G-NAV

Soaring the clouds with **AdaWebPack**

The adventure of developing a new soaring application based on **Ada**, WASM and WebGL

Guillermo Augusto Hazebrouck

- Aeronautical engineer (UNC / 2010)
- Argentina > Belgium
- ANSP (Ada developer)

What is soaring?

Flying without engine! Really?

Circling on the upwards air streams and avoiding the downwards air streams.

Tactical sport!

Meteorology
Aircraft performance
Navigation skills
Flying skills
Confidence

Motivation

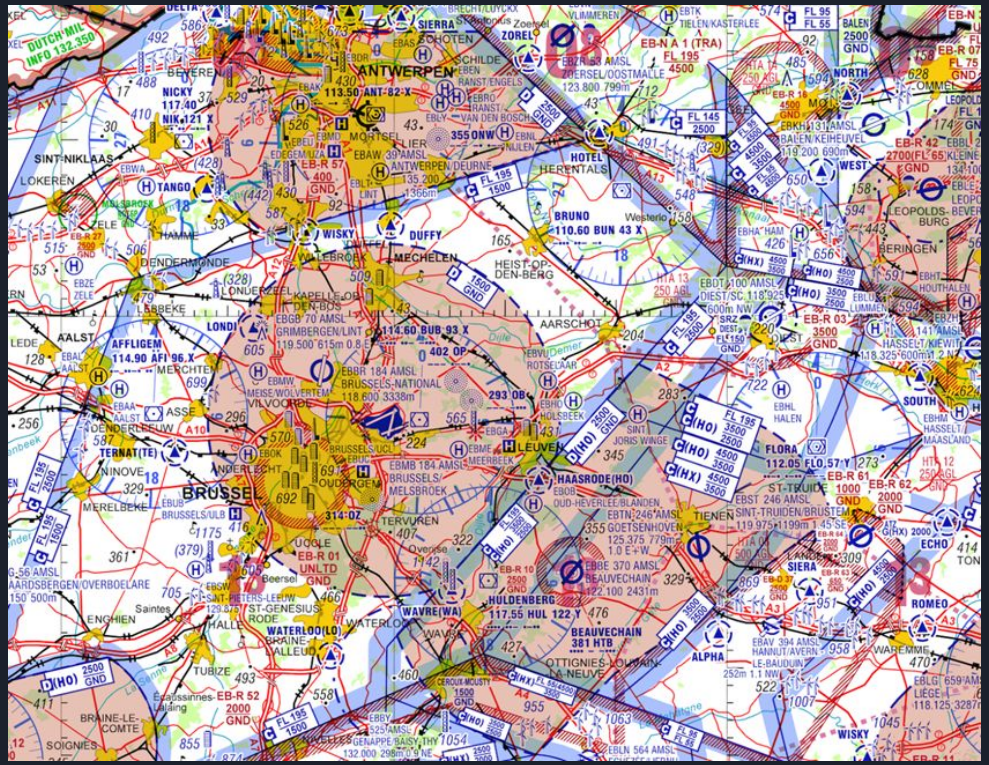
How far can I go?

The answer depends on:

- Altitude
- Topography
- Aircraft performance
- Direction to go
- Wind
- Sink

HARD TO ASSES FOR THE HUMAN BRAIN

Where am I allowed to fly?



Soaring computers

XCSoar	● Open source (2005) ● C++/C based ● Not in iOS ● Broad scope: multiple sensor integration ● Desktop-like environment ● Huge community (love + passion)
SeeYou	● Commercial ● It looks great ● Feature rich ● Android + iOS
LXNav + others	● Commercial ● Only available as dashboard equipment

G-NAV

- Focus on mobile (**Android + iOS**)
- Open source
- Restrict dependencies
- Restrict scope
- Focus on simplicity
- Robustness
- Ada

G-NAV

Project evolution

Hobby project for fun

2020 → LINUX (x86-64)

2021 → RPi-4 (arm64)

2023

- Ada native compiler (FSF)
- OpenGL ES
- GLWF / SDL2
- UDP / UART / Files

Community project?

2024 → PWA (WASM)

AdaWebPack

- Ada LLVM
- WASM
- WebGL
- Web API's

G-NAV

Elements of an EFIS

We want to have all this...

