GPGPUトレンド現状・今後の動向及び DMPにおける取り組みについて

株式会社ディジタルメディアプロフェッショナル
勝又　大満

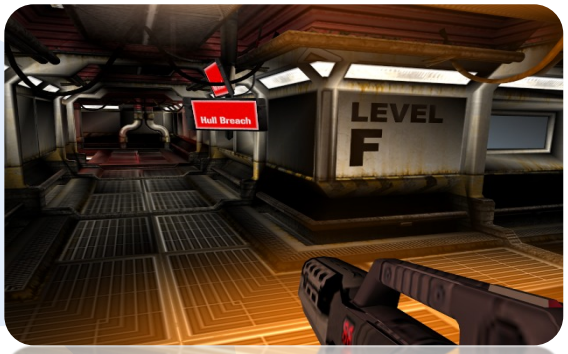
Jan/2013

Outline

- DMPの取り組み
- GPGPUの業界動向
- DMPグラフィックス・コンピューティング　アーキテクチャ

DMP グラフィックス・コンピューティングIPソリューション

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 3.0)
最大128SPコア

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 (OpenGL ES 1.1, OpenVG1.1)
SMAPH-H2 (OpenGL ES 3.0, OpenVG1.1)

Case study - PICA200

Case study: PICA200開発で目指したもの

- 低消費電力 + 高機能・高性能を両立するグラフィックス・アーキテクチャの開発
  - ユーザーニーズに適合した独自表現方法(Configurable shader)
  - 実装アルゴリズム最適化による、低消費電力ハードウェア
  - 少ないデータ量、バス帯域、コンテンツサイズで豊かなCG表現を実現
  - スケーラブルなアーキテクチャー（ローエンドからハイエンドまで対応可能）

本発表におけるPICA200のContribution

- 頂点、ジオメトリシェーダにおけるメモリトラフィック、演算最適化
- コンフィギュラブルなフラグメントシェーダ

上記技術を組み込みシステムで実装可能にした

Mikage scene example

- BASE
- Environment cube map
- Planar reflection
- Pixel lighting
- Bump mapping
- Procedural texture
- Polygon subdivision
- Shadow

PICA200アーキテクチャ

PICA200 - 設計目標

- 組込みシステムで高品位なグラフィックスを提供したい
  - 3Dグラフィックスアルゴリズムの共通項をモデリングを行いこのモデルを実現するハードウェアアクセラレータを導入
  - ジオメトリシェーダなどによるメモリバンド幅の削減の仕掛けを導入

この結果

- 低消費電力、高性能、高品質な3Dグラフィックスハードウェアを実現

PICA200アーキテクチャ

- High Performance Graphics 2011で発表した内容＋αについて今回説明
  - Max Kazakov, Eisaku Ohbuchi: Primitive Processing and Advanced Shading Architecture for Embedded Space. High Performance Graphics 2011: 169-176

Primitive processing and advanced shading architecture for embedded space

Max Kazakov*
Digital Media Professionals, Inc.

Eisaku Ohbuchi†
Digital Media Professionals, Inc.

Figure 1: *The extended 3D Mark Mobile OpenGL ES 1.X (left) and 2.0 (right) engines running on ASIC implementations of our architecture. Extensions include primitive processing and per-fragment shading presented in the paper with content featuring on-chip subdivision for the curtain, coal pot and bear, on-chip silhouette rendering-based soft shadow mapping and particle systems rendering with application-specified geometry shaders, Cook-Torrance shading on the main characters and other scene objects and Fresnel-like planar reflections on the floor.*

Abstract

This paper presents a new graphics architecture enabling content-rich applications for the embedded space by extending hardware architecture in two main areas - geometry processing and configurable per-fragment shading. Our first contribution combines vertex cache and a programmable geometry engine that handles both fixed and variable size geometrical primitives completely on-chip. It enables subdivision surface tessellation, silhouette rendering and other geometry processing algorithms to be implemented in one pass and without external memory access. Our second contribution is in configurable per-fragment shading that is mainly a dot product + lookup table machine being versatile enough to realize Cook-Torrance shading, Schlick anisotropy model and others.

