GUIT

Document all the working process

https://github.com/winsa24/GuitATM

⚙️ Download from this Git repo …
Compile and Run `myapp.xcodeproj` ...
Follow the interface instruction ...
The correct password is “13579” 🔒
Enjoy ... 😊

Week 1

Install GUIT on my laptop, make it runnable.

> Attention: To run `.xcodeproj` , we need to synchronize all **OS version** on Xcode, including “General” and “Build Settings” columns.

Week 2 & 3

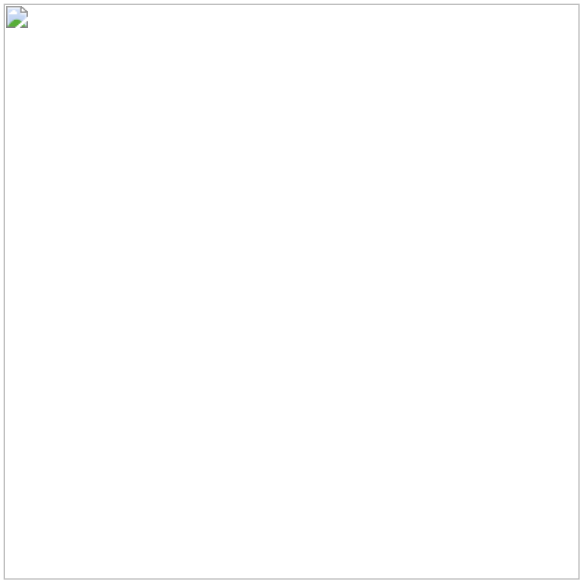
▼ Create workspace

- Code on `CodeTester.xcodeproj`
  - Hard to start : Want to create a simple label window, but don’t know should we start from `#include` or `*this<<Label()`.
  - No shortcuts : Textfield box restrict a lot of shortcuts, eg. *ctrl + A, ctrl + C, ctrl + V*.
  - Scrolling Stuck : Scrolling is not very smooth when selecting.
- Code on an existing app(?)
  - Make a copy of *CodeTester* folder
  - Modify small part of the code to understand better
  - Rename the fold, executable…
  - Structure the code : keep the main function (mostly code of the interface layout) and clean the interact functions

▼ Code Interface

Reference

> Abstract general layout, refer to GUIT component



Abstracted Layout

- Keyboard : ~~Table~~ VBox * HBox (inside Button)

  > Can be replaced with `Grid` gadget

- Panel : Label
- Cash port : Label
- Receipt : Label
- Card : Button
- Side Buttons : Button

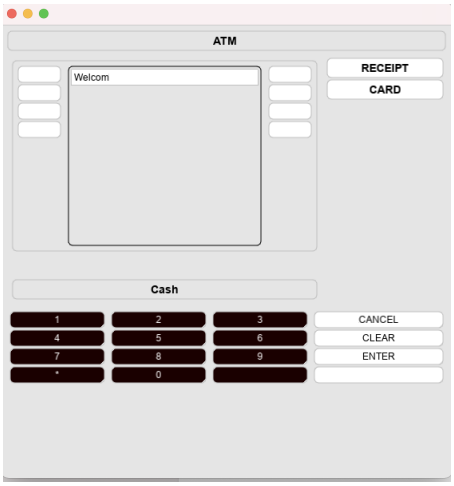
TODOs

- [x] ~~General Layout (Create all box)~~
- [x] ~~Adjust Box size~~
- [x] ~~Add component~~
- [ ] Adjust component position and size
- [ ] Add Interaction (Button function)

🥵 Although I started with existing code, it still takes me longer time than I expected to build my general layout. I want to keep (reuse) as many existing code as possible, but end up with spending more time figuring out the existing hierarchy.

🤯 And I spent tons of time trying to work on the CSS problem : try to make my side buttons bars come to the bottom. Although it's obviously a CSS issue, I tried to all the solutions on internet but none of them work. So I doubt on is there perhaps any override or restriction on the Guit component?

