
NVIDIA
PhysX™
by NVIDIA
PhysX
by NVIDIA

PhysXが「グリーン」になった！

描画と表示

動きと接触

Great Visuals - Great Gameplay

NVIDIA PhysX is Everywhere

- 最も採用されているAPI
  - *150タイトル以上*
  - *25,000のユーザー登録*
  - *世界中の開発者・パブリッシャー*
- 幅広い採用
  - *ゲームエンジン（UE3など）*
  - *ミドルウェア（NaturalMotion）*
  - *DCCツール*
  - *次世代ゲーム機とPCの最適化済み*

PhysXの実績

- **Epic UE3 Engine**
  - 業界最強の総合エンジン：開発中のタイトルが200本以上
  - PhysXは既に全体的に組み込まれている

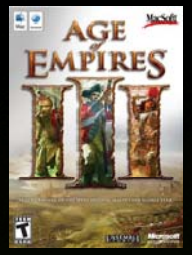
- **Emergent Gamebryo**
  - #3の総合エンジン
  - PhysX組み込み

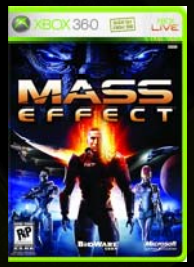
- **その他**
  - Grin、Bioware，EAなど多くの開発会社の社内エンジンでも、PhysXを採用！

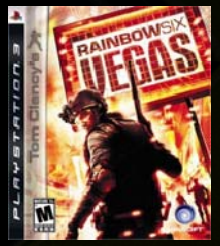
Commitment to ALL Platforms: not just PC

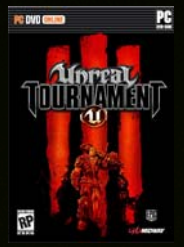
PhysX PS3	NVIDIAが最適化チームを発動して、最適化を進行中。
Wii	ソースとバイナリを公開中
XBOX 360	ソースとバイナリを公開中
Multi-Platform	今後の新機能は、変わりなく、各機種に対等させる予定

各ゲーム機種の採用タイトルの例









PLAYSTATION 3

PhysX by NVIDIA

PhysX PS3の高速化

- PS3 Devnetにて現在配信中
- 2008年9月 – SDKアップグレード一大幅な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！

ナローフェーズの最適化

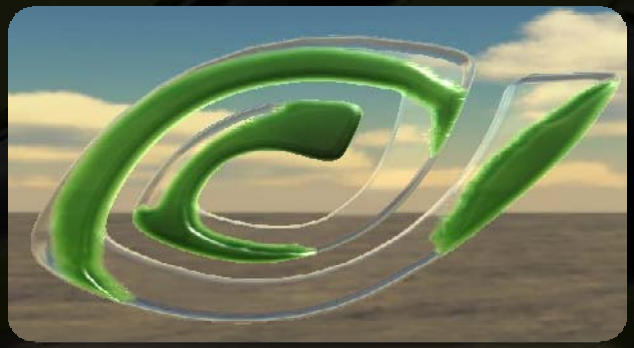
- ナローフェーズの組み合わせを全てSPU化
- メッシュ用のナローとミッドフェーズを統合させる
  - 同じSPUプログラムでナローフェーズとミッドフェーズを行う
  - ミッドフェーズから取得した三角がメモリに残る。
- コンタクトポイントには制限なし

PhysX 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

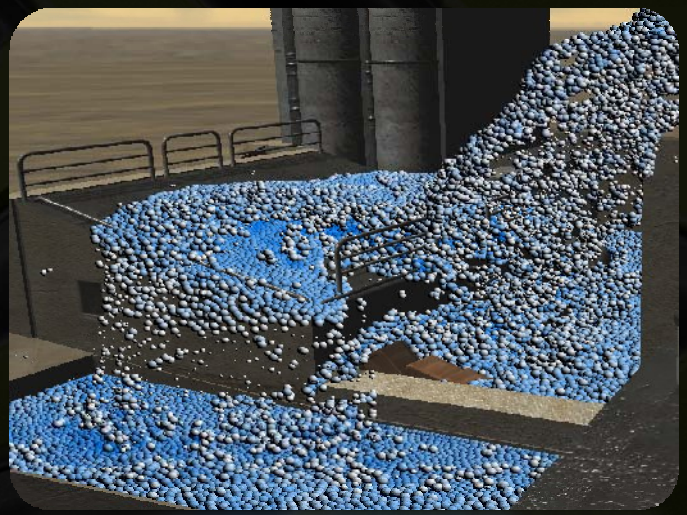
GPUならではの大量物理

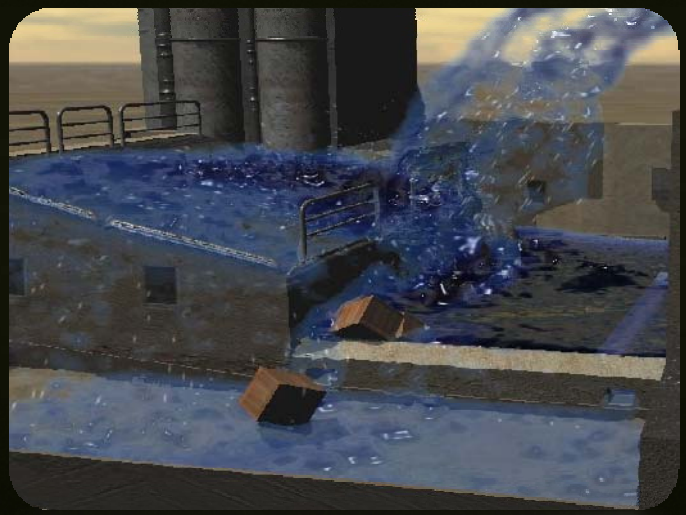
- CUDAを用いて、Geforce 8シリーズ以上に対等
- 推奨の機能
  - パーティクルシステム
  - 柔軟体



