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

特許第4101275号
(P4101275)

(45)発行日　平成20年6月18日(2008.6.18)　(24)登録日　平成20年3月28日(2008.3.28)

(51)Int.Cl.　F I
G06T 15/00 (2006.01)　G06T 15/00 100A
G06T 15/00 400

請求項の数 12　(全 98 頁)

(21)出願番号　特願2006-515272(P2006-515272)
(86)(22)出願日　平成16年6月25日(2004.6.25)
(65)公表番号　特表2008-519318(P2008-519318A)
(43)公表日　平成20年6月5日(2008.6.5)
(86)国際出願番号　PCT/AU2004/000842
(87)国際公開番号　WO2004/114223
(87)国際公開日　平成16年12月29日(2004.12.29)
審査請求日　平成17年12月21日(2005.12.21)
(31)優先権主張番号　2003903448
(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】走査線ベースのラスタ画像プロセッサにおける奥行き追跡の方法

(57)【特許請求の範囲】
【請求項１】
走査線レンダラにおいて、走査線と交差する、ｘに関して順序付けられた連続するエッジ間に横たわる画素のスパンについて、図形オブジェクト画像の走査線のレンダリングを行う方法であって、
前記走査線のレンダリングの間に存在するフィルのセットを奥行きによりアドレスされるメモリに維持し、各スパン毎に前記セットは少なくとも前記スパン内でアクティブである図形オブジェクトのフィルを含み、単一走査線に沿ってｘ順序で前記メモリに維持されている前記セットをスパンごとに更新する際に、アクティブになった図形オブジェクトに関連するフィルを追加し、非アクティブになった図形オブジェクトに関連するフィルを除去することを特徴とする方法。
【請求項２】
前記フィルのセットの前記奥行きの順序が、
前記フィルのセット中の前記奥行きを、前記スパン上での奥行きの相対順序を示す、対応する第２の奥行きと関連付けるステップと、
マップを使用して、前記各第２の奥行きから前記フィルのセットの関連付けられた各奥行きにマッピングするステップと、
ｘについて昇順に、各スパン毎に前記フィルのセット及び前記マップを再利用するステップと、
サブセットへの奥行きの追加及びサブセットからの奥行きの除去の際に、前記フィルの

セット及び前記マップを順序付けられた状態に維持するステップと従ったマップを使用して維持されることを特徴とする請求項１に記載の方法。

【請求項３】

前記フィルのセットに奥行きが追加される際に、前記維持は、

前記フィルのセット内の各奥行きに、前記追加された奥行きのＺレベルよりも高いか低いかについてマーキングし、それによって１つのマークセットを作り出すステップと、

前記メモリに格納された前記フィルのセットと同じ順序に前記マークセットを順序付けるステップと、

前記マークセットを、前記マップによって決定される、それに関連付けられた第２の奥行きと同じ順序に順序変更するステップと、

前記マップを、前記追加された奥行きを含めて、それぞれに関連付けられた第２の奥行きに従って前記メモリ内に格納された、前記奥行きの新しい順序を反映するように順序変更するステップとを含み、前記マップの前記順序変更では前記順序付けられたマークセットを使用することを特徴とする請求項２に記載の方法。

【請求項４】

前記フィルのセットを、現スパンのラスタデータ出力に寄与している奥行きと、現在寄与していない奥行きとに区画化することを更に含み、前記フィルのセットは前記奥行き順に維持され、前記奥行きに関する順序付けが前記寄与区画及び前記非寄与区画に個別に施され、前記方法が、

前記スパン内の前記各奥行き毎に、前記区画のうちの対応する１つを示す最上位部分及び前記別個に施された奥行き順序付けを示す最小位部分をもつ複合インデックスを形成し、

マップを用いて前記複合インデックスを順序付けることによって前記区画化及び前記奥行きを順序付け、

前記フィルのセット中の前記各奥行きの前記複合インデックスを、寄与による前記区画化及び前記スパン上の各区画内の前記奥行きの相対的順序付けのどちらも示す、対応する更なる奥行きに関連付け、

前記マップを使用して、前記各更なる奥行きから前記フィルのセット中の前記各奥行きの前記複合インデックスにマッピングし、及び

ｘについて昇順に、各スパン毎に前記１つのフィルのセット及び前記マップを再使用し、

前記フィルのセット及び前記マップは、追加又は除去による前記フィルのセット中のある奥行きの寄与ステータスの変更に際して、順序付けられた状態を維持することを特徴とする請求項１に記載の方法。

【請求項５】

更に、前記フィルのセットへ奥行きが追加される際に、

前記追加奥行きを、寄与している又は寄与していないとタグ付けし、それによって前記追加奥行き用の前記複合インデックスを提供し、

前記フィルのセット内の各奥行きを、対応する複合インデックスに基づいて前記追加奥行きよりも高い又は低いとマーキングし、それによって前記フィルのセットと同じ順序に順序付けられ、前記メモリに格納されるマークセットを作成し、

前記マークセットを、前記マップによって決定された、それぞれに対応する前記複合インデックスと同じ順序に順序変更し、及び

前記マップを、前記追加された奥行きを含めて、それぞれに関連付けられた更なる奥行きに従って前記メモリ内に格納された、前記奥行きの新しい順序を反映するように順序変更し、

前記マップの前記順序変更は、前記順序付けられたマークセットによって決定されることを特徴とする請求項４に記載の方法。

【請求項６】

前記フィルのセット内の奥行きが寄与から非寄与に変わる際、

前記寄与区画内の前記フィルのセット内の各奥行きに対して、変更される奥行きのＺレベルよりも低い場合にマーキングを行い、同様に前記非寄与区画内の前記フィルのセット内の各奥行きに対して、変更される奥行きのＺレベルよりも高い場合にマーキングを行い、それによって前記フィルのセットが前記メモリに格納されている順序と同じ順序で順序付けられるマークセットを作成するステップと、

前記マークセットを、前記マップによって決定されたそれぞれの複合インデックスの順序に順序変更するステップと、

前記マップを、前記変更された奥行きの寄与ステータスの変更を含めて、それぞれに関連付けられた更なる奥行きに従って、前記メモリ内に格納された前記奥行きの新しい順序を反映するように順序変更するステップとを含み、

前記マップの前記順序変更は、前記順序変更されたマークセットを使用することを特徴とする請求項４に記載の方法。

【請求項７】

前記レンダリングが前記スパンの前記ラスタデータ出力を決定し、前記各奥行きは対応するフィルと関連付けられ、

前記フィルのセットと、前記フィルのセットの各奥行きに関連付けられた前記フィルとを使用して、前記１つのフィルのセットの前記奥行き順序によって提供される奥行き順序をもつフィルのサブセットを作成するステップを更に含むことを特徴とする請求項１に記載の方法。

【請求項８】

ラスタ画像プロセッサに入力を提供し、前記入力が図形オブジェクトからなるツリーの形状であり、前記ツリーがローカル奥行きによって順序付けられ、前記各図形オブジェクトが１つ又は複数のフィルに関連付けられ、前記各図形オブジェクト内の各フィルは、フィルのローカル奥行きがそれぞれの図形オブジェクトにとってローカルであり、各図形オブジェクトがそれぞれあるローカル奥行きと関連付けられるようにローカル奥行きに関連付けられ、前記各図形オブジェクトの前記ローカル奥行きが対応する親図形オブジェクトにとってローカルであり、前記方法は、

前記図形オブジェクトのツリーの奥行き優先トラバースを行うことにより、対応する前記図形オブジェクトに関連付けられた前記奥行きを決定するステップと、

各図形オブジェクトに関連付けられた前記フィルをローカル奥行き順序でトラバースし、前記トラバース中に遭遇した順序でフィルに連続する奥行きを割り当てるステップと、を含むことを特徴とする請求項１に記載の方法。

【請求項９】

前記メモリは、ハードウェア内に形成されることを特徴とする請求項１に記載の方法。

【請求項１０】

前記メモリは、ソフトウェアによって実装されることを特徴とする請求項１に記載の方法。

【請求項１１】

プログラムを記録しており、コンピュータデバイスに、走査線と交差しｘに関して順序付けられた連続するエッジ間に横たわる画素のスパンについて、走査線レンダラにおける図形オブジェクト画像の走査線をレンダリングさせるように構成されたコンピュータ可読媒体であって、

前記走査線のレンダリングの間に存在するフィルのセットを奥行きによりアドレスされるメモリに維持し、各スパンについて、前記セットが少なくとも前記スパン内でアクティブである図形オブジェクトのフィルを含み、単一走査線に沿ってｘ順序で前記メモリに維持されている前記セットをスパン毎に更新する際に、アクティブになった図形オブジェクトに関連するフィルを追加し、アクティブでなくなった図形オブジェクトに関連するフィルを除去することを特徴とするコンピュータ可読媒体。

【請求項１２】

走査線と交差しｘに関して順序付けられた連続するエッジ間に横たわる画素のスパンに

ついて、走査線レンダラにおける図形オブジェクト画像の前記走査線のレンダリングを行うためのコンピュータ装置であって、

前記走査線のレンダリングの間に存在するフィルのセットを奥行きによりアドレスされるメモリに維持し、各スパンに関して、前記セットが少なくとも前記スパン内でアクティブである図形オブジェクトのフィルを含み、単一走査線に沿ってｘ順序で前記メモリに維持されている前記セットをスパン毎に更新する際に、アクティブになった図形オブジェクトに関連するフィルを追加し、アクティブでなくなった図形オブジェクトに関連するフィルを除去することを特徴とするコンピュータ装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本出願は、米国特許法（35 U.S.C.）第１１９条のもとで、２００３年6月２６日に出願され、参照により、ここにその全体が記載されているかのようにここに完全に組み込まれるオーストラリア特許出願第２００３９０３４４８号に基づく優先権を主張するものである。

【０００２】

本明細書は、著作権保護の対象となる内容を含んでいる。著作権所有者は、検討を目的とした本明細書又は関連する特許庁のファイルからの関連資料の複製に異存はないが、その他の一切の著作権を所有するものとする。

【０００３】

本発明は、一般に図形(graphical)オブジェクトのレンダリングに関し、詳しくは、レンダリングを加速させるための図形オブジェクトのｚ順序(z-orders)の解決に関する。

【背景技術】

【０００４】

ラスタ画像プロセッサは、コンピュータグラフィックの分野において基本的なものである。ラスタ画像プロセッサ（ＲＩＰ）は、入力として図形プリミティブを取り込み、出力としてラスタ格子上に画素値のアレイを作り出す。本明細書では、２Ｄのレンダリングシステムについて考察する。

【０００５】

ＲＩＰの２つの重要なクラスは、オブジェクトベースのＲＩＰ及び走査線ベースのＲＩＰである。これらの２クラスのＲＩＰの違いは、それぞれの内部ループ、即ちそれぞれのアルゴリズムがその上を反復するエンティティにある。オブジェクトベースのＲＩＰは、レンダリングしなければならない全ての図形プリミティブ上を反復するが、走査線ベースのＲＩＰは、ラスタ順にレンダリングすべき各走査線上を反復する。

【０００６】

互いにその輪郭、フィル及びレンダリング奥行き（即ちｚ順序）という点で区別される１つの多角形セットをレンダリングするケースを考える。オブジェクトベースのＲＩＰは、一般的に、ｚ順にこの１つの多角形セット上を反復し、それぞれの多角形を順にレンダリングする。これは「画家のアルゴリズム」的手法である。対照的に、走査線ＲＩＰは、順に各走査線を検討して、各多角形のエッジが現在の走査線上のどこにあるかを判断する。次いで、現在の走査線と交差するエッジ間の画素スパンが埋められる。

【０００７】

走査線ベースのレンダリングシステムの１つの改良点は、アクティブエッジリストの使用にある。このような手法では、全ての走査線上の全ての多角形のエッジを検討する代わりに、走査線ＲＩＰは現在の走査線と交差するエッジだけのリストを維持し、それらのエッジを走査線毎に追跡する。この手法は、一般に前方差分技法を使用し、エッジはブレゼンハムの走査－変換アルゴリズムの一形態によって追跡される。

【０００８】

各走査線のレンダリングの開始時に、アクティブエッジのリストに新たなエッジを追加してもよい。その際に、アクティブエッジリストに追加する際に検討すべきエッジの数を

大幅に減らすために、ソートが重要である。具体的には、エッジのセット全体が、一般に始点のラスタ順に並べられる。ここで、始点とは、走査全体で最初に遭遇した端点を意味する。これにより、走査線の初めに、アクティブ化しつつあるエッジがあるとすれば、それはどのエッジなのかを迅速に判断することができる。走査線の終わりで、あるエッジが次の走査線と交差していないと判断された場合、そのエッジはアクティブエッジリストから外される。

【０００９】

上述のＲＩＰのもとでの一般的な走査線を考える。この走査線と交差する全てのエッジのｘ座標が知られており、この走査線のラスタ画像データを出力することが望まれると仮定した場合、次に解決すべき問題は、交差点間の画素スパンのラスタデータの内容を判断することである。即ち、ｘ順序で連続する交差点間のスパンが、ラスタ画像空間から、図形モデル空間内の１つの多角形セットへとマップされる。この手法は、ＲＩＰがアンチエイリアス処理を行うことができないことを見越している。また、この手法では、重なり合う多角形は、それぞれの奥行きによって互いに区別されることを意味している。即ち、交差点間のスパン内で露見した１つの多角形セットにおいて、最上位の多角形は、それが完全に不透明である場合、そのスパン全体に亘ってレンダリングされる唯一の多角形となる。最上位の多角形が完全に不透明でない（即ち、ある程度の透明さを含む）場合、それより下の多角形もスパンに寄与する。

【００１０】

走査線のレンダリングでは、オブジェクトベース（画家のアルゴリズム）のレンダリングシステムでは一般的な特徴及び欠陥であった、レンダリング結果を蓄積するための画面（フレーム）バッファの必要がなくなる。しかし、ここで、走査線レンダリングシステムは、上述の問題をどのように解決するかを検討することにより更に特徴付けることができる。具体的には、一部の走査線レンダリングシステムは、完全な画面バッファの必要性を走査線バッファまで軽減しており、他の走査線レンダリングシステムは、走査線バッファの必要をなくすことで、それよりも下位の出力ラスタデータバッファに直接レンダリングすることができるようになっている。

【００１１】

様々なラインバッファＲＩＰモデルを考慮することが適切である。まず、ラスタ画像データのラインバッファを維持できるＲＩＰがある。エッジはｘ座標（走査線内の画素位置）ではなく、ｙ座標（即ち、走査線）でソートされ、スパンはランダムなパターンでバッファ内に蓄積される。即ち、スパンのレンダリングが行われる順序は、それぞれのｘ順序に関連しない。このようなシステムでは、画素が書き込まれる際に、高いグレースケール出力値が低い値より優先されるようなｚ順序の概念は存在しない。現在の走査線について全てのスパンが蓄積された後、ｘ順序でラインバッファから出力バッファに書き出される。

【００１２】

上記のＲＩＰモデルの変型が、ラスタデータの代わりに交差点のラインバッファを維持する他の走査線ベースのＲＩＰに見ることができる。この場合も、エッジはｘではなくｙでソートされる。各走査線毎に、全ての交差点が判断され、次いでｘでソートされる。言い換えると、ラスタデータのラインバッファではなく、交差点のラインバッファが維持される。しかし、交差メッセージのラインバッファはランダムな順序で蓄積されることに再び留意されたい。

【００１３】

走査線ベースのＲＩＰの第３のクラスは、ラスタデータのラインバッファ、交差メッセージのラインバッファのいずれも持たない。これは、レンダリングに先立って全てのエッジをフルラスタ順序でソートすることで避けることができる。即ち、全てのエッジは開始のｘ座標及びｙ座標でソートされる。走査線の各画素が検討されるとき、アクティブとなりつつあるエッジがないかどうか、エッジのリストがチェックされる。

【００１４】

次に、奥行きによって分離された重なり合う多角形をサポートする全ての走査線レンダラに共通する問題である、所与の走査線上のスパンのラスタデータの内容を判断するという問題を検討する。先に紹介したように、その奥行きで順序付けられた、１つの一定多角形フィルセットが、潜在的にスパンに寄与する。各多角形が多角形の奥行きを継承する単一のフィルをもつと仮定すると、この問題は、フィルとそれらが起こる奥行きという点から検討することができる。

【００１５】

ｘ順序でスパンを検討するシステムでは、この問題の解決には、レンダリングがスパンからスパンへと進むに連れて現在アクティブ化している奥行きのテーブルを維持することが必要となる。スパンからスパンへの進展は、エッジによって交差された任意の画素上で起こる。即ち、スパンとは、現在アクティブ化している１つの奥行きセットがその上で一定である、現在の走査線上の区間である。アクティブとは、スパンと交差する全ての多角形を通る概念上の断面が、全てのアクティブな奥行き（即ち、最終画素値に寄与又は影響する全ての図形オブジェクト）を含むという意味である。

【００１６】

アクティブな奥行きセットは、それが奥行きの降順で維持されている場合、スパンのラスタ出力の決定を可能にする。フィルが透明さを含むことができない場合、アクティブな奥行きのテーブルの先頭のフィルが最上位となり、フィルのラスタデータに寄与する唯一のフィルとなる。透明さを含むことができる場合、テーブル中で後に続く奥行きに関連するフィルもラスタ出力に寄与することができる。

【００１７】

この現在アクティブな奥行きのテーブルを維持する問題の解決方法が、従来技術に存在する。例えば、レンダリングすべきページ画像に存在する全ての奥行きの完全なステータステーブルを作成することができる。この手法は、奥行きが線形アドレスメモリに存在しており、ある奥行きにフィルが存在する場合、その奥行きからフィルデータを記憶することができるメモリ内の位置が暗示されることを特徴とする。言い換えれば、各奥行きに１つずつ、フィル情報を指すポインタ用の１つの完全スロットセットがあり、スロットは奥行きの順序に配列されている。このようなテーブルへのフィルの挿入は、所与の奥行きについて必ずスロットがあるため、簡単である。しかし、スロット占有のパターンは、一般にまばらである。実際には、エッジ横断によって参照されるフィル及び奥行きを考慮した場合、この奥行きスロットの完全なテーブルをインデックスするためにフィルの奥行きが使用される。これは本質的には一次動作（order one operation)であり、ＲＩＰがリアルタイムの速さで動作できるようにする。

【００１８】

サマリテーブルと呼ぶことができるキャッシュが、そのような完全なテーブルへのルックアップの速さを向上させるために使用できる。それが必要なのは、一般に、ページに存在する奥行きの総数が非常に大きい可能性があるからである。例えば、現実のポストスクリプト（PostScript）印刷レンダリング作業のいくつかは、極めて大きな数の奥行きをもつ。非常に大きいサイズの完全な奥行きスロットテーブルのもう１つの結果は、このＲＩＰモデルのハードウェア実装が、往々にして完全なリストを内部的に維持できない点である。従って、テーブルは外部メモリに保持する必要があるが、その場合参照が遅くなる。奥行きスロットテーブルで対応できる数以上の奥行きが存在する場合、複雑さが大幅に増大するという代償を払って、追加のテーブルをスワップインすることができる。

【００１９】

更に、このモデルをもつＲＩＰは、フィルで埋められた奥行きのセットがフレーム毎に変化するアプリケーションである、アニメーションには容易に適合されない。また、アニメーションレンダラへの入力が確定的でなかった場合、占有された１つの奥行きセットも確定的でない。将来のフレーム上で見つかるフィルのために、多くの奥行きスロットを取っておかなければならない。或は、フィルはフレーム毎に異なる奥行きを与えられる可能性があるが、現在検討中の従来技術ＲＩＰのクラスでは、フィルは対応する奥行きと組み

合わせて密結合されている。即ち、フィルは、特定のメモリ位置にあることにより、示唆される奥行きをもつ。その結果、フィルの再割当てはデータの大幅な移動を伴うことがあるが、これは望ましくない。

【００２０】

合成というコンテクストでアクティブ奥行きのテーブルを維持する際に、更なる欠陥が見つかる。サマリテーブルに新たなフィルを挿入することは、フィルがページ上で見つかる全ての奥行きスロットのテーブルと同じ奥行きをもち、各フィルが対応する待機スロットをもつので、費用はかからない。しかし、スパンが合成される場合、サマリテーブルは非常に大きく、かつ普通は非常にまばらである。一方コンポジタが考慮すべき１つのフィルセットは一般に比較的小さくかつコンパクトであるため、動作にはコストがかかる。実際には、ステータステーブルと同じ深さの奥行き符号器では、多くの複雑さとシリコンとが消費される。この符号器は、次いで、先に詳述した奥行きで順序付けられた１つのアクティブフィルセットを検討するプロセスにより、問題となっているスパン上に合成される、コンパクトな１つのフィルセットを作成するために使用される。

【発明の開示】

【発明が解決しようとする課題】

【００２１】

本発明の目的は、既存の構成の１つ又は複数の欠点を大幅に克服し、或は少なくとも改善することにある。

【課題を解決するための手段】

【００２２】

本発明の一態様によれば、走査線と交差するｘ順序で連続する２つのエッジ間にある画素スパンについて、走査線レンダラによって図形オブジェクト画像の前記走査線をレンダリングする方法が提供され、この方法は、画素の前記スパンについて、走査線のレンダリングの間に存在するフィルのセットを奥行きの順に維持することを特徴とし、前記セットは前記スパン内でアクティブである図形オブジェクトのフィルを含み、単一走査線に沿ってｘ順序でメモリに維持されているセットをスパン毎に更新する際に、アクティブになった図形オブジェクトに関連するフィルを追加し、アクティブでない図形オブジェクトに関連するフィルが取り除かれる。

【００２３】

好ましくは、この奥行きのサブセットは、内容アドレス可能なメモリによって維持される。

【００２４】

本発明の他の態様によれば、前述のいずれか１つの方法を実装するための装置が提供される。

【００２５】

本発明の他の態様によれば、前述のいずれか１つの方法を実装するためのコンピュータプログラムを記録したコンピュータ可読媒体を含むコンピュータプログラム製品が提供される。

【００２６】

また、本発明の他の態様も開示されている。

【００２７】

本発明の少なくとも１つの実施形態を、図面を参照して説明する。

【発明を実施するための最良の形態】

【００２８】

目次

１．序論
　１．１　座標空間
　１．２　図形オブジェクト
　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：変換マトリックス
２．１．２　図形オブジェクト及びその奥行き
２．２　表示リスト
２．２．１　フレームレンダリング
２．２．２　図形オブジェクト及びｚレベル
２．２．３　ローカル奥行きと絶対奥行き
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　エッジの生成
３．４　ストロークのエッジ及びｚレベルへの分解
３．４．１　直線エッジのストローキング
３．４．２　曲線エッジのストローキング
３．４．３　継目のストローキング
３．４．４　等しい又は向かい合うエッジ継目のストローキング
３．４．５　パス終点のエンドキャップの生成
３．４．６　ストロークされたエッジとストロークされていないエッジとの間のエンドキャップ
３．４．７　不透明ストロークへのｚレベルの割当て
３．４．８　透明ストロークへのｚレベルの割当て
３．４．９　ストローキングプリミティブの変換
３．５　フィルタリング
３．６　エッジ追跡パラメータの生成
３．６．１　直線エッジ追跡パラメータの生成
３．６．２　２次ベジエ曲線追跡パラメータの生成
３．６．３　符号の判定
３．６．４　楕円弧追跡パラメータの生成
３．６．５　グリフエッジ追跡パラメータの生成
４．ソーティング
５．エッジ処理
５．１　入力及び出力
５．２　トップレベル動作
５．３　アクティブエッジ追跡
５．４　エッジ処理の例
５．５　エッジのアクティブエッジへの変換及びエッジパーシスタンス
５．５．１　静的エッジパーシスタンス
５．６　アクティブエッジ処理
５．７　エッジの追跡
５．７．１　直線の追跡
５．７．２　２次ベジエ曲線の追跡
５．７．３　２次多項式フラグメントの追跡
５．７．４　楕円弧の追跡
５．７．５　グリフの追跡
５．８　アンチエイリアス処置及び交差メッセージの生成
５．８．１　グリフのための交差メッセージの生成

５．９　交差メッセージの例
５．９．１　他の例
５．１０　アクティブエッジ及び交差メッセージの順序変更
６．Ｚレベル・アクティブ化モジュール
６．１　順序付けられた１つの関心対象Ｚレベルセット
６．２　Ｚレベル・アクティブ化モジュールにおける制御のフロー
６．３　Ｚレベルのアクティブ化及び非アクティブ化：屈曲カウント
６．４　順序付けられた１つの関心対象Ｚレベルセット：続き
６．５　順序付けられた１つの関心対象Ｚレベルセットへの新規Ｚレベルの追加
６．５．１　ハードウェアにおける順序付けられた１つの関心対象Ｚレベルセットの維持
６．６　ランの処理
６．７　ＳバッファのＡバッファへの変換：屈曲規則
６．８　ランの処理：続き
７．合成モジュール
７．１　中間
７．２　ｚレベルのフィル
７．３　基本フロー
７．４　図的概要
７．５　寄与計算
７．６　ボトムアップ合成
７．７　トップダウン合成
７．８　他の合成手法
７．９　トップダウンの利点
８．画素生成
８．１　線形勾配画素生成
８．２　半径方向の勾配画素生成
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　ハーフトーニング
１０．実装

１．序論

本明細書では、最小限のコンピュータ資源を使用して２Ｄの図形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）を記述する。そのような資源レベルが適用される可能性があるものの例としては、ポータブルデバイス又は小さなディスプレイをもつものが含まれ、そのようなハンドヘルドコンピューティングデバイスとしては、携帯電話機及び携帯ゲーム機、並びにプリンタ及びコピー機などのオフィス機器が含まれる。ＴＣＩＥシステム６９９のトップレベル図が図５６に示されるが、そこでは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。各モジュールを、システム中のデータの流れの順に説明する。これらのモジュールは、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、所望の出力を表す情報を準備し、レンダリングエンジン６１０はこの情報を使用してその出力画像を生成（即ち、ディスプレイ装置又はフレームバッファにレンダリング）する。ＴＣＩＥシステム６９９は、時間的に分散された一連の出力画像を生成するために使用することができ、以下、これらの出力画像を「フレーム」と称する。このＴＣＩＥシステム６９９の使用は、出力ディスプレイ上でアニメーション（又は「映画」）が再生されるという効果を生み出す。

【００２９】
１．１　座標空間
図５６を参照すると、システム６９９の第１のモジュールは、図形オブジェクト及びそれらに関する情報の集合体を維持するドライバモジュール６１５である。図３４は、システム６９９によって使用される空間を示す。図３４には、まず、オブジェクト空間３３５内に示されている図形オブジェクトが示されている。次いで、同じ図形オブジェクトが、グローバルな論理空間３３６内に変換されているところが示されている。次いで、同じ図形オブジェクトが、レンダリング空間３３７内に変換されているところが示されており、最後に、同じ図形オブジェクトが、表示空間３３８内に変換されているところが示されている。
【００３０】
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、図形オブジェクトの配置変換によって実現される。この配置変換は、後述する変換マトリックスの階層構造の積であってもよい。グローバル論理空間３３６からレンダリング空間３３７への変換は、グローバル論理座標を（アンチエイリアス処理を目的として）サブピクセルに変換する視野変換によって実現される。レンダリング空間３３７から表示空間３３８への変換は、構成要素であるサブピクセルから表示画素を作り出すアンチエイリアス処理によって実現される。１対１のアンチエイリアス処理での縮退の場合には、レンダリング空間３３７と表示空間３３８とは同一であり、即ち論理空間３３６を直接、表示空間３３８に変換することができる。
【００３１】
１．２　図形オブジェクト
システム６９９への入力は、１つの図形オブジェクトセット及びそれに関連するメタデータからなる。図１６（ｃ）は、ディスプレイ上に図形オブジェクト１７１がレンダリングされているところを示しており、対応する構成要素は図１６（ａ）及び１６（ｂ）に示されている。図形オブジェクトは、新しい描画位置、直線及び曲線という描画プリミティブのうちの１つ又は複数からなる、順序付けられた１つの描画プリミティブセットとして記述される２次元表示プリミティブである。描画プリミティブは、ある図形オブジェクトの輪郭の一部を示す。各プリミティブは、１つ又は複数の座標に関連付けられている。新しい描画位置は、オブジェクトの基点からの絶対オフセット、又は直前のプリミティブの終点との相対オフセットとして指定することができる。新しい描画位置は単一の座標、直線は座標の対、曲線は一連の３つの座標として、それぞれ示される。直線は、線の始点及び終点を定義するために座標の対を使用する。曲線は、最初の座標がベジエ曲線の始点、第２の座標が制御点、第３の座標がベジエ曲線の終点をそれぞれ定義する２次ベジエ曲線として実装される。ベジエ曲線は、当業者には周知である。
【００３２】
エッジの座標は、順序付けられた１つの相対座標セットとして記憶され、これによってメモリ記憶要件が軽減され、エッジ方向も決定される。直線又は曲線の最初の座標は、直前の描画プリミティブの最後の座標であることが示唆される。表１は、図１６（ｃ）に示された表示オブジェクトを形成することができる、順序付けられた１つのプリミティブセットの例である。この例では、Ｙ座標は下方に増加し、Ｘ座標は右方に増加する。また、新しい描画位置、直線、及び曲線に対応する用語はそれぞれＭＯＶＥＴＯ＿ＡＢＳ、ＭＯＶＥＴＯ＿ＲＥＬ、ＬＩＮＥＴＯ及びＣＵＲＶＥＴＯである。この例の始点は（０，０）１４１である。
【００３３】

【表１】

表1

プリミティブのタイプ	座標（相対、MOVETO_ABS を除く）
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 から (-20,0) 151 へ
CURVETO 167	(-20,0) 152 から (0,20) 153 へ
CURVETO 169	(0,20) 154 から (20,0) 155 へ
CURVETO 170	(20,0) 156 から (0,-20) 176 へ

【００３４】

１．３　グリフ(Glyphs)

グリフは、必ず表示空間に直接書き込まれるという制約を更にもつ、特別なタイプの図形オブジェクトである。グリフは、レンダリングされる形状が（i）小さく、（ii）画素境界上にぴったりと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。

【００３５】

グリフで適切に表現される形状の例には、ヒンテッドフォント(hinted font)の文字が含まれる。

【００３６】

グリフは、パスの代わりに、２つの関連するフィルをもつ、画素当たり１ビットのビットマップとして表現される。このピットマップはマスクのように働き、ビットが設定されている場合には「オン」フィルが表示され、ビットが設定されていない場合には「オフ」フィルが表示される。

【００３７】

１．４　ｚレベル

ｚレベルは、図形オブジェクトのエッジのサブセットによって囲まれたディスプレイの部分がどのように色付けされるかを記述するために使用される表示プリミティブである。例えば、ｚレベルは、囲まれた領域を単一の色で塗りつぶされているものとして記述することができる。ｚレベルには、絶対奥行きも割り当てられるが、この絶対奥行きは、どのｚレベルがどのｚレベルの上に表示されるかを指定するために使用される整数値である。より高い絶対奥行きをもつｚレベルは、より低い絶対奥行きをもつｚレベルの上にレンダリングされる。

【００３８】

直線及び曲線描画プリミティブには、最大２つのｚレベルが関連付けられる。即ち、２つのｚレベルがある場合は、第１のｚレベルはプリミティブの描画方向の左にレンダリングされ、第２のｚレベルはプリミティブの描画方向の右にレンダリングされる。図３３（ａ）～（ｃ）は、先に図１６（ｃ）でオブジェクト１７１として図示された図形オブジェ

クト３３１を作成するのに使用される左側及び右側フィルを示すことで、この概念を図示している。

【００３９】

図３３（ａ）は、描画プリミティブ３１６～３２９を示す。これらプリミティブ３１６～３２９は、図３３（ｃ）に示されるｚレベル３３３，３３２及び３３４を参照している。ｚレベルは絶対奥行き順に示されている。即ち、それぞれ２，３及び４の絶対奥行きをもつことができる。表２は、描画プリミティブがどのｚレベルを参照しているかを示す表である。例えば、ＬＩＮＥＴＯ３１６は下向きであり、描画方向の左側にｚレベル３３２をもつ。レンダリング結果は、図３３（ｃ）に示されている。

【００４０】

【表２】

表２

描画プリミティブ	左側ｚレベル	右側ｚレベル
316	332	None
317	332	None
330	332	None
318	332	None
319	332	None
320	332	None
321	332	None
322	333	None
323	333	None
324	333	None
325	333	332
327	334	332
326	334	332
328	334	332
329	334	332

【００４１】

ｚレベルのスタイルとしては、単色、１つ又は複数の色で記述される線形ブレンド、１つ又は複数の色で記述される径方向ブレンド、又はビットマップ画像があり得るが、それだけには限定されない。これら全てのｚレベルのスタイルはまた、透明度（アルファ）チャネルもサポートする。図３３のｚレベル３３３，３３２及び３３４は、単色スタイルのｚレベルを表している。これらのｚレベルは、パイプラインの大部分を通して、変更されることなしに使用される。

【００４２】

１．５　ストローキング

描画プリミティブは、ペン幅に関連付けることができる。ペン幅をもつ描画プリミティブは、複数のエッジ（エッジは幅をもたない）に変換される。これらのエッジは、ペンのストロークを表す、閉じられ塗りつぶされた形状を形成する。詳細な議論については、「変換、モーフィング及びストローキング」と題するセクションを参照されたい。

【００４３】

１．６　モーフィング

モーフィングもまた、当技術分野でよく知られている。モーフィングは、ある図形オブジェクト用の２つの描画プリミティブセットと、２セット間の補間によってこの図形オブジェクトを描画するよう指定するある比率を供給することとして定義することができる。

これも、「モーフィング、ストローキング及び変換モジュール」と題するセクションで詳しく論じる。

【００４４】

２．ドライバモジュール

ドライバモジュール６１５を、それが取り扱う情報と、レンダリングパイプラインの残りの部分にどの情報を渡すかという点から論じる。ドライバモジュール６１５の役割は、描画プリミティブの集合体をまとめ上げることである。描画プリミティブはまず、先述のように、図形オブジェクトにまとめることができる。

【００４５】

図形オブジェクトは、後述のようにスプライトにまとめることができる。これらのスプライトは、集合体全体に当てはまるプロパティをもつものとして指定することができる。ドライバの主な役割は、図形パイプラインの残りの部分を複雑にせずに、スプライトに対する効率的なハイレベル操作を可能にすることである。ドライバモジュール６１５が、図形パイプライン内の後続モジュールのために描画情報を出力するとき、あるスプライトのプロパティ（例えば、その変換マトリックス）が、その各図形オブジェクトの各描画プリミティブに適用される。これにより、後続モジュールが有向エッジ及びｚレベルを処理するだけでよくなる。

【００４６】

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。スプライトは当技術分野において周知であり、ドライバモジュール６１５においては、そのスプライトのコンテクスト内に存在する変換マトリックス、奥行き及び図形オブジェクトのリストをもつプリミティブを参照する。スプライトは、ゼロ又は１つ又は複数の図形オフジェクト及びゼロ又は１つ又は複数のスプライトを含むことができる。ここで「含む」とは、スプライトの変換マトリックスが、問題のスプライトが所有する全ての図形オブジェクト及びスプライトに適用されることを意味する。スプライトが他のプリミティブを含むという概念は、そのスプライトが含む全ての図形オブジェクト及びスプライトの奥行きが、そのスプライトにとって「ローカル」であることも意味する。図形オブジェクトは、他の図形オブジェクトやスプライトを含まない。

【００４７】

２．１．１　スプライト：変換マトリックス

変換マトリックスは、当技術分野で周知である。スプライトの変換マトリックスは、そのスプライトが所有する全ての図形オブジェクトに適用され、そのスプライトのローカル空間を画定する。図１７（ｂ）では、２つのスプライト及び２つの図形オブジェクトが提供され、それらのレンダリングで用いられるべき方法がツリーによって記述されている。スプライト１８５は、スプライト１８９、図形オブジェクト１８０のいずれも含む。即ち、リンク１８８及び１８６は所有関係を表す。スプライト１８９は、第２の図形オブジェクト１８２を含む。

【００４８】

図１７（ａ）は、この例で存在する変換マトリックスのジオメトリを表す。空間１７９は、スプライト１８５内のオブジェクトを含む。オブジェクト１８０は、空間１７９内に位置するのが見え、オブジェクト１８０の構成要素である描画プリミティブの座標が空間１７９を参照している。図１７（ｂ）のスプライト１８９も、この空間１７９内に位置している。空間１８１は、スプライト１８９の図形オブジェクトが位置する空間を表している。

【００４９】

この例では、スプライト１８９は回転、平行移動及び拡大縮小をもつ変換を有する。平行移動は点線１８３、回転は角１８４、拡大縮小はスプライト１８５及び１８９の空間を表すために使用される軸の目盛１７９及び１８１の相対的な大きさによって、それぞれ表されている。この例では、拡大縮小はいずれの軸でも同一である。

【００５０】
引き続き図１７（ａ）を参照すると、スプライトが所有する図形オブジェクトの配置を表す変換マトリックスが、そのスプライトの配置を記述する変換マトリックスと連結されている。図１７（ａ）の例では、図形オブジェクト１８２に適用される変換マトリックスは、スプライト１８９及び１８５のそれぞれの変換マトリックスが連結されたものである。オブジェクト１８２のためのこの合成変換マトリックスは、オブジェクト１８２が含む全ての図形プリミティブに適用される。
【００５１】
２．１．２　図形オブジェクト及びその奥行き
図形オブジェクトの奥行きは、そのオブジェクトを含むスプライトに対してローカルである。これは、図３１（ａ）～３１（ｃ）に示される。図３１（ａ）に示されるレンダリングツリーにおいて、ノード２９４は他のスプライト２９６及び図形オブジェクト３０２を含むスプライトを表している。所有関係は、それぞれ有向直線２９５及び３０１で示されている。同様に、スプライト２９６は、それぞれ所有関係２９７及び３００で示される図形オブジェクト２９８及び２９９を含む。表３に、これらの全てのプリミティブのローカル奥行きを示す。
【００５２】
【表３】

表３

ラベル	プリミティブ	ローカル奥行き	絶対奥行き
294	スプライト	2	n/a
296	スプライト	2	n/a
298	図形オブジェクト（円）	1	3
299	図形オブジェクト（四角形）	2	4
302	図形オブジェクト（三角形）	1	2
309	背景ｚレベル	1	1

【００５３】
ローカル奥行きの概念は、図３１（ａ）のサブツリーの外観を見ることで明らかにすることができる。図３１（ｂ）は、スプライト２９６（オブジェクト２９８及び２９９として分離してレンダリングされる）の視覚的外観を示す。オブジェクト２９８は、オブジェクト２９９よりも小さい奥行き値をもつため、オブジェクト２９８はオブジェクト２９９の下に見える。図３１（ｃ）は、図形オブジェクト３０２及びスプライト２９６の外観を示す。図形オブジェクト３０２の方が奥行き値が小さいため、２９６の下に見える。スプライト２９６の子のローカル奥行きは、ローカル奥行きの凡例３０４に従って保存されることに留意されたい。
【００５４】
ノード３０３は、所有ツリーのルートであり、全ての最上位スプライトを含むスプライトを表している。ルート３０３は、バックグラウンドｚレベル３０９を所有し、このｚレベルは常にその最下位のローカル奥行き（これは最下位のグローバル奥行きでもある）にある。従って、このバックグラウンドｚレベルは、全ての図形オブジェクトよりも下位にある。
【００５５】
２．２　表示リスト
それぞれの所有関係、ローカル奥行き及び変換を含めた、単一のフレームにおける全てのスプライト及び図形オブジェクトのリストは、表示リストと呼ばれる。表示リストは、当技術分野で周知である。ドライバモジュール６１５は、表示リストをツリーの形式で維

持する。このツリーは、所有関係に従ってスプライト及び図形オブジェクトをまとめたものであり、これらはローカル奥行きに従って順序付けられる。図３１（ａ）は、この構成も示している。例えば、スプライト２９４の子はローカル奥行きに従って順序付けられている。第１の子３０２は、最下位のローカル奥行き１にあり、次の子２９６はより高いローカル奥行き２にある。

【００５６】

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、ディスプレイ上にレンダリングされる前に改変することができる。表示リストは、フレーム間で保持される。

【００５７】

２．２．２　図形オブジェクト及びｚレベル

スプライトが複数の図形オブジェクトを参照できるのと同様に、図形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。これらの各ｚレベルは、図形オブジェクト内に奥行きをもつ。例えば、図形オブジェクト図３３（ｃ）は、図３３（ｂ）に示されるような奥行きをもつ３つのｚレベルを必要とする。

【００５８】

２．２．３　ローカル奥行きと絶対奥行き

ドライバモジュール６１５内ではローカル奥行きが使用されているが、後続のモジュールは絶対奥行きを必要とする。ドライバモジュール６１５は、各ｚレベルに絶対奥行きを割り当てる。

【００５９】

図形オブジェクトのｚレベルの絶対奥行きは、ドライバモジュール６１５から後続モジュールに渡され、そこでは単に奥行きと呼ばれる。当然のことながら、全ての後続モジュールは、絶対奥行きが割り当てられたｚレベルを扱う。これらの割当ては、レンダリングの直前に、各フレーム毎に行われる。これは、表示リストに新しい表示オブジェクト又はスプライトが挿入されるか、或は古いものが廃棄された場合にのみ行われる。

【００６０】

図３１（ａ）が単一のフレームにおける全てのプリミティブを示しているとすると、上の表３は図３１（ａ）の全てのプリミティブの絶対奥行きを示している。単一のフレームにおける全てのｚフレームの絶対奥行きは１で始まるが、これは特別なバックグラウンドｚレベルに割り当てられる。ｚレベルは、バックグラウンドｚレベルから上向きに連続する整数で番号付けされている。スプライトは、絶対奥行き自体をもたない（但し、その子孫の奥行きに関する情報を記録することはできる）。スプライトは、それ自体が含む図形オブジェクトに関する情報（例えば配置変換）用の容器として存在するが、それ自体は図形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モジュールは、スプライトを使用しない。また、図形オブジェクトはドライバモジュール６１５外には存在しない。図形オブジェクトのローカル奥行きは、その図形オブジェクトを構成する描画プリミティブによって参照されるｚレベルの絶対奥行きを計算するために使用される。

【００６１】

絶対奥行きは、フレーム毎に１回だけ、表示リストの内容がレンダリングされる際に計算される。これらの絶対奥行きを計算する方法が、表示リストツリーを示す図４４に示されている。表示リストツリーのルートノード４４０は、他の全てのｚレベルの下位にあるバックグラウンドｚレベル４４１を所有する。即ち、バックグラウンド４４１は、最低位の絶対奥行きである１をもつ。ノード４３２～４３９によって、表示リストツリーの残り部分が構成される。

【００６２】

絶対奥行きは、点線４４２で示される、ツリーの奥行き優先トラバース中に各図形オブジェクトの各ｚレベルに割り当てられる。図形オブジェクトのｚレベルへの絶対奥行きの割当ては、その図形オブジェクト内の最下位のｚレベルから始まるが、これには「直前」の図形オブジェクトの最高位のｚレベルよりも１つだけ高い絶対奥行きが割り当てられる

。ここで「直前」とは、その図形オブジェクトが奥行き優先トラバースで直前に訪問した図形オブジェクトであることを意味する。図４４では、奥行き優先トラバースが図形オブジェクト４３８に到達しているところが示されている。オブジェクト４３８の最下位ｚレベルに絶対奥行きを割り当てる際、直前に訪問した図形オブジェクト４３５内の最高位ｚレベルよりも１つだけ高い絶対奥行きが割り当てられる。

