(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4499292号
(P4499292)
(45)発行日　平成22年7月7日(2010.7.7)　(24)登録日　平成22年4月23日(2010.4.23)

(51)Int.Cl.　F I
G06T 15/50 (2006.01)　G06T 15/50 200
G06T 7/60 (2006.01)　G06T 7/60 180Z

請求項の数 4　(全 12 頁)

(21)出願番号　特願2000-581598(P2000-581598)
(86)(22)出願日　平成11年11月8日(1999.11.8)
(65)公表番号　特表2002-529871(P2002-529871A)
(43)公表日　平成14年9月10日(2002.9.10)
(86)国際出願番号　PCT/GB1999/003707
(87)国際公開番号　WO2000/028483
(87)国際公開日　平成12年5月18日(2000.5.18)
審査請求日　平成18年11月2日(2006.11.2)
(31)優先権主張番号　9824414.8
(32)優先日　平成10年11月6日(1998.11.6)
(33)優先権主張国　英国(GB)

(73)特許権者　501176037
イマジネイション　テクノロジーズ　リミテッド
イギリス　ハートフォードシャー　ダブリューディー4　8エルゼット　キングス　ラングリー　ホーム　パーク　エステイト　(番地なし)
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100065189
弁理士　宍戸　嘉一

最終頁に続く

(54)【発明の名称】３次元コンピュータ生成画像のシェーディング

(57)【特許請求の範囲】
【請求項１】
コンピュータが生成した３次元画像をシェーディングする方法であって、
画像内の各オブジェクトを一組のポリゴンとして表すステップと、
画像が表示される画像面を複数の長方形領域に分割するステップと、
各ポリゴンの位置を定義するデータを供給するステップと、
各ポリゴンについて、長方形領域の長方形のバウンディングボックスを定義するステップと、
ポリゴンについて、前記バウンディングボックス内のどの長方形領域が当該ポリゴンと交差しているかを決定するステップと、
各長方形領域の中の各ピクセルについて、交差している各ポリゴンの面の画像面からの距離について深さの値を決め、そのピクセルにおいて面が見えるかどうかを決定し、そして、この決定に基づいてピクセルをシェーディングするステップと、を備え
前記ポリゴンについて、前記バウンディングボックス内のどの長方形領域が当該ポリゴンが交差しているかを決定するステップは、
ポリゴンの各辺について、前記バウンディングボックス内の各長方形領域の角においてテスト点を決める段階と、その際、テスト点の位置は辺の方向と、点の組がポリゴンを定義する時計回り又は反時計回りの順番で決められ、
テスト点が辺の外側にあるかどうかを判定する段階と、
テスト点が辺の外側にある場合には、バウンディングボックスによって定義される長方

形領域の組からその長方形領域を除外する段階とを含み、
ここで、長方形領域が辺の外側である場合には、前記辺の方向と点の組がポリゴンを定義する時計回り又は反時計回りの順番で決められたテスト点の位置は、辺に最も近い長方形領域の角であることを特徴とする３次元画像をシェーディングする方法。
【請求項２】
各長方形領域についてシェーディングされる各オブジェクトが特定されるように、各長方形領域における各ポリゴンについてオブジェクトリストを格納するステップをさらに含んでいる、請求項１に記載の３次元画像をシェーディングする方法。
【請求項３】
コンピュータが生成した３次元画像をシェーディングする装置であって、
画像内の各オブジェクトを一組のポリゴンとして表す手段（２）と、
画像が表示される画像面を複数の長方形領域に分割する手段と、
各ポリゴンの位置を定義するデータを供給する手段（４４）と、
各ポリゴンについて長方形領域の長方形のバウンディングボックスを定義する手段と、
ポリゴンについてバウンディングボックス内のどの長方形領域がポリゴンと交差するかを決定する手段（５０，４６）と、
各長方形領域内の各ピクセルについて、交差する各ポリゴンの面の画像面からの距離に対して深さの値を決定する手段と、
その面がそのピクセルにおいて見えるかどうかを決定する手段と、
この決定の結果に従ってピクセルをシェーディングする手段と、を含み、
前記ポリゴンについてバウンディングボックス内のどの長方形領域がポリゴンと交差するかを決定する手段は、ポリゴンの各辺についてバウンディングボックス内各長方形領域の角においてテスト点を決め、その際にテスト点の位置は辺の方向と点の組がポリゴンを定義する時計回り（ｃｗ）又は反時計回り（ａｃｗ）の順番で決められ、そして、テスト点が辺の外側にあるかどうかを判定し、テスト点が辺の外側にある場合には、バウンディングボックスによって定義される長方形領域の組からその長方形領域を除外し、そして、長方形領域が辺の外側である場合には、前記辺の方向と点の組がポリゴンを定義する時計回り又は反時計回りの順番で決められたテスト点の位置は、辺に最も近い長方形領域の角であることを特徴とする装置。
【請求項４】
各長方形領域についてシェーディングされる各オブジェクトが特定されるように、各長方形領域における各ポリゴンについてオブジェクトリストを格納するステップをさらに含んでいる、請求項３に記載の装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
コンピュータで生成された３次元画像のシェーディング及びこれを行うための方法及び装置に関する。
【０００２】
出願人が有する英国特許第２２８１６８２号では、ある場面の中の見る対象となっている各オブジェクトは、一組の無限大の面として定義される、ポリゴンのための３－Ｄレンダリングシステムについて述べられている。画像が表示されるスクリーンの各要素領域は、ここを通って視点から３次元的な場面へ投射される光線を有している。そして、投射された光線と各面との交点の位置が決定される。これらの交点から、交差した面がその要素領域において見えるかどうかを決定することが可能となる。そして、この要素領域は、決定結果に従ってシェーディングされて表示される。
【０００３】
このシステムは、多数のセルからなるパイプライン型のプロセッサによって実現され、各セルは面との交差の計算を実行することができる。これによって、非常に多くの面の交点を同時に計算することが可能となる。各セルには、交差するかどうかのテストを実行する

