社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－CG－126 (14)
2007／2／20

GPUを用いたリアルタイム剛体シミュレーション

原田 隆宏† 田中 正幸††
越塚 誠一†† 河口 洋一郎†

剛体シミュレーションでは衝突計算の計算コストが高く，リアルタイムで計算することは難しい．本研究では剛体形状を球の集合体として近似し，効率的に剛体間の衝突計算を行なった．また本衝突計算手法は並列化可能であるため，剛体計算を全てGPU上で実装し，高速化することが可能である．本手法をGPUで実装した結果，剛体計算を大幅に高速化することができた．数万のチェスの駒を用いたシミュレーションでもリアルタイムで計算することが可能になった．

Real-time Rigid Body Simulation using GPUs

Takahiro Harada,† Masayuki Tanaka ,†† Seiichi Koshizuka††
and Yoichiro Kawaguchi†

Real-time computation of rigid body simulation is a difficult problem as the computational of collision detection in rigid body simulation is expensive. In this study, collisions are computed by approximating a rigid body by a set of spheres and colliding forces of rigid bodies are computed by using them. As the method is able to be parallelized, Graphics Processing Units (GPUs) can be used for computation of rigid body simulation. By using of GPUs, a rigid body simulation is accelerated drastically. The proposed method could compute one time step of a rigid body simulation with several tens of thousands of chess pieces in real-time.

1. 序　　論

我々の身近にある物体は力が加えられても変形量はごくわずかなものが多い．これらの挙動を計算する場合は変形しない物体，つまり剛体として近似することができる．また近年，コンピュータグラフィックスでアニメーションを作成するときに物理シミュレーションを行ない，物体の動きを計算することが行なわれている．このようなシミュレーションベースCGで望まれている計算は，物理的に正確な計算ではなく，もっともらしい計算である．また計算速度の速さや，安定性も重要である．

本研究ではリアルタイムで剛体シミュレーションを行なう手法を提案する．本手法は剛体形状を球の集合体として近似して衝突計算を行なう．全ての形状を同一の大きさの球を用いて近似するため，全剛体を構成する球同士の衝突計算を行なうことで衝突によって剛体に働く力を計算することが可能である．また本手法は並列化を行なうことができるため，全ての計算をGPU上で行ない高速化することが可能である．

† 東京大学大学院情報学環学際情報学府
Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, The University of Tokyo
†† 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo

2. 関 連 研 究

剛体シミュレーションはコンピュータグラフィックスにおいて重要であるため数多く研究が行なわれてきた．Baraffは多面体で表される剛体間の接触の際に働く力を解析的に求める手法を開発し[1]，それを曲面間の接触問題にも拡張した[2]．またBaraffは接触の際に働く摩擦力の計算手法も開発するなど，剛体シミュレーションに関して数多くの研究を行なった[3)4)5)]．Sauer and Schomerは拘束ベースの手法を用いて1剛体のリアルタイムシミュレーションを行なった[19)]．しかしこの拘束ベースの手法は接触点の数の3乗の計算量になるため，多くの剛体を取り扱うリアルタイムシミュレーションに適用するのは困難である[8)]．

多数の剛体を扱った研究も存在する．Guendelmanらは剛体をポリゴンと符号付き距離関数として表し，衝突検出に距離関数を用いることで非凸形状の剛体の衝突計算を行なった[10)]．Kaufmanらは衝突の際に物

体の速度に拘束を課す接触計算手法を開発し，多くの剛体を計算している[13]．しかしリアルタイムで多量の剛体シミュレーションを行なった研究は存在しない．

体積ベースの研究も存在する．Hasegawa らはペナルティ法を用いてリアルタイムシミュレーションを行なった[12]．Dingliana and Sullivan は剛体を階層構造を持った球で表現し 2 次元シミュレーションを行なった[7]．Suzuki らはこの手法の次元を拡張し 3 次元シミュレーションを行なった[20]．

また Graphics Processing Units (GPUs) は内部に多数のプロセッサを持ち並列計算機として用いることができる．GPU の性能向上は著しく，様々な計算を GPU を用いて高速化する研究が行なわれている[9),11),17),18),21),22)]．

