3-Toed-Sloth Follow

May 21, 2022 · 9 min read · Listen

Save

# No Good Go GUI

Created Saturday 14 May 2022

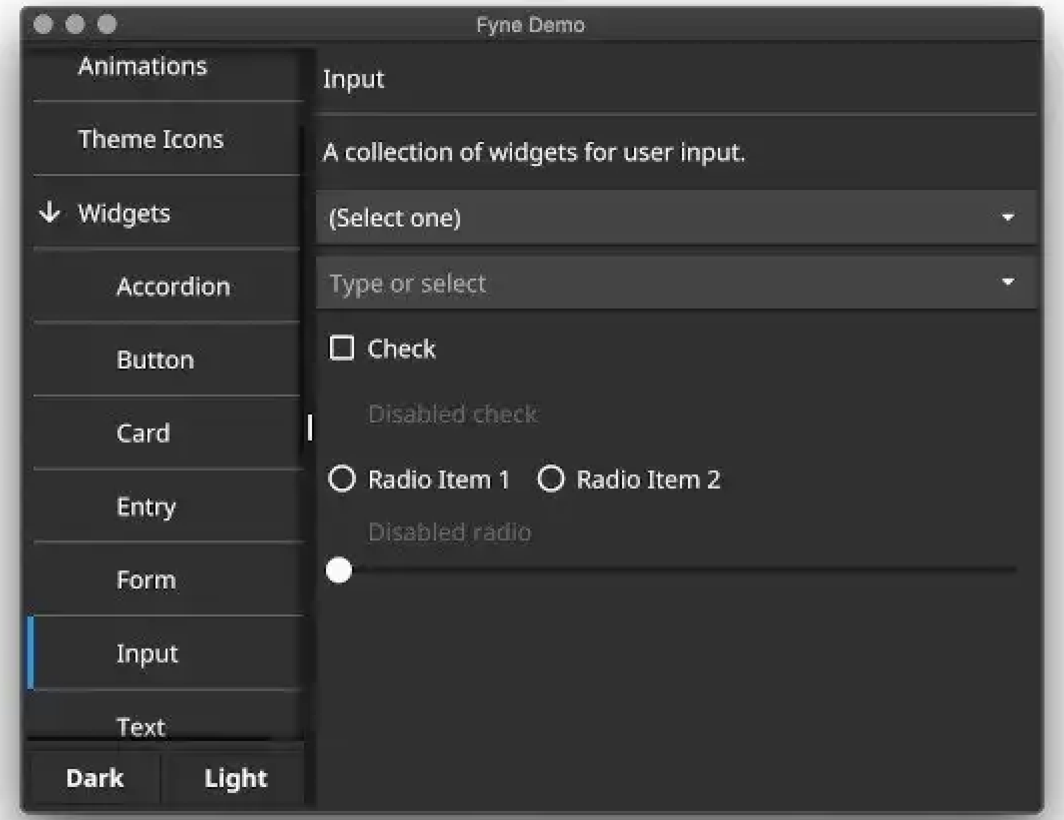
I’ve been programming since the 80s and recently evaluated the options for writing GUI programs in Go. I’m mostly interested in cross-platform desktop GUI software, although a mobile version using the same technology can be a appealing. Anything that allows a programmer save efforts is appealing.

The bottomline of my investigation is that there is not yet any good GUI option for Go. This is unfortunate, since otherwise Go would be the perfect replacement for easy-to-use rapid application development like I was used to with REALBasic — now known as the prohibitively expensive Xojo. No framework or language+IDE+framework combination so far has surpassed the productivity I experienced with REALbasic. I’ve tried Lazarus but Object Pascal is not a good replacement, the language simply shows its age and having to sometimes manage memory manually has many disadvantages for a desktop GUI application. I could have chosen another language from those I like and know well enough. Racket has a very complete GUI. I’ve written and maintained GUI applications in it for many years. However, I’ve moved away from it because GUI applications in it can feel sluggish and application startup time makes Racket a suboptimal choice for “snappy” small GUI applications. Ada feels like your investing time into technical debt and it lacks 3rd-party support. The only halfway good GUI option for it is GTK3. Even though I tried CommonLisp again and again over the years, it continues to feel hacky and I find its available 3rd-party libraries lacking documentation and truthworthiness. As for Rust, I’ve learned it, read Klabnik & Nichols’s book from start to end, and I have to admit that I find Rust unappealing and overengineered. Its usability for high integrity systems is doubtful — I’d still recommend Ada/Spark for that -, it offers no advantages for GUI programming, and has an overall toxic community.

So in the end Go was chosen for a larger project and some choice of a GUI library had to be made. Here is an opinionated overview of Go GUI frameworks, as of May 2022.

