OpenGL ES御紹介と アップデート

大渕 栄作、桐井 敬祐
株式会社ディジタルメディアプロフェッショナル

Outline

- OpenGL ES 概要
- アップデート
- 活動状況
  - 議論中の拡張機能
  - OpenGL ES 2.0適合試験

Outline

- OpenGL ES 概要
- アップデート
- 活動状況
  - 議論中の拡張機能
  - OpenGL ES 2.0適合試験

OpenGL ESとは

- 最も広く使われている、組込みシステム向けの3Dグラフィックス API
  - 様々なデバイス、アプリケーションで採用
  - 現在、OpenGL ES 1.1が中心的な役割を果たしている。
- 先進的なユーザーインターフェイス
  - iPhone でも OpenGL ESを採用
- モバイルゲーム
  - 2010年には72億ドル規模の市場に(Informa Telecom & Media社)
- 3D ナビゲーション
  - Google Earth 等の3Dでの地図情報表示

ソニー **PS3**、ノキア**N93**、**iPhone** に、
OpenGL ES 1.1が使用されています

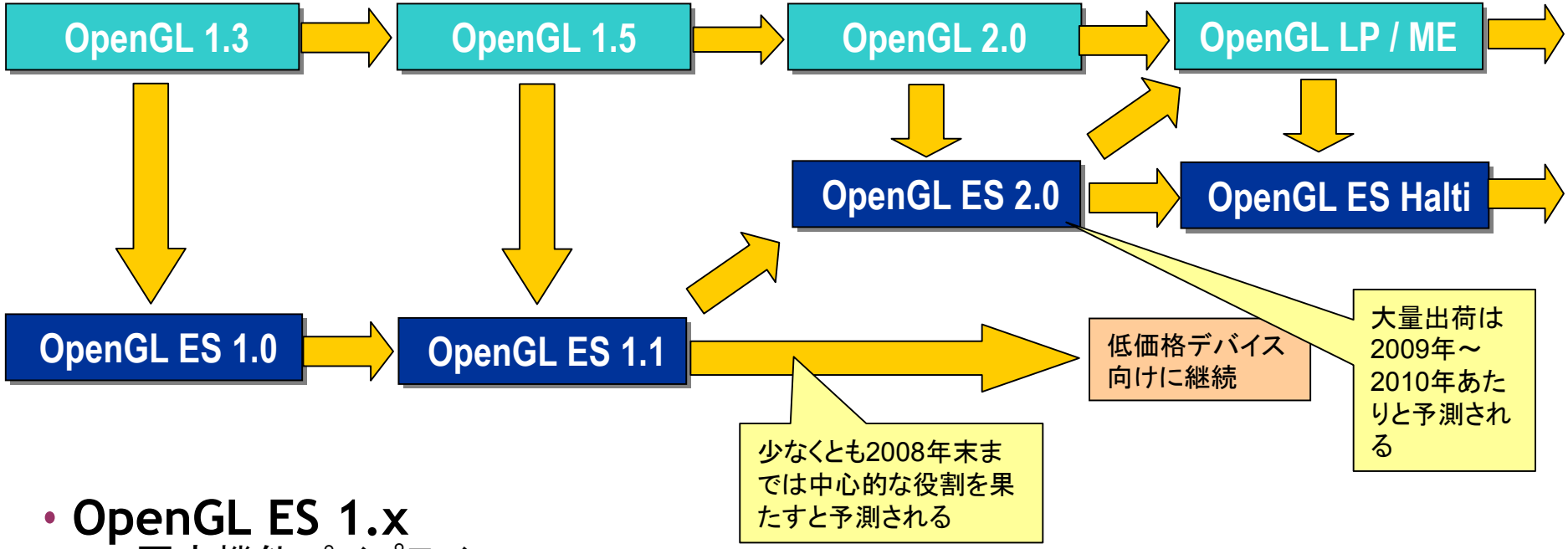
OpenGL ESとは

- **組込み機器向け3D API リード**
  - デスクトップ 向けのOpenGLがベース
  - 組込み機器向けに最適化
    - 無駄な部分や使われない機能を削除
    - 組込み機器で使われるデータ型を追加
  - *デスクトップ向けOpenGLの大部分の機能サポートしながら、モバイルでもサポート可能なAPIを定義*

- **実績**
  - 1500万台以上のOpenGL ES 1.1搭載携帯電話が出荷済み
  - OpenGL ES搭載セットトップボックス・カーナビ製品の出荷が始まる

OpenGL および OpenGL ES のバージョン

デスクトップ向け(ARB)　　組込みシステム向け(OpenGL ES WG)



- **OpenGL ES 1.x**
  - 固定機能パイプライン

- **OpenGL ES 2.x**
  - プログラマブルパイプライン
  - OpenGLES 1.xとの後方互換はない

LP: Longs Peak (OpenGL 3)
ME: Mount Evans(OpenGL ?)

