Breakthrough Games with Tizen

Tizen Graphics

Samsung
Agenda

1. Introduction
2. Game Porting to Tizen
3. Tips for Development
4. Monetization
5. Summary
Introduction

• Market Status
  – 27 Games in TOP rank 50 (2015.06)
Why Tizen?

- Expandability and Convergence
Why Tizen?

• Unexplored and New Game Area

Multi-user Gaming Experience

Multiple screens Game Experience
Why Tizen?

- New devices, New marketplace, New opportunity
  - Hard to make your games visible to users

![Graph showing the growth of apps from 2010 to 2014.](image-url)
Why Tizen?

• Efficiency
  – Development efficiency
    • C-based modules accelerate porting of open source modules
Why Tizen?

• Efficiency
  – Device performance

  • Graphics is highly optimized, which is deeply impressive for mass model with limited resources
Why Tizen?

• Developer friendly Tizen Store Seller Promotion

100% is YOURS!
No need to share your revenue for one year!
Join us and Submit your Applications Now

PERIOD:
One Year (January 14 2015 ~ January 31 2016)
Game Porting to Tizen

• Typical way
  – Develop game for Tizen with the same scenario
Game Porting to Tizen

• Better way
  – Divide porting layer from game core, and adapt only porting layer
Game Porting to Tizen

• Best way
  – Adopt game engines, such as cocos2d-x & Unity3D
Game Porting to Tizen

- Famous Game Engines are ready for Tizen
  - Cocos2d-x (since ver.3.5.1)
  - Unity3D (since ver.5.1)
Tips for Development - evasgl basics (1)

- EFL (Enlightenment Foundation Libraries)
  - Collection of open source libraries from Enlightenment
- evas (Efl + canVAS)
  - evas is Scene Graph composed of ‘evas objects’
• EFL View Hierarchy
• GPU Accelerated Rendering in EFL
  – How to make a surface for GLES?
  – How the surface is composited with other widgets?

• evasgl
  – Abstraction for EGL and OpenGL-ES
    • EGL related operations are automatically and internally processed in evas
    • Provides wrappers for the native OpenGL-ES calls
  – Rendering results by evasgl goes to evas object
    • All evas objects are smoothly composited in EFL view hierarchy
evasgl basics (4)

- Revisit EFL View Hierarchy
Sameple code – Draw one cube

• Overall sequence of sample code
  • Application initialization
  • evasgl initialization
  • Animation and rendering settings
  • Add animator and renderer to ecore main loop
  • Define rendering with evasgl functions
1. Application Initialization

```c
#include <Elementary.h>
#include <Evas_GL.h>
......

// Define a global context for the application
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *img;
    Evas_GL *evasgl;
    Evas_GL_API *glapi;
    Evas_GL_Context *ctx;
    Evas_GL_Surface *sfc;
    Evas_GL_Config *cfg;
    unsigned int program;
    unsigned int vtx_shader;
    unsigned int fgmt_shader;
    unsigned int vbo;
} appdata_s;

app_main part

int main(int argc, char *argv[])
{
    appdata_s ad = {0,};
    int ret = 0;

    ui_app_lifecycle_callback_s event_callback = {0,};
    ......
    event_callback.create = app_create;
    event_callback.terminate = app_terminate;
    event_callback.pause = app_pause;
    event_callback.resume = app_resume;
    event_callback.app_control = app_control;

    ......
    ret = ui_app_main(argc, argv, &event_callback, &ad);
    return ret;
}
```
2. evasgl Initialization

/* Set config of the surface for evas gl */
ad->cfg = evas_gl_config_new();
ad->cfg->color_format = EVAS_GL_RGBA_8888; // Surface Color Format
ad->cfg->depth_bits = EVAS_GL_DEPTH_BIT_24; // Surface Depth Format
ad->cfg->stencil_bits = EVAS_GL_STENCIL_NONE; // Surface Stencil Format
ad->cfg->options_bits = EVAS_GL_OPTIONS_NONE; // Configuration options (here, no extra options)

/* Add Window */
ad->win = elm_win_util_standard_add("Evas_GL Example", "Evas_Gl Example");

/* Get the evas gl handle for doing gl things */
ad->evasgl = evas_gl_new(evas_object_evas_get(ad->win));
ad->glapi = evas_gl_api_get(ad->evasgl);

/* Get the window size */
Evas_Coord w, h;
evas_object_geometry_get(ad->win, NULL, NULL, &w, &h);

/* Create a surface and context */
ad->sfc = evas_gl_surface_create(ad->evasgl, ad->cfg, w, h);
ad->ctx = evas_gl_context_create(ad->evasgl, NULL);

