モバイルグラフィックスAPI アップデート

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

4/Nov/2011

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

（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 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 (OpenGLES 1.1)
SMAPH-S (OpenGLES 2.0)

OpenVG 1.1対応
ベクターグラフィックス IPコア
SMAPH-F

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

Khronos公認OpenGLESトレーニングコース

- 会場
  - 株式会社ディジタルメディアプロフェッショナル
    （JR三鷹駅徒歩2分）セミナールーム
- 開催スケジュール
  - OpenGL ESプログラミング・トレーニング Ⅰ
    - 2012年1月12日（木）～13日（金） 10:00～17:00
  - OpenGL ESプログラミング・トレーニング Ⅱ
    - 2012年1月19日（木）～20日（金） 10:00～17:00
  - GLSLシェーダプログラミング　基礎コース
    - 2011年12月21日（水）～22日（木）　10:00～17:00
- 詳細は - www.dmprof.com または
  http://www.khronos.org/news/events/をご覧ください

Android 3Dグラフィックス・ラーニングキット

- Android上で、組込み用途向け標準3DグラフィックスAPIであるOpenGL ESを使用したプログラミングの基礎を学習します。
  - キット名: Android 3Dグラフィックス・ラーニングキット
  - 利用料金: 4,998円（税込）
  - 利用期間 : 1年間
  - ページ数: 231ページ
  - 再生時間: 約4時間

英語版も
販売開始！

Android™
3Dグラフィックス
ラーニング・キット

- 詳細は www.dmprof.comをご覧ください。

Agenda

- モバイルグラフィックスAPIの動向
  - OpenGL ES
  - OpenVG
- ポータビリティの高いOpenGL ESアプリ開発tips

OpenGL ESアップデート

2010年のアップデートを振り返る：OpenGL ES 2.0 対応プラットフォームが普及

- すでにメジャーな携帯電話では対応済み
- SDK, 解説本が入手可能
- WebGLなどで活用され始めている。

2011年のアップデート

本格的なコンテンツの登場

- ES2.0向けゲームエンジン対応
  - UE3
  - Unigine
- ゲーム・デモ等

http://www.epicgames.com/technology/epic-citadel

http://www.idsoftware.com/rage-mobile/

http://www.epicgames.com/infinityblade/

プラットフォーム動向

- グラフィックス性能が引き続き改善
  - いくつかのベンダでは前世代に比べて5倍改善

- 2011年の最先端仕様は?
  - WVGA超の解像度
  - Dual core CPU with vector / SIMD float
  - Quad core GPU with OpenGL ES 2.0

ワーキンググループアップデート

- 次世代OpenGL ES
  - ワーキングループでは2009年から次世代OpenGL ESの議論に注力
  - マーケットがレディになった段階でリリース予定
  - ハイエンドコンテンツからの要望を元にAPI設計
- ARB / ES Convergence TSG
  - ロードマップの共有および、非互換部分の最小化の活動を引き続き行っている
- コンフォーマンステストの統合
  - OpenGL ES WGとARBの共同プロジェクトにより開発
  - OpenGL 4.x と次世代OpenGL ESで同じプラットフォームを使用
  - Khronosで最大の投資
  - ポータビリティの改善を促進

OpenVGアップデート
- OpenVG Lite Profile

OpenVG概要

- 2Dベクターグラフィクス向けAPI
  - OpenGLスタイルなプログラミングモデル(オブジェクトモデル等)
- デバイス非依存、ベンダー中立なポータブルなオープン標準
  - ハードウェア実装によるアクセラレーション、低消費電力を実現が可能
- 高品位なグラフィクスを提供
  - ユーザインターフェイス
  - テキスト
  - 地図描画
  - SVG, Flash, HTML5 Canvas
  - PDF, Postscript, Java (JSR 287), …

OpenVG Lite Profileについて

- OpenVG 1.1のサブセット
  - 機能の削減、制限の追加を行ったもの
  - ただし、残っている機能について仕様変更はなし。

- Lite profile向けのアプリは1.1で動かす際変更は不要

なぜ、Lite profileを議論しているか

- VG on OpenGLを実現したい
  - VGの普及を促進するために既存OpenGL上で動作するようにしたい
  - この際、OpenGLのハードウェアアクセラレーションを生かせる
  - 様々なデバイス間でOpenVGアプリを共有化
    - 理論上OpenGLスタックの上にOpenVGの機能をライブラリにより実装可能
  - PCでOpenVGをサポートする最短のパス

