GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

CLASSIFIED TEXTURE RESIZING FOR MOBILE DEVICES

Jae-Ho Nah Byeongjun Choi Yeongkyu Lim

SW Platform Lab, CTO Division, LG Electronics

Photography & Recording Encouraged

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

INTRO & RELATED WORK

EVOLUTION OF MOBILE GAME GRAPHICS

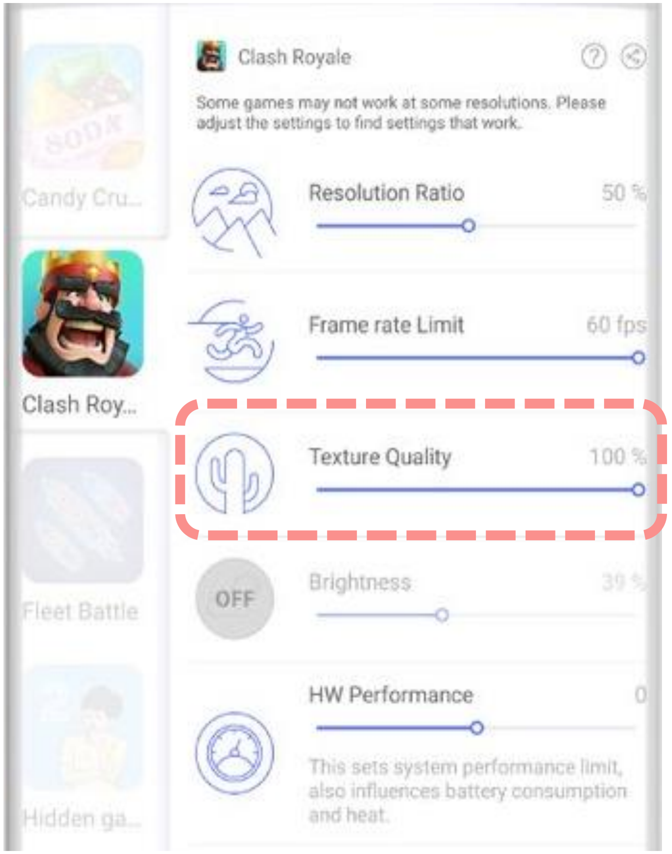
- For the last 10 years, graphics on mobile games have rapidly evolved
  - Nowadays mobile AAA games exploit lots of high-resolution textures for spectacular visuals

Angry Birds (2009)

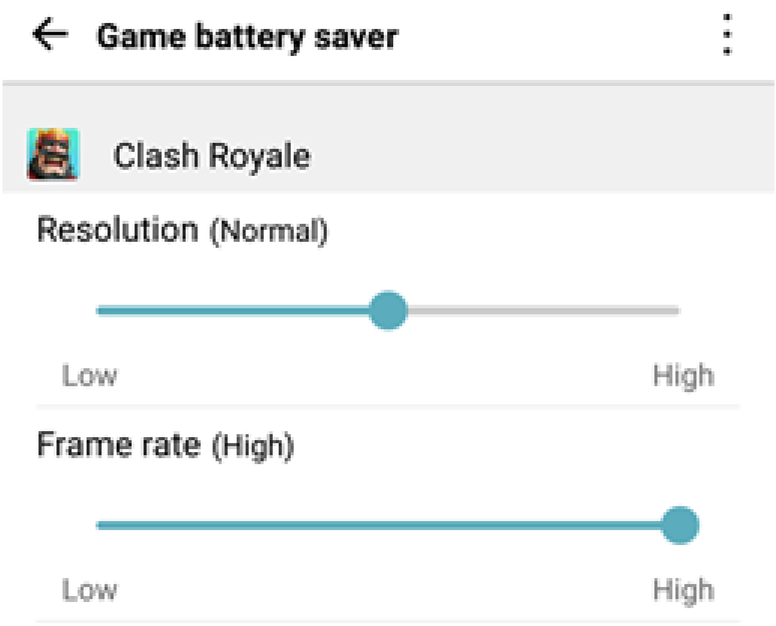
Black Desert (2018)

- Use of many high-quality textures increases the required GPU power and memory bandwidth
  - Can cause low frame rates, overheating, or fast battery drain

