# WebGL and WebCL
## 3D Graphics and Compute on the Web

**Neil Trevett**
**VP Mobile Content, NVIDIA**
**President, Khronos Group**

# Two WebGL Sessions Today

## Industry ecosystem and standards overview

- **Web and Mobile Ecosystem, Khronos and WebGL Overview**
  - Neil Trevett – VP Mobile Content at NVIDIA, President of Khronos
- **WebCL Overview**
  - Tasneem Brutch, Sr. Staff Engineer, Samsung Electronics, WebCL Chair

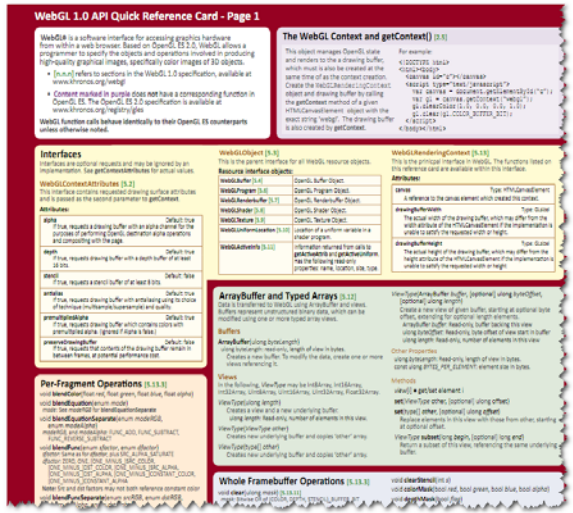
## Hands-On with WebGL

- **WebGL technical overview**
  - Ken Russell - Software Engineer, Chrome GPU team, WebGL Chair
- **CubicVR overview, tutorial and demos**
  - Bobby Richter, Creative Tech Lead, Web Made Movies, Mozilla Foundation

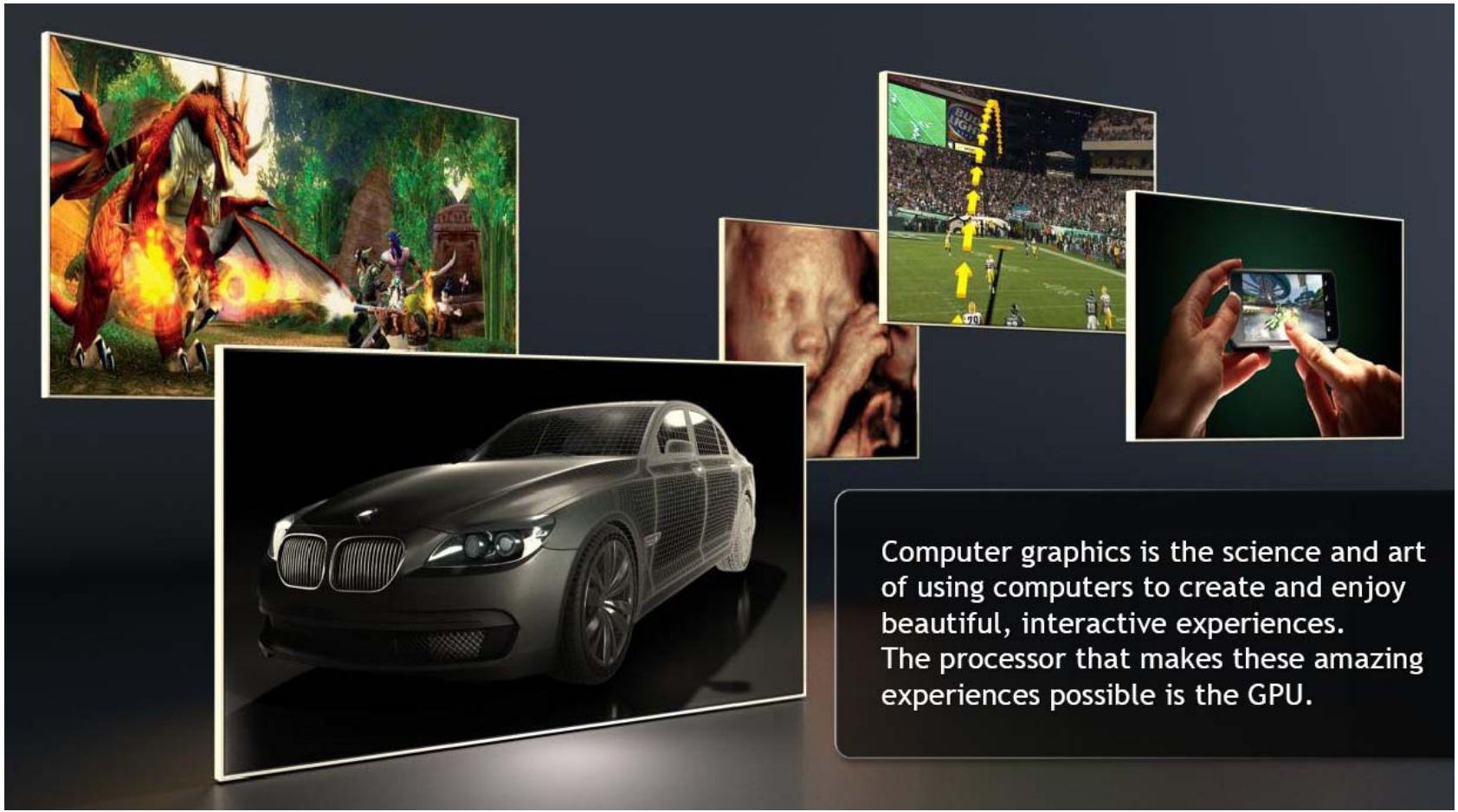
# Topics for this Session

- What is 3D anyway?
- Khronos and open standard acceleration APIs
- The evolution of pervasive 3D on mobile platforms
- WebGL and HTML5
- 3D use cases beyond games – augmented reality
- Looking ahead to what's next

**WebGL Reference Cards at end of session!**

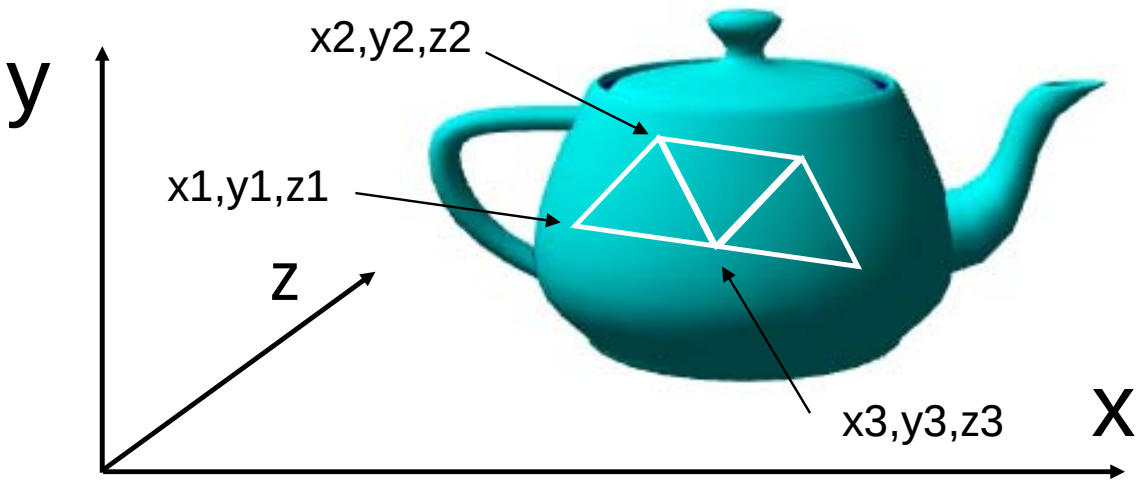


# What is Real-time 3D Graphics?



# 3D Pipeline Basics

- The art of “faking” a realistic looking scene or objects using heuristic techniques learned over the years
- Surfaces of objects are broken down into a grid of polygons
- The vertices of the polygons are located in 3D coordinate space - x,y,z
- Each vertex has a “material” – color and reflective properties
- The objects making up a scene are held in a database



