What is Embroidermodder ?

Embroidermodder is a free machine embroidery application. The newest version, Embroidermodder 2 can:

- edit and create embroidery designs
- estimate the amount of thread and machine time needed to stitch a design
- convert embroidery files to a variety of formats
- upscale or downscale designs
- run on Windows, Mac and Linux

For more information, see our website.

Embroidermodder 2 is very much a work in progress since we're doing a ground up rewrite to an SDL2 GUI. The reasoning for this is detailed in the issues tab.

To see what we're focussing on at the moment check this table.

Date	*Event*
Dec 2021 - Jan 2022	libembroidery 1.0 work and bugfixing
31st of Jan 2022	libembroidery 1.0 will be released, then updates will slow down and the Embroidermodder 2 development version will be fixed to the API of this version.
Feb 2022	An overview of what has changed will be written up for the website as a news update, along with better documentation of libembroidery.
Feb-April	Finish the SDL2 conversion
April-May 2022	Finish all the targets in the Design, or assign them to 2.1.
May-June 2022	Bugfixing, Testing, QA
Summer Solstice (21st of June) 2022	Embroidermodder 2 is officially released.
July 2022	News and Documentation work for Embroidermodder 2

Build and Install

Dependencies

To build Embroidermodder 2 from source run:

On most systems:

```
./build.sh --get-dependencies
```

If you have a more unusual package installer, try:

`./build.sh --build-dependencies`

On Windows:

`.\build.bat --build-dependencies`

SDL2

We're working on an SDL2 version of the library that will require no non-standard dependencies not included in the source.

On systems where you use `--build-dependencies` the system will build and install the libraries if they are not already present from the versions in `extern/`. This way a copy of the Embroidermodder 2 source code on a machine with a build environment can be built without a connection to the internet access and insures against SDL2 going out of support.

Building

Assuming you have the above dependancies these commands should build `embroidermodder`

`./build.sh`

or (on Windows)

`.\build.bat`

with the install argument it will also install the program to user space

`./build.sh --install`

or (on Windows)

`.\build.bat --install`

Documentation

The documentation is in the form of the website (included in the `docs/` directory) and the printed docs in the three files:

- docs/libembroidery_0.1_manual.pdf
- docs/embroidermodder_1.90.0_user_manual.pdf
- docs/embroidermodder_1.90.0_developer_notes.pdf.

Development

Current work:

1. Converting C++ to C throughout.
    1. All comments to multiline `/* C-style comments */`.

    2. Replace variables with variables of C or libembroidery type. (QColor to EmbColor, QPointF to EmbVector)
    3. Reduce the reliance on Qt functions while allowing boot of the program.
    4. Turn settings into array type, to aid read/write in loops.
    5. QCheckBoxes into an array to simplify `Settings_Dialog::createTabOpenSave`.
2. OpenGL Rendering
    1. "Real" rendering to see what the embroidery looks like.
    2. Icons and toolbars.
    3. Menu bar
3. Libembroidery interfacing:
    1. Get all classes to use the proper libembroidery types within them. So `EllipseObject` has `EmbEllipse` as public data within it.
    2. Move calculations of rotation and scaling into `EmbVector` calls.
4. Get undo history widget back (BUG).
5. Switch website to a CMake build.
6. GUI frontend for embroider features that aren't supported by embroidermodder: flag selector from a table
7. Update all formats without color to check for edr or rgb files.
8. EmbroideryFLOSS - Color picker that displays catalog numbers and names
9. Setting for reverse scrolling direction (for zoom, vertical pan)
10. Stitching simulation
11. User designed custom fill
12. Keyboard zooming, panning
13. Advanced printing
14. Libembroidery 1.0
15. Better integrated help: I don't think the help should backend to a html file somewhere on the user's system. A better system would be a custom widget within the program that's searchable.
16. New embroidermodder2.ico 16x16 logo that looks good at that scale.
17. saving dst, pes, jef
18. Settings dialog: notify when the user is switching tabs that the setting has been changed, adding apply button is what would make sense for this to happen.
19. Update language translations
20. Replace KDE4 thumbnailer.
21. Import raster image
22. Statistics from 1.0, needs histogram.
23. SNAP/ORTHO/POLAR
24. Cut/copy allow post-selection
25. Layout into config
26. Notify user of data loss if not saving to an object format.
27. Add which formats to work with to preferences.
28. Cannot open file with # in the name when opening multiple files but works with opening a single file.
29. Closing settings dialog with the X in the window saves settings rather than

discarding them.
30. Otto theme icons: units, render, selectors, what's this icon doesn't scale
31. Layer manager and Layer switcher dock widget
32. test that all formats read data in correct scale (format details should match other programs).
33. Custom filter bug – doesn't save changes in some cases.

For more details read on into the Design section.