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





大量パーティクルシステムのデモ

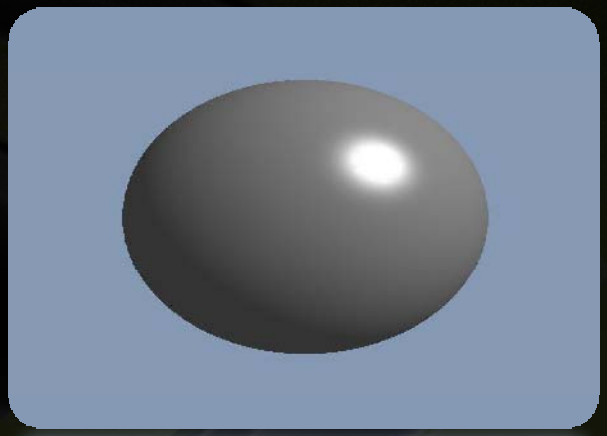
環境効果

PhysX機能 – 柔軟体

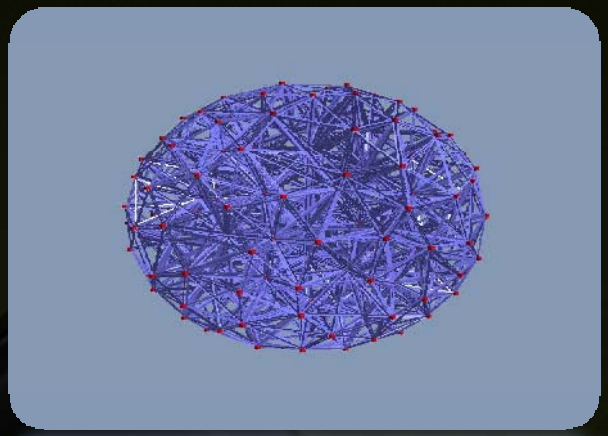
- アニメーションに基づいた二次的な動きの自動生成
- 環境との接触
- 有機的かつリアルな動き

