

Breakthrough Games with Tizen

Sungyul Choe

TIZEN™
**DEVELOPER
SUMMIT**
2015 BENGALURU 
JULY 30-31, THE RITZ-CARLTON

Agenda

1. Introduction
2. Game Porting to Tizen
3. Tips for Development
4. Monetization
5. Demonstration – Games2Win
6. Summary

Introduction



Introduction

- **Market Status**

27 Games in TOP rank 50 (2015.06)

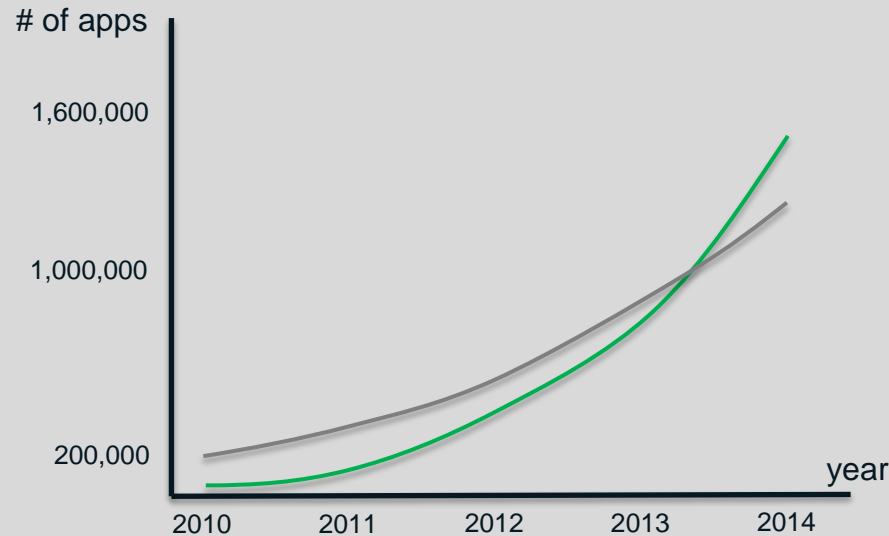
Why Tizen?

- **Expandability and Convergence**
 - Tizen Mobile, Tizen Wearable, Tizen TV, Home Appliances, IoT and more



Why Tizen?

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



* source: Business Insider

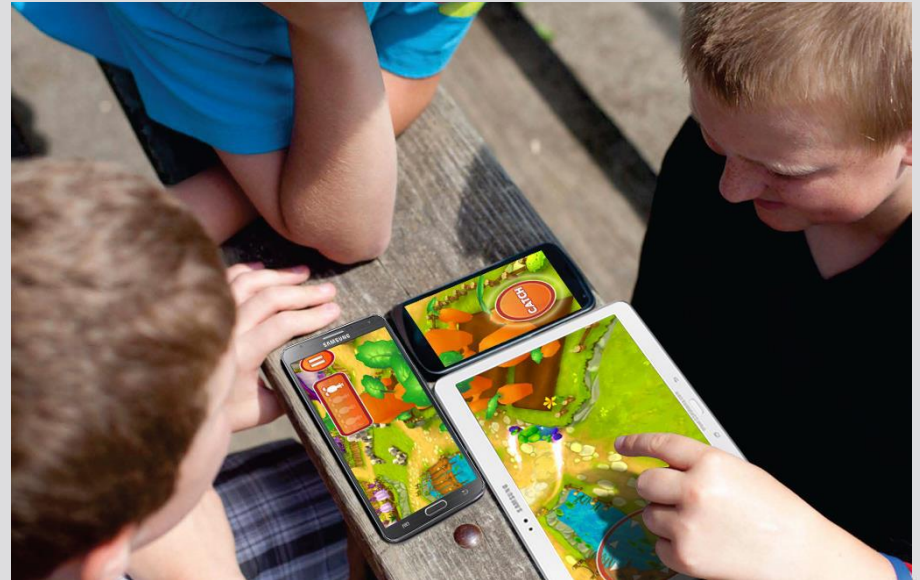
Why Tizen?

- **Unexplored and New Game Area**

Multi-user Gaming Experience



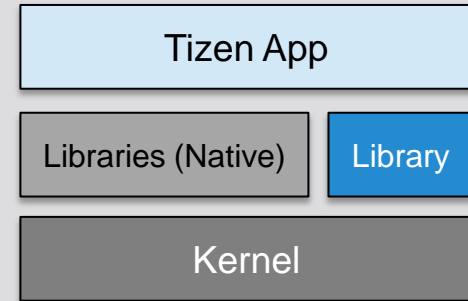
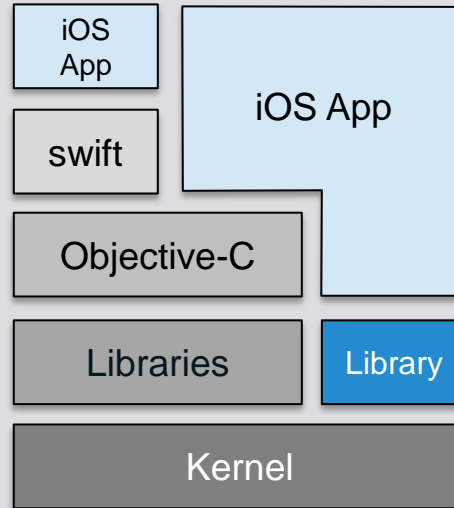
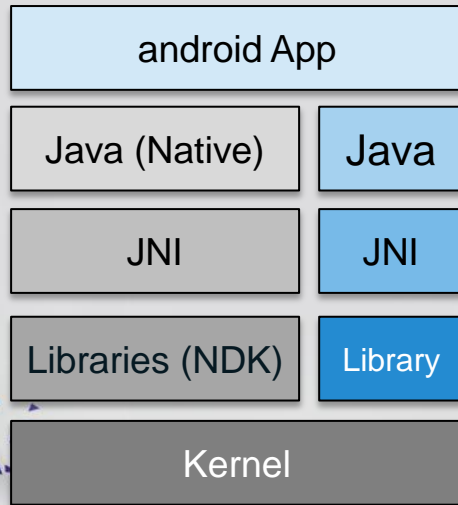
Multiple screens Game Experience



Why Tizen?

- **Efficiency**

- Development efficiency
 - C-based modules accelerate porting of open source modules



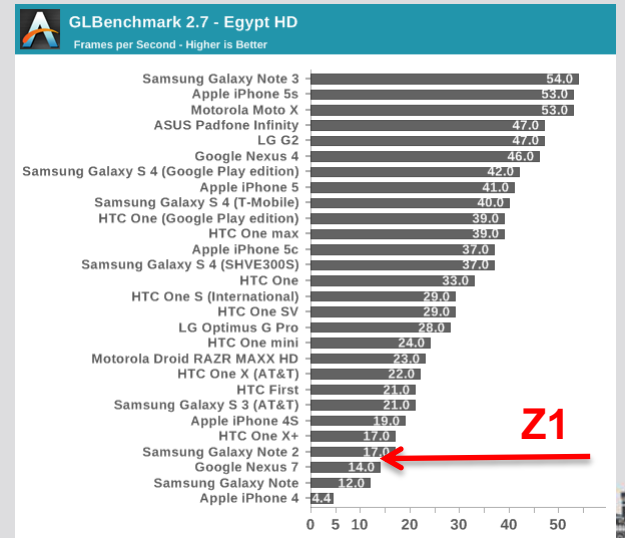
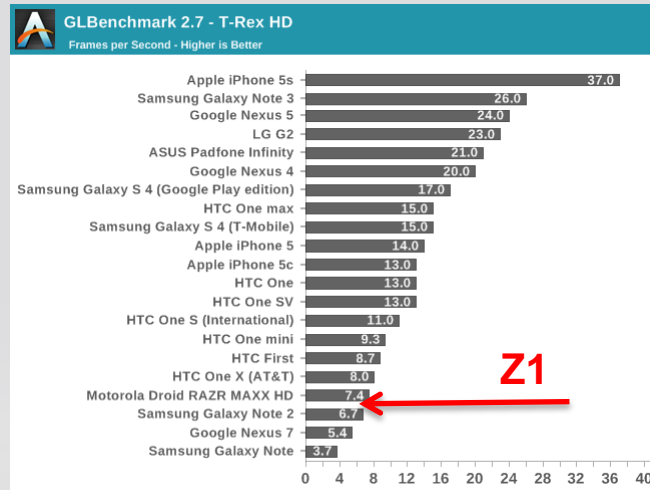
Why Tizen?

- **Efficiency**

- Device performance

- Graphics is highly optimized, which is deeply impressive for mass model with limited resources

Result	
Direct	
GLBenchmark 2.7 T-Rex HD C24Z16 Onscreen ETC1	385 frames 6.9 fps
GLBenchmark 2.7 T-Rex HD C24Z16 Offscreen ETC1	111 frames 2.0 fps
GLBenchmark 2.5 Egypt HD C24Z16 Onscreen ETC1	1739 frames 15 fps
GLBenchmark 2.5 Egypt HD C24Z16 Offscreen ETC1	960 frames 8.5 fps
None	
No results yet	-1



Why Tizen?

