(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6419827号
(P6419827)
(45)発行日　平成30年11月7日(2018.11.7)　(24)登録日　平成30年10月19日(2018.10.19)

(51)Int.Cl.　F I
G06T 11/20 (2006.01)　G06T 11/20 300

請求項の数 22 (全 51 頁)

(21)出願番号	特願2016-538916(P2016-538916)
(86)(22)出願日	平成26年7月2日(2014.7.2)
(65)公表番号	特表2016-532215(P2016-532215A)
(43)公表日	平成28年10月13日(2016.10.13)
(86)国際出願番号	PCT/US2014/045309
(87)国際公開番号	WO2015/030933
(87)国際公開日	平成27年3月5日(2015.3.5)
審査請求日	平成29年6月7日(2017.6.7)
(31)優先権主張番号	61/871,260
(32)優先日	平成25年8月28日(2013.8.28)
(33)優先権主張国	米国(US)
(31)優先権主張番号	14/321,409
(32)優先日	平成26年7月1日(2014.7.1)
(33)優先権主張国	米国(US)

(73)特許権者　595020643
クゥアルコム・インコーポレイテッド
ＱＵＡＬＣＯＭＭ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100109830
弁理士　福原　淑弘
(74)代理人　100158805
弁理士　井関　守三
(74)代理人　100194814
弁理士　奥村　元宏

最終頁に続く

(54)【発明の名称】グラフィックス処理におけるプレフィックス総和長さ

(57)【特許請求の範囲】
【請求項１】
破線をレンダリングする方法であって、
グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントより前のセグメントをピクセルシェーディングすることと、ここで、各セグメントは長さを有し、
前記グラフィックス処理ユニット（ＧＰＵ）を用いて、前記破線の前記現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前の前記セグメントの前記長さの累積に基づく、
前記現在のセグメントのためのテクスチャロケーションを決定するために、前記テクスチャオフセットを適用することと、
前記決定されたテクスチャロケーションにおいてテクスチャが適用される前記現在のセグメントをピクセルシェーディングすることと
を備える方法。
【請求項２】
破線をレンダリングする方法であって、
グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントより前のセグメントをピクセルシェーディングすることと、ここで、各セグメントは長さを有し、

前記グラフィックス処理ユニット（ＧＰＵ）を用いて、前記破線の前記現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前の前記セグメントの前記長さの累積に基づく、

前記現在のセグメントのためのテクスチャロケーションを決定するために、前記テクスチャオフセットを適用することと、

前記決定されたテクスチャロケーションにおいてテクスチャが適用される前記現在のセグメントをピクセルシェーディングすることと

を備え、

前記テクスチャロケーションは前記現在のセグメントのための開始ロケーションを含み、前記長さの蓄積は、前記複数の順序付きセグメントのジオメトリシェーディング中に実行される、方法。

【請求項３】

前記複数の順序付きセグメントが、１つまたは複数の可視セグメントと１つまたは複数の不可視セグメントとを含み、前記方法が、

前記現在のセグメントの前記決定されたテクスチャロケーションに基づいて、前記現在のセグメントが可視セグメントであるかどうかを決定することと、

前記現在のセグメントが可視セグメントであることに基づいて前記現在のセグメントを保持する、または、前記現在のセグメントが可視セグメントではないことに基づいて前記現在のセグメントを破棄することと

をさらに備える、請求項１または２に記載の方法。

【請求項４】

前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線に対してジオメトリシェーディングを行うことと、

長さの前記累積に基づいて前記テクスチャオフセットを決定することが前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように、前記長さ値を決定することと

をさらに備える、請求項１に記載の方法。

【請求項５】

前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線に対して前記ジオメトリシェーディングを行うことと、

長さの前記累積に基づいて前記テクスチャオフセットを決定することが前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように、前記長さ値を決定することと

をさらに備える、請求項２に記載の方法。

【請求項６】

前記長さ値を決定することが、前記順序が前のセグメントの長さを指定するｌｉｎｅｌｅｎｇｔｈスカラー値を生成することを備える、請求項４または５に記載の方法。

【請求項７】

前記複数の順序付きセグメントの前記セグメントの各々をラスタライズすること

をさらに備え、

ここにおいて、前記テクスチャオフセットを適用することが、前記現在のセグメントがラスタライズされた後に前記ラスタライズされた現在のセグメントに前記テクスチャオフセットを適用することを備える、請求項１または２に記載の方法。

【請求項８】

前記テクスチャオフセットを適用することが、前記ロケーションを示す、前記現在のセグメントのテクスチャ座標値を決定することを備える、請求項１または２に記載の方法。

【請求項９】

前記ピクセルシェーディングが、前記破線をストロークすることを含む、前記破線のためのパスレンダリングプロセス中に含まれる、請求項１または２に記載の方法。
【請求項１０】
前記複数の順序付きセグメントの前記順序がプリミティブ順序であるように、前記複数の順序付きセグメントのジオメトリシェーディング中に前記複数の順序付きセグメントの前記順序を決定することをさらに備える、請求項１に記載の方法。
【請求項１１】
前記複数の順序付きセグメントの前記順序がプリミティブ順序であるように、前記複数の順序付きセグメントのジオメトリシェーディング中に前記複数の順序付きセグメントの前記順序を決定することをさらに備える、請求項２に記載の方法。
【請求項１２】
グラフィックスデータをレンダリングするための装置であって、
グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントより前のセグメントをピクセルシェーディングするための手段と、ここで、各セグメントは長さを有し、
前記グラフィックス処理ユニット（ＧＰＵ）を用いて、前記破線の前記現在のセグメントのためのテクスチャオフセットを決定するための手段と、ここにおいて、前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前の前記セグメントの長さの累積に基づく、
前記現在のセグメントのためのテクスチャロケーションを決定するために、前記テクスチャオフセットを適用するための手段と、
前記決定されたテクスチャロケーションにおいてテクスチャが適用される前記現在のセグメントをピクセルシェーディングするための手段と
を備える装置。
【請求項１３】
グラフィックスデータをレンダリングするための装置であって、
グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントより前のセグメントをピクセルシェーディングするための手段と、ここで、各セグメントは長さを有し、
前記グラフィックス処理ユニット（ＧＰＵ）を用いて、前記破線の前記現在のセグメントのためのテクスチャオフセットを決定するための手段と、ここにおいて、前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前の前記セグメントの長さの累積に基づく、
前記現在のセグメントのためのテクスチャロケーションを決定するために、前記テクスチャオフセットを適用するための手段と、
前記決定されたテクスチャロケーションにおいてテクスチャが適用される前記現在のセグメントをピクセルシェーディングするための手段と
を備え、
前記テクスチャロケーションは前記現在のセグメントのための開始ロケーションを含み、前記長さの蓄積は、前記複数の順序付きセグメントのジオメトリシェーディング中に実行される、装置。
【請求項１４】
前記複数の順序付きセグメントが、１つまたは複数の可視セグメントと１つまたは複数の不可視セグメントとを含み、前記装置が、
前記現在のセグメントの前記決定されたテクスチャロケーションに基づいて、前記現在のセグメントが可視セグメントであるかどうかを決定するための手段と、
前記現在のセグメントが可視セグメントであることに基づいて前記現在のセグメントを保持する、または、前記現在のセグメントが可視セグメントではないことに基づいて前記現在のセグメントを破棄するための手段と
をさらに備える、請求項１２または１３に記載の装置。

【請求項１５】
前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線に対してジオメトリシェーディングを行うための手段と、
長さの前記累積に基づいて前記テクスチャオフセットを決定することが前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように、前記長さ値を決定するための手段と
をさらに備える、請求項１２に記載の装置。
【請求項１６】
前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線に対して前記ジオメトリシェーディングを行うための手段と、
長さの前記累積に基づいて前記テクスチャオフセットを決定することが前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように、前記長さ値を決定するための手段と
をさらに備える、請求項１３に記載の装置。
【請求項１７】
前記長さ値を決定するための前記手段が、前記順序が前のセグメントの長さを指定するｌｉｎｅｌｅｎｇｔｈスカラー値を生成するための手段を備える、請求項１５または１６に記載の装置。
【請求項１８】
前記複数の順序付きセグメントの前記セグメントの各々をラスタライズするための手段をさらに備え、
ここにおいて、前記テクスチャオフセットを適用するための前記手段が、前記現在のセグメントがラスタライズされた後に前記ラスタライズされた現在のセグメントに前記テクスチャオフセットを適用するための手段を備える、請求項１２または１３に記載の装置。
【請求項１９】
前記テクスチャオフセットを適用するための前記手段が、前記ロケーションを示す、前記現在のセグメントのテクスチャ座標値を決定するための手段を備える、請求項１２または１３に記載の装置。
【請求項２０】
前記複数の順序付きセグメントの前記順序がプリミティブ順序であるように、前記複数の順序付きセグメントのジオメトリシェーディング中に前記複数の順序付きセグメントの前記順序を決定するための手段をさらに備える、請求項１２に記載の装置。
【請求項２１】
前記複数の順序付きセグメントの前記順序がプリミティブ順序であるように、前記複数の順序付きセグメントの前記ジオメトリシェーディング中に前記複数の順序付きセグメントの前記順序を決定するための手段をさらに備える、請求項１３に記載の装置。
【請求項２２】
実行されたとき、グラフィックス処理ユニット（ＧＰＵ）に、請求項１から請求項１１のうちのいずれか一項に記載の方法を行わせる命令を記憶した非一時的コンピュータ可読媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
[0001]本出願は、その内容全体が参照により本明細書に組み込まれる、２０１３年８月２８日に出願された米国仮特許出願第６１／８７１，２６０号の利益を主張する。
【０００２】
[0002]本開示は、グラフィックス処理に関し、より詳細には、パスレンダリングのための技法に関する。

【背景技術】
【０００３】
[0003]パスレンダリングは、その各々が１つまたは複数のパスセグメントを含み得る、（あるいは、本明細書で「パス」と呼ばれる）２次元（２Ｄ）ベクタグラフィックスパス（vector graphics paths）のレンダリングを指す場合がある。パスが２つ以上のパスセグメントを含むとき、個々のパスセグメントは、同じタイプまたは異なるタイプのものであり得る。パスセグメントのタイプは、たとえば、線と、楕円弧と、２次ベジェ曲線と、３次ベジェ曲線とを含み得る。いくつかの例では、パスセグメントタイプは、たとえば、Ｏｐｅｎ　Ｖｅｃｔｏｒ　Ｇｒａｐｈｉｃｓ（ＯｐｅｎＶＧ）ＡＰＩなど、標準ベクタグラフィックスアプリケーションプログラミングインターフェース（ＡＰＩ）に従って定義され得る。
【０００４】
[0004]パスレンダリングは、中央処理装置（ＣＰＵ）で実装され得る。しかしながら、そのような手法は、ＣＰＵ集中的であり得、したがって、他のＣＰＵタスクに利用可能なＣＰＵ処理サイクルの量を制限する可能性がある。さらに、場合によっては、所望の詳細レベルでパスセグメントをレンダリングするために、比較的大量のデータがグラフィックス処理ユニット（ＧＰＵ）に転送される必要があり得る。比較的大量のデータは、データを記憶するとき、かなりの量のメモリストレージスペースを消費する場合があり、データをＧＰＵに転送するとき、かなりの量のメモリ帯域を消費する場合がある。
【発明の概要】
【０００５】
[0005]本開示は、パスのフィル（filling）とダッシング（dashing）とを用いてグラフィックスデータを生成するための技法を含む。たとえば、パスをフィルするときに、本開示の態様によれば、ＧＰＵは、メモリが（レンダターゲットと呼ばれる）レンダリングされたデータに割り当てられるレートとは異なるレートでステンシル（stenciling）動作を実行することができる。すなわち、ステンシル動作を実行するためのステンシルパラメータは、レンダリングされたデータを記憶するためのレンダターゲットパラメータから独立して指定され得る。
【０００６】
[0006]さらに、ダッシングに関して、本開示の態様によれば、ＧＰＵは、ダッシュ特性を決定し、単一のレンダリングパスでダッシングを実行することができる。たとえば、ＧＰＵは、セグメントが決定されるときにセグメントの各々の長さを計算し、各ダッシュセグメントのための開始ロケーション（たとえば、テクスチャ座標）を決定するために長さ情報を適用することができる。
【０００７】
[0007]一例では、グラフィックスデータをレンダリングする方法は、画像のパスの各アンチエイリアス画素のカバレージ値を決定するためのサンプリングレートを示すステンシルパラメータを決定することと、ステンシルパラメータとは別々に、パスの各アンチエイリアス画素のためのメモリ割当てを示すレンダターゲットパラメータを決定することと、ステンシルパラメータとレンダターゲットパラメータとを使用してパスをレンダリングすることとを含む。
【０００８】
[0008]別の例では、グラフィックスをレンディングするための装置は、画像のパスの各アンチエイリアス画素のカバレージ値を決定するためのサンプリングレートを示すステンシルパラメータを決定することと、ステンシルパラメータとは別々に、パスの各アンチエイリアス画素のためのメモリ割当てを示すレンダターゲットパラメータを決定することと、ステンシルパラメータとレンダターゲットパラメータとを使用してパスをレンダリングすることとを行うように構成されたグラフィックス処理ユニット（ＧＰＵ）を含む。
【０００９】
[0009]別の例では、グラフィックスデータをレンダリングするための装置は、画像のパ

スの各アンチエイリアス画素のカバレージ値を決定するためのサンプリングレートを示すステンシルパラメータを決定するための手段と、ステンシルパラメータとは別々に、パスの各アンチエイリアス画素のためのメモリ割当てを示すレンダターゲットパラメータを決定するための手段と、ステンシルパラメータとレンダターゲットパラメータとを使用してパスをレンダリングするための手段とを含む。

【００１０】

[0010]別の例では、非一時的コンピュータ可読媒体は、実行されたとき、グラフィックス処理ユニット（ＧＰＵ）に、画像のパスの各アンチエイリアス画素のカバレージ値を決定するためのサンプリングレートを示すステンシルパラメータを決定することと、ステンシルパラメータとは別々に、パスの各アンチエイリアス画素のためのメモリ割当てを示すレンダターゲットパラメータを決定することと、ステンシルパラメータとレンダターゲットパラメータとを使用してパスをレンダリングすることとを行わせる命令を記憶する。

【００１１】

[0011]別の例では、グラフィックスデータをレンダリングする方法は、グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットが、現在のセグメントより順序が前のセグメントの長さの累積に基づく、現在のセグメントのロケーションを決定するために、テクスチャオフセットを適用することを含めて現在のセグメントをピクセルシェーディングすることとを含む。

【００１２】

[0012]別の例では、グラフィックスデータをレンダリングするための装置は、破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットが、現在のセグメントより順序が前のセグメントの長さの累積に基づく、現在のセグメントのロケーションを決定するために、テクスチャオフセットを適用することを含めて現在のセグメントをピクセルシェーディングすることとを行うように構成されたグラフィックス処理ユニット（ＧＰＵ）を含む。

【００１３】

[0013]別の例では、グラフィックスデータをレンダリングするための装置は、グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定するための手段と、ここにおいて、複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットが、現在のセグメントより順序が前のセグメントの長さの累積に基づく、現在のセグメントのロケーションを決定するために、テクスチャオフセットを適用することを含めて現在のセグメントをピクセルシェーディングするための手段とを含む。

【００１４】

[0014]別の例では、非一時的コンピュータ可読媒体は、実行されたとき、構成されたグラフィックス処理ユニット（ＧＰＵ）に、破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットが、現在のセグメントより順序が前のセグメントの長さの累積に基づく、現在のセグメントのロケーションを決定するために、テクスチャオフセットを適用することを含めて現在のセグメントをピクセルシェーディングすることとを行わせる、命令を記憶する。

【００１５】

[0015]本開示の１つまたは複数の例の詳細が、添付の図面および下記の説明に記載される。本開示の他の特徴、目的、および利点は、説明および図面、ならびに特許請求の範囲から明らかになろう。

【図面の簡単な説明】

【００１６】

【図１】[0016]本開示の技法を実装するために使用され得る例示的なコンピューティングデバイスを示すブロック図。
【図２】[0017]図１のコンピューティングデバイスのＣＰＵ、ＧＰＵ、およびメモリをより詳細に示すブロック図。
【図３】[0018]本開示の技法を実行するために使用され得る例示的なグラフィックスパイプラインを示す概念図。
【図４】[0019]レンダリングされることになる例示的なパスの図。
【図５Ａ】[0020]図４に示されるパスのためのフィル動作の例示的なシーケンスを示す図。
【図５Ｂ】図４に示されるパスのためのフィル動作の例示的なシーケンスを示す図。
【図５Ｃ】図４に示されるパスのためのフィル動作の例示的なシーケンスを示す図。
【図６】[0021]ステンシル動作を示す概念図。
【図７】[0022]本開示の態様による、例示的なフィル動作を示す概念図。
【図８】[0023]本開示の態様による、レンダリング中の帯域幅を示すグラフ。
【図９Ａ】[0024]図４に示されるパスのための例示的なダッシング動作を示す一連の図。
【図９Ｂ】図４に示されるパスのための例示的なダッシング動作を示す一連の図。
【図９Ｃ】図４に示されるパスのための例示的なダッシング動作を示す一連の図。
【図９Ｄ】図４に示されるパスのための例示的なダッシング動作を示す一連の図。
【図１０】[0025]本開示の態様による、グラフィックスデータをレンダリングするための例示的なプロセスを示す流れ図。
【図１１】[0026]本開示の態様による、ダッシングするための例示的なプロセスを示す流れ図。
【発明を実施するための形態】
【００１７】
[0027]本開示は、ＧＰＵベースのパスレンダリングを実行するための技法に関する。パスレンダリングは、その各々が１つまたは複数のパスセグメントを含み得る（あるいは、本明細書で「パス」と呼ばれる）２次元（２Ｄ）ベクタグラフィックスパスのレンダリングを指す場合がある。パスが２つ以上のパスセグメントを含むとき、個々のパスセグメントは、同じタイプまたは異なるタイプのものであり得る。パスセグメントのタイプは、たとえば、線と、楕円弧と、２次ベジェ曲線と、３次ベジェ曲線とを含み得る。いくつかの例では、パスセグメントタイプは、たとえば、Ｏｐｅｎ　Ｖｅｃｔｏｒ　Ｇｒａｐｈｉｃｓ（ＯｐｅｎＶＧ）ＡＰＩなど、標準ベクタグラフィックスアプリケーションプログラミングインターフェース（ＡＰＩ）に従って定義され得る。
【００１８】
[0028]ＧＰＵは、通常、１つまたは複数の３ＤグラフィックスＡＰＩに対応するように設計された３次元（３Ｄ）グラフィックスパイプラインを実装する。今日使用されている一般的な３ＤグラフィックスＡＰＩは、対応デバイスがパスレンダリングコマンドをサポートすることを必要としないため、多くの場合、現代的なＧＰＵがパスレンダリングコマンド用のハードウェアアクセラレーションを提供することはほとんどない。たとえば、現代的なＧＰＵで実装される典型的な３Ｄグラフィックスパイプラインは、（たとえば、点、線、および三角形など）低次の、湾曲していない３Ｄグラフィックスプリミティブをラスタライズするように設計されるが、（たとえば、楕円弧、およびベジェ曲線など）湾曲したパスレンダリングプリミティブを直接的にレンダリングすることができないラスタライザを含み得る。
【００１９】
[0029]パスレンダリングに関する一手法は、パスレンダリングコマンドを実行する目的で部分的なＧＰＵハードウェアアクセラレーションを提供するために３Ｄ　ＧＰＵプラインを使用することに関連し得る。この手法は、パスセグメントを、ＧＰＵによってラスタライズされ得る、１つまたは複数の低次の、湾曲していない３Ｄグラフィックスプリミティブに変換するために、中央処理装置（ＣＰＵ）を用いてパスセグメントを前処理するこ

とに関連する。たとえば、ＣＰＵは、湾曲したパスセグメント（たとえば、楕円弧またはベジェ曲線）を、パスセグメントの曲率を近似する比較的小さな三角形のセットにテッセレートすることができ、ＧＰＵを使用して三角形のセットをレンダリングさせることができる。しかしながら、そのような手法は、ＣＰＵ集中的であり得、したがって、他のＣＰＵタスクに利用可能なＣＰＵ処理サイクルの量を制限する可能性がある。さらに、場合によっては、所望の詳細レベルでパスセグメントをレンダリングするために、比較的に多数の三角形が必要とされる場合がある。比較的に多数の三角形は、データを記憶するとき、かなりの量のメモリストレージスペースを消費する場合があり、データをＧＰＵに転送するとき、かなりの量のメモリ帯域を消費する場合がある。

【００２０】

[0030]パスレンダリングコマンドの実行に部分的－全体的ＧＰＵハードウェアアクセラレーションを提供するための別の手法は、専用の、ハードウェアアクセラレーションされたパスレンダリングパイプラインをサポートするようにＧＰＵのアーキテクチャを変更することを伴い得る。しかしながら、一般的な３ＤグラフィックスＡＰＩ（たとえば、Ｍｉｃｒｏｓｏｆｔ（登録商標）Ｄｉｒｅｃｔ　Ｘ１１（ＤＸ）ＡＰＩ）は、一般に、ＧＰＵアーキテクチャが専用のパスレンダリングパイプラインを含むことを必要としないので、そのような手法は、特定の３ＤグラフィックスＡＰＩ（たとえば、ＤＸ　１１　ＡＰＩ）に準拠するすべてのＧＰＵによってサポートされることが保証されるであろうクロスプラットフォームな、ハードウェアアクセラレーションされたパスレンダリングソリューションを生じない。

【００２１】

[0031]いくつかの例では、受け取ったパスセグメントを複数のラインセグメントにテッセレートすることと、３Ｄグラフィックスパイプラインを使用してテッセレートされたラインセグメントをレンダリングすることとを行うようにＧＰＵが構成されたＧＰＵベースのパスレンダリング技法が使用され得る。パスセグメントをラインセグメントにテッセレートするためにＧＰＵを使用することによって、パスセグメントを前処理する負担をＣＰＵから取り除き、それによって、他のＣＰＵタスクのために処理リソースを解放する。さらに、いくつかの例では、ＧＰＵは、テッセレーション演算を実行するために、いくつかの例では、ＧＰＵが、ＣＰＵよりも効率的な形でパスセグメントをレンダリングするのを可能にする、高並列の現代的なＧＰＵテッセレーションアーキテクチャを利用することができる。加えて、テッセレーションは、ＣＰＵ内ではなく、ＧＰＵ内で発生するため、多数のテッセレートされたプリミティブは、システムメモリ内に記憶される必要がなく、ＣＰＵからＧＰＵに渡される必要がなく、それによって、パスレンダリングに必要とされるメモリフットプリント、ならびに、パスレンダリングに必要とされるメモリ帯域幅を削減する。

