CRAFTY

v1.0.4 Documentation

Contents:

Important Note:

We strongly advice on cleaning Crafty from the project (that also include the in project folder created by Crafty) before downloading new version.

1. Introduction

Welcome to the tutorial and presentation section of the Crafty program, made by Developers for Developers.

Crafty is a two in one program that allows you to create patches from comparing two of your projects on the binary data level. Patches made from" Patchie" part of the Crafty will be able to change, replace and delete data that you don't want or need in your projects. The second part of the Crafty program is" Launchie" it's an in game Launcher made from your own GUI design. No matter how complicated it is you have as much power over it as your knowledge on Unity GUI or other program like NGUI, iGUI and so on. Together these programs will check for updates and have instant reaction when there is patches needed to be downloaded.

Crafty itself is a program that will work on Windows, Mac and Linux and in the future we plan to make it available for all mobile devices. We have under our sleeves many tricks and add-ons that we want to add to the Crafty: speeding it up, giving it new possibilities, and of course keep in mind that price of 50 $ will NOT be increase. As Indie developers we understand low income of developers, we respect that and we will make sure to keep price as low as possible while not taking away anything from Crafty Project.

NOTE: Crafty works on both Indie and Pro version of Unity from Unity 3.5.7 and all version of Unity 4+.

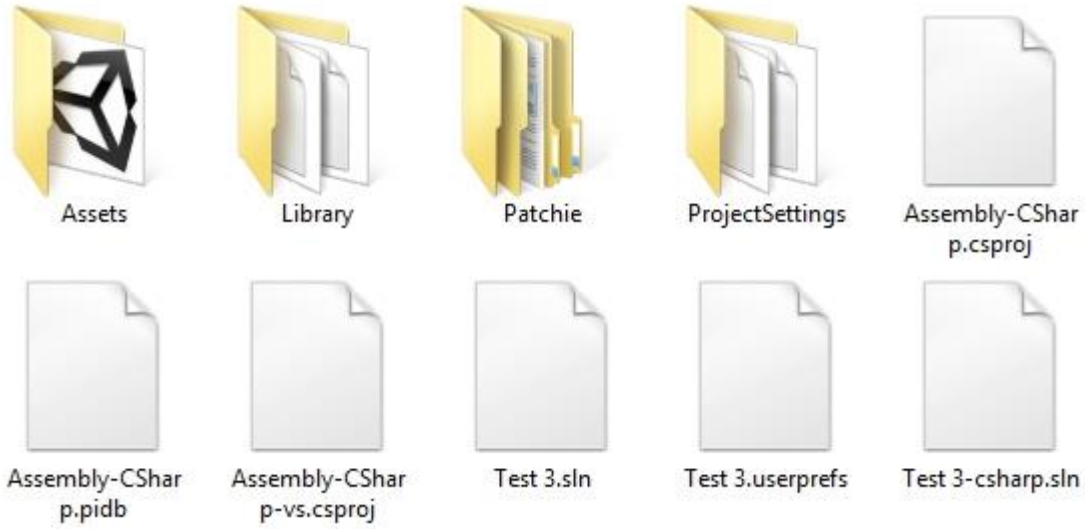
With that said let's move on to the tutorial part of t section.

2. Installing

The moment you will download Crafty it will be sent to the EDITOR folder please, do NOT move it from that place, it is there so that developed game will not copy the Crafty plug-in making loop problems while creating the patches.

Crafty is smart, it will create its own work place after you will turn it on for the first time (NOTE it will not work properly with a build in games if you don't turn on Patchie from the window first). It will create a folder, in that project folder that folder will be named" Patchie" so it will be easier for you to find, in it will be one more folder it will be called "_current ". Folder" Patchie" this is where the magic happens.

Patchie folder in the game project.



Test patches creation in patchie folder. (_output folder is the location of all your patches, more on that in the Patch Creation section).