- Developer friendly Tizen Store Seller Promotion

100%
is YOURS!

ADD NEW APPLICATION >

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)

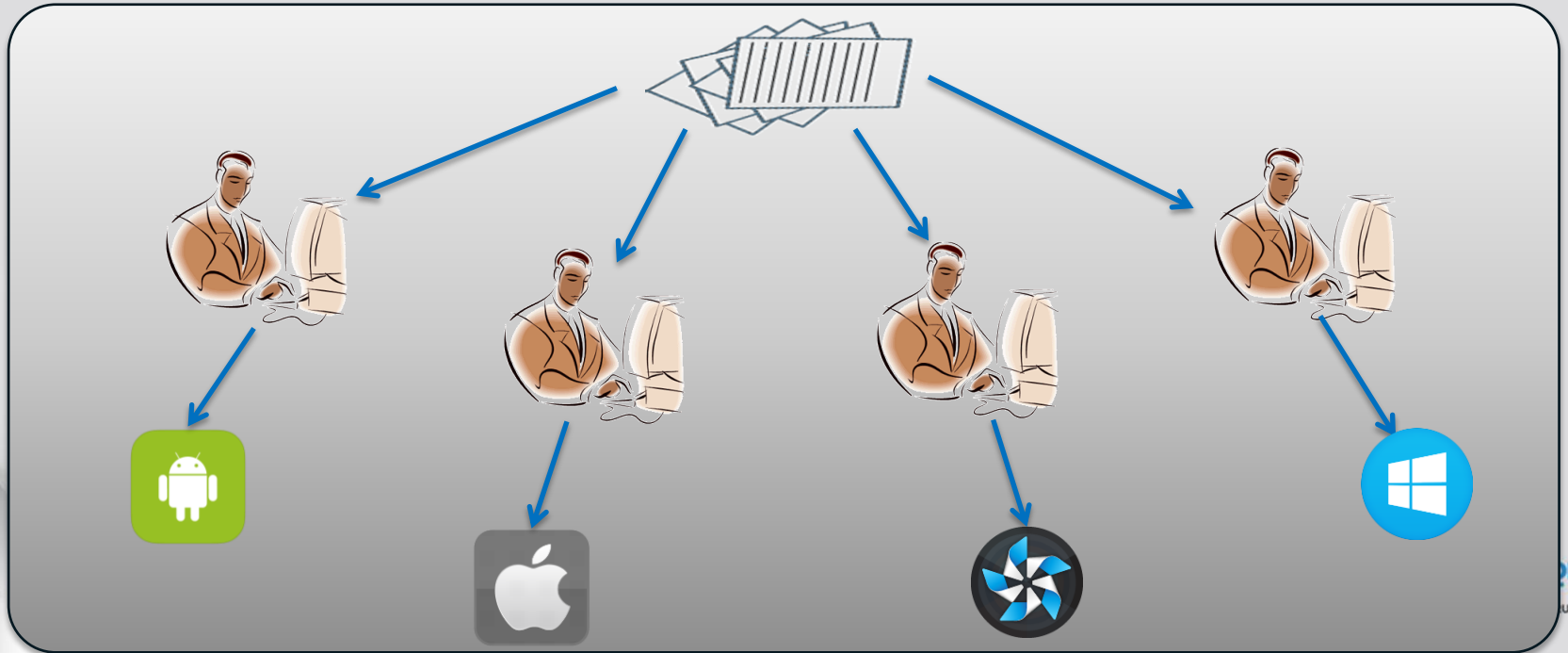
EVELOPER
JMMIT
5 BENGALURU

Game Porting to Tizen



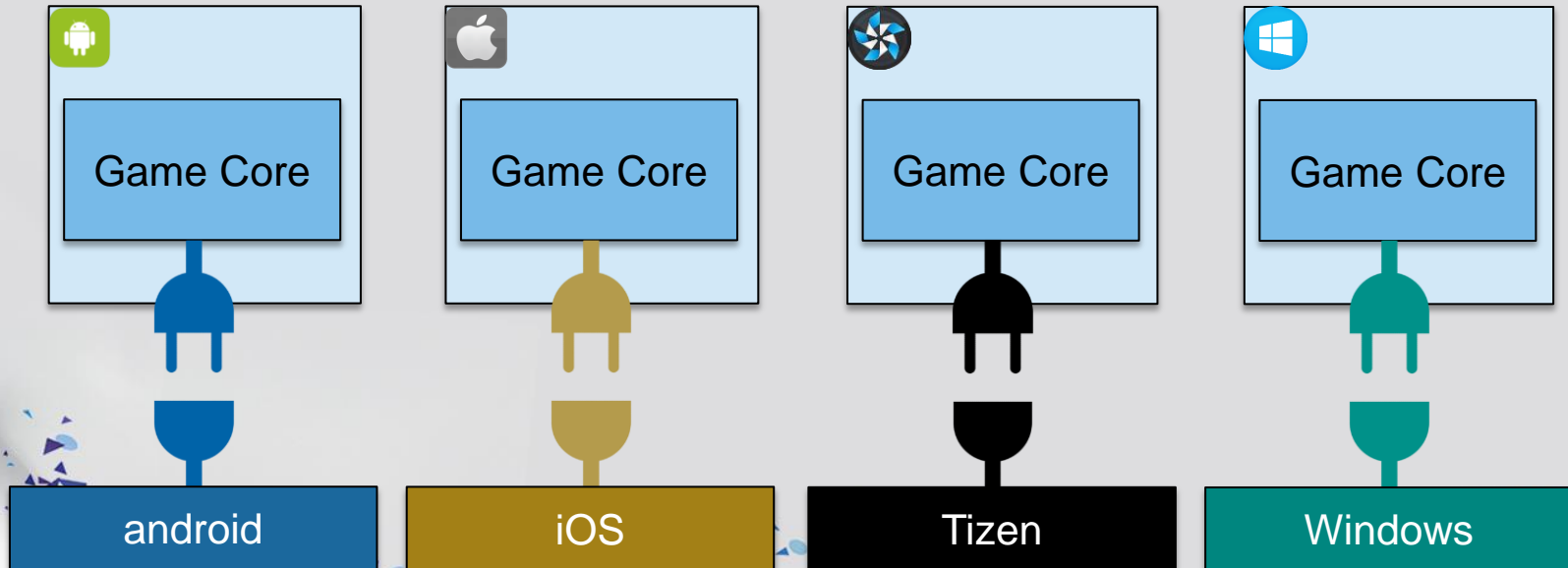
Game Porting to Tizen

- **Typical way**
 - Develop game for Tizen with the same scenario



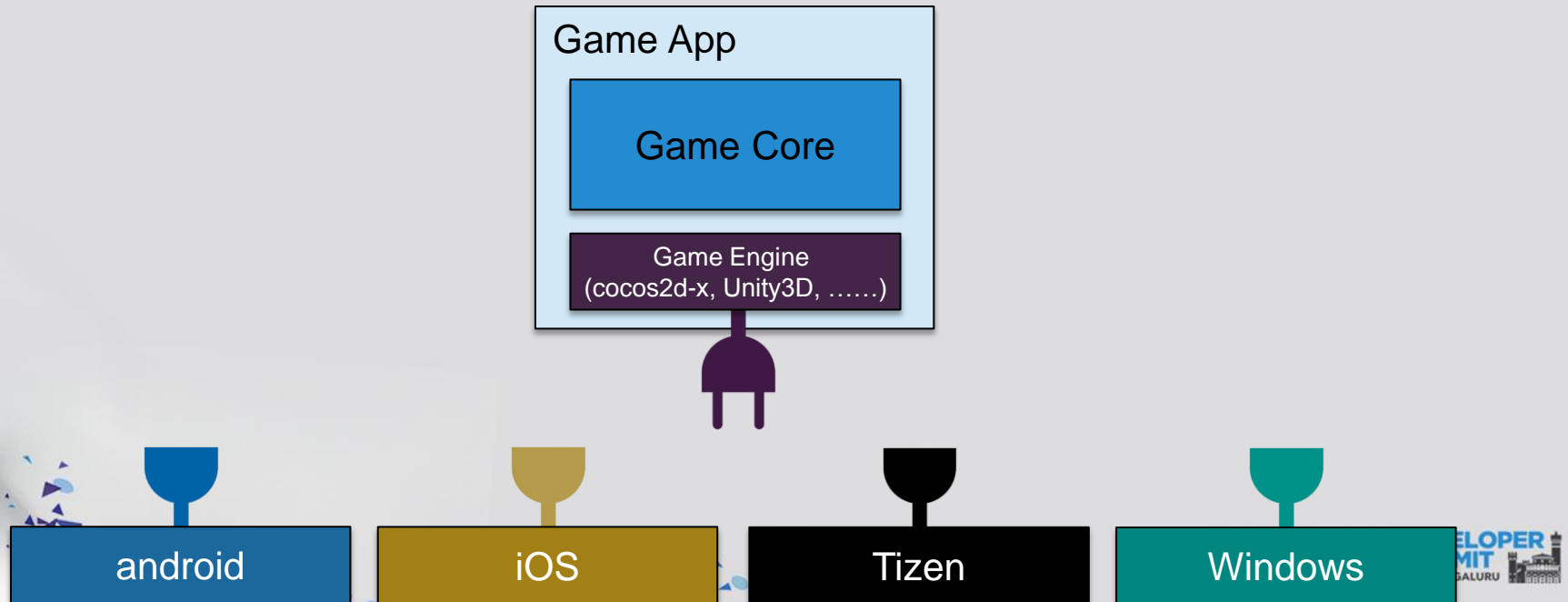
Game Porting to Tizen

- **Better way**
 - Divide porting layer from game core, and adapt only porting layer



