EFL

Enlightenment Foundation Libraries
http://www.enlightenment.org

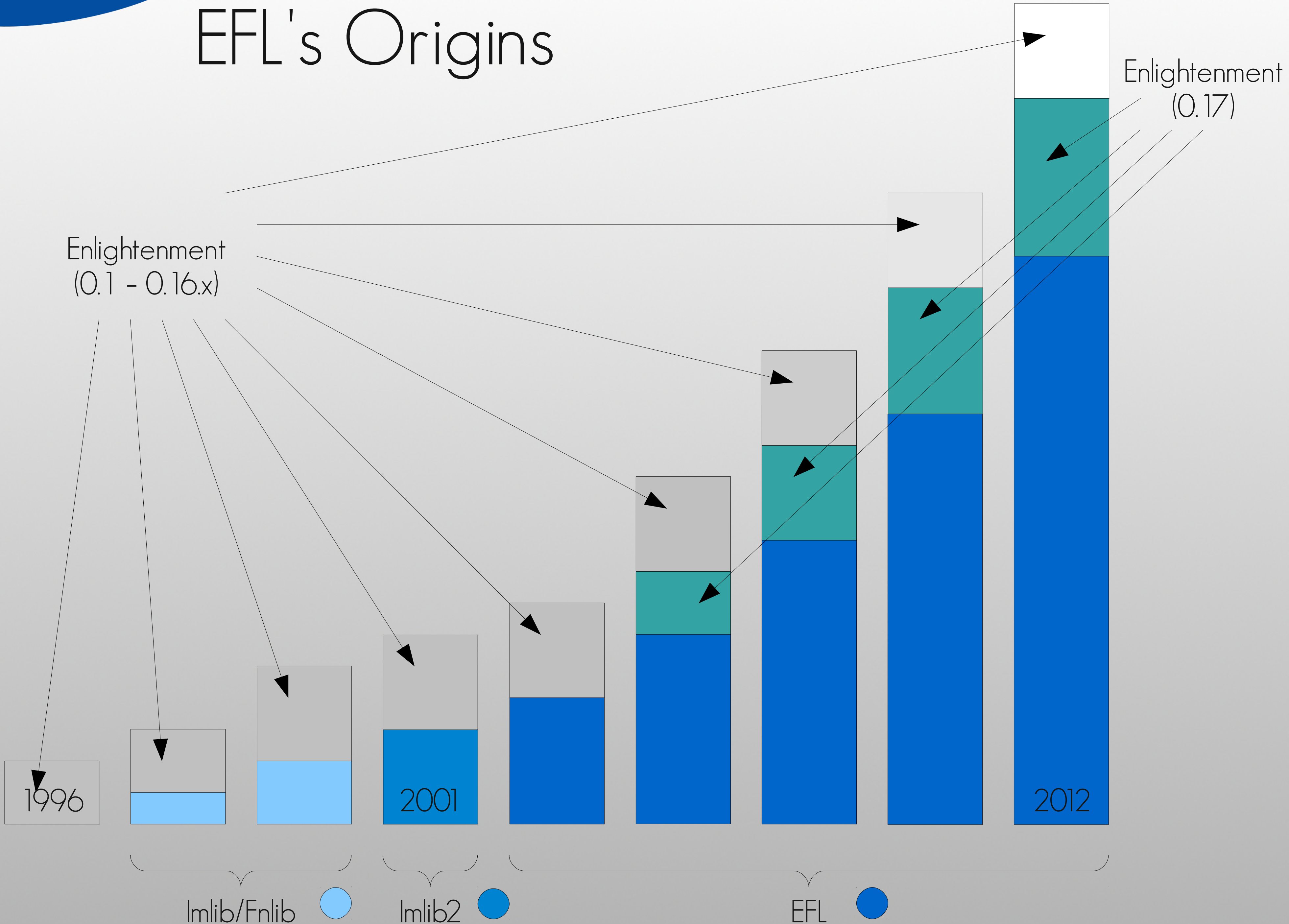
Tizen native display layer – Architecture & Usage

Carsten Haitzler
Enlightenment project lead & founder
Principal Engineer
raster@rasterman.com
c.haitzler@samsung.com

What is EFL?

- A collection of libraries
  - Built by the same team working on Enlightenment
  - Built for the purpose of making E17 (Enlightenment 0.17)
  - Always focused on staying lean and still providing fanciness
  - Almost all development focus sunk into EFL vs E17
  - Cover a wide range of functionality due to broad needs
- 26% of code for E17 is E, rest is EFL.
  - E17+EFL make up only 50% of code in SVN though

EFL's Origins



Historical Details

- 1996 – Enlightenment development started
- 1997 – Imaging layer split off into Imlib and Fnlib
- 1997 – Imlib adds GTK+/GDK support
- 1999 – Imlib2 combines images, fonts, alpha channels etc.
- 2001 – Evas (using Imlib2 and OpenGL) first appears
- And then EFL really began as more libs were added:
  - Ecore, Ebits (later replaced by Edje), Edb (deprecated in favor of Eet), Eina, Embryo, Efreet, EDbus, Ethumb, Emotion, Elementary, Epdf, Eeze.

What's inside

- Canvas scene-graph (Evas)
  - OpenGL, OpenGL-ES2.0, Software renderer and more
- Core mainloop, connection, input and glue libraries (Ecore)
- Data codec and storage (Eet)
- Bytecode VM (Embryo)
- Pre-made data objects with scripting, animation etc. (Edje)
- Freedesktop.org standards support (Efreet)

What's inside

- Data structure, modules and base (Eina)
- Dbus integration and wrapping (Edbus)
- Asynchronous I/O (Eio) (Currently not in Tizen)
- Video playback glue (Emotion)
- Udev hardware detection (Eeze) (Currently not in Tizen)
- Thumbnailer & cacher (Ethumb)
- Widgets & convenience (Elementary)

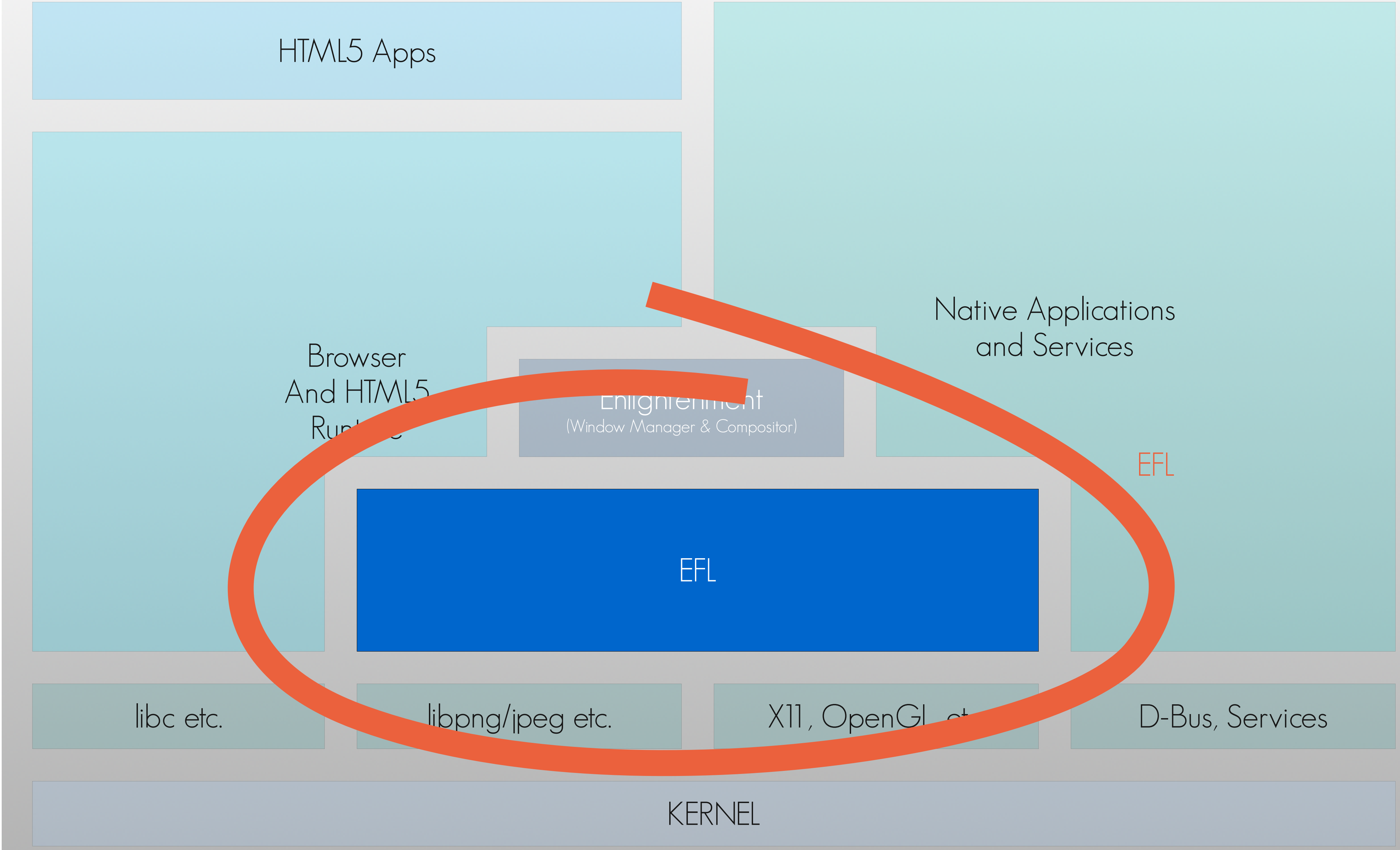
On its way

- Javascript runtime environment (Elev8) (Currently not in Tizen)
  - Still in alpha, not complete
  - Like Node.js, but for UI.
    - Both Node.js and Elev8 highly similar structures & design
    - Reaching out to Node.js developers (positive response)
    - Harmonizing Elev8 vs Node.js libvu
- Object model enhancements (Currently not in Tizen)
- Expanding performance, threading (Currently not in Tizen)
- Decreasing memory footprint & more (Currently not in Tizen)

So why does this matter?

- EFL is the core toolkit being used in Tizen
- EFL is built for performance and low footprint
  - Still heavy focus on customization and power
- Tizen is open, just like EFL
- Native apps can use EFL as opposed to shipping their own toolkits
  - Smaller footprint for shipping devices
  - Continued support
- Provides much of what you need
  - It's an open source project, so contributions always welcomed
- API's all in C, thus easily usable from both C and C++

Where does it lurk?



HTML5 Apps

Browser
And HTML5
Runtime

Native Applications
and Services

Enlightenment
(Window Manager & Compositor)

EFL

EFL

libc etc.

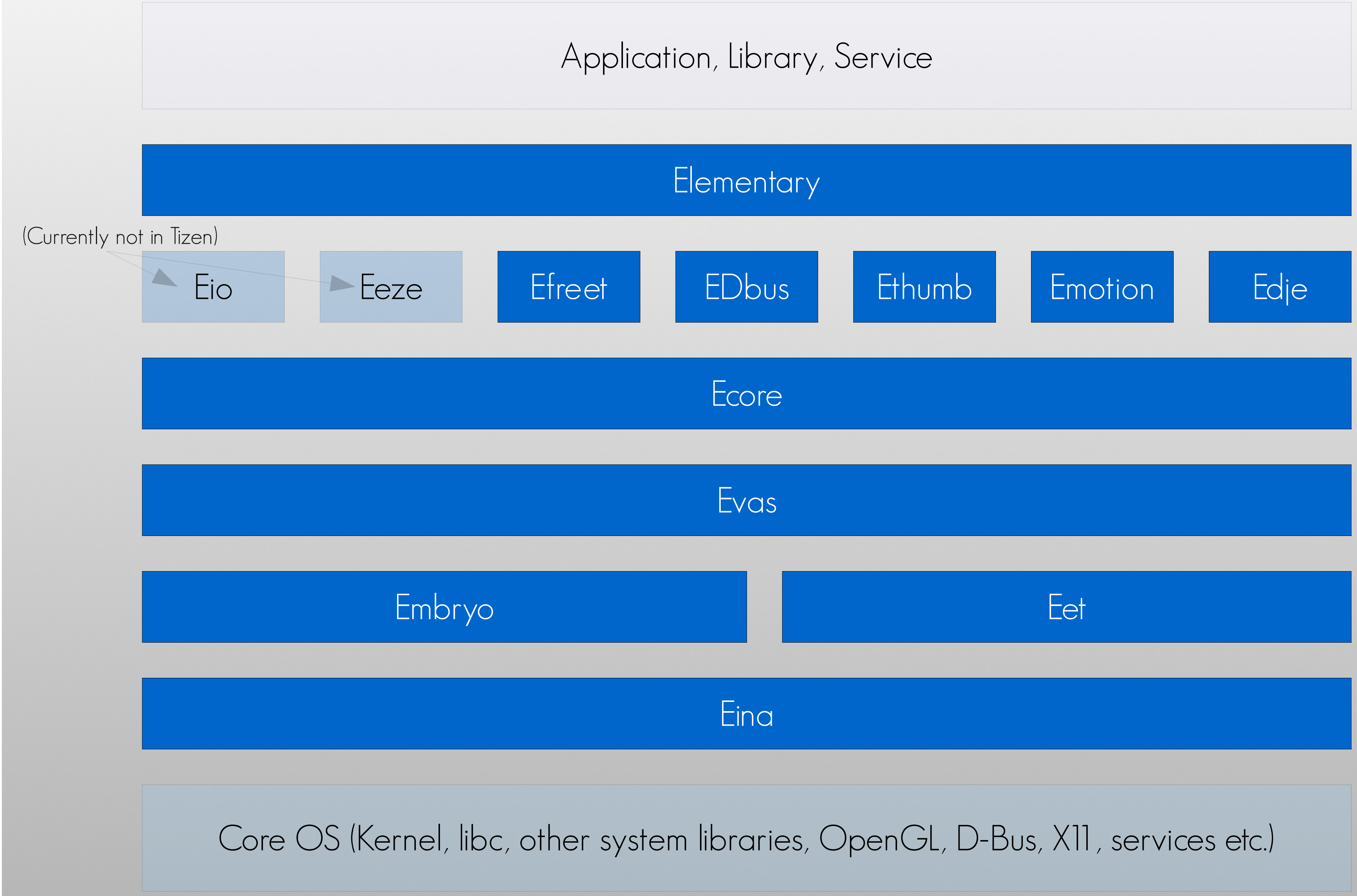
libpng/jpeg etc.

X11, OpenGL etc.

D-Bus, Services

KERNEL

Duplo time!



Application, Library, Service

Elementary

(Currently not in Tizen)

Eio

Eeze

Efreet

EDbus

Ethumb

Emotion

Edje

Ecore

Evas

Embryo

Eet

Eina

Core OS (Kernel, libc, other system libraries, OpenGL, D-Bus, X11, services etc.)

Why EFL?

- Why is EFL being used as opposed to GTK+ or Qt or something else?
  - Speed
    - Samsung used GTK+, X11 and DirectFB (in combinations) and once EFL was tried, it soundly beat these hands-down in performance
    - Very fast software rendering (for all occasions)
    - Solid Accelerated OpenGL and OpenGL-ES2.0 support for many years
    - 60fps+ on common smartphones equaling android with higher quality

Why EFL?

- Why is EFL being used as opposed to GTK+ or Qt or something else?
  - Memory (Ubuntu 11.04) beyond base X11 "failsafe" session
    - Unity - 168Mb
    - Enlightenment 0.17 - 65Mb
      - Numbers based on "free" minus disk cache and buffers - Base 199Mb
  - Both Unity and Enlightenment have roughly similar features and setup
    - Compositor (OpenGL), fullscreen wallpaper, launcher, icons, filemanager, etc.

How is this relevant?

- Mobile devices ship with limited memory
  - 128Mb, 256Mb, maybe 512Mb
- These devices almost never use swap
  - Flash has limited writes, so swap can hurt device lifespan
- Lower end devices may not have GPU's
  - Require decent software rendering to make up for it
- OpenGL has overhead that may not be worth it for all situations
  - Texture atlases to keep speed, but lose memory & more

Where do I start?

