

Vulkan Game Development

Unity x Samsung

Samsung

Inae Kim, SeungHwan Lee, Daemyung Jang

Contents - Introduce Vulkan

1. Introductions
2. Vulkan is
3. Why Vulkan
4. Explicit API
5. Portability
6. Comparison

Introduction

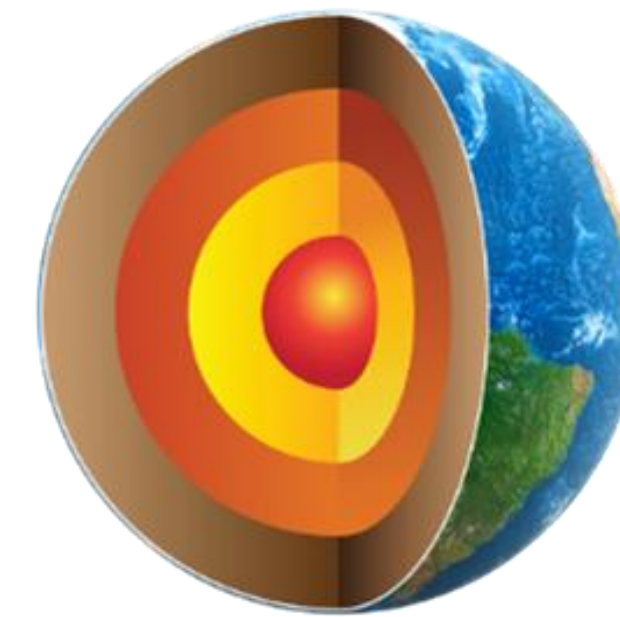
History

- The Force Awakens: October 2012
 - glCommon TSG formed to consider redesign of OpenGL / ES
 - Brainstorming and design sketches

- A New Hope: June / July 2014
 - Effort rebooted as glNext – becomes the top priority
 - Unprecedented participation from key ISVs
 - AMD donates Mantle as a starting point

- Renamed and disclosed at GDC 2015

- Public Launch on February 16th, 2016



Introduction

Vulkan vision and goals at project launch

➤ An open-standard, cross-platform graphics+compute API

- Compatibility break with OpenGL
- Start from first principles

➤ Goals

- Clean, modern architecture
- Multi-thread / multicore-friendly
- Greatly reduced CPU overhead
- Full support for both PC and mobile GPU architectures
- More predictable performance – no driver magic
- Improved reliability and consistency between implementations

Introduction

Wide industry support



* Image from Khronos 3D BoF of GDC 2016

- A whole industry, working together
 - GPU and SoC Vendors
 - Game and middleware developers
 - Platform owners, Content providers
- All Khronos resources are open source
 - <http://github.com/KhronosGroup/>

Vulkan is

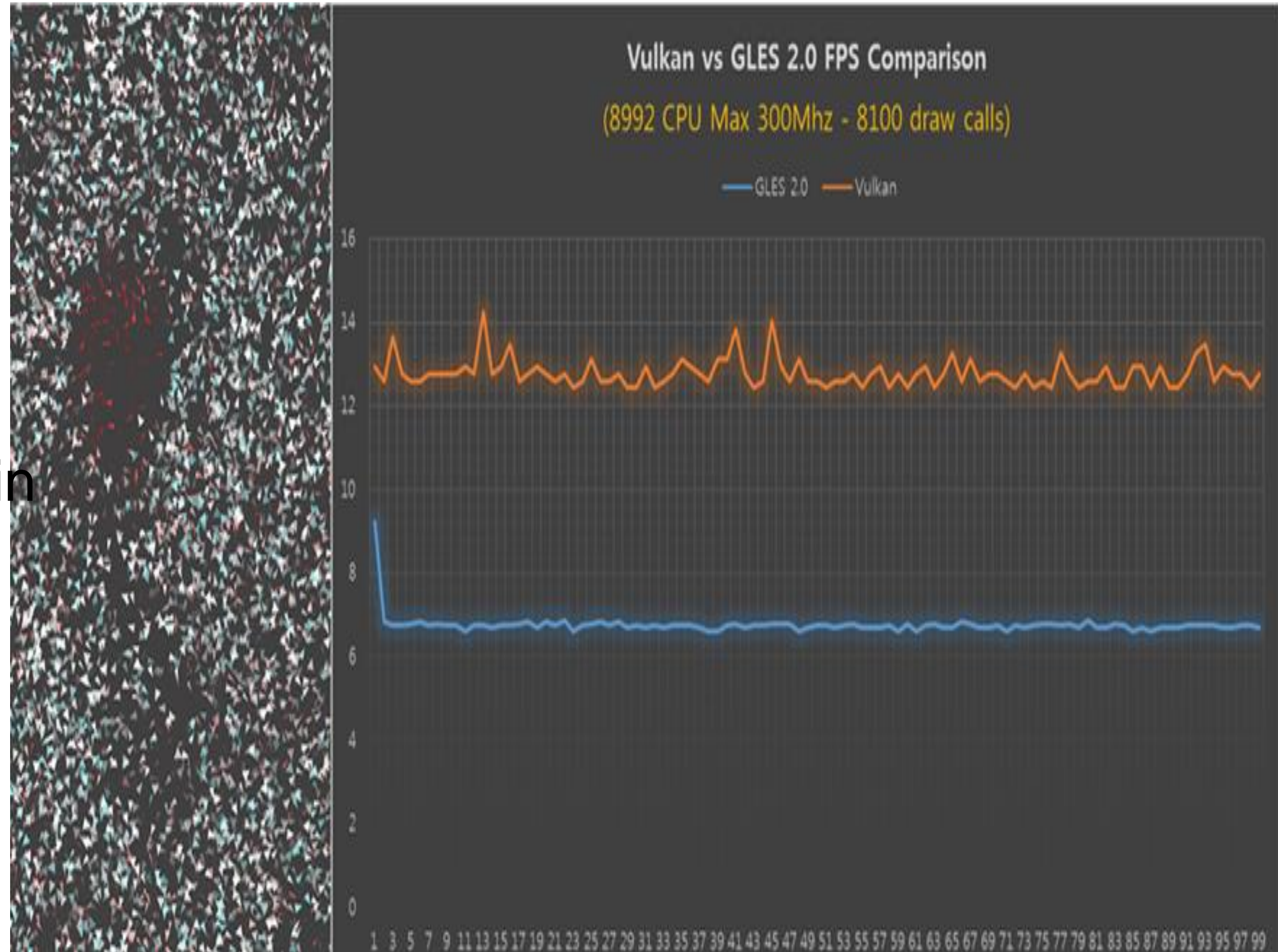
- Open Standard
- Cross Platform
- Ultra light weight
- Predictable/Explicit Control
- Highly efficient API, so we can expect
 - Higher and more stable performance
 - Longer battery life, less thermal problems
 - Allows efficient use of the GPU for higher quality visual graphics



Why Vulkan?

Super efficient

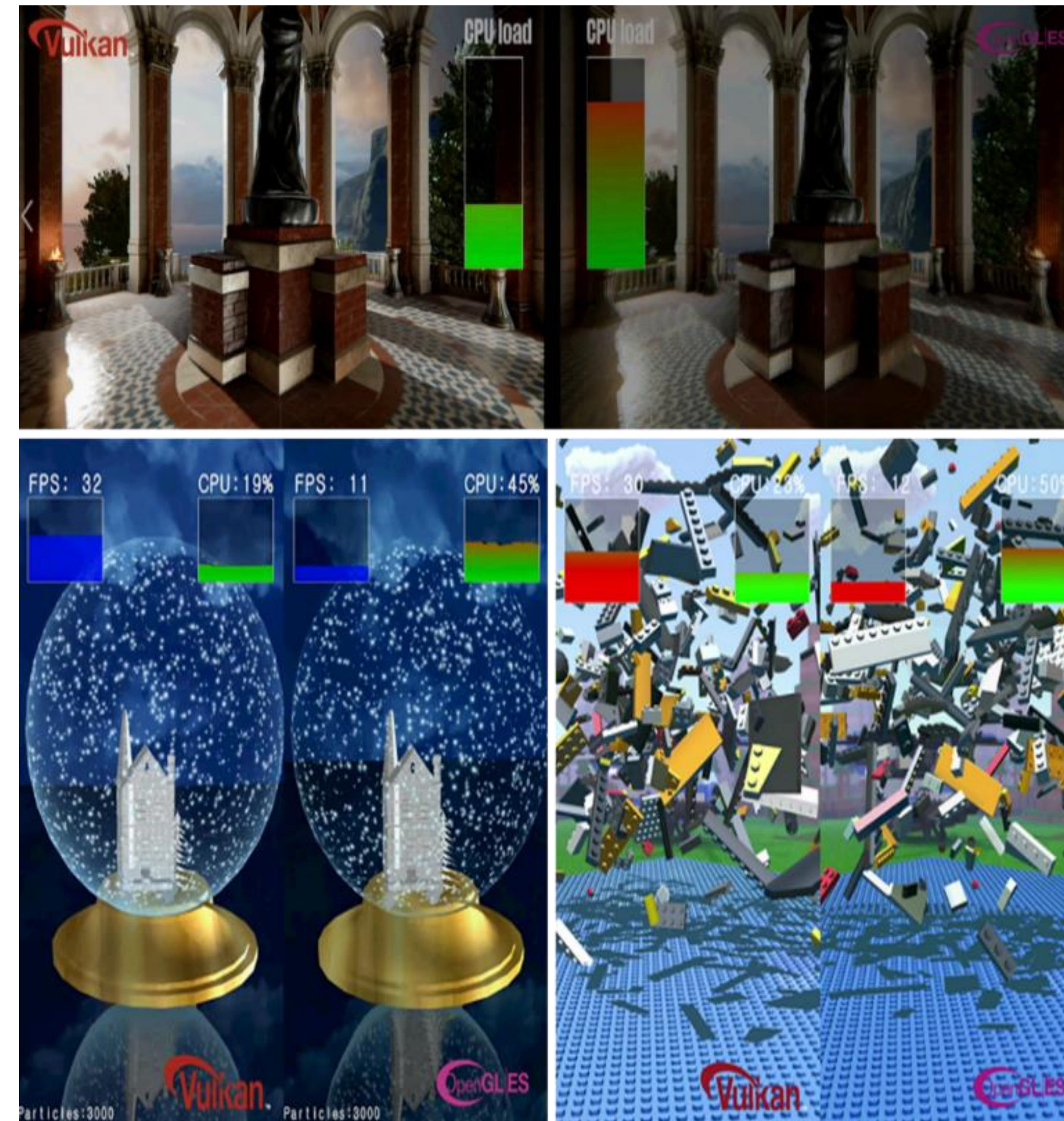
- Did many investigation from the very early stage like mid 2015
- Limitless draw calls and render passes now allowed in mobile product
- It gives real gains like 2X FPS with some scenes



Why Vulkan?

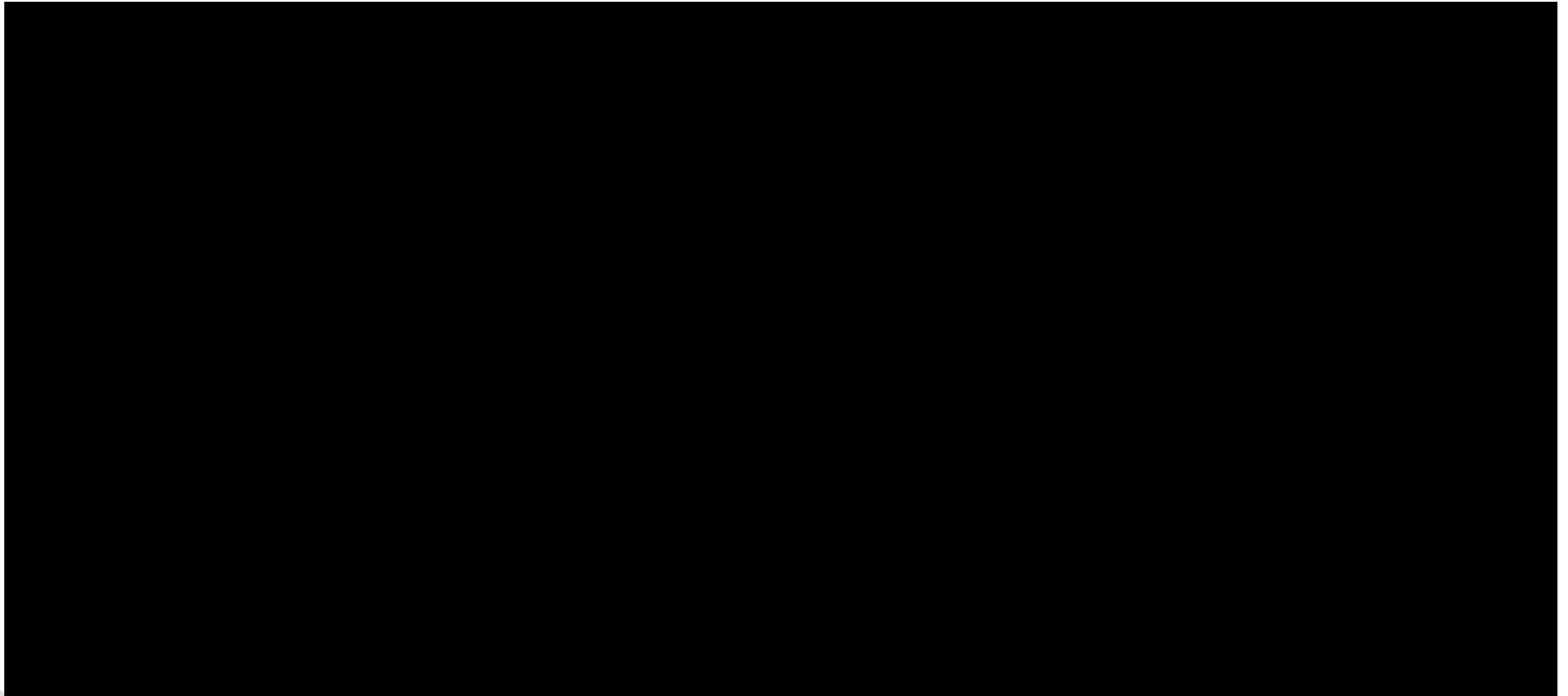
Real beneficial

- More performance or less power / thermal
- More visual effect and post processing can be covered within same hardware resources
- Various explicit ways to optimize application
- Means you can make your game runs faster and look better



Why Vulkan?

