(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4365950号
(P4365950)
(45)発行日　平成21年11月18日(2009.11.18)　(24)登録日　平成21年8月28日(2009.8.28)

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

請求項の数 21　外国語出願　(全 62 頁)

(21)出願番号　特願平11-255123
(22)出願日　平成11年9月9日(1999.9.9)
(65)公開番号　特開2000-149035(P2000-149035A)
(43)公開日　平成12年5月30日(2000.5.30)
審査請求日　平成18年9月11日(2006.9.11)
(31)優先権主張番号　PP5854
(32)優先日　平成10年9月11日(1998.9.11)
(33)優先権主張国　オーストラリア(AU)
(31)優先権主張番号　PP5858
(32)優先日　平成10年9月11日(1998.9.11)
(33)優先権主張国　オーストラリア(AU)
(31)優先権主張番号　PP5859
(32)優先日　平成10年9月11日(1998.9.11)
(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】高速ラスタ形式レンダリングのためのグラフィックオブジェクト処理方法および装置

(57)【特許請求の範囲】
【請求項１】
ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する方法であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、該処理は前記辺レコードの処理の間に、
限られた数の処理された辺レコードを順序付けされていない第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて、順序付け可能な前記辺レコードを漸進的に順次に第２バッファへ送るステップと、
順序付けできない処理済の辺を、第３バッファにおいて並べるために該第３バッファへ送るステップと、
前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定するステップとを備えることを特徴とする方法。
【請求項２】
前記第３バッファにおける前記辺レコードの順序付けは、前記第３バッファへ前記辺を追加する際にな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】
前記辺レコードは、前記第３バッファへ挿入され、ソートされることを特徴とする請求項２に記載の方法。

【請求項４】
前記第３バッファ内の辺は、前記現在のスキャンラインの処理の完了時に順序付けられることを特徴とする請求項３に記載の方法。
【請求項５】
前記第２及び第３バッファは結合されて、前記選択的処理に用いられる第４バッファのそれぞれの部分を形成し、それにより、前記部分からの順序付けられた辺が漸進的に比較され、前記処理に対する次の辺を決定することを特徴とする請求項２に記載の方法。
【請求項６】
前記現在のスキャンラインの処理の完了時において、前記第２及び第３バッファの内容と前記第４バッファの内容をスワップすることを特徴とする請求項５に記載の方法。
【請求項７】
前記第２、第３、第４バッファの各々は、そのスタート位置及びエンド位置のポインタを含み、前記現在のスキャンラインの終了時に、バッファをスワップするために該ポインタをスワップすることを特徴とする請求項６に記載の方法。
【請求項８】
前記第１バッファは、メモリに形成された辺プールを備え、前記方法は、前記メモリに形成された複数のアクティブ辺レコードから、前記現在のスキャンラインのためのアクティブ辺値を決定し、前記アクティブ辺値の各々に対し、対応する辺レコードを前記辺プールへ送るステップを更に備えることを特徴とする請求項１に記載の方法。
【請求項９】
前記第２バッファは、前記メモリに形成される現在辺出力リストを備え、前記方法は、前記アクティブ辺値の各々に対して、前記辺プール内のレコードと前記辺レコードの対応する辺レコードを調べ、該辺レコードの順序付けられたものを前記現在辺リストに送るステップを更に備えることを特徴とする請求項８に記載の方法。
【請求項１０】
前記第３バッファは、前記メモリ内に形成された現在スピル出力リストを備え、前記方法は、前記現在のスピルリストへ順次に前記辺プールからの前記現在辺リストに並べることのできない辺レコードを送るステップを更に備えることを特徴とする請求項９に記載の方法。
【請求項１１】
前記方法は、前記スキャンラインの処理の完了時において、前記後続のスキャンラインの処理のために前記出力リストの対応する一つへ順次に前記辺プールからの辺レコードをフラッシュするステップと、ここで、このフラッシュに際しては前記出力リストは現在辺入力リスト及び現在辺入力スピルリストとしてそれぞれ割り当てられるものであり、該後続のスキャンラインに対する前記アクティブ辺値を決定するために前記入力リストから順次に辺レコードを前記アクティブ辺レコードの対応する一つに転送するステップとを備えることを特徴とする請求項１０に記載の方法。
【請求項１２】
グラフィックオブジェクト描画システムにおいて、ラスター画素イメージを形成するべくグラフィックオブジェクトを処理する方法であって、該処理は、前記グラフィックオブジェクトの辺と前記ラスター画素イメージの現在のスキャンラインとの交差の順序を決定する第１処理を備え、
前記システムが、
現在のスキャンラインと後続のスキャンラインの各々に関する複数の辺レコードと、該辺レコードの各々は少なくとも対応するスキャンライン上の対応する辺の画素位置の値を保持し、前記現在の辺レコード及び後続の辺レコードの各々は、少なくとも主部分とスピル部分に分割され、少なくとも前記現在の辺レコードの主部分はラスター画素順に並べられ、
少なくとも一つの現在のアクティブ辺レコードと、
スピルアクティブ辺レコードと、

制限された所定数の辺レコードを含むプールとを備え、
前記方法が、
（ａ）前記現在の辺レコードの前記主部分及びスピル部分の各々からの第１の辺レコードを対応するアクティブ辺レコードに送るステップと、
（ｂ）前記ラスタ画素順において最も低い値を有するアクティブ辺レコードを決定し、現在の辺の値及びレコードとしてその値とレコードを出力するために、前記アクティブ辺レコードの値を比較するステップと、
（ｃ）前記現在の辺レコードを、前記後続のスキャンラインのための対応する辺の値で更新するステップと、
（ｄ）更新された辺の値を前記プール内の辺の値と比較するステップと、ここで、更新された辺の値が前記プール内の辺の値より小さい場合に、
（ｄａ）前記更新された現在の辺レコードが後続の辺レコードのスピル部分へ送られ、さもなければ、
（ｄｂ）（ｄｂａ）最小の辺値を有する辺レコードが前記プールから、前記後続の辺レコードの主部分の次のレコードへ送られ、
（ｄｂｂ）前記更新された辺レコードは、前記サブステップ（ｄｂａ）で空になった前記プールのレコードへ送られ、
（ｄｃ）更なる辺レコードが、現在の辺レコードの対応する部分から更新された辺レコードによって空にされたアクティブ辺レコードへ送られ、
（ｅ）前記プール内の前記レコードの各々が占領されるまで、ステップ（ｂ）乃至（ｄ）を繰り返すステップと、これによって前記プールの最小の辺値レコードが前記後続の辺レコードの前記主部分へ送られ、
（ｆ）前記現在のレコードの全てのレコードが更新されるまで前記ステップ（ｂ）乃至（ｅ）を繰り返し、次いで、前記プールからのレコードを順次に前記後続のレコードの前記主部分内の次のレコードへフラッシュするステップと、
（ｇ）前記レコードを、前記後続のレコードにおける前記スピル部分において、ラスタ画素順にソートするステップと、
（ｈ）前記後続の辺レコードを前記現在の辺レコードへ送るステップと、
（ｉ）前記ラスタ画素イメージの更なるスキャンラインの各々に関して、前記ステップ（ａ）乃至（ｈ）を繰り返すステップとを備えることを特徴とする方法。
【請求項１３】
ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する装置であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、
未整列の第１バッファ、第２バッファ及び第３バッファを有するメモリと、
限られた数の処理された辺レコードを順序付けされていない前記第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて順序付け可能な前記辺レコードを漸進的に順次に前記第２バッファへ送り、順序付けできない処理済の辺を前記第３バッファにおいて並べるために該第３バッファへ送り、前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定するプロセッサとを備えることを特徴とする装置。
【請求項１４】
前記第１バッファは、メモリに形成された辺プールを備え、前記プロセッサは、前記メモリに形成された複数のアクティブ辺レコードから、前記現在のスキャンラインのためのアクティブ辺値を決定し、前記アクティブ辺値の各々に対し、対応する辺レコードを前記辺プールへ送るように構成されることを特徴とする請求項１３に記載の装置。
【請求項１５】
前記第２バッファは、前記メモリに形成される現在辺出力リストを備え、前記プロセッ

サは、前記アクティブ辺値の各々に対して、前記辺プール内のレコードと前記辺レコードの対応する辺レコードを調べ、該辺レコードの順序付けられたものを前記現在辺リストに送るように構成されることを特徴とする請求項１４に記載の装置。

【請求項１６】

前記第３バッファは、前記メモリ内に形成された現在スピル出力リストを備え、前記プロセッサは、前記現在のスピルリストへ順次に前記辺プールからの前記現在辺リストに並べることのできない辺レコードを送るように構成されることを特徴とする請求項１５に記載の装置。

【請求項１７】

前記プロセッサは、前記スキャンラインの処理の完了時において、前記後続のスキャンラインの処理のために前記出力リストの対応する一つへ順次に前記辺プールからの辺レコードをフラッシュし、ここで、このフラッシュに際しては前記出力リストは現在辺入力リスト及び現在辺入力スピルリストとしてそれぞれ割り当てられるものであり、該後続のスキャンラインに対する前記アクティブ辺値を決定するために前記入力リストから順次に辺レコードを前記アクティブ辺レコードの対応する一つに転送するように構成されることを特徴とする請求項１６に記載の装置。

【請求項１８】

ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する装置であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、

限られた数の処理された辺レコードを順序付けされていない第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて、順序付け可能な前記辺レコードを漸進的に順次に第２バッファへ送る手段と、

順序付けできない処理済の辺を、第３バッファにおいて並べるために該第３バッファへ送る手段と、

前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定する手段とを備えることを特徴とする装置。

【請求項１９】

ラスター画素イメージを形成するべくグラフィックオブジェクトを処理するグラフィックオブジェクト描画システムの一部を形成する装置であって、該処理は、前記グラフィックオブジェクトの辺と前記ラスター画素イメージの現在のスキャンラインとの交差の順序を決定する第１処理を備え、

前記装置が、

現在のスキャンラインと後続のスキャンラインの各々に関する複数の辺レコードと、該辺レコードの各々は少なくとも対応するスキャンライン上の対応する辺の画素位置の値を保持し、前記現在の辺レコード及び後続の辺レコードの各々は、少なくとも主部分とスピル部分に分割され、少なくとも前記現在の辺レコードの主部分はラスター画素順に並べられ、

少なくとも一つの現在のアクティブ辺レコードと、

スピルアクティブ辺レコードと、

制限された所定数の辺レコードを含むプールと、

辺レコード処理のために、

（ａ）前記現在の辺レコードの前記主部分及びスピル部分の各々からの第１の辺レコードを対応するアクティブ辺レコードに送り、

（ｂ）前記ラスタ画素順において最も低い値を有するアクティブ辺レコードを決定し、現在の辺の値及びレコードとしてその値とレコードを出力するために、前記アクティブ辺レコードの値を比較し、

（ｃ）前記現在の辺レコードを、前記後続のスキャンラインのための対応する辺の値で

更新し、
（ｄ）更新された辺の値を前記プール内の辺の値と比較し、ここで、更新された辺の値が前記プール内の辺の値より小さい場合に、
（ｄａ）前記更新された現在の辺レコードが後続の辺レコードのスピル部分へ送られ、さもなければ、
（ｄｂ）（ｄｂａ）最小の辺値を有する辺レコードが前記プールから、前記後続の辺レコードの主部分の次のレコードへ送られ、
（ｄｂｂ）前記更新された辺レコードは、前記サブステップ（ｄｂａ）で空になった前記プールのレコードへ送られ、
（ｄｃ）更なる辺レコードが、現在の辺レコードの対応する部分から更新された辺レコードによって空にされたアクティブ辺レコードへ送られ、
（ｅ）前記プール内の前記レコードの各々が占領されるまで、ステップ（ｂ）乃至（ｄ）を繰り返し、これによって前記プールの最小の辺値レコードが前記後続の辺レコードの前記主部分へ送られ、
（ｆ）前記現在のレコードの全てのレコードが更新されるまで前記ステップ（ｂ）乃至（ｅ）を繰り返し、次いで、前記プールからのレコードを順次に前記後続のレコードの前記主部分内の次のレコードへフラッシュし、
（ｇ）前記レコードを、前記後続のレコードにおける前記スピル部分において、ラスタ画素順にソートし、
（ｈ）前記後続の辺レコードを前記現在の辺レコードへ送り、
（ｉ）前記ラスタ画素イメージの更なるスキャンラインの各々に関して、前記ステップ（ａ）乃至（ｈ）を繰り返す構成を有することを特徴とする装置。

【請求項２０】
ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する装置のためのプログラムを格納するコンピュータ可読媒体であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、該プログラムが、
限られた数の処理された辺レコードを順序付けされていない第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて、順序付け可能な前記辺レコードを漸進的に順次に第２バッファへ送る保持ステップのコードと、
順序付けできない処理済の辺を、第３バッファにおいて並べるために該第３バッファへ送る転送ステップのコードと、
前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定する処理ステップのコードとを備えることを特徴とするコンピュータ可読媒体。

【請求項２１】
グラフィックオブジェクト描画システムにおいて、ラスター画素イメージを形成するべくグラフィックオブジェクトを処理するためのコンピュータプログラム製品を備えたコンピュータ可読媒体であって、該処理は、前記グラフィックオブジェクトの辺と前記ラスター画素イメージの現在のスキャンラインとの交差の順序を決定する第１処理を備え、該コンピュータ製品は、
現在のスキャンラインと後続のスキャンラインの各々に関する複数の辺レコードと、該辺レコードの各々は少なくとも対応するスキャンライン上の対応する辺の画素位置の値を保持し、前記現在の辺レコード及び後続の辺レコードの各々は、少なくとも主部分とスピル部分に分割され、少なくとも前記現在の辺レコードの主部分はラスター画素順に並べられ、
少なくとも一つの現在のアクティブ辺レコードと、
スピルアクティブ辺レコードと、
制限された所定数の辺レコードを含むプールと関連し、

前記第１処理が、
（ａ）前記現在の辺レコードの前記主部分及びスピル部分の各々からの第１の辺レコードを対応するアクティブ辺レコードに送るステップと、
（ｂ）前記ラスタ画素順において最も低い値を有するアクティブ辺レコードを決定し、現在の辺の値及びレコードとしてその値とレコードを出力するために、前記アクティブ辺レコードの値を比較するステップと、
（ｃ）前記現在の辺レコードを、前記後続のスキャンラインのための対応する辺の値で更新するステップと、
（ｄ）更新された辺の値を前記プール内の辺の値と比較するステップと、ここで、更新された辺の値が前記プール内の辺の値より小さい場合に、
（ｄａ）前記更新された現在の辺レコードが後続の辺レコードのスピル部分へ送られ、さもなければ、
（ｄｂ）（ｄｂａ）最小の辺値を有する辺レコードが前記プールから、前記後続の辺レコードの主部分の次のレコードへ送られ、
（ｄｂｂ）前記更新された辺レコードは、前記サブステップ（ｄｂａ）で空になった前記プールのレコードへ送られ、
（ｄｃ）更なる辺レコードが、現在の辺レコードの対応する部分から更新された辺レコードによって空にされたアクティブ辺レコードへ送られ、
（ｅ）前記プール内の前記レコードの各々が占領されるまで、ステップ（ｂ）乃至（ｄ）を繰り返すステップと、これによって前記プールの最小の辺値レコードが前記後続の辺レコードの前記主部分へ送られ、
（ｆ）前記現在のレコードの全てのレコードが更新されるまで前記ステップ（ｂ）乃至（ｅ）を繰り返し、次いで、前記プールからのレコードを順次に前記後続のレコードの前記主部分内の次のレコードへフラッシュするステップと、
（ｇ）前記レコードを、前記後続のレコードにおける前記スピル部分において、ラスタ画素順にソートするステップと、
（ｈ）前記後続の辺レコードを前記現在の辺レコードへ送るステップと、
（ｉ）前記ラスタ画素イメージの更なるスキャンラインの各々に関して、前記ステップ（ａ）乃至（ｈ）を繰り返すステップとを実行するように構成されてなることを特徴とするコンピュータ可読媒体。

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、オブジェクト・グラフィック要素のラスタ画素画像へのレンダリングに関し、具体的には、レンダリング処理の一部としての画素データのフレーム記憶装置またはライン記憶装置を使用しない、そのようなオブジェクト・グラフィック要素の画素画像データへの効率的なレンダリングに関する。
【０００２】
【従来の技術】
ほとんどのオブジェクト・ベース・グラフィックス・システムでは、ページまたは画面の画素ベース画像を保持するためにフレーム・ストアまたはページ・バッファが使用される。通常、グラフィック・オブジェクトの輪郭は、計算され、塗り潰され、フレーム・ストアに書き込まれる。二次元グラフィックスの場合、他のオブジェクトの手前にあるオブジェクトは、単純に背景オブジェクトの書込み後にフレーム・ストアに書き込まれ、これによって、画素単位で背景を置換する。これは、当技術分野では、「ペインタのアルゴリズム（Painter's algorithm）」として一般に知られている。オブジェクトは、最も奥のオブジェクトから最も手前のオブジェクトという優先順位で検討され、通常は、各オブジェクトが、スキャン・ラインの順序でラスタ化され、画素が、各スキャン・ラインに沿ったシーケンシャルな並びでフレーム・ストアに書き込まれる。
【０００３】

この技法には、基本的に２つの問題がある。第１の問題は、フレーム・ストア内のすべての画素への高速なランダム・アクセスが必要であることである。これは、新たに検討されるオブジェクトのそれぞれが、フレーム・ストア内のどの画素にも影響する可能性があるからである。このため、フレーム・ストアは、通常は半導体のランダム・アクセス・メモリ（ＲＡＭ）内に保持される。高解像度カラー・プリンタの場合、必要なＲＡＭの量は非常に多くなり、通常は１００Ｍバイトを超えるが、これは、コストがかかり、高速での動作が困難である。第２の問題は、多数の画素がペイント（レンダリング）され、時間的に後のオブジェクトによって上塗り（再レンダリング）されることである。時間的に前のオブジェクトによる画素のペイントは、時間の浪費になる。

【０００４】

大量のフレーム・ストアの問題を克服するための方法の１つが、「バンディング（banding）」の使用である。バンディングを使用する時には、一時にフレーム・ストアの一部だけがメモリ内に存在する。描画されるオブジェクトのすべてが、「表示リスト」に保存される。画像全体は上記と同様にレンダリングされるが、存在するフレーム・ストアの部分の外側をペイント（レンダリング）しようとする画素ペイント（レンダリング）動作は、「クリップ」アウトされる。オブジェクトのすべてが描画された後に、フレーム・ストアの部分を、プリンタ（または他の位置）に送り、フレーム・ストアの別の部分を選択し、この処理を繰り返す。この技法には、ペナルティが伴う。たとえば、描画されるオブジェクトは検討され、何度も（バンドごとに１回）再検討されなければならない。バンドの数が増えるにつれて、レンダリングが必要なオブジェクトの検査が繰り返されるし回数も増える。バンディングの技法は、上塗りのコストという問題を解決しない。

【０００５】

いくつかの他のグラフィック・システムでは、スキャン・ラインの順で画像が検討される。やはり、描画されるオブジェクトのすべてが、表示リストに保存される。各スキャン・ライン上で、そのスキャン・ラインと交差するオブジェクトが優先順位の順でオブジェクトごとに検討され、オブジェクトの辺の交差点の間の画素のスパンが、ライン・ストアにセットされる。この技法も、大量のフレーム・ストアの問題を克服するが、やはり上塗りの問題をこうむる。

【０００６】

これらのほかに、大きいフレーム・ストアの問題と上塗りの問題の両方を解決する技法がある。そのような技法の１つでは、各スキャン・ラインが順番に作られる。やはり、描画されるオブジェクトのすべてが、表示リストに保存される。各スキャン・ラインでは、そのスキャン・ラインと交差するオブジェクトの辺が、スキャン・ラインとの交差の座標の昇順で保持される。これらの交差の点または辺の交点が、順番に検討され、アクティブ・フラグの配列のトグルに使用される。そのスキャン・ライン上での対象となるオブジェクト優先順位ごとに１つのアクティブ・フラグがある。検討される辺の対のそれぞれの間で、最初の辺と次の辺の間にある各画素の色データが、アクティブ・フラグに対する優先順位エンコーダを使用して、どの優先順位が最も上にあるかを判定すること、および２つの辺の間のスパンの画素に関するその優先順位に関連する色を使用することによって生成される。次のスキャン・ラインに備えて、各辺の交差の座標が、各辺の性質に応じて更新される。この更新の結果として誤ってソートされた状態になった隣接する辺は、交換される。新しい辺も、辺のリストに合併される。

【０００７】

この技法は、フレーム・ストアまたはライン・ストアがなく、上塗りがなく、位数Ｎ倍（このＮは優先順位の数）ではなく定数位数の時間でオブジェクト優先順位が処理されるという大きい長所を有する。

【０００８】

【発明が解決しようとする課題】

しかし、この技法には下記の複数の制限がある。

【０００９】

（ｉ）この技法は、辺からオブジェクトの内外の状態を決定するための、当技術分野で既知の「奇数－偶数」塗潰し規則だけをサポートする。多数のグラフィック記述言語の必須機能である「非ゼロ・ワインディング」塗潰し規則は、この技法によってサポートされない。

【００１０】

（ｉｉ）単純な交換技法が修復に不適切である場合に、大量の誤ったソートが発生する可能性がある。各スキャン・ライン上で辺リスト全体のブルート・フォース・ソート（brute-force sort）を実行することができるが、これは非常に低速である。

【００１１】

（ｉｉｉ）この技法は、オブジェクト・タイプとしてのラスタ（画素ベース）画像をサポートしない。このような画像は、ほとんどのグラフィック記述言語の必須機能である。

【００１２】

（ｉｖ）この技法は、ペイントされる画素のそれぞれが、より低い優先順位のオブジェクトの画素を厳密に隠す、不透明のオブジェクトだけをサポートする。よって、この技法は、複数のグラフィック・オブジェクトの色が相互作用するラスタ演算をサポートしない。このような演算には、ＸＯＲモードでの描画または部分的に透明なオブジェクトの合成が含まれる。これらの変更演算は、ほとんどのグラフィック記述言語の必須機能である。

【００１３】

（ｖ）この技法は、１つまたは複数のクリップ形状によって、クリップ形状の境界の内側（または外側）にあるいくつかの他のグラフィック・オブジェクトを削除する、クリッピングをサポートしない。クリッピングは、ほとんどのグラフィック記述言語の必須機能である。

【００１４】

（ｖｉ）この技法では、オブジェクトの辺の、具体的にはテキストに関する、大量の非効率的な符号化が使用される。このようなグラフィック記述の極端に一般的な要素は、より単純な形で表現されることが望ましい。

【００１５】

（ｖｉｉ）この技法は、いくつかの場合に、１つまたは複数のオブジェクトのアクティビティが変数である複雑な合成式の正確な評価を提供しない。

【００１６】

この技法は、既存のグラフィック記述言語が必要とする多数の機能を実施できないので、その使用が極度に制限されている。

【００１７】

さらに、既存のレンダリング・インターフェースの中には、複数のグラフィック・オブジェクトの色のビット単位の論理組合せの実装を要求するものがあり、複数のグラフィック・オブジェクトの色のアルファ・チャネル（透明度、不透明度またはマットと称する）に基づく組合せの実装を要求するものもある。現在の技法では、この２つの機能を統一された形で実装することができない。

【００１８】

【課題を解決するための手段】

本発明の目的は、従来技術のシステムに伴う１つ又は複数の欠陥を、実質的に克服するか、少なくとも改善することである。

【００１９】

本発明の一つの態様によれば、ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する方法であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、該処理は前記辺レコードの処理の間に、

限られた数の処理された辺レコードを順序付けされていない第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて、順序付け

可能な前記辺レコードを漸進的に順次に第２バッファへ送るステップと、
順序付けできない処理済の辺を、第３バッファにおいて並べるために該第３バッファへ送るステップと、
前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定するステップとを備える方法が開示される。
【００２０】
また、本発明の他の態様によれば、グラフィックオブジェクト描画システムにおいて、ラスター画素イメージを形成するべくグラフィックオブジェクトを処理する方法であって、該処理は、前記グラフィックオブジェクトの辺と前記ラスター画素イメージの現在のスキャンラインとの交差の順序を決定する第１処理を備え、
前記システムが、
現在のスキャンラインと後続のスキャンラインの各々に関する複数の辺レコードと、該辺レコードの各々は少なくとも対応するスキャンライン上の対応する辺の画素位置の値を保持し、前記現在の辺レコード及び後続の辺レコードの各々は、少なくとも主部分とスピル部分に分割され、少なくとも前記現在の辺レコードの主部分はラスター画素順に並べられ、
少なくとも一つの現在のアクティブ辺レコードと、
スピルアクティブ辺レコードと、
制限された所定数の辺レコードを含むプールとを備え、
前記方法が、
（ａ）前記現在の辺レコードの前記主部分及びスピル部分の各々からの第１の辺レコードを対応するアクティブ辺レコードに送るステップと、
（ｂ）前記ラスタ画素順において最も低い値を有するアクティブ辺レコードを決定し、現在の辺の値及びレコードとしてその値とレコードを出力するために、前記アクティブ辺レコードの値を比較するステップと、
（ｃ）前記現在の辺レコードを、前記後続のスキャンラインのための対応する辺の値で更新するステップと、
（ｄ）更新された辺の値を前記プール内の辺の値と比較するステップと、ここで、更新された辺の値が前記プール内の辺の値より小さい場合に、
（ｄａ）前記更新された現在の辺レコードが後続の辺レコードのスピル部分へ送られ、さもなければ、
（ｄｂ）（ｄｂａ）最小の辺値を有する辺レコードが前記プールから、前記後続の辺レコードの主部分の次のレコードへ送られ、
（ｄｂｂ）前記更新された辺レコードは、前記サブステップ（ｄｂａ）で空になった前記プールのレコードへ送られ、
（ｄｃ）更なる辺レコードが、現在の辺レコードの対応する部分から更新された辺レコードによって空にされたアクティブ辺レコードへ送られ、
（ｅ）前記プール内の前記レコードの各々が占領されるまで、ステップ（ｂ）乃至（ｄ）を繰り返すステップと、これによって前記プールの最小の辺値レコードが前記後続の辺レコードの前記主部分へ送られ、
（ｆ）前記現在のレコードの全てのレコードが更新されるまで前記ステップ（ｂ）乃至（ｅ）を繰り返し、次いで、前記プールからのレコードを順次に前記後続のレコードの前記主部分内の次のレコードへフラッシュするステップと、
（ｇ）前記レコードを、前記後続のレコードにおける前記スピル部分において、ラスタ画素順にソートするステップと、
（ｈ）前記後続の辺レコードを前記現在の辺レコードへ送るステップと、
（ｉ）前記ラスタ画素イメージの更なるスキャンラインの各々に関して、前記ステップ（ａ）乃至（ｈ）を繰り返すステップとを備える方法が開示される。
【００２１】
また、本発明の他の態様によれば、ラスター画素イメージを形成するべく、グラフィッ

