OpenGL ES 3.0最新情報とOpenGL 20周年

株式会社ディジタルメディアプロフェッショナル
大渕　栄作

03/Dec/2012

Agenda

- OpenGL ES 3.0
- OpenGL 20周年

DMP概要

- 会社名：（株）ディジタルメディアプロフェッショナル
  （略称　DMP）　www.dmprof.com
- 事業内容：
  - 2D/3Dグラフィックス IPのライセンス
  - グラフィックスプロセッサ開発・販売
  - グラフィックス関連ソフトウエア製品の開発・販売
  - 3Dグラフィックスに関する技術コンサルティング
- 所在地：東京都武蔵野市　JR三鷹駅から徒歩2分
- 会社設立：2002年7月
- 特許等：日米欧において58件の特許出願済み成立30件
  （2012年6月現在）

DMP グラフィックスIPソリューション

■ 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）

標準VG,3Dグラフィックス IPコア
SMAPH-H (OpenGL ES 1.1, OpenVG1.1)
SMAPH-H2 (OpenGL ES 2.0, OpenVG1.1)

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

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

OpenVG 1.1対応
ベクターグラフィックス IPコア
SMAPH-F

Khronosアップデート

Khronosアップデート

- **OpenGL ES 3.0 API仕様をリリース**
  - モバイル3Dグラフィックス APIの6年ぶりメジャーアップデートリリース
- **OpenGL 4.3 API仕様リリース**
  - Compute shader機能の追加等
- **OpenGL及び、OpenGL ES向け共通テクスチャ圧縮フォーマット**
  - ETC2をコア拡張、ASTCを拡張APIとして提供
- **OpenCL向けCLUユーティリティオープンソースプロジェクト**
  - OpenCLプログラミングの簡素化を実現
- **COLLADA標準のISO標準化**
  - OpenCOLLADAオープンソースインポータ・エキスポータ及び、認証テストスイート
- **ビジョン系処理アクセラレーション及び、各種センサーデバイス融合向け新ワーキンググループ**
  - OpenVLとStreamInputの新ワーキンググループ

ソフトウェアからシリコンへのブリッジ

- **Khronos APIがプロセッサアクセラレーションの可能性を定義**
  - グラフィクス、ビデオ、オーディオ、コンピューティング、ビジョン、センサー分野

OpenGL ES 3.0

これまでの歩み

- **SIGGRAPH 2002**
  - OpenGL ESワーキンググループ立ち上げ
- **SIGGRAPH 2003**
  - OpenGL ES 1.0 API仕様リリース
- **SIGGRAPH 2004**
  - OpenGL ES 1.1 API仕様リリース
- **GDC 2007**
  - OpenGL ES 2.0 API仕様リリース

これまでの歩み(続き)

