**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. Summary**

1. **History**
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**

**IV. 도움말**

1. **도움말**

**Chapter 1. History**

**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 Patcher를 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 "Easy Patcher"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**

에셋번들을 다운받은 클라이언트에서 관리되는 에셋번들 별 버전를 관리하는 파일입니다.

<AssetBundleVersion>

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

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

.

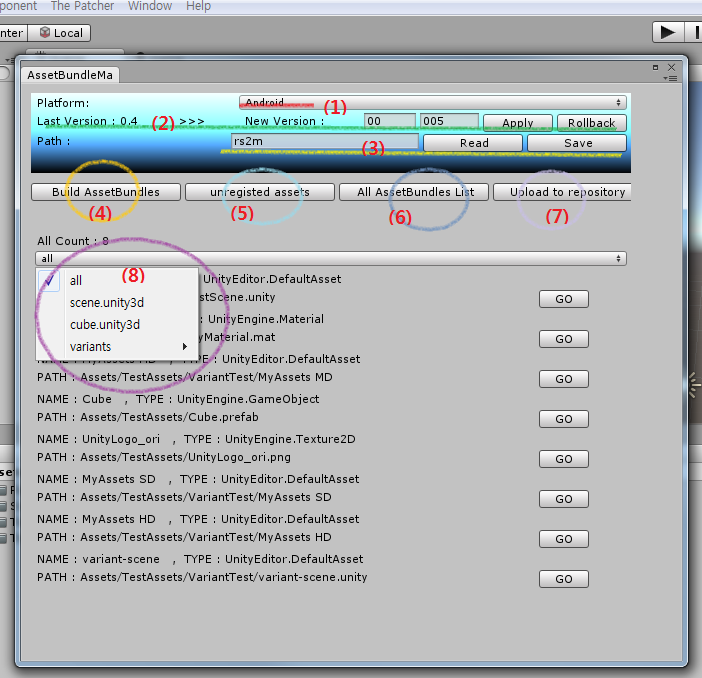
.

.

</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 에셋번들들을 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 에셋번들로 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 에셋번들의 이름으로 포함되어 있는지 리스팅 해주며, (8)에 보이는 것처럼 상단에 에셋번들 이름 리스트를 선택할 수 있게 하여 특정 에셋번들에 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. 도움말**

Easy Patcher는 GPL 라이센스 하에 오픈된 프로젝트로, 공식적인 서포트를 지원하지 않습니다.

공개된 Github 포럼을 활용해 주세요.