【００２２】

[0032]いくつかの例では、ＧＰＵは、アンチエイリアシングを実行するためにマルチサンプルアンチエイリアシング（ＭＳＡＡ：multi-sample anti-aliasing）技法を使用することができる。たとえば、画素は、一様に着色され、常に同じ形状にあり、これにより、レンダリングされた画像の線の外観にジャギーが生じ得る。ＭＳＡＡでは、単一の画素に対して複数のサンプルが生成され得る。サンプルは、次いで、最終画素値を決定するために、組み合わされ得る（たとえば、平均化され得る）。

【００２３】

[0033]したがって、いくつかの事例では、ＧＰＵは、表示されている解像度よりも高い解像度で画像をレンダリングすることができる。ＧＰＵは、次いで、表示より前に適切なサイズに画像をダウンサンプリングすることができる。結果は、オブジェクトの縁に沿って画素の１つの線から別の線へのよりスムーズな遷移であり得る。ＭＳＡＡは、４、８、１６、または他の値の係数を使用して実行され得る。ＭＳＡＡを実行するとき、ＧＰＵは、ＭＳＡＡレートで深度およびステンシル動作をサンプリングし、ＭＳＡＡレートでメモリを割り当て、ＭＳＡＡレートにおいて画素をラスタライズすることができる（たとえば

、１６ｘのＭＳＡＡは、画素ごとに１６ｘの深度／ステンシルサンプルと、画素ごとに１６ｘのメモリ割当てと、画素ごとに１６ｘのラスタライゼーションサンプルとを含む）。

【００２４】

[0034]概して、「ターゲット」は、レンダリングされた画素に割り当てられるメモリを指す場合がある。一般に、アンチエイリアス画像に関して、レンダリングされたターゲットのためのラスタライゼーションおよびメモリ割当てなどのグラフィックス演算を実行するサンプリングレートは、互いに対応し、たとえば、１：１になる。したがって、説明のための一例では、ＧＰＵは、ラスタライゼーションのために画素ごとに１６ｘのサンプリングレートを使用し、画素ごとに１６個のサンプルを記憶するようにメモリを割り当て得る。しかしながら、ターゲット独立ラスタライゼーション（ＴＩＲ：target independent rasterization）では、ラスタライゼーションプロセスのサンプリングレートが、レンダリングされた画像に割り当てられるメモリから独立して指定され得る。たとえば、画素ごとに４つのサンプルのサンプリングレートがラスタライゼーションのために使用され得、一方、画像の画素の色を記憶するためのメモリ割当ては、画像中の画素ごとに１色になり得る。

【００２５】

[0035]ＴＩＲにより、ターゲットに割り当てられるメモリから独立してラスタライゼーションレートを指定することが可能になるが、他のレンダリング動作は、結び合わされたままであり得る。たとえば、（以下でより詳細に説明する）深度およびステンシル動作は、一般に、レンダターゲットに関連付けられ得る。したがって、単一のレンダターゲットは、画素ごとに指定され、深度およびステンシル動作はまた、同じレート（すなわち、１ｘのサンプリングレート）で実行され得る。

【００２６】

[0036]本開示の態様によれば、ＧＰＵは、ステンシル動作中にＴＩＲの概念を活用することができる。たとえば、ＧＰＵは、特定の画素に割り当てられるメモリの量よりも高いレートでステンシルを実行することができる。すなわち、ステンシル動作がスーパーサンプリングされる（super sampled）、たとえば、各画素が１６個のサンプルを有することになるプロセスでは、ＧＰＵは、（スーパーサンプリングされた画素のうちの）画素のどのサンプルが、ステンシルテストにパスしたか、たとえば、特定のパスの内側にあったかに基づいて画素ごとにカバレージ値を計算することによってレンダリングすることができる。パフォーマンスの改善のために、レンダターゲットは、１ｘでサンプリングされ得、一方、ステンシルは、１６ｘでサンプリングされ得る。ＧＰＵは、サンプルごとのステンシルテストに基づいて各画素にカバレージ値を割り当て得る。ターゲットとラスタライゼーションレートとから独立してステンシルサンプリングレートを指定することは、本明細書ではステンシルＴＩＲと呼ばれる場合がある。

【００２７】

[0037]ステンシルＴＩＲプロセスは、パスレンダリング中に適用され得る。たとえば、パスレンダリング時に、ＧＰＵは、一般に、パスをフィルするために、ラインセグメントにパスをテッセレートし、ラインセグメントをピボット点に接続して三角形を形成し、（いくつかの事例では、深度テストを実行することを含めて）三角形をステンシルバッファにレンダリングするという例示的な機能を実行し得、ここで、ステンシルバッファは、画像の可視画素インを示す。フィルプロセスの次の、場合によっては最後のステップは、ステンシルテストを有効化した状態でバウンディングボックスをレンダリングすることと、フレームバッファにステンシルバッファのコンテンツをコピーすることとを行うことである。この手法は、２つのレンダリングパス、たとえば、バウンディングボックスをレンダリングする１つのパスと、テクスチャをレンダリングする１つのパスとを必要とする。

【００２８】

[0038]本開示の態様によれば、ＧＰＵは、バウンディングボックスを前処理する必要なしに単一のレンダリングパスでパスをフィルすることができる。たとえば、いくつかの例では、ＧＰＵは、ラスタライザ段階において使用されるハードウェアを含み得るバウンデ

ィングボックスユニットを組み込み得る。たとえば、プリミティブがステンシルバッファにレンダリングされるとき、バウンディングボックスユニットは、所与のパスの最外座標点（たとえば、上極値、下極値、左極値、および右極値）を追跡することができる。最外座標点は、これらの点がパスの最外境界を示すという点で、最大境界点と呼ばれる場合もある。ステンシルが完了した後、バウンディングボックスユニットは、最外座標点に基づいて境界長方形を決定している。

【００２９】

[0039]上記の例では、パスのプリミティブがステンシルバッファにレンダリングされる（プリミティブはステンシルにのみ影響を及ぼす）ので、ＧＰＵは、パスのプリミティブをシェーディングしない。ＧＰＵは、次に、色を割り当てるためにステンシルバッファを使用してバウンディングボックスをレンダリングすることができる。本開示の態様によれば、ステンシルを実行し、バウンディングボックスを決定した後に別の描画呼出しを必要としない。むしろ、ＧＰＵは、単一のパスでステンシルＴＩＲを使用してバウンディングボックスをラスタライズする。

【００３０】

[0040]このようにして、ＧＰＵは、たとえば、ＧＰＵにおいてプリミティブを決定すること、ＣＰＵにおいてバウンディングボックスを決定すること、ＧＰＵ上で着色動作を実行することを行うのではなく、単一のパスでフィルすることができる（たとえば、ステンシルおよび着色動作を実行することができる）。すなわち、本開示の技法は、ＧＰＵが、ステンシルと着色の両方が単一のパスで実行され得るように、（たとえば、ＧＰＵが次いでラスタライザにプッシュすることができるテッセレーション中に）バウンディングボックスを決定することを可能にするバウンディングボックス最適化を含む。

【００３１】

[0041]本開示の他の態様は、ダッシング（破線など）に関する。たとえば、ストロークされたパスをダッシングするとき、ＧＰＵは、（セグメント順序と呼ばれる）順序でダッシュセグメントをレンダリングすることができ、前のセグメントが途切れた場所に１つのセグメントを生成することができる。すなわち、ＧＰＵは、前のセグメントをシェーディングした後にのみ、ダッシュパターンの各セグメントのための開始ロケーションを決定する。そのような計算は、正しい開始ロケーションを決定するためにダッシュの各セクションのためのロケーションが処理される必要があるので、グラフィックス処理の並列性を低減し、２つ以上のレンダリングパスを実行することを必要とし得る。

【００３２】

[0042]本開示の態様によれば、ＧＰＵは、ダッシュ特性を決定し、単一のパス、たとえば、単一のレンダリングパスでダッシングを実行することができる。たとえば、ＧＰＵは、たとえば、ジオメトリシェーディング中にセグメントが決定されると、セグメントの各々の長さを計算することができる。すなわち、ＧＰＵは、現在のセグメントの開始ロケーションを決定するために、セグメント、たとえば、セグメント順序で現在のセグメントの前のセグメントの長さを累積することができる。長さのこの累積は、本明細書では、「プレフィックス長さ」または「プレフィックス総和長さ」と呼ばれる場合がある。ＧＰＵはまた、線の全長を決定することができる。

【００３３】

[0043]説明のための一例では、ＧＰＵは、破線の第１のセグメントを決定することができる。ＧＰＵはまた、破線の第２のセグメントを決定することができる。ＧＰＵは、前のセグメントのプレフィックス総和長さに基づいて第２のセグメントのための開始ロケーションを決定することができる。すなわち、ＧＰＵは、前のセグメント、すなわち、第１のセグメントの長さの累積に基づいて第２のセグメントのための開始ロケーションを決定することができる。ＧＰＵはまた、破線の第３のセグメントを決定することができる。この場合も、ＧＰＵは、前のセグメントのプレフィックス総和長さに基づいて第３のセグメントのための開始ロケーションを決定することができる。すなわち、ＧＰＵは、前のセグメント、すなわち、第１のセグメントと第２のセグメントとの長さの累積に基づいて第３の

セグメントのための開始ロケーションを決定することができる。ＧＰＵは、線のセグメントの各々の開始ロケーションが決定されるまで、このようにして継続することができる。

【００３４】

[0044]いくつかの例では、破線は、可視セグメントと不可視セグメントとを含み得る。たとえば、ＧＰＵ１２は、可視（たとえば、線のダッシュ）であるセグメントのための色を決定し、不可視（たとえば、着色ダッシュ間の破線の部分）であるセグメントを破棄することができる。ＧＰＵ１２は、シェーディングされているセグメントのロケーションに基づいて（たとえば、本明細書では、ピクセルシェーディング中のフラグメントと互換的に呼ばれる場合がある）セグメントを保持すべきかどうかを決定することができる。一例として上記で説明した３つのセグメントに関して、破線の第１および第３のセグメントを仮定し、第２のセグメントは、着色されていない、第１のセグメントと第３のセグメントとを分離する不可視セグメントである。ＧＰＵ１２は、セグメントのロケーションに基づいて、ピクセルシェーディング中にセグメントを保持（たとえば、色を用いてシェーディング）すべきか、または破棄すべきかを決定することができる。すなわち、ＧＰＵ１２は、第１のセグメントのロケーションに基づいて第１のセグメントが保持されることを決定し、第２のセグメントのロケーションに基づいて第２のセグメントが破棄されることを決定し、第３のセグメントのロケーションに基づいて第３のセグメントが保持されることを決定する。

【００３５】

[0045]本開示の態様によれば、ＧＰＵは、レンダリング中に各セグメントのためのプレフィックス総和長さをテクスチャオフセットとして適用することができる。たとえば、セグメントをラスタライズした後に、ＧＰＵは、テクスチャオフセット値としてピクセルシェーダにセグメントのためのプレフィックス総和長さの値を供給することができる。ＧＰＵは、シェーディングされているセグメントのロケーションを決定するために、線の始端のテクスチャ座標にテクスチャオフセットを適用することができる。

【００３６】

[0046]図１は、本開示の技法を実装するために使用され得る例示的なコンピューティングシステム２を示すブロック図である。コンピューティングデバイス２は、パーソナルコンピュータ、デスクトップコンピュータ、ラップトップコンピュータ、コンピュータワークステーション、ビデオゲームプラットフォームもしくはコンソール、（たとえば、モバイル電話、セルラー電話、衛星電話、および／もしくはモバイル電話ハンドセットなど）ワイヤレス通信デバイス、固定電話、インターネット電話、ポータブルビデオゲームデバイスもしくは携帯情報端末（ＰＤＡ）などのハンドヘルドデバイス、パーソナル音楽プレーヤ、ビデオプレーヤ、ディスプレイデバイス、テレビジョン、テレビジョンセットトップボックス、サーバ、中間ネットワークデバイス、メインフレームコンピュータ、またはグラフィカルデータを処理および／もしくは表示する任意の他のタイプのデバイスを備えることができる。

【００３７】

[0047]図１の例に示すように、コンピューティングデバイス２は、ユーザインターフェース４と、ＣＰＵ６と、メモリコントローラ８と、メモリ１０と、グラフィックス処理ユニット（ＧＰＵ）１２と、ＧＰＵキャッシュ１４と、ディスプレイインターフェース１６と、ディスプレイ１８と、バス２０とを含む。ユーザインターフェース４、ＣＰＵ６、メモリコントローラ８、ＧＰＵ１２、およびディスプレイインターフェース１６は、バス２０を使用して互いと通信することができる。図１に示す異なる構成要素同士の間のバスおよび通信インターフェースの特定の構成は単なる例示であり、本開示の本技法を実装するために、同じもしくは異なる構成要素を備えたコンピューティングデバイスおよび／または他のグラフィックス処理システムの他の構成が使用され得ることに留意されたい。

【００３８】

[0048]ＣＰＵ６は、コンピューティングデバイス２の動作を制御する汎用プロセッサまたは専用プロセッサを備えることができる。ユーザは、ＣＰＵ６に１つまたは複数のソフ

トウェアアプリケーションを実行させるための入力をコンピューティングデバイス２に与えることができる。ＣＰＵ６上で実行されるそれらのソフトウェアアプリケーションは、たとえば、オペレーティングシステム、ワードプロセッサアプリケーション、電子メールアプリケーション、スプレッドシートアプリケーション、メディアプレーヤアプリケーション、ビデオゲームアプリケーション、グラフィカルユーザインターフェースアプリケーション、または別のプログラムを含み得る。ユーザは、ユーザインターフェース４を介してコンピューティングデバイス２に結合される、キーボード、マウス、マイクロフォン、タッチパッドまたは別の入力デバイスなど、１つもしくは複数の入力デバイス（図示せず）を介してコンピューティングデバイス２に入力を与えることができる。

【００３９】

[0049]ＣＰＵ６上で実行するソフトウェアアプリケーションは、グラフィックスデータをディスプレイ１８にレンダリングさせるようにＧＰＵ１２に命令する、１つまたは複数のグラフィックスレンダリング命令を含み得る。いくつかの例では、ソフトウェア命令は、たとえば、Ｏｐｅｎ Ｇｒａｐｈｉｃｓ Ｌｉｂｒａｒｙ（ＯｐｅｎＧＬ（商標登録））ＡＰＩ、Ｏｐｅｎ Ｇｒａｐｈｉｃｓ Ｌｉｂｒａｒｙ Ｅｍｂｅｄｄｅｄ Ｓｙｓｔｅｍ（ＯｐｅｎＧＬ ＥＳ）ＡＰＩ、Ｄｉｒｅｃｔ３Ｄ ＡＰＩ、ＤｉｒｅｃｔＸ ＡＰＩ、ＲｅｎｄｅｒＭａｎ ＡＰＩ、ＷｅｂＧＬ ＡＰＩ、または任意の他の公的もしくは所有権を主張できる標準グラフィックスＡＰＩなど、グラフィックスアプリケーションプログラミングインターフェース（ＡＰＩ）に準拠し得る。グラフィックスレンダリング命令を処理するために、ＣＰＵ６は、ＧＰＵ１２にグラフィックスデータのレンダリングのうちの一部またはすべてを実行させるようにＧＰＵ１２に命令するための、１つまたは複数のグラフィックスレンダリングコマンドを発行することができる。いくつかの例では、レンダリングされることになるグラフィックスデータは、たとえば、点、線、三角形、クアドララテラル（quadralaterals）、トライアングルストリップ（triangle strips）、パッチなど、グラフィックスプリミティブのリストを含み得る。さらなる例では、レンダリングされることになるグラフィックスデータは、たとえば、ラインセグメント、楕円弧、２次ベジェ曲線、および３次ベジェ曲線など、１つまたは複数のパスレンダリングプリミティブを含み得る。

【００４０】

[0050]メモリコントローラ８は、メモリ１０との間を行き来するデータの転送を容易にする。たとえば、メモリコントローラ８は、メモリ読取り要求とメモリ書込み要求とをＣＰＵ６および／またはＧＰＵ１２から受け取って、コンピューティングデバイス２内の構成要素にメモリサービスを提供するために、メモリ１０に関するそのような要求にサービス提供することができる。メモリコントローラ８はメモリ１０に通信可能に結合される。メモリコントローラ８は、図１の例示的なコンピューティングデバイス２内で、ＣＰＵ６、ＧＰＵ１２、およびメモリ１０の各々とは別である処理モジュールとして示されているが、他の例では、メモリコントローラ８の機能の一部またはすべては、ＣＰＵ６、ＧＰＵ１２、およびメモリ１０のうちの１つもしくは複数の上で実装され得る。

【００４１】

[0051]メモリ１０は、ＣＰＵ６による実行のためにアクセス可能なプログラムモジュールおよび／もしくは命令、ならびに／またはＣＰＵ６上で実行するプログラムによって使用するためのデータを記憶することができる。たとえば、メモリ１０は、ユーザアプリケーションと、それらのアプリケーションと関連付けられたグラフィックスデータとを記憶することができる。メモリ１０は、コンピューティングデバイス２の他の構成要素によって使用するため、および／または生成されるための情報を記憶することも可能である。たとえば、メモリ１０は、ＧＰＵ１２のデバイスメモリとして機能することができ、ＧＰＵ１２によって演算されことになるデータ、ならびにＧＰＵ１２によって実行される演算の結果生じるデータを記憶することができる。たとえば、メモリ１０は、パスデータ、パスセグメントデータ、表面、テクスチャバッファ、デプスバッファ、ステンシルバッファ、頂点バッファ、フレームバッファなどの任意の組合せを記憶することができる。加えて、

メモリ１０は、ＧＰＵ１２によって処理するためのコマンドストリームを記憶することができる。たとえば、メモリ１０は、パスレンダリングコマンド、３Ｄグラフィックスレンダリングコマンド、および／または汎用ＧＰＵコンピューティングコマンドを記憶することができる。メモリ１０は、たとえば、ランダムアクセスメモリ（ＲＡＭ）、スタティックＲＡＭ（ＳＲＡＭ）、ダイナミックＲＡＭ（ＤＲＡＭ）、同期式ダイナミックランダムアクセスメモリ（ＳＤＲＡＭ）、読取り専用メモリ（ＲＯＭ）、消去可能プログラマブルＲＯＭ（ＥＰＲＯＭ）、電気的消去可能プログラマブルＲＯＭ（ＥＥＰＲＯＭ（登録商標））、フラッシュメモリ、磁気データ媒体または光記憶媒体など、１つもしくは複数の揮発性または不揮発性のメモリあるいは記憶デバイスを含み得る。

【００４２】

[0052]ＧＰＵ１２は、ＣＰＵ６によってＧＰＵ１２に発行されたコマンドを実行するように構成され得る。ＧＰＵ１２によって実行されるコマンドは、グラフィックスコマンド、描画呼出し（draw call）コマンド、ＧＰＵ状態プログラミングコマンド、メモリ転送コマンド、汎用コンピューティングコマンド、カーネル実行コマンドなどを含み得る。

【００４３】

[0053]いくつかの例では、ＧＰＵ１２は、ディスプレイ１８に１つまたは複数のグラフィックスプリミティブをレンダリングするためのグラフィックス演算を実行するように構成され得る。そのような例では、ＣＰＵ６上で実行するソフトウェアアプリケーションの１つがグラフィックス処理を必要とするとき、ＣＰＵ６は、ディスプレイ１８にレンダリングするためのグラフィックスデータをＧＰＵ１２に提供して、ＧＰＵ１２に対する１つまたは複数のグラフィックスコマンドを発行することができる。グラフィックスコマンドは、たとえば、描画呼出しコマンド、ＧＰＵ状態プログラミングコマンド、メモリ転送コマンド、ブリッティング（blitting）コマンドなどを含み得る。グラフィカルデータは、頂点バッファ、テクスチャデータ、表面データなどを含み得る。いくつかの例では、ＣＰＵ６は、コマンドとグラフィックスデータとをＧＰＵ１２によってアクセス可能なメモリ１０に書き込むことによって、コマンドとグラフィックスデータとをＧＰＵ１２に提供することができる。

【００４４】

[0054]さらなる例では、ＧＰＵ１２は、ＣＰＵ６上で実行するアプリケーションに関して、汎用コンピューティングを実行するように構成され得る。そのような例では、ＣＰＵ６上で実行するソフトウェアアプリケーションのうちの１つが計算タスクをＧＰＵ１２にオフロードすることを決定するとき、ＣＰＵ６は、汎用コンピューティングデータをＧＰＵ１２に提供して、ＧＰＵ１２に対する１つまたは複数の汎用コンピューティングコマンドを発行することができる。汎用コンピューティングコマンドは、たとえば、カーネル実行コマンド、メモリ転送コマンドなどを含み得る。いくつかの例では、ＣＰＵ６は、コマンドとグラフィックスデータとをＧＰＵ１２によってアクセス可能なメモリ１０に書き込むことによって、コマンドと汎用コンピューティングデータとをＧＰＵ１２に提供することができる。

【００４５】

[0055]ＧＰＵ１２は、いくつかの例では、ベクタ演算についてＣＰＵ６よりも効率的な処理を行う高並列構造を用いて構築され得る。たとえば、ＧＰＵ１２は、複数の頂点、制御点、画素および／または他のデータに関して並列な形で演算するように構成された複数の処理要素を含み得る。ＧＰＵ１２の高並列性質は、いくつかの例では、ＧＰＵ１２が、ＣＰＵ６を使用して画像をレンダリングするよりもより迅速にグラフィックス画像（たとえば、ＧＵＩおよび２次元（２Ｄ）ならびに／または３次元（３Ｄ）のグラフィックスシーン）をディスプレイ１８上にレンダリングするのを可能にする。加えて、ＧＰＵ１２の高並列性質は、ＧＰＵ１２が、ＣＰＵ６よりもより迅速に、汎用コンピューティングアプリケーションに関して、ある種のタイプのベクトル演算および行列演算を処理するのを可能にし得る。

