accommodate players with hearing impairments.
- **Test with Assistive Technologies:** Use assistive technologies such as screen readers, voice recognition software, and switch control devices to assess the game's accessibility.

*Want to know more about automated testing tips? This blog may pique your interest!*

[5 Software Testing Tips That Any Automation Testers Should Know](#)



## Final thoughts:

Game testing automation is a critical tool for game developers seeking to deliver high-quality, bug-free gaming experiences. By following the step-by-step tutorials and best practices outlined in this guide, developers can set up effective automated testing processes. If you find automated game testing too complicated and time-consuming, VNEXT Global's always at your disposal! Check out our [game-testing portfolio](#) and reach out anytime!

If you are looking for a trusted IT partner, VNEXT Global is the ideal choice. With 14+ years of experience, we surely can help you to optimize your business digitalization within a small budget and short time. Currently, we have 400+ IT consultants and developers in **Mobile App**, **Web App**, **System Development**, **Blockchain Development** and **Testing Services**. We have provided solutions to 600+ projects in several industries for clients worldwide. We are willing to become a companion on your way to success. Please tell us when is convenient for you to have an online meeting to discuss this further. Have a nice day!

# Read more on our Blog

[V-Blog](#)[System Development](#)

## How to build an HRM software development?

[1 mins reading](#)[V-Blog](#)[System Development](#)

## Agile Environment: A to know about

[1 mins reading](#)



---

Contact

024-3765-9555

info@vnext.vn

Head office

18F-19F-20F, C Tower,  
Central Point, 219 Trung Kinh  
Str., Cau Giay Dist., Hanoi,  
Vietnam

Da Nang office

3rd-5th Floor, UST Tower, 639  
Nguyen Tat Thanh Str., Thanh  
Khe Dist., Da Nang City,  
Vietnam





[Support](#)   [Privacy Policy](#)   [Terms and Conditions](#)

2022 VNEXT Global, All Rights Reserved