【００６３】

オブジェクト４３８が更にｚレベルをもつ場合、最下位ｚレベルのすぐ上のｚレベルから最高位のｚレベルへと順次絶対奥行きが割り当てられる。全てのスプライト及び図形オブジェクトは独立したローカル奥行き範囲をもっており、従ってインターリービングが不可能であることを、読者は想起されよう。表示リストツリーの全てのノードには、絶対奥行きが記憶される。図形オブジェクトは、その最高位のｚレベルの絶対奥行きを格納する。スプライトには、その子孫の最高位のｚレベルの絶対奥行きが記憶される。ツリーのトラバースは、必ずルートから開始される。

【００６４】

ノードの挿入又は除去によって、表示リストの一部のみが変更された場合、効率を高めるために、ツリーの最小限更新が実施される。表示リストツリーのトラバース中、絶対奥行きの更新は、変更が加えられた最初のノードからのみ実施される。これより（トラバースの順序で）前の図形オブジェクトのｚレベルは、それぞれの有効絶対奥行きを保持する。最初の変更ノードに到達するまでは、絶対奥行きの追跡は、各ノードの最上位絶対奥行きに注目することで行われる。最初の変更ノードに遭遇すると、絶対奥行きの割当てはこの注目された値から続行される。

【００６５】

例えば図４４で、オブジェクト４３８が表示リストツリーにおける新しいノードであり、これが直前のフレームからの唯一の変更であった場合、絶対奥行きの割当てはオブジェクト４３８から開始される。

【００６６】

３．変換、モーフィング及びストローキング

入力として、図５６の変換、モーフィング及びストローキングモジュール６１６は、ドライバモジュール６１５から、レンダリングすべき描画オブジェクトの１つ又は複数の輪郭を全体で画定する順序付けられた１つ又は複数の座標セットを取る。順序付けられた座標の各セットは、下記のものを含む、追加のパラメータを伴うことができる。

【００６７】

－　モーフ比
－　変換マトリックス
－　ストローク幅
－　左側ｚレベルの参照
－　右側ｚレベルの参照、及び
－　ストローク色ｚレベルの参照

モーフ比、変換マトリックス及びストローク幅は、図形オブジェクトの輪郭を記述する順序付けられた座標セットがどのようにディスプレイ上に配置されるべきかを記述する。残りのパラメータは、順序付けられた座標セットによって画定される輪郭内の表示画素の色、混合又はテクスチャを示す複数のｚレベルの参照である。

【００６８】

図４６は、モジュール６１６が、順序付けられた各段階的座標セットに対して行う処理を示すフロー図である。この処理の目的は、順序付けられた各段階的座標セットを、それぞれが少なくともレンダリング空間内の始点座標及び終点座標によって記述される、対応する順序付けられた１つのエッジセットへと変換することである（図１６（ａ）を参照されたい）。即ち、エッジは方向性をもつ。エッジが所有するｚレベルの左及び右参照は、この方向性に関する。モーフ比、変換マトリックス及びストローク幅の各パラメータは、この処理を制御するために使用される。

【００６９】
更に、モジュール６１６は、任意選択で１つ又は複数のグリフ記述を受け入れる。これらの記述には、以下の情報が含まれる。
【００７０】
－　グリフ位置
－　グリフ高及びグリフ幅
－　グリフビットマップ
－　「オン」のｚレベルの参照（場合によりヌル（NULL））
－　「オフ」のｚレベルの参照（場合によりヌル）、及び
－　グリフ記述は、ステージ４７０でパイプラインに入る。
【００７１】
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順序付けられた１つの座標セットを受け取る（即ち、図４６においてステージ４６４で「yes」）とき、各座標の２つのバージョンを、モーフ比パラメータとともに受け取る。受け取った各座標の１つのバージョンは、開始バージョンと呼ばれる。受領した各座標の第２のバージョンは、終了バージョンと呼ばれる。モジュール６１６は、ステージ４６７で、モーフ比パラメータを使用して、補間によって受領した各座標の単一の中間バージョンを得る。例えば、図３５（ａ）及び（ｃ）は、それぞれモーフ「開始」形状及びモーフ「終了」形状を表す順序付けられた座標セットを示している。いずれの順序付けられた座標セットも、「論理」座標空間内の原点３４２で開始し終了するように示されている。モーフ処理の目的は、図３５（ａ）の「開始」バージョンと、図３５（ｃ）の「終了」バージョンとの間を補間することによって、図３５（ｂ）で示されるようなこの形状の中間バージョンを作成することである。各座標の（「開始」及び「終了」）バージョンは、順序付けられた座標セットの対として提供され、この例では３３９及び３４６が第１の対となり、その後に３４３及び３４０，３４４及び３４７、更に最後の対３４１及び３４２が続く。モーフ比は、各座標対から中間バージョンを補間するために使用される。図３５（ａ）の「開始」形状を表すのに０のモーフ比が使用され、図３５（ｃ）の「終了」形状を表すのに１のモーフ比が使用される場合、０．２５のモーフ比が図３５（ｂ）に示された形状に対応する。従って、モーフ処理で、出力として３５０、３５１、３５２及び３５３で示される順序付けられた複数座標セットが作成される。
【００７２】
３．２　変換
次いで図４６において、ステップ４６８で座標に変換マトリックスが適用される。変換マトリックスを用いると、座標（従って、描画オブジェクトも）の回転、拡大縮小及び／又は平行移動を指定できるようになる。この説明では、変換マトリックスは座標を（従って、それらが記述する描画オブジェクトも）「論理空間」から「レンダリング空間」へと変換するといわれる。
【００７３】
３．３　エッジの生成
順序付けられた座標セットが（適切な場合）モーフィングされ、変換された後、図４６のフローチャートの後続ステージ４６５で、順序付けられた座標セットは順序付けられたエッジセットに変換される。ステージ４６５がどのように実現されるかを示す例が、図１５のフローチャートに示されている。この例で、座標は直線、２次ベジエ曲線及び新しい描画位置からなる３タイプのうちの１つの描画プリミティブの一部を形成することができるものと仮定する。この例ではまた、どんなタイプの描画プリミティブが後に続き、描画プリミティブの終点にいつ到達するかを示す追加情報（例えば、タグバイト）が提供されるものと仮定する。
【００７４】
まず、図１５で示されるように、ステップ１４０で、現描画位置が（０、０）に初期化

され、次いでステップ１２４，１２５，１２６及び１２７で次に遭遇する「後続の」描画プリミティブのタイプが決定される。

【００７５】

前述のタイプが、後続座標が新しい描画位置を表すことを示す場合（ステップ１２４）、ステップ１２８で座標が読み取られ、ステップ１３１で新しい現描画位置を設定するために使用される。

【００７６】

タイプが、後続座標が直線エッジを表すことを示す場合（ステップ１２５）、ステップ１２９で、現在描画位置に等しい開始座標が与えられた新しい直線エッジが生成される。次いで、ステップ１３２で座標が読み取られ、これはステップ１３４で新しい直線エッジの終了座標用に、ステップ１３６で新しい描画位置用にそれぞれ使われる。

【００７７】

タイプが、後続する２座標が（例えば、２次ベジエ）曲線を表すことを示す場合（ステップ１２６）、ステップ１３０で、現在描画位置に等しい開始座標が与えられた新しい曲線が生成される。次いで、ステップ１３３で第１の座標が読み取られ、ステップ１３５で新しい曲線の「制御点」座標として使用される。次いで、ステップ１３７で第２の座標が読み取られ、これはステップ１３８で新しい曲線の終了座標用に、ステップ１３９で新しい描画位置用にそれぞれ使われる。

【００７８】

ステップ１２７で、順序付けられた座標セットの終わりに到達するまで、処理が続けられる。

【００７９】

３．４　ストロークのエッジ及びｚレベルへの分解

図４６に戻ると、ステップ４７１で、処理の次のステップが任意選択で利用できるが、それはストローキングステップ４６６である。ストローキングは、曲線及びベクトルのパスに沿ってトレースを行う際に所与の太さをもつペンを使用したかのような効果をシミュレートするような１つ又は複数の輪郭を生成する処理である。ストローキングの例が、図５０（ａ）～（ｅ）に示されている。図５０（ａ）は、ストロークすべき順序付けられた３つのエッジのセット５２３を示す。図５０（ｂ）には、生成されたストローク輪郭５２４が示されているが、これはストロークすべき３つのエッジのパスを、所与のペン幅５２５をもつ、先端が円形のペンでトレースした効果をもたらす。

【００８０】

図４６では、ストローキングステップ４６６を変換ステップ４６８の後で実施しなければならないと示唆されているが、変換の前にストローキングが行えるようにし、それによって異なる結果を生み出すことができるようにすることも望ましいかもしれない。例えば、再び図５０（ａ）を参照すると、エッジ５２３によって表される元の形状がまず図５０（ｃ）に示される形状５２６を作成するように変換され、次いでストロークされる場合、結果は図５０（ｄ）に示される形状５２８となる。しかし、エッジ５２３によって表される原形がまずストローク５２４され、次いで同じ変換が適用される場合、結果は図５０（ｅ）に示される形状５２７となる。ＴＣＩＥシステム６９９は、ストローキング４６６の後にステップ４６８の座標変換を任意選択で実施することで、どちらの結果も実現することができる。

【００８１】

ストローキングに関する以下の議論では、ストローキングは、正のＹ軸が正のＸ軸に対して時計回りに９０度ずれている座標系（例えば、Ｘは右方に向かって増加し、Ｙは下方に向かって増加する）で実施されると仮定していることに留意されたい。

【００８２】

モジュール６１６は、元の順序付けられた１つのエッジセット内のエッジを次々に通過することによって、その順序付けられたエッジセットのストローキングを行う。各エッジ毎に、原エッジのパスに沿ってセンタリングされトレースされた先端が円形のペンのスト

ロークの左側及び右側限界に対応する、新しい左側エッジ及び新しい右側エッジが生成される。２つの連続する原エッジの間、及び閉じた形態を形成しない元の順序付けられたエッジセットの始点と終点のエンドキップとの間で継目を生成するために、楕円弧が使用される。順序付けられたエッジセットは、順序付けられたエッジセットの最初のエッジの始点座標が、順序付けられたエッジセットの最後のエッジの終点座標と等しい場合にのみ、閉じる。

【００８３】

孤は、オブジェクト空間内に円弧として配置され、次いで後のセクションで説明される処理によって、表示空間内の楕円弧へと変換される。このプリミティブは現在、内部的にのみ使用されているが、孤はユーザにとって利用可能な描画プリミティブとして簡単に露出させることができることに留意されたい。

【００８４】

３．４．１　直線エッジのストローキング

図４９（ａ）は、始点座標４７４及び終点座標４７２で表される、ストロークすべき直線エッジ４７３の例を示す。図４９（ｂ）は、直線エッジ４７３のストローキングの望ましい結果を示しており、この結果は左側エッジ４７７及び右側エッジ４７８を含む。

【００８５】

直線エッジをストローキングする際、直線エッジの方向に対して反時計回りに９０度の方向をもち、ストロークのペン幅の半分の長さをもつ左側法線ベクトル４７９が生成される。

【００８６】

次いで、直線エッジ４７３の座標４７４（Ｘｓ，Ｙｓ）及び４７２（Ｘｅ，Ｙｅ）、ならびに左側法線ベクトル４７９（Ｘｎ，Ｙｎ）を使用して、左側ストロークエッジ及び右側ストロークエッジが計算される。

【００８７】

左側ストロークエッジ４７７において、始点座標（Ｘｓ_ｌｅｆｔ，Ｙｓ_ｌｅｆｔ）及び終点座標（Ｘｅ_ｌｅｆｔ、Ｙｅ_ｌｅｆｔ）は以下の通り計算される。

【００８８】

Ｘｓ_ｌｅｆｔ＝Ｘｓ＋Ｘｎ　　Ｙｓ_ｌｅｆｔ＝Ｙｓ＋Ｙｎ

Ｘｅ_ｌｅｆｔ＝Ｘｅ＋Ｘｎ　　Ｙｅ_ｌｅｆｔ＝Ｙｅ＋Ｙｎ

右側ストロークエッジ４７８は、現在のエッジの座標から法線ベクトルを差し引くことによって生成される。

【００８９】

Ｘｓ_ｒｉｇｈｔ＝Ｘｓ－Ｘｎ　　Ｙｓ_ｒｉｇｈｔ＝Ｙｓ－Ｙｎ

Ｘｅ_ｒｉｇｈｔ＝Ｘｅ－Ｘｎ　　Ｙｅ_ｒｉｇｈｔ＝Ｙｅ－Ｙｎ

３．４．２　曲線エッジのストローキング

図４９（ｃ）は、ストロークすべき曲線エッジ４８３を示している。曲線エッジ（２次ベジエ曲線又は２次多項式フラグメント（ＱＰＦ））は、始点４８０、制御点４８１及び終点４８２によって表される。

【００９０】

図４９（ｄ）は、このエッジをストロークするために生成しなければならない２つの左側法線ベクトル４８７及び４８８を示す。２つの左側法線ベクトルはいずれも、ストロークのペン幅の半分の長さをもつ。第１の左側法線ベクトル４８７は、始点４８０から制御点４８１に至るベクトルに対して反時計回りに９０度である。第２の左側法線ベクトル４８８は、制御点４８１から終点４８２に達するベクトルに対して反時計回りに９０度である。

【００９１】

図４９（ｅ）を参照すると、始点４９３、制御点４９４及び終点４９５をもつ曲線の左側ストロークエッジが生成される。左側ストロークエッジの始点４９３（Ｘｓ_ｌｅｆｔ，Ｙｓ_ｌｅｆｔ）は、曲線エッジの始点（Ｘｓ，Ｙｓ）を第１の左側法線ベクトル４８７（Ｘｎ1，Ｙｎ1

）だけ平行移動４８０させることによって計算される。

【００９２】

Ｘs_left＝Ｘs＋Ｘn1　　Ｙs_left＝Ｙs＋Ｙn1

左側ストロークエッジの終点（Ｘe_left，Ｙe_left）は、曲線エッジの終点（Ｘe，Ｙe）を第２の左側法線ベクトル４８８（Ｘn2、Ｙn2）だけ平行移動４８２させることによって計算される。

【００９３】

Ｘe_left＝Ｘe＋Ｘn2　　Ｙe_left＝Ｙe＋Ｙn2

左側ストロークエッジの制御点４９４は、２つの直線の交点を使用して計算される。第１の直線は、左側ストロークエッジの始点４９３を通り、曲線エッジの始点４８０から曲線エッジの制御点４８１に至る直線と平行である。第２の直線は、左側ストロークエッジの終点４９５を通り、曲線エッジの終点４８２から曲線エッジの制御点４８１に至る直線と平行である。

【００９４】

始点５０３、制御点５０４及び終点５０５を示す図４９（ｆ）の例で示されるように、曲線の右側ストロークエッジも同様に生成される。右側ストロークエッジの始点座標（Ｘs_right，Ｙs_right）及び終点座標（Ｘe_right，Ｙe_right）は以下の通り計算される。

【００９５】

Ｘs_right＝Ｘs－Ｘn1　　Ｙs_right＝Ｙs－Ｙn1

Ｘe_right＝Ｘe－Ｘn2　　Ｙe_right＝Ｙe－Ｙn2

右側ストロークエッジの制御点５０４は、２つの直線の交点を使用して計算される。第１の直線は、右側ストロークエッジの始点５０３を通り、曲線エッジの始点４８０から曲線エッジの制御点４８１に達する直線と平行である。第２の直線は、右側ストロークエッジの終点５０５を通り、曲線エッジの終点４８２から曲線エッジの制御点４８１に達する直線と平行である。

【００９６】

図４９（ｇ）は、様々な制御点に関して描写された、得られた左側ストロークエッジ５１４及び右側ストロークエッジ５１５を示す。

【００９７】

上述の技法は曲線のストローキングの近似であり、図４９（ｈ）に示される曲線５２０などの狭い角度をもつ曲線には不向きである。曲線の始点５１７、制御点５１８及び終点５１９で形成される三角形において、制御角は、始点５１７と終点５１９とを結ぶ直線の対角（即ち、向かい合った）として定義される。制御角が１２０度以下である場合、曲線は狭い角度をもつと考えることができる。便利なテストは、２つの左側法線ベクトル５２１（Ｘn1，Ｙn1）及び５２２（Ｘn2，Ｙn2）の内積の絶対値を使用するものである。

［数１］

If（Ｘn1Ｘn2＋Ｙn1Ｙn2）^{2}＜1/2（Ｘn1^2＋Ｙn1^2）＊（Ｘn2^2＋Ｙn2^2）

ならば、曲線は狭い角度をもつ。

【００９８】

始点、制御点及び終点で表される２次ベジエ曲線を２等分する複数の方法が、当技術分野では周知である。狭い角度をもつと判断される曲線エッジは、まず２つの互いに等しい曲線エッジに２等分され、次いで再び２分割されて、結果的に合計４つの曲線エッジとなる。次いで外側の２つの新規エッジ（元の狭い角度に隣接していないもの）が、先に述べた処理を使用してストローキングされる。内側の２つの新規エッジが、当技術分野で周知の技法を使用して、近似直線エッジのパスへとベクトル化される。次いで、このパスがセクション３．４．１に記載される処理を使用してストローキングされる。

【００９９】

３．４．３　継目のストローキング

パスのエッジが接合する領域をストロークするには、追加の曲線を生成する必要がある。継目は、入りエッジと出エッジとをもつと定義される。図４５（ａ）は、２つの直線エ

ッジが継目４４４を形成しているところを示しているが、ここでエッジ４４３は、継目がその終点を形成するので入りエッジであり、エッジ４４５は、継目がその始点を形成するので出エッジである。左側ストロークエッジ４４６，４４７及び右側ストロークエッジ４４８，４４９は、先に述べた処理を使用して生成されている。入りエッジ４６２の終点及び出エッジ４６３の始点に対する左法線ベクトルは決定されているはずである。まず、左側を検討すると、入りエッジの終点４５３と出エッジの始点４５４とを連結する継目に対する曲線の輪郭線を生成する必要があるかどうかを判断するテストが行われる。ストロークすべき入りエッジの終点について成分（Ｘn1，Ｙn1）をもつ左側法線ベクトル４６２と、ストロークすべき出エッジの始点について成分（Ｘn2，Ｙn2）をもつ左側法線ベクトル４６３とが既知であると仮定すると、曲線の輪郭線を生成する必要があるかどうかを判断するために以下のテストを使用することができる。

【０１００】

Ｙn1Ｘn2＜Ｙn2Ｘn1であるならば、ストロークの左側エッジを連結するために曲線エッジが使用され、ストロークの右側エッジを連結するために直線エッジが使用される。

【０１０１】

同様に、Ｙn1Ｘn2＞Ｙn2Ｘn1であるならば、ストロークの右側エッジを連結するために曲線エッジが使用され、ストロークの左側エッジを連結するために直線エッジが使用される。

【０１０２】

Ｙn1Ｘn2＝Ｙn2Ｘn1であるならば、継目を形成するエッジは、継目において同一又は反対の方向をもつ。このような場合における処理が、以下のセクション３．４．４に示されている。

【０１０３】

図４５（ａ）に示される例では、上述のテストは、継目の左側において曲線エッジを生成する必要があることを示すはずである。この曲線エッジは、以下の性質をもつ真の円弧となる。即ち

－　孤の始点及び終点は、それぞれ点４５３及び４５４であり、

－　孤の半径は、左側ベクトル４６２の長さであり、

－　孤の中心は、オリジナルパスポイント４４４であり、

－　孤は、始点から終点へと時計回りに進む（継目の右側が処理されている場合、孤は反時計回りに進む）。

【０１０４】

これらの値は、楕円弧エッジを生成するために使用され、その後の処理については後で説明される。

【０１０５】

図４５（ａ）に示される例で話を続けると、上述のテストは、継目の右側において、第１の右側エッジ４５１の終点を第２の右側エッジ４５０の始点と連結する直線エッジを生成する必要があることを示すはずである。

【０１０６】

図４５（ｂ）は、互いに異なるストロークパスの同じ側にあるエッジ４５５及び４５６が交差しない状況を示す。対応する頂点４６０及び４６１から延び、エッジ４５５及び４５６に関連する制御点又は中心の周りで曲がった新規エッジを生成することができる。図４５（ｂ）はまた、交差しない対応する向かい合ったエッジ４５７ｂ及び４５７ａを示す。パスを閉じるために、新規エッジが頂点４５８及び４５９の間に挿入される。これは、ストロークフィルに対する効果がゼロであるため、直線エッジである。

【０１０７】

３．４．４　等しい又は向かい合うエッジの継目のストローキング

上述の継目のストローキングの技法についての説明では、継目を形成する２つの原エッジがいつ等しい又は反対の方向をもつかを示すためのテストについて説明した。即ち、（Ｘn1，Ｙn1）がストロークすべき入りエッジの終点における左側法線ベクトルで、（Ｘn2

，Ｙn2）がストロークすべき出エッジの始点における左側法線ベクトルであるとき、
　Ｙn1Ｘn2＝Ｙn2Ｘn1
となる。
【０１０８】
更に、Ｘn2＝Ｘn1であるならば、左側法線ベクトルは等しく、従って入りエッジと出エッジとは継目において同じ方向をもつ。継目をストローキングするための追加のエッジは不要である。
【０１０９】
Ｙn1Ｘn2＝Ｙn2Ｘn1であるがＸn2≠Ｘn1であるならば、入りエッジと出エッジとは継目において反対の方向をもち、従って以下に説明するように、継目のためのエンドキャップを生成しなければならない。
【０１１０】
３．４．５　パス終点のエンドキャップの生成
エンドキャップは、ストロークすべき閉じていないパスの端点のために生成される。エンドキャップはまた、（上述したように）入りエッジと出エッジとが反対の方向である場合に、エッジ間の継目のために生成しなければならない。
【０１１１】
処理は、楕円弧を使用して端点の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）に示されているが、図１８（ａ）はストロークすべきパスの元のエッジ１９２が点１９３で終端することを示している。左側ストロークエッジ１９５及び右側ストロークエッジ２００は、それぞれ終点１９６及び１９８で終わっていることが示されている。有限エッジの終点における成分（Ｘn，Ｙn）をもつ左側法線ベクトル１９４が示されている。
【０１１２】
エンドキャップを生成するため、以下の特性をもつ楕円弧が配置される。
【０１１３】
－　孤は、左側終点１９６から始まり、右側終点１９８で終わり、
－　孤の中心は、オリジナルパスポイント１９３であり、
－　孤の半径は、左側法線ベクトル１９４の長さであり、
－　孤は、円の周りを時計回りに進む。
【０１１４】
有効な左側ｚレベル参照及び／又は有効な右側ｚレベル参照を（有効なストロークｚレベル参照に加えて）もつパスをストローキングする場合、それはストロークされたパスがフィルすべき囲まれた領域をも形成することを示している。このようなパスにおいては、任意のエンドキャップに２つの直線エッジを追加する必要がある。
【０１１５】
この場合、時計回りに９０度回転させた有限エッジの終点における左側法線ブクトルに等しい、成分（Ｘn_knee，Ｙn_knee）をもつ新しい左側法線ベクトル１９９が生成される。次いで、端点１９３（Ｘj，Ｙj）の座標及び成分（Ｘn_knee，Ｙn_knee）をもつ新しい左側法線ベクトル１９９を使用して、屈曲点座標１９７（Ｘknee，Ｙknee）が生成される。即ち、
　Ｘknee＝Ｘj＋Ｘn_knee　　Ｙknee＝Ｙj＋Ｙn_knee
図１８（ｂ）は、いずれも屈曲点１９７から始まり、ストロークすべき元のエッジの端点で終わる追加の２つのエッジ２０７ａ及び２０７ｂを示す。第１の追加エッジ２０７ａは、追加の左側ストロークエッジであり、第２の追加エッジ２０７ｂは、追加の右側ストロークエッジである。左側ストロークエッジ及び右側ストロークエッジに、どのようにｚ値参照が割り当てられるかについての説明は下記で行う。
【０１１６】
３．４．６　ストロークされたエッジとストロークされていないエッジとの間のエンドキャップ

パス内で、ストロークフィルを参照するエッジの後にストロークフィルを参照しないエッジが続く場合も、エンドキャップが生成される。図２８（ａ）は、部分的にストロークされた三角形の形状を示す。頂点２７５は、ストロークされたエッジ２７６がストロークされていないエッジ２７７と交差するところである。頂点２７５では、エンドキャップが生成される。

【０１１７】

このエンドキャップは、前のセクションに示されるのと同じ方法で作成される。即ち、パスの終わりで生成されるエンドキャップに使用されるのと同じ方法と、以下に記載される追加のステップとが用いられる。この方法を適用した結果は、図２８（ｂ）に詳しく示されている。

【０１１８】

エッジ２７８によって参照される左側フィルである、形状の内部フィルを考慮されたい。これから、小さなエンドキャップエッジ２８０を追加することで、内部フィルが完全に閉じた１つのエッジセットによって囲まれることが理解できよう。エッジ２７９，２８０，２８２，２７８は、継目の領域において図形のフィルを囲む１つのエッジセットである。図示されていない追加のエッジによって、継目ではない領域にあるフィルが囲まれる。形状をフィルする方法は、特にエッジによって囲まれた領域をフィルするために屈曲規則及び屈曲カウントが使用される場合には、閉じられる形状に依存することが、当技術分野において周知である。

【０１１９】

エッジ２７８は、図２８（ｂ）に示されるエッジの中で、ストローキングの方法で生成されなかった唯一のエッジである。このエッジは、入りエッジセットから取り出され、出エッジセットに加えられる。

【０１２０】

３．４．７　不透明ストロークへのｚレベルの割当て

順序付けられた１つの座標セットが入力として提供され、その順序付けられた１つの座標セットがストロークパスを形成することになる場合、左側ｚレベル参照及び右側ｚレベル参照に加えて、有効なストロークｚレベル参照及びペン幅が入力として必要となる。ストロークに使用されるｚレベルのフォーマットと、パスをフィルするのに使用されるｚレベルのフォーマットとの間に違いはない。順序付けられた１つの座標セットが処理されて順序付けられた１つのエッジセットになると、それらは先に述べた技法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、１つの左側ストロークエッジセットと１つの右側ストロークエッジセットとを作成する。モジュール６１６が入力として左側ｚレベル、右側ｚレベル及びストロークｚレベルを受け取り、ストロークｚレベルが不透明であると判断される場合、作成された左側ストロークエッジに以下のものが割り当てられる。

【０１２１】

－　入力として受領した左側ｚレベルを参照する左側ｚレベル参照、及び
－　入力として受領したストロークｚレベルを参照する右側ｚレベル参照

また、作成された右側ストロークエッジには以下が割り当てられる。

【０１２２】

－　入力として受領したストロークｚレベルを参照する左側ｚレベル参照、及び
－　入力として受領した右側ｚレベルを参照する右側ｚレベル参照

このセクションでは、左側又は右側ストロークエッジの参照とは、エッジ方向に見て、ストロークエッジの外側境界上のエッジの参照である。この点に関連して、このエッジがディスプレイの左上から右下に向いているとき、エッジの右側に左側ストロークエッジが現れる。

【０１２３】

このｚレベル割当てが使用されるとき、ストロークパスの元のエッジは廃棄され、従って作成された左側ストロークエッジ及び右側ストロークエッジで完全に置き換えられる。

【０１２４】

一例が、図５１（ａ）～（ｅ）に示される。図５１（ａ）は、上に述べたストローキング方法における、閉じたパスを画定する１つの入力エッジセットを示す。エッジ５３３，５３２及び５３４は、いずれも左側ｚレベル５３０、右側ｚレベル５３１及びストロークｚレベル５２９に関連付けられている。不透明のストロークの場合、入力エッジ５３３に対するストローキング方法のエッジ出力が、それ自体のｚレベル割当てとともに、図５１（ｄ）に示されている。図５１（ａ）に示されるパスをストローキングすることの視覚的結果を、図５１（ｂ）に示す。

【０１２５】

作成された左側ストロークエッジ５３５は、入力として提供された左側ｚレベル５３０を参照する左側ｚレベル参照５３８、及び入力として提供されたストロークｚレベル５２９を参照する右側ｚレベル参照５３９をもつことに留意されたい。また、作成された右側ストロークエッジ５３６は、入力として提供されたストロークｚレベル５２９を参照する左側ｚレベル参照５４０、及び入力として提供された右側ｚレベル５３１を参照する右側ｚレベル参照５４１をもつ。元のストロークエッジ５３３は廃棄される。

【０１２６】

３．４．８　透明ストロークへのｚレベルの割当て

以下に、上述のｚレベル割当ての代替方法を記載する。この割当ては、いくらかの透明度をもつストロークｚレベルをもつ順序付けられた１つのエッジセットのストローキングに特に適している。このｚレベル割当てにおいて、作成された左側ストロークエッジには

－　ヌルの左側ｚレベル参照、及び

－　入力として受領したストロークｚレベルを参照する右側ｚレベル参照

が割り当てられ、作成された右側ストロークエッジには

－　入力として受領したストロークｚレベルを参照する左側ｚレベル参照、及び

－　ヌルの右側ｚレベル参照

が割り当てられる。

【０１２７】

このｚレベル割当てが使用されるときは、元のエッジは廃棄されない。即ち、ストローキング処理の出力は元の１つのエッジセット、作成された１つの左側ストロークエッジセット及び作成された１つの右側ストロークエッジセットの和集合である。元のエッジには、入力された左側ｚレベルを参照として提供された左側ｚレベル参照、及び入力された右側ｚレベルを参照として提供された右側ｚレベル参照が割り当てられる。

【０１２８】

図５１（ａ）は、上述のストローキング方法における１つの入力エッジセットを示す。エッジ５３３，５３２及び５３４は、いずれも左側ｚレベル５３０、右側ｚレベル５３１及びストロークｚレベル５２９に関連付けられている。透明なストロークの場合、入力エッジ５３３に対するストローキング方法のエッジ出力が、それ自体のｚレベル割当てとともに、図５１（ｅ）に示されている。図５１（ａ）に示されるパスをストローキングした視覚的結果を図５１（ｃ）に示す。

【０１２９】

作成された左側ストロークエッジ５４２は、ヌルの左側ｚレベル参照５４５、及び入力として提供されたストロークｚレベル５２９を参照する右側ｚレベル参照５４６をもつことに留意されたい。元のエッジ５３３は廃棄されず、入力として提供された左側ｚレベル５３０を参照する左側ｚレベル参照５５０、及び入力として提供された右側ｚレベル５３１を参照する右側ｚレベル参照５４９をもつ。また、作成された右側ストロークエッジ５３６は、入力として提供されたストロークｚレベル５２９を参照する左側ｚレベル参照５４７、及びヌルの右側ｚレベル参照５４８をもつ。

【０１３０】

図６３は、望ましくはストローキングされ、いくらかの有向エッジ６３０２，６３０４，６３０６及び６３０８によって形成されたパス６３００を示す。示されるように、方向付けられたエッジは、頂点６３１０，６３１２，６３１４，６３１６及び６３１８から延

び、これらの頂点で接合する。

【０１３１】

この例では、パス６３００は、方向付けられた各エッジ６３０２～６３０６に関連付けられたストローク幅に従ってストローキングされることが望ましい。この点について、方向付けられたエッジ６３０４及び６３０６は同じストローク幅及び色をもち、そのため、これらの２つのエッジはあいまって、ストロークされるべき元のパス６３００の離散部分を画定する。

【０１３２】

先に図１８（ａ）及び１８（ｂ）に関連して示したように、エッジの終点をストローキングする際、アーティファクトが生ずる可能性がある。具体的には、図６３に示されるように、互いに異なるストローク幅をもつエッジが（例えば、頂点６３１２及び６３１６で）交差する場合には、エッジ情報から正確な合成が実施できるように、全てのエッジについて正確なストローキングが行われることが不可欠である。この点について、ストローク幅が互いに異なると、一方のストロークが他方のストロークの上に合成される恐れがあり、その際、透明なストロークが使用される場合には非常に重大なエラーをもたらすことがある。更に、透明ストロークを使用する際には、重要度はそれより低いものの、それでも重大なエラーが起こる可能性がある。確実にパス６３００が正確にストローキングされるには、パス６３００の離散エッジ部分によって形成される個別のストロークパスが確実に正確に再生されるようにする必要がある。

【０１３３】

これは、まず、ストローグ幅が変化する、元のストロークパス６３００の頂点を識別することによって実施される。図６３から、ストローグ幅は頂点６３１２及び６３１６で変化することが分かる。他の全ての頂点では、ストローク幅は一定であり、作成されるべき対応するストロークパスのストローク幅に対応している。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、変化しない他の頂点（例えば、頂点６３１４）が正確にストロークされることが可能になる。頂点が適切に識別されると、次いで元のパス６３００の各離散部分毎に、別々のストロークパスを作成することができる。エッジ６３０２で画定される第１の離散部分において、ストロークパス６３２０及び６３２２は、それぞれ頂点６３１２から発し、方向付けられたエッジ６３０２の拡張部分６３３８上にあり、頂点６３１０から延びる紙面上方に向かっている。頂点６３１０の周りにおけるパス６３２０及び６３２２のストローキングは、先に図１８（ａ）で示した方法で実施してもよい。頂点６３１２の周りにおけるパス６３２０及び６３２２のストローキングは、先に図１８（ｂ）に関連して説明した方法で実施される。図６３では、頂点６３１２から延びるパス６３２０及び６３２２（図１８（ｂ）に示すエッジ２０７ａ及び２０７ｂと比較されたい）を誇張してあることに留意されたい。とりわけ、ストロークパス６３２０及び６３２２は、あいまって元のパス６３００の（エッジ６３０２で形成された）対応する離散部分を包み込んでいる。

【０１３４】

エッジ６３０４及び６３０６に関連付けられたストロークパスについても、同様の手法が実施される。この場合、頂点６３１２及び６３１６によって画定されるそれぞれのパスの端部には、先に図１８（ｂ）で示した構成に対応する構成が組み込まれている。エッジ６３０８に関連付けられたストロークパス６３２８及び６３３０は、エッジ６３０２のストロークパスを作成したのと同じ方法で作成される。重要なことには、どの例においても、ストロークパスが、ストローク幅が変化するパス上の点に位置すると識別された頂点を通過する必要がある。

【０１３５】

図６３の構成の拡張例が、閉じたパスを示す図５１（ｂ）又は５１（ｃ）を参照すれば理解されよう。図６３の構成では、パスの内側が先に図５１（ａ）～（ｅ）に関して説明した負の屈曲規則及びフィル規則を使用して正確にフィルできるように、離散部分を組み込んだ閉じたパスがストローキングされることが可能である。このように、先に述べた、

元のパスのエッジを左側及び右側フィルをもつストロークパスを画定するエッジで置き換える方法は、閉じたパスの内部、そのストローク部分の結果としてのフィルレベル及び内部を画定するのに使用することができる。

【０１３６】

３．４．９　ストローキングプリミティブの変換

ストローキングがレンダリング空間で実施される場合、ストローキングモジュールによって作成された線分、ベジエ曲線及び孤を変換する必要はない。しかし、ストローキングがオブジェクト空間で実施される場合、これらのプリミティブは、形状の元のエッジと同じ変換を受けなければならない。

【０１３７】

ストローク線及びベジエ線は、セクション３．２で論じたような、形状の線及びベジエ曲線の場合と同様に変換される。一方、ストロークのエンドキャップ及び継目は、円弧にアフィン変換を適用すると一般化された楕円弧が得られることから、別の手順を経なければならない。

【０１３８】

図１９に、一般化楕円弧を示す。下位の楕円２１１は、長軸２０８、短軸２０９、中心点２１５及び回転角２１０で表すことができる。この楕円上に横たわる楕円弧フラグメント２１２は、更に始点Ｙ座標２１３、終点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）を提供することによって、完全に表すことができる。楕円フラグメント２１２のケースでは、方向は反時計回りである。

【０１３９】

下位の円及び変換から楕円座標を取り出すには、いくつかの手法がある。１つの手法は、以下の式の２つの解ｍｏｄπを見つけることである。

【０１４０】

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

ここで、

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

は変換マトリックスの非変換部分である。

【０１４１】

次いで、正規化された（楕円の中心に対し、単位円に対する）長軸及び短軸ベクトルは、θの２つの解について

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

で与えられる。これらの正規化軸（又はその逆向きの軸）Ａ’のうちの１つは、第１象限にあるはずである。この軸ベクトルの長さに変換前の円の半径を乗じることで、変換後の楕円の長軸の長さが与えられる。この軸ベクトルの水平からの角度により楕円の回転が与えられる。他方の軸ベクトルの長さを変換前の円の半径で乗じることで、変換後の楕円の短軸の長さが与えられる。

【０１４２】

楕円の中心は、円の中心にこのユーザ供給の変換を適用することによって求められる。楕円弧の始点及び終点座標は、この同じ変換を元の円の始点及び終点座標に適用することによって求められる。

【０１４３】

ユーザ供給変換は、１つの次元で反転を行っている可能性がある。そうであるならば、孤フラグメントの方向を変えなければならない。変換が反転しているかどうかを判定するには、ベクトル

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

｜　｜　ａｎｄ｜　｜
｜０｜　　｜１｜

にΓを適用し、求められたベクトルの水平からの回転の角度を求めることで十分である。２つのベクトルは、１８０度よりも小さい角をなす。

【０１４４】

｜１｜
Γ｜　｜
｜０｜

がこの角度の反時計回り側にある場合、この変換は反転する変換である。そうでない場合、この変換は反転しない。

【０１４５】

３．５　フィルタリング

エッジを始点座標（Ｘｓ，Ｙｓ）及び終点座標（Ｘｅ，Ｙｅ）、ならびに曲線の場合のみ追加の制御座標（Ｘｃ，Ｙｃ）で表す座標を作成した後、次の処理は明らかに出力フレームに影響を与えないエッジを廃棄することであり、これは図４６に示されるフィルタリング処理４７０である。処理のこの段階では、エッジの全ての座標は（既に変換されているために）「レンダリング空間」内にあり、従ってディスプレイのバウンディング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、高さｈをもち、フレームの左上が（０、０）、フレームの右下が（ｗ－１、ｈ－１）でそれぞれ表されるとすると、

（Ｘｓ≧ｗ）かつ（Ｘｅ≧ｗ）／／右方にずれている
又は
（Ｙｓ＜０）かつ（Ｙｅ＜０）／／上方にずれている
又は
（Ｙｓ≧ｈ）かつ（Ｙｅ≧ｈ）／／下方にずれている

である場合、直線エッジを安全にフィルタリングすることができる。

【０１４６】

フィルタリングを行う場合、グリフは、そのグリフの左上隅から右下隅に延びる直線エッジであるかのように扱われる。

【０１４７】

曲線（追加の制御座標をもつ）のための同様のテストは

（Ｘｓ≧ｗ）かつ（Ｘｃ≧ｗ）かつ（Ｘｅ≧ｗ）　／／右方にずれている
又は
（Ｙｓ＜０）かつ（Ｙｃ＜０）かつ（Ｙｅ＜０）　／／上方にずれている
又は
（Ｙｓ≧ｈ）かつ（Ｙｃ≧ｈ）かつ（Ｙｅ≧ｈ）　／／下方にずれている

である。

【０１４８】

水平である直線エッジ（即ち、Ｙｓ＝Ｙｅ）もフィルタリングされる。

【０１４９】

完全にディスプレイの左方にずれているエッジは、表示出力に影響するため、廃棄されないことに留意されたい。

【０１５０】

３．６　エッジ追跡パラメータの生成

図４６において、次の処理４６９の目的は、エッジをある修正表現に変換することにあり、この処理は追跡パラメータの計算を含む可能性がある。この修正表現は、後続モジュールによる処理により適しており、少なくとも

－　始点座標（Ｘ，Ｙ）、
－　左側ｚレベル及び／又は右側ｚレベルの参照、
－　終点Ｙ座標、及び
－　追跡パラメータ

を含む。

【０１５１】

本明細書では、「追跡パラメータ」という用語は、別段の意図が示されていない限り、エッジのＸ座標がＹの単位増分に関連して増分的に計算できるようにする、１つ又は複数のパラメータを表すために使用される。例えば、直線は始点座標（Ｘｓ，Ｙｓ）、終点Ｙ座標Ｙｅ、及びＹ座標における単位ステップ変更（即ち、勾配）に対応するＸ座標の変更を示すデルタＸ項によって表すことができる。デルタＸ項は、エッジの単一の追跡パラメータとなるはずである。

【０１５２】

ＴＣＩＥシステム６９９は常に左上から右下にレンダリングを行うため、エッジ（直線又は曲線）は、始点Ｙ座標よりも大きい終点Ｙ座標をもつように指定する必要がある。これは、始点及び終点Ｙ座標を比較し、適切な場合は始点座標と終点座標を交換することで、確実にすることができる。始点座標と終点座標が交換される場合、次いで左側ｚレベル参照と右側ｚレベル参照を交換しなければならない。

【０１５３】

３．６．１　直線エッジ追跡パラメータの生成

直線が、デルタＸ項を追跡パラメータとして使用することにより記述できることは、既に示されている。しかし、デルタＸはしばしば、小数点データを含むことが必要となる。デルタＸを浮動小数点又は固定小数点フォーマットを使用して表す代わりに使用される、より正確な表現方法は、勾配の整数部と小数部を記憶することである。

【０１５４】

ソフトウェア実装では、この処理をＣ言語で実装して、勾配の２つの部分を計算することができる。

【０１５５】

Ｇｒａｄ＝（Ｘｅ－Ｘｓ）／（Ｙｅ－Ｙｓ）

Ｉｅｎｔ＝（Ｘｅ－Ｘｓ）％（Ｙｅ－Ｙｓ）

ここで（Ｘｓ，Ｙｓ）はエッジの始点座標であり、（Ｘｅ，Ｙｅ）はエッジの終点座標である。

【０１５６】

エッジが適切なフォーマットで正確に表されるために、上述のＧｒａｄ及びＩｅｎｔ項が、始点座標（Ｘｓ，Ｙｓ）、終点Ｙ座標Ｙｅ、左側及び右側ｚレベル参照と共に提供され、それに加えてleft flag（ブール値）が提供される。ただし、

Leftflag＝ＴＲＵＥ　（Ｘｅ＜Ｘｓ）の場合

＝ＦＡＬＳＥ　（Ｘｅ≧Ｘｓ）の場合

である。

【０１５７】

３．６．２　２次ベジエ曲線追跡パラメータの生成

先に述べたように、図４６のステップ４６５は、各曲線毎に

－　始点座標、

－　制御座標、及び

－　終点座標

を生成する。

【０１５８】

曲線の座標は既に（ステップ４６８における、対応する順序付けられた１つの座標セットの処理中に）変換されているため、曲線の座標は「レンダリング空間」内の位置に対応する。図７は、原点（アニメーションのフレームの左上隅６１を表す）、Ｘ及びＹ軸（それぞれ６３及び６２）で示されるレンダリング空間内で、始点座標５８、制御座標５９及び終点座標６０で表されるベジエ曲線の例を示す。以下に、図４１に関連して、始点座標５８（Ｘｓ，Ｙｓ）、制御座標５９（Ｘｃ，Ｙｃ）及び終点座標６０（Ｘｅ，Ｙｅ）から２次ベジエ曲線の追跡パラメータを生成する処理を説明する。