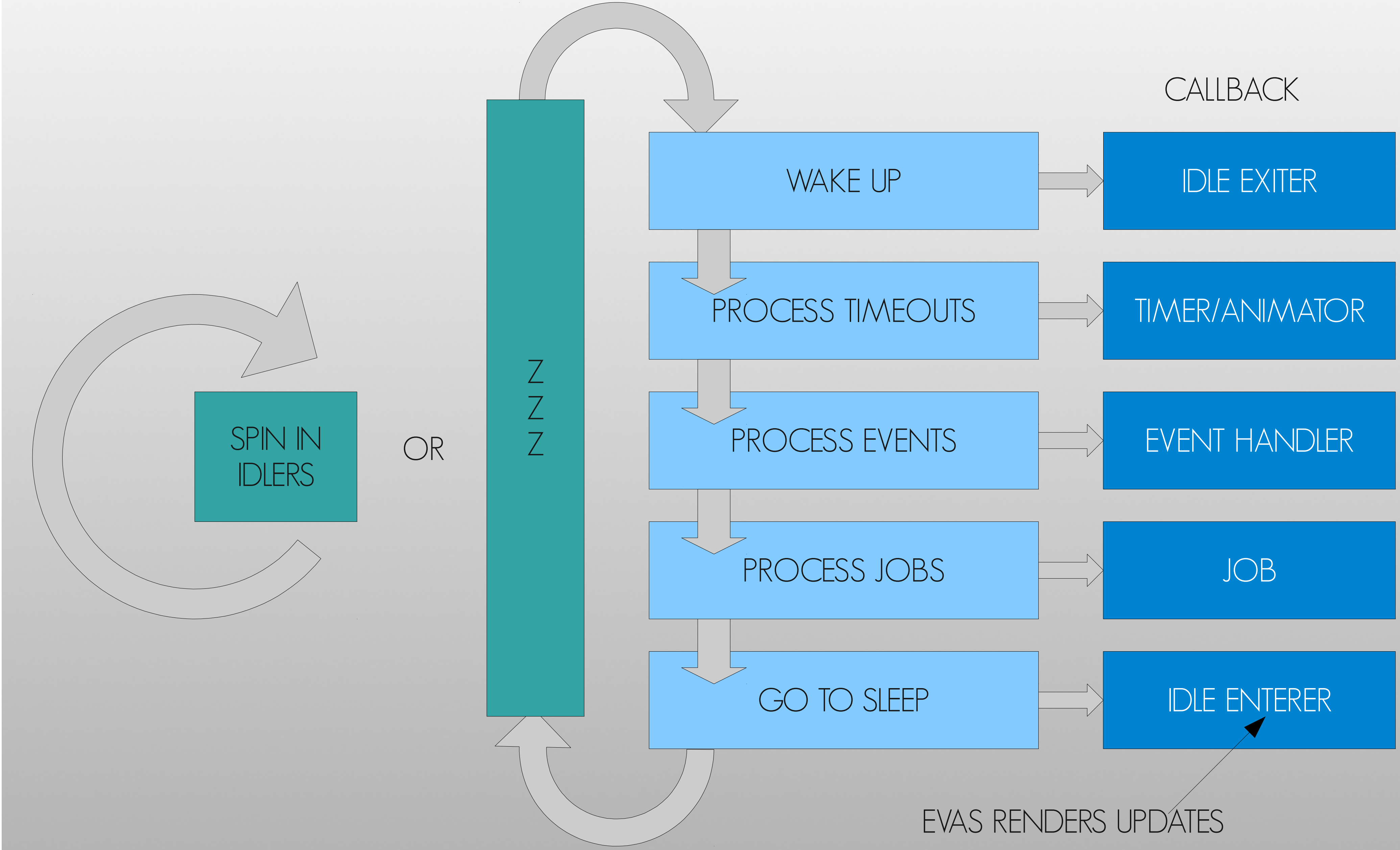
- On Tizen itself, or packages for several distros
- Some distros EFL based
- Source fetch methods
  - http://www.enlightenment.org/p.php?p=download
    - Build order and package info etc.
      - (Build Elementary and E17 last)
  - svn co http://svn.enlightenment.org/svn/e/trunk
    - Get everything yourself directly
  - http://omicron.homeip.net/projects/easy_e17/easy_e17.sh
    - Fetch from svn and build, install dependencies

ECORE

Core concepts

- Event driven mainloop
  - Rendering (UI)
  - Application state management
  - Small miscellaneous tasks (non-blocking)
- Support for threaded work
  - Similar to OSX and iOS with dispatching (Grand Central Dispatch) as well as manual dispatch and feedback
  - Added thread models with mainloop begin/end blocks and mainloop call dispatch (from threads).
  - More on threading
    - http://docs.enlightenment.org/auto/elementary/threading.html

The Mainloop (Ecore)



CALLBACK

WAKE UP

IDLE EXITER

PROCESS TIMEOUTS

TIMER/ANIMATOR

SPIN IN IDLERS

OR

Z Z Z

PROCESS EVENTS

EVENT HANDLER

PROCESS JOBS

JOB

GO TO SLEEP

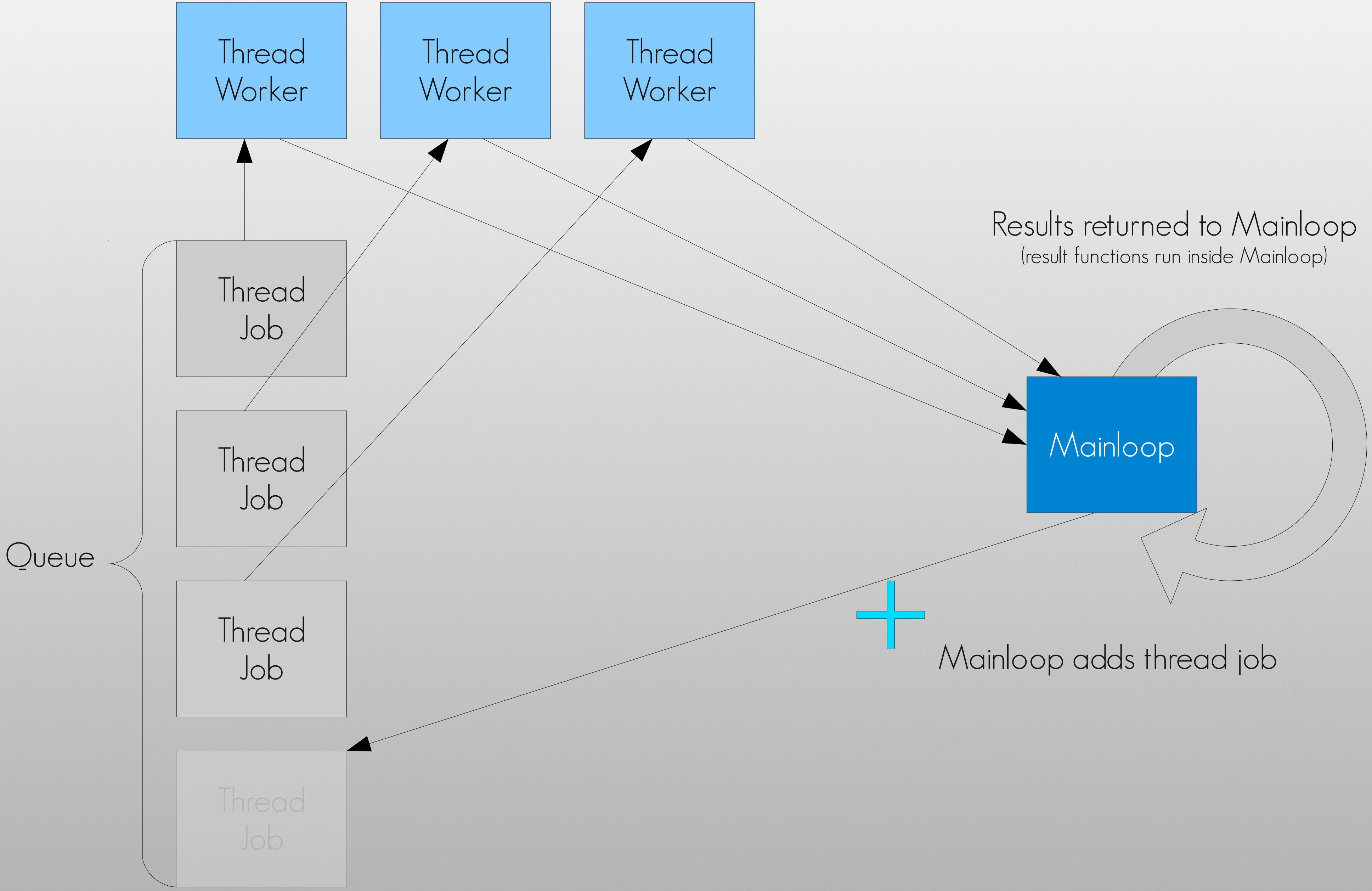
IDLE ENTERER

EVAS RENDERS UPDATES

To keep a smooth UI

- Put I/O work or heavy computation into threads
- Use the constructs provided to make this easy
- Keep state in Mainloop consistent
- Only deliver changes as a whole (UI tracks state)
  - (automatic within mainloop)
- Use Animators, not Timers for animation
- Remember that mainloop is for keeping application state
  - Blocking it blocks state (and UI) updates

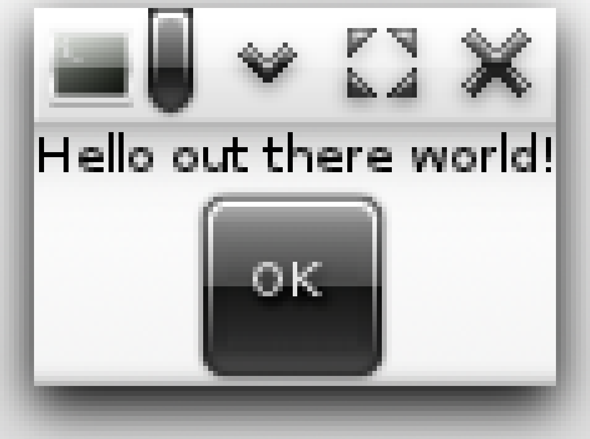
Threading the Mainloop (Ecore Thread)



Hello EFL

(in C)

```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

static void on_ok(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

int elm_main(int argc, char **argv) {
   Evas_Object *win, *box, *label, *button;

   win = elm_win_util_standard_add("main", "Hello");
   evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

   box = elm_box_add(win);

   label = elm_label_add(win);
   elm_object_text_set(label, "Hello out there world");
   elm_box_pack_end(box, label);
   evas_object_show(label);

   button = elm_button_add(win);
   elm_object_text_set(button, "OK");
   elm_box_pack_end(box, button);
   evas_object_show(button);
   evas_object_smart_callback_add(button, "clicked", on_ok, NULL);

   elm_win_resize_object_add(win, box);
   evas_object_show(box);

   evas_object_show(win);
   elm_run();
}
ELM_MAIN();
```

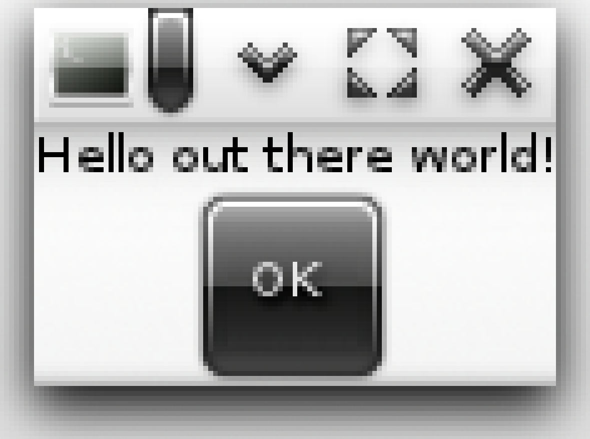


Hello out there world!

Hello EFL

(in C)

```c
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

static void on_ok(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

int elm_main(int argc, char **argv) {
   Evas_Object *win, *box, *label, *button;

   win = elm_win_util_standard_add("main", "Hello");
   evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

   box = elm_box_add(win);

   label = elm_label_add(win);
   elm_object_text_set(label, "Hello out there world");
   elm_box_pack_end(box, label);
   evas_object_show(label);

   button = elm_button_add(win);
   elm_object_text_set(button, "OK");
   elm_box_pack_end(box, button);
   evas_object_show(button);
   evas_object_smart_callback_add(button, "clicked", on_ok, NULL);

   elm_win_resize_object_add(win, box);
   evas_object_show(box);

   evas_object_show(win);
   elm_run();
}
ELM_MAIN();
```

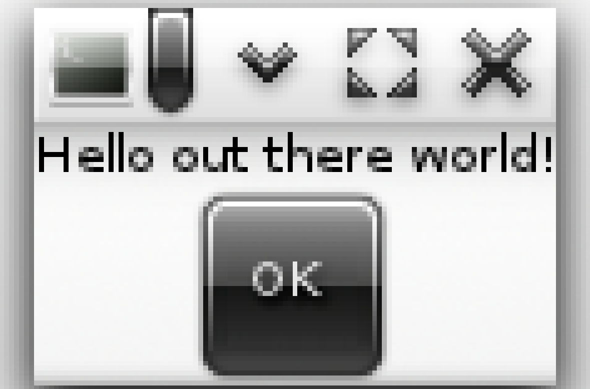


Hello out there world!

Hello EFL

(in C)

```c
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

static void on_ok(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

int elm_main(int argc, char **argv) {
   Evas_Object *win, *box, *label, *button;

   win = elm_win_util_standard_add("main", "Hello");
   evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

   box = elm_box_add(win);

   label = elm_label_add(win);
   elm_object_text_set(label, "Hello out there world");
   elm_box_pack_end(box, label);
   evas_object_show(label);

   button = elm_button_add(win);
   elm_object_text_set(button, "OK");
   elm_box_pack_end(box, button);
   evas_object_show(button);
   evas_object_smart_callback_add(button, "clicked", on_ok, NULL);


   elm_win_resize_object_add(win, box);
   evas_object_show(box);

   evas_object_show(win);
   elm_run();
}
ELM_MAIN();
```

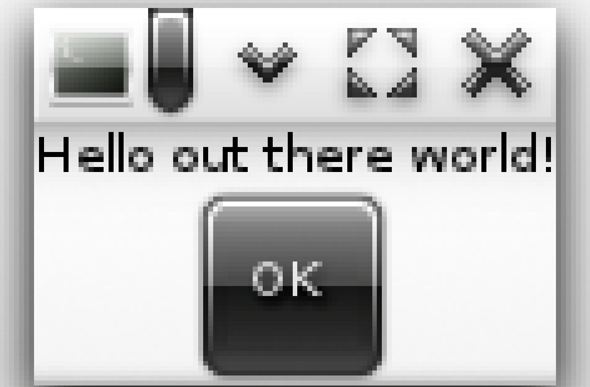


Hello out there world!

Hello EFL

(in C)

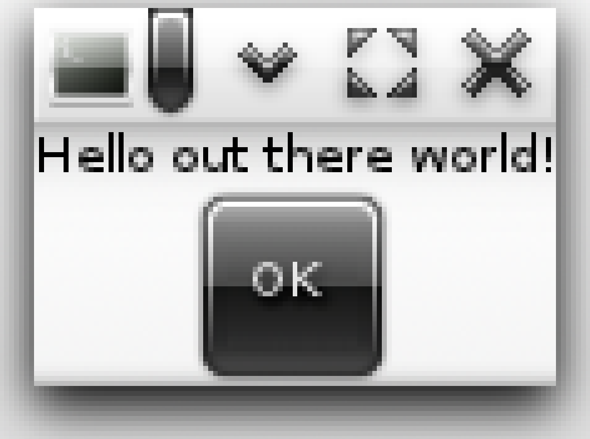
```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

static void on_ok(void *data, Evas_Object *obj, void *event_info) {
   elm_exit();
}

int elm_main(int argc, char **argv) {
   Evas_Object *win, *box, *label, *button;

   win = elm_win_util_standard_add("main", "Hello");
   evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

   box = elm_box_add(win);

   label = elm_label_add(win);
   elm_object_text_set(label, "Hello out there world");
   elm_box_pack_end(box, label);
   evas_object_show(label);

   button = elm_button_add(win);
   elm_object_text_set(button, "OK");
   elm_box_pack_end(box, button);
   evas_object_show(button);
   evas_object_smart_callback_add(button, "clicked", on_ok, NULL);

   elm_win_resize_object_add(win, box);
   evas_object_show(box);

   evas_object_show(win);
   elm_run();
}
ELM_MAIN();
```



Hello out there world!