面を定義する一組の係数がロードされる。

【０００４】

出願人による英国特許出願第２２９８１１１において説明されている更なる改良では、画像面をさらに複数のサブ領域もしくは複数のタイルに分解している。これは、可変タイルサイト（variable tile site）を用いること、および複雑なオブジェクトの周囲にバウンディングボックス（bounding box）を投影することを提案している。これは、まずタイルの適切な寸法を決定するために、可視のスクリーンの周囲のオブジェクトの分布が決定される。そして、種々のオブジェクトを定義する複数の面が、一つの隣接するリストに格納される。これによって、多くの面から作られているオブジェクトが多くのタイルに入ることができるので、各タイルについて同じ面を格納する必要性を回避できる。そしてこのようなタイルを、前述のレイキャスティング技術を用いて、順番に、そのタイルに含まれる全てのオブジェクトを一度にレンダリングすることができる。これは、特定のタイルにおいて見えないことが分かっているオブジェクトをレンダリングする労力が不要になるため、能率的な方法である。

【０００５】

我々は、実際に見える面の部分に関連するデータだけを処理すれば、合計の処理をさらに削減できることを認識した。すなわち、本発明の好適な実施例によれば、見る方向に対して垂直な平面を有する目に見える面の複数の辺を定義するための方法が提供される。

【０００６】

本発明の第二の態様によれば、我々は、サイズを変えられるタイルを用いるのではなく、画像平面の全体にわたって一定寸法のタイルを用いることによって、処理を最適化できることを認識した。この場合、タイルの境界はオブジェクトと交差するかもしれないが、辺での切り落とし（edge clipping）は必要ない。すると、特定のオブジェクトに対してバウンディングボックスを定義する一組のタイルを選択することができ、そしてそのオブジェクトをレンダリングするためには、その特定のバウンティングボックスの中のタイルだけを処理すればよくなる。タイルの中に入っている面の表示リストは、バウンディングボックスの中にあるオブジェクトを定義するのに用いられる。

【０００７】

この方法におけるさらなる改良は、実際にレンダリングすべきオブジェクトを含まないバウンディングボックスの中のタイルを排除することである。

【０００８】

ここから本発明の好適な実施例について、添付図面を参照しながら詳細に説明する。

【０００９】

この明細書の導入部で要約して述べたレンダリングシステムが、出願人の有する英国特許第２２８１６８２号において説明されている。我々は、三角形の組み合わせとして任意のオブジェクトをモデル化できることを認識した。前記の特許において処理されるのは、無限大の面である。前記の特許では、オブジェクトの辺は、複数の無限大の面の交線と、特定の面が見えるかどうかを決定するための前方のフェーシング面と後方のフェーシング面（後方のフェーシング面が前方のフェーシング面よりも近ければ、特定のピクセルにおいてはいずれも見えない）の相対的な深さとからる。

【００１０】

我々は、視点に対して垂直な無限大の面によって三角形の辺を定義することによって、処理を改善できることを認識した。すなわち、三角形の場合は、フェース面に対して一つと、各辺に一つずつの三つ、合計四つの面が必要となる。

