Tizen Native Development Tips to Save Time & Trouble

Tizen Developer Conference 2015

深圳，中华人民共和国

Shenzhen, People's Republic of China

Carsten Haitzler <c.haitzler@samsung.com>

Master Engineer Samsung Electronics, Korea
Enlightenment / EFL Founder

Tizen Development Today

Tizen Development Today

HTML5

HTML5

Tizen Development Today

HTML5

HTML5 + Native

HTML5 + Native

HTML5

HTML5 + Native

Why Native or HTML5?

Pros & Cons

HTML5

Native

Pros & Cons

HTML5	Native
+ Re-use Web Tech knowledge	

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms

Native

Pros & Cons

HTML5	Native
+ Re-use Web Tech knowledge	
+ Share code across platforms	
+ Develop apps quickly	

Pros & Cons

HTML5

+ Re-use W

+ S

+ F

Native

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 

Native

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)

Native

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory

Native

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)
+ Use less memory

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)
+ Use less memory
+ Less restricted on Tizen

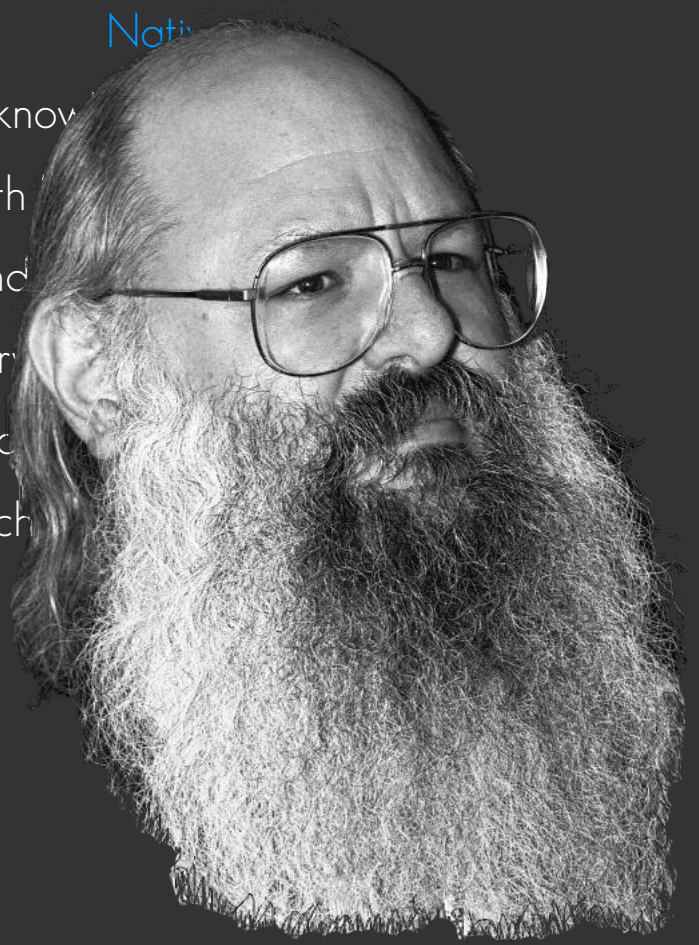
Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Nati

+ Re-use C/C++ know
+ Share code with
+ Fast (startup and
+ Use less memor
+ Less restricted c
+ Grow an old-sch



Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)
+ Use less memory
+ Less restricted on Tizen
+ Grow an old-school UNIX beard 

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)
+ Use less memory
+ Less restricted on Tizen
+ Grow an old-school UNIX beard 
- More prone to programmer error

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)
+ Use less memory
+ Less restricted on Tizen
+ Grow an old-school UNIX beard 
- More prone to programmer error
- Takes more time to develop

Pros & Cons

HTML5

+ Re-use Web Tech knowledge
+ Share code across platforms
+ Develop apps quickly
+ Be hip & trendy 
- Slow (startup and execute)
- Use more memory
- More restricted on Tizen

Native

+ Re-use C/C++ knowledge
+ Share code with large codebases
+ Fast (startup and execute)
+ Use less memory
+ Less restricted on Tizen
+ Grow an old-school UNIX beard 
- More prone to programmer error
- Takes more time to develop
- Different APIs to other platforms

Native

Native

- Used by system developers for most apps

Native

- Used by system developers for most apps
- Apps can be C or C++

Native

- Used by system developers for most apps
- Apps can be C or C++
  - APIs to call in Tizen are C (Can be called from C++ as well as C)

Native

- Used by system developers for most apps
- Apps can be C or C++
  - APIs to call in Tizen are C (Can be called from C++ as well as C)
- Can use Tizen SDK or GBS to build

Native

- Used by system developers for most apps
- Apps can be C or C++
  - APIs to call in Tizen are C (Can be called from C++ as well as C)
- Can use Tizen SDK or GBS to build
  - GBS is a Linux command-line tool to cross-compile Tizen software

Native

- Used by system developers for most apps
- Apps can be C or C++
  - APIs to call in Tizen are C (Can be called from C++ as well as C)
- Can use Tizen SDK or GBS to build
  - GBS is a Linux command-line tool to cross-compile Tizen software
    - Entire OS is built using GBS on servers

Native

- Used by system developers for most apps
- Apps can be C or C++
  - APIs to call in Tizen are C (Can be called from C++ as well as C)
- Can use Tizen SDK or GBS to build
  - GBS is a Linux command-line tool to cross-compile Tizen software
    - Entire OS is built using GBS on servers
  - SDK is Eclipse based

Native API Content

Native API Content

- Majority of Native API comes from EFL

Native API Content

- Majority of Native API comes from EFL
  - Open Source Project - http://www.enlightenment.org

Native API Content

- Majority of Native API comes from EFL
  - Open Source Project – http://www.enlightenment.org
  - Covers Main loop (UI), Rendering, OpenGL, Widgets, Comms, IPC, Threading, Theme, etc.

Native API Content

- Majority of Native API comes from EFL
  - Open Source Project – http://www.enlightenment.org
  - Covers Main loop (UI), Rendering, OpenGL, Widgets, Comms, IPC, Threading, Theme, etc.
- Other Tizen Native APIs

Native API Content

- Majority of Native API comes from EFL
  - Open Source Project – http://www.enlightenment.org
  - Covers Main loop (UI), Rendering, OpenGL, Widgets, Comms, IPC, Threading, Theme, etc.
- Other Tizen Native APIs
  - Multimedia, Messaging, Sensors, Alarm, Location, Lifecycle, Network, Security, Social, Telephony, etc.

Native API Content

- Majority of Native API comes from EFL
  - Open Source Project – http://www.enlightenment.org
  - Covers Main loop (UI), Rendering, OpenGL, Widgets, Comms, IPC, Threading, Theme, etc.
- Other Tizen Native APIs
  - Multimedia, Messaging, Sensors, Alarm, Location, Lifecycle, Network, Security, Social, Telephony, etc.
  - Other open source project APIs suported like cURL, libEXIF, libXML, Sqlite etc.

Why Speak?

Why Speak?

- Founded Enlightenment / EFL development

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code
- Wrote the Window Manager / Compositor for Tizen

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code
- Wrote the Window Manager / Compositor for Tizen
  - Enlightenment is over 220,000 Lines of C code

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code
- Wrote the Window Manager / Compositor for Tizen
  - Enlightenment is over 220,000 Lines of C code
- Have worked on Tizen since before 1.0

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code
- Wrote the Window Manager / Compositor for Tizen
  - Enlightenment is over 220,000 Lines of C code
- Have worked on Tizen since before 1.0
  - 7+ years – before it was even Tizen

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code
- Wrote the Window Manager / Compositor for Tizen
  - Enlightenment is over 220,000 Lines of C code
- Have worked on Tizen since before 1.0
  - 7+ years – before it was even Tizen
  - Full-time Engineer at Samsung for 5+ Years

Why Speak?

- Founded Enlightenment / EFL development
- Wrote the largest part of EFL code personally
  - EFL is over 1,000,000 lines of C code
- Wrote the Window Manager / Compositor for Tizen
  - Enlightenment is over 220,000 Lines of C code
- Have worked on Tizen since before 1.0
  - 7+ years – before it was even Tizen
  - Full-time Engineer at Samsung for 5+ Years
- So if you have Questions – Please Ask I don't bite

Tips to make your life easier

Tips to make your life easier

- Covering the EFL parts of Tizen...

Tips to make your life easier

- Covering the EFL parts of Tizen...
- Show you some pointers to avoid mistakes

Tips to make your life easier

- Covering the EFL parts of Tizen...
- Show you some pointers to avoid mistakes
- *"If it's hard, you're doing it wrong"*

Tips to make your life easier

- Covering the EFL parts of Tizen...
- Show you some pointers to avoid mistakes
- *"If it's hard, you're doing it wrong"*
  - There is often an easier way

Tips to make your life easier

- Covering the EFL parts of Tizen...
- Show you some pointers to avoid mistakes
- *"If it's hard, you're doing it wrong"*
  - There is often an easier way
- Ask for help if you get stuck or think you could do something better

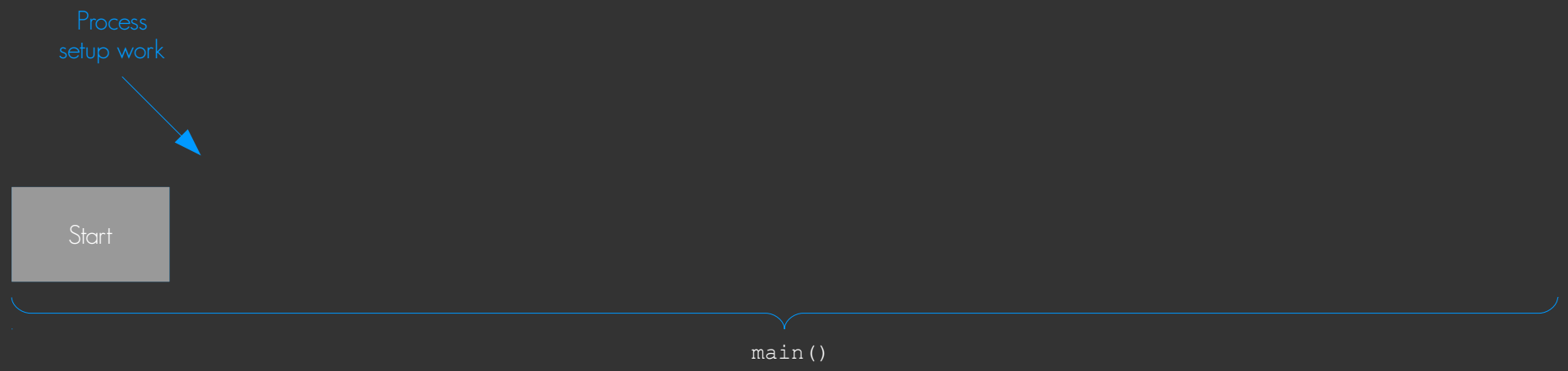
Tips to make your life easier

- Covering the EFL parts of Tizen...
- Show you some pointers to avoid mistakes
- *"If it's hard, you're doing it wrong"*
  - There is often an easier way
- Ask for help if you get stuck or think you could do something better
  - There are real humans willing to help

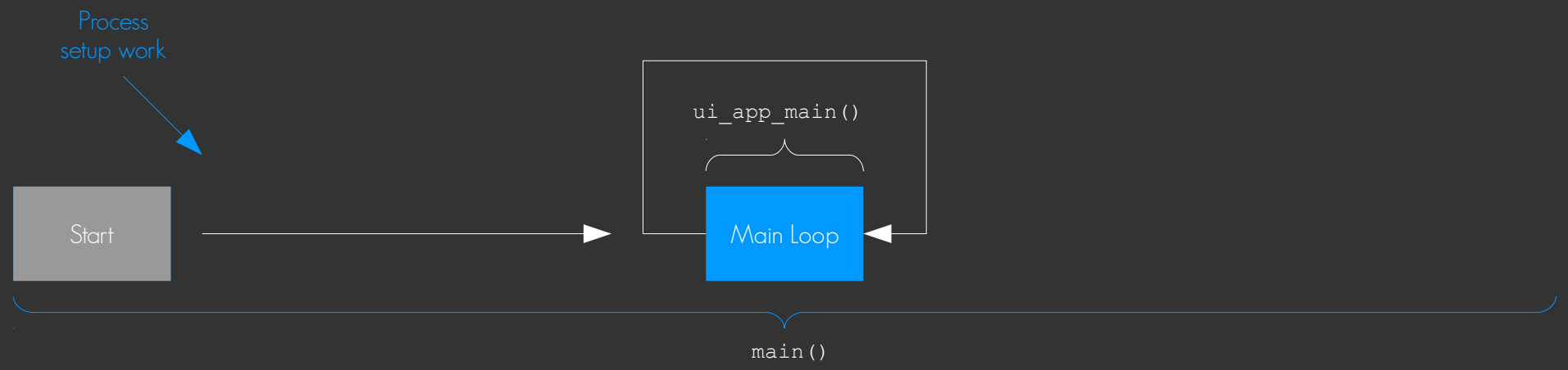
Tips to make your life easier

- Covering the EFL parts of Tizen...
- Show you some pointers to avoid mistakes
- *"If it's hard, you're doing it wrong"*
  - There is often an easier way
- Ask for help if you get stuck or think you could do something better
  - There are real humans willing to help
  - Links at end of presentation