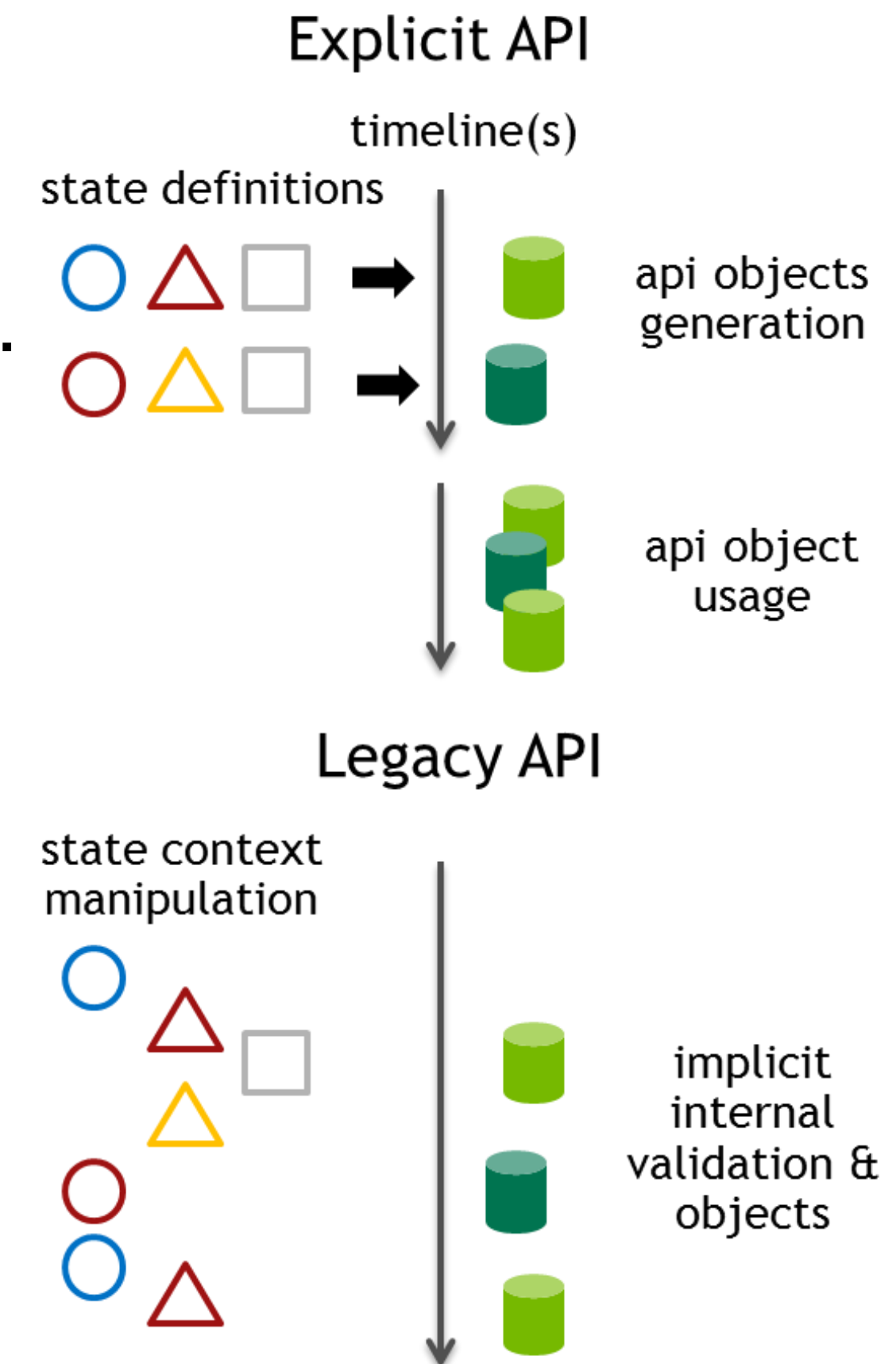
Real beneficial



Explicit API

What it is?

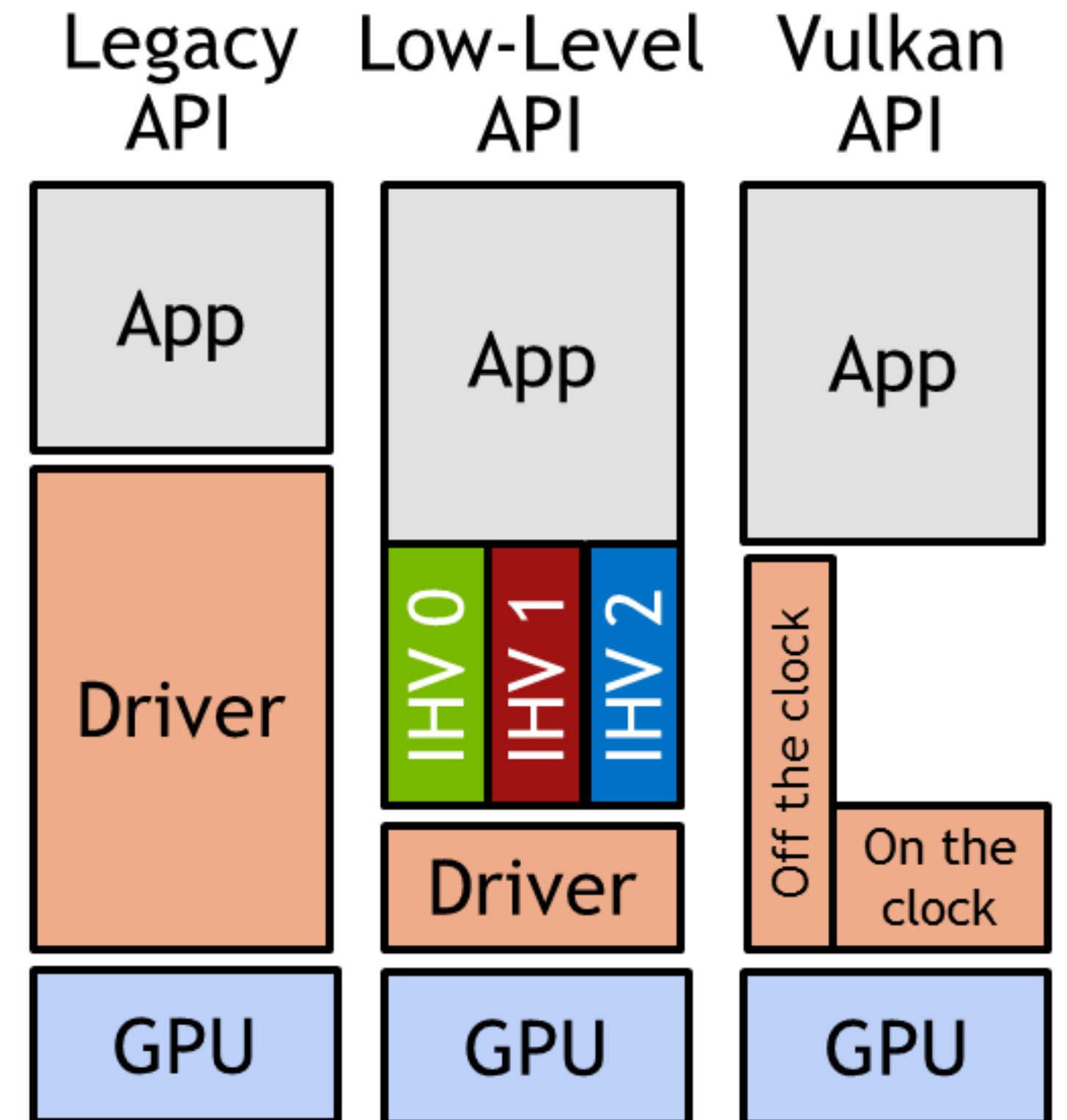
- Providing information at the right time
- Predictable performance costs
 - Creating pipelines, allocating memory, more...
- No driver magic on the clock
 - Remove guesswork and late decision making
- Simpler drivers
- Better scheduling over CPU & GPU work



Explicit API

What it is not?

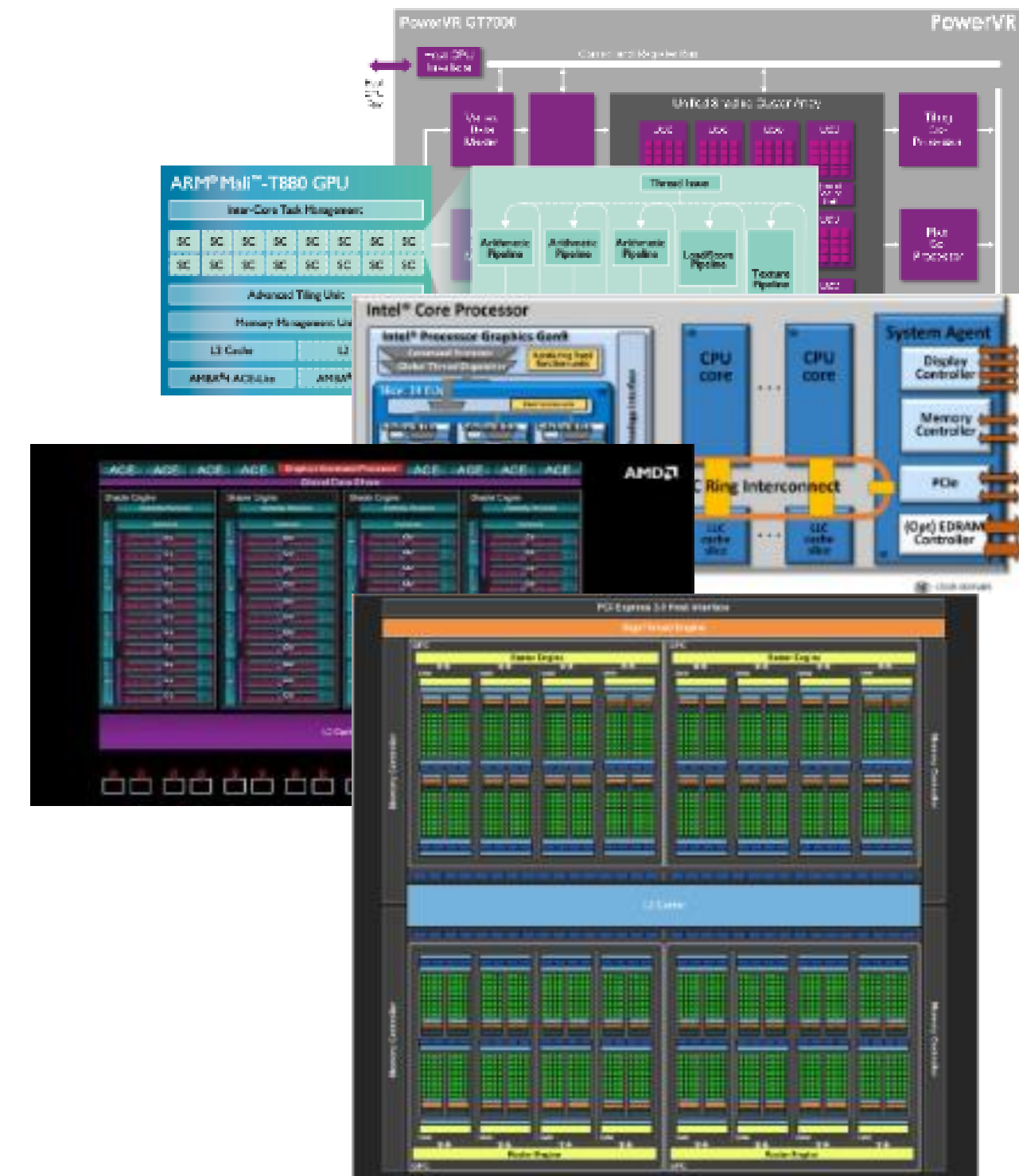
- Low-level == Thin layer over specific HW implementation, little abstraction
 - Not possible given wide variety of hardware
- Making everything the app's problem
- Getting the driver out of the way
- Solves a different problem than we were asked to



Portability

Write once, run anywhere

- Strong desire to avoid forking the ecosystem
- A single API(desktop, mobile)
- Supports various GPU hardware(IMR, TBR, TBDR)



Portability of SPIR-V

- SPIR-V is the new shading language format used in Vulkan
- Cross vendor
- Cross API
- Cross supports Graphics & compute
 - Separates shader source from vendor implementations



Comparison

OpenGL|ES and Vulkan

Issue	Naïve GL	Vulkan
Deterministic state validation/pre-compilation	no	Yes
Improved single thread performance	no	Yes
Multi-threaded work creation	no	yes
Multi-threaded work submission (to driver)	no	yes
GPU based work creation	no	partial (through MDI)
Ability to re-use created work	no	yes
Multi-threaded resource updates	no	Yes
Learning curve	low	Significant
Effort	low	Significant

Unlikely to Benefit

- Scenarios to reconsider coding to Vulkan
 - Need to compatibility to pre-Vulkan platform
 - Heavily CPU-bound application due to non-graphics work
 - Single-threaded application, unlikely to change
 - App can target middle-ware engine, avoiding 3D graphics API dependencies(Consider using an engine targeting Vulkan, instead of coding Vulkan yourself)

Vulkan Game with Unity

Samsung

Lee SeungHwan

Contents

1. Developing Vulkan
2. Unity x Samsung
3. Vulkan in Unity
4. Vulkan Benefits
5. Optimazation
6. Performance check
7. Unity Games

