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

特許第4154336号
(P4154336)

(45)発行日　平成20年9月24日(2008.9.24)　(24)登録日　平成20年7月11日(2008.7.11)

(51)Int.Cl.　F I
G06T 11/40 (2006.01)　G06T 11/40 200
G06T 15/40 (2006.01)　G06T 15/40 200

請求項の数 9　(全 31 頁)

(21)出願番号	特願2003-564824 (P2003-564824)	(73)特許権者	000001007
(86)(22)出願日	平成15年1月31日 (2003.1.31)		キヤノン株式会社
(65)公表番号	特表2005-516315 (P2005-516315A)		東京都大田区下丸子３丁目３０番２号
(43)公表日	平成17年6月2日 (2005.6.2)	(74)代理人	100076428
(86)国際出願番号	PCT/JP2003/000994		弁理士　大塚　康徳
(87)国際公開番号	WO2003/065310	(74)代理人	100112508
(87)国際公開日	平成15年8月7日 (2003.8.7)		弁理士　高柳　司郎
審査請求日	平成18年1月24日 (2006.1.24)	(74)代理人	100115071
(31)優先権主張番号	PS 0287		弁理士　大塚　康弘
(32)優先日	平成14年2月1日 (2002.2.1)	(74)代理人	100116894
(33)優先権主張国	オーストラリア(AU)		弁理士　木村　秀二

最終頁に続く

(54)【発明の名称】ラスター画像のフレームを描画する方法及び装置

(57)【特許請求の範囲】

【請求項１】

各々が複数の画素を有する複数のラスター画像のフレームを描画する方法であって、

（ａ）保持する手段が、第１のフレームを描画し、該第１のフレームの画素のフィルランを記述する第１のデータを保持する工程と、

（ｂ）描画する手段が、前記第１のフレームの画素を更新するために第２のフレームを描画する工程と、

（ｃ）繰り返す手段が、後続のフレームに対して、前記工程（ｂ）を繰り返す工程とを備え、

前記第２のフレームを描画する工程は、

（ｂａ）決定する手段が、前記第２のフレームの画素のフィルランを記述する第２のデータを決定する工程と、

（ｂｂ）比較する手段が、前記第２のデータを前記第１のデータと比較する工程と、

（ｂｃ）上書きする手段が、前記比較の結果、異なる画素値であった場合に、前記第２のデータを用いて新しい画素を生成し、前記第１のフレームの画素を上書きする工程と、

（ｂｄ）更新する手段が、前記新しい画素に対応する前記第２のデータで前記第１のデータを更新する工程と、

（ｂｅ）再生する手段が、前記工程（ｂｂ）の比較の結果が同じ画素値であること

を示す場合、前記第２のフレームにおいて前記第１のフレームの対応する画素を再生する工程とを備えており、
前記フィルランの各々を記述する前記データは、少なくとも、前記ランの画素及び長さに寄与するグラフィックオブジェクトの優先度のリストを含み、
前記工程（ｂｂ）は、前記フレームのスキャンラインに沿った順序で前記フィルランに関するデータを比較することを含み、
前記比較の処理は、
（ｂｂａ）前記第２のデータの寄与するオブジェクトの優先度と前記第１のデータのそれらとを比較し、
優先度が同じであれば、（ｂｂａａ）対応するランの長さを比較し、それらが同じであれば前記工程（ｂｅ）は、（ｂｅａ）前記ランに対する前記第１のフレームの画素を再生して前記第２のデータを破棄し、それらが同じでなければ前記工程（ｂｅ）は、（ｂｅｂ)それら２つの長さのうちの短い方に対応するスパンにおける前記第１フレームの画素を再生し、前記第２のデータにおける前記ランの長さを前記短い方の長さに更新し、前記第２のデータの寄与する優先度と前記２つの長さの間の相違部分である残りの長さに対応する第２のデータから新しいランを形成し、
優先度が同じでない場合は、（ｂｂａｂ）前記第２のフレームに対する前記第２のデータから画素値を決定することを特徴とする方法。
【請求項２】
前記第１のデータは順序付けされたリストに保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】
前記リストは、最高優先度側の予め定められた数のオブジェクトに制限されていることを特徴とする請求項１に記載の方法。
【請求項４】
前記描画がスキャンライン順で実行されることを特徴とする請求項１乃至３のいずれか１項に記載の方法。
【請求項５】
請求項１乃至４のいずれか１項に記載の方法を実行するように構成された、一連の画像フレームを描画する装置。
【請求項６】
請求項１乃至４のいずれか１項に記載の方法に従ってコンピュータに一連の画像フレームを描画させるためのコンピュータプログラムが記録されたコンピュータ可読媒体。
【請求項７】
各々が複数の画素を有する一連のラスター画像フレームを描画する装置であって、
前記一連の画像フレームのうちの一つの画像フレームを描画し、該画像フレームの画素のフィルランを記述するデータを保持する描画部と、
前記一連の画像フレームにおける次の画像フレームの画素のフィルランを記述する更なるデータを決定する手段と、
前記更なるデータを前記保持されたデータと比較する手段と、
前記比較の結果が異なる画素値を表す場合に、前記更なるデータを用いて新しい画素を描画し、前記次の画像フレームを形成するために前記画像フレームにおいて画素を上書きする手段とを備え、
前記保持されたデータと前記更なるデータの各々は、対応する画像フレームにおける画素のランを備え、
前記比較する手段は、前記画像と更なるフレームの間の画素のランの類似性を判定するためにデータを比較し、
前記装置は、更に、
前記保持されたデータを、前記新しい画素に対応する前記更なるデータで更新する手段と、

前記比較する手段による結果が同じ画素値を表す場合、前記画像フレームの対応する画素を前記更なるフレームにおいて再生する手段と、
前記一連の画像フレームにおける後続のフレームのために前記装置の処理を繰り返す手段とを備え、
前記フィルランの各々を記述する前記データは前記ランにおける画素に寄与するグラフィックオブジェクトの優先度とランの長さを含み、
前記リストは、最高優先度側の予め定められた数のオブジェクトに制限され、
前記装置は、更に、前記フレームのスキャンラインに沿った順番で前記フィルランに対するデータを比較する手段を備え、前記比較する手段は更に、
前記更なるデータの寄与するオブジェクト優先度と前記画像データの優先度とを比較する第１手段と、
前記第１手段において前記優先度が同じであると判断された場合に動作可能であり、対応するランの長さを比較する第２手段と、
前記第２手段において前記長さが同じであると判断された場合に動作可能であり、前記更なるフレームにおける前記ランに対する前記画像フレームの画素を再生し、前記更なるデータを破棄する第３手段と、
前記第２手段において前記長さが異なると判断された場合に動作可能であり、前記画素フレーム中の、長さの短い方に対応する画素スパンにおける画素を再生し、前記短い方の長さによって該画素データにおける前記ランの長さを更新して、前記更なるデータから新しい画素のランを形成する第４手段と、ここで該新しいランは前記更なるデータの寄与する優先度の、前記２つの長さの相違による残り部分に対応し、
前記第１手段において前記優先度が異なると判断された場合に動作可能であり、前記次の画像フレームに対する前記更なるデータから画素値を決定する第５手段とを備えることを特徴とする装置。

【請求項８】
保持された前記データを順序付けされたリストとして保持する手段をさらに備えることを特徴とする請求項７に記載の装置。

【請求項９】
前記描画は、スキャンライン毎に実行されることを特徴とする請求項７または８に記載の装置。

【発明の詳細な説明】
【技術分野】
【０００１】
本特許明細書は著作権保護の対象となる内容を含む。著作権の所有者は、関連する特許庁のファイルからの本特許明細書或いは資料を検討を目的として複製することに異議を唱えるものではないが、他のいかなる場合にも全ての著作権を保有するものである。
【０００２】
技術分野
本発明はオブジェクトグラフィックエレメントをラスター画像に描画する技術に関する。特に、オブジェクトグラフィックエレメントの変化が生じる場合の効率的なフレームストアの更新に関する。
【背景技術】
【０００３】
大部分のオブジェクトベースのグラフィックシステムは、画素ベースイメージのページ或いは画面を保持するのに、フレームストア或いはページバッファを利用する。オブジェクトの外形が計算され、内部が満たされ、フレームストアに書き込まれる。２次元のグラフィックスのために、オブジェクトはイメージの特定のｚ－レベルで出現する。他のオブジェクトの前方に現れるオブジェクトは、単純に、背景となるオブジェクトの書き込みの後にフレームストアに書き込まれることにより、画素単位をベースとして背景オブジェクトを置き換えていく。これは、ペインターズアルゴリズム（Painter's Algorithm）とし

て本技術分野において公知のものである。そのようなオブジェクトは、最も背景側のオブジェクトから前方側のオブジェクトへと優先順に考慮される。典型的には、各オブジェクトはスキャンライン順にラスタライズされ、画素は各スキャンラインに沿って順番にフレームストアに書き込まれる。

【０００４】

この技術の問題点は、ペイントされた（すなわち描画された）多くの画素が、後のオブジェクトによって重複してペイントされることである。従って、そのような、先のオブジェクトに関してなされた画素のペインティングは、時間とコンピューティングリソースの浪費の要因となる。

【０００５】

上述したような重複ペインティングの問題を解決する技術がある。一つの技術では、画素は、オブジェクト単位をベースとするのではなく、全体画像ベースでラスター順に生成される。各スキャンライン上で、当該スキャンラインに交差する全てのオブジェクトのエッジがそのスキャンライン内の交差の座標を増加させる順に保持される。これらの交差ポイント、即ちエッジ交差は、順番に考慮され、アクティブフラグの配列のオン・オフを切り換えるのに使用される。スキャンライン上の注目の各オブジェクト優先度に対して一つのアクティブフラグがある。考慮されるエッジ対によってそれらの間の画素のスパン（範囲）が決定され、そのエッジ対の間に存在する各ピクセルのカラーデータがプライオリティエンコーダを用いて（或いは、ソフトウエアによって実現する場合はそれと等価なソフトウエアルーチンによって）生成される。プライオリティエンコーダはアクティブフラグに関して動作し、どのプライオリティが最高であるかを判断する。そして、そのプライオリティに関連するペイントを、２つのエッジの間のスパンの画素のために用いる。次のスキャンラインの準備において、各エッジの交差の座標が、各エッジの性質に従って更新される。例えば、単純な直線ベクトルに関して、次のスキャンラインの交差の座標を得るためにデルタｘ値が交差の現在の座標に加算される。この更新の結果ミスーストアになる隣接エッジは互いに入れ替えられる。新しいスキャンライン上でスタートするオブジェクトの新しいエッジもまた、エッジのリストにマージされる。この技術は、その開発者によって、クイセルアルゴリズム（Quixel Algorithm）と呼ばれている。

【０００６】

クイセルアルゴリズムは、重複ペインティングがないという重要な利点を有する。更に、ハードウエアによる実施においては、オブジェクト優先度は、オーダーＮタイムではなく（Ｎはプライオリティの数）、一定のオーダータイムで扱われる（通常は１クロックサイクル）。ソフトウエアによる実施においてさえも、時折発生するデータ依存の例外、すなわちlogNタイムはあるが、プライオリティは通常は一定時間で扱われ得る。これらの特性はクイセルアルゴリズムに、特に、重複するオブジェクトが存在する場合に、グラフィックオブジェクトのセットをラスター画像へ変換するための周知のペインターズアルゴリズムに対して大きなスピードの利点を与える。

【０００７】

対話型のグラフィックシステムにおいては、ＣＲＴあるいはＬＣＤ画面のような表示に対してリフレッシュされるフレームストアを有することは一般的である。そのようなシステムにおいて、表示上で表されるイメージは通常は高いフレーム同士の類似度を有する。すなわち、１つのフレームはその次のフレームと非常に似通っている。通常は、ディスプレイ上のイメージに寄与するオブジェクトグラフィックエレメントの一部分の集合のみが連続するフレーム間において変化する。コンピュータ的に負荷の大きい、画素の描画作業に関して実行が必要な量を最小化するために、フレーム間の高い類似度の利点を用いるべく、多くの技術が開発された。

【０００８】

グラフィックオブジェクトの集合によってディスプレイをリフレッシュするのにペインターズアルゴリズムを用いる場合、これらの技術は、通常は表示に寄与するオブジェクトグラフィックスにおいて生じた相違の監視を含む。バウンディングボックス或いはより複

雑な境界の記述はその相違の比較により生成され得る。それによってディスプレイ領域を、グラフィックオブジェクトの変化に対して変化せずに残るエリアと、変化してリフレッシュが必要な領域とに区切る。そして、オブジェクトグラフィックエレメントが描画される。しかし、通常は、リフレッシュ領域の完全に外にあるオブジェクトは、排除され、画素の生成はリフレッシュ領域内でのみ生じる。

【０００９】

この技術はディスプレイのリフレッシュ時間を大きく減少することができるが、依然としていくつかの欠点を有する。たとえば、一般に、大きなオブジェクトの小さい部分が変化することがある。実際に変化した領域を決定するためにオブジェクトの内部解析をすることはコンピュータにとって多大な負荷であり、そのため、過剰に大きなリフレッシュ領域が見積もられてしまう。更に、その大部分の画素に対して最終イメージにおいて変化のないオブジェクトグラフィックエレメントに変更が加えられることがある。例えば、大きな赤い矩形を数画素動かす場合、大部分の画素は赤いままである。繰り返すが、そのようなケースを検出するための全オブジェクトの内部解析はコンピュータにとって多大な負荷であり、過剰に大きなリフレッシュ領域が用いられる。同様な厄介な状況はよく起こることである。これらの技術は依然として、ペインターズアルゴリズムが有する、重複ペイントの非効率性という問題を抱えている。

【００１０】

説明はしなかったが、そのような技術は、重複ペイントの非効率性を軽減するために、クイセルアルゴリズムに適用できる。しかし、それらはなお他の課題を有している。

【００１１】

本発明の目的は、既知の構成の１つ又は複数の欠点を実質的に解消或いは改善することにある。

【発明の開示】

【課題を解決するための手段】

【００１２】

本発明の第１の観点によって提供される方法は、オブジェクトグラフィック要素から一連のラスター画像フレームを描画する方法であって、少なくとも一つの古いフィルランが第１のフレームの描画の間に保持され、前記保持されたフィルランが後続のフレームのための少なくとも一つの新しいフィルランと比較され、少なくとも一つの新しいフィルランにおいて当該新しいフィルランの少なくとも部分に対する画素データの生成を抑制し、代わりに、前記第１のフレームから保持された画素を用いる。

【００１３】

また、好ましくは、前記保持されたフィルランの記述は順番付けられたリストに格納される。更に好ましくは、前記保持されたフィルランの記述の数は、第１のフレームの完全な再生に必要な数よりも少なく制限される。