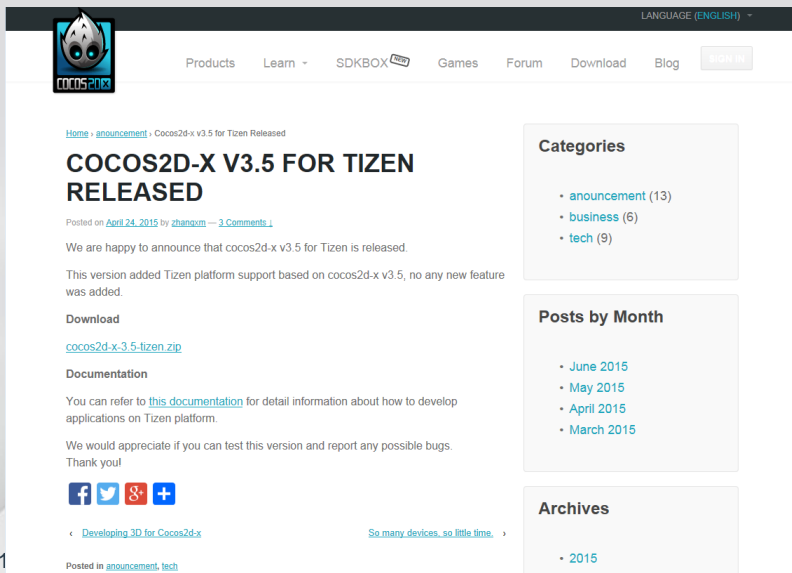
Game Porting to Tizen

- **Best way**
 - Adopt game engines, such as cocos2d-x & Unity3D

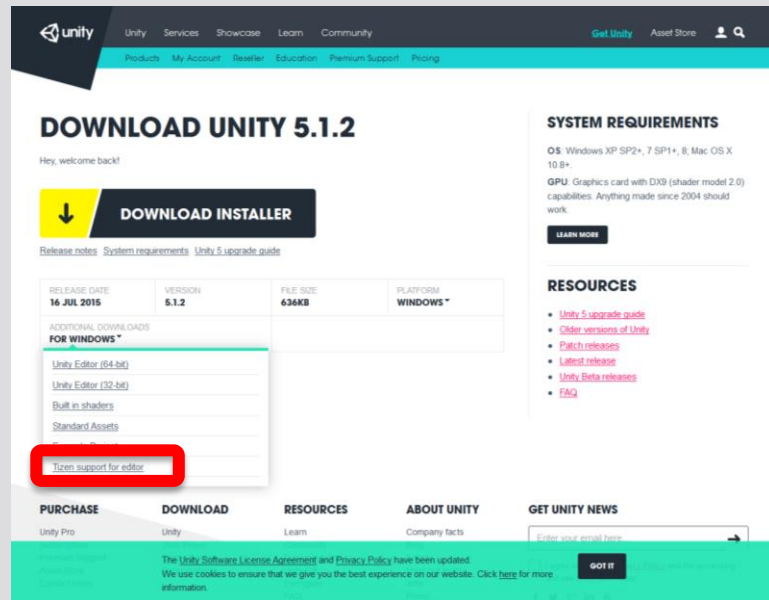


Game Porting to Tizen

- Famous Game Engines are ready for Tizen
 - Cocos2d-x (since ver.3.5.1)
 - Unity3D (since ver.5.1)



The screenshot shows the Cocos2d-x website with a navigation bar at the top containing 'Products', 'Learn', 'SDKBOX', 'Games', 'Forum', 'Download', and 'Blog'. The main content area features a large heading 'COCOS2D-X V3.5 FOR TIZEN RELEASED' with a sub-heading 'Cocos2d-x v3.5 for Tizen Released'. The text below the heading states: 'We are happy to announce that cocos2d-x v3.5 for Tizen is released. This version added Tizen platform support based on cocos2d-x v3.5, no any new feature was added.' There are sections for 'Download' (with a link to 'cocos2d-x-3.5-tizen.zip') and 'Documentation' (with a link to 'this documentation'). A 'Categories' sidebar on the right lists 'announcement (13)', 'business (6)', and 'tech (9)'. A 'Posts by Month' sidebar lists dates from June 2015 to March 2015. An 'Archives' sidebar shows '2015'. At the bottom, there are social media icons for Facebook, Twitter, Google+, and a plus sign for more. The footer includes 'Developing 3D for Cocos2d-x' and 'So many devices, so little time.'



The screenshot shows the Unity website's download page for Unity 5.1.2. The navigation bar at the top includes 'unity', 'Unity', 'Services', 'Showcase', 'Learn', 'Community', 'Get Unity', 'Asset Store', and a search icon. The main heading is 'DOWNLOAD UNITY 5.1.2'. Below the heading is a 'DOWNLOAD INSTALLER' button with a download icon. A table below the button shows release information:

RELEASE DATE	VERSION	FILE SIZE	PLATFORM
16 JUL 2015	5.1.2	636KB	WINDOWS*

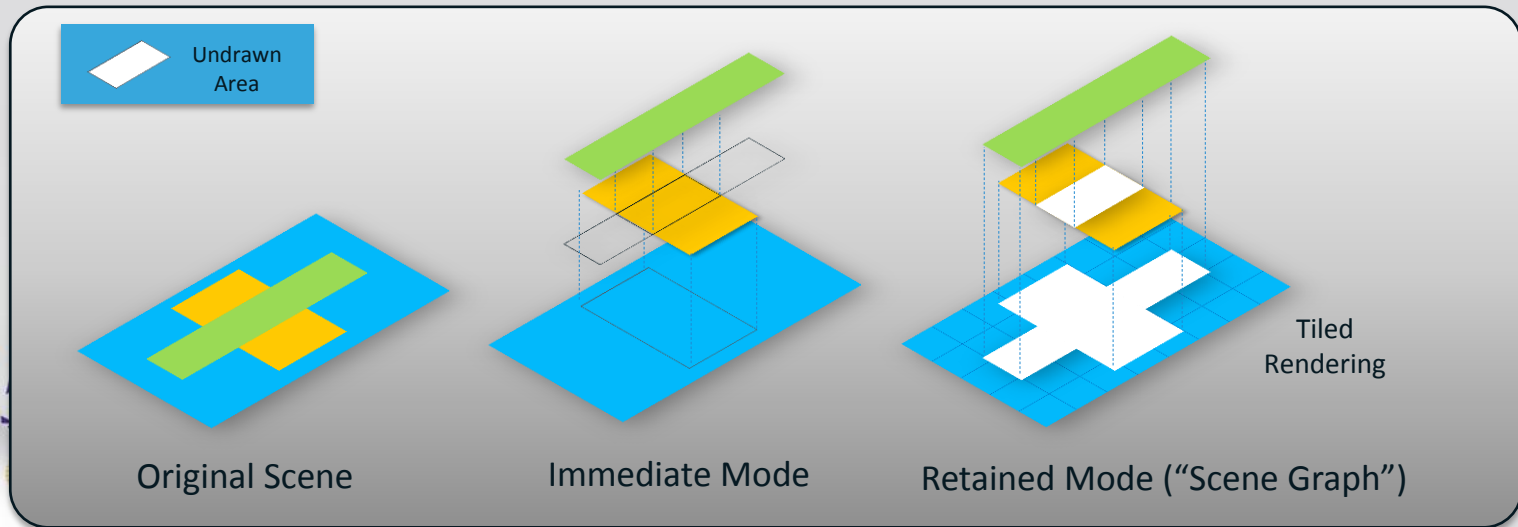
Below the table, there is a section for 'ADDITIONAL DOWNLOADS FOR WINDOWS*' with a list of items: 'Unity Editor (64-bit)', 'Unity Editor (32-bit)', 'Built-in shaders', and 'Standard Assets'. A red box highlights the 'Tizen support for editor' link. To the right, there are sections for 'SYSTEM REQUIREMENTS' (listing OS and GPU requirements) and 'RESOURCES' (with links to 'Unity 5 upgrade guide', 'Older versions of Unity', 'Patch releases', 'Latest release', 'Unity Beta releases', and 'FAQ'). At the bottom, there are sections for 'PURCHASE', 'DOWNLOAD', 'RESOURCES', 'ABOUT UNITY', and 'GET UNITY NEWS'. The footer contains the Unity Software License Agreement and Privacy Policy information.

