Embroidermodder v2.0-alpha Manual

Abstract

Table of Contents

Features

Embroidermodder 2 has many advanced features that enable you to create awesome designs quicker, tweak existing designs to perfection, and can be fully customized to fit your workflow.

A summary of these features:

- Cross Platform
- Realistic rendering
- Various grid types and auto-adjusting rulers
- Many measurement tools
- Add text to any design
- Supports many formats
- Batch Conversion
- Scripting API

Cross Platform

If you use multiple operating systems, it's important to choose software that works on all of them.

Embroidermodder 2 runs on Windows, Linux and Mac OS X. Let's not forget the Raspberry Pi.

Realistic Rendering

It is important to be able to visualize what a design will look like when stitched and our pseudo "3D" realistic rendering helps achieve this.

Realistic rendering sample #1:

Realistic rendering sample #2:

Realistic rendering sample #3:

Various grid types and auto-adjusting rulers

Making use of the automatically adjusting ruler in conjunction with the grid will ensure your design is properly sized and fits within your embroidery hoop area.

Use rectangular, circular or isometric grids to construct your masterpiece!

Multiple grids and rulers in action:

Many measurement tools

Taking measurements is a critical part of creating great designs. Whether you are designing mission critical embroidered space suits for NASA or some other far out design for your next meet-up, you will have precise measurement tools at your command to make it happen. You can locate individual points or find distances between any 2 points anywhere in the design!

Take quick and accurate measurements:

Add text to any design

Need to make company apparel for all of your employees with individual names on them? No sweat. Just simply add text to your existing design or create one from scratch, quickly and easily. Didn't get it the right size or made a typo? No problem. Just select the text and update it with the property editor.

Add text and adjust its properties quickly:

Supports many formats

Embroidery machines all accept different formats. There are so many formats available that it can sometimes be confusing whether a design will work with your machine.

Embroidermodder 2 supports a wide variety of embroidery formats as well as several vector formats, such as SVG and DXF. This allows you to worry less about which designs you can use.

Batch Conversion

Need to send a client several different formats? Just use libembroidery-convert, our command line utility which supports batch file conversion.

There are a multitude of formats to choose from:

Scripting API

If you've got programming skills and there is a feature that isn't currently available that you absolutely cannot live without, you have the capability to create your own custom commands for Embroidermodder 2. We provide an QtScript API which exposes various application functionality so that it is possible to extend the application without requiring a new release. If you have created a command that you think is worth including in the next release, just contact us and we will review it for functionality, bugs, and finally inclusion.

An Embroidermodder 2 command excerpt:

Contributing

Version Control

Being an open source project, developers can grab the latest code at any time and attempt to build it themselves. We try our best to ensure that it will build smoothly at any time, although occasionally we do break the build. In these instances, please provide a patch, pull request which fixes the issue or open an issue and notify us of the problem, as we may not be aware of it and we can build fine.

Try to group commits based on what they are related to: features/bugs/comments/graphics/commands/etc...

See the coding style here # Embroidermodder 1.90.0 Manual

Table of Contents

Introduction

Basic Features

Move a single stitch in an existing pattern

1. In the `File` menu, click `Open...`. When the open dialog appears find and select your file by double clicking the name of the file. Alternatively, left click the file once then click the `Open` button.
2.
3. In the `File` menu

TIP: For users who prefer

Convert one pattern to another format

1. In the `File` menu, click `Open...`.
2. The
3. In the dropdown menu within the save dialog select the

Advanced Features

Other Projects

References

Planning

To see what's planned open the Projects tab which sorts all of the GitHub Issues into columns.

