P.A.T.C.H. - Ultimate Patching System

Professional Autopatcher Tool, Customizable and Hackable

1. Introduction

Thank you for purchasing *P.A.T.C.H. - Ultimate Patching System*! I am proud to offer you the best solution (yes, I'm so optimistic! :P) for the game patching process!

2. What is it?

P.A.T.C.H. is a professional solution for application patching and updating. It can generate very small patches thanks to its included file binary diffing algorithm. What does it mean? It means that if you change only 5 bytes in your build, *P.A.T.C.H.* will create a patch that will change only those 5 bytes on users' builds, instead of downloading the entire edited file.
Your users will be able to maintain updated their application copies with no pain or headaches.
In addition, you will be able to create patches quickly with our included tool!

Index

3. Features

P.A.T.C.H. comes with many features:

- Binary diffing algorithm;
- Enables very small patches;
- Bandwidth saving;
- Checking for patches hash;
- Strong patches compression;
- Callbacks system to monitor what *P.A.T.C.H.* is doing;
- Encrypted configs;
- Linear and non-linear patches application;
- Unity GUI integrated;
- One-click-deploy;
- Upload builds and patches on FTP;
- Customizable settings;
- Outstanding flexibility;
- Comes with launcher source code;
- If download fails, it will take care to download again failed patch for customizable amount of attempts;
- If patch process fails, it will take care to rollback your game to previous version;
- Launching argument to avoid obsolete clients;
- Files download over HTTP, FTP and file system;
- Localization system integrated;

4. How to use it

P.A.T.C.H. is divided in two softwares: patches builder and launcher. Let's see how they work!

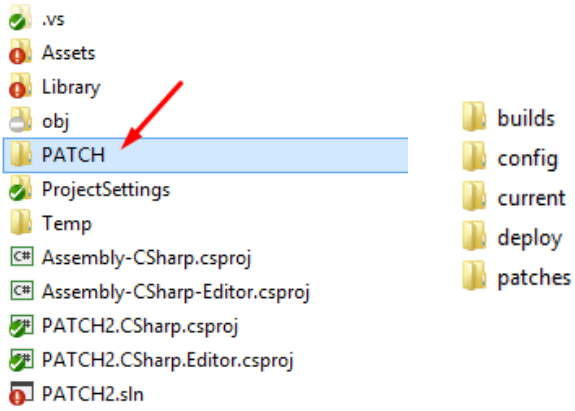
IMPORTANT!

First of all, remember to switch your API Compatibility Level (located in *Edit > Project Settings > Player > Other Settings*) to "***.NET 2.0***" instead of default "***.NET 2.0 Subset***".
P.A.T.C.H. uses cryptography, not included in Subset version of .NET 2.0 API!

4.1 Patches builder

Using the patches builder you can quickly and easily create your build versions or create a patch between two build versions. You will find it in “Window/P.A.T.C.H.” menu on Unity (or you will find it in *Assets/MHLab/PATCH/Source/PatchBuilder/PatchBuilderWinForms/PatchBuilderWinForms.zip* if you want the external version). In case you want to use the external one, extract it where you want (I recommend out of Unity project!).

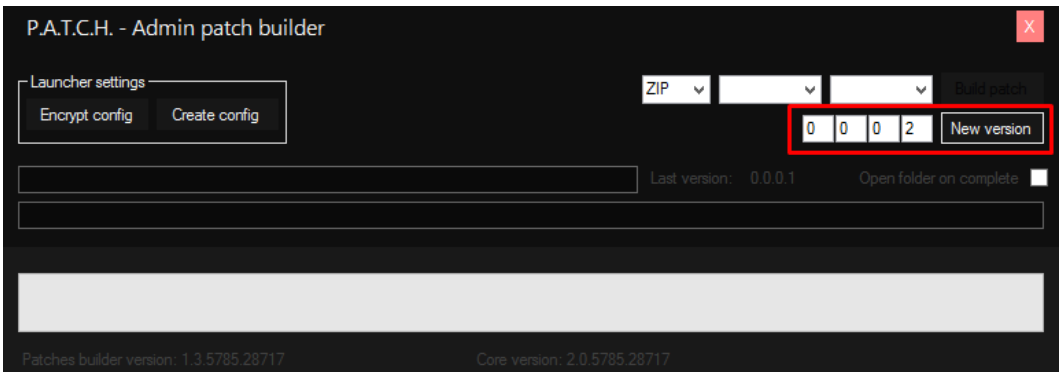
At first open patches builder will create PATCH directory and five main subdirectories in your Unity project root (or your executable root, if you are using standalone version).



