FOMOD Designer Documentation

Release 0.8.1.0

Daniel Nunes

Mar 06, 2017

Contents:

A visual editor to quickly create FOMOD installers for Nexus based mods.

If you're new around here, head to the *Getting Started* page!

CHAPTER 1

Main Features

- **No need to know XML** - Use a simple node tree and/or the node wizards to navigate, create and modify your installer without having to know a single bit of XML syntax;
- **No need to know FOMOD by heart** - The possible children nodes and all the options to customize each node are all laid out for you. The wizards include a small description to help you out at the top;
- **Includes all the features from FOMOD Validator** - Validate and check for possible errors when loading and saving;
- **Faster and simpler workflow** - With a very simple node tree to transverse instead of bulky documents you can save a lot of time and dedicate more to your content;
- **Don't get lost in your own installer** - With the ability to rename repeatable nodes (like the Pattern node) in-app with little to no interference in the final output you will no longer spend unnecessary time looking for the right tag to edit;
- **Don't want it? Hide it** - Spent lots of time in something you're not sure you want to keep? Hide it instead of deleting it, the node and its children are still fully editable but won't show up in the output, saving you the time of recreating everything later;
- **Preview your work** - Whether you want to preview how the xml will look like or which files a certain install step will install, it has you covered;
- **Default them!** - Let's face it, everyone uses Explicit Installation Steps even though the default value is Ascending. Worry no more, there is an option for that;
- **Forgot the flags?** - Auto-completers are included for flag labels and values, never mistype another flag;
- **Forget the path** - No more need to guess at paths and path separators, all path fields have a little button to the right that will open a dialog to point to the file directly;
- **Hotkeys everywhere** - Pretty much all actions have hotkeys associated with them, speed up that workflow;
- **Not a fan of hotkeys?** - Don't worry, we got your back, every action is stored in an easy to access menu, at the top or by right-clicking;
- **Customize** - Almost everything in this app is customizable, check the Settings menu!

Todo

- Finish the damn Wizards!
- Improve the documentation (I suck at this part)
- Start the Full Installer Preview

Getting Started

Welcome to the *FOMOD Designer* documentation! Let's get right to it.

Run the executable that comes with the package. If you need help with getting the correct package for you see the *Installation* page.

First, you'll see the **Intro** window. At the bottom of this window you'll see your most recently opened installers, in the future you can select one here to open it more quickly. Since you most likely have no recent installers, click the `New/Open` button. Here you'll choose the folder where the package you want to make an installer for is located.

Note: Now for an important distinction from other apps you may have used: the *FOMOD Designer* does not have separate **New** and **Open** buttons. Simply select the correct folder and it'll auto detect an existing installer. If you want to know about the behind the scenes for this, check the *F.A.Q.*.

The **Main** window should now appear. If you're a first-time user it should load the *Basic View* and you should head on to the *Basic Usage*. In case you're a returning user and/or you've enabled the *Advanced View* head to the *Advanced Usage*.

Attention: If you need help with a button or something else on the window, try hovering over it and checking the bottom left of the screen, in the status bar.

Installation

TL;DR: All you need to do is download the package, extract it somewhere and run the `FOMOD Designer` executable.

Pre-Built Executables

There are pre-built, ready-to-use executables always available for 64-bit Windows and often for 64-bit Linux as well.

It is recommended to use the latest stable version since it's less likely to have critical bugs. If you need to use a feature that hasn't made it to the stable builds, feel free to download the bleeding edge build.

If there are no builds for your system or you just love to have tons of work try building from source.

