

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B1)

(11) 特許番号

特許第6013661号

(P6013661)

(45) 発行日 平成28年10月25日(2016.10.25)

(24) 登録日 平成28年9月30日(2016.9.30)

(51) Int.Cl. F I  
**GO 6 T 11/20 (2006.01)** GO 6 T 11/20  
**GO 6 T 15/80 (2011.01)** GO 6 T 15/80

請求項の数 26 (全 52 頁)

(21) 出願番号	特願2016-534976 (P2016-534976)	(73) 特許権者	595020643
(86) (22) 出願日	平成26年7月2日(2014.7.2)		クァアルコム・インコーポレイテッド
(86) 国際出願番号	PCT/US2014/045299		QUALCOMM INCORPORATED
(87) 国際公開番号	W02015/030932		ED
(87) 国際公開日	平成27年3月5日(2015.3.5)		アメリカ合衆国、カリフォルニア州 92
審査請求日	平成28年5月26日(2016.5.26)		121-1714、サン・ディエゴ、モア
(31) 優先権主張番号	61/871, 260		ハウス・ドライブ 5775
(32) 優先日	平成25年8月28日(2013.8.28)	(74) 代理人	100108855
(33) 優先権主張国	米国 (US)		弁理士 蔵田 昌俊
(31) 優先権主張番号	14/321, 309	(74) 代理人	100109830
(32) 優先日	平成26年7月1日(2014.7.1)		弁理士 福原 淑弘
(33) 優先権主張国	米国 (US)	(74) 代理人	100158805
早期審査対象出願			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ターゲット独立ステンシル

(57) 【特許請求の範囲】

【請求項 1】

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、

画像のパスの1つまたは複数のアンチエイリアスの各々についてのカバレッジ値を決定するためのステンシルサンプリングレートを決定することと、ここにおいて、前記ステンシルサンプリングレートは、前記アンチエイリアス画素の各々について、前記ステンシルサンプリングレートでサンプリングされる前記アンチエイリアス画素の各々のサンプルを記憶するためのステンシルバッファにおけるメモリ割当てを示す、

前記ステンシルサンプリングレートとは別々に、前記ステンシルサンプリングレートよりも少ない値を有するレンダターゲットパラメータを決定することと、ここにおいて、前記レンダターゲットパラメータは、前記アンチエイリアス画素の各々について、前記パスの各レンダリングされたアンチエイリアス画素を記憶するためのフレームバッファにおけるメモリ割当てを示す、

前記ステンシルサンプリングレートと前記レンダターゲットパラメータとを使用して前記パスをレンダリングすることと

を備える方法。

【請求項 2】

前記ステンシルサンプリングレートとは異なる値である深度パラメータ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 3】

前記深度パラメータを決定することが、前記レンダターゲットパラメータに等しい深度パラメータを決定すること、ここにおいて、前記深度パラメータが、1つまたは複数の他の画素に対する前記パスの前記アンチエイリアス画素の各々の深度を示す、を備える、請求項2に記載の方法。

【請求項4】

前記レンダターゲットパラメータとは異なる値であるラスライゼーションパラメータを決定すること、ここにおいて、前記ラスライゼーションパラメータが、前記パスの前記アンチエイリアス画素の各々についてのラスライゼーションサンプリングレートを示す、をさらに備える、請求項1に記載の方法。

【請求項5】

前記パスをレンダリングすることが、  
前記ステンシルサンプリングレートを使用して前記ステンシルバッファに前記パスをレンダリングすることと、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスのアンチエイリアス画素を示す、  
前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスライズすることと  
を備える、請求項1に記載の方法。

【請求項6】

前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定すること、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、  
をさらに備え、

前記パスをラスライズすることが、前記パスのための前記バウンディングボックスを決定することと、前記パスをラスライズすることとが、1つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラスライズすることを備える、請求項5に記載の方法。

【請求項7】

前記パスの前記複数の最大境界点を決定することが、上部境界点、下部境界点、右境界点および左境界点を決定することを備える、請求項6に記載の方法。

【請求項8】

グラフィックスをレンダリングするための装置であって、  
ステンシルサンプリングレートでサンプリングされるアンチエイリアス画素のサンプルを記憶するように構成されたステンシルバッファと、前記アンチエイリアス画素は、少なくとも画像のパスの一部を形成する、

レンダリングされたアンチエイリアス画素を記憶するように構成されたフレームバッファと、

画像のパスの1つまたは複数のアンチエイリアス画素の各々についてのカバレッジ値を決定するためのステンシルサンプリングレートを決定することと、ここにおいて、前記ステンシルサンプリングレートは、前記アンチエイリアス画素の各々について、前記ステンシルサンプリングレートでサンプリングされる前記アンチエイリアス画素の各々のサンプルを記憶するためのステンシルバッファにおけるメモリ割当てを示す、

前記ステンシルサンプリングレートとは別々に、前記ステンシルサンプリングレートよりも少ない値を有するレンダターゲットパラメータを決定することと、ここにおいて、前記レンダターゲットパラメータは、前記アンチエイリアス画素の各々について、前記パスの各レンダリングされたアンチエイリアス画素を記憶するためのフレームバッファにおけるメモリ割当てを示す、

前記ステンシルサンプリングレートと前記レンダターゲットパラメータとを使用して前記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)と

10

20

30

40

50

を備える装置。

【請求項 9】

前記 GPU が、前記ステンシルサンプリングレートとは異なる値である深度パラメータを決定することを行うようにさらに構成された、請求項 8 に記載の装置。

【請求項 10】

前記深度パラメータを決定するために、前記 GPU が、前記レンダターゲットパラメータに等しい深度パラメータを決定すること、ここにおいて、前記深度パラメータが、1つまたは複数の他の画素に対する前記パスの前記アンチエイリアス画素の各々の深度を示す、を行うように構成された、請求項 9 に記載の装置。

【請求項 11】

前記 GPU が、前記レンダターゲットパラメータとは異なる値であるラスタライゼーションパラメータを決定すること、ここにおいて、前記ラスタライゼーションパラメータが、前記パスの前記アンチエイリアス画素の各々についてのラスタライゼーションサンプリングレートを示す、を行うようにさらに構成された、請求項 8 に記載の装置。