Those directories will be our workspace; in them we will place our builds and patches.
builds: this directory will contain our processed builds
config: this directory will contain config files (for standalone version)
current: in this directory we will place our current build. We can directly build our project in this directory from Unity3D without problems.
deploy: in this directory we will find deployed build, composed by your game build and launcher build
patches: in this directory we will find computed patch archives and versions list, with archives hash.
logs: if some error occurs, patches builder will create a log directory with a log file to store information about errors

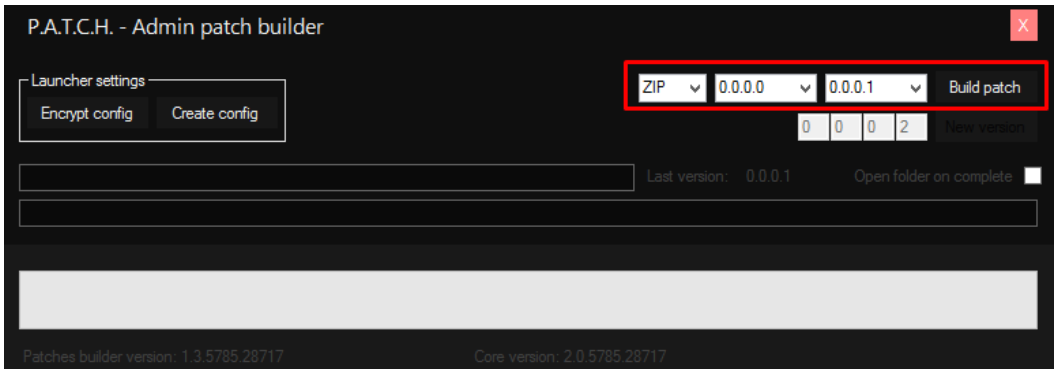
Patches builder is composed by three main parts: new version creator, patch builder and deploy manager.

If you already put your build in ***current*** directory, you can now create your new build version. By choosing version name and clicking on New version button, *P.A.T.C.H.* will create your new build in ***builds*** directory and it will add an encrypted version file to identify build version. You can check ***“Open folder on complete”*** if you want to open directly your target directory.

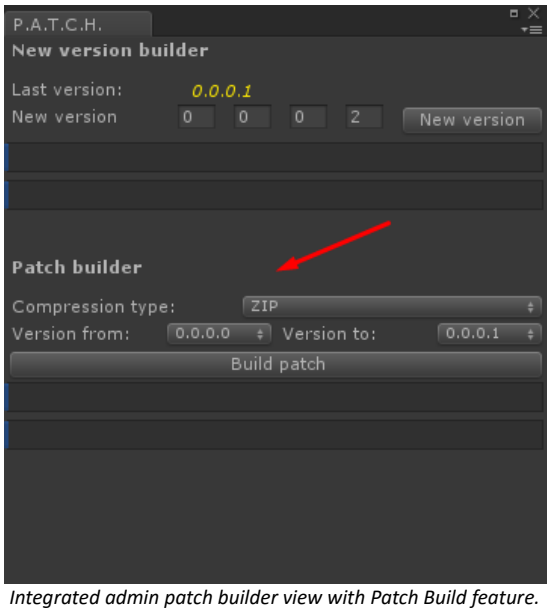


Standalone admin patch builder view with New Version feature.

When you created two or more builds, in patches builder you can also create a patch. Creating a patch is easy: just choose version-from and version-to in comboboxes, choose compression type and click Build patch button.
P.A.T.C.H. will create a new diff patch between those builds, it will inform you on what it is doing and, finally, it will create or update the versions list.