【００１１】

三角形をレンダリングする前に、各面について方程式を計算する必要がある。これらは、ポリゴンのセットアップユニットにおいて、アプリケーションソフトウェアによって提供される頂点のデータから計算される。図１に示した（x_1，y_1，z_1）及び（x_2，y_2，z_2）に位置する二つの頂点v_1及びv_2の間の垂直な辺の面についての方程式は、

$$(y_2-y_1)x+(x_1-x_2)y+(x_2y_1-y_2x_1)=0$$

によって定義され、これは、
Ａｘ＋Ｂｙ＋Ｃ＝０
という平面を表す方程式の形になっている。
【００１２】
特定のｘｙの値（ピクセル位置）に対してこの方程式の答えが正のときは、ｘｙの位置はエッジ面の前方のフェーシング面上にあり、値が負のときはｘｙ位置はこの面の方向のフェーシング面上にある。したがって、図１の三角形を表す四つの方程式すべてが正の値を有するときは、ピクセルの位置は図２に示すように三角形の内側に存在する。この規則は、三角形に優先して四辺形など、使用する任意の形状に対して当てはまる。
【００１３】
本発明の好適な実施例を図３に示す。ここには、三角形を定義する頂点データを受け取り、３２個の面プロセッサ４の組のそれぞれに対して三角形のフェーシング面データを供給するポリゴンセットアップユニット２が設けられている。これは同時に、面プロセッサ４によって処理される各三角形について、３組の辺データを３列の辺プロセッサ６の一つずつに供給する。これらにはそれぞれ、３２個の特定のピクセル位置のそれぞれについて処理しているエッジ面に対する値が正か負かを決定する深さ評価ユニットが含まれている。これらのそれぞれの出力は正又は負の符号ビットであり、三つの面についてのこれらの符号ビットは、その三角形についての適当な面プロセッサに供給される。前述のようにすべての符号ビットが正ならば、その面プロセッサは処理している三角形の面が見えるものであること、すなわちそれが三角形の辺の外側ではないことを知り、したがって出力として深さ値を与え、これは深さ記憶装置へ行き、その後さらに、これについて、処理されている画像への寄与に用いられるかどうかを決定するたのテストを行うことができる。符号ビットの一つが負のときは、面プロセッサ４は何もする必要がない。
【００１４】
辺プロセッサはｘ方向で、すなわち、画像内のスキャンラインに沿って動作し、そして、３２個の面プロセッサ４の配列を用いるシステムにおいて、通常は３２×３２ピクセルのブロックを処理するタイルベースのシステム内で動作する。したがって、各辺プロセッサへの入力値は、Ｂｙ＋Ｃに等しくなる。辺プロセッサは、三角形の辺上で動作する精密でないストアされていない割り算アルゴリズムを用いる。このアルゴリズムは、実際には
ｘ＝Ｃ／Ａ
という計算を行う。これが可能な理由は、ｙの値が特定のｘについて値定数であり、したがってＢｙ＋Ｃが特定のスキャンラインに沿って定数だからである。
【００１５】
表１は、辺の内側から外側へ（正から負の深さへ）の転換点の位置を計算する場合に含まれる算術演算を示している。
【００１６】
【表１】

A	C	演算
Aを左に4だけシフト(16×A)	C	加算
Aを左に3だけシフト(18×A)	$16A + C = C_1$	C_1>＝0かつA>＝0のときは減算、それ以外は加算
Aを左に2だけシフト(4×A)	$C_1 \pm 8A = C_2$	C_2>＝0かつA＝0のときは減算、それ以外は加算
Aを左に1だけシフト(2×A)	$C_2 \pm 4A = C_3$	C_3>＝0かつA>＝0のときは減算、それ以外は加算
A	$C_3 \pm 2A = C_4$	C_4>＝0かつA>＝0のときは減算、それ以外は加算
A	$C_4 - A = C_5$	C_5>＝0かつA>＝0のときは減算、それ以外はC_5
	$C_5 (-A) = C_6$	

表１－辺プロセッサ内の不正確な非格納の分割

【００１７】

ステージ１Ａで実行される演算は、サンプル点をｘについて中央に有効に移動させる。これが可能な理由は、セットアップユニットが原点の位置（ｘ，ｙ）＝（０，０）をタイルの右上の角へ移動させるからである。演算の列は、次のクロックサイクルで積算されたＣの値について加算又は減算を行うべきかどうかを計算するために実行されるテストを示している。これらのテストは、本質的に、各加算／減算が我々をゼロ交差点（zero crossing point）に近づけるバイナリーサーチの形態である。例えば、ゼロの遷移が１３であるとしよう。

