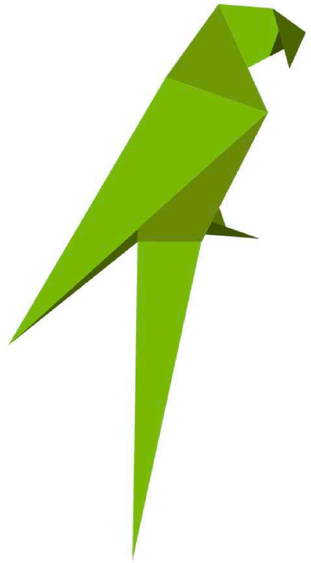
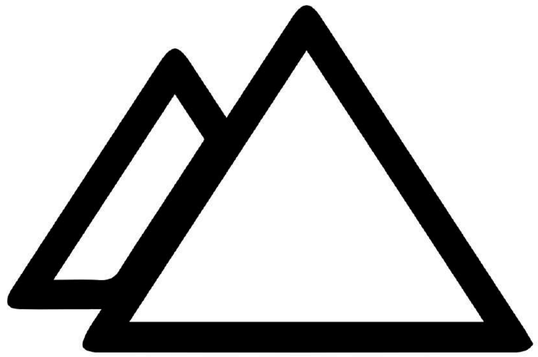
Functional UI with Scenic

Code BEAM STO 2019



boyd multerer
kry10


300 km
300 miles
Bay of Islands
Ninety Mile Beach
SOUTH PACIFIC OCEAN
Wangarei
Hauraki Gulf
TASMAN SEA
NORTHLAND
Great Barrier Island
Coromandel Peninsula
Auckland
North Island
Bay of Plenty
Hamilton
Tauranga
East Cape
Rotorua
Gisborne
New Plymouth
Lake Taupo
Napier
Hastings
NEW ZEALAND
COOK STRAIT
Palmerston North
Nelson
WELLINGTON
South Island
SOUTHERN ALPS
Kaikoura
Fox Glacier
Christchurch
Milford Sound
Mt Cook
Timaru
Queenstown
Oamaru
FJORDLAND
Te Anau
Dunedin
Invercargill
SOUTH PACIFIC OCEAN
Stewart Island
© wordtravels.com



background

consoles

does one thing. play games!

