最新モバイル3Dグラフィックプログラミング入門

株式会社ヒュージスケールリアリティ　香田夏雄

自己紹介

- 株式会社ヒュージスケールリアリティ
  - 次世代3DCG技術を研究開発する技術者集団
- 主な製品プロジェクト

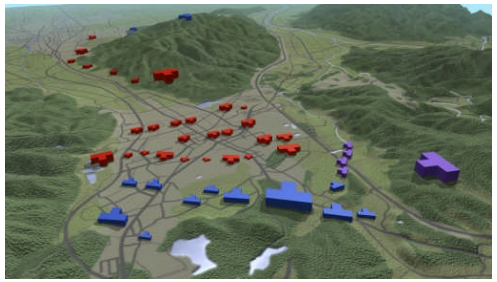
AREM Advanced Rendering Engine for Mobile　モバイル向け3次元描画ミドルウェア

Dioramax3D　高精彩3D自然景観モデルデータ生成技術

AREMによる街並みの描画例

使用データ:「GCM」株式会社ゼンリン

Dioramaxによる関ヶ原合戦布陣図



本日のAGENDA

- モバイル3Dグラフィックス最新動向
- モバイル3Dグラフィックスプログラミング入門
  - OpenGL-ES入門
  - 高速描画テクニック
  - 高画質化テクニック
  - アニメーションテクニック
  - その他の応用テクニック
    - FBO（フレームバッファオブジェクト）
    - ポイントスプライト
    - 2D描画

AREM
Advanced Rendering Engine for Mobile

モバイル3Dグラフィックス
最新動向

モバイル3Dグラフィックス最新動向

★DEMO-0

- モバイル向けGPU
  - 現状では「PowerVR MBX」がデファクト！
    - タイリングベースの描画
    - そこそこ高速
      - 2万ポリゴンで15FPS(Nokia N95の場合：弊社調べ)
    - 省消費電力
      - ※ 2万ポリゴンx15FSPで連続3.5時間駆動(Nokia N95の場合：弊社調べ)

Dell X51v

Xscale+PowerVR

Nokia N95

Omap+PowerVR

iPhone3G/iPod touch

Samsung+PowerVR

モバイル3Dグラフィックス最新動向

- 出そろったモバイル向け次世代主力GPU
  - PowerVR SGX
    - PowerVR-MBXの後継
    - CPU統合用GPU（CPUは選択可能＝IPコア）
    - Omap3ベースの携帯電話やiPhone 3GSに搭載済
  - NVIDIA Tegra
    - CPU統合用GPU（CPUは選択可能）
    - 性能はGeForce6レベル
    - Microsoft Zune HDに搭載予定
  - Qualcomm Snapdragon
    - CPU統合用GPU（ATI由来）
    - 東芝製スマートフォン：T-01A に搭載済

AREM
Advanced Rendering Engine for Mobile

モバイル3Dグラフィックス最新動向

- 主なGPUの機能比較

(M = million)

	パフォーマンス	機能	プログラマブル シェーダ	OpenGL-ES サポート
PowerVR MBX	頂点:3.4M - 7M ピクセル:270M - 600M	-PVRTC	NO	OpenGL-ES1.0 OpenGL-ES1.1
PowerVR SGX (Series5)	頂点:7M – 40M ピクセル:250mM– 4000M	-PVRTC	Yes	OpenGL-ES2.0 (OpenGL-ES1.1)
Tegra	GeForce6相当	- S3TC(DXTn)	Yes	OpenGL-ES1.1 OpenGL-ES2.0
Snapdragon QSS8X50	頂点:～22M ピクセル:130M	ATITC	Yes	OpenGL-ES2.0 (OpenGL-ES1.1)

※2009年8月　弊社調べ

AREM
Advanced Rendering Engine for Mobile

モバイル3Dグラフィックス最新動向

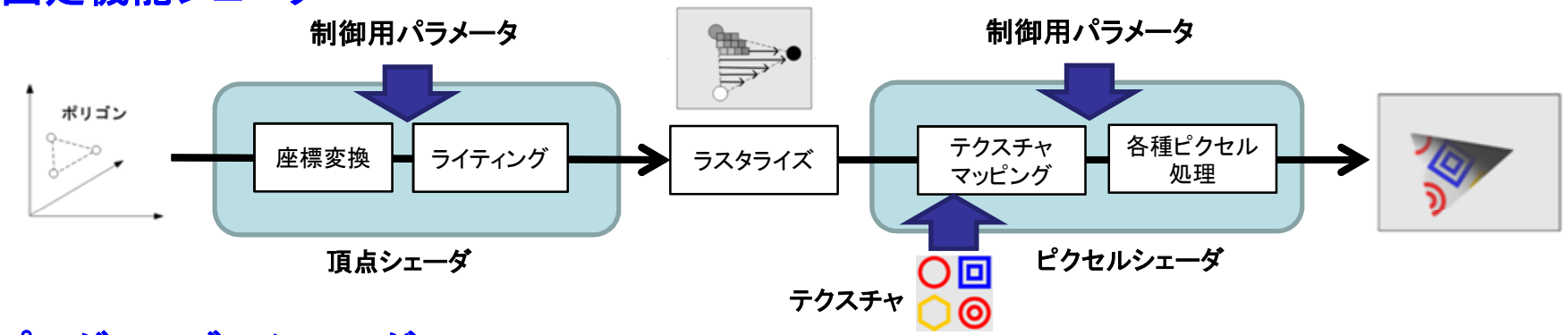
- 今後のロードマップ
  - PowerVR
    - SGX Series5 →SGX Series5XT
      - 複数コアによる並列化
      - GPGPUへも展開
  - Nvidia Tegra
    - 2010年には、同じ消費電力で現行の4倍の性能へ
  - Snapdragon
    - 高クロック版あり（UMPC向けか？）
    - 製造プロセス移行によりパフォーマンス向上
      - 2009年後半にもサンプル出荷？

