DADIU

Automated Buid System

SPOKEN

DADIU
12-4-2015

Contents

Introduction

For automating the builds we decided to use Jenkins. Initially we were introduced to it in the Joint Curriculum at the programmers' workshop. Jenkins is continues integration server installed on one of the computers in the production room. Basically Continuous Integration is the practice of running your tests on a non-developer machine automatically every time someone pushes new code into the source repository. This has the tremendous advantage of always knowing if all tests work and getting fast feedback. The fast feedback is important so you always know right after you broke the build (introduced changes that made either the compile/build cycle or the tests fail) what you did that failed and how to revert it. If you only run your tests occasionally the problem is that a lot of code changes may have happened since the last time and it is rather hard to figure out which change introduced the problem. When it is run automatically on every push then it is always pretty obvious what and who introduced the problem.

Built on top of Continuous Integration are Continuous Deployment/Delivery where after a successful test run your instantly and automatically release the latest version of your codebase. Makes deployment a non-issue and helps you speed up your development. Because Jenkins is widely spread and used in software development, it has support for many different platforms, including support for Unity engine. Furthermore, we are using GitHub for handling the version control of our project, which is also supported in Jenkins.

Set up

After we have set up Jenkins and installed all the needed plugins (Unity3d Plugin, GitHub Plugin) we were ready to create a Job (Job is a procedure in Jenkins which runs a build depending on the settings that are provided).

GitHub plugin is used to link GitHub repository with Jenkins. By using it we are able to add our GitHub login credentials so Jenkins can use them when trying to connect to GitHub. Jenkins needs that in order to pull the latest changes from the repository. Furthermore, GitHub provides WebHook support, so by adding that feature, Jenkins can register when a new content is pushed to GitHub and schedule a build.

Unity3d plugin is used to locate the Unity engine installation and invoke a method execution through external commands the method invoked is located in a script we have added in Unity's Editor folder. In this method we configure which scenes to build and what tests to run. To invoke that method from Jenkins we use the following command:

-quit -batchmode -executeMethod PerformBuild.CommandLineBuildAndroid -nographics -bversion ${BUILD_NUMBER}

We first quit Unity if it is open, then we execute a method called ***CommandLineBuildAndroid*** in the class ***PerformBuild*** which is located in the Editor folder in our project. ***-nographics*** command makes the build faster. ***-bversion*** is a function which gets the latest version of the build and increment it once, so we can keep track of the builds.

In the Job we specified the branch to take the code from version control, the commands invoked in Unity and any pre and post processes that have to be executed for the job to be done (like creating a zip archive).

Testing

For security purposes we implemented couple of simple unity tests, in order to make sure our build is stable. Then we created a function, which triggers only whenever a build is to be made. This function picks randomly one or more of the above-mentioned tests and runs it in Unity Engine using the following command: **-batchmode -executeMethod UnityTest.Batch.RunUnitTests –quit.** If the test passes, this means that the build is ready to be sent to the QA managers for QA and UX evaluation. Else, the code has to be revised for critical bugs.

References

- https://www.quora.com/What-is-Jenkins-When-and-why-is-it-used, 25.11.2015