(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-260537
(P2006-260537A)
(43)公開日　平成18年9月28日(2006.9.28)

(51)Int.Cl.		F I			テーマコード（参考）
G06T 15/70	(2006.01)	G06T	15/70	A	5B050
G06T 15/00	(2006.01)	G06T	15/00	100A	5B080

審査請求 未請求　請求項の数 18　ＯＬ　（全 47 頁）

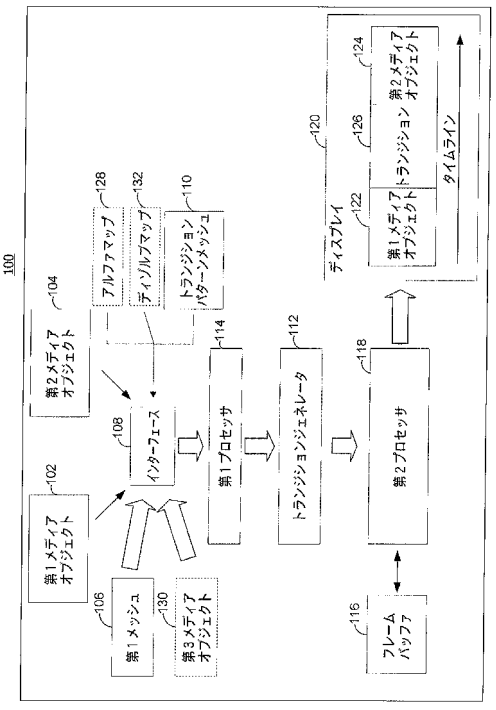
(21)出願番号　特願2006-37064(P2006-37064)
(22)出願日　平成18年2月14日(2006.2.14)
(31)優先権主張番号　11/083,378
(32)優先日　平成17年3月18日(2005.3.18)
(33)優先権主張国　米国(US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ツォン－ジャイ　ワン
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】３Ｄモデルを使用する２Ｄトランジションの生成

(57)【要約】
【課題】第１および第２メディアオブジェクト間の２Ｄトランジションを生成するシステムおよび方法を提供すること。
【解決手段】所望の２Ｄトランジションパターンに対応する３Ｄモデルを定義する。グラフィックス処理ユニット（ＧＰＵ）または命令のセットが、定義された３Ｄモデルの関数として、第１および第２メディアオブジェクト間のトランジションを生成する。トランジションジェネレータが、第１メディアオブジェクトの複数の頂点を第１メッシュに写像して第１の写像されたメディアオブジェクトを作成し、第２メディアオブジェクトの複数の頂点をトランジションパターンメッシュに写像し、第２の写像されたメディアオブジェクトを作成する。ここで、第１メッシュとトランジションパターンメッシュは、３Ｄモデルに関連する。ＧＰＵまたは命令のセットは、第１および第２の写像されたメディアオブジェクトからトランジションをレンダリングする。
【選択図】図１



【特許請求の範囲】
【請求項１】
グラフィックス処理ユニットを使用して第１メディアオブジェクトと第２メディアオブジェクトとの間の２次元トランジションを生成する方法であって、
所望の２次元トランジションパターンに対応する３次元モデルを定義することと、
前記グラフィックス処理ユニットを使用することによって、前記定義された３次元モデルの関数として、前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間のトランジションを生成することと
を含むことを特徴とする方法。
【請求項２】
第１の写像されたメディアオブジェクトを作成するために、２次元座標系での前記第１メディアオブジェクトの複数の頂点を第１メッシュに写像することであって、前記第１メッシュは、前記３次元モデルに関連することと、
第２の写像されたメディアオブジェクトを作成するために、前記第２メディアオブジェクトの複数の頂点をトランジションパターンメッシュに写像することであって、前記トランジションパターンメッシュは、前記３次元モデルに関連することと、
前記グラフィックス処理ユニットを使用することによって、前記第１の写像されたメディアオブジェクトおよび前記第２の写像されたメディアオブジェクトから前記トランジションをレンダリングすることと
をさらに含むことを特徴とする請求項１に記載の方法。
【請求項３】
アルファ値を含むアルファマップを生成することをさらに含み、前記レンダリングすることは、前記アルファマップのランタイム生成の関数として、前記第１の写像されたメディアオブジェクト、前記第２の写像されたメディアオブジェクト、および前記アルファマップから前記トランジションをレンダリングすることを含むことを特徴とする請求項２に記載の方法。
【請求項４】
前記アルファマップの関数として、前記第１メディアオブジェクトと前記第２メディアオブジェクトとが交差するエッジに沿ったピクセルに対応するアルファ値を計算することと、
前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションの前記レンダリングと協調してピクセルに前記計算されたアルファ値を適用することと
をさらに含むことを特徴とする請求項３に記載の方法。
【請求項５】
ディゾルブマップの関数として、前記第１メディアオブジェクトと前記第２メディアオブジェクトとが交差するエッジに沿ったピクセルに対応するディゾルブされたアルファ値を計算することと、
前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションの前記レンダリングと協調してピクセルに前記計算されたディゾルブされたアルファ値を適用することと
をさらに含むことを特徴とする請求項４に記載の方法。
【請求項６】
前記トランジションパターンメッシュは、３次元ボリュームキューブを表し、前記３次元ボリュームキューブは、ｘ座標パラメータ、ｙ座標パラメータ、およびｚ座標パラメータを含むことを特徴とする請求項１に記載の方法。
【請求項７】
第３の写像されたメディアオブジェクトを作成するために、第３メディアオブジェクトの複数の頂点を前記受け取られたトランジションパターンメッシュに写像することと、
前記第１の写像されたメディアオブジェクト、前記第２の写像されたメディアオブジェ

クト、および前記第３の写像されたメディアオブジェクトから前記トランジションをレンダリングすることと
をさらに含むことを特徴とする請求項１に記載の方法。
【請求項８】
１つまたは複数のコンピュータ可読媒体は、請求項１に記載の方法を実行するコンピュータ実行可能命令を有することを特徴とする請求項１に記載の方法。
【請求項９】
第１メディアオブジェクトと第２メディアオブジェクトとの間の２次元トランジションを生成するシステムであって、
前記第１メディアオブジェクト、前記第２メディアオブジェクト、およびトランジションパターンメッシュを受け取るインターフェースと、
第１の写像されたメディアオブジェクトを作成するために、前記受け取られた第１メディアオブジェクトの複数の頂点を第１メッシュに写像するトランジションジェネレータであって、第２の写像されたメディアオブジェクトを作成するために、前記受け取られた第２メディアオブジェクトの複数の頂点を前記トランジションパターンメッシュに写像するトランジションジェネレータと、
前記第１の写像されたメディアオブジェクトおよび前記第２の写像されたメディアオブジェクトから前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションをレンダリングするグラフィックスプロセッサと
を含むことを特徴とするシステム。
【請求項１０】
前記グラフィックスプロセッサは、前記第１の写像されたメディアオブジェクト、前記第２の写像されたメディアオブジェクト、およびアルファマップから前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションをレンダリングし、前記アルファマップは、前記第１の写像されたメディアオブジェクトと前記第２の写像されたメディアオブジェクトとが交差するエッジに沿って位置する複数のピクセルに対応するアルファ値を含むことを特徴とする請求項９に記載のシステム。
【請求項１１】
前記トランジションパターンメッシュは、３次元ボリュームキューブを表し、前記３次元ボリュームキューブは、ｘ座標パラメータ、ｙ座標パラメータ、およびｚ座標パラメータを含むことを特徴とする請求項９に記載のシステム。
【請求項１２】
前記トランジションパターンメッシュのｚ座標パラメータは、レンダリングの進行の関数を表すことを特徴とする請求項９に記載のシステム。
【請求項１３】
前記インターフェースは、第３メディアオブジェクトをさらに受け取り、前記グラフィックスプロセッサは、前記第１の写像されたメディアオブジェクト、前記第２の写像されたメディアオブジェクト、および前記第３メディアオブジェクトから前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションをさらにレンダリングすることを特徴とする請求項９に記載のシステム。
【請求項１４】
第１メディアオブジェクトと第２メディアオブジェクトとの間のトランジションを生成するシステムであって、
前記第１メディアオブジェクトおよび前記第２メディアオブジェクトをレンダリングする第１プロセッサまたは命令の第１セットと、
前記第１メディアオブジェクト、前記第２メディアオブジェクト、およびトランジションパターンメッシュを受け取るインターフェースと、
第１の写像されたメディアオブジェクトを作成するために、前記受け取られた第１メディアオブジェクトの複数の頂点を第１メッシュに写像するトランジションジェネレータであって、第２の写像されたメディアオブジェクトを作成するために、前記受け取られた第

２メディアオブジェクトの複数の頂点を前記受け取られたトランジションパターンメッシュに写像する、トランジションジェネレータと、
前記第１の写像されたメディアオブジェクトおよび前記第２の写像されたメディアオブジェクトから前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションをレンダリングする、前記第１プロセッサまたは命令の第１セットと独立の、第２プロセッサまたは命令の第２セットと
を含むことを特徴とするシステム。
【請求項１５】
前記第２プロセッサまたは命令の前記第２セットは、前記第１の写像されたメディアオブジェクト、前記第２の写像されたメディアオブジェクト、およびアルファマップから前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションをレンダリングし、前記アルファマップは、前記第１の写像されたメディアオブジェクトと前記第２の写像されたメディアオブジェクトとが交差するエッジに沿って位置する複数のピクセルに対応するアルファ値を含むことを特徴とする請求項１４に記載のシステム。
【請求項１６】
頂点データ、テクスチャデータ、またはステンシルデータのうちの少なくとも１つを前記第２プロセッサのために保管するフレームバッファをさらに含むことを特徴とする請求項１４に記載のシステム。
【請求項１７】
前記トランジションパターンメッシュは、３次元ボリュームキューブを表し、前記３次元ボリュームキューブは、ｘ座標パラメータ、ｙ座標パラメータ、およびｚ座標パラメータを含むことを特徴とする請求項１４に記載のシステム。
【請求項１８】
前記インターフェースは、さらに、ディゾルブマップを受け取り、前記第２プロセッサまたは命令の前記第２セットは、さらに、前記第１の写像されたメディアオブジェクト、前記第２の写像されたメディアオブジェクト、および前記ディゾルブマップ内のディゾルブされたアルファ値から、前記第１メディアオブジェクトと前記第２メディアオブジェクトとの間の前記トランジションをレンダリングすることを特徴とする請求項１４に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施形態は、一般的にはグラフィックスアニメーションおよびグラフィックス処理に関する。具体的には、本発明の実施形態は、３次元モデルおよびグラフィックス処理ユニットの処理能力を使用する２次元トランジションの生成に関する。
【背景技術】
【０００２】
通常のグラフィックス処理では、ソフトウェアアプリケーションは、ハードウェアインターフェース（たとえば、グラフィカルデバイスインターフェース（ＧＤＩ））を使用して、ビットマップテキスト（たとえば、ＴｒｕｅＴｙｐｅフォント）、イメージ、および他のグラフィカル要素を表示し、または印刷する。たとえば、ＧＤＩは、適当なディスプレイドライバを呼び出し、描画されるアイテム（ダイアログボックス、ボタン、および他のグラフィカル要素）に関する情報を渡すことによって、スクリーンに一貫したスタイルでアイテムを描画する責任を負う。ＧＤＩを利用するために、ソフトウェアプログラムは、通常、ソフトウェア開発キット（ＳＤＫ）ライブラリで提供されるものなどのルーチンのセットを使用し、ＧＤＩ関数を呼び出すようにこれらのソフトウェアアプリケーションを構成する。たとえば、ＳＤＫライブラリに、アイコン操作、長方形操作、または類似物などの関数が含まれる。
【０００３】

現在のコンピュータシステムおよびコンピュータ構成の下では、ほとんどのＧＤＩ機能が、中央処理ユニット（ＣＰＵ）によって実行される。その結果、他の処理の実行は別として、ＣＰＵは、２Ｄイメージ、ビデオストリーム、または類似物のレンダリングなど、グラフィックス要素およびイメージを処理する必要がある。このような追加のグラフィックス処理は、ＣＰＵおよびコンピュータシステムの全体的なパフォーマンスを妨げる。というのは、グラフィッカル処理が、ピクセルまたはテクスチャの計算、イメージのレンダリングまたはアニメーション、あるいは類似物など、グラフィカルパラメータの広範囲の計算を必要とするからである。これらのグラフィックスまたはイメージのＣＰＵ処理は、妥当であるが、グラフィックス処理またはイメージ処理の改善が望ましい。

【０００４】

さらに、ほとんどの現在のシステムまたはアプリケーションは、２次元アルゴリズムを使用してグラフィックス効果を処理する。たとえば、ユーザが、２つのメディアオブジェクトまたはイメージの間にトランジションを挿入し、２つのメディアオブジェクトの表示の結果のシーケンスが滑らかなトランジションを有するようにすることができる。既存のシステムでは、ＣＰＵが、結果のシーケンスの表示のほかに、アルゴリズムおよびプログラミング命令を実行するのにリソースおよび時間を必要とする。

【０００５】

【非特許文献１】"DirectX(R) Software Development Kit" of MICROSOFT(R) Direct3D(R), available on the MICROSOFT(R) Developer Network (MSDN(R)) subdomain of MICROSOFT(R) web site

【発明の開示】

【発明が解決しようとする課題】

【０００６】

したがって、グラフィックス処理ユニット（ＧＰＵ）の３Ｄモデルおよび固定パイプライン機能を使用して２次元グラフィカル効果を作成することによって、グラフィックス処理タスクをＧＰＵに委任またはシフトするシステムおよび方法が望ましい。あるいは、３Ｄ座標系で２次元効果を処理する命令のセットを使用することによって２次元グラフィックス効果を作成するシステムおよび方法が望まれる。

【課題を解決するための手段】

【０００７】

本発明の実施形態は、３次元座標系を使用して２次元トランジションを実装することによって、従来技術の１つまたは複数の不利益を克服する。２次元トランジションに関する変換されたデータをＣＰＵに送って行列計算、ピクセル計算、またはテクスチャ計算を実行するのではなく、本発明の実施形態は、データをＣＰＵに送らずに、データをＧＰＵに送り、その結果、ＧＰＵが、このデータをより効率的に処理できるようになる。ＧＰＵは、メディアオブジェクト、アニメーション、または他のグラフィカル要素をビデオバッファに送って、表示の準備をすることができる。あるいは、コンピュータ実行可能命令のセットまたはソフトウェアアプリケーションが、３Ｄ座標系でデータを処理して、２次元効果を作成することができる。したがって、本発明の実施形態は、バストラフィック、ＣＰＵ負荷を減らし、トランジション、アニメーション、または他のグラフィカル変換の処理のレンダリング時間を短縮するのを助ける。

【０００８】

さらに、本発明の実施形態は、メディアオブジェクトまたはイメージ間のいくつかのトランジション（たとえば、ディゾルブトランジション、クロスフェードトランジション、または単純なワイプトランジション）の所要ＣＰＵ時間を短縮し、新しいビジュアル変形物およびトランジションパターンの開発をより効率的にする。この設計は、ユーザに、ワイプパターン、レンダリング技法、およびパラメータを混合することによるビジュアル変形物の豊かなコレクションを提供する。本発明の実施形態は、現在市販されているマシンの大部分において、２Ｄグラフィカル効果を作成する際にＧＰＵ固定機能パイプラインも使用して、３次元メディアオブジェクトを実行し、処理する。あるいは、本発明の実施形

態は、３次元メディアオブジェクトを処理し、２次元グラフィカル効果を作るために、命令のセットまたはソフトウェアアプリケーションの固定機能パイプラインを使用する。

【０００９】

本発明の一態様によれば、方法が、グラフィックス処理ユニットを使用して、第１メディアオブジェクトと第２メディアオブジェクトの間の２次元トランジションを生成する。所望の２次元トランジションパターンに対応する３次元モデルが、定義される。第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションは、グラフィックス処理ユニットを使用して、定義された３次元モデルの関数として生成される。

【００１０】

本発明のもう１つの態様によれば、システムが、第１メディアオブジェクトと第２メディアオブジェクトの間の２次元トランジションを生成する。インターフェースが、第１メディアオブジェクト、第２メディアオブジェクト、およびトランジションパターンメッシュを受け取る。トランジションジェネレータが、受け取られた第１メディアオブジェクトの複数の頂点を第１メッシュに写像して、第１の写像されたメディアオブジェクトを作成する。トランジションジェネレータは、受け取られた第２メディアオブジェクトの複数の頂点をトランジションパターンメッシュに写像して、第２の写像されたメディアオブジェクトを作成する。グラフィックスプロセッサが、第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトから、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションをレンダリングする。

【００１１】

本発明のもう１つの態様によれば、システムが、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションを生成する。第１プロセッサまたは命令の第１セットが、第１メディアオブジェクトおよび第２メディアオブジェクトをレンダリングする。インターフェースが、第１メディアオブジェクト、第２メディアオブジェクト、およびトランジションパターンメッシュを受け取る。トランジションジェネレータが、受け取られた第１メディアオブジェクトの複数の頂点を第１メッシュに写像して、第１の写像されたメディアオブジェクトを作成する。トランジションジェネレータは、受け取られた第２メディアオブジェクトの複数の頂点を受け取られたトランジションパターンメッシュに写像して、第２の写像されたメディアオブジェクトを作成する。第１プロセッサまたは命令の第１セットと独立の第２プロセッサまたは命令の第２セットが、第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトから、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションをレンダリングする。