AREM
Advanced Rendering Engine for Mobile

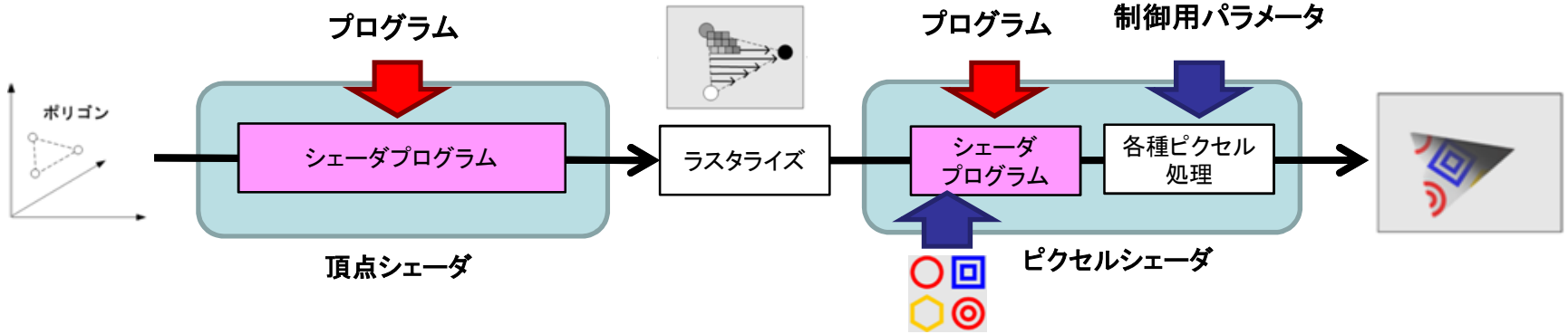
モバイル3Dグラフィックス最新動向

- キーワードは「プログラマブルシェーダ」
  - 頂点シェーダ、ピクセルシェーダがプログラム投入可能に

固定機能シェーダ



プログラマブルシェーダ



頂点単位＆ピクセル単位で思い通りの画像処理が可能に！

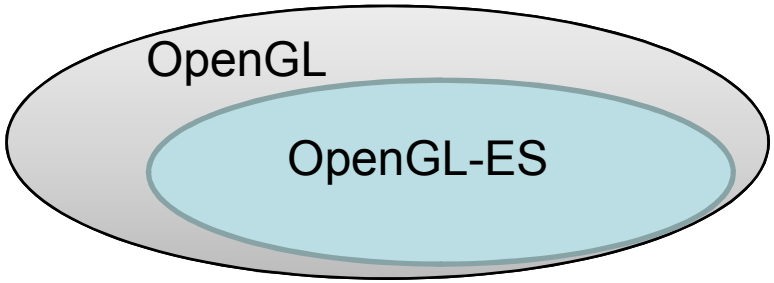
モバイル3Dグラフィックス プログラミング入門

OpenGL-ESの基礎

AREM
Advanced Rendering Engine for Mobile

モバイル3Dグラフィックスプログラミング入門

- GPU搭載モバイル機器のほぼすべてが、3D描画ライブラリとして「OpenGL-ES」を採用
- OpenGL-ESは、3D描画用の低レベルライブラリ「OpenGL」のサブセット
  - 「ES」＝Embedded（組み込み）



AREM
Advanced Rendering Engine for Mobile

モバイル3Dグラフィックスプログラミング入門

- OpenGL-ESのバージョン
  - OpenGL-ES 1.1　　←OpenGL 1.5のサブセット
    - 固定機能シェーダ対応
  - OpenGL-ES 2.0　　←OpenGL 2.0のサブセット
    - プログラマブルシェーダ対応
    - 固定機能シェーダ**非対応！**

AREM
Advanced Rendering Engine for Mobile

- モバイルデバイスのOpenGL ESの対応状況

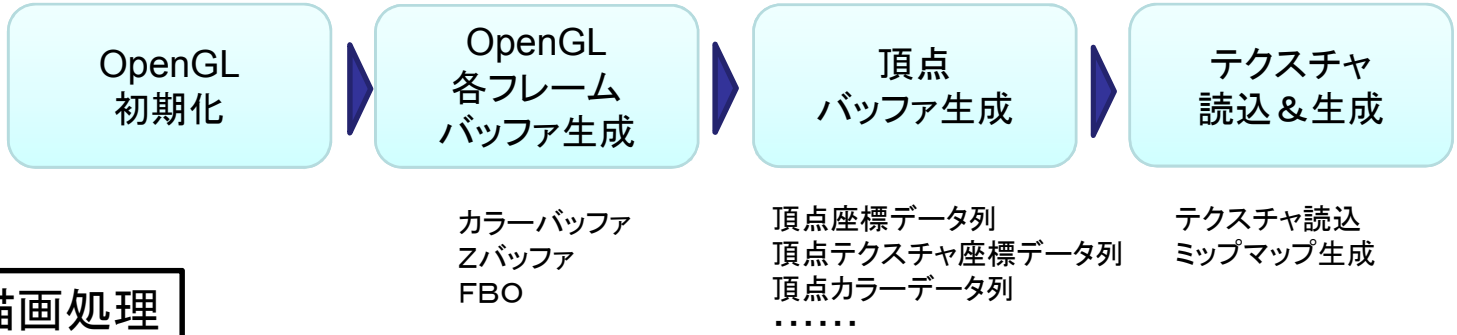
デバイス	OpenGL 1.1	OpenGL 2.0
iPhone 3G	○	×
iPod touch 2G	○	×
iPhone 3GS	○	○
HT-03A	○	×

→当面のターゲットはOpenGL-ES **1.1**

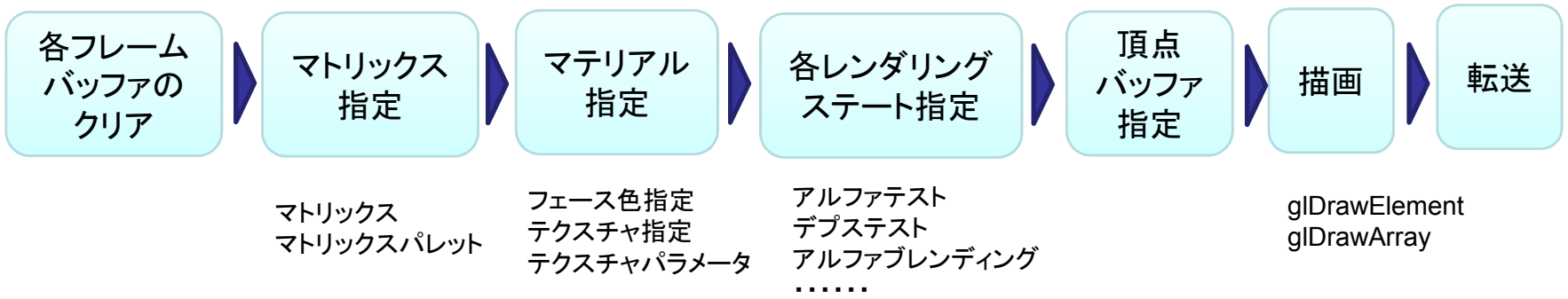
AREM
Advanced Rendering Engine for Mobile

- OpenGL-ES 1.Xの描画処理の基本

初期化＆データ読込



描画処理

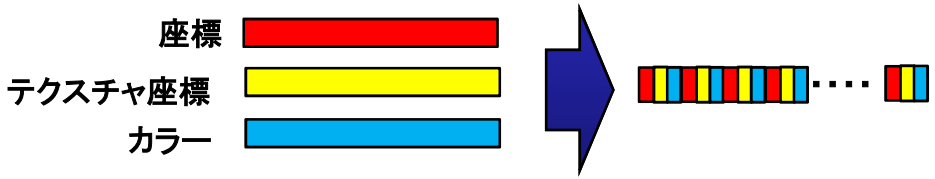


- OpenGL-ESの初期化
  - 一般的な環境の場合
    - EGL関数を使用（標準的なデバイス）
    - EGLは、クラスではないが、きめ細かな制御が可能
  - iPhoneの場合
    - ObjectiveCで記述されたEAGLクラスを利用して初期化
    - EAGL機能は、最低限に絞られている（MSAAが使えない）
    - ただし、OpenGL-ES2.0指向のFBOが実装されている
  - Androidの場合
    - そもそもJava（JNIだと配布時にセキュリティの問題あり？）
    - GLSurfaceViewクラス（Android1.5SDK以降）を利用

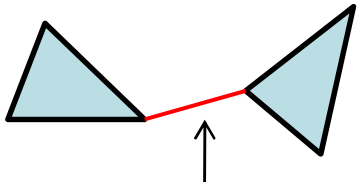
高速描画テクニック

モバイル3Dグラフィックスプログラミング入門

- データの工夫
  - データの並びに注意
  - サイズをコンパクトに
    - 例）頂点カラーデータ：float型→char型にする



- 無駄な呼び出しをしない
  - TriangleStripにすると細切れになる
    - →Trianglesで1つのバッファにする
  - 縮退三角形を使ってつなぐ



- バッファーオブジェクトの使用

- ライティングの工夫
  - OpenGL標準のライティング機能は使用しない
    - 処理が遅い
    - 狙った陰影を出すのが難しい
  - 頂点カラーを使用する
    - 頂点ごとに、あらかじめ陰影を計算しておく
    - ただし、頂点カラーの処理分だけ処理落ちする

