Unity3dBuilder Plugin

Plugin Information
View Unity3d on the plugin site for more information.

Join the community?

If you want to build Unity3d with Jenkins, or are already doing so, please drop a line on the Unity3d forums, and tell us more about what you do or would like to do.

A problem, An idea ?

Please use our tasks and issues tracker to report bugs, improvements or new feature.

- Report a bug
- Ask for a new feature
- Ask for an improvement of an existing feature

Also if you want to propose some code change using a Github pull request, please open also a Jira issue. It is easier for developers to track them.

Unity3d is a powerful 3d game creation editor and engine that runs on Mac and Windows.

This plugin adds the ability to call the Unity3d Editor from the command line to automate build and packaging of Unity3d applications.

Table of content

Background

Automating Unity3d builds from the command line is possible. There are a few problems though:

- the unity runner writes its output to a separate log file, instead of the output
- tool and file locations are platform specific
- the editor is very GUI centered and only provides default build strategies

Features

This plugin aims to make it easier to run Unity3d builds easily in Jenkins, by adding the following features:

- log file redirection
- distributed builds

The plugin was tested with versions ranging from unity3d 3.4.2 to 5.0.1. Tested on distributed and single server environments

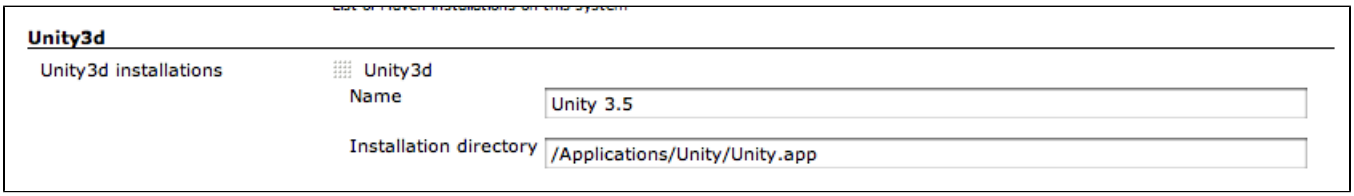
Documentation

Installation guide

As Unity3d is multi-platform, you may need to install the proper third party OS & tools (Android SDK, XCode, etc) depending on the type of build targets you intend to exercise.

Install the latest version of the plugin from the update center and configure a freestyle job (see #Usage Guide). If necessary restrict the job to the node(s) that will perform the build(s).

On the node(s) you are going to build Unity projects, add at least one unity3d installation (Manage Jenkins -> Global Tool Configuration) and configure the location of the Unity3d installation. This location is by default `/Applications/Unity/Unity.app` on Mac OS X and `C:\Program Files (x86)\Unity` on Windows. The plugin will automatically suffix the installation path with the proper executable location.



Usage guide

Prerequisites

The plugin assumes you've created a special Editor class with at least one method responsible for your build.

Here's an example extracted from one of our projects:

Assets/Editor/MyEditorScript.cs

```
class MyEditorScript {
        static string[] SCENES = FindEnabledEditorScenes();

        static string APP_NAME = "YourProject";
        static string TARGET_DIR = "target";

        [MenuItem ("Custom/CI/Build Mac OS X")]
        static void PerformMacOSXBuild ()
        {
                 string target_dir = APP_NAME + ".app";
                 GenericBuild(SCENES, TARGET_DIR + "/" + target_dir, BuildTarget.StandaloneOSXIntel,
BuildOptions.None);
        }

        private static string[] FindEnabledEditorScenes() {
                List<string> EditorScenes = new List<string>();
                foreach(EditorBuildSettingsScene scene in EditorBuildSettings.scenes) {
                        if (!scene.enabled) continue;
                        EditorScenes.Add(scene.path);
                }
                return EditorScenes.ToArray();
        }

        static void GenericBuild(string[] scenes, string target_dir, BuildTarget build_target, BuildOptions
build_options)
        {
                EditorUserBuildSettings.SwitchActiveBuildTarget(build_target);
                string res = BuildPipeline.BuildPlayer(scenes,target_dir,build_target,build_options);
                if (res.Length > 0) {
                        throw new Exception("BuildPlayer failure: " + res);
                }
        }
}
```

User/OS setup

Mac OS X

On Mac OS X, the user running needs to be logged in otherwise the Unity3d editor might fail to acquire the graphical resources.