クオブジェクトを処理する装置であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、

未整列の第１バッファ、第２バッファ及び第３バッファを有するメモリと、

限られた数の処理された辺レコードを順序付けされていない前記第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて順序付け可能な前記辺レコードを漸進的に順次に前記第２バッファへ送り、順序付けできない処理済の辺を前記第３バッファにおいて並べるために該第３バッファへ送り、前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定するプロセッサとを備える装置が開示される。

【００２２】

また、本発明の他の態様によれば、ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する装置であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、

限られた数の処理された辺レコードを順序付けされていない第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられのに応じて、順序付け可能な前記辺レコードを漸進的に順次に第２バッファへ送る手段と、

順序付けできない処理済の辺を、第３バッファにおいて並べるために該第３バッファへ送る手段と、

前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定する手段とを備える装置が開示される。

【００２３】

また、本発明の他の態様によれば、ラスター画素イメージを形成するべく、グラフィックオブジェクトを処理する装置のためのプログラムを格納するコンピュータ可読媒体であって、該処理は、ラスタライズされた表示順における現在のスキャンラインのための対応する辺レコードを評価することにより、前記グラフィックオブジェクトの辺間の交差順序を判断し、後続のスキャンラインのための各辺に関する辺交差値を決定するものであって、該プログラムが、

限られた数の処理された辺レコードを順序付けされていない第１バッファに保持し、順序付け可能な処理済の辺レコードが前記第１バッファに加えられたのに応じて、順序付け可能な前記辺レコードを漸進的に順次に第２バッファへ送る保持ステップのコードと、

順序付けできない処理済の辺を、第３バッファにおいて並べるために該第３バッファへ送る転送ステップのコードと、

前記第２及び第３バッファからの辺レコードを選択的に処理し、後続のスキャンラインに対する並べられた交差を決定する処理ステップのコードとを備えるコンピュータ可読媒体が開示される。

【００２４】

本発明の更に他の態様は以下の説明から明らかとなる。

【００２５】

【発明の実施の形態】

以下、添付の図面を参照して本発明の実施形態を説明する。なお、以下の実施形態の記載において、数学記号を下記のように表記する。

【表１】

以下では、

$\lfloor A \rfloor$ を└A┘で表わし、

$\overline{A}$ をA‾で表わし、

$\overline{Dest\_Active}$ を(Dest_Active)‾と表わす

図１は、コンピュータ・グラフィック・オブジェクト画像のレンダリングおよびプレゼンテーションのために構成されたコンピュータ・システム１を概略的に示す図である。このシステムには、システム・ランダム・アクセス・メモリ（ＲＡＭ）３に関連するホスト・プロセッサ２が含まれ、システムＲＡＭ３には、不揮発性のハード・ディスク・ドライブ５または類似の装置と、揮発性の半導体ＲＡＭ４を含めることができる。システム１には、システム読取専用メモリ（ＲＯＭ）６も含まれ、システムＲＯＭ６は、通常は半導体ＲＯＭ７を基礎とし、多くの場合に、コンパクト・ディスク装置（ＣＤ－ＲＯＭ）８によって補足することができる。システム１には、ラスタ式に動作するビデオ表示装置（ＶＤＵ）またはプリンタなどの、画像を表示するための手段１０も組み込むことができる。

【００２６】

システム１に関して上で説明した構成要素は、バス・システム９を介して相互接続され、ＩＢＭ　ＰＣ／ＡＴタイプのパーソナル・コンピュータおよびそれから発展した構成、Ｓｕｎ　Ｓｐａｒｃｓｔａｔｉｏｎｓおよび類似物など、当技術分野で周知のコンピュータ・システムの通常動作モードで動作可能である。

【００２７】

やはり図１に図示されているように、画素シーケンシャル・レンダリング装置２０は、バス９に接続され、好ましい実施形態では、システム１からバス９を介して命令およびデータを供給されるグラフィック・オブジェクト・ベースの記述から導出される画素ベースの画像のシーケンシャル・レンダリングのために構成される。装置２０は、オブジェクト記述のレンダリングのためにシステムＲＡＭ３を使用することができるが、レンダリング装置２０は、通常は半導体ＲＡＭから形成される、専用のレンダリング・ストア配置３０と関連付けられることが好ましい。

【００２８】

次に図２を参照すると、好ましい実施形態の機能データ流れ図が示されている。図２の機能流れ図は、オブジェクト・グラフィック記述１１から始まる。このオブジェクト・グラフィック記述１１は、ホスト・プロセッサ２によって生成されるか、且つ/又は、システムＲＡＭ３内に記憶されるかシステムＲＯＭ６から導出され、グラフィック・オブジェクトのパラメータを記述するのに使用され、そこから画素ベース画像をレンダリングするために、画素シーケンシャル・レンダリング装置２０によって解釈され得る。たとえば、オブジェクト・グラフィック記述１１には、ディスプレイ上の１点から別の点まで横断する直線の辺（単純ベクトル）または、直交する線を含む複数の辺によって二次元オブジェクトが定義される直交辺フォーマットを含むいくつかのフォーマットで辺を有するオブジェクトを組み込むことができる。これ以外に、連続曲線によってオブジェクトが定義されるフォーマットも、適当であり、これらには、乗算を実行する必要なしに二次曲線を単一の出力空間内でレンダリングできるようにするいくつかのパラメータによって単一の曲線を記述できる二次多項式の線分を含めることができる。三次スプラインや類似物などのそれ以外のデータ・フォーマットを使用することもできる。オブジェクトには、多数の異なる辺タイプの混合物を含めることができる。通常、すべてのフォーマットに共通するのは、

それぞれの線（直線であれ曲線であれ）の始点と終点の識別子であり、通常は、これらは、スキャン・ライン番号によって識別され、したがって、その曲線をレンダリングすることのできる特定の出力空間が定義される。

【００２９】

たとえば、図１６Ａに、線分を適当に記述し、レンダリングするために、２つの線分６０１および６０２に分割する必要がある辺６００の従来技術の辺記述を示す。分割の必要が生じるのは、従来技術の辺記述が、二次式を介して簡単に計算されるが、変曲点６０４に適応することができないからである。したがって、辺６００は、それぞれ終点６０３および６０４または終点６０４および６０５を有する２つの別々の辺として扱われた。図１６Ｂに、終点６１１および６１２と制御点６１３および６１４によって記述される三次スプライン６１０を示す。このフォーマットでは、レンダリングのために三次多項式の計算が必要であり、したがって、計算時間がかかる。

【００３０】

図１６Ｃに、好ましい実施形態に適用可能な辺の例を示す。好ましい実施形態では、辺は、単一の実体とみなされ、必要であれば、異なるフォーマットで記述できる辺の部分を示すために区分されるが、その具体的な目的は、各部分の記述の複雑さが最小限になるようにすることである。

【００３１】

図１６Ｃの左側には、スキャン・ラインＡ～Ｍの間にまたがる単一の辺６２０が示されている。辺は、ｓｔａｒｔ＿ｘ、ｓｔａｒｔ＿ｙ、辺の次の線分を指すアドレスを含む１つまたは複数の線分記述、および、辺の終了に使用される最終線分を含む複数のパラメータによって記述される。好ましい実施形態によれば、辺６２０は、３つのステップ線分、１つのベクトル線分および１つの二次線分として記述することができる。ステップ線分は、単純に、ｘステップ値とｙステップ値を有するものとして定義される。図示の３つのステップ線分の場合、線分記述は［０，２］、［＋２，２］および［＋２，０］である。ｘステップ値は符号付きであり、これによってステップの向きが示されるが、ｙステップ値は、必ず、スキャン・ラインの値が増えるラスタ・スキャン方向であるから符号なしであることに留意されたい。次の線分は、通常はパラメータｓｔａｒｔ＿ｘ、ｓｔａｒｔ＿ｙ、ｆｉｎｉｓｈ＿ｙおよび傾斜（ＤＸ）を必要とするベクトル線分である。この例では、ベクトル線分が辺６２０の中間線分であるから、ｓｔａｒｔ＿ｘおよびｓｔａｒｔ＿ｙは、前の線分から生じるので、省略することができる。傾斜値（ＤＸ）は、符号付きであり、前のスキャン・ラインのｘ値に加算されて、現行スキャン・ラインのｘ値を与え、図示の例では、ＤＸ＝＋１である。次の線分は、二次線分であり、これは、ベクトル線分に対応する構造を有するが、さらに、やはり符号付きであり、線分の傾斜を変更するためにＤＸに加算される２階値（ＤＤＸ）も有する。

【００３２】

図１６Ｃの右側の線分は、好ましい実施形態による三次曲線の例を示す図であり、これには上記の二次線分に対応する記述が含まれるが、線分の傾斜の変化の割合を変更するためにＤＤＸに加算される符号付きの３階値（ＤＤＤＸ）が追加されている。同様に、多数の他の階も実施することができる。

【００３３】

上記から、辺の線分を記述する複数のデータ・フォーマットを処理する能力があると、複雑で計算コストの高い数学演算に頼らずに、辺の記述と評価を単純化することができることが明白である。これに対して、図１６Ａの従来技術のシステムでは、直交、ベクトルまたは二次のいずれであれ、すべての辺を二次形式で記述する必要があった。

【００３４】

好ましい実施形態の動作を、図８に示された画像７８のレンダリングという単純な例に関して説明する。画像７８は、２つのグラフィカル・オブジェクト、具体的に言うと、不透明の赤色の長方形９０とその上にレンダリングされ、これによって長方形９０を部分的に隠す、部分的に透明の青色の三角形８０を含む。図からわかるように、長方形９０には、

種々の画素位置（Ｘ）とスキャン・ライン位置（Ｙ）の間で定義された横の辺９２、９４、９６および９８が含まれる。辺９６および９８は、スキャン・ライン上に形成される（したがってこれらと平行である）ので、長方形９０の実際のオブジェクト記述は、図９Ａに示されているように、横の辺９２および９４だけに基づくものとすることができる。これに関連して、辺９２は、画素位置（４０，３５）から始まり、ラスタ方向で画面の下側へ延びて、画素位置（４０，１０５）で終わる。同様に、辺９４は、画素位置（１６０，３５）から位置（１６０，１０５）まで延びる。長方形グラフィック・オブジェクト９０の水平部分は、単に辺９２から辺９４へラスタ化された形で走査することによって得ることができる。

【００３５】

しかし、青い三角形のオブジェクト８０は、３つのオブジェクト辺８２、８４および８６によって定義され、各辺は、三角形の頂点を定義するベクトルとみなされる。辺８２および８４は、画素位置（１００，２０）から始まり、それぞれ画素位置（１７０，９０）または（３０，９０）まで延びる。辺８６は、これら２つの画素位置の間で、従来のラスタ化された左から右への方向に延びる。この特定の例では、辺８６が、上で述べた辺９６および９８と同様に水平なので、辺８６が定義されることは必須ではない。というのは、辺８６が、辺８２および８４に関する終点をもつという特徴があるからである。辺８２および８４の記述に使用される始点および終点の画素位置のほかに、これらの辺のそれぞれに、この場合ではそれぞれ＋１または－１の傾斜値が関連付けられる。

【００３６】

図１０に、スキャン・ライン３５で始まる長方形９０がレンダリングされる様子と、辺８２および８４がスキャン・ライン３５とどのように交差するかを示す。図１０から、画像７８のラスタ化には、高い優先順位レベルを有するオブジェクトが、低い優先順位レベルを有するオブジェクトの「上」にレンダリングされる形で２つのオブジェクト９０および８０が描画される必要のあることがわかる。これを図１１に示す。図１１は、画像７８のレンダリングに使用される辺リスト・レコードを示す図である。図１１のレコードには、オブジェクトごとに１つずつの、２つの項目が含まれる。これらの項目は、それぞれのオブジェクトのラスタ・レンダリング順での始点に対応するスキャン・ライン値で配置される。図１１から、辺レコードのそれぞれが、オブジェクトの関連する優先順位レベルと、記述される辺の性質に関する詳細（たとえば色、傾斜など）を有することがわかる。

【００３７】

再び図２に戻って説明する。レンダリングされるグラフィック・オブジェクトの記述に必要なデータを識別したので、グラフィック・システム１は、表示リスト生成ステップ１２を実行する。

【００３８】

表示リスト生成１２は、取り付けられたＲＯＭ６およびＲＡＭ３を有するホスト・プロセッサ２上で実行されるソフトウェア・モジュールとして実施されることが好ましい。表示リスト生成１２では、周知のグラフィック記述言語、グラフィック・ライブラリ呼出しまたは他のアプリケーション固有フォーマットのうちの１つまたは複数で表現されたオブジェクト・グラフィック記述を表示リストに変換する。表示リストは、通常は、表示リスト・ストア１３に書き込まれる。表示リスト・ストア１３は、一般にＲＡＭ４内で形成されるが、その代わりにレンダリング・ストア３０内で形成することもできる。図３からわかるように、表示リスト・ストア１３には、複数の構成要素を含めることができ、その１つは命令ストリーム１４であり、もう１つは辺情報１５であり、ラスタ画像画素データ１６を含めることができる。

【００３９】

命令ストリーム１４には、特定の画像内で所望される特定のグラフィック・オブジェクトをレンダリングするために画素シーケンシャル・レンダリング装置２０によって読み取られる、命令として解釈可能なコードが含まれる。図８に示された画像の例では、命令ストリーム１４が、下記の形になり得る。

（１）スキャン・ライン２０までレンダリングする（何もしない）
（２）スキャン・ライン２０で２つの青い辺８２および８４を追加する
（３）スキャン・ライン３５までレンダリングする
（４）スキャン・ライン３５で２つの赤い辺９２および９４を追加する
（５）最後までレンダリングする。

【００４０】

同様に、図８の例によれば、辺情報１５には、下記が含まれることになる。
・辺８４は画素位置１００から始まり、辺８２は画素位置１００から始まる；
・辺９２は画素位置４０から始まり、辺９４は画素位置１６０から始まる；
・辺８４は７０スキャン・ラインだけ延び、辺８２は７０スキャン・ラインだけ延びる；
・辺８４は傾斜＝－１を有し、辺８４は傾斜＝＋１を有する；
・辺９２は傾斜＝０を有し、辺９４は傾斜＝０を有する；
・辺９２および９４は７０スキャン・ラインだけ延びる。

【００４１】

上の命令ストリーム１４および辺情報１５の例とそれぞれが表現される形から、図８の画像７８では、画素位置（Ｘ）とスキャン・ライン値（Ｙ）によって、画像７８がレンダリングされる単一の出力空間が定義されることが認められる。しかし、本開示の原理を使用して、他の出力空間構成を実現することもできる。

【００４２】

図８には、ラスタ画像画素データが含まれず、したがって、表示リスト１３の記憶部分１６には何も記憶する必要がない。この特徴については後で説明する。

【００４３】

表示リスト・ストア１３は、画素シーケンシャル・レンダリング装置２０によって読み取られる。画素シーケンシャル・レンダリング装置２０は、通常は集積回路として実施される。画素シーケンシャル・レンダリング装置２０は、表示リストをラスタ画素のストリームに変換し、このストリームは、たとえばプリンタ、ディスプレイまたはメモリ・ストアなどの別の装置に転送することができる。

【００４４】

好ましい実施形態では、集積回路として画素シーケンシャル・レンダリング装置２０を説明するが、これは、ホスト・プロセッサ２などの汎用処理ユニット上で実行可能な同等のソフトウェア・モジュールとして実施することができる。このソフトウェア・モジュールは、ディスク装置またはコンピュータ・ネットワークなどのコンピュータ可読媒体を介してユーザに配布することのできるコンピュータ・プログラム製品の一部を形成することができる。

【００４５】

図３は、画素シーケンシャル・レンダリング装置２０、表示リスト・ストア１３および一時的レンダリング・ストア３０の構成を示す図である。画素シーケンシャル・レンダリング装置２０の処理ステージ２２には、命令実行機構３００、辺処理モジュール４００、優先順位決定モジュール５００、塗潰し色決定モジュール６００、画素合成モジュール７００および画素出力モジュール８００が含まれる。処理動作では、一時的ストア３０を使用するが、これは、上で述べたように表示リスト・ストア１３と同一の装置（たとえば磁気ディスクまたは半導体ＲＡＭ）を共用するか、速度最適化のために個々の格納部（ストア）として実施することができる。辺処理モジュール４００は、辺レコード・ストア３２を使用して、スキャン・ラインからスキャン・ラインへ順方向に運ばれる辺情報を保持する。優先順位決定モジュール５００は、優先順位特性および状況テーブル３４を使用して、各優先順位に関する情報と、スキャン・ラインがレンダリングされている間の辺交差に関する各優先順位の現在の状態を保持する。塗潰し色決定モジュール６００は、塗潰しデータ・テーブル３６を使用して、特定の位置で特定の優先順位の塗潰し色を決定するのに必要な情報を保持する。画素合成モジュール７００は、画素合成スタック３８を使用して、出力画素の値を決定するために複数の優先順位からの色が必要になる出力画素の、決定中

の中間結果を保持する。
【００４６】
表示リスト・ストア１３および上で詳細を示した他のストア３２ないし３８は、ＲＡＭ内で実施するか、他のデータ記憶技術で実施することができる。
【００４７】
図３の実施形態に示された処理ステップは、処理パイプライン２２の形をとる。この場合、パイプラインのモジュールは、以下で説明する形でそれらの間でメッセージを受け渡しながら、画像データの異なる部分に対して並列に同時に実行することができる。もう１つの実施形態では、以下で説明するメッセージのそれぞれが、下流モジュールへの制御の同期転送の形をとることができ、上流処理は、下流モジュールがそのメッセージの処理を完了するまで中断される。
【００４８】
命令実行機構３００は、命令ストリーム１４から命令を読み取り、処理し、その命令を、出力３９８を介してパイプライン２２内の他のモジュール４００、５００、６００および７００に転送されるメッセージにフォーマットする。好ましい実施形態では、命令ストリーム１４に、以下の命令を含めることができる。
【００４９】
ＬＯＡＤ＿ＰＲＩＯＲＩＴＹ＿ＰＲＯＰＥＲＴＩＥＳ：　この命令は、優先順位特性および状況テーブル３４にロードされるデータと、そのデータがロードされるテーブル内のアドレスに関連する。命令実行機構３００は、この命令に出会った時に、優先順位特性および状況テーブル３４の指定された位置でのデータの記憶のためのメッセージを発行する。これは、このデータを含むメッセージをフォーマットし、処理パイプライン２２を介して、ストア動作を実行する優先順位決定モジュール５００に渡すことによって達成できる。
【００５０】
ＬＯＡＤ＿ＦＩＬＬ＿ＤＡＴＡ：　この命令は、塗潰しデータ・テーブル３６にロードされるデータと、そのデータがロードされるテーブル内のアドレスに関連する。命令実行機構３００は、この命令に出会った時に、塗潰しデータ・テーブル３６の指定されたアドレスでのデータの記憶のためのメッセージを発行する。これは、このデータを含むメッセージをフォーマットし、処理パイプライン２２を介して、ストア動作を実行する塗潰しデータ決定モジュールに渡すことによって達成できる。
【００５１】
ＬＯＡＤ＿ＮＥＷ＿ＥＤＧＥＳ＿ＡＮＤ＿ＲＥＮＤＥＲ：　この命令は、次のスキャン・ラインをレンダリングする時にレンダリング処理に導入される新しい辺１５の表示リスト・ストア１３内のアドレスに関連する。命令実行機構３００は、この命令に出会った時に、このデータを含むメッセージをフォーマットし、辺処理モジュール４００に渡す。辺処理モジュール４００は、新しい辺のアドレスを辺レコード・ストア３２に記憶する。指定されたアドレスにある辺は、次のスキャン・ラインをレンダリングする前に、最初のスキャン・ライン交差座標に基づいてソートされる。一実施形態では、辺は、表示リスト生成処理１２によってソートされる。別の実施形態では、辺は、画素シーケンシャル・レンダリング装置２０によってソートされる。
【００５２】
ＳＥＴ＿ＳＣＡＮ　ＬＩＮＥ＿ＬＥＮＧＴＨ：　この命令は、レンダリングされるスキャン・ラインのそれぞれで作られる画素数に関連する。命令実行機構３００は、この命令に出会った時に、この値を辺処理モジュール４００および画素合成モジュール７００に渡す。
【００５３】
ＳＥＴ＿ＯＰＡＣＩＴＹ＿ＭＯＤＥ：　この命令は、画素合成演算で不透明度チャネル（当技術分野ではアルファ・チャネルとも称する）を使用するかどうかを示すフラグに関連する。命令実行機構３００は、この命令に出会った時に、このフラグの値を画素合成モジュール７００に渡す。

【００５４】
命令実行機構３００は、通常は、命令をマッピングし、パイプライン動作に復号して、さまざまなモジュールに渡す、マイクロコードステートマシンによって形成される。或いは、その代わりに、対応するソフトウェア処理を使用することもできる。
【００５５】
スキャン・ラインのレンダリング動作中の辺処理モジュール４００の動作を、図４を参照して以下に説明する。スキャン・ラインのレンダリングのための初期条件は、以下の３つの辺レコードのストが使用可能であることである。これら３つのリストのいずれかまたはすべては空でもよい。これらのリストは、辺情報１５から取得されＬＯＡＤ＿ＮＥＷ＿ＥＤＧＥＳ＿ＡＮＤ＿ＲＥＮＤＥＲ命令によってセットされる新しい辺を含む新辺リスト４０２、前のスキャン・ラインから順方向に運ばれた辺レコードを含む主辺リスト４０４および、やはり前のスキャン・ラインから順方向に運ばれた辺レコードを含むスピル辺リスト４０６である。各辺レコードには、下記が含まれ得る。
【００５６】
（ｉ）現在のスキャン・ライン交差座標（本明細書ではＸ座標と称する）
（ｉｉ）この辺の現在の線分が続くスキャン・ライン数のカウント（本明細書ではＮＹと称する。いくつかの実施形態では、これをＹ限界と表現する場合がある）
（ｉｉｉ）各スキャン・ラインの後でこの辺レコードのＸ座標に加算される値（本明細書ではＤＸと称する）
（ｉｖ）各スキャン・ラインの後でこの辺レコードのＤＸに加算される値（本明細書ではＤＤＸと称する）
（ｖ）１つまたは複数の優先順位番号（Ｐ）
（ｖｉ）辺がスキャンラインと上向き（＋）に交差するか下向き（－）に交差するかを示す方向（ＤＩＲ）フラグ
（ｖｉｉ）リスト内の次の辺線分のアドレス（ＡＤＤ）。
【００５７】
このようなフォーマットは、ベクトル、直交配置された辺および二次曲線に適合する。これ以外のパラメータ、たとえばＤＤＤＸの追加によって、このような配置が三次曲線に適合できるようになる。三次ベジェ・スプラインなど、いくつかの応用分野では、６階多項式（すなわちＤＤＤＤＤＤＸまで）が必要になる場合がある。
【００５８】
図８の辺８４および９４の例では、スキャン・ライン２０での対応する辺レコードが、以下の表１に示されたものになる。
【００５９】
【表２】

TABLE 1

Edge 84	Edge 92
X = 100	X = 40
NY = 70	NY = 70
DX = 1	DX = 0
DDX = 0	DDX = 0
P = 1	P = 0
DIR = (-)	DIR = (+)
ADD = (irrelevant in this example)	ADD = (irrelevant in this example)

