柔らかい地面上の物体による痕跡の高速表示法

尾上　耕一　　西田　友是
東京大学

<あらまし> コンピュータグラフィックスによるアニメーションに関して、近年、特にキャラクターの動作に伴う二次的な現象も研究対象となっている。例えば、キャラクターが柔らかい地面（砂、泥、雪など変形しやすいもの）に残す足跡や自動車や自転車のタイヤ跡、ボールが落とされたり地面の上を引きずられたりした跡も、二次的な現象に含まれる。本論文では物体の衝突により地面に残る痕跡を計算する方法を提案する。提案法では、(1)地面と物体との衝突判定、(2) 衝突部分の地面の物質の圧縮と周囲への移動、(3)急斜面での崩落、をそれぞれ計算することによって物体の接触による地面の変形を計算する。衝突判定にグラフィックスハードウェアを利用し、崩落計算を局所的に行うことによってリアルタイム表示を実現した。
キーワード: ハイトフィールド、Zバッファ、アニメーション、地形
<Summary> In recent years there has been a particular interest in secondary phenomena caused by the motion of objects in CG animation. Examples of such secondary phenomena include footprints that a character leaves in its wake, tire marks caused by cars and bicycles, and marks made by objects such as balls falling onto or being dragged along the ground. In this paper, we propose a deformation algorithm for the ground surface when it is in contact with objects. The deformation algorithm is divided into three steps: (1) the detection of the collision between the objects and the ground surface, (2) the compression and displacement of the ground material, (3) the erosion at a steep slope. The deformation of the ground surface is calculated in real time by using graphics hardware for collision detection, and by calculating the erosion step in limited regions on the surface of the ground.
Key words: Height field, Z buffer, Animation, Terrain

1. まえがき

従来からコンピュータグラフィックスを用いたアニメーションに関する研究が盛んに行われてきたが、近年、特にキャラクターの動作によって生じる二次的な現象も研究対象となっている[1]。例えば、物体が地面に残す痕跡[2,3,4]、物体による水面の変化[5]、キャラクターの動きに伴う衣服の変化[6,7]などが研究されている。
このような二次的な現象の中でも、本研究の目的は物体の衝突による地面の変形である。例えば、地面に残されるキャラクターの足跡、自動車や自転車のタイヤ跡、ボールが落とされたり地面の上を引きずられたりした跡などの計算を目的としている。これらは特に砂・泥・雪などに残されやすいので、本研究では変形しやすい柔らかい地面を扱う。

本稿では地面の変形についての定性的な性質に着目したアルゴリズムを提案している。
地面を構成する粒子の重力や摩擦などの物理法則に厳密には基づいていないが、ユーザーが直感的にパラメータを設定可能となるように考慮している。

提案法では地面をハイトフィールドで表す。ハイトフィールドとは一様な正方格子であり、各セルに高さを持たせたものである。そして、提案法では各セルの高さの変化のみを計算する。また提案法は地面の変形を高速に計算することに重点を置いている。インタラクティブな速度で計算することにより、バーチャルリアリティのアプリケーションや、モデラー、ゲーム等に応用することができる。

2. 関連研究

本節では地面の変形に関連した研究を紹介する。

物体の衝突による地面の変形を扱った研究はすでにいくつか存在する。本研究と特に関連が深いものとして Li

"An Efficient Method for Displaying Marks on Soft Grounds Created by Objects",
Koichi Onoue and Tomoyuki Nishita, The University of Tokyo.

ら[2]、Chanclou ら[3]、および Sumner ら[4]の研究が挙げられる。

まず Li らはハイトフィールドを用いた、物理則に基づく土の崩落モデルを提案した[2]。彼らは土の崩落とブルドーザーによる土の掘削、積載、運搬をリアルタイムに計算した。しかし、彼らの論文では物体としてはブルドーザーについてモデルが紹介されているだけである。他の物体を扱うためにはそれぞれモデル化する必要があり一般性に欠ける。また地面の圧縮は考慮されておらず、雪などは扱えない。さらにブルドーザーのタイヤ跡のような、物体と地面の接触の跡は計算できない。

次に、パーティクルを用いた地面のモデルが Chanclou らによって提案された[3]。彼らの提案したモデルでは地面に残る物体の痕跡を生成できる。しかし、彼等の方法は計算コストが高く、ユーザーによるパラメータ設定が困難である。

Sumner らは単純なアルゴリズムを用いて様々な種類の地面の上にできる物体の接触の痕跡を計算した[4]。彼らの目的は本研究と同じであり、彼等の方法の基本的な考え方は本研究でも採用した。しかしながら、インタラクティブな計算速度は実現できていない。また、地面の上で物体を引きずった場合にも問題がある。これらの問題点については本研究で改良したので後で詳しく述べる。