## Fyne

Website: https://fyne.io/
Tested: Yes, extensively.
Score: 5/10
Updates/fixes: often, fast
Recommendation: Only recommended for mobile applications

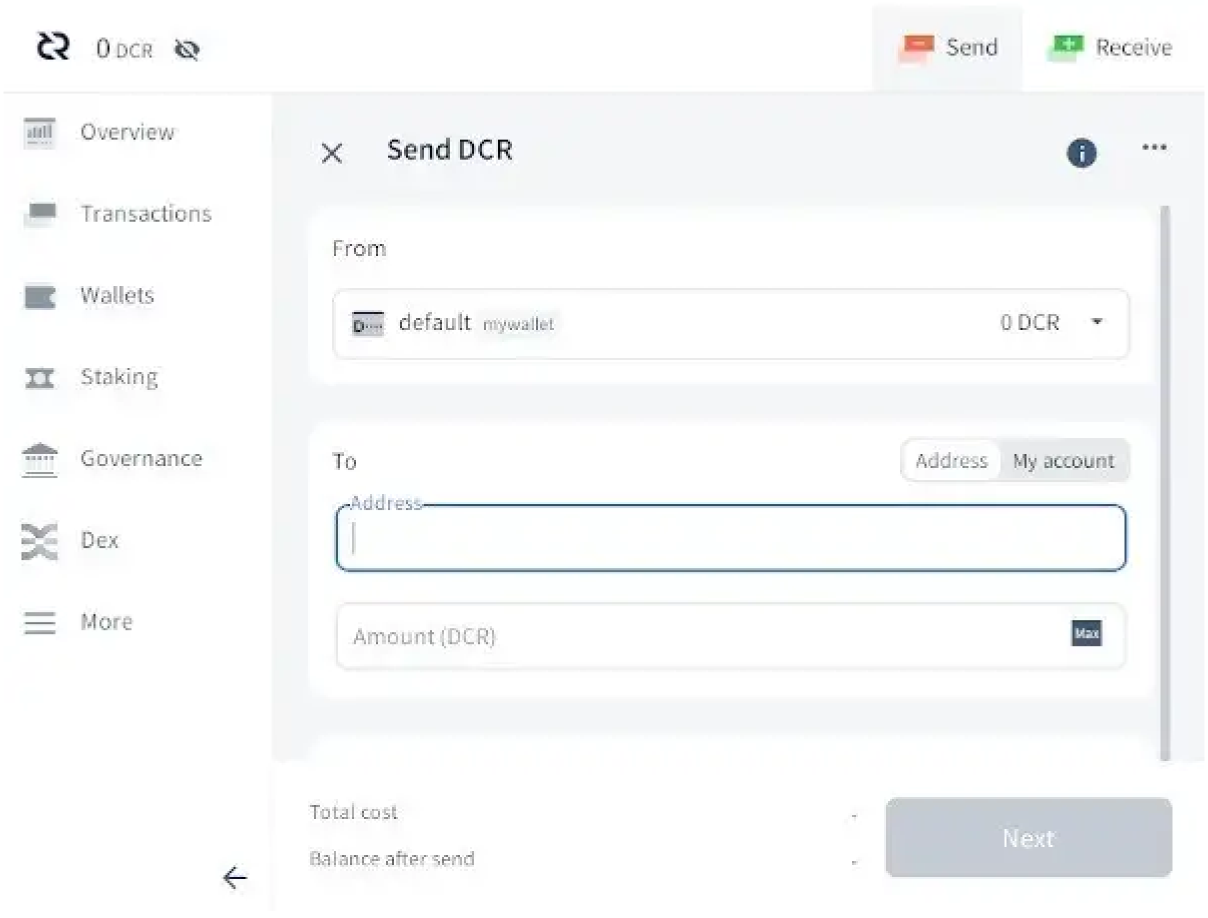


Fyne is basically the effort of one guy and a bunch of collaborators. The bottomline is that the Fyne developers are better at promoting their framework than at writing GUI frameworks. Fyne has *fundamental* design flaws. The biggest flaw is the idea that you can use the very same code for desktop and mobile projects. The fact that the form factor of phones is totally different should alone tell you that this makes no sense. Desktop applications *are* and *have to be* very different from mobile applications. But the Fyne developers don't think so. Consequently, they've made a number of dubious choices. They separate layout from widgets, which makes everything needlessly complicated. Fyne is not at all simple to use. The available layout mechanisms tend to minimize their contents and widgets only store a minimal size (the default for everything) and no preferred or maximum sizes. Widgets have no locking top, left, bottom, right or other reasonable mechanisms of how they should resize when their embedding window or panel is resized. There is also no Z-order for tabbing through widgets. Another big issue is that reasonable file path abstractions don't exist in Fyne. The developers insist that everything should be an URL, apparently unaware of the fact that the URL standard does not specify absolute paths with volume information in a cross-platform way. There are extensions of URI schemes with drive letters, and Fyne uses whatever they think is suitable, but these / path based translations from local filesystems are very problematic. How do you deal with removable drives? How do you deal with drive letters on Windows? How do you construct a path, traverse the file system? As of now, Fyne does nothing to help you with this and even insists on considering every URI stand for an arbitrary resource — you shouldn't assume that it points to a file, it could also be a web resource. As if web access and file access could even remotely be treated in the same way!

The result is unsuitable for desktop applications, although it might be fine on mobile platforms. It's hard to get a complex desktop design right. By default, Fyne applications violate every Human Interface Guideline on every platform, and the developers seem to be very proud of this. You're going to spend days to weeks creating your own specialized layout algorithms for simple things that would take minutes or hours in a reasonable, standard layout system.

