Embroidermodder 1.90.0 Manual

Table of Contents

1. Introduction
2. Basic Features
3. Advanced Features
4. Other Projects
5. References

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)
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)

FORMAT	READ	WRITE	NOTES
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)
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

FORMAT	READ	WRITE	NOTES
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