If you don't you might see something like

```
Piping unity Editor.log from /Users/Shared/Jenkins/Library/Logs/Unity/Editor.log
[Schicksalsklinge HD Windows] $ /Applications/Unity/Unity.app/Contents/MacOS/Unity -projectPath "/Users/Shared
/Jenkins/Home/workspace/ProjectXYZ/repo" -quit -batchmode -buildLinux32Player -executeMethod BuildProject.
PerformLinuxTestBuild
Initialize mono
Mono path[0] = '/Applications/Unity/Unity.app/Contents/Frameworks/Managed'
Mono path[1] = '/Applications/Unity/Unity.app/Contents/Frameworks/Mono/lib/mono/2.0'
Mono config path = '/Applications/Unity/Unity.app/Contents/Frameworks/Mono/etc'
Using monoOptions --debugger-agent=transport=dt_socket,embedding=1,defer=y
_RegisterApplication(), FAILED TO establish the default connection to the WindowServer, _CGSDefaultConnection()
is NULL.
2015-04-18 21:20:08.497 Unity[702:27617] NSDocumentController Info.plist warning: The values of
CFBundleTypeRole entries must be 'Editor', 'Viewer', 'None', or 'Shell'.
```

Windows

FIXME describe and add a log about what happens on Windows.

Linux

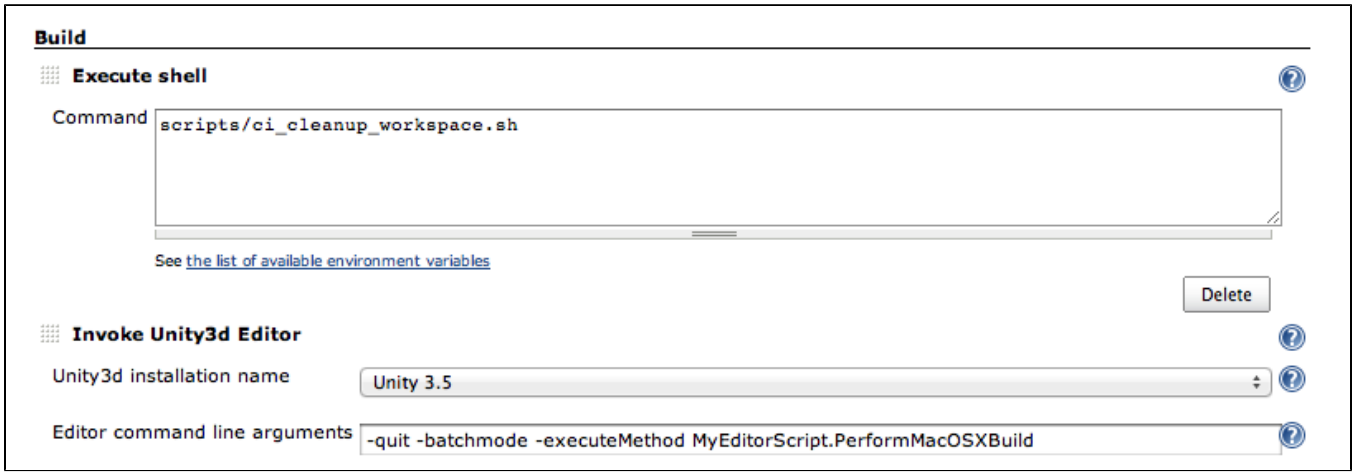
On Linux, you will need an X server. If you are running Jenkins on a headless server, use the Xvfb Plugin. See also this thread.

Build queue

The Unity Editor can only perform one build at a time on a given projectPath. If you want to run multiple builds in parallels for the same project, you will need to create multiple jobs, each with their workspace.

Setting up a build step

Add the Unity3d build step to a free-style project, select the unity3d installation and set your command line arguments (e.g. `-quit -batchmode -executeMethod MyEditorScript.PerformMacOSXBuild`). If you do not specify -projectPath (case-sensitive), the plugin will use the current workspace. You may want to add an extra step to clean the project before you build to make sure the build starts in a clean state.



If you want to build for iOS, you will have to add extra build steps to trigger xcode build. This step isn't covered here. Same for Windows 8/10.

Unity3d Builder configuration parameters

Parameter	Since version	Description
command line	0.1	The full command line, the builder adding the -projectPath (case-sensitive) if it isn't specified
unstable return codes	1.0	The optional comma separated list of command line return codes that should result in unstable builds instead of failures. E.g '2,3' if you use Unity3d Test Results

Tips

Using Unity3d with large set of jobs

- use multiple executors. Ensure that you do not run multiple concurrent build from the same job
- use the global argLine (from 0.6) to configure default configuration in one place
- specify the -logFile argument to be relative to each project. You don't want all concurrent projects to use the same standard editor.log file
- combine it with a plugin like EnvInject to differentiate between jobs
- automate the install of unity

Automatically installing unity3d (MacOSX)

To automatically install unity3d from jenkins (even beta versions),

1. install this set of scripts.

