



## PhysX PS3の高速化



- PS3 Devnetにて現在配信中
- 2008年9月 - SDKアップグレード - 大幅なPS3向けの最適化
  - 機能追加を予定していない。
  - PS3の最適化
    - SPUによるブロードフェーズの並列処理
    - 衝突判定に対するミッド・ナローフェーズのSPU上最適化
    - 制約の生成とソルバーを分離させる
    - 大きなアイランドの分割とSPUによる分散処理
- 場合により、60%以上のパフォーマンスアップが見られている！

PhysX

## ナローフェーズの最適化



- ナローフェーズの組み合わせを全てSPU化
- メッシュ用のナローとミッドフェーズを統合させる
  - 同じSPUプログラムでナローフェーズとミッドフェーズを行う
  - ミッドフェーズから取得した三角がメモリに残る。
- コンタクトポイントには制限なし

PhysX

## PhysX SDK: Pushing the Envelope on Every Platform



### 一般的な物理機能

- 剛体
- 衝突判定
- コンストレイント

### リジッドを超える新機能(beyond the basics)

- クロス: 敗れるクロス、LODクロス
- 変形可能なメタルクロス: 破壊のオブジェクト、植物など
- 流体: SPH "Smart Particles"
- 柔軟体
- ツール: Max/Maya、COLLADA対応、VisualDebugger、APEX

PhysX

## GPUならではの大量物理



- CUDAを用いて、Geforce 8シリーズ以上に対等
- 推奨の機能
  - パーティクルシステム
  - 柔軟体

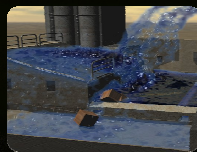


PhysX

## PhysX機能 - 大量パーティクル



- 背景や剛体との衝突
- 外力から影響されている
- パーティクル同士でも影響しあっている

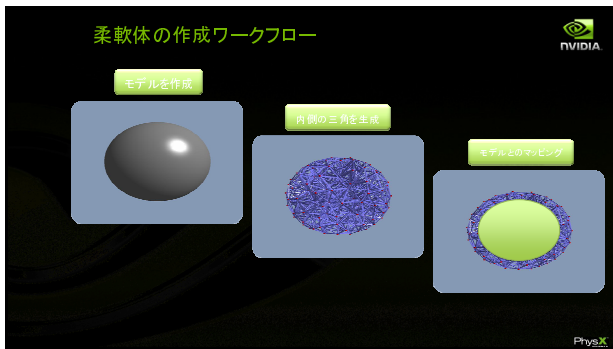


PhysX

## 大量パーティクルシステムのデモ



PhysX



## APEX Modules: Clothing (洋服モジュール)

- APEX Clothing Module
  - LODの制御
  - 部分的または全体的なシミュレーション
  - ツールサポート
  - スケルトンとのコンストレイント作成
  - 敷けるクロス
  - いくつかのマテリアルを提供



PhysX

# CUDA

CUDAとは:

並列コンピューティング用の拡大可能なプログラミングモデルと開発環境

拡張付きのなじみやすいC/C++環境

PhysX

## ゲーム開発では、CUDAを使って何が出来る??

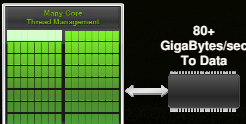
例えば...

- ルート探索
- コンテンツ作成の高速化
  - ライトマップやシャドウマップの生成、大域照明の計算
  - テクスチャの圧縮や解凍
- 高画質ビデオの形式変換や再生
- 等々

PhysX

## GPU=高並列性のプロセッサ

- GPUは、
  - 広帯域幅のある専用メモリを持つ
  - 数多くのスレッドを並列的に実行できる
- アプリの中では、並列化できる部分をカーネル(kernel)として処理する
  - カーネルはプログラムとなり、複数のスレッドによって処理される
- CUDAのスレッド
  - 非常に軽い
  - 生成時のオーバーヘッドが小さい
  - 切り替えが早い
  - GPUが数1000もスレッドを同時に実行できる



PhysX

## 並列的なスレッドの配列

- CUDAのカーネルはスレッドの配列として処理される
  - スレッドはすべて同じプログラムを実行する
  - 各スレッドが固有のIDで、対象のメモリアドレスを計算し、プログラム制御が出来る

```

threadID [ 0 1 2 3 4 5 6 7 ]
          | | | | | | | |
          v v v v v v v v
...
float x = input[threadID];
float y = func(x);
output[threadID] = y;
          | | | | | | | |
          v v v v v v v v
  
