From the Dark Side of the Moon

GPU Programming with BGFX and Eclipse

Tony McCrary, l33t labs

What is BGFX?

- Open source C/C++ GPU rendering library
- Developed by Branimir Karadzic
- Provides a way to render 2D/3D graphics
- BGFX executes rendering commands on OpenGL, Direct3D, Apple Metal

What is BGFX?

- BGFX provides a low level style API
- BGFX is NOT a scene graph, based around sorted draw calls

What is Twilight BGFX?

- Manual Java binding to BGFX
- Also includes a binding to NanoVG, a 2D vector graphics library
- Simple windowing library based on SDL (Simple Direct media Layer)
- SWT integration - Render 2D/3D graphics into SWT controls
- Eclipse integration - Shader editor and project builder

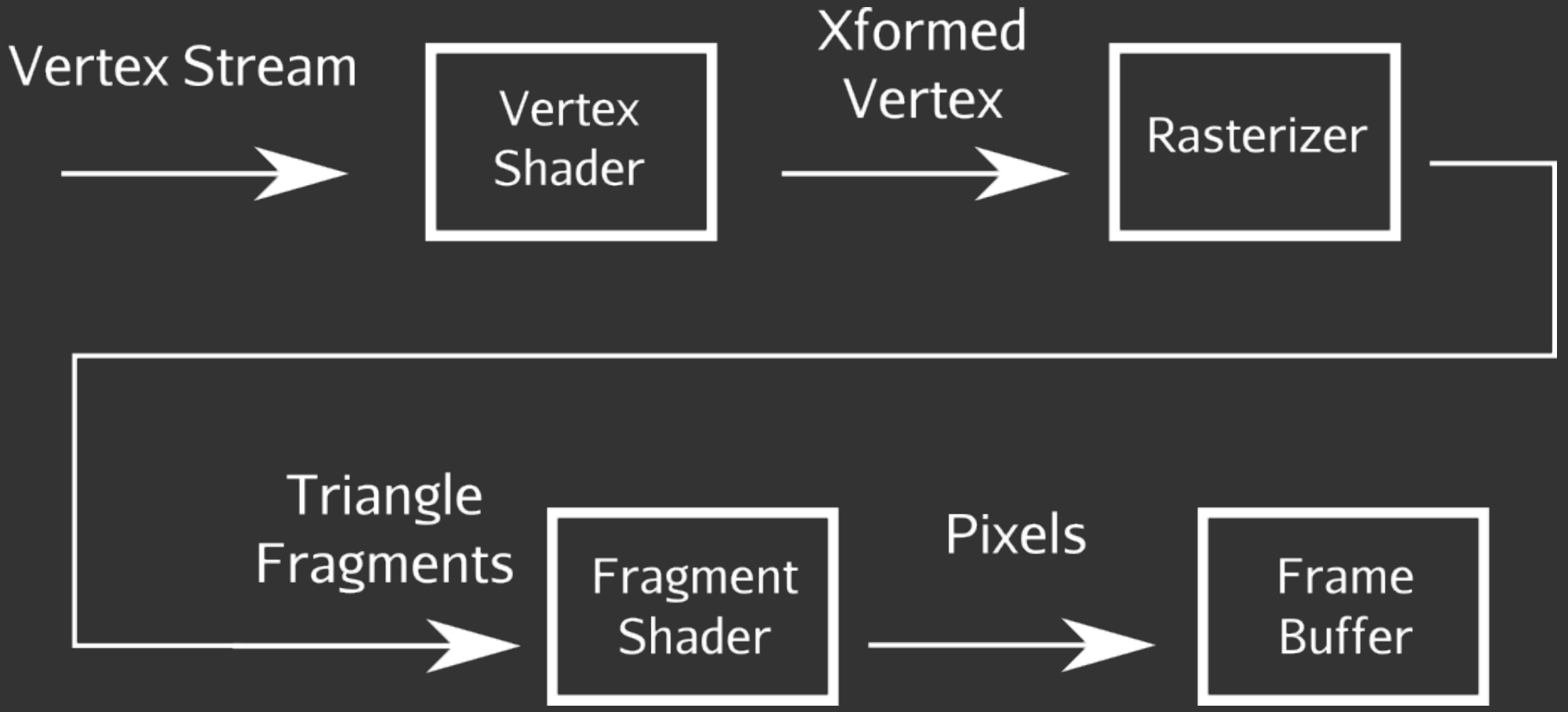

SWT
SDL
Java
Twilight BGFX
Java Binding
BGFX
OpenGL
Direct3D
Metal

What is a GPU?