さまざまな種類の地面の中でも、特に雪を扱った方法がいくつか発表されている。Nishita らは雪をメタボールを使ってモデル化し雪が様々な物体に積もるのを表現した[8]。彼らは雪の中で起こる光の多重散乱を考慮したレンダリング法も提案した。Fearing は物体と地面への降雪と積雪のモデル、および雪の安定性のモデルを提案した[9]。彼らは物体の下側の空間に積もる雪や風の効果までシミュレートした。これら二つの研究によって積雪は表現できるが、雪に衝突した物体の痕跡は扱えない。

3. 地面変形アルゴリズム

本節では地面変形アルゴリズムを説明する。まず地面のモデルについて述べた後で、変形アルゴリズムの概要を述べる。そして、アルゴリズムの各ステップについて詳しく説明する。

3.1 地面のモデル

提案法では地面をハイトフィールド、すなわち高さをもつセルの二次元配列で表す。各セルの高さは単位面積当たりの地面の物質の量に等しい。ハイトフィールドの初期状態は様々な方法で決定することができる。本稿では初期状態の高さを乱数によって決定したが、実際の地形データや他のモデラーの出力を利用することも可能である。ハイトフィールドという単純なモデルで地面を表すことによって、地面の変形が高速に計算できる。また、例えば、地面の一部が砂で一部が泥であるような場合も、ハイトフィールドの領域ごとに異なったパラメータ（後述するα、θ）を持たせることによって表現できる。

提案法では地面が物体に被さらない場面を想定している。これは、ハイトフィールドが一価関数であるという制限から、提案法は物体の上側に地面が被さるような場合を表現できないからである。

3.2 地面の変形

提案法は地面の変形を対象としているため、物体は地面との衝突によって変化しないものとする。本稿ではユーザーが物体の動きを制御しているが、力学シミュレーションやモーションキャプチャデータによって物体の動きを決めることも可能である。また、物体としてはポリゴンで表されるものを扱う。

地面は物体の衝突によって圧縮され、衝突領域の境界部分に移動され、そして周辺に崩落が起こることによって変形される。１タイムステップの地面変形の計算は次の順で行う。

衝突判定 Z バッファ法を用いて物体と地面の衝突判定を行う。

地面の圧縮・移動 物体と接触しているセルは圧縮、あるいは衝突領域の境界部分に押し出される。

崩落 衝突領域の境界部分からその周囲に向かって順に、急斜面を検出し、高いセルから低いセルへ地面を移動させることによって傾斜を緩やかにする。

次節以降で各ステップについて詳しく説明する。

3.3 物体と地面との衝突判定

ハイトフィールドが一価関数であるという性質を利用して、地面と物体の衝突判定を Z バッファを用いて行う。Z バッファ法を用いることによりグラフィックスハードウェアを利用して高速に衝突判定を行うことができる。物体同士の衝突判定を Z バッファ法を用いて行うアルゴリズムはいくつか提案されているが[10,11,12]、ここではハイトフィールドと物体の衝突に最適化したアルゴリズムを提案する。

まず本研究で用いた物体と地面との衝突判定法の基本的な考え方を述べる。地面を表面のみの曲面とみなして、地面と物体を地面の下から上を見上げるようにして隠面消去して描画する。このとき、物体が可視となる地点では物体と地面が衝突していることがわかる。

以上のことを OpenGL を用いてグラフィックスハードウェアを利用したZバッファ法により以下のように行う。

1. 物体のバウンディングボックスと地面のバウンディングボックスの交差判定を行う。交差していなかったら、物体は地面と衝突していないと判定する。
2. 視線方向を高さ方向に設定し、平行投影でスクリーンの１ピクセルがハイトフィールドの１セルに対応するようにセットする。これにより地面の高さとデプスバッファの奥行きは同一なものとみなせる。またスクリーンのサイズは物体のバウンディングボックスを水平面に平行投影したサイズとなるようにする（図1の水平面上の太線部）。
3. 地面の高さをデプスバッファに書き込む。
4. 3.の結果をデプスバッファに残したままZバッファ法により物体の奥行き値によって奥行き値の小さい方でデプスバッファを更新する。更新された位置に対応するステンシルバッファの値を１にセットする。

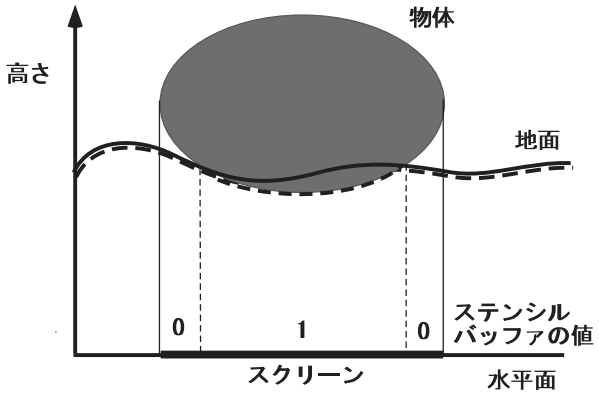


図 1 衝突判定（断面図）
Fig. 1 Collision detection.