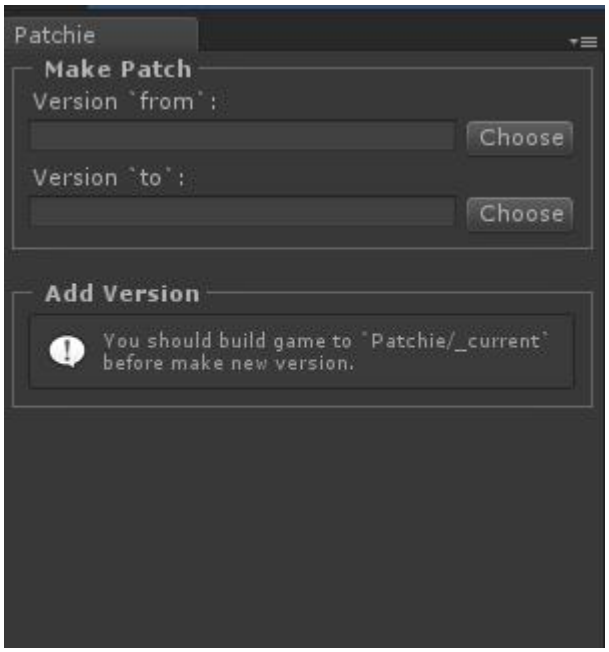
Remember to build your games (that you want to use for patch creation) in" _current" folder, so that while making patches Crafty will know that it's Unity Game.

Don't worry you can call your game whatever you want, you don't have to call it (for example) Penkura version. 1.0.0 BETA, you can just call it whatever you wish (we moved from that problem long time ago). After Patches will be made or when you will feel that the Project folder is getting too big for your taste just delete previous versions of the game. Only the Current version is important because from that version you will created update patches. More information's about the "_current" folder will be given in the next part of our tutorial (Patch Creation).

3. Patch Creation

Patchie itself is under the Window's button at the top of Unity, when you turn it on a window will appear, allowing you to create the gaming versions, and the overall patches (also the first time opening" Patchie" will create a Patchie folder in your project where all the data will be stored).

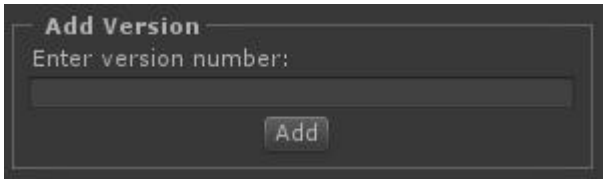
For Video version of this tutorial please click HERE.



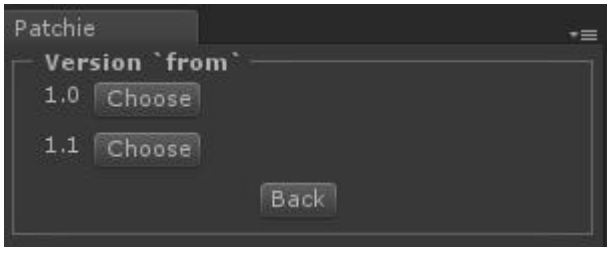
This is the Patchie window, as you see there is information that the game should be built in the " Patchie/_current " besides that there are two Versions Placements "From" and "To", that's the place where you will insert your build in game versions .

Side note:

Don't be worried we already created some security for you, so you won't be able to create two of the same versions or create any patches (or game version) without a name.



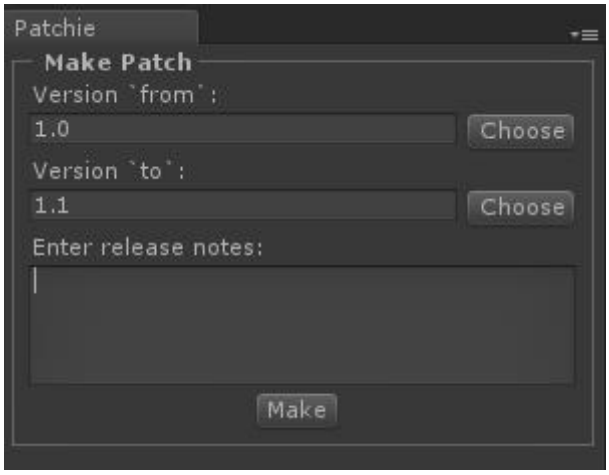
The moment you will build a new version of the game the Add Version will change to" Enter version number" this is the Version of your game. Tutorial we will call it 1.0 and 1.1 to make it easier to understand.