hacking target

hardware efficient

centrally managed

connected devices

single purpose devices

each does something important

minimal hardware for the job

often managed in fleets

important themes

Resilience

Security

Efficiency

Management

BEAM & OTP

a great fit for for connected devices

resilient / efficient / manageable

not perfect, but very very good

M vs A processors

UI considerations

specific compute vs. generic

resource constraints

security constraints

must be accessible

local browser

very flexible. easy to update. well understood

big

complex

importing security issues

local web server

very flexible. well understood

open inbound ports -> security issue

a viable option for some scenarios

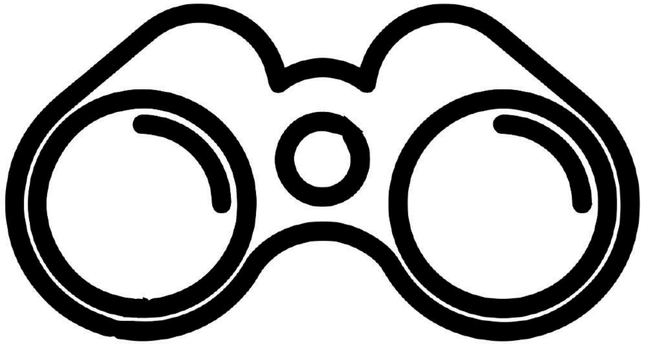
Scenic goals

robust

small / efficient

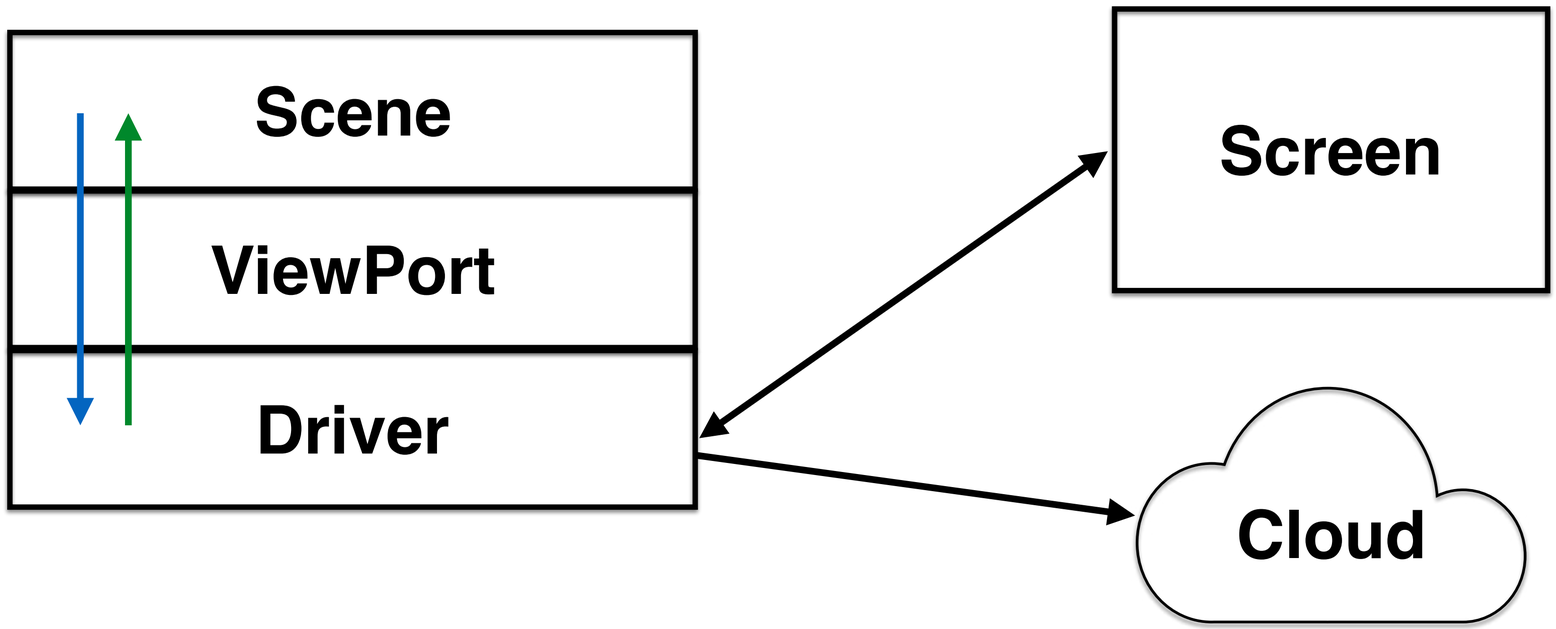
approachable

remote-able yet secure



scenic overview

scenic architecture