【０１５９】
図４１で、第１のステップ４０３は、始点、制御及び終点座標が同一直線上にあるケースかどうかを判定する。そうである場合、制御座標は無視され、ステップ４０４で、エッジの追跡パラメータが、直線エッジについて説明したように計算される。
【０１６０】
次のステップ４０５で、２次ベジエ曲線がＹに関して単調（即ち、各Ｙ座標に対し、曲線が交差する対応するＸ座標が１つだけある）かどうかがチェックされる。例えば、図７のベジエ曲線は、明らかにＹに関して非単調である。このような非単調の２次ベジエ曲線は、Ｙに関して単調である２つの等しい曲線エッジを使用して表す必要がある。始点（Ｘs，Ｙs）、制御点（Ｘc，Ｙc）及び終点（Ｘe，Ｙe）で画定される２次ベジエ曲線が２つの曲線エッジを必要とするかどうかを判定するには、以下の論理を使用することができる。
【０１６１】
オペランドが正であるときにブール値ＴＲＵＥを返す（オペランドが負のときにブール値ＦＡＬＳＥを返す）関数をsign（）とし、
（ｓｉｇｎ（Ｙs－Ｙc）＝ｓｉｇｎ（Ｙs－２Ｙc＋Ｙe））
かつ
（｜Ｙs－Ｙc｜＜｜Ｙs－２Ｙc＋Ｙe｜）
が成り立つならば、ベジエは２つの曲線エッジを必要とする。
【０１６２】
２つの曲線エッジが必要と判定された場合、元の非単調ベジエ上のある点（Ｘsplit，Ｙsplit）は、以下の公式を使用して計算される。
【０１６３】
Ｘsplit＝｛Ｘs（Ｙe－Ｙc）^{2}＋２Ｘc（Ｙs－Ｙc）（Ｙe－Ｙc）＋Ｘe（Ｙs－Ｙc）^{2}｝／（Ｙs－２Ｙc＋Ｙe）^{2}
Ｙsplit＝（ＹsＹe－Ｙc^2）／（Ｙs－２Ｙc＋Ｙe）
次いで、ステップ４０９で、元の非単調２次ベジエを表す２つの単調曲線エッジは、以下の各座標を使用して作成される。
【０１６４】
第１の曲線エッジ：
始点座標＝元の非単調２次ベジエの始点座標（Ｘs，Ｙs）、及び
終点座標＝（Ｘsplit，Ｙsplit）
第２の曲線エッジ：
始点座標＝（Ｘsplit，Ｙsplit）、及び
終点座標＝元の非単調２次ベジエの終点座標（Ｘe，Ｙe）
エッジ（直線又は曲線）は、始点Ｙ座標よりも大きい終点Ｙ座標をもつように指定されることが必要であることは、先に述べた通りである。これは、ステップ４０６で各曲線の始点及び終点Ｙ座標を比較し、該当する場合はステップ４１０で始点及び終点座標を交換することによって保証することができる。始点及び終点座標が交換される場合、左側及び右側ｚレベル参照も交換しなければならない。
【０１６５】
次のステップ４０７は、曲線の一部がレンダリングされるフレームよりも上にある（即ち、０よりも小さい始点Ｙ座標Ｙsをもつ、ただしＹ＝０はフレームの上端topを表す）場合、曲線を「クリップ」するように動作する。最上位座標Ｙtopは、次式によって求められる。
【０１６６】
Ｙtop＝ｍａｘ（０，Ｙs）
ただし、ｍａｘ（ａ，ｂ）は、２つのオペランドａ又はｂのうち大きい方を返す関数である。
【０１６７】

次のテストでは、各曲線毎に、その曲線を２次多項式フラグメント（ＱＰＦ）として表す必要があるかどうかを判定する。これは、後に続く２次方程式を解く際にゼロによる除算を避けるために必要である。この状況にあるかどうかを判断するための、始点、終点及び制御点に基づく手短なテストが、ステップ４１３で、以下に示すように行われる。

【０１６８】

$$Ys+Ye=2Yc$$

は曲線がＱＰＦであることを示す。

【０１６９】

曲線がＱＰＦであると判定された場合、ステップ４１１で３つの追跡パラメータＡ，Ｃ及びＤが生成される。これらは、曲線の始点、終点及び制御点、ならびに最上位座標Ｙtopから、以下のように導かれる。

【０１７０】

中間値が与えられた場合

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

$$Bi=(4XcYc-2XsYe-2XeYs)\}/(Ye-Ys)^2$$

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

ここでＡ、Ｃ及びＤは、以下の通り導かれる。

【０１７１】

$$A=Ai+CiYtop+DiYtop^2$$

$$C=Ci+Di(2Ytop+1)$$

$$D=2Di$$

ステップ４１３で、曲線がＱＰＦでないと判定された場合、ステップ４０８で、追跡パラメータＡ，Ｂ，Ｃ及びＤが生成される。Ａ，Ｂ，Ｃ及びＤは、曲線の始点、終点及び制御点、ならびに最上位座標Ｙtopから、以下の計算で導かれる。

【０１７２】

$$Xg=Xs-Xe$$

$$Yg=Ys-Ye$$

$$Xh=Xc-Xe$$

$$Yh=Yc-Ye$$

$$Yterm=Yg-2Yh$$

$$mix=XgYh-XhYg$$

とするとき、

$$B=(Xg-2Xh)/Yterm^2$$

$$A=(2mix\times Yh)/Yterm^2+B(Ytop-Ye)+Xe$$

$$D=(Yterm\times mix^2)/Yterm^4)$$

$$C=(mix\times Yh^2)/Yterm^4+D(Ytop-Ye)$$

となる。

【０１７３】

３．６．３　符号の判定

パラメータＡ，Ｂ，Ｃ，Ｄは、後の処理で、以下の形式の式を使用して２次ベジエ曲線のＸ座標を追跡するために使用される。

【０１７４】

$$x=A\pm2\sqrt{C}$$

追跡パラメータを準備するための最後のステップ、即ち図４１のステップ４１２は、平方根の項をＡに加算すべきか、或はＡから減算すべきかを判定するステップである。Ａ，Ｘs，Ｙs，Ｘe，Ｙe及びＤに基づいて、以下の擬似コードを使用することができる。ここで、結果「ｓｉｇｎ＝＋１」は平方根の項の加算を示し、「ｓｉｇｎ＝－１」は平方根の項の減算を示す。