S/W APPROACHES TO CONTROL GRAPHICS QUALITY



Samsung Game Tuner



LG Game Battery Saver

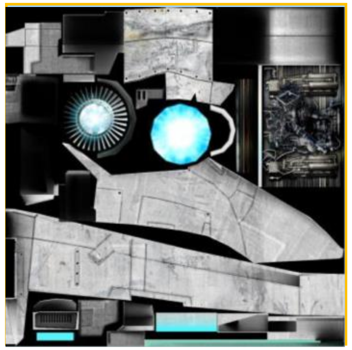
- Trade-off between quality, performance & battery life
- Game tuner additionally supports texture quality control

HOW TO CONTROL TEXTURE QUALITY BY GAME TUNER

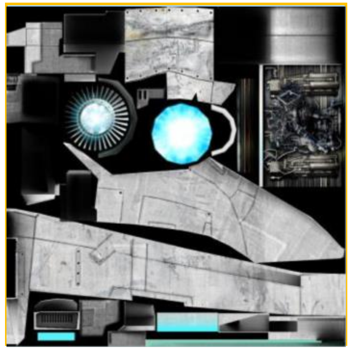
- Procedure (according to our analysis on v2.3)
  1) Renders each texture to an off-screen frame buffer
  2) Reads the rendered results by glReadPixels()
  3) Resizes the texture on a CPU
     e.g., a setting value of 25%: width×0.5, height×0.5
  4) Uploads the resized texture to the GPU memory again

GPU Memory

Texture Memory

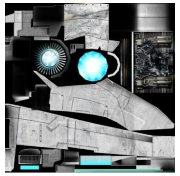


Off-screen Frame Buffer



glReadPixels()

CPU Memory



HOW TO CONTROL TEXTURE QUALITY BY GAME TUNER

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018

- Procedure (according to our analysis on v2.3)
    1) Renders each texture to an off-screen frame buffer
    2) Reads the rendered results by glReadPixels()
    3) Resizes the texture on a CPU
       e.g., a setting value of 25%: width×0.5, height×0.5
    4) Uploads the resized texture to the GPU memory again

- Feature
    - Increases frame rates or decreases power consumption by resizing “uncompressed” textures

- Limitations
    - **Quality degradation:** Resizing uncompressed textures can bring out visible blur effects
    - **Limited application range:** Compressed or dynamic textures are not resized
    - **Loading time increase:** GPU → CPU → GPU processing (~2 seconds)

OUR APPROACH

CLASSIFIED TEXTURE RESIZING (CTR)

- Goal
  - Improves power efficiency or frame rates on texture-heavy games
  - Better implementation than existing solutions

- Our considerations
  - How to minimize quality degradation after resizing textures
  - How to support various types of textures
  - How to minimize loading-time increase

- Main idea
  - After classifying textures, apply a different approach to each type at the OpenGL ES driver level