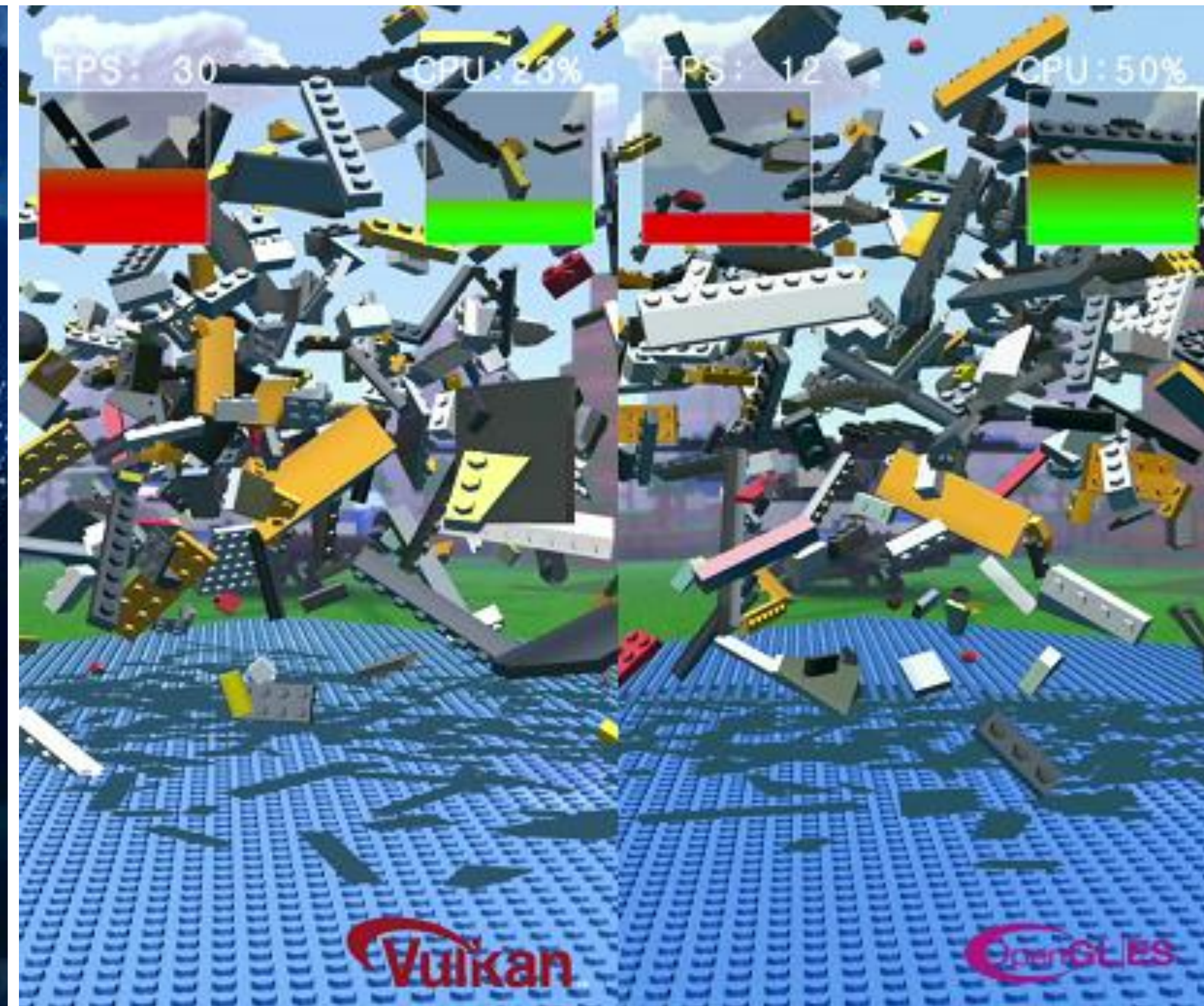
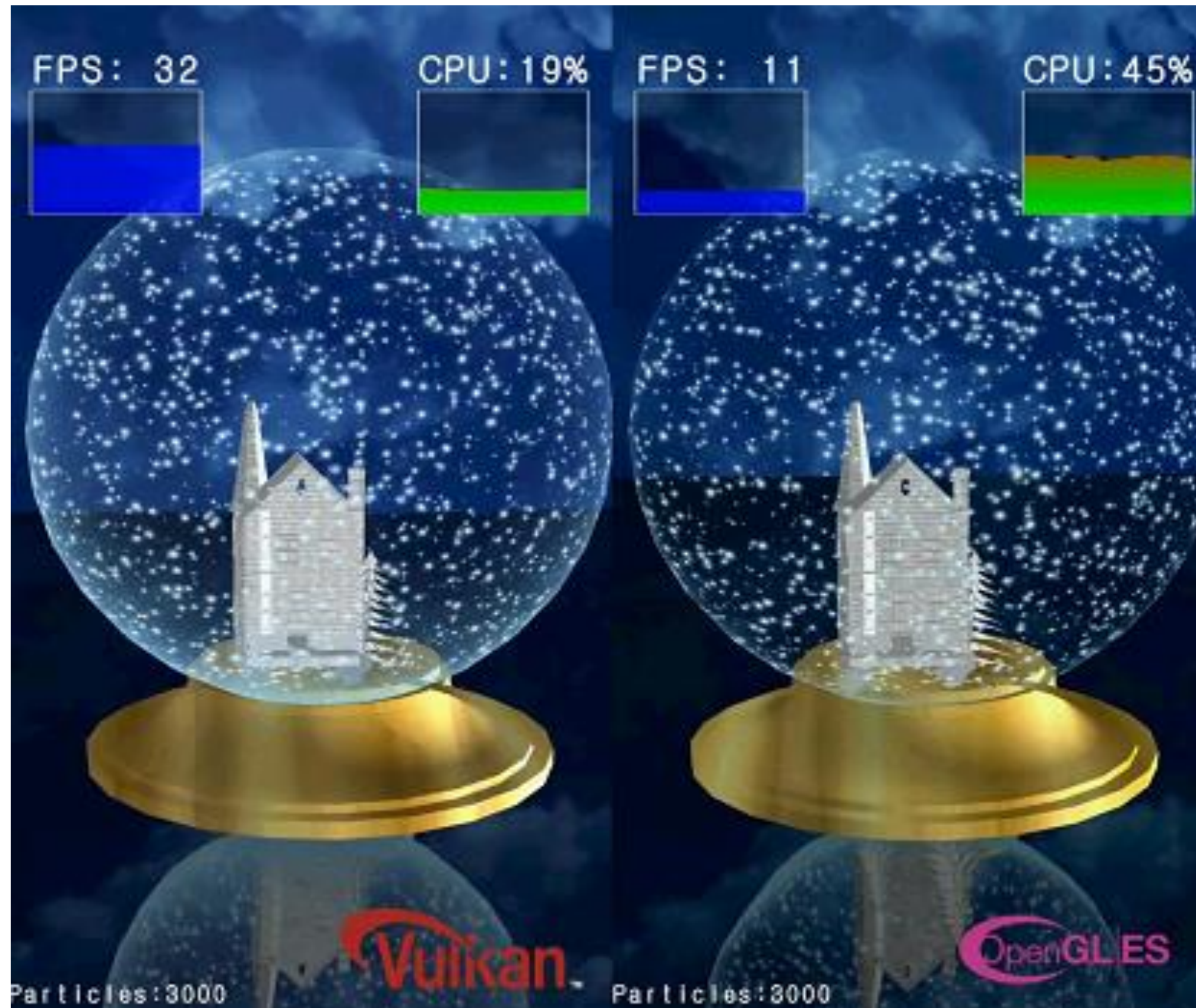
Developing Vulkan

- Khronos in Samsung



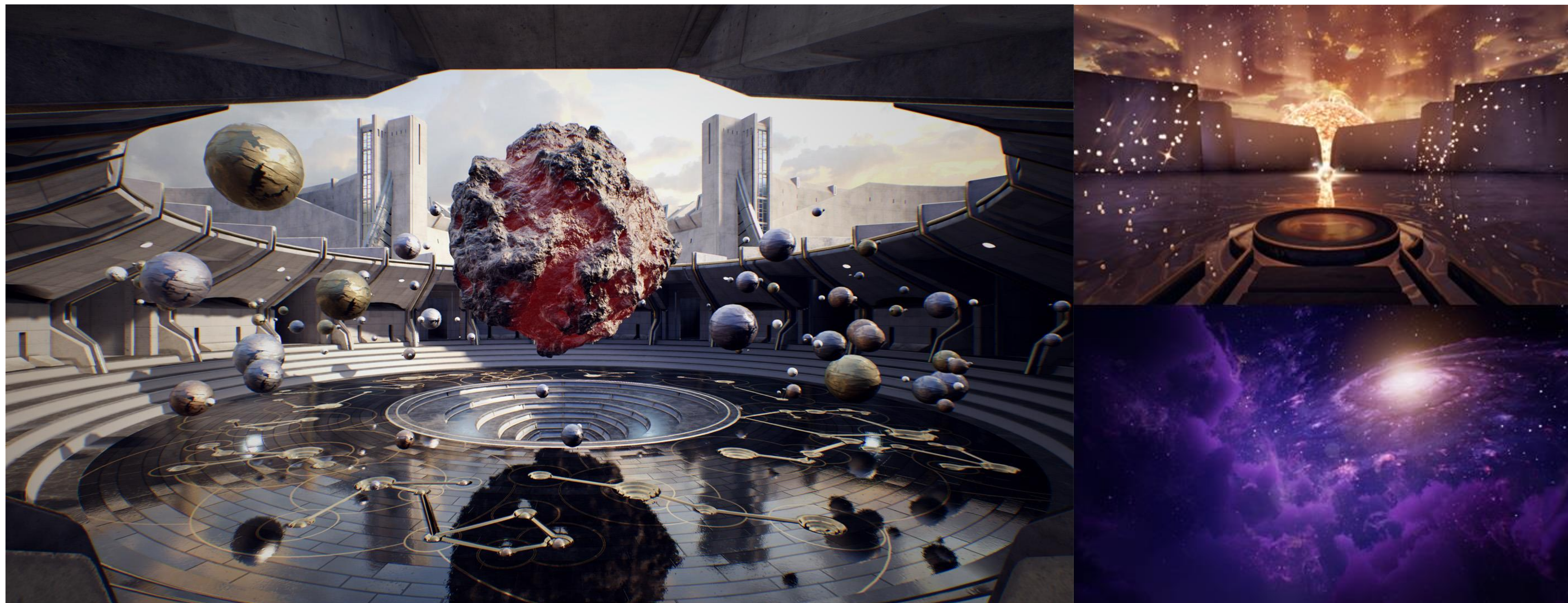
Developing Vulkan

- Demo - Snowball, Lego



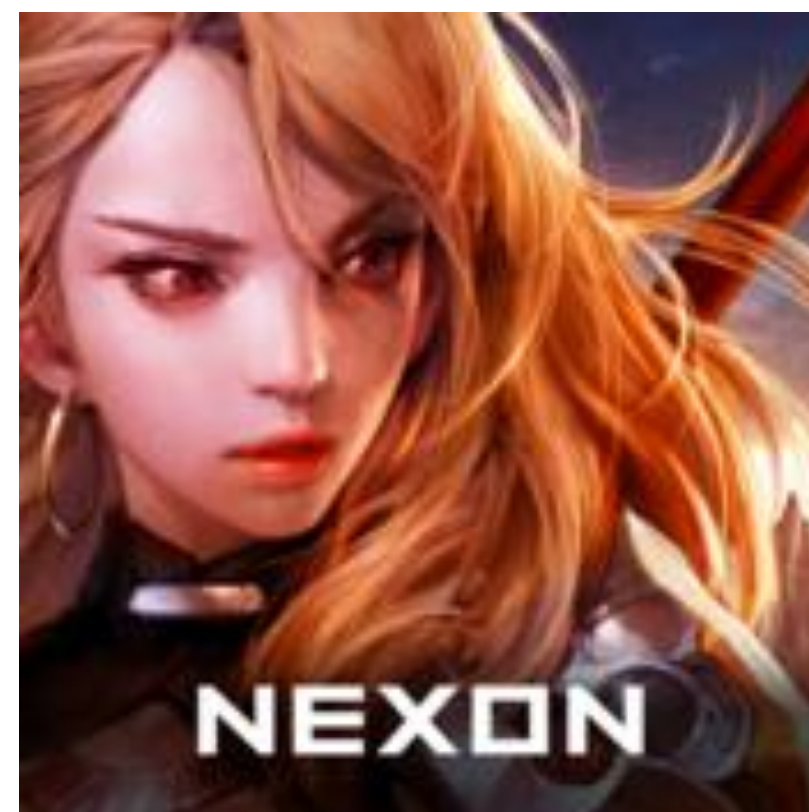
Developing Vulkan

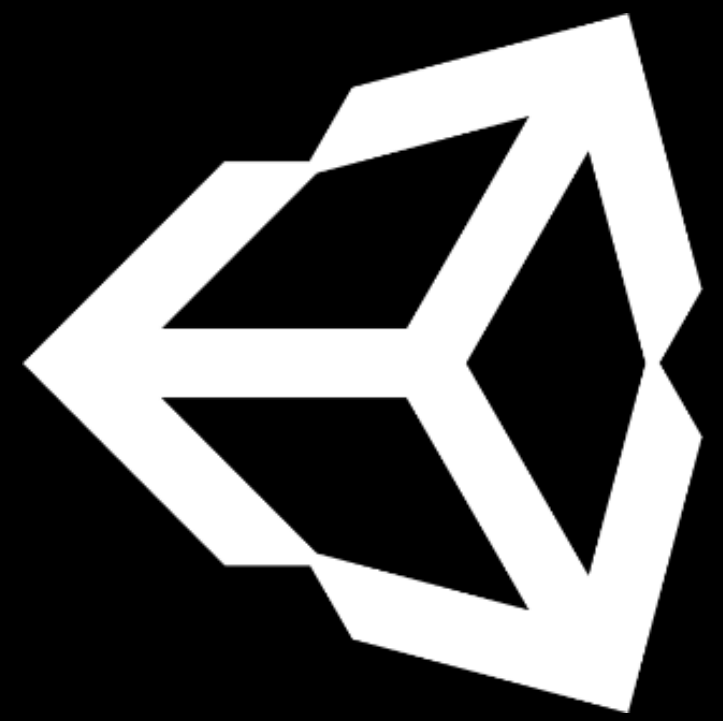
- Demo Game - Protostar



Developing Vulkan

- We decided to support game companies to port their games
- Tight schedule pushed us to focus on some specific directions