【００１８】

ｘ位置

スタート　Ｃ＝－ｖｅ　Ａ＋　＋ｖｅ　　０
加算１６　Ｃ＝＋ｖｅ　　１６
減算８Ａ　Ｃ＝－ｖｅ　　８
加算４Ａ　Ｃ＝－ｖｅ　　１２
加算２Ａ　Ｃ＝＋ｖｅ　　１４
減算Ａ　　Ｃ＝０（＋ｖｅ）　　１３
減算Ａ　　１２

【００１９】

辺プロセッサによって実行される加算／減算の符号は、遷移点又は辺を計算するのに用いられる。一旦このピクセル位置が決定されると、これをタイルのライン全体のためのマスクを生成するために用いることができる。このマスクは、線の中の各ピクセルについての正／負の深さの値を表す。この演算は、タイル内のピクセルの線のための辺マスクをクロックサイクルごとに生成することができるように、上で述べた深さプロセッサのアレーを用いてパイプライン処理することができる。上で説明したように、辺方程式のｙ係数は、辺が処理される前に定数Ｃに蓄積される。これにより３２×３２個のピクセルからなる完全なタイルの辺マスクを３２クロックサイクルにわたって生成することができる。ここでｈはタイルの高さである。

【００２０】

三角形の三つの辺すべてのマスクは、ＡＮＤ演算が行われてこの三角形の深さマスクが生成される。ピクセル位置における蓄積された深さの符号は、面プロセッサ４に渡される。深さが正のときは、その面は見える。したがって、この方法を用いると、三角形を単一の面と同じ速度で処理することができる。四つ又はそれ以上の辺プロセッサが利用可能な場

合には、四辺形やこれよりもっと複雑な形状を処理できることは明らかである。

【００２１】

画像のスクリーンが複数のタイルに分割されるときは、現在のハードウェア手段は、各タイルについて処理されるスクリーンの中のすべてのオブジェクトを必要とする。これは、すべてのタイルがすべてのオブジェクトについて処理されなければならないので、非能率的である。

【００２２】

従来からのレンダリングシステムでは、タイルを基礎とするタイル上のスクリーンのレンダリングは、オブジェクトがタイルの境界にクリップされる必要があり、したがってタイルの境界との交点を定義するデータが定義されなければならない（図４参照）。

【００２３】

特定の領域と交差するオブジェクトを処理することだけが必要である。上で説明したように、オブジェクトがスクリーン空間で定義されている場合は、三角形などのオブジェクトを定義する頂点の比較が、そのオブジェクトに対するバウンディングボックスをもたらす。バウンディングボックスは、オブジェクトを含むスクリーン内の長方形領域を定義する。図４は、タイルで覆われたスクリーンの領域を、内側の多数の三角形で表したオブジェクトとともに示している。特定のオブジェクトのためのバウンディングボックスを、バウンディングボックス内のタイルのリストが得られるように、タイルの境界に並べることができる。このタイルのリストは、スクリーン内のすべてのタイルの部分集合であり、オブジェクトと交差するタイルを近似したものとなる。オブジェクトを有するバウンディングボックスがスクリーン領域全体と交差する場合には、オブジェクトのパラメータ（座標、シェーディングデータなど）は、システム内のメモリの領域に書き込まれ、オブジェクトデータの開始を指し示すポインタが生成される。

【００２４】

このレンダリングシステムは、タイルを基礎としてタイルについて動作し、次のものに進む前に各タイルについてオブジェクトを処理する。したがってデータ構造は、各タイルについて処理されなければならないオブジェクトを特定するのに用いられる。これは図５に示されている。ここでは、スクリーン内のタイルのリストは、領域又はタイルのアレー３０において生成される。各タイルは、ｘ及びｙの限界によって定義される。各タイルについて、そのタイルについて処理されなければならないオブジェクトへのポインタのリストが、オブジェクトリスト３２として生成される。領域アレーによって指し示されている各タイルについては、別のオブジェクトリストが存在する。バウンディングボックスの考え方は、オブジェクトポインタが加算されなければならないタイルのリスト（オブジェクトリストを伴う）を生成するのに用いられ、このオブジェクトポインタはデータがメモリに書き込まれたときに生成される。しかしながら、ハードウェアは、書き込まれるオブジェクトポインタに対するアドレスが得られるように、各オブジェクトリストの末尾を特定しなければならない。これを行う最も単純な方法は、リスト上の次の自由な位置を指し示す末尾ポインタを格納することである。これはオブジェクトリストのヘッダーとなりうる。