# Fundamental 3D Processing Stages

**Operations on Vertices**

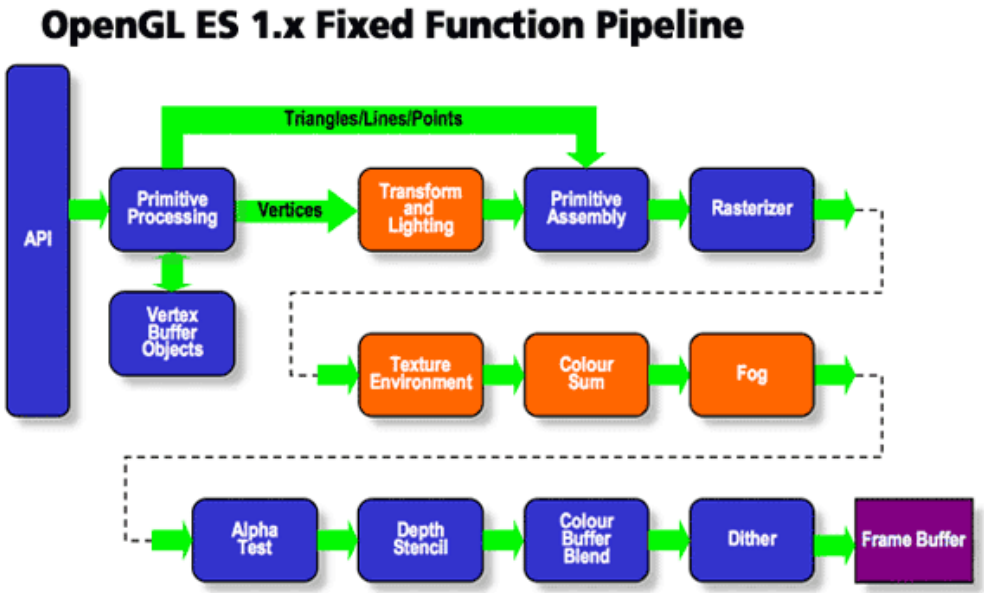
Geometry

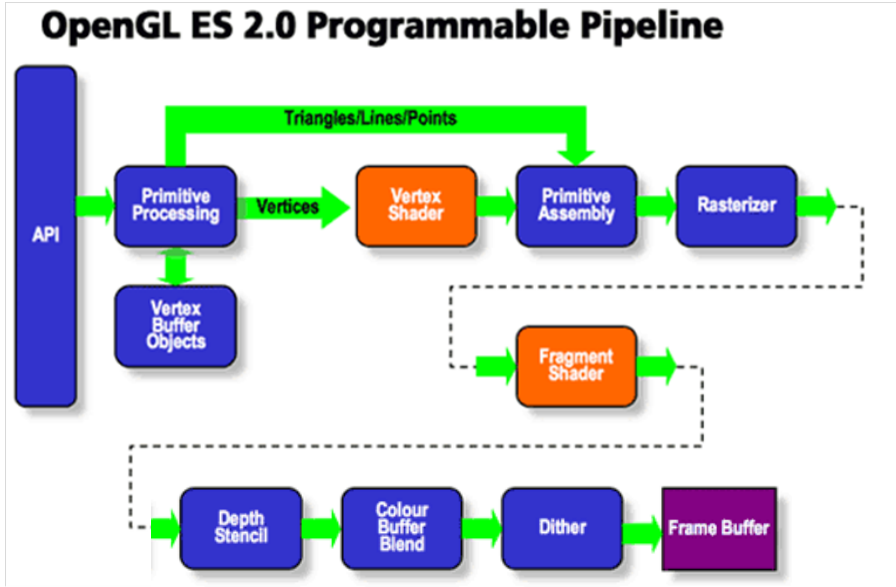
Rasterization

**Operations on Pixels**

| | |
|---|---|
| Traversal | What objects are in current scene? |
| Transforms | Where are the polygons? |
| Lighting | What color are the polygons? |
| Rasterize | What shape are they on the screen? |
| Color | What color is each pixel? |
| Clip | Which pixels are visible? |
| Write | Write the pixels to the framebuffer |

# Actual 3D Pipelines



# 3D evolving over more than 30 years


'Doom' 1993

'Samaritan' Real-time Demo - 2011

# Khronos and Hardware APIs

- **Khronos defines open, royalty-free standards to access graphics, media, compute and input hardware**
- **Khronos APIs are low-level – just above raw silicon - to create the "foundation" functionality needed on every platform**
- **Safe forum for industry cooperation**
  - Open to any company to join
  - IP framework to protect members and industry
- **By the industry for the industry**

accenture
Acrodea
Adobe
ALT Software Inc.
Altera
Antix
Aptina Imaging
axell Wireless
bixdroid
Broadcom
codeplay
Core Logic
Creative
csr
Dell
Digital Aria
dts Digital Entertainment
DMP
draw Elements
EA
Fixstars
Fujitsu
Google
Hanyang University
Hi Corp.
Hisilicon
hp
htc quietly brilliant
Huone
IBM
Kishonti Informatics

AMD
ARM
Epic Games
Apple
Sony Computer Entertainment
intel
Samsung
Imagination

**KHRONOS GROUP**

**Over 100 members – any company worldwide is welcome to join**

**Board of Promoters**

freescale semiconductor
Ericsson
Nokia
Oracle
nVidia
Qualcomm
Texas Instruments

KETI
KNU Kyungpook National University
Los Alamos National Laboratory
Marvell
matrox
mechnicality
MediaTek
Mentor Graphics
Mitsubishi Electric
Monotype Imaging
Motorola
Movidius
mozilla
mtekvision
Multicore Ware
NDS
NEC
Netallied Systems
NetLogic Microsystems
Nexus Chips
OpenEye Scientific Software
Opera Software
packetvideo
Panasonic
Presagis
QNX Software Systems
Renesas
RightWare
S3 Graphics
sasken
서울대학교 Seoul National University
Sigma
SK telecom
SmithMicro Software
SoftBank
SoftKinetic
Sony Ericsson
srs
ST
symbio
tactel
Takumi
Think Silicon
Toshiba
Transgaming
Universitet
Vivante
ViXS
vmware
The University of Western Australia
Xilinx
Xylon
Yamaha
Yumetech

# Khronos Family of Standards

**Authoring and accessibility**

- **COLLADA** — 3D Digital Asset Exchange format
- **WebGL** — Plugin-free 3D Web Content
- **WebCL** — Web Compute
- **StreamInput** — Unified Sensor and Input Processing

**Application Acceleration**

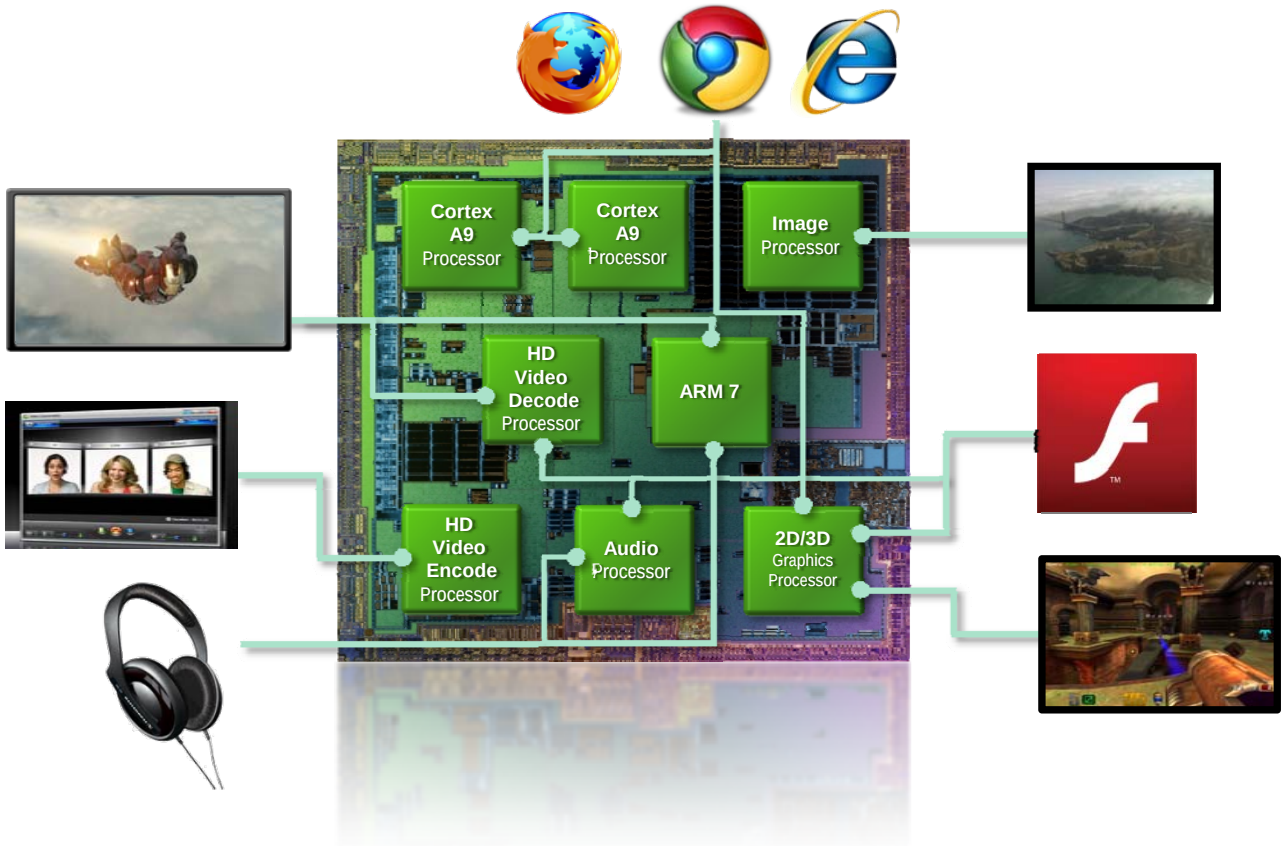
- **OpenGL** — Cross platform desktop 3D
- **OpenCL** — Parallel Computing
- **OpenGL|ES** — Embedded and Mobile 3D
- **OpenVG** — Vector 2D
- **OpenMAX** — Streaming Media
- **OpenSL|ES** — Advanced Audio
- **EGL** — Context, Sync and Surface Management