頂点カラー使用例
約10%程度の処理落ち

★DEMO-1

- さらなるデータ効率化！
  - アーリーZカリング
    - 視点から見えるものだけにデータを限定する技術
    - PowerVRは、その機能を持っているが…

AREMの場合

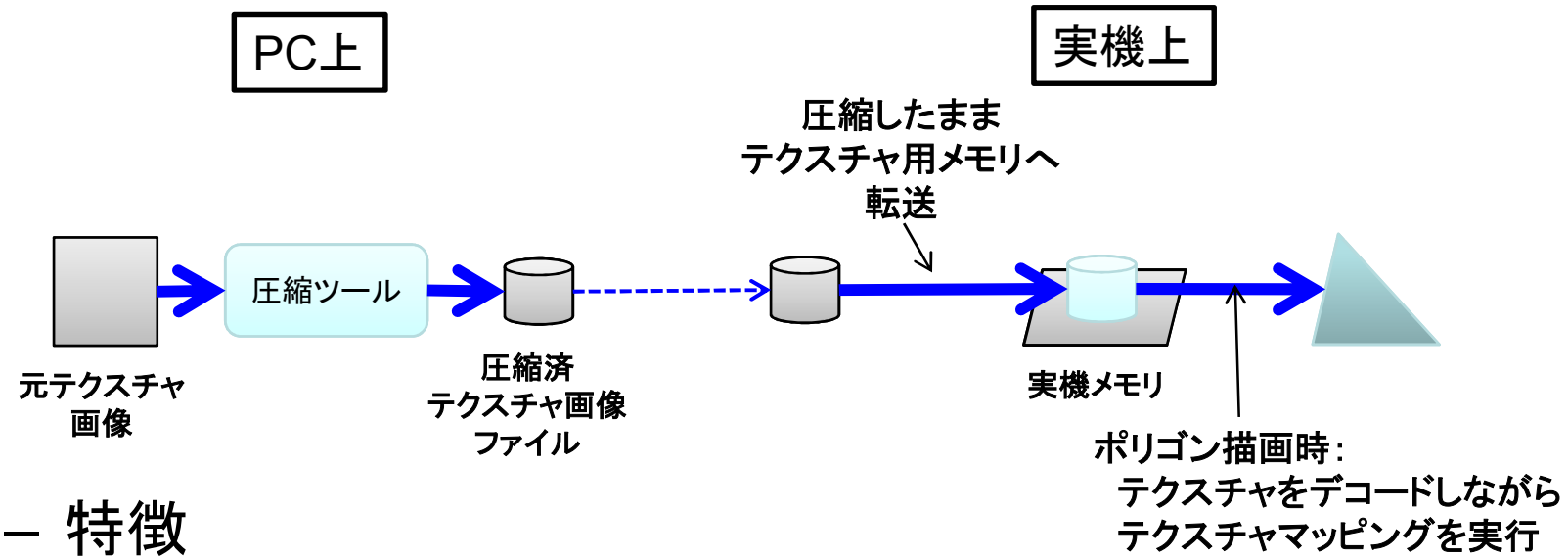
ユーザー視点

充実した街並み

左の俯瞰図

街並みがスカスカ
→目で見える面だけを事前に抽出

- テクスチャ圧縮機能の利用
  - PVRTC, S3TC(DXTn), ATITC



  - 特徴
    - ローディングの高速化（デコード必要なし）
    - 実機上の省メモリ化
    - 描画時のロスはほとんどない
    - ただし、ファイルの圧縮効率は悪い

- テクスチャ圧縮機能の利用：サンプルコード

iPhoneSDK3.0

```
glGenTextures(1, &textureHandle);
glBindTexture(GL_TEXTURE_2D, textureHandle);

….圧縮済テクスチャデータの先頭アドレス→dataBuff

glCompressedTexImage2D(GL_TEXTURE_2D, 0,
                       GL_COMPRESSED_RGBA_PVRTC_4BPPV1_IMG,
                       width, height, 0, dataSize, dataBuff);
```