Building from Source

1. Download the repository from Github;

2. Unpack the archive into a folder;
3. Install Conda;
4. Open the command line/terminal in the folder from step 2;
5. Create the necessary environment within Conda:
   - Windows 64-bit:

```
conda create -y -n fomod-designer^
 -c https://conda.anaconda.org/mmcauliffe^
 pyqt5=5.5.1 python=3.5.1 lxml=3.5.0
```

   - Linux 64-bit:

```
conda create -y -n fomod-designer \
 -c https://conda.anaconda.org/mmcauliffe \
 pyqt5=5.5.1 python=3.5.1 lxml=3.5.0
```

   - For other platforms you'll have to figure out where the correct Conda packages are. As of now, you'll need these:

Package	Version
PyQt5	5.5.1
lxml	3.5.0

6. Activate the environment:
   - Windows:

```
activate fomod-designer
```

   - Other:

```
source activate fomod-designer
```

7. Install other dependencies:
   - Windows:

```
pip install pip -U
pip install setuptools -U --ignore-installed
pip install -r dev\reqs.txt
```

   - Other:

```
pip install pip -U
pip install setuptools -U --ignore-installed
pip install -r dev/reqs.txt
```

8. Build the app:

```
inv build
```

9. Done! The built package is in the `dist` folder within the folder in step 2.

Basic Usage

Todo

Describe basic usage - basic view and wizards.

For first-time users and those who don't really want to think too much about it. Follow each wizard's instructions in the app to fully build an installer. Remember than you can only save or open a new installer when on the very first page! If you're mid-way through your work but you want to save and leave, simply hit `Finish` until you reach that first page.

Basic View

Todo

Describe basic view here - pretty much just the initial page.

Wizards

This section contains the descriptions of all the wizards so if you have any doubts simply come check here! To search for a specific wizard use the search box on the left sidebar.

Todo

Finish wizards, not sure what to write here though.

Advanced Usage

For the advanced users and everyone who knows their way around a *FOMOD* installer. In this section you'll find descriptions of the tags and nodes themselves - what they are, how to use them and examples when needed.

There are no restrictions when using the *Advanced View*, we trust that you know what you're doing. This is recommended for people who already know how to create/modify XML installers and are interested in speeding up their work or for users who want more customization options than the *Basic View* offers.

Advanced View

The *Advanced View* can be divided in 4 parts: **Node Tree**, **Previews**, **Property Editor** and **Children Box**. All of these, with the exception of the **Previews**, can be moved around by the user.

The **Node Tree**, by default situated on the left, contains all the nodes in the installer's two trees: the *Info* and the *Config* tree. You can right-click the tree to see all the actions available - some of these, like *Delete*, are not available for the root nodes. You can also traverse the tree with the arrow keys and use the Enter key or left-click to select the node, this will update the **Property Editor** and the **Children Box** (and the **Previews** in case that is enabled).

The **Previews**, situated on the center, has two tabs: *GUI Preview* and *XML Preview*. The *GUI Preview* has a Mod Organizer-like interface that simulates the current *Install Step* - you can choose the options and the bottom half reflects the flags that would be set and/or the files that would be installed. The *XML Preview* has a preview of the XML code that that node and its children would output.

The **Property Editor**, by default situated on the top right, contains all the editable properties for the currently selected node. You can find more information for each node's properties in the *FOMOD Bible*.

The **Children Box**, by default situated on the bottom right, contains all the available children to add to the currently selected node. Click on a child button here to add the corresponding node.

Learn you a FOMOD For Great Good

This section contains the *FOMOD Bible* - a description of all the tags/nodes with examples. This is not meant to be read from top to bottom but rather as a dictionary or a glossary - search for the tag/node you need more info on with the search box on the left sidebar.

Tag vs Node

A **Tag** is any item within an xml document. Within the *FOMOD* schema (the document that defines the rules for installer documents) all the allowed tags for FOMOD are defined. A tag has the format `<tag/>` or `<tag>text goes here<tag/>` if it contains text.

Similarly, any item in the *FOMOD Designer*'s *Node Tree* is a **Node**. Every node has a direct correspondence to a xml tag. These two terms are use interchangeably in the *FOMOD Bible*.

Attribute vs Property

An **Attribute** is a way to customize a tag. These are also defined in the *FOMOD* schema and have the format: `<tag attribute="value"/>`.