Outline

- OpenGL ES 概要
- アップデート
- 活動状況
  - 議論中の拡張機能
  - OpenGL ES 2.0適合試験

News #1 - OpenGL ES 2.0 仕様のリリース

- 主な機能
  - ﾊﾞｰﾃｯｸｽｼｪｰﾀﾞ、ﾌﾗｸﾞﾒﾝﾄｼｪｰﾀﾞを追加
  - 固定機能パイプラインを削除
  - コンパクトで、効率的なAPI
  - シェーダ記述言語 (GLSL ES 1.00)
  - オンラインに加え、オフラインのGLSLコンパイルのサポート

- 先進的な機能セット
  - 8個のattribute / varying / sampler
  - キューブマップのサポート
  - フレームバッファオブジェクト(FBO)のサポート

News #2 - OpenGL ES 1.1 完全版仕様書

- 当初は、デスクトップ向けOpenGLとの機能比較で仕様書を定義
  - コンパクトなOpenGL ESの仕様書
  - ターゲットは、デスクトップOpenGLをよく知っている人
  - ただ、デスクトップOpenGLをあまり知らない人にはわかりにくい
- OpenGL ES1.1の完全版仕様書をリリース
  - Jon Leech氏によって作成
  - OpenGL 1.5 仕様書をベースに作成
  - 差分仕様書もリファレンスとして引き続きアップデート

OpenGL ESワーキング・グループでの優先順位

- 適合試験(Conformance test)
  - ES 2.0 向けの適合試験 – 2007年10月ぐらいのリリースを計画
  - ES 1.1 向けの適合試験 - 引き続きテストの改良を進めている
- ES 1.x and 2.0の拡張
  - 新しい技術の追加
  - 整理整頓
- エコシステムサポート
  - TSGによるドキュメントの整備
  - SDK
- 次世代API (‘Halti’)
  - 新しいオブジェクトモデルおよび、”Mount Evans”の特徴を導入
  - 後方互換性は無い
  - 2010年をターゲット

Outline

- OpenGL ES 概要
- アップデート
- 活動状況
  - 議論中の拡張機能
  - OpenGL ES 2.0 適合試験

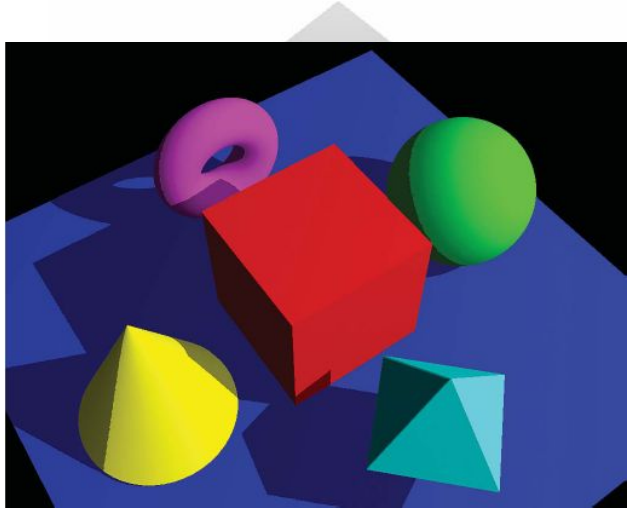
OpenGL ES 2.1への道?

- 現在の拡張
  - GL_OES_blend_equation_separate
  - GL_OES_blend_func_separate
  - GL_OES_blend_subtract
  - GL_OES_byte_coordinates
  - GL_OES_compressed_ETC1_RGB8_texture
  - GL_OES_compressed_paletted_texture
  - GL_OES_draw_texture
  - GL_OES_extended_matrix_palette
  - GL_OES_fixed_point
  - GL_OES_framebuffer_object
  - GL_OES_matrix_get
  - GL_OES_matrix_palette
  - GL_OES_point_size_array
  - GL_OES_point_sprite
  - GL_OES_query_matrix
  - GL_OES_read_format
  - GL_OES_single_precision
  - GL_OES_stencil_wrap
  - GL_OES_texture_cube_map
  - GL_OES_texture_env_crossbar
  - GL_OES_texture_mirrored_repeat
  - GL_OES_egl_image
  - GL_OES_depth24
  - GL_OES_depth32
  - GL_OES_element_index_uint
  - GL_OES_fbo_render_mipmap
  - GL_OES_fragment_precision_high
  - GL_OES_mapbuffer
  - GL_OES_rgb8_rgba8
  - GL_OES_stencil1
  - GL_OES_stencil4
  - GL_OES_stencil8
  - GL_OES_texture3D
  - GL_OES_texture_{half_}float{_linear}
  - GL_OES_texture_npot
  - GL_OES_vertex_half_float
- **議論中の拡張:**
  - **OES_depth_texture**
  - **OES_standard_derivatives**
  - **OES_vertex_data_10_10_10_2**
  - **OES_required_internal_format**