【００６０】
この説明では、レンダリング処理によって生成されつつあるスキャン・ラインに沿って画素から画素へステップする座標を、Ｘ座標と称し、スキャン・ラインからスキャン・ラインへとステップする座標を、Ｙ座標と称する。各辺リストには、メモリ内で連続的に配置された０個以上のレコードが含まれることが好ましい。ポインタ・チェインの使用を含む他の記憶配置も可能である。３つのリスト４０２、４０４および４０６のそれぞれのレコードは、スキャン・ライン交差（Ｘ）座標の順で配置される。これは、通常は、当初は辺入力モジュール４０８によって管理されるソート処理によって得られ、辺入力モジュール４０８は、辺情報を含むメッセージを命令実行機構３００から受け取る。各スキャン・ライン交差座標の整数部分だけを有意とみなすためにソートを緩和することが可能である。また、各スキャン・ライン交差座標を、現在のレンダリング処理によって作られる最小および最大のＸ座標にクランプされるとみなすことによってさらにソートを緩和することが可能である。適当な場合には、辺入力モジュール４０８は、メッセージを、出力４９８を介してパイプライン２２の下流のモジュール５００、６００および７００に受け渡す。
【００６１】
辺入力モジュール４０８は、３つのリスト４０２、４０４および４０６のそれぞれへの参照を維持し、これらのリストから辺データを受け取る。これらの参照のそれぞれは、スキャン・ラインの処理の開始時に、各リスト内の最初の辺を参照するように初期設定される。その後、辺入力モジュール４０８は、選択されるレコードが３つの参照されるレコードからの最小のＸ座標を有するものになるように、３つの参照された辺レコードのうちの１つから辺レコードを選択する。複数のＸレコードが等しい場合には、それぞれが任意の順序で処理され、対応する辺交差が、下記の形で出力される。そのレコードの選択に使用された参照は、その後、そのリストの次のレコードに進められる。選択されたばかりの辺は、メッセージにフォーマットされ、辺更新モジュール４１０に送られる。また、辺のいくつかのフィールド、具体的には現在のＸ、優先順位番号および方向フラグが、メッセージにフォーマットされ、このメッセージは、辺処理モジュール４００の出力４９８として優先順位決定モジュール５００に転送される。なお、本明細書に記載されたものより多数または少数のリストを使用する実施形態も可能である。
【００６２】
辺を受け取った時に、辺更新モジュール４１０は、現在の線分が続くスキャン・ライン数のカウントをデクリメントする。そのカウントが０に達した場合には、次の線分アドレスによって示されるアドレスから新しい線分を読み取る。線分では、下記が指定される。
（ｉ）線分を読み取った直後に現在のＸ座標に加算される値
（ｉｉ）その辺の新しいＤＸ値

（ｉｉｉ）その辺の新しいＤＤＸ値
（ｉｖ）新しい線分が続くスキャン・ライン数の新しいカウント。
【００６３】
示されたアドレスに使用可能な次の線分がない場合には、その辺に対してそれ以上の処理は実行されない。そうでない場合には、辺更新モジュール４１０は、その辺の次のスキャン・ラインのＸ座標を計算する。これには、通常は、現在のＸ座標をとり、これにＤＸ値を加算することが用いられる。ＤＸは、処理される辺の種類に応じて、必要であればＤＤＸ値を加算される場合がある。その後、辺は、複数の辺レコードの配列である辺プール４１２内の使用可能な空きスロットに書き込まれる。空きスロットがない場合には、辺更新モジュール４１０は、スロットが使用可能になるのを待つ。辺レコードが辺プール４１２に書き込まれたならば、辺更新モジュール４１０は、新しい辺が辺プール４１２に追加されたことを、信号線４１６を介して辺出力モジュール４１４に知らせる。
【００６４】
スキャン・ラインのレンダリングのための初期条件として、辺出力モジュール４１４は、図４には図示されていないが、辺レコード３２内のリスト４０４および４０６に関連する次主辺リスト４２０および次スピル辺リスト４２２のそれぞれへの参照を有する。これらの参照のそれぞれは、当初は空のリスト４２０および４２２が構築される位置に初期設定される。辺が辺プール４１２に追加されたことを示す信号４１６を受け取った時に、辺出力モジュール４１４は、追加された辺が、次主辺リスト４２０に最後に書き込まれた辺（があれば）より小さいＸ座標を有するかどうかを判定する。これが真である場合には、順序付けの基準を侵害せずにその辺を主辺リスト４０４の末尾に追加することができないので、「スピル」が発生したという。スピルが発生した時には、その辺は、好ましくはソートされた次スピル辺リスト４２２を維持する形で、次スピル辺リスト４２２に挿入される。たとえば、これは、ソフトウェア・ソート・ルーチンを使用して達成することができる。いくつかの実施形態では、スピルが、極端に大きいＸ座標など、他の条件によってトリガされる場合がある。
【００６５】
辺プール４１２に追加された辺が、次主辺リスト４２０に最後に書き込まれた辺（があれば）以上のＸ座標を有し、辺プール４１２に使用可能な空きスロットがない場合には、辺出力モジュール４１４は、辺プール４１２から、最小のＸ座標を有する辺を選択し、その辺を次主辺リスト４２０の末尾に追加し、この処理で次主辺リスト４２０を延長する。辺プール４１２内の、その辺によって占められていたスロットは、空きとしてマークされる。
【００６６】
辺入力モジュール４０８は、これら３つの入力リスト４０２、４０４および４０６のすべてからすべての辺を読み取り、転送した後に、スキャン・ラインの終りに達したことを示すメッセージをフォーマットし、そのメッセージを、優先順位決定モジュール５００と辺更新モジュール４１０の両方に送る。そのメッセージを受け取った時に、辺更新モジュール４１０は、それが現在実行しているすべての処理が完了するのを待ち、その後、このメッセージを辺出力モジュール４１４に転送する。そのメッセージを受け取った時に、辺出力モジュール４１４は、辺プール４１２からの残りの辺レコードのすべてを、Ｘ順で次の主辺リスト４０４に書き込む。その後、次主辺リスト４２０および主辺リスト４０４への参照を、辺入力モジュール４０８と辺出力モジュール４１４の間で交換し、同様の交換を、次スピル辺リスト４２２とスピル辺リスト４０６に関しても実行する。この形で、次のスキャン・ラインのための初期条件が確立される。
【００６７】
辺レコードの挿入時に次スピル辺リスト４２２をソートするのではなく、そのような辺レコードを、単にリスト４２２の末尾に追加することができ、スキャン・ラインの末尾で、現在のスピル・リスト４０６との交換の前にソートされるリスト４２２は、次のスキャン・ラインの辺ラスタ化でアクティブになる。エッジをソートする他の方法では、より少数

またはより多数のリストを使用することができ、また、異なるソート・アルゴリズムを使用することができる。

【００６８】

上記から、辺交差メッセージが、スキャン・ライン順および画素順（すなわち、まずＹでソートされ、次にＸでソートされる）で優先順位決定モジュール５００に送られること、および各辺交差メッセージに、それに適用される優先順位のラベルが付けられることを推論することができる。

【００６９】

図１２Ａは、辺の線分が受け取られる時に辺処理モジュール４００によって作成される可能性があるアクティブ辺レコード４１８の具体的な構造を示す図である。辺の最初の線分がステップ（直交）線分である場合には、辺のＸ値を、最初の線分の「Ｘステップ」と称する変数に加算して、アクティブ化された辺のＸ位置を得る。そうでない場合には、辺のＸ値を使用する。これは、新しい辺レコードの辺が、$X_{edge}+X_{step}$によってソートされなければならないことを意味する。したがって、最初の線分のX_{step}は、辺のソートを単純化するために０でなければならない。最初の線分のＹ値は、アクティブ辺レコード４１８のＮＹフィールドにロードされる。アクティブな辺のＤＸフィールドは、ベクトルまたは二次線分のＤＸフィールド識別子からコピーされ、ステップ線分の場合には０がセットされる。図１２Ａに示されたｕフラグは、線分が上向き（図１３Ａに関連する説明を参照されたい）である場合にセットされる。ｑフラグは、線分が二次線分の場合にセットされ、そうでない場合にはクリアされる。ｉフラグが設けられ、これは、線分が不可視である場合にセットされる。ｄフラグは、辺が、関連するクリッピング・レベルなしに直接クリッピング・オブジェクトして使用され、閉じた曲線に適用可能である時にセットされる。線分の実際の優先順位レベルまたはレベル・アドレスは、新しい辺レコードの対応するフィールドからアクティブ辺レコード４１８のレベル（ＡＤＤＲ）フィールド（Level(Addr)）にコピーされる。アクティブ辺レコード４１８の線分アドレス／ＤＤＸフィールド（Seg.Addr(DDX)）は、線分リスト内の次の線分のアドレスであるか、線分が二次線分の場合にはセグメントのＤＤＸ値からコピーされるかのいずれかである。線分アドレスは、辺レコードを終了させるのに使用される。その結果、好ましい実施形態では、二次曲線のすべて（すなわち、ＤＤＸフィールドを使用する曲線）が、辺レコードの終端線分になる。

【００７０】

図１２Ａから、他のデータ構造も可能であり、たとえばより高次の多項式の実装が使用される場合にはそれが必要であることを諒解されたい。さらに、線分アドレスおよびＤＤＸフィールドは、異なるフィールドに分離することができ、代替実施態様に合わせて追加のフラグを設けることができる。

【００７１】

図１２Ｂは、辺処理モジュール４００で使用される、上で説明した好ましい実施形態の辺レコードの配置を示す図である。辺プール４１２は、新アクティブ辺レコード４２８、現アクティブ辺レコード４３０およびスピル・アクティブ辺レコード４３２によって補足される。図１２Ｂからわかるように、レコード４０２、４０４、４０６、４２０および４２２は、ある時点でレンダリングされる辺の数に応じて、サイズを動的に変更することができる。各レコードには、新辺リスト４０２の場合にはＬＯＡＤ＿ＥＤＧＥＳ＿ＡＮＤ＿ＲＥＮＤＥＲ命令に組み込まれたＳＩＺＥ値によって決定される、限界値が含まれる。このような命令に出会った時には、ＳＩＺＥを検査し、０でない場合には、新しい辺レコードのアドレスをロードし、リスト４０２の限界サイズを決定する限界値を計算する。

【００７２】

好ましい実施形態では、辺レコードの処理のために配列とそれに関連するポインタを使用するが、たとえばリンク・リストなどの、他の実施態様を使用することができる。これらの他の実施態様は、ハードウェア・ベース、ソフトウェア・ベースまたはその組合せとすることができる。

【００７３】

図８に示された画像７８の具体的なレンダリングを、図１０に示されたスキャン・ライン３４、３５および３６に関連してこれから説明する。この例では、次のスキャン・ラインの新しいＸ座標の計算が、説明を明瞭にするために省略され、図１２Ｃないし図１２Ｉでは、出力辺交差が、辺プール４１２のレジスタ４２８、４３０および４３２の１つから導出される。

【００７４】

図１２Ｃは、スキャン・ライン３４（半透明の青い三角形８０の最上部）のレンダリングの終りでの、上で述べたリストの状態を示す図である。スキャン・ライン３４には、新しい辺がなく、したがって、リスト４０２が空であることに留意されたい。主辺リスト４０４および次主辺リスト４２０のそれぞれには、辺８２および８４だけが含まれる。リストのそれぞれには、対応するポインタ４３４、４３６および４４０が含まれ、これらは、スキャン・ライン３４の完了時に、対応するリスト内の次の空きレコードを指す。各リストには、対応するリストの末尾を指すために必要な、アスタリスク（＊）によって示される限界ポインタ４５０も含まれる。リンク・リストを使用する場合には、リンク・リストに、対応する機能を実行するヌル・ポインタ終端子が含まれるので、このような限界ポインタは不要になる。

【００７５】

上で述べたように、各スキャン・ラインの始めに、次主辺リスト４２０と主辺リスト４０４が交換され、新しい辺が、新辺リスト４０２に受け取られる。残りのリストはクリアされ、ポインタのそれぞれは、各リストの最初のメンバを指すようにセットされる。スキャン・ライン３５の始めでは、配置は図１２Ｄのようになる。図１２Ｄからわかるように、レコードには、図１０から辺９２、９４、８４および８２に対応することがわかる４つのアクティブな辺が含まれる。

【００７６】

図１２Ｅを参照すると、レンダリングが開始される時に、新辺レコード４０２の最初の線分が、アクティブ辺レコード４２８にロードされ、主辺リスト４０４およびスピル辺リスト４０６の最初のアクティブな辺レコードが、それぞれレコード４３０および４３２にコピーされる。この例では、スピル辺リスト４０６は空であり、したがって、ローディングは行われない。レコード４２８、４３０および４３２内の辺のＸ位置が比較され、辺交差が、最小のＸ位置を有する辺について発行される。この場合、発行される辺は、辺９２に対応する辺であり、この辺がその優先順位値と共に出力される。その後、ポインタ４３４、４３６および４３８が、リスト内の次のレコードを指すように更新される。

【００７７】

その後、辺交差が発行される辺が更新され（この場合、その位置にＤＸ＝０を加算することによって）、辺プール４１２にバッファリングされ、この辺プール４１２は、この例では３つの辺レコードを保持するサイズになる。発行された辺が現れたリスト（この場合ではリスト４０２）内の次の項目が、対応するレコード（この場合ではレコード４２８）にロードされる。これを図１２Ｆに示す。

【００７８】

さらに、図１２Ｆから明らかなとおり、レジスタ４２８、４３０および４３２の間の比較によって、もう一度最小のＸ値を有する辺が選択され、適切な次の辺交差（Ｘ＝８５、Ｐ＝２）として出力される。そして、同様に、選択され出力された辺は、更新され、辺プール４１２に追加され、適当なポインタのすべてがインクリメントされる。この場合、更新される値は、Ｘ←Ｘ＋ＤＸによって与えられ、ここでは、８４＝８５－１として与えられる。また、図からわかるように、新しい辺のポインタ４３４が、この場合では新辺リスト４０２の末尾に移動される。

【００７９】

図１２Ｇでは、最小の現行Ｘ値を有すると識別された次の辺が、やはり、レジスタ４３０から取得され、辺交差（Ｘ＝１１５、Ｐ＝２）として出力される。そして、辺の更新が発生し、図示のように、その値が辺プール４１２に追加される。この時、辺プール４１２は

満杯になり、ここから、最小のＸ値を有する辺が選択され、出力リスト４２０に発行され、対応する限界ポインタが、それ相応に移動される。

【００８０】

図１２Ｈからわかるように、次の最小の辺交差は、レジスタ４２８からのものであり、これが出力される（Ｘ＝１６０、Ｐ＝１）。やはり辺プール４１２が更新され、次に小さいＸ値が出力リスト４２０に発行される。

【００８１】

スキャン・ライン３５の終りに、図１２Ｉからわかるように、Ｘ値の小さい順で、辺プール４１２の内容が出力リスト４２０にフラッシュされる。図１２Ｊからわかるように、次主辺リスト４２０と主辺リスト４０４は、次のスキャン・ライン３６のレンダリングに備えて、ポインタを交換することによって交換される。交換の後に、図１２Ｊからわかるように、主辺リスト４０４の内容には、スキャン・ライン３６上の現在の辺のすべてが含まれ、これらはＸ位置の順で配置され、これによって、高速のレンダリングを容易にする便利なアクセスが可能になる。

【００８２】

通常、新しい辺は、Ｘ位置の昇順で辺処理モジュール４００によって受け取られる。新しい辺が現れる時には、その位置が更新され（次にレンダリングされるスキャン・ラインのために計算され）、これによって、以下の処置が決定される。

【００８３】

（ａ）更新された位置が信号線４９８に出力された最後のＸ位置より小さい場合には、新しい辺は、主スピル・リスト４０６へのソートされる挿入であり、対応する限界レジスタが更新される。

【００８４】

（ｂ）そうでない場合に、空間があるならば、その辺は辺プール４１２内で保存される。

【００８５】

前述から明白なとおり、辺プール４１２は、ラスタ化画像の次のスキャン・ラインのレンダリングに備えた順序付きの形でのリストの更新を助ける。さらに、辺プール４１２のサイズは、多数の順序付けられていない辺に適合するために変更することができる。しかし、実際には、辺プール４１２が、一般にグラフィック処理システムの処理速度および使用可能なメモリに依存する、実用的な限界を有することは明白である。制限的な意味では、辺プール４１２を省略することができるが、これは、通常は、更新された辺を、次出力辺リスト４２０へのソートされた挿入にすることを必要とする。しかし、好ましい実施形態では、上で述べたスピル・リストの使用を介する通常の出来事として、この状況が回避される。スピル・リストを設けることによって、好ましい実施形態を、実用的なサイズの辺プールを用いて実施でき、なおかつ、ソフトウェア集中的なソート手順に頼らずに比較的複雑な辺交差を処理することができる。辺プールとスピル・リストが辺交差の複雑さに適合するのに不十分になる場合では、これは稀な場合であるが、ソート法を使用することができる。

【００８６】

スピル・リスト手順が使用される場合の例を、図１４Ａに示す。図１４Ａでは、３つの任意の辺６０、６１および６３が、スキャン・ラインＡおよびＢの間の相対位置で任意の辺６２と交差する。さらに、スキャン・ラインＡおよびＢのそれぞれについて実際に表示される画素位置６４が、スパン画素位置ＣないしＪとして図示されている。辺プール４１２が３つの辺レコードを保存するサイズである、上で説明した例では、このような配置だけでは、図１４Ａに示された隣接するスキャン・ラインの間で発生する３つの辺の交差に適合するのに不十分であることは明白である。

【００８７】

図１４Ｂに、スキャン・ライン上の辺６０、６１および６３をレンダリングした後の辺レコードの状態を示す。辺交差Ｈは、最も最近に発行された辺交差であり、辺プール４１２は、次のスキャン・ラインであるスキャン・ラインＢのための、それぞれ辺６０、６１お

よび６３の更新されたＸ値Ｅ、ＧおよびＩで満杯になっている。辺６２は、現アクティブ辺レコード４３０にロードされ、辺プール４１２が満杯なので、辺６０に対応する最小のＸ値が、出力辺リスト４２０に出力される。

【００８８】

図１４Ｃでは、次の辺交差が発行され（辺６２のＸ＝Ｊ）、対応する更新された値、この場合はスキャン・ラインＢのＸ＝Ｃが決定される。新たに更新された値Ｘ＝Ｃは、出力リスト４２０からコピーされた最も最近の値Ｘ＝Ｅより小さいので、現在の辺レコードとそれに対応する新たに更新された値が、出力スピル・リスト４２２に直接に転送される。

【００８９】

図１４Ｄに、スキャン・ラインＢの開始時の辺レコードの状態を示す。この図では、主リストおよび出力リストと、それらに対応するスピル構成要素が交換されていることがわかる。最初に発行される辺を決定するために、辺６０を現アクティブ辺レジスタ４３０にロードし、辺６２を、スピル・アクティブ辺レジスタ４３２にロードする。Ｘ値が比較され、最小のＸ値（Ｘ＝Ｃ）を有する辺６２が発行され、更新され、辺プール４１２にロードされる。

【００９０】

辺の発行と更新は、主辺リスト４０４内の残りの辺について継続され、スキャン・ラインの終りに、辺プール４１２がフラッシュされて、図１４Ｅに示された状況になる。図１４Ｅからは、辺６０ないし６３のそれぞれが、次のスキャン・ラインでのレンダリングのために適当に順序付けられ、スキャン・ラインＢ上で正しく発行され、レンダリングされたことがわかる。

【００９１】

上記から明らかになるように、スピル・リストは、複雑な辺交差状況の存在のもとで、辺ラスタ化順序の維持をもたらす。さらに、このリストがサイズにおいて動的に変更可能であることによって、辺交差の数と複雑さの大きい変化を、例外的に複雑な辺交差以外のすべての辺交差でソート手順に頼る必要なしに処理できる。

【００９２】

好ましい実施形態では、辺プール４１２は、８つの辺レコードを保存するサイズにされ、リスト４０４および４２０のサイズは、それらに関連するスピル・リスト４０６および４２２と一緒に、動的に変更可能であり、これによって、複雑な辺交差要件を有する大きい画像を処理するために十分な範囲が提供される。

【００９３】

優先順位決定モジュール５００の動作を、図５を参照して以下に説明する。辺処理モジュール４００からの着信メッセージ４９８は、優先順位データ設定メッセージ、塗潰しデータ設定メッセージ、辺交差メッセージおよびスキャン・ラインの終りメッセージが含まれる可能性があるが、優先順位更新モジュール５０６によって読み取られる前に、まず先入れ先出し（ＦＩＦＯ）バッファ５１８を通過する。ＦＩＦＯ５１８は、辺処理モジュール４００の動作と優先順位決定モジュール５００の動作を分離するように働く。優先順位状態テーブル５０２は、上で述べたテーブル３４の一部を含むが、各オブジェクトの優先順位に関する情報を保持するのに使用される。ＦＩＦＯ５１８は、エッジ交差のスキャン・ライン全体の辺処理モジュール４００からの受取りと優先順位状態テーブル５０２への転送を単一の処置で可能にするサイズであることが好ましい。これによって、優先順位決定モジュール５００が、同一の画素（Ｘ）位置での複数の辺交差を効率的に処理できるようになる。優先順位状態テーブル５０２の各レコードには、以下が記録される。

【００９４】

（ｉ）この優先順位が、奇数－偶数塗潰し規則または非ゼロ・ワインディング塗潰し規則の適用によって決定される内部／外部状態を有するかどうかを示す塗潰し規則フラグ

（ｉｉ）この優先順位をもたらす辺が交差するたびに塗潰し規則によって示される形で変更される塗潰しカウント

（ｉｉｉ）この優先順位がクリッピングと塗潰しのどちらに使用されるかを示すクリッパ

・フラグ
（ｉｖ）クリッパ・フラグがセットされている辺について、クリッピング・タイプが「クリップ・イン」と「クリップ・アウト」のどちらであるかを記録するクリップ・タイプ・フラグ
（ｖ）この優先順位をもたらすクリップ・イン・タイプのクリップ領域に入る時にデクリメントされ、出る時にインクリメントされ、この優先順位をもたらすクリップ・アウト・タイプのクリップ領域に入る時にインクリメントされ、出る時にデクリメントされる、クリップ・カウント
（ｖｉ）「ニード・ビロウ」フラグと称する、この優先順位がそれ未満のレベルを最初に計算することを必要とするかどうかを記録するフラグ。

【００９５】
クリッピング・オブジェクトは、当技術分野で既知であり、特定の新しいオブジェクトを表示するように働くのではなく、画像内の別のオブジェクトの形状を変更するように働く。クリッピング・オブジェクトは、さまざまな視覚的効果を達成するために、オンにし、オフにすることもできる。たとえば、図８のオブジェクト８０は、オブジェクト９０に対して、オブジェクト９０のうちでクリッピング・オブジェクト８０の下にある部分を除去するように働くクリッピング・オブジェクトとして構成することができる。これは、クリッピング境界内における、オブジェクト９０の下にある、クリッピングされていなければオブジェクト９０の不透明さによって隠されるオブジェクトまたは画像を見せるという効果を有する。

【００９６】
図１３Ａおよび図１３Ｂに、奇数－偶数規則と非ゼロ・ワインディング規則の応用例を示す。非ゼロ・ワインディング規則の目的のために、図１３Ａに、オブジェクト７０の辺７１および７２が、辺が下向きと上向きのどちらであるかに従って概念上の向きをどのように割り振られるかを示す。閉じた境界を形成するために、辺は、境界に沿って始点と終点が重なる形でリンクする。塗潰し規則（後で適用され、説明される）の目的のために辺に与えられる向きは、線分が定義される順序と独立である。辺の線分は、レンダリングの方向に対応する、追跡される順序で定義される。

【００９７】
図１３Ｂに、２つの下向きの辺７３および７６と、３つの上向きの辺７４、７５および７７を有する単一のオブジェクト（五線星形）を示す。奇数－偶数規則は、各辺が対象のスキャン・ラインと交差するたびにブール値を単純にトグルし、したがって、効果的にオブジェクト色をオンにするかオフにすることによって動作する。非ゼロ・ワインディング規則では、交差する辺の向きに応じて塗潰しカウント値をインクリメントまたはデクリメントする。図１３Ｂでは、このスキャン・ラインで出会う最初の２つの辺７３および７６が下向きであり、したがって、これらの辺の横断によって、塗潰しカウントがインクリメントされ、それぞれ＋１および＋２になる。このスキャン・ラインが出会う次の２つの辺７４および７７は、上向きであり、したがって、塗潰しカウントがそれぞれ＋１および０にデクリメントされる。

【００９８】
いくつかの実施形態では、この情報の一部が、表示リスト１３および前に説明したさまざまな辺リストの辺に関連し、辺交差メッセージの一部として優先順位決定モジュール５００に転送される。具体的に言うと、塗潰し規則フラグ、クリッパ・フラグ、クリップ・タイプ・フラグおよびニード・ビロウ・フラグを、この形で処理することができる。

【００９９】
図５に戻って、優先順位更新モジュール５０６は、それが処理を完了したものまでのスキャン・ライン交差座標を記録するカウンタ５２４を維持する。これを、優先順位更新モジュール５０６の現行Ｘと称する。スキャン・ラインの開始時の初期値は０である。

【０１００】
ＦＩＦＯ５１８の先頭で受け取った辺交差メッセージを検査する際に、優先順位更新モジ

ュール５０６は、辺交差メッセージのＸ交差値とその現行Ｘを比較する。辺交差メッセージのＸ交差値が、優先順位更新モジュール５０６の現行Ｘ以下の場合には、優先順位更新モジュール５０６は、その辺交差メッセージを処理する。辺交差メッセージの処理は、２つの形態になり、「通常辺処理」（下で説明する）は、辺交差メッセージの最初の優先順位によって示される優先順位状態テーブル５０２内のレコードが、クリップ優先順位でないことを示すクリッパ・フラグを有する時に使用され、そうでない場合には、「クリップ辺処理」（下で説明する）が実行される。

