

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4078716号
(P4078716)

(45) 発行日 平成20年4月23日 (2008. 4. 23)

(24) 登録日 平成20年2月15日 (2008. 2. 15)

(51) Int. Cl.

F I

G 0 6 T 15/50 (2006.01)

G 0 6 T 15/50 2 0 0

請求項の数 18 (全 19 頁)

(21) 出願番号	特願平10-181873	(73) 特許権者	000002185
(22) 出願日	平成10年6月29日 (1998. 6. 29)		ソニー株式会社
(65) 公開番号	特開2000-20746 (P2000-20746A)		東京都港区港南1丁目7番1号
(43) 公開日	平成12年1月21日 (2000. 1. 21)	(74) 代理人	100082131
審査請求日	平成17年6月24日 (2005. 6. 24)		弁理士 稲本 義雄
		(72) 発明者	大森 睦弘
			東京都品川区北品川6丁目7番35号 ソ
			ニー株式会社内
		(72) 発明者	ユーゴ ハベッツ
			東京都品川区北品川6丁目7番35号 ソ
			ニー株式会社内
		審査官	伊知地 和之

最終頁に続く

(54) 【発明の名称】 画像処理装置および方法、並びに提供媒体

(57) 【特許請求の範囲】

【請求項 1】

レイトラッシング方式により3次元座標空間上の複数のオブジェクトのスクリーン上の画像データを生成する画像処理装置において、

前記3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力手段と、

前記スクリーン座標系の画像データから、前記表面の内部の画素データを補間する補間手段と、

前記補間手段により補間された画素データを、それに対応するレイによるレイトラッシングの成分を演算して補正する補正手段と、

前記補正手段により補正された画素データを描画する描画手段と
を備えることを特徴とする画像処理装置。

【請求項 2】

レイトラッシング方式により3次元座標空間上の複数のオブジェクトのスクリーン上の画像データを生成する画像処理装置の画像処理方法において、

前記3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力ステップと、

前記スクリーン座標系の画像データから、前記表面の内部の画素データを補間する補間ステップと、

前記補間ステップで補間された画素データを、それに対応するレイによるレイトラシ

10

20

ングの成分を演算して補正する補正ステップと、

前記補正ステップで補正された画素データを描画する描画ステップと
を含むことを特徴とする画像処理方法。

【請求項 3】

レイトレーシング方式により 3 次元座標空間上の複数のオブジェクトのスクリーン上の
画像データを生成する画像処理装置に、

前記 3 次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより
生成されたスクリーン座標系の画像データを出力する出力ステップと、

前記スクリーン座標系の画像データから、前記表面の内部の画素データを補間する補間
ステップと、

前記補間ステップで補間された画素データを、それに対応するレイによるレイトレーシ
ングの成分を演算して補正する補正ステップと、

前記補正ステップで補正された画素データを描画する描画ステップと
を含むことを特徴とするプログラムを提供する提供媒体。

【請求項 4】

レイトレーシング方式により 3 次元座標空間上の複数のオブジェクトのスクリーン上の
画像データを生成する画像処理装置において、

第 1 のレイによるレイトレーシングに代えて、前記 3 次元座標空間上のオブジェクトの
表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像デー
タを出力する出力手段と、

前記スクリーン座標系の画像データから、前記表面の内部の画素データを補間する補間
手段と、

前記補間手段により補間された画素データを、それに対応する第 2 のレイによるレイト
レーシングの成分を演算して補正する補正手段と、

前記補正手段により補正された画素データを描画する描画手段と
を備えることを特徴とする画像処理装置。

【請求項 5】

前記補間手段により補間された画素データの奥行き方向の座標を基に、最も手前にある
画素データを検索し、記憶する記憶手段を

さらに備えることを特徴とする請求項 4 に記載の画像処理装置。

【請求項 6】

前記オブジェクトの表面の代表的な奥行き座標を基に、前記オブジェクトの表面をソー
トするソート手段をさらに備え、

前記補正手段は、前記ソート手段によりソートされた順序で、奥から順番に、前記第 2
のレイによるレイトレーシングの成分を演算して補正処理を行う

ことを特徴とする請求項 4 に記載の画像処理装置。

【請求項 7】

前記補間手段は、補間演算ごとに、その奥行き座標を記憶し、

前記補正手段は、前記補間手段に記憶されている奥行き座標を基に、前記第 2 のレイに
よるレイトレーシングの成分を演算して補正処理を行う

ことを特徴とする請求項 4 に記載の画像処理装置。

【請求項 8】

前記 3 次元座標空間上のオブジェクトの表面の画像は、トライアングルである

ことを特徴とする請求項 4 に記載の画像処理装置。

【請求項 9】

前記オブジェクトの表面の代表的な奥行き座標を基に、前記オブジェクトの表面をソー
トするソート手段をさらに備え、

前記補間手段は、前記ソート手段によりソートされた順序で、手前から順番に補間演算
を行い、各画素の補間演算ごとに、その奥行き座標を記憶し、

前記補正手段は、前記補間手段に記憶されている奥行き座標を基に、前記第 2 のレイに

10

20

30

40

50

よるレイトレーシングの成分を演算して補正処理を行う
ことを特徴とする請求項4に記載の画像処理装置。

【請求項10】

前記補間手段と補正手段は、手前に位置する他の前記オブジェクトの表面が既に描画されている場合、その処理をスキップする

ことを特徴とする請求項4に記載の画像処理装置。

【請求項11】

前記補間手段は、透明度が所定の基準値以下の前記オブジェクトについては、第2のレイまでの処理を行い、透明度が前記基準値より大きい前記オブジェクトについては、前記透明度が前記基準値以下の前記オブジェクトに衝突するまで、より高次のレイの処理を行う

10

ことを特徴とする請求項4に記載の画像処理装置。

【請求項12】

前記補正手段は、前記補間手段による補間演算された画素の反射ベクトルまたは屈折ベクトルの値が、所定の基準値以下のとき、その画素の補正を行わない

ことを特徴とする請求項4に記載の画像処理装置。

【請求項13】

前記補正手段は、影を生成する処理を行う場合、または反射もしくは屈折レイの処理を行う場合において、複数の前記オブジェクトとの衝突判定を順次行うとき、所定のスパンにおいて、直前の画素が衝突した前記オブジェクトとの衝突判定を最初に行う

20

ことを特徴とする請求項4に記載の画像処理装置。

【請求項14】

前記補正手段は、影を生成する処理を行う場合、反射もしくは屈折レイの処理は第2のレイまで行い、シャドウレイの処理は、第3のレイまで行う

ことを特徴とする請求項4に記載の画像処理装置。

【請求項15】

前記描画手段は、前記オブジェクトの透明度が所定の基準値より大きいとき、奥から順番に描画するとともに、既に描画されている画素とブレンドする

ことを特徴とする請求項4に記載の画像処理装置。

【請求項16】

30

前記描画手段は、1つの画素に複数の前記オブジェクトが対応する場合、前記画素を覆う面積に基づいて重み付けを行う

ことを特徴とする請求項4に記載の画像処理装置。

【請求項17】

レイトレーシング方式により3次元座標空間上の複数のオブジェクトのスクリーン上の画像データを生成する画像処理装置の画像処理方法において、

第1のレイによるレイトレーシングに代えて、前記3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力ステップと、

