

*The 20th
Visualization
Conference*

OpenGL/OpenGL ES最新情報及び 次世代OpenGL

株式会社デジタルメディアプロフェッショナル
大淵 栄作

07/Nov/2014

Agenda

- **DMP概要**
- **Khronos graphics APIのマーケット状況**
- **OpenGL アップデート**
- **OpenGL ES アップデート**
- **次世代OpenGL/OpenGLES**

デジタルメディアプロフェッショナル概要

- 設立: 2002年7月
- 所在地: 東京都中野区
- 資本金: 8億2千万円
- 主要株主: UKCホールディングス、他
- 事業内容:
 1. グラフィックスIPコアのライセンス
 2. グラフィックスLSIのファブレス開発・製造



当社IPコアを使った顧客製品例



本社(中野セントラルパーク)



当社LSI製品

- 特許等: 日米欧において63件の特許申請済み 成立38件

DMPグラフィックスIP製品

■ 組込み機器向け高性能・低消費電力グラフィックスIP コア

- 高性能2D/3DグラフィックスIP
- 低電力モバイルから高性能アミューズメントまでサポート
- ビルディング・ブロック構造によるスケーラブルなアーキテクチャ



(企業部門 最高賞)



フォトリアリスティック
3Dグラフィックス
(OpenGL ES 1.1 互換 + 独自拡張)
PICA200

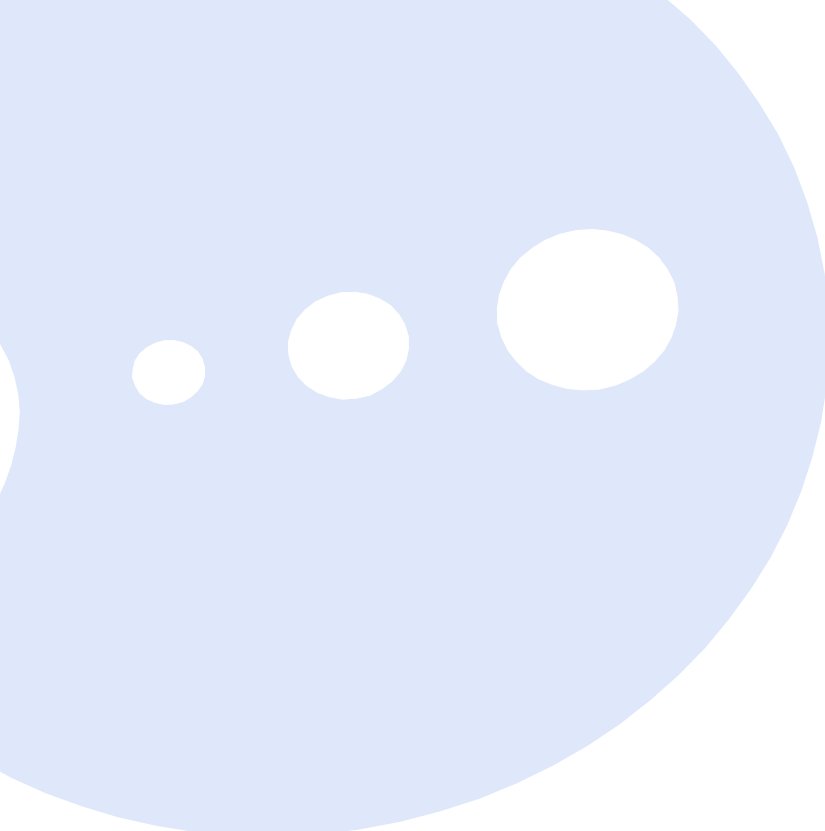


標準3Dグラフィックス
PICA200Lite (OpenGL ES 1.1)
SMAPH-S (OpenGL ES 2.0/3.0)



標準2Dベクターグラフィックス
SMAPH-F (OpenVG 1.1)

標準3D/2Dベクターグラフィックス
SMAPH-H (OpenGL ES/OpenVG)



Khronos graphics APIの マーケット状況

Access to 3D on Over 2 BILLION Devices



Source: Gartner (December 2013)