> Check `GFlex` ➡️ `src::/Tutorial/Presentation`



Interface

🧐 The return function is not so intuitive. For example, after I clicked on a button, I want to just disable the clicked one. But I the `enable(false)` applied to all the buttons…

▼ GUIT version iterate

> When I moved my runnable project folder directly into `Guit-1.2.0`, it first returned me an error about pid (not sure). But when I moved it back to `Guit-1.0.0` and successfully run it again. Then I moved it back to 1.2.0 again, and it worked. (Don't know the reason)

Week 4 & 5

▼ Solve previous issues

▼ VBox * HBox ⇒ Grid

More clear and intuitive

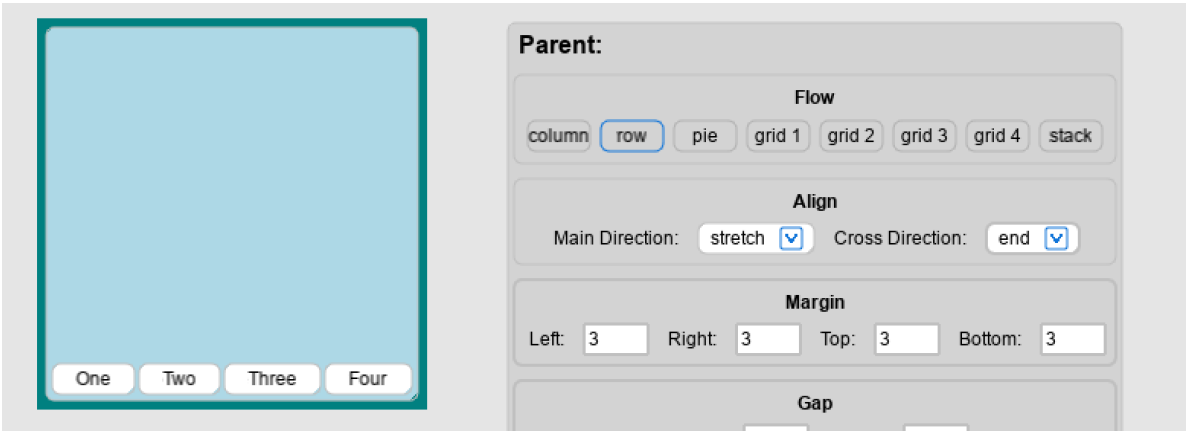
▼ [CSS] Align bottom 💥

- Wrt. code in *tutoiral/overview/layout.cpp*

❓ `box_align <<= ~main_align + " " + ~cross_align;` what is `< =` ?

> `<=` means dependency. Detail explanation check ➡️ `tutorial/binding`

```
auto box_flow = Flow();
*box_flow = "row";
auto box_align = Align();
box_align <<= "stretch end";
```



- Wrt. *tutorial/06_presentation.md*

❌ `<< (HBox("{border: rounded; flow: row; main-direction: stretch; cross-direction: end; }")`

❌ `<< (GFlex("{flow: row; main-direction: stretch; cross-direction: end; }")`

✅ `(HBox("{align: stretch end;}")`

I was so stuck with the “flow” layout. Didn’t realize that align to bottom is **align** function more than **layout**.

💡 Suggestion:

Clarify how to change the attribute to code. eg. Have some example code under the demo app

▼ Single Button Disable

`*en = true / false` doesn’t synchronize with definition `auto en = Enabled(true)`

Intuitively want to change the value by `*en = Enabled(false)`

Try to use **State** to adjust more situation ⬅️

`Guard(~sethour == true )`

▼ TODO list

- [x] ~~Insert card~~
- [x] ~~Delay display~~

checked ‘animation.cpp’

- [x] ~~Enter password~~

checked ‘tictactoe.cpp’ | found in ‘layout.cpp’

I want to have a function that only applied on number buttons to add chars, but don’t want to add individually, so I added to the parent grid box.

I want to override the parent function by adding on each functional buttons. But it doesn’t override.

❓

can’t override the parent callback

```
if (e->child()->textValue() != "*" && e->child()->textValue() != "\\#" )*psw += e->child()->textValue();
```

search GText function

can’t directly use normal string function (ie. str.size() / str.length())

need the sense of distinguish pointer and variable

want to delete the last character of a `Text()`. `psw->erase(-1);` doesn’t work.

current solution `psw->erase(psw->stringValue().size() - 1, 1);`

- [x] ~~Validate~~
- [ ] Wrong psw release card

```
*mesg = "<color=red> Wrong Password \n Have " + ~wrongCount + " attempts";
```