【００２５】

これは、より小さい寸法のキャッシュを用いるとさらに改善される。キャッシュは末尾ポインタの部分集合を格納する。オブジェクトは一般に複数のタイルを横切るので、キャッシュ上の書き落としによって、複数の末尾ポインタが読み込まれ、オブジェクトが横切るタイルを予測することになる。これはキャッシュの効率を上げる。これはまた、オブジェクトデータをインターリーブし、キャッシュの内容を変更することによって、複数の画像を同時にタイルで覆うことを可能とする。この切り換えには、キャッシュにリンクされたメモリ領域及びオブジェクトの格納に用いられるメモリ領域を調整して、末尾ポインタキャッシュの内容を格納することが含まれる。これでコンテキストの効果は変わる。つまり、キャッシュが無効とされ、異なるデータの組が、タイルで覆うのに利用可能な状態となる。コンテキストを元の状態に切り換えるのは逆の演算であり、新しいコンテキストの格納、キャッシュ及びオブジェクトのメモリ位置の逆転、そして現在のキャッシュを無効に

する動作を含む。
【００２６】
これでオブジェクトリストのための情報が利用可能となる。末尾ポインタキャッシュに由来するポインタのためのアドレス及びそのタイルと交差するバウンディングボックスを有するオブジェクトを指し示すオブジェクトポインタである。そして、処理されているオブジェクトのために登録されているすべてのオブジェクトリストを、メモリ及び処理すべき次のオブジェクトに書き込むことができる。
【００２７】
これは、図１０の回路を用いて実行することができる。ここでは、オブジェクトデータは、アプリケーションプログラムから三角形、扇、細片、そして点という形態で受け取られる。初めに、オブジェクトデータは、変換ユニット４０においてすべて細片に変換される。これらはメモリを使う上で効率的である。変換部４０は、扇及び面を細片に変換する変換部４２と、点及び線を細片に変換する変換部４４とを含んでいる。そして細片データは、バウンディングボックス生成部４６へ供給される。これは、細片内の各三角形についてバウンディングボックス及びすべての細片についてのバウンディングボックスを計算する。バウンディングボックスがスクリーン領域と交差する場合は、オブジェクトデータはローカルリード／ライトアービタ４８を介してメモリに書き込まれる。そうでない場合は、システムは次の細片へ進む。このデータが書き込まれるアドレスは、パイプラインの下流側へ渡される。
【００２８】
領域生成部５０は、バウンディングボックスの情報を受け取り、マスクと、すべての細片についてのバウンディングボックス内の各タイルについてのタイルアイデンティティを生成する。タイルアイデンティティは、末尾ポインタキャッシュ５２にアクセスして次の利用可能なポインタ位置を読み出すのに使われる。もしもこれがそのブロック内の最後のアドレスであると、このタイルに対して新たなポインタブロックが割り当てられ、現在のブロックから新しいブロックへのリンクが生成される。
【００２９】
オブジェクトのアドレスを有するポインタへのフリーのアドレスへの書き込み要求、およびそのオブジェクトについてのマスクは、待ち行列に置かれる。そしてタイルについての末尾ポインタは、キャッシュを通して次に利用可能なポインタによって更新される。書き込みの待ち行列に１６の要素があるときは、要求はページアドレスによって、ポインタソータ５４によってソートされる。これらは第一アクセスのメモリへ書き込まれる。これによって、メモリへのページブレークの数が削減される。
【００３０】
最も普通の安価なブロックＲＡＭは、ＤＲＡＭである。これはページと、ページを横断するアクセスによって構成されている。これは、現在のページを閉じて新たなページを開くことについてパフォーマンスコストがあるからである。しかしながら、同じオブジェクトへのポインタを複数のリストに書き込むと、各リストは異なるページにある場合もあるので、非常に多くのページの移り変わりが必要となる。しかしながら、入ってくる一つのオブジェクトと次のオブジェクトとの間には類似性がある可能性が高い。これは、次のオブジェクトが、現在及びその前のオブジェクト同じオブジェクトリストに置かれる可能性が高いことを意味する。オブジェクトリストによってアドレスは本質的に連続的となり、したがって、ポインタ同士の間のアドレスの緊密さがあるのと同じだけ多くのポインタを同時に同じリスト内に書き込むのが望ましく、これは、多くのポインタ（例えばオブジェクトの範囲を超える）を格納し、これらをメモリへ書き込む前にページグループ内に格納することによって達成される。これにより、ページの移り変わりは大幅に削減され、したがってシステムの効率が向上する。
【００３１】
画像のオブジェクトの所与のデータセットについて、各タイル内で見えるオブジェクトの数を決定することは可能ではない。最悪のシナリオは、ポインタのための十分なメモリを