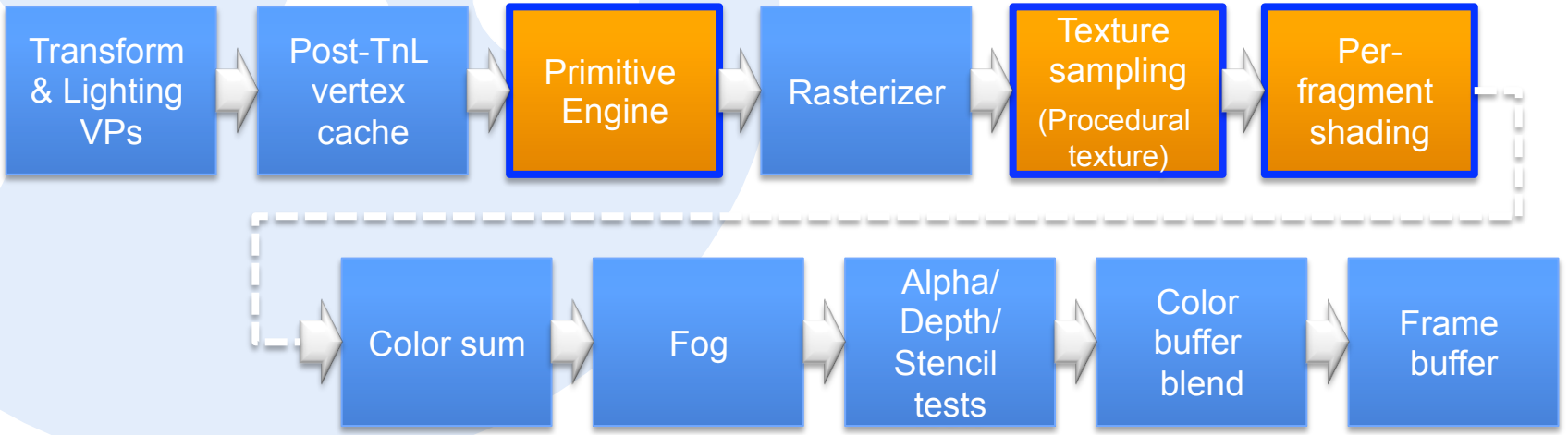
Memory storage and memory bandwidth are reduced in proposed

1 Introduction

3D graphics has become common in a variety of applications on numerous embedded platforms. Mobile phones, car navigation systems, game consoles, cameras etc often provide rendering capabilities used in games, 3D interfaces and so on. While low-poly models and simple shading is often an only option for mobile phone applications, those for car navigation and game consoles usually have access to several times bigger screens, both in resolution and physical dimensions. Here, power source is not so limiting, and applications rival with their desktop counterparts in terms of visual effects, but memory bandwidth might still be a constraint. Custom architectures and proprietary solutions prevail in this area, targeting high performance in the XBox 360 case [Andrews and Baker 2006] and reduced silicon size and power consumption for Sony PSP [Imai

PICA200アーキテクチャ

- プログラマブルなジオメトリ処理と固定処理のフラグメントシェーダを搭載
- OpenGL ES 1.1パイプラインを拡張して実装
  - プリミティブエンジン：頂点シェーダを拡張
  - プロシージャルテクスチャ：専用テクスチャサンプラとして拡張
  - 固定フラグメントシェーダ
    - » フラグメントごとに補間されたベクトル情報、バンプ・タンジェント・シャド・テクスチャマップを入力しシェーディング処理を実施

