DEV.BYTES

OpenGL [ES] Optimizations

+Shanee Nishry
@Lunarsong

Performance. Why should I care?

Performance. Why should I care?

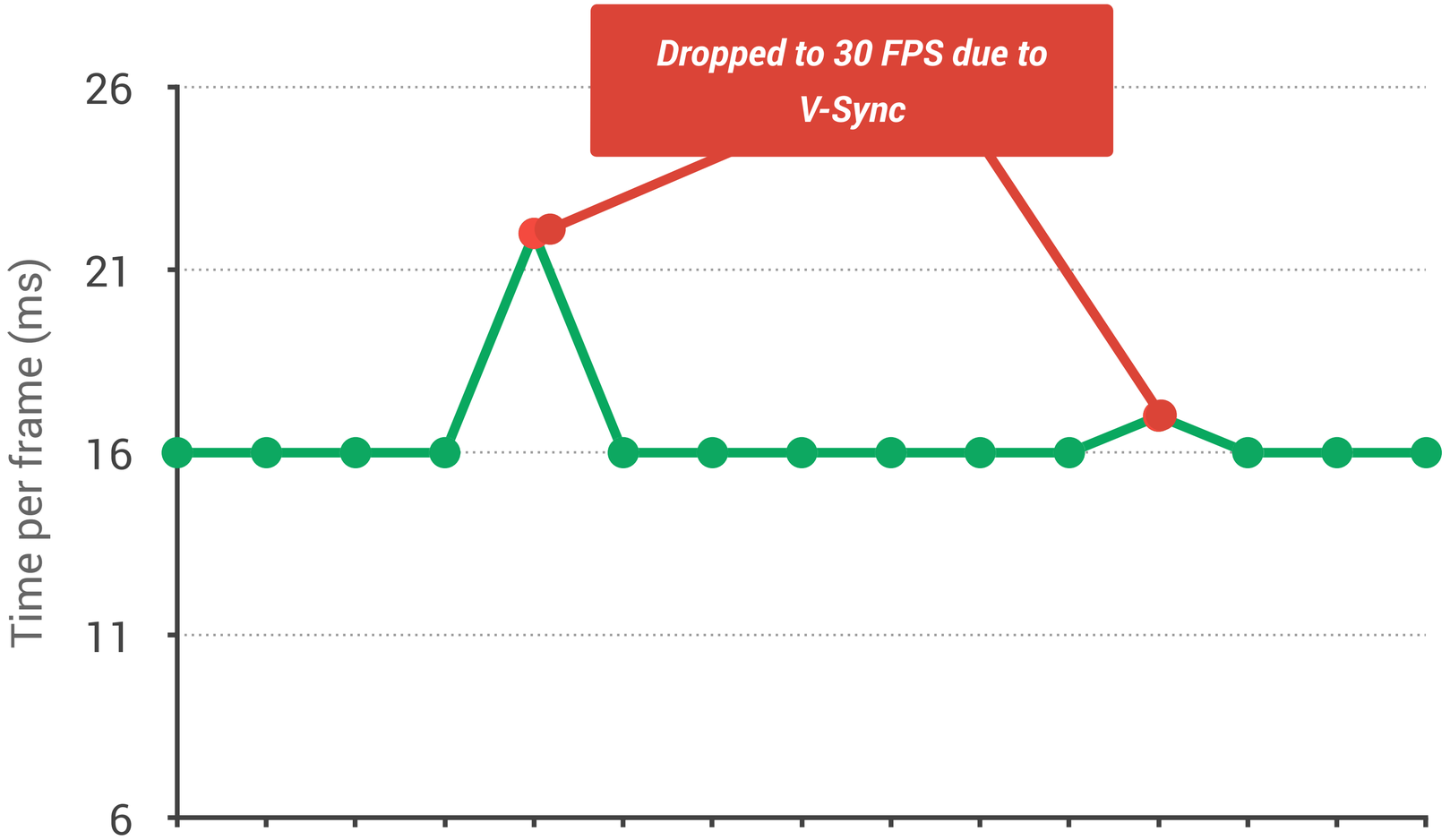


Chart data source info here

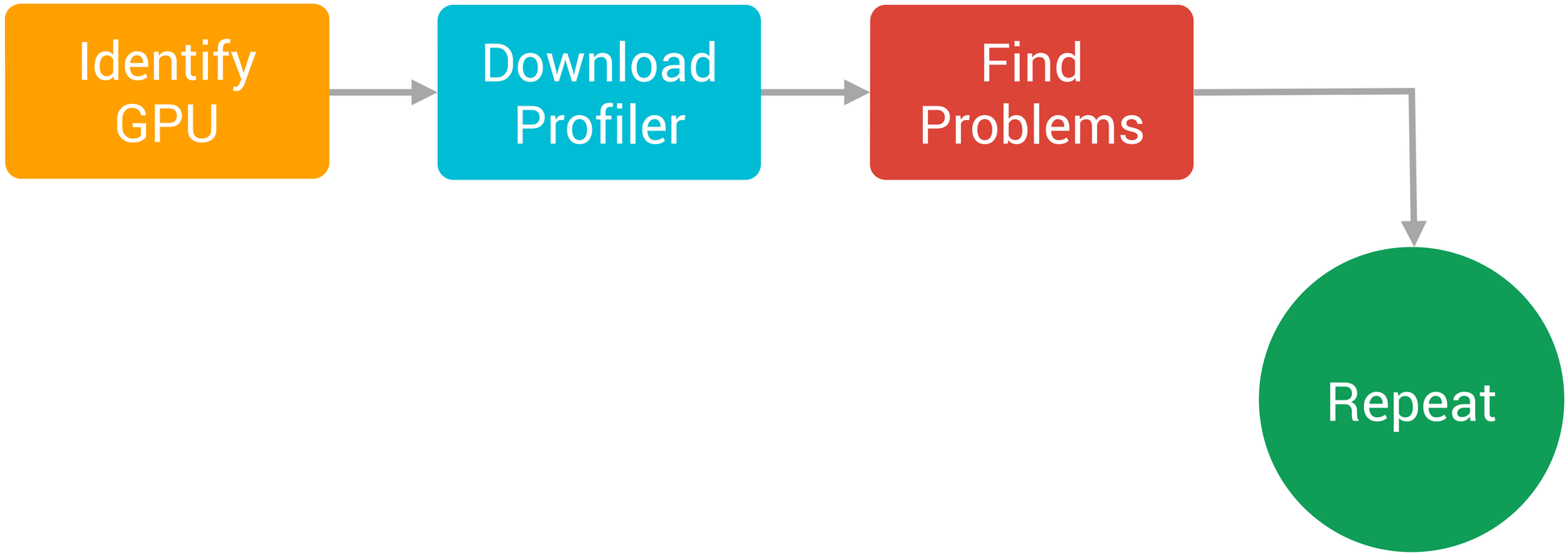
Performance. Why should I care?

Smooth Game Experience

More Features

Fast Loading Times

Rule number #1 of optimizing…



Random good news

Random good news

What is Vulkan

What is Vulkan

- Low overhead graphics & compute API.
- Explicit control over command execution on the GPU.

  “Not a closed box”.
- Multi-threading friendly, allows better CPU usage.
- SPIR-V: Shaders represented in intermediate state,

  therefore fast compile & private.
- Vulkan is Coming to Android! *(And so is OpenGL ES 3.2)*

Back to topic...