柔軟体の作成ワークフロー

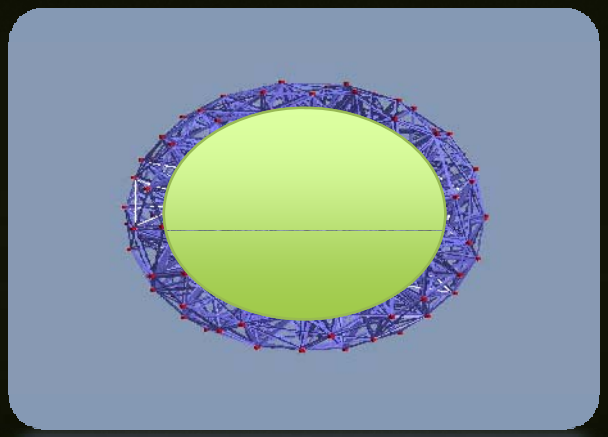
モデルを作成



内側の三角を生成

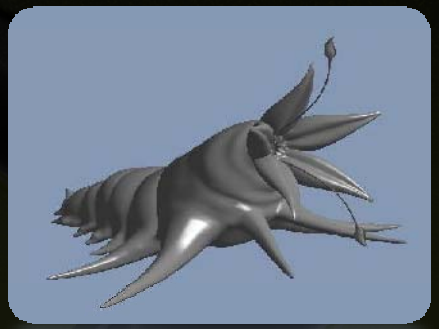


モデルとのマッピング

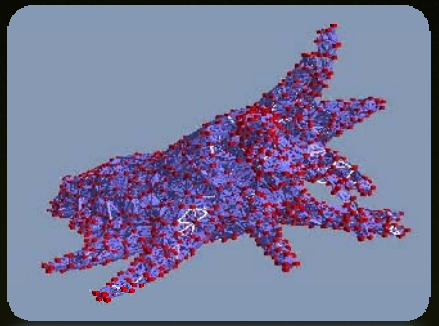


デモ・モンスターの作成

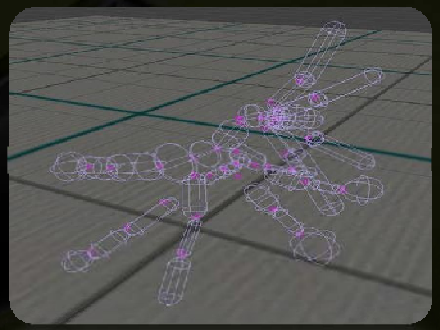
モデルを作成



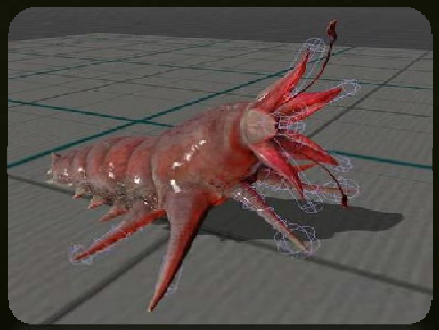
内側の三角生成



モデルとのマッピング



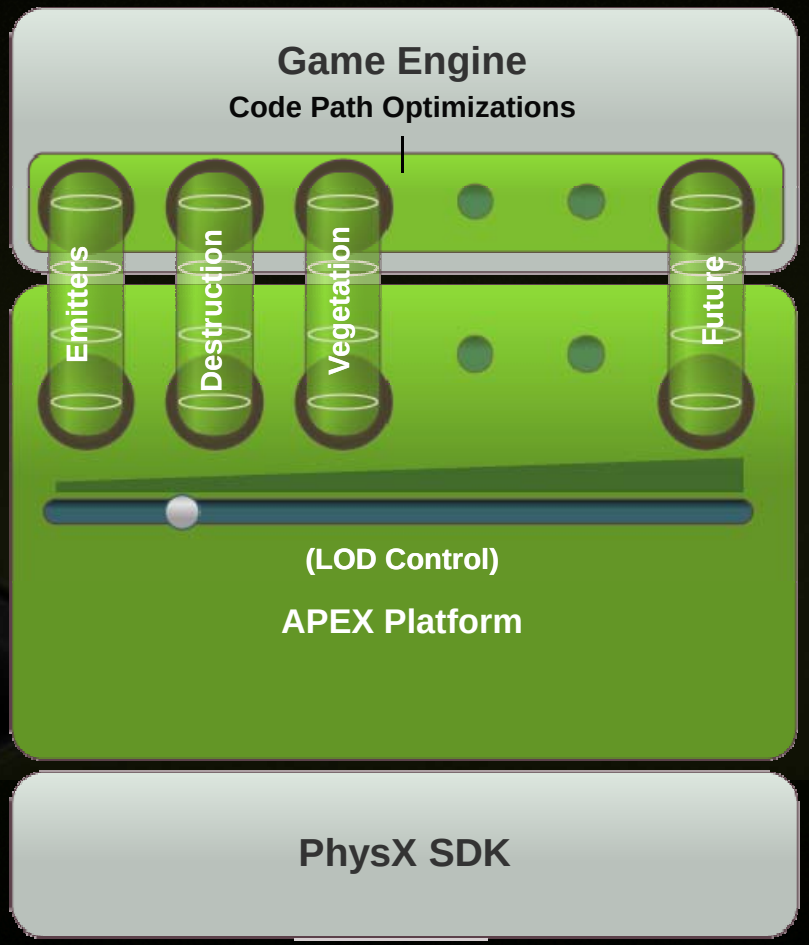
最終結果



スケールと「作る」がキーワードの APEX

apex

- ツールを通して、デザイナーがクロスや破壊対象のオブジェクトなどを作る
- PhyXの上に乗り、物理の規模のスケーリングと描画の高速化を簡単にする
- プラグイン形式のモージュル



APEX Modules: Destruction(破壊モジュール)

- APEX 破壊モジュール(Destruction)
  - Closedメッシュのオブジェクトの破壊を可能にする
  - 破壊の種類やオブジェクトの種類を提供
  - コントロールやスケールのできる破壊
    - 作成時と実行時のパラメーター
  - ベータは既にUE3に組み込み

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ント作成
  - 敗れるクロス
  - いくつかのマテリアルを提供

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

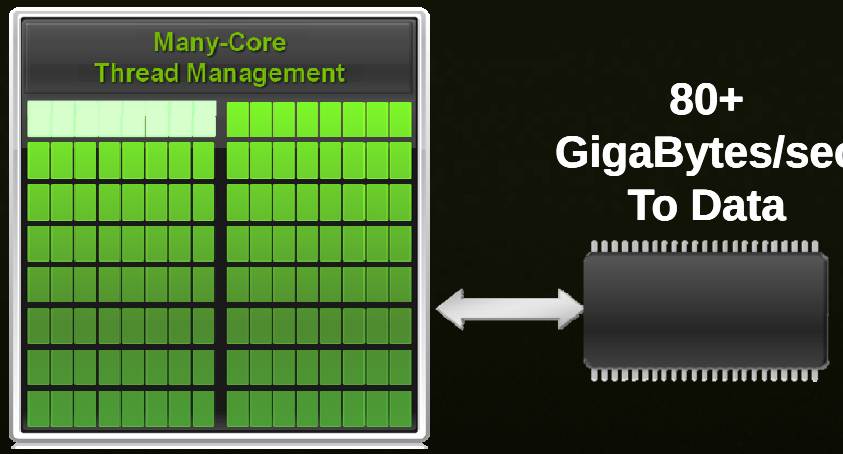
ゲーム開発では、CUDAを使って何ができる？？

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ーマップの生成、大域照明の計算
  - テクスチャーの圧縮や解凍
- 高速なビデオの形式変換や再生
- 等々