Tips for Development - evasgl Basics



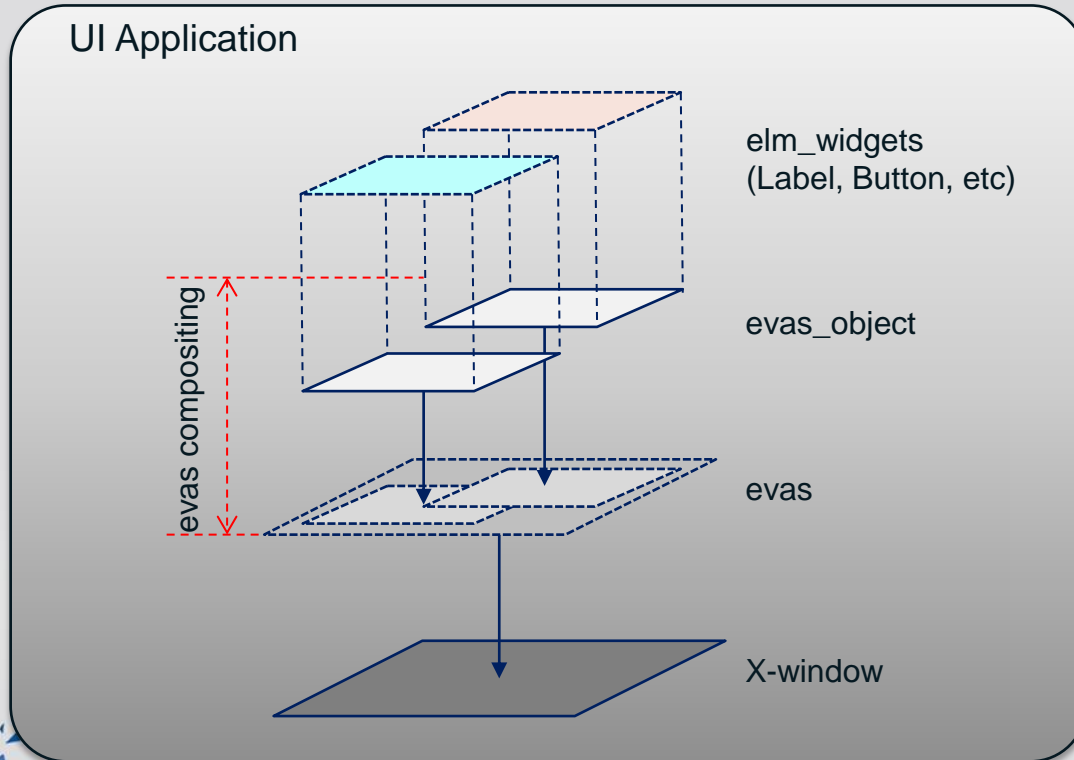
evasgl basics (1)

- **EFL (Enlightenment Foundation Libraries)**
 - Collection of open source libraries from Enlightenment
- **evas (Efl + canVAS)**
 - evas is Scene Graph composed of 'evas objects'



evasgl basics (2)

- EFL View Hierarchy

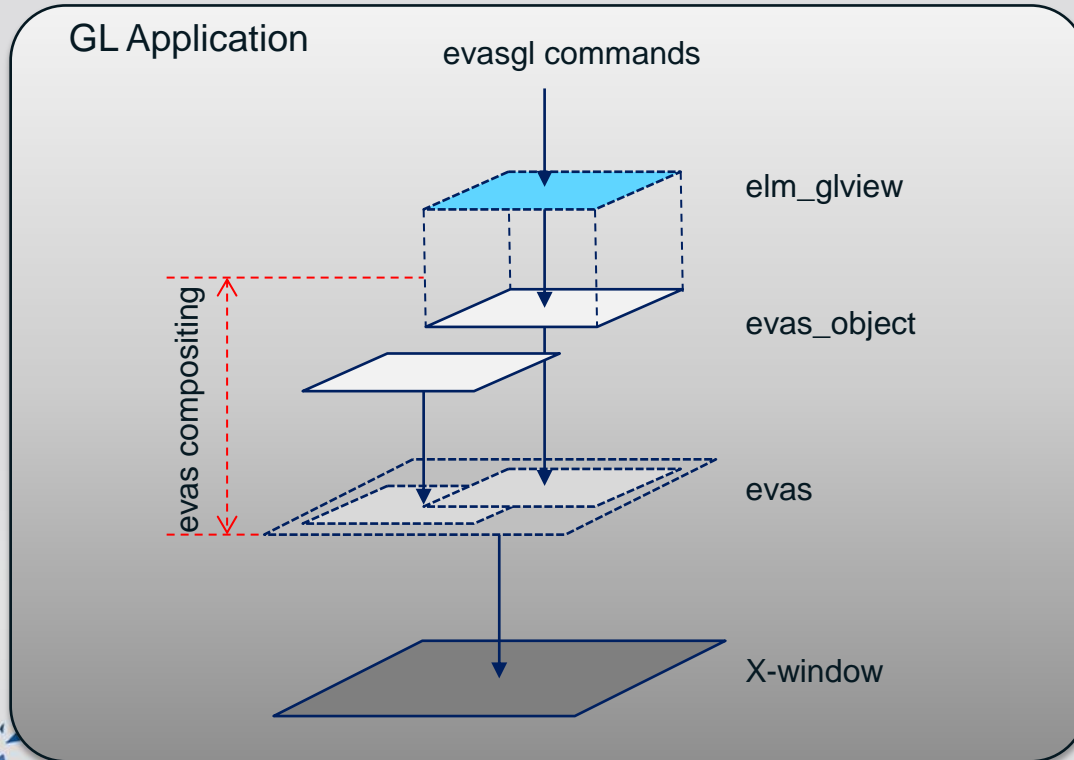


evasgl basics (3)

- **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 codes**
 - 1 Application initialization
 - 2 evasgl initialization
 - 3 Animation and rendering settings
 - Add animator and renderer to ecore main loop
 - 4 Define rendering with evasgl functions



1. Application Initialization

app_main part

```
#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;
```

```
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;
}
```

1. Application Initialization

app_main part

```
#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;
```

```
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;
}
```

1. Application Initialization

app_main part

```
#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;
```

```
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

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 gneration and other stuffs */
.....
```

2. evasgl Initialization

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)
```

```
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 */
.....
```

2. evasgl Initialization

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 */
.....
```

2. evasgl Initialization

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 gneration and other stuffs */
.....
```

2. evasgl Initialization

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 */
.....
```

2. evasgl Initialization

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));
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
    .....
}
```

3. Animation and Rendering setting

Animation and Rendering

```
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;
```

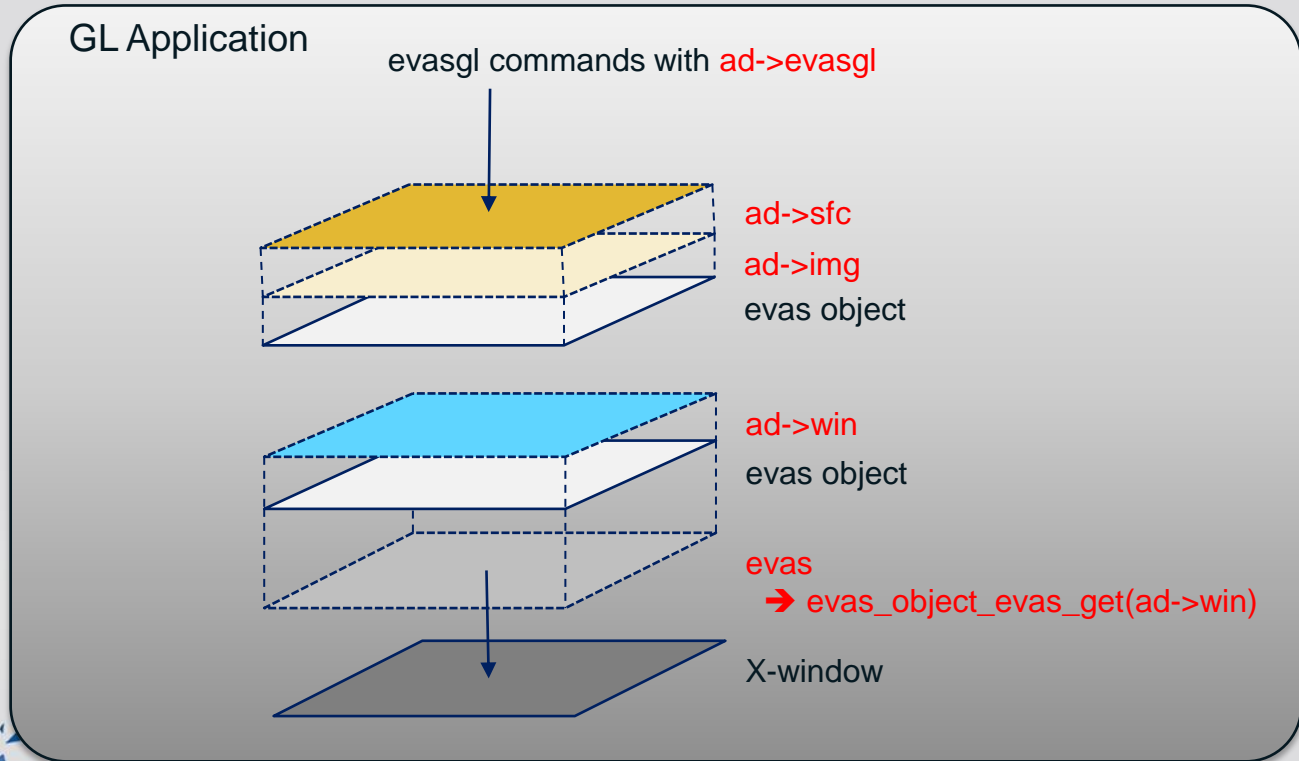
```
    // Rendering process here
```