OUR TEXTURE CLASSIFICATION

Texture

Static

Textures stored in an app

Uncompressed

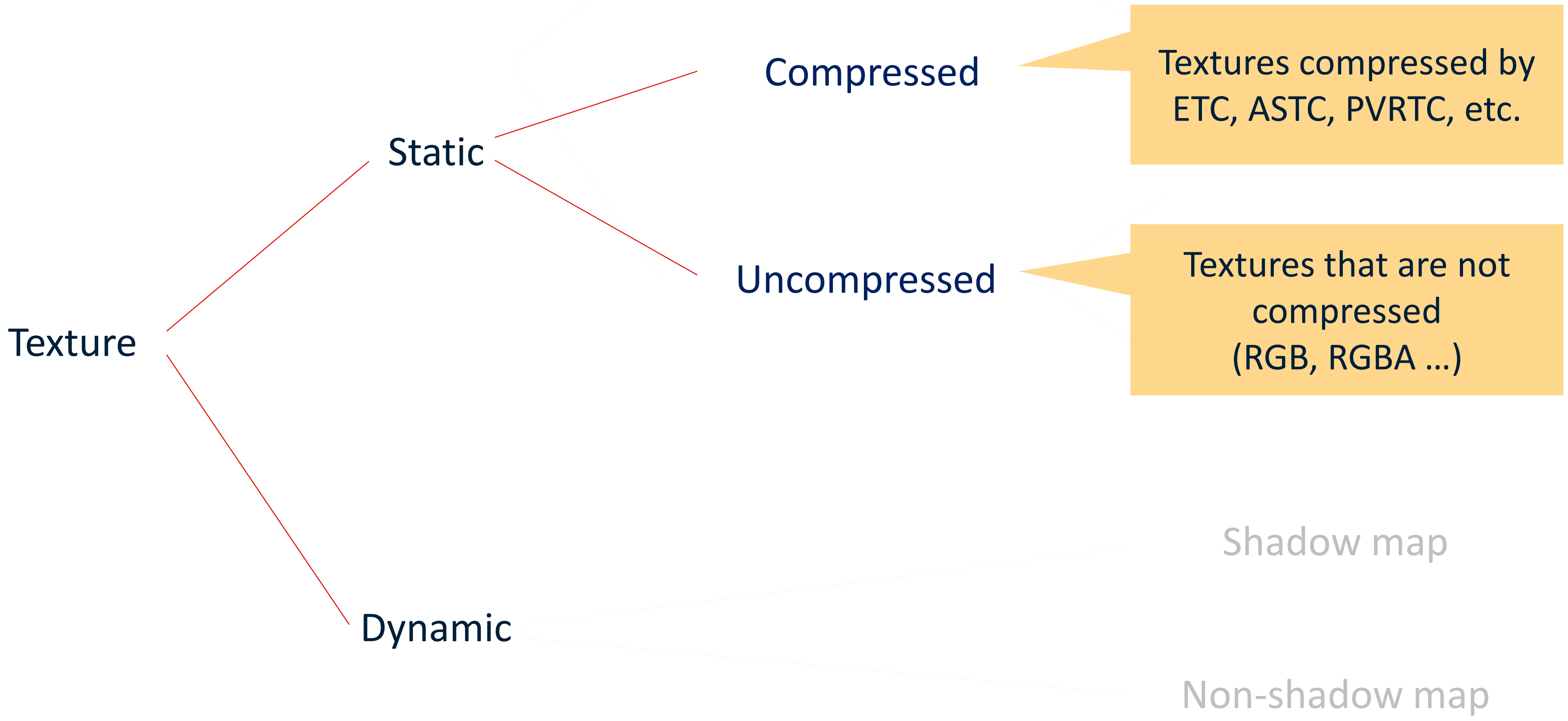
Mipmapped

Non-mipmapped

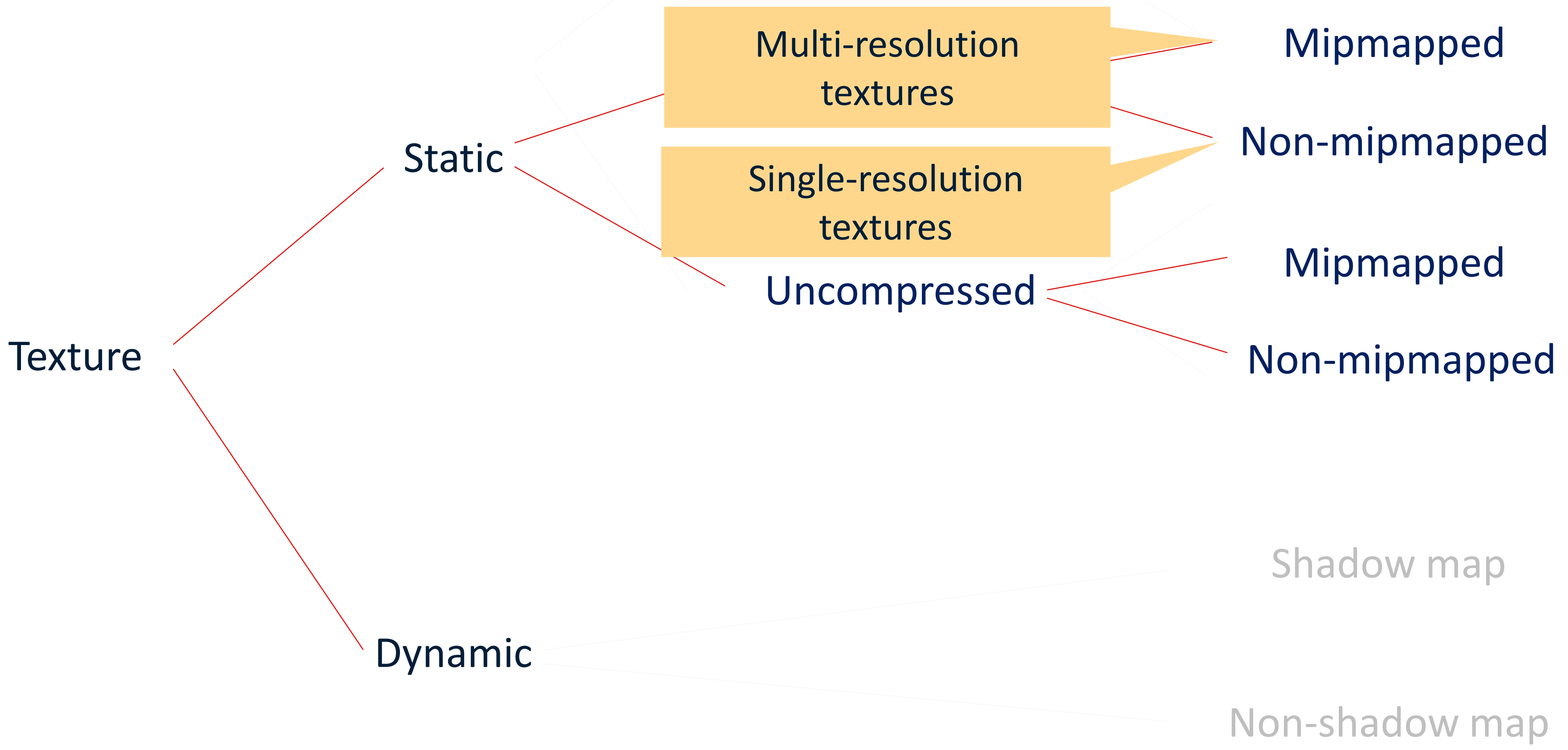
Dynamic

Dynamically generated textures during each frame with render-to-texture techniques

OUR TEXTURE CLASSIFICATION



OUR TEXTURE CLASSIFICATION



OUR TEXTURE CLASSIFICATION

- Texture
  - Static
    - Compressed
      - Mipmapped
      - Non-mipmapped
    - Uncompressed
      - Mipmapped
      - Non-mipmapped
  - Dynamic
    - Shadow map
    - Non-shadow map