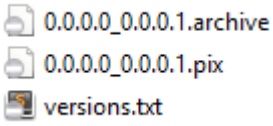


Standalone admin patch builder view with Build Patch feature.



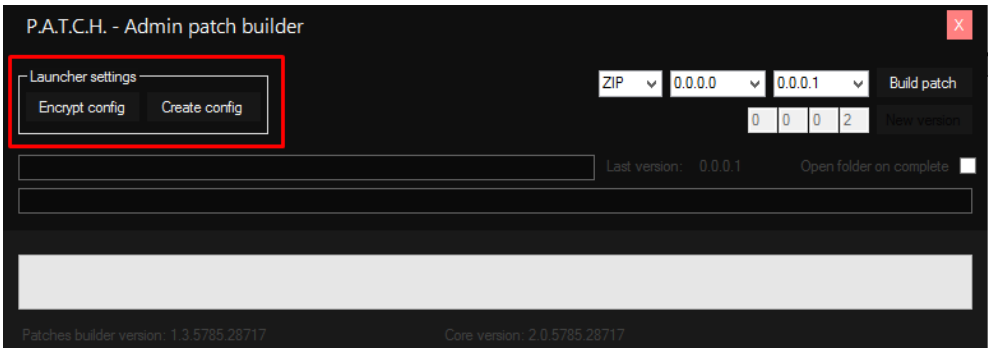
Integrated admin patch builder view with Patch Build feature.

When your patch is ready, you can open your ***patches*** directory to found your new, small, shiny patch archive and its indexer! You can check ***"Open folder on complete"*** if you want to open directly your target directory.



Now it's time to upload your ***patch archive***, your ***indexer*** and your ***versions.txt*** on your web space! My personal advice is to upload all archives on your web space, so your new users can download all patches!

You certainly noted that there are two more buttons in standalone patches builder, that are missing on Unity integrated one. Those buttons can create a config file, a file used by standalone version only. *So don't care about it if you want to use Unity integrated version, because you can set your credentials directly on a MonoBehaviour script.*



Standalone admin patch builder view with Config generation features.

Now click on Create config button: *P.A.T.C.H.* will inform you that it created a plain config file in ***config*** directory. Open this file with your favorite text editor:

```
<?xml version="1.0" encoding="utf-8"?>
<Settings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <VERSIONS_FILE_DOWNLOAD_URL>http://your/url/to/versions.txt</VERSIONS_FILE_DOWNLOAD_URL>
  <PATCHES_DOWNLOAD_URL>http://your/url/to/patches/directory/</PATCHES_DOWNLOAD_URL>
  <PATCH_DOWNLOAD_RETRY_ATTEMPTS>0</PATCH_DOWNLOAD_RETRY_ATTEMPTS>
  <LAUNCH_APP>Build.exe</LAUNCH_APP>
  <LAUNCH_ARG>default</LAUNCH_ARG>
  <ENABLE_FTP>false</ENABLE_FTP>
  <FTP_USERNAME>YourFTPUsernameHere</FTP_USERNAME>
  <FTP_PASSWORD>YourFTPPasswordHere</FTP_PASSWORD>
</Settings>
```

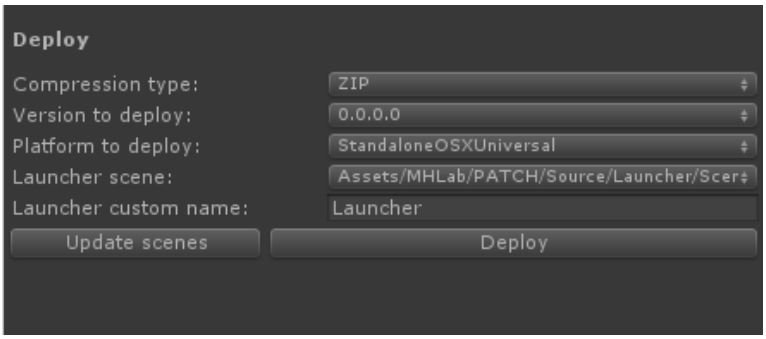
This is what you will find in it. You have to set these fields to fit with your settings!
VERSIONS_FILE_DOWNLOAD_URL: here is the URL at your uploaded *versions.txt*
PATCHES_DOWNLOAD_URL: here is the URL at your patches directory (do not forget "/" at the end!)
PATCH_DOWNLOAD_RETRY_ATTEMPTS: here is the amount of download retry attempts when one of them fails
LAUNCH_APP: here is the relative path to your application executable
LAUNCH_ARG: here is the argument to launch your application
ENABLE_FTP: if your PATCHES_DOWNLOAD_URL or your VERSIONS_FILE_DOWNLOAD_URL is referring to a FTP URL, you can allow credentials setting by switching to true this field
FTP_USERNAME: your FTP username
FTP_PASSWORD: your FTP password