KHRONOS GROUP

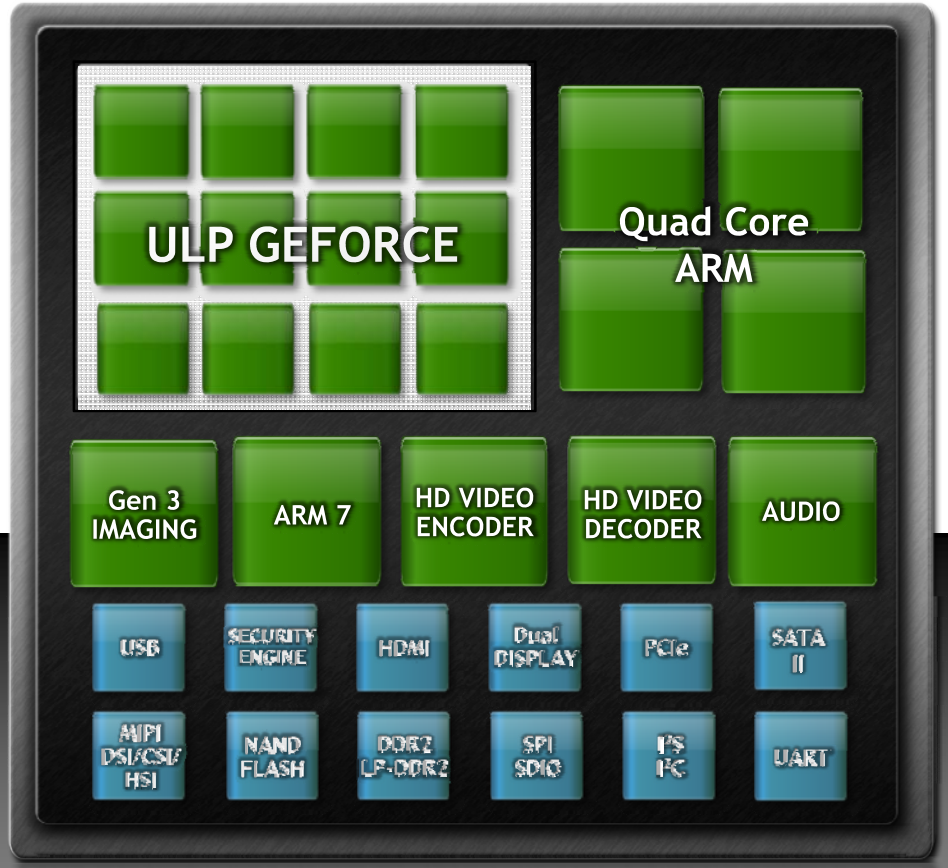
**A coordinated ecosystem of compute, graphics and media standards and APIs**

**Khronos creates royalty-free specifications to meet real market needs and helps drive industry adoption across multiple platforms**

# Mobile Silicon Experiential Processing

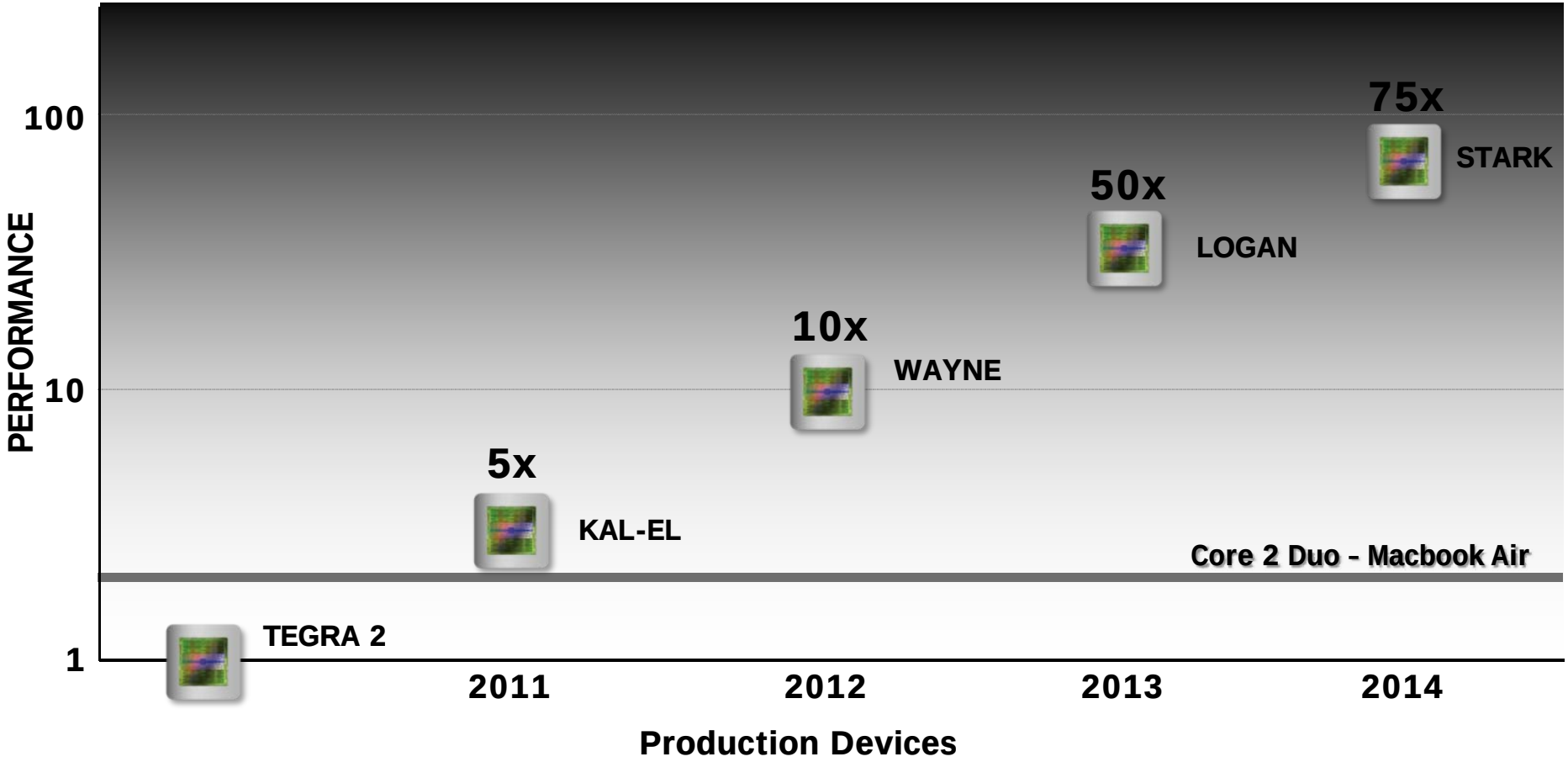


# Next Generation Mobile Processors

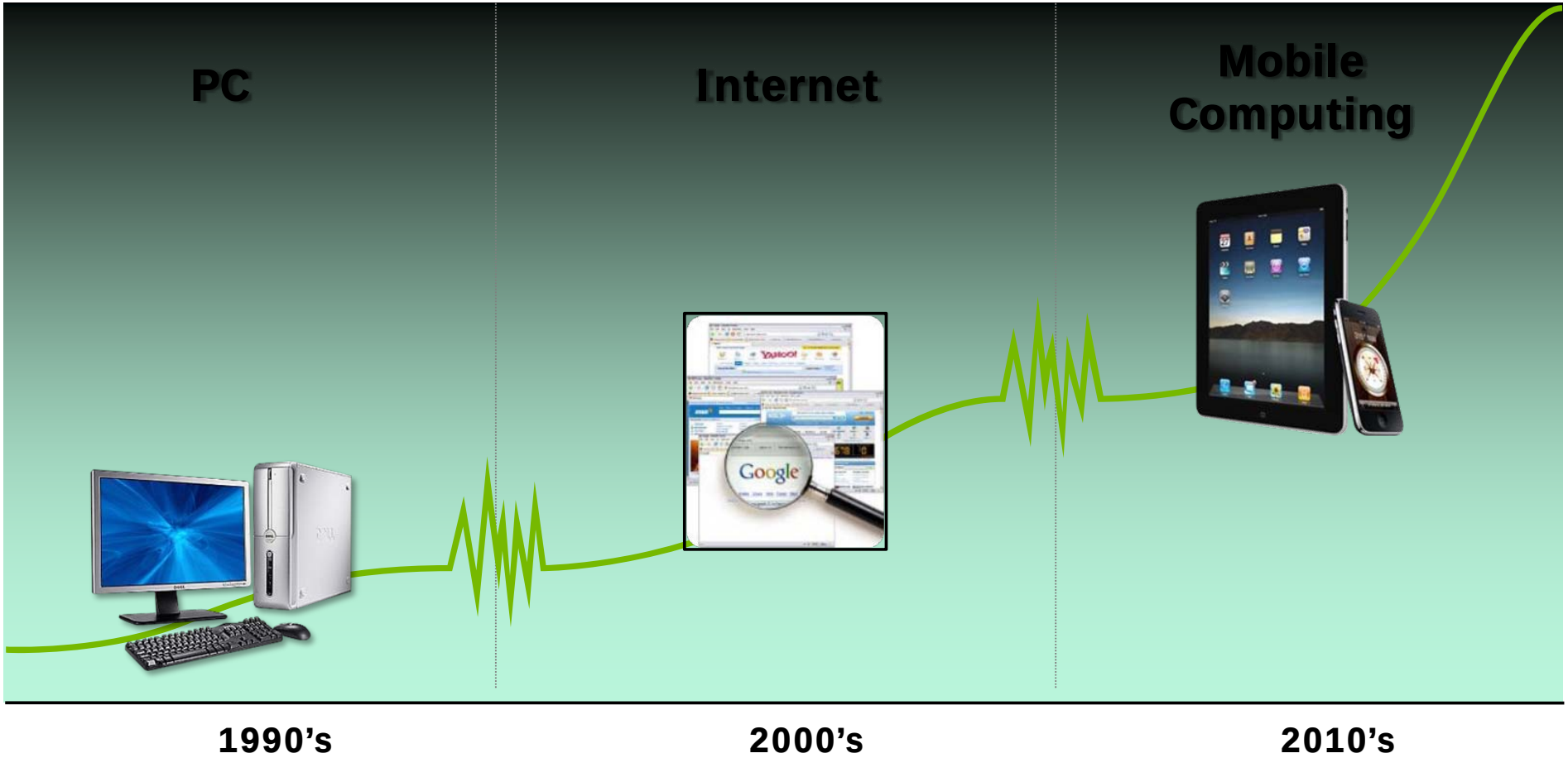



| | |
|---|---|
| CPU | **3X Performance**<br>*Quad Core up to 1.5GHz, NEON* |
| POWER | **20x Lower Standby Power**<br>*Due to ULP mode* |
| VIDEO | **4X Complexity**<br>*1080i/p High Profile* |
| GRAPHICS | **3X Performance**<br>*12 Core, Dual Pixel Pipe* |
| MEMORY | **3X bandwidth**<br>*DDR3L up to 1600 data rate* |
| IMAGING | **Better noise reduction & color rendition**<br>*Two simultaneous streams* |
| AUDIO | **HD Audio**<br>*7.1 channel surround* |
| STORAGE | **2 – 6X faster**<br>*e.MMC 4.4 and SATA-II* |