【００４６】

[0056]いくつかの例では、ＧＰＵ１２は、コンピューティングデバイス２のマザーボードに統合され得る。他の例では、ＧＰＵ１２は、コンピューティングデバイス２のマザーボードにおけるポートに設置されるグラフィックスカード上に存在し得るか、または場合によっては、コンピューティングデバイス２と相互運用するように構成された周辺デバイス内に組み込まれ得る。さらなる例では、ＧＰＵ１２は、システムオンチップ（ＳｏＣ）を形成するＣＰＵ６と同じマイクロチップ上に配置され得る。ＧＰＵ１２は、１つもしくは複数のマイクロプロセッサ、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、デジタル信号プロセッサ（ＤＳＰ）、あるいは他の等価な集積論理回路または個別論理回路など、１つもしくは複数のプロセッサを含み得る。

【００４７】

[0057]いくつかの例では、ＧＰＵ１２はＧＰＵキャッシュ１４に直接結合され得る。したがって、ＧＰＵ１２は、必ずしもバス２０を使用せずに、ＧＰＵキャッシュ１４からデータを読み取り、ＧＰＵキャッシュ１４にデータを書き込むことができる。言い換えれば、ＧＰＵ１２は、オフチップメモリの代わりに、ローカルストレージを使用してデータをローカルで処理することができる。これにより、ＧＰＵ１２は、大量のバストラフィックを受けることがある、バス２０を介したデータの読取りおよび書込みの必要がなくなるので、より効率的な方法で動作できるようになる。しかしながら、いくつかの例では、ＧＰＵ１２は、別個のキャッシュを含まず、代わりに、バス２０を介してメモリ１０を利用することができる。ＧＰＵキャッシュ１４は、たとえば、ランダムアクセスメモリ（ＲＡＭ）、スタティックＲＡＭ（ＳＲＡＭ）、ダイナミックＲＡＭ（ＤＲＡＭ）、消去可能プログラマブルＲＯＭ（ＥＰＲＯＭ）、電気消去可能プログラマブルＲＯＭ（ＥＥＰＲＯＭ）、フラッシュメモリ、磁気データ媒体または光学データ媒体など、１つもしくは複数の揮発性または不揮発性のメモリあるいは記憶デバイスを含み得る。

【００４８】

[0058]ＣＰＵ６および／またはＧＰＵ１２は、レンダリングされた画像データをメモリ１０内に割り振られたフレームバッファ内に記憶することができる。ベクタグラフィックスに関して、レンダリングされた画像データは、レンダリングされることになるパスセグメントに関してレンダリングされたフィル領域とストローク領域とを含み得る。ディスプレイインターフェース１６は、データをフレームバッファから取り出して、レンダリングされた画像データによって表される画像を表示するようにディスプレイ１８を構成することができる。いくつかの例では、ディスプレイインターフェース１６は、フレームバッファから取り出されたデジタル値をディスプレイ１８によって消費され得るアナログ信号に変換するように構成されたデジタルアナログ変換器（ＤＡＣ）を含み得る。他の例では、ディスプレイインターフェース１６は、処理のために、デジタル値をディスプレイ１８に直接的に渡すことができる。

【００４９】

[0059]ディスプレイ１８は、モニタ、テレビジョン、投影デバイス、液晶ディスプレイ（ＬＣＤ）、プラズマディスプレイパネル、発光ダイオード（ＬＥＤ）アレイ、陰極線管（ＣＲＴ）ディスプレイ、電子ペーパー、表面伝導型電子放出素子ディスプレイ（ＳＥＤ）、レーザテレビジョンディスプレイ、ナノ結晶ディスプレイまたは別のタイプのディスプレイユニットを含み得る。ディスプレイ１８はコンピューティングデバイス２内に統合され得る。たとえば、ディスプレイ１８は、モバイル電話ハンドセットまたはタブレットコンピュータのスクリーンとすることができる。あるいは、ディスプレイ１８は、ワイヤード通信リンクまたはワイヤレス通信リンクを介してコンピュータデバイス２に結合されるスタンドアロンデバイスとすることができる。たとえば、ディスプレイ１８は、ケーブルリンクまたはワイヤレスリンクを介してパーソナルコンピュータに接続されるコンピュータモニタまたはフラットパネルディスプレイとすることができる。

【００５０】

[0060]バス２０は、第１世代、第２世代、および第３世代のバス構造ならびにバスプロトコルと、共有バス構造およびバスプロトコルと、ポイントツーポイントバス構造および

バスプロトコルと、一方向バス構造およびバスプロトコルと、双方向バス構造およびバスプロトコルとを含めて、バス構造およびバスプロトコルの任意の組合せを使用して実装され得る。バス２０を実装するために使用され得る様々なバス構造およびバスプロトコルの例は、たとえば、ＨｙｐｅｒＴｒａｎｓｐｏｒｔバス、ＩｎｆｉｎｉＢａｎｄバス、Ａｄｖａｎｃｅｄ Ｇｒａｐｈｉｃｓ Ｐｏｒｔバス、Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ（ＰＣＩ）バス、ＰＣＩ Ｅｘｐｒｅｓｓバス、Ａｄｖａｎｃｅｄ Ｍｉｃｒｏｃｏｎｔｒｏｌｌｅｒ Ｂｕｓ Ａｒｃｈｉｔｅｃｔｕｒｅ（ＡＭＢＡ）Ａｄｖａｎｃｅｄ Ｈｉｇｈ－ｐｅｒｆｏｒｍａｎｃｅ Ｂｕｓ（ＡＨＢ）、ＡＭＢＡ Ａｄｖａｎｃｅｄ Ｐｅｒｉｐｈｅｒａｌ Ｂｕｓ（ＡＰＢ）、およびＡＭＢＡ Ａｄｖａｎｃｅｄ ｅＸｅｎｔｉｓｉｂｌｅ Ｉｎｔｅｒｆａｃｅ（ＡＸＩ）バスを含む。他のタイプのバス構造およびバスプロトコルも使用され得る。

【００５１】

[0061]いくつかの事例では、ＧＰＵ１２は、部分的－全体的（partial-to-total）ＧＰＵベースの様々なパスレンダリングコマンドの実行を実現するように構成され得る。たとえば、ＣＰＵ６は、ＧＰＵ１２に対する１つまたは複数のパスレンダリングコマンドを発行することができ、ＧＰＵ１２は、パスレンダリングコマンドを実行することができる。一例として、ＣＰＵ６は、ＧＰＵ１２にパスフィル動作を実行するように命令する１つまたは複数のパスフィルコマンドをＧＰＵ１２に発行することができ、ＧＰＵ１２は、パスフィルコマンドを実行することができる。別の例として、ＣＰＵ６は、ＧＰＵ１２にパスストローク動作を実行するように命令する１つまたは複数のパスストロークコマンドをＧＰＵ１２に発行することができ、ＧＰＵ１２は、パスストロークコマンドを実行することができる。

【００５２】

[0062]いくつかの例では、ＧＰＵ１２は、レンダリングされることになるパスのパスセグメントを示すデータを受け取ることと、複数のプリミティブにパスセグメントをテッセレートすることと、複数のプリミティブに基づいてパスセグメントに関するフィル領域とストローク領域とのうちの少なくとも１つをレンダリングすることとを行うように構成され得る。ＧＰＵは、フィル動作を実行するときにパスセグメントに関するフィル領域をレンダリングすることができ、ストローク動作を実行するときにパスセグメントに関するストローク領域をレンダリングすることができる。複数のプリミティブは、いくつかの例では、複数のラインセグメントであり得る。

【００５３】

[0063]いくつかの例では、ＧＰＵ１２は、パスフィル動作を実行するために、２パスレンダリング手法を使用することができる。たとえば、第１のレンダリングパスの一部として、ＧＰＵ１２は、レンダリングされることになるパスのパスセグメントを示すデータを受け取り、複数のラインセグメントにパスセグメントをテッセレートし、複数のラインセグメントに基づいて複数の三角形プリミティブを生成することができる。ＧＰＵ１２は、複数のラインセグメントのそれぞれに基づいて複数の三角形プリミティブの各々を生成することができる。ＧＰＵ１２は、どの画素がパスセグメントに関するフィル領域の内側にあるかを示すデータを共通ステンシルバッファが記憶するように、共通のステンシルバッファに複数の三角形プリミティブの各々をレンダリングすることができる。共通のステンシルバッファにプリミティブをレンダリングした後に、ＧＰＵ１２は、第２のレンダリングパスを実行することができる。第２のレンダリングパス中に、ＧＰＵ１２は、パスセグメントに関するフィル領域のラスタライズされたバージョンを生成するために、ステンシルバッファに記憶されたデータとフィル色に基づいてパスセグメントに関するフィル領域の内側にある画素を包含する１つまたは複数のプリミティブをレンダリングすることができる。

【００５４】

[0064]パスフィル動作のための複数の三角形プリミティブを生成するために、ＧＰＵ１２は、いくつかの例では、パスセグメントに関して生成される三角形プリミティブのすべ

てに対して同じである共通の頂点を三角形プリミティブの各々が有するように、複数の三角形プリミティブを生成することができる。そのような例では、ＧＰＵ１２は、複数のラインセグメントのそれぞれの終点に対応する２つの追加の頂点（すなわち、共通の頂点に加えて２つの頂点）を三角形プリミティブの各々が有するように、複数の三角形プリミティブを生成することができる。追加の各頂点は、対応するラインセグメントの終点のそれぞれに対応し得る。

【００５５】

[0065]したがって、パスレンダリングを実行するとき、ＧＰＵ１２は、パスをフィルするために、ラインセグメントにパスをテッセレートし、ラインセグメントをピボット点に接続して三角形プリミティブを形成し、三角形をステンシルバッファにレンダリングするという以下の例示的な機能を実行することができる。フィルプロセスの次の、場合によっては最後のステップは、ステンシルテストを有効化した状態で（たとえば、図５Ｃに関してより詳細に説明するように）パスを包含するバウンディングボックスをレンダリングすることと、フレームバッファにステンシルコンテンツをコピーすることとを行うことである。いくつかの事例では、バウンディングボックスは、ＣＰＵ６から受け取ったコマンドに基づいて決定され得る。上記のように、この手法は、バウンディングボックスを計算するために、２つのレンダリングパスとパスの前処理とを必要とする。

【００５６】

[0066]さらに、ＭＳＡＡなどのアンチエイリアシングを実行するとき、ＧＰＵ１２は、レンダターゲットと同じレートでステンシルバッファをサンプリングすることができる。たとえば、ステンシルバッファとレンダターゲットとがどちらもＭＳＡＡレートでサンプリングされる場合、フレームバッファにステンシルバッファをコピーするときに消費されるメモリ帯域幅は、比較的大きくなり得る。ＧＰＵ１２がＴＩＲを実行し、レンダターゲットのために比較的小さい割当てを使用する場合、ステンシルサンプリングレートも影響を受け、それによって、ステンシルバッファの精度を低減し得る。

【００５７】

[0067]本開示の態様によれば、ＧＰＵ１２は、ステンシルＴＩＲを実行することができる。たとえば、ＧＰＵ１２は、画像のパスの各アンチエイリアス画素のカバレージ値を決定するためのサンプリングレートを示すステンシルパラメータを決定することができる。ＧＰＵ１２はまた、ステンシルパラメータとは別々に、パスの各アンチエイリアス画素のためのメモリ割当てを示すレンダターゲットパラメータを決定することができる。ＧＰＵ１２は、ステンシルパラメータとレンダターゲットパラメータとを使用してパスをレンダリングすることができる。

【００５８】

[0068]いくつかの例では、ＧＰＵ１２は、画素に割り当てられるメモリの量よりも高いレートでステンシルを実行することができる。たとえば、１６ｘのＭＳＡＡに関して、ＧＰＵ１２は、スーパーサンプリングされる、たとえば、各画素が１６個のサンプルを有するステンシル動作を実行することができる。ＧＰＵ１２は、ステンシルテストにパスした（たとえば、パスの内側にあると決定された）画素のサンプルの数に基づいて画素ごとにカバレージ値を計算することによって所与のパスをレンダリングすることができる。本開示の態様によれば、ステンシルが１６ｘでサンプリングされ得るにもかかわらず、ＧＰＵ１２、画素のためのレンダターゲットは１ｘでサンプリングされ得る。

【００５９】

[0069]さらに、本開示の態様によれば、ＧＰＵ１２は、バウンディングボックスを前処理する必要なしに単一のレンダリングパスでパスをフィルすることができる。たとえば、ＧＰＵ１２は、ステンシル動作中にバウンディングボックスを決定することができる。この例では、ＧＰＵ１２が、（たとえば、画素をシェーディングすることなしに）ステンシル中にプリミティブをレンダリングするので、ＧＰＵ１２は、パスのプリミティブの最外点（たとえば、最外境界点）を決定することができる。いくつかの例では、ＧＰＵ１２は、（パスの相対上部にある）上位点と、（パスの相対下部にある）下位点と、（パスの右

端点にある）右点と、（パスの左端点にある）左点とを決定することができる。ＧＰＵ１２は、ステンシル中に決定された最外点を使用してバウンディングボックスを決定することができる。すなわち、ＧＰＵ１２は、パスのプリミティブのすべてを包含するバウンディングボックスを決定することができる。いくつかの例では、バウンディングボックスは、２つの三角形から構成され得る。

【００６０】

[0070]バウンディングボックスを完了した後に、ＧＰＵ１２は、さらに、バウンディングボックスの上でステンシルＴＩＲを実行することによって、（たとえば、同じレンダリングパス中で）バウンディングボックスを処理することができる。すなわち、上記のように、ＧＰＵ１２は、各画素のカバレージ値を決定し、ステンシル内に位置するとＧＰＵ１２が決定した画素をシェーディングすることができる。この例では、ＧＰＵ１２は、画素に対して別個の深度テストを実行する必要がない。

【００６１】

[0071]ストロークに関して、ＧＰＵ１２は、いくつかの事例では、ストロークされたパスをダッシングすることができる。すなわち、ＧＰＵ１２は、ストークしたパスのための複数のセグメントを決定することができ、したがって、レンダリングされたパスが破線として現れる。一般に、ＧＰＵ１２は、順番にダッシングされたパスのセグメントを決定することができる。たとえば、ＧＰＵ１２は、パスの次のセグメントに移る前に、１つのセグメントをレンダリングするためにＣＰＵ６からコマンドを受け取ることができる。そのようなプロセスは、並列性（たとえば、特定の時間インスタンスにおいて２つ以上のセグメントをラスタライズおよび／またはシェーディングすること）を阻止し得、ＧＰＵ１２がパスを単独でレンダリングするのを防ぎ得る。

【００６２】

[0072]本開示の態様によれば、ＧＰＵ１２は、パスの各セグメントのロケーション（ならびにパスの長さ）を決定し、レンダリング中に長さ情報を適用することができる。たとえば、ＧＰＵ１２は、破線の複数の順序付きセグメントの各セグメントのためのテクスチャオフセットを決定することができる。いくつかの事例では、セグメント順序は、以下でより詳細に説明するように、ジオメトリシェーディング中に決定され得る。この例では、複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットは、現在のセグメントより順序が前のセグメントの長さの累積に基づき得る。ＧＰＵ１２はまた、セグメントのロケーションを決定するために、各セグメントにテクスチャオフセットを適用することを含めてセグメントをピクセルシェーディングすることができる。たとえば、たとえば、ＧＰＵ１２は、セグメントのロケーションに基づいてセグメントが可視であるのか不可視であるのかを決定することができる。ＧＰＵ１２は、可視のセグメントについて保持し（たとえば、色を決定し）、可視でない（たとえば、可視ダッシュ間の空間である）ダッシュのセグメントを破棄することができる。このようにして、ＧＰＵ１２は、たとえば、ＣＰＵ６からダッシングコマンドを受け取ることなしに、破線のパスレンダリングを実行することができる。

【００６３】

[0073]パスレンダリングに関して説明したが、上記で説明したプレフィックス総和動作はベクタグラフィックスに限定されない。たとえば、プレフィックス総和を決定するための技法は、ＧＰＵ１２が累積値を追跡するあらゆる適用例において使用され得る。説明のための一例では、ＧＰＵ１２は、勾配を決定するときに上記で説明したプレフィックス総和動作を実行することができる。たとえば、画像処理中に、勾配を作成することは、色を決定するために、何らかの長さ情報の累積を必要とし得る。この例では、ＧＰＵ１２は、長さ情報を決定するために、上記で説明したプレフィックス総和動作を適用することができる。

【００６４】

[0074]本開示で説明するパスレンダリング技法は、たとえば、ＣＰＵ６と、ＧＰＵ１２と、メモリ１０とを含めて、図１に示されるコンピューティングデバイス２の構成要素の

うちのいずれかの中で実装され得る。いくつかの例では、パスレンダリング技法は、（たとえば、図３に関して説明するようにＧＰＵ１２のグラフィックスパイプラインにおいて）ＧＰＵ１２によって完全にまたはほぼ完全に実装され得る。追加の例では、ＣＰＵ６は、本開示のパスレンダリング技法を実行するＧＰＵ１２内のパスレンダリングパイプラインを実装するために、グラフィックスパイプラインの状態を構成して、シェーダプログラムをグラフィックスパイプラインと結合させるための技法を実装することができる。さらなる例では、ＣＰＵ６は、レンダリングされることになるパスを示すデータを、１つまたは複数のパスをレンダリングするためにＧＰＵ１２によってアクセスされ得る１つまたは複数のバッファ（たとえば、１つまたは複数の頂点バッファ）内に配置するように構成され得る。

【００６５】

[0075]図２は、図１のコンピューティングデバイス２のＣＰＵ６、ＧＰＵ１２、およびメモリ１０をさらに詳細に示すブロック図である。図２に示すように、ＣＰＵ６はＧＰＵ１２とメモリ１０とに通信可能に結合され、ＧＰＵ１２はＣＰＵ６とメモリ１０とに通信可能に結合される。いくつかの例では、ＧＰＵ１２は、ＣＰＵ６によってマザーボードに統合され得る。追加の例では、ＧＰＵ１２は、ＣＰＵ６を含むマザーボードのポート内にインストールされたグラフィックスカード上で実装され得る。さらなる例では、ＧＰＵ１２は、ＣＰＵ６と相互作用するように構成された周辺デバイス内に組み込まれることが可能である。追加の例では、ＧＰＵ１２は、システムオンチップ（ＳｏＣ）を形成するＣＰＵ６と同じマイクロチップ上に配置され得る。

【００６６】

[0076]ＣＰＵ６は、ソフトウェアアプリケーション２４と、グラフィックスＡＰＩ２６と、ＧＰＵドライバ２８と、オペレーティングシステム３０とを実行するように構成される。ソフトウェアアプリケーション２４は、グラフィックス画像を表示させる１つもしくは複数の命令および／または非グラフィックスタスク（たとえば、汎用コンピューティングタスク）をＧＰＵ１２上で実行させる１つもしくは複数の命令を含み得る。ソフトウェアアプリケーション２４は、グラフィックスＡＰＩ２６に対する命令を発行することができる。グラフィックスＡＰＩ２６は、ソフトウェアアプリケーション２４から受け取った命令をＧＰＵドライバ２８によって消費され得るフォーマットに変換するランタイムサービスであり得る。ＧＰＵドライバ２８は、グラフィックスＡＰＩ２６を介して、ソフトウェアアプリケーション２４から命令を受け取って、それらの命令にサービス提供するためにＧＰＵ１２の演算を制御する。たとえば、ＧＰＵドライバ２８は、１つまたは複数のコマンド３８を構築して、コマンド３８をメモリ１０内に配置して、コマンド３８を実行するようにＧＰＵ１２に命令することができる。いくつかの例では、ＧＰＵドライバ２８は、コマンド３８をメモリ１０内に配置して、オペレーティングシステム３０、たとえば、１つまたは複数のシステム呼出しを介してＧＰＵ１２と通信することができる。

【００６７】

[0077]ＧＰＵ１２は、コマンドエンジン３２と、１つまたは複数の処理ユニット３４とを含む。いくつかの例では、１つまたは複数の処理ユニット３４は、３Ｄグラフィックスレンダリングパイプライン、たとえば、ＤＸ　１１グラフィックスレンダリングパイプライン（すなわち、ＤＸ　１１グラフィックスＡＰＩに準拠する３Ｄグラフィックスパイプライン）を形成および／または実装することができる。

【００６８】

[0078]コマンドエンジン３２は、（たとえば、メモリ１０を介して）ＣＰＵ６からコマンドを受け取って、ＧＰＵ１２にそれらのコマンドを実行させるように構成される。状態コマンドを受け取ることに応答して、コマンドエンジン３２は、状態コマンドに基づいて、ＧＰＵ１２内の１つもしくは複数の状態レジスタを特定の値に設定するように、および／または状態コマンドに基づいて、固定関数処理ユニット３４のうちの１つもしくは複数を構成するように構成され得る。描画呼出しコマンドを受け取ることに応答して、コマンドエンジン３２は、処理ユニット３４に、レンダリングされることになる１つまたは複数

のパスセグメントのジオメトリを定義するデータに基づいて、およびレンダリングされることになるパスセグメントの各々のためのパスセグメントのタイプを示すデータに基づいて１つまたは複数のパスセグメントをレンダリングさせるように構成され得る。いくつかの例では、レンダリングされることになる１つまたは複数のパスセグメントのジオメトリを定義するデータと、パスセグメントの各々のためのパスセグメントのタイプを定義するデータとは、メモリ１０中の１つまたは複数の頂点データ構造に記憶され得る。コマンドエンジン３２は、シェーダプログラム結合コマンドを受け取って、それらのシェーダプログラム結合コマンドに基づいて、特定のシェーダプログラムをプログラマブル処理ユニット３４のうちの１つまたは複数にロードすることも可能である。

