bada 개발 환경에서
기기간 호환성을 보장하는 컨텐츠 개발

Presentation

Ph.D. 최성열
2011.11.09

Introduction

Mobile Platform에서 프로그램을 개발하는 이유는?

재미

운명의 데스티니

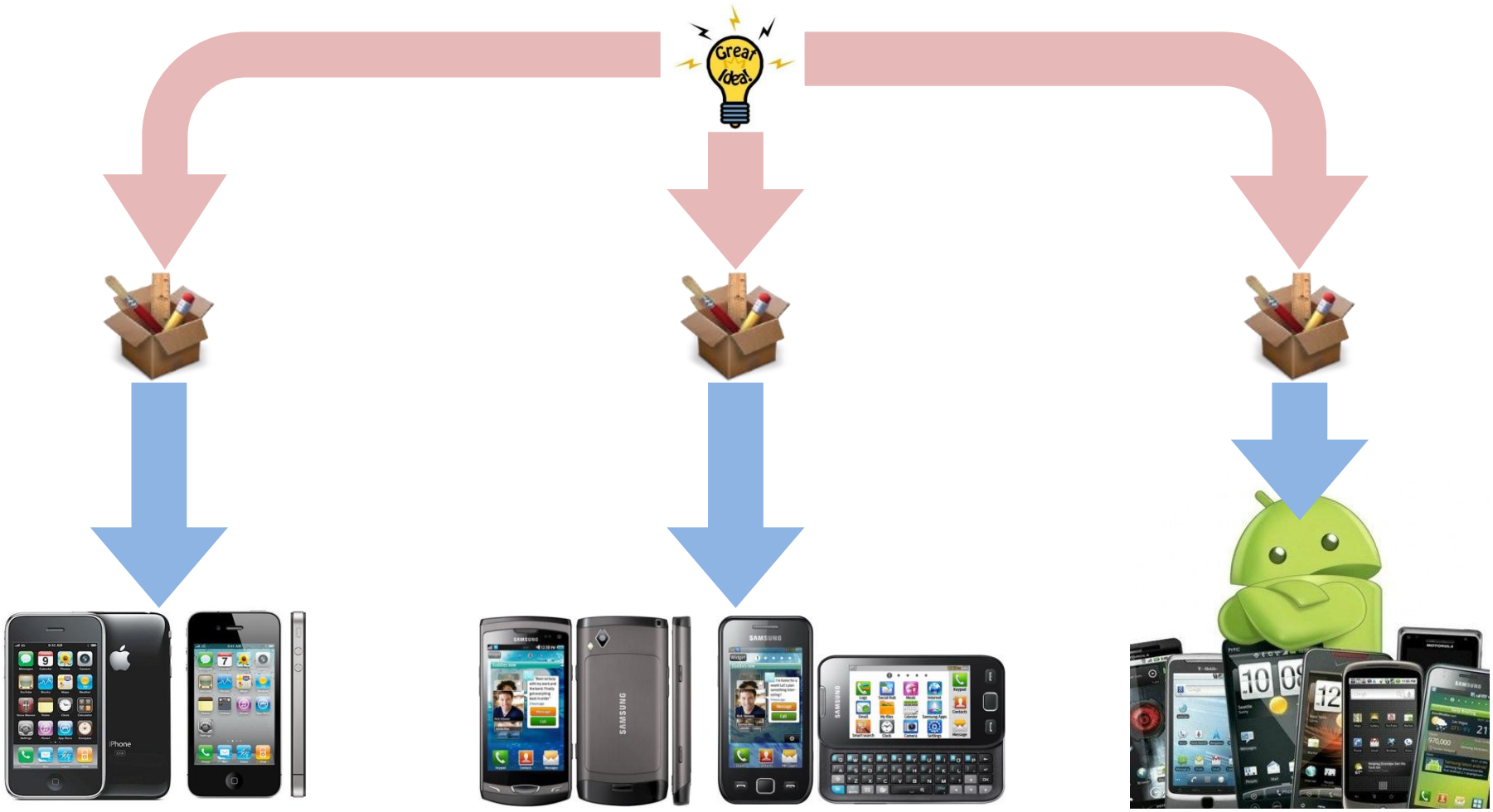
자기만족

돈

Introduction

목적 달성을 위해서 해야 하는 일?

되도록 많은 사람들이 내가 개발한 application을 쓰도록 하자! ➔ 가능한 많은 platform으로 개발하자!



Problem

수많은 Platform

Problem

다양한 version들

- android
  - Cupcake (ver. 1.5)
  - Donut (ver. 1.6)
  - Éclair (ver. 2.0/2.1)
  - Froyo (ver. 2.2)
  - GingerBread (ver. 2.3.x)
  - HoneyComb (ver. 3.0)
  - IcecreamSandwitch (ver. 4.0.x)

Problem

OpenGL-ES

높은 이식성을 바탕으로 최소의 노력으로 복수의 platform으로 application을 출시할 수 있도록 도와준다.

Problem

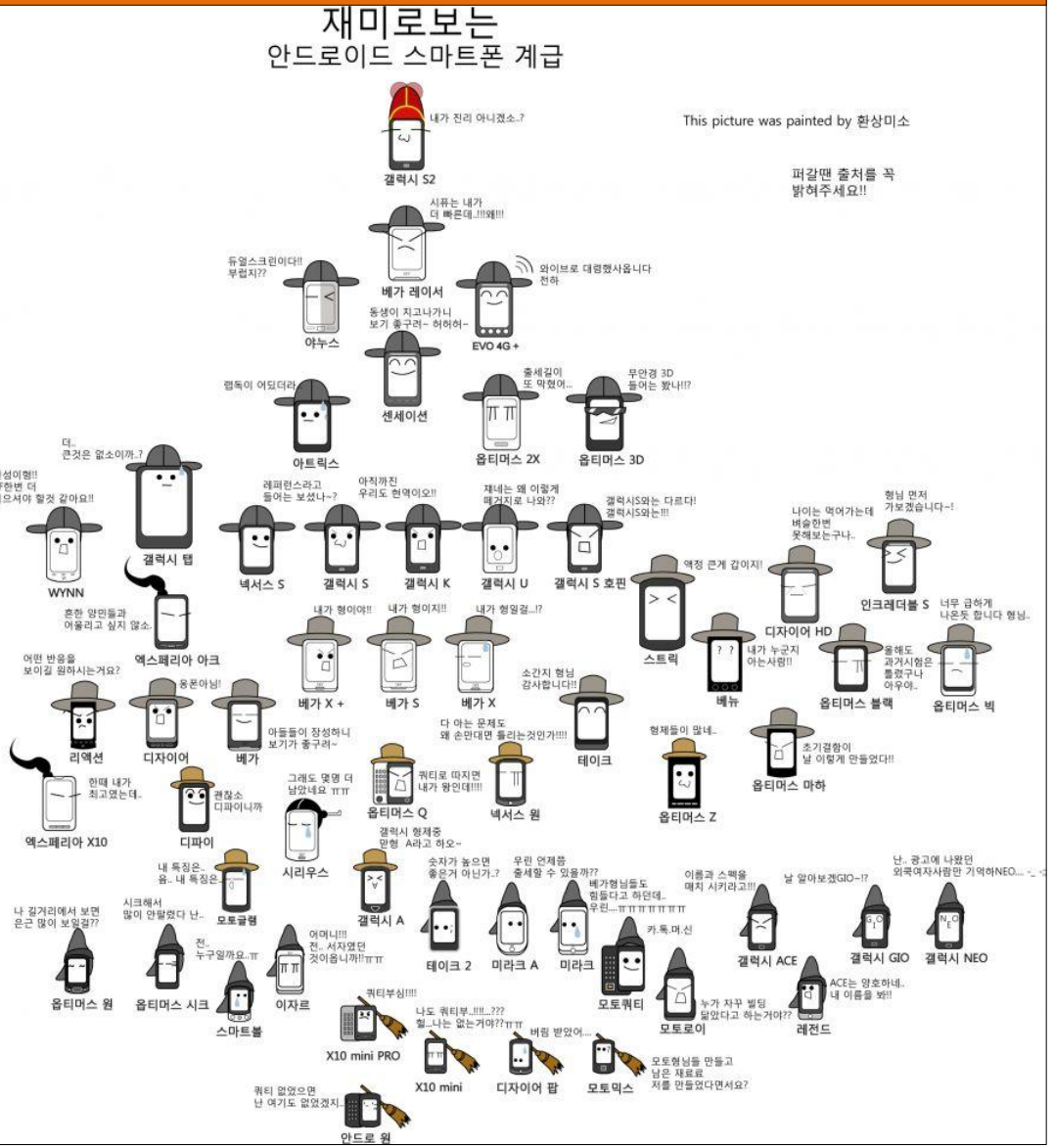
OpenGL-ES 에서의 문제점.

동일 platform, 동일 version, 그러나 **다른 GPU**

Problem

OpenGL-ES 에서의 문제점.

미칠 것 같이 다양한 디바이스들.



Problem

월
화
수
목
금
금
금

<야근>
오늘 야근이다
약속 있다구
데이트 좀하자
캬-악!
즐
그럼 주말에 나오든가
저녁 시킬까요?

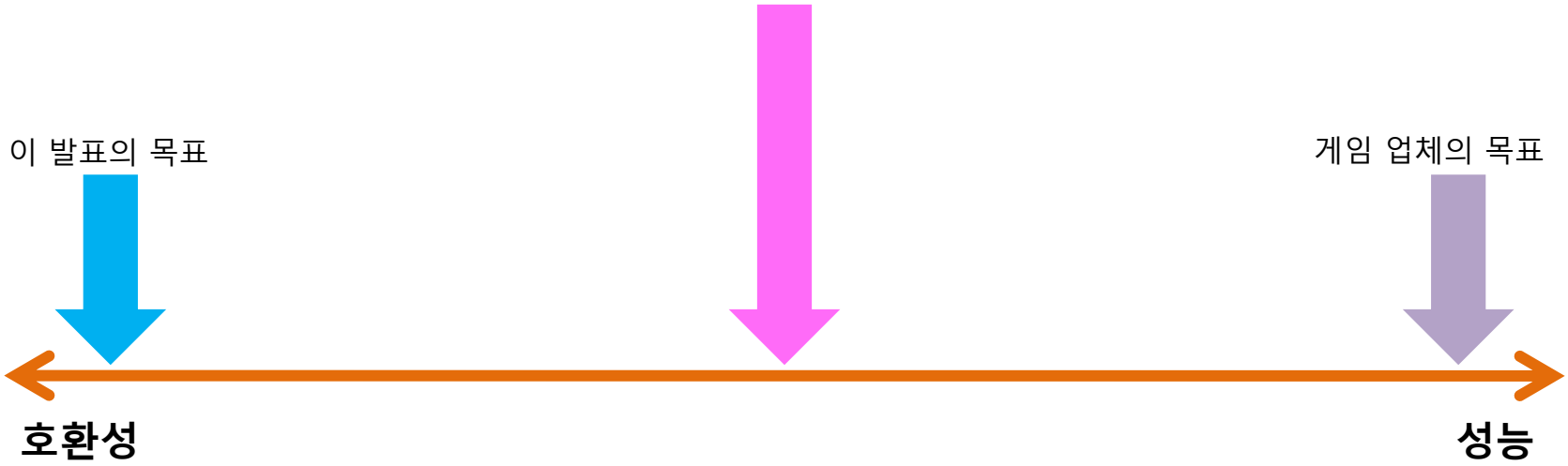
!?
저녁 6시니 퇴근해주마
아-앗
5시간은 더 야근하고 와라
와아앗--!
부장님!
neo-geta@hanmail.net

체력
김대리님, 저 집에 가고 싶어요
철야 4일째
나두..

야근
야근
우리는 무적의 야근부대이다
저녁 6시 이후에 집에 있는 것은 우리에겐 사치일 뿐이다

Problem

호환성/이식성 vs. 성능

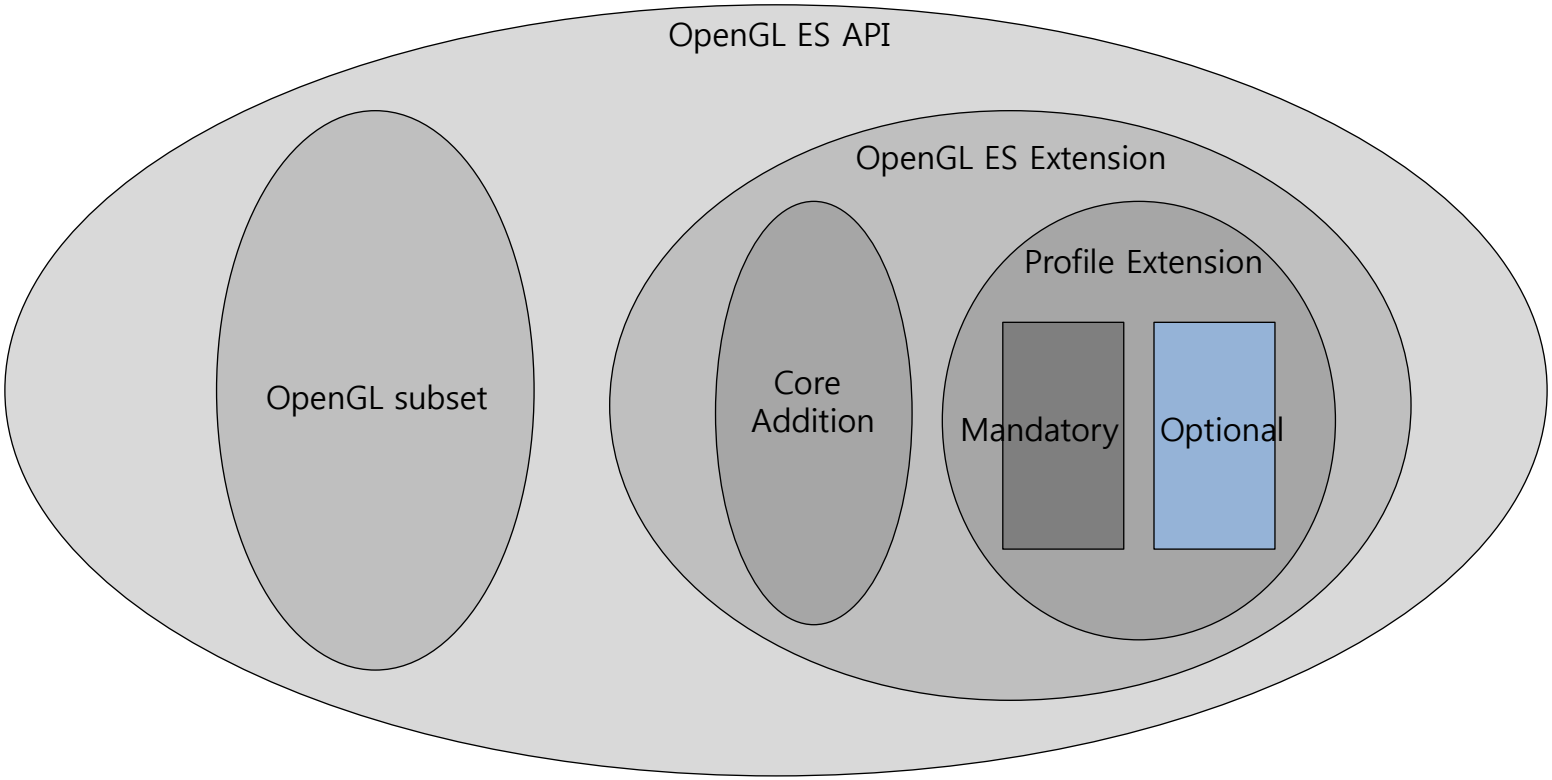


호환성 위협 요소

기기 의존적인 기능

특정 device를 기준으로 개발하는 경우 다른 device에서의 동작 호환성을 고려하지 않으면,
동일 application package를 타 device에 설치하였을 때 정상 동작을 보장할 수 없다.

