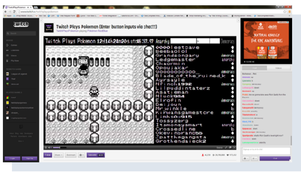
Automating Gameplay with TestComplete



Gregory Mooney
February 28, 2014

The recent Pokemon experiment on Twitch.Tv has been a blast to follow. If you haven't been keeping up with the madness, basically Pokemon is being streamed via Twitch.Tv and the game is played by subscribers typing commands into the chat menu. The problem here, as with most places on the internet, is that since any subscriber can type a command into the chat there is a lot of room for trolls to completely sabotage the game being played.

It's been a couple of weeks (with upwards of 30 million page visits) since the experiment started and believe it or not, the game has progressed pretty far. Despite all the trolls trying to cause as much chaos as possible, progress has been made. Last time I checked they had beaten almost all of the gym leaders.

This recent stunt by gamers got me thinking: When I see hundreds of gamers inputting commands at the same time, it really looks like some kind of ad hoc testing. The test results would be limited, but I could see someone finding a bug in the game from playing it in a way that wasn't a use case when it was originally developed.

When you contemplate automated testing tools, it's hard to imagine using them for something like ad hoc testing for video games. In reality, test automation is rarely used in video game testing at all -- I can think of a few use cases in which you could use it for data-driven testing if the game uses databases or for certain regression tests.

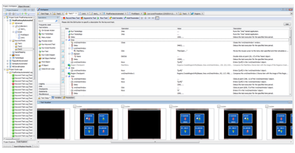
But that doesn't mean you can't use test automation for gameplay, it just might now be the most effective way since user interaction with a game is the most important aspect of gaming that you will need to test to. Much in the same way that Twitch.Tv playing Pokemon is possibly a unique way to test, I figured I'd see how a test automation tool could be used to to automate the gameplay itself.

I have an emulator of Nintendo and a Rom of Final Fantasy (I own both) and I set up TestComplete to run the emulator, load the game and then attempt to progress through the game.

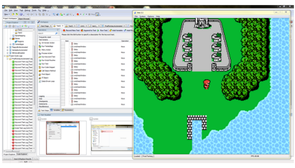
The staging process was a bit timelier than I had anticipated, since I had to find an emulator and a rom of the game I wanted to test. I chose Final Fantasy because of the fond memories I have playing the game and because the UI makes for gameplay that can be easily automated—a game like Contra would have had tests failing seconds into gameplay due to the amount of enemies.

Next was setting up my test environment and then creating the first part of the test.

In order for the same test to run correctly each time, I would have to start the test by automating the boot process and setting up the game - this means creating a new game, selecting the type of character and naming my party. To confirm that the first part of the test automation completed correctly, I set up a series of checkpoint to make sure the test progressed as planned.



Once the test reached actual gameplay, I was in the clear and the game basically became a test with a long list of commands to input. What I had to keep into consideration now was the fact that this game has many random encounters with enemies. Luckily, your party starts off near a town, so in order to avoid any battles, all I had to do was input commands that would allow my party to reach the town and then do some actions inside the town.



With a bit of elbow grease I actually was able to get the game to play automatically... sort of. To be honest, I'm still trying to work out some of the kinks where commands are not being recognized by the emulator, as you will see in the video below *(I am currently troubleshooting this issue and will hopefully have a fix soon)*:

If I were to go a step further, I could set up specific test logic. For instance, if I enter a random monster encounter, I could have each of the party members attack the monster in a certain order with particular skills. Obviously, this could turn into as much of a project as developing the game itself, but may prove useful for development and testing teams in their regression suites.

Getting back to ad hoc testing, you could also have the game run through a series of randomized command inputs to see how it reacts. You never know, you may find some test results that could prove useful in ensuring that your game is as defect-free as possible.

That said, the time and work it takes to automate gameplay using test automation for video game testing will all depend on the resources your team has available. Frankly, it's a lot of work. Traditional manual testing may be the best approach, but at least this little experiment proves that test automation is a viable option when necessary.

In any case, it was fun to watch one of my favorite games play by itself.

See also:

- Is Gaming the Hottest Bed of Innovation in Software Development?
- The Half-Life of Innovation
- Creative Software Testing in Agile Environment | Iain McCowatt

[dfads params='groups=937&limit=1&orderby=random']

YOU MIGHT ALSO LIKE

The Hidden Cost of Software Glitches: How Quality Drives Your Business

What if a single software glitch could cost your company millions? In today&rsquo...

Manage

Transform Your Software Delivery: Boost Quality and Speed with Unified API and Testing Strategies

APIs are vital in today's digital transformation, but their seamless delivery is...

Test

Four Strategies to Level-up Your Test Automation in Jira

For years, the traditional approach to automating tests relied heavily on scripted...

Test

Explore SmartBear Products

AlertSite
AQTime Pro
BitBar
BugSnag
Capture for Jira
Collaborator
Cucumber for Jira
Cucumber Open
LoadNinja
Pact
PactFlow
ReadyAPI
SoapUI
Swagger
SwaggerHub
SwaggerHub Explore
TestComplete
TestEngine
TestLeft
VisualTest
Zephyr

About Us | Careers | Solutions | Partners | Responsibility

Contact Us | +1 617-684-2600 USA | +353 91 398300 EUR | +61 391929960 AUS