【０１７５】

```
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　楕円弧追跡パラメータの生成

（図４６に関連して先に説明した）ステップ４６６は、各孤毎に

－　楕円の長軸及び短軸半径「ａ」及び「ｂ」、

－　水平からの（半径ａの）楕円角「θ」、

－　楕円の中心（ｘ，ｙ）、

－　孤の始点座標（ｘｓ，ｙｓ）及び終点座標（ｘｅ，ｙｅ）、ならびに

－　左側及び右側ｚレベル参照

を生成する。

【０１７６】

楕円は、Ｙに関してのみスキューされ、Ｘに関してのみ拡大縮小された円と記述することができる。従って、楕円追跡を簡略化するために、ＴＣＩＥシステム６９９は全ての楕円を変換済みの円として扱う。

【０１７７】

楕円エッジを変換する際に、追跡パラメータの計算４６９が最初に行わなければならないステップは、楕円の基礎となる円を決定することである。図２０は、この処理を示している。楕円２１６は、スキューファクタによって、レクティリニア型楕円２１７に関連付けられている。このスキューの大きさは、線２１８の勾配であるが、この線は２１６の軸ではなく、楕円の最下点から最高点を結ぶ線である。

【０１７８】

この線の（最下点から中心までの）距離は、次式で計算することができる。

【０１７９】

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

この線の（やはり、最下点から中心までの）幅は、次式で与えられる。

【０１８０】

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

従って、スキュー「ｅ」は、ｗ／ｈ

となる。

【０１８１】

次いで、楕円２１７は高さｈをもつ円２１９に変換される。これは、楕円２１７にスケールファクタ

$f=h^2/ab$

を適用することで簡単に求められる。

【０１８２】

その長さに比べて大幅に幅が広い（即ち、ａ＞＞ｂでθが小さい）楕円の場合、生成された円を変換しても十分な解像度が得られず、得られる楕円はギザギザに見える。この問題はスーパーサンプリングで解決される。即ち、はるかに大きい半径をもつ複数の円を追跡し、いくつかの線からの結果を合算してから変換を行う。

【０１８３】

スーパーサンプリングを行う円の大きさを決定する上での慎重な手法は、ａ＞２＊スケール＊ｂとして、スケールファクタ（１に初期化）を続けて倍加させることである。この手法は、ａ≧ｂかつ－９０＜θ＜９０であると仮定しており、－４５＜θ＜４５である場合のみ必要である。

【０１８４】

以下に説明するように、円はブレゼンハムの円アルゴリズムの改良型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の故に、Ｙに関して単調に増加するフラグメントのみが扱われる。従って、図４６のステップ４６９が実施する第２のステップは、楕円の上又は下の頂点をトラバースする孤フラグメントを２つ以上のフラグメントに分割することである。

【０１８５】

ここでは、この分割の可能な実装の１つを説明する。まず、楕円の始点及び終点座標が、スキュー「ｅ」及び尺度「ｆ」を適用することによって円の始点及び終点座標に変換される。次いで、以下のアルゴリズムが実行される。このアルゴリズムは、始点（Ｓ）、終点（Ｅ）、及び方向（ｄ）（時計回り又は反時計回り）を受け付ける。更なる入力として、それぞれ（ｘｃ，ｙｃ－ｈ）及び（ｘｃ，ｙｃ＋ｈ）で与えられる円の最上点（Ｔ）及び最下点（Ｂ）がある。アルゴリズムは、始点ｙ座標、終点ｙ座標、及び円の「側」（左又は右）を含むフラグメントのリストを作成する。

【０１８６】

図６７を見ると、ステップ６７０１でＳ及びＥのｘ座標が、円の中心点（ｘｃ，ｙｃ）のｘ座標と比較される。ＳとＥが互いに反対の側にある場合、ステップ６７０２で、円の上部又は下部が横断されているかどうかを調べるためにテストが行われる。即ち、Ｓが左側にあってｄが時計回りか、或はＳが右側にあってｄが反時計回りか、である。真であるならば、円の上部が横断され、出力は６７０３で示されている通りとなる（Ｔ－＞Ｓ、Ｔ－＞Ｅ）。円の側はセグメントの最下点の側から続くため、６７０４で示されるケースでは、出力はＴ－＞Ｓ（Ｌ），Ｔ－＞Ｅ（Ｒ））となる。図６７で、６７０４及び６７０５は対称性によって関連付けられる、可能な構成である。これ以降、この議論では、全ての出力は、そのセグメントにおける最下点の側を引き継ぐものとする。ステップ６７０２で、このテストによって円の下部が横断されたことが示された場合、出力は６７０６で示され、６７０７及び６７０８で可能な構成が示される。ステップ６７０１で、ＳとＥとが円の同じ側にあることが判明した場合に、ＥがＳの上方であるとステップ６７０９で判断された場合、セグメントの短点及び方向はステップ６７１０で示されるように交換され、それによって更なる検討が簡単になる。

【０１８７】

ＳとＥとが円の同じ側にあるケースを引き続き検討すると、ステップ６７１１で、Ｓ及びＥがＴとＢのいずれも横断するか、それともいずれも横断しないかが判断される。この条件は：Ｓ及びＥが左側にあってｄが時計回りか、或はＳ及びＥが右側にあってｄが反時計回りか、である。真であるならば、６７１２で示されているように、出力は（Ｓ－＞Ｅ）となり、可能な構成が６７１３及び６７１４に示される。偽であるならば、ステップ６７１５で示されているように、出力は（Ｔ－＞Ｂ，Ｔ－＞Ｓ，Ｅ－＞Ｂ）となる。この場合、セグメントＴ－＞ＢはＳ及びＥの反対側にあることになる。可能な構成が６７１６及び６７１７で示される。

【０１８８】

３．６．５　グリフエッジ追跡パラメータの生成

グリフは、ＴＣＩＥシステムパイプライン６９９の下部の部分によって、特殊なエッジとして扱われる。従って、ストローキング、モーフィング及び変換モジュール６１６は、グリフ表示をエッジ表示に変換しなければならない。

【０１８９】

ＴＣＩＥシステム６９９は、グリフをその左側エッジではなく、右側エッジによって追跡する。従って、ストローキング、モーフィング及び変換モジュール６１６に提供されたグリフの左上座標を（Ｓx，Ｓy）、グリフの幅をＷ、グリフの高さをＨとすると、

－　グリフエッジの始点座標は（Ｓx＋Ｗ－１，Ｓy）であり、

－　左側ｚレベルの参照は「イン」ｚレベルとなるように設定され、

－　右側ｚレベルの参照は「アウト」ｚレベルとなるように設定され、

－　終点のＹ座標はＳy＋Ｈである。

【０１９０】
グリフ固有の追跡パラメータは、グリフを示すビットマップ及びグリフの幅からなる。
４．ソーティング
ストローキング、モーフィング及び変換モジュール６１６によって作成された１つのエッジセットは、図５６のソーティングモジュール６１７に供給される。ソーティングモジュール６１７は、これらのエッジが第１にＹ軸について昇順となり、第２にＸ軸について昇順となる（即ち、エッジはＹについて厳密にソートされ、同じＹ値をもつエッジのそれぞれのサブセット内でＸについてソートされる）ように、これらのエッジの順序変更を行う。エッジをこのように順序付けることは、レンダリングプロセスの後の段が「画素順」に実行されるようにするために必要である。画素順とは、レンダリングが１画素ずつ実行されることを意味する。即ち、レンダリングは画面の左上の画素から開始されて、各画素線を左から右に進んでから次の画素線に降り、最後に処理される画素は画面の右下の画素となる。画素順レンダリングでは、従来型レンダリングシステムで使用されるランダムアクセスフレームストアが不要となり、メモリ使用が軽減され、一般にレンダリング速度が向上する。本明細書でのＸ軸及びＹ軸の方向及び優先順位に関する規約は、記載のシステム６９９の動作を損なうことなく任意の代替（但し、直交の）の規約で置き換えられることは当業者には明らかであろう。
【０１９１】
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を使用してその機能を実現する。基数ソートアルゴリズムは、１つの要素セットを、それらに対して一連の部分ソートを行うことによってソートする。各部分ソートは、各要素のソートインデックスの一部（ソートインデックスの、各段で使用される部分は、基数の２を底とする対数に等しいビット長をもつ）のみに関連するソーティングによって実行される。ソートインデックスのこの部分は、要素をどのソーティングビンに入れるかを選択するために使用される。必要なソーティングビンの数は、基数の数に等しい（即ち、必要なソーティングビンの数は、ソートインデックスの各部分によって表現できる、とり得る値の総数に等しい）。要素が各ソーティングビンに入れられる際、それらの要素の既存の順序は各ソーティングビン内で維持され、複数のビン間では維持されない。これによって、後続の部分ソートが、前の部分ソートによって実行された有用なソート作業を取り消さないようになる。各部分ソートが完了すると、全てのビンの中身がひとまとめに順に連結され、それによってソートされている１つの要素セットが（新しい）簡単なシーケンスに復元される。最初の部分ソートは、ソートインデックスの最下位ビットを使い、後続の部分ソートはソートインデックスの次々に大きくなる部分を使う。
【０１９２】
基数ソートアルゴリズムの例を、図４７－１及び図４７－２に示す。
【０１９３】
図４７－１（ａ）は、ソートを必要とする一連の１０個の要素を示す。各要素は、２桁の８進法のソートインデックスをもつ。図番号との混同を避けるため、８進数字「０，１，２，３，４，５，６，７」は記号「Ａ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆ，Ｇ，Ｈ」で置き換えられていることに留意されたい。
【０１９４】
図４７－１（ｂ）は、図４７－１（ａ）に示した一連の要素に対する基数８のソートの第１段の適用を示す。８個のソーティングビンが使用され、各要素はそれ自体のソートインデックスの第１の部分に基づいてこれらのビンに入れられる。この第１の部分ソートに使用されるソートインデックスの部分は、最下位の３ビットである（基数Ｒのソートは、２を底とするＲの対数に等しいビット長（この事例では３ビット）をもつ、ソートインデックスの部分を用いて行われることを想起されたい）。ソートインデックスの最下位の３ビットは最下位の８進数字に対応しているので、第１の部分ソートは、各要素をその要素のソートインデックスの最も右にある８進数字に対応するソーティングビンに割り当てる、ソーティングを必要とする一連の要素のトラバースであると記述することができる。

【０１９５】
図４７－１（ｃ）は、このソートの第２段を示す。この段で、８個全てのソーティングビンの内容が連結され、これは最初のビンの内容から始めて順次に最後のビンまで進む。この結果、ソートを必要とする１０個の要素の新しいシーケンスが得られる。この結果得られたシーケンスは、最も右の８進数字に関して正しく順序付けられており、最上位の８進数字に関しては正しく順序付けられていないことに留意されたい。
【０１９６】
図４７－２（ａ）は、このソートの第３段を示す。この段は、入力シーケンスは図４７－１（ｃ）で示したシーケンスであり、使用されるソートインデックスの部分は第２の３ビット、即ち左側の８進数字（最上位の８進数字）である点以外は第１段の繰り返しである。ソートインデックスこの部分は最上位ビットを含むため、この部分ソートの段階は最後に必要となる段である。この段は、各要素をその要素のソートインデックスの最も左にある８進数字に対応するソーティングビンに割り当てる、ソーティングを必要とする、一連の要素のトラバースであると説明することができる。
【０１９７】
図４７－２（ｂ）は、このソートの第４段を示す。この段は、第２段の繰り返しであるが、ソーティングビンの内容は変更され、今は図４７－２（ａ）に示すようになっている。８個全てのソーティングビンの内容は、最初のビンの内容から始めて最後のビンに至るまで順次再び連結される。この結果、ソーティングが必要な、１０個の要素の別の新しいシーケンスが得られる。得られたこのシーケンスは、最も左（最上位）の８進数字に関して正しく順序付けられているが、最も右（最下位）の８進数字に関しては正しく順序付けられていない。しかし、同じ最も左の８進数字を共有するこの新しいシーケンスのサブシーケンス内では、最も右の８進数字の順序は正しいことに留意されたい。この結果、この新しいシーケンスは、ソートインデックスに関して正しく順序付けられている。このシーケンスを、分かり易いように、図４７－２（ｃ）に再び示す。
【０１９８】
図４７－１，４７－２は、それぞれ６ビットのソートインデックスをもつ一連の１０個の要素に関する基数８のソートを例示している。基数ソートアルゴリズムの原理の、他の２の整数べき乗の基数、他のソートインデックスのサイズ、及び他のシーケンス長への拡張は当業者には明らかであろう。基数が大きくなるに従って、必要な部分ソートの数が減少する（必要な部分ソートの数は、ソートインデックスのビット長を基数の２を底とする対数で除し、それを切り上げた数と等しい）ことに留意されたい。基数が小さくなれば、必要なソーティングビンの数が減少する（必要なソーティングビンの数は、基数の数と等しい）。特定のソーティングタスク用の基数の大きさの選択には、ソーティング速度と、ソーティングビンによって消費されるメモリとの間のトレードオフが伴うため、最適な基数の大きさは、動作環境毎に異なる。
【０１９９】
ここに記載する図形レンダリングシステムに基数ソートアルゴリズムを適用する際、使用されるソートインデックスは、エッジの始点のＸ及びＹ座標を連結して作成される。Ｙ座標の後にＸ座標を連結すると、ソート操作がまず初めにＹについてソートするようになり、Ｘについては副次的にソートするようになる。このようにエッジがソートされた後、所望の画素順次的レンダリングを簡単に実行することができる。
【０２００】
アンチエイリアス処理に対応するために、レンダリングシステム６１０は、エッジ座標をサブピクセル精度で記述することができる。このアンチエイリアス処理については本明細書の他の部分で詳述されているが、ここで言及したのは、エッジ座標の表現がソート操作に影響するからである。ソーティングは、画素順次的レンダリングを促進するために実行されるので、エッジ座標のサブピクセル詳細はソートインデックス作成時に廃棄される。座標が、小数桁内にサブピクセル精度が含まれるような固定小数点表示で表示されている場合、これは小数桁の桁数に等しい桁数だけその座標を右シフトすることによって簡単

に実現することができる。

【０２０１】

基数ソートアルゴリズムの動作は、簡単な最適化によって、多少加速させることができる。この最適化では、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、モーフィング及びストローキングモジュール６１６）の動作に部分的に重ね合わせる。これらのモジュールの動作を厳密に分割する、最適化されていないシステムでは、モジュール間のインタラクションは以下のように論じることができる。変換、モーフィング及びストローキングモジュール６１６は、１つのエッジセットを生成する。これらのエッジは、変換、モーフィング及びストローキングモジュール６１６に都合のいい任意の順序で作成され、この順序でシーケンスと連結される。この一連のエッジが完成すると、基数ソートの適用のために、ひとまとめにソーティングモジュール６１７に供給される。先述の最適化を備えるシステムでは、以下で論じるように、モジュール間のインタラクションを改変する。変換、モーフィング及びストローキングモジュール６１６は、先に述べたようにエッジを生成するが、連結は行わない。エッジは、連結される代わりにソーティングモジュール６１７に直接送られ、ソーティングモジュール６１７はこれらのエッジを第１の部分ソートを適用して処理する（即ち、エッジはそれぞれの最下位ビットに基づいてソーティングビンに分配される）。後続の部分ソート操作は、変換、モーフィング及び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にエッジを供給し終えてから実行される。この最適化が可能なのは、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１つのオブジェクトセットが作成されている間に「急いで」実行することができ、後続の部分ソートは１つのオブジェクトセットが完成した後でしか実行できないからである。

【０２０２】

５．エッジ処理

図５６に見られるように、表示リストコンパイラ６０８の出力は、画面上の（単一フレームの）所望の出力を記述する順序付けられた１つのエッジセットである。レンダリングエンジン６１０は、この順序付けられた１つのエッジセットを入力として受け取る。表示リストコンパイラ６０８が次のフレーム用にエッジを準備する間に、レンダリングエンジン６１０は１つのフレームについてエッジを処理できるようにするために、ダブルバッファ６０９を使用することができることに留意されたい。

【０２０３】

５．１　入力及び出力

レンダリングエンジン６１０を通る表示データの流れは、エッジ処理モジュール６１４の動作で始まるが、そのデータフローパラメータは図２２（ａ）及び（ｂ）に要約されている。アニメーションムービー内の現フレームは、順序付けられた１つの新規エッジセット２３２及び順序付けられた１つの静的エッジセット２３３によって記述される。順序付けられた１つの新規エッジセット２３２は、（先に説明した）表示リストコンパイラ６０８によって現フレームのために作成されたエッジを含んでいる。これらの新規エッジ２３２は、前のフレームにはなかったか、又は前のフレーム以降に新しい位置に移動された図形オブジェクトを記述する。順序付けられた１つの静的エッジセット２３３は、前のフレームで新規エッジとして導入された後、存続している（かつ、動かないでいる）オブジェクトを記述する。

【０２０４】

エッジ処理２４２がフレームの全てのエッジ（新規エッジ２３２及び静的エッジ２３３のどちらも含む）を処理した後には、エッジ処理２４２は、Ｚレベル・アクティブ化モジュール６１３に入力として提供される順序付けられた１つの交差メッセージセット、及び次フレームでエッジ処理２３９が使用する順序付けられた１つの静的エッジセット２４１を出力として生成したことになる（即ち、次フレームの開始時にセット２４１はセット２３３となる）。

【０２０５】

エッジ処理２４２はまた、現フレーム以降出力表示に寄与しなくなるエッジを廃棄する責も担う。このようなエッジは、除去のマークを付けなければならない。この除去マークは、外部のエンティティ（例えば、ドライバ）によって付けられ、本明細書の範囲外である。各エッジが順序付けられた１つの新規エッジセット又は順序付けられた１つの静的エッジセットから取り出された後、エッジ処理２４２は、そのエッジが除去用にマークされているかどうかをチェックする。そうである場合、エッジは、次フレーム２４１用の順序付けられた１つの静的エッジセットに移動されない。

【０２０６】

５．２　トップレベル動作

図２４に、単一フレームでのエッジ処理２４２の要約を示す。各フレームをレンダリングするために、エッジ処理２４２は、ディスプレイの下方に向かって走査線（行）毎に反復し、順序付けられた１つの新規エッジセット又は順序付けられた１つの静的エッジセットの任意のエッジが現在の走査線と交差する位置を計算するように動作する。当業者なら、この基本アルゴリズムは他の走査方向にも適用可能である。

【０２０７】

図２４を参照すると、フレームの処理の開始時に、ステップ２５５で現走査線（例えば、一番上の走査線）のカウンタが０に初期化される。ステップ２５６で、任意のエッジが現走査線と交差すると判定されたとき、ステップ２５７でこれらのエッジが処理される。現走査線カウンタが、最後の走査線が処理されたことを示す（ステップ２５８＝yes）場合、そのフレームについての処理が完了し、そうでない場合、ステップ２５９で現走査線カウンタが増分され、後続の走査線のエッジ処理が実施される。

【０２０８】

５．３　アクティブエッジ追跡

図２２（ｂ）は、現走査線２５７のエッジ処理を行う際にエッジ処理２３９を通るデータの流れの外観を示す。現走査線と交差する（従って処理が必要な）エッジはアクティブエッジと呼ばれ、順序付けられた１つのアクティブエッジセット２３６内に走査順に格納される。エッジ処理２３９は、新規エッジ２３４又は静的エッジ２３５の始点座標が現走査線と交差するとき、その新規エッジ２３４又は静的エッジ２３５からアクティブエッジを生成する。アクティブエッジは、新規エッジ又は静的エッジのフォーマットとはわずかに異なるフォーマットをもつ。例えば、始点座標（Ｘs，Ｙs）をもつ代わりに、アクティブエッジは、走査線毎に更新されるcurrent_Xフィールドをもつ。current_Xフィールドは、エッジが現走査線と交差（横断）する場所に対応する。各アクティブエッジが処理された後、アクティブエッジが次の走査線も交差すると判定された場合、そのアクティブエッジは走査順に、次の走査線２３８のための順序付けられた１つのアクティブエッジセット内に置かれる。

【０２０９】

図２３は、単一の走査線におけるエッジ処理２３９の動作（図２４のステップ２５７に対応）をより詳細に示す。ディスプレイの最初の走査線をレンダリングする前は、順序付けられた１つのアクティブエッジセット２３６は空である。この順序付けられたセットは、エッジ処理２３９が、順序付けられた１つの新規エッジセット２３２のうちのある新規エッジ又は順序付けられた１つの静的エッジセット２３３内のある静的エッジが交差する走査線を処理するまで、空のままである。このような走査線に到達するまで、エッジ処理２３９は何もすることがない。即ち、順序付けられた１つのアクティブエッジセットが空（ステップ２４４で判定＝yes）であり、かつ残っている静的又は新規エッジはない（ステップ２４６＝yes）か、或は全ての静的又は新規エッジが後の走査線にならないと開始しない（ステップ２４７＝no）。

【０２１０】

順序付けられた１つのアクティブエッジセットが空（ステップ２４４＝yes）であり、ある新規又は静的エッジが現走査線上のどこかで開始すると判定された場合（ステップ２４６＝noかつステップ２４７＝yes）、その新規又は静的エッジは、対応する順序付けら

れた１セットの静的又は新規エッジから外され、ステップ２５０で処理すべきアクティブエッジに変換される。

【０２１１】

エッジ処理２３９は、常にエッジを走査線に沿って走査順（Ｘ座標について昇順）に処理しなければならない。前の走査線の処理の結果、順序付けられた１つのアクティブエッジセット中に既にアクティブエッジがある可能性がある（即ち、ステップ２４４＝no）。既にアクティブエッジがあり、かつ、任意の新規又は静的エッジがあり（ステップ２４５＝no）、それが現走査線上で開始する（ステップ２４８＝yes）場合、次にどれを処理すべきかを決定するために、ステップ２４９で次の新規／静的エッジのＸ座標を次のアクティブエッジのＸ座標と比較しなければならない。エッジの３つのソース（新規、静的及びアクティブ）は走査順に格納されているので、処理すべき次のエッジは常に、各ソース１４（図２（ａ）を参照のこと）の次に利用可能なエッジを検討することによって決定できる。

【０２１２】

５．４　エッジ処理の例

図２（ａ）の例は、四角形の形状１４の上に重なり、それを隠蔽する三角形の形状１５を示す、ムービの最初のフレームの望ましい出力を示す。この例では、表示画素の現走査線１６が重ね合わせてある。現走査線の最も左の表示画素は１３である。この表示画素はＸ座標０に対応し、そのすぐ右の画素はＸ座標１に対応し、以下同様である。

【０２１３】

エッジ処理２３９で図２（ａ）に示されるフレームのための任意の出力を生成する前に、表示リストコンパイラ６０８は、このフレームの望ましい出力を記述する４つの新規エッジを含む順序付けられた１つの新規エッジセットを生成しているはずである（水平エッジは除去されることを想起されたい）。図２（ｂ）は、この４つのエッジが現走査線１６と交差しているところを示す。エッジ１９及び２２は、それぞれ四角形に対応するｚレベルのアクティブ化及び非アクティブ化を行う。エッジ２０及び２１は、それぞれ三角形に対応するｚレベルのアクティブ化及び非アクティブ化を行う。

【０２１４】

エッジ処理２３９が図２（ｂ）に示される現走査線を処理するまでに、エッジ１９及び２２は順序付けられた１つの新規エッジセット内に置かれ、エッジ２０及び２１は順序付けられた１つのアクティブエッジセット内に置かれる。エッジ２０及び２１は、前の（より上位の）走査線によって、既に新規エッジからアクティブエッジに変換されている。

【０２１５】

これらのエッジは、図２３に概略を示すように、現走査線について処理される。要約すると、この例では、（最少のＸをもつため）エッジ１９を最初に処理することになる。エッジ１９はまず、そこから何らかの交差メッセージが生成される前に、アクティブエッジに変換される。次にエッジ２０及び２１が処理され、最後にエッジ２２がアクティブエッジに変換され、次いで処理される。この例では、現走査線についてこれら４つのエッジを処理することにより、図２（ｃ）に示される１つの交差メッセージセットが生成される。これらの生成及びフォーマットについては、セクション５．８でより詳しく説明する。

【０２１６】

５．５　エッジのアクティブエッジへの変換及びエッジパーシスタンス

再び図２３を参照すると、処理すべき次エッジが順序付けられた新規エッジセット又は順序付けられた静的エッジセットに由来する場合は、ステップ２５０で処理のためにそのエッジをまずアクティブエッジに変換しなければならず、そうでない場合は、ステップ２５１で次のアクティブエッジが取られる。処理すべき次エッジが新規又は静的エッジであり、その新規又は静的エッジは除去用にマークされていない場合、その新規又は静的エッジは、次フレームのための順序付けられた１つの静的エッジセット２３７内に入れられる。これは、エッジが複数フレームに渡って存続できることを意味する。特定フレーム上のエッジの除去を可能にする機構を実装する様々な方法がある。１つの技法は、全てのエッ

ジに識別フィールドを割り当て、各フレーム毎に、そのフレームで除去すべきエッジに対応する１つの識別フィールドセットを提供できるようにすることである。別の技法は、エッジによって参照されたｚレベル内でフラグを提供することによって、全ての参照エッジが除去すべきであることを示すことである。

【０２１７】

５．５．１　静的エッジパーシスタンス

静的エッジバッファの動作例を、図６８（ａ）～（ｃ）及び図６９（ａ）～（ｃ）に示す。図６８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６８（ａ）は、家６８０６からなる最初のフレーム６８００を示す。図６８（ｂ）は、前のフレーム６８００から家６８０６を保持しながら、家６８０６の玄関に立っている人６８００を追加する第２のフレーム６８０２を示す。図６８（ｃ）は、再び家６８０６を保持しながら、今度は人を家の玄関から家６８０６の右の位置６８１０まで動かす第３のフレーム６８０４を示す。

【０２１８】

図６９（ａ）～（ｃ）は、図６８に見えるフレームを作成するために必要な処理動作の時間シーケンスをそれぞれ示す。図６８（ａ）で分かるように、まず、現フレーム６８００に追加すべき新規エッジを含む新規エッジバッファ６９０１を作成するために、家６８０６を表すオブジェクトが表示リストコンパイラ６０８に入力される。他の場所で説明した図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て新規エッジバッファ６９０１が保持するエッジが生成されるかを示す。入り静的エッジバッファ６９０２は、前のフレームから保持すべき静的エッジを含む。次いで、バッファ６９０１及び６９０２は、現フレーム６８００のレンダリングされたディスプレイ６９０３を出力するために、レンダリングエンジン６１０によってレンダリングされる。図６９（ａ）はまた、次フレームのために保持すべき静的エッジを含む出静的エッジバッファ６９０４も示す。

【０２１９】

図６９（ａ）で、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を実行して図１６（ａ）の座標を処理する。この特定の例では、モーフィング及びストローキングは必要ないが、それでも表示リストコンパイラ６０８は座標を正しい画面位置に変換し、エッジを画面順にソートするために処理時間を消費する。処理済エッジは、レンダリング操作に備え、新規エッジバッファ６９０１内に置かれる。アニメーションの最初のフレーム６８００である（従って、前のフレームからエッジを保持することができない）ため、入力静的エッジバッファ６９０２は空である。エッジバッファ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のための動作を終了させ、レンダリングエンジン６１０がこのフレーム６８００のための動作を開始する。レンダリングエンジン６１０は、エッジバッファ６９０１及び６９０２からのエッジをマージさせ、ディスプレイ６９０３上にレンダリングし、次フレーム用に保持すべきエッジを出力静的エッジバッファ６９０４内に出力する。これで、アニメーションの最初のフレーム６８００の処理は全て完了する。

【０２２０】

図６８（ｂ）及び（ｃ）で、参照番号６８０８及び６８１０は、それぞれ第２及び第３のアニメーションフレーム６８０２及び６８０４でレンダリングされることになる人の形状の輪郭を形成する座標を指す。

【０２２１】

図６９（ｂ）で、アニメーションの第２のフレーム６８０２の処理が行われる。図６９（ｂ）の入力静的エッジバッファ６９０２は、図６９（ａ）の出力静的エッジバッファ６９０４と同じであることに留意されたい。これが可能なのは、入力及び出力静的エッジバッファがフレーム間で「ピンポン」式にスワップされるからである。図６９（ｂ）の出力静的エッジバッファ６９０４は、「ピンポン」スワップの直後にクリアされる。

【０２２２】

表示リストコンパイラ６０８は、このフレーム６８０２に追加すべき新規エッジの座標６８０８を処理し、処理済エッジを新規エッジバッファ６９０１に入れる。家６８０６のエッジは、既に処理された形で前のフレームから保持されているため、表示リストコンパイラ６０８が処理する必要はない。この静的エッジ技法で所要処理時間が短縮されるのは、このためである。

【０２２３】

前のフレームの処理と同様に、表示リストコンパイラ６０８は停止され、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。エッジバッファ６９０１及び６９０２の内容はマージされ、ディスプレイ６９０３上にレンダリングされ、次フレーム６８０４のために保持すべきエッジは出力静的エッジバッファ６９０４に格納される。

【０２２４】

図６９（ｃ）で、アニメーションの第３のフレーム６８０４の処理が行われる。表示リストコンパイラ６０８は、このフレーム６８０４に追加すべき新規エッジの座標６８１０を処理し、処理済エッジを新規エッジバッファ６９０１に入れる。前のフレーム６８０２と同様に、家６８０６のエッジは、既に処理された形で前のフレーム６８０２から保持されているため、表示リストコンパイラ６０８が処理する必要はない。しかし、人の形状６８１０は前のフレームに比べて移動され拡大されているため、処理を行う必要がある。従って、表示リストコンパイラ６０８は座標６８１０を処理し、人の形状の処理済エッジを新規エッジバッファ６９０１に入れる。

【０２２５】

前と同様に、表示リストコンパイラ６０８は停止され、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。エッジバッファ６９０１及び６９０２の内容はマージされ、ディスプレイ６９０３上にレンダリングされ、次フレームのために保持すべきエッジは出力静的エッジバッファ６９０４に格納される。ここでは、家オブジェクト６８０６はアニメーションの第４のフレームに含めるために、改変することなく保持されると仮定されていることに留意されたい。そうでない場合、家の形状は出力静的エッジバッファ６９０４に含まれない。

【０２２６】

５．６　アクティブエッジ処理

ステップ２５２での次のアクティブエッジの処理で、後続モジュールであるＺレベル・アクティブ化モジュール６１３に渡される１つ又は複数の交差メッセージが生成される。各メッセージは、アクティブエッジの現Ｘ座標と、エッジの左側及び右側ｚレベル参照を共に含む。

【０２２７】

アクティブエッジが、少なくとも次の走査線まで表示画面を下方に延びると判定された場合（図２３でステップ２５３＝no）、ステップ２５４で、このアクティブエッジは次の走査線での処理用の順序付けられた１つのアクティブエッジセットに追加される。

【０２２８】

５．７　エッジの追跡

図２３のステップ２５２で実行される、走査線毎にエッジのＸ座標を追跡するこのプロセスは、従来技術ではしばしば「エッジ追跡」と呼ばれる。本明細書のシステム６９９では、エッジは直線、２次ベジエ曲線、楕円弧又は２次多項式フラグメントでよい。４つのエッジタイプのそれぞれについてのエッジ追跡技法を以下に説明する。

【０２２９】

５．７．１　直線の追跡

直線の描画の方法として、ブレゼンハムの線アルゴリズムが当技術分野で周知である。エッジ処理２３９は、ステップ２５２で、各走査線毎にあるアクティブ直線エッジのＸ座標を計算するために、改良型ブレゼンハムアルゴリズムを使用する。図５のＣ言語のソースコードは、ある始点座標（Ｘs，Ｙs）からある終点座標（Ｘe，Ｙe）まで直線を追跡（及び描画）するために使用される改良型ブレゼンハムアルゴリズムの例を示す。この例は

、事前計算された３つの値、err，delta_err1及びdelta_err2に基づく、線のＸ軸の範囲Ｙｓ～Ｙｅの各Ｙ座標に関する増分式の計算を示す。

【０２３０】

直線であるエッジのデータが表示リストコンパイラ６０８（セクション３．６．１－直線エッジ追跡パラメータの生成を参照のこと）によって生成される際、データは少なくとも

－　始点座標（Ｘｓ，Ｙｓ）、
－　終点Ｙ座標（Ｙｅ）、
－　勾配の整数項（ｇｒａｄ）、
－　勾配の剰余項（ｉｅｎｔ）、
－　左フラグ、ならびに
－　左側及び／又は右側ｚレベルの参照

を含むように生成される。

【０２３１】

ステップ２５０で直線エッジがアクティブ直線エッジに変換されると、ient項は、直線エッジのデータから以下のようにして計算されるerr，delta_err1及びdelta_err2で置き換えられる。

【０２３２】

ｔｙ＝Ｙｓ－Ｙｅ

とすると、次式が成立する。

【０２３３】

err＝（２×ient）－ｔｙ
delta_err1＝２×ient
delta_err2＝２×（ient－ｔｙ）

次いで、各走査線上で、アクティブエッジの新しい現Ｘ座標が、ステップ２５２に従って改良型ブレゼンハムアルゴリズムを以下のように使用して、既存の現Ｘ座標ならびにパラメータerr，delta_err1及びdelta_err2から計算することができる。

【０２３４】

errが０よりも少ない場合、
err＝err＋delta_err1
current_X＝current_X＋grad
そうでない場合、
err＝err＋delta_err2
current_X＝current_X＋grad
左フラグが真である場合、
current_X＝current_X－１

となる。

【０２３５】

そうでない場合、
current_X＝current_X＋１

５．７．２　２次ベジエ曲線の追跡

２次ベジエは、図７の例で示されるように、始点、終点及び制御点によって表される。２次ベジエ曲線であるエッジのデータが表示リストコンパイラ６０８によって処理される際（セクション３．６．２－２次ベジエ曲線追跡パラメータの生成を参照のこと）、２次ベジエエッジ記述は、少なくとも

－　始点座標（Ｘｓ，Ｙｓ）、
－　終点Ｙ座標（Ｙｅ）、
－　追跡パラメータＡ，Ｂ，Ｃ及びＤ
－　符号フラグ（＋１又は－１）、及び
－　左側及び／又は右側ｚレベルの参照

を含むように再フォーマットされ、エッジ処理２３９に提示される。

【０２３６】

各走査線毎に、ベジエ曲線のcurrent_X位置は、図２３のステップ２５２で、Ａ，Ｂ，Ｃ及びＤに基づいて以下のように再計算される。

【０２３７】

Ａ＝Ａ＋Ｂ

Ｃ＝Ｃ＋Ｄ

符号フラグが＋１の場合、

Current_X＝Ａ＋２√Ｃ

そうでない場合、

Current_X＝Ａ－２√Ｃ

５．７．３　２次多項式フラグメントの追跡

表示リストコンパイラ６０８（セクション３．６．２－２次ベジエ曲線追跡パラメータの生成を参照のこと）は、２次多項式フラグメント用のエッジ表現を、それらが少なくとも

－　始点座標（Ｘs，Ｙs）、

－　終点Ｙ座標（Ｙe）、

－　追跡パラメータＡ，Ｃ及びＤならびに

－　左側及び／又は右側ｚレベルの参照

を含むように生成する。

【０２３８】

各走査線毎に、２次多項式フラグメントのcurrent_X位置は、ステップ２５２で、Ａ，Ｃ及びＤに基づいて以下のように再計算される。

【０２３９】

Ａ＝Ａ＋Ｃ

Ｃ＝Ｃ＋Ｄ

current_X＝Ａ

５．７．４　楕円弧の追跡

表示リストコンパイラ６０８は、楕円弧のエッジ表現を、それらが少なくとも

－　等価円の中心（Ｃx，Ｃy）、

－　等価円の半径「Ｒ」、

－　等価円の始点座標（Ｓx，Ｓy）、

－　始点エラー値（Ｅ）、

－　円から楕円への変換（スキュー「ｅ」及びスケール「ｆ」）、

－　終点Ｙ座標（Ｅy）、及び

－　左又は右の向き

を含むように生成される。

【０２４０】

図２３のステップ２５０がこのデータをアクティブな楕円弧に変換するとき、現座標、現エラー値及び（以下に説明する）現モードがこのデータから初期化される。