OpenGL-ES Extensions



호환성 위협 요소

기기 의존적인 기능

OpenGL-ES Extensions

- Extension Specifications (http://www.khronos.org/registry/gles/)

GL_OES_blend_equation_separate
GL_OES_blend_func_separate
GL_OES_blend_subtract
GL_OES_byte_coordinates
GL_OES_compressed_ETC1_RGB8_texture
GL_OES_compressed_paletted_texture
GL_OES_draw_texture
GL_OES_extended_matrix_palette
GL_OES_fixed_point
GL_OES_framebuffer_object
GL_OES_matrix_get
GL_OES_matrix_palette
GL_OES_point_size_array
GL_OES_point_sprite
GL_OES_query_matrix
GL_OES_read_format
GL_OES_single_precision
GL_OES_stencil_wrap
GL_OES_texture_cube_map
GL_OES_texture_env_crossbar
GL_OES_texture_mirrored_repeat
GL_OES_EGL_image
GL_OES_depth24
GL_OES_depth32
GL_OES_element_index_uint
GL_OES_fbo_render_mipmap
GL_OES_fragment_precision_high
GL_OES_mapbuffer
GL_OES_rgb8_rgba8
GL_OES_stencil1
GL_OES_stencil4
GL_OES_stencil8

GL_OES_texture_3D
GL_OES_texture_float_linear
GL_OES_texture_half_float_linear
GL_OES_texture_float
GL_OES_texture_half_float
GL_OES_texture_npot
GL_OES_vertex_half_float
GL_AMD_compressed_3DC_texture
GL_AMD_compressed_ATC_texture
GL_EXT_texture_filter_anisotropic
GL_EXT_texture_type_2_10_10_10_REV
GL_OES_depth_texture
GL_OES_packed_depth_stencil
GL_OES_standard_derivatives
GL_OES_vertex_type_10_10_10_2
GL_OES_get_program_binary
GL_AMD_program_binary_Z400
GL_EXT_texture_compression_dxt1
GL_AMD_performance_monitor
GL_EXT_texture_format_BGRA8888
GL_NV_fence
GL_IMG_read_format
GL_IMG_texture_compression_pvrtc
GL_QCOM_driver_control
GL_QCOM_performance_monitor_global_mode
GL_IMG_user_clip_plane
GL_IMG_texture_env_enhanced_fixed_function
GL_APPLE_texture_2D_limited_npot
GL_EXT_texture_lod_bias
GL_QCOM_writeonly_rendering
GL_QCOM_extended_get
GL_QCOM_extended_get2

GL_EXT_discard_framebuffer
GL_EXT_blend_minmax
GL_EXT_read_format_bgra
GL_IMG_program_binary
GL_IMG_shader_binary
GL_EXT_multi_draw_arrays
GL_SUN_multi_draw_arrays
GL_QCOM_tiled_rendering
GL_OES_vertex_array_object
GL_NV_coverage_sample
GL_NV_depth_nonlinear
GL_IMG_multisampled_render_to_texture
GL_OES_EGL_sync
GL_APPLE_rgb_422
GL_EXT_shader_texture_lod
GL_APPLE_framebuffer_multisample
GL_APPLE_texture_format_BGRA8888
GL_APPLE_texture_max_level
GL_ARM_mali_shader_binary
GL_ARM_rgba8
GL_ANGLE_framebuffer_blit
GL_ANGLE_framebuffer_multisample
GL_VIV_shader_binary
GL_EXT_frag_depth
GL_OES_EGL_image_external
GL_DMP_shader_binary
GL_QCOM_alpha_test
GL_EXT_unpack_subimage
GL_NV_draw_buffers
GL_NV_fbo_color_attachments
GL_NV_read_buffer
GL_NV_read_depth_stencil
GL_NV_texture_compression_s3tc_update
GL_NV_texture_npot_2D_mipmap

호환성 위협 요소

기기 의존적인 기능

기기의 특성 차이

• 화면 크기 및 aspect ratio의 차이.

Display: 320 x 480
AspectRatio is **2:3**

Display: 640 x 960
AspectRatio is **2:3**

iOS

Display: 480 x 800
AspectRatio is **3:5**

Display: 480 x 800
AspectRatio is **3:5**

Display: 600 x 1024
AspectRatio is **1:1.7066**

Display: 320 x 480
AspectRatio is **2:3**

Display: 768 x 1280
AspectRatio is **3:5**

Display: 800 x 1280
AspectRatio is **1:1.6**

android

Display: 480 x 800
AspectRatio is **3:5**

Display: 480 x 800
AspectRatio is **3:5**

bada

호환성 위협 요소

기기 의존적인 기능

기기의 특성 차이

• Graphics Hardware의 차이

SGX535
(Imagination co.)

SGX540
(Imagination co.)

SGX540
(Imagination co.)

Mali-400MP
(ARM)

SGX540
(Imagination co.)

SGX535
(Imagination co.)

Mali-400MP
(ARM)

MSM7227
(Qualcomm)

Tegra2 T20
(Nvidia)

Adreno205
(Qualcomm)

iOS

android

bada

호환성 위협 요소

기기 의존적인 기능

기기의 특성 차이

• Graphics resource 차이 – Texture memory

Appendix : Wave Series


256MB 공유


512MB 공유

iOS


???


???


???


???


???


???

android


app당 32MB


app당 32MB

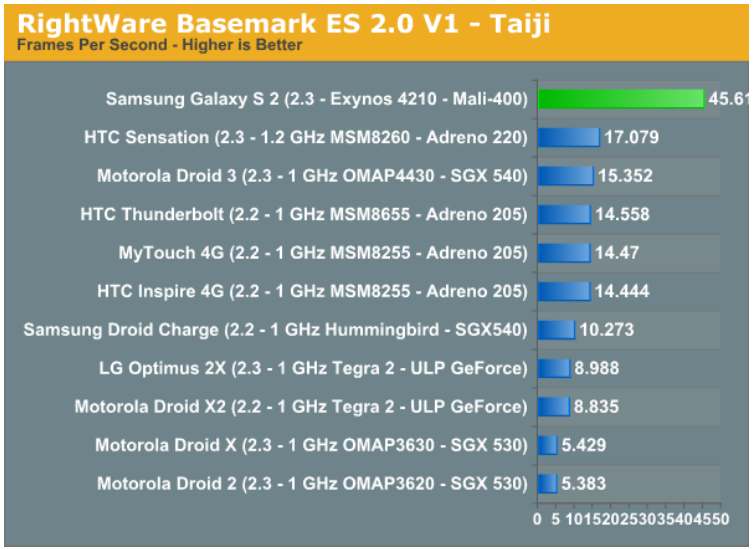
bada

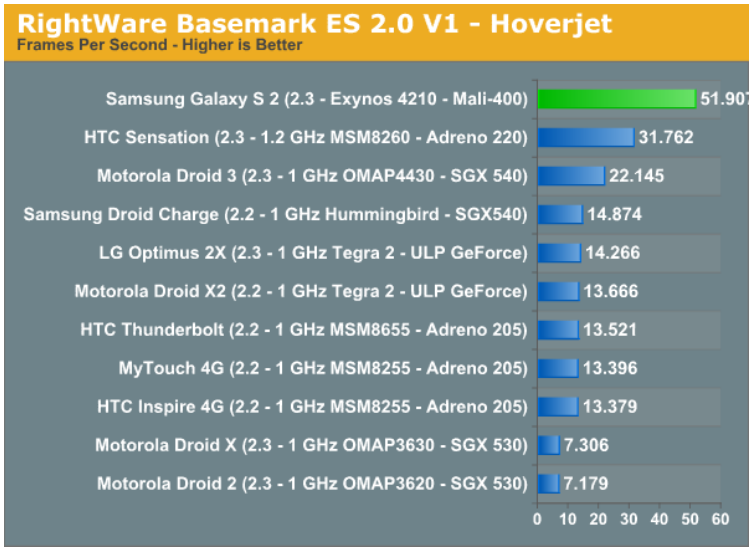
호환성 위협 요소

기기 의존적인 기능

기기의 특성 차이

- Graphics hardware의 성능 차이.



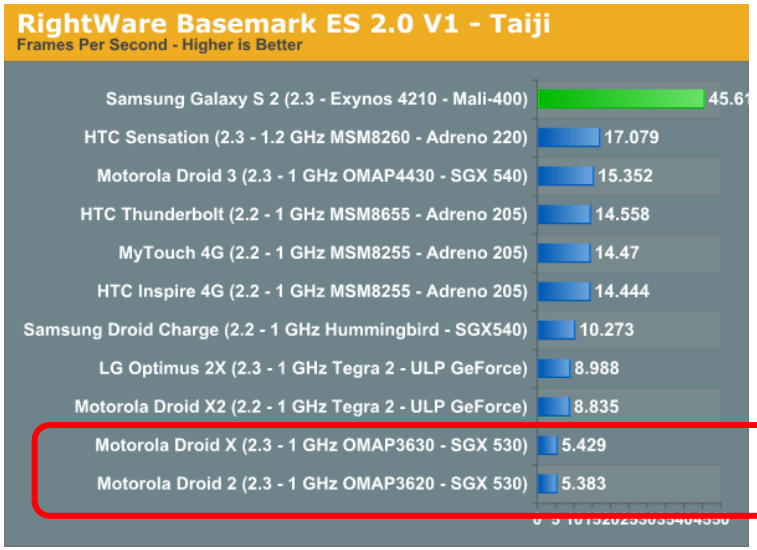


호환성 위협 요소

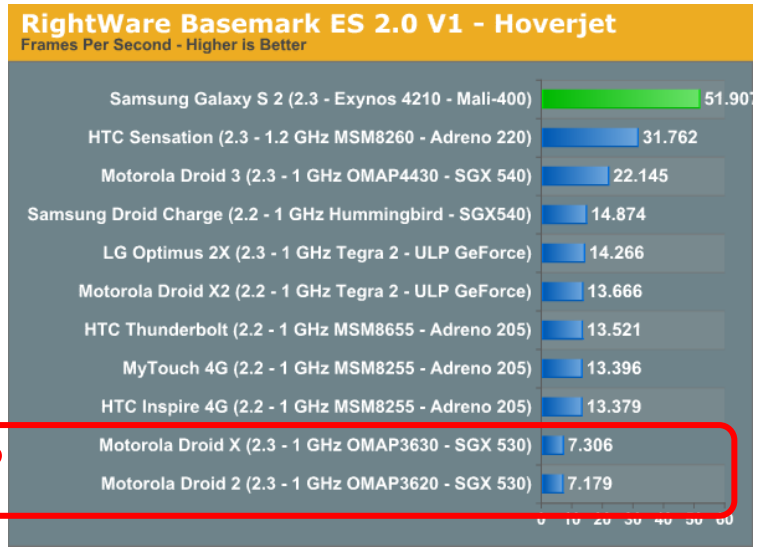
기기 의존적인 기능

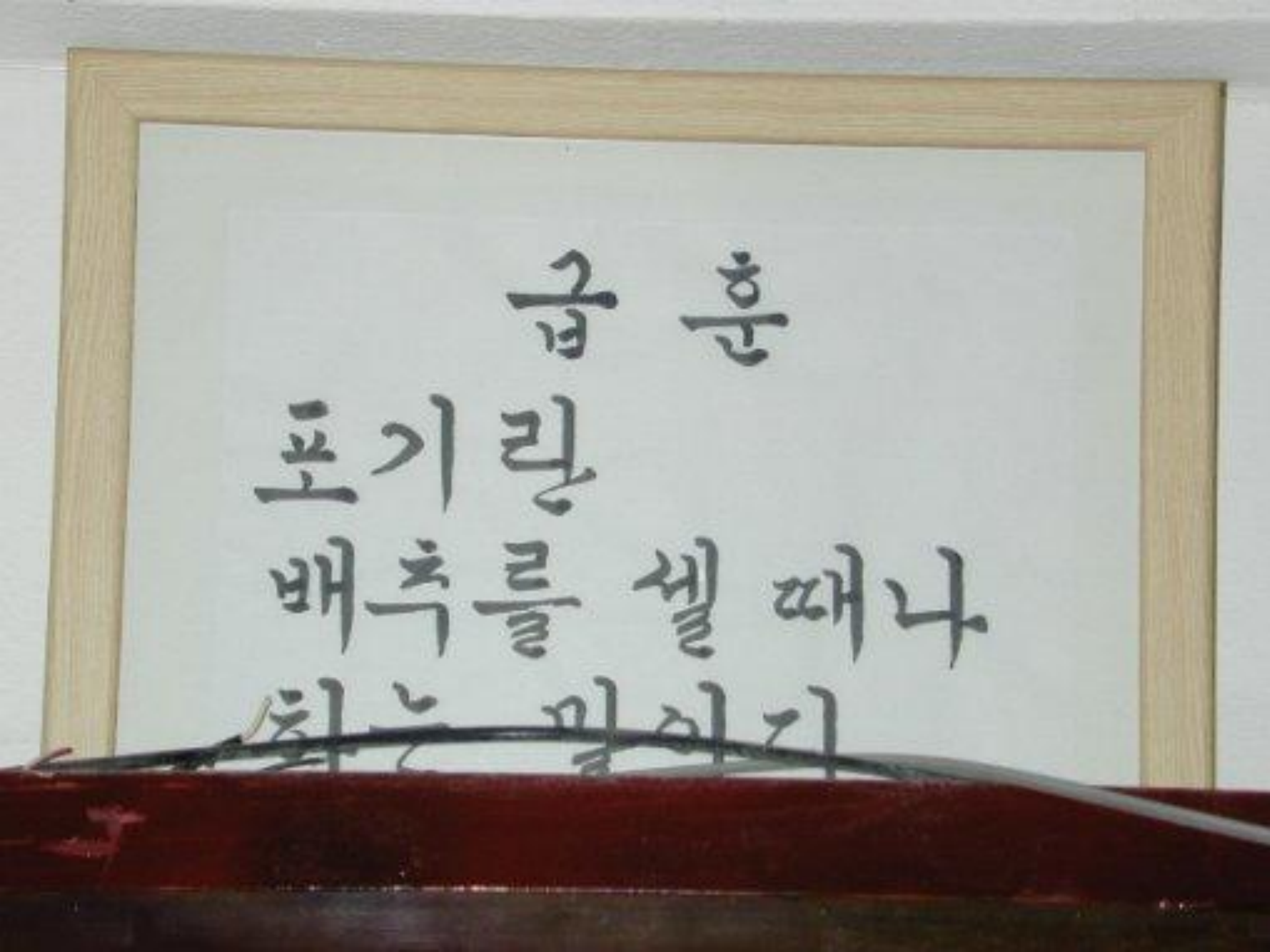
기기의 특성 차이

- Graphics hardware의 성능 차이.



포기?