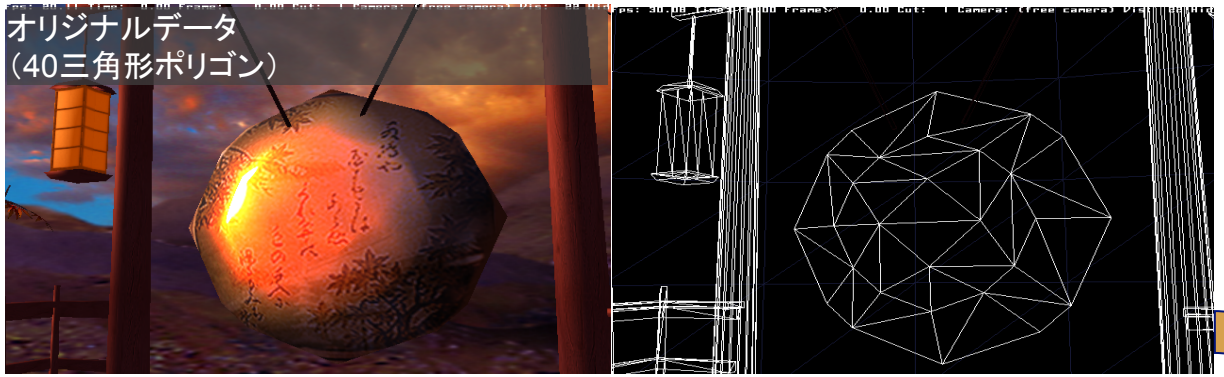


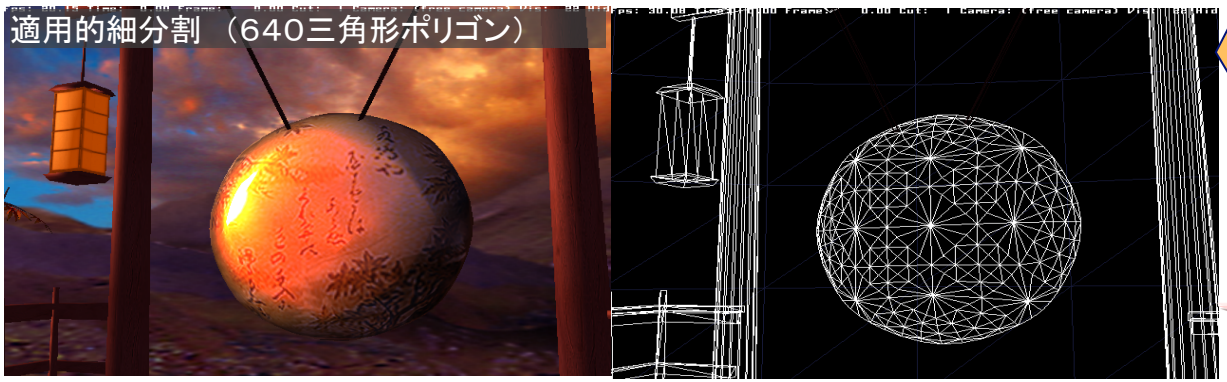
PICA200アーキテクチャ

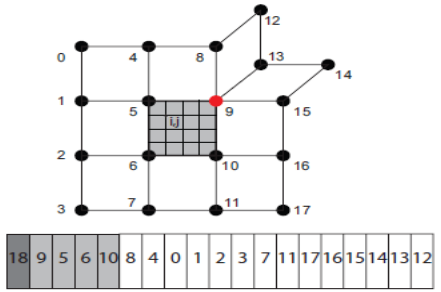
ジオメトリエンジン

ジオメトリシェーダの実装：サブディビジョン例

■少ないポリゴンのデータをプロセッサ内でハイポリゴン化する
■独自プリミティブジェネレーションアーキテクチャで実現







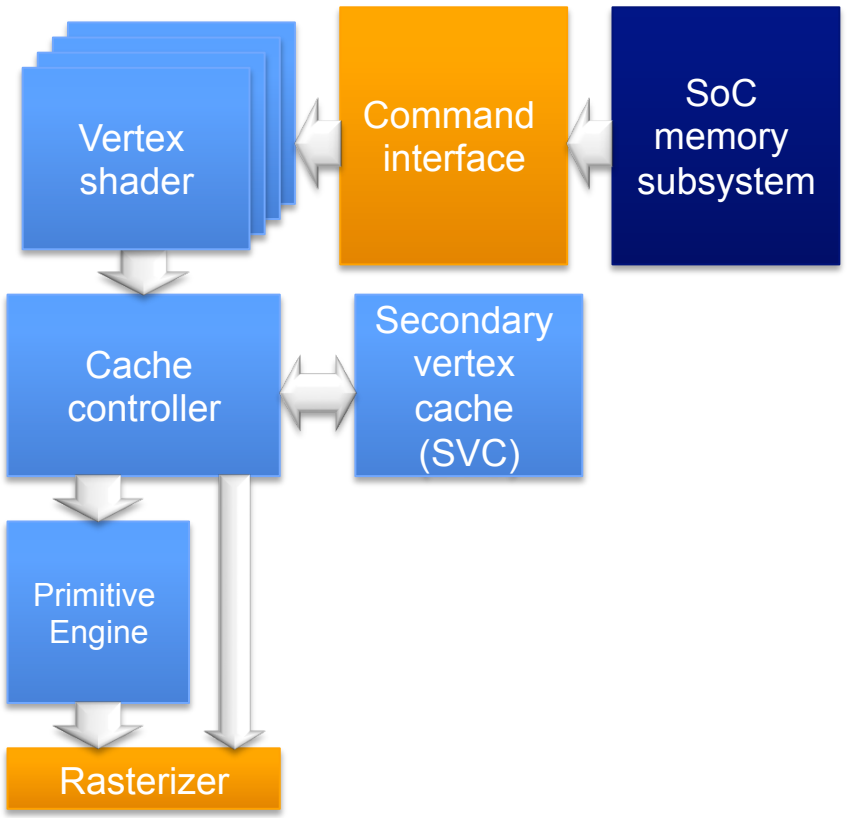
入力データ量：約1 / 16