Hello EFL

(in C)



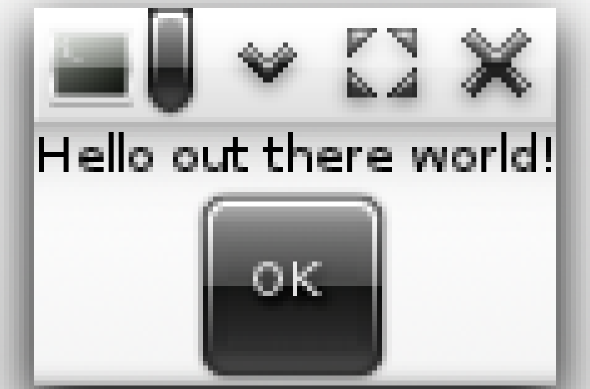
Hello out there world!

```
$ gcc hello.c -o hello `pkg-config --cflags --libs elementary`
$ ./hello
```

Hello EFL

(in Elev8 / Javascript)

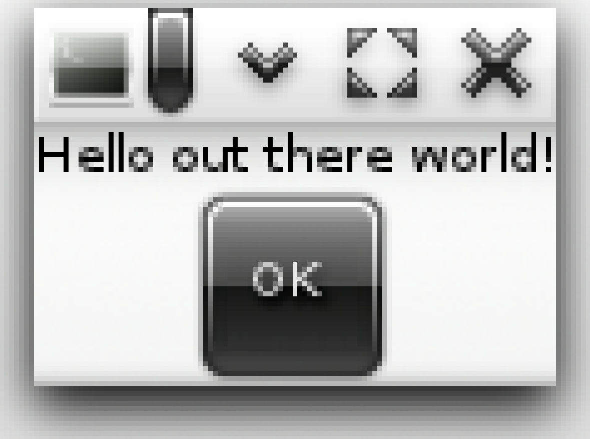
```
var my_window = new elm.window({
   label: "Hello",
   elements: {
      bg: { type: "background", resize: true },
      box: { type: "box",
         resize: true,
         elements: {
           label: { type: "label",
              label: "Hello out there world"
           },
           button: { type: "button",
              label: "OK",
              on_clicked: function(arg) {
                 elm.exit();
              }
           }
         }
      }
   }
});
```



Hello out there world!

Hello EFL

(in Elev8 / Javascript)

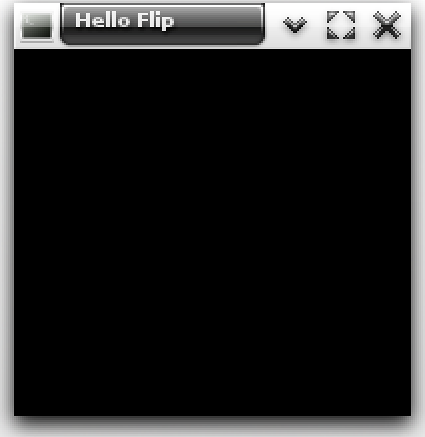


Hello out there world!

```
$ elev8 hello.js
```

Getting fancy

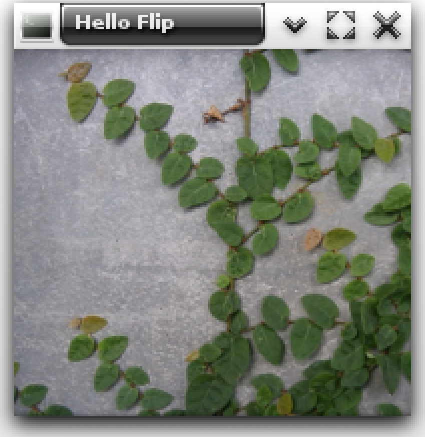
(in C)



```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
    elm_exit();
}

int elm_main(int argc, char **argv) {
    Evas_Object *win;

    win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
    elm_win_title_set(win, "Hello Flip");
    evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

    evas_object_show(win);
    elm_run();
}
ELM_MAIN();
```

Getting fancy

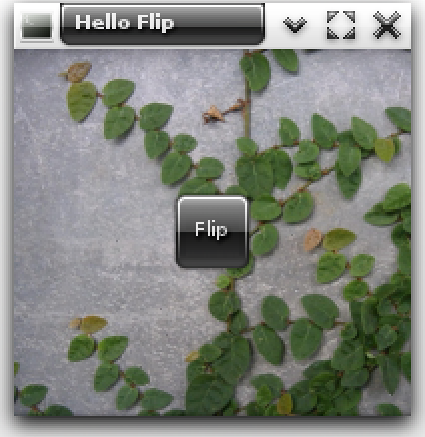
(in C)



```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
          elm_exit();
}

int elm_main(int argc, char **argv) {
          Evas_Object *win, *bg;

          win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
          elm_win_title_set(win, "Hello Flip");
          evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

          bg = elm_bg_add(win);
          elm_bg_file_set(bg, "plant.jpg", NULL);
          elm_win_resize_object_add(win, bg);
          evas_object_show(bg);

          evas_object_show(win);
          elm_run();
}
ELM_MAIN();
```

Getting fancy

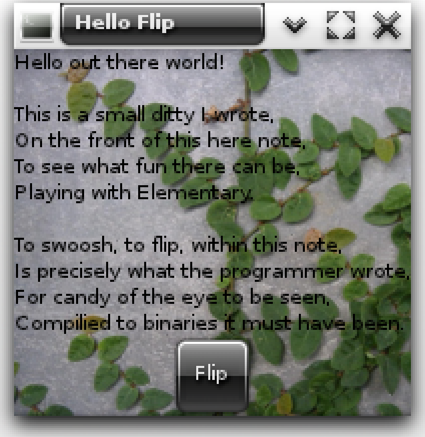
(in C)



```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
          elm_exit();
}

int elm_main(int argc, char **argv) {
          Evas_Object *win, *bg, *box, *flip, *button;

          win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
          elm_win_title_set(win, "Hello Flip");
          evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

          bg = elm_bg_add(win);
          elm_bg_file_set(bg, "plant.jpg", NULL);
          elm_win_resize_object_add(win, bg);
          evas_object_show(bg);

          box = elm_box_add(win);

          flip = elm_flip_add(win);
          elm_box_pack_end(box, flip);
          evas_object_show(flip);

          button = elm_button_add(win);
          elm_object_text_set(button, "Flip");
          elm_box_pack_end(box, button);
          evas_object_show(button);
          evas_object_smart_callback_add(button, "clicked", on_flip, flip);

          elm_win_resize_object_add(win, box);
          evas_object_show(box);

          evas_object_show(win);
          elm_run();
}
ELM_MAIN();
```

Getting fancy

(in C)



Hello out there world!

This is a small ditty I wrote,
On the front of this here note,
To see what fun there can be,
Playing with Elementary.

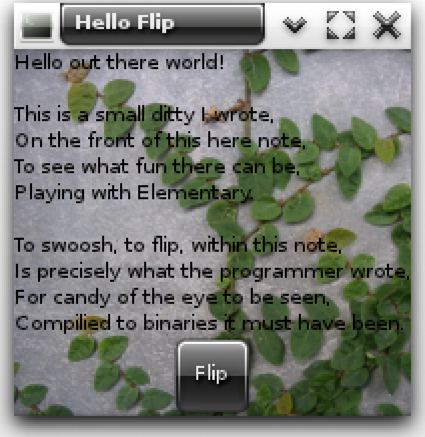
To swoosh, to flip, within this note,
Is precisely what the programmer wrote,
For candy of the eye to be seen,
Compilied to binaries it must have been.

```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
        elm_exit();
}

static void on_flip(void *data, Evas_Object *obj, void *event_info) {
        Evas_Object *flip = data;
        elm_flip_go(flip, ELM_FLIP_CUBE_UP);
}

int elm_main(int argc, char **argv) {
        Evas_Object *win, *bg, *box, *flip, *label, *button;

        win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
        elm_win_title_set(win, "Hello Flip");
        evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

        bg = elm_bg_add(win);
        elm_bg_file_set(bg, "plant.jpg", NULL);
        elm_win_resize_object_add(win, bg);
        evas_object_show(bg);

        box = elm_box_add(win);

        flip = elm_flip_add(win);
        elm_box_pack_end(box, flip);
        evas_object_show(flip);

        label = elm_label_add(win);
        elm_object_text_set(label,
                            "Hello out there world!<br>"
                            "<br>"
                            "This is a small ditty I wrote,<br>"
                            "On the front of this here note,<br>"
                            "To see what fun there can be,<br>"
                            "Playing with Elementary.<br>"
                            "<br>"
                            "To swoosh, to flip, within this note,<br>"
                            "Is precisely what the programmer wrote,<br>"
                            "For candy of the eye to be seen,<br>"
                            "Compiled to binaries it must have been.");
        evas_object_show(label);
        elm_flip_content_front_set(flip, label);

        button = elm_button_add(win);
        elm_object_text_set(button, "Flip");
        elm_box_pack_end(box, button);
        evas_object_show(button);
        evas_object_smart_callback_add(button, "clicked", on_flip, flip);

        elm_win_resize_object_add(win, box);
        evas_object_show(box);

        evas_object_show(win);
        elm_run();
```

Getting fancy

(in C)

```
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
        elm_exit();
}

static void on_flip(void *data, Evas_Object *obj, void *event_info) {
        Evas_Object *flip = data;
        elm_flip_go(flip, ELM_FLIP_CUBE_UP);
}

int elm_main(int argc, char **argv) {
        Evas_Object *win, *bg, *box, *flip, *label, *list, *button;

        win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
        elm_win_title_set(win, "Hello Flip");
        evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

        bg = elm_bg_add(win);
        elm_bg_file_set(bg, "plant.jpg", NULL);
        elm_win_resize_object_add(win, bg);
        evas_object_show(bg);

        box = elm_box_add(win);

        flip = elm_flip_add(win);
        elm_box_pack_end(box, flip);
        evas_object_show(flip);

        label = elm_label_add(win);
        elm_object_text_set(label,
                            "Hello out there world!<br>"
                            "<br>"
                            "This is a small ditty I wrote,<br>"
                            "On the front of this here note,<br>"
                            "To see what fun there can be,<br>"
                            "Playing with Elementary.<br>"
                            "<br>"
                            "To swoosh, to flip, within this note,<br>"
                            "Is precisely what the programmer wrote,<br>"
                            "For candy of the eye to be seen,<br>"
                            "Compiled to binaries it must have been.");
        evas_object_show(label);
        elm_flip_content_front_set(flip, label);

        list = elm_list_add(win);
        elm_list_item_append(list, "Eye of newt,", NULL, NULL, NULL, NULL);
        elm_list_item_append(list, "And toe of frog,", NULL, NULL, NULL, NULL);
        elm_list_item_append(list, "Wool of bat,", NULL, NULL, NULL, NULL);
        elm_list_item_append(list, "And tongue of dog,", NULL, NULL, NULL, NULL);
        elm_list_item_append(list, "Adder's fork,", NULL, NULL, NULL, NULL);
        elm_list_go(list);
        evas_object_show(list);
        elm_flip_content_back_set(flip, list);

        button = elm_button_add(win);
        elm_object_text_set(button, "Flip");
        elm_box_pack_end(box, button);
        evas_object_show(button);
        evas_object_smart_callback_add(button, "clicked", on_flip, flip);

        elm_win_resize_object_add(win, box);
        evas_object_show(box);

        evas_object_show(win);
```



Hello out there world!

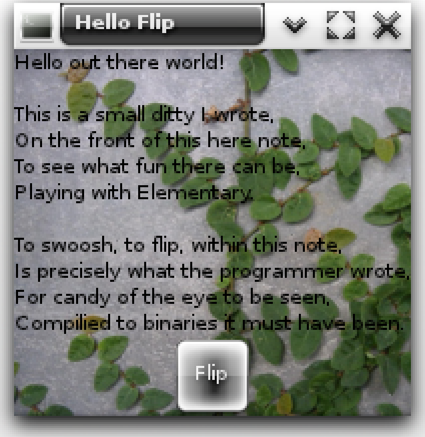
This is a small ditty I wrote,
On the front of this here note,
To see what fun there can be,
Playing with Elementary.

To swoosh, to flip, within this note,
Is precisely what the programmer wrote,
For candy of the eye to be seen,
Compiled to binaries it must have been.

Getting fancy

(in C)

```c
#include <Elementary.h>

static void on_win_del(void *data, Evas_Object *obj, void *event_info) {
          elm_exit();
}

static void on_flip(void *data, Evas_Object *obj, void *event_info) {
          Evas_Object *flip = data;
          elm_flip_go(flip, ELM_FLIP_CUBE_UP);
}

int elm_main(int argc, char **argv) {
          Evas_Object *win, *bg, *box, *flip, *label, *list, *button;

          win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
          elm_win_title_set(win, "Hello Flip");
          evas_object_smart_callback_add(win, "delete,request", on_win_del, NULL);

          bg = elm_bg_add(win);
          elm_bg_file_set(bg, "plant.jpg", NULL);
          elm_win_resize_object_add(win, bg);
          evas_object_show(bg);

          box = elm_box_add(win);

          flip = elm_flip_add(win);
          elm_box_pack_end(box, flip);
          evas_object_show(flip);

          label = elm_label_add(win);
          elm_object_text_set(label,
                              "Hello out there world!<br>"
                              "<br>"
                              "This is a small ditty I wrote,<br>"
                              "On the front of this here note,<br>"
                              "To see what fun there can be,<br>"
                              "Playing with Elementary.<br>"
                              "<br>"
                              "To swoosh, to flip, within this note,<br>"
                              "Is precisely what the programmer wrote,<br>"
                              "For candy of the eye to be seen,<br>"
                              "Compiled to binaries it must have been.");
          evas_object_show(label);
          elm_flip_content_front_set(flip, label);

          list = elm_list_add(win);
          elm_list_item_append(list, "Eye of newt,", NULL, NULL, NULL, NULL);
          elm_list_item_append(list, "And toe of frog,", NULL, NULL, NULL, NULL);
          elm_list_item_append(list, "Wool of bat,", NULL, NULL, NULL, NULL);
          elm_list_item_append(list, "And tongue of dog,", NULL, NULL, NULL, NULL);
          elm_list_item_append(list, "Adder's fork,", NULL, NULL, NULL, NULL);
          elm_list_go(list);
          evas_object_show(list);
          elm_flip_content_back_set(flip, list);

          button = elm_button_add(win);
          elm_object_text_set(button, "Flip");
          elm_box_pack_end(box, button);
          evas_object_show(button);
          evas_object_smart_callback_add(button, "clicked", on_flip, flip);

          elm_win_resize_object_add(win, box);
          evas_object_show(box);

          evas_object_show(win);
          elm_run();
}
ELM_MAIN();
```



Hello out there world!

This is a small ditty I wrote,
On the front of this here note,
To see what fun there can be,
Playing with Elementary.

To swoosh, to flip, within this note,
Is precisely what the programmer wrote,
For candy of the eye to be seen,
Compiled to binaries it must have been.

EVAS

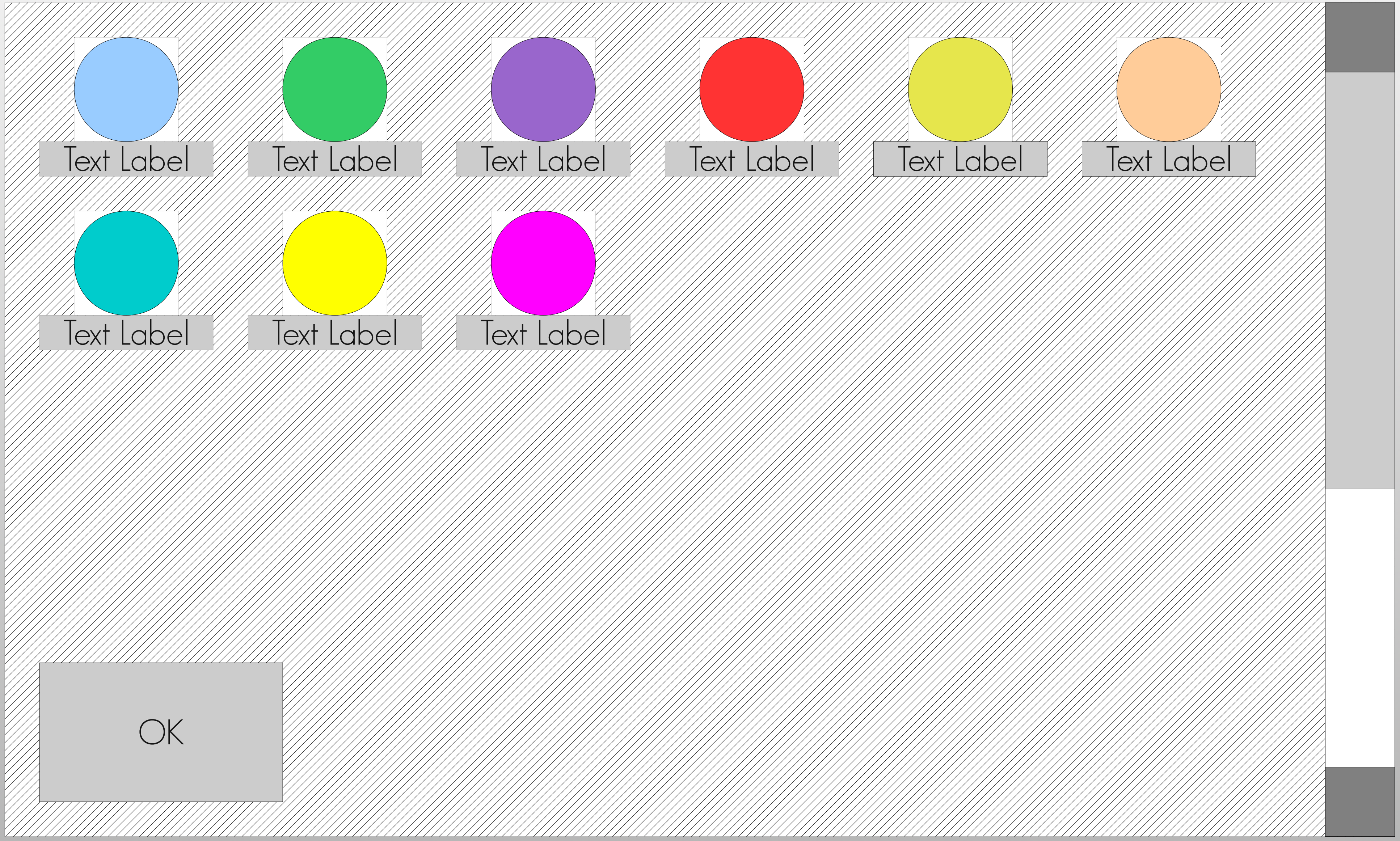
What is a scene graph? (Evas)