Can’t have `Int( )` been added directly in a string

but works like this

```
screen << "Have " + (3 - ~wrongCount) + " attempts";
```

▼ Question List:

1. how to simply get string size

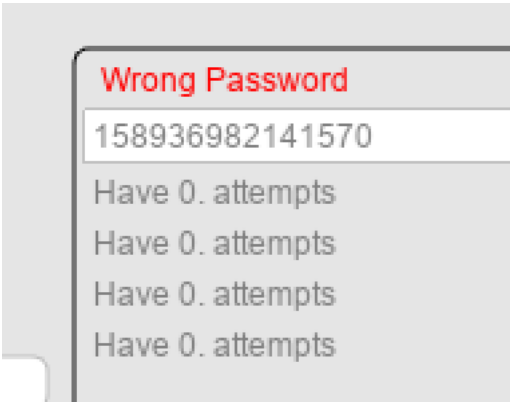
   > if only utf8 : **byteSize**()

2. how to show “#”

   > \\#

3. delete “.” after a int

   > Label() << Text(Int()) doesn’t contain “.”



Week 6 & 7

▼ SleepFor

> After verifying correct the password, the screen first show a confirmation (”Successful Login”), then in 2s, it changed into next instruction (”Choose operation”)

`On.click / SleepFor(1000)` can’t work inside a function

Tried C++ syntax:

```
#include <unistd.h>     //required for usleep()
...
// inside callback function
usleep(1000);
```

doesn't work neither.

▼ Bool

Follow the code example `file:///Users/weixiang/Downloads/Guit%203/doc/html/classguit_1_1_g_bool.html#:~:text=auto%20operator~,)`

Have code piece

`Button() << (~login == true ) / [ this ]{*mesg = "<color=blue> Select operation";  *err = "";}`

Get error

`Invalid operands to binary expression ('guit::GBoolFormula' and '(lambda at /Users/weixiang/Downloads/Guit 3/ATMMachine/ATM.cpp:137:43)')`

Only work with syntax

`<< On(~boolVariable == true) / actionOnItem()`

Can't be used in an `if` statement

```
auto depos = Bool(false);
...
// in a callback function
[=](auto){
  if(~depos == false)
```

`Error msg: Value of type 'guit::GBoolFormula' is not contextually convertible to 'bool'`

▼ StateMachine

Can't change state under a `if` sentence

```
<< [=](auto e){
         if(~psw == ~userPsw) {
             << (auth >> ops)
            << setState(ops)
          }
  }
<< Guard(~psw == ~userPsw) / (auth >> ops). //x
```

▼ Remove child

Remove all children : `screen->removeChildren();`

How to remove a specific child?

▼ Callback function (on Buttons)

Syntax

Lambda function

Status detect

Guard => like a if when the button is clicked
On => activated when the value of auth or idle changes

Week 8

▼ Modify state machine

To have proper display

▼ Disable buttons to avoid wrong click

```
// Guard => like a if when the button is clicked

// On => activated when the value of auth or idle changes
```

move `<< Guard(~wrongCount > 3) / (auth >> idle)` to [Enter] Button.