anatomy of a Scene

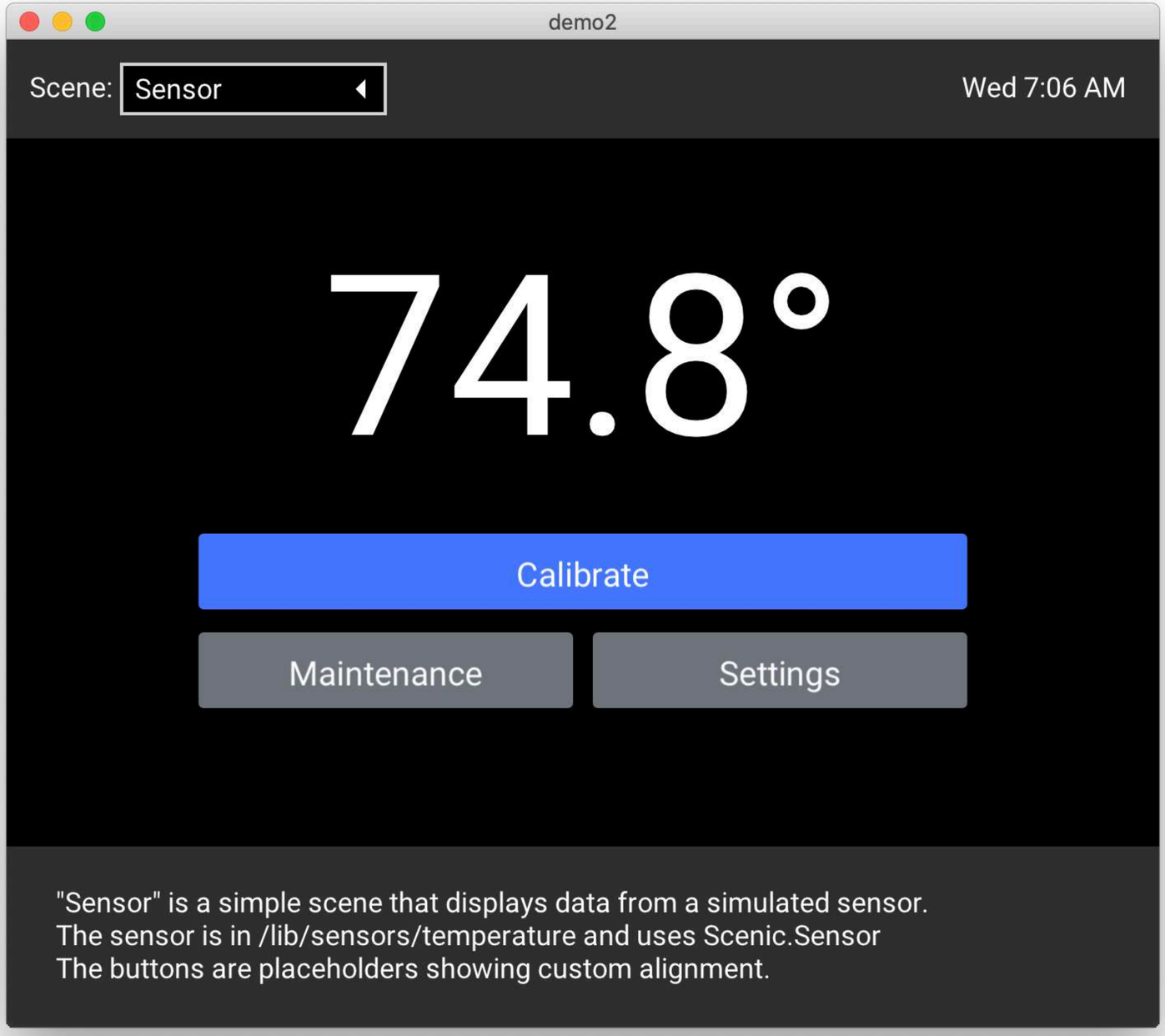
is a process

defines a graph

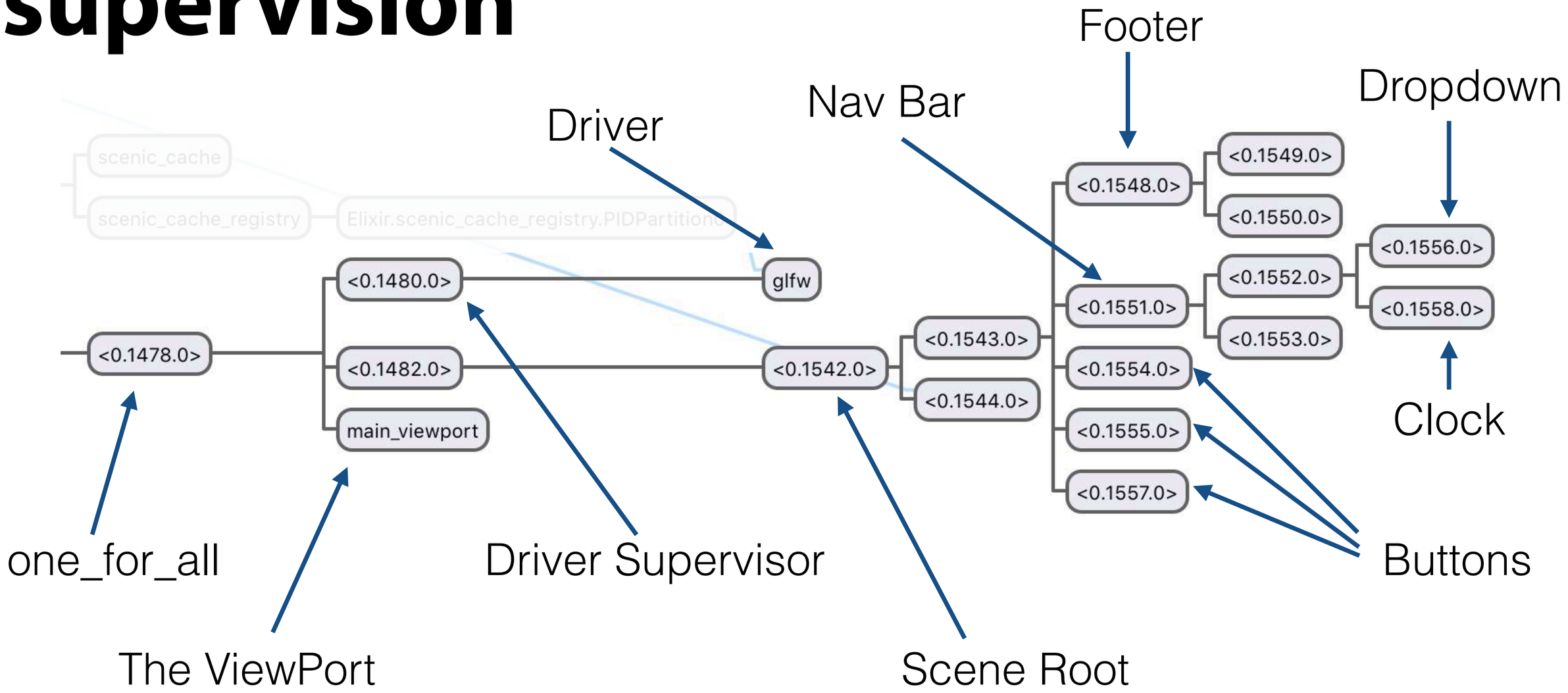
handles input and events

knows nothing about the drivers

reusable as Components


demo2
Scene:
Sensor
Wed 7:06 AM
74.8°
Calibrate
Maintenance
Settings
"Sensor" is a simple scene that displays data from a simulated sensor.
The sensor is in /lib/sensors/temperature and uses Scenic.Sensor
The buttons are placeholders showing custom alignment.