- Tracks state of all display objects
  - Position, size, visibility, color, properties etc.
- Handles rendering of each object
  - Loading fonts, images, rendering glyphs, scaling, fading etc.
- Handles minimizing of rendering
  - Only update areas changed
  - If changes obscured, reduce to a NOP
- Optimize rendering
  - Abstract to OpenGL, software, or anything else

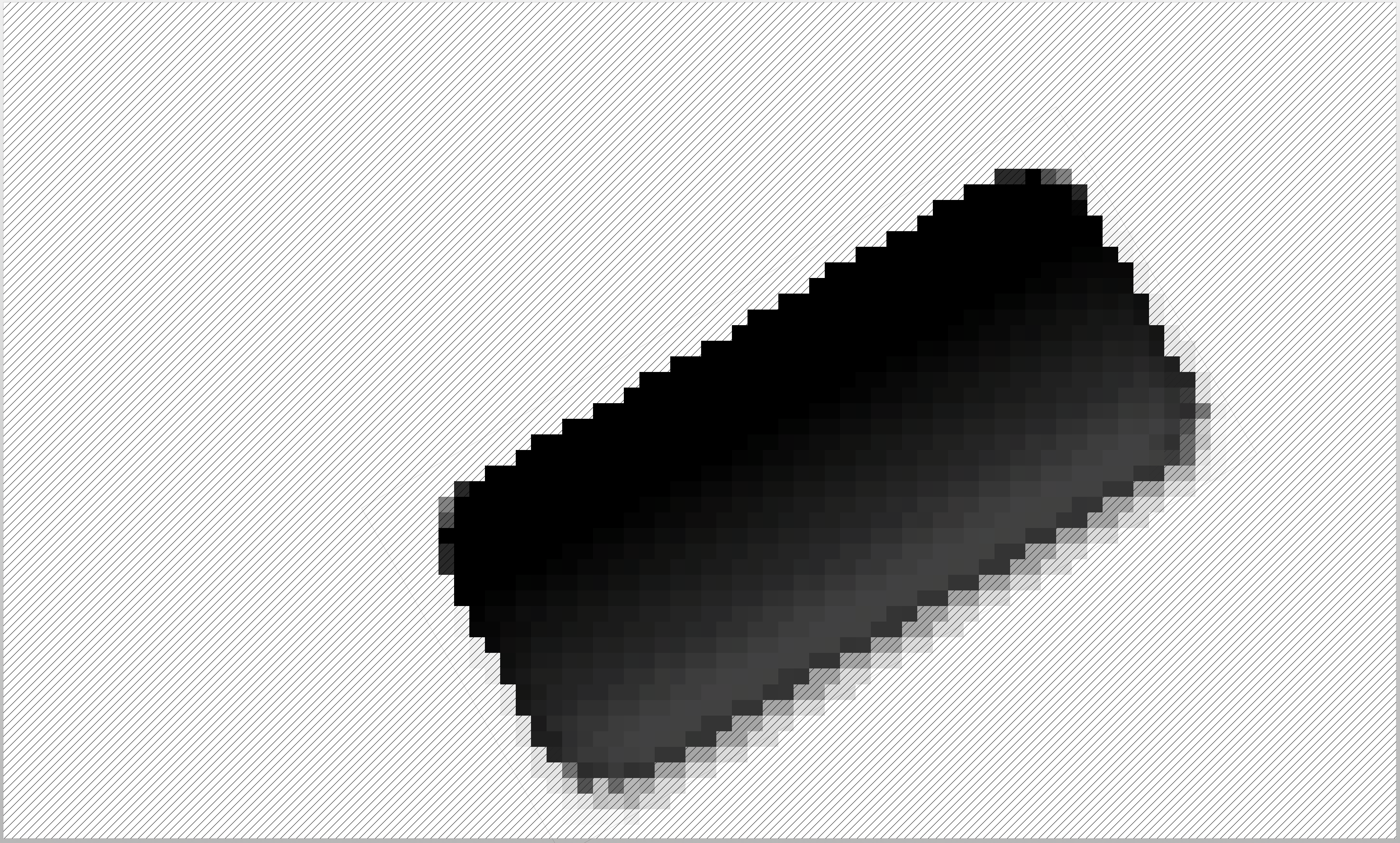
What is a scene graph? (Evas)

- Allows you to build your own composite objects
  - Creates parent/child relationship
  - Is used throughout EFL to build widgets etc.
- Handles input direction and event callbacks
- Text formatting & layout

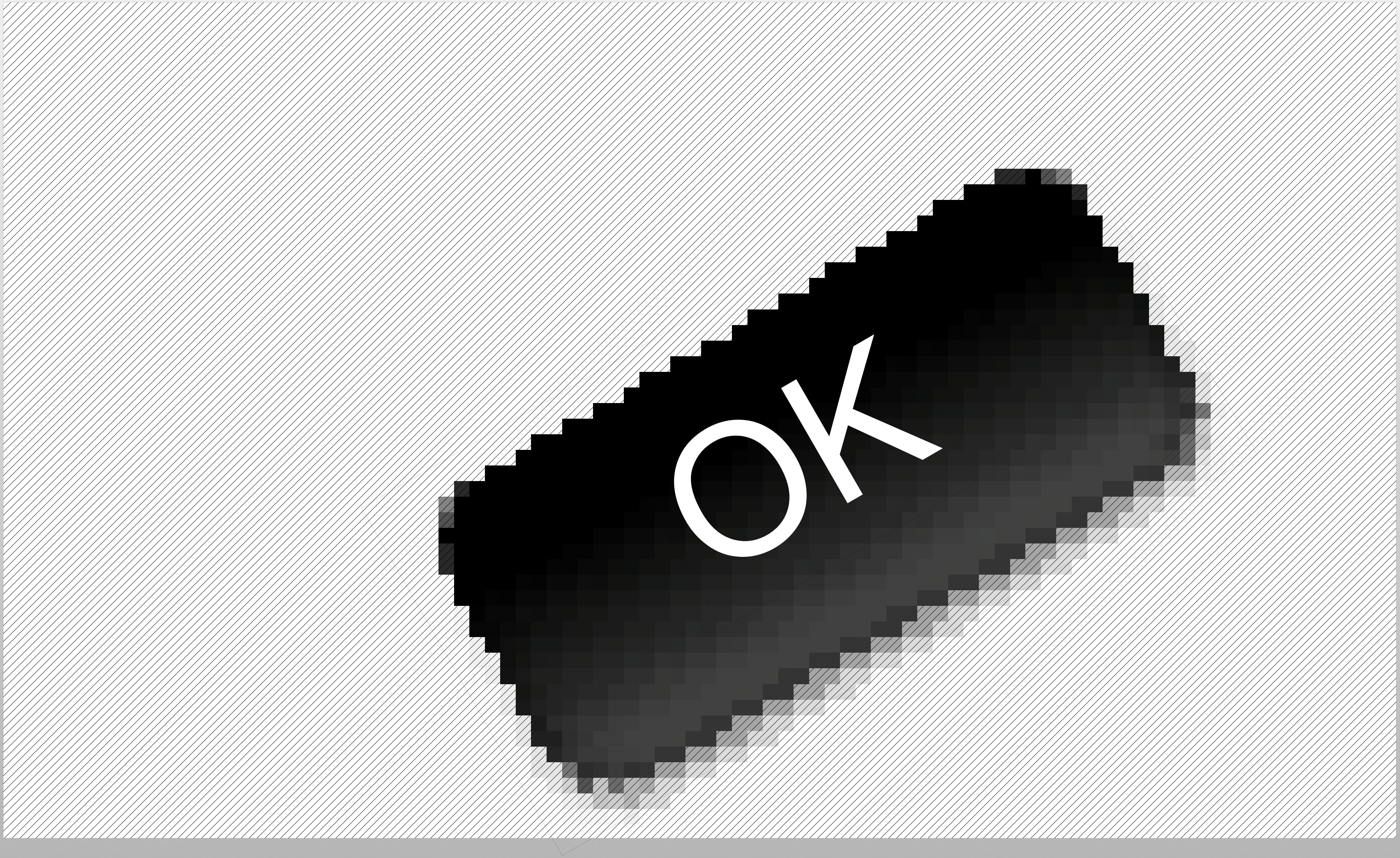
What is a scene graph? (Evas)