### Gio

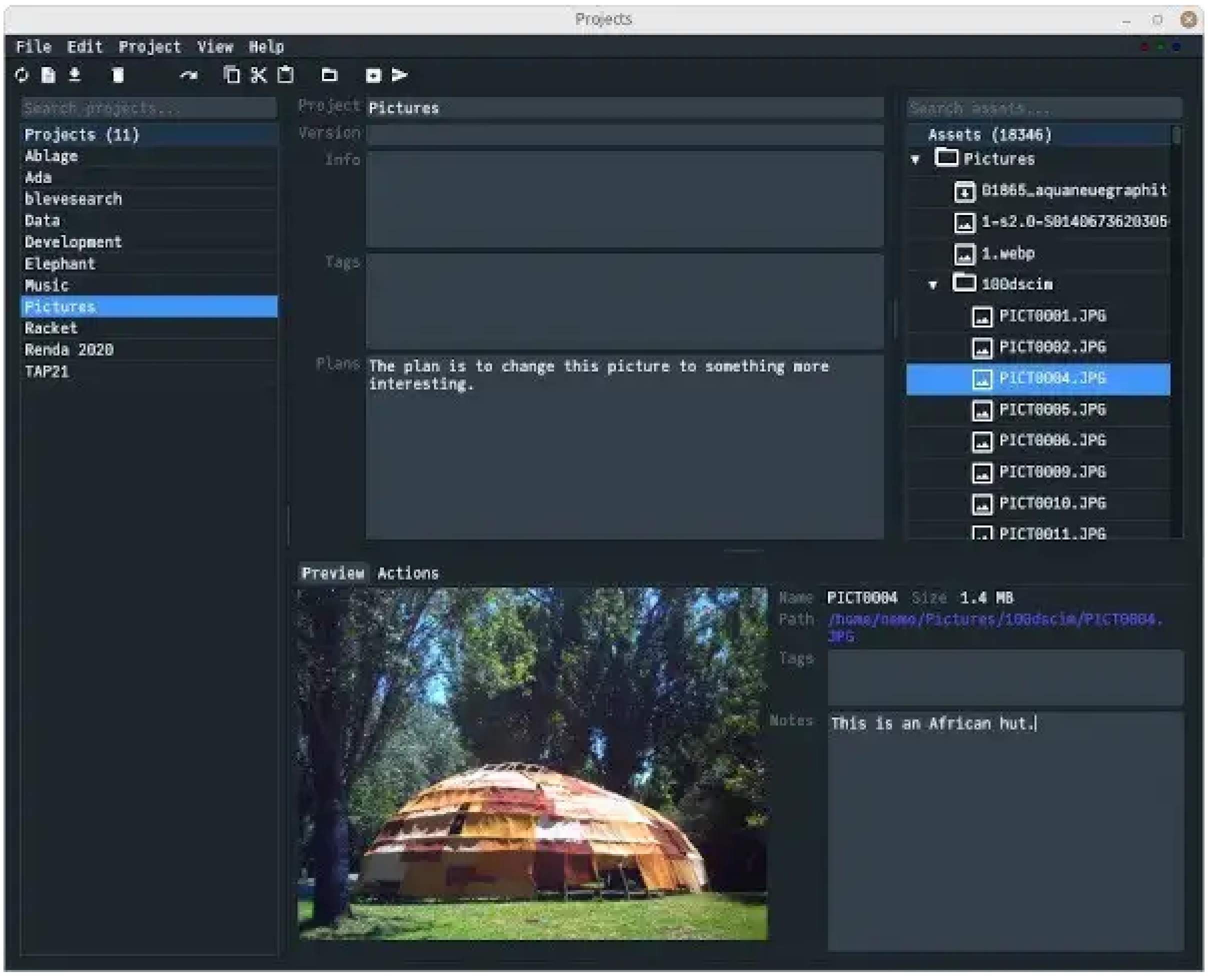
Website: https://gioui.org/
Tested: No.
Score: 6/10
Updates/fixes: moderate, slow
Recommendation: Promising but I haven't tested it in detail.



Gio is a very promising immediate mode GUI framework with some stateful widgets, too. Some people have reported it as a good alternative to Fyne, and it has most of the widgets you'd expect in an application. The usage appears kind of complex and cumbersome to me — if you look at the examples you'll see that it requires a decent amount of boilerplate code to even get started. The biggest drawback is a lack of documentation, however, which makes it hard to get started and might make you get lost initially. Still, I plan to give it a go sometime in the future, at least for testing it as an alternative to GIU. I haven't tested it for its editor control support — the abilities of frameworks to allow for complex text editing, multiline and ideally with rich text and even embedded images, is one of the lackmus tests for GUI frameworks because multiline text editing is by far the hardest to implement correctly. I haven't checked in detail but wager it will be similar to Giu in respect to that.

## Giu

Website: https://github.com/AllenDang/giu
Tested: Yes, extensively.
Score: 6/10
Updates/fixes: moderate, slow
Recommendation: Recommended for small projects but check carefully for limitations before committing.



Search projects...

Projects (11)
Ablage
Ada
blevesearch
Data
Development
Elephant
Music
Pictures
Racket
Renda 2020
TAP21

Project Pictures
Version
Info
Tags
Plans The plan is to change this picture to something more interesting.

Search assets...