なぜ、Lite profileを議論しているか

- ただし、OpenVGのフル実装をOpenGLで実現するのは結構大変
  - 機能上、フォーマット上、性能上の問題
  - VG on OpenGLの大きな障害
- そこで、Lite profileで、あまり使われていないかつ、VG on OpenGLで実現が困難な機能を削除
  - 実装のシンプル化によりOpenGL機能でLite profile全機能の実現が可能
  - 性能上のメリットも様々なプラットフォームで得られる

ステータス

- 現在WGでは詳細仕様を議論中
  - プロファイル仕様の策定
  - テスト実装による問題点の洗い出し

ポータビリティの高い OpenGL ESアプリ開発tips

PortableなOpenGLES2.0アプリ作成tips

- 課題
  - OpenGLES2.0での移植性の問題点とは
  - この問題に対する、最良の解はなにか

- 注意
  - これらは仕様を一側面から見たtipsでKhronosオフィシャルなものではありません。
  - 本tipsは初心者向けガイドです。

OpenGL ES 2.0のポータビリティ問題?

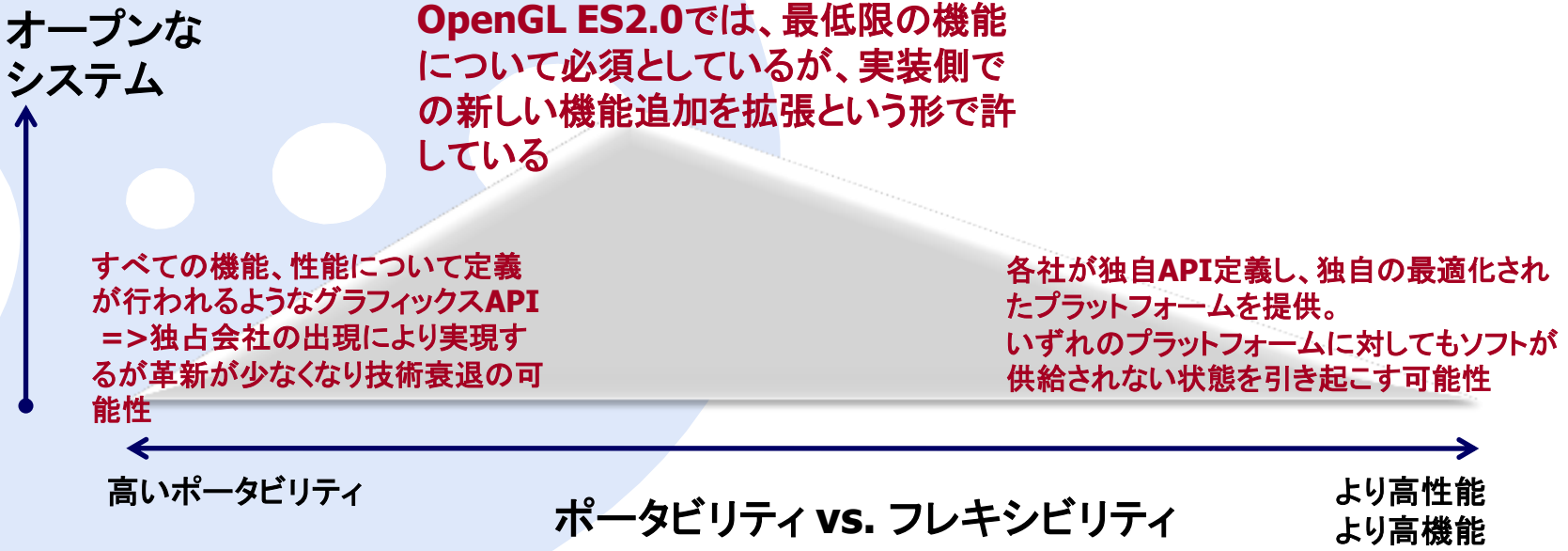
- OpenGL ES 2.0 自体は様々な環境でサポート済み
  - iOS
  - Android
  - WebOS
  - Symbian

  etc

- だけど、実際の開発では...