【０１０１】

優先順位がある画素でアクティブになるのは、その画素が、その優先順位の塗潰し規則に従って優先順位に適用される境界辺の内部にあり、その優先順位のクリップ・カウントが０の場合である。優先順位は、それが最も上のアクティブな優先順位である場合、または、それより上位のアクティブな優先順位のすべてが、対応するニード・ビロウ・フラグをセットしている場合に、公開される。この形で、画素値を、公開された優先順位の塗潰しデータだけを使用して生成することができる。

【０１０２】

ある優先順位のニード・ビロウ・フラグは、表示リストの情報内で確立され、そのフラグがセットされていなければ、問題の優先順位の下のアクティブな優先順位のすべてが、レンダリング中の画素値に寄与しないことを画素生成システムに知らせるのに使用される。このフラグは、最終的な画素値に全く寄与しないはずの余分な合成演算を防ぐのに適当な場合にクリアされる。

【０１０３】

「通常辺処理」には、辺交差メッセージ内の優先順位のそれぞれについて、その優先順位によって示される優先順位状態テーブル・レコードのフィールドに関して、以下のステップが含まれる。

【０１０４】

（ｉ）現在の優先順位の現在の塗潰しカウントを記録し、

（ｉｉ）現在の優先順位の塗潰し規則が、

（ａ）奇数－偶数である場合には、塗潰しカウントが現在０以外であれば塗潰しカウントに０をセットし、そうでない場合には０以外の値をセットし、

（ｂ）非ゼロ・ワインディングである場合には、塗潰しカウントをインクリメントまたはデクリメント（辺の方向フラグに応じて）し、

（ｉｉｉ）新しい塗潰しカウントと記録された塗潰しカウントを比較し、一方が０で他方が０以外の場合には、現在の優先順位に対して「アクティブ・フラグ更新」（下で説明する）動作を実行する。

【０１０５】

いくつかの実施形態では、辺交差メッセージのそれぞれに複数の優先順位を置くのではなく、優先順位ごとに別々の辺交差メッセージを使用することができる。

【０１０６】

アクティブ・フラグ更新動作には、まず現在の優先順位のための新しいアクティブ・フラグを確立することが含まれる。アクティブ・フラグは、優先順位状態テーブル５０２のその優先順位の塗潰しカウントが０以外であり、その優先順位のクリップ・カウントが０の場合に０以外になり、そうでない場合には、アクティブ・フラグは０になる。アクティブ・フラグ更新動作の第２ステップは、アクティブ・フラグ配列５０８内の現在の優先順位によって示される位置に、決定されたアクティブ・フラグを格納し、現在の優先順位の優先順位状態テーブルのニード・ビロウ・フラグが０の場合には、不透明アクティブ・フラグ配列５１０の現在の優先順位によって示される位置にもアクティブ・フラグを格納することである。

【０１０７】

「クリップ辺処理」には、辺交差メッセージの最初の優先順位によって示される優先順位状態テーブル・レコードのフィールドに関して、以下のステップが含まれる。

【０１０８】
（ｉ）現在の優先順位の現在の塗潰しカウントを記録し、
（ｉｉ）現在の優先順位の塗潰し規則が、
（ａ）奇数－偶数である場合には、塗潰しカウントが現在０以外であれば塗潰しカウントに０をセットし、そうでない場合には０以外の値をセットし、
（ｂ）非ゼロ・ワインディングである場合には、塗潰しカウントをインクリメントまたはデクリメント（辺の方向フラグに応じて）し、
（ｉｉｉ）新しい塗潰しカウントと記録された塗潰しカウントを比較し、
（ａ）新しい塗潰しカウントが０であり、記録された塗潰しカウントが０である場合、または、新しい塗潰しカウントが０以外であり、記録された塗潰しカウントが０以外である場合には、０、
（ｂ）現在の優先順位のクリップ・タイプ・フラグがクリップ・アウトであり、記録された塗潰しカウントが０であり、新しい塗潰しカウントが０以外である場合、または、現在の優先順位のクリップ・タイプ・フラグがクリップ・インであり、記録された塗潰しカウントが０以外であり、新しい塗潰しカウントが０である場合には、＋１、
（ｃ）それ以外の場合には、－１
のクリップ・デルタ値を決定し、
（ｉｖ）辺交差メッセージの最初の優先順位の後のすべての後続優先順位について、その後続優先順位によって示される優先順位状態テーブル内のレコードのクリップ・カウントに決定されたクリップ・デルタ値を加算し、その処理で、クリップ・カウントが、０以外から０になった場合または０から０以外になった場合に、その後続優先順位に対して上で説明したアクティブ・フラグ更新動作を実行する。各クリップ・カウントの初期値は、前に説明したＬＯＡＤ＿ＬＥＶＥＬ＿ＰＲＯＰＥＲＴＩＥＳ命令によってセットされることに留意されたい。クリップ・カウントは、通常は、各優先順位に影響するクリップ・イン優先順位の数に初期設定される。
【０１０９】
いくつかの実施形態では、優先順位がクリップに関連付けられず、その代わりに、辺交差メッセージで与えられるすべての優先順位のクリップ・カウントが直接にインクリメントまたはデクリメントされる。この技法は、たとえば、クリップ形状が単純であり、複雑な塗潰し規則の適用が不要な時に使用することができる。この特定の応用例では、辺によって制御されるレベルのクリップ・カウントは、上向きの辺の場合にインクリメントされ、下向きの辺の場合にデクリメントされる。反時計回りに記述された単純な閉じた曲線は、クリップ・インとして働き、時計回りに記述された単純な閉じた曲線は、クリップ・アウトとして働く。
【０１１０】
辺交差メッセージのＸ交差値が、優先順位更新モジュール５０６の現行Ｘを超える時には、優先順位更新モジュール５０６は、生成する画素数のカウントすなわち、辺交差メッセージのＸ交差値と現行Ｘの間の差を形成し、このカウントを優先順位生成メッセージにフォーマットし、接続５２０を介して優先順位生成モジュール５１６に送る。その後、優先順位更新モジュール５０６は、所与の個数の画素の処理が完了したことを示す優先順位生成モジュール５１６からの信号５２２を待つ。信号５２２を受け取った時に、優先順位更新モジュール５０６は、その現行Ｘに、辺交差メッセージのＸ交差値をセットし、上で説明した処理を継続する。
【０１１１】
優先順位生成モジュール５１６は、テーブル３４内でも形成される、各優先順位に関する情報を保持するのに使用される、優先順位データ・テーブル５０４に関して動作する。優先順位データ・テーブル５０４の各レコードには、以下が含まれる可能性がある。
【０１１２】
（ｉ）塗潰しテーブル・アドレス
（ｉｉ）塗潰しタイプ

（ｉｉｉ）ラスタ演算コード
（ｉｖ）アルファ・チャネル演算コード
（ｖ）「ソース・ポップ」フラグ
（ｖｉ）「デスティネーション・ポップ」フラグ
（ｖｉｉ）本明細書で「ｘ独立」フラグと称する、この優先順位の色が所与のＹに対して一定であるかどうかを記録するフラグ。

【０１１３】
優先順位生成メッセージの受取り５２０の際に、優先順位生成モジュール５１６は、供給されたカウントによって示される回数だけ「画素優先順位生成動作」（下で説明する）を実行し、その後すぐに、動作を完了したことを知らせる信号５２２を優先順位更新モジュール５０６に送る。

【０１１４】
各画素の優先順位生成動作には、まず不透明アクティブ・フラグ配列５１０に対して優先順位エンコーダ５１４（たとえば、４０９６対１２ビット優先順位エンコーダ）を使用して、最高の不透明アクティブ・フラグの優先順位の数を決定することが含まれる。この優先順位（もし存在すれば）は、優先順位データ・テーブル５０４のインデクシングに使用され、そのように参照されたレコードの内容は、優先順位生成モジュール５１６から塗潰し優先順位メッセージ出力５９８に形成され、塗潰し色決定モジュール６００に送られる。さらに、優先順位が前のステップによって決定された（すなわち、少なくとも１つの不透明アクティブ・フラグがセットされた）場合には、決定された優先順位が保持され、これを「現行優先順位」と称する。優先順位が決定されなかった場合には、現行優先順位に０をセットする。その後、優先順位生成モジュール５１６は、アクティブ・フラグ配列５０８に対して変更済み優先順位エンコーダ５１２を繰り返し使用して、現行優先順位を超える最小のアクティブ・フラグを決定する。そのように決定された優先順位（があれば）は、優先順位データ・テーブル５０４のインデクシングに使用され、そのように参照されたレコードの内容は、塗潰し優先順位メッセージに形成され、塗潰し色決定モジュール６００に送られ５９８、その後、決定された優先順位が、現行優先順位の更新に使用される。このステップは、優先順位が決定されなくなる（すなわち、現行優先順位を超える、アクティブ・フラグでフラグを立てられた優先順位がなくなる）まで、繰り返し使用される。その後、優先順位生成モジュール５１６は、画素の終りメッセージを形成し、塗潰し色決定モジュール６００に送る。

【０１１５】
上で説明した基本動作に対する好ましい特徴として、優先順位生成モジュール５１６は、シーケンスの最初の画素を処理する間に、塗潰し色決定モジュール６００に転送する各メッセージのｘ独立フラグの値を記録する。転送されるメッセージのすべてでｘ独立フラグが指定されている場合には、隣接する辺交差の間の画素のスパンのすべての後続メッセージを、カウント－１の単一の反復指定によって置換することができる。これは、反復メッセージを作り、このシーケンスのその後の処理のすべての代わりに塗潰し色決定モジュール６００に送ることによって行われる。

【０１１６】
上で説明した基本動作に対するもう１つの好ましい特徴として、優先順位生成モジュール５１６は、各レベル生成メッセージの後に、接続５２２を介して優先順位更新モジュール５０６に最高の不透明優先順位を送る。優先順位更新モジュール５０６は、これをストア５２６に保持する。その後、優先順位更新モジュール５０６は、メッセージのＸ交差が現行Ｘより大きいことを単純にテストするのではなく、画素優先順位生成メッセージを作る前に、メッセージのＸ交差が現行Ｘより大きいこと、およびメッセージ内のレベルのうちの少なくとも１つが、最高の不透明優先順位以上であることをテストする。これを行うことによって、より少ない数の画素優先順位決定動作を実行することができ、より長い反復シーケンスを生成することができる。

【０１１７】

いくつかの実装で所望されるように、動作の反復メッセージまたはシーケンスが使用されない場合、同様の機能を、優先順位生成モジュール５１６の出力にキャッシュまたはＦＩＦＯ（図示せず）を組み込むことを介して達成できる。これは、たとえば４セル・キャッシュによって実施できる。キャッシュを用いると、優先順位更新モジュール５０６が、キャッシュがロードされると同時に作業を継続でき、これによって、出力５９８へのキャッシュのフラッシュと独立に次の優先順位レベルを生成できるようになる。

【０１１８】

図８および図９Ａ，Ｂに示されたグラフィック・オブジェクトの例を使用して、上で説明した優先順位更新処理を、図１２Ｃないし図１２Ｊおよび図１５Ａないし図１５Ｅに示された辺の交差を使用して、スキャン・ライン３５に関して示すことができる。

【０１１９】

図１５Ａないし図１５Ｅは、優先順位テーブル５０２および５０４の動作を示す図であり、優先順位テーブル５０２および５０４は、好ましい実施形態では、配列５０８および５１０とエンコーダ５１２および５１４と共に、レベル・アクティブ化テーブル５３０と称する単一のテーブルに合併される。図１５Ａからわかるように、辺交差メッセージは、辺処理モジュール４００からスキャン・ラインの順で受け取られ、テーブル５３０にロードされ、テーブル５３０は、優先順位の順で配置される。辺交差メッセージには、この例では、辺横断の非ゼロ・ワインディング規則に従うインクリメント方向が含まれる。優先順位テーブル５３０内の項目をセットしないことも可能である。

【０１２０】

図示のレベル・アクティブ化テーブル５３０には、非ゼロ・ワインディング規則または、適当な場合には奇数－偶数規則に従って辺から決定される塗潰しカウントのための列項目が含まれる。ニード・ビロウ・フラグは、優先順位の特性であり、ＬＯＡＤ＿ＰＲＩＯＲＩＴＩＥＳ＿ＰＲＯＰＥＲＴＩＥＳ命令の一部としてセットされる。ニード・ビロウは、テーブル５３０がロードされる時に、すべての優先順位レベルについてセットされる。「クリップ・カウント」および「塗潰しインデックス・テーブル」などの他の列を使用することができるが、この例では、説明を簡単にするために省略する。どのレベルもアクティブでない場合、対応する項目に０がセットされる。さらに、配列５１０および５０８の値は、後続の辺交差を受け取った後に、テーブル５３０から更新される。

【０１２１】

図１５Ａから、便宜上、複数のレコードが明瞭さのために省略されていることは明白である。通常、レベル・アクティブ化テーブル５３０には、優先順位の順で配置された、以下のレコードが含まれるはずである。

【０１２２】

・塗潰しカウント
・クリップ・カウント
・塗潰しタイプ
・下記を含むアクティブ化条件およびフラグ
（ｉ）ニード・ビロウ・フラグ
（ｉｉ）クリップ・タイプ
（ｉｉｉ）クリッパ・フラグ
・下記を含む合成用グラフィック演算およびフラグ
（ｉ）ラスタ演算コード
（ｉｉ）アルファ・チャネル演算コード
（ｉｉｉ）「ソース・ポップ」フラグ
（ｉｖ）「デスティネーション・ポップ」フラグ
（ｖ）ｘ独立フラグ
・塗潰し規則
・属性
・塗潰しテーブル・インデックス。

【０１２３】
テーブル５３０の内容は、優先順位決定モジュール５００で使用されない場合に、画素生成用の塗潰し色決定モジュール６００と合成演算用の画素合成モジュール７００のそれぞれにメッセージとして渡される。
【０１２４】
スキャン・ライン３５の最初の辺交差（図１２Ｅ）を図１５Ａに示すが、図１５Ａでは、Ｐ＝１の場合に、塗潰しカウントが、非ゼロ・ワインディング規則に従って辺の値に更新される。下のオブジェクトは存在しないので、「ニード・ビロウ」は、０にセットされるレベルである。
【０１２５】
テーブル５３０の前の状態はセットされていないので、配列５１０および５０８は、セットされないままになり、優先順位エンコーダ５１４は、優先順位の出力をディスエーブルされる。これは、優先順位生成モジュール５１６によって解釈され、優先順位生成モジュール５１６は、スキャン・ライン３５の最初の空白の部分である「オブジェクトなし」優先順位（たとえばＰ＝０）のカウントｎ＝４０（画素）を出力する。
【０１２６】
図１５Ｂは、図１２Ｆの辺交差を受け取った時の配置を示す図である。塗潰しカウントが更新される。その後、配列５１０および５０８は、テーブル５３０からの前の最高レベルを用いてセットされる。この時点で、モジュール５１６は、半透明の三角形８０との交差の前の不透明な赤いオブジェクト９０の辺９６を表す、カウントｎ＝４５、Ｐ＝１を出力する。
【０１２７】
図１５Ｃは、図１２Ｇの辺交差を受け取った時の配置を示す図である。非ゼロ・ワインディング規則のゆえに、塗潰しカウントが下向きに調節されていることに留意されたい。現在の辺交差を受け取る前に有効であるオブジェクトは、不透明ではないので、変更済み優先順位エンコーダ５１２が、ｎ＝（１１５－８５）＝３０画素の現行として出力される最高のアクティブ・レベルとして優先順位Ｐ＝２を選択するのに使用される。
【０１２８】
図１５Ｄは、図１２Ｈの辺交差を受け取った時の配置を示す図である。前に変更されたＰ＝２の「ニード・ビロウ」が、アクティブ配列５０８に転送され、したがって、優先順位エンコーダが、ｎ＝（１６０－１１５）＝４５画素の現行である値Ｐ＝１を出力することができることに留意されたい。
【０１２９】
図１５Ｅは、図１２Ｉの辺交差を受け取った時の結果を示す図であり、ｎ＝（１８０－１６０）＝２０画素のＰ＝０の出力が供給される。
【０１３０】
したがって、優先順位モジュール５００は、スキャン・ラインのすべての画素に関する、画素のカウントと、対応する優先順位表示値を出力する。
【０１３１】
塗潰し色決定モジュール６００の動作を、図６を参照して、以下に説明する。優先順位判定モジュール５００からの着信メッセージ５９８は、塗潰しデータ設定メッセージ、反復メッセージ、塗潰し優先順位メッセージ、画素の終りメッセージおよびスキャン・ラインの終りメッセージを含み、まず、塗潰しルックアップおよび制御モジュール６０４に渡される。塗潰しルックアップおよび制御モジュール６０４は、塗潰し色決定モジュール６００のさまざまな構成要素による使用のために、現行Ｘ位置カウンタ６１４および現行Ｙ位置カウンタ６１６を維持する。
【０１３２】
スキャン・ラインの終りメッセージを受け取った時には、塗潰しルックアップおよび制御モジュール６０４は、現行Ｘ位置カウンタ６１４を０にリセットし、現行Ｙ位置カウンタ６１６をインクリメントする。スキャン・ラインの終りメッセージは、その後、画素合成

モジュール７００に渡される。
【０１３３】
塗潰しデータ設定メッセージを受け取った時には、塗潰しルックアップおよび制御モジュール６０４は、塗潰しデータ・テーブル３６の指定された位置６０２にそのデータを記憶する。
【０１３４】
反復メッセージを受け取った時には、塗潰しルックアップおよび制御モジュール６０４は、反復メッセージからのカウントだけ現行Ｘ位置カウンタ６１４をインクリメントする。反復メッセージは、その後、画素合成モジュール７００に渡される。
【０１３５】
画素の終りメッセージを受け取った時には、塗潰しルックアップおよび制御モジュール６０４は、やはり現行Ｘ位置カウンタ６１４をインクリメントし、この画素の終りメッセージは、その後、画素合成モジュール７００に渡される。
【０１３６】
塗潰し優先順位メッセージを受け取った時には、塗潰しルックアップおよび制御モジュール６０４は、下記の処理を含む動作を実行する。
【０１３７】
（ｉ）塗潰し優先順位メッセージからの塗潰しタイプを使用して、テーブル３６のレコード・サイズを選択する。
【０１３８】
（ｉｉ）塗潰し優先順位メッセージからの塗潰しテーブル・アドレスと、上で決定されたレコード・サイズを使用して、塗潰しデータ・テーブル３６からレコードを選択する。
【０１３９】
（ｉｉｉ）塗潰し優先順位メッセージからの塗潰しタイプを使用して、塗潰し色の生成を実行するサブモジュールを決定し、選択する。サブモジュールには、ラスタ画像モジュール６０６、フラット・カラー・モジュール６０８、リニア・ランプ・カラー・モジュール６１０および不透明タイル・モジュール６１２を含めることができる。
【０１４０】
（ｉｖ）決定されたレコードを、選択されたサブモジュール６０６ないし６１２に供給する。
【０１４１】
（ｖ）選択されたサブモジュール６０６ないし６１２が、供給されたデータを使用して、色と不透明度の値を決定する。
【０１４２】
（ｖｉ）決定された色と不透明度は、塗潰し色メッセージからの残りの情報、すなわち、ラスタ演算コード、アルファ・チャネル演算コード、ソース・ポップ・フラグおよびデスティネーション・ポップ・フラグと組み合わされて、色合成メッセージを形成し、この色合成メッセージは、接続６９８を介して画素合成モジュール７００に送られる。
【０１４３】
好ましい実施形態では、決定された色と不透明度が、８ビットの精度の赤、緑、青および不透明度の４つ組であり、通常の形で３２ビット／画素が与えられる。しかし、シアン、マゼンタ、黄および黒の、不透明度を含有する４つ組か、他の多数の既知の色表現のうちの１つを、その代わりに使用することができる。赤、緑、青および不透明度の場合を、下の説明で使用するが、この説明は、他の場合にも適用することができる。
【０１４４】
ラスタ画像モジュール６０６、フラット・カラー・モジュール６０８、リニア・ランプ・カラー・モジュール６１０および不透明タイル・モジュール６１２の動作を、これから説明する。
【０１４５】
フラット・カラー・モジュール６０８は、供給されたレコードを、３つの８ビットの色成

分（通常は赤、緑および青の成分と解釈される）と１つの８ビットの不透明度値（通常は、指定された色によって覆われる画素の割合の尺度と解釈され、０は覆われないすなわち完全な透明を意味し、２５５は完全に覆われるすなわち完全な不透明を意味する）を含む固定フォーマットのレコードと解釈する。この色と不透明度の値は、接続６９８を介して直接に出力され、それ以上の処理なしで、決定された色および不透明度を形成する。

【０１４６】

リニア・ランプ・カラー・モジュール６１０は、供給されたレコードを、３つの色成分および１つの不透明度成分に関連する４組の定数ｃｘ、ｃｙおよびｄと、リニア・ランプの基準点の座標である２つの位置値ｘｂａｓｅおよびｙｂａｓｅを含む固定フォーマットのレコードと解釈する。基準点では、色成分と不透明度成分が、それに関連付けられたｄ値を有する。

【０１４７】

４組のそれぞれについて、結果の値ｒは、次式を使用して、現在のＸＹ座標とｘｂａｓｅおよびｙｂａｓｅ定数に３つの定数を組み合わせることによって計算される。

【０１４８】

ｒ＝ｃｌａｍｐ（ｃｘ×（Ｘ－ｘｂａｓｅ）＋ｃｙ×（Ｙ－ｙｂａｓｅ）＋ｄ）

ここで、関数ｃｌａｍｐは、次のように定義される。

【０１４９】

$$\mathrm{clamp}(x)=\begin{cases}255 & 255\leqq x\\ \lfloor x\rfloor & 0\leqq x<255\\ 0 & x<0\end{cases}\quad 。$$

【０１５０】

代替実装では、リニア・ランプ・カラー・モジュール６１０は、供給されたレコードを、３つの色成分および１つの不透明度成分に関連する、４組の３つの定数ｃｘ、ｃｙおよびｄを含む固定フォーマットのレコードと解釈する。これらの４組のそれぞれについて、結果の値ｒは、次式を使用して、３つの定数を現行Ｘカウントであるｘおよび現行Ｙカウントであるｙと組み合わせることによって計算される。

【０１５１】

ｒ＝ｃｌａｍｐ（ｃｘ×ｘ＋ｃｙ×ｙ＋ｄ）

ここで、関数ｃｌａｍｐは、上で定義されたものである。

【０１５２】

そのように作られた４つの結果は、色と不透明度の値に形成される。この色と不透明度の値は、接続６９８を介して直接に出力され、それ以上の処理なしで決定された色および不透明度を形成する。

【０１５３】

同一の結果を与える他の算術計算を使用することができる。

【０１５４】

不透明タイル・モジュール６１２は、供給されたレコードを、３つの８ビット色成分、１つの８ビット不透明度値、整数のＸ位相（ｐｘ）、Ｙ位相（ｐｙ）、Ｘスケール（ｓｘ）、Ｙスケール（ｓｙ）および６４ビットのマスクを含む固定フォーマットのレコードと解釈する。これらの値は、表示リスト生成から発し、通常は、元のページ記述に含まれる。ビット・マスク内のビット・アドレスａは、次の式で決定される。

【０１５５】

$$a=((x/2^{sx}+px)\bmod 8)+((y/2^{sy}+py)\bmod 8)\times 8\quad 。$$

【０１５６】

ビット・マスク内のアドレス「ａ」のビットが検査される。検査されるビットが１の場合には、レコードからの色と不透明度が、モジュール６１２の出力に直接にコピーされ、決定された色および不透明度を形成する。検査されるビットが０の場合には、３つの０成分

値を有する色と０の不透明度値が形成され、決定された色および不透明度として出力される。

【０１５７】

ラスタ画像モジュール６０６は、供給されたレコードを、６つの定数ａ、ｂ、ｃ、ｄ、ｘｂａｓｅおよびｙｂａｓｅと、サンプリングされるラスタ画像画素データ１６の各ラスタ・ライン内のビット数の整数カウント（ｂｐｌ）と、画素タイプとを含む固定フォーマットのレコードと解釈する。画素タイプは、ラスタ画像画素データ内の画素データ１６を、次のうちのどれとして解釈するかを示す。

【０１５８】

（ｉ）１ビット毎画素の白黒不透明画素
（ｉｉ）１ビット毎画素の不透明黒または透明の画素
（ｉｉｉ）８ビット毎画素のグレイ・スケール不透明画素
（ｉｖ）８ビット毎画素の黒不透明スケール画素
（ｖ）２４ビット毎画素の不透明３色成分画素
（ｖｉ）３２ビット毎画素の３色成分と不透明度の画素
他の多くのフォーマットが可能である。

【０１５９】

ラスタ画像モジュール６０６は、画素タイプ・インジケータを使用して、ビット単位の画素サイズ（ｂｐｐ）を決定する。その後、ラスタ画像画素データ１６内のビット・アドレスａを、次式から計算する。

【０１６０】

$$a = bpp * \lfloor a \times (x - xbase) + c \times (y - ybase) \rfloor + bpl \times \lfloor b \times (x - xbase) + d \times (y - ybase) \rfloor \quad 。$$