前記スクリーン座標系の画像データから、前記表面の内部の画素データを補間する補間ステップと、

40

前記補間ステップで補間された画素データを、それに対応する第2のレイによるレイトレーシングの成分を演算して補正する補正ステップと、

前記補正ステップで補正された画素データを描画する描画ステップと

を含むことを特徴とする画像処理方法。

【請求項18】

レイトレーシング方式により3次元座標空間上の複数のオブジェクトのスクリーン上の画像データを生成する画像処理装置に、

第1のレイによるレイトレーシングに代えて、前記3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データ

50

を出力する出力ステップと、

前記スクリーン座標系の画像データから、前記表面の内部の画素データを補間する補間ステップと、

前記補間ステップで補間された画素データを、それに対応する第2のレイによるレイトレーシングの成分を演算して補正する補正ステップと、

前記補正ステップで補正された画素データを描画する描画ステップと

を含むことを特徴とするプログラムを提供する提供媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、画像処理装置および方法、並びに提供媒体に関し、特に、レイトレーシング方式で、より迅速に、画像データを生成することができるようにした画像処理装置および方法、並びに提供媒体に関する。

【0002】

【従来の技術】

図16は、従来のレイトレーシング方式の原理を表している。同図に示すように、スクリーン52上の所定の位置の画素のデータを生成するとき、予め設定された所定の位置に配置されているアイポイント51から、スクリーン52上の、その位置の画素に向かう光線（レイ）を想定し、そのレイ（第1のレイ）が3次元座標空間上に配置されている所定のオブジェクト（図16の例の場合、オブジェクト53）に衝突させる。このとき、スクリーン52上の、その画素には、オブジェクト53の第1のレイが衝突した位置の画像が描かれるべきものとなる。

【0003】

さらに、このオブジェクト53の第1のレイが衝突する点の画像は、ライトソース55からの光が、オブジェクト54で反射され、オブジェクト53の、その点に照射されている場合、その影響も受けることになる。そこで、オブジェクト53の、その点から、ライトソース55に向けて、光線を逆にたどり、オブジェクト53からオブジェクト54に向かって、第2のレイを想定し、オブジェクト54からライトソース55に向かって、第3のレイを想定する。そして、オブジェクト53上の第1のレイの衝突する点の画像を、第2のレイおよび第3のレイに基づく影響を計算することで補正する。そして、その補正した結果が、スクリーン52上の対応する画素に反映される。

【0004】

このようにして、例えばCADにおいて、所定の3次元座標空間上に複数のオブジェクトを作成配置し、その3次元上のオブジェクトを所定のアイポイントから見た場合の画像を、スクリーン52上の画像として演算し、表示することができる。

【0005】

【発明が解決しようとする課題】

しかしながら、このようなレイトレーシングによる方法は、各段階のレイがどのオブジェクトに衝突するのかを判定しなければならず、そのための計算量が膨大なものとなり、ポリゴンレンダリング手法などに較べて、高速化が困難となる課題があった。

【0006】

本発明はこのような状況に鑑みてなされたものであり、より迅速に画像データを生成することができるようにするものである。

【0007】

【課題を解決するための手段】

請求項1に記載の画像処理装置は、3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力手段と、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間手段と、補間手段により補間された画素データを、それに対応するレイによるレイトレーシングの成分を演算して補正する補正手段と、補正手段により補正された画素データを描画

10

20

30

40

50

する描画手段とを備えることを特徴とする。

請求項 2 に記載の画像処理方法は、3 次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力ステップと、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間ステップと、補間ステップで補間された画素データを、それに対応するレイによるレイトレーシングの成分を演算して補正する補正ステップと、補正ステップで補正された画素データを描画する描画ステップとを含むことを特徴とする。

請求項 3 に記載の提供媒体のプログラムは、3 次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力ステップと、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間ステップと、補間ステップで補間された画素データを、それに対応するレイによるレイトレーシングの成分を演算して補正する補正ステップと、補正ステップで補正された画素データを描画する描画ステップとを含むことを特徴とする。

請求項 4 に記載の画像処理装置は、第 1 のレイによるレイトレーシングに代えて、3 次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力手段と、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間手段と、補間手段により補間された画素データを、それに対応する第 2 のレイによるレイトレーシングの成分を演算して補正する補正手段と、補正手段により補正された画素データを描画する描画手段とを備えることを特徴とする。

【 0 0 0 8 】

請求項 1 7 に記載の画像処理方法は、第 1 のレイによるレイトレーシングに代えて、3 次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力ステップと、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間ステップと、補間ステップで補間された画素データを、それに対応する第 2 のレイによるレイトレーシングの成分を演算して補正する補正ステップと、補正ステップで補正された画素データを描画する描画ステップとを含むことを特徴とする。

【 0 0 0 9 】

請求項 1 8 に記載の提供媒体のプログラムは、レイトレーシング方式により 3 次元座標空間上の複数のオブジェクトのスクリーン上の画像データを生成する画像処理装置に、第 1 のレイによるレイトレーシングに代えて、3 次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力ステップと、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間ステップと、補間ステップで補間された画素データを、それに対応する第 2 のレイによるレイトレーシングの成分を演算して補正する補正ステップと、補正ステップで補正された画素データを描画する描画ステップとを含むことを特徴とする。

【 0 0 1 0 】

請求項 1 に記載の画像処理装置、請求項 2 に記載の画像処理方法、および請求項 3 に記載の提供媒体のプログラムにおいては、オブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを補間することで、表面の内部の画素データが生成される。そして、補間された画素データが、レイによるレイトレーシング成分で補正される。

請求項 4 に記載の画像処理装置、請求項 1 7 に記載の画像処理方法、および請求項 1 8 に記載の提供媒体のプログラムにおいては、第 1 のレイによるレイトレーシングに代えて、オブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを補間することで、表面の内部の画素データが生成される。そして、補間された画素データが、第 2 のレイによるレイトレーシング成分で補正される。

【 0 0 1 1 】

【 発明の実施の形態 】

以下に本発明の実施例を説明するが、特許請求の範囲に記載の発明の各手段と以下の実施例との対応関係を明らかにするために、各手段の後の括弧内に、対応する実施例（但し一例）を付加して本発明の特徴を記述すると、次のようになる。但し勿論この記載は、各手段を記載したものに限定することを意味するものではない。

【 0 0 1 2 】

請求項 1 に記載の画像処理装置は、3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力手段（例えば、図 1 のパーソナルコンピュータ 1、ホストインタフェース 2）と、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間手段（例えば、図 1 の DDA 3）と、補間手段により補間された画素データを、それに対応するレイによるレイトレーシングの成分を演算して補正する補正手段（例えば、図 1 のレイトレーサ 6）と、補正手段により補正された画素データを描画する描画手段（例えば、図 1 のフレームバッファ 7）とを備えることを特徴とする。

10

請求項 4 に記載の画像処理装置は、第 1 のレイによるレイトレーシングに代えて、3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データを出力する出力手段（例えば、図 1 のパーソナルコンピュータ 1、ホストインタフェース 2）と、スクリーン座標系の画像データから、表面の内部の画素データを補間する補間手段（例えば、図 1 の DDA 3）と、補間手段により補間された画素データを、それに対応する第 2 のレイによるレイトレーシングの成分を演算して補正する補正手段（例えば、図 1 のレイトレーサ 6）と、補正手段により補正された画素データを描画する描画手段（例えば、図 1 のフレームバッファ 7）とを備えることを特徴とする。

20

【 0 0 1 3 】

請求項 5 に記載の画像処理装置は、補間手段により補間された画素データの奥行き方向の座標を基に、最も手前にある画素データを検索し、記憶する記憶手段（例えば、図 1 のプリトレースバッファ 5）をさらに備えることを特徴とする。