Cross Platform Challenge



One family of
GPUs



One OS



One GPU on
one OS



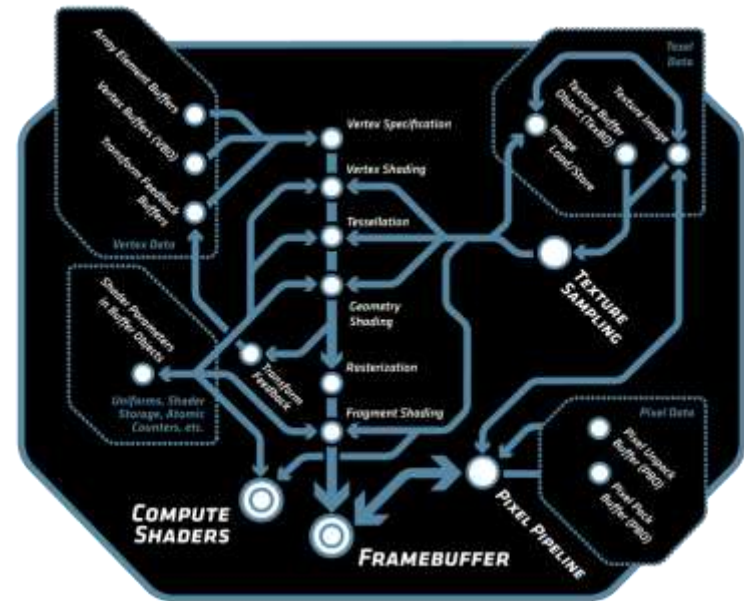
All Modern Platforms and GPUs

Participation of key players
Proven IP Framework

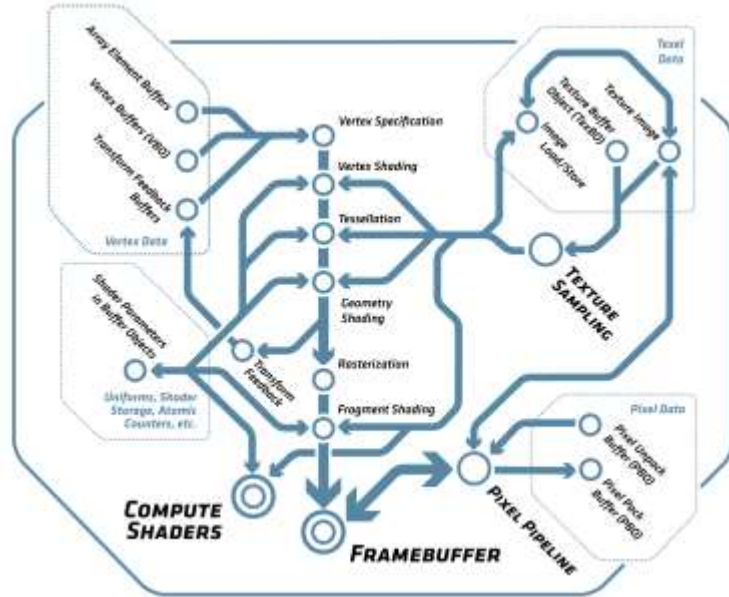
Battle-tested cooperative model

The *drive* to not let the 3D industry fragment

OpenGLアップデート



Continuing OpenGL Innovation



Bringing state-of-the-art
functionality to cross-
platform graphics

OpenGL 4.5

OpenGL 4.4

OpenGL 4.3

OpenGL 4.2

OpenGL 4.1

OpenGL 3.3/4.0

OpenGL 3.2

OpenGL 3.1

OpenGL 2.0

OpenGL 2.1

OpenGL 3.0

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

**DirectX
9.0c**

**DirectX
10.0**

**DirectX
10.1**

**DirectX
11**

**DirectX
11.1**

**DirectX
11.2**

OpenGL 4.5で新しくなったこと

- **ダイレクトステートアクセス - Direct State Access (DSA)**
 - バインドをせずに直接オブジェクト内のステートの参照、変更を可能
- **ロバストネス - Robustness**
 - WebGLサポートブラウザのために、セキュアなプラットフォームを提供(他の実行中のアプリケーションに影響するGPUリセットを避けるなど)
- **DirectX11エミュレーション機能**
 - OpenGL-Direct3D間でのアプリケーションポーティングを容易にする
- **OpenGL ES 3.1 API及びシェーダの互換機能**
 - デスクトップ上でのOpenGL ESアプリケーション開発・実行が可能に
- **MakeCurrentの高速化**
 - コンテキストのフラッシュ処理の明示的な制御
- **テクスチャバリア - Texture Barrier**
 - Read-after-write(RAW)ハザードを避ける



GLSL 4.5で新しくなったこと(続き)

- **ARB_derivative_control**
 - 傾き(derivative)演算における空間精度の制御
- **ARB_shader_texture_image_samples**
 - シェーダからテクスチャのサンプル数を照会することが可能
 - `textureSamples()`, `imageSamples()`
- **ARB_cull_distance**
 - 新たなGLSL `gl_CullDistance` シェーダ出力を追加(`gl_ClipDistance`の様なものだけど全体のプリミティブカリングに対して使用)