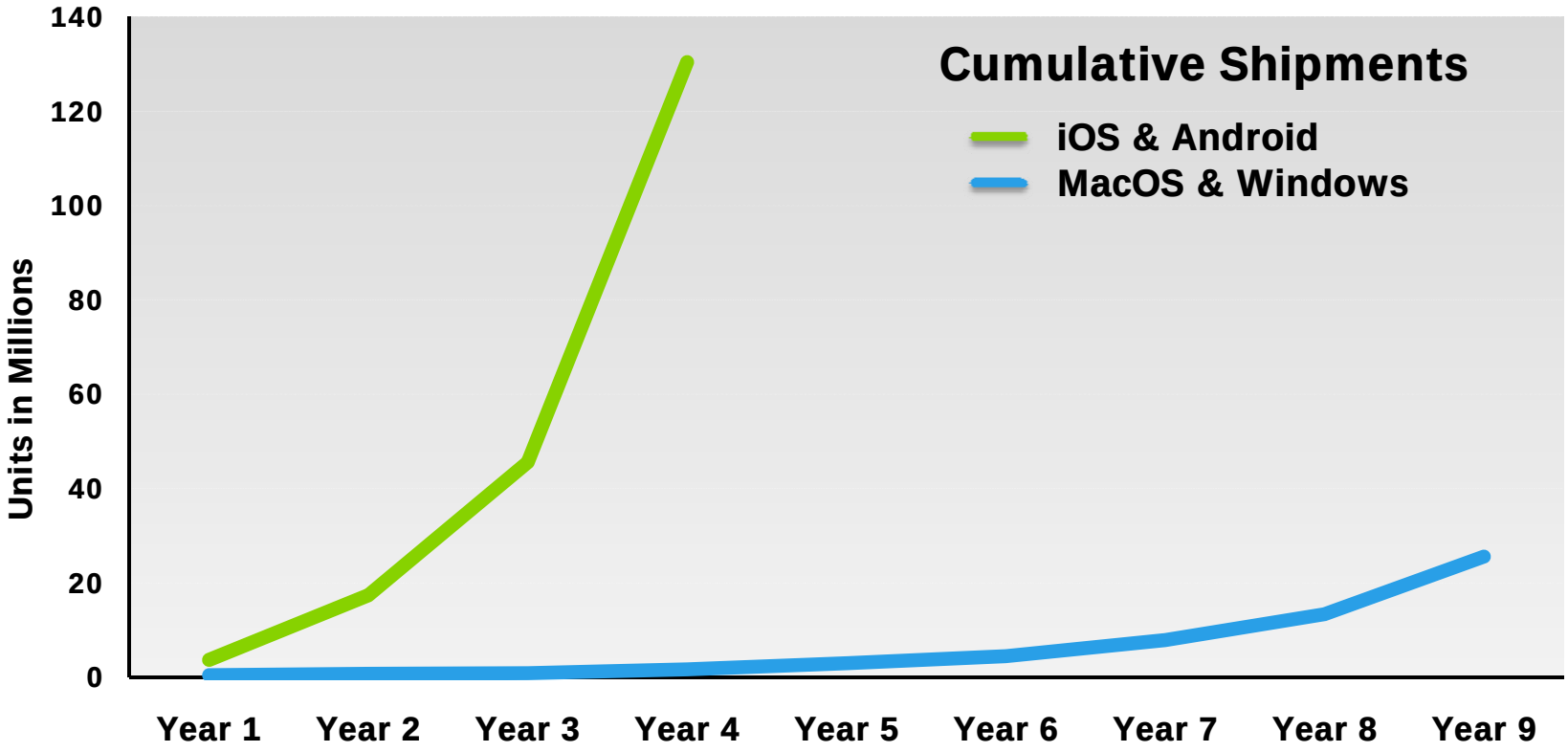
# Mobile Roadmap Acceleration



# A New Era in Personal Computing

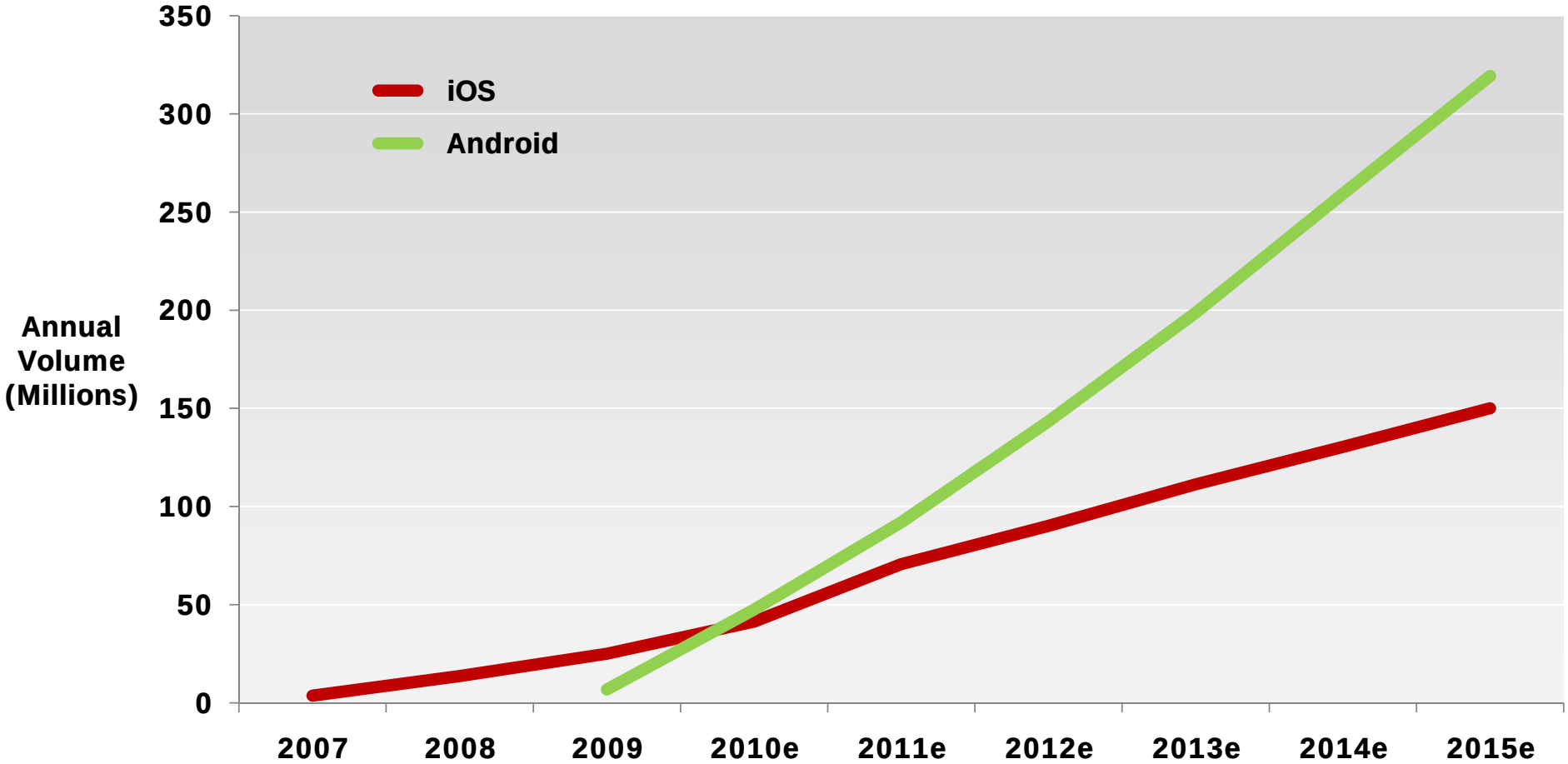


# 20 Years Faster to 100M Per Year




*Source: Gartner, Apple, NVIDIA*

# Mobile - Android Becoming Dominant OS




*Source: Gartner, NVIDIA*

# Mobile Form Factor Innovation

Motorola
Atrix Media Dock

Motorola
Atrix Nettop

Sony S2

Asus
Transformer

# OpenGL Ecosystem – 3D Everywhere

Leading-edge functionality developed first on desktop

OpenGL ES 2.0 on desktop as subset of OpenGL 4.1 for mobile content flexibility – including native support for WebGL

WebGL will drive new-generation security into OpenGL family

Mobile functionality subset that is deployed on billions of devices

Pervasive OpenGL ES 2.0 availability enables Browser vendors to build 3D directly into HTML5

# OpenGL ES Pervasiveness

- **OpenGL ES 1.1 – fixed-function pipeline**
  - Based on OpenGL 1.5
  - Vertex Arrays / Buffer Objects
  - Transform & Lighting
  - Multi-texturing (min 2 units)
  - Fixed-point & Floating-point profiles

- **OpenGL ES 2.0 – programmable pipeline**
  - Based on OpenGL 2.0
  - Adds vertex and fragment shader programming
  - Removes fixed function pipeline
  - Super-compact, efficient API
  - High level language (GLSL ES)
  - On-line or off-line compilation