【００１４】

本発明の第２の観点によれば、各々が複数の画素を有する複数のラスター画像のフレームを描画する方法であって、

（ａ）第１のフレームを描画し、該第１のフレームの画素のフィルランを記述する第１のデータを保持する工程と、

（ｂ）前記第１のフレームの画素を更新するために第２のフレームを描画する工程とを有し、前記第２のフレームを描画する工程は、

（ｂａ）前記第２のフレームの画素のフィルランを記述する第２のデータを決定する工程と、

（ｂｂ）前記第２のデータを前記第１のデータと比較する工程と、

（ｂｃ）前記比較の結果、異なる画素値であった場合に、前記第２のデータを用いて新しい画素を生成し、前記第１のフレームの画素を上書きする工程とを備える方法が提供される。

【００１５】

更に本発明の他の観点によれば、上述した方法のいずれかを実施するコンピュータプログラムが記録され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プロダクトが提供される。

【００１６】

本発明の他の態様も開示されている。これらは画素のランを用いた最適化されたサーバ構成や、高速なレンダリングのためにそのような最適化されたデータをサーバから受信するように構成された遠隔デバイスを含む。

【００１７】

上述した目的は、好ましくは、クイセルアルゴリズムを改良することにより達成される。そのような改良は、第１のフレームの描画の間に、あるランの画素フィルのランの情報が保持される。そして、次のフレームの描画の間に、これらのランは、その新しいフレームを生成するのに用いられる新しい画素フィルのランの情報と比較される。その比較の結果が、既に描画されたフレームに存在するがそのスパンがその要求された値を既に有することを示す場合、これらのスパンの画素のフィル動作が回避される。また、画素フィルラン情報の新しいリストは後続のフレームに対して処理が繰り返されるように保持される。

【発明を実施するための最良の形態】

【００１８】

１つ又はそれ以上の図面において同一の参照番号が付されたステップ及び／又は機構が参照されるが、それらのステップ及び／又は機構は、特に断りのない限り、本発明の目的に対して同一の機能或いは動作をするものである。

【００１９】

図１は、従来技術における、上述のクイセルアルゴリズムに基づく描画装置１００を示す。図１において、グラフィックオブジェクト記述１０２は、ディスプレイリストコンパイラモジュール１１０に入力される。ディスプレイリストコンパイラモジュール１１０は、個々のグラフィックオブジェクトを解釈し、描画を要求された個々のイメージの１つ又はそれ以上のディスプレイリスト１１２をコンパイルして格納する。通常は、画像は表示可能なシーケンスを形成し、それにより、グラフィックオブジェクトシーンのアニメーションを表現する。表示可能なシーケンスが形成されると、各ディスプレイリスト１２２は、一連のフレームの１つを提供するために描画される。描画において、エッジ追跡モジュール１２０は、アクティブエッジのリスト１２２を決定するために、最初にディスプレイリスト１１２内の描画中のイメージを形成するオブジェクトを調べる。エッジのアクティビティがラスタースキャン順に決定され、ｚ－レベルアクティベーションモジュール１３０に提供される。ｚ－レベルアクティベーションモジュール１３０は、各スキャンライン上で交差する各エッジについて、描画されたイメージにおける特定のスキャンライン上の、隣接するエッジ対の間のスパンにおいてアクティブなオブジェクトを決定する。これは、テーブル１３２の補助を用いて決定される。テーブル１３２内のスパンに関るエントリに対して、オブジェクトはそれらの優先度或いはｚ－オーダーでランク付けされる。テーブル１３２における最高位の不透明オブジェクトが、ｚ－オーダーに関して下位にあたる全てのオブジェクトを排除するために動作する。そして、そのオブジェクトは、高位オーダーの透明オブジェクトとともに、そのスパンに関して、フィル生成モジュール１４０に出力される。モジュール１４０は、モジュール１３０から出力された各オブジェクトに関してフィルカラーを見出すためにテーブル１４２を調べる。合成モジュール１５０は、スパンを横切る種々のオブジェクトに対するフィル値から、実際の画素値を合成するべく動作する。そのスパンの画素値は、フレームストア１６０に出力される。次のスキャンラインへ移動する前に、描画はスキャンライン上の各スパンについてなされ、この処理は、フレームストアがディスプレイに出力される画素データのフレームで満たされるまで行われる。

【００２０】

図２は、本実施形態に従った処理２１０を含むように図１の構成が変更された状態の描画部２００を示す。この好ましい実施形態を、本明細書では「シンクライアントイメージ

ングエンジン（thin client imaging engine）」或いは「ＴＣＩＥ」と称する。図２において、ランカリングモジュール２１０と、それに関連して保持されたランリスト２２０が、描画処理におけるｚ－レベルアクティベーションモジュール１３０とフィル生成モジュール１４０の間に挿入される。定性的には、ランカリングモジュール２１０は図１０に示される様式で動作する。以下、これを説明する。

【００２１】

図１０は三角形１０００により形成されるイメージを示す。三角形１０００は三角形１０２０と部分的に重なり、重なった部分を覆っている。三角形１０００はエッジ１００２，１００４及び１００６によって形成され、三角形１０２０はエッジ１０２２，１０２４，１０２６によって形成される。図には、第１のフレームにおけるスキャンラインがＡで示されている。このスキャンラインは、イメージの左周辺とエッジ１００２の間のスパンＡ１、エッジ１００２と１００６の間のスパンＡ２、エッジ１００６と１０２６の間のスパンＡ３、そしてエッジ１２２６とイメージの右周辺との間のスパンＡ４を有する。スパンＡ１～Ａ４の各々のラン長（ランレングス）は、注目しているスキャンラインとのそれぞれのエッジのＸ交差から数学的に決定されてもよい。

【００２２】

後続の第２のフレームにおいて、三角形１０００は、エッジ１００６を置き換える新たなエッジ１０１０によって、図１０に示すように形状を変える。その結果として、エッジ１００４の延長１００８が現れる。第２のフレームにおける同一のスキャンラインをＢとする。スキャンラインＢは、左外周とエッジ１００２の間のスパンＢ１、エッジ１００２と１０１０の間のスパンＢ２、エッジ１０１０と１０２６の間のスパンＢ３、及びエッジ１０２６と右外周との間のスパンＢ４を有する。スパンＢ１～Ｂ４の各々のランレングスは上述のようにして決定される。

【００２３】

ランカリングモジュール２１０は、まず最初のフレームに対して、スパンＡ１，Ａ２，Ａ３およびＡ４の種々の詳細をリスト２２０内に保持するべく動作する。次のフレームの同じスキャンラインを処理するときに、ランカリングモジュール２１０はフレームストア１６０内の当該スキャンラインの画素値を決めるのに用いられる。それらの画素値はスパンのあらゆる変化により変更されることが要求される。これはスパンＢ１，Ｂ２，Ｂ３およびＢ４とランリスト２２０に格納されたそれらとの比較を通じてなされる。スパンは好ましくはそれらが生成された順序、すなわちラスター順で処理される。この例において、スパンＢ１はＡ１と比較される。これらは一致するので、スパンＢ１はイメージの変化に何等寄与せず、現在のレンダリングから破棄されることになる。一方、スパンＡ１はフレーム格納部１６０に格納されているためにそのまま表示され続け、次のフレームの処理のためにランリスト２２０に維持される。本説明において、スパンの破棄をカリングと称し、スパンの維持をコンスーミングと称する。

【００２４】

次に、スパンＢ２はスパンＡ２と比較される。これらは同じスタート位置を有するが、Ｂ２の方が長い。したがってＡ２はコンスーミングされ、Ａ２に対応するＢ２の部分はカリングされる。そしてエッジ１００６と１０１０の間のスパンを表す新たなスパン$B2^1$を生成する。次に、$B2^1$がＡ３と比較され、相違が見出される。したがって、$B2^1$はレンダリングのためにフィル生成モジュール１４０へ渡され、ランリスト２２０に格納される。Ａ３は$B2^1$よりも長いので、リスト２２０におけるＡ３の表現は、エッジ１０１０と１０２６の間のスパンである$A3^1$へと縮められる。

【００２５】

次に、スパンＢ３がスパン$A3^1$と比較される。これらは同じエンドポイントを有するので、Ｂ３はカリングされ、$A3^1$はコンスーミングされる。次にスパンＢ４がＡ４と比較される。これらは同一であるため、Ｂ４がカリングされ、Ａ４はコンスーミングされる。

【００２６】

図１０の例では、イメージの部分を形成するオブジェクトが変化したが、その変化はランカリングモジュール２１０によって解釈され、注目のスキャンラインの部分だけを実際にレンダリングすることを必要とさせる。本例において、その部分は、エッジ１００６と１０１０との間のスパンである。こうして、スキャンライン上の多数の画素のためのフィル生成と合成を回避することができ、それによりレンダリングスピードの改善が図られる。

【００２７】

図１と図２の構成は、図３に示すような汎用コンピュータシステム３００を用いて実現することができる。その場合、図１、図２に示したプロセスは、コンピュータ３００内で稼動するアプリケーションプログラムのようなソフトウエアで実施することができる。特に、図１と図２の処理ステップは、コンピュータによって実行されるソフトウエアのインストラクションによって実現される。ソフトウエアは２つの別の部分に分けられる。一方の部分はレンダリング処理を実行し、他の部分は後者とユーザの間のユーザインターフェースを管理する。これらのパーツはさらにプロセスを実現するソフトウエアコードのモジュールと、上述した、或いは後述されるようなメソッドに分けられる。ソフトウエアは、コンピュータ可読媒体に格納され得る。コンピュータ可読媒体としては、たとえば、以下に説明する格納デバイスが含まれる。ソフトウエアは、コンピュータ可読媒体からコンピュータへロードされ、コンピュータによって実行される。そのようなソフトウエアあるいはコンピュータプログラムが記録されたコンピュータ可読媒体は、コンピュータプログラムプロダクトである。コンピュータにおけるそのコンピュータプロダクトの使用は、グラフィックオブジェクトの動画レンダリングのための有利な装置をもたらす。

【００２８】

コンピュータシステム３００は、コンピュータモジュール３０１、キーボード３０２およびマウス３０３のような入力デバイスと、プリンタ３１５およびディスプレイデバイス３１４を含む出力デバイスを有する。変調－復調（モデム）トランシーバデバイス３１６は、通信ネットワーク３２０との間の通信のためのコンピュータモジュール３０１によって使用される。モデム３１６は、たとえば、電話回線３２１もしくは他の機能メディアを介して接続可能である。モデム３１６はインターネットおよびローカルエリアネットワーク（ＬＡＮ）あるいはワイドエリアネットワーク（ＷＡＮ）のような他のネットワークシステムへのアクセスを取得するのに用いることができる。この例において、ネットワーク３２０は移動電話のハンドセット３５０に接続する。ここで、このハンドセットは、画素ベースの比較的大きいディスプレイ画面３５２を有する。コンピュータモジュール３０１は、いくつかの実施形態において、ネットワーク３２０において動作可能なサーバコンピュータをあらわす。

【００２９】

コンピュータモジュール３０１は、典型的には、少なくとも１つのプロセッサユニット３０５、たとえば半導体ランダムアクセスメモリ（ＲＡＭ）とリードオンリメモリ（ＲＯＭ）で形成されたメモリユニット３０６、および入出力インターフェースを含む。入出力インターフェースは、ビデオインターフェース３０７や、キーボード３０２やマウス３０３、また、オプションとしてのジョイスティック（不図示）を含むＩ／Ｏインターフェース３１３や、モデム３１６のためのインターフェース３０８を含む。格納デバイス３０９が提供され、それは典型的にはハードディスクドライブ３１０とフロッピー（登録商標）ディスクドライブ３１１を含む。磁気テープドライブ（不図示）も利用可能である。典型的には、データの不揮発性ソースとしてＣＤ－ＲＯＭドライブ３１２が提供される。コンピュータモジュール３０１の構成要素３０５～３１３は、典型的には相互に接続されたパス３０４を介して、当業者には既知の、コンピュータシステム３００の動作の一般的なモードで通信する。上述した構成を実現し得るコンピュータの例としては、ＩＢＭ－ＰＣコンパチブル、サンスパークステーション、あるいはそれらから展開されたコンピュータシステム等が挙げられる。

【００３０】

典型的には、アプリケーションプログラムは、ハードディスクドライブ３１０に常駐し、その実行時にプロセッサ３０５によって読まれ、制御される。プログラムの中間的な格納、およびネットワーク３２０からフェッチされたあらゆるデータの格納は、半導体メモリ３０６を用いて、場合によってはハードディスク３１０と共同して達成され得る。いくつかの例において、アプリケーションプログラムはＣＤ－ＲＯＭあるいはフロッピー（登録商標）ディスクにエンコードされてユーザに提供され、対応するドライブ３１２あるいは３１１を介して読み取られる。あるいは、アプリケーションプログラムは、モデムデバイス３１６を介してネットワーク３２０からユーザによって読み取られてもよい。なお、ソフトウエアは、他のコンピュータ可読媒体からコンピュータシステム３００へロードされ得る。そのようなコンピュータ可読媒体は、たとえば、磁気テープ、ＲＯＭあるいは集積回路、光磁気ディスク、コンピュータモジュール３０１と他のデバイスとの間の無線あるいは光／赤外線伝送チャンネル、ＰＣＭＣＩＡのようなコンピュータ可読カード、インターネットやイントラネットのようなネットワークを含み、また、それによってＥメール伝送およびウエブサイト等に記録された情報とが含まれる。上記は適切なコンピュータ可読媒体の好例に過ぎず、他のコンピュータ可読媒体も利用可能である。

【００３１】

あるいは、図１、図２の構成は、オブジェクトベースレンダリングの機能またはサブ機能を実行する１つまたは複数の集積回路のような専用のハードウエアで実現されてもよい。そのような専用のハードウエアはグラフィックプロセッサ、あるいは１つまたはそれ以上のマイクロプロセッサと関連するメモリを含むことができる。

【００３２】

図４は、シンクライアントイメージングエンジン（thin client imaging engine、ＴＣＩＥ）４１０を含む、レンダリングシステム４００のデータフローを示す。ＴＣＩＥ４１０は、２つのメイン機能ユニットを有する。第１の機能ユニットは、ディスプレイリストコンパイラ４２０であり、第２の機能ユニットは、レンダリングエンジン４３０である。ディスプレイリストコンパイラ４２０は、複数の命令を収容するメモリ４５４、複数のオブジェクトを収容するメモリ４５６及び複数のフィルを収容するメモリ４５８に格納されたディスプレイオブジェクトデータをフェッチし解釈する。メモリ４５４，４５６，４５８はＲＡＭ３０６によって実現され、それらに格納されるコンテンツはホストプロセッサ４５０によって生成される。また、ホストプロセッサ４５０は、ＴＣＩＥ４１０との間でコントロールデータやステータスデータをやり取りする。