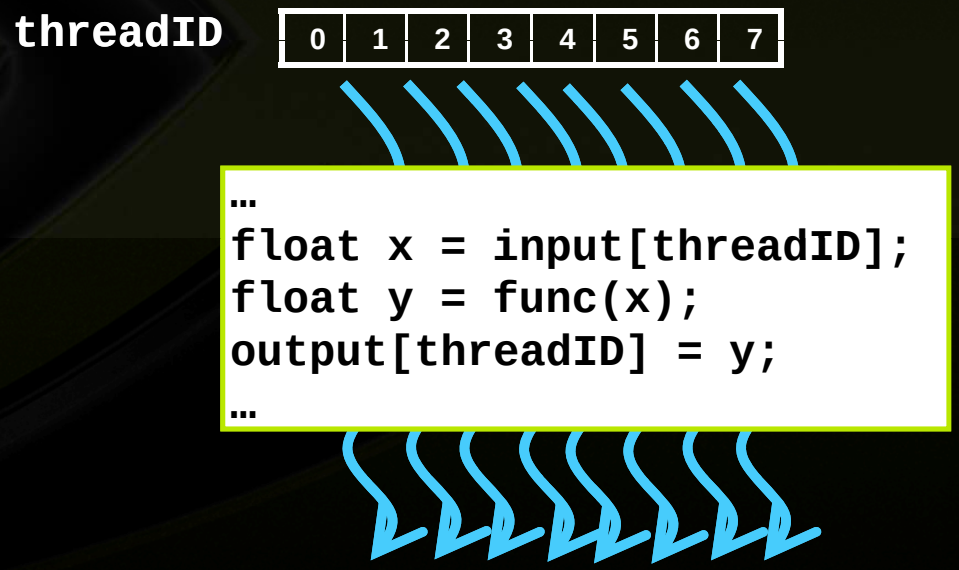
GPU＝高並列性のプロセッサー

- 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



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



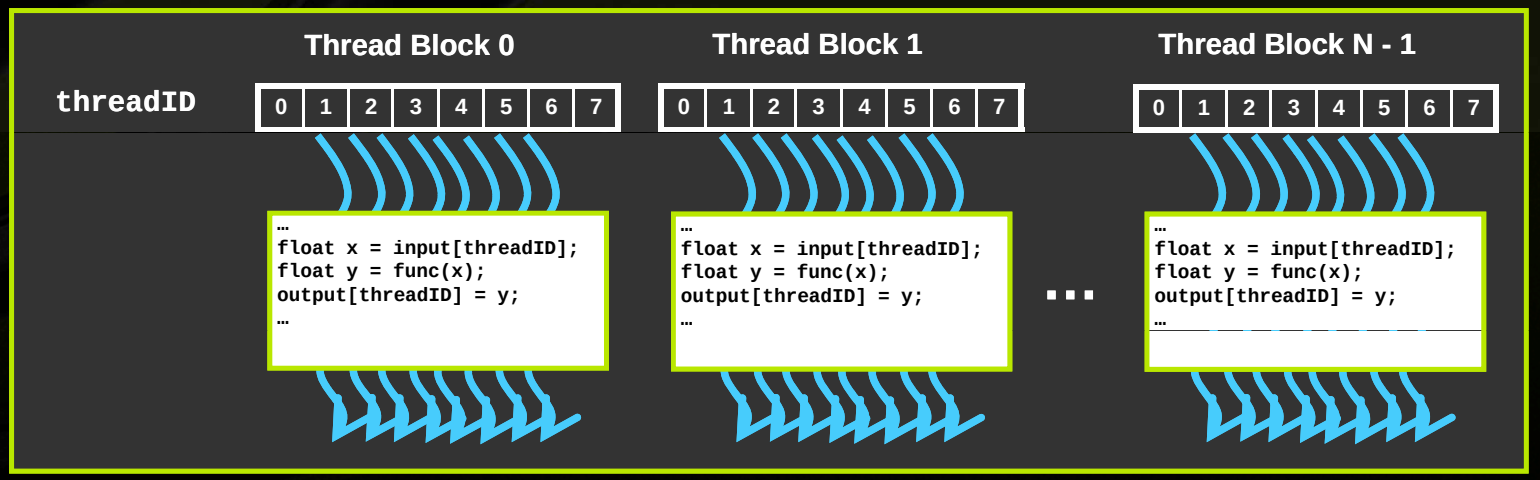
スレッド同士のやり取り

- 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、大幅な帯域幅の節約

- 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

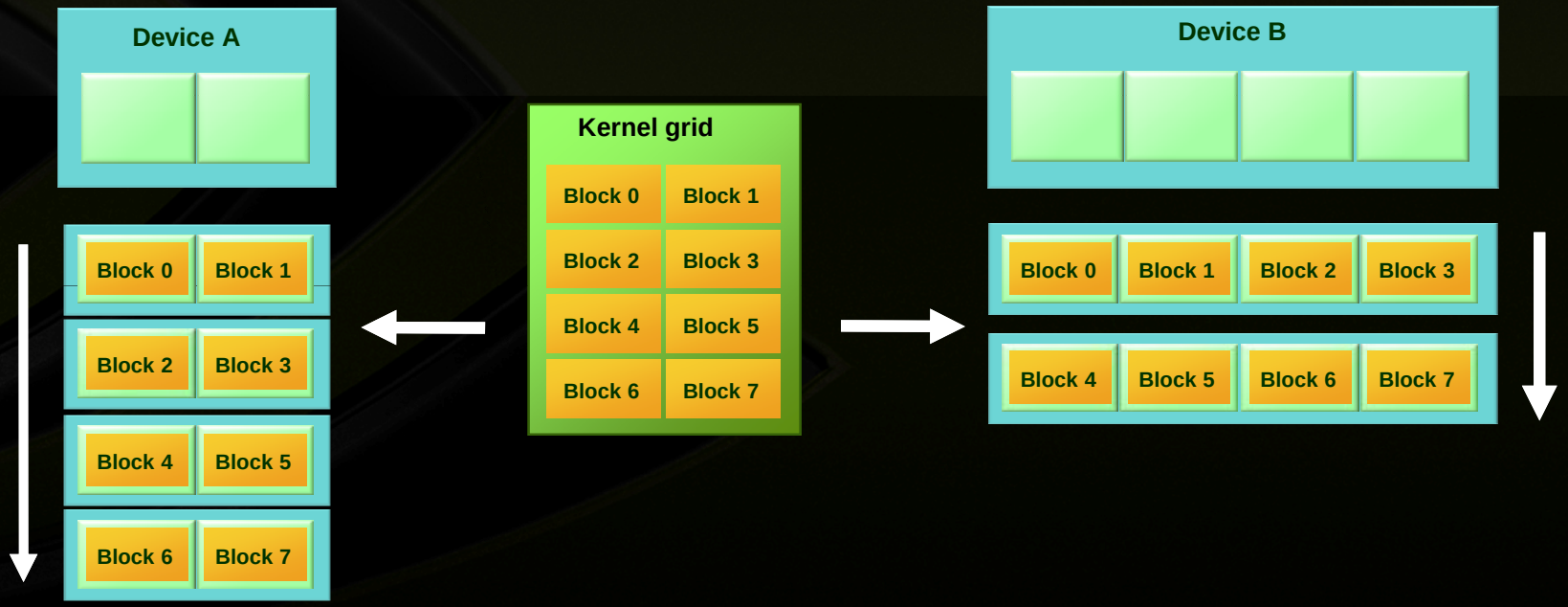
スレッドのブロック: 拡大可能な協力n

- 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やり取りする
  - 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、プロセッサーの数に合わせて、透明的に拡大できる！