- **2007/8**
  - OpenGL ES 1.x 対応デバイス発売

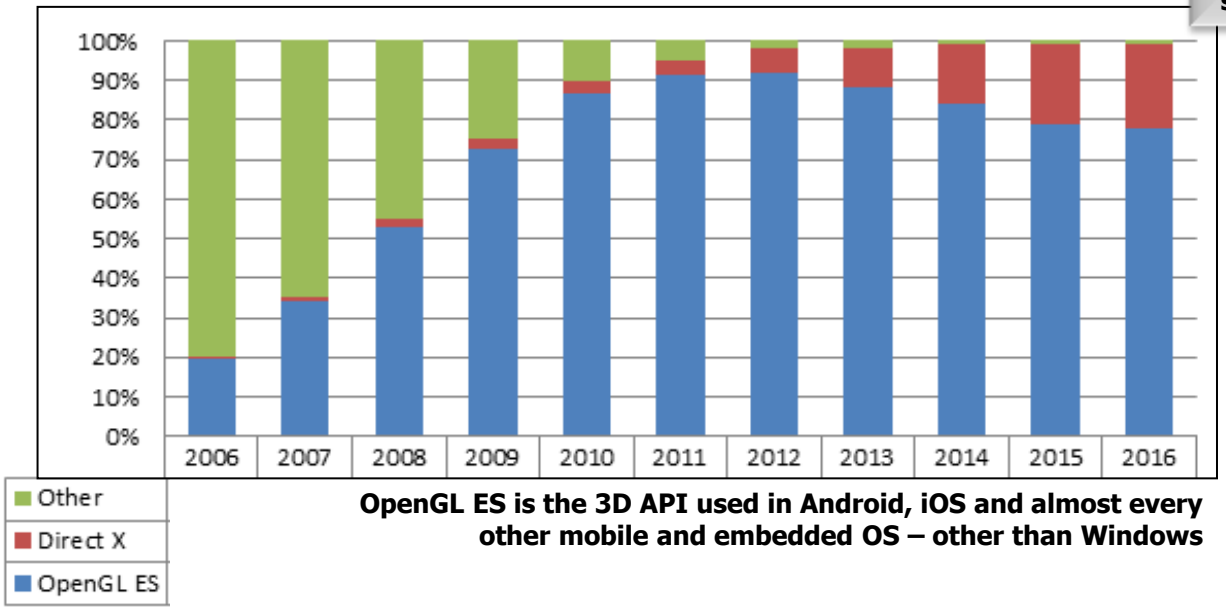
- **2009/10**
  - OpenGL ES 2.0 対応デバイス発売

- **2011:　ハイエンド系3Dコンテンツがモバイル上で動作**

モバイルにおけるOpenGL ES採用状況

Use of 3D APIs in Mobile Devices

Source: Jon Peddie Research



On PC – DirectX is used for most apps. On mobile – the situation is reversed

OpenGL ES is the 3D API used in Android, iOS and almost every other mobile and embedded OS – other than Windows

OpenGL ES 3.0の目標

- **近年のコンテンツで必要とされる機能の提供**
  - OpenGL 3.3 / 4.xをベースにした機能セット
  - 必須な拡張の削減
- **プログラマーに優しいAPI**
  - サポート機能に対して、要求定義の厳格化
  - 実装依存部分による差異の削減
- **OpenGL ES 2.0エコシステムの活用**
  - OpenGL ES 2.0との互換性を保持

- **ゲームなどのアプリケーション向けに、よりよいAPI, 高速なAPIを目指す**

プログラマに優しいAPI

- **拡張されたシェーディング言語：GLSL ES 3.0**
  - よりOpenGLと互換性を確保するために文法、機能のアップデート
  - 32-bit精度の整数、浮動小数点型のフルサポート
  - OpenGL ES 2.0であった多くの制限を削除
- **予測可能な機能、挙動**
  - 様々なテクスチャ、レンダーバッファフォーマット
  - 各フォーマット仕様の明確化（バッファサイズなど）
  - エラー時の挙動を明確化

シェーダ機能について

- **GLSL v2.0機能を包含**
- **シェーダプロセッサの要件**
  - 頂点、フラグメントシェーダいずれも
    - 浮動小数点型 (IEEE 32 bit)
    - 整数型 (32 bit signed and unsigned)
- **ユニフォームバッファオブジェクト**
  - 12 Vertex + 12 Fragment binding points, 16KB
- **トランスフォームフィードバックバッファ**
- **マルチビューバッファ**
- **テクスチャサンプラ**
  - sampler3D, sampler2DArray, sampler2DShadow, samplerCubeShadow
  - integer and unsigned integer samplers
- **Appenix Aに定義されていた制限の撤廃**

シェーダ機能について

- **OpenGL 3.3のサブセット**
  - ジオメトリシェーダは未サポート
- **いくつかの機能をOpenGL 4.xから追加**
- **OpenGL ES 2.0機能**
  - 例 – precision qualifiers

ポーティング

これまでの記述

```
#version 100 // implicit
precision mediump float;

varying vec4 color;



void main()
{
    gl_FragColor = color;
}
```

組み込み varyingの一般化

GLSL ES 3.0での記述

```
#version 300 es // required
precision mediump float;

in vec4 color;

layout(location = 0) out vec4 data;

void main()
{
    data = color;
}
```