After creating two versions types of your game just press the Choose button, the window will change so that you can choose from all the versions you created, choose the two from which you wish to create the Patch. Remember the "FROM" version is the previous version and "TO" is the next version of your game.

Side note:

As it will be stated in the" Launcher creation" section, the version name of the patch needs to be the same as the version name in the launcher. This will be explained later on.



After choosing which versions you want to use to create the patch a new window will appear where you can write all your “Release notes”, those are the Patch changes that will appear in the game while player will download the patch, last part is to click Make Button and you are done.

Two blue lines will appear that will indicate how long you have to wait till your patch will be finished. your patch will be nicely com-press and will appear in the” Patchie” folder in your Project Files, there will be also a TXT file with your patch release note, it was made that way so you can change the in-formation, even after the patch creation. This is the End of the Tutorial Part your Patch is ready to be sent to the game. Let’s move on to the launcher creation.

4. Launcher Creation.

Launchie is a second part of the Crafty plug-in that allows you to create an in-game launcher that connects with your website or homemade server.

For Video version of this tutorial please click HERE.

Since Crafty 1.0.4 there is a prefab of Launchie in your Crafty folder.
This allows you to simple drag and drop Launchie in to your scene.

In the prefab you will have to important scripts connecting your prefab (launcher) with the Crafty. Note that it is important to keep both of the script in one object. Those scripts are, LaunchieGUI and LaunchieLogic. Thanks to this new solution you will be able to update Crafty without damaging your work put in to adjusting LaunchieGUI to your game/project.

After connecting Launchie with your scene (note that we strongly advice to connect Launchie to your main menu scene) you will have two empty bars to fill.

Url which represents address to your folder with patches on your website.

(Note that the link starts with http://) and Version which represents the current version of the game you are working on right now.

Keep in mind that the version in the Launchie Logic needs to be exactly the same as the version name you are using while creating the patch. Inside the script itself at the end there is a simple GUI script that is the main graphic script it uses build in Unity GUI functions. It means that 100 % of the power in Launchie

Logic is controlled by build in Unity GUI, giving you as much power as Unity allows.

That is the end of the Launchie Logic setup (for a basic users)

All information below is for more advance users that want to reprogram Launchie for more advance purposes.

Launchie(string url, string currentVersion) - constructor, it creates instance of Launchie class and setup location of patches and current version of game

```
// url and currentVersion can be hardcoded of taken from public vars
Launchie l = new Launchie( url, currentVersion );
```

bool Check() - method for checking server if there is any patches to download. After result is cached to reduce web requests.

```
bool can_download = l.Check();
```

bool ForceCheck() - same as above, but it force check for patches (don't use cached result)

```
// 1 web request
bool can_download = l.Check();
bool can_download2 = l.Check(); //will be same as can_download

// 2 web requests
bool can_download = l.Check();
bool can_download2 = l.ForceCheck(); //can be other than can_download
```

void DownloadReleaseNotes() - method for download release notes for patch

```
// because it's async function, you need use
// setOnProgress / setOnDone methods

// l.setOnProgress it's not needed, due to small size of Release Notes
l.setOnDone( OnDownloadReleaseNotesDone );
l.DownloadReleaseNotes();

---

void OnDownloadReleaseNotesDone()
{
  Debug.Log( "Release Notes:" + l.getReleaseNotes() );
}
```

double getProgress() - returns progress for current task (DownloadReleaseNotes, Download or Extract)

```
l.getProgress();
```

void Download() - method for download patch

```
// because it's async function, you need use
// setOnProgress / setOnDone methods

l.setOnProgress( OnDownloadProgress );
l.setOnDone( OnDownloadDone );
l.Download();

---

void OnDownloadDone()
{
  Debug.Log( "Patch" + l.getAvaliableVersion() + " has been downloaded." );
}

void OnDownloadProgress( double progress )
{
  Debug.Log( "Downloading patch" + l.getAvaliableVersion() + ": " + progress + "% com-
plete." );
}
```