新しいOpenGL ARB拡張

- ARB_sparse_buffer
 - バッファオブジェクト向けのARB_sparse_textureの様な拡張。仮想的に大きなバッファが作れる(実メモリ上は一部置かれるだけ)
- ARB_pipeline_statistics_query
 - OpenGLパイプラインの異なるパートにおける統計情報の照会
 - DirectX 11互換
- ARB_transform_feedback_overflow_query
 - トランスフォームフィードバックバッファがオーバーフローした場合のクエリ
 - DirectX 11互換

OpenGL 4.5 - DirectX11 エミュレーション

- ARB_conditional_render_inverted
 - BeginConditionalRenderに対して、描画、描画しないの状態の意味を反転させるモードの追加
- ARB_get_texture_subimage
 - テクスチャのサブ領域の取得
- ARB_clip_control
 - D3Dの座標系と合わせる
 - Z座標のマッピングについて演算精度改善

OpenGL 4.5 - OpenGL ES 3.1互換

- ARB_ES3_1_compatibility

- MemoryBarrierRegion APIの追加
- 最小SSBO(Shader Storage Buffer Object)サイズの増加(2^{27})
- GLSL ES version 310サポート
- r32f FPイメージサポートのためimageAtomicExchangeの追加
- int, uint, bool選択についてmix()を拡張
- GLSLビルトインフラグメントシェーダ入力追加: gl_HelperInvocation (近接ピクセルの参照)
- GLSLビルトインコンスタント
 - gl_MaxSamples, gl_Max*ImageUniforms, gl_MaxCombinedShaderOutputResources
- atomic* とimageAtomic*向けにすべてのメモリ変数にコヒーレント修飾子を追加

ダイレクトステートアクセス - Direct State Access

DSA前

```
void Texture2D::SetMagFilter(GLenum filter)
{
    GLuint oldTex;
    glGetIntegerv(GL_TEXTURE_BINDING_2D, &oldTex);
    glBindTexture(GL_TEXTURE_2D, m_tex);
    glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER, filter);
    glBindTexture(GL_TEXTURE_2D, oldTex);
}
```

DSA後

```
void Texture2D::SetMagFilter(GLenum filter)
{
    glTextureParameteri(m_tex, GL_TEXTURE_MAG_FILTER,
filter);
}
```

EXT_direct_state_accessとの違い

- OpenGL 4.5では**コア**機能としてサポート
- いくつかの機能について微妙に名前を変更
 - EXTサフィックスの削除
 - `TextureParameterfEXT` -> `TextureParameterf`
 - VAO機能名を短縮
 - `glVertexArrayVertexBindingDivisorEXT` -> `glVertexArrayBindingDivisor`
 - Texture機能でTargetパラメータの指定が不要に
 - Targetパラメータは`glCreateTextures(<target>, )`で設定
 - zが面を指定するCUBE_MAPとともに3Dテクスチャ機能を使用
- DSA機能はオブジェクト生成不要
 - 名前とオブジェクトを生成には `glCreate*` を使用

ロバストネス - Robustness

- ARB_robustnessがOpenGL 4.5機能の一部に
 - OpenGL ES と使うときと同様にKHR_robustnessを呼び出す
 - この際OES互換機能を含んではいけない
- ユーザポイントに返されたデータの照会のために安全なAPIを追加
- GPUリセットについてアプリが覚えておくための仕掛けを追加
 - 自分やそのほか不正動作したアプリによるリセット
- より強力な範囲外の挙動のカバー
 - 未定義な挙動はない
- WebGL実装使用

ロバストネスSafe API例

Robustness前

```
GLubyte tooSmall[NOT_BIG_ENOUGH];  
glReadPixels(0, 0, H, W, GL_RGBA, GL_UNSIGNED_BYTE,  
tooSmall);  
// CRASH
```

Robustness後

```
GLubyte tooSmall[NOT_BIG_ENOUGH];  
glReadnPixels(0, 0, H, W, GL_RGBA, GL_UNSIGNED_BYTE,  
              sizeof tooSmall, tooSmall);  
  
// No CRASH, glGetError() returns INVALID_OPERATION
```

ロバストネス・リセット通知例

- 典型的なリセット確認付きレンダリングループ

```
while (!quit) {  
    DrawStuff();  
    SwapBuffers();  
    if (glGetGraphicsResetStatus() != GL_NO_ERROR) {  
        quit = true;  
    }  
}  
DestroyContext(glrc);
```

MakeCurrentの高速化

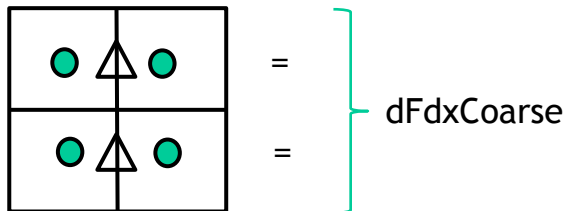
- これまでMakeCurrentコール時に暗示的にglFlushが呼ばれる
 - コンテキストスイッチの性能に影響
- 新しいWGL, GLX 拡張ではglFlushをスキップすることが可能
 - コンテキストキュー内でコマンドは待ちとなる
 - アプリはよりFlushを細かく制御可能
- **MakeCurrent** により性能改善が可能

```
StartTimer();  
for (int i = 0; i < iterations; ++i) {  
    DrawSimpleTriangle();  
    wglMakeCurrent(context[i % 2]);  
}  
StopTimer();
```