unity

X

SAMSUNG

Galaxy

Unity – Samsung Collaboration

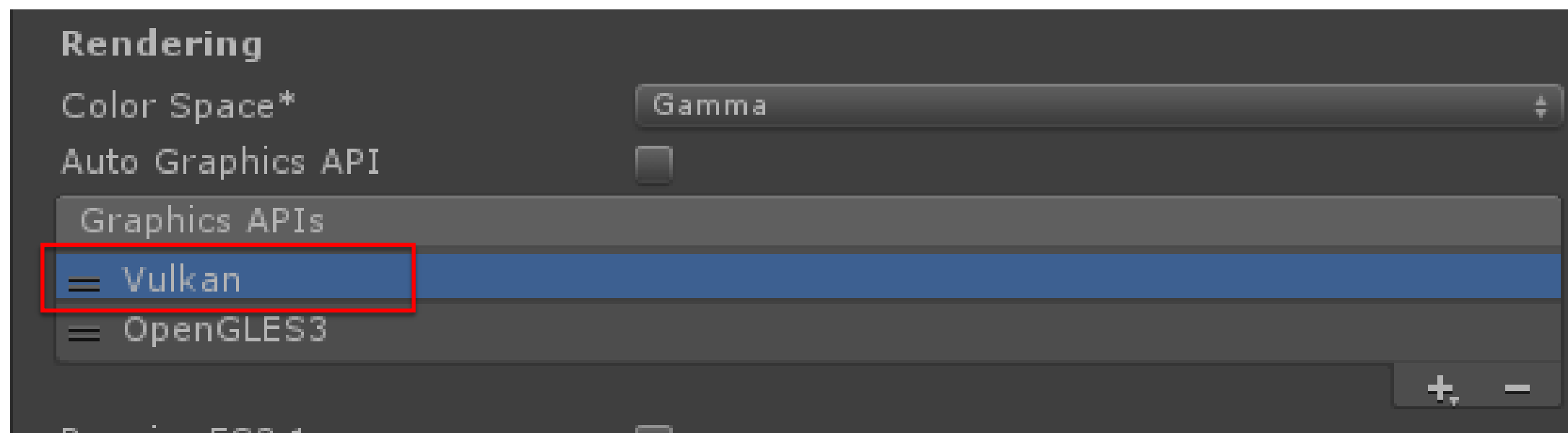
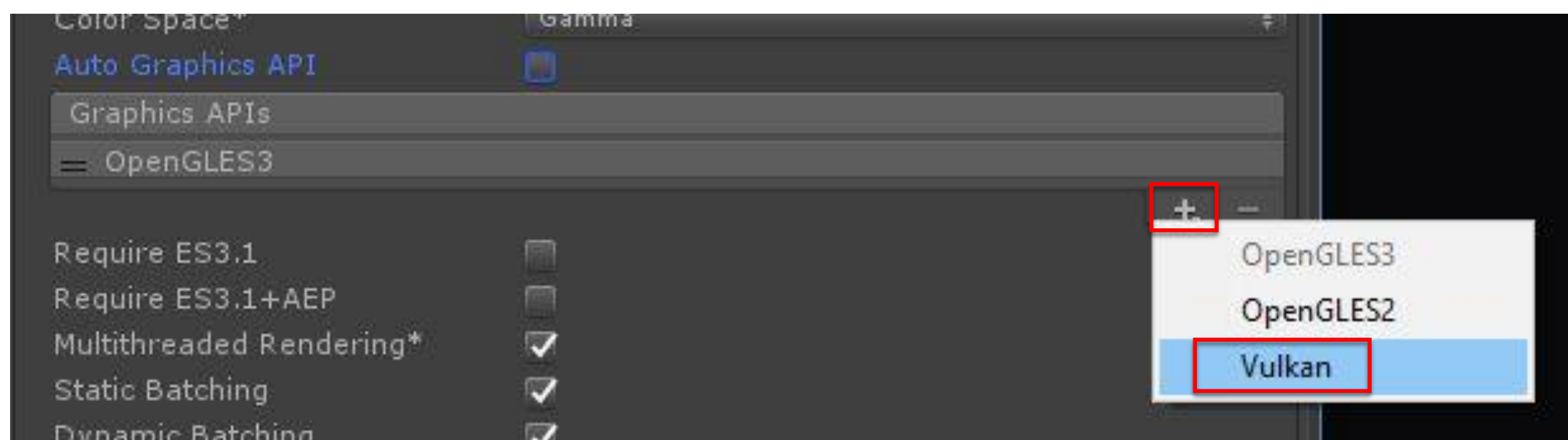
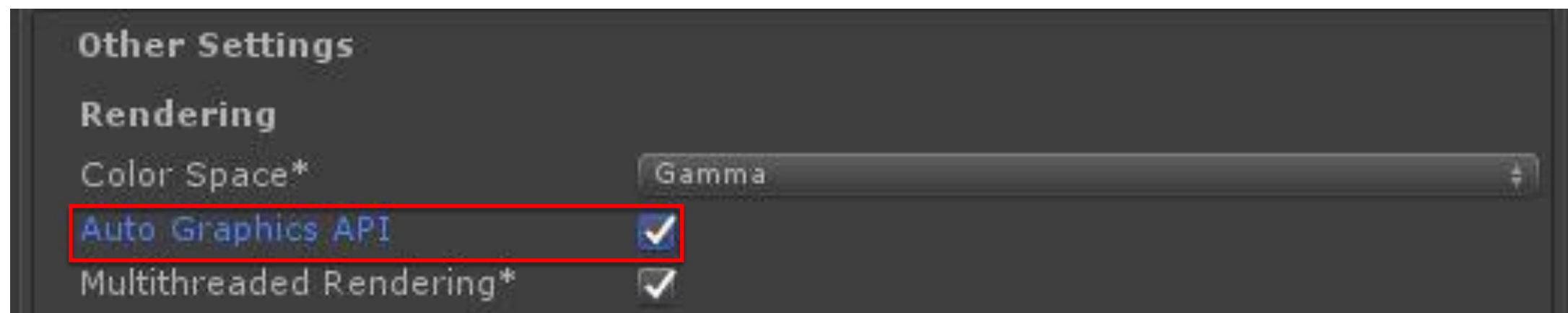
- Samsung – Unity collaborate to improve Vulkan support in Unity
 - Co work in optimizing the Unity Vulkan renderer
 - Support Game developers to make their game with Vulkan
 - Updating GPU driver with better quality and performance
 - Guarantee Galaxy's support for Unity

Why Vulkan is good for Unity?

- Not low-level, but explicit API
 - Lots of the responsibility shifted to the developer
 - Not a beginner's graphics API!
- Allows multithreaded rendering
- Supported in Unity 5.6 and later on Android, Win, Linux
 - "Experimental" on Win and Linux due to no editor support (yet)

Enabling Vulkan in Unity

1. Go to Player Settings inspector
2. Uncheck "Auto Graphics API"
3. Click '+', add Vulkan to the list
4. Drag Vulkan to the top of the list
5. Profit!



Special considerations

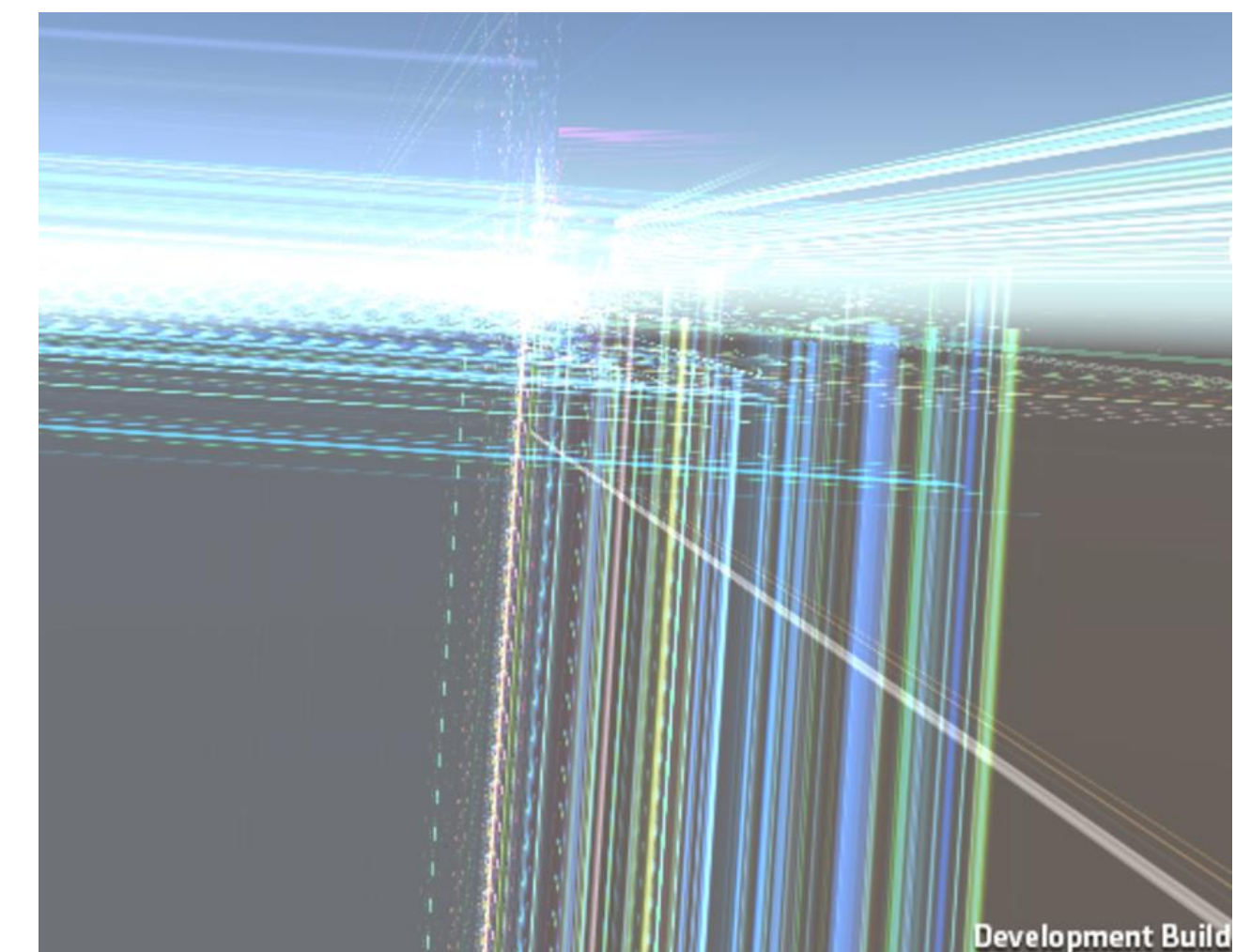
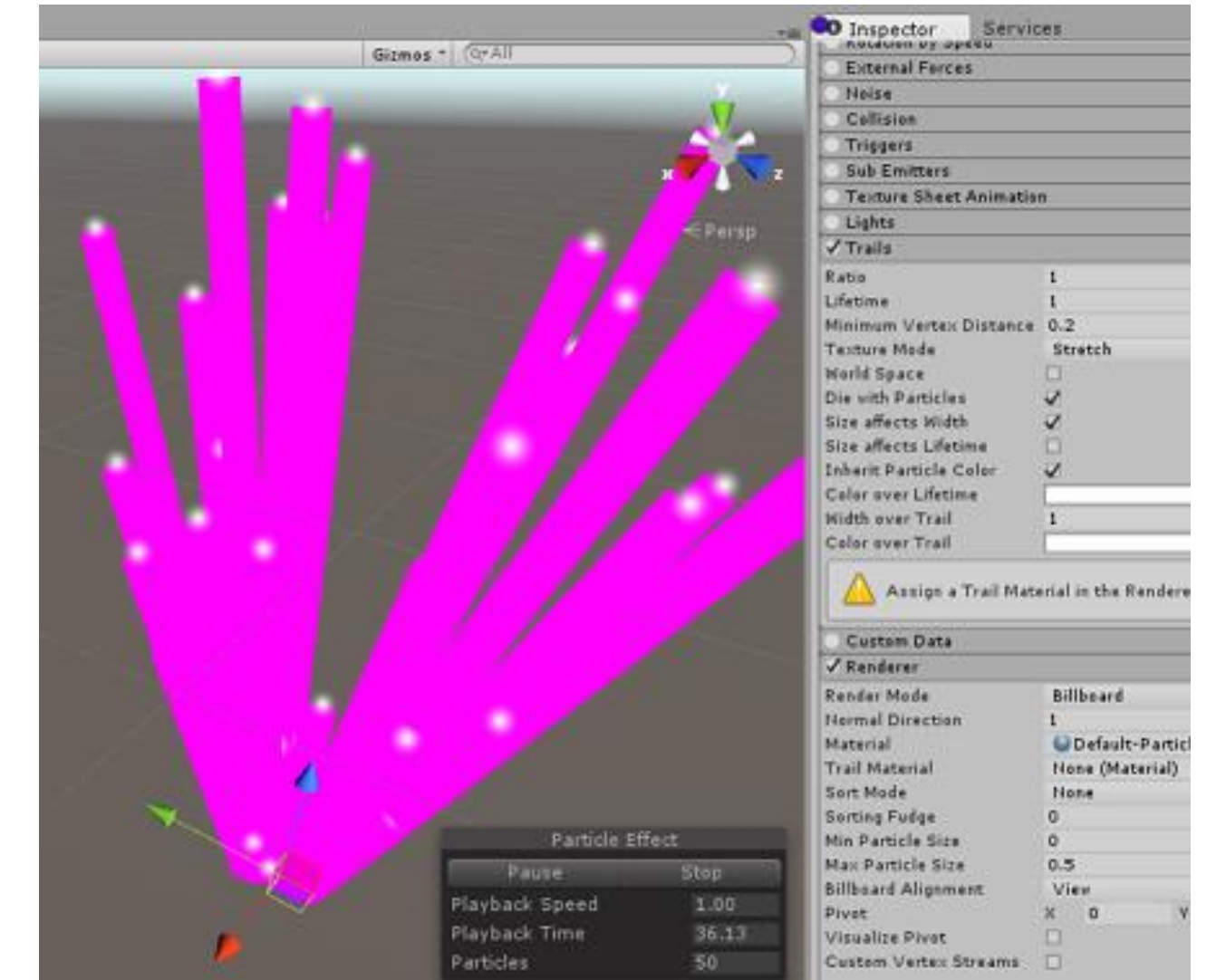
- **None!**
- Actually, a choice between:
 - More eye candy on the screen “for free”, or:
 - Longer battery life
- Draw calls are “almost free” in Vulkan
 - State changes, texture/geometry upload etc has a cost
 - Means more animated stuff on screen at the same time
- Not a magic bullet!
 - GPU still has to draw the same pixels!