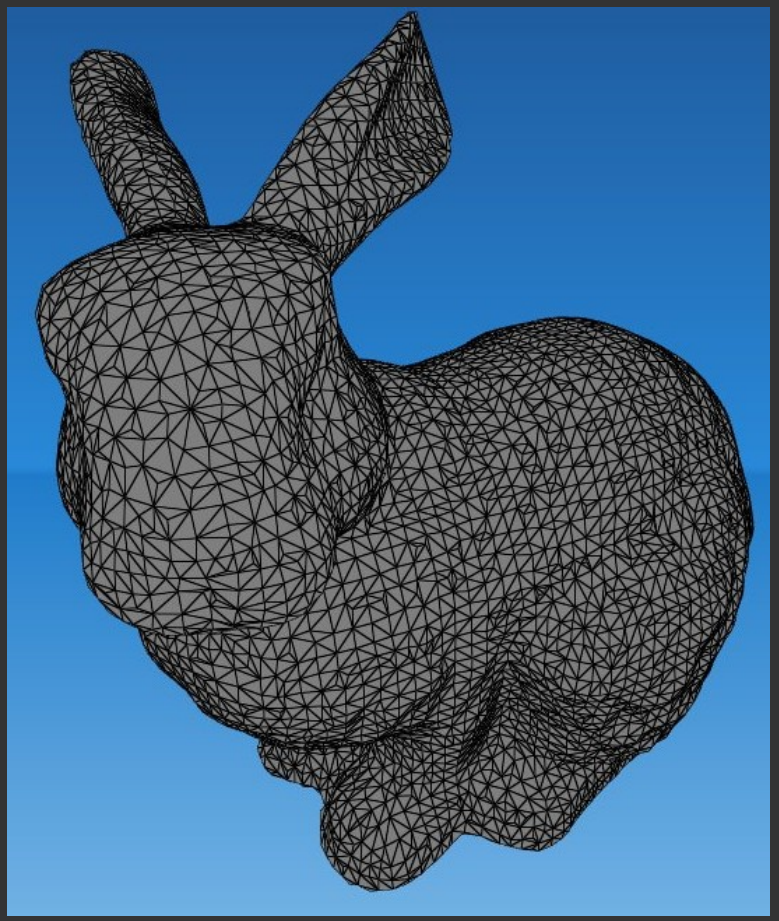
- GPU stands for Graphics Programming Unit
- Originally designed to accelerate graphics for video games and scientific/engineering purposes
- Specialized hardware designed for high floating point performance
- Extremely parallel execution model