Embed links

💡 Tutorial : *file:///Users/weixiang/Downloads/Guit%203/doc/Documentation.html*

file:///Users/weixiang/Downloads/Guit%203/doc/Documentation.html

Code backup: (mix : bool state + state machine)

```
//
//  Guit GUI Code Tester
//  Copyright © 2022 Eric Lecolinet. All rights reserved.
//  http://www.telecom-paris.fr/~elc
//

#include <guit.hpp>
#include <gcodereader.hpp>
#include <gcodeblock.hpp>
#include <ginspector.hpp>
#include <gstatechart.hpp>
using namespace guit;
using namespace std;

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

class ATM : public GWindow {
public:
  ATM();
  ~ATM();
  void readScript(GString const& filename);
  void writeScript(GString const& filename);
  void build();

private:
  GBool changed;
  GText errors, filename;
  gptr<class GCodeBlock> codeblock;
};

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

ATM::~ATM() {
  auto inspector = GInspector::inspector();
  inspector->setInspectedCB(nullptr);
  inspector->setInspectingCB(nullptr);
}

ATM::ATM() : GWindow("#code-edit") {
  GApp()
    << "#screen {size: 30ch 15ch; border: rounded dashed}"
    << "#keyboard {size: 80ch 12ch}"
    << ".window {size:80ch 40ch}"
    << ".label {font: bold 14px Arial,sans-serif; format: 2 2 0; "
    "                  size: 4ch auto; align: center center; border: rounded}"
    << ".nobtn Button {background: #1a0000; color: white;}"
```

```
    << ".sidebtn Button {background: #000000;}"

    << ".clock .buttonbox {size: auto 7ch; align: center stretch}"
    << ".label .buttonbox .set-btn {size:6ch; font:bold}";

//    auto userPsw = "13579";
    auto userPsw = Text("13579");


    auto statemac = StateChart();      // creates a state machine

    auto init = SuperState(statemac);  // hierarchical state belonging to 'statemac'
    statemac->setInitial(init);        // 'init' is the initial state of 'statemac'

    auto idle = State(init);           // substate of the 'init' state
    init->setInitial(idle);            // 'idle' is the initial state of 'init'
    auto auth      = State(init);
    auto ops    = State(init);
    auto exit  = State(init);
    auto withdraw    = State(init);
    auto deposit     = State(init);

    auto display = Label("Welcome")
    << On(~idle == true)         / ("Please insert a card")
    << On(~auth == true)    / ("Please enter password")
    << On(~ops == true)  / ("Select operation")
    << On(~exit == true)    / ("Safely Exit")
    << On(~withdraw == true)     / ("Withdraw")
    << On(~deposit == true)  / ("Deposit");

    auto login = Bool(false);
    auto hasCard = Bool(false);
    auto depos = Bool(false);
    auto withd = Bool(false);

    auto mesg = Label("Welcome");
    auto psw = Text();
    auto err = Text();
    auto sideTextL = VBox()
    << On(~login == true) / "Deposit";
    auto sideTextR = VBox()
    << On(~login == true) / "Withdraw";

    auto screen = Box("#screen {flow: column; align: stretch stretch;}")
        << display
        << mesg
        << psw
        << (Label() << err)
        << (HBox("{flow: row; align: stretch end;}")
            << sideTextL
            << Box()
            << sideTextR
            );


    auto bg = Background("pink");
    auto bd = Border("rounded");

    auto en = Enabled(false);

    auto wrongCount = Int(0);
    auto cashBox = Box();
    auto cashCount = Int();
//    auto callback = [=](auto e){
//        *psw += e->from()->textValue();
//    };
    auto interactMoney = [=](auto e){

    };
    auto cancelBtn = [=](auto e){
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "Deposit";
        sideTextR << "Withdraw";
    };
    auto confirmBtn = [=](auto e){
        *en = true;
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "Exit";
        sideTextR << "Continue";
```

```
    };
    auto depositMoney = [=](auto e){
//        if(~withd == true){
//            sideTextL->removeChildren();
//            sideTextR->removeChildren();
//            sideTextL << "Deposit";
//            sideTextR << "Withdraw";
//        }
        *depos = true;
        *mesg = "deposit";
        cashBox << bg << bd << interactMoney;
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "Cancel";
        sideTextR << "Confirm";
    };
    auto withdrawMoney = [=](auto e){
        *withd = true;
        screen->remove(psw);
        *mesg = "Select the amount of money";
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "10" << "20" << "50" << "Cancel";
        sideTextR << "100" << "300" << "Others" << "";
    };


  // main box
  *this
    << QuitDialog("Quit","Changes will be lost", &changed)
    << (VBox(".window {align: stretch stretch;}")
//        << On(~hasCard == false)  / (*en = true)
        << On(~wrongCount > 3)  / [en] {*en = false;}
        << Label(".label ATM ")
        << (HBox()
            << (VBox()
                << (HBox("{align: stretch end;}")
                    << (VBox()
                        << On(~login == true) / Enabled(true)
                        << On(~login == false) / Enabled(false)
                        << Button("") << Button() << Button()
                        << (Button("{#deposit}") << (ops >> deposit) << depositMoney << On(~depos == true || ~withd == true)/cancelBtn)
                        )
                    << screen
                    << (VBox()
                        << On(~login == true) / Enabled(true)
                        << On(~login == false) / Enabled(false)
                        << Button("") << Button("") << Button()
                        << (Button("{#withdraw}") << (ops >> withdraw) << withdrawMoney << On(~depos == true || ~withd == true)/confirmBtn)
                        )
                << (VBox()
                    << Label(".label Cash")
                    << cashBox
                    )
                )
            << (VBox()
                << (Button(".label RECEIPT")
                    << en
                    << On.click / [=](GMouseEvent* e){ *mesg = "Print RECEIPT"; }
                    )
                << (Button(".label CARD")
                    << (idle >> auth)
                    << On.click
                    / [=]{ *hasCard = true; *mesg = "<color=green> Card Inserted"; }

                    // waits for 1 second whitout blocking the event loop
                    / SleepFor(2000)

                    // done once the delay is over
                    / [=]{ *mesg = "<color=blue> Enter Password"; }
                        << Box()
                   )
                )
            )
        << (HBox("#keyboard {align: stretch stretch;}")
            << (Box("#noboard {flow: grid 3}")
                << [=](auto e){ if(e->child()->textValue() != "*" && e->child()->textValue() != "\\#" )*psw += e->child()->textValue();}
//                << Guard(~auth == true)   / Enabled(true)
//                << Guard(~idle == true)   / Enabled(false)
                << On(~hasCard == true) / Enabled(true)
                << On(~hasCard == false) / Enabled(false)
```

```
                << Button("1") << Button("2") << Button("3")
                << Button("4") << Button("5") << Button("6")
                << Button("7") << Button("8") << Button("9")
                << Button("*") << Button("0") << (Button("\\#") << []{})
                )
            << (Box("#funboard {flow: grid 1}")
                << On(~hasCard == true) / Enabled(true)
                << On(~hasCard == false) / Enabled(false)
                << (Button("CANCEL") << Background("yellow") << [=](auto e){ psw->erase(psw->stringValue().size() - 1, 1); })
                << (Button("CLEAR") << bg << [=](auto e){ *psw = ""; })
                << (Button("ENTER") << Background("green")
                    << Guard(~psw == ~userPsw) / (auth >> ops)
                    << [=](auto e){
                            if(psw->stringValue() == userPsw->stringValue()) {
//                                << (autg >> ops)
//                                *mesg = "<color=green> Login Successfully";
//                                screen->removeChildren();
//                                screen << display
//                                    << mesg
//                                    << "select operation"
//                                    << (HBox("{flow: row; align: stretch end;}")
//                                        << sideTextL
//                                        << Box()
//                                        << sideTextR
//                                        );
//                                *login = true;

//                                // c++
//                                << SleepFor(1000) //callback
//                                << On(~login == true) /[=]{*mesg = "<color=green> s";}
                            }
                            else { ++ wrongCount;
                                *mesg = "<color=red> Wrong Password";
                                *err = "Left " + to_string(4 - *wrongCount) + " Attempts";
                            }
                            *psw = "";
                        }

                    )
                << Button("")
                )
            )

    );
    // starts the state machine
    statemac->start();
    makeInternal();   // disable inspection
}



int main(int argc, const char** argv)
{
  GApp app(argc, argv);
  app.canInspect();

  return app.start(new ATM);
}
```