Vulkan benefits

- “Zero Driver Overhead”
- Fine-grained control over the GPU
- No need to fight drivers attempting (and failing) to be smart
- “No surprises”
 - Queue submits, flushes, uploads etc. happen exactly when we say so.
 - No surprise shader recompilations

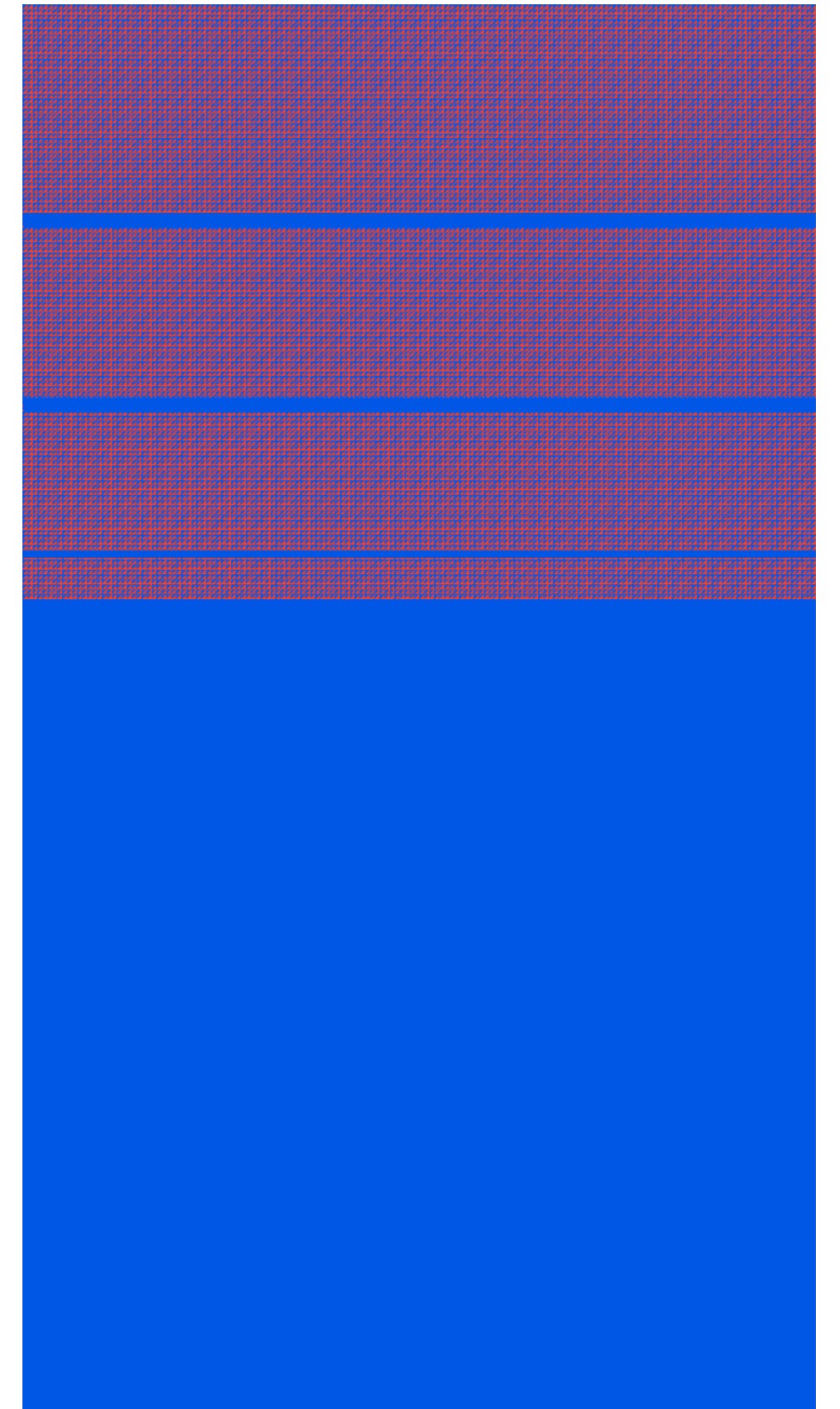
Optimization - Unity

- Use PipelineCache (Disabled by default)
- Use Primary CommandBuffer
- Editor issue



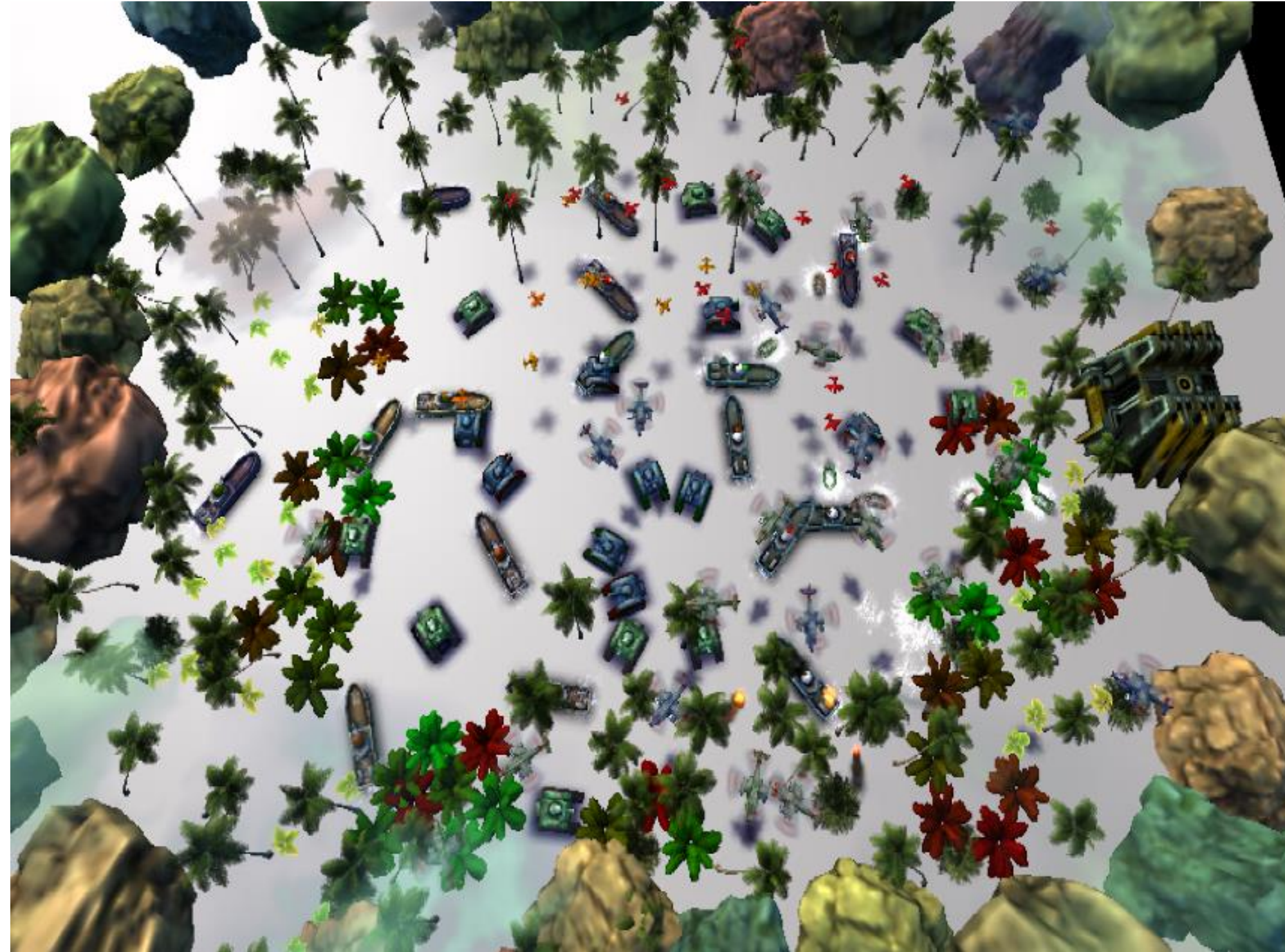
Optimization - Driver

- Use Push Constants
- RenderPass optimization
 - Remove vkCmdClearAttachments
 - Use RenderPass clear flags
- Increase JIT Region