OES_depth_texture

- OpenGL 2.0の拡張機能（OpenGL ESでは仕様から削除された）を復活
- 主にシャドウマッピングで使用する
- デプス値のための単一チャンネルのテクスチャ内部フォーマット
  - 16-, 24-, or 32-bit unsigned integer
  - サンプラからは、(D,D,D,1)が返される

＜シャドウマッピング＞
•FBOにアタッチ
- 光源系を1stパスで取得
•デプステクスチャをシーンに投影
•影部分の比較判定に使用
- デプステクスチャからのサンプル値と、レンダリングしているピクセルの光源からの距離を、フラグメントシェーダで比較し、デプステクスチャのサンプル値よりも遠くにある場合は、そのピクセルの位置には影が落ちている（光源からは見えない）と判定。

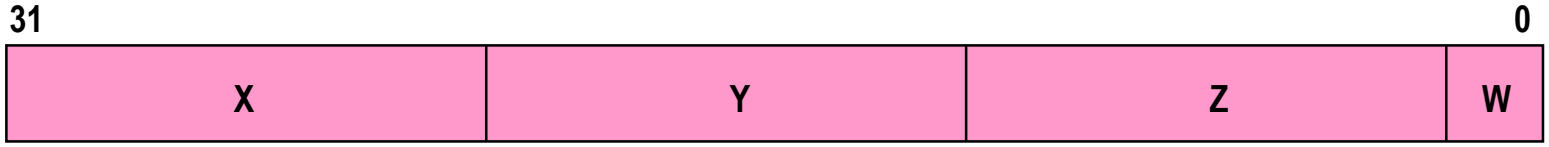


OES_standard_derivatives

- デスクトップ向けGLSL 1.10から削除したフラグメントシェーダ用組込み関数のデリバティブ関数を拡張機能として復活
  - genType **dFdx**(genType p): ウィンドウ空間におけるX方向の偏微分量
  - genType **dFdy**(genType p): ウィンドウ空間におけるY方向の偏微分量
  - genType **fwidth**(genType p): abs(dFdx(p)) + abs(dFdy(p))
- プロシージャルテクスチャでエイリアスを避けるのに使用
  - 通常のテクスチャマッピングでは、この目的でMipmapを使う
  - フラグメントシェーダではこの関数を使いエイリアスの回避処理が可能
  - ウィンドウ空間で値が大きく変わる場合には、別の処理に移行
  - 別処理では、平均色を使うなどの処理を行う
- ENABLEにしないと使えないので、注意
  - #extension GL_OES_standard_derivatives : enable
- 描画品質と性能のトレードオフのためのhint
  - glHint(GL_FRAGMENT_SHADER_DERIVATIVE_HINT_OES, GL_NICEST)
  - 演算精度に関するヒント

OES_vertex_data_10_10_10_2

- バーテックスシェーダーの入力変数（頂点属性）に新しいデータフォーマットを提供する
- 2つの新しいデータフォーマット
  - GL_UNSIGNED_INT_10_10_10_2_OES
  - GL_INT_10_10_10_2_OES
  - 3- or 4-component arrays



- 保持データ量を減らし、バンド幅を節約が目的
  - Vertex属性データは、少ないビット幅ですむ場合がある
  - ¼のデータ量削減 (32-bits vs. 128-bits(Vec4))

- プリミティブ単位の頂点属性設定関数glVertexAttrib{1234}f{v} に、このフォーマットを使うことはできない。
  - 新しいsuffixをつける必要があり。
  - 本extenstionは、Vertex arrayのみ、現実的には有用

OES_required_internal_format

現状

- テクスチャロードの方法:
  - glTexImage2D(…, internalformat, …, type, &data);
- デスクトップ向けOpenGLでは、自由度が高い
  - 基本内部フォーマット: RGB, RGBA, …
  - サイズ付フォーマット: RGBA2, RGB5, LUMINANCE12, …
  - 多くのtype: 3_3_2, 5_6_5, byte, short, float…
  - ドライバは、一番近い内部フォーマットに変換する。
- OpenGL ES は基本内部フォーマットのみ
  - ドライバ内のパスの数を節約できる
  - 各フォーマット間の変換は必要ない
  - ドライバは、<type>をサイズのヒントとして使用
  - アプリケーションは、必要なサイズでデータを渡す事が前提
    - もし内部フォーマット5_6_5が必要なら、ピクセルデータ型5_6_5を使用

OES_required_internal_format

問題点

- 1) ドライバがどの内部フォーマットを使うか保証がない
  - テクスチャ・データタイプはhintでしかない
  - 理屈としては、RGB8のデータを2bit/pixelに入れても適合テストは合格できる
  - 実際は、適合テストは4bit/pixelを要求する
- 2) アプリケーションのカラー型に関する自由度は意外に低い
  - 例: JavaのImageDataは常に32bit/pixel
  - 通常は16bit/pixelで十分かつ、性能も向上
  - なので、ある実装では常にRGB8 => RGB565に変換
- アプリケーションが望む精度になるとは限らない...