【００３３】

図５Ａから図５Ｃは、図５Ｃにおいて表示されるオブジェクトの例を示し、そのコンポーネントエッジを図５Ａに示し、図５Ｂにフィルスタイルを示す。オブジェクトは、２次元ディスプレイプリミティブである。それらは、複数のエッジリストによってメモリに記述され、各エッジリストは複数の座標によって記述される。座標は新たな描画位置と直線を記述する。新たな描画位置は、単一の座標で記述され、直線は座標のペアによって記述される。直線は線のスタートポイントとエンドポイントを定義する２つの座標を用いる。曲線は、２次のベジェ曲線を用いて実施することができ、この場合、第１の座標はベジェ曲線のスタートポイントを定義し、第２の座標は制御点を定義し、第３の座標はベジェ曲線のエンドポイントを定義する。エッジリストの座標は、一連の相対的ステップとして格納される。これは、格納に必要なメモリ量を低減し、また、エッジの方向を決定する。エッジリストは、集団によってある形状の外形を現す。オブジェクトは、それら自身の座標空間を有し、それゆえ、それらに独自の起点を有し、そこからエッジは相対的に描画される。図５Ａにおいて、ポイント５０２は“家”オブジェクトの起点を表す。オブジェクトは常に２つの付加的な特別にマークされた境界座標を含む。それら２つの境界座標は他の座標とは異なり、オブジェクトの部分をどのように表示するかといったことを記述するものではなく、その中にすべての描画エッジを含むバウンディングボックス（境界箱）を示すものである。第１の境界座標は、バウンディングボックスの左上隅を特定し、第２の境界座標はバウンディングボックスの右下隅を定義する。

【００３４】
フィルは、オブジェクトのエッジリストの部分集合によって囲まれたディスプレイの部分をどのように色づけするかを記述するのに用いられるディスプレイプリミティブである。たとえば、基本フィルは赤のようなべた色を記述する。各エッジリストには２つのフィルが関連付けられている。第１のフィルはそのエッジリストの描画方向に対して左側に描かれ、第２のフィルはそのエッジリストの描画方向に対して右側に描かれる。フィルの主なスタイルは、単純色、複数色によって表現されるリニアブレンド（linear blend）、複数色によって表現されるラジアルブレンド（radial blend）、あるいはビットマップイメージである。これらのフィルスタイルのすべては、透明チャンネルをサポートする。なお、エッジがその左および右側の両方においてフィルを参照しない場合、フィル＝０の値が用いられる。

【００３５】
図５Ａ～５Ｃにおいて、エッジ５０４は、左フィル＝２、右フィル＝０の直線エッジベクトルである。エッジ５０６は左フィル＝３、右フィル＝２の直線エッジベクトルである。エッジ５０８は左フィル＝２、右フィル＝１の直線エッジベクトルであり、エッジ５１０は左フィル＝０、右フィル＝１の直線エッジベクトルである。

【００３６】
図４に戻り、メモリ４５４に格納されたインストラクション（命令）は、メモリ４５６に格納されたオブジェクト４６２がいつ、どのように出力デバイス４７０に描画されるかを記述する。ディスプレイリストコンパイラ４２０は、命令によって指示されたとおりにオブジェクトのデータを処理し、この処理の結果をディスプレイリストデータ４２２として、レンダリングエンジン４３０による利用のためにメモリ４４０に配置する。レンダリングエンジン４３０は、ディスプレイリストデータ４２２を画素に変換する。画素は、フレームストア（例えば１６０）に送られる。フレームストア１６０は連続的に、ＣＲＴあるいはＬＣＤのような物理的なディスプレイ４７０に関連してリフレッシュされる。

【００３７】
図６Ａおよび図６Ｂは、システム４００をより詳細に示す図であり、左側にディスプレイリストコンパイラ４２０の機能モジュールが、右側にレンダリングエンジン４３０が示している。角を丸めて示したボックスがメモリ手段を表すのに用いられている。図６Ａの左側において、メモリ手段４５４，４５６，４５８はそれぞれ複数の命令４６０、オブジェクト４６２、フィル４６４を格納する。これらは図４に示したとおりである。

【００３８】
一つの実施形態において、機能モジュールは、パイプライン構成のハードウエアプロセスとして実施され、各モジュールは上流のモジュールからメッセージを受信するためのファーストインファーストアウト（ＦＩＦＯ）バッファを実現する。ハードウエア開発の当業者には、ハードウエアプロセスのパイプライン化により、連続的にそのような処理を通過するデータのスループットを最大にできるということは周知である。また、好ましい実施形態において、ＴＣＩＥ４００は、図３のプロセッサ３０５のような、汎用プロセッサ上で動作するソフトウエアとして実施される。この場合、機能モジュールは、一つあるいは複数のスレッドにより実行されるプログラム機能として実施され、メッセージは、同期関数コールとして、あるいは共有メモリに関連するインタースレッド信号として実施されることになる。以下、ディスプレイデータが通過する順に従って各機能モジュールについて説明する。

【００３９】
図６Ａ、図６Ｂにおいて、命令実行モジュール５００は、ディスプレイオブジェクトデータをフェッチし、解析する役割を担う。モジュール５００は、ホストプロセッサ４５０から直接命令を受信してもよいし、あるいは、ホストプロセッサ４５０の命令によりメモリ４５４から命令をフェッチしてもよい。ＴＣＩＥ４１０の具体的な実施形態で実行される命令のいくつかの例について、以下に説明する。

【００４０】

INST_PLACE_OBJECTは、ＴＣＩＥ４１０に対し、出力デバイス４７０上にオブジェクトを描画することを命令する。INST_PLACE_OBJECTのパラメータは、描画されるべきオブジェクトへの参照と、当該オブジェクトのディスプレイリスト上における所望の位置、スケールおよび向きを特定する変換マトリクスを含む。命令実行モジュール５００が、INST_PLACE_OBJECTコマンドを実行するときは、参照されるオブジェクトのエッジをメモリ４５６からシーケンシャルに読み、エッジデータをそれらの関連する左右のフィルデータへの参照とともに変換モジュール５０２へ送る。命令実行モジュール５００も、（INST_PLACE_OBJECTの）変換マトリクスパラメータをオブジェクトエッジと一緒に変換モジュール５０２へ送る。

【００４１】

INST_WRITE_FILLは、命令実行モジュール５００に対し、グラフィックオブジェクトのためのフィルデータを格納するメモリ５１４内の定められた位置へフィルタデータ４６４を書き込むように指示する命令である。レンダリングエンジン４３０は、エンジン４３０がフレームストア１６０への画素のストリームを生成するときに、そのフィルデータ５３２を用いる。INST_PLACE_OBJECTが出力ディスプレイデバイス４７０にオブジェクトを配置するために実行されるとき、当該オブジェクトのエッジによって参照されるあらゆるフィルデータは、INST_WRITE_FILLへの事前のコールという手段によってあらかじめメモリ５１４へ書き込まれていなければならない。

【００４２】

しばしば、INST_PLACE_OBJECT命令は、出力ディスプレイデバイス４７０上に、オブジェクトをオーバーラップするように配置することがある。すなわち、これは、出力デバイス４７０のピクセルの部分集合が、グラフィカルオブジェクトに対するフィルデータを格納するメモリ５１４内の、複数のフィルデータ５３２によって決定される出力色を有するという状況を表す。このような場合が生じると、出力ディスプレイデバイス４７０上で観察されるときに、いくつかのオブジェクトが他のオブジェクトの前方あるいは後方に現れることが予想される。ＴＣＩＥ４１０は、ｚ－レベルテーブル５１６，５１８を実施して、これを容易にする。ここで、各フィルデータ５０８は、ｚ－レベルテーブル５１６，５１８内のｚ－レベル５１０に関連付けられている。ｚ－レベル５１０の各々には、固定の、ユニークな優先度が与えられ、ｚ－レベルテーブルにおいて最低の優先度から最高の優先度へと並べられる。また、各ｚ－レベル５１０は、当該ｚ－レベル５１０の色を定義するフィルデータを参照する。こうして、テーブル５１６，５１８の低位置のｚ－レベルによって参照されるフィルデータ５０８は、高位置を有するｚ－レベル５１０によって参照されるフィルデータ５０８の後方または下方に現れるように描画されることになる。INST_WRITE_FILL命令は、命令実行モジュールに、フィルデータ５０８にｚ－レベル５１０を関連付けさせる。

【００４３】

INST_SHOW_FRAMEは、命令実行モジュール５００が異なる命令をフェッチすること及び／又は処理することを、出力ディスプレイデバイス４７０が新しいフレームのためのディスプレイデータを要求するまで停止させるのに用いられる命令である。

【００４４】

以下のＴＣＩＥ機能モジュールの説明においては、ディスプレイのスキャンラインに沿ったピクセル－ピクセル間のステップをＸ座標とし、スキャンライン－スキャンライン間のステップをＹ座標とする。

【００４５】

データが渡される次の機能モジュールは変換モジュール５０２である。変換モジュール５０２は、命令実行モジュール５００から受信した変換マトリクスを、やはり命令実行モジュールから受信したエッジの座標に適用する。変換モジュール５０２によって処理された後、エッジはディスプレイ空間におけるスタートのＸ，Ｙ座標と、エンドのＸ，Ｙ座標によって記述され、それらの左右のフィルデータへの参照とともにフィルタモジュール５０４へ渡される。

【００４６】
フィルタモジュール５０４は、変換モジュール５０２によって渡された、表示に全く影響しないすべてのエッジを破棄する。表示に影響しないエッジとは、エッジが水平であるか、或いはすべての座標が表示境界の外側に存在するという理由により表示に影響しないものである。また、いくつかのエッジは、部分的にのみ表示に影響する。表示境界の外側にスタート座標を有し、表示境界の内側にエンド座標を有するエッジは、ある中間の座標（すなわち、当該エッジが画面の境界を横切る位置）にて表示内へ入り込み、表示に出現することになる。そのようなエッジに対しては、フィルタモジュール５０４はそのエッジの新たなスタート座標を計算する。ここで、新たなスタート座標は、エッジが画面に入る上記中間座標に等しくなる。フィルタモジュール５０４は、エッジに縦方向フラグを付与する。そして、エッジのスタートのＹ座標がエッジのエンドのＹ座標よりも常に低くなるようにするために、必要に応じてスタート座標とエンド座標を入れ替える。例えば、スタート座標（５，２２）、エンド座標（８，４）でフィルタモジュールに入ったエッジは、４が２２よりも小さいので、フィルタモジュールによってそのスタート座標とエンド座標が入れ替えられる。縦方向フラグは座標の入れ替えが生じたエッジに対して、それらのエッジが上方向に進むエッジであることを示すためにセットされる。このステップは、レンダリングエンジン４３０がこの方法で表されるエッジに頼るので、必要となる。縦方向フラグは、また、そのエッジによって参照されるフィルデータがそのエッジの正しい側（左と右）に関連するように維持されるためにも重要である。

【００４７】
ディスプレイリストコンパイラ４２０の次のモジュールは、ソートモジュール５０６である。ソートモジュール５０６はフィルタモジュール５０４からエッジを受け取る。受け取られたエッジは、最初にそれらのスタートのＹの表示位置によってソートされ、ついで、それらのスタートのＸの表示位置によってソートされる。ソートされたエッジ５１２は、内部メモリ手段４４０に配置され、そこからレンダリングエンジン４３０によって読み出され、処理される。エッジは内部メモリ手段４４０内に設けられたフレームエッジバッファ５２４，５２６の部分に書き込まれる。出力データの現在のフレームを記述するのに用いられるすべてのエッジは、レンダリングエンジン４３０が必要とする前に、フレームエッジバッファ５２４，５２６に存在していなければならない。フレームエッジバッファ５２４，５２６は、好ましくはダブルバッファにより実施される。ディスプレイリストコンパイラ４０２がエッジを処理し、第１のフレームエッジバッファ５２４に並べる間に、レンダリングエンジン４３０は、ディスプレイリストコンパイラ４２０によって事前に準備された第２のフレームエッジバッファ５２６からディスプレイ出力を生成することができる。次いで、現在のフレームに対するディスプレイデータの出力を完了すると、フレームエッジバッファ５２４，５２６は入れ替わり、レンダリングエンジン４３０は、次のフレームのためにディスプレイリストコンパイラ４２０によって提供されたエッジの処理を開始できるようになる。

【００４８】
レンダリングエンジン４３０は、フレームのためのエッジがフレームエッジバッファ５２４，５２６にスキャン順で書き込まれていることを要求する。スキャン順とは、ディスプレイデバイスがそのディスプレイデータを受け取りリフレッシュする順番である。ここでは、本説明のために、スキャン順とはディスプレイの左上隅の位置、すなわち左上隅の画素から始まるものとする。本例のスキャン順においては、そこからディスプレイ画素の最上行を左から右へ進み、ディスプレイの右上隅の画素に到達する。そして、最上行の次の行の左端の画素へと続き、再び左から右へと進む。スキャン順は、このような形態で、最終の画素に到達するまで継続する。ここで、最終画素は、ディスプレイの最下行の右端となる。このようなスキャン順はしばしばラスタースキャン順と呼ばれる。

【００４９】
ソートモジュール５０６は好ましくはバケットラディックス－ソーティングアルゴリズム（bucket radix-sorting algorithm）を用いて、フレームに対するすべてのエッジを、

それらのスタート座標がスキャン順となるように、内部メモリ手段４４０に並べる。ソフトウエア又はハードウエア開発の当業者は、ラディックス－ソーティングアルゴリズムがエレメントをN回（ここでNはソートされるべきエレメントの数）で順番に並べることに気づくであろう。ラディックスソーティングアルゴリズムは、有効なメモリ手段に依存して、すべてのエレメントを通して１回以上の反復を要求する。ソートの最初の反復を、ソートモジュールがフレームのためのエッジを受信している間に実行することができる。一つの実施形態において、エッジは、ＤＲＡＭを用いて実現される内部メモリ手段４４０内でソートされる。

【００５０】

レンダリングエンジン４３０を通じたディスプレイデータの流れは、エッジ処理モジュール５４８から始まる。特定のフレームのために必要なディスプレイ出力を集合的に記述するエッジ５４０の一次資源は、ディスプレイリストコンパイラ４２０によってフレームエッジバッファメモリ５２４，５２６に準備された、ソート済みエッジのリストである。