3. 剛体計算

3.1 物理量

剛体の計算に必要な物理量は，重心の座標 $\mathbf{x}$，並進速度 $\mathbf{v}$，回転量を表すクオータニオン $\mathbf{q}$，角速度 $\mathbf{w}$，慣性テンソル $\mathbf{I}$ である．剛体の運動は並進運動と回転運動に分けて計算する．

3.2 並進運動

剛体の並進運動ではまず剛体に働く力 $\mathbf{F}$ を計算する．$\mathbf{F}$ は以下のように衝突によって生じる力 $\mathbf{F}_c$ とそれ以外の外力 $\mathbf{F}_e$ に分解することができる．

$$\mathbf{F} = \mathbf{F}_c + \mathbf{F}_e \tag{1}$$

剛体の並進運動量 $\mathbf{P}$ の時間微分は剛体に働く力 $\mathbf{F}$ を用いて以下のように計算される．

$$\frac{d\mathbf{P}}{dt} = \mathbf{F} \tag{2}$$

計算された並進運動量 $\mathbf{P}$ を用いて $\mathbf{P} = M\mathbf{v}$ より重心の速度 $\mathbf{v}$ を計算する．そして重心の速度 $\mathbf{v}$ を用いると剛体の重心位置 $\mathbf{x}$ の時間微分は

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{3}$$

と表される．

3.3 回転運動

剛体に働く力 $\mathbf{F}$ は剛体の回転運動を生み，角運動量 $\mathbf{L}$ を変化させる．角運動量 $\mathbf{L}$ の時間微分は

$$\frac{d\mathbf{L}}{dt} = \mathbf{r} \times \mathbf{F} \tag{4}$$

と表される．ここで $\mathbf{r}$ は剛体の重心から力の作用点までのベクトルである．角運動量を用いて角速度 $\mathbf{w}$ は以下のように計算される．

$$\mathbf{w} = \mathbf{I}(t)^{-1}\mathbf{L} \tag{5}$$

ここで $\mathbf{I}(t)^{-1}$ は時刻 t での慣性テンソルの逆行列である．この時刻 t での慣性テンソルの逆行列 $\mathbf{I}(t)^{-1}$ は初期状態での剛体の慣性テンソル $\mathbf{I}(0)$ の逆行列 $\mathbf{I}(0)^{-1}$ を用いて

$$\mathbf{I}(t)^{-1} = \mathbf{R}(t)\mathbf{I}(0)^{-1}\mathbf{R}(t)^T \tag{6}$$

と計算される．ここで $\mathbf{R}(t)$ と $\mathbf{R}(t)^T$ は時刻 t での回転行列とその転置行列である．

次に式 (5) で計算された角速度 $\mathbf{w}$ を用いてクオータニオン $\mathbf{q}$ を更新する．角速度 $\mathbf{w}$ からクオータニオンの変化量 $d\mathbf{q}$ は以下のように計算される．

$$d\mathbf{q} = [cos\left(\frac{\theta}{2}\right), \mathbf{a}sin\left(\frac{\theta}{2}\right)] \tag{7}$$

ここで回転軸 $\mathbf{a}$ と回転角 θ は

$$\mathbf{a} = \frac{\mathbf{w}}{|\mathbf{w}|} \tag{8}$$

$$\theta = |\mathbf{w}dt| \tag{9}$$

と計算される．

式 (7) で計算されたクオータニオンの変化量と時刻 t でのクオータニオン $\mathbf{q}(t)$ を用いて dt 後のクオータニオン $\mathbf{q}(t+dt)$ は

$$\mathbf{q}(t+dt) = d\mathbf{q} \times \mathbf{q}(t) \tag{10}$$

と計算される．

4. 衝突計算

4.1 概要

本手法では剛体を球の集合と近似し，それらを用いることで衝突計算を行なう．剛体のポリゴンモデルからの球の近似形状は原田，越塚の GPU を用いたボクセル化手法を用いて生成した[11]．用いる球の空間解像度を変化させることで計算精度と計算速度をコントロールすることができる (図 1)．