少ないポリゴン入力数でハイクオリティな3D描画を実現

ソフトシャドウ実現のため、シルエットレンダリングにも本技術を使用

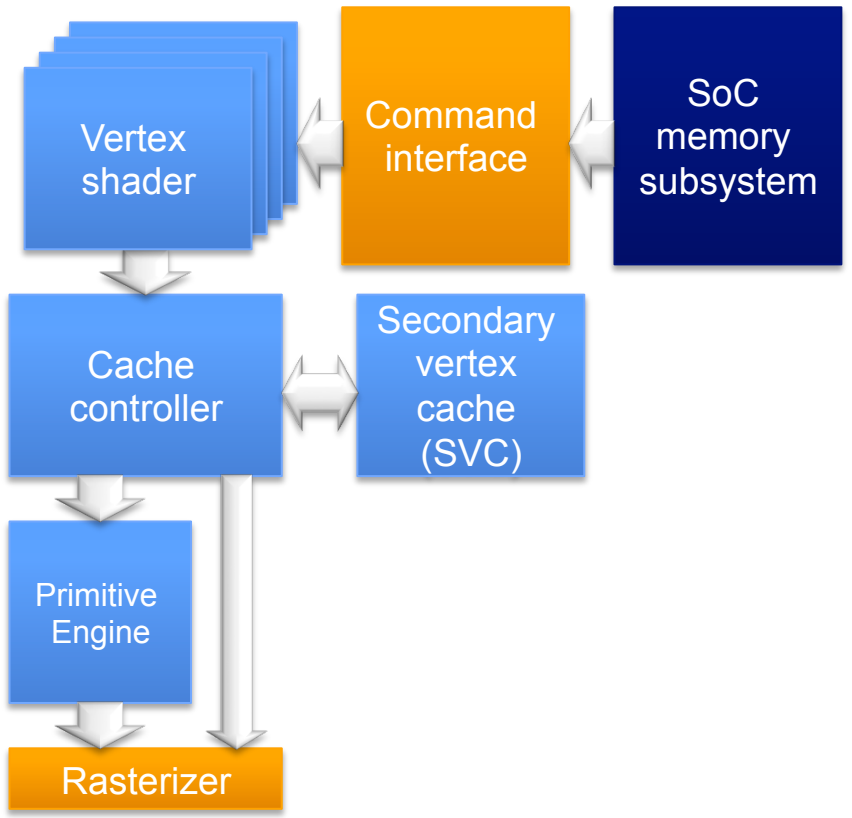
ジオメトリエンジン

- Shader Model 3.0レベルの頂点シェーダ
- プリミティブエンジン+2次頂点キャッシュ(SVC)によるジオメトリシェーダ機能を実現
  - プリミティブエンジンは頂点シェーダにプログラマブルなプリミティブ出力を追加したモジュール
- 固定、可変長サイズのジオメトリプリミティブの扱いが可能
  - 扱えるプリミティブ最大サイズは、SVCのサイズで規定



