(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4035521号

(P4035521)

(45)発行日　平成20年1月23日(2008.1.23)　(24)登録日　平成19年11月2日(2007.11.2)

(51)Int.Cl.		F I		
G06T 11/00	(2006.01)	G06T	11/00	100A
G06T 11/40	(2006.01)	G06T	11/40	200
H04N 1/46	(2006.01)	H04N	1/46	C
H04N 1/60	(2006.01)	H04N	1/40	D

請求項の数 4　外国語出願　(全 100 頁)

(21)出願番号　特願2004-190406 (P2004-190406)
(22)出願日　平成16年6月28日 (2004.6.28)
(65)公開番号　特開2005-44346 (P2005-44346A)
(43)公開日　平成17年2月17日 (2005.2.17)
審査請求日　平成16年6月28日 (2004.6.28)
(31)優先権主張番号　2003903445
(32)優先日　平成15年6月26日 (2003.6.26)
(33)優先権主張国　オーストラリア(AU)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二

最終頁に続く

(54)【発明の名称】画素のランのための合成計算の最適化

(57)【特許請求の範囲】

【請求項１】

画像を構成する各層を合成する画素生成方法であって、

前記各層をＺ値でソートするソート工程と、

前記ソート工程でソートされた各層のうち、可変アルファ塗り潰しの層より上方の塗り潰しの層に対する貢献度を計算する計算工程と、

前記ソート工程でソートされた各層の中に、可変アルファ塗り潰しの層が含まれているか否かを判定する判定工程と、

前記判定工程で可変アルファ塗り潰しの層が含まれていると判定された場合、可変アルファ塗り潰しの層を含むその下方の塗り潰しの層をボトムアップで合成し、ボトムアップで合成された塗り潰しの層に対する貢献度を計算し、

可変アルファの塗り潰しの層より上方の塗り潰しの層を、対応する貢献度に基づいた計算値が閾値に達するまでトップダウンで合成することで、出力画素を生成する生成工程と

を有することを特徴とする画素生成方法。

【請求項２】

画像を構成する各層を合成する画素生成装置であって、

前記各層をＺ値でソートするソート手段と、

前記ソート手段によってソートされた各層のうち、可変アルファ塗り潰しの層より上方の塗り潰しの層に対する貢献度を計算する計算手段と、

前記ソート手段によってソートされた各層の中に、可変アルファ塗り潰しの層が含まれ

ているか否かを判定する判定手段と、
前記判定手段によって可変アルファ塗り潰しの層が含まれていると判定された場合、可変アルファ塗り潰しの層を含むその下方の塗り潰しの層をボトムアップで合成し、ボトムアップで合成された塗り潰しの層に対する貢献度を計算し、
可変アルファの塗り潰しの層より上方の塗り潰しの層を、対応する貢献度に基づいた計算値が閾値に達するまでトップダウンで合成することで、出力画素を生成する生成工手段と
を有することを特徴とする画素生成装置。
【請求項３】
コンピュータに請求項１に記載の画像生成方法を実行させるためのプログラム。
【請求項４】
請求項３に記載のプログラムを格納した、コンピュータ読み取り可能な記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
[著作権表示]
本特許明細書は、著作権保護を受ける資料を含む。著作権者は、審査を目的として関連する特許庁のファイルから本特許明細書又は関係資料を複製することに異議はないが、それ以外の場合については全ての著作権を保持する。
本発明は、一般的に、走査線レンダリングに関し、特に、画素のランに対して行なわれる合成動作の数を最適化（好ましくは最小化）することを目的としたグラフィカルオブジェクト合成の手法に関する。
【背景技術】
【０００２】
コンピュータグラフィックにおける合成は、２枚以上の画像を組み合わせて結果画像を形成することを意味する。最も単純な形態では、合成は不透明な前景を不透明な背景に重ねることを意味する。前景画像が背景画像を覆い隠す箇所では必ず前景色が対応する背景色に取って代わるが、それ以外の箇所では背景色はそのまま変化しない。このとき得られた画像が新規の背景画像となる。同様にして、第３の不透明な画像を新規の背景画像に重ねることができる。このプロセスは、任意の枚数の画像に対して繰り返すことができ、広く普及しているＰａｉｎｔｅｒ’ｓ　ａｌｇｏｒｉｔｈｍ（画家のアルゴリズム）を効果的に説明するものである。
【０００３】
２枚の画像が重なる領域では、前景と背景の何らかの組み合わせの出現が要求される場合、状況は更に複雑になる。前景色と背景色とが適切にブレンドされる必要があるからである。T. Porter及びT. Duffによる論文「Compositing Digital Images」（Computer Graphics, Vol. 18, No. 3 ACM 1984）では、合成代数の１２個の可能な動作を体系化している。合成プロセスは、Ｐｏｒｔｅｒ－Ｄｕｆｆ演算子を用いて前景画像と背景画像とを組み合わせて新規の背景画像を形成し、このプロセスを新規の前景画像の枚数分繰り返すことへと一般化することができる。
【０００４】
この「back to front」技術では正しい結果が得られるが、場合によっては非常に計算コストがかさむ。画像ディスプレイなどの出力装置は、通常、個別の画素から構成されるものと考えられる。従来技術では、合成動作は画素ごとに行なわれなければならない。多くのコンピュータシステムにおいて、画像フレームバッファへの書き込みはとりわけ低速であることが多く、同じ画素に複数回書き込みを行なうシステムでは厳しい性能上のペナルティに直面することになる。また、画素が何度も書き込まれるためにディスプレイ上で見苦しいちらつきを発生させる可能性がある。画像はオフスクリーンバッファで生成されても良いが、オフスクリーンバッファに割り当てるための追加のメモリが必要であり、オフスクリーンバッファからフレームバッファへと転送するのに余計な時間がかかる。

【０００５】
また、現在の適用例で合成される画像は複雑化している。画像は、かつては一様な色の多角形領域であったかもしれないが、現在では、ブレンド又はテクスチャで構成されることが多い。ブレンド又はテクスチャにおける画像画素の計算はコストがかかるので、この値を生成し、新規の画像からの異なる値と完全に置き換えるようなシステムは重大なペナルティを負うことになる。現在の画像は、ブレンドとして一般的に知られているより複雑な透明度のオブジェクトを含む。
【発明の開示】
【発明が解決しようとする課題】
【０００６】
本発明の目的は、既存の構成の１つ以上の欠点をほぼ克服、あるいは、少なくとも改善することである。
【課題を解決するための手段】
【０００７】
本発明の第１の様態によれば、画像を構成する各層を合成する画素生成方法であって、
前記各層をＺ値でソートするソート工程と、
前記ソート工程でソートされた各層のうち、可変アルファ塗り潰しの層より上方の塗り潰しの層に対する貢献度を計算する計算工程と、
前記ソート工程でソートされた各層の中に、可変アルファ塗り潰しの層が含まれているか否かを判定する判定工程と、
前記判定工程で可変アルファ塗り潰しの層が含まれていると判定された場合、可変アルファ塗り潰しの層を含むその下方の塗り潰しの層をボトムアップで合成し、ボトムアップで合成された塗り潰しの層に対する貢献度を計算し、
可変アルファの塗り潰しの層より上方の塗り潰しの層を、対応する貢献度に基づいた計算値が閾値に達するまでトップダウンで合成することで、出力画素を生成する生成工程と
を有することを特徴とする。
【００１２】
本発明の別の面によると、上述の方法のうちのいずれか１つを実現する装置を提供する。
【００１４】
本発明の他の面も開示される。
【発明を実施するための最良の形態】
【００１５】
目次
１．序論
１．１．座標空間
１．２．グラフィックオブジェクト
１．３．グリフ
１．４．ｚ－レベル
１．５．ストローキング
１．６．モーフィング
２．ドライバモジュール
２．１．スプライト
２．１．１．スプライト：変換行列
２．１．２．グラフィックオブジェクトとその深度
２．２．表示リスト
２．２．１．フレームレンダリング
２．２．２．グラフィックオブジェクト及びｚ－レベル
２．２．３．ローカル深度及び絶対深度
３．変換、モーフィング、及びストローキング

３．１．モーフィング
３．２．変換
３．３．稜線の生成
３．４．ストロークの稜線及びｚ－レベルへの分解
３．４．１．直線稜線のストローキング
３．４．２．曲線稜線のストローキング
３．４．３．結合のストローキング
３．４．４．同方向又は逆方向稜線結合のストローキング
３．４．５．経路の終了でのエンドキャップの生成
３．４．６．ストロークされた稜線とストロークされていない稜線との間のエンドキャップ
３．４．７．不透明なストロークに対するｚ－レベル割当て
３．４．８．透明なストロークに対するｚ－レベル割当て
３．４．９．ストローキング基本要素の変換
３．５．フィルタリング
３．６．稜線トラッキングパラメータの生成
３．６．１．直線稜線トラッキングパラメータの生成
３．６．２．２次ベジェ曲線トラッキングパラメータの生成
３．６．３．符号の判定
３．６．４．楕円弧トラッキングパラメータの生成
３．６．５．グリフ稜線トラッキングパラメータの生成
４．ソーティング
５．稜線処理
５．１．入出力
５．２．トップレベルの動作
５．３．アクティブ稜線トラッキング
５．４．稜線処理の例
５．５．稜線のアクティブ稜線への変換及び稜線持続性
５．５．１．静止稜線の持続性
５．６．アクティブ稜線処理
５．７．稜線のトラッキング
５．７．１．直線のトラッキング
５．７．２．２次ベジェのトラッキング
５．７．３．２次多項フラグメントのトラッキング
５．７．４．楕円弧のトラッキング
５．７．５．グリフのトラッキング
５．８．アンチエイリアシング及び交差メッセージ生成
５．８．１．グリフに対する交差メッセージの生成
５．９．交差メッセージの例
５．９．１．別の例
５．１０．アクティブ稜線及び交差メッセージの再順序付け
６．ｚ－レベル活動化モジュール
６．１．注目ｚ－レベルの順序付けられた集合
６．２．ｚ－レベル活動化モジュールにおける制御のフロー
６．３．ｚ－レベルの活動化及び非活動化：ワインディングカウント
６．４．続注目ｚ－レベルの順序付けられた集合
６．５．注目ｚ－レベルの順序付けられた集合への新規のｚ－レベルの追加
６．５．１．注目ｚ－レベルの順序付けられた集合のハードウェアでの維持
６．６．ランの処理
６．７．Ｓ－ｂｕｆｆｅｒのＡ－ｂｕｆｆｅｒへの変換：ワインディング規則

６．８．続ランの処理
７．合成モジュール
７．１．中間生成物
７．２．ｚ－レベル塗りつぶし
７．３．基本的なフロー
７．４．グラフィカルな概要
７．５．貢献度計算
７．６．ボトムアップ合成
７．７．トップダウン合成
７．８．代替の合成手法
７．９．トップダウンの利点
８．画素生成
８．１．線形傾斜画素生成
８．２．放射傾斜画素生成
８．３．ビットマップ画像画素生成
９．画素抽出
９．１．入力データ
９．２．フレームバッファへの出力
９．３．ディスプレイへの直接出力
９．４．ハーフトーン処理
１０．実現例

１．序論

本文書は、最小限のコンピューティングリソースを使用して２Ｄグラフィックオブジェクトをレンダリングするシステムであるシン・クライアントイメージングエンジン（ＴＣＩＥ）について説明する。このリソースレベルが適用される場合の例として、小型のディスプレイを備えた又は備えていない携帯型装置と、プリンタ及び複写機などのオフィス機器とがある。ハンドヘルドのコンピューティング装置には携帯電話機及びゲームが含まれる。ＴＣＩＥシステム６９９のトップレベルの図が図５６に示される。図５６において、ＴＣＩＥシステム６９９は処理モジュールのパイプラインとして構成されている。各モジュールについては、システム内をデータが流れる順序で説明する。各モジュールは、概念的に表示リストコンパイラ６０８とレンダリングエンジン６１０とに分けられる。表示リストコンパイラ６０８は所望の出力を説明する情報を用意し、レンダリングエンジン６１０はこの情報を使用して出力画像を生成する（例えば、表示装置又はフレームバッファへのレンダリング）。ＴＣＩＥシステム６９９は、時間的に間隔の空いた一連の出力画像を生成するのに使用することができる。この出力画像を以降は「フレーム」と呼ぶ。ＴＣＩＥシステム６９９の使用により、アニメーション（すなわち、「動画」）が出力ディスプレイ上で再生される効果が生まれる。

【００１６】

１．１．座標空間

図５６において、システム６９９の第１のモジュールはドライバモジュール６１５である。ドライバモジュール６１５は、グラフィックオブジェクトの集合及びこれらの情報を維持する。図３４は、システム６９９により使用される空間を示す。図３４は、まず、オブジェクト空間３３５に描かれたグラフィックオブジェクトを示す。次に、同グラフィックオブジェクトがグローバル論理空間３３６へと変換された状態を示す。更に、同グラフィックオブジェクトがレンダ空間３３７へと変換された状態を示す。最後に、同グラフィックオブジェクトが表示空間３３８へと変換された状態を示す。

【００１７】

オブジェクト空間３３５からグローバル論理空間３３６への変換は、グラフィックオブジェクトの配置変換により達成される。この配置変換は、後述の変換行列の階層による生成物であっても良い。グローバル論理空間３３６からレンダ空間３３７への変換は、グロ

ーバル論理座標を小画素へと変換するビューイング変換により達成される（アンチエイリアシングのため）。レンダ空間３３７から表示空間３３８への変換は、構成小画素から表示画素を生成するアンチエイリアシングプロセスにより達成される。逐一アンチエイリアシングを行なう縮退したケースでは、レンダ空間３３７及び表示空間３３８は同じである。すなわち、グローバル論理空間３３６は表示空間３３８へと直接変換することができる。

【００１８】

１．２．グラフィックオブジェクト

システム６９９への入力は、グラフィックオブジェクトの集合と関連するメタデータとから成る。図１６（ｃ）は、ディスプレイへとレンダリングされたグラフィックオブジェクト１７１を示し、対応する構成要素は図１６（ａ）及び図１６（ｂ）に示されている。グラフィックオブジェクトは、１つ以上の描画基本要素（新規の描画位置、直線、及び曲線）から成る順序付けられた集合により記述される２次元表示基本要素である。描画基本要素はグラフィックオブジェクトの輪郭の一部を記述する。各基本要素は１つ以上の座標と関連付けられている。新規の描画位置は、オブジェクトの原点からの絶対オフセットとして指定されても、あるいは、前の基本要素の終点に対する相対オフセットとして指定されても良い。新規の描画位置は単一の座標により記述され、直線は１対の座標により記述され、曲線は３つの座標のシーケンスにより記述される。直線は１対の座標を使用して線の始点及び終点を定義する。曲線は２次ベジェ曲線として実現される。第１の座標はベジェ曲線の開始を定義し、第２の座標は制御点を定義し、第３の座標はベジェ曲線の終点を定義する。ベジェ曲線は当業者には周知である。

【００１９】

各稜線の座標は、相対座標の順序付けられた集合として格納される。この相対座標により、必要なメモリ記憶容量が削減されると共に稜線の方向が判定される。直線又は曲線の第１の座標は前の描画基本要素の最後の座標であることが暗示されている。表１は、図１６（ｃ）に示す表示オブジェクトを形成することができる基本要素の順序付けられた集合の例である。この例では、Ｙ座標は下に向かって増加し、Ｘ座標は右に向かって増加する。また、新規の描画位置、直線、及び曲線に対する用語は、それぞれＭＯＶＥＴＯ＿ＡＢＳ、ＭＯＶＥＴＯ＿ＲＥＬ、ＬＩＮＥＴＯ、及びＣＵＲＶＥＴＯである。この例での始点は(0,0)１４１である。

【００２０】

表１

基本要素の型		座標（ＭＯＶＥＴＯ＿ＡＢＳ以外は相対）
ＭＯＶＥＴＯ＿ＡＢＳ	１７３	（０，０）１４１
ＭＯＶＥＴＯ＿ＲＥＬ	１７４	（４０，－５０）１７７
ＬＩＮＥＴＯ	１５７	（０，８０）１４２
ＬＩＮＥＴＯ	１５８	（１０，０）１４３
ＬＩＮＥＴＯ	１７２	（０，－５０）１４４
ＬＩＮＥＴＯ	１５９	（３０，０）１４５
ＬＩＮＥＴＯ	１６０	（０，５０）１４６
ＬＩＮＥＴＯ	１６１	（１００，０）１４７
ＬＩＮＥＴＯ	１６２	（０，－８０）１７８
ＬＩＮＥＴＯ	１６３	（－３０，－３０）１４８
ＬＩＮＥＴＯ	１６４	（－８０，０）１４９
ＬＩＮＥＴＯ	１６５	（－３０，３０）１７７
ＬＩＮＥＴＯ	１６６	（１４０，０）１７８
ＭＯＶＥＴＯ＿ＲＥＬ	１７５	（－３０，４０）１７６
ＣＵＲＶＥＴＯ	１６８	（０，－２０）１５０、その後（－２０，０）１５１
ＣＵＲＶＥＴＯ	１６７	（－２０，０）１５２、その後（０，２０）１５３

ＣＵＲＶＥＴＯ　１６９　　（０，２０）１５４、その後（２０，０）１５５
ＣＵＲＶＥＴＯ　１７０　　（２０，０）１５６、その後（０，－２０）１７６

１．３．グリフ
グリフは、特殊な型のグラフィックオブジェクトである。グリフは、常に表示空間へと直接描画されるという更なる制限を有する。グリフは、レンダリング中の形状が
（ｉ）小さく、
（ii）ちょうど画素境界上に配置されるように設計されている
状況向けに設計されている。

【００２１】
グリフにより表現されるのに適した形状の例には、暗示されたフォント文字が含まれる。

【００２２】
経路の代わりに、グリフは２つの関連付けられた塗りつぶしを有する１画素当たり１ビットのビットマップにより表現される。このビットマップはマスクのような役割を果たす。ビットが設定される場合には「オン」塗りつぶしが表示され、ビットが設定されない場合には「オフ」塗りつぶしが表示される。

【００２３】
１．４．ｚ－レベル
ｚ－レベルはグラフィックオブジェクトの稜線の部分集合により包囲されるディスプレイの一部がどのように彩色されるべきかを記述するのに使用される表示基本要素である。例えば、ｚ－レベルは包囲された領域を単色で塗りつぶされた領域として記述することができるだろう。また、ｚ－レベルには絶対深度が割り当てられるが、この絶対深度はどのｚ－レベルがどのｚ－レベルの上に出現すべきかを指定する整数値である。高い絶対深度を有するｚ－レベルは、低い絶対深度を有するｚ－レベルの上でレンダリングされる。

【００２４】
直線描画基本要素及び曲線描画基本要素には最大２つのｚ－レベル－基本要素の描画方向の左側にレンダリングされる予想第１ｚ－レベル及び基本要素の描画方向の右側にレンダリングされる予想第２ｚ－レベル－が関連付けられる。図３３（ａ）から図３３（ｃ）は、先に図１６（ｃ）においてオブジェクト１７１として示されたグラフィカルオブジェクト３３１を作成するために使用される左側及び右側の塗りつぶしを記述することによって、この概念を実証する。

【００２５】
図３３（ａ）は描画基本要素３１６から３２９を示す。基本要素３１６から３２９は図３３（ｃ）に示すｚ－レベル３３３、３３２、及び３３４を参照する。ｚ－レベルは絶対深度の順序で示される。すなわち、ｚ－レベルは絶対深度２、３、及び４をそれぞれ有することができる。表２はどのｚ－レベルを描画基本要素が参照するかを示す表である。例えば、ＬＩＮＥＴＯ　３１６はページの下に向かい、描画方向の左側にｚ－レベル３３２を有する。レンダリング結果を図３３（ｃ）に示す。

【００２６】

表２

描画基本要素	左ｚ－レベル	右ｚ－レベル
３１６	３３２	なし
３１７	３３２	なし
３３０	３３２	なし
３１８	３３２	なし
３１９	３３２	なし

３２０	３３２	なし
３２１	３３２	なし
３２２	３３３	なし
３２３	３３３	なし
３２４	３３３	なし
３２５	３３３	３３２
３２７	３３４	３３２
３２６	３３４	３３２
３２８	３３４	３３２
３２９	３３４	３３２

ｚ－レベルの書式は、単色、１色以上により記述される線形ブレンド、１色以上により記述される放射状ブレンド、又はビットマップ画像を含んでも良いが、これに限定されない。また、これらのｚ－レベル書式の全ては透明度（アルファ）チャネルをサポートする。図３３のｚ－レベル３３３、３３２、及び３３４は、単色書式ｚ－レベルを表現する。これらのｚ－レベルはパイプラインの大部分において不変のまま使用される。

【００２７】

１．５．ストローキング

描画基本要素はペン幅と関連付けることができる。ペン幅を有する描画基本要素は複数の稜線（稜線には幅がない）へと変換される。これらの稜線は、ペンストロークを表現する閉じた塗りつぶし形状を形成する。詳細については、変換、モーフィング、及びストローキングの表題の節を参照のこと。

【００２８】

１．６．モーフィング

モーフィングも従来技術において周知である。モーフィングは、２つのグラフィックオブジェクト用の描画基本要素の集合とグラフィックオブジェクトが２つの集合間の補間に従って描かれることを指定する比率とを供給することとして定義することができる。これは、モーフィング、ストローキング、及び変換モジュールの表題の節においても詳細に説明されている。

【００２９】

２．ドライバモジュール

ドライバモジュール６１５については、処理する情報とどの情報をレンダリングパイプラインの残りの部分に渡すかという観点で考察する。ドライバモジュール６１５の役割は、描画基本要素の集合を編成することである。描画基本要素は、まず、上述のようにグラフィックオブジェクトへとまとめられる。

【００３０】

グラフィックオブジェクトは後述するようにスプライトへとまとめることができる。このスプライトは、集合全体に当てはまる特性を有するものとして指定することができる。ドライバの第１の役割は、グラフィカルパイプラインの残りの部分を複雑化することなく、スプライト上で効率的且つ高レベルな動作ができるようにすることである。ドライバモジュール６１５がグラフィックパイプラインの後続のモジュールに対して描画情報を出力すると、スプライトの特性が各グラフィックオブジェクトの各描画基本要素へと適用される（例えば、変換行列）。これにより、後続のモジュールは、有向稜線及びｚ－レベルのみを処理することができる。

【００３１】

２．１．スプライト

ドライバモジュール６１５は、入力の一部としてスプライトを受け入れる。スプライトは従来技術において周知であり、ドライバモジュール６１５において、スプライトは変換行列と、深度と、スプライトのコンテキスト内に存在するグラフィックオブジェクトのリ

ストとを有する基本要素を参照する。スプライトは０個以上のグラフィックオブジェクトと０個以上の他のスプライトとを包含することができる。「包含する」は、スプライトの変換行列が当該スプライトを所有する全グラフィックオブジェクト及び全スプライトに適用されることを意味する。他の基本要素を包含するスプライトの概念は、それが包含する全グラフィックオブジェクト及び全スプライトの深度がそのスプライトにとって「ローカル」であることを意味する。グラフィックオブジェクトが他のグラフィックオブジェクト又はスプライトを含むことはない。

【００３２】

２．１．１．スプライト：変換行列

変換行列は従来技術において周知である。スプライトの変換行列は、スプライトが所有する全てのグラフィックオブジェクトに適用される。変換行列はスプライトに対してローカルの空間を定義する。図１７（ｂ）において、２つのスプライト及び２つのグラフィックオブジェクトが提供される。これらをレンダリングする方法はツリーにより記述される。スプライト１８５は、スプライト１８９及びグラフィックオブジェクト１８０の双方を包含する。すなわち、リンク１８８及び１８６は所有関係を表現する。スプライト１８９は第２のグラフィックオブジェクト１８２を包含する。

【００３３】

図１７（ａ）は、この例において存在する変換行列の幾何学的配置を表現する。空間１７９はスプライト１８５内にオブジェクトを包含する。オブジェクト１８０は空間１７９内に位置する。このため、オブジェクト１８０の構成描画基本要素の座標は空間１７９を参照する。図１７（ｂ）のスプライト１８９もこの空間１７９に位置する。空間１８１は、スプライト１８９のグラフィックオブジェクトが位置する空間を表現する。

【００３４】

この例では、スプライト１８９は、回転、平行移動、及びスケーリングを有する変換を有する。平行移動は点線１８３により表現され、回転は角度１８４により表現され、スケーリングは目盛り１７９及び１８１の相対的な大きさにより表現される。各軸はスプライト１８５及びスプライト１８９の空間をそれぞれ表現するのに使用される。この例では、スケーリングは両軸とも同じである。

【００３５】

更に、図１７（ａ）を参照すると、スプライトが所有するグラフィックオブジェクトの配置を記述する変換行列は、そのスプライトの配置を記述する変換行列と連結される。図１７（ａ）の例において、グラフィックオブジェクト１８２に適用される変換行列は、スプライト１８９及びスプライト１８５の変換行列の連結である。オブジェクト１８２に対するこの結果の変換行列は、オブジェクト１８２が包含する全描画基本要素に適用される。

【００３６】

２．１．２．グラフィックオブジェクトとその深度

グラフィックオブジェクトの深度は、そのオブジェクトを包含するスプライトにとってローカルである。これは図３１（ａ）から図３１（ｃ）により示される。図３１（ａ）に示すレンダリングツリーでは、ノード２９４は別のスプライト２９６及びグラフィックオブジェクト３０２を包含するスプライトを表現する。所有関係は有向線２９５及び３０１によりそれぞれ示される。スプライト２９６は、それぞれ所有関係２９７及び３００により示されるグラフィックオブジェクト２９８及び２９９を包含する。表４はこれら全ての基本要素のローカル深度を提供する。

【００３７】

表４

ラベル	基本要素	ローカル深度	絶対深度

２９４	スプライト	２	なし
２９６	スプライト	２	なし
２９８	グラフィックオブジェクト（円）	１	３
２９９	グラフィックオブジェクト（四角形）	２	４
３０２	グラフィックオブジェクト（三角形）	１	２
３０９	背景ｚ－レベル	１	１

ローカル深度の概念は図３１（ａ）のサブツリーの外観を見ることによって説明することができる。図３１（ｂ）はスプライト２９６の外観を示す（個別にレンダリングされるオブジェクト２９８及び２９９）。オブジェクト２９８はオブジェクト２９９よりも深度値が小さいので、オブジェクト２９８はオブジェクト２９９の下方に出現する。図３１（ｃ）は、グラフィックオブジェクト３０２及びスプライト２９６の外観を示す。オブジェクト３０２の深度値の方が小さいので、オブジェクト２９６の下方に出現する。スプライト２９６の子のローカル深度はローカル深度凡例３０４に従って保管される。

【００３８】

ノード３０３は所有権ツリーのルートであり、全ての最上のスプライトを包含するスプライトを表現する。ルート３０３は、常に、最低のローカル深度（グローバル深度においても最低）において背景ｚ－レベル３０９を所有する。従って、この背景ｚ－レベルは全グラフィックオブジェクトの下方に位置する。

【００３９】

２．２．表示リスト

単一のフレームに対する全スプライト及び全グラフィックオブジェクトリスト（所有関係、ローカル深度、及び変換を含む）は表示リストと呼ばれる。表示リストは従来技術において周知である。ドライバモジュール６１５はツリーの形態で表示リストを維持する。このツリーは、所有関係の観点からスプライト及びグラフィックオブジェクトをまとめ、ローカル深度の観点から順序付けられる。図３１（ａ）はこの構成も示す。例えば、スプライト２９４の子はローカル深度により順序付けられる。第１の子３０２は最低ローカル深度１にあり、次の子２９６はより高いローカル深度２にある。

【００４０】

２．２．１．フレームレンダリング

アニメーションの各フレームにおいて、表示リストはディスプレイへとレンダリングされる前に修正されても良い。表示リストはフレーム間で保持される。

【００４１】

２．２．２．グラフィックオブジェクト及びｚ－レベル

スプライトが複数のグラフィックオブジェクトを参照可能であるのと同様に、グラフィックオブジェクトは複数のｚ－レベルを参照することができる。これらのｚ－レベルの各々は、グラフィックオブジェクト内に深度を有する。例えば、図３３（ｃ）のグラフィックオブジェクトは、図３３（ｂ）に示すような深度を伴う３つのｚ－レベルを必要とする。

【００４２】

２．２．３．ローカル深度及び絶対深度

ドライバモジュール６１５内ではローカル深度が使用されるが、後続の各モジュールは絶対深度を必要とする。ドライバモジュール６１５は各ｚ－レベルに絶対深度を割り当てる。

【００４３】

グラフィックオブジェクトのｚ－レベルに対する絶対深度は、ドライバモジュール６１５から後続の各モジュールへと渡される。これらのモジュールでは絶対深度は単に深度と呼ばれる。後続の全てのモジュールが絶対深度を割り当てられたｚ－レベルを用いて動作することが理解されるだろう。この割当ては、レンダリングの直前にフレームごとに行な

われる。これは、新規の表示オブジェクト又はスプライトが表示リストへと挿入された場合、あるいは、古いものが破棄された場合にのみ行なわれる。

【００４４】

図３１（ａ）は単一のフレームに対する全基本要素を示すものとすると、上記表４は図３１（ａ）の全基本要素に対する絶対深度を示す。単一のフレームの全ｚ－レベルに対する絶対深度は１で開始する。この１は特殊な背景ｚ－レベルに割り当てられる。ｚ－レベルは背景ｚ－レベルから昇順に並んだ連続整数により番号付けされる。スプライトはそれ自体が絶対深度を持つことはない（子孫の深度についての情報を記録する可能性はある）。スプライトは、包含する描画オブジェクトに関連する情報（例えば、配置変換）のコンテナとして存在するが、それ自体は描画オブジェクトではない。レンダリングパイプラインの後続のモジュールはスプライトを使用しない。グラフィックオブジェクトはドライバモジュール６１５の外側には存在しない。グラフィックオブジェクトのローカル深度はグラフィックオブジェクトを構成する描画基本要素により参照されるｚ－レベルの絶対深度を計算するのに使用される。

【００４５】

絶対深度は、表示リストの内容がレンダリングされる際にフレームごとに１度だけ計算される。この絶対深度を計算する方法は図４４に示される。図４４は表示リストツリーを示す。表示リストツリーのルートノード４４０は、他の全てのｚ－レベルの下方に位置する背景ｚ－レベルを所有する。すなわち、背景４４１は最低の予想絶対深度１を有する。ノード４３２から４３９は表示リストツリーの残りの部分を構成する。

【００４６】

絶対深度は、点線４４２により示されるツリーの深さ優先縦断中に各グラフィックオブジェクトの各ｚ－レベルに割り当てられる。グラフィックオブジェクトのｚ－レベルへの絶対深度の割当ては、そのグラフィックオブジェクトにおける最低ｚ－レベルで開始し、直「前」のグラフィックオブジェクトの最高ｚ－レベルよりも１つ高い絶対深度が割り当てられる。「前」とは、このグラフィックオブジェクトが深さ優先縦断において先に訪れたオブジェクトであることを意味する。図４４において、深さ優先縦断がグラフィックオブジェクト４３８に到達する様子が示される。オブジェクト４３８の最低ｚ－レベルに絶対深度を割り当てるために、先に訪れたグラフィックオブジェクトであるオブジェクト４３５の最高ｚ－レベルよりも１つ高い絶対深度が割り当てられる。

【００４７】

オブジェクト４３８が更にｚ－レベルを有する場合、次に低いｚ－レベルから最高のレベルまで連続の絶対深度が割り当てられる。全てのスプライト及びグラフィックオブジェクトは独自のローカル深度範囲を有するのでインターリービングが不可能であることを喚起するであろう。表示リストツリーの全ノードは絶対深度を格納する。グラフィックオブジェクトは最上のｚ－レベルの絶対深度を格納する。スプライトは子孫の最上のｚ－レベルの絶対深度を格納する。ツリーの縦断は常にルートから開始する。

【００４８】

表示リストの一部のみがノードの挿入又は削除により変更された場合、効率化のためにツリーの最小限の更新が行なわれる。表示リストツリーの縦断中、絶対深度の更新は変更された第１のノードから実行される。これより前の（縦断順）グラフィックオブジェクトのｚ－レベルは、それぞれの有効な絶対深度を保持する。第１の変更されたノードに到達する前に各ノードの最上の絶対深度を記録することによって、絶対深度の追跡が達成される。第１の変更されたノードに遭遇したときには、この記録された値から絶対深度の割当てが続けられる。

【００４９】

例えば、図４４において、オブジェクト４３８が表示リストツリーの新規のノードであり、前のフレーム以来の唯一の変更である場合、絶対深度の割当てはオブジェクト４３８から開始するであろう。

【００５０】

３．変換、モーフィング、及びストローキング

図５６のモーフ／変換／ストローキングモジュール６１６は、レンダリング対象の描画オブジェクトの１つ以上の輪郭をまとめて定義するドライバモジュール６１５からの座標の１つ以上の順序付けられた集合を入力とする。座標の順序付けられた集合は、以下のものを含む追加のパラメータにより達成されても良い：

－モーフ率
－変換行列
－ストローク幅
－左側ｚ－レベルへの参照
－右側ｚ－レベルへの参照、及び
－ストロークカラーｚ－レベルへの参照

モーフ率、変換行列、及びストローク幅のパラメータは、グラフィックオブジェクトの輪郭を記述する座標の順序付けられた集合をディスプレイ上にどのように配置すべきかを記述する。その他のパラメータは、座標の順序付けられた集合により定義される輪郭内の表示画素に対する色、ブレンド、又はテクスチャを示すｚ－レベルへの参照である。

【００５１】

図４６は、モジュール６１６により段差のある座標の各順序付けられた集合上で行なわれる処理を示すフロー図である。この処理の目的は、段差のある座標の各順序付けられた集合（例えば、図１６（ａ）を参照）を稜線の対応する順序付けられた集合へと変換することである。稜線の順序付けられた集合の各稜線は、少なくともレンダ空間における開始座標及び終了座標により記述される。すなわち、稜線は指向性を有する。稜線が所有するｚ－レベルへの左参照及び右参照は、この指向性によるものである。モーフ率、変換行列、及びストローク幅のパラメータはこの処理を制御するのに使用される。

【００５２】

更に、モジュール６１６は、任意で１つ以上のグリフ記述を受け入れる。これらの記述は以下の情報を含む：

－グリフ位置、
－グリフ高さ／幅、
－グリフビットマップ、
－「オン」ｚ－レベルへの参照（ＮＵＬＬの可能性有）
－「オフ」ｚ－レベルへの参照（ＮＵＬＬの可能性有）、及び
－ステージ４７０においてパイプラインに入るグリフ記述

３．１．モーフィング

モジュール６１６がモーフオブジェクトに対応する座標の順序付けられた集合を受信する場合（すなわち、図４６のステージ４６４＝ｙｅｓ）、各座標の２つのバージョンがモーフ率パラメータと共に受信される。受信された座標の一方のバージョンは開始バージョンと呼ばれる。受信された座標の第２のバージョンは終了バージョンと呼ばれる。ステージ４６７において、モジュール６１６はモーフ率パラメータを使用して受信された各座標の単一の中間バージョンを補間する。例えば、図３５（ａ）及び図３５（ｃ）は、モーフ「開始」形状及びモーフ「終了」形状をそれぞれ表現する座標の順序付けられた集合を示す。座標の順序付けられた両集合は、「論理」座標空間中の起点３４２で開始／終了するのが示されている。モーフプロセスの目的は、図３５（ａ）の「開始」バージョンと図３５（ｃ）の「終了」バージョンとの間で補間することによって、図３５（ｂ）に示されるような形状の中間バージョンを生成することである。各座標のバージョン（「開始」及び「終了」）は、座標対の順序付けられた集合として提示される。この例では、３３９と３４６とが第１の対であり、それに続くのが、３４３と３４０の対、３４４と３４７の対、及び最後の対３４１と３４２である。モーフ率は各座標対からの中間バージョンを補間するのに使用される。モーフ率０が図３５（ａ）の「開始」形状を表現するのに使用され、モーフ率１が図３５（ｃ）の「終了」形状を表現するのに使用される場合、モーフ率０．２５は図３５（ｂ）に示す形状に対応するであろう。従って、モーフプロセスは出力とし

て３５０、３５１、３５２、及び３５３により示される座標の順序付けられた集合を生成するであろう。

【００５３】

３．２．変換

次に、図４６のステージ４６８において、変換行列が各座標に適用される。変換行列により、座標（及び描画オブジェクト）の回転、スケーリング、及び／又は平行移動を指定することができる。この記述では、変換行列は「論理空間」から「レンダ空間」へと座標（及びそれが記述する描画オブジェクト）を変換するものとされる。

【００５４】

３．３．稜線の生成

座標の順序付けられた集合がモーフィングされ（必要に応じて）、変換された後、図４６のフローチャートの後続のステージ４６５において、座標の順序付けられた集合が稜線の順序付けられた集合へと変換される。ステージ４６５の達成方法の例は図１５のフローチャートにより示される。この例では、座標が３つの型の描画基本要素のうちの１つの一部を形成可能であることを想定している。３つの型とは、直線、２次ベジェ曲線、及び新規の描画位置である。例では、どの型の描画基本要素が後続するか及び描画基本要素の終了にいつ到達するかを示すために追加の情報（例えば、タグバイト）が設けられることも想定している。

【００５５】

まず図１５に示すように、ステップ１４０において現在の描画位置が(0,0)に初期化される。続いて、ステップ１２４、１２５、１２６、及び１２７において、次に遭遇する「続く」描画基本要素の型が判定される。

【００５６】

続く座標が新規の描画位置を記述することを型が示す場合（ステップ１２４）、ステップ１２８において座標が読み出され、ステップ１３１において新規の現在の描画位置を設定するのに使用される。

【００５７】

続く座標が直線稜線を記述することを型が示す場合（ステップ１２５）、ステップ１２９において現在の描画位置と等しい開始座標が与えられる新規の直線稜線が生成される。ステップ１３２において座標が読み出される。この座標は、ステップ１３４における新規の直線稜線の終了座標及びステップ１３６における新規の描画位置の双方に使用される。

【００５８】

続く２つの座標が（例えば、２次ベジェ）曲線を記述することを型が示す場合（ステップ１２６）、ステップ１３０において現在の描画位置と等しい開始座標を持つ新規の曲線が生成される。ステップ１３３において第１の座標が読み出され、ステップ１３５において新規の曲線の「制御点」座標として使用される。ステップ１３７において第２の座標が読み出され、ステップ１３８における新規の曲線の終了座標及びステップ１３９における新規の描画位置として使用される。

【００５９】

ステップ１２７において、処理は座標の順序付けられた集合が終了に到達するまで継続する。

【００６０】

３．４．ストロークの稜線及びｚ－レベルへの分解

図４６に戻って、処理の次のステップはステップ４７１において任意で選択されるストローキングステップ４６６である。ストローキングは、ある特定の太さのペンを使用して曲線及びベクトルの経路をたどる効果を再現できるように１つ以上の輪郭を生成するプロセスである。ストローキングの例は、図５０（ａ）から図５０（ｅ）に示される。図５０（ａ）は、ストローク対象の３本の稜線５２３の順序付けられた集合を示す。生成されたストローク輪郭５２４は図５０（ｂ）に示される。図５０（ｂ）は、ある特定のペン幅５２５で先端が円形のペンでストローク対象の３本の稜線の経路をたどったときの効果を示

している。

【００６１】

図４６では、ストローキングステップ４６６が変換ステップ４６８の後に行なわれなければならないことを示唆しているが、ストローキングが変換に先立って行なわれる可能性を残すのも望ましいであろう。この場合は異なる結果が得られる。例えば、図５０（ａ）を再度参照すると、稜線５２３により記述される元の形状が最初に図５０（ｃ）に示す形状５２６を生成するように変換され、その後にストロークされる場合、結果は図５０（ｄ）に示す形状５２８になるであろう。しかし、稜線５２３により記述される元の形状が最初にストロークされ（５２４）、その後に同じ変換が適用される場合、結果は図５０（ｅ）に示す形状５２７になる。ＴＣＩＥシステム６９９は、任意でストローキング４６６の後にステップ４６８の座標変換を実行することによって、いずれの結果を達成することもできる。

【００６２】

尚、ストローキングについての以下の考察では、正のＹ軸が正のＸ軸に対して９０度時計回りである座標系（例えば、Ｘが右に向かって増加し、Ｙが下に向かって増加する）においてストローキングが行なわれるものと想定する。

【００６３】

モジュール６１６は、稜線の順序付けられた集合において稜線を繰り返し処理することによって、稜線の元の順序付けられた集合をストロークする。稜線ごとに、元の稜線の経路に中心を置いて経路に沿ってたどった先端が円形のペンのストロークの左限度及び右限度に対応する新規の左側稜線及び新規の右側稜線が生成される。閉じた形状を形成しない稜線の元の順序付けられた集合の開始及び終了におけるエンドキャップと同様に、２本の連続的な元の稜線間で結合を生成するのには楕円弧が使用される。稜線の順序付けられた集合の第１の稜線の開始座標が稜線の順序付けられた集合の最後の稜線の終了座標と等しい場合にのみ、稜線の順序付けられた集合は閉じたものとなる。

【００６４】

各弧は、オブジェクト空間に円弧として配置され、後続の各節で説明するプロセスを介して表示空間の楕円弧へと変換される。この基本要素は現在のところ内部的にのみ使用されているが、弧はユーザが利用可能な描画基本要素として容易に公表されるであろう。

【００６５】

３．４．１．直線稜線のストローキング

図４９（ａ）は、ストローク対象の直線稜線４７３の一例を示す。この稜線は開始座標４７４及び終了座標４７２により記述される。図４９（ｂ）は、直線稜線４７３をストローキングした所望の結果を示す。結果は左側稜線４７７及び右側稜線４７８から構成される。

【００６６】

直線稜線をストロークするには、

－直線稜線の方向に対して９０度反時計回りの方向を有し、

－ストロークのペン幅の半分に等しい長さの

左法線ベクトル４７９が生成される。

【００６７】

左側ストローク稜線及び右側ストローク稜線が、直線稜線４７３、４７４(X_s,Y_s)及び４７２(X_e,Y_e)と、左法線ベクトル４７９(X_n,Y_n)の座標を使用して計算される。

【００６８】

左側ストローク稜線４７７に対して、以下に示すように開始座標(X_{s_left},Y_{s_left})及び終了座標(X_{e_left},Y_{e_left})が計算される：

$X_{s\_left}=X_s+X_n$　　$Y_{s\_left}=Y_s+Y_n$

$X_{e\_left}=X_e+X_n$　　$Y_{e\_left}=Y_e+Y_n$

現在の稜線の座標から法線ベクトルを差し引くことによって右側ストローク稜線４７８が生成される：

$X_{s\_right}=X_s-X_n$　　$Y_{s\_right}=Y_s-Y_n$
$X_{e\_right}=X_e-X_n$　　$Y_{e\_right}=Y_e-Y_n$

３．４．２．曲線稜線のストローキング

図４９（ｃ）は、ストローク対象の曲線稜線４８３を示す。曲線稜線（２次ベジェ曲線又は２次多項フラグメント－ＱＰＦ）は、始点４８０、制御点４８１、及び終点４８２により記述される。

【００６９】

図４９（ｄ）は、２本の左法線ベクトル４８７及び４８８を示す。これらの法線ベクトルはこの稜線をストロークするために生成されなければならない。２本の左法線ベクトルの双方はストロークのペン幅の半分に等しい長さを有する。２本の左法線ベクトルのうちの最初のベクトル４８７は、始点４８０から制御点４８１へと延出するベクトルに対して９０度反時計回りである。２本の左法線ベクトルのうちの第２のベクトル４８８は制御点４８１から終点４８２まで延出するベクトルに対して９０度反時計回りである。

【００７０】

図４９（ｅ）において、始点４９３、制御点４９４、及び終点４９５を有する曲線の左側ストローク稜線が生成される。左側ストローク稜線の始点４９３(X_{s_left},Y_{s_left})は、平行移動４８０により計算され、曲線稜線の開始座標(X_s,Y_s)は第１の左法線ベクトル４８７$(X_{n1}Y_{n1})$により計算される：

$X_{s\_left}=X_s+X_{n1}$　　$Y_{s\_left}=Y_s+Y_{n1}$

左側ストローク稜線の終点(X_{e_left},Y_{e_left})は、平行移動４８２により計算され、曲線稜線の終了座標(X_e,Y_e)は第２の左法線ベクトル４８８(X_{n2},Y_{n2})により計算される：

$X_{e\_left}=X_e+X_{n2}$　　$Y_{e\_left}=Y_e+Y_{n2}$

左側ストローク稜線の制御点４９４は２本の直線の交点を使用して計算される。第１の直線は左側ストローク稜線始点４９３を通り、曲線稜線始点４８０から曲線稜線制御点４８１まで延出する直線に平行である。第２の直線は左側ストローク稜線終点４９５を通り、曲線稜線終点４８２から曲線稜線制御点４８１まで延出する直線に平行である。

【００７１】

曲線の右側ストローク稜線は、始点５０３、制御点５０４、及び終点５０５が描かれた図４９（ｆ）の例に示すように生成される。右側ストローク稜線の始点座標$(X_{s_right},Y_{s_right})$及び終点座標$(X_{e_right},Y_{e_right})$は以下の式により計算される：

$X_{s\_right}=X_s-X_{n1}$　　$Y_{s\_right}=Y_s-Y_{n1}$
$X_{e\_right}=X_e-X_{n2}$　　$Y_{e\_right}=Y_e-Y_{n2}$

右側ストローク稜線の制御点５０４は、２本の直線の交点を使用して計算される。第１の直線は右側ストローク稜線始点５０３を通り、曲線稜線始点４８０から曲線稜線制御点４８１まで延出する直線に平行である。第２の直線は右側ストローク稜線始点５０５を通り、曲線稜線終点４８２から曲線稜線制御点４８１まで延出する直線に平行である。

【００７２】

図４９（ｇ）は、種々の制御点を参照して描かれた結果の左側ストローク稜線５１４及び右側ストローク稜線を示す。

【００７３】

尚、ここで説明した技術は、曲線をストロークすることの類似であり、図４９（ｈ）に示す曲線５２０のような急な曲線には適していない。曲線の始点５１７、制御点５１８、及び終点５１９により形成される三角形において、制御角は始点５１７及び終点５１９を結合する直線に対する（すなわち、向かい合う）角度として定義される。制御角が１２０度以下である場合、曲線は鋭角であると考えられるであろう。便利な検査は、２本の左法線ベクトル５２１(X_{n1},Y_{n1})及び５２２(X_{n2},Y_{n2})のドット積の大きさを使用することである：