【請求項 12】

前記パスをレンダリングするために、前記 GPU が、  
前記ステンシルサンプリングレートを使用して前記ステンシルバッファに前記パスをレンダリングすることと、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスのアンチエイリアス画素を示す、  
前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスタライズすることと  
を行うように構成された、請求項 8 に記載の装置。

【請求項 13】

前記 GPU が、  
前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定すること、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、  
を行うようにさらに構成された、  
前記パスをラスタライズするために、前記 GPU が、前記パスのための前記バウンディングボックスを決定することと、前記パスをラスタライズすることとが、1つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラスタライズすることを行うように構成された、請求項 12 に記載の装置。

【請求項 14】

前記パスの前記複数の最大境界点を決定するために、前記 GPU が、上部境界点、下部境界点、右境界点および左境界点を決定することを行うように構成された、請求項 13 に記載の装置。

【請求項 15】

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、  
画像のパスの1つまたは複数のアンチエイリアス画素の各々についてのカバレッジ値を決定するためのステンシルサンプリングレートを決定するための手段と、ここにおいて、前記ステンシルサンプリングレートは、前記アンチエイリアス画素の各々について、前記ステンシルサンプリングレートでサンプリングされる前記アンチエイリアス画素の各々のサンプルを記憶するためのステンシルバッファにおけるメモリ割当てを示す、  
前記ステンシルサンプリングレートとは別々に、前記ステンシルサンプリングレートよりも少ない値を有するレンダターゲットパラメータを決定するための手段と、ここにおいて、前記レンダターゲットパラメータは、前記アンチエイリアス画素の各々について、前記パスの各レンダリングされたアンチエイリアス画素を記憶するためのフレームバッファにおけるメモリ割当てを示す、  
前記ステンシルサンプリングレートと前記レンダターゲットパラメータとを使用して前

記パスをレンダリングするための手段と  
を備える装置。

【請求項 1 6】

前記ステンシルサンプリングレートとは異なる値である深度パラメータを決定するための手段をさらに備える、請求項 1 5 に記載の装置。

【請求項 1 7】

前記深度パラメータを決定するための前記手段が、前記レンダターゲットパラメータに等しい深度パラメータを決定するための手段、ここにおいて、前記深度パラメータが、1 つまたは複数の他の画素に対する前記パスの前記アンチエイリアス画素の各々の深度を示す、を備える、請求項 1 6 に記載の装置。

10

【請求項 1 8】

前記レンダターゲットパラメータとは異なる値であるラスタライゼーションパラメータを決定するための手段、ここにおいて、前記ラスタライゼーションパラメータが、前記パスの前記アンチエイリアス画素の各々についてのラスタライゼーションサンプリングレートを示す、をさらに備える、請求項 1 5 に記載の装置。

【請求項 1 9】

前記パスをレンダリングするための前記手段が、

前記ステンシルサンプリングレートを使用して前記ステンシルバッファに前記パスをレンダリングするための手段と、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスのアンチエイリアス画素を示す、

20

前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスタライズするための手段と

を備える、請求項 1 5 に記載の装置。

【請求項 2 0】

前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定するための手段、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、

をさらに備え、

前記パスをラスタライズするための前記手段が、前記パスのための前記バウンディングボックスを決定することと、前記パスをラスタライズすることとが、1 つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラスタライズするための手段を備える、請求項 1 9 に記載の装置。

30

【請求項 2 1】

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1 つまたは複数のアンチエイリアス画素の各々についてのカバレッジ値を決定するためのステンシルサンプリングレートを決定することと、ここにおいて、前記ステンシルサンプリングレートは、前記アンチエイリアス画素の各々について、前記ステンシルサンプリングレートでサンプリングされる前記アンチエイリアス画素の各々のサンプルを記憶するためのステンシルバッファにおけるメモリ割当てを示す、

40

前記ステンシルサンプリングレートとは別々に、前記ステンシルサンプリングレートよりも少ない値を有するレンダターゲットパラメータを決定することと、ここにおいて、前記レンダターゲットパラメータは、前記アンチエイリアス画素の各々について、前記パスの各レンダリングされたアンチエイリアス画素を記憶するためのフレームバッファにおけるメモリ割当てを示す、

前記ステンシルサンプリングレートと前記レンダターゲットパラメータとを使用して前記パスをレンダリングすることと

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。

【請求項 2 2】

前記命令が、前記 GPU に、前記ステンシルサンプリングレートとは異なる値である深

50

度パラメータを決定することをさらに行わせる、請求項 2 1 に記載の非一時的コンピュータ可読媒体。

【請求項 2 3】

前記深度パラメータを決定するために、前記命令が、前記 GPU に、前記レンダーターゲットパラメータに等しい深度パラメータを決定すること、ここにおいて、前記深度パラメータが、1 つまたは複数の他の画素に対する前記パスの前記アンチエイリアス画素の各々の深度を示す、を行わせる、請求項 2 2 に記載の非一時的コンピュータ可読媒体。

【請求項 2 4】

前記命令が、前記 GPU に、前記レンダーターゲットパラメータとは異なる値であるラストライゼーションパラメータを決定すること、ここにおいて、前記ラストライゼーションパラメータが、前記パスの前記アンチエイリアス画素の各々についてのラストライゼーションサンプリングレートを示す、をさらに行わせる、請求項 2 1 に記載の非一時的コンピュータ可読媒体。

【請求項 2 5】

前記パスをレンダリングするために、前記命令が、前記 GPU に、  
前記ステンシルサンプリングレートを使用して前記ステンシルバッファに前記パスをレンダリングすることと、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスのアンチエイリアス画素を示す、

前記レンダーターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラストライズすることと

を行わせる、請求項 2 1 に記載の非一時的コンピュータ可読媒体。

【請求項 2 6】

前記命令が、前記 GPU に、

前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定すること、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、

を行わせる、

前記パスをラストライズするために、前記 GPU が、前記パスのための前記バウンディングボックスを決定することと、前記パスをラストライズすることとが、1 つのレンダリングパス中で実行されるように、前記レンダーターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラストライズすることを行うように構成された、請求項 2 5 に記載の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。

【 0 0 0 4 】

[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ます、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ります。

【発明の概要】

10

【 0 0 0 5 】

[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。

【 0 0 0 6 】

[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口ケーション（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。