```

PhysX

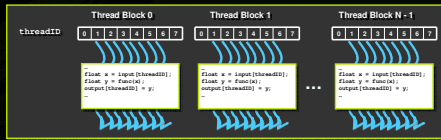
## スレッド同士のやり取り

- 結果を共有することによって、計算が省ける
- メモリアクセスの共有によって、大規模な帯域幅の節約
- このスレッド協力がCUDAの主な特徴の一つ
  - 共有メモリと同期により、チップ上のスレッドが簡単に協力できる

PhysX

## スレッドのブロック: 拡大可能な協力n

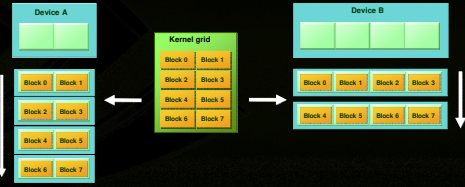
- 大きなスレッド配列を複数のブロックに分割する
  - ブロック内のスレッドが共有メモリにより取り回す
  - ただ、違うブロックのスレッドとはやり取りができない
- **プログラムの処理速度が、プロセッサの数に合わせて、透明的に拡大できる!**



PhysX

## 透明性のある拡大性

- ハードウェアは自由にスレッドをプロセッサ一別に
- カーネルの処理性能が任意の数のプロセッサに拡大できる



PhysX

## 簡単な“C”拡張で並列処理のプログラムを組める

### 標準のCコード

```
void saxpy_serial(int n, float a, float *x, float *y)
{
    for (int i = 0; i < n; ++i)
        y[i] = a * x[i] + y[i];
}
// シリアルなSAXPY カーネルを実行
saxpy_serial(n, 2.0, x, y);
```

