



MARI
SOFTWARE API OVERVIEW
VERSION 2.0v1

Software API Overview™. Copyright © 2013 The Foundry Visionmongers Ltd. All Rights Reserved. Use of this user guide and the Mari software is subject to an End User License Agreement (the "EULA"), the terms of which are incorporated herein by reference. This user guide and the Mari software may be used or copied only in accordance with the terms of the EULA. This user guide, the Mari software and all intellectual property rights relating thereto are and shall remain the sole property of The Foundry Visionmongers Ltd. ("The Foundry") and/or The Foundry's licensors.

The EULA can be read in the Mari User Guide Appendices.

The Foundry assumes no responsibility or liability for any errors or inaccuracies that may appear in this user guide and this user guide is subject to change without notice. The content of this user guide is furnished for informational use only.

Except as permitted by the EULA, no part of this user guide may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, recording or otherwise, without the prior written permission of The Foundry. To the extent that the EULA authorizes the making of copies of this user guide, such copies shall be reproduced with all copyright, trademark and other proprietary rights notices included herein. The EULA expressly prohibits any action that could adversely affect the property rights of The Foundry and/or The Foundry's licensors, including, but not limited to, the removal of the following (or any other copyright, trademark or other proprietary rights notice included herein):

Mari™ software © 2013 The Foundry Visionmongers Ltd. All Rights Reserved.

Mari™ is a trademark of The Foundry Visionmongers Ltd.

In addition to those names set forth on this page, the names of other actual companies and products mentioned in this Getting Started Guide (including, but not the to, those set forth below) may be the trademarks or service marks, or registered trademarks or service marks, of their respective owners in the United States and/or other countries. No association with any company or product is intended or inferred by the mention of its name in this Software API Overview.

Linux ® is a registered trademark of Linus Torvalds.

Mari software engineering: Jack Greasley, Kiyoyuki Nakagaki, Marcus Shoo, Kevin Atkinson, Tim Ebling, Jed Soane, Daniel Lond, Robert Fanner, Duncan Hopkins, Mark Final, Chris Bevan, Carl Rand, Phil Hunter and Tim Smith
Product testing: Michael Zannetou, Mark Titchener, Robert Elphick, Antoni Kujawa and Chris Hiess
Writing and layout design: Jack Elder, Jon Hertzog, Eija Närvänen, Charles Quinn and Erica Cargle
Proof reading: Jack Elder and Eija Närvänen

Mari includes Disney technology licensed from Walt Disney Animation Studios.

The Foundry
6th Floor, The Communications Building,
48 Leicester Square,
London
WC2H 7LT

Rev: March 8, 2013

Contents

PREFACE	About this Guide	4
	Contact Customer Support	4
USING PYTHON IN MARI	Introduction	5
	Steps for Using Python in Mari	5
	Review Mari’s Python Documentation	5
	Enter Python Statements in the Python Console Palette	6
	Save or Import a Script.	8
	Load a Script.	8
	Terminal Mode	9
	On Linux	9
	On Windows	9
USING MARI’S C API	Introduction	11
	Review Mari’s C API Documentation.	11
CUSTOM SHADERS	Modular Shaders.	12

1 PREFACE

Mari is a creative texture-painting tool that can handle extremely complex or texture heavy projects. It was developed at Weta Digital and has been used on films such as *The Adventures of Tintin: The Secret of the Unicorn*, *District 9*, *The Day the Earth Stood Still*, *The Lovely Bones*, and *Avatar*.

The name Mari comes from the Swahili 'Maridadi', meaning 'beautiful' and carrying connotations of 'usefulness'.

About this Guide

This guide is aimed at developers and provides you with the basic information you need to get started using the Python, Shader and C APIs in Mari.

For further information on Mari and its functions, see the accompanying *Mari User Guide* and *Mari Reference Guide*.

Contact Customer Support

Should questions arise that this guide fails to address, you can contact Customer Support directly via e-mail at support@thefoundry.co.uk or via telephone to our London office on +44 (0)20 7968 6828 or to our Los Angeles office on (310) 399 4555 during office hours.

2 USING PYTHON IN MARI

Introduction

Mari uses Python 2.6.5 and Unicode Character Set (UCS) 4.

PythonQt provides access to almost all of the Qt libraries (Qt is a C++ GUI library developed by Qt Development Frameworks). If you've used PyQt in the past, PythonQt looks very similar:

```
# PyQt version
from PyQt4 import QtGui
# PythonQt version
from PythonQt import QtGui
# Everything else is the same in this example
w = QtGui.QLabel("hello")
w.show()
```

For more information on Qt and PythonQt, visit <http://qt.nokia.com/> and <http://pythonqt.sourceforge.net/>.

Most of Mari's functions are implemented through the use of manager objects, such as `mari.projects`, `mari.menus`, and so on.