Shadow maps with the aspect ratio of 1:1

Other maps that do not have the aspect ratio of 1:1

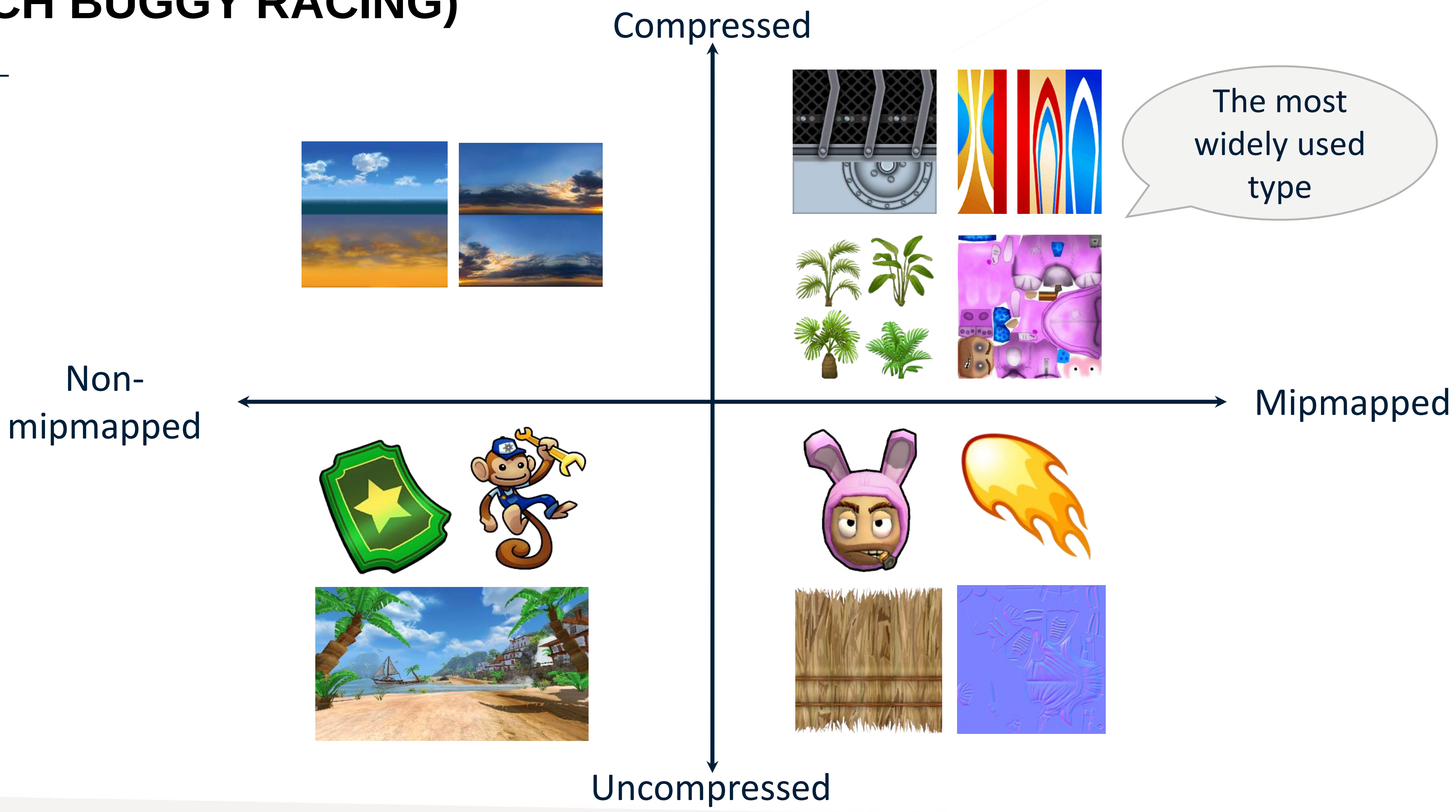
OUR TEXTURE CLASSIFICATION

The classification is possible by analyzing OpenGL ES commands;
we modified the command dispatcher in the GPU driver