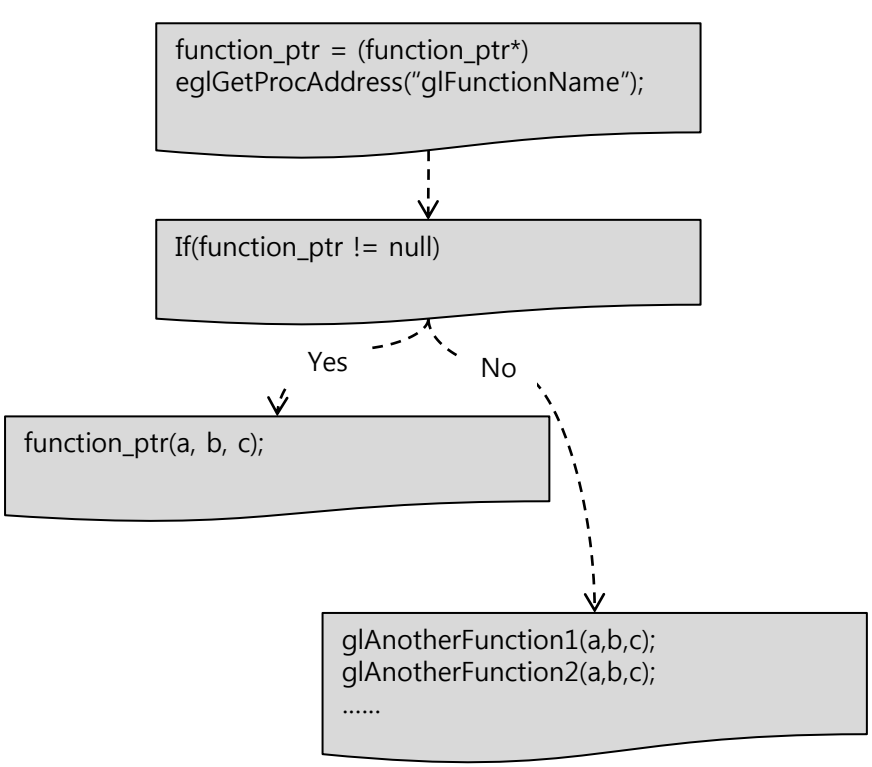

금 훈
포기란
배추를 셀 때나

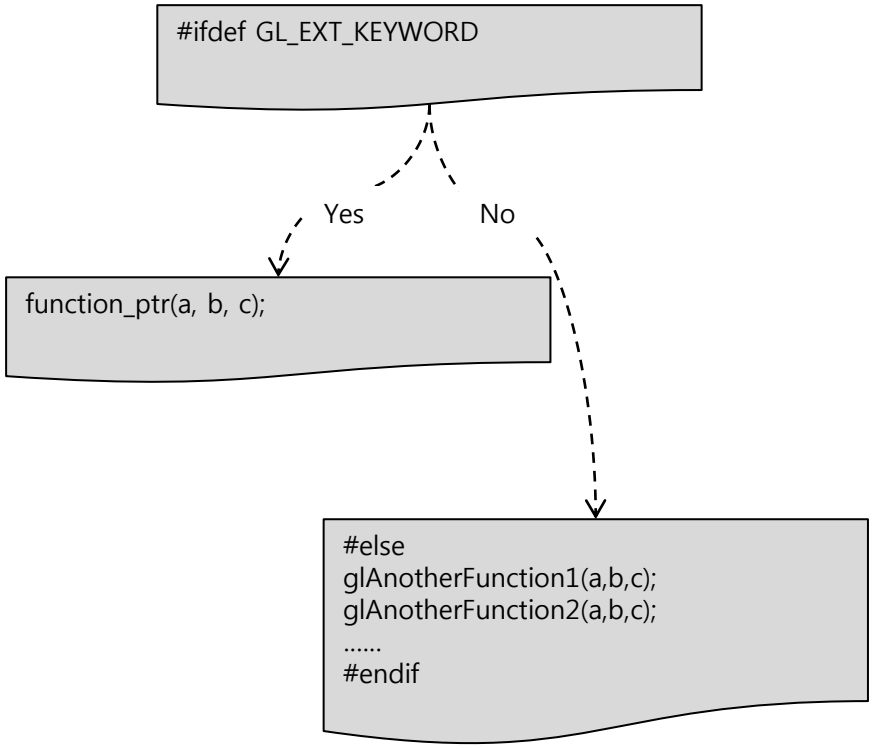
호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

OpenGL-ES Extension 기능 중 특정 device에서만 지원하는 기능들에 대해서는 사용에 주의한다.

- Best solution
  - 되도록 OpenGL-ES Extension을 사용하지 않으며, 특히 optional 또는 hardware dependent feature 사용시 주의한다.
- Plan B
  - OpenGL-ES extension 사용 전, 해당 feature가 device에서 지원하는지를 **체크**하고 지원하는 경우에만 기능을 사용.
  - Device에서 지원하지 않는 경우에는 반드시 **우회방법**을 구현하여 app실행에 문제 해야만 한다.



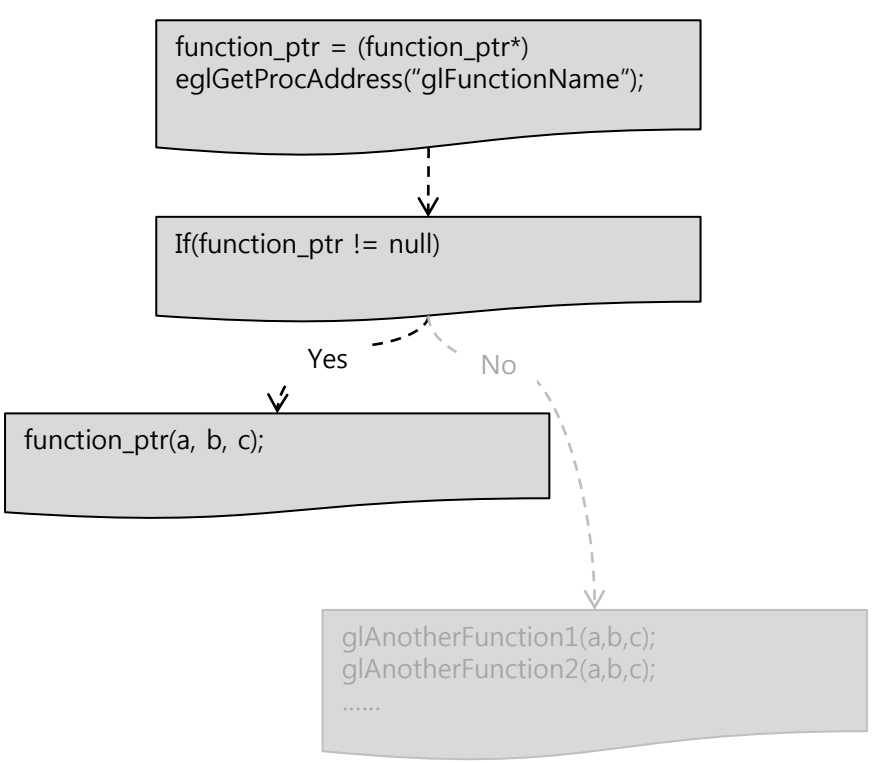


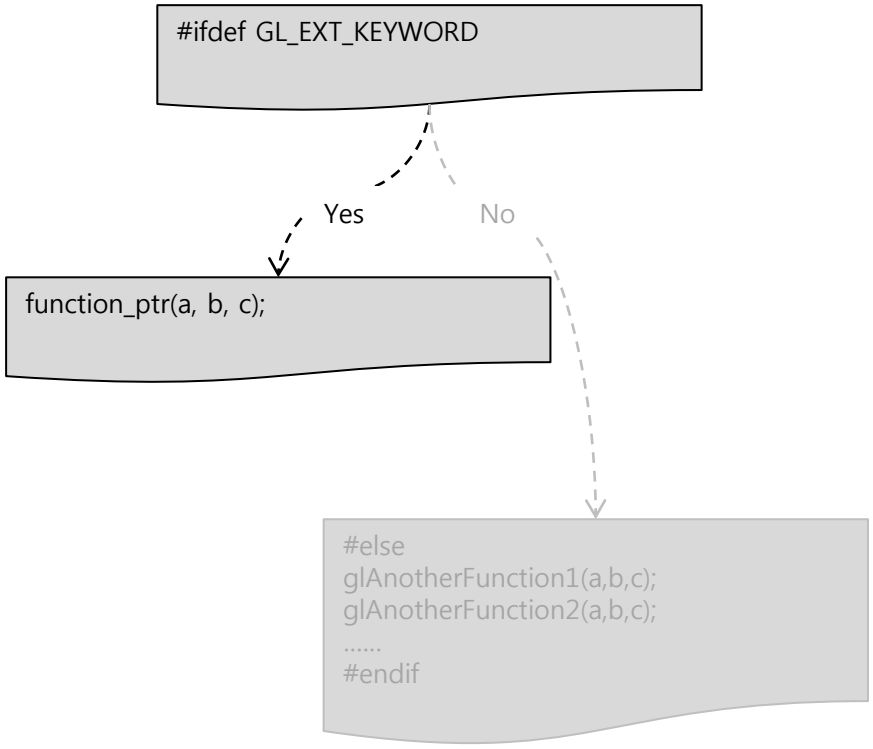
호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

OpenGL-ES Extension 기능 중 특정 device에서만 지원하는 기능들에 대해서는 사용에 주의한다.

- Best solution
  - 되도록 OpenGL-ES Extension을 사용하지 않으며, 특히 optional 또는 hardware dependent feature 사용시 주의한다.
- Plan B
  - OpenGL-ES extension 사용 전, 해당 feature가 device에서 지원하는지를 **체크**하고 지원하는 경우에만 기능을 사용.
  - Device에서 지원하지 않는 경우에는 반드시 **우회방법**을 구현하여 app실행에 문제 해야만 한다.



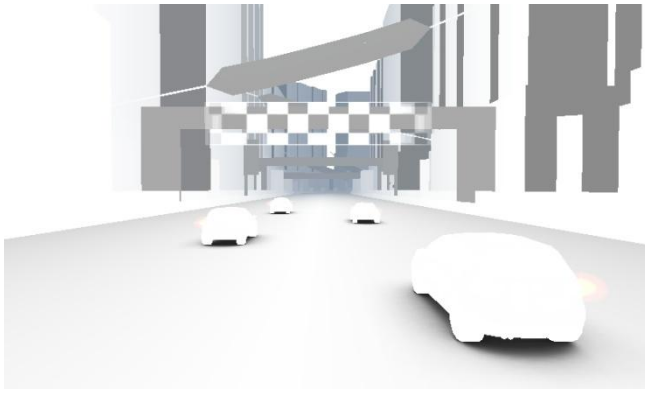


호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

Compressed texture format
- **ETC** (Ericsson Texture Compression) by Ericsson.
- **PVRTC** (PowerVR Texture Compression) by Imagination
- **ATC** (ATI Texture Compression) and **3DC** by Qualcomm

Wave2 with PVRTC texture

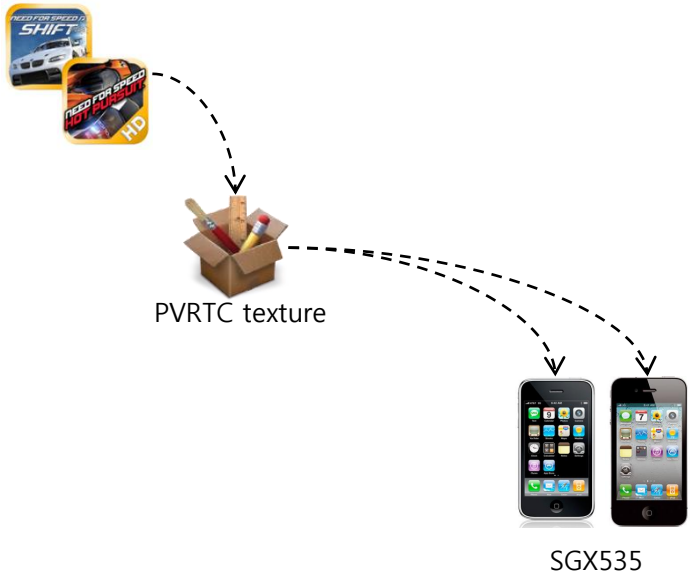


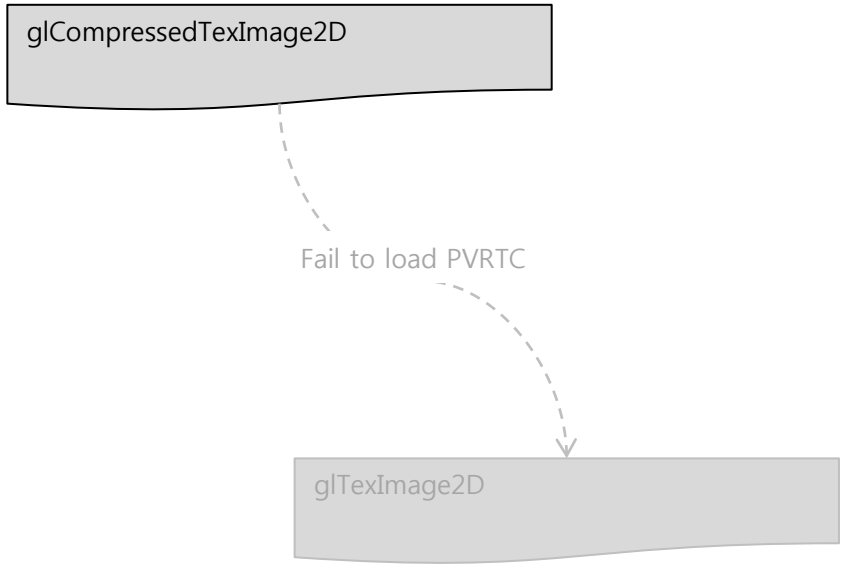
Wave3 with PVRTC texture

호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

Compressed texture format
- **ETC** (Ericsson Texture Compression) by Ericsson.
- **PVRTC** (PowerVR Texture Compression) by Imagination
- **ATC** (ATI Texture Compression) and **3DC** by Qualcomm

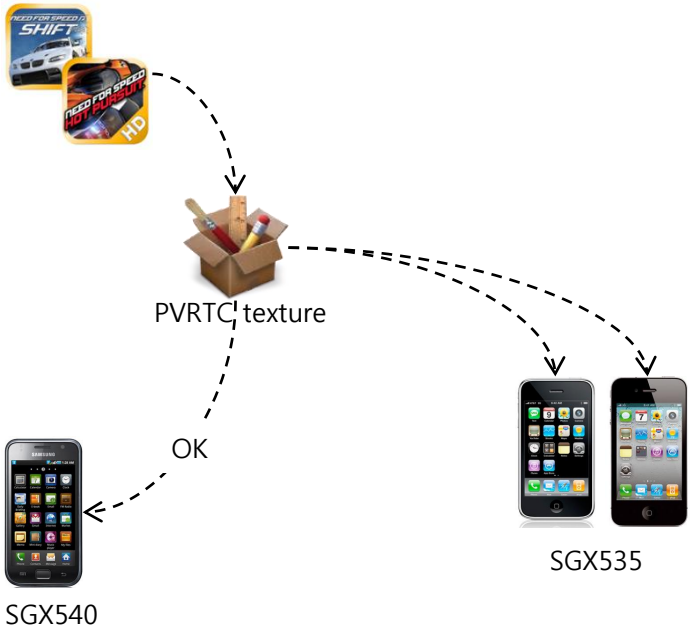


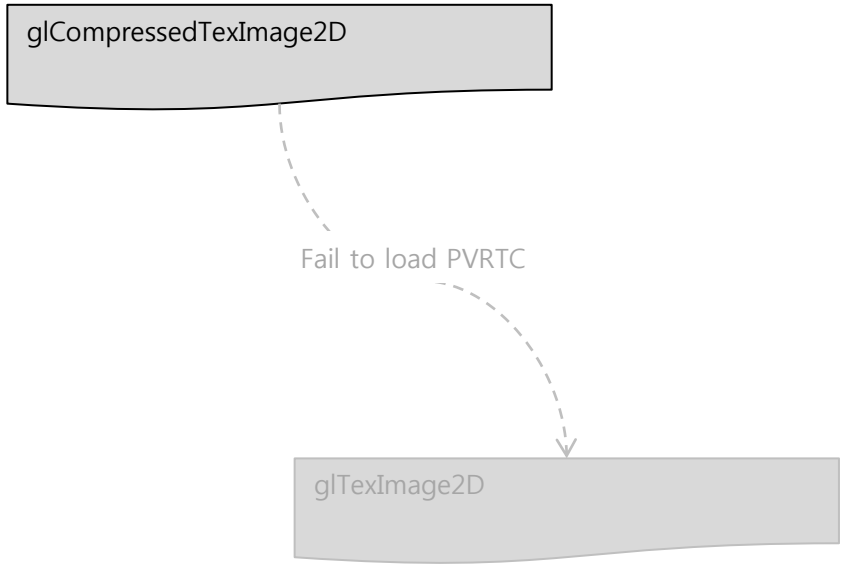


호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

Compressed texture format
- **ETC** (Ericsson Texture Compression) by Ericsson.
- **PVRTC** (PowerVR Texture Compression) by Imagination
- **ATC** (ATI Texture Compression) and **3DC** by Qualcomm

