**Easy Patcher Manual**

**Jihyun LEE**

**Easy Patcher Manual**

by Jihyun LEE

Published 2015

Copyright (c) 2015 L&K Logic Korea Co., Ltd.

This project is licensed under the terms of the GPL. This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. All product names are property of their respective owners. Such names are used for identification purposes only and are not indicative of endorsement by or of any company, organization, product, or platform.

이 프로그램은 자유 소프트웨어입니다. 소프트웨어의 피양도자는 자유 소프트웨어 재단의 GNU General Public License의 규정에 의해서 이 프로그램을, 개작된 2차적 프로그램과 함께 또는 개별적으로 재배포할 수 있습니다. 이 프로그램은 보다 유용하게 사용될 수 있으리라는 희망에서 배포되고 있지만 제품에 대한 어떠한 형태의 보증도 제공하지 않습니다. 보다 자세한 사항에 대해서는 GNU General Public License를 참고하시기 바랍니다. "제품의 모든 이름들은 원 제작자의 소유물입니다. 이런 이름들은 단지 구별을 위한 목적으로 사용되었으며, 어떤 회사, 조직, 제품 혹은 플렛폼의 혹은 그것들을 지지하는 목적으로 사용되지 않았음을 밝힙니다."(확인 필요)

**Table of Contents**

**I. 개요**

1. **메뉴얼 변경 사항 리스트**
2. **최종 목표**
3. **주요 기능들**
4. **주의사항**

**II. 설치하기**

1. **설치하기**
   1. **패키지 파일 받기**
   2. **프로젝트에 추가하기**

**III. 기본 사항**

1. **패치 흐름**
   1. **패치 파일 빌드 흐름**
   2. **클라이언트에서의 패치 흐름**
2. **패치를 위한 모든 파일의 구성**
   1. **로컬 저장소 구조**
   2. **툴 설정 파일**
   3. **패치 버전의 history 파일**
   4. **AssetBundle 리스트 파일**
   5. **클라이언트에서 사용되는 AssetBundle 버전 관리 파일**
3. **사용하는 방법**
4. **클라이언트에 적용하기**
5. **패치 테스트 해보기**
   1. **로컬에 가상 원격 저장소 설정하기**
   2. **"Easy Patcher" 설치하기**
   3. **데모 씬에 "Easy Patcher" 적용**
   4. **첫 빌드**
   5. **모바일 기기에서 테스트**
   6. **첫 패치 제작 및 테스트**

**IV. 도움말**

1. **도움말**

**Chapter 1. History**

**2015. 04. 21**

**Chapter 10. 패치 테스트 해보기 추가**

**2015. 04. 07**

**메뉴얼이 릴리즈 되었습니다.**

**Chapter 2. 최종 목표**

Unity 5.0 이상을 사용하는 프로젝트의 AssetBundle 패치 툴 및 클라이언트 패치 모듈을 제공합니다.

**Chapter 3. 주요 기능들**

**Easy Patcher**의 주요 기능은 아래 세가지로, 다음과 같습니다.

**버전 관리**

패치 버전을 지원되는 GUI를 통해 쉽게 생성, 관리할 수 있습니다.

**패치 리소스 제작**

버튼 하나로, 이전 버전 리소스와 비교하여 패치에 필요한 리소스들만 추려줍니다.

**클라이언트 패치 모듈**

당신의 클라이언트에 단 세 줄의 코드를 추가하여 손쉽게 사용할 수 있습니다.

**Chapter 4. 주의사항**

**Easy Patcher**는Unity 5.0 이상만 지원합니다. 그 이하 버전에서는 작동하지 않으니 주의하시기 바랍니다.

초기에는 0.0 버전이 세팅되어 있습니다. 0.0 버전은 존재하지 않는 버전이니 최소 0.1 버전을 시작 버전으로 세팅하여 진행 해주세요. "8.How to use"를 참고 해주세요.

**Chapter 5. 설치하기**

**5.1. 패키지 파일 받기**

아래의 사이트에서 **Easy Patcher**를 다운받으실 수 있습니다.

Unity Asset Store (free download) :

Github : https://github.com/blackwitch/EasyPatcher/

**5.2. 프로젝트에 추가하기**

1. project 창의 Assets 최상위 폴더에 마우스 오른쪽 버튼을 누르고 "Import package -> Custom" 을 누른다.