supervision



Graph

drawing state of a scene

hierarchical collection

primitives, styles, and transforms

Primitive, Style, Transform

all are a fixed set

Primitive: basic shapes

Style: affect **how** a Primitive is drawn

Transform: affect **where** a Primitive is drawn

components

encapsulated scenes

meant to be reused

both drawing and event handling

a growing library

input & events

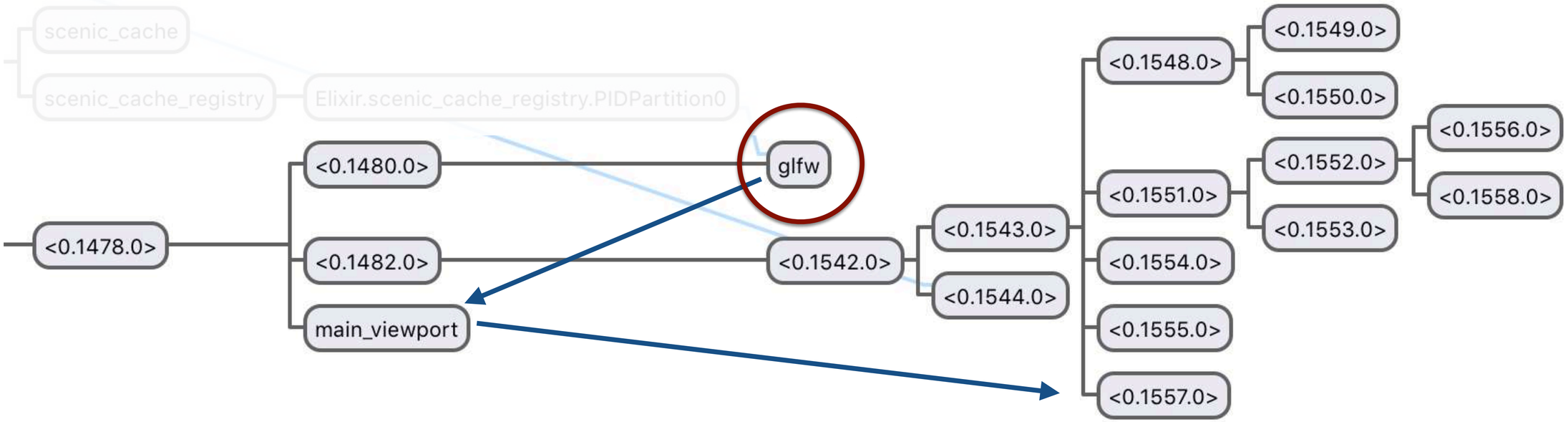
input is generated by the drivers

flow through the ViewPort

events are generated by components

flow backwards up supervision tree

user input

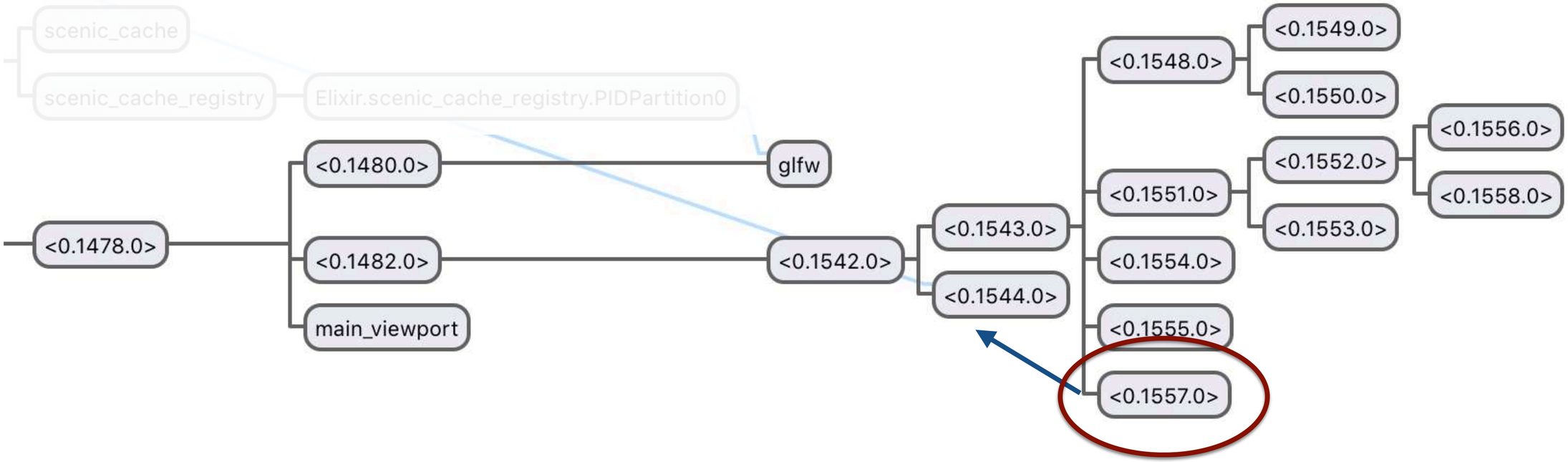


User input is generated at a driver

The input is passed to the ViewPort for processing