【０１６１】

レコード６０２からの画素タイプに従って解釈された画素は、ラスタ画像画素データ１６内の計算されたアドレス「ａ」から取り出される。この画素は、３つの８ビット色成分と１つの８ビット不透明度成分を有するために、必要に応じて展開される。「展開」とは、たとえば、８ビット／画素のグレイ・スケール不透明ラスタ画像からの画素が、赤、緑および青の成分のそれぞれに適用されるサンプリングされた８ビット値と、完全な不透明がセットされた不透明度成分を持つことを意味する。これが、画素合成モジュール７００への決定された色および不透明度の出力６９８を形成する。

【０１６２】

その結果、表示可能オブジェクト内で有効なラスタ画素データは、メモリ内の画像画素データ１６へのマッピングの決定を介して得られる。これによって、ラスタ画素データのオブジェクト・ベース画像へのアフィン変換が効率的に実施され、これは、グラフィック・オブジェクトとの合成が発生する可能性がある場合に画像ソースからの画素データをフレーム・ストアに転送する従来技術の方法より効率的である。

【０１６３】

上記に対する好ましい特徴として、任意選択として、ラスタ画像画素データ１６内の画素間の補間を、まず中間結果ｐおよびｑを次式に従って計算することによって実行することができる。

【０１６４】

$$p = a \times (x - xbase) + c \times (y - ybase)$$
$$q = b \times (x - xbase) + d \times (y - ybase) \quad 。$$

【０１６５】

次に、ラスタ画像画素データ１６内の４画素のビット・アドレスa_{00}、a_{01}、a_{10}およびa_{11}を、次式に従って決定する。

$$a_{00} = bpp \times \lfloor p \rfloor + bpl \times \lfloor q \rfloor$$
$$a_{01} = a_{00} + bpp$$
$$a_{10} = a_{00} + bpl$$

$a_{11}=a_{00}+bpl+bpp$ 。

【０１６６】

次に、結果の画素成分値ｒを、次式に従って、色成分および不透明度成分のそれぞれについて決定する。

$r=interp(interp(get(a_{00}),get(a_{01}),p),interp(get(a_{10}),get(a_{11}),p),q)$

ここで、関数ｉｎｔｅｒｐは、次のように定義される。

$interp(a,b,c)=a+(b-a)\times(c-\llcorner c\lrcorner)$ 。

【０１６７】

上の諸式で、表現└値┘＝ｆｌｏｏｒ（値）であり、ｆｌｏｏｒ演算では、値の端数部分の切捨が行われる。

【０１６８】

ｇｅｔ関数は、所与のビット・アドレスでの、ラスタ画像画素データ１６からサンプリングされた現行画素成分の値を返す。いくつかの画像タイプのいくつかの成分について、これが暗黙の値になる可能性があることに留意されたい。

【０１６９】

上記に対する好ましい特徴として、任意選択として、供給されたレコードから読み取られるタイル・サイズを用いる剰余演算によって現行Ｘ位置カウンタ６１４および現行Ｙ位置カウンタ６１６から導出されるｘおよびｙの値を上式で使用することによって、画像タイリングを実行することができる。

【０１７０】

多数の他のこのような塗潰し色生成サブモジュールが可能である。

【０１７１】

画素合成モジュール７００の動作をこれから説明する。塗潰し色決定モジュール６００からの着信メッセージには、反復メッセージ、色合成メッセージ、画素の終りメッセージおよびスキャン・ラインの終りメッセージが含まれるが、このメッセージは、順番に処理される。

【０１７２】

反復メッセージまたはスキャン・ラインの終りメッセージを受け取った時には、画素合成モジュール７００は、それ以上の処理なしで画素出力ＦＩＦＯ７０２にそのメッセージを転送する。

【０１７３】

色合成メッセージを受け取った時には、画素合成モジュール７００は、概要を示せば、色合成メッセージからの色と不透明度を、色合成メッセージからのラスタ演算およびアルファ・チャネル演算に従って、画素合成スタック３８からポップされた色および不透明度と組み合わせる。その後、結果を画素合成スタック３８にプッシュする。色合成メッセージの受取り時に実行される処理の説明は、下で与える。

【０１７４】

画素の終りメッセージを受け取った時には、画素合成モジュール７００は、画素合成スタック３８から色と不透明度をポップする。ただし、スタック３８が空の場合には不透明の白の値が使用される。結果の色と不透明度は、画素出力ＦＩＦＯに転送される画素出力メッセージに形成される。

【０１７５】

既知の合成アプローチは、ポーター（Ｐｏｒｔｅｒ，Ｔ）およびダフ（Ｄｕｆｆ，Ｔ）著「Ｃｏｍｐｏｓｉｔｉｎｇ　Ｄｉｇｉｔａｌ　Ｉｍａｇｅｓ」、Ｃｏｍｐｕｔｅｒ　Ｇｒａｐｈｉｃｓ誌、Ｖｏｌ．１８　Ｎｏ．３、１９８４年、第２５３～２５９ページに記載されている。ポーターダフ合成演算の例を、図２１に示す。しかし、このようなアプローチでは、合成によって形成される交差領域内のソースおよびデスティネーションの色の処理だけが可能であり、その結果、交差領域内の透明度の影響に適合できないという点で、不完全である。その結果、ポーターおよびダフによって定義されたラスタ演算は、透明

度の存在下で基本的に動作不能である。
【０１７６】
色合成メッセージを受け取った時に画素合成モジュール７００によって実行される処理を、これから説明する。
【０１７７】
色合成メッセージを受け取った時に、画素合成モジュール７００は、まず、ソースの色と不透明度を形成する。これは、色合成メッセージでポップ・ソース・フラグがセットされていない限り、色合成メッセージで供給された色と不透明度が使用され、ポップ・ソース・フラグがセットされている場合には、その代わりに画素合成スタック３８から色がポップされる。この時、すなわち、画素合成スタックのポップ中に、画素合成スタック３８が空であることがわかった場合、エラー表示なしで不透明の白の色値が使用される。次に、デスティネーションの色と不透明度が、画素合成スタック３８からポップされる。ただし、色合成メッセージでデスティネーション・ポップ・フラグがセットされていない場合には、スタック・ポインタが「ポップ」動作中に変更されず、その結果スタック３８の最上部からの読み取りになる。
【０１７８】
ソースの色および不透明度をデスティネーションの色および不透明度と組み合わせる方法を、図７Ａないし図７Ｃを参照して以下に説明する。色および不透明度の値は、通常は０から２５５までの範囲の８ビット値として記憶されるが、ここでは説明の目的のために、０から１までの範囲（すなわち正規化されている）とみなす。２画素を合成する目的のために、各画素が２つの領域、すなわち、完全に不透明な領域と完全に透明な領域に分割されるものとみなし、不透明度値は、これら２つの領域の比率の表示であるとみなされる。図７Ａに、図示されない３成分色値と、不透明度値（ｓｏ）を有するソース画素７０２を示す。ソース画素７０２の影付きの領域は、画素７０２の完全に不透明な部分７０４を表す。同様に、図７Ａの影付きでない領域は、ソース画素７０２のうちで、完全に透明であるとみなされる部分７０６を表す。図７Ｂに、ある不透明度値（ｄｏ）を有するデスティネーション画素７１０を示す。デスティネーション画素７１０の影付きの領域は、画素７１０の完全に不透明な部分７１２を表す。同様に、画素７１０は、完全に透明な部分７１４を有する。ソース画素７０２およびデスティネーション画素７１０の不透明領域は、組合せのために、互いに直交するとみなされる。これら２つの画素のオーバーレイ７１６を、図７Ｃに示す。対象の３つの領域が存在し、これには、ｓｏ×（１－ｄｏ）の面積を有するデスティネーション外ソース７１８、面積ｓｏ×ｄｏを有するソース・デスティネーション交差７２０および、（１－ｓｏ）×ｄｏの面積を有するソース外デスティネーション７２２が含まれる。これら３つの領域のそれぞれの色値は、概念上は独立に計算される。デスティネーション外ソース領域７１８は、その色をソース色から直接にとる。ソース外デスティネーション領域７２２は、その色をデスティネーション色から直接にとる。ソース・デスティネーション交差領域７２０は、その色をソース色とデスティネーション色の組合せからとる。上で説明した他の動作とは別個の、ソース色とデスティネーション色の組合せの処理は、ラスタ演算と称され、これは、画素合成メッセージからのラスタ演算コードによって指定される、１組の関数のうちの１つである。好ましい実施形態に含まれるラスタ演算の一部を、表２に示す。
【０１７９】
【表３】

TABLE 2

Raster operation code	Operation	Comment
0x00	r = 0	BLACKNESS
0x01	r = src & dest	SRCAND
0x02	r = src & ~dest	SRCERASE
0x03	r = src	SRCCOPY
0x04	r = ~src & dest	
0x05	r = dest	NOP
0x06	r = src ^ dest	SRCINVERT
0x07	r = src | dest	SRCPAINT
0x08	r = ~(src | dest)	NOTSRCERASE
0x09	r = ~(src ^ dest)	
0x0a	r = ~dest	DSTINVERT
0x0b	r = src | ~dest	
0x0c	r = ~src	NOTSRCCOPY
0x0d	r = ~src | dest	MERGEPAINT
0x0e	r = ~(src & dest)	
0x0f	r = 0xff	WHITENESS
0x10	r = min(src, dest)	
0x11	r = max(src, dest)	
0x12	r = clamp(src + dest)	
0x13	r = src	
0x14	r = clamp(src - dest)	
0x15	r = dest	
0x16	r = clamp(dest - src)	
0x17	r = clamp(src + dest) where dest is signed	
0x18	r = threshold (dest, src)	
0x19	r = threshold (src, dest)	
0x1a	r = ~ dest	
0x1b	o = luminance (dest, src)	
0x1c	r = ~src	
0x1d	o = ckey(dest; src +/- o)	

【０１８０】
各関数は、ソース色およびデスティネーション色の対応する色成分の対のそれぞれに適用されて、結果の色において同様の成分が得られる。多くの他の関数が可能である。
【０１８１】
画素合成メッセージからのアルファ・チャネル演算も考慮されている。アルファ・チャネル演算は、３つのフラグを使用して実行され、フラグのそれぞれは、ソース画素７０２とデスティネーション画素７１０のオーバーレイ７１６内の対象の領域のうちの１つに対応する。領域のそれぞれについて、領域の不透明度値が形成され、この不透明度値は、アル

ファ・チャネル演算で対応するフラグがセットされていない場合には０、それ以外の場合にはその領域の面積である。

【０１８２】

結果の不透明度は、領域の不透明度の合計から形成される。結果の色の各成分は、領域の色と領域の不透明度の対のそれぞれの積の和を結果の不透明度で除算することによって形成される。

【０１８３】

結果の色および不透明度は、画素合成スタック３８にプッシュされる。

【０１８４】

エクスプレッションツリーは、画像の形成に必要な合成演算を記述するのにしばしば使用され、通常は、葉ノード、単項ノードおよび２項ノードを含む複数のノードが含まれる。葉ノードは、エクスプレッションツリーの最も外側のノードであり、子ノードを持たず、画像のプリミティブ要素を表す。単項ノードは、ツリーの単項演算子の下の部分から来る画素データを変更する演算を表す。２項ノードは、通常は、左と右のサブツリーに分岐し、各サブツリー自体は、少なくとも１つの葉ノードを含むエクスプレッションツリーである。

【０１８５】

任意の形状のオブジェクトとの合成の時には、上で述べたさまざまなスタック演算が、画像の異なる区域について異なり、特定の位置でアクティブなオブジェクトに依存するという問題が生じる。

【０１８６】

図１７Ａおよび図１７Ｂに、ソース（Ｓ）とデスティネーション（Ｄ）の間の通常の２項演算（エクスプレッションツリーとして図示）を示す。実際に実行される演算とは無関係に、図１７Ａの２項演算は、下に示すアクティビティの４つの場合または領域に分解される。

【０１８７】

１．（Ａ）Ｓがアクティブ、（Ｂ）Ｄが非アクティブ
２．（Ａ）Ｓがアクティブ、（Ｂ）Ｄがアクティブ
３．（Ａ）Ｓが非アクティブ、（Ｂ）Ｄがアクティブ
４．（Ａ）Ｓが非アクティブ、（Ｂ）Ｄが非アクティブ。

【０１８８】

ケース４は、必ず無演算（ＮＯＰ）が実行される必要があることをもたらし、その結果、２進木のアクティブ・レベルには３つの異なる組合せが存在することになる。この概念の３項、４項およびさらに高次の木への拡張は、当業者には明白である。

【０１８９】

その結果、合成スタック３８（２項の例の場合）を構築する時に、上で識別した３つの演算のうちの１つを、スタックによって実施する必要がある。さらに、スタック内の各オブジェクトに関連する異なる演算は、レベル・アクティブ化テーブルでそのオブジェクトの下にあるものに依存する。ペインタのアルゴリズムで発生する、単純なＯＶＥＲ演算を使用するオブジェクトのレンダリングの場合に、これは問題にならない。しかし、他の合成演算に関して、スタック演算は、合成演算のオペランドのアクティビティに依存して変更される必要がある。これは、スタック演算を提供するレベルをクリッピングすることによって行うことができるが、各レベルに適用されるクリップの数が、急激に増加し、スタック演算の処理が困難になる。問題になる演算の例が、あるオブジェクト（オペランド）が、他方のオブジェクトをクリッピング（その境界を変更）し、交差領域で可変透明度を有する場合の、図２１に示されたポーターダフ演算のＯＵＴおよびＲＯＵＴである。

【０１９０】

この問題に対処するために、本明細書で「アクティビティ」テーブルと称するもう１つのテーブルを設ける。このテーブルは、レベル・アクティブ化テーブルの付属品として働いて、上で述べた代替処置の論理的決定をもたらす。

【０１９１】
図１８Ａに、基本的に３つの部分を含む、包括的なアクティビティ・テーブル８００を示す。第１部分８０２は、処理中の特定の塗潰しレベルのアクティブ化条件を提供する。第２部分８０４には、それぞれのレベル（具体的には２項の例の）に適当な、上で参照した異なる処置のそれぞれが含まれる。第３部分８０６は、ソース・オブジェクトまたはデスティネーション・オブジェクトが特定のレベルでアクティブであるかどうかを示す。処置部分８０４に含まれる項目は、具体的な演算そのものとするか、その代わりに、適当な場合にレベル・アクティブ化テーブルへのポインタとすることができることに留意されたい。

【０１９２】
また、さまざまな演算によって、他の演算にデータを供給することができる場合と、できない場合があることに留意されたい。その結果、アクティビティ・テーブル８００は、演算がデータを供給するさまざまな条件を示すフラグを組み込むように変更することができる。

【０１９３】
アクティビティ・テーブルの好ましい形態のデータ構造を、図１８Ｂにテーブル８１０として示す。テーブル８１０には、そのデータを使用する次の演算の項目へのポインタ８１４（Ｎｅｘｔ＿Ｌｅｖｅｌ）と、その演算がデータを提供するエクスプレッションツリーの枝（Ｓｒｃ＿Ａｃｔｉｖｅ／Ｄｅｓｔ＿ａｃｔｉｖｅ）を示すフラグ８０６（または、３項以上のツリーが使用される場合にはフラグの組）が含まれる。テーブル８１０には、その演算がソース枝またはデスティネーション枝にデータを供給するかどうかを示すフラグ８１６も含まれる。そうである場合には、次のレベルの項目のＳｒｃ＿ａｃｔｉｖｅまたはＤｅｓｔ＿Ａｃｔｉｖｅフラグ８０６は、この演算のアクティビティ状態８１８が変化した時に、それ相応に調節される。演算は、特定の組合せにおいてのみデータを供給するので、これを示すために、別のフラグ８１２（ｄａｔａ＿ｉｎ＿＊）が設けられる。フラグ８１２とＳｒｃ／Ｄｅｓｔ＿Ａｃｔｉｖｅフラグ８０６の組合せによって、あるレベルのアクティビティ状態が決定される。さらに、どの演算も、それ自体のアクティビティ状態が変化した場合には次のレベルの状態を変更するだけでよいので、ノード・アクティブ・フラグ８１８が、そのような状況を監視するために設けられる。

【０１９４】
したがって、右側の葉ノードについて、Ｐｕｓｈ動作と、次の演算レコードのＤｅｓｔ＿ａｃｔｉｖｅフラグをアクティブ化することが必要である。左側の葉ノードについて、デスティネーションがすでにアクティブである可能性があることに留意して、辺交差のＳｒｃ＿ａｃｔｉｖｅフラグをアクティブ化することが必要である。

【０１９５】
図１８Ｂでは、アクティブ化条件８０２に、葉ノードのアクティブ化を決定する塗潰し規則と、レベル・アクティブ化テーブルについて上で説明した形で使用される塗潰しカウントが含まれる。クリップ・カウントも、上で説明した形で動作する。辺の交差によって、テーブル８１０の（ソース）レベルがアクティブ化される。

【０１９６】
レベルのアクティブ化状態が変化した時には、その変化が、Ｎｅｘｔ＿Ｌｅｖｅｌ項目８１４によって指されるレベルに伝播される。Ｄａｔａ＿ｉｓ＿Ｓｒｃフラグ８１６の状態に応じて、Ｓｒｃ＿Ａｃｔｉｖｅ／Ｄｅｓｔ＿Ａｃｔｉｖｅフラグ８０６が、次のレベルで変更される。この変更は、次のレベルの状態も変化する場合に伝播される。テーブル項目には、それぞれ場合１、２および３の演算が格納される。これは、レベル・アクティブ化テーブル内のレベルへのポインタとするか、実際の演算（たとえば、アルファ演算、カラー演算、塗潰しタイプおよびスタック演算フラグ）とすることができる。その代わりに、演算が必要でない場合には、これらをＮＵＬＬにすることができる。

【０１９７】
あるノード・レベルのアクティブ化状態は、Ｓ∩Ｄ¯ｏｐ、Ｓ∩Ｄｏｐ、Ｓ¯∩Ｄｏｐの

それぞれのｄａｔａ＿ｉｎフラグ群８１２と、そのノード・レベルのＳｒｃ／Ｄｅｓｔ＿ａｃｔｉｖｅフラグ８０６によって決まる、エクスプレッションツリーの次の演算のためのデータがあるかどうかによって決定される。これは、このテーブルではＮｏｄｅ＿Ａｃｔｉｖｅフラグ８１８として図示されている。

【０１９８】

この実装の具体的な例を、図１９に示されるエクスプレッションツリー８３０について、図２０Ａに示される、対応する初期アクティビティ・テーブル８２０を用いて示す。

【０１９９】

エクスプレッションツリー８３０は、オペランドＡ、ＢおよびＣのページへのレンダリングを提供し、後者は、図示の完全さのために、ツリー８３０では右の葉ノード、ＰＡＧＥとして示されている。ＰＡＧＥは、常にアクティブであり、画像出力空間全体を含み、したがって、アクティビティ・テーブル８２０から省略することができる。

【０２００】

ＢとＣは葉ノードなので、これらは、テーブル８２０の低位レベルを形成し、それぞれが、それぞれの演算子の合成スタックへのプッシュを引き起こすことのできるアクティブ化演算８０４をもたらす。これらのそれぞれが右側の葉ノードなので、まずＣがプッシュされ、オペランドＣに対する演算がないので、Ｓ¯∩ＤｏｐはＮＯＰになる。ｄａｔａ＿ｉｎ＿＊ｏｐフラグ８１２、Ｎｅｘｔ＿Ｌｅｖｅｌフラグ８１４およびＤａｔａ＿ｉｓ＿Ｓｒｃフラグ８１６も更新される。オペランドＢは、対応する処置をもたらす。

【０２０１】

アクティビティ・テーブル８２０の次のレベルは、左の葉ノードＡとそれに対応する演算子Ｏｐ２によって形成される。このレベルのアクティブ化演算８０４は、それぞれが演算の間の差を表す修飾子ａまたはｂによってそれぞれが変更されるＡｏｐ２Ｂであるの∩ＤｏｐおよびＳ∩Ｄｏｐのそれぞれを用いて更新される。演算Ｏｐ２は、Ｓのデータを提供するだけであり、これは、Ｄがアクティブでないことを（すなわちＳ¯∩Ｄｏｐ）場合にスタックからＢをポップするアクティビティによって表される。

【０２０２】

テーブル８２０の次のレベルは、Ｏｐ１に関し、アクティブ化演算８０４でのそれぞれ修飾された結果ａ、ｂおよびｃを作る。最後のレベルｏｖｅｒでは、ＰＡＧＥが常にアクティブなので、Ｓ∩Ｄ¯ｏｐとの∩Ｄｏｐは、スタックにプッシュされるＮＯＰをもたらす。単純な交差Ｓ∩Ｄｏｐ内だけで、ｏｖｅｒがアクティブになる。

【０２０３】

この例について、Ａ、ＢおよびＣが、当初は非アクティブであり、これらのオペランドが、その後、順番にアクティブ化される場合に何が起きるかを検討する。

【０２０４】

Ａが最初にアクティブ化される場合、ＡＯｐ２ａがこのレベルでアクティブ化され、Ｓｒｃ＿Ａｃｔｉｖｅフラグ８０６の設定に反映される。Ｓ∩Ｄ¯フラグ８１２がセットされているので（Ｂがまだアクティブではないので）、Ｎｏｄｅ＿Ａｃｔｉｖｅフラグ８１８が、このレベルについてセットされる。Ｎｏｄｅ＿Ａｃｔｉｖｅの状態が変化したので、Ｏｐ１レベルのの∩Ｄｏｐは、ＡＯｐ２レベルでセットされることに留意されたい。Ｏｐ１レベルは、Ｓｒｃ＿Ａｃｔｉｖｅと（Ｄｅｓｔ＿Ａｃｔｉｖｅ）¯を有するので（Ｃがまだアクティブではないので）、Ｏｐ１ａが、このレベルでアクティブ化される。Ｓ∩Ｄ¯がセットされているので、Ｎｏｄｅ＿Ａｃｔｉｖｅ８１８が、Ｏｐ１レベルについてセットされる。Ｎｏｄｅ＿Ａｃｔｉｖｅ８１８の状態が変化したので、ｏｖｅｒレベルのＳｒｃ＿Ａｃｔｉｖｅフラグ８０６がセットされる。ｏｖｅｒレベルはＳｒｃ＿ａｃｔｉｖｅとＤｅｓｔ＿Ａｃｔｉｖｅを有する（ＰＡＧＥが常にアクティブなので）ので、ｏｖｅｒはこのレベルでアクティブ化される。Ｓ∩Ｄがセットされているので、Ｎｏｄｅ＿Ａｃｔｉｖｅがセットされ、Ｎｏｄｅ＿Ａｃｔｉｖｅの状態は変化しない。その後、これ以上の処置は不要である。合成スタック３８は、この段階で、テーブル８２０から確立でき、図２０Ｂに示さ

れたものになる。

【０２０５】

図２０Ｃに移って、Ｂが次にアクティブ化される場合、Ｓ∩Ｄ¯がセットされている（いずれにせよＤは関係ない）ので、このレベルでＰｕｓｈ　Ｂがアクティブ化される。Ｎｏｄｅ＿Ａｃｔｉｖｅ８１８は、このレベルについてセットされる。Ｎｏｄｅ＿Ａｃｔｉｖｅの状態が変化したので、ＡＯｐ２レベルのＤｅｓｔ＿Ａｃｔｉｖｅフラグがセットされる。その後、ＡＯｐ２ｂＢがアクティブ化され、ＡＯｐ２ａＢが非アクティブ化される。Ｓ∩Ｄがセットされているので、Ｎｏｄｅ＿Ａｃｔｉｖｅはセットされたままになり、Ｎｏｄｅ＿Ａｃｔｉｖｅの状態は、ＡＯｐ２ａに関しては変化しない。これ以上の処置は発生しない。合成スタック３８は、図２０Ｄに示されたものになる。

【０２０６】

図２０Ｅからわかるように、Ｃが次にアクティブ化される場合、このレベルでＰｕｓｈ　Ｃがアクティブ化される。Ｓはアクティブであり、Ｄは無関係なので、Ｎｏｄｅ＿Ａｃｔｉｖｅ８１８は、このレベルについてセットされる。Ｎｏｄｅ＿ａｃｔｉｖｅが変化したので、Ｏｐ１レベルのＤｅｓｔ＿Ａｃｔｉｖｅフラグがセットされる。Ｏｐ１ｂがアクティブ化されるので、Ｏｐ１ａは非アクティブ化される。Ｓ∩Ｄのデータがセットされているので、Ｎｏｄｅ＿Ａｃｔｉｖｅはセットされたままになる。Ｏｐ１レベルではは Ｎｏｄｅ＿Ａｃｔｉｖｅが変化しないので、これ以上の処置は不要である。合成スタック３８は、図２０Ｆに示されたものになる。

【０２０７】

この手順は、図１９のエクスプレッションツリー全体の評価について継続され、したがって、さまざまなアクティブ化条件８０２およびアクティビティ・インジケータ８０４による要求に応じて合成スタックにプッシュすることのできるさまざまな演算が確立される形で確立されるアクティビティ・テーブル８２０をもたらす。この形で、クリッピングまたは実行される他の演算の種類に無関係に、正しいレベルの正しい演算である状態で、評価されるエクスプレッションツリーの複雑さと無関係に、スタックを維持することができる。重要なことに、エクスプレッションツリーの大部分が、表示リストの形成を介して評価されるが、表示リストの生成は、通常は、クリッピングなどのさまざまなオブジェクトの演算の変化を説明することができず、これらの演算は、合成式の評価中に実施することが必要になる。