$(X_{n1}X_{n2}+Y_{n1}Y_{n2})^2<1/2(X_{n1}^2+Y_{n1}^2)\times(X_{n2}^2+Y_{n2}^2)$

である場合、曲線は鋭角である。

【００７４】

始点、制御点、及び終点により記述される２次ベジェ曲線を２等分する方法は従来技術において周知である。鋭角であると判定された曲線稜線は、まず、２つの等価な曲線稜線へと２等分され、更に２等分される。結果的に、合計４本の曲線稜線となる。外側の２本の新規の稜線（元の鋭角に隣接しない）が上述のプロセスを使用してストロークされる。内側の２本の新規の稜線は、従来技術において周知の技術を使用して直線稜線を類似する経路へとベクトル化される。この経路は第３．４．１節で説明されるプロセスを使用してストロークされる。

【００７５】

３．４．３．結合のストローキング

追加の曲線は、経路の稜線が結合する領域をストロークするために生成されなければならない。結合は入口稜線及び出口稜線を有するように定義される。図４５（ａ）は、結合４４４を形成する２本の直線稜線を示す。稜線４４３は結合が終点を形成するので入口稜線であり、稜線４４５は結合が始点を形成するので出口稜線である。左側ストローク稜線４４６及び４４７と右側ストローク稜線４４８及び４４９は上述のプロセスを使用して生成されたものである。左法線ベクトルは、入口稜線４６２の終了及び出口稜線４６３の開始に対して判定されるであろう。左側を先に考察する。入口稜線４５３の終了を出口稜線４５４の開始と接続する結合に対する曲線の輪郭を生成する必要があるか否かを判定するための検査が行なわれる。成分(X_{n1},Y_{n1})を有する左法線ベクトル４６２がストローク対象の入口稜線の終点に対して既知であり、且つ成分(X_{n2},Y_{n2})を有する左法線ベクトル４６３がストローク対象の稜線の開始に対して既知である場合、曲線の輪郭を生成する必要性を判定するために以下の検査を使用することができる：

$Y_{n1}X_{n2}<Y_{n2}X_{n1}$である場合、曲線稜線を使用してストロークの左側の各稜線をリンクし、直線稜線を使用してストロークの右側の各稜線をリンクする。

【００７６】

同様に、$Y_{n1}X_{n2}>Y_{n2}X_{n1}$である場合、曲線稜線を使用してストロークの右側の各稜線をリンクし、直線稜線を使用してストロークの左側の各稜線をリンクする。

【００７７】

$Y_{n1}X_{n2}=Y_{n2}X_{n1}$である場合、結合を形成する各稜線は結合において同じ又は反対の方向を有する。

【００７８】

図４５（ａ）に示す例では、上述の検査は曲線稜線が結合の左側に対して生成されなければならないことを示すであろう。この曲線稜線は以下の特性を有する厳密な円弧となる：

－弧の始点及び終点は点４５３及び４５４であり、
－弧の半径は左側ベクトル４６２の長さであり、
－弧の中心は元の経路点４４４であり、且つ
－弧は始点から終点まで時計回りに進む（弧は結合の右側が処理されている場合には反時計回りに進むであろう）。

【００７９】

これらの値は楕円弧の稜線を生成するのに使用され、その更なる処理については後述する。

【００８０】

図４５（ａ）に示す例を継続すると、上述の検査は、直線稜線が結合の右側に対して生成されなければならないことを示すだろう。結合は第１の右側稜線４５１の終点から第２の右側稜線４５０の始点までをリンクする。

【００８１】

図４５（ｂ）は、異なるストローク経路の同じ側の稜線４５５及び４５６が結合しない状況を示す。対応する頂点４６０及び４６１から延出し、稜線４５５及び４５６と関連付けられた制御点又は焦点を中心に湾曲するように新規の稜線を作成する。また、図４５（ｂ）は、結合しない対応する逆方向の稜線４５７ｂ及び４５７ａを示す。新規の稜線が頂

点４５８と４５９との間に挿入され、経路の終結を提供する。ストロークに対する影響がゼロであるので、これは直線稜線である。

【００８２】

３．４．４．同方向又は逆方向稜線結合のストローキング

結合をストロークする技術の上記説明では、ストローク対象の結合を形成する２本の元の稜線が同じ又は反対の方向をいつ有するのかを示すための検査について述べた：

$Y_{n1} X_{n2} = Y_{n2} X_{n1}$

ここで、(X_{n1},Y_{n1})はストローク対象の入口稜線の終了に対する左法線ベクトルであり、(X_{n2},Y_{n2})はストローク対称の出口稜線の開始に対する左法線ベクトルである。

【００８３】

更に、$X_{n2} = X_{n1}$である場合、左法線ベクトルは等しいため、入口稜線及び出口稜線は結合において同じ方向を有する。結合をストロークするために追加の稜線は必要とされない。

【００８４】

$Y_{n1}X_{n2} = Y_{n2}X_{n1}$であるが$X_{n2} \neq X_{n1}$である場合、入口稜線及び出口稜線は結合において反対方向であり、後述するように結合に対してエンドキャップを生成しなければならない。

【００８５】

３．４．５．経路の終了でのエンドキャップの生成

エンドキャップは、ストローク対象の閉じられていない経路の終端点に対して生成される。また、エンドキャップは入口稜線及び出口稜線の方向が反対の場合（上述のように）に稜線間の結合に対して生成されなければならない。

【００８６】

プロセスは楕円弧を使用して終端点に対するエンドキャップを生成する。ストロークのエンドキャップの例が図１８（ａ）に示される。図１８（ａ）は、点１９３で終端するストローク対象の経路１９２の元の稜線を示す。左側ストローク稜線１９５及び右側ストローク稜線２００は、それぞれ終点１９６及び１９８で終端するように描かれている。終端稜線の終点に対して成分(Xn,Yn)を有する左法線ベクトル１９４が示される。

【００８７】

エンドキャップを生成するために、楕円弧が以下の特性を有するように配置される：

－弧は左側終点１９６で開始し、右側終点１９８で終了し、
－弧の中心は元の経路点１９３であり、
－弧の半径は左法線ベクトル１９４の長さであり、
－弧は円の周りを時計回りに移動する。

有効な左ｚ－レベル参照及び／又は有効な右ｚ－レベル参照（有効なストロークｚ－レベル参照に加えて）を有する経路をストロークする際に、これはストロークされた経路が塗りつぶし対象の閉じた領域の一部を形成することを示す。このような経路に対して、任意のエンドキャップに追加の２本の直線稜線を追加する必要がある。

【００８８】

この場合、９０度時計回りに回転された終端稜線の終点に対する左法線ベクトルに等しくなるように、成分(X_{n_knee},Y_{n_knee})を有する新規の左法線ベクトル１９９が作成される。続いて、終端点１９３(X_j,Y_j)の座標及び成分(X_{n_knee},Y_{n_knee})を有する新規の左法線ベクトル１９９を使用してｋｎｅｅ座標１９７（X_{knee},Y_{knee}）が生成される：

$X_{knee}=X_j+X_{n\_knee}, Y_{knee}=Y_j+Y_{n\_knee}$

図１８（ｂ）は追加の２本の稜線２０７ａ及び２０７ｂを示す。両稜線はｋｎｅｅ点１９７で開始し、ストローク対象の元の稜線１９３の終端点で終了する。第１の追加の稜線２０７ａは追加の左側のストロークされた稜線であり、第２の追加の稜線２０７ｂは追加の右側のストロークされた稜線である。ｚ－レベル参照が左側のストロークされた稜線及び右側のストロークされた稜線へと割り当てられる方法が以下に記述される。

【００８９】

３．４．６．ストロークされた稜線とストロークされていない稜線との間のエンドキャップ

エンドキャップは、ストローク塗りつぶしを参照する稜線がストローク塗りつぶしを参照しない稜線により経路においてたどられるときにも生成される。図２８（ａ）は部分的にストロークされた三角形状を示す。頂点２７５は、ストロークされた稜線２７６がストロークされていない稜線２７７に結合する箇所である。エンドキャップが頂点２７５において生成される。

【００９０】

このエンドキャップは、前節に示したのと同じ方法、すなわち、経路の終了において生成されるエンドキャップに対して使用される同じ方法と後述する追加のステップで構成される。この方法を適用した結果が図２８（ｂ）に詳細に示されている。

【００９１】

形状の内部の塗りつぶし、すなわち、稜線２７８により参照される左側塗りつぶしを考慮する。これにより、小さなエンドキャップ稜線２８０の追加が内部の塗りつぶしが稜線の閉じた集合により完全に結ばれることを確実にすることが明らかである。稜線２７９、２８０、２８２、及び２７８は、結合の領域における形状の塗りつぶしを取り囲む稜線の集合の中にある。追加の稜線（不図示）が、結合内にない領域における塗りつぶしを取り囲む。従来技術においては周知であるが、形状を塗りつぶす方法は閉じようとする形状によって決まるーワインディング規則及びワインディングカウントが稜線により縛られる領域を塗りつぶすのに使用される場合は特にそうである。

【００９２】

図２８（ｂ）に示す稜線２７８は、ストローキング方法によって生成されたのではない唯一の稜線である。この稜線は入力稜線の集合から得られたものであり、出力稜線の集合に含まれる。

【００９３】

３．４．７．不透明なストロークに対するｚーレベル割当て

座標の順序付けられた集合が入力として提供され、その座標の順序付けられた集合がストロークされた経路を形成する場合、左ｚーレベル参照及び右ｚーレベル参照に加えて、有用ストロークｚーレベル参照及びペン幅が入力として必要である。ストロークに使用されるｚーレベルと経路を塗りつぶすのに使用されるｚーレベルの書式には差がない。一度座標の順序付けられた集合が稜線の順序付けられた集合へと処理されると、上述の技術を使用してストロークすることができる。モジュール６１６は左側ストローク稜線の集合と右側ストローク稜線の集合とを生成する。モジュール６１６が左ｚーレベル、右ｚーレベル、及びストロークｚーレベルを入力として受け取った場合、ストロークｚーレベルは不透明であると判定され、生成された左側ストローク稜線は：

ー入力として受信された左ｚーレベルを参照する左ｚーレベル参照、

ー入力として受信されたストロークｚーレベルを参照する右ｚーレベル参照を割り当てられ、生成された右側ストローク稜線は：

ー入力として受信されたストロークｚーレベルを参照する左ｚーレベル参照、

ー入力として受信された右ｚーレベルを参照する右ｚーレベル参照を割り当てられる。

【００９４】

本節では、左ストローク稜線又は右ストローク稜線への参照は、稜線の方向に従って見たときにストローク稜線の外側境界上の稜線への参照である。この点に関して、稜線がディスプレイの左上から右下へと向かうときには、左ストローク稜線は稜線の右側に出現するであろう。

【００９５】

このｚーレベル割当てが使用される場合、ストロークされた経路の元の稜線は破棄され、生成された左側ストローク稜線及び右側ストローク稜線により完全に置き換えられる。

【００９６】

一例が図５１（ａ）から図５１（ｅ）に示される。図５１（ａ）は、上述のストローキ

ング方法に対する入力稜線の集合を示し、この入力稜線が閉じた経路を定義する。稜線５３３、５３２、及び５３４は左ｚ－レベル５３０、右ｚ－レベル５３１、及びストロークｚ－レベル５２９と関連付けられる。不透明なストロークの場合、入力稜線５３３に対するストローキング方法から出力される稜線が、ｚ－レベル割当てと共に図５１（ｄ）に示される。図５１（ａ）の経路をストロークした視覚的な結果が図５１（ｂ）に示される。

【００９７】

尚、生成された左側ストローク稜線５３５は、入力として提供された左ｚ－レベル５３０を参照する左ｚ－レベル参照５３８を有すると共に、入力として提供されたストロークｚ－レベル５２９を参照する右ｚ－レベル参照５３９を有する。また、生成された右ストローク稜線５３６は、入力として提供されたストロークｚ－レベル５２９を参照する左ｚ－レベル参照５４０を有すると共に、入力として提供された右ｚ－レベル５３１を参照する右ｚ－レベル参照５４１を有する。元のストローク稜線５３３は破棄される。

【００９８】

３．４．８．透明なストロークに対するｚ－レベル割当て

上述のものに対する代替のｚ－レベル割当てについて説明する。割当ては、ある程度の透明度のストロークｚ－レベルを有する稜線の順序付けられた集合のストローキングに特に適している。ｚ－レベル割当てにおいて、生成された左側ストローク稜線には：

－ゼロの左ｚ－レベル参照、

－入力として受信されるストロークｚ－レベルを参照する右ｚ－レベル参照

が割り当てられ、生成された右側ストローク稜線には：

－入力として受信されるストロークｚ－レベルを参照する左ｚ－レベル参照、

－ゼロの右ｚ－レベル参照が割り当てられる。

【００９９】

このｚ－レベル割当てが使用される場合、元の稜線は破棄されない－ストローキングプロセスの出力は元の稜線の集合、生成された左側ストローク稜線の集合、及び生成された右側ストローク稜線の集合の和集合である。元の稜線には、入力左ｚ－レベル参照として提供された左ｚ－レベル参照と入力右ｚ－レベル参照として提供された右ｚ－レベル参照が割り当てられる。

【０１００】

図５１（ａ）は、上述のストローキング方法に対する入力稜線の集合を示す。稜線５３３、５３２、及び５３４は左ｚ－レベル５３０、右ｚ－レベル５３１、及びストロークｚ－レベル５２９と関連付けられる。透明なストロークの場合、入力稜線５３３に対するストローキング方法から出力された稜線は、ｚ－レベル割当てと共に図５１（ｅ）に示される。図５１（ａ）の経路をストロークした視覚的結果は図５１（ｃ）に示される。

【０１０１】

尚、生成された左側稜線５４２は、ヌルの左ｚ－レベル参照５４５と入力として提供されるストロークｚ－レベル５２９を参照する右ｚ－レベル参照５４６とを有する。元の稜線５３３は破棄されず、入力として提供される左ｚ－レベル５３０を参照する左ｚ－レベル参照５５０と入力として提供される右ｚ－レベル５３１を参照する右ｚ－レベル参照５４９とを有する。また、生成された右側稜線５４３は、入力として提供されるストロークｚ－レベル５２９を参照する左ｚ－レベル参照５４７とヌルであることを参照する右ｚ－レベル参照５４８とを有する。

【０１０２】

図６３は、ストロークされることが望まれ、複数の有向稜線６３０２、６３０４、６３０６、及び６３０８により形成される経路６３００を示す。図に示すように、頂点６３１０、６３１２、６３１４、６３１６、及び６３１８のお陰で有向稜線は延出／結合する。

【０１０３】

この例では、経路６３００は有向稜線６３０２から６３０８の各々と関連付けられたストローク幅に従ってストロークすることが望まれる。この点に関して、有向稜線６３０４及び６３０６は同じストローク幅及び色を有し、これらの２本の稜線は一括的にストロー

ク対象の元の経路６３００の個別の部分を定義する。

【０１０４】

図１８（ａ）及び図１８（ｂ）を参照して先に示したように、稜線の終了をストロークする際にアーティファクトが起こる可能性がある。特に、図６３に示すように、異なるストローク幅を有する稜線が交差する（例えば、頂点６３１２及び６３１６において）場合、稜線情報から正確な合成が行なわれるように全ての稜線に対して正確なストローキングが発生することが不可欠である。この点に関して、異なるストローク幅の結果、１つのストロークが他のストロークの上に合成される可能性があり、そうなると、不透明なストロークが使用される場合に実質的なエラーを提供する可能性がある。更に、透明なストロークが使用される場合には実質的ではないが重大なエラーが発生する恐れがある。経路６３００が正確にストロークされることを確実にするためには、経路６３００の個別の稜線部分により形成される個々のストローク経路が正確に再生されることを確実にすることが必要である。

【０１０５】

これは、ストローク幅が変化する元のストローク経路６３００の頂点を識別することによって最初に行なわれる。図６３から、ストローク幅が頂点６３１２及び６３１６において変化することが明らかである。他の全ての頂点では、ストローク幅は一定であり、作成対象の対応するストローク経路に対応する。更に、ストローク幅が変化する頂点を識別することにより、変化がない他の頂点（例えば、頂点６３１４）が正確にストロークされるようになる。頂点が適切に識別されると、元の経路６３００の個別の部分ごとに別々のストローク経路を作成できる。稜線６３０２により定義される第１の個別の部分に関して、ストローク経路６３２０及び６３２２は、それぞれ頂点６３１２から始まり、有向稜線６３０２の延長線６３３８上に位置する頂点に向かい、頂点６３１０から延出する。頂点６３１０を中心とする経路６３２０及び６３２２のストローキングは、図１８（ａ）において先に説明したように行なわれても良い。頂点６３１２を中心とする経路６３２０及び６３２２のストローキングは、図１８（ｂ）を参照して先に説明したように行なわれる。尚、図６３は、頂点６３１２（図１８（ｂ）に示す稜線２０７ａ及び２０７ｂ参照）から延出する経路６３２０及び６３２２を強調するものである。特に、ストローク経路６３２０及び６３２２は、元の経路６３００の対応する個別の部分（稜線６３０２により形成）を一括して包囲する。

【０１０６】

稜線６３０４及び６３０６と関連付けられたストローク経路に対して同様の手法が実行される。この場合、頂点６３１２及び６３１６により定義される経路の両端部は、図１８（ｂ）において先に説明したものに対応する構成を組み込む。稜線６３０８と関連付けられるストローク経路６３２８及び６３３０は、稜線６３０２に対するものと同様に作成される。重要なことは、これらの例では、ストローク経路が、ストローク幅が変化する経路上の点にあるものとして識別される頂点を通過する必要があることである。

【０１０７】

図６３の構成の拡張を図５１（ｂ）又は図５１（ｃ）を参照して理解することができるだろう。図５１（ｂ）又は図５１（ｃ）において、閉じた経路が示される。図６３の構成は、個別の部分を組み込む閉じた経路が、経路の内部がネガティブワインディング規則及び図５１（ａ）から図５１（ｅ）を参照して先に説明した塗りつぶし規則を使用して正確に塗りつぶしできるように、ストロークできるようにする。このようにして、前に説明した元の経路の稜線を左塗りつぶし及び右塗りつぶしを有するストローク経路を定義する稜線と取り換える方法は、閉じた経路の内部とストロークされた部分及びその内部に対する結果的な塗りつぶしレベルとを定義するのに使用することができる。

【０１０８】

３．４．９．ストローキング基本要素の変換

ストローキングがレンダ空間において行なわれる場合、ストローキングモジュールにより生成される線、ベジェ曲線、及び弧は、変換する必要がない。しかし、ストローキング

がオブジェクト空間で行なわれる場合、これらの基本要素は形状の元の稜線と同じ変換を受けなければならない。

【０１０９】

ストローク線及びベジェ線は、第３．２節で考察したように、線及びベジェ曲線を整形するために同様の方法で変換される。一方、ストロークエンドキャップ及び結合は、別の手順を踏まなければならない。これは、アフィン変換を円弧に適用した結果が汎用的な楕円弧であるからである。

【０１１０】

汎用型楕円弧は図１９に示される。下に有る楕円２１１は長軸２０８、短軸２０９、中心点２１５、及び回転角２１０により記述することができる。この楕円上にある楕円弧フラグメント２１２は、開始Ｙ座標２１３、終了Ｙ座標２１４、及びフラグメント方向（時計回り又は反時計回り）の更なる提供により完全に記述される。楕円フラグメント２１２の場合、方向は反時計回りである。

【０１１１】

下の円及び変換から楕円座標を取得するには幾つかの手法がある。１つの手法は、以下の式のｐを法とする２つの解を見つけ出すことである：

$$\tan 2\theta = (2ab + 2cd) / (a^2 - b^2 + c^2 - d^2)$$

ここで、

$$\Gamma = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

は変換行列の非平行移動部分である。

【０１１２】

正規化された長軸ベクトル及び短軸ベクトル（楕円の中心に対して且つ単位円に関して）は、？の２つの解に対して以下の式により得られる：

$$A = \Gamma \begin{vmatrix} \cos\theta \\ \sin\theta \end{vmatrix}$$

これらの正規化された軸（又はその反対）Ａ’は第１象限にあるだろう。事前に変換された円の半径を掛け合わせたこの軸ベクトルにより、楕円の長軸の長さ軸ベクトルが得られる。水平位置からの軸ベクトルの角度により楕円の回転が得られる。円の半径を掛け合わせた他方の軸ベクトルの長さにより楕円の短軸の長さが得られる。

【０１１３】

楕円の中心は、ユーザが提供した変換を円の中心に適用することによって見つけられる。楕円の開始座標及び終了座標は、同じ変換を元の円の開始座標及び終了座標に適用することによって見つけられる。

【０１１４】

ユーザ提供の変換は１次元で反転する。この場合、弧フラグメントの方向は変更されなければならない。変換が反転しているか否かを検査するためには、ベクトル

$$\begin{vmatrix} 1 \\ 0 \end{vmatrix} \text{ 及び } \begin{vmatrix} 0 \\ 1 \end{vmatrix}$$

を取り上げ、Ｇを適用し、結果のベクトルの水平位置からの回転を判別するだけで十分である。２つのベクトルは１８０度未満の角度を刻む。

【０１１５】

$$\Gamma = \begin{vmatrix} 1 \\ 0 \end{vmatrix}$$

がこの角度の反時計回り側にある場合、変換は逆である。反時計回り側にない場合、変換は逆にならない。

【０１１６】

３．５．フィルタリング

開始座標(Xs,Ys)及び終了座標(Xe,Ye)の観点から、曲線に対してのみは追加の制御座標(Xc,Yc)の観点から稜線を記述する座標を生成した後は、次のプロセスは明らかに出力フレームに影響しない稜線を破棄することである。これは、図４６に示されるフィルタリングプロセス４７０である。処理のこのステージでは、稜線の全ての座標は「レンダ空間」（変換済であるので）にあり、そのため、ディスプレイの境界座標と直接比較することができる。例えば、フレームが、幅ｗ及び高さｈを有し、フレームの左上が（０，０）、右下が（ｗ－１，ｈ－１）により記述される場合、直線稜線は以下の場合に安全にフィルタリング（すなわち、破棄）することができる：

（Ｘｓ＞＝ｗ）　ＡＮＤ　（Ｘｅ＞＝ｗ）　　／／右側に外れる場合

ＯＲ

（Ｙｓ＜０　）　ＡＮＤ　（Ｙｅ＜　０）　　／／上側に外れる場合

ＯＲ

（Ｙｓ＞＝ｈ）　ＡＮＤ　（Ｙｅ＞＝ｈ）　　／／下側に外れる場合

フィルタリングのために、グリフはグリフの右上端から左下端へと伸びる直線稜線であるかのように扱われる。

【０１１７】

曲線（追加の制御座標を有する）に対する同様の検査は以下の通りである：

（Ｘｓ＞＝ｗ）ＡＮＤ（Ｘｃ＞＝ｗ）ＡＮＤ（Ｘｅ＞＝ｗ）　／／右側に外れる場合

ＯＲ

（Ｙｓ＜　０）ＡＮＤ（Ｙｃ＜　０）ＡＮＤ（Ｙｅ＜　０）　／／上側に外れる場合

ＯＲ

（Ｙｓ＞＝ｈ）ＡＮＤ（Ｙｃ＞＝ｈ）ＡＮＤ（Ｙｅ＞＝ｈ）　／／下側に外れる場合

水平な直線稜線（すなわち、Ys == Ye）もフィルタリングされる。

尚、ディスプレイの左側から完全に外れた稜線は破棄されないーこのような稜線は表示出力に影響しない。

【０１１８】

３．６．稜線トラッキングパラメータの生成

図４６において、以下のプロセス４６９の目的は、稜線を修正された表現へと変換することであり、トラッキングパラメータの計算を含んでも良い。この修正された表現は、後続のモジュールによる処理により適しており、修正された表現は少なくとも以下のものを含む：

－開始座標(X,Y)、
－左ｚ－レベルへの参照及び／又は右ｚ－レベルへの参照、
－終了Ｙ座標、及び
－トラッキングパラメータ。

【０１１９】

ここでは、用語「トラッキングパラメータ」は、別途記載のない限り、Ｙの単位増分に対して稜線のＸ座標を増分的に計算できるようにする１つ以上のパラメータを記述するのに使用される。例えば、直線は開始座標(Xs,Ys)、終了Ｙ座標Ye及びＹ座標（すなわち、傾斜）における単位ステップ変更に対応するＸ座標における変更を表現するデルタＸ項により記述することができる。デルタＸ項は、稜線に対する単一のトラッキングパラメータであるだろう。

【０１２０】

ＴＣＩＥシステム６９９は、常に左上から右下にかけてレンダリングを行なうので、稜線（直線又は曲線）は開始Ｙ座標よりも大きい終了Ｙ座標を有するように指定する必要がある。これは、開始Ｙ座標と終了Ｙ座標とを比較し、必要に応じて開始座標と終了座標とを交換することによって確実にすることができる。開始座標と終了座標とが交換される場合、左ｚ－レベル参照及び右ｚ－レベル参照も交換されなければならない。

【０１２１】

３．６．１．直線稜線トラッキングパラメータの生成

直線はトラッキングパラメータとしてデルタＸ項を使用することで記述できることについて既に説明した。しかし、デルタＸは小数データを含むことが要求されることが多い。浮動小数点又は固定小数点形式を使用してデルタＸを表現する代替の方法として、より正確な表現は整数部分と傾斜のための残り部分を格納することである。

【０１２２】

ソフトウェアの実現例では、このプロセスはＣ言語コードで実現され、傾斜の２つの部分を計算しても良い：

Ｇｒａｄ＝（Ｘｅ－Ｘｓ）／（Ｙｅ－Ｙｓ）

Ｉｅｎｔ＝（Ｘｅ－Ｘｓ）％（Ｙｅ－Ｙｓ）

ここで、(Xs,Ys)は稜線の開始座標であり、(Xe,Ye)は稜線の終了座標である。

【０１２３】

適切な書式で記述される稜線に対して、上述のGrad項及びIent項が、開始座標(Xs,Ys)、終了Ｙ座標(Ye)、左ｚ－レベル参照及び右ｚ－レベル参照、及び左フラグ（ブール値）と共に提供される：

（Ｘｅ＜　Ｘｓ）の場合、左フラグ　＝　ＴＲＵＥ

（Ｘｅ＞＝Ｘｓ）の場合、　　＝　ＦＡＬＳＥ

３．６．２．２次ベジェ曲線トラッキングパラメータの生成

上述の図４６のステップ４６５は曲線ごとに以下のものを生成する：

－開始座標、

－制御座標、及び

－終了座標。

【０１２４】

曲線の座標は既に変換されている（ステップ４６８での座標の対応する順序付けられた集合の処理中）ので、曲線の座標は「レンダ空間」中の位置に関連する。図７は、起点（アニメーション６１のフレームの左上を表現）とＸ軸及びＹ軸（それぞれ６３及び６２）とにより示されるレンダ空間中の開始座標５８、制御座標５９、及び終了座標６０により表現されるベジェ曲線の例を示す。開始座標５８(Xs,Ys)、制御座標５９(Xc,Yc)、及び終了座標６０(Xe,Ye)から２次ベジェ曲線に対するトラッキングパラメータを生成するプロセスを図４１を参照して説明する。

【０１２５】

図４１において、第１のステップ４０３では、開始座標、制御座標、及び終了座標が同一直線上にある場合に関して検査を行なう。同一直線上にある場合、ステップ４０４において、制御座標は無視され、直線稜線に対して説明されたように稜線に対するトラッキングパラメータが計算される。

【０１２６】

次のステップ４０５では、２次ベジェ曲線をチェックし、Ｙにおいて単調であるか否かを確認する（すなわち、各Ｙ座標に対して曲線が通過する唯一の対応するＸ座標がある）。例えば、図７のベジェ曲線は明らかにＹにおいて単調ではない。このような非単調な２次ベジェ曲線はＹにおいて単調な２つの等価な曲線稜線を使用して記述される必要がある。以下の論理は、開始点(Xs,Ys)、制御点(Xc,Yc)、及び終了点(Xe,Ye)により定義される２次ベジェ曲線が２つの曲線稜線を必要とするか否かを判定するのに使用することができる：

ｓｉｇｎ（）をオペランドが正の場合にブール値ＴＲＵＥ（及びオペランドが負の場合ＦＡＬＳＥ）を戻す関数であるとすると：

（ｓｉｇｎ（Ｙｓ－Ｙｃ）＝ｓｉｇｎ（Ｙｓ－２Ｙｃ＋Ｙｅ））

ＡＮＤ

（｜Ｙｓ－Ｙｃ｜＜｜Ｙｓ－２Ｙｃ＋Ｙｅ｜）

である場合、ベジェは２つの曲線稜線を必要とする。

【０１２７】

２本の曲線稜線が必要であると判定される場合、元の非単調ベジェ上の点（X_{split},Y_{split}）は以下の公式を使用して計算される：

$X_{split}=\{X_s(Y_e-Y_c)^2+2X_c(Y_s-Y_c)(Y_e-Y_c)+X_e(Y_s-Y_c)^2\}/(Y_s-2Y_c+Y_e)^2$

$Y_{split}=(YsYe-Yc^2)/(Ys-2Yc+Ye)$

ステップ４０９において、元の非単調２次ベジェを表現する２つの単調曲線稜線が以下のものを使用して作成される：

－開始座標（＝元の非単調２次ベジェの開始座標(X_s,Y_s)）と、

－終了座標(X_{split},Y_{split})とを有する

－第１の曲線稜線

－開始座標(X_{split},Y_{split})と、

－終了座標（＝元の非単調２次ベジェの終了座標(X_e,Y_e)）とを有する

－第２の曲線稜線

稜線（直線又は曲線）が開始Ｙ座標よりも大きい終了Ｙ座標を有するように指定される必要があることが既に示されている。これは、ステップ４０６において各曲線の開始Ｙ座標と終了Ｙ座標とを比較し、ステップ４１０において必要に応じて開始座標と終了座標とを取り換えることによって確実にすることができる。開始座標と終了座標とが取り換えられる場合、左ｚ－レベル参照と右ｚ－レベル参照も取り換えられなければならない。

次のステップ４０７では、曲線の一部がレンダリング中のフレームの上方にある（すなわち、Ｙ＝０がフレームの最上部を表現する場合に０未満である開始Ｙ座標Y_sを有する）場合に、曲線を切り取るように動作する。最上の座標Y_{top}は以下の式により判定される：

$Y_{top}=max(0,Y_s)$

ここで、max(a,b)は、２つのオペラントａ及びｂのうちの大きい方を戻す関数である。

s

次の検査では、曲線ごとに２次多項フラグメント（ＱＰＦ）として曲線を記述する必要があるか否かを判定する。これは、後続の２次方程式を解く際に０での割り算を回避するのに必要である。開始点、終了点、及び制御点に基づくこの状況に対する簡易検査が以下に示すようにステップ４１３において行なわれる：

Ｙｓ＋Ｙｅ＝２Ｙｃ

は曲線がＱＰＦであることを暗示する。

【０１２８】

曲線がＱＰＦであると判定される場合、３つのトラッキングパラメータＡ、Ｃ、及びＤがステップ４１１において生成される。これらのトラッキングパラメータは以下に示すように曲線の開始点、終了点、及び制御点と最上座標Y_{top}から得られる：

与えられる中間値は以下の通りである：

$Ai=\{Ye(XsYe-XcYs)+Ys(XeYs-XcYe)\}/(Ye-Ys)^2$

$Ci=(4XcYc-2XsYe-2XeYs)/(Ye-Ys)^2$

$Di=(Xs+Xe-2Xc)/(Ye-Ys)^2$

Ａ、Ｃ、及びＤは以下のようにして得られる：

$A=Ai+CiY_{top}+DiY_{top}^2$

$C=Ci+Di(2Y_{top}+1)$

$D=2Di$

ステップ４１３において曲線がＱＰＦではないと判定される場合、トラッキングパラメータＡ、Ｂ、Ｃ、及びＤはステップ４０８において生成される。Ａ、Ｂ、Ｃ、及びＤは以下の計算により開始座標、終了座標、制御座標、及び最上座標Ytopから得られる：

$Xg=Xs-Xe$

$Yg=Ys-Ye$

$Xh=Xc-Xe$

$Yh=Yc-Ye$

$Y_{term}=Yg-2Yh$
$mix=XgYh-XhYg$
とすると：
$B=(Xg-2Xh)/Y_{term}^{2}$
$A=(2mix\times Yh)/Y_{term}^{2}+B(Ytop-Ye)+Xe$
$D=(Y_{term}\times mix^{2})/Y_{term}^{4}$
$C=(mix\times Yh^{2})/Y_{term}^{4}+D(Y_{top}-Ye)$

３．６．３．符号の判定

パラメータＡ、Ｂ、Ｃ、Ｄは後述のプロセスが以下の形式の式を使用して２次ベジェ曲線のＸ座標を追跡する際に使用する：

$x=A\pm 2\sqrt{C}$

トラッキングパラメータを準備する際に図４１の最終ステップ４１２は、平方根項がＡに加算されるべきか、あるいは、Ａから減算されるべきかを判定する。以下の擬似コードはＡ、Ｘｓ、Ｙｓ、Ｘｅ、Ｙｅ、及びＤに基づいて使用することができる。ここで、「ｓｉｇｎ＝　＋１」は平方根項の加算を示し、「ｓｉｇｎ＝　－１」は平方根項の減算を示す：

```
if (A > 0) sign = -1;
else if (A < 0) sign = +1;
  else
  {
    if (Ys < Ye) sign = -1;
    else sign = +1;
    if (D < 0) sign = -sign;
    if (Xs < Xe) sign = -sign;
  }
```

３．６．４．楕円弧トラッキングパラメータの生成

ステップ４６６（図４６を参照して説明）は弧ごとに以下のものを生成する：
－楕円長軸半径及び楕円短軸半径「ａ」及び「ｂ」、
－水平「θ」からの楕円角（軸ａの）、
－楕円中心(x,y)、
－弧の開始座標(xs,ys)及び終了座標(xe,ye)、及び
－左ｚ－レベル参照及び右ｚ－レベル参照

楕円はＹに対してのみ曲がり、Ｘに対してのみスケーリングされた円として記述することができる。このため、楕円トラッキングを簡略化するために、ＴＣＩＥシステム６９９は全ての楕円を変換された円として扱う。

【０１２９】

楕円の稜線を変換する際にトラッキングパラメータ４６９の計算が行なわれなければならない第１のステップは、楕円の下に位置する円を判定する。図２０はこのプロセスを実証する。楕円２１６は斜めスロット係数により直線の楕円２１７と関連する。この曲がりの大きさは線２１８の傾斜である。この線は２１６の軸ではないが、楕円の最低範囲から最高範囲まで延出する線である。

【０１３０】

この線の高さ（最低点から中心へかけての）は、以下の公式によって計算することができる：

$h=\sqrt{\{b^{2}\cos^{2}(\theta)+a^{2}\sin^{2}(\theta)\}}$

この線（最低点から中心にかけての）は以下の式により示される：

$w=\{\cos\theta\sin\theta(b^{2}-a^{2})\}/h$

従って、曲がり「ｅ」は：ｗ／ｈとなる。

【０１３１】

楕円２１７は高さｈを伴う円２１９へと変換される。これは単に換算係数：

$f=h^2/(ab)$

を楕円２１７に適用することで達成される。

【０１３２】

長さよりも幅の方が格段に大きい楕円（すなわち、ａ＞＞ｂ　ａｎｄ　？が小さい）に関しては、生成された円を変換することでは十分な解像度を得られず、結果的な楕円はむらのあるものとなる。この問題はスーパーサンプリングにより解決される－より半径の大きい円が追跡され、数本の線は変換前にまとめられる。

【０１３３】

スーパーサンプリングのための円のサイズを判定するための保守的な手法は、ａ＞２＊ｓｃａｌｅ＊ｂである間、換算係数（１に初期化）を２倍にし続けることである。この手法では、ａ＞＝ｂ及び－９０＜θ＜９０であると想定し、－４５＜θ＜４５の場合にのみ必要である。

【０１３４】

後述するように、円はBresenhamの円アルゴリズムの修正を使用して追跡される。リアルタイムオブジェクトレンダリングの性質により、Ｙにおいて単調増加するフラグメントのみが処理される。このため、図４６のステップ４６９により行なわれる第２のステップは、楕円の上側又は下側の頂点を横断する弧フラグメントを２つ以上のフラグメントへと分割することである。

【０１３５】

この分割の１つの可能な実現例を説明する。まず、楕円開始座標及び楕円終了座標が、曲がり「ｅ」とスケール「ｆ」を適用することにより円の開始座標及び終了座標へと変換される。次に、以下のアルゴリズムが実行される。このアルゴリズムは、開始点（Ｓ）及び終了点（Ｅ）と方向（ｄ）（時計回り又は反時計回り）とを受け入れる。更に入力されるのは円の最上点（Ｔ）及び最下点（Ｂ）であり、それぞれ(xc,y_c-h)及び(xc,y_c+h)により示される。アルゴリズムにより開始ｙ座標、終了ｙ座標、及び円「側面」（左又は右）から成るフラグメントのリストが生成される。

【０１３６】

図６７を考察すると、Ｓ及びＥのｘ座標が円の中心(xc,yc)のｘ座標に対してステップ６７０１において比較される。ＳとＥとが反対側に位置する場合、ステップ６７０２において、円の最上部又は最下部が横断されるか否かを確認するための検査が行なわれる。検査は、Ｓが左側に位置し且つｄが時計回りであるか、あるいは、Ｓが右側に位置し且つｄが反時計回りであるかを検査する。真の場合、円の最上部が横断され、出力は６７０３において示されるようになる（Ｔ－－＞Ｓ、Ｔ－－＞Ｅ）。円の側面がセグメントの最低点の側から続き、６７０４に示す場合、出力は（Ｔ－－＞Ｓ（Ｌ）、Ｔ－－＞Ｅ（Ｒ））になる。図６７において６７０４及び６７０５は可能な構成であり、左右対称に関連している。この考察の残りの部分では、全ての出力がセグメントの最低点の側面を継承するであろうことが理解される。ステップ６７０２において、円の最下部が横断されることを検査が示す場合、出力は６７０６において示され、可能な構成は６７０７及び６７０８において示される。ステップ６７０１において、Ｓ及びＥが円の同じ側で見出される場合、ステップ６７０９において判定されるようにＥがＳの上方にある場合、セグメントの終点及び方向はステップ６７１０において示されるように取り換えられる。これにより、更なる考察が簡略化される。

【０１３７】

円の同じ側にＳ及びＥがある場合を続けて考える。ステップ６７１１において、Ｓ及びＥが共にＴ及びＢを横断するか、あるいは、いずれも横断しないかが判定される。条件は、Ｓ及びＥが左側に位置し且つｄが時計回りであるか、あるいは、Ｓ及びＥが右側に位置し且つｄが反時計回りであるかである。真の場合、６７１２において示すように、出力（Ｓ－－＞Ｅ）及び可能な構成は６７１３及び６７１４において示される。偽の場合、ステップ６７１５において示すように、出力は（Ｔ－－＞Ｂ、Ｔ－－＞Ｓ、Ｅ－－＞Ｂ）である。この場合、Ｔ－－＞ＢセグメントはＳ及びＥの反対側に位置することになる。可能な

構成は６７１６及び６７１７において示される。

【０１３８】

３．６．５．グリフ稜線トラッキングパラメータの生成

グリフはＴＣＩＥシステムパイプライン６９９の下位の部分により特殊な稜線として扱われる。このため、ストローキング／モーフィング／変換モジュール６１６はグリフ記述を稜線記述へと変換しなければならない。

【０１３９】

ＴＣＩＥシステム６９９は、左稜線ではなく右稜線によりグリフを追跡する。従って、(Sx,Sy)がストローク／モーフ／変換モジュール６１６に提供されたグリフの左上座標である場合、Ｗはグリフの幅であり、Ｈはグリフの高さである：

－グリフ稜線の開始座標は(Sx+W-1,Sy)であり、

－左ｚ－レベルへの参照は「イン」ｚ－レベルに設定され、

－右ｚ－レベルへの参照は「アウト」ｚ－レベルに設定され、

－終了Ｙ座標はSy+Hである。

グリフ固有のトラッキングパラメータはグリフを表現するビットマップとグリフの幅とから成る。

【０１４０】

４．ソーティング

ストローキング／モーフィング／変換モジュール６１６により生成される稜線の集合は、図５６のソーティングモジュール６１７へと供給される。ソーティングモジュール６１７は、Ｙ軸の値を１番目に優先して昇順とし、Ｘ軸の値を２番目に優先して昇順としながらこれらの稜線が順序付けられる（すなわち、稜線は厳密にＹにおいてソートされ、Ｙ値を共有する稜線の部分集合ごとにＸにおいてソートされる）ように再度順序付けを行なう。レンダリングプロセスの後の各ステージが「画素順次式に」実行可能であるように、稜線はこのように順序付けられる必要がある。画素順次式とは、レンダリングが一度に１画素実行され、画面の左上の画素から開始し、画素の次の行へと下る前に画素の各行を左から右へ進み、最後に処理される画素は画面の右下にある画素である。画素順次式レンダリングは、従来のレンダリングシステムにより使用されるランダムアクセスフレーム格納を行なう必要がなく、メモリの使用を削減し、通常、レンダリング速度を向上させる。Ｘ軸及びＹ軸の方向及び優先順位に関してここで使用される規則は、上述のシステム６９９の動作に支障をきたすことなく、任意の代替の（ただし、直交の）規則と取り変えることができることは、当業者には明らかであるだろう。

【０１４１】

ソーティングモジュール６１７は、周知の基数ソートアルゴリズムの使用によりその機能を達成する。基数ソートアルゴリズムは、部分ソートのシーケンスを実行することによって要素の集合をソートする。部分ソートの各々は、要素ごとにソーティングインデックスの部分のみに対してソートすることによって行なわれる（各ステージで使用されるソーティングインデックスの部分は基数の対数の底２とビット長において等しい）。ソーティングインデックスのこの部分は、どのソーティングビンに要素を配置するかを選択するのに使用される。必要なソーティングビンの数は基数の数に等しい（すなわち、必要なソーティングビンの数はソーティングインデックスの各部分により表現可能な値の総数に等しい）。要素は各ソーティングビンに配置されるので、既存の順序はビン間にまたがっては破損するが、各ビン内では維持される。これにより、順次式部分ソートが前の部分ソートにより行なわれる有用なソーティング作業を取り消さないことが確実になる。各部分ソートの最後に、全ビンの内容が順番に連結され、ソート中の要素の集合は（新規の）単純なシーケンスへと復元される。第１の部分ソートは、ソーティングインデックスの最下位のビットと共に動作し、後続の部分ソートは連続的にソーティングインデックスのより重要な部分と共に動作する。

【０１４２】

基数ソートアルゴリズムは図４７（ａ）から図４７（ｆ）により例証される。

【０１４３】
図４７（ａ）は、ソーティングを必要とする１０個の要素のシーケンスを示す。各要素は２桁の８進ソーティングインデックスを有する。尚、８進の数字「０、１、２、３、４、５、６、７」は、数字との混同を回避するために記号「Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ」と取り換えられる。
【０１４４】
図４７（ｂ）は、基数８ソートの第１のステージの、図４７（ａ）において説明した要素のシーケンスへの適用を示す。８つのソートビンが使用され、要素はソーティングインデックスの第１の部分に従ってこれらのビンに配置される。この第１の部分ソートに対して使用されるソーティングインデックスの部分は、最下位の３ビット（基数Ｒソートは２Ｒの対数の底２にビット長が等しいソーティングインデックスの部分と共に動作することに注意；この場合は３ビット）である。ソートインデックスの最下位の３ビットは、最下位の８進数字に対応するので、第１の部分ソートは、ソーティングを必要とする要素のシーケンスの縦断として記述することができる。各要素は要素のソーティングインデックスの右端の８進数字に対応するソーティングビンに割り当てられる。
【０１４５】
図４７（ｃ）は、ソートの第２ステージを示す。このステージでは、８つのソーティングビンの全ての内容が連結され、第１のビンの内容で開始し、最後のビンまで順次処理が進められる。この結果、ソーティングを必要とする１０個の要素の新規のシーケンスが得られる。尚、この結果のシーケンスは右端の８進数字に対して正しく順序付けられる。最上位の８進数字に対して正しく順序付けられることはない。
【０１４６】
図４７（ｄ）はソートの第３ステージを示す。このステージは、入力シーケンスが図４７（ｃ）により記述されるシーケンスであり、使用されるソーティングインデックスの部分が２番目の３ビット、すなわち、左側の８進数字（最上位の８進数字）であることを除いては、第１ステージの繰り返しである。ソーティングインデックスのこの部分は最上位のビットを含むので、この部分ソーティングステージは必要とされる最後の１つである。このステージはソーティングを必要とする要素のシーケンスの縦断として記述することができる。各要素は要素のソーティングインデックスの左端の８進数字に対応するソーティングビンに割り当てられる。
【０１４７】
図４７（ｅ）はソートの第４ステージを示す。このステージはソーティングビンの内容が変更され、図４７（ｄ）に示すようになっていることを除いては、第２ステージの繰り返しである。このステージでは、８つのソーティングビンの全ての内容が連結され、第１のビンの内容で開始し、最後のビンまで順次処理が進められる。この結果、ソーティングを必要とする１０個の要素の新規のシーケンスが得られる。この結果のシーケンスは左端の（最上位の）８進数字に対して正しく順序付けられる。右端（最下位）の８進数字に対して正しく順序付けられることはない。尚、同一の左端８進数字を共有する新規の下位シーケンス内では、右端の８進数字の順序は正しい。結果的に、この新規のシーケンスはソーティングインデックスに従って正しく順序付けられる。シーケンスは明瞭さを期すために図４７（ｆ）に再度描画される。
【０１４８】
図４７（ａ）から図４７（ｆ）は、６ビット各々のソーティングインデックスを伴う１０個の要素のシーケンスの基数８ソートを例証する。基数ソートアルゴリズムの原理を２つの基数の他の整数乗、他のソートインデックスサイズ、及び他のシーケンス長にまで拡張することは、当業者には明らかであろう。尚、基数サイズが大きくなると、必要な部分ソートの数（必要な部分ソートの数は基数の対数の底２により除算をし、四捨五入したソートインデックスのビットサイズに等しい）が削減される。基数サイズが減少すると、必要なソーティングビンの数（必要なソーティングビンの数は基数の数に等しい）が削減される。ある特定のソーティングタスクに対する基数サイズの選択は、ソーティング速度と