2. 다운받은 "EasyPatcher.unitypackage"를 선택한다.

3. 유니티 툴 상단 메뉴에 **EasyPatcher**가 추가된 것을 확인할 수 있다.

**Chapter 6. 패치 흐름**

**6-1. 패치 파일 빌드 흐름**

**6-2. 클라이언트에서의 패치 흐름**

**Chapter 7. 패치를 위한 모든 파일의 구성**

**7.1 로컬 저장소 구조**

**7.2 툴 설정 파일 : 파일명>> ABConfig.xml**

저장소 정보를 기록하는 용도로 사용됩니다.

<ToolConfig>

<Repository path="sample" />

</ ToolConfig >

- 저장소 이름의 히스토리는 기록되지 않습니다.

- 새로운 저장소를 지정한 경우, Save 버튼을 눌러 저장 후, 패치 툴 다이얼로그를 재시작 해야 합니다.

**7.3 패치 버전의 history 파일 : 파일명 >> Patch.xml**

지금까지 업데이트 된 모든 패치 정보를 저장하고 관리합니다.

<VERSIONS>

<PATCH LastVersion="VER\_XX\_XXX" />

<MAJOR value = "XX">

<MINOR value="XXX" />

<MINOR value="XXX" />

<MINOR value="XXX" />

</MAJOR>

<MAJOR value = "XX">

<MINOR value="XXX" />

</MAJOR>

</VERSIONS>

**7.4 AssetBundle 리스트 파일 : 파일명 >> Assets.xml**

해당 버전에 패치되는 파일 리스트를 관리합니다.

<AssetBundles>

<VERSION value="VER\_XX\_XXX" />

<CREATE>

<FILE name="filename\_1.unity3d" />

<FILE name="filename\_2.unity3d" />

.

.

</CREATE>

<MODIFY>

<FILE name="filename\_1.unity3d" />

<FILE name="filename\_2.unity3d" />

.

.

</MODIFY>

<DELETE> // 실제 클라이언트에서 삭제할 수는 없지만, 패치 확인용으로 정리된다.

<FILE name="filename\_1.unity3d" />

<FILE name="filename\_2.unity3d" />

.

.

</DELETE>

</AssetBundles>

**7.5 클라이언트에서 사용되는 AssetBundle 버전 관리 파일 : 파일명 >> ABVerList.xml**

AssetBundle을 다운받은 클라이언트에서 관리되는 AssetBundle별 버전를 관리하는 파일입니다.

<AssetBundleVersion>

<FILE name="filename\_1.unity3d" version="1"/>

<FILE name="filename\_2.unity3d" version="2"/>

.

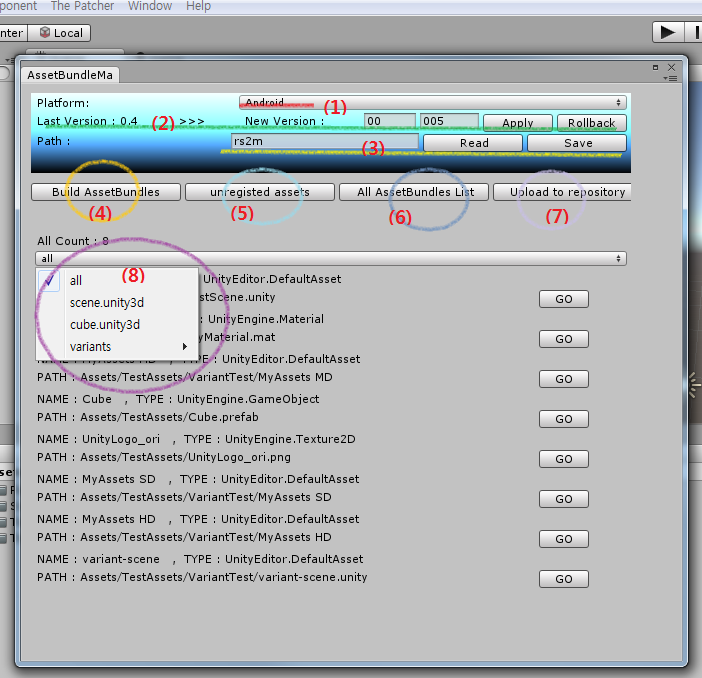
.

.

</AssetBundleVersion>

해당 파일이 업데이트 되면 attribute version값이 1이 증가하게 됩니다.