【０２０８】

さらなるフラグ、辺の交差によってアクティブ化することのできるｓｒｃ＿ｉｓ＿ｌｅａｆ＿ｎｏｄｅ用のフラグと、ｄｅｓｔ＿ｉｓ＿ＰＡＧＥ（常にアクティブ）用のフラグが、有用になる可能性があることに、さらに留意されたい。ｄｅｓｔ＿ｉｓ＿ＰＡＧＥの場合、Ｓ∩Ｄ¯の場合は絶対に発生しないので、これを無視することが可能である。

【０２０９】

上では、アクティビティ・テーブル８２０が、エクスプレッションツリー８３０の構造に基づいてどのように構築され、その項目が、ツリー８３０のさまざまなオペランドの変化するアクティブ化を介してどのように完了される（すなわち書き込まれる）かを示した。テーブル８２０の具体的な例では、異なるアクティブ化と可能な結果を説明するために、７２（＝２×２×３×３×２）個のスタック構造が生じる可能性がある。この形で、条件８０２、８０６、８１２、８１４、８１６および８１８の論理評価が、特定のレベルの適当なスタック演算として識別される正しいアクティビティ８０４をもたらす。

【０２１０】

代替実装では、独立のテーブルとして構成されるのではなく、アクティビティ・テーブル８２０を、レベル・アクティブ化テーブル５３０に合併して、組み合わされたテーブル８３０を与えることができる。これによって、データの複製が行われなくなると同時に、優先順位エンコーダ５１２および５１４が正しい辺優先順位だけではなく、アクティブ化演算も選択できるようになり、後者は、モジュール６００から導出された塗潰し色を使用する合成モジュール７００による評価のために画素合成スタック３８に転送（漸進的に）さ

れる。このような配置を、図２０Ｇに機能的に図示する。

【０２１１】

その代わりに、図２０Ｈに機能的に示されているように、アクティビティ・テーブル８２０が、レベル・アクティブ化テーブル５３０の前にあってもよい。このような場合には、塗潰しカウントとクリップ・カウントの列を、アクティビティ・テーブル８２０に含め、レベル・アクティブ化テーブル５３０から省略することができる。図２０Ｉに機能的に示されているもう１つの代替構成では、アクティビティ・テーブル８２０が、レベル・アクティブ化テーブル５３０の後にある。この場合、塗潰しカウントとクリップ・カウントは、レベル・アクティブ化テーブル５３０に含まれるので、アクティビティ・テーブル８２０から省略することができる。アクティビティ・テーブル８２０が図２０Ａに示されるように構成されるいくつかの応用例では、レベル・アクティブ化テーブル５３０を省略することができる。

【０２１２】

アクティビティ・テーブル８２０およびスタック３８に関して上で説明した演算コードは、表示リストから、具体的には命令ストリーム１４（図３参照）から導出される。演算コードは、図３に示された処理ステージのパイプラインを介して、他のデータ（たとえば辺交差、塗潰しデータなど）と共に並列に転送され、画素合成モジュール７００は、優先順位決定、レベル・アクティブ化および塗潰し決定の結果として決定される順序でｏｐコードをスタックに置く。

【０２１３】

画素出力モジュール８００の動作を、これから説明する。着信メッセージは、画素出力ＦＩＦＯから読み取られ、これには、画素出力メッセージ、反復メッセージおよびスキャン・ラインの終りメッセージが含まれ、順番に処理される。

【０２１４】

画素出力メッセージを受け取った時に、画素出力モジュール８００は、画素を記憶し、その画素をその出力に転送する。反復メッセージを受け取った時には、最後に記憶した画素が、反復メッセージからのカウントによって指定される回数だけ、出力８９８に転送される。スキャン・ラインの終りメッセージを受け取った時には、画素出力モジュール８００は、そのメッセージをその出力に渡す。

【０２１５】

出力８９８は、必要に応じて、画素画像データを使用する装置に接続することができる。このような装置には、ビデオ表示ユニットまたはプリンタなどの出力装置、または、ハード・ディスク、ライン・ストア、バンド・ストアまたはフレーム・ストアを含む半導体ＲＡＭなどのメモリ記憶装置、または、コンピュータ・ネットワークが含まれる。

【０２１６】

ここまでに説明してきたシステムの実施に伴う困難の１つが、多くの場合に、このシステムが、関連する不透明度値を有する画素を含むオブジェクトの合成を適切に処理しないことである。具体的に言うと、いくつかの状況で、スキャン・ラインに沿った、オブジェクトのうちの１つまたは複数がアクティブでない位置において、重なり合うオブジェクトが不正に合成される。この実施形態でのその状況は、ポーターーダフ合成に伴って生じるので、その特定の合成方式に関して説明する。しかし、下で詳細に説明する本発明の態様は、他の合成方式の下で生じる同様の困難にも適用可能であることを諒解されたい。

【０２１７】

たとえば、式ＰＡＧＥ←（Ａ ｉｎ Ｂ）ｏｖｅｒ（Ｃ ｏｕｔ Ｄ）ｏｖｅｒ ＰＡＧＥは、図２２に示された２進木構造１０００として表すことができる。ここで、Ａ、Ｂ、ＣおよびＤは、辺、表示優先順位および不透明度データによって定義される異なるグラフィカル・オブジェクトである。この例のレベル・アクティブ化テーブル１００１を図２３に示す。図２３では、行１００２で、図２２に示された式のＣ ｏｕｔ Ｄ部分が実施されることがわかる。残念ながら、この構成を使用すると、Ｄがある画素で非アクティブの場合に、ｏｕｔ演算が、ＤではなくＰＡＧＥに対して不正に実行される。Ｃが不透明でな

い場合には、これが、レンダリング時に視覚的に不正な外見をもたらす。オブジェクトが潜在的に１未満の不透明度を有する場合のこの種の問題を回避するためには、非アクティブ・オブジェクトが、合成式内でｏｖｅｒ演算の左側以外の位置に現れてはならない。ほとんどのオブジェクトが、それらが現れるすべてのスキャン・ラインの少なくとも一部で非アクティブになるとすると、これは大問題になる可能性がある。

【０２１８】

この問題を克服する方法の１つが、透明画素を使用してオブジェクトを「パッド・アウト」することである。この形では、各オブジェクトが、効果的に、合成されるオブジェクトが存在するすべての点に存在するので、ポーターダフ式が正しく適用される。これを上の例に適用すると、２進木１０００の各葉ノード（すなわちＡ、Ｂ、ＣおよびＤ）は、図２４および図２５に示されるように、オブジェクトの対の和集合を囲む境界ボックスのサイズの透明（ガラス）オブジェクトの上に（ｏｖｅｒ）そのオブジェクトを置く式に置換される。

【０２１９】

図２４では、２つの透明なガラス境界ボックスＧ１およびＧ２が示されている。Ｇ１は、ＡおよびＢの境界ボックスを表し、Ｇ２は、ＣおよびＤの境界ボックスを表す。エクスプレッションツリー１０００でこれを実施すると、図２５に示された変更されたエクスプレッションツリー１００４がもたらされる。対応するレベル・アクティブ化テーブル１００５を図２６に示す。レベル・アクティブ化テーブル１００５と変更されたエクスプレッションツリー１００４から明白になるとおり、この変更は、各画素に必要なスタック演算の数のかなりの増加をもたらす。これは、スタック演算が、合成される画素の対の一方または両方が透明の場合に実行されるからである。最終的に達成される、実際にレンダリングされる出力に関しては、これらの演算は、スタック資源とプロセッサ時間について比較的浪費的である。

【０２２０】

演算の数を減らす方法の１つは、ガラス・オブジェクトが合成される葉ノード・オブジェクトを用いてガラス・オブジェクトをクリップ・アウトすることである。その場合、葉ノードと透明オブジェクトの間のｏｖｅｒ演算は、もはや不要になる。クリッピング動作が、一般に、複数の合成演算よりプロセッサ集中的でないという事実に鑑みて、これは、より効率的なスタックの操作をもたらす。

【０２２１】

ＡおよびＢによる透明ボックスＧ１のクリッピングとＣおよびＤによる透明ボックスＧ２のクリッピングを使用する、上の例のレベル・アクティブ化テーブル１００６を、図２７に示す。この実装では、１画素について、複数のｏｖｅｒ演算を使用してＡ、Ｂ、ＣおよびＤがパディングされる場合の１２個のスタック演算と比較して、８つのスタック演算だけが実行されることに留意されたい。これらのレベルのうちの４つは、クリッピング・レベル９９９である。しかし、それでも透明画素を用いる複数の冗長な合成演算が必要であることを諒解されたい。

【０２２２】

１つまたは複数の重なり合うオブジェクトが不透明でない場合にポーターダフ合成を正しく実施する好ましい方法を、これから説明する。図２８に示された例を参照すると、演算Ｃ　ｏｕｔ　Ｄは、領域Ｃ∩Ｄ¯、Ｃ∩ＤおよびＣ¯∩Ｄのそれぞれについて１つずつの、３つの形で表現できる。領域Ｃ∩Ｄ¯は、Ｃによって表現できる。というのは、その領域でｏｕｔ演算に関してＤからの寄与がなく、Ｃが必要だからである。領域Ｃ∩Ｄでは、ＣとＤの両方がアクティブであり、したがって、Ｃ　ｏｕｔ　Ｄ演算を使用する合成が必要である。領域Ｃ¯∩Ｄでは、ＣまたはＤからの寄与がないので、この領域は、その領域内の画素について合成スタックを構築する時に検討する必要がない。

【０２２３】

Ｃ　ｏｕｔ　Ｄ演算を簡略化した表現を、図４２および図４３に示す。図４２には、ｏｕｔ演算が実行されるオブジェクトＣおよびＤが示されている。上で説明したように、スキ

ャン・ラインは、３つの領域Ｃ∩Ｄ¯、Ｃ∩ＤおよびＣ∩Ｄ¯に分割できる。この場合では、Ｃが不透明でない場合に、領域Ｃ∩Ｄ¯内で、ｏｕｔ演算がＰＡＧＥに対して不正に実行される。

【０２２４】

図４３に、前に説明した解決策を示す。オブジェクトのそれぞれが、それがアクティブになる領域にクリッピングされていることに留意されたい。領域Ｃ∩Ｄ¯は、単にレベルＣ１によって表され、領域Ｃ∩Ｄは、レベルＣ１およびＤ１によって表され、領域Ｃ¯∩Ｄは、どちらのレベルからの入力も必要としない。必要な領域だけへのオブジェクトのクリッピングを制御するために特定の演算に固有の必要条件を使用することによって、透明のパディング画素がもはや不要になるので、かなり少ない数の合成演算がもたらされることを諒解されたい。

【０２２５】

この例のクリッピングを実施するためのレベル・アクティブ化テーブル１００８を、図２９に示す。前に説明した方法と比較して、レベル・アクティブ化テーブル１００８では、４つの追加レベルが必要であることがわかる。しかし、これら４つの追加レベルのうちの３つは、クリッピング・レベル１０１５であり、これは、通常は合成演算より集中的でない。また、各画素について作られる実際の合成スタックは、平均して、前に説明した方法の場合より小さい。これは、合成演算が、関連する領域内でのみ行われることを保証するために、クリッピング演算が使用されているという事実に起因する。要するに、プロセッサ時間は、最終的な画像に寄与しない画素の合成に浪費されない。

【０２２６】

非自明（non-trivial）ポーター－ダフ合成演算子のための合成スタック演算を、図３０および図３１に示す。図３０に示されたテーブル１０１０では、Ｄがアクティブの場合にはＤがすでにスタックの最上部にあり、合成演算子は、それが適用される領域に従って分割されると仮定する。アルファ・チャネル（不透明度）演算は、Ｓの辺上でアクティブ化し、Ｄの辺を用いてクリッピングすることによって、交差領域Ｓ∩Ｄに制限されることに留意されたい。アスタリスクでマークされた事例では、Ｄに寄与するオブジェクトのすべてが、Ｓの辺によってクリップ・インされる必要がある。

【０２２７】

図３１のテーブル１０１６は、Ｓ画素値が、それがアクティブになる領域内で、すでにスタック上にある場合の演算を提供する。そのような場合には、Ｄ画素値は、Ｄもアクティブである次の下位レベルにある。アスタリスクでマークされた事例では、スタックは不正な状態になり、その状態が発生しないようにするために「アクティブな枝」のオブジェクトを「非アクティブ」な枝のオブジェクトによってクリッピングするか、最上部の値をスタックからポップするかのいずれかが必要になる。

【０２２８】

複雑な式の場合、オブジェクトが複数の他のオブジェクトから構築され、その結果、領域ＳおよびＤの形状を、それらの構築に使用された形状から判定する必要が生じる場合がある。このより複雑な事例を実施するのに必要なステップは、ポーター－ダフ合成演算の当業者には明白である。

【０２２９】

さまざまな合成演算によって作られる結果のアクティブ領域を、図３２に示されたテーブル１０１１に示す。

【０２３０】

好ましい実施形態を使用してレンダリングのためにエクスプレッションツリーを分析する場合には、アプリケーションは、効果的に木の枝ノードのそれぞれのアクティブ領域を決定し、それを木の次の上位ノードに渡す必要がある。図２２に示されたエクスプレッションツリーの詳細な分析を、図３３ないし図４１に関して、この方針に沿って行う。

【０２３１】

図３３に、各ノードに関連する演算から生じるアクティブ領域に基づく、図２２に示され

たエクスプレッションツリー１０００の予備分析を示す。図３４を参照すると、Ａ　ｉｎ　Ｂ式が、アクティビティにおいてＡとＢの交差領域に制限されることがわかる。したがって、オブジェクトＡの辺が、オブジェクトＢのクリッピングに使用され、その結果、不要な場合にスタックに色が残されなくなる。同様に、ｉｎ演算を実行するレベルは、オブジェクトＢによる交差領域に制限される。これを達成するのに必要なスタック演算を、図３５に示す。

【０２３２】

オブジェクトＢの辺を使用して両方のレベルをアクティブ化し、オブジェクトＡを用いてこの両方のレベルをクリッピングすることによって、クリッピング・レベルを節約しながら、同一の効果を得ることができる。しかし、エクスプレッションツリーのより上位のオブジェクトが、この両方のクリッピング・オブジェクトを必要とする。

【０２３３】

次の枝は、図３６に示されるｏｖｅｒ演算であり、この図には、２つの枝のアクティブ領域が示されている。検討しなければならない領域を、図３７に示す。図３７の右側の箱に囲まれたテキスト１０１２は、左側に示される対応する領域１０１３のそれぞれのスタックの状態を示す。

【０２３４】

後者の２つの場合、ｏｖｅｒ演算の結果がすでにスタックにある。したがって、交差領域は、さらに演算が必要な領域だけになる。結果のレベル・アクティブ化テーブル項目を、図３８に示す。オブジェクトＣがレベルＯ１をアクティブ化し、ＡおよびＢが、それを領域Ａ∩Ｂ∩Ｃにクリッピングすることがわかる。この演算全体としては、領域（Ａ∩Ｂ）∪Ｃ内の画素に寄与する。

【０２３５】

この領域は、エクスプレッションツリーの次のノードに渡されるが、そのノードは、最後のｏｖｅｒであり、これによって、この式がＰＡＧＥに合成される。２つの枝のそれぞれについて検討しなければならない関連領域を図４０に示す。図４０では、やはり、右側の箱に囲まれたテキスト１０１４に、左側に示される対応するアクティブ領域１０１５のスタックの状態を示す。

【０２３６】

和集合演算の画素スタックの正しい動作を得ることは、多少の労力を必要とする。１つの解決策は、区域を相互に排他的な区域に分割し、同一の演算のために２つのレベルを使用することである。これは、前のノードの枝のそれぞれのアクティブ領域を使用して、レベルのうちの１つをアクティブ化することによって行うことができる。レベルのうちの１つのアクティブ領域は、他方のレベルのクリップ・アウトに使用される。たとえば、当業者に諒解されるように、区域Ｃと区域Ａ∩Ｂ∩Ｃ¯を使用して、（Ａ∩Ｂ）∪Ｃを作ることができる。

【０２３７】

この例の式のレベル・アクティブ化テーブルの結果の項目１０２０を、図４１に示す。レベルＯ２（オブジェクトＣによってアクティブ化される）およびＯ３（オブジェクトＡによってアクティブ化され、Ｂによってクリップ・インされ、Ｃによってクリップ・アウトされる）を組み合わせて、領域（Ａ∩Ｂ）∪Ｃに対する最終的なｏｖｅｒ演算をクリッピングする。

【０２３８】

クリッピング・レベルは、画素の合成に寄与しないことに留意されたい。寄与するレベルの数、したがって、各画素のスタック演算の数は、平均して、レベルが透明画素によってパッド・アウトされる場合の方法よりかなり少ない。また、所与のスキャン・ライン内で特定のレベルがアクティブになる可能性があると期待される画素の数も減る。

【０２３９】

多数のレベルにわたるオブジェクトの合成も、本明細書に記載の方法の外挿によって可能であることを、当業者は諒解するであろう。さらに、示されたさまざまな操作を、フレー

ムストア・ベース・システムおよび他のスタック・ベース、ライン・ベースまたはバンド・ベースの合成システムを含む異なる合成パラダイムで使用することができることも諒解されるであろう。

【０２４０】

前述から、本明細書に記載の方法および装置が、レンダリング処理中の画素画像データの中間記憶を必要とせずに、洗練されたグラフィック記述言語が必要とする機能性のすべてを備えたグラフィック・オブジェクトのレンダリングを提供することは、明白であろう。

【０２４１】

前述は、本発明の複数の実施形態を記述したのみであり、本発明の趣旨および範囲と、ここまでに付加されたさまざまな態様から逸脱せずに変更を行うことができる。

【図面の簡単な説明】

【図１】好ましい実施形態を組み込まれるコンピュータ・システムの概略ブロック図である。

【図２】好ましい実施形態の機能データ・フローを示すブロック図である。

【図３】好ましい実施形態の画素シーケンシャル・レンダリング装置と、関連する表示リストおよび一時ストアの概略ブロック図である。

【図４】図２の辺処理モジュールの概略機能図である。

【図５】図２の優先順位決定モジュールの概略機能図である。

【図６】図２の塗潰しデータ決定モジュールの概略機能図である。

【図７Ａ】ソースとデスティネーションの間の画素の組合せを示す図である。

【図７Ｂ】ソースとデスティネーションの間の画素の組合せを示す図である。

【図７Ｃ】ソースとデスティネーションの間の画素の組合せを示す図である。

【図８】好ましい実施形態の演算を説明するための例として使用される、オブジェクトが２つある画像を示す図である。

【図９Ａ】図８のオブジェクトのベクトル辺を示す図である。

【図９Ｂ】図８のオブジェクトのベクトル辺を示す図である。

【図１０】図８の画像の複数のスキャン・ラインのレンダリングを示す図である。

【図１１】図８の画像の辺レコードの配置を示す図である。

【図１２Ａ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｂ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｃ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｄ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｅ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｆ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｇ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｈ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｉ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１２Ｊ】図１０の例のために図４の配置によって実施される辺更新ルーチンを示す図である。

【図１３Ａ】奇数－偶数塗潰し規則と非ゼロ・ワインディング塗潰し規則を示す図である。

【図１３Ｂ】奇数－偶数塗潰し規則と非ゼロ・ワインディング塗潰し規則を示す図である。
【図１４Ａ】Ｘ座標の大きな変化がスピル条件にどのように寄与し、それらがどのように処理されるかを示す図である。
【図１４Ｂ】Ｘ座標の大きな変化がスピル条件にどのように寄与し、それらがどのように処理されるかを示す図である。
【図１４Ｃ】Ｘ座標の大きな変化がスピル条件にどのように寄与し、それらがどのように処理されるかを示す図である。
【図１４Ｄ】Ｘ座標の大きな変化がスピル条件にどのように寄与し、それらがどのように処理されるかを示す図である。
【図１４Ｅ】Ｘ座標の大きな変化がスピル条件にどのように寄与し、それらがどのように処理されるかを示す図である。
【図１５Ａ】図５の配置によって実施される優先順位更新ルーチンを示す図である。
【図１５Ｂ】図５の配置によって実施される優先順位更新ルーチンを示す図である。
【図１５Ｃ】図５の配置によって実施される優先順位更新ルーチンを示す図である。
【図１５Ｄ】図５の配置によって実施される優先順位更新ルーチンを示す図である。
【図１５Ｅ】図５の配置によって実施される優先順位更新ルーチンを示す図である。
【図１６Ａ】２つの従来技術の辺記述フォーマットと、好ましい実施形態で使用される辺記述フォーマットを比較するための図である。
【図１６Ｂ】２つの従来技術の辺記述フォーマットと、好ましい実施形態で使用される辺記述フォーマットを比較するための図である。
【図１６Ｃ】２つの従来技術の辺記述フォーマットと、好ましい実施形態で使用される辺記述フォーマットを比較するための図である。
【図１７Ａ】エクスプレッションツリーとして示される単純な合成式と対応する図を示す図である。
【図１７Ｂ】エクスプレッションツリーとして示される単純な合成式と対応する図を示す図である。
【図１８Ａ】正確なスタック演算を保証するために構成されたテーブルを示す図である。
【図１８Ｂ】図１８Ａのテーブルの好ましい形態を示す図である。
【図１９】例のエクスプレッションツリーを示す図である。
【図２０Ａ】図１９の式のアクティビティ・テーブル評価と、そのような評価の間の対応する合成スタックを示す図である。
【図２０Ｂ】図１９の式のアクティビティ・テーブル評価と、そのような評価の間の対応する合成スタックを示す図である。
【図２０Ｃ】図１９の式のアクティビティ・テーブル評価と、そのような評価の間の対応する合成スタックを示す図である。
【図２０Ｄ】図１９の式のアクティビティ・テーブル評価と、そのような評価の間の対応する合成スタックを示す図である。
【図２０Ｅ】図１９の式のアクティビティ・テーブル評価と、そのような評価の間の対応する合成スタックを示す図である。
【図２０Ｆ】図１９の式のアクティビティ・テーブル評価と、そのような評価の間の対応する合成スタックを示す図である。
【図２０Ｇ】アクティビティ・テーブルと関連モジュールのさまざまな構成を示す図である。
【図２０Ｈ】アクティビティ・テーブルと関連モジュールのさまざまな構成を示す図である。
【図２０Ｉ】アクティビティ・テーブルと関連モジュールのさまざまな構成を示す図である。
【図２１】複数の合成演算の結果を示す図である。
【図２２】オブジェクトＡ、Ｂ、ＣおよびＤに対して一連のポーター－ダフ合成演算を実

施するためのエクスプレッションツリーを示す図である。
【図２３】図２２に示された２進木構造を実施するためのレベル・アクティブ化テーブルを示す図である。
【図２４】透明なガラス・オブジェクトの上に置かれた、図２２に示された２進木のオブジェクトを示す図である。
【図２５】図２４に示された配置を実施するための、変更されたエクスプレッションツリーを示す図である。
【図２６】図２５のエクスプレッションツリーを実施するためのレベル・アクティブ化テーブルを示す図である。
【図２７Ａ】透明な箱がＡ、Ｂ、ＣおよびＤによってクリッピングされる、図２４および図２５に示された例の代替のレベル・アクティブ化テーブルを示す図である。
【図２７Ｂ】透明な箱がＡ、Ｂ、ＣおよびＤによってクリッピングされる、図２４および図２５に示された例の代替のレベル・アクティブ化テーブルを示す図である。
【図２８】演算Ｃ　ｏｕｔ　Ｄのエクスプレッションツリーを示す図である。
【図２９Ａ】図２８に示されたクリッピングを実施するためのレベル・アクティブ化テーブルを示す図である。
【図２９Ｂ】図２８に示されたクリッピングを実施するためのレベル・アクティブ化テーブルを示す図である。
【図３０】自明でないポーター－ダフ合成演算子のための合成スタック演算を示す図である。
【図３１】自明でないポーター－ダフ合成演算子のための合成スタック演算を示す図である。
【図３２】さまざまな演算子のアクティブ領域を示す図である。
【図３３】式の実施に使用されるステップの予備分析を含む、図２２に示されたものに類似のエクスプレッションツリーを示す図である。
【図３４】図３３に示されたエクスプレッションツリーからのＡ　ｉｎ　Ｂ式のアクティブ領域を示す図である。
【図３５】図３４に示されたｉｎ演算を実行するのに必要なスタック演算を示す図である。
【図３６】下のｉｎ演算およびｏｕｔ演算からのアクティブ領域を示す、図３３のｏｖｅｒ演算の詳細を示す図である。
【図３７】図３６の式に関連して検討される領域のそれぞれのスタックの状態を示す図である。
【図３８】図３６および図３７に示された演算を実施するためのレベル・アクティブ化テーブル項目を示す図である。
【図３９】下のｏｖｅｒ式およびｐａｇｅ式からのアクティブ領域を示す、図３３のエクスプレッションツリーからの最終的なｏｖｅｒ演算を示す図である。
【図４０】図３９の式で検討される領域を示す図である。
【図４１Ａ】図３９に示された式を実施するためのレベル・アクティブ化テーブルの結果の項目を示す図である。
【図４１Ｂ】図３９に示された式を実施するためのレベル・アクティブ化テーブルの結果の項目を示す図である。
【図４２】アクティブ領域へのクリッピングの実施の前のＣ　ｏｕｔ　Ｄ演算を示す図である。
【図４３】個々のレベルのアクティブ領域へのクリッピングの後のＣ　ｏｕｔ　Ｄ演算を示す図である。

