パーティクルマッピング

齋藤宏治、高桑昌男
国際情報科学芸術アカデミー

Particle Mapping

Koji Saito, Masao Takakuwa
International Academy of Media Arts and Sciences

アブストラクト

本論文において、新しい、絵画的レンダリングアニメーション手法を提案する。既存の絵画的アニメーションの製作方法“Painterly Rendering for Animation”(1996,SIGGRAPH)では１フレーム毎に、色情報、ストロークの向き、ストロークのサイズを指定する必要があった。我々が開発した方法、パーティクルマッピングを用いると、これらの情報を一括し生成することができる。

先の論文では絵画調レンダラは単にＺバッファを利用しているため、実際のイメージ生成には、レイヤー合成等のテクニックを使う必要があった。我々が開発した絵画調レンダラでは、このようなテクニックは必要なく、実用レベルで絵画的アニメーションの製作が可能になる。

１．はじめに

Meier[1]により、ちらつき（shower door effect）のない絵画的レンダリングアニメーションの製作が可能になった。

この方法は、モデル情報から生成されるパーティクルを利用しイメージを描く。このパーティクルには最終的なレンダリングに必要な色などの情報が不足している。このため、リファレンスピクチャが必要不可欠である。

本論文の方法では、リファレンスピクチャは必要ない。絵画調レンダラに必要な情報はパーティクルマップをもとに生成され、かつ、それらの情報はパーティクルの付加情報としてパーティクルに埋め込まれる。絵画調レンダラが必要とするパーティクルの属性情報はすべてパーティクルに含まれるので、他の画像データを参照する必要はない。

２．既存の方法

Meier の方法ではモデル表面にパーティクルを生成し、その位置について３Ｄ２Ｄ変換を行うことで、ブラシストロークを作成する。また、通常の方法でレンダリングしておいた画像を色のリファレンスピクチャとして用意しておく。ストロークの方向やサイズといった参照情報も同様にリファレンスピクチャとして用意する。

パーティクルを３Ｄ２Ｄ変換した後、２次元座標から、色、サイズ、方向をリファレンスピクチャから読み取り、ブラシのストローク方向・色、サイズを決定する。このブラシストロークを絵画調レンダラが処理して最終的な画像が生成される。

この方法はＺバッファを用いているので、静物画のように接近したモデルを処理するには、レイヤーを用い複数回レンダリングする必要があった。

本論文で提案するパーティクルマッピングではこのようなレイヤー合成の必要はない。パーティクルを生成し、絵画調レンダラでレンダリングすれば絵画的レンダリングアニメーションが生成できる。

３．パーティクルマッピング

絵画的レンダリングアニメーションに必要な手続は、パーティクル生成と絵画調レンダラである。まず初めにパーティクルマッピングを用いたパーティクル生成に関し説明する。

絵画調の絵を作成するパーティクルに必要な情報は以下の通りである。

・パーティクルの位置
・パーティクルの向き
・パーティクルの色
・uv 空間での位置
・モデルＩＤ

パーティクルの位置および向き等は、モデル情報より生成する。しかし、色情報はレンダリングしなければ得ることができない。

本論文で我々が提案する方法は、レイトレーシングを用い、色情報を含んだパーティクル生成が可能である。

パーティクルマッピングが適応可能な形状は微小三角形で近似されたモデルに限定する。ほとんどのレンダラが微小三角形で近似した後、レンダリングを行うため、この制限は一般性を失わない。

以下、実際にパーティクルが生成されるプロセスを追いながら、パーティクルマッピングの説明を行う。

３．１．マッピングデータ

実際にパーティクルを生成するためには、パーティクルリストと、ポインタマップの２つを用意し、モデルに貼り付けなければならない。

パーティクルリストは、生成され得るパーティクルのモデル表面上の uv 座標からなるリストである。リスト中の i 番目の要素の値を list(i)と表すことにする。また list(i)を i 番目のパーティクルシードと呼ぶことにする。

ポインタマップは、パーティクルリストより生成される２次元のマッピングデータである。(u_0,v_0)におけるポインタマップの値を $map(u_0,v_0)$ と表すと、$map(u_0,v_0)$には(u_0,v_0)に最も近いパーティクルシードの番号が格納される。

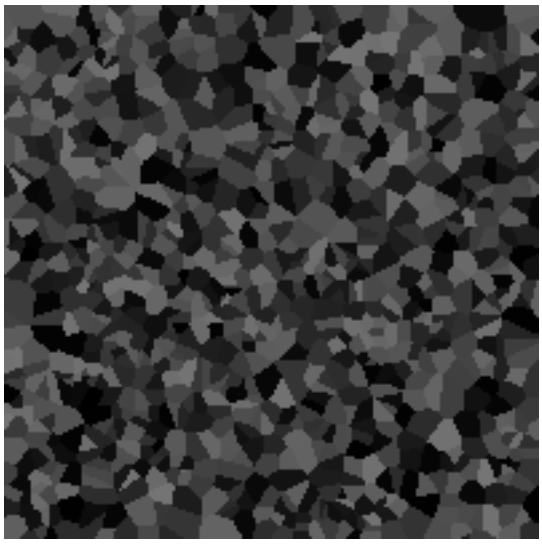

図１：ポインタマップの例

その他モデルに付加される情報として、**吸着半径**がある。吸着半径の意味は次の「パーティクルの生成」で説明する。また、各モデル毎にブラシの太さやブラシの種類等のパラメータを与える場合もある。