、すべてのタイルに対するすべてのオブジェクトに割り当てなければならない場合である。これは非常に多くのメモリを必要とし、システムのコストを押し上げる。これは、各オブジェクトリストに対してブロックを割り当て、そしてブロックがいっぱいになったときは、新たなブロックを割り当てるとともにこの新たなブロックへのリンクを挿入することによって、削減することが可能である。これは、使っているメモリが、オブジェクトリストの格納のために必要な最少量に近づくことを意味する。ブロックの大きさは、メモリの幅及び利用可能なバンド幅など多くのファクターに依存する。

【００３２】

オブジェクトポインタの数及びオブジェクトデータのサイズをさらに削減するために、別の観点のオブジェクトの緊密性を用いることができる。一般に一群の三角形は、ティーポット、球、あるいは動物などの大きなオブジェクトを表すのに用いられるので、三角形の間には非常に多くの共通性が存在する。すなわち、三角形は、それらの間で頂点を共有する。頂点を互いに比較することによって、三角形を細片に変換することが可能である。新しい三角形を定義するのに必要な頂点は一つ又は二つだけであり、そして細片内のすべてのオブジェクトを指し示すのに必要なポインタは一つだけなので、細片が必要とするメモリの領域は少ない。これによって、オブジェクトポインタの数はさらに削減され、必要とするメモリの量も少なくなり、これによりメモリに関する効率が向上し、バンド幅の効率化によって性能が向上する。図６に、三角形とバウンディングボックスを示してあり、後者には陰影を付してある。これを通常の方法を用いて処理すると、この中の領域は５×５列のタイルをカバーし、この処理を２５回行う必要がある。しかしながら、画像をまず、三角形が用いるｘ，ｙ座標の範囲を保持する領域を定義するのにバウンディングボックスを用いて処理すると、三角形は１２回だけ処理すればよいこと、すなわち、三角形は１２個のタイルをカバーしていることが示される。

【００３３】

我々はさらに、三角形は実際には、４×３個の中のタイルの１０個だけの中にあることを認識した。これによって更に処理のオーバーヘッドは削減される。

【００３４】

三角形を処理するのに必要な長方形のバウンディングボックスの全体をカバーしていない別の三角形の例を、図７のａ～ｄに示す。このうち最も極端な例は図７ｄであり、ここでは三角形は図８に示した１２個のタイルの中に収まる。この三角形をレンダリングするには、このタイルの組だけを処理するのが望ましい。

【００３５】

三角形を表すタイルの最少の組の計算は、粗い長方形のバウンディングボックスの全体の計算から始める。もしバウンディングボックスが高さにおいても幅においても単一のタイルであれば、実行すべきさらなる最適化は存在しない。そうでなければ、その三角形の各辺を考慮することによって、このタイルの組は削減される。

【００３６】

最初に、その三角形が時計回り（ｃｗ）の点の組によって定義されるのか、それとも反時計回り（ａｃｗ）の点の組によって定義されるのかを知らなければならない。この情報が分からないときは、簡単に計算することができる。

【００３７】

次に、辺を、空間を半分ずつ二つに分割する無限に長い線と見なすことができる。辺の両側のサブスペースは、上で述べた辺プロセッサによって辺の内側か外側として記述することができる。ここで内側のサブスペースとは、辺が属している三角形を含む方である。三角形はその頂角を、辺の線の交点として有しており、面は三つの辺の内側のサブスペースの重なり部分である。

【００３８】

その全体が辺の外側にある任意のタイルは、その中に三角形が見えないので、そのタイルは最少の組の一部ではない。辺が完全に水平又は垂直の場合には、長方形のバウンディングボックスの中のすべてのタイルは既に全体又は一部が辺の内側にあるので、これを考慮