【００６９】

[0079]処理ユニット３４は、その各々がプログラマブル処理ユニットまたは固定関数処理ユニットであり得る、１つもしくは複数の処理ユニットを含み得る。プログラマブル処理ユニットは、たとえば、ＣＰＵ６からＧＰＵ１２上にダウンロードされた１つまたは複数のシェーダプログラムを実行するように構成されたプログラマブルシェーダユニットを含み得る。いくつかの例では、シェーダプログラムは、たとえば、ＯｐｅｎＧＬ Ｓｈａｄｉｎｇ Ｌａｎｇｕａｇｅ（ＧＬＳＬ）、Ｈｉｇｈ Ｌｅｖｅｌ Ｓｈａｄｉｎｇ Ｌａｎｇｕａｇｅ（ＨＬＳＬ）、Ｃ ｆｏｒ Ｇｒａｐｈｉｃｓ（Ｃｇ）シェーディング言語など、ハイレベルシェーディング言語で書き込まれたプログラムのコンパイルバージョンであり得る。

【００７０】

[0080]いくつかの例では、プログラマブルシェーダユニットは、並列して動作するように構成された複数の処理ユニット、たとえば、ＳＩＭＤパイプラインを含み得る。プログラマブルシェーダユニットは、シェーダプログラム命令を記憶するプログラムメモリと、実行状態レジスタ、たとえば、実行されているプログラムメモリ内の現在の命令またはフェッチされることになる次の命令を示すプログラムカウンタレジスタとを有し得る。処理ユニット３４中のプログラマブルシェーダユニットは、たとえば、頂点シェーダユニット、ピクセルシェーダユニット、ジオメトリシェーダユニット、ハルシェーダユニット、ドメインシェーダユニット、テッセレーション制御シェーダユニット、テッセレーション評価シェーダユニット、コンピュートシェーダユニット、および／またはユニファイドシェーダユニットを含み得る。図２に示されるように、処理ユニット３４はまた、バウンディングボックスユニット４０とプレフィックス総和ユニットとを含み得る。

【００７１】

[0081]固定関数処理ユニットは、ある種の機能を実行するために配線接続されたハードウェアを含み得る。固定関数ハードウェアは、１つまたは複数の制御信号を介して、たとえば、異なる機能を実行するように構成され得るが、固定関数ハードウェアは、通常、ユーザコンパイルプログラムを受け取ることができるプログラムメモリを含まない。いくつかの例では、処理ユニット３４内の固定関数処理ユニットは、たとえば、デプステスト、シザーテスト、アルファブレンディングなど、ラスタ演算を実行する処理ユニットを含み得る。

【００７２】

[0082]メモリ１０は、パスデータ３６と、１つまたは複数のコマンド３８とを記憶することができる。いくつかの例では、パスデータ３６は、複数の頂点（すなわち、制御点）として、メモリ１０内に割り当てられた１つまたは複数の頂点バッファ内に記憶され得る。いくつかの例では、パスデータは、パッチリストデータ構造（たとえば、４制御点パッチリスト）内に記憶され得る。コマンド３８は、１つまたは複数のコマンドバッファ（たとえば、リングバッファ）内に記憶され得る。ＣＰＵ６（たとえば、オペレーティングシステム３０を介したＧＰＵドライバ２８）は、ＧＰＵ１２によって消費するために、パスデータ３６とコマンド３８とをメモリ１０内に配置することができる。ＧＰＵ１２（たとえば、コマンドエンジン３２）は、メモリ１０内に記憶されたコマンド３８を検索および実行することができる。

【００７３】
[0083]パスデータ３６が頂点（たとえば、制御点）として記憶される例では、頂点はレンダリングされることになるパスセグメントを形状的に定義する１つまたは複数の属性を含み得る。たとえば、線の場合、パッチ制御リスト内の頂点は、線の終点に関する座標（たとえば、（ｘ０，ｙ０）および（ｘ１，ｙ１））を示すデータを含み得る。３次ベジェ曲線の場合、パッチ制御リスト内の頂点は、その曲線を定義する４つの制御点の座標（たとえば、（ｘ０，ｙ０）、（ｘ１，ｙ１）、（ｘ２，ｙ２）、（ｘ３，ｙ３））を示すデータを含み得る。２次ベジェ曲線の場合、パッチ制御リスト内の頂点は、４つの制御点の代わりに、３つの制御点に関する座標を示すデータを含み得る。楕円弧の場合、パッチ制御リスト内の頂点は、楕円弧の終点パラメータ表示を示すデータ、または楕円弧の中心パラメータ表示を示すデータを含み得る。

【００７４】
[0084]いくつかの例では、レンダリングされることになるパスセグメントを形状的に定義する１つまたは複数の属性は解像度と無関係であり得る。言い換えれば、パスセグメントを形状的に定義する属性は、パスセグメントをレンダリングするときに実行されることになるテッセレーションの量と無関係であり得、および／またはパスセグメントをレンダリングするときに生成されることになる頂点の数量と無関係であり得る。

【００７５】
[0085]ＣＰＵ６は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を頂点バッファ内の１つまたは複数の、場合によっては未使用の頂点属性内に配置することも可能である。いくつかの例では、異なるパスセグメントタイプは、ベクタグラフィックスＡＰＩによって定義されたパスセグメントタイプのセットに対応し得、ソフトウェアアプリケーション２４によって使用するために利用可能である。いくつかの例では、異なるパスセグメントタイプは、ＯｐｅｎＶＧ　ＡＰＩによって定義されたパスセグメントタイプのセットに対応し得る。

【００７６】
[0086]コマンド３８は、１つもしくは複数の状態コマンドおよび／または１つもしくは複数の描画呼出しコマンドを含み得る。状態コマンドは、たとえば、描画色、フィル色、ストローク色など、ＧＰＵ１２内の状態変数のうちの１つまたは複数を変更するようにＧＰＵ１２に命令することができる。いくつかの例では、状態コマンドは、パスをレンダリングすることと関連付けられた１つまたは複数の状態変数を設定するように構成されたパスレンダリング状態コマンドを含み得る。たとえば、状態コマンドは、レンダリングされることになるパスがフィルされるか、ストロークされるか、またはそれらの両方かを示すように構成されたペイントモードコマンドを含み得る。別の例として、状態コマンドは、フィル動作のために使用されることになる色を指定するフィル色コマンドおよび／またはストローク動作のために使用されることになる色を指定するストローク色コマンドを含み得る。さらなる例として、状態コマンドは、たとえば、ストローク幅、エンドキャップスタイル（たとえば、円形、方形）、ライン接合スタイル（たとえば、マイター、ラウンド、ベベル）、マイターリミットなど、ストローク動作に関する１つまたは複数のパラメータを指定することができる。いくつかの例では、１つもしくは複数の状態パラメータを設定するために状態コマンドを使用することに加えて、またはその代わりに、描画呼出しコマンドを使用することによって、あるいはパスデータ３６を含む頂点バッファ内に状態インジケータを配置することによって、状態パラメータのうちの１つもしくは複数が設定され得る。

【００７７】
[0087]描画呼出しコマンドは、メモリ１０内に記憶された（たとえば、頂点バッファ内で定義された）１つまたは複数の頂点のグループによって定義された形状をレンダリングするようにＧＰＵ１２に命令することができる。いくつかの例では、描画呼出しコマンドは、ＧＰＵ１２にメモリ１０の定義されたセクション（たとえば、頂点バッファまたはパスデータ３６）内に記憶された頂点のすべてをレンダリングさせることができる。言い換

えれば、ＧＰＵ１２が描画呼出しコマンドを受け取ると、メモリ１０の定義されたセクション（たとえば、頂点バッファまたはパスデータ３６）内の頂点によって表された形状およびプリミティブをレンダリングするための制御がＧＰＵ１２に渡される。

【００７８】

[0088]描写呼出しコマンドは、３Ｄ描写呼出しコマンドおよびパスレンダリング描写呼出しコマンドのうちの１つまたは両方を含み得る。３Ｄレンダリング描画呼出しコマンドの場合、頂点バッファ内の１つまたは複数の頂点のグループによって定義された形状は、レンダリングされることになる１つまたは複数の３Ｄグラフィックスプリミティブ（たとえば、点、線、三角形、クアドララテラル（quadralaterals）、トライアングルストリップ、パッチなど）に対応し得、３Ｄレンダリング描画呼出しコマンドは、１つまたは複数の３ＤグラフィックスプリミティブをレンダリングするようにＧＰＵ１２に命令することができる。パスレンダリング描画呼出しコマンドの場合、頂点バッファ内の１つまたは複数の頂点のグループによって定義された形状は、レンダリングされることになる１つまたは複数のパスプリミティブ（たとえば、ラインセグメント、楕円弧、２次ベジェ曲線、および３次ベジェ曲線など）に対応し得、パスレンダリング描画呼出しコマンドは、１つまたは複数のパスプリミティブをレンダリングするようにＧＰＵ１２に命令することができる。いくつかの例では、ＧＰＵ１２によってレンダリングされることが可能なパスプリミティブは、本開示で説明する異なるタイプのパスセグメントに対応し得る。

【００７９】

[0089]いくつかの例では、本開示で説明するパスレンダリング技法は、たとえば、グラフィックスＡＰＩ２６と、ＧＰＵドライバ２８と、コマンドエンジン３２と、処理ユニット３４とを含めて、図２に示す構成要素のうちのいずれかの内で実装され得る。さらなる例では、パスレンダリング技法のすべてまたは大部分は、処理ユニット３４によって形成されたＧＰＵ１２内のグラフィカルパイプライン内で実装され得る。追加の例では、ＣＰＵ６のソフトウェアアプリケーション２４、グラフィックスＡＰＩ２６および／またはＧＰＵドライバ２８は、本開示で説明するパスレンダリング技法を実行するＧＰＵ１２内のパスレンダリングパイプラインを実装するために、グラフィックスパイプラインの状態を構成して、シェーダプログラムをグラフィックスパイプラインと結合させるための技法を実装することができる。さらなる例では、ＣＰＵ６のソフトウェアアプリケーション２４、グラフィックスＡＰＩ２６および／またはＧＰＵドライバ２８は、レンダリングされることになるパスを示すデータを、１つまたは複数のパスをレンダリングするためにＧＰＵ１２によってアクセスされ得る１つまたは複数のバッファ（たとえば、１つまたは複数の頂点バッファ）内に配置するように構成され得る。

【００８０】

[0090]本開示の態様によれば、処理ユニット３４は、バウンディングボックスユニット４０を含む。バウンディングボックスユニット４０は、バウンディングボックスを決定するための１つまたは複数のプログラマブルおよび／または固定機能ユニットを含み得る。たとえば、本開示の技法は、（たとえば、以下の図３に関してより詳細に説明するように）バウンディングボックスを決定することと、単一のレンダリングパスでバウンディングボックスをレンダリングすることとを含む。ＧＰＵ１２がパスフィル動作を実行するとき、バウンディングボックスユニット４０は、パスの境界を決定することを担当し得る。

【００８１】

[0091]本開示の態様によれば、バウンディングボックスユニット４０は、ＡＰＩ呼出しを使用して開始され得る。たとえば、グラフィックスＡＰＩ２６は、パスのレンダリング中に、バウンディングボックスユニット４０の使用をトリガするための１つまたは複数の命令を含み得る。ＡＰＩ呼出しにより、ＧＰＵ１２は、バウンディングボックスユニット４０がバウンディングボックスを決定するまで、プリミティブのシェーディングをスキップすることが可能になり得る。ＧＰＵ１２は、次いで、上記のように、バウンディングボックスの上でステンシルＴＩＲを実行することができる。さらに、バウンディングボックスユニット４０を組み込むことによって、ＧＰＵ１２は、深度バッファを使用せずに単一

のパスでパスをフィルすることができる。

【００８２】

[0092]バウンディングボックスユニット４０により、ＧＰＵ１２は、バウンディングボックスを前処理することなしにパスをフィルすることが可能になり得る。たとえば、バウンディングボックスユニット４０は、たとえば、ＣＰＵ６において、制御多角形を使用してバウンディングボックスを決定することができる。すなわち、バウンディングボックスユニット４０は、生成されたプリミティブのすべての境界に基づいてバウンディングボックスを決定することができる。

【００８３】

[0093]本開示の態様によれば、ＧＰＵ１２は、バウンディングボックスユニット４０がバウンディングボックスを決定するまで、プリミティブのシェーディングをスキップするように構成され得る。すなわち、バウンディングボックスの生成中に、ＧＰＵ１２は、プリミティブをシェーディングすることなしにＧＰＵ１２のステンシルバッファにパスのプリミティブを書き込むことができる。バウンディングボックスユニット４０を組み込むことによって、ＧＰＵ１２は、深度バッファを使用せずに単一のパスでパスをフィルすることができる。たとえば、ＧＰＵ１２は、バウンディングボックスの上でステンシルＴＩＲを実行することができる。

【００８４】

[0094]説明のための一例では、バウンディングボックスユニット４０によって決定されたバウンディングボックスをＧＰＵ１２がラスタライズした後、ＧＰＵ１２は、バウンディングボックス中の各画素のカバレージ値を決定することができる。いくつかの例では、ＧＰＵ１２は、画素のクワッド（quad）（一度に４つの画素）のカバレージ値を決定することができる。そのような例では、処理するための画素波を形成する前に、ＧＰＵ１２は、クワッドの各画素のサンプルに対してステンシルテストを実行することができる。ＧＰＵ１２は、テストの結果に基づいて各画素のカバレージマスクを更新することができる。このカバレージ値は、ｓｔｅｎｃｉｌｅｄ＿ＴＩＲ属性と呼ばれる場合があり、その場合、ＧＰＵ１２は、シェーディング中に使用することができる。たとえば、各画素のＩｎｐｕｔＣｏｖｅｒａｇｅ値は、ｓｔｅｎｃｉｌｅｄ＿ＴＩＲに基づき得る。たとえば、ＧＰＵ１２は、ステンシルテストをパスした各画素をピクセルシェーディングする（たとえば、着色する）ことができる（たとえば、ここで、画素のより多くのサンプルが可視である（シェーディングされている）とき、画素はステンシルテストをパスする）。すなわち、ＧＰＵ１２は、分散プロセッサ（ＤＰｒｏｃ）からサンプラに（ＩｎｐｕｔＣｏｖｅｒａｇｅのための）ステンシルテストの後に（中心についての）カバレージマスクとサンプルマスクの両方をパスすることができる。

【００８５】

[0095]本開示のいくつかの態様によれば、ＡＰＩ呼出しは、レンダリングのステンシルＴＩＲモードをサポートするために使用され得る。たとえば、グラフィックスＡＰＩ２６は、パスのレンダリング中に、ステンシルＴＩＲの使用をトリガするための１つまたは複数の命令を含み得る。ステンシルＴＩＲがアクティブであるとき、（ＧＰＵ１２のメモリおよび／またはメモリ１０中に割り当てられ得る）色バッファと深度／ステンシルバッファとは異なり得る。たとえば、ＭＳＡＡを実行するとき、ＧＰＵ１２は、１ｘのＭＳＡＡである色バッファと１６ｘのＭＳＡＡであるステンシルバッファとにレンダリングすることができる。

【００８６】

[0096]本開示の他の態様によれば、処理ユニット３４はまた、ダッシングされたセグメントをレンダリングすること、たとえば、ダッシングされたパスをストロークすることを行うためのプレフィックス総和ユニット４２を含む。プレフィックス総和ユニット４２は、破線の複数の順序付きセグメントの各セグメントのためのテクスチャオフセットを決定することができる。いくつかの例では、テッセレーションまたはジオメトリシェーダ段階は、セグメントを生成するときにセグメント順序を決定することができる。複数の順序付

きセグメントの現在のセグメントのためのテクスチャオフセットは、現在のセグメントより順序が前のセグメントの長さの累積に基づき得る。プレフィックス総和ユニット４２は、ピクセルシェーダ段階などのシェーダ段階にテクスチャオフセットを与えることができる。シェーダ段階は、テクスチャオフセットを適用し、適切なロケーションでセグメントをレンダリングすることができる。

【００８７】

[0097]したがって、プレフィックス総和装置４２は、破線のセグメントの長さを累積する１つまたは複数のプログラマブルまたは固定機能ユニットを含み得る。いくつかの例では、プレフィックス総和装置４２は、ラスタライザ段階に組み込まれ得る。たとえば、ＧＰＵ１２は、パスをテッセレートすることができ、ジオメトリシェーダ段階は、パスの長さを決定することができる。他の例では、長さは、１つまたは複数の他のシェーダユニットによって決定され得る。たとえば、本開示の態様によれば、プレフィックス総和装置４２は、（点プリミティブのサイズを示す）属性ｐｏｉｎｔｓｉｚｅのシステム解釈値と同様の方法でｌｉｎｅｌｅｎｇｔｈ値を計算することができる。すなわち、ｌｉｎｅｌｅｎｇｔｈは、ダッシングされたパターン中の（フラグメントとも呼ばれる場合がある）セグメントのロケーションを示すシステム解釈値であり得る。

【００８８】

[0098]ＧＰＵ１２の（たとえば、以下の図３に関して説明する）ピクセルシェーダが、プレフィックス総和したｌｉｎｅｌｅｎｇｔｈ値を受け取ると、ピクセルシェーダは、ダッシュパターン中でシェーディングされているフラグメントのロケーションを決定することができる。ピクセルシェーダは、次いで、決定されたロケーションに基づいて、（フラグメントが可視ダッシュの一部を形成する場合）フラグメントを保持するか、あるいは（フラグメントが可視ダッシュの一部でない場合）フラグメントを破棄することができる。いずれの場合も、プレフィックス総和ユニット４２は、プレフィックス総和として長さ情報を累積し、ピクセルシェーダなどのダウンストリームシェーダ段階にテクスチャオフセットとしてプレフィックス総和を与えることができる。

【００８９】

[0099]レンダリング中に、ＧＰＵ１２は、ｐｒｅｆｉｘ＿ｓｕｍパラメータをリセットするために（以下の図３に関してより詳細に説明するように、ハルシェーダ、テッセレータ、および／またはドメインシェーダを含み得る）テッセレーションエンジン（ＴＳＥ）にイベントｐｒｅｓｕｍ＿ｓｔａｒｔを送ることができる。プリミティブごとに、プレフィックス総和ユニット４２は、新しい値としてｐｒｅｆｉｘ＿ｓｕｍにプリミティブのスカラー値（たとえば、ｐｏｉｎｔｓｉｚｅと同じフィールド）を追加することができる。プレフィックス総和ユニット４２は、画素ごとの古いプレフィックス総和値をテクスチャオフセットとしてパスすることができる。

【００９０】

[0100]いくつかの例では、テッセレーションエンジンは、ｐｒｅｆｉｘ＿ｓｕｍパラメータを累積するためにレジスタを組み込むことができる。プレフィックス総和ユニット４２は、イベントｐｒｅｓｕｍ＿ｓｔａｒｔによってレジスタをリセットすることができる。テッセレーションエンジンは、（テクスチャオフセットを送ることと同様であり得る）プリミティブフェイスネス（faceness）と同様の重心平面インターフェース中でバックエンド（ＲＢ）をレンダリングするためにプリミティブごとの属性としてｐｒｅｆｉｘ＿ｓｕｍをパスする。この例では、属性は、このプリミティブごとの属性を表すインターフェースを高レベルシーケンサ（ＨＬＳＱ）に与えるためにＲＢに追加され得る。

【００９１】

[0101]図３は、本開示のパスレンダリング技法を実行することができる例示的なグラフィックスパイプライン４３を示す概念図である。いくつかの例では、グラフィックスパイプラインは、Ｍｉｃｒｏｓｏｆｔ（登録商標）ＤｉｒｅｃｔＸ（ＤＸ）１１グラフィックスパイプラインに対応し得る。図３に示すように、グラフィックスパイプライン４３は、入力アセンブラ（ＩＡ）４４と、頂点シェーダ（ＶＳ）４６と、ハルシェーダ（ＨＳ）４

８と、テッセレータ５０と、ドメインシェーダ（ＤＳ）５２と、ジオメトリシェーダ（ＧＳ）５４と、ラスタライザ５６と、ピクセルシェーダ（ＰＳ）５８と、出力統合器６０とを含む複数の処理段階を含む。ハルシェーダ４８、テッセレータ５０、およびドメインシェーダ５２は、グラフィックスパイプライン４３のテッセレーション段階６２を形成し得る。さらに、パイプライン４３はまた、リソースブロック６４を含む。いくつかの例では、パイプライン４３は、以下に述べるように、ＧＰＵ１２によって実装され、および／またはＧＰＵ１２中に組み込まれ得る。

【００９２】

[0102]リソースブロック６４は、たとえば、１つもしくは複数のテクスチャおよび／または１つもしくは複数のバッファなど、グラフィックスパイプライン４３によって使用される１つもしくは複数のメモリリソースに対応し得る。リソースブロック６４は、グラフィカルパイプライン４３内の処理段階のうちの１つもしくは複数によって処理されることになる入力データおよび／またはグラフィックスパイプライン４３内の処理段階のうちの１つもしくは複数からの出力データを記憶することができる。一例として、リソースブロック６４は、本開示で説明するパスフィル動作を実行するために使用されるステンシルバッファを記憶することができる。別の例として、リソースブロック６４は、本開示で説明するようにパスセグメントに関するフィル領域のラスタライズされたバージョンおよび／またはパスセグメントに関するストローク領域のラスタライズされたバージョンを保持するフレームバッファを記憶することができる。いくつかの例では、リソースブロック６４を形成するメモリリソースは、コンピューティングデバイス２のメモリ１０および／またはＧＰＵキャッシュ１４内に存在し得る。

【００９３】

[0103]図３に示す直角の処理段階は固定関数処理段階を表し、図３に示す丸角の処理段階はプログラマブル処理段階を表す。たとえば、図３に示すように、入力アセンブラ４４、テッセレータ５０、ラスタライザ５６、および出力統合器６０は固定関数処理段階であり、頂点シェーダ４６、ハルシェーダ４８、ドメインシェーダ５２、ジオメトリシェーダ５４、およびピクセルシェーダ５８はプログラマブル処理段階である。プログラマブル段階の各々は、特定のタイプのシェーダプログラムを実行するように構成され得る。たとえば、頂点シェーダ４６は、頂点シェーダプログラムを実行するように構成され得、ハルシェーダ４８は、ハルシェーダプログラムを実行するように構成され得る、等々である。異なるタイプのシェーダプログラムの各々は、ＧＰＵ１２の共通シェーダユニット上、または１つもしくは複数の特定のタイプのシェーダプログラムを実行するための専用である１つもしくは複数の専用シェーダユニット上のいずれかで実行することができる。

