using System.IO;

using System.Collections.Generic;

using UnityEditor;

using UnityEngine;

public class BuildAssetBundles : EditorWindow

{

static BuildAssetBundles BABWindow;

static List<AssetBundle> listAssetBundle = new List<AssetBundle>();

static string buildVersion;

static BuildTarget buildTarget;

Vector2 scrollPos;

static bool isStreamingAssetsPath;

[System.Serializable]

public class AssetBundle

{

public string name;

public bool isBuild;

public AssetBundle(string \_name, bool \_isBuild = true)

{

this.name = \_name;

this.isBuild = \_isBuild;

}

}

[MenuItem("Window/Build AssetBundles")]

static void Open()

{

if (BABWindow == null)

BABWindow = CreateInstance<BuildAssetBundles>();

buildVersion = PlayerPrefs.GetString("version", "1.0");

buildTarget = EditorUserBuildSettings.activeBuildTarget;

isStreamingAssetsPath = false;

listAssetBundle.Clear();

string[] assetBundleNames = AssetDatabase.GetAllAssetBundleNames();

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

listAssetBundle.Add(new AssetBundle(assetBundleNames[i]));

BABWindow.Show();

}

private void OnGUI()

{

if (EditorApplication.isPlaying == true)

{

PlayerPrefs.SetString("version", buildVersion);

Close();

}

buildVersion = EditorGUILayout.TextField("Version: ", buildVersion);

buildTarget = (BuildTarget)EditorGUILayout.EnumPopup("Build Target:", buildTarget);

isStreamingAssetsPath = EditorGUILayout.ToggleLeft("Output StreamingAssetsPath", isStreamingAssetsPath);

DrawUILine(Color.gray);

EditorGUILayout.LabelField("Asset Bundles");

++EditorGUI.indentLevel;

EditorGUI.BeginChangeCheck();

EditorGUILayout.BeginHorizontal();

float Height = 20 \* listAssetBundle.Count;

if (Height > 600)

Height = 600;

scrollPos = EditorGUILayout.BeginScrollView(scrollPos, GUILayout.Width(400), GUILayout.Height(Height));

for (int i = 0; i < listAssetBundle.Count; ++i)

{

listAssetBundle[i].isBuild = EditorGUILayout.ToggleLeft(listAssetBundle[i].name, listAssetBundle[i].isBuild);

}

EditorGUILayout.EndScrollView();

EditorGUILayout.EndHorizontal();

if (EditorGUI.EndChangeCheck())

{

*// Change Check*

}

--EditorGUI.indentLevel;

using (new EditorGUILayout.HorizontalScope())

{

if (GUILayout.Button("Select All"))

{

for (int i = 0; i < listAssetBundle.Count; ++i)

listAssetBundle[i].isBuild = true;

}

if (GUILayout.Button("Unselect All"))

{

for (int i = 0; i < listAssetBundle.Count; ++i)

listAssetBundle[i].isBuild = false;

}

}

EditorGUILayout.Space();

if (GUILayout.Button("Build Selected Bundles") == true)

{

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

for (int i = 0; i < listAssetBundle.Count; ++i)

{

if (listAssetBundle[i].isBuild == true)

buildBundles.Add(listAssetBundle[i].name);

}

if (buildBundles.Count > 0)

BuildSelectedAssetBundle(buildBundles.ToArray());

}

*//EditorGUILayout.Space();*

DrawUILine(Color.gray);

*//GUILayout.FlexibleSpace();*

if (GUILayout.Button("Build All Bundles") == true)

{

BuildAllAssetBundles();

}

if (BABWindow != null)

BABWindow.maxSize = BABWindow.minSize = new Vector2(400, Height+180);

}

void OnDestroy()

{

PlayerPrefs.SetString("version", buildVersion);

}

public static void DrawUILine(Color color, int thickness = 2, int padding = 10)

{

Rect r = EditorGUILayout.GetControlRect(GUILayout.Height(padding + thickness));

r.height = thickness;

r.y += padding / 2;

r.x -= 2;

r.width += 6;