【０２４１】

追跡アルゴリズムは、円の中心に対して座標を追跡する。従って、現座標は、始点座標から等価円の中心を差し引いたものに設定される。現エラー値は、表示リストコンパイラ６０８から提供されたエラー値に設定される。モードは、現在走査されている８分円を把握しておくために使用され、ステップ２５０に供給されている始点座標に基づいて初期化される。

【０２４２】

いつでも、絶対楕円座標は関連する現座標（ｃx，ｃy）から、以下の公式を使用して導出することができる。

【０２４３】

（ＳＥx，ＳＥy）＝（ｆｃx＋ｅｃy＋Cx，cy＋Cy）
円は、現座標及びエラー値を反復的に更新するために、ブレゼンハムの円アルゴリズムを使用して追跡される。当技術分野で周知のこのアルゴリズムは、円の右上の８分円（図１４の８分円１２０）だけを追跡し、他の７つの８分円を生成するために一連の反転を使用する。更なる詳細は、「Computer Graphics：Principles and Practical」：Fole andVan Dam, Second Edition; Section 3.3 Scan COnverting Circles. に記載されている。
【０２４４】
５．７．５　グリフの追跡
アクティブなグリフオブジェクトは、グリフビットマップ中への現ポインタを維持する。このポインタは、グリフエッジが追跡されている間、各行毎に増分される。グリフの左側エッジは垂直であるため、エッジが追跡されている間に現Ｘ位置を更新する必要はない。
【０２４５】
５．８　アンチエイリアス処置及び交差メッセージの生成
アンチエイリアス処置は、追加の（サブピクセル）解像度を使用してエッジを追跡することによって実現される。当技術分野で周知のように、この追加の解像度は、ある図形プリミティブが各画素のどのフラクションを占めているかを判定するために使用することができる。このフラクション（一部の従来技術では画素カバレージと呼ばれる）は、各出力画素の色への図形プリミティブの寄与を減衰させ、それによって大幅に改善されたレンダリング結果を生むために使用することができる。例えば、エッジの座標はＸとＹ共に２ビットを追加した解像度で提供することができ、それによってＸとＹ共に４倍の解像度を提供する。
【０２４６】
エッジ処理２３９が追加の解像度をどのように使用するかについて、図１（ａ）～（ｃ）を参照して以下に説明する。図１（ａ）は、アクティブエッジ０が、現在レンダリングされている表示画素の走査線１に入るところを示す。図１（ｂ）で、この走査線は、エッジデータのＹ座標内への２ビットの追加によって提供される追加のサブピクセル解像度を表す４つのサブピクセル線（spel線）３，４，５及び６に分割される。次いで、エッジのＸ位置が、先に述べたエッジ追跡技法のうち適切な一技法を使用して、４つのspel線それぞれについて、サブピクセル精度で反復的に決定される。追跡されているエッジが少なくとも１つのspel線と交差する各表示画素毎に、エッジがどのようにその画素に影響するかを表す４×４ビットマスク表現が作成される。このようなビットマスクは、当技術分野でしばしばＡバッファと呼ばれ、「The A-buffer, an Anti-aliased Hidden Surface method」, Carpenter, L., Computer Graphics, Vol. 18, No. 3, Jul. 1984（以下、「Carpenter」と称する）で初めて記述された。Ａバッファは、表示画素内のどのspelがエッジの影響を受けるかを示すために使用され、エッジ処理モジュール６１４からＺレベル・アクティブ化モジュール６１３に渡される交差メッセージに含まれる。この例では、図１（ｃ）に見られるように、３つのＡバッファが生成される。即ち、現走査線画素Ｘ＝２，Ｘ＝３及びＸ＝４に対応するＡバッファ８，１１及び１２である。図１（ｃ）に見られるように、spel９はエッジの影響を受けないが、spel１０はエッジの影響を受ける。
【０２４７】
図２３のステップ２５２で、あるアクティブエッジについて、交差メッセージ及び関連するＡバッファの生成が実施される。このステップを、図３を参照して以下に詳しく説明する。
【０２４８】
図３の最初のステップ３５で、current_spel_line値が０に初期化される。これは、現走査線の一番上のspel線を最初に処理すべきであることを示す。Ａバッファが空に初期化され、current_pixel値が、アクティブエッジが現走査線に入る場所の表示ピクセルに対応するＸ値で初期化される。このＸ値は、アクティブエッジのcurrent_X座標の整数部分である。

【０２４９】
次いで、ステップ３６で、処理中のアクティブエッジのcurrent_X座標がcurrent_pixel値と比較して、current_Xがcurrent_pixelで表される表示画素内にあるかどうかを調べる。第１の反復では、ステップ３５の結果、これは真となる。
【０２５０】
次のステップ３９は、current_spel_lineに対応する、Ａバッファの単一の行を改変するように動作する。但し、Ａバッファの一番上の行はcurrent_spel_line＝０に対応する。current_X座標の非整数データは、current_spel_lineに対応するＡバッファの行上に何個のspelを「置く」べきかを判断するのに使用される。例えば、current_Xによって２ビットのサブピクセル精度が提供されている場合、Ａバッファ８，１１及び１２で示されるように、１行あたり４個のspelを表すＡバッファを生成することができる。表４は、current_spel_lineに対応するＡバッファの行をどのように設定するかを決定するために２ビットの非整数データをどのように使用するかを示す。
【０２５１】
【表４】

表４

current_X の非整数ビット	Aバッファ行			
	最左端の spel	二番目の spel	三番目の spel	最右端の spel
00b	1	1	1	1
01b	0	1	1	1
10b	0	0	1	1
11b	0	0	0	1

【０２５２】
再び図１（ｂ）の例を参照すると、一番上のspel線３（current_spel_line＝０）の処理中、アクティブエッジ０のcurrent_X値は整数成分４（１００ｂ）及び非整数成分０．７５（非整数ビットは１１ｂ）をもつ。この事例では、Ａバッファ１２の一番上の行の一番右のspelのみが表４に従って設定される。
【０２５３】
Ａバッファが改変された後、図３に見られるように、ステップ４０で、アクティブエッジが次のspel線４と交差する場所に対応する新しいcurrent_Xが計算される。これは、先に記述した追跡技法の１つをアクティブエッジのデータに適用することによって実現される。ステップ４１で、current_spel_lineが増分され、更に処理すべきspel線がある場合（ステップ４２＝no）、このプロセスはアクティブエッジのcurrent_Xをcurrent_pixelと比較するステップ３６に戻る。
【０２５４】
アクティブエッジはcurrent_spel_lineとcurrent_pixelが示す表示画素とは異なる表示画素内で交差することを、current_Xが示す場合、ステップ３７が実行され、このステップは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。
【０２５５】
－　Ｘ座標＝current_pixel
－　Ａバッファ
－　（アクティブエッジの左側ｚレベル参照から複製された）アクティブ化されたｚレベル参照
－　（アクティブエッジの右側ｚレベル参照から複製された）非アクティブ化されたｚ

レベル参照
これらのｚレベル参照は、ヌルのフィルへの参照でもよいことに留意されたい。
【０２５６】
生成された交差メッセージは、出力として、ソート手段（例えば、ソートバッファ）を介してＺレベル・アクティブ化モジュール６１３に渡される。このソート手段は、Ｚレベル・アクティブ化モジュール６１３が交差メッセージを走査順（即ち、Ｘ座標昇順でソート）に受け取るようにするために必要である。
【０２５７】
次いで、Ａバッファが空となるように再初期化され、current_pixelはアクティブエッジ（０）のcurrent_X値の整数部分に設定される。次いで、Ａバッファは、先に述べたように、current_pixelのcurrent_spel_line５に対するアクティブエッジの影響を示すために、Ａバッファ１１になるように改変される。
【０２５８】
残りのspel線６について処理が続行され、それによってＡバッファ１１を含む交差メッセージが出力される。
【０２５９】
全てのspel線の処理が終わる（ステップ４２＝yes）と、先に述べたように、ステップ４３で、Ａバッファ８を含む最終交差メッセージが生成され出力される。
【０２６０】
５．８．１　グリフのための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、１バイト幅（又は８画素幅）のブロックに分割され、それが順次処理される。各バイトは、２５６エントリのルックアップテーブル（ＬＵＴ）へのインデックスとして使用され、各エントリは３２ビットを占める。ＬＵＴ中の単一のエントリは、８画素からなる行のどの画素で交差メッセージを生成しなければならないかを示す。
【０２６１】
各エントリは、ニブル毎に処理される。最初のニブルはバイトによって生成される交差の個数を記録し、後続の各ニブルは、領域の始点に対する交差の相対位置を記録する。全ての画素に交差をもつバイトという特異な例では、８個の交差と０個の位置を記録した特別なエントリが使用される。
【０２６２】
図２６を参照すると、幅１６、高さ２のビットマップ２６５に関する例が示されている。２進表現２６５は、｛0xD6，0x54，0x99，0xcc｝となる（各バイト中のビット「７」は、対応する画素ブロック内の最も左の画素を指していることに留意されたい）。このビットマップの最初のバイトを取り上げ、その絶対値をＬＵＴ２６６中にルックアップすると、０，２，３，４，５及び７の位置に６つの交差があることが分かる。これは、ＬＵＴ２６５が画面座標（100,100）に置かれていると仮定すると、（100,100），（102,100），（103,100），（104,100），（105,100），（107,100）で交差メッセージを生成しなければならないことを意味する。８個の交差メッセージを生成するバイトも、この図で見ることができる。
【０２６３】
ＬＵＴ２６６がインデックスされ、バイトについて１つの交差点セットが得られた後で、追加の交差点を挿入し、又は正しくない交差点を削除することができる。これは、ビットマップの各行の初めで０に初期化されるフラグによって規定される。このビットがクリアされている場合、交差点は改変されない。このビットが設定されているときは、画素ブロックの一番左の位置に対応する交差点（「ゼロ交差点」）が存在しない場合、交差点が１つ挿入される。反対に、ゼロ交差点が存在する場合は、それが削除される。
【０２６４】
フラグのステータスは、行内の各バイトの処理完了時に更新される。交差メッセージが奇数個あるとフラグは反転され、一方交差メッセージが偶数個あるとフラグの現在の値が

保持される。フラグが行の末尾でセットされる場合、行を正しく終了させるために最終交差メッセージが出力される。このメッセージの位置は、現画素ブロックのすぐ右の画素ブロックの先頭である。

【０２６５】

再び図２６を参照すると、ビットマップ内の４バイトを横断してこのフラグのステータスを追跡することが可能である。第１行の初めで、フラグは０に初期化される。第１のバイトは偶数個の交差点に分解されるので、フラグはこのバイトの終わりではトグルされない。しかし、第２のバイトは、奇数個の交差点に分解され、その結果、フラグは１に切り替えられる。第２のバイトはその行の最後のバイトであるため、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。

【０２６６】

次に第２行に移ると、フラグは０に初期化される。第３のバイトは、奇数個の交差点に分解されその結果、フラグはこのバイトの終わりで１に切り替えられる。第４のバイトは、０，１，２，３，４，５，６及び７で８個の交差メッセージに分解される。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ため、このメッセージは削除され、その結果７個の交差メッセージが出力される。交差点は奇数個なので、フラグは再び０に切り替えられる。このとき、処理は行の終わりに来ており、フラグが設定されてないため、最終交差メッセージは出力されない。

【０２６７】

どの場合でも、交差メッセージは、全てのサブピクセルが－１に設定されたＳバッファを備える。この結果、グリフは常に偶奇屈曲規則を使用してレンダリングすべきである。

【０２６８】

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ス処理

図６６は、図２６に示すものと同じ走査線を示す。その走査線上の単一の画素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、全てのサブピクセルが－１に設定されている。この交差メッセージの結果、ある画素６６０３全体が出力される。このため、グリフはアンチエイリアス処理されず、サブピクセルは検討されない。これは、「ゼロ交差点」のフラグを考慮して追加された交差メッセージも含めて、あるグリフ用に生成された全ての交差メッセージについて言える。

【０２６９】

５．９　交差メッセージの例

引き続き図２（ｃ）に示される例を参照すると、エッジ処理モジュール６１４によって図２（ｂ）に示されるアクティブエッジ用に生成された交差メッセージが示されている。この例では、アクティブエッジ座標が表示画素内のサブピクセル（spel）に対応する２ビット追加の解像度で記述される、４×４アンチエイリアス処理が使用される。最初に、エッジ１９の処理が行われ、従って対応する交差メッセージ２３が最初に生成される。アクティブエッジは、この時点で四角形のフィルを記述した左側Ｚレベル参照をもっているので、生成された交差メッセージ２３のアクティブ化されたＺレベル参照は、四角形のフィルを記述したアクティブ化されたｚレベル参照をもつことになる。

【０２７０】

次に、アクティブエッジ２０の処理が行われる。モジュール６１４（図３で概説した）がアクティブエッジ２０の４個のspel線のうち最初の３個を反復処理する間に、交差メッセージ２５が生成される。アクティブエッジは、別の画素上で第４（一番下）のspel線と交差するため、このspel線用に別の交差メッセージを生成する必要があり、その結果、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、どちらも、三角形の形状のフィルを記述したアクティブ化されたｚレベル参照をもつ。

【０２７１】

同様に、アクティブエッジ２１用に交差メッセージ２６及び２７が生成される。これらは、三角形の形状のフィルへの非アクティブ化された（元のアクティブエッジ２１の右側ｚレベル参照から複製された）Ｚレベル参照をもつ。最後に、エッジ２２から、四角形の

形状フィルへの非アクティブ化されたｚレベル参照をもつ交差メッセージ２８が生成される。

【０２７２】

５．９．１　他の例

図７０は、ゼロ交差メッセージの受領によるＳバッファの更新を示す更なる例を示す。ある走査線内の画素＃１について、最初は交差するエッジがないので、その画素に関する交差メッセージもＳバッファｎもない。そのため、その画素＃１について、Ｓバッファｎは、０で満たされるようにセットされる。次いで、このＳバッファｎの一番右側の列が、次の画素＃２用のＳバッファｎ－１の最初（即ち、一番左）の列に複製される。この列が、次いでＳバッファｎ－１の各列全体に複製され、その結果、この例ではこのバッファも０で満たされる。図に示されるように、あるエッジが画素＃２と交差しており、サブラインとの交差（４、２又は１本のサブラインとの交差）に関する図示された規則に従って、画素＃２についてゼロ交差メッセージが決定される。図を見ると分かるように、このメッセージの右下の４分の１は＋１で満たされ、それによって走査線と交差するエッジを補完している。画素＃２では、ゼロ交差メッセージとＳバッファｎ－１が合計され、これによってこの画素のＳバッファｎが与えられる。この事例では、バッファはヌルであったため、交差メッセージはそれだけである。次の画素＃３で、このＳバッファｎの一番右側の行が再び複製され、この事例ではＳバッファｎ－１が１で満たされる。画素＃３での交差メッセージは、交差に関する規則に従って決定され、当該のエッジが画素の右上の４分の１の部分で２つのサブラインのみと交差したことが分かる。交差メッセージ及びＳバッファｎ－１は再び合計され、画素＃３のＳバッファｎが与えられる。画素＃４で、一番右側の列が再び複製され、Ｓバッファｎ－１を横断して満たされる。規則を使用して再び交差メッセージが決定され、それによって画素＃４のＳバッファｎが与えられる。ここで説明する、交差メッセージ及びＳバッファを使用してアンチエイリアス処理を行う手法は、サブピクセルベースでのレベル合成を必要とせずに、走査線上の画素位置のアクティブレベルを正確に判定する能力を提供する。全てのサブピクセルプロセスは、Ｓバッファ及び交差識別子を使用して行われ、従って合成の責務は画素レベルに委ねられる。

【０２７３】

５．１０　アクティブエッジ及び交差メッセージの順序変更

走査線のアクティブエッジ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昇順）に生成、出力されない。例えば、図２（ｂ）でエッジ２０が処理される際、交差メッセージ２５が最初に生成、出力され、その後にメッセージ２４が続く。交差メッセージ２５は、メッセージ２４よりも大きいＸ座標をもつため、交差メッセージは走査順に出力されない。更に、エッジ処理モジュール６１４の動作中の新しいcurrent_Xの計算結果により、第１のアクティブエッジが、この走査線上で既に処理されている第２のアクティブエッジよりも低い走査位置をもつようになる可能性がある。

【０２７４】

図２１に、この状況の例を示す。図中、２本の水平破線２３０及び２３１は、出力表示の走査線を表している。上の破線２３０は、エッジ処理モジュール６１４が現在処理中の走査線を表し、下の破線２３１は次に処理すべき走査線を示す。図２１は、現走査線２３０とそれぞれ交差点２２４，２２５及び２２６で交差する３個のアクティブエッジ２２１，２２２及び２２３を示す。アクティブエッジのcurrent_Xフィールドは、交差点の位置を示している。エッジ処理モジュール６１４は、次の走査線２３１上の交差点２２７，２２８及び２２９に対応する各アクティブエッジ毎に新しいcurrent_X値を計算する。この例では、アクティブエッジ２２３がアクティブエッジ２２１及びアクティブエッジ２２２と交差したため、これらの改変されたアクティブエッジはもはや走査順にはない。従って、改変されたアクティブエッジは、次の走査線用の順序付けられた１つのアクティブエッジセットに入れられる前に、再ソートされる必要がある。この例では、次の走査線用の順序付けられた１つのアクティブエッジセット中のアクティブエッジの望ましい順序は、まず２２３、次いで２２１、そして最後に２２２である。

【０２７５】
エッジ処理モジュール６１４は、改変されたアクティブエッジが、次の走査線用の順序付けられた１つのアクティブエッジセット中に移される前に走査順に並ぶように、改変されたアクティブエッジをあるソート手段（例えば、ソートバッファ）に挿入する。また、Ｚレベル・アクティブ化モジュール６１３が交差メッセージを走査順に受け取るようにするために、（図２３のステップ２５２で生成されるような）出力交差メッセージもソート手段（例えば、ソートバッファ）に通す必要がある。
【０２７６】
６．Ｚレベル・アクティブ化モジュール
Ｚレベル・アクティブ化モジュール６１３は、フレームがレンダリングされている間にどのｚレベルが出力色に寄与するかを把握するために、エッジ処理モジュール６１４から渡されたエッジ交差点データを使用する。出力画素のストリームが、レンダリングエンジン６１０によって走査順に生成されることになる。エッジ処理モジュール６１４からの各交差メッセージは、走査順に作成される際に要求される出力色が変化する表示座標を表している。以下の説明では、特定のラスタ空間位置で始まる走査順に連続する１つ又は複数の画素は画素のランと呼ばれる。この説明では、レンダリングシステム６１０がアンチエイリアス処理を行う場合、エッジ交差点においてＡバッファが生成されると仮定する。
【０２７７】
６．１　順序付けられた１つの関心対象Ｚレベルセット
Ｚレベル・アクティブ化モジュール６１３は、図５９にその概略が示される「関心対象の」ｚレベルのリストを維持する。ここで、関心対象とは、問題のｚレベルがＺレベル・アクティブ化モジュール６１３によって考慮されなければならないことを意味する。Ｚレベル・アクティブ化モジュール６１３は、順序付けられた１つの関心対象ｚレベルセットを使用して、交差メッセージの入力を、どのｚレベルが現走査線に沿った画素のランのための出力色に寄与するかを示す領域メッセージの出力に変換する。即ち、ランは２つの隣り合った交差メッセージで表される点の間にある。
【０２７８】
Ｚレベル・アクティブ化モジュール６１３の動作は、交差メッセージのストリームを、スパン上にどの奥行きが存在し、スパン全体に渡るそれらの奥行きの相対奥行き順序が何であるかを示す領域メッセージの出力に変換するという面から見ることもできる。この場合のスパンは、交差メッセージによってバウンディングされた走査線の内部を意味する。ｚレベルは奥行きをフィルにバウンディングするため、このような順序付けられた１つの奥行きセットは、どのｚレベルが現走査線に沿った画素のランのための出力色に寄与するかを判定するために使用することができる。これは、後で検討するコンポジタの役割である。反対に、スパン上に存在する１つのｚレベルセットを判定することは、スパン上に存在する奥行きを判定することに等しい。
【０２７９】
６．２　Ｚレベル・アクティブ化モジュールにおける制御のフロー
エッジ処理モジュール６１４の場合と同様に、Ｚレベル・アクティブ化モジュール６１３の動作はレンダリング作業を走査線毎に考慮し、入力（交差メッセージ）を走査順に受け取ることを予想している。単一の走査線に関するＺレベル・アクティブ化モジュール６１３の機能が、図５８に要約されている。ある走査線での処理は、ステップ６２８で、アクティブｚレベルのリストを空として初期化し、Current_X値を０にセットすることによって開始される。Current_Xは、現在処理されている画素を示すために使用され、０の値が走査線の最初（一番左）の画素を表す。
【０２８０】
走査線に対する交差メッセージは、左から右に向かう順序（Ｘ昇順）で処理される。
【０２８１】
この走査線に対するエッジ交差メッセージがない、即ちステップ６２７でnoと判定された場合、ステップ６２９で、Ｚレベル・アクティブ化モジュール６１３が直ちに走査線全

体のレンダリングを要求する。この段階では関心対象のｚレベルのリストは空であるため、この要求の結果、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。

【０２８２】

この走査線に対する交差メッセージがあると判定された場合（ステップ６２７＝yes）、ステップ６３０で、次に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値と交差メッセージのＸ座標が等しくない場合、Ｚレベル・アクティブ化モジュール６１３は、この交差メッセージを処理する前に、Current_X値と交差メッセージのＸ座標との間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。この要求により、ステップ６３１で、レンダリングエンジン６１０内の合成モジュール６１２は、順序付けられた１つの関心対象ｚレベルセットの内容を使用して、この画素領域用の出力色を生成する。

【０２８３】

Current_X値と交差メッセージのＸ座標が等しい場合（ステップ６３０でyes）、交差メッセージは現画素用の色生成に対応し、従って交差メッセージの処理が進められる。

【０２８４】

６．３　Ｚレベルのアクティブ化及び非アクティブ化：屈曲カウント

自己交差するオブジェクトのレンダリングを容易にするために、Ｚレベル・アクティブ化モジュール６１３は屈曲規則を利用する。単に、このモジュールは、ｚレベルがアクティブ化されているかどうかについてのブール指標を記憶するのではなく、各アクティブｚレベル毎に小さな屈曲カウント整数値を記憶する。屈曲カウントは、当業者には周知である。使用できる屈曲カウントのうちの２つが、従来技術で一般に記述されている。即ち、非ゼロと偶奇（パリティ）である。使用できる第３の規則（前の２つよりも一般的ではないが、従来技術に記述されている）は、以下の「セクション６．７－ＳバッファのＡバッファへの変換：屈曲規則」に記述される負の屈曲である。

【０２８５】

交差メッセージは、「アクティブ化」すべきｚレベルの参照を含むことがある。即ち、そのｚレベルでは屈曲カウントが減分されなければならない。交差メッセージはまた、「非アクティブ化」すべきｚレベルの参照を含むことがある。即ち、そのｚレベルでは屈曲カウントが増分されなければならない。このｚレベルの増分及び／又は減分は、図２（ｄ）に示されるように、画素の交差メッセージのＡバッファに示される部分のみに適用される。Ａバッファ２９，３０及び３１は、交差メッセージ２３，２４及び２５によってアクティブ化されたサブピクセルの例である。３２，３３及び３４は、交差メッセージ２６，２７及び２８によって非アクティブ化されたサブピクセルの例である。

【０２８６】

出力画素へのｚレベルの寄与に対するこれらアクティブ化及び非アクティブ化の影響は、どの屈曲規則がそのｚレベルと共に使用されているかによって変わる。

【０２８７】

６．４　順序付けられた１つの関心対象Ｚレベルセット：続き

図５９に見られるように、Ｚレベル・アクティブ化モジュール６１３によって、屈曲カウントの２次元アレイが関心対象のｚレベルのリスト内の各ｚレベル毎に格納される。屈曲カウントの２次元アレイは、Ｓバッファと呼ばれる。各屈曲カウントは、現在レンダリングされている画素内のうちの１サブピクセルについてのｚレベルの状態を表す（６４３、６４７及び６５１）。それによって、Ｚレベル・アクティブ化モジュール６１３が、各ｚレベルが現画素のどの部分の上でアクティブであるかを把握することができるようになる。順序付けられた１つの関心対象ｚレベルセットの順序は奥行きの順序である。図５９に示される例では、ｚレベル６４０，６４４及び６４８はそれぞれの奥行き６４１，６４５及び６４９によって順序付けられている。フィルデータ６４２，６４６及び６５０は、ｚレベル６４０，６４４及び６４８の色を定義する。

【０２８８】

６．５　順序付けられた１つの関心対象Ｚレベルセットへの新規ｚレベルの追加

再び図５８を参照すると、新しい交差メッセージがアクティブｚレベルをもたない場合（ステップ６３８＝no）、このアクティブｚレベルに関連する処理ステップは実行されない。同様に、交差メッセージが非アクティブｚレベルをもたない場合（ステップ６３９＝no）、この非アクティブｚレベルに関連する処理ステップは実行されない。

【０２８９】

新しい交差メッセージがｚレベルをアクティブ化した場合、そのｚレベルがまだ関心対象のｚレベルのリスト内にはない場合（ステップ６３２＝no）、ステップ６４０ａで関心対象のｚレベルのリストがチェックされる。関心対象のｚレベルが満杯である場合、ゼロ屈曲カウントだけで満たされたＺレベルがリストから外される。その後、ステップ６３８で検出されたｚレベルに対応する新しいエントリが、ステップ６３４に示されるように、このリストのｚレベル奥行き順序が維持されるようにこのリストに挿入される。交差メッセージのＡバッファが、新しく挿入されたｚレベルに関連付けられることになるサブピクセルカウントのアレイ（即ち、このｚレベルのＳバッファ）を初期化するのに使用される。Ａバッファで示されるサブピクセルが、－１の値で初期化されるべきＳバッファ内のサブピクセルカウントに対応する。残りのサブピクセルカウントは、０の値で初期化される。Ａバッファから屈曲カウントのアレイへのこの変換の例を、図２（ｄ）に示す。

【０２９０】

新しい交差メッセージが、既に順序付けられた１つの関心対象ｚレベルセット中にあるｚレベルをアクティブ化した場合（６３２＝yes）、その交差メッセージのＡバッファは、そのｚレベルに関連付けられたＳバッファ内のどのサブピクセルカウントが減分されるべきかを判定するために使用される。

【０２９１】

引き続き図５８を参照すると、新しい交差メッセージがまだ関心対象のｚレベルのリスト内にはないｚレベルを非アクティブ化した場合、ステップ６３６で、対応する新しいエントリがこのリストのｚレベル奥行き順序が維持されるようにこのリストに挿入される。交差メッセージのＡバッファが、新しく挿入されたｚレベルに関連付けられることになるサブピクセルカウントのアレイ（即ち、このｚレベルのＳバッファ）を初期化するのに使用される。Ａバッファで示されるサブピクセルが、＋１の値で初期化されるべきＳバッファ内のサブピクセルカウントに対応する。残りのサブピクセルカウントは、０の値で初期化される。Ａバッファから屈曲カウントのアレイ、即ちＳバッファへのこの変換の例が図２（ｄ）に示されている。ここで、それぞれＸ＝１４，Ｘ＝１５及びＸ＝１５をもつ交差メッセージは、いずれもｚレベルを非アクティブ化する交差メッセージである。

【０２９２】

新しい交差メッセージが、関心対象のｚレベルのリスト内に既にあるｚレベルを非アクティブ化した場合（６３５＝yes）、その交差メッセージのＡバッファは、そのｚレベルのＳバッファ内のどのサブピクセルカウントが増分されるべきかを判定するために使用される。Ａバッファ内で示される各サブピクセルについて、ステップ６３７で、Ｓバッファ内の対応するサブピクセルの屈曲カウントが増分される。

【０２９３】

図７１は、図５８に示される交差メッセージの処理手法とは別の手法を示す。図７１で、図５８に示される参照番号と一致する全ての参照番号は同じ機能をもつ。図７１は、図５８のステップ６４０ａが取り除かれ、それによってステップ６３４がステップ６３２での「no」判定の直後にくる点で、図５８と異なる。図７１はまた、ステップ６３７の後に、関心対象のＺレベルのリスト内のエントリが、全てゼロである屈曲カウントをもつかどうかをチェックするように動作する追加のステップ６４１ａを含む。そのような屈曲カウントをもつ場合、そのエントリはもはや有効ではなく、従ってリストから取り除かれる。このように、図７１の手法は、レンダリングされている画像に寄与できなくなったエントリをリストから取り除くように動作し、新規エントリのリストに空きが必要となるときのみエントリを取り除く図５８とは対照的である。

【０２９４】

６．５．１　ハードウェアにおける順序付けられた１つの関心対象Ｚレベルセットの維持

ここで、議論をＡＳＩＣなどのハードウェア実行に転じることができる。Ｚレベル・アクティブ化モジュール６１３は、図５２（ｂ）に示されるように、ＺレベルをＺレベル・アクティブ化テーブル（ＺＬＡＴ）内で走査線毎に管理する。走査線が左から右へと処理される間に、交差メッセージによって参照されたＺレベルが徐々にＺＬＡＴに取り込まれる。各入りＺレベルには、Ｓバッファ５７３、フィル情報５７２を格納するためのＺＬＡＴエントリが割り当てられ、Ｚレベル奥行き５７１によってアドレス可能である。

【０２９５】

ＺＬＡＴ内のＺレベルエントリは、奥行きで順序付けられてはいない。むしろ、交差メッセージに入れられて受け取られた順序で並べられている可能性が高い。図５２（ａ）は、特定のＸ座標でアクティブである５つのＺレベルと、その相対奥行きの例を示しており、一方図５２（ｂ）は、ＺＬＡＴ内の５つのＺレベルエントリが奥行きで順序付けられていないことを示している。Ｚレベルエントリは、マップ５７０によって、Ｚレベル・アクティブ化モジュール６１３内で奥行きについて降順で維持されている。マップ５７０は、ＺＬＡＴと同じ個数のエントリを含むテーブルであり、それぞれがＺＬＡＴ内のそれぞれのエントリに対するマップング参照を含むローカル奥行きの順序付けられたリストである。例えば、Ｚレベル５５５は１のローカル奥行き（０のローカル奥行きが最も高いので、これは２番目に高いＺレベルである）をもち、ＺＬＡＴエントリ３、即ちＺレベル５５５にマッピングされている。更新されたマップ５７４も図示されている。

【０２９６】

Ｚレベルは、この機構を使用して２通りのやり方でアクセスできる。本開示のＺＬＡＴは、Ｚレベル奥行きを供給することによって対応するＺレベルエントリが返される内容アドレスメモリとして実装される。このため、ＺレベルのＺＬＡＴへの容易な取込み及びＺＬＡＴからの容易な取出しを可能にしながら、スパン内のアクティブなオブジェクトの個数に応じてＺＬＡＴのサイズを変えることができる。ＺＬＡＴは、交差メッセージを受け取ったとき、Ｓバッファの状態を更新する際にこのようにしてアクセスされる。反対に、ＺＬＡＴエントリはまた、マップテーブルを介してローカル奥行きでアドレスすることもできる。即ち、特定のローカル奥行きでのＺレベルは、マップをローカル奥行きに適用することで割り出すことができる。ＺＬＡＴは、合成すべき特定のスパン上でアクティブな順序付けられた奥行きのサブセットを割り出すときにこのようにしてアクセスされる。

【０２９７】

６．５．１．１　ハードウェアでの新しい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って初めて参照されるときにＺＬＡＴに追加される。その後の参照では、それぞれのエントリは更新のみを必要とし、ＺＬＡＴ内の同じ位置に留まる。次の新しいＺレベルエントリが挿入されるＺＬＡＴ内の位置を追跡するために、入力カウンタが維持される。カウンタはまず、走査線の始点で位置０にリセットされ、ＺＬＡＴに新しいＺレベルが追加される毎に増分される。従って、ＺＬＡＴが満杯になるまでは、Ｚレベルエントリは、交差メッセージに入れられて受け取られた順序に維持される。

【０２９８】

走査線の始点で、ＺＬＡＴは最初は空である。マップ５７０も、ローカル奥行き０はＺＬＡＴエントリ０へのマッピングを含み、ローカル奥行き１はＺＬＡＴエントリ１へのマッピングを含み、以下同様であるような初期状態になる。次いで、新しいＺレベルがエントリとしてＺＬＡＴに追加されたとき、このＺレベルは入力カウンタが指し示す位置に挿入され、マップ５７０はＺレベルの順序変更があればそれ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は、入りＺレベルのローカル奥行きが、現在ＺＬＡＴ内で維持されているＺレベルのローカル奥行きに対して相対的に決定されなければならない。即ち、どのＺレベルが入力Ｚレベルよりも高いローカル奥行きをもち、どのＺレベルが入力Ｚレベルよりも低いローカル奥行きをもっているかを判断する。

【０２９９】
このプロセスを、図５３（ａ）及び５３（ｂ）に示す。図５３（ａ）に示される、このプロセスの第１のステップとして、ＺＬＡＴ５７８内で現在維持されているＺレベルエントリが、追加されるＺレベルと比較される。ＺＬＡＴエントリは、それぞれのＺレベル奥行きを比較することにより、入力Ｚレベルよりもローカルで高いか低いかがマークされる。図５３（ａ）で、ＺＬＡＴエントリは、リスト５７７に追加されるＺレベル５７５よりも高い（マーク＝０）か、又は低いか（マーク＝１）がマークされる。
【０３００】
第２のステップは、マップ５７９によって判断された通りに、マーキングをそれぞれが関連するローカル奥行きの順序に順序変更することである。これは、マップ内のローカル奥行きを、入力Ｚレベルのローカル奥行きよりも高いものと低いものとに判別する効果がある。順序変更されたマーキング５７６を使用して、マップ５７９が、新しいＺレベルの追加によるローカル奥行きの新しい順序を反映するように更新される。
【０３０１】
マップ５７９を更新するために、５７５として示される追加されるべきＺレベルよりも低いとマークされたローカル奥行きはその順序が下げられ、一方新しいローカル奥行きはその順序が上げられ、その結果次のマップ５８０が作成される。それぞれのＺＬＡＴエントリのローカル奥行きが変更されるが、ＺＬＡＴエントリはＺＬＡＴ内で静的に維持されるので、そのマッピングは変更されないことに留意されたい。例えば、ＺＬＡＴ５７８内で、奥行き９のＺレベルは、新しいＺレベル５７５が追加される前は現マップ５７９内で４のローカル奥行きをもっている。そのマッピングは、ＺＬＡＴ５７８の位置１へのマッピングであり、これは新しいＺレベル５７５が追加された後にそのローカル奥行きが次のマップ５８０で５に下げられた後でも変更されない。
【０３０２】
図５３（ｂ）は、図５３（ａ）から生じるＺＬＡＴ５８１に更にＺレベル５８６を追加する別の例と、マップ５８４及び５８５を適切に更新するステップとを示す。
【０３０３】
６．５．１．２　ハードウェアでのＺレベルエントリの再ソート