【 0 0 1 4 】

請求項 6 に記載の画像処理装置は、オブジェクトの表面の代表的な奥行き座標を基に、オブジェクトの表面をソートするソート手段（例えば、図 1 1 のトライアングルソート部 2 1）をさらに備え、補正手段は、ソート手段によりソートされた順序で、奥から順番に、第 2 のレイによるレイトレーシングの成分を演算して補正処理を行うことを特徴とする。

30

【 0 0 1 5 】

請求項 9 に記載の画像処理装置は、オブジェクトの表面の代表的な奥行き座標を基に、前記オブジェクトの表面をソートするソート手段（例えば、図 4 のトライアングルソート部 2 1）をさらに備え、補間手段は、ソート手段によりソートされた順序で、手前から順番に補間演算を行い、各画素の補間演算ごとに、その奥行き座標を記憶し、補正手段は、補間手段に記憶されている奥行き座標を基に、第 2 のレイによるレイトレーシングの成分を演算して補正処理を行うことを特徴とする。

【 0 0 1 6 】

図 1 は、本発明を適用した画像処理システムの構成例を表している。ホストコンピュータとしてのパーソナルコンピュータ 1 は、3次元座標空間上のオブジェクトの各表面を複数のトライアングルに分割し、各トライアングルを所定のスクリーン座標系に投影して得られた画像データを生成する。そして、そのスクリーン座標系の画像データと、光源に関する画像データを、ホストインタフェース 2 を介して、DDA (Digital Differential Analyzer) に供給する。具体的には、例えば、トライアングルの 3 つの頂点における座標 (x, y, z) 面の法線ベクトル、表面の拡散係数、表面の反射係数、屈折係数、透明度、表面の色 (R, G, B) といったデータと、光源の座標 (x, y, z)、明るさと色 (R, G, B)、光源の種類 (点、平行、スポット径) といった情報が、DDA 3 に入力される。

40

【 0 0 1 7 】

50

DDA 3 は、ホストインタフェース 2 を介して、パーソナルコンピュータ 1 より入力された各トライアングルの頂点座標を元に、そのトライアングルの内部の画素データを線形補間により生成する。生成された画素データは、Z バッファ 4 に入力され、奥行き方向（Z 軸方向）の座標を元に、ソート処理が行われる。すなわち、より手前（アイポイントに近い方）の画素データが選択され、最も手前に位置する画素データだけがプリトレースバッファ 5 に供給され、記憶されるようになされている。

【 0 0 1 8 】

レイトレーサ 6 は、プリトレースバッファ 5 に記憶された画素データに対して、第 2 のレイ以降のレイトレーシング処理を行い、プリトレースバッファ 5 に記憶された画素データを補正し、補正した画素データをフレームバッファ 7 に出力し、描画するようになされている。フレームバッファ 7 に描画された画素データは、CRT コントローラ（CRTC）8 により適宜読み出され、ディスプレイ 9 に出力され、表示されるようになされている。

10

【 0 0 1 9 】

次に、図 2 のフローチャートを参照して、その動作について説明する。最初に、ステップ S 1 において、パーソナルコンピュータ 1 は、3 次元座標空間上におけるオブジェクトを複数のトライアングルに分割し、そのトライアングルを所定のスクリーンに投影した画像データを生成する。すなわち、図 3 に示すように、オブジェクト 5 3 を複数のトライアングル 5 3 - i（ $i = 1, 2, 3, \dots$ ）に分割する。そして、各トライアングル 5 3 - i をスクリーン 5 2 に投影した画像データを生成する。図 3 においては、オブジェクト 5 3 のトライアングル 5 3 - 1 が、スクリーン 5 2 上に、トライアングル 5 3 - 1' として投影されている様子が表されている。

20

【 0 0 2 0 】

すなわち、この例においては、レイトレーシング方式における第 1 のレイ（Ray）によるトレーシングが省略され、それに代わって、このような投影処理が実行される。これにより、第 1 のレイが、どのオブジェクトに衝突するのかを判定する演算が、従来の場合の 1 / 1 0 0 程度に軽減することができる。

【 0 0 2 1 】

DDA 3 は、ステップ S 2 において、パーソナルコンピュータ 1 より供給されたトライアングルの 3 つの頂点座標を元に、そのトライアングルの内部の画素データを線形補間する処理を実行する。ステップ S 3 において、DDA 3 は、補間演算の結果得られた画素データの Z 座標を Z バッファ 4 に記憶させ、これを利用して、最も手前の画素データだけをプリトレースバッファ 5 に記憶させる。以上のようにして、第 1 のレイによるトレーシングは行われないが、行った場合と同様の画像が得られる。

30

【 0 0 2 2 】

以上のようにして、スクリーン 5 2 の 1 画面分の画像データがプリトレースバッファ 5 に記憶されたとき、次に、ステップ S 4 に進み、レイトレーサ 6 は、第 1 のレイの条件に基づいて、第 2 のレイを各オブジェクトに飛ばし、周辺光による色を決定し、その色で、ステップ S 5 において、対応する画素データを補正する。この第 2 のレイによる処理は、例えば、そのオブジェクトが光を反射しないオブジェクトであるような場合には省略される。

40

【 0 0 2 3 】

レイトレーサ 6 は、ステップ S 5 において、第 2 のレイによる周辺光による色を決定したとき、その色、テクスチャ、および、そのオブジェクトの色をブレンドし、そのオブジェクトの第 1 のレイが衝突した位置の色とする。ステップ S 6 において、レイトレーサ 6 は、最終的に決定された色を、フレームバッファ 7 の対応する画素の位置に描画する。

【 0 0 2 4 】

CRTC 8 は、ステップ S 7 において、フレームバッファ 7 に描画された画素データを読み出し、これを表示信号に変換して、ディスプレイ 9 に出力し、表示させる。

【 0 0 2 5 】

図 3 には、以上の処理が模式的に表されている。すなわち、スクリーン 5 2 に投影された

50

トライアングルの頂点の座標で規定されるトライアングルの内部の画素の光ベクトル、反射ベクトル、屈折ベクトル、法線ベクトル、 (x, y, z) 座標、 u, v ベクトルなどが生成される。

【0026】

以上のようにして、第1のレイによるトレーシング処理に代えて、画像データが生成されたとき、次に、第2のレイ以降のトレーシング処理が実行される。

【0027】

図1の実施の形態においては、最も手前に位置する画像データだけをプリトレースバッファ5に記憶させるようにしたが、以下においては、このようなプリトレースバッファ5が設けられていない場合の構成例について説明する。

【0028】

図4の構成例においては、ホストインタフェース2とDDA3の間に、トライアングルソート部21とソートバッファ22が配置されている。DDA3は、Zバッファ4を利用して、補間処理を実行するが、プリトレースバッファ5は省略されている。レイトレーサ6は、DDA3に接続され、DDA3との間でデータを授受するようになされている。DDA3はまた、画像データをフレームバッファ7に出力し、記憶させるようになされている。また、この例においては、ピクセルカバレッジバッファ23が設けられており、フレームバッファ7には、このピクセルカバレッジバッファ23の記憶値に対応して、所定の重み付け処理が実行され、画素データが描画されるようになされている。その他の構成は、図1における場合と同様である。