※iPhoneの場合
-正方形の2のべき乗のサイズであることが必須

AREM
Advanced Rendering Engine for Mobile

- [コラム] iPod/iPhoneの3Dパフォーマンス

  iPhone3G < iPod touch < iPhone3GS

iPhone3G

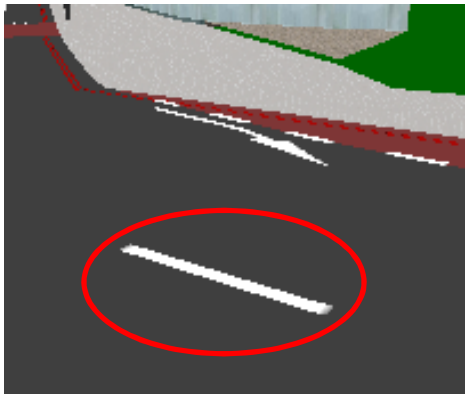
iPod touch

iPhone3GS

高画質化テクニック

- 主な画質低下の原因
  - ポリゴン面のモアレ
  - ポリゴンエッジのジャギー
  - Zファイティング

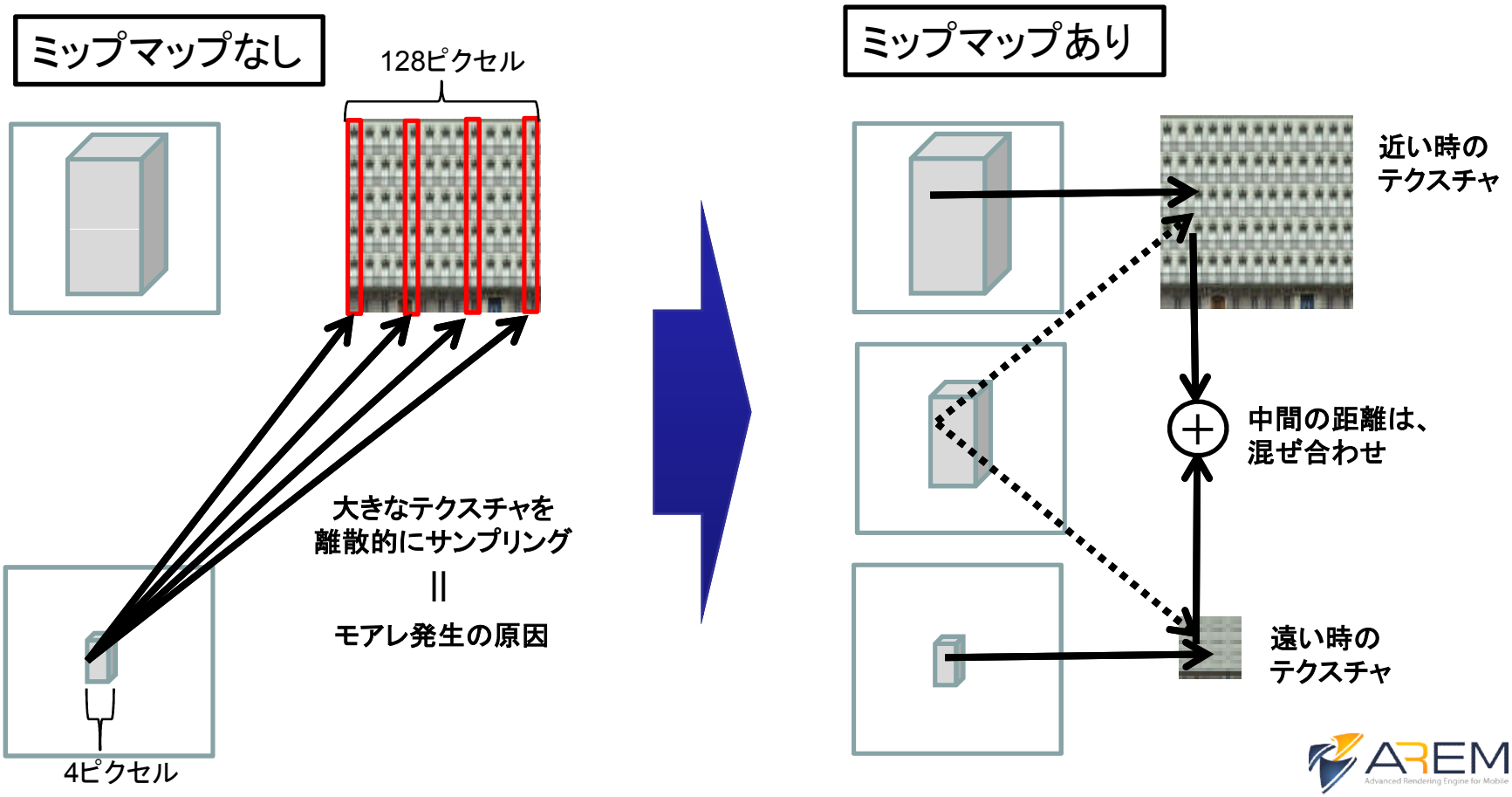
モアレ（ちらつき感）



ジャギー

Zファイティング

- ミップマップでモアレを消す
  - 視点からの距離に応じてテクスチャを切り替える



- ミップマップ でモアレを消す：サンプルコード

テクスチャ読み込み時

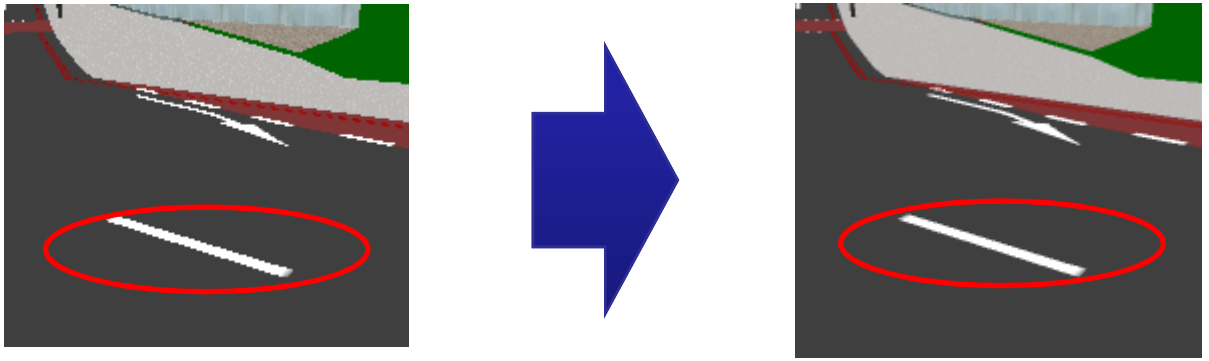
```
glGenTextures(1, &textureHandle);
glBindTexture(GL_TEXTURE_2D, textureHandle);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height,
                    0, GL_RGBA, GL_UNSIGNED_BYTE, &tmp_tex);
glGenerateMipmapOES(GL_TEXTURE_2D);
```

iPhoneSDK3.0

描画処理時

```
glBindTexture(GL_TEXTURE_2D, textureHandle);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR_MIPMAP_LINEAR);
```

- アンチエイリアスでジャギーを消す
  - PowerVRに搭載されているMSAA機能を使う
    - MSAA（Multi Sampling Anti Alias）＝FSAAの1/4程度のフレームメモリ量で同等の効果の得られるアンチエイリアス手法。必要メモリ量が少ないので、モバイル機器と相性が良い

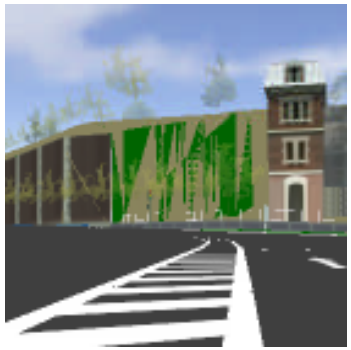


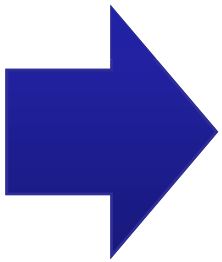
- アンチエイリアスでジャギーを消す：サンプル

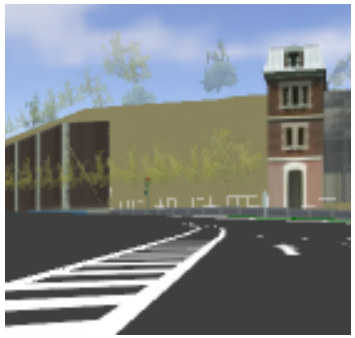
```
EGLint configAttribs[] =
{
                EGL_SURFACE_TYPE,   EGL_WINDOW_BIT,
                EGL_RED_SIZE,                   5,
                EGL_GREEN_SIZE,                 6,
                EGL_BLUE_SIZE,                  5,
                EGL_ALPHA_SIZE,                 0,
                EGL_DEPTH_SIZE,                 24,
                EGL_SAMPLE_BUFFERS,             1,
                EGL_SAMPLES,                    4,
                EGL_NONE
};

eglChooseConfig(display, configAttribs, &m_config, 1, &config);
```

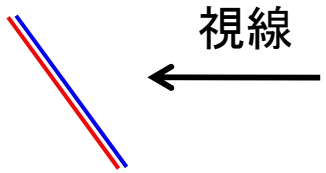
※一般的なOpenGL-ES環境で動作。iPhoneではなぜかMSAA機能が無効にされている

- Zバイアスで Zファイティングを消す
  - 視線方向と逆(Z方向)にポリゴンを浮かせる機能
  - 「ポリゴンオフセット」ともいう

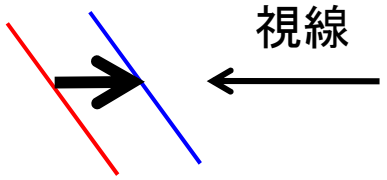








2つのポリゴンが近すぎるため
前後関係を正しく判定できない



Zバイアス機能で手前のポリゴンを
浮かせれば、正しく判定可能に

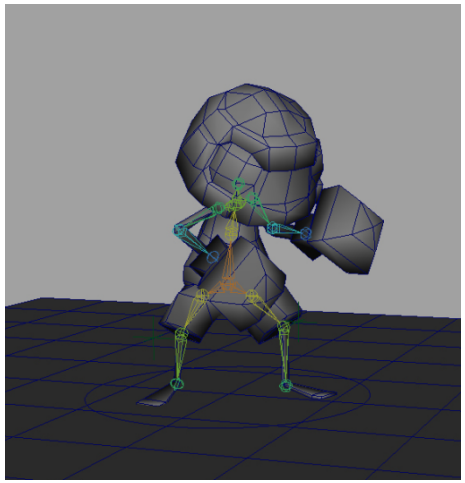
- Zバイアスで Zファイティングを消す : サンプル

```
シーンの描画……

glEnable(GL_POLYGON_OFFSET_FILL);
glPolygonOffset( 1,  15);

Zバイアス対象オブジェクトの描画
glDisable(GL_POLYGON_OFFSET_FILL);

シーンの描画
```