Storage

Data banks

- Topography (20MB)
- ATC sectors
- References
- Aircrafts

Processing

Numerical performance model for predictions

Integration

Sensors

- GPS
- Total pressure
- Static pressure
- Probe pressure

Connectivity

Link

- Traffic
- Meteorology
- Sharing info

Interactive graphical interface

G-NAV

System architecture

The system is now divided in two worlds:

JavaScript + HTML

- Loading WASM module
- HMI definition
- HMI events
- Timer
- Web API's

AS SMALL AS POSSIBLE!

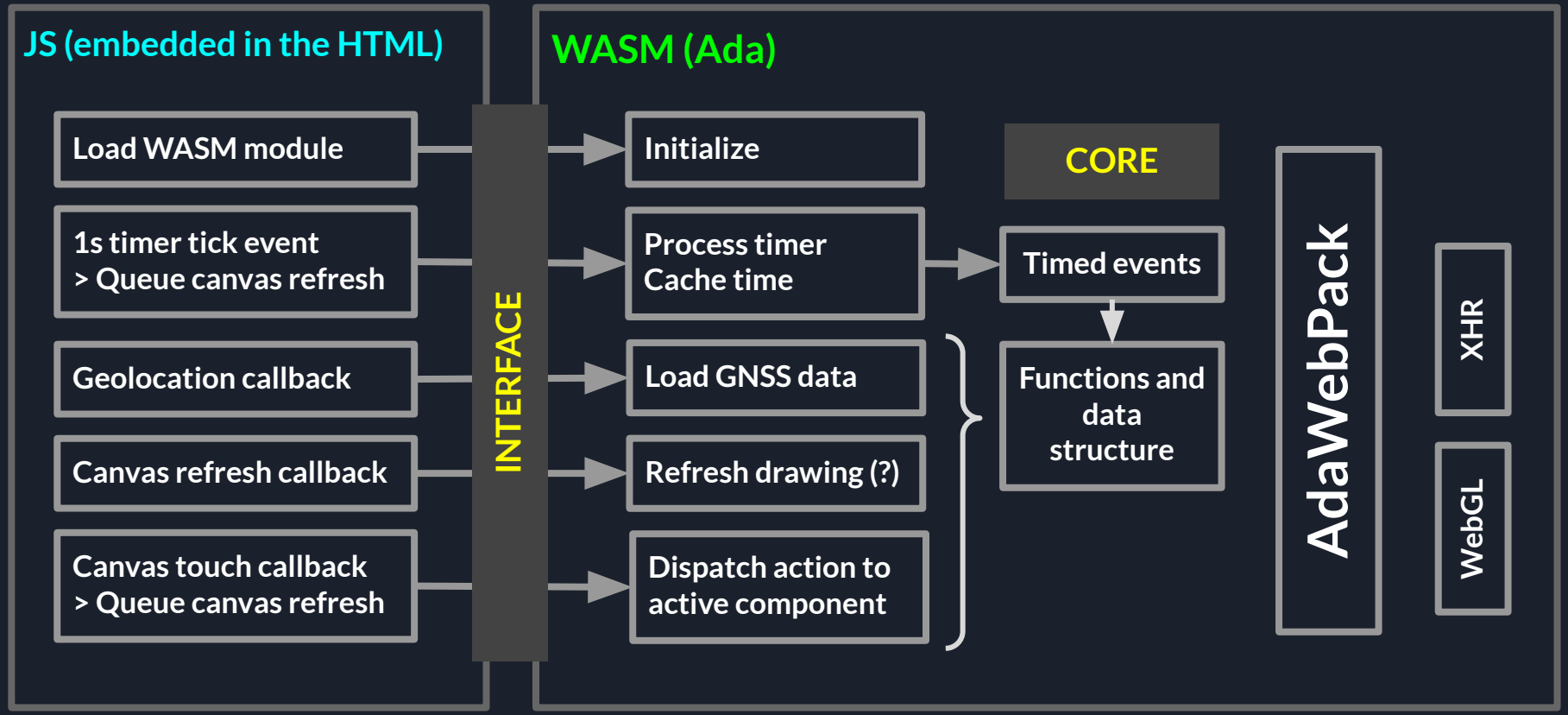
WASM module (Ada)

- Data structures
- Actual computations
- Rendering

ALMOST EVERYTHING IS DONE HERE!