拡張例として、Ｚレベルエントリは、ＺＬＡＭ内でまず関心対象順で、続いてＺレベル奥行きの降順で維持することができる。関心対象のＺレベルは、それ自体のＳバッファに対するそれ自体のフィル規則の適用が非ゼロＡバッファをもたらすようなＺレベルである。即ち、関心対象のＺレベルは潜在的に寄与するＺレベルである。Ｚレベル奥行きが変化する可能性はないが、Ｚレベルの状態が関心対象から関心対象外へ変化する状況があり、その逆の状況もある。これは、可能な２つのシナリオで起こる。Ｚレベル・アクティブ化モジュール６１３が、ＺＬＡＴ内に既にあるＺレベルを参照する交差メッセージ、及びその交差メッセージにやはり関連付けられたＡバッファを受け取り、Ｚレベルにその関心対象に関する状態を変更させたときに起こり得る。また、ＺＬＡＴエントリのＳバッファの一番右のspelがＳバッファ全体に複製されなければならないスパンの処理の完了時に起こり得る。このステップは、関心対象のＺレベルを関心対象から外す効果をもつことができる。
【０３０４】
Ｚレベルがその関心対象に関する状態を変える事象は、それぞれのＺレベルのＳバッファの更新以外には、ＺＬＡＴ内のエントリに何の影響も与えない。しかし、この事象では、マップが新しいローカル奥行き順序を反映するように更新される必要がある。従って、マップによって維持されるローカル奥行きは、挿入－ソートではなしに再ソートされる必要がある。
【０３０５】
図５４（ａ）は、ＺＬＡＴ５９０内にあるＺレベル５８７が関心対象でなくなるシナリオを示す。ＺＬＡＴ５９０の現状態５９９は、いくつかの奥行きが関心対象であり、いくつかの奥行きが関心対象外であることを示しており、現マップ５９１は現ローカル奥行き

順序を反映している。これは、ＺＬＡＴ５９０のパーティション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、ＺＬＡＴ５９０内のエントリに対して、新しいＺレベルを追加するプロセスの場合と同様にしてマーキングが行われる。エントリは、ローカル奥行き順序に関して、Ｚレベルが変化しつつある状態よりも上にあるか下にあるかがマークされる。マーク列５８９中、エントリはそのローカル奥行き順序が、変化するＺレベルが更新された後にそのＺレベルよりも高い場合、１でマークされる。マーキングも、ローカル奥行きステータス５８８を反映するように同様に順序変更され、マップ５９１は、変化しつつあるＺレベルよりも高いローカル奥行きマッピングを上げ、変化しつつあるＺレベルのマッピングをより低いローカル奥行きに下げることによって、マップ５９２に更新される。

【０３０６】

図５４（ｂ）は、エントリ５９３が関心対象になるとき６００の、このプロセスを示す。この例では、マーク列５９５に示されるように、ＺＬＡＴ５９６内のエントリはそのローカル奥行きが変化するＺレベル５９３のローカル奥行きよりも低くなる場合に１でマークされる。マークは同様に順序変更５９４され、マップ５９７は、マークされたローカル奥行きマッピングが低いローカル奥行きに下げられ、変化しつつあるＺレベルのローカル奥行きが上げられるように５９８に更新される。

【０３０７】

６．５．１．３　Ｚレベルエントリの置換

現走査線上の非アクティブのＺＬＡＴエントリを除去するための機構は不要である。新しいＺレベルを追加する必要があり、ＺＬＡＴのサイズが有限であるためにＺＬＡＴ内の空きがなくならない限り非アクティブエントリが無視される遅延ポリシーが実装できる。このような状況では、一番下のローカル奥行きによってマップされたＺレベルエントリは取り除かれ、追加されるＺレベルがＺＬＡＴ内のそのエントリを占める。マップテーブルは、ＺＬＡＴ内の明渡しを要求されるエントリがそれまで空であったかのように更新される。

【０３０８】

６．５．１．４　単一走査線上の簡略型ＺＬＡＭ管理

図６５Ａ及び６５Ｂは、走査線上の順序付けられた１つの関心対象ｚレベルセットが、図５８に記述した方法で更新される際にどのように変化するかについての外観を提供する。分かり易いように、セクション６．５．１．３に記載の非アクティブｚレベルの再利用に関する任意選択の遅延ポリシーは図示されておらず、非アクティブｚレベルはテーブルから取り除いてある。

【０３０９】

図６５Ａは、範囲［０、１８］にＸ座標をもち、特定の走査線６５０８上でレンダリングされる、四角形６５０２、円６５０４及び三角形６５０６からなる画像６５００を示す。図６５Ｂは、走査線６５０８上でアクティブとなっているエッジに関連して、各エッジに遭遇したときのｚレベル・アクティブ化テーブルを形成する順序付けられた１つの関心対象レベルセット６５３０を示す。

【０３１０】

６５１１で、四角形６５０２に関連付けられた左側エッジが走査線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、四角形６５０２に関連付けられたｚレベル６５２１が順序付けられたセット６５３０に追加される。

【０３１１】

６５１２で、三角形６５０６に関連付けられた左側エッジが画素＃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に関連付けられたｚレベル６５２２が順序付けられた１つの関心対象ｚレベルセット６５３０に追加される。この場合、図５８の方法の適用は、三角形６５０４のフィルが順序付けられたセット６５３０の最高位に挿入される結果をもたらす。フィルは、セット６５３０が奥行き順序に維持され、三角形６５０６が四角形６５０２の上になるように挿入される。

【０３１２】
同様に６５１３で、円６５０４のｚレベル又はフィル６５２３は、三角形６５０６及び四角形６５０２のそれぞれのｚレベル６５２２及び６５２１の間に挿入される。６５１４で、四角形６５０２が非アクティブとなったときに、その四角形６５０２のフィル６５２１が除去されることにより、順序付けられたセット６５３０の順序付けが影響を受けることはない。６５１５で、円６５０４が非アクティブとなり、そのフィル６５２１は順序付けられたセット６５３０から除去され、その結果空となる。
【０３１３】
６．６　ランの処理
ここで、ハードウェアとの関連がより高い問題の議論から説明を戻して、セクション６．５でのより一般的な方法の説明を続ける。
【０３１４】
交差メッセージの処理は、Current_Xに対応する交差メッセージがなくなる（６３０）まで続く。このとき、図４０に示されるように、Ｚレベル・アクティブ化モジュール６１３は、交差メッセージが他にもある場合はその処理に進む前に、合成モジュール６１２に画素のランのための出力色６３１を生成するよう要求しなければならない。
【０３１５】
図４０は、合成モジュール６１２へのランデータの出力、及びそれに関連する、１つの関心対象ｚレベルセットが変更されない画素のランの追跡を示す。
【０３１６】
まず、ステップ３９９で、表示画素Current_Xのための色の生成が要求されなければならない。このとき、順序付けられた１つの関心対象ｚレベルセットの内容は、この画素の色に寄与する全てのｚレベルを記述しており、各ｚレベル毎に、それぞれのＳバッファによってサブピクセルカバレージ情報が提供される。図４０の残りの部分については、セクション６．８で説明する。
【０３１７】
６．７　ＳバッファのＡバッファへの変換：屈曲規則
合成モジュール６１２は、それぞれの関心対象のｚレベルに関連付けられたＡバッファを入力として取る。従って、Ｚレベル・アクティブ化モジュール６１３は、合成モジュール６１２にランデータを渡す前に、それぞれの関心対象のｚレベルに関連付けられたＳバッファをＡバッファに変換する。その場合、Ｚレベル・アクティブ化モジュール６１３は屈曲規則を使用する。先に説明した３つの屈曲規則は図５７（ａ）及び５７（ｂ）に示されている。図中、Ｓバッファ６２４及び６２０は、それぞれの屈曲規則により、以下のようにそれぞれＡバッファ６２１，６２２，６２３及び６２５，６２５，６１９に変換される。
【０３１８】
－　偶奇：Ａバッファ６２３及び６１９で示されるように、屈曲カウントが奇数の場合、フィルはサブピクセルに寄与する
－　非ゼロ：Ａバッファ６２１及び６２６で示されるように、屈曲カウントがゼロ以外の場合、フィルはサブピクセルに寄与する
－　負の屈曲：Ａバッファ６２２及び６２５で示されるように、屈曲カウントが負の場合、フィルはサブピクセルに寄与する
ある形状４２１全体に対するこれらの屈曲規則の結果を、図４３（ａ）に示す。図４３（ａ）で、走査線４２４の上のいくつかの屈曲カウント４２０，４２６及び４２２が示されている。同様に、屈曲カウント４２９，４２７及び４２３は走査線４２５の上に存在する。図４３（ｂ）では、非ゼロ屈曲４２８、負の屈曲４３０及び偶奇４３１の各規則を適用した結果が、形状のエッジがハッチされたフィルを画定するものとして、図４３（ｂ）の表現で形状の境界内に見られる。
【０３１９】
負の屈曲規則は、最低限の計算を使用してストロークされたエッジを作成しレンダリン

グする上で特に有用である。パスのストローキングのプロセスは、本明細書中、変換、モーフ及びストローキングモジュール６１６（セクション３．４を参照のこと）の一部として説明した。ストロークされたパスの一部分の例を、図３６（ａ）～（ｃ）に示す。図３６（ａ）は、原点３５９から時計回りに三角形のパスを形成する、ストロークされるべき３個のエッジ３６２、３６１及び３６０からなる順序付けられたセットを示す。ストロークされるべき順序付けられた１つのエッジセットが、「ヌルの」左側ｚレベル３６６、右側ｚレベル３６７及びストロークｚレベル（即ち、ペンの色）３６５と共に提供される。この３個のエッジの順序付けられたセット及び関連付けられたｚレベルによって表される形状の望ましい出力例が、図３６（ｂ）に示されている。図３６（ｂ）に示される望ましい出力を表すエッジを実現するために、変換、モーフ及びストローキングモジュール６１６は、原点３５４での一番上の継目の拡大表示を示す図３６（ｃ）に示される左側ストロークエッジ３５５及び右側ストロークエッジ３５６を作成しているはずである。ストロークｚレベル３６５が透明度をもたない場合、ストロークプロセスは、

－　全ての左側ストロークエッジ３５５の左側ｚレベル参照にヌルの左側ｚレベル３６６、

－　全ての左側ストロークエッジ３５５の右側ｚレベル参照にストロークｚレベル３６５、

－　全ての右側ストロークエッジ３５６の左側ｚレベル参照にストロークｚレベル３６５、

－　全ての右側ストロークエッジ３５６の右側ｚレベル参照に右側ｚレベル３６７

をそれぞれ割り当てているはずである。

【０３２０】

図３６（ｃ）は、２本の走査線３５７及び３５８を示す。２本の各走査線の上方には、走査線に沿った様々な点に、元の１つのエッジセットの右側ｚレベル参照３６７の屈曲カウントを表す数字が書かれている。偶奇又は非ゼロのいずれかの屈曲規則が使用される場合、図３６（ｃ）に示すように、この領域の屈曲カウントは「１」であるため、右側ストロークエッジによって囲まれた自己交差するエッジによって形成された望ましくない三角形区域３６８がｚレベル３６７によってレンダリングされる。負の屈曲規則を使用すると、「１」の屈曲カウントは非アクティブと見なされ、従って継目が正しくレンダリングされる。

【０３２１】

６．８　ランの処理：続き

図４０のランの処理の議論に戻ると、ステップ３９９で合成モジュール６１２はＡバッファ及び順序付けられた１つの関心対象フィルセットを使用して、単一の画素のアンチエイリアス処理された色を計算する。

【０３２２】

関心対象のｚレベルのリストはまた、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の色を生成するために使用される。ただし、各ｚレベルの屈曲カウントのアレイ（Ｓバッファ）が、まず、後続の画素に対する各ｚレベルの正しいカバレージを示すように改変されなければならない。というのは、カバレージが画素Current_Xに対するカバレージと異なる可能性があるからである。これらの後続の画素では、各ｚレベルのカバレージは、次の交差メッセージのＸ座標まで変化しない。

【０３２３】

再び図２を参照すると、四角形１４の左側エッジは、エッジ処理モジュール６１４に交差メッセージ２３（Ｘ＝４）を生成させた。Ａバッファは、画素Ｘ＝４について、四角形をフィルするのに使用したｚレベルが出力画素の右下隅の４つのサブピクセルでアクティブであることを示す。しかし、この四角形を正しくレンダリングするためには、このｚレベルは更に、ｚレベルが非アクティブ化されるＸ＝１５の右側エッジまでの走査線に沿った全ての後続の画素上の下方の２本のサブ走査線に対してもアクティブであり続けるべきである。Ｘが５以上１５未満の表示画素へのｚレベルの寄与を表す正しいＳバッファは、

図８に示されるように、一番右のサブピクセル値を同じ行の他の全てのサブピクセル値に複製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における、交差メッセージ６４に対応するＳバッファ６５を示す。Ｓバッファ６６は、一番右の屈曲カウントを各行について他の全ての屈曲カウントに複製した結果を表す。この結果は、四角形をレンダリングするのに使用した、Ｘが５以上１４以下のｚレベルの屈曲カウントを正しく表している。

【０３２４】

再び図４０を参照すると、ステップ４００は、順序付けられた１つの関心対象ｚレベルセット中の全てのｚレベルについて、一番右の屈曲カウントを全体に複製するように動作し、Current_Xが増分される。ステップ４０１に従って、ここでランの残りの画素のために合成モジュール６１２を呼び出すことができる。現走査線で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から現走査線の終わりまでの画素を生成するよう要求される。

【０３２５】

これらの画素に対して生成された色は、依然として、例えば１つ又は複数のアクティブｚレベルが混色又はビットマップである場合、ラン全体を通して一定ではない可能性があることに留意されたい。ラン全体を通して一定であるのはカバレージ情報及び関与するｚレベルだけである。関心対象のｚレベルのリストの内容に基づく画素色の生成についての詳細が、合成モジュール６１２に関する記述（セクション７を参照のこと）にある。図４０の最後のステップ４０２は、Current_Xを、利用可能な次の交差がある場合、その交差のＸ座標となるように改変することである。

【０３２６】

順序付けられた１つの関心対象ｚレベルセット中の各ｚレベルのＳバッファからバッファを生成する手順は、単一の画素を合成する手順と同じであり、上述の「Ｓバッファのバッファへの変換：屈曲規則」と題するセクションに記述されている。

【０３２７】

最後に、図５８に戻ると、現走査線について、エッジ交差メッセージは、それがなくなるまで処理されるが、それはステップ６２７で判断される。図５８の操作は、レンダリングされるべき全ての走査線毎に繰り返される。

【０３２８】

７．合成モジュール

合成モジュール６１２は、入力として

－　順序付けられた１つのｚレベルセット及びＺレベル・アクティブ化モジュール６１３によって作成された、それぞれのＡバッファ、及び

－　作成されるべき出力画素のランの長さを規定する出力表示空間内の座標

を取り、出力として

－　出力表示装置での表示に適した画素のラン

を作成する。

【０３２９】

順序付けられた１つの入力ｚレベルセットは、奥行き順にソートされている。ｚレベル奥行きの概念については、既に説明してある。以下の議論では、上と下という用語が使用される。２つのｚレベルＡ及びＢが与えられ、最終出力でｚレベルＡがｚレベルＢを覆い隠しているように表される場合、ｚレベルＡはｚレベルＢの上又は上方にあると見なされる。逆に、ｚレベルＢはｚレベルＡの下又は下方にあると見なされる。更に、一番上のｚレベルとは、概念的に他の全てのｚレベルの「上」にあると見なされるｚレベルを指し、即ち他の全てのｚレベルを覆い隠すことができ、逆に他の全てのｚレベルに覆い隠されることはない。逆に、一番下のｚレベルとは、他のどのｚレベルをも覆い隠すことはできず、他の全てのｚレベルに覆い隠されることができるｚレベルである。更に、入力セット中の一番下のｚレベルの下にあると見なされる、背景ｚレベルが常に存在する。背景ｚレベルは、順序付けられた１つの入力ｚレベルセットに含まれず、入力セットの一部をなすｚ

レベルについてのいかなる議論も背景ｚレベルを含まない。背景ｚレベルが必要である場合は、その旨を明示する。

【０３３０】

７．１　中間

合成プロセスの間、ある中間合成バッファへのアクセスが必要となる。このバッファは、最大サイズ、即ち出力装置の幅のランに対するどんな中間合成結果を保持するのにも十分な大きさをもつことができる。この合成バッファは任意の大きさをとることができ、従って記載の各プロセスを繰り返し適用することが必要となることがあることを、当業者は理解されるであろう。

【０３３１】

合成モジュール６１２はまた、単一の画素についての任意の合成結果を格納するのに十分な大きさをもつ作業バッファも利用する。これらのバッファは、ハードウェア実装の場合はＡＳＩＣ内に形成することができ、或はソフトウェア実装の場合は汎用メモリに形成することができ、一般に赤、青、緑及びアルファ（透明度）の構成要素をもつ。

【０３３２】

７．２　ｚレベルのフィル

各ｚレベルは、以下のいずれか１つでよいフィルの参照を含む。

【０３３３】

ソリッドフィルは、ランの全長を通じて、カラー値及びアルファ値が一定であるようなフィルである。ソリッドフィルの例としては、不透明な赤のフィル、又はアルファが５０％の青のフィルなどがある。

【０３３４】

ｘ依存フィルは、画素のランの全長を通じて、カラー値が必ずしも一定ではないフィルである。この定義では、ｘ依存フィルは、ランの全長を通じて一定のアルファ値をもつ。ｘ依存フィルの例としては、全ての混合制御点で同じアルファをもつ線形又は径方向の混色、或は（一定のアルファをもつ）ビットマップテクスチャフィルなどがある。

【０３３５】

可変アルファフィルは、ランの全長を通じて、アルファが可変であるようなフィルである。可変アルファフィルの例としては、全ての混合制御点でアルファが一定ではない線形又は径方向の混色、或は一定ではないアルファをもつビットマップテクスチャフィルなどがある。

【０３３６】

７．３　基本フロー

合成アルゴリズムの基本フローを図１３に示す。これは高レベルの概略であり、いくつかのステップに関するより詳細な記述は後ほど提供される。

【０３３７】

最初のステップ１１０は、入力セットが空であるかどうかを調べる。そのセットが空であるときは、フィルが存在せず、描画されるべきフィルは背景フィルであることを意味し、ステップ１１１で、それが生成される。これで合成プロセスは終了する。

【０３３８】

入力セットが空でない場合、次のステップ１１２で、順序付けられた１つの入力ｚレベルセットに可変アルファに分類されるフィルがあるかどうかが判断される。合成アルゴリズムでは、関心対象となるべきフィルは一番上の可変アルファフィルである。このチェックは、順序付けられた１つの入力ｚレベルセットについて１つずつ反復して、可変アルファフィルかどうかをテストすることで行われる。好ましい順序は、最初にチェックされる要素が一番上のフィルであるように順序付けられたセットについて反復することである。チェックは、最初の可変アルファフィルが見つかるか、又はセット全体をチェックした結果可変アルファフィルが見つからなかったときに終了する。

【０３３９】

次のステップ１１３で、ステップ１１２で見つかった一番上の可変アルファフィルの上

方にある全てのフィルについて寄与計算が実行される。可変アルファフィルがない場合、この寄与計算は順序付けられた入力セット内にある全てのフィルについて行われる。

【０３４０】

次のステップは、判断１１４である。ステップ１１２で可変アルファフィルが見つかった場合、処理実行はステップ１１５に進む。そうでない場合、処理実行はステップ１１６に進む。

【０３４１】

可変アルファフィルがない場合、ステップ１１６で合成バッファを空に初期化する。

【０３４２】

ステップ１１５で、一番上の可変アルファフィルを含めて、それより下にある全てのフィルのボトムアップ合成が実行される。次いで、ステップ１１７で、ステップ１１５でボトムアップ合成された全てのフィルの合計寄与が計算される。

【０３４３】

ステップ１１８で、トップダウン合成を使用して、一番上の可変アルファフィルよりも上にある全てのフィルが合成される。

【０３４４】

最後のステップ１１９で、出力表示装置上での表示に適した形で出力画素が作成される。

【０３４５】

７．４　図的概要

処理の各段は、図９（ａ）、（ｂ）及び１０に図示されている。これらの図は、それぞれ異なる合成シナリオを示す。

【０３４６】

図９（ａ）は、簡単な例を示す。順序付けられた１つの入力フィルセット７０，７１，７２が示されている。これらは、いずれもソリッドフィルである。寄与計算が行われ、例示的なそれぞれの寄与が６７，６８及び６９に示されている。可変アルファフィルがないため、次のプロセスはトップダウン合成である。その結果は作業バッファ７３に格納され、次いで必要回数だけ反復されて必要な出力画素のラン７４が生成される。この例では、４個の画素のランが作成される。

【０３４７】

第２のシナリオ図９（ｂ）では、ｘ依存フィルを用いる。フィル７５及び７７はいずれもソリッドであり、一方フィル７６はｘ依存である。寄与計算が行われ、その結果が８０、８１及び８２で示されている。可変アルファフィルがないため、実行される次のプロセスはトップダウン合成である。ｘ依存フィル７６に関する画素が生成され、合成バッファ７９に合成により追加される。一方、ソリッドフィル７５及び７７用の色が生成され、作業バッファ７８に合成により追加される。合成バッファ及び作業バッファのそれぞれの寄与が示されている。最後のステップは、合成バッファの各画素に作業バッファの単一の画素を合成により追加することによって出力画素を作成することであり、出力画素８３が生成される。

【０３４８】

図１０に示す最後の例はより複雑であり、可変アルファフィルを含む。フィル８４はソリッドフィル、フィル８５はｘ依存フィル、フィル８６は可変アルファフィル、フィル８７はソリッドフィルである。寄与計算の後、フィル８４及び８５はそれぞそれに関連付けられた寄与値（それぞれ８８，８９で示されている）をもち、一方可変アルファフィル８６を含めて、それ以下の全てのフィルは寄与値をもたない。次に実行されるプロセスは、ボトムアップ合成である。フィル８６及び８７がボトムアップ合成され、その結果が合成バッファ９２（ｃ１）に格納される。ボトムアップ合成の後、１００％からフィル８８及び８９の寄与を差し引くだけで、合成バッファに格納された結果の寄与を求めることができる。次のステップは、トップダウン合成である。一番上のフィル８４はソリッドであり、従って作業バッファ９５に合成により追加される。フィル８５はｘ依存であるため、合

成バッファ９７（ｃ２）に合成により追加される。トップダウン合成のステップの後、作業バッファ９５は単一のフィル８４に関する合成結果を含み、合成バッファ９７（ｃ２）はフィル８５，８６及び８７に関する合成結果を含むことが分かる。最後の段は、出力画素を生成することである。作業バッファの単一の画素が合成バッファの各画素内に合成され、それによって出力画素９６が生成される。

【０３４９】

７．５　寄与計算

図１３のステップ１１３を、図１１を参照しながらより詳細に検討する。このプロセスは、ステップ１１２で見つかった一番上の可変アルファフィルの上方にある順序付けられた１つの入力ｚレベルセット中の全てのフィルに対して実行されることを想起されたい。入力セット中に可変アルファフィルが存在しない場合、このプロセスは入力セット中の全てのフィルに対して実行される。一番上の可変アルファフィルが入力セット中の一番上のフィルである場合、寄与計算を行う必要はない。

【０３５０】

寄与計算で、このプロセスの対象となる各フィル毎に新しい変数が導入される。即ち、フィルのcontribution_valueである。これは、ゼロの寄与を表すために常に初期化されている。例えば、寄与を記憶するために８ビットの整数値を使用することができ、この場合０はゼロの寄与、２５５は１００％の寄与を表す。寄与計算の目的は、各フィルがどれだけ出力画素に寄与しているかを決定することである。

【０３５１】

ここで、２つの変数、accumulated_opacity及びcoverageを導入する。coverageは、処理中のフィルに関連付けられたＡバッファのどのspel要素が現在考慮されているかについての記述である。coverageをＡバッファとして表現するのが好都合である。accumulated_opacityは、現在考慮されているspelが、現フィルの上方のフィルによってどの程度覆い隠されているかについての測度である。以下の議論では、０％のaccumulated_opacityは完全な透明を指し、１００％のaccumulated_opacityは完全な不透明を指す。これら２つの変数は、処理中のcurrent_fillの指示と共に、寄与計算アルゴリズムの状態を記述する１つのパラメータセットを形成する。寄与計算アルゴリズムが読み込み、それに基づいて動作することができるこれら３パラメータの複数のセットを格納するために、当技術分野で周知であるようなスタックの概念が使用される。以下の議論では、プッシュという用語は、スタックの一番上に１つのパラメータセットを追加することを意味し、ポップはスタックの一番上の１つのパラメータセットが取り出されることを意味する。このスタックは、パラメータスタックと称される。

【０３５２】

図４８は、レンダリング中のＡバッファ内のaccumulated_opacityの効果を、上から下へ順に示す。複数のオブジェクトＡ４８００，Ｂ４８０１，Ｃ４８０２及びＤ４８０３はそれぞれ、対応するＡバッファ内の特定のspelに適用可能な不透明度をもつ。１００％の不透明度とは、オブジェクトが完全に不透明であることを意味することに留意されたい。これらのオブジェクトは、０％の不透明度（即ち、１００％の透明度）をもつ背景オブジェクト４８０４の上にレンダリングされている。図４８で、黒の陰影は、Ａバッファの関連するサブピクセルの上の、そのオブジェクトによるカバレージを示す。まず、オブジェクトＡ４８００を背景４８０４に適用すると、結果４８０５が得られる。Ａのうち２つのspelだけが３５％の不透明度をもつため、結果として得られる画素値に対するオブジェクトＡ４８００の寄与は１７．５％に留まる。０％の不透明度は０％の寄与をもたらすことに留意されたい。累積不透明度は、図示されるように、ツリーに分解される。ツリーの枝の個数はマスク内のサブピクセルの個数、この場合は４に限定される。背景４８０４へのオブジェクトＡ４８００の適用の２つの結果４８０６及び４８０７が示されている。次いで、オブジェクトＢ４８０１が適用されるとき、これはＡバッファの下の２つのspelだけに適用される。左下のspel４８０８はオブジェクトＡと交差するので、２つの寄与は累積され、右下のspel４８０９ではオブジェクトＡ４８００が寄与していなかったので、この

段階でのそのspelの結果はオブジェクトＢ４８０１の寄与に限定される。示される計算から、オブジェクトＢの正味の寄与は１６．５％であると判断される。従って、オブジェクトＢの適用後、spelはそれぞれ４８１０で６１％、４８１１で３５％、４８１２で４０％、４８１３で０％の不透明度をもつ。オブジェクトＣ４８０２は１００％の不透明度をもち、右の２つのspelに影響し、それによって結果４８１４，４８１５，４８１６及び４８１７が得られ、これらは画素に対する４０％の寄与を示している。オブジェクトＣは完全に不透明であるため、オブジェクトＣと交差する可能性がある、それよりも下にあるオブジェクト（オブジェクトＤ）を考慮する必要はない。従って、右側のspelについては、ツリーの枝はオブジェクトＣで止まる。オブジェクトＤ４８０３は１００％の不透明度をもち、４８１８，４８１９，４８２０及び４８２１で見られるように、画素に対して２６％寄与している。図４８から、アクティブ図形オブジェクトの不透明度から求められた累積寄与は、合計すると１００％になることが分かる。

【０３５３】

図４８のＡバッファについて、結果として得られる寄与は下記のように表すことができる。

【０３５４】

【表５】

表5

A=35% B=0% C=0% D=65%	A=0% B=0% C=100% D=0%
A=35% B=26% C=0% D=39%	A=0% B=40% C=60% D=0%

【０３５５】

図４８から、レベルの数がサブピクセルマスク（Ａバッファ）のサイズによって決定されるようにツリーがトラバースされることは明らかである。更に、オブジェクトＣ４８０２に関して見られるように、ツリーの任意の枝のトラバースは、不透明度が１００％に達し、それ以上そのサブピクセルに寄与することができなくなった時点で停止することができる。

【０３５６】

ここで、実際の処理に戻ると、図１１のプロセスの最初のステップ９８で、パラメータの初期化が行われる。current_fillは、入力セットの一番上のフィルになるように初期化される。accumulated_opacityは０％に初期化され、coverageはＡバッファの全体カバレージを表すように初期化される。このパラメータの最初のセットは、パラメータスタック上にプッシュされる。更に、処理されるべき全てのフィルの全てのcontribution_valueは０に初期化される。

【０３５７】

主プロセスループの最初のステップ９９で、パラメータスタックから次の１つのパラメータセットをポップする。これに関して、ステップ１００で、空の結果かどうかがチェックされる。スタック上にそれ以上のパラメータセットが存在しない場合、寄与計算は終了する。１つのパラメータセットが無事スタックからポップされた場合、次のステップ１０１で、現accumulated_opacity及びcoverageからcurrent_fillの寄与を計算し、これをcurrent_fillの総contribution_valueに加算する。この計算は、以下の式に従って実行される。

【０３５８】

current_contrib＝current_alpha＊（1－accumulated_opacity）＊coverage_ratio
但し、
current_alpha＝current_fillのアルファ値
accumulated_opacity＝現パラメータセット内で与えられるaccumulated_opacity
coverage_ratio＝active_ratio（coverage∩Ａバッファ）
A-buffer＝current_fillに関連付けられたＡバッファ
Active_ratio（A-buffer）＝所与のＡバッファ内のアクティブspelの数の、所与のＡバッファ内のspelの総数に対する比率
である。

【０３５９】

主式の全ての要素は、［０～１］の範囲をもつ実数として表される値を意味することに留意されたい。これは、この式の表記を簡単にするためであり、これらの要素についての他の表現手段に対応できるようにするためにこの式をどのように操作すべきかについては、当業者には明らかであろう。current_coverage_ratioの場合、［０～１］の範囲は、coverageのＡバッファとcurrent_fillのＡバッファとの交差点での「アクティブな」spelの個数の、Ａバッファ内のspelの総数に対する比を表している。

【０３６０】

この計算されたcurrent_contrib値は、次いでcurrent_fillのcontribution_valueに加算される。

【０３６１】

図１１のプロセスの次のステップ１０２は、次に処理すべきフィルがあるかどうかをチェックする。current_fillが入力セットの最後のフィルであるか、又は次のフィルが可変アルファフィルである場合、このチェックの結果は負となる。プロセスの実行は、次いでステップ９９に戻る。チェック結果が正の場合、このcurrent_fillは次のフィルを識別するように更新される。

【０３６２】

次のステップ１０３で、１つのパラメータセットがパラメータスタック上にプッシュされる。これらのパラメータとは以下のものである。

【０３６３】

current_fill←current_fill
accumulated_opacity←accumulated_opacity
coverage←coverage∩{（coverage∩A-buffer）/}
ここで「/」は、コンプリメントを示す。

【０３６４】

次のステップ１０４は、現在のパラメータセットを以下のように更新する。

【０３６５】

current_fill←current_fill
accumulated_opacity←（１－current_alpha）＊（１－accumulated_opacity）
coverage←coverage∩A-buffer
accumulated_opacity及びcurrent_alphaは、［０～１］の範囲をもつ実数として表される値を意図していることに留意されたい。

【０３６６】

coverageに適用される操作の図形表示が、図１２（ａ）～（ｄ）に示されている。Ａバッファ１０７は、coverageを表す例を示している。第２の例示Ａバッファ１０６は、フィルのＡバッファを表している。ステップ１０３に示されるcoverageＡバッファを新しく生成するための操作が適用されると、その結果、Ａバッファ１０８が得られる。ステップ１０４に示されるcoverageＡバッファを新しく生成するための操作が適用されると、その結果、Ａバッファ１０９が得られる。

【０３６７】

この更新が終わると、十分な度合いの不透明度が得られたかどうかを調べるためにaccu

mulated_opacityがテストされる。閾値の例を挙げると、完全な不透明の２５５／２５６である。十分な不透明度に達した場合、プロセスの実行はステップ９９に戻る。そうでない場合、プロセスの実行はステップ１０１に進む。

【０３６８】

７．６　ボトムアップ合成

再び図１３に戻ると、ステップ１１４で入力セット中に可変アルファエッジが検出された場合、ボトムアップ合成のステップ１１５が実行される。

【０３６９】

ボトムアップ合成プロセスは、一番下のフィルから一番上の可変アルファフィルまでの順序でフィルを合成する。以下の議論で、「次」のフィルに言及する場合、これはcurrent_fillのすぐ上のフィルを指す。

【０３７０】

ボトムアップ合成プロセスは、フィル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じだけの画素数を格納できなければならない。例えば、このバッファは合成バッファの半分を利用して実現することもできる。これによって、合成バッファのサイズが半分になり、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、合成バッファ６１２の２回目の適用が必要となる。

【０３７１】

図６を参照すると、ボトムアップ合成プロセスが図示されている。その最初のステップ５０で、背景フィルを合成バッファ内に合成する。これは、合成バッファに合成により追加される最初のフィルであることが保証されているため、特別な計算は不要である。フィル色は、単に合成バッファ内へ複製される。次のステップの１回目の実行のために、current_fillは背景フィルであると見なす。

【０３７２】

次のステップ５１は、ボトムアップ合成を必要とするフィルが他にあるかどうかを調べるテストである。current_fillが一番上の可変アルファフィルである場合、ボトムアップ合成すべき更なるフィルは存在せず、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に進む。

【０３７３】

ステップ５５でのボトムアップ合成結果の寄与の判断は、以下のアルゴリズムに従って実行される：

total_contrib＝０
一番上の可変アルファフィルより上の全てのフィルについて
total_contrib＝total_contrib＋fill［n］.contrib
bottom_up_contrib＝１－total_contrib

但し、

fill［n］．contrib＝図１３のフローチャートのステップ１１３で計算される、一番上の可変アルファフィルより上のｎ番目のフィルの寄与値である。

【０３７４】

ボトムアップ合成結果の寄与値が計算されると、寄与値は以下のアルゴリズムに従って乗算されて合成バッファに入れられる。

【０３７５】

合成バッファ内の全ての画素について
cb［n］＝cb［n］＊bottom_up_contrib

但し、cb［n］＝合成バッファのｎ番目の画素

である。

【０３７６】

スカラ値（bottom_up_contrib）にカラー値（cb［n］）を乗じる上記の式は、カラーの全ての成分のスカラによる乗算を指す。

【０３７７】

ステップ５１で、ボトムアップ合成すべき更なるフィルが存在すると判断された場合、次いでステップ５２で、current_fillが次のフィルになるように設定される。次いでステップ５３で、current_fillがｘ依存フィルであるかどうかを調べるためにテストされる。フィルがｘ依存でない場合、ステップ５４に従って、フィル色が作業バッファ内に生成される。次いで、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内に繰り返し合成により追加される。

【０３７８】

合成バッファ内の全ての画素について、

cb［n］＝cb［n］＋work_alpha＊（cb［n］－work）

但し、

cb［n］＝合成バッファのｎ番目の画素

work_alpha＝作業バッファ内の画素のアルファ

work＝作業バッファ内の画素

である。

【０３７９】

ボトムアップ合成の実行は、次いでステップ５１に戻る。

【０３８０】

フィルがｘ依存である場合、ステップ５６で、ランの全ての画素がフィル生成バッファ内に生成される。次いで、ステップ５７で、フィル生成バッファが以下のアルゴリズムを使用して合成バッファに合成により追加される。