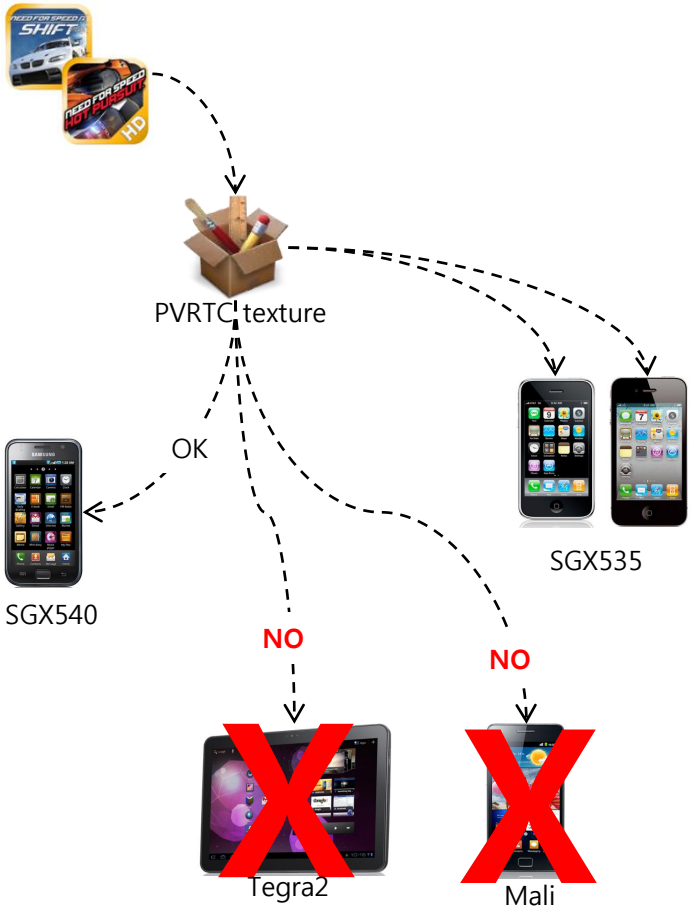


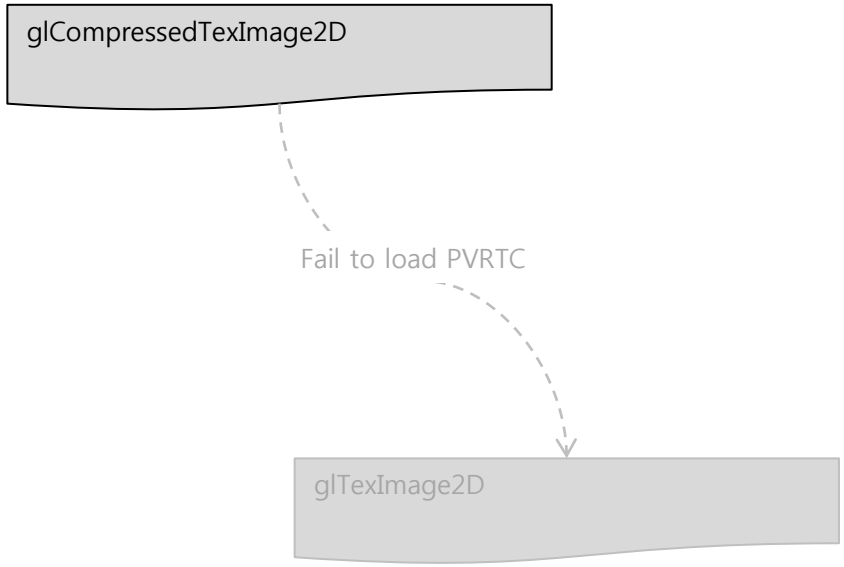


호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

Compressed texture format
- **ETC** (Ericsson Texture Compression) by Ericsson.
- **PVRTC** (PowerVR Texture Compression) by Imagination
- **ATC** (ATI Texture Compression) and **3DC** by Qualcomm

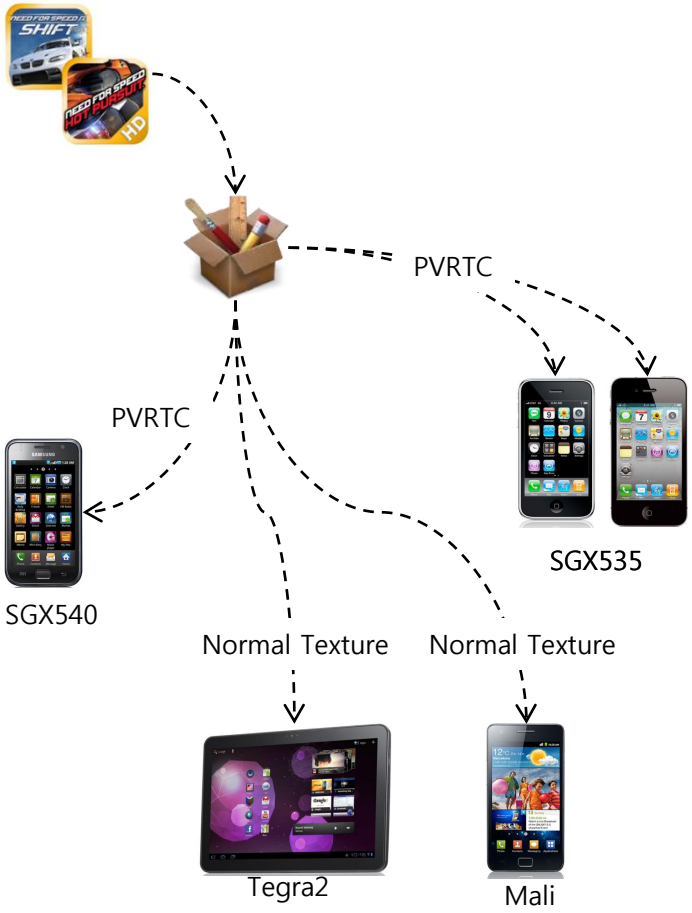


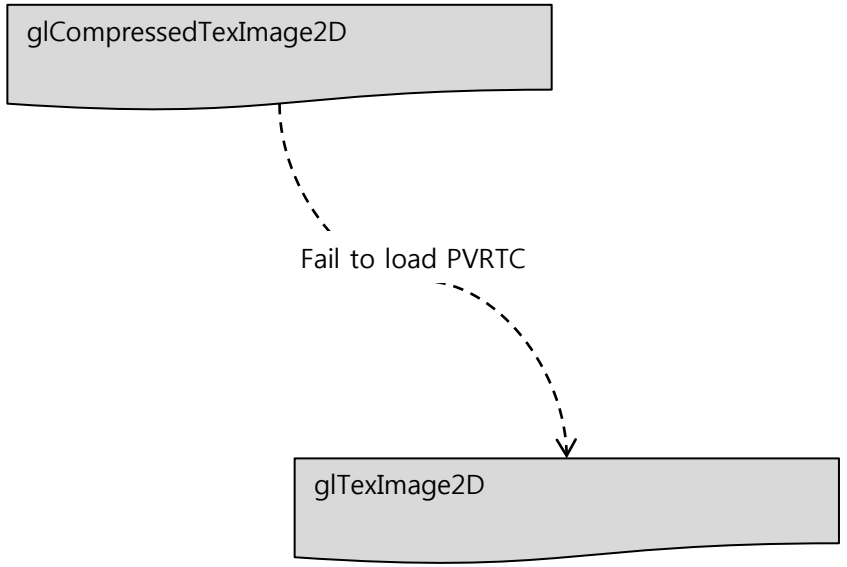


호환성을 위한 고려사항

OpenGL-ES Extension 사용에 유의할 것.

Compressed texture format
- **ETC** (Ericsson Texture Compression) by Ericsson.
- **PVRTC** (PowerVR Texture Compression) by Imagination
- **ATC** (ATI Texture Compression) and **3DC** by Qualcomm





Appendix : Application Filtering by Market

호환성을 위한 고려사항

하드웨어 특성에 유의할 것.

Graphics hardware가 가지는 하드웨어적 특성에 의해 동작이나 rendering 결과에 영향을 받는 기능들은 주의해서 사용하도록 한다.

• POT(Power-Of-Two) texture 만을 지원하는 하드웨어를 고려하여 app을 제작할 것.

POT (Power-Of-Two) Texture

2^n

2^m

- 특징
  NPOT texture로 texture mapping을 시도하면 정상적으로 texture가 그려지지 않은 결과를 얻게 된다.

NPOT (Non-Power-Of-Two) Texture

b

a

- 특징
  POT/NPOT texture에 대해서 모든 경우 정상적인 rendering 결과를 얻을 수 있다.

호환성을 위한 고려사항

하드웨어 특성에 유의할 것.

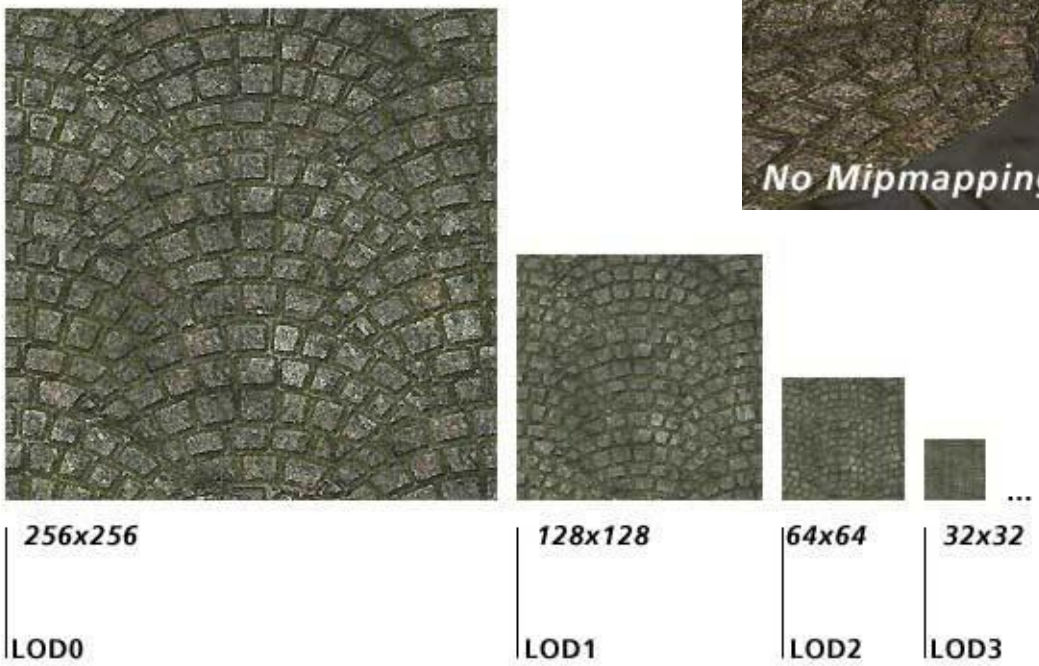
Graphics hardware가 가지는 하드웨어적 특성에
도록 한다.
• POT(Power-Of-Two) texture 만을 지원하는 하드

POT (Power-Of-Two) Texture

2^n

2^m

- 특징
  NPOT texture로 texture mappin
면 정상적으로 texture가 그려지지
르 어게 된다



호환성을 위한 고려사항

하드웨어 특성에 유의할 것.

Graphics hardware가 가지는 하드웨어적 특성에 의해 동작이나 rendering 결과에 영향을 받는 기능들은 주의해서 사용하도록 한다.

• POT(Power-Of-Two) texture 만을 지원하는 하드웨어를 고려하여 app을 제작할 것.

POT (Power-Of-Two) Texture

- 특징
  NPOT texture로 texture mapping을 시도하면 정상적으로 texture가 그려지지 않은 결과를 얻게 된다.

- Mipmap
  곧바로 Mipmap 적용 가능

NPOT (Non-Power-Of-Two) Texture

- 특징
  POT/NPOT texture에 대해서 모든 경우 정상적인 rendering 결과를 얻을 수 있다.

- Mipmap
  Mipmap을 적용하기 위해서 texture size를 2의 승수가 되도록 맞추어 주어야만 함.

호환성을 위한 고려사항

하드웨어 특성에 유의할 것.

Graphics hardware가 가지는 하드웨어적 특성에 의해 동작이나 rendering 결과에 영향을 받는 기능들은 주의해서 사용하도록 한다.

• POT(Power-Of-Two) texture 만을 지원하는 하드웨어를 고려하여 app을 제작할 것.

POT (Power-Of-Two) Texture

2^n

2^m

- 특징
  NPOT texture로 texture mapping을 시도하면 정상적으로 texture가 그려지지 않은 결과를 얻게 된다.

- Mipmap
  곧바로 Mipmap 적용 가능

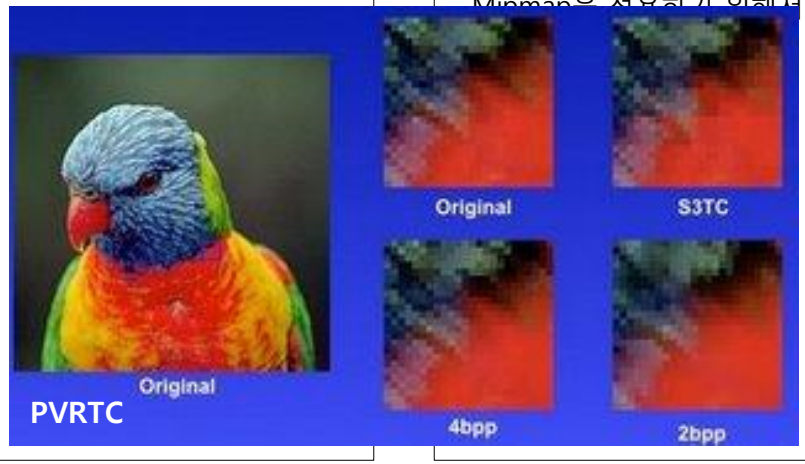
NPOT (Non-Power-Of-Two) Texture

b

a

- 특징
  POT/NPOT texture에 대해서 모든 경우 정상적인 rendering 결과를 얻을 수 있다.

- Mipmap
  texture size를 2의 승수가 되도록 맞



호환성을 위한 고려사항

하드웨어 특성에 유의할 것.

Graphics hardware가 가지는 하드웨어적 특성에 의해 동작이나 rendering 결과에 영향을 받는 기능들은 주의해서 사용하도록 한다.
• POT(Power-Of-Two) texture 만을 지원하는 하드웨어를 고려하여 app을 제작할 것.

POT (Power-Of-Two) Texture

- 특징
  NPOT texture로 texture mapping을 시도하면 정상적으로 texture가 그려지지 않은 결과를 얻게 된다.

- Mipmap
  곧바로 Mipmap 적용 가능

- Texture compression
  대부분의 texture compression 알고리즘은 texture size가 2의 승수라고 가정하고 동작하기 때문에 compressed texture를 곧바로 GPU로 보내어 texture mapping이 가능.

NPOT (Non-Power-Of-Two) Texture

- 특징
  POT/NPOT texture에 대해서 모든 경우 정상적인 rendering 결과를 얻을 수 있다.

- Mipmap
  Mipmap을 적용하기 위해서 texture size를 2의 승수가 되도록 맞추어 주어야만 함.

- Texture compression
  GPU에서 지원하는 compressed texture format을 사용할 수 없기 때문에 texture사용 직전에 image를 복구하여 texture mapping을 수행.

호환성을 위한 고려사항

하드웨어 특성에 유의할 것.

Graphics hardware가 가지는 하드웨어적 특성에 의해 동작이나 rendering 결과에 영향을 받는 기능들은 주의해서 사용하도록 한다.

• POT(Power-Of-Two) texture 만을 지원하는 하드웨어를 고려하여 app을 제작할 것.

POT (Power-Of-Two) Texture

NPOT (Non-Power-Of-Two) Texture

호환성을 위한 고려사항

Shader programming시 유의할 것.

Pre-compiled shader binary 사용 지양.

Shader source code를 text 형태로 제공하여 실행시 shader souce를 빌드하여 GPU에 보내는 방법 대신, offline 작업을 통해 생성된 shader binary를 곧바로 GPU에 보내는 방법.

```
glShaderBinary(1, frag_shader, frag_bin_format, frag_binary, null);
glShaderSource(frag_shader, 1, frag_source, null);
glCompileShader(frag_shader);
glShaderSource(vert_shader, 1, vert_source, null);
glCompileShader(vert_shader);
program = glCreateProgram();
glAttachShader(program, frag_shader);
……
```

특징 및 장점
- shader program을 실시간으로 빌드하여 생성하기 때문에, shader program만 올바르게 작성되었다면 graphics hardware에 상관없이 shader binary를 생성할 수 있다.
- 실행 시 mobile device를 참조하여 shader engine의 arithmetic precision을 결정할 수 있다.