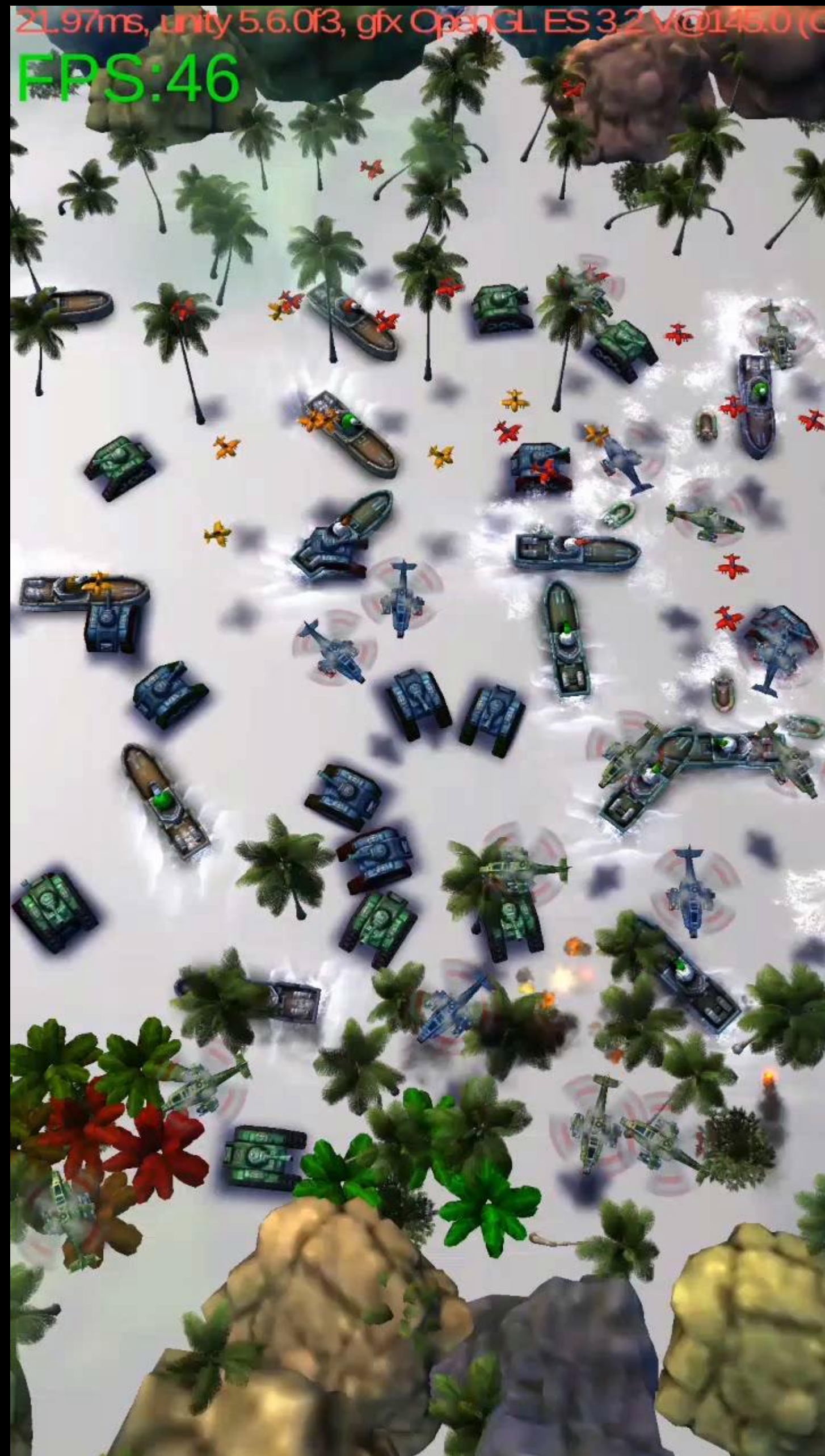


Performance - Skyforce

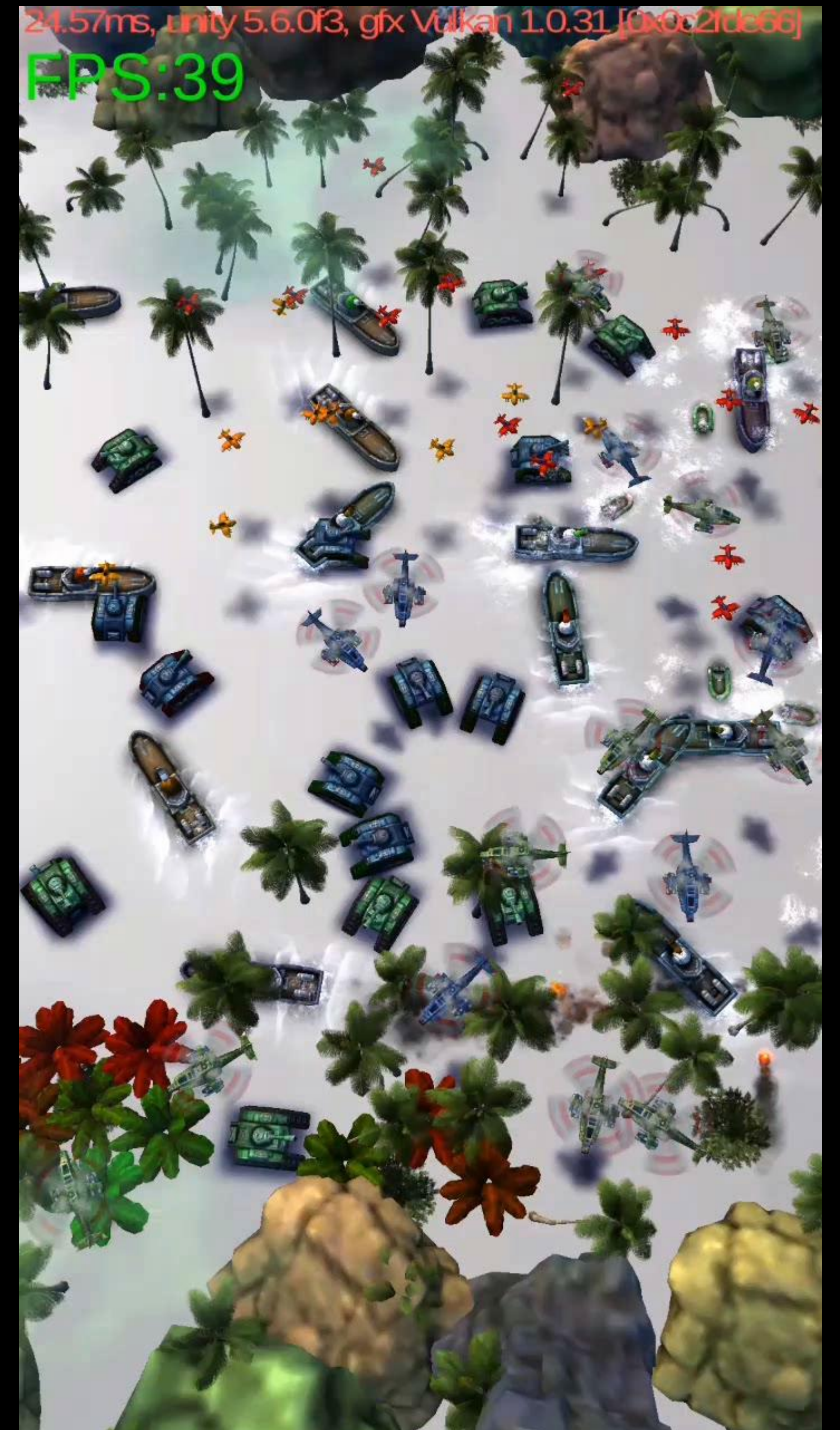
- 600 draw calls
- 100k triangles
- Simple shaders
- Client / worker threading
- 1280 x 720



• GLES



• VULKAN



Performance - Skyforce (cont'd)

- 1198 draw calls, 223k triangles

Frame times [ms]	OpenGL ES 3.x	Vulkan
Samsung Galaxy S7 G930F (Mali T880)	35	25
Samsung Galaxy S7 G930V (Adreno 530)	40	21

- Stripped down, 198 draw calls, 56k triangles, 60fps

CPU utilization of worker thread	OpenGL ES 3.x	Vulkan
Samsung Galaxy S7 G930F (Mali T880)	70%, ~1100MHz	58%, ~1000MHz
Samsung Galaxy S7 G930V (Adreno 530)	59%, ~1500MHz	38%, ~1200MHz

Performance - Adam

- Project Adam

Draw calls	381
Triangles	820 k
Vertices	1.2 M
Resolution	1920 x 1080



• GLES

FPS:23



• VULKAN

FPS:32



Performance – Adam (cont'd)

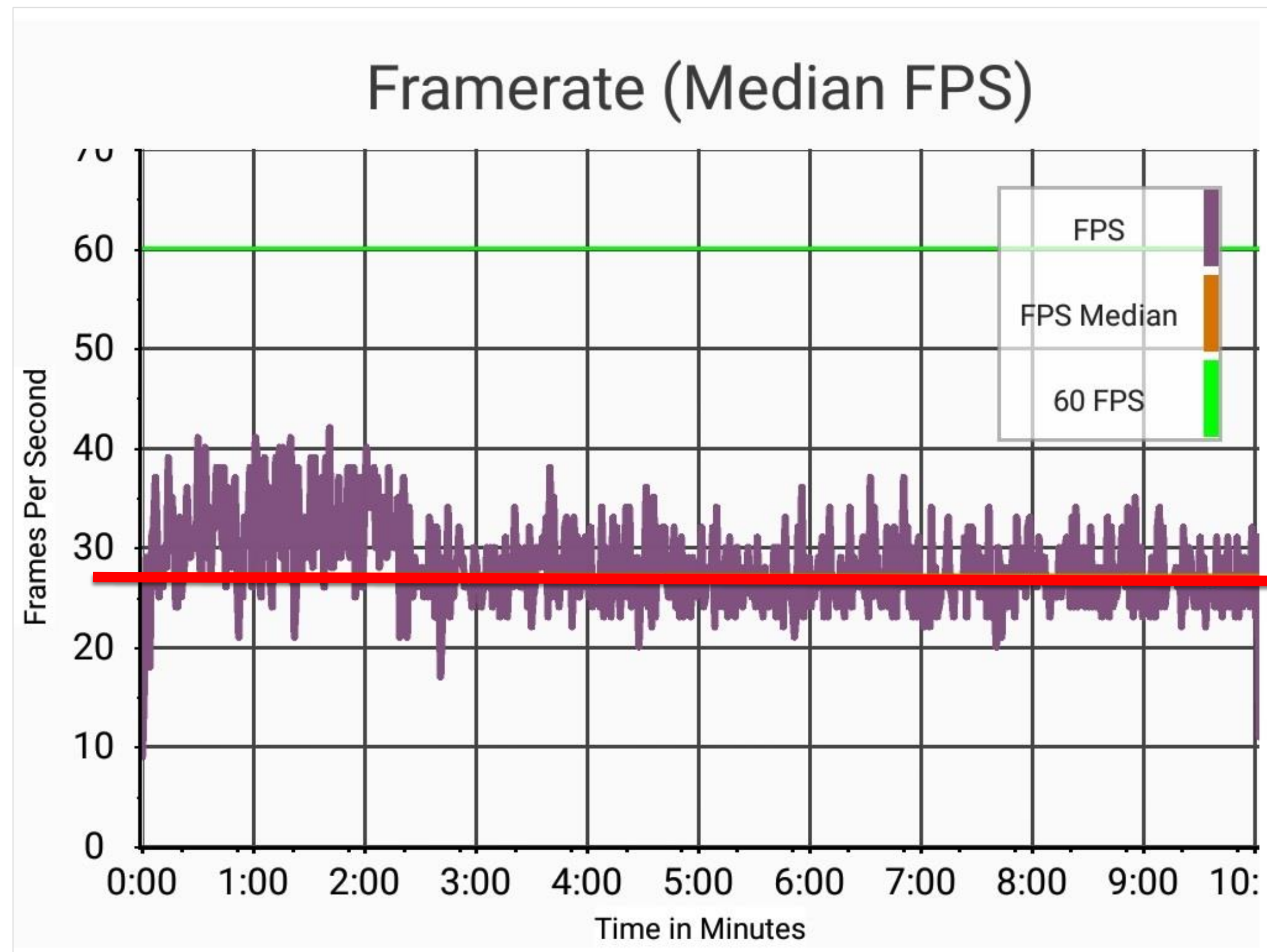
- Test results with Galaxy S7 G930V (Adreno 530)

	OpenGL ES 3.x	Vulkan
FPS	27 FPS	39 FPS
CPU usage	22 %	24 %
GPU usage	49 %	87 %
Resource usage	563 MB	618 MB
FPS stability	82 %	96 %

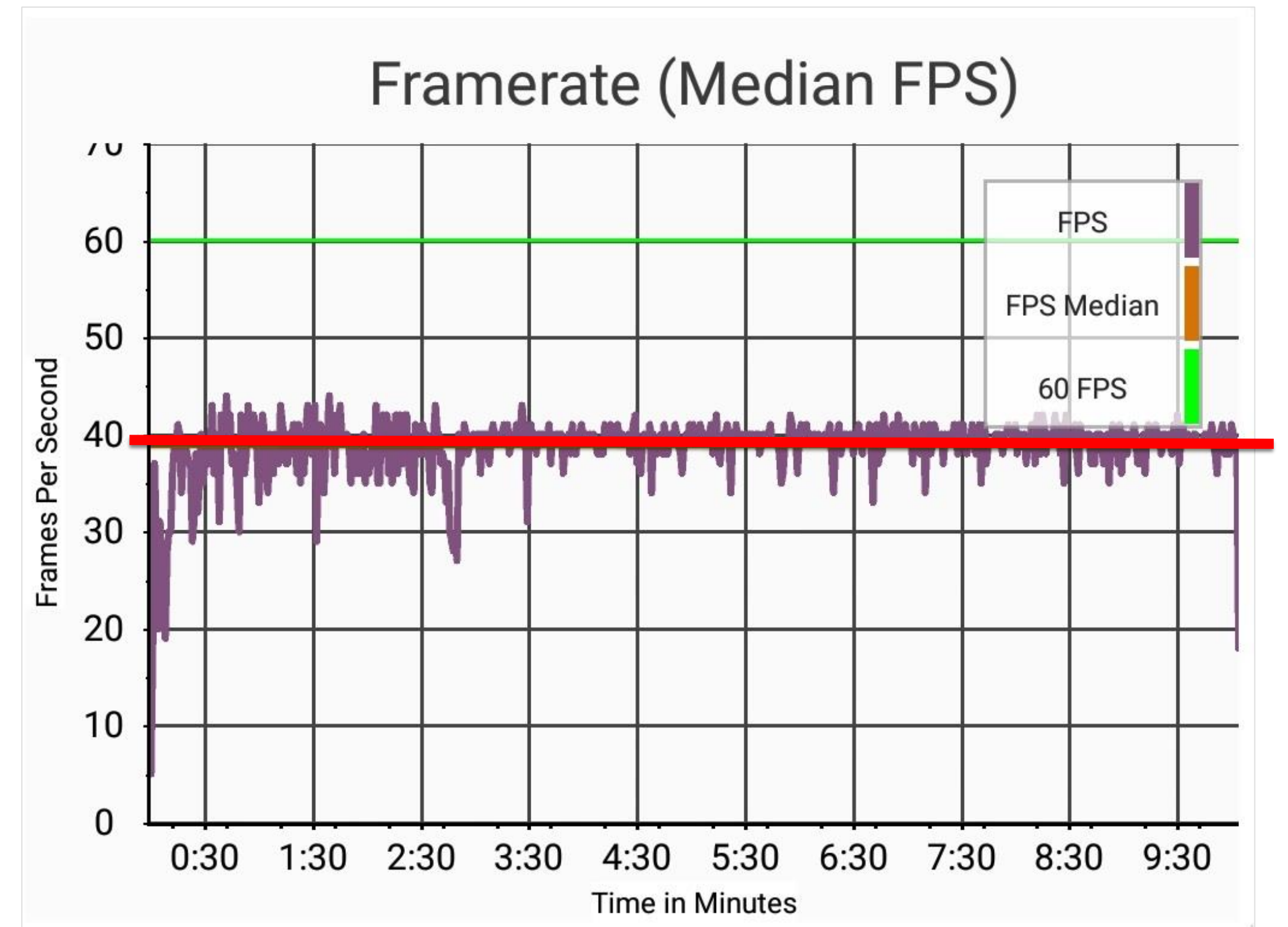
Performance – Adam (cont'd)

- FPS Stability

- GLES (27 FPS, 82%)



- Vulkan (39 FPS, 96%)



Unity Games with Vulkan

- Unity Game we are working on

	OpenGL ES	Vulkan
FPS	44 FPS	48 FPS
CPU usage	36 %	35 %
GPU usage	65 %	61 %
Resource usage	627 MB	895 MB