A **Property** is the attribute equivalent for nodes. They can be edited via the *Property Editor*. In the *Bible* the properties are always followed by the corresponding attribute in square brackets (p.e. Name [name]).

> **Attention:** While a tag's text (p.e. `<tag>text goes here<tag/>`) is not an attribute, in a node it is bundled together with its properties for convenience. In order to distinguish text from other properties, it is marked with [...] as its attribute.
>
> So for a node that can have text its properties will have the line: Text [...]

Tag Order

Some tags are enforced a specific order by the *FOMOD* schema. When applicable, the possible/required children listed in each node are ordered.

This enforced order is reflected in the node tree. The user is able to modify the order of repeatable nodes through drag and drop.

FOMOD Bible

Please take note this isn't a fully comprehensive document (at least so far). If you want something more complete, feel free to look at the revised *FOMOD* schema.

Info

Tag fomod

Description The root node for the document containing all the information relative to the installer.

Children

Node	Repeatable
Name	**No**
Author	**No**
Description	**No**
ID	**No**
Categories Group	**No**
Version	**No**
Website	**No**

Properties *None*

Name

Tag Name

Description The node that holds the mod's name.

Children *None*

Properties

Property	Attribute	Description
Text	[...]	The name of the mod.

Author

Tag Author

Description The node that holds the mod's author(s).

Children *None*

Properties

Property	Attribute	Description
Text	[...]	The author(s) of the mod.

Description

Tag Description

Description The node that holds the mod's description.

Children *None*

Properties

Property	Attribute	Description
Text	[...]	The description of the mod.

ID

Tag Id

Description The node that holds the mod's ID. The ID is the last part of the nexus' link. Example:

Nexus mod link: http://www.nexusmods.com/skyrim/mods/548961 -> ID's text is 548961

Children *None*

Properties

Property	Attribute	Description
Text	[...]	The ID of the mod.

Categories Group

Tag Groups

Description This node's purpose is solely to group the categories this mod belongs to together.

Children

Node	Repeatable
Category	**Yes**

Properties *None*

Category

Tag element

Description The node that holds one of the mod's category.

Children *None*

Properties

Property	Attribute	Description
Text	[...]	A category this mod belongs to.

Version

Tag Version

Description The node that holds the mod's version.

Children *None*

Properties

Property	Attribute	Description
Text	[...]	This mod's version.

Website

Tag Website

Description The node that holds the mod's home website.

Children *None*

Properties

Property	Attribute	Description
Text	[...]	The mod's home website.

Config

Tag config

Description The main element containing the module configuration info.

Children

Node	Repeatable	Notes
Name	**No**	
Image	**No**	
Mod Dependencies	**No**	At least one of the following is required for the installer to have any effect: *Mod Dependencies*, *Installation Steps*, *Mod Requirements*, *Conditional Installation*
Installation Steps	**No**	At least one of the following is required for the installer to have any effect: *Mod Dependencies*, *Installation Steps*, *Mod Requirements*, *Conditional Installation*
Mod Requirements	**No**	At least one of the following is required for the installer to have any effect: *Mod Dependencies*, *Installation Steps*, *Mod Requirements*, *Conditional Installation*
Conditional Installation	**No**	At least one of the following is required for the installer to have any effect: *Mod Dependencies*, *Installation Steps*, *Mod Requirements*, *Conditional Installation*

Properties

Property	Attribute	Description
N/A	{http://www.w3.org/2001/XMLSchema-instance}noNamespaceSchemaLocation	This attribute contains the namespace for this file. This property is not editable. The value should always be: `"http://qconsulting.ca/fo3/ModConfig5.0.xsd"`

Name

Tag moduleName

Description The name of the module. Used to describe the display properties of the module title.

Children *None*

Properties

Property	Attribute	Description
Text	[...]	The name of the mod.
Position	position	The position of the mod's name in the header. Accepts the values: `"Left"`, `"Right"` or `"RightOfImage"`
Colour	colour	The colour of the mod's name in the header. Accepts RGB hex values.