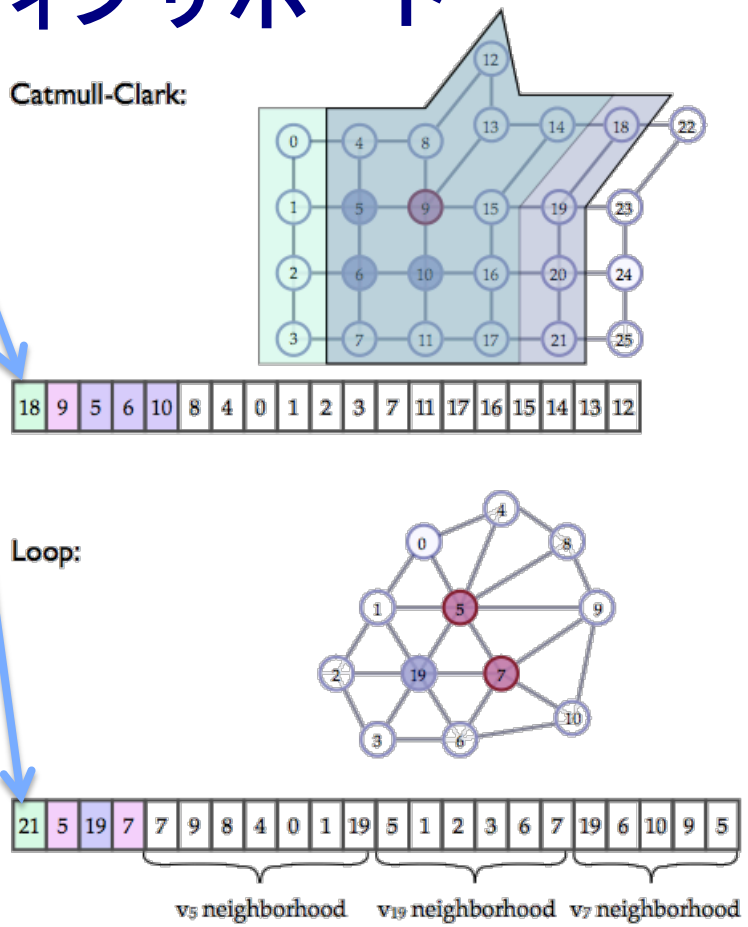
ジオメトリエンジン（cont'd）

- ジオメトリシェーダ向けジオメトリ情報の入力はすべてSVCから入力することで、テクスチャアクセスを削減
- SVCにより演算済み頂点データを再利用
  - 頂点バッファのトラフィックを削減
  - 複数頂点プリミティブや複雑なジオメトリ処理アルゴリズム向けに活用
- 内部頂点アトリビュートのトラフィックも削減可能
  - 少ない初期プリミティブ頂点入力からフルセットのアトリビュート生成
- 回路サイズを最適化
  - 頂点シェーダをプリミティブエンジンに転用可能
  - SVCの回路変更を最小化