結果として図 1の横軸のように物体が地面と交差している部分のステンシルバッファの値が１となるので衝突位置がわかり、デプスバッファには地面と物体のうち低い方の奥行きが書かれている(図 1の点線部)ので衝突した物体の底面の高さが得られる。

地面の上を物体が引きずられている場合には、物体がアニメーションのフレーム間で動いた軌跡を考慮しないと地面の正しい変形結果が得られない場合がある。たとえば球を引きずった場合、１フレームでの移動距離が１セルの幅より大きいと、図 2(a)の斜線部に示すように一フレーム前の球と現在のフレームの球の間に隙間ができてしまう。この結果地面にスジが残ってしまう。この問題を解決する単純な方法は、物体が１フレームでセルの幅より大きく動いたら、物体の動きを細かくして逐次衝

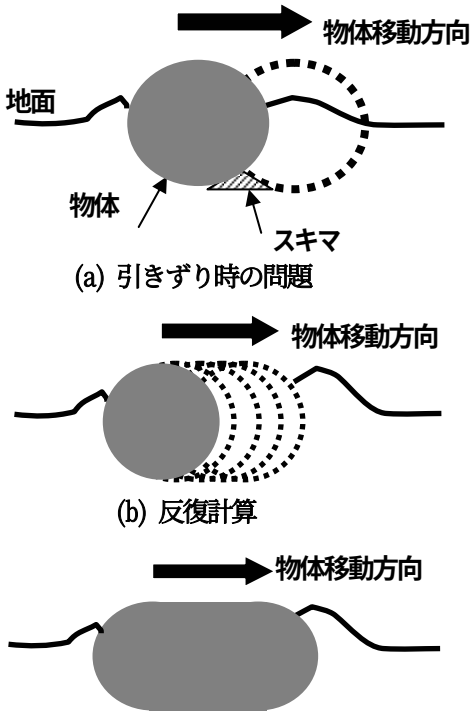


図 2 掃引した物体と地面の衝突判定
Fig. 2 Collision detection between a swept object and a ground surface.

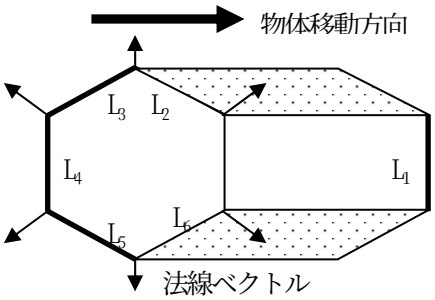


図3 掃引物体の生成（二次元）
Fig. 3 Creation of swept objects.

突判定から崩落までの計算を繰り返し行うというものであるが（図 2(b)参照）、これでは非常に計算量が大きくなってしまう。

そこで、物体が引きずられている場合にはステップ４で物体の奥行き値によってデプスバッファを更新するとき、前フレームでの物体と現在のフレームの物体の間を掃引してできた物体の奥行き値によってデプスバッファを更新する（図 2(c)参照）。掃引は物体の移動方向ベクトルと、物体を構成するポリゴンの各頂点での法線ベクトル[1]との内積を計算して、(1)内積が正負両方の値を持

[1] ここでは法線ベクトルは物体の外側を向いているもの

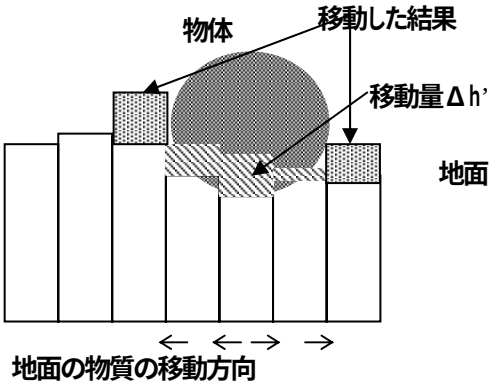


図 4 地面の物質の圧縮と移動

Fig. 4 Compression and displacement of ground material.