【００１２】

代替案では、本発明は、さまざまな他の方法および装置を含むことができる。

【００１３】

他の特徴は、一部は明白であり、一部は以下で指摘される。

【００１４】

対応する符号は、複数の図面を通じて対応する部分を示す。

【発明を実施するための最良の形態】

【００１５】

まず図１を参照すると、ブロック図に、本発明の一実施形態による第１メディアオブジェクト１０２と第２メディアオブジェクト１０４の間の２次元（２Ｄ）トランジションを生成するシステム１００が示されている。トランジションとは、あるパターンに従う、第１メディアオブジェクトから第２メディアオブジェクトへのディスプレイ上の漸進的な変化である。たとえば、第１メディアオブジェクト１０２が白い色のイメージであり、第２メディアオブジェクト１０４が黒い色のイメージであり、ユーザが、この白い色のイメージから黒い色のイメージへのトランジションを望むと仮定する。トランジションの１実装は、黒い色のイメージが全体的に可視になり、白い色のイメージを完全に置換するまで、黒い色のイメージが左から右へ徐々に白い色のイメージを置換する、左から右へのワイプトランジションとすることができる。そのような左から右へのワイプトランジションに、

垂直の線が白い色のイメージを水平に横切って左から右へ移動し、黒い色のイメージが白い色のイメージを完全に置換するまで垂直の線が左から右に徐々に移動する際に黒い色のイメージを垂直の線の左側に導入することなどのパターンを含めることができる。水平線が縦に移動する、２つの垂直線が中心から移動する、または類似するパターンなどの他のパターンを実装することができる。

【００１６】

システム１００は、図８に示されたシステム１３０、または第１プロセッサ１１４、第２プロセッサ１１８、ディスプレイ１２０、およびストレージ（図示せず）を含む他のコンピュータ化されたシステムとすることができる。もう１つの実施形態で、システム１００に、３次元（３Ｄ）オブジェクトを操作し、表示するアプリケーションプログラムインターフェース（ＡＰＩ）が含まれる。たとえば、そのようなＡＰＩを、Ｍｉｃｒｏｓｏｆｔ Ｃｏｒｐｏｒａｔｉｏｎ社によるＤｉｒｅｃｔ３Ｄ（登録商標）、Ａｐｐｌｅ Ｃｏｍｐｕｔｅｒ，Ｉｎｃ．社によるＱｕｉｃｋＤｒａｗ（登録商標）３Ｄ、またはＳｉｌｉｃｏｎ Ｇｒａｐｈｉｃｓ Ｉｎｃｏｒｐｏｒａｔｅｄ社によるＯｐｅｎＧＬ（登録商標）とすることができる。第１プロセッサ１１４は、ＣＰＵ、他の集中プロセッサまたはメインプロセッサ、コンピュータ実行可能命令の第１セット、または処理用のソフトウェアアプリケーションとすることができる。第２プロセッサ１１８は、ＧＰＵ、他の副プロセッサ、コンピュータ実行可能命令の第２セット、または３Ｄモデルに関するデータを処理するソフトウェアアプリケーションとすることができる。

【００１７】

当初、第１メディアオブジェクト１０２および第２メディアオブジェクト１０４は、ディスプレイ１２０に表示される。たとえば、第１メディアオブジェクト１０２または第２メディアオブジェクト１０４は、ディスプレイ１２０に表示される静止画像、ビデオストリーム、または他のグラフィカル要素（たとえば、アイコン、ボタン、ウェブページ、または類似物）の１つのファイルまたはファイルの集合とすることができる。ユーザが、タイムライン内で第１メディアオブジェクト１０２と第２メディアオブジェクト１０４の間の滑らかなまたは他の興味深いトランジションを作成することを望む場合がある。興味深いトランジションの作成のほかに、ユーザは、見る人が、トランジションおよび／または効果ではなくメディアオブジェクトを介して伝えられるストーリーに焦点を合わせることを望む場合がある。したがって、ユーザが、微妙な効果およびバリエーションを有するトランジションを作成することを望む場合がある。たとえば、ユーザが、トロフィをプレゼントするなど、見る人に興奮を与えるか栄誉を共有することを望むと仮定すると、ユーザは、トロフィ贈呈が明白になるように第１メディアオブジェクト１０２から第２メディアオブジェクト１０４へのトランジションを作成し、そのトロフィがシーンに堂々と登場することを望む場合がある。その一方で、ユーザが、見る人をムービーシーンの終りに移らせるために微妙なトランジション（ソフトエッジ、ディゾルブエッジ、および／またはファンシイエッジ（ｆａｎｃｙ ｅｄｇｅ）トランジションなど）を生成する場合がある。

【００１８】

第１メディアオブジェクト１０２および第２メディアオブジェクト１０４が、インターフェース１０８によって受け取られる。たとえば、インターフェース１０８は、第１メディアオブジェクト１０２および第２メディアオブジェクト１０４を受け取る、ソフトウェアアプリケーションのグラフィカルユーザインターフェース（ＧＵＩ）または他のインターフェース手段とすることができる。第１メディアオブジェクト１０２および第２メディアオブジェクト１０４を、不揮発性ストレージ、揮発性ストレージ、または類似物から、通信ネットワーク（たとえば、システムバス、インターネット、イントラネット、または類似物）などの直接伝送を介してインターフェース１０８に入力することができる。

【００１９】

一例で、ユーザは、第１メディアオブジェクト１０２と第２メディアオブジェクト１０４の間にワイプトランジションを作成することができる。ワイプトランジションは、２Ｄグラフィカル効果（たとえば、第２メディアオブジェクト１０４が、第１メディアオブジ

ェクト１０２を置換するためにコンピュータディスプレイを横切って左から右に移動する）であるが、本発明の実施形態は、任意の所与のトランジションについて３Ｄモデル（たとえば、ｘ－、ｙ－、ｚ－座標系）を定義する。たとえば、図２Ａは、本発明の一実施形態による３Ｄモデル２００を示すブロック図である。モデル２００に、３つの軸すなわちｘ軸、ｙ軸、およびｚ軸を有する３Ｄビュー空間２０４が示されている。もう１つの実施形態で、３Ｄビュー空間２０４が、ディスプレイ１２０の異なるスクリーンアスペクト比（たとえば、４×３または１６×９）に適合するように構成される。モデル２００は、見る人がｚ軸と反対の方向で見る（矢印２１２によって示される）ことを示すために目２０２も示す。言い換えると、ｚ軸は、見る人を指している。

【００２０】

図２Ｂを参照すると、図に、本発明の一実施形態による、２Ｄメディアオブジェクトを３Ｄモデルに写像する正射影変換２１４が示されている。変換２１４は、水平エッジに沿ったｘ軸およびｘ軸に垂直のｙ軸を有する２Ｄ正方形である。一実施形態で、変換２１４の幅および長さに、１単位がセットされ、可視範囲が０から１までになっている。すなわち、変換２１４は、１×１単位の正方形である。他の形状またはポリゴン（たとえば、三角形、長方形、または類似物）を、２Ｄ形状を３Ｄモデルに写像するのに使用できることを理解されたい。ｚ軸は、ｘ－ｙ平面の上に位置する目２１６を指している。図２Ｂに示された例として、目は、位置（０．５，０．５，１．０）にある（たとえば、ｘ－ｙ平面の上から見ている）。言い換えると、２Ｄメディアオブジェクトを変換２１４に写像でき（すなわち、０，０→１．０，１．０空間）、その結果、２Ｄメディアオブジェクトを、３Ｄモデルの３Ｄ座標系に関するデータによって表すことができるようになる。そのようなデータを、ＧＰＵ、コンピュータ実行可能命令のセット、またはソフトウェアアプリケーションによって処理して、２Ｄグラフィカル効果を作成することができる。

【００２１】

図２Ｃは、本発明の一実施形態による、異なる次元でのトランジションパターンのビューの図である。例として、ビュー２０６、２０８、および２１０は、異なるビュー空間でホイールトランジションを示す。透視側面図２０６は、ｘ－ｙ－ｚ座標シンボルによって示される３Ｄモデルのホイールトランジションである。３Ｄ変換を使用する多数の既存システムで、透視図２０６は、人間の目に近いオブジェクトを人間の目から遠いオブジェクトより大きくすることによって、人間の視覚をシミュレートする。その一方で、直交図２０８は、ｘ－ｙ座標シンボルによって示される２Ｄでホイールトランジションを上から眺めた楕円に平坦化する。これは、第１メディアオブジェクト１０２から第２メディアオブジェクト１０４に２Ｄトランジションとしてトランジションする時に表示される、ホイールトランジションの例示的なビューである。その一方で、ｚ軸からのトランジションパターンメッシュビュー２１０は、トランジションパターンメッシュ１１０が適用された後のホイールトランジションのビューである。

【００２２】

トランジションパターンメッシュビュー２１０をさらに説明するために、図３Ａ～Ｆは、本発明の一実施形態による、第１メディアオブジェクト３０４を、第１メディアオブジェクト３０４の中心から第２メディアオブジェクト３０６に置換するワイプトランジションなど、トランジションパターンメッシュ１１０の実装を示す図である。たとえば、図３Ａに示されているように、第１メディアオブジェクト３０４は、線を書き込まれたイメージであり、第２メディアオブジェクト３０６は、白い色のイメージである。ユーザは、目３０２によって表される方向から見ているが、第２メディアオブジェクト３０６を中心から第１メディアオブジェクト３０４に導入することを望む。当初、ユーザは、図３Ｄの３０８によって示されているように、第１メディアオブジェクト３０４だけを見る。図３Ｂでは、ワイプトランジションパターンメッシュが、第１メディアオブジェクト３０４に写像されて、第１の写像されたメディアオブジェクト３０４’が形成される。たとえば、第１メディアオブジェクト３０４の各頂点またはピクセル点を、３Ｄであるトランジションパターンメッシュに変換するか登録し、その結果、３Ｄのトランジションパターンメッシ

ュ上の各ピクセル点が、２Ｄの第１メディアオブジェクト３０４の対応する頂点またはピクセル点と一致するようになる。一実施形態で、第２メディアオブジェクト３０６が、第１メッシュ１０６に写像されて、第２の写像されたメディアオブジェクト３０６’が形成されるが、第１メッシュ１０６は、以下で説明する。

【００２３】

まだ図３Ｂを参照すると、ワイプトランジションが進行する時に、第１の写像されたメディアオブジェクト３０４’が、第２メディアオブジェクト３０６’に向かって移動し、ユーザが、図３Ｅの３０８’によって示されるものを見るようになる。言い換えると、ユーザは、第１メディアオブジェクト３０４から第２メディアオブジェクト３０６へのトランジションを見始める。図３Ｃでは、トランジション進行が完了した時に、ユーザは、図３Ｆの３０８”によって示されるように、結果の第２メディアオブジェクト３０６”だけを見、第１の写像されたメディアオブジェクト３０４”は、ユーザには見えない。

【００２４】

一実施形態で、トランジション進行中に、第２メディアオブジェクト３０６’が、第１の写像されたメディアオブジェクト３０４’に向かって移動する。もう１つの実施形態で、第１の写像されたメディアオブジェクト３０４’および第２メディアオブジェクト３０６’が、トランジション進行中にお互いに向かって移動する。したがって、３Ｄのトランジションパターンメッシュを使用することによって、当業者に既知であるが、２Ｄトランジションが、ＣＰＵではなくＧＰＵまたはソフトウェアアプリケーションによって処理され、その結果、これらのトランジションが、より効率的に処理されるようになる。

【００２５】

図１に戻って、インターフェース１０８は、第１メッシュ１０６およびトランジションパターンメッシュ１１０も受け取るが、この第１メッシュ１０６およびトランジションパターンメッシュ１１０に対して、メディアオブジェクトが写像され、その結果、メディアオブジェクトの間のトランジションをレンダリングできるようになる。たとえば、第１メッシュ１０６は、第１の写像されたメディアオブジェクトを形成するために第１メディアオブジェクト１０２の頂点を写像するフルスクリーン長方形メッシュとすることができる。言い換えると、第１メディアオブジェクト１０２の各頂点は、３Ｄ座標で第１メッシュ１０６に写像され、ＧＰＵまたはソフトウェアアプリケーションが、表示のために各頂点に対応するピクセル値を処理し、または計算することができるようになる。図２Ｂをもう１つの例として使用すると、変換２１４は、１単位の幅および１単位の長さを有し、１×１正方形が作成される。変換２１４の２Ｄ座標系での４つの角のそれぞれは、図２Ｂで示すように（０，０）、（１，０）、（１，１）、および（０，１）として表すことができる。第１メッシュ１０６に第１メディアオブジェクト１０２を写像する際に、第１メディアオブジェクト１０２の各ピクセル点または頂点が、変換２１４に登録されるか変換される。また、変換のｚ座標に０をセットして、平らな３Ｄオブジェクトを表すことができる。

【００２６】

一実施形態で、第１メッシュ１０６に、テクスチャ座標に関する第１メディアオブジェクト１０２の頂点データ（たとえば、３Ｄのテクスチャ空間での個々のカラー値に関するテクスチャ値）が含まれる。もう１つの実施形態で、各頂点のテクスチャ座標値が、ｘ、ｙ座標の値と等しい、すなわち、（ｔｕ，ｔｖ）＝（ｘ，ｙ）であり、テクスチャの４つの角が、第１メッシュ１０６の４つの角に写像される。もう１つの実施形態で、グーローシェーディングモデルが使用され、テクスチャ値は、第１メッシュ１０６を塗り潰すのに使用される。付録Ａのセクション２で、第１メッシュ１０６を作成する例示的実装を説明する。第１メッシュおよび／またはトランジションパターンメッシュ実装の詳細について、非特許文献１が、参照によって本明細書に組み込まれている。

【００２７】

もう１つの実施形態で、第１メッシュ１０６が、頂点のセットおよびポリゴン面のセットを定義する。第１メッシュ１０６および第１メディアオブジェクト１０２を受け取った

後に、トランジションジェネレータ１１２が、第１メディアオブジェクト１０２を第１メッシュ１０６に写像して、第１の写像されたメディアオブジェクトを形成する。また、トランジションパターンメッシュ１１０に、第１メディアオブジェクト１０２から第２メディアオブジェクト１０４へまたはその逆へのトランジションのためにユーザが選択するトランジションパターンが含まれる。たとえば、インターフェース１０８は、さまざまな形状を有するワイプトランジションまたは類似物などのさまざまなトランジションの例をディスプレイ１２０に表示し、ユーザは、入力デバイス（たとえば、マウスまたはキーボード）を使用して、所望のトランジションパターンを選択することができる。

【００２８】

図４Ａ～Ｃは、本発明の一実施形態による３Ｄモデルを使用するサークルアウト（ｃｉｒｃｌｅ　ｏｕｔ）およびサークルイン（ｃｉｒｃｌｅ　ｉｎ）のワイプトランジションパターンの実装を示す図である。まず、図４Ａに、トランジションパターンとしてサークルアウトワイプトランジションの透視側面図を示す。入ってくるメディアオブジェクトまたは第１メディアオブジェクト４０２（サークルトランジションパターンによって写像された後の）が、出てゆくメディアオブジェクトまたは第２メディアオブジェクト４１８にトランジションすることが示されている。たとえば、始め（またはｚ座標値が０に近い時）に、入ってくるメディアオブジェクトまたは第１メディアオブジェクト４０２は、出てゆくメディアオブジェクトまたは第２メディアオブジェクト４１８上の小さい円に見える。このサークルアウトワイプトランジションでは、ユーザは、円が画面またはディスプレイ１２０の中心から現れるのを見る。トランジションが進行するにつれて、円が大きくなる。入ってくるメディアオブジェクトまたは第１メディアオブジェクト４０２は、円の中に現れ、出てゆくメディアオブジェクトまたは第２メディアオブジェクト４１８は、円が大きくなるにつれて入ってくるメディアオブジェクトまたは第１メディアオブジェクト４０２によって置換される。したがって、入ってくるメディアオブジェクトまたは第１メディアオブジェクト４０２のｚ座標は、トランジションの進行を表す。ｚ座標値が増えるにつれて、点または小さい円が、その半径を小から大に増やし、出てゆくメディアオブジェクトまたは第２メディアオブジェクト４１８が、入ってくるメディアオブジェクトまたは第１メディアオブジェクト４０２にトランジションされる。一実施形態で、ビューイング空間が、見る人のディスプレイ（たとえば、ディスプレイ１２０）に制限され、したがって、サークルトランジションの諸部分が、ディスプレイ１２０によって表示できない時にクロッピングされる場合がある。

【００２９】

同様に、もう１つの透視側面図に、入ってくるメディアオブジェクトまたは第２メディアオブジェクト４０４と出てゆくメディアオブジェクトまたは第１メディアオブジェクト４２２のサークルインワイプトランジションが示されている。始め（またはｚ座標値が０に近い時）に点から始めるのではなく、サークルインワイプトランジションは、大きい円から始まり、入ってくるメディアオブジェクトまたは第２メディアオブジェクト４０４が出てゆくメディアオブジェクトまたは第１メディアオブジェクト４２２にトランジションする時に点になる。