対象オブジェクトのときだけ
Zバイアスを有効にする

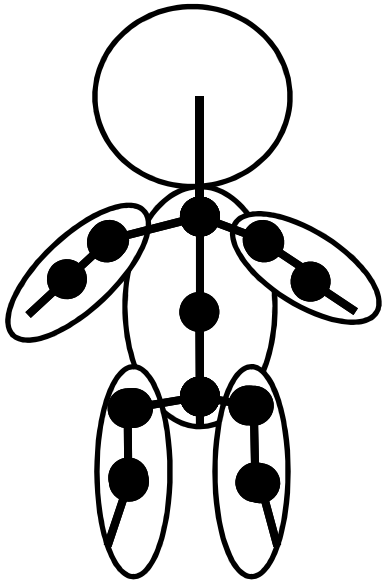
★DEMO-2

アニメーションテクニック



- ボーンアニメーション&スキニング

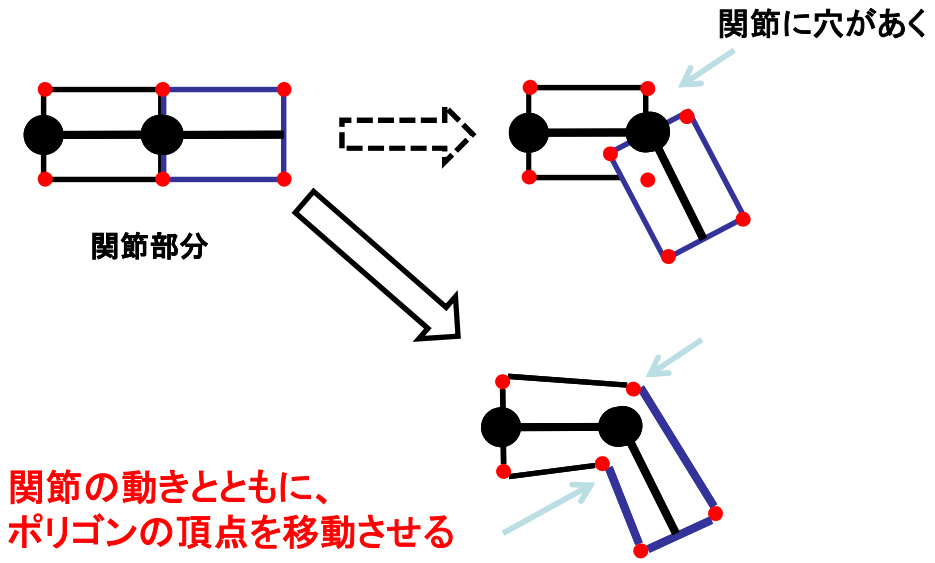
ボーンアニメーション



● 関節

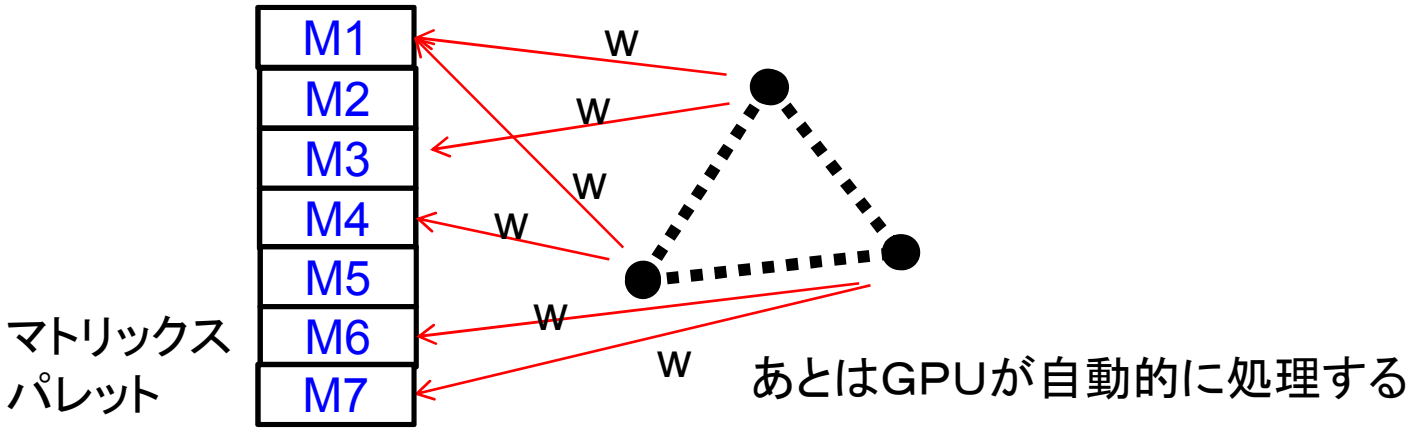
— 関節のマトリックス

スキニング



1つの頂点が複数のマトリックスの影響を受ける

- GPUのマトリックスパレット機能で高速化可能
  - 自前で処理した場合の問題点
    - 頂点ごとに複数のマトリックスの設定が必要
  - マトリックスパレットによる解決
    - あらかじめ必要なマトリックスをすべてGPUへセット
    - 各頂点には、使用するマトリックスIDと重み（影響度）を設定



- マトリックスパレット：サンプルコード

マトリックスパレット設定

```
glMatrixMode(GL_MODELVIEW);

… ここにモデルビューマトリックスの設定を行う。

glEnable(GL_MATRIX_PALETTE_OES);
glMatrixMode(GL_MATRIX_PALETTE_OES);
glCurrentPaletteMatrixOES(0);
glLoadPaletteFromModelViewMatrixOES();
```

iPhoneSDK3.0

マトリックスパレットの指定

モデルビューマトリックスからコピー

ポリゴン描画時

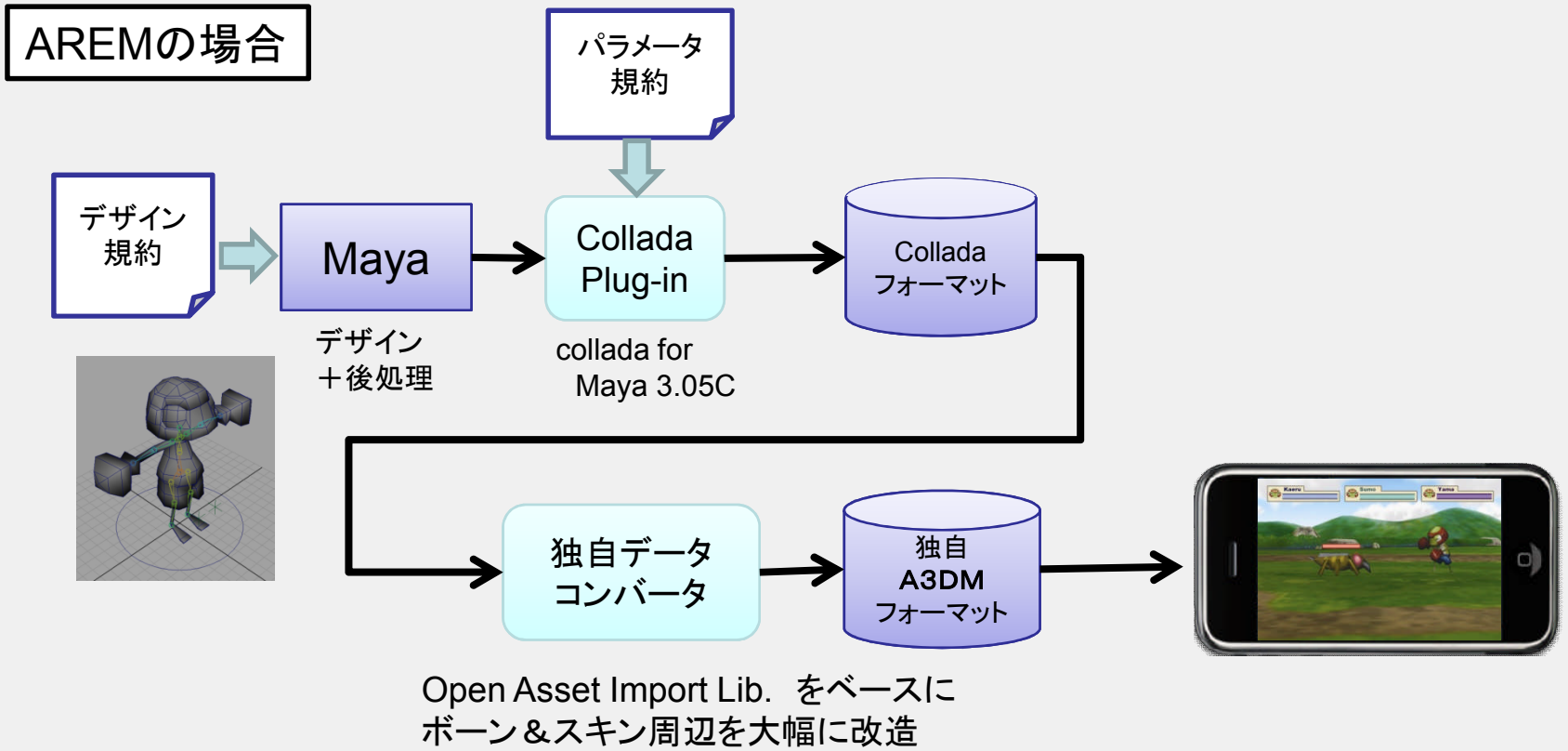
```
 const GLuint    boneID[] = {
    0, 1,    0, 2,   1, 2,   2, 4
};
const Glfloat  boneWeight[] = {
    0.5, 0.5,    0.1, 0.9,   0.5, 0.5,   0.6, 0.4
};

glEnableClientState(GL_MATRIX_INDEX_ARRAY_OES);
glMatrixIndexPointerOES(2, GL_UNSIGNED_BYTE,0, boneID);

glEnableClientState(GL_WEIGHT_ARRAY_OES);
glWeightPointerOES(2, GL_FLOAT,0, boneWeight);
```