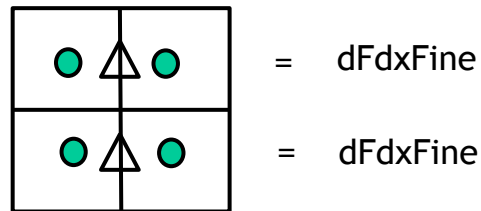
GLSL derivative Control

- Derivative演算の精度に関するヴァリアント”coarse”/”fine”をGLSLに追加
- `dFdxCoarse`, `dFdyCoarse`
 - 高速に動く
- `dFdxFine`, `dFdyFine`
 - 正確な演算が行われる
 - Default デフォルトは昔の`dFdx`/`dFdy`関数
- `fwidthCoarse`及び`fwidthFine`も追加

2x2 Quad Fragment



2x2 Quad Fragment



ARB_sparse_buffer

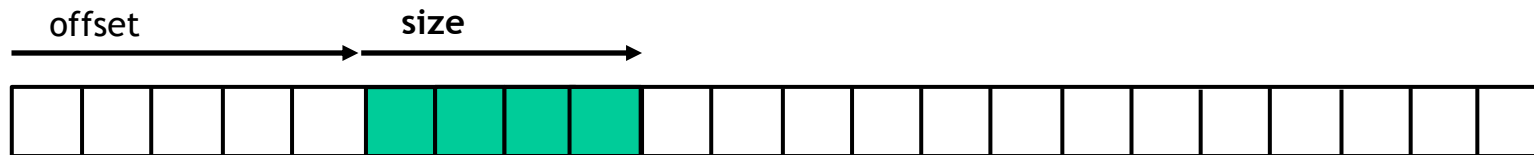
- 仮想的に大きなバッファをもつけど、すべてメモリアロケーションされるわけではない
 - [ARB_sparse_texture](#) のbuffer版
 - アプリはページで制御する

Create uncommitted buffer: `glBufferStorage(,SPARSE_STORAGE_BIT_ARB)`




`GL_SPARSE_BUFFER_PAGE_SIZE_ARB`

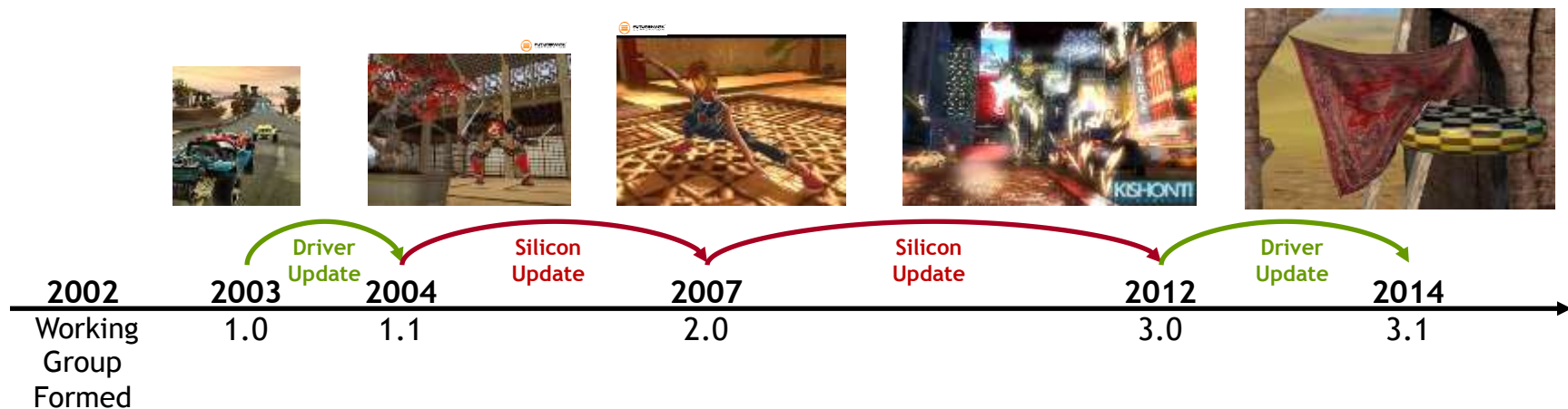
Make pages resident: `glBufferPageCommitmentARB(, offset, size, GL_TRUE);`