【００３０】

さらに例示するために、図４Ｂに、サークルワイプトランジションの３つのビューを示す。スクリーンショット４０６に、サークルアウトワイプを用いて第２メディアオブジェクトＢにトランジションしている第１メディアオブジェクトＡのスナップショットを示す。直交図４０８に、２Ｄビューのサークルを示し、透視側面図４１０に、円錐形のサークルを示す。したがって、本発明の実施形態は、モデル２００などの３Ｄモデルへのサークル形状のトランジションを適用し、その結果、２Ｄサークルが、円錐形上として変換される。３Ｄモデルでトランジションパターンメッシュ１１０を使用することによって、本発明の実施形態は、２Ｄトランジションの３Ｄモデルに関するデータを第２プロセッサまたは命令の第２セット１１８（集合的に「第２プロセッサ１１８」と称する）に送り、その結果、２Ｄトランジションをより効率的に処理できるようになる。

【００３１】
図４Ｃに、本発明の一実施形態によるサークルアウトトランジションおよびサークルイントランジションの進行のもう１つの例を示す。たとえば、ブロック４１２に、出てゆくメディアオブジェクトとしての第１メディアオブジェクト４１４が、サークルアウトワイプトランジションを介して入ってくるメディアオブジェクトとしての第２メディアオブジェクト４１６にトランジションするのを示す。トランジション進行０％で、ブロック４１２に、第１メディアオブジェクト４１４だけが示されている。進行２５％で、サークルアウトトランジションが第２メディアオブジェクト４１６を導入するので、第２メディアオブジェクト４１６が現れ始める。ブロック４１２に、進行が１００％まで進んだ時に、２Ｄサークルアウトワイプトランジションがどのように表示されるかが示されている。同様に、ブロック４２０に、出てゆくメディアオブジェクトとしての第１メディアオブジェクト４１４が、サークルインワイプトランジションを介して第２メディアオブジェクト４１６にトランジションするのが示されている。
【００３２】
異なるパターンを有するワイプトランジションのユーザ体験が、当技術分野で既知である可能性があるが、ＧＰＵまたはソフトウェアアプリケーションを用いる３Ｄトランジションパターンを使用する、２Ｄトランジションの処理は、より効率的であり、より少ないＣＰＵ処理を使用することを理解されたい。
【００３３】
図４Ａから４Ｃに示されたトランジションパターン（たとえば、サークルインワイプトランジションおよびサークルアウトワイプトランジション）が、例として示され、制限でないことも理解されたい。たとえば、スワイプ（ｓｗｉｐｅ）、ダイアゴナルダウン（ｄｉａｇｏｎａｌ　ｄｏｗｎ）、スター（ｓｔａｒ）、ホイール（ｗｈｅｅｌ）、ボウタイ（ｂｏｗ　ｔｉｅ）、サークル、長方形、スプリット（ｓｐｌｉｔ）、リビールアンドインサート（ｒｅｖｅａｌ　ａｎｄ　ｉｎｓｅｔ）、アイリス（ｉｒｉｓ）、フィルドＶ（ｆｉｌｌｅｄ　Ｖ）、ジグザグ、チェッカーボード、ダイアゴナルボックス（ｄｉａｇｏｎａｌ　ｂｏｘ）、ダイアゴナルクロス（ｄｉａｇｏｎａｌ　ｃｒｏｓｓ）、ダイヤモンド、レーダー（ｒａｄａｒ）、スターズ（ｓｔａｒｓ）、サークルズ（ｃｉｒｃｌｅｓ）、キーホール（ｋｅｙｈｏｌｅ）、ハート（ｈｅａｒｔ）、スイープ（ｓｗｅｅｐ）（－イン、－アップ、－アウト、－ダウン）、ファン（ｆａｎ）（－アップ、－アウト、－ダウン、－イン）、アイ（ｅｙｅ）、または他のトランジションなどの、他のトランジションパターンを、本発明の趣旨から逸脱することなく実装することができる。追加の例として、本発明の一実施形態による例示的なトランジションパターンの実装を示す図５Ａおよび５Ｂを参照されたい。図５Ａに、３０％進行時の長方形トランジションを示す。図５Ａに、直交上面図および透視側面図も示されている。同様に、図５Ｂに、３０％進行時のジグザグトランジション、そのトランジションパターンの直交上面図および透視側面図を示す。さらに、付録Ａのセクション１で、３Ｄモデルを使用するトランジションパターン作成の実装を説明する。
【００３４】
第１の写像されたメディアオブジェクトの形成に似て、トランジションジェネレータ１１２は、第２メディアオブジェクト１０４およびトランジションパターンメッシュ１１０をインターフェース１０８から受け取った後に、トランジションパターンメッシュ１１０を第２メディアオブジェクト１０４の頂点に写像して、第２の写像されたメディアオブジェクトを作成する。第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトは、第１の写像されたメディアオブジェクト１２２および第２の写像されたメディアオブジェクト１２４から第１メディアオブジェクト１０２と第２メディアオブジェクト１０４の間のトランジション１２６をディスプレイ１２０にレンダリングするために、第２プロセッサ１１８に送られる。たとえば、図４Ｃのブロック４１２に、第２プロセッサまたは命令の第２セット１１８を使用する、第１の写像されたメディアオブジェクト４１４および第２の写像されたメディアオブジェクト４１６からのサークルアウト

トランジションのレンダリングの結果が示されている。

【００３５】

一実施形態で、システム１００に、トランジション１２６をレンダリングするために、第２プロセッサ１１８のための頂点、テクスチャ、ステンシル、または他のデータを一時的に保管するフレームバッファ１１６も含まれる。たとえば、第２プロセッサ１１８は、次のレンダリングシーケンスでワイプトランジションを処理することができる：（１）必要な場合にマスクにトランジションパターンメッシュをレンダリングし、（２）第１メッシュ１０６をフレームバッファ１１６にレンダリングし、（３）トランジションパターンメッシュ１１０をフレームバッファ１１６にレンダリングする。たとえば、各ステップで、第２プロセッサ１１８は、シーケンスのレンダリング状態を維持し、その結果、次のメディアオブジェクトまたはトランジションを表示する前に、システム１００が、表示されるメディアオブジェクトまたはトランジションに正しいレンダリング状態を適用できるようにする。もう１つの例で、フレームバッファ１１６を、揮発性ストレージ、不揮発性ストレージ、または他の記憶媒体内のビデオメモリとすることができる。

【００３６】

もう１つの実施形態で、第２プロセッサ１１８は、一時オブジェクトを作成して、トランジション１２６のレンダリングを助ける。たとえば、トランジション（たとえば、ワイプトランジション）のレンダリング中に、第２プロセッサ１１８は、一時オブジェクトを作成して、アクセラレーテッドラスタライゼーションおよびハードウェア／ソフトウェア頂点処理などの追加のハードウェア機能およびソフトウェア機能にアクセスすることができる。もう１つの実施形態で、第２プロセッサ１１８は、追加の一時オブジェクトを作成して、第１メッシュ、トランジションパターンメッシュ、アルファブレンディング、およびディゾルブブレンディング（以下で説明する）を処理する。たとえば、第２プロセッサ１１８は、一時オブジェクトを作成して、第１メッシュ１０６、トランジションパターンメッシュ１１０、アルファブレンディング、およびディゾルブブレンディングを作成し、またはロードし、その結果、これらのオブジェクトが、ハードウェアまたはソフトウェアから特定の機能を呼び出すかまたは実行できるようにする。これらの一時オブジェクトは、第２プロセッサ１１８がトランジション１２６または他の処理をレンダリングした後に、ストレージから除去または破棄される。

【００３７】

一実施形態で、本発明の実施形態は、トランジションと共に追加の効果を作成する。たとえば、図３Ｂ～Ｃおよび図４Ａ～Ｂに示されたトランジションは、トランジションパターンが第１メディアオブジェクトから第２メディアオブジェクトに変化する時に、トランジションパターンの鋭いエッジを示す。図６は、本発明の一実施形態による、トランジションパターンの鋭いエッジを有する、第１メディアオブジェクトと第２メディアオブジェクトの間の２Ｄトランジションを生成する方法を示す流れ図である。まず、６０２で、システム１００が、第１メッシュ（たとえば、第１メッシュ１０６）およびトランジションパターンメッシュ（たとえば、トランジションパターンメッシュ１１０）をロードする。たとえば、インターフェース１０８は、ユーザから、ソフトウェアアプリケーションによって供給されるかまたは他のソースから供給されるトランジションパターンのコレクションから選択した後に、トランジションパターンメッシュ（たとえば、サークルイン、ジグザグ、または類似物）を受け取る。６０４で、システム１００は、選択された２Ｄトランジションを生成するための３Ｄモデルを定義する。たとえば、モデル２００および直交変換２１４を使用することによって、システム１００は、定義される変換２１４として、０．０から１．０に進行するアニメーションループを定義する。

【００３８】

６０６で、インターフェース１０８が、出てゆくメディアオブジェクトとして第１メディアオブジェクト（たとえば、第１メディアオブジェクト３０４）を受け取る。６０８で、トランジションジェネレータ１１２が、この第１メディアオブジェクトを第１メッシュに写像して、第１の写像されたメディアオブジェクトを形成する。６１０で、インターフ

ェース１０８が、第２メディアオブジェクト（たとえば、第２メディアオブジェクト３０６）を受け取り、６１２で、トランジションジェネレータ１１２が、第２メディアオブジェクトを、インターフェース１０８からの、選択された／受け取られたトランジションパターンメッシュに写像する（写像の実装の例については図３Ａ～Ｆの説明を参照されたい）。次に、６１４で、第２プロセッサ１１８が、第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトから、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジション（たとえば、トランジション１２６）をレンダリングする。たとえば、レンダリングにおいて、第２プロセッサ１１８は、まず、トランジションを行う前に、第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトの頂点、ピクセル、および／またはテクスチャに関するデータを計算する。第２プロセッサ１１８は、次に、進行の関数としてすなわち、第１メディアオブジェクト１０２から第２メディアオブジェクト１０４への変化でトランジション１２６が進行する際に、データを計算する。データの計算を完了した時に、第２プロセッサ１１８は、計算されたデータを、ディスプレイ１２０に表示されるフォーマットに変換する。たとえば、ディスプレイ１２０が、アスペクト比４×３を有するコンピュータモニタである場合があり、第２プロセッサ１１８は、アスペクト比４×３を有するディスプレイ１２０でトランジション１２６を表示するのに適するように、計算されたデータを変換することができる。その結果、レンダリングの後に、ディスプレイ１２０に、第１メディアオブジェクト１０２から第２メディアオブジェクト１０４にトランジションするトランジション１２６が表示される。鋭いエッジを作成する１つの実装を説明した付録Ａのセクション３を参照されたい。

【００３９】

上で述べたように、本発明の実施形態は、３Ｄモデルを使用してトランジションを生成することに加えて、メディアオブジェクトの間でのトランジションに対する追加効果を作成する。たとえば、本発明の実施形態は、トランジション中に第１メディアオブジェクトが第２メディアオブジェクトと交差する時に、エッジを柔らかくすることができる。

【００４０】

前の技法による固いエッジまたは鋭いエッジを柔らかくするために、アルファブレンディング技法を使用することができる。たとえば、インターフェース１０８が、トランジションパターンの鋭いエッジを柔らかくするアルファマップ１２８を受け取る。一実施形態で、第２プロセッサ１１８は、プリミティブごとの（材料アルファ）パラメータ、頂点ごとの（頂点アルファ）パラメータ、ピクセルごとのアルファ（テクスチャアルファ、フレームバッファアルファ、およびレンダターゲットアルファ）パラメータ、または類似するアルファブレンディングパラメータなどのパラメータを受け取り、処理する。アルファブレンディング技法を介して、アルファ値は、第１メディアオブジェクトと第２メディアオブジェクトが交差するエッジの周囲で徐々に減少する。一実施形態で、アルファマップは、アルファ値を保管するのに使用される。

【００４１】

図９および図１０は、本発明の一実施形態によるアルファマップ９００の使用の例を示す図である。図９Ａのオブジェクト９０４に、３０％トランジション進行時のダイアゴナルクロストランジションが示され、図９Ｂのオブジェクト９０６に、７０％トランジション進行時の同一のダイアゴナルクロストランジションが示されている。３０％進行時に、線９０２に沿ったアルファ値が、トランジション進行を定義するために選択される。たとえば、点９２２のアルファ値は、図９Ａのオブジェクト９０４内の２つの対応する点９１２および９１４のトランジション進行値を定義する。同様に、点９５０のアルファ値は、図９Ａのオブジェクト９０４内の対応する点９０８のトランジション進行値を定義する。アルファマップ９００内のアルファ値を使用して、第１メディアオブジェクト１０２および第２メディアオブジェクト１０４のレンダリング中の１つまたは複数の対応する点を表すことができることを理解されたい。また、図９Ａ～Ｃが、白黒の陰影を用いて示されているが、他のカラーシェーディングおよび／またはカラーテクスチャの度合いを、本発明

の範囲から外れずに使用できることを理解されたい。

【００４２】

７０％進行時に、線９１０に沿ったアルファ値が、トランジション進行を定義するために選択される。たとえば、点９２４のアルファ値は、図９Ｂのオブジェクト９０６内の２つの対応する点９１６および９１８のトランジション進行値を定義する。同様に、点９５２のアルファ値は、図９Ｂのオブジェクト９０６内の対応する点９２０のトランジション進行値を定義する。トランジションが進行するにつれて、トランジション進行を示す別の線が、アルファマップ９００からアルファ値が選択される場所を示すことを理解されたい。

【００４３】

図１０に、本発明の一実施形態によるアルファマップ９２６の勾配半透明領域を示す。アルファマップ９２６に、３つの領域すなわち、不透明領域９３８、透明領域９４０、および勾配半透明領域（破線の箱によって定義される）が含まれ、この勾配半透明領域では、第１メディアオブジェクト１０２および第２メディアオブジェクト１０４が一緒にブレンドされる。アルファマップ９２６では、点９２８のアルファ値に、１００％白および０％黒のカラーシェーディングに対応する値が含まれ、点９３２のアルファ値に、０％白および１００％黒のカラーシェーディングに対応する値が含まれる。アルファマップ９２６を分割する対角線９４４は、白の５０％シェーディングおよび黒の５０％シェーディングを示す。５％幅領域（破線によって囲まれた）は、５０％白および５０％黒のカラーシェーディングに対応するアルファ値を選択できる領域を示す。たとえば、領域９３０および９４２のアルファ値は、白の５０％に対応する値を有することができる。一実施形態で、勾配半透明領域の幅は、０％から５０％までの範囲内とすることができる。たとえば、勾配半透明領域の０％幅には、勾配半透明領域がなく、その結果、アルファマップ９２６内のアルファ値に、１００％白または０％白に対応する値のいずれかが含まれる。その代わりに、５０％幅では、アルファマップ９２６内のアルファ値のすべてに、５０％白に対応する値が含まれる。アルファマップ９２６の勾配半透明領域の幅を変更することを定義することによって、本発明の実施形態は、アルファマップ９２６内のアルファ値を選択して、２次元トランジションのソフトエッジ効果を作ることができる。付録Ａのセクション５で、アルファマップの例示的実装を説明する。

【００４４】

図７は、本発明の一実施形態による、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションの一部である追加効果を作成する方法を示す流れ図である。図７で、アルファブレンディング技法の使用に関する本発明の代替実施形態を示す。ステップ６０２から６１２が、図７にも共通することを理解されたい。代替実施形態では、７１４で、アルファマップを受け取る。その結果、第２プロセッサ１１８が、第１の写像されたメディアオブジェクト、第２の写像されたメディアオブジェクト、およびアルファマップから、第１メディアオブジェクト１０２と第２メディアオブジェクト１０４の間のトランジション１２６をレンダリングする。この場合に、第２プロセッサ１１８は、上で説明したように第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトをレンダリングすることに加えて、アルファマップをレンダリングする。図４Ｂのスクリーンショット４０６を例として使用すると、第１の写像されたメディアオブジェクトのテクスチャまたはシェーディングは、第２の写像されたメディアオブジェクトのテクスチャまたはシェーディングと別個である。アルファマップと共に第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトをレンダリングする際に、第２プロセッサ１１８は、アルファマップの関数として、第１メディアオブジェクトおよび第２メディアオブジェクトが交差するエッジに沿ったピクセルに対応するアルファ値を計算することによって、第１および第２の写像されたメディアオブジェクトが交差するエッジを柔らかくする。第２プロセッサ１１８が、進行の関数として第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクト１２４からデータを計算する時に、計算されたアルファ値が適用され、交差のエッジに沿ったピクセルを置

換する。第２プロセッサ１１８は、ピクセルの適用を完了した時に、第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトから計算されたデータに加えて、表示のために、ピクセルをディスプレイ１２０に変換する。付録Ａのセクション４で、第１メディアオブジェクトと第２メディアオブジェクトの交差の間のエッジを柔らかくすることの実装を説明する。