OpenGL [ES] Optimizations

Some basics…

Enable Face Culling

Generate Mip Maps

Only for 3D games!
or 2d with scaled sprites

Reduce OpenGL Redundancy

Z Pre-pass / Sorted Draw

Not needed on tile renderer
such as PowerVR and Mali

Disable Alpha Blend

Interleaved Vertices

Adaptive
Scalable
Texture
Compression

DOWNLOADING
ADDITIONAL ASSETS

200MB / 2.7GB

DOWNLOADING
ADDITIONAL ASSETS

200MB / 2.7GB

I HOPE YOU ARE
ON WI-FI

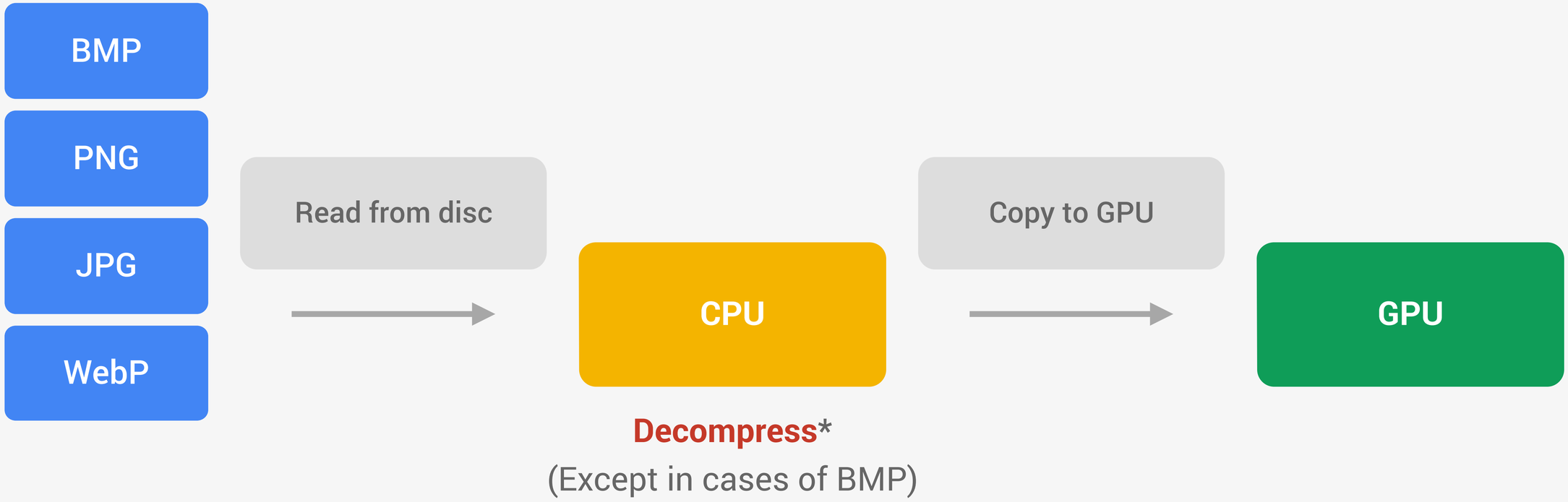
LOADING...

PLEASE WAIT

LOADING...

PLEASE WAIT

NOW WHAT?

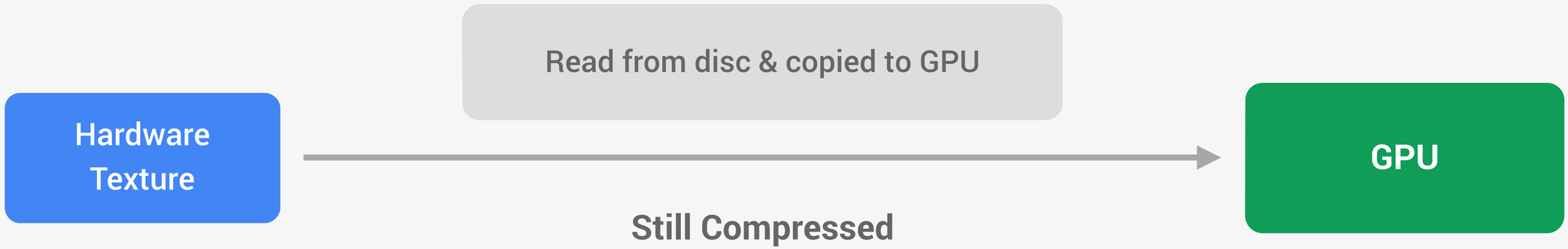

BMP
PNG
JPG
WebP
Read from disc
CPU
Copy to GPU
GPU
Decompress*
(Except in cases of BMP)

Decompression is Expensive!

Use hardware accelerated textures…

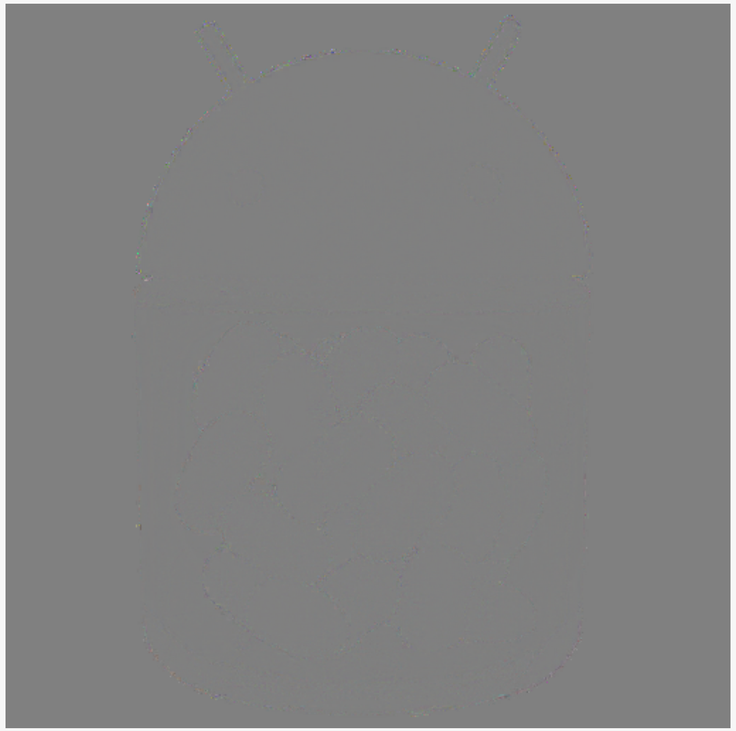
***ASTC** is **awesome**!*
But requires Extension / GLES 3.2…

