Lightning Talk on July 10th, 2024

Would you like a GUI with that?

Andy Joseph

Founder of ProntoGUI

andy@prontogui.com

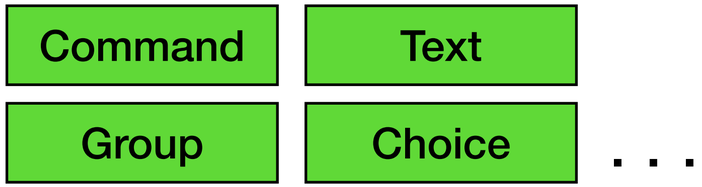
Gopher Slack @Andy Joseph

GitHub/ProntoGUI

ProntoGUI - A FAST way to build a GUI in Go!

What is ProntoGUI?

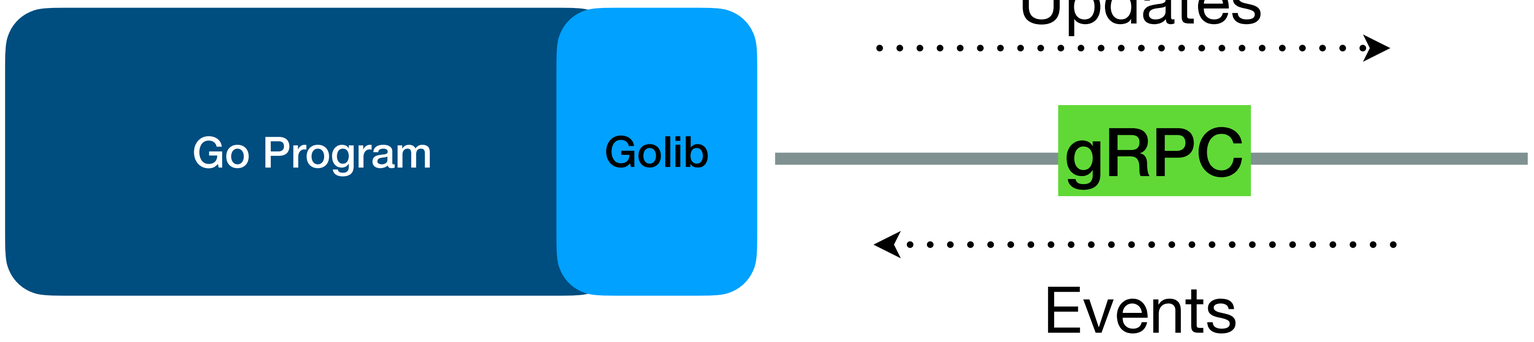
- Go module (Golib)
  - primitives to construct a user interface



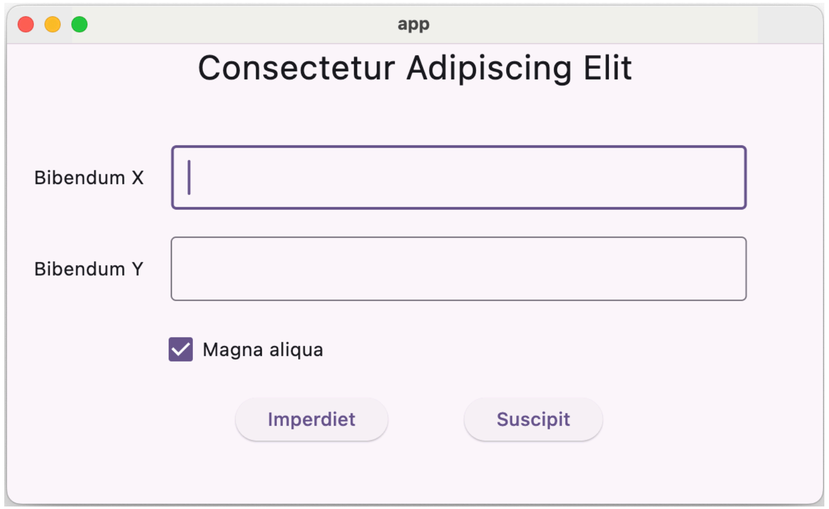
- App written in Flutter that does rendering
  - gesture detection / Material / cross-platform