【0029】

次に、図5のフローチャートを参照して、その動作について説明する。最初に、ステップS21において、パーソナルコンピュータ1により、トライアングルをスクリーンに投影した画像データが生成される。この処理は、図2のステップS1における場合と同様の処理である。トライアングルソート部21は、パーソナルコンピュータ1から、ホストインタフェース2を介して、スクリーン座標系の画素データの入力を受けたとき、各トライアングルの z 座標の代表値を決定し、その代表値を元に、トライアングルをソートする。ソートした結果は、ソートバッファ22に記憶される。

【0030】

各トライアングルの z 座標の代表値としては、例えば3つの頂点の z 座標の平均値、あるいは3つの頂点のうちのいずれか任意の1つの頂点の z 座標を用いることができる。

【0031】

次に、ステップS23において、DDA3は、ソートバッファ22から入力された各トライアングルの内部の画素データを線形補間により生成する。この処理は、図2のステップS2における場合と同様の処理である。

【0032】

DDA3は、手前と判断されたトライアングルから順番に補間演算処理を行い、各画素の補間演算毎に、Zバッファ4を用いて、さらに第2のレイの計算のオン/オフを行う。

【0033】

すなわち、ステップS24において、DDA3は、処理対象とする画素が最も手前の画素であるか否かを判定し、手前の画素である場合には、ステップS25において、その画素データをレイトレーサ6に出力し、第2のレイによるデータを計算させる。レイトレーサ6は、DDA3より入力された第1のレイの条件から、第2のレイをオブジェクトに飛ばし、周辺光による色を決定し、その色のデータをDDA3に出力する。

【0034】

DDA3は、ステップS24において、対象画素が最も手前の画素ではないと判定された場合、ステップS25の処理(第2のレイの処理)を省略させる。また、DDA3は、補間演算により求められた画素の反射ベクトルの強さあるいは屈折ベクトルの強さが、予め設定してある所定の基準値以下の場合、第2のレイの計算を行わない(ステップS25の処理をスキップする)ようにすることができる。例えば、光を反射しないオブジェクトの第2

10

20

30

40

50

の処理は省略されることになる。

【 0 0 3 5 】

DDA 3 は、トライアングル（面状オブジェクト）の頂点での第 2 のレイのための情報を補間演算することにより、トライアングルの内部の第 2 のレイのための情報を計算することになるのであるが、補間演算の途中で第 2 のレイを考慮する必要がないと判断できる場合には、その処理を省略するようにすることで、第 2 のレイの演算を減少させることができる。

【 0 0 3 6 】

このように、トライアングルを奥行順にソートしておき、手前のトライアングルから順番に、その画素を補間演算し、補間演算毎に、Z バッファ 4 を用いて、第 2 のレイの計算を制御するようにすると、トライアングルは、3 つの頂点の z 座標の代表値でソートが行われているため、例えば図 6（A）に示すように、図中白で示されているトライアングルと、黒で示されているトライアングルとが交差しているような場合、最初に図 6（B）に示すように、代表値で手前と判断された白の三角形が描画された後、次に図 6（C）に示すように、代表値で後方と判断された黒で示す三角形が描画されることになる。そして、白の三角形の内部に描画されている黒のトライアングルの領域では、白の三角形の画素が一旦描画された後、その部分が白のトライアングルより前方に位置しているため、黒のトライアングルの画素として再描画（上書き）される。描画が二重に行われるので、その分、処理速度が遅くなるが、実際には、このようなトライアングルの関係（z 座標の代表値と個々の z 座標との関係が逆転するような関係）が発生する可能性は少ないので、それほど問題にはならない。

【 0 0 3 7 】

DDA 3 は、オブジェクトの透明度が予め設定してある所定の基準値より大きい場合（透明である場合）、透明度が基準値より小さい（透明でない）オブジェクトに第 2 のレイが衝突するまで、第 2 のレイの演算処理を実行する。このようにすることで、描画速度は低下するが、反射レイを第 1 のレイと同様に考慮することとなり、透明なオブジェクトを完全な形で描画することができる。オブジェクトのうち、透明なオブジェクトの占める割合が比較的少ない場合には、透明なオブジェクトについてのみ、完全な計算を行うようにしたとしても、全体としては、処理速度が大きく低下するようなことにはならない。

【 0 0 3 8 】

手前にあるトライアングルが既に描画されていて、実際の描画に関与しない画素に対しては、DDA 3 は、その補間演算と第 2 のレイ以降のレイトレーシング処理をスキップさせるようにすることができる。このようなスキップを行った場合、補間演算に使用する変分（単位当たりの変化量）にスキップする画素数を乗算した値を補間変数の新たな値として設定することができる。また、スキップした場合のスキップ後の座標値は、スキップ前の x 座標にスキップする画素の数を加算した値から求めることができる。

【 0 0 3 9 】

このように、必要のない第 2 のレイ以降の計算を省略することで、全体の演算量を減らすことができる。

【 0 0 4 0 】

また、各トライアングルのスパン（トライアングルの一方の端部から水平方向の他方の端部までの範囲）の最後の画素まで検査した結果、手前にあるトライアングルが既に描画されている場合には、検査でそのような検査結果が得られた時点において、その後の処理を省略することができる。

【 0 0 4 1 】

スパンの端部までスキップできる場合には、変分にスキップする画素の数を乗算せずに、そのままスパンの処理を終了させることができる。スパンの途中が一旦他の物体で覆われたような場合、そのスパンの端までずっと覆われている場合も多く、実際に、変分にスキップする画素の数を乗算する必要がないことになり、そのような場合、補間演算処理を大幅に減少させることができる。

10

20

30

40

50

【 0 0 4 2 】

また、レイトレーサ 6 が影の生成処理を行う場合、あるいは、反射レイ、もしくは屈折レイに関して、第 2 のレイの計算を行う場合、スパンの先頭の画素においては、例えばバウンディングボックス (Bounding box)、オクトリー (Octree) 法などにより、ある程度衝突が予想される物体との交差判定を行う。しかしながら、第 2 番目以降の画素においては、直前の画素が衝突したオブジェクトとの衝突判定を最初に行い、もしその判定において、衝突していると判定された場合には、他のオブジェクトとの衝突判定は行わないようにする。

【 0 0 4 3 】

すなわち、図 7 に示すように、所定のスパン内において、隣接する画素からのレイが所定のオブジェクトに衝突した場合、その隣に位置する画素からのレイも同一のオブジェクトに衝突する可能性が高い。

10

【 0 0 4 4 】

図 7 の例では、図中最も下側に位置している黒い画素は最初の画素であるので、関連しそうな全てのオブジェクトに対する衝突判定処理が行われるが、続く上 2 つの白い画素は、直前に隣接する画素と同一のオブジェクト (オブジェクト A とオブジェクト B) に衝突している。そこで、これらの画素からのレイに対しては、他のオブジェクトに対する衝突は判定されない。

【 0 0 4 5 】

それに対して、図中上側の黒い画素 (下から 4 番目の画素) からのレイは、直前の画素と同一のオブジェクトに衝突していないので (直前のオブジェクトは、オブジェクト A とオブジェクト B に衝突しているが、この画素のレイは、オブジェクト A には衝突しているが、オブジェクト B には衝突していないので)、関連のありそうな全てのオブジェクトとの衝突判定処理が行われる。

20

【 0 0 4 6 】

影の処理に際しては、衝突していることが分かればよく、衝突したと分かった時点で、その後の処理を打ち切ることが可能である。その結果、非常に多くの画素の計算において、特定のオブジェクトとの衝突が検出されたとき、その時点において処理を終了させることができる。その結果、比較的演算量が多い、不特定多数のオブジェクトとの衝突判定における演算量を減らすことが可能となる。

