GUI library integration (QML.jl)

Ziyi Xi
Mentors: Bart Janssens and Shashi Gowda

March 2020

1 Introduction

QML.jl is a library to integrate Qt especially QML with the Julia programing language. By combining the widely used QML and the efficient Julia language, it could not only provide an alternative way for programming with QML, but also be more efficient comparing with using other programming languages. However, there could still be some improvements for this package such as reducing the programming difficulty by making the package more "Julian" or make the GUI more "Reactive" by writing the pure Julia code. In this proposal, I will talk about the ideas about doing that.

2 Ways to improve the current QML.jl

At the moment, QML.jl provides a way to read in QML, and use Julia as the backend. The use of QML provides what the interface looks like and some simple logic. The use of Julia mainly controls the signals and some more complicated logic. It has provided some awesome features like calling Julia functions from QML, read and set context properties from Julia and QML, emit signals from Julia to QML and use data models. These features have provided the needed components for the communication between QML and the pure Julia.

However, writing reactive GUI seems to be tedious by using the current package. For example, if we want to bind the text for the input box with the value of a slider, at the moment we have to do:

```
import QtQuick 2.0
import QtQuick.Controls 1.0
import QtQuick.Layouts 1.0

ApplicationWindow {
  id: root

  ColumnLayout {
    Slider {
      value: input
      onValueChanged: {
        input = value;
        output = 2*input;
      }
    }

    TextInput {
      text: output
      onTextEdited: {
        output = text;
        input = parseInt(text)/2;
      }
    }
  }

}
```

This is a simplified QML script to bind the values input and output. Note here if we want to bind the value with the slider and the text from the text input box, we will have to write two signal functions separately. It may seem to be easy to handle the example case. But image if we have lots of sliders and lots of text input boxes, should we still write so many signal functions? There must be a way to simplify this work, and the idea of QmlReactive is on the way!

Instead of writing QML script like this, we can do something similar to GtkReactive.jl, like this:

```
Using QMLReactive
```

```
win = Window("Testing") |> (bx = Box(:v));  # a window containing a
↪  vertical Box for layout
reactive_signal=signal()
sl = slider(1:100,signal=reactive_value)
tb = textbox(Int; signal=reactive_value/2)
push!(bx, sl);
push!(bx, tb);
showall()
```

That's it! Same functions, less code, easier to write, although it has not be implemented yet.

So how to implement it?

Luckily there is a very good feature in QML.jl: QQmlComponent. Using QQmlComponent, the QML code can be set from a Julia string wrapped in QByteArray, as noted in the documentation in QML.jl:

```
    qml_data = QByteArray("""
    import ...

    ApplicationWindow {
      ...
    }
    """)

    qengine = init_qmlengine()
    qcomp = QQmlComponent(qengine)
    set_data(qcomp, qml_data, "")
    create(qcomp, qmlcontext());

    # Run the application
    exec()
```

Firstly we can set an empty global QByteArray to store our generated QML script. And when we call functions like

```
sl = slider(1:100,signal=reactive_value)
```

We can insert the string:

```
 Slider {
     value: slider_status
   }
```

And in Julia, thanks to Observables.jl, we can write the code as:

```
using Observables
slider_status=Observable(0)
on(slider_status) do x
  do_things_when_slider_status_changes()
end
```

By using Observables.jl, the bi-direction communication is enabled. The communication flow will not just from the Julia to QML, but also from QML to Julia without writing the callback functions. So when the ”slider_status” is changed in the slider, assuming we have passed the same value to the text input box, the value there should also be changed.

Isn't it fantastic?

3 Goals and Milestones

To develop the QmlReactive package, I'd like to implement APIs described in GtkReactive.jl. It includes the input widgets:

- button
- checkbox
- togglebutton
- slider
- textbox
- textarea

- dropdown
- player

The output widgets:

- label

And also we can provide some reactive object from Observable.jl like MouseButton. It can be an enumerated object containing several Observable objects, with a set of functions on it. We can also provide something like mouseButton.x to the value of a slider.

4 What benefits can users and community members get from this package?

I want to help develop a package for users and community members to do the GUI programming in pure Julia. We don't have to call python's Pyqt5 to write the GUI anymore. It can also help people to show their results with their own GUI, as Julia is widely used in the science.

As for the milestones, if I am admitted into the program, I will have three months to work on the project. And my plan will be:

- develop QmlReactive.jl It will take one or two weeks to write the prototype and several weeks to finish the testing part.
- develop more widgets for QML.jl It will take several weeks to enrich QML.jl by supporting more widgets! (like integration with Plots.jl and DataFrames.jl)

5 Others I can contribute to QML.jl

Apart from the questions described above, it will also be interesting to combine Plots.jl and DataFrames.jl with QML. It will be straightforward to bind the data in DataFrames.jl with the chart in QML. And it will be nice if we show figures from Plots.jl in QML.

The difficult part here is that how to show the Plots.jl's figure in QML, without losing the interactive ability in Plots.jl. It should be easiest to support plotly since QML supports javascript. Using Pyplot seems to be more difficult but referring to the pyqt5 backend for matplotlib might be a way to go.

6 Timeline

- **before starting** Talk with advisors to set goals for each week, and get me more familiar with QML and QML.jl.
- **week 1** Write the prototype for the interactive widgets.
- **week 2-4** Finish writing a package QmlReactive package, and also finish writing the test.
- **week 5** Support DataFrames.jl directly for creating an editable TableView.
- **week 6-7** Support Plots.jl in QML.jl.
- **week 8** Write tests and do benchmark for the code that has been written.
- **week 9-11** Buffer time to solve problems, prettify the code, and do things that has not been considered in this proposal.

7 About me

Currently I am a second year Ph.D student in the department of computational mathematics, science and engineering from Michigan State University. And I get my bachelor degree in geophysics and dual degree in computer science from the University of Science and Technology of China.

I have some experience in programming Julia, Python, C++, Fortran, CUDA and Javascript. And since I am in a computational science major, I also have some background in numerical computing, that's the reason why I am so fond of Julia.

For my previous programming experience, I have widely used Julia in my research, such as the repo in `https://github.com/ziyixi/seisflow/tree/master/seisflow/julia/specfem_gll.jl/src/program` . There are also some interesting packages like `https://github.com/ziyixi/seisflow`, which is a package I develop for my research. Besides that, I have also worked on some interesting projects like `https://github.com/ziyixi/wechat_mpvue`, which I use Vue.js to develop a we-chat mini program.

It will be welcome for anyone to contact me. My email address is xiziyi@msu.edu and my github username is ziyixi. My website is `https://ziyixi.github.io/` which may still be in development.