- iPhoneにおけるマトリックスパレット
  - iPod touch / iPhone3G
    - 9パレット
    - 3ウエイト(同時に影響できるマトリックスの数)
  - iPhone3GS
    - 11パレット
    - 4ウエイト
- 一般的なキャラを表現するには、最低でも15個くらいボーンが必要
    - 全く足りない！
    - →マトリックスパレットのスケジューリングが必要！

AREM
Advanced Rendering Engine for Mobile

[コラム] コンテンツパイプライン

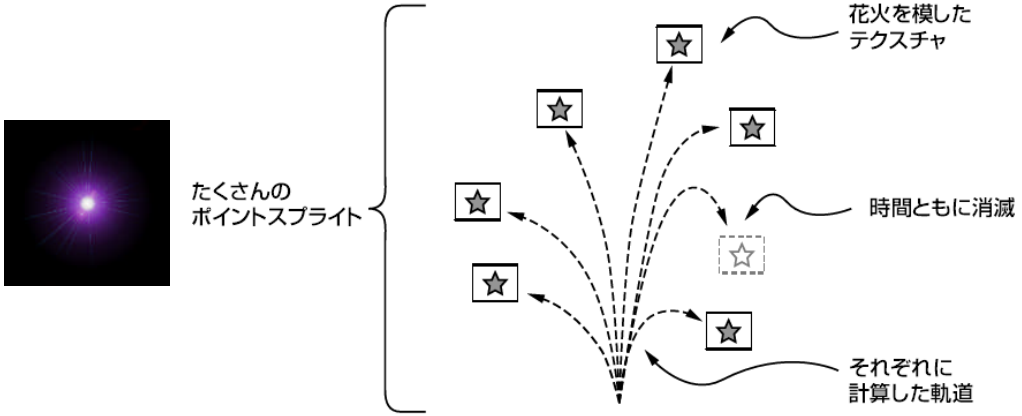
- モデリングツールから実機までの遠い道のり

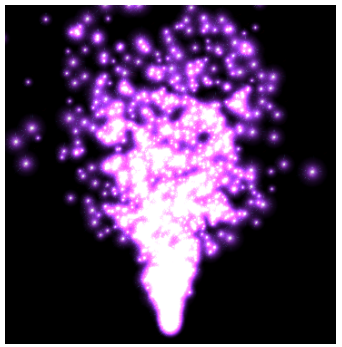


その他の応用テクニック

★DEMO-3

- ポイントスプライト
  - 三次元空間上の点にテクスチャを配置する機能
  - 各テクスチャは点（ポイント）として管理できる
  - テクスチャ画像や点の挙動を変えるだけで、様々なパーティクルエフェクトを実現できる



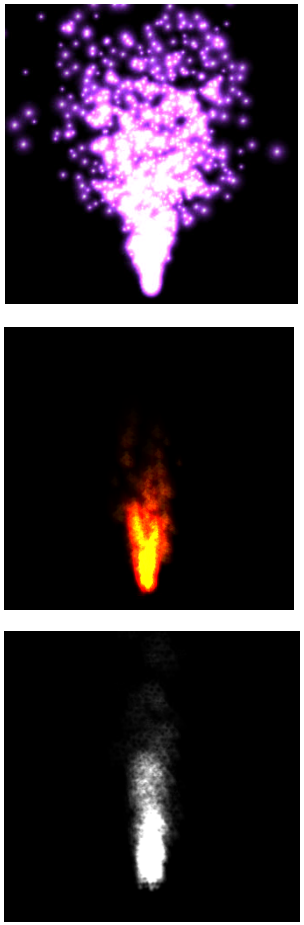


- ポイントスプライト：サンプルコード

```
const float quadratic[] =  { 0.0f, 0.0f, 0.001f };
glPointParameterfv(GL_POINT_DISTANCE_ATTENUATION, quadratic);

glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, m_id);
glTexEnvf(GL_POINT_SPRITE_OES, GL_COORD_REPLACE_OES, GL_TRUE);
glEnable(GL_POINT_SPRITE_OES);
```

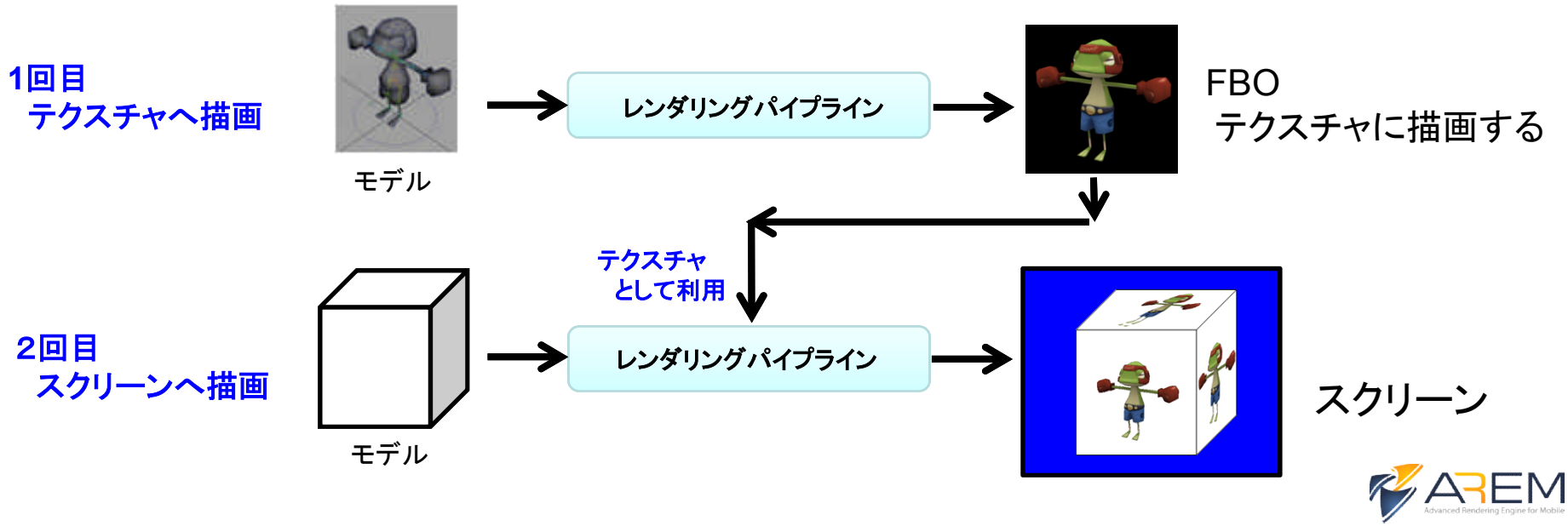
```
glColor4f(  f2vt(1.0f),  f2vt(1.0f),  f2vt(1.0f),  f2vt(1.0f));

glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, vertex);
glEnableClientState(GL_POINT_SIZE_ARRAY_OES);
glPointSizePointerOES(GL_FLOAT,0, pointSize);

glDrawArrays(GL_POINTS, 0, AREM_NUM_PS_BASIC_SPRITE);
```