ソーティングビンにより消費されるメモリとの間のトレードオフを伴うので、最適な基数サイズは、動作環境ごとに異なるであろう。

【０１４９】

基数ソートアルゴリズムをグラフィックスレンダリングシステムに適用する際に、使用されるソーティングインデックスは稜線の開始点のＹ座標とＸ座標とを連結することにより作成される。Ｙ座標の後にＸ座標を連結することにより、ソーティング動作が最初にＹにおいてソートを行ない、続いてＸにおいてソートを行なうことが確実になる。稜線がソートされた後、所望の画素順次式レンダリングはすぐに実行することができる。

【０１５０】

アンチエイリアシングを吸収するために、レンダリングシステム６１０は小画素の正確さを伴って稜線座標を記述しても良い。アンチエイリアシングについては本文書の他の箇所で詳細に説明するが、ここで言及しているのは、稜線座標がソーティング動作に影響するためである。ソーティングは、画素順次式レンダリングを促進するために行なわれるが、ソーティングインデックスの作成時に稜線座標の小画素の詳細は破棄される。小画素の正確さが小数部分の数字内に含まれる固定小数点表現で座標が表現される場合、小数部分の桁数に等しい桁数だけ座標を右シフトすることによって容易に達成される。

【０１５１】

基数ソートアルゴリズムの動作は、単純な最適化により幾分か加速することができる。この最適化には、ソーティングモジュール６１７の動作を前のモジュール（変換／モーフ／ストローキングモジュール６１６）の動作と部分的に重ねることを伴う。これらのモジュールの動作を厳密に分離する最適化されていないシステムでは、各モジュールの相互作用は以下のように記述することができる。変換／モーフ／ストローキングモジュール６１６は稜線の集合を作成する。これらの稜線は、変換／モーフ／ストローキングモジュール６１６にとって便利な順番を問わず作成され、その順序でシーケンスへと連結される。この稜線のシーケンスが完成すると、基数ソートの適用のために一緒にソーティングモジュール６１７へと供給される。上述の最適化を所有するシステムは、以下のテキストにおいて説明されるように各モジュールの相互作用を修正する。変換／モーフ／ストローキングモジュール６１６は、先に説明したように稜線を生成するが、連結は行なわない。稜線を連結する代わりに、第１の部分ソートの適用によりこれらを処理するソーティングモジュール６１７へと直接供給される（すなわち、稜線は最下位ビットに従ってソーティングビンへと分配される）。後続の部分ソート動作は、変換／モーフ／ストローキングモジュール６１６がソーティングモジュール６１７への稜線の供給を終了した後に行なわれる。基数ソートの第１の部分ソートは、ソーティングを必要とするオブジェクトの集合の生成中に「高速で」実行されても良く、後続の部分ソートはオブジェクトの集合が完成した後でしか実行できないので、この最適化が可能になる。

【０１５２】

５．稜線処理

図５６において明らかなように、表示リストコンパイラ６０８の出力は、画面上に所望の出力（単一のフレームに対して）を記述する稜線の集合である。レンダリングエンジン６１０は、これらの稜線の順序付けられた集合を入力とする。尚、表示リストコンパイラ６０８が連続的なフレームに対して稜線を準備する間に、レンダリングエンジン６１０が１つのフレームに対して稜線を処理できるようにするのに２重バッファ６０９を使用することができる。

【０１５３】

５．１．入出力

レンダリングエンジン６１０を介する表示データのフローは、稜線処理モジュール６１４の動作で開始する。この稜線処理モジュール６１４のデータフローパラメータは図２２（ａ）及び図２２（ｂ）において要約される。動画における現在のフレームは新規の稜線の順序付けられた集合２３２及び静止稜線の順序付けられた集合２３３により記述される。新規の稜線の順序付けられた集合２３２は、現在のフレームに対して表示リストコンパ

イラ６０８（先に説明）により生成された稜線から構成される。これらの新規の稜線２３２は、前のフレームに存在しないか、あるいは、前のフレームの後に新規の位置へと移動した表示オブジェクトを記述する。静止稜線の順序付けられた集合２３３は、前のフレーム上に新規の稜線として導入されてから存続する（移動していない）オブジェクトを記述する。

【０１５４】

フレームに対して全稜線（新規２３２及び静止２３３の双方）を処理し終わった後に、稜線処理２４２は出力としてｚ－レベル活動化モジュール６１３への入力として提供される交差メッセージ２４３の順序付けられた集合と、次のフレームのために稜線処理２３９により使用されることになる静止稜線２４１の順序付けられた集合とを生成しているだろう（すなわち、集合２４１は次のフレームの開始時には集合２３３になる）。

【０１５５】

稜線処理２４２は、現在のフレームの終了後出力表示に貢献していない稜線を破棄する役割も果たす。このような稜線は削除のためにマーク付けされる。この削除マークは外部のエンティティ（例えば、ドライバ）により配置されるため、本文書の範囲外である。各稜線が新規の稜線の順序付けられた集合又は静止稜線の順序付けられた集合から得られると、稜線処理２４２は稜線が削除のためにマーク付けされたか否かを確認する。マーク付けされている場合、稜線は次のフレーム２４１に関して静止稜線の順序付けられた集合へと転送されない。

【０１５６】

５．２．トップレベルの動作

単一のフレームに対する稜線処理２４２の要約が図２４に示される。レンダリング対象のフレームごとに、稜線処理２４２はディスプレイを下りながら走査線から走査線へと（行から行へ）繰り返し、新規の稜線の順序付けられた集合又は静止稜線の順序付けられた集合中の任意の稜線が現在の走査線と交差する位置を計算することによって動作する。当業者は、基本的なアルゴリズムを他の走査方向へと適用することができる。

【０１５７】

図２４において、ステップ２５５のフレームの処理開始時に、現在の走査線カウンタが０に初期化される（例えば、最上の走査線）。ステップ２５６において判定されるように、いずれかの稜線が現在の走査線と交差する場合、この稜線はステップ２５７において処理される。現在の走査線カウンタが最後の走査線の処理が終了したことを示す場合（ステップ２５８＝ｙｅｓ）、そのフレームに対する処理が終了する。示さない場合、ステップ２５９において現在の走査線カウンタが増分され、後続する走査線に対する稜線の処理が行なわれる。

【０１５８】

５．３．アクティブ稜線トラッキング

図２２（ｂ）は、現在の走査線２５７に対する稜線を処理する際の稜線処理２３９を介するデータのフローの概要を示す。現在の走査線と交差する（従って処理を必要とする）稜線はアクティブ稜線と呼ばれ、アクティブ稜線の順序付けられた集合２３６に走査順で格納される。新規の稜線又は静止稜線の開始座標が現在の走査線と交差する場合、稜線処理２３９は、新規の稜線２３４又は静止稜線２３５からアクティブ稜線を生成する。アクティブ稜線は新規の稜線又は静止稜線とは少々異なる書式を有する－例えば、開始座標(Xs,Ys)を有する代わりに、アクティブ稜線は走査線ごとに更新されるｃｕｒｒｅｎｔ＿Ｘフィールドを有する。ｃｕｒｒｅｎｔ＿Ｘフィールドは、稜線が現在の走査線と交差する箇所に対応する。アクティブ稜線が処理されるごとに、アクティブ稜線が後続の走査線と交差する場合、次の走査線２３８に対してアクティブ稜線の順序付けられた集合へとアクティブ稜線が走査順に配置される。

【０１５９】

図２３は、単一の走査線（図２４のステップ２５７に対応）に対する稜線処理２３９の動作をより詳細に示す。ディスプレイの第１の走査線をレンダリングする前に、アクティ

ブ稜線の順序付けられた集合２３６が空になる。この順序付けられた集合は、稜線処理２３９が新規の稜線の順序付けられた集合２３２中の新規の稜線又は静止稜線の順序付けられた集合中の静止稜線２３３の開始（終了）により交差される走査線を処理するまで空のままであろう。このような走査線に到達するまで、稜線処理２３９はすることがない。すなわち、アクティブ稜線の順序付けられた集合が空であり（ステップ２４４においてｙｅｓと判定）、残存する静止稜線又は新規の稜線がない（ステップ２４６＝ｙｅｓ）か、あるいは、全ての静止稜線又は新規の稜線がレーザ走査線まで開始しない（ステップ２４７＝ｙｅｓ）。

【０１６０】

アクティブ稜線の順序付けられた集合が空であり（ステップ２４４＝ｙｅｓ）、新規の稜線又は静止稜線が現在の走査線のいずれかの箇所で開始する（ステップ２４６＝ｎｏ、ステップ２４７＝ｙｅｓ）場合、新規の稜線又は静止稜線は対応する静止の順序付けられた集合又は新規の順序付けられた集合から削除され、ステップ２５０において処理のためのアクティブ稜線へと変換される。

【０１６１】

稜線処理２３９は、常に走査線に沿って走査順で（Ｘ軸増加）稜線を生成しなければならない。前の走査線を処理した結果、アクティブ稜線の順序付けられた集合に既にアクティブ稜線が存在しているかもしれない（すなわち、ステップ２４４＝ｎｏ）。既にアクティブ稜線が存在し、現在の走査線で開始する新規の稜線又は静止稜線（ステップ２４５＝ｎｏ）がある場合（ステップ２４８＝ｙｅｓ）、ステップ２４９において次の新規の／静止稜線のＸ座標は次のアクティブ稜線のＸ座標と比較され、どちらを次に処理すべきかを判定しなければならない。３つの稜線のソース（新規、静止、及びアクティブ）が走査順で格納されているために、処理する次の稜線は常に各ソース１４の次の利用可能な稜線を考慮することによって判定することができる（図２（ａ）参照）。

【０１６２】

５．４．稜線処理の例

図２（ａ）における例は、三角形状１５により部分的に覆われ閉塞される矩形形状１４を示す動画の第１フレームの所望の出力を示す。表示画素の現在の走査線１６は、この例の目的のために重畳される。現在の走査線の左端表示画素は１３である。この表示画素はＸ座標０に対応し、右側に後続する画素はＸ座標１に対応し、以下同様に対応する。

【０１６３】

稜線処理２３９が図２（ａ）に示すフレームに対して出力を生成する前に、表示リストコンパイラ６０８はこのフレームに対する所望の出力を記述する４本の新規の稜線を包含する新規の稜線の順序付けられた集合を生成しているだろう（水平稜線は塗りつぶされていることに注意）。これらの４本の稜線は図２（ｂ）の現在の走査線１６と交差することが示されている。稜線１９及び２２は矩形に対応するｚ－レベルをそれぞれ活動化及び非活動化する。稜線２０及び２１は三角形に対応するｚ－レベルをそれぞれ活動化及び非活動化する。

【０１６４】

稜線処理２３９が図２（ｂ）に示す現在の走査線を処理するまでに、稜線１９及び２２は新規の稜線の順序付けられた集合に存在し、稜線２０及び２１はアクティブ稜線の順序付けられた集合に存在するであろう。稜線２０及び２１は、前の（高い）走査線上で新規の稜線からアクティブ稜線へと既に変換されているだろう。

【０１６５】

図２３において概要が示されるように、稜線は現在の走査線に対して処理されることになる。要約すると、これは、例えば、稜線１９を処理することを伴う（最小のＸを有するため）。稜線１９は、まず、任意の交差メッセージが生成される前にアクティブ稜線へと変換される。稜線２０及び２１は次に処理され、最終的に稜線２２がアクティブ稜線へと変換されて処理されるだろう。この例では、現在の走査線に対する４本の稜線の処理は、図２（ｃ）に示す交差メッセージの集合を生成する。これらのメッセージの生成及び書式

については第５．８節で詳細に説明する。

【０１６６】

５．５．稜線のアクティブ稜線への変換及び稜線持続性

図２３に戻ると、処理する次の稜線が新規の稜線又は静止稜線の順序付けられた集合において始まる場合、ステップ２５０においてその稜線は最初に処理の都合上アクティブ稜線へと変換されなければならない。集合において始まらない場合、ステップ２５１において次のアクティブ稜線が得られる。処理する次の稜線が新規の稜線又は静止稜線であり、その新規の稜線又は静止稜線が削除のためにマーク付けされていない場合、新規の稜線又は静止稜線は次のフレームのために静止稜線２３７の順序付けられた集合へと配置されることになる。これにより、稜線は複数のフレームにわたって存続することができる。特定のフレーム上の稜線の削除を可能にするメカニズムを実現するのには種々の方法がある。１つの技術は、全ての稜線に識別フィールドを割り当てることであり、フレームごとにそのフレーム上で削除されるべき稜線に対応する識別フィールドの集合を提供することを考慮に入れている。代替の技術は、稜線により参照されるｚ－レベル内でフラグを提供し、全ての参照稜線が削除されることを示すことである。

【０１６７】

５．５．１．静止稜線の持続性

静止稜線バッファの動作の一例が図６８Ａから図６８Ｃ及び図６９Ａから図６９Ｃにおいて示される。図６８Ａから図６８Ｃは、アニメーションシーケンスの３つのフレーム６８００、６８０２、及び６８０４を示す。図６８Ａは、家６８０６から成る第１のフレーム６８００を示す。図６８Ｂは、前のフレーム６８００から家６８０６を保持するが、家６８０６の戸口に立つ人６８０６を追加する第２のフレーム６８０２を示す。図６８Ｃは第３のフレーム６８０４を示し、このフレームは家６８０６を保持するが、人が戸口から家６８０６の右側の位置６８１０へと移動している。

【０１６８】

図６９Ａから図６９Ｃは、それぞれ図６８Ａから図６８Ｃに示すフレームを生成するのに必要な処理動作の時系列を示す。図６８Ａに示すように、家６８０６を表現するオブジェクトは、まず、表示リストコンパイラ６０８へと入力され、新規の稜線バッファ６９０１を生成する。新規の稜線バッファ６９０１は、現在のフレーム６８００に追加される新規の稜線を含む。上述の図１６（ａ）は、家の形状の輪郭を描く座標と、これらの座標がどのように解釈され、新規の稜線バッファ６９０１により保持される稜線を生成するかを指し示す。入力の静止稜線バッファ６９０２は、前のフレームから保持される静止稜線を含む。バッファ６９０１及び６９０２は、レンダリングエンジン６１０を介してレンダリングされ、現在のフレーム６８００のレンダリングされた表示６９０３を出力する。また、図６９Ａは、出力の静止稜線バッファ６９０４を示す。出力の静止稜線バッファ６９０４は、後続のフレームのために保持される静止稜線を含む。

【０１６９】

図６９Ａにおいて、表示リストコンパイラ６０８は、変換、ストローキング、モーフィング、及びソーティングにより図１６（ａ）の座標を処理する。この例では、モーフィング又はストローキングは必要ないが、表示リストコンパイラ６０８は座標の正しい画面位置への変換及び稜線を画面順へとソートすることに時間を費やす。処理された稜線は、レンダリング動作に備えて新規の稜線バッファ６９０１へと配置される。入力の静止稜線バッファ６９０２は空であるが、これはアニメーションにおける最初のフレーム６８００であるからである（従って、前のフレームから稜線が保持されている可能性はない）。稜線バッファ６９０１及び６９０２の準備が整うと、表示リストコンパイラ６０８はこのフレーム６８００に対する動作を中止し、レンダリングエンジン６１０がこのフレーム６８００に対する動作を開始する。レンダリングエンジン６１０は、稜線バッファ６９０１及び６９０２からの各稜線をマージし、ディスプレイ６９０３へとレンダリングし、後続のフレームに対して保持される稜線を出力の静止稜線バッファ６９０４へと出力する。アニメーションのこのフレーム６８００の処理が完了する。

【０１７０】
図６８Ｂ及び図６８Ｃにおいて、参照６８０８及び６８１０は、第２及び第３のアニメーションフレーム６８０２及び６８０４へとレンダリングされる人の形状の輪郭を描く座標をそれぞれ参照する。
【０１７１】
図６９Ｂにおいて、アニメーションの第２のフレーム６８０２の処理が行なわれる。図６９Ｂの入力の静止稜線バッファ６９０２は図６９Ａの出力稜線バッファ６９０４と同じである。これが可能であるのは、入力及び出力の静止稜線バッファがフレーム間で「ピンポン方式で」取り換えられるからである。図６９Ｂの出力の静止稜線バッファ６９０４は「ピンポン」取換えの直後にクリアされる。
【０１７２】
表示リストコンパイラ６０８は、このフレーム６８０２に対して追加される新規の稜線の座標６８０８を処理し、処理された稜線を新規の稜線バッファ６９０１に配置する。家６８０６の稜線は表示リストコンパイラ６０８による処理を必要としないが、既に処理された形態の前のフレームから保持されないからである。静止稜線技術が必要な処理時間を削減するのはこのためである。
【０１７３】
前のフレームの処理におけるのと同様に、表示リストコンパイラ６０８は停止させられ、レンダリングエンジン６１０がこのフレーム６８０２の処理を開始する。稜線バッファ６９０１及び６９０２の内容はマージされ、ディスプレイ６９０３へとレンダリングされ、次のフレーム６８０４に対して保持される稜線は出力稜線バッファ６９０４に格納される。
【０１７４】
図６９Ｃにおいて、アニメーションの第３のフレーム６８０４の処理が行なわれる。表示リストコンパイラ６０８は、このフレーム６８０４に対して追加される新規の稜線の座標６８１０を処理し、処理された稜線を新規の稜線バッファ６９０１に配置する。前のフレーム６８０２と同様に、家６８０６の稜線は表示リストコンパイラ６０８による処理を必要としない。既に処理された形態で前のフレームから保持されるからである。人の形状の稜線は、人の形状６８１０が移動して前のフレームに比べて大きくなっているので処理を必要としない。このため、表示リストコンパイラ６０８は座標６８１０を処理し、人の形状の処理された稜線を新規の稜線バッファ６９０１に配置する。
【０１７５】
先に述べたように、表示リストコンパイラ６０８は停止させられ、レンダリングエンジン６１０がこのフレーム６８０４の処理を開始する。稜線バッファ６９０１及び６９０２の内容はマージされ、ディスプレイ６９０３へとレンダリングされ、次のフレームに対して保持される稜線は出力静止稜線バッファ６９０４に格納される。尚、家のオブジェクト６８０６は、アニメーションの第４のフレームに含めるために変更なしに保持されるものと想定される。変更なしに保持されない場合、家の形状は出力の静止稜線バッファ６９０４に含まれないであろう。
【０１７６】
５．６．アクティブ稜線処理
ステップ２５２における次のアクティブ稜線の処理は、後続のモジュールｚ－レベル活動化モジュール６１３に渡す１つ以上の交差メッセージを生成することを伴う。各メッセージはアクティブ稜線の左ｚ－レベル参照及び右ｚ－レベル参照と共にｃｕｒｒｅｎｔ－Ｘ座標を含む。
【０１７７】
アクティブ稜線が、少なくとも次の走査線まで続いている場合（図２３においてステップ２５３＝ｎｏ）、ステップ２５４においてアクティブ稜線は次の走査線の間の処理に対するアクティブ稜線の順序付けられた集合に追加される。
【０１７８】

５．７．稜線のトラッキング

図２３のステップ２５２において行なわれるように、走査線から走査線へと稜線のＸ座標を追跡するプロセスは、従来技術において「稜線トラッキング」と呼ばれることが多い。説明したシステム６９９では、稜線は直線、２次ベジェ曲線、楕円弧、又は２次多項フラグメントとすることができる。

【０１７９】

５．７．１．直線のトラッキング

Bresenhamの線アルゴリズムは、直線を描く方法として従来技術において周知である。ステップ２５２において、稜線処理２３９は、修正されたBresenhamのアルゴリズムを使用して走査線ごとにアクティブ直線稜線のＸ座標を計算する。図５のＣ言語ソースコードは、開始座標(Xs,Ys)から終了座標(Xe,Ye)までの直線を追跡（及び描画）するのに使用される修正されたBresenhamのアルゴリズムの一例を示す。この例は３つの事前に計算された値：ｅｒｒ、ｄｅｌｔａ＿ｅｒｒ１、及びｄｅｌｔａ＿ｅｒｒ２に基づいてYsからYeの範囲でＹ座標ごとに増分的に計算される線のＸ座標を示す。

【０１８０】

直線である稜線に対するデータが表示リストコンパイラ６０８により生成される場合（第３．６．１節－直線稜線のトラッキングパラメータの生成－参照）、少なくとも以下のものを含むように生成される：

－開始座標(Xs,Ys)、
－終了Ｙ座標(Ye)、
－整数傾斜項（ｇｒａｄ）、
－残り傾斜項（ｉｅｎｔ）、
－左フラグ、及び
－左ｚ－レベル及び／又は右ｚ－レベル

ステップ２５０において直線稜線がアクティブ直線稜線へと変換されると、ｉｅｎｔ項がｅｒｒ、ｄｅｌｔａ＿ｅｒｒ１、及びｄｅｌｔａ＿ｅｒｒ２により置き換えられる。ｅｒｒ、ｄｅｌｔａ＿ｅｒｒ１、及びｄｅｌｔａ＿ｅｒｒ２は直線稜線のデータから以下に示すように計算される：

ｔｙ＝Ｙｅ－Ｙｓ

とすると：

ｅｒｒ＝（２×ｉｅｎｔ）－ｔｙ
ｄｅｌｔａ＿ｅｒｒ１＝２×ｉｅｎｔ
ｄｅｌｔａ＿ｅｒｒ２＝２×（ｉｅｎｔ－ｔｙ）

走査線ごとに、アクティブ稜線に対する新規のｃｕｒｒｅｎｔ－Ｘ座標が、修正されたBresenhamのアルゴリズムを使用してｇｒａｄ、ｅｒｒ、ｄｅｌｔａ＿ｅｒｒ１、及びｄｅｌａｔａ＿ｅｒｒ２パラメータからステップ２５２に従って以下に示すように計算することができる：

ｅｒｒが０未満の場合：
ｅｒｒ＝ｅｒｒ＋ｄｅｌｔａ＿ｅｒｒ１
ｃｕｒｒｅｎｔ＿Ｘ＝ｃｕｒｒｅｎｔ＿Ｘ＋ｇｒａｄ
０以上の場合
ｅｒｒ＝ｅｒｒ＋ｄｅｌｔａ＿ｅｒｒ２
ｃｕｒｒｅｎｔ＿Ｘ＝ｃｕｒｒｅｎｔ＿Ｘ＋ｇｒａｄ
左フラグがＴＲＵＥの場合：
ｃｕｒｒｅｎｔ＿Ｘ＝ｃｕｒｒｅｎｔ＿Ｘ－１
ＴＲＵＥでない場合
ｃｕｒｒｅｎｔ＿Ｘ＝ｃｕｒｒｅｎｔ＿Ｘ＋１

５．７．２．２次ベジェのトラッキング

２次ベジェは、図７の例に示すように開始点、終了点、及び制御点により記述される。２次ベジェ曲線である稜線に対するデータが表示リストコンパイラ６０８により処理され

る場合（第３．６．２節－２次ベジェ曲線トラッキングパラメーター参照）、２次ベジェ稜線記述は、少なくとも以下のものを含むように再度書式化され稜線処理２３９に提示される：
－開始座標(Xs,Ys)、
－終了Ｙ座標(Ye)、
－トラッキングパラメータＡ、Ｂ、Ｃ、及びＤ、
－符号フラグ（＋１又は－１）、及び
－左ｚ－レベル及び／又は右ｚ－レベルへの参照
図２３のステップ２５２において、走査線ごとに、ベジェ曲線のｃｕｒｒｅｎｔ＿Ｘ位置が、以下に示すようにＡ、Ｂ、Ｃ、及びＤに基づいて再計算される：
Ａ＝Ａ＋Ｂ
Ｃ＝Ｃ＋Ｄ
符号フラグが＋１の場合：
Ｃｕｒｒｅｎｔ＿ｘ＝Ａ＋２√Ｃ
＋１でない場合：
Ｃｕｒｒｅｎｔ＿ｘ＝Ａ－２√Ｃ
５．７．３．２次多項フラグメントのトラッキング
表示リストコンパイラ６０８（第３．６．２節－２次ベジェ曲線トラッキングパラメーターの生成参照）は、少なくとも以下のものを含むように２次多項フラグメントに対する稜線記述を生成する：
－開始座標(Xs,Ys)、
－終了Ｙ座標(Ye)、
－トラッキングパラメータＡ、Ｃ、及びＤ、及び
－左ｚ－レベル及び／又は右ｚ－レベルへの参照
ステップ２５２において、走査線ごとに、２次多項フラグメントのｃｕｒｒｅｎｔ＿Ｘ位置が、以下に示すようにＡ、Ｃ、及びＤに基づいて再計算される：
Ａ＝Ａ＋Ｃ
Ｃ＝Ｃ＋Ｄ
Ｃｕｒｒｅｎｔ＿Ｘ＝Ａ
５．７．４．楕円弧のトラッキング
表示リストコンパイラ６０８は少なくとも以下のものを含む楕円弧に対して稜線記述を生成する：
－等価な円中心(Cx,Cy)
－等価な円半径「Ｒ」、
－等価な円開始座標(Sx,Sy)、
－開始誤差値「Ｅ」、
－円から楕円への変換（曲がり「ｅ」＋スケーリング「ｆ」）、
－終了Ｙ座標「Ey」、及び
－左向き又は右向き
図２３のステップ２５０がこのデータをアクティブ楕円弧へと変換する場合、現在の座標、現在のエラー値、及び現在のモード（後述）がこのデータにより初期化される。
トラッキングアルゴリズムは、円の中心に対する座標を追跡する。このため、現在の座標は開始座標－等価な円中心に設定される。現在のエラー値は表示リストコンパイラ６０８により提供されるエラー値に設定される。モードは現在走査中の８分角を追跡するのに使用され、ステップ２５０において提供された開始座標に基づいて初期化される。
尚、いずれのステージにおいても、絶対楕円座標は以下の公式を使用して相対的な現在座標(c_x,c_y)から得ることができる：
（SE_x，SE_y）　　（$fc_x+ec_y+C_x,c_y+C_y$）
各円は、Bresenhamの円アルゴリズムを使用して追跡され、現在座標及びエラー値が繰り返し更新される。従来技術において周知のこのアルゴリズムは、円の右上の８分角（図

１４の８分角１２０）のみを追跡し、反映のシーケンスを使用して他の７つの８分角を生成する。更なる詳細については、「Computer Graphics: Principles and Practice」（Foley & Van Dam Second Edition）の第３．３節Scan Converting Circlesにおいて見出すことができるだろう。

【０１８１】

５．７．５．グリフのトラッキング

アクティブグリフオブジェクトは、グリフビットマップへの現在のポインタを維持する。このポインタは、グリフ稜線が追跡されるにつれ行ごとに増分される。グリフの左側稜線は垂直であるので、現在のＸ位置は稜線が追跡されるに従って更新される必要はない。

【０１８２】

５．８．アンチエイリアシング及び交差メッセージ生成

アンチエイリアシングは、追加の（小画素）解像度を使用して稜線を追跡することによって達成される。従来技術においては周知であるが、この追加の解像度は各画素のどの部分がグラフィカル基本要素により占められるかを判定するのに使用することができる。この部分（一部の従来技術では画素カバレッジと呼ばれる）は、グラフィカル基本要素の各出力画素の色への貢献を弱めるのに使用することができ、レンダリング結果が大幅に改善される。例えば、稜線の座標において、Ｘ及びＹの双方で解像度が２ビット追加することができるので、Ｘ及びＹの双方において解像度が４倍となる。

【０１８３】

稜線処理２３９が追加の解像度をどのように使用するかを図１（ａ）から図１（ｃ）を参照して説明する。図１（ａ）は、アクティブ稜線０が現在レンダリング中の表示画素の走査線１に入るのを示す。図１（ｂ）において、この走査線は稜線データのＹ座標における追加の２ビットにより提供される追加の小画素解像度を表現する４つの小画素線（ｓｐｅｌ線）３、４、５、及び６へと分割される。稜線のＸ位置は、先に説明した稜線トラッキング技術のうちの適切な１つを使用して、４本のｓｐｅｌ線の各々に対して小画素の精度で繰り返し判定される。追跡中の稜線が少なくとも１本のｓｐｅｌ線と交差する表示画素ごとに、稜線がその画素にどのように影響するかを４ｘ４ビットマスクで表現するものが生成される。このようなビットマスクは、従来技術ではしばしばＡ－ｂｕｆｆｅｒと呼ばれる。このＡ－ｂｕｆｆｅｒが最初に記述されたのはCarpenter, L.による「The A?buffer, an Anti?aliased Hidden Surface Method」（Computer Graphics, Vol. 18, No. 3, Jul. 1984）（以降、「Carpenter」）である。Ａ－ｂｕｆｆｅｒは、表示画素内のどのｓｐｅｌが稜線により影響されるかを示すために使用され、稜線処理モジュール６１４からｚ－レベル活動化モジュール６１３へと渡される交差メッセージに含まれる。この例では、図１（ｃ）に示すように３つのＡ－ｂｕｆｆｅｒ、すなわち、Ａ－ｂｕｆｆｅｒ８、１１、及び１２が生成される。これらのＡ－ｂｕｆｆｅｒは、走査線画素Ｘ＝２、Ｘ＝３、及びＸ＝４に対応する。図１（ｃ）に示すように、ｓｐｅｌ９は稜線により影響されないが、ｓｐｅｌ１０は稜線により影響される。

【０１８４】

交差メッセージ及び関連するＡ－ｂｕｆｆｅｒの生成は、図２３のステップ２５２においてアクティブ稜線に対して行なわれる。このステップについては図３を参照して詳細に説明する。

【０１８５】

図３の第１のステップ３５において、ｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅ値が０に初期化される。これは、現在の走査線の最上のｓｐｅｌ線が最初に処理されることを示す。Ａ－ｂｕｆｆｅｒが空として初期化され、ｃｕｒｒｅｎｔ＿ｐｉｘｅｌ値は、アクティブ稜線が現在の走査線に入る表示画素に対応するＸ値を伴って初期化される。このＸ値はアクティブ稜線のｃｕｒｒｅｎｔ＿Ｘ座標の非小数部分である。

【０１８６】

次にステップ３６において、処理中のアクティブ稜線のｃｕｒｒｅｎｔ＿Ｘ座標はｃｕｒｒｅｎｔ＿ｐｉｘｅｌ値と比較され、ｃｕｒｒｅｎｔ＿Ｘがｃｕｒｒｅｎｔ＿ｐｉｘｅ

ｌにより表現される表示画素内にあるか否かが確認される。これは、ステップ３５の結果としての最初の繰り返しでは真である。

【０１８７】

次のステップ３９はｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅに対応するＡ－ｂｕｆｆｅｒの単一の行を修正するように動作する。ここで、Ａ－ｂｕｆｆｅｒの最上行はｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅ＝０に対応する。ｃｕｒｒｅｎｔ＿Ｘの小数データは、ｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅに対応するＡ－ｂｕｆｆｅｒの行に幾つのｓｐｅｌが「設定される」べきかを判定するのに使用される。小画素の精度の２ビットが、例えば、ｃｕｒｒｅｎｔ＿Ｘにより提供される場合、Ａ－ｂｕｆｆｅｒ８、１１、及び１２に示すように１行当たり４つのｓｐｅｌを表現するＡ－ｂｕｆｆｅｒを生成することができる。表５は、ｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅに対応するＡ－ｂｕｆｆｅｒの行をどのように設定すべきかを判定するのに２ビットの小数データをどのように使用することができるかを示す。

【０１８８】

表５

ｃｕｒｒｅｎｔ＿Ｘの小数ビット	Ａ－ｂｕｆｆｅｒの行			
	左端ｓｐｅｌ	第２ｓｐｅｌ	第３ｓｐｅｌ	右端ｓｐｅｌ
００ｂ	１	１	１	１
０１ｂ	０	１	１	１
１０ｂ	０	０	１	１
１１ｂ	０	０	０	１

図１（ｂ）の例に戻ると、最上ｓｐｅｌ線３に対する処理中に、（ｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅ＝０）、アクティブ稜線０に対するｃｕｒｒｅｎｔ＿Ｘ値が非小数成分４（１００ｂ）と小数成分０．７５（小数ビットは１１ｂ）とを有する。この場合、Ａ－ｂｕｆｆｅｒ１２の最上行の右端ｓｐｅｌのみが表５に従って設定される。

【０１８９】

Ａ－ｂｕｆｆｅｒの修正後、図３に示すようにステップ４０において以下のｓｐｅｌ線４と交差するアクティブ稜線に対応する新規のｃｕｒｒｅｎｔ＿Ｘが計算される。これは、先に説明したトラッキング技術のうちの１つをアクティブ稜線に対するデータに適用することにより達成される。ステップ４１においてｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅは増分され、処理すべきｓｐｅｌ線が更に存在する場合（ステップ４２＝ｎｏ）、処理はアクティブ稜線のｃｕｒｒｅｎｔ＿Ｘをｃｕｒｒｅｎｔ＿ｐｉｘｅｌと比較するステップ３６へと戻る。

【０１９０】

アクティブ稜線がｃｕｒｒｅｎｔ＿ｐｉｘｅｌにより示されるものと異なる表示画素内でｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅと交差することをｃｕｒｒｅｎｔ＿Ｘが示す場合、少なくとも以下のものを含む交差メッセージを生成するステップ３７が行なわれる：

－Ｘ座標＝ｃｕｒｒｅｎｔ＿ｐｉｘｅｌ
－Ａ－ｂｕｆｆｅｒ
－活動化されたｚ－レベル参照（アクティブ稜線の左ｚ－レベル参照からコピー）
－非活動化されたｚ－レベル参照（アクティブ稜線の右ｚ－レベル参照からコピー）

尚、ｚ－レベル参照はヌルの塗りつぶしを参照することができる。

【０１９１】

生成された交差メッセージは、出力としてソーティング手段（例えば、ソートバッファ

）を介してｚ－レベル活動化モジュール６１３に渡される。ソーティング手段は、ｚ－レベル活動化モジュール６１３が走査順（すなわち、Ｘ座標の昇順にソート）交差メッセージを受信することを確実にするのに必要である。

【０１９２】

Ａ－ｂｕｆｆｅｒは空に再初期化され、ｃｕｒｒｅｎｔ＿ｐｉｘｅｌはアクティブ稜線に対するｃｕｒｒｅｎｔ＿Ｘ値（０）の非小数部分に設定される。続いてＡ－ｂｕｆｆｅｒは、先に説明したようにＡ－ｂｕｆｆｅｒ１１に修正され、ｃｕｒｒｅｎｔ＿ｐｉｘｅｌのｃｕｒｒｅｎｔ＿ｓｐｅｌ＿ｌｉｎｅ５に対するアクティブ稜線の影響を示す。

【０１９３】

処理は残りのｓｐｅｌ線６に対して継続されるため、Ａ－ｂｕｆｆｅｒ１１を含む交差メッセージが出力される。

【０１９４】

全てのｓｐｅｌ線が処理されると（ステップ４２＝ｙｅｓ）、先に説明したようにステップ４３においてＡ－ｂｕｆｆｅｒ８を含む最終交差メッセージが生成され、出力される。

【０１９５】

５．８．１．グリフに対する交差メッセージの生成

レンダリングされたグリフの各行はバイト幅（すなわち、８画素幅）ブロックへと分割され、このブロックが順番に処理される。各バイトは２５６エントリルックアップテーブル（ＬＵＴ）への索引として使用される。各エントリは３２ビットを占める。ＬＵＴの単一のエントリは８画素行交差メッセージにおけるどの画素が生成されなければならないかを識別する。

【０１９６】

各エントリはニブル単位で処理される－最初のニブルはバイトにより生成される交差の数を記録し、後続する各ニブルは領域の開始に対する交差の位置を記録する。８つの交差を記録するが位置を記録しない特殊なエントリは、バイトが各画素に交差を有する独特な場合に使用される。

【０１９７】

図２６において、幅１６高さ２のビットマップ２６５を伴う例が示される。２進表現２６５は{0xD6,0x54,0x99,0xCC}であるだろう（尚、各バイトのビット７は対応する画素ブロックにおいて左端の画素を指し示す）。このビットマップの第１バイトを取り上げ、ＬＵＴ２６６においてその符号なし値を参照すると、位置０、２、３、４、５、及び７に６つの交差があることが分かる。ＬＵＴ２６６が画素座標(100,100)に配置されるものとすると、交差メッセージは(100,100)、(102,100)、(103,100)、(104,100)、(105,100)、(107,100)に対して生成されなければならないことを意味するであろう。８つの交差メッセージを生成するバイトをこの数字において観察することができる。

【０１９８】

ＬＵＴ２６６が索引付けされ、バイトに対する交差の集合が得られると、余分な交差が挿入されるか、あるいは、不正確な交差が削除されても良い。これは、ビットマップの各行の開始で０に初期化されるフラグにより管理される。このビットがクリアされると、交差は修正されていない。ビットが設定されると、画素ブロックにおける左端位置（「ゼロ交差」）に対応する交差が存在しない場合、１つが挿入される。逆に、ゼロ交差が存在する場合、削除される。

【０１９９】

フラグの状態は、行の各バイトの処理の終了時に更新される。奇数個の交差メッセージはフラグを反転させるが、偶数個の交差メッセージはフラグの現在値を維持する。フラグが行の終了で設定される場合、最終交差メッセージが出力されて行を正しく終了させる。このメッセージの位置は現在の画素ブロックの１つ右の画素ブロックの開始にある。

【０２００】

図２６に戻ると、ビットマップにおいて４バイトにまたがってこのフラグの状態を追跡

することができる。第１行の開始において、フラグは０に初期化される。第１バイトは偶数の交差に分解するので、フラグはこのバイトの終了では切り換えられない。これに対し、第２のバイトは奇数の交差に分解する。結果として、フラグは１に切り換えられる。第２のバイトは行の最後であるので、行を終了させるために最終交差メッセージが生成される。

【０２０１】

第２の行に移動し、フラグが０に初期化される。第３のバイトは奇数の交差メッセージに分解し、フラグはこのバイトの終了において１に設定される。第４のバイトは、０、１、２、３、４、５、６、及び７において８つの交差メッセージに分解する。しかし、フラグは設定される。ゼロ交差メッセージが存在するので、このメッセージは削除され、結果として７つの交差メッセージが出力される。これは奇数の交差であるので、フラグは再度０に切り換えられる。処理は行の終了にあるのでフラグは設定されず、最終交差メッセージは出力されない。

【０２０２】

いずれにしても交差メッセージは、全ての小画素が－１に設定されているＳバッファを備えている。結果として、グリフは奇偶ワインディング規則を使用して常にレンダリングされるべきである。

【０２０３】

５．８．１．１．グリフ交差メッセージの生成及びアンチエイリアシング

図６６は図２６に示すのと同じ走査線を示す。この走査線上で単一の画素６６０１を考慮すると、交差メッセージ６６０２が生成される。交差メッセージ６６０２は全ての小画素が－１に設定されている。この交差メッセージにより画素全体６６０３が出力される。要するに、グリフはアンチエイリアシングされず、小画素は考慮されない。これは、「ゼロ交差」フラグを考慮して追加されるものを含むグリフに対して生成される全ての交差メッセージに関して真実である。

【０２０４】

５．９．交差メッセージの例

図２（ｃ）に示す例を継続すると、図２（ｂ）に示すように、アクティブ稜線に対して稜線処理モジュール６１４により生成される交差メッセージを示す。この例では４ｘ４アンチエイリアシングが使用される。４ｘ４アンチエイリアシングでは、アクティブ稜線座標が表示画素内の小画素（ｓｐｅｌ）に対応する追加の２ビットの解像度を伴って記述される。まず稜線１９の処理が行なわれ、対応する交差メッセージ２３が生成されるであろう。アクティブ稜線は矩形に対する塗りつぶしを記述する左ｚ－レベル参照を持っているので、生成された交差メッセージ２３の活動化されたｚ－レベル参照は矩形に対する塗りつぶしを記述する活動化されたｚ－レベル参照を有するであろう。

【０２０５】

次にアクティブ稜線２０の処理が行なわれる。モジュール６１４（図３で概略を示す）は、アクティブ稜線２０に対する４つのｓｐｅｌ線のうちの最初の３つを繰り返す際に交差メッセージ２５が生成される。アクティブ稜線は異なる画素上の第４（最下部のｓｐｅｌ線）と交差するので、これに対して異なる交差メッセージが生成される必要があるので、交差メッセージ２４が次に生成される。これらの交差メッセージは、三角形に対する塗りつぶしを記述する活動化されたｚ－レベル参照を活動化している。

【０２０６】

同様に、交差メッセージ２６及び２７がアクティブ稜線２１に対して生成され、これらのメッセージが三角形に対する塗りつぶしへの非活動化されたｚ－レベル参照（元のアクティブ稜線２１の右ｚ－レベル参照からコピー）を有する。交差メッセージ２８が稜線２２から最後に生成される。このメッセージは矩形に対する塗りつぶしへの非活動化されたｚ－レベル参照を有する。

【０２０７】

５．９．１．別の例

図７０はＳ－ｂｕｆｆｅｒがゼロ交差メッセージの受信を伴って更新されるのを示す更なる例である。最初に、走査線における画素＃１に関して、稜線交差はないので、この画素に対して交差メッセージ又はＳ－ｂｕｆｆｅｒ$_n$は存在しない。このように、画素＃１に対してＳ－ｂｕｆｆｅｒ$_n$は０を伴って設定される。このＳ－ｂｕｆｆｅｒ$_n$の右端列が、次の画素＃２に対するＳ－ｂｕｆｆｅｒ$_{n-1}$の第１（すなわち、左端）の列にコピーされる。この列はＳ－ｂｕｆｆｅｒ$_{n-1}$の列をまたいでコピーされ、この場合、バッファもゼロを伴って設定される。図示するように、稜線が画素＃２と交差し、下位線の交差（交差４、２又は１本の下位線）に対する説明した規則に従って、画素＃２に対するゼロ交差メッセージが判定される。メッセージはより低い右側の１／４において＋１が配置されるので、走査線と交差する稜線が補足される。画素＃２に関して、ゼロ交差メッセージ及びＳ－ｂｕｆｆｅｒ$_{n-1}$が合計され、その画素に対するＳ－ｂｕｆｆｅｒ$_n$が与えられる。この場合、バッファはヌルであるので、これは単なる交差メッセージである。次の画素＃３に関して、Ｓ－ｂｕｆｆｅｒ$_n$から右端列がコピーされ、Ｓ－ｂｕｆｆｅｒ$_{n-1}$に配置される。この場合では１が配置される。関連する稜線が画素の右上４分円において２本の下位線のみと交差することが明らかな場合、画素＃３における交差メッセージが、交差規則に従って判定される。再度、交差メッセージ及びＳ－ｂｕｆｆｅｒ$_{n-1}$が合計され、画素＃３に対するＳ－ｂｕｆｆｅｒ$_n$が与えられる。画素＃４に関して、右端列がコピーされ、Ｓ－ｂｕｆｆｅｒ$_{n-1}$をまたがって配置される。交差メッセージが規則を使用して判定され、合計されて画素＃４に対してＳ－ｂｕｆｆｅｒ$_n$が与えられる。交差メッセージ及びＳ－ｂｕｆｆｅｒを使用してアンチエイリアシングを実施するこの手法により、小画素ベースでレベルを合成することなしに、走査線上の画素位置に対するアクティブレベルを正確に判定できるようになる。全ての小画素プロセスは、Ｓ－ｂｕｆｆｅｒ及び交差標識を使用して行なわれ、画素レベルで合成作業を残す。

【０２０８】

５．１０．アクティブ稜線及び交差メッセージの再順序付け

走査線に対するアクティブ稜線は走査順に処理されるが、対応する交差メッセージは必ずしも走査順（Ｘの昇順）に生成・出力されなくても良い。例えば、稜線２０が図２（ｂ）において処理される場合、交差メッセージ２５が最初に生成・出力され、それにメッセージ２４が続く。交差メッセージ２５は、メッセージ２４よりも高いＸ座標を有するので、交差メッセージは走査順には出力されない。更に、稜線処理モジュール６１４の動作中に新規のｃｕｒｒｅｎｔ＿Ｘを計算すると、第１のアクティブ稜線がこの走査線上で既に処理された第２のアクティブ稜線よりも低い走査位置を有する結果となる恐れがある。

【０２０９】

この状況の例が図２１に示される。ここで、２本の水平な破線２３０及び２３１は出力表示の走査線を表現する。上側の破線２３０は現在稜線処理モジュール６１４により処理されている走査線を表現し、下側の破線２３１は次に処理される走査線を表現する。図２１は、それぞれ、交差２２４、２２５、及び２２６で現在の走査線２３０と交差する３本のアクティブ稜線２２１、２２２、及び２２３を示す。アクティブ稜線のｃｕｒｒｅｎｔ＿Ｘフィールドは交差の位置を示す。稜線処理モジュール６１４は次の走査線２３１上の交点２２７、２２８、２２９に対応するアクティブ稜線の各々に対して新規のｃｕｒｒｅｎｔ＿Ｘ値を計算する。この例では、これらの修正されたアクティブ稜線は走査順になっていない。アクティブ稜線２２３はアクティブ稜線２２１及びアクティブ稜線２２２と交差しているからである。従って、修正されたアクティブ稜線は、次の走査線に対してアクティブ稜線の順序付けられた集合に配置される前に再度のソーティングを必要とする。この例では、次の走査線に対するアクティブ稜線の順序付けられた集合におけるアクティブ稜線の所望の順序は、まず２２３、次に２２１、最後に２２２である。

【０２１０】

稜線処理モジュール６１４は、次の走査線のためにアクティブ稜線の順序付けられた集合へと転送される前に走査順になるように、修正されたアクティブ稜線をソーティング手段（例えば、ソートバッファ）へと挿入する。また、ｚ－レベル活動化モジュール６１３

が走査順で交差メッセージを受信するように、出力の交差メッセージ（図２３のステップ２５２において生成されるように）をソーティング手段（例えば、ソートバッファ）を介して渡す必要がある。

【０２１１】

６．ｚ－レベル活動化モジュール

ｚ－レベル活動化モジュール６１３は、稜線処理モジュール６１４から渡される稜線交点データを使用して、フレームがレンダリングされるときの出力色にどのｚ－レベルが貢献するかを追跡する。出力画素のストリームがレンダリングエンジン６１０により走査順に生成される。稜線処理モジュール６１４からの各交差メッセージは、走査順で生成されるときに所要の出力色が変化する表示座標を表現する。以下の説明では、特定のラスタ空間位置で開始する１つ以上の走査順の連続画素は画素のランと呼ばれる。この説明では、Ａ－ｂｕｆｆｅｒが稜線交差に対して生成されることを、すなわち、レンダリングシステム６１０がアンチエイリアシングを行なう場合を想定する。