4.2 衝突検出

剛体の衝突を検出するために，剛体を構成する球の間の距離を計算する[23]．球の間の距離が球の直径よりも小さければ衝突しているとみなす．衝突検出は全ての剛体を構成する球と距離計算を行なわなければならない．この計算コストは球の数の 2 乗に比例し，剛体の数が多くなると球の数も多くなり，計算コストは膨大なものとなる．そこで計算量を減らすためにまず空間をバケットと呼ばれる格子に分割して，全ての球をバケットに格納する．バケットの一辺の長さを球の直径と同じ長さにすると，このデータ構造を用いることで球 i と衝突している可能性のある球は球 i が格納されているバケットと隣接している 3^3 個のバケット内に格納されている球に限定される．このようにバケットを構築して衝突検出を行なうことで計算コストを大幅に下げることができる．

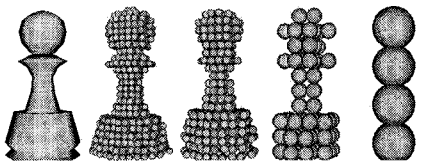

図 1 球を用いた形状表現.
Fig. 1 Shape representation using spheres.

4.3 衝突応答

衝突応答では剛体を構成する球にかかる力をまず計算する．衝突力の計算では個別要素法で用いられる線形バネとダッシュポットを用いた[6]．バネ定数と減衰定数を k, η として，衝突している球にめり込み量に比例した反発力と相対速度に比例した減衰力を働かせる．2 つの球 i と球 j があり，これらの球の直径を d とすると，2 つの球の距離 $|\mathbf{r}_{ij}|$ が d より小さいときに衝突しており，以下のバネによる力 $\mathbf{f}_s$ とダンパによる減衰力 $\mathbf{f}_d$ を働かせる．

$$\mathbf{f}_s = -k(d - |\mathbf{r}_{ij}|)\frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|} \tag{11}$$

$$\mathbf{f}_d = \eta(\mathbf{v}_j - \mathbf{v}_i) \tag{12}$$

ここで $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$ であり $\mathbf{r}_i, \mathbf{r}_j$ はそれぞれ球 i, j の位置ベクトルである．また剪断方向の摩擦力 $\mathbf{f}_t$ は剪断方向の速度 $\mathbf{v}_t$ に比例した力として以下のように計算される．

$$\mathbf{f}_t = k_t \mathbf{v}_t \tag{13}$$

これらの力から球 i に働く力 $\mathbf{f}_i$ は

$$\mathbf{f}_i = \mathbf{f}_s + \mathbf{f}_d + \mathbf{f}_t \tag{14}$$

と計算される．

計算された球 i に働く力 $\mathbf{f}_i$ を用いて衝突によって剛体に働く力 $\mathbf{F}_c$ とトルク $\mathbf{T}_c$ は以下のように計算される．

$$\mathbf{F}_c = \sum_{i \in RigidBody} \mathbf{f}_i \tag{15}$$

$$\mathbf{T}_c = \sum_{i \in RigidBody} \mathbf{r}'_i \times \mathbf{f}_i \tag{16}$$

と計算される．ここで $\mathbf{r}'_i$ は剛体の重心に対する球の相対位置ベクトルである．

5. GPU を用いた高速化

本手法の計算は全て GPU 上で実装し，高速化することができる．本章では GPU での計算アルゴリズムについて説明していく．

5.1 データ構造

GPU で計算するときに物理量はビデオメモリにテクスチャとして保持される．剛体計算を行なうために剛体重心座標，剛体クオータニオン，並進運動量，角運動量テクスチャをそれぞれ 2 枚ずつ必要とする．衝突の計算を行なうためには球の中心座標，球の速度，バケット，球に働く力，球に働くトルクにテクスチャをそれぞれ 1 枚ずつ用意する．