可変サイズのプリミティブサポート

- インデックスバッファ上の頭にプリミティブサイズを格納
- ジオメトリシェーダ向けに可変長プリミティブサイズの扱いが可能
- サブディビジョン実装例
  - 可変パッチサイズシーケンスのサポート
  - 1つのシェーダですべてのパッチ向けの記述をサポート
  - 頂点の接続情報を読み出すためのテクスチャアクセスはなし
  - サブディビジョンパッチをできるだけ大きな単位かつ、シェアして定義することで、頂点キャッシュのヒット率向上が可能



PICA200アーキテクチャ

フラグメントシェーダ

フラグメントライティングのアクセラレーション

■ PCグラフィックスの表現力を組込みグラフィックスで実現
■独自最適化アルゴリズムを開発し、ハードウェア実装


OpenGL ES 1.1 (頂点ライティング)


フラグメント・ライティング (拡散光)


フラグメントライティング(拡散光＋鏡面反射光)


フラグメントライティング(拡散光＋鏡面反射光)
＋ バンプマップ


フラグメントライティング(拡散光＋鏡面反射光)
＋ バンプマップ ＋ キューブマップ

フラグメントシェーダ

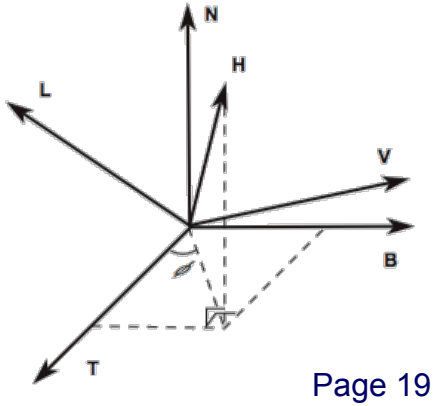
- 複数光源、バンプ・タンジェントマップ、シャドマップのサポート
- スクリーン座標系でのノーマル情報をクオータニオン形式の入力から生成

$$C_p = m_e + m_a s_a + \sum_{i=0}^{n-1} AS(d_0) f_i H(m_d l_{id} + m_d l_{id}(\mathbf{L}\cdot\mathbf{N})) \quad (1)$$

$$C_{s\lambda} = \sum_{i=0}^{n-1} AS(d_0) f_i H(m_{s\lambda} D_0(d_1) G_0 + R_\lambda(d_2) D_1(d_3) G_1) I_{si\lambda} \quad (2)$$