【００４５】

もう１つの代替実施形態では、ディゾルブブレンディング技法を、第１メディアオブジェクトと第２メディアオブジェクトの間で交差するエッジに適用する。たとえば、この技法は、第２プロセッサ１１８による２つのレンダリングプロセスを必要とする。第２プロセッサは、まず、トランジションパターンメッシュ１１０をレンダリングして、ディゾルブされたアルファ値を有するディゾルブマップを生成する。第２のレンダリングプロセスで、第２プロセッサ１１８は、ディゾルブマップと共に第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトをレンダリングする。言い換えると、アルファマップを用いるレンダリングに似て、第２プロセッサ１１８は、第１の写像されたメディアオブジェクトおよび第２の写像されたメディアオブジェクトと共にレンダリングされる、ディゾルブされたアルファ値を有する独立したディゾルブマップを生成する。

【００４６】

もう１つの実施形態では、アルファブレンディング技法およびディゾルブブレンディング技法の両方を使用することができる。たとえば、アルファマップからのアルファ値を、アルファブレンディング技法と同様に取り出す。次に、ディゾルブされたアルファ値を有し、アルファマップからのアルファ値を用いて計算されたディゾルブマップを作成して、ディゾルブブレンディング効果を作る。たとえば、この計算は、ＡＤＤＳＩＧＮＥＤ動作を適用し（たとえば、ＡＤＤＳＩＧＮＥＤは、Ａｌｐｈａ　１（アルファマップから）＋Ａｌｐｈａ　２（ディゾルブマップから）－０．５を計算する）、結果を０と１の間に制限することによって行われる。もう１つの実施形態で、０．２５を超える計算されたアルファ値を有するピクセルだけが、トランジションの一部として表示される。この例では、これらのピクセルは表示されないが、計算されたアルファ値は、他の使用のために保管される（たとえば、ステンシルバッファにこれらを保管する）。付録Ａのセクション８で、エッジをディゾルブする技法の実装を説明する。

【００４７】

一つの代替実施形態で、７１８で、インターフェース１０８が第３メディアオブジェクトを受け取る。たとえば、アルファブレンディング技法、ディゾルブブレンディング技法、またはソフトエッジ技法のほかに、ユーザが、第１メディアオブジェクトおよび／または第２メディアオブジェクトのテクスチャまたはシェーディングと異なるテクスチャデータまたはシェーディングデータからなるエッジを作成することができる。したがって、７２０で、トランジションジェネレータ１１２が、第３メディアオブジェクトをトランジションパターンメッシュに写像して、第３の写像されたメディアオブジェクトを形成する。７１６で、第２プロセッサ１１８が、第１の写像されたメディアオブジェクト、第２の写像されたメディアオブジェクト、および第３の写像されたメディアオブジェクトから、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションをレンダリングする。付録Ａのセクション６で、第３メディアオブジェクトを使用するトランジション作成の実装を説明する。

【００４８】

もう１つの実施形態で、第２プロセッサ１１８は、アルファブレンディング技法、ディゾルブブレンディング技法、ソフトエッジ技法、シャープエッジ技法、第３メディアオブジェクトブレンディング技法、または上で説明した技法の組合せなどの技法を組み込んだトランジションをレンダリングする。たとえば、ユーザは、第３メディアオブジェクトブレンディング技法およびシャープエッジ技法の組合せによるトランジションを作成して、どのピクセルが結果的に表示されるかを制御することができる。この例では、トランジシ

ョンジェネレータ１１２が、第２メディアオブジェクトを写像して第２の写像されたメディアオブジェクトを形成することに加えて、第３メディアオブジェクトを写像して第３の写像されたメディアオブジェクトを形成する。次に、トランジションジェネレータ１１２は、第２の写像されたメディアオブジェクトを、第３の写像されたメディアオブジェクトの背後に置く（たとえば、現在のアニメーション進行の量に伴うｚ座標方向と反対の方向に）。すなわち、第２の写像されたメディアオブジェクトの頂点値またはピクセル値が、第３の写像されたメディアオブジェクトの頂点値またはピクセル値の背後に写像され、その結果、ユーザは、第３の写像されたメディアオブジェクトのピクセルしか見られない。次に、トランジションジェネレータ１１２は、第３の写像されたメディアオブジェクトを使用して、第１メディアオブジェクトと第２メディアオブジェクトが交差する場所のピクセルを置換する。言い換えると、前に説明したアルファブレンディング技法またはディゾルブブレンディング技法と異なって、第３メディアオブジェクトブレンディング技法は、第３メディアオブジェクトを使用して、第１メディアオブジェクト１０２と第２メディアオブジェクト１０４が交差するエッジに沿ったピクセルのテクスチャ値を置換する。したがって、第２プロセッサ１１８が、第３メディアオブジェクトを使用して第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションをレンダリングする時に、ユーザは、ディスプレイ１２０で、第１メディアオブジェクトと第２メディアオブジェクトが交差するエッジに沿って第３メディアオブジェクトを見る。一実施形態で、第３の写像されたメディアオブジェクトの幅は、始めは狭く、トランジション１２６のアニメーションが進行するにつれて広くなる。付録Ａのセクション７で、上で説明したトランジション効果の実装を説明する。

【００４９】

上で説明した技法の組合せの使用のもう１つの例では、ソフトエッジ技法と第３メディアオブジェクトブレンディング技法の組合せを使用することができる。一実施形態で、第２プロセッサ１１８が、まず第２の写像されたメディアオブジェクトをレンダリングし、次に第３の写像されたメディアオブジェクトをレンダリングする、またはその逆を行う。アルファマップのアルファ値が、第３の写像されたメディアオブジェクトが第１のメディアオブジェクトおよび／または第２メディアオブジェクトと交差するエッジのピクセルをブレンディングするのに使用される。

【００５０】

図８に、コンピュータ１３０の形の汎用コンピューティングデバイスの一例を示す。本発明の一実施形態では、コンピュータ１３０などのコンピュータが、本明細書で示され、説明された他の図での使用に適する。コンピュータ１３０は、１つまたは複数のプロセッサまたは処理ユニット１３２およびシステムメモリ１３４を有する。図示の実施形態では、システムバス１３６が、システムメモリ１３４を含むさまざまなシステムコンポーネントをプロセッサ１３２に結合する。バス１３６は、メモリバスまたはメモリコントローラ、周辺バス、ＡＧＰ（ａｃｃｅｌｅｒａｔｅｄ ｇｒａｐｈｉｃｓ ｐｏｒｔ）、およびさまざまなバスアーキテクチャのいずれかを使用するプロセッサバスまたはローカルバスを含む複数のタイプのバス構造のいずれかのうちの１つまたは複数とすることができる。制限ではなく例として、そのようなアーキテクチャに、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｈｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、およびメザニンバスとも称するＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バスが含まれる。

【００５１】

コンピュータ１３０に、通常は、少なくとも１つの形のコンピュータ可読媒体が含まれる。揮発性媒体および不揮発性媒体、リムーバブル媒体およびノンリムーバブル媒体の両方を含むコンピュータ可読媒体は、コンピュータ１３０によってアクセスできる使用可能な媒体のいずれかとすることができる。制限ではなく例として、コンピュータ可読媒体に

、コンピュータ記憶媒体および通信媒体が含まれる。コンピュータ記憶媒体に、コンピュータ可読命令、データ構造、プログラムモジュール、または他のデータなどの情報の保管のための方法またはテクノロジのいずれかで実装された、揮発性および不揮発性、リムーバブルおよびノンリムーバブルの媒体が含まれる。たとえば、コンピュータ記憶媒体に、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリ、または他のメモリテクノロジ、ＣＤ－ＲＯＭ、ディジタル多用途ディスク（ＤＶＤ）、または他の光学ディスクストレージ、磁気カセット、磁気テープ、磁気ディスクストレージ、または他の磁気ストレージデバイス、あるいは所望の情報の保管に使用でき、コンピュータ１３０によってアクセスできる他のすべての媒体が含まれる。通信媒体は、通常は、コンピュータ可読命令、データ構造、プログラムモジュール、または他のデータを、搬送波または他のトランスポート機構などの変調されたデータ信号内に具体化し、通信媒体には、すべての情報配布媒体が含まれる。当業者は、変調されたデータ信号に精通しているが、この変調されたデータ信号は、信号内に情報をエンコードする形でその特性の１つまたは複数が設定または変更されている。有線ネットワークまたは直接配線接続などの有線媒体と、音響、ＲＦ、赤外線、および他の無線媒体などの無線媒体が、通信媒体の例である。上記のいずれかの組合せも、コンピュータ可読媒体の範囲に含まれる。

【００５２】

システムメモリ１３４に、リムーバブルおよび／またはノンリムーバブルで揮発性および／または不揮発性のメモリの形のコンピュータ記憶媒体が含まれる。図示の実施形態では、システムメモリ１３４に、読取専用メモリ（ＲＯＭ）１３８およびランダムアクセスメモリ（ＲＡＭ）１４０が含まれる。起動中などにコンピュータ１３０内の要素の間での情報の転送を助ける基本ルーチンを含む基本入出力システム１４２（ＢＩＯＳ）が、通常はＲＯＭ　１３８に保管される。ＲＡＭ　１４０には、通常は、処理ユニット１３２から即座にアクセス可能、かつ／または処理ユニット１３２によって現在操作中のデータおよび／またはプログラムモジュールが含まれる。制限ではなく例として、図８に、オペレーティングシステム１４４、アプリケーションプログラム１４６、他のプログラムモジュール１４８、およびプログラムデータ１５０を示す。

【００５３】

コンピュータ１３０に、他のリムーバブル／ノンリムーバブルで揮発性／不揮発性のコンピュータ記憶媒体も含めることができる。たとえば、図８に、ノンリムーバブル不揮発性磁気媒体から読み取るかこれに書き込むハードディスクドライブ１５４を示す。図８に、リムーバブル不揮発性磁気ディスク１５８から読み取るかこれに書き込む磁気ディスクドライブ１５６と、ＣＤ－ＲＯＭまたは他の光媒体などのリムーバブル不揮発性光ディスク１６２から読み取るかこれに書き込む光ディスクドライブ１６０も示す。例示的なオペレーティング環境で使用できる他のリムーバブル／ノンリムーバブルで揮発性／不揮発性のコンピュータ記憶媒体に、磁気テープカセット、フラッシュメモリカード、ディジタル多用途ディスク、ディジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭ、および類似物が含まれるが、これに制限はされない。ハードディスクドライブ１５４、磁気ディスクドライブ１５６、および光ディスクドライブ１６０は、通常は、インターフェース１６６などの不揮発性メモリインターフェースによってシステムバス１３６に接続される。

【００５４】

上で説明し、図８に示したドライブまたは他の大容量ストレージデバイスおよびそれに関連するコンピュータ記憶媒体は、コンピュータ１３０のコンピュータ可読命令、データ構造、プログラムモジュール、および他のデータのストレージを提供する。図８では、たとえば、ハードディスクドライブ１５４が、オペレーティングシステム１７０、アプリケーションプログラム１７２、他のプログラムモジュール１７４、およびプログラムデータ１７６を保管するものとして図示されている。これらのコンポーネントを、オペレーティングシステム１４４、アプリケーションプログラム１４６、他のプログラムモジュール１４８、およびプログラムデータ１５０と同一のまたは異なるもののいずれかとすることが

できることに留意されたい。オペレーティングシステム１７０、アプリケーションプログラム１７２、他のプログラムモジュール１７４、およびプログラムデータ１７６は、最低限でも異なるコピーであることを示すために、異なる符号を与えられている。

【００５５】

ユーザは、キーボード１８０およびポインティングデバイス１８２（たとえば、マウス、トラックボール、ペン、またはタッチパッド）などの入力デバイスまたはユーザインターフェース選択デバイスを介してコンピュータ１３０にコマンドおよび情報を入力することができる。他の入力デバイス（図示せず）に、マイクロホン、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、スキャナ、または類似物を含めることができる。上記および他の入力デバイスは、システムバス１３６に結合されたユーザ入力インターフェース１８４を介して処理ユニット１３２に接続されるが、パラレルポート、ゲームポート、またはＵｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　Ｂｕｓ（ＵＳＢ）などの他のインターフェースおよびバス構造によって接続することができる。モニタ１８８または他のタイプのディスプレイデバイスも、ビデオインターフェース１９０などのインターフェースを介してシステムバス１３６に接続される。モニタ１８８のほかに、コンピュータに、しばしば、プリンタおよびスピーカなどの他の周辺出力デバイス（図示せず）が含まれ、これらの周辺出力デバイスは、出力周辺インターフェース（図示せず）を介して接続することができる。

【００５６】

コンピュータ１３０は、リモートコンピュータ１９４などの１つまたは複数のリモートコンピュータへの論理接続を使用してネットワーク化された環境で動作することができる。リモートコンピュータ１９４は、パーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピアデバイス、または他の一般的なネットワークノードとすることができ、通常は、上でコンピュータ１３０に関して説明した要素の多くまたはすべてを含む。図８に示された論理接続に、ローカルエリアネットワーク（ＬＡＮ）１９６および広域ネットワーク（ＷＡＮ）１９８が含まれるが、他のネットワークも含めることができる。ＬＡＮ　１３６および／またはＷＡＮ　１３８は、有線ネットワーク、無線ネットワーク、またはこれらの組合せなどとすることができる。そのようなネットワーキング環境は、オフィス、会社全体のコンピュータネットワーク、イントラネット、およびグローバルコンピュータネットワーク（たとえば、インターネット）でありふれたものである。

【００５７】

ローカルエリアネットワーキング環境で使用されるときに、コンピュータ１３０は、ネットワークインターフェースまたはネットワークアダプタ１８６を介してＬＡＮ　１９６に接続される。広域ネットワーキング環境で使用されるときに、コンピュータ１３０に、通常は、インターネットなどのＷＡＮ　１９８を介する通信を確立する、モデム１７８または他の手段が含まれる。モデム１７８は、内蔵または外付けとすることができるが、ユーザ入力インターフェース１８４または他の適当な機構を介してシステムバス１３６に接続される。ネットワーク化された環境では、コンピュータ１３０に関して図示されたプログラムモジュールまたはその一部を、リモートメモリストレージデバイス（図示せず）に保管することができる。制限ではなく例として、図８に、メモリデバイスに常駐するものとしてリモートアプリケーションプログラム１９２を示す。図示のネットワーク接続は例示的であり、コンピュータの間の通信リンクを確立する他の手段を使用することができる。

【００５８】

一般に、コンピュータ１３０のデータプロセッサは、異なる時にコンピュータのさまざまなコンピュータ可読記憶媒体に保管される命令によってプログラムされる。プログラムおよびオペレーティングシステムは、通常、たとえばフロッピディスクまたはＣＤ－ＲＯＭで配布される。そこから、これらが、コンピュータの副メモリにインストールまたはロードされる。実行時に、これらは、少なくとも部分的にコンピュータの主電子メモリにロードされる。本明細書に記載の発明には、これらおよび他のさまざまなタイプのコンピュ

ータ可読記憶媒体にマイクロプロセッサまたは他のデータプロセッサに関連して以下で説明するステップを実装する命令またはプログラムが含まれるときに、そのような媒体が含まれる。本発明に、本明細書に記載の方法および技法に従ってプログラムされたときのコンピュータ自体も含まれる。

【００５９】

例示のために、オペレーティングシステムなどのプログラムおよび他の実行可能プログラムコンポーネントを、本明細書では別個のブロックとして図示した。しかし、そのようなプログラムおよびコンポーネントが、さまざまな時にコンピュータの異なるストレージコンポーネントに常駐し、コンピュータのデータプロセッサによって実行されることを理解されたい。

【００６０】

コンピュータ１３０を含む例示的なコンピューティングシステム環境に関して説明したが、本発明は、多数の他の汎用のまたは特殊目的のコンピューティングシステム環境またはコンピューティングシステム構成と共に動作する。このコンピューティングシステム環境は、本発明の使用または機能性の範囲に関する制限を提案することを意図したものではない。さらに、このコンピューティングシステム環境を、この例示的なオペレーティング環境に示されたコンポーネントのいずれかまたはその組合せに対する依存性または要件を有すると解釈してはならない。本発明と共に使用するのに適する可能性がある周知のコンピューティングシステム、コンピューティング環境、および／またはコンピューティング構成の例に、パーソナルコンピュータ、サーバコンピュータ、ハンドヘルドデバイス、ラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラマブルコンシューマエレクトロニクス、携帯電話機、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記のシステムまたはデバイスのいずれかを含む分散コンピューティング環境、および類似物が含まれるが、これらに制限はされない。

【００６１】

本発明を、プログラムモジュールなど、１つまたは複数のコンピュータまたは他のデバイスによって実行されるコンピュータ実行可能命令の全般的な文脈で説明することができる。一般に、プログラムモジュールには、特定のタスクを実行するかまたは特定の抽象データ型を実装する、ルーチン、プログラム、オブジェクト、コンポーネント、およびデータ構造が含まれるが、これに制限はされない。本発明は、通信ネットワークを介してリンクされたリモート処理デバイスによってタスクが実行される分散コンピューティング環境で実践することもできる。分散コンピューティング環境では、プログラムモジュールを、メモリストレージデバイスを含むローカルとリモートの両方のコンピュータ記憶媒体に配置することができる。