Image

Tag moduleImage

Description The module logo/banner.

[Ignored in Mod Organizer]

Children *None*

Properties

Property	Attribute	Description
Path	path	The path to the image file.
Show Image	showImage	Whether the image is visible. Accepts `true` or `false`
Show Fade	showFade	Whether the image's opacity is fixed. Accepts `true` or `false`
Height	height	The maximum height of the image. Accepts any integer larger than `-1`

Mod Dependencies

Tag moduleDependencies

Description Items upon which the module depends. The installation process will only start after these conditions have been met.

While flag dependencies are allowed they should not be used since no flag will have been set at the time these conditions are checked.

Children

Node	Repeatable
File Dependency	**Yes**
Flag Dependency	**Yes**
Game Dependency	**No**
Dependencies	**Yes**

Properties

Property	Attribute	Description
Type	operator	The type of the dependency: `And` or `Or` If the type is `And`, all conditions under this node must be met. If the type is `Or`, only one condition must be met.

File Dependency

Tag fileDependency

Description Specifies that a mod must be in a specified state.

Children *None*

Properties

Property	Attribute	Description
File	file	The path to the file to be checked.
State	state	The supposed state of the file.

Flag Dependency

Tag flagDependency

Description Specifies that a condition flag must have a specific value.

Children *None*

Properties

Property	Attribute	Description
Flag	flag	The flag where this condition falls upon.
Value	value	The value of the flag to be checked.

Game Dependency

Tag gameDependency

Description Specifies a minimum required version of the installed game.

[Ignored in Mod Organizer]

Children *None*

Properties

Property	Attribute	Description
Version	version	The minimum version of the game.

Installation Steps

Tag installSteps

Description The list of install steps that determine which files (or plugins) that may optionally be installed for this module.

Children

Node	Repeatable	Notes
Install Step	**Yes**	At least one of *Install Step* is required.

Properties

Property	Attribute	Description
Order	order	The order of the install steps beneath this node. `"Explicit"` follows document order while the others order alphabetically. Accepts `"Ascending"`, `"Descending"` or `"Explicit"`

Install Step

Tag installStep

Description A step in the install process containing groups of optional plugins.

Children

Node	Repeatable	Notes
Visibility	**No**	
Option Group	**No**	At least one of *Option Group* is required.

Properties

Property	Attribute	Description
Name	name	The name of this install step.

Visibility

Tag visible

Description The pattern against which to match the conditional flags and installed files. If the pattern is matched, then the install step will be visible.

Children

Node	Repeatable
File Dependency	**Yes**
Flag Dependency	**Yes**
Game Dependency	**No**
Dependencies	**Yes**

Properties

Property	Attribute	Description
Type	operator	The type of the dependency: `And` or `Or` If the type is `And`, all conditions under this node must be met. If the type is `Or`, only one condition must be met.

Dependencies

Tag dependencies

Description A dependency that is made up of one or more dependencies.

Children

Node	Repeatable
File Dependency	**Yes**
Flag Dependency	**Yes**
Game Dependency	**No**
Dependencies	**Yes**

Properties

Property	Attribute	Description
Type	operator	The type of the dependency: `And` or `Or` If the type is `And`, all conditions under this node must be met. If the type is `Or`, only one condition must be met.

Option Group

Tag optionalFileGroups

Description The list of optional files (or plugins) that may optionally be installed for this module.

Children

Node	Repeatable	Notes
Group	**Yes**	At least one of *Group* is required.

Properties

Property	Attribute	Description
Order	order	The order of the install steps beneath this node. `"Explicit"` follows document order while the others order alphabetically. Accepts `"Ascending"`, `"Descending"` or `"Explicit"`

Group

Tag group

Description A group of plugins for the mod.

Children

Node	Repeatable	Notes
Plugins	**No**	At least one of *Plugins* is required.