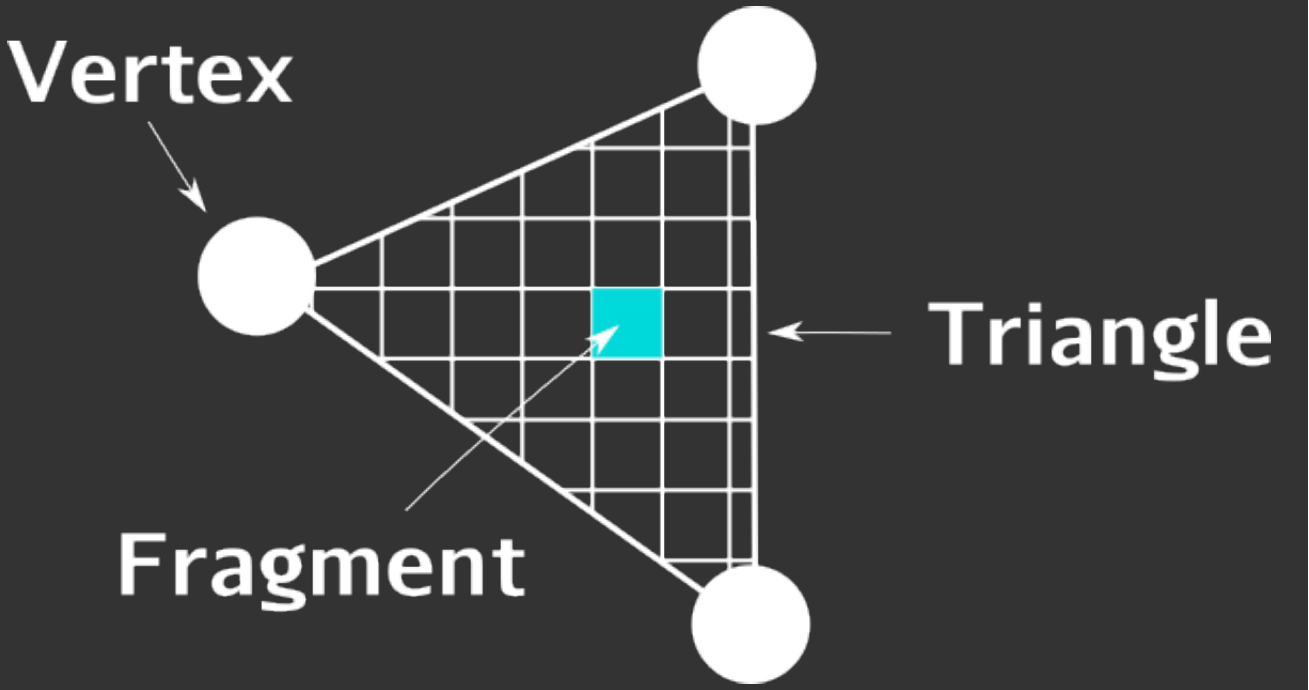
GPU graphics pipeline

- Based around rasterizing triangles
- Triangles are made of up a series of vertices
- Vertices are transformed from 3D world space and projected into screen space
- Once projected onto the 2D framebuffer, the fragments contained within the triangle are shaded


Vertex Stream
Vertex Shader
Xformed Vertex
Rasterizer
Triangle Fragments
Fragment Shader
Pixels
Frame Buffer

GPU graphics pipeline





CPU vs GPU

High end Intel i7 clock speed

3.60 Ghz

High end Nvidia Titan clock speed

836 Mhz

CPU vs GPU

- High end Intel i7 cores

  6 cores (12 threads)

- High end Nvidia GPU cores

  2688 cores (stream processors)

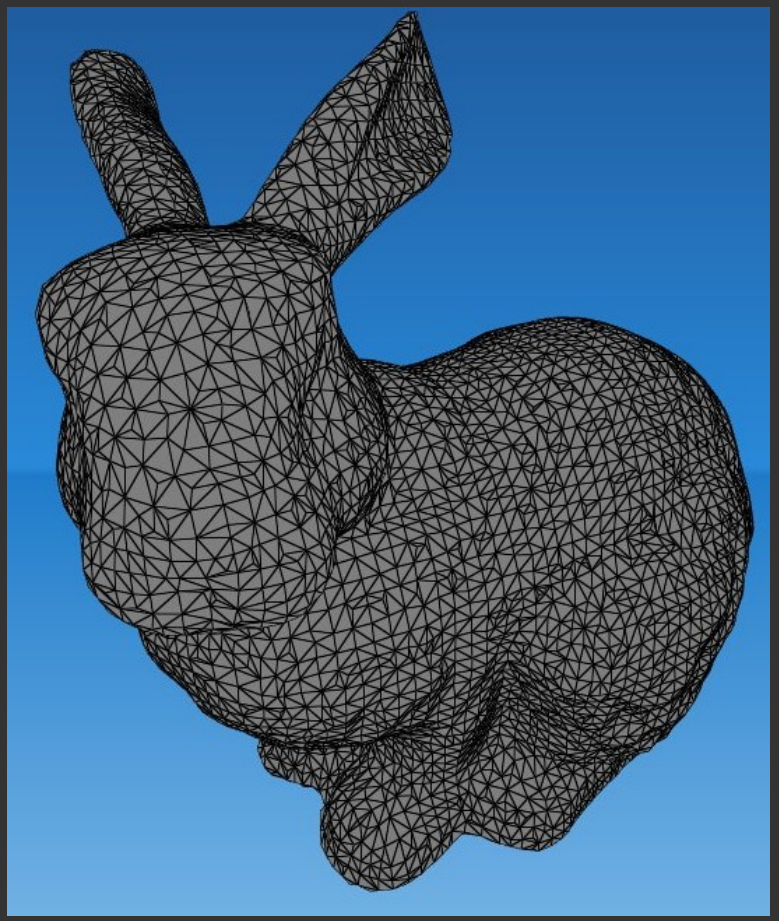
OpenGL? Direct3D? Metal? Who Cares?