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ー別に
  - カーネルの処理性能が任意の数のプロセッサーに拡大できる



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);
```

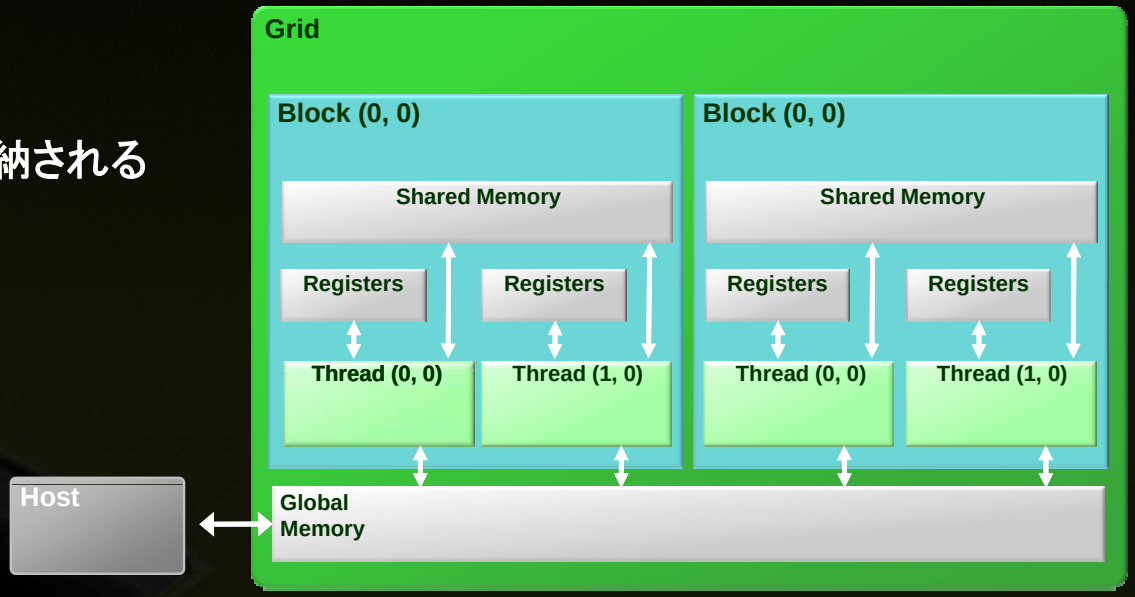
CUDA の C コード