【０２１２】

６．１．関心のあるｚ－レベルの順序付けられた集合

ｚ－レベル活動化モジュール６１３は、図５９に一般的に示されるように、「関心のある」ｚ－レベルのリストを維持する。関心のあるとは、当該のｚ－レベルがｚ－レベル活動化モジュール６１３により考慮されなければならないことを意味する。ｚ－レベル活動化モジュール６１３は、関心のあるｚ－レベルの順序付けられた集合を使用して交差メッセージの入力を現在の走査線に沿って画素のランに対する出力色に貢献するのがどのｚ－レベルであるかを示す領域メッセージの出力へと変換する。すなわち、ランは、２つの隣接する交差メッセージにより記述される各点内に位置する。

【０２１３】

ｚ－レベル活動化モジュール６１３の動作は、交差メッセージのストリームをどの深度がスパン上にあり、スパンにまたがるのはどの相対深度順序付けであるかを示す領域メッセージの出力へと変換する観点からも見ることができる。ここでのスパンは、交差メッセージが境界となる走査線上の内部を意味する。ｚ－レベルは深度を塗りつぶしに結合するので、深度の順序付けられた集合は、どのｚ－レベルが現在の走査線に沿って画素のランに対する出力色に貢献するかを判定するのに使用することができる。これが合成器の役割であり、これについては後述する。スパン上に存在するｚ－レベルの集合を判定することは、スパン上に存在する深度を判定するのと同等である。

【０２１４】

６．２．ｚ－レベル活動化モジュールにおける制御のフロー

稜線処理モジュール６１４と同様に、ｚ－レベル活動化モジュール６１３は、レンダリングのタスクを走査線ごとに考慮し、入力（交差メッセージ）を受信することを予期する。単一の走査線に対するｚ－レベル活動化モジュール６１３の機能は、図５８により要約される。走査線に対する処理は、ステップ６２８においてアクティブｚ－レベルのリストを空に初期化し、Ｃｕｒｒｅｎｔ＿Ｘ値を０に設定することによって開始する。Ｃｕｒｒｅｎｔ＿Ｘは、現在処理中の画素を示すために使用され、０の値は走査線の最初（左端）の画素を表現する。

【０２１５】

走査線に対する交差メッセージは、左から右の順（Ｘの昇順）に処理される。

【０２１６】

ステップ６２７において「ｎｏ」により判定されるように、この走査線に対して稜線交差メッセージが存在しない場合、ｚ－レベル活動化モジュール６１３はステップ６２９において走査線全体のレンダリングを要求するであろう。この段階では、関心のあるｚ－レベルのリストは空であるので、走査線はデフォルトの背景色のみでレンダリングされる結果となるであろう。

【０２１７】

現在の走査線に対する交差メッセージが存在する場合（ステップ６２７＝ｙｅｓ）、ス

テップ６３０において次の利用可能な交差メッセージのＸ座標がＣｕｒｒｅｎｔ＿Ｘ値と比較される。Ｃｕｒｒｅｎｔ＿Ｘ及び交差メッセージのＸ座標が等しくない場合、ｚ－レベル活動化モジュール６１３はこの交差メッセージを処理する前に、Ｃｕｒｒｅｎｔ＿Ｘと交差メッセージのＸ座標との間での画素のランのレンダリングを要求しなければならない。この要求はステップ６３１においてレンダリングエンジン６１０における合成モジュール６１２に関心のあるｚ－レベルの順序付けられた集合の内容を使用してこの画素領域に対して出力色を生成させる。

【０２１８】

Ｃｕｒｒｅｎｔ＿Ｘ及び交差メッセージのＸ座標が等しい場合（ステップ６３０において「ｙｅｓ」）、交差メッセージは現在の画素に対する色生成に対応するので、交差メッセージの処理は進行する。

【０２１９】

６．３．ｚ－レベルの活動化及び非活動化：ワインディングカウント

自己交差オブジェクトのレンダリングを促進するために、ｚ－レベル活動化モジュール６１３はワインディング規則を利用する。ｚ－レベルが活動化されたか否かのブール表現だけを格納するのではなく、モジュールはアクティブｚ－レベルごとに小さいワインディングカウント整数値を格納する。ワインディングカウントは当業者には周知である。使用可能なワインディング規則のうちの２つは、従来技術ではノンゼロと奇偶（パリティ）として広く記述されている。使用可能な第３の規則（やや一般性には掛けるが従来技術には記載されている）は、ネガティブワインディングである。これは第６．７節－Ｓ－ｂｕｆｆｅｒのＡ－ｂｕｆｆｅｒへの変換：ワインディング規則で説明される。

【０２２０】

交差メッセージは、「活動化」されるｚ－レベルへの参照を含んでも良い、すなわち、ｚ－レベルはワインディングカウントを減分させなければならない。また、交差メッセージは、「非活動化」されるｚ－レベルへの参照を含んでも良い、すなわち、ｚ－レベルはワインディングカウントを増分させなければならない。このｚ－レベルの増分及び／又は減分は、図２（ｄ）に示すように、交差メッセージのＡ－ｂｕｆｆｅｒにおいて示される画素の部分にのみ適用される。Ａ－ｂｕｆｆｅｒ２９、３０、及び３１は、交差メッセージ２３、２４、及び２５により活動化される小画素の例である。３２、３３、及び３４は、交差メッセージ２６、２７、及び２８により非活動化される小画素の例である。ｚ－レベルの出力画素への貢献に対する活動化及び非活動化の効果は、どのワインディング規則がｚ－レベルと共に使用されるかによって決まる。

【０２２１】

これらの活動化及び非活動化の出力画素へのｚ－レベルの貢献に対する影響は、どのワインディング規則がｚ－レベルと共に使用されるかによって決まる。

【０２２２】

６．４．続関心のあるｚ－レベルの順序付けられた集合

ワインディングカウントの２次元アレイは、図５９に示すように、関心のあるｚ－レベルのリストにおけるｚ－レベルごとにｚ－レベル活動化モジュール６１３により格納される。ワインディングカウントの２次元アレイは「Ｓ－ｂｕｆｆｅｒ」と呼ばれる。各ワインディングカウントは、レンダリング中の現在の画素（６４３、６４７、及び６５１）内の小画素のうちの１つに対してｚ－レベルの状態を表現する。これにより、ｚ－レベル活動化モジュール６１３は現在の画素のどの部分において各ｚ－レベルがアクティブであるかを追跡することができる。関心のあるｚ－レベルの順序付けられた集合の順序は深度順である。図５９に示す例では、ｚ－レベル６４０、６４４、及び６４８はその深度６４１、６４５、及び６４９により順序付けられる。塗りつぶしデータ６４２、６４６、及び６５０はｚ－レベル６４０、６４４、及び６４８の色を定義する。

【０２２３】

６．５．関心のあるｚ－レベルの順序付けられた集合への新規のｚ－レベルの追加

図５８に戻ると、新規の交差メッセージが活動化ｚ－レベルを持たない場合（ステップ

６３８＝ｎｏ）、活動化ｚ－レベルに関連する処理ステップは行なわれない。同様に、交差メッセージが非活動化ｚ－レベルを持たない場合（ステップ６３９＝ｎｏ）、非活動化ｚ－レベルに関連する処理ステップは行なわれない。

【０２２４】

新規の交差メッセージがｚ－レベルを活動化し、ｚ－レベルが既に関心のあるｚ－レベルのリストに存在しない場合（ステップ６３２＝ｎｏ）、ステップ６４０ａにおいて、関心のあるｚ－レベルのリストがチェックされる。満杯の場合、全てがゼロワインディングカウントのｚ－レベルがリストから削除される。この後、ステップ６３４において示すようにリストがｚ－レベル深度順を維持するように、ステップ６３８において検出されたものに対応する新規のエントリがこのリストへと挿入される。交差メッセージのＡ－ｂｕｆｆｅｒが使用され、新規に挿入されたｚ－レベルと関連付けられる小画素カウントのアレイ（すなわち、ｚ－レベルのＳ－ｂｕｆｆｅｒ）が初期化される。Ａ－ｂｕｆｆｅｒにより示される小画素は、－１の値で初期化されるべきＳ－ｂｕｆｆｅｒの小画素カウントに対応する。残りの小画素カウントは０の値で初期化される。このｚレベルからワインディングカウントのアレイへの変換の例は図２（ｄ）に示される。

【０２２５】

新規の交差メッセージが関心のあるｚ－レベルの順序付けられた集合に既に存在するｚ－レベルを活動化する場合（６３２＝ｙｅｓ）、その交差メッセージのＡ－ｂｕｆｆｅｒが使用され、ｚ－レベルと関連付けられたＳ－ｂｕｆｆｅｒのどの小画素が減分されるかが判定される。

【０２２６】

更に図５８において、新規の交差メッセージが関心のあるｚ－レベルのリストにまだないｚ－レベルを非活動化する場合、ステップ６３６においてリストがｚ－レベル深度順を維持するように対応する新規のエントリがリストへと挿入される。交差メッセージのＡ－ｂｕｆｆｅｒは、小画素カウントのアレイ（すなわち、ｚ－レベルのＳ－ｂｕｆｆｅｒ）を初期化するのに使用される。Ａ－ｂｕｆｆｅｒにより示される小画素は＋１に初期化されるべきＳ－ｂｕｆｆｅｒの小画素カウントに対応する。残りの小画素カウントは０の値で初期化される。このＡ－ｂｕｆｆｅｒからワインディングカウントのアレイＳ－ｂｕｆｆｅｒへの変換の例は図２（ｄ）に示される。図２（ｄ）では、Ｘ＝１４，Ｘ＝１５、及びＸ＝１５の交差メッセージがｚ－レベルを非活動化する交差メッセージである。

【０２２７】

新規の交差メッセージが既に関心のあるｚ－レベルのリストに存在するｚ－レベルを非活動化する場合（６３５＝ｙｅｓ）、交差メッセージのＡ－ｂｕｆｆｅｒはｚ－レベルのＳ－ｂｕｆｆｅｒ中のどの小画素カウントが増分されるのか判定するのに使用される。ステップ６３７において、Ａ－ｂｕｆｆｅｒにおいて示される小画素ごとに、Ｓ－ｂｕｆｆｅｒの対応する小画素のワインディングカウントが増分される。

【０２２８】

図７１は、図５８に示すものに対して交差メッセージを処理する代替の手法を示す。図７１において、図５８に示すものと一致する全ての符合は同じ機能を有する。図７１は、図５８のステップ６４０ａが削除されており、ステップ６３４はステップ６３２の「ｎｏ」の指示に直接に続く点において図５８と異なる。また、図７１は、後続のステップ６３７に関心のあるｚ－レベルのリスト中のエントリが全てのゼロのワインディングカウントを有するか否かを確認するように動作させる追加のステップ６４１ａを含む。このようなワインディングカウントを有する場合、このエントリは既に有効ではなく、リストから削除される。このように、図７１の方法は、レンダリング中の画像に貢献することができなくなるとリストからエントリを削除するように動作するので、新規のエントリのためにリストにおいて空間が必要になったときのみエントリが削除される図５８とは対照的である。

【０２２９】

６．５．１．関心のあるｚ－レベルの順序付けられた集合のハードウェアでの維持

考察は、ＡＳＩＣなどのハードウェア実現の考察へと転換することができる。ｚ－レベル活動化モジュール６１３は、図５２（ｂ）に示すように、走査線ごとにｚ－レベル活動化テーブル（ＺＬＡＴ）においてｚ－レベルを管理する。走査線は左から右へと処理されるので、交差メッセージにより参照されるｚ－レベルはＺＬＡＴへと徐々に導入される。各入力ｚ－レベルにはＳ－ｂｕｆｆｅｒ５７３及び塗りつぶし情報５７２を格納するためのＺＬＡＴエントリが割り当てられ、ｚ－レベル深度５７１によりアドレス参照可能である。

【０２３０】

ＺＬＡＴにおけるｚ－レベルエントリは深度により順序付けられていない。交差メッセージにおいて受信された順序で整列される可能性の方が高い。図５２（ａ）は、特定のＸ座標においてアクティブな５つのｚ－レベルとその相対深度との例を示す。図５２（ｂ）は、ＺＬＡＴにおける５つのｚ－レベルエントリが深度により順序付けられていないことを示す。ｚ－レベルエントリは、マップ５７０による深度の降順にｚ－レベル活動化モジュール６１３において維持される。マップ５７０は、ＺＬＡＴと同数のエントリを包含するテーブルであり、ＺＬＡＴの各エントリへのマッピング参照を含むローカル深度の順序付けられた集合である。例えば、ｚ－レベル５５５はローカル深度１（ローカル深度０が最高である場合、２番目に高いｚ－レベルである）を有し、ＺＬＡＴエントリ３、すなわち、ｚ－レベル５５５にマップされる。また、更新されたマップ５７４も示される。

【０２３１】

このメカニズムを使用して、ｚ－レベルエントリは２通りでアクセスすることができる。本開示のＺＬＡＴは、ｚ－レベル深度の供給が対応するｚ－レベルエントリを戻すように、内容アドレス参照可能なメモリとして実現される。これにより、ＺＬＡＴのサイズはスパンにおいてアクティブなオブジェクトの数に従って変動可能であり、ＺＬＡＴへのｚ－レベルの包含及びＺＬＡＴからのｚ－レベルの削除が容易になる。交差メッセージが受信されてＳ－ｂｕｆｆｅｒの状態が更新されるときにＺＬＡＴはこのようにアクセスされる。また、ＺＬＡＴエントリはマップテーブルを介してローカル深度によりアドレス参照可能である。すなわち、特定のローカル深度に存在するｚ－レベルはマップをローカル深度に適用することによって判定することができる。合成のための特定のスパン上でアクティブな深度の順序付けられた部分集合を判定するときに、ＺＬＡＴはこのようにアクセスされる。

【０２３２】

６．５．１．１．ハードウェアにおける新規のｚ－レベルエントリの追加

ｚ－レベルは、走査線上の交差メッセージにより最初に参照されるときにＺＬＡＴに追加される。この後の参照では、各エントリは更新だけすれば良く、ＺＬＡＴ内の同じ位置に残る。入力カウンタが、次の新規のｚ－レベルエントリが挿入されるＺＬＡＴ中の位置を追跡するべく維持される。カウンタは、まず、走査線の開始で位置０にリセットされ、新規のｚ－レベルがＺＬＡＴに追加されるたびに増分される。従って、ＺＬＡＴが満杯になるまで、ｚ－レベルエントリは交差メッセージにおいて受信された順序で維持されることになる。

【０２３３】

走査線の開始において、ＺＬＡＴは最初空である。マップ５７０も、ローカル深度０がＺＬＡＴエントリ０へのマッピングを含み、ローカル深度１がＺＬＡＴエントリ１へのマッピングを含むような初期状態にある。新規のｚ－レベルがＺＬＡＴへのエントリとして追加されると、入力カウンタにより指し示される位置に挿入され、マップ５７０がｚ－レベルの順序付けの変更を反映するように更新され、入力カウンタが増分される。マップを更新するために、入力ｚ－レベルのローカル深度はＺＬＡＴにおいて現在維持されているｚ－レベルのローカル深度に対して判定されなければならない。すなわち、どのｚ－レベルが入力ｚ－レベルの上方にローカル深度を有し、どのｚ－レベルが入力ｚ－レベルの下方にローカル深度を有するかを判定することである。

【０２３４】

このプロセスは図５３（ａ）及び図５３（ｂ）において示される。このプロセスの第１のステップとして、図５３（ａ）に示されるように、現在ＺＬＡＴ５７８において維持されるｚ－レベルエントリは追加対象のｚ－レベルと比較される。ＺＬＡＴエントリはそのｚ－レベル深度を比較することによって、ローカルに入力ｚ－レベルの上方又は下方にあるとマーク付けされる。図５３（ａ）において、ＺＬＡＴエントリはリスト５７７において追加されるｚ－レベル５７５の上方（マーク＝０）又は下方（マーク＝１）にあるとマーク付けされる。

【０２３５】

第２のステップは、マップ５７９により判定された関連付けられたローカル深度の順にマーキングを再度順序付けすることである。これには、入力ｚ－レベルのローカル深度の上方又は下方にあるマップ中のローカル深度間で区別する効果がある。再度順序付けされたマーキング５７６を使用して、マップ５７９は新規のｚ－レベルの追加のためにローカル深度の新規の順序付けを判定するべく更新される。

【０２３６】

マップ５７９を更新するために、５７５において示されるように、追加対象のｚ－レベルのローカル深度よりも下方にあるものとしてマーク付けされたローカル深度が順序において降格され、新規のローカル深度は昇格されて次のマップ５８０を生成する。尚、各ＺＬＡＴエントリのローカル深度は変化するが、ＺＬＡＴエントリはＺＬＡＴ内で静止に維持されるので、そのマッピングは変動しない。例えば、ＺＬＡＴ５７８において、深度９のｚ－レベルは新規のｚ－レベル５７５が追加される前に現在のマップ５７９においてローカル深度４を有する。そのマッピングは、ＺＬＡＴ５７８における位置１に対するものであり、マッピングは、新規のｚ－レベル５７５が追加されて次のマップ５８０においてローカル深度が５に降格された後でも変化しない。

【０２３７】

図５３（ｂ）は、更なるｚ－レベル５８６を図５３（ａ）から生じるＺＬＡＴ５８１に追加する例とマップ５８４、５８５を適切に更新するステップとを示す。

【０２３８】

６．５．１．２．ハードウェアにおけるｚ－レベルエントリの再ソート

拡張として、ｚ－レベルエントリは、関心度とこれに続く降順のｚ－レベル深度の順でＺＬＡＴにおいて維持することができる。関心のあるｚ－レベルは、塗りつぶし規則のＳ－ｂｕｆｆｅｒへの適用がノンゼロＡ－ｂｕｆｆｅｒになるｚ－レベルである。すなわち、関心のあるｚ－レベルは潜在的に貢献するものである。ｚ－レベルの深度の変更が可能でない場合、ｚ－レベルの状態が関心のある状態から関心のない状態へと変化する状況又はその逆の状況がある。これが起こるのは２つのシナリオにおいてである。ｚ－レベル活動化モジュール６１３が、既にＺＬＡＴ内にあるｚ－レベルを参照する交差メッセージを受信し、交差メッセージと関連付けられたＡ－ｂｕｆｆｅｒによりｚ－レベルが関心度の状態を変更する場合に起こり得る。また、ＺＬＡＴエントリのＳ－ｂｕｆｆｅｒの右端ｓｐｅｌがＳ－ｂｕｆｆｅｒ全体にまたがってコピーされなければならないスパンの処理が終了する時に起こり得る。このステップは関心のあるｚ－レベルを関心のない状態にする効果がある。

【０２３９】

ｚ－レベルが関心度の状態を変更するイベントは、各ｚ－レベルのＳ－ｂｕｆｆｅｒの更新を除いてはＺＬＡＴ中のエントリに影響を及ぼさない。しかしながら、新規のローカル深度順序付けを反映させるためにマップを更新する必要がある。このため、マップにより維持されるローカル深度は挿入ソートよりは再ソートされる必要がある。

【０２４０】

図５４（ａ）は、ＺＬＡＴ５９０にあるｚ－レベル５８７が関心のない状態にある場合のシナリオを示す。ＺＬＡＴ５９０の現在の状態５９９は、一部の深度が関心のある状態で、別の一部が関心のない状態であることを示している。現在のマップ５９１は現在のローカル深度順序付けを反映する。これは、ＺＬＡＴ５９０の分割を形成するのに使用する

ことができる。マップ５９１を更新するためには、新規のｚ－レベルを追加するプロセスに関して同様に、ＺＬＡＴ５９０中のエントリに対してマークが作成される。エントリはローカル深度順序付けに対してｚ－レベル変更状態の上方又は下方にあるものとしてマーク付けされる。マーク列５８９では、ローカル深度順序付けが更新されたらエントリのローカル深度順序付けが変動するｚ－レベルの上方になるであろう場合、エントリは１でマーク付けされる。マーキングは、ローカル深度状態５８８を反映するように同様に再度順序付けされる。マップ５９１は、変動するｚ－レベルの上方にあるローカル深度マッピングを昇格し、変動するｚ－レベルのマッピングをより低いローカル深度へと降格することによって、マップ５９２へと更新される。

【０２４１】

図５４（ｂ）は、エントリ５９３が関心のある状態６００になりつつあるときのこのプロセスを示す。この例では、ローカル深度が変動するｚ－レベル５９３のものの下方になるであろう場合は、マーク列５９５に示すようにＺＬＡＴ５９６中のエントリは１でマーク付けされる。マークは同様に再度順序付け（５９４）され、マップ５９７はローカル深度マッピングがより低いローカル深度へと降格され、変動するｚ－レベルのローカル深度が昇格されるように５９８へと更新される。

【０２４２】

６．５．１．３．ｚ－レベルエントリの置換え

メカニズムは現在の走査線上で非アクティブＺＬＡＴエントリを削除するのに必要ではない。レイジー（lazy）方策を実現することができる。新規のｚ－レベルを追加することになったもののサイズに限りがあるためにＺＬＡＴに余裕がない場合を除いては、非アクティブエントリは無視される。このような状況では、最低のローカル深度によりマップされるｚ－レベルエントリが削除され、追加されるｚ－レベルがＺＬＡＴにエントリをとる。請求中のＺＬＡＴのエントリが空であるかのようにマップテーブルが更新される。

【０２４３】

６．５．１．４．単一の走査線上での簡略化ＺＬＡＴ管理

図６５Ａ及び図６５Ｂは、関心のあるｚ－レベルの順序付けられた集合が、図５８で説明された方法により更新されて走査線上で変化する様子の概要を提供する。明確さを期すために、非アクティブｚ－レベル再利用の任意のレイジー方策（第６．５．１．３節で説明）は示されていない－非アクティブｚ－レベルはテーブルから削除される。

【０２４４】

図６５Ａは、範囲[0,18]上にＸ座標を有する特定の走査線６５０８上でレンダリング中の矩形６５０２、円６５０４、及び三角形６５０６から形成される画像６５００を表現する。図６５Ｂは、走査線６５０８上でアクティブな稜線に対して、各稜線と遭遇した際のｚ－レベル活動化テーブルを形成する関心のあるレベル６５３０の順序付けられた集合を示す。

【０２４５】

６５１１において、矩形６５０２と関連付けられた左側稜線が走査線６５０８上の画素＃４と交差し、交差メッセージが生成される。この交差メッセージにより、矩形６５０２と関連付けられたｚ－レベル６５２１は順序付けられた集合６５３０に追加される。

【０２４６】

６５１２において、三角形６５０６と関連付けられた左側稜線が画素＃６と交差し、交差メッセージが生成される。この交差メッセージにより、三角形６５０６と関連付けられたｚ－レベル６５２２は関心のあるｚ－レベルの順序付けられた集合６５３０に追加される。この場合、図５８の方法の適用の結果、三角形６５０４に対する塗りつぶしは順序付けられた集合６５３０の最上部に挿入されることになる。塗りつぶしは、集合６５３０が深度順に維持され、三角形６５０６が矩形６５０２の上方に位置するように挿入される。

【０２４７】

同様に６５１３において、円６５０４に対するｚ－レベル又は塗りつぶし６５２３が三角形６５０６のｚ－レベル６５２２と矩形６５０２のｚ－レベル６５２１との間に挿入さ

れる。６５１４において、矩形６５０２の塗りつぶし６５２１の削除は非アクティブになり、順序付けられた集合６５３０の順序付けを実施しない。６５１５において、円６５０４は非アクティブになり、その塗りつぶし６５２１は順序付けられた集合６５３０から削除される。順序付けられた集合６５３０は空になる。

【０２４８】

６．６．ランの処理

説明はハードウェアにより関係のある問題の考察から戻り、第６．５節からのより一般的な方法の説明を継続する。

【０２４９】

交差メッセージの処理はＣｕｒｒｅｎｔ＿Ｘ（６３０）に対応する更なる交差メッセージがなくなるまで続けられる。これが起こると、図４０に示すように、ｚ－レベル活動化モジュール６１３は、更なる交差メッセージの処理へと進む前に画素のランに対する出力色６３１を生成するように合成モジュール６１２に対して要求する。

【０２５０】

図４０は合成モジュール６１２へのランデータの出力と、関心のあるｚ－レベルの集合が変化しない画素のランの関連追跡とを示す。

【０２５１】

まず、表示画素Ｃｕｒｒｅｎｔ＿Ｘに対する色の生成がステップ３９９において要求されなければならない。このとき、関心のあるｚ－レベルの順序付けられた集合の内容はこの画素の色に貢献する全てのｚ－レベルを記述し、小画素カバレッジ上方がｚ－レベルごとにＳ－ｂｕｆｆｅｒにより提供される。図４０の残りの部分については第６．８節で説明する。

【０２５２】

６．７．Ｓ－ｂｕｆｆｅｒのＡ－ｂｕｆｆｅｒへの変換：ワインディング規則

合成モジュール６１２は関心のあるｚ－レベルと関連付けられたＡ－ｂｕｆｆｅｒを入力としてとる。従って、ｚ－レベル活動化モジュール６１３は、ランデータを合成モジュール６１２に渡す前に、関心のあるｚ－レベルと関連付けられたＳ－ｂｕｆｆｅｒをＡ－ｂｕｆｆｅｒに変換する。そうする際に、ｚ－レベル活動化モジュール６１３はワインディング規則を利用する。上述の３つのワインディング規則は図５７（ａ）及び図５７（ｂ）において示され、Ｓ－ｂｕｆｆｅｒ６２４及び６２０は、各ワインディング規則によりＡ－ｂｕｆｆｅｒ６２１、６２２、６２３及び６２６、６２５、６１９へとそれぞれ変換される：

－奇偶：Ａ－ｂｕｆｆｅｒ６２３及び６１９に示すように、ワインディングカウントが０以外の場合、塗りつぶしは小画素に貢献する

－ノンゼロ：Ａ－ｂｕｆｆｅｒ６２１及び６２６に示すように、ワインディングカウントが奇数の場合、塗りつぶしは小画素に貢献する

－ネガティブワインディング：Ａ－ｂｕｆｆｅｒ６２２及び６２５に示すように、ワインディングカウントが負の数の場合、塗りつぶしは小画素に貢献する

形状全体４２１に対するワインディング規則の結果が図４３（ａ）に示される。図４３（ａ）において、走査線４２４に対する複数のワインディングカウント４２０、４２６、及び４２２が示される。同様に、走査線４２５に対するワインディングカウント４２９、４２７、及び４２３が存在する。図４３（ｂ）において、形状の稜線が図４３（ｂ）の表現において示される形状の範囲内の斜線の塗りつぶしを定義する場合、ノンゼロワインディング規則４２８、ネガティブワインディング規則４３０、及び奇偶規則４３１の適用結果が示される。

【０２５３】

ネガティブワインディング規則は、最小限の計算でストロークされた稜線を生成・レンダリングするのに特に有用である。経路をストロークするプロセスはここでは変換／モーフ／ストローキングモジュール６１６の一部として説明した（第３．４節参照）。ストロークされた経路の一部の例が図３６（ａ）から図３６（ｃ）に示される。図３６（ａ）は

、ストローク対象の３本の稜線３６２、３６１、及び３６０の集合を示す。これらの稜線は起点３５９から時計回りの三角形のパスを形成する。ストローク対象の順序付けられた集合は、「ヌル」左ｚ－レベル３６６、右ｚ－レベル３６７、及びストロークｚ－レベル（すなわち、ペン色）３６５に沿って設けられる。３本の稜線の順序付けられた集合により表現され、ｚ－レベルと関連付けられた形状に対する所望の出力の例が図３６（ｂ）に示される。図３６（ｂ）に示される所望の出力を記述する稜線を達成するには、変換／モーフ／ストローキングモジュール６１６は、図３６（ｃ）に示す左側ストローク稜線３５５及び右側ストローク稜線３５６を生成していることだろう。図３６（ｃ）は、起点３５４において最上の結合の拡大表現を示す。ストロークｚ－レベル３６５が透明度を持たない場合、ストローキングプロセスは以下のものを割り当てるであろう：

－全ての左側ストローク稜線３５５のうちのヌル左ｚ－レベル３６６から左ｚ－レベル参照まで、
－全ての左側ストローク稜線３５５のうちのストロークｚ－レベル３６５から右ｚ－レベル参照まで、
－全ての右側ストローク稜線３５６のうちのストロークｚ－レベル３６５から左ｚ－レベル参照まで、及び
－全ての右側ストローク稜線３５６のうちの右ｚ－レベル３６７から右ｚ－レベル参照まで

図３６（ｃ）は２本の走査線３５７及び３５８を示す。２本の走査線の各々の上方には、走査線に沿って異なる点において稜線の元の集合の右ｚ－レベル参照３６７のワインディングカウントを表現する数字がある。奇偶ワインディング規則又はノンゼロワインディング規則が使用される場合、図３６（ｃ）において明らかなように、自己交差稜線により形成される不要三角領域３６８が右側ストローク稜線により囲まれ、ｚ－レベル３６７でアクティブにレンダリングされるであろう。これは、この領域におけるワインディングカウントが「１」であるからである。ネガティブワインディング規則を使用することによって、ワインディングカウント「１」は非アクティブと見なされ、結合は正しくレンダリングされることになる。

【０２５４】

６．８．続ランの処理

図４０－ランの処理－の考察に戻る。Ａ－ｂｕｆｆｅｒ及び関心のある塗りつぶしの順序付けられた集合を使用して、ステップ３９９の合成モジュール６１２は、単一の画素に対してアンチエイリアシングを行なった色を計算する。

【０２５５】

関心のあるｚ－レベルのリストも使用して、次の交差メッセージのＸ座標まで走査線に沿って後続の画素に対する色を生成する。しかし、ｚ－レベルごとのワインディングカウントのアレイ（Ｓ－ｂｕｆｆｅｒ）は、後続の画素に対する各レベルの正しいカバレッジを示すために修正されなければならない。これは、画素Ｃｕｒｒｅｎｔ＿Ｘに対する場合と同じでないかもしれないからである。これらの後続の画素に対しては、各ｚ－レベルのカバレッジは次の交差メッセージのＸ座標まで変化しない。

【０２５６】

再度、図２の例を参照する。矩形１４の左稜線は、稜線処理モジュール６１４に交差メッセージ２３（Ｘ＝４）を生成させる。Ａ－ｂｕｆｆｅｒは、画素Ｘ＝４に対して、矩形を塗りつぶすのに使用されるｚ－レベルが出力画素の右下隅の４つの小画素に対してアクティブであることを示す。しかし、正しく矩形をレンダリングするために、このｚ－レベルは、ｚ－レベルが非活動化されるＸ＝１５における右側稜線まで走査線に沿って全ての後続の画素上で２本の下位走査線に対してアクティブであり続けるべきである。Ｘ＝５からＸ＝１５まで（Ｘ＝１５を含まない）までの表示画素へのｚ－レベルの貢献を表現する正しいＳ－ｂｕｆｆｅｒは、図８に示すように、右端小画素値を同じ行の他の全ての小画素値へとコピーすることによって生成することができる。図８は、図２の画素Ｘ＝４に対する交差メッセージ６４に対応するＳ－ｂｕｆｆｅｒ６５を示す。Ｓ－ｂｕｆｆｅｒ６６

は、右端ワインディングカウントにわたって各列に対する全ての他のワインディングカウントにコピーした結果を表現する。この結果は、画素Ｘ＝５からＸ＝１４（両端含む）に対する矩形をレンダリングするために使用されるｚ－レベルに対するワインディングカウントを正しく表現する。

【０２５７】

図４０に戻ると、ステップ４００は、関心のあるｚ－レベルの順序付けられた集合における全てのｚ－レベルに対する右端ワインディングカウントにまたがってコピーするように動作し、Ｃｕｒｒｅｎｔ＿Ｘは増分される。ステップ４０１に従ってラン中の残りの画素に対して合成モジュール６１２を呼び出すことができる。現在の走査線に対して交差メッセージが残っていない場合、合成モジュール６１２は、Ｃｕｒｒｅｎｔ＿Ｘから現在の走査線の終了までの画素を生成するように要求される。

【０２５８】

尚、これらの画素に対して生成される色は、アクティブｚ－レベルのうちの１つ以上がブレンド又はビットマップである場合のように、ランにまたがって変化しても良い。ランに対して一定であるのは、カバレッジ情報及びそれに関係するｚ－レベルのみである。関心のあるｚ－レベルのリストの内容に基づく画素色生成の詳細は、合成モジュール６１２の説明の中にある（第７節参照）。図４０の最後のステップ４０２は、Ｃｕｒｒｅｎｔ＿Ｘを次の利用可能な交差があればそのＸ座標のそれに修正する。

【０２５９】

関心のあるｚ－レベルの順序付けられた集合中のｚ－レベルの各々のＳ－ｂｕｆｆｅｒからＡ－ｂｕｆｆｅｒを生成する手順は、単一の画素を合成する手順と同じであり、これについてはＳ－ｂｕｆｆｅｒからＡ－ｂｕｆｆｅｒへの変換：ワインディング規則の節で説明している。

【０２６０】

最後に図５８に戻って説明する。稜線交差メッセージが残らなくなるまで現在の走査線に対して処理される。これはステップ６２７で判定される。図５８の動作はレンダリング対象の全ての走査線に対して繰り返される。

【０２６１】

７．合成モジュール

合成モジュール６１２は：

－ｚ－レベル活動化モジュール６１３により生成されたｚ－レベルの順序付けられた集合及びそれぞれのＡ－ｂｕｆｆｅｒ、及び

－生成対象の出力画素のランの長さを定義する出力表示空間における座標の各々を入力とし：

－出力表示装置上での提示に適した画素のラン

を出力として生成する。

【０２６２】

入力ｚ－レベルの順序付けられた集合は深度順にソートされる。ｚ－レベル深度の概念については先に説明した。以下の考察では、用語「上」及び「下」が使用される。２つのｚ－レベルＡ及びＢがあるとすると、最終の出力がｚ－レベルＡをｚ－レベルＢを覆い隠すものとして描く場合、ｚ－レベルＡはｚ－レベルＢの上、すなわち、上方に位置するものと考えられる。逆にｚ－レベルＢは、ｚ－レベルＡの下、すなわち、下方に位置するものと考えられる。また、最上ｚ－レベルは、他の全てのものを覆い隠す可能性があるが、他の全てのものにより覆い隠される可能性はない点において、概念的に他の全てのものの「上に」あると考えられる。逆に最下ｚ－レベルは、他のどのｚ－レベルも覆い隠すことができないが、他の全てのｚ－レベルにより覆い隠すことができる。また、常に背景ｚ－レベルがあり、この背景ｚ－レベルは入力集合において最下ｚ－レベルの下方にあるものと考えられる。背景ｚ－レベルは入力ｚ－レベルの順序付けられた集合に含まれず、入力の一部であるｚ－レベルの考察が背景ｚ－レベルを含むことはない。背景ｚ－レベルが必要な場合には、明示的にその旨が言及されるであろう。

【０２６３】
７．１．中間生成物
合成プロセス中に、中間合成バッファへのアクセスが必要になる。このバッファは、最大サイズ－すなわち、出力装置の幅－のランに対して中間の合成結果を保持するのに十分であるだろう。当業者は、合成バッファが任意のサイズであっても良く、説明したプロセスの繰り返しの利用を必要としても良いことを理解するであろう。
【０２６４】
合成モジュール６１２は、単一の画素に対する中間の合成結果を格納するのに十分な作業バッファを利用する。これらのバッファはハードウェア実現に対してはＡＳＩＣ内に形成されても、あるいは、ソフトウェア実現に対しては汎用メモリに形成されても良い。バッファは、通常、赤色、緑色、青色、及びアルファ（透明度）成分を有する。
【０２６５】
７．２．ｚ－レベル塗りつぶし
各ｚ－レベルは塗りつぶしへの参照を含む。これは、以下の型のうちの１つであっても良い。
【０２６６】
単色塗りつぶしは、ランの長さに対して明度及びアルファ値が一定である場合の塗りつぶしである。単色塗りつぶしの例としては不透明な赤色の塗りつぶし、５０％アルファの青色の塗りつぶしがある。
【０２６７】
ｘ依存塗りつぶしは、画素のランにまたがって明度が必ずしも一定でない場合の塗りつぶしである。この定義では、ｘ依存塗りつぶしはランにまたがって一定のアルファを有する。ｘ依存塗りつぶしの例としては全てのブレンド制御点で同じアルファの線形又は放射状ブレンド、（アルファが一定の）ビットマップテクスチャ塗りつぶしがある。
【０２６８】
可変アルファ塗りつぶしは、アルファがランにまたがって変動しうる場合の塗りつぶしである。可変アルファ塗りつぶしの例としては、各ブレンド制御点でアルファが同じでない線形又は放射状ブレンド、アルファが一定でないビットマップテクスチャ塗りつぶしがある。
【０２６９】
７．３．基本的なフロー
合成アルゴリズムの基本的なフローが図１３に示される。これは、高レベルの概要であり、一部のステップの更に詳細な説明は後程提供する。
【０２７０】
第１のステップ１１０は、入力集合が空であるか否かを確認するための検査である。集合が空の場合、これは塗りつぶしが存在せず、背景塗りつぶしを描くべきであることを意味する。これはステップ１１１で生成される。合成プロセスはこうして終了する。
【０２７１】
入力集合が空ではない場合、次のステップ１１２では、ｚ－レベルの入力の順序付けられた集合が可変アルファとして分類される塗りつぶしを含むか否かが判定される。合成アルゴリズムにおいて重要なのは最上の可変アルファである。この確認は、可変アルファ塗りつぶしを求めて１つ１つ入力ｚ－レベルの順序付けられた集合を繰り返すことによって行なわれる。好適な順序は、確認される最初の要素が最上の塗りつぶしであるように順序付けられた集合を繰り返すことである。確認は、最初の可変アルファ塗りつぶしが見つかるか、あるいは、集合全体が確認され、可変アルファ塗りつぶしが見つからなかったときに終了する。
【０２７２】
次のステップ１１３では、ステップ１１２において発見された最上の可変アルファ塗りつぶしの上方にある全ての塗りつぶしに対して貢献度計算を行なう。可変アルファ塗りつぶしが存在しない場合、この貢献度計算は入力の順序付けられた集合に存在する全ての塗

りつぶしに対して行なわれる。

【０２７３】

決定１１４が次のステップである。ステップ１１２において可変アルファ塗りつぶしが発見された場合、実行はステップ１１５へと移る。発見されなかった場合、実行は１１６へと移る。

【０２７４】

可変アルファ塗りつぶしが存在しない場合、ステップ１１６において合成バッファを空に初期化する。

【０２７５】

ステップ１１５では、最上の可変アルファ塗りつぶしを含むその下方の全ての塗りつぶしのボトムアップ合成を行なう。次に、ステップ１１７において、ステップ１１５でボトムアップ合成された全ての塗りつぶしの全体の貢献度が計算される。

【０２７６】

ステップ１１８は、最上の可変アルファ塗りつぶしの上方の全ての塗りつぶしを合成するのにトップダウン合成を使用する。

【０２７７】

最終ステップ１１９では、出力表示装置上での提示に適するものとして出力画素を生成する。

【０２７８】

７．４．グラフィカルな概要

処理の各段階は、図９（ａ）、図９（ｂ）、及び図１０にグラフィカルに示される。これらの図は異なる合成シナリオを実証する。

【０２７９】

図９（ａ）は単純な例を示す。入力塗りつぶし７０、７１、７２の順序付けられた集合が示される。これらの塗りつぶしはそれぞれ単色である。貢献度計算が行なわれ、各例の貢献度が６７、６８、及び６９において示される。可変アルファ塗りつぶしは存在しないので、次のプロセスはトップダウン合成である。結果は作業バッファ７３に格納され、更に所要の回数繰り返され、所要の出力画素ラン７４が生成される。この例では、４つの画素のランが生成される。

【０２８０】

第２のシナリオ図９（ｂ）はｘ依存塗りつぶしを伴う。塗りつぶし７５及び７７は共に単色であるが、塗りつぶし７６はｘ依存である。貢献度計算が行なわれ、結果は８０、８１、及び８２により示される。可変アルファ塗りつぶしは存在しないので、次に行なわれるのはトップダウン合成である。ｘ依存塗りつぶし７６に対する画素が生成され、合成バッファ７９へと合成される一方で、単色塗りつぶし７５及び７７に対する色が生成され、作業バッファ７８へと合成される。作業バッファ及び合成バッファの双方の貢献度が示される。最終ステップでは、作業バッファの単一の画素を合成バッファの各画素へと合成することによって出力画素を生成し、出力画素８３を生成する。

【０２８１】

図１０の最終例はより複雑であり、可変アルファ塗りつぶしを含む。塗りつぶし８４は単色塗りつぶし、塗りつぶし８５はｘ依存塗りつぶし、塗りつぶし８６は可変アルファ塗りつぶし、塗りつぶし８７は単色塗りつぶしである。貢献度計算後、塗りつぶし８４及び８５は関連付けられた貢献度値を有する（８８及び８９により示すように）。可変アルファ塗りつぶし８６を含むその下方の塗りつぶしは全て貢献度値を持たない。実行する次のプロセスはボトムアップ合成である。塗りつぶし８６及び８７はボトムアップ合成され、結果は合成バッファ９２（ｃ１）に格納される。ボトムアップ合成後、塗りつぶし８８及び８９の貢献度を１００％から差し引くことによって、合成バッファに格納された結果の貢献度を考え出すことができる。次のステップはトップダウン合成である。最上の塗りつぶし８４は単色であるので、作業バッファ９５へと合成される。塗りつぶし８５はｘ依存塗りつぶしであるので、合成バッファ９７（ｃ２）へと合成される。トップダウン合成後

、作業バッファ９５が単一の塗りつぶし８４に対する合成結果を含み、合成バッファ９７（ｃ２）が塗りつぶし８５、８６、及び８７に対する合成結果を含むことが明らかである。最終段階では出力画素を生成する。作業バッファの単一の画素が合成バッファの各画素へと合成され、出力画素９６が生成される。

【０２８２】

７．５．貢献度計算

図１３のステップ１１３が、図１１を参照して更に詳細に検査される。このプロセスは、ステップ１１２で見つかった最上の可変アルファ塗りつぶしの上方にある入力ｚ－レベルの順序付けられた集合中の全ての塗りつぶしに対して行なわれる。入力集合に可変アルファ塗りつぶしが存在しない場合、このプロセスは入力集合中の全ての塗りつぶしに対して行なわれる。最上の可変アルファ塗りつぶしが入力集合中においても最上の塗りつぶしである場合、貢献度計算を行なう必要はない。

【０２８３】

貢献度計算は、このプロセスを受けさせる各塗りつぶしに対して新規の変数を導入する。これが塗りつぶしのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅである。これは、常に、ゼロの貢献度を表現するように初期化される。例えば、貢献度を格納するのに８ビット整数を使用することができる。０は貢献度ゼロを表現し、２５５は貢献度１００％を表現する。貢献度計算の目的は、各塗りつぶしが出力画素にどの程度貢献しているかを判定することである。

【０２８４】

２つの変数ａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙ及びｃｏｖｅｒａｇｅはここで導入される。ｃｏｖｅｒａｇｅは、処理中の塗りつぶしと関連付けられたＡ－ｂｕｆｆｅｒのどのｓｐｅｌ要素が現在考慮中であるかの記述である。ｃｏｖｅｒａｇｅはＡ－ｂｕｆｆｅｒとして表現するのが便利である。ａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙは、考慮中のｓｐｅｌが現在の塗りつぶしの成功の塗りつぶしによりどの程度覆い隠されるかの測定値である。続く考察では、０％のａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙは完全な透明度を指し示し、１００％のａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙは完全な不透明度を指し示す。これらの２つの変数は、処理中のｃｕｒｒｅｎｔ＿ｆｉｌｌの表示を伴って、貢献度計算アルゴリズムの状態を記述するパラメータの集合を形成する。従来技術において周知であるように、スタックの概念は、貢献度計算アルゴリズムにより読み込むことが可能であり、作用されることが可能である３つのパラメータの集合を格納するのに使用される。以下の説明では、用語プッシュはスタックの最上部にパラメータの集合を追加することを意味し、ポップはスタックの最上部のパラメータの集合が取り除かれることを意味する。スタックはパラメータスタックと呼ぶ。

【０２８５】

図４８は、上から下まで順番にレンダリングするときのＡ－ｂｕｆｆｅｒにおけるａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙの影響を示す。オブジェクトＡ ４８００、Ｂ ４８０１、Ｃ ４８０２、及びＤ ４８０３の各部分は、それぞれ対応するＡ－ｂｕｆｆｅｒにおける特定のｓｐｅｌに適用可能な不透明度値を有する。尚、１００％の不透明度はオブジェクトが完全に不透明であることを意味する。オブジェクトは、０％の不透明度（すなわち、１００％透明）を有する背景オブジェクト４８０４にまたがってレンダリングされている。図４８において、黒色の陰影は、Ａ－ｂｕｆｆｅｒの関連する小画素にまたがるオブジェクトによるカバレッジを示す。最初に、オブジェクトＡ ４８００を背景４８０４に適用することにより結果４８０５が現れる。Ａの２つのｓｐｅｌのみが３５％不透明であるので、オブジェクトＡ ４８００の貢献度は結果の画素値に対して１７．５％でしかない。尚、０％の不透明度は０％の貢献度をもたらす。累積の不透明度は図示されるようにツリーへと分解する。このツリーでは、枝の本数はマスク中の小画素の数に限定され、この場合では４本である。オブジェクトＡ ４８００の背景４８０４への適用の２つの結果４８０６及び４８０７が示される。続いて、オブジェクトＢ ４８０１が適用されると、Ａ－バッファの下側の２つのｓｐｅｌ上でのみ動作する。左下ｓｐｅｌ４８０８がオ

ブジェクトＡのものと交差すると、２つの貢献度が蓄積する。右下ｓｐｅｌ４８０９に関しては、オブジェクトＡ　４８００は、ｓｐｅｌに対するこの段階での結果がオブジェクトＢ　４８０１のものに限定されるように貢献度を持たない。オブジェクトＢの合計貢献度は計算から１６．５％に判定される。従って、オブジェクトＢの適用後、各ｓｐｅｌはそれぞれの不透明度レベルを有し、４８１０において６１％、４８１１において３５％、４８１２において４０％、４８１３において０％を有する。オブジェクトＣ　４８０２は１００％の不透明度を有し、結果４８１４、４８１５、４８１６、及び４８１７を与える右の２つのｓｐｅｌに衝突する。これにより、画素に対する４０％の貢献度が示される。オブジェクトＣは完全に不透明であるので、オブジェクトＣと交差する可能性のある下側のオブジェクト（オブジェクトＤ）を考慮する必要はない。このため、右ｓｐｅｌに対して、３本の枝がオブジェクトＣにおいて停止する。オブジェクトＤ　４８０３は１００％の不透明度を有し、４８１８、４８１９、４８２０、及び４８２１において見られ、画素への２６％の貢献度が提供される。図４８によりアクティブグラフィックオブジェクトに対する不透明度値から判定される蓄積された貢献度は１００％に達することが明らかである。