OpenGL ES アップデート



これまでの歩み



- 今年のアップデート
 - GDCでOpenGL ES 3.1を発表
 - 認証テストのアップグレード

OpenGL ES 3.1で目指したこと

- OpenGL 4.xの適切な機能をモバイルで使える様にする
 - 目玉機能: Compute shaders, Indirect drawing
 - その他、進化した機能セット、プログラミングスタイルの改善
- 急速な普及を実現する
 - 現状のデバイスに隠れている機能を公開
 - ES 2.0/3.0と後位互換
 - OpenGL ES 3.0のハードウェアで実行可能
- アプリケーションのポータビリティを改善
 - 未定義な動作を最小化
 - より厳しい認証テストを提供



OpenGL ES 3.1

- Compute shaders (コンピュートシェーダ)
 - グラフィックスとGPUコンピューティングの融合
- Draw-Indirect
 - 描画コマンドパラメータを引数からではなくメモリから直接読み出す事が可能
- 個別のShader Object (SSO)
 - グラフィックスシェーダのミックス及びマッチ
- 新しいシェーダ言語機能
- 新しいテクスチャ機能



Compute Shaderとは

- GPUの性能を一般的なコンピューティング向けに使用
 - グラフィックスとは別な新しいシングルパイプラインステージ
- OpenGL ES グラフィックスパイプライン
 - HWが勝手に頂点・フラグメント毎のスレッドを作成
 - すべてのスレッドの入出力データ、同期化はHW側で自動で行う
- OpenGL ES コンピュートパイプライン
 - 整数格子状のポイント毎に明示的にスレッドを生成
 - データ入力、結果出力、同期化をすべて明示的に制御



Image processing



Wave Simulation



AI Simulation



Global Illumination



Ray Tracing

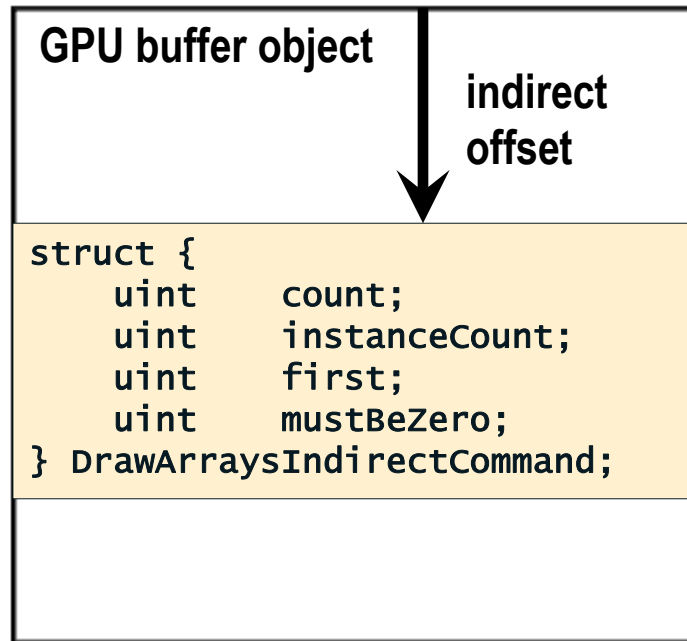
Compute Shader コンポーネント

- 新しいシェーダタイプ及びディスパッチコール
 - `glDispatchCompute()`
- シェーダストレージオブジェクト (SSBOs)
 - 書き込み可能なUBOで、最小の最大サイズは128MB
 - Load/Store/Atomicsで明示的にアクセス
- イメージ
 - 読み書き可能なテクスチャの様なモノ
 - すべての実装は少なくとも四つのイメージバインディングを提供
 - イメージアトミックも拡張として使用可能
- シリアライゼーション、シンクロナイズেশョンプリミティブ
 - Atomic counters
 - Shader Memory barriers
 - API Memory barriers

Draw-Indirect

- GPUにワークを割り当てる新たな方法
 - 描画コールパラメータをGPUメモリに格納
 - `glDrawArraysIndirect()`
 - `glDrawElementsIndirect()`
 - `glDispatchComputeIndirect()`
- GPUが描画パラメータをセットする場合に有効
 - 例えばコンピュートシェーダ
 - CPU/GPUの同期が不要

`glDrawArraysIndirect(mode, indirect);`



Shading Languageの改善

- 配列の配列

- `int a[2][3][4];`

- 明示的なuniform location

- `layout(location=0) in vec4 position; // vertex shader example`
 - `layout(location=2) out vec4 diffuse; // fragment shader example`

- Shaderビットフィールド操作

- Extract, Insert, Reverse, pack/unpack, Count, LSB, MSB,
 - Extended precision support
 - frexp / ldexp