なぜポータビリティの問題を解決できないか

- グラフィックスAPIの宿命: ポータビリティと実装上の工夫・イノベーションのトレードオフ



互換性を保つための指針

グラフィックスハードウェアは、コア機能を包含する形でOpenGL ES 2.0機能をサポートしている

基本的な推奨

- 互換性を保つようにグラフィックスエンジン・ミドルウェアをデザイン
- その後、実装依存の機能について、開発時間・必要性に応じて追加

OpenGL ES 2.0における互換性問題

- 性能特性
- 実装オプション
  - 実装依存の制約値
  - シェーダの演算精度
  - シェーダ言語の制限
  - エラー時の挙動
- 拡張API
  - テクスチャ圧縮
  - "サイレント"拡張

性能特性

- **基本的な問題**
  - 1ベンダのGPUアーキテクチャ、シリーズ内でも様々な実装バリエーションがある
  - また各アーキテクチャにおいても強み、弱みとなるポイントが違う
  - あるアーキテクチャ向けに最適することで、そのコードがほかのアーキテクチャでは最悪の動きをする可能性もあり

- **Best practices**
  - まずは一般的の最適化に注力する
  - GPUデベロッパが準備している最適化方針のドキュメントを入手し、開発の初期段階でターゲットとなる各社GPUの性能確認、特性理解を行う。
  - エンジンの設計にあたっては、各デバイスの性能の違い、機能の違いを吸収できるような柔軟性を持たせる

実装依存の制約値

- OpenGL ES 2.0仕様内で、実装によって差異がありうるパラメータを定義
  - たとえば、「使える頂点アトリビュート数」など
  - ちなみにこの値はread-onlyのクエリとして提供 => MAX_VERTEX_ATTRIBS
  - 仕様ではすべての実装で最低限サポートしなければいけないパラメータの値を定義
    - » 仕様の Table(s) 6.18-6.21(OpenGL ES 2.0.25 Full Specification)参照

- Best practices
  - 可能であれば仕様のChapter 6内のminimum-maximum値範囲内で実装を行う
  - または、クエリを用いて値を確認し、その値範囲内で実装できるようにする
  - または、ターゲットプラットフォームにおける最大値を確認し、その値の範囲で実装する。

シェーダの演算精度

- GLSL ESでは、変数に対する精度のqualifierが使用可能
  - Lowp (10-bit), mediump (16-bit), highp (24-bit)
- Qualifierにより最低限の精度を特定することができる
  - GPUはここで明示された精度を超える精度を提供
  - `GetShaderPrecisionFormat()` のクエリで実際の精度は確認可能
- ちなみにhighpはフラグメントシェーダではオプション

- Best practices
  - 大きなテクスチャとwrapモードの組み合わせなどの演算精度に気をつける
  - `#precision float mediump` により、フラグメントシェーダの演算精度を明示化する。
    - もしhighpを使った場合、fallbackにて使えない旨が通知される
  - GPUにて本当にhighpが提供されていないかテストする

シェーダ言語の制約

- GLSL ES 1.0 では、コアシェーダ言語に対していくつかの制約を許している
  - 仮想化の未サポート -HWリソース(temporal register)が足らない場合、コンパイルは”out of resoueces”エラーによりfailしてもいい。
  - ループ回数をコンパイル時にわかる記述でなければならない
  - GPUは配列のインデックスの扱いについて制限がある可能性がある(e.g. dynamic indexing)。
    - » コンパイラ時間で分かる範囲でのindexingが実装されている必要がある
  - GLSL ES 1.0 仕様のAppendix Aを参照
- 各社ともできるだけ制約を緩和しているが、この制約に関するクエリはなし

- Best practices
  - 仕様Appendix Aのガイドラインにのっとるかfallbackがある前提で設計をする
  - シェーダ開発のフローにオフラインでのシェーダコンパイラでのシェーダバイナリ生成プロセスを追加する

ポータビリティに影響のある悪いコード

- ガベージポインタの指定 => GPUによって動作が変わる可能性
  - Attribute buffers
- シェーダのエラー
  - Divide by zero
  - NaN 生成 / 伝搬
- API制限
  - TexImage2D type / format / internalformat 制限

拡張API

- 各GPUは新しい機能をextension specを通じて宣伝
  - コア仕様からの差分を定義
  - 例 `OES_texture_npot`
  - GetString[v]()でクエリ
- 分類
  - OES: OpenGL ES WG公認の拡張仕様。認証試験で試験済み
  - EXT: 複数ベンダでサポートされる拡張仕様
  - Vendor (e.g. DMP): そのベンダーでのみサポートされる仕様
- Best Practices
  - できるだけ使わない
  - 使う場合、OES => EXT => Vendorの順で使うことを検討
  - 常に、クエリをしたあとに使う