G-NAV

Data flow architecture



G-NAV

Data flow architecture

To sum up:

- One timed event every 1 second + render
- All internal application events run synchronously with the timer
- Touch event + render
- Render only when necessary
- One GNSS input event (every update is injected ~ 1s)

ADA CODE RUNS IN A SINGLE THREAD

G-NAV

Data structure

✓ Only static buffers

🚫 Dynamic allocation

- No leaks
- No overflow

Server side pre-processing

- No load-time errors
- Transport efficiency
- Parsing efficiency

Encapsulated data packages with object oriented structures

Flight

- Timeline
- Wind
- Route
- Traffic
- Aircraft

Maps

- Terrain (10 million nodes)
- Layers
- References

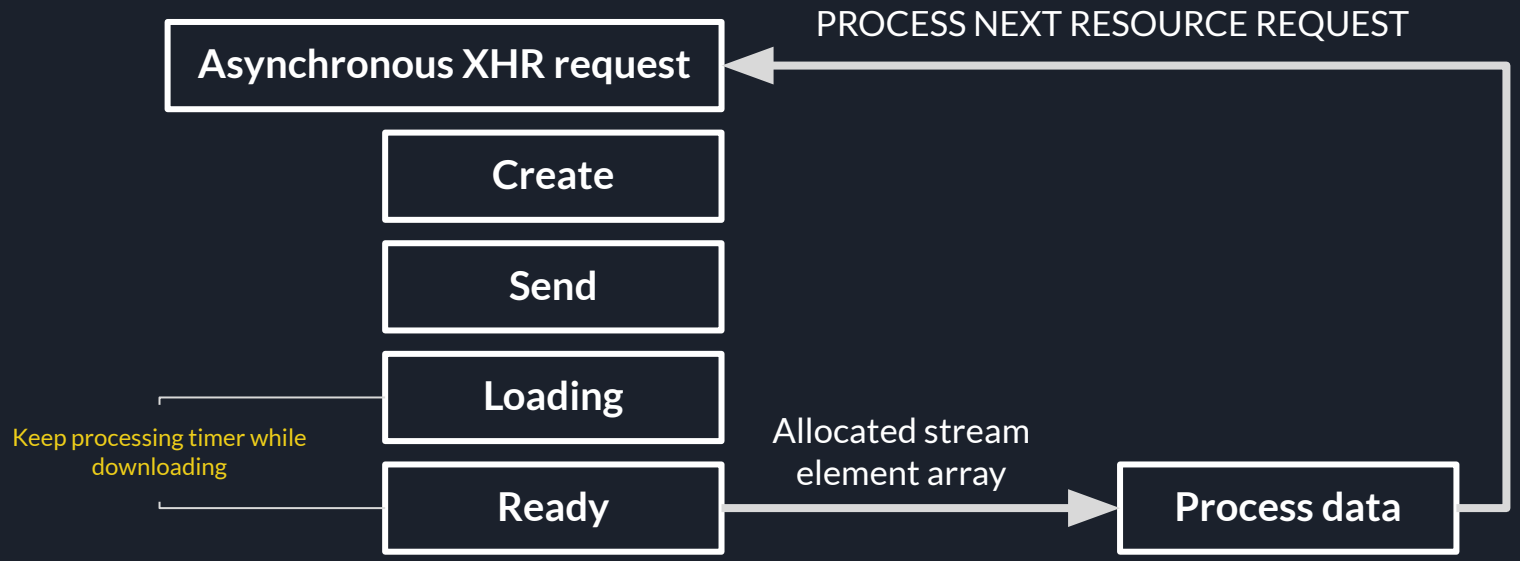
G-NAV

Static data transfer

File system → AdaWebPack XHR API (XML HTTP Request)