- **기기간 이식성이 높다.** ☺

단점
- text source code를 가지고 있어야 하기 때문에 더 많은 storage와 memory, 그리고 memory bandwidth를 차지한다.
- shader 실행 시 항상 shader binary를 빌드하여 생성하기 위한 시간이 필요하다.

```
glShaderBinary(1, frag_shader, frag_bin_format, frag_binary, null);
glShaderBinary(1, vert_shader, vert_bin_format, vert_binary, null);
program = glCreateProgram();
glAttachShader(program, frag_shader);
……
```

특징 및 장점
- application 실행 과정에서 shader program의 빌드가 필요하지 않기 때문에, application 실행 시 수행속도를 높일 수 있다.
- text source code가 아닌 binary 결과물만을 가지고 있기 때문에, memory bandwidth와 storage 측면에서 크게 이득을 얻을 수 있다.

단점
- shader compiler는 graphics hardware 제작사에 따라 다르고, 동일 제작사의 제품이더라도 driver 버전에 따라 호환되지 못하는 경우가 있다. 따라서 특정 device 향으로 pre-compiled된 shader binary (일반적으로 SGX시리즈)의 경우 타기기에서 shader program이 생성되지 못하는 경우가 발생한다.
- binary 생성 시 결정된 arithmetic precision만을 사용.

- **기기간 이식성이 낮다.** ☹

호환성을 위한 고려사항

Shader programming시 유의할 것.

Shader program의 표준 문법 준수

아래 fragment shader source code에서의 문제점은?

Fragment shader source code from ConformanceTest11

```
#ifdef GL_ES
precision mediump float;
#endif

//mutiple line macros - test case.

#define test 5
#define t1 1
#define t2 2
#define token (t1+t2)
#define test1 int sum =1; sum = test; sum = test+test;

#define test2 do{ test1 sum = sum +token; sum = t2*t1; }while(sum<=0)

void main(void)
{
    int test3=1;
    test1
    test2;
    test3 = test;
    sum = test3;
}
```

호환성을 위한 고려사항

Shader programming시 유의할 것.

Shader program의 표준 문법 준수

아래 fragment shader source code에서의 문제점은?

Fragment shader source code from ConformanceTest11

```
#ifdef GL_ES
precision mediump float;
#endif

//mutiple line macros - test case.

#define test 5
#define t1 1
#define t2 2
#define token (t1+t2)
#define test1 int sum =1; sum = test; sum = test+test;

#define test2 do{ test1 sum = sum +token; sum = t2*t1; }while(sum<=0)

void main(void)
{
    int test3=1;
    test1
    test2;
    test3 = test;
    sum = test3;
}
```

현재 OpenGL-ES2.0의 specification에 따르면, do-while loop은 선택적인 지원 대상으로 driver 제작사에게 구현 및 지원을 강제하지 않는다. 따라서, 해당 shader source code는 일부 shader compiler에서만 정상적으로 compile되고 대부분의 shader compiler에서는 compile되지 않고 compile error를 발생시킨다.

호환성을 위한 고려사항

Shader programming시 유의할 것.

Shader program의 표준 문법 준수

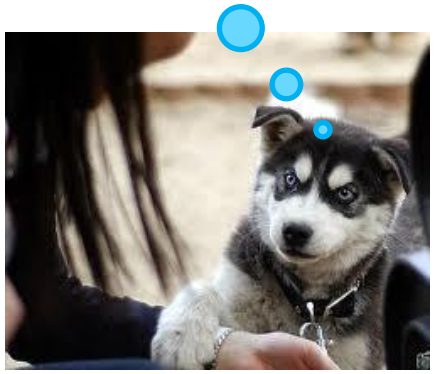
아래 fragment shader source code에서의 문제점은?

Fragment shader source code from ConformanceTest11

```
#ifdef GL_ES
precision mediump float;
#endif

//mutiple line macros - test case.

#define test 5
#define t1 1
#define t2 2
#define token (t1+t2)
#define test1 int sum =1; sum = test; sum = test+test;

#define test2 do{ test1 sum = sum +token; sum = t2*t1; }while(sum<=0)

void main(void)
{
    int test3=1;
    test1
    test2;
    test3 = test;
    sum = test3;
}
```

현재 OpenGL-ES2.0의 specification에 따르면, do-while loop은 선택적인 지원 대상으로 driver 제작사에게 구현 및 지원을 강제하지 않는다. 따라서, 해당 shader source code는 일부 shader compiler에서만 정상적으로 compile되고 대부분의 shader compiler에서는 compile되지 않고 compile error를 발생시킨다.

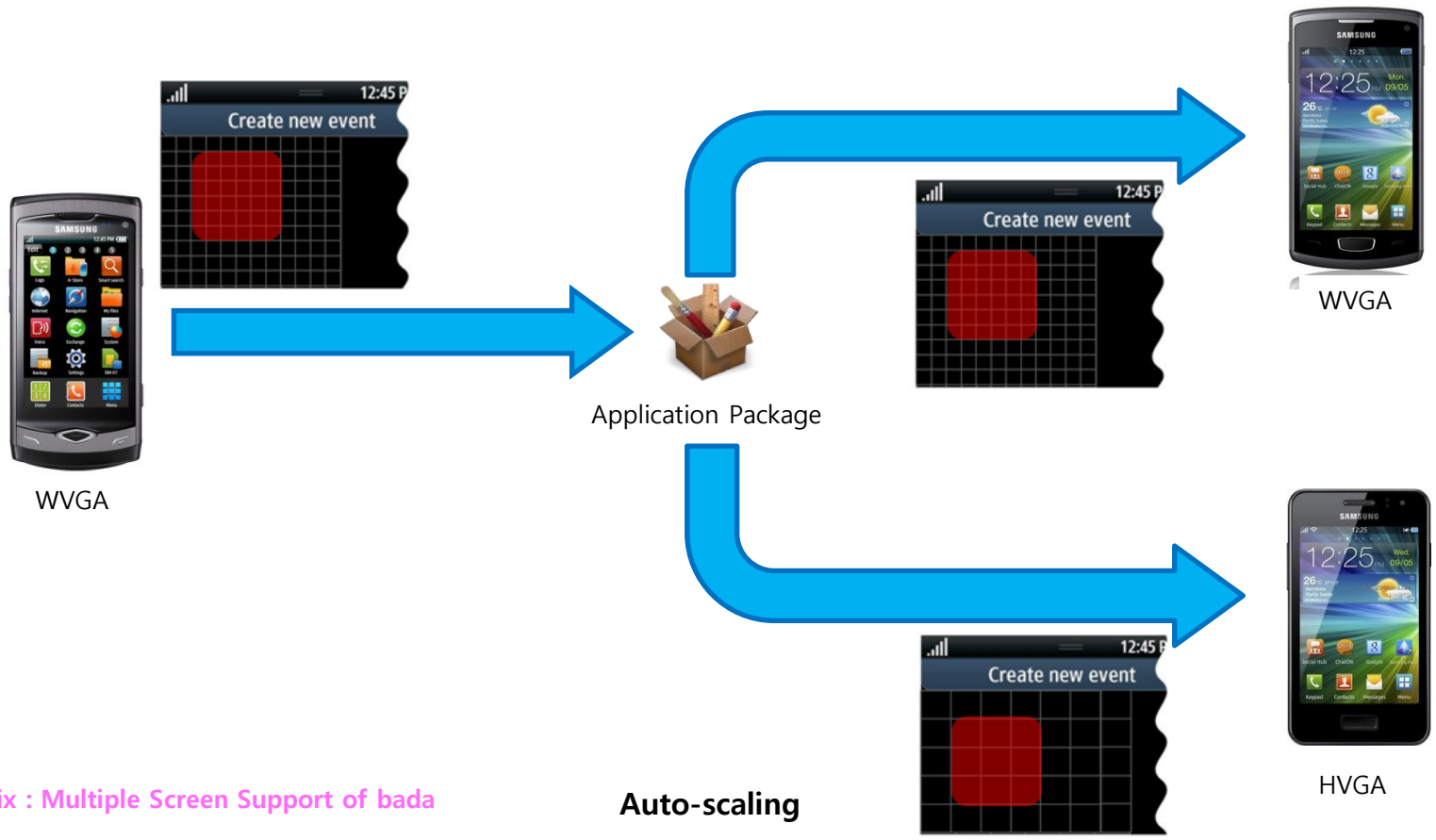
1. Specification상으로 driver 제작사에 구현을 강요하지 않는 선택적인 지원 사항들.
2. GPU/driver 제작사에서 특정 hardware에서 지원하는 기능을 위해서 정의한 shader program의 함수나 기능들. (eg. vector-wise operation들)



호환성을 위한 고려사항

OpenGL-ES API 사용시 Auto-scaling 기능 사용을 자제할 것.

bada 플랫폼에서의 Multiple screen support
- Device screen size나 aspect ratio에 상관없이 동일한 UX를 제공하기 위한 방법
- android: Supporting Multiple Screens 참조 (http://developer.android.com/guide/practices/screens_support.html)
- apple과는 달리 android와 bada의 경우 다양한 screen size를 가지는 mobile device들에 탑재되기 때문에, 한번 개발된 application이 모든 device들에서 동일한 UX를 제공하기 위해서는 multiple screen supporting을 지원해야만 한다.

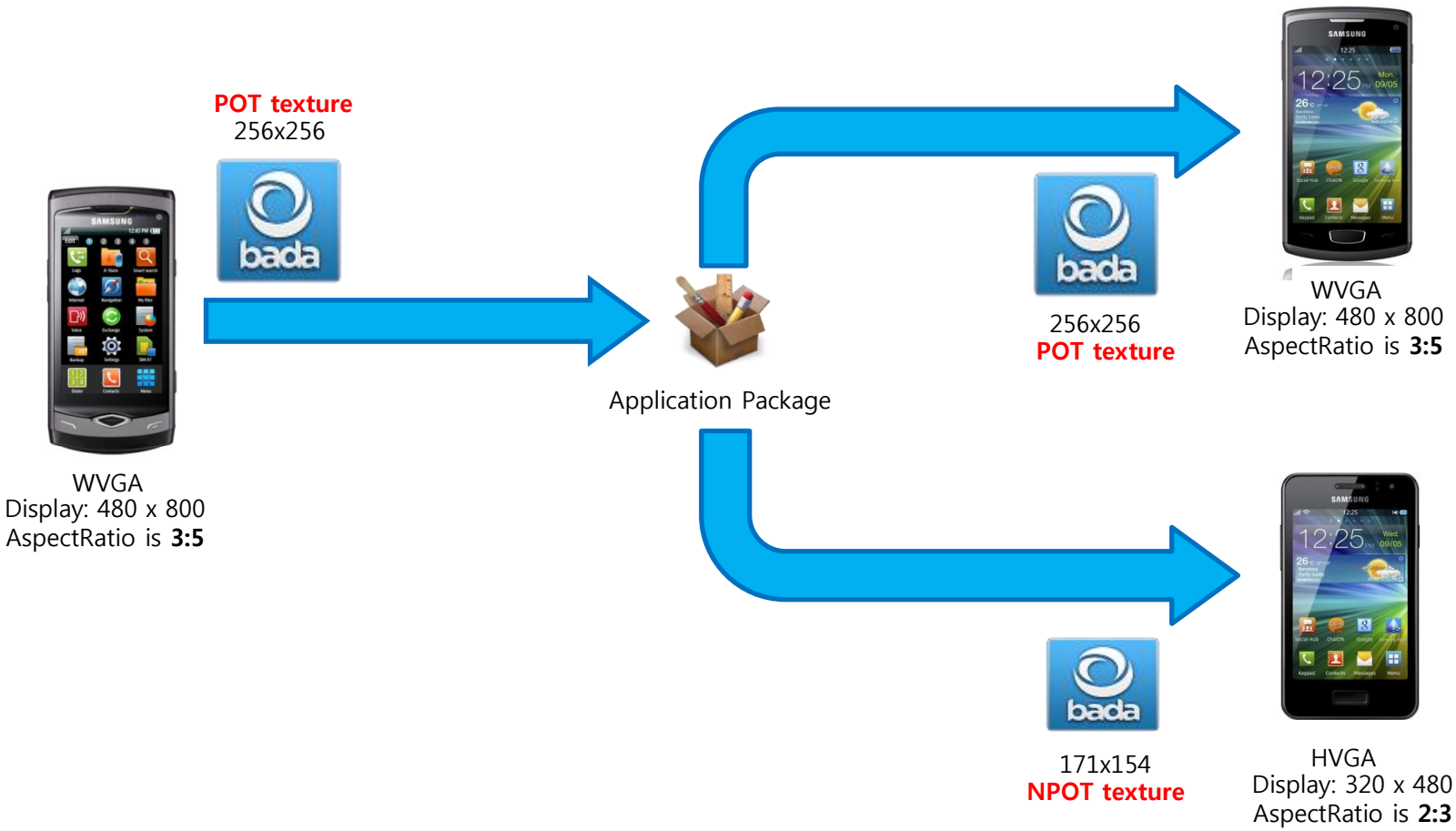


Auto-scaling

호환성을 위한 고려사항

OpenGL-ES API 사용시 Auto-scaling 기능 사용을 자제할 것.

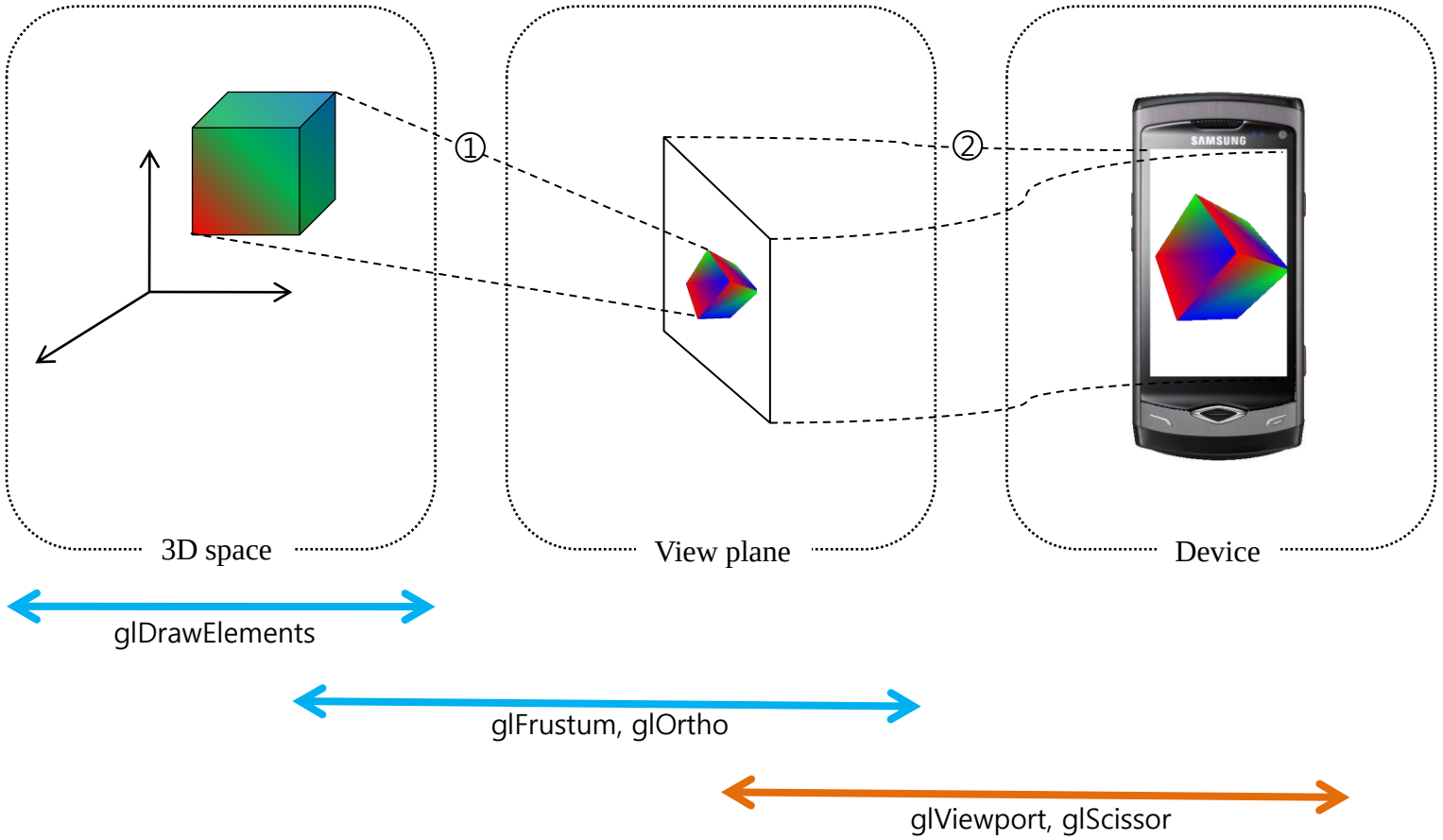
문제점
- Application 실행 시 loading된 Resource의 상태 (특히 texture의 크기)를 개발자가 개발하는 과정에서 알 수 없다.