Assets (18346)
Pictures
01865_aquaneuegraphit
1-s2.0-S0140673620306
1.webp
100dscim
PICT0001.JPG
PICT0002.JPG
PICT0004.JPG
PICT0005.JPG
PICT0006.JPG
PICT0009.JPG
PICT0010.JPG
PICT0011.JPG

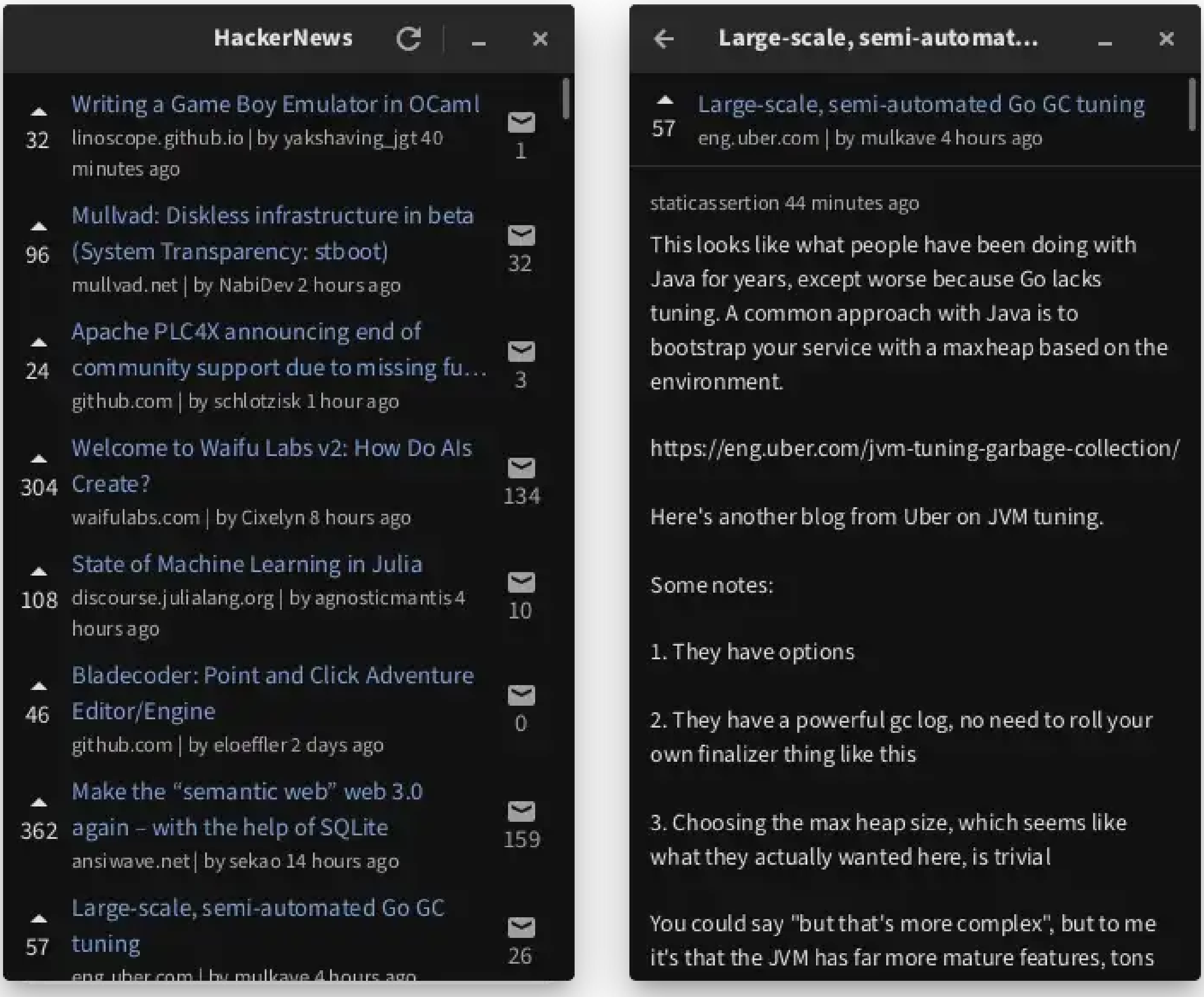
Name PICT0004 Size 1.4 MB
Path /home/nemo/Pictures/100dscim/PICT0004.JPG
Tags
Notes This is an African hut.

Giu is a lesser known immediate mode GIU based on a fork of imgui for Go, which uses the C library. It is very easy to use, probably got me a good looking result the fastest of all GUI frameworks I've tested, though not as fast as a traditional RAD tool like Lazarus. It has almost all widgets you might expect except for two rather big omissions: First, it does not have file and folder chooser/save dialogs and other common simple dialogs. I've managed to implement good replacements that are good enough for my use case, but you should be aware that not having native file dialogs can amount to a severe limitation and may come with other drawbacks (e.g. lack of integration with "remember last file access" features, drag & drop, etc.). Second, multiline text edit fields do not yet have automatic word wrapping. I've found a workaround to this but it's not online on Github yet. So you should be aware that this can be a pretty hefty limitation. However, unlike Fyne it gets all the defaults right and it's very easy to make a simple to moderately complex GUI with it, e.g. one containing a list, a few edit fields, buttons, combo or radio boxes, menus, and a window splitter or two. The look & feel is not native at all but overall pretty good. You have access to the underlying imgui but be aware that imgui itself is very rudimentary and the Go version does not come with advanced stateful controls like the C version.

## GTK

Website: https://github.com/gotk3/gotk3 (gtk3), https://github.com/diamondburned/gotk4 (gtk4)

Tested: Yes, GTK3.
Score: 7/10
Updates: auto-generated
Recommentation: GTK4 might currently be the best choice for a larger and complex desktop application that normally be written with Qt.