**Chapter 8. 사용하는 방법**



(1) 타겟 플렛폼을 지정할 수 있습니다.

(2) 버전정보를 확인, 지정할 수 있습니다.

앞의 "Last Version" 정보가 현재 패치가 적용된 버전이며, 뒤쪽의 "New Version"이 다음 버전을 지정할 수 있는 곳입니다.

직접 수정이 가능하며, 번호가 이어지지 않아도, **Easy Patcher**는 앞/뒤 버전을 연결하여 처리할 수 있습니다.

"Apply" 버튼을 누르면, "New Version"의 값이 "Last Version"값으로 적용되며, "New Version"은 새롭게 갱신됩니다. **단, 아직 패치 정보가 모두 갱신된 것은 아닙니다. 이 버튼은 다음 버전을 위한 준비를 진행하는 과정일 뿐입니다. 버전 번호를 바꾸고, 변경된 파일들을 저장할 공간을 마련하는 역할입니다.**

"Rollback"은 로컬 저장소에 생성된 각종 정보를 다시 이전 버전에 자동으로 맞춰줍니다. "Last Version"에는 이전에 지정했던 번호가 다시 지정될 것이며, "New Version"에는 "Last Version"에 있던 번호가 다시 돌아올 것입니다. 내부적으로는 로컬 저장소에 생성한 마지막 버전의 모든 파일을 삭제하고, 이전 버전으로 돌립니다. **주의하세요. 이 작업을 취소할 수 있는 방법은 없습니다.**

(3) 패치를 위한 로컬 저장소 명을 기록합니다. **추후 패치를 위한 저장소의 root 명도 이와 같아야 합니다. 신중하게 결정하세요.**

**"Save"을 눌러야 기록됩니다.** "Save"버튼을 누르지 않은 상태에서 이전값으로 돌리고 싶다면 "Read"버튼을 다시 눌러주세요.

**새로운 이름으로 변경했다면 "Save"를 누른 후 다이얼로그를 재시작 해주세요. Easy Patcher**는 당신이 지정한 새로운 이름의 저장소와 버전 정보를 재구성하여 다음 작업을 진행할 준비를 할 것입니다.

(4) 당신이 지정한 AssetBundle들을 빌드합니다. 이 작업은 단순히 빌드를 진행할 뿐, 로컬 저장소에 어떤 영향을 미치지 않습니다.

(5) **"Project" 탭의 최상위 폴더인 "Assets"을 마우스로 선택 후 버튼을 누르세요.** (특정한 폴더의 리소스만을 확인하길 원한다면 해당 폴더를 선택 후 버튼을 누르시면 됩니다.) 이 작업은 해당 폴더의 리소스 중 AssetBundle로 지정되지 않은 리소스가 어떤 것들이 있는지를 (8)번 영역에 출력해주고 해당 리소스의 "GO"버튼을 누르면 Inspector 창에 출력하여 확인/수정 할 수 있도록 해줍니다.

(6) **"Project" 탭의 최상위 폴더인 "Assets"을 마우스로 선택 후 버튼을 누르세요.** (특정한 폴더의 리소스만을 확인하길 원한다면 해당 폴더를 선택 후 버튼을 누르시면 됩니다.) 이 작업은 해당 폴더의 리소스들이 어떤 AssetBundle의 이름으로 포함되어 있는지 리스팅 해주며, (8)에 보이는 것처럼 상단에 AssetBundle 이름 리스트를 선택할 수 있게 하여 특정 AssetBundle에 포함된 리소스만 볼 수 있도록 지원해 줍니다.

(7) 빌드를 진행하고, (1) ~(3)에 지정된 정보를 바탕으로 로컬 저장소에 패치 데이타를 업데이트 합니다. **주의하세요. 새로운 패치를 준비중에 있다면 반듯이 (2)의 "Apply"버튼을 눌러 새로운 패치 데이타가 저장될 장소, 버전 정보를 준비 한 후 이 버튼이 진행되어야 합니다.**

**이 작업이 정상적으로 진행된다면, 로컬 저장소에는 이전 버전의 전체 버전이 "latest\_VER\_XX\_XXX"와 같은 이름으로 백업되어 있습니다. 잘못 작동했을 경우 이 폴더의 파일들을 사용해서 복구를 진행하시기 바랍니다.**

**Chapter 9. 클라이언트에 적용하기**

EasyPatcher.unitypackage를 당신의 프로젝트에 Import했다면 대부분의 준비를 끝난 상태입니다.