Code backup (State machine):

```
//
//  Guit GUI Code Tester
//  Copyright © 2022 Eric Lecolinet. All rights reserved.
//  http://www.telecom-paris.fr/~elc
//

#include <guit.hpp>
#include <gcodereader.hpp>
#include <gcodeblock.hpp>
#include <ginspector.hpp>
#include <gstatechart.hpp>
using namespace guit;
using namespace std;
```

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

class ATM : public GWindow {
public:
  ATM();
  ~ATM();
  void readScript(GString const& filename);
  void writeScript(GString const& filename);
  void build();

private:
  GBool changed;
  GText errors, filename;
  gptr<class GCodeBlock> codeblock;
};

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

ATM::~ATM() {
  auto inspector = GInspector::inspector();
  inspector->setInspectedCB(nullptr);
  inspector->setInspectingCB(nullptr);
}

ATM::ATM() : GWindow("#code-edit") {
  GApp()
    << "#screen {size: 30ch 15ch; border: rounded dashed}"
    << "#keyboard {size: 80ch 12ch}"
    << ".window {size:80ch 40ch}"
    << ".label {font: bold 14px Arial,sans-serif; format: 2 2 0; "
    "                   size: 4ch auto; align: center center; border: rounded}"
    << ".nobtn Button {background: #1a0000; color: white;}"
    << ".sidebtn Button {background: #000000;}"