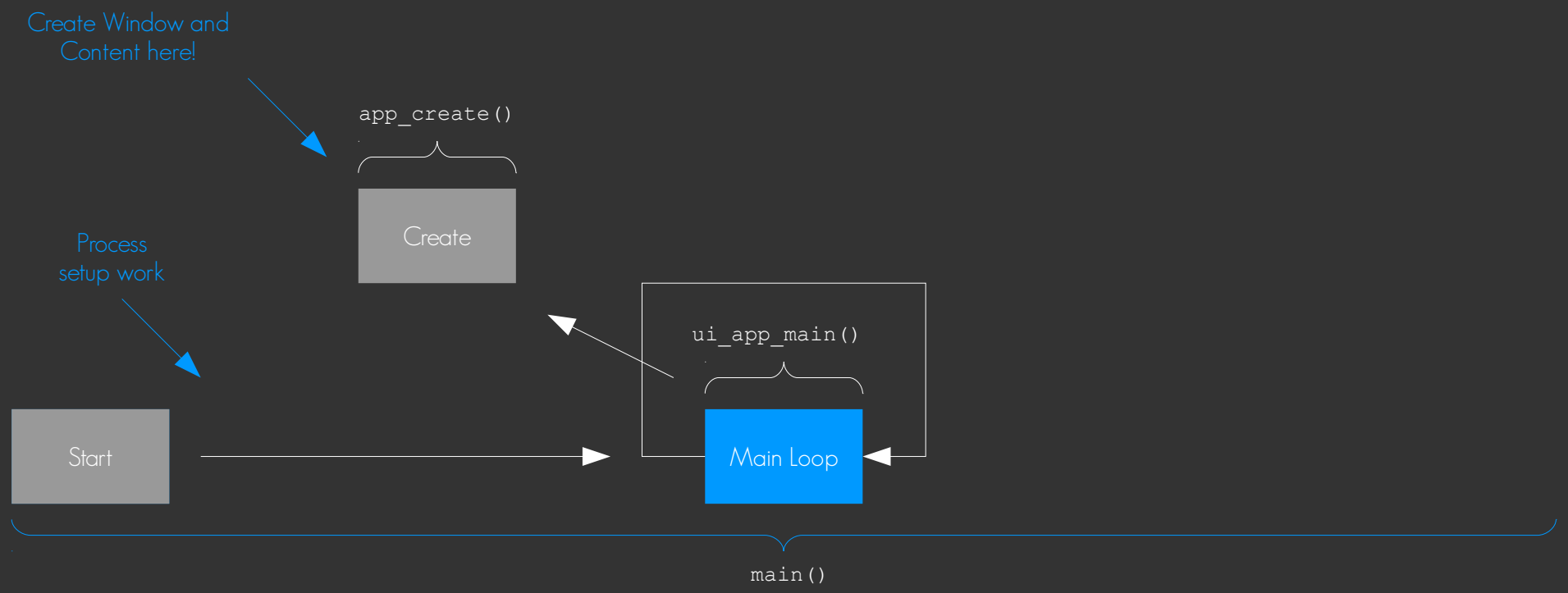
Anatomy of the Application Core



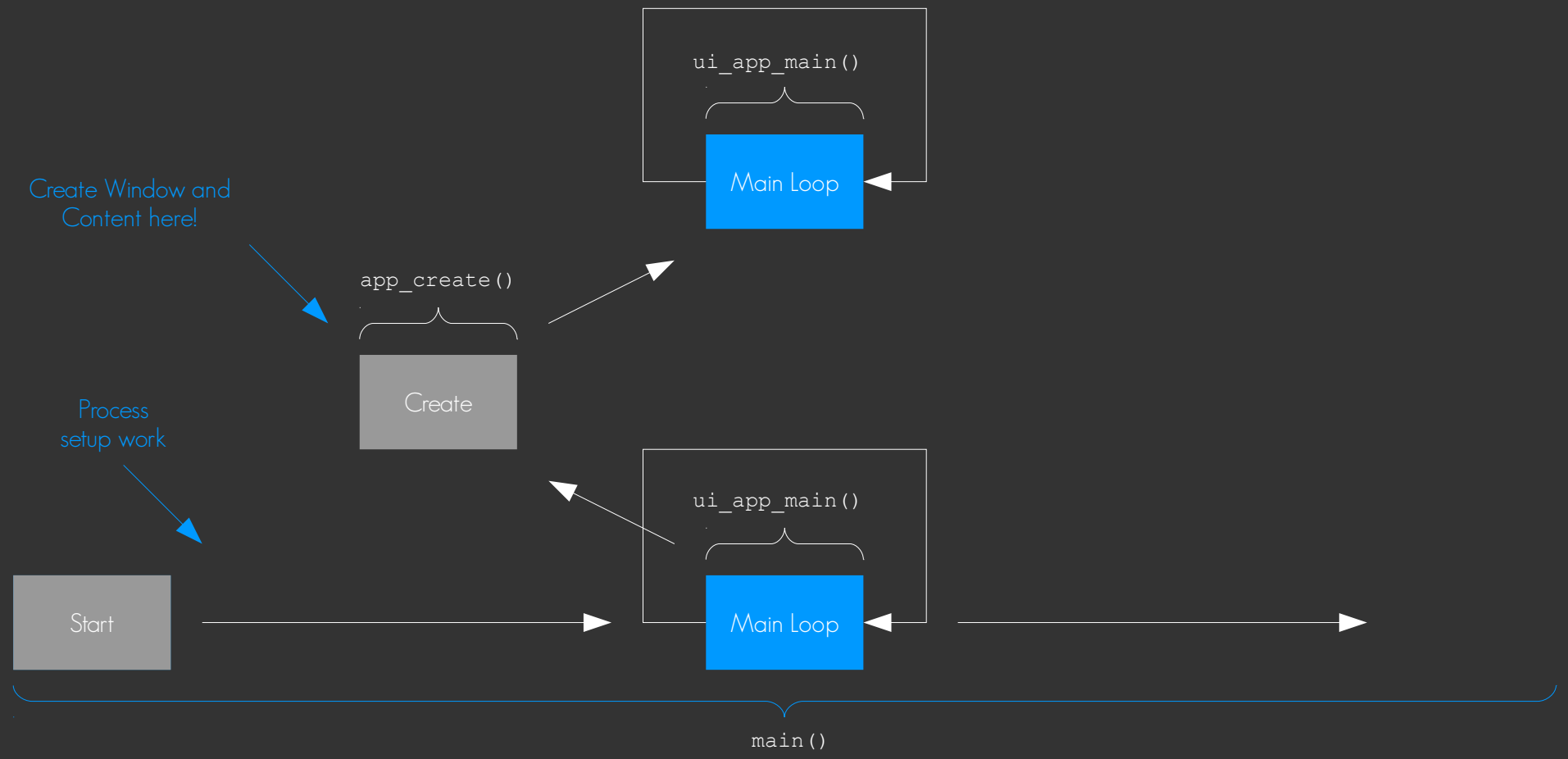
Anatomy of the Application Core



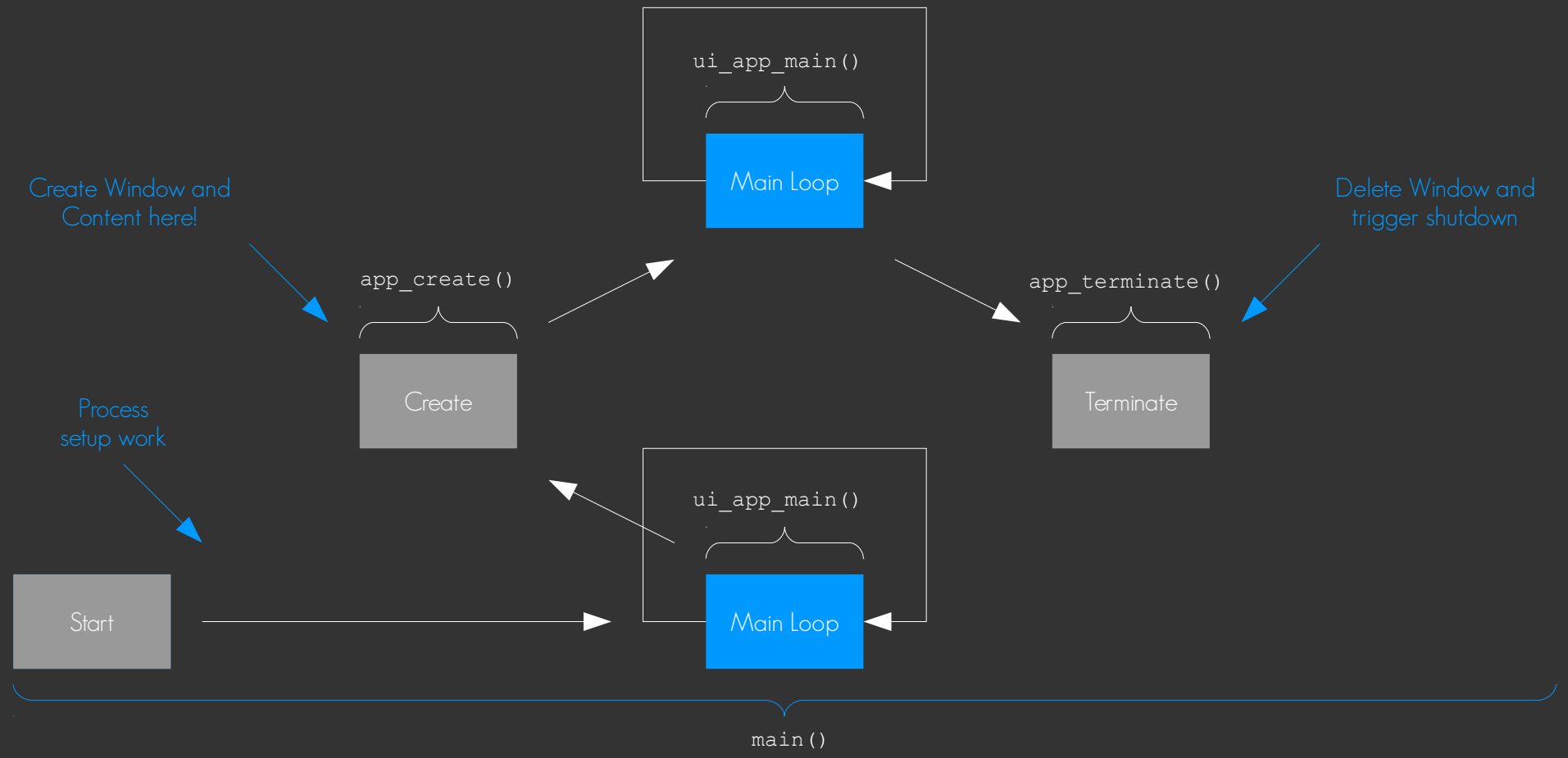
Anatomy of the Application Core



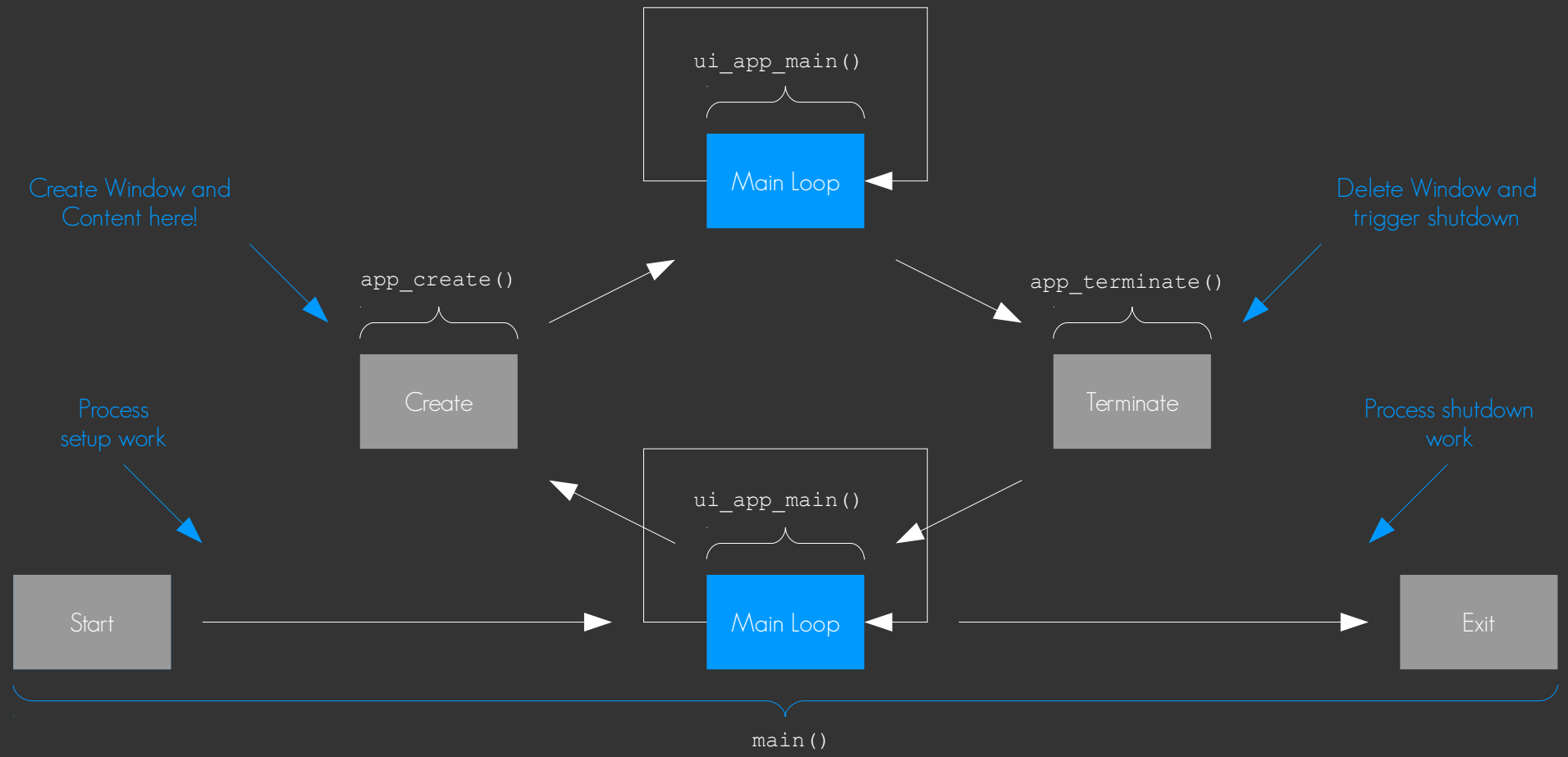
Anatomy of the Application Core



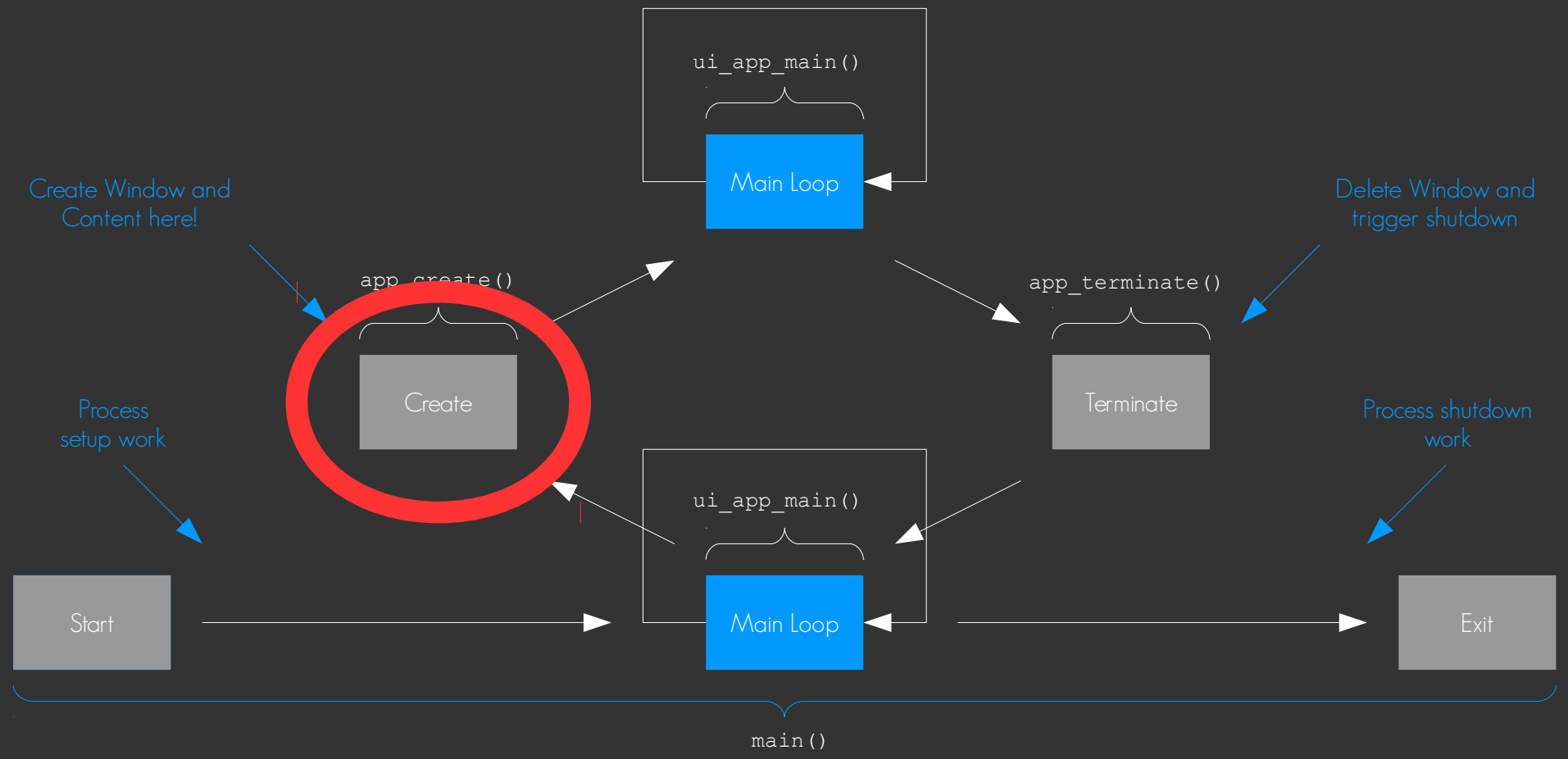
Anatomy of the Application Core



Anatomy of the Application Core



Now ... Creation of your UI



Create a Window & Content

- Add a window

Create a Window & Content

- Add a window
- Add a conformant (if you want to handle keyboard and indicator)

Create a Window & Content

- Add a window
- Add a conformant (if you want to handle keyboard and indicator)
- Add more containers/content (Naviframe, Box, Table etc.)

Create a Window & Content

- Add a window
- Add a conformant (if you want to handle keyboard and indicator)
- Add more containers/content (Naviframe, Box, Table etc.)
- Add more content (Lists, Buttons, Labels etc.)

Create a Window & Content

- Add a window
- Add a conformant (if you want to handle keyboard and indicator)
- Add more containers/content (Naviframe, Box, Table etc.)
- Add more content (Lists, Buttons, Labels etc.)
- Add callbacks to objects to listen to events

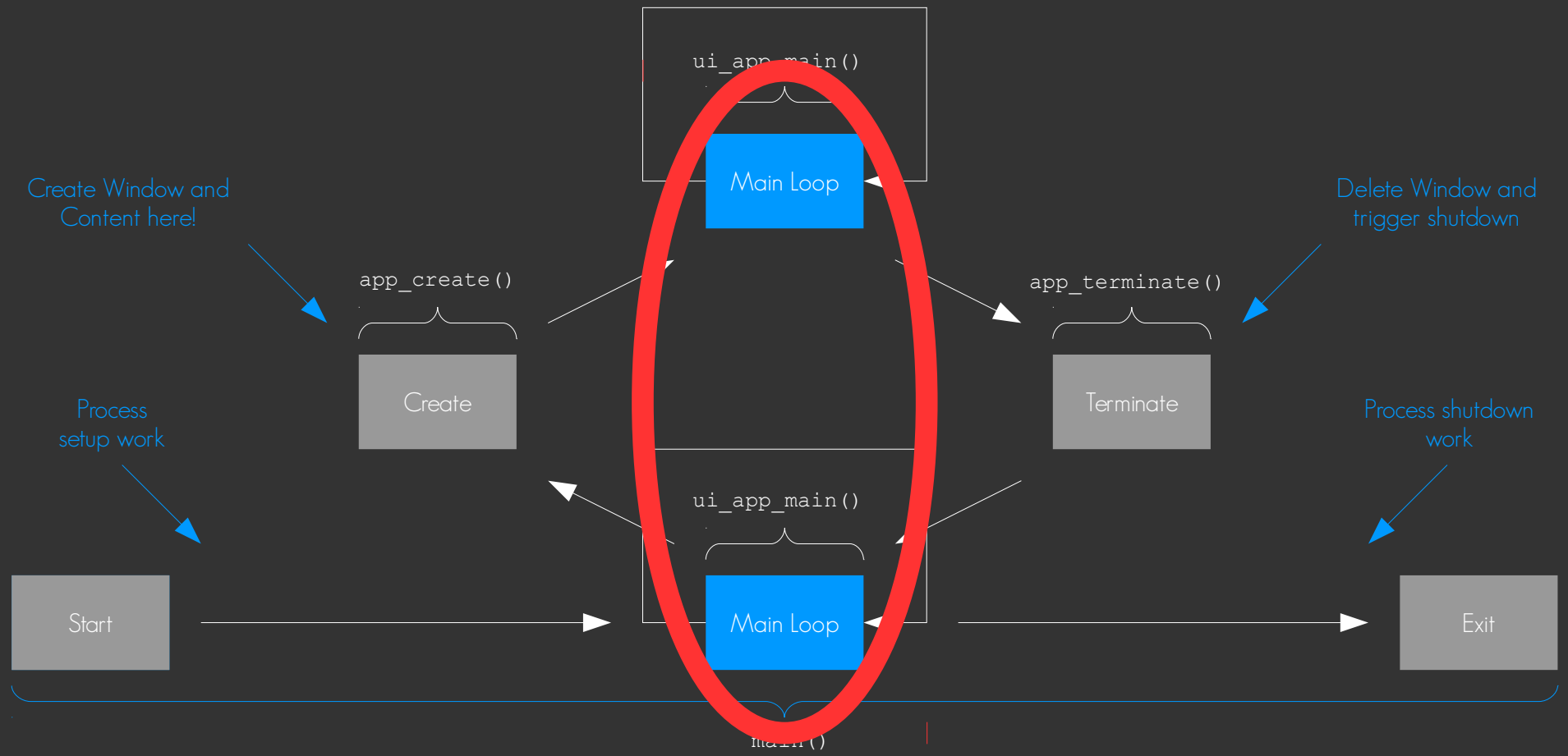
Create a Window & Content

- Add a window
- Add a conformant (if you want to handle keyboard and indicator)
- Add more containers/content (Naviframe, Box, Table etc.)
- Add more content (Lists, Buttons, Labels etc.)
- Add callbacks to objects to listen to events
- Show window

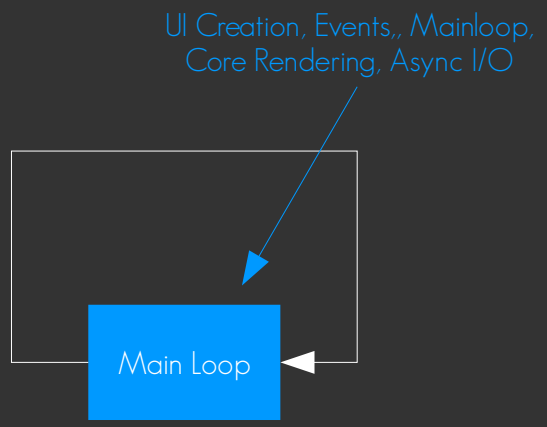
Create a Window & Content

- Add a window
- Add a conformant (if you want to handle keyboard and indicator)
- Add more containers/content (Naviframe, Box, Table etc.)
- Add more content (Lists, Buttons, Labels etc.)
- Add callbacks to objects to listen to events
- Show window
- ... Let main loop run and call callbacks