20

【 0 0 0 7 】

[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ッ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。

30

【 0 0 0 8 】

[0008]別の例では、グラフィックス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ッ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。

【 0 0 0 9 】

[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ッ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。

40

【 0 0 1 0 】

[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ッ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

50

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を含む。

10

【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。

20

【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。

30

【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る。

【図面の簡単な説明】

【0016】

40

【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。

50

【図 5 C】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 9 A】[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 9 B】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 9 C】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 9 D】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。

10

【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。

20

【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。

30

【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パイ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。

40

【0020】

50



[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）DirectX 11（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。

10

#### 【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

20

#### 【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、組み合わせられ得る（たとえば、平均化され得る）。

30

#### 【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）。

40

#### 【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

50

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。

10

#### 【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ッ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ッ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。

20

#### 【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。

30

#### 【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、ラスタライゼーション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。

40

#### 【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

50

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。

10

#### 【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ため、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。

20

#### 【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。

30

#### 【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。

40

#### 【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

50

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】

10

20

30

40

50

[0049] CPU 6 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 GPU 12 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、CPU 6 は、GPU 12 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 GPU 12 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 CPU 6 および / または GPU 12 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、CPU 6、GPU 12、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、CPU 6、GPU 12、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、CPU 6 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 CPU 6 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、GPU 12 のデバイスメモリとして機能することができ、GPU 12 によって演算されことになるデータ、ならびに GPU 12 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、GPU 12 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)、電氣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] GPU 12 は、CPU 6 によって GPU 12 に発行されたコマンドを実行するように構成され得る。GPU 12 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、GPU 12 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、CPU 6 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、CPU 6 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 GPU 12 に提供して、GPU 12 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、CPU 6 は、コマンドとグラフィックスデータとを GPU 12 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 GPU 12 に提供することができる。

【0044】

[0054]さらなる例では、GPU 12 は、CPU 6 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、CPU 6 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 GPU 12 にオフロードすることを決定するとき、CPU 6 は、汎用コンピューティングデータを GPU 12 に提供して、GPU 12 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、CPU 6 は、コマンドとグラフィックスデータとを GPU 12 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 GPU 12 に提供することができる。

【0045】

[0055]GPU 12 は、いくつかの例では、ベクタ演算について CPU 6 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、GPU 12 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。GPU 12 の高並列性質は、いくつかの例では、GPU 12 が、CPU 6 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、GPU 12 の高並列性質は、GPU 12 が、CPU 6 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、GPU 12 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、GPU 12 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、GPU 12 は、システムオンチップ (SoC) を形成する CPU 6 と同じマイクロチップ上に配置され得る。GPU 12 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(SEED)、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 eXtensible 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。

【 0 0 5 1 】

[0061]いくつかの事例では、GPU 12は、部分的 - 全体的 (partial-to-total) GPUベースの様々なバ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、CPU 6は、GPU 12に対する1つまたは複数のバスレンダリングコマンドを発行することができ、GPU 12は、バ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、CPU 6は、GPU 12にバスフィル動作を実行するように命令する1つまたは複数のバスフィルコマンドをGPU 12に発行することができ、GPU 12は、バ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、CPU 6は、GPU 12にバスストローク動作を実行するように命令する1つまたは複数のバスストロークコマンドをGPU 12に発行することができ、GPU 12は、バスストロークコマンドを実行することができる。

10

【 0 0 5 2 】

[0062]いくつかの例では、GPU 12は、レンダリングされることになるバスのバ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バ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バ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バ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バ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。

20

【 0 0 5 3 】

[0063]いくつかの例では、GPU 12は、バスフィル動作を実行するために、2バ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、GPU 12は、レンダリングされることになるバスのバ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バ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。GPU 12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。GPU 12は、どの画素がバ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、GPU 12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、GPU 12は、バ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バ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。

30

【 0 0 5 4 】

[0064]バ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、GPU 12は、いくつかの例では、バ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、GPU 12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。

40

【 0 0 5 5 】

[0065]したがって、バスレンダリングを実行するとき、GPU 12は、バスをフィルするために、ラインセグメントにバ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

50



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ッ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ッ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、各画素のカバレッジ値を決定し、ステンシル内に位置するとGPU1

10

20

30

40

50

2 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、GPU 12 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、GPU 12 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、GPU 12 は、ストロ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、GPU 12 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、GPU 12 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CPU 6 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、GPU 12 がパスを単独でレンダリングするのを防ぎ得る。

10

【0062】

[0072] 本開示の態様によれば、GPU 12 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、GPU 12 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。GPU 12 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、GPU 12 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。GPU 12 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、GPU 12 は、たとえば、CPU 6 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。

20

【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、GPU 12 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、GPU 12 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、GPU 12 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。

30

【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、CPU 6 と、GPU 12 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GPU 12 のグラフィックスパイプラインにおいて）GPU 12 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、CPU 6 は、本開示のパスレンダリング技法を実行するGPU 12 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、シェード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、CPU 6 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GPU 12 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。

40

50

## 【 0 0 6 5 】

[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。

10

## 【 0 0 6 6 】

[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。

20

## 【 0 0 6 7 】

[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、DX11グラフィックスレンダリングパイプライン(すなわち、DX11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。

30

## 【 0 0 6 8 】

[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つもしくは複数の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。

40

## 【 0 0 6 9 】

50

[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】

10

[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。

20

【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、シザータスト、アルファブレンディングなど、ラスト演算を実行する処理ユニットを含み得る。

【0072】

30

[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。

40

【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、 $(x_0, y_0)$ および $(x_1, y_1)$ ）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、 $(x_0, y_0)$ 、 $(x_1, y_1)$ 、 $(x_2, y_2)$ 、 $(x_3, y_3)$ ）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

50

ラメータ表示を示すデータを含み得る。

【 0 0 7 4 】

[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。

【 0 0 7 5 】

[0085]CPU 6 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、OpenGL 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。

【 0 0 7 6 】

[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、GPU 12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GPU 12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つもしくは複数の設定され得る。

【 0 0 7 7 】

[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GPU 12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、GPU 12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、GPU 12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GPU 12に渡される。