つ、(2)内積がすべて０、(3)内積が正と０、の三通りの中のいずれかにあてはまるようなポリゴンのみを掃引して多角柱を生成するという方法で行う。内積がすべて正となるポリゴンは現在の位置に置く。それ以外のポリゴンは前フレームでの位置に置く。次に掃引物体の生成について二次元の例を用いて説明する。図 3 は六角形の物体を掃引する場合の例である。この場合、L_1 は現在の位置に置き、L_3 から L_5 は前フレームでの位置に置く。そして、L_2 と L_6 を掃引する（図 3 の点線部）ことによって、六角形全体についての掃引物体を生成している。ただしここで述べた掃引物体の生成法は物体が平行移動しているときのみ使用できる。物体が回転しているときは、微小角度ずつ回転させて地面の変形を反復計算する必要がある。

3.4 地面の圧縮・移動

物体と衝突したセルにある地面の物質は圧縮されるか周囲に分散される（図 4）。分散される物質の量は$\Delta h = \alpha \Delta h'$となる。ここで、$\Delta h'$は物体と地面の交差部分の高さであり、物体と衝突したセルでのハイトフィールドの高さとデプスバッファの値の差から求められる。またαは分散率でユーザーが指定する。衝突領域の内側のセルから順にΔh を計算し、衝突領域の境界に近い方のセルに分配する[4]。

物体が引きずられている場合、すなわち移動方向が水平より上向きの場合には、現実には物体の移動方向の反対側には地面が分配されない。そこで、この場合は掃引した物体と地面との衝突領域内で、物体の移動方向の反対側から順に地面の圧縮・移動計算を行う。このとき、物体と地面の衝突範囲の外側へも分配する。ただし、物体の移動方向と反対側には分配させないようにする。

上述のアルゴリズムを用いて地面の圧縮・移動計算を

とする。

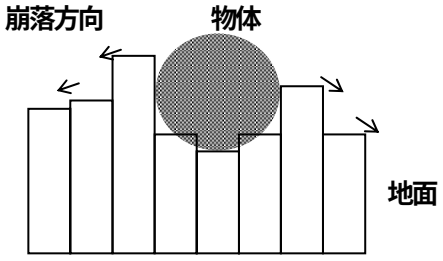


図 5 崩落

Fig. 5 Erosion.

行った後、衝突領域の輪郭部分のセルの高さがその位置での物体底面の高さより高くなる場合がある。この場合は衝突領域の外側のセルにも地面の物質を分配させて、各セルの高さが物体底面より低くなるようにする。

3.5 崩落

一般的に柔らかい地面では、ある位置での傾きが閾値より大きくなると崩落が起こる。本研究では基本的には Sumner 等の崩落計算法[4]、すなわち隣接するセル間の傾きがθ（地面の物質に固有な崩落後の傾斜）より大きい場合、高い方のセルから低い方のセルに地面の物質を移動させるという方法を用いる。このときの移動量を、$\sigma \Delta h$ とする。ここで、σは崩落後の地面の斜面の粗さを決めるパラメータであり、値が小さいほどなめらかになる。本稿ではσの値を 0.2 とした。また、Δh は隣接するセル間の高さの差である。

提案法で用いる崩落計算法は従来法[4]に以下の修正を施したものである。まず、圧縮・移動処理後、物体と地面の衝突範囲の境界部分のセルに地面の物質が蓄積されるので、そこから周囲のセルに向かって順に、衝突範囲の周辺のみ局所的に崩落計算を行うように最適化する（図 5 参照）。また、Sumner 等の方法では崩落計算中は物体の存在を考慮していないため、地面が物体と交差してしまう場合がある。そこで、崩落計算時にセルの高さがその位置での物体底面の高さより高くならないように制限を加える。

物体が地面から離れた後に崩落が起こる場合のような、崩落の時間経過を表現するために、どのセルで崩落が起こったかを一定時間記憶しておく。そして、記憶されたセルについて各タイムステップで崩落計算を行う。

3.6 多重解像度のハイトフィールドによる最適化

基準となる解像度のハイトフィールドより低解像度のハイトフィールドをいくつか準備しておき、以下に示すような場合には低い解像度のハイトフィールドを用いて地面の変形の計算を行うことによって地面の変形を効率的に行うことができる。

A)　物体が視点から遠い位置に存在する場合

B)　砂など崩落が起こりやすい地面の上に単純な形状の物体を置いた場合や引きずった場合など、物体が残す跡の形状が比較的単純になる場合

低解像度のハイトフィールドを用いて地面を変形した結果は、変形された部分だけ線形補間によって基準解像度のハイトフィールドに変換し、基準解像度のハイトフィールドだけをレンダリングする。この方法は Benes らの方法[13]と類似しているが、彼等の方法はハイトフィールド全体について地形の侵食計算を行うものである。それに対して提案法は、上述の A)、B)のような場合だけ、物体と地面の衝突判定から地面の物質の圧縮・移動、崩落までを低解像度で計算するという点が異なる。ハイトフィールドを切り替える基準は A)の場合、物体が接触した部分の地面のメッシュをスクリーンに投影したときのメッシュの面積とピクセルの面積の比率である。また B)の場合はユーザーが経験的に指定する。ただし、B)の場合は単純に低解像度のハイトフィールドを用いて地面の変形を計算しただけでは、物体の位置が視点から近い場合に変形結果の粗さが目立つ。そこで、前節で述べたように、崩落が起こったセルを記憶しておく際に高解像度のセルを記憶しておく。これにより物体衝突時に低解像度のハイトフィールドで地面を変形しても次のタイムステップで地面の形状がスムーズになる。