Format Support

FORMAT	READ	WRITE	NOTES
10o	YES		read (need to fix external color loading) (maybe find out what ctrl
100			none (4 byte codes) 61 00 10 09 (type, type2, x, y ?) x & y (signed char)
art			none
bro	YES		read (complete)(maybe figure out detail of header)
cnd			none
col			(color file no design) read(final) write(final)
csd	YES		read (complete)
dat			read ()
dem			none (looks like just encrypted cnd)
dsb	YES		read (unknown how well) (stitch data looks same as 10o)
dst	YES		read (complete) / write(unknown)
dsz	YES		read (unknown)

FORMAT	READ	WRITE	NOTES
dxf			read (Port to C. needs refactored)
edr			read (C version is broken) / write (complete)
emd			read (unknown)
exp	YES		read (unknown) / write(unknown)
exy	YES		read (need to fix external color loading)
fxy	YES		read (need to fix external color loading)
gnc			none
gt			read (need to fix external color loading)
hus	YES		read (unknown) / write (C version is broken)
inb	YES		read (buggy?)
jef	YES		write (need to fix the offsets when it is moving to another spot)
ksm	YES		read (unknown) / write (unknown)
pcd			
pcm			
pcq			read (Port to C)
pcs	BUGGY		read (buggy / colors are not correct / after reading, writing any other format is messed up)

FORMAT	READ	WRITE	NOTES
pec			read / write (without embedded images, sometimes overlooks some stitches leaving a gap)
pel			none
pem			none
pes	YES		
phb			
phc			
rgb			
sew	YES		
shv			read (C version is broken)
sst			none
svg		YES	
tap	YES		read (unknown)
u01			
vip	YES		
vp3	YES		
xxx	YES		
zsk			read (complete)

Support for Singer FHE, CHE (Compucon) formats?

Embroidermodder Project Coding Standards

A basic set of guidelines to use when submitting code.

Naming Conventions

Name variables and functions intelligently to minimize the need for comments. It should be immediately obvious what information it represents. Short names such as x and y are fine when referring to coordinates. Short names such as i and j are fine when doing loops.

Variable names should be "camelCase", starting with a lowercase word followed by uppercase word(s). C++ Class Names should be "CamelCase", using all uppercase word(s). C Functions that attempt to simulate namespacing, should be "nameSpace_camelCase".

All files and directories shall be lowercase and contain no spaces.

Code Style

Tabs should not be used when indenting. Setup your IDE or text editor to use 4 spaces.

Braces

For functions: please put each brace on a new line.

```
void function_definition(int argument)
{

}
```

For control statements: please put the first brace on the same line.

```
if (condition) {

}
```

Use exceptions sparingly.

Do not use ternary operator (?:) in place of if/else.

Do not repeat a variable name that already occurs in an outer scope.

Version Control

Being an open source project, developers can grab the latest code at any time and attempt to build it themselves. We try our best to ensure that it will build smoothly at any time, although occasionally we do break the build. In these instances, please provide a patch, pull request which fixes the issue or open an issue and notify us of the problem, as we may not be aware of it and we can build fine.

Try to group commits based on what they are related to: features/bugs/comments/graphics/commands/etc. . .

Comments

When writing code, sometimes there are items that we know can be improved, incomplete or need special clarification. In these cases, use the types of comments shown below. They are pretty standard and are highlighted by many editors to make reviewing code easier. We also use shell scripts to parse the code to find all of these occurrences so someone wanting to go on a bug hunt will be able to easily see which areas of the code need more love. Use the same convention as libembroidery.

libembroidery is written in C and adheres to C89 standards. This means that any C99 or C++ comments will show up as errors when compiling with gcc. In any C code, you must use:

```
/* C Style Comments */
/* TODO: This code clearly needs more work or further review. */
/* BUG: This code is definitely wrong. It needs fixed. */
/* HACK: This code shouldn't be written this way or I don't feel right about it. There may a
/* WARNING: Think twice (or more times) before changing this code. I put this here for a goo
/* NOTE: This comment is much more important than lesser comments. */
```

Bibilography

Donations

Creating software that interfaces with hardware is costly. A summary of some of the costs involved:

- Developer time for 2 core developers
- Computer equipment and parts
- Embroidery machinery
- Various electronics
- Consumable materials (thread, fabric, stabilizer, etc. . .)

If you have found our software useful, please consider funding further development by donating to the project on Open Collective.