Jacob Alzén

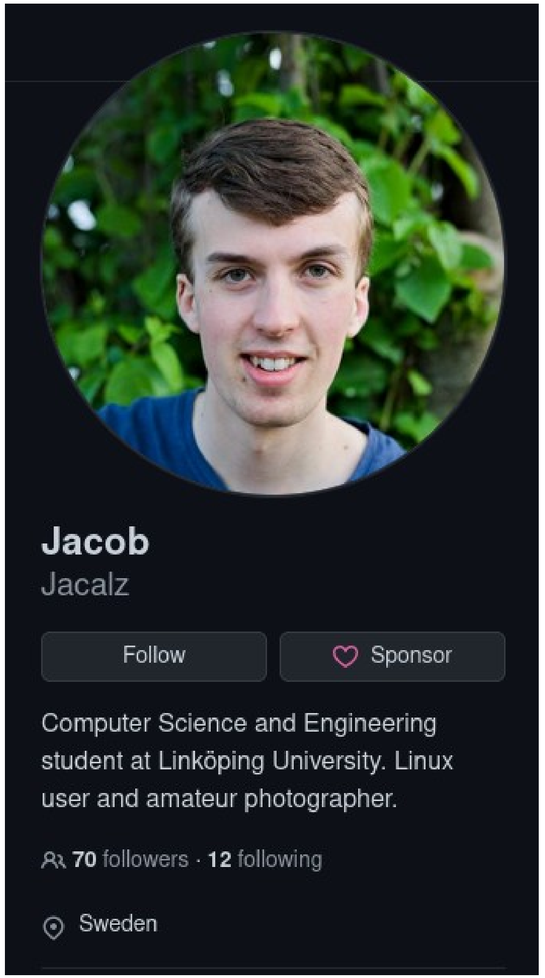
fyne conf 2022

The interfaces behind custom widgets

An overview of tuning widget behaviour

About me

- Second year Computer Science student.
- Open source philanthropist, amateur photographer and runner.
- Gophers Slack: @jacalz
- GitHub: https://github.com/jacalz

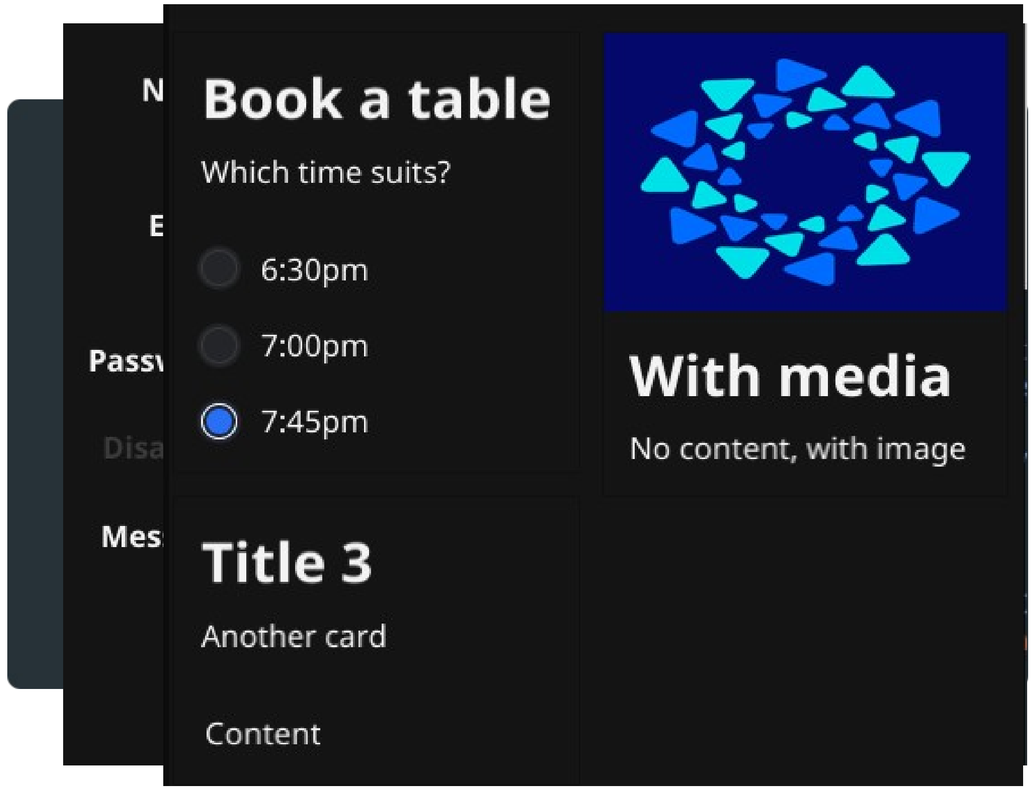


Objects on the screen

- CanvasObject: The most basic graphical object. Simple and performant.
- Use the provided canvas primitives.
- Containers are also CanvasObjects; see my talk about Layouts from FyneConf 2021 for more information.

