Building Exciting User Interfaces on new Tizen Platform with EFL/DALi

Govindaraju S M
Samsung R&D Institute, Bangalore
Tools to realize your App Ideas

Features  
Ease of Use  
Performance  
Light Weight
Key take away from this session

- Basics & Features
- How to create UI?
- What to Choose?

Coverage
- Tizen Native UI Framework Architecture
- EFL Features & APIs
- EFL Sample App & Demo
- DALi Features, Sample App
- Summary
Tizen Native UI Framework - Subsystems

Applications

- **EFL App**
  - EFL
  - Cairo
  - OpenGL ES / EGL
- **DALi App**
  - DALi
  - Font (Fontconfig, freetype2)
  - Windowing System (X11)
  - Window Manager & Compositor (E17)
  - ISF, Voice FW (isf, ise, STT, TTS)

**Base Operating System / display etc.** (Linux)
Tizen Native UI Framework

2D ~ 2.5D UI Framework
- A collection of libraries supporting 2D ~ 2.5D UI
- Splits GUI Design & Functionality
- Enlightenment Open source Project: http://www.enlightenment.org

3D Scene Graph UI Rendering Engine
- Open GL ES 2.0 Shaders, Vertices and Textures based Rendering and Effects
- 2D world is the Z plane 0 in the 3D world
Enlightenment Foundation Libraries (EFL)
EFL Architecture

Applications

EFL

Bindings

Elementary

Edje

Efreet

Eio

Eeze

EDbus

Ethumb

Emotion

Ecore

Evas

Eina

Embryo

Eet

Base Operating System / display etc. (Linux)
Evas : (E + CanVAS)

Canvas Library

Rendering Objects
Evas Scene Graph

Hierarchy setup with parent and child objects
Tracks state of objects
Handles rendering of each object
Retained Rendering Mode

No “rendering” API – Scene Graph

Application

Build Scene (UI layout)

EFL

Update

Drawing Commands

Scene (Model)

Primitives Object

- Rectangle
- Line
- Polygon
- Text
- Textblock
- Textgrid
- Image
- Vector Graphic

Smart Object

- box
- grid
- table
- customized smart objects (elm widgets)
Evas Map - 3D Transformation Interface

Transformations: Rotation, Zoom, 3D Perspective

Texture mapping

[Image of maps and texture mapping process]
Evas VG

Vector Graphics objects Scene graph
Eina is a library for data types and some useful tools.

- eina_list
- eina_file
- eina_array
- eina_stringshare
- eina_rbtree
- eina_benachmark
- eina_log
- eina_mempool
- eina_module
Ecore (E + Core)

Clean and tiny event loop library with many convenience modules

Setup → Idler → FD Handler calls → Event Handlers → Timers, Pollers, Animators

Ecore - Mainloop
Ecore

- Ecore Main loop
- Ecore Thread
- Ecore Pipe
- Ecore Timer
- Ecore Job

- Ecore Idler
- Ecore Animator
- Ecore Audio
- Ecore IMF
Edje

- **EFL Application**
  - Edje text script file (.edc) / GUI Layout binary (.edj)
  - Executable logic binary (C)

- **Edje**: Graphical design and layout library

- **Edje Data Collection (EDC)**: Layout Script, Enables UI layout development outside C code
Scalable UI

- Saleable for multi-size/resolution/aspect-ratio of screens
- Continuous scaling based on a scale value
- Screen coordinates are mapped between 0 & 1.

**Full HD 60”**  
**WXGA 10.1”**  
**HD 4.65”**  
**WVGA 3.8”**  
**HVGA 3.27”**
collections {
  group {
    name: "demo_group";
    images.image: "ball.png" COMP;
    parts{
      part{
        name: "part_image";
        type: IMAGE;

        description{ state: "default" 0.0;
          rel1.relative: 0.1 0.1;
          rel2.relative: 0.3 0.3;
          image{
            normal: "ball.png";
          }
        }

        description{ state: "bottom" 0.0;
          inherit: "default" 0.0;
          rel1.relative: 0.9 0.9;
          rel2.relative: 1.0 1.0;
        }
      }
    }
  }
}

program { name: "start";
  signal: "show";
  action: STATE_SET "bottom" 0.0;
  target: "part_image";
  transition: ACCELERATE 0.5;
  after: "bounce";
}

program { name: "bounce";
  action: STATE_SET "default" 0.0;
  target: "part_image";
  transition: DECELERATE 1.0;
  after: "start";
}
Elementary : Widget Set Library

Window & Conformant

Page Control

NaviFrame

Gengrid

Color Selector

Button

Radio

Date Time

Tool Bar

Edit Field

Progress Bar

MultiButtonEntry

Map

Slider

Ctx Popup

Popup
//Create window
Evas_Object *win = elm_win_add(NULL, "hello", ELM_WIN_BASIC);
elm_win_title_set(win, "Hello");

//Create a layout object
Evas_Object* ly = elm_layout_add(win);

//Load an edj file
elm_layout_file_set(ly, "sample.edj", "group_name");
...
evas_object_show(ly);
How to combine an Edje with C?

Send a signal from C to Edje

```c
void fn(Evas_Object *parent) {
    Evas_Object *ly = elm_layout_add(parent);
    //...
    elm_object_signal_emit(ly, "elm,state,test_part,hidden", "elm");
}
```

```edc
... part {
    name: "test_part";
    //...
    //show state
    description {
        name: "show" 0.0;
        visible: 1;
    }
    //hide state
    description {
        name: "hide" 0.0
        visible: 0;
    }
}
```

```
programs {
    program {
        name: "hide_test_part";
        signal: "elm,state,test_part,hidden"; //signal
        source: "elm";
        target: "test_part"; //action target
        action: STATE_SET "hide" 0.0; //action definition
    }
    ...
}
```
How to combine an Edje with C?