이제 세 가지 작업을 진행하면 패치를 위한 준비가 마무리 됩니다.

1. EasyPatcherprefab.prefab을 Hierarchy에 등록하세요. 이는 가능한 당신의 프로젝트 첫번째 씬에 포함되는 것이 좋습니다.

2. 패치가 진행될 시점을 찾아 아래 코드를 삽입하세요. 원격 저장소의 주소는 적절히 변경해야 합니다.

EasyPatcher patcher = GameObject.Find ("EasyPatchprefab").GetComponent< EasyPatcher > ();

patcher.SetRepogitory("your repository address"); // ex) http://localhost/

yield return patcher.StartCoroutine (patcher.startPatch ());

3. **Easy Patcher** 는 패치가 진행되는 몇가지 정보를 당신에게 제공합니다. EasyPatcher.cs 파일 상단을 확인하세요.

- PatchProgress : 패치가 몇 % 진행되었는지를 확인할 수 있습니다. 백분률로 제공됩니다.

- PatchMessage : 현재 어떤 파일의 패치가 진행되는지를 확인할 수 있습니다.

- PatchLastLog : 패치에 심각한 문제가 발생할 경우 이 메세지를 통해 확인하실 수 있습니다.

**- PatchErrorCount : 이 변수는 패치 중 문제가 얼마나 발생했는지 확인할 수 있으며, 1개 이상 발생했을 경우 클라이언트의 버전 정보를 갱신하지 않습니다. 패치 완료 후 꼭 확인하시기 바랍니다**

- Patcher.isDone() : 이 함수로 패치가 완료되었는지를 확인할 수 있습니다. 제공되는 Patcher의 기본 함수 (startPatch)가 Coroutine으로 되어 있기 때문에 필요한 상황에서만 사용하시기 바랍니다.

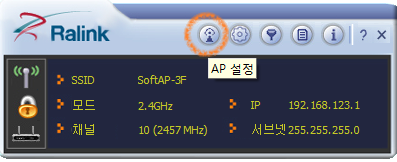
**Chapter 10. 패치 테스트 해보기**

**10.1 로컬에 가상 원격 저장소 설정하기**

패치 테스트를 위해 사용할 가상의 원격 저장소를 설치해 보겠습니다. 만약 테스트로 사용할 원격 저장소를 별도로 구성되어 있다면, 10.2로 넘어가세요. 아래는 자신의 PC로 Access Point를 만들어 자신의 핸드폰을 구성한 AP로 접속 후 모바일 기기에서 패치를 테스트 할 수 있도록 하기 위한 과정입니다.

테스트를 위해 쉽게 구할 수 있는 (AP로 활용 가능한) 무선 공유기를 준비하세요. 저는 ipTime N300UA 제품을 사용해서 테스트 했습니다. AP모드 지원하고 가격은 2015년 4월 현재 17,000원 정도에 구매 가능합니다. AP모드 지원한다면 어떤 제품이든 큰 문제 없습니다.

제품을 자신의 PC에 연결 후 AP모드로 전환합니다. ipTime 유틸리티에는 아래와 같은 위치에 AP 설정 버튼이 존재합니다.