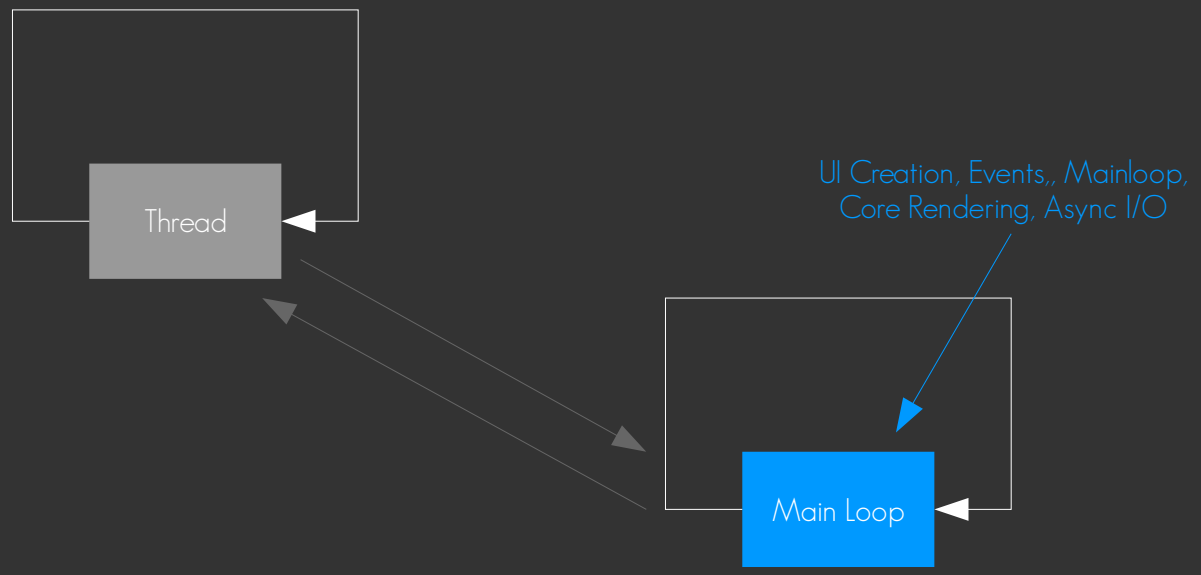
Now ... Mainloop Interactions



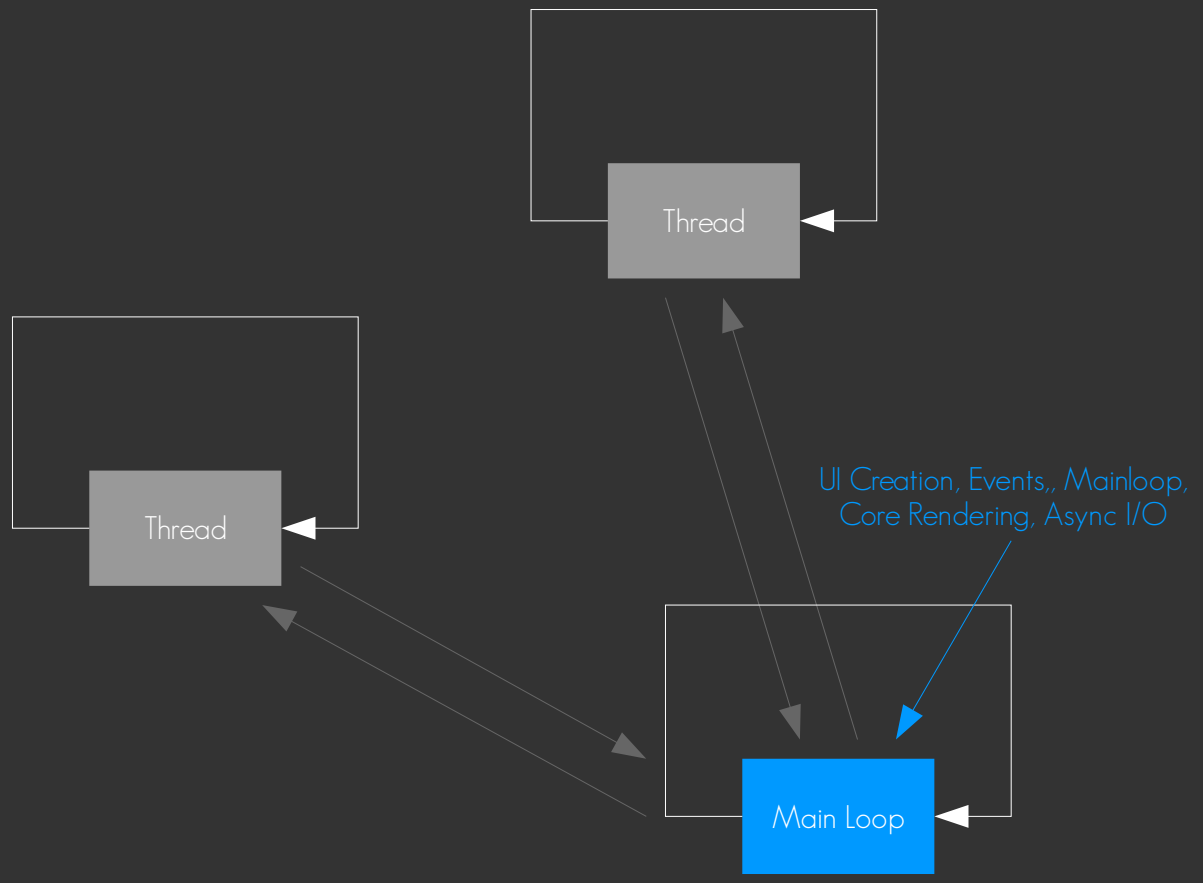
Application Core & Threads



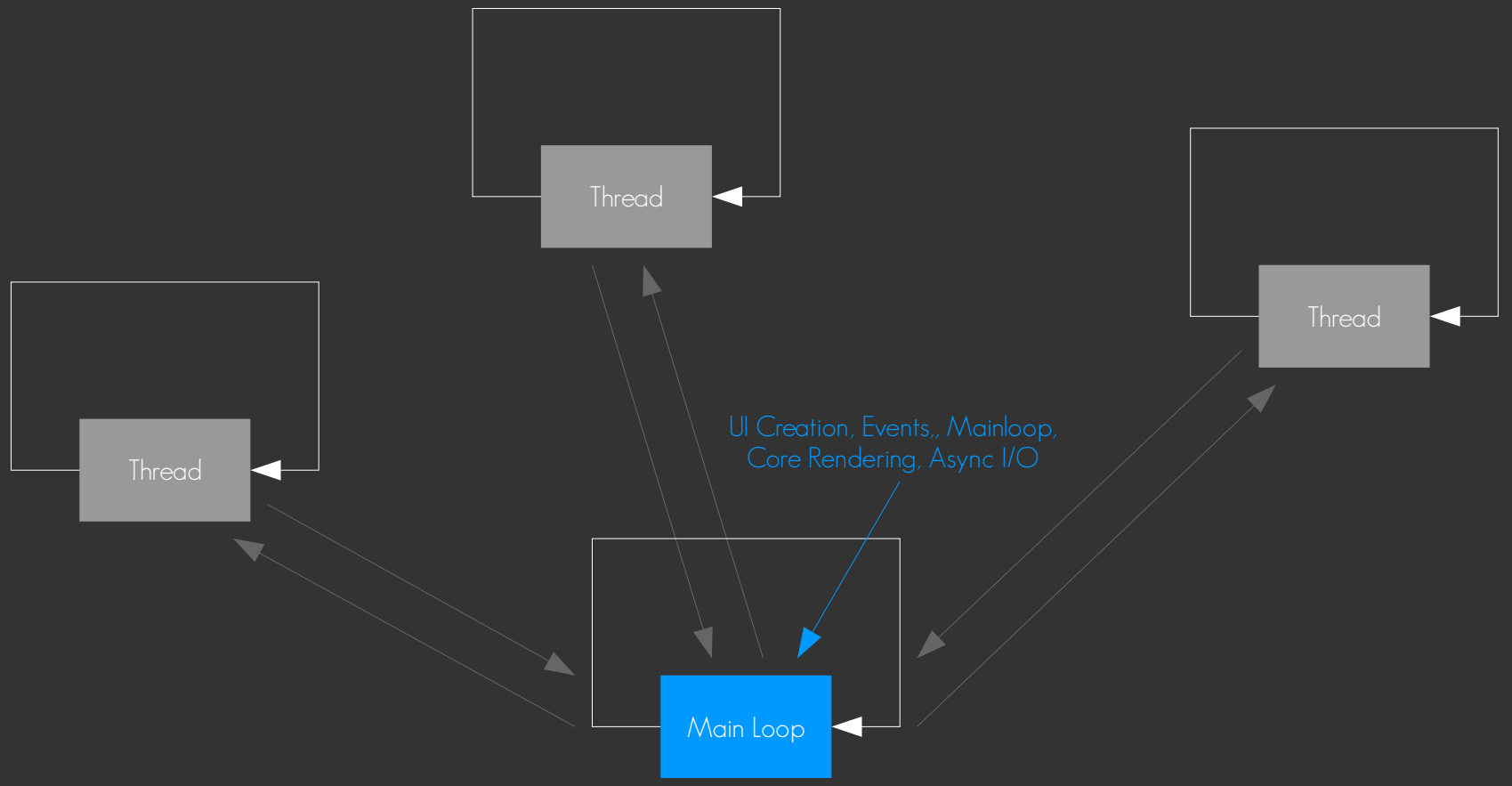
Application Core & Threads



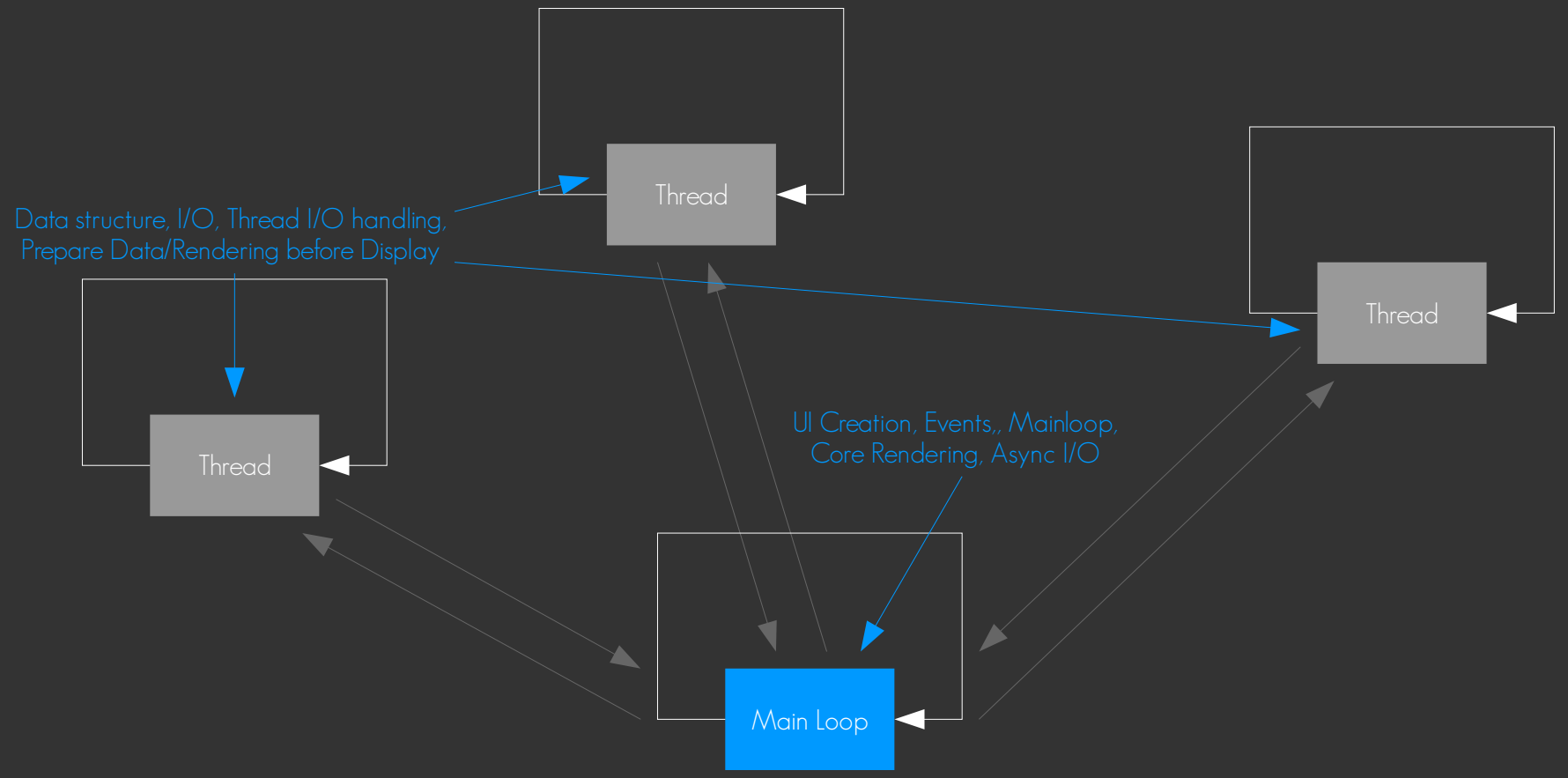
Application Core & Threads



Application Core & Threads



Application Core & Threads



Application Core & Threads

Thread Safe

Main Loop Only

Application Core & Threads

Thread Safe

```
LibC Calls…
eina_…
ecore_main_loop_thread_safe_call_async()
ecore_main_loop_thread_safe_call_sync()
ecore_thread_main_loop_begin()
ecore_thread_main_loop_end()
ecore_thread_check()
ecore_thread_feedback()
ecore_thread_reschedule()
ecore_thread_local_data_add()
ecore_thread_local_data_set()
ecore_thread_local_data_find()
ecore_thread_local_data_del()
ecore_thread_global_data_add()
ecore_thread_global_data_set()
ecore_thread_global_data_find()
ecore_thread_global_data_del()
ecore_thread_global_data_wait()
ecore_pipe_write()
ecore_pipe_read_fd()
ecore_pipe_write_fd()
ecore_pipe_wait()
```

Main Loop Only

Application Core & Threads

Thread Safe

```
LibC Calls…
eina_…
ecore_main_loop_thread_safe_call_async()
ecore_main_loop_thread_safe_call_sync()
ecore_thread_main_loop_begin()
ecore_thread_main_loop_end()
ecore_thread_check()
ecore_thread_feedback()
ecore_thread_reschedule()
ecore_thread_local_data_add()
ecore_thread_local_data_set()
ecore_thread_local_data_find()
ecore_thread_local_data_del()
ecore_thread_global_data_add()
ecore_thread_global_data_set()
ecore_thread_global_data_find()
ecore_thread_global_data_del()
ecore_thread_global_data_wait()
ecore_pipe_write()
ecore_pipe_read_fd()
ecore_pipe_write_fd()
ecore_pipe_wait()
```

Main Loop Only

```
elm_…
evas_…
edje_…
ecore_…
…
```

Call “Main Loop” Calls from Thread

Call “Main Loop” Calls from Thread

```
…
ecore_thread_main_loop_begin(); { // Begin main loop code
  …
  timer = ecore_timer_add(42.0, my_timer_callback, NULL);
  evas_object_move(my_object, x, y);
  …
} ecore_thread_main_loop_end(); // End main loop code
…
```

Very simple & easy way to call rare blocks of code within main loop context from a thread

Call "Main Loop" Calls from Thread

```
…
ecore_thread_main_loop_begin(); { // Begin main loop code
  …
  timer = ecore_timer_add(42.0, my_timer_callback, NULL);
  evas_object_move(my_object, x, y);
  …
} ecore_thread_main_loop_end(); // End main loop code
…
```

Very simple & easy way to call rare blocks of code within main loop context from a thread

or

```
…
void my_main_loop_code(void *data) { // Begin main loop code
  My_Data *my_data = data;
  …
  timer = ecore_timer_add(42.0, my_timer_callback, NULL);
  evas_object_move(my_data->obj, my_data->x, my_data->y);
  free(my_data);
  …
} // End main loop code
…

…
// Queue function to be called in the main loop
My_Data *my_data = calloc(1, sizeof(My_Data));
my_data->x = x;
my_data->y = y;
my_data->obj = my_object;
ecore_main_loop_thread_safe_call_async(my_main_loop_code, my_data);
…
```

Very efficent way to call code async very frequently from thread in main loop context

Thread Worker Pool

Thread Worker Pool

- You can also use standard worker pool

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue
    - Keep work items short

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue
    - Keep work items short
    - Pool of workers pull thread items off the queue and hand results back to mainloop

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue
    - Keep work items short
    - Pool of workers pull thread items off the queue and hand results back to mainloop
    - Limited size of pool to avoid overloading CPU (pool size based on number of cores)

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue
    - Keep work items short
    - Pool of workers pull thread items off the queue and hand results back to mainloop
    - Limited size of pool to avoid overloading CPU (pool size based on number of cores)
    - Saves managing your own thread pool

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue
    - Keep work items short
    - Pool of workers pull thread items off the queue and hand results back to mainloop
    - Limited size of pool to avoid overloading CPU (pool size based on number of cores)
    - Saves managing your own thread pool
    - Simple to use for tasks easily divided up into N discrete small units

Thread Worker Pool

- You can also use standard worker pool
  - `ecore_thread_new()` adds a thread work item to the queue
    - Keep work items short
    - Pool of workers pull thread items off the queue and hand results back to mainloop
    - Limited size of pool to avoid overloading CPU (pool size based on number of cores)
    - Saves managing your own thread pool
    - Simple to use for tasks easily divided up into N discrete small units
- See Tizen and Elementary docs for more threading usage examples

Threading Summary

Threading Summary

- Use threads, but design carefully

Threading Summary

- Use threads, but design carefully
  - Divide tasks into isolated work per thread

Threading Summary

- Use threads, but design carefully
  - Divide tasks into isolated work per thread
    - Minimizes possible bugs by minimizing locking requirements

Threading Summary

- Use threads, but design carefully
  - Divide tasks into isolated work per thread
    - Minimizes possible bugs by minimizing locking requirements
    - Mainloop "collects results & implements display state"

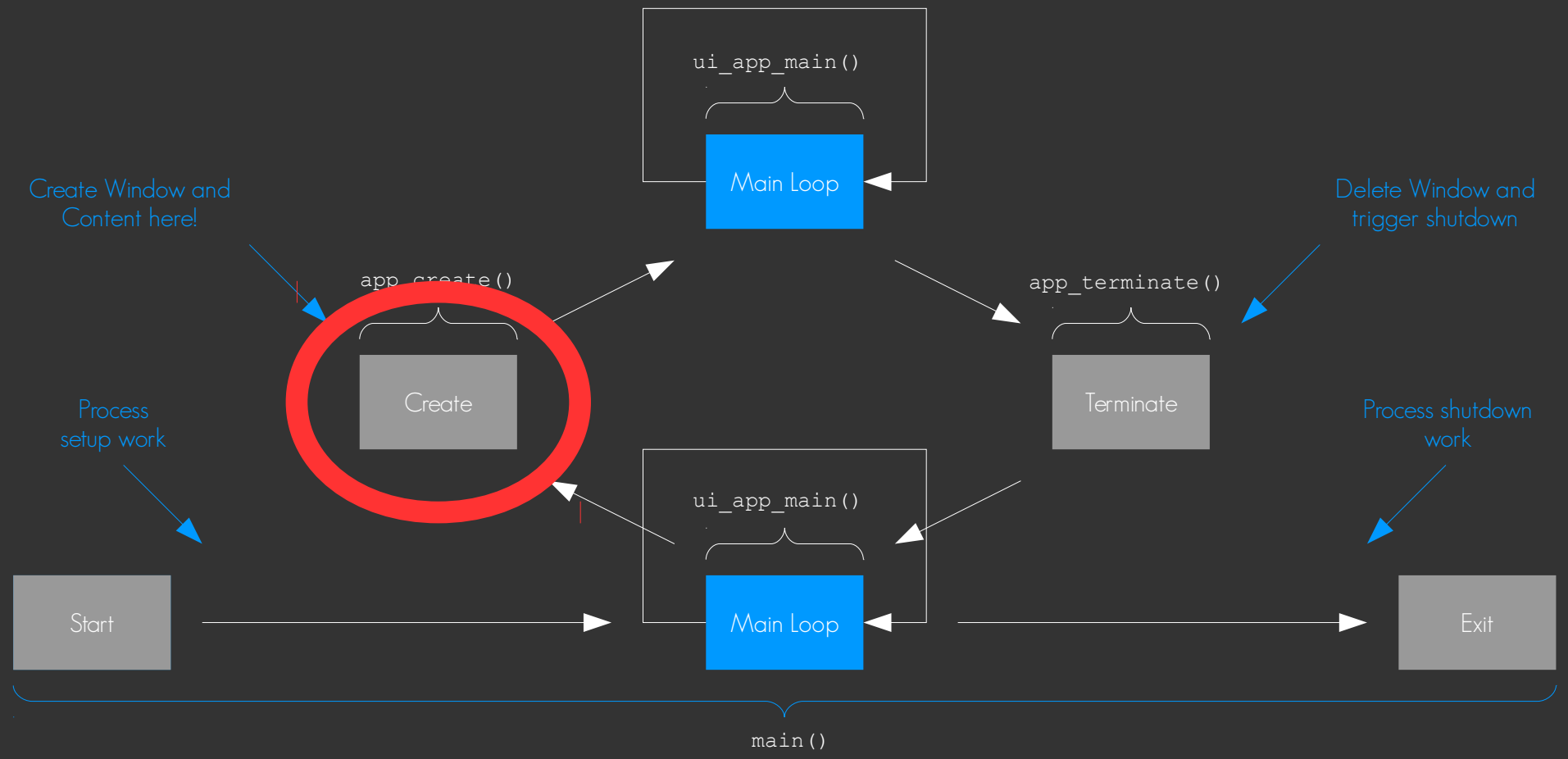
Threading Summary

- Use threads, but design carefully
  - Divide tasks into isolated work per thread
    - Minimizes possible bugs by minimizing locking requirements
    - Mainloop "collects results & implements display state"
    - Use ecore thread infra to save re-inventing the wheel

Threading Summary

- Use threads, but design carefully
  - Divide tasks into isolated work per thread
    - Minimizes possible bugs by minimizing locking requirements
    - Mainloop "collects results & implements display state"
    - Use ecore thread infra to save re-inventing the wheel
- If you have issues, please report/bring them up!