【００５１】

これらのリスト中の各エッジ５４０は、以下のデータフィールドを含む。

●スタートのＸ座標；
●スタートのＹ座標；
●エンドのＹ座標；
●前のＸ及びＹ座標（又はスキャンライン）から、新しいＹ座標（又はスキャンライン）に対応するエッジの新しいＸ座標を決定するために用いられる一つ以上のパラメータ。これは例えば、１つのスキャンラインを横切る直線エッジのＸ座標に加算された場合に次のスキャンライン（下のライン）に対する当該エッジのＸ座標を生成するデルタＸ値である。曲線エッジの場合は、そのようなパラメータが複数用いられる；
●フィルタモジュール５０４によって用意された縦方向フラグ；
●リスト中の次のエッジのアドレス；
●左右フィルのｚ－レベルへの参照。

【００５２】

エッジ処理モジュール５４８は、２つのエッジのソースを有する。１つは上述のようなメモリ５２４，５２６である。もう１つのエッジソースは、エッジ処理モジュール５４８によって維持される、アクティブエッジバッファ（２の１）を含むメモリ５２２，５２３である。これらの使用とエッジ処理モジュール５４８の全体的な動作について図７Ａ、図７Ｂを参照して以下に説明する。図７Ａ，図７Ｂは、エッジ処理モジュール５４８を実現するソフトウエアによって実行される処理ステップを表す。

【００５３】

描画されるべき各フレームについて、エッジ処理モジュール５４８は、ディスプレイ４７０の下へ向かって、スキャンラインからスキャンラインへ（行から行へ）の繰り返しにより動作する。モジュール５４８は、フレームエッジバッファ５２４，５２６あるいはスタティックエッジバッファ５２８，５３０内のいずれかのエッジが現在のスキャンラインと交差する位置を計算する。各交差のＸ位置が、その交差するエッジの左右のフィル参照とともにｚ－レベルアクションモジュール５５０に渡される。

【００５４】

図７Ａ、図７Ｂは、一つのスキャンラインに対するエッジ処理モジュール５４８の動作を示す。その処理は、エントリポイント７００からスタートする。ステップ７０２では、スタティックエッジバッファ５２８，５３０が空であるかどうかを調べる。空であれば、ステップ７０４において、フレームエッジバッファ５２４が空かどうかを調べる。空であれば、ステップ７１６において、アクティブエッジバッファ５２２，５２３が空かどうかを調べる。これらのソースのいずれにもエッジが存在しない場合は、現在のスキャンラインと交差するエッジが存在し得ないことが明らかである。したがって、当該スキャンラインに対するエッジ処理を終了し、本方法をステップ７２４で完了する。フレームエッジバッファ５２４，５２６は、スキャン順でリストされるようにエッジを格納しているので、

その中のいずれかのエッジが現在のスキャンラインと交差するならば、それらはこれらのリストの次の有効なエッジとなる。スタティックバッファ５２８，５３０において、エッジが存在する場合は、ステップ７０６にて、フレームエッジバッファ５２４，５２５についてエッジの存在が調べられる。ステップ７０６でエッジが存在しない場合、ステップ７１２で、スタティックエッジバッファ５２８，５３０において次のエッジとなるべき次のエッジがセットされる。ステップ７０６でフレームエッジバッファ５２４，５２６にエッジが存在する場合は、ステップ７０８に進み、フレームエッジバッファ５２４，５２６内の次のエッジのスタート座標と、スタティックエッジバッファ５２８，５３０における次のエッジが比較される。その座標のほうが大きい場合、ステップ７０８から上述のステップ７１２に進む。そうでない場合は、次のエッジがステップ７１０で、フレームエッジバッファ５２４，５２６からのものにセットされる。ステップ７１０及び７１２の各々から続くステップ７１４では、スタートＹ座標が現在のスキャンラインに対応するＹ座標よりも大きいかどうかを判断する。大きい場合、当該エッジは現在のスキャンラインとは交差せず、後の（すなわち下方の）スキャンラインでの処理のためにバッファに残され、処理はステップ７１６へ進む。他の場合、エッジは現在のスキャンラインと交差し、処理は、ステップ７１８へ進む。エッジがほぼ平行でない限り、そのエッジのエンドＹ座標よりも大きいＹ座標に対応する最初のラインに至るまで、そのエッジは後続の（下の）スキャンラインと交差する。当該エッジと次のスキャンラインが交差するかどうかの判断処理を促進するために、エッジ処理モジュール５４８は、エッジが以降“アクティブエッジ”と記述されるものとなるようそのエッジのフォーマットを変換する。アクティブエッジは、スタートＸ、Ｙ座標とエンドＹ座標を表すデータを有するのではなく、少なくとも現在Ｘ座標とエンドＹ座標を表すデータを有する。アクティブエッジの現在Ｙ座標は、暗に、現在のスキャンラインのＹ座標でもある。アクティブエッジは、エッジ処理モジュール５４８が現在のスキャンライン上の現在Ｘ座標から次のスキャンラインに対する当該エッジのＸ座標を算出可能にするデータ（例えば、デルタＸ）を含んでもよい。

【００５５】

エッジ処理モジュール５４８が後続のスキャンラインに向かって下方へ続くエッジからアクティブエッジを生成するとき、このアクティブエッジは、次のスキャンラインに対する処理において用いられるアクティブエッジのリストへ追加される。アクティブエッジバッファ５２２，５２３は、このリストを格納するために用いられる。アクティブエッジバッファはダブルバッファであり、次のスキャンラインのために生成されるアクティブエッジのリストを含む第１バッファ５２２と、前のスキャンラインの処理中に現在のスキャンラインのために既に生成されたアクティブエッジのリストを含む第２バッファ５２３とを備える。フレームエッジバッファ５２４，５２６内のエッジと同様に、アクティブエッジのリストはスキャン順に並ぶ。

【００５６】

エッジ処理モジュール５４８が、エッジのソースにかかわらず、スキャン順で交差を処理することは重要である。このため、現在のスキャンラインを横切るフレームエッジバッファ５２４，５２６からのエッジは、より小さいＸ座標で交差するアクティブエッジがアクティブエッジバッファ５２２，５２３の中に存在しなくなるまでは処理されない。ステップ７１８と７２０はこのテストを実行する。そして、そのときだけ、ステップ７２８において、フレームエッジバッファ５２４，５２６からのエッジが新たなアクティブエッジに変換される。スタティックエッジバッファ５２８，５３０及びフレームエッジバッファ５２４，５２６の各々が空であるという場合においては、次のアクティブエッジをバッファ５２２，５２３から直接に派生してもよい。

【００５７】

大部分のスキャンラインに対して、当該スキャンラインを横切るエッジのみが、前のスキャンラインから画面下方へ継続したエッジとなる。その意味において、すべての交差するエッジが、前のスキャンラインにより生成されたアクティブエッジバッファ５２２，５２３に端を発する場合もある。このような状況においては、ステップ７１４の結果がＮＯ

となるか、あるいは７０２と７０４の両方の結果がＹＥＳとなるかのいずれかとなる。
【００５８】
ステップ７２２あるいは７２８から次の交差が決定されると、対応するアクティブエッジからのデータの部分集合がステップ７３０において、レンダリングエンジン４３０の次のモジュールに渡される。そして、そのアクティブエッジは、ステップ７３２で、エンドＹ座標をチェックすることにより、次のスキャンラインに続くかどうかを見るためにテストされる。アクティブエッジが続く場合、ステップ７３４において、アクティブエッジの現在Ｘ座標が、後続のスキャンラインのために再計算される。そして、アクティブエッジは、次のスキャンラインのためのアクティブエッジバッファに置かれる。本エッジ処理は、次のスキャンラインのために、ステップ７３２及び７３４の各々から、ステップ７００のスタートへと戻る。
【００５９】
このスキャンラインからスキャンラインへエッジのＸ座標を追跡する処理は、しばしば、“エッジ追跡（edge tracking）”と呼ばれる。好ましい実施において、エッジは直線として記述される。直線のエッジを追跡するためには、シンプルな、エッジ毎のデルタＸの調整が各スキャンラインに適用される。
【００６０】
アクティブエッジはスキャン順で処理されるが、ステップ７３４においてなされる新しい現在Ｘの計算の結果、このアクティブエッジが、このスキャンライン上で既に処理されたアクティブエッジよりも小さい値の位置を有するようになる可能性がある。この状況の例を、図８に示す。図８に示されるように、２つの破線で示された水平線は、出力ディスプレイのスキャンラインを表し、上の破線は、エッジ処理モジュール５４８によって現在処理中のスキャンライン（Current Scan Line）を表し、下の破線は次に処理されるスキャンライン（Next Scan Line）を表す。図８では、３つのアクティブエッジ、すなわち、アクティブエッジＡ（Active Edge A）、アクティブエッジＢ（Active Edge B）、アクティブエッジＣ（Active Edge C）が示されており、それぞれ交差点Ａ（Intersect A）、交差点Ｂ（Intersect B）、交差点Ｃ（Intersect C）で現在のスキャンラインと交差する。アクティブエッジの現在Ｘフィールドは交差の位置を示す。すべてのエッジのソースはスキャン順となっているので、エッジ処理モジュールもｚ－レベルアクティベーションモジュール５５０への出力をスキャン順に生成する。このことは望ましいことである。しかし、エッジ処理モジュール５４８が、次のスキャンライン上の交差（交差Ａ’（Intersect A'）、交差Ｂ’（Intersect B'）、交差Ｃ’（Intersect C'））に対応するアクティブエッジに関して新たな現在Ｘ値を計算すると、アクティブエッジＣはアクティブエッジＡ及びアクティブエッジＢと交差しているため、これらのエッジのスキャン順は失われてしまう。したがって修正されたアクティブエッジが、それらが次のスキャンラインのためのアクティブエッジバッファに配置される前に再格納することを要求する。図８の例において、次のスキャンラインのためのアクティブエッジバッファ５２２，５２３におけるアクティブエッジの望ましい順序は、最初にアクティブエッジＣ、次にアクティブエッジＡ、そして最後にアクティブエッジＢである。これを克服するために、エッジ処理モジュール５４８は、修正されたアクティブエッジをソートバッファ（不図示であるが、バッファ５２２から５３０に対応した方法で実施される）に、それらがスキャン順となるように挿入する。ソートバッファ内のアクティブエッジのリストは、次のスキャンラインのためのアクティブエッジバッファ５２２，５２３へ送られる。ステップ７３０において、エッジ処理モジュール５４８によってｚ－レベルアクティベーションモジュール５５０に渡されるＸメッセージを形成するアクティブエッジデータの部分集合は以下のものを含む。
●アクティブエッジのＸ座標（それが現在のスキャンラインと交差する位置）
●アクティブエッジと関連するフィルｚ－レベルへの参照
●アクティブエッジの縦方向インディケータ。
【００６１】
ｚ－レベルアクティベーションモジュール５５０は、フィルバッファ５１４内のどのフ

ィルデータ５３２が出力ディスプレイ画素の色に寄与するかを判断するためのｚ－レベルアクティベーションテーブル５６０を維持するために、エッジ処理モジュール５４８から渡されたエッジ交差のデータを用いる。出力ディスプレイ画素のストリームは、レンダリングエンジン４３０によってスキャン順で生成される。スキャンラインとエッジとの各交差は、スキャン順で生成されるときに、要求される出力色が変化することになる表示座標を表すことになる。以下の説明において、“領域”とは、１つの交差から次の交差の間のスキャンラインの座標の間隔に対応する。あらゆる領域の画素データは、ｚ－レベルアクティベーションテーブル５６０のｚ－レベルによって参照される１つ又は複数のフィルデータ（フィルスタイル）によって決定される。ｚ－レベルアクティベーションテーブル５６０の各ｚ－レベルのデータは、カウントフィールドに対応するフィルデータ５３２への参照を含む。カウントフィールドは符号付である。カウントフィールドの使用について図９Ａ，図９Ｂを参照して、以下、説明する。

【００６２】

ｚ－レベルアクティベーションモジュール５５０は、ステップ９００で、エッジ処理モジュール５４８からメッセージを受信すると、ステップ９０２において示されるように、そのメッセージによって参照されるｚ－レベルのカウントフィールドがメッセージの縦方向インディケータに依存して加減算される。ＴＣＩＥ４１０は、“（非ゼロ）ワインディングカウンティングフィルルール（(none-zero) winding counting fill rule）”を用いて、フィルデータのどのｚ－レベルが出力画素に寄与するかを判断する。或いは、“奇数／偶数”又は“ネガティブ”のような他のフィルルールを用いることもできる。ここでは、ｚ－レベルは、当該ｚ－レベルのフィルデータがレンダリングエンジン４３０によって現在生成されている出力画素に寄与すべく要求される場合に、アクティブであるとして説明されている。各スキャンラインに対する処理のはじめに、すべてのｚ－レベルのためのカウントフィールドは０にセットされる。ｚ－レベルは、ｚ－レベルアクティベーションテーブル５６０内の対応するカウントが加算或いは減算されて、正あるいは負の値になったときにアクティブとなり、ゼロに戻るまでアクティブ状態を維持する。後続の画素の領域にどのフィルｚ－レベルが寄与するかを判断するのに重要なのは、アクティブ／インアクティブになるｚ－レベルに対するディスプレイ座標のみである。したがって、メッセージデータをレンダリングエンジン５３０の次のモジュール（すなわち、ランカリングモジュール５２２）に渡す必要があるのは、ｚ－レベルのカウントフィールドがゼロと非ゼロの間を変化するときのみである。

【００６３】

受信されたメッセージの縦方向インディケータが、ステップ９０２で“下方向”であると判断される場合、当該メッセージのデータによって参照されるｚ－レベルのカウントフィールドは、ステップ９０４でインクリメントされる。また、ステップ９０２でメッセージが“上方向”であると判断された場合、当該メッセージのデータにより参照されるｚ－レベルはステップ９０６でデクリメントされる。その後、図９Ａ，９Ｂからわかるように、フローチャートは２つの実質的なミラーイメージパス（mirror-image paths）に分けられ、ステップ９２８で合流する。

【００６４】

具体的には、ステップ９０４に続くステップ９０８は、左のｚ－レベルの０から１への変化をテストする。左のｚ－レベルが０から１へ変化しているのであれば、ステップ９１０において、当該ｚ－レベルに対するＯＮメッセージをＸメッセージの一部として出力に付加する。ステップ９０８で否と判定された場合、或いは、ステップ９１０の処理を終えた後は、ステップ９１６において右ｚ－レベルメッセージに対するテーブル中のカウントフィールドをデクリメントする。そして、ステップ９２０において、右ｚ－レベルの１から０への変化をテストする。右ｚ－レベルが１から０へ変化したのであれば、ステップ９２４において、当該ｚ－レベルに対するＯＦＦメッセージを出力に付加する。ステップ９２０で否と判定された場合、或いは、ステップＳ９２４の処理を終えた後は、ステップ９２８において、入力メッセージのディスプレイ座標とともに、ｚ－レベルＯＮ／ＯＦＦメ