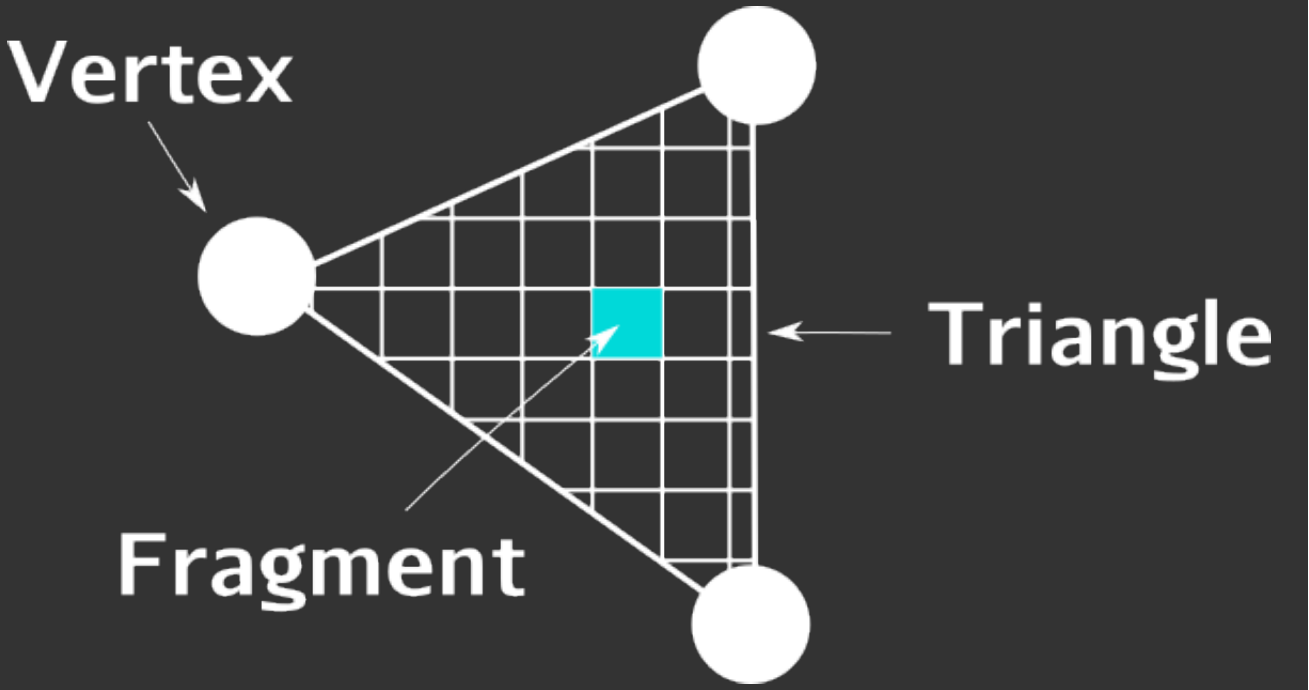
- Things are not what they seem with OpenGL
- Direct3D is Windows only
- Metal is OS X only
- Madness: They all do the same things, in basically the same way, on the same hardware but are totally incompatible!
- Most developers don't really care which API is used, as long as their triangles show up!