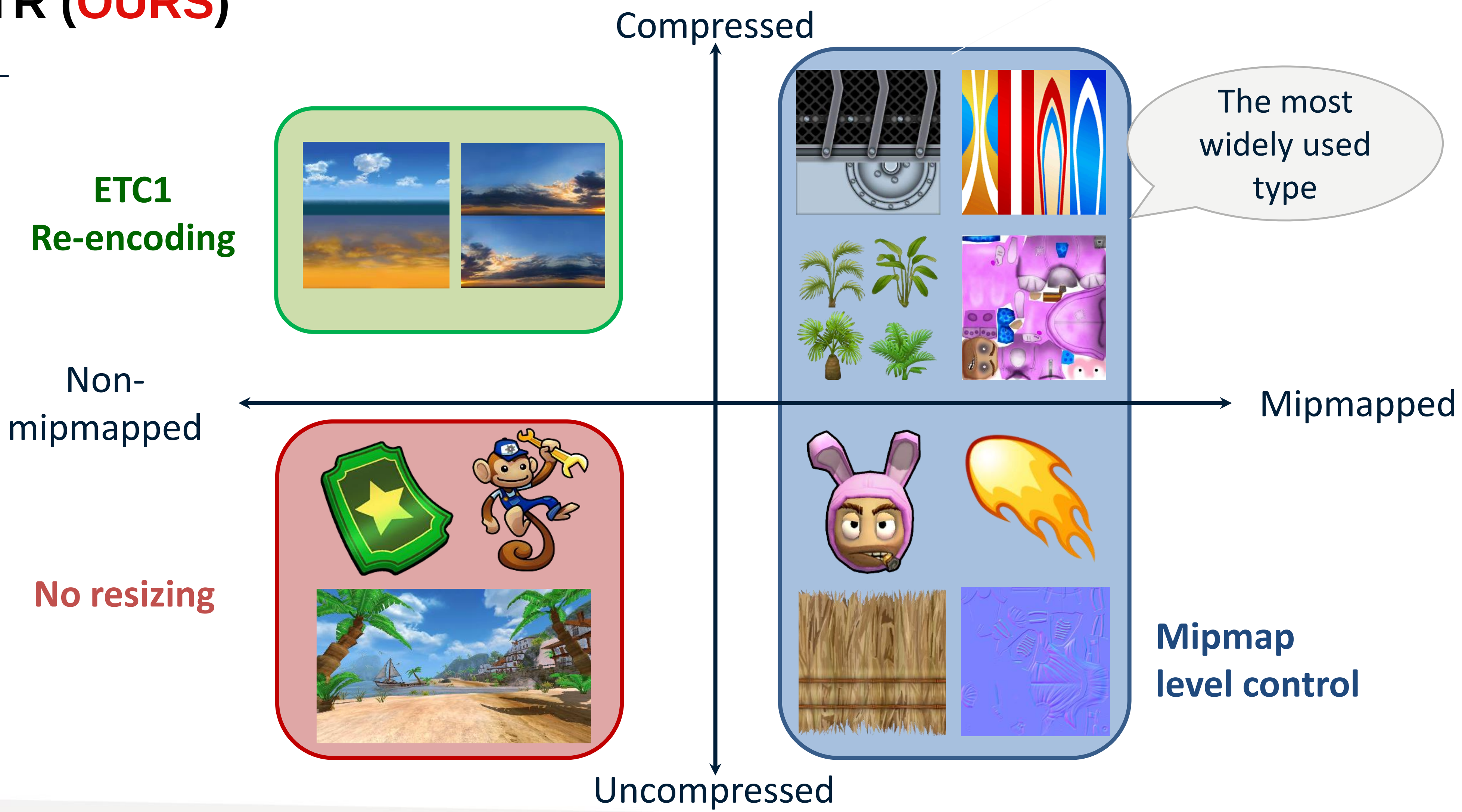
```
…
glGenTextures(1, &texture_id);
glBindTexture(GL_TEXTURE_2D, texture_id);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, texture_width, texture_height, 0, GL_RGBA, GL_UNSIGNED_BYTE, texture_data);
…
```