【図１】

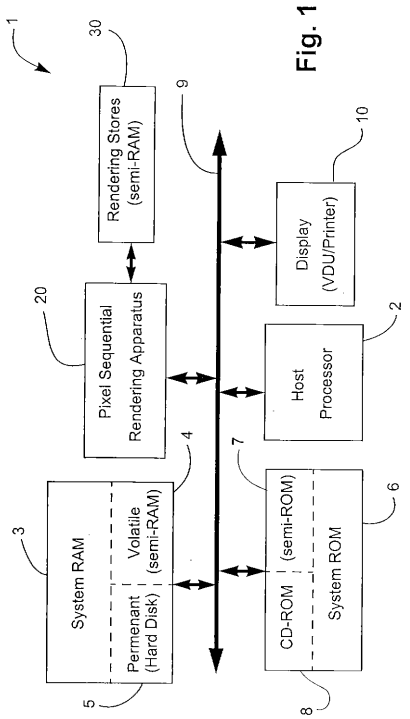


Fig. 1

【図２】

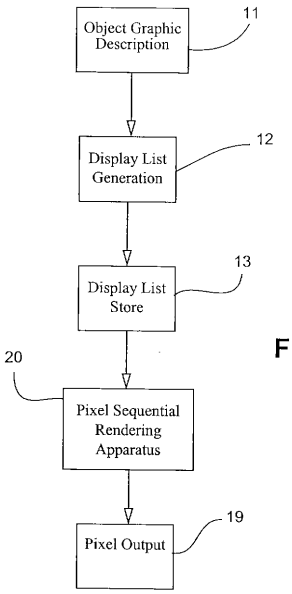


Fig. 2

【図３】

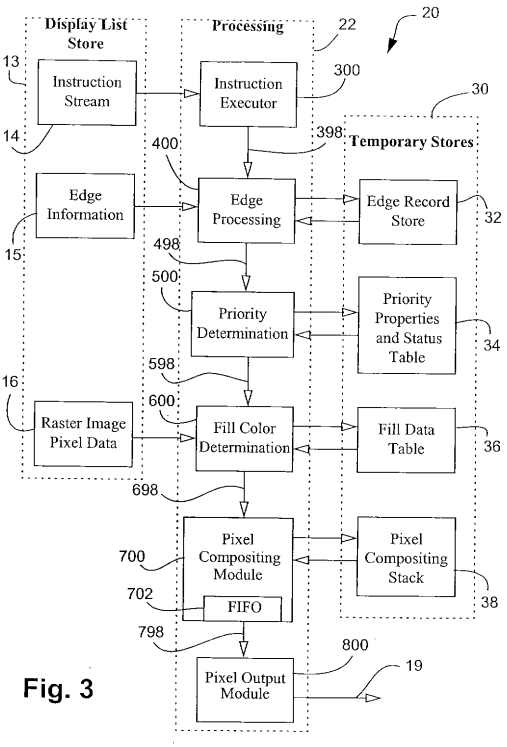


Fig. 3

【図４】

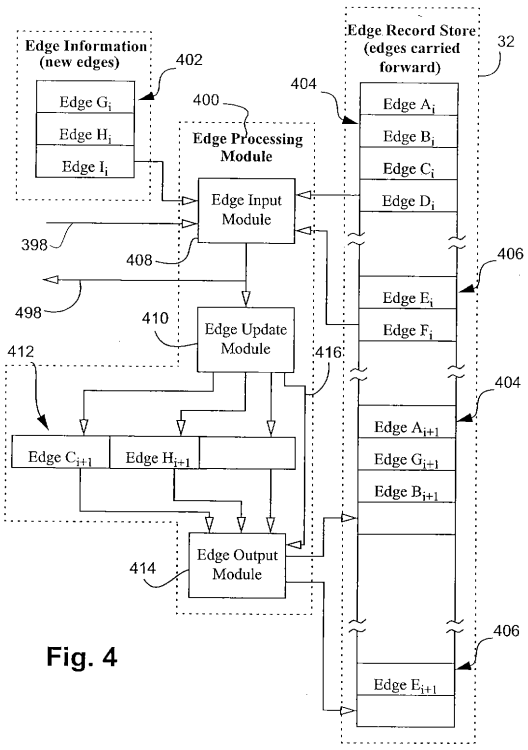


Fig. 4

【図５】

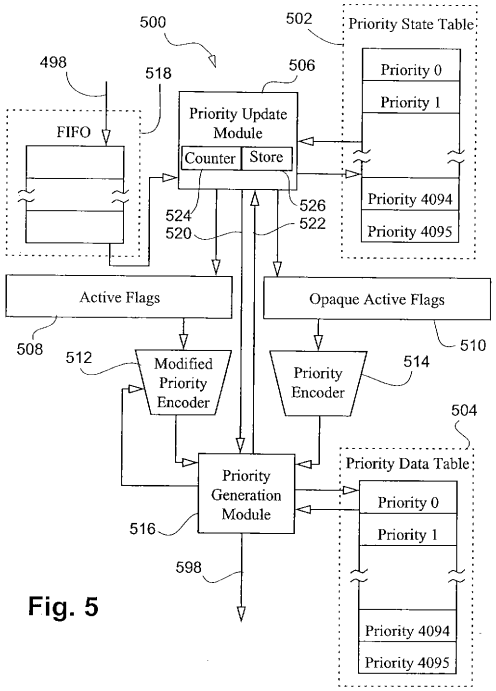


Fig. 5

【図６】

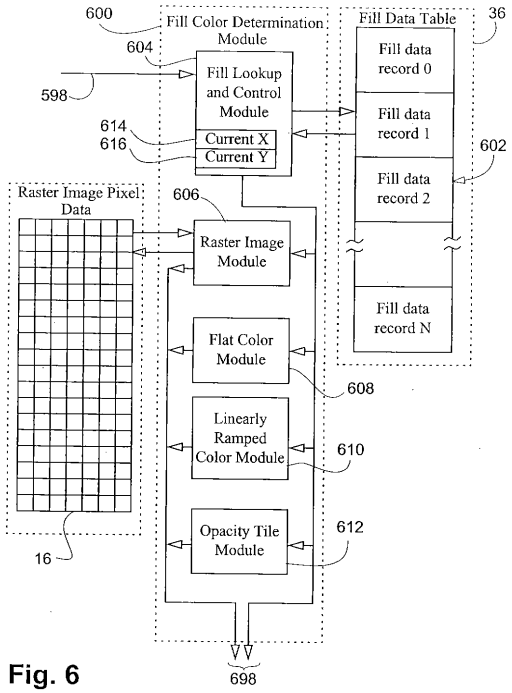


Fig. 6

【図７Ａ】

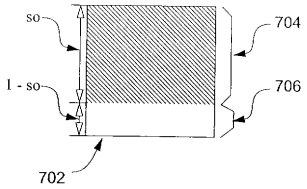


Fig. 7A

【図７Ｂ】

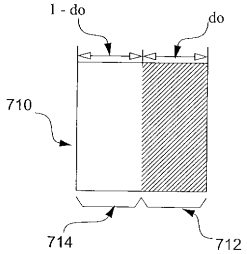


Fig. 7B

【図７Ｃ】

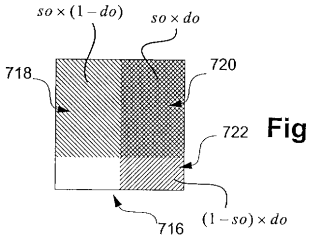


Fig. 7C

【図８】

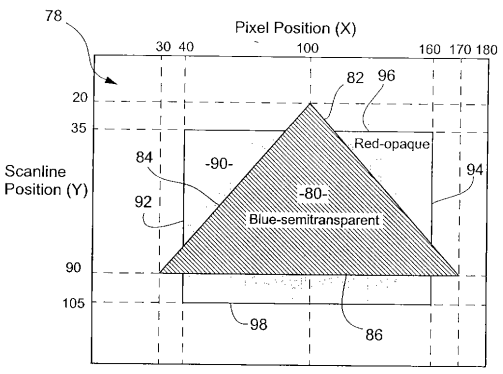


Fig. 8

【図９Ａ】

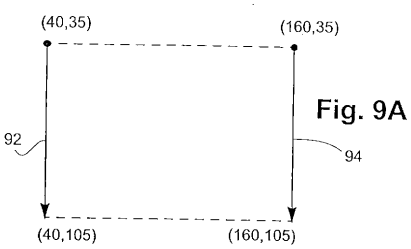


Fig. 9A

【図９Ｂ】

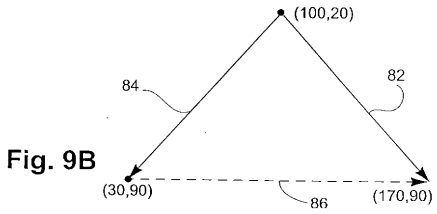


Fig. 9B

【図１０】

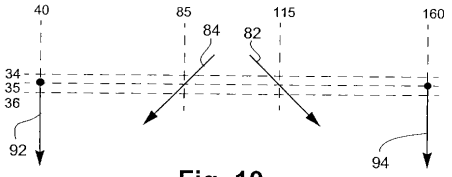


Fig. 10

【図１１】

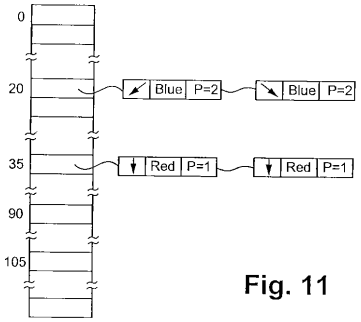


Fig. 11

【図１２Ａ】

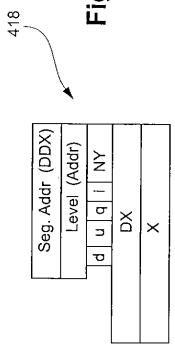


Fig. 12A

【図１２Ｂ】

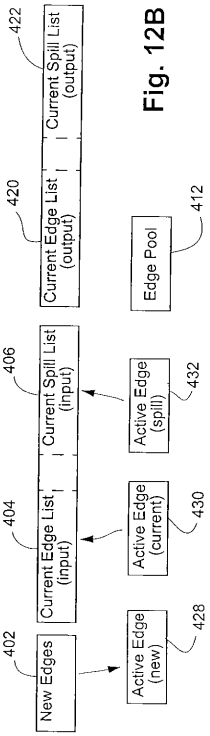
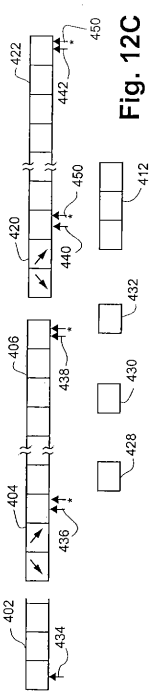
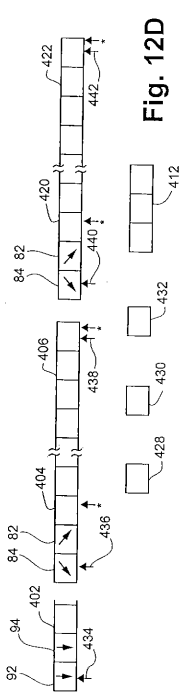


Fig. 12B

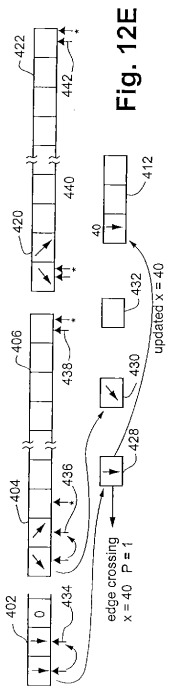
【図１２Ｃ】



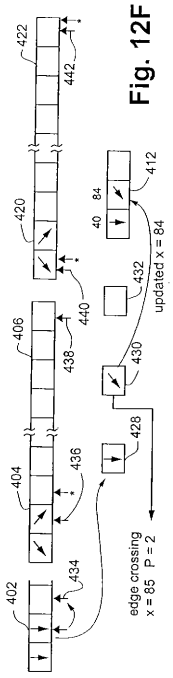
【図１２Ｄ】



【図１２Ｅ】



【図１２Ｆ】



【図１２Ｇ】

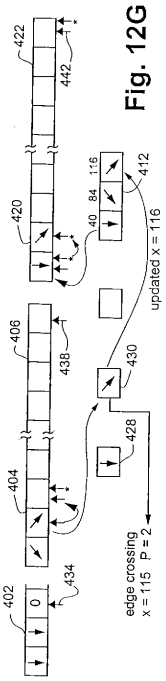


Fig. 12G

【図１２Ｈ】

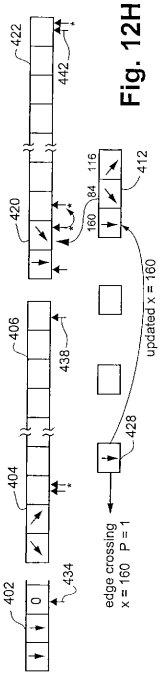


Fig. 12H

【図１２Ｉ】

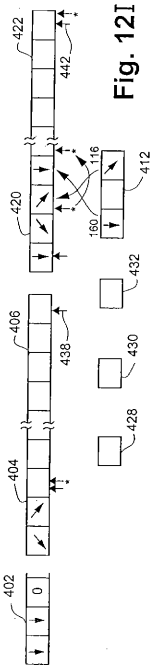


Fig. 12I

【図１２Ｊ】

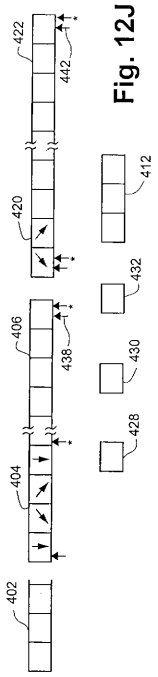


Fig. 12J

【図１３Ａ】

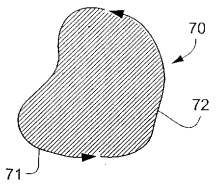


Fig. 13A

【図１３Ｂ】

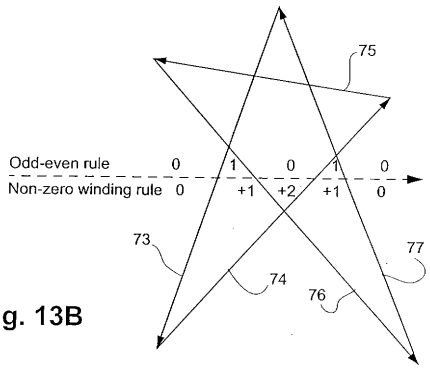


Fig. 13B

【図１４Ａ】

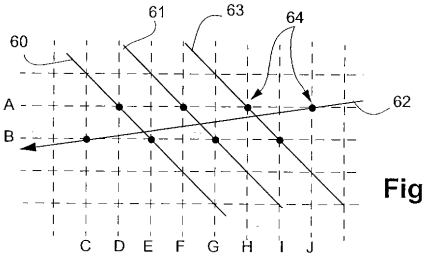


Fig. 14A

【図１４Ｂ】

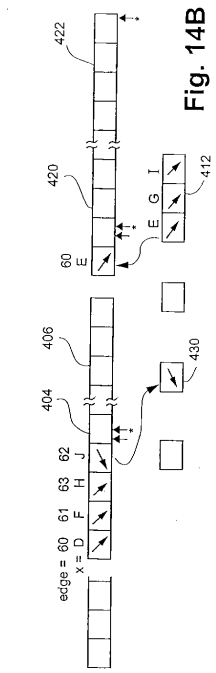


Fig. 14B

【図１４Ｃ】

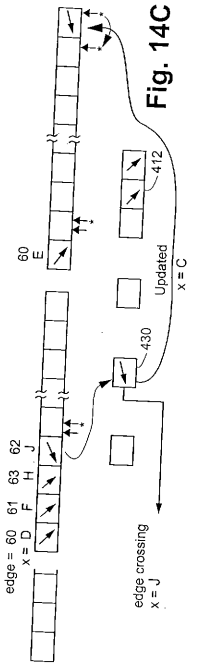
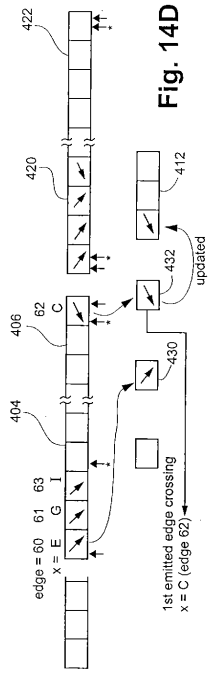
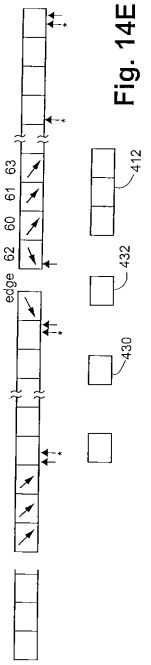


Fig. 14C

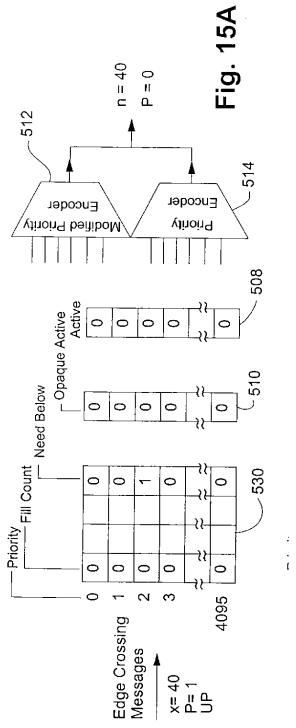
【図１４Ｄ】



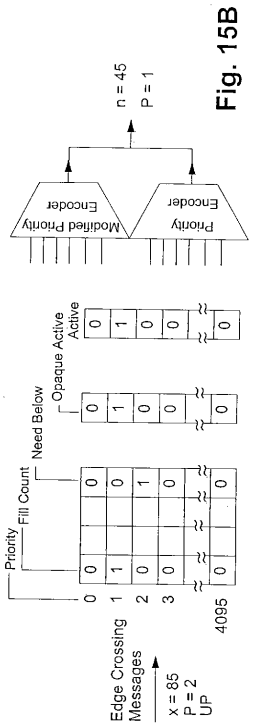
【図１４Ｅ】



【図１５Ａ】



【図１５Ｂ】



【図１５Ｃ】

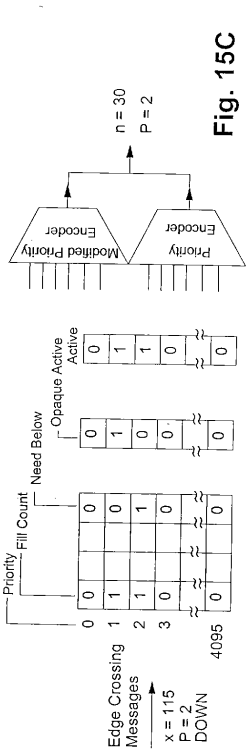


Fig. 15C

【図１５Ｄ】

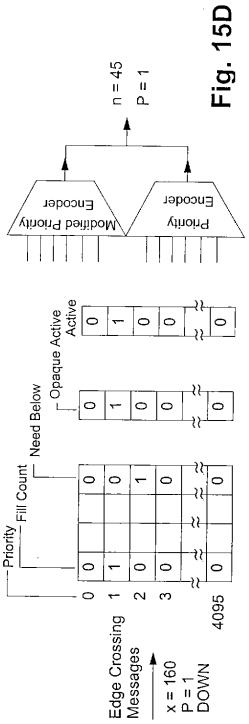


Fig. 15D

【図１５Ｅ】

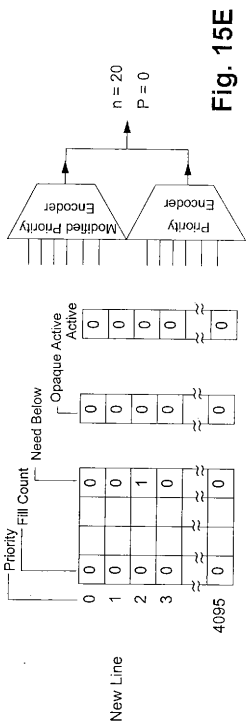


Fig. 15E

【図１６Ａ】

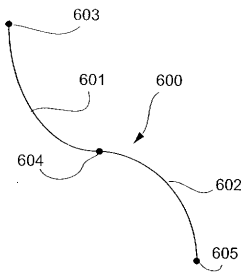


Fig. 16A
(Prior Art)

【図１６Ｂ】

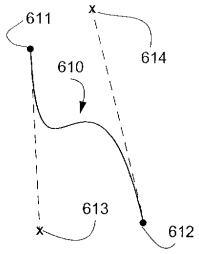
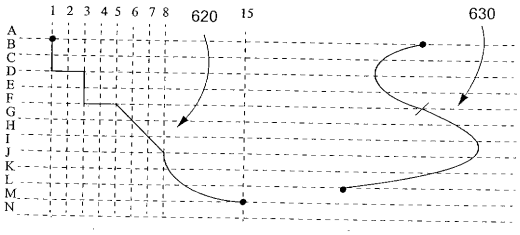


Fig. 16B
(Prior Art)

【図１６Ｃ】



【図１７Ａ】

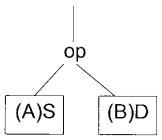


Fig. 17A

【図１７Ｂ】

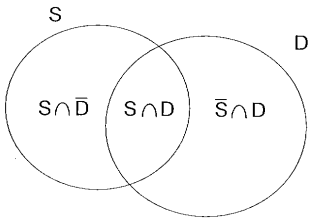


Fig. 17B

【図１８Ａ】

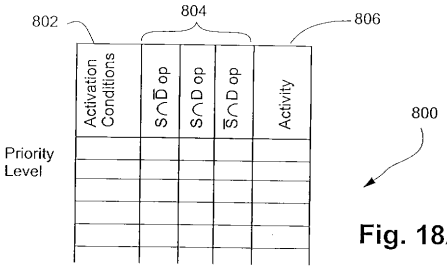


Fig. 18A

【図１８Ｂ】

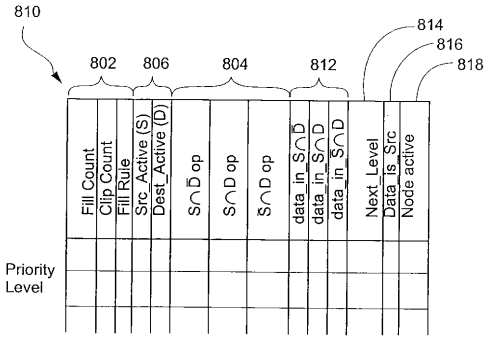
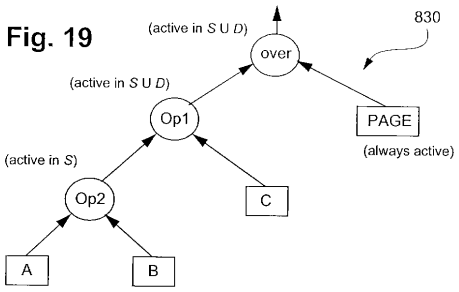


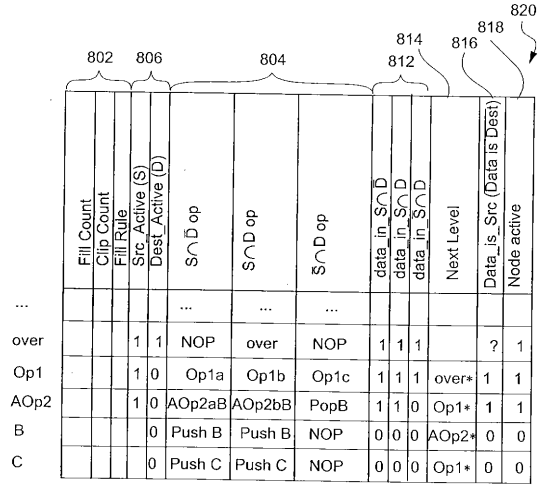
Fig. 18B

【図１９】

Fig. 19



【図２０Ａ】



	Fill Count	Clip Count	Fill Rule	Src_Active (S)	Dest_Active (D)	S∩D̄ op	S∩D op	S̄∩D op	data_in_S∩D̄	data_in_S∩D	data_in_S̄∩D	Next Level	Data_is_Src (Data is Dest)	Node active
...						...	...	...						
over				1	1	NOP	over	NOP	1	1	1		?	1
Op1				1	0	Op1a	Op1b	Op1c	1	1	1	over*	1	1
AOp2				1	0	AOp2aB	AOp2bB	PopB	1	1	0	Op1*	1	1
B					0	Push B	Push B	NOP	0	0	0	AOp2*	0	0
C					0	Push C	Push C	NOP	0	0	0	Op1*	0	0

Fig. 20A

【図２０Ｂ】

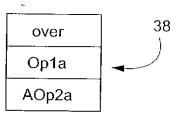
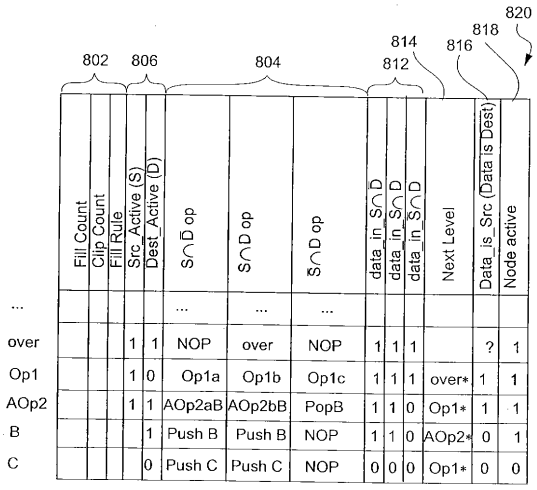