Properties

Property	Attribute	Description
Name	name	The name of this group.
Type	type	The selection type for this group. Accepts `"SelectAny"`, `"SelectAtMostOne"`, `"SelectExactlyOne"`, `"SelectAll"` or `"SelectAtLeastOne"`

Plugins

Tag plugins

Description The list of plugins in the group.

Children

Node	Repeatable	Notes
Plugin	**Yes**	At least one of *Plugin* is required.

Properties

Property	Attribute	Description
Order	order	The order of the plugins beneath this node. `"Explicit"` follows document order while the others order alphabetically. Accepts `"Ascending"`, `"Descending"` or `"Explicit"`

Plugin

Tag plugin

Description A mod plugin belonging to a group.

Children

Node	Repeatable	Notes
Description	**No**	At least one of *Description* is required.
Image	**No**	
Files	**No**	
Flags	**No**	
Type Descriptor	**No**	At least one of *Type Descriptor* is required.

Properties

Property	Attribute	Description
Name	name	The name of this plugin.

Description

Tag description

Description A description of the plugin.

Children *None*

Properties

Property	Attribute	Description
Description	[...]	The plugin's description.

Image

Tag image

Description The optional image associated with a plugin.

Children *None*

Properties

Property	Attribute	Description
Path	path	The path to the image.

Files

Tag files

Description A list of files and folders to be installed.

Children

Node	Repeatable
File	**Yes**
Folder	**Yes**

Properties *None*

File

Tag file

Description A file belonging to the plugin or module.

Children *None*

Properties

Property	Attribute	Description
Source	source	The path to the file.
Destination	destination	The path from the game's mod folder to the destination of this file.
Priority	priority	The priority of the file. Higher priority means the file will overwrite other files with lower priority.
Always Install	alwaysInstall	If `true`, this file will be always installed, regardless of the user's choice. Accepts `true` or `false`
Install If Usable	installIfUsable	If `true`, this file will be installed unless the plugin's type is `NotUsable`, regardless of the user's choice. Accepts `true` or `false`

Folder

Tag folder

Description A folder belonging to the plugin or module.

Children *None*

Properties

Property	Attribute	Description
Source	source	The path to the folder.
Destination	destination	The path from the game's mod folder to the destination of this folder.
Priority	priority	The priority of the folder. Higher priority means the folder will overwrite other files with lower priority.
Always Install	alwaysInstall	If `true`, this folder will be always installed, regardless of the user's choice. Accepts `true` or `false`
Install If Usable	installIfUsable	If `true`, this folder will be installed unless the plugin's type is `NotUsable`, regardless of the user's choice. Accepts `true` or `false`

Flags

Tag conditionFlags

Description The list of condition flags to set if the plugin is in the appropriate state.

Children

Node	Repeatable	Notes
Flag	**Yes**	At least one of *Flag* is required.

Properties *None*

Flag

Tag flag

Description A condition flag to set if the plugin is selected.

Children *None*

Properties

Property	Attribute	Description
Label	name	The flag's identifying label.
Value	[...]	The flag's new value.

Type Descriptor

Tag typeDescriptor

Description Describes the type of a plugin.

Children

Node	Repeatable	Notes
Dependency Type	**No**	Either *Dependency Type* or *Type* must be used.
Type	**No**	Either *Dependency Type* or *Type* must be used.

Properties *None*

Dependency Type

Tag dependencyType

Description Used when the plugin type is dependent upon the state of other mods.

Children

Node	Repeatable	Notes
Patterns	**No**	At least one of *Patterns* is required.
Default Type	**No**	At least one of *Default Type* is required.

Properties *None*

Patterns

Tag patterns

Description The list of dependency patterns against which to match the user's installation. The first pattern that matches the user's installation determines the type of the plugin.

Children

Node	Repeatable	Notes
Pattern	**Yes**	At least one of *Pattern* is required.

Properties *None*

Pattern

Tag pattern

Description A specific pattern of mod files and condition flags against which to match the user's installation.

Children

Node	Repeatable	Notes
Dependencies	**No**	At least one of *Dependencies* is required.
Type	**No**	At least one of *Type* is required.