30

【 0 0 4 7 】

反射レイおよび屈折レイの場合も同様である。

【 0 0 4 8 】

また、DDA 3 は、影の生成処理を行う場合、影の計算以外の反射レイおよび屈折レイに関しては、第 2 のレイまでの計算を行い、第 3 のレイ以降の計算は行わないようにし、影の計算のため、シャドーレイに関しては、第 3 のレイまで計算するようにすることができる。

【 0 0 4 9 】

2 つの同じオブジェクトの間で繰り返し反射または屈折が発生する場合がある。このような反射および屈折に関しては、第 3 のレイ以降の計算を行っても、それほど良質な画像が得られるようになるわけではない。従って、このような計算は省略するのが好ましい。第 3 のレイ以降の計算が画質に影響する場合は、例えば鏡張りの部屋の画像を描画するような場合であるが、このような場合以外には、その計算を行う実用的な意味が殆どない。

40

【 0 0 5 0 】

しかしながら、シャドーレイに関しては、計算した第 2 のレイが影のオブジェクトに当たっているのか、影ではなく、光が当たっているオブジェクトに当たっているのかは、第 2 のレイの値に大きく影響する。従って、第 2 のレイの計算結果をできるだけ正しく使うために、シャドーレイに関しては、第 3 のレイまで計算するのが好ましい。また、シャドーレイの計算は、反射レイおよび屈折レイに較べて、その計算量は軽いものである。

【 0 0 5 1 】

50

図5に戻って、ステップS25の第2のレイのデータ計算処理が終了した後、ステップS26において、DDA3は、第2のレイで決定された周辺光による色、テクスチャ、およびオブジェクトの色をブレンドし、第1のレイが最初に衝突したオブジェクトの表面の、その衝突した位置の最終の色として合成する処理を行う。

【0052】

なお、透明なオブジェクトを描画する場合、図8(A)に示すように、手前のオブジェクトから順番に、不透明なオブジェクトに達するまで順番に描画して、第2のレイを計算して行けば、奥にあるオブジェクトの色も透けて見えることになる。しかしながら、図8(A)に示す場合、重い演算処理である反射ベクトル演算と屈折ベクトル演算処理が5回必要となる。

10

【0053】

反射レイと屈折レイの計算には、参照点におけるオブジェクト表面のノーマルベクトルと視線ベクトルとの演算とを行い、反射方向と屈折方向を計算し、参照している点から、それぞれの方向にレイを飛ばして、関連のありそうな全部のオブジェクトとの衝突判定を行い、その参照点に最も近いオブジェクトを決定する必要がある。このため、非常に重い処理となる。

【0054】

これに対して、シャドウは、ライトソースとの間にオブジェクトがあるかないかの判定を行うだけでよいので、参照点とライトソースとの間の線分上にオブジェクトがあるかないかの判定を行うことになり、最初に衝突した段階で処理を打ち切ることができるので、反射レイと屈折レイの処理に較べて軽い処理である。

20

【0055】

そこで、図8(B)に示すように、透明なオブジェクトだけを手前から描画するようにすることができる。このようにすれば、重い演算は、反射ベクトルと屈折ベクトルのそれぞれ1回ずつの演算だけで済み、演算量を少なくすることができる。しかしながら、このような処理を行うと、不完全な画像しか得ることができない。

【0056】

そこで、図8(C)に示すように、透明度が予め設定してある所定の基準値より大きい(より透明に近い)オブジェクトに関しては、遠くから順番に描画するが、既に描画された画素の色とブレンドしながら描画を行うようにすることで、正確ではないが、比較的高速に透明感のある描画を行うことができる。

30

【0057】

この透明なオブジェクトの描画処理においては、屈折レイの計算は省略され、シャドウレイと反射レイの計算だけが行われる。屈折レイの計算時には、そのレイと他のオブジェクトとの衝突判定が必要になるため、ブレンディングの計算に較べて非常に重い処理となる。従って、この屈折レイの計算とブレンディングの計算のその分だけ処理を迅速に行うことが可能となる。

【0058】

このように、この例では、第2のレイの計算を中止しておいて、奥にあるオブジェクトから順番に描画しながら、単純にその上に透明なオブジェクトを重ねて、色のブレンディングを行っていく。このような方法で正しく描画ができるのは、透明なオブジェクトが視線に対して垂直なため、屈折光が曲がらない場合であり、入射角が大きい、屈折角が大きいときほど、誤差が大きくなるが、それらの点が小さい場合には、その誤差は無視することができる。

40

【0059】

このようなブレンディングによる方法を用いると、透明なオブジェクトの重なりが多くなるにつれて、図8(A)に示した完全な方法に較べて、重い演算の数が1/2に近付いてくるので、それだけより迅速な処理が可能となる。

【0060】

図5に戻って、ステップS26で、色の合成処理を行った後、ステップS27に進み、DD

50

A3は、ステップS26で合成した色をフレームバッファ7に書き込ませる。このとき、DDA3は、アンチエイリアシング処理を行う。この処理の詳細は、図9のフローチャートに示されている。

【0061】

すなわち、DDA3は、最初に、ステップS51において、フレームバッファ7とカバレッジバッファ23の記憶値(Coverage Buffer)をクリアし、ステップS52において、対象画素のカバレッジ(Coverage)を計算する。例えば、図10(A)に示すように、手前のトライアングルAが対象画素の面積のうち、80%をカバーしているとすると、Coverageは80とされる。

【0062】

次に、ステップS53において、DDA3は、次式を演算する。

$\text{Limit} = 100\% - \text{Coverage Buffer}$

【0063】

いまの場合、ステップS51において、カバレッジバッファの値は0にクリアされているので、Limitは100となる。

【0064】

次に、ステップS54において、ステップS53で計算されたLimitの値が、ステップS52で計算されたCoverageの値より小さいか否かが判定される。いまの場合、Limit = 100であり、Coverage = 80であるため、NOの判定が行われる。そこで、次に、ステップS56において、DDA3は、ステップS52で計算されたCoverageの値に、対象としているトライアングルの画素の色Caを乗算し、その乗算結果Ca × 80%をフレームバッファ7に出力し、値FBとして記憶させる。

【0065】

次に、ステップS57において、カバレッジバッファ23に、ステップS52で計算されたCoverageの値を加算させる。いまの場合、Coverage Bufferの値は、ステップS51で最初にクリアされているので、ステップS52で計算された値80がCoverage Bufferに設定される。

【0066】

次に、ステップS58に進み、その画素に含まれるトライアングルについての演算が全て終了したか否かが判定される。図10の例の場合、図10(B)に示すように、対象画素には、後方のトライアングルBも存在しているため、ここでNOの判定が行われ、ステップS52に戻る。

【0067】

ステップS52においては、対象画素でのトライアングルBのCoverageが計算される。いまの場合、トライアングルBの対象画素に占める面積は40%であるので、Coverageとして40が設定される。

【0068】

次に、ステップS53において、

$\text{Limit} = 100\% - \text{Coverage Buffer}$

が演算される。いまの場合、ステップS57において、Coverage Bufferに80が設定されているため、Limit = 20となる。

【0069】

次に、ステップS54に進み、LimitとCoverageの値が比較される。いまの場合、Limit = 20、Coverage = 40であるため、YESの判定が行われ、ステップS55に進む。ステップS55において、DDA3は、CoverageにLimitの値を設定する。すなわち、いまの場合、Coverageに20が設定される。