【 0 0 7 8 】

[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

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ッ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)のカバレッ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。このカバレッジ値は、stencil ed\_\_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Input Coverage値は、stencil ed\_\_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)からサンブラに(Input Coverage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

10

20

30

40

50

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 `point size` のシステム解釈値と同様の方法で `line length` 値を計算することができる。すなわち、`line length` 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] GPU 12 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 `line length` 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、GPU 12 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値(たとえば、`point size` 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)48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、GPU 12 によって実装され、および/またはGPU 12 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

10

20

30

40

50



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。

10

#### 【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。

20

#### 【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ンネルが提供され得る。

30

#### 【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。

40

#### 【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

50

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、 $(x_0, y_0)$  および  $(x_1, y_1)$ ）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、 $(x_0, y_0)$ 、 $(x_1, y_1)$ 、 $(x_2, y_2)$ 、 $(x_3, y_3)$ ）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つの制御点を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 ) },
```

## 【 0 1 1 0 】

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。

10

## 【 0 1 1 1 】

[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 ) },
```

20

## 【 0 1 1 2 】

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、（スケール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、（スケール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。

30

40

## 【 0 1 1 3 】

[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】

10

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。

20

## 【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。

30

## 【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】

40

```

{ 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

50

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]シオン段階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

10

20

30

40

50

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。

10

#### 【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散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。

20

#### 【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、 $t = 0/T, 1/T, 2/T, \dots, T/T$ 、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。

30

#### 【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。

40

#### 【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

50

、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再度起動され得る。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、GPU 12 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、GPU 12 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。GPU 12 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

10

20

30

40

50

イバ 2 8 ( 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 6 4 中に割り当てられ得る。テッセレーション段階 6 2 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、GPU 1 2 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 4 3 を使用することができる。

【 0 1 5 0 】

[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 4 3 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 5 4 は、ドメインシェーダ 5 2 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 5 4 ( またはグラフィックスパイプライン 4 3 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。

【 0 1 5 1 】

[0145] さらに、ジオメトリシェーダ 5 4 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。

【 0 1 5 2 】

[0146] ラスタライザ 5 6 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 5 8 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 5 8 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 5 8 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。

【 0 1 5 3 】

[0147] 図 4 は、レンダリングされることになる例示的なパス 8 0 の図である。たとえば、パス 8 0 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 8 0 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、patch 4 プリム ( プリミティブ ) にパックされ得る。たとえば、パス 8 0 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】

10