Properties *None*

Type

Tag type

Description The type of the plugin.

Children *None*

Properties

Property	Attribute	Description
Type	name	Describes the plugin's type. Accepts `Required`, `Recommended`, `Optional`, `CouldBeUsable` or `NotUsable`

Default Type

Tag defaultType

Description The default type of the plugin used if none of the specified dependency states are satisfied.

Children *None*

Properties

Property	Attribute	Description
Type	name	Describes the plugin's type. Accepts `Required`, `Recommended`, `Optional`, `CouldBeUsable` or `NotUsable`

Mod Requirements

Tag requiredInstallFiles

Description The list of files and folders that must be installed for this module.

Children

Node	Repeatable
File	**Yes**
Folder	**Yes**

Properties *None*

Conditional Installation

Tag conditionalFileInstalls

Description The list of optional files that may optionally be installed for this module, based on condition flags.

Children

Node	Repeatable	Notes
Patterns	**No**	At least one of *Patterns* is required.

Properties *None*

Patterns

Tag patterns

Description The list of patterns against which to match the conditional flags and installed files. All matching patterns will have their files installed.

Children

Node	Repeatable	Notes
Pattern	**Yes**	At least one of *Pattern* is required.

Properties *None*

Pattern

Tag pattern

Description A specific pattern of mod files and condition flags against which to match the user's installation.

Children

Node	Repeatable	Notes
Files	**No**	At least one of *Files* is required.
Dependencies	**No**	At least one of *Dependencies* is required.

Properties *None*

Contributing

We love contributions from everyone. By participating in this project, you agree to abide by the thoughtbot code of conduct.

Issues

Before submitting your issue, please make sure that you've provided all the info required in the issue template.

Pull Requests

Before submitting your pull request, please make sure that you've provided all the info required in the pull request template.

Contributing Code

General Guidelines:

- This repo uses the gitflow branching model. Don't commit directly to the `master` or `develop` branches.
- Make sure the tests pass on the CI server. Local tests are not available at the moment.
- Follow the style guide.
- Write decent commit messages.
- Run **`inv docs`** to generate documentation locally, **`inv build`** to build the executable and **`inv preview`** to preview the app without building it.

Setup the work environment:

1. Fork the repo.
2. Setup your fork locally.
3. This repo uses a `.settings` file to define all the necessary settings. This file follows this syntax:

   ```
   [git]
   user=git_username
   email=git_email
   ```

   Create and add this file to your clone's root.
4. Install Vagrant.
5. Run this in the clone's root:
   - If you have Python available:

     ```
     pip install invoke
     inv create enter
     ```

   - If not:

     ```
     vagrant up
     vagrant ssh -- -Yt 'cd /vagrant/; /bin/bash'
     ```

   It will take a while.

6. You should now be inside an Ubuntu Trusty virtual machine, this is where you'll work. Make, commit and push your changes.
7. Create a pull request.

Thank you, contributors!

Changelog

0.8.1 (2016-08-11)

- Fixed remote CI whitelisted builds.

0.8.0 (2016-08-11)

- Documentation is now available.
- Users are now able to manipulate and add comments.
- Users are now able to hide non-comment nodes.
- 32 bit builds are now available.

0.7.2 (2016-07-13)

- Plugin node should now have the correct required child nodes.
- Fixed validation and warning dialogs and ignore process.
- Fixed files processing in preview.

0.7.1 (2016-07-12)

- Fixed preview issue with non-existent nodes under info root.
- Updated validation and children groups.

0.7.0 (2016-07-10)

- Fixed rare bug with the validator.
- Added Dependencies Wizard.
- Children box now lists invalid child nodes greyed out instead of deleting them.
- Fixed issues with saving.
- Nodes should now be properly sorted.
- Added specialized child nodes with colours.
- Added auto-completion for flag labels and their values.
- Fixed recent files issues.
- Properties are now ordered.
- Added node-specific metadata.