Send a signal from Edje to C

```c
static void _cb( void *data,
    Evas_Object *obj,
    const char *emission,
    const char *source)
{
    //...
}

static void fn(Evas_Object *parent) {
    Evas_Object *ly = elm_layout_add(parent);
    //...
    elm_object_signal_callback_add( ly, “clicked”, “*”, _cb, NULL);
}
```

```edc
//...
parts {
    part {
        name: “test_part”;
        //...
    }
    //...
}

programs {
    program {
        name: “clicked”;
        source: “test_part”;
        signal: “mouse,clicked,1”;
        action: SIGNAL_EMIT “clicked” “*”;
    }
    //...
    //...
}
```

How to combine an Edje with C?

Send a signal from Edje to C

C Program (.c)

Signal Emit

Mouse Clicked Event

GUI Layout .edj

Evas Object

signal callback _cb

Clicked program
Elementary Application : Widgets & Layout

#include <Elementary.h>

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

    // Create Window
    win = elm_win_util_standard_add("Name", "Win Title");

    // Add Layout
    layout = elm_layout_add(win);
    elm_layout_file_set(layout, "layout.edj", "demo_group");
    elm_win_resize_object_add(win, layout);

    // Create Controls
    btn = elm_button_add(win);
    elm_object_text_set(btn, "Click me!");
    elm_object_part_content_set(layout, "part_swallow", btn);

    // Show the Controls & Window
    evas_object_show(layout);
    evas_object_show(win);

    // Blocking mainloop: process events and callbacks
    elm_run();
    elm_shutdown();
    return 0;
}

ELM_MAIN()
collections {
  group {
    name: "demo_group";
    parts {
      part {
        name: "bg";
        ...
      }
      part{
        name: "part_swallow";
        type: SWALLOW;
        description{
          state: "default" 0.0;
          max: 200 100;
          ...
          map{
            on: 1;
            perspective_on: 1;
            rotation {
              y: 360;
            }
          }
        }
        description{ state: "bottom" 0.0;
          inherit: "default" 0.0;
          ...
          map {
            rotation { y: 360; }
          }
        }
      }
    }
  }
  ...
}

program { name: "start";
  signal: "show";
  action: STATE_SET "bottom" 0.0;
  target: "part_swallow";
  transition: LINEAR 3.0;
  after: "bounce";
}

program { name: "bounce";
  action: STATE_SET "default" 0.0;
  target: "part_swallow";
  transition: LINEAR 3.0;
  after: "start";
}

TIZEN DEVELOPER SUMMIT 2015 BENGALURU
Theme : Priority

System based Theme (Basic Themes)
- ELM_THEME = tizen:tizen2:tizen3

Application based Theme
- Extension Themes (lower than Basic Themes)
  - elm_theme_extension_add();
  - elm_theme_extension_del();
- Overlay Themes (higher than Basic Themes)
  - elm_theme_overlay_add();
  - elm_theme_overlay_del();
Custom 2D/3D Rendering on EFL

- 2D / Vector Rendering: use Cairo + Evas Image
- Elm GLView simplifies 3D rendering
Dynamic Animation Library (DALi)
DALi

3D Animation/effect based UIFW library

World Origin (0, 0, 0)

+ve X
+ve Y
+ve Z

Default Camera

3D Dali World

Demo4 Dali_Features
DALi Architecture

Applications

DALi API (C++, Java Script)

DALi Toolkit
- UI Controls
- Shaders
- 3D Animation
- Scene Graph
- Styling
- Layout
- Events
- Rendering

DALi Adaptor
- EFL Plugin
- JS Binding
- X11 Adaptor

DALi
- Geometry
- 3D models

OpenGL ES

Future Plan, beyond Tizen 2.4
DALi Basics

- Scene graph based rendering
- **Stage** defines whole 3D world
- **Actors** handle input events
- Effects with **Shaders**
- 3D Model support

# Future Plan, beyond Tizen 2.4
DALi Actors

Image Actor

Camera Actor

Custom Actor

Text Actor

Light Actor

Model Actor
Constraints and Property Notifications

- **Constraint**: Property based on function of other properties
- **Property notification**: Notified on property reaches or crosses a value
Animation

- Property animation
- Vertex & Mesh animation
- Shader Uniform animation
- Model / Key frame animation
Effects: Modifies Object Appearance

Actor Default Shader: Override Geometry (vertex) / Pixels (fragment)
3D Models

- Collada, Maya®, 3DS Max®, DALi format, etc…
- Open Asset Import Library (assimp) Model Importer
UI Controls

Basic Building blocks: Image, Text and Mesh Actors

UI Controls provide additional Layout, Effects, Scrolling

- Image View
- Button, Pop-up
- Item-view
- Scroll-view
- Text-field
using namespace Dali::Toolkit;

#include <dali.h>

class DaliExample
{
  public:
    DaliExample(Application& mApplication)
    {
      mApplication.InitSignal().Connect(this,&DaliExample::OnInit);
    }

    virtual ~DaliExample() {}

    void OnInit(Application& app)
    {
      ResourceImage image = ResourceImage::New("Path");

      ImageView imageView = ImageView::New(image);

      Stage::GetCurrent().Add(imageView);
    }

  }

int main(int argc, char** argv)
{
  Application app = Application::New(&argc, &argv);
  DaliExample test(app);
  app.MainLoop();
  return 0;
}
EFL or DALi?

**EFL**
- 2D ~ 2.5D Effects
- Software or GPU based

**DALi**
- Rich 3D effects / Physics
- 3D Models
THANK YOU