The ViewPort then passes it to the appropriate scene to handle

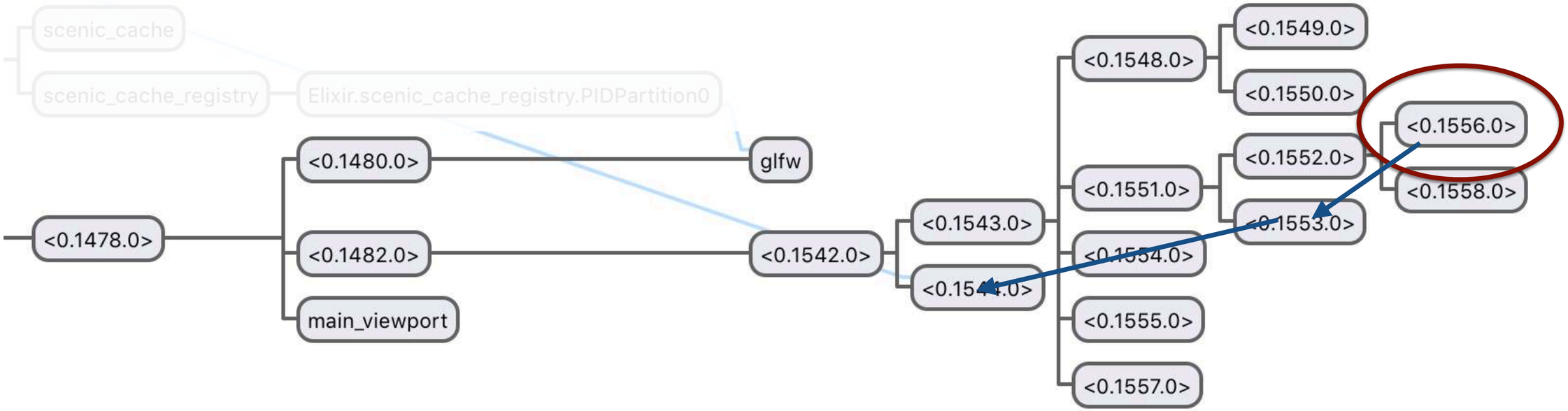
events



A component generates an event in response to… something

That event is then passed to it’s parent scene for handling

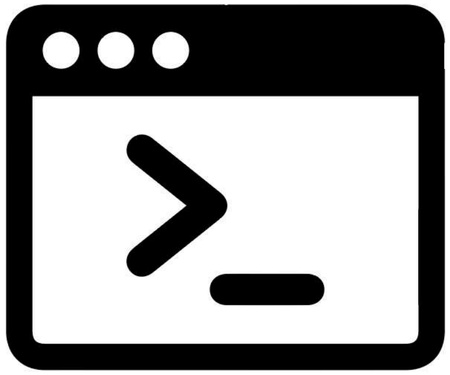
events



A component generates an event in response to… something

That event is then passed to it's parent scene for handling

If the event is unhandled, it is passed to that scene's parent



demos

recent additions

proper support for Nerves

font metrics / TrueType parser

various components

much more…

todo list

finish modal dialogs

live recompile

release the cloud service

iex scene / observer light

more…

fin

twitter: boydmulterer
website: www.kry10.com