DXT | S3TC | PVRTC | ASTC

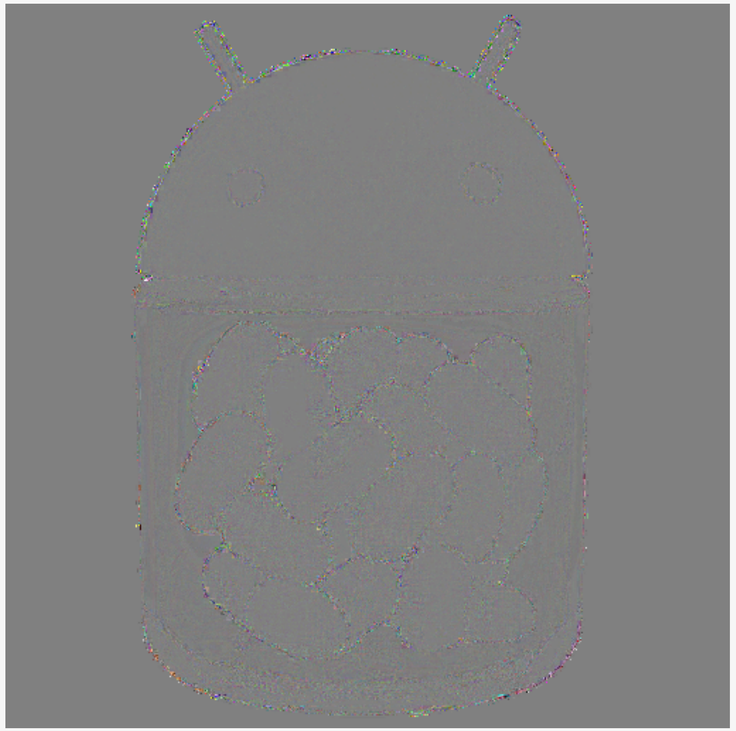


Original
Bitmap: 1.1MB
PNG: 299KB

4x4 ASTC
Size: 262KB



Difference Map



Difference Map
5x Enhanced

Original:

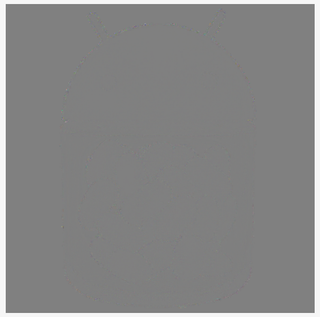
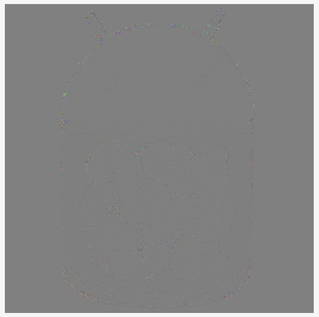
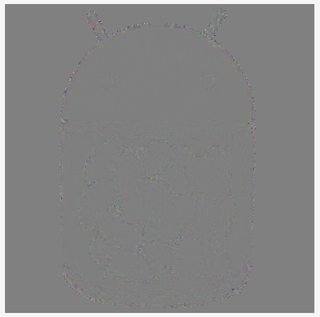
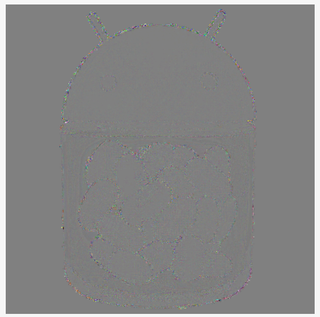
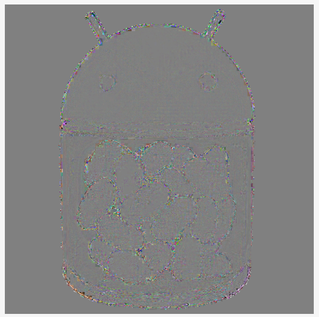
Resolution: 512 x 512

Format: RGBA

Size on disk:

Bitmap: 1.1MB

PNG: 299KB

	4x4 (8 bitrate) 262KB	6x6 (3.56 bit) 119KB	8x8 (2 bitrate) 70KB
Difference Map			
5x Enhanced			

Mali Texture Compression Tool:

http://goo.gl/2Rhnyl

Optimizing Shader Compilation

(Skip if not using many shaders…Profile!)

Shader Compilation

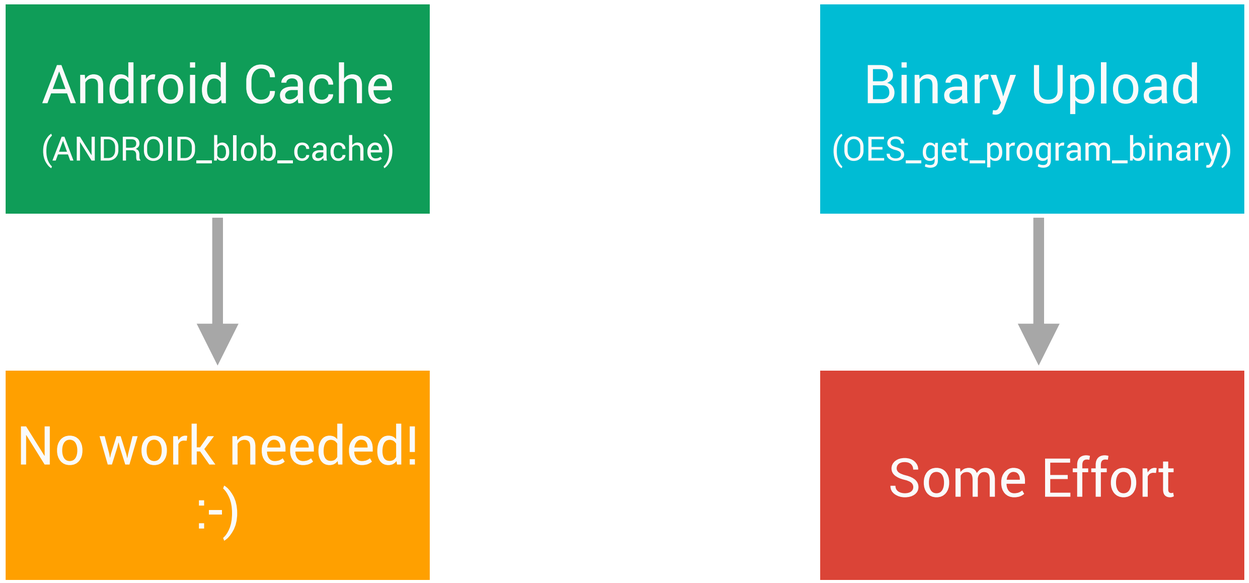
```
GLuint vertexShader = glCreateShader( GL_VERTEX_SHADER );
glShaderSource( vertexShader, 1, &vertexSource, 0 );
glCompileShader( vertexShader );

GLuint fragmentShader = glCreateShader( GL_FRAGMENT_SHADER );
glShaderSource( fragmentShader, 1, &fragmentSource, 0 );
glCompileShader( fragmentShader );

GLuint program = glCreateProgram();
glAttachShader( program, vertexShader );
glAttachShader( program, fragmentShader );
glLinkProgram( program );

glDetachShader( program, vertexShader );
glDetachShader( program, fragmentShader );
```

Shader Compilation

```
GLuint vertexShader = glCreateShader( GL_VERTEX_SHADER );
glShaderSource( vertexShader, 1, &vertexSource, 0 );
glCompileShader( vertexShader );

GLuint fragmentShader = glCreateShader( GL_FRAGMENT_SHADER );
glShaderSource( fragmentShader, 1, &fragmentSource, 0 );
glCompileShader( fragmentShader );

GLuint program = glCreateProgram();
glAttachShader( program, vertexShader );
glAttachShader( program, fragmentShader );
glLinkProgram( program );

glDetachShader( program, vertexShader );
glDetachShader( program, fragmentShader );
```

Slow. Increased Loading Times.