    << ".clock .buttonbox {size: auto 7ch; align: center stretch}"
    << ".label .buttonbox .set-btn {size:6ch; font:bold}";

//    auto userPsw = "13579";
    auto userPsw = Text("13579");


    auto statemac = StateChart();      // creates a state machine

    auto init = SuperState(statemac);  // hierarchical state belonging to 'statemac'
    statemac->setInitial(init);        // 'init' is the initial state of 'statemac'

    auto idle = State(init);           // substate of the 'init' state
    init->setInitial(idle);            // 'idle' is the initial state of 'init'
    auto auth      = State(init);
    auto ops    = State(init);
    auto exit  = State(init);
    auto withdraw    = State(init);
    auto deposit     = State(init);

    auto display = Label("Welcome")
    << On(~idle == true)         / ("Please insert a card")
    << On(~auth == true)    / ("Please enter password")
    << On(~ops == true)  / ("Select operation")
    << On(~exit == true)    / ("Safely Exit")
    << On(~withdraw == true)     / ("Withdraw")
    << On(~deposit == true)  / ("Deposit");

    auto mesg = Label("Welcome");
    auto psw = Text();
    auto err = Text();
    auto sideTextL = VBox();
    auto sideTextR = VBox();

    auto screen = Box("#screen {flow: column; align: stretch stretch;}")
        << display
        << mesg
        << psw
        << (Label() << err)
        << (HBox("{flow: row; align: stretch end;}")
            << sideTextL
            << Box()
            << sideTextR
            );
```

```
  auto bg = Background("pink");
  auto bd = Border("rounded");

  auto en = Enabled(false);

  auto wrongCount = Int(0);
  auto cashBox = Box();
  auto cashCount = Int();

  auto interactMoney = [=](auto e){

  };
  auto cancelBtn = [=](auto e){
      sideTextL->removeChildren();
      sideTextR->removeChildren();
      sideTextL << "Deposit";
      sideTextR << "Withdraw";
  };
  auto confirmBtn = [=](auto e){
      *en = true;
      sideTextL->removeChildren();
      sideTextR->removeChildren();
      sideTextL << "Exit";
      sideTextR << "Continue";
  };
  auto depositMoney = [=](auto e){
      *mesg = "deposit";
      cashBox << bg << bd << interactMoney;
      sideTextL->removeChildren();
      sideTextR->removeChildren();
      sideTextL << "Cancel";
      sideTextR << "Confirm";
  };
  auto withdrawMoney = [=](auto e){
      screen->remove(psw);
      *mesg = "Select the amount of money";
      sideTextL->removeChildren();
      sideTextR->removeChildren();
      sideTextL << "10" << "20" << "50" << "Cancel";
      sideTextR << "100" << "300" << "Others" << "";
  };


// main box
*this
  << QuitDialog("Quit","Changes will be lost", &changed)
  << (VBox(".window {align: stretch stretch;}")
      << Guard(~wrongCount > 3)  / (auth >> idle)
      << Label(".label ATM ")
      << (HBox()
          << (VBox()
              << (HBox("{align: stretch end;}")
                  << (VBox()
                      << Button("") << Button() << Button()
                      << (Button("{#deposit}") << (ops >> deposit) << depositMoney)
                      )
                  << screen
                  << (VBox()
                      << Button("") << Button("") << Button()
                      << (Button("{#withdraw}") << (ops >> withdraw) << withdrawMoney)
                      )
                  )
              << (VBox()
                  << Label(".label Cash")
                  << cashBox
                  )
              )
          << (VBox()
              << (Button(".label RECEIPT")
                  << en
                  << On.click / [=](GMouseEvent* e){ *mesg = "Print RECEIPT"; }
                  )
              << (Button(".label CARD")
                  << (idle >> auth)  // is working
                  << On.click
                  / [=]{ *mesg = "<color=green> Card Inserted"; }
                  // waits for 1 second whitout blocking the event loop
                  / SleepFor(2000)
                  // done once the delay is over
                  / [=]{ *mesg = "<color=blue> Enter Password"; }
                      << Box()
```