3 次元計算の場合，バケットとして 3 次元空間格子を用意しなければならない．現在の GPU では 3 次元テクスチャへの書き込みを行なうことができないため，$L \times L \times L$ の格子数の 3 次元バケットを複数枚の 2 次元格子を $D \times D$ 枚敷き詰めた 1 枚のテクスチャを用いて表現する．ここで 3 次元バケットのインデックス (i, j, k) から 2 次元テクスチャ上でのインデックス (s, t) は以下のように計算することができる．

$$s = i + L(k - D[k/D]) \tag{17}$$

$$t = j + Lf[k/D] \tag{18}$$

ここで i, j, k と s, t はそれぞれ $[0, L-1]$ と $[0, D \times L-1]$ の範囲の値を取る．

5.2 アルゴリズム

1 タイムステップでの計算は以下の 8 段階で計算される．

(1) 球の物理量の計算
(2) バケット構築
(3) 衝突検出
(4) 衝突応答 (球に働く力とトルクの計算)
(5) 剛体並進運動量の計算
(6) 剛体角運動量の計算
(7) 剛体重心座標の計算
(8) 剛体クオータニオンの計算

そして計算された剛体の重心座標とクオータニオンを用いてレンダリングを行なう．

5.3 球の物理量の計算

まず衝突計算のために各剛体を構成する球の中心座標と速度を計算する．剛体 j の重心座標とクオータニオン，速度，角速度をそれぞれ $\mathbf{X}_j, \mathbf{Q}_j, \mathbf{V}_j, \mathbf{W}_j$ とする．回転していない状態での剛体の重心に対する球 i の中心の相対位置ベクトルを $\mathbf{r}_i$ とするとクオータニオン $\mathbf{Q}_j$ で回転している状態での相対位置ベクトル $\mathbf{r}'_i$ は

$$\mathbf{r}'_i = \mathbf{Q}_j \mathbf{r}_i \mathbf{Q}_j^T \tag{19}$$

と計算される．

剛体 j を構成する球 i の中心座標と速度をそれぞれ $\mathbf{x}_i, \mathbf{v}_i$ とすると，これらは

$$\mathbf{x}_i = \mathbf{X}_j + \mathbf{r}'_i \tag{20}$$

$$\mathbf{v}_i = \mathbf{V}_j + \mathbf{W}_j \times \mathbf{r}'_i \tag{21}$$

と計算することができる．

5.4 バケット構築

バケットの構築は球の番号を3次元格子内に格納する処理であり，データを分散させる処理である．この処理はフラグメントシェーダでは行なうことができない．バーテックスシェーダでは入力とされる座標に頂点をレンダリングすることができる．バーテックステクスチャフェッチ (VTF) はバーテックスシェーダでテクスチャメモリにアクセスする機能であり，この機能を用いることでバーテックスシェーダで球の座標を参照することができる[16]．そしてその球の座標に対応するバケット座標に頂点を配置することでGPUでデータを分散させる処理を行なうことができる．

まず球の数 n と同数の頂点 $p_0, p_1 \cdots, p_n$ を用意し，各頂点に球1つを対応付ける．これらの頂点に対応する球の番号は $i_0, i_1 \cdots, i_n$ とし，球の番号は $i_0 < i_1 < i_2 < \cdots < i_n$ の順に並んでいるとする．バーテックスシェーダではVTFを用いて前のタイムステップでの球の座標を参照する．そしてその座標からバケットテクスチャ上での座標を計算し，その座標に球の番号を出力することで，バケットテクスチャに球の番号を書き込むことができる．

しかしこの処理ではバケットを正しく構築することができない．1つのバケットの内部に複数個の球が存在することがあるからである．そのため，バケットを正しく構築するためには球1個1個を順にバケットに格納し，そのバケットに格納された球の数も数えなければならない．そのため前述のGPUを用いた並列処理では各バケットに格納された球の数を数えることができないため，この処理後には各バケットには1個の球の番号しか書き込まれない．例えば1つのバケットに4個の球が存在するときには4頂点が同一ピクセルに書き込まれるため，最終的にそのピクセルにはそれらのうちの1個の番号が書き込まれる．