1. Install this script somewhere on your machines. E.g. /Users/Shared/Jenkins/Home/bin/
2. Create a parametrized job that takes a String parameter
   name: UNITY3D_URL
   default value: e.g. http://netstorage.unity3d.com/unity/unity-4.2.1.dmg
   description: The URL of the DMG package to install
3. Add a shell builder to your job:

   shell build step

   ```
   cd
   echo "Installing $UNITY3D_URL"
   download_install_unity3d.sh "$UNITY3D_URL"
   ```

Friendly plugins

- The EnvInject Plugin can help you parametrize your command line for maintaining large amount of projects in a similar manner
- The Log Parser Plugin can help you to quickly set some parsing rules for your Unity3d builds. Here's a tentative set of rules that we use in one project:

```
start /^Initialize mono/
start /^- starting compile/
start /^Mono dependencies included in the build/
start /^Textures/
info /^Complete size/
warning /warning CS/
error /error CS/
start /^-----CompilerOutput:-stdout/
info /^Compilation succeeded/
error /^Compilation failed/
start /^Used Assets, sorted by uncompressed size/
info /^\*\*/
info /^Exiting batchmode successfully now/
start /^=== BUILD NATIVE TARGET/
start /^Packaging IPA/
start /^Archiving artifacts/
start /Uploading to testflight/
```

- Xvfb Plugin for running Unity on Linux headless servers.

Known issues

type	key	summary	assignee	reporter	priority	status	resolution	created	updated	due

⚠ Can't show details. Ask your admin to whitelist this Jira URL.

View these issues in Jira

Troubleshooting failures

If the plugin fails to run the command you want it to run here are some steps you can perform to help identifying the issue:

- check the editor.log and/or the job console for any errors. If you don't see any log, check if you specify the -logFile argument.
- if you are trying to run an -executeMethod argument, expose the editor method in the Unity3d menus ([MenuItem ("../...")]) and run it. Bonus if you are able to run it on the machine it is supposed to run. If the command doesn't run here, then the problem isn't in the plugin
- run your build command from a Jenkins "Run in a Windows batch command" build step. If the job fails, then check the arguments (the plugins may have fiddled with the arguments incorrectly).
- run your build command from a CMD on the machine you intend to. If that works, and the run from jenkins doesn't, compare the environments (user, permissions, access to graphics devices, etc)

Related information

- Automated management of iOS provisioning profiles
- Unity3d build pipeline using jenkins
- Command line install of Unity3d on mac

Changelog

Version 1.4 (DEV IN PROGRESS)

Version 1.3 (11.09.2015)

- fix broken mac build support (JENKINS-30396)

Version 1.2 (10.09.2015)

- added Linux support (JENKINS-30321)

Version 1.1 (07.07.2015)

- improve documentation and feedback when using parametrized path for jenkins installation homes (JENKINS-29218)
- improved command line parsing WRT environment and build parameters (JENKINS-29226)

Version 1.0 (29.05.2015)

- allow some return codes to turn the build to UNSTABLE, easing UnityTestTools integration (JENKINS-24386)

Version 0.9 (26.05.2015)

- fixed the detection of the proper location of the Editor.log on Windows (JENKINS-24265)
- improved Unity3d installation directory configuration documentation and error checks for both distributed and non distributed setups (JENKINS-20349)

Version 0.8 (25.05.2015)

- fixed the "Pipe broken" issue (JENKINS-23958) in distributed builds

Version 0.7 (02.04.2015)

- Prevent hanging if the Editor.log file we are looking at isn't been written to (JENKINS-27710). Consequence of us not finding the Editor.log on Windows 2008 installations.

Version 0.6 (24.03.2014)

- reduce risks of truncating console
- fix command line documentation issue
- properly handle editor.log piping when using the -logFile argument
- global argLine

Version 0.5 (27.09.2003)

- fix command line setting been overwritten at execution time

Version 0.4 (16.09.2013)

- support build and environment variables injection into the command line

Version 0.3 (06.06.202)

- Validity of Unity3D project folder was not correctly checked when projectPath parameter was used.

Version 0.2 (30.01.2012)

- (JENKINS-12590)

Version 0.1 (24.01.2012)

- live redirection of the Editor.log file into the console
- supports distributed builds
- automatically adds the -projectPath command line