する必要はない。
【００３９】
タイル全体が辺の外側にあるかどうかをテストするためには、その辺に最も近いタイルの頂点上の点だけをテストすればよい。もしこの点が辺の外側にあれば、タイル全体も辺の外側にあると確信できる。このテスト点の位置は、図９に与えた表に示すように、辺の方向によって決定される。
【００４０】
辺それ自身は、
ｙ＝ｍｘ＋ｃ
という方程式を用いて記述することができる。ここで、ｘ及びｙは、スクリーンの座標であり、ｍは直線の傾きを表し、ｃは定数である。タイルの頂点におけるｍｘ＋ｃの値を評価することによって、その点のｙ座標よりも大きいか、小さいか、あるいは等しい値を与える。この二つの値の比較によって、その点が辺の内側にあるか外側にあるかが示される。この結果の解釈は、図９の表に与えるように、辺の方向に依存する。
【００４１】
タイルが最少の組から排除されるかどうかを決定するには、三角形の各辺について、長方形のバウンディングボックスの中の各タイルをこのようにして処理しなければならない。
【００４２】
タイルの頂点におけるテスト点は、また、より大きな長方形のタイルの組についてのテスト点でもあるという点に注意すべきである。図１０において、テストポイントが辺の外側である所にタイルがマークされたことを知ることは、陰影を付けられたすべてのタイルは、同じく辺の外側になければならないことを意味する。テストポイントが右下にあるこの例では、長方形のバウンディングボックスのタイルを、右から左へ、下から上へ処理するのがもっとも効率的であり、合計では、最少の数のテストを有する最少セットから排除されるタイルが非常に多くある。テストポイントがタイルの別の角にあるときは、それに従って処理の順序が変更される。
【図面の簡単な説明】
【図１】　オブジェクトの一部を描くのに用いる三角形の面のグラフィック表示を示す図である。
【図２】　図１の三角形の見える部分を決定するために、面の正の側と負の側をどのように用いるかを示した図である。
【図３】　図２の三角形をシェーディングするのに用いる辺プロセッサ及び面プロセッサをブロック図として図である。
【図４】　２５のタイル列に分割されたスクリーンにオブジェクトが描かれ、オブジェクトが通常の方法で三角形に分解された様子を示している。
【図５】　本発明の一実施例に従って用いられるオブジェクトリストを示した図である。
【図６】　１２のタイルのバウンディングボックスの中の三角形を示した図である。
【図７】　ａ，ｂ，ｃ，ｄは、異なるバウンディングボックスを有する多くの異なる三角形と、これらを表示するのに必要とされる異なる数のタイルを示した図である。
【図８】　図７のｄにおける三角形のためのタイルの最適な選択を示した図である。
【図９】　三角形を表示するのに必要ない図８のタイルを決定するのに用いるテストを例示した表を示した図である。
【図１０】　長方形のタイルの組を、テストポイントと共に示した図である。
【図１１】　バウンディングボックスを生成するのに用いる回路のブロック図である。