ここでバケット b に4個の球が入る場合を考える．これらの球の番号はそれぞれ $i_{b0}, i_{b1}, i_{b2}, i_{b3}$ として，RGBAチャネルにそれぞれの値を書き込む．この球の番号は $i_{b0} < i_{b1} < i_{b2} < i_{b3}$ の順に並んでいるとする．バケット構築を4回のレンダリングに分けることで1個のバケット内に複数個の球が存在する場合もGPUを用いて行なうことができるようになる．バーテックスシェーダでは頂点に対応する球の座標を用いてその座標のバケット内での座標を計算し，球の番号をフラグメントシェーダに渡す．そしてフラグメントシェーダでは球の番号を色と深度として書き出す．

まず1パス目ではRチャンネルに球番号 i_{b0} を書き込む．i_{b0} は4個の球のうち最も小さい番号を持つ．

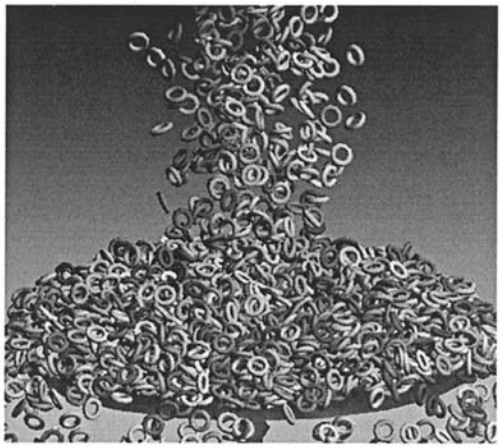

図 3 A real-time simulation result in which 10,923 toruses are used.

すなわち最小深度を持つピクセルとしてレンダリングされる．深度バッファを最大値で初期化し，深度テストを用いて小さい値を持つものを合格させるようにすることで i_{b0} を書き込むことができるようになる．2パスではRチャンネルに書き入れた値を上書きしないようにカラーマスクを用いてRチャンネルへのレンダリングを禁止する．そして深度バッファは1パス目で用いたものをそのまま用い，深度テストは大きな値を持つものを合格させるように設定する．しかしこれではピクセルに最大の番号の i_{b3} が書き入れられるため，ステンシルテストを用いる．ステンシルバッファを0で初期化し，ステンシル処理は値を増加させるように設定する．そしてステンシルテストでは値が1以上なら失敗するように設定することでピクセルに1回レンダリングした後は書き込まないようにすることができる．このようにして i_{b1} を書き込む．3,4パスでも同様な処理を行ない，それぞれ i_{b2}, i_{b3} をBAチャンネルに書き込む．

5.5 衝突検出

球 i と衝突している可能性のある近傍に存在する球の探索に球の座標テクスチャと生成したバケットテクスチャを入力として，フラグメントシェーダで処理を行なう．バケットを参照することである球 i が格納されているバケットを囲む 3^3 個のバケット内に格納されている球の番号を得ることができる．これらの球と球 i との距離計算を行ない衝突検出を行なう．

5.6 球に働く力とトルクの計算

衝突している2つの球の座標 $\mathbf{r}_i, \mathbf{r}_j$ とそれらの速度 $\mathbf{v}_i, \mathbf{v}_j$ と剛体の重心に対する相対位置ベクトル $\mathbf{r}'_i, \mathbf{r}'_j$ を用いて式(12)から，球に働く力 $\mathbf{f}_i$ とその力によって