【０３８１】

合成バッファ及びフィルバッファの全ての画素について

cb［n］＝cb［n］＋fb［n］．alpha＊（cb［n］－fb［n］）　　式（１）

但し、cb［n］＝合成バッファのｎ番目の画素

fb［n］＝フィルバッファのｎ番目の画素

fb［n］．alpha＝フィルバッファのｎ番目の画素のアルファ

である。

【０３８２】

７．７　トップダウン合成

図１３のステップ１１８で示されるトップダウン合成プロセスを、図５５に関連して詳細に説明する。これは、ステップ１１２で判断される一番上の可変アルファフィルより上方にある全てのフィルに適用される。入力セット内に可変アルファフィルが存在しない場合、トップダウンアルファプロセスは入力セット内の全てのフィルに適用される。

【０３８３】

トップダウンプロセスでは、ステップ１１３で計算されるフィルに対してcontribution_valueを利用する。トップダウン合成プロセスでは、フィルを処理する順序は最終結果の点からは重要ではないことに留意されたい。以下の例では、フィルは入力セット内に現れる順序で処理され、セット内の一番の上のフィルから処理が開始される。

【０３８４】

図５５で見られるように、トップダウン合成プロセスの最初のステップ６０１で、処理すべきフィルが他にあるかどうかが判断される。処理すべき次のフィルがない場合、又は次のフィルが、ステップ１１２で判断される一番上の可変アルファフィルである場合、プロセスはステップ６０７に進む。

【０３８５】

処理すべき更なるフィルがある場合、ステップ６０２で、現フィルが可変current_fillに格納される。次のステップ６０４で、current_fillのcontribution_valueが０よりも大きいかどうかがチェックされる。このテストが偽を返す場合、実行はステップ６０１に進む。

【０３８６】

contribution_valueが０よりも大きい場合、実行はステップ６０３に進む。ステップ６

０３で、current_fillがｘ依存フィルを表しているかどうかがテストされる。このフィルがｘ依存でない場合、実行はステップ６０６に進む。そうではない場合、即ちステップ６０３が真を返す場合、実行はステップ６０５に進む。

【０３８７】

非ｘ依存フィルは、ステップ６０６で次のように処理される。フィルの色は、以下のアルゴリズムに従って生成され、作業バッファに合成により追加される：

wb＝wb＋current_fill．contrib＊current_fill．color　　式（２）

但し、wb＝作業バッファ

current_fill．contrib＝current_fillのcontribution_value

current_fill．color＝current_fillのカラー値

である。

【０３８８】

ｘ依存フィルは、ステップ６０５で次のように処理される。ｘ依存フィルの各画素が生成されると、それは以下のアルゴリズムに従って直ちに合成バッファに合成により追加される：

cb［n］＝cb［n］＋current_fill．contrib＊gp

但し、cb［n］＝合成バッファのｎ番目の画素

current_fill．contrib＝current_fillのcontribution_value

gp＝現在検討中の生成されたフィル画素

である。

【０３８９】

ステップ６０５又はステップ６０６のいずれかが完了すると、実行はステップ６０１に戻る。

【０３９０】

ステップ６０１で、処理すべき次のフィルがない場合、実行はステップ６０７に進む。このステップでは、作業バッファは、最終結果を作成するために合成バッファに合成により追加される。このステップは、以下のアルゴリズムに従って行われる：

合成バッファ内の各画素について

cb［n］＝cb［n］＋wb

但し、cb［n］＝合成バッファのｎ番目の画素

wb＝作業バッファ

である。

【０３９１】

トップダウン合成はこれで完了し、composite_bufferには出力ランについて完全に合成された画素が含まれる。

【０３９２】

７．８　他の合成手法

合成プロセスについての代替手法を以下に説明する。この代替案により、可変アルファ層に遭遇したとき、潜在的に大きなボトムアップ合成を行う必要がなくなる。その代わりに、可能ならいつでもトップダウン合成プロセスを使用し、それによってトップダウン合成が提供するメリットを最大限引き出す。

【０３９３】

先に述べたように、寄与計算プロセスは、一番上の可変アルファ層に遭遇したときに停止する。この代替合成手法は、この一番上の可変アルファ層よりも下に層が存在するかどうかをチェックする。そうである場合、コンポジタが繰り返し呼び出され、一番上の可変アルファ層の下の層がこの新しい呼出しの一番上の層となる。この新しい呼出し、及びその後の全ての呼出しで、各層が通常通り処理され、完全に不透明になったとき、画素が通常通り作業バッファ及び合成バッファに生成により追加される。任意の反復的呼出しがその親に戻った後、合成バッファ内の結果が、親の中で見つかった可変アルファ層と合成される。次いで、作業バッファ及び合成バッファ内の結果の寄与が図６のステップ５５の通

りに計算され、両バッファ内の画素値がそれに応じて調整される。

【０３９４】

画素のランについて、これらの合成手法の１つの一般化は以下の通りである。

【０３９５】

（ｉ）層を、可変アルファ層によって互いに隔てられたグループに分け、

（ｉｉ）グループ画素値を決定するために、各グループについてトップダウン合成を行い、それにより元の層を、可変アルファ層によって互いに隔てられたグループ画素値を含む、より少ない数のグループに分解し、

（ｉｉｉ）任意の１グループの寄与が１００％（即ち完全に不透明）の所定閾値内にあるとき、その閾値グループの下にある全ての可変アルファ層及びグループを寄与しないものとして無視することができ、

（ｉｖ）ラン内の各画素について、対応する画素値を決定するために、寄与するグループ及び可変アルファ層のボトムアップ合成を実施する。

【０３９６】

合成プロセスの更なる実装が、２つのバッファを以下のように使用することによって実現できる。

【０３９７】

（ｉ）第１のバッファを使用して、一番上の層から始めて、第１の可変アルファ層を含めて、第１の可変アルファ層までの全ての層についてトップダウン合成を行い、

（ｉｉ）第２のバッファを使用して、次の可変アルファ層を含めて、この層までの全ての層についてトップダウン合成を行い、

（ｉｉｉ）第２のバッファの上に第１のバッファを合成し、その結果を第１のバッファ内に保持し、

（ｉｖ）全ての層が合成されるまでステップ（ｉｉ）及び（ｉｉｉ）を繰り返す。前と同様、ある可変アルファ層の上の任意の１つの層グループの寄与が１００％（即ちほぼ又は完全に不透明）の所定閾値内にあるとき、その閾値グループの下にある全ての可変アルファ層及びグループを寄与しないものとして無視することができ、そこで合成を終えることができる。

【０３９８】

上述した更なる実装は、可変アルファ層の累加がそれよりも低いグループ内の層の寄与を正しく表さないため、画像を正確には表現しないことに留意されたい。例えば、第１の可変アルファ層を除き、第１の可変アルファ層までのそれよりも上の全ての層の寄与が８０％である場合、第１の可変アルファ層及びそれよりも下の全ての層の寄与は２０％のはずである。第１の可変アルファ層を、それよりも上の層と合成することで、寄与のバランスが歪められ、より速いが不正確な合成が得られる。そのような結果は、最終的な画像レンダリングには望ましくない可能性があるが、全ての層についての正確な詳細を必要としない、画像構成のプレビュー段階では十分である可能性がある。

【０３９９】

所定閾値は、一般に、合成結果についての所望の精度、及び処理すべき動作の数に依存する。閾値は、例えばあるカラーチャネルの最下位ビット以内であってもよく、これは８ビット色の場合、２５５／２５６（９９．６％）の寄与となる。他の閾値を選ぶこともできる。

【０４００】

図６１Ａは、ある画素のラン内の多数の層のトップダウン合成に対して実施できる一般化を示す。図示されるように、それぞれ異なるタイプをもつ多数の層６１０１～６１０７が示されている。タイプの１つは、その性質上、一定であるアルファ成分を含むことができる一定色レベルである。レベル６１０１，６１０２，６１０４及び６１０６がこれに該当する。別のタイプのレベルは、可変色レベルであるが一定のアルファをもつレベルである。一例を挙げると、画素のラン全体に渡る線形混色がある。ここでも、アルファ値が一定であるため、単一の可変色と見なすことができる。このレベルの例がレベル６１０３及

び６１０５である。レベル６１０７で例示される、最後のタイプのレベルは、可変色及び可変アルファを共に含むレベルである。その性質上、このレベルを可変アルファレベルとして一般化することができ、画素のスパンのレンダリングの際には、そのレベルの下の全てのレベルの寄与を変更するように動作する。

【０４０１】

図６１Ａの一般化とは、トップダウン合成を用いて、ある可変アルファレベルより上の一定アルファをもつ全てのレベルをトップダウン合成して、単一の色レベルを提供することができることである。この単一の色レベルは、元のソースレベル次第では可変色レベルとなることもある。この一般化を、レベル６１０１～６１０６のトップダウン合成を表す新しい色レベル６１０８によって示す。レベル６１０８は、前の可変アルファレベル６１０７の上に置かれる。

【０４０２】

従って、図６１Ａは、一定のアルファ成分をもつ任意の数のレベルをトップダウン合成して、一定アルファをもつ単一のレベル（例えば６１０８）を提供することができ、次いでこの単一のレベルを可変アルファをもつレベルの上に合成することができることを示している。

【０４０３】

図６１Ｂは、可変色の層のグループ６１１０，６１１２及び６１１４がそれぞれ可変アルファの層６１１１及び６１１３で隔てられている、複数の層を含む合成スタックに対して、この一般化がどのように適用できるかを示す。このような構成では、可変色及び一定アルファをもつ各層グループは、一定アルファをもつ対応する単一の層を作成するためにトップダウン合成することができ、これらの新しい層は６１１５，６１１６及び６１１７として示されている。これらの新しい各層は、次いで、画素のランの正確な合成を提供するために、ボトムアップ合成でそれぞれの間の可変アルファ層６１１１及び６１１３と合成することができる。最も重要なことには、合成操作を可能な限り多くのトップダウン合成に分割することによって、画素のラン全体に渡る合成操作の数が最小限に抑えられる。この点について、任意の可変アルファレベルの動作のせいで、合成操作には画素のラン内の各画素での合成が必要となる。従って、トップダウン合成と、その後に続くボトムダウン合成を組み合わせることで、以前には得られなかった最適化が提供される。

【０４０４】

この原理の拡張型を図６２に示す。図中、一定色、可変色及び可変アルファ層の組合せを含む様々な層６２１０～６２１２を使用して、画素のスパンに対する合成スタック６２００が形成される。この変形形態によれば、合成スタック６２００は、画素のランの画素値を提供するためにトップダウン合成操作だけを使用して合成される。実質的に、スタック６２００は、それぞれ可変アルファ層で線引きされたセクションに分けられる。この構成では、第１の可変アルファ層を含めてその可変アルファ層までの第１の層グループ、即ち層６２０１～６２０４のグループに対して、トップダウン合成６２１３が行われる。この合成の結果は、第１のバッファ６２１４内に保持される。次いで、次の可変アルファ層を含めてその可変アルファ層までの次の層グループ、即ち層６２０５～６２０７のグループに対して、トップダウン合成６２１５が行われる。この合成６２１５の結果は、第２のバッファ６２１６内に保持される。次いで、第１のバッファ６２１４を第２のバッファ６２１６の上に合成することによって、それぞれのバッファに含まれる合成結果が合成され、この合成結果が、図６２の６２１７で示されるように、第１のバッファに戻されてそこに書き込まれる。次いで、次の層グループ６２０８～６２１１がトップダウン合成６２１８され、この合成結果が再び第２のバッファ６２１９に格納される。第１のバッファ６２１７が再び第２のバッファ６２１９の上に合成され、この合成結果６２２０が第１のバッファに格納される。最後に、図６２の例で、一定色層６２１２のみを含む最後の層グループがトップダウン合成６２２１される。この場合、層グループは単一の一定色のみを含むため、合成操作は事実上ヌルであり、層６２１２の値が第２のバッファ６２２２内に複製される。バッファが再び合成され、最終結果６２２３が第１のバッファに保持される。

【０４０５】
図６２の記述から、トップダウン合成だけに依存することは、必ずしも最終合成結果がその特定の画素のスパンについて完全に正確であることを保証するものではないことが理解されよう。しかし、本発明者らは、確定できる精度レベルで最終合成結果が作成できるように、合成操作に関する所望の精度レベルが確保できる可能性があると判断した。これは、画質を犠牲にする可能性があるものの、非常に速い合成が必要である場合に有用である。この例としては、例えばグラフィックアーティストによる、画像製作プロセスのプレビューが含まれるかもしれない。他の実装としては、互いに異なる処理能力をもついくつかの異なるプラットフォーム上で合成処理を行うことができる場合が考えられる。特に、合成に関する時間的な制約が設定されている場合（例えば、リアルタイム動作で）、システム要求を満たすために、合成スタックの複雑さを用いて、特定タイプの合成（例えば、図６１Ａの正確であるが遅い合成、又は図６２の速いが恐らくは正確さに欠ける合成）をトリガすることができる。
【０４０６】
更に別の代替手法を図６４に示す。この手法は、ボトムアップ合成の必要性をなくすことにより、先に記述した構成に対する改良型を提供する。トップダウン合成を排他的に使用することによって、不透明度が１００％近くまで十分に分解された場合にプロセスが早期終了するという、トップダウン合成のメリットの更なる利用が可能になる。この代替手法の代償は、図９及び６２に示した手法に比べて、作業バッファ（ここで「作業バッファ２」と称する）が追加されることである。この新しい作業バッファは、作業バッファ１（例えば、７８）と同じ要素をもつが、各要素はｒｇｂではなく、ｒｇｂｏ（ｒ＝赤、ｇ＝緑、ｂ＝青及びｏ＝不透明度）である。
【０４０７】
動作に際して、合成スタックが、互いに可変不透明度層によって隔てられたセクションに区切られる。前の実装と同様、一番上のセクションに対して作業バッファ０（例えば、７３）及び作業バッファ１が使用される。次いで、作業バッファ１が第１の可変不透明度層の上に合成され、その結果が作業バッファ２に格納される。次いで、作業バッファ０及び作業バッファ１が、一番上のセクションに適用されたのと同じように次に低いセクションに適用される。ただしその最後に、
（ａ）作業バッファ２を作業バッファ１の上に合成した結果が作業バッファ２に格納され、
（ｂ）作業バッファ２を次の可変不透明度層の上に合成した結果が作業バッファ２に格納される。
【０４０８】
図６４に示す合成手法６４００では、ステップ６４０１で、合成スタックから固定色が合成され、それによって作業バッファ０内に固定色が形成される。ステップ６４０２で、作業バッファ１内に２つの可変色が合成される。ステップ６４０３で、作業バッファ０が作業バッファ１の上に合成され、その結果が作業バッファ２に格納される。これらのステップは、実質的に前の手法のステップに対応する。ステップ６４０４で、作業バッファ１が第１の可変不透明度層の上に合成され、その結果が作業バッファ２に格納される。ステップ６４０５で、前と同様に次の２つの固定色が合成されるが、ただしそれらの寄与は、作業バッファ０内にローカルに保持される。次いで、ステップ６４０６で、作業バッファ２が作業バッファ０の上に合成され、その結果が作業バッファ２に格納される。次いで、ステップ６４０７で、可変不透明度色である作業バッファ２が次の可変不透明度層の上に合成され、その結果が作業バッファ２に格納される。ステップ６４０８～６４１０は、実質的にステップ６４０１～６４０３の繰返しである。ステップ６４１１で、作業バッファ２が作業バッファ１の上に合成され、その結果が再び作業バッファ２に格納される。ステップ６４１２で、作業バッファ２が次の可変不透明度層の上に処理され、その最終結果が作業バッファ２に保持されてこの合成プロセスは完了する。
【０４０９】

従って、この手法は、合成スタックのトップダウントラバースを維持しながら、可変不透明度層に対応できることが分かる。方法６４００の早期終了は、先に述べたように、累積不透明度がある所定閾値に達したときに実現できる。累積不透明度を計算する場合、可変不透明度層の不透明度寄与は、問題なくゼロと見なすことができる。別法として、各可変不透明度層の不透明度寄与は、例えばその可変不透明度層内の最小の不透明度をもつ画素を探すことによって得られる、最小の値と見なすことができる。この原理の更なる拡張型は、画素の累積不透明度が個別に所定閾値に達し、それによって早期終了が実現されるように、スタックの合成全体に渡って不透明度寄与をベクトルとして（例えば、画素毎に）追跡することである。

【０４１０】

７．９　トップダウンの利点

上述したトップダウン合成技法は、ボトムアップ技法よりも好ましい。トップダウン技法の方が、２つの層を合成するために必要な計算の量という点で、より効率的である。これは、セクション７．６及び７．７で提供された式１及び２を参照すると理解されよう。

【０４１１】

トップダウン合成はまた、その後の合成操作が出力に対して目に見える効果がないと判断され次第、ただちに合成プロセスが終了するようにできるという重要な利点をもつ。この「早期終了」は、実際には頻繁に起こり、パフォーマンスの向上をもたらす。

【０４１２】

８．画素生成

図５６の画素生成モジュール６１１は、単一のＺレベルについて画素値のランを生成することに関係する。

【０４１３】

合成モジュール６１２は、出力画素を生成するために、ｚレベルフィルのカラー値を組み合わせる。合成モジュール６１２が組み合わせるソースカラー値は、明示的に定義されており、或は何らかの形のフィル記述から生成されなければならない。ソリッド色のフィルのＺレベルは、すぐに合成できる状態のソースカラー値を明示的に定義する。勾配（線形又は径方向）及びビットマップ画像フィルなどの複合フィルタイプのＺレベルの場合、ソースカラー値が、生成すべき画素のラスタ空間位置に依存するように生成される必要がある。複合フィルのｚレベルのソースカラー値を決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によって実行される。

【０４１４】

画素生成モジュール６１１の処理フローの概略を図３９に示す。ステップ３９３で、モジュール６１１に、入力としてラスタ空間内の開始及び終了位置が、必要なフィル記述を含む複合Ｚレベルの参照と共に提供される。ステップ３９４及び３９５で判断されるＺレベルのフィルタイプに応じて、コンポジタが以下のいずれかに対するカラー値のランを生成する：

ステップ３９６で線形勾配、
ステップ３９７で径方向勾配、又は
ステップ３９８でビットマップフィル

生成されたカラー値は、合成モジュール６１２によってすぐに合成できる状態で合成バッファ又はフィル生成バッファ内に格納される。各フィルタイプの色生成の詳細については、次に説明する。

【０４１５】

８．１　線形勾配画素生成

ｚレベルは、線形勾配フィルをもつことができる。当技術分野では、線形勾配フィルはしばしば線形混色と呼ばれる。線形勾配フィルでは、ある特定の仮想線から陰影を付けるべき点までの垂直距離によって決まるカラー値を使用して表面の陰影付けを行う。

【０４１６】

線形勾配は、勾配ルックアップテーブルを使用して記述することができる。勾配ルック

アップテーブルは、あるインデックス値の範囲にわたってある勾配に対するカラー値を定義する。勾配ルックアップテーブルは、順序付けられた１つ又は複数の勾配ルックアップテーブルエントリのセットとして実装することができる。各勾配ルックアップテーブルエントリは、インデックス値と対応するカラー値とを含む。勾配ルックアップテーブルエントリは、インデックス値の最も低いものから最も高いものへと順序付けられている。

【０４１７】

図２５を参照すると、勾配ルックアップテーブル２６０が示されている。勾配ルックアップテーブル２６０のインデックス値の範囲は、０から２５５を含めて２５５までであることが分かる。勾配ルックアップテーブル２６０は、２つの勾配ルックアップテーブルエントリ２６１及び２６２を含む。第１の勾配ルックアップテーブルエントリ２６１は、インデックス値６３で白のカラー値を定義する。第２の勾配ルックアップテーブルエントリ２６２は、インデックス値１９１で黒のカラー値を定義する。

【０４１８】

勾配ルックアップテーブルの２つのエントリ間のインデックス値について、それに対応するカラー値が、その両側の２つの勾配ルックアップテーブルエントリのカラー値の線形補間を使用して計算できる。全ての勾配ルックアップテーブルエントリの最小インデックス値よりも低いインデックス値、又は全ての勾配ルックアップテーブルエントリの最大インデックス値よりも大きいインデックス値については、それに対応するカラー値を、それに最も近いインデックスをもつ勾配ルックアップテーブルエントリのカラー値を直接使用して決定することができる。

【０４１９】

例えば、図２５で、２つのルックアップテーブルエントリ２６１と２６２の間のインデックス値に対応するカラー値が、これらのエントリの白カラー値及び黒カラー値の線形補間を使用して決定される。この例を続けると、第１のルックアップテーブルエントリ２６１のインデックス値、例えばインデックス値５０よりも低いインデックス値の場合、関連付けられるカラー値は最も近いルックアップテーブルエントリのカラー値（従って、白カラー値）である。逆に、最後のルックアップテーブルエントリ２６２のインデックス値、例えばインデックス値２５５よりも高いインデックス値の場合、関連付けられるカラー値は最も近いルックアップテーブルエントリのカラー値（従って、黒カラー値）である。

【０４２０】

勾配フィルからラスタ画素カラー値を生成する場合、画素位置に対応する、勾配ルックアップテーブルのインデックスを計算する必要がある。これを行うために、画素のラスタ空間座標位置の勾配空間内への変換が行われる。勾配空間をルックアップテーブルインデックス範囲内にマッピングするためには、勾配空間表面を規定することが有用である。勾配空間表面内の各点は、特定の勾配ルックアップテーブルインデックスにマッピングされる。ラスタ画素のカラー値は、インデックス値及び勾配ルックアップテーブルを使用して決定できる。

【０４２１】

線形勾配の場合、勾配空間表面は、その勾配空間表面内の点の水平位置のみによって決まるやり方で、点をインデックス値にマッピングすることができる。これは、線形勾配の場合、勾配空間位置の垂直成分を計算する必要がないことを意味する。ラスタ空間の線形勾配の外観を変更するには、勾配の位置、スケール、回転などのパラメータを改変することが望ましいことがある。このような効果を得るために、勾配からラスタ空間への変換を使用して、線形勾配空間表面をラスタ空間内に変換することができる。

【０４２２】

図２５はまた、勾配空間原点に中心を合わせた、幅３２７６８をもつ四角形の勾配空間表面２６３の例を示す。この勾配空間表面全域のカラー値は、上述した線形勾配用の技法を使用して示され、従ってカラー値は勾配の水平空間位置のみによって決まる。図２５では、この勾配空間表面２６３が、出力のためにラスタ空間装置２６４上にマッピングされている。このマッピングは、勾配空間表面に対してある程度の回転及び拡大縮小を伴うこ

とが理解できる。

【０４２３】

線形勾配フィルは、勾配ルックアップテーブル中のインデックスを増分的に追跡することによってレンダリングできる。この技法では、勾配からラスタ空間への変換によって決まる３つのパラメータが事前計算される。図３２を参照すると、勾配ルックアップテーブル及び勾配からラスタ空間への変換を含む線形勾配記述が与えられると、出力画素カラー値を計算する前に、ステップ３１１で以下のものを計算することができる。

【０４２４】

ラスタ空間水平方向への画素ステップ当りの勾配空間水平位置の変化　∂ｕ／∂ｘ

ラスタ空間垂直方向への画素ステップ当りの勾配空間水平位置の変化　∂ｕ／∂ｙ

及び

ラスタ空間原点に対応する勾配空間位置ｕo。

【０４２５】

これらの事前計算値は、勾配からラスタ空間への変換によって決まり、再計算は変換が改変される場合にのみ必要である。

【０４２６】

これらの勾配追跡パラメータの図形表現が図２７に示されている。図２７で、ラスタ空間Ｘ－Ｙ２６９のラスタ出力装置の表示が示されている。勾配空間Ｕ－Ｖ２７０内で規定された勾配がラスタ装置空間２６９内にラスタ化されている。ラスタ装置のラスタ走査線２６７は、走査線内の開始画素２６８から開始される、勾配フィルされた画素のランを含むためのものである。正のラスタ空間水平方向への１ラスタ画素ステップについて、対応する勾配空間水平位置の変化（∂ｕ／∂ｘ）２７１が示されている。正のラスタ空間垂直方向への１ラスタ画素ステップについて、対応する勾配空間水平位置の変化（∂ｕ／∂ｙ）２７３が示されている。図には、勾配空間垂直位置に対応する同様のパラメータ２７２、２７４も示されている。これらの値の計算は、線形勾配の場合は不要である。

【０４２７】

これで、フィル生成バッファに出力すべきランを形成する１つ又は複数の連続画素の線形勾配画素カラー値の生成に進むことができる。生成されるべきランの一番左のラスタ画素のラスタ空間座標位置（ｘstart，ｙstart）が与えられると、図３２のステップ３１２で、まず勾配空間水平位置ｕを初期化する必要がある。図２７で、生成されるべきランの一番左のラスタ画素の例が２６８で示される。初期ｕ値は、以下の式を使用して計算することができる。

【０４２８】

ｕ＝ｕo＋ｘstart×（∂ｕ／∂ｘ）＋ｙstart×（∂ｕ／∂ｙ）

従って、ｕ値を直接使用することによって、勾配ルックアップテーブル内への実際のインデックスを決定することが可能である。ステップ３１３で、そのようなインデックス値を使用して、勾配ルックアップテーブルからの対応するカラー値を、先述したように決定することができる。

【０４２９】

画素のカラー値を生成した後、ラン内に線形勾配から生成すべき画素カラー値が他にもまだあるとステップ３１０で判定された場合、ステップ３１５で、勾配空間水平位置の値ｕを以下の漸化式によって増分することができる。

【０４３０】

ｕ＝ｕn-1＋（∂ｕ／∂ｘ）

このプロセスが、ラン内の必要な全ての画素が生成されるまで繰り返される。その後、ステップ３１４で判断されるように、同じ線形勾配を使用したレンダリングを必要とするランが更に存在する場合、この同じプロセスを繰り返して、まずステップ３１２でレンダリングすべき新しいランの開始画素用に初期勾配空間水平位置ｕを再計算し、次いでステップ３１３でそのランの各画素用に画素カラー値を生成するプロセスを繰り返すことができる。

【０４３１】
８．２　径方向の勾配画素生成
ｚレベルは、径方向勾配フィルをもつことができる。径方向勾配フィルは、径方向混色とも呼ばれる。径方向勾配フィルは、径方向勾配のある特定の中心点から陰影を付けるべき点までの距離によって決まるカラー値を使用した表面の陰影付けと考えることができる。径方向勾配画素を生成するプロセスは、先に述べた線形勾配画素を生成するプロセスと類似している。
【０４３２】
線形勾配の場合と同様、径方向勾配は、順序付けられた１つ又は複数の勾配ルックアップテーブルエントリのセットとして実装される勾配ルックアップテーブルを使用して記述することができる。径方向勾配はまた、勾配空間表面も使用する。この場合も、勾配空間表面内の各点が、特定の勾配ルックアップテーブルインデックスにマッピングされる。この場合も、ルックアップテーブルに明示的に定義されていないインデックスの色を取得するために、２つの勾配ルックアップテーブルエントリのカラー値を線形に補間することができる。
【０４３３】
径方向勾配の場合、勾配空間表面は、勾配空間表面の中心点からある点までの距離によって決まるやり方で、その点をあるインデックス値にマッピングする。ラスタ空間内の径方向勾配の外観を変えるために、中心点や径方向勾配のスケールなどのパラメータを改変することができる。このような効果を得るために、勾配からラスタ空間への変換を使用して、径方向勾配空間表面をラスタ空間内に変換することができる。
【０４３４】
従来のやり方で径方向勾配をレンダリングするのは、コンピュータ的により負担が大きい。一般に、各画素の座標を勾配空間内に変換し、次いで勾配空間の中心から勾配空間点までの距離を計算し、最後に対応するカラー値をルックアップすることが必要である。しかし、径方向勾配からラスタ画素の連続するシーケンスを生成する場合、勾配ルックアップテーブルインデックスの計算に増分的手法を使用することで、パフォーマンスを向上させることができる。
【０４３５】
図４２を参照すると、勾配ルックアップテーブル及び勾配からラスタ空間への変換を含む径方向勾配記述が与えられると、出力画素カラー値を計算する前に、ステップ４１４で以下のものを計算することができる。
【０４３６】
（ｉ）ラスタ空間水平方向への画素ステップ当りの勾配空間水平位置の変化　∂ｕ／∂ｘ
（ｉｉ）ラスタ空間水平方向への画素ステップ当りの勾配空間垂直位置の変化　∂ｖ／∂ｙ
（ｉｉｉ）ラスタ空間垂直方向への画素ステップ当りの勾配空間水平位置の変化　∂ｕ／∂ｙ
（ｉｖ）ラスタ空間垂直方向への画素ステップ当りの勾配空間垂直位置の変化　∂ｖ／∂ｙ
（ｖ）ラスタ空間原点に対応する勾配空間位置（ｕｏ，ｖｏ）
これらの事前計算値は、勾配からラスタ空間への変換によって決まり、変換が改変される場合にのみ再計算を必要とする。これらの事前計算値を使用することによって、勾配ルックアップテーブルインデックスの２乗（$index^2$）の増分的な追跡ができる。ラスタ走査レンダリングの特定の実装において、indexは、径方向勾配の中心点からレンダリングされるべき画素位置への距離に等しい。
【０４３７】
これらの勾配追跡パラメータの図的表現が、図２７に示されている。図は線形勾配フィルを示しているが、その内容を径方向勾配の生成に同様に適用できることに注意すべきで

ある。図２７に、ラスタ空間Ｘ－Ｙ２６９のラスタ出力装置が示されている。勾配空間Ｕ－Ｖ２７０内で画定された勾配がラスタ装置空間２６９内にラスタ化されている。ラスタ装置のラスタ走査線２６７は、走査線内の開始画素２６８から開始される、勾配で満たされた画素のランを含むべきものである。正のラスタ空間水平方向への１ラスタ画素ステップに対する、対応する勾配空間水平位置の変化（∂ｕ／∂ｘ）２７１及び対応する勾配空間垂直位置の変化（∂ｖ／∂ｘ）２７２が示されている。正のラスタ空間垂直方向への１ラスタ画素ステップに対する、対応する勾配空間水平位置の変化（∂ｕ／∂ｙ）２７３及び対応する勾配空間垂直位置の変化（∂ｖ／∂ｙ）２７４が示されている。

【０４３８】

これで、フィル生成バッファに出力すべきランを形成する１つ又は複数の連続画素の画素カラー値の生成に進むことができる。開始画素のラスタ空間座標位置（ｘstart，ｙstart）が与えられると、最初に、対応する初期勾配空間位置（ｕstart，ｖstart）の計算が行われる。図２７で、この開始ラスタ画素の例が２６８で示される。対応する初期勾配空間位置は、以下の式を使用して計算することができる。

【０４３９】

$$ustart = uo + xstart \times (\partial u / \partial x) + ystart \times (\partial u / \partial y)$$

$$vstart = vo + xstart \times (\partial v / \partial x) + ystart \times (\partial v / \partial y)$$

これらの計算から、図４２のステップ４１５で、勾配ルックアップテーブル内へのインデックスを維持するための初期追跡値が決定される。より具体的には、初期勾配ルックアップテーブルインデックスの２乗（$index^2$）及びこの２乗されたインデックス値の初期増分値｛∂（$index^2$）／∂ｘ｝は、以下の通り計算される。

【０４４０】

$$index^2 = ustart^2 + vstart^2$$

$$\partial(index^2)/\partial x = 2 \times \{(ustart \times (\partial u/\partial x) + vstart \times (\partial v/\partial x)\} + (\partial u/\partial x)^2 + (\partial v/\partial x)^2$$

$index^2$値が計算されると、ステップ４１６で、先に述べたような勾配ルックアップテーブルを使用して対応するカラー値がルックアップできるようにするために、この値の平方根を取ることができる。

【０４４１】

画素のカラー値を生成した後、ステップ４１７で、画素のラン内に径方向勾配から生成すべき画素カラー値が他にあると決定された場合、$index^2$及び｛∂（$index^2$）／∂ｘ｝の各値は、ステップ４１８で以下の漸化式によって増分することができる。

【０４４２】

$$index^2n = index^2n\text{-}1 + \partial\, index^2n\text{-}1$$

$$\partial(index^2n)/\partial x = \partial(index^2n\text{-}1)/\partial x + 2 \times \{(\partial u/\partial x)^2 + (\partial v/\partial x)^2\}$$

ここで値２×｛（∂ｕ／∂ｘ）^{2}＋（∂ｖ／∂ｘ）^{2}｝は一定であり、事前計算することができることに注意すべきである。

【０４４３】

このプロセスは、画素の現ラン内の必要な全ての画素が生成されるまで繰り返される。その後、ステップ４１９で、同じ径方向勾配を使用したレンダリングを必要とする画素のランが更に存在すると判断された場合、ステップ４１５に戻って新しいランの開始画素用の初期インデックス追跡値を再計算することにより、この同じプロセスを繰り返すことができる。要約すると、初期追跡値が計算された後、この技法では、各画素毎に２つの加算と平方根の計算を行うだけで画素のランに沿って径方向勾配がレンダリングできるようになる。

【０４４４】

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素早く実行でき、或は、「Reciprocation, Square Root, inverse Square Root, and Some Elementary Functions Using Small Multipliers」, M. D. Ercegovac et al, IEEE Transactions on

Computers, Vol. 49, No. 7, July 2000に記述されたプロセスにより、２つの２５６エントリのルックアップテーブルを使用して、より効率的にかつ同じ精度で実行することができる。

【０４４５】

記述された方法では、径方向勾配の画素を、左上から右下に至るラスタ走査順にレンダリングする。即ち、走査線がディスプレイの上から下に向かってレンダリングされ、走査線内の画素が左から右へとレンダリングされる。同じ技法を適切に改変することにより、他の動作のラスタ走査順序に同様に適用することができる。この点に関連して、ラスタ走査順序は、２次元空間の任意の１次元走査と定義することができる。好ましくは、ラスタ走査順序は左上から右下に至る順序である。

【０４４６】

記載の径方向勾配生成法は、他の用途での使用に適していることが認められるはずである。一般に、記載の方法は、入力値のシーケンスが全て直線状に並び、各結果値が関数入力によって記述されるベクトルの絶対値の関数であるとき、１つの２次元入力値セットから任意の１つの結果値セットを生成するのに適している。より具体的には、記載の方法は、１つの離散位置セットがラスタ走査順にあるとき、その１つの離散位置セットに対して径方向の関数の値を計算するのに適している。径方向関数は、２次元空間内の特定位置からの入力の距離の関数である結果値（又は結果値の組）にマッピングを行う、２変数の任意の関数と定義することができる。

【０４４７】

８．３　ビットマップ画像画素生成

ｚレベルは、ビットマップ画像フィルをもつことができる。これは、ｚレベルのフィルが、所定の色空間内の点サンプルを規定する画素データ行を含むデジタル画像に基づくものでもよいことを意味する。ビットマップ画像の画素データは、任意のフレームの前に更新することができる。そのような技法により、動画ビデオに似たアニメーションビットマップ画像フィル効果が容易になる。ビットマップ画像フィルのレンダリングでは、勾配フィルのレンダリングに関して説明したものと同じ手法が用いられる。

【０４４８】

ラスタ空間画素位置をビットマップ画像値にマッピングするために、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定義することが適切である。このビットマップ空間表面の幅及び高さは、ビットマップ画像の幅及び高さのピクセル値である。ビットマップ空間表面の原点は、ビットマップ画像の左上画素に対応する。勾配空間表面がラスタ空間に変換できるのと同様にして、ビットマップ空間表面はラスタ空間に変換できる。このため、ビットマップを平行移動、拡大縮小、回転及びスキューすることが可能になる。