【００９４】

[0104]図３に示すように、入力アセンブラ４４、頂点シェーダ４６、ハルシェーダ４８、ドメインシェーダ５２、ジオメトリシェーダ５４、ピクセルシェーダ５８、および出力マージャ６０はリソースブロック６４に通信可能に結合される。入力アセンブラ４４、頂点シェーダ４６、ハルシェーダ４８、ドメインシェーダ５２、ジオメトリシェーダ５４、ピクセルシェーダ５８、および出力統合器６０はリソースブロック６４から入力データを検索ならびに／または受け取るように構成される。ジオメトリシェーダ５４および出力統合器６０は、出力データをリソースブロック６４に書き込むように構成される。グラフィックスパイプライン４３内の処理段階とリソースブロック６４との間の通信の上述の構成は、グラフィックスパイプライン４３の処理段階とリソースブロック６４との間の通信がどのように構成され得るかの単なる一例である。他の例では、グラフィックスパイプライン４３の処理段階とリソースブロック６４との間により多くのもしくはより少ない一方向および／または双方向の通信チャネルが提供され得る。

【００９５】

[0105]ＤｉｒｅｃｔＸ　１１グラフィックスパイプラインの一般的な動作に関する追加の背景情報は、ｈｔｔｐ：／／ｍｓｄｎ．ｍｉｃｒｏｓｏｆｔ．ｃｏｍ／ｅｎ－ｕｓ／ｌｉｂｒａｒｙ/ｗｉｎｄｏｗｓ/ｄｅｓｋｔｏｐ/ｆｆ４７６８８２%２８ｖ＝ｖｓ．８５

%２９．ａｓｐｘにあり得る。ＤｉｒｅｃｔＸ １１グラフィックスパイプラインの一般的な動作に関するさらなる情報は、Ｚｉｎｋら、「Ｐｒａｃｔｉｃａｌ Ｒｅｎｄｅｒｉｎｇ ＆ Ｃｏｍｐｕｔａｔｉｏｎ ｗｉｔｈ Ｄｉｒｅｃｔ３Ｄ １１」、ＣＲＣ Ｐｒｅｓｓ（２０１１年）に見出すことができる。

【００９６】

[0106]２つの主要なパスレンダリング動作は、（１）パスセグメントをフィルすることと、（２）パスセグメントをストロークすることとを含み得るいくつかの事例では、フィル動作は、以下のステップを概して伴い得る２パス手法を利用することができる。

【００９７】

パス１

１．複数のラインセグメントにパスセグメントをテッセレートする。

【００９８】

２．ラインセグメントごとに三角形プリミティブを生成する。

【００９９】

３．ステンシルバッファに三角形プリミティブのすべてをレンダリングする。

【０１００】

パス２

４．ステンシルバッファを使用してパスセグメントに関するバウンディングボックスをレンダリングする。

【０１０１】

[0107]第１のパスの場合、ＣＰＵ６は、レンダリングされることになるパスセグメントを示すデータを頂点バッファの１つまたは複数の頂点内に配置することができる。いくつかの例では、頂点バッファは図２に示すパスデータ３６に対応し得る。頂点バッファ内の頂点に関するプリミティブトポロジは、いくつかの例では、パッチ制御リストであり得る。線の場合、パッチ制御リスト内の頂点は、線の終点に関する座標（たとえば、（ｘ０，ｙ０）および（ｘ１，ｙ１））を示すデータを含み得る。３次ベジェ曲線の場合、パッチ制御リスト内の頂点は、その曲線を定義する４つの制御点の座標（たとえば、（ｘ０，ｙ０）、（ｘ１，ｙ１）、（ｘ２，ｙ２）、（ｘ３，ｙ３））を示すデータを含み得る。２次ベジェ曲線の場合、パッチ制御リスト内の頂点は、４つの制御点の代わりに、曲線を定義する３つの制御点に関する座標を示すデータを含み得る。楕円弧の場合、パッチ制御リスト内の頂点は、楕円弧の終点パラメータ表示を示すデータ、または楕円弧の中心パラメータ表示を示すデータを含み得る。ＣＰＵ６は、レンダリングされることになるパスセグメントのタイプを示すデータをパッチ制御リストの、場合によっては未使用の頂点属性内に配置することも可能である。

【０１０２】

[0108]パスレンダリングを実行するためにＧＰＵ１２によって受け取られ、使用されるパスデータ３６の１つの例示的なフォーマットが次に説明される。これは、レンダリングされることになるパスおよび／またはレンダリングされることになるパスセグメントを示すデータがＣＰＵ６によってＧＰＵ１２にどのように提供され得るかの単なる一例であり、他の例が可能であり、本開示の範囲内であることを理解されたい。この例では、ＧＰＵ１２は４（４）つの制御点パッチリストプリミティブとして各パスセグメントを受け取る。この例では、パッチリスト内の頂点（たとえば、制御点）の各々は、それぞれの頂点（たとえば、制御点）に関する属性を定義する３（３）つの浮動属性を含む。

【０１０３】

[0109]ラインパスセグメントの場合、入力パスデータは、以下の形または類似の形をとることができる。

【数１】

```
{ XMFLOAT3(    X0,    Y0, 2.0f ) },
{ XMFLOAT3(    X1,    Y1, 1.0f ) },
{ XMFLOAT3(  0.0f,  0.0f, 1.0f ) },
{ XMFLOAT3(  0.0f,  0.0f, 1.0f ) },
```

【０１０４】
この例では、各行は４つの制御点パッチリストの頂点すなわち制御点を表し、括弧内の各パラメータは、それぞれの頂点すなわち制御点の属性を表す。この例では、第１の制御点の最後の属性は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を記憶する。具体的には、この例では、パスセグメントタイプインジケータは、パスセグメントがラインパスセグメントであることを意味する２．０ｆである。Ｘ０、Ｙ０、Ｘ１、Ｙ１はラインパスセグメントの終点に関する座標であり、この場合、（Ｘ０，Ｙ０）は第１の終点を表し、（Ｘ１，Ｙ１）は第２の終点を表す。
【０１０５】
[0110]この例では、残りの頂点および属性は、パスセグメントに関する他の属性を示すために使用されなくよく、および／または使用されてもよい。パスセグメントに関する他の属性は、たとえば、パスセグメントがオープンパスの始端であるかまたは終端であるかと、そのパスに関してパスセグメントが表示されるべきかどうかと、エンドキャップがパスセグメントの両方の終端上に配置されるべきかどうかと、もしあれば、何のタイプのエンドキャップが使用されるべきかと、接合がパスセグメントのいずれかの終端上に配置されるべきかどうかと、もしあれば、何のタイプの接合を使用するかと、を含み得る。
【０１０６】
[0111]３次ベジェパスセグメントに関する入力パスデータは、以下の形または類似の形をとることができる。
【数２】

```
{ XMFLOAT3(    X0,    Y0, 3.0f ) },
{ XMFLOAT3(    X1,    Y1, 1.0f ) },
{ XMFLOAT3(    X2,    Y2, 1.0f ) },
{ XMFLOAT3(    X3,    Y3, 1.0f ) },
```

【０１０７】
この例では、各行は４つの制御点パッチリストの頂点すなわち制御点を表し、括弧内の各パラメータは、それぞれの頂点すなわち制御点の属性を表す。この例では、第１の制御点の最後の属性は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を記憶する。具体的には、この例では、パスセグメントタイプインジケータは、パスセグメントが３次ベジェパスセグメントであることを意味する３．０ｆである。Ｘ０～Ｘ３およびＹ０～Ｙ３は、３次ベジェパスセグメントに関する制御点の座標であり、この場合、（Ｘ０，Ｙ０）は第１の制御点を表し、（Ｘ１，Ｙ１）は第２の制御点を表す、等々である。この例では、残りの頂点および属性は、パスセグメントに関する他の属性を示すために使用されなくよく、および／または使用されてもよい。パスセグメントに関する他の属性は、いくつかの例では、ラインパスセグメントに関して上記で説明した属性と同様の属性を含み得る。

【０１０８】
[0112]４つの制御点の代わりに、３つの制御点が提供され得ることを除いて、類似の入力が２次ベジェパスセグメントに関して使用されてよく、パスセグメントタイプインジケータは３次ベジェパスセグメントからのプリミティブと区別するために異なってよい。
【０１０９】
[0113]たとえば、２次ベジェパスセグメントに関する入力パスデータは、以下の形または類似の形をとることができる。
【数３】

```
{ XMFLOAT3(   X0,    Y0, 1.0f ) },
{ XMFLOAT3(   X1,    Y1, 1.0f ) },
{ XMFLOAT3(   X2,    Y2, 1.0f ) },
{ XMFLOAT3( 0.0f,  0.0f, 1.0f ) },
```

【０１１０】
この例では、各行は４つの制御点パッチリストの頂点すなわち制御点を表し、括弧内の各パラメータは、それぞれの頂点すなわち制御点の属性を表す。この例では、第１の制御点の最後の属性は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を記憶する。具体的には、この例では、パスセグメントタイプインジケータは、パスセグメントが２次ベジェパスセグメントであることを意味する１．０ｆである。Ｘ０～Ｘ２およびＹ０～Ｙ２は、２次ベジェパスセグメントに関する制御点の座標であり、この場合、（Ｘ０，Ｙ０）は第１の制御点を表し、（Ｘ１，Ｙ１）は第２の制御点を表す、等々である。この例では、残りの頂点および属性は、パスセグメントに関する他の属性を示すために使用されなくよく、および／または使用されてもよい。パスセグメントに関する他の属性は、いくつかの例では、ラインパスセグメントに関して上記で説明した属性と同様の属性を含み得る。
【０１１１】
[0114]いくつかの例では、楕円弧パスセグメントに関する入力パスデータは、楕円弧パスセグメントの中心パラメータ表示を示すデータを含み得る。たとえば、楕円弧パスセグメントに関する入力パスデータは、以下の形または類似の形をとることができる。
【数４】

```
{ XMFLOAT3(     X0,     Y0,  4.0f ) },
{ XMFLOAT3(     X1,     Y1,  1.0f ) },
{ XMFLOAT3(    c.x,    c.y,  1.0f ) },
{ XMFLOAT3( theta0, theta1, angle ) },
```

【０１１２】
この例では、各行は４つの制御点パッチリストの頂点すなわち制御点を表し、括弧内の各パラメータは、それぞれの頂点すなわち制御点の属性を表す。この例では、第１の制御点の最後の属性は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を記憶する。この例では、パスセグメントタイプインジケータは、それぞれ、大型時計回り（ＬＣＷ）楕円弧、大型反時計回り（ＬＣＣＷ）楕円弧、小型時計回り（ＳＣＷ）楕円弧、および小型反時計回り（ＳＣＣＷ）楕円弧に対応する４．０、４．１、４．２、または４．３のうちのいずれかであり得る。Ｘ０，Ｘ１およびＹ０，Ｙ１は、楕円弧パスセグメントの終点座標であり、この場合、（Ｘ０，Ｙ０）は弧の最初の終点を表し、（Ｘ１，Ｙ１）は弧の最終の終点を表す。加えて、ｔｈｅｔａ０は、（スケーリングされていない（unscaled）円上で測定された）楕

円弧の初期点の角度を表し、ｔｈｅｔａ１は、（スケーリングされていない円上で測定された）楕円弧の最終点の角度を表す。特に、上で指定された例示的な入力データ形式は中央パラメータ表示であるが、入力データ形式は、弧の最初の終点および最後の終点に関する座標（すなわち、（Ｘ０，Ｙ０）、（Ｘ１，Ｙ１））を依然として含み得る。いくつかの例では、そのような座標は、結果として生じる形状の水密性を確実にするために使用され得る。

【０１１３】

[0115]さらなる例では、楕円弧パスセグメントに関する入力パスデータは、楕円弧パスセグメントの終点パラメータ表示を示すデータを含み得る。たとえば、楕円弧パスセグメントに関する入力パスデータは、以下の形または類似の形をとることができる。

【数５】

```
{ XMFLOAT3(    X0,    Y0, 4.0f ) },
{ XMFLOAT3(    X1,    Y1, 1.0f ) },
{ XMFLOAT3(    rH,    rV, 1.0f ) },
{ XMFLOAT3( angle,  0.0f, 1.0f ) },
```

【０１１４】

この例では、各行は４つの制御点パッチリストの頂点すなわち制御点を表し、括弧内の各パラメータは、それぞれの頂点すなわち制御点の属性を表す。この例では、第１の制御点の最後の属性は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を記憶する。この例では、パスセグメントタイプインジケータは、それぞれ、大型時計回り（ＬＣＷ）楕円弧、大型反時計回り（ＬＣＣＷ）楕円弧、小型時計回り（ＳＣＷ）楕円弧、および小型反時計回り（ＳＣＣＷ）楕円弧に対応する４．０、４．１、４．２、または４．３のうちのいずれかであり得る。Ｘ０，Ｘ１およびＹ０，Ｙ１は、楕円弧パスセグメントの終点座標であり、この場合、（Ｘ０，Ｙ０）は弧の最初の終点を表し、（Ｘ１，Ｙ１）は弧の最終の終点を表す。加えて、ａｎｇｌｅは、（ｒｈ，ｒｖ）によるスケーリングに先立って測定されたｘ軸に対する楕円の反時計回り回転角度を表す。

【０１１５】

[0116]入力パスデータが終点パラメータ形式で表される楕円弧を含む例では、ＣＰＵ６は、いくつかの例では、レンダリングのためのＧＰＵ１２に楕円弧を示すデータを送る前に、終点パラメータ形式から中心パラメトリック形式に楕円弧の表現を変換することができる。たとえば、ＣＰＵ６は、楕円弧の終点パラメータ表示に基づいて、楕円弧の中心パラメータ表示を生成して、楕円弧の中央パラメータ表示をＧＰＵ１２に送ることができる。楕円弧に関する中央パラメータ表示は、上に指定された例示的な入力データ形式に準拠し得る。中央パラメータ表示は、次に、ＧＰＵ１２によってレンダリングする目的で接合プリミティブを生成するためにＣＰＵ６によって使用され得る、楕円弧のための終点接線（tangents）および／または法線を見出すためにＣＰＵ６によって使用され得る。

【０１１６】

[0117]いくつかの例では、ストローク動作は、エンドキャップと、接合と、オープンパスとを処理するために頂点パスデータ入力の３つの追加のフィールドを使用することができる。たとえば、ある種の頂点座標は、パスセグメントがオープンパスの始端であるか、オープンパスの終端であるか、およびパスセグメントが破棄され得る（たとえば、パスセグメントがオープンパスの終結パスセグメントである）かどうかを示すデータを記憶することができる。以下は、上述の頂点属性を含む例示的なテンプレートである。

【数６】

```
{ XMFLOAT3(    X0,    Y0, 2.0f ) },
{ XMFLOAT3(    X1,    Y1, 2.0f ) },
{ XMFLOAT3(  0.0f,  0.0f, 2.0f ) },
{ XMFLOAT3(  0.0f,  0.0f, 2.0f ) },
```

【０１１７】
このテンプレートでは、第２の頂点のｚ座標（たとえば、第３の座標、すなわち属性）に関する２．０ｆは、そのパスセグメントがオープンパスの始端であることを示し、そのパスセグメントの始端にエンドキャップ（すなわち、スタートキャップ）を入れるようにＧＰＵ１２に信号伝達することができる。第３の頂点のｚ座標に関する２．０ｆは、そのパスセグメントがオープンパスの終端であることを示し、そのパスセグメントの終端にエンドキャップを入れるようにＧＰＵ１２に信号伝達することができる。最後の頂点のｚ座標の２．０ｆは、現在のプリミティブが破棄される（たとえば、そのプリミティブがオープンパスの終結ラインまたはパスセグメントである）ことを示す。
【０１１８】
[0118]パスフィル動作を実行するために、入力アセンブラ４４は、パスデータ３６をメモリ１０から取得して、パスデータ３６によって指定されたパスセグメント（たとえば、パスプリミティブ）をレンダリングするために、そのパスデータをグラフィックパイプライン４３の後続の１つまたは複数の段階に渡す。たとえば、入力アセンブラ４４は、複数の頂点をメモリ１０内に記憶された頂点バッファから取得して、頂点シェーダ４６にそれらの頂点を処理させることができる。いくつかの例では、入力アセンブラ４４は、処理されることになる頂点を頂点シェーダ４６に直接的に渡すことができる。追加の例では、入力アセンブラ４４は、リソースブロック６４内の頂点バッファから、処理のために特定の頂点を検索するように頂点シェーダ４６に指示することができる。
【０１１９】
[0119]頂点シェーダ４６は、入力アセンブラ４４および／またはリソースブロック６４から受け取った頂点を処理して、頂点シェーダ４６によって処理された各入力頂点に関する出力頂点を生成するように構成される。たとえば、各入力頂点に関して、頂点シェーダ４６は、ＧＰＵ１２のシェーダユニット上で頂点シェーダプログラムのインスタンスを実行することができる。いくつかの例では、頂点シェーダ４６は、各入力頂点に対して「パススルー」頂点シェーダプログラムを実行することができる。「パススルー」頂点シェーダプログラムは、頂点シェーダ４６に、入力頂点ごとに、入力頂点に対応する頂点を出力させることができる。この場合、出力頂点が入力頂点と同じ属性を有する場合、出力頂点は、入力頂点に対応し得る。「パススルー」頂点シェーダプログラムを実装するために、いくつかの例では、頂点シェーダ４６は、同じ属性をもつ出力頂点を生成するために、各入力頂点に識別変換を適用することができる。頂点シェーダ４６によって受け取られた入力頂点、および頂点シェーダ４６によって生成された出力頂点は、あるいは、それぞれ、入力制御点および出力制御点と呼ばれる場合がある。
【０１２０】
[0120]さらなる例では、頂点シェーダ４６は、対応する入力頂点の入力属性と同一でない出力頂点に関する１つまたは複数の出力属性を生成することができる。たとえば、頂点シェーダ４６は、出力頂点に関する１つまたは複数の属性を生成するために、入力頂点の属性のうちの１つまたは複数に関して実質的な処理を実行することができる。一例として、頂点シェーダ４６は、出力頂点のための１つまたは複数の属性を生成するために、入力頂点の位置属性に対して世界変換、ビュー変換、投影変換、またはそれらの任意の組合せのうちの１つまたは複数を実行することができる。別の例として、頂点シェーダ４６は、

出力頂点に関する出力属性のセットを生成するために、入力属性のセットから追加および／または属性を削除することができる。

【０１２１】

[0121]ション段階６２（すなわち、ハルシェーダ４８、テッセレータ５０、およびドメインシェーダ５２）は、テッセレーションエンジンを形成することができ、入力パスデータによって定義されたパスセグメントを複数のラインセグメントにテッセレートすることができる。複数のラインセグメントは、レンダリングされることになるパスセグメントの曲率を近似することができる。一般に、ハルシェーダ４８は、さらなる処理のために、頂点シェーダ４６から受け取った制御点をドメインシェーダ５２に渡して、構成データをテッセレータ５０に提供することができる。テッセレータ５０は、特定のタイプのパスセグメントを表す１つもしくは複数のパラメータ式が評価されるべき値を決定することができる。ドメインシェーダ５２は、テッセレータ５０によって決定された値でパラメータ式を評価して、各評価に関する頂点を出力することができる。いくつかの例では、ドメインシェーダ５２によって出力された頂点の各々は、その頂点の位置を示す１つまたは複数の属性を含み得る。追加の例では、ドメインシェーダ５２によって出力された頂点の各々は、その頂点と関連付けられたパスレンダリングプリミティブのタイプを示す１つまたは複数の属性を含み得る。

【０１２２】

[0122]いくつかの例では、ハルシェーダ４８は、頂点シェーダ４６および／またはリソースブロック６４から受け取った制御点を処理することができ、ハルシェーダ４８によって実行されたハルシェーダプログラムの各インスタンスに関する出力制御点を生成することができる。たとえば、ハルシェーダ４８によって生成されることになる各出力制御点に関して、ハルシェーダ４８は、ＧＰＵ１２のシェーダユニット上でハルシェーダプログラムのインスタンスを実行することができる。いくつかの例では、ハルシェーダ４８は、各出力制御点に対して「パススルー」ハルシェーダプログラムを実行することができる。「パススルー」ハルシェーダプログラムは、ハルシェーダ４８に、出力制御点ごとに、入力制御点のそれぞれに対応する制御点を出力させることができる。この場合、出力制御点が入力制御点と同じ属性を有する場合、出力制御点は、入力制御点に対応し得る。

【０１２３】

[0123]さらなる例では、ハルシェーダ４８は、入力制御点のうちのそれぞれ１つの入力属性と同一でない出力制御点に関する１つまたは複数の出力属性を生成することができる。たとえば、ハルシェーダ４８は、出力制御点に関する１つまたは複数の属性を生成するために、入力制御点の属性のうちの１つまたは複数に関して実質的な処理を実行することができる。別の例として、ハルシェーダ４８は、出力制御点に関する出力属性のセットを生成するために、入力属性のセットから属性を追加および／または削除することができる。いくつかの例では、下でさらに詳細に説明するように、ＧＰＵ１２が終点パラメータ表示の形である、楕円弧に関するパスデータを受け取る場合、ハルシェーダ４８は、その楕円弧の終点パラメータ表示をその楕円弧の中心パラメータ表示に変換することができる。

【０１２４】

[0124]追加の例では、ハルシェーダ４８は、特定のレンダリング動作に関して、レンダリングされるべきではないプリミティブを破棄することができる。プリミティブを破棄することは、プリミティブに対応するデータをグラフィックスパイプライン４３のさらなる段階に渡させず、それによって、そのようなプリミティブをパイプラインの残りによって効果的にレンダリングさせないプロセスを指す場合がある。たとえば、グラフィックスパイプライン４３がフィル動作を実行しているとき、ハルシェーダ４８は、接合プリミティブとキャッププリミティブとを破棄することができる。別の例として、グラフィックスパイプライン４３がストローク動作を実行しているとき、ハルシェーダ４８は、オープンパスのためのクローズパスプリミティブを破棄することができる。クローズパスプリミティブは、ループを閉じるラインパスセグメントを表すプリミティブを指す場合がある。クローズパスプリミティブは、一般に、オープンパスではなくクローズパスであるパスのため

に使用される。いくつかの例では、クローズパスプリミティブは、パス中の他のラインパスセグメントを識別するために使用されるプリミティブタイプ識別子とは異なるプリミティブタイプ識別子によって識別され得る。たとえば、クローズパスプリミティブは、２．０ｆの代わりに２．１ｆのプリミティブタイプ識別子によって識別され得る。