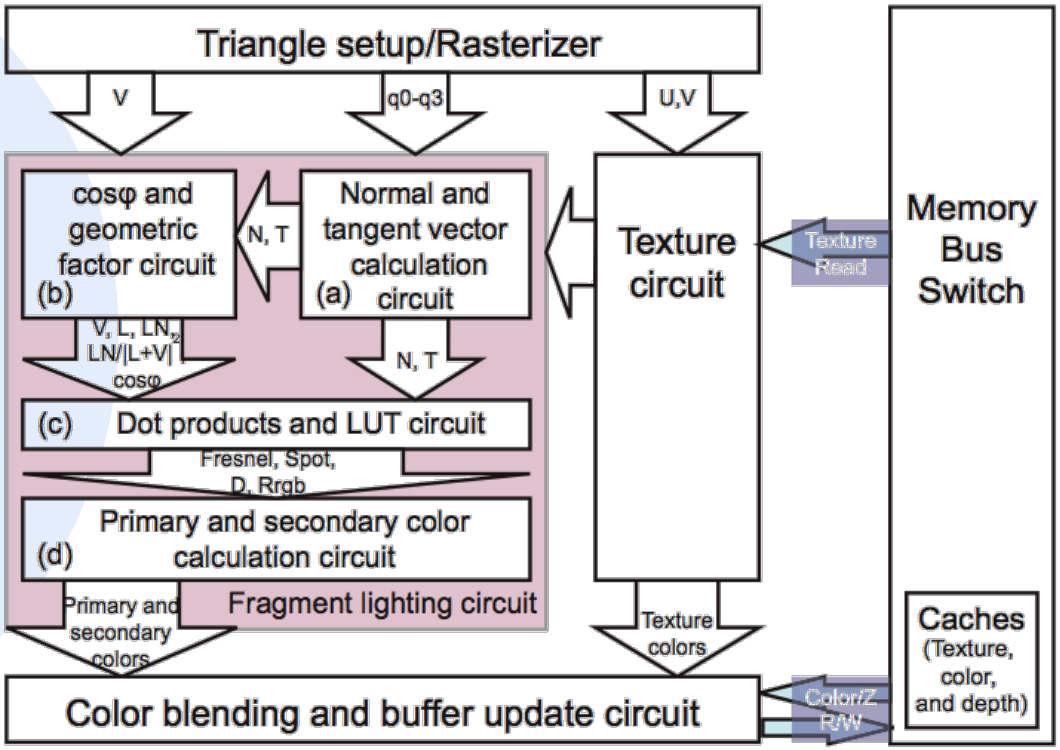
$$G_{0,1} = G' \quad or \quad 1,$$
$$G' = (\mathbf{L}\cdot\mathbf{N}) / |\mathbf{L}+\mathbf{V}|^2$$

Angle ϕ:

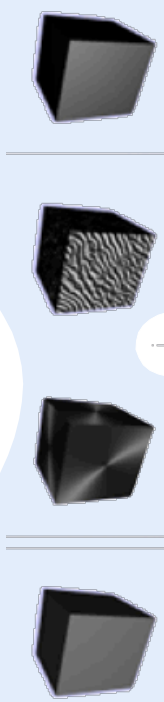


フラグメントシェーダ (cont'd)

- 30-35段の固定機能パイプライン
- オンチップSRAMのLookup table (LUT)へシェーディング関数を格納
- フラグメントシェーダの処理時間は固定
  - 1-4サイクル/フラグメント/光源 （コンフィギュレーションごとに定義される）
  - これにより性能の予測可能性を実現



フラグメントシェーダ性能



Shading model	Clk/frag	SM 3.0 asm steps
Phong shading model $D_0 = cos^{s}(\mathbf{N} \cdot \mathbf{L})$ $G_{0,1}=1$	1	35
Phong + bump $D_0 = cos^{s}(\mathbf{N'} \cdot \mathbf{L})$ $G_{0,1}=1$	1	38
Schlick anisotropic model $D_1 = Z(\mathbf{N} \cdot \mathbf{H}), R_\lambda=F_\lambda(\mathbf{V} \cdot \mathbf{H})$ $S=A(cos\phi), G_{0,1}=G'$	4	61
Cook-Torrance shading model $D_1 = D(\mathbf{N} \cdot \mathbf{H}), R_\lambda=F_\lambda(\mathbf{V} \cdot \mathbf{H})$ $G_{0,1}=G'$	2	48