- Shader helper invocation

- `bool gl_HelperInvocation`

Reference compiler

- GLSL ES 3.1を新たにサポート
- Compute shaders
 - (minus arrays of arrays, coming soon)
- Images
- Atomic Counters
- SSBOs
- Texture Gather
- Texture LOD
- SSO

など

test.comp

```
#version 310 es
```

```
layout(local_size_z = 1000) in;
```

```
void main()  
{  
    barrier();  
}
```

```
$ glslangValidator test.comp
```

```
ERROR: 0:3: 'local_size' : too large; see gl_MaxComputeWorkGroupSize
```

```
ERROR: 1 compilation errors. No code generated.
```

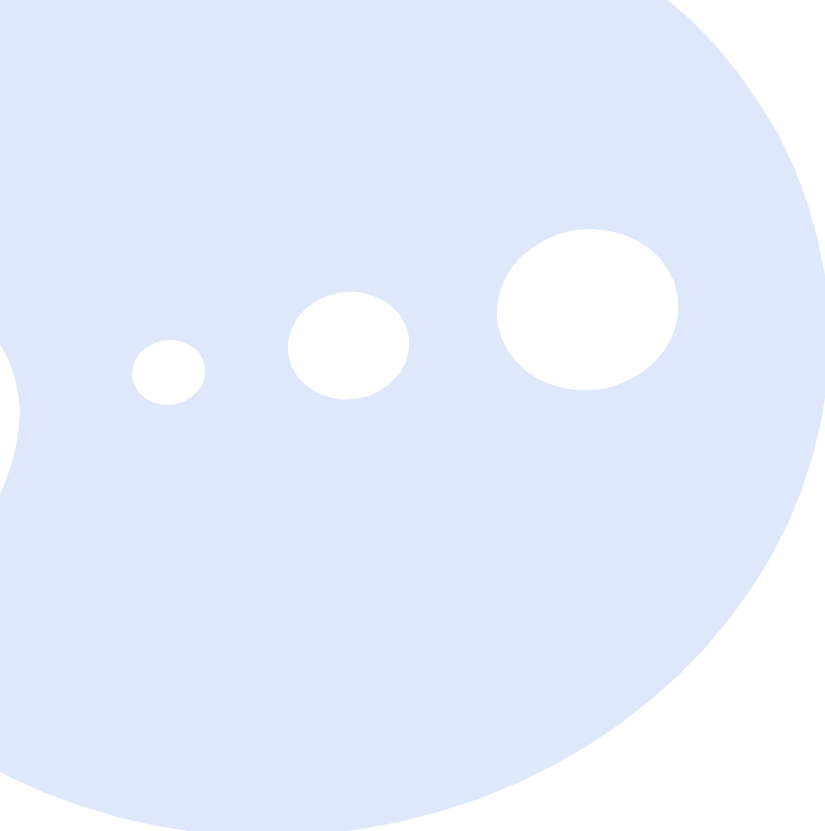
See: <http://www.khronos.org/opengles/sdk/tools/Reference-Compiler/>

Google Android Extension Pack (AEP)

- Android L mandates OpenGL ES 3.1
 - On every device
- Android Extension Pack (AEP)
 - Set of OpenGL ES 3.1 extensions accessible through a single query
- AEP includes OpenGL AAA gaming functionality
 - Tessellation
 - Geometry shaders
 - ASTC Texture Compression + more...
 - https://www.khronos.org/registry/gles/extensions/ANDROID/ANDROID_extension_pack_es31a.txt
- Enables AAA graphics effects
 - Global Illumination
 - Physically-based shading
 - Deferred rendering
 - HDR tone mapping
 - Smoke and particle effects