For the static data we work with a queue (FIFO) of asynchronous request

No requests in parallel to respect loading sequence

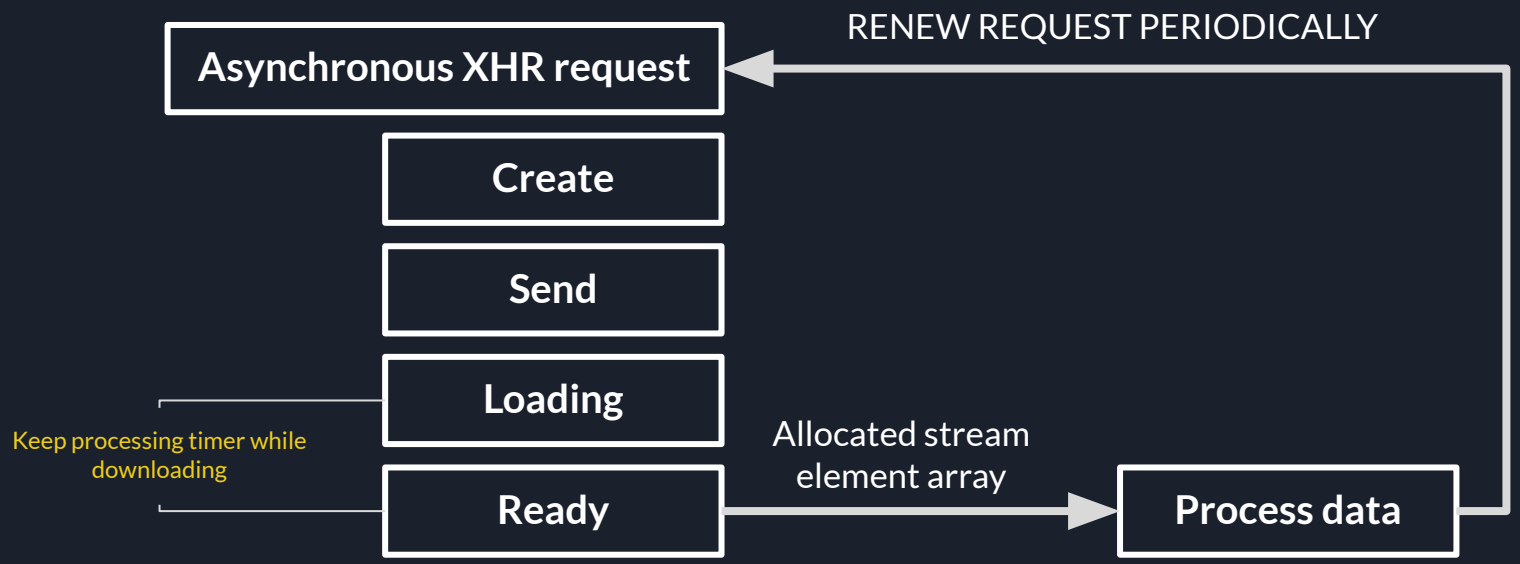


G-NAV

Dynamic data transfer

AdaWebPack XHR API

Dynamic data is requested by cyclic polling: METAR and TRAFFIC



G-NAV

Data transfer

BE AWARE OF THE NEXT ISSUES

BUG 1: AdaWebPack XHR restricts array buffers to 65 KB, attempting to read larger requests will collapse the module.

BUG 2: when BUG1 is solved, memory allocation for the buffer still seems to require multiples of 65 KB (the size of WASM pages).

In G-NAV, the original XHR package has been reshuffled!
Some changes must be proposed to AdaWebPack developers...

G-NAV

WHAT TO DO WHEN THERE IS NO CONNECTION?

XHR offline? → Service worker!

- The service worker intercepts the XHR requests and is able to provide local cached content.
- Static content will be cached locally by the browser after first arrival and it will be served locally afterwards. This is very efficient! Almost like reading from the file system.
- Dynamic content (traffic + metar) is still always directed to the web server.

It feels like a native application!

G-NAV

WHAT DO WE DO WITH THE LOCAL CONFIGURATION?

User configuration? ⟶ Local Storage web API

- Temporal configuration can be stored locally so that it is restored after reopening the app.
- Local storage is a list of key/value pairs.
- This has been included as a binding in the adawebpack.msj file.
  In Ada, the binding looks simply like this:

```
function  Get_Item (Key : String) return String;
procedure Set_Item (Key : String; Value : String);
```