Desktop Platforms

- Linux (OpenGL), OSX (Metal), Windows (Direct3D)
- Nvidia (Nouveau, Proprietary), AMD (Open source, Proprietary), Intel (Gallium)
- Each generation of hw from the same vendor can be vastly different
- Even drivers for the same hardware are different from OS to OS

GPU graphics pipeline





Write once, Run on any GPU

- Java platform is an idealized computer that allows programmers to focus on solving problems
- Platform issues, quirks and oddities are encapsulated by the underlying runtime
- BGFX provides a similar idealized interface to the GPU

BGFX Supported Platforms

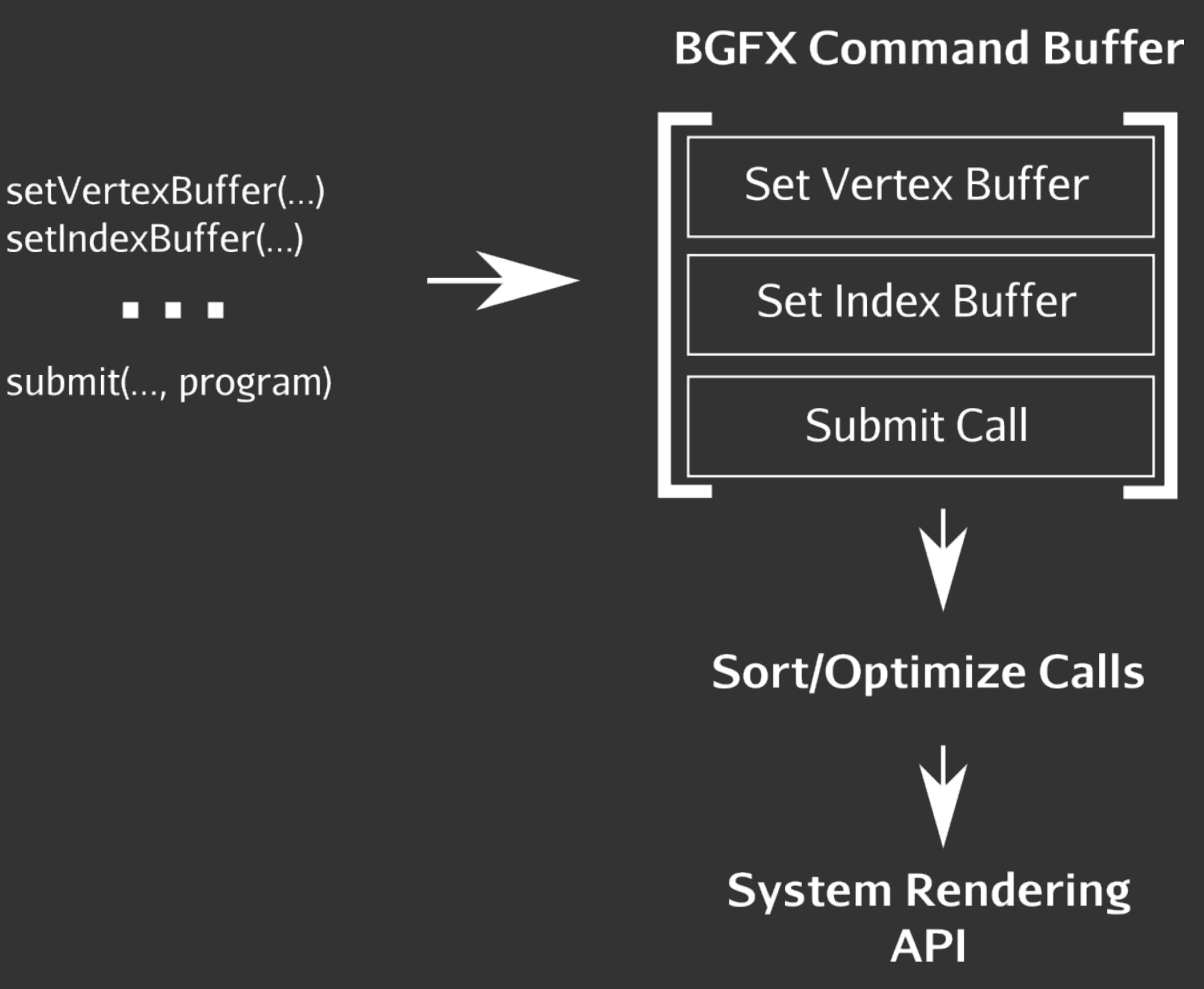
- Direct3D 9
- Direct3D 11
- Direct3D 12 (WIP)
- Metal (WIP)
- OpenGL 2.1
- OpenGL 3.1+
- OpenGL ES 2
- OpenGL ES 3.1
- WebGL 1.0