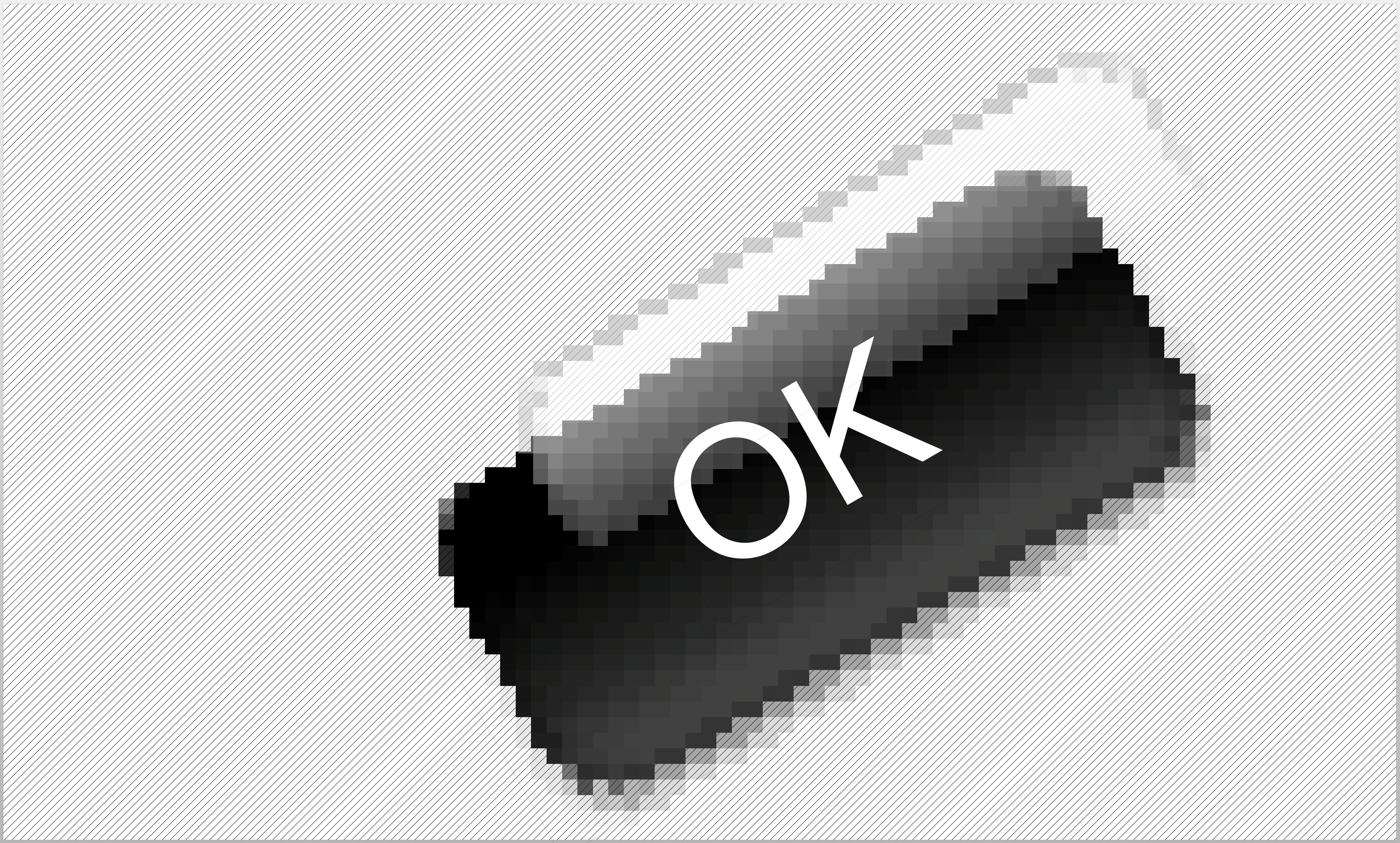
Complex objects out of simple ones



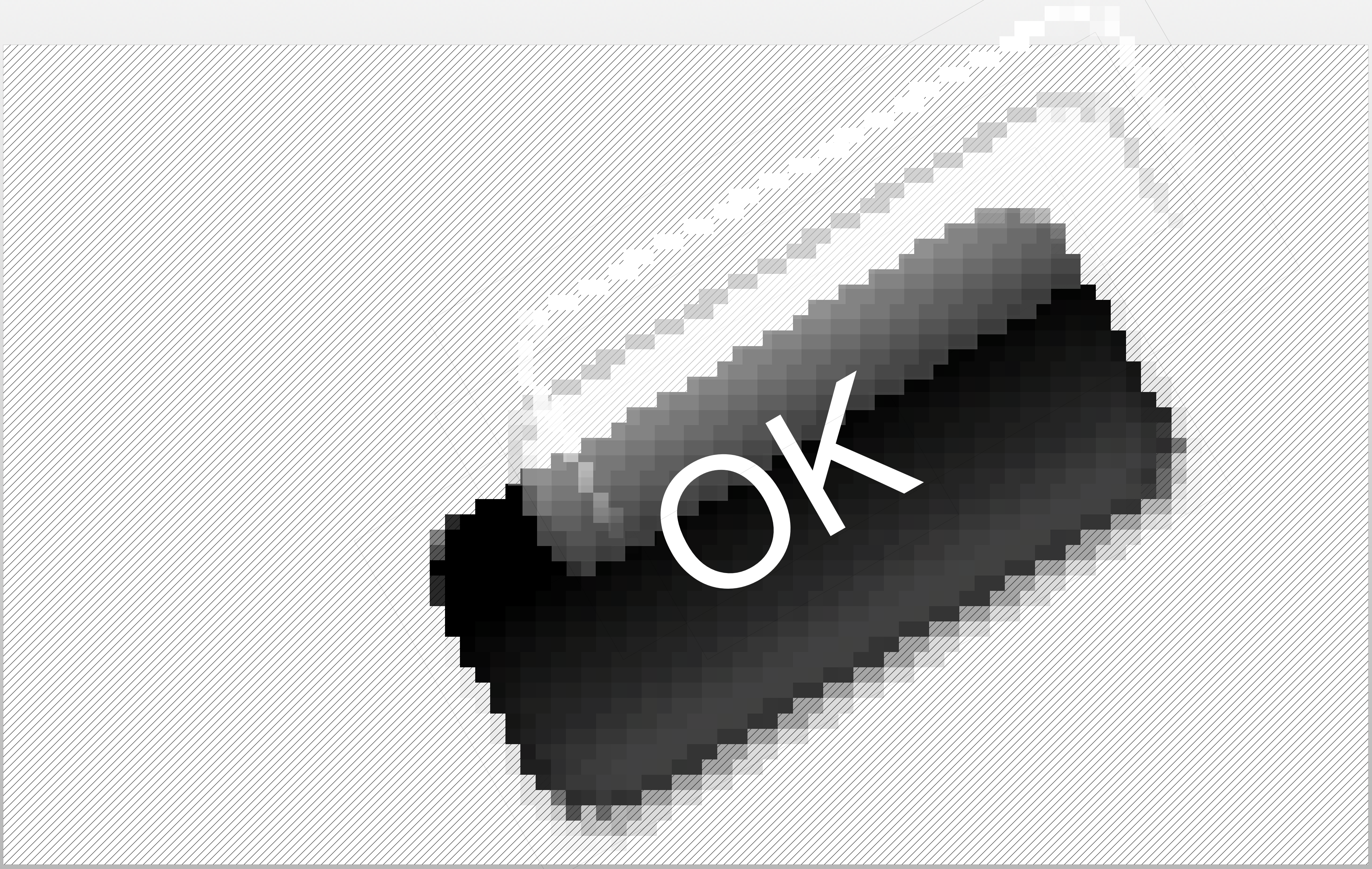
Complex objects out of simple ones



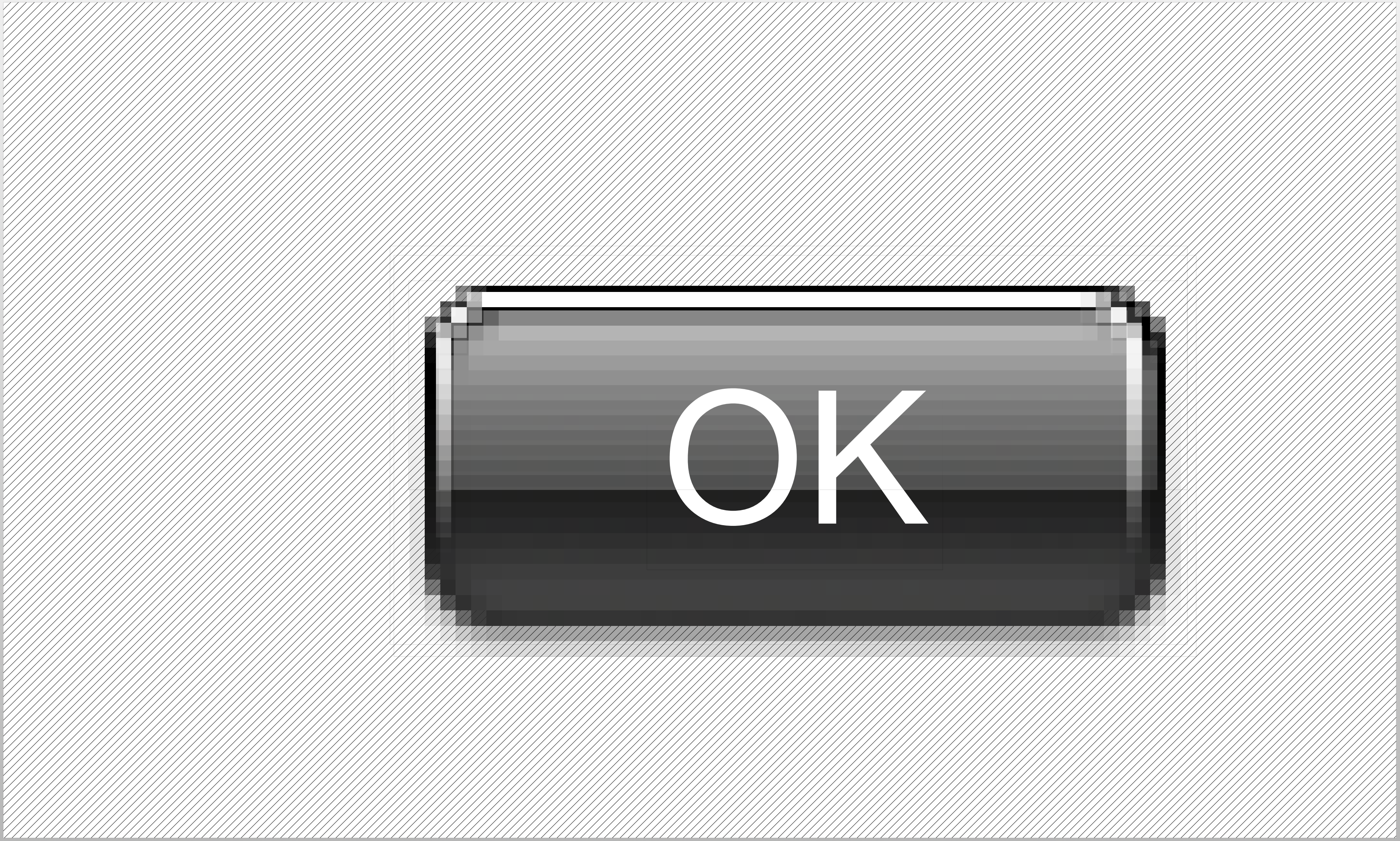
Complex objects out of simple ones



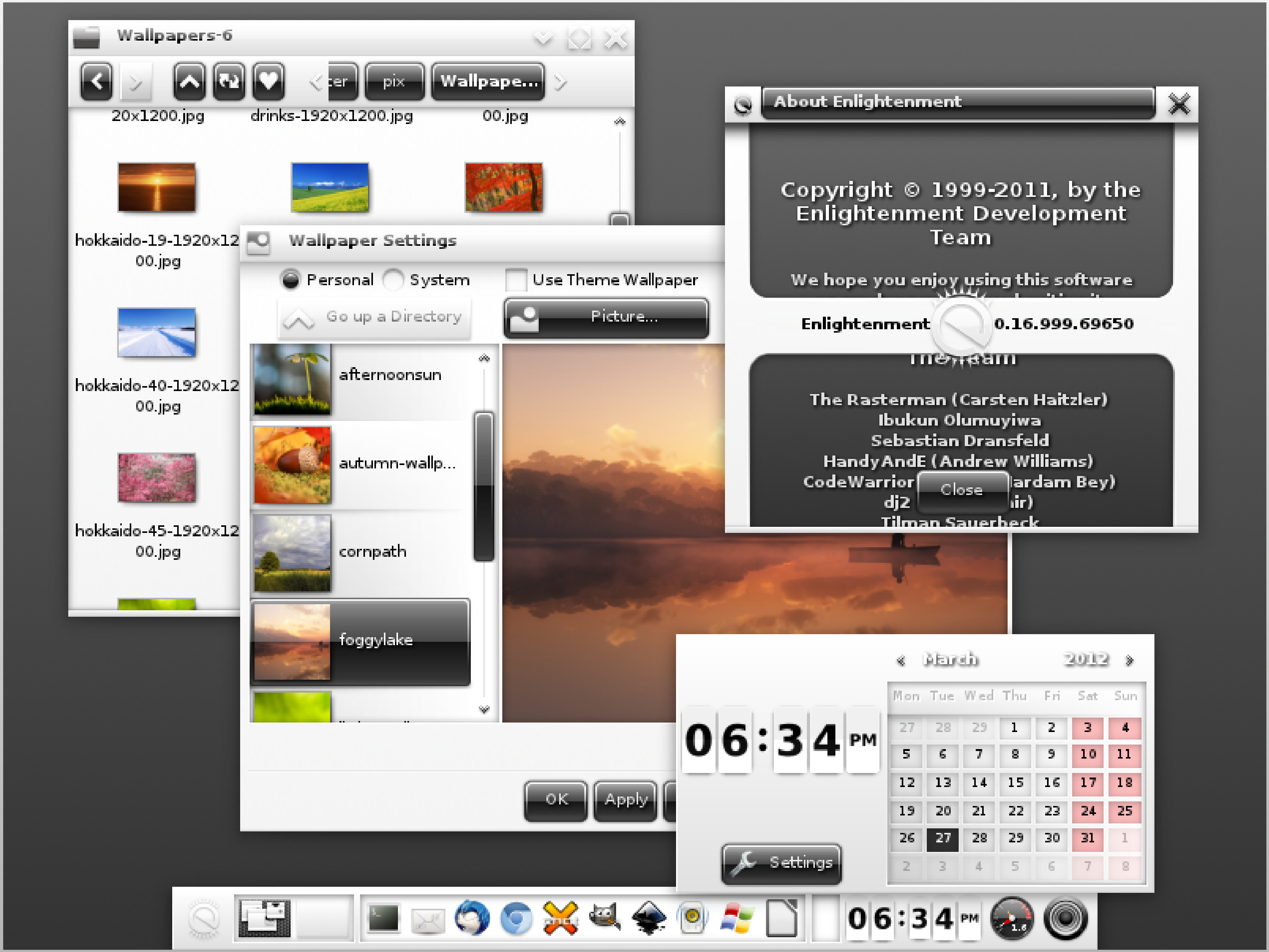
Complex objects out of simple ones



Complex objects out of simple ones



Putting together objects



Copyright © 1999-2011, by the Enlightenment Development Team

We hope you enjoy using this software

Enlightenment 0.16.999.69650

The Team

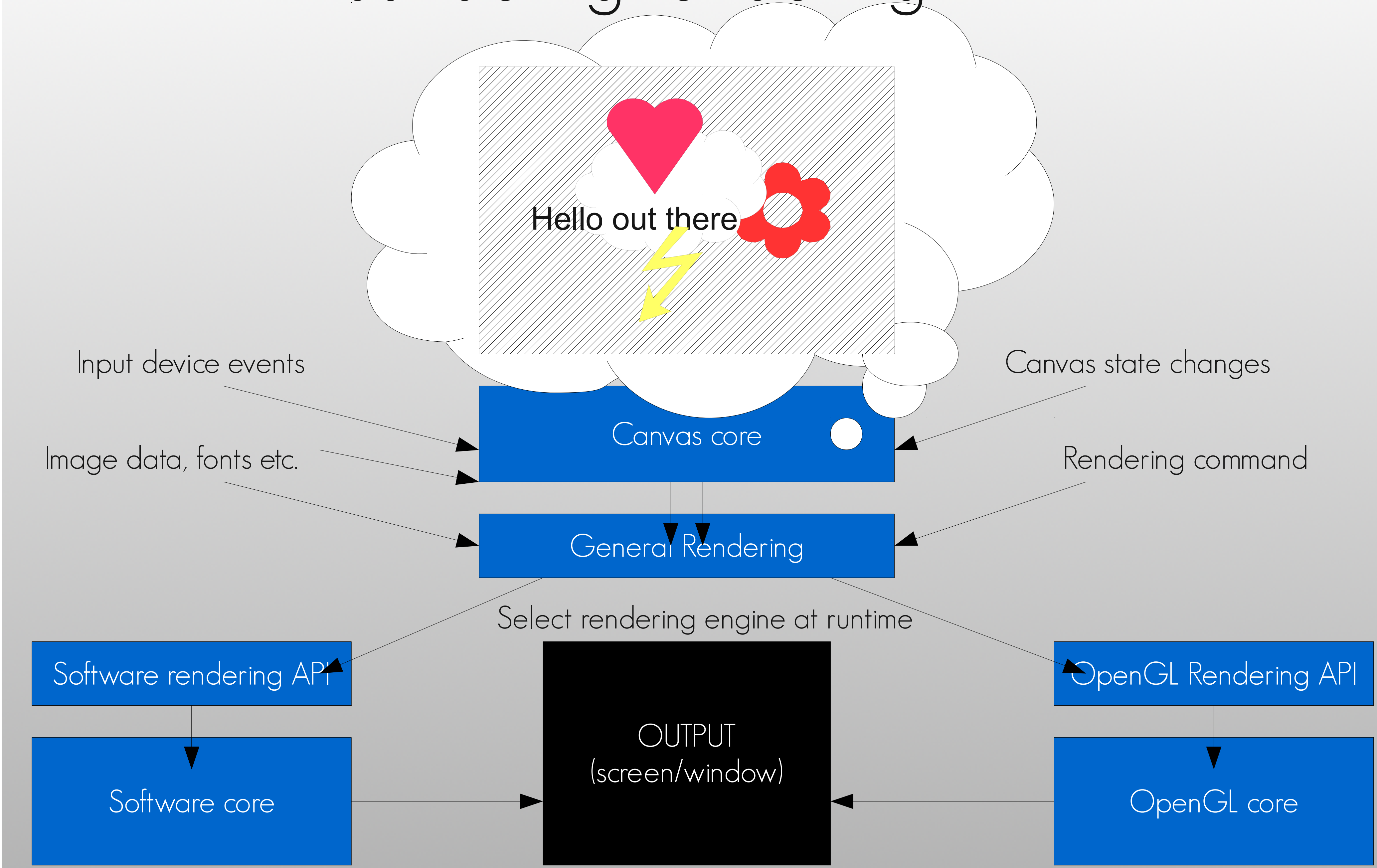
The Rasterman (Carsten Haitzler)
Ibukun Olumuyiwa
Sebastian Dransfeld
HandyAndE (Andrew Williams)
CodeWarrior ardam Bey)
dj2 ir)
Tilman Sauerbeck

06:34 PM

March 2012

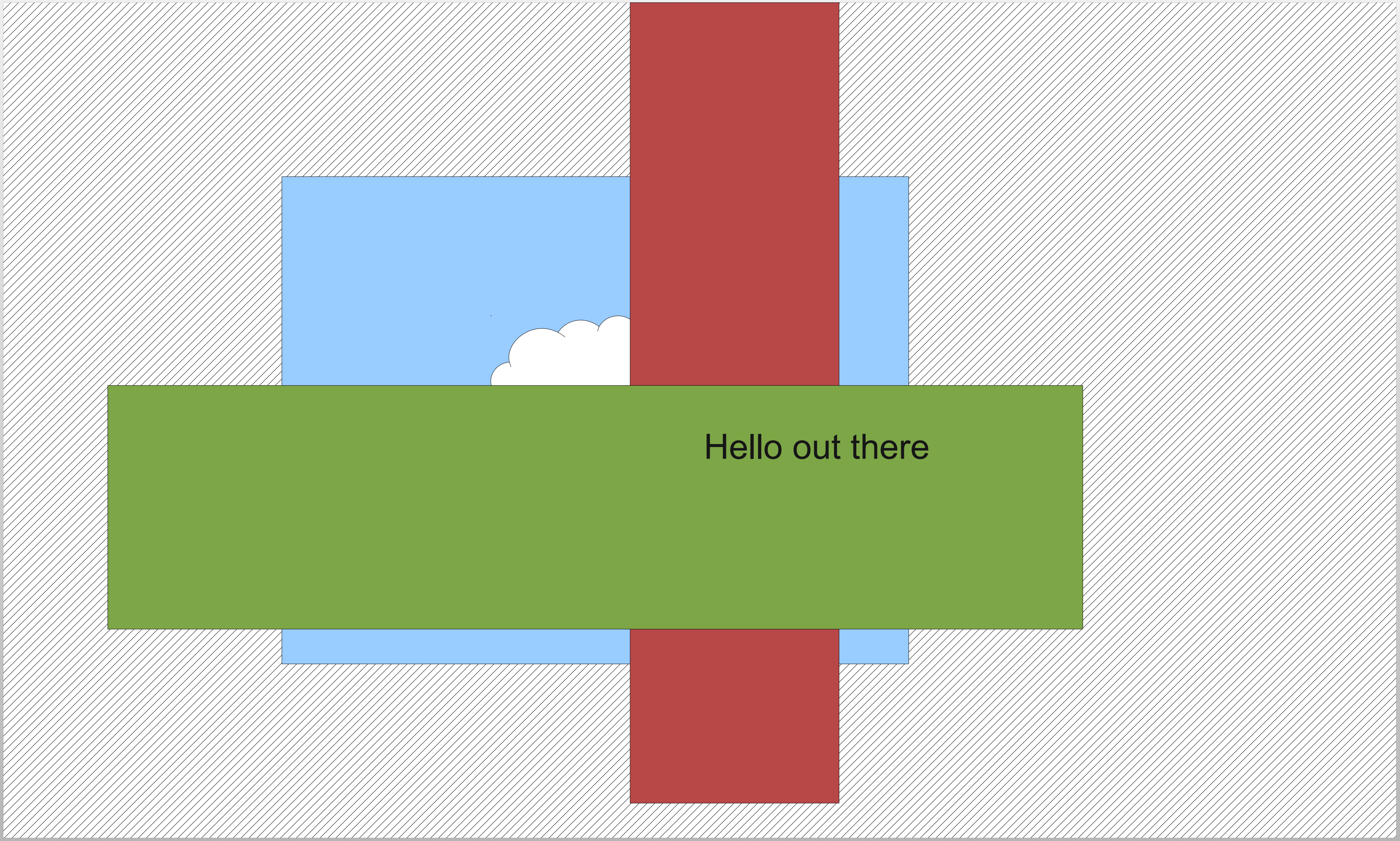
Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Abstracting rendering