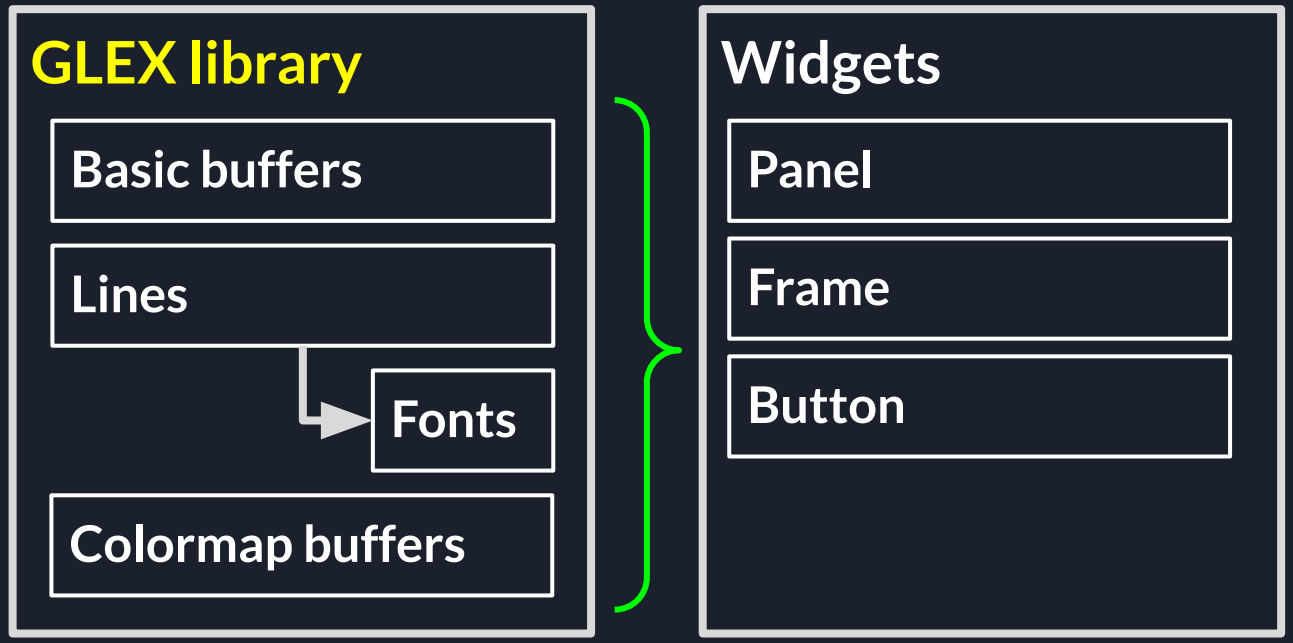
This could be included in AdaWebPack!

```
  __adawebpack__storage__setItem: function(key_address, key_size, value_address, value_size) {
      window.localStorage.setItem(string_to_js(key_address, key_size), string_to_js(value_address,
value_size));
  },
  __adawebpack__storage__getItem: function(key_address, key_size) {
      return string_to_wasm(window.localStorage.getItem(string_to_js(key_address, key_size)));
  }
```