Shader Compilation: Fixed with…



Not using many shader? Cache might be enough… Profile!

Shader Compilation: Binary Upload

```
GLuint program = glCreateProgram();
// ... Shader compiled as normal

// Retrieve shader length
GLint binaryLength;
glGetProgramiv( program, GL_PROGRAM_BINARY_LENGTH,
                                                &binaryLength );

// Create shader binary
GLenum binaryFormat;
void* binary = (void*)malloc( binaryLength );
glGetProgramBinary( program, binaryLength, NULL,
                                        &binaryFormat, binary );

// Save binary & binaryFormat to file…
```

Compile as usual

Get program binary length
for creating a sufficient buffer

Retrieve program binary
and save it to disk

Shader Compilation: Binary Upload

```
// Program variables
GLint       binaryLength;
GLenum      binaryFormat;
void*       binary;

// Read binary & format from file

GLuint program = glCreateProgram();
glProgramBinary( progObj, binaryFormat, binary,
                                        binaryLength );

free(binary);

// Error checking! :D
Glint success;
glGetProgramiv( progObj, GL_LINK_STATUS, &success );
```

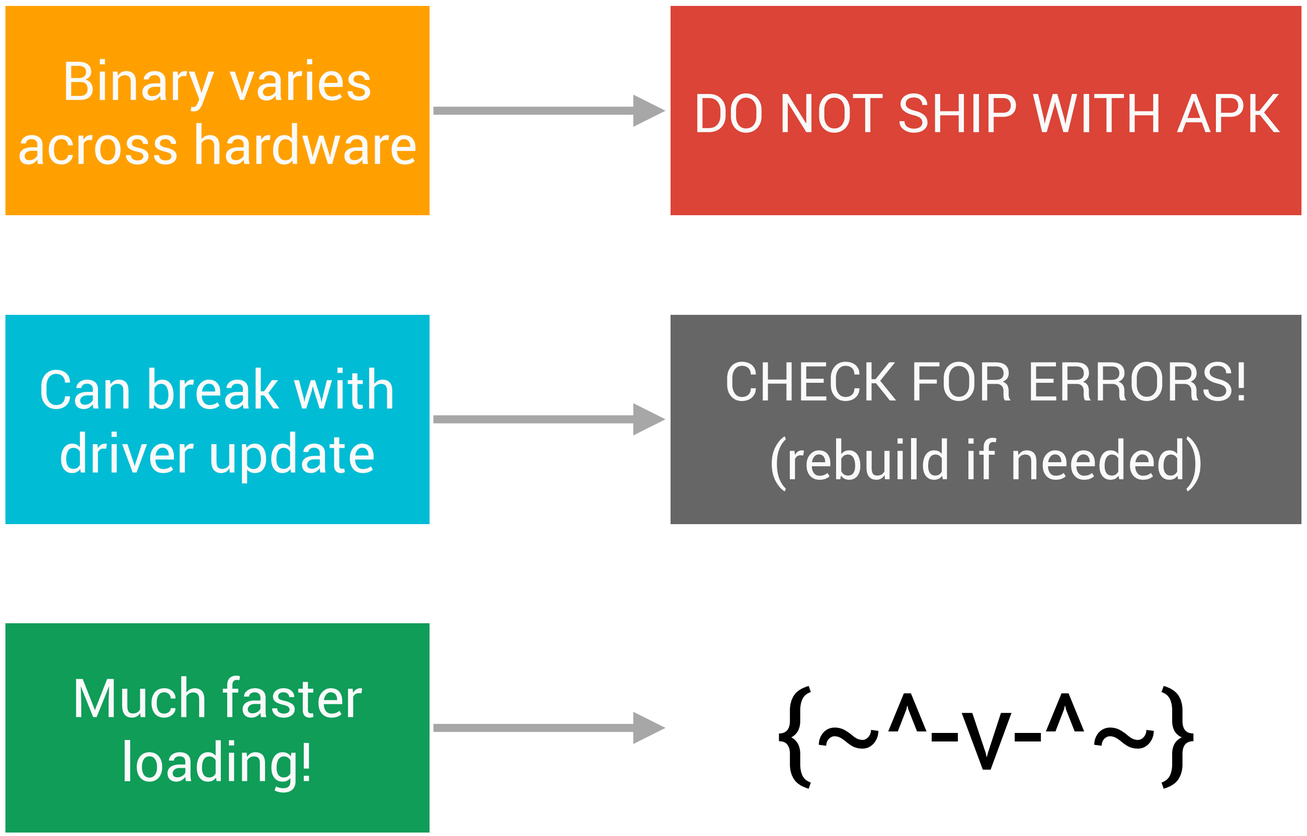
Read binary from file

fill binary buffer and format

Create Shader Program

and set it using the binary

Shader Binary caveats…



Reducing Driver Overhead

Reducing driver overhead…

Vulkan

DirectX 12

Metal API

Reducing driver overhead...

Vulkan

DirectX 12

Metal API

What can we do now?

Step One

Easy improvements

Example 1: Layout Qualifiers

```
GLuint shaderProgram;
GLuint positionHandle;

positionHandle = glGetAttribLocation( shaderProgram,
                                      “u_vPosition”);

glVertexAttribPointer( positionHandle, 3, GL_FLOAT,
                                     GL_FALSE, 0, 0 );

glEnableVertexAttribArray( positionHandle );

// ...
```

Example 1: Layout Qualifiers

```
GLuint shaderProgram;
GLuint positionHandle;

positionHandle = glGetAttribLocation( shaderProgram,
                                      “u_vPosition”);

glVertexAttribPointer( positionHandle, 3, GL_FLOAT,
                                     GL_FALSE, 0, 0 );

glEnableVertexAttribArray( positionHandle );

// ...
```

Slow, non-deterministic
(can use glBindAttribLocation)

Example 2: Setting Uniform

Example 2: Setting Uniform

```
GLuint shaderProgram;
float matWorldViewProjection[16];


// Get the uniform handle
GLint handleWVP = glGetUniformLocation( shaderProgram,
                                        “g_matWorldViewProj” );

// Assign our uniform
glUniformMatrix4fv( handleWVP, 1, false,
                                    matWorldViewProjection );
```

Example 2: Setting Uniform

```
GLuint shaderProgram;
float matWorldViewProjection[16];
```

```
// Get the uniform handle
GLint handleWVP = glGetUniformLocation( shaderProgram,
                                        “g_matWorldViewProj” );
```

```
// Assign our uniform
glUniformMatrix4fv( handleWVP, 1, false,
                                matWorldViewProjection );
```

Slow, non deterministic. Again.