# WebGL – 3D on the Web – No Plug-in!

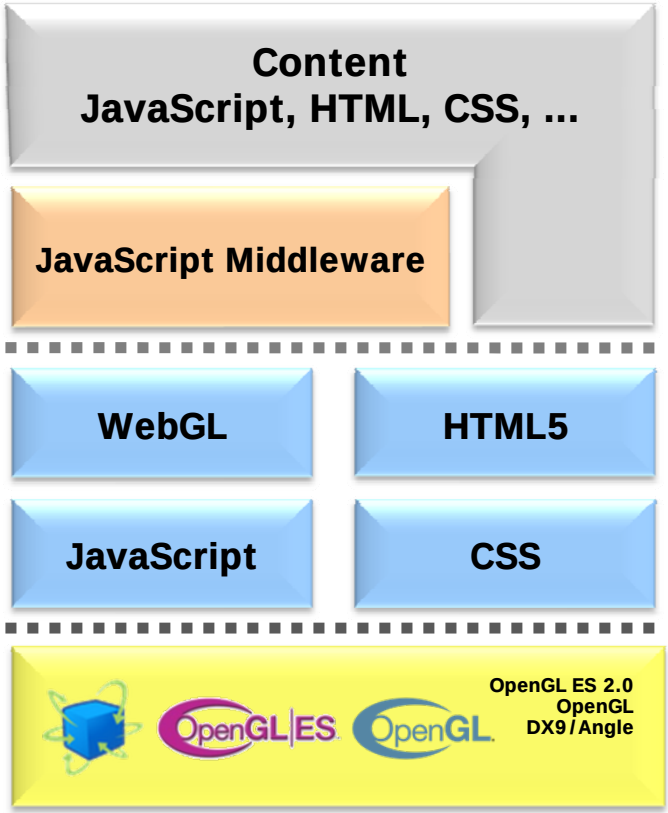
- **Historic opportunity to bring accelerated 3D graphics to web**
  - WebGL defines JavaScript binding to OpenGL ES 2.0
- **Leveraging HTML 5 and uses <canvas> element**
  - Enables a 3D context for the canvas
- **WebGL 1.0 Released at GDC March 2011**
  - Mozilla, Apple, Google and Opera working closely with GPU vendors

# WebGL Implementation Anatomy

Content downloaded from the Web. Middleware can make WebGL accessible to non-expert 3D programmers

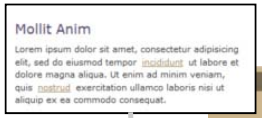
Browser provides WebGL functionality alongside other HTML5 specs - no plug-in required

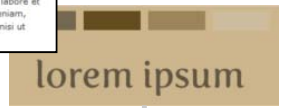
OS Provided Drivers. WebGL on Windows can use Google Angle to create conformant OpenGL ES 2.0 over DX9



# HTML5 Content Architecture

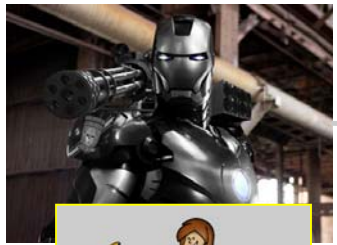
HTML content generated by layout engine 'on page'



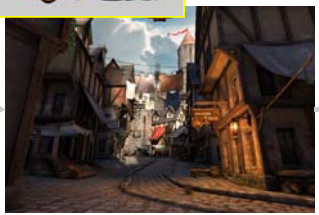

All content passes through CSS layout

<video> tag



<canvas> tag

JavaScript drives interactivity for 2D and 3D graphics



Composition of off-screen buffers

CSS Layout and Transforms

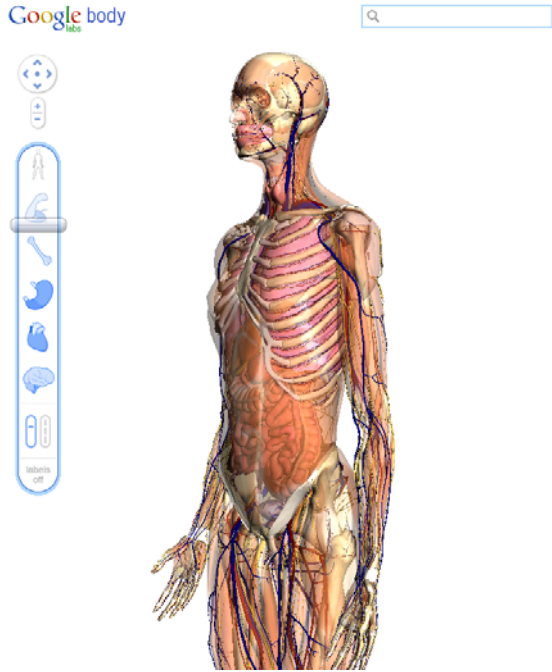

Composition needs to be GPU accelerated

Video, Vector Graphics and 3D created off-screen buffers

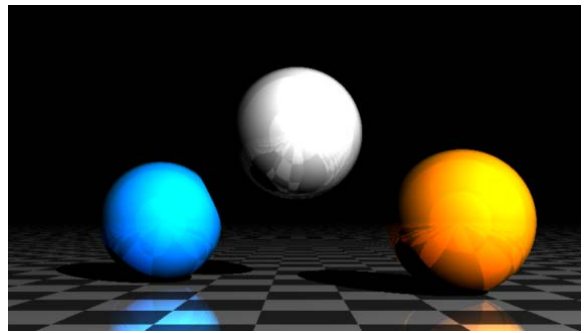
WebGL

# WebGL and HTML Interaction

- **3D is not trapped in a rectangular window**
  - 3D can overlay and underlay HTML content
  - Easy to make HUDs or user interfaces
- **Strong ties with other advanced HTML5**
  - WebGL can use HTML5 <video> or canvas as a texture
- **3D for core web UI – as well as content**
  - Advanced transforms and special effects

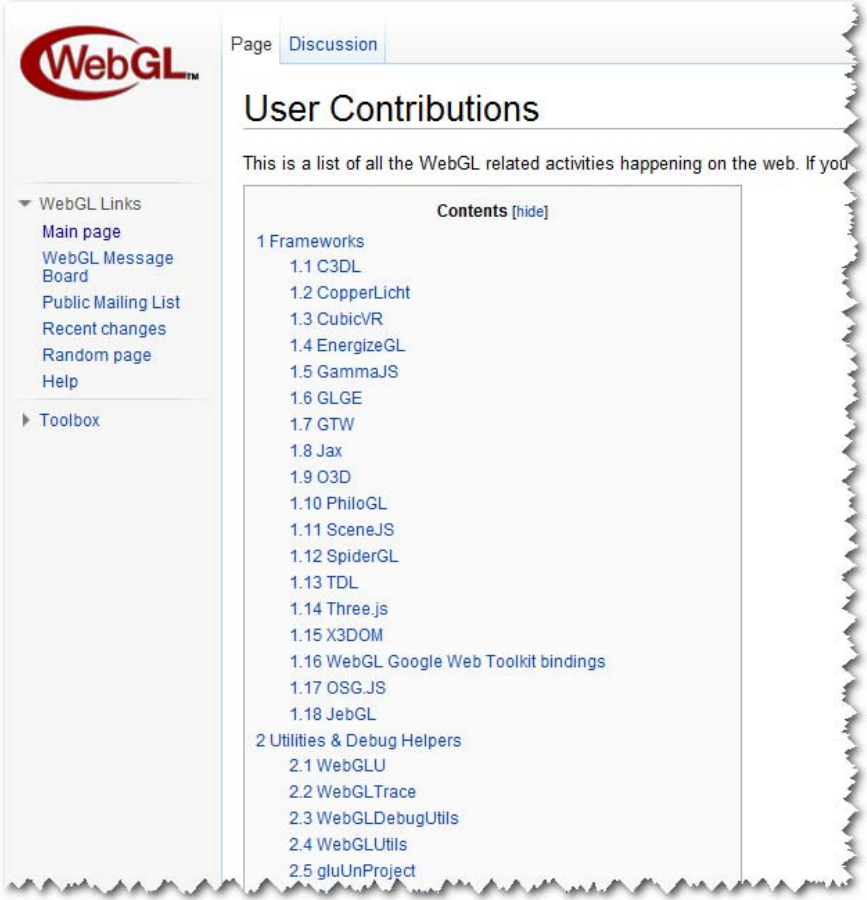



- **WebGL is democratization of 3D**
  - Accessible, pervasive, enabling
  - Spawning amazing innovation



# Frameworks and Tools

- WebGL is deliberately low level to enable the full power and flexibility of OpenGL ES 2.0
- If you are not an expert 3D programmer – don't panic!
- WebGL is perfect foundational layer for JavaScript middleware frameworks
- Lots of utilities and tools already appearing



# WebGL Deployment

- **Typed array 1.0 spec ratified by Khronos in May**
  - Supporting bulk data transfer between threads (workers)
  - Many use cases - background mesh loading, generation, deformation, Physics …
- **1.0.1 release of WebGL spec and conformance suite imminent**
  - 100% robust stance on security
  - Fixing bugs in 1.0.0 conformance suite
  - Implementations will report getContext("webgl") (not experimental)
- **Render HTML DOM sub-tree as texture prototype extension**
  - Support user interaction when in 3D
  - Mozilla and Google prototyping

http://caniuse.com/#search=webgl

| Show all versions | IE | Firefox | Safari | Chrome | Opera |
|---|---|---|---|---|---|
| 3 versions back | 6.0 | 3.5 | 3.2 | 9.0 | 10.6 |
| 2 versions back | 7.0 | 3.6 | 4.0 | 10.0 | 11.0 |
| Previous version | 8.0 | 4.0 | 5.0 | 11.0 | 11.1 |
| Current | 9.0 | 5.0 | 5.1 | 12.0 | 11.5 |
| Near future | | 6.0 | | 13.0 | 12.0 |
| Farther future | 10.0 | 7.0 | 6.0 | 14.0 | 12.1 |