```
__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);
```

カーネルのメモリアクセス

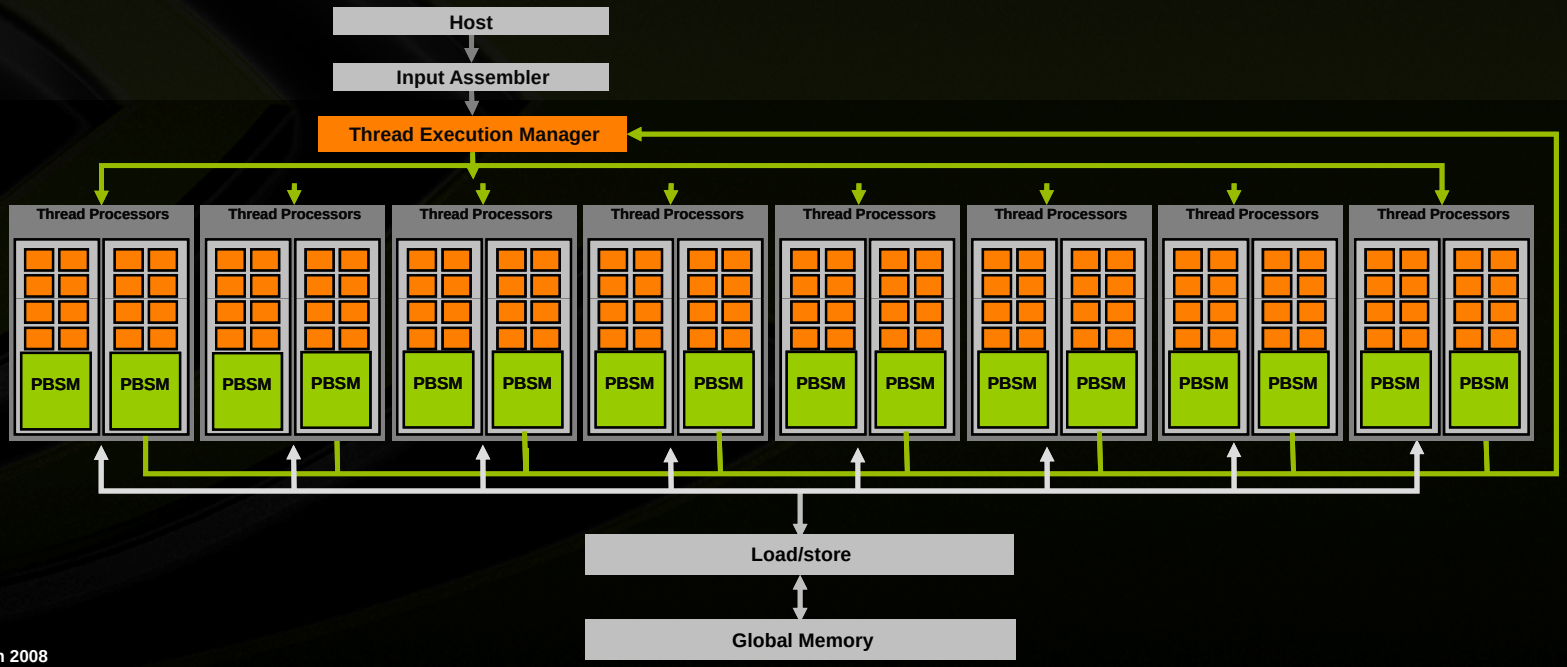
- レジスター
- グロバルのメモリ
  - カーネルの入出力用のデータが格納される
  - チップ外、サイズが大きい
  - キャッシュなし
- 共有メモリ
  - ブロックのスレッドが共有する
  - チップ上、サイズが
  - レジスター並みのの高速性



- ホストがグロ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。

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ー (16 マルチプロセッサー)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



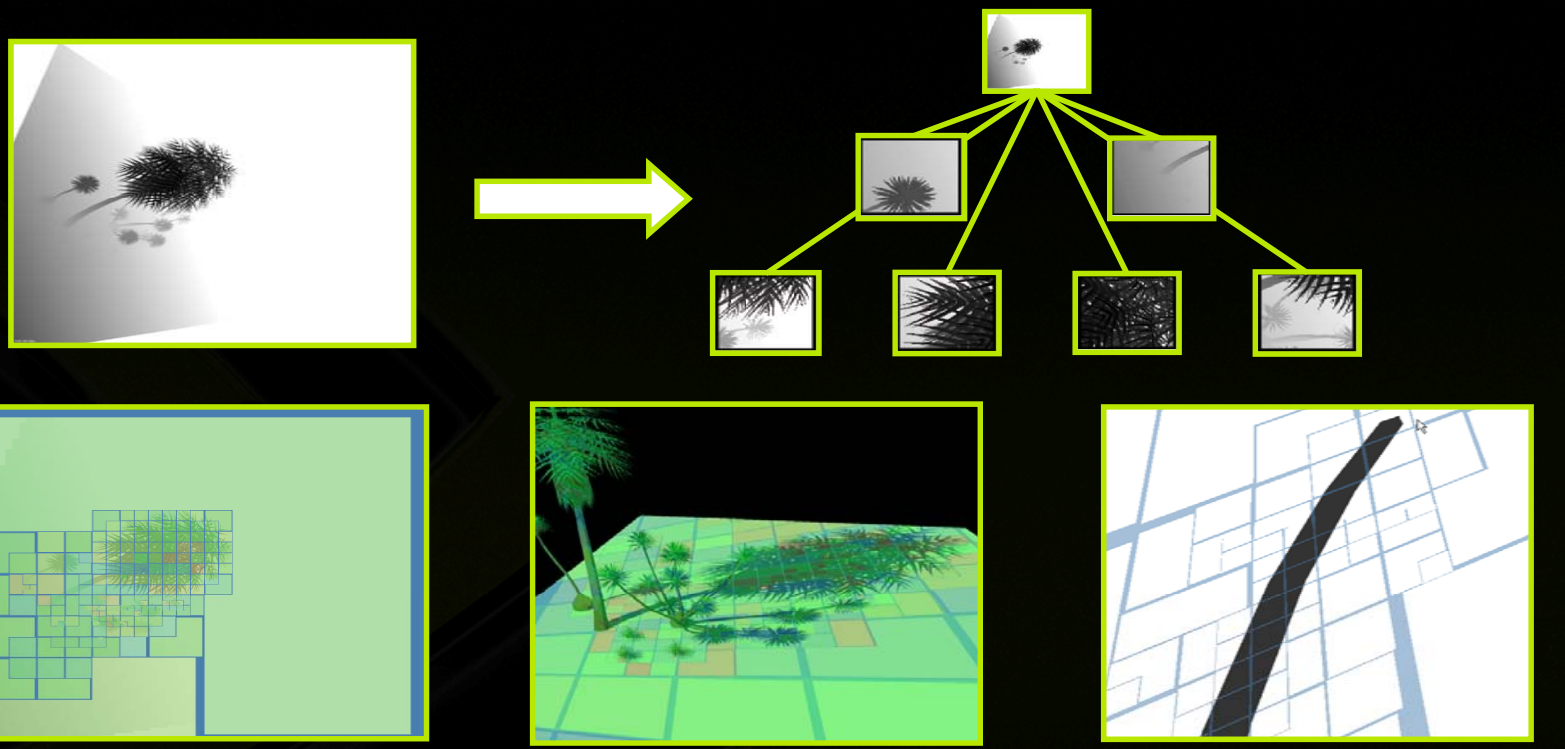
ゲーム用のCUDA

- 描画
- 物理（PhysX）
- 人工知能
- 画像処理

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**
  - マルチテクスチャー、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ーの生成、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。