iPhoneSDK3.0



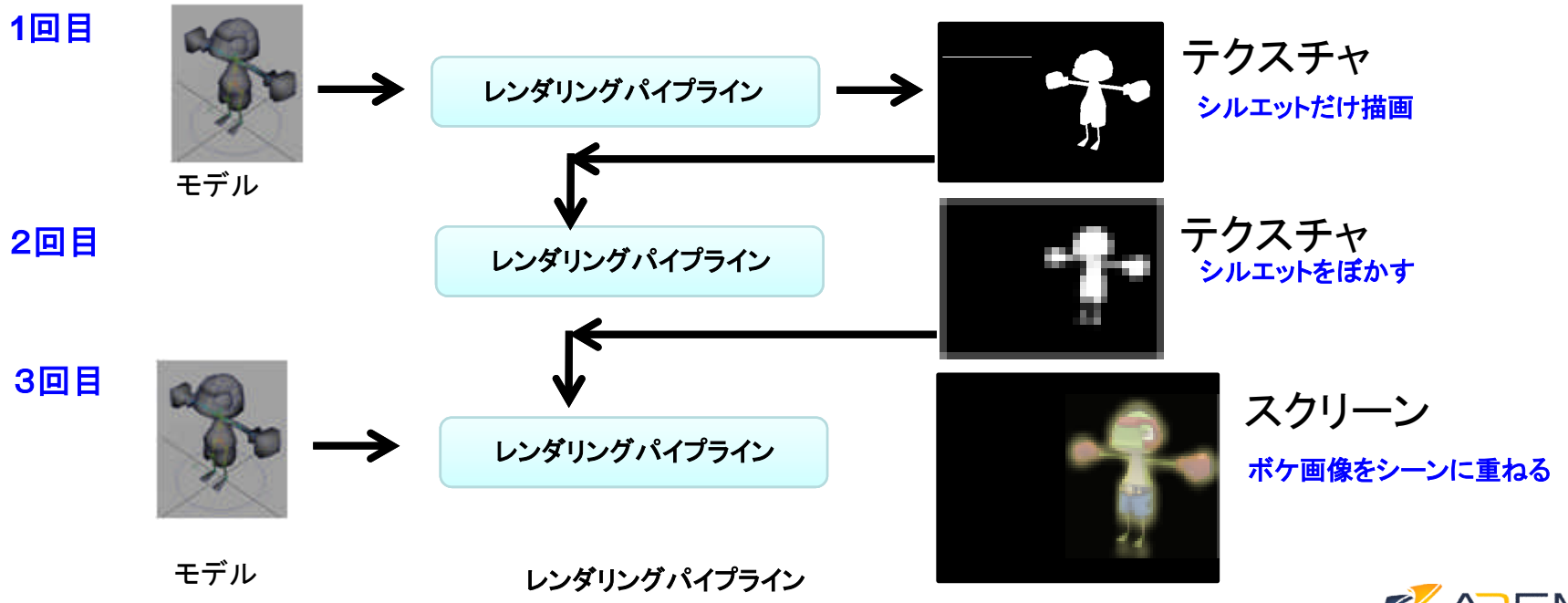
★DEMO-4

- FBO（Frame Buffer Object）
  - FBOを使えば、テクスチャやZバッファへシーンを描画することができる。
  - 描画したテクスチャ等は、別のシーンの描画時に利用可能。



★DEMO-5

- FBOの応用例：ポストエフェクト
  - 描画したシーンにいろいろなエフェクトを加えてゆく
  - 例）キャラだけをボワッと光らせる



- FBO：サンプルコード

テクスチャを描画先に指定する方法

iPhoneSDK3.0

```
GLuint  texturebuffer;

glGenFramebuffersOES(1, & texturebuffer);
glBindFramebufferOES(GL_FRAMEBUFFER_OES, texturebuffer);

glGenTextures(1, &textureHandle);
glBindTexture(GL_TEXTURE_2D, textureHandle);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA,GL_UNSIGNED_BYTE, NULL);

glFramebufferTexture2DOES(GL_FRAMEBUFFER_OES,
                          GL_COLOR_ATTACHMENT0_OES, GL_TEXTURE_2D, textureHandle, 0);

GLuint depthBuf;
glGenRenderbuffersOES(1, & depthBuf);
glBindRenderbufferOES(GL_RENDERBUFFER_OES, depthBuf);
glRenderbufferStorageOES(GL_RENDERBUFFER_OES, GL_DEPTH_COMPONENT16_OES, width, height);
glFramebufferRenderbufferOES(GL_FRAMEBUFFER_OES,
                             GL_DEPTH_ATTACHMENT_OES, GL_RENDERBUFFER_OES, depthBuf);
```