Non-shadow map

STATIC TEXTURE CLASSIFICATION (BEACH BUGGY RACING)



STATIC TEXTURE RESIZING BY CTR (OURS)

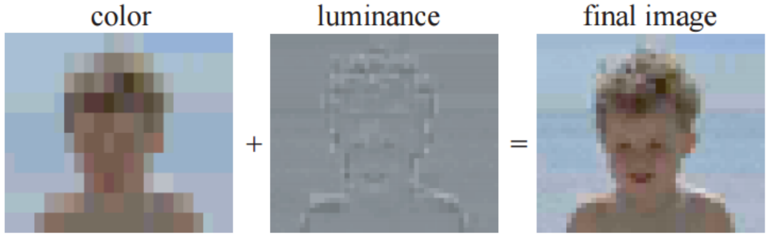


STATIC TEXTURE RESIZING (1): ETC1 RE-ENCODING

- Non-mipmapped, compressed textures
  - Decoding → Resizing (¼) → Re-encoding
  - Needs to minimize the encoding time rather than the decoding time
  - Only handles ETC1/2 formats, which are the standard formats in the OpenGL ES spec

- Used ETC compression library
  - Etcpak 0.5 (https://bitbucket.org/wolfpld/etcpak/wiki/Home)
  - The fastest library (with slightly lower quality than others)

- Our implementation
  - Ported the etcpak code to the Mali driver
  - Single-threaded & no SIMD-optimizations
  - Additional ETC2 modes (T, H & Planar) are not applied to resized textures to reduce the encoding time; ETC2 RGB is re-encoded to ETC1

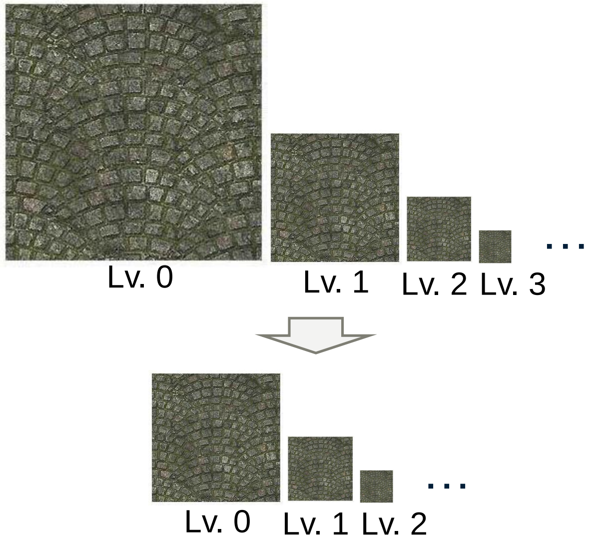


The core idea of ETC1*

* Jacob Ström and Tomas Akenine-Möller, "*i*PACKMAN: High-quality, low-complexity texture compression for mobile phones," *Graphics Hardware 2005*.

STATIC TEXTURE RESIZING (2): MIPMAP LEVEL CONTROL

- Mipmapped textures
  - If a texture is judged as a mipmap (level≥1),
    the level-0 image is thrown out and
    the other images are leveled down by one (¼ of the max size)



- Advantages
  - Very simple & low overhead
  - Can be applied to both compressed and uncompressed textures

- Current limitation
  - Hard to detect whether a texture is mipmapped or not before loading its level-0 image*
  - Throwing out the level-0 image of a mipmap after re-encoding → a loading-time increase

* The OpenGL spec allows glTexParameteri() both before and after glTexImage2D()

STATIC TEXTURE RESIZING (3): NO RESIZING

- Non-mipmapped, uncompressed textures
  - Usually one-to-one mapped to a screen (e.g., 2D menus, icons, etc.); not compressed on purpose by graphics developers
  - Compression or resizing of this type of textures can decrease visibility
  - Not resized by CTR

Two examples of uncompressed, non-mipmapped textures in Beach buggy racing

DYNAMIC TEXTURE CLASSIFICATION (BEACH BUGGY RACING)

Main framebuffer*, G-buffer, reflection map, velocity map, etc.