３．２．パーティクルの生成

レイトレーシングにより、最も視点の近くにあ

る交点についてのみパーティクルの生成プロセスを実行する。まず、交点の uv 座標(u_0,v_0)より、ポインタマップを参照する。これにより、吸着すべきパーティクルシードが決定される。

吸着すべきパーティクルシードの番号 i_0が決定したら、まず list(i_0)がレイとの交点と同じ微小三角形の上に存在するかを調べる。このとき同じ微小三角形上に存在しなければ、このレイでパーティクル生成は行わない。同一の微小三角形上に存在しなければならないという条件は、パーティクルシード吸着時の誤差をなくすためである。

次に list(i_0)と(u_0,v_0)の距離を測る。この距離が吸着半径以下であれば、次のステップに進み、そうでなければ、このレイにおいてパーティクルは生成しない。

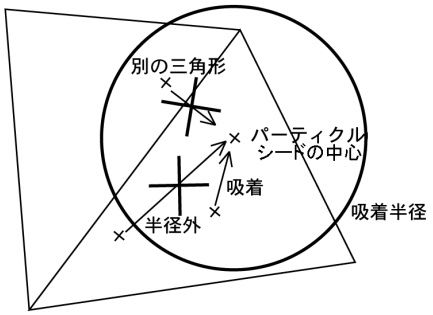

図２：パーティクルシードへの吸着

パーティクルを生成する前に、テクスチャ座標を list(i_0)にし、色および輝度計算等を行う。その結果を list(i_0)の累積色情報に加算し、 list(i_0)の吸着カウンタを一つ増やし、その後パーティクルを生成する。

i_0番目のパーティクルシードに吸着するレイは、テクスチャ座標が list(i_0)になるので、すべて同じ色になる。異なるのは輝度だけである。色情報を累積し吸着カウント数で割ることによって得られる平均の色を使うことにより影の部分のちらつきを押さえることができる。

生成されるパーティクルに含まれる情報は、

・吸着後の位置（dPdu*du+dPdv*dv）

・方向(uv 座標の基底ベクトル、dPdu or dPdv)

・色（list(i_0)の累積色情報／吸着カウント数）

・交点における法線

・その他（ブラシの太さ等）

である（ただし、(du,dv)は１次レイの交点のテクスチャ座標と list(i_0)との差である）。

以下にパーティクル生成アルゴリズムを示す。

```
各情報初期化
for all pixels {
  レイを飛ばす;
  P=最も視点に近い１次レイの交点;
  (u,v)=交点の uv 座標;
  ar=吸着半径;
  N=P における法線;
  i=map(u,v);
  if(list(i)と(u,v)が同一微小三角形上にない) {
    continue;
  }
  if(|list(i)-(u,v)|>ar) {
    continue;
  }
  (du,dv)=list(i)-(u,v);
  (u,v)=list(i);
  color=ピクセルの色
  colorTable[i]+=color;
  countTable[i]++;
  particle.position=P+dPdu*du+dPdv*dv;
  particle.direction=dPdu(または dPdv);
  particle.color=colorTable[i]/countTable[i];
  particle.normal=N;
  particle.modelID=モデル ID;
  particle.uv=list(i);
  パーティクル書出し;
}
```

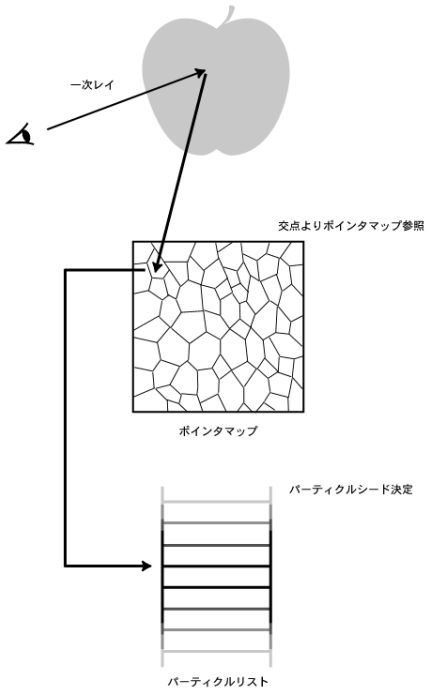

図３：パーティクル生成メカニズム

パーティクルマッピングを用いパーティクル生成を行うと、「見える」部分のパーティクルのみが生成される。絵画調レンダラは物理法則を利用しないので、見えない部分のパーティクルは必要無い。