OES_required_internal_format

解決方法

- **OES_required_internal_formatを導入**
  - OES 拡張 (authored by Jeremy Sandmel, Apple)

- **何ができるか**
  - ﾃﾞｽｸﾄｯﾌﾟ向けOpenGLからサイズ付内部フォーマットの限定的選択を復活
  - アプリケーションから精度を落とすよう指定可能
  - ドライバが少なくとも要求した精度を使うよう指定可能

- **何ができないか**
  - 基本フォーマットの変換。(例: LUMINANCE → RGB)
  - 上位精度への変換。(例:565データ → RGB8テクスチャ)

- **特徴**
  - 後方互換性 – 基本内部フォーマットについては既存のルールを適用
  - 低コスト – ドライバに新に追加しなければならない変換パスを最小限に抑える
  - テクスチャが必要最小の精度でストアされる事を保証
  - 今年の後半にリリース予定

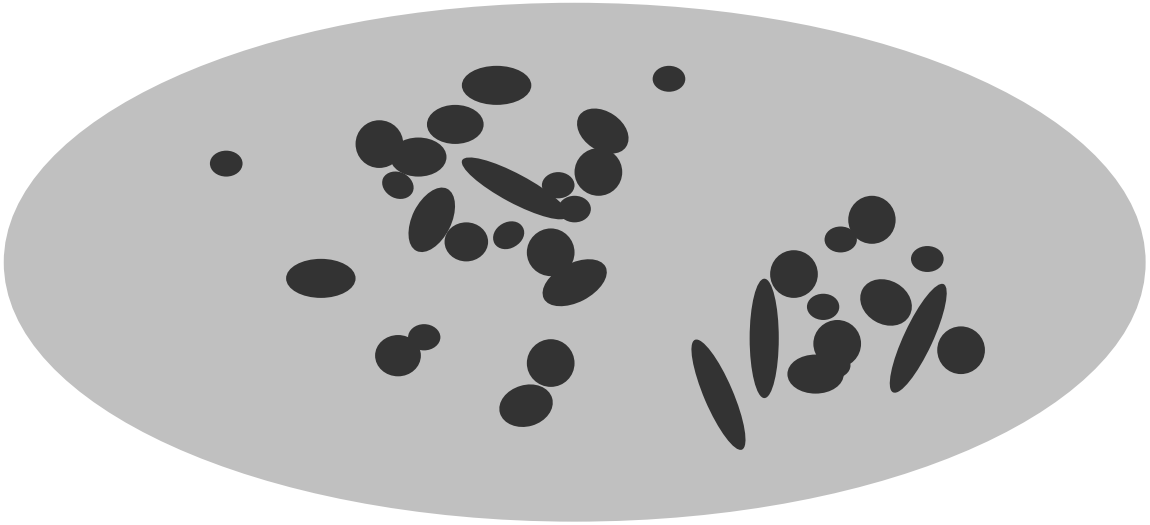
Outline

- OpenGL ES 概要
- アップデート
- 活動状況
  - 議論中の拡張機能
  - OpenGL ES 2.0 適合試験

OpenGL ES 2.0適合試験

- **OpenGL ES 2.0適合試験スイート**
  - OpenGL ES Common Profile Spec 2.0
  - The OpenGL ES Shading Language(GLSL ES) V1.00

- **適合試験スイートの方針**
  - 徹底的(完璧)にテストしない
  - 厳密な完全一致を求めない
  - テストスイート　＝　適合したプログラムの「サンプリング」



GLSL ESの制限

GLSL ES 1.00仕様書の付録

- **Appendix A: Limitations for ES 2.0**
  - 2 Length of Shader Executable
    This is defined by the conformance tests
  - 3 Usage of Temporary Variables
    The maximum number of variables is defined by the conformance tests.