- Embodiments of primitives look/behave how you want

How it works



ProntoGUI App



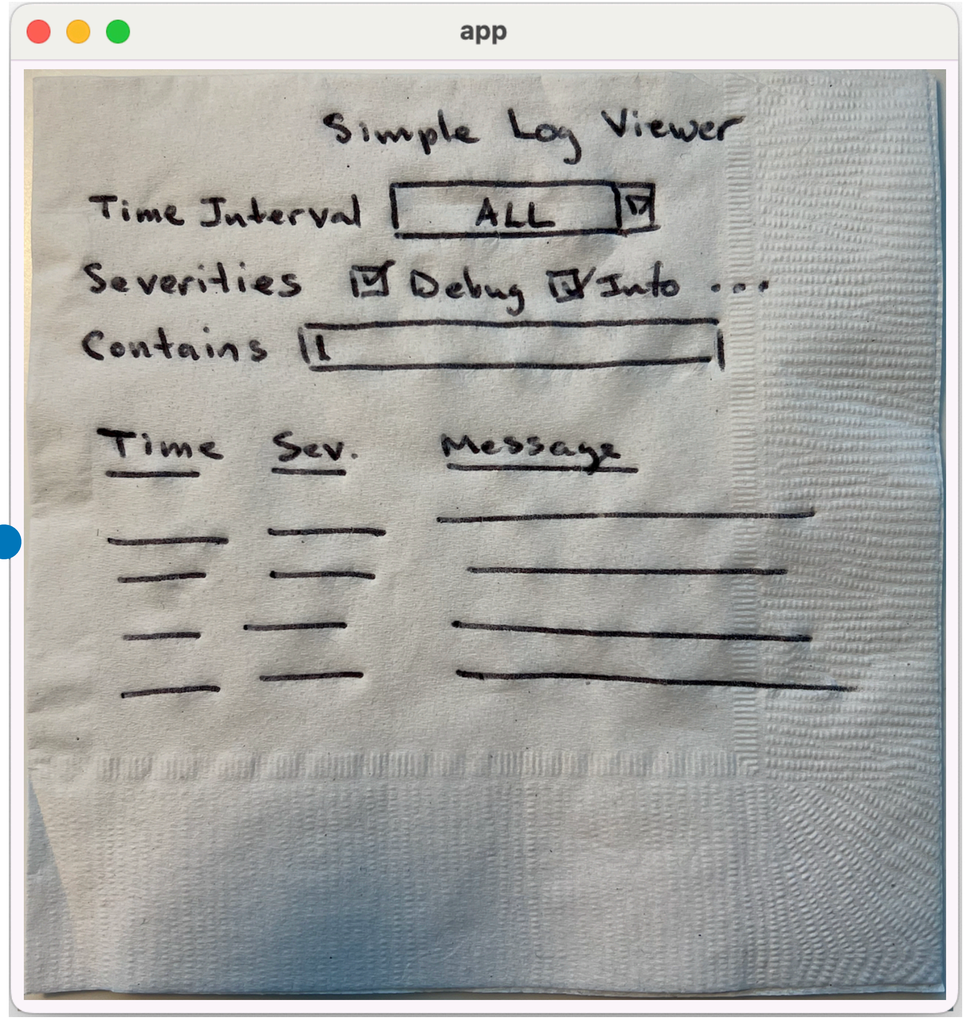
Build GUI using primitives
Wait for events…
(App connects) …(Full update)
React to events and update GUI