Quadtreeのシャドーマップ

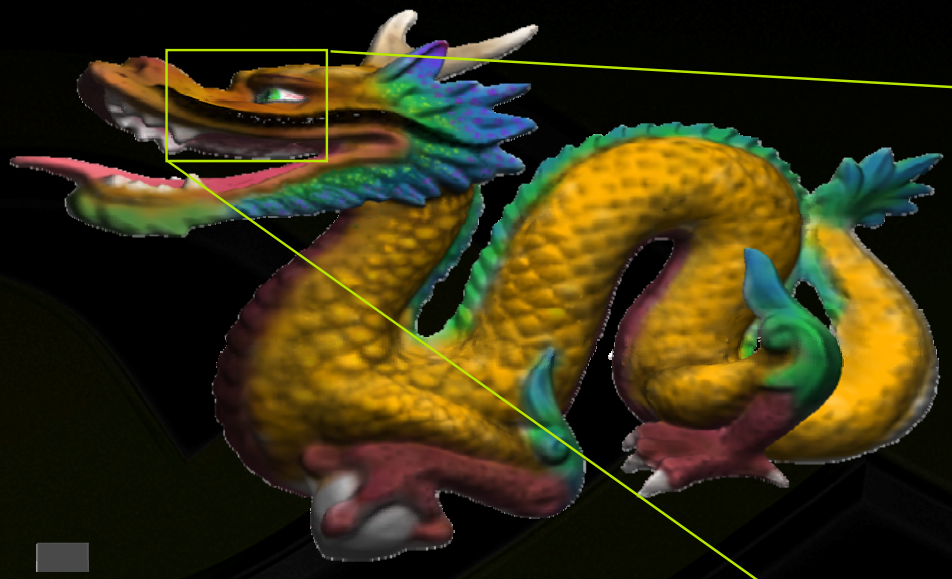


"Adaptive Shadow Maps," Fernando et al., *SIGGRAPH 2001*

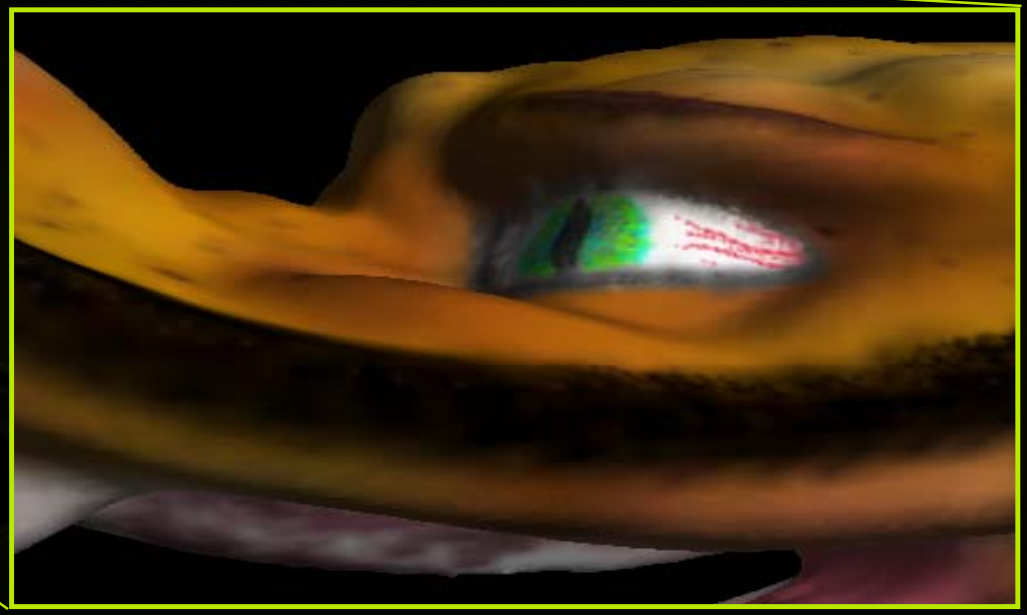
"Resolution Matched Shadow Maps", Aaron E. Lefohn, Shubhabrata Sengupta, John D. Owens. *ACM Transactions on Graphics,* Oct. 2007.

Courtesy of Aaron Lefohn

Octree 3D Paint



Store surface color
in octree texture



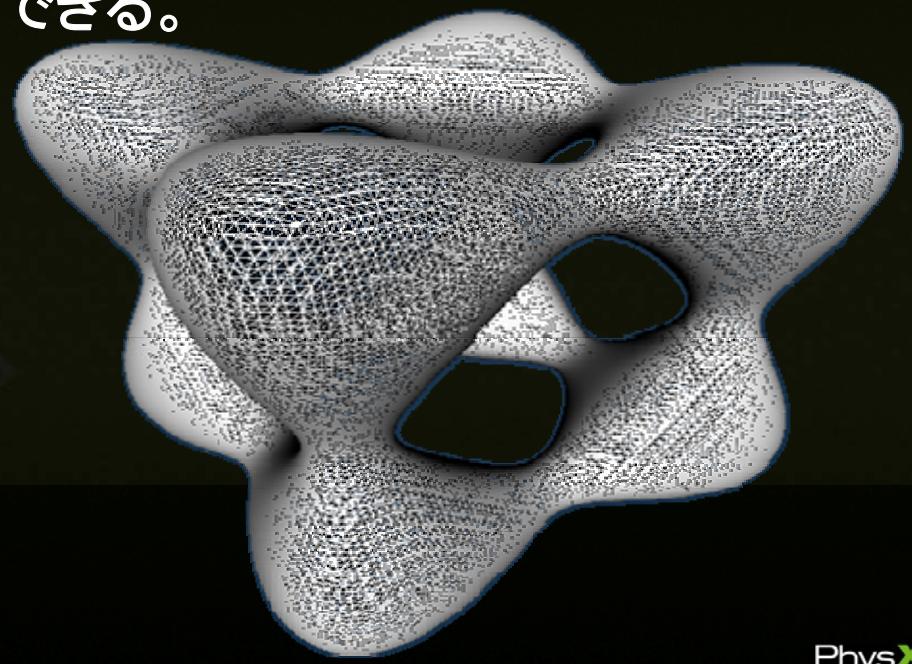
必要性に合わせて、解像度を調整する

2048^3 Effective Texture Resolution

Courtesy of Aaron Lefohn

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ーが生成できる
- 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。