【００６２】

ソフトウェアアーキテクチャの文脈でのインターフェースは、ソフトウェアモジュール、コンポーネント、コード部分、またはコンピュータ実行可能命令の他のシーケンスを含む。インターフェースには、たとえば、第１モジュールの代わりにコンピューティングタスクを実行する第２モジュールにアクセスする第１モジュールが含まれる。第１および第２のモジュールに、一例で、オペレーティングシステムによって提供されるアプリケーションプログラミングインターフェース（ＡＰＩ）、コンポーネントオブジェクトモデル（ＣＯＭ）インターフェース（たとえば、ピアツーピアアプリケーション通信用）、およびＸＭＩ（ｅｘｔｅｎｓｉｂｌｅ ｍａｒｋｕｐ ｌａｎｇｕａｇｅ ｍｅｔａｄａｔａ ｉｎｔｅｒｃｈａｎｇｅ ｆｏｒｍａｔ）インターフェース（たとえば、ウェブサービス間の通信用）が含まれる。

【００６３】

インターフェースは、Ｊａｖａ（登録商標） ２ Ｐｌａｔｆｏｒｍ Ｅｎｔｅｒｐｒｉｓｅ Ｅｄｉｔｉｏｎ（Ｊ２ＥＥ）、ＣＯＭ、または分散ＣＯＭ（ＤＣＯＭ）の例など、密結合同期式実装とすることができる。その代わりにまたはそれに加えて、インターフ

ェースを、ウェブサービス（たとえば、ｓｉｍｐｌｅ　ｏｂｊｅｃｔ　ａｃｃｅｓｓ　ｐｒｏｔｏｃｏｌを使用する）など、疎結合非同期式実装とすることができる。一般に、インターフェースに、密結合、疎結合、同期式、および非同期式という特性の任意の組合せが含まれる。さらに、インターフェースは、標準プロトコル、独自プロトコル、または標準プロトコルおよび独自プロトコルの任意の組合せに準拠することができる。

【００６４】

本明細書に記載のインターフェースは、そのすべてを、単一のインターフェースの一部とするか、または別々のインターフェースとして実装するか、あるいはその組合せとすることができる。インターフェースは、機能性を提供するためにローカルにまたはリモートで実行することができる。さらに、インターフェースに、本明細書で図示または説明されたものより多くのまたは少数の機能性を含めることができる。

【００６５】

本明細書で図示され説明された方法の実行の順序または、そうでないと指定されていない限り、本質的でない。すなわち、この方法の要素を、そうでないと指定されていない限り任意の順序で実行することができ、この方法に、本明細書で開示されたものより多数または少数の要素を含めることができる。たとえば、別の要素の前、それと同時、またはその後に特定の要素を実行または実施することが、本発明の範囲内であることが企図されている。

【００６６】

上記に鑑みて、本発明の複数の目的が達成され、他の有利な結果が達成されることがわかる。

【００６７】

本発明の範囲から外れずに、上記のシステムおよび方法においてさまざまな変更を行うことができるので、上記の説明に含まれ、添付図面に示されたすべてが、制限的な意味ではなく例示と解釈されなければならない。

【００６８】

付録Ａ

（セクション１：ジオメトリ生成サンプルコード）

我々は、生成したい「インセット」ワイプパターンに基づいて、クロスセクションとして２５％、５０％、および７５％でのトランジションパターンを有する。これらを積み重ね、頂点を接続し、次のようなモデルを得ることができる。Ｘは出力の幅であり、Ｙは出力の高さであり、Ｚはトランジションの進行である。

```
const DWORD VERTEX_XYZ_TEX2::FVF = D3DFVF_XYZ | D3DFVF_TEX2;
m_dwNumPatternIndices = 6;
m_dwNumPatternVertices = 4;
D3DXCreateMeshFVF(            // D3DXMeshobjectを作成する
                              // DirectX API
  m_dwNumPatternIndices /  3, // 面/三角形の個数
  m_dwNumPatternVertices,     // 頂点の個数
  D3DXMESH_SYSTEMMEM,         // システムメモリ内にD3DXMeshを作成する
  VERTEX_XYZ_TEX2::FVF,       // 頂点フォーマット：ＸＹＺ位置および
                              // ２セットのテクスチャ座標
  pd3dDevice,                 // GPUプロセスのインスタンス
  &pSysMemMesh                // D3DXMESHポインタを返す
);
VERTEX_XYZ_TEX2* pVertices = NULL;
pSysMemMesh->LockVertexBuffer( D3DLOCK_NOSYSLOCK,
(void**)&pVertices );
// D3DXMeshオブジェクトからVertexBufferを得る。
// 上記の描画に基づくハードコード化された(x, y, z)座標を用いて
```

```
// 頂点を書き込む。
pVertices[0].v.x  = 1.0f;
pVertices[0].v.y  = 1.0f;
pVertices[0].v.z  = 1.0f;
// (t0.x, t0.y) = (x, y)かつ(t1.x, t1.y) = (z, Progress)にする
pVertices[0].t0.x = pVertices[0].v.x;
pVertices[0].t0.y = pVertices[0].v.y;
pVertices[0].t1.x = pVertices[0].v.z;
pVertices[0].t1.y = 0.50f; // Progress
pVertices[1].v.x  = 0.0f;
pVertices[1].v.y  = 1.0f;
pVertices[1].v.z  = 0.0f;
pVertices[1].t0.x = pVertices[1].v.x;
pVertices[1].t0.y = pVertices[1].v.y;
pVertices[1].t1.x = pVertices[1].v.z;
pVertices[1].t1.y = 0.50f; // Progress
pVertices[2].v.x  = 0.0f;
pVertices[2].v.y  = 0.0f;
pVertices[2].v.z  = 1.0f;
pVertices[2].t0.x = pVertices[2].v.x;
pVertices[2].t0.y = pVertices[2].v.y;
pVertices[2].t1.x = pVertices[2].v.z;
pVertices[2].t1.y = 0.50f; // Progress
pVertices[3].v.x  = 1.0f;
pVertices[3].v.y  = 0.0f;
pVertices[3].v.z  = 1.0f;
pVertices[3].t0.x = pVertices[3].v.x;
pVertices[3].t0.y = pVertices[3].v.y;
pVertices[3].t1.x = pVertices[3].v.z;
pVertices[3].t1.y = 0.50f; // Progress

pSysMemMesh->UnlockVertexBuffer();
// このメッシュオブジェクト（現在使用中でない）を
// スワップアウトできることをDirect3Dに知らせる。

WORD* pIndices = NULL;
pSysMemMesh->LockIndexBuffer( D3DLOCK_NOSYSLOCK,
(void**)&pIndices);
// D3DXMeshオブジェクトからインデックスバッファを得る。
// インデックスを書き込む（時計回りの順序で）。
// 三角形１
pIndices[0] = 0;
pIndices[1] = 1;
pIndices[2] = 3;
// 三角形２
pIndices[3] = 1;
pIndices[4] = 2;
pIndices[5] = 3;

pSysMemMesh->UnlockIndexBuffer();
```

```
// システムメモリおよびデバイスメモリでD3DXMeshオブジェクトを
// 管理するためにCDXUTMeshオブジェクトを作成した。
*ppMesh = new CDXUTMesh{ L"CInsetPatternMesh" );

(*ppMesh)->m_pSysMemMesh = pSysMemMesh.Detach();
// 前に作成したD3DXMeshオブジェクトをCDXUTMeshによって管理
// させる。
```

【００６９】

（セクション２：メッシュ生成サンプルコード）

メッシュは、ＤＸＵＴＭｅｓｈオブジェクトとして作成される。

【００７０】

各ＤＸＵＴＭｅｓｈオブジェクトは、２つのＤ３ＤＸＭｅｓｈオブジェクトを管理し、一方はシステムメモリ内（ＣＰＵがアクセスできる）、他方はデバイスメモリ内（ＧＰＵ／ディスプレイアダプタがアクセスできる）にある。システムメモリ内のＤ３ＤＸＭｅｓｈオブジェクトが変更されるたびに、アプリケーションが、これをデバイスメモリにダウンロードする必要がある。そうでなければ、ＧＰＵは、すべての描画コマンドにＤ３ＤＸＭｅｓｈオブジェクトを再利用する。両方のＤ３ＤＸＭｅｓｈオブジェクトの内容が、同一である。これには、１つの頂点バッファおよび１つのインデックスバッファが含まれる。

【００７１】

頂点バッファは、頂点の配列である。たとえば、ワイプトランジションは、どのデータが頂点に含まれるかを定義する。

【００７２】

すべての頂点が、以下に示された例など、３つのエントリを有する：

３Ｄ座標（ｘ，ｙ，ｚ）は、頂点位置用である。各頂点位置は、円、スター、ハートなど、ジオメトリ形状に基づいて計算される。

【００７３】

２Ｄテクスチャ座標（ｔｕ１，ｔｖ１）は、ＧＰＵまたは命令のセットに、頂点の赤／緑／青（ＲＧＢ）値をビデオソースのどこで探すかを指示するためのものである。ワイプトランジションの例では、ｔｕ１がｘ値に割り当てられ、ｔｖ１がｙ値に割り当てられる。

【００７４】

および、第２の２Ｄテクスチャ座標（ｔｕ２，ｔｖ２）は、ＧＰＵまたは命令のセットに、この頂点のアルファ値をアルファマップのどこで探すかを指示するためのものである。ｔｕ２は、ｚ値に割り当てられ、ｔｖ２は、ランタイム／アニメーション時の進行％値に割り当てられる。

【００７５】

インデックスバッファは、整数インデックスの配列である。各インデックスは、頂点バッファにインデクシングされる。各３つのインデックスが、三角形を形成する。すべての三角形が、１つの３Ｄモデルを形成する。

【００７６】

Ａｌｉａｓ社のＭａｙａ（登録商標）などの他の３Ｄモデリングツールを使用して、．Ｘファイルを作成することができる。作成された．Ｘファイルをロードする時に、０．０から１．０の範囲内の可視領域におさまるように３Ｄモデルをスケーリングすることによって、各（ｘ，ｙ，ｚ）座標の正規化が必要になる場合がある。

【００７７】

（セクション３：シャープエッジサンプルコード）

```
// Direct3D状態をセットアップする
// pd3dDeviceは、Direct3D/GPUのインスタンスポインタである。
pd3dDevice->SetRenderState( D3DRS_ALPHATESTENABLE, FALSE);
```

```
// アルファテストを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, FALSE);
// アルファブレンディングを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_STENCILENABLE, FALSE);
// ステンシルテストを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_ZENABLE, D3DZB_TRUE);
// デプステストをイネーブルする
pd3dDevice->SetRenderState( D3DRS_ZWRITEENABLE,TRUE);
// Z値をデプスバッファに保存する

// Direct3Dテクスチャブレンディング状態をセットアップする。
// Direct3Dは８つまでの状態を許容するが、１つだけが必要である。
pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 0 );
// ステージ０にt0テクスチャ座標を使用する。
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);
// RGB値を出力する。
pd3dDevice->SetTextureStageState( 0,
D3DTSS_COLORARG1, D3DTA_TEXTURE );
// テクスチャからRGB値を出力する。
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAOP,
D3DTOP_DISABLE );
// ステージ０以上からのアルファ出力はない。
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
D3DTOP_DISABLE );
// ステージ１以上からのRGB出力はない。

// 入ってくるソースを描画する
// テクスチャ２および第１メッシュを使用する。
pd3dDevice->SetTexture(0, pTextureIn);
// pTextureInは、入ってくるビデオフレームを保管するTexture
// オブジェクトである。このテクスチャをテクスチャブレンディン
// グステージ０にセットする。
LPD3DXMESH pPlaneMesh = m_pPlaneMesh->GetLocalMesh();
// デバイス/GPUメモリに保管されたD3DXMeshオブジェクトを
// CDXUTMeshから得る。
pPlaneMesh->DrawSubset( 0 ); // D3D/GPUにこのPlaneMesh、
// TextureIn、および現在のセッティングを使用する描画を
// 実行させる。

// 出てゆくソースを描画する。
// テクスチャ１およびトランジションパターンメッシュを使用する。
// 並進変換行列を適用して、トランジションパターンメッシュをｚ方向の
// 小さい方に移動する。
D3DXMATRIX matTranslation;
D3DXMatrixTranslation( &matTranslation, 0, 0, -fltProgress);
// 並進変換行列を作成して、PatternMeshをｚ方向の小さい方に移動する。
pd3dDevice->SetTransform( D3DTS_WORLD, &matTranslation );
// D3Dワールド変換行列をセットする。
// D3Dワールド変換行列はモデル空間座標をワールド空間に変換する。
```

```
// トランジションパターンメッシュの(0, 0, 0)は、(0, 0, -
// fltProgress)になる。
pd3dDevice->SetTexture(0, pTextureOut);
LPD3DXMESH pPatternMesh = m_pPatternMesh->GetLocalMesh();
pPatternMesh->DrawSubset( 0 ); // D3D/GPUにこのPatternMesh、
// TextureOut、および現在のセッティングを使用する描画を
// 実行させる。
```

【００７８】

このルーチンは、フレームごとに繰り返される。ｆｌｔＰｒｏｇｒｅｓｓは、トランジション持続時間、フレームレート、および現在のフレームに基づいて計算される。

【００７９】

Ｄ３Ｄ／ＧＰＵは、第１メッシュ／入ってくるビデオソースを描画する時に、ＴｅｘｔｕｒｅＩｎ全体をレンダターゲット面に置き、レンダターゲット面のピクセルごとにｚ値をデプスバッファにセットする。

【００８０】

Ｄ３Ｄ／ＧＰＵは、トランジションパターンメッシュ／出てゆくビデオソースを描画する時に、デプステストを実行する。Ｄ３Ｄ／ＧＰＵは、トランジションパターンメッシュピクセルのｚ値をデプスバッファ内のそのピクセルのｚ値と比較する。見る人から遠いピクセルは捨てられる。

【００８１】

（セクション４：ソフトエッジサンプルコード）

```
// 入ってくるビデオソースをまず描画する。
// Direct3D状態をセットアップする。
// pd3dDeviceは、Direct3D/GPUのインスタンスポインタである。
pd3dDevice->SetRenderState( D3DRS_SHADEMODE, D3DSHADE_GOURAUD);
// スムーズシェーディングモードにセットする。したがって、D3D/GPU
// は、各頂点でテクスチャ座標を計算し、次に、各三角形の３頂点の
// 間で補間を実行して、その三角形内の各ピクセルのテクスチャ座標
// を計算する。そのテクスチャ座標は、後に、選択されたテクスチャ
// からRGB値およびアルファ値をルックアップするのに使用される。

pd3dDevice->SetRenderState( D3DRS_ALPHATESTENABLE, FALSE);
// アルファテストを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, FALSE);
// アルファブレンディングを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_STENCILENABLE, FALSE);
// ステンシルテストを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_ZENABLE, D3DZB_FALSE);
// デプステストの必要はない。
pd3dDevice->SetRenderState( D3DRS_ZWRITEENABLE, FALSE);
// Z値をデプスバッファに保存する必要はない。

// Direct3Dテクスチャブレンディング状態をセットアップする。
// Direct3Dは８つまでの状態を許容するが、１つだけが必要である。

// pTextureInの(tu1, tv1)から各頂点のRGB値をルックアップする。
pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 0 );
// tu1, tv1テクスチャ座標をステージ０に使用する。
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
D3DTOP_SELECTARG1); // RGB値を出力する。
```

```
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
D3DTA_TEXTURE ); // テクスチャからRGB値を出力する。
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAOP,
D3DTOP_DISABLE ); // ステージ０以上からのアルファ出力はない。
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
D3DTOP_DISABLE );
// ステージ１以上からのRGB出力はない。

// 入ってくるソースを描画する。
// テクスチャ２および第１メッシュを使用する。

pd3dDevice->SetTexture(0, pTextureIn);
// pTextureInは、入ってくるビデオフレームを保管するテクスチャ
// オブジェクトである。このテクスチャをテクスチャブレンディン
// グステージ０にセットする。

LPD3DXMESH pPlaneMesh = m_pPlaneMesh->GetLocalMesh();
// デバイス/GPUメモリに保管されたD3DXMeshオブジェクトを
// CDXUTMeshから得る。
pPlaneMesh->DrawSubset( 0 ); // D3D/GPUにこのPlaneMesh、
// TextureIn、および現在のセッティングを使用する描画を
// 実行させる。

// 出てゆくソースを描画する。

// レンダ状態をセットアップする。
// アルファブレンディング動作用のセットアップ。
pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, TRUE );
pd3dDevice->SetRenderState( D3DRS_SRCBLEND, D3DBLEND_SRCALPHA );
pd3dDevice->SetRenderState( D3DRS_DESTBLEND, DBLEND_INVSRCALPHA);

// テクスチャブレンディング
// ステージ０：
// ColorOp : Select Texture
// AlphaOp : Current
// pTextureOutの(tu1, tv1)から各頂点のRGB値をルックアップする。
pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 0 );
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
D3DTA_TEXTURE );
// アルファ値をセットしない。
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAOP,
D3DTOP_SELECTARG1);
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAARG1,
D3DTA_CURRENT );

// ステージ１：
// ColorOp : Current
// AlphaOp : Select Texture
```

```
RTN_HR_IF_FAILED(pd3dDevice->SetTexture(1, m_pAlphaMapTexture));
// 前のステージからのRGB値の変更なし。
pd3dDevice->SetTextureStageState( 1, D3DTSS_TEXCOORDINDEX , 1 );
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLORARG1,
D3DTA_CURRENT );
// pAlphaMapTexture(tu2, tv2)から各頂点のアルファ値を
// ルックアップする。
pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAOP,
D3DTOP_SELECTARG1);
pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAARG1,
D3DTA_TEXTURE );

// ステージ２：ディスエーブル
pd3dDevice->SetTextureStageState( 2, D3DTSS_ALPHAOP,
D3DTOP_DISABLE);
pd3dDevice->SetTextureStageState( 2, D3DTSS_COLOROP,
D3DTOP_DISABLE );

pd3dDevice->SetSamplerState( 0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR
);
pd3dDevice->SetSamplerState( 0, D3DSAMP_MAGFILTER,
D3DTEXF_LINEAR);

pd3dDevice->SetSamplerState( 1, D3DSAMP_MINFILTER,
D3DTEXF_LINEAR);
pd3dDevice->SetSamplerState( 1, D3DSAMP_MAGFILTER,
D3DTEXF_LINEAR);

pd3dDevice->SetSamplerState(1, D3DSAMP_ADDRESSU,
D3DTADDRESS_CLAMP);
pd3dDevice->SetSamplerState(1, D3DSAMP_ADDRESSV,
D3DTADDRESS_CLAMP );

// トランジションパターンメッシュの各頂点のtv2をProgressにセットする。
VERTEX_XYZ_TEX2* pVertices = NULL;
pSysMemMesh->LockVertexBuffer( D3DLOCK_NOSYSLOCK,
(void**)&pVertices );

DWORD dwNumPatternVertices = pSysMemMesh->GetNumVertices();
for(DWORD i = 0; i < dwNumPatternVertices; i++)
  pVertices[i].t2.y = fltProgress; // Progress
// fltProgressは、トランジション持続時間、フレームレート、
// および現在のフレームに基づいて計算される。

pSysMemMesh->UnlockVertexBuffer();
m_pMesh->InvalidateDeviceObjects();
m_pMesh->RestoreDeviceObjects(pd3dDevice);
// RestoreDeviceObjects()は、ローカルメッシュを作成する。
```

```
// テクスチャ１およびトランジションパターンメッシュを使用する。
pd3dDevice->SetTexture(0, pTextureOut);
// pTextureOutは、出てゆくビデオフレームを保管するテクスチャオブ
// ジェクトである。このテクスチャをテクスチャブレンディングステー
// ジ０にセットする。

LPD3DXMESH pPatternMesh = m_pPatternMesh->GetLocalMesh();
pPatternMesh->DrawSubset( 0 ); // D3D/GPUにこのPatternMesh、
// TextureOut、および現在のセッティングを使用する描画を
// 実行させる。
```