Now … Back to the Window that is created



Window Content

Window Content

- The window is a full scene graph

Window Content

- The window is a full scene graph
  - Everything you see is an object that persists

Window Content

- The window is a full scene graph
  - Everything you see is an object that persists
  - Changes to all objects are stateful

Window Content

- The window is a full scene graph
  - Everything you see is an object that persists
  - Changes to all objects are stateful
    - They retain their state as it was last set

Window Content

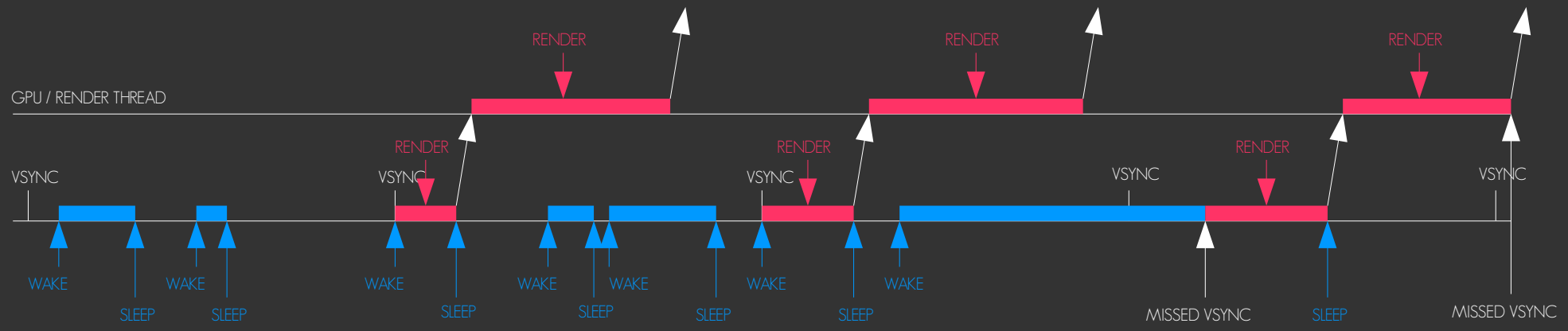
- The window is a full scene graph
  - Everything you see is an object that persists
  - Changes to all objects are stateful
    - They retain their state as it was last set
  - Rendering is “automatic” after going idle at next VSYNC event

Window Content

- The window is a full scene graph
  - Everything you see is an object that persists
  - Changes to all objects are stateful
    - They retain their state as it was last set
  - Rendering is “automatic” after going idle at next VSYNC event
    - Hidden/abstracted so acceleration methods can be dramatically changed

Window Content

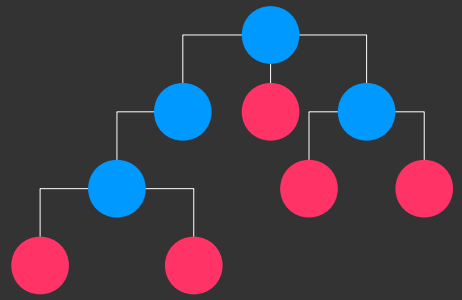
- The window is a full scene graph
  - Everything you see is an object that persists
  - Changes to all objects are stateful
    - They retain their state as it was last set
  - Rendering is "automatic" after going idle at next VSYNC event
    - Hidden/abstracted so acceleration methods can be dramatically changed



Scene Graph

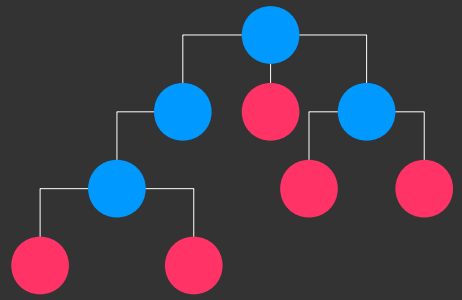
Scene Graph

- Windows hold a tree of objects



Scene Graph

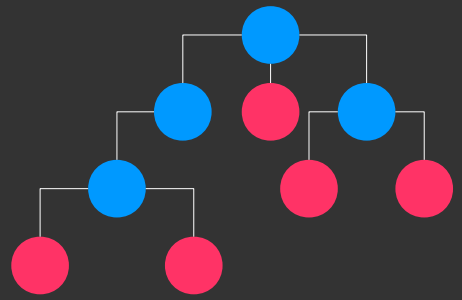
- Windows hold a tree of objects



- Each object can be a basic object or a container (Smart Object)

Scene Graph

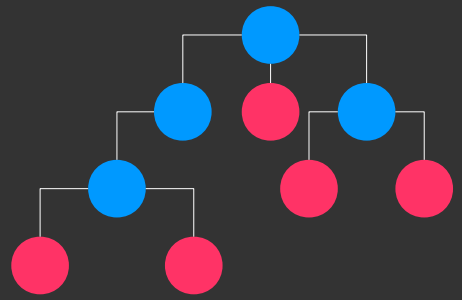
- Windows hold a tree of objects



- Each object can be a basic object or a container (Smart Object)
  - All objects are stacked and have geometry (x, y, width & height)

Scene Graph

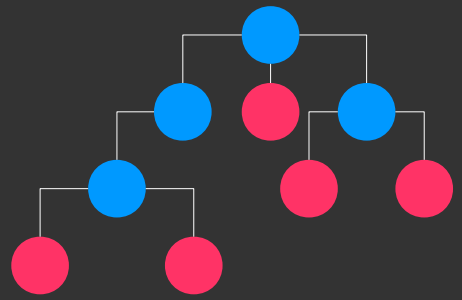
- Windows hold a tree of objects



- Each object can be a basic object or a container (Smart Object)
  - All objects are stacked and have geometry (x, y, width & height)
  - All child objects stack within the parent object (recursively)

Scene Graph

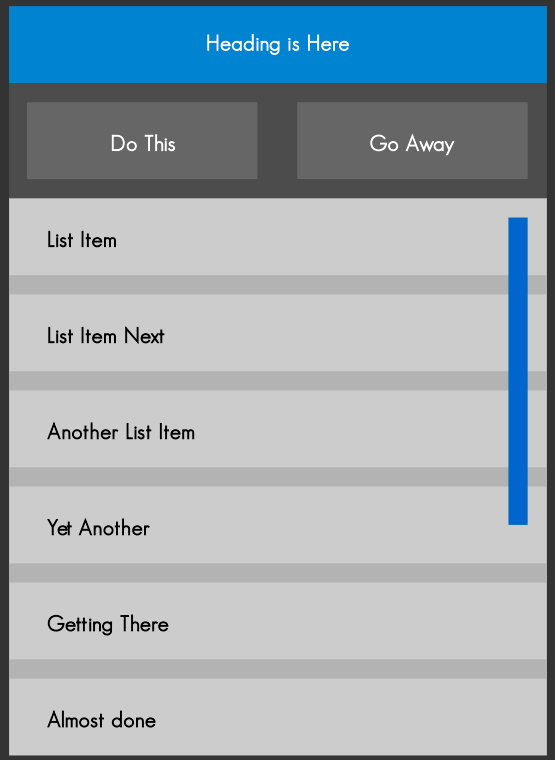
- Windows hold a tree of objects



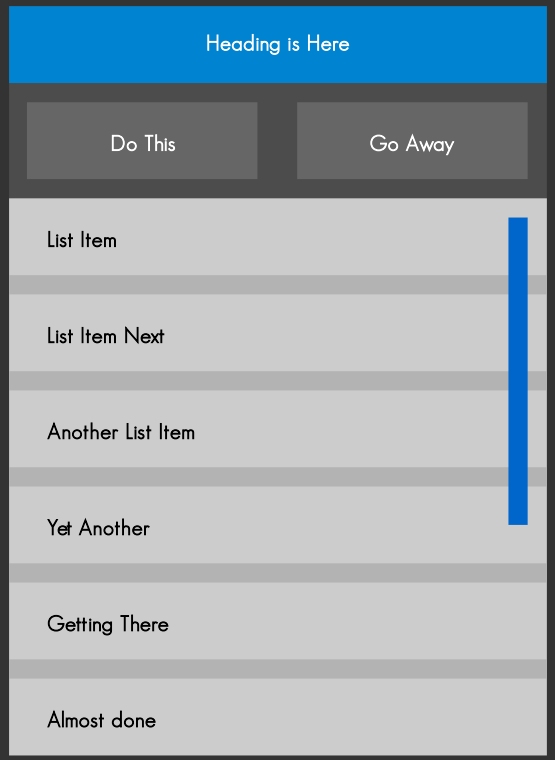
- Each object can be a basic object or a container (Smart Object)
  - All objects are stacked and have geometry (x, y, width & height)
  - All child objects stack within the parent object (recursively)

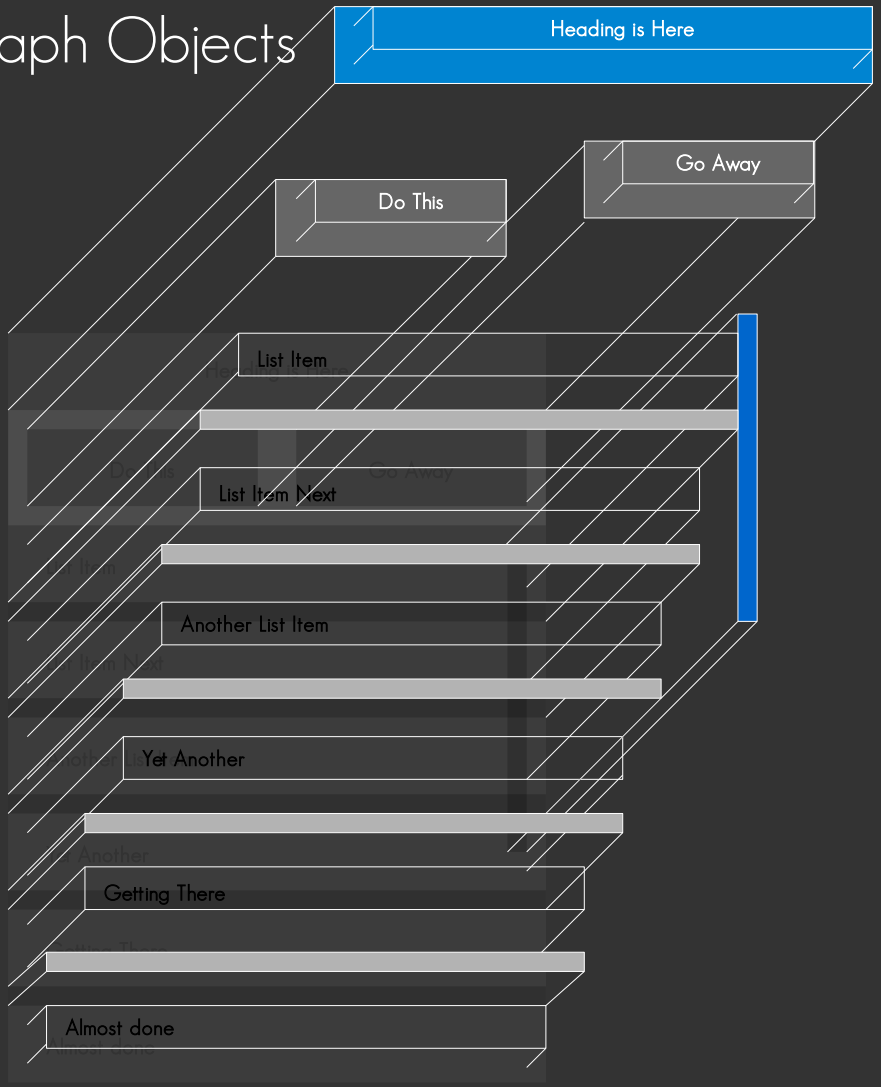
- Basic objects
  - Rectangle
  - Image (images, buffers, proxies)
  - Text (single line simple text)
  - Textblock (multi-line formatted text)
  - Textgrid (for grids of chars)
  - Polygon (used for event regions)
  - Line (really limited)
  - VG (Vector Graphic)
  - 3D

Scene Graph Objects

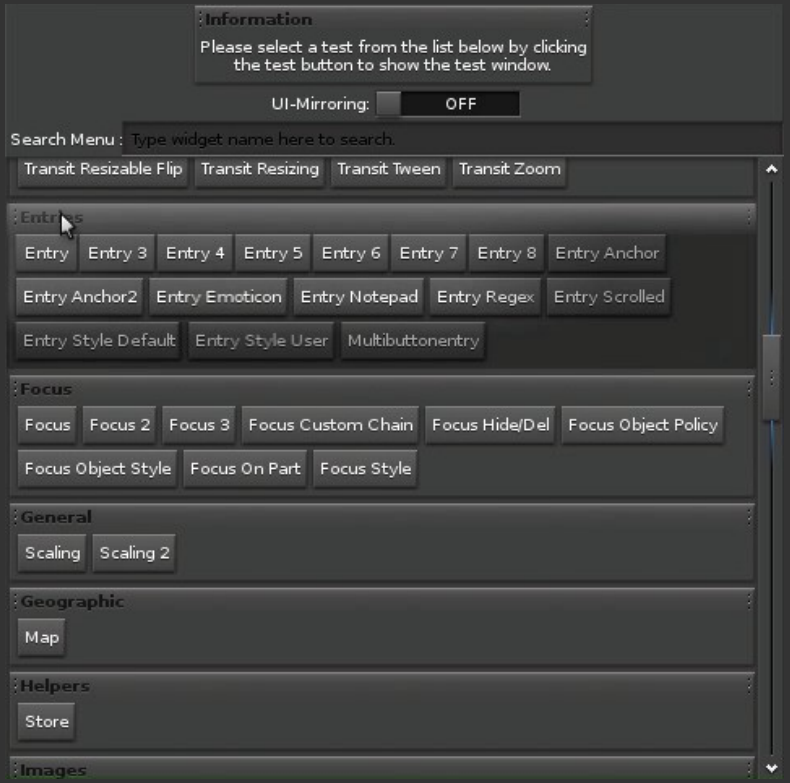


Scene Graph Objects





Object Components & Layers



Queue deferred render

Queue deferred render

- All objects retain state

Queue deferred render

- All objects retain state
  - Show object once – it stays visible UNTIL hidden

Queue deferred render

- All objects retain state
  - Show object once – it stays visible UNTIL hidden
  - Set a color of an object – it stays that color until it is changed

Queue deferred render

- All objects retain state
  - Show object once – it stays visible UNTIL hidden
  - Set a color of an object – it stays that color until it is changed
  - Set text of text or textblock object – it shows that text until changed

Queue deferred render

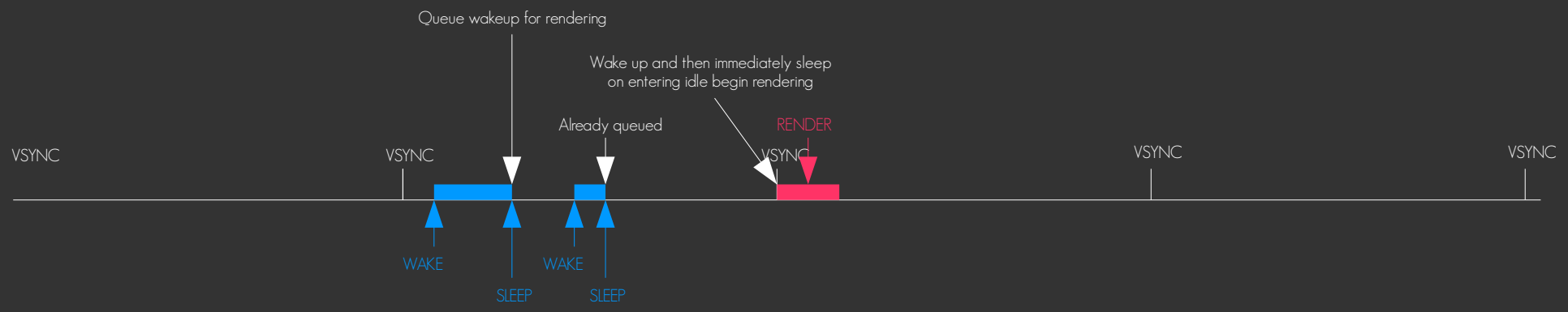
- All objects retain state
  - Show object once – it stays visible UNTIL hidden
  - Set a color of an object – it stays that color until it is changed
  - Set text of text or textblock object – it shows that text until changed
- Rendering is automatic after VSYNC wakeup idle

Queue deferred render

- All objects retain state
  - Show object once – it stays visible UNTIL hidden
  - Set a color of an object – it stays that color until it is changed
  - Set text of text or textblock object – it shows that text until changed
- Rendering is automatic after VSYNC wakeup idle
  - Tries to only re-render updated regions / objects

Queue deferred render

- All objects retain state
  - Show object once – it stays visible UNTIL hidden
  - Set a color of an object – it stays that color until it is changed
  - Set text of text or textblock object – it shows that text until changed
- Rendering is automatic after VSYNC wakeup idle
  - Tries to only re-render updated regions / objects



Elementary Widgets

Elementary Widgets

- All widgets (elm widgets) are just smart objects

Elementary Widgets

- All widgets (elm widgets) are just smart objects
  - Provide extra behavior semantics

Elementary Widgets

- All widgets (elm widgets) are just smart objects
  - Provide extra behavior semantics
    - Focus, accessibility, packing, child deletion, ...