【0070】

次に、ステップS56に進み、DDA3は、Coverageに画素の色を乗算する。いまの場合、Coverage = 20%であり、対象画素の色はCbであるから、20% × Cbが演算される。そして、演算された値が、フレームバッファ7の対応する画素の位置に加算される。上述

10

20

30

40

50

した処理により、フレームバッファ 7 には、既に $80\% \times C_a$ が記録されているため、結局、この対象画素の色 C_p は、次式で設定されることになる。

$$C_p = 80\% \times C_a + 20\% \times C_b$$

【 0 0 7 1 】

すなわち、図 1 0 (C) に示すように、トライアングル A とトライアングル B の対象画素において占める面積の割合に応じて、対象画素の色が重み付けされることになる。

【 0 0 7 2 】

次に、ステップ S 5 7 において、Coverage Buffer に、ステップ S 5 2 で演算された Coverage の値が加算される。Coverage Buffer には、既に 8 0 の値が記憶されているため、そこにさらに 2 0 の値が加算され、Coverage Buffer には、1 0 0 が設定される。

10

【 0 0 7 3 】

次に、ステップ S 5 8 に進み、全てのピクセルの重み付け処理を行ったか否かが判定される。いまの場合、全てのトライアングルの重み付け処理が終了したので、処理が終了される。

【 0 0 7 4 】

このような重み付け処理が行われた後、図 5 に戻って、ステップ S 2 8 において、CRTC 8 は、フレームバッファ 7 に記憶されている画像データを読み出し、これを表示信号に変換して、ディスプレイ 9 に出力し、表示させる。

【 0 0 7 5 】

図 1 1 は、さらに他の構成例を表している。この構成例においては、図 4 における Z バッファ 4 が省略されている。その他の構成は、図 4 における場合と同様である。

20

【 0 0 7 6 】

すなわち、この例においては、図 4 における Z バッファ 4 を用いずに補間処理が実行されるようになされている。その他の構成は、図 4 における場合と同様である。

【 0 0 7 7 】

図 1 2 は、図 1 1 の構成例の処理例を表している。そのステップ S 7 1 乃至ステップ S 7 7 の処理は、基本的に、図 5 のステップ S 2 1 乃至ステップ S 2 8 の処理と同様の処理であるが、図 5 のステップ S 2 4 の Z バッファ 4 における判定処理が省略されている。また、図 4 の構成例においては、DDA 3 が、ステップ S 2 3 において、手前のトライアングルから順番に補間処理を行うようにしているが、図 1 1 の構成例においては、DDA 3 が、ステップ S 7 3 において、奥のトライアングルから順番に補間処理を行うようにしている。その他の処理は、図 5 における場合と同様である。

30

【 0 0 7 8 】

このように処理しても、トライアングルの重なりが少ない場合には、第 1 の例による演算処理は減少しているので、全体的に高速に画像データを得ることが可能である。

【 0 0 7 9 】

図 1 3 は、さらに他の構成例を表している。この構成例においては、図 4 におけるトライアングルソート部 2 とソートバッファ 3 が省略された構成とされている。その他の構成は、図 4 における場合と同様である。

【 0 0 8 0 】

40

図 1 4 は、図 1 3 の構成例の処理例を表している。そのステップ S 9 1 乃至ステップ S 9 7 の処理は、基本的に、図 5 のステップ S 2 1 乃至ステップ S 2 8 の処理と同様の処理である。但し、図 5 のステップ S 2 2 のトライアングルの z 座標の代表値を元にトライアングルをソートする処理が省略されている。

【 0 0 8 1 】

すなわち、この例の場合には、ステップ S 9 3 において、Z バッファ 4 の記憶値を用いて、各トライアングルの画素の z 方向の判定処理が行われる。図 1 5 (A) に示すように、最初に、より手前の白の三角形 A がフレームバッファ 7 に既に描画された後、次に、図 1 5 (B) に示すように、白の三角形より奥の黒の三角形 B が描画されるとき、図中白い丸印で示す手前側の三角形の画素については、第 2 のレイの計算処理は行われず、その他

50

の処理は、図 5 における場合と同様である。

【 0 0 8 2 】

なお、図 1 5 (C) は、フレームバッファ 7 上の、最初に手前のトライアングル A を描画した後、次に奥側のトライアングル B を描画している途中の状態を表している。

【 0 0 8 3 】

なお、本明細書において、システムとは、複数の装置により構成される装置全体を表すものとする。

【 0 0 8 4 】

なお、上記したような処理を行うコンピュータプログラムをユーザに提供する提供媒体としては、磁気ディスク、CD-ROM、固体メモリなどの記録媒体の他、ネットワーク、衛星などの通信媒体を利用することができる。

10

【 0 0 8 5 】

【発明の効果】

以上の如く、本発明によれば、3次元座標空間上のオブジェクトの表面の画像をスクリーンに投影することにより生成されたスクリーン座標系の画像データの表面の内部の画素データを補間するようにしたので、より迅速に画像データを生成することが可能となる。

【図面の簡単な説明】

【図 1】本発明の画像処理装置の構成例を示すブロック図である。

【図 2】図 1 の装置の動作を説明するフローチャートである。

【図 3】図 1 の装置の動作を説明する図である。

20

【図 4】本発明の画像処理装置の他の構成例を示すブロック図である。

【図 5】図 4 の装置の動作を説明するフローチャートである。

【図 6】互いに交差するオブジェクトを描画する動作を説明する図である。

【図 7】衝突判定を説明する図である。

【図 8】透明物体を描画する処理を説明する図である。

【図 9】アンチエイリアシング処理を説明するフローチャートである。

【図 1 0】アンチエイリアシング処理を説明する図である。

【図 1 1】本発明の画像処理装置の他の構成例を示すブロック図である。

【図 1 2】図 1 1 の装置の動作を説明するフローチャートである。

【図 1 3】本発明の画像処理装置のさらに他の構成例を示すブロック図である。

30

【図 1 4】図 1 3 の装置の動作を説明するフローチャートである。

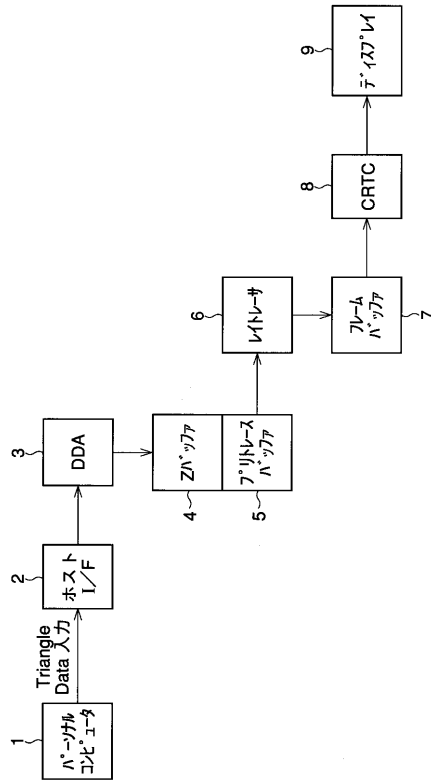
【図 1 5】Z バッファの動作を説明する図である。

【図 1 6】従来のレイトレーシング方式を説明する図である。

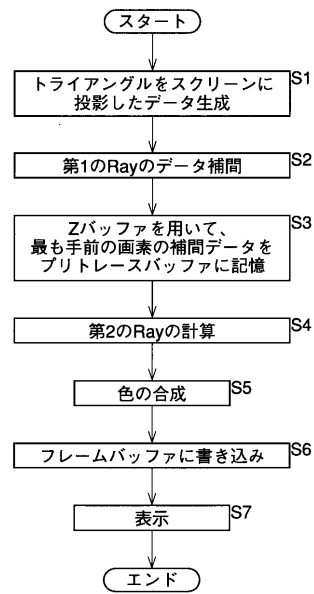
【符号の説明】

1 パーソナルコンピュータ, 3 DDA, 4 Z バッファ, 5 プリトレースバッファ, 6 レイトレーサ, 7 フレームバッファ, 8 CRTコントローラ, 9 ディスプレイ, 21 トライアングルソート部, 22 ソートバッファ, 23 カバレッジバッファ