MacOS, Windows, or Linux

Demo

Create a simple log viewer

Service / Program
Written in Go



Import ProntoGUI

```go
import (
    "fmt"
    "time"

    // Import the Go library for ProntoGUI
    pg "github.com/prontogui/golib"
)

func main() {

    // Initialize ProntoGUI
    pgui := pg.NewProntoGUI()
    err := pgui.StartServing("127.0.0.1", 50053)

    if err != nil {
        fmt.Printf("Error trying to start server:  %s", err.Error())
        return
    }

    // <<== BEGIN - Building the GUI

    // Big and bold heading for the GUI
    guiHeading := pg.TextWith{
        Content:    "Simple Log Viewer",
        Embodiment: "{\"fontFamily\":\"Roboto\", \"fontSize\":\"20.0\"}",
    }.Make()

    // Time interval choice
    timeHeading := pg.TextWith{Content: "Time Interval "}.Make()
    timeChoice := pg.ChoiceWith{
```

Initialize things

```go
import (
    "fmt"
    "time"

    // Import the Go library for ProntoGUI
    pg "github.com/prontogui/golib"
)

func main() {

    // Initialize ProntoGUI
    pgui := pg.NewProntoGUI()
    err := pgui.StartServing("127.0.0.1", 50053)

    if err != nil {
        fmt.Printf("Error trying to start server:  %s", err.Error())
        return
    }

    // <<== BEGIN - Building the GUI

    // Big and bold heading for the GUI
    guiHeading := pg.TextWith{
        Content:    "Simple Log Viewer",
        Embodiment: "{\"fontFamily\":\"Roboto\", \"fontSize\":\"20.0\"}",
    }.Make()

    // Time interval choice
    timeHeading := pg.TextWith{Content: "Time Interval "}.Make()
    timeChoice := pg.ChoiceWith{
```

Build the GUI...

```
    // Import the Go library for ProntoGUI
    pg "github.com/prontogui/golib"
)

func main() {

    // Initialize ProntoGUI
    pgui := pg.NewProntoGUI()
    err := pgui.StartServing("127.0.0.1", 50053)

    if err != nil {
        fmt.Printf("Error trying to start server:  %s", err.Error())
        return
    }

    // <<== BEGIN - Building the GUI

    // Big and bold heading for the GUI
    guiHeading := pg.TextWith{
        Content:    "Simple Log Viewer",
        Embodiment: "{\"fontFamily\":\"Roboto\", \"fontSize\":\"20.0\"}",
    }.Make()

    // Time interval choice
    timeHeading := pg.TextWith{Content: "Time Interval "}.Make()
    timeChoice := pg.ChoiceWith{
        Choice:  AllTime,
        Choices: []string{RecentMinute, RecentHour, RecentDay, AllTime},
    }.Make()
    timeGroup := pg.GroupWith{
        GroupItems: []pg.Primitive{timeHeading, timeChoice},
    }.Make()

    // Severities check boxes
```

Build the GUI...

```
// <<== BEGIN - Building the GUI

// Big and bold heading for the GUI
guiHeading := pg.TextWith{
    Content:    "Simple Log Viewer",
    Embodiment: "{\"fontFamily\":\"Roboto\", \"fontSize\":\"20.0\"}",
}.Make()

// Time interval choice
timeHeading := pg.TextWith{Content: "Time Interval "}.Make()
timeChoice := pg.ChoiceWith{
    Choice:  AllTime,
    Choices: []string{RecentMinute, RecentHour, RecentDay, AllTime},
}.Make()
timeGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{timeHeading, timeChoice},
}.Make()

// Severities check boxes
severitiesHeading := pg.TextWith{Content: "Severities "}.Make()
debugCheck := pg.CheckWith{Label: "Debug", Checked: true}.Make()
infoCheck := pg.CheckWith{Label: "Info", Checked: true}.Make()
warningCheck := pg.CheckWith{Label: "Warning", Checked: true}.Make()
errorCheck := pg.CheckWith{Label: "Error", Checked: true}.Make()
panicCheck := pg.CheckWith{Label: "Panic", Checked: true}.Make()
severitiesGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{
        severitiesHeading, debugCheck, infoCheck, warningCheck, errorCheck, panicCheck,
    },
}.Make()

// Message text filter
messageHeading := pg.TextWith{Content: "Message Contains "}.Make()
messageTextField := pg.TextFieldWith{}.Make()
```