Uniforms & Attributes

```
uniform mat4 g_matWorldViewProj;
uniform vec4 g_vecCameraPos;

in vec4 u_vPosition;
in vec2 u_vTexCoords;

// Insert awesome vertex shader here
```

Unknown uniform location

Unknown attribute location

Uniforms & Attributes

```
uniform mat4 g_matWorldViewProj;
uniform vec4 g_vecCameraPos;

in vec4 u_vPosition;
in vec2 u_vTexCoords;

// Insert awesome vertex shader here
```

Explicit Locations!

```
layout(location = 0) uniform mat4 g_matWorldViewProj;
layout(location = 1) uniform vec4 g_vecCameraPos;

layout(location = 0) in vec4 u_vPosition;
layout(location = 1) in vec2 u_vTexCoords;

void main() {
```

Uniforms & Attributes

```
uniform mat4 g_matWorldViewProj;
uniform vec4 g_vecCameraPos;

in vec4 u_vPosition;
in vec2 u_vTexCoords;

// Insert awesome vertex shad
```

More performant, deterministic

Explicit Locations!

```
layout(location = 0) uniform mat4 g_matWorldViewProj;
layout(location = 1) uniform vec4 g_vecCameraPos;

layout(location = 0) in vec4 u_vPosition;
layout(location = 1) in vec2 u_vTexCoords;

void main() {
```

Example 3: Binding for Draw

Example 3: Binding for Draw

```
// draw
glBindBuffer( GL_ARRAY_BUFFER, vertex_buffer_object );

glEnableVertexAttribArray( 0 );
glVertexAttribPointer( 0, 3, GL_FLOAT, GL_FALSE, 32, 0 );

glEnableVertexAttribArray( 1 );
glVertexAttribPointer( 1, 2, GL_FLOAT, GL_FALSE, 32, 12 );

glEnableVertexAttribArray( 2 );
glVertexAttribPointer( 2, 3, GL_FLOAT, GL_FALSE, 32, 20 );

// Draw elements
glDrawElements( GL_TRIANGLES, count,
                                GL_UNSIGNED_SHORT, 0 );
```

Example 3: Binding for Draw

```
// draw
glBindBuffer( GL_ARRAY_BUFFER, vertex_buffer_object );

glEnableVertexAttribArray( 0 );
glVertexAttribPointer( 0, 3, GL_FLOAT, GL_FALSE, 0, 0 );

glEnableVertexAttribArray( 1 );
glVertexAttribPointer( 1, 3, GL_FLOAT, GL_FALSE, 32, 0 );

glEnableVertexAttribArray( 2 );
glVertexAttribPointer( 2, 3, GL_FLOAT, GL_FALSE, 32, 20 );

// Draw elements
glDrawElements( GL_TRIANGLES, count,
                                GL_UNSIGNED_SHORT, 0 );
```

And just plain ugly…

Slow, in-effecient

Example 3: Vertex Array Object (VAO)

```
glBindBuffer( GL_ARRAY_BUFFER, vertex_buffer_object );

// Create VAO
GLuint vao;
glGenVertexArrays( 1, &vao );
glBindVertexArray( vao );

// Set the VAO State
glEnableVertexAttribArray( 0 );
glVertexAttribPointer( 0, 3, GL_FLOAT, GL_FALSE, 32, 0 );

glEnableVertexAttribArray( 1 );
glVertexAttribPointer( 1, 2, GL_FLOAT, GL_FALSE, 32, 12 );

glEnableVertexAttribArray( 2 );
glVertexAttribPointer( 2, 3, GL_FLOAT, GL_FALSE, 32, 20 );
```

Bind vertex buffer object
for correct vertex attirbs

Generate Vertex Array Object
and set the vertex attributes

Example 3: Vertex Array Object (VAO)

```
{
// ...
// on draw

// Bind Vertex Array Object
glBindVertexArray( vao );

// Draw elements
glDrawElements( GL_TRIANGLES, count,
                                GL_UNSIGNED_SHORT, 0 );

}
```

Bind Vertex Array Object
vertex attirbs are saved!

Profit!

Vertex Array Object (VAO): Caveats

John Carmack
@ID_AA_Carmack

Following

The fact that GL ES Vertex Array Objects are not shared between contexts just really made me miserable.