Warning!!! These credentials will go on client side! They are encrypted, but you shouldn't trust your malicious users! For this reason, use FTP accounts with limitated rights!

When you complete this task, just come back in patches builder and click on Encrypt config button. It will generates an encrypted config file in ***config/Encrypted*** directory. You need to place it in same directory of your game launcher.

4.1.2 One-click-deploy

Since v2.0.3, P.A.T.C.H. includes deploy feature. You will find it in Patch Builder window:



Deploy feature allow you to create a deploy-ready build that contains selected game build and launcher build, with relative version file.
Compression type: select your compression method, your deploy-ready build will be compressed with selected algorithm
Version to deploy: select one of your already processed builds
Platform to deploy: select your favourite platform to deploy your launcher. Of course, if your build is for Windows platform (as example) your launcher must be built for Windows platform
Launcher scene: select your launcher scene to build. You have to add this scene to “Scenes in Build”, in Unity Build Settings
Launcher custom name: the Launcher's resulting name, you don't need to add extension to launcher's name, P.A.T.C.H. will deal with extensions for you. Add file extension only if P.A.T.C.H. can't manage it: when P.A.T.C.H. can't manage an extension, it simply add nothing as extension.
Update scenes: click it to update Launcher scene list
Deploy: click it to deploy your build with selected settings

Supported extensions:

- Android - .apk
- StandaloneLinux - .x86
- StandaloneLinux64 - .x86_x64
- StandaloneLinuxUniversal - .x86
- StandaloneOSXIntel - .app
- StandaloneOSXIntel64 - .app
- StandaloneOSXUniversal - .app
- StandaloneWindows - .exe
- StandaloneWindows64 - .exe
- Tizen - .tpk

4.2 Launcher

With the launcher you can perform two tasks: update users' clients and avoid incorrect use of your application. In other words, launcher will patch users' clients and will launch your application with a customizable argument: without it, your application will quit and users will not be able to use obsolete clients.

You can find launcher in *Assets/MHLab/PATCH/Source/Launcher folder*: here you will find all Launcher examples (also standalone version, if you want to use it, extract it in your build's root directory).

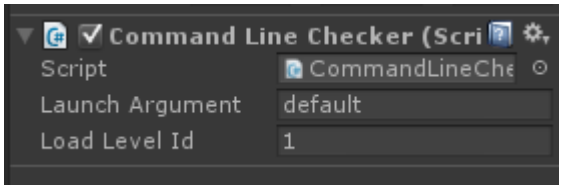
To set properly your game launcher your need to do some tasks; do not worry, nothing too complicated, I will guide you! :)

Standalone Launcher view on patch completed.

Unity's GUI integrated Launcher view on patch completed.

4.2.1 Standalone - Setup Unity Project

This is the easiest thing: open your Unity project and create a new scene. In this scene we don't need anything, just a single game object. So create one!
With *P.A.T.C.H.* comes a MonoBehaviour script called CommandLineChecker, add it to your created game object!



Nicely done!
In this script you can setup some options:

- ***LaunchArgument*** determines your launching argument; insert here exactly the string that you inserted in config file!
- ***LoadLevelId*** determines what level will be load if argument checking will be correct. You can load a menu scene or the first level, in example.