Build the GUI...

```
timeChoice := pg.ChoiceWith{
    Choice:  AllTime,
    Choices: []string{RecentMinute, RecentHour, RecentDay, AllTime},
}.Make()
timeGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{timeHeading, timeChoice},
}.Make()

// Severities check boxes
severitiesHeading := pg.TextWith{Content: "Severities "}.Make()
debugCheck := pg.CheckWith{Label: "Debug", Checked: true}.Make()
infoCheck := pg.CheckWith{Label: "Info", Checked: true}.Make()
warningCheck := pg.CheckWith{Label: "Warning", Checked: true}.Make()
errorCheck := pg.CheckWith{Label: "Error", Checked: true}.Make()
panicCheck := pg.CheckWith{Label: "Panic", Checked: true}.Make()
severitiesGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{
        severitiesHeading, debugCheck, infoCheck, warningCheck, errorCheck, panicCheck,
    },
}.Make()

// Message text filter
messageHeading := pg.TextWith{Content: "Message Contains "}.Make()
messageTextField := pg.TextFieldWith{}.Make()
messageGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{messageHeading, messageTextField},
}.Make()

// A table to show log items
table := pg.TableWith{
    TemplateRow: []pg.Primitive{
        &pg.Text{}, &pg.Text{}, &pg.Text{},
    },
    Headings: []string{
```

Build the GUI...

```
warningCheck := pg.CheckWith{Label: "Warning", Checked: true}.Make()
errorCheck := pg.CheckWith{Label: "Error", Checked: true}.Make()
panicCheck := pg.CheckWith{Label: "Panic", Checked: true}.Make()
severitiesGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{
        severitiesHeading, debugCheck, infoCheck, warningCheck, errorCheck, panicCheck,
    },
}.Make()

// Message text filter
messageHeading := pg.TextWith{Content: "Message Contains "}.Make()
messageTextField := pg.TextFieldWith{}.Make()
messageGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{messageHeading, messageTextField},
}.Make()

// A table to show log items
table := pg.TableWith{
    TemplateRow: []pg.Primitive{
        &pg.Text{}, &pg.Text{}, &pg.Text{},
    },
    Headings: []string{
        "Time", "Severity", "Message",
    },
}.Make()

pgui.SetGUI(guiHeading, timeGroup, severitiesGroup, messageGroup, table)

// <<== END - Building the GUI

// Loop while handling the events occuring in the GUI
for {
    // Query for the log items as of this moment
    logItems := RetrieveFilteredLogItems(
        timeChoice.Choice(),
```

Build the GUI...

```
warningCheck := pg.CheckWith{Label: "Warning", Checked: true}.Make()
errorCheck := pg.CheckWith{Label: "Error", Checked: true}.Make()
panicCheck := pg.CheckWith{Label: "Panic", Checked: true}.Make()
severitiesGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{
        severitiesHeading, debugCheck, infoCheck, warningCheck, errorCheck, panicCheck,
    },
}.Make()

// Message text filter
messageHeading := pg.TextWith{Content: "Message Contains "}.Make()
messageTextField := pg.TextFieldWith{}.Make()
messageGroup := pg.GroupWith{
    GroupItems: []pg.Primitive{messageHeading, messageTextField},
}.Make()

// A table to show log items
table := pg.TableWith{
    TemplateRow: []pg.Primitive{
        &pg.Text{}, &pg.Text{}, &pg.Text{},
    },
    Headings: []string{
        "Time", "Severity", "Message",
    },
}.Make()

pgui.SetGUI(guiHeading, timeGroup, severitiesGroup, messageGroup, table)

// <<== END - Building the GUI

// Loop while handling the events occuring in the GUI
for {
    // Query for the log items as of this moment
    logItems := RetrieveFilteredLogItems(
        timeChoice.Choice(),
```