Fig. 20B

【図２０Ｃ】



	Fill Count	Clip Count	Fill Rule	Src_Active (S)	Dest_Active (D)	S∩D̄ op	S∩D op	S̄∩D op	data_in_S∩D̄	data_in_S∩D	data_in_S̄∩D	Next Level	Data_is_Src (Data is Dest)	Node active
...						...	...	...						
over				1	1	NOP	over	NOP	1	1	1		?	1
Op1				1	0	Op1a	Op1b	Op1c	1	1	1	over*	1	1
AOp2				1	1	AOp2aB	AOp2bB	PopB	1	1	0	Op1*	1	1
B					1	Push B	Push B	NOP	1	1	0	AOp2*	0	1
C					0	Push C	Push C	NOP	0	0	0	Op1*	0	0

Fig. 20C

【図２０Ｄ】

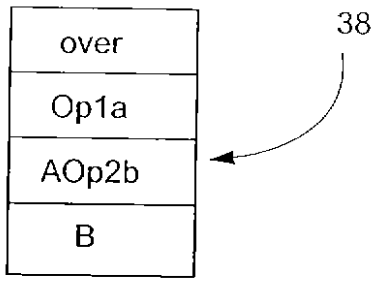
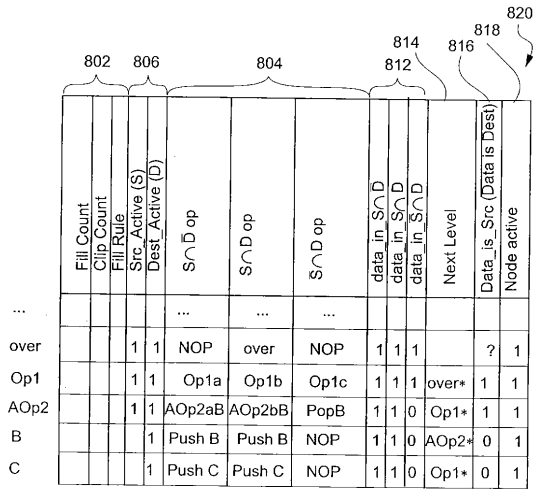


Fig. 20D

【図２０Ｅ】



	Fill Count	Clip Count	Fill Rule	Src_Active (S)	Dest_Active (D)	S∩D̄ op	S∩D op	S̄∩D op	data_in_S∩D̄	data_in_S∩D	data_in_S̄∩D	Next Level	Data_is_Src (Data is Dest)	Node active
...						...	...	...						
over				1	1	NOP	over	NOP	1	1	1		?	1
Op1				1	1	Op1a	Op1b	Op1c	1	1	1	over*	1	1
AOp2				1	1	AOp2aB	AOp2bB	PopB	1	1	0	Op1*	1	1
B					1	Push B	Push B	NOP	1	1	0	AOp2*	0	1
C					1	Push C	Push C	NOP	1	1	0	Op1*	0	1

Fig. 20E

【図２０Ｆ】

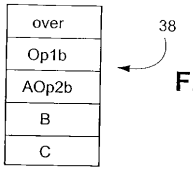


Fig. 20F

【図２０Ｇ】

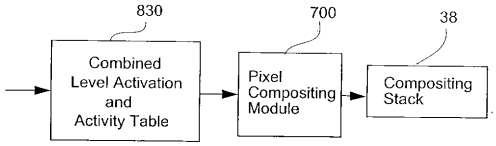


Fig. 20G

【図２０Ｈ】

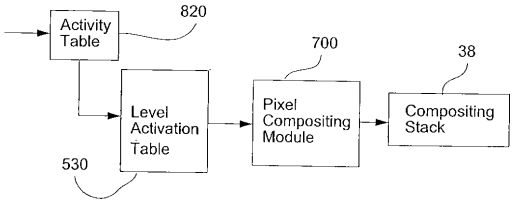


Fig. 20H

【図２０Ｉ】

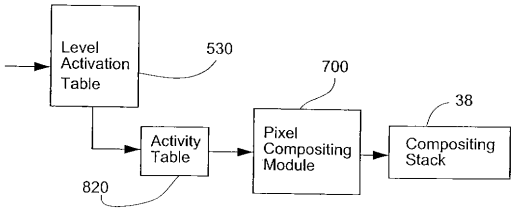


Fig. 20I

【図２１】

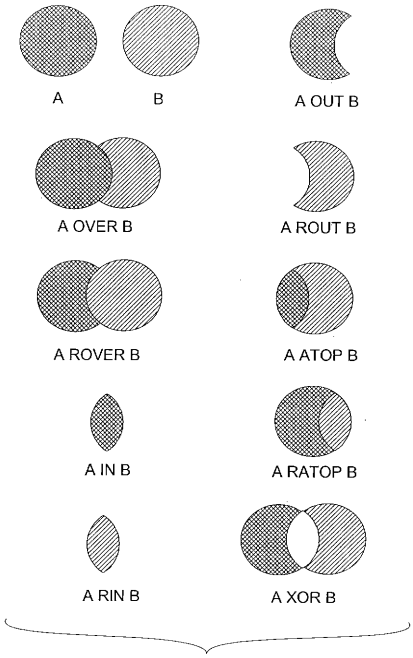


Fig. 21

【図２２】

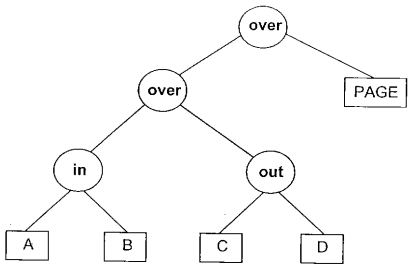


Fig. 22

【図２３】

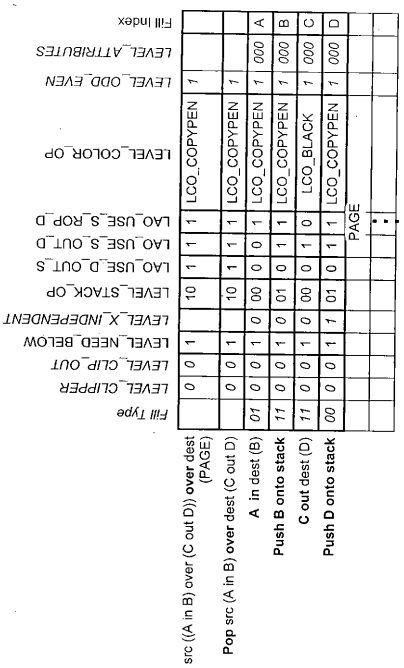


Fig. 23

【図２４】

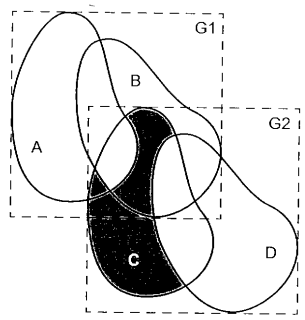


Fig. 24

【図２５】

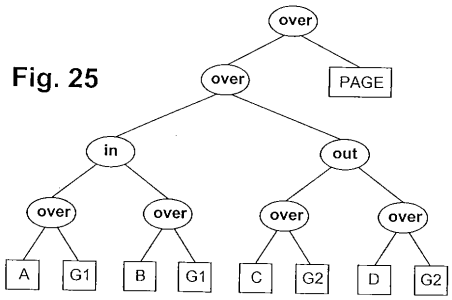


Fig. 25

【図２６】

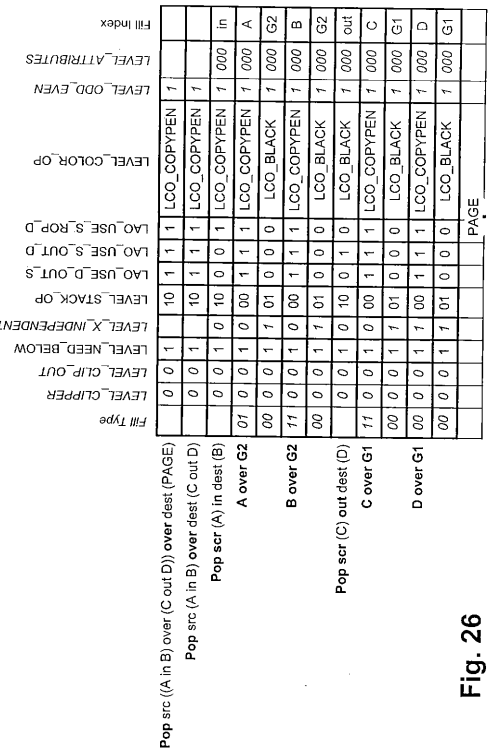


Fig. 26

【図２７Ａ】

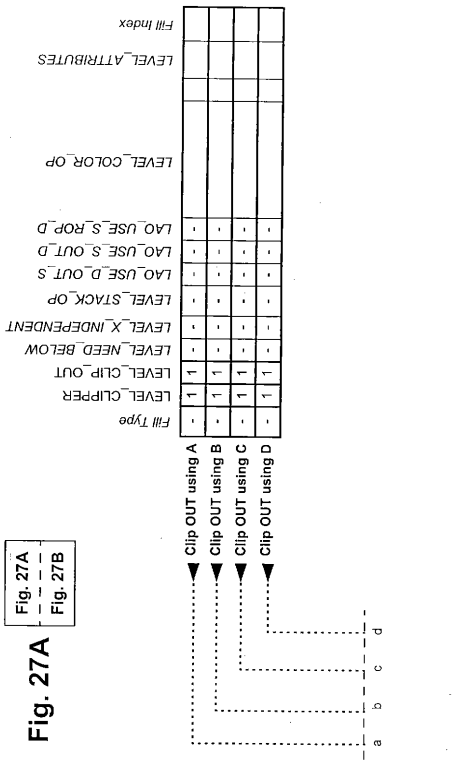
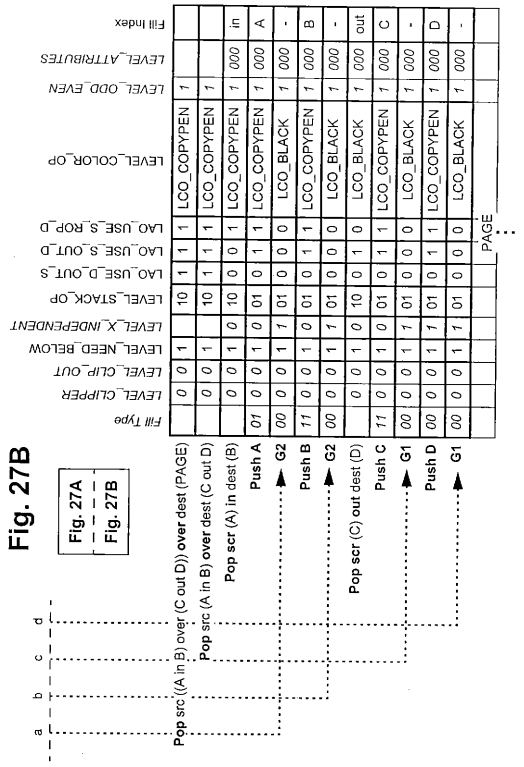
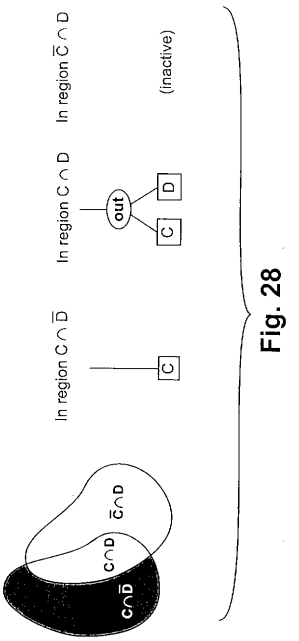


Fig. 27A

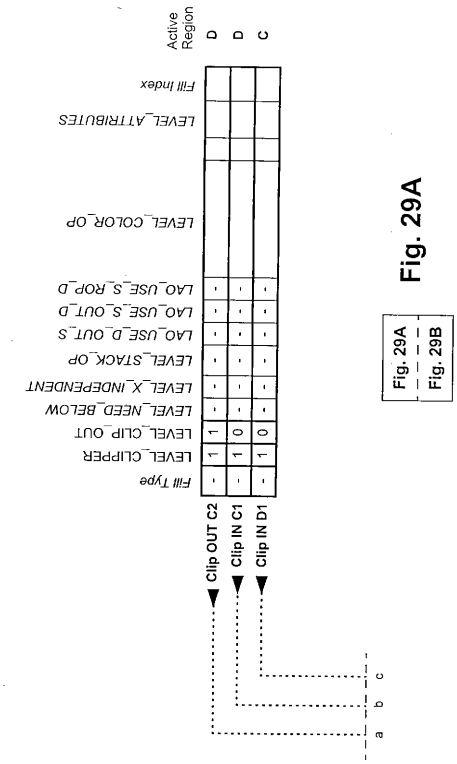
【図２７Ｂ】



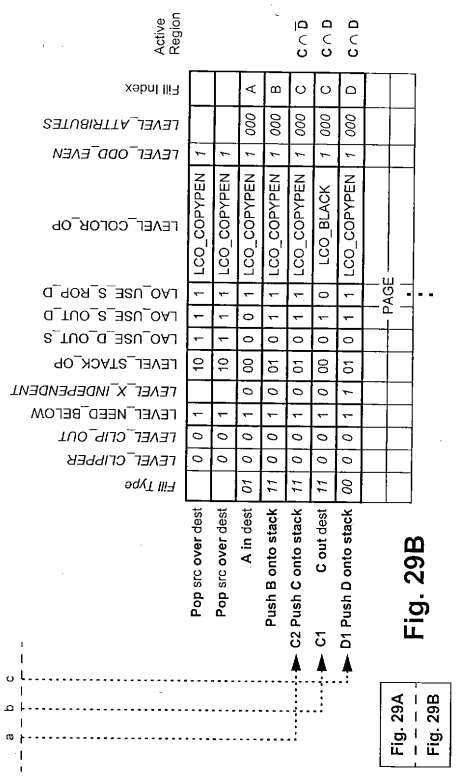
【図２８】



【図２９Ａ】



【図２９Ｂ】



【図３０】

Operator	Region	Compositing Stack Operations: LEVEL_COLOR_OP	S_ROP_D	S_OUT_D	D_OUT_S	Fill Object
S **over** D	$\bar{S}\cap D$	Result is on the Stack				
	$S\cap D$	LCO_COPYPEN	1	1	1	S
	$S\cap\bar{D}$	LCO_COPYPEN	1	1	0	S
S **rover** D	$\bar{S}\cap D$	Result is on the Stack				
	$S\cap D$	LCO_NOP	1	1	1	S
	$S\cap\bar{D}$	LCO_COPYPEN	1	1	0	S
S **in** D	$\bar{S}\cap D$	* Clip D with edges of S				
	$S\cap D$	LCO_COPYPEN	1	0	0	S
	$S\cap\bar{D}$	No Operation				
S **rin** D	$\bar{S}\cap D$	* Clip D with edges of S				
	$S\cap D$	LCO_NOP	1	0	0	S
	$S\cap\bar{D}$	No Operation				
S **out** D	$\bar{S}\cap D$	* Clip D with edges of S				
	$S\cap D$	any	0	1	0	S
	$S\cap\bar{D}$	LCO_COPYPEN	1	1	0	S
S **rout** D	$\bar{S}\cap D$	Result is on the Stack				
	$S\cap D$	any	0	0	1	S
	$S\cap\bar{D}$	No Operation				
S **atop** D	$\bar{S}\cap D$	Result is on the Stack				
	$S\cap D$	LCO_COPYPEN	1	0	1	S
	$S\cap\bar{D}$	No Operation				
S **ratop** D	$\bar{S}\cap D$	* Clip D with edges of S				
	$S\cap D$	LCO_NOP	1	1	0	S
	$S\cap\bar{D}$	LCO_COPYPEN	1	1	0	S
S **xor** D	$\bar{S}\cap D$	Result is on the Stack				
	$S\cap D$	any	0	1	1	S
	$S\cap\bar{D}$	LCO_COPYPEN	1	1	0	S

Fig. 30

【図３１】

Operator	Region	Compositing Stack Operations: STACK_OP	LEVEL_COLOR_OP	S_ROP_D	S_OUT_D	D_OUT_S
S **over** D	$\bar{S}\cap D$	Result on Stack				
	$S\cap D$	10	LCO_COPYPEN	1	1	1
	$S\cap\bar{D}$	Result on stack				
S **rover** D	$\bar{S}\cap D$	Result on stack				
	$S\cap D$	10	LCO_NOP	1	1	1
	$S\cap\bar{D}$	Result on Stack				
S **in** D	$\bar{S}\cap D$	* Clip D with S edges				
	$S\cap D$	10	LCO_COPYPEN	1	0	0
	$S\cap\bar{D}$	* Clip S with D edges				
S **rin** D	$\bar{S}\cap D$	* Clip D with S edges				
	$S\cap D$	10	LCO_NOP	1	0	0
	$S\cap\bar{D}$	* Clip S with D edges				
S **out** D	$\bar{S}\cap D$	* Clip D with S edges				
	$S\cap D$	10	any	0	1	0
	$S\cap\bar{D}$	Result on stack				
S **rout** D	$\bar{S}\cap D$	Result on stack				
	$S\cap D$	10	any	0	0	1
	$S\cap\bar{D}$	* Clip S with D edges				
S **atop** D	$\bar{S}\cap D$	Result on stack				
	$S\cap D$	10	LCO_COPYPEN	1	0	1
	$S\cap\bar{D}$	* Clip S with D edges				
S **ratop** D	$\bar{S}\cap D$	* Clip D with S edges				
	$S\cap D$	10	LCO_NOP	1	1	0
	$S\cap\bar{D}$	Result on stack				
S **xor** D	$\bar{S}\cap D$	Result on stack				
	$S\cap D$	10	any	0	1	1
	$S\cap\bar{D}$	Result on stack				

Fig. 31

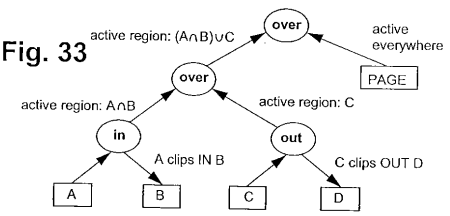
【図３２】

Fig. 32

Operator	Active Region
S **over** D	$S\cup D$
S **rover** D	$S\cup D$
S **in** D	$S\cap D$
S **rin** D	$S\cap D$
S **out** D	S
S **rout** D	D
S **atop** D	D
S **ratop** D	S
S **xor** D	$S\cup D$

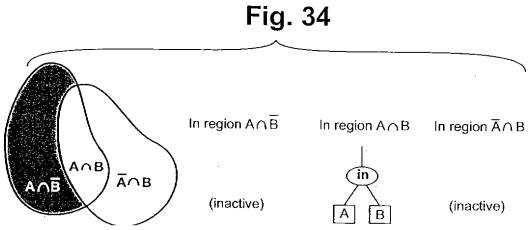
【図３３】

Fig. 33



【図３４】

Fig. 34



【図３５】

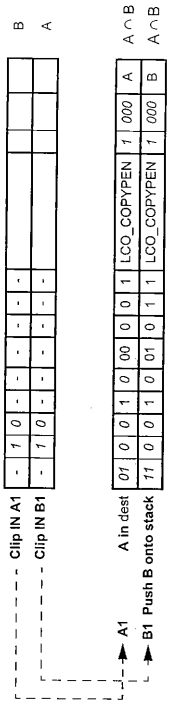


Fig. 35

【図３６】

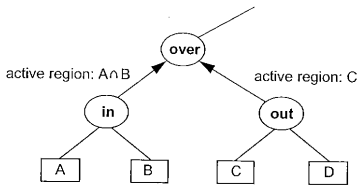


Fig. 36

【図３７】

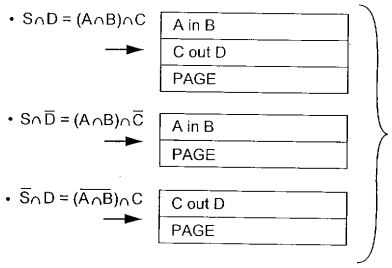


Fig. 37

【図３８】

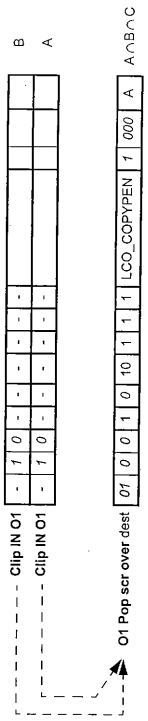


Fig. 38

【図３９】

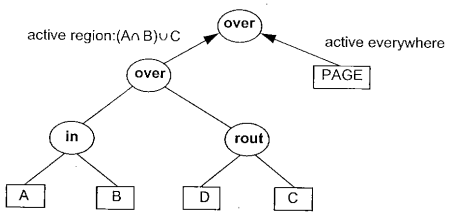


Fig. 39

【図４０】

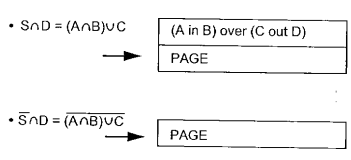


Fig. 40

【図４１Ａ】

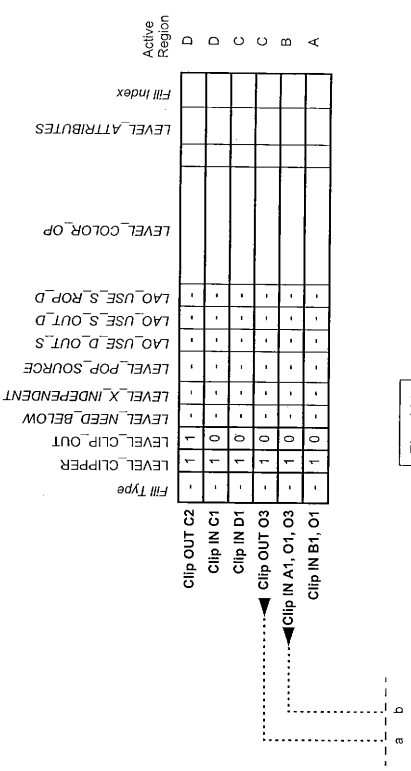


Fig. 41A

Fig. 41A
Fig. 41B

【図４１Ｂ】

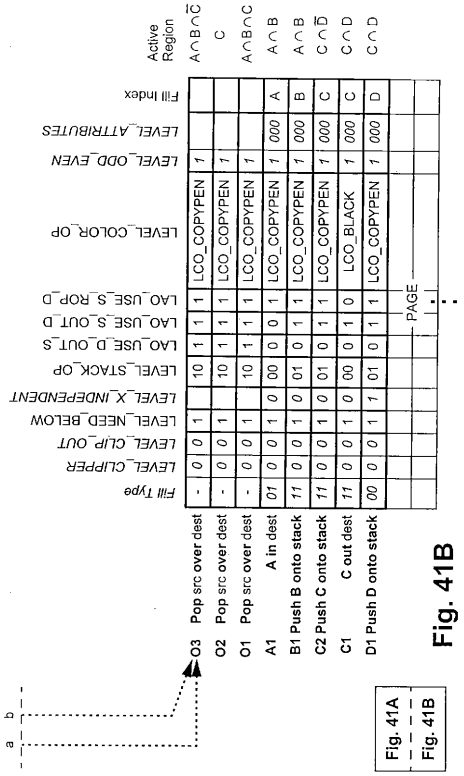


Fig. 41B

Fig. 41A
Fig. 41B

【図４２】

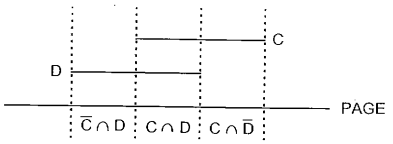


Fig. 42

【図４３】

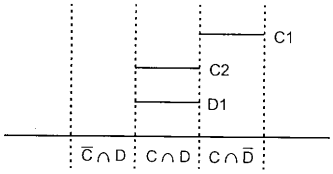


Fig. 43

フロントページの続き

(31)優先権主張番号 PP5862
(32)優先日 平成10年9月11日(1998.9.11)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 PP9234
(32)優先日 平成11年3月16日(1999.3.16)
(33)優先権主張国 オーストラリア(AU)
(31)優先権主張番号 PQ0049
(32)優先日 平成11年4月29日(1999.4.29)
(33)優先権主張国 オーストラリア(AU)

(72)発明者 ティモシー メリック ロング
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 リミテッド 内
(72)発明者 クリストファー フレイザー
オーストラリア国 ２１１３ ニューサウス ウェールズ州、 ノース ライド、 トーマス ホルト ドライブ １、 キヤノン インフォメーション システムズ リサーチ オーストラリア プロプライエタリー リミテッド 内
(72)発明者 ケビン ムーア
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 リミテッド 内

審査官 伊知地 和之

(56)参考文献 特開昭６３－２８０３８８（ＪＰ，Ａ）
特開平０９－０１６７４５（ＪＰ，Ａ）
特開平０４－２８２７８４（ＪＰ，Ａ）
特開平１１－２３２４７３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06T 11/00 - 11/80
G09G 5/00 - 5/36
CSDB(日本国特許庁)