```
    .....
```

```
}
```


Sameple code – Draw one cube

- EFL View Hierarchy for evasgl initialization



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_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  
    .....  
}
```

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));
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);
```

```
/* 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
    .....
}
```

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));  
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  
    .....  
}
```

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));
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);
```

```
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

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);
    .....
}
```

4. Rendering with evasgl

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);

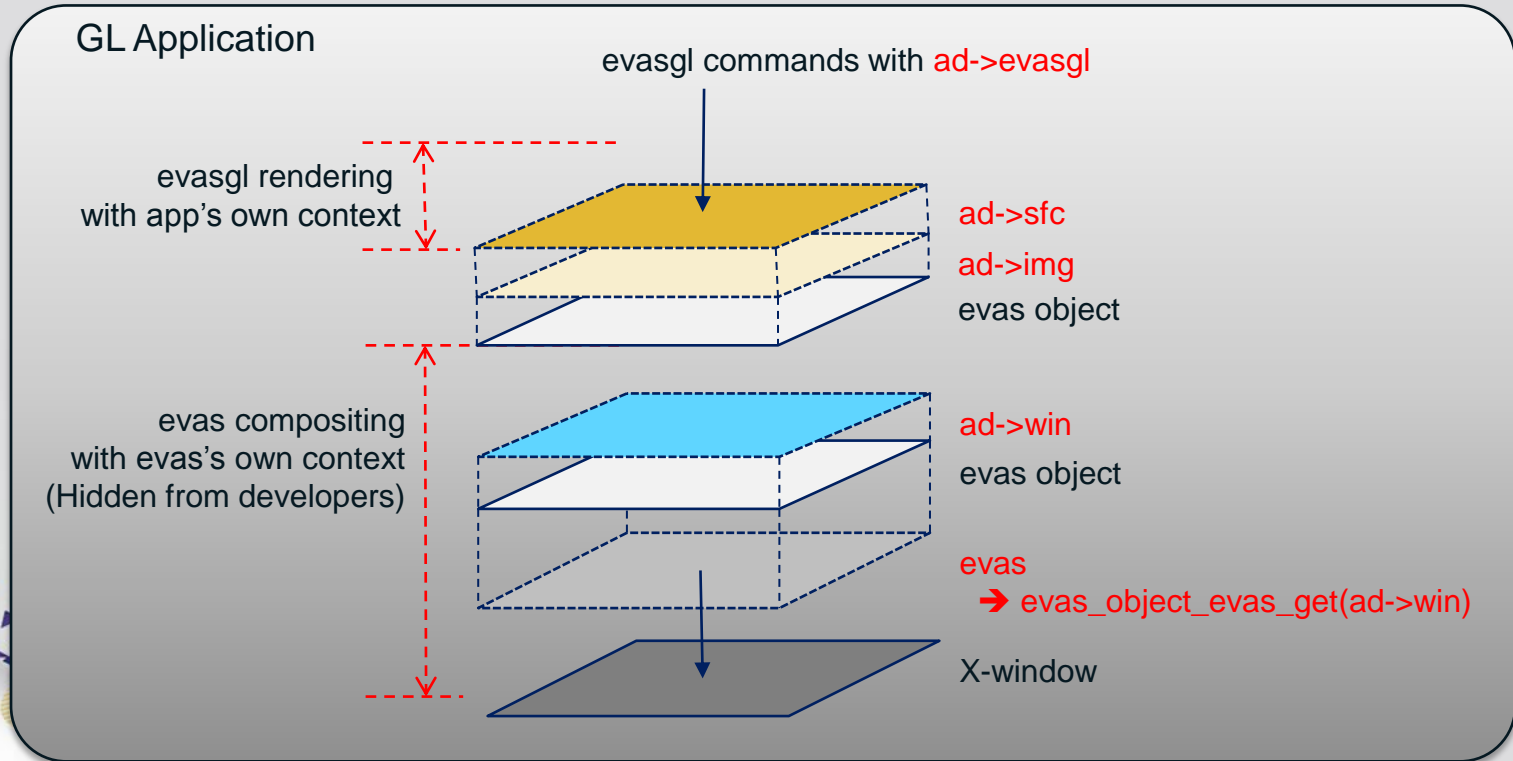
    gl->glUseProgram(ad->program);

    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`



Tips for Development - elm_glview



elm_glview

- **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

```
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);
    }
    .....
}
```



Sample Code – Change Initialization

```
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);
}
.....
}
```



Sample Code – Change Initialization

```
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);
}
.....
}
```



Sample Code – Change Initialization