호환성을 위한 고려사항

OpenGL-ES API 사용시 Auto-scaling 기능 사용을 자제할 것.

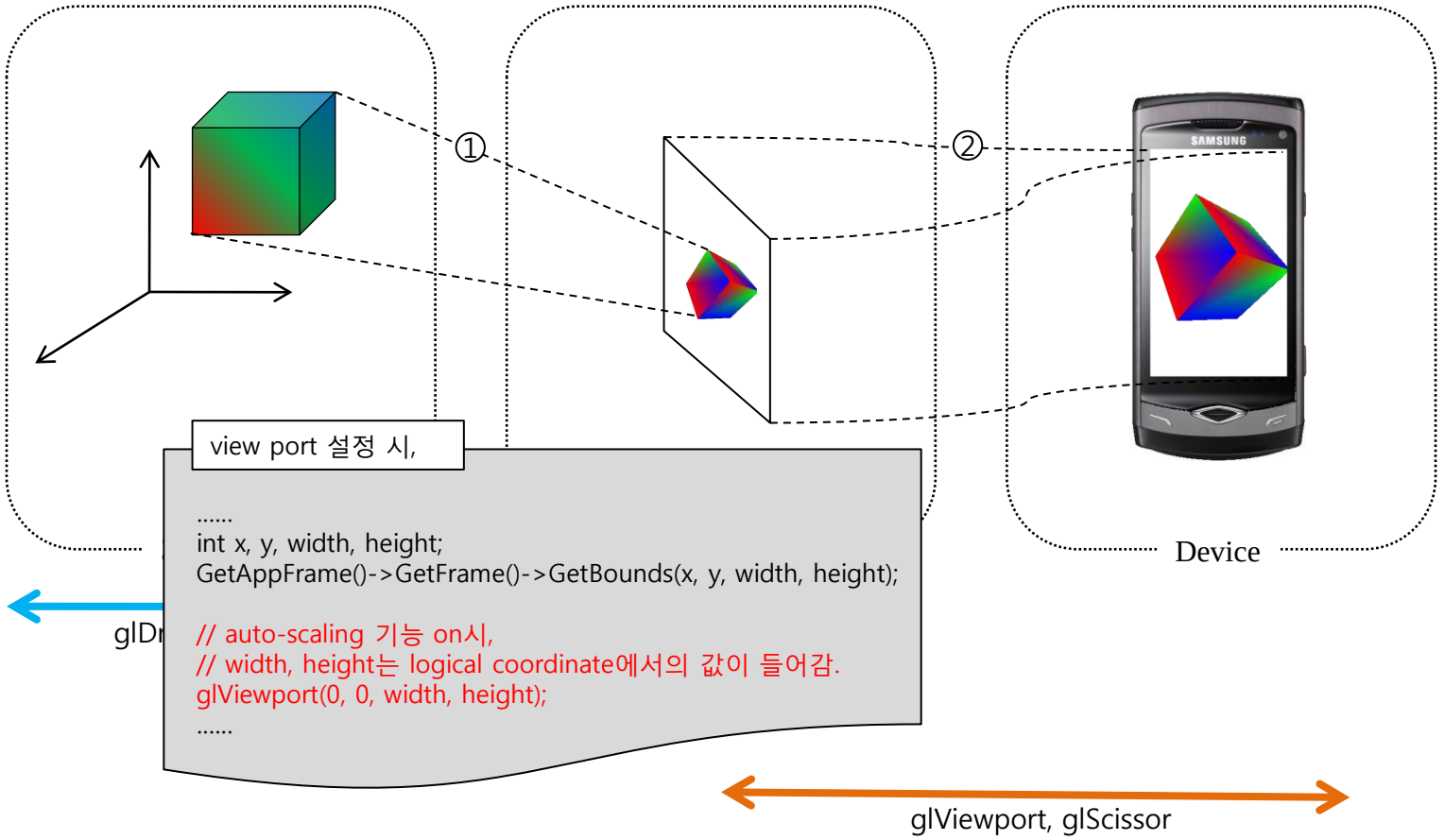
문제점
- Logical coordinate와 physical coordinate의 혼용.



호환성을 위한 고려사항

OpenGL-ES API 사용시 Auto-scaling 기능 사용을 자제할 것.

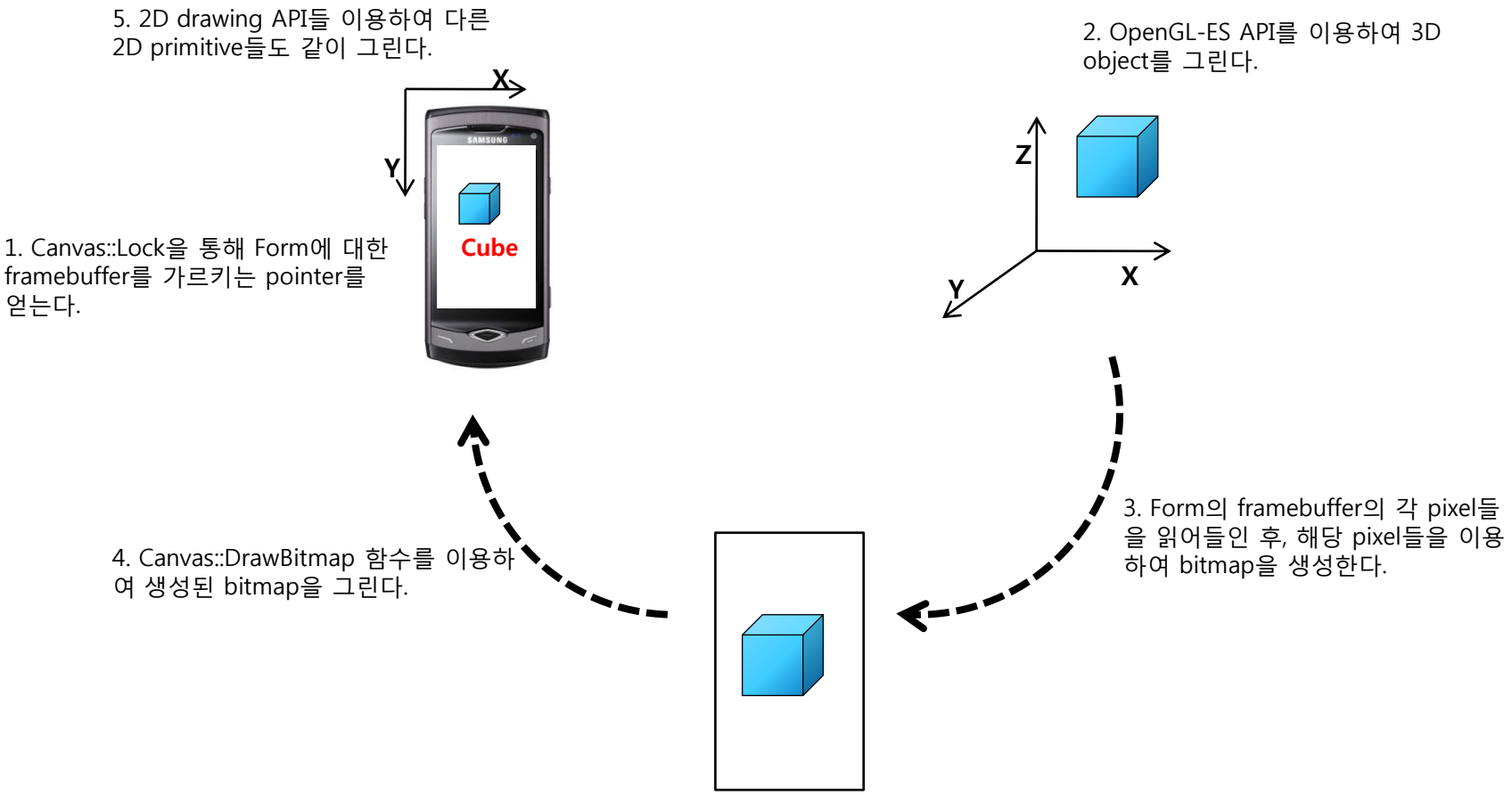
문제점
- Logical coordinate와 physical coordinate의 혼용.



호환성을 위한 고려사항

2D drawing API와 OpenGL-ES API를 혼용해서 사용하지 말 것.

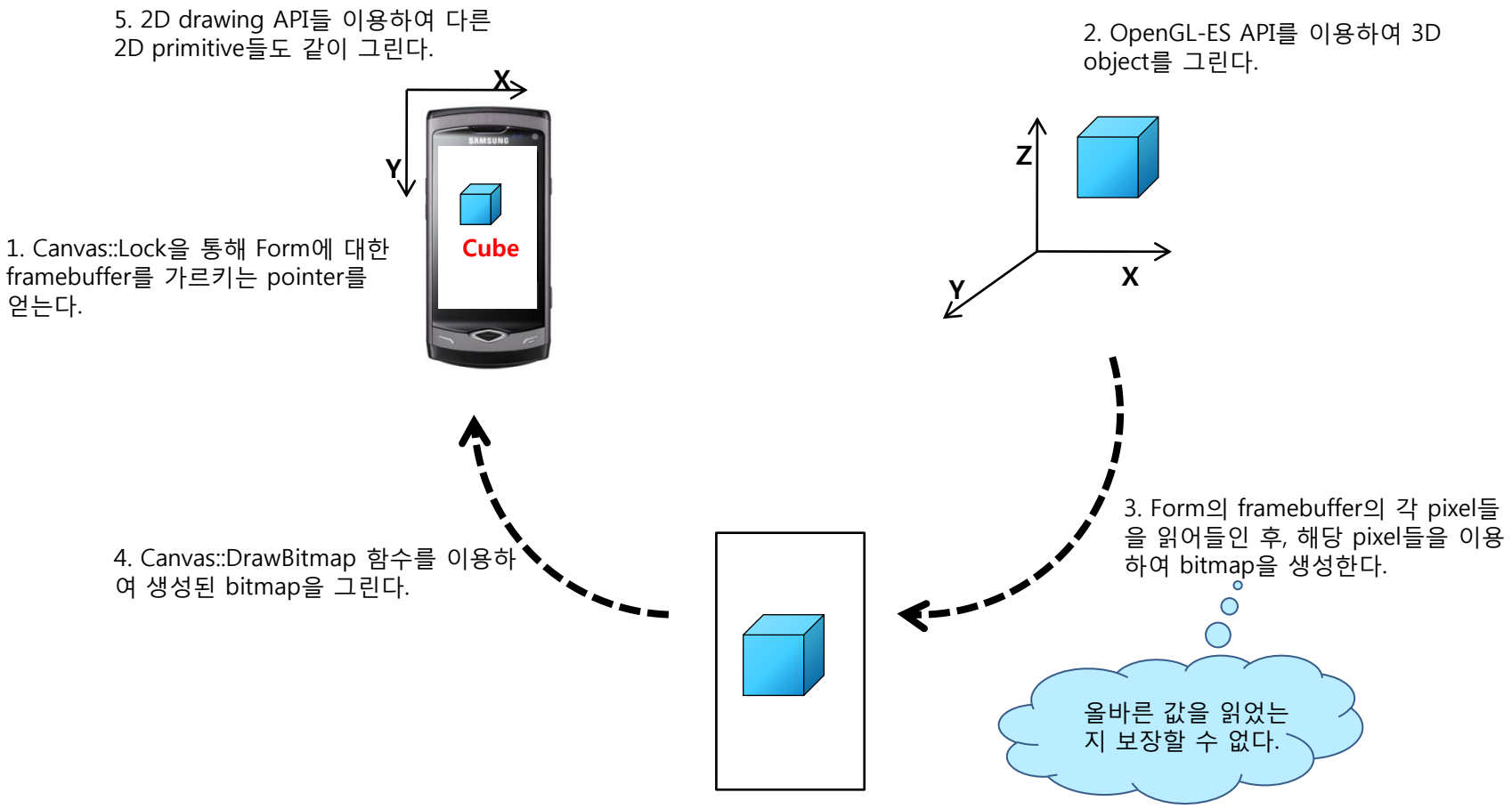
- 2D screen coordinate와 3D space coordinate를 혼용해서 사용하지 말 것.
- 2D drawing API를 사용하기 위해 auto-scaling 기능을 활성화 시키지 말 것.
- Framebuffer에 접근하는 API인 Canvas::Lock이나 Bitmap::Lock 함수를 함께 혼용하지 않도록 주의한다.



호환성을 위한 고려사항

2D drawing API와 OpenGL-ES API를 혼용해서 사용하지 말 것.

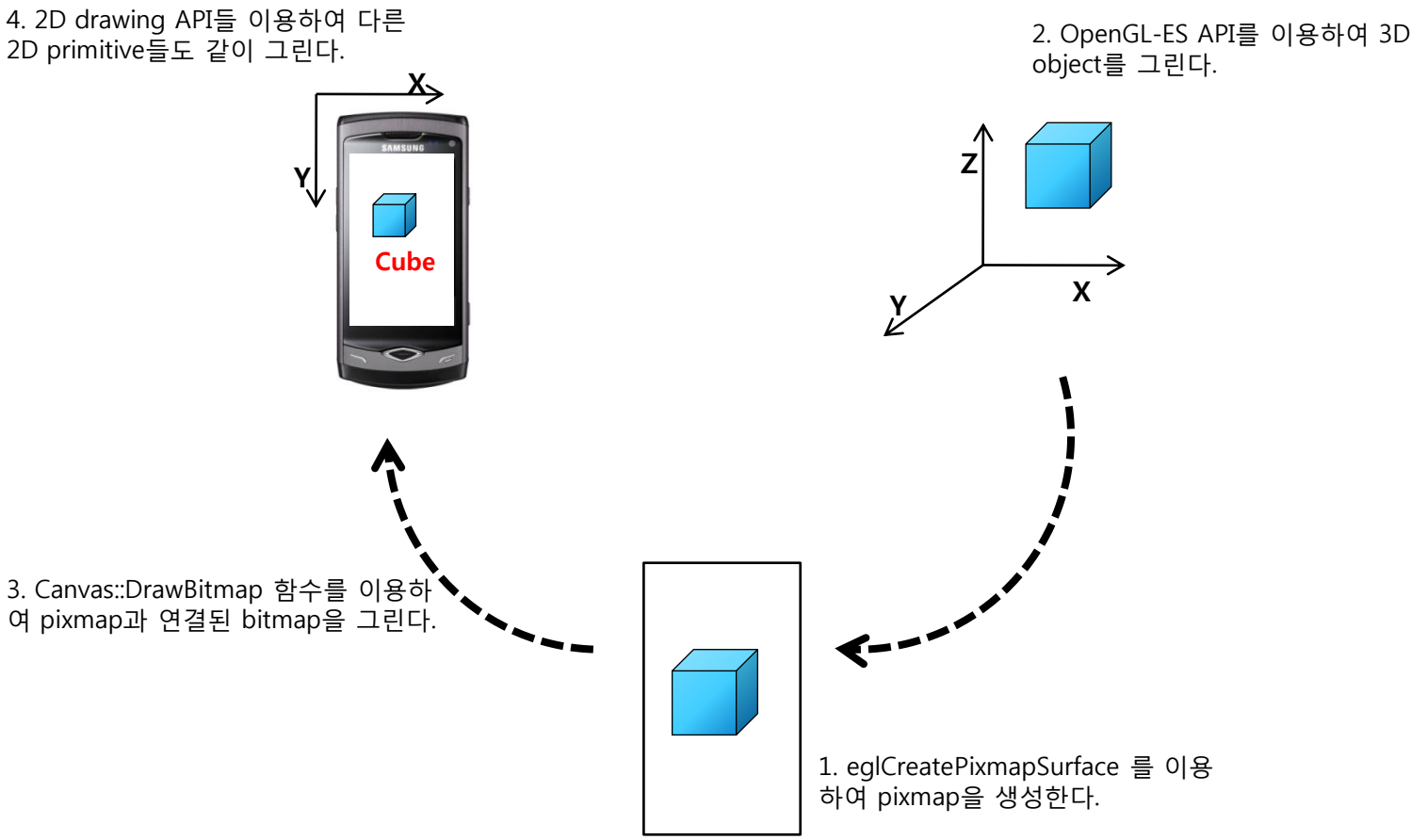
- 2D screen coordinate와 3D space coordinate를 혼용해서 사용하지 말 것.
- 2D drawing API를 사용하기 위해 auto-scaling 기능을 활성화 시키지 말 것.
- Framebuffer에 접근하는 API인 Canvas::Lock이나 Bitmap::Lock 함수를 함께 혼용하지 않도록 주의한다.