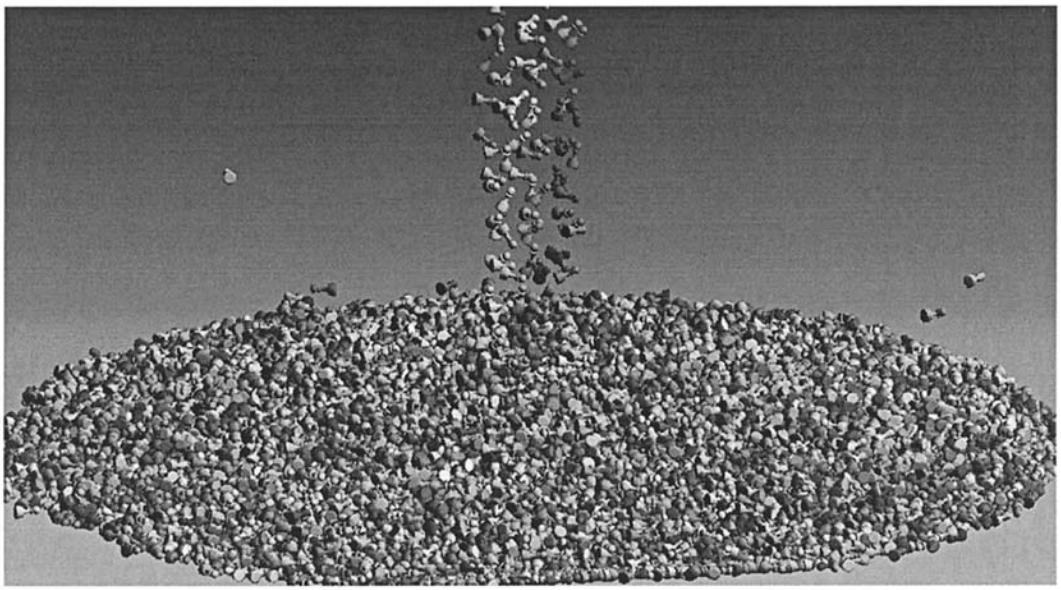

図 2　A real-time simulation result in which 16,385 chess pieces are used.

剛体に生じるトルク $\mathbf{r}'_i \times \mathbf{f}_i$ を計算することができる．

5.7　剛体並進運動量と角運動量の計算

球に働く力とその力によって剛体に生じるトルクを計算したので，次に式 (15) と式 (16) を用いてそれらを足し合わせ剛体に働く力を計算する．そして式 (2) と式 (4) を用いて剛体の並進運動量と角運動量を更新する．

5.8　剛体重心座標とクオータニオンの計算

剛体の重心座標は式 (3) を用いて計算される．ここでは剛体の重心座標と重心速度テクスチャを入力としてもう 1 枚の剛体の重心座標テクスチャに出力する．

剛体のクオータニオンの更新は式 (10) を用いて計算される．ここでは剛体のクオータニオンと角速度テクスチャを入力としてもう 1 枚の剛体のクオータニオンテクスチャに更新された値を書き出す．

5.9　レンダリング

このように計算された剛体の重心座標とクオータニオンを用いてレンダリングを行なう．頂点 i の回転していない時の剛体の中心からの相対座標ベクトル $\mathbf{r}_i$ をまずクオータニオン $\mathbf{Q}$ を用いて現在の剛体の回転量だけ回転させ，剛体の重心座標 $\mathbf{X}$ 移動させると頂点 i の現在の座標 $\mathbf{r}'_i$ を計算することができる．

$$\mathbf{r}'_i = \mathbf{X} + \mathbf{Q}\mathbf{r}_i\mathbf{Q}^T \tag{22}$$

そしてモデルビュープロジェクション行列 $\mathbf{M}$ を用いて，以下のように現在レンダリングされている座標系での座標 $\mathbf{r}''_i$ に変換してレンダリングする．

$$\mathbf{r}''_i = \mathbf{M}\mathbf{r}'_i \tag{23}$$

6.　結果と考察

本手法を Core 2 CPU X6800 2.93GHz, 2.0GB のメモリ，GeForce7900GTX を搭載した PC 上で実装した．プログラムは C++, OpenGL, C for Graphics を用いて開発した．

図 2 にチェスの駒 16,385 個落とした結果を示す．この計算時間は表 1 に示すように 1 タイムステップにかかる計算時間は 12.8 ミリ秒であった．剛体数を変え計算時間とフレームレートを測定した．1 タイムステップ計算した後にシーンをレンダリングしたフレームレートも，表 1 に示すように，チェスの駒 4,096 個落とした場合でも 53FPS であった．また図 3 に 10,923 個のトーラスを落下させた計算結果を示す．トーラスも落下させる個数を変化させて計算時間を測定した．表 2 に示すように 10,922 個のトーラスを用いた場合でも 1 タイムステップにかかる計算時間は 16.6 ミリ秒であった．そしてレンダリングも行なった場合ではフレームレートは 23FPS であった．

7.　結　　論

本研究では剛体を球の集合体として近似し，衝突計算を行なった．そしてこの手法を完全に GPU 上で実装することによって，多数の剛体の衝突計算をリアルタイムで計算することが可能になった．

表 1 Simulation times and frame rates of simulation using chess pieces (in miliseconds).