void Extract() - method for uncompress patch

```
// because it's async function, you need use
// setOnProgress / setOnDone methods

l.setOnProgress( OnExtractProgress );
l.setOnDone( OnExtractDone );
l.Extract();

---

void OnExtractDone()
{
  Debug.Log( "Patch" + l.getAvaliableVersion() + " has been unpacked." );
}

void OnExtractProgress( double progress )
{
  Debug.Log( "Unpacking patch" + l.getAvaliableVersion() + ": " + progress + "% com-
plete." );
}
```

void Finish() - method for apply patch

```
// Finish isn't async function.
l.Finish();

// after patch you should exit game
Application.Quit();
```

string getAvaliableVersion() - returns highest version of available patch

```
l.getAvailableVersion();
```

string getReleaseNotes() - returns release notes

```
l.getReleaseNotes();
```

bool debug – this field is responsible for create crafty_error.txt files when something gone wrong

```
// if there will be error no file will be created
l.debug = false;

// create file on error
l.debug = true;
```

void setOnDone(OnDone handler) - sets OnDone event handler for DownloadRelease-Notes, Download and Extract methods

```
l.setOnDone( handlerForOnDone );

---

void handlerForOnDone()
{
  // do something when current task ( DownloadReleaseNotes, Download or Extract ) finish
}
```

void setOnProgress(OnProgress handler) - sets OnProgress event handler for DownloadReleaseNotes, Download and Extract methods

```
l.setOnProgress( handlerForOnProgress );

---

void handlerForOnProgress( double progress )
{
  // for ex. display progress of current task
}
```

void setOnError(OnError handler) - sets OnError event handler. OnError event is triggered regardless of value of debug field

```
l.setOnError( handlerForOnError );

---

void handlerForOnError( Exception ex )
{
  // handle any errors
}
```

An example can be found in package */Crafty/LaunchieGUI.cs*

5. How to setup your server

This is the easiest part, but even so we plan to also make a tutorial video on our YouTube page to help as much as possible.

You should only create *versions.txt* and enter any available patches `[OLD_VERSION][SPACE][NEW_VERSION]`, for ex.:

```
1.0 1.1
1.1 1.2
```

It means that there is 2 patches `1.0 -> 1.1` and `1.1 -> 1.2`.

But if you add `1.0 1.2` before first line:

```
1.0 1.2
1.0 1.1
1.1 1.2
```

it means that there is patch `1.0 -> 1.2` and it’s better to download 1 patch, than 2 patches `1.0 -> 1.1 -> 1.2`.

IF YOU WANT USE MERGED PATCHES LIKE ABOVE, REMEMBER TO ADD IT ON TOP OF FILE.

After configuration of *versions.txt* file is done, then copy patches from your game directory *Patchie/_output*.

You should have something like this:

```
1.0_1.1
1.0_1.2
1.1_1.2
versions.txt
```

Patches can be hosted on private computer or on web server (it can be without any interpreters like: PHP, CGI, Ruby).

Side note:

Remember that there is only one spacebar between the patch version numbers.

6. Changelog.

- ❖ 1.0.4
  + Added Launchie Prefab.
  + New patch creation algorithm fixing many issues and errors appearing with the old algorithm.
  + Url and Version are now public.
  + Launcher is built on layers allowing much easier adjustment to the project and future updates. (Separated Logic and GUI)
  + New Documentation.
  + Dropped Support for Unity 3.4+ (now minimum Unity version is 3.5.7)
  + Fixed BUG#0003 (Loading bar not showing download process after downloading the patch)

- ❖ 1.0.3
  + Visual refresh of default Lauchie skin.
  + Fixed BUG#0001 (Crafty crashing on Windows 8).
  + _output folder is now inside the Patchie folder cleaning the build order.

- ❖ 1.0.1
  + Added Support for Unity 3.4+.

7. Known issues.

- ~~BUG#0003~~

  ~~Loading bar not showing download process after downloading the patch.~~

- BUG#0002

  Rearranging scenes in Unity3d "Build settings" also change name of levels. It can cause Crafty to add them to patch even if nothing changed.

- ~~BUG#0001~~

  ~~Crafty crashes on Windows 8~~

8. Support and additional information.

Copy of this documentation can be found in the Documentation folder inside the Crafty plug-in folder.

- Asset Store.
- Official Penkura website.
- Crafty online documentation.
- Unity Forum.
- Contact Us.
- Youtube.