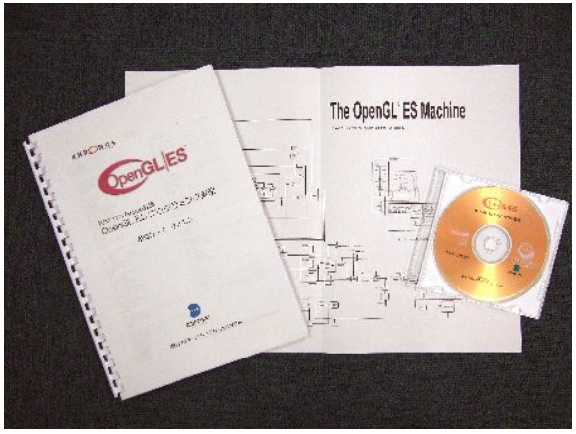
- シェーダの実行プログラムサイズ
- 一時変数の最大数

は、適合試験で定義される

クロノスグループ公認 トレーニングコース

- **OpenGL ESプログラミング 基礎コース**
  - 次回日程: 2007年11月7日(水)～8日(木)
  - OpenGL ES概要 / EGLについて / ﾌﾟﾘﾐﾃｨﾌﾞの描画 / ﾋﾞｭｰﾎﾟｰﾄの設定 / 変換
    ﾃﾞﾌﾟｽﾊﾞｯﾌｧ / シーンの描画 / ｱﾆﾒｰｼｮﾝ / 視野 / ﾗｲﾃｨﾝｸﾞ / ﾌﾞﾚﾝﾃﾞｨﾝｸﾞ
    ﾋﾟｸｾﾙｵﾍﾟﾚｰｼｮﾝ / ﾃｸｽﾁｬﾏｯﾋﾟﾝｸﾞ(Ⅰ) / ﾌｫｸﾞ / ﾌﾚｰﾑﾊﾞｯﾌｧ・ｵﾍﾟﾚｰｼｮﾝ
    ﾊﾞｯﾌｧ･ｵﾌﾞｼﾞｪｸﾄ / 拡張:ﾎﾟｲﾝﾄｽﾌﾟﾗｲﾄ / 拡張:ﾏﾄﾘｯｸｽﾊﾟﾚｯﾄ / その他のTips

- **OpenGL ESプログラミング 応用コース**
  - 次回日程: 2007年11月9日(金)
  - ﾃｸｽﾁｬﾏｯﾋﾟﾝｸﾞ(Ⅱ) / 圧縮ﾃｸｽﾁｬ / ﾄﾞﾛｰﾃｸｽﾁｬ
    動的ｷｭｰﾌﾞﾏｯﾋﾟﾝｸﾞ / 平面反射 / Dot3ﾊﾞﾝﾌﾟﾏｯﾋﾟﾝｸﾞとｽﾍﾟｷｭﾗｰﾏｯﾋﾟﾝｸﾞ
    ｽﾃﾝｼﾙｼｬﾄﾞｳ / ﾊﾟﾌｫｰﾏﾝｽ評価のﾋﾝﾄ / OpenGL ES 2.0の概要



- **GLSLシェーダプログラミング 基礎コース**
  - 次回日程: 2007年12月
  - 第Ⅰ部 OpenGL ES 2.0およびシェーダの概要
    OpenGL ES 2.0の概要 / シェーダの概要
    第Ⅱ部 GLSL 1.10
    シェーダ用APIの基礎 / ミックスモード / 文法 / 変数 / 関数
    ﾊﾞｰﾃｯｸｽｼｪｰﾀﾞ / ﾌﾗｸﾞﾒﾝﾄｼｪｰﾀﾞ / ﾃﾞﾊﾞｯｸﾞおよび開発のためのTIPS
    第Ⅲ部 GLSL ES 1.00
    OpenGL ES Shading Language