Automated update handling

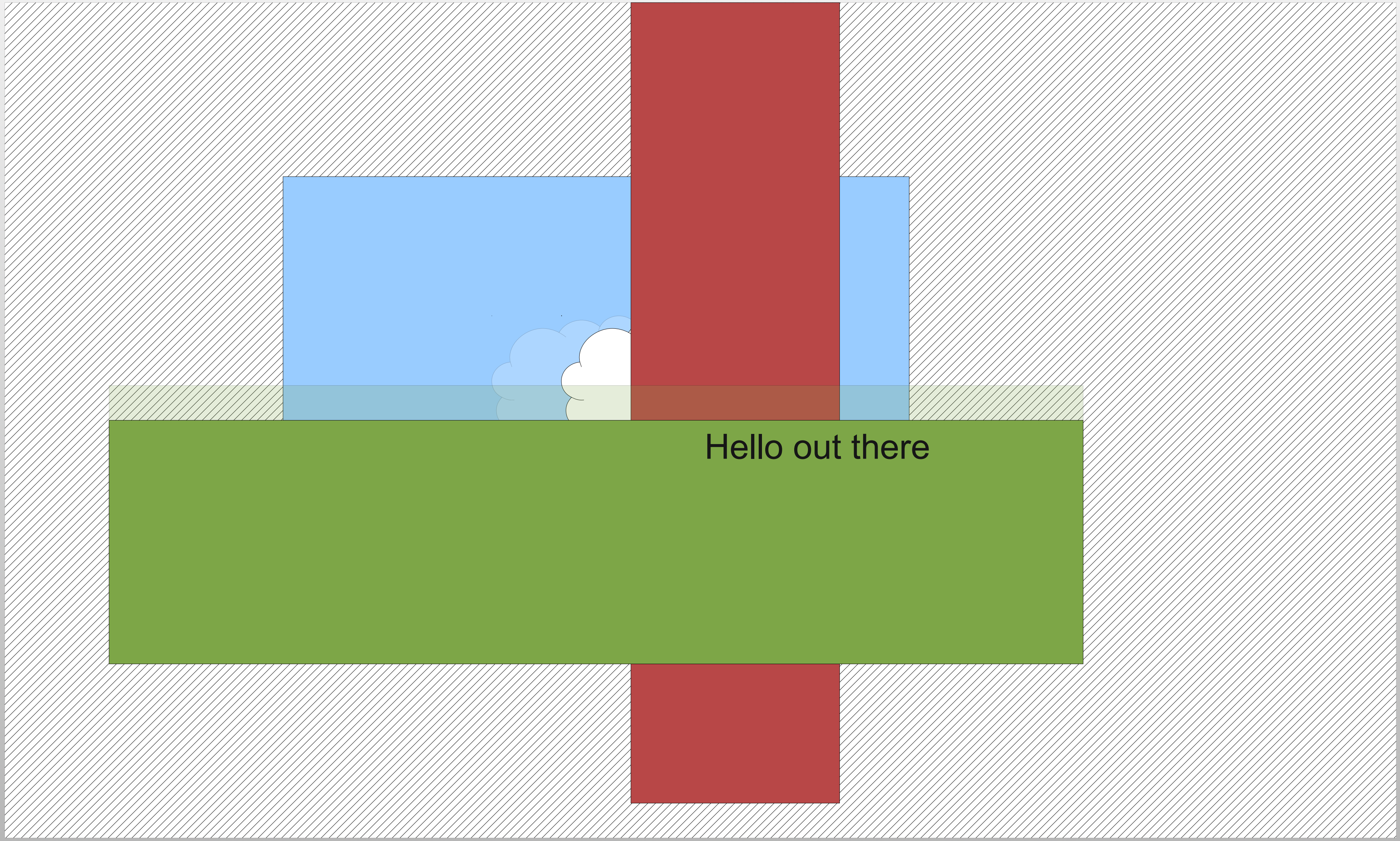
Start here



Hello out there

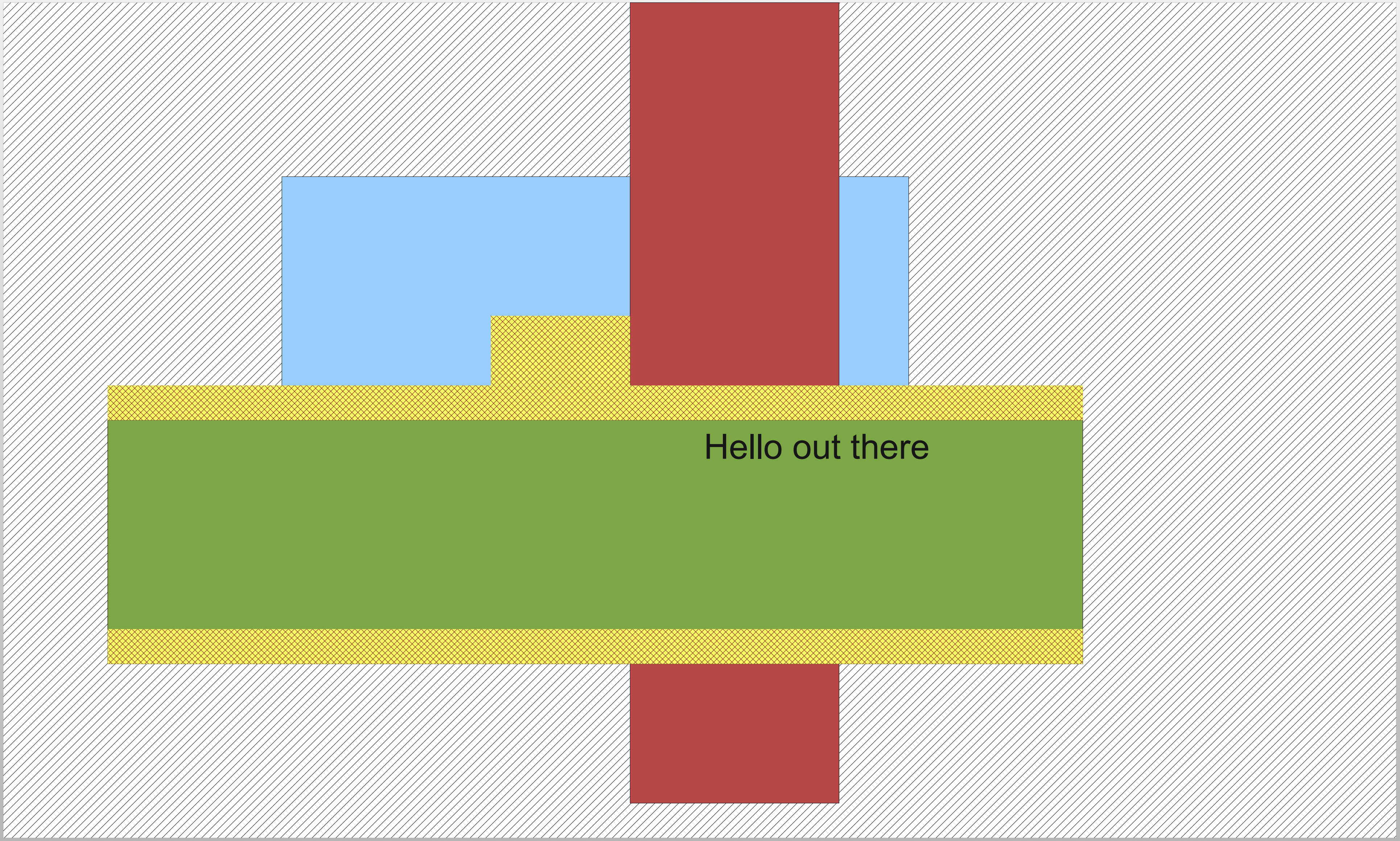
Automated update handling

Next frame is here



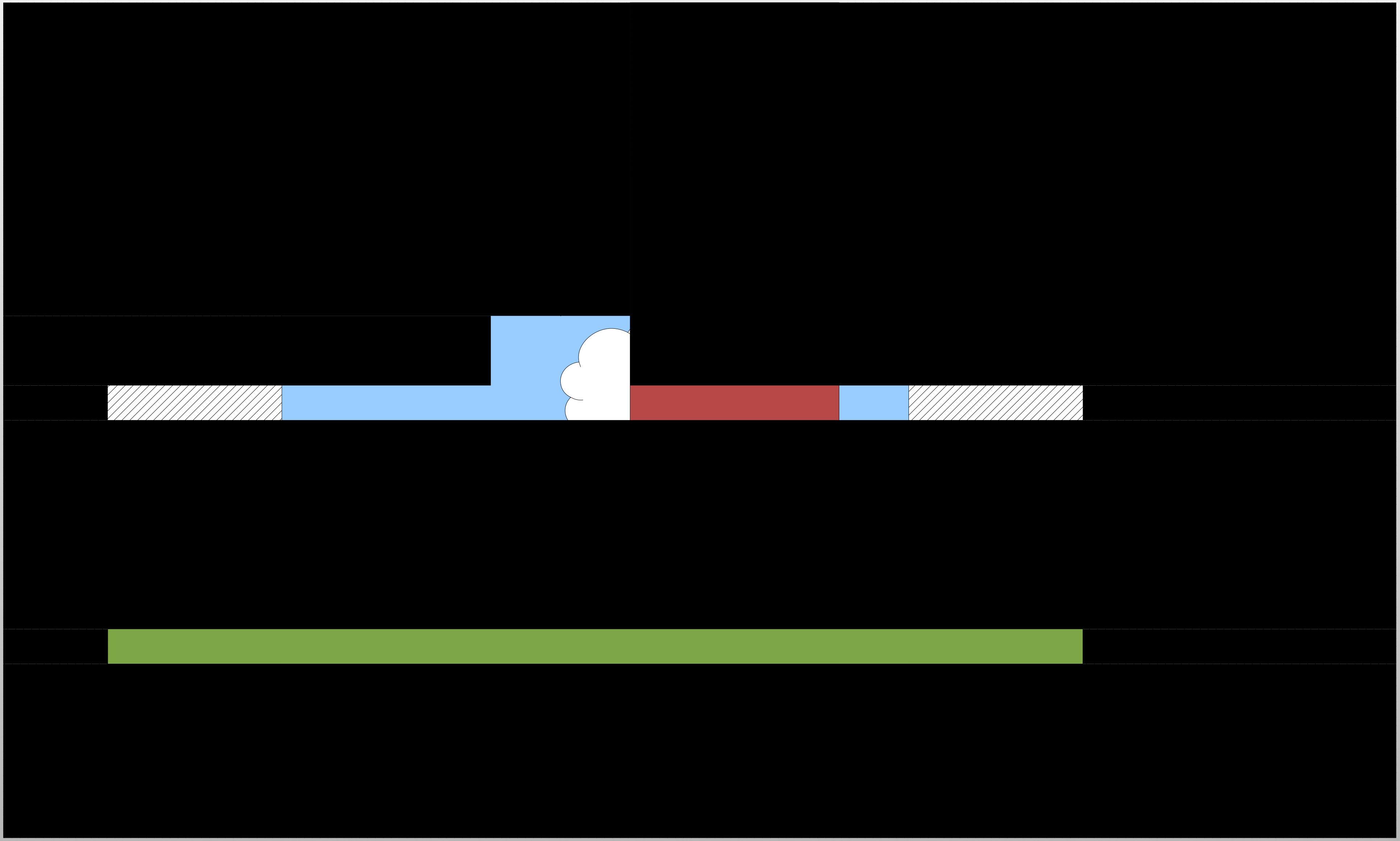
Automated update handling

Calculate actual update region deltas (up to each engine to implement)



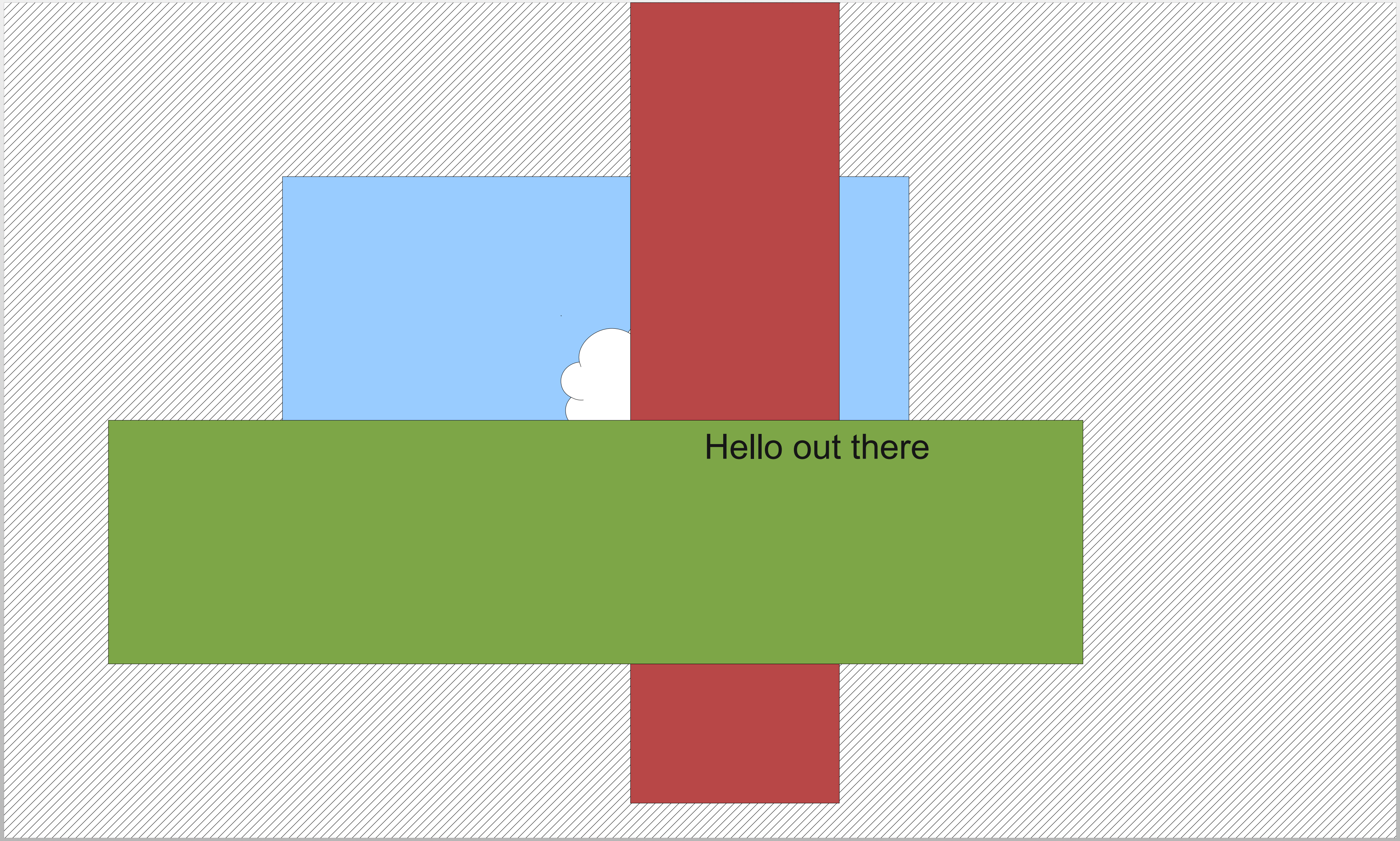
Automated update handling

Only draw updated regions (up to each engine to implement)



Automated update handling

Result



Multiple output paths

- Pure software
  - Universal (works everywhere)
  - MMX, SSE, SSE3, NEON ASM (runtime detected)
  - High quality scaling (super-sampling + linear-interpolation)
  - Caching of scaled image data on the fly
  - Output rotation and down-convert

Multiple output paths

- OpenGL/OpenGL-ES2
  - Uses texture atlases where possible
  - Defers texture upload and removes duplication where it can
  - Multi-pipeline out-of-order rendering optimizing
  - Batches up as much geometry as it can for best performance
  - Specialized shaders for performance
  - Pushes all rendering via GL (not just compositing surfaces)
    - Text, polygons too
  - Tries to remove texture uploads with zero-copy (if possible)

Multiple output paths

- X11 (OpenGL, Xlib & XCB)
- Wayland (OpenGL & SHM)
- Raw Framebuffer
- Memory buffers
- PS3 Native
- SDL (OpenGL)
- Windows (32/64/CE) (GDI & DirectDraw)
- ... others too