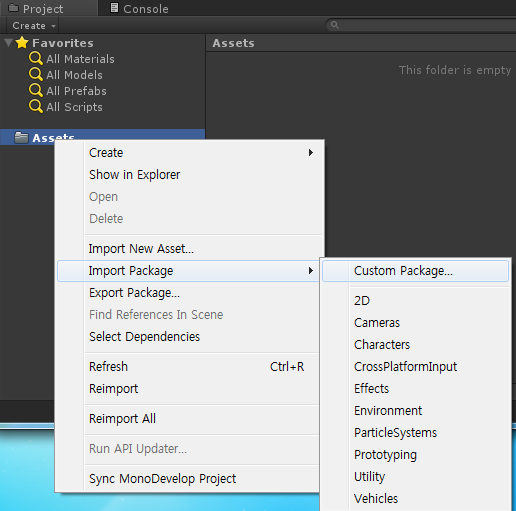
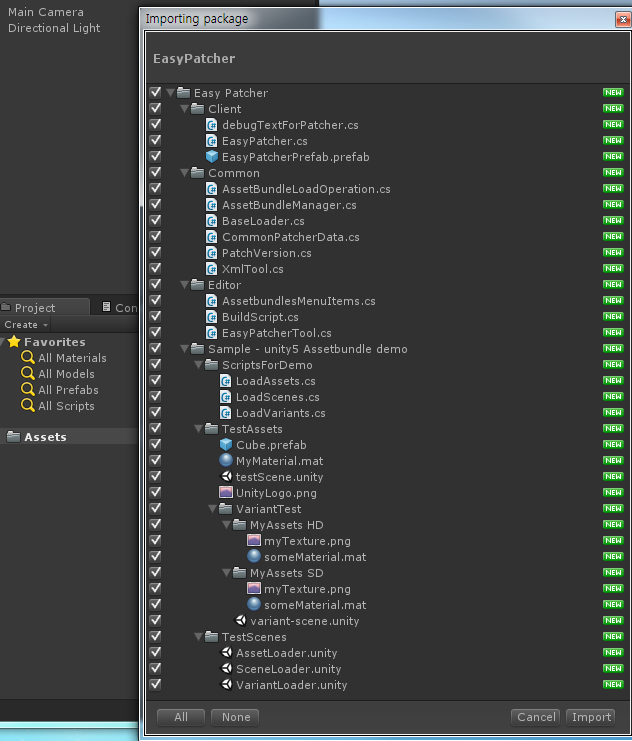


이 기능을 통해 AP를 설정하시고 모바일 기기에서 wifi 설정화면을 보시면, 설정한 AP로 연결할 수 있습니다.

이제 디바이스에서 패치 테스트를 할 수 있는 기본 환경이 갖추어졌습니다. 다음으로 툴에서 패치를 진행해 보겠습니다.

**10.2 "Easy Patcher" 설치하기**

Unity5를 기동하여 빈 프로젝트를 하나 만들어 주세요. 우선 "EasyPatcher"를 Import 해주시기 바랍니다. 아래 그림을 참고하세요.

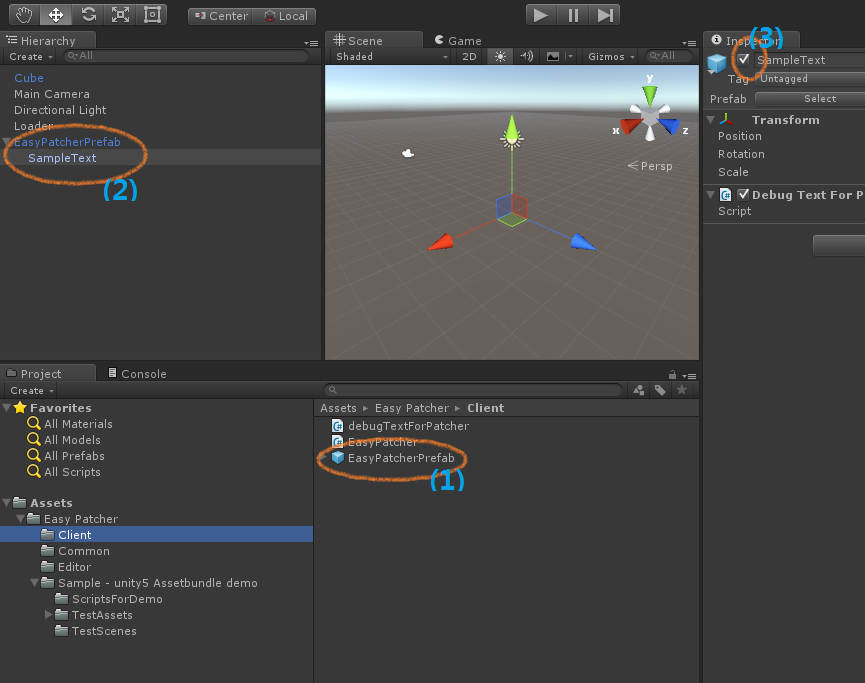


그리고 프로젝트 세팅에서 자신의 디바이스에 맞은 OS로 맞춰주시기 바랍니다.

**10.3 데모 씬에 "Easy Patcher" 적용**

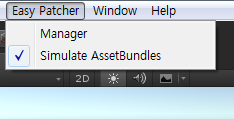
Project 탭에서 "Assets" ->"Easy Patcher" -> "Sample - unity5 Assetbundle demo" -> "TestScenes"안의 Scene파일인 "AssetLoader"를 더블 클릭하여 해당 씬을 오픈합니다.