ッセージをランカリングモジュール５５２に出力する。このデータは、表示のためのピクセルランを記述したものとなる。

【００６５】

相補的な方法で、ステップ９１２が、左ｚ－レベルの非ゼロ（例えば１）からゼロへの変化をテストする。ゼロへ変化したならば、ステップ９１４に進み、当該ｚ－レベルに対するＯＦＦメッセージをメッセージの一部として出力に付加する。ステップ９１２において否と判定された場合、或いは、ステップ９１４の処理の後は、ステップ９１８において、右ｚ－レベルメッセージに対するテーブル内のカウントフィールドをインクリメントする。そして、ステップ９２２において、右ｚ－レベルの０から１への変化をテストする。０から１へ変化したと判定された場合、ステップ９２６において、当該ｚ－レベルに対するＯＮメッセージを出力に付加する。ステップ９２２において否と判定された場合、或いは、ステップ９２６の処理の後は、ステップ９２８において、入力メッセージのディスプレイ座標とともにｚ－レベルのＯＮ／ＯＦＦメッセージを出力する。

【００６６】

ｚ－レベルアクティベーションは、ステップ９３０で終了する。

【００６７】

ステップ９１０と９１４は、フィルｚ－レベルがオンになった（アクティベートされた）或いはオフになった（デアクティベートされた）ときを表すメッセージを、ランカリングモジュール５５２へのメッセージの一部として生成する。

【００６８】

ｚ－レベルアクティベーションテーブル５６０の下位インデックスのエントリは、高位インデックスのエントリの“下に”描画されることになるフィルデータを参照する。例えば、インデックス１（ｚ－レベル１）のｚ－レベルがべたの赤色を参照し、インデックス２（ｚ－レベル２）のｚ－レベルがべたの緑色を参照し、そしてこれらが画素領域に対する唯２つのアクティブなｚ－レベルである場合、ｚ－レベル１は完全にｚ－レベル２によって隠され、したがって当該領域は緑色のべたで描画される。この例において、ｚ－レベル２が部分的に透明色のフィルを参照する場合は、その領域は、ｚ－レベル２のフィルを通してｚ－レベル１の赤べたが部分的に現れるように描画される。ｚ－レベルアクティベーションテーブル５６０は、対応するｚ－レベルが下位のインデックスのｚ－レベルを完全に隠すかどうか（すなわち、対応するｚ－レベルが完全に不透明なスタイルのフィルであるかどうか）を示す追加フィールドをエントリ毎に含むようにしてもよい。

【００６９】

１つの実施の形態において、ｚ－レベルアクティベーションモジュール５５０は、出力データを生成するのに必要な時間を減らすために付加的な機能を実行する。ｚ－レベルアクティベーションモジュール５５０は、フィル生成モジュール５５４で使用されることになるあらゆるｚ－レベルがアクティブ／インアクティブになったときを示すメッセージを出力する代わりに、ｚ－レベルの部分集合がアクティブ／インアクティブになったときだけフィル生成モジュール５５４が通知を受けるようにメッセージを出力する。この部分集合は、最高位（最上位）のインデックスを有し、現在アクティブなｚ－レベルの集合である。具体的な実施形態において、レンダリングエンジン４３０は、最高位の、例えば４つのｚ－レベルのみが常に画素の領域の色に寄与することを許容されるように構成される。このような妥協は、出力に誤差を招くことになるが、フィルカラーを生成するのに用いられるすべての（４つの）上位のアクティブなｚ－レベルは、この誤差が生じるか或いは視認可能になる前に重要な透明度を持たねばならない。このような減縮の利点は、フィルカラーの生成が、非常に大量の処理を要求する、潜在的にｚ－レベルアクティベーションテーブル５６０におけるすべてのｚ－レベルからのフィルデータの合成をするのではなく、フィルデータの合成が最大で４つのｚ－レベルによりなされることを保証することである。なお、この説明では、出力色の合成のためのｚ－レベルの最大数が４である実施形態に関して説明したが、ｚ－レベルの最大数は任意にして実施できる。

【００７０】

ランカリングモジュール５５２の動作について、図１１のフローチャートを参照して以下に説明する。図１１はランカリングモジュールを実現するソフトウエアを表している。この説明において、“ラン”とは、類似のフィルを有するスキャンラインの座標のスパン（間隔）を言う。“フィル”は、緑のようなべた色、カラーランプあるいはラジアルブレンドのようなより複雑な色関数、ビットマップ（おそらくはサンプルされた）、或いはスパン内の各画素の色を決定するデータセットのようなあらゆるものによって定義され得る。ランカリングモジュール５５２は、前回の描画されたフレームからの、その前回のフレームを生成するのに用いられたランの集合のリンクされたリストを維持する。この説明のために、そのリストは空ではなく、いくつかのエントリが前回のフレームから生成されている場合を考慮するのがよい。これを説明する過程において、現在のフレームのためのランを後続のフレームで利用するためにいかにして維持するかを見ることもできるであろう。

【００７１】

ランカリングモジュール５５２は、図６Ｂの内部メモリ手段４４０内に維持されるランレコードのプール５２０を用いる。このプール５２０は、一般には、トータルのメモリサイズがフレームストアのトータルのメモリサイズの一部分となるように、選択された固定サイズを有する。例えば、フレームストアサイズの１／４又は１／８が一般的に用いられる。ランカリングモジュール５５２は、フレームストアのサイズに匹敵するデータ量を格納することが、画素の直接の生成に匹敵する計算量を要するということに基づいて、プール５２０に適合した量の記録を行うように徐々に抑制される。全体として、ランカリングモジュール５５２の目的は、作業の回避による時間の節約であり、より多量のメモリを使用することになるあらゆる試みは、関連する特定のデータに対してそのテクニックは効果を示さず、避けるべきである。プール５２０の各レコードは以下を維持する。

●次のランレコードへのリンク
●スタートｙ座標
●スタートｘ座標
●長さ
●フィルテーブルインデックス。

【００７２】

現在の維持されたステートの一部ではないランレコードはフリーリストにリンクされる。フレームが描画されるとき、生成中のフレームのステートを記録するランレコードが“新たな維持されたランリスト”に格納される。このリストは、次のフレームの描画へ進んだときに、“古い維持されたランリスト”になる。各リストは単一のリストヘッドポインタで記録される。これは図１１において示される初期化ステップ１１００内で確立される。

【００７３】

ステップ１１０２において、ｚ－レベルアクティベーションモジュール５５０からランが受け取られると、このランを保持されたランプール５２０に記録するかどうかが判断される。この処理では、更なるランが存在するかどうかを判断するステップ１１０４を最初に伴う。存在しないのであれば、ステップ１１２８において古いリストをフリーリストにダンプし、ステップ１１３０で古いリストを新しいリストに割り当てる。そして、ステップ１１３２で新しいフレームの開始を待ち、それが開始されると制御はステップ１１０２へ戻る。

【００７４】

ステップ１１０４でランが存在する場合、ステップ１１０６において、そのランを保持するかどうか決定する。この決定は、以下のようにメモリ容量に基づく。すなわち、描画されるべきスキャンラインの残数によって除されたフリーランレコードの数が判断される。これは、各残存スキャンラインに対して扱われることが必要となるランレコード（設計の特徴で決まる）の数の平均である。これが、現在のスキャンラインに対してそれまでに記録されたランの数と比較される。リミットに達していない場合、フリーリストからのフ

リーランレコードが取得され得る。そして、対応するランの詳細のセット、及びそのレコードは、新たに保持されたランリストの先頭へプッシュされる。これは、ステップ１１０８に対応する。このプロセスにより、ランは、次のフレームの生成中における使用のための現在のフレームに寄与するレコードとなる。

【００７５】

いずれにしても（現在のランが記録されてもされなくても）、受信されたランは、ステップ１１１０において、保持された古いリストの先頭にあるランと比較される。保持された古いリストのスタート前に発生するランの、あらゆる先端部分（或いはすべて）は、ステップ１１１２においてフィル生成モジュール５５４へ渡される。ステップ１１１４であらゆる残り部分は、残り部分と保持された古いランとの間のあらゆる先頭のオーバーラップと比較される。フィルインデックスが異なる場合、ステップ１１１６において、残りの先頭部分がフィル生成モジュール５５４に送られる。いずれにしても、保持されている古いランは、ステップ１１１８で先頭のオーバーラップにより短縮される。そして、ステップ１１２０において、その長さがゼロに減少したかテストされ、そうであれば、ステップ１１２２でレコード全体がフリーリストへ送られる。残りもステップ１１２４にて、同様の方法で短縮される。まだ残りがある場合は、その残り部分とともに、ステップ１１１０へ戻り、受信したランとして扱われる。そうでなければ、次の受信されたランを扱うために、制御はステップ１１０２へ戻る。

【００７６】

なお、保持されたレコードが有効であり、フィルインデックスが直前のフレームのものと同じである、受信されたランの部分に対して、フィル生成モジュール５５４へはフィルリクエストは送られない。こうして画素生成に関連する多くの作業を回避できる。また、処理パイプラインのこのステージは、特定の色ではなくフィルインデックスを参照するので、このアプローチは、べた色ばかりでなく、カラーランプやビットマップを含むすべてのタイプのフィルに対するフィルリクエストを渡すことを回避する。

【００７７】

多重に重ねられた（或いは合成された）透明オブジェクトがサポートされる場合、上記のフィルインデックスは、サポートされた同時オーバーレイの最大数までのフィルインデックスのショートアレイに置き換えられる。本実施形態ではこの最大数は４である。

【００７８】

次に、図１２を参照して、ＴＣＩＥ４１０のフィル生成モジュール５５４の動作について説明する。以下の説明は、画素の領域の色に寄与するのに用いられるｚ－レベルの数を常時最大４つに制限する実施形態に関する。そのような実施形態において、ＴＣＩＥ４４０のフィル生成モジュール５５４は、データを生成可能なフィル生成の４つの手段１２１０（１２１０Ａから１２１０Ｄ）を有する。これらのフィル生成手段１２１０は、上述したランカリングモジュール５５２からのメッセージによって制御される。フィル生成手段１２１０の出力は、描画されている現在の表示座標に対する関連するｚ－レベルに対して要求された画素色を記述するデータである。各フィル生成手段１２１０の画素カラーデータは、領域コンポジットモジュール５５６に渡される。これは、アクティブな各ｚ－レベルに対して生成された色をブレンドし、表示のために要求された出力画素データを生成する。

【００７９】

ランカリングモジュール５５２からのメッセージは以下のデータを含む。

●Ｘディスプレイ座標

●その座標でアクティブになるｚ－レベルのインデックス。

【００８０】

図１２に示されるように、フィル生成モジュール５５４は、フィルデータテーブル５１４と、ｚ－レベルテーブル５１６に結合するフィルルックアップ及びコントロールモジュール１２０４を含む。モジュール５５４は、ランカリングモジュール５５２より入力１２０２で注目されたメッセージを受け取る。その値はモジュール１２０４に維持される。

【００８１】
フィル生成モジュール５５４は、４つのフィル生成手段１２１０の各々をｚ－レベルアクティベーションテーブル５６０内のｚ－レベルにインデックスする、ハードウエアレジスタ或いはソフトエア変数のような、メモリ手段１２０８を保持する。ｚ－レベルをデアクティベートするメッセージが受信されると、そのｚ－レベルに関連するフィル生成手段１２１０は、対応するインデックスの手段によりそのｚ－レベルと無関係にされる。ｚ－レベルに関連しないフィル生成手段１２１０は、画素データを生成しない。ｚ－レベルがアクティブになったことを示すメッセージが受信されると、ｚ－レベルと関連していないフィル生成手段１２１０の一つが、アクティブになったそのｚ－レベルと関連するようになる。

【００８２】
各フィル生成手段１２１０によって使用されるフィルデータには２つのソースがある。第１のソースはｚ－レベルテーブル５１６，５１８であり、第２のソースはフィルデータメモリ５１４である。ｚ－レベルテーブル５１６，５１８はダブルバッファであり、バッファ５１６，５１８はレンダリングエンジンがフレームのレンダリングを完了したとき入れ替わる。第１のフィルテーブルを有する第１バッファ５１６がディスプレイリストコンパイラによって準備されている間に、前のフレームのレンダリングの間にディスプレイリストコンパイラによって準備された第２バッファが各フィル生成手段１２１０によって読まれる。ｚ－レベルテーブル５１６，５１８は、ｚ－レベルのインデックスと、当該ｚ－レベルに対する画素データを生成ために必要な対応データ（フィルテーブルに格納された）との間の間接的な関連（indirection）を提供する。ｚ－レベルテーブルはｚ－レベル毎に１つのエントリを含み、ｚ－レベルテーブル内のエントリは、同じインデックスを持つ、ｚ－レベルアクティベーションテーブル５６０内に対応するエントリを有する。各ｚ－レベルテーブルのエントリは、フィルデータメモリ５１４のフィルデータ５３２編参照を含む。各ｚ－レベルテーブルエントリは追加的に以下のものを含んでもよい。
●そのｚ－レベルが不透明かどうかを表すフラグ（NEED_BELOW）
●フィルデータがＸ位置の変化に依存する（例えば、ビットマップ或いはブレンド）か、フィルデータが一定に保たれる（例えば単色のフィル）かを示すフラグ（X_INDEPENDENT）。

【００８３】
上述の追加フラグは、フィル生成モジュール５５４が要求されるフィルデータの処理を最小にすることを可能とする。

【００８４】
単純な単色のフィルを参照するｚ－レベルに対して、フィルテーブル５１４のフィルデータ５３２は、たとえば、赤、緑、青及びアルファ（透明度）成分を備えた、単純な色記述となろう。階調フィルのためのフィルデータは、色のテーブルと現在の出力ディスプレイ座標からこのテーブル内へインデックスを生成するのに用いられる付加的なパラメータとして実現される。

【００８５】
ＴＣＩＥ４１０内の階調フィルのために格納されたデータ、及び、フィル生成モジュール５５４内のフィル生成手段１２１０が、出力データを生成するのにどのように動作するかについて以下に説明する。階調のためのフィルデータは、１７色のテーブルとして実施される。０から２５５の間の値が１７色のこのテーブルをインデックスするのに用いられる。テーブル内の各連続的な色エントリが、直前のものよりも１６だけ大きいインデックス値と関連付けられる。従って、最初のエントリは、インデックス０を有し、２番目のエントリはインデックス１６を有し、最後のエントリはインデックス２５６を有する。１６の倍数とならないインデックスに対応する色は、テーブルにおいて要求されたインデックスに近いインデックスを持つ２つの隣接した色からリニアに補間される。階調に対するフィルデータは、また、カラーテーブルへのインデックスが、特定のディスプレイ座標に対してどのように取得され得るかを示すパラメータを要求する。