Elementary Widgets

- All widgets (elm widgets) are just smart objects
  - Provide extra behavior semantics
    - Focus, accessibility, packing, child deletion, ...
  - Uses Edje to define some internal layout/look and animation

Evas - OpenGL

Evas - OpenGL

- Use `elm_glview` widget for OpenGL

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed
  - Provides portability beyond OpenGL-ES based systems beyond Tizen

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed
  - Provides portability beyond OpenGL-ES based systems beyond Tizen
    - Works on Desktop OpenGL too (Linux, OSX, SDL)

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed
  - Provides portability beyond OpenGL-ES based systems beyond Tizen
    - Works on Desktop OpenGL too (Linux, OSX, SDL)
      - Gives you an OpenGL-ES 1.1/2.0 API to use across all targets

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed
  - Provides portability beyond OpenGL-ES based systems beyond Tizen
    - Works on Desktop OpenGL too (Linux, OSX, SDL)
      - Gives you an OpenGL-ES 1.1/2.0 API to use across all targets
  - You can have multiple GL Views in a window

Evas - OpenGL

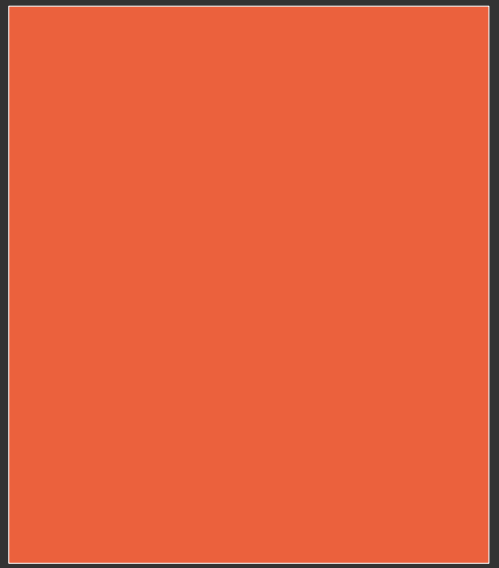
- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed
  - Provides portability beyond OpenGL-ES based systems beyond Tizen
    - Works on Desktop OpenGL too (Linux, OSX, SDL)
      - Gives you an OpenGL-ES 1.1/2.0 API to use across all targets
  - You can have multiple GL Views in a window
    - Even inside lists, scrollers

Evas - OpenGL

- Use `elm_glview` widget for OpenGL
  - Handles abstraction details of lower level Evas GL APIs
  - Handles correctly switching from direct (zero-copy) or indirect rendering
  - Handles calling your render function when rendering is needed
  - Provides portability beyond OpenGL-ES based systems beyond Tizen
    - Works on Desktop OpenGL too (Linux, OSX, SDL)
      - Gives you an OpenGL-ES 1.1/2.0 API to use across all targets
  - You can have multiple GL Views in a window
    - Even inside lists, scrollers
    - Mixed with other standard widgets and objects that can be used for game HUD

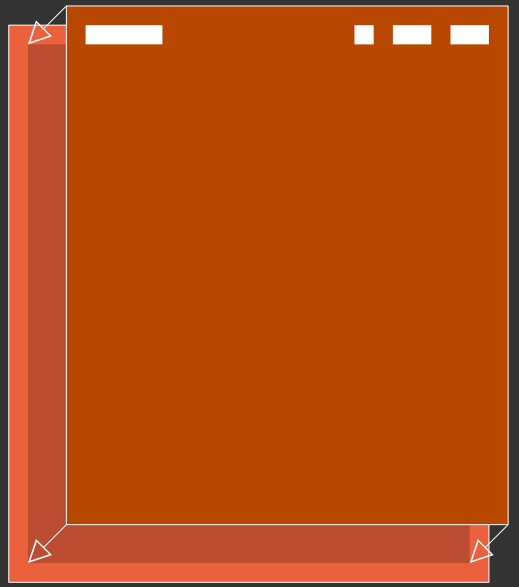
Containers & Scaling

- EFL loves containers



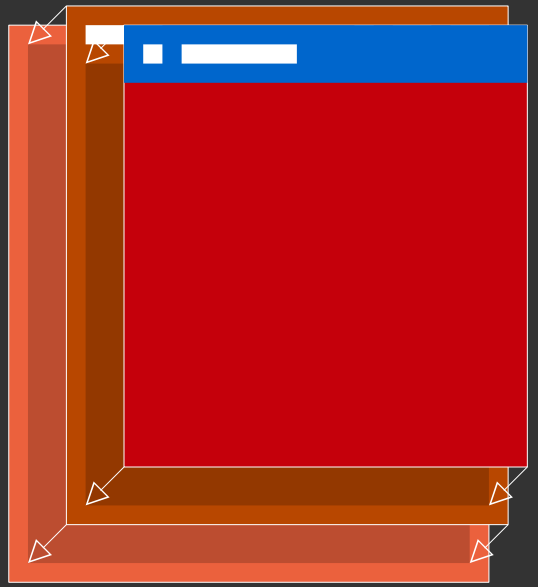
Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window



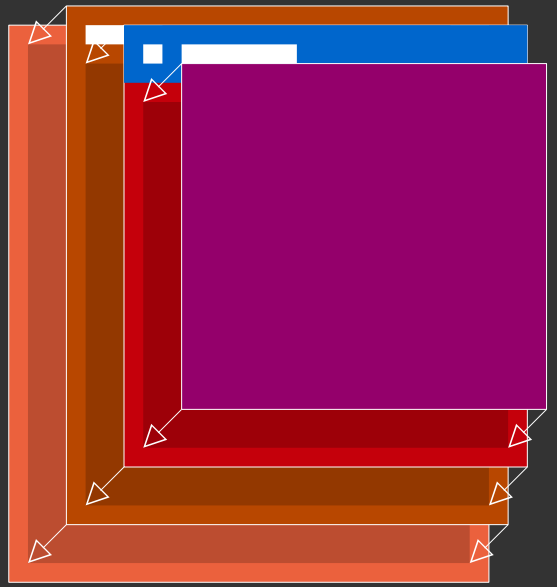
Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant



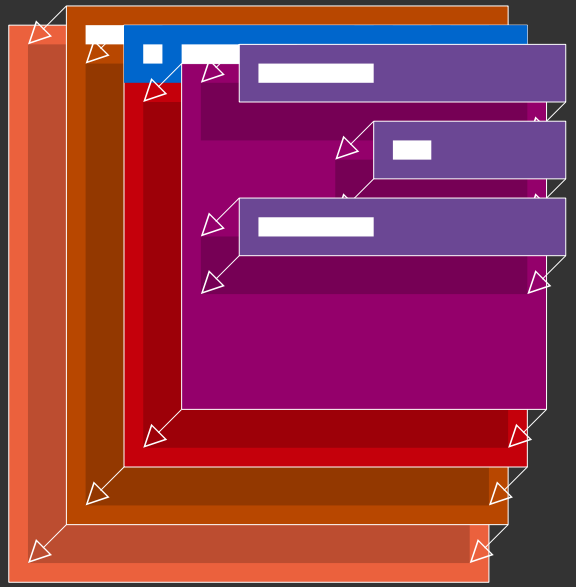
Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant
  - You add a Naviframe page
  - You put a Table in the Naviframe page



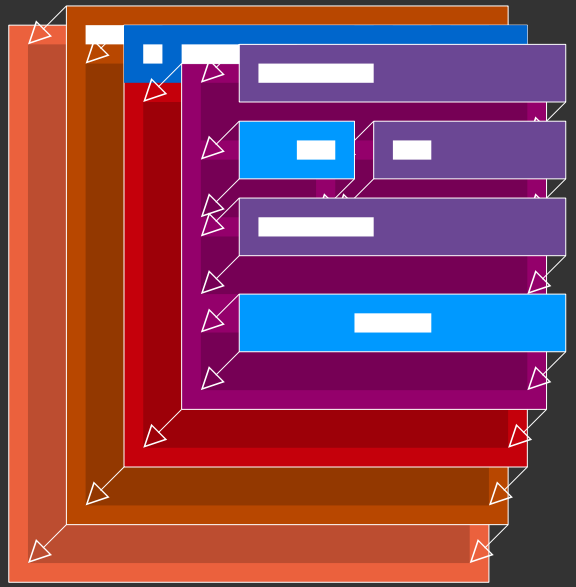
Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant
  - You add a Naviframe page
  - You put a Table in the Naviframe page
  - You put Entries in the Table



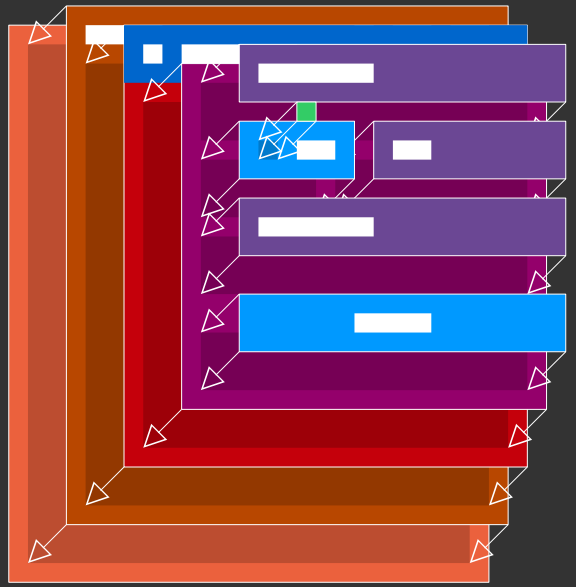
Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant
  - You add a Naviframe page
  - You put a Table in the Naviframe page
  - You put Entries in the Table
  - You put Buttons in the Table



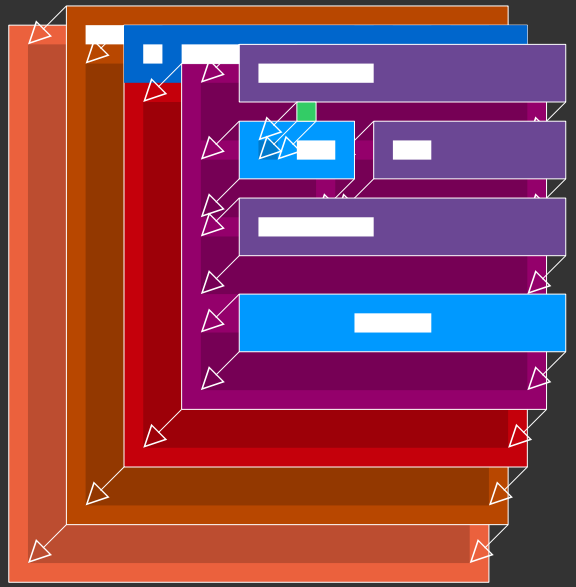
Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant
  - You add a Naviframe page
  - You put a Table in the Naviframe page
  - You put Entries in the Table
  - You put Buttons in the Table
  - You put and Icon in the Button



Containers & Scaling

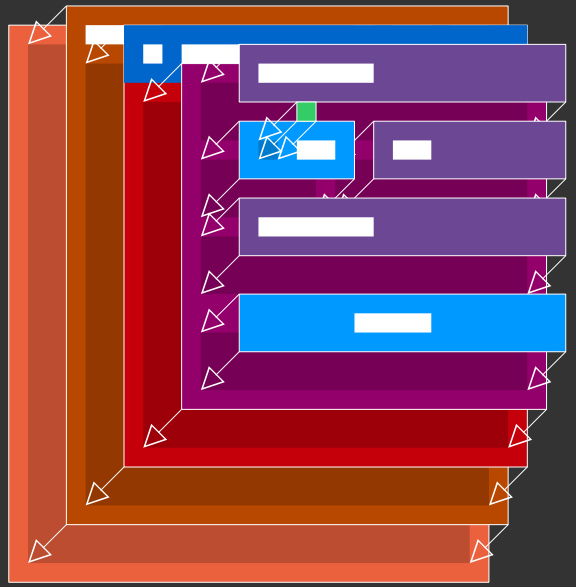
- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant
  - You add a Naviframe page
  - You put a Table in the Naviframe page
  - You put Entries in the Table
  - You put Buttons in the Table
  - You put and Icon in the Button



- If you use containers correctly, your UI can scale AND resize properly

Containers & Scaling

- EFL loves containers
  - You put a Conformant in a Window
  - You put a Naviframe in a Conformant
  - You add a Naviframe page
  - You put a Table in the Naviframe page
  - You put Entries in the Table
  - You put Buttons in the Table
  - You put and Icon in the Button



- If you use containers correctly, your UI can scale AND resize properly
  - This is like HTML with <DIV> in a <DIV> in a <TABLE> in a ...

Size Hinting

Defaults

Weight	0.0	0.0
Align	0.5	0.5

Weight	1.0	0.0
Align	0.5	0.5

Weight	1.0	1.0
Align	0.5	0.5

Weight	1.0	1.0
Align	1.0	-1

-1 = FILL

Weight	1.0	1.0
Align	0.0	1.0

Size Hinting

- Parent Widget decides how to arrange children

Size Hinting

- Parent Widget decides how to arrange children
  - Different parents have different rules

Size Hinting

- Parent Widget decides how to arrange children
  - Different parents have different rules
- Object hints determine if a child fills an/or expands its allocated area

Size Hinting

- Parent Widget decides how to arrange children
  - Different parents have different rules
- Object hints determine if a child fills an/or expands its allocated area
  - Align and Weight do this

Size Hinting

- Parent Widget decides how to arrange children
  - Different parents have different rules
- Object hints determine if a child fills an/or expands its allocated area
  - Align and Weight do this
  - Some widgets ONLY use Weight

Size Hinting

- Parent Widget decides how to arrange children
  - Different parents have different rules
- Object hints determine if a child fills an/or expands its allocated area
  - Align and Weight do this
  - Some widgets ONLY use Weight
  - Objects do NOT Fill and do NOT Expand by default

Size Hinting

- Window, Conformant, Part content (Layouts and other widgets)
  - Use only Weight

Size Hinting

- Window, Conformant, Part content (Layouts and other widgets)
  - Use only Weight
- Everything else uses both Weight and Align

Size Hinting

- Window, Conformant, Part content (Layouts and other widgets)
  - Use only Weight
- Everything else uses both Weight and Align
- All elm widgets control their own min size

Size Hinting

- Window, Conformant, Part content (Layouts and other widgets)
  - Use only Weight
- Everything else uses both Weight and Align
- All elm widgets control their own min size EXCEPT
  - Grids never calculate their own min size – you may control it

Size Hinting

- Window, Conformant, Part content (Layouts and other widgets)
  - Use only Weight
- Everything else uses both Weight and Align
- All elm widgets control their own min size EXCEPT
  - Grids never calculate their own min size – you may control it
  - GLview never calculates its own size

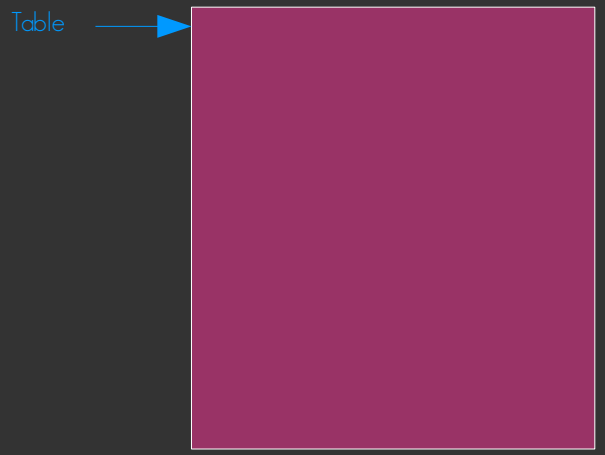
Size Hinting

- Window, Conformant, Part content (Layouts and other widgets)
  - Use only Weight
- Everything else uses both Weight and Align
- All elm widgets control their own min size EXCEPT
  - Grids never calculate their own min size – you may control it
  - GLview never calculates its own size
- Never set min (or max) size if already controlled by object

Min Size Control Trick

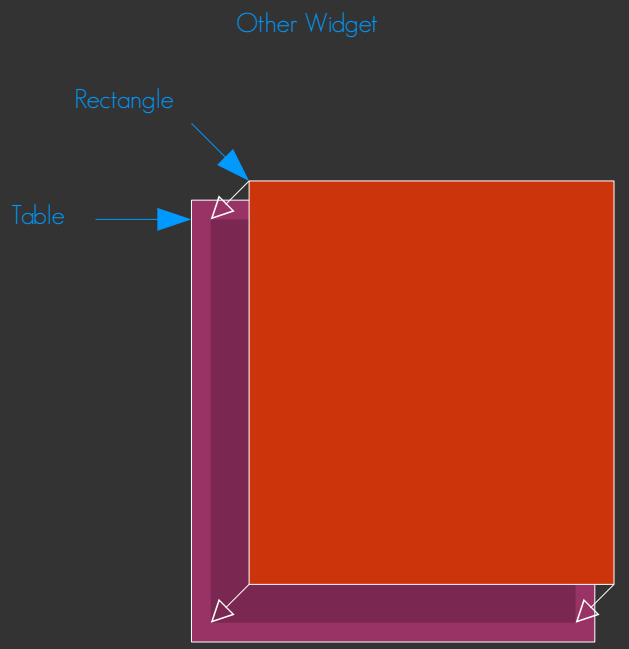
Min Size Control Trick

- Make Elm Table



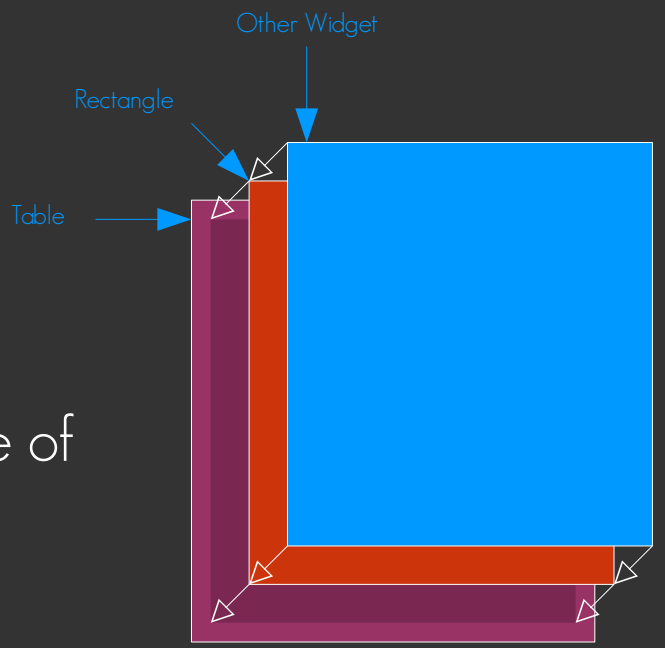
Min Size Control Trick

- Make Elm Table
- Make Evas Rectangle (do not show it)
- Pack Rectangle at 0, 0, 1x1



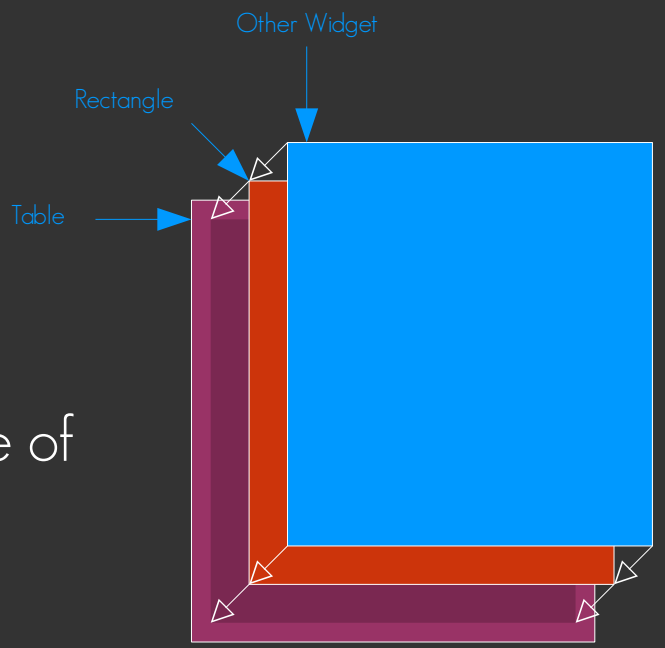
Min Size Control Trick

- Make Elm Table
- Make Evas Rectangle (do not show it)
- Pack Rectangle at 0, 0, 1x1
- Create other widget you want to control min size of
- Pack other widget in same Table at 0, 0, 1x1

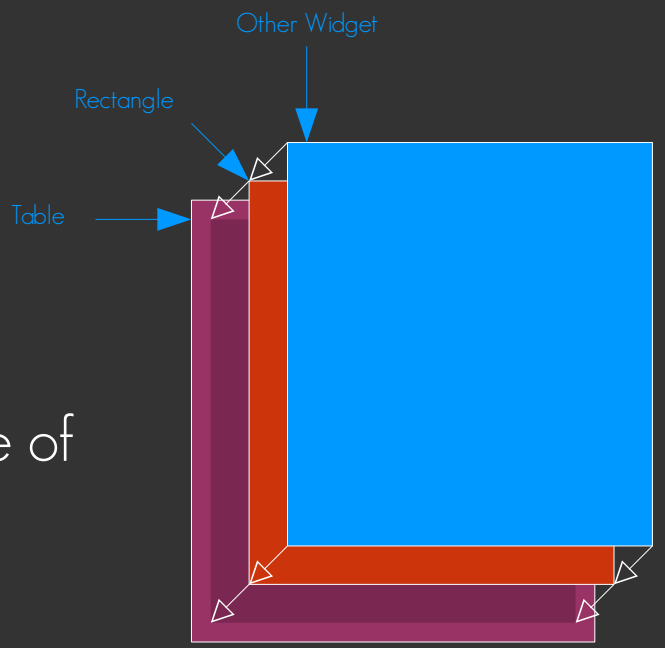


Min Size Control Trick

- Make Elm Table
- Make Evas Rectangle (do not show it)
- Pack Rectangle at 0, 0, 1x1
- Create other widget you want to control min size of
- Pack other widget in same Table at 0, 0, 1x1
- Set min size desired on Rectangle



Min Size Control Trick



- Make Elm Table
- Make Evas Rectangle (do not show it)
- Pack Rectangle at 0, 0, 1x1
- Create other widget you want to control min size of
- Pack other widget in same Table at 0, 0, 1x1
- Set min size desired on Rectangle
- This gives a second control point (Rectangle object) to set hints on.