프로젝트 탭에서 "Easy Patcher" 아래 탭 중 Client 폴더 안에 EasyPatcherPrefab을 Hierarchy에 등록해 주세요.



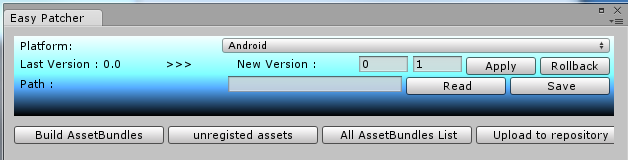
위 그림에서 (1) 위치의 프리팹을 (2) 위치로 드래그하여 등록하시면 됩니다. 이 프리팹의 역할은 기동 시 지정한 저장소에서 데이터를 다운받아 패치를 하는 역할이며, 이 프리팹에 포함된 sampleText는 패치가 진행되는 과정을 볼 수 있는 샘플입니다. 실제 프로젝트에서는 (3) 위치의 flag를 꺼두시기 바랍니다.

이제 "Easy Patcher" 툴을 기동해 보겠습니다. 메뉴 위치는 유니티 툴 상단의 메뉴에 추가되어 있습니다.



"Simulate AssetBundles"는 툴에서 AssetBundles 을 사용하는 코드들을 위해 존재하는 메뉴인데, 빌드된 AssetBundles을 읽어 처리하는 것이 아닌, Assets에 포함된 리소스에 직접 접근합니다. 그래서 AssetBundle 빌드와 상관없이 Asset자체가 바뀌어 있으면 변경된 리소스가 적용되도록 되어있습니다. 툴을 위한 시뮬레이션 기능이니 오해 없으시기 바랍니다.

이제 메뉴 "Manager"를 클릭하여 툴을 기동해 보시기 바랍니다.



툴을 기동하면 다이얼로그가 뜨며, 상단에는 Chapter 8에서 언급한 메뉴가 눈에 띕니다.

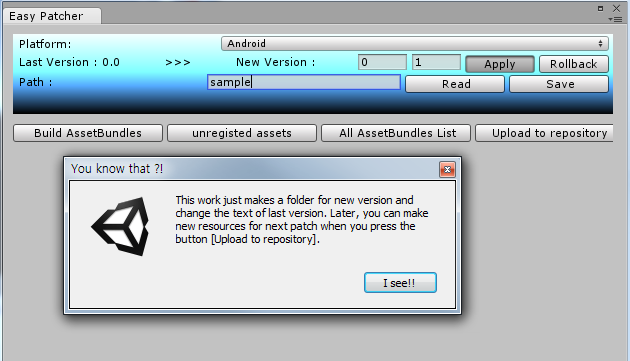
*\* 참고 : 지금 상태에서는 어떤 플렛폼으로 빌드가 되는지, 로컬 저장소는 어디로 할지를 결정한 상태가 이니기 때문에 두 가지 에러가 콘솔창에 보일 수 있습니다. 이는 무시할 수 있습니다.*

**10.4 첫 빌드**

초기에는 지정할 수 있는 플렛폼이 세 가지로 고정되어 있습니다. (Android, iOS, Windows) 만약 당신이 지원하고자 하는 플렛폼이 등록되어 있지 않다면, EasyPatcherTool.cs의 ShowWindow함수에 리스트인 window.listPlatform에 Add 함수를 통해 추가할 수 있습니다.

원하는 플렛폼을 설정했다면, 로컬 저장소를 지정해야 합니다. Path Lable 옆에 위치한 Input 컨트롤에 알맞은 이름을 입력하세요. 앞에 /를 붙이지 말아야 합니다. 이 폴더는 당신이 생성한 프로젝트 폴더에 포함될 것입니다. 입력했다면 오른쪽의 "Save"버튼을 눌러주세요. 그럼 "프로젝트폴더/지정한 이름"의 위치에 로컬 저장소를 제작할 준비가 되었습니다.

이제 최초 버전인 0.1 버전을 만들어 보겠습니다. 현재 버전은 0.0이며 어떤 데이터도 없는 상태입니다.