**Not enabled by default**

# WebGL Security

- **Any new functionality in the browser increases exposure to attack**
  - True since the beginning of the web – the new functionality becomes hardened
- **ANY graphics in the browser need the GPU drivers to be hardened**
  - HTML, Canvas, WebGL, Adobe Molehill, Silverlight 5 ...
- **WebGL is designed with security as the highest priority**
  - Hardening is being strongly promoted and enabled
- **Short term – browser vendors will maintain white and black lists**
  - Compromised system can have WebGL disabled until mitigation developed
- **Longer term – GPUs will provide increasingly robust security and tasking**
  - GPU becoming a first-class computing platform alongside CPU

# WebGL Security in the Press!

- **Confusion in the industry as we start this hardening process**
  - Shader programs *cannot* access general system resources or perform out of range memory access!
- **Issues in the Press**
  - Cross domain image access – timed loop attack
    - WebGL *and HTML* spec updates - mandating CORS for video, images and audio
    - Servers have to grant cross-domain access to media resources
  - General hardening
    - ARB_robustness extensions that provide additional protection being mandated
    - New robustness spec limits the side-effects of a GPU reset after a DOS attack
    - ANGLE shader validator improved; more improvements coming

# Why Khronos?

- Unique forum where browser and GPU vendors can cooperate
- Opened process to enable cooperation with web community

# Flash Stage 3D aka 'Molehill'

- **GPU-friendly 3D 'stage' behind classic Flash graphics**
  - No interaction with classic Flash except as overlay to 3D
- **Healthy competition to WebGL**
  - Competition *IS* a good thing
- **Contrasting design vector to WebGL**
  - OpenGL ES 2.0 assembler
- **Portability at the cost of functionality**
  - No loops
  - Lowest common denominator
- **Competition will ensure WebGL keeps it's eyes on the ball for *security* and *portability***

# 3D is much more than ‘just’ games

- **Augmented Reality is a great Lighthouse use case**
  - Need consistent APIs
  - AND
  - Reliable interop between them
- **Significant Functionality**
  - Camera control
  - Image processing
  - Positional sensors
  - Parallel computing
  - Graphics rendering
  - Video/graphics composition
  - Positional Audio
  - 3D models over the network

# Visual-based Augmented Reality

Camera video stream
sent to the compositor

Camera
images used
to track the
camera's
location and
orientation

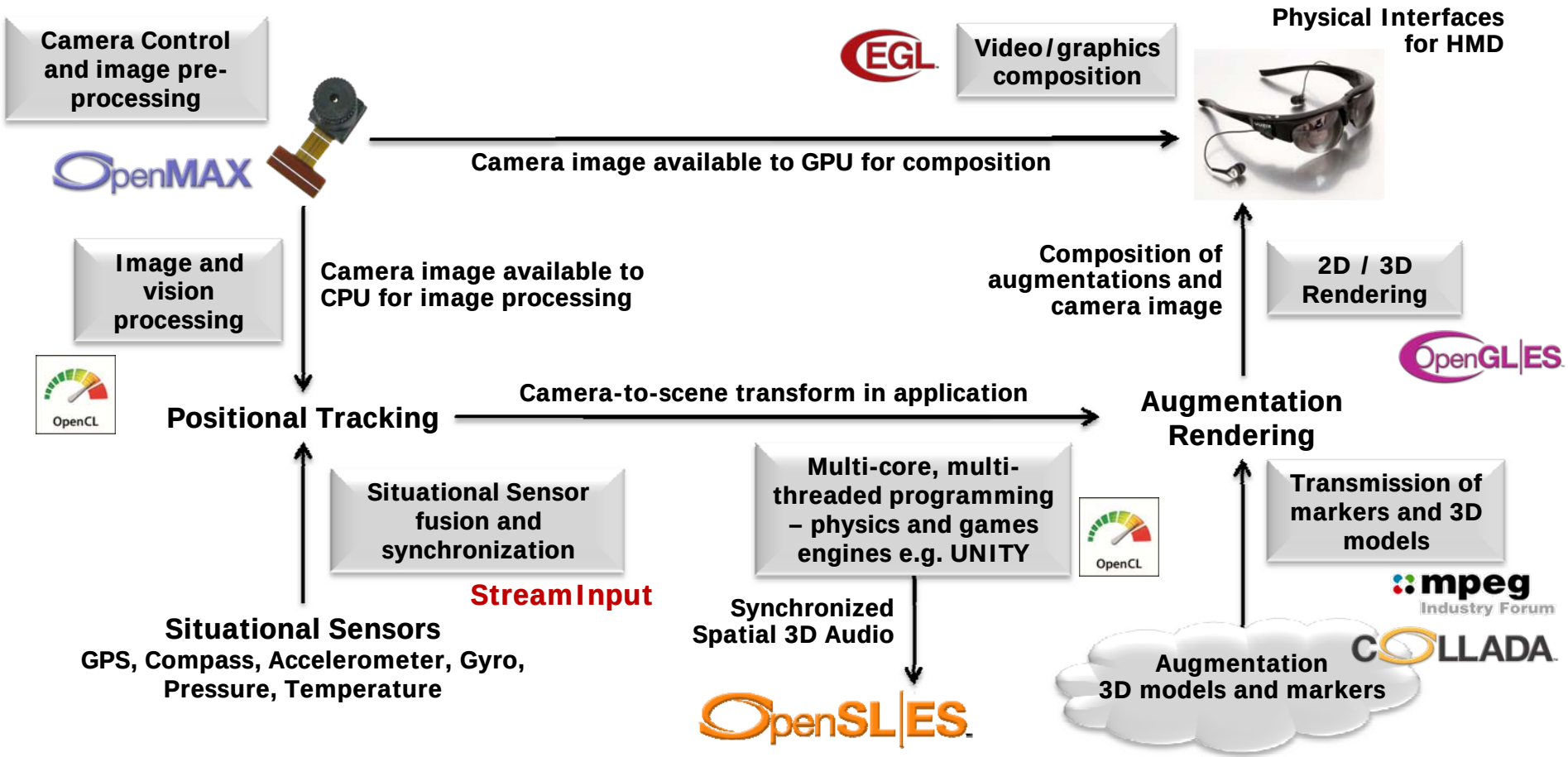

3D
augmentations
composited with
video stream

Camera
Tracking

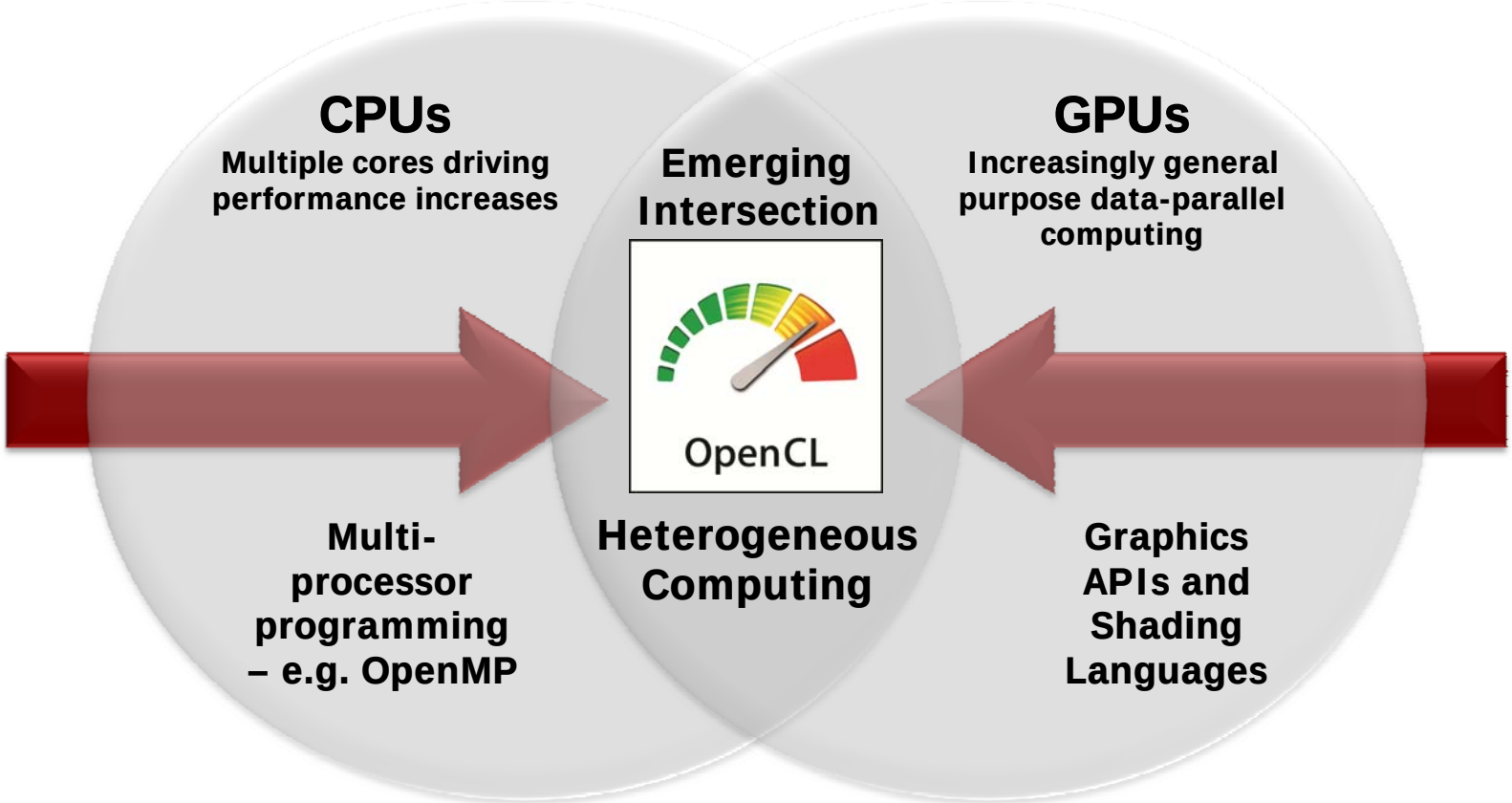
Camera-to-scene transform locks
the 3D rendering to the real world