【００８２】

上に記載のルーチンは、フレームごとに繰り返される。Ｄ３Ｄ／ＧＰＵは、第１メッシュ／入ってくるビデオソースを描画する時に、ＴｅｘｔｕｒｅＩｎ全体をレンダターゲット面に置く。Ｄ３Ｄ／ＧＰＵは、トランジションパターンメッシュ／出てゆくビデオソースを描画する時に、アルファマップからアルファ値をルックアップし、これをＴｅｘｔｕｒｅＯｕｔからのＲＧＢ値と共にアルファブレンディング計算に使用し、ＲＧＢ値はレンダターゲット内に置かれる。

【００８３】

（セクション５：アルファマップ生成サンプルコード）

各頂点に、位置および色の情報が含まれる。

【００８４】

右上角（１，０）は、不透明である。左下角（０，１）は、透明である。左上から右下への線は、５０％透明である。透明から不透明への勾配は、正弦曲線に従う。このメッシュをテクスチャにレンダリングすると、ソフトエッジ作成に使用されるアルファマップが得られる。バーンエッジ用のアルファマップは、同一の形状であるが、異なるカラー値／アルファ値を有する。右上角（１，０）および左下角（０，１）が、透明である。左上から右下への線は、バーンエッジ強度によってセットされる透明度を有する。

```
struct ALPHAMAPVERTEX
{
    D3DXVECTOR3 v;        // 位置
    D3DCOLOR color;       // 色
    static const DWORD FVF;
};
const DWORD ALPHAMAPVERTEX::FVF = D3DFVF_XYZ | D3DFVF_DIFFUSE;
          m_wNumSegments = 36;
          m_dwNumVertice
                    = 2 // 左上および右下
                    + (m_wNumSegments + 1) * 3; // 3: 左-中-右
          m_dwNumIndices
                    = 2 * 4
                    + (m_wNumSegments - 1) * 6;
          m_dwNumIndices *= 3;
D3DXCreateMeshFVF(
          m_dwNumIndices / 3, // 面の個数、
          m_dwNumVertices,    // 頂点の個数、
          D3DXMESH_SYSTEMMEM, // システムメモリに保存、
          ALPHAMAPVERTEX::FVF,// 頂点フォーマット、
          pd3dDevice,         // GPUインスタンス、
          &pSysMemMesh        // LPD3DXMESHを返す
  );
```

```
// 書き込みのためにVertexBufferをロックする。
ALPHAMAPVERTEX* pVertices = NULL;
pSysMemMesh->LockVertexBuffer( D3DLOCK_NOSYSLOCK,
(void**)&pVertices);
          D3DCOLOR clrRight;
          D3DCOLOR clrMiddle;
          D3DCOLOR clrLeft;
          if(m_Type == SoftEdge)
          {
                    clrLeft = 0;
                    clrMiddle = 0x80808080;
                    clrRight = 0xffffffff;
          }
          else if(m_Type == BurnEdge)
          {
                    clrLeft = 0;
                    clrMiddle = D3DCOLOR_COLORVALUE
(m_fltBurnIntensity,m_fltBurnIntensity,m_fltBurnIntensity,
m_fltBurnIntensity);
                    clrRight = 0;
          }
          else
          {
                    clrLeft = 0xffffffff;
                    clrMiddle = 0;
                    clrRight = 0xffffffff;
          }
          // 左下
  pVertices[0].v.x   = 0.0f;
  pVertices[0].v.y   = 1.0f;
  pVertices[0].v.z   = 0.0f;
  pVertices[0].color = clrLeft;

          // 右上（SoftEdgeは１００％；BurnEdgeは０）
  pVertices[1].v.x   = 1.0f;
  pVertices[1].v.y   = 0.0f;
  pVertices[1].v.z   = 0.0f;
  pVertices[1].color = clrRight;

// 左上
  pVertices[2].v.x   = 0.0f;
  pVertices[2].v.y   = 0.0f;
  pVertices[2].v.z   = 0.0f;
  pVertices[2].color = clrLeft;
  pVertices[3].v.x   = 0.0f;
  pVertices[3].v.y   = 0.0f;
  pVertices[3].v.z   = 0.0f;
  pVertices[3].color = clrMiddle;

  pVertices[4].v.x   = 0.0f;
```

```
  pVertices[4].v.y   = 0.0f;
  pVertices[4].v.z   = 0.0f;
  pVertices[4].color = clrRight;

        // 右下
  pVertices[5].v.x   = 1.0f;
  pVertices[5].v.y   = 1.0f;
  pVertices[5].v.z   = 0.0f;
  pVertices[5].color = clrLeft;

  pVertices[6].v.x   = 1.0f;
  pVertices[6].v.y   = 1.0f;
  pVertices[6].v.z   = 0.0f;
  pVertices[6].color = clrMiddle;

  pVertices[7].v.x   = 1.0f;
  pVertices[7].v.y   = 1.0f;
  pVertices[7].v.z   = 0.0f;
  pVertices[7].color = clrRight;

    WORD index = 8; // ０から７は上で割り当て済み。

       // ０°から１８０°までの正弦曲線を使用する。
       // sinの値は０から１を経て０まで。
     WORD i;
           for(i = 1; i < m_wNumSegments; i ++)
           {
                FLOAT fltXY = (FLOAT)i / (FLOAT)m_wNumSegments;
                FLOAT fltAngle = D3DX_PI * fltXY;
                FLOAT fltDistance = ::sin(fltAngle);
                fltDistance *= m_fltGradientWidth / 2.0f;
                // 勾配領域の幅を調整する。
                // fltDistance範囲を検査する。
                if(fltXY < 0.5f)
                // 上半分なのでfltXY - fltDistance > 0
                      fltDistance = min(fltXY, fltDistance);
                else if(fltXY > 0.5f)
                // 下半分なのでfltXY + fltDistance < 1
                      fltDistance = min(1.0f - fltXY,
fltDistance);
                else
                // 中央なのでfltXY + fltDistance < 1かつ
                // fltXY- fltDistance > 0
                      fltDistance = min(0.5f, fltDistance);
                // 左
                pVertices[index].v.x   = fltXY - fltDistance;
                pVertices[index].v.y   = fltXY;
                pVertices[index].v.z   = 0.0f;
                pVertices[index].color = clrLeft;
```

```
                // 中
                pVertices[index + 1].v.x   = fltXY;
                pVertices[index + 1].v.y   = fltXY;
                pVertices[index + 1].v.z   = 0.0f;
                pVertices[index + 1].color = clrMiddle;

                // 右
                pVertices[index + 2].v.x   = fltXY +
fltDistance;
                pVertices[index + 2].v.y   = fltXY;
                pVertices[index + 2].v.z   = 0.0f;
                pVertices[index + 2].color = clrRight;

                index += 3;
            }
            RTN_HR_IF_FAILED( pSysMemMesh->UnlockVertexBuffer());
            // 書き込みのためにIndexBufferをロックする。
            WORD* pIndices = NULL;
            RTN_HR_IF_FAILED(
pSysMemMesh->LockIndexBuffer(D3DLOCK_NOSYSLOCK, (void**)&pIndices));
            RTN_HR_IF_BADPTR(pIndices);
  // インデックスを書き込む。
  // 左端
  pIndices[0] = 2;
  pIndices[1] = 8;
  pIndices[2] = 0;

  // 左中
  pIndices[3] = 3;
  pIndices[4] = 9;
  pIndices[5] = 8;

  // 右中
  pIndices[6] = 3;
  pIndices[7] = 10;
  pIndices[8] = 9;

        // 一番上のスキャンラインがすべて不透明になるように
        // 一番上を最後に描画する。
        index = 9;
        for(i = 1; i < m_wNumSegments - 1; i ++)
        {
                // 左へ
                pIndices[index + 0] = 0;
                pIndices[index + 1] = i * 3 + 5;
                pIndices[index + 2] = i * 3 + 8;

                // 左象限
                pIndices[index + 3] = i * 3 + 5;
                pIndices[index + 4] = i * 3 + 6;
```

```
                pIndices[index + 5] = i * 3 + 8;

                pIndices[index + 6] = i * 3 + 6;
                pIndices[index + 7] = i * 3 + 9;
                pIndices[index + 8] = i * 3 + 8;

                // 右象限
                pIndices[index + 9] = i * 3 + 6;
                pIndices[index + 10] = i * 3 + 7;
                pIndices[index + 11] = i * 3 + 9;

                pIndices[index + 12] = i * 3 + 7;
                pIndices[index + 13] = i * 3 + 10;
                pIndices[index + 14] = i * 3 + 9;

                // 右へ
                pIndices[index + 15] = 1;
                pIndices[index + 16] = i * 3 + 10;
                pIndices[index + 17] = i * 3 + 7;

                index += 18;
            }
            // 右端
            pIndices[index + 0] = 7;
            pIndices[index + 1] = i * 3 + 7;
            pIndices[index + 2] = 1;

            // 右中
            pIndices[index + 3] = 6;
            pIndices[index + 4] = i * 3 + 6;
            pIndices[index + 5] = i * 3 + 7;

            // 左中
            pIndices[index + 6] = 6;
            pIndices[index + 7] = i * 3 + 5;
            pIndices[index + 8] = i * 3 + 6;

            // 一番下
            pIndices[index + 9] = 5;
            pIndices[index + 10] = 0;
            pIndices[index + 11] = i * 3 + 5;

            // 一番上
            pIndices[index + 12] = 4;
            pIndices[index + 13] = 1;
            pIndices[index + 14] = 10;
    }
    RTN_HR_IF_FAILED( pSysMemMesh->UnlockIndexBuffer());
    // D3DXMeshオブジェクトをシステムメモリおよびデバイスメ
    // モリ内で管理するためにCDXUTMeshオブジェクトを作成した。
```

```
  m_pMesh = new CDXUTMesh( L"CAlphaMap" );
    RTN_HR_IF_BADNEW(m_pMesh);

  m_pMesh->m_pSysMemMesh = pSysMemMesh.Detach();

  // 前に作成したD3DXMeshオブジェクトをCDXUTMeshによって管理
  // させる。
     m_pMesh->RestoreDeviceObjects(pd3dDevice);
     // デバイスメモリ内にコピーを作成する。

#ifdef _DEBUG
// メッシュを.Xファイルに保存することができる。
     if(get_MeshFilename( ))
     {
          VERIFY_OK(SaveMesh(pd3dDevice, get_MeshFilename( )));
     }
#endif
  // 128×256アルファマップを使用して、最初のスキャンライン
  // が全部不透明になり、最後のスキャンラインが全部透明になる
  // ようにする。
  CComPtr<IDirect3DSurface9> pSurfaceLevel;
    pd3dDevice->CreateRenderTarget(
         128, 256,           //UINT Width, UINT Height、
         D3DFMT_A8R8G8B8,    //レンダターゲットフォーマット、
         D3DMULTISAMPLE_NONE, //D3DMULTISAMPLE_TYPE、
         D3DMULTISAMPLE_NONE, //マルチサンプル品質、
         TRUE,               //BOOL Lockable,
         &pSurfaceLevel,     // IDirect3DSurface9**を返す
         ppSurface,
         NULL                //HANDLE* pSharedHandle
    );
    // D3D/GPUにこの面をレンダリングさせる。
    pd3dDevice->SetRenderTarget(0, pSurfaceLevel);

  // D3D状態をセットアップする。
  pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, TRUE );

  pd3dDevice->SetRenderState( D3DRS_SRCBLEND, D3DBLEND_SRCALPHA
  );
  pd3dDevice->SetRenderState( D3DRS_DESTBLEND, D3DBLEND_ZERO );
  // 既存のピクセルをブレンドする必要はない。

  pd3dDevice->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_GOURAUD);
  //スムーズシェーディング
  pd3dDevice->SetRenderState(D3DRS_COLORWRITEENABLE,
         D3DCOLORWRITEENABLE_RED | D3DCOLORWRITEENABLE_GREEN
         | D3DCOLORWRITEENABLE_BLUE |
         D3DCOLORWRITEENABLE_ALPHA );

  D3DXMATRIX matWorld;
```

```
    D3DXMatrixIdentity( &matWorld );
  pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld );
  // スケーリング／並進／回転なし。

  // D3Dカメラをセットアップする。
  D3DXMATRIX matView;
  D3DXVECTOR3 vFromPt   = D3DXVECTOR3( 0.5f, 0.5f, 10.0f );
  D3DXVECTOR3 vLookatPt = D3DXVECTOR3( 0.5f, 0.5f, 0.0f );
  D3DXVECTOR3 vUpVec    = D3DXVECTOR3( 0.0f, -1.0f, 0.0f );
  D3DXMatrixLookAtLH( &matView, &vFromPt, &vLookatPt, &vUpVec );
  pd3dDevice->SetTransform( D3DTS_VIEW, &matView );

  D3DXMATRIX matProj;
  D3DXMatrixOrthoLH( &matProj, 1.0f, 1.0f, 0.01f, 100.0f );
  pd3dDevice->SetTransform( D3DTS_PROJECTION, &matProj );

    // D3Dテクスチャブレンディングステージ状態をセットアップする。
  pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
  D3DTOP_SELECTARG1 );
  pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
  D3DTA_DIFFUSE);
  pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAOP,
  D3DTOP_SELECTARG1 );
  pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAARG1,
  D3DTA_DIFFUSE);

  pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
  D3DTOP_DISABLE );
  pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAOP,
  D3DTOP_DISABLE );

    RTN_HR_IF_BADNEW(m_pMesh);
  LPD3DXMESH pMesh = m_pMesh->GetLocalMesh();
    RTN_HR_IF_BADNEW(pMesh);

  // ビューポートをクリアする。
  pd3dDevice->Clear( 0L, NULL, D3DCLEAR_TARGET, 0x00000000,
1.0f, 0L);

  // シーンを開始する。
  pd3dDevice->BeginScene();
    HRESULT hr = S_OK;

    // 描画を実行する。
    pMesh->DrawSubset( 0 ));

  // シーンを終了する。
  pd3dDevice->EndScene();

  // アルファマップを保管するテクスチャを作成する。
```

```
  CComPtr<IDirect3DTexture9> pTexture;
     if(S_OK != pd3dDevice->CreateTexture(
          128, 256,   //UINT Width, UINT Height、
          1,          //MIPレベル、
          D3DUSAGE_DYNAMIC,
          D3DFMT_A8,  //テクスチャフォーマット、アルファのみ。
          D3DPOOL_DEFAULT, // デバイスメモリに置く。
          &pTexture,  // テクスチャを返す。
          NULL)       //HANDLE* pSharedHandle) ||
          || pTexture == NULL)
     {
          // A8フォーマットテクスチャの作成に失敗した。
          // A8R8G8B8フォーマットを試す。より多くのメモリを
          // 使用せよ。

          pd3dDevice->CreateTexture(
               128, 256,
               1,
               D3DUSAGE_DYNAMIC,
               D3DFMT_A8R8G8B8,
               D3DPOOL_DEFAULT,
               &pTexture,
               NULL
          );
  }
CComPtr<IDirect3DSurface9> pDestSurfaceLevel;
  pTexture->GetSurfaceLevel(0/* 第１MIPレベル */,
  &pDestSurfaceLevel);
  // RenderTargetピクセルをテクスチャにコピーする。
  D3DXLoadSurfaceFromSurface(
          pDestSurfaceLevel, //LPDIRECT3DSURFACE9 pDestSurface,
          NULL,              //CONST PALETTEENTRY *pDestPalette,
          NULL,              //CONST RECT *pDestRect,
          pSurfaceLevel,     //LPDIRECT3DSURFACE9 pSrcSurface,
          NULL,              //CONST PALETTEENTRY *pSrcPalette,
          NULL,              //CONST RECT *pSrcRect,
          D3DX_FILTER_POINT, //DWORD Filter,
          0                  //D3DCOLOR ColorKey
   );
   *ppTexture = pTexture.Detach();
   // このテクスチャはアルファマップである。

#ifdef _DEBUG
// アルファマップを.ddsファイルまたは
// jpgファイルに保存することができる。
     D3DXSaveSurfaceToFile(
     _T(".＼＼Shared＼＼RuntimeAlphaMap.dds"),   //ファイル名
     D3DXIFF_DDS,                          //ファイルフォーマット、
     pDestSurfaceLevel,             //この面をファイルに保存、
     NULL,                                 //NO PALETTEENTRY,
```

```
    NULL                              //NO RECT; 面全体
  );
#endif
```