【００８６】
ＴＣＩＥ４１０によってターゲットディスプレイ４７０にオブジェクトが配置されると、オブジェクトデータは、エッジがディスプレイ座標と対応するように変換される。従って、オブジェクトに関して階調の現れ方（例えば、位置や姿勢）が矛盾しないように、そのオブジェクトのエッジに含まれるあらゆる階調フィルも変換される必要がある。
【００８７】
各ディスプレイ座標に対してカラーインデックステーブルを決定するために必要な計算を最小にするために、ＴＣＩＥ４１０は、フィルデータ５３２に対して、ディスプレイ座標においてバウンディングボックスの概念を実施する。バウンディングボックスは、ディスプレイ座標のＸ軸に平行な２つのエッジと、ディスプレイ座標のＹ軸に平行な２つのエッジで囲まれる、ディスプレイの矩形領域を記述する。ＴＣＩＥ４１０の一つの実施形態において、バウンディングボックスは、階調フィルのためのフィルテーブル５１４におけるフィルデータ５３２の一部を形成する。
【００８８】
線形の階調フィルに対して、フィルテーブル５１４内のデータは、付加的にカラーテーブルへのスタートインデックスを含む。スタートインデックスは、バウンディングボックスの左上隅に最も近いディスプレイ画素に対する出力色を表す。フィルデータ５３２もまた、デルタＸ値とデルタＹ値を含む。デルタＸ値は、右側の次の画素（Ｘ座標の繰り返し）に対するカラーテーブルへの新しいインデックスを取得するために、スタートインデックスをインクリメントするのに用いられる。インデックスのこの前方へのインクリメントは、スキャンラインに沿って左から右へディスプレイ画素データが生成される間継続する。こうして、フィルテーブルからのフィルカラーの線形の階調がバウンディングボックスの右側の辺に至るまで生成される。デルタＹ値は、次のスキャンライン上のバウンディングボックスの左辺に対するスタートインデックスを取得するために、カラーテーブルへのスタートインデックスをインクリメントするのに用いられる。スタートインデックス、デルタＸ、デルタＹ及びバウンディングボックスは、まとめて、カラーテーブルから種々の線形の階調フィルを生成する手段を提供する。スタートインデックスの値、デルタＸ、デルタＹは、通常は整数部と小数部で実現される（例えば固定小数点）。フィルテーブル５１４内のフィルデータ５３２は、インストラクション（例えば上述のINST_WRITE_FILL）によって変形され得る。これは、階調フィルの姿勢、位置及びスケールが、オブジェクトの姿勢、位置及びスケールに対して矛盾しないように、必要に応じて階調フィルのパラメータを変換可能とする。
【００８９】
１つの実施形態において、階調フィルのためのデータは、バウンディングボックスを含まない。その代わりに、当該階調フィルを含むオブジェクトのバウンディングボックスが、各エッジが配置されたＸ及びＹの最大最小値を記録することにより、動的に計算される。スタートインデックス、デルタＸ、デルタＹの値が、このバウンディングボックスに関連して提供される。これらの最大／最小値の記録は、ＴＣＩＥ４１０が、ディスプレイ座標のＸ軸に平行な２つのエッジと、ディスプレイ座標のＹ軸に平行な２つのエッジで矩形を記述し、オブジェクトの全てのエッジを含むバウンディングボックスを維持できることを確実にする。この実施形態の利点は、バウンディングボックスデータがフィルデータテーブル５１４を消費せず、フィルテーブル５１４においてバウンディングボックスを更新するための命令（それもメモリ手段、或いはホストプロセッサの労力を消費する）を要求するのではなく、わずかな付加的な処理で、バウンディングボックスがＴＣＩＥ４１０により再計算されることにある。
【００９０】
フィル生成モジュール５５４はまた、ビットマップイメージに基づいてフィルを実施する。階調フィルに関して説明したのと類似のテクニックが用いられる。ビットマップファイルは、フィルを含むオブジェクトの一部として定義されているか、或いは、フィルテーブル内のフィルを記述するデータの一部として定義されているバウンディングボックスの

いずれかに依存する。バウンディングボックス内のピクセルの値は、ビットマップイメージにおいて定義された画素に対する値から決定される。このビットマップイメージは、ビットマップファイルに対するフィルデータ５３２によって参照される。階調フィルについては、ビットマップフィルは、含まれるオブジェクトの姿勢、位置及びスケーリングと整合する姿勢、位置及びスケーリングで描かれなければならない。これを可能にするため、フィルテーブル内のビットマップフィルデータは、ディスプレイデータがどのようにビットマップファイルのソースビットマップからリトリーブされるかを制御するために重ね書き（メモリ手段３０６，３０９からフェッチされた命令を介して、或いはホストプロセッサ４５０を介して）されても良い。ビットマップフィルを生成するフィル生成手段１２１０の動作は、バウンディングボックス内の画素に関して増加的にデータが計算されるという点において、階調フィルを生成する動作に似ている。階調フィルが増加的にカラーテーブルインデックスを計算するのに対して、ビットマップファイルは、ソースビットマップ内のピクセルのメモリアドレスを増加的に計算する。フィルテーブル５１４内のビットマップフィルのためのパラメータは以下を含む。

●ビットマップスタートＸとビットマップスタートＹ座標
●デルタＸとデルタＹ
●デルタスキャンラインＸとデルタスキャンラインＹ
●最大Ｘと最大Ｙ。

【００９１】

パラメータはまた、以下を含んでもよい。

●ビットマップベースアドレス
●ソースビットマップで使用される画素毎のバイト数の表示。

【００９２】

ビットマップＸ、Ｙ座標は、ソースビットマップ内の位置に対応する。座標は、サブピクセルの精度を有する。すなわち、それらは、ソースビットマップ内の画素を参照する整数部とその画素内の位置に関連する小数部を有する。ビットマップフィルを生成するフィル生成手段１２１０は、レンダリングエンジン４３０がバウンディングボックスの左上に最も近い画素に対するデータを生成しなければならなくなる前に、スタートＸとスタートＹをローカルメモリ手段に格納する。スタートＸは２つのローカルメモリ手段（例えばレジスタ）に格納され、以降、現在Ｘとラインスタートと称する。スタートＹは２つのローカルメモリ手段に格納され、以降、現在Ｙとラインスタートと称する。現在Ｘと現在Ｙはソースビットマップ内の現在の位置を参照し、従って、対応する画素色を参照する。ビットマップフィルを生成するフィル生成手段１２１０は、この画素色を、現在のディスプレイ出力色として用いる。レンダリングエンジン４３０が、スキャンラインに沿って右へ隣接する画素に対して繰り返すにつれて、フィル生成手段１２１０はビットマップフィルデータのデルタＸの値だけカレントＸを増加し、ビットマップフィルデータのデルタＹの値だけカレントＹを増加する。この手段により、ソースビットマップの画素の座標は、出力を描画するための要求されたレートと要求された順序で追跡され得る。

【００９３】

レンダリングエンジンがスキャンラインのためのデータ出力を終えると、バウンディングボックスの左側の辺が出力ディスプレイの次のスキャンラインと出会う位置に対応するソースビットマップ中の位置を表すように、ラインスタートＸとラインスタートＹの新たな値が計算される。フィル生成手段１２１０はラインスタートＸをデルタラインスタートＸだけ増加することにより、そしてラインスタートＹをデルタラインスタートＹだけ増加することにより、これを実行する。これらのラインスタートＸ及びラインスタートＹに対する新しい値も現在Ｘ及び現在Ｙにロードされる。これらは、再び、新しいスキャンラインに対してビットマップ中の位置を追跡する。

【００９４】

ビットマップフィルデータのパラメータ、最大Ｘと最大Ｙは、ソースビットマップの大きさを表す整数値である。フィル生成手段１２１０が、ローカルに格納されたカレントＸ

とカレントＹの値が最大Ｘと最大Ｙを越えることを検出した場合、新たなカレントＸとカレントＹが、それらから最大Ｘと最大Ｙを差し引くことによって算出される。同様に、カレントＸとカレントＹがゼロより小さくなったときは、最大Ｘと最大Ｙをそれぞれに加算することで新しいカレントＸとカレントＹが算出される。

【００９５】

ソース画素のアドレスは、ビットマップフィルデータからのビットマップベースアドレスと、画素毎のバイト数を用いて、カレントＸとカレントＹから以下の式により決定される。

画素アドレス＝ビットマップベースアドレス＋（floor(カレントＹ)×最大Ｘ＋floor(カレントＸ)）×画素毎のピクセル数

ここで、floor(カレントＸ)とfloor(カレントＹ)は、それぞれカレントＸとカレントＹの整数部である。

【００９６】

この計算は、フィルデータの出力画素毎に２つの乗算を実行するという望ましくない要求を含む。これは以下の好ましい実施形態によって解消される。

【００９７】

フィル生成手段１２１０は、ローカルメモリ手段内に“現在アドレス”値を格納する。それは、現在Ｘと現在Ｙにおいて維持される座標により参照されるソースビットマップ内の画素のアドレスに対応する。これは、フィルテーブル内のビットマップフィルデータの付加的なパラメータとして提供される、“スタートリードアドレス”の値とともに最初にロードされる。スキャンラインに沿ってレンダリングエンジンが繰り返す毎に、現在Ｘと現在Ｙは増加される。現在アドレスは、ソースビットマップ内の次の要求された画素のアドレスを決定するために増加され得る。しかし、現在アドレスの増加する量は、現在Ｘと現在Ｙが増加されたときに、それらの小数部がそれらのそれぞれの整数部にキャリーを生成したかどうかに依存する。現在アドレスの要求される増加量は、以下に示す４つの値のうちのいずれかである。

●現在Ｘの少数がキャリーを発生せず、現在Ｙの小数部がキャリーを発生しない場合、増加＝[ビットマップ内の画素毎のバイト数]×（[デルタＸの整数部]＋（[デルタＹの整数部]×[最大Ｘ]））

●現在Ｘの少数がキャリーを発生し、現在Ｙの小数部がキャリーを発生しない場合、増加＝[ビットマップ内の画素毎のバイト数]×（[デルタＸの整数部]＋１＋（[デルタＹの整数部]×[最大Ｘ]））

●現在Ｘの少数がキャリーを発生せず、現在Ｙの小数部がキャリーを発生した場合、増加＝[ビットマップ内の画素毎のバイト数]×（[デルタＸの整数部]＋（（[デルタＹの整数部]＋１）×[最大Ｘ]））

●現在Ｘの少数がキャリーを発生し、現在Ｙの小数部がキャリーを発生した場合、増加＝[ビットマップ内の画素毎のバイト数]×（[デルタＸの整数部]＋１（[デルタＹの整数部]＋１×[最大Ｘ]））。

【００９８】

フィル生成モジュール５５４は、ビットマップフィルデータ中の事前に計算されたデータとして提供されることになるこれら４つの増加値を要求しても良い。フィル生成手段１２１０は、現在Ｘと現在Ｙの増加の結果に依存してどの増加を用いるかを判断する。

【００９９】

一つのディスプレイスキャンラインと次との間のカレントアドレスを追跡するときに、上述したのと同じテクニックが用いられる。フィル生成手段１２１０は最初にカレントアドレスメモリ手段に“スタートリードアドレス”の値を格納する。また、それは、“スタートリードアドレス”を更なるメモリ手段へ格納する。以下、これを“ラインスタートアドレス”と称する。このアドレスは、ラインスタートＸとラインスタートＹによって参照されるソースビットマップ中の画素のアドレスである。ラインスタートＸとラインスタートＹは、レンダリングエンジンが新しいスキャンラインに対して繰り返すときに増加され

るので、それにつれてラインスタートアドレスも増加される。その結果値は、カレントアドレスに書き込まれる。要求される増加は、ラインスタートＸとラインスタートＹのいずれかの増加の間にキャリーが発生したかどうかに依存して、上記４つの値のうちの一つとなる。ＴＣＩＥ４１０は、これら４つの可能な増加値が、ビットマップフィルデータ内の事前に計算されたデータとして提供されることを要求する。

【０１００】

ＴＣＩＥ４１０は、単一ビットマップとして、或いは、リスト又は配列によって参照される複数の小さなタイルとして、ソースビットマップがメモリに提供されることを許容する。後者の表現において、タイルビットマップはメモリ中の任意の位置を占め、リスト又は配列は、ビットマップイメージ全体の左上に対応するタイルが最初に参照されるように、これらをスキャン順に参照するのに用いられる。一つの実施形態において、タイルの大きさは１６画素×１６画を、或いは３２画素×３２画素である。タイル化して格納することによる利点は、レンダリングに際してＴＣＩＥ４１０が大きなソースビットマップの限定された領域を要求する場合に明らかとなる。その領域に必要なタイルのみがローカルメモリにおいて有効となるように要求される。さらに、タイルはイメージのローカライズされた領域を表し、そして、これらの領域のためのデータが隣接したメモリに格納されることを確実にするので、ローテーションのような繰り返しによる画素演算を、メモリ内の頻繁なランダムサイズのジャンプ無しに実行できる。これは、メモリ手段が、メモリページの切り替えにかなりの待ち時間を課するＤＲＡＭである場合に得に好ましい。ビットマップを表すためのタイルの使用は、イメージに対する全てのタイルが必要なときに有効となっていることを保証できない場合に特に有用である。タイルへの参照の配列又はリストは、不在を示すことができ、このような事態を最小にするためにアクションをとることができるからである。

【０１０１】

フィル生成モジュール５５４の各フィル生成手段１２１０は、ｚ－レベルアクティベーションモジュールによってアクティブであると判断されたｚ－レベル（ランカリングモジュールによってフィルタされたであろう）に対応する出力画素データを生成する。複数の出力画素データは、メッセージの手段により領域合成モジュール５５６へ渡される。フィル生成モジュール５５４は、フィル生成手段１２１０のいずれかの出力データが変化するとき、メッセージを生成する。例えば、ｚ－レベルアクティベーションモジュール５５０が、ｚ－レベルがデアクティベートされたことを示すメッセージをフィル生成モジュール５５４へ渡した場合、フィル生成モジュール５５４は当該ｚ－レベルに関連するフィル生成手段をデアクティベートし、このことが発生したことを示すメッセージを領域合成モジュールへ渡すことにより応答する。

【０１０２】

領域合成モジュール５５６へ渡されたメッセージは描画中の最高位のアクティブなｚ－レベルに対応するＸディスプレイ座標と複数の画素データを含む。すでにわかるように、フィルテーブル５１４は、関連するｚ－レベルのフィルデータが透明度を有するかどうかを示すNEED_BELOWフラグを含んでも良い。フィル生成手段１２１０の一つが、NEED_BELOWフラグをfalse（すなわちクリア）にセットした、テーブル中のｚ－レベルに対するデータを生成する場合、フィル生成モジュール５５４は、より低いインデックスのあらゆるｚ－レベルについて、画素データを領域合成モジュール５５６に送る必要はない。