詳細、お申込み方法については、お手元のパンフレット、および下記URLをご参照ください。
http://www.dmprof.com/

OpenVG紹介とアップデート

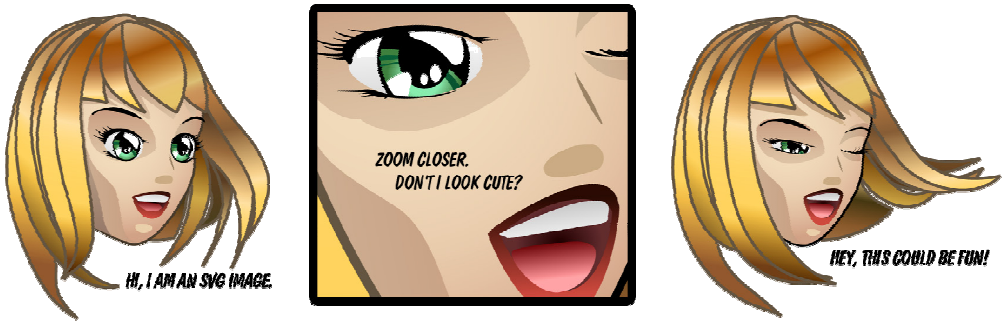
大渕 栄作、桐井 敬祐
株式会社ディジタルメディアプロフェッショナル

ベクターグラフィックス

- 点、線、曲線、多角形などの2Dプリミティブとして描画

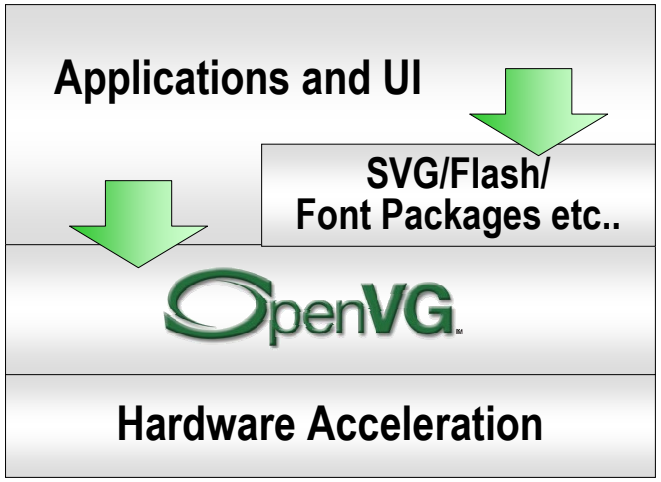
  （ラスタグラフィックスでは、ピクセル集合としてイメージを描画）

- スケーラブルな描画表現を実現
  - 画質を落とさずにリサイズ・引き延ばしが可能
    → 解像度の違う環境でも画質の維持が可能
  - 描画データサイズの削減、メモリフットプリント節約
- コンテンツ制作環境が充実
  - Flash , illustrator …

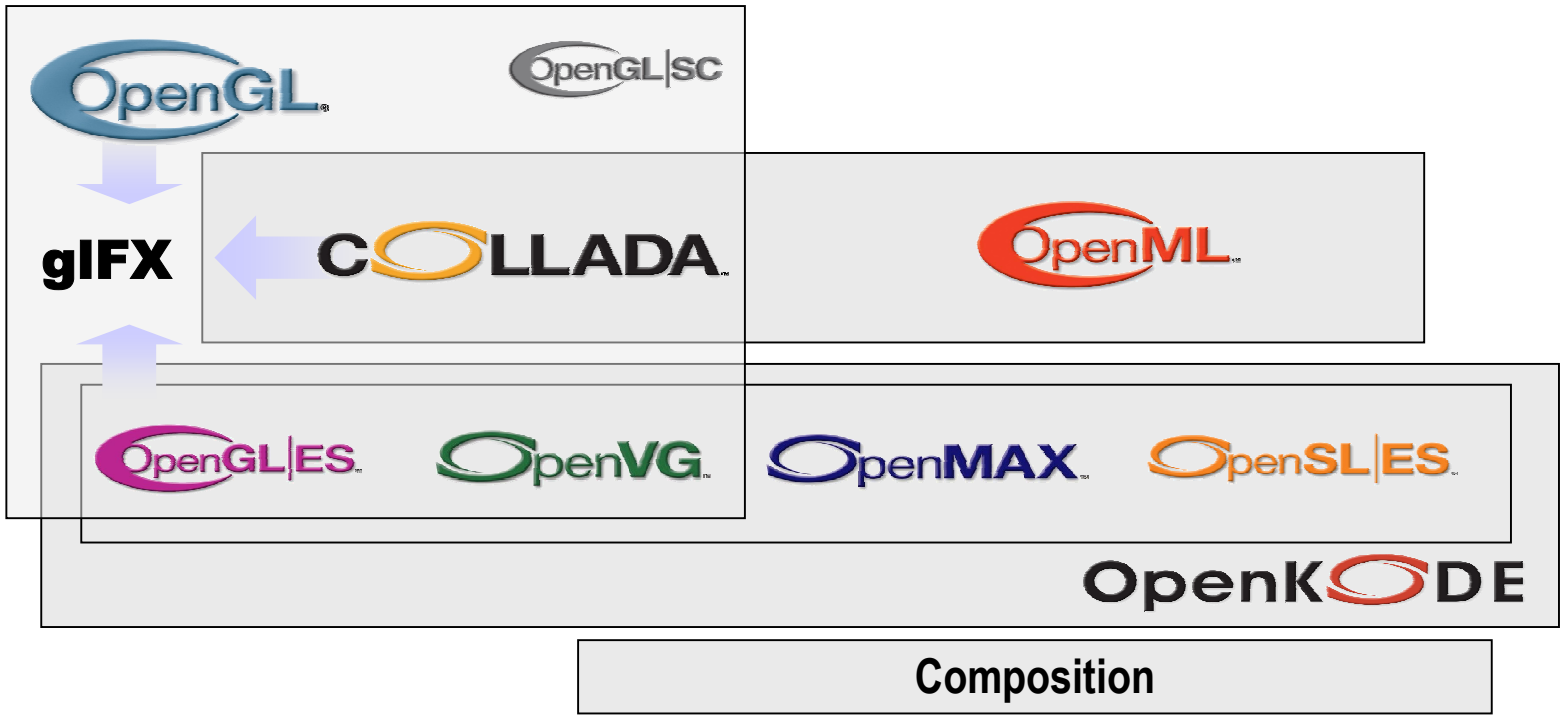


OpenVG

- ロイヤリティフリー・オープンスタンダードAPI
- 低レベル2Dベクターグラフィックスレンダリング API
- SVG, Flash, Vector fontsなどの機能を生かす機能セット
- コンテンツのポータビリティ
- ハードウェアアクセラレーション