【００８５】

（セクション６：バーンエッジサンプルコード）

```
// シャープエッジ技法またはソフトエッジ技法を用いるワイプトランジ
// ションのレンダリングの後。バーンエッジアルファマップを用いてフ
// ァンシイエッジテクスチャをトランジションパターンメッシュにレン
// ダリングする。バーンエッジアルファマップは、ソフトエッジ技法の
// アルファマップと同一の形状頂点配置を有する。しかし、カラー配置
// は異なる。これは、左上から右下への線の両側で透明である。左上か
// ら右下への線のアルファ値は、ＢｕｒｎＩｎｔｅｎｓｉｔｙである。
// アルファ値は、中心の線から徐々に０まで減る。

// レンダ状態をセットアップする。
pd3dDevice->SetRenderState( D3DRS_SHADEMODE, D3DSHADE_GOURAUD);
pd3dDevice->SetRenderState( D3DRS_ZENABLE, D3DZB_FALSE);
pd3dDevice->SetRenderState( D3DRS_ZWRITEENABLE, FALSE);
pd3dDevice->SetRenderState( D3DRS_ALPHATESTENABLE, FALSE);
pd3dDevice->SetRenderState( D3DRS_STENCILENABLE, FALSE);
pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, TRUE );
pd3dDevice->SetRenderState( D3DRS_SRCBLEND,D3DBLEND_SRCALPHA );
pd3dDevice->SetRenderState(D3DRS_DESTBLEND,D3DBLEND_INVSRCALPHA);
if(m_enumFancyEdge != FancyEdgeColor && m_enumFancyEdge !=
FancyEdgeColors)
{
    pd3dDevice->SetRenderState( D3DRS_DESTBLEND, D3DBLEND_ONE );
    // したがって、ファンシイエッジテクスチャの黒ピクセルは
    // 表示されない。
}
// テクスチャブレンディング
// ステージ０：
// ColorOp : Select Texture
// AlphaOp : Current
pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 0 );
pd3dDevice->SetTextureStageState( 0,
D3DTSS_COLOROP, D3DTOP_SELECTARG1 );
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
D3DTA_TEXTURE );
if (m_enumFancyEdge == FancyEdgeColor || m_enumFancyEdge ==
FancyEdgeColors)
{
    pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
    D3DTA_TFACTOR );
    // RenderState TextureFactorからRGB値を得る。
}
pd3dDevice->SetTextureStageState( 0,
D3DTSS_ALPHAOP, D3DTOP_SELECTARG1 );
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAARG1,
D3DTA_CURRENT );
```

```
// ステージ１：
// ColorOp : Current
// AlphaOp : Select Texture
pd3dDevice->SetTexture(1, m_pAlphaMapTexture);
pd3dDevice->SetTextureStageState( 1, D3DTSS_TEXCOORDINDEX , 1 );
pd3dDevice->SetTextureStageState( 1,
D3DTSS_COLOROP, D3DTOP_SELECTARG1 );
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLORARG1,
D3DTA_CURRENT );
pd3dDevice->SetTextureStageState( 1,
D3DTSS_ALPHAOP, D3DTOP_SELECTARG1 );
pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAARG1,
D3DTA_TEXTURE );

// ステージ２：ディスエーブル
pd3dDevice->SetTextureStageState( 2,
D3DTSS_ALPHAOP, D3DTOP_DISABLE );
pd3dDevice->SetTextureStageState( 2,
D3DTSS_COLOROP, D3DTOP_DISABLE );

pd3dDevice->SetSamplerState( 0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR
);
pd3dDevice->SetSamplerState( 0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR
);
pd3dDevice->SetSamplerState( 1, D3DSAMP_MINFILTER, D3DTEXF_LINEAR
);
pd3dDevice->SetSamplerState( 1, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR
);

pd3dDevice->SetSamplerState(1, D3DSAMP_ADDRESSU,
D3DTADDRESS_CLAMP );
pd3dDevice->SetSamplerState(1, D3DSAMP_ADDRESSV,
D3DTADDRESS_CLAMP );
if (m_enumFancyEdge == FancyEdgeColor || m_enumFancyEdge ==
FancyEdgeColors)
{
     if(m_enumFancyEdge == FancyEdgeColors)
     {
          D3DXCOLOR clrStart( m_colorFancyEdgeStart );
          D3DXCOLOR clrStop( m_colorFancyEdgeStop );
          D3DXCOLOR clrOut( (DWORD)0 );

          D3DXColorLerp(&clrOut, &clrStart, &clrStop,
          (FLOAT)fTime);

          D3DCOLOR clr =
          D3DCOLOR_COLORVALUE(clrOut.r,clrOut.g,clrOut.b,clrOut.a
          );
          pd3dDevice->SetRenderState( D3DRS_TEXTUREFACTOR, clr );
```

```
    }
    else
    {
        pd3dDevice->SetRenderState( D3DRS_TEXTUREFACTOR,
        m_colorFancyEdgeStart );
    }
}
else
{
    pd3dDevice->SetTexture(0, m_pFancyEdgeTexture);
}

pPatternMesh->DrawSubset( 0 ); // D3D/GPUにこのPatternMesh、
// TextureOut、および現在のセッティングを使用する描画を
// 実行させる。
```

【００８６】

（セクション７：ファンシイエッジサンプルコード）

```
FLOAT fltShift = m_fltGradientWidth;
if(m_fltGradientWidth == 0.0f)
{
     fltShift = 0.05f;
}

fltProgress = (FLOAT)fTime;

if(m_fProgressShift)
{
     if(fltProgress > fltShift)
     {
          fltProgress = fltShift + fltProgress;
     }
     else
     {
          fltProgress *= 2.0f;
     }
}
```

【００８７】

一実施形態で、トランジションパターンメッシュは、シャープ／ソフト／ディゾルブエッジ技法と正確に同一のプロセスを介して２回レンダリングされる。第１パスでは、炎のイメージまたは波のイメージとすることができるファンシイエッジテクスチャをレンダリングする。第２パスでは、上のソースコードを用いてシフトされた進行を用いて、出てゆくビデオソースをレンダリングする。

【００８８】

（セクション８：ディゾルブエッジサンプルコード）

```
// 入ってくるビデオソースを最初に描画する。

// Direct3D状態をセットアップする。

// pd3dDeviceは、Direct3D/GPUのインスタンスポインタである。
```

```
pd3dDevice->SetRenderState( D3DRS_ALPHATESTENABLE, FALSE);
// アルファテストを実行する必要はない。

pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, FALSE);
// アルファブレンディングを実行する必要はない。

pd3dDevice->SetRenderState( D3DRS_STENCILENABLE, FALSE);
// ステンシルテストを実行する必要はない。
pd3dDevice->SetRenderState( D3DRS_ZENABLE, D3DZB_FALSE);
// デプステストの必要はない。
pd3dDevice->SetRenderState( D3DRS_ZWRITEENABLE, FALSE);
// Z値をデプスバッファに保存する必要はない。

// Direct3Dテクスチャブレンディング状態をセットアップする。
// Direct3Dは８つまでの状態を許容するが、１つだけが必要である。

// pTextureInの(tu1, tv1)から各頂点のRGB値をルックアップする。
pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 0 );
// t0テクスチャ座標をステージ０に使用する。
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);  // RGB値を出力する。

pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
D3DTA_TEXTURE );// テクスチャからRGB値を出力する

pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAOP,
D3DTOP_DISABLE );
// ステージ０以上からのアルファ出力はない。
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
D3DTOP_DISABLE );

// ステージ１以上からのRGB出力はない。
// 入ってくるソースを描画する。
// テクスチャ２および第１メッシュを使用する。

pd3dDevice->SetTexture(0, pTextureIn);
// pTextureInは、入ってくるビデオフレームを保管するテクスチャ
// オブジェクトである。このテクスチャをテクスチャブレンディン
// グステージ０にセットする。

LPD3DXMESH pPlaneMesh = m_pPlaneMesh->GetLocalMesh();
// デバイス/GPUメモリに保管されたD3DXMeshオブジェクトを
// CDXUTMeshから得る。

pPlaneMesh->DrawSubset( 0 ); // D3D/GPUにこのPlaneMesh、
// TextureIn、および現在のセッティングを使用する描画を
// 実行させる。

// 出てゆくソースを描画する。
```

```
// トランジションパターンメッシュの各頂点のtv2をProgressにセットする。
VERTEX_XYZ_TEX2* pVertices = NULL;
pSysMemMesh->LockVertexBuffer( D3DLOCK_NOSYSLOCK,
(void**)&pVertices );

DWORD dwNumPatternVertices = pSysMemMesh->GetNumVertices();
for(DWORD i = 0; i < dwNumPatternVertices; i++)
  pVertices[i].t2.y = fltProgress; // Progress

// fltProgressは、トランジション持続時間、フレームレート、
// および現在のフレームに基づいて計算される。

pSysMemMesh->UnlockVertexBuffer();
m_pMesh->InvalidateDeviceObjects();
m_pMesh->RestoreDeviceObjects(pd3dDevice);

// RestoreDeviceObjects()は、ローカルメッシュを作成する。

// ディゾルブ効果技法は、PatternMeshを２回レンダリングすることを
// 必要とする。第１パスでは、ディゾルブマップおよびアルファマップ
// からのアルファ値を加算する。この値を、AlphaTest動作で保持され
// た事前定義の閾値と比較する。ピクセルについて増やされたステンシ
// ル値は、テストに合格する。RGB値は、レンダターゲットに書き込ま
// れない。

// src=zeroおよびdest=oneのアルファブレンディングを使用して、カラ
// ーバッファへの書き込みを防ぐ。

pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, TRUE);
pd3dDevice->SetRenderState( D3DRS_SRCBLEND,
D3DBLEND_ZERO);
pd3dDevice->SetRenderState( D3DRS_DESTBLEND,
D3DBLEND_ONE);

// デプスバッファリングをディスエーブルする。
pd3dDevice->SetRenderState( D3DRS_ZENABLE,      D3DZB_FALSE);
pd3dDevice->SetRenderState( D3DRS_ZWRITEENABLE, FALSE);

// アルファテストに合格したピクセルのステンシルを増やすために、
// ステンシル関数をセットアップする。
pd3dDevice->SetRenderState( D3DRS_STENCILPASS,
D3DSTENCILOP_INCR);
pd3dDevice->SetRenderState( D3DRS_STENCILENABLE, TRUE);
pd3dDevice->SetRenderState( D3DRS_STENCILFUNC,   D3DCMP_ALWAYS);

// アルファテストをセットアップする。
pd3dDevice->SetRenderState( D3DRS_ALPHATESTENABLE,TRUE);
pd3dDevice->SetRenderState( D3DRS_ALPHAFUNC,D3DCMP_GREATER);
pd3dDevice->SetRenderState( D3DRS_ALPHAREF, 0x40);
```

```
// テクスチャブレンディング
// ステージ０：
// ColorOp : Current
// AlphaOp : Select Texture
RTN_HR_IF_FAILED(pd3dDevice->SetTexture(0, m_pAlphaMapTexture));
pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 1 );
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
D3DTA_CURRENT );
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAOP,
D3DTOP_SELECTARG1
);
pd3dDevice->SetTextureStageState( 0, D3DTSS_ALPHAARG1,
D3DTA_TEXTURE );

// ステージ１：
// ColorOp : Current
// AlphaOp : Add - Signed
RTN_HR_IF_FAILED(pd3dDevice->SetTexture(1,
m_pDissolveMapTexture));
pd3dDevice->SetTextureStageState( 1, D3DTSS_TEXCOORDINDEX , 0 );
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLORARG1,
D3DTA_CURRENT );
// ２つのアルファテクスチャからのアルファ値を加算する。
pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAOP,
D3DTOP_ADDSIGNED);
pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAARG1,
D3DTA_TEXTURE );

// ステージ２：ディスエーブル
pd3dDevice->SetTextureStageState( 2, D3DTSS_ALPHAOP,
D3DTOP_DISABLE );
pd3dDevice->SetTextureStageState( 2, D3DTSS_COLOROP,
D3DTOP_DISABLE );

pd3dDevice->SetSamplerState( 0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR
);
pd3dDevice->SetSamplerState( 0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR
);

pd3dDevice->SetSamplerState( 1, D3DSAMP_MINFILTER, D3DTEXF_POINT
);
pd3dDevice->SetSamplerState( 1, D3DSAMP_MAGFILTER, D3DTEXF_POINT
);

pd3dDevice->SetSamplerState(0, D3DSAMP_ADDRESSU,
D3DTADDRESS_CLAMP );
```

```
pd3dDevice->SetSamplerState(0, D3DSAMP_ADDRESSV,
D3DTADDRESS_CLAMP );

pd3dDevice->SetSamplerState(1, D3DSAMP_ADDRESSU,
D3DTADDRESS_CLAMP );
pd3dDevice->SetSamplerState(1, D3DSAMP_ADDRESSV,
D3DTADDRESS_CLAMP );

LPD3DXMESH pPatternMesh = m_pPatternMesh->GetLocalMesh();
pPatternMesh->DrawSubset( 0 ); // D3D/GPUにこのPatternMesh
// および現在のセッティングを使用する描画を実行させる。

// 第２パス。デプステストおよびステンシルテストを実行する。
// 両方のテストに合格したピクセルだけがレンダターゲットに描画される。
// ステンシルテストは、ワイプエッジに沿ったランダムパターンを形成する。
// デプステストは、ピクセルがエッジの間違った側に現れないようにする。

// レンダビデオ１のレンダ状態をセットアップする。
// アルファブレンディングおよびアルファテストをディスエーブルする。
pd3dDevice->SetRenderState( D3DRS_ALPHATESTENABLE, FALSE);
pd3dDevice->SetRenderState( D3DRS_ALPHABLENDENABLE, FALSE);

// 第１シーンのレンダリングのためにステンシル状態をセットアップする。
pd3dDevice->SetRenderState( D3DRS_STENCILENABLE, TRUE);
pd3dDevice->SetRenderState( D3DRS_STENCILPASS,
D3DSTENCILOP_KEEP);
pd3dDevice->SetRenderState( D3DRS_STENCILREF,  1);
pd3dDevice->SetRenderState( D3DRS_STENCILFUNC, D3DCMP_EQUAL);

pd3dDevice->SetRenderState( D3DRS_ZENABLE,      D3DZB_TRUE);
pd3dDevice->SetRenderState( D3DRS_ZWRITEENABLE, TRUE);

pd3dDevice->SetTextureStageState( 0, D3DTSS_TEXCOORDINDEX , 0 );
pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,
D3DTOP_SELECTARG1);

pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1,
D3DTA_TEXTURE );
pd3dDevice->SetTextureStageState( 1, D3DTSS_ALPHAOP,
D3DTOP_DISABLE );
pd3dDevice->SetTextureStageState( 1, D3DTSS_COLOROP,
D3DTOP_DISABLE );

pd3dDevice->SetSamplerState( 0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR
);
pd3dDevice->SetSamplerState( 0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR
);

// テクスチャ１およびトランジションパターンメッシュを使用する。
pd3dDevice->SetTexture(0, pTextureOut);
```

```
// pTextureOutは、出てゆくビデオフレームを保管するテクスチャ
// オブジェクトである。このテクスチャをテクスチャブレンディン
// グステージ０にセットする。

// 並進変換行列を適用して、トランジションパターンメッシュをｚ方向の
// 小さい方に移動する。

D3DXMATRIX matTranslation;
D3DXMatrixTranslation( &matTranslation, 0, 0, -fltProgress +
m_fltGradientWidth);

// 並進変換行列を作成して、PatternMeshをｚ方向の小さい方に移動する。
pd3dDevice->SetTransform( D3DTS_WORLD, &matTranslation );

// D3Dワールド変換行列をセットする。
// D3Dワールド変換行列はモデル空間座標をワールド空間に変換する。
// トランジションパターンメッシュの(0, 0, 0)は、(0, 0, -
// fltProgress + m_fltGradientWidth)になる。
pPatternMesh->DrawSubset( 0 ); // D3D/GPUにこのPatternMesh、
// TextureOut、および現在のセッティングを使用する描画を
// 実行させる。
```