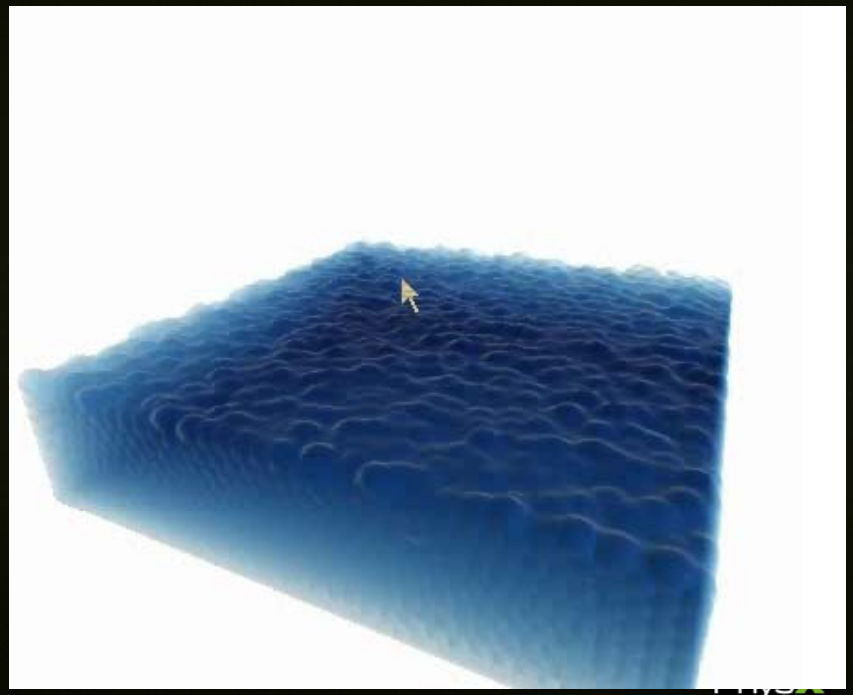


ソートのアルゴリズム

- 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

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ー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、並列例が高まってくる
    - 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, 輪郭検出など
  - ミップマップの生成
  - Subsurface scattering （表面下散乱)

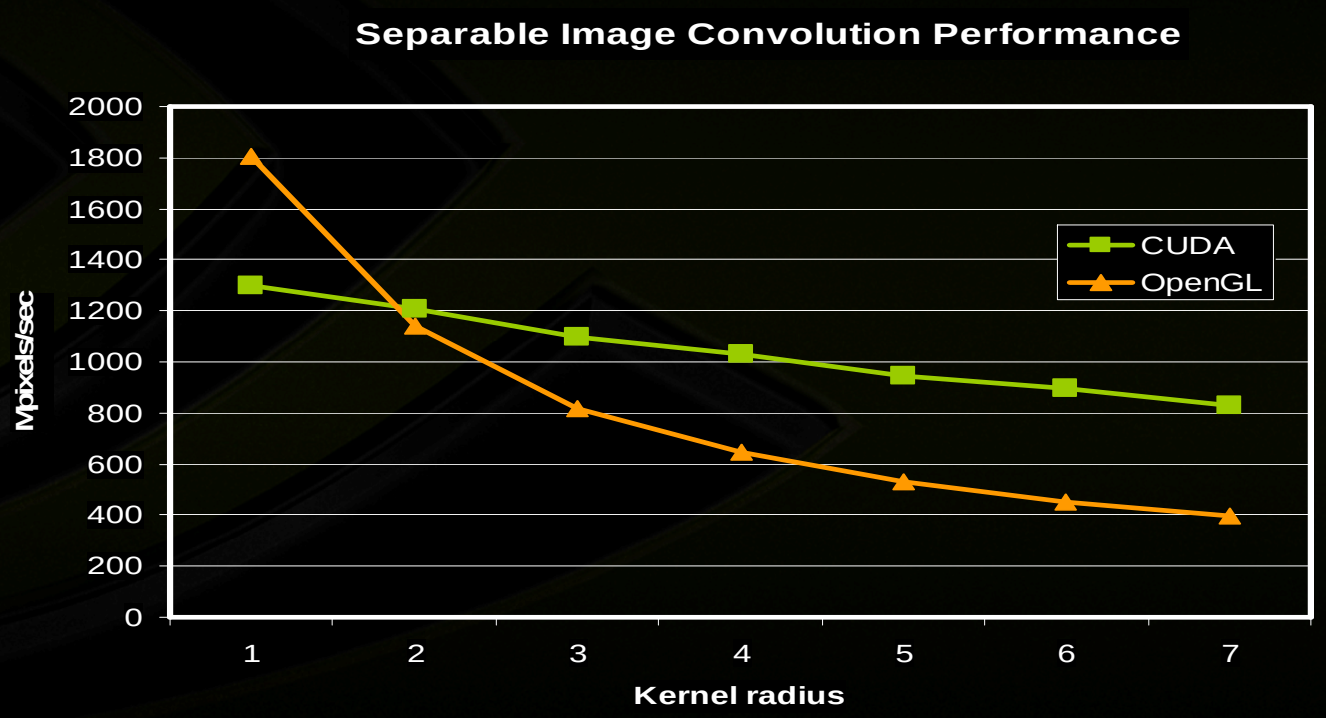
元の画像

ぼかしフィルター

輪郭検出のフィルター

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増加



DXTの圧縮

- オフライン
- ランタイム
- 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 X6800 @ 2.9GHz	AMD Athlon64 Dual Core 4400
Lena 512x512	**42 ms**	**56 ms**	**155 ms**	**563 ms**	**1,251 ms**