```
{ 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)
```

## 【 0 1 5 4 】

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 . 0 f 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。

20

## 【 0 1 5 5 】

[0148] 図 5 A ~ 図 5 C は、図 4 に示されるパス 8 0 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 8 0 は、例示のために、図 5 A ~ 図 5 C 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 G P U 1 2 に関して説明したが、図 5 A ~ 図 5 C 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。

30

## 【 0 1 5 6 】

[0149] 図 5 A に示されるように、G P U 1 2 は、ラインストリップ方式 8 4 で接続されたいくつかの頂点 8 2 を含めるためにパス 8 0 をテッセレートする。図 5 B に示されるように、G P U 1 2 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 8 8 に接続されたいくつかのラインセグメント 8 6 を生成する。図 5 B の例では、パス 8 0 の相対的な第 1 の頂点は、ピボット点 8 8 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 8 8 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）。

40

## 【 0 1 5 7 】

[0150] この例では、G P U 1 2 は、ステンシル中に図 5 B 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 9 0 にのみ影響を及ぼす。すなわち、ステンシルテクスチャ 9 0 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。

50

## 【 0 1 5 8 】

[0151]図 5 C に示されるように、GPU 12 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。GPU 12 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、GPU 12 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。

## 【 0 1 5 9 】

[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、GPU 12 が 16 x 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。

10

## 【 0 1 6 0 】

[0153]本開示の態様によれば、GPU 12 は、ステンシル TIR を実行することができる。したがって、GPU 12 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、GPU 12 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ッジ値を決定することができる。

## 【 0 1 6 1 】

[0154]いくつかの例では、GPU 12 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、GPU 12 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、GPU 12 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、GPU 12 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、GPU 12 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。

20

## 【 0 1 6 2 】

[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 のためのカバレッジマスクは画素 102 を含み得、GPU 12 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。

30

## 【 0 1 6 3 】

[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。

## 【 0 1 6 4 】

[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 GPU 12 によって実行され得る。

40

## 【 0 1 6 5 】

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

50

//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】

10

[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】

20

[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

30

40

【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。

50

【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)

10

## 【 0 1 6 9 】

[0161] 図 9 A ~ 図 9 D 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 9 A ~ 図 9 D の例では、パス 8 0 はテッセレートされている。さらに、説明のために G P U 1 2 に関して説明したが、図 9 A ~ 図 9 D

20

## 【 0 1 7 0 】

[0162] 図 9 A に示されるように、G P U 1 2 は、ラインストリップ 8 4 中で接続されたいくつかの頂点 8 2 を含めるためにパス 8 0 をテッセレートする。さらに、G P U 1 2 は、（頂点から延びる矢印として示されている）法線 1 3 0 の数を決定する。図 9 A の例では、二重の法線 1 3 0 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 9 A はまた、例示的なセグメント 1 3 2 を含む。

## 【 0 1 7 1 】

[0163] 図 9 B に、セグメント 1 3 2 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 G P U 1 2 がセグメント 1 3 2 に対して実行することを示す。説明のために図 9 B の例に追加の太いストロークが示される。図 9 B に、膨張されたセグメント 1 3 4 と、ストロークされ膨張されたセグメント 1 3 6 とを示す。

30