EditorGUI.DrawRect(r, color);

}

public void BuildSelectedAssetBundle(string[] \_bundleNames)

{

if (\_bundleNames == null)

return;

string assetBundleDirectory = "AssetBundles/" + buildVersion + "/" + buildTarget;*//에셋번들의 파일경로.*

if (isStreamingAssetsPath == true)

assetBundleDirectory = UnityEngine.Application.streamingAssetsPath;*//에셋번들의 파일경로.*

if (!Directory.Exists(assetBundleDirectory)) *//해당 파일이 있는지 확인하고 없다면 새롭게 생성.*

{

Directory.CreateDirectory(assetBundleDirectory);

}

AssetBundleBuild[] buildBundles = new AssetBundleBuild[\_bundleNames.Length];

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

{

string[] names = \_bundleNames[i].Split('.');

buildBundles[i].assetBundleName = names[0];

if (names.Length > 1)

buildBundles[i].assetBundleVariant = names[1];

buildBundles[i].assetNames = AssetDatabase.GetAssetPathsFromAssetBundle(\_bundleNames[i]);

}

BuildPipeline.BuildAssetBundles(assetBundleDirectory, buildBundles, BuildAssetBundleOptions.ChunkBasedCompression, EditorUserBuildSettings.activeBuildTarget);

}

*//유니티의 상단 메뉴 창, Assets 메뉴 가장 하단에 Build AssetBundles 라는 항목을 추가. 그 항목을 누르면 아래의 함수가 실행된다.(실행중이 아닌 에디터 편집)*

[MenuItem("Assets/Build AssetBundles")]

public static void BuildAllAssetBundles()

{

BuildAllAssetBundles(null, null);

}

public static void BuildAllAssetBundles(string target, string version)

{

if (target == null)

{

if (buildTarget == 0)

buildTarget = EditorUserBuildSettings.activeBuildTarget;

}

else

{

if (target.ToLower().CompareTo("android") == 0)

buildTarget = BuildTarget.Android;

else if (target.ToLower().CompareTo("standalonewindows") == 0)

buildTarget = BuildTarget.StandaloneWindows64;

else if (target.ToLower().CompareTo("ios") == 0)

buildTarget = BuildTarget.iOS;

}

if (version != null)

buildVersion = version;

string assetBundleDirectory = "AssetBundles/" + buildVersion + "/" + buildTarget;*//에셋번들의 파일경로.*

if (isStreamingAssetsPath == true)

assetBundleDirectory = UnityEngine.Application.streamingAssetsPath;*//에셋번들의 파일경로.*

if (!Directory.Exists(assetBundleDirectory)) *//해당 파일이 있는지 확인하고 없다면 새롭게 생성.*

{

Directory.CreateDirectory(assetBundleDirectory);

}

*//모든 에셋번들을 빌드. 가운데 옵션이 None 이면, 디폴트로 에셋번들에 포함된 Prefab 과 연계된 것들(메테리얼 등)도 함께 첨부된다. 스크립트는 예외.(특수처리 요구, 기존에서 삭제하면 안됨)*

*//None 옵션의 경우 LZMA 압축한다. 가장 작은 용량이나, 하나를 사용하기 위해서 모든 압축을 풀어야만 한다. 다운 후에는 LZ4 로 압축된다.(모든 압축을 풀지 않고 일부만 사용가능)*

*//UncompressedAssetBundle 무압축. ChunkBasedCompression LZ4 압축. 청크 단위 압축으로 부분만 압축을 풀어 사용 가능.*

*//EditorUserBuildSettings.activeBuildTarget 는 현재 빌드 설정된 플랫폼을 타겟으로.*

BuildPipeline.BuildAssetBundles(assetBundleDirectory, BuildAssetBundleOptions.ChunkBasedCompression, buildTarget);

*//Refresh the Project folder*

AssetDatabase.Refresh();

}

}