- Android (14+, ARM, x86, MIPS)
- asm.js/Emscripten (1.25.0)
- FreeBSD
- iOS (iPhone, iPad, AppleTV)
- Linux
- MIPS Creator CI20
- Native Client (PPAPI 37+, ARM, x86, x64, PNaCl)
- OSX (10.9+)
- RaspberryPi
- SteamLink
- Windows (XP, Vista, 7, 8, 10)
- WinRT (WinPhone 8.0+)

```
bgfx.init();

bgfx.reset(width, height, reset);

while(rendering) {
    bgfx.setViewRect(0, 0, 0, width, height);

    bgfx.touch(0);

    bgfx.setVertexBuffer(myVertexBuffer);
    bgfx.setIndexBuffer(myIndexBuffer);

    bgfx.submit(0, myShaderProgram);

    bgfx.frame();
}

bgfx.shutdown();
```

BGFX Execution Model

- Render calls are encoded as data into a command buffer
- Multi-threaded or single threaded rendering
- Multi-threaded rendering allows you to generate calls for a frame while the last frame is rendering


BGFX Command Buffer
setVertexBuffer(...)
setIndexBuffer(...)
. . .
submit(..., program)
Set Vertex Buffer
Set Index Buffer
Submit Call
Sort/Optimize Calls
System Rendering API

BGFX Execution Model

- When a frame is finished being built, the calls are sorted and optimized according to state
- Calls are sorted according to state changes, drastically improving performance
- After call optimization, a platform specific renderer is executed
- Command optimization is roughly analagous to HotSpot's JIT optimization

Buffers, Buffers, and... Buffers

- Triangles are defined primarily by two kinds of buffers: Vertex and Index
- Think of Vertex and Index buffers are specialized types of data arrays (not java arrays)
- Buffers are generally created by CPU code and then used by shader programs to draw graphics
- Textures are a specialized kind of buffer that can be sampled at various 2D/3D locations

Buffer Updates

- Variants of each type of buffer exist depending on how you use them
- Static buffers are never updated, you define them once and they are immutable
- Dynamic buffers are used for buffers that are updated regularly (once a frame or more)
- Transient buffers are for buffers that are updated every frame

Vertex Buffers

- Define a set of vertices, which are a set of "Vertex Attributes"
- Attributes typically include position in local space, vertex color, texture coordinates and more
- Vertex Attributes are user definable are used by shaders during rendering
- Example vertex [5.0, 5.0, 5.0, 1.0, 0.0, 0.0, 1.0, 0.5, 0.5]

  | pos | color | texcoord |
- The above vertex has 3 attributes: 3 float position (XYZ), 4 float color (RGBA), 2 texture coodinates (UV)
- BGFX defines how a vertex is structured via VertexDecl

Index Buffers

- Define which vertices belong to a given rendering primitive (3 vertices -> Triangle)
- Usually defined as 16-bit or 32-bit integers, literally as offsets into an associated vertex buffer
- Example index buffer [0, 1, 2]
- Defines a trangles whose constituent vertices are at location 0, 1, 2 in the vertex buffer
- Note that these indices are not byte or data type offsets but for the entire vertex