## 【 0 1 7 2 】

[0164] 図 9 C に、ダッシングし、ストロークされ、膨張されたセグメント 1 3 6 を示す。たとえば、G P U 1 2 は、第 1 のダッシュセグメント 1 3 8 と、第 2 のダッシュセグメント 1 4 0 と、第 3 のダッシュセグメント 1 4 2 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 1 3 8 と第 3 のダッシュセグメント 1 4 2 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 1 4 0 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、G P U 1 2 は、各ダッシュセグメント（線）1 3 8、1 4 0、および 1 4 2 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 4 2 は、ジオメトリシェーディング中にダッシュセグメント 1 3 8 ~ 1 4 2 の長さを累積することができる。

40

## 【 0 1 7 3 】

[0165] G P U 1 2 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 4 2 は、セグメント 1 3 8 ~ 1 4 2 の各々のロケーションを示す line length 値を計算することができる。プレフィックス総和装置 4 2 は、ピクセルシェーディング段階にプレフィックス総和した line length 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 1 3 8 ~ 1 4 2 のそれぞれのロケーションを決定す

50



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、ピクセルシェード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。

10

【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。

20

【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ッ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。

30

【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。

40

【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】

50

[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。

【 0 1 8 1 】

[0173] GPU 12 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 ( 1 7 2 )。たとえば、本開示の態様によれば、GPU 12 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 T I R 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。

【 0 1 8 2 】

[0174] 図 1 1 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 GPU 12 によって実行されるものとして説明したが、図 1 1 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。

【 0 1 8 3 】

[0175] GPU 12 は、パスデータを受け取る ( 1 8 0 )。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。GPU 12 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 ( 1 8 2 )。たとえば、GPU 12 は、図 9 A 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。

【 0 1 8 4 】

[0176] GPU 12 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 ( 1 8 4 )。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、GPU 12 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。GPU 12 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、GPU 12 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。

【 0 1 8 5 】

[0177] ダッシュするとき、GPU 12 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 ( 1 8 6 )。たとえば、GPU 12 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、GPU 12 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。

【 0 1 8 6 】

[0178] GPU 12 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 ( 1 8 8 )。たとえば、上記のように、GPU 12 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。GPU 12 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 ( 1 9 0 )。たとえば、GPU 12 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

ツ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 X 11ハードウェアのユーザは、Direct X 11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(DSP)、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)、電氣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

10

20

30

40

50

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、出願当初の特許請求の範囲に記載された発明を付記する。

[請求項1]

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、  
画像のパスの各アンチエイリアス画素のカバレッ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、  
前記ステンシルパラメータとは別々に、前記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、  
前記ステンシルパラメータと前記レンダターゲットパラメータとを使用して前記パスをレンダリングすることと  
を備える方法。

[請求項2]

前記ステンシルパラメータを決定することが、前記レンダターゲットパラメータよりも大きいステンシルパラメータを決定することを備える、請求項1に記載の方法。

[請求項3]