Aspect ratio ≠ 1:1

Shadow map

Aspect ratio = 1:1

* A framebuffer object (FBO) which has a different target resolution with the display resolution and is blitted to the framebuffer #0 at the end of the frame

DYNAMIC TEXTURE RESIZING BY CTR (OURS)

No Resizing

Main framebuffer, G-buffer, reflection map, velocity map, etc.

Aspect ratio ≠ 1:1

- The main framebuffer and G-buffer
  - Should not be resized to prevent duplicated resolution resizing
- Non-shadow maps are not resized

Resizing

Shadow map

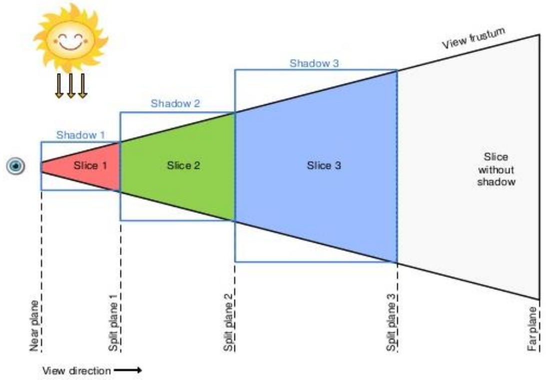
Aspect ratio = 1:1

- Shadow maps are resized

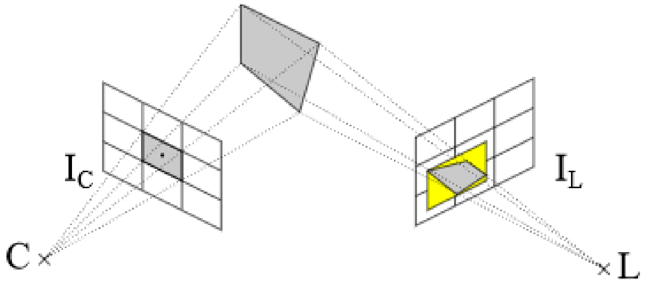
OUR SHADOW MAP RESIZING

- Three conditions for being detected as a shadow map
  - The data pointer in glTexImage2D() is null
  - Width = Height
  - The texture type is a depth image, or depth data is attached using a renderbuffer object (RBO)
- Resizing factor: ¼ of the original size (as other types)
- Cases when the resizing cannot be applied
  - Using glTexStorage2D(); hard to distinguish dynamic from static textures (e.g., Temple Run 2)

  - Using glTexImage3D() for cascaded shadow maps (e.g., Adreno SDK)



  - Width ≠ Height (rare but possible*)



* Michal Ferko, "Resolution estimation for shadow mapping," *EGUK Theory and Practice of Computer Graphics 2013.*

RESULTS

POWER CONSUMPTION MEASUREMENT (GPU+DRAM)

- Experimental environment
  - Our own MADK board with/ ARM 64bit CPU, Mali GPU & LPDDR4
  - NI USB-6363 to measure power consumption

- Results* on the three Android games (at the same FPS)

Beach Buggy Racing
1.4W → 1.2W (▽10.2%)

Implosion
1.9W → 1.8W (▽4.2%)

Xenowerk
1.7W → 1.4W (▽16.3%)

* Measured during the game play; the loading time is excluded.