Input Data

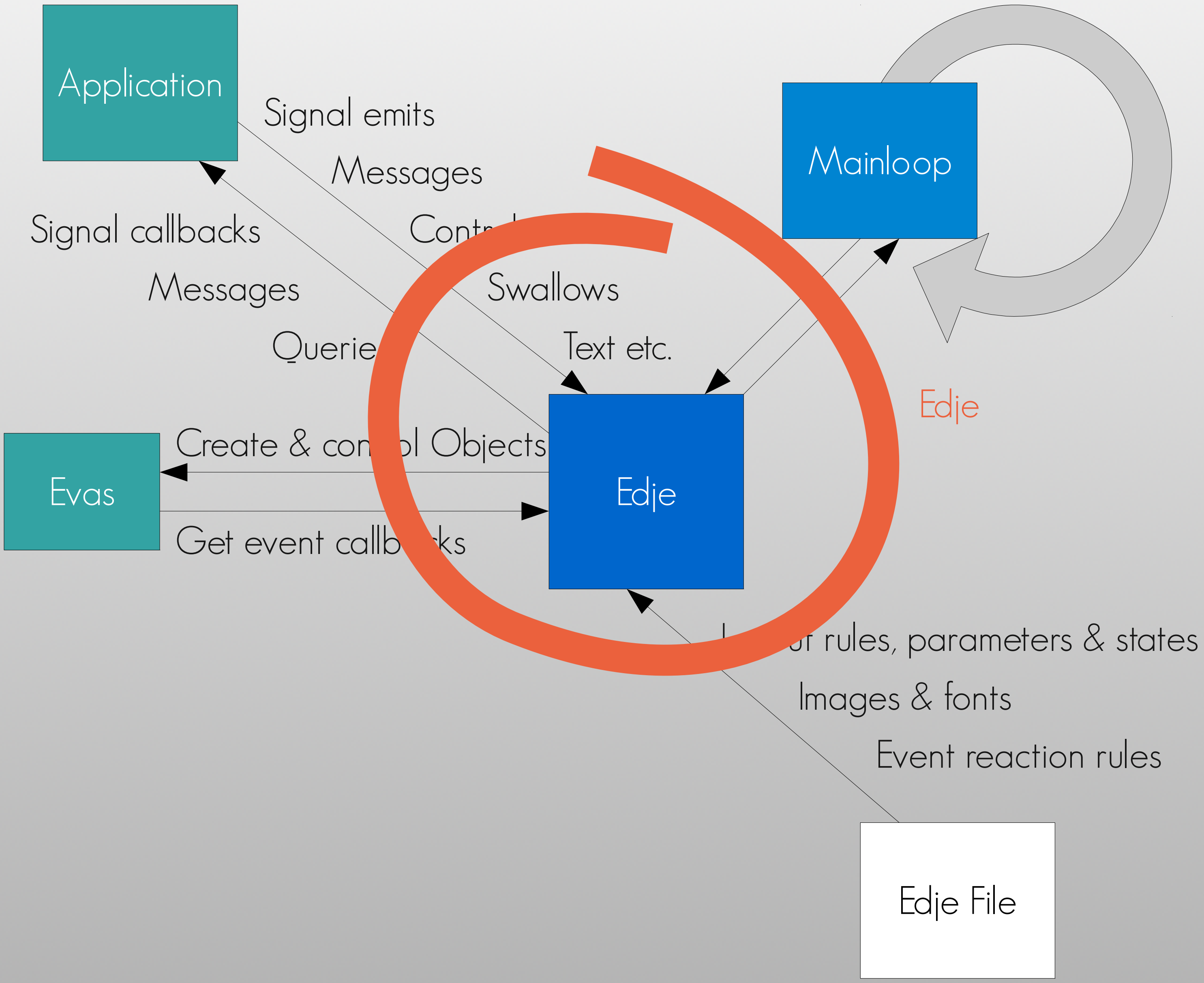
- Images
  - CMP, EDB, EET, GIF (animated + still), ICO, JPEG, PPM/PGM/PBM, PSD, SVG, TGA, TIFF, WBMP, XPM, XCF, PS, PDF, RAW, MOV/AVI/MPG/etc.
- Fonts
  - TTF, OpenType (anything Freetype 2 supports)
- Text
  - UTF-8 Unicode
  - Complex text formatting (LTR, RTL, Composition)

EDJE

Pre-made objects for designers (Edje)

- Edje allows a designer to store objects in files
  - Pre-made layout with rules and reactions to events
  - Stored separately to code in binary files for runtime replacement
  - Fast & compact random access designed for realtime use
  - All layout, image data, etc. etc. all in 1 file (zero-unpacking)
  - Intended for designers & developers to work independently
  - Supports scalable and resizeable layouts
  - Provides the core ability to re-theme and entire UI or OS

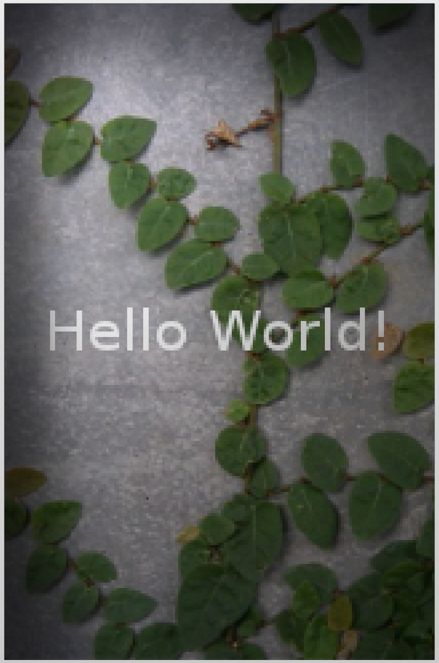
How it works

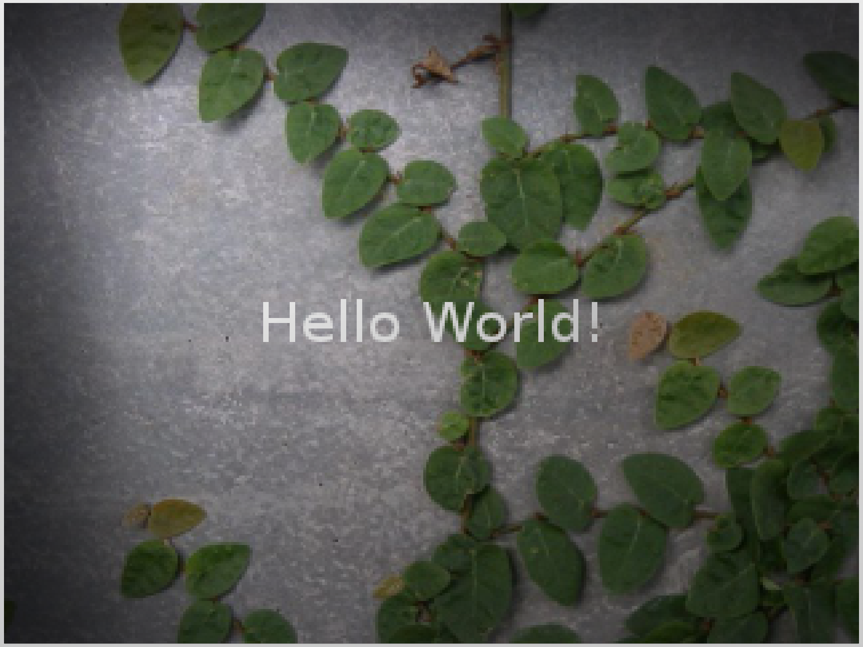

Application
Signal emits
Messages
Contr l
Swallows
Text etc.
Signal callbacks
Messages
Querie
Mainloop
Edje
Evas
Create & con ol Objects
Get event callb ks
Edje
l r rules, parameters & states
Images & fonts
Event reaction rules
Edje File

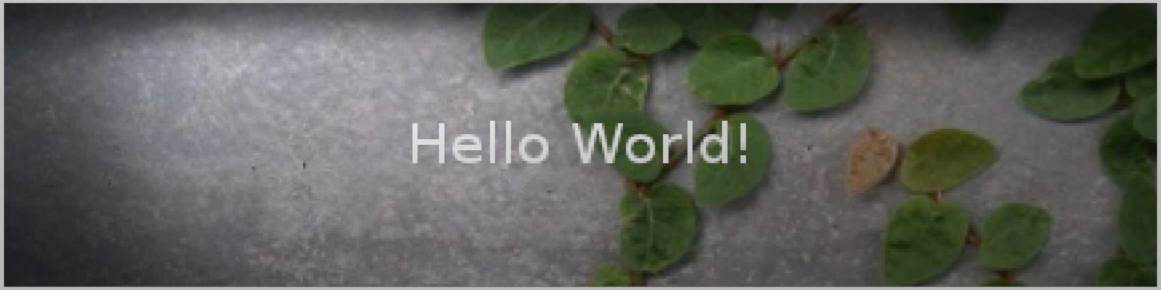
An example

```
collections {
   group { name: "hello";
      images {
         image: "plant.jpg" LOSSY 80;
         image: "shadow.png" COMP;
      }
      parts {
         part { name: "bg";
            description { state: "default" 0.0;
               aspect: 1.0 1.0; aspect_preference: NONE;
               image.normal: "plant.jpg";
            }
         }
         part { name: "label"; type: TEXT; scale: true;
            description { state: "default" 0.0;
               text {
                  font: "Sans"; size: 20;
                  text: "Hello World!";
               }
            }
         }
         part { name: "shadow";
            description { state: "default" 0.0;
               image.normal: "shadow.png";
            }
         }
      }
   }
}
```

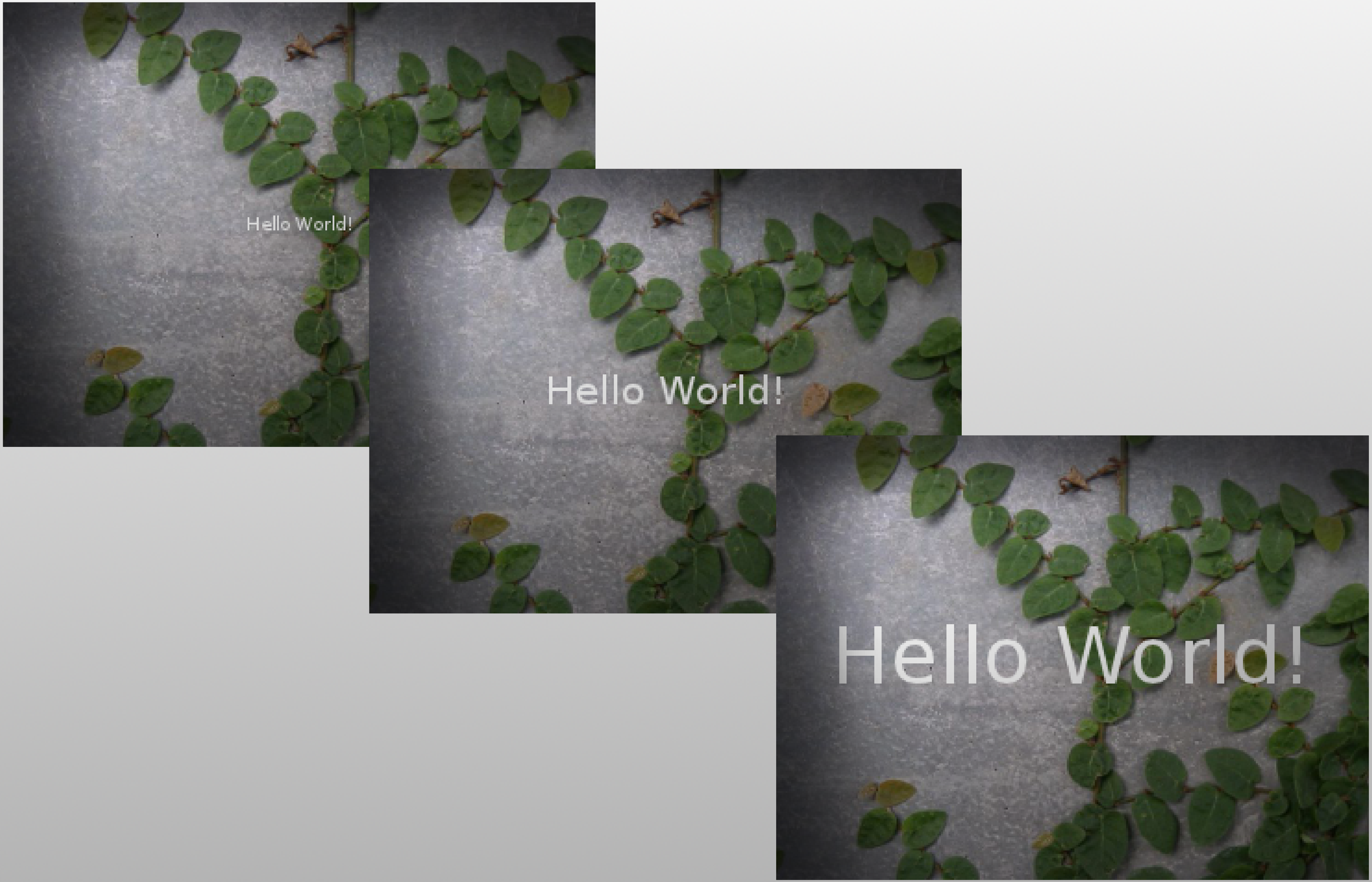
Sizing







Scaling



ELEMENTARY

So what is Elementary?

- A widget set built on top of the lower-level EFL layers
- Brings coherent policy and consistency to widgets
- Pre-made common widgets most applications need
- Central theme setup so applications look consistent
- Utilities saving extra footwork by the developer
- Touch friendly design
- Scaling of UI from the get-go
- Also adjusts for input resolution (finger vs mouse etc.)

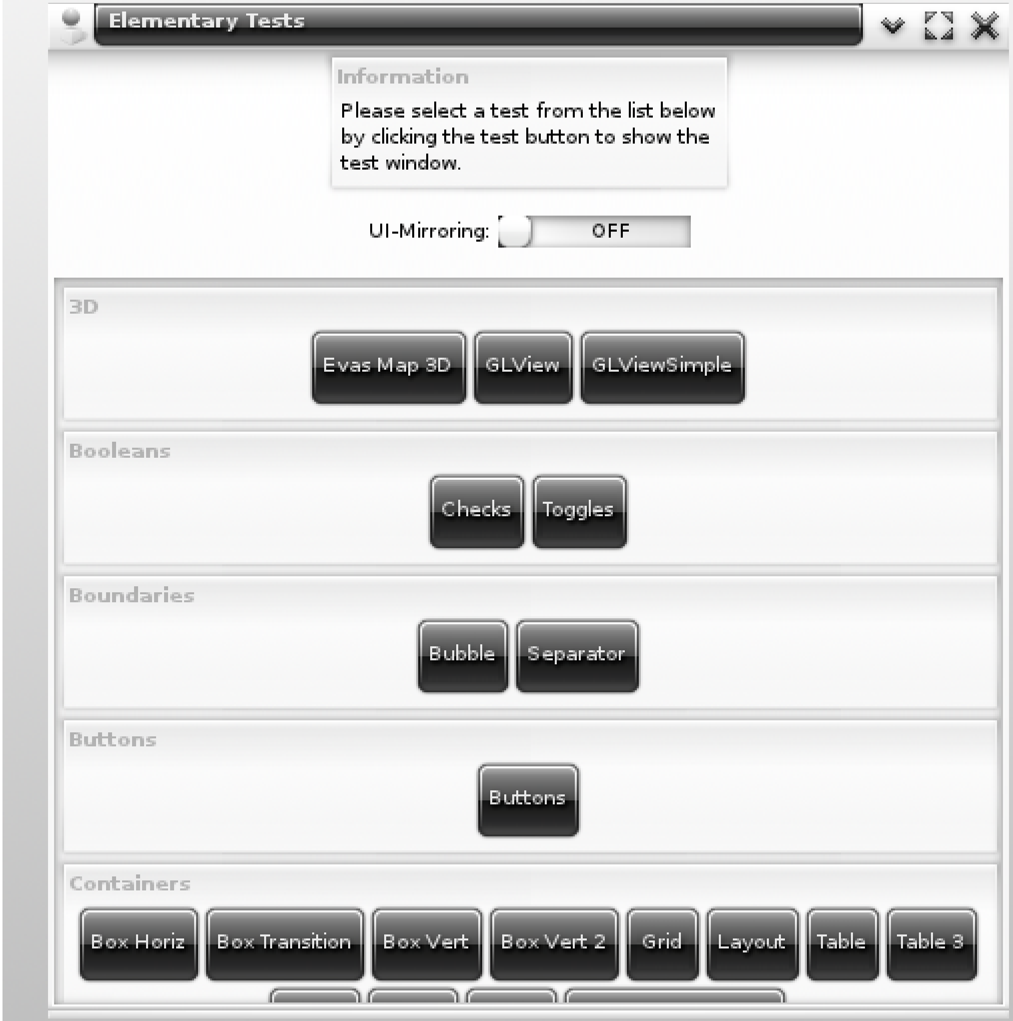
So what is Elementary?

- It can be seamlessly merged with lower level objects
- Programmer can use Elementary containers or hand-arrange widgets and control them
- Since all objects can be stacked and layered, so can elementary widgets
- Widgets can be transformed like any object
- Handles dealing with IME (Virtual keyboard) for you
- Does many other useful things to save you time

So what is Elementary?