【０２８６】

図４８のＡ－ｂｕｆｆｅｒに関して、結果の貢献度は以下のように表現しても良い：

```
－－－－－－－－－－－－－－－－
｜Ａ＝３５％｜Ａ＝０％　　｜
｜Ｂ＝０％　｜Ｂ＝０％　　｜
｜Ｃ＝０％　｜Ｃ＝１００％｜
｜Ｄ＝６５％｜Ｄ＝０％　　｜
｜－－－－－＋－－－－－－－｜
｜Ａ＝３５％｜Ａ＝０％　　｜
｜Ｂ＝２６％｜Ｂ＝４０％　｜
｜Ｃ＝０％　｜Ｃ＝６０％　｜
｜Ｄ＝３９％｜Ｄ＝０％　　｜
－－－－－－－－－－－－－－－－
```

図４８により、レベルの数が小画素マスク（Ａ－ｂｕｆｆｅｒ）のサイズにより判定されるツリーが横断されることが明らかである。オブジェクトＣ　４８０２に対して明らかであるように、ツリーの任意の枝の横断は１００％の不透明度に到達するとすぐに停止することができ、その小画素に対しては更なる貢献度が作成されないかもしれない。

【０２８７】

実際の処理に戻ると、図１１のプロセスの第１のステップ９８ではパラメータを初期化する。ｃｕｒｒｅｎｔ＿ｆｉｌｌは入力集合における最上の塗りつぶしに初期化される。ａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙは０％に初期化され、ｃｏｖｅｒａｇｅはＡ－ｂｕｆｆｅｒのカバレッジ全体を表現するように初期化される。このパラメータの最初の集合は、パラメータスタックへとプッシュされる。更に、処理対象の全ての塗りつぶしの全てのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅは0に初期化される。

【０２８８】

メインプロセスループの初期ステップ９９は、パラメータスタックからパラメータの次の集合をポップする。これは、ステップ１００において空の結果に対して調べられる。スタックにパラメータが存在しない場合、貢献度計算は終了する。パラメータ集合がスタックから首尾良くポップされた場合、次のステップ１０１では、現在のａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙ及びｃｏｖｅｒａｇｅを考慮してｃｕｒｒｅｎｔ＿ｆｉｌｌの貢献度を計算し、これをｃｕｒｒｅｎｔ＿ｆｉｌｌのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅ全体に加える。計算は以下の式に従って行なわれる：

```
current_contrib=current_alpha*(1-accumulated_opacity)*coverage_ratio
```

ここで、
current_alpha=current_fillのアルファ値、
accumulated_opacity＝現在のパラメータ集合において与えられるようなaccumulated_opacity、
coverage_ratio=active_ratio (coverage? A-buffer)、
A-buffer=current_fillと関連付けられたA－buffer、及び
Active-ratio (A-buffer)＝与えられたA-bufferにおけるspelの総数に対する与えられたA-bufferにおけるアクティブspelの数の比率。

【０２８９】

尚、主な式の全ての要素は範囲[0..1]の実数として表現される値として意図される。これは、式の表現を簡略化するためであり、これらの要素の表現の他の手段を収容するために式を操作する方法は当業者には明らかであろう。ｃｕｒｒｅｎｔ＿ｃｏｖｅｒａｇｅ比率の場合、[0..1]範囲はＡ－ｂｕｆｆｅｒ中のｓｐｅｌの総数に対するｃｕｒｒｅｎｔ＿ｆｉｌｌのＡ－ｂｕｆｆｅｒを伴うｃｏｖｅｒａｇｅＡ－ｂｕｆｆｅｒの交点における「アクティブ」ｓｐｅｌの数の比率を表現する。

【０２９０】

この計算されたｃｕｒｒｅｎｔ＿ｃｏｎｔｒｉｂ値はｃｕｒｒｅｎｔ＿ｆｉｌｌのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅに追加される。

【０２９１】

図１１のプロセスの次のステップ１０２は、処理すべき次の塗りつぶしが存在するか否かを確認するためにチェックする。ｃｕｒｒｅｎｔ＿ｆｉｌｌが入力集合の最後の塗りつぶしである、あるいは、次の塗りつぶしが可変アルファ塗りつぶしである場合、このチェックの結果は正になり、ｃｕｒｒｅｎｔ＿ｆｉｌｌは更新されて次の塗りつぶしが識別される。

【０２９２】

次のステップ１０３において、パラメータの集合がパラメータスタックへとプッシュされる。これらのパラメータは：

current_fill ← current_fill
accumulated_opacity ← accumulated_opacity
coverage←coverage?(coverage?A-buffer)

次のステップ１０４は以下のように現在のパラメータ集合を更新する：

current_fill ← current_fill
accumulated_opacity ← (1-current_alpha)*(1-accumulated_opacity)
coverage ← coverage?A-buffer

尚、accumulated_opacity及びcurrent_alphaは範囲［０．．１］の実数として表現される値として意図される。

【０２９３】

coverageに適用される動作のグラフィカルな表示は、図１２（ａ）から図１２（ｄ）において見ることができる。A-buffer１０７はcoverageを表現する例を提供する。第２の例のA-buffer１０６は塗りつぶしのA-bufferを表現する。ステップ１０３において与えられるような新規のcoverageのA-bufferを生成する動作が適用され、結果はA-buffer１０８である。ステップ１０４において与えられるような新規のcoverageのA-bufferを生成する動作が適用される場合、結果はＡ－ｂｕｆｆｅｒ１０９である。

【０２９４】

この更新が発生すると、ａｃｃｕｍｕｌａｔｅｄ＿ｏｐａｃｉｔｙは十分な不透明度が達成されたかを確認するために検査される。一例としての閾値は完全な不透明である２５５／２５６である。十分な不透明性が達成される場合、プロセスの実行はステップ９９へと移る。達成されない場合、プロセスの実行はステップ１０１から継続する。

【０２９５】

７．６．ボトムアップ合成

図１３に戻ると、ステップ１１４で入力集合において可変アルファ塗りつぶしが検出された場合、ボトムアップ合成ステップ１１５が行なわれる。

【０２９６】

ボトムアップ合成プロセスは、最下部の可変アルファ塗りつぶしから最上部の可変アルファ塗りつぶしまで（最上部を含む）の順序で塗りつぶしを合成する。以下の考察において「次の」塗りつぶしに言及する場合、これはｃｕｒｒｅｎｔ＿ｆｉｌｌの真上の塗りつぶしを指し示す。

【０２９７】

ボトムアップ合成プロセスは、塗りつぶし生成バッファへのアクセスを必要とする。このバッファは少なくとも合成バッファと同等の画素を格納することが可能でなければならない。例えば、このバッファは合成バッファの半分を利用することによって実現されても良い。これは、合成バッファのサイズを半分にし、合成バッファの元の長さの半分よりも長い出力ランの場合、合成モジュール６１２の第２の適用例が必要となるであろう。

【０２９８】

図６において、ボトムアップ合成プロセスは、第１のステップ５０が背景塗りつぶしを合成バッファへと合成することであると描かれる。これは、合成バッファへと合成される第１の塗りつぶしであると保証されるので、特別な計算は必要でない。塗りつぶし色がコピーされるだけである。次のステップの最初の実行のために、ｃｕｒｒｅｎｔ＿ｆｉｌｌが背景塗りつぶしであると見なされる。

【０２９９】

次のステップ５１は、ボトムアップ合成を必要とする次の塗りつぶしがあるか否かを確認するための検査である。ｃｕｒｒｅｎｔ＿ｆｉｌｌが最上の可変アルファ塗りつぶしである場合、ボトムアップ合成への更なる塗りつぶしはなく、ボトムアップ合成プロセスは最後から２番目のステップ５５に移る。

【０３００】

ステップ５５におけるボトムアップ合成結果の貢献度の判定は以下のアルゴリズムに従って行なわれる：

ｔｏｔａｌ＿ｃｏｎｔｒｉｂ　＝　０

最上の可変アルファ塗りつぶしの上方の塗りつぶしごとに

ｔｏｔａｌ＿ｃｏｎｔｒｉｂ＝ｔｏｔａｌ＿ｃｏｎｔｒｉｂ＋ｆｉｌｌ［ｎ］．ｃｏｎｔｒｉｂ

ｂｏｔｔｏｎ＿ｕｐ＿ｃｏｎｔｒｉｂ＝１－ｔｏｔａｌ＿ｃｏｎｔｒｉｂ

ここで、

ｆｉｌｌ［ｎ］．ｃｏｎｔｒｉｂ＝図１３のフローチャートのステップ１１３において計算されるような、最上の可変アルファ塗りつぶしの上方のｎ番目の塗りつぶしの貢献度値

ボトムアップ合成結果に対する貢献度値が計算されると、以下のアルゴリズムに従って合成バッファへと倍増される：

合成バッファの画素ごとに

ｃｂ［ｎ］＝ｃｂ［ｎ］＊ｂｏｔｔｏｍ＿ｕｐ＿ｃｏｎｔｒｉｂ

ここで、

ｃｂ［ｎ］＝合成バッファのｎ番目の画素

明度（ｃｂ［ｎ］）を乗じたスカラー値（ｂｏｔｔｏｍ＿ｕｐ＿ｃｏｎｔｒｉｂ）の上述の表現は、色の全成分のスカラーによる乗算を指し示す。

【０３０１】

ステップ５１において、ボトムアップ合成対象の塗りつぶしが更に存在することが判定される場合、ｃｕｒｒｅｎｔ＿ｆｉｌｌはステップ５２を介して次の塗りつぶしに設定される。ステップ５３においてｘ依存塗りつぶしであるか否かを確認するためにｃｕｒｒｅｎｔ＿ｆｉｌｌは検査される。塗りつぶしがｘ依存でない場合、ステップ５４に従って塗りつぶし色は作業バッファへと生成される。作業バッファの内容は以下のアルゴリズムを

使用して合成バッファへと繰り返し合成される：
ｃｏｍｐｏｓｉｔｅ＿ｂｕｆｆｅｒの画素ごとに
ｃｂ［ｎ］＝ｃｂ［ｎ］＋ｗｏｒｋ＿ａｌｐｈａ＊（ｃｂ［ｎ］－ｗｏｒｋ）
ここで
ｃｂ［ｎ］＝合成バッファのｎ番目の画素
ｗｏｒｋ＿ａｌｐｈａ＝作業バッファの画素のアルファ
ｗｏｒｋ＝作業バッファの画素
ボトムアップ合成の実行はステップ５１へと戻る。

【０３０２】

塗りつぶしがｘ依存である場合、ステップ５６においてランに対する全ての画素は塗りつぶし生成バッファへと生成される。ステップ５７において塗りつぶし生成バッファは以下のアルゴリズムを使用して合成バッファへと合成される：
合成バッファ及び塗りつぶしバッファの画素ごとに
ｃｂ［ｎ］＝ｃｂ［ｎ］＋ｆｂ［ｎ］．ａｌｐｈａ＊（ｃｂ［ｎ］－ｆｂ［ｎ］）　式１
ここで
ｃｂ［ｎ］＝合成バッファのｎ番目の画素
ｆｂ［ｎ］＝塗りつぶしバッファのｎ番目の画素
ｆｂ［ｎ］．ａｌｐｈａ＝塗りつぶしバッファのｎ番目の画素のアルファ
ボトムアップ合成の実行は５１へと続く。

【０３０３】

７．７．トップダウン合成

図１３のステップ１１８において示されるトップダウン合成プロセスについて図５５を参照して詳細に説明する。このプロセスは、ステップ１１２において判定されるような最上の可変アルファ塗りつぶしの上方の全ての塗りつぶしに適用される。可変アルファ塗りつぶしが入力集合に存在しない場合、トップダウンアルファプロセスが入力集合の全ての塗りつぶしに適用される。

【０３０４】

トップダウンプロセスは、ステップ１１３において計算される塗りつぶしに対してｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅを利用する。尚、トップダウン合成プロセスでは、塗りつぶしが処理される順序は最終的な結果の観点からすると重要ではない。以下の例では、入力集合に出現するのと同じ順序で処理される。すなわち、集合中の最上の塗りつぶしで開始される。

【０３０５】

図５５において示すように、トップダウン合成プロセスの第１のステップ６０１では、処理対象の塗りつぶしが更にあるか否かを判定する。次に処理すべき塗りつぶしが存在しない場合、あるいは、次の塗りつぶしがステップ１１２において判定されるように最上の可変アルファ塗りつぶしである場合、プロセスはステップ６０７へと移る。

【０３０６】

処理対象の塗りつぶしが更にある場合、ステップ６０２において、現在の塗りつぶしは可変のｃｕｒｒｅｎｔ＿ｆｉｌｌに格納される。次のステップ６０４では、０より大きいか否かを確認するために、ｃｕｒｒｅｎｔ＿ｆｉｌｌのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅが調べられる。この検査が偽を戻す場合、実行はステップ６０１へと移る。

【０３０７】

ｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅが０よりも大きい場合、実行はステップ６０３へと移る。ステップ６０３は、ｘ依存塗りつぶしを記述するか否かに関してｃｕｒｒｅｎｔ＿ｆｉｌｌを検査する。塗りつぶしがｘ依存でない場合、実行はステップ６０６へと移る。ｘ依存である場合、ステップ６０３は正の結果を戻し、実行はステップ６０５へと移る。

【０３０８】

ステップ６０６では、非ｘ依存塗りつぶしが扱われ、以下のように進められる。塗りつぶしに対する色が生成され、以下のアルゴリズムに従って作業バッファへと合成される：

ｗｂ＝ｗｂ＋ｃｕｒｒｅｎｔ＿ｆｉｌｌ．ｃｏｎｔｒｉｂ＊ｃｕｒｒｅｎｔ＿ｆｉｌｌ．ｃｏｌｏｒ　式２

ここで

ｗｂ＝作業バッファ

ｃｕｒｒｅｎｔ＿ｆｉｌｌ．ｃｏｎｔｒｉｂ＝ｃｕｒｒｅｎｔ＿ｆｉｌｌのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅ

ｃｕｒｒｅｎｔ＿ｆｉｌｌ．ｃｏｌｏｒ＝ｃｕｒｒｅｎｔ＿ｆｉｌｌの明度

ステップ６０５では、ｘ依存塗りつぶしが扱われ、以下のように進められる。ｘ依存塗りつぶしの各画素が生成され、以下のアルゴリズムに従って合成バッファへとただちに合成される：

ｃｂ［ｎ］　＝　ｃｂ［ｎ］　＋　ｃｕｒｒｅｎｔ＿ｆｉｌｌ．ｃｏｎｔｒｉｂ　＊　ｇｐ

ここで

ｃｂ［ｎ］＝合成バッファのｎ番目の画素

Ｃｕｒｒｅｎｔ＿ｆｉｌｌ．ｃｏｎｔｒｉｂ＝ｃｕｒｒｅｎｔ＿ｆｉｌｌのｃｏｎｔｒｉｂｕｔｉｏｎ＿ｖａｌｕｅ

ｇｐ＝現在考慮中の生成された塗りつぶし画素

ステップ６０５又はステップ６０６が終了すると、実行はステップ６０１へと戻る。

【０３０９】

ステップ６０１において処理すべき塗りつぶしが残っていないと判定される場合、実行はステップ６０７へと移る。このステップでは、作業バッファは合成バッファへと合成され、最終結果が生成される。このステップは以下のアルゴリズムに従って行なわれる：

ｃｏｍｐｏｓｉｔｅ＿ｂｕｆｆｅｒの画素ごとに

ｃｂ［ｎ］＝ｃｂ［ｎ］＋ｗｂ

ここで

ｃｂ［ｎ］＝ｃｏｍｐｏｓｉｔｅ＿ｂｕｆｆｅｒのｎ番目の画素

ｗｂ＝作業バッファ

これでトップダウン合成は完了する。ｃｏｍｐｏｓｉｔｅ＿ｂｕｆｆｅｒは出力ランの完全に合成された画素を含む。

【０３１０】

７．８．代替の合成手法

合成プロセスの代替の手法について説明する。可能な限りトップダウン合成プロセスを使用する代わりにこの手法を使用することにより、可変アルファ層が遭遇する場合に大規模なボトムアップ合成を潜在的に行なう必要がなくなる。このため、トップダウン合成が提供する利点が最大になる。

【０３１１】

上述の貢献度計算プロセスは、最上の可変アルファ層に遭遇すると停止する。代替の合成方法は、最上の可変アルファ層の下方に層が存在するか否かを確認するためにチェックする。層が存在する場合、合成器が再帰的に呼び出され、最上の可変アルファ層の下方のこの層は新規の呼出しにおける最上である。この新規の呼出し及び後続の呼出しは、各層を通常通りに処理し、完全な不透明に到達すると、通常通り画素は作業バッファ及び合成バッファへと生成される。再帰的な呼出しが親に戻る場合、合成バッファにおける結果は親において発見される可変アルファ層を伴って合成される。結果の作業バッファ及び合成バッファにおける貢献度は、図６のステップ５５におけるように計算され、これらのバッファの双方における画素値はそれに従って調整される。

【０３１２】

画素のランに対するこれらの合成プロセスの一般化は以下の通りである：

（ｉ）　可変アルファ層により分けられる各グループへと各層を分割する

（ii） 各グループに対してトップダウン合成を実行してグループ画素値を判定し、それにより、可変アルファ層により分けられるグループ画素値から構成される減少した数へと元の層を分割する

（iii） 任意の１つのグループの貢献度が所定の閾値１００％（すなわち、完全に不透明）内にある場合、全ての可変アルファ層及び閾値グループの下方のグループは貢献しないものとして無視することができる

（iv） ラン中の画素ごとに、これらの貢献グループ及び可変アルファ層のボトムアップ合成を行ない、対応する画素値を判定する

合成プロセスの更なる実現は以下のように２つのバッファを使用することで達成することができる：

（ｉ） 最上層から第１のバッファを使用して開始し、第１の可変アルファ層まで（第１の可変アルファ層を含む）トップダウン合成を行なう

（ii） 第２のバッファを使用して次の可変アルファ層まで（次の可変アルファ層を含む）これらの層のトップダウン合成を行なう

（iii） 第１の層を第２の層の上に合成し、結果を第１のバッファに保持する

（iv） 全ての層を合成し終わるまでステップ（ii）及び（iii）を繰り返す。かつてのように、可変アルファ層の上方の層の任意の集合の貢献度が所定の閾値１００％（すなわち、ほぼ又は完全に不透明）に到達する場合、全ての可変アルファ層及び閾値グループの下方のグループは貢献しないものとして無視することができ、合成は中止することができる

尚、上述の更なる情報は、可変アルファ層の進歩的な包含が下方のグループから層の貢献度を正確に説明しないため、画像を正確に描くことはない。例えば、第１の可変アルファ層まで（第１の可変アルファ層を含まない）の全ての最上の層の貢献度が８０％である場合、第１の可変アルファ層及び全ての下方の層の貢献度は２０％となるはずである。第１の可変アルファ層を上述のものとの合成は、貢献度のバランスを歪め、不正確ではあるが高速な合成が得られる。これは、最終的な画像レンダリングには望ましくないかもしれないが、全ての層の詳細が正確でなくても良い画像構成のプレビュー段階では十分であるだろう。

【０３１３】

所定の閾値は、通常、合成の所望の精度及び行なわれる動作の数によって決まる。これは、例えば、カラーチャンネルの最下位ビット内であるかもしれない。８ビット色の場合、２５５／２５６の貢献度（９９．６％）が得られる。他の閾値が選ばれても良い。

【０３１４】

図６１（ａ）は、画素のラン内で複数の層のトップダウン合成に対して実行可能な一般化を示す。図示するように、異なる型を有する複数の層６１０１から６１０７が示される。１つの型は一定の色レベルであり、本質的に一定のアルファ成分を含むことができる。これらはレベル６１０１、６１０２、６１０４、及び６１０６である。別の型のレベルは可変色レベルであるが、一定のアルファを有する。この例は、画素のランにまたがって起こる線形ブレンドである。アルファ値は一定であるので単一の可変色として見なしても良く、この例はレベル６１０３及び６１０５である。レベルの最後の型の例はレベル６１０７であり、可変色及び可変アルファの双方を含むレベルである。その性質により、このレベルは可変アルファレベルとして一般化しても良く、画素のスパンのレンダリングにおけるレベルの下方にある全てのレベルの貢献度を変更するように動作する。

【０３１５】

図６１（ａ）の一般化は、トップダウン合成に関するものであり、各々が一定のアルファを有する可変アルファレベルの上方の全てのレベルは、元のソースレベルによって可変色レベルであっても良い単一の色レベルを提供するようにトップダウン合成されても良い。この一般化は、レベル６１０１から６１０６のトップダウン合成を表現する新規の色レベル６１０８により示される。レベル６１０８は前の可変アルファレベル６１０７の上方に配置される。

【０３１６】
このように図６１（ａ）は、一定のアルファ成分を有する任意の数のレベルが、可変アルファを有するレベルにまたがって合成されても良い一定のアルファの単一のレベル（例えば、６１０８）を提供するようにトップダウン合成されても良いことを示す。
【０３１７】
図６１（ｂ）は、この一般化が、可変色の層のグループ６１１０、６１１２、及び６１１４が可変アルファの層６１１１及び６１１３により分離される多数の層を組み込む合成スタックへとどのように適用することができるのかを示す。この構成では、可変色一定のアルファを有する層のグループの各々が、一定のアルファの対応する単一の層を生成するようにトップダウン合成することができる。これは、新規の層６１１５、６１１６、及び６１１７として示される。これらの新規の層の各々は、画素のランの正確な合成を提供するために、ボトムアップ合成に配置された可変アルファ層６１１１及び６１１３を伴って合成することができる。最も重要なことは、合成動作を可能な限り多くのトップダウン合成へと分割することによって、合成動作の数が画素のランにまたがって最小化される。この点に関して、任意の可変アルファレベルの動作により、画素のランにおける各画素で合成を必要とする。このように、トップダウン合成とボトムアップ合成の組み合わせは、以前は可能でなかった最適化を提供する。
【０３１８】
この原理の拡張は図６２に示される。図６２において、画素のスパンに対する合成スタック６２００が、一定の色、可変色、及び可変アルファ層の組み合わせを含む種々の層６２０１から６２１２を使用して形成される。変動に従って、合成スタック６２００は、画素のランに対する画素値を提供するように、トップダウン合成動作のみを使用して合成される。必然的にスタック６２００は複数のセクションへと分割され、各セクションは可変アルファ層により輪郭を描かれる。この構成では、トップダウン合成６２１３は第１の可変アルファ層まで（第１の可変アルファ層を含む）の層の第１のグループ上で行なわれ、このグループは層６２０１から６２０４である。合成の結果は第１のバッファ６２１４に保持される。第２のトップダウン合成６２１５は、次の可変アルファ層まで（次の可変アルファ層を含む）の層の次のグループ上で行なわれ、このグループは層６２０５から６２０７である。合成６２１５の結果は第２のバッファ６２１６に格納される。バッファに含まれる合成結果は、第１のバッファ６２１４が第２のバッファ６２１６上に合成されるように合成され、この合成の結果が図６２の６２１７に示すように第１のバッファへと書き込まれる。層６２０８から６２１１の次のグループが、第２のバッファ６２１９に格納されている合成値を伴って再度トップダウン合成（６２１８）される。再度、第１のバッファ６２１７が第２のバッファ６２１９上に合成され、第１のバッファに格納される結果６２２０が得られる。最終的に、図６２の例では、一定の色層６２１２のみを含む層の最後のグループがトップダウン合成（６２２１）される。この場合、層のグループは単一の一定の色のみを含むので、合成動作は事実上ヌルであるが、層６２１２の値は第２のバッファ６２２２にコピーされる。再度、バッファは第１のバッファに保持されている最終結果６２２３を伴って合成される。
【０３１９】
図６２の説明から理解されるように、トップダウン合成のみへの依存は最終合成結果が画素の特定のスパンに対して完全に正確であることを保証するとは限らない。しかし、本発明者等は、最終的な合成結果が精度の決定できるレベルへと作成可能であるように合成動作に対して確立されても良いことを判定した。これは、場合によっては画質を犠牲にしても非常に高速な合成が必要であるという含みを持つ。この例は、例えば、グラフィックアーティストにより行なわれるように画像作成プロセスをプレビューすることにある。他の実現例は、合成プロセスが変動する処理能力の複数の異なるプラットフォームで実行することができる場合である。特に、合成時間要求が指定される場合（例えば、リアルタイムの動作で）、システム要件を満たすために、合成スタックの複雑度は特定の型の合成（例えば、図６１（ａ）－正確であるが低速又は図６２－高速であるが正確性に劣る）を引

き起こすのに使用される。

【０３２０】

更なる代替の手法が図６４に示され、ボトムアップ合成を不要にすることで先に説明した構成よりも改良を提供する。トップダウン合成の排他的使用により、トップダウン合成の利益の更なる活用が可能になり、不透明度が１００％に十分に近づいたときにプロセスから早く抜けられる。この代替の手法のコストは、図９及び図６２と比較して、追加の作業バッファ（ここでは、「作業バッファ２」と呼ばれる）である。この新規の作業バッファは作業バッファ１（例えば、７８）と同じ要素を持つが、各要素はｒｇｂではなくｒｇｂｏ（ｒ＝赤色、ｇ＝緑色、ｂ＝青色、及びｏ＝不透明度）である。

【０３２１】

動作中、合成スタックは可変不透明度層により分離されるセクションへと分割される。作業バッファ０（例えば、７３）及び作業バッファ１は先の実現例と同様に最上のセクション上で使用される。作業バッファ１は、第１の可変不透明度層上に合成され、結果は作業バッファ２に格納される。作業バッファ０及び１は、最終的に以下のようになることを除いて最上のセクションに適用されるときと同じように次の最下のセクションに適用される：

（ａ）　作業バッファ１上の作業バッファ２が作業バッファ２に格納され、

（ｂ）　次の可変不透明度層上の作業バッファ２が作業バッファ２に格納される

図６４に示す合成手法６４００を参照すると、ステップ６４０１は合成スタックから固定色を合成し、作業バッファ０において固定色を形成する。ステップ６４０２は、２つの可変色を作業バッファ１へと合成する。ステップ６４０３は、作業バッファ０を作業バッファ１上に合成し、結果を作業バッファ２に格納する。これらのステップは必然的に前の手法と一致する。ステップ６４０４において、作業バッファ１は第１の可変不透明度層上に合成され、結果は作業バッファ２に格納される。ステップ６４０５は前のように動作し、貢献度が作業バッファ０内にローカルで保持されることを除いては、次の２つの固定色を合成する。ステップ６４０６は作業バッファ２を作業バッファ０上に合成し、結果は作業バッファ２に格納される。ステップ６４０７は作業バッファ２を合成し、これは、次の可変不透明度層上の可変不透明度色であり、結果は作業バッファ２に格納される。ステップ６４０８から６４１０は、事実上、ステップ６４０１から６４０３を繰り返す。ステップ６４１１は、作業バッファ２を作業バッファ１上に合成し、結果は作業バッファ２に保持される。ステップ６４１２は、次の可変不透明度層上の作業バッファ２を処理することによって、合成プロセスを終了し、最終結果は作業バッファ２に保持される。

【０３２２】

このように、この手法は可変不透明度層を収容する一方で、合成スタックのトップダウン縦断を維持することが明らかである。方法６４００から早く抜け出すことは、先に説明したように、蓄積された透明度が定義済の閾値に到達したときに達成することができる。蓄積された不透明度を計算する際に、可変不透明度層の不透明度貢献度は、ゼロとして無事にとることができる。また、各可変不透明度層の不透明度貢献度は、例えば、最小不透明度を有する可変不透明度層内の画素を検索することによって得られた最小値であると想定しても良い。この原理の更なる拡張は、ベクトルとして（例えば、画素ごとに）スタックの合成全体にまたがって不透明度貢献度を追跡することであり、その結果、画素の蓄積された不透明度が個々に定義済の閾値に到達し、これにより早い抜け出しが起こる。

【０３２３】

７．９．トップダウンの利点

上述のトップダウン合成技術は、ボトムアップ合成技術よりも好まれる。トップダウン合成技術は、２つの層を合成するのに必要な計算の量においてより効率的である。これは、第７．６節及び第７．７節の式１及び２を参照することによって明らかである。

【０３２４】

トップダウン合成には、後続の合成動作が出力に対して明確な効果がないであろうと判定されるとすぐに合成プロセスが終了できるようにする重要な利点がある。この「早い終

了」は実際に非常によく起こり、性能向上をもたらす。

【０３２５】

８．画素生成

図５６の画素生成モジュール６１１は、単一のｚ－レベルに対して画素値のランを生成することに関わっている。

【０３２６】

合成モジュール６１２は、出力画素を生成するためにｚ－レベル塗りつぶしの明度を組み合わせる。合成モジュール６１２が組み合わせるソース明度は、明示的に定義されるか、あるいは、塗りつぶし記述の何らかの形態から生成されなければならない。単色塗りつぶしのｚ－レベルは、合成の準備ができているソース明度を明示的に定義する。傾斜（線形又は放射状）及びビットマップ画像塗りつぶしなどの複雑な塗りつぶし型のｚ－レベルは、生成する画素のラスタ空間位置に依存するようにソース明度が生成されることを必要とする。複雑な塗りつぶしのｚ－レベルに対するソース明度を判定するこのプロセスは、画素生成と呼ばれ、画素生成モジュール６１１により行なわれる。

【０３２７】

画素生成モジュール６１１の処理フローが図３９に概略的に示される。モジュール６１１には、ステップ３９３において入力として所要の塗りつぶし記述を含む複雑なｚ－レベルへの参照と共にラスタ空間における開始位置及び終了位置が提供される。ステップ３９４及び３９５において判定されるｚ－レベルの塗りつぶし型によって、合成器は以下のものに対して明度のランを生成するであろう：

－ステップ３９６における線形傾斜、

－ステップ３９７における放射状傾斜、及び

－ステップ３９８におけるビットマップ塗りつぶし

生成された明度は、合成モジュール６１２による合成に対して準備のできた合成バッファ又は塗りつぶし生成バッファへと格納される。

【０３２８】

８．１．線形傾斜画素生成

ｚ－レベルは線形傾斜塗りつぶしを有しても良い。従来技術では、線形傾斜塗りつぶしは線形ブレンドと呼ばれることが多い。線形傾斜塗りつぶしは、指定の想像線から陰影をつけられる点の垂直距離によって決まる明度を使用して表面に陰影をつけることを伴う。

【０３２９】

線形傾斜は、傾斜ルックアップテーブルを使用して記述されても良い。傾斜ルックアップテーブルは索引値のある範囲にわたる傾斜に対する明度を定義する。傾斜ルックアップテーブルは、１つ以上の傾斜ルックアップテーブルエントリの順序付けられた集合として実現されても良い。各傾斜ルックアップテーブルエントリは索引値と対応する明度とから構成される。傾斜ルックアップテーブルエントリは索引値の昇順に順序付けされる。

【０３３０】

図２５において、傾斜ルックアップテーブル２６０が示される。傾斜ルックアップテーブル２６０の索引値範囲は、０から２５５（両端含む）であることが分かる。傾斜ルックアップテーブル２６０は２つの傾斜ルックアップテーブルエントリ２６１及び２６２から構成される。傾斜ルックアップテーブルエントリの第１のエントリ２６１は、索引値６３において白色明度を定義する。傾斜ルックアップテーブルエントリの第２のエントリ２６２は索引値１９１において黒色明度を定義する。

【０３３１】

傾斜ルックアップテーブルの２つのエントリ間の索引値に対して、２つの周囲の傾斜ルックアップテーブルエントリの明度の線形補間を使用して対応する明度が計算されても良い。全ての傾斜ルックアップテーブルエントリのうちの最小の索引値よりも小さい索引値、あるいは、全ての傾斜ルックアップテーブルエントリのうちの最大の索引値よりも大きい索引値に対して、最も近い索引を有する傾斜ルックアップテーブルエントリの明度を直接使用することによって対応する明度を判定しても良い。

【０３３２】
例えば、図２５において、２つのルックアップテーブルエントリ２６１及び２６２間の索引値に対応する明度は、エントリの白色明度及び黒色明度の線形補間を使用して判定される。この例を継続すると、第１のルックアップテーブルエントリ２６１の索引値より小さい索引値、例えば、索引値５０に対しては、関連付けられる明度は最も近いルックアップテーブルエントリの明度である（このため、白色明度である）。逆に、最後のルックアップテーブルエントリ２６２の索引値より大きい索引値、例えば、索引値２５５に対しては、関連付けられる明度は最も近いルックアップテーブルエントリの明度である（このため、黒色明度である）。
【０３３３】
傾斜塗りつぶしからラスタ画素明度を生成する際には、画素の位置に対応する傾斜ルックアップテーブルへの索引を計算する必要がある。これを行なうためには、画素のラスタ空間座標位置の傾斜空間への変換が行なわれる。傾斜空間をルックアップテーブル索引範囲へとマップするためには、傾斜空間表面を定義するのが有用である。傾斜空間表面の各点は、特定の傾斜ルックアップテーブル索引へとマップする。ラスタ画素の明度は索引値及び傾斜ルックアップテーブルを使用して判定されても良い。
【０３３４】
線形傾斜に対しては、傾斜空間表面内の点の水平位置にのみ依存するように、傾斜空間表面は点を索引値へマップしても良い。これは、線形傾斜に関して傾斜空間中の位置の垂直成分を計算するする必要がないことを意味する。ラスタ空間において線形傾斜の外観を変更するには、傾斜の位置、スケール、及び回転などのパラメータを修正するのが望ましいであろう。このような効果を達成するには、傾斜－ラスタ空間変換を使用して線形傾斜空間表面をラスタ空間へと変換することが行なわれても良い。
【０３３５】
図２５は、傾斜空間起点に中心がある幅３２７６８の四角傾斜空間表面２６３の例を示す。この傾斜空間表面にまたがる明度が上述の線形傾斜の技術を使用して示される。明度は水平傾斜空間位置のみに依存する。図２５は、ラスタ空間装置２６４へと出力のためにマップされるこの傾斜空間表面２６３を示す。このマッピングは傾斜空間表面の回転及びスケーリングを伴うことが分かる。
【０３３６】
線形傾斜塗りつぶしは、索引を傾斜ルックアップテーブルへと増分的に追跡することによってレンダリングされても良い。このような技術は、ラスタ空間変換に対する傾斜に依存する３つのパラメータを事前に計算することを伴う。図３２において、出力画素明度を計算する前に、傾斜ルックアップテーブル及びラスタ空間変換から構成される線形傾斜記述があるので、ステップ３１１において以下の項目を計算することができる：
－ラスタ空間水平方向における画素ステップごとの傾斜空間水平位置における変化 ∂ｕ／∂ｘ、
－ラスタ空間垂直方向における画素ステップごとの傾斜空間水平位置における変化∂ｕ／∂ｙ 、及び
－ラスタ空間起点に対応する傾斜空間位置ｕ。
これらの事前に計算された値は、ラスタ空間変換に対する傾斜に依存し、変換が修正された場合にのみ再計算を必要とする。
【０３３７】
これらの傾斜トラッキングパラメータのグラフィカル表現は図２７において示されるであろう。図２７において、ラスタ空間Ｘ－Ｙ２６９のラスタ出力装置のディスプレイが示されるであろう。傾斜空間Ｕ－Ｖ２７０において定義される傾斜は、ラスタ装置空間２６９へとラスタ化される。ラスタ装置のラスタ走査線２６７は、走査線内の開始画素２６８で開始する傾斜塗りつぶし画素のランを含む。正の水平ラスタ空間方向における単一のラスタ画素ステップに対して、傾斜空間水平位置における対応する変化∂ｕ／∂ｘ２７１が示される。正の垂直ラスタ空間方向における単一のラスタ画素ステップに対して、傾斜空

間水平位置における対応する変化∂ｕ／∂ｙ２７３が示される。図２７において、垂直傾斜空間位置に対応する同様のパラメータ２７２及び２７４が示される。これらの値の計算は線形傾斜に必要とされない。

【０３３８】

このため、塗りつぶし生成バッファへと出力されるランを形成する１つ以上の連続画素に対する線形傾斜画素明度を生成することに進むことが可能である。生成されるラン中の左端ラスタ画素のラスタ空間座標位置(x_{start},y_{start})から判断して、図３２のステップ３１２において傾斜空間水平位置ｕを最初に初期化することが必要である。図２７において、生成されるラン中の左端ラスタ画素の例が２６８において示される。初期のｕ値は以下の公式を使用して計算されるであろう：

$$u = u_o + x_{start} \times \partial u / \partial x + y_{start} \times \partial u / \partial y$$

従って、ｕ値を直接使用することによって傾斜ルックアップテーブルへの実際の索引を判定することができる。このような索引値を使用することで、傾斜ルックアップテーブルからの対応する明度をステップ３１３で先に説明したように判定することができる。

【０３３９】

画素の明度の生成後、ステップ３１０で判定されたように、線形傾斜から生成されるランに更に画素明度が存在する場合、傾斜空間水平位置値ｕが以下の再帰関係に従ってステップ３１５において増分されても良い：

$$u_n = u_{n-1} + \partial u / \partial x$$

このプロセスは、ラン中の全ての所要の画素が生成されるまで繰り返される。これに続いて、ステップ３１４において判定されたように、同じ線形傾斜を使用してレンダリングすることが必要なランが更に存在する場合、ステップ３１２においてレンダリングする新規のランの開始画素に対して最初の傾斜空間水平位置ｕを再計算し、ステップ３１３においてランの各画素に対して画素明度を生成するプロセスを繰り返すことによって同じプロセスを繰り返しても良い。

【０３４０】

８．２．放射傾斜画素生成

ｚ－レベルは放射状傾斜塗りつぶしを有しても良い。放射状傾斜塗りつぶしは放射状ブレンドとも呼ばれる。放射状傾斜塗りつぶしは、放射状傾斜の指定の中心点から陰影をつけられる点の距離に依存する明度を使用して表面に陰影をつけることとして考えられる。放射状傾斜画素を生成するプロセスは先に説明した線形傾斜画素のプロセスに類似している。

【０３４１】

線形傾斜と同様に、放射状傾斜は１つ以上の傾斜ルックアップテーブルエントリの順序付けられた集合として実現される傾斜ルックアップテーブルを使用して記述されても良い。放射状傾斜も傾斜空間表面を利用する。傾斜空間表面の各点は特定の傾斜ルックアップテーブル索引にマップする。２つの傾斜ルックアップテーブルエントリの明度は、ルックアップテーブルに明確に定義されない索引に対して色を取得するために線形に補間されても良い。

【０３４２】

放射状傾斜に関して、傾斜空間表面の中心点からの点の距離に依存するように、傾斜空間表面は点を索引にマップする。ラスタ空間中の放射状傾斜の外観を変更するためには、中心点及び放射状傾斜のスケールなどのパラメータを修正しても良い。このような効果を達成するには、放射状傾斜空間表面はラスタ空間変換に対する傾斜を使用してラスタ空間へと変換しても良い。

【０３４３】

従来の方法で放射状傾斜をレンダリングするのは計算上コストがかかる。通常、各画素の座標を傾斜空間へと変換し、傾斜空間の中心から傾斜空間点の距離を計算し、対応する明度を参照することが必要となるであろう。しかし、放射状傾斜からラスタ画素の連続シーケンスを生成する際には、傾斜ルックアップテーブル索引の計算への増分的な手法を使

用することによって性能を改善することができる。

【０３４４】

図４２において、出力画素明度を計算する前に、傾斜ルックアップテーブル及びラスタ空間変換から構成される線形傾斜記述があるので、ステップ４１４において以下の項目を計算することができる：

（ｉ） ラスタ空間水平方向における画素ステップごとの傾斜空間水平位置における変化 ∂ｕ／∂ｘ、

（ｉｉ） ラスタ空間水平方向における画素ステップごとの傾斜空間垂直位置における変化 ∂ｖ／∂ｘ、

（ｉｉｉ） ラスタ空間垂直方向における画素ステップごとの傾斜空間水平位置における変化 ∂ｕ／∂ｙ、

（ｉｖ） ラスタ空間垂直方向における画素ステップごとの傾斜空間垂直位置における変化 ∂ｖ／∂ｙ、及び

（ｖ） ラスタ空間起点に対応する傾斜空間位置（u_o，v_o）

これらの事前に計算された値は、ラスタ空間変換に対する傾斜に依存し、変換が修正されたときに再計算のみを必要とする。これらの事前に計算された値を使用することにより、傾斜ルックアップテーブル索引の２乗（$index^2$）の増分的トラッキングが可能になる。ラスタスキャンレンダリングへの指定の実現例において、ｉｎｄｅｘはレンダリング中の画素位置に対する放射状傾斜の中心点からの距離と同等である。

これらの傾斜トラッキングパラメータのグラフィカル表現は図２７に示されるであろう。尚、図では線形傾斜塗りつぶしを示すが、内容は放射状傾斜生成にも同様に適用可能である。図２７において、ラスタ空間Ｘ－Ｙ２６９のラスタ出力装置が示される。傾斜空間Ｕ－Ｖ２７０において定義される傾斜はラスタ装置空間２６９へとラスタ化される。ラスタ装置のラスタ走査線２６７は、走査線内の開始画素２６８で開始する傾斜塗りつぶし画素のランを含む。正の水平ラスタ空間方向における単一のラスタ画素ステップに対して、傾斜空間水平位置における対応する変化∂ｕ／∂ｘ２７１及び傾斜空間垂直位置における対応する変化∂ｖ／∂ｘ２７２が示される。正の垂直ラスタ空間方向における単一のラスタ画素ステップに対して、傾斜空間水平位置における対応する変化∂ｕ／∂ｙ２７３及び傾斜空間垂直位置における対応する変化∂ｖ／∂ｙ２７４が示される。

このため、塗りつぶし生成バッファへと出力されるランを形成する１つ以上の連続画素に対する画素明度を生成することに進むことが可能である。開始画素のラスタ空間座標位置（x_{start}，y_{start}）から判断して、対応する最初の傾斜空間位置（u_{start}，v_{start}）の計算が行なわれる。図２７において、この開始ラスタ画素の例が２６８において示される。対応する最初の傾斜空間位置は以下の公式を使用して計算されるであろう：

$$u_{start} = u_o + x_{start} \times \partial u / \partial x + y_{start} \times \partial u / \partial y$$

$$v_{start} = v_o + x_{start} \times \partial v / \partial x + y_{start} \times \partial v / \partial y$$

これらの計算から、傾斜ルックアップテーブルへの索引を維持するのに使用される最初のトラッキング値が図４２のステップ４１５で判定される。具体的には、この２乗された索引値∂（$index^2$）／∂ｘに対する最初の２乗された傾斜ルックアップテーブル索引（$index^2$）及び最初の増分値が以下のように計算される：

$$index^2 = u^2_{start} + v^2_{start}$$

$$\partial(index^2)/\partial x = 2 \times (u_{start} \times \partial u/\partial x + v_{start} \times \partial v/\partial x) + (\partial u/\partial x)^2 + (\partial v/\partial x)^2$$

（$index^2$）値が計算されると、先に説明したように傾斜ルックアップテーブルを使用してステップ４１６において対応する明度が参照できるように平方根が取られても良い。

【０３４５】

画素の明度の生成後、ステップ４１７において判定されたように放射状傾斜から生成される画素のランに更なる画素明度が存在する場合、$index^2$ 及び∂（$index^2$）

／∂ｘ値は、以下の再帰関係に従ってステップ４１８において増分されるであろう：
ｉｎｄｅｘ$^{2}{}_{n}$＝ｉｎｄｅｘ$^{2}{}_{n-1}$＋∂（ｉｎｄｅｘ$^{2}{}_{n-1}$）／∂ｘ
∂（ｉｎｄｅｘ$^{2}{}_{n}$）／∂ｘ＝∂（ｉｎｄｅｘ$^{2}{}_{n-1}$）／∂ｘ＋２×｛（∂ｕ／∂ｘ）^{2}＋（∂ｖ／∂ｘ）^{2}｝
尚、値２×｛（∂ｕ／∂ｘ）^{2}＋（∂ｖ／∂ｘ）^{2}｝は一定であり、事前に計算されても良い。

【０３４６】
このプロセスは、現在の画素のランの全ての所要の画素が生成されるまで繰り返される。これに続き、ステップ４１９において判定されるように、同じ放射状傾斜を使用する生成を必要とする画素のランが更に存在する場合、ステップ４１５に戻り、新規のランの開始画素に対する最初の索引トラッキング値を再度計算することによって、同じプロセスが繰り返されるかもしれない。要約すると、最初のトラッキング値が計算されると、この技術により放射状傾斜は、画素ごとに２つの加算及び平行根を行なうことによって画素のランに沿ってレンダリングすることができる。

【０３４７】
平方根計算は、単一の６５Ｋエントリルックアップテーブルを使用して、より効率的であり且つ同じ精度では、２５６エントリルックアップテーブルを使用して、M.D. Ercegovac等による「Reciprocation, Square Root, inverse Square Root, and Some Elementary Functions Using Small Multipliers」（IEEE Transactions on Computers, Vol. 49, No. 7, July 2000）に記載されるプロセスに従って迅速に行なわれるであろう。

【０３４８】
記載された方法では、ラスタ走査順ディスプレイの最上部から最下部へと放射状傾斜の各画素をレンダリングする。すなわち、この方法では、ディスプレイの最上部から最下部までレンダリングし、左から右まで走査線内の各画素をレンダリングする。適切な修正を行なった同じ技術も、動作の他のラスタ走査順に等しく適用可能である。この点に関して、ラスタ走査順は２次元空間の任意の１次元走査として定義しても良い。ラスタ走査順は左上から右下へと行なわれる。

【０３４９】
尚、上述の放射状傾斜生成方法は、代替の適用例にも適している。一般的に、説明された方法は、２次元入力値の集合から結果値の集合を生成するのに適している。入力値のシーケンスは直線に位置し、各結果値は関数入力により記述されるベクタの大きさの関数である。すなわち、説明された方法は、位置の個別の集合において放射状傾斜の値を計算するのに適している。ここで、位置の個別の集合はラスタ走査順である。動径関数は、２Ｄ空間内の特定の位置からの入力の距離の関数である結果値（すなわち、値集合）へとマップする２つの変数の関数として定義されても良い。