同様に多重解像度のハイトフィールドを用いることによって、衝突判定時に低解像度のハイトフィールドと物体の衝突判定を行って、衝突していた場合のみ高解像度のハイトフィールドとの衝突判定を行う場合や、エリアシングの生じやすい、物体と地面の衝突領域の輪郭付近だけ高解像度のハイトフィールドを用いて地面の変形計算を行うこともできる。

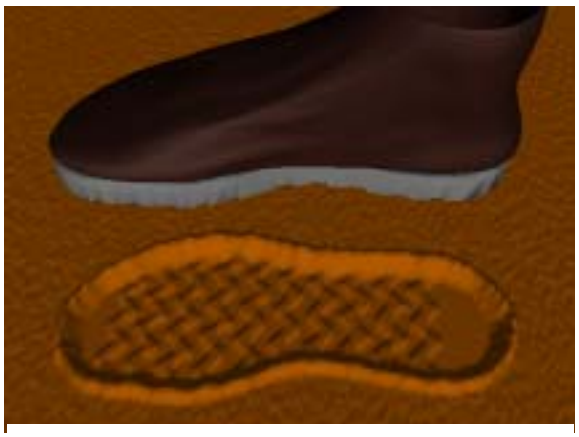


図 8 地面が土の場合の足跡
Fig. 8 A footprint on mud.

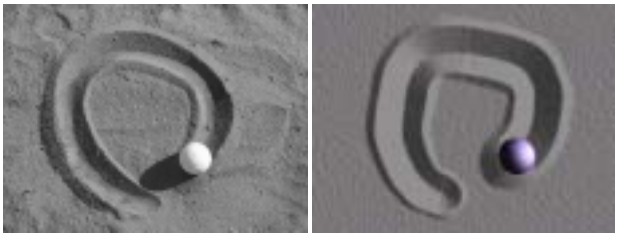


図 6 実写画像(左)と提案法による計算結果(右)の比較
Fig. 6 Comparison of Photograph (left) and a result of the proposed method (right).

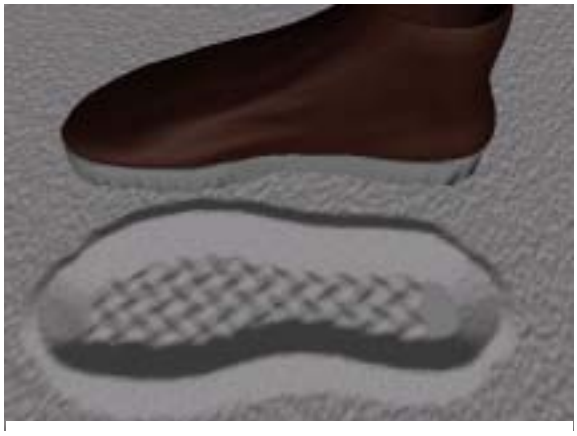


図 7 地面が砂の場合の足跡
Fig. 7 A footprint on sand.

4. 結果と考察

この節では提案法による計算結果を示す。

まず、砂の上で球を引きずってできた跡を実写画像と比較した。図 6（左）が実写画像で図 6（右）が提案法による計算結果である。提案法によりリアルな跡が計算できていることがわかる。

次に砂、泥、雪でできた地面に靴で足跡をつけた場合の計算結果を示す。図 7 は地面が砂の場合で用いたパラメータは、傾斜$\theta=\pi/6$、分散率$\alpha=1.0$ である。図 8 は泥の場合でパラメータは、$\theta=\pi/3$、$\alpha=0.8$ である。そして図 9 は雪の場合でパラメータは、$\theta=\pi/2$、$\alpha=0.0$ である。これらの結果が示すように、パラメータを変えることによって様々な種類の地面を表現することができる。

衝突判定時に物体を掃引して物体の軌跡を考慮した結果と、考慮していない場合の結果の比較を図 10 に示す。物体を掃引したことによってスムーズな変形結果が得られていることがわかる。また表 1に、(1)物体の軌跡を考慮しない場合、(2)物体の移動距離を小さく分割して地面の変形を反復計算をした場合、(3)物体を掃引した場合、の３通りの計算時間を比較した結果を示す。計測時

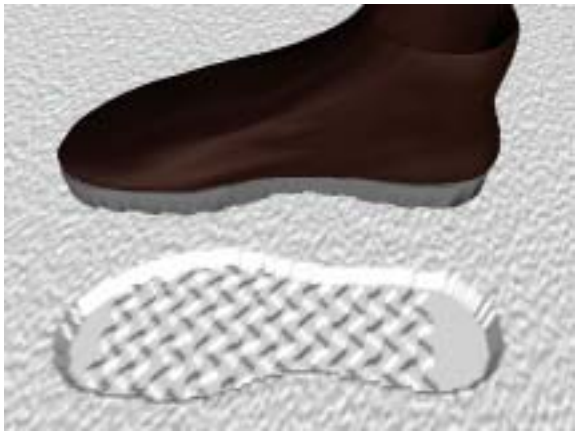


図 9 地面が雪の場合の足跡
Fig. 9 A footprint on snow.