プロシージャルテクスチャ

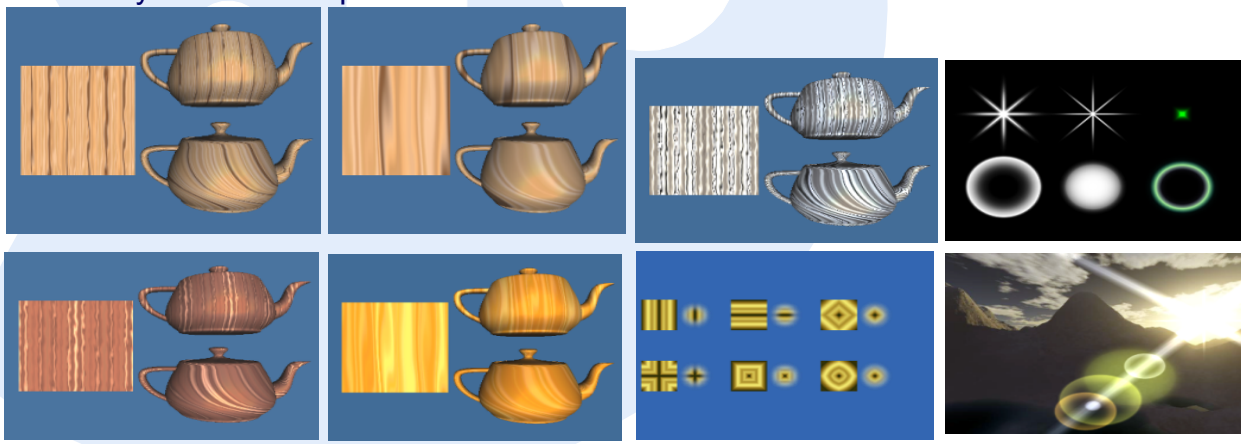
■ テクスチャバッファを使わずにIP内部の回路にてテクスチャパターンを動的に生成する技術
■コンテンツサイズ削減・性能改善

Target texture：256x256 (RGBA 8888) case

モード	入力テクスチャサイズ
通常のテクスチャ	256 Kbyte
プロシージャルテクスチャ	3Kbyte

入力テクスチャパターンのサイズを1 / 85 に削減

Texture synthesis examples



パターン生成例

Case study: PICA200開発で目指したもの

- 低消費電力 + 高機能・高性能を両立するグラフィックス・アーキテクチャの開発
  - 3Dの世界ではシェーディング手段として、ある程度決まった手段がよく使われるため、シェーダのアルゴリズムの共通項を抜き出してハードウェア化したのがPICA200。
  - 多くの場合，消費電力の点でハードウェア化はきわめて有利になる。
  - その点に着目し、共通処理は専用HWを用いることで高速化、低消費電力を行った。

GPGPUの業界動向

GPGPUに関わる主な規格

Standard	Owner	概要
OpenCL	Khronos	包括的なGPGPU向けAPI、言語環境
WebCL	Khronos	OpenCLのJavascriptバインディング
CUDA	NVIDIA	NVIDIAのハードウェアに特化したGPGPU環境
Aparapi	AMD	Java+GPU向け環境 BytecodeをOpenCL,CPUにマッピング可能
HSA	HSA foundation	AMDが中心となり提唱されているAPU向けヘテロジニアスシステムアーキテクチャ
Renderscript Compute	Google	Android上のCompute拡張.
Direct Compute	Microsoft	Computing版DirectX. Windowsのみで動作
C++ AMP	Microsoft	DirectCompute上で動作 GPU向けC++の言語拡張
OpenGL4.3	Khronos	Compute shader機能の追加

最後に

まとめ

- W/Wで見た場合、GPU + CPU = APUの流れは、モバイル、デスクトップの両方で次のトレンドとしてとらえられている
- 同時にAPU環境を生かしたアプリケーションに関わる活動も活発
  - KhronosのVision WG
  - Stanford, NokiaのFcam(http://fcam.garage.maemo.org)
  - NVIDIAのOpenCV　CUDAサポート
  - SIGGRAPH2012で発表のあった”Decoupling algorithms from schedules for easy optimization of image processing pipelines”のHalide言語

  と、気がついたらモバイル分野ではComputer Vision系が多数。
- CPUコアは、PPC,MIPS,ARM程度、GPUコアもヨーロッパ、アメリカがメインで、アジアにはメジャーなコアが少ない。

Thank you!