【０３５０】
８．３．ビットマップ画像画素生成
ｚ－レベルはビットマップ画層塗りつぶしを有しても良い。これにより、ｚ－レベルの塗りつぶしはある色空間における点サンプルを定義する画素データの列から構成されるデジタル画像に基づいても良いことを意味する。ビットマップ画像の画素データは、任意のフレームの前で更新されても良い。このような技術により、動画ビデオに類似する動画ビットマップ画像塗りつぶし効果を促進する。ビットマップ画像塗りつぶしは、傾斜塗りつぶしをレンダリングするために記述されたものと同様の手法を伴う。

【０３５１】
ラスタ空間画素位置をビットマップ画素値へとマップするためには、ビットマップ空間表面をビットマップ画像を含む表面として定義することが適切である。ビットマップ空間表面の起点はビットマップ画像の右上画素に対応する。同様に、傾斜空間表面はラスタ空間へと変換することができ、ビットマップ空間表面はラスタ空間へと変換しても良い。これにより、ビットマップに対して平行移動、スケーリング、回転、曲げを行なうことができる。

【０３５２】
ビットマップ画像塗りつぶしに関して、ラスタ空間の各画素位置は、何らかの明度へとマップする。ビットマップ画像内の点にマップするラスタ空間画素位置は、ビットマップ画像データにより定義されるように対応する画素明度を使用しても良い。ビットマップ画像の外側の点にマップするラスタ空間画素位置は、種々の手段を使用して画素明度へとマップするように定義されても良い。例えば、ビットマップ画像の外側の点にマップするラスタ空間画素位置は、最も近いビットマップ画像画素にマップするように定義されても良い。別の例は、タイル表示のビットマップ画像効果を作成するように、ラスタ空間画素位置をビットマップ画素へとマップするものであろう。これにより、ビットマップ画像はラスタ空間にわたって無限に繰り返される。

【０３５３】
図４において、出力画素明度を計算する前に、ビットマップ画像画素データ及びラスタ空間変換へのビットマップから構成されるビットマップ画像記述があるので、ステップ４４において以下の項目を計算するのが適切である：
（ｉ） ラスタ空間水平方向における画素ステップごとのビットマップ空間水平位置における変化∂ｕ／∂ｘ、
（ii） ラスタ空間水平方向における画素ステップごとのビットマップ空間垂直位置における変化∂ｖ／∂ｘ、
（iii） ラスタ空間垂直方向における画素ステップごとのビットマップ空間水平位置における変化∂ｕ／∂ｙ、
（iv） ラスタ空間垂直方向における画素ステップごとのビットマップ空間垂直位置における変化∂ｖ／∂ｙ、及び
（ｖ） ラスタ空間起点に対応するビットマップ空間位置(u_0,v_0)
これらの事前に計算された値は、ラスタ空間変換に対するビットマップに依存し、変換が修正されたときに再計算のみを必要とする。これらの事前に計算された値を使用することにより、ビットマップ画像データの現在の画素のアドレス（ａｄｄｒｅｓｓ）の増分的トラッキングが可能になる。

【０３５４】
続いて、塗りつぶし生成バッファに出力すべきランを形成する１つ以上の連続画素に対する画素明度の生成が行なわれるであろう。開始画素のラスタ空間座標位置(x_{start},y_{start})から判断して、対応する最初のビットマップ空間位置(u_{start},v_{start})の計算が行なわれる。これは以下の公式を使用して計算されるであろう：

$$u_{start}=u_0+x_{start}\times\partial u/\partial x+y_{start}\times\partial u/\partial y$$
$$v_{start}=v_0+x_{start}\times\partial v/\partial x+y_{start}\times\partial v/\partial y$$

これらの計算から、ビットマップ画像データへの最初のアドレス（ａｄｄｒｅｓｓ）がステップ４５で判定される。アドレスｂｉｔｍａｐｂａｓｅで開始する走査順を伴うビットマップ画像に関して、ビットマップ画像データは画素ごとにｂｐｐバイトであり、ｒｏｗｓｔｒｉｄｅバイトの列ストライドを有し、ビットマップ画像データへの最初のアドレスは以下のように計算されても良い：

$$address=bitmapbase+v_{start}\times rowstride+u_{start}\times bpp$$

ａｄｄｒｅｓｓ値の計算が終わると、そのアドレスの画素明度は、現在の出力ラスタ画素に対する出力画素明度としてステップ４６において使用されても良い。画素の明度の生成後、ステップ４７において判定されるように、ビットマップ画像から生成される現在の画素のランに更なる画素明度が存在する場合、ａｄｄｒｅｓｓ値は以下の再帰関係に従ってステップ４８において増分されても良い：

$$address_n=address_{n-1}+\partial v/\partial x\times rowstride+\partial u/\partial x\times bpp$$

尚、ａｄｄｒｅｓｓが増加する値は定数として事前に計算されても良い。このプロセスは、現在の画素のランの全ての画素が生成されるまで繰り返される。これに続いて、ステップ４９において判定されるように同じビットマップ画像を使用するレンダリングを必要とする更なる画素のランが存在する場合、ステップ４５において画素の次のランの開始画

素に対する最初の索引ａｄｄｒｅｓｓを再度計算することによって同じプロセスが繰り返されても良い。要約すると、このモジュールにより、画素ごとに２つの加算のみを行なうことによって出力画素のランに沿ってレンダリングできるようになる。

【０３５５】

９．画素抽出

図５６の画素抽出モジュール６１８は、ラスタ化プロセスの最終ステップを行なって終了した画素を出力する。これらの画素は、直接表示装置へ出力することも、後で表示装置に出力できるようにフレームバッファメモリへと格納することもできる。画素出力の目標はシステム設計の選択である。一部のシステムは、画素出力目標のうちの１つのみを使用するであろうが、他のシステムは出力目標に関してランタイム構成可能であっても良い。

【０３５６】

出力画素の精度が画素抽出モジュール６１８に供給される明度の精度よりも低いシステムでは、画素抽出モジュール６１８はハーフトーン処理の任意のステップも行なって良い。例えば、画素抽出モジュール６１８がチャネルごとに８ビットの色情報（ＲＧＢ８：８：８）を受信するがチャネルごとに５ビット（ＲＧＢ５：５：５）の出力画素を受信するシステムでは、ハーフトーン処理が実施されても良い。

【０３５７】

９．１．入力データ

画素抽出モジュール６１８は、入力データとして「スパン」を受け入れる。スパンは、同じ色を有する表示装置走査順の一連の連続画素である。スパンの長さは、１画素の小さささであっても、画素のフルディスプレイの大きさであっても良い。各スパンはスパンの色及び画素におけるその長さという２つの値として画素抽出モジュール６１８へと入力される。

【０３５８】

９．２．フレームバッファへの出力

動作のこのモードでは、画素抽出モジュール６１８はフレームバッファメモリへと画素を出力する。このモードの動作を図３７のフローチャートを参照して説明する。ステップ３７０において、処理が開始する。ステップ３７１において、フレームバッファ位置変数Ｘ及びＹの双方がゼロに初期化される。これは、フレームバッファの右上の出力位置を示す。ステップ３７８において、画素抽出モジュール６１８は表示すべき次のスパンを受け入れる。ステップ３７７において、変数Ｎ及びＣにはそれぞれスパンの長さ及び色が割り当てられる。ステップ３７６が続く。ステップ３７６において、画素抽出モジュール６１８はフレームバッファの位置(X,Y)における画素の色を色Ｃに設定する。ステップ３７５において、フレームバッファ位置Ｘ変数が増分される。ステップ３７９において、フレームバッファ変数Ｘはフレームバッファの幅と比較される。Ｘがフレームバッファの幅に到達又は幅を超える場合、処理はステップ３７４へと移る。幅に至らない場合、処理はステップステップ３６９で継続する。ステップ３６９において、スパン長さカウンタ変数Ｎが減分される。ステップ３８０において、スパン長さカウンタ変数Ｎがゼロと比較される。ゼロの場合、処理はステップ３７８へと移る。ゼロでない場合、ステップ３７６へと移る。図３７のステップ３７４において、フレームバッファ変数Ｘがゼロに設定され、フレームバッファ変数Ｙが増分される。ステップ３７３において、フレームバッファ変数Ｙはフレームバッファ高さと比較される。Ｙが高さ以上である場合、処理はステップ３７２で終了し、未満である場合、ステップ３６９で継続する。

【０３５９】

９．３．ディスプレイへの直接出力

この動作モードでは、画素抽出モジュール６１８は、画素を直接表示装置へと出力する。画素抽出モジュール６１８に対するスパン情報の到達は、画素が表示装置上に表示される速度と非同期であることが多いので、画素抽出モジュール６１８はＦＩＦＯ（先入れ先出し）バッファを使用してスパンを表示装置に出力する前に待ち行列に入れる。ＦＩＦＯバッファは、入力スパンのタイミングと出力画素のタイミングとの間で弾性の尺度を提供

する。

【０３６０】

この動作モードを図３８（ａ）及び図３８（ｂ）に示すフローチャートを参照して説明する。スパンをＦＩＦＯバッファに受け入れるプロセスは図３８（ａ）のステップ３８８から３８９により説明され、スパンをＦＩＦＯから出力するプロセスは図３８（ｂ）のステップ３８２から３８３により説明される。これらの２つのステップのタイミングはＦＩＦＯバッファにより大幅に非同期に維持される。

【０３６１】

図３８（ａ）に示すように、ステップ３８８において、スパン（画素のラン）を受け入れるプロセスが開始する。ステップ３９２において、スパンは画素抽出モジュール６１８により受け入れられる。ステップ３９１において、ＦＩＦＯバッファが満杯か否かが検査される。満杯の場合、処理はステップ３９０へと移るが、満杯でない場合、処理はステップ３８９で続けられる。ステップ３９０において、処理はＦＩＦＯバッファ満杯状態が過ぎるのを待つ。ステップ３８９において、ステップ３９２で受け入れられたスパンがＦＩＦＯバッファに格納される。

【０３６２】

図３８（ｂ）に示すように、ステップ３８２において、画素を出力するプロセスが開始する。ステップ３８７において、スパンがＦＩＦＯから読み出される。ステップ３８６においてスパンの長さ及び色が変数Ｎ及びＣにそれぞれ格納される。ステップ３８５において、次の画素が出力される時期であるか否かを判定するための検査が行なわれる。まだ時期でない場合、処理は現在のステップを繰り返す。次の画素が出力される時期である場合、処理はステップ３８４へと移る。ステップ３８４において、色Ｃの画素がディスプレイに出力される。ステップ３８１において、変数Ｎが減分される。ステップ３８３において、変数Ｎがゼロと比較される。ゼロに等しい場合、処理はステップ３８７へと移る。等しくない場合、処理はステップ３８５で継続される。

【０３６３】

９．４．ハーフトーン処理

図２９は、画素抽出モジュール６１８により使用されるハーフトーン技術を示す。この技術は誤差拡散の一種である。図２９（ａ）は、次の走査線上の４つの画素を含む５つの隣接する画素に画素の量子化誤差を拡散する従来技術の誤差拡散技術を示す。図２９（ｂ）は画素抽出モジュール６１８の誤差拡散技術を示し、この技術では、画素の量子化誤差を現在の画素と同じ走査線上の１つの隣接する画素にのみ拡散する。図２９（ｃ）は画素抽出モジュール６１８の誤差拡散システム２９００のシステムブロック図である。システム２９００は単純な１次負帰還制御システムである。現在の画素の５ビット精度までの量子化によりもたらされる誤差が次の画素に追加され、負帰還がもたらされる。システム２９００は、図２９（ｃ）に示すように結合された量子化器２９０１及び加算器２９０２を使用してハードウェアで構成されても良い。あるいは、図３０に示す方法を使用してソフトウェアで誤差拡散システムが構成されても良い。誤差拡散アルゴリズムは説明を簡略にするために１つの色チャネルに関してのみ説明する。実際のシステムは、全ての色チャネルが同様に処理される。

【０３６４】

図３０は、ハーフトーンアルゴリズムのフローチャート記述を提供する。ステップ２９２においてハーフトーン処理が開始する。ステップ２８３において変数が初期値に設定され、ステップ２９０において８ビット精度の画素純度が読まれる。ステップ２８９において、ステップ２９０で読まれた画素純度はＥｒｒｏｒ変数を付加される。結果は変数Ｄｅｓｉｒｅｄに格納される。これにはステップ２８８が続き、変数Ｄｅｓｉｒｅｄは５ビット精度まで量子化され、変数Ａｃｔｕａｌに格納される。８ビットから５ビットまでの量子化は、３ビットを右に論理シフトすることによって達成される。ステップ２８７において量子化明度Ａｃｔｕａｌが出力される。次に、ステップ２８６において実際の出力色Ａｃｔｕａｌが８ビット値として再表現される。これは、論理的に２ビット右シフトしたＡ

ｃｔｕａｌに３ビット左シフトしたＡｃｔｕａｌを加えることによって達成される。結果値は変数ＡｃｔＥｃｈｏに格納される。ステップ２８５において量子化誤差はＡｃｔＥｃｈｏをＤｅｓｉｒｅｄから差し引くことによって計算される。結果は変数Ｅｒｒｏｒに格納される。ステップ２８４において最後の画素が到達したか否かを確認するために検査が行なわれる。到達していた場合、処理はステップ２９３で終了する。ステップ２９１において変数Ｘが増分される。

【０３６５】

１０．実現例

上述のＴＣＩＥシステム６９９は、ハードウェア及びソフトウェアのいずれかにより実現されても、あるいは、これらの組み合わせにより実現されても良い。ハードウェア実現例では、図５６の各モジュールは１つ以上のＡＳＩＣ装置へと統合され、携帯電話機のような目標装置へと組み込まれても良い。ソフトウェア実現例では、このようなソフトウェアは汎用に動作するように構成されても良いが、通常は、限られたコンピューティングシステムでは、限られた容量のプロセッサが使用されている。ハイブリッド実現例は、ソフトウェアで実現される表示リストコンパイラ６０８、あるいは、ハードウェアで実現されるレンダリングエンジン６１０又はその一部に向いている。このような例では、通常、データ（画像）入力の汎用性及びレンダリング速度の高速性がもたらされる。更に、システム６９９がシン・クライアント上での利用のために開発されているにも関わらず、このようなものでは、デスクトップコンピュータ、セットトップボックスなどの重要なコンピューティングリソースを有する装置に適用される同じイメージング／レンダリング手法を妨げない。

【０３６６】

上述のシン・クライアントイメージングの方法は、図６０に示すようなシステム６０００を使用して実施することができる。図６０において、説明された各プロセスはシステム６０００内で実行されるアプリケーションプログラムなどのソフトウェアとして完全に実現される。特に、図５６の方法の各ステップは、コンピューティング装置６００１により実行されるソフトウェアの命令により実施される。命令は、それぞれが１つ以上の特定のタスクを実行する１つ以上のコードモジュールとして形成されても良い。また、ソフトウェアは２つの別々の部分に分割されても良く、第１の部分は指定のイメージング／レンダリング方法を行ない、第２の部分は第１の部分とユーザとのユーザインタフェースを管理する。ソフトウェアは、例えば、後述の記憶装置を含むコンピュータ可読な媒体に格納されても良い。ソフトウェアは、コンピュータ可読な媒体からコンピュータへとロードされ、コンピュータにより実行される。このようなソフトウェア又はコンピュータプログラムを記録したコンピュータ可読な媒体はコンピュータプログラム製品である。コンピュータにおけるコンピュータプログラム製品の使用は、シン・クライアントイメージングのための有利な装置を実現するのが好ましい。

【０３６７】

システム６０００は、通信リンク６０２１により相互接続されたコンピューティング装置６００１及びネットワーク６０２０を含む。装置６００１は、例えば、携帯電話機であり、この場合、ネットワーク６０２０はセルラーネットワークなどの携帯加入電話網であるだろう。装置６００１が、例えば、複写機又は印刷機である場合、ネットワーク６０２０はインターネット又は構内通信網（ＬＡＮ）又は広域通信網（ＷＡＮ）などの別のネットワークシステムであるかもしれない。コンピューティング装置６００１は、キーボード、タッチパネル、及びマウス又はトラックボールなどのポインティングデバイスを組み込むであろうユーザ入力インタフェース６００２を含む。コンピューティング装置６００１は、実現される例によって携帯電話送受信素子又は複写機エンジンなどの機能ハードウェア６０１２を組み込む。機能インタフェース６００８は、コンピューティング装置６００１がネットワーク６０２０と通信を行なうと共に、第１の機能的な役割を果たすために使用される。

【０３６８】

コンピューティングモジュール６００１は、通常、少なくとも１つのプロセッサ部６００５と、揮発性及び不揮発性メモリとして機能するメモリ部６００６及び６０１０を含む。揮発性メモリ６００６は半導体のランダムアクセスメモリ（ＲＡＭ）から形成され、不揮発性メモリは読出し専用メモリ（ＲＯＭ）、ハードディスクドライブ、又は光学メモリプラットフォームから形成される。例えば、メモリ６００６は図５６のメモリ６０９として動作しても良い。また、モジュール６００１は、ディスプレイ６０１４に接続する画像出力バッファ６００７及びユーザインタフェース６００２のためのＩ／Ｏインタフェース６０１３を含む複数の入出力（Ｉ／Ｏ）インタフェースを含む。コンピューティング装置６００１のモジュール６００２から６０１４は、通常、相互接続バス６００４を介して、当業者に既知のコンピューティング装置６００１の従来の動作モードになるように通信を行なう。

【０３６９】

一般的に、不揮発性メモリ６０１０には、シン・クライアントイメージング用のアプリケーションプログラムが常駐し、実行時には、プロセッサ６００５により読出し／制御が行なわれる。プログラムの中間記憶装置及びネットワーク６０２０から取り出されたデータは、揮発性メモリ６００６を使用し、場合によってはハードディスクドライブを含む不揮発性メモリと協働させるように使用して達成されても良い。アプリケーションプログラムは不揮発性メモリ６０１０においてエンコードされた形でユーザに提供される場合もあれば、インタフェース６００８を介してネットワーク６０２０からユーザが読み取る場合もある。更に、このソフトウェアは他のコンピュータ可読な媒体からコンピューティング装置６００１へとロードすることもできる。ここで使用される「コンピュータ可読な媒体」という用語は、実行及び／又は処理のためにコンピュータシステム６０００に命令及び／又はデータを供給することに関わる記憶媒体又は伝送媒体を指し示す。記憶媒体の例としては、フロッピーディスク、磁気テープ、ＣＤ－ＲＯＭ、ハードディスクドライブ、ＲＯＭ又は集積回路、光磁気ディスク、又はＰＣＭＣＩＡカードなどのコンピュータ可読なカードが含まれ、コンピューティング装置６００１の内側又は外側にあるかを問わない。伝送媒体の例としては、別のコンピュータ装置又はネットワーク化装置へのネットワーク接続と同様に無線又は赤外線の伝送チャネル、及び電子メール送受信並びにウェブサイトなどに記録された情報を含むインターネット並びにイントラネットが含まれる。

【０３７０】

携帯電話機の例では、ネットワーク６０２０が装置６００１にレンダリングすべきゲームの各フレームに対する表示リストを伝送するように、個人設定及びリアルタイムゲーム再生入力などのユーザ入力に従って、表示リストコンパイラ６０８により操作されるグラフィックスベースの動画ゲームをするためにこれが使用されても良い。その結果がレンダリングエンジン６１０に配布される。レンダリングエンジン６１０は出力画素画像をバッファ６００７に配布し、続いてディスプレイ６０１４に配布される。コンパイラ６０８及びレンダラ６１０はプロセッサ６００５により呼び出され実行される各ソフトウェアモジュールにより形成されても良い。

【産業上の利用可能性】

【０３７１】

上述の構成は、コンピュータ及びデータ処理産業に適用可能であり、特に、画像再生が必要であり、画像再生及びレンダリングのためのコンピューティングリソースが限られている場合に適用可能である。

【０３７２】

以上の説明では、本発明の幾つかの実施例について述べた。本発明の趣旨から逸脱することなく、修正及び／又は変更を行なうことができる。各実施例は例示のためのものであり、限定するためのものではない。

【図面の簡単な説明】

【０３７３】

【図１（ａ）】、　小画素レベルでのアクティブ稜線の判定を示す図。

【図１（ｂ）】、　小画素レベルでのアクティブ稜線の判定を示す図。
【図１（ｃ）】小画素レベルでのアクティブ稜線の判定を示す図。
【図２（ａ）】、　小画素レベルでの走査線に対する稜線処理の一例を示す図。
【図２（ｂ）】、　小画素レベルでの走査線に対する稜線処理の一例を示す図。
【図２（ｃ）】、　小画素レベルでの走査線に対する稜線処理の一例を示す図。
【図２（ｄ）】小画素レベルでの走査線に対する稜線処理の一例を示す図。
【図３】図２３のステップ２５２の交差メッセージを生成する方法のフローチャート。
【図４】ビットマップ画像画素生成の方法を示す図。
【図５】修正Bresenhamのアルゴリズムの Ｃ言語ソースコードを示す図。
【図６】ボトムアップ合成プロセスのフローチャート。
【図７】図４１のトラッキングパラメータ生成プロセスに関連するベジェ曲線の一例を示す。
【図８】図２の例において小画素値に対するワインディングカウントの処理を示す図。
【図９（ａ）】、　異なる合成シナリオを示す図。
【図９（ｂ）】異なる合成シナリオを示す図。
【図１０】異なる合成シナリオを示す図。
【図１１】図１３のステップ１１３のフローチャート。
【図１２（ａ）】、　カバレッジ動作及びＡ－ｂｕｆｆｅｒ動作のグラフィカルな表示を提供する図。
【図１２（ｂ）】、　カバレッジ動作及びＡ－ｂｕｆｆｅｒ動作のグラフィカルな表示を提供する図。
【図１２（ｃ）】、　カバレッジ動作及びＡ－ｂｕｆｆｅｒ動作のグラフィカルな表示を提供する図。
【図１２（ｄ）】カバレッジ動作及びＡ－ｂｕｆｆｅｒ動作のグラフィカルな表示を提供する図。
【図１３】ここで使用される合成手法の高レベルなフローチャート。
【図１４】Bresenhamのアルゴリズムの動作を示す円の図。
【図１５】順序付けされた１組の座標を順序付けされた稜線の集合へと変換する図４６のステップ４６５のフローチャート。
【図１６（ａ）】、　２次元基本要素を組み合わせてグラフィカルオブジェクトを形成する方法を示す図。
【図１６（ｂ）】、　２次元基本要素を組み合わせてグラフィカルオブジェクトを形成する方法を示す図。
【図１６（ｃ）】２次元基本要素を組み合わせてグラフィカルオブジェクトを形成する方法を示す図。
【図１７（ａ）】、　スプライトと変換行列との間の関係を示す図。
【図１７（ｂ）】スプライトと変換行列との間の関係を示す図。
【図１８（ａ）】、　ストロークのエンドキャップを示す図。
【図１８（ｂ）】ストロークのエンドキャップを示す図。
【図１９】汎用的な楕円を示す図。
【図２０】楕円の下方にある円の判定を示す図。
【図２１】交差点におけるアクティブ稜線の再順序付けの一例を示す図。
【図２２（ａ）】、　図５６の稜線処理モジュールに対するデータフローを示す図。
【図２２（ｂ）】図５６の稜線処理モジュールに対するデータフローを示す図。
【図２３】単一の走査線に対する稜線処理を示す図２４のステップ２５７のフローチャート。
【図２４】単一のフレームに対する稜線処理を示すフローチャート。
【図２５】傾斜塗りつぶし参照の動作を示す図。
【図２６】グリフに対する交差メッセージを示す図。
【図２７】種々の傾斜トラッキングパラメータを示す図。

【図２８（ａ）】、　数種類のエンドキャップに対する種々の塗りつぶしを示す図。
【図２８（ｂ）】数種類のエンドキャップに対する種々の塗りつぶしを示す図。
【図２９（ａ）】、　２つの従来技術のエラー拡散手法と図５６の画素抽出モジュールにより使用される誤差拡散手法とをそれぞれ示す図。
【図２９（ｂ）】、　２つの従来技術のエラー拡散手法と図５６の画素抽出モジュールにより使用される誤差拡散手法とをそれぞれ示す図。
【図２９（ｃ）】２つの従来技術のエラー拡散手法と図５６の画素抽出モジュールにより使用される誤差拡散手法とをそれぞれ示す図。
【図３０】モジュール６１８の誤差拡散プロセス（ハーフトーン処理）のフローチャート。
【図３１（ａ）】、　スプライトとローカルグラフィックオブジェクトとの間のｚ－レベル関係を示す図。
【図３１（ｂ）】、　スプライトとローカルグラフィックオブジェクトとの間のｚ－レベル関係を示す図。
【図３１（ｃ）】スプライトとローカルグラフィックオブジェクトとの間のｚ－レベル関係を示す図。
【図３２】傾斜参照テーブルへの索引を判定するためのフローチャート。
【図３３（ａ）】、　図１６のオブジェクトを作成するのに使用される左右の塗りつぶしを示す図。
【図３３（ｂ）】、　図１６のオブジェクトを作成するのに使用される左右の塗りつぶしを示す図。
【図３３（ｃ）】図１６のオブジェクトを作成するのに使用される左右の塗りつぶしを示す図。
【図３４】このレンダリングシステムで使用される種々の像空間を示す図。
【図３５（ａ）】、　モーフィングプロセスにおける１組の座標を示す図。
【図３５（ｂ）】、　モーフィングプロセスにおける１組の座標を示す図。
【図３５（ｃ）】モーフィングプロセスにおける１組の座標を示す図。
【図３６（ａ）】、　ストロークされた経路の一例を示す図。
【図３６（ｂ）】、　ストロークされた経路の一例を示す図。
【図３６（ｃ）】ストロークされた経路の一例を示す図。
【図３７】フレームバッファメモリに画素を出力するための画素抽出モジュールの動作を示すフローチャート。
【図３８（ａ）】、　ディスプレイに直接出力するための画素抽出モジュールの動作を示すフローチャート。
【図３８（ｂ）】ディスプレイに直接出力するための画素抽出モジュールの動作を示すフローチャート。
【図３９】画素生成モジュールの処理フロー表現の図。
【図４０】画素のランの出力色を生成する処理のフローチャート。
【図４１】２次ベジェ曲線に対するトラッキングパラメータの生成のためのフローチャート。
【図４２】増分的な手法の放射状傾斜判定のフローチャート。
【図４３（ａ）】、　ノンゼロワインディング、ネガティブワインディング、及び奇偶ワインディングからの異なる塗りつぶし結果を示す図。
【図４３（ｂ）】ノンゼロワインディング、ネガティブワインディング、及び奇偶ワインディングからの異なる塗りつぶし結果を示す図。
【図４４】表示リストにおける絶対深度の計算を示す図。
【図４５（ａ）】、　結合をストロークするための稜線管理を示す図。
【図４５（ｂ）】結合をストロークするための稜線管理を示す図。
【図４６】モーフィング／変換／ストローキングモジュールにより各順序付けられた座標に対して実行される処理を示すフロー図。

【図４７（ａ）】、　基数ソートアルゴリズムの動作を示す図。
【図４７（ｂ）】、　基数ソートアルゴリズムの動作を示す図。
【図４７（ｃ）】、　基数ソートアルゴリズムの動作を示す図。
【図４７（ｄ）】、　基数ソートアルゴリズムの動作を示す図。
【図４７（ｅ）】、　基数ソートアルゴリズムの動作を示す図。
【図４７（ｆ）】基数ソートアルゴリズムの動作を示す図。
【図４８】小画素レベルでの不透明度の貢献度を示す図。
【図４９（ａ）】、　ストローキング稜線の例を示す図。
【図４９（ｂ）】、　ストローキング稜線の例を示す図。
【図４９（ｃ）】、　ストローキング稜線の例を示す図。
【図４９（ｄ）】、　ストローキング稜線の例を示す図。
【図４９（ｅ）】、　ストローキング稜線の例を示す図。
【図４９（ｆ）】、　ストローキング稜線の例を示す図。
【図５０（ａ）】、　ストローキング稜線及び変換稜線の例を示す図。
【図５０（ｂ）】、　ストローキング稜線及び変換稜線の例を示す図。
【図５０（ｃ）】、　ストローキング稜線及び変換稜線の例を示す図。
【図５０（ｄ）】、　ストローキング稜線及び変換稜線の例を示す図。
【図５０（ｅ）】ストローキング稜線及び変換稜線の例を示す図。
【図５１（ａ）】、　ストローキング稜線に対する左右の塗りつぶしを示す図。
【図５１（ｂ）】、　ストローキング稜線に対する左右の塗りつぶしを示す図。
【図５１（ｃ）】、　ストローキング稜線に対する左右の塗りつぶしを示す図。
【図５１（ｄ）】、　ストローキング稜線に対する左右の塗りつぶしを示す図。
【図５１（ｅ）】ストローキング稜線に対する左右の塗りつぶしを示す図。
【図５２（ａ）】、　ｚ－レベル活動化テーブルの動作を示す図。
【図５２（ｂ）】ｚ－レベル活動化テーブルの動作を示す図。
【図５３（ａ）】、　ｚ－レベル活動化テーブルの再構成の例を示す図。
【図５３（ｂ）】ｚ－レベル活動化テーブルの再構成の例を示す図。
【図５４（ａ）】、　変動する関心のｚ－レベルに対するｚ－レベル活動化テーブルを更新する例を示す図。
【図５４（ｂ）】変動する関心のｚ－レベルに対するｚ－レベル活動化テーブルを更新する例を示す図。
【図５５】トップダウン合成プロセスのフローチャート。
【図５６】本開示によるシン・クライアントイメージングエンジンを表現する概略的なブロック図。
【図５７（ａ）】、　３つのワインディング規則によるＳ－ｂｕｆｆｅｒからＡ－ｂｕｆｆｅｒへの変換を示す図。
【図５７（ｂ）】３つのワインディング規則によるＳ－ｂｕｆｆｅｒからＡ－ｂｕｆｆｅｒへの変換を示す図。
【図５８】ｚ－レベル活動化モジュールの動作を示すフローチャート。
【図５９】ｚ－レベル活動化モジュールにより維持される関心あるｚ－レベルのリストを示す図。
【図６０】ここで説明する一部の構成が実施されるコンピュータ構成の概略的なブロック図。
【図６１（ａ）】、　合成スタックのトップダウン及びボトムアップ分割化を使用する合成手法を示す図。
【図６１（ｂ）】合成スタックのトップダウン及びボトムアップ分割化を使用する合成手法を示す図。
【図６２】合成スタックのトップダウン及びボトムアップ分割化を使用する合成手法を示す図。
【図６３】経路の個別の部分に適用される異なるストロークを示す図。

【図６４】トップダウン合成の代替の形態を示す図。
【図６５Ａ】、　走査線にまたがるｚ－レベル活動化テーブルの更新を示す図。
【図６５Ｂ】走査線にまたがるｚ－レベル活動化テーブルの更新を示す図。
【図６６】グリフ稜線に対する交差メッセージ生成を示す図。
【図６７】楕円セグメントをｙ縦座標において単調なセグメントへと分割するプロセスを示す図。
【図６８Ａ】画像フレームのアニメーションシーケンスを示す図。
【図６８Ｂ】画像フレームのアニメーションシーケンスを示す図。
【図６８Ｃ】画像フレームのアニメーションシーケンスを示す図。
【図６９Ａ】図６８Ａから図６８Ｃのアニメーションシーケンスの生成に対する新規の静止稜線バッファの処理を示す図。
【図６９Ｂ】図６８Ａから図６８Ｃのアニメーションシーケンスの生成に対する新規の静止稜線バッファの処理を示す図。
【図６９Ｃ】図６８Ａから図６８Ｃのアニメーションシーケンスの生成に対する新規の静止稜線バッファの処理を示す図。
【図７０】ゼロ交差メッセージに対するＳ－ｂｕｆｆｅｒの更新を示す図。
【図７１】図５８に類似するが、レベルをリストから破棄するためのｚ－レベル活動化モジュールの動作を示すフローチャート。

【図１（ａ）】

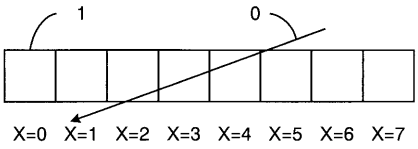


Fig. 1(a)

【図１（ｂ）】

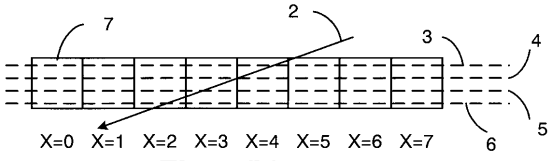


Fig. 1(b)

【図１（ｃ）】

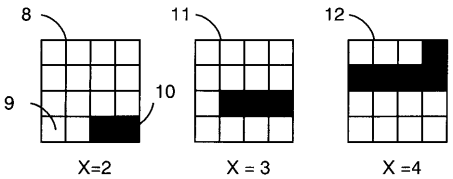


Fig. 1(c)

【図２（ａ）】

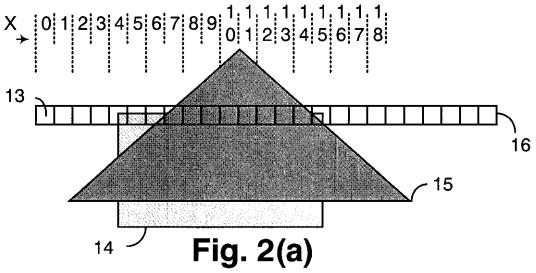


Fig. 2(a)

【図２（ｂ）】

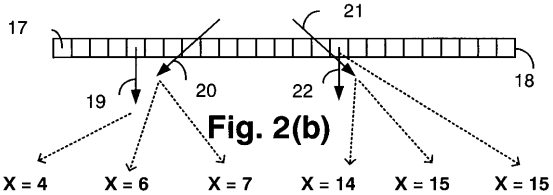


Fig. 2(b)

【図２（ｃ）】

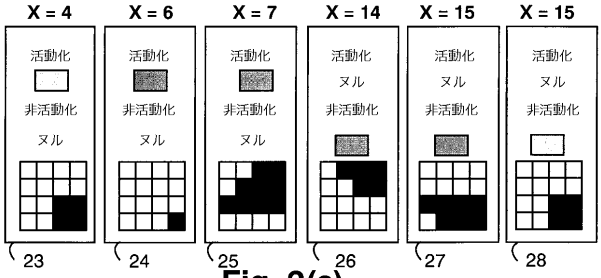


Fig. 2(c)

【図２（ｄ）】

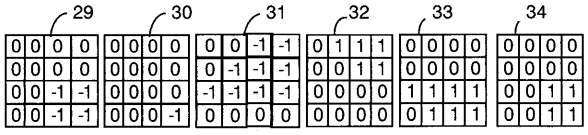


Fig. 2(d)

【図３】

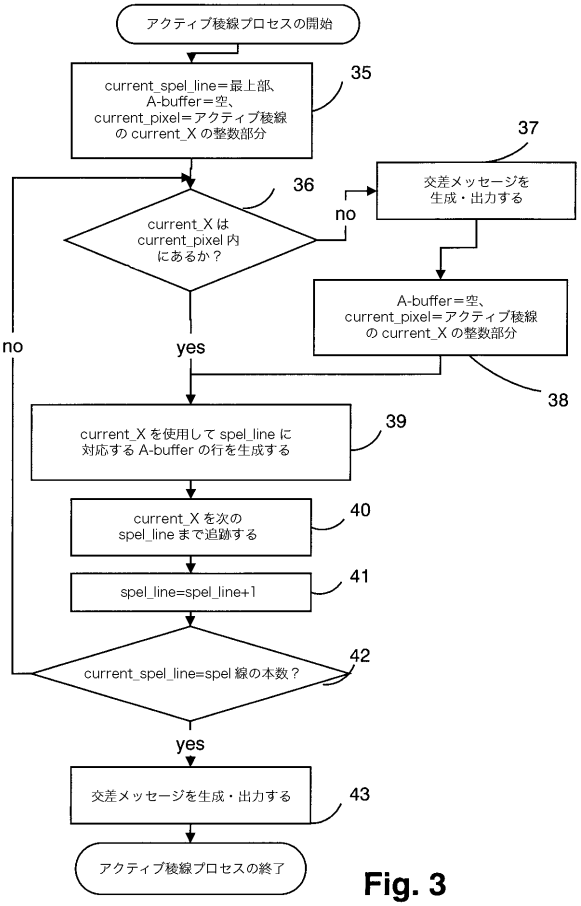


Fig. 3

【図４】

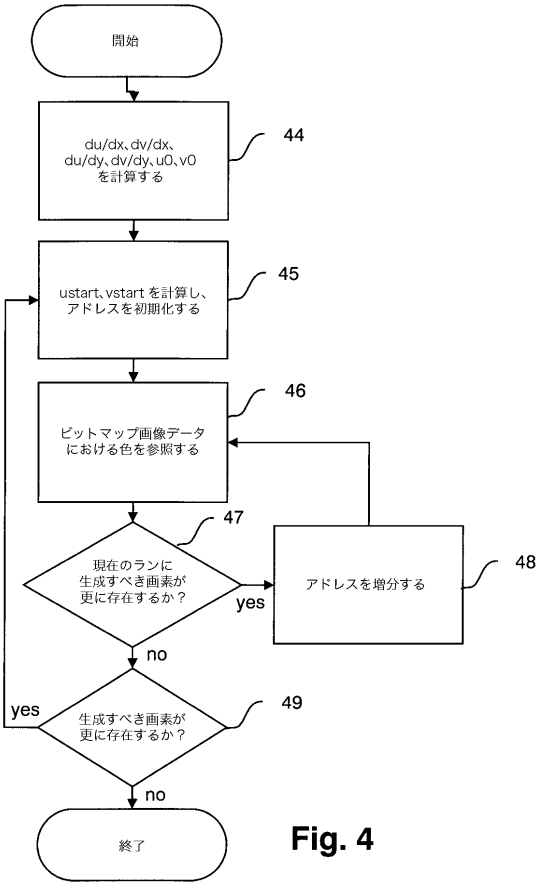


Fig. 4

【図５】

```
void
DrawLine
(
    int Xs,
    int Ys,
    int Xe,
    int Ye
)
{
    int    tx = Xe - Xs;
    int    ty = Ye - Ys;
    int    grad = tx / ty;
    int    ient = tx % ty;
    int    err = (2 * ient) - ty;
    int    delta_err1 = 2 * ient;
    int    delta_err2 = 2 * (ient - ty);
    int    x = Xs;
    int    y;

    for (y = Ys; y <= Ye; y++)
    {
        SetPixel(x, y);
        if (err < 0)
        {
            err += delta_err1;
            x += grad;
        }
        else
        {
            err += delta_err2;
            x += grad + 1;
        }
    }
}
```

Fig. 5

【図６】

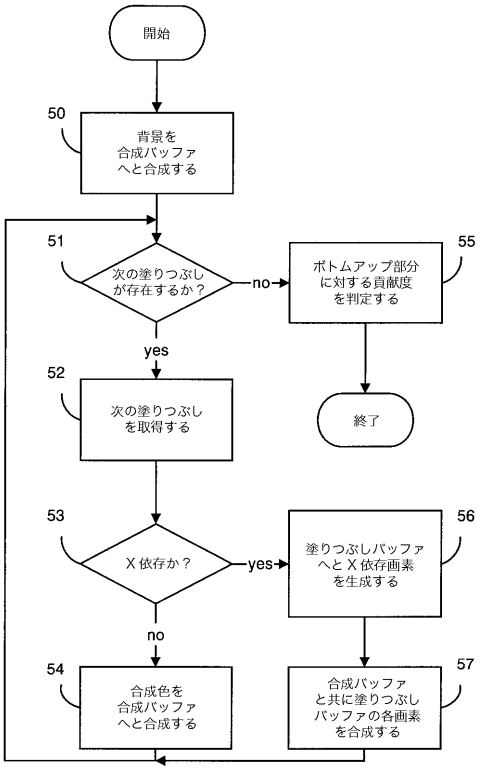


Fig. 6

【図７】

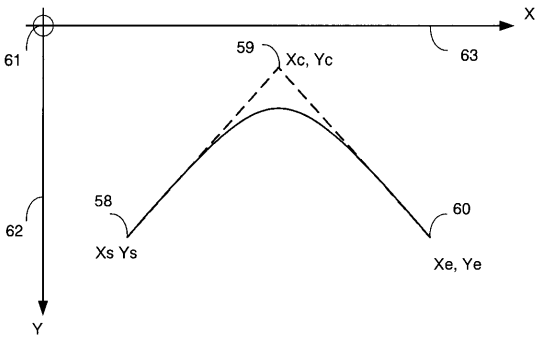


Fig. 7

【図８】

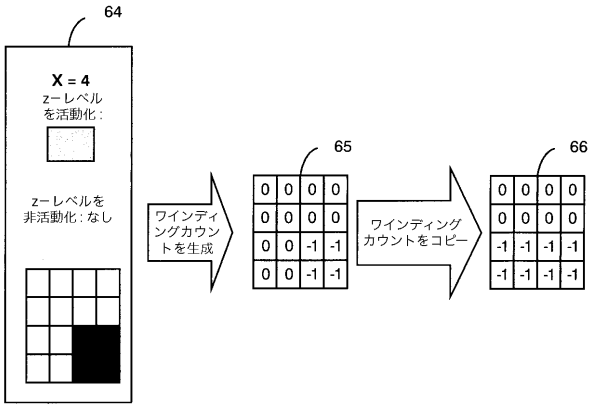


Fig. 8

【図９（ａ）】

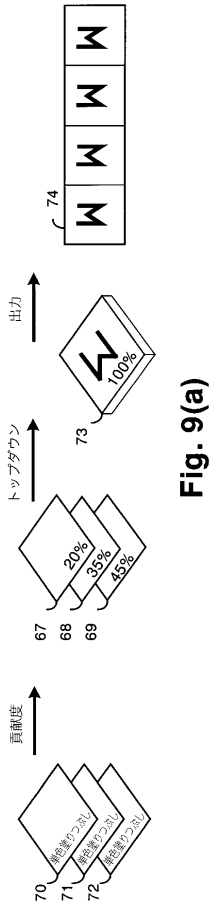


Fig. 9(a)

【図９（ｂ）】

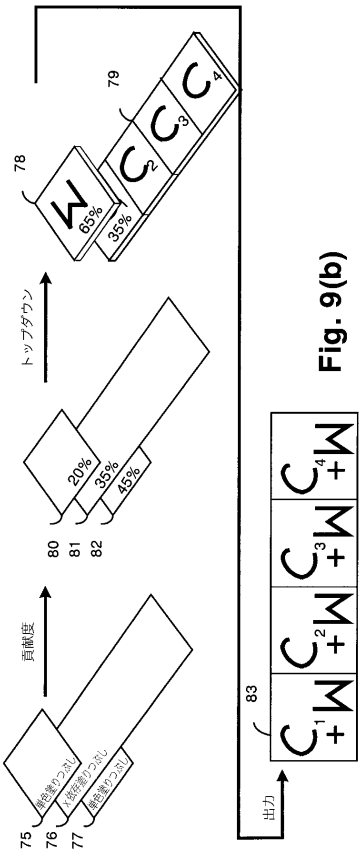


Fig. 9(b)

【図１０】

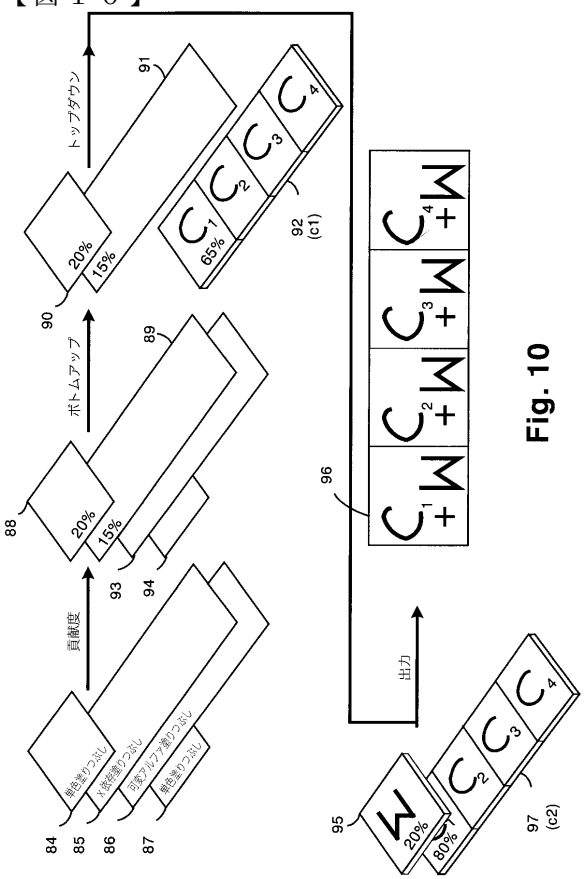


Fig. 10

【図１１】

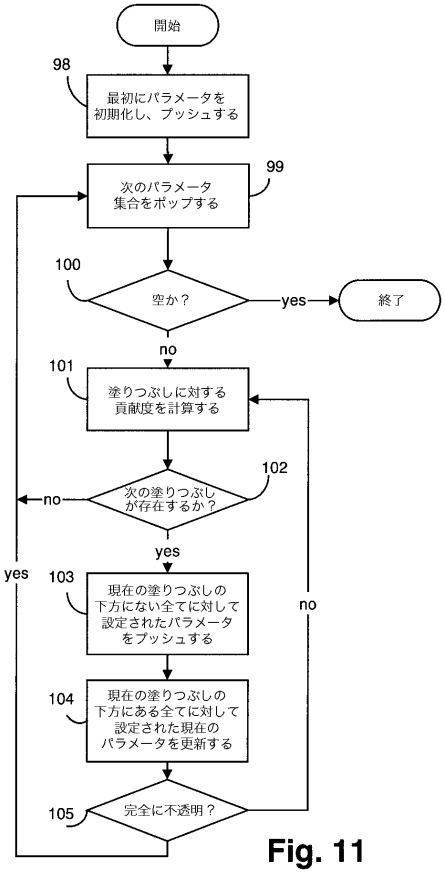
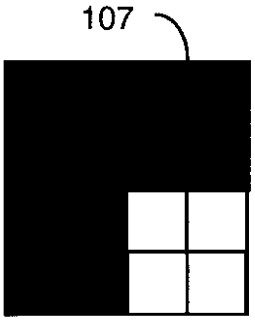