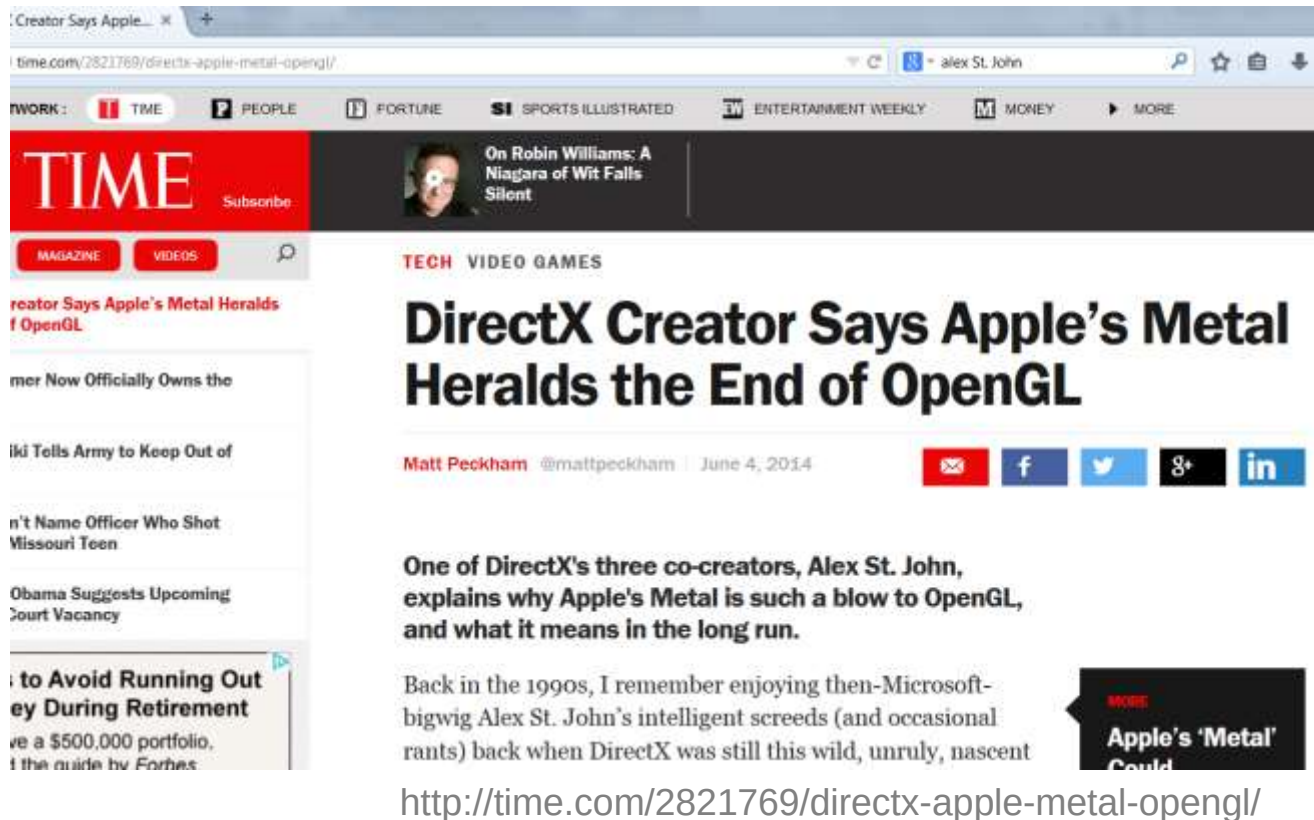


Epic's Rivalry demo using full Unreal Engine 4
Running in real-time on NVIDIA Tegra K1 with OpenGL ES 3.1 + AEP
<https://www.youtube.com/watch?v=jRr-G95GdaM>



次世代 OpenGL/OpenGL ES

OpenGL is a *huge* success



The screenshot shows a web browser displaying a Time magazine article. The browser's address bar shows the URL `time.com/2821769/directx-apple-metal-opengl/`. The page header includes the Time logo and navigation links for various categories like TIME, PEOPLE, FORTUNE, etc. The article title is 'DirectX Creator Says Apple's Metal Heralds the End of OpenGL' by Matt Peckham, dated June 4, 2014. The article text begins with 'One of DirectX's three co-creators, Alex St. John, explains why Apple's Metal is such a blow to OpenGL, and what it means in the long run.' A small video player is visible on the right side of the article.