Number of chess pieces	Simulation time	FPS
1,024	4.23	168
4,096	8.62	53
16,384	12.8	21.2

表 2 Simulation times and frame rates of simulation using toruses (in miliseconds).

Number of toruses	Simulation time	FPS
682	4.08	200
2,730	8.00	79
10,922	16.6	23

本研究で用いた球の衝突計算を GPU 上で行なうアルゴリズムは球を用いた他の計算にも応用することが可能である．このアルゴリズムを用いることで個別要素法や Smoothed Particle Hydrodynamics[15], Moving Particle Semi-Implicit Method[14] などの粒子法も GPU 上で高速化することが可能であると考えられ，今後これらの高速化も研究していく．

参 考 文 献

1) D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–232, 1989.
2) D.Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics*, 24(4):19–28, 1990.
3) D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4):31–40, 1991.
4) D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics Proc., Annual Conference Series*, pages 23–34, 1994.
5) D.Baraff. Interactive simulation of solid rigid bodies. *IEEE Computer Graphics and Applications*, 15:63–75, 1995.
6) P.A. Cundall and O.D.L. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29:47–65, 1979.
7) J. Dingliana and C. O'Sullivan. Graceful degradation of collision handling in plysically based animation. *Computer Graphics Forum*, 19(3):239–247, 2000.
8) K. Erleben, J. Sporring, K. Henriksen, and H. Dohlmann. *Physics-Based Animation*. Charles River Media, Inc., 2005.
9) R. Fernando. *GPU gems*. Addison-Wesley Professional, 2004.
10) E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics*, 22:871–878, 2003.
11) T.Harada and S.Koshizuka. Fast solid voxelization using graphics hardware. *Transactions of JSCES*, page No.20060023, 2006.
12) S.Hasegawa, N.Fujii, Y.Koike, and M.Sato. Real-time rigid body simulation based on volumetric penalty method. In *Proc. of the 11th Symposium on Haptic Interfaces for VirtualEnvironment and Teleoperator Systems*, pages 326–332, 2003.
13) D.M. Kaufman, T.Edmunds, and D.K. Pai. Fast frictional dynamics for rigid bodies. *ACM Transactions on Graphics*, 24(3):946–956, 2005.
14) S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nucl.Sci.Eng.*, 123:421–434, 1996.
15) J.J. Monaghan. Smoothed particle hydrodynamics. *Annu.Rev.Astrophys.*, 30:543–574, 1992.
16) M. Olano, R.J. Rost, J.M. Kessenich, and B. Lichtenbelt. *OpenGL Shading Language*. Addison-Wesley Pearson Education, 2004.
17) JohnD. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, AaronE. Lefohn, and TimothyJ. Purcell. A survey of general-purpose computation on graphics hardware. *Eurographics 2005, State of the Art Reports*, pages 21–51, 2005.
18) M. Pharr and R. Fernando. *GPU Gems 2*. Addison-Wesley Professional, 2005.
19) J.Sauer and E.Schomer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proc. of the ACM Symposium on Virtual Reality Software and Technology*, pages 153–162, 1998.
20) K.Suzuki, J.Kubota, and H.Ohtsubo. Rigid body simulation using volume based collision detection. *Structural Engineering / Earthquake Engineering*, 22:185–192, 2003.
21) 原田隆宏 and 越塚誠一. 布と高解像度のモデルとのリアルタイム衝突計算. **情報処理学会論文誌, 掲載予定**, 48(4), 2007.
22) 原田隆宏 and 越塚誠一. 離散境界を用いた距離関数の構築手法. **情報処理学会論文誌, 掲載予定**, 48(4), 2007.
23) 田中正幸, 酒井幹夫, and 越塚誠一. 粒子法を用いた剛体計算手法の開発とコンピュータ・グラフィックスへの適用. In **日本機械学会第 *19* 回計算力学講演会講演論文集**, pages 701–702, 2006.