【０４４９】

ビットマップ画像フィルに関して、ラスタ空間内の各画素位置は何らかのカラー値にマッピングされる。ビットマップ画像内の点にマッピングされるラスタ空間画素位置は、ビットマップ画像データによって規定される、対応する画素カラー値を使用することができる。ビットマップ画像の外の点にマッピングされるラスタ空間画素位置は、様々な手段を用いて画素カラー値にマッピングされるものと定義することができる。例えば、ビットマップ画像の外の点にマッピングされるラスタ空間画素位置は、その画素位置に最も近いビットマップ画像画素にマッピングされるものと定義することができる。別の例として、ビットマップ画像がラスタ空間全体に渡って無限に繰り返されるタイル状のビットマップ画像効果が得られるように、ラスタ画像画素位置をビットマップ画素にマッピングすることがある。

【０４５０】

図４を参照すると、ラスタ画像画素データ及びラスタ空間に変換すべきビットマップを含むビットマップ画像記述が与えられると、出力画素カラー値を計算する前に、ステップ４４で以下のものを計算することが適切である。

【０４５１】

（ｉ）ラスタ空間水平方向への画素ステップあたりのビットマップ空間水平位置の変化（∂ｕ／∂ｘ）
（ｉｉ）ラスタ空間水平方向への画素ステップあたりのビットマップ空間垂直位置の変化（∂ｖ／∂ｘ）
（ｉｉｉ）ラスタ空間垂直方向への画素ステップあたりのビットマップ空間水平位置の変化　（∂ｕ／∂ｙ）
（ｉｖ）ラスタ空間垂直方向への画素ステップあたりのビットマップ空間垂直位置の変化（∂ｖ／∂ｙ）
（ｖ）ラスタ空間原点に対応するビットマップ空間位置（ｕｏ，ｖｏ）

これらの事前計算値は、ビットマップからラスタ空間への変換によって決まり、変換が改変される場合のみ再計算を必要とする。これらの事前計算値を使用することによって、ビットマップ画像データ内の現画素のアドレスを増分的に追跡することができる（address）。

【０４５２】

次いで、フィル生成バッファに出力すべきランを形成する１つ又は複数の連続する画素の画素カラー値の生成に進むことができる。開始画素のラスタ空間座標位置（ｘstart，ｙstart）が与えられると、対応する初期ビットマップ空間位置（ｕstart，ｖstart）がまず計算される。これは、以下の式を使用して計算される。

【０４５３】

ｕstart＝ｕo＋ｘstart×（∂ｕ／∂ｘ）＋ｙstart×（∂ｕ／∂ｙ）
ｖstart＝ｖo＋ｘstart×（∂ｖ／∂ｘ）＋ｙstart×（∂ｖ／∂ｙ）

これらの計算から、ステップ４５で、ビットマップ画像データ内への初期アドレス（address）が決定される。アドレス（bitmapbase）で始まる走査順のデータをもつビットマップ画像について、ビットマップ画像データが１画素あたり（bpp）バイト及び行幅（rowstride）バイトをもつビットマップ画像データであるとき、ビットマップ画像データ内への初期アドレスは以下のように計算される。

【０４５４】

address＝bitmapbase＋ｖstart×rowstride＋ｕstart×bpp

address値が計算されると、ステップ４６で、そのアドレスでの画素カラー値が現出力ラスタ画素の出力画素カラー値として使用することができる。画素のカラー値が生成された後、ステップ４７で、ビットマップ画像から生成すべき他の画素カラー値が現在の画素のラン内にあると判定された場合、ステップ４８で、address値は以下の漸化式によって増分することができる。

【０４５５】

address n＝address n-1＋（∂ｖ／∂ｘ）×rowstride＋（∂ｕ／∂ｘ）×bpp

addressの増分値は定数として事前計算することができることに留意されたい。このプロセスは、現在の画素のラン内の全ての画素が生成されるまで繰り返される。その後、ステップ４９で、同じビットマップ画像を使用したレンダリングを必要とする画素のランが更に存在すると判定された場合、ステップ４５で次の画素のランの開始画素用の初期インデックスaddressを再計算することにより、この同じプロセスを繰り返すことができる。要約すると、このモジュールは、１画素あたり２つの追加を行うだけで、出力画素のランに沿ってビットマップ画像がレンダリングされるようにする。

【０４５６】

９．画素抽出

図５６の画素抽出モジュール６１８は、ラスタ化処理の最後のステップを行い、完成画素を出力する。これらの画素は、ディスプレイ装置に直接出力することができ、或は後でディスプレイ装置に出力できるようにフレームバッファメモリ内に格納することができる。画素の出力先は、システム設計上の選択である。画素の出力先を１つだけ使用するシステムもあれば、複数の出力先に関してランタイム構成ができるシステムもあり得る。

【０４５７】

出力画素の精度が画素抽出モジュール６１８に供給されるカラー値の精度よりも低いシステムの場合、画素抽出モジュール６１８はハーフトーニングの追加ステップも実行することができる。例えば、画素抽出モジュール６１８が１チャネルあたり８ビットのカラー情報（ＲＧＢ　８：８：８）を受け取り、１チャネルあたり５ビットの画素（ＲＧＢ　５：５：５）を出力するシステムでは、ハーフトーニングを実行してよい。

【０４５８】

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」を受け取る。スパンは、ディスプレイ装置の走査順にある、同じ色をもつ一連の連続する画素である。スパン長は、短い場合は１画素分でもよく、長い場合はディスプレイ全体の画素分でもよい。各スパンは２つの値、即ちスパンの色及び画素単位で表したその長さとして画素抽出モジュール６１８に入力される。

【０４５９】

９．２　フレームバッファへの出力

この動作モードでは、画素抽出モジュール６１８はフレームバッファメモリに画素を出力する。このモードでの動作を、図３７のフローチャートを参照しながら説明する。ステップ３７０で、処理が開始される。ステップ３７１で、フレームバッファ位置変数Ｘ及びＹがともに０に初期化され、それによりフレームバッファの左上の出力位置が示される。ステップ３７８で、画素抽出モジュール６１８が表示すべき次のスパンを受け取る。ステップ３７７で、変数Ｎ及びＣにそれぞれスパンの長さ及び色が割り当てられる。続いて、ステップ３７６で、画素抽出モジュール６１８がフレームバッファ内の位置（Ｘ、Ｙ）の画素の色を色Ｃに設定する。ステップ３７５で、フレームバッファ位置Ｘ変数が増分される。ステップ３７９で、フレームバッファ変数Ｘが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達したか、又はそれを超えた場合、処理はステップ３７４に移るが、そうでない場合、処理はステップ３６９に進む。ステップ３６９で、スパン長カウンタ変数Ｎが減分される。ステップ３８０で、スパン長カウンタ変数Ｎが０と比較される。Ｎが０の場合、処理はステップ３７８に移るが、そうでない場合、処理はステップ３７６に移る。

【０４６０】

図３７のステップ３７４で、フレームバッファ変数Ｘが０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で、フレームバッファ変数Ｙがフレームバッファの高さと比較される。Ｙが高さ以上である場合、処理はステップ３７２で終了し、そうでない場合、処理はステップ３６９に進む。

【０４６１】

９．３　ディスプレイへの直接出力

この実施形態では、画素抽出モジュール６１８は画素を直接ディスプレイ装置へ出力する。画素抽出モジュール６１８でのスパン情報の到着は、ディスプレイ装置上の画素の表示レートに対して非同期である可能性が高いため、画素抽出モジュール６１８は、ディスプレイ装置にスパンを出力する前に、ＦＩＦＯ（先入れ先出し）バッファを使用してスパンをキューする。ＦＩＦＯバッファは、入力スパンと出力画素のそれぞれのタイミング間にある程度の融通性を提供し、タイミング差を許容できるようにする。

【０４６２】

以下、この実施形態を図３８（ａ）及び（ｂ）のフローチャートを参照することにより説明する。ＦＩＦＯバッファにスパンを受け入れるプロセスは図３８（ａ）のステップ３８８から３８９で記述され、ＦＩＦＯバッファからスパンを出力するプロセスは図３８（ｂ）のステップ３８２から３８３で記述される。これらの２つのステップのタイミングは、ＦＩＦＯバッファによって大体は非同期に保たれる。

【０４６３】

図３８（ａ）に見られるように、ステップ３８８で、スパン（画素のラン）の受け入れプロセスが始まる。ステップ３９２で、スパンが画素抽出モジュール６１８に受け入れら

れる。ステップ３９１で、ＦＩＦＯバッファが満杯であるかどうかがテストされる。満杯である場合、処理はステップ３９０に移るが、そうでない場合、処理はステップ３８９に進む。ステップ３９０で、ＦＩＦＯバッファが満杯でなくなるのを待つ。ステップ３８９で、ステップ３９２で受け入れられたスパンがＦＩＦＯバッファ内に格納される。

【０４６４】

図３８（ｂ）に見られるように、ステップ３８２で、画素の出力プロセスが始まる。ステップ３８７で、ＦＩＦＯからスパンが読み出される。ステップ３８６で、スパンの長さ及び色がそれぞれ変数Ｎ及びＣに格納される。ステップ３８５で、次の画素を出力する時機であるかどうかを判断するテストが行われる。まだその時機でない場合、処理は現ステップを繰り返す。次の画素を出力する時機である場合、処理はステップ３８４に移る。ステップ３８４で、色Ｃの画素がディスプレイに出力される。ステップ３８１で、変数Ｎが減分される。ステップ３８３で、変数Ｎが０と比較される。０と等しい場合、処理はステップ３８７に移るが、そうでない場合、処理はステップ３８５に進む。

【０４６５】

９．４　ハーフトーニング

図２９は、画素抽出モジュール６１８によって使用されるハーフトーニング技法を示す。この技法は、誤差拡散法の一種である。図２９（ａ）は、１画素の量子化誤差を、次の走査線上にある４つの画素を含む隣接した５つの画素に拡散する、従来技術の誤差拡散技法を示す。図２９（ｂ）は、１画素の量子化誤差を、その現画素と同一走査線上にある隣接した１つだけの画素に拡散する、画素抽出モジュール６１８の誤差拡散法を示す。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を示す。システム２９００は、簡単な一次の負のフィードバック制御システムである。現画素の量子化によって導入された５ビット精度以内の誤差が次の画素に加えられ、負のフォードバックが提供される。システム２９００は、図２９（ｃ）に示されるように結合された量子化器２９０１及び加算器２９０２を使用したハードウェアで構成することができる。或は、誤差拡散システムは、図３０に示された方法を使用してソフトウェアで構成することができる。説明を分かり易くするために、誤差拡散アルゴリズムを１つのカラーチャネルのみに関して説明する。実際のシステムでは、３つのカラーチャネル全てが同様に処理される。

【０４６６】

図３０は、ハーフトーニングアルゴリズムのフローチャート表現を提供する。ステップ２９２で、ハーフトーニング処理が開始される。ステップ２８３で、変数がその初期値に設定され、ステップ２９０で８ビット精度の画素強度が読み出される。ステップ２８９で、ステップ２９０で読み出された画素強度にError変数が加えられる。その加算結果は、Desired変数に格納される。続いて、ステップ２８８で、Desired変数が５ビット精度に量子化され、Actual変数に格納される。８ビットから５ビットへの量子化は、３ビットを右に論理シフトさせることによって実現される。ステップ２８７で、量子化されたカラー値であるActualが出力される。次いで、ステップ２８６で、実際の出力色であるActualが８ビット値として再表現される。これは、Actualの左シフトした３ビットをActualの右シフトした２ビットに加えることで実現される。その結果得られた値は、ActEcho変数に格納される。ステップ２８５で、DesiredからActEchoが減算され、それによって量子化誤差が計算される。この結果は、Error変数に格納される。ステップ２８４で、最後の画素に達したかどうかを調べるためにテストが行われる。最後の画素に達していた場合、ステップ２９３で処理が終了する。ステップ２９１で、変数Ｘが増分される。

【０４６７】

１０．実装

上述したＴＣＩＥシステム６９９は、ハードウェア又はソフトウェアで、或は両者の組合せで実装することができる。ハードウェア実装では、図５６のモジュールは１つ又は複数のＡＳＩＣデバイスに組み入れ、携帯電話機などの目的装置に組み込むことができる。ソフトウェア実装では、そのようなソフトウェアを限定されたコンピューティングシステ

ムでよく使われる、汎用ではあるがその能力が限定されたプロセッサ上で動作するように構成することができる。混成の実装は、表示リストコンパイラ３０８のソフトウェア実装、及びレンダリングエンジン６１０の全部又は一部のハードウェア実装に役立ち、それにより、一般に、より柔軟性が高いデータ（画像）入力及びより速いレンダリングレートを与える。更に、システム６９９はシンクライエントでの使用を目的として開発されてはいるが、これ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大きなコンピューティングリソースをもつ装置への同じイメージング及びレンダリング手法の適用を妨げるものではない。

【０４６８】

上述したシンクライエントイメージングの方法は、図６９に示すようなシステム６０００を使用して実行することができる。そこでは、記載の諸プロセスが完全に、例えばシステム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実装される。特に、図５６の方法の各ステップは、コンピューティングデバイス６００１によって実行されるソフトウェア内の命令によって実現される。命令は、それぞれ１つ又は複数の特定のタスクを実行するための１つ又は複数の符号モジュールとして形成することができる。このソフトウェアはまた、その第１部分が具体的なイメージング及びレンダリング方法を実行し、第２部分が第１部分とユーザとの間のユーザインターフェースを管理するような２つの別々の部分に分割することもできる。ソフトウェアは、例として後で述べる記憶装置を含む、コンピュータ可読媒体に格納することができる。ソフトウェアはコンピュータ可読媒体からコンピュータにロードされ、次いでコンピュータによって実行される。そのようなソフトウェア又はコンピュータプログラムを格納したコンピュータ可読媒体はコンピュータプログラム製品である。コンピュータでのコンピュータプログラム製品の使用は、好ましくはシンクライエントイメージングにおける有利な装置をもたらす。

【０４６９】

システム６０００は、通信リンク６０２１によって相互接続されたコンピューティングデバイス６００１及びネットワーク６０２０を含む。コンピューティングデバイス６００１は、例えば携帯電話機でもよく、その場合ネットワーク６０２０はセルラーネットワークなどの加入者電話ネットワークでもよい。デバイス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、或はローカルエリアネットワーク（ＬＡＮ）又はワイドエリアネットワーク（ＷＡＮ）など他のネットワークシステムでもよい。コンピューティングデバイス６００１は、キーボード、タッチパネル、及びマウスやトラックボールなどのポインティングデバイスを装備できるようにするユーザ入力インターフェース６００２を含む。コンピューティングデバイス６００１は、実装される上述の例に応じて、携帯電話トランシーバエレクトロニクス又は複写機エンジンなどの機能ハードウェア６０１２を内蔵する。コンピューティングデバイス６００１は、ネットワーク６０２０と通信を行い、一般にはデバイス６００１の主要な機能的役割を果たすために機能インターフェース６００８を使用する。

【０４７０】

コンピューティングモジュール６００１は、一般に、少なくとも１つの処理ユニット６００５と、それぞれ揮発性及び不揮発性メモリとして働くメモリユニット６００６及び６０１０を含む。揮発性メモリ６００６は、半導体ランダムアクセスメモリ（ＲＡＭ）によって形成されてもよく、不揮発性メモリは、半導体リードオンリーメモリ（ＲＯＭ）、ハードディスクドライブ又は光メモリプラットフォームによって形成されてもよい。例えば、メモリ６００６は図５６のメモリ６０９として動作できる。モジュール６００１はまた、ディスプレイ６０１４と結合される画像出力バッファ６００７、及びユーザインターフェース６００２用の入出力（Ｉ／Ｏ）インターフェース６０１３を含めて、いくつかのＩ／Ｏインターフェースを含む。コンピューティングデバイス６００１のモジュール６００２～６０１４は、一般に相互接続されたバス６００４を介して、関連技術分野の技術者に周知の、コンピューティングデバイス６００１の従来型動作モードをもたらすように通信する。

【０４７１】
一般に、シンクライエントイメージング用のアプリケーションプログラムは不揮発性メモリ６０１０上に常駐し、実行される際にはプロセッサ６００５によって読み出され制御される。プログラム及びネットワーク６０２０から取得したデータの中間記憶は、揮発性メモリ６００６を、場合によっては、例としてはハードディスクドライブがある不揮発性メモリと一緒に使用して実現することができる。ある場合には、アプリケーションプログラムを不揮発性メモリ６０１０内に符号化してユーザに供給することができ、或はユーザによってインターフェース６００８を介してネットワーク６０２０から読み取ることもできる。更に、ソフトウェアを他のコンピュータ可読媒体からコンピューティングデバイスシステム６００１にロードすることができる。ここで、「コンピュータ可読媒体」という用語は、実行及び／又は処理のための命令及び／データのコンピュータシステム６０００への提供に関与する任意の記憶又は通信媒体を指す。記憶媒体の例としては、コンピューティングデバイス６００１の内部にあろうと外部にあろうと、フロッピー（登録商標）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カードが挙げられる。通信媒体の例としては、無線又は赤外線通信チャネル、並びに他のコンピュータ又はネットワークデバイスへのネットワーク接続や、電子メール送信及び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及びイントラネットへのネットワーク接続が挙げられる。
【０４７２】
携帯電話機の例では、本発明を使用してグラフィックベースのアニメーションゲームを行うことができ、その際にネットワーク６０２０はゲームの各フレーム毎にレンダリングされる表示リストをデバイス６００１に送信し、次いでこの表示リストが個人設定及びリアルタイムのゲームプレー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によって操作され、その結果レンダリングエンジン６１０に送達され、そこから出力画素画像がバッファ６００７、ディスプレイ６０１４に順に送達される。コンパイラ６０８及びレンダラ６１０は、それぞれプロセッサ６００５によって呼ばれ実行されるソフトウェアモジュールで形成することができる。
【産業上の利用可能性】
【０４７３】
説明した構成は、コンピュータ及びデータ処理業界に適用可能であり、特に画像再生を必要とする場合、とりわけ画像生成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定されている場合に適用可能である。
【０４７４】
上記は本発明の一部の実施形態を記述したものに過ぎず、これらの実施形態は例示的であって制限的ではなく、本発明の範囲及び精神から逸脱することなくこれらの実施形態を改変及び／又は変更することができる。
【０４７５】
（オーストラリアのみ）本明細書の文脈において、「備える（comprising）」という単語は「主に含むが、それだけには限定されない」或は「有する（having）」又は「含む（including）」という意味であり、「それだけから成る（consisting only of）」という意味ではない。「comprise」及び「comprises」など、「備える（comprising）」の変形も、相応しい様々な意味をもつものとする。
【図面の簡単な説明】
【０４７６】
【図１】サブピクセルレベルにおけるアクティブエッジ判断を示す図である。
【図２】サブピクセルレベルにおける、走査線のためのエッジ処理を例示する図である。
【図３】交差メッセージを生成するための図２３のステップ２５２の方法のフローチャートである。
【図４】ビットマップ画像画素生成の方法を示す図である。
【図５】改良型のブレゼンハムアルゴリズムのＣ言語ソースコードを示す図である。

【図６】ボトムアップ合成処理のフローチャートである。
【図７】図４１の追跡パラメータ生成処理に関連する例示的なベジエ曲線を示す図である。
【図８】図２の例でサブエクセル値の屈曲カウントの複製を示す図である。
【図９】、
【図１０】合成の様々なシナリオを示す図である。
【図１１】図１３のステップ１１３のフローチャートである。
【図１２】カバレージ及びＡバッファの動作を示す図である。
【図１３】本実施の形態で使用される合成手法のハイレベルフローチャートである。
【図１４】ブレゼンハムアルゴリズムの動作を図示した円を示す図である。
【図１５】順序付けられた１つの座標セットを順序付けられた１つのエッジセットへと変換するための図４６のステップ４５６のフローチャートである。
【図１６】２次元のプリミティブがどのようにして組み合わされて図形オブジェクトを形成するかを示す図である。
【図１７】スプライトと変換マトリックスとの間の関係を示す図である。
【図１８】ストロークのエンドキャップを示す図である。
【図１９】一般化楕円を示す図である。
【図２０】楕円の下位にある円の決定を示す図である。
【図２１】交差点でのアクティブエッジの再順序付けの例を示す図である。
【図２２】図５６のエッジ処理モジュールにおけるデータフローを示す図である。
【図２３】単一の走査線におけるエッジ処理を示す、図２４のステップ２５７のフローチャートである。
【図２４】単一のフレームにおけるエッジ処理を示すフローチャートである。
【図２５】勾配フィルルックアップの動作を示す図である。
【図２６】グリフのための交差メッセージの生成を示す図である。
【図２７】様々な勾配追跡パラメータを示す図である。
【図２８】いくつかのエンドキャップのための様々なフィルを示す図である。
【図２９】従来技術における２つの誤差拡散手法を示す図である。
【図３０】モジュール６１８の誤差拡散処理（ハーフトーニング）のフローチャートである。
【図３１】スプライトとローカルな図形オブジェクトとの間のｚレベルの関係を示す図である。
【図３２】勾配ルックアップテーブルへのインデックスを決定する際のフローチャートである。
【図３３】図１６のオブジェクトを作成するために使用された左及び右フィルを示す図である。
【図３４】現在のレンダリングシステムで使用される様々な画像空間を示す図である。
【図３５】モーフィング処理における複数の座標セットを示す図である。
【図３６】ストロークされたパスの例を示す図である。
【図３７】フレームバッファメモリへ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である。
【図３８】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である。
【図３９】画素生成モジュールの処理フローを示す図である。
【図４０】出力色を生成するための画素のランの処理を示す図である。
【図４１】２次ベジエ曲線のための追跡パラメータの生成を示すフローチャートである。
【図４２】半径方向の勾配判断における漸進的手法のフローチャートである。
【図４３】非ゼロ屈曲、負の屈曲及び奇偶フィル規則から生じる様々なフィル結果を示す図である。
【図４４】表示リスト内の絶対奥行きの計算を示す図である。

【図４５】継目をストロークするためのエッジ管理を示す図である。
【図４６】モーフィング、変換及びストローキングモジュールによって順序付けられた座標の各セットに対して行われる処理を示すフロー図である。
【図４７－１】、
【図４７－２】基数ソートアルゴリズムの動作を示す図である。
【図４８】サブピクセルレベルにおける不透明度の寄与を示す図である。
【図４９】ストローキングエッジの例を示す図である。
【図５０】ストローキングエッジ及び変換エッジの例を示す図である。
【図５１】ストローキングエッジのための左及び右フィルを示す図である。
【図５２】ｚレベル・アクティブ化テーブルの動作を示す図である。
【図５３】ｚレベル・アクティブ化テーブルの再設定の例を示す図である。
【図５４】関心対象に関する状態が変化するｚレベルのためにｚレベル・アクティブ化テーブルを更新する例を示す図である。
【図５５】トップダウン合成処理のフローチャートである。
【図５６】本実施の形態に係るシンクライアントイメージングエンジンの概略ブロック図である。
【図５７】３つの屈曲規則による、ＳバッファのＡバッファへの変換を示す図である。
【図５８】ｚレベル・アクティブ化モジュールの動作を示すフローチャートである。
【図５９】ｚレベル・アクティブ化モジュールによって維持される関心対象のｚレベルのリストを示す図である。
【図６０】本明細書で説明されるいくつかの装置を実施することができるコンピュータ装置の概略ブロック図である。
【図６１Ａ】、
【図６１Ｂ】、
【図６２】合成スタックのトップダウン及びボトムアップ区分化を使用する合成手法を示す図である。
【図６３】パスの離散部分に、互いに異なるストロークが適用される様子を示す図である。
【図６４】トップダウン合成の代替形態を示す図である。
【図６５】走査線全体でのｚレベル・アクティブ化テーブルの更新を示す図である。
【図６６】グリフエッジのための交差エッジ生成を示す図である。
【図６７】楕円セグメントを、ｙ座標について単調である複数のセグメントに分割する処理を示す図である。
【図６８】画像フレームのアニメーションシーケンスを示す図である。
【図６９】図６８のアニメーションシーケンスを作成するための新しい静的エッジバッファの処理を示す図である。
【図７０】メッセージをゼロ交差させるためのＳバッファの更新を示す図である。
【図７１】図５８と類似しているが、ｚレベル・アクティブ化モジュールのリストからレベルを破棄する動作を示す図である。

【図１】

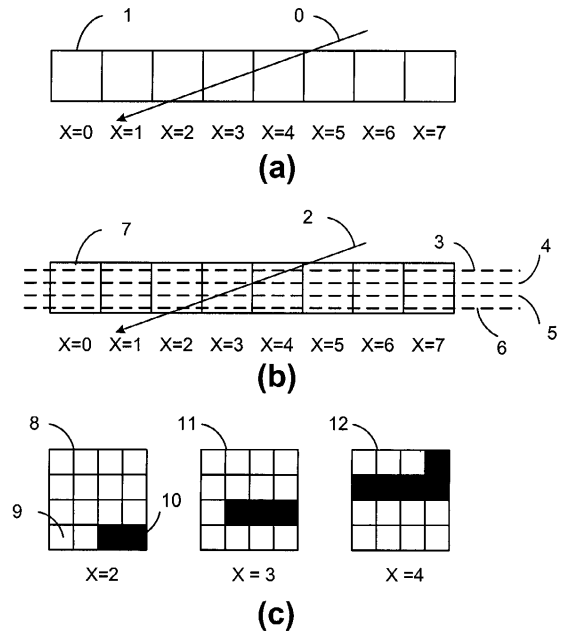


Fig. 1

【図２】

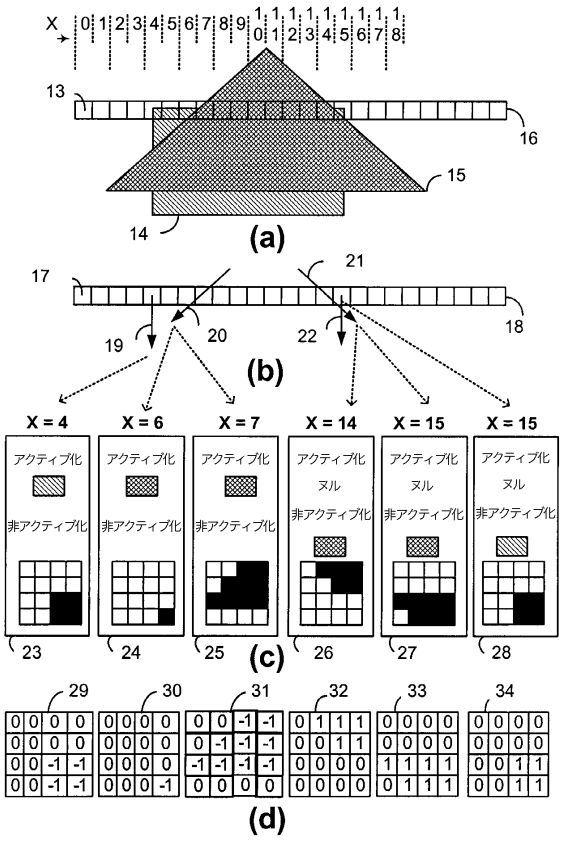


Fig. 2

【図３】

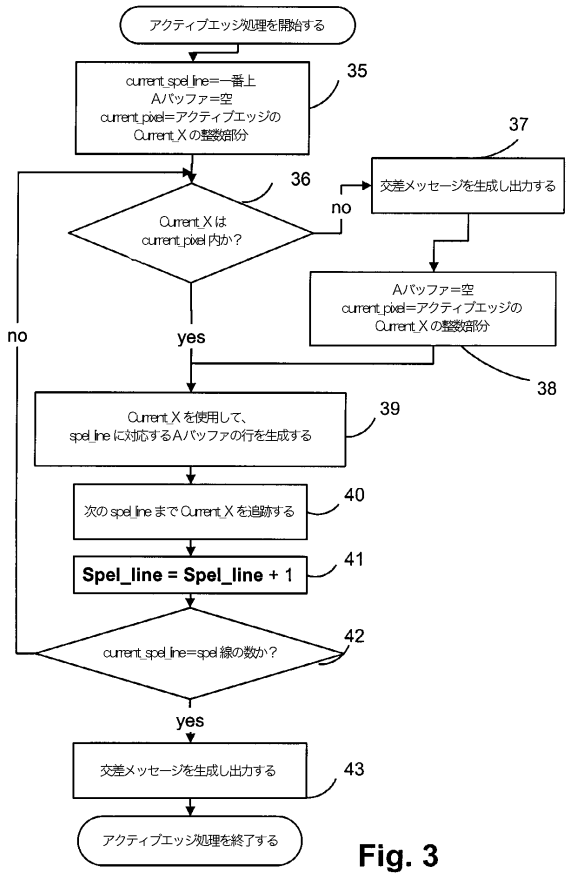


Fig. 3

【図４】

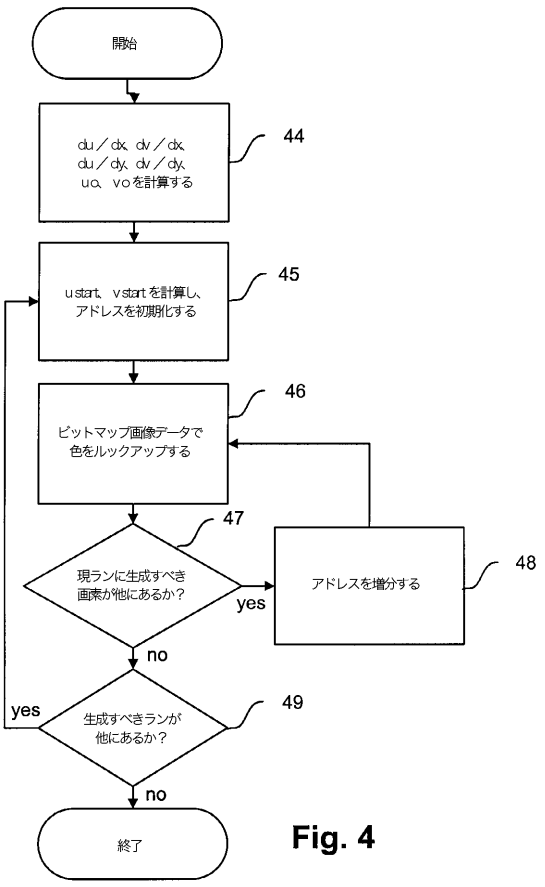


Fig. 4

【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】

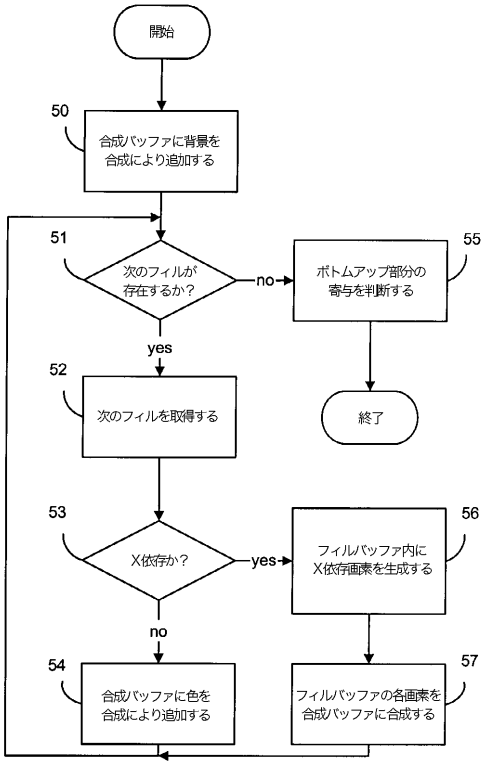


Fig. 6

【図７】

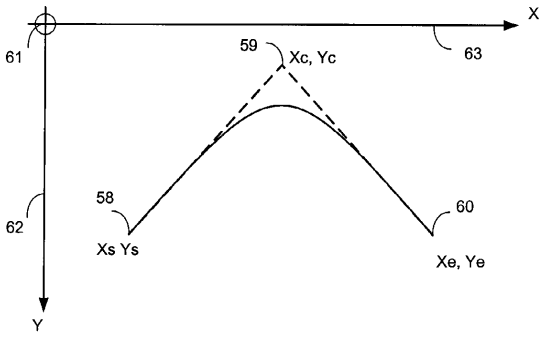


Fig. 7

【図８】

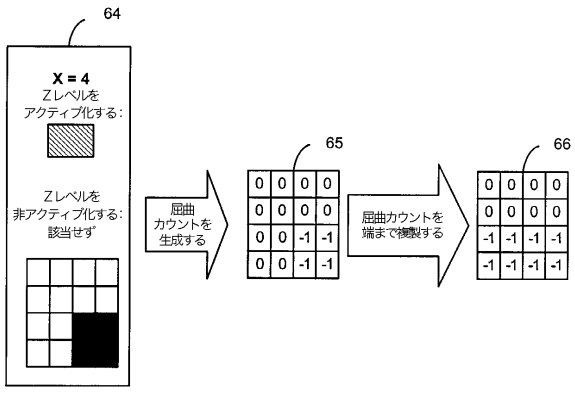


Fig. 8

【図９】

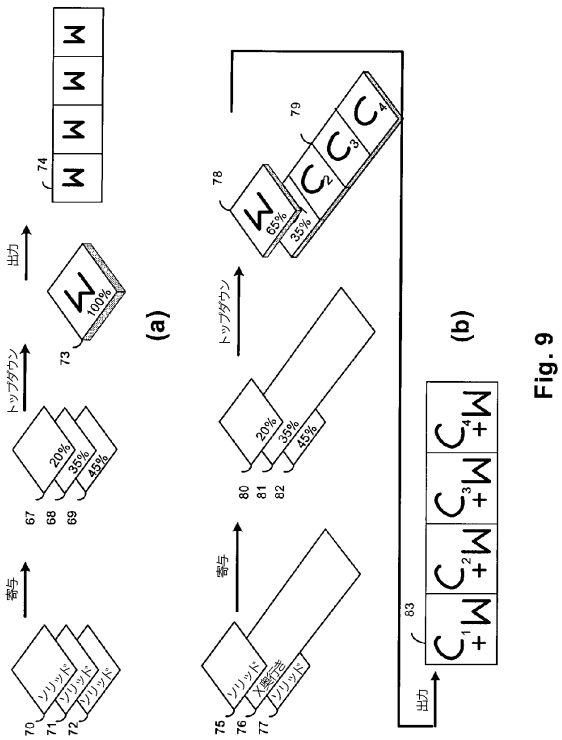


Fig. 9

【図１０】

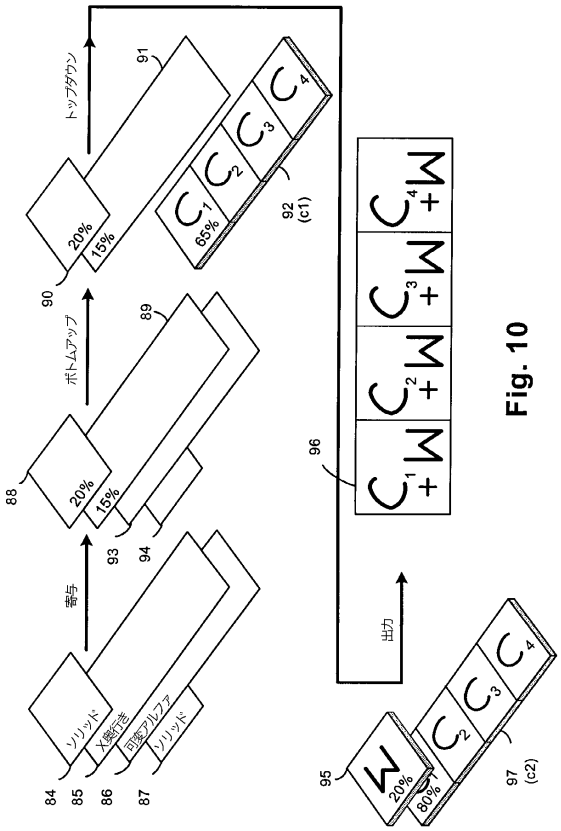


Fig. 10

【図１１】

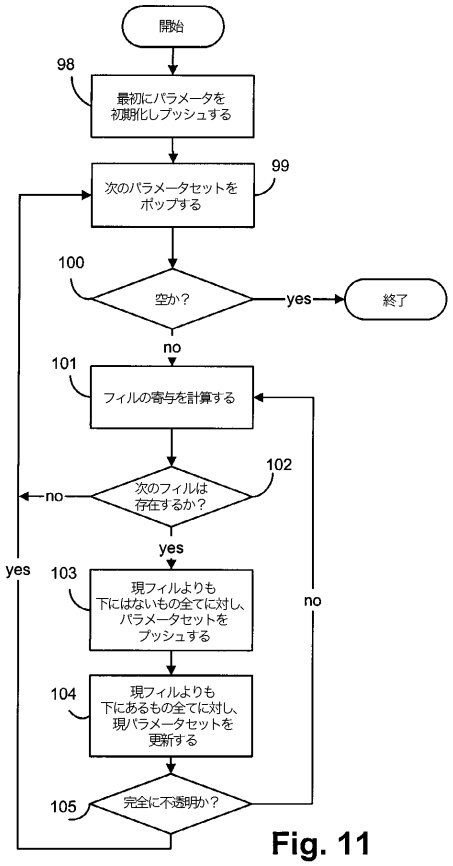


Fig. 11

【図１２】

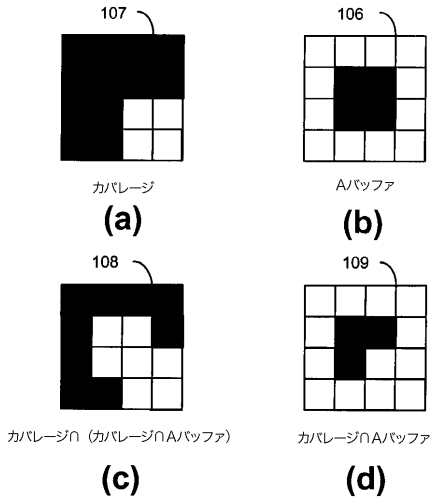


Fig. 12

【図１３】

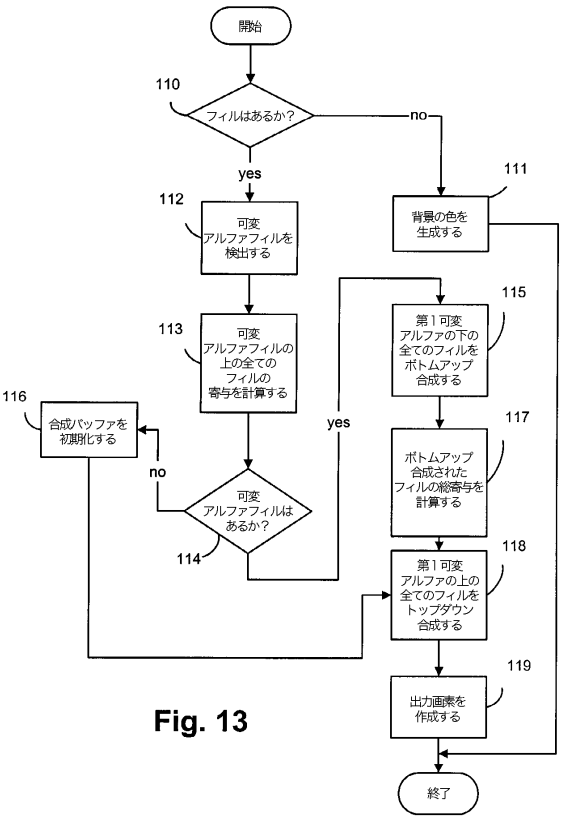


Fig. 13

【図１４】

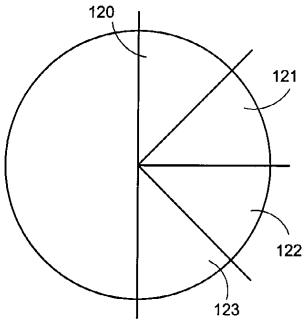


Fig. 14

【図１５】

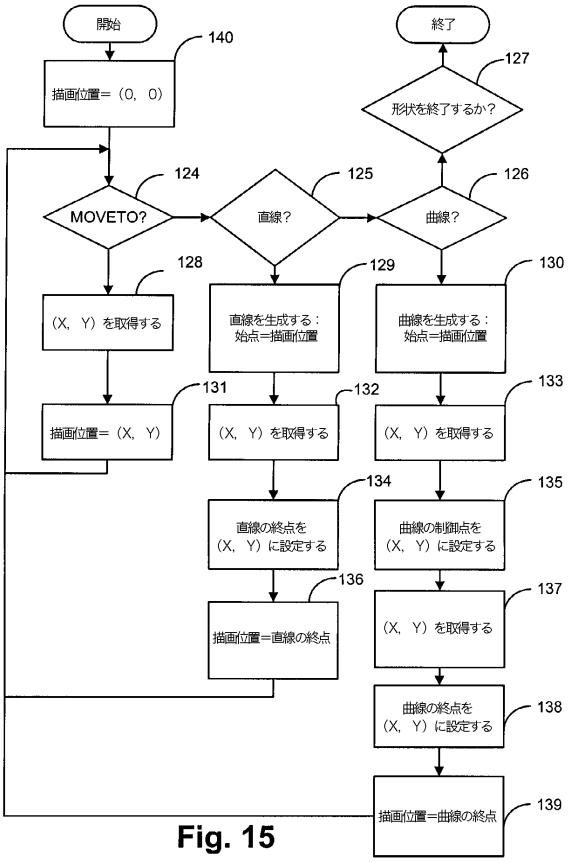


Fig. 15

【図１６】

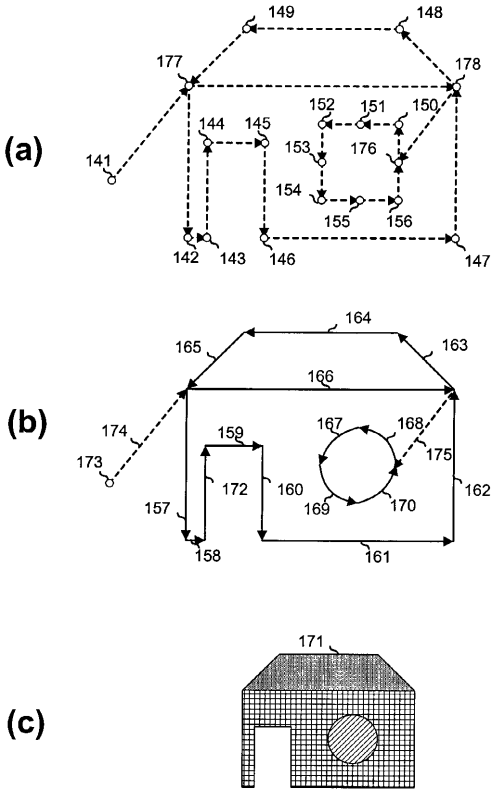


Fig. 16

【図１７】

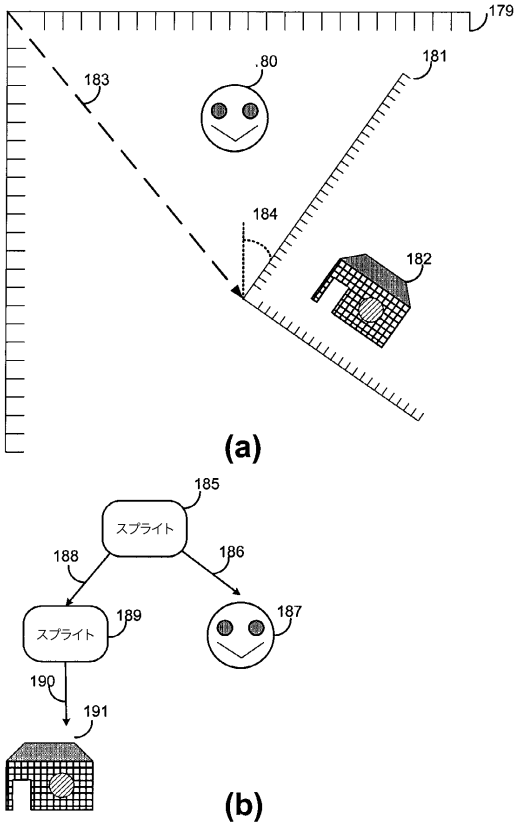


Fig. 17

【図１８】

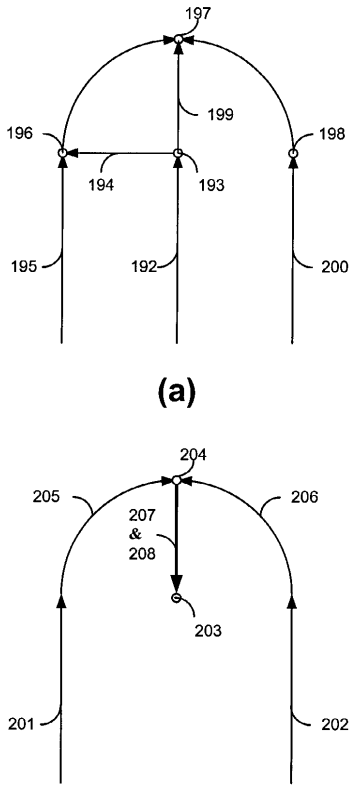


Fig. 18

【図１９】

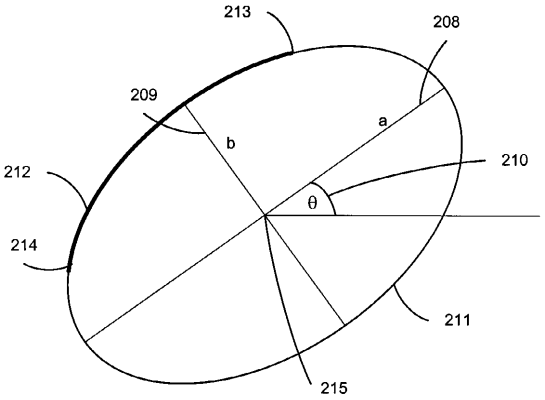


Fig. 19

【図２０】

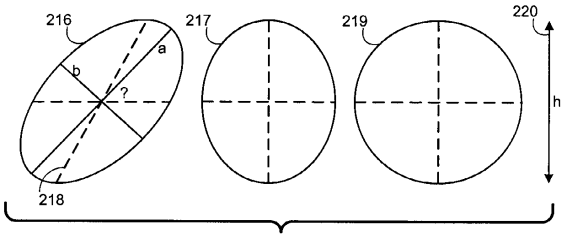


Fig. 20

【図２１】

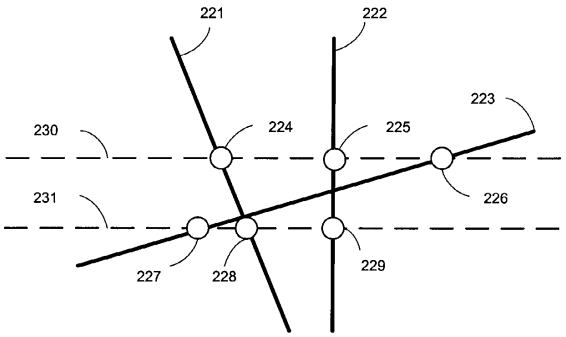


Fig. 21

【図２２】

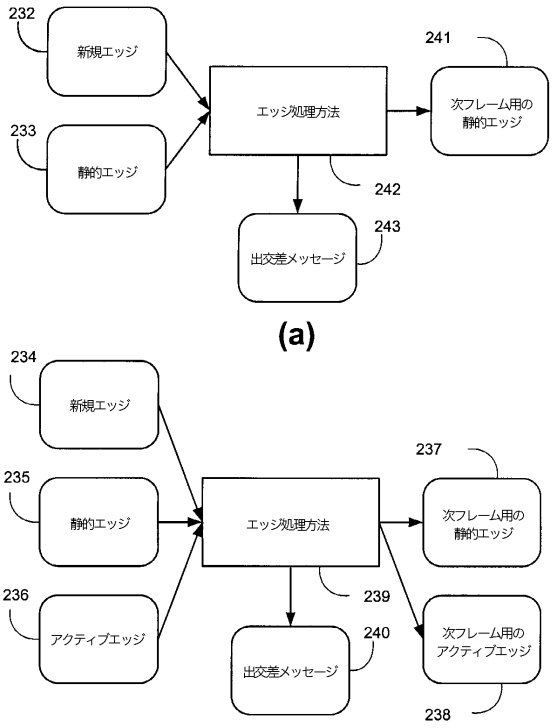


Fig. 22

【図２３】

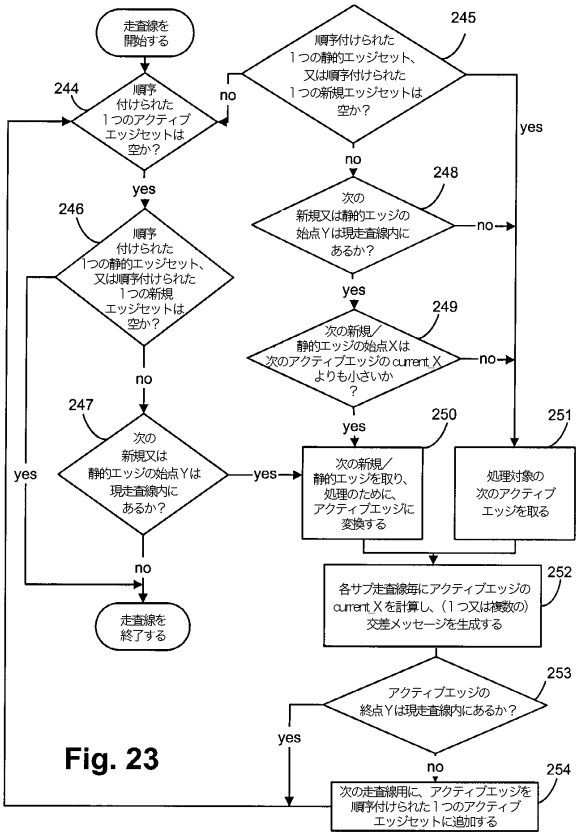


Fig. 23

【図２４】

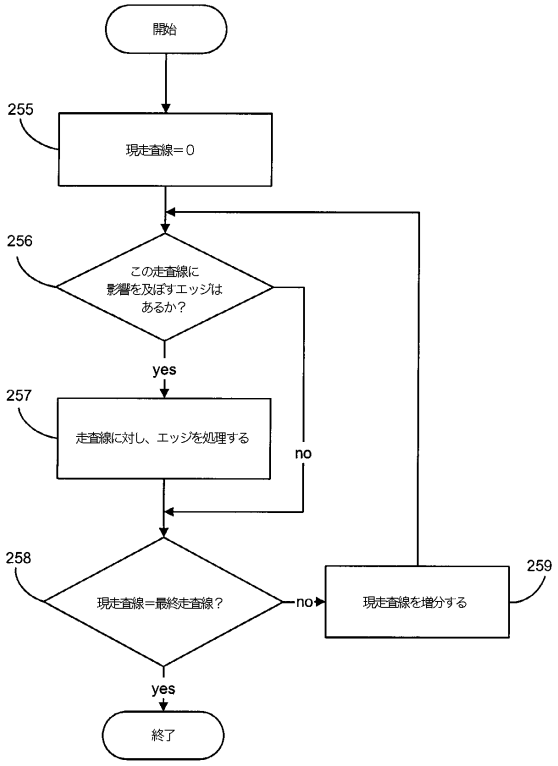


Fig. 24

【図２５】

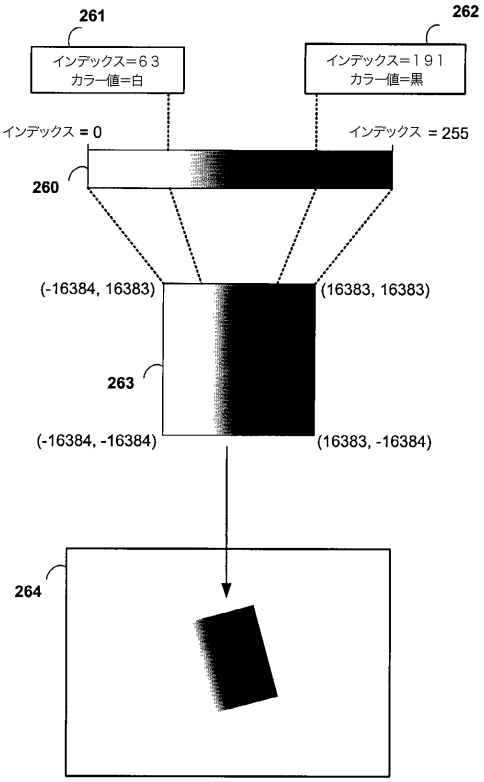


Fig. 25

【図２６】

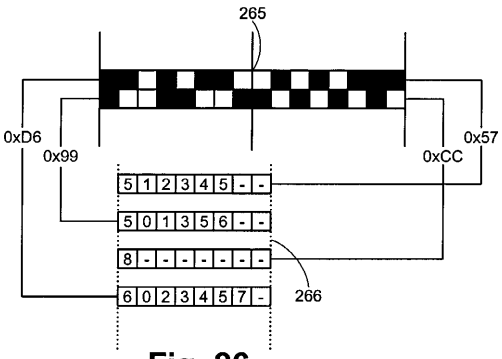


Fig. 26

【図２７】

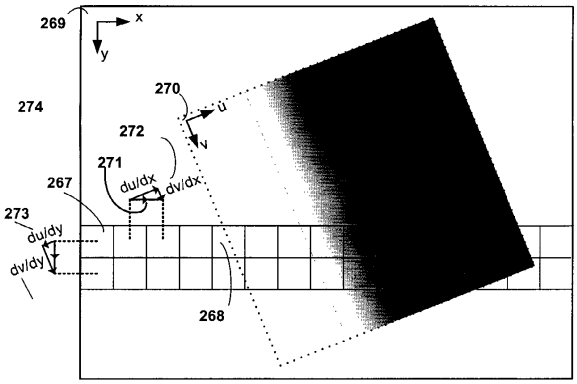


Fig. 27

【図２８】

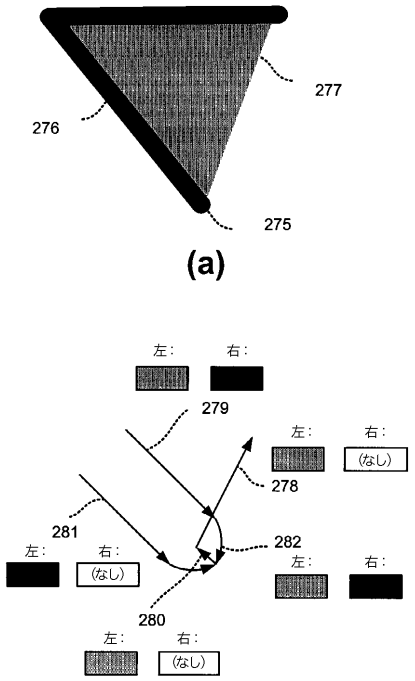


Fig. 28

【図２９】

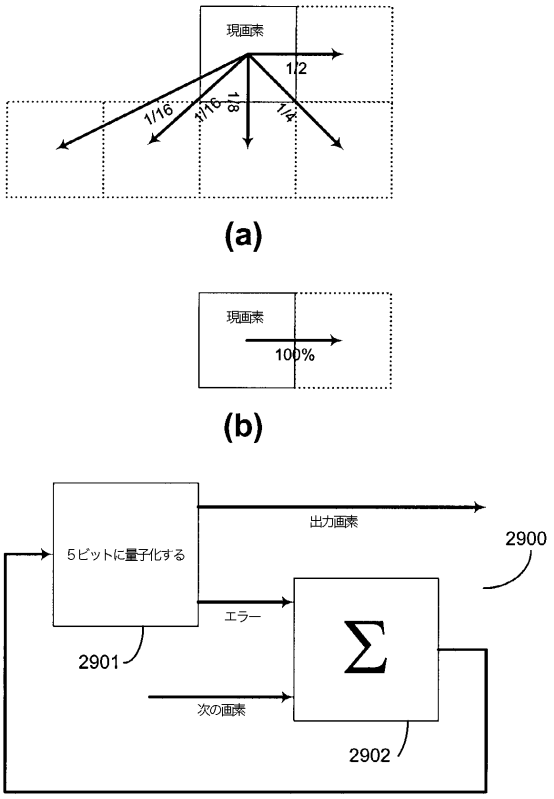


Fig. 29

【図３０】

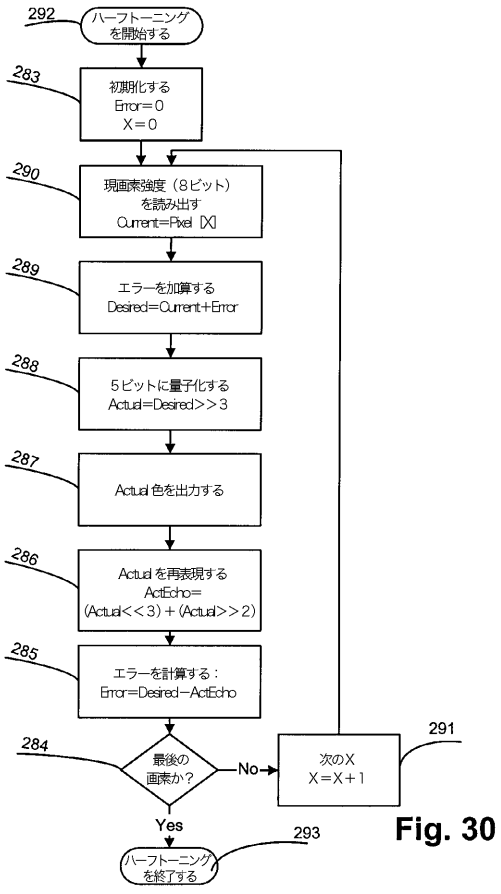


Fig. 30

【図３１】

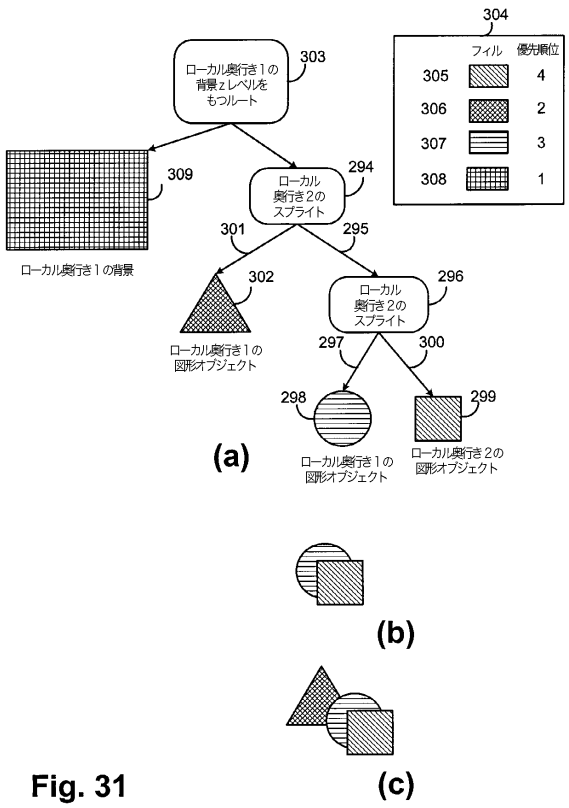


Fig. 31

【図３２】

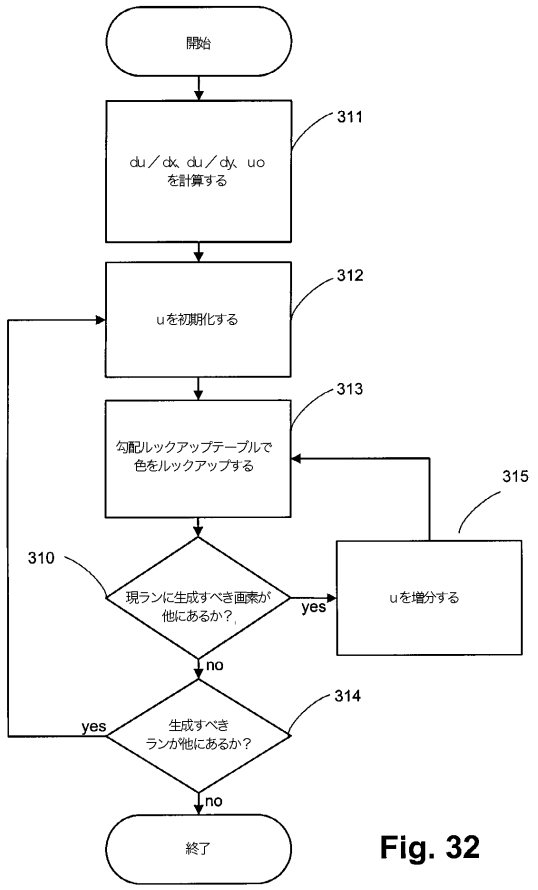


Fig. 32

【図３３】

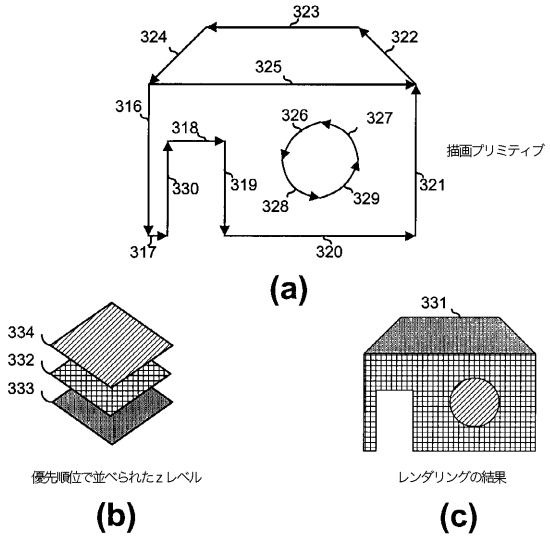


Fig. 33

【図３４】

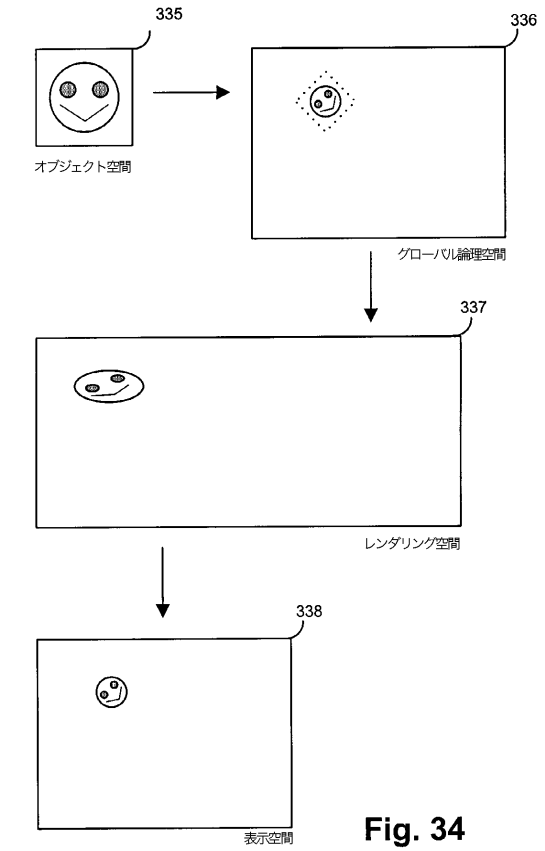


Fig. 34

【図３５】

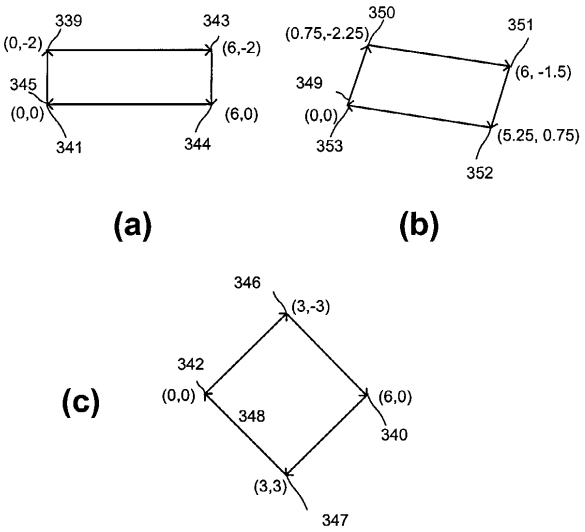


Fig. 35

【図３６】

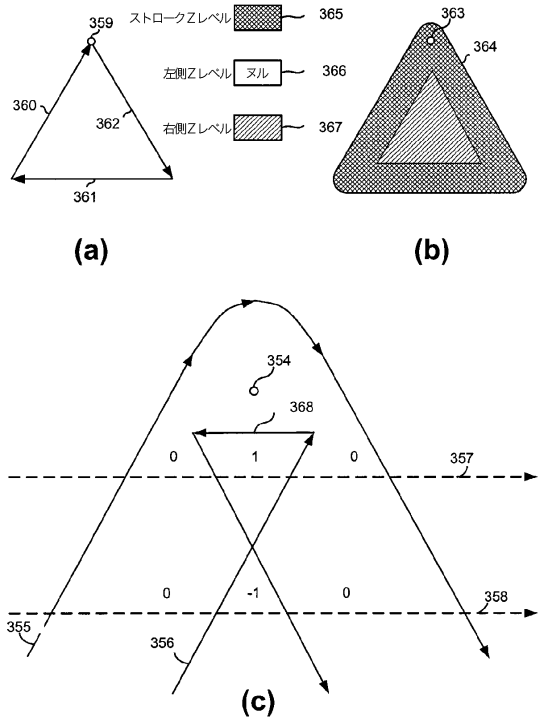


Fig. 36

【図３７】

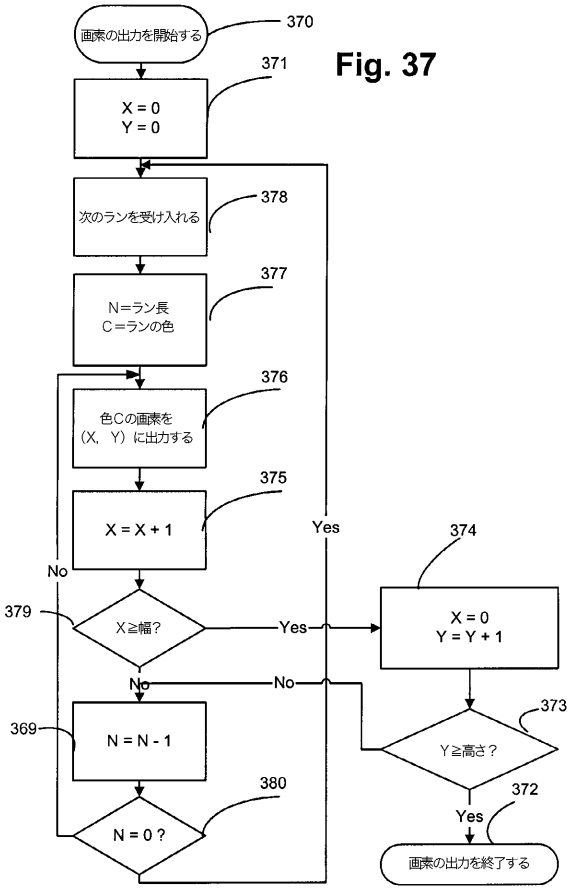


Fig. 37

【図３８】

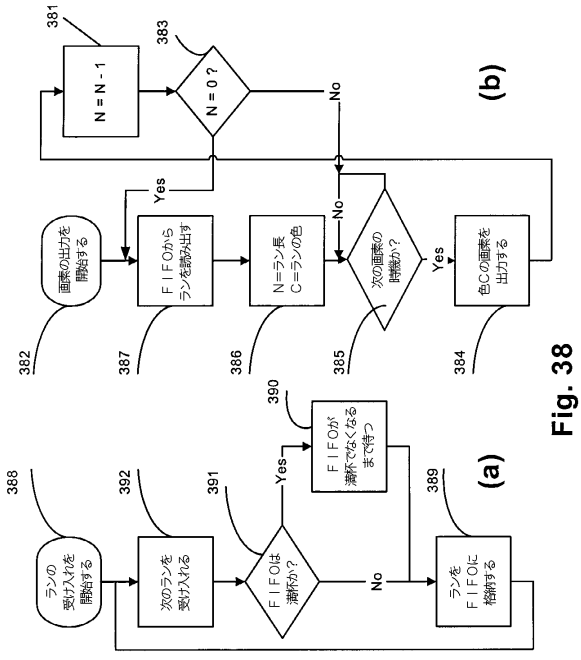


Fig. 38

【図３９】

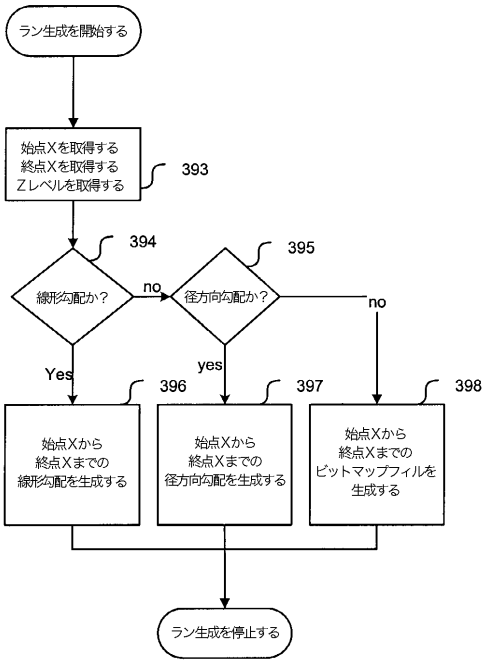


Fig. 39

【図４０】

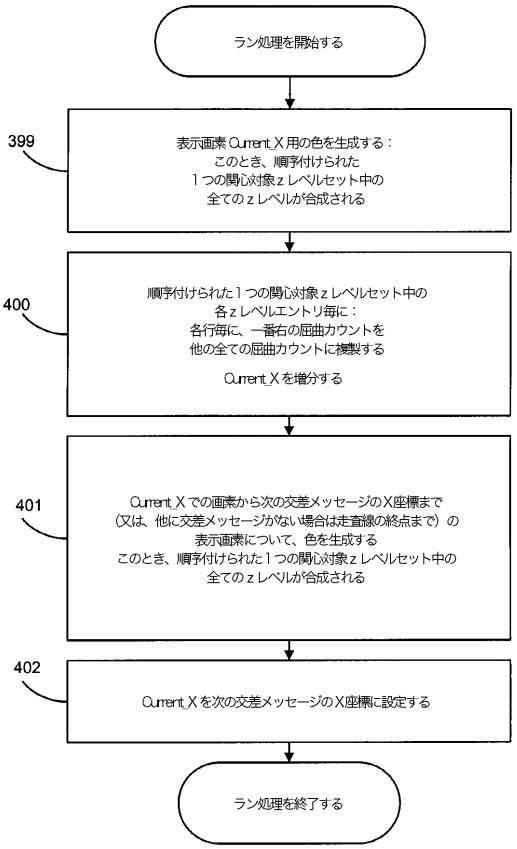


Fig. 40

【図４１】

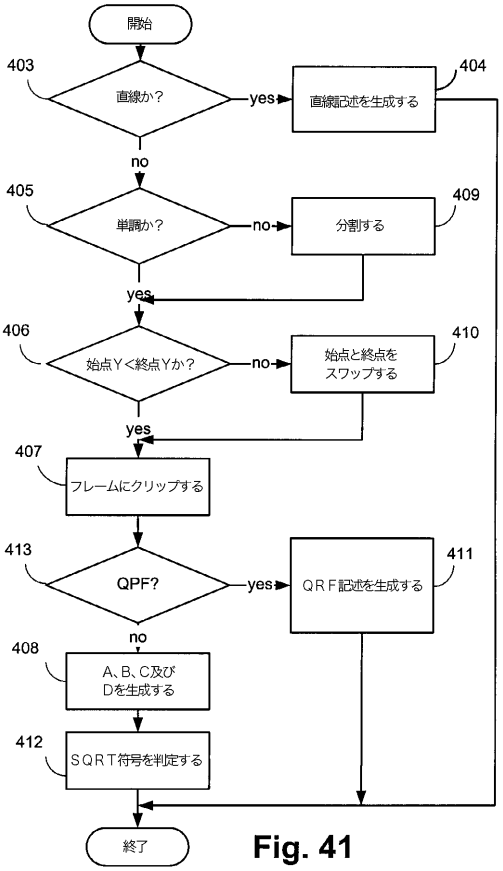


Fig. 41

【図４２】

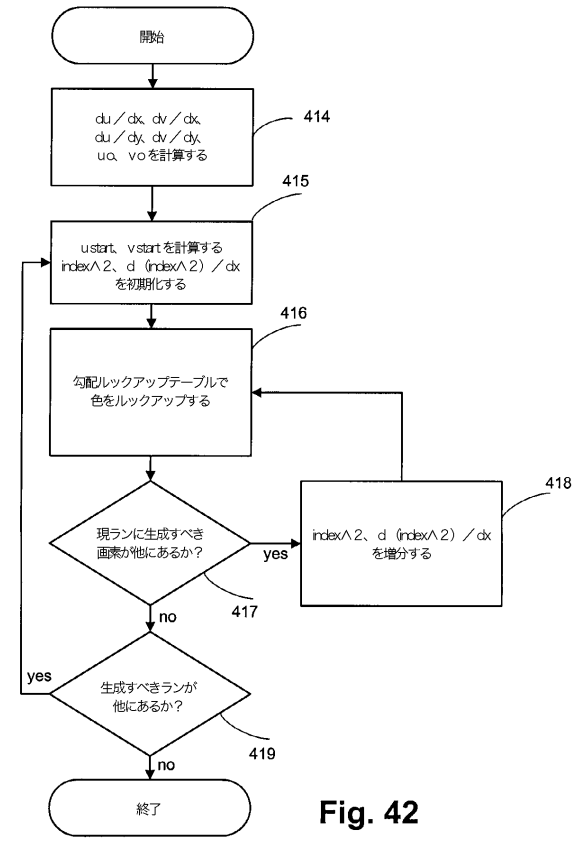


Fig. 42

【図４３】

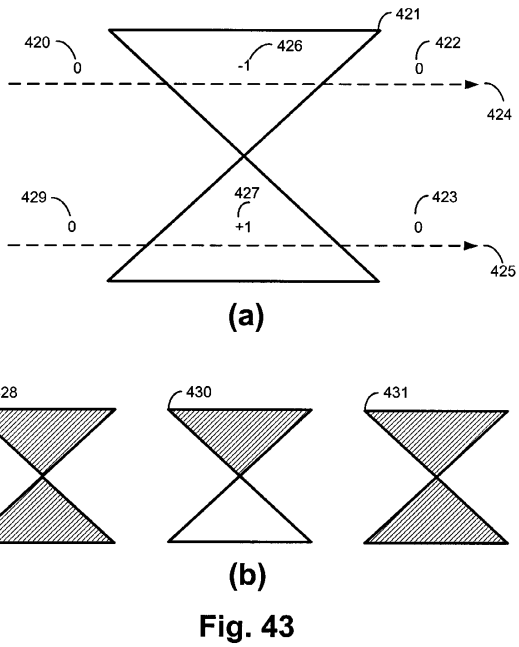


Fig. 43

【図４４】

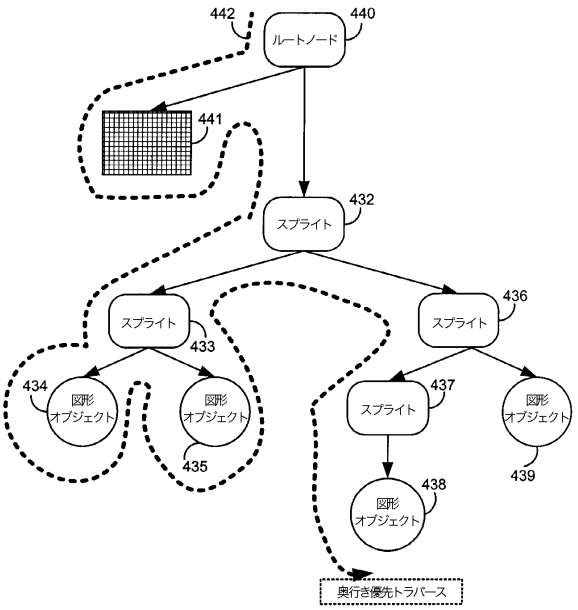


Fig. 44

【図４５】

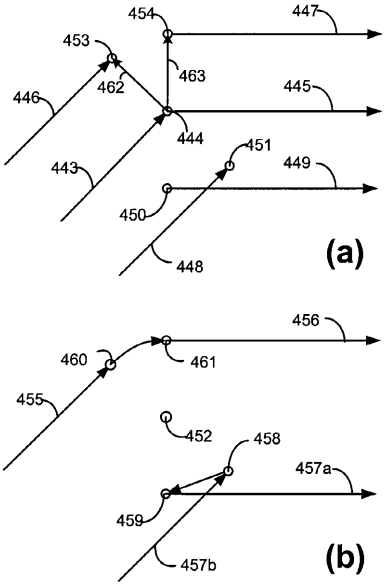


Fig. 45

【図４６】

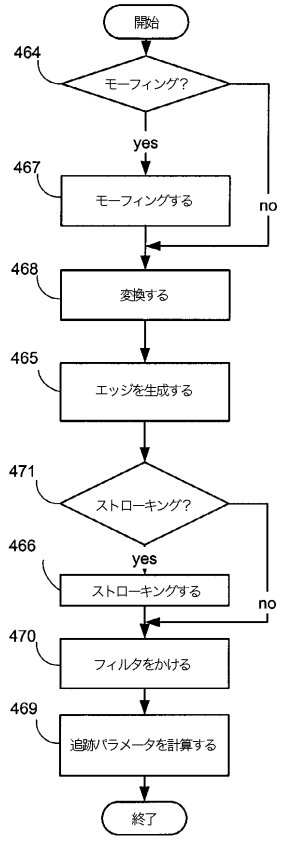


Fig. 46

【図４７－１】

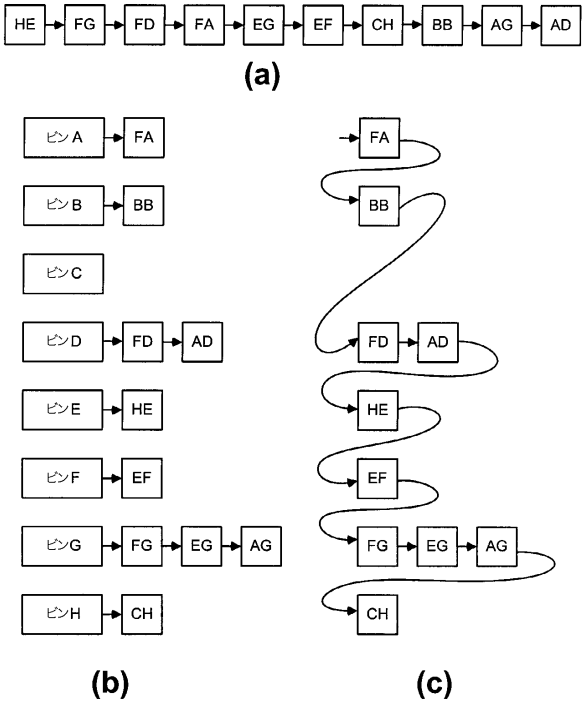


Fig. 47-1

【図４７－２】

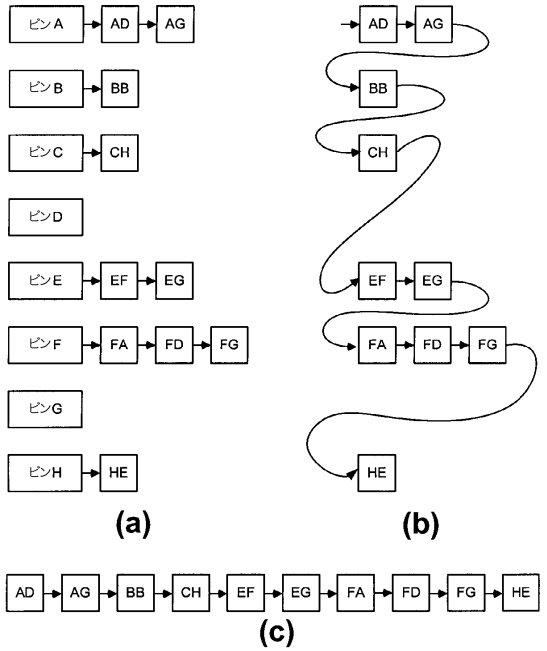


Fig. 47-2

【図４８】

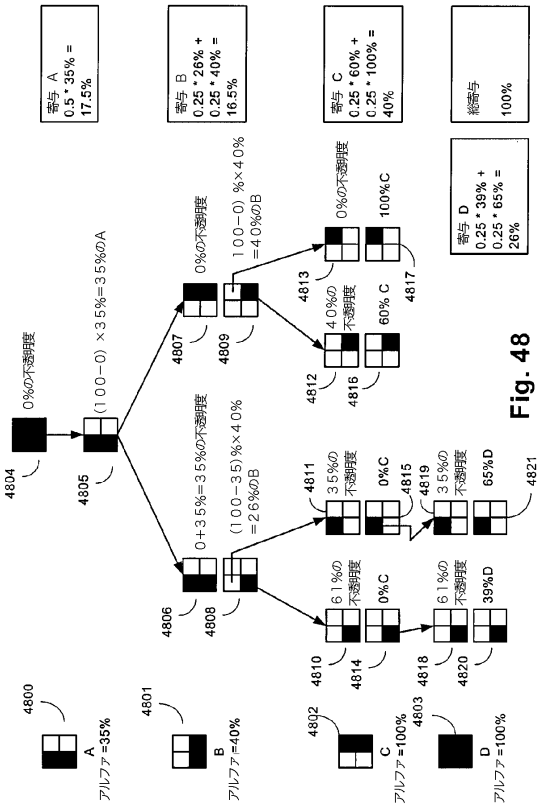


Fig. 48

【図４９】

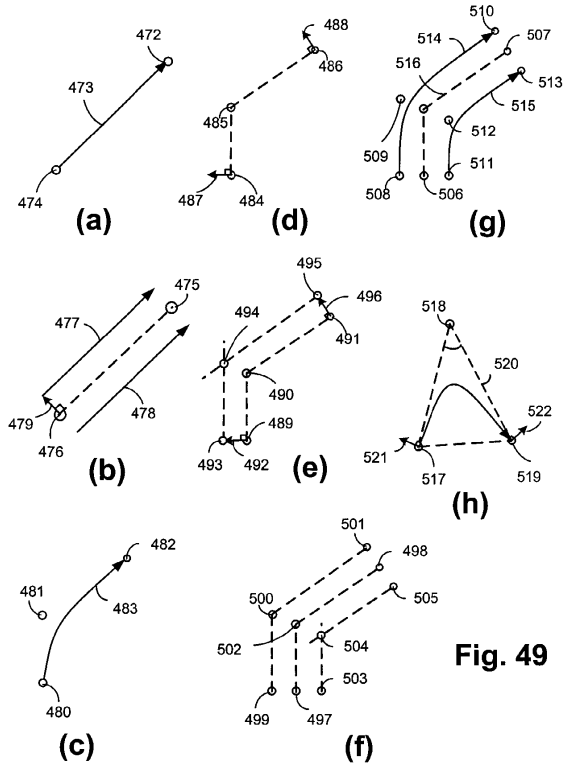


Fig. 49

【図５０】

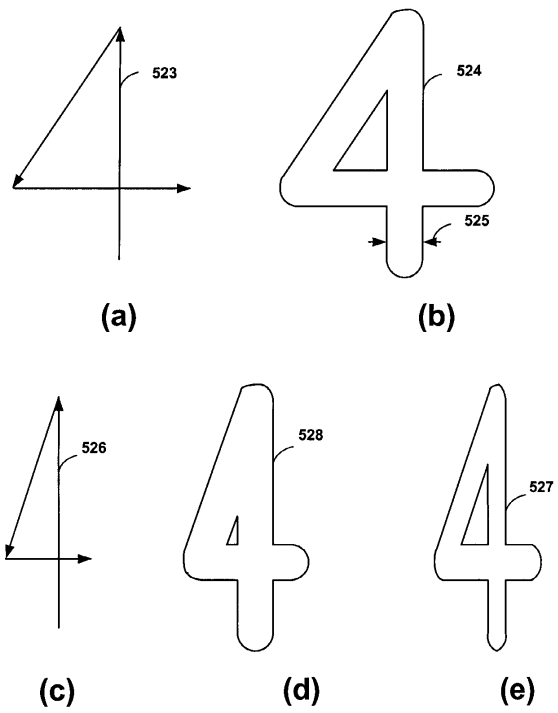


Fig. 50

【図５１】

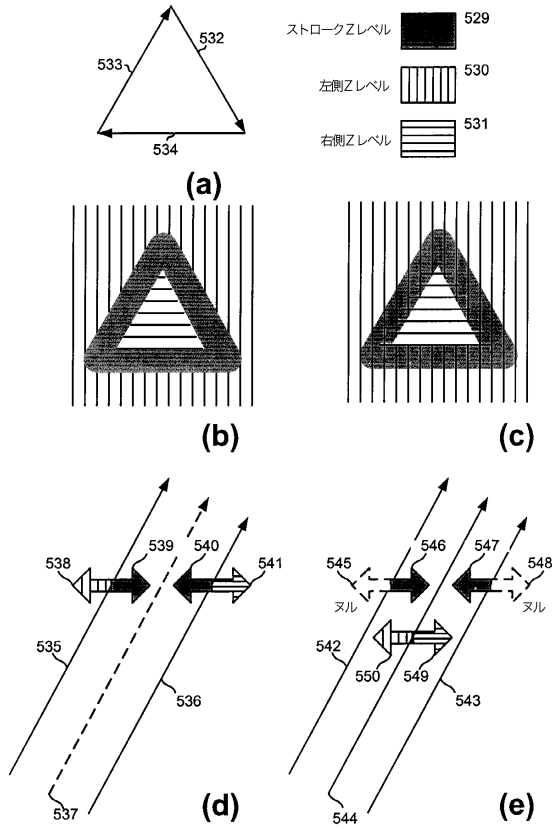


Fig 51

【図５２】

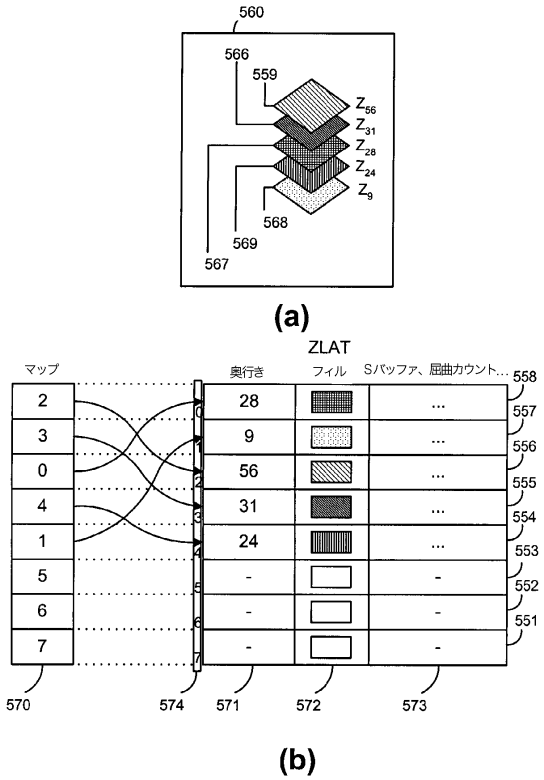


Fig. 52

【図５３】

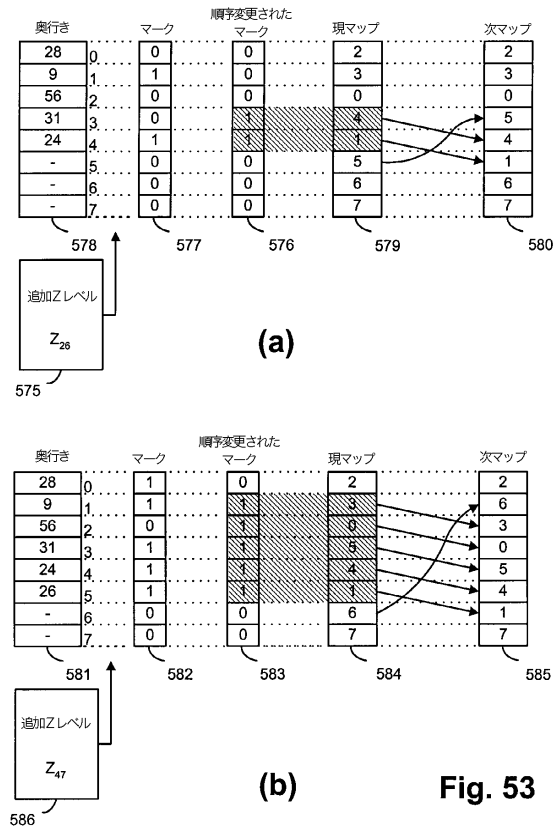


Fig. 53

【図５４】

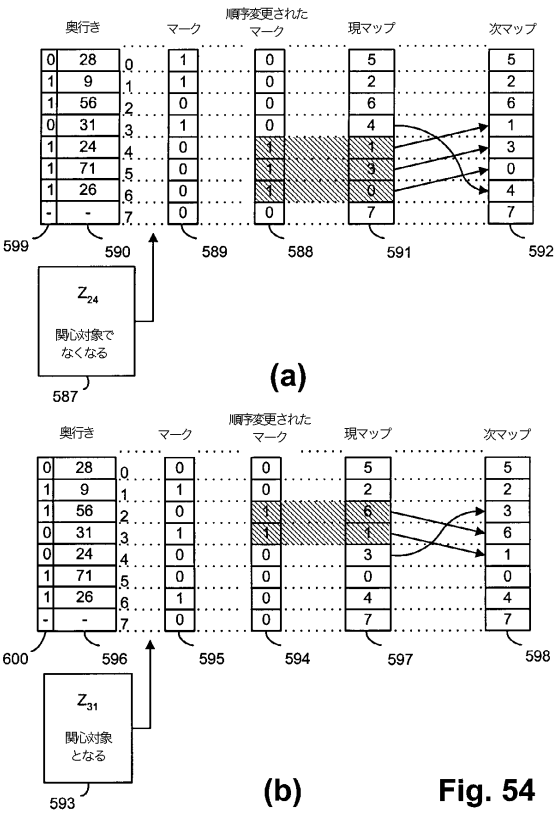


Fig. 54

【図５５】

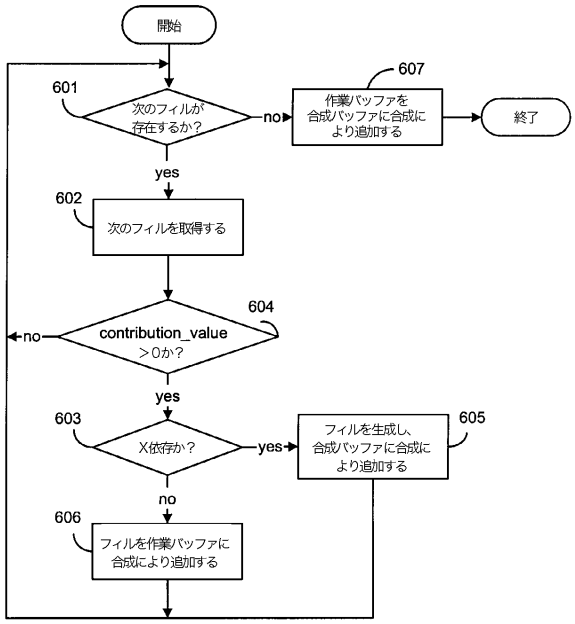


Fig. 55

【図５６】

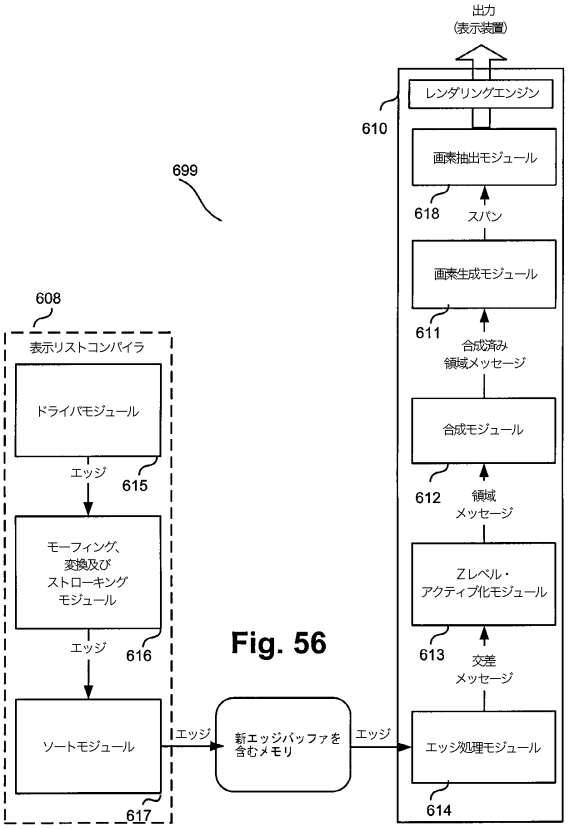


Fig. 56

【図５７】

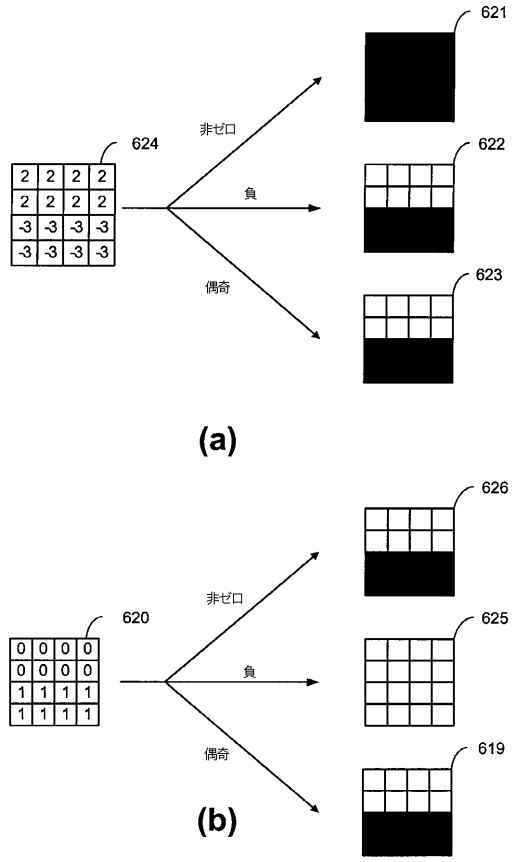


Fig. 57

【図５８】

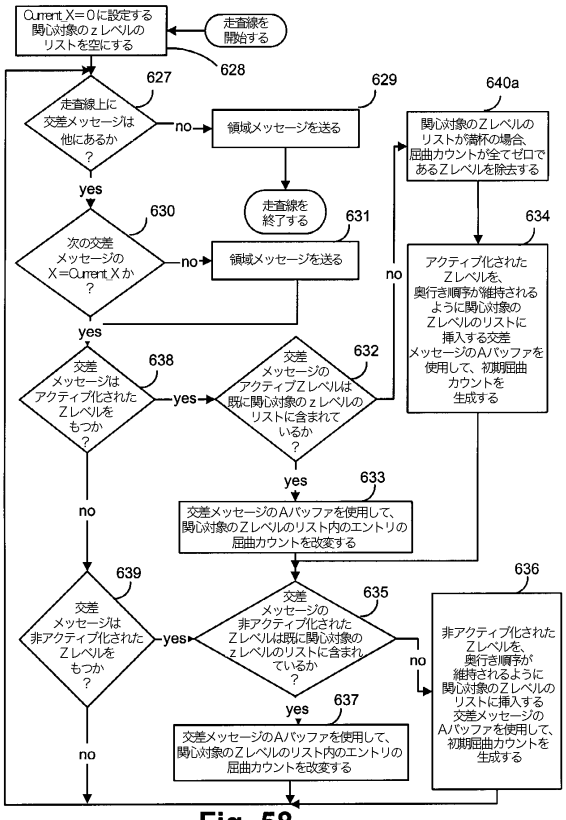


Fig. 58

【図５９】

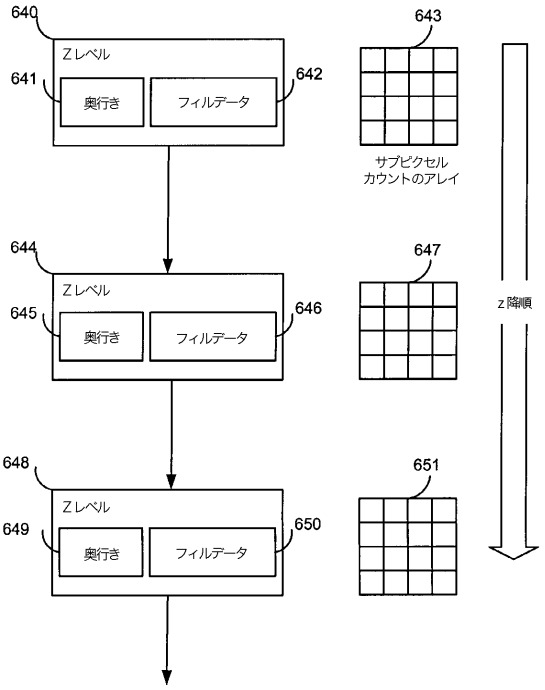


Fig. 59

【図６０】

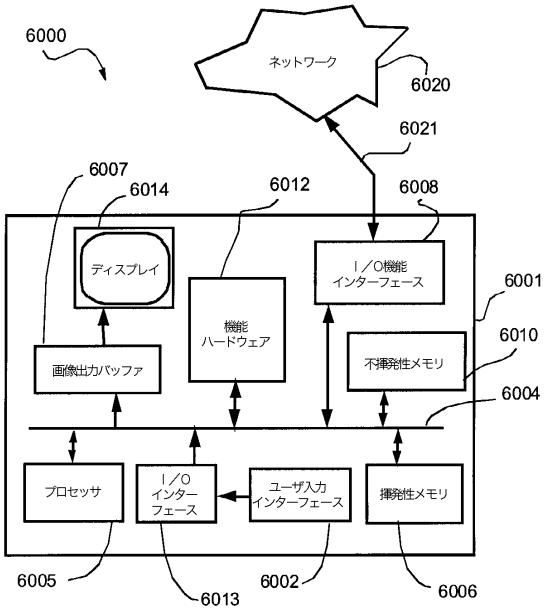


Fig. 60

【図６１Ａ】

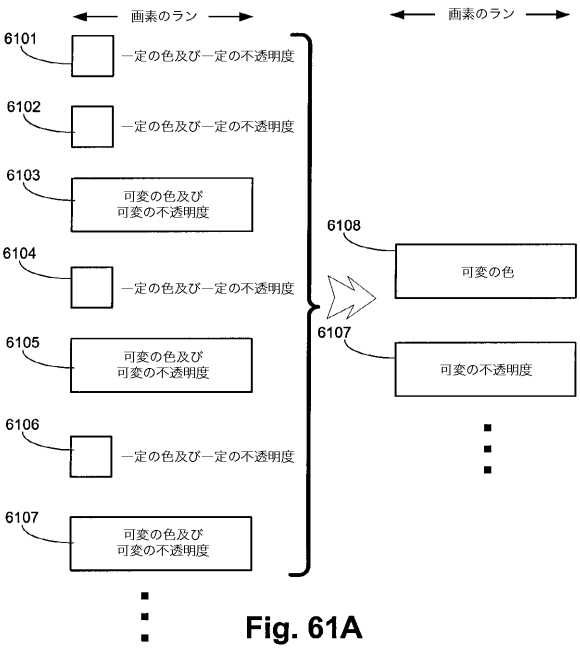


Fig. 61A

【図６１Ｂ】

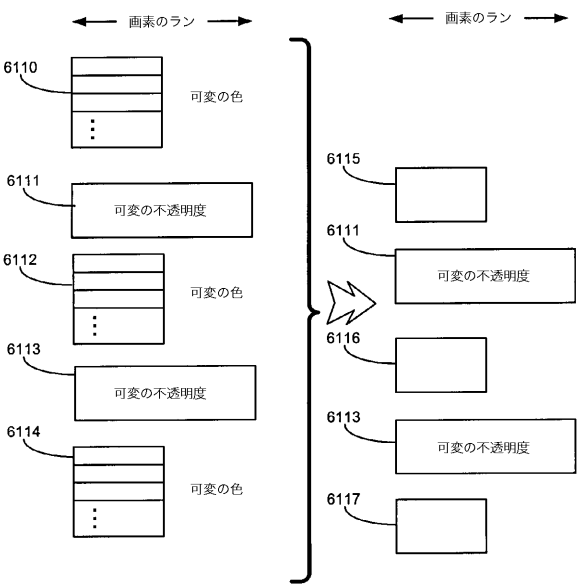


Fig. 61B

【図６２】

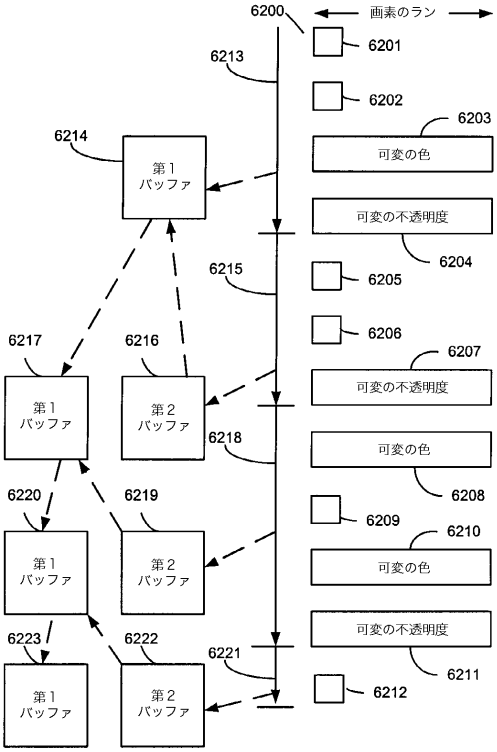


Fig. 62

【図６３】

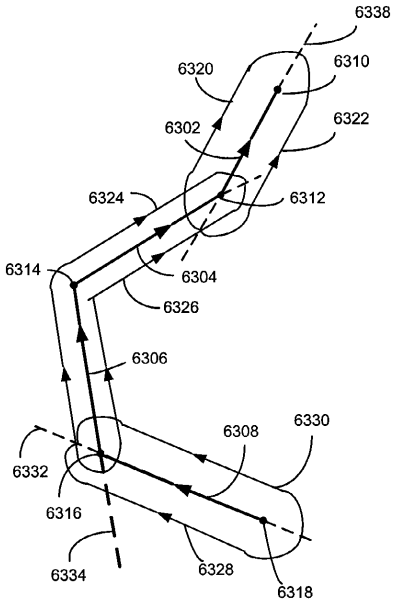


Fig. 63

【図６４】

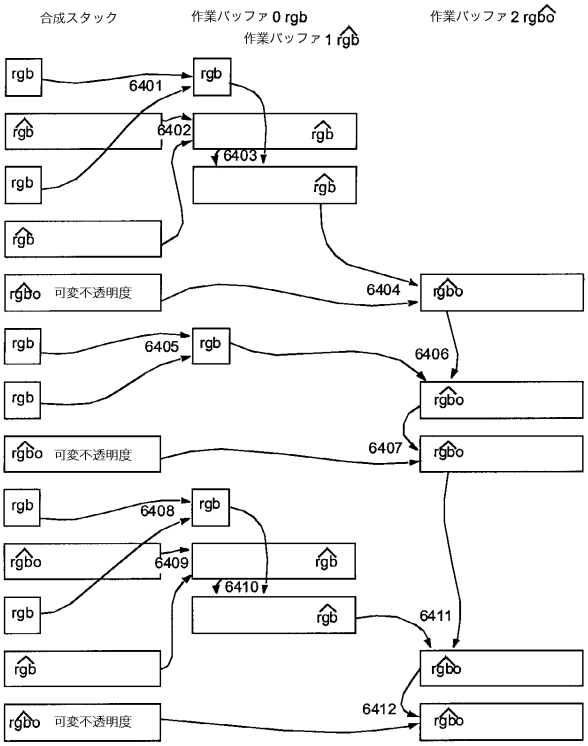


Fig. 64

【図６５】

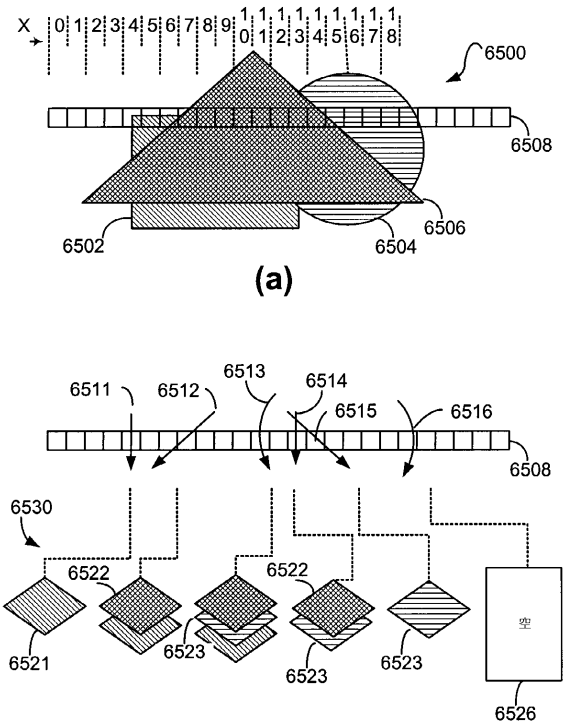


Fig. 65

【図６６】

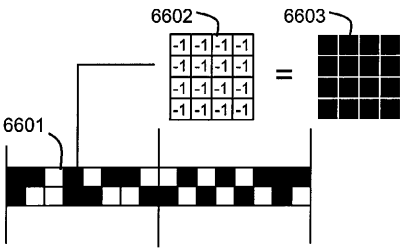


Fig. 66

【図６７】

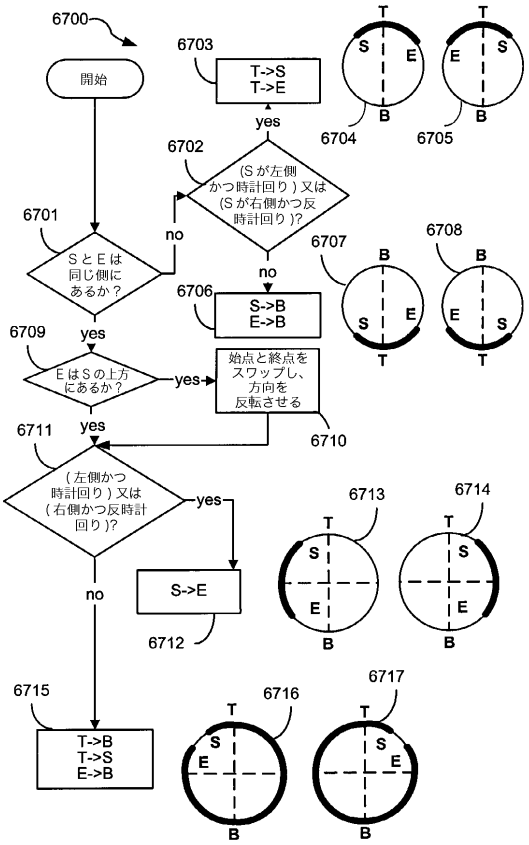
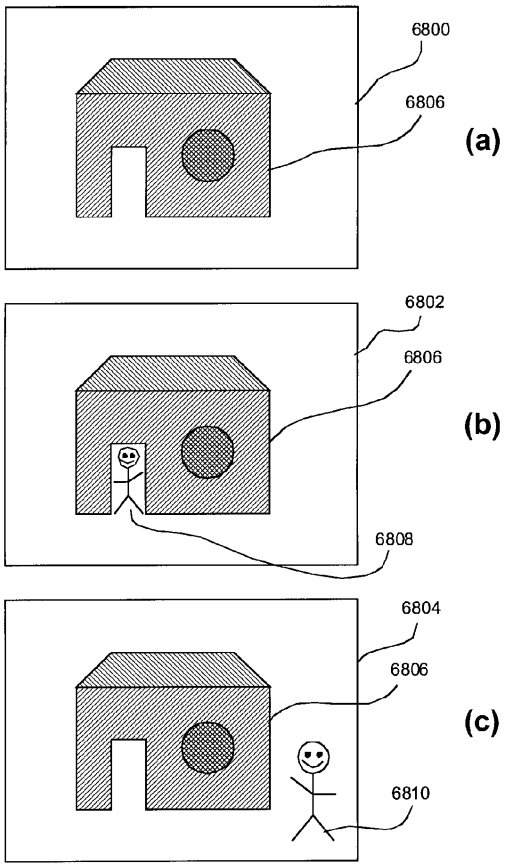


Fig. 67

【図６８】

Fig. 68



【図６９】

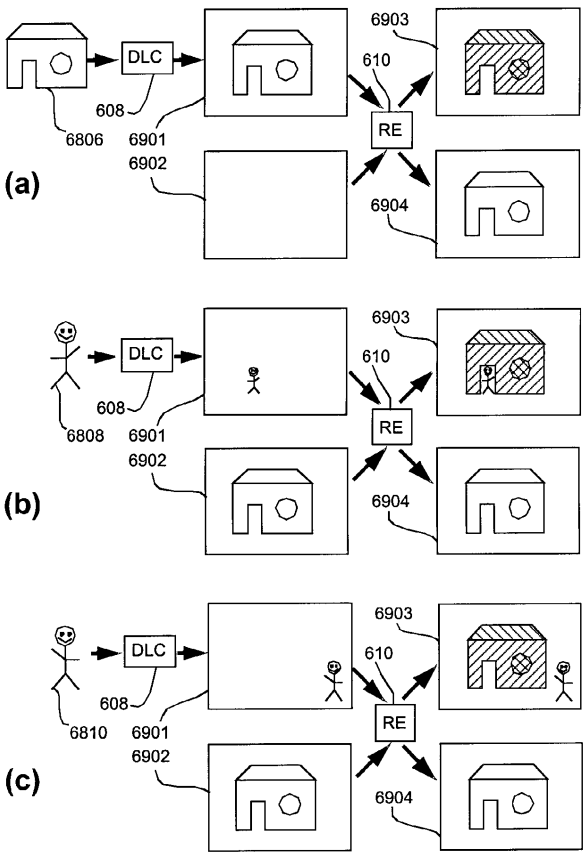


Fig. 69

【図７０】

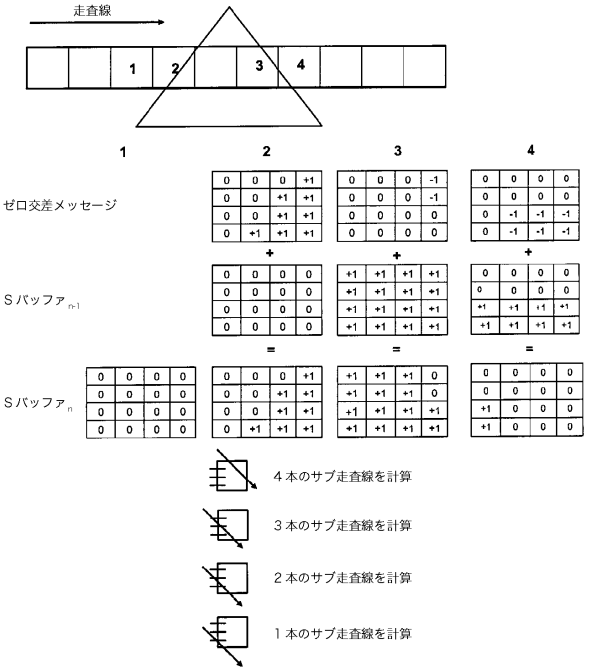


Fig. 70

【図７１】

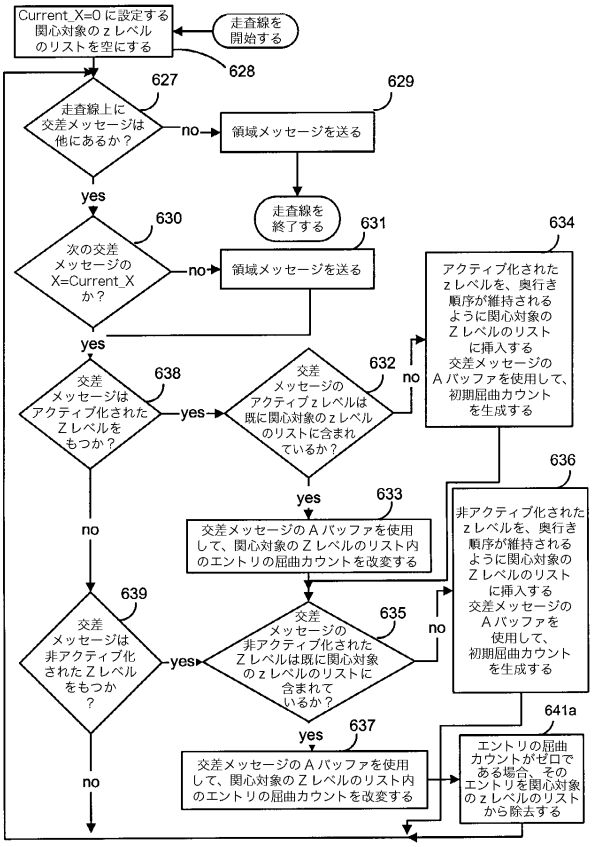


Fig. 71

フロントページの続き

(72)発明者 ロング, ティモシー, メリック
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 ブラッドリー, スコット
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 イーコブ, ステファン, エドワード
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 リーヴァー, ベンジャミン
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 田中 幸雄

(56)参考文献 特開昭６２－１５２０７７（ＪＰ，Ａ）
特開平１１－５１５１２１（ＪＰ，Ａ）
Ella Barkanほか, The scanline principle: efficient conversion of display algorithms into scanline mode, Visual Computer, Springer-Verlag, １９９９年, Vol. 15, No. 5, p249-264

(58)調査した分野(Int.Cl., ＤＢ名)
G06T 15/00