using UnityEngine;

using UnityEditor;

using System;

using System.Collections.Generic;

using System.IO;

using UnityEngine.SceneManagement;

namespace Dorothy

{

public class ProjectBuilder

{

[MenuItem("NEXT/Tools/CI/Build Android")]

public static void PerformAndroidBuild()

{

WriteLineLog("");

WriteLineLog("PerformAndroidBuild Time: " + DateTime.Now);

*// 입력된 파라미터를 통해 빌드 설정을 가져온다.*

String[] arguments = System.Environment.GetCommandLineArgs();

string APP\_VERSION = string.Empty;

string SVN\_REVISION = string.Empty;

string APP\_BUILD = string.Empty;

string ASSETBUNDLE\_BUILD = string.Empty;

string SCENES\_IN\_BUILD = string.Empty;

string ASSET\_BUNDLE = string.Empty;

string ENVIROMENT = string.Empty;

string FLATFORM = string.Empty;

string DEFINE\_SYMBOLS = string.Empty;

for (int i = 0; i < arguments.Length; i++)

{

string[] splitArguments = arguments[i].Split(':');

if (splitArguments.Length == 0)

continue;

string argumentKey = splitArguments[0];

string argumentValue = splitArguments.Length > 1 ? argumentValue = splitArguments[1] : string.Empty;

if (argumentKey == "APP\_VERSION")

APP\_VERSION = argumentValue;

if (argumentKey == "SVN\_REVISION")

SVN\_REVISION = argumentValue;

if (argumentKey == "APP\_BUILD")

APP\_BUILD = argumentValue;

if (argumentKey == "ASSETBUNDLE\_BUILD")

ASSETBUNDLE\_BUILD = argumentValue;

*//if (argumentKey == "SCENES\_IN\_BUILD")*

*// SCENES\_IN\_BUILD = argumentValue;*

*//if (argumentKey == "ASSET\_BUNDLE")*

*// ASSET\_BUNDLE = argumentValue;*

if (argumentKey == "ENVIROMENT")

ENVIROMENT = argumentValue;

if (argumentKey == "FLATFORM")

FLATFORM = argumentValue;

if (argumentKey == "DEFINE\_SYMBOLS")

DEFINE\_SYMBOLS = argumentValue;

}

WriteLineLog("FLATFORM: " + FLATFORM + " /APP\_VERSION: " + APP\_VERSION + " /APP\_BUILD: " + APP\_BUILD + " /ASSETBUNDLE\_BUILD: " + ASSETBUNDLE\_BUILD + " /SVN\_REVISION: " + SVN\_REVISION);

*// 환경설정 변경*

if (APP\_VERSION != string.Empty && SVN\_REVISION != string.Empty)

{

*//Dorothy.Environment.Instance.Version = APP\_VERSION + "." + SVN\_REVISION;*

Dorothy.Environment.Instance.Version = APP\_VERSION;

bool changeTargetResult = Dorothy.EnvironmentInspector.SelectTarget(ENVIROMENT);

WriteLineLog("Dorothy.Environment.Instance.Version: " + Dorothy.Environment.Instance.Version + " /Target: " + Dorothy.Environment.Instance.Target + " /changeTargetResult: " + changeTargetResult);

*// 현재는 DEV, LIVE 모두 같은 에셋번들 버전을 사용하므로 통일*

Dorothy.Environment.Instance.Dev.AssetBundleVersion = SVN\_REVISION;

Dorothy.Environment.Instance.DQ.AssetBundleVersion = SVN\_REVISION;

Dorothy.Environment.Instance.Live.AssetBundleVersion = SVN\_REVISION;

*//if (Dorothy.Environment.Instance.Target == EnvironmentField.Target.Dev)*

*// Dorothy.Environment.Instance.Dev.AssetBundleVersion = SVN\_REVISION;*

*//else if (Dorothy.Environment.Instance.Target == EnvironmentField.Target.DQ)*

*// Dorothy.Environment.Instance.DQ.AssetBundleVersion = SVN\_REVISION;*

*//else if (Dorothy.Environment.Instance.Target == EnvironmentField.Target.Live)*