Build the GUI...

```
table := pg.TableWith{
    TemplateRow: []pg.Primitive{
        &pg.Text{}, &pg.Text{}, &pg.Text{},
    },
    Headings: []string{
        "Time", "Severity", "Message",
    },
}.Make()

pgui.SetGUI(guiHeading, timeGroup, severitiesGroup, messageGroup, table)

// <<== END - Building the GUI

// Loop while handling the events occuring in the GUI
for {
    // Query for the log items as of this moment
    logItems := RetrieveFilteredLogItems(
        timeChoice.Choice(),
        debugCheck.Checked(),
        infoCheck.Checked(),
        warningCheck.Checked(),
        errorCheck.Checked(),
        panicCheck.Checked(),
        messageTextField.TextEntry(),
    )

    // Convert messages into a table row of ProntoGUI primitives
    rows := [][]pg.Primitive{}

    for _, item := range logItems {
        rows = append(rows, []pg.Primitive{
            timePrimitive(item.Time),
            severityPrimitive(item.Severity),
            messagePrimitive(item.Message),
        })
```

Retrieve Log Items

```
// <<== END - Building the GUI

// Loop while handling the events occuring in the GUI
for {
    // Query for the log items as of this moment
    logItems := RetrieveFilteredLogItems(
        timeChoice.Choice(),
        debugCheck.Checked(),
        infoCheck.Checked(),
        warningCheck.Checked(),
        errorCheck.Checked(),
        panicCheck.Checked(),
        messageTextField.TextEntry(),
    )

    // Convert messages into a table row of ProntoGUI primitives
    rows := [][]pg.Primitive{}

    for _, item := range logItems {
        rows = append(rows, []pg.Primitive{
            timePrimitive(item.Time),
            severityPrimitive(item.Severity),
            messagePrimitive(item.Message),
        })
    }

    // Update the table contents
    table.SetRows(rows)

    // Wait for something to happen in the GUI
    _, err := pgui.Wait()
    if err != nil {
        fmt.Printf("error from Wait() is:  %s\n", err.Error())
        break
```

Convert to primitives

```
// <<== END - Building the GUI

// Loop while handling the events occuring in the GUI
for {
    // Query for the log items as of this moment
    logItems := RetrieveFilteredLogItems(
        timeChoice.Choice(),
        debugCheck.Checked(),
        infoCheck.Checked(),
        warningCheck.Checked(),
        errorCheck.Checked(),
        panicCheck.Checked(),
        messageTextField.TextEntry(),
    )

    // Convert messages into a table row of ProntoGUI primitives
    rows := [][]pg.Primitive{}

    for _, item := range logItems {
        rows = append(rows, []pg.Primitive{
            timePrimitive(item.Time),
            severityPrimitive(item.Severity),
            messagePrimitive(item.Message),
        })
    }

    // Update the table contents
    table.SetRows(rows)

    // Wait for something to happen in the GUI
    _, err := pgui.Wait()
    if err != nil {
        fmt.Printf("error from Wait() is:  %s\n", err.Error())
        break
```

Update the table

```
// <<== END - Building the GUI

// Loop while handling the events occuring in the GUI
for {
    // Query for the log items as of this moment
    logItems := RetrieveFilteredLogItems(
        timeChoice.Choice(),
        debugCheck.Checked(),
        infoCheck.Checked(),
        warningCheck.Checked(),
        errorCheck.Checked(),
        panicCheck.Checked(),
        messageTextField.TextEntry(),
    )

    // Convert messages into a table row of ProntoGUI primitives
    rows := [][]pg.Primitive{}

    for _, item := range logItems {
        rows = append(rows, []pg.Primitive{
            timePrimitive(item.Time),
            severityPrimitive(item.Severity),
            messagePrimitive(item.Message),
        })
    }

    // Update the table contents
    table.SetRows(rows)

    // Wait for something to happen in the GUI
    _, err := pgui.Wait()
    if err != nil {
        fmt.Printf("error from Wait() is:  %s\n", err.Error())
        break
```