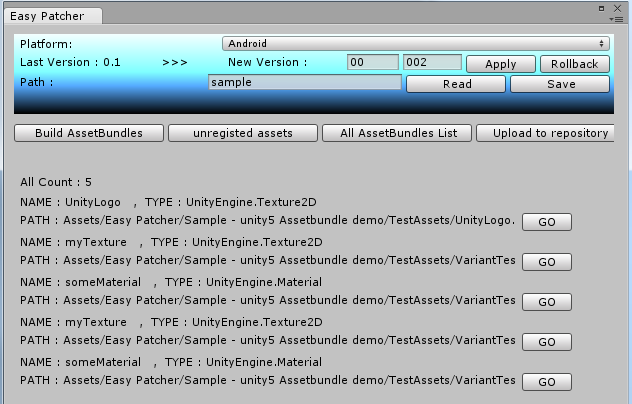


버전 정보를 변경하고 적용하는 일은 패치될 파일들의 정보를 저장하는 공간을 만드는 작업일 뿐입니다. 추후 "Upload to repository" 버튼을 눌러 실제 빌드를 적용해야만, 패치 될 파일들이 지금 만드는 폴더로 들어가게 됩니다. 유의하시기 바랍니다.

이후 프로젝트 폴더에는 지정된 저장소 이름의 폴더가 생성되었으며, 그 안에는 저장소 구조를 가진 폴더 및 xml 파일들이 생성되어 있음을 보실 수 있습니다.

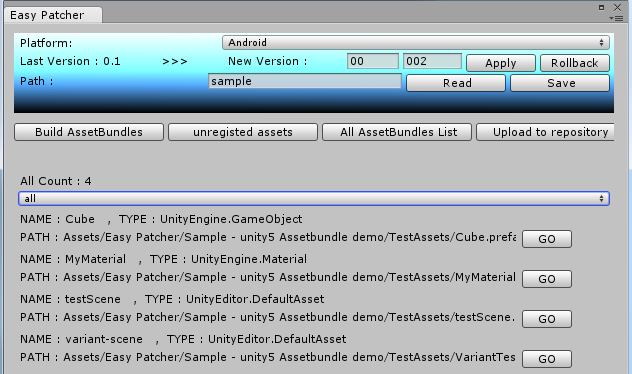
이제 "Build AssetBundles" 버튼을 눌러 AssetBundle을 빌드해 보겠습니다. console창에 별다른 에러가 없다면 정상적으로 빌드가 된 상태입니다. 프로젝트 폴더로 가보면 "AssetBundles"라는 폴더가 생성된 것을 확인할 수 있습니다.

"unregisted assets" 버튼은 지정한 폴더에 어떤 Asset들이 AssetBundle로 지정되지 않았는지 확인할 수 있습니다. Project 탭에서 "Assets" ->"Easy Patcher" -> "Sample - unity5 Assetbundle demo" 폴더를 선택 후, "unregisted assets" 버튼을 눌러보시면 아래와 같은 결과를 보실 수 있습니다.



지정된 폴더 안의 Assets 중 AssetBundle 이름이 지정되지 않은 리스트를 볼 수 있으며, GO 버튼을 누르시면 해당 리소스가 Inspector 창에 띄워집니다.

"All AssetBundles List"버튼을 누르면 AssetBundle로 지정된 모든 리스트를 아래와 같이 보실 수 있습니다.



중간에 위치한 콤보버튼을 누르시면 AssetBundle 이름 리스트가 보이며, 하나의 AssetBundle 이름을 선택한 후 해당 AssetBundle 이름이 지정된 리소스만 보실 수 있습니다.

마지막 버튼인 "Upload to repository" 버튼은 빌드와 로컬 저장소로 파일을 보내는 두 가지 역할을 동시에 합니다. (이미 빌드가 된 경우는 유니티툴 자체에서 추가 빌드를 하지 않습니다.)

실행 후 로컬 저장소에 가면, Assetbundle 파일들이 해당 버전 폴더(이 샘플에서는 VER\_00/001 폴더)에 저장되어 있으며, 각종 xml 파일들이 갱신되어 있음을 보실 수 있습니다.