```
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);
```

```
    }
```

```
    .....
```

```
}
```



Sample Code – Change Initialization



```
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);
    }
    .....
}
```

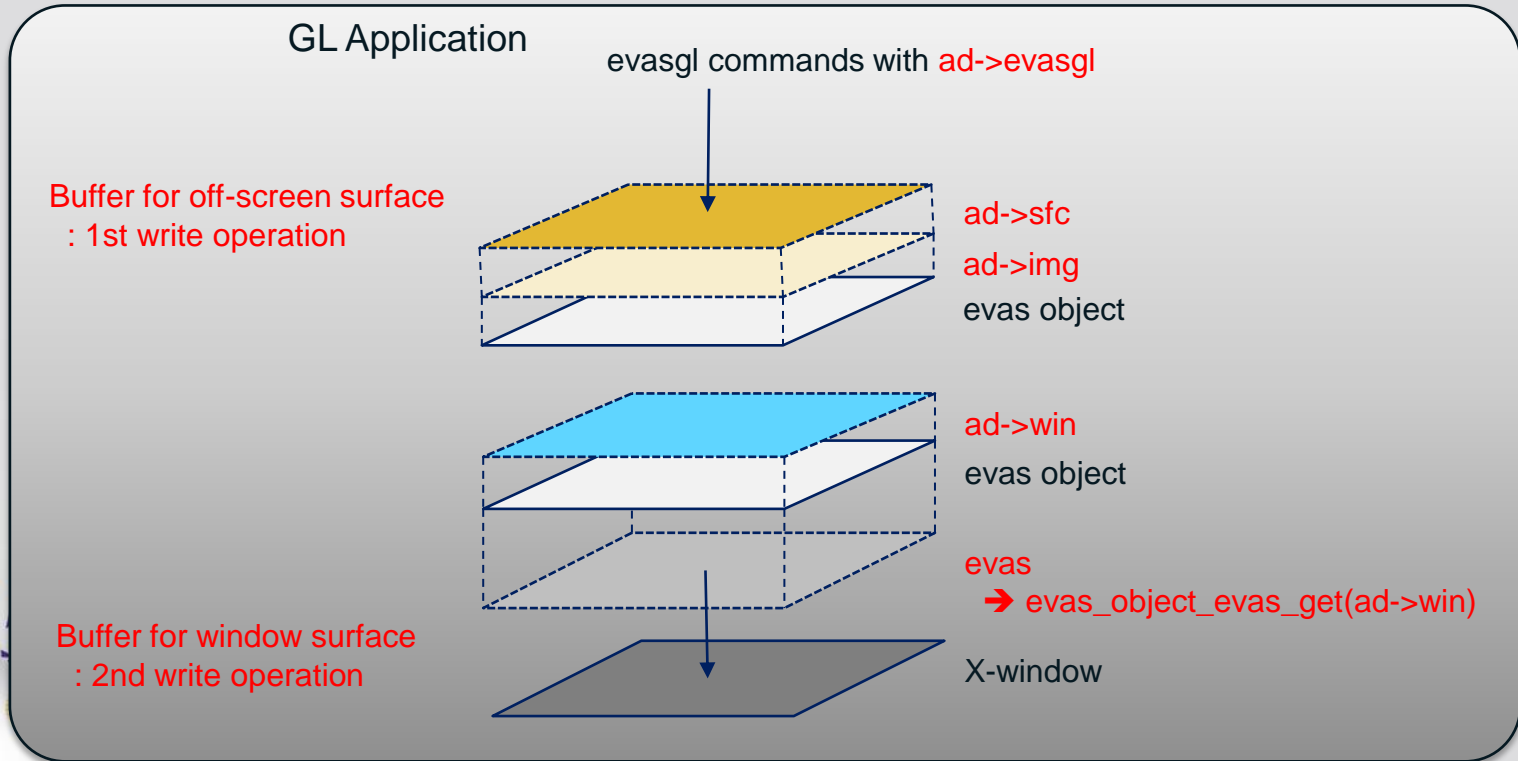
The background features a complex geometric design. It includes several white, 3D-style folded paper shapes that create a sense of depth. These shapes are decorated with various patterns: some have solid blue circles, others have white circles, and some have blue and white stripes. Scattered throughout the scene are numerous small, dark blue triangles and circles. In the upper right quadrant, there are black silhouettes of architectural structures, including a building with a dome and a tower with a crown-like top. The overall color palette is primarily white, blue, and black, with a touch of yellow in one of the smaller circles.

Tips for Development

- Performance Improvement (1)
- : DIRECT mode

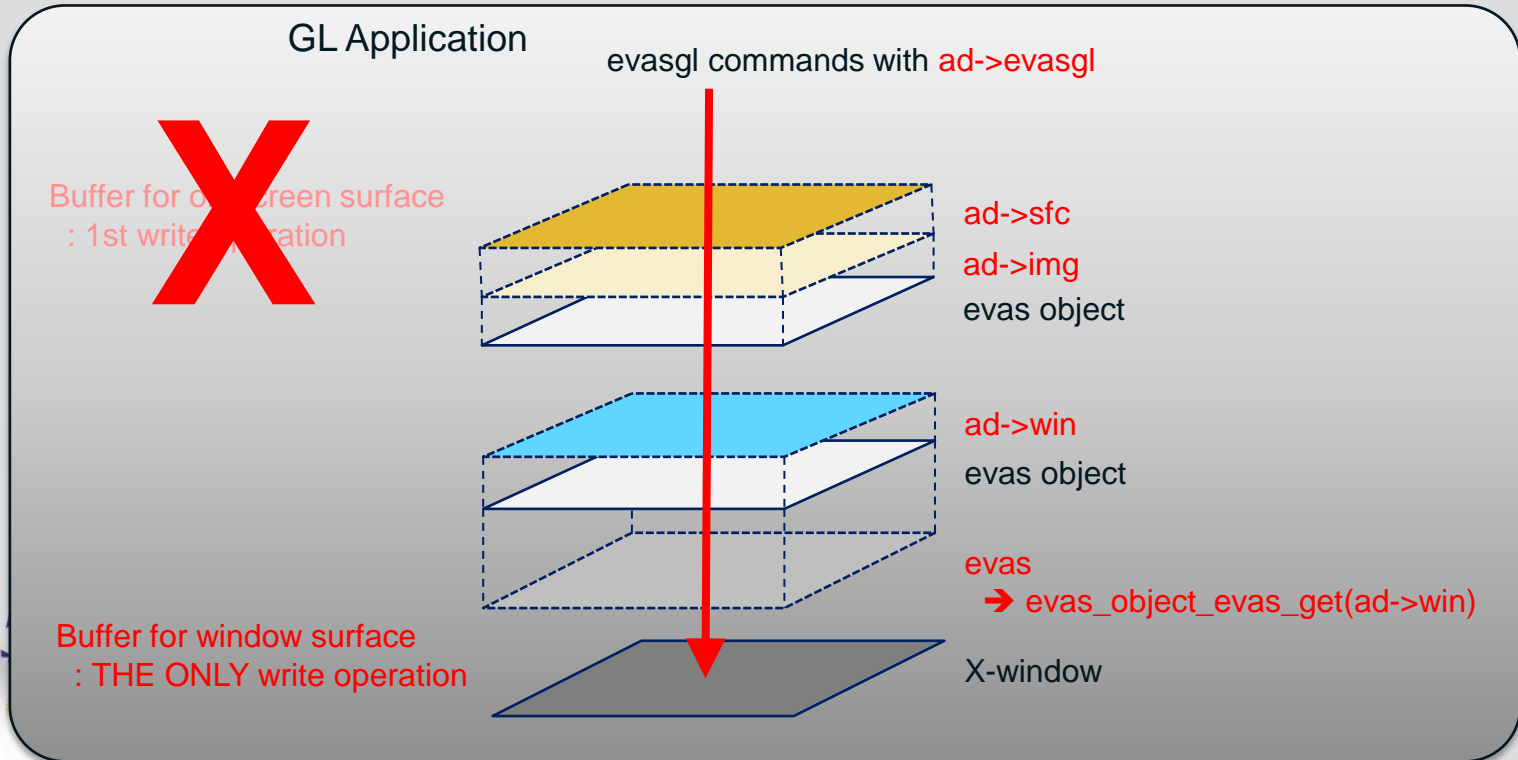
evasgl INDIRECT mode

- EFL View Hierarchy of full-screen GLES application



evasgl DIRECT mode (1)

- EFL View Hierarchy of full-screen GLES application



Sample Code – Change Initialization



elm_glvview case

```
EAPI int elm_main(int argc EINA_UNUSED, char **argv EINA_UNUSED)
{
    .....
    /* Initialize & Setup elm_glvview */
    {
        glview = elm_glvview_add(win);
        elm_win_resize_object_add(win, glview);
        elm_glvview_mode_set(glview, ELM_GLVVIEW_ALPHA | ELM_GLVVIEW_DEPTH | ELM_GLVVIEW_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);
.....
```

Sample Code – Change Initialization



elm_glvview case