3D Augmentation
Rendering

# APIs Needed for Visual AR



# Processor Parallelism




**OpenCL is a programming framework for heterogeneous compute resources**

# The BIG Idea behind OpenCL

- **OpenCL execution model …**
  - Define N-dimensional computation domain
  - Execute a kernel at each point in computation domain
- **C Derivative to write kernels – based on ISO C99**
  - APIs to discover devices in a system and distribute work to them
- **Targeting many types of device**
  - GPUs, CPUs, DSPs, embedded systems, mobile phones.. Even FPGAs

**Traditional loops**

```
void
trad_mul(int n,
         const float *a,
         const float *b,
         float *c)
{
  int i;
  for (i=0; i<n; i++)
    c[i] = a[i] * b[i];       }
```

**Data Parallel OpenCL**

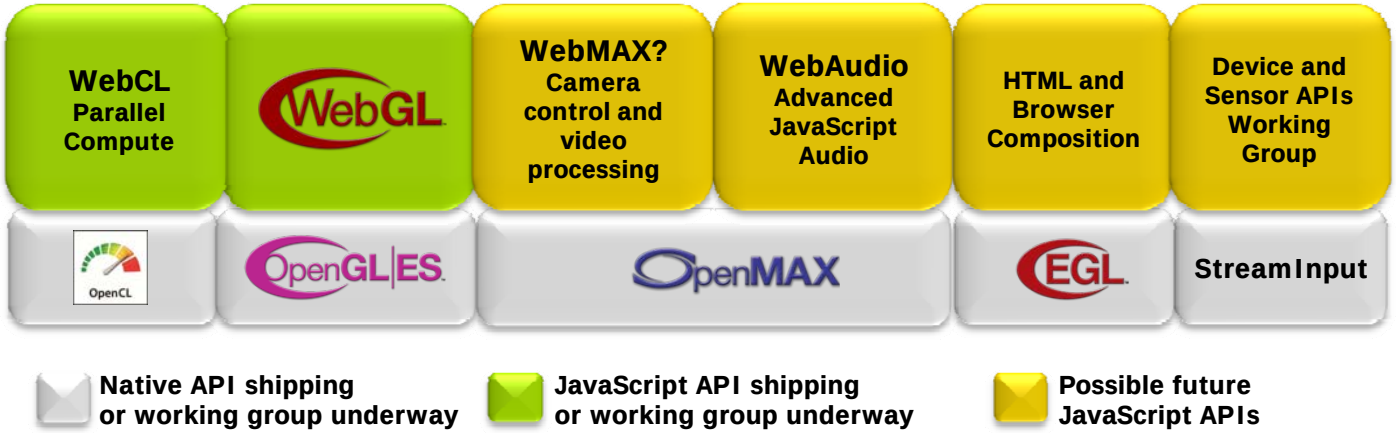
```
kernel void
dp_mul(global const float *a,
       global const float *b,
       global float *c)
{
  int id = get_global_id(0);

  c[id] = a[id] * b[id];

} // execute over “n” work-items
```

# WebCL – Call for Participation

- **At GDC Khronos announced new WebCL initiative**
  - 'To bring parallel computing to browsers'
- **E.g. Physics engines to complement WebGL**
  - Image and video editing in browser
- **One possible direction is JavaScript binding to OpenCL**
  - Security is top priority
- **Khronos welcomes new members to help define and drive WebCL**
  - info@khronos.org

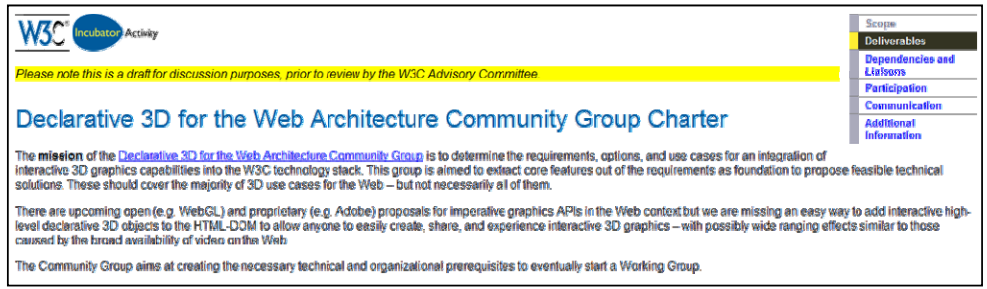
# Expanding HTML5 Capability

- **Web is the most widespread cross-platform programming platform**
  - HTML5 Canvas tag is opening the door to API innovation
- **JavaScript is now a viable language for visual computing**
  - Most native APIs enable local caching of geometry/configuration
- **Opportunity to synergize between Web and native APIs**
  - Increase leverage, reduce developer learning cycles



# Declarative 3D for the Web

- **Need to enable 'non-expert' web programmers with layers over WebGL**
  - 10,000s of 3D programmers worldwide versus millions of web developers
  - Middleware and layered architectures play a vital role
- **W3C Incubator for Declarative 3D**
  - *"easy way to add interactive high-level declarative 3D objects to the HTML-DOM"*
  - X3DOM (www.x3dom.org/) and XML3D (www.xml3d.org/)
- **Bind 3D even closer into the browser stack**
  - Use as much HTML5 machinery as possible – DOM, JavaScript, CSS
  - Focus on driving optimized WebGL/OpenGL ES 2.0 back-end
  - Use Typed Arrays and drive for optimal performance



W3C Incubator Activity

Scope
Deliverables
Dependencies and Liaisons
Participation
Communication
Additional Information

*Please note this is a draft for discussion purposes, prior to review by the W3C Advisory Committee*

Declarative 3D for the Web Architecture Community Group Charter

The **mission** of the Declarative 3D for the Web Architecture Community Group is to determine the requirements, options, and use cases for an integration of interactive 3D graphics capabilities into the W3C technology stack. This group is aimed to extract core features out of the requirements as foundation to propose feasible technical solutions. These should cover the majority of 3D use cases for the Web – but not necessarily all of them.

There are upcoming open (e.g. WebGL) and proprietary (e.g. Adobe) proposals for imperative graphics APIs in the Web context but we are missing an easy way to add interactive high-level declarative 3D objects to the HTML-DOM to allow anyone to easily create, share, and experience interactive 3D graphics – with possibly wide ranging effects similar to those caused by the broad availability of video on the Web.

The Community Group aims at creating the necessary technical and organizational prerequisites to eventually start a Working Group.

# Its Time for a 3D Delivery Format!

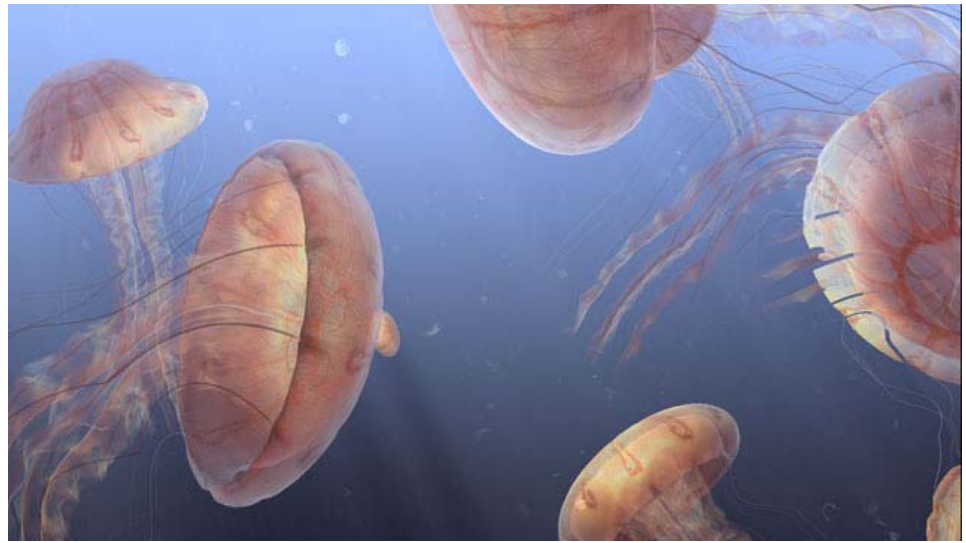
- **Format soup – only 3D does not have any widely agreed delivery formats**
  - COLLADA, KML, MPEG, VRML, JSON, X3D binary, PowerVR POD, GZIP etc. etc.
- **Fundamental to a '3D on the Web' infrastructure**
  - Compression reduces delivery time
  - Streaming with LOD flexibility increases end-user responsiveness
  - Browsers, apps and silicon can implement native accelerated decoders
  - Enables widely accessible, efficient content repositories
- **Khronos and MPEG starting discussions**
  - Leverage MPEG-4 AFX...
  - Encode COLLADA full-scene - geometry, textures, materials, animations, physics...
  - Restful API to negotiate precise served content...