G-NAV

Graphics

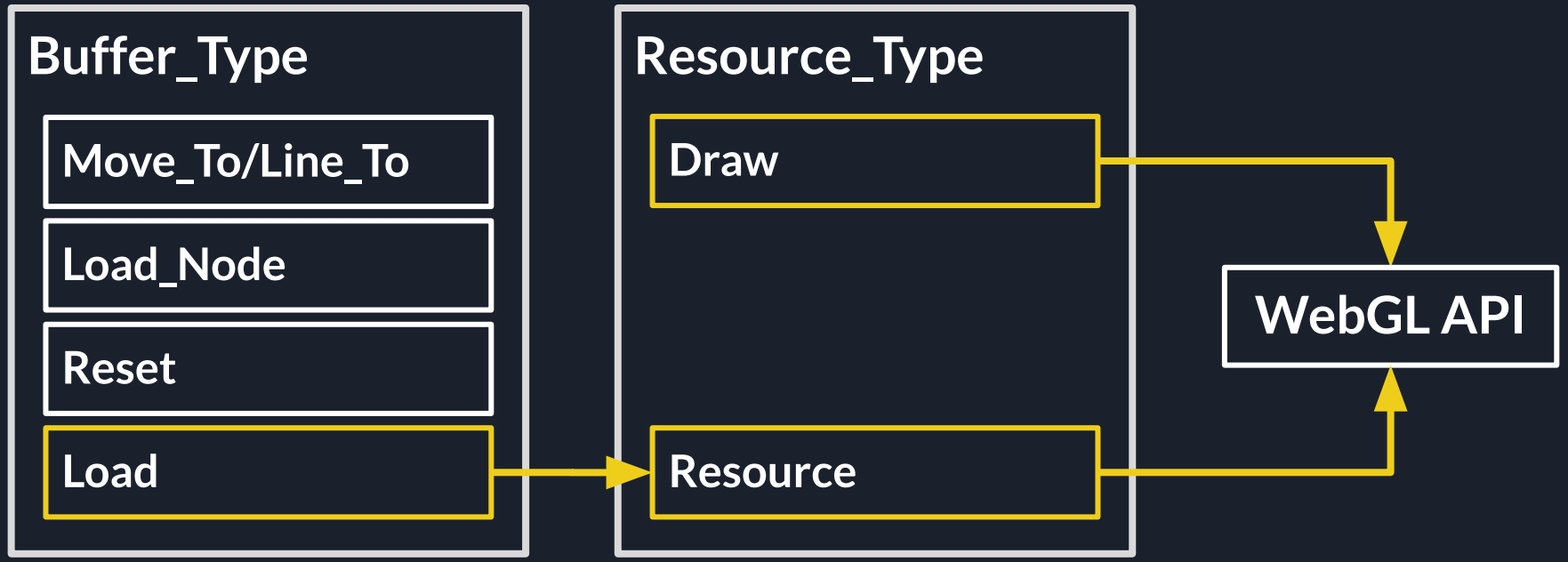
100% WebGL vector graphics



G-NAV

Graphics

GLEX: a library on top of WebGL that is easier to use



G-NAV

Graphics

Little example of GLEX library… note how GL complexity is hidden:

```
L1 : Resource_Type; -- Static resources (it hides a GL buffer ID)
L2 : Resource_Type;
...
declare
   B : Buffer_Type := New_Line_Buffer (Lines => 2); --> Reserved
begin
   B.Move_To (0.0, 0.0); --> Prepare
   B.Line_To (0.0, 1.0); --> Construct 1st line in triangles
   L1.Load (B);          --> Load into GPU
   B.Move_To (1.0, 1.0); --> Prepare
   B.Line_To (1.0, 0.0); --> Construct 2nd line in triangles
   L2.Load (B);          --> Load into GPU
end;
L1.Draw (...); --> Draw the triangles
L2.Draw (...); --> Draw the triangles
...
```

Find more details in the source code!

G-NAV

Graphics

Widgets: an object oriented API for a simple GUI

Fanciness

Readability
Consistency
Organization

Basic

- Widget
  - Frame
  - Button

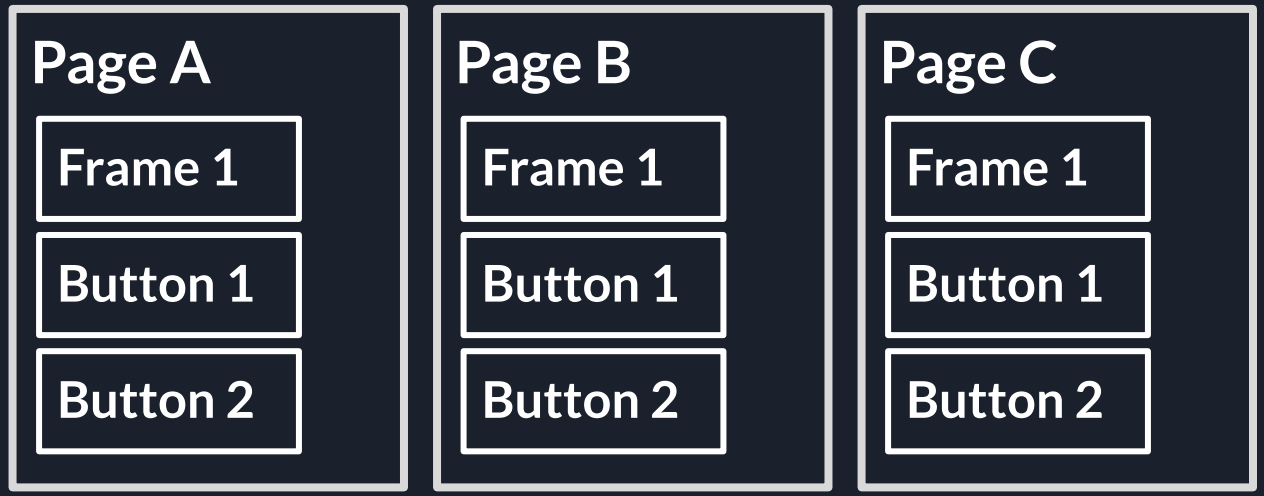
Composed

- Dialog
- Keyboard

G-NAV

Graphics

Widgets: not driven by events (not like GTK)
Pages pass the event directly to their child's



Thank you