/* Initialization GLES including shader generation and other stuffs */
......
3. Animation and Rendering setting

Animation and Rendering

/* Set up the image object. A filled one by default. */

ad->img = evas_object_image_filled_add(evas_object_evas_get(ad->win));
ev_object_image_pixels_get_callback_set(ad->img, img_pixels_get_cb, ad);

/* Add Event Callbacks */
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_DEL, img_del_cb, ad);
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_MOUSE_DOWN, mouse_down_cb, ad);
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_MOUSE_UP, mouse_up_cb, ad);
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_MOUSE_MOVE, mouse_move_cb, ad);
evas_object_event_callback_add(ad->win, EVAS_CALLBACK_RESIZE, win_resize_cb, ad);

/* Add animator */
ani = ecore_animator_add/animate_cb, ad->img);

static Eina_Bool
animate_cb(void *data)
{
  Evas_Object *img = data;
  evas_object_image_pixels_dirty_set(img, EINA_TRUE);
  return ECORE_CALLBACK_RENEW;
}

static void
img_pixels_get_cb(void *data, Evas_Object *obj)
{
  appdata_s *ad = data;
  Evas_GL_API *gl = ad->glapi;

  // Rendering process here
  .......
}
Sameple code – Draw one cube

- EFL View Hierarchy for evasgl initialization

GL Application

- evasgl commands with `ad->evasgl`

- `ad->sfc`
- `ad->img`
- evas object

- `ad->win`
- evas object

- evas
  - evas_object_evas_get(ad->win)

- X-window
3. Animation and Rendering setting

/* Set up the image object. A filled one by default. */
ad->img = evas_object_image_filled_add(evas_object_evas_get(ad->win));

evas_object_image_pixels_get_callback_set(ad->img, img_pixels_get_cb, ad);

/* Add Event Callbacks */
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_DEL, img_del_cb, ad);
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_MOUSE_DOWN, mouse_down_cb, ad);
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_MOUSE_UP, mouse_up_cb, ad);
evas_object_event_callback_add(ad->img, EVAS_CALLBACK_MOUSE_MOVE, mouse_move_cb, ad);
evas_object_event_callback_add(ad->win, EVAS_CALLBACK_RESIZE, win_resize_cb, ad);

/* Add animator */
ani = ecore_animator_add(animate_cb, ad->img);

static EINA_Bool animate_cb(void *data)
{
    Evas_Object *img = data;
evas_object_image_pixels_dirty_set(img, EINA_TRUE);
    return ECORE_CALLBACK_RENEW;
}

static void
img_pixels_get_cb(void *data, Evas_Object *obj)
{
    appdata_s *ad = data;
    Evas_GL_API *gl = ad->glapi;
    // Rendering process here
    .......
}
4. Rendering with evasgl

static void
img_pixels_get_cb(void *data, Evas_Object *obj)
{
    appdata_s *ad = data;
    Evas_GL_API *gl = ad->glapi;
    .......

    /* Make the application context as current */
    evas_gl_make_current(ad->evasgl, ad->sfc, ad->ctx);

    /* Render the scene with evasgl functions */
    gl->glViewport(0, 0, WIDTH, HEIGHT);
    gl->glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    gl->glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gl->glUseProgram(ad->program);
    gl->glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(float) * 6, 0);
    gl->glEnableVertexAttribArray(0);
    gl->glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, sizeof(float) * 6, (void*)(sizeof(float)*3));
    gl->glEnableVertexAttribArray(1);
    .......
}
[Caution] Context Handling

- GLES context maintaining with `evas_gl_make_current`

GL Application

- `evasgl` rendering with app’s own context
- `evas` compositing with `evas`'s own context (Hidden from developers)