- Pattern node's names are now editable.
- Improved Setting's dialog.
- Added Defaults section to settings.
- Added Appearance tab to settings dialog.
- Fixed rare bug with xml preview.
- Disabled Wizards.
- Added user-defined noe sorting with drag and drop.
- Improved logos.
- Added copy and paste functionality.
- Added undo and redo functionality.
- Loading ui should now be slightly faster - ui is now pre-compiled.
- Full keyboard navigation is now supported on the node tree.
- Added context menu to the node tree.
- Actions should now be properly disabled/enabled when appropriate.
- All nodes should now be correct on their allowed number.
- Added plain text editor to most simple text properties and html editor to plugin's description text.
- Added install step ui preview.
- Added tutorial at startup. Added setting to re-enable the tutorial.

0.6.0 (2016-06-13)

- Added check for updates at startup.
- Added line numbers to code preview.
- Moved previews to separate threads, loading each node should now be much faster.
- Improved button look on object box.

0.5.1 (2016-06-12)

- Fixed versioning issues.

0.5.0 (2016-06-12)

- Added intro window.
- Added Files wizard.
- Added wizard environment setup.
- Updated app and file icons.
- The object box now consists of independent buttons for each child instead of a list.
- A message box asking for confirmation should now appear when trying to open a new installer while there unsaved changes.

- Property editor should now be properly cleared when opening a new installer.
- A message box asking for action should now appear when using the recent files menu and the path no longer exists.
- Fixed relation between view menu and docked widget states.
- Dialog windows should now properly be placed on top of other windows.
- Improved some nodes' names.

0.4.1 (2016-05-16)

- Fixed wrong default attributes in file and folder tags.
- Added wizard framework.

0.4.0 (2016-05-14)

- Added file and window icons.
- Fixed combo boxes not being set at start.
- Added recent files menu.
- Added about dialog.
- Added view menu.
- Closing the main window with unsaved changes should now display a warning.
- Not identified tags should be properly handled now.
- Syntax errors in the xml should be properly handled now.
- File, folder and colour properties now have a proper specific widget.
- Added sorting to xml elements when saving.
- Added xml code preview.
- Added settings window.
- Attribute parsing should now properly ignore the ones that are unknown.
- Validation and warning checks added.
- Multiple bugfixes.

0.3.1 (2016-04-17)

- Tags/item with name/source property now have that as the title instead of the tag's name.
- Fixed all keyboard shortcuts.
- Everything is now included within a single executable.
- Added full linux support.
- Included build number in version.
- Fixed no error raised when no required child exist.
- Window title now includes an asterisk when any content has been modified.

- Missing files in fomod folder are now properly checked.
- Fixed spinbox property.

0.3.0 (2016-04-07)

- All basic functionality is now done.
- Tag properties are now properly displayed and editable.
- XML comments are now ignored by the parser.
- Child objects are now auto-selected when created.
- Fixed error when opening an installer over an already opened one.
- Fixed dependencies tag not being able to be self nested.
- Fixed deployed archive structure.

0.2.1 (2016-04-05)

- In-tag text is now properly parsed and saved along with everything else.

0.2.0 (2016-04-05)

- Users can now modify the installer's objects.

0.1.0 (2016-04-03)

- Users can now open and save FOMOD installers.
- Main windows title now shows which package you are currently working on.

0.0.1 (2016-03-15)

- GUI draft completed.

Frequently Asked Questions

Why are the *New* and *Open* buttons merged?

Ok, let's run through what would happen in the code for the **New** button:

1. Get the package folder from the user;
2. Check if an installer exists in that folder;
3. If it doesn't exist, create a new one;
4. It it does exist, complain to the user.

Now for the **Open** button:

1. Get the package folder from the user;

2. Check if an installer exists in that folder;
3. If it doesn't exist, complain to the user;
4. It it does exist, open it up.

Do you see how similar these two are? It wouldn't really make sense to have two completely separate actions that do pretty much the same thing. This way everything is much simpler on our side and we never have to complain to you!