Fig. 11

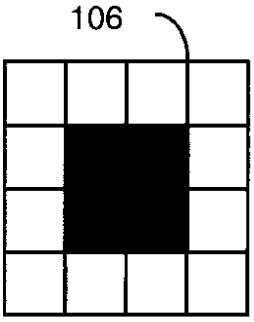
【図１２（ａ）】



カバレッジ

Fig. 12(a)

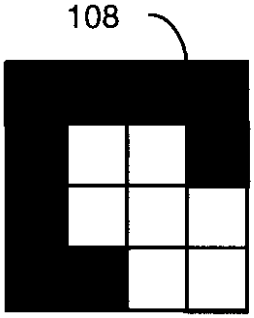
【図１２（ｂ）】



A-buffer

Fig. 12(b)

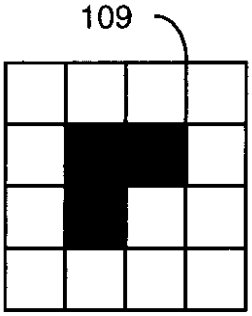
【図１２（ｃ）】



カバレッジ∩～（カバレッジ∩A-buffer)

Fig. 12(c)

【図１２（ｄ）】



カバレッジ∩A-buffer

Fig. 12(d)

【図１３】

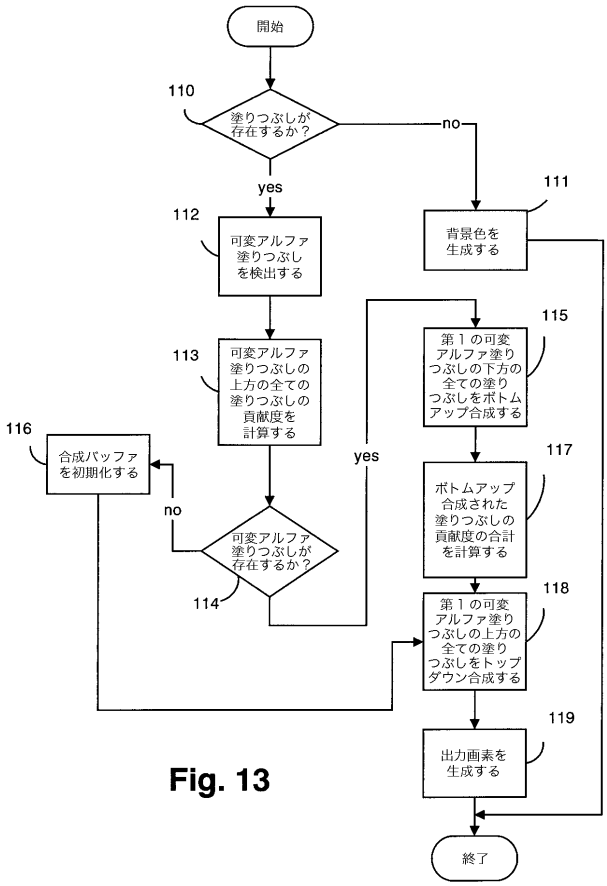


Fig. 13

【図１４】

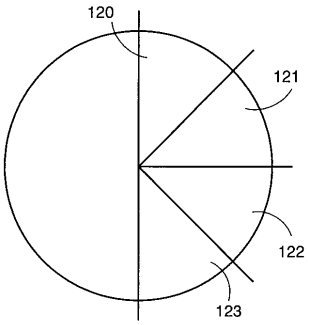


Fig. 14

【図１５】

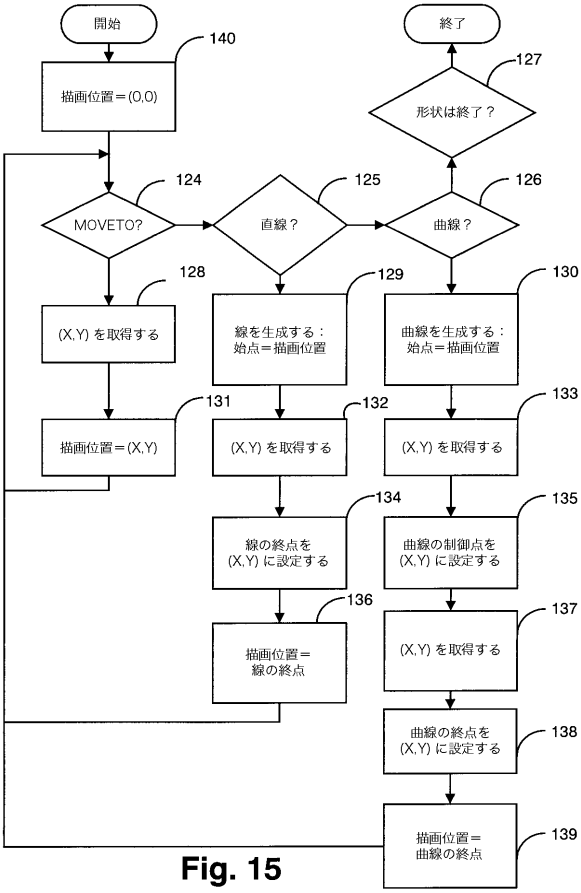


Fig. 15

【図１６（ａ）】

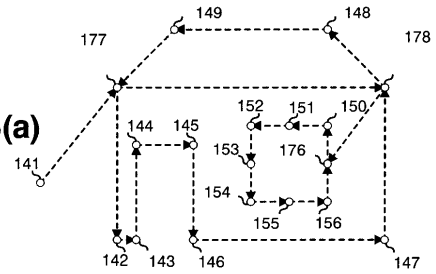


Fig. 16(a)

【図１６（ｂ）】

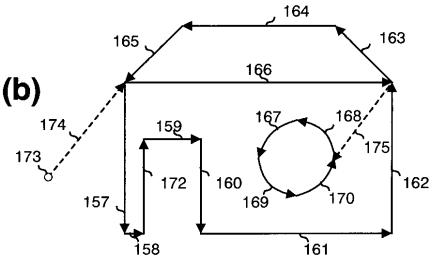


Fig. 16(b)

【図１６（ｃ）】

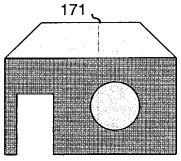


Fig. 16(c)

【図１７（ａ）】

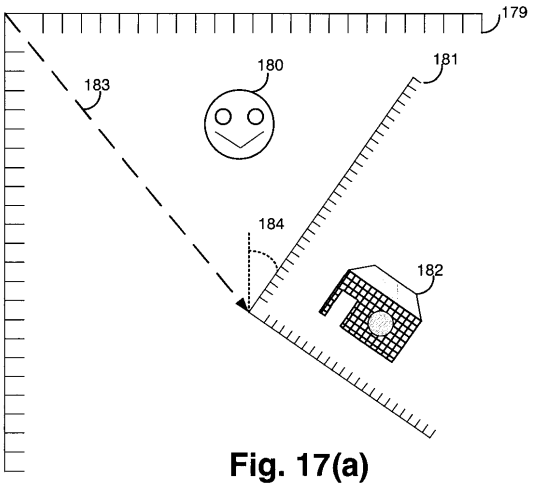


Fig. 17(a)

【図１７（ｂ）】

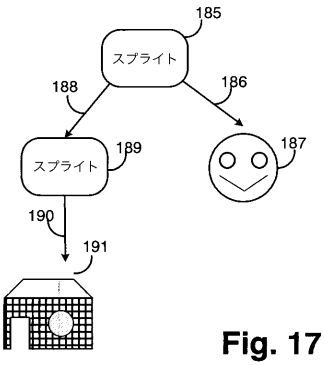


Fig. 17(b)

【図１８（ａ）】

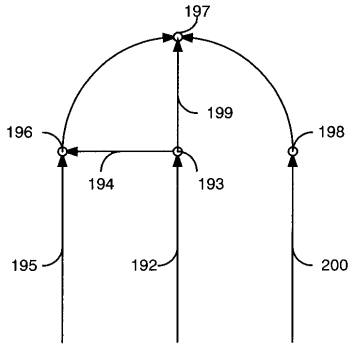


Fig. 18(a)

【図１８（ｂ）】

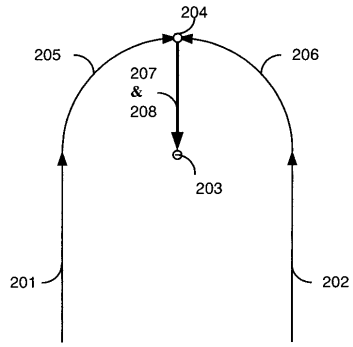


Fig. 18(b)

【図１９】

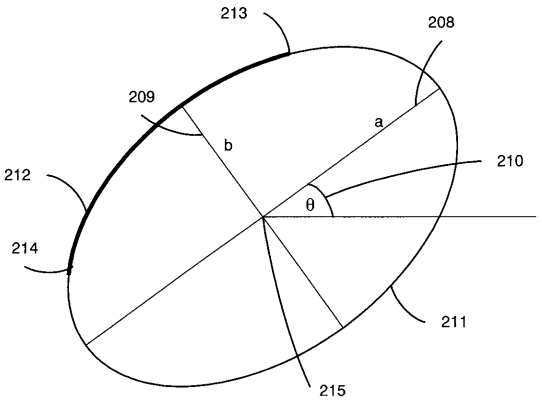


Fig. 19

【図２０】

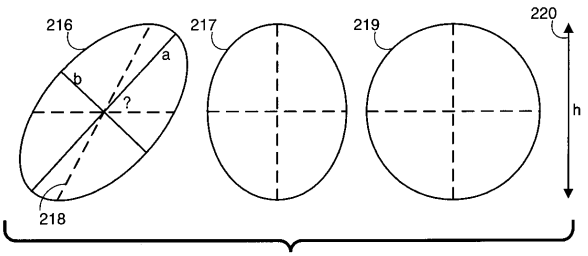


Fig. 20

【図２１】

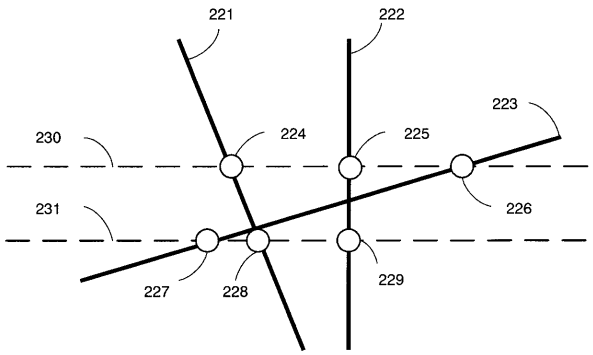


Fig. 21

【図２２（ｂ）】

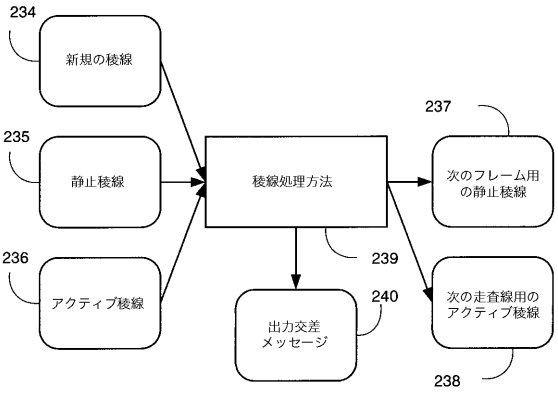


Fig. 22(b)

【図２２（ａ）】

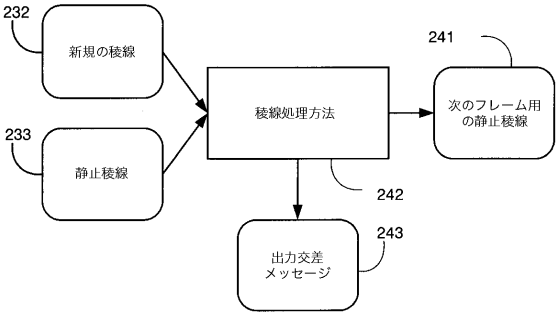


Fig. 22(a)

【図２３】

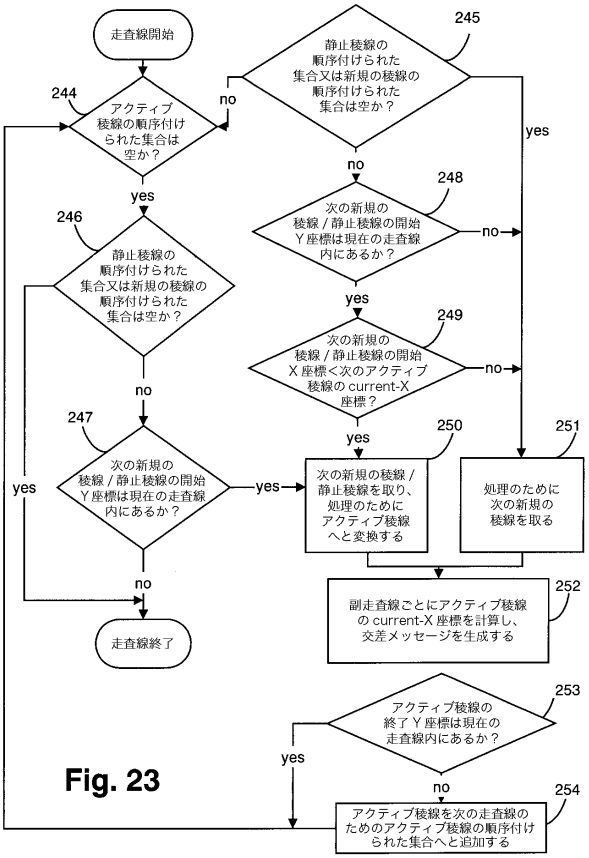


Fig. 23

【図２４】

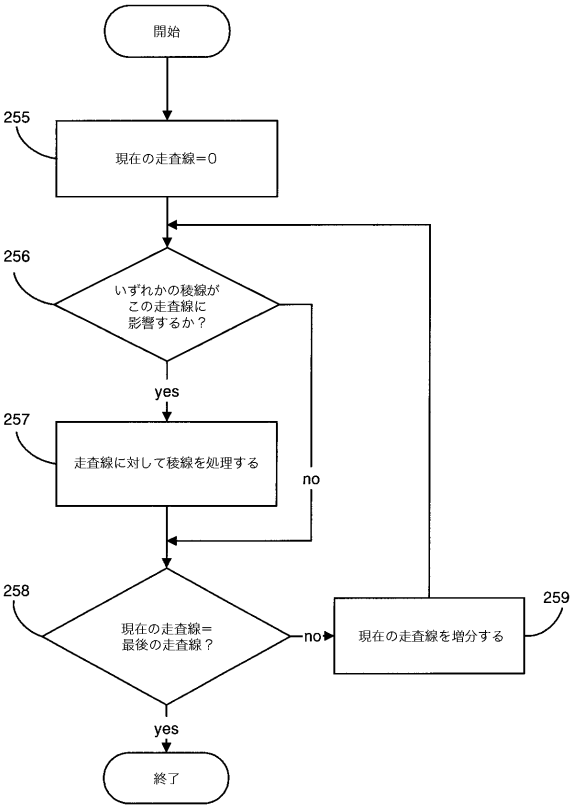


Fig. 24

【図２５】

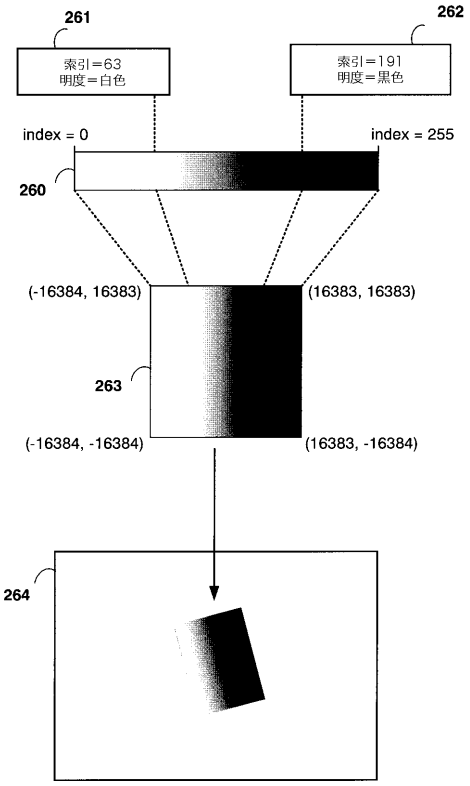


Fig. 25

【図２６】

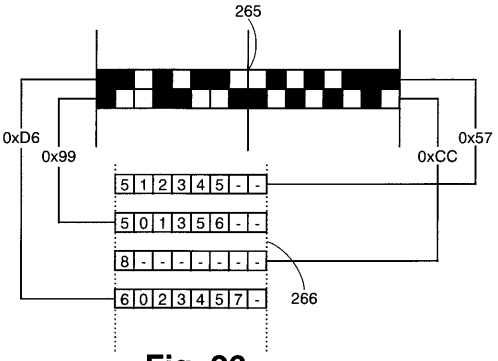


Fig. 26

【図２７】

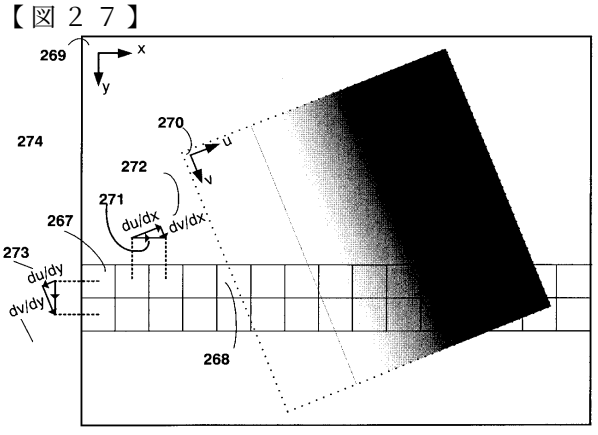


Fig. 27

【図２８（ａ）】

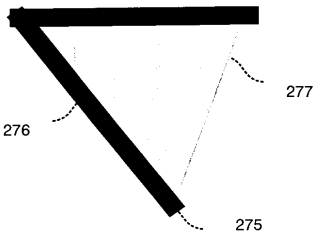


Fig. 28(a)

【図２８（ｂ）】

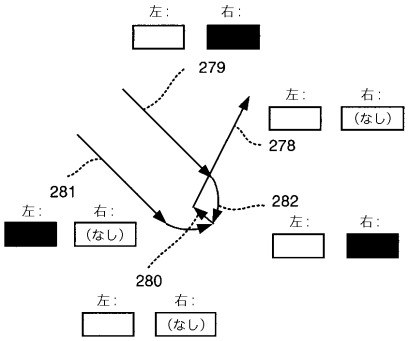


Fig. 28(b)

【図２９（ａ）】

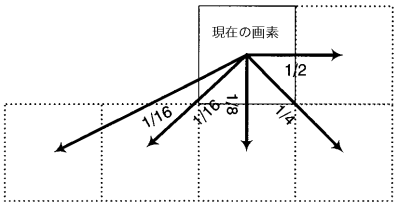


Fig. 29(a)

【図２９（ｂ）】

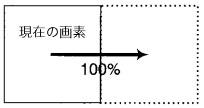


Fig. 29(b)

【図２９（ｃ）】

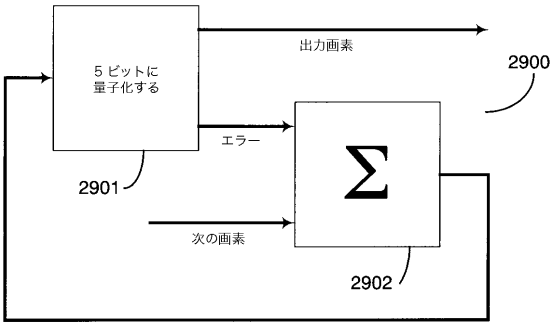


Fig. 29(c)

【図３０】

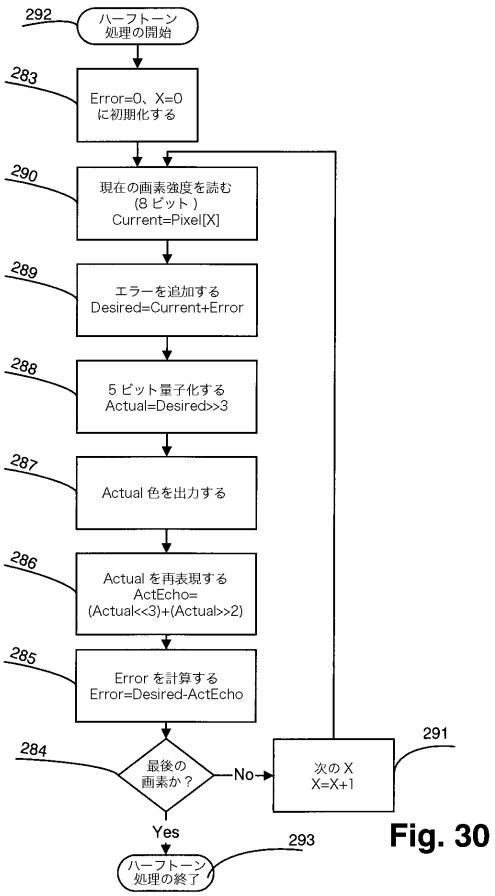


Fig. 30

【図３１（ａ）】

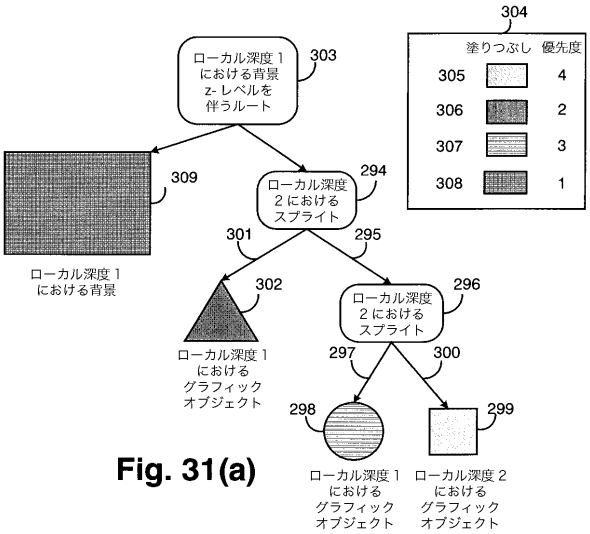


Fig. 31(a)

【図３１（ｂ）】

Fig. 31(b)

【図３１（ｃ）】

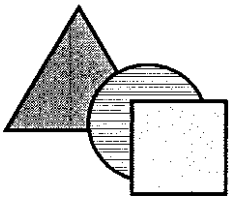


Fig. 31(c)

【図３２】

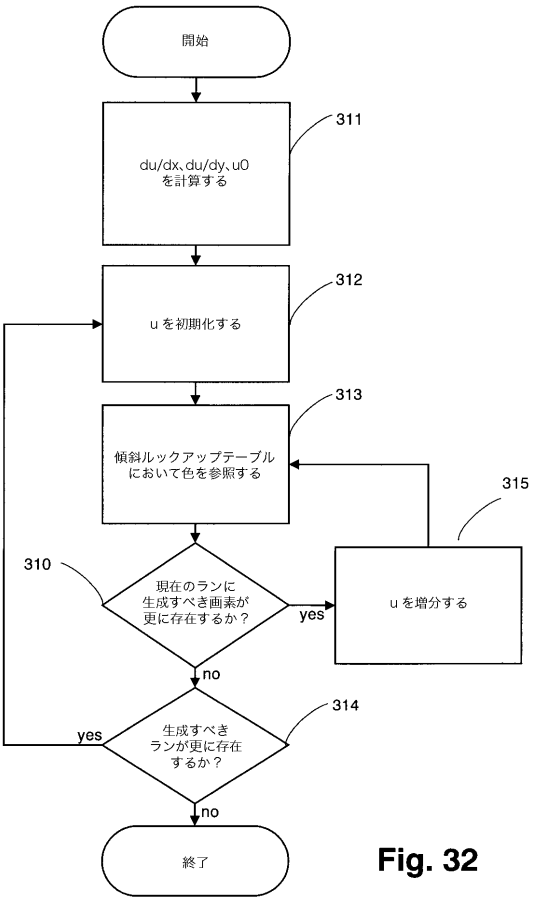


Fig. 32

【図３３（ａ）】

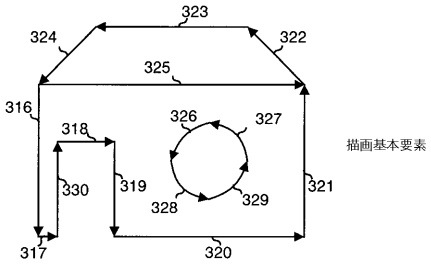
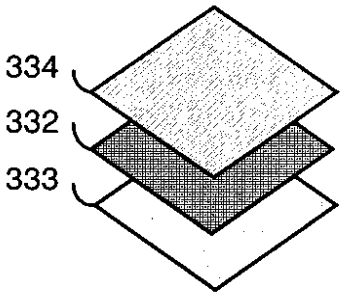


Fig. 33(a)

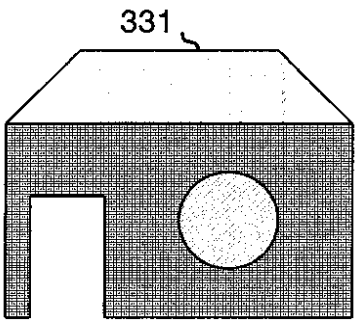
【図３３（ｂ）】



優先度順のz-レベル

Fig. 33(b)

【図３３（ｃ）】



レンダリング結果

Fig. 33(c)

【図３４】

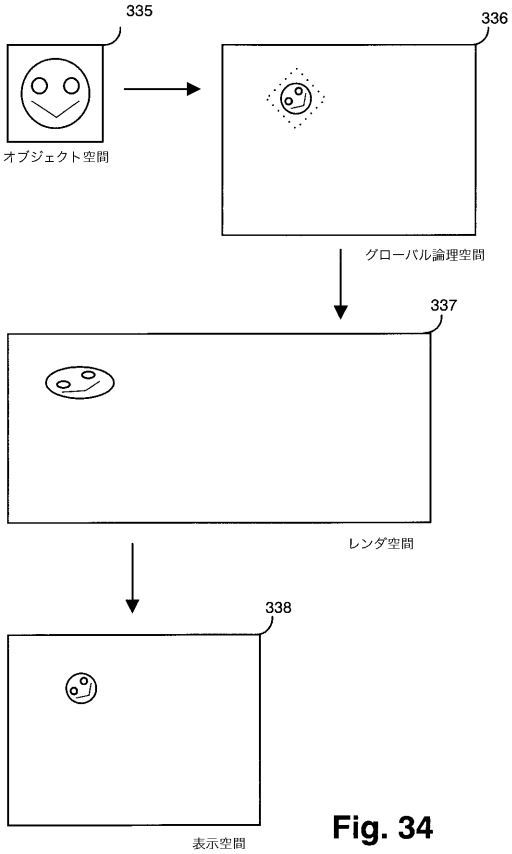


Fig. 34

【図３５（ａ）】

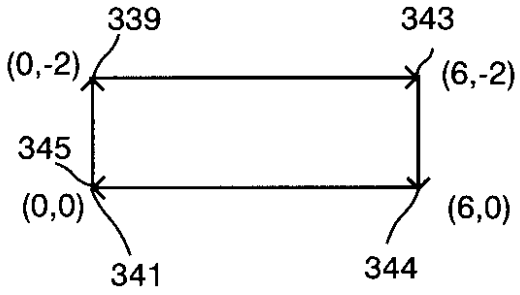


Fig. 35(a)

【図３５（ｂ）】

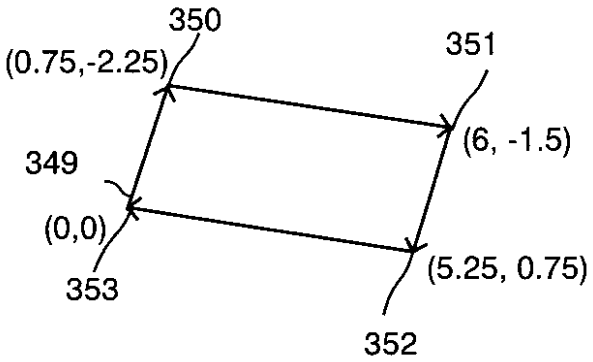
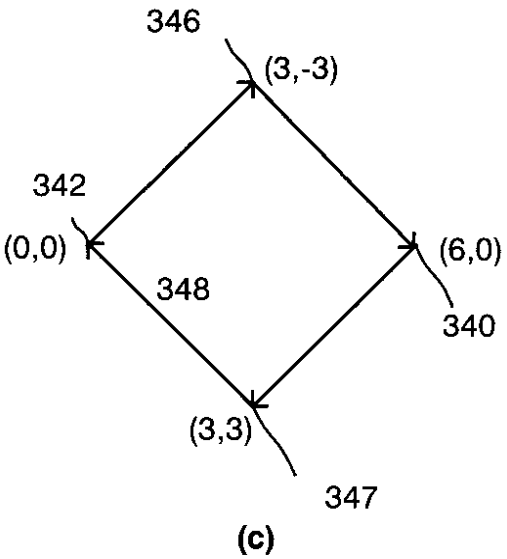


Fig. 35(b)

【図３５（ｃ）】



(c)

Fig. 35(c)

【図３６（ａ）】

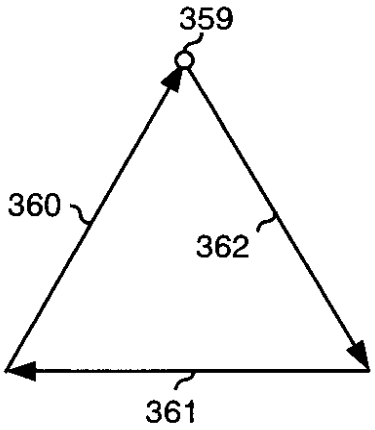


Fig. 36(a)

【図３６（ｂ）】

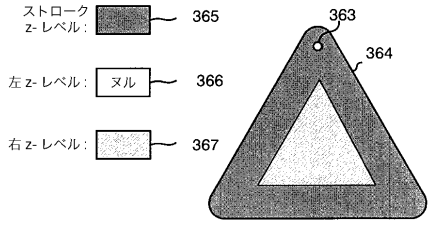


Fig. 36(b)

【図３６（ｃ）】

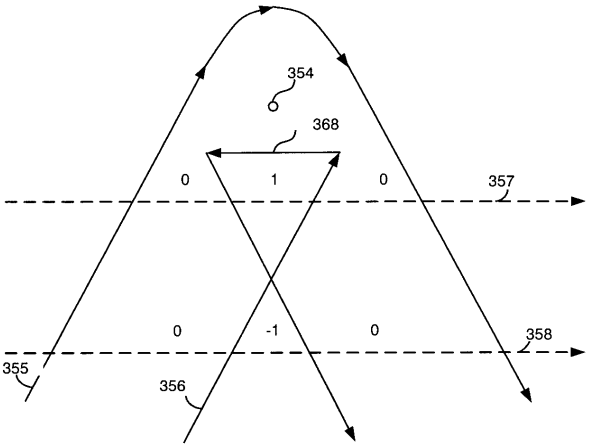


Fig. 36(c)

【図３７】

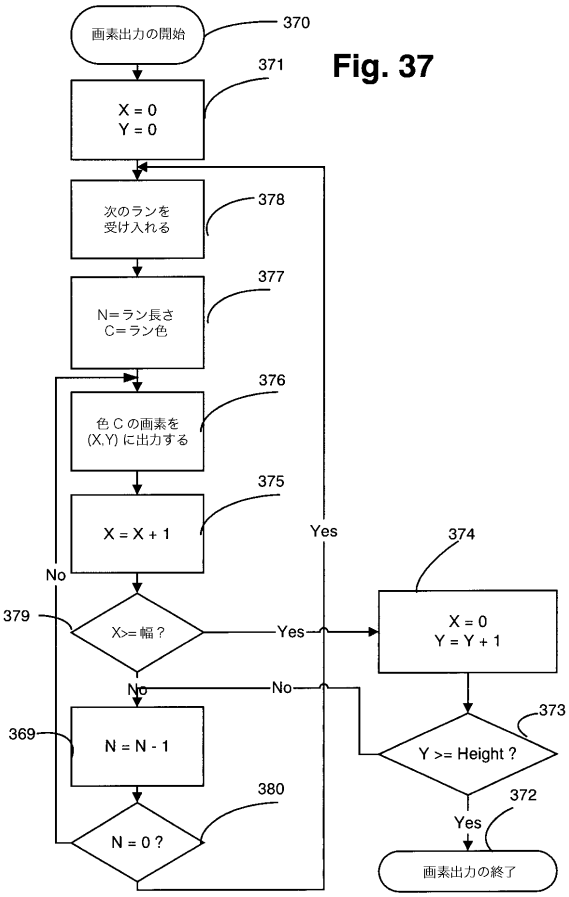


Fig. 37

【図３８（ａ）】

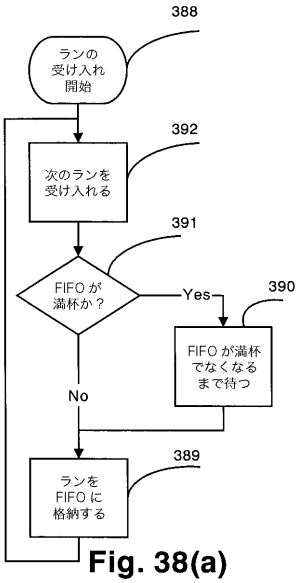


Fig. 38(a)

【図３８（ｂ）】

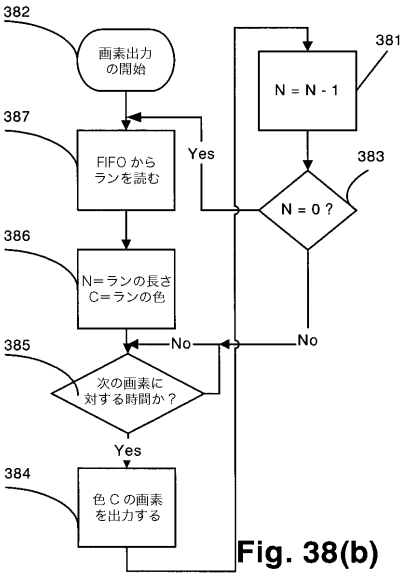


Fig. 38(b)

【図３９】

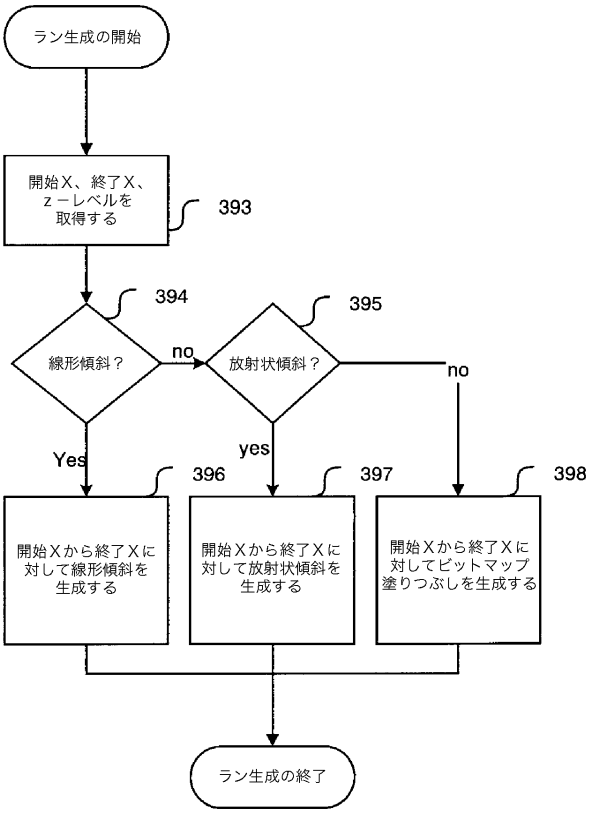


Fig. 39

【図４０】

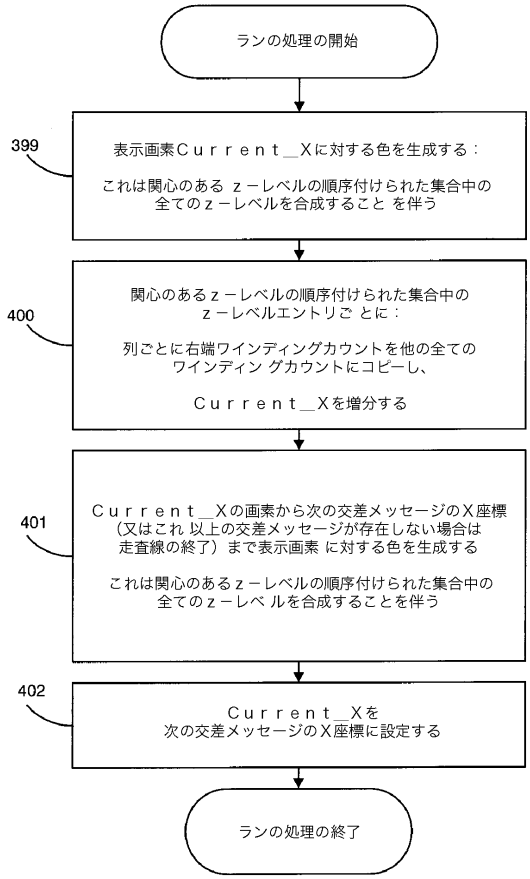


Fig. 40

【図４１】

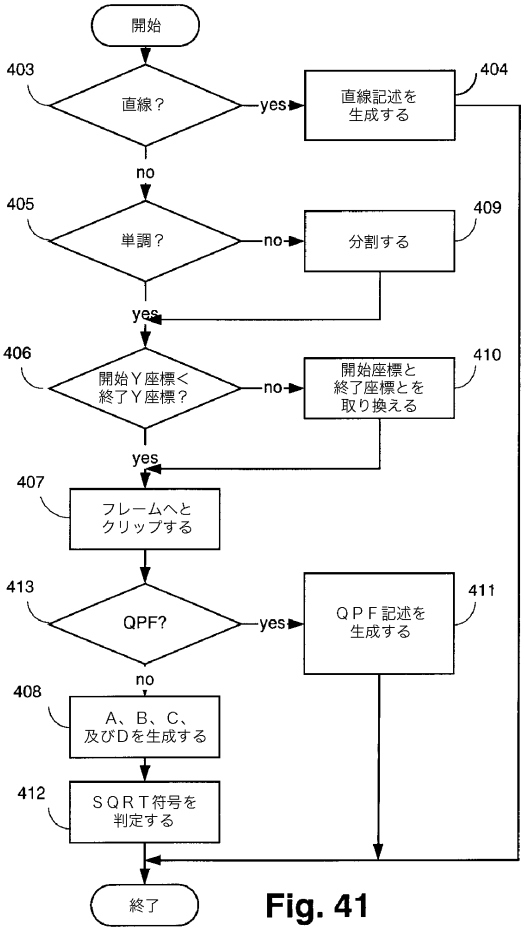


Fig. 41

【図４２】

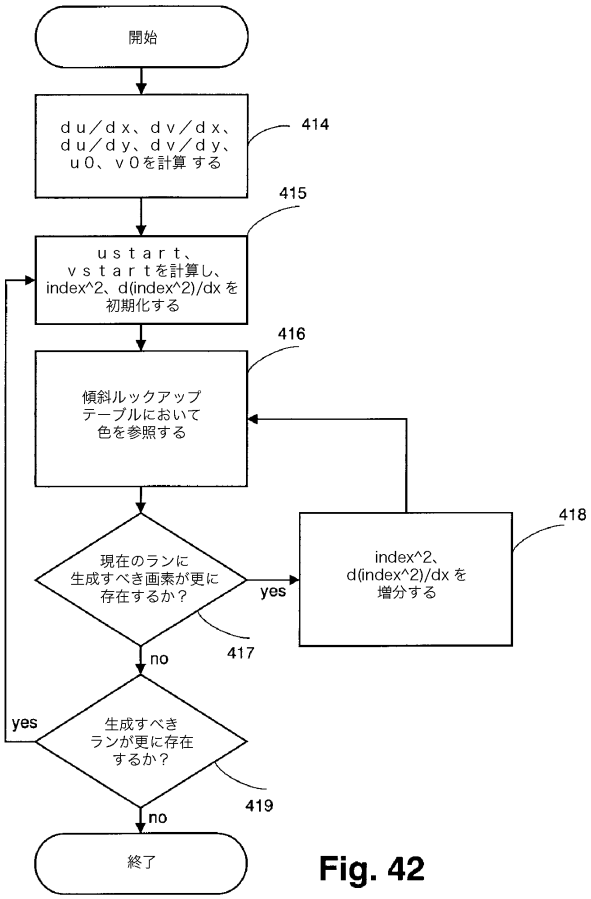


Fig. 42

【図４３（ａ）】

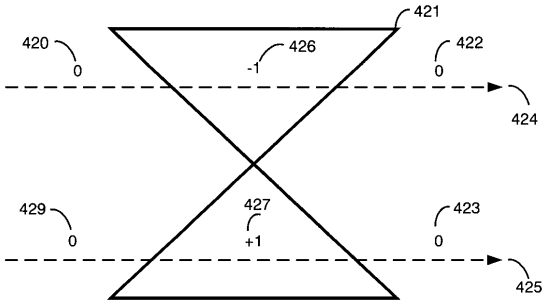


Fig. 43(a)

【図４３（ｂ）】

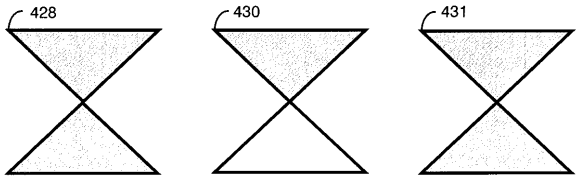


Fig. 43(b)

【図４４】

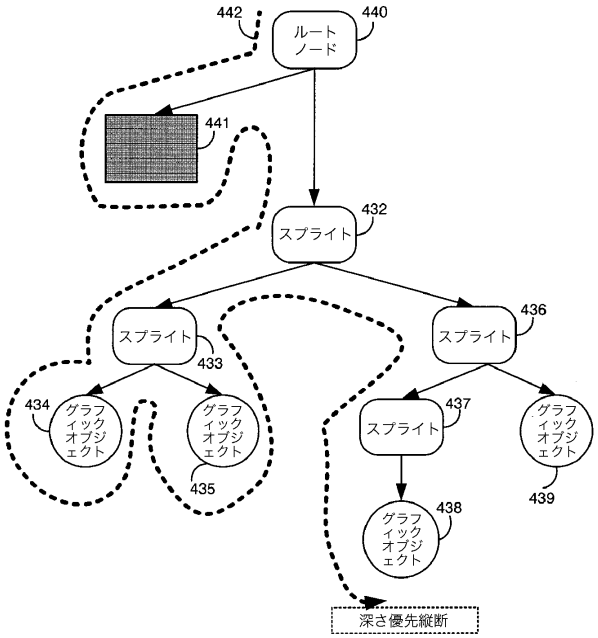


Fig. 44

【図４５（ａ）】

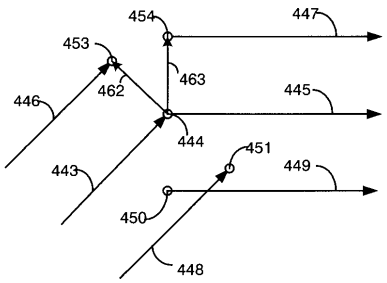


Fig. 45(a)

【図４５（ｂ）】

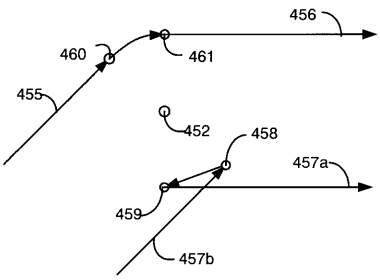


Fig. 45(b)

【図４６】

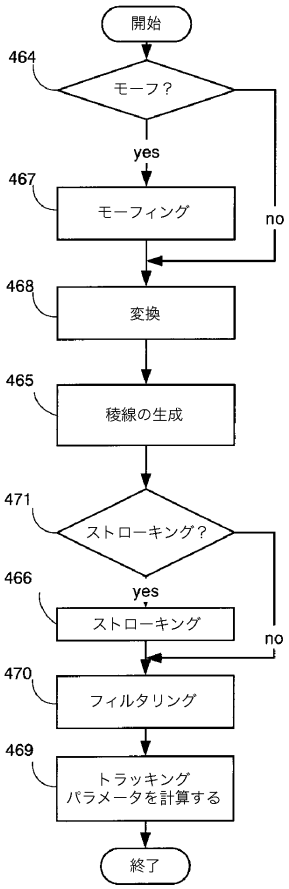


Fig. 46

【図４７（ａ）】

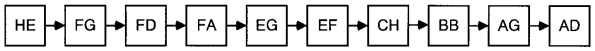


Fig. 47(a)

【図４７（ｂ）】

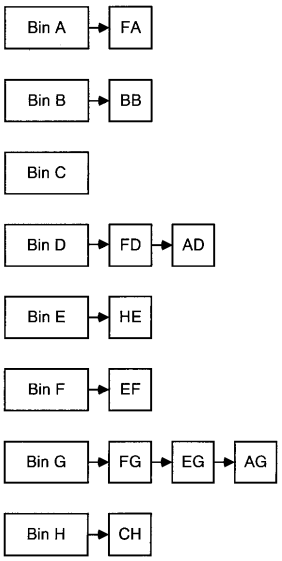


Fig. 47(b)

【図４７（ｃ）】

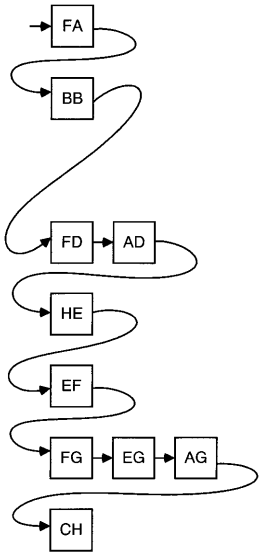


Fig. 47(c)