マルチレンダーターゲットの指定

サンプラーに対する精度定義

```
#version 300 es
precision highp float;

sampler2D floatMap;// lowp
sampler3D cubeMap; //
error
```

```
#version 300 es
precision highp float;

precision highp sampler2D floatMap;
precision lowp sampler3D cubeMap;
```

この記述はエラーとなる

Fragment shaderについて、floatについては以前から精度指定が必要だったが、GLSL ES 3.0からsamplerに対しても精度指定が必要

テクスチャ機能の拡張

- 32枚のテクスチャサポート (16V + 16F)
- テクスチャサイズ、2のべき乗制限の撤廃
- 豊富な内部フォーマット
- 1成分, 2成分テクスチャのサポート (R, RG)
- 3Dテクスチャ、テクスチャアレイの標準サポート
- シャドウ比較付きデプステクスチャ

描画関連

- **プリミティブリスタート**
  - 複数トライアングルストリップを1つのコマンドで描画
- **インスタントレンダリング**
  - オブジェクトのコピーを1つのコマンドで複数描画
- **トランスフォームフィードバック**
  - ジオメトリ変換後のデータをバッファに出力可能

レンダリング関連

- **4つまで複数レンダーターゲットをサポート**
  - ディファードレンダリングに適用可能
- **マルチサンプル・フレームバッファオブジェクト**
  - テクスチャに対するアンチエイリアスレンダリング
- **オクルージョンクエリ**
  - Visibleなピクセルがターゲットのオブジェクトで書かれたかの情報が格納されるクエリ
  - 投入不要なオブジェクトを判定するために使用

OpenGL･OpenGL ES共通拡張

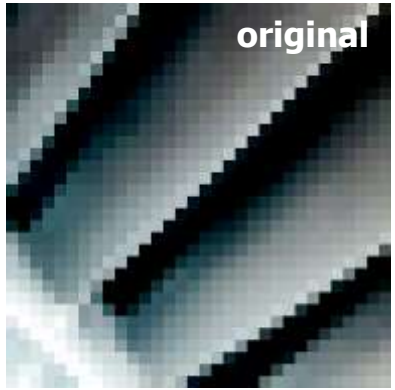
- **KHR拡張タグ**
  - OpenGL及びOpenGL ESのワーキングループが一緒に設計、承認した拡張
- **KHR_debug**
  - ４つのARBデバグ拡張を合わせたもの
  - コールバックまたはログ経由でデバグ情報を出力
  - オブジェクト･イベントに関連するメッセージを示すラベル、マーカー
- **ARB_robustness_isolation**
  - GPUをリセットする必要があった場合、他のアプリケーションに影響しないことを要求する拡張

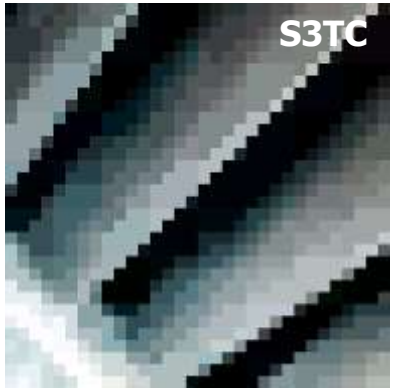
テクスチャ圧縮

- **テクスチャ圧縮を有効的に使うことで、使用リソースを削減**
  - アプリサイズを小さくすることで、ネットワークのバンド幅、プログラム自体のメモリフットプリントの削減
  - GPUからメインメモリへのメモリアクセス量を削減
- **開発者はどの環境でもおなじテクスチャ圧縮フォーマットを使いたい**
  - もし共通フォーマットがないと、例えばWebGLアプリで複数のテクスチャフォーマットを持つ必要がある。
  - マルチプラットフォーム向け開発では最大の懸念事項になりつつある。
  - α付き1,2bit/pixelの圧縮フォーマットの要望も大きい
- **Khronosではこの問題の積極的な解決を推進**
  - ETC2/EAC はOpenGL ES 3.0とOpenGL 4.3 で必須サポート
  - ASTCフォーマットのサポートを拡張フォーマットとしてリリース