【０１２５】

[0125]ハルシェーダ４８は、各パスセグメントに関するパッチ定数関数のインスタンスを実行することもできる。パッチ定数関数は、出力値を生成するとき、テッセレータ５０によって使用されることになる構成パラメータを決定して、テッセレータ５０に提供することができる。たとえば、パッチ定数関数は、ハルシェーダ４８にテッセレーション係数をテッセレータ５０に提供させることができる。テッセレーション係数は、テッセレータ５０が特定のテッセレーションドメインに適用されるテッセレーションの程度（たとえば、ドメインがどの程度微細に再分割されるべきか、および／またはドメインが再分割されるべきより小さなオブジェクトの数）を指定することができる。いくつかの例では、ハルシェーダ４８は、テッセレータ５０に、３次ベジェ曲線に対して４ｘのテッセレーションを実行することと、ラウンド接合および円形キャップに対して４ｘのテッセレーションを実行することと、ラインセグメントに対して１ｘテッセレーションを実行することとを行わせることができる。

【０１２６】

[0126]別の例として、パッチ定数関数は、ハルシェーダ４８に、テッセレーション中に使用されることになるテッセレーションドメインのタイプをテッセレータ５０に提供させることができる。テッセレーションドメインは、ドメインシェーダ５２によって使用されるための複数の座標を生成するために、テッセレータ５０によって使用されるオブジェクトを指す場合がある。概念的に、テッセレーションドメインは、テッセレータ５０によって複数のより小さなオブジェクトに再分割されるオブジェクトに対応し得る。より小さなオブジェクトの頂点の位置座標は、次いで、さらなる処理のために、ドメインシェーダ５２に送られる。いくつかの例では、テッセレーションドメインのタイプは、クワッド、トライ、および等値線のうちの１つになるように選択され得る。いくつかの例では、ドメインが再分割される先である、より小さいオブジェクトは、三角形、ラインセグメント、または点に対応し得る。いくつかの例では、ハルシェーダ４８は、等値線テッセレーションドメインタイプを指定して、テッセレータ５０が等値線ドメインをラインセグメントに再分割すべきであることを指定することができる。

【０１２７】

[0127]テッセレータ５０はまた、テッセレーション段階６２によって処理される各パスセグメントに関する複数の出力値も生成することができる。出力値は、特定のタイプのパスセグメントを表す１つまたは複数のパラメータ式がドメインシェーダ５２によって評価されるべき値を決定することができる。いくつかの例では、テッセレータ５０は、ハルシェーダ４８によってテッセレータ５０に提供された１つもしくは複数のテッセレーション係数および／またはテッセレーションドメインタイプに基づいて、複数の出力値を生成することができる。たとえば、テッセレータ５０は、等値線を複数のラインセグメントに再分割して、正規化された座標系内の複数のラインセグメントの各終点に関する出力値を生成することができる。

【０１２８】

[0128]ドメインシェーダ５２は、テッセレータ５０から出力値を受け取り、ハルシェーダ４８からパスセグメントに関する制御点を受け取り、パスセグメントの曲率および／または形状を近似する複数のテッセレートされたラインセグメントに対応する出力頂点を生成することができる。たとえば、テッセレータ５０から受け取られた出力値の各々に関して、ドメインシェーダ５２は、ＧＰＵ１２のシェーダユニット上でドメインシェーダプログラムのインスタンスを実行することができる。ドメインシェーダプログラムは、ドメインシェーダ５２に、テッセレータ５０から受け取った出力値の各々について、それぞれの出力値に対応する出力頂点に関する位置座標を生成するために、それぞれの出力値に基づ

いて決定された特定の値で、１つまたは複数のパラメータ式を評価させることができる。出力頂点座標を生成するために使用されるパラメータ式の係数のうちの１つまたは複数は、ハルシェーダ４８から受け取った制御点のうちの１つまたは複数に基づいて定義され得る。各出力頂点は、複数のテッセレートされたラインセグメントのうちの１つの終点に対応し得る。２つの連続する出力頂点は、単一のテッセレートされたラインセグメントの終点に対応し得る。

【０１２９】

[0129]追加の例では、ドメインシェーダプログラムは、ドメインシェーダ５２に、テッセレータ５０から受け取った出力値の各々に対応する出力頂点に関する正規座標を生成させることができる。たとえば、ドメインシェーダプログラムは、ドメインシェーダ５２に、テッセレータ５０から受け取った出力値の各々について、それぞれの出力値に対応する出力頂点に関する接線座標を生成するために、それぞれの出力値に基づいて決定された特定の値で、１つまたは複数の追加のパラメータ式を評価させることができる。出力頂点に関する接線座標は、出力頂点においてパスセグメントと交差するパスセグメントの接線の方向を示すことができる。ドメインシェーダ５２は、それぞれの出力頂点に対応する接線座標に基づいて出力頂点の各々に関する正規座標を生成することができる。特定の出力頂点のために生成された正規座標は、出力頂点においてパスセグメントと交差するパスセグメントの接線に対して直角である方向を示す法線ベクトルを示すことができる。

【０１３０】

[0130]いくつかの例では、グラフィックスパイプライン４３がフィル動作を実行しているとき、ドメインシェーダ５２は、そのようなロケーションのためのいかなる法線も生成することなしに、テッセレートされたラインセグメントの終点のロケーションに対応する頂点を生成することができる。そのような例では、グラフィックスパイプライン４３がストローク動作を実行しているとき、ドメインシェーダ５２は、いくつかの例では、テッセレートされたラインセグメントの終点のロケーションに対応する頂点を生成し、そのようなロケーションに対応する法線を生成することができる。

【０１３１】

[0131]ドメインシェーダ５２は、隣接する頂点の各セットがテッセレートされたラインセグメントを表す、順序付けられた順番で頂点を出力することができる。ラインセグメントは、頂点バッファ内で定義されたパスセグメントを集合的に近似することができる。たとえば、ドメインシェーダ５２は、以下のラインセグメント｛０，１｝、｛１，２｝、｛２，３｝、｛３，４｝、｛４，５｝を定義する頂点の以下のセット｛０，１，２，３，４，５｝を出力することができる。追加の例では、ドメインシェーダ５２は、前の例で列挙されたのと同じラインセグメントを定義し得る頂点の以下のセット｛０，１，１，２，２，３，３，４，４，５｝を出力することができる。

【０１３２】

[0132]いくつかの例では、テッセレータ５０およびドメインシェーダ５２は、以下の技法に従って、パスセグメントを複数のラインセグメントに均一にテッセレートするように構成され得る。具体的には、テッセレータ５０は、パラメータ評価のための座標を出力することができる（たとえば、ｔ＝０／Ｔ、１／Ｔ、２／Ｔ．．．Ｔ／Ｔ、式中、Ｔはテッセレーション係数である）。プリミティブのタイプに応じて、ドメインシェーダ５２は、テッセレータ５０によって出力された値で１つまたは複数のパラメータ式を評価することができる。

【０１３３】

[0133]ジオメトリシェーダ５４は、テッセレートされたラインセグメントをドメインシェーダ５２から受け取って、それらのテッセレートされたラインセグメントに基づいて、複数のプリミティブを生成することができる。このようにして、ジオメトリシェーダ５４は、ラインセグメントのためのセグメント順序を決定することができる。テッセレートされたラインセグメントの各々に関して、ジオメトリシェーダ５４は、ＧＰＵ１２のシェーダユニット上でジオメトリシェーダプログラムのインスタンスを実行して、それぞれのテ

ッセレートされたラインセグメントに基づいて、テッセレートされたラインセグメントに関する三角形プリミティブを生成することができる。いくつかの例では、テッセレートされたラインセグメントの各々に関して、ジオメトリシェーダ５４は、それぞれのテッセレートされたラインセグメントに対応する２つの頂点をドメインシェーダ５２から受け取って、三角形プリミティブに対応する３つの頂点のセットを生成することができる。

【０１３４】

[0134]いくつかの例では、三角形プリミティブの頂点のうちの２つは、２つの受け取った頂点と同じ頂点であり得る（たとえば、同じ位置座標を有し得る）。そのような例では、ジオメトリシェーダ５４は、レンダリングされることになるパスセグメントと関連付けられたすべてのテッセレートされたラインセグメントに共通である共通の頂点に基づいて、第３の頂点を生成することができる。共通の頂点は、テッセレートされたラインセグメントの終点のうちの１つに対応してよく、または対応しなくてもよい。いくつかの例では、共通の頂点は、レンダリングされることになるパスセグメントに関する、テッセレートされたラインセグメントに対応する頂点のセット内の第１の頂点に対応し得る。

【０１３５】

[0135]ジオメトリシェーダ５４は、ドメインシェーダ５２によって作り出された、テッセレートされたラインセグメントの各々に関して一回度起動され得る。テッセレートされたラインセグメントの各々に関して、ジオメトリシェーダ５４は、三角形の第１の頂点として共通の制御点を使用し、三角形の第２の頂点および第３の頂点として、それぞれのテッセレートされたラインセグメントの２つの終点を使用して、三角形プリミティブを生成することができる。たとえば、ドメインシェーダ５２が、以下のラインセグメント｛０，１｝、｛１，２｝、｛２，３｝、｛３，４｝、｛４，５｝を定義する、以下の頂点のセット｛０，１，２，３，４，５｝を生成した例が上で提供された。上記の一連のラインセグメントの場合、ジオメトリシェーダ５４は、以下の三角形、｛Ｃ，０，１｝、｛Ｃ，１，２｝、｛Ｃ，２，３｝、｛Ｃ，３，４｝、｛Ｃ，４，５｝、｛Ｃ，４，５｝を生成することができ、式中、Ｃは三角形のすべてに共通する任意の単一の頂点である。

【０１３６】

[0136]ラスタライザ５６は、複数の３Ｄグラフィックスプリミティブ（たとえば、点、線、および三角形）をそれらの３Ｄグラフィックスプリミティブに対応する複数の画素に変換するように構成され得る。たとえば、ラスタライザ５６は、三角形プリミティブに対応する３つの頂点を受け取って、それらの３つの頂点を、その三角形プリミティブによってカバーされたスクリーン画素位置に対応する複数の画素に変換することができる。三角形プリミティブによってカバーされたスクリーン画素位置は、三角形の頂点、三角形の縁、および三角形の内部に対応するスクリーン画素位置を含み得る。

【０１３７】

[0137]ピクセルシェーダ５８は、画素をラスタライザ５６から受け取って、ピクセルシェーダプログラムに従って、受け取った画素に基づいて、影付き画素を生成することができる。たとえば、ラスタライザ５６から受け取った各画素に関して、ピクセルシェーダ５８は、ＧＰＵ１２のシェーダユニット上でピクセルシェーダプログラムのインスタンスを実行することができる。いくつかの例では、ピクセルシェーダ５８は、各画素に対して「パススルー」ピクセルシェーダプログラムを実行することができる。「パススルー」ピクセルシェーダプログラムは、ピクセルシェーダ５８に、画素ごとに、入力画素のそれぞれに対応する画素を出力させることができる。この場合、出力画素が入力画素と同じ属性を有する場合、出力画素は、入力画素に対応し得る。

【０１３８】

[0138]さらなる例では、ピクセルシェーダ５８は、入力画素のうちのそれぞれの１つの入力画素の入力属性と同一でない、出力画素に関する１つまたは複数の出力属性を生成することができる。たとえば、ピクセルシェーダ５８は、出力画素に関する１つまたは複数の属性を生成するために、入力画素の属性のうちの１つまたは複数に関して実質的な処理を実行することができる。別の例として、ピクセルシェーダ５８は、出力画素に関する出

力属性のセットを生成するために、入力属性のセットから属性を追加および／または削除することができる。

【０１３９】

[0139]出力統合器６０は、ピクセルシェーダ５８から受け取った画素データをレンダターゲット（たとえば、フレームバッファまたはステンシルバッファ）内に配置することができる。いくつかの例では、出力マージャ６０は、ピクセルシェーダ５８から受け取った画素データをラスタ演算に基づいてレンダターゲット内にすでに記憶されている画素データと併合することができる。

【０１４０】

[0140]パスフィル動作を実行するために、ラスタライザ５６は、共通のステンシルバッファ（たとえば、リソースブロック６４に記憶されたバッファ）にジオメトリシェーダ５４によって受け取った三角形の各々をラスタライズすることができる。第１のパス中に、ピクセルシェーダ５８は、出力統合器６０に入力画素を直接パスするために、無効化されるか、または「パススルー」モードに設定され得る。出力統合器６０は、１つまたは複数のステンシルバッファフィル技法に従ってパスセグメントに関するフィル領域を示す値をステンシルバッファが記憶するようにステンシルバッファをポピュレートするように構成され得る。

【０１４１】

[0141]本開示の態様によれば、上記のように、ＧＰＵ１２は、以下のステップを伴うステンシルＴＩＲとバウンディングボックスとを使用する単一のパス手法を使用してフィル動作を実行することができる。

【０１４２】

１．複数のラインセグメントにパスセグメントをテッセレートする。

【０１４３】

２．ラインセグメントごとに三角形プリミティブを生成する。

【０１４４】

３．ステンシルバッファに三角形プリミティブのすべてをレンダリングする。

【０１４５】

４．ステンシル中にバウンディングボックスを決定する。

【０１４６】

５．ステンシルＴＩＲを用いてバウンディングボックスをレンダリングする。

【０１４７】

上記の例では、ＧＰＵ１２は、（本明細書ではテッセレーションエンジンと呼ばれる場合もある）テッセレーション段階６２に、バウンディングボックスパラメータ（たとえば、ｂｂ＿ｂｏｘ）をリセットすべきであることを示すイベント（たとえば、ｂｂ＿ｓｔａｒｔ）を送ることができる。ＧＰＵ１２は、次いで、上記で説明したプロセスを使用してステンシルバッファを更新しながら、三角形プリミティブを生成することができる。さらに、テッセレーション段階６２は、最小～最大パラメータを頂点データと比較することによってバウンディングボックスパラメータ（ｂｂ＿ｂｏｘ）を更新する。すなわち、テッセレーション段階６２は、たとえば、デカルト座標を使用して前に決定された頂点のさらに上、下、右側、左側に位置するロケーションを頂点が有するかどうかを決定するために頂点の各々を検査することができる。頂点が他の頂点に対して最外ロケーションに位置する場合、テッセレーション段階６２は、バウンディングボックスパラメータ（ｂｂ＿ｂｏｘ）を更新することができる。

【０１４８】

[0142]テッセレーション段階６２がバウンディングボックス終了イベント（たとえば、ｂｂ＿ｅｎｄ）を受け取ると、テッセレーション段階は、たとえば、パスの三角形プリミティブを包含するバウンディングボックスを形成する決定されたバウンディングボックス座標に対応するｒｅｃｔｌｉｓｔを生成することができる。ラスタライザ５６は、次いで、バウンディングボックスをラスタライズすることができる。本開示の態様によれば、ラ

スタライザ５６は、レンダターゲットに対してステンシルされた画素をスーパーサンプリングするステンシルＴＩＲを実行することができ、ピクセルシェーダ５８は、ステンシルされた画素のみをシェーディングする。ピクセルシェーダ５８が画素をシェーディングするとき、画素のステンシル値がステンシルバッファから消去され得る。

【０１４９】

[0143]したがって、上記で説明した例では、テッセレーション段階６２は、バウンディングボックス開始イベント（ｂｂ＿ｓｔａｒｔ）とバウンディングボックス終了イベント（ｂｂ＿ｅｎｄ）との間にバウンディングボックスパラメータ（ｂｂ＿ｂｏｘ）の累積を維持する。（たとえば、ジオメトリシェーダ５４、ラスタライザ５６、ピクセルシェーダ５８および／または出力統合器６０を含む）レンダバックエンドは、バウンディングボックス開始イベント（ｂｂ＿ｓｔａｒｔ）とバウンディングボックス終了イベント（ｂｂ＿ｅｎｄ）との間に固定動作を予想する。すなわち、レンダバックエンドは、（ＧＰＵドライバ２８（図２）などの）ドライバがレンダバックエンドレジスタをプログラムすることなしにバウンディングボックスを決定することに関連する動作を実行することができ、これはリソースブロック６４中に割り当てられ得る。テッセレーション段階６２に関して説明したが、上記の技法がグラフィックスパイプラインの１つまたは複数の他の段階によって実行され得ることを理解されたい。このようにして、ＧＰＵ１２は、別個のパス中にバウンディングボックスをレンダリングする必要なしに単一のパスでパスをフィルするためにグラフィックスパイプライン４３を使用することができる。

【０１５０】

[0144]本開示の他の態様によれば、グラフィックスパイプライン４３は、ストロークされたパスセグメントに関するダッシングを実行するように構成され得る。説明のための一例では、ジオメトリシェーダ５４は、ドメインシェーダ５２からテッセレートされたラインセグメントを受け取り、テッセレートされたラインセグメントに基づいて複数の三角形プリミティブを生成することができる。複数のプリミティブは、シェーディングされることになるダッシュセグメントを含み得、複数のプリミティブは、特定の順序、たとえば、セグメント順序であり得る。ジオメトリシェーダ５４（またはグラフィックスパイプライン４３の別の構成要素）はまた、ダッシュの各々の長さを決定することができる。

【０１５１】

[0145]さらに、ジオメトリシェーダ５４は,各ダッシュが生成されるとダッシュの長さを累積し、前のダッシュセグメント、たとえば、セグメント順序で現在のセグメントに先行するダッシュセグメントの長さのプレフィックス総和を各ダッシュセグメントに割り当てる。たとえば、第１のダッシュセグメントには、０のプレフィックス総和が割り当てられ得、第２のダッシュセグメントには、長さ優先ダッシュセグメントのプレフィックス総和が割り当てられ得、第３のダッシュセグメントには、第１のダッシュセグメントと第２のダッシュセグメントとの組合せの長さのプレフィックス総和が割り当てられ得、以下同様である。

【０１５２】

[0146]ラスタライザ５６は、一般に、ダッシュセグメントを受け取り、ラスタライゼーション中にプリミティブ順序に従い、ここで、プリミティブ順序は、レンダリングの順序を指す。ラスタライゼーションの後に、各ダッシュセムグメントのためのプレフィックス総和が、ダッシュセグメントをシェーディングするときに使用するためにピクセルシェーダ５８に送られ得る。たとえば、適切なロケーションにあるダッシュセグメントをシェーディングするために、ピクセルシェーダ５８は、テクスチャオフセットとして各ダッシュセグメントのためのプレフィックス総和を適用することができる。テクスチャオフセットは、前のダッシュセグメントのロケーションを示し、それによって、ピクセルシェーダ５８が、前のセグメントに対して適切なロケーションにある次のダッシュセグメントをシェーディングすることが可能になる。

【０１５３】

[0147]図４は、レンダリングされることになる例示的なパス８０の図である。たとえば

、パス８０は、上部が丸く、下部が細長い「アイスクリームコーン」形状を表す。パス８０は、２つの３次方程式からなるクローズパスであり得る。セグメントは、ｐａｔｃｈ４プリム（プリミティブ）にパックされ得る。たとえば、パス８０に関する入力パスデータは、以下の形または類似の形をとることができる。

【数７】

```
{ XMFLOAT4(0.0f, 0.0f,     CUBIC, 0.0f) },
{ XMFLOAT4(0.4f, 1.2f,     0.0f,  0.0f) },
{ XMFLOAT4(1.4f, 1.2f,     0.0f,  0.0f) },
{ XMFLOAT4(1.8f, 0.0f,     0.0f,  0.0f) },
```

【数８】

```
{ XMFLOAT4(1.8f, 0.0f,     CUBIC, 0.0f) },
{ XMFLOAT4(0.5f,-3.0f,     0.0f,  0.0f) },
{ XMFLOAT4(1.3f,-3.0f,     0.0f,  0.0f) },
{ XMFLOAT4(0.0f, 0.0f,     0.0f,  0.0f)
```

【０１５４】

この例では、各行は、頂点または制御点を表し、括弧内の各パラメータは、それぞれの頂点すなわち制御点の属性を表す。この例では、第１の制御点の最後の属性は、レンダリングされることになるパスセグメントのタイプを示すデータ（すなわち、「パスセグメントタイプインジケータ」）を記憶する。具体的には、この例では、パスセグメントタイプインジケータは、パスセグメントが３次ベジェパスセグメントであることを意味する０．０ｆである。パスセグメントに関する他の属性は、いくつかの例では、ラインパスセグメントに関して上記で説明した属性と同様の属性を含み得る。

【０１５５】

[0148]図５Ａ～図５Ｃは、図４に示されるパス８０のための例示的なフィル動作を示す一連の図である。パス８０は、例示のために、図５Ａ～図５Ｃの例においてテッセレートされている（たとえば、通常より少ないセグメントを有する）。さらに、説明のためにＧＰＵ１２に関して説明したが、図５Ａ～図５Ｃにおいて実行されるプロセスは、様々な他のプロセッサによって実行され得る。

【０１５６】

[0149]図５Ａに示されるように、ＧＰＵ１２は、ラインストリップ方式８４で接続されたいくつかの頂点８２を含めるためにパス８０をテッセレートする。図５Ｂに示されるように、ＧＰＵ１２は、いくつかの三角形プリミティブを形成するためにピボット点８８に接続されたいくつかのラインセグメント８６を生成する。図５Ｂの例では、パス８０の相対的な第１の頂点は、ピボット点８８として使用される。三角形の巻上げ順序が適切なステンシル動作を決定する。たとえば、あらゆる生成されたラインセグメントがピボット点８８に接続される。三角形の得られた配向（たとえば、時計回りまたは反時計回り）は、三角形プリミティブの巻上げ順序を決定することができる。巻上げ順序は、様々な方法でステンシル値に影響を及ぼすことができる（たとえば、時計回りの巻上げ順序の場合にステンシル値を増分するか、または反時計回りの巻上げ順序の場合にステンシル値を減分する）。

【０１５７】

[0150]この例では、ＧＰＵ１２は、ステンシル中に図５Ｂに示される三角形プリミティブをシェーディングしない。むしろ、上記のように、ステンシル中にレンダリングされる三角形プリミティブは、ステンシルテクスチャ９０にのみ影響を及ぼす。すなわち、ステンシルテクスチャ９０は、画像中に現れる、たとえば、レンダリングされ、シェーディングされるパスの部分を示す。