RETWEETS 21 FAVORITES 44

8:18 PM - 17 Jun 2015

Step Two

Rendering all the things

Mesh

Scene

Draw Call

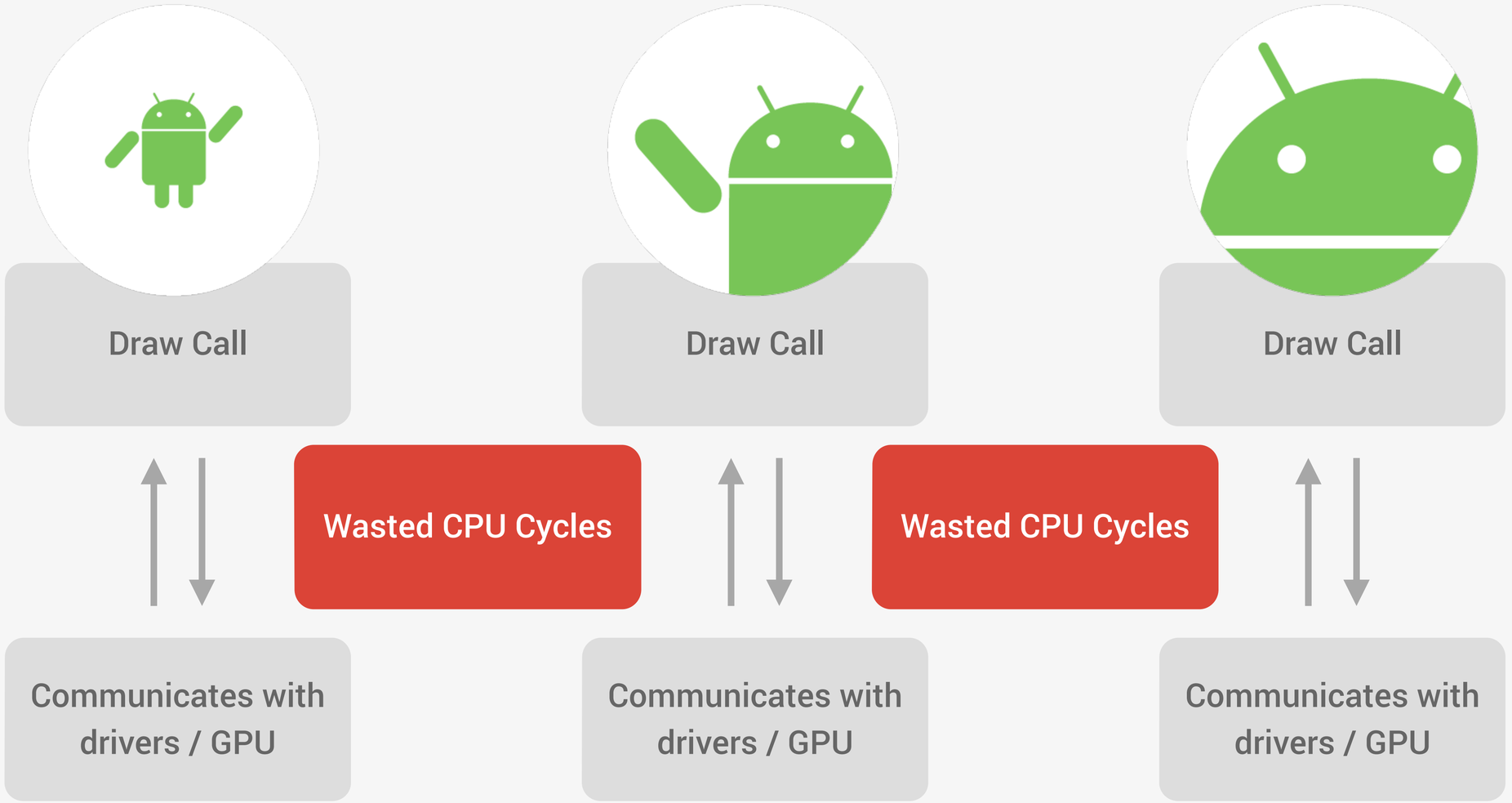
Communicates with drivers / GPU

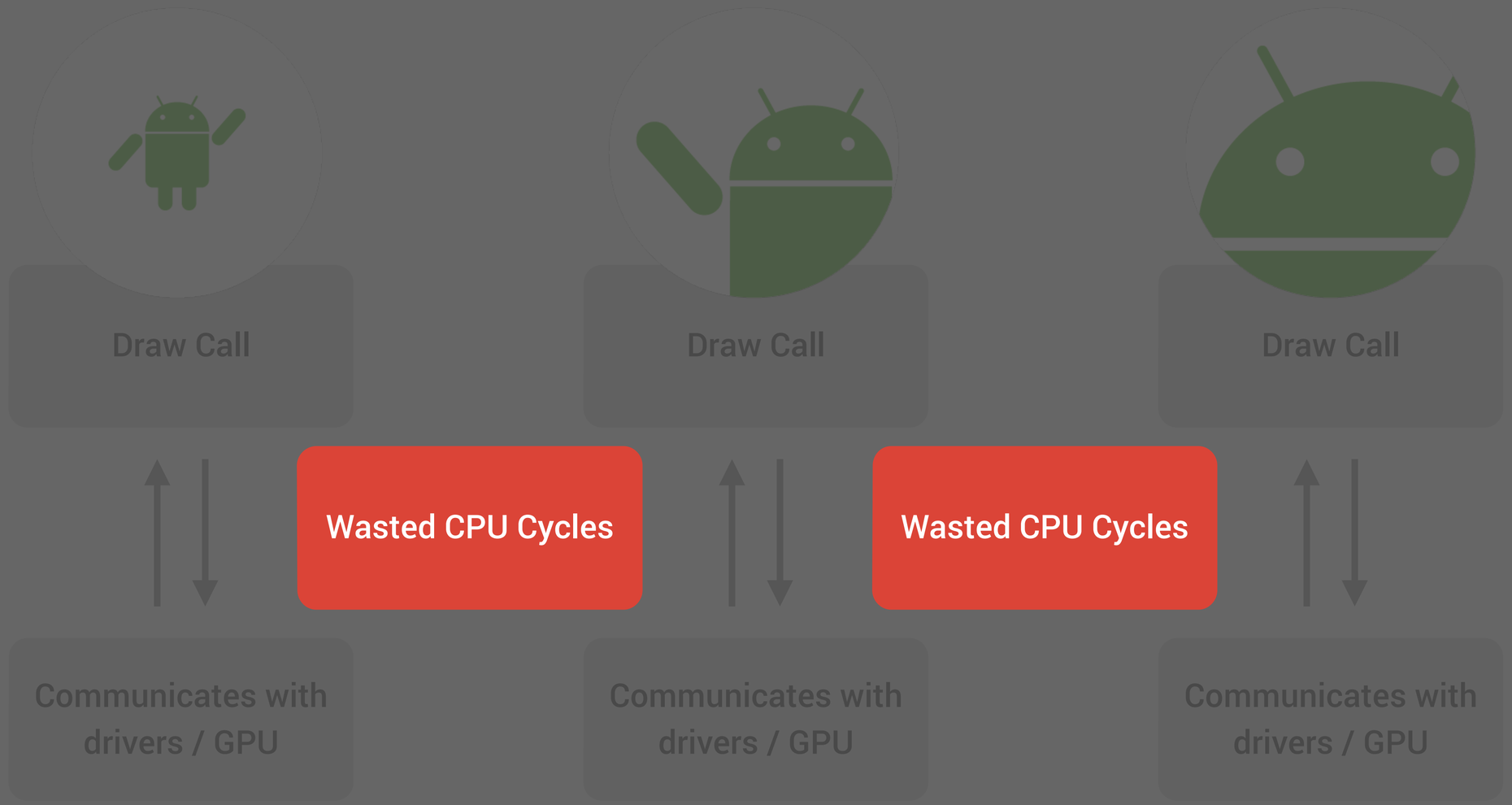
Draw Call

Communicates with drivers / GPU

Draw Call

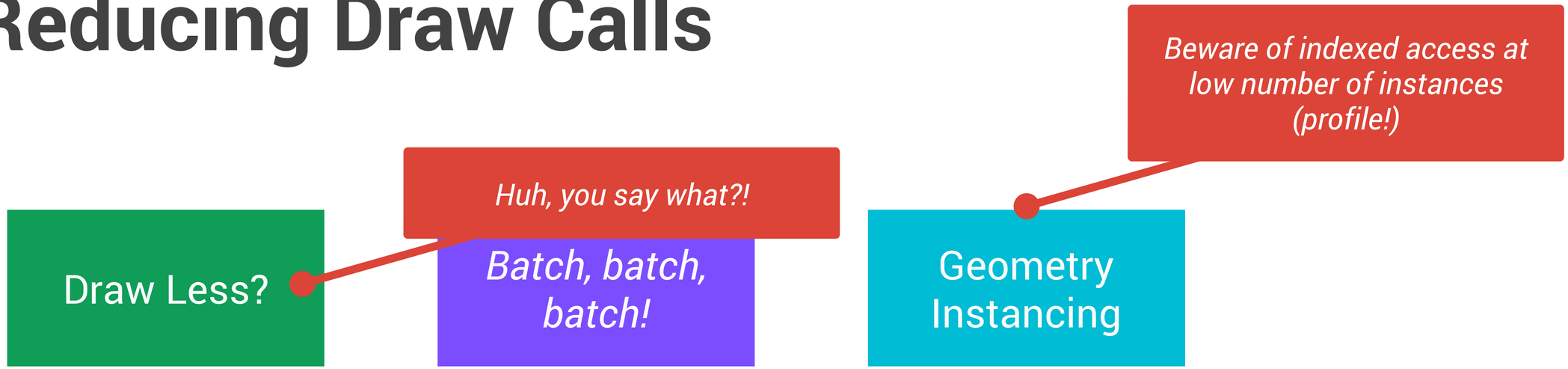
Communicates with drivers / GPU


Draw Call
Draw Call
Draw Call
Wasted CPU Cycles
Wasted CPU Cycles
Communicates with drivers / GPU
Communicates with drivers / GPU
Communicates with drivers / GPU


Draw Call
Draw Call
Draw Call
Wasted CPU Cycles
Wasted CPU Cycles
Communicates with drivers / GPU
Communicates with drivers / GPU
Communicates with drivers / GPU