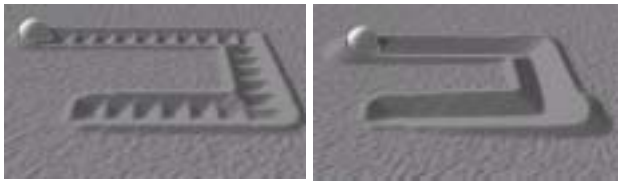


図 10 衝突判定時に多面体の掃引を行う効果を示す例（左が掃引しない場合、右が掃引した場合）
Fig. 10 A ball dragged with/without sweeping process at the collision detection step.

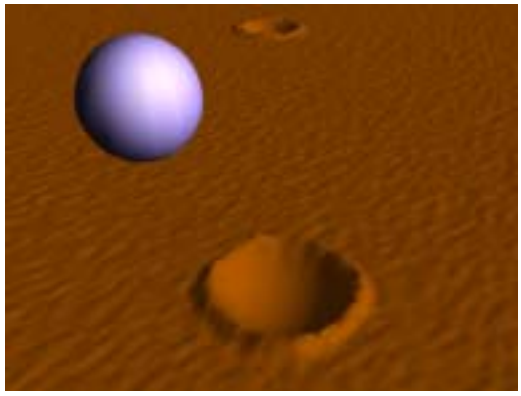
の物体の動きは１フレームでハイトフィールドの１０セル分、水平に移動した場合であり、表 1に示したのは１フレームの地面変形に要した時間である。(2)の場合、ここでは１０回の反復計算を行った。表 1から、物体を掃引することにより効率的に物体が引きずった場合の地面の変形を計算できていることがわかる。

表 1 物体をひきずった場合の計算時間の比較（単位は秒）

(1) 掃引なし	(2) 反復計算	(3) 掃引
0.022110	0.220487	0.026525

次に従来法[3]との計算時間の比較を行った。図 10 で使用したのと同じ物体を、砂の上に置いた場合で計算速度を計測した結果、従来法では、約 4fps(frames per second)であったが、提案法では約 19fps であった。すなわち、提案法は従来法に比べて約５倍の計算速度となった。

多重解像度のハイトフィールドを用いた計算結果を図 11 と図 12 に示す。図 11 は物体と視点の距離に応じて地面の変形計算に用いるハイトフィールドの解像度を変え



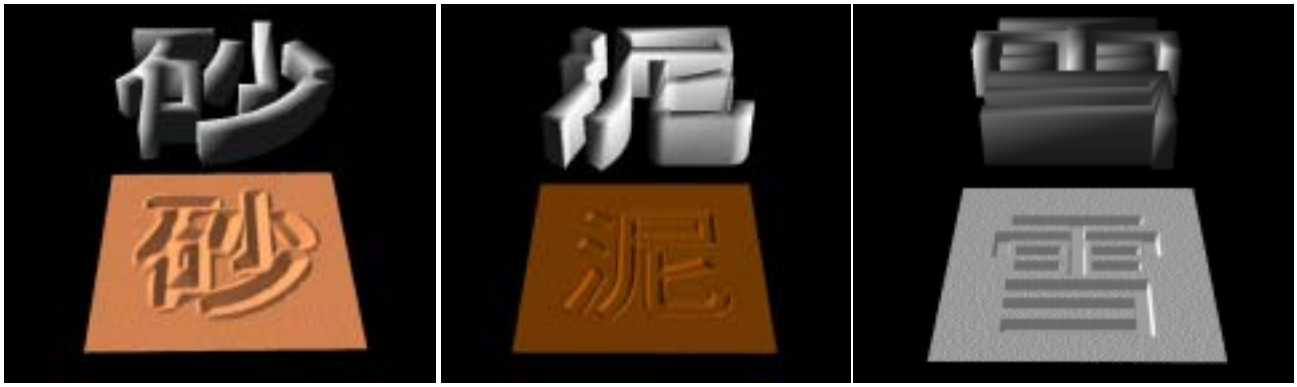
(a) 泥の上に球を置いて跡をつけた例

(b) 砂の上に自動車がタイヤ跡をつけた例

図 11 視点からの距離によって変形計算に用いるハイトフィールドの解像度を変化させた結果
Fig. 11 A result of changing the resolution of a height field due to distances between an object and the viewpoint.