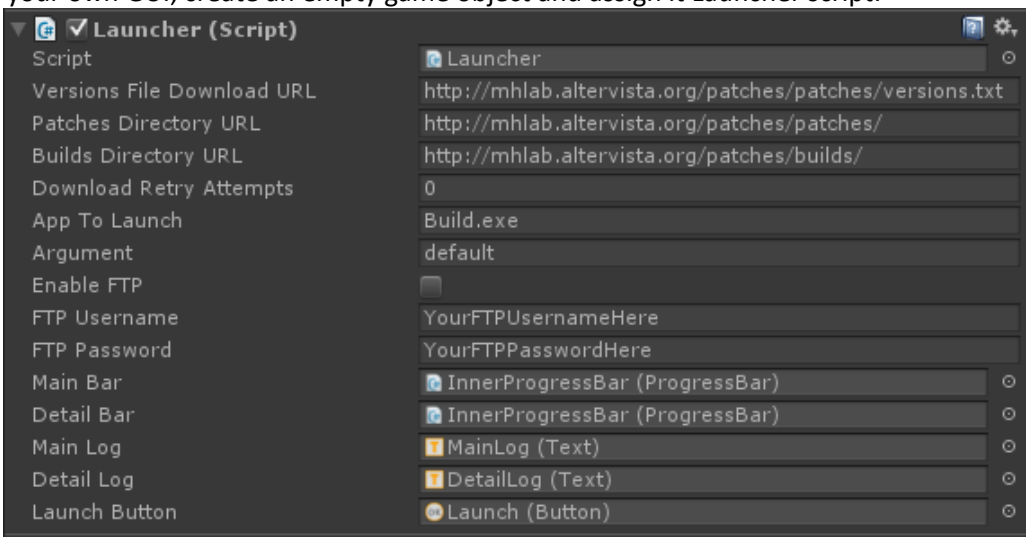
You do not need other stuff on your standalone launcher! That is it, you ask? Yes, that's it, sorry to disappoint you... :)
CommandLineChecker script will take care to close your app/game if LaunchArgument does not match or load your specified level!

4.2.2 Standalone - Customize Launcher

This part is a bit more laborious, but optional.
You will find Launcher project in *Assets/MHLab/PATCH/Source/Launcher/Scenes/LauncherWinFormsStyle/Launcher_src.zip*. Extract it in any folder (but out of Unity project, as I recommended!).
You can open it with Visual Studio and customize it at your pleasure!

4.2.3 Unity integrated – Setup Unity Project

With P.A.T.C.H. you can create your Launcher GUI directly with Unity's GUI system. It's simple: create your scene with your own GUI, create an empty game object and assign it Launcher script:



As you can see, you have to fill all configuration options in this script.
Also, you have to drag 'n drop some of your GUI component in this script: in this way P.A.T.C.H. can interact with them.
A ***ProgressBar*** prefab (and script) is attached to this package, so don't worry about it. :P
You can see an example of how it works on *Assets/MHLab/PATCH/Source/Launcher/Scenes/LauncherStyle1.*

5. F.A.Q.

When I create a build and put it in current directory, I need to include also launcher or version/config files?
No, you must not include those files in your build! You have to include them only in the first build that you send to your users, but never include them in builds that will be computed by patches builder!
For this reason, use One-click-deploy feature! It will create a deploy-ready build of your game, inclusive of launcher and version file!

Can I use your patching system for projects unrelated to Unity?
Yes, you can. *P.A.T.C.H.* is built to work well with any application. If your system can run your Launcher, you can use *P.A.T.C.H. to update your application*.

Can I develop my own launcher logic and skin?
Yes, you can. Just open launcher project or Unity scene and edit it.

Can I sell on Assets Store my own skins or logics?
Yes, you can. Develop what you want and sell it! It will help my product to gain a good visibility and help you to earn much money!

Can I use it with Linux or Mac?
Yes, you can. With Unity integrated version you can compile your build for each platform you want.
With standalone version it is a bit different: at this time, standalone *P.A.T.C.H.* comes with a precompiled Launcher that runs only on Windows. This doesn't mean that you can't run it on other platforms. :)
Attached to package there is full Launcher source code, you can open it with Xamarin/Mono and compile it targeting your favourite platform! This will make the trick! See Launcher Development section for more info.