```
EAPI int elm_main(int argc EINA_UNUSED, char **argv EINA_UNUSED)
{
    .....
    /* Initialize & Setup elm_glvview */
    {
        glview = elm_glvview_add(win);
        elm_win_resize_object_add(win, glview);
        elm_glvview_mode_set(glview, ELM_GLVVIEW_ALPHA | ELM_GLVVIEW_DEPTH ELM_GLVVIEW_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);
.....
```

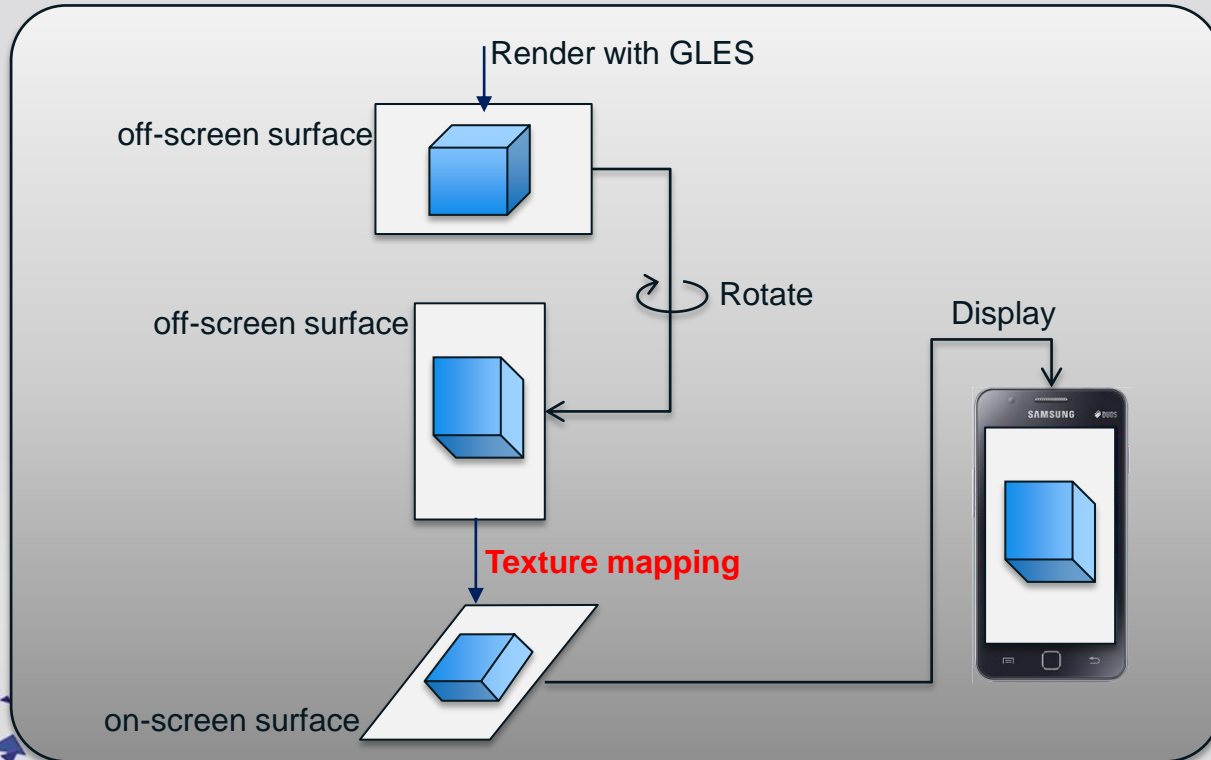
Tips for Development

- Performance Improvement (2)
- : Pre-rotation feature



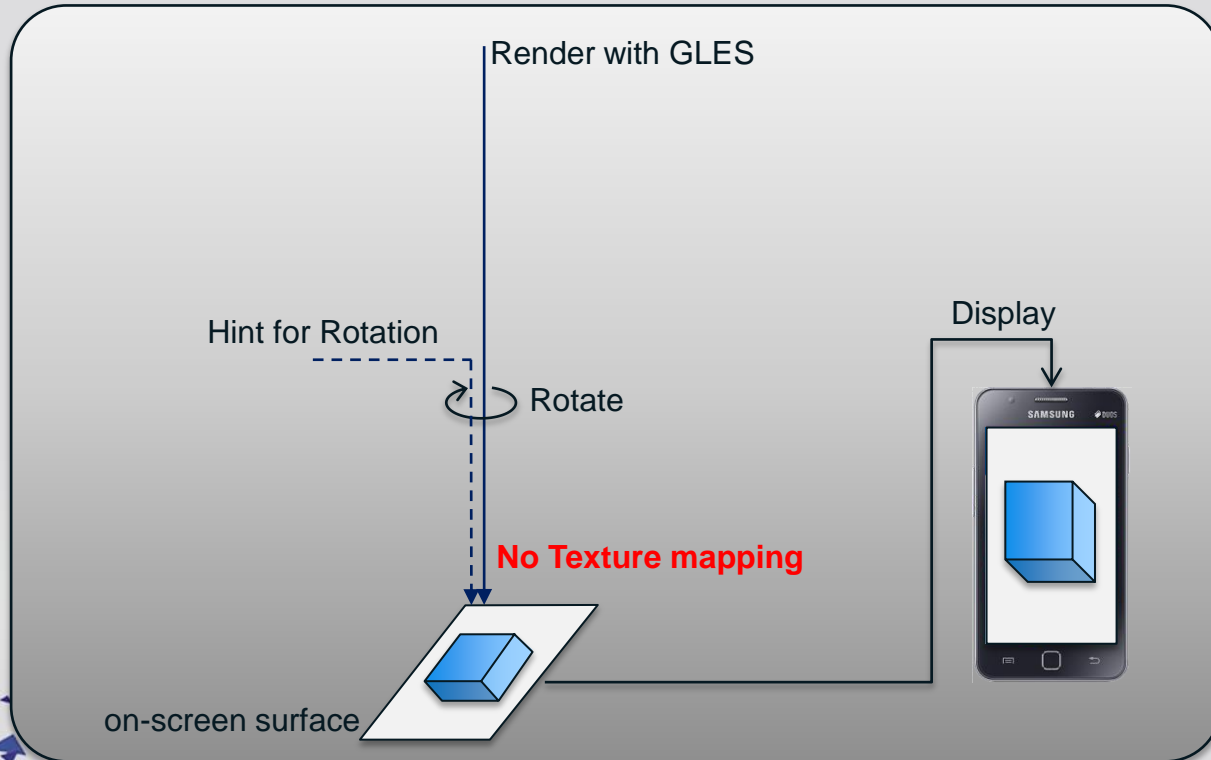
Typical Way for Landscape

- Use Intermediate off-screen Surface for Rotation



Efficient Way for Landscape

- Pre-rotation which does not need the Intermediate Surface



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

Case 1: EVAS_GL_OPTIONS_DIRECT mode

```
/* 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 LANDSCAPE 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

Case 2: EVAS_GL_OPTIONS_CLIENT_SIDE_ROTATION

```
/* 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 shows the current device orientation
                                           // developers must rotate the rendered scene according to angle
.....
```

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

Case 2: EVAS_GL_OPTIONS_CLIENT_SIDE_ROTATION

```
/* 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
.....
```

Monetization



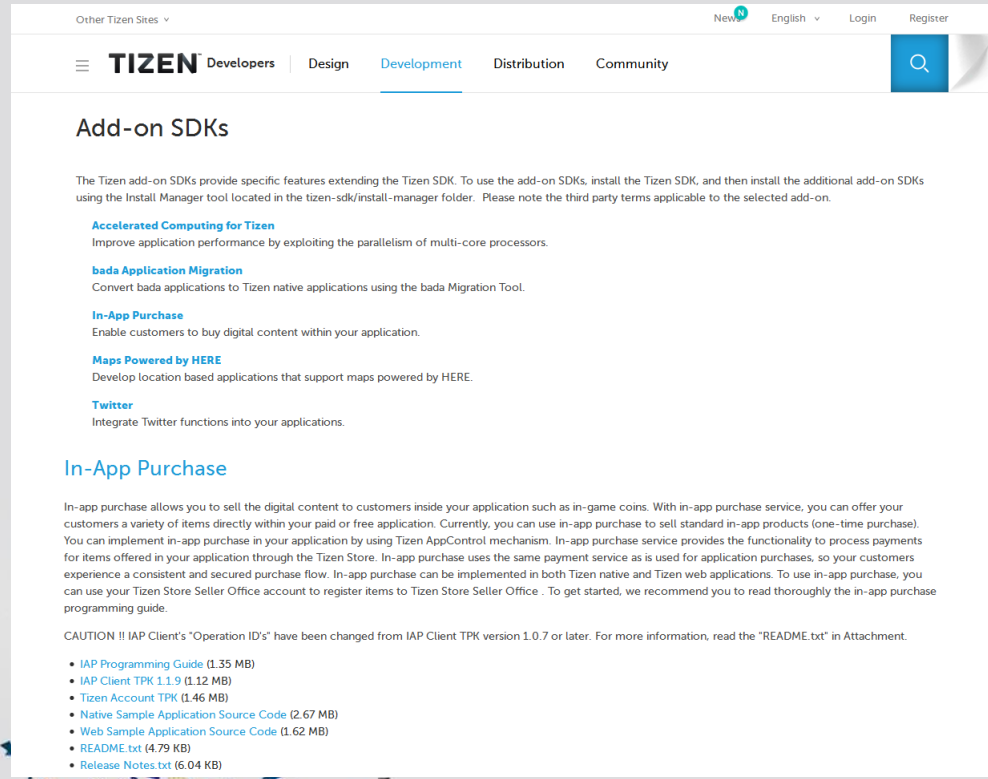
In-App-Purchase (IAP) in Tizen (1)

- **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>



Other Tizen Sites ▾ New ^N English ▾ Login Register

≡ **TIZEN** Developers | Design Development | Distribution | Community 🔍

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 exploiting 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

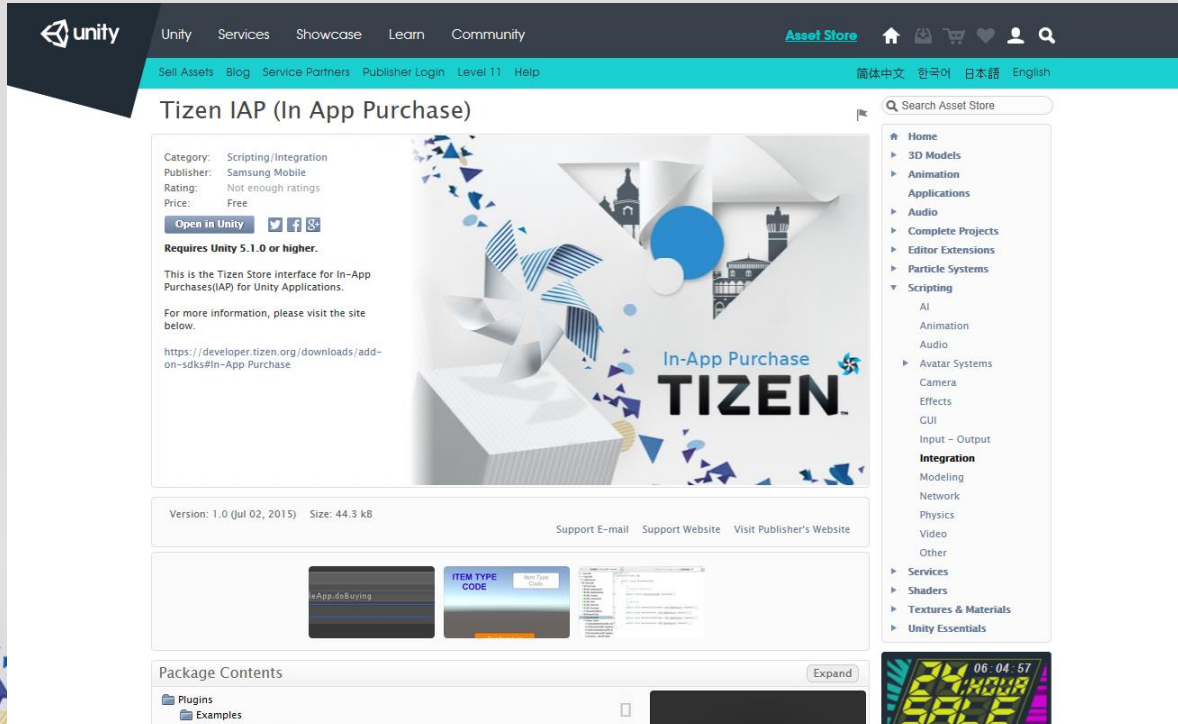
In-app purchase allows you to sell the digital content to customers inside your application such as in-game coins. With in-app purchase service, you can offer your customers a variety of items directly within your paid or free application. Currently, you can use in-app purchase to sell standard in-app products (one-time purchase). You can implement in-app purchase in your application by using Tizen AppControl mechanism. In-app purchase service 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 secured purchase flow. In-app purchase 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 !! IAP Client's "Operation ID's" have been changed from IAP Client TPK version 1.0.7 or later. For more information, read the "README.txt" in Attachment.