```
                   )
                )
            )
        << (HBox("#keyboard {align: stretch stretch;}")
            << (Box("#noboard {flow: grid 3}")
                << Guard(~auth == true)   / Enabled(true)
                << Guard(~idle == true)   / Enabled(false)  // not working
                << [=](auto e){ if(e->child()->textValue() != "*" && e->child()->textValue() != "\\#" )*psw += e->child()->textValue();}
                << Button("1") << Button("2") << Button("3")
                << Button("4") << Button("5") << Button("6")
                << Button("7") << Button("8") << Button("9")
                << Button("*") << Button("0") << (Button("\\#") << []{})
                )
            << (Box("#funboard {flow: grid 1}")
                << (Button("CANCEL") << Background("yellow") << [=](auto e){ psw->erase(psw->stringValue().size() - 1, 1); })
                << (Button("CLEAR") << bg << [=](auto e){ *psw = ""; })
                << (Button("ENTER") << Background("green")
                    << Guard(~psw == ~userPsw) / (auth >> ops) // not working
                    << [=](auto e){
                            if(psw->stringValue() == userPsw->stringValue()) {
//                                *mesg = "<color=green> Login Successfully";
                            }
                            else { ++ wrongCount;
//                                *mesg = "<color=red> Wrong Password";
                               *err = "Left " + to_string(4 - *wrongCount) + " Attempts";
                            }
                            *psw = "";
                        }
                    )
                << Button("")
                )
            )
    );
    // starts the state machine
    statemac->start();
    makeInternal();   // disable inspection
}



int main(int argc, const char** argv)
{
  GApp app(argc, argv);
  app.canInspect();

  return app.start(new ATM);
}
```

Backup code : [Bool Value]

```
//
//  Guit GUI Code Tester
//  Copyright © 2022 Eric Lecolinet. All rights reserved.
//  http://www.telecom-paris.fr/~elc
//

#include <guit.hpp>
#include <gcodereader.hpp>
#include <gcodeblock.hpp>
#include <ginspector.hpp>
using namespace guit;
using namespace std;

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

class ATM : public GWindow {
public:
  ATM();
  ~ATM();
  void readScript(GString const& filename);
  void writeScript(GString const& filename);
  void build();

private:
  GBool changed;
  GText errors, filename;
```

```
  gptr<class GCodeBlock> codeblock;
};

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

ATM::~ATM() {
  auto inspector = GInspector::inspector();
  inspector->setInspectedCB(nullptr);
  inspector->setInspectingCB(nullptr);
}

ATM::ATM() : GWindow("#code-edit") {
  GApp()
    << "#screen {size: 30ch 15ch; border: rounded dashed}"
    << "#keyboard {size: 80ch 12ch}"
    << ".window {size:80ch 40ch}"
    << ".label {font: bold 14px Arial,sans-serif; format: 2 2 0; "
    "                  size: 4ch auto; align: center center; border: rounded}"
    << ".nobtn Button {background: #1a0000; color: white;}"
    << ".sidebtn Button {background: #000000;}"

    << ".clock .buttonbox {size: auto 7ch; align: center stretch}"
    << ".label .buttonbox .set-btn {size:6ch; font:bold}";

    auto userPsw = Text("13579");

    auto login = Bool(false);
    auto hasCard = Bool(false);
    auto depos = Bool(false);
    auto withd = Bool(false);

    auto mesg = Label("Welcome");
    auto psw = Text();
    auto err = Text();
    auto sideTextL = VBox();
    auto sideTextR = VBox();

    auto screen = Box("#screen {flow: column; align: stretch stretch;}")
        << mesg
        << psw
        << (Label() << err)
        << (HBox("{flow: row; align: stretch end;}")
            << sideTextL
            << Box()
            << sideTextR
            );


    auto bg = Background("pink");
    auto bd = Border("rounded");

    auto en = Enabled(false);

    auto wrongCount = Int(0);
    auto cashBox = Box();
    auto cashCount = Int();

    auto interactMoney = [=](auto e){

    };
    auto cancelBtn = [=](auto e){
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "Deposit";
        sideTextR << "Withdraw";
        *depos = false;
        *withd = false;
    };
    auto confirmBtn = [=](auto e){
        *en = true;
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "Exit";
        sideTextR << "Continue";
    };
    auto depositMoney = [=](auto){
        if(*depos == false){
            *depos = true;
            *mesg = "deposit";
            cashBox << bg << bd << interactMoney;
            sideTextL->removeChildren();
```