호환성을 위한 고려사항

2D drawing API와 OpenGL-ES API를 혼용해서 사용하지 말 것.

해결책
- 2D drawing 과 OpenGL-ES를 완전히 분리하여 사용한다.
- OpenGL-ES는 off-line rendering 에만 사용. (ex, glReadBuffers)
- 2D drawing시 off-line rendering의 결과물 bitmap을 이용한다.



호환성을 위한 고려사항

Application의 성능 최적화.

Application의 성능을 최적화 하여, 다른 device에서 구동 되더라도 성능상 차이로 인해 사용자 경험이 바뀌지 않도록 한다.
- 특정 Device를 위한 최적화 작업을 지양할 것.
- 특정 mobile device에 대해서 최적화를 시도하지 말고, 모든 device에서 동일하게 적용되도록 최적화를 시도할 것.
- 출시 목표 device중 성능이 가장 낮은 것을 목표로 개발할 것.

Application에 심각한 문제가 있더라도 높은 하드웨어 사양으로 인해 감춰질 수 있으며, 이 경우 해당 문제는 낮은 사양의 device에서 발생하게 된다.

호환성을 위한 고려사항

Application의 성능 최적화.

Application의 성능을 최적화 하여, 다른 device에서 구동 되더라도 성능상 차이로 인해 사용자 경험이 바뀌지 않도록 한다.
- 특정 Device를 위한 최적화 작업을 지양할 것.
- 특정 mobile device에 대해서 최적화를 시도하지 말고, 모든 device에서 동일하게 적용되도록 최적화를 시도할 것.
- 출시 목표 device중 성능이 가장 낮은 것을 목표로 개발할 것.
  Application에 심각한 문제가 있더라도 높은 하드웨어 사양으로 인해 감춰질 수 있으며, 이 경우 해당 문제는 낮은 사양의 device에서 발생하게 된다.

State 동기화를 야기하는 API를 최소화 할 것.

State 동기화를 야기하는 API들
: glFinish, glFlush, eglSwapBuffers, glReadPixels, glCopyTexImage2D, glCopyTexSubImage2D 등.

왜 피해야만 하는가?
: state 동기화가 발생하면, 해당 시점에서 gl command queue 쌓여 있는 모든 gl command들을 강제로 모두 처리하도록 한다. 이와 같이 gl command들을 강제로 실행하는 과정에서 다음과 같은 문제가 야기된다.

- 한꺼번에 많은 gl command가 실행되는 과정에서 화면이 멈추거나 frame skipping이 발생할 수 있다. Low end device의 경우 이러한 멈춤 현상이 빈번하거나 심각하게 발생할 수 있다.
- GPU가 내부적으로 수행하고 있던 최적화 작업들을 수행할 기회를 박탈하여 전반적인 실행속도를 저하시킨다. Low end device의 경우 이러한 실행속도 저하가 더 크게 나타난다.
- GPU제조사마다 driver쪽에서 GL 함수를 구현하는 방법이나 함수의 behavior에 조금씩 차이가 나며, 특정 GPU에서는 속도 향상을 위해 state를 동기화 시키는 함수들의 작동 방식을 일반적인 경우와 차이를 두는 경우가 있다. 이와 같은 구현 방식의 차이는 application의 이식성에 좋지 못한 영향을 끼친다.

호환성을 위한 고려사항

Application의 성능 최적화.

성능 향상을 위한 고려사항

1. **glDrawElement와 glDrawArray**는 상당량의 memory 연산을 일으키기 때문에 되도록이면 호출의 횟수를 최대한 줄여 memory 연산 횟수를 줄이도록 한다.

2. **glDrawElement와 glDrawArray** 호출 시 상당량의 memory 연산을 일으키기 때문에 sound play나 g-sensor 정보 획득 등 대량의 정보가 오고 가는 작업을 진행 중인 상태에서 해당 함수가 호출되면 bus speed와 bandwidth 한계로 인한 병목현상이 발생하기 쉽다. 이러한 병목현상을 최소화 하기 위해 되도록이면 glDrawElement, glDrawArray 함수의 호출 시점을 대량의 정보가 오고 가는 작업과는 떨어뜨려 두도록 한다.

3. **eglSwapBuffers 호출 회수를 최대한 줄인다.**
   : eglSwapBuffers함수는 state를 동기화 시키는 API로, 호출 직후 GPU-CPU가 동시에 작업을 진행하고 gl command들을 처리하는 과정에서 새로운 gl call이 발생하면 순간적으로 stall이 발생하게 된다. 따라서 gl commands가 수행 중인 동안에는 CPU가 작업을 진행하도록 하도록 하여 stall을 막고 작업 효율을 높이도록 해야한다. 이와 같은 시점을 개발자가 알기는 쉽지 않지만, 일반적으로 모든 gl call을 호출한 후 마지막으로 eglSwapBuffers를 호출하는 것이 가장 좋다고 알려져 있다.

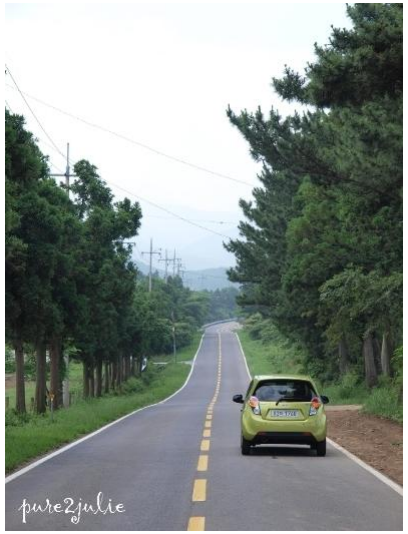
4. **fragment shader program의 최적화에 최대한 투자한다.**
   : 일반적으로 shader에 의해서 처리되는 vertex 개수보다 pixel 개수가 훨씬 많다. 특히 device간 화면 사이즈가 변하더라도 vertex개수에는 큰 변화가 없지만, pixel의 수는 크게 변화하기 때문에 device간 application 성능에 큰 영향을 끼치게 된다. 따라서 성능 측면에서의 device간 호환성/이식성을 위해 fragment shader의 최적화에 최대한 신경을 쓰고, vertex shader와 pixel shader 양쪽으로 구현 가능한 알고리즘의 경우에는 vertex shader가 처리하도록 한다.

5. **각 screen size별로 적정한 texture size를 결정하여 적정한 texture를 이용할 수 있도록 한다.**
   : 이론적으로 가장 좋은 texture size는 rendering시 texture의 one texel이 display device의 one pixel에 대응되는 경우이다. 각 screen size별로 적정한 texture를 load 할 수 있도록 모든 resource 를 제공하거나 multiple APK support와 같이 동일 application에 대해서 복수의 application package를 준비하도록 한다.

호환성을 위한 고려사항

외부 Game engine을 개발하는 경우.

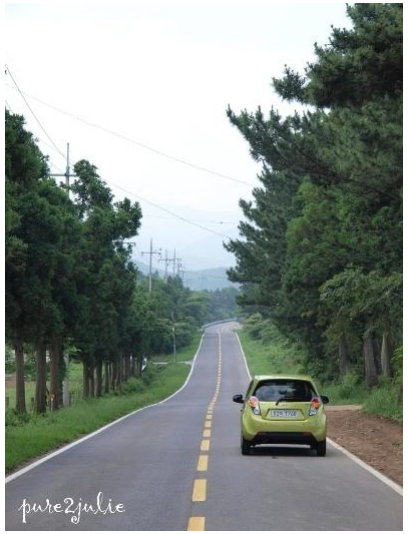
Game engine의 API의 내부 동작 방식을 파악해서 사용할 것.
- 러시안 페인트공 문제



호환성을 위한 고려사항

외부 Game engine을 개발하는 경우.

Game engine의 API의 내부 동작 방식을 파악해서 사용할 것.
- 러시안 페인트공 문제

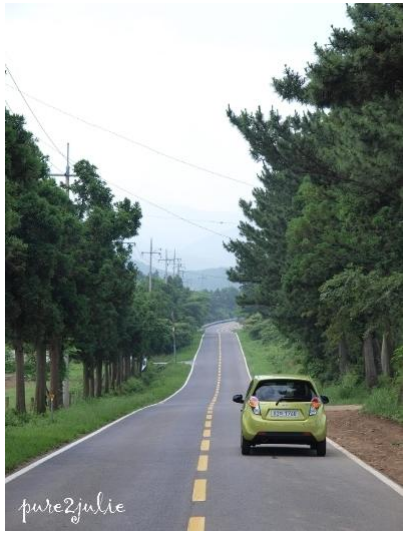


첫째날 둘째날 셋째날 넷째날

호환성을 위한 고려사항

외부 Game engine을 개발하는 경우.

Game engine의 API의 내부 동작 방식을 파악해서 사용할 것.
- 러시안 페인트공 문제

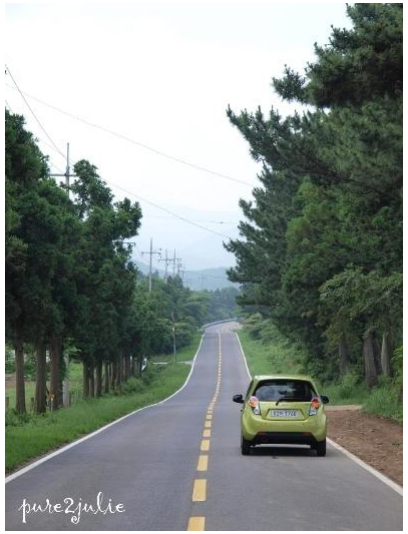


첫째날 둘째날 셋째날 넷째날

호환성을 위한 고려사항

외부 Game engine을 개발하는 경우.

Game engine의 API의 내부 동작 방식을 파악해서 사용할 것.
- 러시안 페인트공 문제
  + strcat() 함수 구현문제
  + 유식한 방법은? → string의 length를 저장하고 가지고 있는다.
  + strcat 내부 구조를 잘 알지 못한 상태에서 사용하는 경우에는 비효율 적으로 사용할 수 있다.
- Game engine API내 내부 동작 방식을 파악하지 못하고 사용하면,
  1. 예기치 못한 성능 저하
  2. 예기치 못한 호환성 문제
     ex) Texture 선택 툴.



첫째날 둘째날 셋째날 넷째날

요약

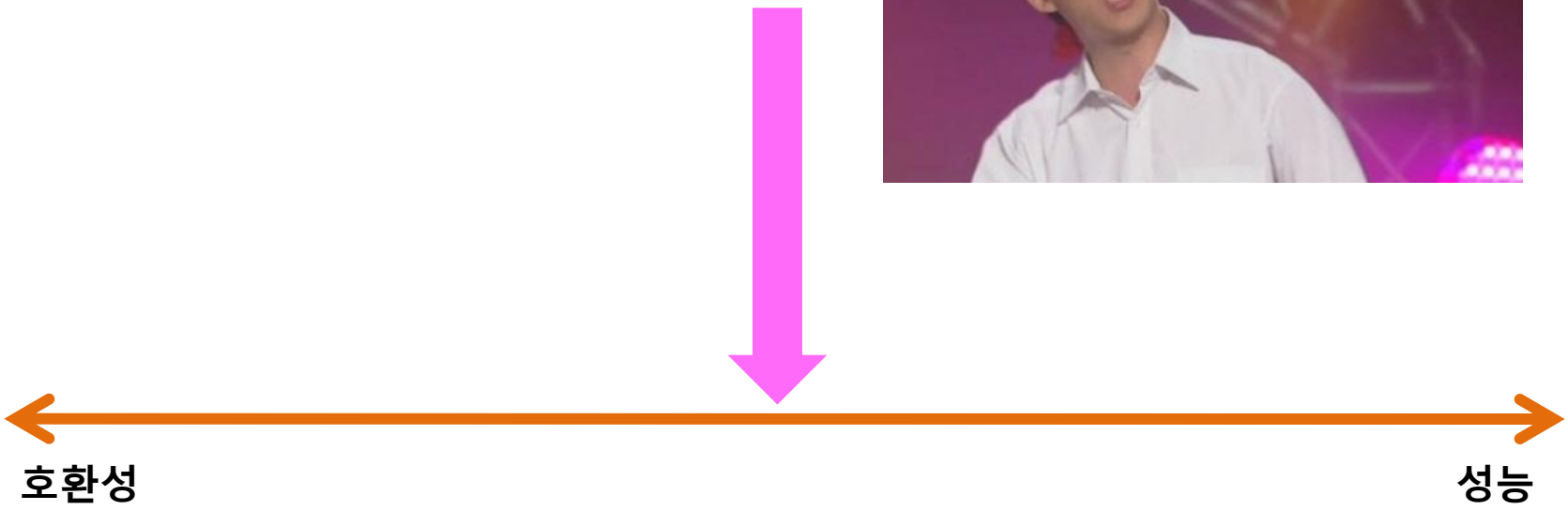
OpenGL-ES를 이용하여 개발하는 과정에서 호환성/이식성 높여 주세요.

- OpenGL-ES extension 사용시 주의를 기울여 주세요.
- 특정 Graphics hardware 에서 제공하는 기능은 되도록이면 피해 주세요.
- 확장 기능 사용시 지원 여부를 확인하고, 지원되지 않는 경우에는 우회방법을 제공해 주세요.
- Shader programming시 호환성/이식성을 신경 써 주세요.
- OpenGL-ES API 사용시 auto-scaling 기능 사용을 자제해 주세요.
- 2D drawing API와 OpenGL-ES API를 혼용하지 마세요.
- Application 성능 최적화에 신경 써 주세요.
- 외부 Game engine 사용 시 조심해서 사용해 주세요.

요약

OpenGL-ES를 이용하여 개발하는 과정에서 호환성/이식성 높여 주세요.

- OpenGL-ES extension 사용시 주의를 기울여 주세요.
- 특정 Graphics hardware 에서 제공하는 기능은 되도록이면 피해 주세요.
- 확장 기능 사용시 지원 여부를 확인하고, 지원되지 않는 경우에는 우회방법을 제공해 주세요.
- Shader programming시 호환성/이식성을 신경 써 주세요.
- OpenGL-ES API 사용시 auto-scaling 기능 사용을 자제해 주세요.
- 2D drawing API와 OpenGL-ES API를 혼용하지 마세요.
- Application 성능 최적화에 신경 써 주세요.
- 외부 Game engine 사용 시 조심해서 사용해 주세요.