- [IAP Programming Guide \(1.35 MB\)](#)
- [IAP Client TPK 1.1.9 \(1.12 MB\)](#)
- [Tizen Account TPK \(1.46 MB\)](#)
- [Native Sample Application Source Code \(2.67 MB\)](#)
- [Web Sample Application Source Code \(1.62 MB\)](#)
- [README.txt \(4.79 KB\)](#)
- [Release Notes.txt \(6.04 KB\)](#)

Tizen IAP with Unity

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



The screenshot shows the Unity Asset Store page for the 'Tizen IAP (In App Purchase)' plugin. The page header includes the Unity logo and navigation links for Services, Showcase, Learn, and Community. The 'Asset Store' tab is active, and the page is in English. The main content area features a large 3D graphic with the text 'In-App Purchase TIZEN'. To the left of the graphic, the following information is displayed:

- Category: Scripting/Integration
- Publisher: Samsung Mobile
- Rating: Not enough ratings
- Price: Free
- Buttons: Open in Unity, and social media icons for Twitter, Facebook, and LinkedIn.
- Requires Unity 5.1.0 or higher.
- Description: This is the Tizen Store interface for In-App Purchases(IAP) for Unity Applications. For more information, please visit the site below. <https://developer.tizen.org/downloads/add-on-sdks#In-App Purchase>
- Version: 1.0 (Jul 02, 2015) Size: 44.3 kB
- Support E-mail, Support Website, Visit Publisher's Website

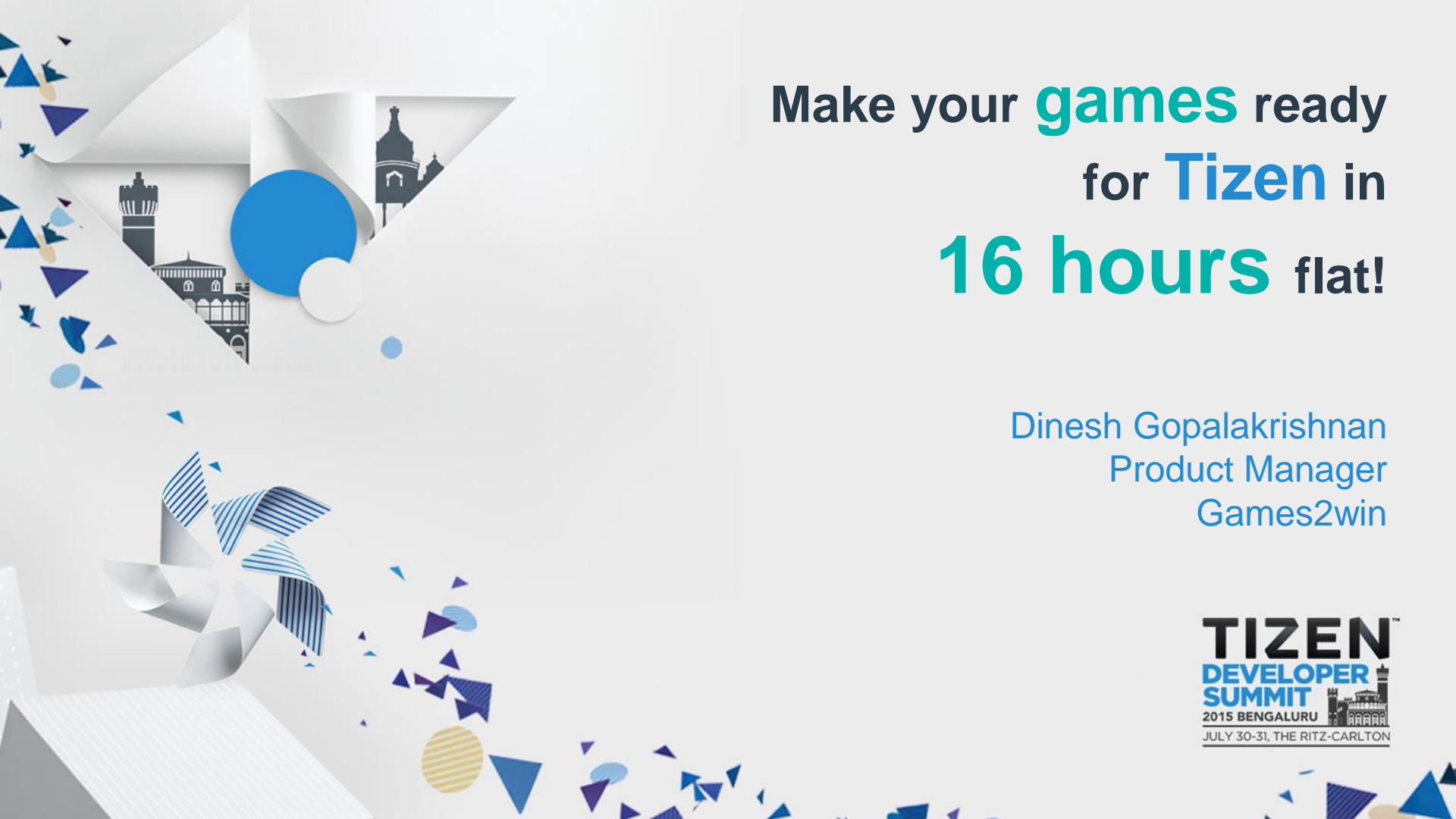
Below the main content, there are three preview images: a screenshot of the plugin's interface, a code snippet showing 'ITEM TYPE CODE', and a screenshot of a Tizen application interface. The 'Package Contents' section is partially visible at the bottom, showing 'Plugins' and 'Examples'.

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

Demonstration : Games2Win





Make your **games** ready
for **Tizen** in
16 hours flat!

Dinesh Gopalakrishnan
Product Manager
Games2win

TIZEN™
DEVELOPER
SUMMIT
2015 BENGALURU 
JULY 30-31, THE RITZ-CARLTON

Who we are...



Games2win is India's #1 Games Company!



- With **59+ million** mobile games downloads (**All Organic & Viral**)
- **2.5 million** mobile monthly actives
- **50+** unique mobile titles
- Including the #1 Top Ranking game "Parking Frenzy"

But, just 16 hours to
port a game for Tizen?



Yes, Here's how you do it....

- Use Unity version 5.1.1 – P3 version with the latest Tizen Player and Tizen SDK.
- Follow step by step instructions given on the Tizen developer site (developer.tizen.org) to port Unity games for the Tizen Appstore.
- Install the latest version of Tizen OS (TIZEN 2.3.0.1) on the Samsung Tizen phone and test it. It Works!

That's It!

Experience with
porting to Tizen!



Painless! Yes, No major problems at all

Porting was quite simple, apart from 3 basic errors:

- **Crash on the IAP interface:**
Solution: Update to the latest version of Tizen OS on the phone.
- **Game file size increased by 10 times:**
Solution: Use the latest Tizen SDK
- **Quality of in-game sounds deteriorated:**
Solution: Use the latest Tizen player

The Tizen Dev Support team needs a special mention here for their prompt support to us!

Any doubts?
Please contact to me on:
dinesh@games2win.com



Summary



Wrap-up

- **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_glvview`
 - DIRECT mode and pre-rotation
- **Monetization**
 - Tizen IAP, and plugin supports
- **Demonstration**
 - Games2Win shares their porting experiences

Do Not Miss.....

DAY2

10:00 - 13:00

■ **Tizen TV - App Development Workshop**

Track: Workshop

Location: Ballroom 2

■ **Tizen Game Development Workshop**

Track: Workshop

Location: Ballroom 2

Q&A

and **THANK YOU** for your time.

Sungyul Choe
sungyul.choe@samsung.com