Manual Positioning

Manual Positioning

- You can manually move/resize widgets and objects, BUT...

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.
    - This dramatically increases YOUR workload

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.
    - This dramatically increases YOUR workload
  - Should only be needed for very specific cases when special results are needed

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.
    - This dramatically increases YOUR workload
  - Should only be needed for very specific cases when special results are needed
  - Creates issues for Accessibility and Focus Movement

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.
    - This dramatically increases YOUR workload
  - Should only be needed for very specific cases when special results are needed
  - Creates issues for Accessibility and Focus Movement
    - You may have to manage a Focus chain by hand if you do this

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.
    - This dramatically increases YOUR workload
  - Should only be needed for very specific cases when special results are needed
  - Creates issues for Accessibility and Focus Movement
    - You may have to manage a Focus chain by hand if you do this
- This can be useful for effects or unusual UIs

Manual Positioning

- You can manually move/resize widgets and objects, BUT...
  - Increases the work to handle multiple orientations, resolutions etc.
  - This should be RARE
  - If you do, YOU are in charge of layout handling, scaling, window resizes etc.
    - This dramatically increases YOUR workload
  - Should only be needed for very specific cases when special results are needed
  - Creates issues for Accessibility and Focus Movement
    - You may have to manage a Focus chain by hand if you do this
- This can be useful for effects or unusual UIs
- Not recommended unless you enjoy ... The way of PAIN

Evas/Elm Object Lifecycle

Evas/Elm Object Lifecycle

- After objects are created (added)

Evas/Elm Object Lifecycle

- After objects are created (added)
  - Can listen for deletion:

```
void func_to_call_on_del(void *data, Evas *e, Evas_Object *obj, void *info) {
  printf("object %p deleted, data is %p\n", obj, data);
}
evas_object_vevent_callback_add(obj, EVAS_CALLBACK_DEL, func_to_call_on_del, data_pointer_for_func);
```

Evas/Elm Object Lifecycle

- After objects are created (added)
  - Can listen for deletion:

```
void func_to_call_on_del(void *data, Evas *e, Evas_Object *obj, void *info) {
  printf("object %p deleted, data is %p\n", obj, data);
}
evas_object_vevent_callback_add(obj, EVAS_CALLBACK_DEL, func_to_call_on_del, data_pointer_for_func);
```

- You can attach key → pointer values to any object

```
evas_object_data_set(obj, "mykey", mypointer);
evas_object_data_del(obj, "mykey");
mypointer = evas_object_data_get(obj, "mykey");
```

Evas/Elm Object Lifecycle

- After objects are created (added)
  - Can listen for deletion:

```
void func_to_call_on_del(void *data, Evas *e, Evas_Object *obj, void *info) {
  printf("object %p deleted, data is %p\n", obj, data);
}
evas_object_vevent_callback_add(obj, EVAS_CALLBACK_DEL, func_to_call_on_del, data_pointer_for_func);
```

- You can attach key → pointer values to any object

```
evas_object_data_set(obj, "mykey", mypointer);
evas_object_data_del(obj, "mykey");
mypointer = evas_object_data_get(obj, "mykey");
```

- Can be used for extending an object - Poor-mans-objects

Evas/Elm Object Lifecycle

- After objects are created (added)
  - Can listen for deletion:

```
void func_to_call_on_del(void *data, Evas *e, Evas_Object *obj, void *info) {
  printf("object %p deleted, data is %p\n", obj, data);
}
evas_object_vevent_callback_add(obj, EVAS_CALLBACK_DEL, func_to_call_on_del, data_pointer_for_func);
```

- You can attach key → pointer values to any object

```
evas_object_data_set(obj, "mykey", mypointer);
evas_object_data_del(obj, "mykey");
mypointer = evas_object_data_get(obj, "mykey");
```

- Can be used for extending an object - Poor-mans-objects
  - Set/get extra data to store/access

Evas/Elm Object Lifecycle

- After objects are created (added)
  - Can listen for deletion:

```
void func_to_call_on_del(void *data, Evas *e, Evas_Object *obj, void *info) {
  printf("object %p deleted, data is %p\n", obj, data);
}
evas_object_vevent_callback_add(obj, EVAS_CALLBACK_DEL, func_to_call_on_del, data_pointer_for_func);
```

- You can attach key → pointer values to any object

```
evas_object_data_set(obj, "mykey", mypointer);
evas_object_data_del(obj, "mykey");
mypointer = evas_object_data_get(obj, "mykey");
```

- Can be used for extending an object - Poor-mans-objects
  - Set/get extra data to store/access
  - On delete, delete any data needing deletion

Other Events

- Can also listen to many other events on objects:

Other Events

- Can also listen to many other events on objects:

…

```
EVAS_CALLBACK_DEL
EVAS_CALLBACK_SHOW
EVAS_CALLBACK_HIDE
EVAS_CALLBACK_MOVE
EVAS_CALLBACK_RESIZE
EVAS_CALLBACK_RESTACK
EVAS_CALLBACK_CHANGED_SIZE_HINTS
EVAS_CALLBACK_IMAGE_PRELOADED
```

…

Dynamic Image Content

Dynamic Image Content

- You can modify pixel data of an Image

Dynamic Image Content

- You can modify pixel data of an Image
  - Suggested always first set alpha, set size, then handle updates after that:

```
evas_object_image_alpha_set(obj, EINA_TRUE);
evas_object_image_size_set(obj, 600, 400);
```

Dynamic Image Content

- You can modify pixel data of an Image
  - Suggested always first set alpha, set size, then handle updates after that:

```
evas_object_image_alpha_set(obj, EINA_TRUE);
evas_object_image_size_set(obj, 600, 400);
```

  - Each update after that, get data, modify, set data, add update:

```
unsigned int *pixels = evas_object_image_data_get(obj, EINA_TRUE);
int stride = evas_obj_image_stride_get(obj);
// modify “pixels” with each row of pixels being “stride” bytes
evas_object_image_data_set(obj, pixels);
evas_object_image_update_add(obj, 0, 0, 600, 400);
```

Dynamic Image Content

- You can modify pixel data of an Image
  - Suggested always first set alpha, set size, then handle updates after that:

```
evas_object_image_alpha_set(obj, EINA_TRUE);
evas_object_image_size_set(obj, 600, 400);
```

  - Each update after that, get data, modify, set data, add update:

```
unsigned int *pixels = evas_object_image_data_get(obj, EINA_TRUE);
int stride = evas_obj_image_stride_get(obj);
// modify “pixels” with each row of pixels being “stride” bytes
evas_object_image_data_set(obj, pixels);
evas_object_image_update_add(obj, 0, 0, 600, 400);
```

    - ALWAYS GET THEN SET. Not doing so will lead to bugs

Dynamic Image Content

- You can modify pixel data of an Image
  - Suggested always first set alpha, set size, then handle updates after that:

```
evas_object_image_alpha_set(obj, EINA_TRUE);
evas_object_image_size_set(obj, 600, 400);
```

  - Each update after that, get data, modify, set data, add update:

```
unsigned int *pixels = evas_object_image_data_get(obj, EINA_TRUE);
int stride = evas_obj_image_stride_get(obj);
// modify "pixels" with each row of pixels being "stride" bytes
evas_object_image_data_set(obj, pixels);
evas_object_image_update_add(obj, 0, 0, 600, 400);
```

    - ALWAYS GET THEN SET. Not doing so will lead to bugs
  - If you modify pixel data OFTEN then you should do this before setting size above

Dynamic Image Content

- You can modify pixel data of an Image
  - Suggested always first set alpha, set size, then handle updates after that:

```
evas_object_image_alpha_set(obj, EINA_TRUE);
evas_object_image_size_set(obj, 600, 400);
```

  - Each update after that, get data, modify, set data, add update:

```
unsigned int *pixels = evas_object_image_data_get(obj, EINA_TRUE);
int stride = evas_obj_image_stride_get(obj);
// modify "pixels" with each row of pixels being "stride" bytes
evas_object_image_data_set(obj, pixels);
evas_object_image_update_add(obj, 0, 0, 600, 400);
```

    - ALWAYS GET THEN SET. Not doing so will lead to bugs
  - If you modify pixel data OFTEN then you should do this before setting size above
    - `evas_obj_image_content_hint_set(obj, EVAS_IMAGE_CONTENT_HINT_DYNAMIC)`

Dynamic Image Content

- You can modify pixel data of an Image
  - Suggested always first set alpha, set size, then handle updates after that:

```
evas_object_image_alpha_set(obj, EINA_TRUE);
evas_object_image_size_set(obj, 600, 400);
```

  - Each update after that, get data, modify, set data, add update:

```
unsigned int *pixels = evas_object_image_data_get(obj, EINA_TRUE);
int stride = evas_obj_image_stride_get(obj);
// modify "pixels" with each row of pixels being "stride" bytes
evas_object_image_data_set(obj, pixels);
evas_object_image_update_add(obj, 0, 0, 600, 400);
```

    - ALWAYS GET THEN SET. Not doing so will lead to bugs
  - If you modify pixel data OFTEN then you should do this before setting size above
    - `evas_obj_image_content_hint_set(obj, EVAS_IMAGE_CONTENT_HINT_DYNAMIC)`
    - This will try and use zero-copy for textures with OpenGL, SW is always zero-copy

Dynamic Image Content

- Pixels by default are ARGB8888 premultiplied Alpha

Dynamic Image Content

- Pixels by default are ARGB8888 premultiplied Alpha
  - Alpha as MSB, Blue LSB per 32bit “integer” word

```
A = (pixel >> 24) & 0xff
R = (pixel >> 16) & 0xff
G = (pixel >> 8) & 0xff
B = pixel & 0xff
```

Dynamic Image Content

- Pixels by default are ARGB8888 premultiplied Alpha
  - Alpha as MSB, Blue LSB per 32bit "integer" word

```
A = (pixel >> 24) & 0xff
R = (pixel >> 16) & 0xff
G = (pixel >> 8) & 0xff
B = pixel & 0xff
```

  - Premultiplied Alpha is where R, G and B are multipled by A

```
R = R * A; // conceptually if RGBA are 0.0 to 1.0
G = G * A; // conceptually if RGBA are 0.0 to 1.0
B = B * A; // conceptually if RGBA are 0.0 to 1.0
```

Dynamic Image Content

- Pixels by default are ARGB8888 premultiplied Alpha
  - Alpha as MSB, Blue LSB per 32bit "integer" word

```
A = (pixel >> 24) & 0xff
R = (pixel >> 16) & 0xff
G = (pixel >> 8) & 0xff
B = pixel & 0xff
```

  - Premultiplied Alpha is where R, G and B are multipled by A

```
R = R * A; // conceptually if RGBA are 0.0 to 1.0
G = G * A; // conceptually if RGBA are 0.0 to 1.0
B = B * A; // conceptually if RGBA are 0.0 to 1.0
```

  - Premultiplied Alpha produces correct results when scaling unlike non-premul

Dynamic Image Content

- Pixels by default are ARGB8888 premultiplied Alpha
  - Alpha as MSB, Blue LSB per 32bit "integer" word

```
A = (pixel >> 24) & 0xff
R = (pixel >> 16) & 0xff
G = (pixel >> 8) & 0xff
B = pixel & 0xff
```

  - Premultiplied Alpha is where R, G and B are multipled by A

```
R = R * A; // conceptually if RGBA are 0.0 to 1.0
G = G * A; // conceptually if RGBA are 0.0 to 1.0
B = B * A; // conceptually if RGBA are 0.0 to 1.0
```

  - Premultiplied Alpha produces correct results when scaling unlike non-premul
  - Premultiplied Alpha is faster in software rendering

Dynamic Image Content

- Pixels by default are ARGB8888 premultiplied Alpha
  - Alpha as MSB, Blue LSB per 32bit "integer" word

```
A = (pixel >> 24) & 0xff
R = (pixel >> 16) & 0xff
G = (pixel >> 8) & 0xff
B = pixel & 0xff
```

  - Premultiplied Alpha is where R, G and B are multipled by A

```
R = R * A; // conceptually if RGBA are 0.0 to 1.0
G = G * A; // conceptually if RGBA are 0.0 to 1.0
B = B * A; // conceptually if RGBA are 0.0 to 1.0
```

  - Premultiplied Alpha produces correct results when scaling unlike non-premul
  - Premultiplied Alpha is faster in software rendering
  - Premultiplied Alpha always produces correct destination buffer Alpha

Object Costs

Object Costs

- More objects cost more overhead

Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render

Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render
  - All objects have to be walked to render them (and clip them)

Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render
  - All objects have to be walked to render them (and clip them)
  - Objects cost memory

Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render
  - All objects have to be walked to render them (and clip them)
  - Objects cost memory
- So … keep number of objects down where possible

Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render
  - All objects have to be walked to render them (and clip them)
  - Objects cost memory
- So … keep number of objects down where possible
  - Widgets like Elm Genlist and Gengrid do this for you for all Items

Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render
  - All objects have to be walked to render them (and clip them)
  - Objects cost memory
- So … keep number of objects down where possible
  - Widgets like Elm Genlist and Gengrid do this for you for all Items
    - An Item is a very minimal piece of data acting as placeholder for an item

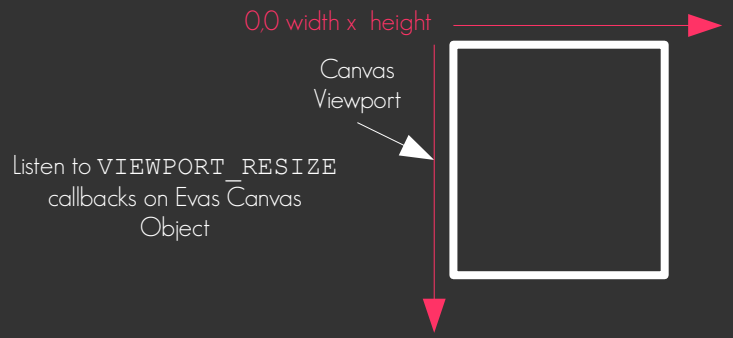
Object Costs

- More objects cost more overhead
  - All objects have to be walked to look for changes at render
  - All objects have to be walked to render them (and clip them)
  - Objects cost memory
- So … keep number of objects down where possible
  - Widgets like Elm Genlist and Gengrid do this for you for all Items
    - An Item is a very minimal piece of data acting as placeholder for an item
  - Do not abuse Genlist and Gengrid for generic scrollable UIs