*// Dorothy.Environment.Instance.Live.AssetBundleVersion = SVN\_REVISION;*

}

*// 번들로 만들 씬 이름 리스트를 만든다.*

*//List<string> bundleSceneList = GetSceneList(ASSET\_BUNDLE);*

*// 빌드에 포함할 씬이름 리스트를 만든다.*

*//List<string> includeInBuildSceneList = GetSceneList(SCENES\_IN\_BUILD);*

*// ST-MARIA 폴더 내 모든 씬의 AssetLabels를 지운다.*

*//ClearBundleName();*

*// 빌드할 씬 SCENES에 저장.*

*//string[] SCENES;*

*//if (includeInBuildSceneList.Count != 0)*

*// SCENES = GetBuildScenePaths(includeInBuildSceneList);*

*//else*

*// SCENES = FindEnabledEditorScenes();*

*// 에셋번들 이름을 설정한다.*

*//AssignBundleName(bundleSceneList, SVN\_REVISION, GetAssetbundleExtention(FLATFORM));*

*// 플레폼을 변경한다.(실행파일, 에셋번들 만들기 전에 플렛폼을 변경해 줘야 함)*

*//if (includeInBuildSceneList.Count > 0 || bundleSceneList.Count > 0)*

SwitchFlatform(FLATFORM, DEFINE\_SYMBOLS);

*// 실행파일을 만든다. (apk, exe..)*

*//if (includeInBuildSceneList.Count > 0)*

*// GenericBuild(SCENES, FLATFORM, ENVIROMENT);*

*// App 빌드*

if ((string.IsNullOrEmpty(APP\_BUILD) == false) && (APP\_BUILD.CompareTo("true") == 0))

GenericBuild(FLATFORM, ENVIROMENT);

*// 에셋번들을 만든다.*

if ((string.IsNullOrEmpty(ASSETBUNDLE\_BUILD) == false) && (ASSETBUNDLE\_BUILD.CompareTo("true") == 0))

{

if ((Dorothy.Environment.Instance.Target == EnvironmentField.Target.Dev) ||

(Dorothy.Environment.Instance.Target == EnvironmentField.Target.DQ) ||

(Dorothy.Environment.Instance.Target == EnvironmentField.Target.Live))

BuildAssetBundles.BuildAllAssetBundles(FLATFORM, SVN\_REVISION);

else

BuildAssetBundles.BuildAllAssetBundles(FLATFORM, APP\_VERSION);

}

}

static string GetAssetbundleExtention(string flatform)

{

switch (flatform)

{

case "StandaloneWindows": return "desktop";

case "Android": return "android";

case "iOS": return "ios";

default:

WriteLineLog("Not define GetAssetbundleExtention: " + flatform);

return string.Empty;

}

}

static string[] GetBuildScenePaths(List<string> includeInBuildSceneList)

{

WriteLineLog("GetBuildScenePaths: " + includeInBuildSceneList.Count);

List<string> editorScenes = new List<string>();

foreach (string iter in includeInBuildSceneList)

{

string scenePath = GetScenePath(iter);

WriteLineLog("scenePath: " + scenePath);

editorScenes.Add(scenePath);

}

return editorScenes.ToArray();

}

*// ST-MARIA 폴더 내 모든 씬의 AssetLabels를 지운다.*

static void ClearBundleName()

{

string baseSceneName = Application.dataPath + "/ST-MARIA";

string[] finedPaths = Directory.GetFiles(baseSceneName, "\*.unity", SearchOption.AllDirectories);

foreach (string sceneName in finedPaths)

{

string assetPath = sceneName.Replace(Application.dataPath, "Assets");

AssetImporter.GetAtPath(assetPath).SetAssetBundleNameAndVariant("", "");

}

}

*// 번들로 만들씬의 에셋번들 이름을 설정한다. (SceneName.SVN\_REVISION.unity.extention)*

static void AssignBundleName(List<string> bundleSceneList, string svnRevision, string extention)

{

if (bundleSceneList.Count == 0)

return;

foreach (string iter in bundleSceneList)

{

string assetPath = GetScenePath(iter);

string assetBundleName = string.Format("{0}.{1}.unity.{2}", iter, svnRevision, extention);

AssetImporter.GetAtPath(assetPath).SetAssetBundleNameAndVariant(assetBundleName, "");

}

}

*// ST-MARIA 폴더 안에 있는 씬 이름을 반환한다. (반환값 예시: Assets/ST-MARIA/GAMES/SLOTS/A0/A001/Scenes/MACHINE-A001.unity)*

static string GetScenePath(string findSceneName)

{

string baseSceneName = Application.dataPath + "/ST-MARIA";

string[] finedPaths = Directory.GetFiles(baseSceneName, findSceneName + ".unity", SearchOption.AllDirectories);

if (finedPaths.Length != 1)

return string.Empty;

finedPaths[0] = finedPaths[0].Replace(Application.dataPath, "Assets");

return finedPaths[0];

}

static List<string> GetSceneList(string sceneStr)

{

List<string> sceneList = new List<string>();

foreach (string iter in sceneStr.Split(','))

{

if (iter == string.Empty)

continue;