【図４７（ｄ）】

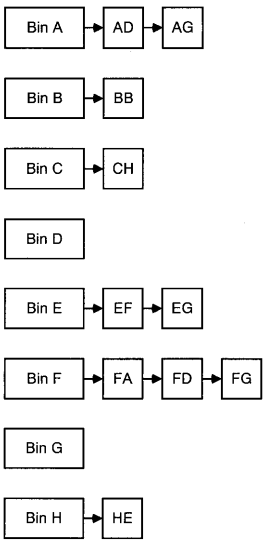


Fig. 47(d)

【図４７（ｅ）】

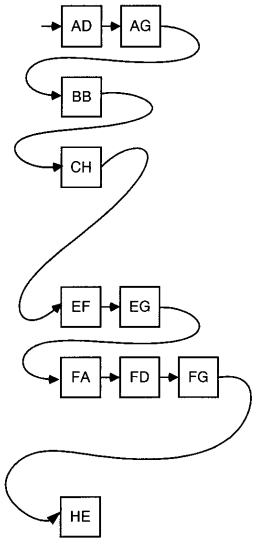


Fig. 47(e)

【図４７（ｆ）】

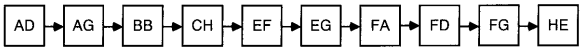


Fig. 47(f)

【図４８】

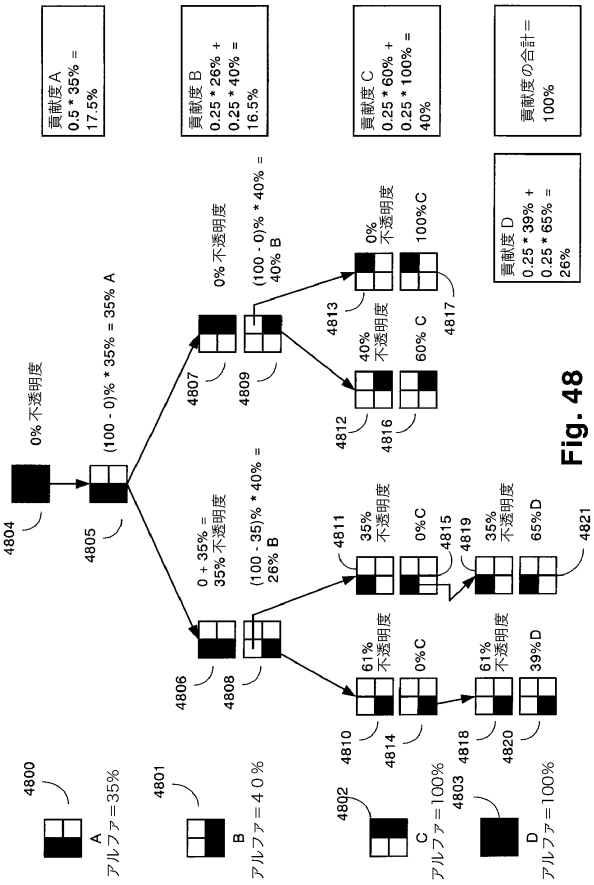


Fig. 48

【図４９（ａ）】

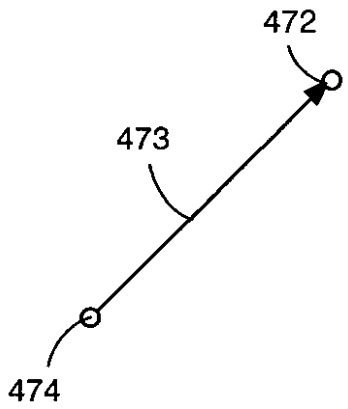


Fig. 49(a)

【図４９（ｂ）】

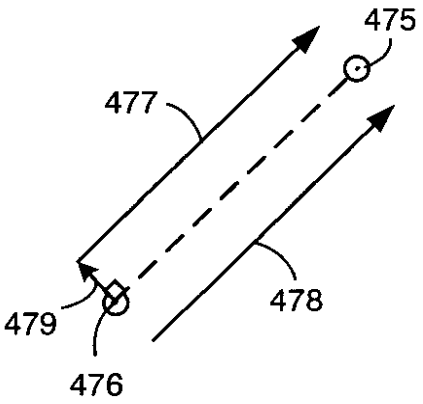


Fig. 49(b)

【図４９（ｃ）】

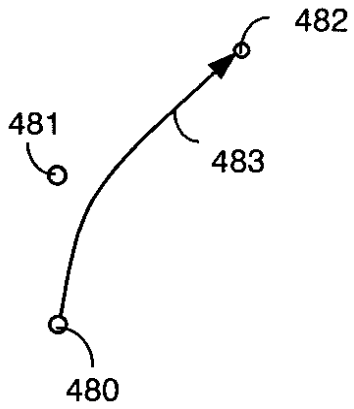


Fig. 49(c)

【図４９（ｄ）】

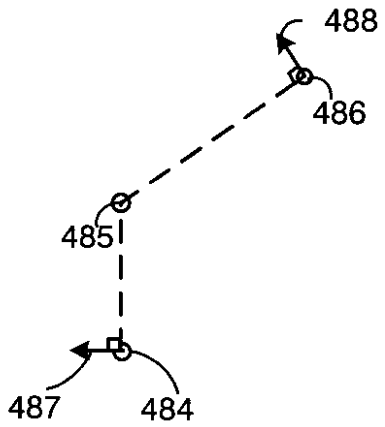


Fig. 49(d)

【図４９（ｅ）】

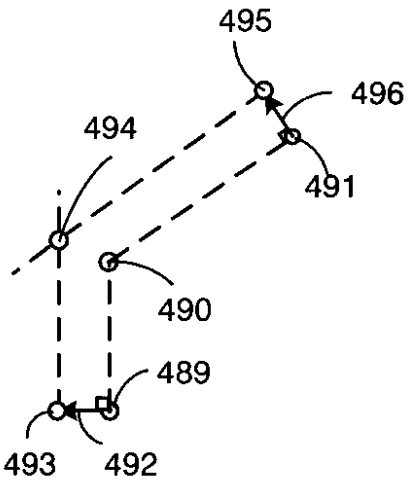


Fig. 49(e)

【図４９（ｆ）】

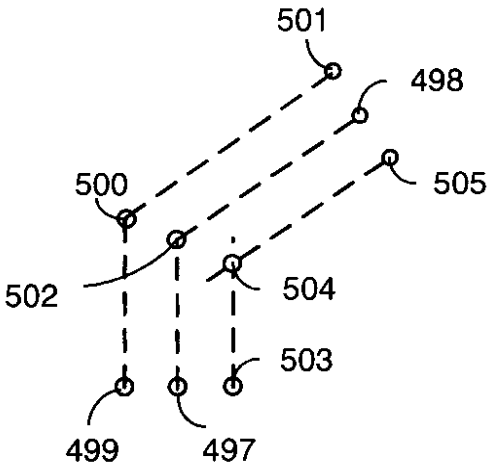


Fig. 49(f)

【図４９（ｇ）】

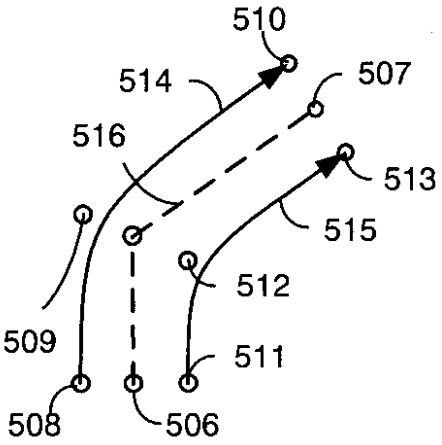


Fig. 49(g)

【図４９（ｈ）】

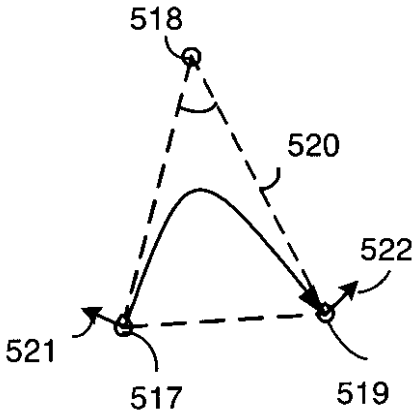


Fig. 49(h)

【図５０（ａ）】

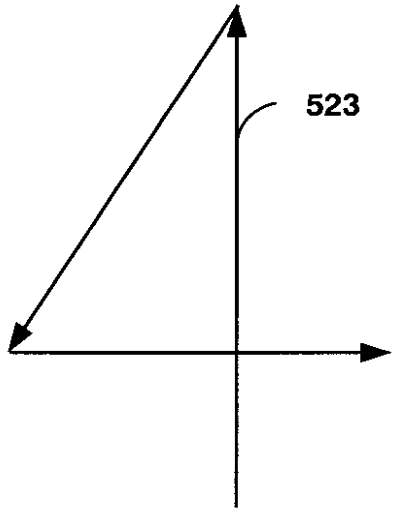


Fig. 50(a)

【図５０（ｂ）】

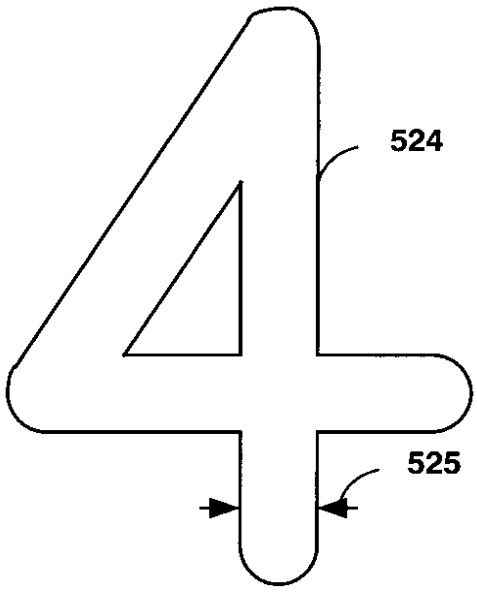


Fig. 50(b)

【図５０（ｃ）】

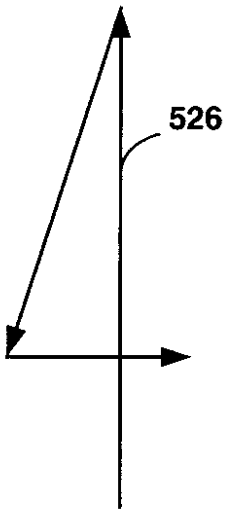


Fig. 50(c)

【図５０（ｄ）】

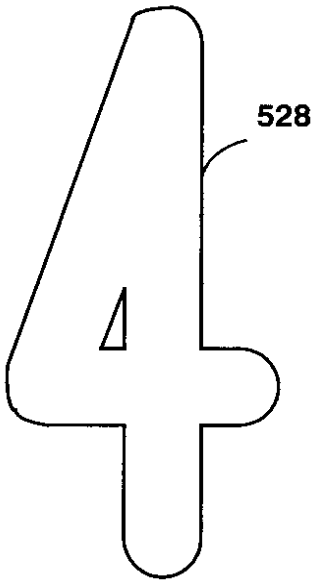


Fig. 50(d)

【図５０（ｅ）】

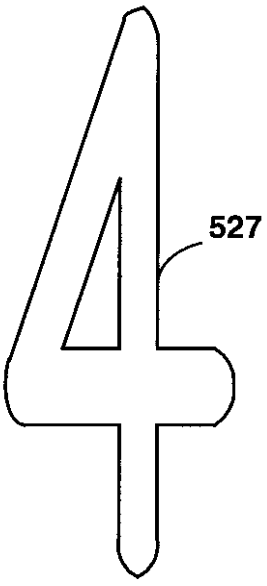


Fig. 50(e)

【図５１（ａ）】

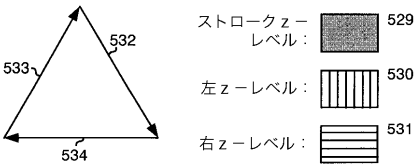


Fig 51(a)

【図５１（ｂ）】

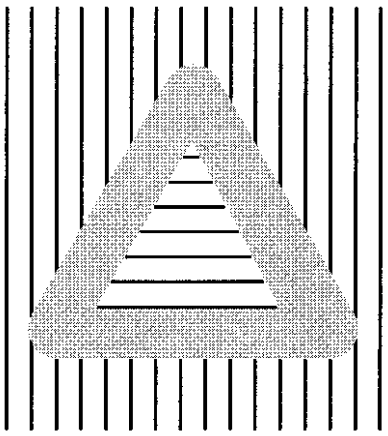


Fig 51(b)

【図５１（ｃ）】

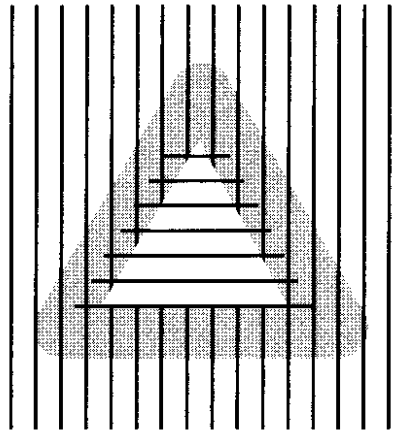


Fig 51(c)

【図５１（ｄ）】

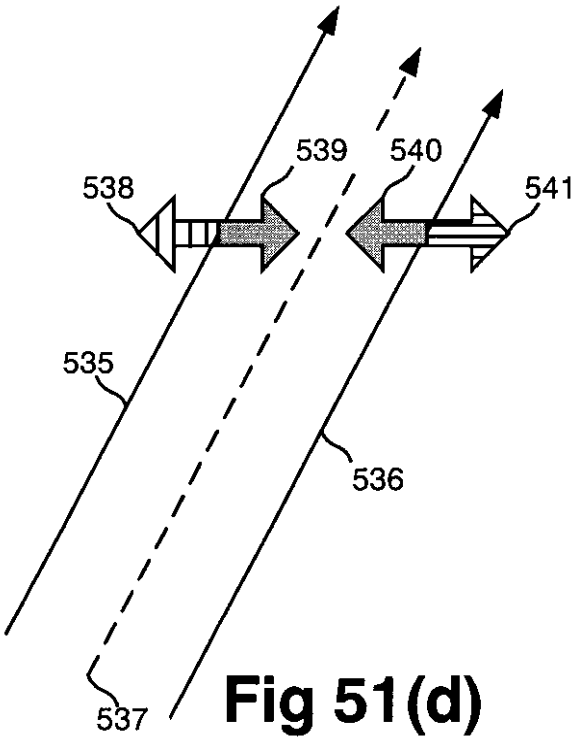


Fig 51(d)

【図５１（ｅ）】

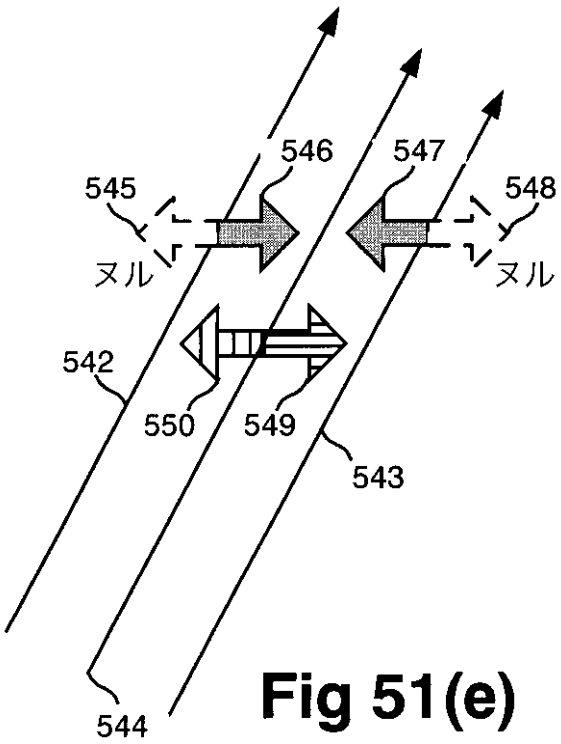


Fig 51(e)

【図５２（ａ）】

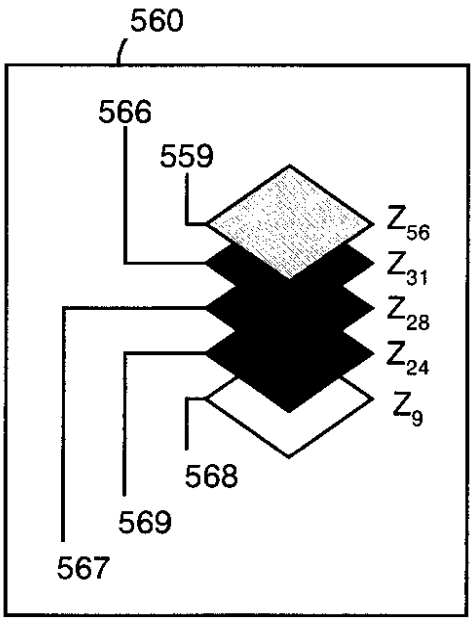


Fig. 52(a)

【図５２（ｂ）】

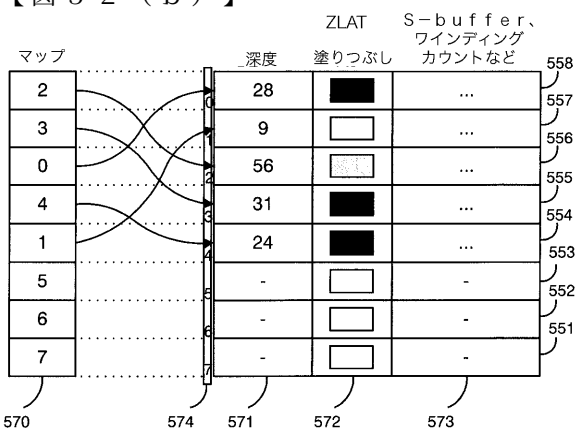


Fig. 52(b)

【図５３（ａ）】

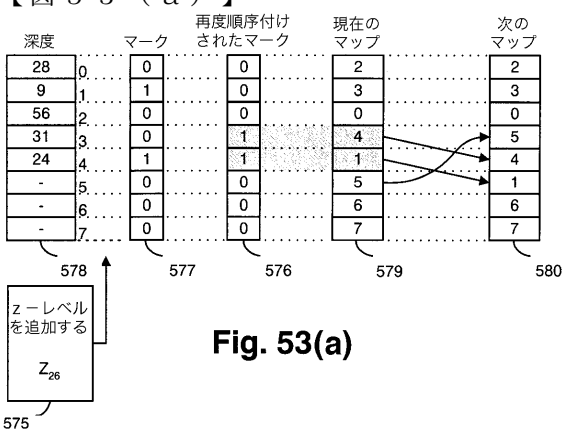


Fig. 53(a)

【図５３（ｂ）】

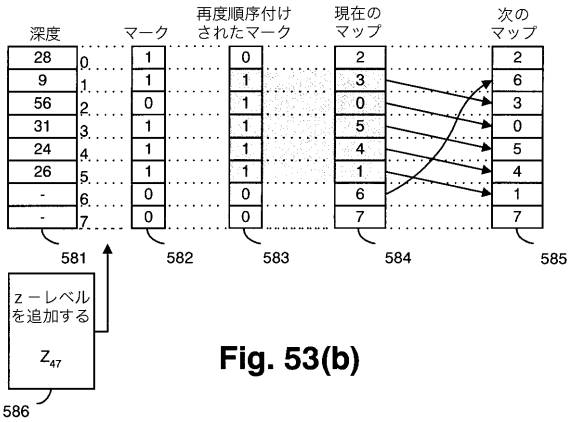


Fig. 53(b)

【図５４（ａ）】

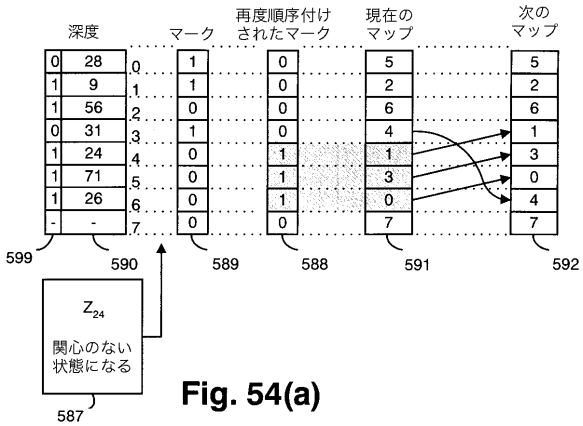


Fig. 54(a)

【図５４（ｂ）】

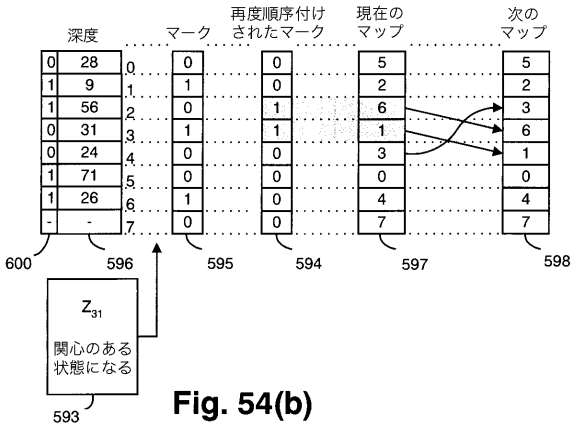


Fig. 54(b)

【図５５】

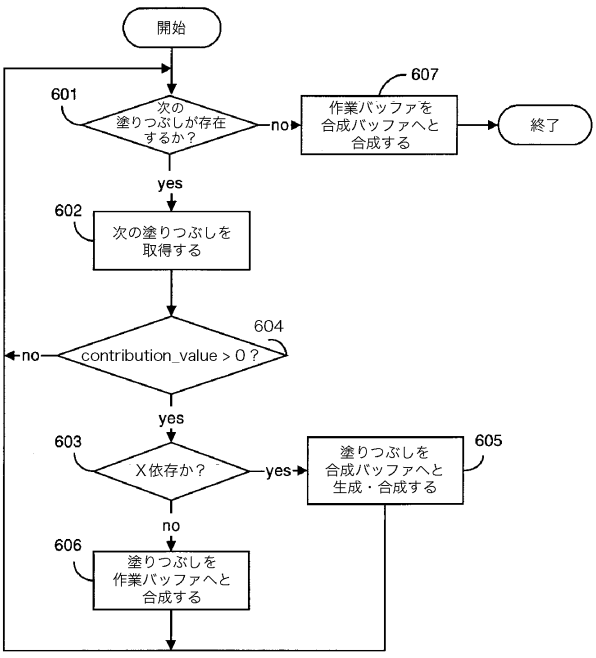


Fig. 55

【図５６】

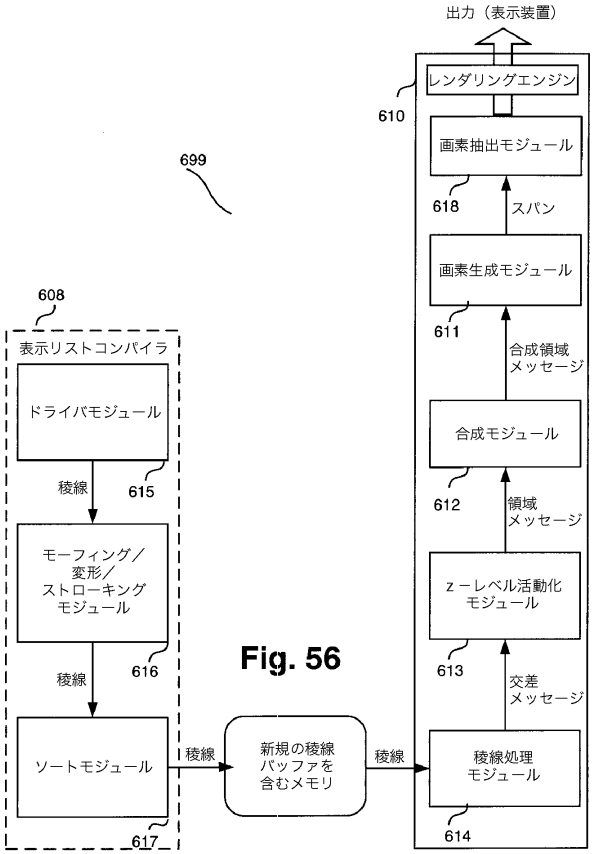


Fig. 56

【図５７（ａ）】

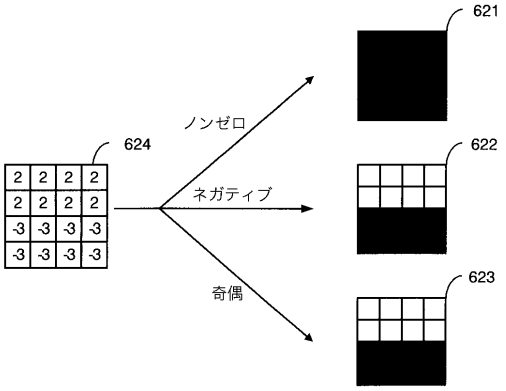


Fig. 57(a)

【図５７（ｂ）】

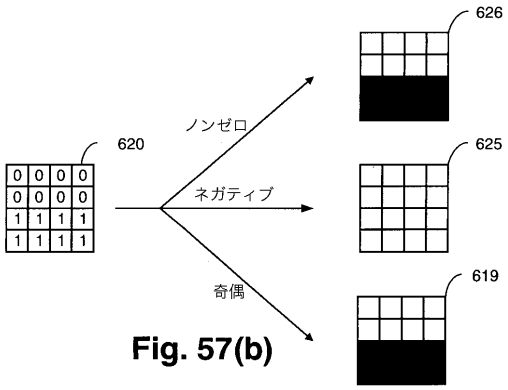


Fig. 57(b)

【図５８】

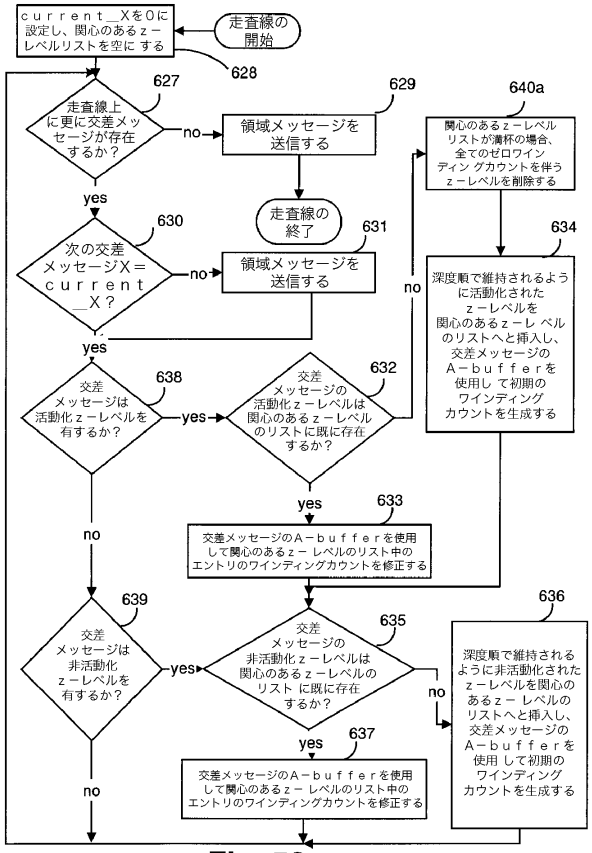


Fig. 58

【図５９】

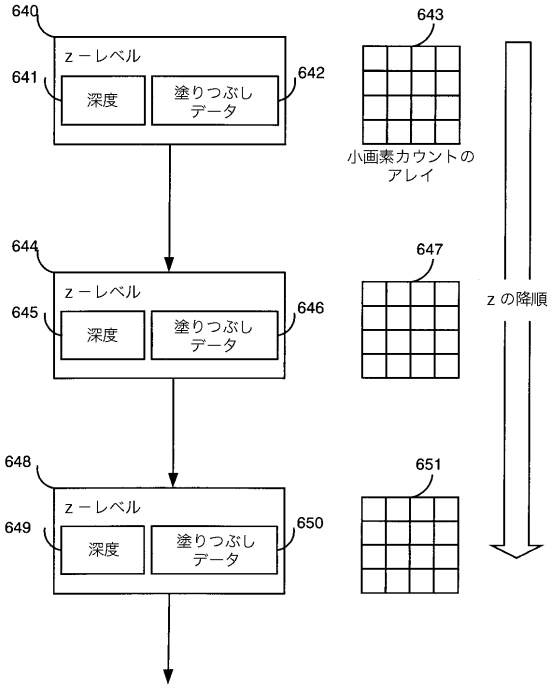


Fig. 59

【図６０】

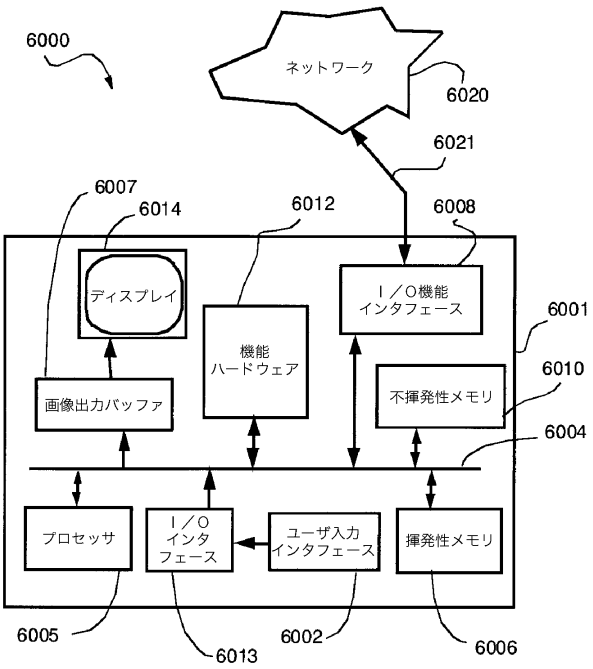


Fig. 60

【図６１（ａ）】

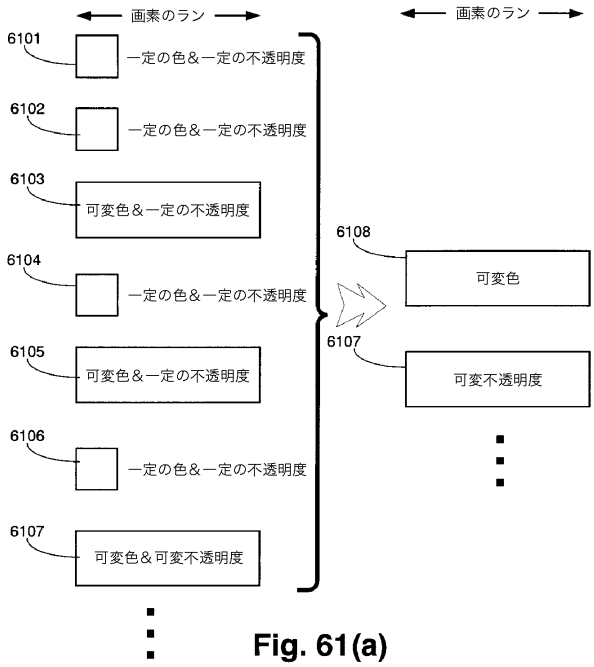


Fig. 61(a)

【図６１（ｂ）】

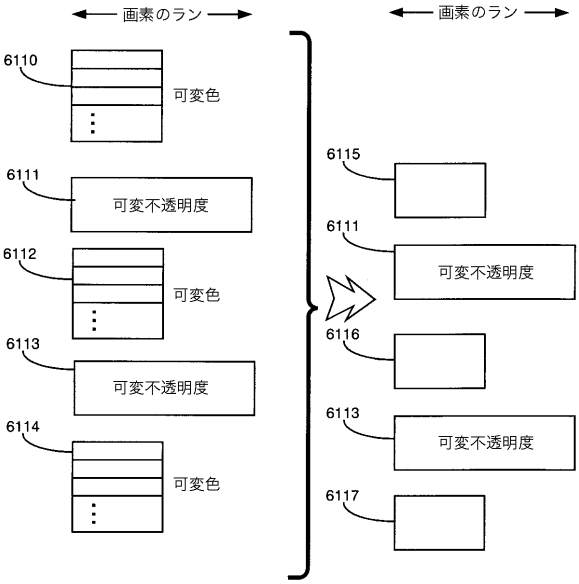


Fig. 61(b)

【図６２】

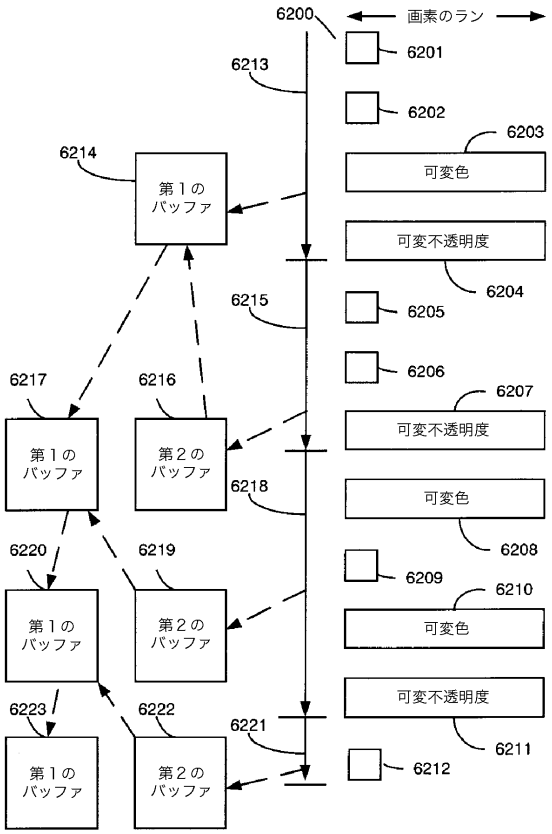


Fig. 62

【図６３】

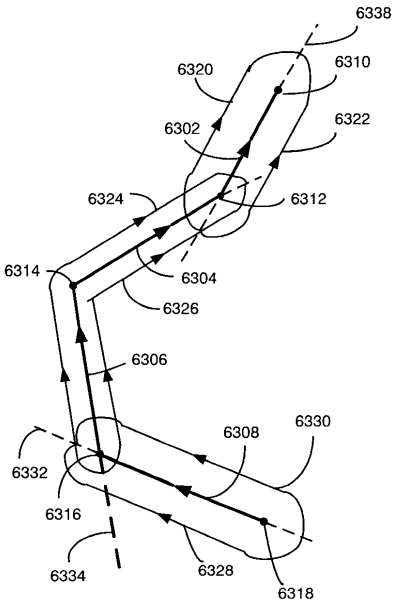


Fig. 63

【図６４】

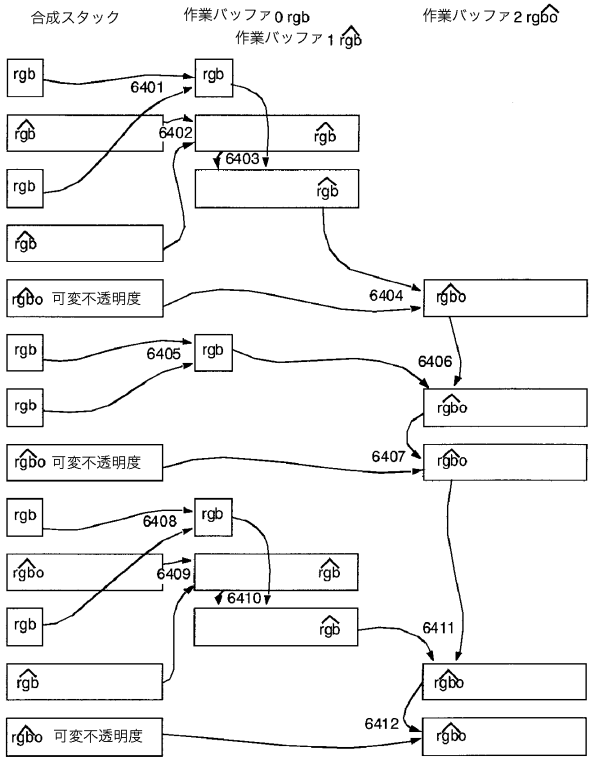


Fig. 64

【図６５Ａ】

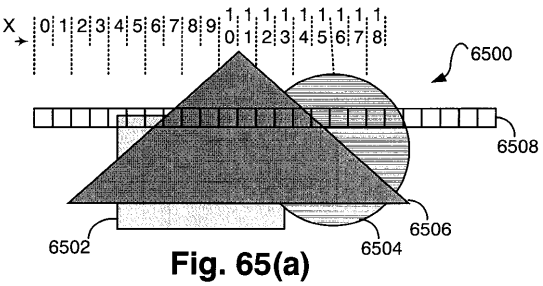


Fig. 65(a)

【図６５Ｂ】

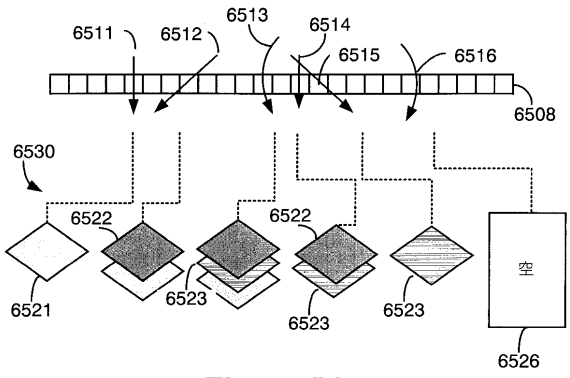


Fig. 65(b)

【図６６】

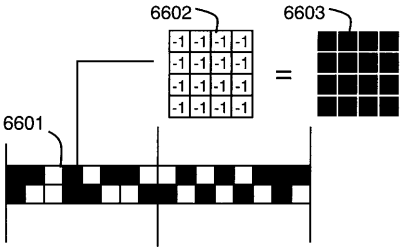


Fig. 66

【図６７】

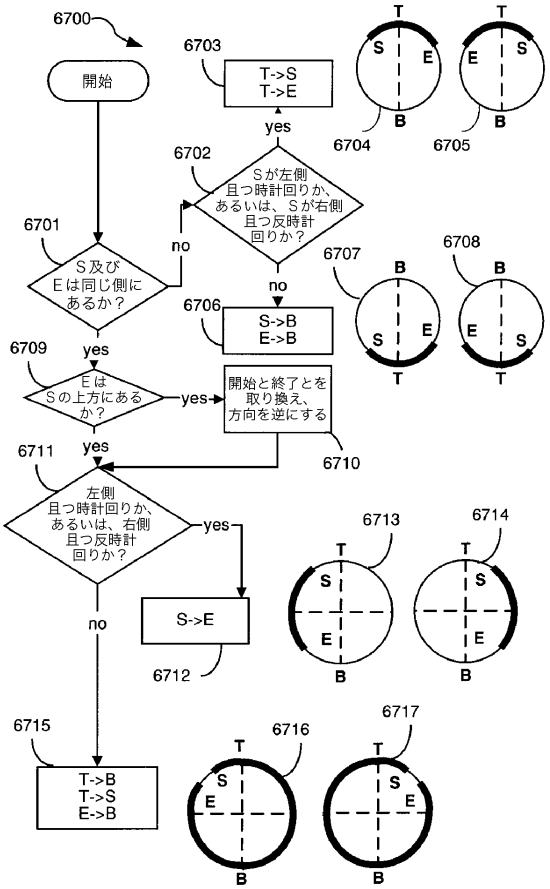


Fig. 67

【図６８Ａ】

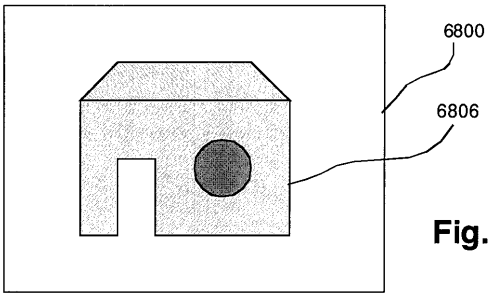


Fig. 68A

【図６８Ｂ】

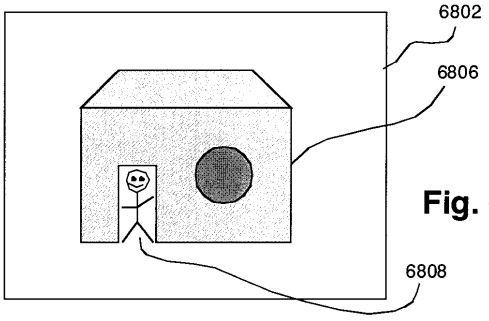


Fig. 68B

【図６８Ｃ】

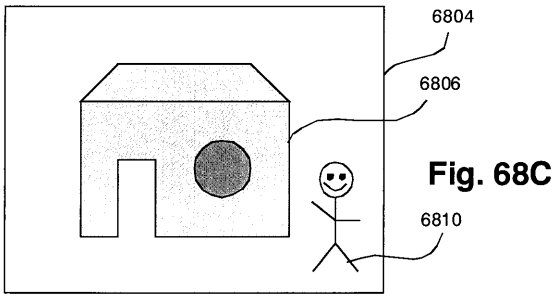


Fig. 68C

【図６９Ａ】

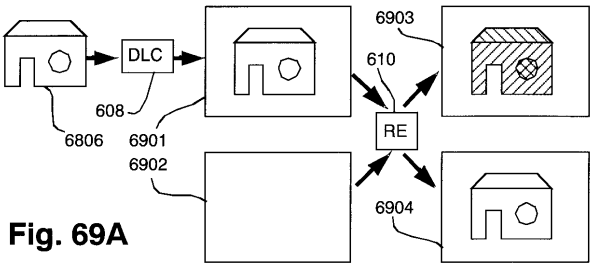


Fig. 69A

【図６９Ｂ】

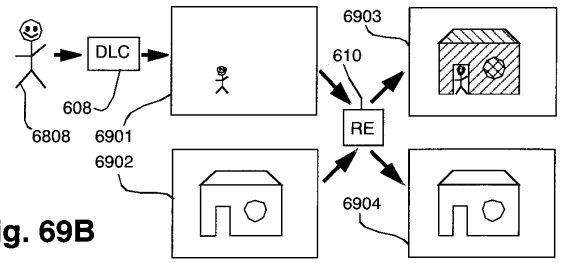


Fig. 69B

【図６９Ｃ】

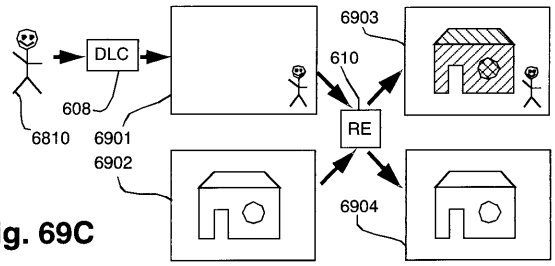


Fig. 69C

【図７０】

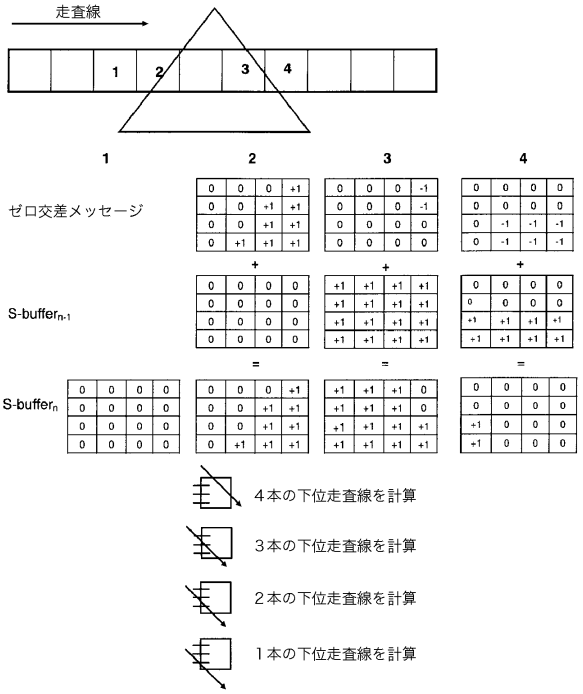


Fig. 70

【図７１】

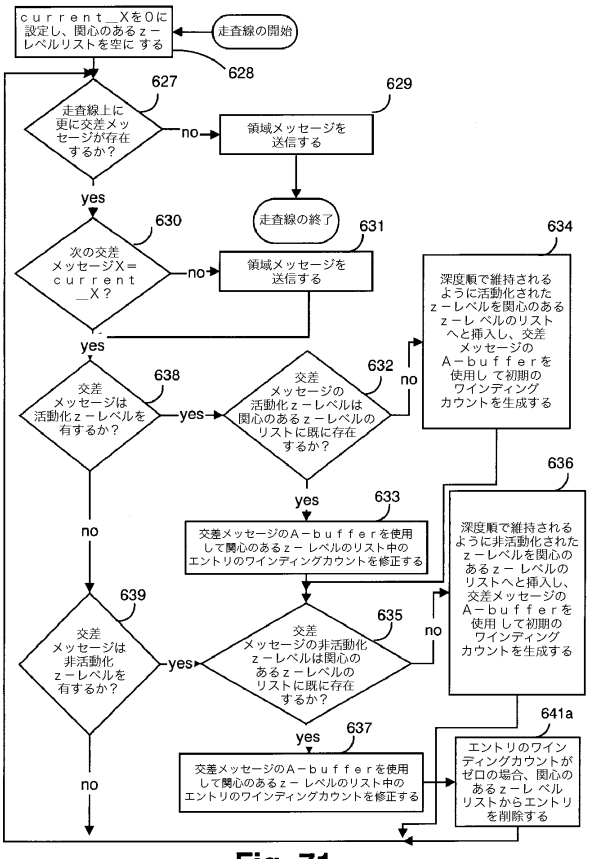


Fig. 71

フロントページの続き

(72)発明者 マイケル ジャン ローサー
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド内
(72)発明者 クリストファー ジェイムズ コーミー
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド内
(72)発明者 ステファン エドワード イーコブ
オーストラリア国 ２１１３ ニュー サウス ウェールズ州, ノース ライド, トーマス ホルト ドライブ １ キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテツド内

審査官 月野 洋一郎

(56)参考文献 米国特許第０６０２８５８３（ＵＳ，Ａ）
特表２００２－５４４５４４（ＪＰ，Ａ）
特開２０００－１４９０５３（ＪＰ，Ａ）
特開平１１－２８３０４７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｔ １／００ － １７／５０
Ｈ０４Ｎ １／４０ － １／６０
Ｇ０９Ｇ ５／００ － ５／３６