Steps for Using Python in Mari

To use Python in Mari, follow these steps:

1. [Review Mari's Python Documentation](#)
2. [Enter Python Statements in the Python Console Palette](#)
3. If you want your script to automatically run on start-up, [Save or Import a Script](#) to:
 - `~/Mari/Scripts`, or
 - your own directory by setting the environment variable `MARI_SCRIPT_PATH` to a custom folder.
4. [Load a Script](#) via the **Script Path** entry box.

Review Mari's Python Documentation

1. To view HTML documentation on Mari's Python API, select **Python > API** in Mari.
2. To view example Python scripts, go to the **Media/Scripts/examples** subdirectory of the Mari application directory and open any of the `.py`

files there in a text editor. You can also see the results of these example scripts in the **Python** menu in Mari.



NOTE: Executing the Python "help" command in the **Python Console** also launches the HTML documentation in a web browser.



TIP: To read more about Python, review its documentation, or interact with other Python users, you can also visit the Python programming language official website at <http://www.python.org/>.

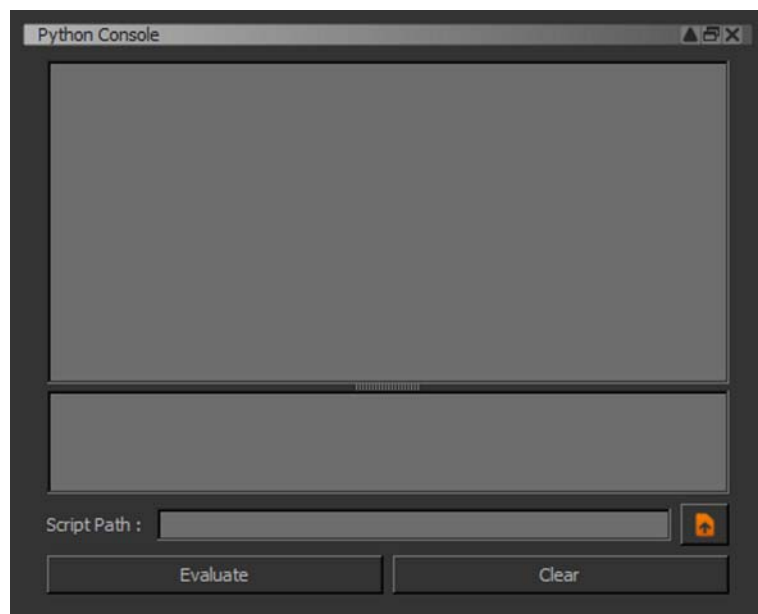
Enter Python Statements in the Python Console Palette

Mari's **Python Console** palette behaves in a very similar way to a standard Python console. Here's how you use it:

1. If the **Python Console** is already open, click the **Console** tab to give it focus; or if it's closed, right-click in the toolbar area on top of the Mari workspace and select **Python Console** to open it. You can also find it in the menu under **View > Palettes**.

*The **Python Console** is divided into two parts. You use the lower part (input pane) to type in and execute your Python statements, and when you have done so, statements and their outputs appear in the upper part of the console (output pane).*

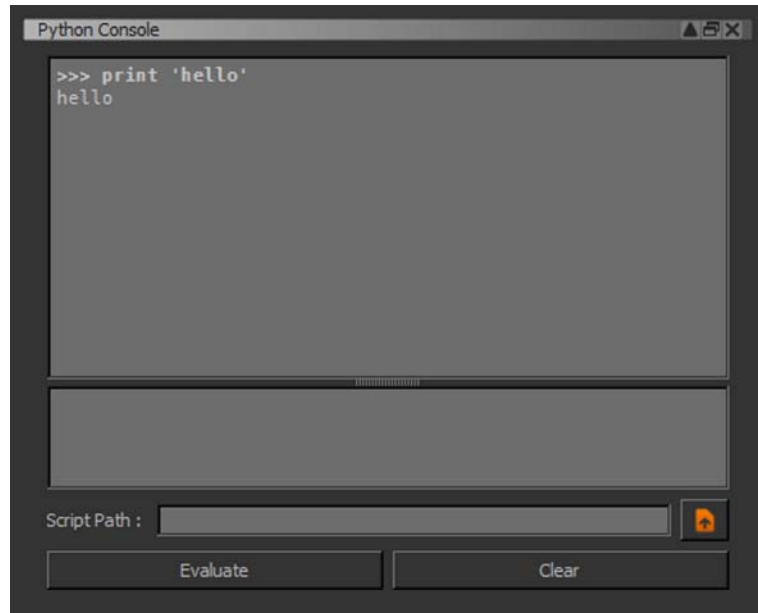
2. To enter a statement, type it into the input pane.



To use the usual editing functions, such as copy and paste, right-click on the input pane and select the desired function.

3. To execute the statement, click the **Evaluate** button or press **Ctrl+Return**.

Your statement disappears from the input pane, and appears in the output pane, preceded by >>> (or ... if your statement spans multiple lines).



If you enter an invalid statement, Mari produces an error in the output pane.