`evasgl` commands with `ad->evasgl`

- `ad->sfc`
- `ad->img`
- `evas object`

- `ad->win`
- `evas object`
- `evas` ➔ `evas_object_evas_get(ad->win)`

- `X-window`
• Elementary widget specialized for evasgl rendering
  – Preset tedious work for evasgl rendering for developers
    • Comparable to android.opengl.GLSurfaceView
    • Help developers to focus on only rendering task
  – What does elm_glview work for you?
    • Context & Drawable Surface generation
    • Setup all required callbacks including all useful events, such as touch and rendering
    • Guarantee the context maintaining automatically
    • Preset all necessary EGL properties according to the user input requirements
      (⇒ elm_glview_mode_set)
Sample Code – Change Initialization

```c
EAPI int elm_main(int argc EINA_UNUSED, char **argv EINA_UNUSED)
{
    Evas_Object *win;
    Evas_Object *glview;
    ......

    win = elm_win_util_standard_add("glview", "GLView");
    evas_object_show(win);

    /* Initialize & Setup elm_glview */
    {
        glview = elm_glview_add(win);
        elm_win_resize_object_add(win, glview);

        elm_glview_mode_set(glview, ELM_GLVIEW_ALPHA | ELM_GLVIEW_DEPTH);
        elm_glview_resize_policy_set(glview, ELM_GLVIEW_RESIZE_POLICY_RECREATE);
        elm_glview_render_policy_set(glview, ELM_GLVIEW_RENDER_POLICY_ON_DEMAND);

        elm_glview_init_func_set(glview, _init_gl);
        elm_glview_del_func_set(glview, _del_gl);
        elm_glview_render_func_set(glview, _draw_gl);
        elm_glview_resize_func_set(glview, _resize_gl);

        evas_object_size_hint_min_set(glview, 250, 250);
        evas_object_show(glview);
    }
    ......
}
```
Performance Improvement (1): DIRECT mode

• EFL View Hierarchy of full-screen GLES application

GL Application-Indirect mode

- Buffer for off-screen surface: 1st write operation
  - ad->sfc
  - ad->img
  - evas object

- Buffer for window surface: 2nd write operation
  - ad->win
  - evas object
  - evas
  - evas_object_evas_get(ad->win)

- X-window
  - evasgl commands with ad->evasgl
Performance Improvement (1): DIRECT mode

- EFL View Hierarchy of full-screen GLES application

GL Application-Direct mode

Evasgl commands with ad->evasgl

Buffer for off-screen surface: 1st write operation

- ad->sfc
- ad->img
- evas object

Buffer for window surface: THE ONLY write operation

- ad->win
- evas object
- evas

⇒ evas_object_evas_get(ad->win)

X-window
Sample Code – Change Initialization

```
EAPI int elm_main(int argc EINA_UNUSED, char **argv EINA_UNUSED)
{
    ....
    /* Initialize & Setup elm_glview */
    {
        glview = elm_glview_add(win);
        elm_win_resize_object_add(win, glview);
        elm_glview_mode_set(glview, ELM_GLVIEW_ALPHA | ELM_GLVIEW_DEPTH | ELM_GLVIEW_DIRECT);
    ....
}
```

evasgl case

```
/* Set config of the surface for evas gl */
......
ad->cfg = evas_gl_config_new();
ad->cfg->options_bits = EVAS_GL_OPTIONS_DIRECT; // Configuration options (here, DIRECT mode on)
......

/* Create a surface and context */
ad->sfc = evas_gl_surface_create(ad->evasgl, ad->cfg, w, h);
ad->ctx = evas_gl_context_create(ad->evasgl, NULL);
......
```
EAPI int elm_main(int argc EINA_UNUSED, char **argv EINA_UNUSED)
{
    ....
    /* Initialize & Setup elm_glview */
    {
        glview = elm_glview_add(win);
        elm_win_resize_object_add(win, glview);
        elm_glview_mode_set(glview, ELM_GLVIEW_ALPHA | ELM_GLVIEW_DEPTH | ELM_GLVIEW_DIRECT);
        ....
    }
    ....
    /* Set config of the surface for evas gl */
    ....
    ad->cfg = evas_gl_config_new();
    ad->cfg->options_bits = EVAS_GL_OPTIONS_DIRECT; // Configuration options (here, DIRECT mode on)
    ....
    /* Create a surface and context */
    ad->sfc = evas_gl_surface_create(ad->evasgl, ad->cfg, w, h);
    ad->ctx = evas_gl_context_create(ad->evasgl, NULL);
    ....
Performance Improvement (2): Pre-rotation feature

• Landscape typical way:
  – Use Intermediate off-screen Surface for Rotation

![Diagram showing the process of rendering and rotation with off-screen and on-screen surfaces]
Performance Improvement (2): Pre-rotation feature

- Landscape efficient way:
  - Pre-rotation which does not need the Intermediate Surface

Diagram:
- Hint for Rotation
- Rotate
- Render with GLES
- No Texture mapping
- on-screen surface
- Display
Pre-rotation in evasgl (1)