- All widgets can and will be rendered with Hardware (OpenGL) if that is the engine chosen
  - This includes all decorations, text etc. not just compositing
- Elementary helps enforce "finger size" so users can always easily use your application
- Works on both desktop (keyboard & mouse) as well as touchscreen and multi-touch

Results with Elementary

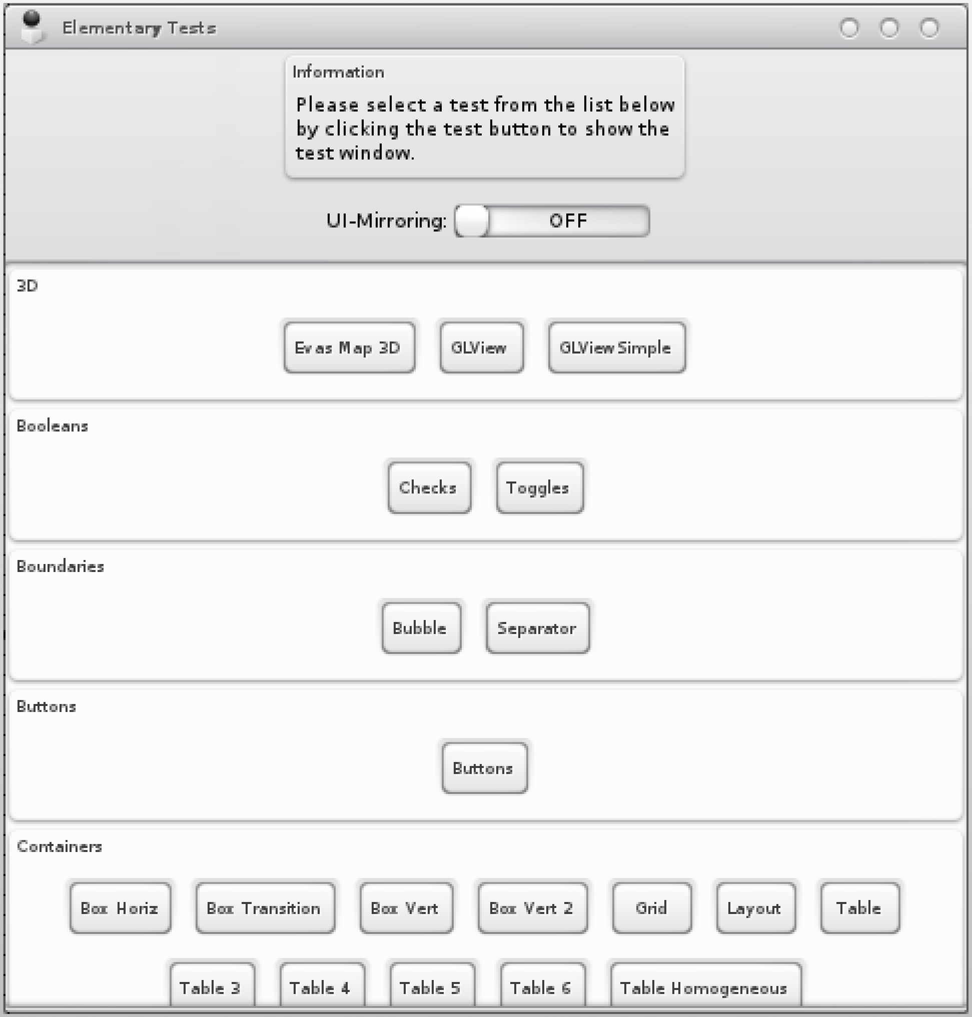


Elementary Tests

Information

Please select a test from the list below by clicking the test button to show the test window.

UI-Mirroring: OFF



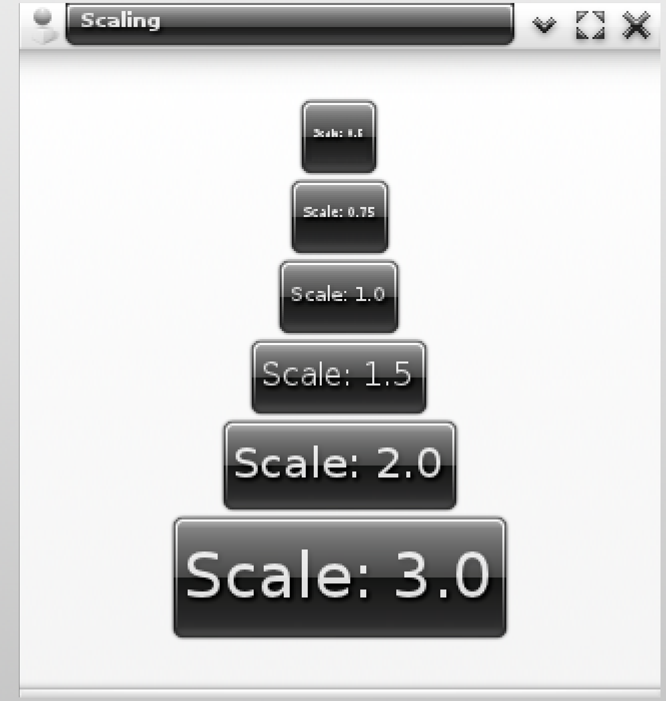
Elementary Tests

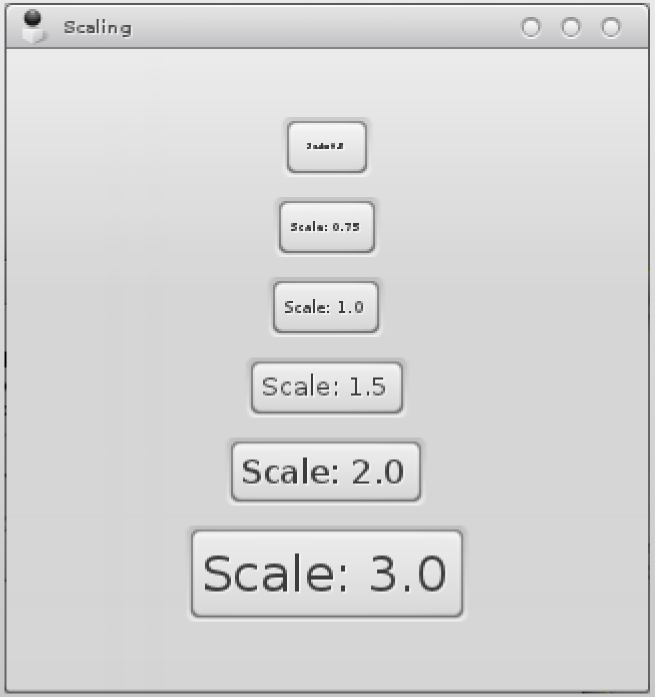
Information

Please select a test from the list below by clicking the test button to show the test window.

UI-Mirroring: OFF

Results with Elementary





EMOTION

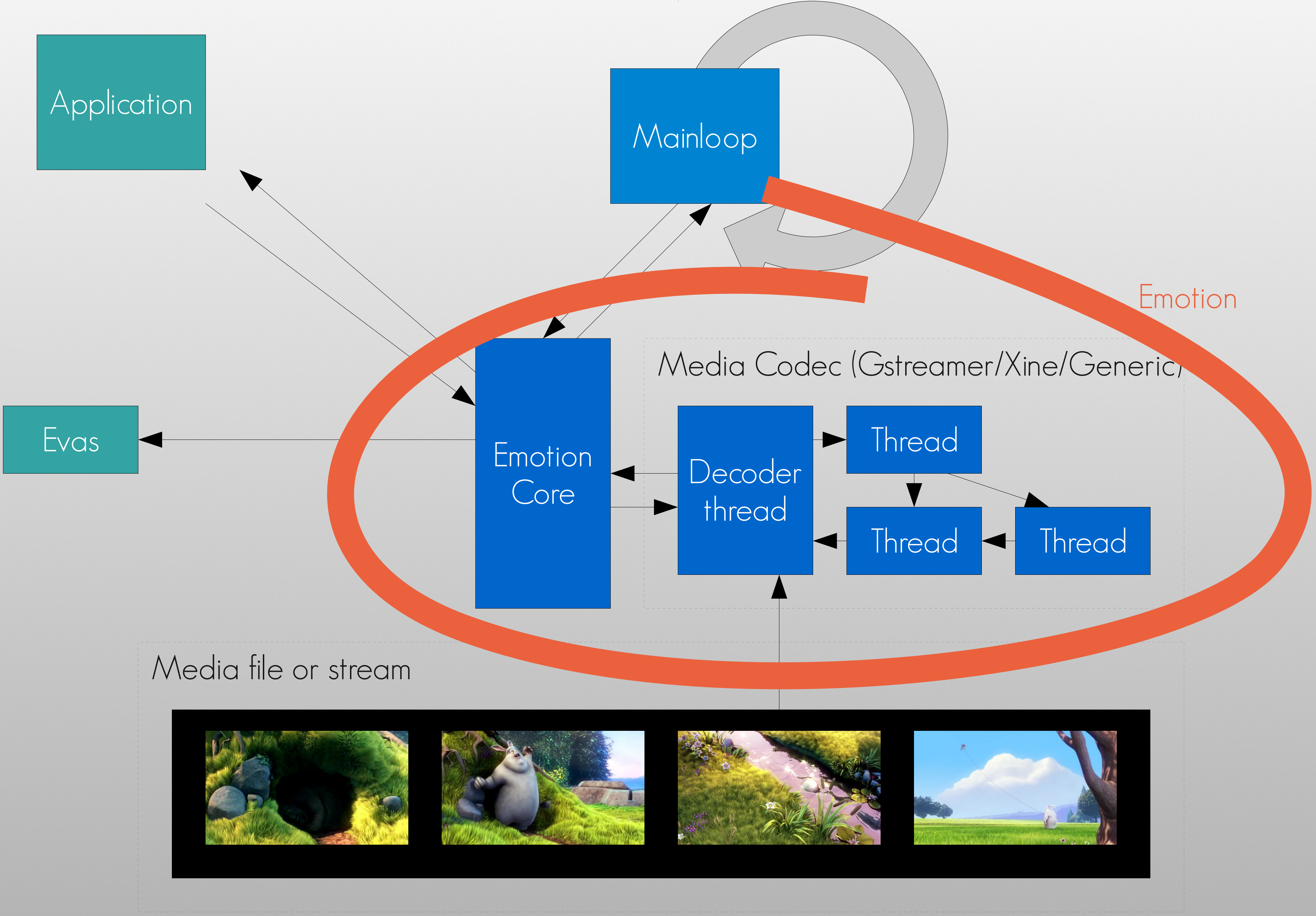
Video & Sound in your world

- Gives you a high level API to include video
- Abstracts to different video decode back-ends
- Optimizes decode via YUV paths or video overlay
- Simple to use

Simple Video

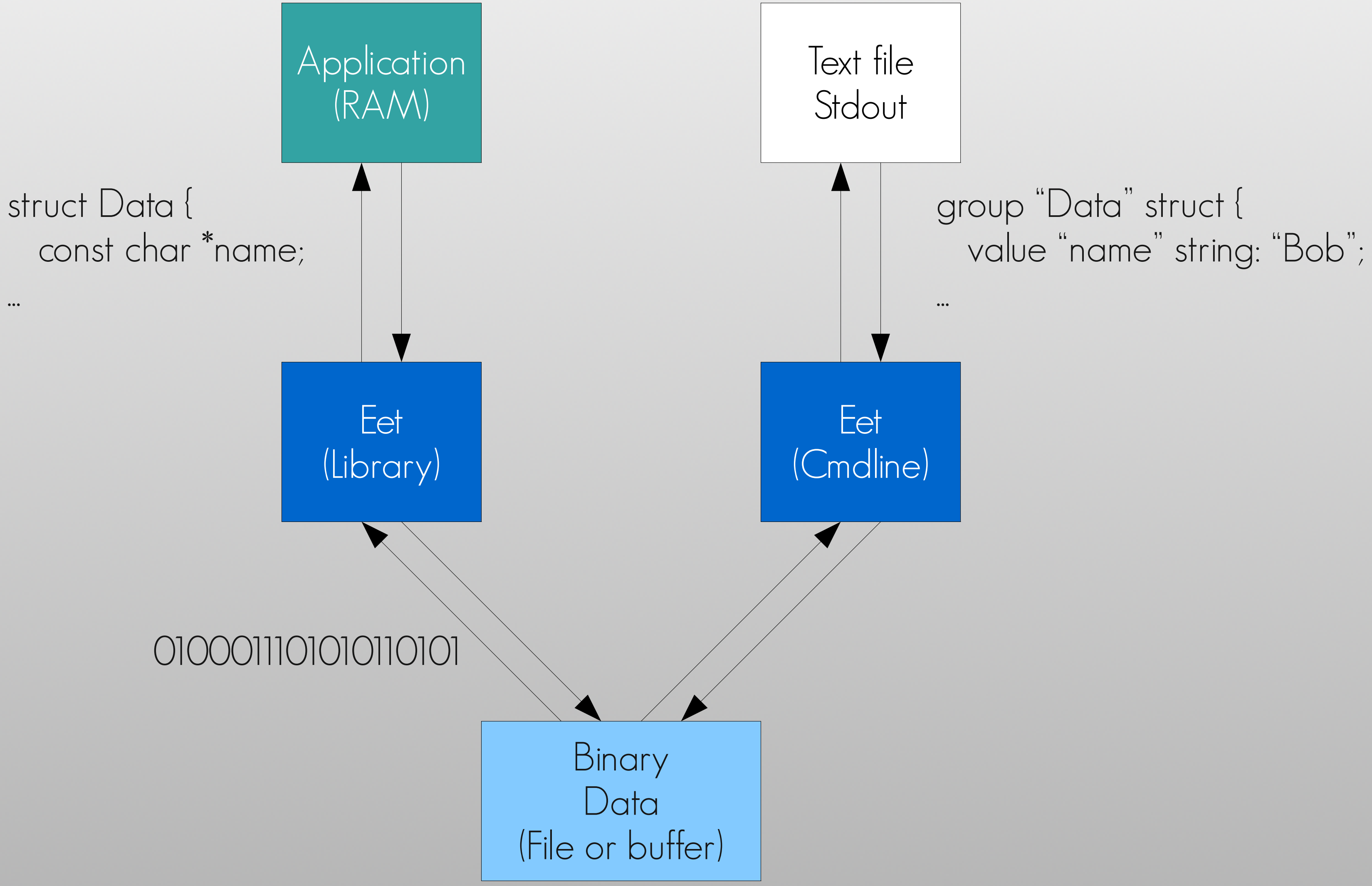
```
Evas_Object *vid = emotion_object_add(canvas);
emotion_object_init(vid, NULL);
emotion_object_file_set(vid, "file.avi");
evas_object_resize(vid, 640, 360);
emotion_object_play_set(vid, EINA_TRUE);
evas_object_show(vid);
```

How it works


Application
Mainloop
Emotion
Media Codec (Gstreamer/Xine/Generic)
Evas
Emotion Core
Decoder thread
Thread
Thread
Thread
Media file or stream

EET

Garbage in, garbage out



Application
(RAM)

Text file
Stdout

```
struct Data {
   const char *name;
...
```

```
group "Data" struct {
   value "name" string: "Bob";
...
```

Eet
(Library)

Eet
(Cmdline)

01000111010101110101

Binary
Data
(File or buffer)

XML/JSON ... for C programmers

- Parsing text is painful
  - Parsing correctly without bugs, overflows is harder
  - Most programmers hate parsing
- XML, JSON etc. optimized for editing, not runtime
- Programs read/write data 1000x more than humans
  - So optimize for the common use case, not the uncommon one
- Make it as easy 1-liners for C code to load or save data
- Edje, Enlightenment config, Elementary config built on EET

Flexible, portable and robust

- Allows you to store data in a file (key/value pair)
  - Random access read optimized
  - Data can be any binary, image, string or struct encoded
  - Compresses separate keys (like zip)
- Allows you to en/decode structs to buffers (for network)
- Provides a protocol buffer handler for decodes
- Files and data all platform agnostic (portable)
- Structs encoded with nested key & types for robustness

Define your data structure

```
typedef struct {
   const char *name;
   int age;
   const char *address;
   const char *phone;
} Data;

Eet_Data_Descriptor_Class dc;
Eet_Data_Descriptor *d;

EET_EINA_FILE_DATA_DESCRIPTOR_CLASS_SET(&dc, Data);
d = eet_data_descriptor_file_new(&dc);
EET_DATA_DESCRIPTOR_ADD_BASIC(d, Data, "name", name, EET_T_STRING);
EET_DATA_DESCRIPTOR_ADD_BASIC(d, Data, "age", age, EET_T_INT);
EET_DATA_DESCRIPTOR_ADD_BASIC(d, Data, "address", address, EET_T_STRING);
EET_DATA_DESCRIPTOR_ADD_BASIC(d, Data, "phone", phone, EET_T_STRING);
```

Declare and save it

```
Eet_File *ef;

Data data = {
    .name = "Bob the blob",
    .age = 7,
    .address = "7 Blob ave.",
    .phone = "+82 10 123 4567"
};

ef = eet_open("data.eet", EET_FILE_MODE_WRITE);
eet_data_write(ef, d, "data", &data, EINA_TRUE);
eet_close(ef);
```

Declare and save it

```
Data *data2;
ef = eet_open("data.eet", EET_FILE_MODE_READ);
data2 = eet_data_read(ef, d, "data");
eet_close(ef);
```

For debugging, decode to text

```
$ eet -d data.eet data
group "Data" struct {
    value "name" string: "Bob the blob";
    value "age" int: 7;
    value "address" string: "7 Blob ave.";
    value "phone" string: "+82 10 123 4567";
}
```

And encode + compress from text

```
$ cat data.txt
group "Data" struct {
    value "name" string: "Bob the blob";
    value "age" int: 7;
    value "address" string: "7 Blob ave.";
    value "phone" string: "+82 10 123 4567";
}
$ eet -e data.eet data data.txt  1
```

SAMSUNG

EDBUS EFREET EINA ETHUMB EEZE EMBRYO EIO ...

And the saga continues

- More EFL libraries with no time to mention them
- Expanding libs and scope on a daily basis

Questions, Answers & Flaming

Enlightenment Foundation Libraries
http://www.enlightenment.org

Carsten Haitzler
Enlightenment project lead & founder
Principal Engineer
raster@rasterman.com
c.haitzler@samsung.com