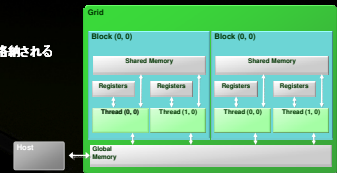
### CUDAのCコード

```
_global_ void saxpy_parallel(int n, float a, float *x, float *y)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < n) y[i] = a * x[i] + y[i];
}
// 並列的なSAXPY カーネルを実行
// 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

PhysX

## カーネルのメモリアクセス

- レジスター
  - **グローバルメモリ**
    - カーネルの入出力用のデータが格納される
    - テップ外、サイズが大きい
    - キャッシュなし
  - **共有メモリ**
    - ブロックのスレッドが共有する
    - テップ上、サイズが
    - レジスター並みの高速性
- ホストがグローバルメモリの読み込み・書き込みが出来るが、共有メモリをアクセスできない。

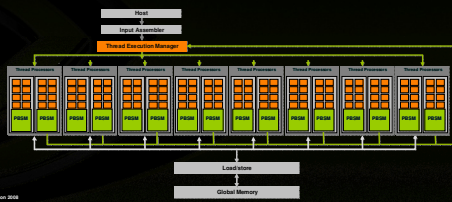


© NVIDIA Corporation 2008

PhysX

## 複数コアを持つGPUの実例

- G80 (Nov 2006 出荷 - GeForce 8800 GTX)
- 128 スレッドプロセッサ (16 マルチプロセッサ) がスレッドを実行する
- 実行可能なスレッド数は12,288まで
- ブロック毎の共有メモリ (PBSM) が処理を高速化してくれる



© NVIDIA Corporation 2008

PhysX

## ゲーム用のCUDA

- 描画
- 物理 (PhysX)
- 人工知能
- 画像処理

PhysX

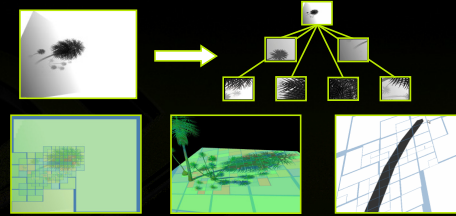
## ハードウェアによるグラフィックス高速化の3段階



- Fixed-Function Graphics
  - マルテクスチャ、マルチパスの描画
- Programmable Shading
  - 頂点やピクセルシェーダーの登場
  - 手続き系テクスチャの生成、複雑な照明など
- Programmable Graphics
  - 複数の並列的に動いているスレッドが複雑なデータ構造を共有しながら、グラフィックスのアルゴリズムを実行する。

PhysX

## Quadtreeのシャドウマップ



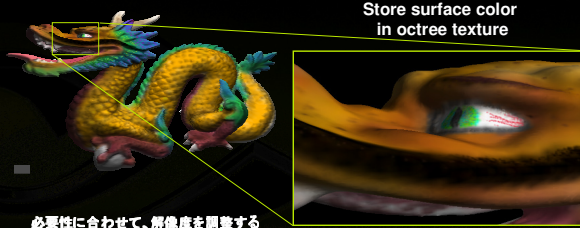
"Adaptive Shadow Maps," Fernando et al., SIGGRAPH 2001

"Resolution Matched Shadow Maps", Aaron E. Lefohn, Shubhabrata Sengupta, John D. Owens. ACM Transactions on Graphics, Oct. 2007.

Courtesy of Aaron Lefohn

PhysX

## Octree 3D Paint



必要性に合わせて、解像度を調整する

2048<sup>2</sup> Effective Texture Resolution

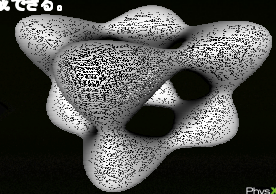
Courtesy of Aaron Lefohn

PhysX

## 手続き系のジオメトリのためのCUDA



- CUDAがジオメトリの自動生成を効率的に行える。
  - 例えば、D3Dなどの頂点バッファが生成できる
- ジオメトリシェーダーより効率的に生成できる。
- 例えば、マーチングキューブ法、曲面の抽出、GeForce 8800 GTXでは、ジオメトリシェーダーより5倍の早い。



PhysX

## ソートのアルゴリズム



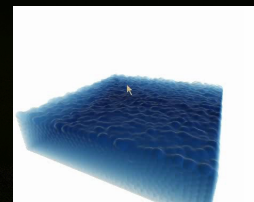
- すべてのコンピューティング分野では、ソートは重大なアルゴリズムの一つ
- CUDAでは、最高に早いソート・アルゴリズムが実装可能
- 高度なテクニックを可能にする
  - レイトレーシング
  - 大域照明
  - TessellationとSubdivision Surfaces
  - 等々

PhysX

## CUDAによる流体シミュレーション




- 従来のグリッド型の流体シミュレーションの制限
  - 処理が高い、形状はボックスの中の流体のみ
- Smoothed Particle Hydrodynamics (SPH) では、流体を大量のパーティクルとしてシミュレートする
  - PhysXがCUDAにより採用しているテクニック
- 計算を高速化するため、CUDAがグリッドのデータ構造を動的に生成できる。
- GeForce 8800 GTXでは、32Kのパーティクルを60fpsでシミュレートできる



## A\* アルゴリズム

- 多く採用されているルート探索アルゴリズム
  - A\* は、並列性が低い
  - ただ、複数のルートを同時に計算することで、並列例が高まる
    - 複数の移動中キャラクター
    - 数多くのキャラクターが住むワールドが可能になる



## 画像処理とCUDA

- 任意のメモリアクセス (scatter) と共有メモリによって、効率的な画像処理のアルゴリズムが実装可能になる
  - ヒストグラム
  - Convolutions
  - DXT 圧縮
- DX10などのグラフィックスAPIと比べ、パフォーマンスが高い

## ヒストグラム


- イメージの色の分布
- 適応できるアプリ: イメージ分析, HDR用トーンマッピング, ...
- CPUと比べ、CUDAが8倍も早い!



Reinhard HDR Tonemapping operator      HDR in Valve's source engine

## 分離型のカーネル

- 実例:
  - 画像処理: ぼかし, シャープ化, 輪廓検出など
  - ミップマップの生成
  - Subsurface scattering (表面下散乱)

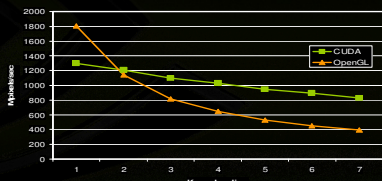


元の画像      ぼかしフィルター      輪廓検出のフィルター

## CUDAの画像処理

- OpenGLのピクセルシェーダーと比べて、2倍の画像処理速度
  - 共有メモリによってデータ再使用の効率が增加


Separable Image Convolution Performance



Kernel radius	CUDA (Mtexels/sec)	OpenGL (Mtexels/sec)
1	~1300	~1800
2	~1100	~1100
3	~1000	~800
4	~900	~600
5	~800	~500
6	~700	~400
7	~600	~300

## DXTの圧縮

- オフライン
- オンライン
- リアルタイム



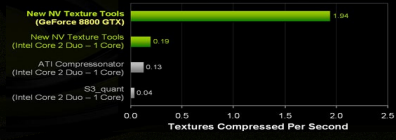
256x256 RGB = 256 kB      128x128 RGB = 64 kB      256x256 DXT1 = 32 kB

# オフラインの DXT 圧縮



## ● 最高質の圧縮

	GeForce 8800 GTX	GeForce 8800 GTS	GeForce 8600	Intel Core 2 X2600 @ 2.9GHz	AMD Athlon64 Dual Core 4400
Lenx 512x512	42 ms	56 ms	155 ms	563 ms	1,251 ms



PhysX