32 Writing a Game Boy Emulator in OCaml
linoscope.github.io | by yakshaving_jgt 40 minutes ago
1

96 Mullvad: Diskless infrastructure in beta (System Transparency: stboot)
mullvad.net | by NabiDev 2 hours ago
32

24 Apache PLC4X announcing end of community support due to missing fu…
github.com | by schlotzisk 1 hour ago
3

304 Welcome to Waifu Labs v2: How Do AIs Create?
waifulabs.com | by Cixelyn 8 hours ago
134

108 State of Machine Learning in Julia
discourse.julialang.org | by agnosticmantis 4 hours ago
10

46 Bladecoder: Point and Click Adventure Editor/Engine
github.com | by eloeffler 2 days ago
0

362 Make the “semantic web” web 3.0 again – with the help of SQLite
ansiwave.net | by sekao 14 hours ago
159

57 Large-scale, semi-automated Go GC tuning
26

57 Large-scale, semi-automated Go GC tuning
eng.uber.com | by mulkave 4 hours ago

staticassertion 44 minutes ago

This looks like what people have been doing with Java for years, except worse because Go lacks tuning. A common approach with Java is to bootstrap your service with a maxheap based on the environment.

https://eng.uber.com/jvm-tuning-garbage-collection/

Here's another blog from Uber on JVM tuning.

Some notes:

1. They have options

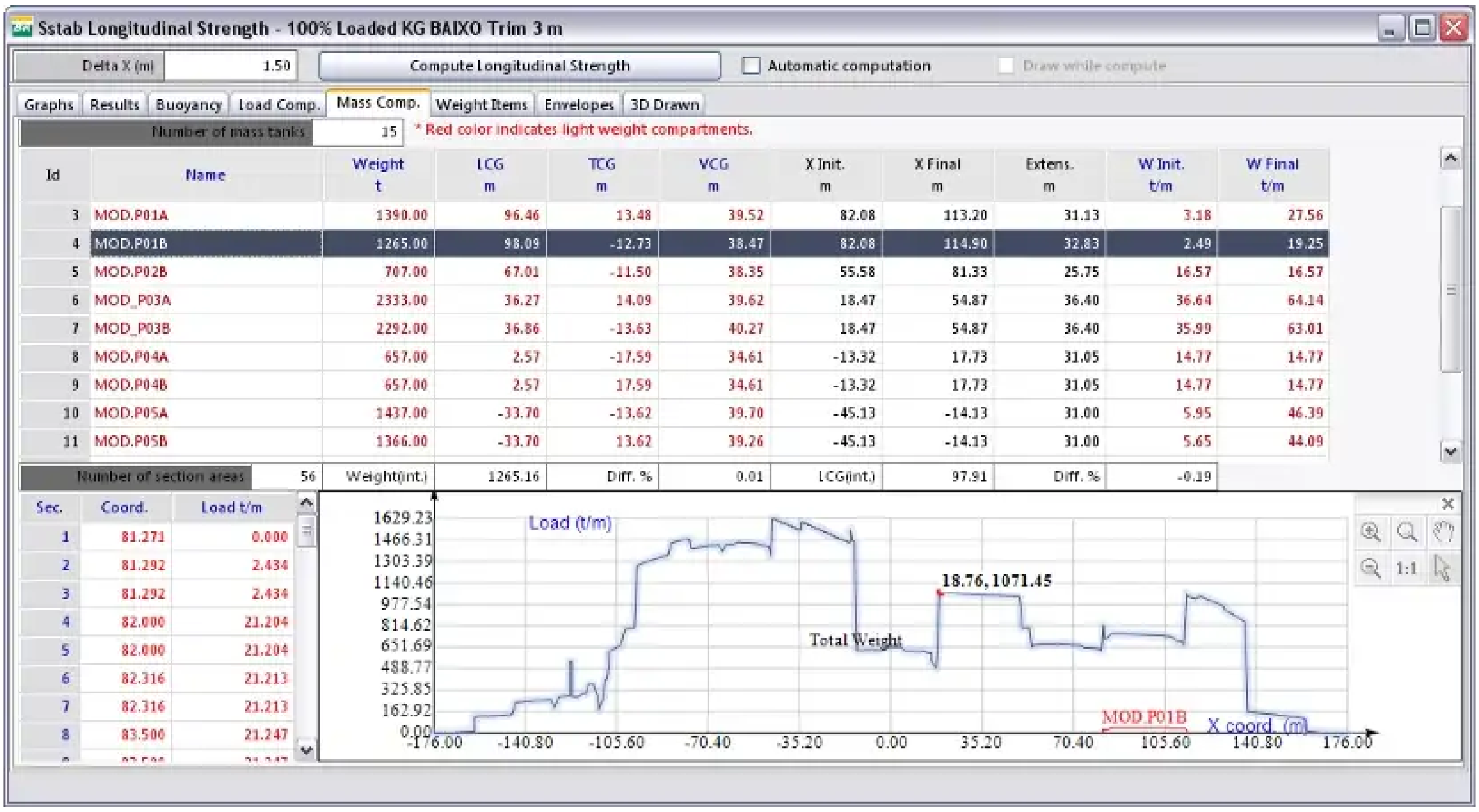
2. They have a powerful gc log, no need to roll your own finalizer thing like this

3. Choosing the max heap size, which seems like what they actually wanted here, is trivial

You could say "but that's more complex", but to me it's that the JVM has far more mature features, tons

## IUP

Website: https://github.com/matwachich/iup
Tested: No.
Score: 5/10
Updates: never
Recommendation: IUP has a good reputation for Linux and Windows, it might be worth a try as an alternative to GTK that is easier to deploy.