Khronos Standards Ecosystem



OpenVGターゲット

- **SVG**
  - SVG Tiny 1.2
- **Adobe PostScript**
- **Adobe PDF**
- **Adobe Flash**
- **Java (JSR 287)**

ターゲットアプリケーション

- ユーザインターフェイス
- SVGビューア
- 地図アプリケーション
- E-bookリーダ
- ゲーム

OpenVGハードウェアアクセラレーション

- 高速なレンダリングが可能
- 消費電力の大幅な低減
  - 同じ描画処理をした場合、ARM11のような組込みCPUと比べ大幅な消費電力低減が可能。
- 高解像度でリッチなVGコンテンツが可能
  - 高解像度ディスプレイでは、ソフトウェアラスタライザでは十分なレンダリング性能が得られない

OpenVGの実装アプローチ

- 専用VG H/W
  - すべてのOpenVG パイプラインをハードウェア化
- 3Dグラフィックス H/Wベース
  - 3Dグラフィックス H/Wにデータを送る前に、前処理が必要
  - いくつかの機能(ブレンディングモード等)については、3Dグラフィックス H/Wでサポートされていないため、一部ソフトウェアで処理が行われる場合も
- OpenGL / OpenGL ES API ベースの実装
  - OpenVG ラッパー
- VGソフトウェアラスタライザ
  - 完全ソフトウェア実装

専用VG H/W

- **AMD社 VG Core**
  - 最初にして唯一のVG専用 H/W
  - OpenVG 1.0のパイプラインをハードウェア実装
  - OpenVG 1.0.1適合試験に対応

3Dグラフィックス H/W ベース

- **Imagination Technologies (IMG)社**
  - PowerVR MBX – すでに出荷済み
  - PowerVR MBX/SGX family でのOpenVGサポートをコミット
- **DMP社**
  - PICA VG – 現在開発中
  - ネイティブハードウェア実装についても計画中

OpenGL/OpenGL ES API ベース

- **Mazatech社**
  - AmanithVG – OpenGL 1.1/2.0上で動作
- **Huone社**
  - AlexVG FORGE – OpenGL ES 上で動作

VGソフトウエアラスタライザ

- **Nvidia社**
  - 公式 OpenVG サンプル実装
    - Win32 プラットフォーム
    - MITライセンス下で公開済み
  - 商用OpenVG実装
    - 主要なハンドセットベンダーがライセンス
    - ARM上で動作
    - 性能最適化 (上記サンプル実装に比べ50 – 200x高速)
    - ﾊﾝﾄﾞﾍﾙﾄﾞ向けGPU・ｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｾｯｻ向けに移植・最適化が行われている。
- **Huone社**
  - AlexVG engine – OpenVG商用ソフトウェア実装
  - AlexVG player – OpenVG上で動くSVGT 1.1 および SVGT1.2 player

これまでのマイルストーン

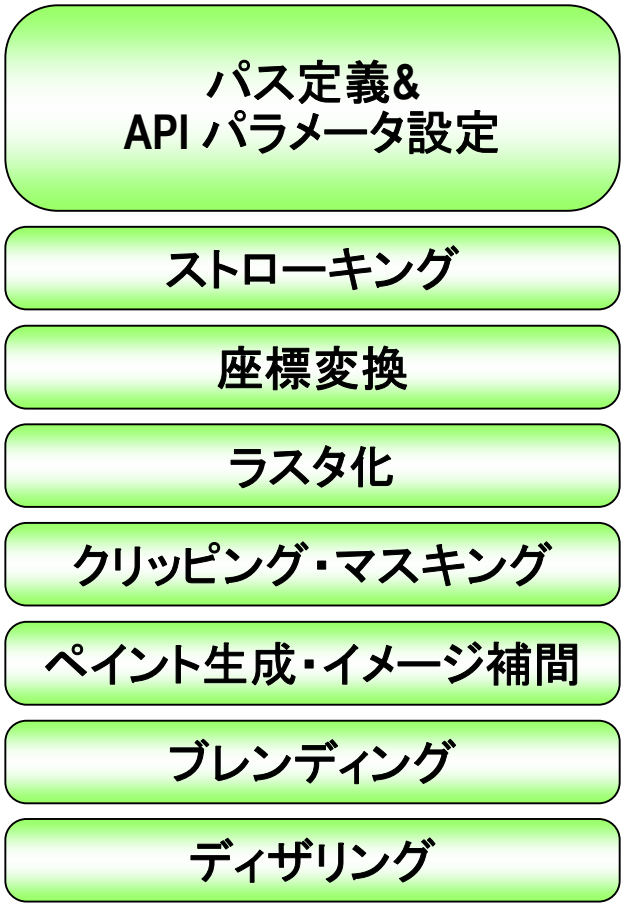
- **2005年8月**
  - OpenVG 1.0仕様リリース
  - サンプル実装1.0リリース（バイナリの公開）
- **2007年3月**
  - OpenVG 1.0.1 - 仕様改訂
  - 適合試験リリース
  - サンプル実装 1.0.1リリース（MITライセンス下でソースコード公開）