• How to use the feature?
  – Just turn-on DIRECT mode
  – Requirements
    • GPU Driver must supports pre-rotation feature
    • When GPU does not support, then the rendering mode fallbacks to INDIRECT mode

```c
/* Set config of the surface for evas gl */
......
ad->cfg = evas_gl_config_new();
ad->cfg->options_bits = EVAS_GL_OPTIONS_DIRECT; // DIRECT mode on
......

/* Get rotation angle for developers */
angle = evas_gl_rotation_get(ad->evas_gl); // angle is zero, and there is nothing for developers to do
  // when pre-rotation is not supported,
  // render mode fallbacks to INDIRECT mode for LANSCAPE state
......
```
Pre-rotation in evasgl (2)

• Workaround for devices not supporting pre-rotation?
  – Rotate the scene by application side
  – EVAS_GL_OPTIONS_CLIENT_SIDE_ROTATION
    • System is rotated (ex. touch), exception the on-screen surface

```c
/* Set config of the surface for evas gl */
......
ad->cfg = evas_gl_config_new();
ad->cfg->options_bits = EVAS_GL_OPTIONS_CLIENT_SIDE_ROTATION; // DIRECT mode on,
                                  // Rendering is always for Portrait
......

/* Get rotation angle for developers */
angle = evas_gl_rotation_get(ad->evas_gl); // angle shows the current device orientation
                                            // developers must rotate the rendered scene according to angle
......
```
In-App-Purchase (IAP) in Tizen (1)

• Tizen IAP
  – IAP feature based on AppControl mechanism
    • You can borrow the functionality of TizenStore Client
    • There is no prerequisite in your projects
  – Basic work flow
    • Register items to Tizen Store Seller Office (http://seller.tizenstore.com)
    • Make your applications to work with IAP
    • Test and upload your application
  – Just check the ‘IAP Programming Guide’ and do IAP right now
In-App-Purchase (IAP) in Tizen (2)

• Materials for IAP feature

http://developer.tizen.org/downloads/2.2.1-add-on-sdks

Add-on SDKs

The Tizen add-on SDKs provide specific features extending the Tizen SDK. To use the add-on SDKs, install the Tizen SDK, and then install the additional add-on SDKs using the Install Manager tool located in the tizen-sdk/install-manager folder. Please note the third party terms applicable to the selected add-on.

- Accelerated Computing for Tizen
  Improve application performance by exploring the parallelism of multi-core processors.

- bada Application Migration
  Convert bada applications to Tizen native applications using the bada Migration Tool.

- In-App Purchase
  Enable customers to buy digital content within your application.

Maps Powered by HERE

Develop location based applications that support maps powered by HERE.

Twitter

Integrate Twitter functions into your applications.

In-App Purchase

In-app purchase allows you to sell the digital content to customers inside your application such as in-game items, skins, in-app purchases. With in-app purchase service, you can offer your customers a variety of items directly within your paid or free application. Currently, you can use in-app purchase to sell standard in-app products (one-time purchase). You can implement in-app purchase in your application using Tizen AppCommerce framework, which provides the functionality to process payments for items offered in your application through the Tizen Store. In-app purchase uses the same payment service as is used for application purchases, so your customers experience a consistent and secure purchase flow. In-app purchases can be implemented in both Tizen native and Tizen web applications. To use in-app purchase, you can use your Tizen Store Seller Office account to register items to Tizen Store Seller Office. To get started, we recommend you to read thoroughly the in-app purchase programming guide.

CAUTION: If IAP Client’s “Operation ID” have been changed from IAP Client TPP version 1.0.7 or later. For more information, read the “SDKME.txt” in Attachment.

- IAP Programming Guide (1.35 MB)
- IAP Client TPP 1.3.0 (1.2 MB)
- Tizen Account TPP (1.46 MB)
- Native Sample Application Source Code (2.67 MB)
- Web Sample Application Source Code (1.62 MB)
- SDKME.txt (4.79 MB)
- Release Notes.txt (6.04 MB)
Tizen IAP with Unity

- Unity Plugin for Tizen IAP
  - Integrate C-based Tizen AppControl into .NET-based Unity scripts
Tizen IAP with cocos2d-x

- Brute force way to integrate Tizen IAP and cocos2d-x
  - Use Tizen AppControls in cocos2d-x app directly

- cocos2d-x plugin for Tizen IAP
  - cocos2d-x is open-source, and we are considering,
    - to integrate Tizen IAP to Plugin-x
    - to integrate Tizen IAP to SDKBOX
Summary

• Games for Tizen
  – New opportunity for business
  – Expandability, convergence and performance

• Porting to Tizen
  – Game engines help your joining to Tizen

• Basics and Tips for your development and optimization
  – evasgl and elm_glview
  – DIRECT mode and pre-rotation

• Monetization
  – Tizen IAP, and plugin supports
Q&A

and THANK YOU for your time.

Qingli Wang
qingli6.wang@samsung.com