Sstab Longitudinal Strength - 100% Loaded KG BAIXO Trim 3 m

Delta X (m) 1.50 | Compute Longitudinal Strength | Automatic computation | Draw while compute

Number of mass tanks 15 * Red color indicates light weight compartments.

| Id | Name | Weight<br>t | LCG<br>m | TCG<br>m | VCG<br>m | X Init.<br>m | X Final<br>m | Extens.<br>m | W Init.<br>t/m | W Final<br>t/m |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | MOD.P01A | 1390.00 | 96.46 | 13.48 | 39.52 | 82.08 | 113.20 | 31.13 | 3.18 | 27.56 |
| 4 | MOD.P01B | 1265.00 | 98.09 | -12.73 | 38.47 | 82.08 | 114.90 | 32.83 | 2.49 | 19.25 |
| 5 | MOD.P02B | 707.00 | 67.01 | -11.50 | 38.35 | 55.58 | 81.33 | 25.75 | 16.57 | 16.57 |
| 6 | MOD_P03A | 2333.00 | 36.27 | 14.09 | 39.62 | 18.47 | 54.87 | 36.40 | 36.64 | 64.14 |
| 7 | MOD_P03B | 2292.00 | 36.86 | -13.63 | 40.27 | 18.47 | 54.87 | 36.40 | 35.99 | 63.01 |
| 8 | MOD.P04A | 657.00 | 2.57 | -17.59 | 34.61 | -13.32 | 17.73 | 31.05 | 14.77 | 14.77 |
| 9 | MOD.P04B | 657.00 | 2.57 | 17.59 | 34.61 | -13.32 | 17.73 | 31.05 | 14.77 | 14.77 |
| 10 | MOD.P05A | 1437.00 | -33.70 | -13.62 | 39.70 | -45.13 | -14.13 | 31.00 | 5.95 | 46.39 |
| 11 | MOD.P05B | 1366.00 | -33.70 | 13.62 | 39.26 | -45.13 | -14.13 | 31.00 | 5.65 | 44.09 |

Number of section areas 56 | Weight(int.) 1265.16 | Diff. % 0.01 | LCG(int.) 97.91 | Diff. % -0.19

| Sec. | Coord. | Load t/m |
|---|---|---|
| 1 | 81.271 | 0.000 |
| 2 | 81.292 | 2.434 |
| 3 | 81.292 | 2.434 |
| 4 | 82.000 | 21.204 |
| 5 | 82.000 | 21.204 |
| 6 | 82.316 | 21.213 |
| 7 | 82.316 | 21.213 |
| 8 | 83.500 | 21.247 |

## Qt

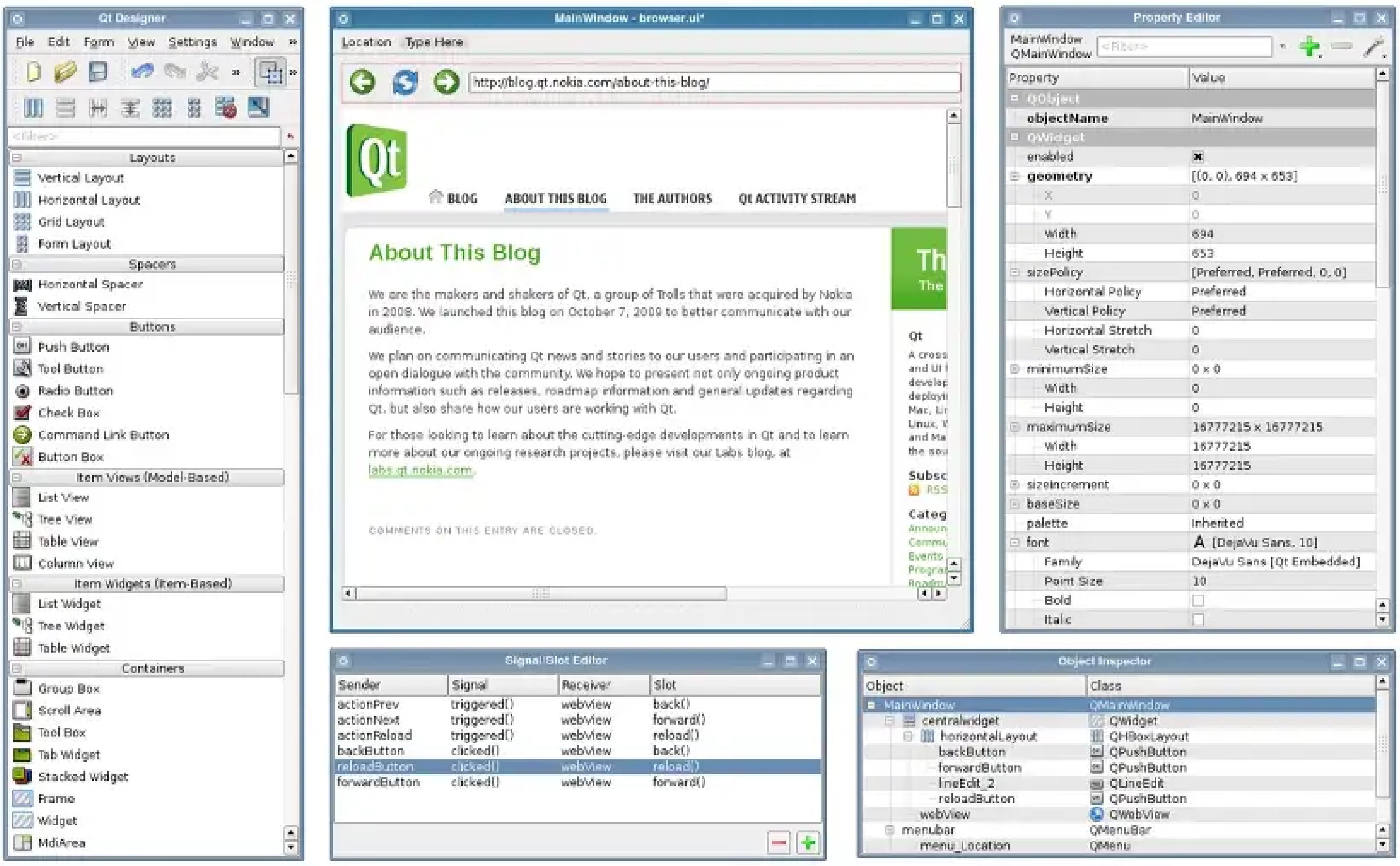
Website: https://github.com/therecipe/qt