이 과정이 진행될 때 기존의 latest 폴더를 백업합니다. 이는 추후 패치 제작 과정에 문제가 되었을 때를 대비해 "Rollback"을 지원하기 위함입니다. 로컬 저장소 안에는 latest 폴더가 존재하는데, 이 폴더에는 최신 버전의 데이터가 포함됩니다. (샘플에서 현재 버전이 0.2버전이므로 0.2 버전의 모든 AssetBundle 파일들이 이 폴더에 유지됩니다.) 그리고 "Upload to repository" 버튼으로 업데이트를 진행하는 과정에서 백업 폴더가 생성되는데 "latest\_VER\_현재버전" 이름으로 생성됩니다. 이는 "현재 버전이 생성되기 이전의 마지막 데이터"를 의미하는 이름으로, 현재 버전의 백업 버전이 아님을 유의하세요.

**10.5 모바일 기기에서 테스트**

이제 10.1에서 지정했던 원격 저장소에 로컬 저장소의 데이터를 수동으로 업데이트 해야 합니다. Unity의 ftp는 현재 anonymous 권한으로만 ftp에 접근해야 하기 때문에 원격 저장소로의 업데이트는 아쉽게도 구현할 수 없었습니다. (추후 구현 예정입니다.)

원격 저장소로 복사할 정보는 "patch.xml" 파일과 "VER\_"로 시작하는 폴더들입니다. 이 예제에서는 VER\_00 폴더가 해당됩니다.

원격 저장소에는 "툴에서 지정한 로컬 저장소 이름"/"플렛폼명" 의 폴더를 생성해야 하며, 이 하위 폴더에 해당하는 데이터들을 복사하시기 바랍니다.

이제 **EasyPatcher**를 클라이언트 최초 실행하는 코드에 아래 코드를 삽입하시기 바랍니다. 이 예제에서는 LoadAssets.cs 파일의 Start 함수를 아래와 같이 교체하시면 됩니다.

|  |
| --- |
| IEnumerator Start () {  yield return StartCoroutine( Initialize() );  EasyPatcher patcher = GameObject.Find("EasyPatcherPrefab").GetComponent<EasyPatcher> ();  EasyPatcher.SetRepogitory("**원격저장소/로컬저장소로 지정한 이름/**");  yield return patcher.StartCoroutine (patcher.startPatch ());  if (EasyPatcher.PatchErrorCount == 0) {  // Load asset.  yield return StartCoroutine (Load (assetBundleName, assetName));  // Unload assetBundles.  AssetBundleManager.UnloadAssetBundle (assetBundleName);  } else  yield break;  } |

이제 샘플을 빌드해서 모바일 기기에 인스톨 하시기 바랍니다. 그리고, 실행 전 10.1에서 세팅한 위치로 모바일 기기에서 정상적으로 접속되는지를 확인 후 실행해 보시기 바랍니다.

**10.6 첫 패치 제작 및 테스트**

지금까지 문제가 없었다면 이후는 더 간단합니다. 아래의 순서로 진행해 보세요.

1) "샘플프로젝트/Assets/Easy Patcher/Sample - unity5 Assetbundle demo/TestAssets/UnityLogo.png" 이미지를 열어 전혀 다른 이미지로 바꿔 보시기 바랍니다.

2) **EasyPatcher**툴을 열어 "Apply" 버튼을 눌러 다음 버전의 로컬 저장소를 만드세요. 만약 연속된 번호가 마음에 안든다면 당신이 원하는 버전으로 수정 후 "Apply"버튼을 누르시기 바랍니다. 굳이 연속된 버전이라도 상관없습니다.

3) "Upload to repository"버튼을 누른 후 로컬 저장소 위치로 가서 해당 버전의 정보를 확인해 보시기 바랍니다. cube.unity3d 관련 정보들만 해당 폴더 위치에 업데이트 된 것을 확인할 수 있습니다. 이 과정에서 추가 데이타 혹은 다른 문제가 발생했다면 "Rollback" 버튼으로 정보를 되돌리고 다시 2번부터 과정을 반복하면 됩니다.

4) 다시 원격 저장소에 수정된 파일들을 업데이트 하시기 바랍니다. 업데이트 할 데이타는 이전과 유사합니다. patch.xml 파일과 VER\_00/002 폴더입니다.

5) 모바일 기기에서 재실행 하시면 변경된 텍스쳐를 출력하는 결과물을 보실 수 있습니다.

**Chapter 11. 도움말**

**EasyPatcher**는 GPL 라이센스 하에 오픈된 프로젝트로, 공식적인 서포트를 지원하지 않습니다.

공개된 Github 포럼을 활용해 주세요.