KBS2 HD

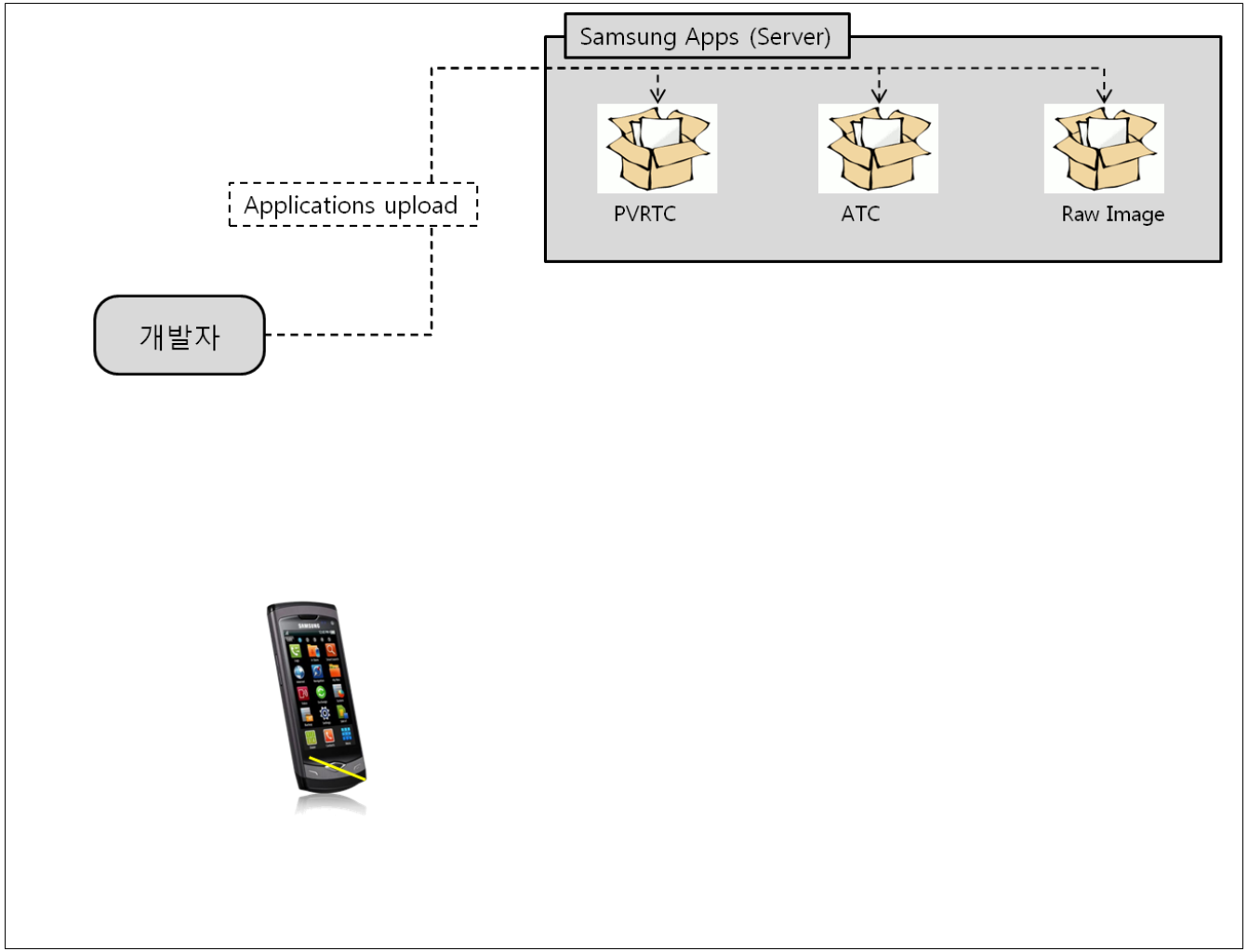
Question?

Backup

Application Filtering by Market

Compressed Texture Format

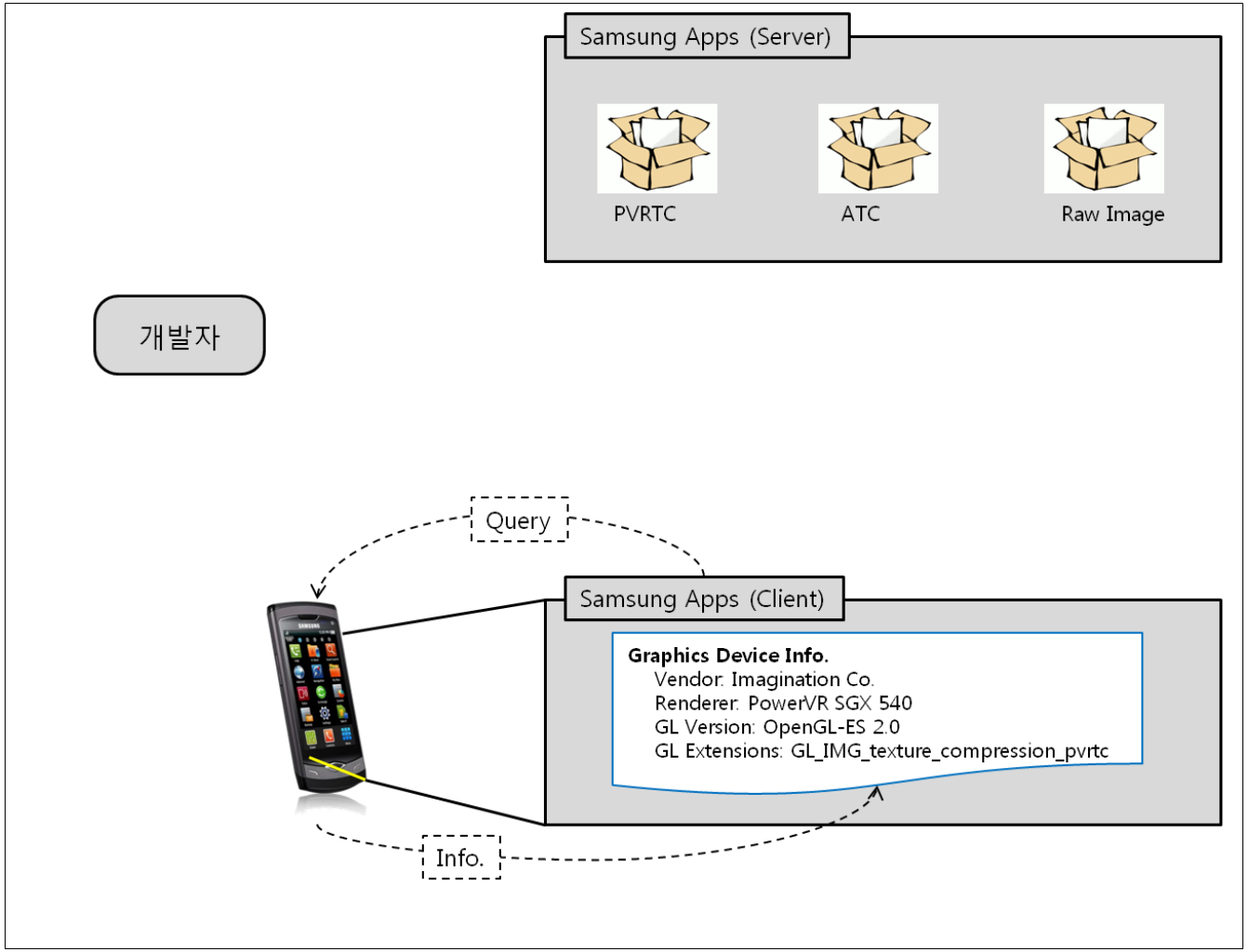
Application package에서 사용하고 있는 compressed texture format에 따라 application을 filtering하여 사용자에게 제공.
- cf) 안드로이드의 Multiple APK support 와 유사



Application Filtering by Market

Compressed Texture Format

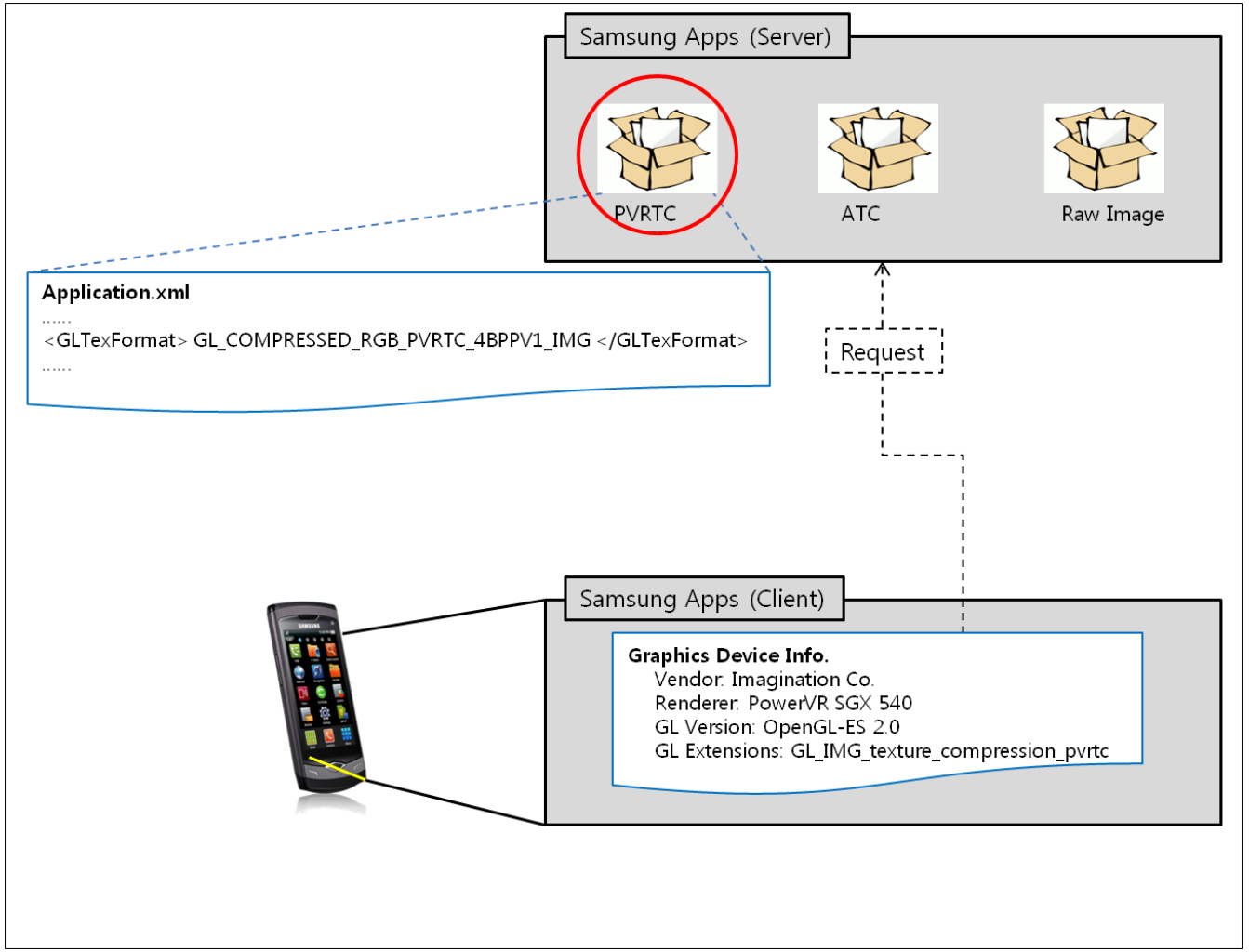
Application package에서 사용하고 있는 compressed texture format에 따라 application을 filtering하여 사용자에게 제공.
• cf) 안드로이드의 Multiple APK support 와 유사



Application Filtering by Market

Compressed Texture Format

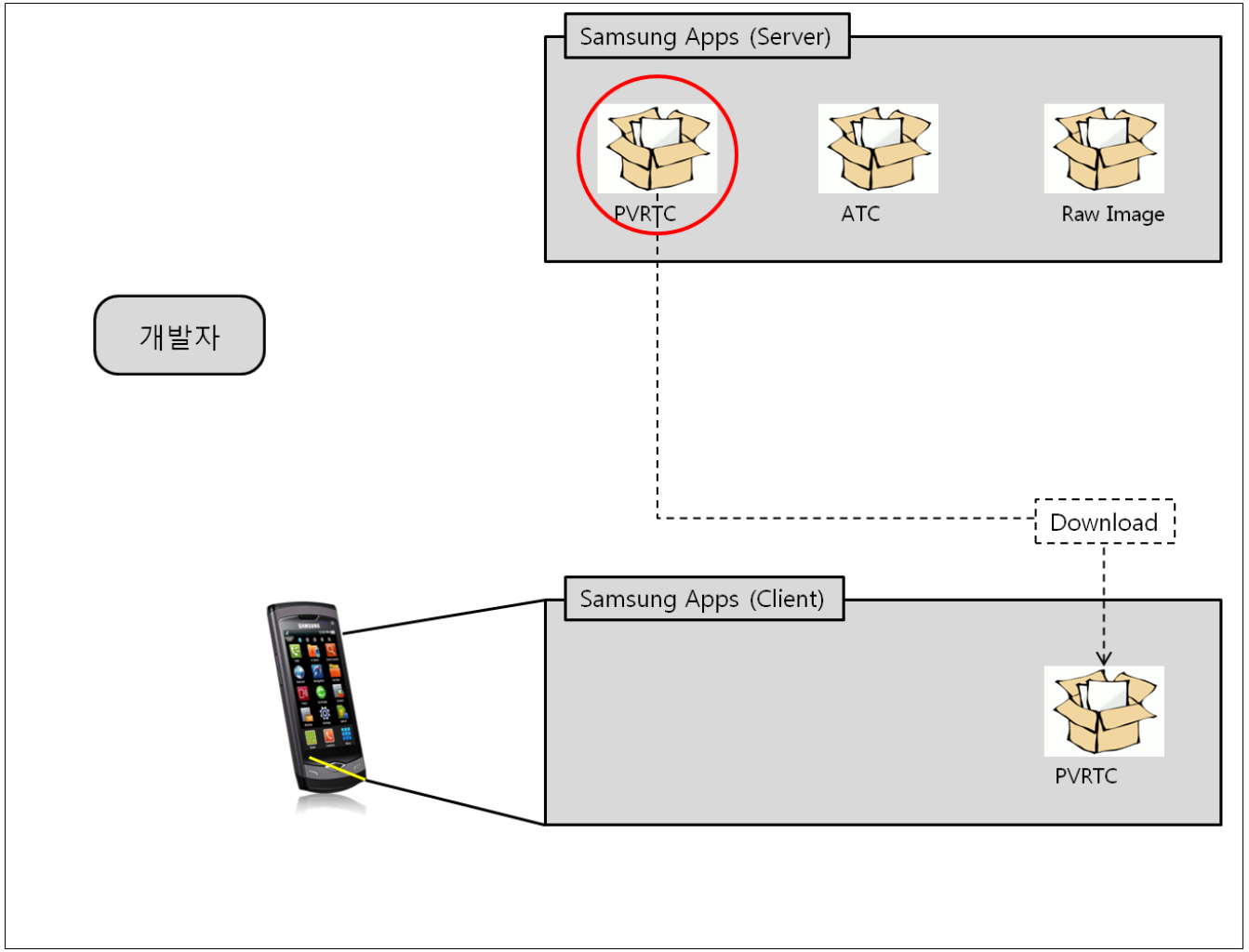
Application package에서 사용하고 있는 compressed texture format에 따라 application을 filtering하여 사용자에게 제공.
- cf) 안드로이드의 Multiple APK support 와 유사



Application Filtering by Market

Compressed Texture Format

Application package에서 사용하고 있는 compressed texture format에 따라 application을 filtering하여 사용자에게 제공.
- cf) 안드로이드의 Multiple APK support 와 유사



Wave series

Wave series

High tier

WaveII
SGX540 (Imagination co.)
480x800 (WVGA)

WaveIII
Adreno205 (Qualcomm)
480x800 (WVGA)

Mid tier

WaveM
BCM21553 (Broadcom)
320x480 (HVGA)

Low tier

?

Multiple Screen Support of bada

Auto-scaling & Layout

Auto-scaling
: 화면 사이즈에 맞게 resource의 크기나 UI control 들을 자동으로 조정해 주는 기능.

Layout
: Mobile device 화면 사이즈나 화면 비율에 가장 어울리도록 UI control들을 정렬하여 사용자에게 동일한 UI를 제공.