Textures

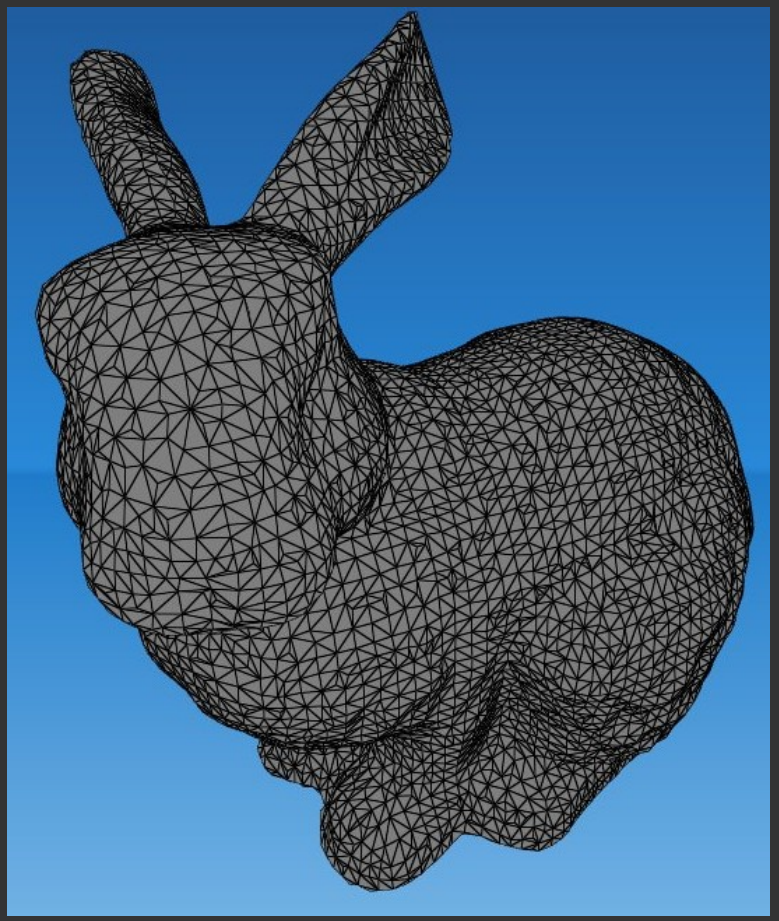
- Contain raster data, similar to image formats
- Data is accessed in shaders via a sampler
- Can be updated dynamically and used for various purposes (colors, normals, etc.)

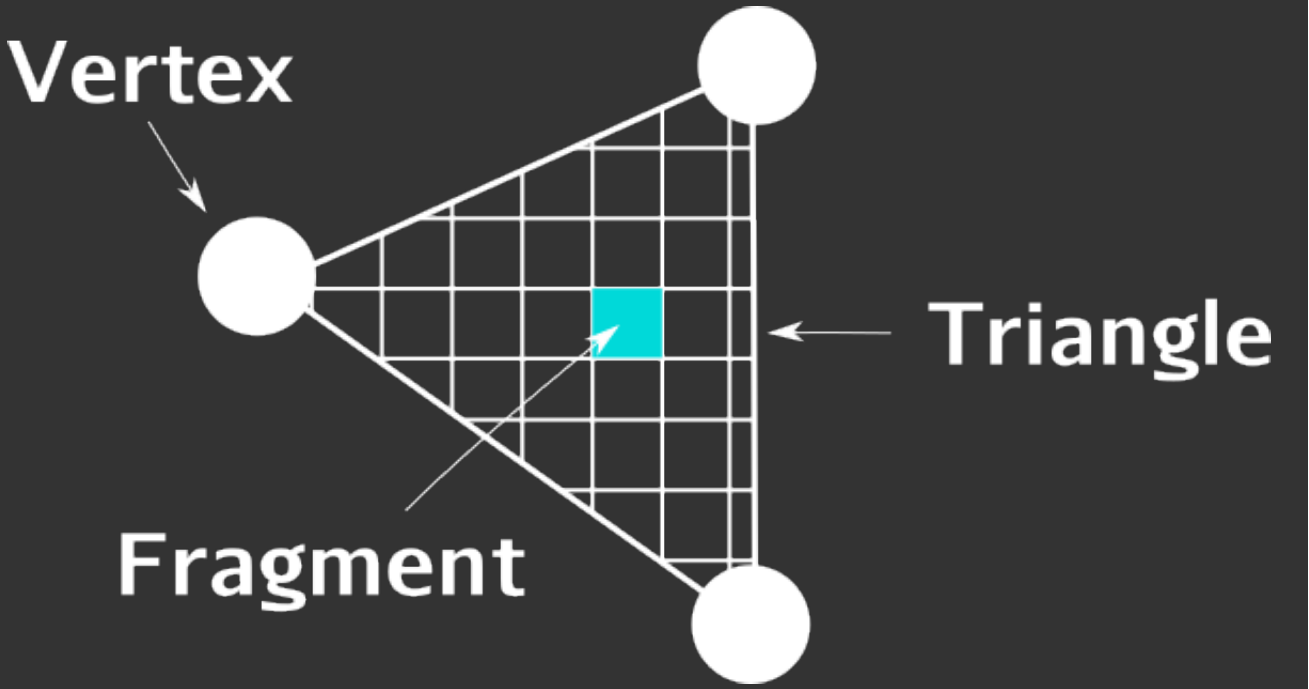
What are Shader Programs?