４．絵画調レンダラ

必要な情報を持ったパーティクルが生成されたので、次は絵画調レンダラについて述べる。

我々が開発した絵画調レンダラも基本的にＺバッファであるが、単純にＺの値だけではソートしない。

４．１ブラシサイズ不要

Meier の絵画調レンダラにはブラシサイズが必要であったが、我々の方法では、ブラシストロークを３次元空間内に配置するので、ストロークのサイズは必要としない。視線方向に傾いているストロークは必然的にスケーリングがかかる。

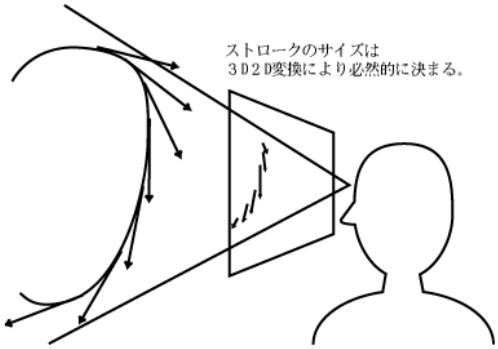

図４

４．２．プライオリティの評価

絵画調レンダラのプライオリティ評価は次のとおりである。

・モデルＩＤが異なる場合：Ｚにより評価。

・モデルＩＤが同じ場合：

Case1：uv 座標により評価。

Case2：明度と彩度により評価。

＊どちらを選択するかはユーザが決定する。

これらの評価基準により、モデル毎や、陰の部分とハイライトの部分といったようにレイヤーに分けてレンダリングする必要がなくなった。

明度と彩度を用いた評価関数の例：

Z=明度－彩度

５．インプリメント

今回は RayCustom[2]にパーティクルマッピングの機能を追加しパーティクルを生成させた。その後、新たに開発した絵画調レンダラに、そのパーティクルを入力し、最終的なイメージを生成させた。

一般的に微小三角形で表現されたモデルには

dPdu、dPdv などの情報が付加されていない場合が多い。付録に三角形の各頂点の uv 座標より dPdu、dPdv を求める方法を挙げておく。

６．評価

サンプル画像の生成は PentiumPro 200Mhz の WindowsNT 上で行った。パーティクル生成に５分、絵画調レンダリングに５分要した。

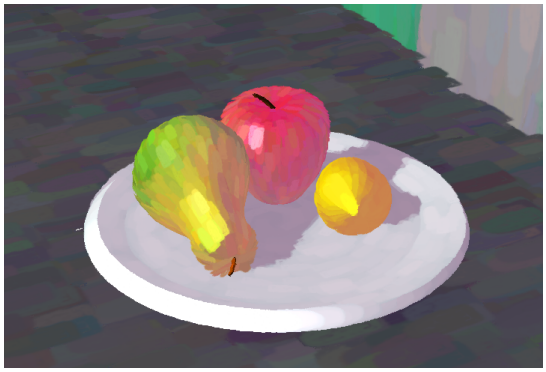

図５：サンプル画像

解像度：720×486

9506 パーティクル

７．まとめと今後の課題

絵画的なレンダリングアニメーションはパーティクルマッピングの１つの応用例にすぎない。パーティクルマッピングにより生成されたパーティクルにはレンダリング後の色情報をはじめ、各種情報を付加することができるので、それらの情報を利用したさまざまな、表現が可能であると期待される。

一方、より絵画らしい表現のためにブラシストロークにマチエールを持たせることも考えられる。

また、ストロークの方向として今回は dPdu や dPdv という１階の微分係数を用いたが、d^2P/du^2 や d^2P/dv^2 等を用いることにより、ストロークを曲げることも可能である。

参考文献

[1] Barbara J.Meier ,“Painterly Rendering for Animation” , In proceeding of SIGGRAPH’96 . Computer Graphics Proceedings, Annual Conference Series, pages 477-484.

[2] 高桑昌男,「拡散レイトレーシング」、NICOGRAPH 論文集 1993、pages 1-9.

A．付録　頂点に uv 座標情報をもつ微小三角形の dPdu,dPdv の求め方

３点 P_1,P_2,P_3 にそれぞれ(u,v)座標が与えられている。これを$(u_1,v_1),(u_2,v_2),(u_3,v_3)$とする。φ および ψ を、

$$\varphi=(u_2-u_1,v_2-v_1)$$

$$\psi=(u_3-u_1,v_3-v_1)$$

とし、さらに φ と ψ が一次独立であるならば、当然、,u 方向の基底ベクトル e_u および v 方向の基底ベクトル e_v は φ と ψ の一次結合として、

$$e_u=a_{11}\varphi+a_{12}\psi$$

$$e_v=a_{21}\varphi+a_{22}\psi$$

と表すことが可能である。この等式より $a_{11},\ldots,a_{22}$ が求まり、dPdu,dPdv は

$$dPdu=a_{11}P_1\ P_2+a_{12}P_1\ P_3$$

$$dPdv=a_{21}P_1\ P_2+a_{22}P_1\ P_3$$

であるので、dPdu,dPdv の値が求まる。ただし、P_nP_m は P_n から P_m へのベクトルである。

φ と ψ が一次従属の場合は、 $a_{11},\ldots,a_{22}$ を求めることが不可能なので、その微小三角形に対しては、パーティクルの生成は行えない。