More functionality with Widgets

- Widgets are CanvasObjects with a renderer attached.
- Gives access to more functionality.
- Allows implementing interfaces for adding and controlling behaviour.



Using BaseWidget and SimpleRenderer

- BaseWidget provides a helper
  that handles basic widget behaviours.
  - Remember to call .ExtendBaseWidget()
- SimpleRenderer is a helper
  for creating a WidgetRenderer for a
  single CanvasObject.

```go
type myWidget struct {
    widget.BaseWidget

    object fyne.CanvasObject
}


func (m *myWidget) CreateRenderer() fyne.WidgetRenderer {
    if m.object == nil {
        return nil
    }

    if w, ok := m.object.(fyne.Widget); ok {
        return w.CreateRenderer()
    }

    return widget.NewSimpleRenderer(m.object)
}

func newMyWidget(object fyne.CanvasObject) fyne.Widget {
    m := &myWidget{Object: object}
    m.ExtendBaseWidget(m)
    return m
}
```

Implementing an interface

fyne conf 2022

- Implement the method or methods from the given interface.
- Optionally, add a check to make sure, at compile-time, that the interface is satisfied.

```
type DoubleTappable interface {
    DoubleTapped(*PointEvent)
}
```

```
type myWidget struct {
    widget.BaseWidget

    object fyne.CanvasObject
}


// Emit compile error if interface is not implemented.
var _ fyne.DoubleTappable = (*myWidget)(nil)

func (m *myWidget) DoubleTapped(tap *fyne.PointEvent) {
  fmt.Println("I was tapped at:", tap.Position)
}
```

Receive events when tapped

- Import the fyne.io/fyne/v2 package.
- fyne.Tappable
- fyne.DoubleTappable
- fyne.SecondaryTappable

```
// fyne.Tappable
type Tappable interface {
    Tapped(*PointEvent)
}

// fyne.DoubleTappable
type DoubleTappable interface {
    DoubleTapped(*PointEvent)
}

// fyne.SecondaryTappable
type SecondaryTappable interface {
    TappedSecondary(*PointEvent)
}
```

Moving the view

- Import the fyne.io/fyne/v2 package.
- fyne.Scrollable
- fyne.Draggable

```go
// fyne.Scrollable
type Scrollable interface {
    Scrolled(*ScrollEvent)
}

// fyne.Draggable
type Draggable interface {
    Dragged(*DragEvent)
    DragEnd()
}
```

Focus related interfaces

- Import the fyne.io/fyne/v2 package.
- fyne.Disableable
- fyne.Focusable
- fyne.Tabbable

```
// fyne.Disableable
type Disableable interface {
    Enable()
    Disable()
    Disabled() bool
}

// fyne.Focusable
type Focusable interface {
    FocusGained()
    FocusLost()
    TypedRune(rune)
    TypedKey(*KeyEvent)
}

// fyne.Tabbable
type Tabbable interface {
    AcceptsTab() bool
}
```

Shortcuts and validation

- Import the fyne.io/fyne/v2 package.
- fyne.Shortcutable
- fyne.Validatable

```
// fyne.Shortcutable
type Shortcutable interface {
    TypedShortcut(Shortcut)
}

// fyne.Validatable
type Validatable interface {
    Validate() error
    SetOnValidationChanged(func(error))
}
```

Desktop specific interfaces

- Import the fyne.io/fyne/v2/driver/desktop package.
- desktop.Cursorable
- desktop.Hoverable
- desktop.Keyable
- desktop.Mouseable

```go
// desktop.Cursorable
type Cursorable interface {
    Cursor() Cursor
}

// desktop.Hoverable
type Hoverable interface {
    MouseIn(*MouseEvent)
    MouseMoved(*MouseEvent)
    MouseOut()
}

// desktop.Keyable
type Keyable interface {
    fyne.Focusable

    KeyDown(*fyne.KeyEvent)
    KeyUp(*fyne.KeyEvent)
}

// desktop.Mouseable
type Mouseable interface {
    MouseDown(*MouseEvent)
    MouseUp(*MouseEvent)
}
```

Mobile specific interfaces

- Import the fyne.io/fyne/v2/driver/mobile package.
- mobile.Keyboardable
- mobile.Touchable