IMAGE QUALITY COMPARISON - ORIGINAL

IMAGE QUALITY COMPARISON - CTR (OURS)

IMAGE QUALITY COMPARISON - GAME TUNER (GALAXY S8, 25%)

LOADING TIME

Beach Buggy Racing

7.4s → 9.6s (+2.2s)

Implosion

25.5s → 25.8s (+0.3s)

Xenowerk

17.4s → 20.2s (+2.8s)

- Currently 0-level ETC1/ETC2 RGB textures are re-encoded
- BB Racing and Xenowerk show around 2s loading-time increases
  - Comparable results to Game Tuner, but needs to be improved
- Implosion (w/ ETC2 RGBA) shows a slight increase (within the error range)
  - Only the mipmap level control without ETC1 re-encoding is executed for static textures

FUTURE WORK

ASTC

HDR

3D textures

Vulkan

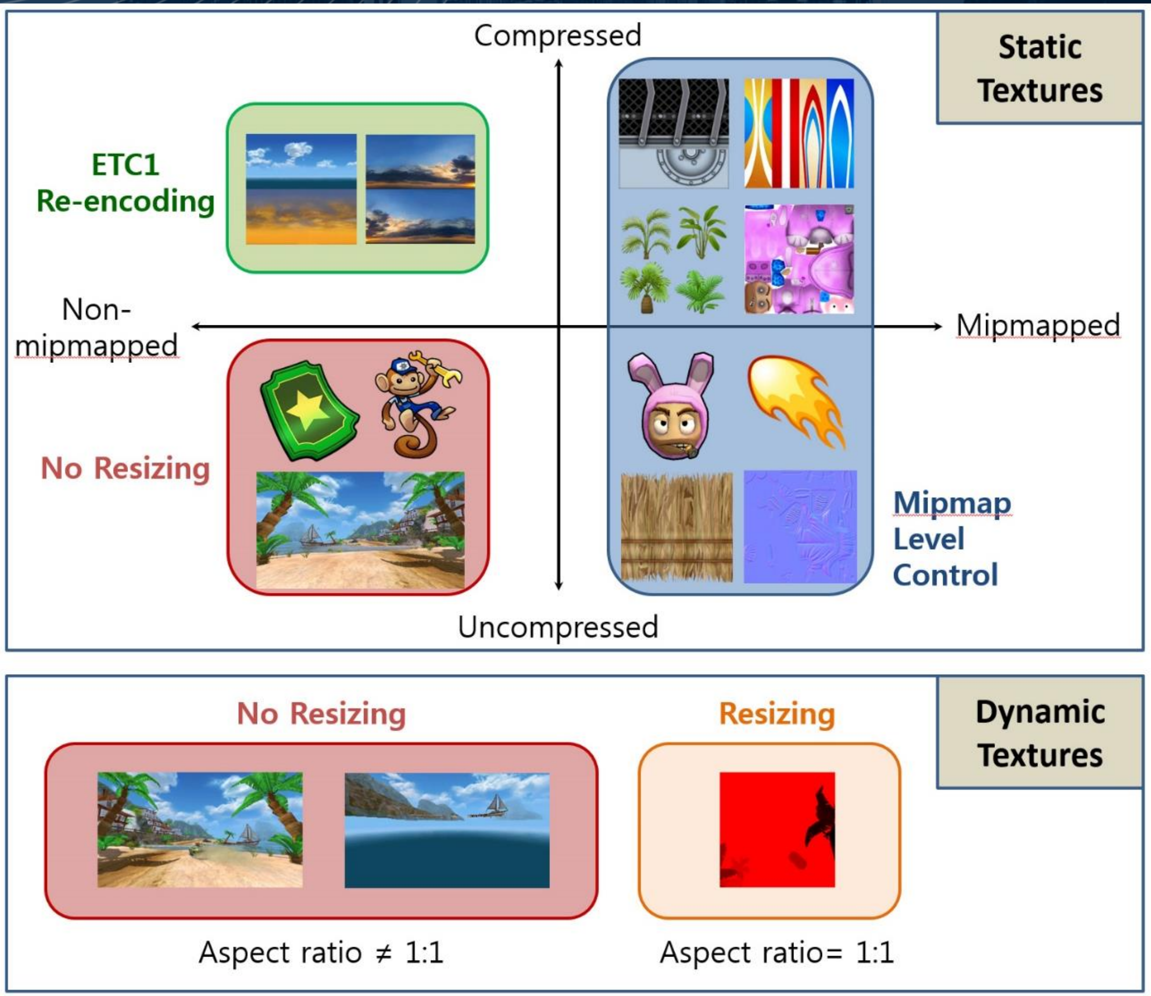
Minimizing loading-time increases

- Clever mipmap detection
- Multi-threaded & SIMD re-encoding

Wider format/API support

- Re-encoding other compression/data types
- Extending to other APIs (e.g., Vulkan)

GENERATIONS / VANCOUVER 12-16 AUGUST
SIGGRAPH2018



THANK YOU

Any suggestion for collaboration with our lab is welcomed

E-mail: nahjaeho@gmail.com