【０１０３】

領域合成モジュール５５６の目的は、受信したメッセージの画素データをフレームストア１６０又はディスプレイ４７０へ渡される単一の色値と合成することである。領域合成モジュール５５６は、最低位のｚ－レベルに関連する画素データが最初に読まれるように画素データを読む。次に最高位のｚ－レベルが読まれ、これの透明度成分がこの高いｚ－レベルの色と前に読まれた低いｚ－レベルの色とを混ぜるための重みファクタとして用いられる。例えば、ｚ－レベルに対する画素データが３つの色成分（赤、緑、青）と、４番目に０から２５５の範囲の値をとる透明度成分を含む場合、２つのｚ－レベルからの色成

分は以下の式を用いて混ぜ合わされる。
$C = ((C_{higher} \times a) + (C_{lower} \times (255 - a)) / 255$
ここで、ａは透明成分であり、ａ＝２５５の場合は完全に不透明な画素データを表し、ａ＝０は完全に透明な画素データを表す。
【０１０４】
この場合によって得られた新しい画素データは、メッセージの次に高いｚ－レベルのデータと、同じ手段を用いて合成される。これは、メッセージの、全てのｚ－レベルが合成されるまで続く。
【０１０５】
これらの動作の結果は、全体として、ランの連続としてのフレームに対するディスプレイ更新を表す。ここで各ランは、スタートＸ、Ｙ座標、長さ、画素色値によって特定される。ランカリングモジュール２１０，５５２が省略されると、フレームバッファのあらゆる画素をカバーするランが生成されることになる。しかし、ランカリングモジュール２１０，５５２を有すると、そのようなランの生成ははるかに少なくなり、ランの生成とフレームバッファ１６０又はディスプレイ４７０へのランのペインティングの両方において、多大な計算時間の節約となる。
【０１０６】
上述したように、ランカリングモジュール５５２の動作は、保持されるランリスト２２０（又は、プール５２０）の格納要求に基づいて最適化される。現実的な実施において、動作基準は、好ましくは、最悪の場合（これはランカリングモジュールの省略と等価である）でも、パフォーマンスが劣化しないことである。これは、もちろん、上述の式に従ってメモリの有効性が失効した場合であろう。この例は、図１０において、メモリ要求が限度を超え、スパンＡ３に関連するレコードの上書きが生じたことを想定することにより理解できよう。そのようなことが発生すると、スパンＢ３は、その全体を描画することが必要となる。しかし、それにもかかわらず、フレームＢで複写されるＡ１とＡ４の保有を通して、及び、スパンＢ２により大部分が複写されるスパンＡ２の一部の保有を通して、節約が得られるであろう。
【０１０７】
また、本発明者らは、処理オーバーヘッドに起因して、ここで説明されたランカリング動作が、例えば３２から６４ピクセルよりも小さい、非常に小さなランに対して、明らかな節約を提供しないと判断した。しかしながら、例えば、不透明のオブジェクトを有する“マンガ”スタイルのアニメーションに関して、実験は８０％までのレンダリング処理時間の節約を示した。
【０１０８】
さらに、図１０の例は不透明オブジェクトに関するが、ここで説明された原理は、透明成分を有するオブジェクトにも等しく適用できる。そのような例において、レンダリングにおける相違は、フィル生成モジュール５５４に渡されるアクティブなオブジェクトの数（説明された実施形態では４つに制限されている）と、領域合成モジュール５５６によって、オブジェクトの透明度のために実行されるべく要求されることになる付加的な処理のみである。
【０１０９】
［産業上の利用性］
説明された構成は、動画の画像が要求されるコンピュータ及びデータ処理の産業に適用される。この例としては、ポータブルゲームデバイスがあげられる。特に、テレフォンハンドセット３５０上で行われるゲームに関連して図３に示したような通信ネットワークを介してゲームがなされるゲームデバイスを挙げることができる。そのような例において、図６Ａ、６Ｂに示された処理の大部分は、ネットワーク３２０内のサーバコンピュータによって実行され、ハンドセット３５０はサーバへのユーザコマンドを入力するのに用いられる。サーバはそれらのコマンドを解釈して、インストラクション４６０、オブジェクト４６２、フィル４６４を形成し、フィルデータ５３２とｚ－レベルデータ５３４をハンド

セット３５０へ渡し、ランカリング５５２を含むまでレンダリング動作を実行する。サーバはランを出力しても良く、それによりハンドセット３５０はディスプレイ３５２へのフィル生成と合成を実行する。それは、ハンドセット３５０の計算のオーバーヘッドを最小にし、それによって主要なコストやバッテリ寿命を延ばしながら、処理スピードやサーバとの対話性を向上する。これは、特に、ハンドセットユーザ間でゲームをする場合に重要である。動作の別の態様は、グラフィックオブジェクト、或いはグラフィックオブジェクトの更新を、ネットワーク３２０からテレフォンハンドセット３５０へ送り、ハンドセット３５０内で描画パイプラインの全体を実現する。

【０１１０】

図３において、リモートデバイスは、ポータブルテレフォンハンドセット３５０であるが、他のデバイスを用いても良いし、ポータブルに限られるものでもない。そのような例としては、コンピュータ駆動の広告表示のようにディスプレイが特定の目的に固定されるもの、或いはコピー機のようなデバイスの操作制御の部分を形成するディスプレイのようなものがあげられる。

【０１１１】

以上の説明は、本発明のいくつかの実施形態について述べて物であり、それらに対する改良及び／又は変更が、本発明の範囲及び精神から逸脱することなくなされ得るものであり、実施形態はその例示であり限定するものではない。

【図面の簡単な説明】

【０１１２】

本発明の少なくとも一つの実施形態が以下の図面を参照して説明される。

【図１】従来技術であるクイゼルアルゴリズムのデータフローの概要を示すブロック図である。

【図２】本発明に従った改造を示す、図１に類似のブロック図である。

【図３】本実施形態の構成を実現するコンピュータシステムを示す図である。

【図４】図２の構成の好ましい実施によるデータフローを示す図である。

【図５Ａ】オブジェクト、ステップ、エッジ及びフィルの例を示す図である。

【図５Ｂ】オブジェクト、ステップ、エッジ及びフィルの例を示す図である。

【図５Ｃ】オブジェクト、ステップ、エッジ及びフィルの例を示す図である。

【図６Ａ】図４に示す構成の詳細を示す図である。

【図６Ｂ】図４に示す構成の詳細を示す図である。

【図７Ａ】エッジ処理モジュールの動作を説明するフローチャートである。

【図７Ｂ】エッジ処理モジュールの動作を説明するフローチャートである。

【図８】エッジのオーバーラップの状態を説明する図である。

【図９Ａ】Ｚ－レベルアクティベーションモジュールの動作を示すフローチャートである。

【図９Ｂ】Ｚ－レベルアクティベーションモジュールの動作を示すフローチャートである。

【図１０】ランカリングモジュールの基本的な動作を示す図である。

【図１１】ランカリングモジュールの動作を説明するフローチャートである。

【図１２】フィル生成モジュールの動作を示す図である。

【図１】

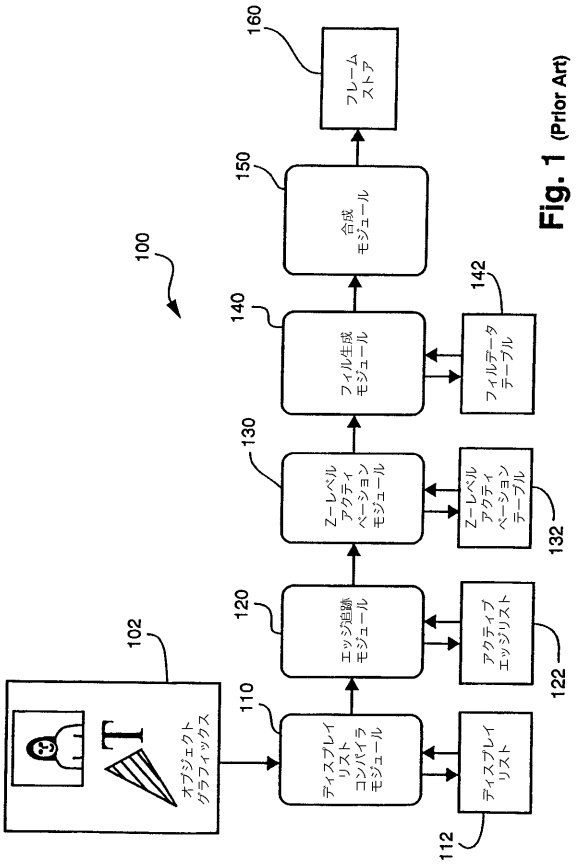


Fig. 1 (Prior Art)