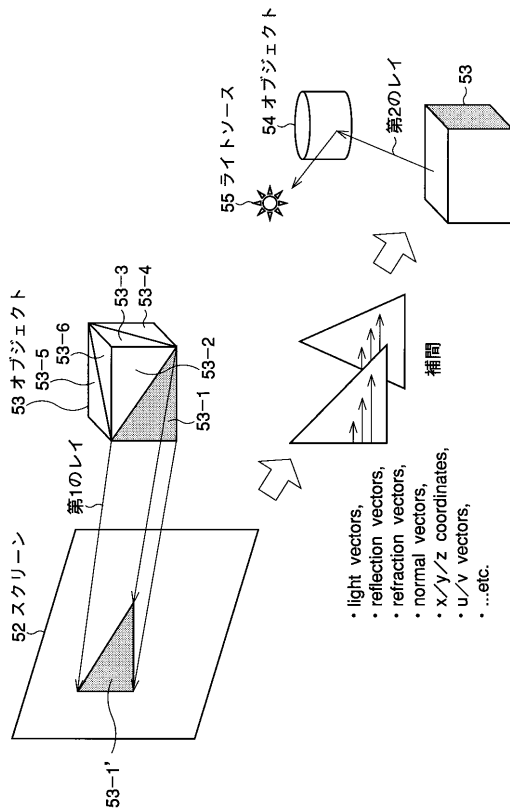
【図 1】



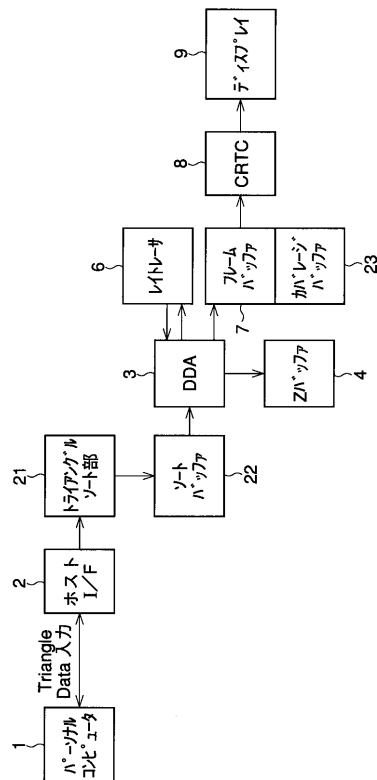
【図 2】



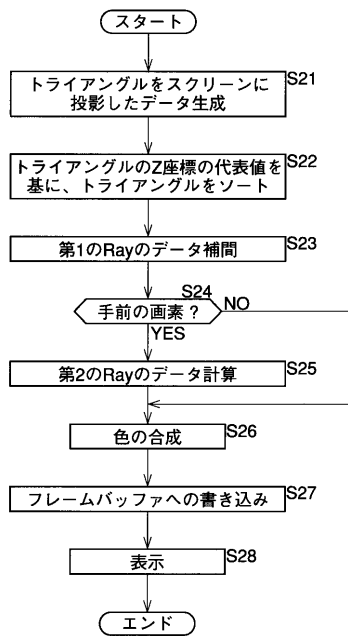
【図 3】



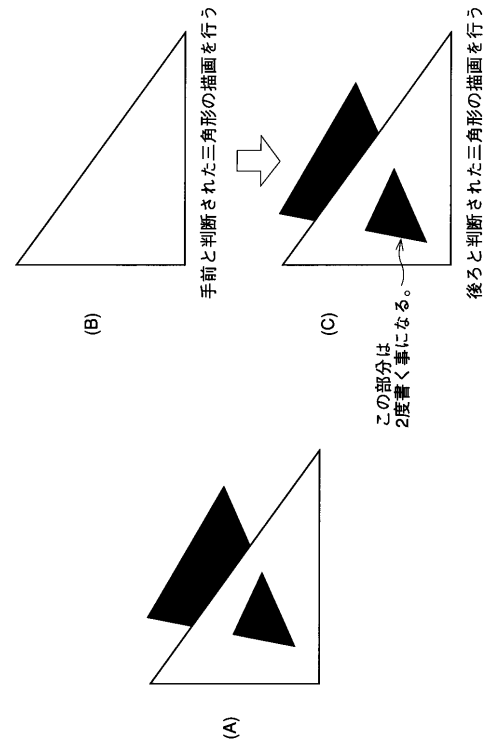
【図 4】



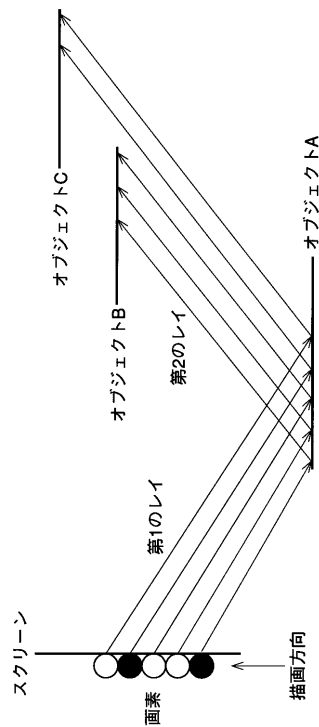
【図 5】



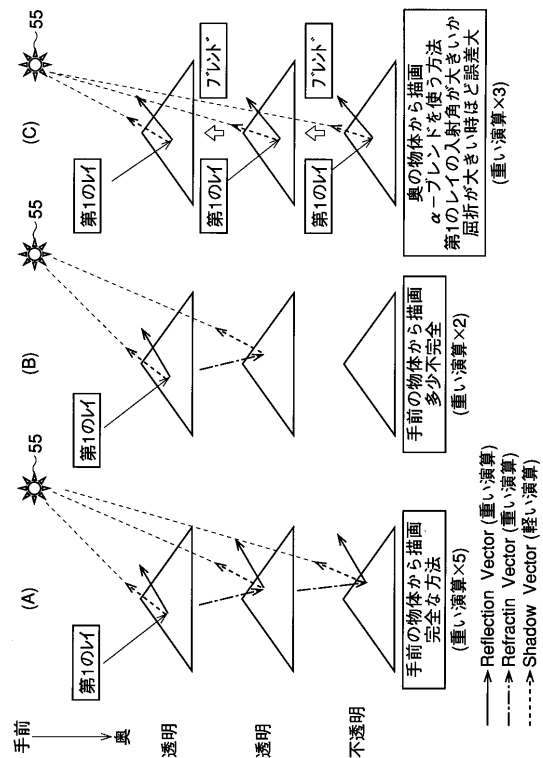
【図 6】



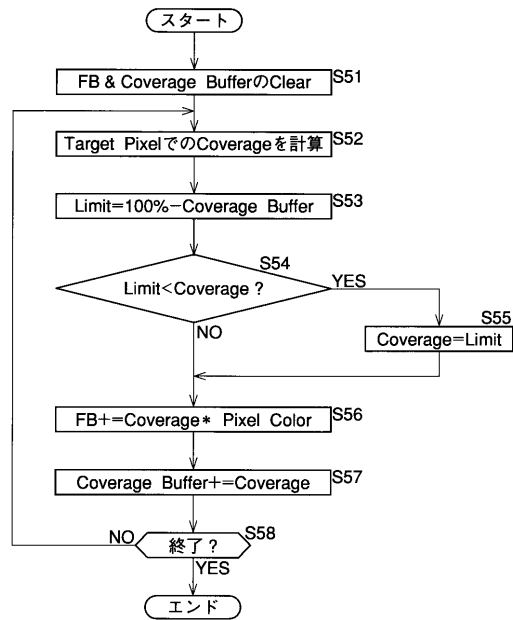
【図 7】



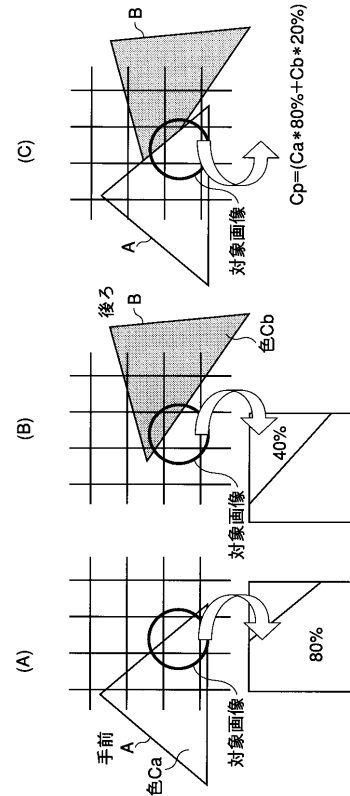
【図 8】



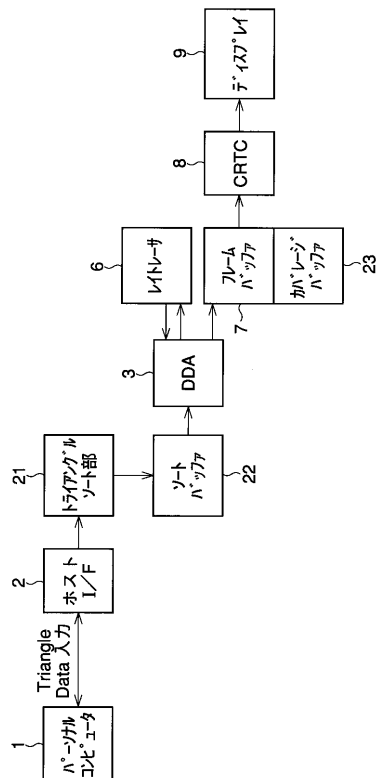
【図 9】