ETC2 / EAC テクスチャ圧縮

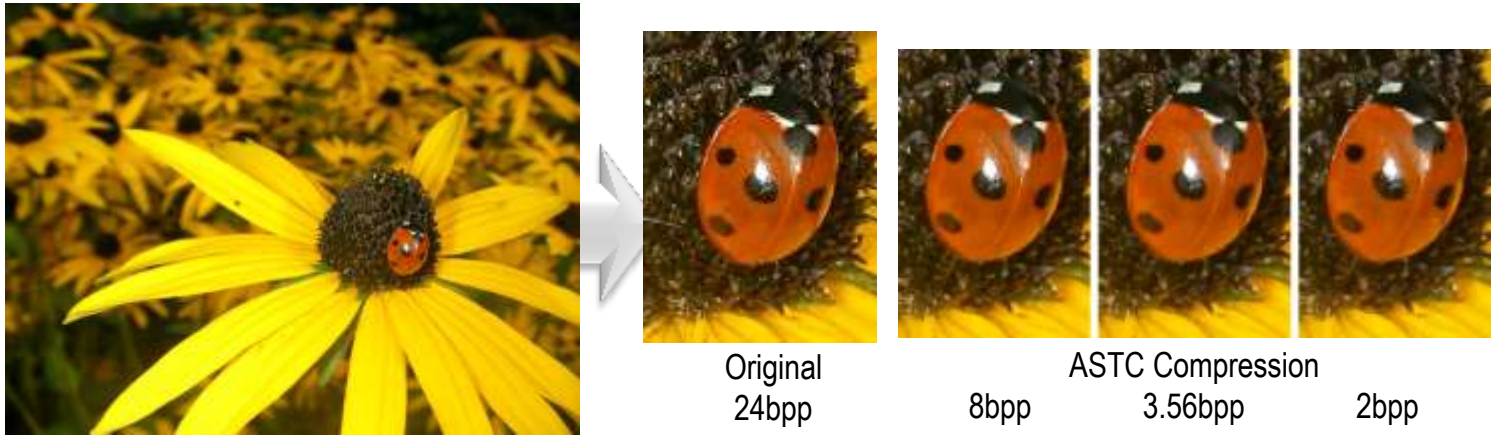
- **OpenGL と OpenGL ESで必須の拡張**
  - モバイル、デスクトップの両方でサポート
- **RGB向け圧縮ETC2**
  - 4bpp, 3成分
  - Linear/sRGB 色空間のサポート
- **R/RG成分向け圧縮EAC**
  - 4-bpp, 1成分
  - 8-bpp, 2成分
- **これらを組み合わせRGBAフォーマットをサポート**
  - 8-bpp, 4-channel
- **画像品質の改善を実施 (ETC→ETC2)**
  - Thanks, Ericsson!





ASTC テクスチャ圧縮

- **Adaptive Scalable Texture Compression (ASTC)**
  - 同ビットレートでS3TC, PVRTCよりも高画質
- **フレキシブルな圧縮レート、フォーマットサポート**
  - 1 から 4 色成分フォーマット: R / RG / RGB / RGBA
  - ビットレート: 8 bppから <1 bpp を細かいステップで定義可能
- **ASTCはロイヤリティフリーで提供**
  - 拡張として提供することで業界からフォードバックを受けながら改善を行っていく