Samsung GameDev Program

Samsung

Jang Daemyung

History

Khronos Standard

Internal Research

Galaxy GameDev

2014

2015

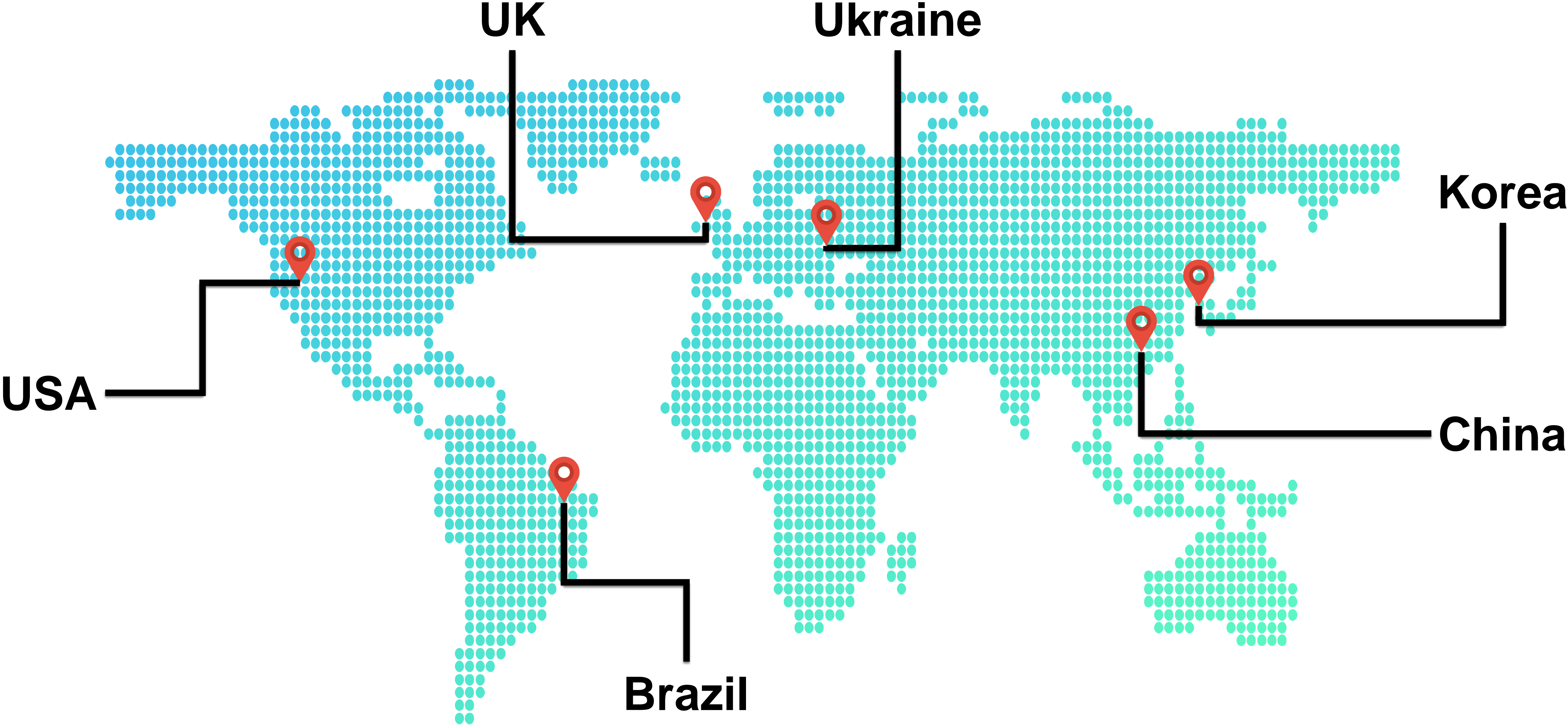
2016

2017

GPU Driver Development

Game Engine Contribution

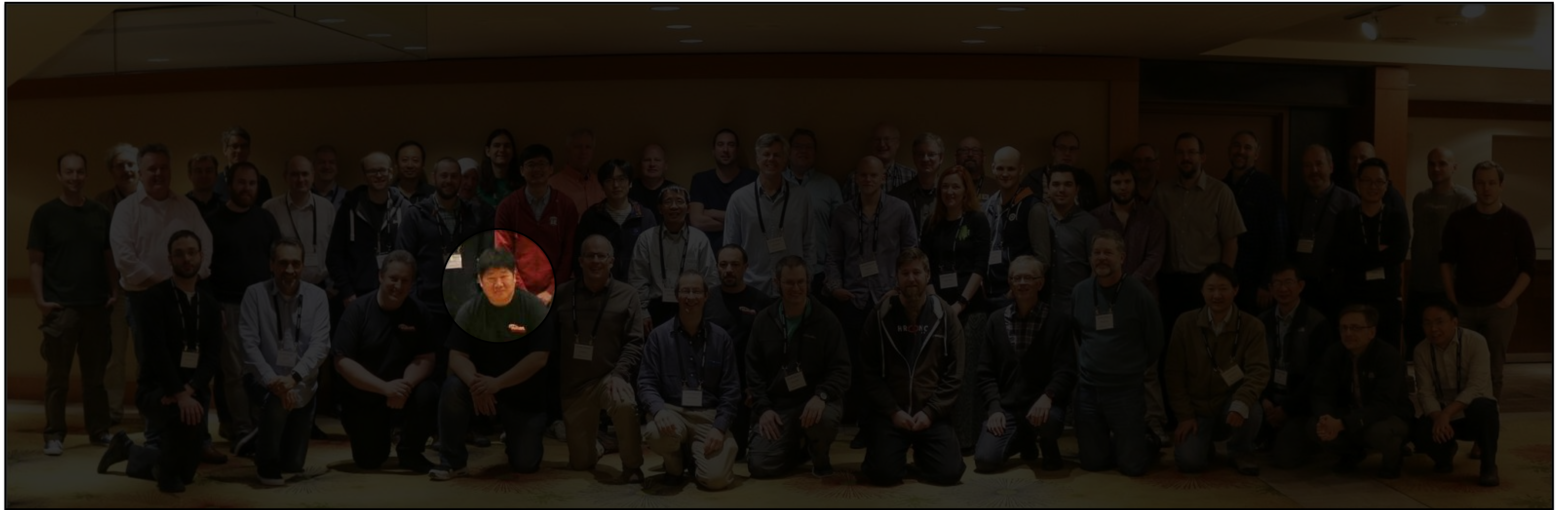
Location



Khronos Vulkan



Khronos Vulkan



Game Engine



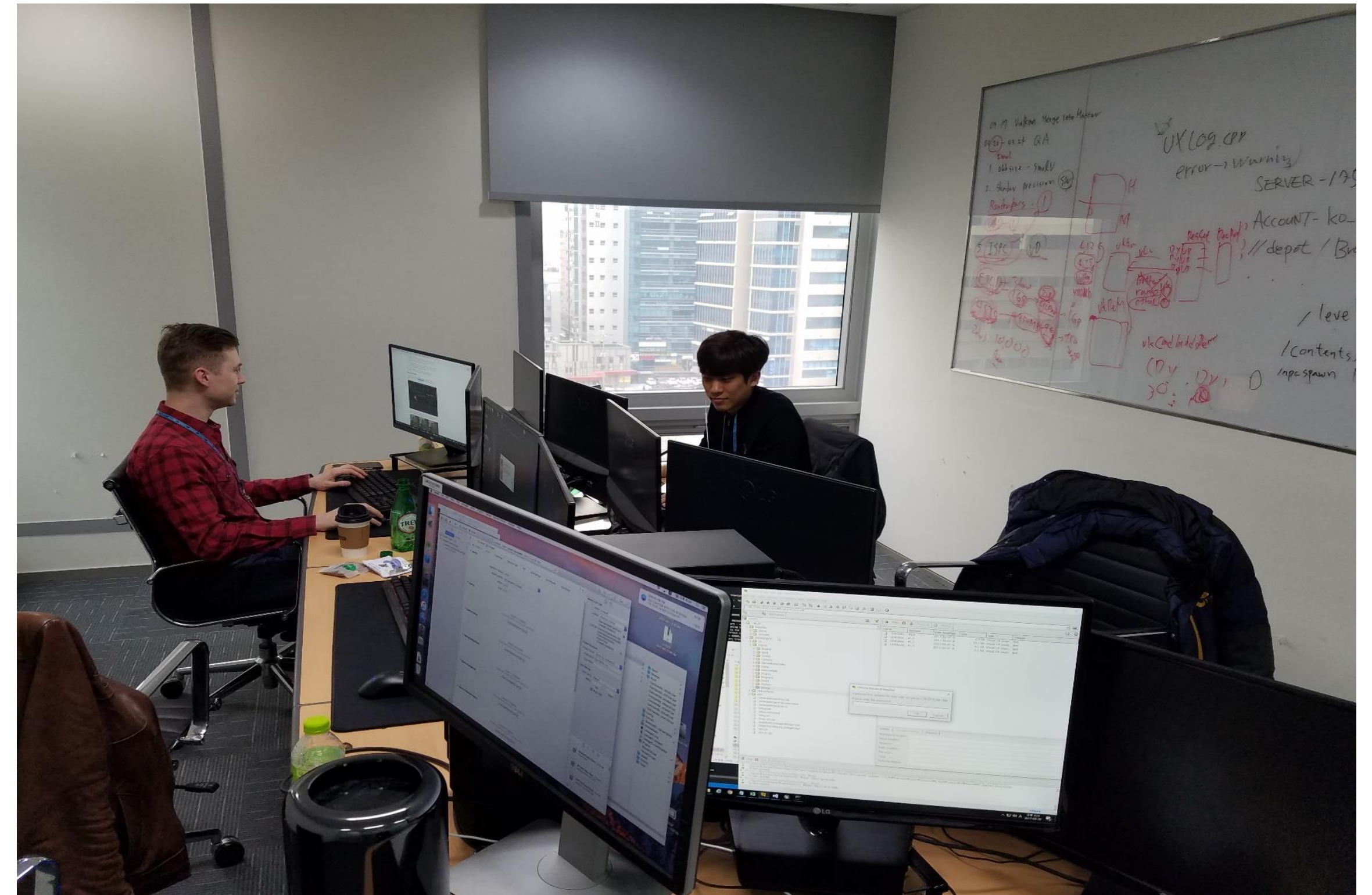
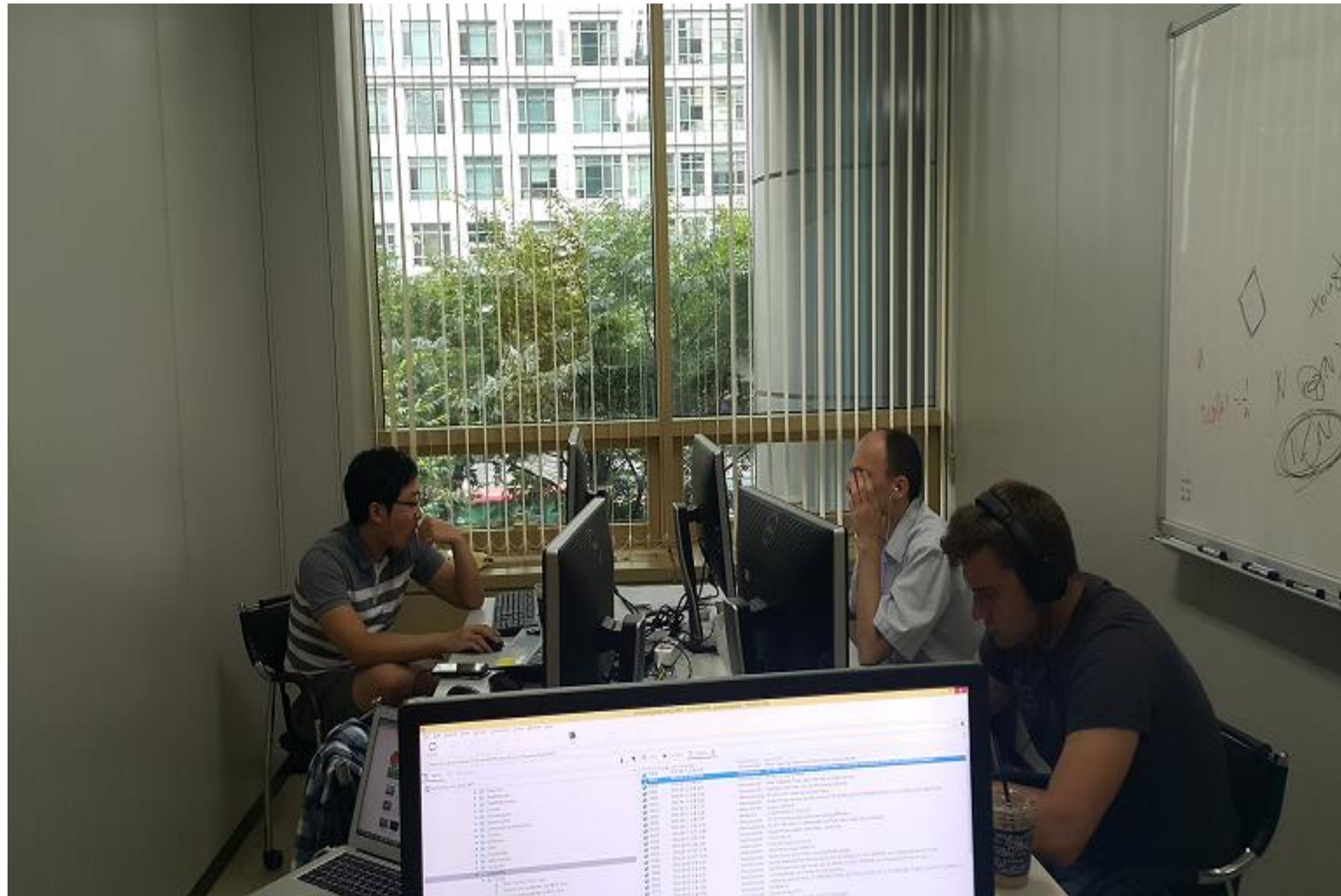
Unreal 4.12 supports Vulkan