6. Developers

6.1 Launcher development

If you want, you can deeply customize or hack launcher.
My lib exposes some classes that you can edit, in example SettingManager. It contains static members that you can modify at launcher startup.
This is the list (not complete):

```
public static string APP_PATH
public static string CURRENT_BUILD_PATH
public static string BUILDS_PATH
public static string PATCHES_PATH
public static string SIGNATURES_PATH
public static string FINAL_PATCHES_PATH
public static string LOGS_ERROR_PATH
public static string PATCHES_TMP_FOLDER
public static string LAUNCHER_CONFIG_GENERATION_PATH
public static char PATCHES_SYMBOL_SEPARATOR
public static string PATCH_VERSION_ENCRYPTION_PASSWORD
public static string PATCH_VERSION_PATH
public static string VERSIONS_FILE_DOWNLOAD_URL
public static string PATCHES_DOWNLOAD_URL
public static ushort PATCH_DOWNLOAD_RETRY_ATTEMPTS
public static string LAUNCH_APP
public static string LAUNCH_ARG
public static string LAUNCH_COMMAND
public static string LAUNCHER_CONFIG_PATH
```

In addition, there are some callbacks that you can set to manage certain aspects of patches application.

```
public void SetPatchBuilderOnFileProcessedAction(Action<string> action)...

public void SetOnFileProcessingAction(Action<string> action)...

public void SetOnTaskStartedAction(Action<string> action)...

public void SetOnTaskCompletedAction(Action<string> action)...

public void SetOnLogAction(Action<string, string> action)...

public void SetOnErrorAction(Action<string, string, Exception> action)...

public void SetOnFatalErrorAction(Action<string, string, Exception> action)...

public void SetOnSetMainProgressBarAction(Action<int, int> action)...

public void SetOnSetDetailProgressBarAction(Action<int, int> action)...

public void SetOnIncreaseMainProgressBarAction(Action action)...

public void SetOnIncreaseDetailProgressBarAction(Action action)...
```

Probably (certainly), you want to run your standalone Launcher on all platforms such as Mac, Linux, etc.
You can open your standalone Launcher project with Xamarin to compile it for these platforms:
http://xamarin.com/
You can also compile it with Mono: http://www.mono-project.com/docs/

6.2 Patches builder development

Standalone patch builder project is not included in this version, but you can create your own because MHLab.PATCH lib exposes the methods that you need to achieve this objective. Just check it out!

You can also develop launcher/patches builder skins or logics and sell them on the Assets Store, if you want!

7. Support

P.A.T.C.H. is already tested, but bugs can always find a way to annoy you. Feel free to contact me (and nag me to death ;]), I will solve your problem!

Email: *m4nu.91@gmail.com*
Skype: *manhunterita*
Twitter: *@MHLabSoftware*

Thank you again! :)

8. Changelogs & Roadmap

Next features to add:

- *Test and fix on Android and iOS*
- *Test and fix on Mac*
- ~~*Improve current download manager*~~ *(introduced in v2.0.3)*
  - ~~*Add download progress*~~ *(introduced in v2.0.3)*
  - ~~*Add download speed*~~ *(introduced in v2.0.3)*
  - *Add download restore after patcher closes*
  - *Add download pause*
- *File blacklisting or something like .gitignore*
- *Create self-update feature*
- *Create documentation for installer and create examples for it*
- *Create documentation for installation repairer and create examples for it*
- *Create in-game integrated patch process*
- ~~*Create automatic patches/builds FTP uploader*~~ *(introduced in v2.0.4)*
- *Create one-click-deploy for standalone version with standalone patcher*
- *Add additional compression methods for patches and builds (LZMA2, LZ4)*
- *Add command line interface*
- *On-the-fly patch for unused assets*
- *Create a multiplatform builds directories structure (so Win builds, Linux builds, etc)*
- ~~*Add button in Unity Menu for quick opening of PATCH folder*~~ *(introduced in v2.0.5)*
- ~~*Add option to close patcher after game start*~~ *(introduced in v2.0.5)*

v2.0.5:

- *Solved pending warnings in FULL version*
- *Added MHLab.PATCH.Settings namespace to Localizatron settings files, to avoid naming collisions*
- *Added button in Unity Menu to quickly open workspace folder*
- *Added flag to close Launcher on game start*
- *Added multiple flags to activate/deactivate features*
- *Adjustments to Launcher script inspector*
  - *All fields are divided by categories*
  - *Added tooltips to help users to better understand what each field can do*
- *IMPORTANT: greatly improved downloader code and its performances*
  - *Greatly improved download bandwidth usage*
  - *Greatly improved disk usage*
- *Solved missing Mono certificates validation for HTTPS download URLs*
- *Added build installer feature*
- *Added build repairer feature*

v2.0.4:

- *Adjustments to P.A.T.C.H.'s Editor GUI*
- *Upload to FTP added*
  - *Recursive directory creation*
  - *Patches and builds upload*
- *Added TAR archiving (uncompressed) for patches and deploy*
- *Added TARGZ compression for patches and deploy*
- *Downloader: solved a bug on downloading process for Unity version Launcher*

v2.0.3:

- *One-click-deploy introduced*

- *Deploy folder structure created and relative settings added*
- *Solved a weird path-relative bug on Mac systems*
- *Solved LaunchArg parameter override bug*
- *FatalError is now called when archive download fails or archive hash checking fails*
- *FatalError is now called after a rollback*
- *Cleaning up of single archive file when download fails or hash checking fails*
- *Detection of OS related files like ".DS_Store" or "desktop.ini"*
- *Important: solved OS reliant patch indexer bug, regenerate your patches!*
- *Added "Roadmap" in this document*
- *Changed CommandLine script to fit with new Unity 5.3 API (about SceneManager)*
- *Improved download manager, now it works also on "file://" protocol (so on file system)*
  - *It offers new information about current download*
  - *Download progress is now available*
  - *Download speed is now available*
- *Added package with Unity 4.6.9 and Unity 5.3.1 to improve compatibility*

v2.0.2:

- *I/O fix: workspace cleaning up during patches application*

v2.0.1:

- *Solved ZIP compression bug on Mac and Linux systems*
- *Solved linear patches application loop*
- *Added missing localization files*
- *Solved three pending warnings*

v2.0.0:

- *Now Unity integrated*
- *Localized thanks to Localizatron (it's my localization software)*
- *All core functions was re-coded*
- *Patches are now more small*
- *Changed file hashing algorithm*
- *New callbacks system added*
- *New versions standard format*

v1.2p1:

- Added a new event to check when current build version changes
- API: PatchManager class now exposes GetCurrentVersion method
- Added a reminder of current build version in Launcher
- Patching process now will apply ever the latest patch available, so if you have 0.1>0.2 and 0.1>0.3, P.A.T.C.H. will apply 0.1>0.3 patch, skipping 0.1>0.2
- FIX: Version constructor now strips correctly "\r" char from remote hashes
- Added some new LogEvents to describe better what is happening during long processing

v1.2:

- Added new in-game GUI that informs your users when checking of launching arguments fails
- Added forced run-to-admin behaviour to Launcher and Patches Builder for Windows
- FIX: now Launcher will apply patches in linear way and non-linear way both, without Launcher restarting
- FIX: added a "s" char in versions.txt example URL, in this way users can't be wrong
- Added support for FTP credentials to download files over Files Transfer Protocol.

v1.1p1:

- FIX: changes to file hashing function is now correctly applied in patching process

v1.1:

- *FIX: patching process run now on a new thread*
- *FIX: delete file process now retry to delete files for a customizable amount (to avoid deadlocks on file deleting)*
- *FIX: PatchFailed event now doesn't shutdown the patching thread before GUI updating*
- Added core version reminder and launcher version reminder to Launcher
- Added core version reminder and patches builder version reminder to Patches Builder
- Added patch rollback feature: if patch process fails all changes will be discarded to avoid a build corruption
- Added generation of patch files indexer in patch building process
- Added hash validation for patched files