Wait for user event

```
                severityPrimitive(item.Severity),
                messagePrimitive(item.Message),
            })
        }

        // Update the table contents
        table.SetRows(rows)

        // Wait for something to happen in the GUI
        _, err := pgui.Wait()
        if err != nil {
            fmt.Printf("error from Wait() is:  %s\n", err.Error())
            break
        }
    }
}

// Convert a time.Time into a pg.Text primitive for the GUI.
func timePrimitive(t time.Time) pg.Primitive {
    timeString := t.Format(time.RFC1123)
    return pg.TextWith{Content: timeString}.Make()
}

// Convert a severity level into a pg.Text primitive for the GUI.
func severityPrimitive(severity int) pg.Primitive {

    var severityString string
    embodiment := ""

    switch severity {
    case LevelDebug:
        severityString = "DEBUG"
        // White text with brown background
        embodiment = "{\"color\":\"argb:0xFFFAFAFA\", \"backgroundColor\":\"argb:0xFF4E342E\"}"
    case LevelInfo:
```

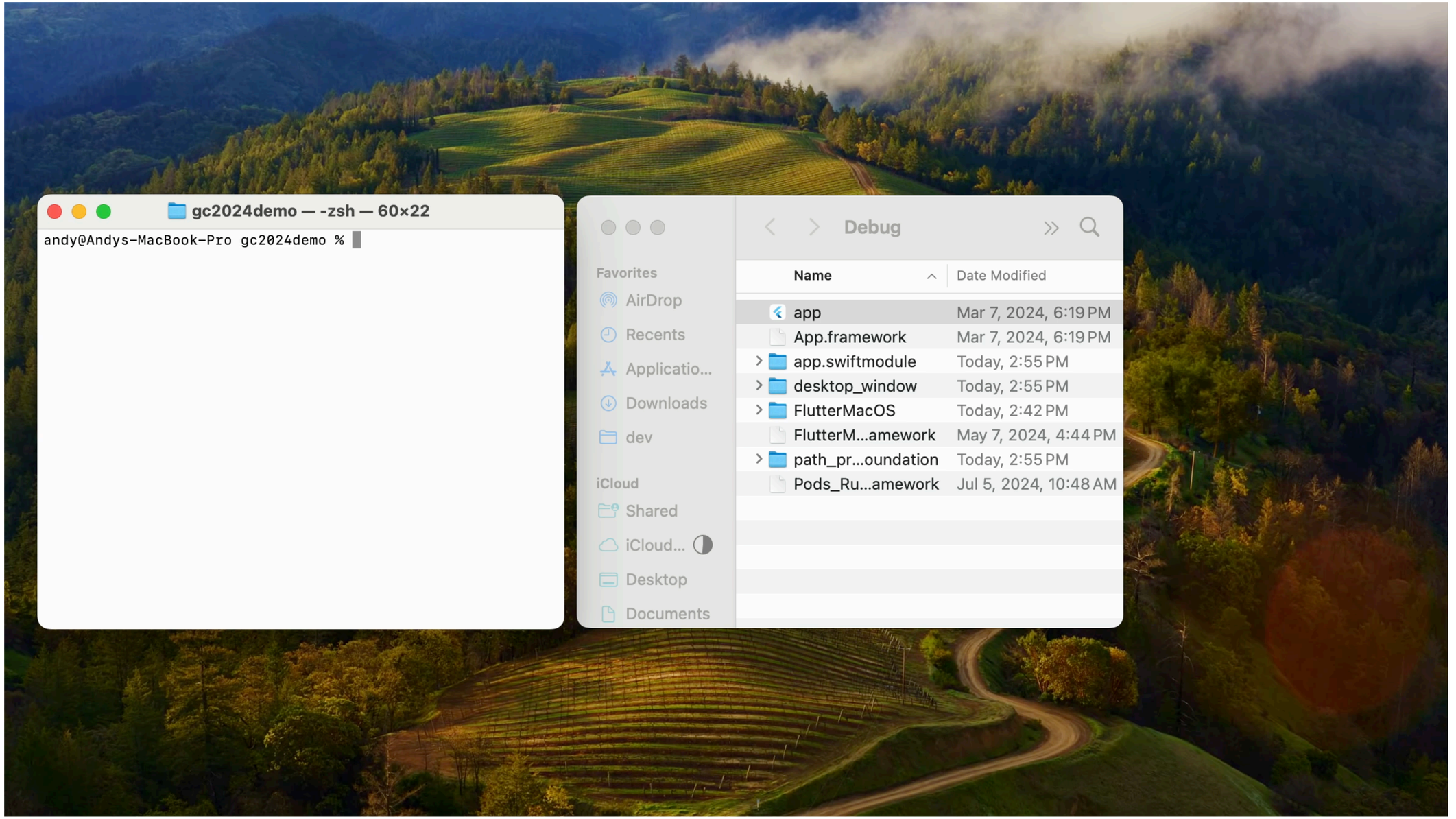
A few more functions

```
    }
}

// Convert a time.Time into a pg.Text primitive for the GUI.
func timePrimitive(t time.Time) pg.Primitive {
    timeString := t.Format(time.RFC1123)
    return pg.TextWith{Content: timeString}.Make()
}

// Convert a severity level into a pg.Text primitive for the GUI.
func severityPrimitive(severity int) pg.Primitive {

    var severityString string
    embodiment := ""

    switch severity {
    case LevelDebug:
        severityString = "DEBUG"
        // White text with brown background
        embodiment = "{\"color\":\"argb:0xFFFAFAFA\", \"backgroundColor\":\"argb:0xFF4E342E\"}"
    case LevelInfo:
        severityString = "INFO"
        embodiment = "{\"color\":\"argb:0xFF1A237E\"}"
    case LevelWarning:
        severityString = "WARNING"
        // Black text on yellow background
        embodiment = "{\"color\":\"argb:0xFF212121\", \"backgroundColor\":\"argb:0xFFFFEE58\"}"
    case LevelError:
        severityString = "ERROR"
        // Red text
        embodiment = "{\"color\":\"argb:0xFFF44336\"}"
    case LevelPanic:
        severityString = "PANIC"
        // White text with red background
```