ASTC と PVRTCの比較

- **LDR 拡張と現在提供**
  - HDR 拡張は開発中

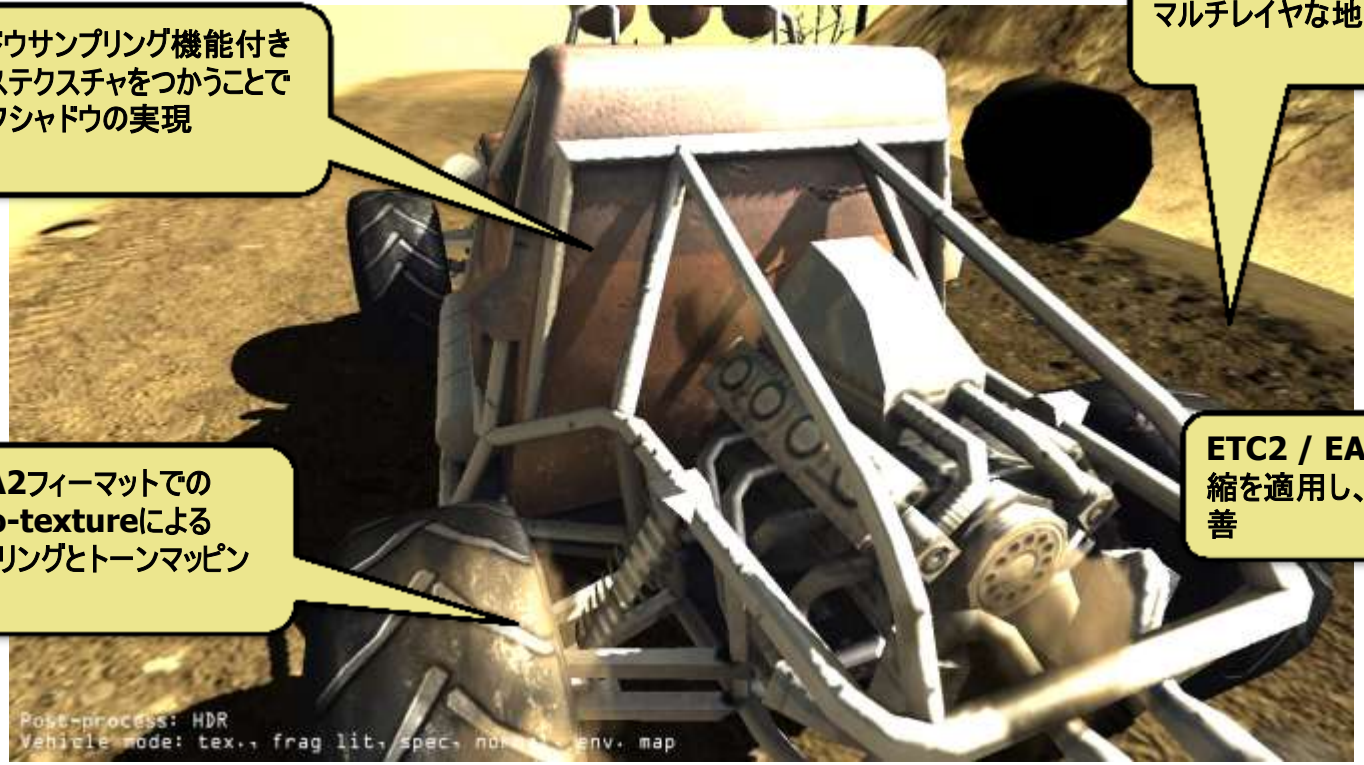
Block Size	Bits Per Pixel	Comp. Ratio
4x4	8.00	4:1
5x4	6.40	5:1
5x5	5.12	6.25:1
6x5	4.27	7.5:1
6x6	3.56	9:1
8x5	3.20	10:1
8x6	2.67	12:1
10x5	2.56	12.5:1
10x6	2.13	15:1
8x8	2.00	16:1
10x8	1.60	20:1
10x10	1.28	25:1
12x10	1.07	30:1
12x12	0.89	36:1