Must Reduce Draw calls

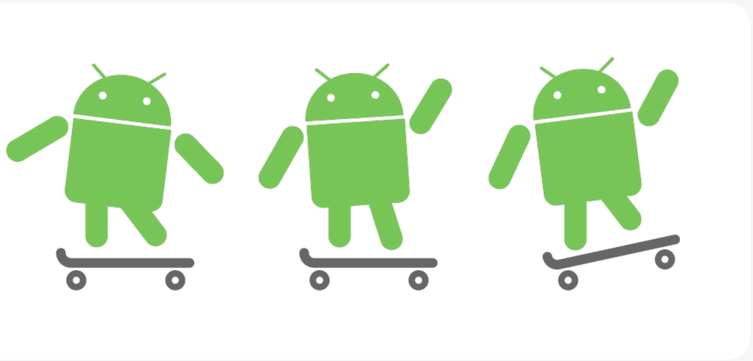
Reducing Draw Calls



Geometry Instancing:

Geometry Data (VBO1)	Position	UV	Normal
Instance Data (VBO2 / UBO / SSBO)	WorldViewProj Color	WorldViewProj Color	WorldViewProj Color

Fill Instance Buffer → Bind buffers → gl Draw Elements Instanced →



Get the sample code

NDK 64-bit target…/samples/MoreTeapots/

Geometry Instancing

```
{ // pre-draw
  for ( int i = 0; i < num_instances; ++i )
  {
    buffer[i].matWorldViewProj = transforms[i] * matViewProj;
  }
}

{ // draw
  glBindBuffer( GL_UNIFORM_BUFFER, ubo );
  glBindVertexArray( vao );

  glDrawElementsInstanced( GL_TRIANGLES, count,
                           GL_UNSIGNED_SHORT, num_instances );
}
```

Copy instance data to buffer

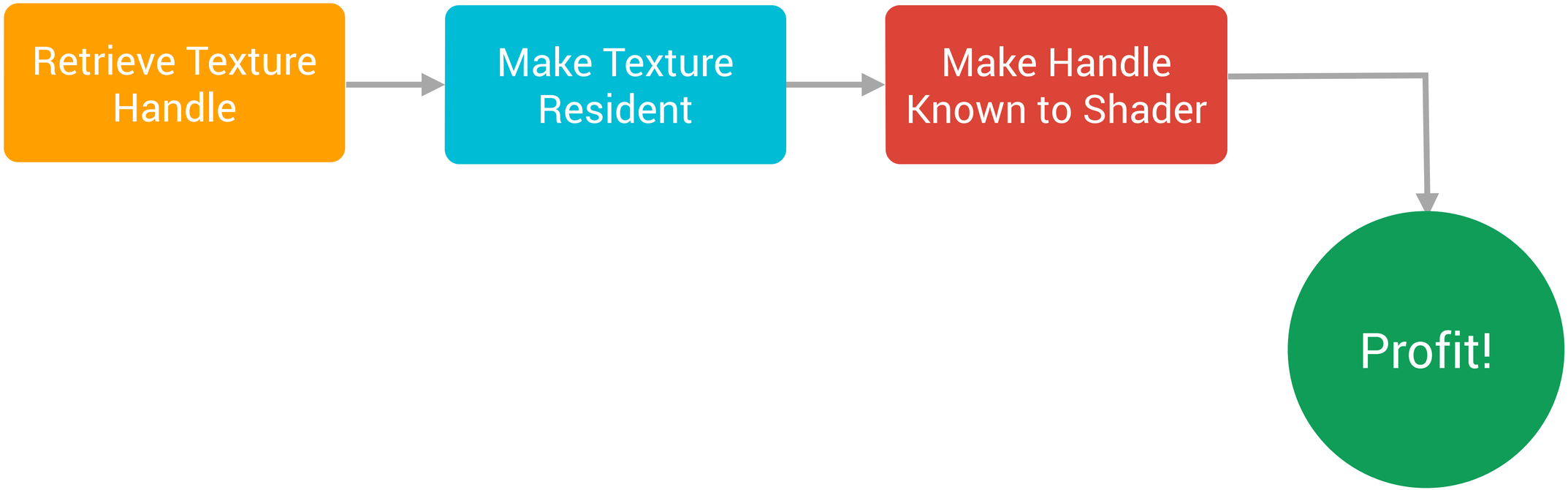
Bind objects and draw
VAO and UBO instance data

But what about multiple meshes?

- glMultiDrawElements — *Not available in OpenGL ES*
- glMultiDrawElementsEXT
- glMultiDrawElementsIndirect

Ok, but what about textures?

- Texture Atlas (not ideal…)
- Texture Arrays
- Bindless Textures



Bindless Textures

Init:

```
GLuint* m_TextureIds[NUM_TEXTURES]; // the textures

// On init after creating the textures
GLuint64EXT* m_TextureHandles = new GLuint64[ NUM_TEXTURES ];

for( i = 0; i < NUM_TEXTURES; ++i )
{
  m_TextureHandles[i] = glGetTextureHandleNV( m_TextureIds[i] );
  glMakeTextureHandleResidentNV( m_TextureHandles[i] );
}
```

Make index known
via instance data buffer

Bind:

```
// On Bind
GLuint handleSamplersLocation( shader->getUniformLocation("samplers") );
glUniform1ui64vNV( handleSamplersLocation, NUM_TEXTURES, m_TextureHandles );
```

Step Three

Love Your Cache

Method #1

```
char data[1000000];
unsigned int sum = 0;

for (int i=0; i<1000000; ++i)
{
  sum += data[i*1];
}
```

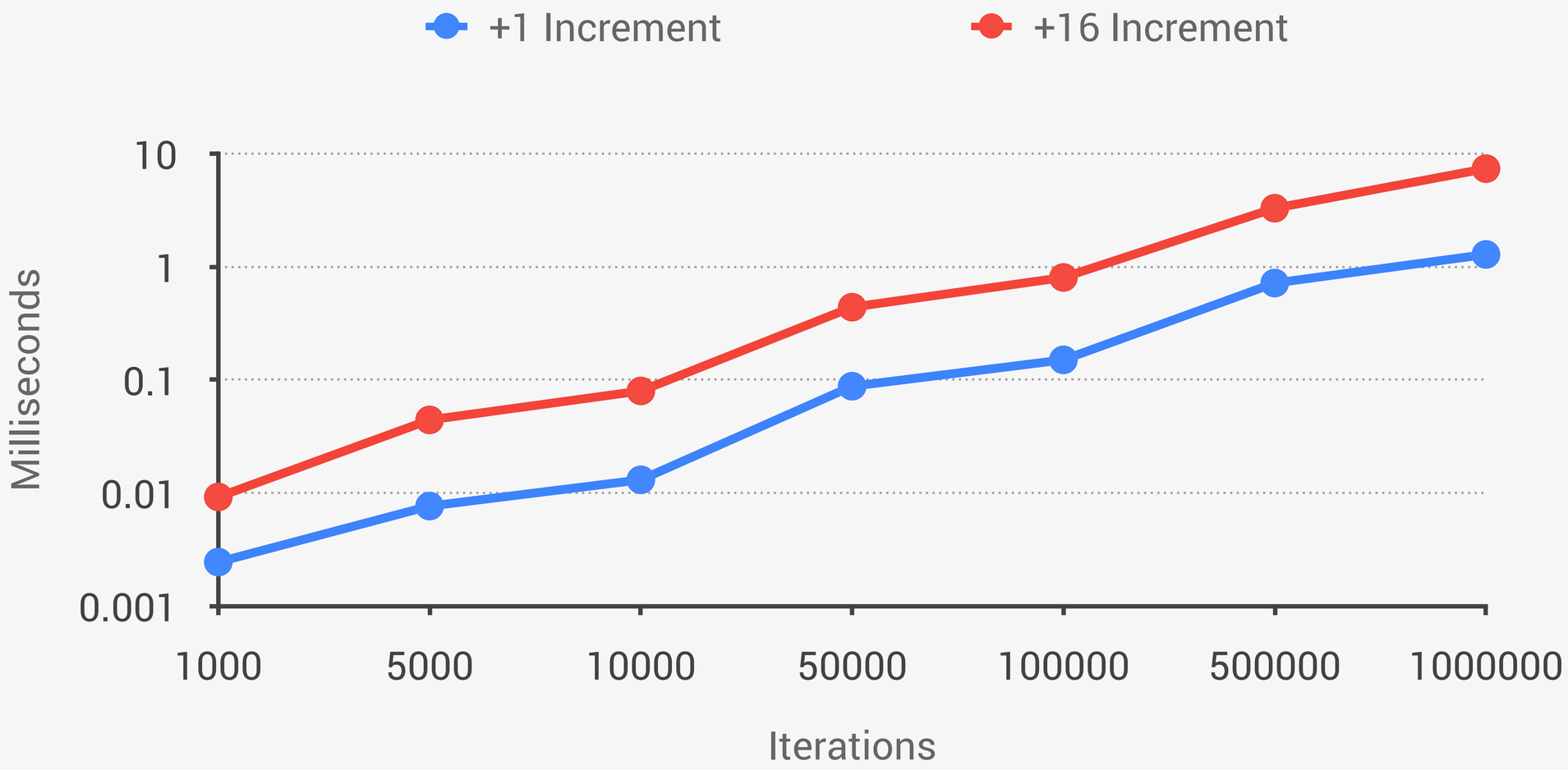
Method #2

```
char data[16000000];
unsigned int sum = 0;

for (int i=0; i<1000000; ++i)
{
  sum += data[i*16];
}
```