A few more functions

```
// Convert a severity level into a pg.Text primitive for the GUI.
func severityPrimitive(severity int) pg.Primitive {

    var severityString string
    embodiment := ""

    switch severity {
    case LevelDebug:
        severityString = "DEBUG"
        // White text with brown background
        embodiment = "{\"color\":\"argb:0xFFFAFAFA\", \"backgroundColor\":\"argb:0xFF4E342E\"}"
    case LevelInfo:
        severityString = "INFO"
        embodiment = "{\"color\":\"argb:0xFF1A237E\"}"
    case LevelWarning:
        severityString = "WARNING"
        // Black text on yellow background
        embodiment = "{\"color\":\"argb:0xFF212121\", \"backgroundColor\":\"argb:0xFFFFEE58\"}"
    case LevelError:
        severityString = "ERROR"
        // Red text
        embodiment = "{\"color\":\"argb:0xFFF44336\"}"
    case LevelPanic:
        severityString = "PANIC"
        // White text with red background
        embodiment = "{\"color\":\"argb:0xFFFAFAFA\", \"backgroundColor\":\"argb:0xFFF44336\"}"
    default:
        severityString = "UNKNOWN"
    }

    return pg.TextWith{Content: severityString, Embodiment: embodiment}.Make()
}

// Convert a message into a pg.Text primitive for the GUI.
```

Software Demo



Lightning Talk on July 10th, 2024

Open source at GitHub/ProntoGUI

Gopher Slack @Andy Joseph

andy@prontogui.com

I would love to hear from you!