今後のロードマップ

- **OpenVG 1.1 (Q4 2007)**
  - Adobe Flash 7 / Flash Lite 2 サポート
  - Glyph API
  - 現在、仕様書に関して最終的な議論が行われている
- **OpenVG 1.2 (2008)**
  - Flash 8 サポート
  - 描画精度について厳密な要求仕様の追加
  - より高品質なアンチエイリアス処理
  - Perspective transformのサポート
- **OpenVG 2.0 (2009/2010)**
  - ブログラマブルハードウェアの導入

OpenVGパイプライン概要

- パスおよびイメージ処理のためのH/Wパイプラインを定義
  - パス定義およびAPIパラメータの設定
  - ストローキング
    - 線幅、線接合部、線終端、ダッシュパターン, etc.
  - 座標変換
    - 2x3 and 3x3 行列変換
  - ラスタ化
  - クリッピング・マスキング
    - 矩形シザリング・アルファマスク
  - ペイント生成・イメージ補間
    - 単色・グラデーション・パターンペイント
  - ブレンディング・アンチエイリアス
    - いくつかのブレンディングモードをサポート
  - 減色ディザリング
  - イメージフィルタリング



OpenVG 1.1:

- Adobe Flash 7 / Flash Lite 2 サポートに必要な機能追加
  (ブレンディングモード等)
- テキスト描画のH/Wアクセラレーションのための Glyph APIを導入

Glyph APIの必要性

- 多くのアプリケーション でスケーラブルなテキストが使われている。
- Glyph(文字の形)は、複雑な形状をとても小さなサイズにスケールダウンされる。
- Glyphが小さなサイズになった場合、文字の読みやすさ・クオリティに重大な影響

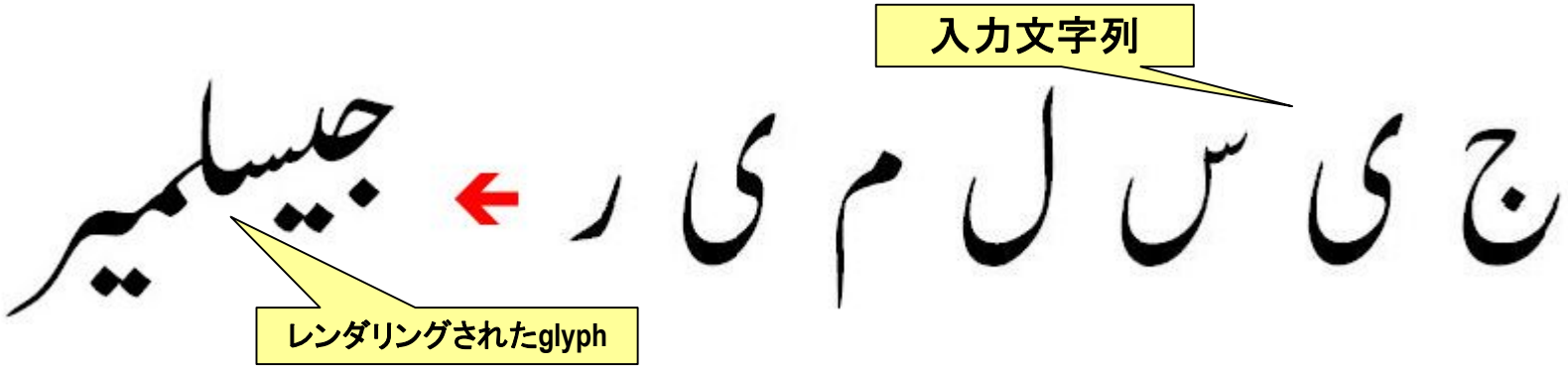
Glyphs

Glyphs

→ テキスト描画に最適化されたH/W実装の要求に対応

OpenVG 1.1でのGlyph API

- VGFontオブジェクトの生成・管理
  - このオブジェクトは、異なるアプリケーション間でシェア可能
- ベクタアウトラインおよびビットマップイメージによるテキストレンダリング
- 複雑なフォントレイアウト
  - 複雑な言語スクリプトにおける、カーニング、グリフ変形、位置の調整



OpenVG 1.1でのGlyph API

- フォント技術に依存しない
  - すでに多く用いられているフォントからVGFontオブジェクトを生成可能
  - ベクターアウトライン、ビットマップイメージのどちらの形式からもVGFontオブジェクトを生成可能。
- オリジナルhintingおよび、自動hintingのどちらも可能
  - APIは既存のフォント技術に関するIPを侵害しない。