【図１】

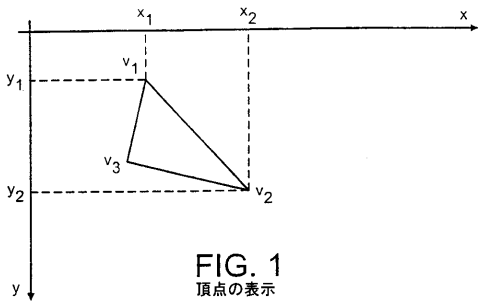

FIG. 1
頂点の表示

【図２】

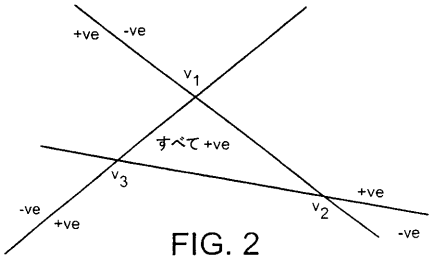

FIG. 2

【図３】

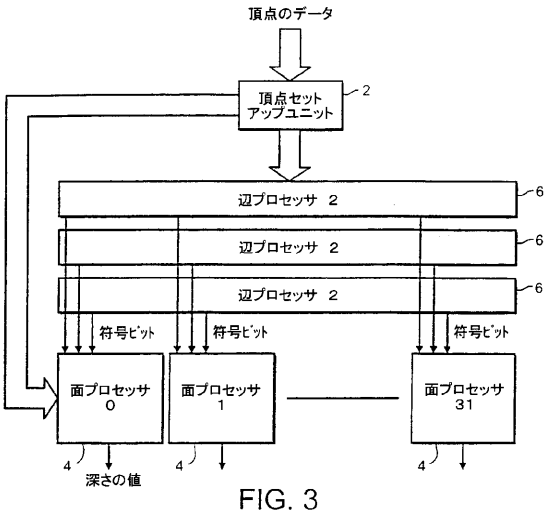

FIG. 3

【図４】

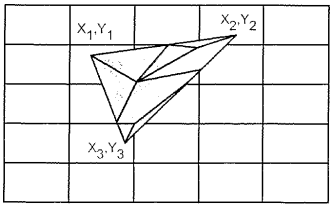

FIG. 4

【図５】

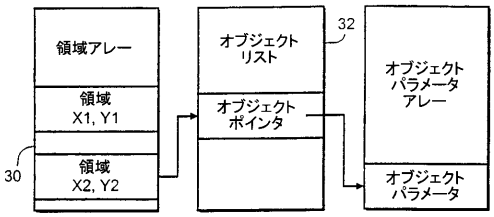

FIG. 5

【図６】

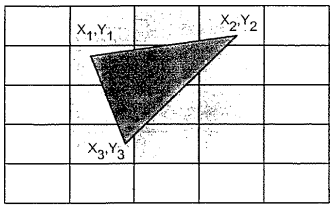

FIG. 6

【図７】

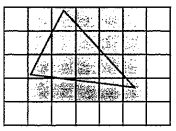

FIG. 7a

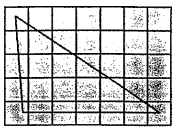

FIG. 7b

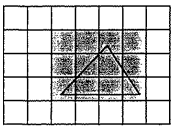

FIG. 7c

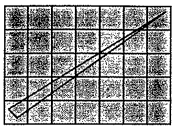

FIG. 7d

【図８】

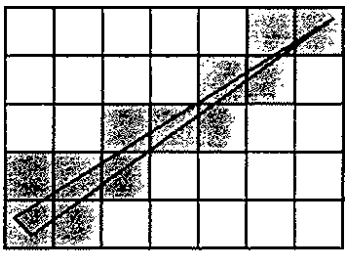

FIG. 8

【図１０】

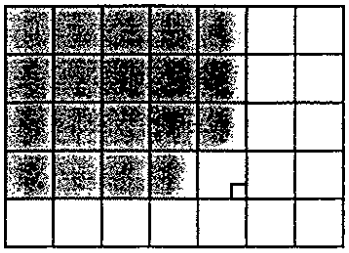

FIG. 10

【図９】

表 1

テスト点	三角形	辺の方向	辺の外側のテスト点の条件
	CW		y < mx + c
	CW		y < mx + c
	CW		y > mx + c
	CW		y > mx + c
	ACW		y > mx + c
	ACW		y > mx + c
	ACW		y < mx + c
	ACW		y < mx + c

FIG. 9

【図１１】

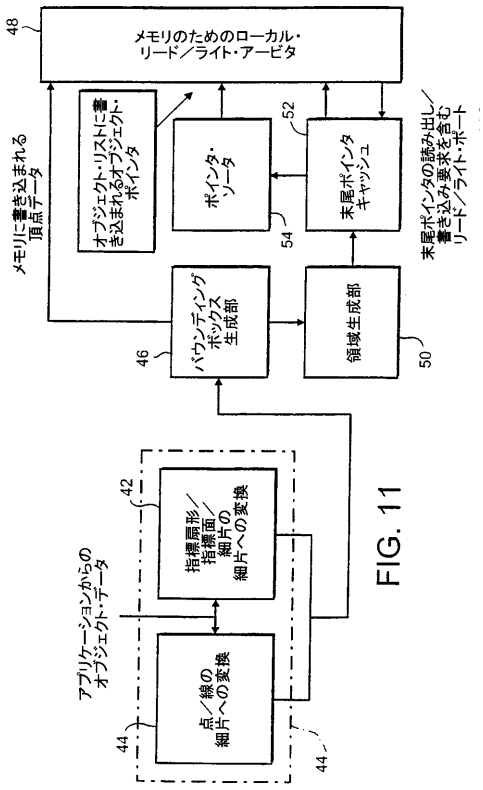

FIG. 11

フロントページの続き

(74)代理人 100096194
弁理士 竹内 英人
(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100084009
弁理士 小川 信夫
(74)代理人 100082821
弁理士 村社 厚夫
(74)代理人 100086771
弁理士 西島 孝喜
(74)代理人 100084663
弁理士 箱田 篤
(72)発明者 レッドシャウ ジョナサン
イギリス ヒーメル ヘンプステッド エイチピー１ １エックスディー カーディー ロード ボックスモア ザ フェイシングス ８
(72)発明者 モーフェット スティーブ
イギリス ヒーメル ヘンプステッド エイチピー３ ９エスエイチ アプスリー ウェイマウス ストリート ８０エイ

審査官 伊知地 和之

(56)参考文献 特開昭６４－０５４５８１（ＪＰ，Ａ）
特開平０４－２２０７８２（ＪＰ，Ａ）
特開平１０－０５４５８１（ＪＰ，Ａ）
米国特許第５５０９１１０（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06T 15/00 - 17/50