前記ステンシルパラメータとは異なる値である深度パラメータ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4]

前記深度パラメータを決定することが、前記レンダターゲットパラメータに等しい深度パラメータを決定すること、ここにおいて、前記深度パラメータが、1つまたは複数の他の画素に対する前記パスの各画素の深度を示す、を備える、請求項3に記載の方法。

[請求項5]

前記レンダターゲットパラメータとは異なる値であるラスタライゼーションパラメータを決定すること、ここにおいて、前記ラスタライゼーションパラメータが、前記パスの各アンチエイリアス画素のラスタライゼーションサンプリングレートを示す、をさらに備える、請求項1に記載の方法。

[請求項6]

前記パスをレンダリングすることが、  
前記ステンシルパラメータを使用してステンシルバッファに前記パスをレンダリングすることと、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスの画素を示す、  
前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスタライズすることと  
を備える、請求項1に記載の方法。

[請求項7]

前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定すること、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、  
をさらに備え、  
前記パスをラスタライズすることが、前記パスのための前記バウンディングボックスを

10

20

30

40

50

決定することと、前記パスをラスタライズすることとが、1つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラスタライズすることを備える、請求項6に記載の方法。

[ 請求項 8 ]

前記パスの前記複数の最大境界点を決定することが、上部境界点、下部境界点、右境界点および左境界点を決定することを備える、請求項7に記載の方法。

[ 請求項 9 ]

グラフィックスをレンダリングするための装置であって、  
画像のパスの各アンチエイリアス画素のカバレッ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、

前記ステンシルパラメータとは別々に、前記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、

前記ステンシルパラメータと前記レンダターゲットパラメータとを使用して前記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) を備える装置。

[ 請求項 10 ]

前記ステンシルパラメータを決定するために、前記GPUが、前記レンダターゲットパラメータよりも大きいステンシルパラメータを決定することを行うように構成された、請求項9に記載の装置。

[ 請求項 11 ]

前記GPUが、前記ステンシルパラメータとは異なる値である深度パラメータを決定することを行うようにさらに構成された、請求項9に記載の装置。

[ 請求項 12 ]

前記深度パラメータを決定するために、前記GPUが、前記レンダターゲットパラメータに等しい深度パラメータを決定すること、ここにおいて、前記深度パラメータが、1つまたは複数の他の画素に対する前記パスの各画素の深度を示す、を行うように構成された、請求項11に記載の装置。

[ 請求項 13 ]

前記GPUが、前記レンダターゲットパラメータとは異なる値であるラスタライゼーションパラメータを決定すること、ここにおいて、前記ラスタライゼーションパラメータが、前記パスの各アンチエイリアス画素のラスタライゼーションサンプリングレートを示す、を行うようにさらに構成された、請求項9に記載の装置。

[ 請求項 14 ]

前記パスをレンダリングするために、前記GPUが、  
前記ステンシルパラメータを使用してステンシルバッファに前記パスをレンダリングすることと、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスの画素を示す、

前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスタライズすることと

を行うように構成された、請求項9に記載の装置。

[ 請求項 15 ]

前記GPUが、  
前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定すること、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、

を行うようにさらに構成された、

前記パスをラスタライズするために、前記GPUが、前記パスのための前記バウンディングボックスを決定することと、前記パスをラスタライズすることとが、1つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッ

10

20

30

40

50

ファトを使用して前記バウンディングボックスをラスタライズすることを行うように構成された、請求項 1 4 に記載の装置。

[ 請求項 1 6 ]

前記パスの前記複数の最大境界点を決定するために、前記 G P U が、上部境界点、下部境界点、右境界点および左境界点を決定することを行うように構成された、請求項 1 5 に記載の装置。

[ 請求項 1 7 ]

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、  
画像のパスの各アンチエイリアス画素のカバレッ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、

前記ステンシルパラメータとは別々に、前記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、

前記ステンシルパラメータと前記レンダターゲットパラメータとを使用して前記パスをレンダリングするための手段と

を備える装置。

[ 請求項 1 8 ]

前記ステンシルパラメータを決定するための前記手段が、前記レンダターゲットパラメータよりも大きいステンシルパラメータを決定するための手段を備える、請求項 1 7 に記載の装置。

[ 請求項 1 9 ]

前記ステンシルパラメータとは異なる値である深度パラメータを決定するための手段をさらに備える、請求項 1 7 に記載の装置。

[ 請求項 2 0 ]

前記深度パラメータを決定するための前記手段が、前記レンダターゲットパラメータに等しい深度パラメータを決定するための手段、ここにおいて、前記深度パラメータが、1 つまたは複数の他の画素に対する前記パスの各画素の深度を示す、を備える、請求項 1 9 に記載の装置。

[ 請求項 2 1 ]

前記レンダターゲットパラメータとは異なる値であるラスタライゼーションパラメータを決定するための手段、ここにおいて、前記ラスタライゼーションパラメータが、前記パスの各アンチエイリアス画素のラスタライゼーションサンプリングレートを示す、をさらに備える、請求項 1 7 に記載の装置。

[ 請求項 2 2 ]

前記パスをレンダリングするための前記手段が、

前記ステンシルパラメータを使用してステンシルバッファに前記パスをレンダリングするための手段と、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスの画素を示す、

前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスタライズするための手段と

を備える、請求項 1 7 に記載の装置。

[ 請求項 2 3 ]