“Infinite Scrolling” Trick

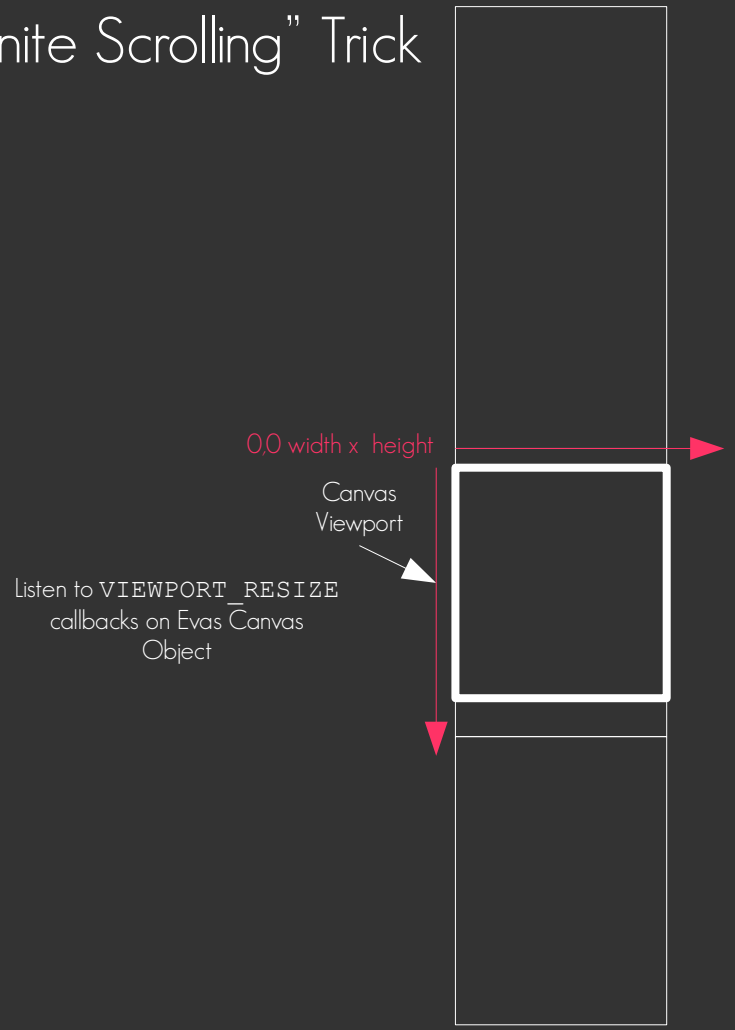
“Infinite Scrolling” Trick

- Add a Scroller



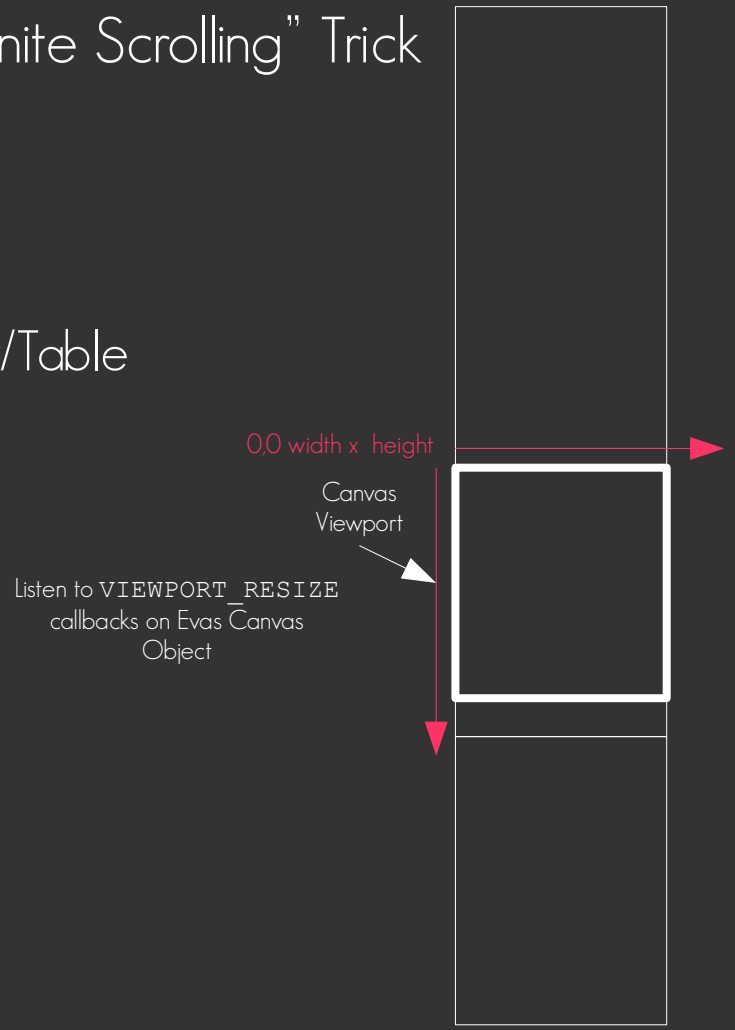
“Infinite Scrolling” Trick

- Add a Scroller
  - Add a Box or Table



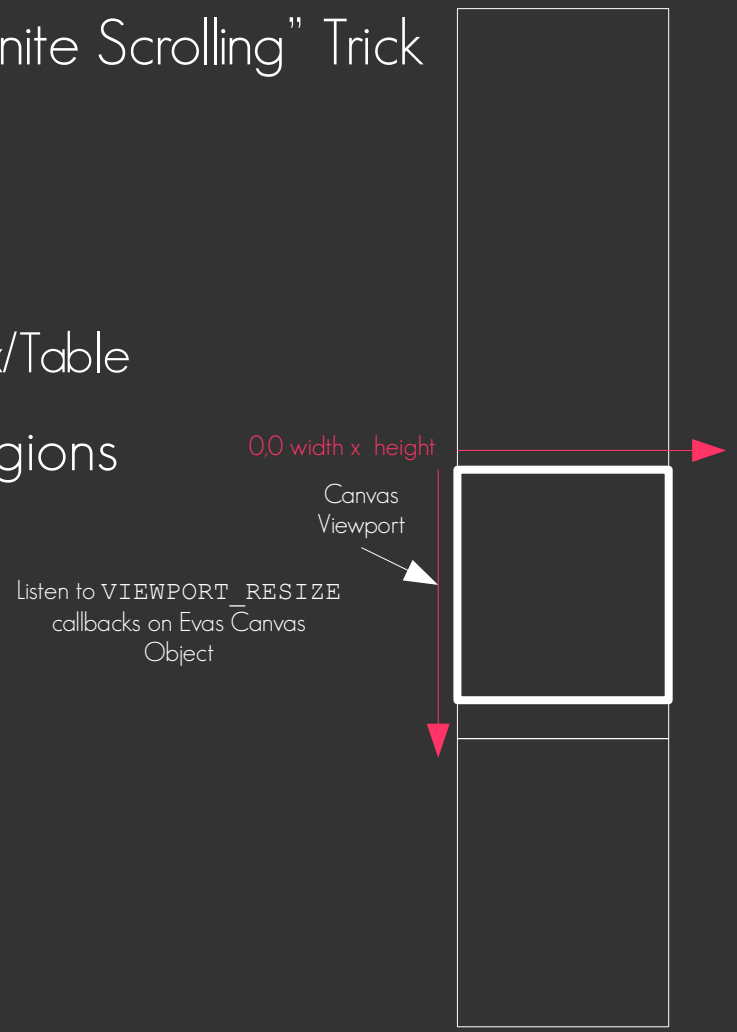
“Infinite Scrolling” Trick

- Add a Scroller
  - Add a Box or Table
  - Set content of Scroller to Box/Table



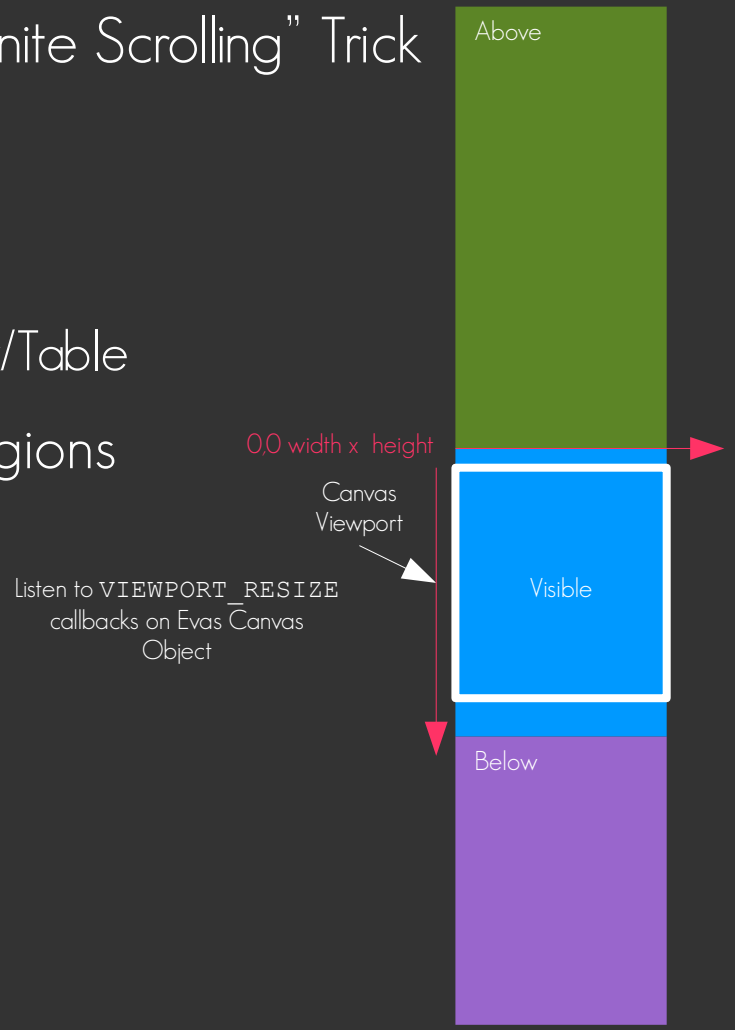
"Infinite Scrolling" Trick

- Add a Scroller
  - Add a Box or Table
  - Set content of Scroller to Box/Table
- Divide Table or Box into 3 regions



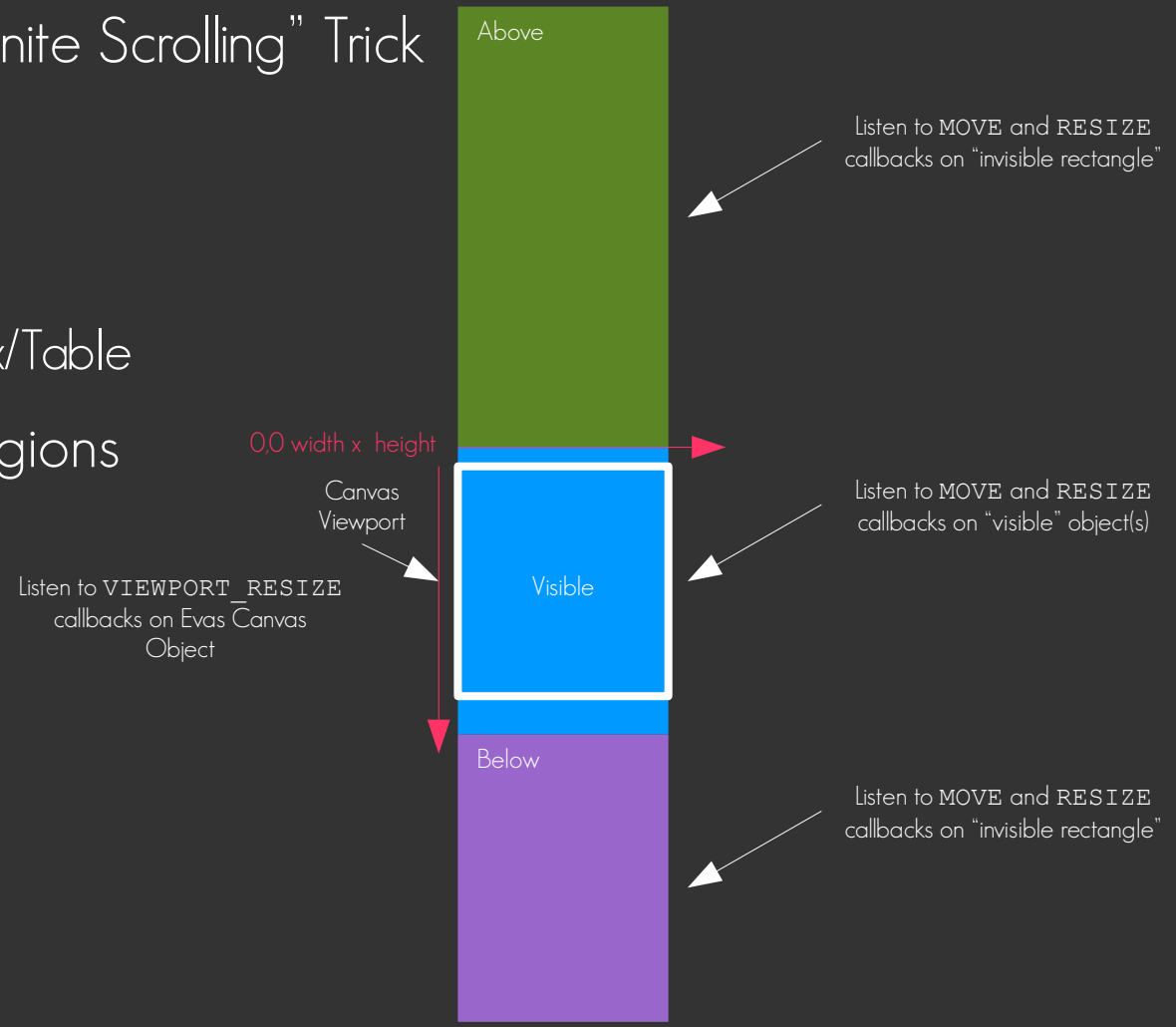
"Infinite Scrolling" Trick

- Add a Scroller
  - Add a Box or Table
  - Set content of Scroller to Box/Table
- Divide Table or Box into 3 regions
  - Above
  - Visible
  - Below



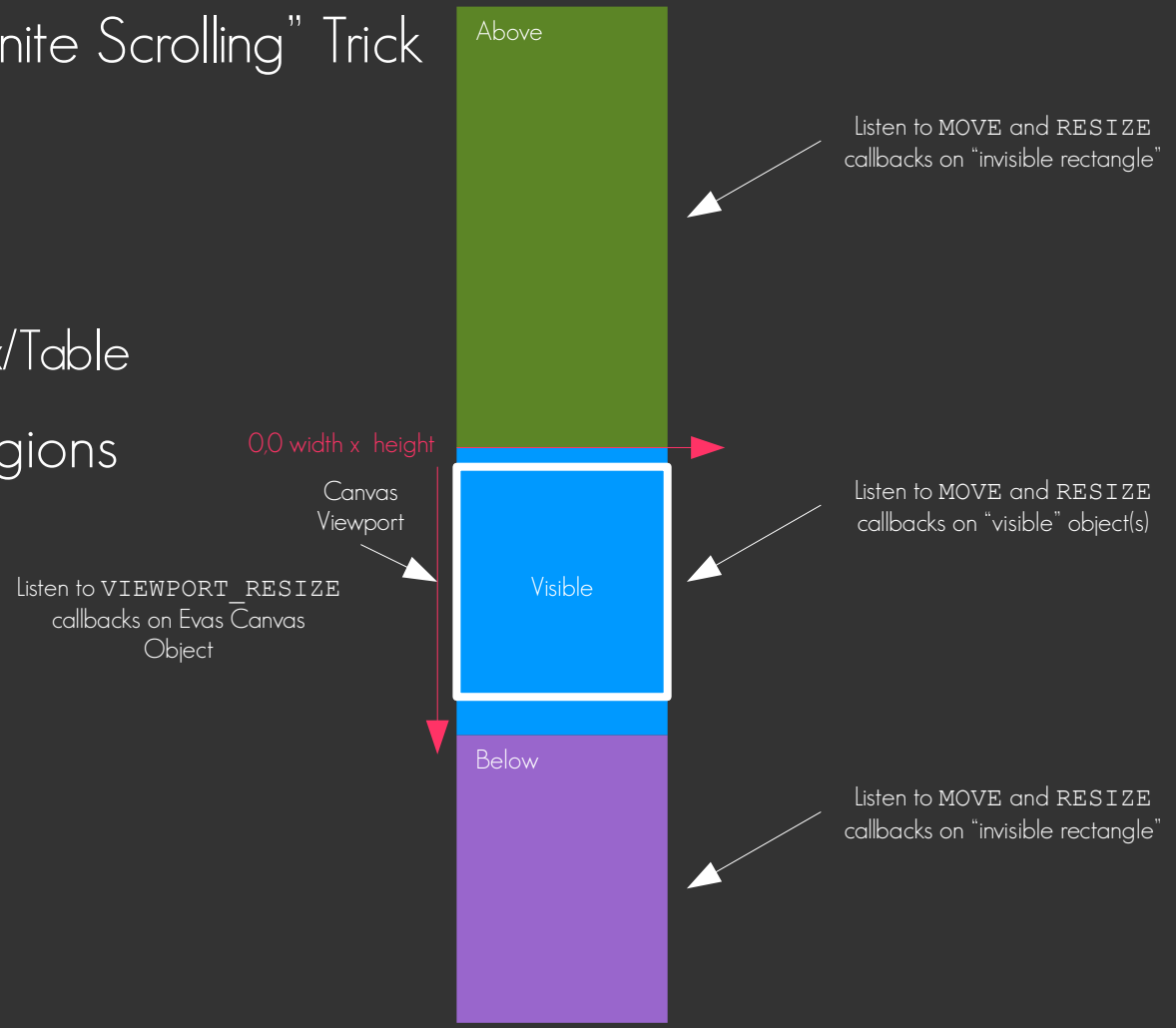
"Infinite Scrolling" Trick

- Add a Scroller
  - Add a Box or Table
  - Set content of Scroller to Box/Table
- Divide Table or Box into 3 regions
  - Above
  - Visible
  - Below
- Track geometry of "dummy rectangles" in Above/Below



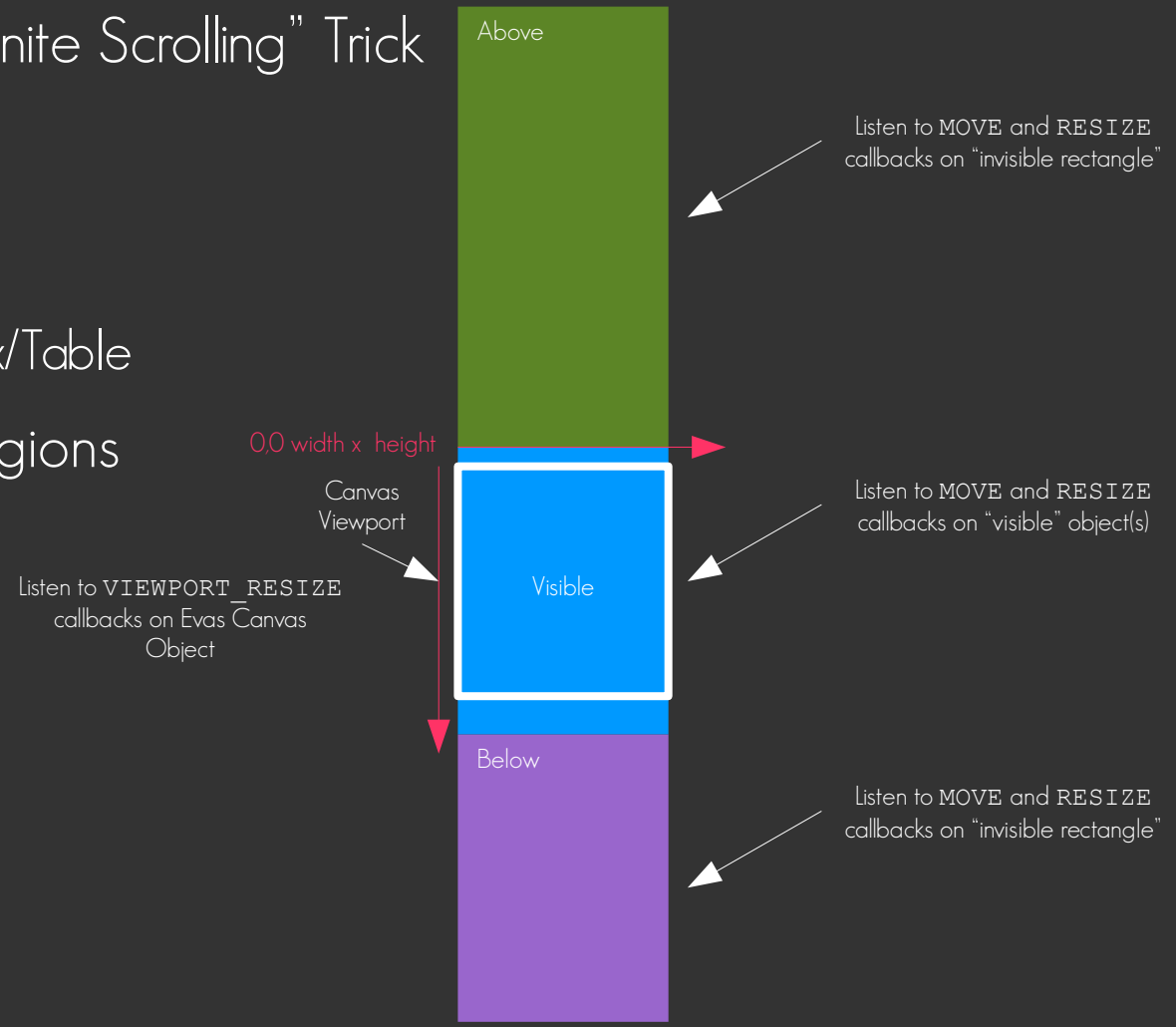
"Infinite Scrolling" Trick

- Add a Scroller
  - Add a Box or Table
  - Set content of Scroller to Box/Table
- Divide Table or Box into 3 regions
  - Above
  - Visible
  - Below
- Track geometry of "dummy rectangles" in Above/Below
  - Replace/insert as needed



"Infinite Scrolling" Trick

- Add a Scroller
  - Add a Box or Table
  - Set content of Scroller to Box/Table
- Divide Table or Box into 3 regions
  - Above
  - Visible
  - Below
- Track geometry of "dummy rectangles" in Above/Below
  - Replace/insert as needed
  - Adjust min size of above/below



“Infinite Scrolling” Trick

- The less content you create when the region becomes visible

“Infinite Scrolling” Trick

- The less content you create when the region becomes visible
  - The smoother things will be

“Infinite Scrolling” Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too

“Infinite Scrolling” Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes

“Infinite Scrolling” Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes
  - Windows in Tizen CAN RESIZE

“Infinite Scrolling” Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes
  - Windows in Tizen CAN RESIZE
    - It is a multi-windowed system without fixed sizes, just like a desktop UI

"Infinite Scrolling" Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes
  - Windows in Tizen CAN RESIZE
    - It is a multi-windowed system without fixed sizes, just like a desktop UI
- Consider loading content data (text labels, other I/O) in threads

"Infinite Scrolling" Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes
  - Windows in Tizen CAN RESIZE
    - It is a multi-windowed system without fixed sizes, just like a desktop UI
- Consider loading content data (text labels, other I/O) in threads
  - Queue/begin this thread worker when visiblity happens

"Infinite Scrolling" Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes
  - Windows in Tizen CAN RESIZE
    - It is a multi-windowed system without fixed sizes, just like a desktop UI
- Consider loading content data (text labels, other I/O) in threads
  - Queue/begin this thread worker when visiblity happens
  - Create widgets with "dummy" content (eg blank labels)

"Infinite Scrolling" Trick

- The less content you create when the region becomes visible
  - The smoother things will be
- Remember to delete content as it exits the viewport too
- Remember to track viewport changes
  - Windows in Tizen CAN RESIZE
    - It is a multi-windowed system without fixed sizes, just like a desktop UI
- Consider loading content data (text labels, other I/O) in threads
  - Queue/begin this thread worker when visiblity happens
  - Create widgets with "dummy" content (eg blank labels)
  - Fill text/content once thread is done

Widget Styles

Widget Styles

- All widgets support styles

Widget Styles

- All widgets support styles except
  - Box, Table, Grid

Widget Styles

- All widgets support styles except
  - Box, Table, Grid
- Styles allow custiomizing of widget look

Widget Styles

- All widgets support styles except
  - Box, Table, Grid
- Styles allow custiomizing of widget look
- You can make new custom styles of your own

Widget Styles

- All widgets support styles except
  - Box, Table, Grid
- Styles allow custiomizing of widget look
- You can make new custom styles of your own
- Themes are in EDJ Files (Edje collections of groups)

Widget Styles

- All widgets support styles except
  - Box, Table, Grid
- Styles allow custiomizing of widget look
- You can make new custom styles of your own
- Themes are in EDJ Files (Edje collections of groups)
- Custom styles are generally a bad idea because application will not fit in