ORIGINAL IMAGE

COMPRESSED WITH PVRTC 2BPP

COMPRESSED WITH ASTC 2BPP

OpenGL ES 3.0で出来る表現



シャドウサンプリング機能付きデプステクスチャをつかうことでセルフシャドウの実現

2Dテクスチャアレイを用い、マルチレイヤな地面を表現

RGB10_A2フィーマットでのrender-to-textureによるHDRレンダリングとトーンマッピング

ETC2 / EAC テクスチャ圧縮を適用し、描画性能の改善

Content: ARM “Timbuktu 2” tech demo

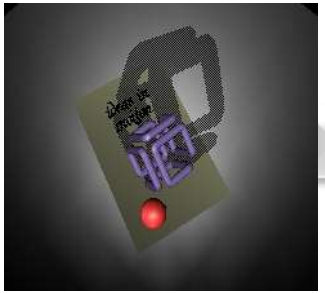
Kishonti GLBenchmark 3.0

Kishonti "GLBenchmark 3.0" preliminary

OpenGL 20周年

OpenGLの20年 - Then and Now

Ideas in Motion - *SGI*

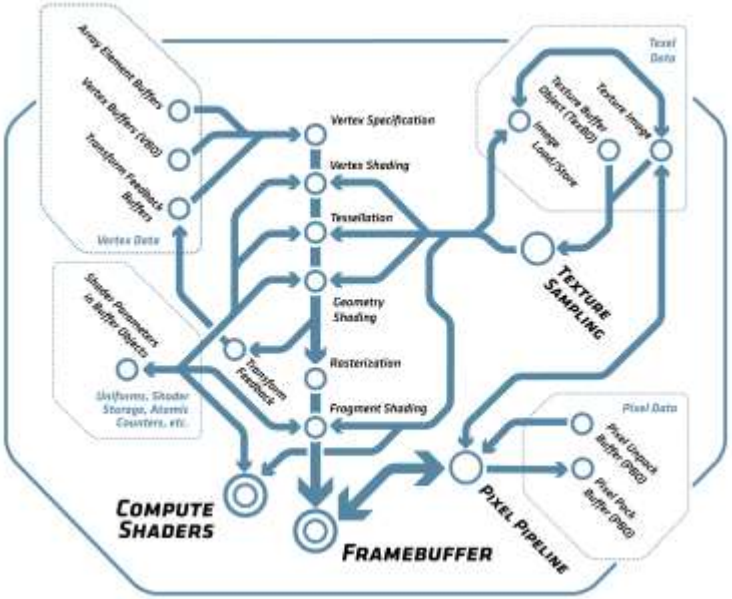


Rage - *id Software*

	1992 Reality Engine 8 Geometry Engines 4 Raster Manager boards	2012 Mobile NVIDIA Tegra 3 Nexus 7 Android Tablet	2012 PC NVIDIA GeForce GTX 680 Kepler GK104
Triangles / sec (millions)	1	103 (x103)	1800 (x1800)
Pixel Fragments / sec (millions)	240	1040 (x4.3)	14,400 (x60)
GigaFLOPS	0.64	15.6 (x25)	3090 (x4830)