Game Engine

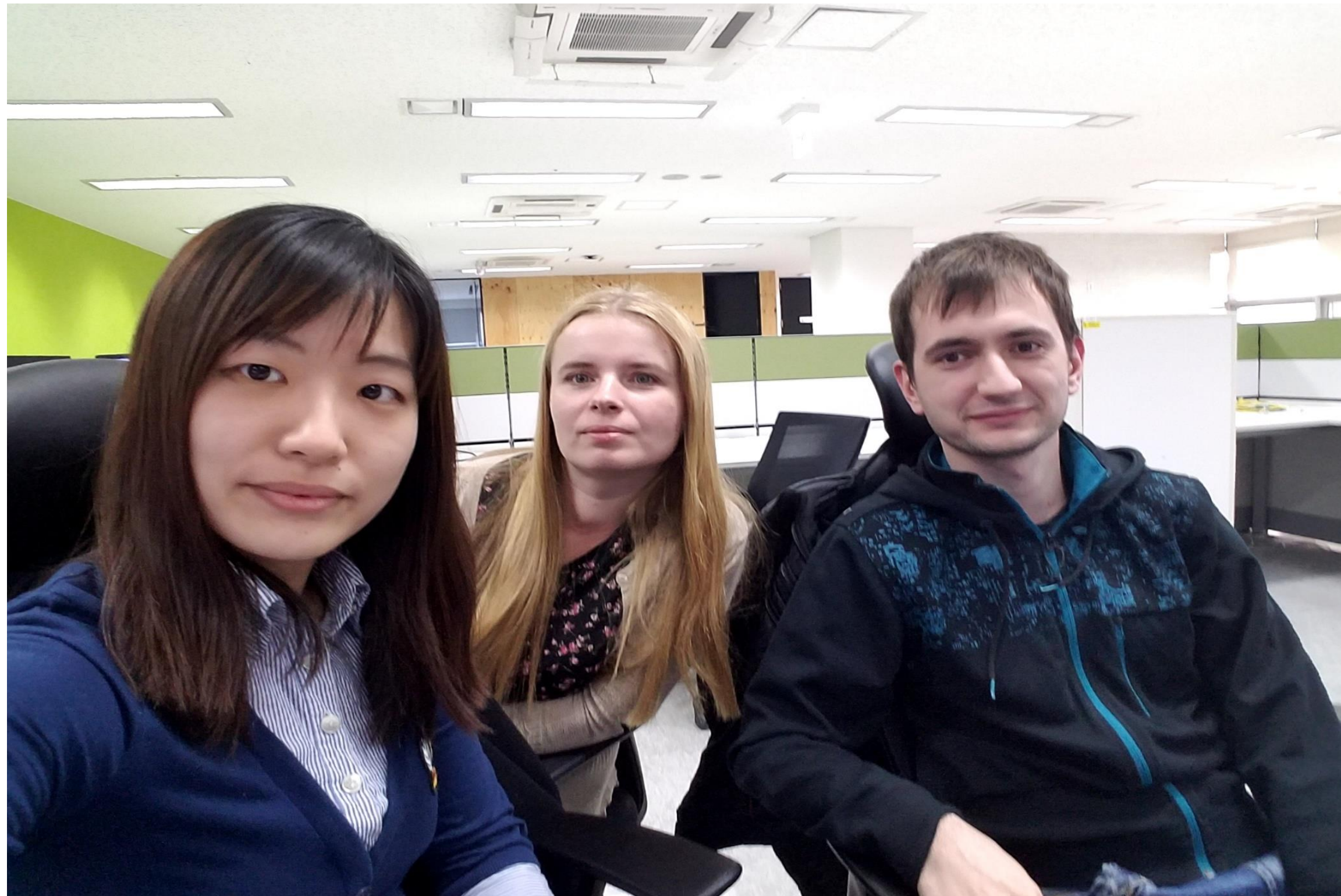


Unity 5.6 supports Vulkan

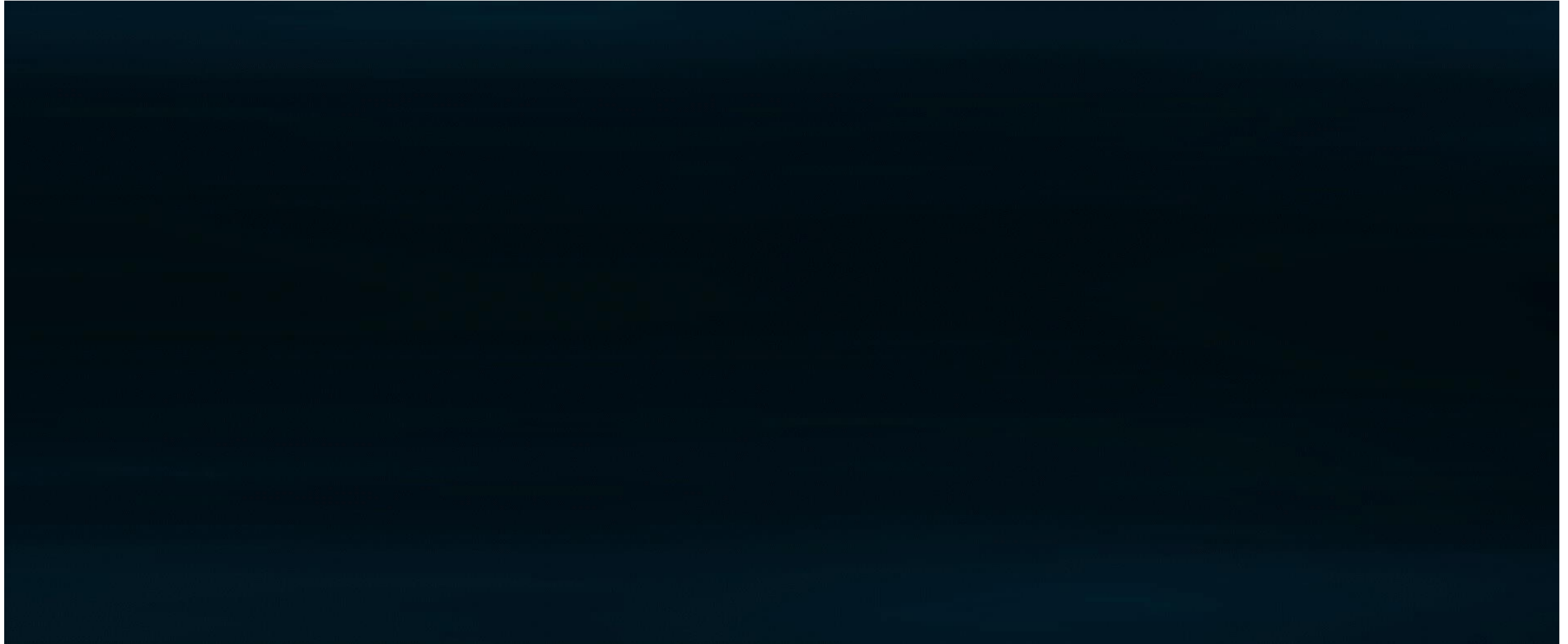
On Site Game Studio



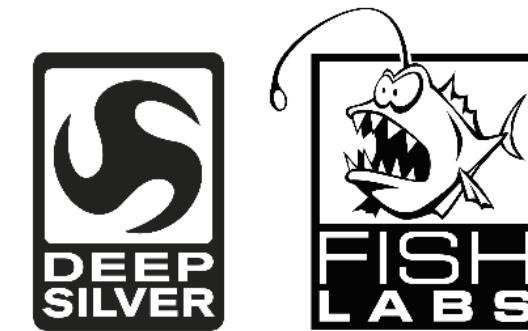
On Site Game Studio



Demo



Partners



Vulkan Devices



More Than 100 Million!!!

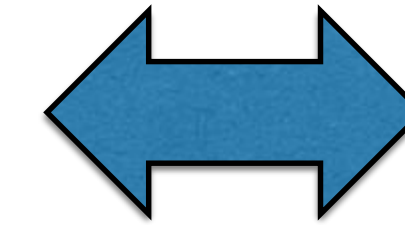
Galaxy GameDev Advantage

Game Engine

OS Platform

GPU Vendor

SAMSUNG



Game Studio

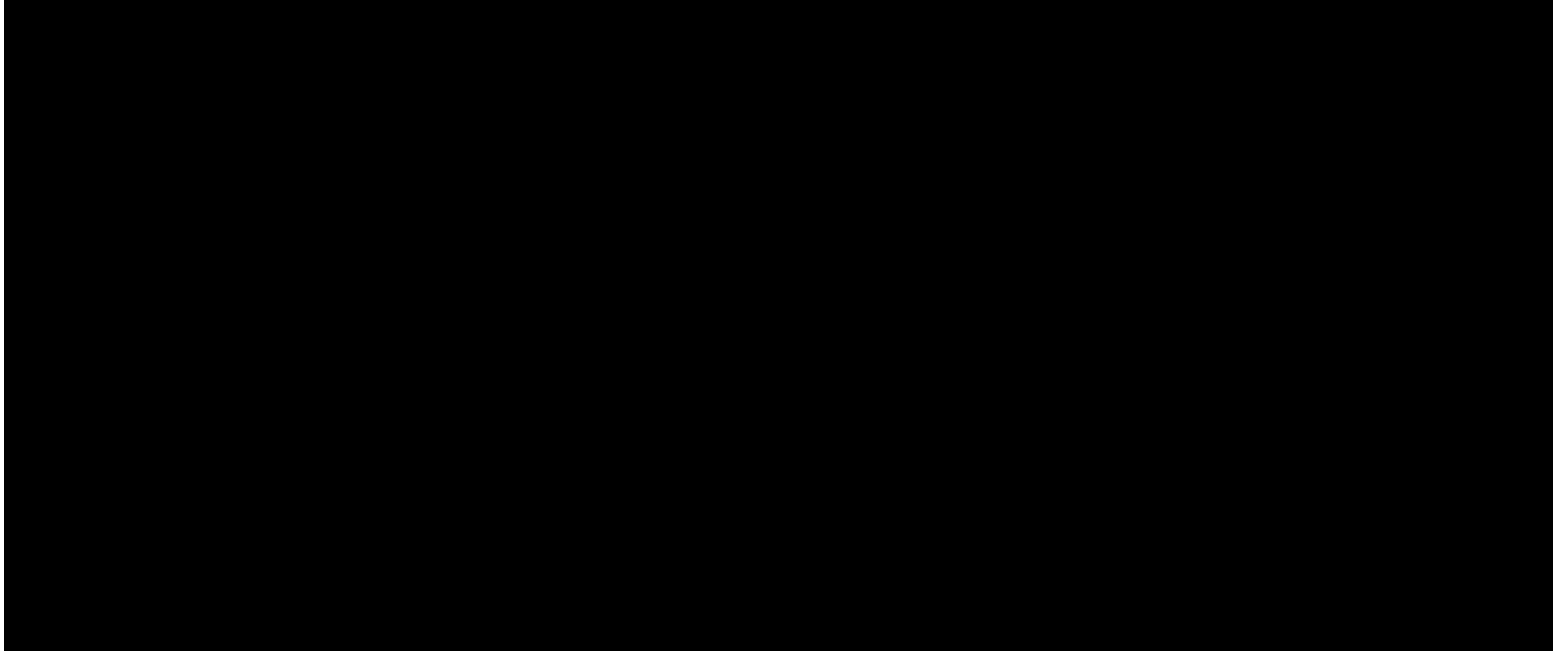
Galaxy GameDev Contact

Samsung will keep go on supporting game developers and players!

If you have any questions, offers or suggestion, please contact

gamedev@samsung.com

Samsung DeX

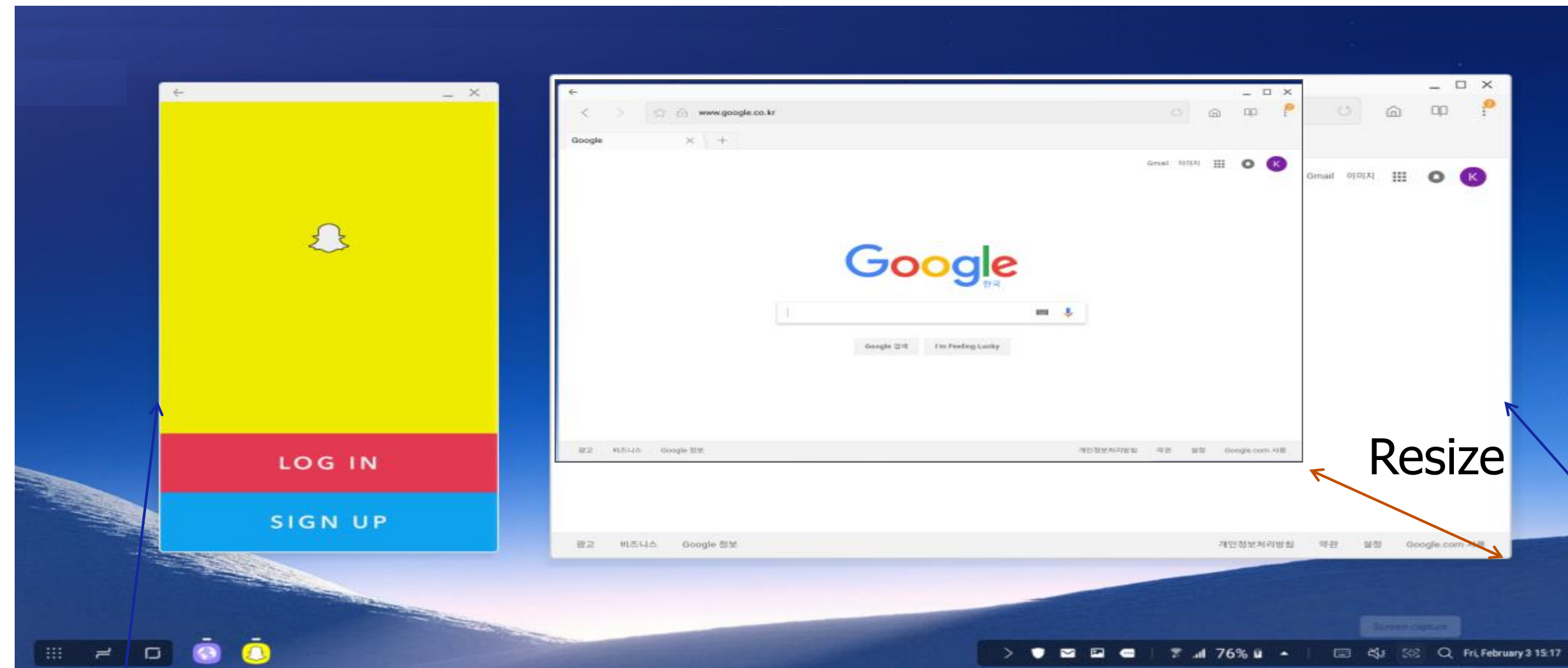


<https://www.youtube.com/watch?v=hWFFpViiamE>

3rd party app eco system support

Basically, Most of Android app will be executed in DeX Mode

There are two options: **Fixed Size** VS **Resizable** Application in DeX Mode



Apps that didn't declare Android 7.0 multi window support will be launch as a Fixed size (731 X 411)

- No resize (Fixed window)
- Minimize
- Close
- Rotation (If app supports rotation)

Apps that **Explicitly** declared Android 7.0 multi window with `android:resizeableActivity="true"` will be launched as Resizable window

- Resizable window
- Minimize
- Maximize
- Close

For more information, please visit <http://developer.samsung.com/samsung-dex>

Game App의 Samsung DeX 내 실행을 위한 요구 사항

- **Multi Density 지원**
 - mdpi 추가 지원 필요
- **Mouse 입력**
 - 마우스 동작 미지원시 DeX 내에서 앱 실행 안됨 혹은 실행 후 동작안됨
 - 1) 실행전 Manifest에서 선언한 경우
 - 2) 실행 도중 런타임으로 Mouse 입력을 막은 경우
- **Freeform MultiWindow**
 - 확대 / 축소 및 구현에 따라 immersive 모드 지원 가능
 - ❖ Freeform 미지원시 : 고정형 사이즈로 실행
- **최적화**

Demo



Thank You.

NEXON COMPANY