```
            sideTextR->removeChildren();
            sideTextL << "Cancel";
            sideTextR << "Confirm";
        }
    };
    auto withdrawMoney = [=](auto e){
        *withd = true;
        screen->remove(psw);
        *mesg = "Select the amount of money";
        sideTextL->removeChildren();
        sideTextR->removeChildren();
        sideTextL << "10" << "20" << "50" << "Cancel";
        sideTextR << "100" << "300" << "Others" << "";
    };


  // main box
  *this
    << QuitDialog("Quit","Changes will be lost", &changed)
    << (VBox(".window {align: stretch stretch;}")
//        << On(~hasCard == false)  / (*en = true)
        << On(~wrongCount > 3)  / [en] {*en = false;}
        << Label(".label ATM ")
        << (HBox()
            << (VBox()
                << (HBox("{align: stretch end;}")
                    << (VBox()
                        << On(~login == true) / Enabled(true)
                        << On(~login == false) / Enabled(false)
                        << (Button()<< depositMoney) << Button() << Button()
                        << (Button() << Guard(~depos == true)/cancelBtn) //TODO::
                        )
                    << screen
                    << (VBox()
                        << On(~login == true) / Enabled(true)
                        << On(~login == false) / Enabled(false)
                        << (Button("{#withdraw}") << withdrawMoney) << Button("") << Button()
                        << (Button("") << On(~depos == true || ~withd == true)/confirmBtn)) //TODO::
                        )
                << (VBox()
                    << Label(".label Cash")
                    << cashBox
                    )
                )
            << (VBox()
                << (Button(".label RECEIPT")
                    << en
                    << On.click / [=](GMouseEvent* e){ *mesg = "Print RECEIPT"; }
                    )
                << (Button(".label CARD")
                    << On.click
                    / [=]{ *hasCard = true; *mesg = "<color=green> Card Inserted"; }

                    // waits for 1 second whitout blocking the event loop
                    / SleepFor(2000)

                    // done once the delay is over
                    / [=]{ *mesg = "<color=blue> Enter Password"; }
                        << Box()
                   )
                )
            )
        << (HBox("#keyboard {align: stretch stretch;}")
            << (Box("#noboard {flow: grid 3}")
                << [=](auto e){ if(e->child()->textValue() != "*" && e->child()->textValue() != "\\#" )*psw += e->child()->textValue();}
//                << Guard(~auth == true)   / Enabled(true)
//                << Guard(~idle == true)   / Enabled(false)
                << On(~hasCard == true) / Enabled(true)
                << On(~hasCard == false) / Enabled(false)
                << Button("1") << Button("2") << Button("3")
                << Button("4") << Button("5") << Button("6")
                << Button("7") << Button("8") << Button("9")
                << Button("*") << Button("0") << (Button("\\#") << []{})
                )
            << (Box("#funboard {flow: grid 1}")
                << On(~hasCard == true) / Enabled(true)
                << On(~hasCard == false) / Enabled(false)
                << (Button("CANCEL") << Background("yellow") << [=](auto e){ psw->erase(psw->stringValue().size() - 1, 1); })
                << (Button("CLEAR") << bg << [=](auto e){ *psw = ""; })
                << (Button("ENTER") << Background("green")
                    << [=](auto e){
```

```
                            if(psw->stringValue() == userPsw->stringValue()) {
                                *mesg = "<color=green> Login Successfully";
//                                *mesg = "<color=blue> Select Operation";
                                screen->removeChildren();
                                screen << mesg
                                    << "select operation"
                                    << (HBox("{flow: row; align: stretch end;}")
                                        << (sideTextL << "Deposit" <<"" <<"" <<"")
                                        << Box()
                                        << (sideTextR << "Withdraw" <<"" <<"" <<"" )
                                        );
                                *login = true;
                            }
                            else { ++ wrongCount;
                                *mesg = "<color=red> Wrong Password";
                                *err = "Left " + to_string(4 - *wrongCount) + " Attempts";
                            }
                            *psw = "";
                        }
                    )
                << Button("")
                )
            )

    );
    makeInternal();   // disable inspection
}



int main(int argc, const char** argv)
{
  GApp app(argc, argv);
  app.canInspect();

  return app.start(new ATM);
}
```