1.5KW <5W

OpenGLによるイノベーションの歴史



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

OpenGL 4.3

OpenGL 4.2

OpenGL 4.1

OpenGL 3.3/4.0

OpenGL 3.2

OpenGL 3.1

OpenGL 3.0

OpenGL 2.0

OpenGL 2.1

2004 2005 2006 2007 2008 2009 2010 2011 2012

DirectX 9.0c DirectX 10.0 DirectX 10.1 DirectX 11 DirectX 11.1

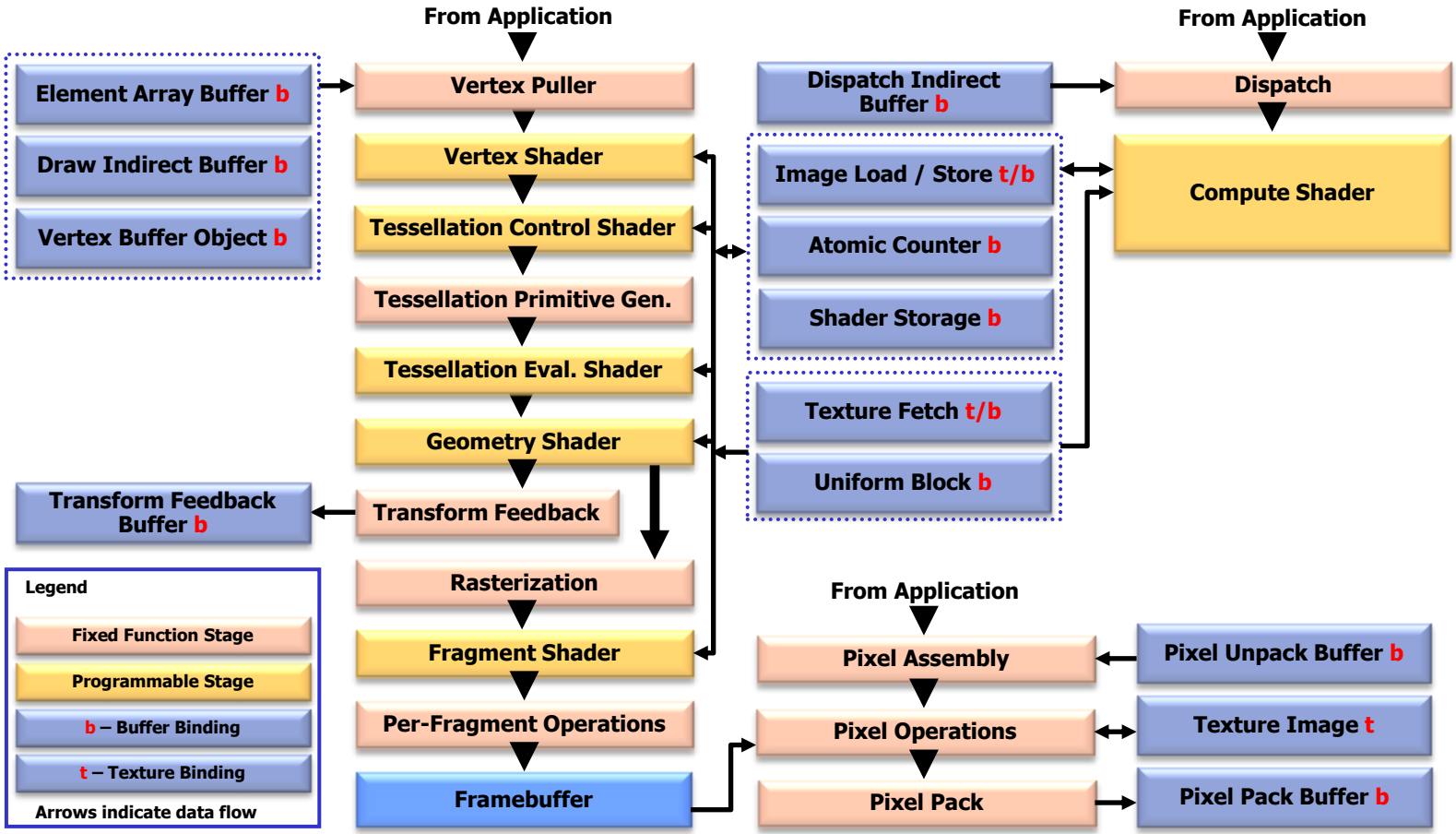
OpenGL 4.3で新しくなったこと

- **texture functionality**
  - ARB_texture_view
  - ARB_internalformat_query2
  - ARB_copy_image
  - ARB_texture_buffer_range
  - ARB_stencil_texturing
  - ARB_texture_storage_multisample
- **buffer functionality**
  - ARB_shader_storage_buffer_object
  - ARB_invalidate_subdata
  - ARB_clear_buffer_object
  - ARB_vertex_attrib_binding
  - ARB_robust_buffer_access_behavior

OpenGL 4.3で新しくなったこと

- **pipeline functionality**
  - ARB_compute_shader
  - ARB_multi_draw_indirect
  - KHR_debug
  - ARB_program_interface_query
  - ARB_ES3_compatibility
- **extensions**
  - KHR_texture_compression_astc_ldr
  - ARB_robustness_isolation
- **GLSL 4.3 functionality**
  - ARB_shader_image_size
  - ARB_explicit_uniform_location
  - ARB_texture_query_levels
  - ARB_arrays_of_arrays
  - ARB_fragment_layer_viewport