テクスチャ圧縮

- テクスチャ圧縮の適用により、性能改善・メモリ削減に大きく貢献
  - フォーマット例: ETC1, PVRTC, DXTn, …
- ただ残念ながら
  - 各社で共通にサポートしているフォーマットはない
  - iOSとAndroidの両方をターゲットにした場合、1つの圧縮対象テクスチャに対して少なくとも2つのテクスチャフォーマット生成を行う必要がある
- Best practices
  - iOSではPVRTC (vendor extension)を使う
  - それ以外では
    - » ETC1 (OES extension) を使う
    - » または、その他ベンダ拡張を使う

ETC1テクスチャ

- ETC1はRGBのみで、アルファチャネルはなし
  - ETC1を2枚用いる
    - wrapモードを使っていなければRGBとAイメージを1つのイメージにマージする
  - または、ETC1をRGB向けにAをほかのテクスチャとして定義する
    - たとえば LA88 に height map と alphaを入れちゃう
- 法線はどうするか
  - ETC1内の2カラーコンポーネントに入れる
- 一般にETC1は輝度成分の解像度が高い
  - R/G/B成分の傾向が同じ場合よい画質が得られる傾向

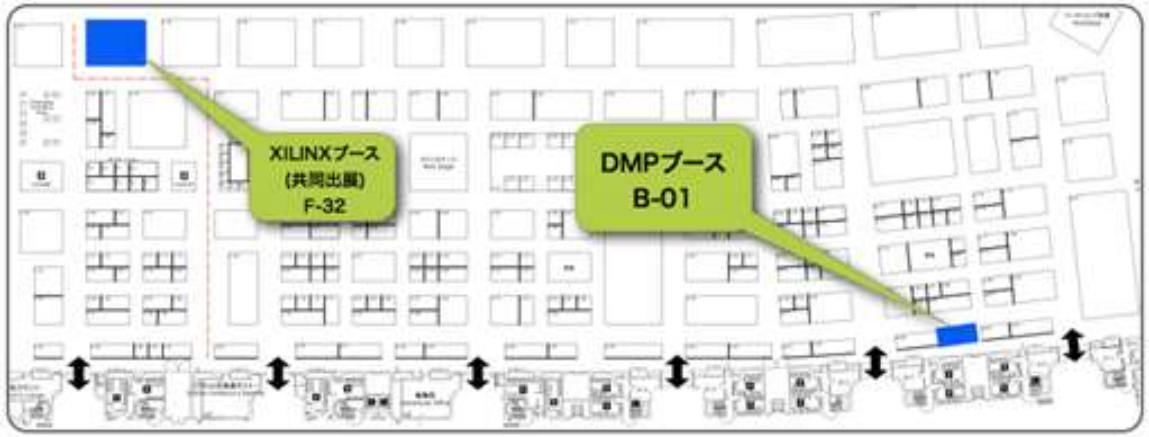
“サイレント”拡張

- OpenGL ES 2.0 はOpenGLの柔軟性の一部に対して制約をかけている
- 例: `glTexImage2D(T, L, internalformat, W, H, B, format, type, d);`
  - ESでは, *format*は *internalformat*と合致しなければならない
- これらのルールはOpenGLESのコンフォーマンステストでは十分なテストが行われているわけではない。
  - いくつかのGPU実装では許しているかもしれないけど、すべてのGPUでかどうかはわからない
- Best practices
  - ルールを知りましょう
  - OpenGL ESのワーキンググループでもヘルプ

まとめ

- ポータブルなOpenGL ES 2.0コードを書くのは大変
  - 技術革新の早い現代における対価と考えましょう
- OpenGL ES2.0とうまくつきあうために
  - APIのコア機能でソフト設計を行う
  - 仕様を理解する
  - きちんとデバッグ
  - できるだけ実装依存な機能を使わない
  - 各GPUの特徴を理解、テストする
- Good luck!

【機器展示】
会期　2011年11月16日(水)～18日(金) 10:00～17:00
(11/17は18:00まで)
会場　パシフィコ横浜 展示ホール
小間番号　DMPブース:B-01



【プライベートカンファレンス】
「DMPの先進モバイルグラフィックス技術」
主催　クロノス・グループ (Khronos Group)
日時　2011年11月18日（金）14：40～15：40
会場　パシフィコ横浜 会議センター3F [313+314]

【講演内容】
オープンな業界標準APIの仕様策定を行うクロノス・グループの最新情報をはじめ、デジタルメディアプロフェッショナルが、クロノスAPIを採用した最新製品・技術をご紹介します。