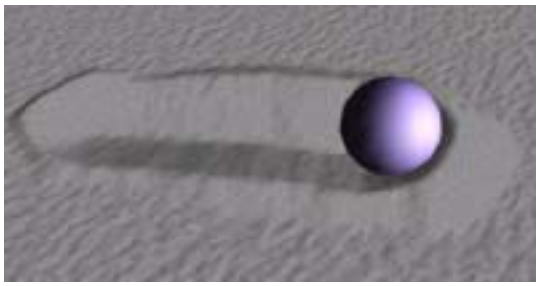
て、球を泥に置いて跡をつけた例(図 11(a))と砂の上に自動車でタイヤ跡をつけた例(図 11(b))である。視点から遠いほうの跡は近いほうの跡に比べて２分の１の解像度のハイトフィールドで計算したため衝突跡の形状が粗くなっているが、視点から遠いため、結果画像中では視点に近いほうの跡と同等の解像度に見えることがわかる。図 12 は比較的単純な物体を砂の上で引きずるときに、地面の変形計算時にハイトフィールドの解像度を変えて実験した結果である。このように砂のような崩れやすい地面に比較的単純な物体の跡をつける場合は、解像度を低くしても解像度を変えない場合とほぼ同等の結果が得られる。また、このときの計算速度は(a)の場合約 25fps、(b)の場合約 16fps となり、低解像度のハイトフィールドを用いることによって約 1.6 倍高速に計算できた。

その他の、より複雑な場合の結果を次に示す。図 13 は素足によって砂漠の表面上につけられた足跡である。足跡がつけられる前に、砂表面上に風紋を生成した[14]。図 14 は砂の上で様々な物体を引きずった跡である。そ

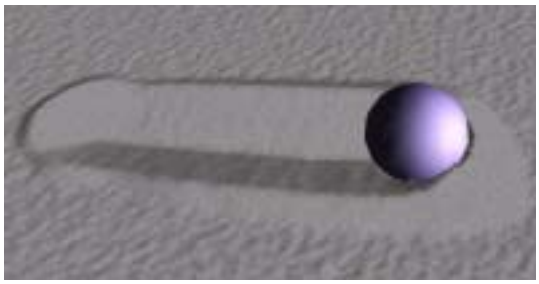


(a) 砂の場合　(b) 泥の場合　(c) 雪の場合

図 15 砂、泥、雪に文字の形をした物体を置いて跡をつけた例
Fig. 15 Marks created by placing letters.



(a) 基準解像度のハイトフィールドの 1/2 の解像度のハイトフィールドで地面の変形を計算した結果



(b) 基準解像度のハイトフィールドを用いて地面の変形を計算した結果

図 12 ハイトフィールドの解像度を変化させた結果
Fig. 12 Results of changing a resolution of a height field.

して、図 15 は地面が砂、泥、雪の場合に、それぞれ文字の形をした物体を地面に置いてできた跡である。図 16 は泥の上にタイヤ跡をつけた例である。

この節で示したすべての例について、計算時間は CPU Pentium4 2.26GHz、ビデオカード GeForce4 MX420 を用いて計測した。また、用いたハイトフィールドの解像度は 256×256 である。

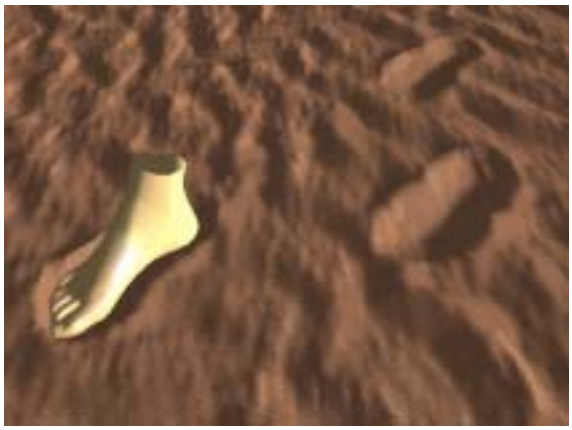


図 13 風紋上の足跡
Fig. 13 Footprints on wind-ripples.

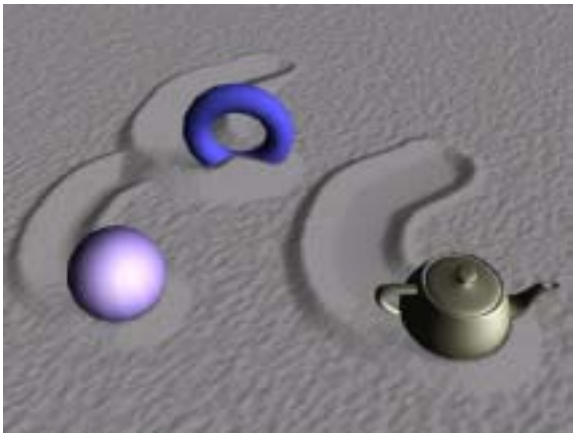


図 14 砂の上で複数の物体を引きずった跡
Fig. 14 Multiple objects dragged on sand.