前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定するための手段、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、

をさらに備え、

前記パスをラスタライズするための前記手段が、前記パスのための前記バウンディングボックスを決定することと、前記パスをラスタライズすることとが、1 つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラスタライズするための手段を備える、請求項 2 2 に記載の装置。

10

20

30

40

50

[ 請求項 2 4 ]

実行されたとき、グラフィックス処理ユニット（ G P U ）に、  
画像のパスの各アンチエイリアス画素のカバレッ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、  
前記ステンシルパラメータとは別々に、前記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、  
前記ステンシルパラメータと前記レンダターゲットパラメータとを使用して前記パスをレンダリングすることと  
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。

[ 請求項 2 5 ]

前記ステンシルパラメータを決定するために、前記命令が、前記 G P U に、前記レンダターゲットパラメータよりも大きいステンシルパラメータを決定することを行わせる、請求項 2 4 に記載の非一時的コンピュータ可読媒体。

[ 請求項 2 6 ]

前記命令が、前記 G P U に、前記ステンシルパラメータとは異なる値である深度パラメータを決定することをさらに行わせる、請求項 2 4 に記載の非一時的コンピュータ可読媒体。

[ 請求項 2 7 ]

前記深度パラメータを決定するために、前記命令が、前記 G P U に、前記レンダターゲットパラメータに等しい深度パラメータを決定すること、ここにおいて、前記深度パラメータが、1 つまたは複数の他の画素に対する前記パスの各画素の深度を示す、を行わせる、請求項 2 6 に記載の非一時的コンピュータ可読媒体。

[ 請求項 2 8 ]

前記命令が、前記 G P U に、前記レンダターゲットパラメータとは異なる値であるラスタライゼーションパラメータを決定すること、ここにおいて、前記ラスタライゼーションパラメータが、前記パスの各アンチエイリアス画素のラスタライゼーションサンプリングレートを示す、をさらに行わせる、請求項 2 4 に記載の非一時的コンピュータ可読媒体。

[ 請求項 2 9 ]

前記パスをレンダリングするために、前記命令が、前記 G P U に、  
前記ステンシルパラメータを使用してステンシルバッファに前記パスをレンダリングすることと、ここにおいて、前記ステンシルバッファが、シェーディングされることになる前記パスの画素を示す、  
前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記パスをラスタライズすることと  
を行わせる、請求項 2 4 に記載の非一時的コンピュータ可読媒体。

[ 請求項 3 0 ]

前記命令が、前記 G P U に、  
前記ステンシルバッファに前記パスをレンダリングする間に、前記パスのためのバウンディングボックスを決定すること、ここにおいて、前記バウンディングボックスを決定することが、前記パスの複数の最大境界点を決定することを備える、  
を行わせる、  
前記パスをラスタライズするために、前記 G P U が、前記パスのための前記バウンディングボックスを決定することと、前記パスをラスタライズすることとが、1 つのレンダリングパス中で実行されるように、前記レンダターゲットパラメータと前記ステンシルバッファとを使用して前記バウンディングボックスをラスタライズすることを行うように構成された、請求項 2 9 に記載の非一時的コンピュータ可読媒体。

【要約】

一例では、グラフィックスデータをレンダリングすることは、画像のパスの各アンチエイリアス画素のカバレッ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

10

20

30

40

50

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、ステンス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を含む。

【図 4】

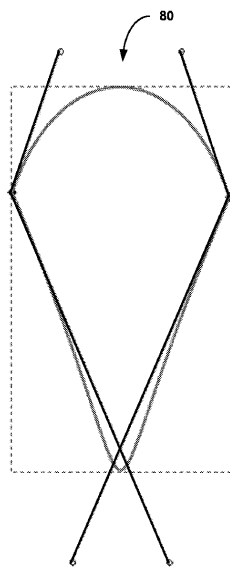
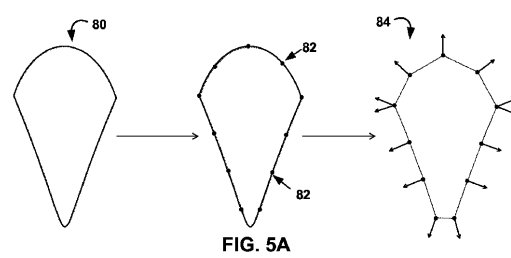
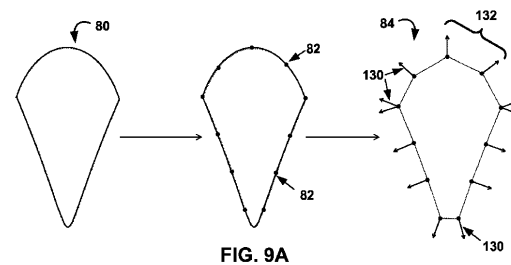


FIG. 4

【図 5 A】



【図 9 A】





【図 9 B】

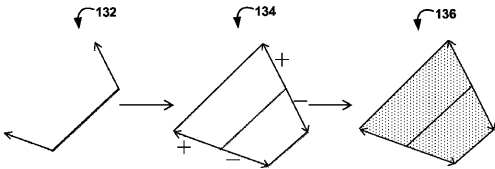


FIG. 9B

【図 9 D】

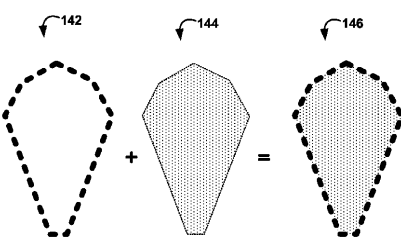


FIG. 9D

【図 9 C】

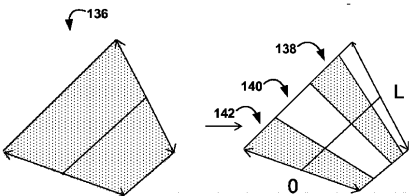


FIG. 9C

【図 1】

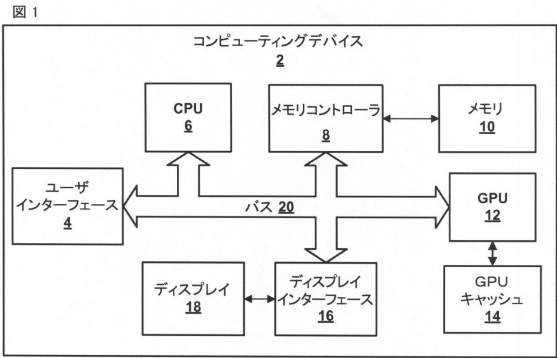


FIG. 1

【図 2】

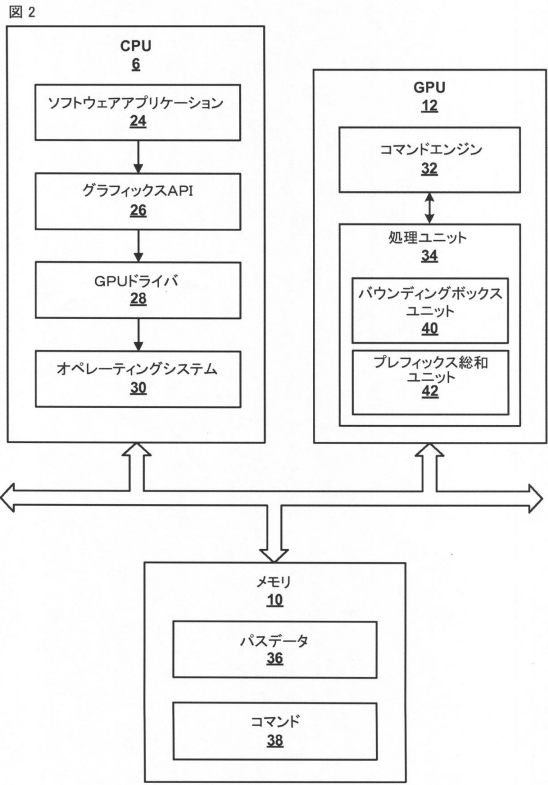


FIG. 2

【図 3】

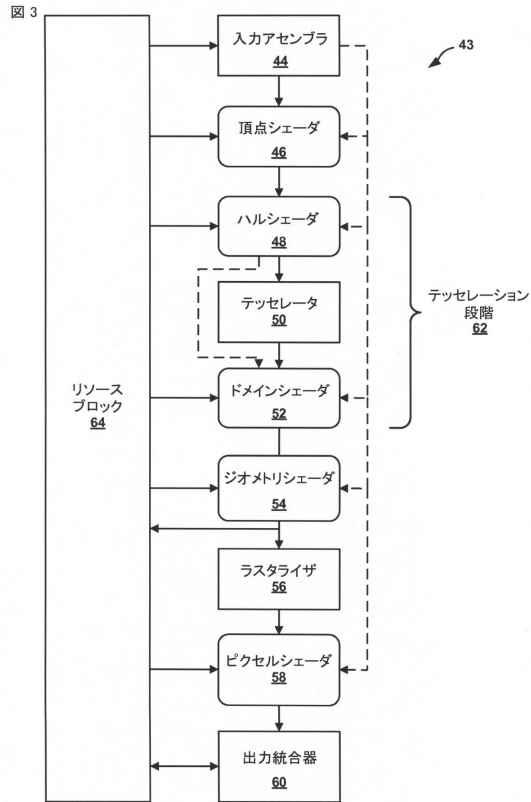


FIG. 3

【図 5 B】

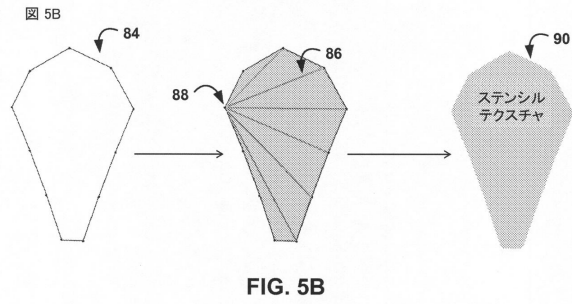


FIG. 5B

【図 5 C】

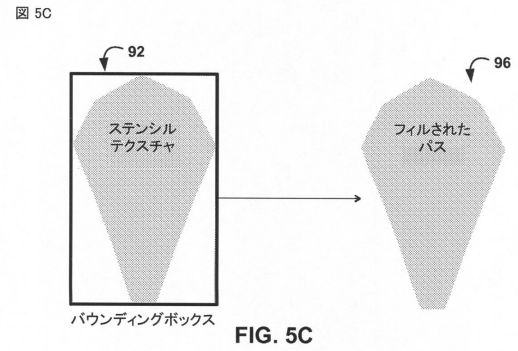


FIG. 5C

【図 6】

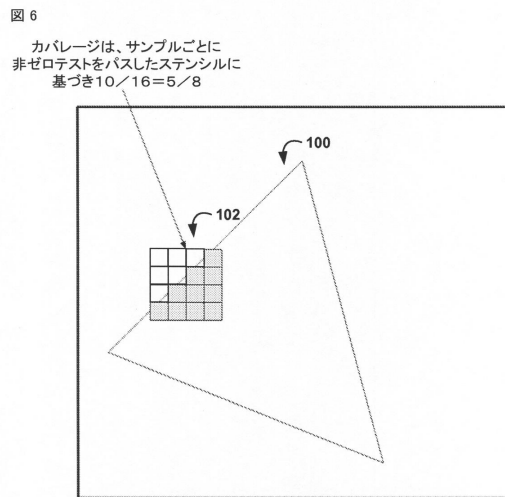


FIG. 6

【図 7】

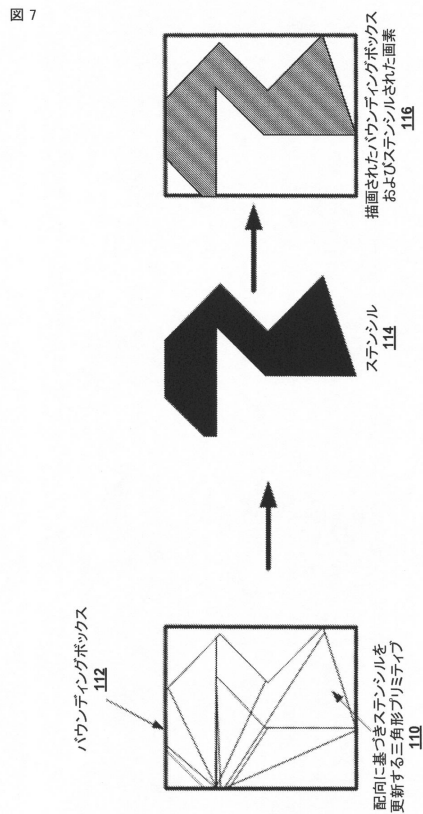


FIG. 7

【図 8】

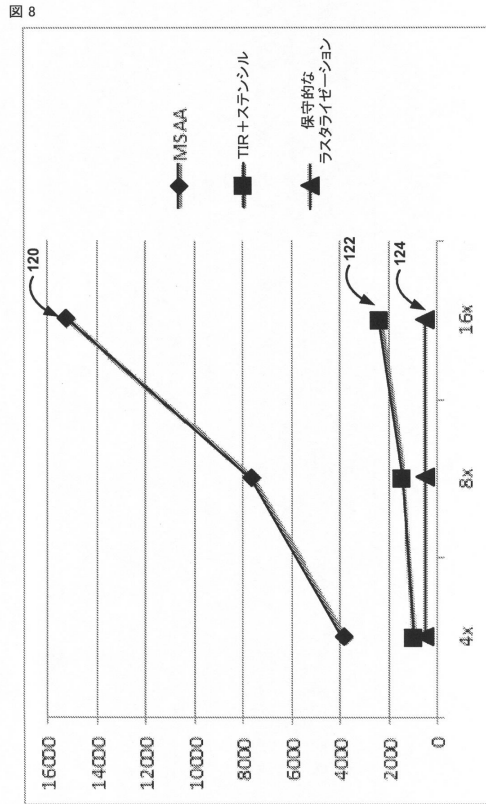


FIG. 8

【図 10】

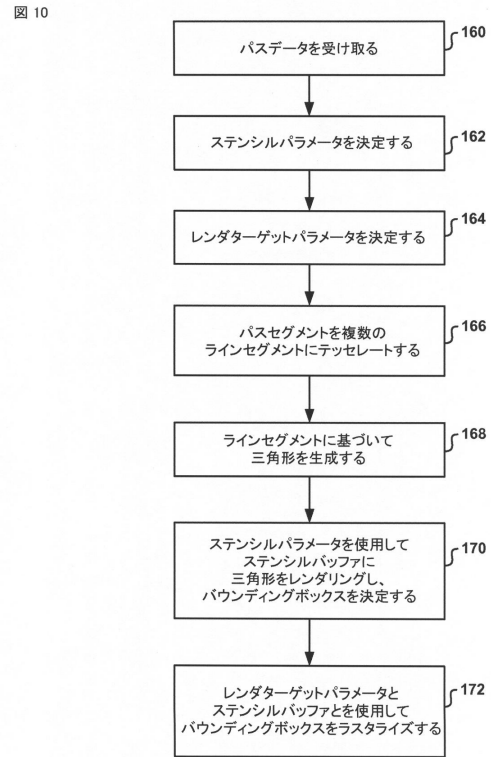


FIG. 10

【図 11】

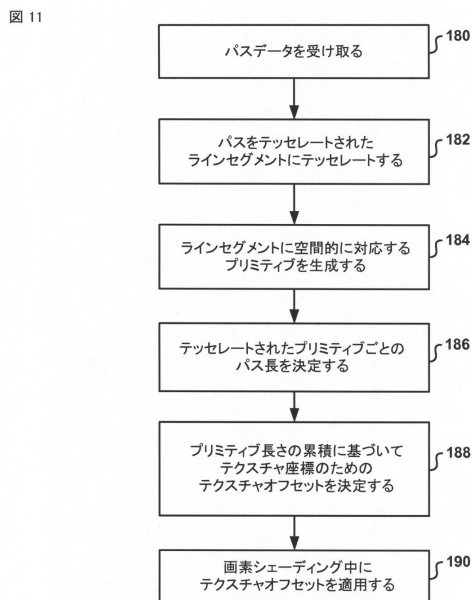


FIG. 11

---

フロントページの続き

(72)発明者 ゴエル、ピネート

アメリカ合衆国、カリフォルニア州 9 2 1 2 1 - 1 7 1 4、サン・ディエゴ、モアハウス・ドライブ 5 7 7 5

(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 9 2 1 2 1 - 1 7 1 4、サン・ディエゴ、モアハウス・ドライブ 5 7 7 5

審査官 真木 健彦

(56)参考文献 特表2010-519618(JP,A)

特表2013-539095(JP,A)

再公表特許第2010/134347(JP,A1)

特表2015-524092(JP,A)

特表2015-528598(JP,A)

特表2015-529859(JP,A)

特表2015-529860(JP,A)

米国特許出願公開第2011/0285724(US,A1)

国際公開第2014/025516(WO,A1)

(58)調査した分野(Int.Cl., DB名)

G 0 6 T 1 1 / 2 0

G 0 6 T 1 5 / 0 0 - 1 5 / 8 7