Creator Says Apple's Metal Heralds the End of OpenGL

DirectX Creator Says Apple's Metal Heralds the End of OpenGL

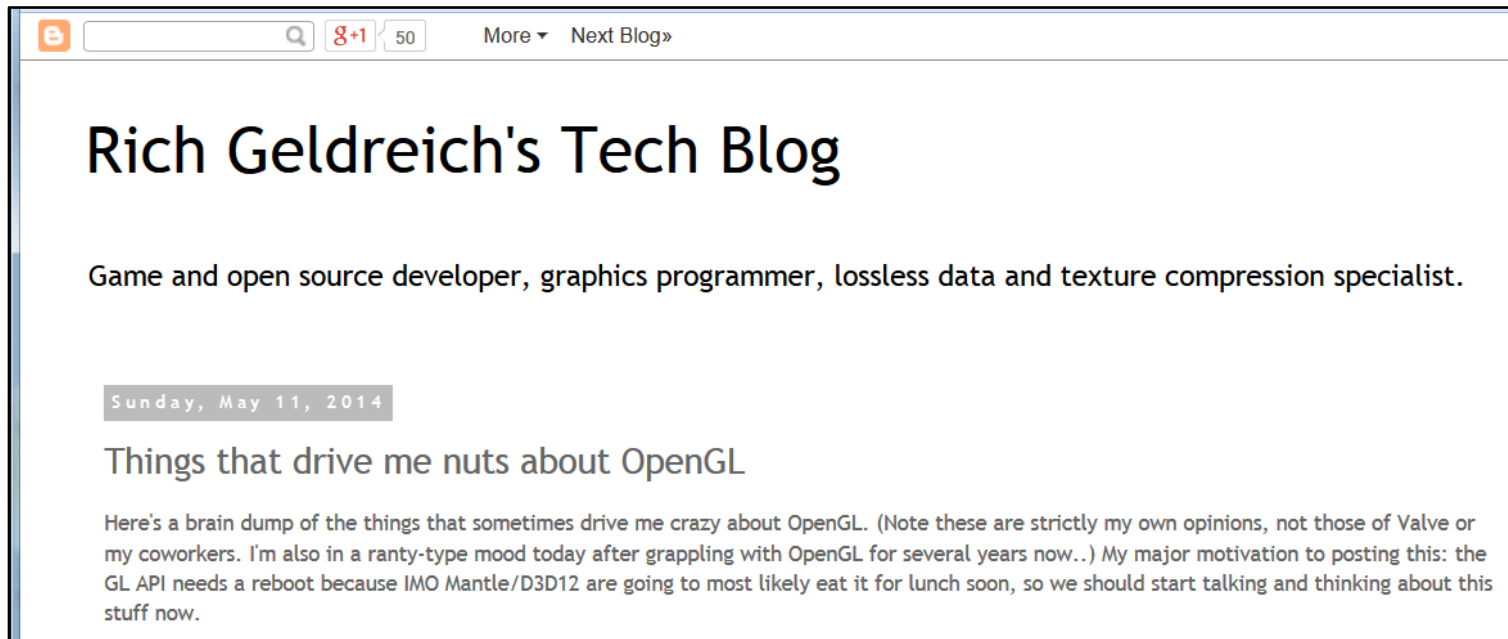
Matt Peckham @mattpeckham June 4, 2014

One of DirectX's three co-creators, Alex St. John, explains why Apple's Metal is such a blow to OpenGL, and what it means in the long run.

Back in the 1990s, I remember enjoying then-Microsoft bigwig Alex St. John's intelligent screeds (and occasional rants) back when DirectX was still this wild, unruly, nascent

<http://time.com/2821769/directx-apple-metal-opengl/>

OpenGL needs to change



<http://richg42.blogspot.com/2014/05/things-that-drive-me-nuts-about-opengl.html>

現状OpenGLの課題

- プログラミングモデルが近年のGPUのアーキテクチャに最適化されていない
 - 特にモバイルで
- 不可解かつ古風な文法
 - あまりよくない20年モノのレガシー
 - 無駄に複雑
- CPUの負荷が高い
 - ステートの検証, 依存関係トラック, エラーハンドリング
 - 多くの実装にてCPUのワークロードについて予測が難しい
- マルチコアのCPUアーキテクチャにあわない
 - プリミティブスレッドモデル、一貫性のない実装
- 多くの実装バリエーション
 - ルーズな仕様, 性能特性の差, ドライババグ

結論：正常進化では解決できない

<https://www.flickr.com/photos/noaaphotolib/5015135752/in/photostream/>



<http://www.tankgirl.info/tankgirl/images/flick/tradingCards/89f.jpg>



- GLはすばらしい適応進化をしてきた実績がある
- ただし、構造的な問題は通常機能進化では対応できない。

次世代OpenGLに関する新提案

- 近代オープンスタンダード、クロスプラットフォーム3D+Compute API ...
 - OpenGLとの互換性をなくす
 - 1から原理・原則からのスタート
- ゴール
 - クリーンでモダンなアーキテクチャ
 - マルチスレッド、マルチコアフレンドリー
 - CPUのオーバーヘッドをものすごく減らす
 - 中立的なアーキテクチャ - イミディエートレンダリングも、タイルベースアーキテクチャも両サポート
 - 予測可能な性能
 - 各実装における信頼性、一貫性の改善
- キーとなる設計原理: 明示的な制御(*explicit control*)
 - アプリはドライバが何を期待しているか理解できる

ステータス

- ARB/OpenGL ESワーキンググループのジョイントプロジェクトとして組織
 - Chair: Tom Olson (ARM)
 - Language / IL group chair: Bill Licea-Kane (Qualcomm)
 - API spec editors: Graham Sellers (AMD) and Jeff Bolz (NVIDIA)
- 今年の6月から議論をスタート
 - メンバーから様々な重要な提案、IP提供がすでに行われている
- シェーダのポータブルな中間言語(IL)を採用する方針
 - ISV(Independent Software Vender)から昔から来ていたリクエスト
 - シェーダの信頼性、ポータビリティの改善可能
 - 機械が生成したシェーダと特殊な高級言語向けによりターゲット
 - シェーダのIP保護の機構を提供