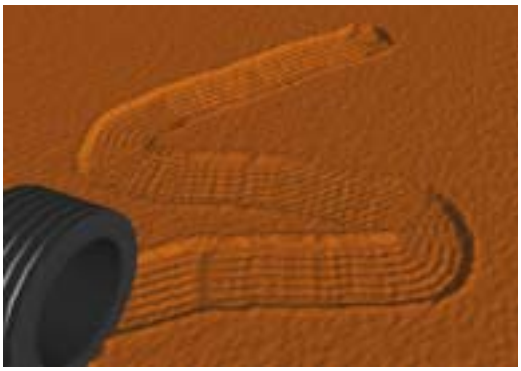


図 16 泥の上のタイヤ跡
Fig. 16 Tire marks on mud.

5. おわりに

本稿では物体と地面との接触によってできる痕跡の生成法について提案した。提案法には以下のような特徴がある。

- グラフィックスハードウェアを利用した物体と地面の衝突判定や、崩落計算の最適化により、物体と地面の接触跡をインタラクティブな速度で計算できる。
- 物体のタイムステップ間の動きの軌跡を考慮して地面と物体の衝突判定を行い、物体の移動方向を考慮して地面の移動を計算することによって、物体が地面の上を引きずられた場合の地面の変形を高速に計算できる。
- 多重解像度のハイトフィールドを用いることによって、視点から離れた物体や単純な物体による跡を効率的に計算できる。
- パラメータの設定により砂、泥、雪などさまざまな地面が表現できる。

提案法では地面が物体に被さらない場面を想定して地面のモデルにハイトフィールドを使用した。しかし、例えば現実には地面が物体に被さる場合も多い。また、提案法では地面の変形を計算する際に地面の物質の慣性を考慮していないので、物体の動きが激しい場合などに地面の物質が物体の移動していた方向に飛び散るようなことがなく、地面の変形結果が不自然なものになる。これらのような場合の表現を今後の課題としたい。

参考文献

[1] J. F. O'Brien, Victor B. Zordan, and J. K. Hodgins: "Combining active and passive simulations for secondary motion," IEEE Computer Graphics and Applications, vol. 20, no. 4, pp. 86-96, July/August 2000.

[2] X. Li and J. M. Moshell: Modeling soil: "Realtime dynamic models for soil slippage and manipulation," ACM Computer Graphics (SIGGRAPH '93 Proceedings), pp. 361-368, August 1993.

[3] B. Chanclou, A. Luciani, and A. Habibi: "Physical Models of Loose Soils Dynamically Marked by a Moving Object," Computer Animation '96, pp. 27-35, 1996.

[4] R. W. Sumner, J. F. O ' Brien, and J. K. Hodgins: "Animating sand, mud, and snow," Computer Graphics Forum, vol. 18, no. 1, pp. 3-15, 1999.

[5] J. F. O ' Brien and J. K. Hodgins: "Dynamic simulation of splashing fluids," Computer Animation '95, pp. 198-205, April 1995.

[6] M. Courshesnes, P. Volino, and N. M. Thalmann: "Versatile and effcient techniques for simulating cloth and other deformable objects," ACM Computer Graphics (SIGGRAPH '95 Proceedings), pp. 137-144, August 1995.

[7] D. Baraf and A. Witkin: "Large steps in cloth simulation, " ACM Computer Graphics (SIGGRAPH '98 Proceedings), pp. 43-54, 1998.

[8] T. Nishita, H. Iwasaki, Y. Dobashi, and E. Nakamae: "A modeling and rendering method for snow by using meatballs," Computer Graphics Forum, vol. 16, no. 3, pp. 357-364, August 1997.

[9] P. Fearing: "Computer modelling of fallen snow," ACM Computer Graphics (SIGGRAPH '00 Proceedings), pp. 37-46, 2000.

[10] H. Yamachi and Y. Shindo: "A technique for object and collision detection by z-buffer," IPSJ Journal, vol. 43, no. 6, pp. 1899-1909, 2002.

[11] J.C. Lombardo, M.P. Cani, and F. Neyret: "Real-time collision detection for virtual surgery," Computer Animation, pp. 82-90, May 1999.

[12] J. Rossignac, A. Megahed, and B. Schneider: "Interactive inspection of solids: Cross-sections and interferences," ACM Computer Graphics (SIGGRAPH '98 Proceedings), pp. 353-360, July 1992.

[13] B. Benes, I. Marak, P. Simek, P. Slavik: "Hierarchical erosion of synthetical terrains," SCCG, pp. 93-100, June 1997.

[14] K. Onoue and T. Nishita: "A method for modeling and rendering dunes with wind-ripples," Pacific Graphics, pp. 427-428, October 2000.