#DataOrientedDesign

Nexus 6

Average 5x slower!



#DataOrientedDesign

Example: CPU cache size 16 bytes

```
int i = 0; i += 1;
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Cache Miss																*Cache Miss*

```
int i = 0; i += 8;
```

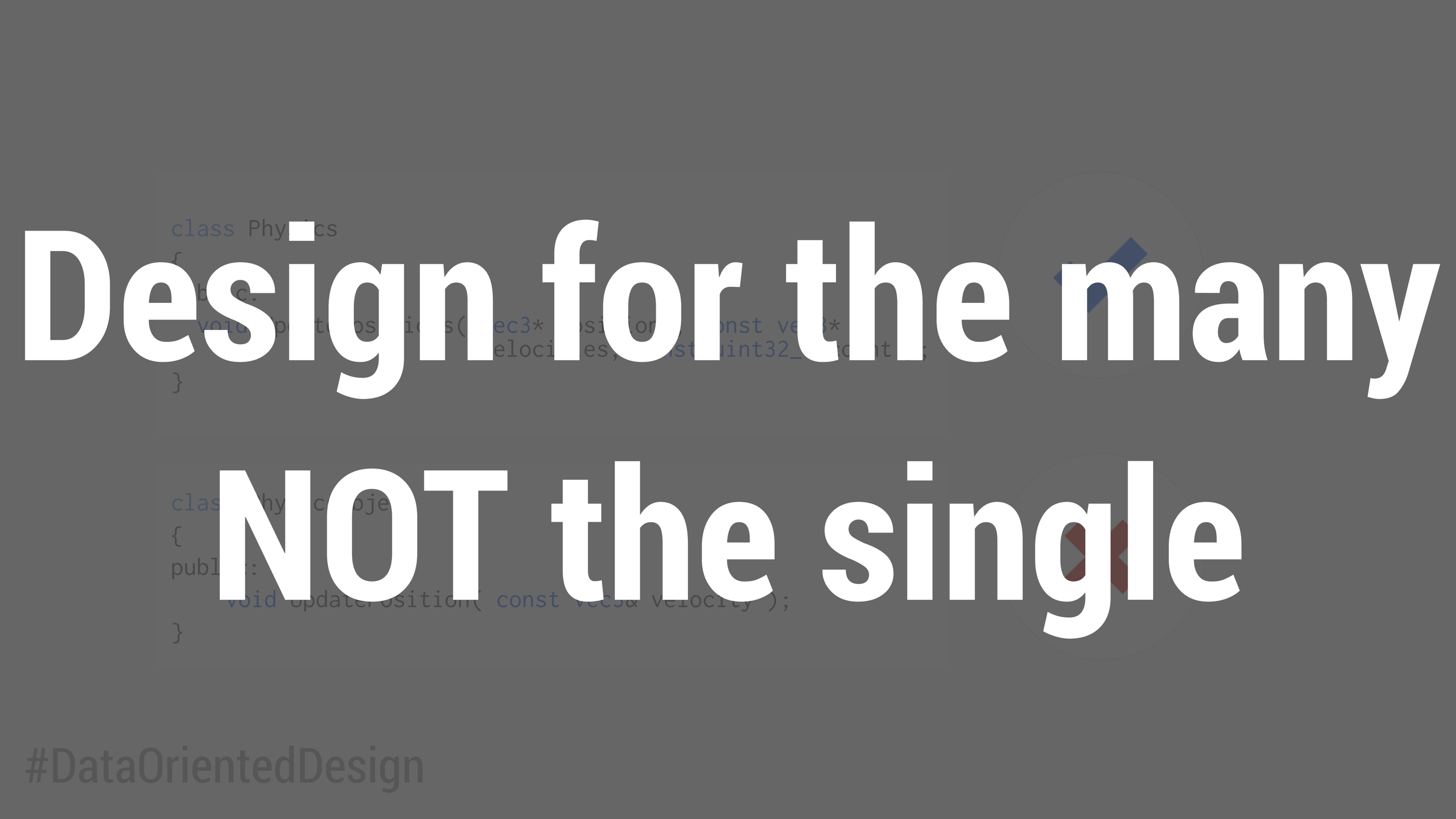
0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128
Cache Miss		*Cache Miss*		*Cache Miss*		*Cache Miss*		*Cache Miss*		*Cache Miss*		*Cache Miss*		*Cache Miss*		*Cache Miss*

#DataOrientedDesign

```
class Physics
{
public:
  void UpdatePositions( vec3* positions, const vec3*
                        velocities, const uint32_t count );
}
```

```
class PhysicsObject
{
public:
    void UpdatePosition( const vec3& velocity );
}
```

#DataOrientedDesign


class Physics
{
public:
void UpdatePhysics(vec3* positions, const vec3*
velocities, const uint32_t count);
}
class PhysicsObject
{
public:
void UpdatePosition(const vec3& velocity);
}
Design for the many
NOT the single
#DataOrientedDesign

Data Oriented Design

Premature Optimization?

Data Oriented Design

~~Premature Optimization~~?

Data Oriented Design

Data Oriented Design: the RIGHT way to code

~~Premature Optimization~~?

How is this related to OpenGL?

Geometry Instancing

Populating Uniforms

Reducing OpenGL Redundancy

Compute Shaders

Faster Iteration

Cleaner, strict code

CPU Bonus:

SIMD

Vectorization

Thank you!

+Shanee Nishry
@Lunarsong

#GameDev #OpenGL