FBO生成

テクスチャ生成

FBOとテクスチャを紐付け

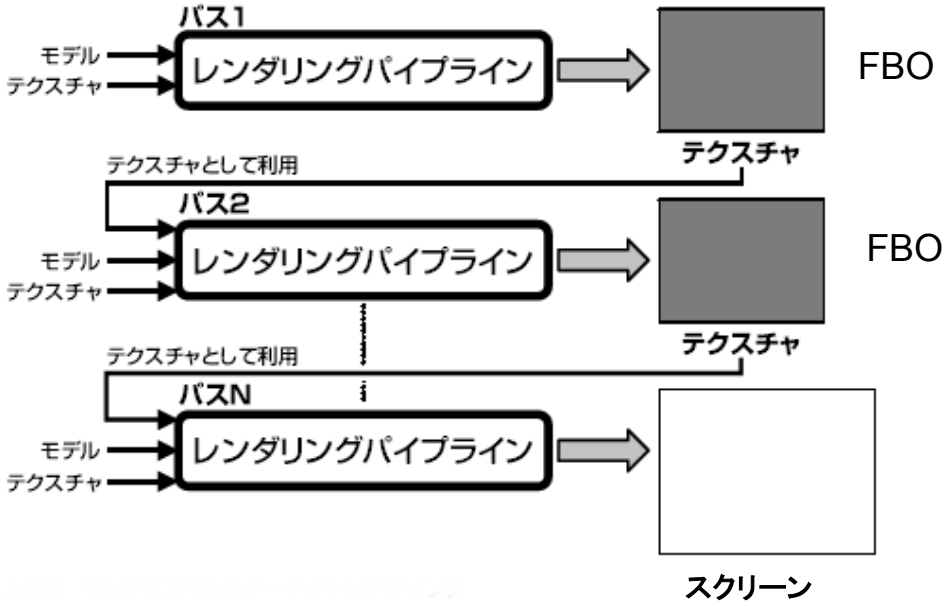
デプスバッファ生成と紐付け

- FBO：サンプルコード
  シーン描画時のFBOの切り替え方法

iPhoneSDK3.0

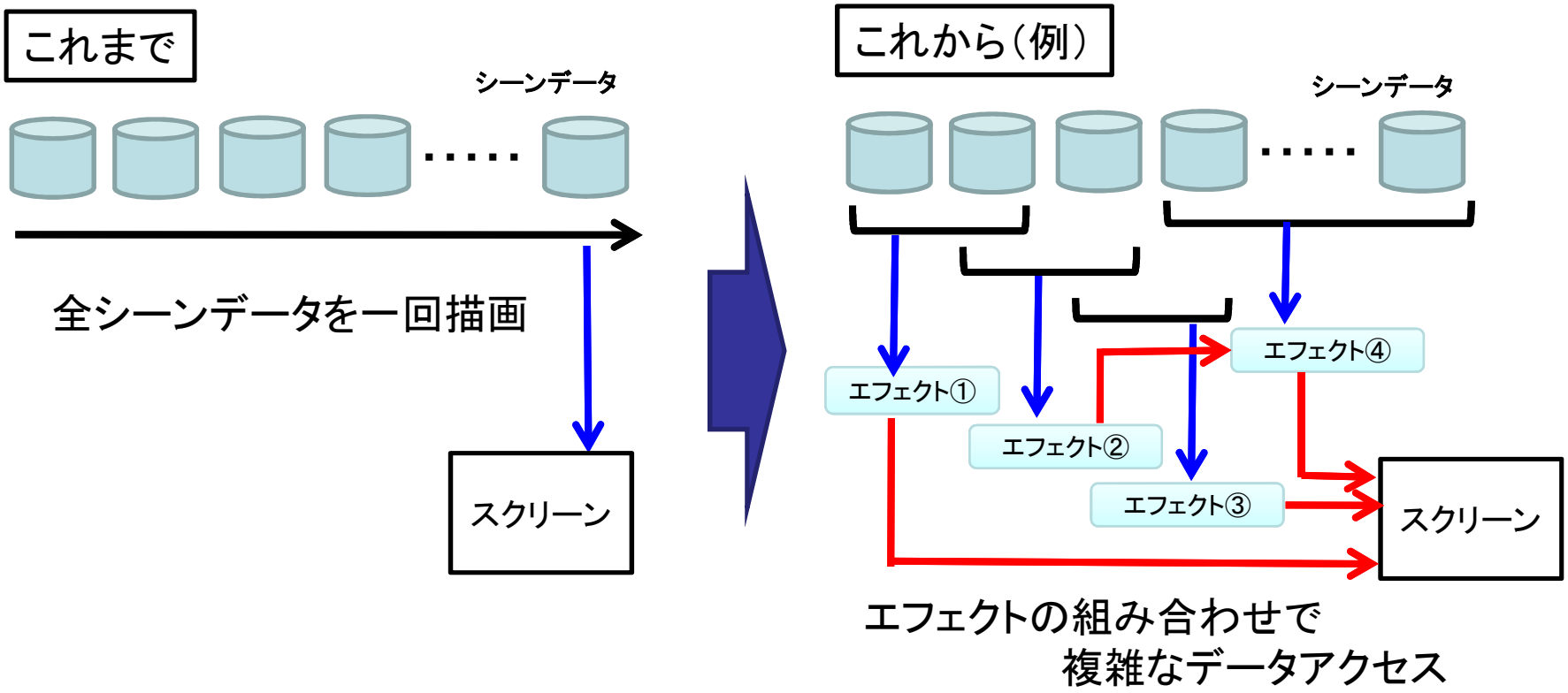
```
glBindFramebufferOES(GL_FRAMEBUFFER_OES, texturebuffer0);

描画処理

glBindFramebufferOES(GL_FRAMEBUFFER_OES, texturebuffer1);
glBindTexture(GL_TEXTURE_2D, textureHandle0);


描画処理・・・1番目ののFBOをテクスチャとして利用


glBindFramebufferOES(GL_FRAMEBUFFER_OES, viewframebuffer);
glBindTexture(GL_TEXTURE_2D, textureHandle1);

描画処理・・・2番目のFBOをテクスチャとして使って、スクリーンへ描画

 glBindRenderbufferOES(GL_RENDERBUFFER_OES, viewrenderbuffer);
 [context presentRenderbuffer:GL_RENDERBUFFER_OES];
```

- FBOと「プログラマブルパイプライン」
  - 非常に相性がよい
    - 美しいエフェクト実現　＝　テクスチャを多段に処理する



しかし、本格的なエフェクトを実現するには、
現GPUでは、パフォーマンスが少し足りないか？

- これからのモバイル向け3Dアプリの
  アーキテクチャはこうなる



OpenGL-ES1.1からOpenGL-ES2.0への完全移行は、2010年の後半からか？
現時点では、ソフト＆データの構造をOpenGL-ES2.0世代へ対応できるようにしておく。

- 2D描画
  - OS標準のGUI描画APIとOpenGL-ESの相性は悪い
    - GUIをOpenGL-ESで構築する必要性
  - glDrawTexXXXを使えば簡単
    - スクリーン座標で描画できる
    - アルファテストやアルファブレンディングも可能

- 2D描画：サンプルコード

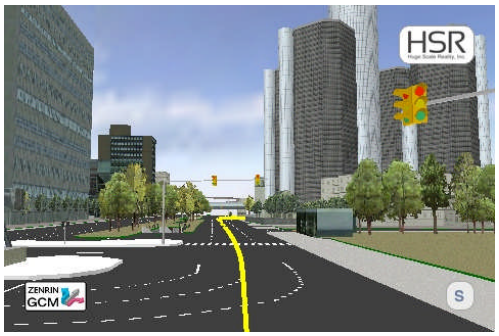
iPhoneSDK3.0

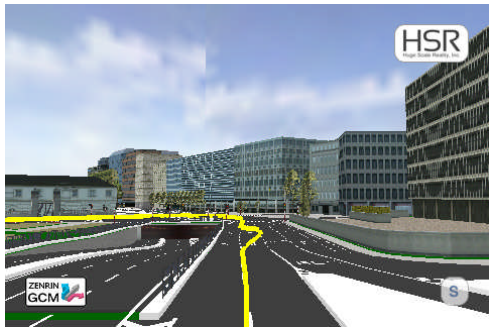
```
glBindTexture(GL_TEXTURE_2D, m_texture);
GLint rect[] = {0, 0, width, height};
glTexParameteriv(GL_TEXTURE_2D,  GL_TEXTURE_CROP_RECT_OES, rect);
glDrawTexfOES(x, y, z,  width*sx, height*sy);
```

謝辞

謝辞

- 株式会社ゼンリン
  - AREMを使用したPND（PortableNaviDevice）向けソリューションを提供中
  - 3次元の街並み地図データ「GCM（Guidable City Model）」をご提供





謝辞

- デジタルハリウッド　CreativeLab：山本Lab
  - iPhone上でAREMを使用したゲームを開発中
  - キャラクタ等コンテンツ提供

デザイナ：

飴田 慎士　川田 和賜

伊藤 大輝　久山 雄作　黒木　厚典

テクニカルデザイナ

篠山　範明

ディレクター

山本　浩司

PlanetSavers開発中画面

謝辞

- デジタルハリウッド　OB/OG

  iPhone上でAREMを使用したゲームを開発中

デザイナ:

足津 真季　　中田 陽平　　山口 愛乃

梅村 有里　　島 麻美

開発中画面

Q&A

連絡先

- support@hs-reality.com
  - 本日の講演に関してご質問等受け付けます。
- http://www.hs-reality.com/
  - 講演資料、サンプルなどをアップする予定です。