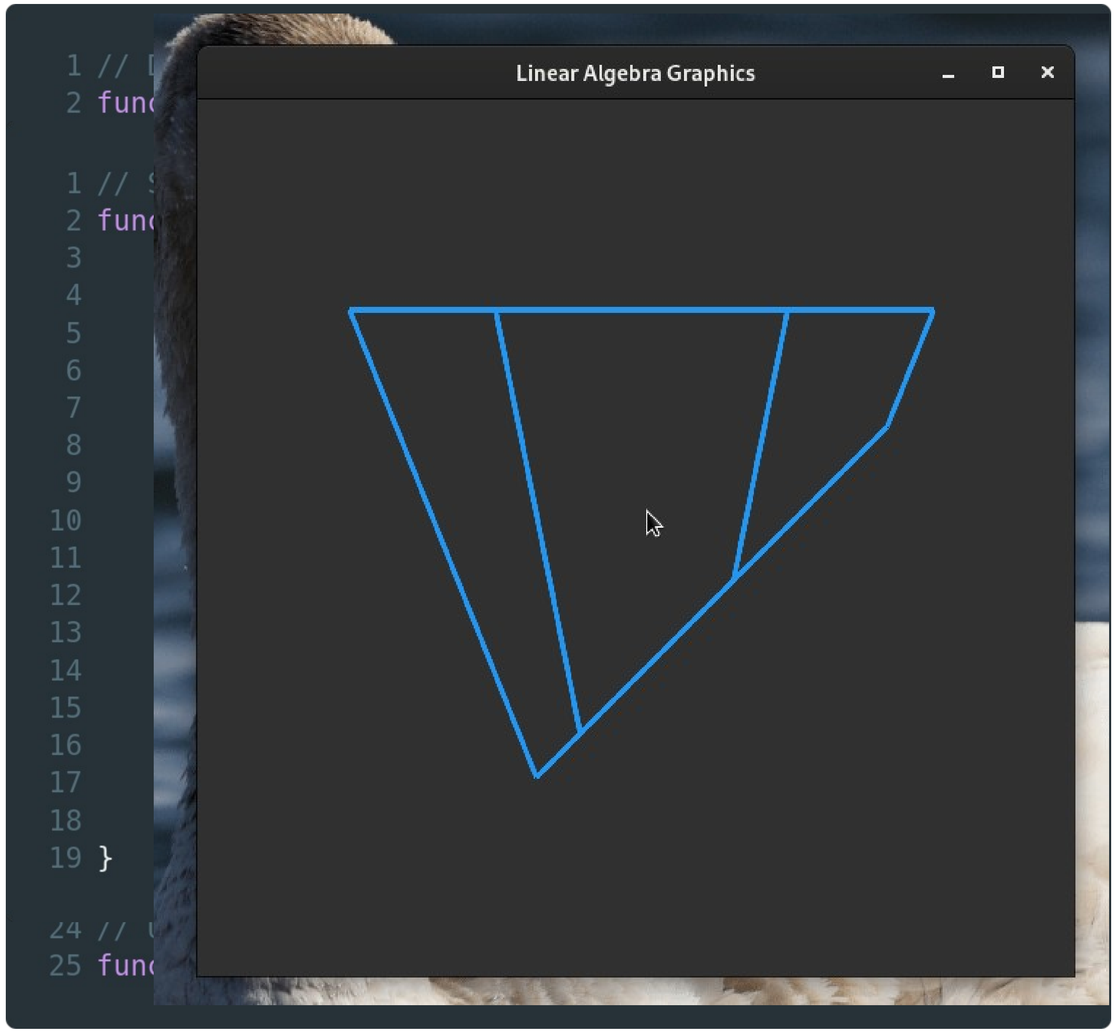
```go
// mobile.Keyboardable
type Keyboardable interface {
    fyne.Focusable

    Keyboard() KeyboardType
}

// mobile.Touchable
type Touchable interface {
    TouchDown(*TouchEvent)
    TouchUp(*TouchEvent)
    TouchCancel(*TouchEvent)
}
```

Implementing dragging and scrolling

- 3D wireframes using Linear Algebra.
- Rotate the view by dragging the mouse.
- Zoom the view using scrolling.
- The code is avaliable at: https://github.com/Jacalz/linedisp



Thanks for listening

- API documentation:
  - Standard interfaces: https://pkg.go.dev/fyne.io/fyne/v2
  - Desktop specific interfaces: https://pkg.go.dev/fyne.io/fyne/v2/driver/desktop
  - Mobile specific interfaces: https://pkg.go.dev/fyne.io/fyne/v2/driver/mobile