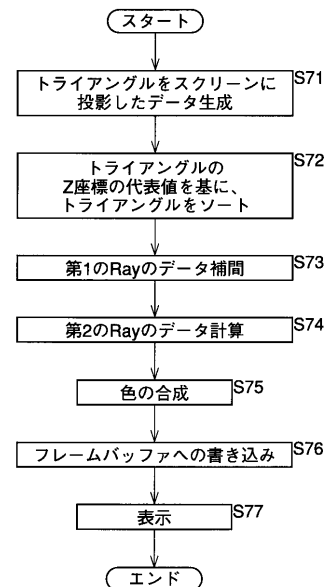
【図 10】



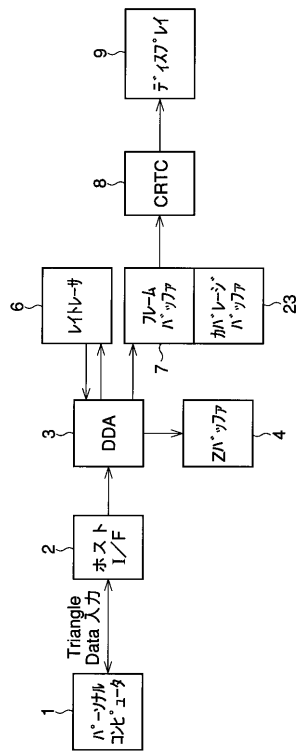
【図 11】



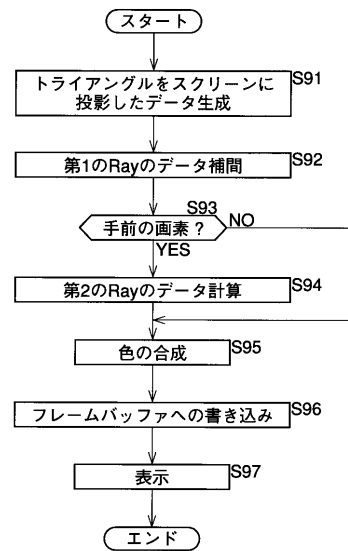
【図 12】



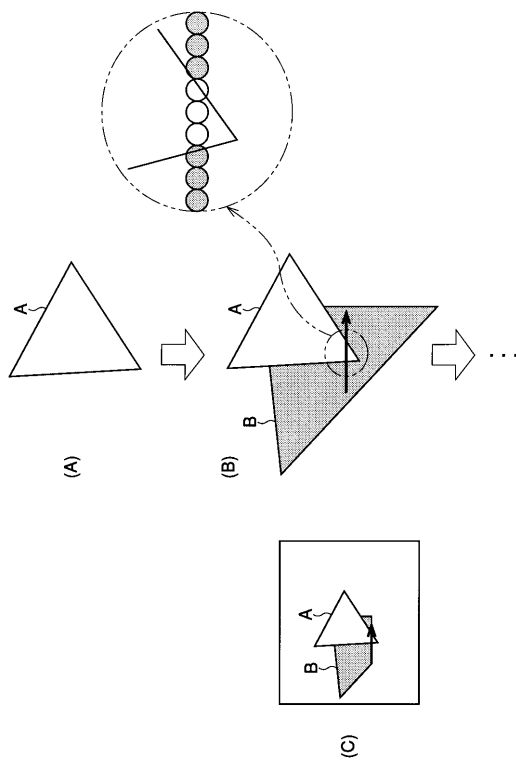
【図13】



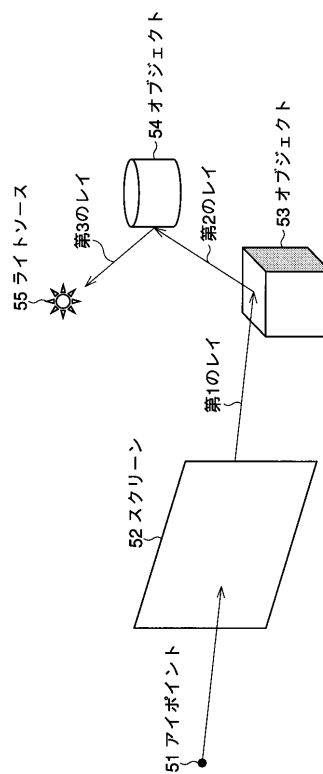
【図14】



【図15】



【図16】



フロントページの続き

(56)参考文献 国際公開第 9 7 / 4 3 7 3 8 (W O , A 1)
特開平 2 - 2 5 3 4 8 5 (J P , A)

(58)調査した分野(Int.Cl. , D B 名)

G06T 15/00 - 17/50

G09G 5/00 - 5/36

A63F 13/00

CSDB(日本国特許庁)