Tested: No.

Updates/fixes: rare, slow

Score: 4/10 for proprietary software without commercial license, but might be a good choice for GPL'ed software

Recommendation: Only suitable for GPL software due to licensing issues.



About This Blog

We are the makers and shakers of Qt, a group of Trolls that were acquired by Nokia in 2008. We launched this blog on October 7, 2009 to better communicate with our audience.

We plan on communicating Qt news and stories to our users and participating in an open dialogue with the community. We hope to present not only ongoing product information such as releases, roadmap information and general updates regarding Qt, but also share how our users are working with Qt.

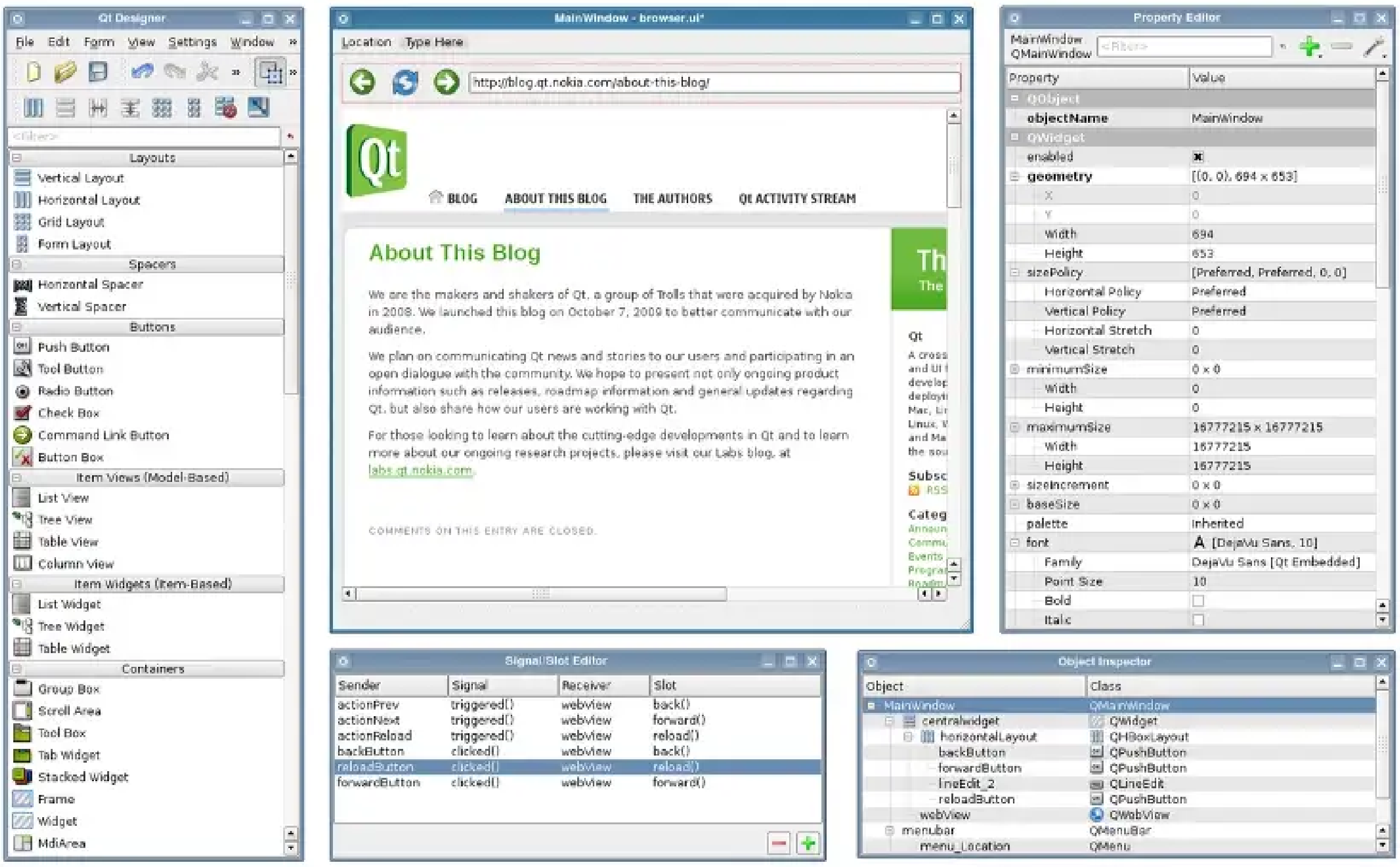
For those looking to learn about the cutting-edge developments in Qt and to learn more about our ongoing research projects, please visit our Labs blog, at labs.qt.nokia.com.