【０１５８】

[0151]図５Ｃに示されるように、ＧＰＵ１２は、ステンシルテクスチャ９０を包含するバウンディングボックス９２を決定する。すなわち、バウンディングボックスは、フィルされることになるパスの全体をカバーする。ＧＰＵ１２は、次いで、フィルされたパス９６を生成するためにバウンディングボックス９２に対してステンシルＴＩＲを実行する。このようにして、ＧＰＵ１２は、バウンディングボックス９２を決定し、単一のレンダリングパスでパス８０をフィルする。

【０１５９】

[0152]図６は、ステンシル動作を示す概念図である。たとえば、説明のために、ＧＰＵ１２が１６ｘのＭＳＡＡを使用してプリミティブ１００をレンダリングすると仮定する。この例では、各方形は、画素１０２のサンプルを表す。

【０１６０】

[0153]本開示の態様によれば、ＧＰＵ１２は、ステンシルＴＩＲを実行することができる。したがって、ＧＰＵ１２は、レンダターゲットパラメータ（たとえば、レンダリングされた画素のためのメモリ割当て）から独立してステンシルパラメータ（たとえば、ステンシルサンプリングレート）を決定することができる。この例では、ＧＰＵ１２は、画素がサンプルごとのステンシルテストをパスしたかどうかに基づいて画素をレンダリングするためのカバレージ値を決定することができる。

【０１６１】

[0154]いくつかの例では、ＧＰＵ１２は、サンプルが非ゼロ値を有するかどうかを決定するためにステンシルテストを実行することができる。たとえば、ＧＰＵ１２は、非ゼロのステンシル値を有するサンプルがレンダリングされるゼロ／非ゼロステンシルテストを実行することができる。別の例では、ＧＰＵ１２は、奇数の（または偶数の）値を有するサンプルがレンダリングされる奇数／偶数ステンシルテストを実行することができる。したがって、いくつかの例では、ＧＰＵ１２は、サンプルが奇数値を有するかどうかを決定するためにステンシルテストを実行することができる。さらに他の例では、ＧＰＵ１２は、サンプルが偶数値を有するかどうかを決定するためにステンシルテストを実行することができる。

【０１６２】

[0155]いずれの場合も、図６に示される例では、画素１０２の（この場合も、ボックスによって表される）１６個サンプルのうちの１０個がプリミティブ１００内に位置する。したがって、プリミティブ１００のためのカバレージマスクは画素１０２を含み得、ＧＰＵ１２は、レンダリング中に画素１０２をシェーディングすることができる。

【０１６３】

[0156]図７は、本開示の態様による、例示的なフィル動作を示す概念図である。たとえば、図７に、三角形プリミティブを決定し、プリミティブ１１０の配向に基づいてステンシルバッファを更新することと、プリミティブの最外点と、ステンシルバッファ１１４のコンテンツと、描画されたバウンディングボックスおよびステンシルされた画素１１６とに基づいてバウンディングボックス１１２を決定することとを示す。

【０１６４】

[0157]いくつかの例によれば、ステンシル中に含まれるプリミティブをステンシルし、レンダリングする間にバウンディングボックスを決定するシーケンスは、以下のＡＰＩ呼出しを使用してＧＰＵ１２によって実行され得る。

【０１６５】

Ｄｒａｗ＿ＢＢ（）　／／描画呼出し内のプリミティブのバウンディングボックスを計

算する
　／／または、ＢｅｇｉｎＱｕｅｒｙ（）．．．ＥｎｄＱｕｅｒｙ（）であり得る
　／／ピクセルシェーダの境界が画定されない場合、境界ボックスだけが
　／／計算され、プリミティブはレンダリングされない
　Ｒｅｎｄｅｒ＿ＢＢ（）　／／以前に計算されたバウンディングボックスを
　／／レンダリングするか、またはプリミティブが送られずに、
　／／ピクセルシェーダおよび他のバックエンド状態が指定されることを除いて、描画呼出しと同じ
　／／ＤｒａｗＩｎｄｉｒｅｃｔのような方法であり得る
ここで、Ｄｒａｗ＿ＢＢは、ＧＰＵ１２に、（画素をレンダリングすることなしに）ステンシル中にバウンディングボックス１１２を決定するように命令し、Ｒｅｎｄｅｒ＿ＢＢは、ＧＰＵ１２に、バウンディングボックスの上でステンシルＴＩＲを実行するように命令する。このようにして、ＧＰＵ１２は、バウンディングボックスを決定し、単一のレンダリングパスでパスのフィルを実行することができる。

【０１６６】

[0158]図８は、本開示の態様による、レンダリング中のメモリ帯域幅を示すグラフである。たとえば、図８に、３つの異なるアンチエイリアシングレート（４ｘ、８ｘ、１６ｘ）、ならびにテッセレーションおよびジオメトリシェーディングが実行されるＭＳＡＡ方式１２０と、ステンシルＴＩＲ方式１２２と、保守的なラスタライゼーション方式１２４（たとえば、ループブリンプロセス）との３つのレンダリング方式の各々のためのレートに関連付けられた関連するデータ転送を示す。図８に示される帯域幅要件は、６０フレーム毎秒（ｆｐｓ）および３２ビットの色と、２４ビットの深度と、８ビットのステンシルとを有するバッファフォーマットでレンダリングされる画像のテストシーケンスに関連付けられる。

【０１６７】

[0159]図８のグラフに示されるように、ＭＳＡＡ１２０のためのメモリ帯域幅要件は、ステンシルＴＩＲ１２２および保守的なラスタライゼーション１２４のためのメモリ帯域幅要件よりもかなり高い。以下に示される表１に、ＭＳＡＡ１２０と、保守的なラスタライゼーション１２４と、ステンシルＴＩＲ１２２との比較を示す。

【表１】

表1

	MSAA	保守的なラスタライゼーション	ステンシルTIR
品質	同様	わずかにより良好	同様
メモリ帯域幅	高い	低い（サンプルごとに1つの画素）	低い（ステンシル＋サンプルごとに1つの画素）
画素シェーディング	なし	極めて高い	なし
HWの変更	すでに存在	小さい	中程度
APIの変更	すでに存在	小さい	中程度
ジオメトリ作業	CPU（テッセレーションエンジンに移動）	CPU（複雑なアルゴリズム）	より少ないCPU（テッセレーションエンジンに移動）
一般的なパスレンダリング	はい	いいえ	はい
重複パス	はい	いいえ	はい

【０１６８】

[0160]以下に示される表２に、ＭＳＡＡ１２０と、ステンシルＴＩＲ１２２と、保守的なラスタライゼーション１２４との間の追加の比較を示す。

【表２】

表2

	Nx MSAA:	NxTIR＋ステンシル:	保守的なラスタライゼーション
サンプルテスト（HW）	Nx	Nx	1x
深度バッファ更新	Nx	なし	なし
ステンシルバッファ更新	Nx	Nx	なし
色バッファ更新	Nx	1x	1x
ピクセルシェーダ起動	1x	1x	1x（比較的複雑な画素シェーディング段階）

【０１６９】

[0161]図９Ａ～図９Ｄは、図４に示されるパスのための例示的なダッシング動作を示す一連の図である。やはり、説明のために、図９Ａ～図９Ｄの例では、パス８０はテッセレートされている。さらに、説明のためにＧＰＵ１２に関して説明したが、図９Ａ～図９Ｄにおいて実行されるプロセスは、様々な他のプロセッサによって実行され得る。

【０１７０】

[0162]図９Ａに示されるように、ＧＰＵ１２は、ラインストリップ８４中で接続されたいくつかの頂点８２を含めるためにパス８０をテッセレートする。さらに、ＧＰＵ１２は、（頂点から延びる矢印として示されている）法線１３０の数を決定する。図９Ａの例では、二重の法線１３０は、接合ロケーションを示す。接合を作成するには、ラインストリップ中の次のプリミティブの終点接線を必要とし得る。図９Ａはまた、例示的なセグメント１３２を含む。

【０１７１】

[0163]図９Ｂに、セグメント１３２がストローク幅／２だけ＋／－法線方向に膨張される膨張動作をＧＰＵ１２がセグメント１３２に対して実行することを示す。説明のために図９Ｂの例に追加の太いストロークが示される。図９Ｂに、膨張されたセグメント１３４と、ストロークされ膨張されたセグメント１３６とを示す。

【０１７２】

[0164]図９Ｃに、ダッシングし、ストロークされ、膨張されたセグメント１３６を示す。たとえば、ＧＰＵ１２は、第１のダッシュセグメント１３８と、第２のダッシュセグメント１４０と、第３のダッシュセグメント１４２とを決定することができる。この例では、第１のダッシュセグメント１３８と第３のダッシュセグメント１４２とは、可視セグメントであり、一方、第２のダッシュセグメント１４０は不可視ダッシュセグメントである。ダッシングのために、ＧＰＵ１２は、各ダッシュセグメント（線）１３８、１４０、および１４２のための開始ロケーションを決定する。上記のように、いくつかの例では、プレフィックス総和ユニット４２は、ジオメトリシェーディング中にダッシュセグメント１３８～１４２の長さを累積することができる。

【０１７３】

[0165]ＧＰＵ１２は、テクスチャ座標のためのテクスチャオフセットとして０から線長さＬまでの長さを適用することができる。たとえば、本開示の態様によれば、プレフィックス総和ユニット４２は、セグメント１３８～１４２の各々のロケーションを示すｌｉｎｅｌｅｎｇｔｈ値を計算することができる。プレフィックス総和装置４２は、ピクセルシェーダ段階にプレフィックス総和したｌｉｎｅｌｅｎｇｔｈ値を送ることができ、これは、画素シェーディング中にセグメント１３８～１４２のそれぞれのロケーションを決定する。ＧＰＵ１２は、可視ダッシュパターンの一部を形成するので可視セグメント１３８および１４２を（シェーディングされたフラグメントとして）保持し、ダッシュパターン中で不可視であるのでセグメント１４０を（シェーディングすることなしに）破棄する。

【０１７４】

[0166]いくつかの例では、ｐｏｉｎｔｓｉｚｅなどのプリミティブごとのスカラー値を決定することをサポートするグラフィックスＡＰＩは、スカラー長さを決定するようにＧＰＵ１２に命令するために使用され得る。本開示の態様によれば、グラフィックスＡＰＩは、ｌｉｎｅｌｅｎｇｔｈ値をサポートすることができる。このｌｉｎｅｌｅｎｇｔｈ値は、プリミティブの同じフラット属性であり得るが、属性は、ピクセルシェーダ段階に与えられ得る。たとえば、ＧＰＵ１２は、画素シェーディング中にオフセット座標を決定するためにテッセレートされたプリミティブごとにプレフィックス総和パスを適用することができる。さらに、ＡＰＩ呼出し（ｑｕｅｒｙ＿ｓｔａｒｔ／ｅｎｄと同様の）ｐｒｓｕｍ＿ｓｔａｒｔ、ｐｒｓｕｍ＿ｅｎｄは、破線の相対的な開始および終了を示す１つまたは複数の描画呼出しをブラケット化することができる。

【０１７５】

[0167]図９Ｄに、フィルされ、ダッシングされたパス１４６を生成するために、ダッシングされたストローク１４２のフィルされたパス１４４への追加を示す。

【０１７６】

[0168]図１０は、本開示による、フィル動作を実行するための例示的な技法を示す流れ図である。説明のためにＧＰＵ１２によって実行されるものとして説明したが、図１０に示される技法が、様々な他のプロセッサによって実行され得ることを理解されたい。さらに、本技法を実行するために、図示したステップよりも少ないステップ、それに追加のス

テップ、またはそれとは異なるステップが使用され得る。

【０１７７】

[0169]図１０の例では、ＧＰＵ１２は、パスデータを受け取る（１６０）。パスデータは、レンダリングされることになるパスの１つまたは複数のパスセグメントを示し得る。ＧＰＵ１２はまた、ステンシルパラメータを決定する（１６２）。いくつかの例では、ステンシルパラメータは、パスの各アンチエイリアス画素のカバレージ値を決定するためのサンプリングレートを示し得る。ＧＰＵ１２はまた、ステンシルパラメータとは別々に、レンダターゲットパラメータを決定する（１６４）。レンダターゲットパラメータは、パスの各アンチエイリアス画素のためのメモリ割当てを示し得る。

【０１７８】

[0170]ＧＰＵ１２は、パスデータによって定義されたパスセグメントを複数のラインセグメントにテッセレートする（１６６）。たとえば、ＧＰＵ１２は、図５Ａに示されるラインストリップなどのラインストリップにパスデータをテッセレートすることができる。ＧＰＵ１２は、次いで、複数のラインセグメントに基づいて複数の三角形プリミティブを生成する（１６８）。複数の三角形プリミティブの各々は、複数のラインセグメントのそれぞれに基づいて生成され得る。所与のパスセグメントに関する複数の三角形プリミティブの各々は、共通の頂点を共有することができる。三角形プリミティブの各々のための他の２つの頂点は、複数のラインセグメントのそれぞれの終点に対応し得る。

【０１７９】

[0171]ＧＰＵ１２は、ステンシルパラメータを使用して共通のステンシルバッファに複数の三角形プリミティブの各々をレンダリングし、バウンディングボックスを決定する（１７０）。たとえば、上記のように、ＧＰＵ１２は、ステンシル中に三角形プリミティブをシェーディングしない。しかしながら、ＧＰＵ１２は、プリミティブのためのバウンディングボックスを決定するために三角形プリミティブの最外点を決定することができる。いくつかの例では、ＧＰＵ１２は、各三角形プリミティブに関する座標を決定し、プリミティブが前の三角形プリミティブの最外点を越えて拡大するたびに、上部境界点、下部境界点、右境界点、および／または左境界点を上書きすることができる。

【０１８０】

[0172]ステンシルバッファに三角形プリミティブのすべてをレンダリングした後に、ステンシルバッファは、どのピクセルがパスセグメントに関するフィル領域の内側にあるかを示すデータを記憶することができる。さらに、バウンディングボックスは、三角形プリミティブの各々を包含する。

【０１８１】

[0173]ＧＰＵ１２は、次いで、レンダターゲットパラメータとステンシルバッファとを使用してバウンディングボックスをラスタライズする（１７２）。たとえば、本開示の態様によれば、ＧＰＵ１２は、パスデータの各画素の色値を決定するために、バウンディングボックスに対してステンシルＴＩＲを実行する。ステンシルバッファ中のデータにより、フィル領域内の画素がフィル色を用いてシェーディングされ、フィル領域の外部にある画素がシェーディングなしのままにされるようになり得る。バウンディングボックスのレンダリングが完了すると、レンダターゲット（たとえば、フレームバッファ）は、レンダターゲットパラメータを使用してパスセグメントに関するフィル領域の、ラスタライズされたバージョンを記憶することができる。

【０１８２】

[0174]図１１は、本開示による、ストローク動作を実行するための例示的な技法を示す流れ図である。この場合も、説明のためにＧＰＵ１２によって実行されるものとして説明したが、図１１に示される技法が、様々な他のプロセッサによって実行され得ることを理解されたい。さらに、本技法を実行するために、図示されるステップよりも少ないステップ、それに追加のステップ、またはそれとは異なるステップが使用され得る。

【０１８３】

[0175]ＧＰＵ１２は、パスデータを受け取る（１８０）。パスデータは、レンダリング

されることになるパスの１つまたは複数のパスセグメントを示し得る。ＧＰＵ１２は、パスデータによって定義されたパスセグメントを複数のラインセグメントにテッセレートする（１８２）。たとえば、ＧＰＵ１２は、図９Ａに示されるラインストリップなどのラインストリップにパスデータをテッセレートすることができる。

【０１８４】

[0176]ＧＰＵ１２は、パスセグメントに関するストローク領域に空間的に対応する複数のプリミティブを生成する（１８４）。たとえば、複数のテッセレートされたラインセグメントの各々について、ＧＰＵ１２は、それぞれのラインセグメントのためのストローク領域に空間的に対応する１つまたは複数のプリミティブを生成することができる。ＧＰＵ１２は、ラインセグメントのジオメトリシェーディング中にラインセグメントごとのテッセレートされたプリミティブの数を決定することができる。すなわち、ジオメトリシェーディング中に、ＧＰＵ１２は、（たとえば、ストロークの特定のセグメントをシェーディングすることなしに）「ダッシングなしの」ストロークを生成することができる。

【０１８５】

[0177]ダッシュするとき、ＧＰＵ１２は、テッセレートされたプリミティブごとのパス長を決定する（１８６）。たとえば、ＧＰＵ１２は、ジオメトリシェーディング中に生成された各ダッシュセグメント（プリミティブ）のための長さの累積を決定することができる。すなわち、ダッシュセグメントは、特定の順序（たとえば、テッセレーションおよび／またはジオメトリシェーディング中に決定された順序）で順序付けられ得る。プリミティブごとに、ＧＰＵ１２は、順序でそれに先行するプリミティブの長さを累積することができる。

【０１８６】

[0178]ＧＰＵ１２は、長さの累積に基づいてレンダリングされている各プリミティブのテクスチャ座標のためのテクスチャオフセットを決定することができる（１８８）。たとえば、上記のように、ＧＰＵ１２は、プリミティブの各々の始端のテクスチャ座標を決定するために長さ情報を使用することができる。ＧＰＵ１２は、画素シェーディング中にテクスチャオフセットを適用することができる（１９０）。たとえば、ＧＰＵ１２は、テクスチャオフセットを適用し、ストロークされたパスデータのための適切な色を使用してダッシュのセグメントの各々をシェーディングする。

【０１８７】

[0179]いくつかの例では、本開示の技法により、Ｄｉｒｅｃｔ　Ｘ１１ハードウェアのユーザは、Ｄｉｒｅｃｔ　Ｘ１１ハードウェアを使用するか、または同様のパフォーマンス特性を有するハードウェアを用いてパスレンダリングを実行することが可能になり得る。さらなる例では、本開示の技法は、パスレンダリングに全ＧＰＵのレンダリングソリューションを与えることができる。

【０１８８】

[0180]本開示の技法について、主に、ＤＸ　１１グラフィックスＡＰＩによって定義されたハードウェアアーキテクチャに関して説明してきたが、本開示の技法はまた、たとえば、ＯｐｅｎＧＬグラフィックスＡＰＩ（たとえば、ＯｐｅｎＧＬバージョン４．０、４．１、４．２、４．３および以降のバージョン）など、他のオンチップのテッセレーション対応グラフィックスＡＰＩに従って定義されたハードウェアアーキテクチャで実行され得る。本開示の技法が、ＯｐｅｎＧＬグラフィックスＡＰＩに従って定義されたハードウェアアーキテクチャで実装される例では、本開示におけるハルシェーダ４８に帰属する機能のうちの１つまたは複数は、テッセレーション制御シェーダによって実行され得、および／または本開示におけるドメインシェーダ５２に帰属する機能のうちの１つまたは複数は、テッセレーション評価シェーダによって実行され得る。

【０１８９】

[0181]本開示で説明する技法は、少なくとも部分的に、ハードウェア、ソフトウェア、ファームウェア、またはそれらの任意の組合せで実装され得る。たとえば、説明した技法の様々な態様は、１つもしくは複数のマイクロプロセッサ、デジタル信号プロセッサ（Ｄ

ＳＰ）、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、あるいは任意の他の等価な集積回路またはディスクリート論理回路を含む、１つもしくは複数のプロセッサ内、ならびにそのような構成要素の任意の組合せ内で実装され得る。「プロセッサ」または「処理回路」という用語は、一般に、単独で、あるいは他の論理回路または、処理を実行する個別ハードウェアなどの他の等価回路との組合せで上記の論理回路のいずれかを指すことがある。

【０１９０】

[0182]そのようなハードウェア、ソフトウェア、およびファームウェアは、本開示で説明する様々な動作および機能をサポートするために、同じデバイス内で、または別のデバイス内で実装され得る。さらに、説明したユニット、モジュール、または構成要素のいずれも、個別であるが相互運用可能な論理デバイスとして、一緒に、または別々に実装され得る。モジュールまたはユニットとしての様々な機能の図は、様々な機能的態様を強調するものであり、そのようなモジュールまたはユニットが別々のハードウェアまたはソフトウェア構成要素によって実現されなければならないことを必ずしも暗示するとは限らない。そうではなく、１つもしくは複数のモジュールまたはユニットに関連する機能は、別々のハードウェア構成要素、ファームウェア構成要素、および／またはソフトウェア構成要素によって実行されるか、あるいは共通もしくは別々のハードウェア構成要素内またはソフトウェア構成要素内に組み込まれることがある。

【０１９１】

[0183]また、本開示で説明する技法は、命令を記憶するコンピュータ可読記憶媒体などのコンピュータ可読媒体中に記憶、実施または符号化され得る。コンピュータ可読媒体中に埋め込まれるか、または符号化される命令は、たとえば、それらの命令が１つまたは複数のプロセッサによって実行されたとき、１つまたは複数のプロセッサに本明細書で説明する技法を実行させ得る。コンピュータ可読記憶媒体は、ランダムアクセスメモリ（ＲＡＭ）、読取り専用メモリ（ＲＯＭ）、プログラマブル読取り専用メモリ（ＰＲＯＭ）、消去可能プログラマブル読取り専用メモリ（ＥＰＲＯＭ）、電気的消去可能プログラマブル読取り専用メモリ（ＥＥＰＲＯＭ）、フラッシュメモリ、ハードディスク、ＣＤ－ＲＯＭ、フロッピー（登録商標）ディスク、カセット、磁気媒体、光学媒体、または他の有形のコンピュータ可読記憶媒体を含み得る。

【０１９２】

[0184]コンピュータ可読媒体は、上記に記載した有形記憶媒体などの有形記憶媒体に対応するコンピュータ可読記憶媒体を含み得る。コンピュータ可読媒体はまた、たとえば、通信プロトコルに従って、ある場所から別の場所へのコンピュータプログラムの転送を可能にする任意の媒体を含む通信媒体を備え得る。このようにして、「コンピュータ可読媒体」という句は、概して、（１）非一時的である有形コンピュータ可読記憶媒体、および（２）一時的な信号または搬送波などの非有形コンピュータ可読通信媒体に対応し得る。