WriteLineLog("GetSceneList Add: " + iter);

sceneList.Add(iter);

}

return sceneList;

}

static List<string> GetIncludeInBuildSceneList(string allSceneStr, List<string> bundleSceneList)

{

List<string> includeInBuildSceneList = new List<string>();

foreach (string iter in allSceneStr.Split(','))

{

if (iter == string.Empty)

continue;

string sceneName = iter.Replace(":selected", "");

if (bundleSceneList.Contains(sceneName) == false)

includeInBuildSceneList.Add(sceneName);

}

return includeInBuildSceneList;

}

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 SwitchFlatform(string flatform, string defineSymbols)

{

switch (flatform)

{

case "StandaloneWindows":

*// Scripting Define Sysbols 세팅*

PlayerSettings.SetScriptingDefineSymbolsForGroup(BuildTargetGroup.Standalone, defineSymbols);

*// 빌드 타켓 변경*

EditorUserBuildSettings.SwitchActiveBuildTarget(BuildTargetGroup.Standalone, BuildTarget.StandaloneWindows64);

break;

case "Android":

*// Scripting Define Sysbols 세팅*

PlayerSettings.SetScriptingDefineSymbolsForGroup(BuildTargetGroup.Android, defineSymbols);

*// Key 설정*

PlayerSettings.Android.keyaliasName = "slotoday";

PlayerSettings.Android.keyaliasPass = "slotoday123";

PlayerSettings.Android.keystoreName = "slotoday.keystore";

PlayerSettings.Android.keystorePass = "slotoday123";

*//PlayerSettings.Android.keyaliasName = "test";*

*//PlayerSettings.Android.keyaliasPass = "st1234!";*

*//PlayerSettings.Android.keystoreName = "ST.keystore";*

*//PlayerSettings.Android.keystorePass = "st1234!";*

*// 빌드 타켓 변경*

EditorUserBuildSettings.SwitchActiveBuildTarget(BuildTargetGroup.Android, BuildTarget.Android);

break;

case "iOS":

break;

}

}

static void GenericBuild(string flatform, string enviromentTarget)

{

string target\_filename = "Dorothy-" + enviromentTarget;

string buildResult = string.Empty;

string[] scenes = new string[SceneManager.sceneCountInBuildSettings];

for (int i = 0; i < SceneManager.sceneCountInBuildSettings; ++i)

{

scenes[i] = SceneUtility.GetScenePathByBuildIndex(i);

*//Debug.Log("Path: " + sceneList[i]);*

}

switch (flatform)

{

case "StandaloneWindows":

*//buildResult = BuildPipeline.BuildPlayer(scenes, target\_filename + ".exe", BuildTarget.StandaloneWindows64, BuildOptions.None);*

*// 버전이 2018.3 이후로 올라가면서 따로 BuildReport 오브젝트를 반환하게 되어있음. 그래서 해당 부분 일단 결과만 string 으로 뿌려주게 변경하였는데,*

*// 자세한 부분은 result 안에 있는 부분 가져와서 Log 찍도록 함. (사용자에 의해 변경 여지가 있어서 주석 남겨둡니다.*

var reportWin = BuildPipeline.BuildPlayer(scenes, target\_filename + ".exe", BuildTarget.StandaloneWindows64, BuildOptions.None);

if (reportWin.summary.result != UnityEditor.Build.Reporting.BuildResult.Succeeded)

buildResult = reportWin.summary.result.ToString();

break;

case "Android":

var reportAnd = BuildPipeline.BuildPlayer(scenes, target\_filename + ".apk", BuildTarget.Android, BuildOptions.None);

if (reportAnd.summary.result != UnityEditor.Build.Reporting.BuildResult.Succeeded)

buildResult = reportAnd.summary.result.ToString();

break;

case "iOS":

break;

}

if (string.IsNullOrEmpty(buildResult))

{

WriteLineLog("Build success! flatform: " + flatform);

}

else

{

WriteLineLog("BuildPlayer flatform: " + flatform + " /failure: " + buildResult);

}

}

static void WriteLineLog(string saveLog)

{

StreamWriter writer = OpenStreamWriter();

writer.WriteLine(saveLog);

writer.Close();

}

static StreamWriter OpenStreamWriter()

{

string path = "DorothyBuildLog.txt";

*//Write some text to the test.txt file*

StreamWriter writer;

if (!File.Exists(path))

{

*// 파일이 존재하지 않으면 만든다.*

writer = File.CreateText(path);

}

else

{

*// 파일이 있을 경우 이어서 쓴다.*

writer = new StreamWriter(path, true);

}

return writer;

}

}

}