4. If you want to repeat a statement, you can step backwards or forwards through the history of your script by pressing **Ctrl+Up** or **Ctrl+Down**.
5. If you want to clear the input and output panes, click the **Clear** button.



NOTE: Sometimes you may get an error if you copy and paste statements into the **Python Console** from another source, like an e-mail. This may be caused by the mark-up or encoding of the source you copied the statement from. To fix the problem, re-enter the statement manually.



TIP: You can increase or decrease the font size in the **Python Console** by holding down **Ctrl** and pressing **+** or **-**.

Save or Import a Script

If you would like your script to be automatically executed at start-up, do the following:

1. Enter your script in any text editor.
2. Save the text file with the extension `.py` (for example, `my_module.py`) to:
 - the `~/Mari/Scripts` subdirectory of the Mari application directory, or
 - your own directory by setting the environment variable `MARI_SCRIPT_PATH` to a custom folder.

If a script called `init.py` exists, it is run first. If you need scripts to run in a specified order, you should put them in a separate folder and use a Python `import` statement in `init.py`.

You can also specify multiple directories to run scripts on start-up, using a path list separated by `:` or `;` as per the standard for your operating system.

On Linux: `/home/username/dir1:/home/username/dir2`

On Windows: `C:\Users\username\dir1;C:\Users\username\dir2`

3. The `import` directory, contained in the following folders of your custom scripts path:

- **On Linux:** `/Mari/Scripts`
- **On Windows:** `Documents\Mari\Scripts`.


Scripts saved to this directory are imported rather than run, which prevents unnecessary pollution of the global namespace. If you save your custom script to the `import` directory, it is automatically imported before the scripts in the main folder are run.



TIP: Python files and subfolders that you wish to import without running in the `import` directory must be `_init_.py` files.

Load a Script

To load a script via the **Script Path** entry box:

1. Manually enter the script's location, or
2. Click on  to launch the **Python Script Path** dialog box and browse to the location of the script.
3. To execute the statement, click the **Evaluate** button.

Custom scripts load from the `import` directory prior to the scripts being run from the main folder. Mari uses particular order to load modules to ensure the proper, full functionality of the program when using Python scripts.

If existing start-up scripts do not work correctly due to the load module

order, you can revert back to the old behavior by setting the environment variable **MARI_OLD_PYTHON_INIT** to any non-empty string other than 0.



NOTE: Mari looks for metadata that is over 250MB in size and discards anything over this size. This check is performed on project load, and is intended to strip corrupt and problematic data. This affects metadata added via the Python API.

Terminal Mode

Mari provides a terminal mode, which allows users to access features through a Python shell on the command line. This is similar to the Python console palette in the Mari GUI, or to a standard Python shell. Terminal mode can be used to perform administrative actions such as archiving projects, exporting textures, and other tasks that do not require graphical interaction from the user.



NOTE: Terminal mode is not a true "headless mode" because it still requires graphics functionality to run. Mari still initializes its GUI when starting up terminal mode - it just won't display it.

You can supply the file name of a Python script to run on the command line, if desired. If supplied, Mari runs the specified script before providing the Python shell input prompt. It is also possible to start in "execute mode", which is the same as terminal mode but exists after running the provided script.

To start the terminal mode from a shell, use the following commands:

On Linux

```
cd /path/to/mari
# Normal terminal mode
./mari -t
# Terminal mode - run the script, then show a Python input prompt
./mari -t /path/to/script_name.py
# Execute mode - run the script, then exit
./mari -x /path/to/script_name.py
```

On Windows

```
cd \path\to\mari
# Normal terminal mode
Mari2.0v1.exe -t
# Terminal mode - run the script, then show a Python input prompt
```

```
Mari2.0v1.exe -t \path\to\script_name.py  
# Execute mode - run the script, then exit  
Mari2.0v1.exe -x \path\to\script_name.py
```

3 USING MARI'S C API

Introduction

Mari provides an SDK for developing binary plug-ins for functionality where Python scripting is not the best solution.

The C API allows users to develop plug-ins to read and write custom image formats, or to read custom geometry formats. In these cases, passing data between the application and plug-ins can be done more efficiently by passing buffers of binary data rather than using intermediate Python types.

The choice to use C instead of C++ for the API provides better compatibility and simpler compilation for users, and as always, can still be used in plug-ins written in C++.

Review Mari's C API Documentation

For full details of the C API, including example plug-ins, please see the C API documentation under **Help > SDK > C API > Documentation**, or via the installation directory in the SDK folder.

4 CUSTOM SHADERS

Modular Shaders

Versions of Mari prior to the 2.0v1 release provided Python API hooks to register custom "modular shaders." However, the modular shader system has since been deprecated in favor of a more flexible graph-based system.

Currently, the Mari team is working towards providing APIs for writing and registering custom shaders in Mari's graph-based shader system. Though there aren't API hooks for the shader system yet, if your pipeline requires custom modifications to the Mari shading system, please contact support@thefoundry.co.uk for assistance in writing your own shaders.