- Small procedural programs that execute on a GPU's numerous processing cores
- Each Shader Program is comprised of individual Shaders
- Each Shader is linked into the Shader Program and execute at various stages (Vertex, Fragment, etc.)
- Contain code for each step in the rendering pipeline

BGFX Shading Language

- Implements in a procedural C style similar to most shading languages
- Compiled with included shaderc shader compiler
- Language compiles down to plaform specific shaders for distribution
- Includes preprocesor for advanced compilation




Vertex
Triangle
Fragment

Vertex Shaders

- Transforms a vertex from one space to another using matrices
- Attributes are the various values that comprise a given vertex (position, color, normal(direction), texture coordinates, etc.
- Passes Varying values to the fragment shaders (interpolation)

Vertex Shaders

- Doesn't actually do any shading (!?)
- Execute by taking processing one vertex at a time
- Uniforms are like fields in a Java class, they are read when the shader executes

```
$input a_position, a_color0, a_texcoord0
$output v_color0, v_texcoord0

#include "common.sh"

void main()
{
	vec4 pos = vec4(a_position, 1.0);
	gl_Position = mul(u_modelViewProj, pos);
	v_color0 = a_color0;
	v_texcoord0 = a_texcoord0;
}
```

Fragment Shaders

- Runs after the vertex shader has transformed the primitive
- Fragment shaders actually shade
- A Fragment is similar in concept to a pixel in a framebuffer
- Texture color can be sampled and written into a fragment

```
$input v_color0, v_texcoord0

SAMPLER2D(u_texColor, 0);

void main()
{
gl_FragColor = texture2D(u_texColor, v_texcoord0);
}
```

Compute Shaders

- Perform general purpose computation on the GPU
- GPU floating performance is often much higher than possible on even a fast CPU

2D Graphics? 3D Graphics? Whats the difference?

- 2D is contained in 3D space, with BGFX you can do both
- 2D is merely a different kind of projection, orthographic instead of perspective

Ultra fast 2D graphics with NanoVG

- NanoVG is a technology originally developed by Mikko Mononen
- Originally an OpenGL 2D vector graphics API but has since been ported to BGFX
- Provides 2D rendering functions like drawing lines, rectangles, text and other drawing primitives
- Similar in principle to SWT's GC operations, Swing's Graphics2D

BGFX Eclipse Integration

- SWT Integration (BGFXCanvas)
- BGFX Project Nature, Builder, Editor

Rendering into SWT and Eclipse with BGFXCanvas

- BGFXCanvas is a standard SWT widget, similar in principle to Canvas but for GPU graphics
- Provides callback functions for submitting and rendering calls
- Provides full access to BGFX and NanoVG APIs

Writing BGFX Shaders within Eclipse

- Uses an incremental project builder to build BGFX shaders on the fly
- Project builder uses shaderc to generate platform specific shaders
- Shader editor provides syntax highlighting

Demos

Future

- More platform support for Java bindings (mobile, web, etc)
- Tesselation and Geometry Shaders
- JavaFX “BGFXNode”
- Zero Overhead JNI Bindings
- Get legal/IP into shape for use with professional open source projects

JavaFX “BGFXNode”

- JavaFX does not expose shader functionality to end users, with no support planned
- JavaFX uses Direct3D on Windows, OpenGL on Linux and Mac, so you can't choose one to integrate
- BGFX could provide a way to create complex and high performance 2D/3D graphics within JavaFX
- BGFXNode would be a way to execute a BGFX renderer on top of your platform's JavaFX Prism implementation

Zero Overhead JNI Bindings

- BGFX's command buffer allows for (near) zero overhead JNI interop
- Generate command data buffer entirely on the Java side into a direct ByteBuffer
- Once a frame is finished, pass the pointer over to the waiting BGFX renderer for execution

Software Licenses

- BGFX, Twilight BGFX bindings are BSD licensed
- SDL, NanoVG are zlib
- SWT integration and Eclipse plugins are licensed under the EPL

Links

- Twilight BGFX Repo
  - https://github.com/enleeten/twilight-bgfx
- BGFX C/C++ Repo
  - https://github.com/bkaradzic/bgfx

Please Evaluate My Talk

Sign-in: www.eclipsecon.org

Select session from schedule