mpeg Industry Forum

| Audio | Video | Images | 3D |
|---|---|---|---|
| MP3 | H.264 | PNG/JPEG | ? |

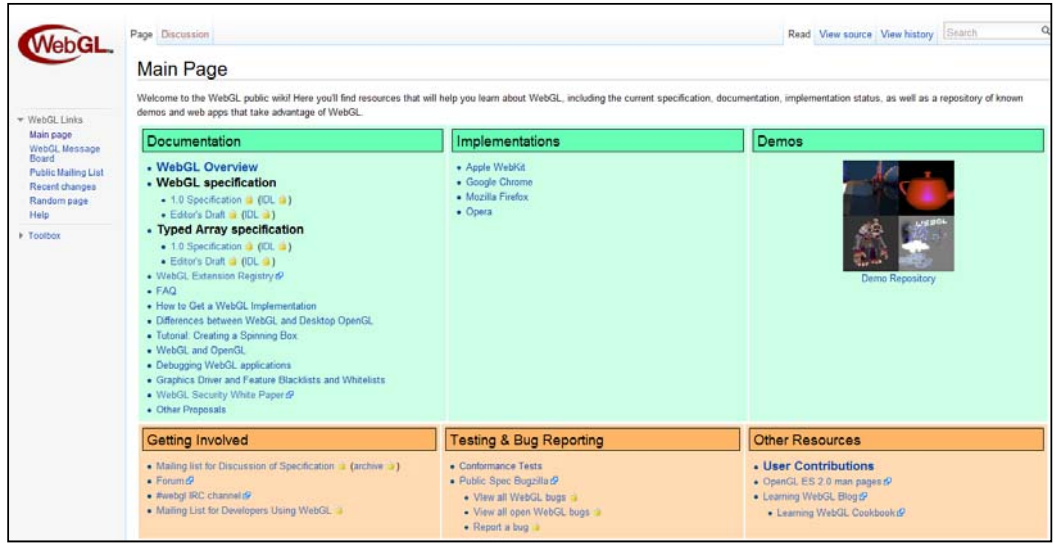
# In Summary

- WebGL brings another vital piece of system capability into the HTML5 browser for web apps – 3D graphics
- WebGL is being deployed right now on PC – soon on mobile – and is being strongly supported by browser and GPU vendors
- WebGL is a low-level, secure technology – that can be used directly and will support a rich ecosystem of tools and frameworks



# Get Involved!

- **Engage with the WebGL working group on Khronos forums and mailing lists**
- **Let us know if you have news or links that Khronos can help highlight**
  - info@khronos.org or edit the Wiki
- **Local WebGL Meetups here in NYC**
  - http://www.meetup.com/NYC-WebGL-Developers/




http://www.khronos.org/webgl/wiki/Main_Page

# Questions?

## Come get a Reference Card!

### WebGL 1.0 API Quick Reference Card - Page 1

WebGL® is a software interface for accessing graphics hardware from within a web browser. Based on OpenGL ES 2.0, WebGL allows a programmer to specify the objects and operations involved in producing high-quality graphical images, specifically color images of 3D objects.

- [n.n.n] refers to sections in the WebGL 1.0 specification, available at www.khronos.org/webgl
- Content marked in purple does **not** have a corresponding function in OpenGL ES. The OpenGL ES 2.0 specification is available at www.khronos.org/registry/gles

**WebGL function calls behave identically to their OpenGL ES counterparts unless otherwise noted.**

#### The WebGL Context and getContext() [2.5]

This object manages OpenGL state and renders to the a drawing buffer, which must is also be created at the same time of as the context creation. Create the `WebGLRenderingContext` object and drawing buffer by calling the **getContext** method of a given HTMLCanvasElement object with the exact string 'webgl'. The drawing buffer is also created by **getContext**.

For example:

```
<!DOCTYPE html>
<html><body>
  <canvas id="c"></canvas>
  <script type="text/javascript">
    var canvas = document.getElementById("c");
    var gl = canvas.getContext("webgl");
    gl.clearColor(1.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);
  </script>
</body></html>
```

#### Interfaces

Interfaces are optional requests and may be ignored by an implementation. See **getContextAttributes** for actual values.

**WebGLContextAttributes [5.2]**

This interface contains requested drawing surface attributes and is passed as the second parameter to **getContext**.

**Attributes:**

| | |
|---|---|
| **alpha** | Default: true<br>If true, requests a drawing buffer with an alpha channel for the purposes of performing OpenGL destination alpha operations and compositing with the page. |
| **depth** | Default: true<br>If true, requests drawing buffer with a depth buffer of at least 16 bits. |
| **stencil** | Default: false<br>If true, requests a stencil buffer of at least 8 bits. |
| **antialias** | Default: true<br>If true, requests drawing buffer with antialiasing using its choice of technique (multisample/supersample) and quality. |
| **premultipliedAlpha** | Default: true<br>If true, requests drawing buffer which contains colors with premultiplied alpha. (Ignored if Alpha is false.) |
| **preserveDrawingBuffer** | Default: false<br>If true, requests that contents of the drawing buffer remain in between frames, at potential performance cost. |

**WebGLObject [5.3]**

This is the parent interface for all WebGL resource objects.

**Resource interface objects:**

| | |
|---|---|
| **WebGLBuffer** [5.4] | OpenGL Buffer Object. |
| **WebGLProgram** [5.6] | OpenGL Program Object. |
| **WebGLRenderbuffer** [5.7] | OpenGL Renderbuffer Object. |
| **WebGLShader** [5.8] | OpenGL Shader Object. |
| **WebGLTexture** [5.9] | OpenGL Texture Object. |
| **WebGLUniformLocation** [5.10] | Location of a uniform variable in a shader program. |
| **WebGLActiveInfo** [5.11] | Information returned from calls to **getActiveAttrib** and **getActiveUniform**. Has the following read-only properties: name, location, size, type. |

**WebGLRenderingContext [5.13]**

This is the prinicpal interface in WebGL. The functions listed on this reference card are available within this interface.

**Attributes:**

| | |
|---|---|
| **canvas** | Type: HTMLCanvasElement<br>A reference to the canvas element which created this context. |
| **drawingBufferWidth** | Type: GLsizei<br>The actual width of the drawing buffer, which may differ from the width attribute of the HTMLCanvasElement if the implementation is unable to satisfy the requested width or height. |
| **drawingBufferHeight** | Type: GLsizei<br>The actual height of the drawing buffer, which may differ from the height attribute of the HTMLCanvasElement if the implementation is unable to satisfy the requested width or height |

#### ArrayBuffer and Typed Arrays [5.12]

Data is transferred to WebGL using *ArrayBuffer* and views. Buffers represent unstructured binary data, which can be modified using one or more typed array views.

**Buffers**

**ArrayBuffer**(ulong *byteLength*)
ulong byteLength: read-only, length of view in bytes.
Creates a new buffer. To modify the data, create one or more views referencing it.

**Views**

In the following, *ViewType* may be Int8Array, Int16Array, Int32Array, UInt8Array, UInt16Array, UInt32Array, Float32Array.

*ViewType*(ulong *length*)
Creates a view and a new underlying buffer.
ulong *length*: Read-only, number of elements in this view.

*ViewType*(*ViewType other*)
Creates new underlying buffer and copies 'other' array.

*ViewType*(type[] *other*)
Creates new underlying buffer and copies 'other' array.

*ViewType*(ArrayBuffer *buffer*, [optional] ulong *byteOffset*, [optional] ulong *length*)
Create a new view of given buffer, starting at optional byte offset, extending for optional length elements.
ArrayBuffer *buffer*: Read-only, buffer backing this view
ulong *byteOffset*: Read-only, byte offset of view start in buffer
ulong *length*: Read-only, number of elements in this view

**Other Properties**

ulong *byteLength*: Read-only, length of view in bytes.
const ulong *BYTES_PER_ELEMENT*: element size in bytes.

**Methods**

*view*[i] = get/set element i

**set**(*ViewType other*, [optional] ulong **offset**)

**set**(type[] *other*, [optional] ulong *offset*)
Replace elements in this view with those from other, starting at optional offset.

*ViewType* **subset**(long *begin*, [optional] long *end*)
Return a subset of this view, referencing the same underlying buffer.

#### Per-Fragment Operations [5.13.3]

void **blendColor**(float *red*, float *green*, float *blue*, float *alpha*)

void **blendEquation**(enum *mode*)
*mode*: See *modeRGB* for **blendEquationSeparate**

void **blendEquationSeparate**(enum *modeRGB*, enum *modeAlpha*)
*modeRGB*, and *modeAlpha*: FUNC_ADD, FUNC_SUBTRACT, FUNC_REVERSE_SUBTRACT

void **blendFunc**(enum *sfactor*, enum *dfactor*)
*sfactor*: Same as for *dfactor*, plus SRC_ALPHA_SATURATE
*dfactor*: ZERO, ONE, [ONE_MINUS_]SRC_COLOR, [ONE_MINUS_]DST_COLOR, [ONE_MINUS_]SRC_ALPHA, [ONE_MINUS_]DST_ALPHA, [ONE_MINUS_]CONSTANT_COLOR, [ONE_MINUS_]CONSTANT_ALPHA
**Note:** Src and dst factors may not both reference constant color

void **blendFuncSeparate**(enum *srcRGB*, enum *dstRGB*, [illegible]

#### Whole Framebuffer Operations [5.13.3]

void **clear**(ulong *mask*) [5.13.11]
*mask*: Bitwise OR of (COLOR, DEPTH, STENCIL)_BUFFER_BIT

void **clearStencil**(int *s*)

void **colorMask**(bool *red*, bool *green*, bool *blue*, bool *alpha*)

void **depthMask**(bool *flag*)