Jamoma

Style Sheet and User Interface Guideline
Version 0.1.1

Jamoma Design Group

Copyright Notices

Credits & Acknowledgments

Documentation: Timothy Place.

Software components: Timothy Place and Trond Lossius

Consulting, feedback, testing, and assistance: Jeremy Bernstein, Oliver Delano, Sebastian Lexer.

Table of Contents

Introduction

What is Jamoma?

Structured Patches.

Who is Created Jamoma?

Well... Hopefully, a growing number of us...

What's Included

The Jamoma Style Sheet

A brief description of the function of this document

Jamoma Module Components

A summary of this

Example Modules

List them

Module Building Wizards

Briefly describe

Getting Started

This will be a short tutorial on using Jamoma modules in your Max patch.

Jamoma Module Specification

Type an overview of the specification here.

Discuss the difference between a message and a parameter.

Discuss that all modules should return their parameter values as they are updated.

File Format

All module filenames should end with the ".mod" extension, and should be saved in Max's text format. Similarly, all algorithms should end with ".alg" extension and be saved in Max's text format. Other underlying patches or abstractions should also be saved in text format. The preferred extensions for text format files (with the exception of .mod or .alg files) is .txt or .mxt.

If there is a situation where a binary format patcher must be used. the preferred extension for binary Max patches is .mxb.

Standardized Messages and Parameters

Some intro.

All modules should, where feasible, implement the following messages:

- restore_defaults: loads a default XML preset file and applies it
- load_settings <optional-argument>: loads a preset file (pattr-XML format) and applies the settings stored therein. The optional argument specifies the path of a file, otherwise a dialog is presented.
- save_settings <optional-argument>: saves the current settings in a preset (XML) file. The optional argument specifies the path of a file, otherwise a dialog is presented.
- view_internals: gives access, where possible, to the internal Max patch (the algorithm) so that it may be viewed or evaluated by the user.

All audio modules should implement these messages, in addition to those above:

- mute <toggle>:
- bypass <toggle>:
- gain <float>:
- midigain <int>:
- sr <int>:
- defeat_meters <toggle>:
-

All jitter-based video modules should implement these messages, in addition to those above:

- mute <toggle>:
- bypass <toggle>:
- freeze <toggle>:
- preview <toggle>:
- genframe: generates a frame, essentially sending a bang to a Jitter matrix operator
-

Reserved Syntax and Messages

No module may accept or use the following:

- Any message beginning with an undescore. These are reserved for internal use by Jamoma module components.
- Any of the following messages. These messages are integral part of standard Max objects, or may in some other way adversely effect the functioning of the internal components of a Jamoma module:
    - bang
    - mode
    - open
    - close
    -
- Any standard message, as previously defined, in a way that differs from the way defined in this style sheet.

User Interface Dimensions

User interface dimensions are based on the paradigm of conventional rack mount hardware. To demonstrate this, the width a “1U” module is 510 pixels. Thus the width of a half rack unit module is 255 pixels. The width of 510 pixels was chosen to maximize the number of modules that can fit on a monitor with a resolution of 1024 pixels.

The height of a 1U module is 60 pixels. Heights then, must be a multiple of 60 pixels high. The size of a module’s interface must follow these guidelines. Furthermore, the size of the module should be accessible to the host patch so that it may query the module about this.

The chart below shows the dimensions of the most common module sizes.

Module Size	Width	Height
1U-half	255	60
1U	510	60
2U-half	255	120
2U	510	120
3U-half	255	180
3U	510	180

Modules may be made which are taller that those in this table, provided they follow the same convention.

User Interface Features

The user interface may designed according to the desires and whims of the developer. To maximize the experience of a user, however, it is ideal to remain somewhat consistent with the conventions established by the example modules that are included with the standard Jamoma download.

Building a New Module Using the Module Wizards

Fun stuff!

Building a New Module by Hand

This is the most flexible way.

Contact and Support

Because Jamoma is open source and freely available, there is no explicit support mechanism for the software. You should start by checking to see what resources are available on the Jamoma project page or by searching in Google. In any correspondance, it will likely be helpful to provide at least the following information:

- Type of computer you are using
- Version of your Operating System
- Type of audio interface or sound card
- Type of cable you are using to connect the Teabox to your sound card
- Version (and name) of the software you are using with the Teabox
- A concrete list of steps to reproduce your problem, from scratch
- What sensors you are using with the system
- Confirm that you've looked through our Troubleshooting Guide in this Document.