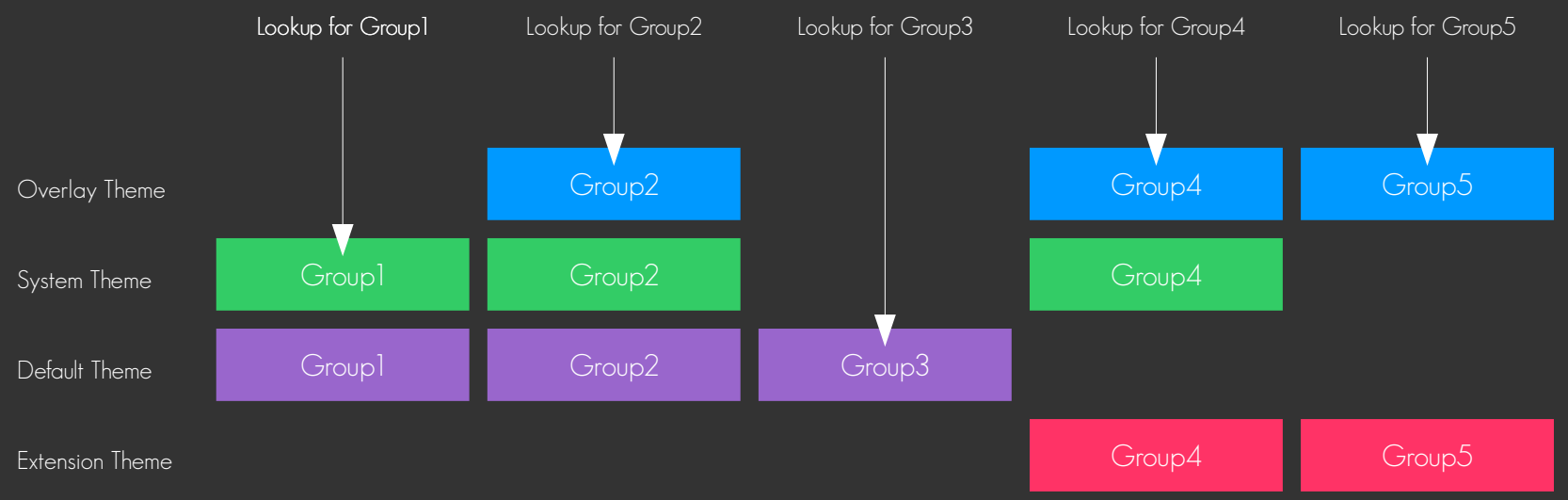
Widget Styles

- All widgets support styles except
  - Box, Table, Grid
- Styles allow custiomizing of widget look
- You can make new custom styles of your own
- Themes are in EDJ Files (Edje collections of groups)
- Custom styles are generally a bad idea because application will not fit in
- Use extensions instead of overlays

Widget Styles

- All widgets support styles except
  - Box, Table, Grid
- Styles allow custiomizing of widget look
- You can make new custom styles of your own
- Themes are in EDJ Files (Edje collections of groups)
- Custom styles are generally a bad idea because application will not fit in
- Use extensions instead of overlays
  - Add new styles instead of override

Theme Style Lookup



Edje - Graphical Blobs

Edje - Graphical Blobs

- What Edje is

Edje - Graphical Blobs

- What Edje is
  - Designed as an “overgrown PSD file”

Edje - Graphical Blobs

- What Edje is
  - Designed as an “overgrown PSD file”
    - Multiple layers

Edje - Graphical Blobs

- What Edje is
  - Designed as an “overgrown PSD file”
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles
    - The stored states of layers can be changed in response to signals (e.g from events or app)

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles
    - The stored states of layers can be changed in response to signals (e.g from events or app)
    - State changes can animate by interpolating between states

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles
    - The stored states of layers can be changed in response to signals (e.g from events or app)
    - State changes can animate by interpolating between states
- What Edje is NOT

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles
    - The stored states of layers can be changed in response to signals (e.g from events or app)
    - State changes can animate by interpolating between states
- What Edje is NOT
  - An application UI design system

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles
    - The stored states of layers can be changed in response to signals (e.g from events or app)
    - State changes can animate by interpolating between states
- What Edje is NOT
  - An application UI design system
  - A widget layout system

Edje - Graphical Blobs

- What Edje is
  - Designed as an "overgrown PSD file"
    - Multiple layers
    - Layers can move/resize/show/hide/change color & transparency
    - Layers can point to image data to use, be text or rectangles
    - The stored states of layers can be changed in response to signals (e.g from events or app)
    - State changes can animate by interpolating between states
- What Edje is NOT
  - An application UI design system
  - A widget layout system
  - A chicken

EDC

```
collections {
  images {
    image: "icon.png" LOSSY 80; // encode icon.png with lossy 80% quality
  }
  group { name: "my/group"; // name of this group int he collections
    parts {
      part { name: "base"; type: RECT; // a "base" part that is a rect
        description { state: "default" 0.0; // the default state
          color: 255 128 0 255; // orange
        }
        description { state: "active" 0.0; // an active state
          color: 255 255 0 255; // yellow
        }
      }
      part { name: "icon"; // icon part - default type is image
        description { state: "default" 0.0; // the default state
          rel2.relative: 0.5 1.0; // rel2 (bottom right) relative to middle
          image.normal: "icon.png";
        }
      }
      part { name: "label"; type: TEXT; // label part
        description { state: "default" 0.0; // the default state
          rel1.to: "icon"; // top-left relative to the icon
          rel1.relative: 1.0 0.0; // relative to top-right of icon
          color: 255 255 255 255; // white
          text.font: "Sans"; text.size: 10; // 10 px Sans font
          text.text: "Hello"; // text content to show
        }
        description { state: "active" 0.0; // an active state
          inherit: "default" 0.0; // copy state from default then modify
          color: 0 0 0 255; // black
          text.text: "Clicked"; // a new text label
        }
      }
    }
```

Layer

Layer

Layer

```
  programs {
    program { // when mouse button 1 is down on label
      signal: "mouse,down,1"; source: "label";
      action: STATE_SET "active" 0.0; // set state to active
      transition: SINUSOIDAL 0.5; // over 0.5 sec with sinusoidal interp
      target: "label"; // do it to label
      target: "base"; // do it to base
    }
    program { // when mouse button 1 is released on label
      signal: "mouse,up,1"; source: "label";
      action: STATE_SET "default" 0.0; // set state to default
      transition: SINUSOIDAL 1.0; // over 1 sec with sinusoidal interp
      target: "label"; // do it to label
      target: "base"; // do it to base
    }
  }
}
}
```

Edje - Graphical Blobs

- Edje has been extended over many years

Edje - Graphical Blobs

- Edje has been extended over many years
- Recently tools like Enventor (ships with Tizen 2.4 SDK) give you a GUI editing tool

Edje - Graphical Blobs

- Edje has been extended over many years
- Recently tools like Enventor (ships with Tizen 2.4 SDK) give you a GUI editing tool
  - Gives you rapid feedback on your changes to save a lot of time and effort

Edje - Graphical Blobs

- Edje has been extended over many years
- Recently tools like Enventor (ships with Tizen 2.4 SDK) give you a GUI editing tool
  - Gives you rapid feedback on your changes to save a lot of time and effort
  - Eflete is even better and is almost pure-GUI EDJE file editing

Edje - Graphical Blobs

- Edje has been extended over many years
- Recently tools like Enventor (ships with Tizen 2.4 SDK) give you a GUI editing tool
  - Gives you rapid feedback on your changes to save a lot of time and effort
  - Eflete is even better and is almost pure-GUI EDJE file editing
    - Coming in future - you can download source from git.enlightenment.org

Edje - Graphical Blobs

- Edje has been extended over many years
- Recently tools like Enventor (ships with Tizen 2.4 SDK) give you a GUI editing tool
  - Gives you rapid feedback on your changes to save a lot of time and effort
  - Eflete is even better and is almost pure-GUI EDJE file editing
    - Coming in future - you can download source from git.enlightenment.org
- EDC files go through CPP so you can use #include, #define, #ifdef etc.

EDC

```
#define IMAGE(x) images.image: x LOSSY 80
#define DEF “default” 0.0
#define ACT “active” 0.0
#define TRANS(x) SINUSOIDAL x
#define TARGETS target: “label”; target: “base”
#define SIGSRC(x, y) signal: x; source: x
collections {
  IMAGE(“icon.png”);
  group { name: "my/group";
    parts {
      part { name: "base"; type: RECT; // a "base" part that is a rect
        description { state: DEF;
          color: 255 128 0 255; // orange
        }
        description { state: ACT; // an active state
          color: 255 255 0 255; // yellow
        }
      }
      part { name: "icon"; // icon part - default type is image
        description { state: DEF;
          rel2.relative: 0.5 1.0; // rel2 (bottom right) relative to middle
          image.normal: "icon.png";
        }
      }
      part { name: "label"; type: TEXT; // label part
        description { state: "default" 0.0; // the default state
          rel1.to: "icon"; // top-left relative to the icon
          rel1.relative: 1.0 0.0; // relative to top-right of icon
          color: 255 255 255 255; // white
          text.font: "Sans"; text.size: 10; // 10 px Sans font
          text.text: "Hello"; // text content to show
        }
        description { state: ACT; // an active state
          inherit: "default" 0.0; // copy state from default then modify
          color: 0 0 0 255; // black
          text.text: "Clicked"; // a new text label
        }
      }
    }
```

```
    programs {
      program { SIGSRC(“mouse,down,1”, “label”);
        action: STATE_SET ACT; // set state to active
        transition: TRANS(0.5); // over 0.5 sec with sinusoidal interp
        TARGETS;
      }
      program { SIGSRC(“mouse,up,1”, “label”);
        action: STATE_SET DEF; // set state to default
        transition: TRANS(1.0);
        TARGETS;
      }
    }
  }
}
```

Last - 3D Effects

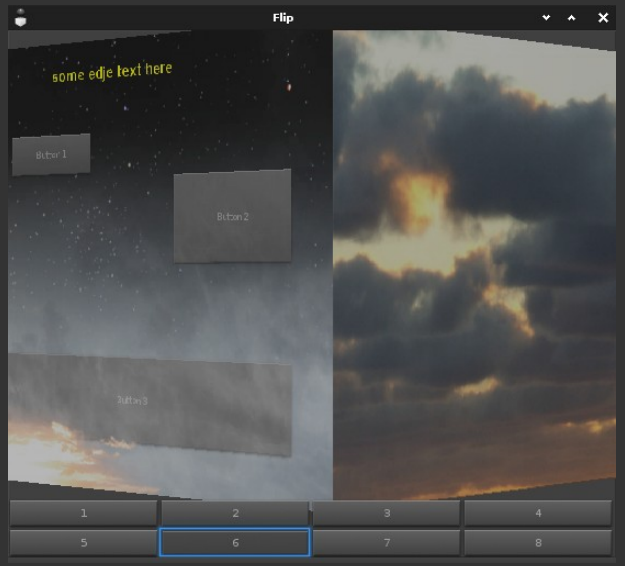
You don't need OpenGL

2.5D Mapping

- You can do 2.5D/3D effects on any object

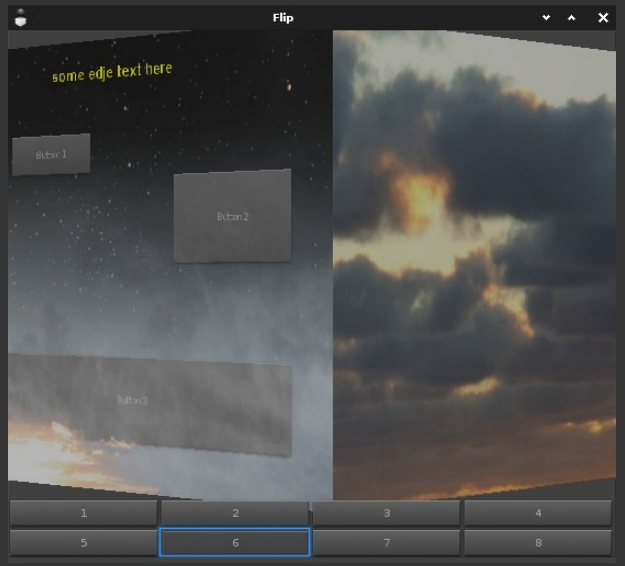
2.5D Mapping

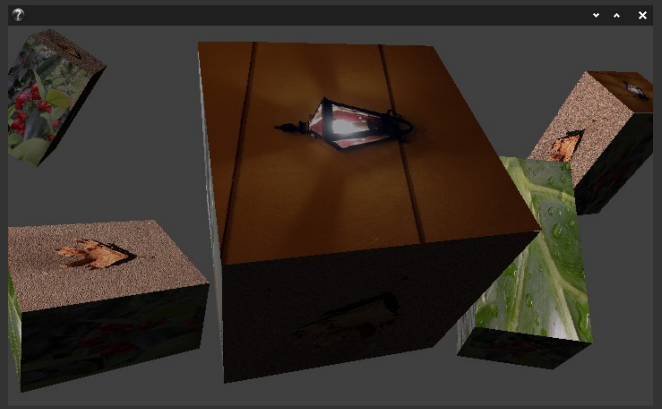
- You can do 2.5D/3D effects on any object
- With imagination you can do 3D simple 3D objects with multiple maps



2.5D Mapping

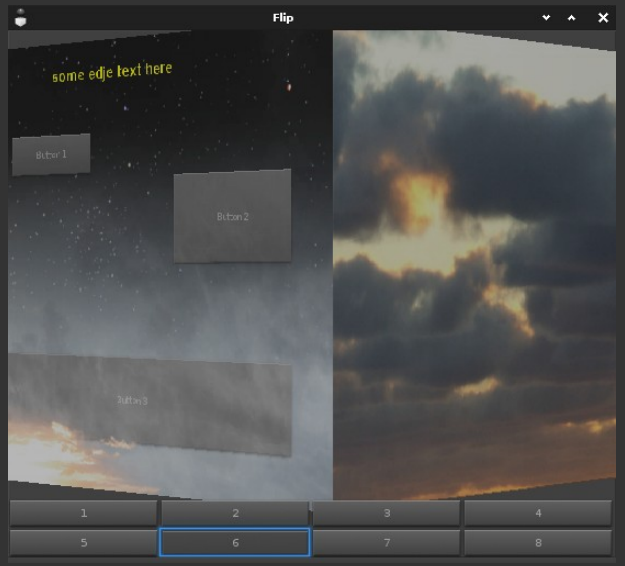
- You can do 2.5D/3D effects on any object
- With imagination you can do 3D simple 3D objects with multiple maps

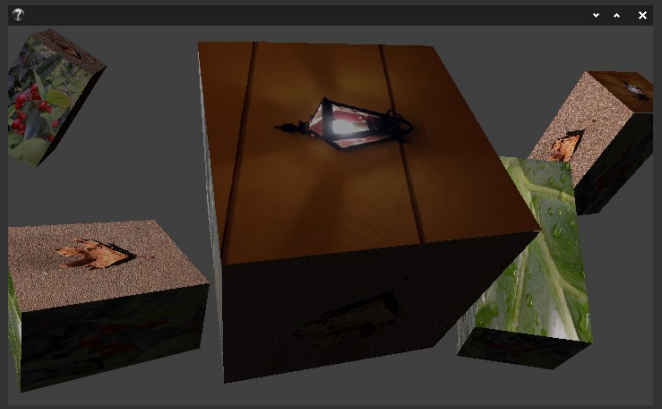


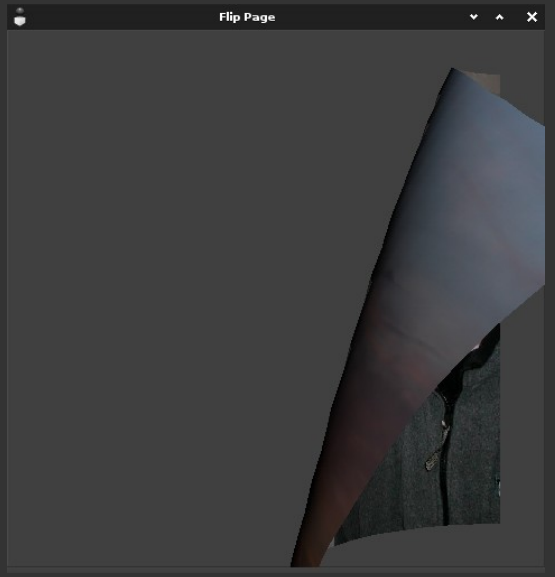


2.5D Mapping

- You can do 2.5D/3D effects on any object
- With imagination you can do 3D simple 3D objects with multiple maps







2.5D Mapping

```
Evas_Map *m = evas_map_new(4); // new 4 point map - only 4 points are actually guaranteed to work

evas_map_util_points_populate_from_object(m, obj); // fill map points from current object geometry

evas_map_util_3d_rotate(m, 30.0, 20.0, 10.0, center_x, center_y, 0); // rotate map around a center point

evas_map_util_3d_perspective(m, center_screen_x, center_screen_y, 0, pixels_from_camera_to_screen_plane);

evas_object_map_set(obj, m); // set map for this object

evas_map_free(m); // free it - object will retain a reference and release when it is done

evas_object_map_enable_set(obj, EINA_TRUE); // enable the map that is set
```

2.5D Mapping

```
Evas_Map *m = evas_map_new(4); // new 4 point map - only 4 points are actually guaranteed to work
evas_map_util_points_populate_from_object(m, obj); // fill map points from current object geometry
evas_map_util_3d_rotate(m, 30.0, 20.0, 10.0, center_x, center_y, 0); // rotate map around a center point
evas_map_util_3d_perspective(m, center_screen_x, center_screen_y, 0, pixels_from_camera_to_screen_plane);
evas_object_map_set(obj, m); // set map for this object
evas_map_free(m); // free it - object will retain a reference and release when it is done
evas_object_map_enable_set(obj, EINA_TRUE); // enable the map that is set
```

- Many more ways to use maps to “map an object somewhere else”

2.5D Mapping

```
Evas_Map *m = evas_map_new(4); // new 4 point map - only 4 points are actually guaranteed to work
evas_map_util_points_populate_from_object(m, obj); // fill map points from current object geometry
evas_map_util_3d_rotate(m, 30.0, 20.0, 10.0, center_x, center_y, 0); // rotate map around a center point
evas_map_util_3d_perspective(m, center_screen_x, center_screen_y, 0, pixels_from_camera_to_screen_plane);
evas_object_map_set(obj, m); // set map for this object
evas_map_free(m); // free it - object will retain a reference and release when it is done
evas_object_map_enable_set(obj, EINA_TRUE); // enable the map that is set
```

- Many more ways to use maps to "map an object somewhere else"
- Is useful even for 2D rotations as well

Remember...

Remember...

- If it's hard – you're probably doing it wrong

Remember...

- If it's hard – you're probably doing it wrong
- Use the facilities provided to save time

Remember...

- If it's hard – you're probably doing it wrong
- Use the facilities provided to save time
  - The higher level ones save more time

Remember...

- If it's hard – you're probably doing it wrong
- Use the facilities provided to save time
  - The higher level ones save more time
- Ask questions to learn and make better code

Remember...

- If it's hard – you're probably doing it wrong
- Use the facilities provided to save time
  - The higher level ones save more time
- Ask questions to learn and make better code
  - Real humans are here and happy to talk to you

Links

- http://developer.tizen.org
  - SDK, API Documentation and more
- http://www.enlightenment.org
  - Upstream EFL API development project
- http://git.enlightenment.org/core/elementary.git/tree/data/themes
  - Lots of sample EDC files for Edje (a complete theme within this tree of files)
- http://git.enlightenment.org/core/elementary.git/tree/src/bin
  - Lots of sample code showing how to use lots of Elm APIs
- http://developer.tizen.org/forums
  - Tizen developer forums for help and advice
- http://www.tizen.org/community/mailing-lists
  - Mailing lists for Tizen development
- http://www.tizen.org/community/irc
  - Internet Chat for Tizen
- https://www.enlightenment.org/contact
  - IRC and E-Mail information for EFL upstream development