COMMENTS ON THIS ENTRY ARE CLOSED.

| Property | Value |
|---|---|
| QObject | |
| objectName | MainWindow |
| QWidget | |
| enabled | ☒ |
| geometry | [(0, 0), 694 x 653] |
| X | 0 |
| Y | 0 |
| Width | 694 |
| Height | 653 |
| sizePolicy | [Preferred, Preferred, 0, 0] |
| Horizontal Policy | Preferred |
| Vertical Policy | Preferred |
| Horizontal Stretch | 0 |
| Vertical Stretch | 0 |
| minimumSize | 0 x 0 |
| Width | 0 |
| Height | 0 |
| maximumSize | 16777215 x 16777215 |
| Width | 16777215 |
| Height | 16777215 |
| sizeIncrement | 0 x 0 |
| baseSize | 0 x 0 |
| palette | Inherited |
| font | A [DejaVu Sans, 10] |
| Family | DejaVu Sans [Qt Embedded] |
| Point Size | 10 |
| Bold | ☐ |
| Italic | ☐ |

| Sender | Signal | Receiver | Slot |
|---|---|---|---|
| actionPrev | triggered() | webView | back() |
| actionNext | triggered() | webView | forward() |
| actionReload | triggered() | webView | reload() |
| backButton | clicked() | webView | back() |
| reloadButton | clicked() | webView | reload() |
| forwardButton | clicked() | webView | forward() |

| Object | Class |
|---|---|
| MainWindow | QMainWindow |
| centralwidget | QWidget |
| horizontalLayout | QHBoxLayout |
| backButton | QPushButton |
| forwardButton | QPushButton |
| lineEdit_2 | QLineEdit |
| reloadButton | QPushButton |
| webView | QWebView |
| menubar | QMenuBar |
| menu_Location | QMenu |


About This Blog

We are the makers and shakers of Qt, a group of Trolls that were acquired by Nokia in 2008. We launched this blog on October 7, 2009 to better communicate with our audience.

We plan on communicating Qt news and stories to our users and participating in an open dialogue with the community. We hope to present not only ongoing product information such as releases, roadmap information and general updates regarding Qt, but also share how our users are working with Qt.

For those looking to learn about the cutting-edge developments in Qt and to learn more about our ongoing research projects, please visit our Labs blog, at labs.qt.nokia.com.

COMMENTS ON THIS ENTRY ARE CLOSED.

## Unison

Website: https://github.com/richardwilkes/unison

Tested: No

Updates/fixes: under development
Score: n/a
[demo build failed while compiling skia, so no screenshot is available]

I found this recently and cannot really judge this yet. This seems to be a nice non-native GUI with many standard widgets and is being developped for a personal project of the author. The code looks very clear & straightforward, so I would give it a try and will test it more extensively later. It almost certainly is not yet ready for production of a larger GUI project, though.

### Wails

Website: https://wails.app/
Tested: No.
Updates/fixes: often, fast
Score: 7/10
[no screenshot, it will look like whatever your chosen frontend looks like]

This is the only web-based GUI framework in the list, since this is not my main priority. It allows you to use Go for the backend and develop a GUI for it by using any web framework you like. Wails links the Go code automatically with Javascript in your HTML and renders the HTML/JS. The result is a full-fledged desktop application. I've seen several strong recommendations for this at various places and in combination with various frontend libraries on the HTML/JS side. If writing web applications is your thing anyway, I would definitely give this a try. My goal was to use a GUI library that allows me to develop everything in Go, and therefore such a framework was out of question. However, a web-based interface can look very good, and people already familiar with web design might find Wails a better alternative than any of the non-native, Go-only GUI libraries which are usually based some Open GL backend rendering engine. If you want your application to look like a web page (or might want to keep the option of making it a real web app), this is probably the easiest way to achieve that.

### Summary

The options are limited. I chose GIU for one project but could have just as well chosen Gio. I can't say I like immediate mode GUIs, though. You'll end up writing extra "layout buffer" glue code to maintain state if your GUI is getting more complex. For example, I get and set data from an sqlite database but Giu only supports strings in its text fields, so I have to maintain strings separately and get/update from the DB at the right time. Maybe GTK4 would have been the better choice and I'm thinking about switching to it in the end. I may also give Fyne a second chance in the future butmy hopes are not high. Fyne is opinionated but in the wrong direction. I’m also considering trying out Wails for a serious project. For me the learning curve would be substantial because my web design knowledge is from the late 90s. However, the rewards in terms of flexibility and redesign possibilities might make it worth it.

Golang GUI Programming Programming Tips UX

Medium

About Help Terms Privacy

Get the Medium app

Download on the App Store

GET IT ON Google Play