【０１９３】

[0185]様々な態様および例について説明した。しかしながら、以下の特許請求の範囲から逸脱することなく本開示の構造または技法に変更を行うことができる。

以下に、本願出願の当初の特許請求の範囲に記載された発明を付記する。

［Ｃ１］

グラフィックスデータをレンダリングする方法であって、

グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、前記複数の順序付きセグメントの前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前のセグメントの長さの累積に基づく、

前記現在のセグメントのロケーションを決定するために、前記テクスチャオフセットを適用することを含めて前記現在のセグメントをピクセルシェーディングすることと

を備える方法。

［Ｃ２］

前記複数のセグメントが、１つまたは複数の可視セグメントと１つまたは複数の不可視セグメントとを含み、前記方法が、
前記現在のセグメントの前記決定されたロケーションに基づいて、前記現在のセグメントが可視セグメントであるかどうかを決定することと、
前記決定に基づいて前記現在のセグメントを保持するか、または破棄することと
をさらに備える、Ｃ１に記載の方法。
［Ｃ３］
前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線をジオメトリシェーディングすることと、
長さの前記累積に基づいて前記テクスチャオフセットを決定することが、前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように前記長さ値を決定することと
をさらに備える、Ｃ１に記載の方法。
［Ｃ４］
前記長さ値を決定することが、前記順序が前のセグメントの長さを指定するｌｉｎｅｌｅｎｇｔｈスカラー値を生成することを備える、Ｃ３に記載の方法。
［Ｃ５］
前記複数の順序付きセグメントの前記セグメントの各々をラスタライズすること、
ここにおいて、前記テクスチャオフセットを適用することが、前記現在のセグメントがラスタライズされた後に前記ラスタライズされた現在のセグメントに前記テクスチャオフセットを適用することを備える、
をさらに備える、Ｃ１に記載の方法。
［Ｃ６］
前記テクスチャオフセットを適用することが、前記ロケーションを示す、前記現在のセグメントのテクスチャ座標値を決定することを備える、Ｃ１に記載の方法。
［Ｃ７］
前記画素シェーディングが、前記破線をストロークすることを含む、前記破線のためのパスレンダリングプロセス中に含まれる、Ｃ１に記載の方法。
［Ｃ８］
前記セグメントの前記順序がプリミティブ順序であるように、前記セグメントのジオメトリシェーディング中に前記セグメントの前記順序を決定することをさらに備える、Ｃ１に記載の方法。
［Ｃ９］
グラフィックスデータをレンダリングするための装置であって、
破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、前記複数の順序付きセグメントの前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前のセグメントの長さの累積に基づく、
前記現在のセグメントのロケーションを決定するために、前記テクスチャオフセットを適用することを含めて前記現在のセグメントをピクセルシェーディングすることと
を行うように構成されたグラフィックス処理ユニット（ＧＰＵ）を備える装置。
［Ｃ１０］
前記複数のセグメントが、１つまたは複数の可視セグメントと１つまたは複数の不可視セグメントとを含み、前記ＧＰＵが、
前記現在のセグメントの前記決定されたロケーションに基づいて、前記現在のセグメントが可視セグメントであるかどうかを決定することと、
前記決定に基づいて前記現在のセグメントを保持するか、または破棄することと
を行うようにさらに構成された、Ｃ９に記載の装置。
［Ｃ１１］
前記ＧＰＵが、

前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線をジオメトリシェーディングすることと、
長さの前記累積に基づいて前記テクスチャオフセットを決定することが、前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように前記長さ値を決定することと
を行うようにさらに構成された、Ｃ９に記載の装置。
［Ｃ１２］
前記長さ値を決定するために、前記ＧＰＵが、前記順序が前のセグメントの長さを指定するｌｉｎｅｌｅｎｇｔｈスカラー値を生成することを行うように構成された、Ｃ１１に記載の装置。
［Ｃ１３］
前記ＧＰＵが、
前記複数の順序付きセグメントの前記セグメントの各々をラスタライズすること、
ここにおいて、前記テクスチャオフセットを適用するために、前記ＧＰＵが、前記現在のセグメントがラスタライズされた後に前記ラスタライズされた現在のセグメントに前記テクスチャオフセットを適用することを行うように構成された、
を行うようにさらに構成された、Ｃ９に記載の装置。
［Ｃ１４］
前記テクスチャオフセットを適用するために、前記ＧＰＵが、前記ロケーションを示す、前記現在のセグメントのテクスチャ座標値を決定することを行うように構成された、Ｃ９に記載の装置。
［Ｃ１５］
前記画素シェーディングが、前記破線をストロークすることを含む、前記破線のためのパスレンダリングプロセス中に含まれる、Ｃ９に記載の装置。
［Ｃ１６］
前記ＧＰＵが、前記セグメントの前記順序がプリミティブ順序であるように、前記セグメントのジオメトリシェーディング中に前記セグメントの前記順序を決定することを行うようにさらに構成された、Ｃ９に記載の装置。
［Ｃ１７］
グラフィックスデータをレンダリングするための装置であって、
グラフィックス処理ユニット（ＧＰＵ）を用いて、破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定するための手段と、ここにおいて、前記複数の順序付きセグメントの前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前のセグメントの長さの累積に基づく、
前記現在のセグメントのロケーションを決定するために、前記テクスチャオフセットを適用することを含めて前記現在のセグメントをピクセルシェーディングするための手段と
を備える装置。
［Ｃ１８］
前記複数のセグメントが、１つまたは複数の可視セグメントと１つまたは複数の不可視セグメントとを含み、前記装置が、
前記現在のセグメントの前記決定されたロケーションに基づいて、前記現在のセグメントが可視セグメントであるかどうかを決定するための手段と、
前記決定に基づいて前記現在のセグメントを保持するか、または破棄するための手段と
をさらに備える、Ｃ１７に記載の装置。
［Ｃ１９］
前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線をジオメトリシェーディングするための手段と、
長さの前記累積に基づいて前記テクスチャオフセットを決定することが、前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように前

記長さ値を決定するための手段と
をさらに備える、Ｃ１７に記載の装置。
［Ｃ２０］
前記長さ値を決定するための前記手段が、前記順序が前のセグメントの長さを指定するｌｉｎｅｌｅｎｇｔｈスカラー値を生成するための手段を備える、Ｃ１９に記載の装置。
［Ｃ２１］
前記複数の順序付きセグメントの前記セグメントの各々をラスタライズするための手段、
ここにおいて、前記テクスチャオフセットを適用するための前記手段が、前記現在のセグメントがラスタライズされた後に前記ラスタライズされた現在のセグメントに前記テクスチャオフセットを適用するための手段を備える、
をさらに備える、Ｃ１７に記載の装置。
［Ｃ２２］
前記テクスチャオフセットを適用するための前記手段が、前記ロケーションを示す、前記現在のセグメントのテクスチャ座標値を決定するための手段を備える、Ｃ１７に記載の装置。
［Ｃ２３］
前記セグメントの前記順序がプリミティブ順序であるように、前記セグメントのジオメトリシェーディング中に前記セグメントの前記順序を決定するための手段をさらに備える、Ｃ１７に記載の装置。
［Ｃ２４］
実行されたとき、構成されたグラフィックス処理ユニット（ＧＰＵ）に、
破線の複数の順序付きセグメントの現在のセグメントのためのテクスチャオフセットを決定することと、ここにおいて、前記複数の順序付きセグメントの前記現在のセグメントのための前記テクスチャオフセットが、前記現在のセグメントより順序が前のセグメントの長さの累積に基づく、
前記現在のセグメントのロケーションを決定するために、前記テクスチャオフセットを適用することを含めて前記現在のセグメントをピクセルシェーディングすることと
を行わせる命令を記憶した非一時的コンピュータ可読媒体。
［Ｃ２５］
前記複数のセグメントが、１つまたは複数の可視セグメントと１つまたは複数の不可視セグメントとを含み、前記命令が、前記ＧＰＵに、
前記現在のセグメントの前記決定されたロケーションに基づいて、前記現在のセグメントが可視セグメントであるかどうかを決定することと、
前記決定に基づいて前記現在のセグメントを保持するか、または破棄することと
をさらに行わせる、Ｃ２４に記載の非一時的コンピュータ可読媒体。
［Ｃ２６］
前記命令が、前記ＧＰＵに、
前記現在のセグメントのための前記テクスチャオフセットを決定する前に、前記複数の順序付きセグメントを形成するために前記破線をジオメトリシェーディングすることと、
長さの前記累積に基づいて前記テクスチャオフセットを決定することが、前記現在のセグメントの長さ値に基づいて前記テクスチャオフセットを決定することを備えるように前記長さ値を決定することと
をさらに行わせる、Ｃ２４に記載の非一時的コンピュータ可読媒体。
［Ｃ２７］
前記長さ値を決定するために、前記命令が前記ＧＰＵに、前記順序が前のセグメントの長さを指定するｌｉｎｅｌｅｎｇｔｈスカラー値を生成することを行わせる、Ｃ２６に記載の非一時的コンピュータ可読媒体。
［Ｃ２８］
前記命令が、前記ＧＰＵに、

前記複数の順序付きセグメントの前記セグメントの各々をラスタライズすること、
ここにおいて、前記テクスチャオフセットを適用するために、前記命令が、前記ＧＰＵに、前記現在のセグメントがラスタライズされた後に前記ラスタライズされた現在のセグメントに前記テクスチャオフセットを適用することを行わせる、
をさらに行わせる、Ｃ２４に記載の非一時的コンピュータ可読媒体。
［Ｃ２９］
前記テクスチャオフセットを適用するために、前記命令が、前記ＧＰＵに、前記ロケーションを示す、前記現在のセグメントのテクスチャ座標値を決定することを行わせる、Ｃ２４に記載の非一時的コンピュータ可読媒体。
［Ｃ３０］
前記命令が、前記ＧＰＵに、前記セグメントの前記順序がプリミティブ順序であるように、前記セグメントのジオメトリシェーディング中に前記セグメントの前記順序を決定することをさらに行わせる、Ｃ２４に記載の非一時的コンピュータ可読媒体。

【図１】

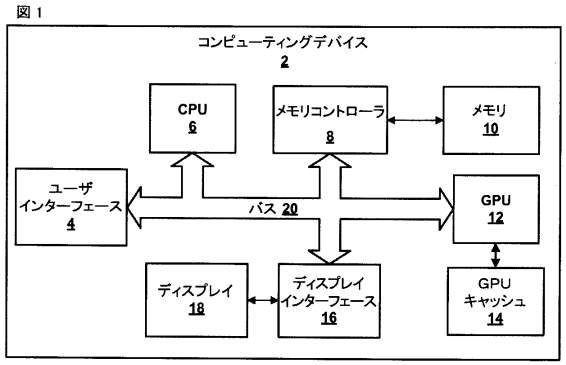

FIG. 1

【図２】

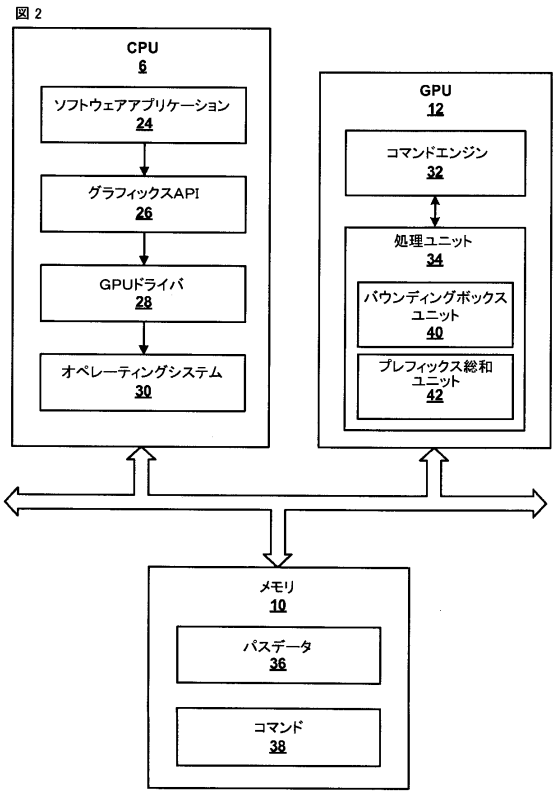

FIG. 2

【図３】

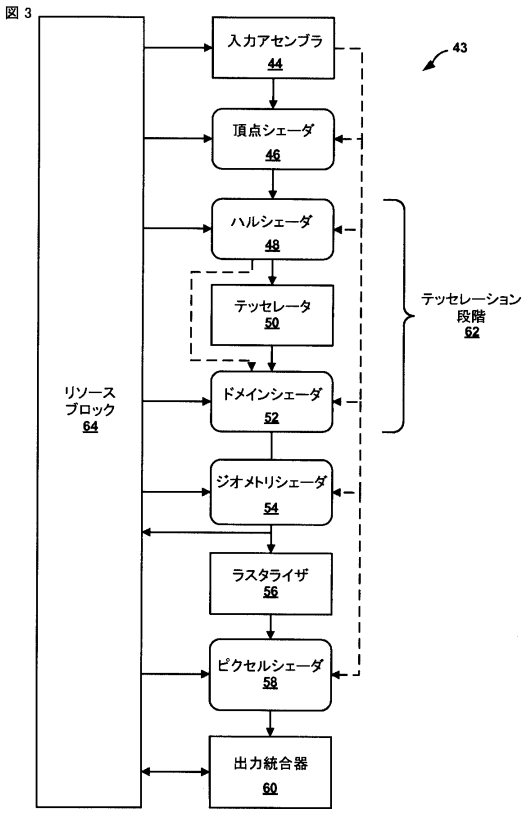

FIG. 3

【図４】

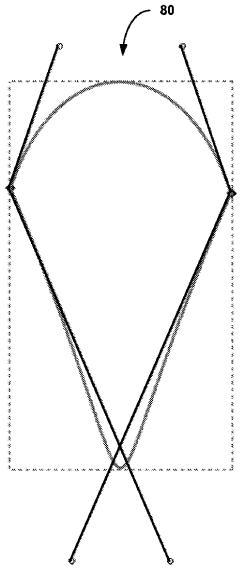

FIG. 4

【図５Ａ】

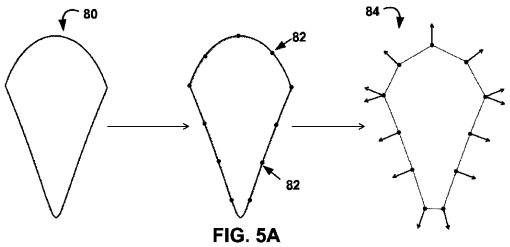

FIG. 5A

【図５Ｂ】

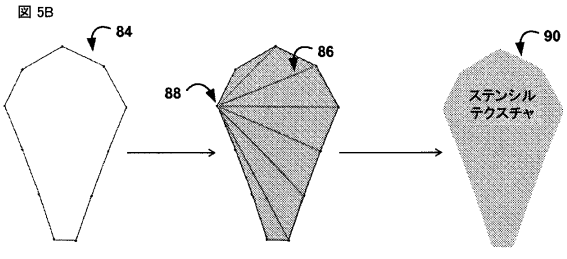

FIG. 5B

【図５Ｃ】

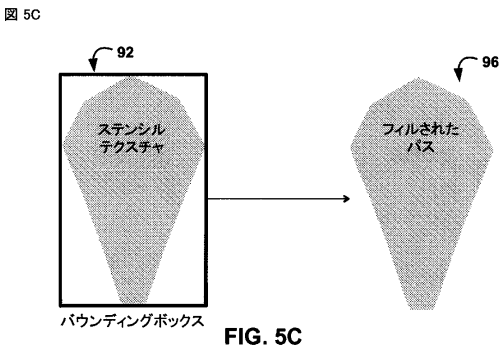

FIG. 5C

【図６】

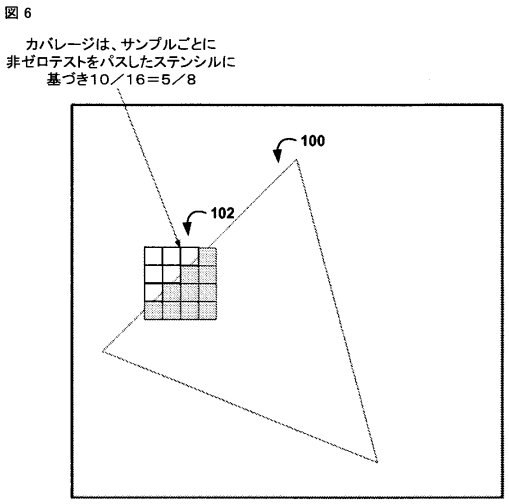

FIG. 6

【図７】

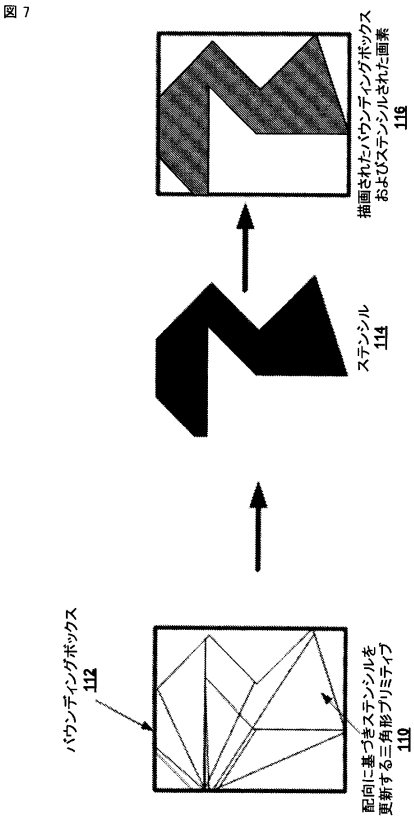

FIG. 7

【図８】

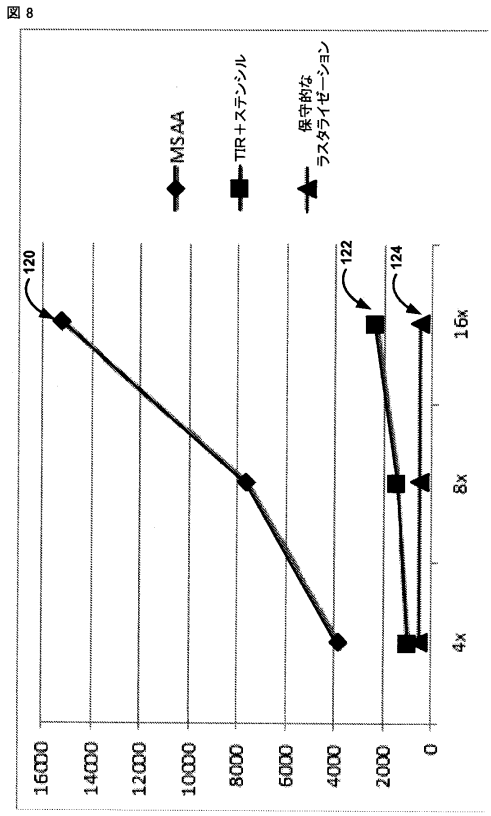

FIG. 8

【図９Ａ】

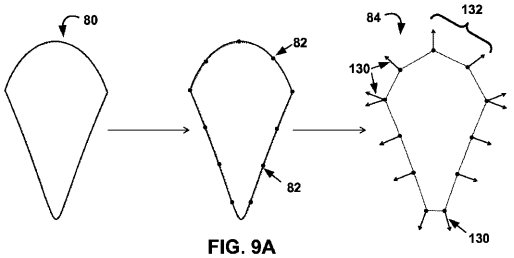

FIG. 9A

【図９Ｂ】

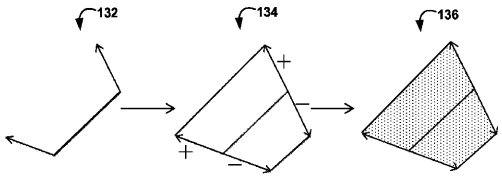

FIG. 9B

【図９Ｃ】

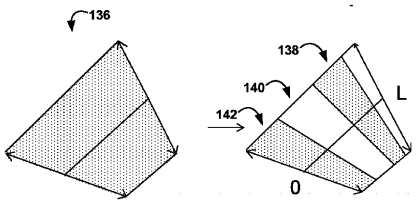

FIG. 9C

【図９Ｄ】

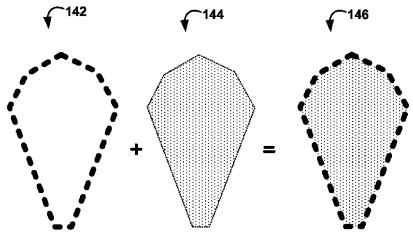

FIG. 9D

【図１０】

図 10

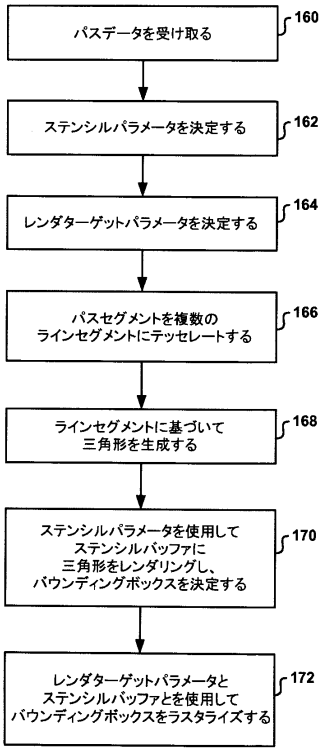

FIG. 10

【図１１】

図 11

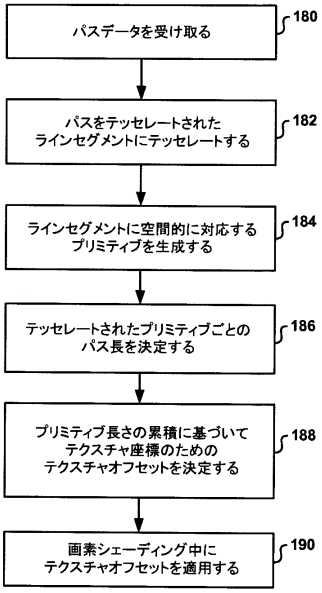

FIG. 11

フロントページの続き

(72)発明者　ゴエル、ビネート
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(72)発明者　セイラン、ウサメ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５

審査官　真木　健彦

(56)参考文献　特開２００２－０７４３７７（ＪＰ，Ａ）
特開２０１２－１８１７２６（ＪＰ，Ａ）
特開２００７－２６５０３５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｔ　１１／２０