【図２】

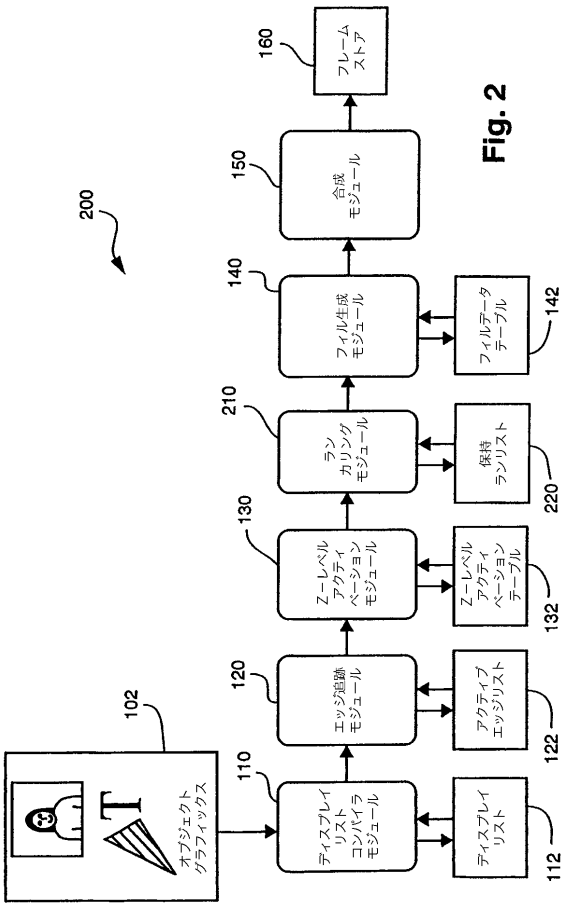


Fig. 2

【図３】

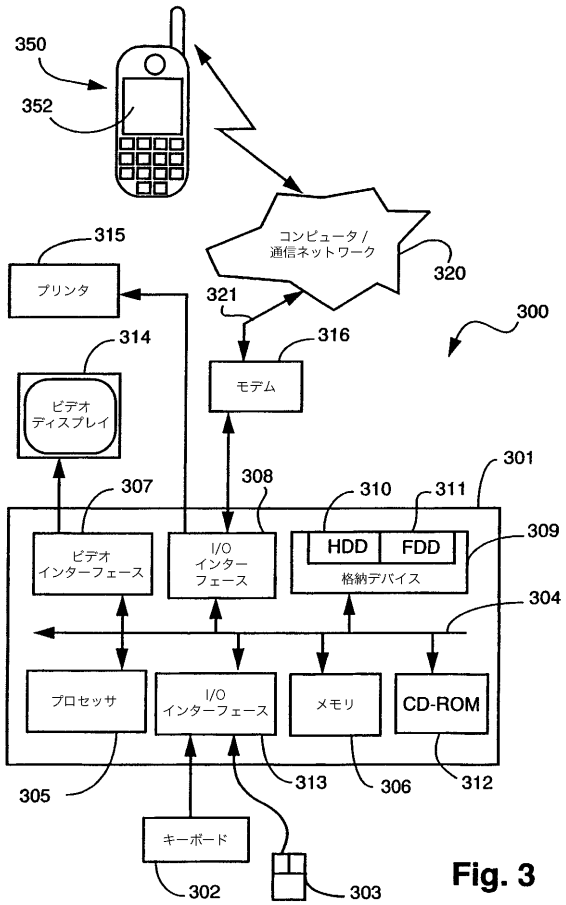


Fig. 3

【図４】

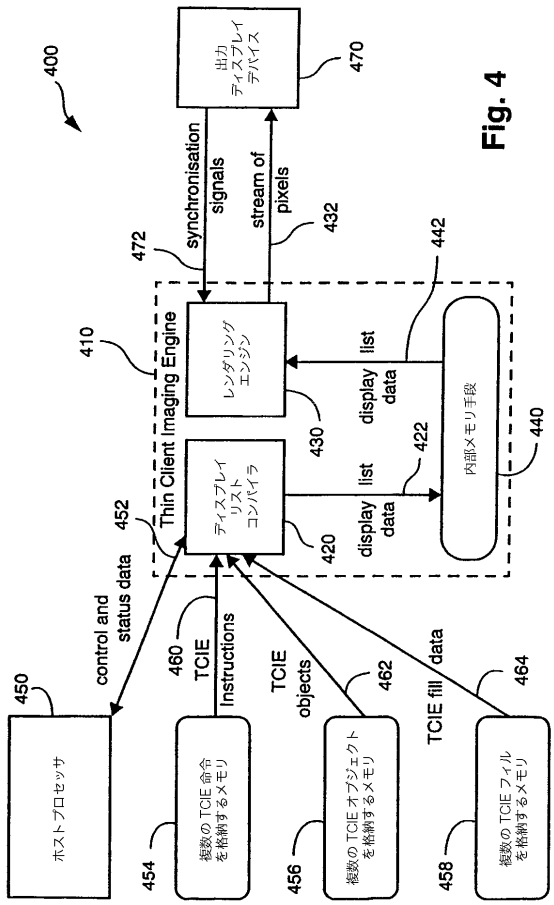


Fig. 4

【図５Ａ】

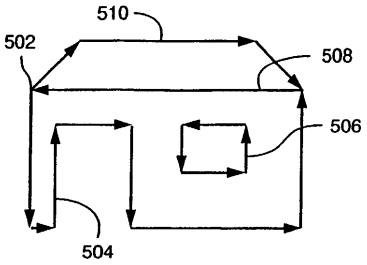


Fig. 5A

【図５Ｂ】

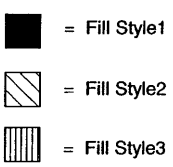


Fig. 5B

【図５Ｃ】

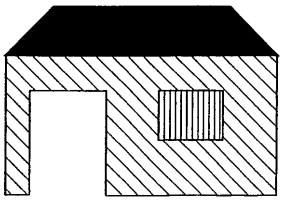


Fig. 5C

【図６Ａ】

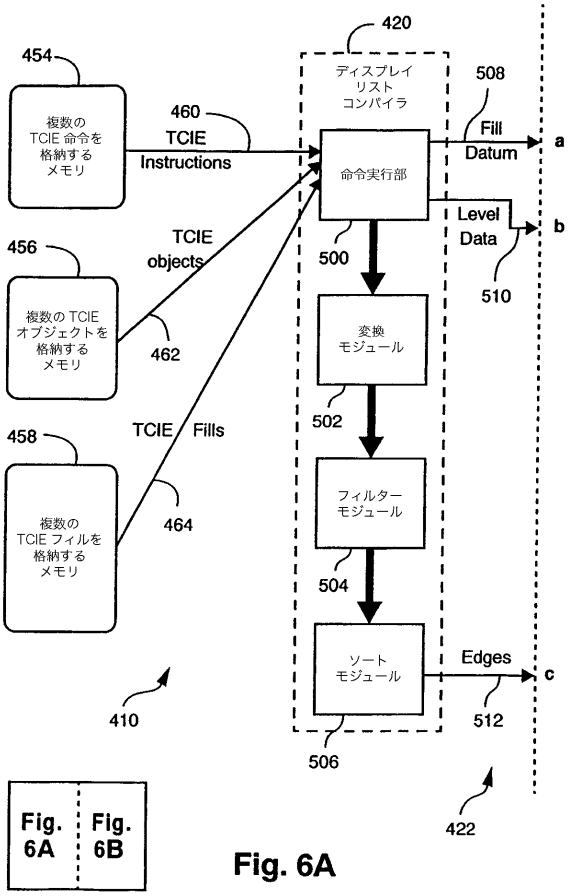


Fig. 6A Fig. 6B

Fig. 6A

【図６Ｂ】

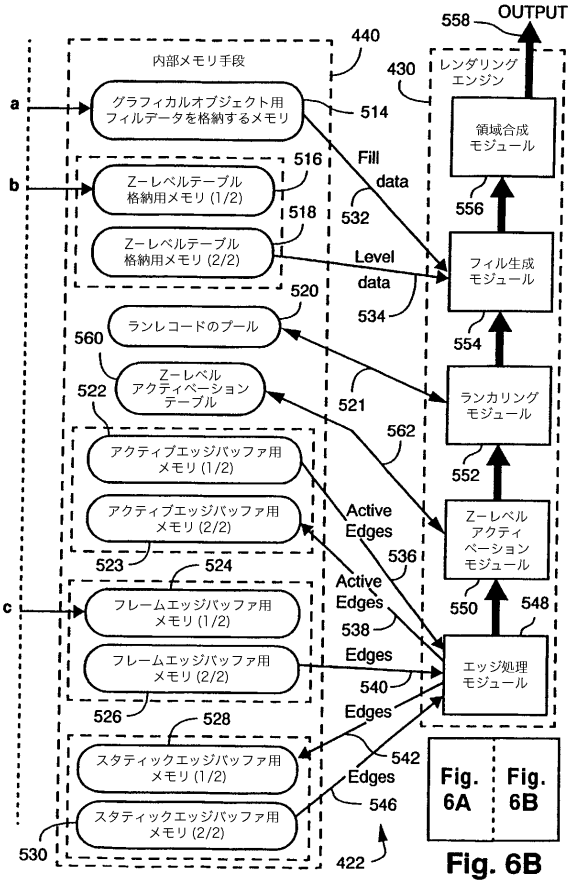


Fig. 6A Fig. 6B

Fig. 6B

【図７Ａ】

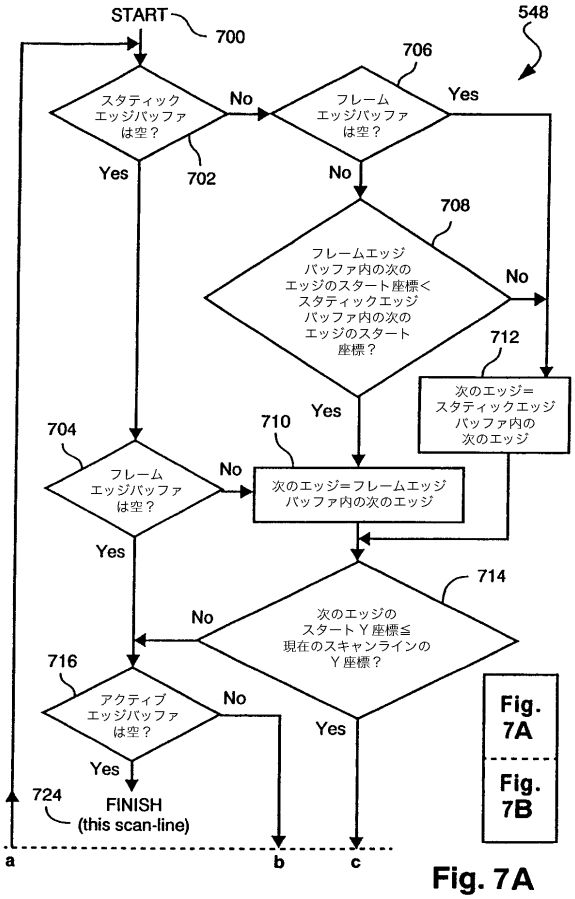


Fig. 7A Fig. 7B

Fig. 7A

【図７Ｂ】

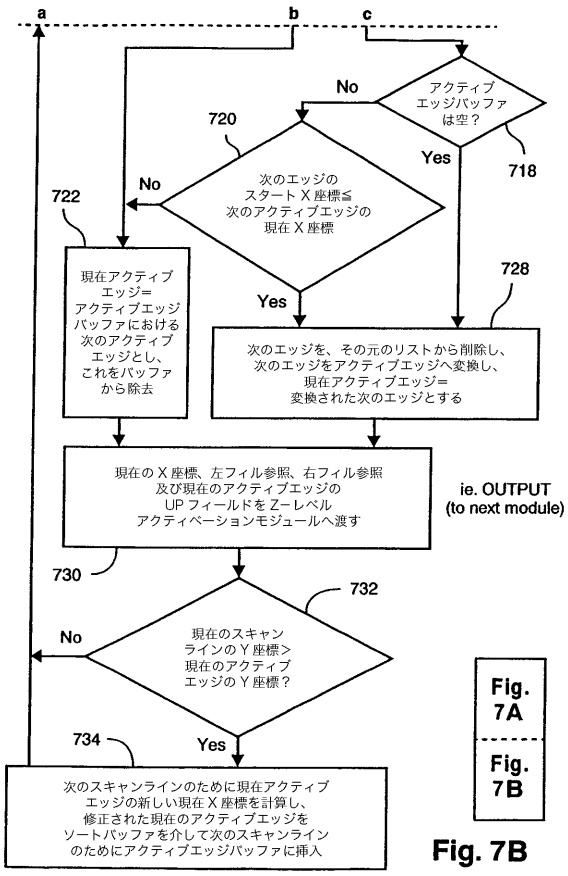


Fig. 7B

【図８】

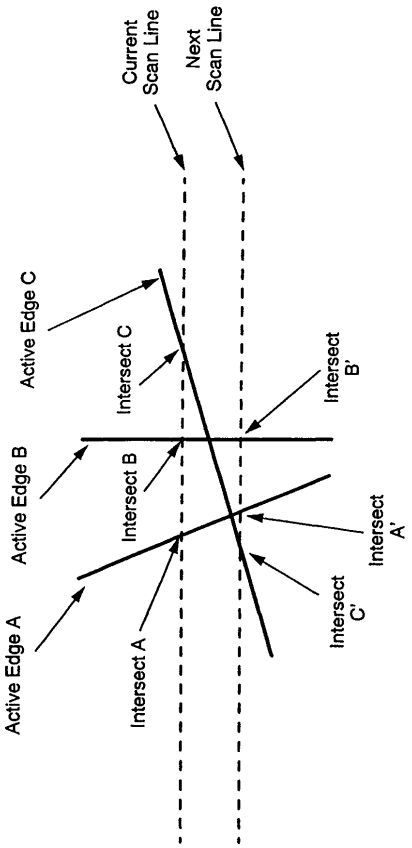
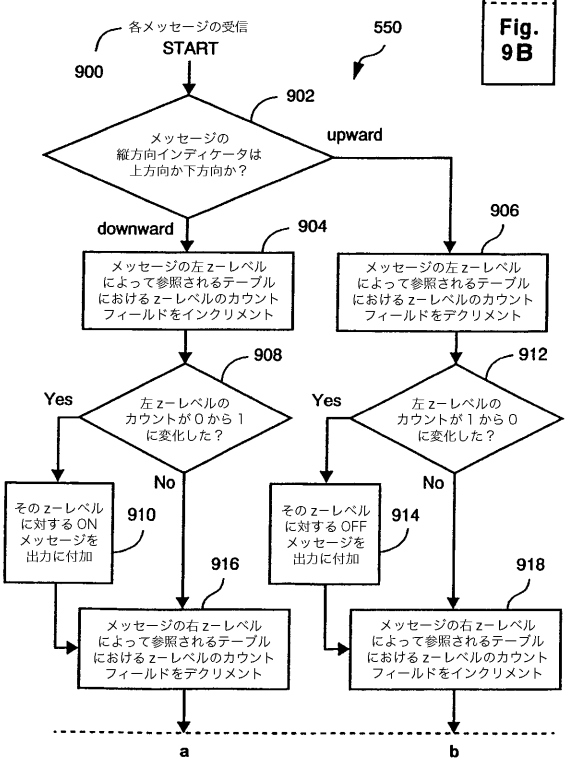


Fig. 8

【図９Ａ】

Fig. 9A



【図９Ｂ】

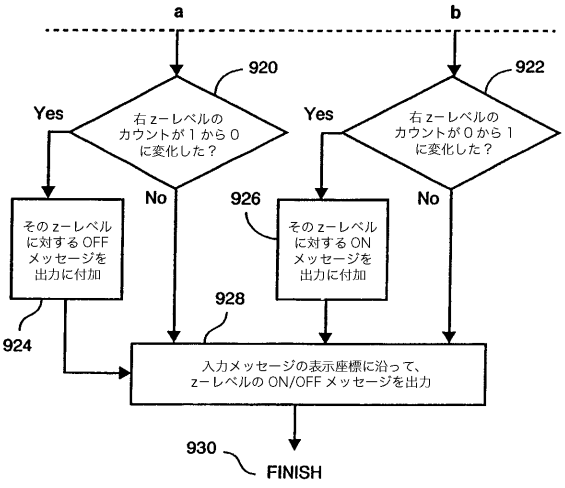


Fig. 9B

【図１０】

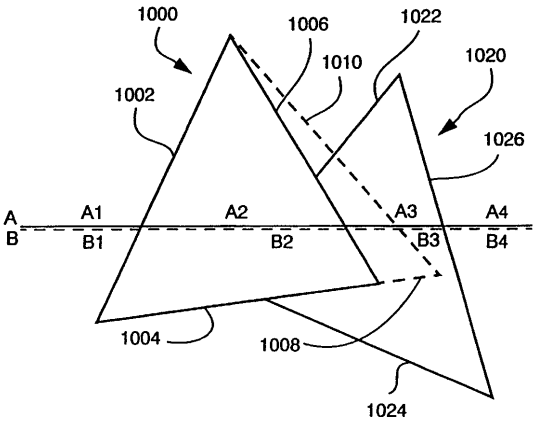


Fig. 10

【図１１】

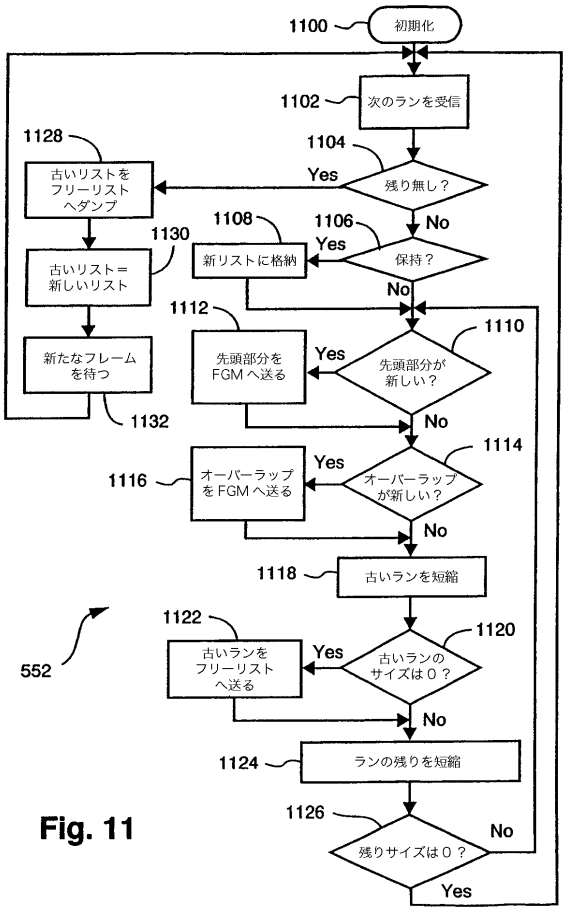


Fig. 11

【図１２】

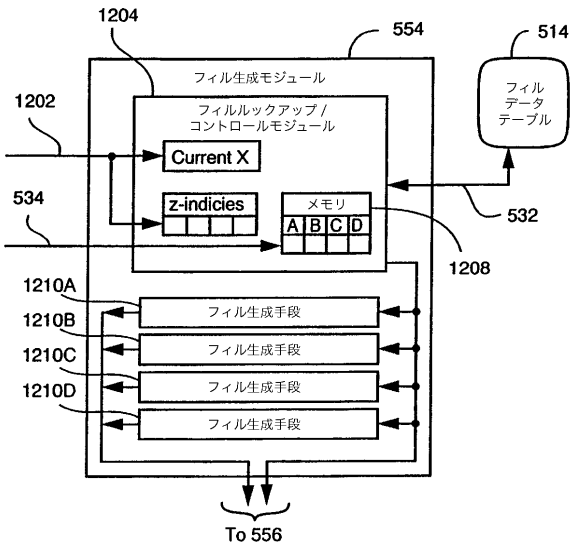


Fig. 12

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 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 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

審査官 松尾 俊介

(56)参考文献 米国特許第０５２９５２３５（ＵＳ，Ａ）
特開２０００－１３７８２５（ＪＰ，Ａ）
米国特許第０６１１１５８２（ＵＳ，Ａ）
特開２０００－１４８１３３（ＪＰ，Ａ）
特開２０００－１４９０３６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06T1/00～15/00