Sample Files

Various sample embroidery design files can be found in the embroidermodder2/samples folder.

Design

These are key bits of reasoning behind why the software is built the way it is.

CAD command review

1. scale
2. circle
3. offset
4. extend
5. trim
6. BreakAtPoint
7. Break2Points
8. Fillet
9. star
10. singlelinetext
11. Chamfer
12. split
13. area
14. time
15. pickadd
16. zoomfactor
17. product
18. program
19. zoomwindow
20. divide
21. find
22. record
23. playback
24. rotate
25. rgb
26. move
27. grid

28. griphot
29. gripcolor
30. gripcool
31. gripsize
32. highlight
33. units
34. locatepoint
35. distance
36. arc
37. ellipse
38. array
39. point
40. polyline
41. polygon
42. rectangle
43. line
44. arc (rt)
45. dolphin
46. heart

So, it means weighing up some simplifications.

Removed Elements

So I've had a few pieces of web infrastructure fail me recently and I think it's worth noting. An issue that affects us is an issue that can effect people who use our software.

googletests In development we attempted using googletests. Googletests require a web connection to update and they update on each compilation.

gtests are non-essential, testing is for developers not users so we can choose our own framework. I think the in-built testing for libembroidery was good and I want to re-instate it.

Qt and dependencies Downloading and installing Qt has been a pain for some users (46Gb on possibly slow connections).

I'm switching to SDL2 (which is a whole other conversation) which means we can ship it with the source code package meaning only a basic build environment is necessary to build it.

Social Platform Github is giving me a server offline (500) error and is still giving a bad ping.

So... all the issues and project boards etc. being on Github is all well and good assuming that we have our own copies. But we don't if Github goes down or

some other major player takes over the space and we have to move (again, since this started on SourceForge).

This file is a backup for that which is why I'm repeating myself between them.

Pandoc Documentation The documentation is, well better in that it's housed in the main repository, but I'm not a fan of the "write once build many" approach as it means trying to weigh up how 3 versions are going to render.

Can we treat the website being a duplicate of the docs a non-starter? I'd be happier with tex/pdf only and (I know this is counter-intuitive) one per project.

OpenGL

OpenGL rendering within the application. This will allow for Realistic Visualization - Bump Mapping/OpenGL/Gradients?

JSON data Ideas

JSON configuration (Started, see `head -n 50 src/mainwindow.cpp`.)

Ok this is changing slightly. embroidermodder should boot from the command line regardless of whether it is or is not installed (this helps with testing and running on machines without root). Therefore, it can create an initiation file but it won't rely on its existence to boot: this is what we currently do with settings.ini.

So:

1. Port `settings.ini` to `settings.json`.
2. Place `settings.json` in `$HOME/.embroidermodder` (or equivalent, see the homedir function in `gui.c`).
3. Parse JSON using cJSON (we have the new parseJSON function).
4. Better structure for settings data so parse and load JSON is easier and there's less junk in global variables. A structure similar to a Python dict that uses constants like the sketch below.

Why JSON over ini?

1. We need to hand-write *a* system because the current system is Qt dependent anyway.
2. This way we can store more complex data structures in the same system including the layout of the widgets which may be user configured (see Blender and GIMP).
3. Also it's easier to share information formatted this way between systems because most systems us JSON or XML data: there's better support for converting complex data this way.

Distribution

- Mac Bundle
- `.tar.gz` and `.zip` source archive.
- NSIS installer for Windows
- Debian package
- RPM package

Scripting Overhaul

Originally Embroidermodder had a terminal widget, this is why we removed it.

> ROBIN: I think supporting scripting within Embroidermodder doesn't make sense.
>
> All features that use scripting can be part of libembroidery instead. Users who are capable of using scripting won't need it, they can alter their embroidery files in CSV format, or import pyembroidery to get access. It makes maintaining the code a lot more complicated, especially if we move away from Qt. Users who don't want the scripting feature will likely be confused by it, since we say that's what libembroidery, embroider and pyembroidery are for.
>
> How about a simpler "call user shell" feature? Similar to texmaker we just call system on a batch or shell script supplied by the user and it processes the file directly then the software reloads the file. Then we aren't parsing it directly.
>
> I don't want to change this without Josh's support because it's a fairly major change.
>
> > JOSH: I totally agree.
> >
> > I like the idea of scripting just so people that know how to code could write their own designs without needing to fully build the app. Scripting would be a very advanced feature that most users would be confused by. Libembroidery would be a good fit for advanced features.

Perennial Jobs

1. Check for memory leaks
2. Clear compiler warnings on `-Wall -ansi -pedantic` for C.
3.

Developing for Android

https://developer.android.com/studio/projects/add-native-code

```
apt install google-android-ndk-installer cmake lldb gradle
```