OpenGL 4.3 パイプライン



コンピュート・シェーダ (Compute shader)

- **GLSLシェーダを一般アルゴリズム実行に使用**
  - バッファ、イメージ、テクスチャをデータの入出力に使用
- **グラフィックスパイプライン上でグラフィックスデータを処理**
  - ピクセルに近いところでの処理をするのであればcompute APIを使うより簡単に実装できるはず
- **OpenCLを補完**
  - 完全にANSI Cベースヘテロジニアスプログラミングのフレームワークをサポートするわけではない
- **OpenGL 4.3標準内でサポート**
  - 機能セットしては、DirectX 11に近い

Image processing

AI Simulation

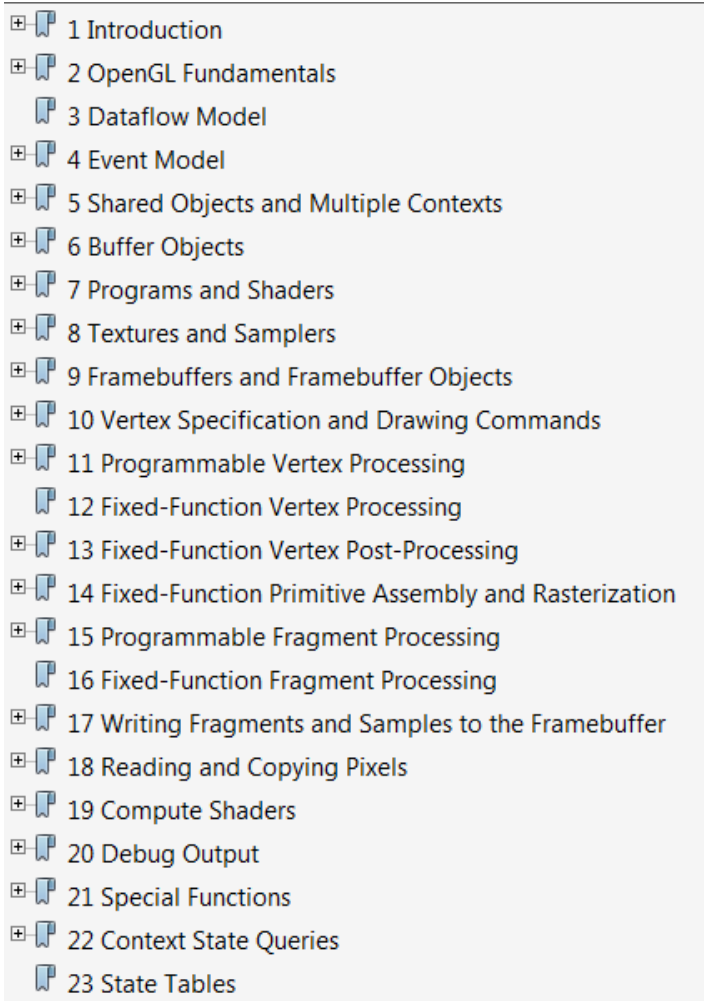
Ray Tracing

Wave Simulation

Global Illumination

仕様書の改善

- シェーダ、バッファセントリックな章立てに変更
  - 固定機能については代替機能として記述
- ハイレベルなコンセプト・オブジェクトを紹介
  - 章立てのはじめの方で登場
- コマンド向けエラーリストのサマリー
- 言語の重複を削除
- フレーズ、用語の一貫性
- 章番号をコアと互換プロファイルの間で合致させる



OpenGL アップデートまとめ

- **OpenGL 4.3 のリリース**
  - Compute shaders
  - Advanced buffer management
  - Advanced texture management
  - Advanced GPU work creation
- **OpenGLの展開**
  - WebGL
  - Mobile platforms
  - Linux
- **OpenGLの20周年**

Thank you!