【図面の簡単な説明】

【００８９】

【図１】本発明の一実施形態による第１メディアオブジェクトと第２メディアオブジェクトの間の２次元トランジションを生成するシステムを示すブロック図である。

【図２Ａ】本発明の一実施形態による３次元モデルを示す図である。

【図２Ｂ】本発明の一実施形態による正射影変換を示す図である。

【図２Ｃ】本発明の一実施形態による異なる次元でのトランジションパターンのビューを示す図である。

【図３】Ａ～Ｆに、本発明の一実施形態によるトランジションパターンメッシュの実装を示す図である。

【図４Ａ】本発明の一実施形態による３次元モデルを使用するサークルワイプトランジションパターンの実装を示す図である。

【図４Ｂ】本発明の一実施形態による３次元モデルを使用するサークルワイプトランジションパターンの実装を示す図である。

【図４Ｃ】本発明の一実施形態による３次元モデルを使用するサークルワイプトランジションパターンの実装を示す図である。

【図５Ａ】本発明の一実施形態による例示的なトランジションパターンの実装を示す図である。

【図５Ｂ】本発明の一実施形態による例示的なトランジションパターンの実装を示す図である。

【図６】本発明の一実施形態による、トランジションパターンの鋭いエッジを有する、第１メディアオブジェクトと第２メディアオブジェクトの間の２次元トランジションを生成する方法を示す流れ図である。

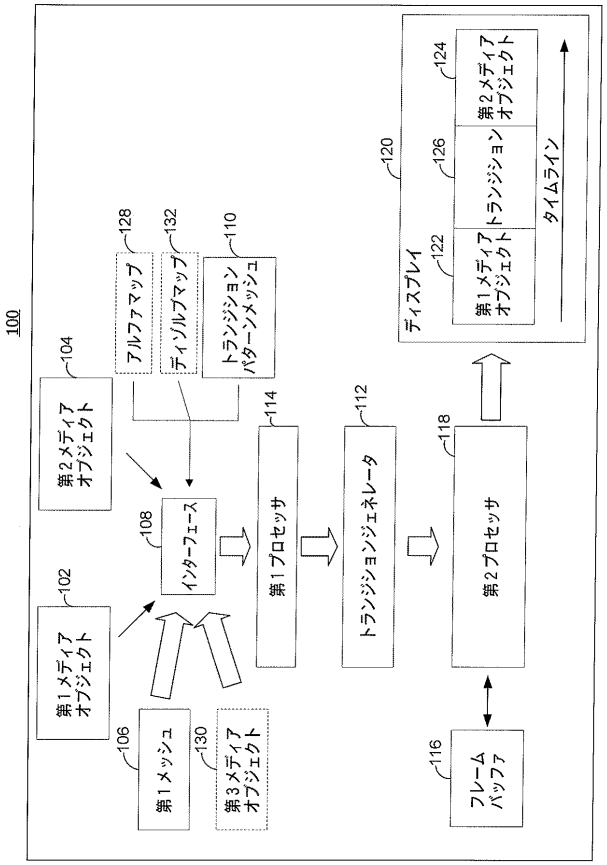
【図７】本発明の一実施形態による、第１メディアオブジェクトと第２メディアオブジェクトの間のトランジションに対する追加効果を作成する方法を示す流れ図である。

【図８】本発明を実装できる適切なコンピューティングシステム環境の一例を示すブロック図である。

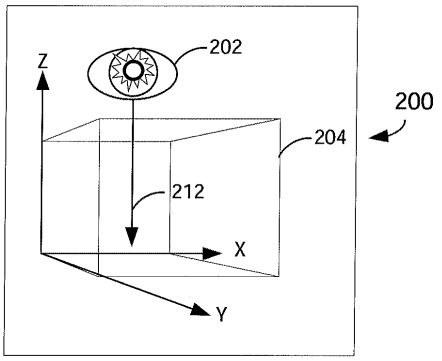
【図９】Ａ～Ｃに、本発明の一実施形態によるアルファマップの例を示す図である。

【図１０】本発明の一実施形態によるアルファマップの例を示す図である。

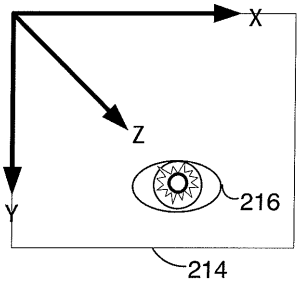
【図１】



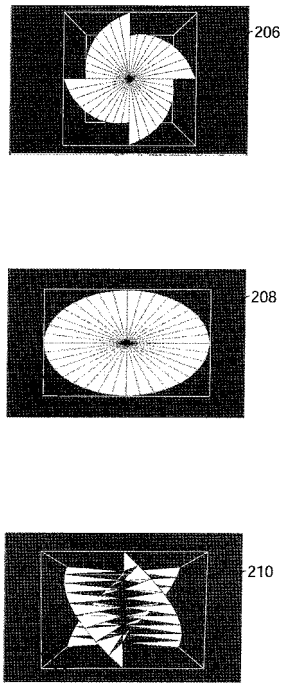
【図２Ａ】



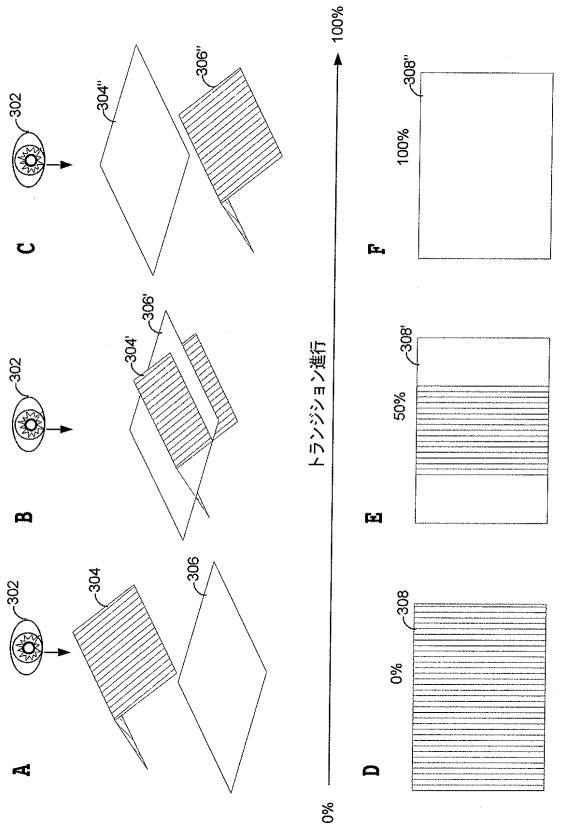
【図２Ｂ】



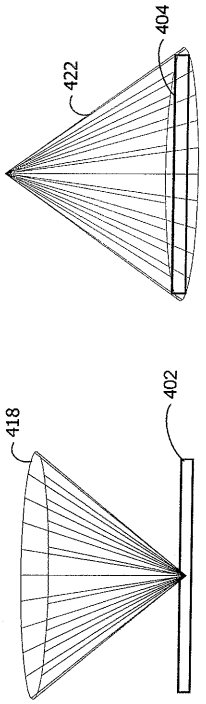
【図２Ｃ】



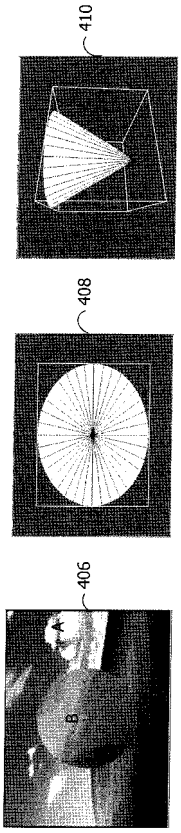
【図３】



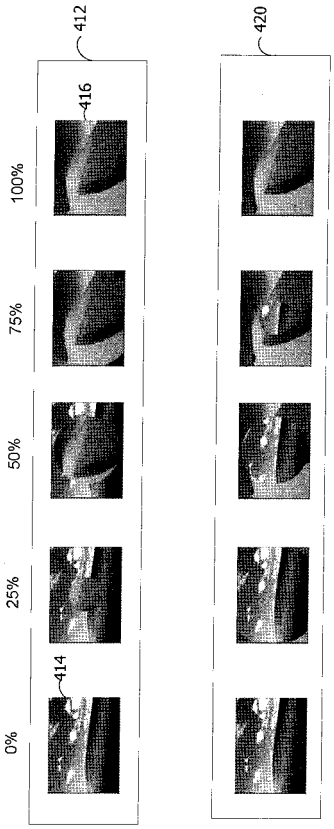
【図４Ａ】



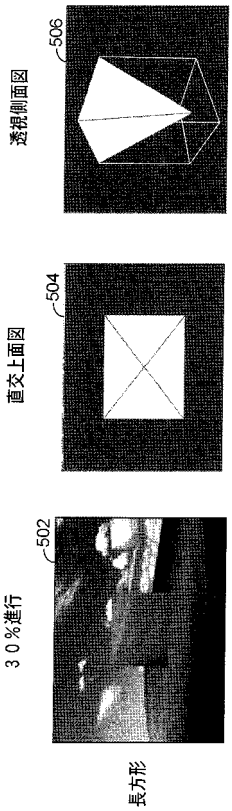
【図４Ｂ】



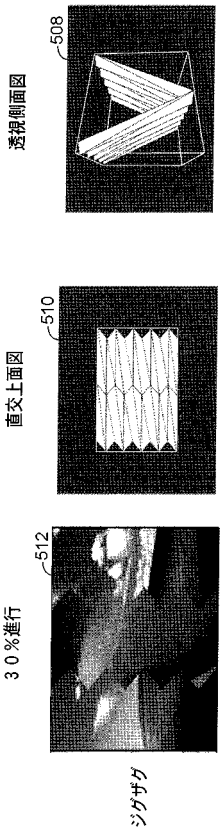
【図４Ｃ】



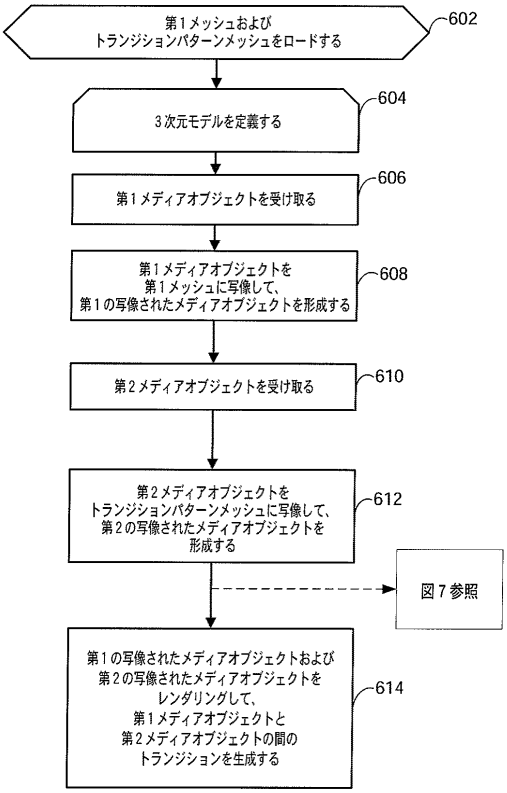
【図５Ａ】



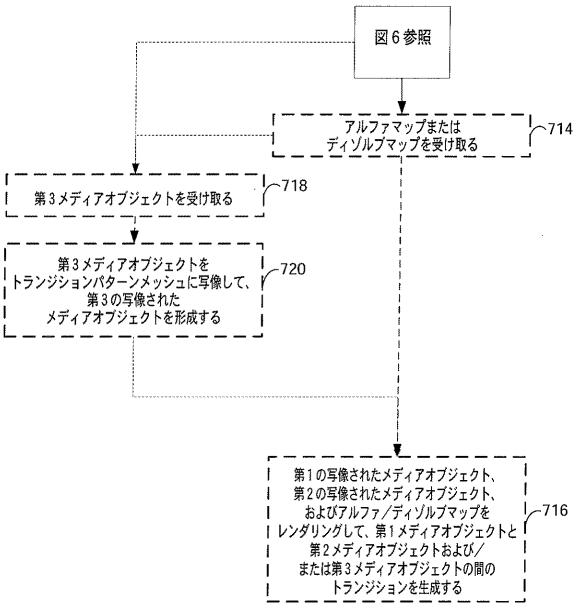
【図５Ｂ】


透視側面図
508
直交上面図
510
３０％進行
512
ジグザグ

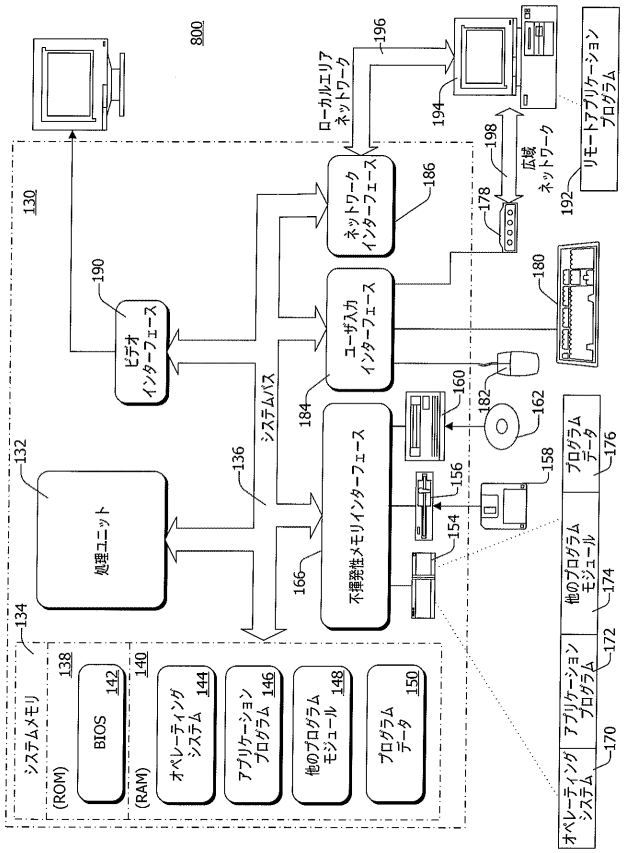
【図６】


第１メッシュおよび
トランジションパターンメッシュをロードする
602
３次元モデルを定義する
604
第１メディアオブジェクトを受け取る
606
第１メディアオブジェクトを
第１メッシュに写像して、
第１の写像されたメディアオブジェクトを形成する
608
第２メディアオブジェクトを受け取る
610
第２メディアオブジェクトを
トランジションパターンメッシュに写像して、
第２の写像されたメディアオブジェクトを
形成する
612
図７参照
第１の写像されたメディアオブジェクトおよび
第２の写像されたメディアオブジェクトを
レンダリングして、
第１メディアオブジェクトと
第２メディアオブジェクトの間の
トランジションを生成する
614

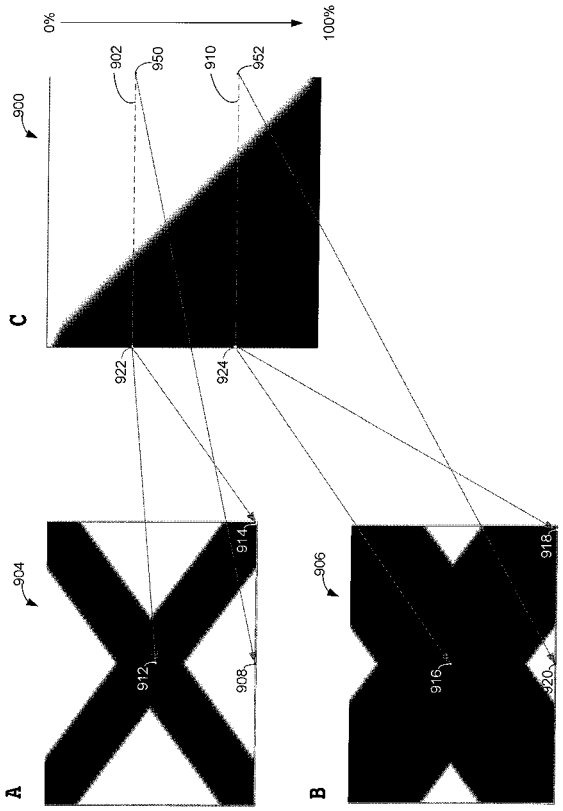
【図７】


図６参照
アルファマップまたは
ディゾルブマップを受け取る
714
第３メディアオブジェクトを受け取る
718
第３メディアオブジェクトを
トランジションパターンメッシュに写像して、
第３の写像された
メディアオブジェクトを形成する
720
第１の写像されたメディアオブジェクト、
第２の写像されたメディアオブジェクト、
およびアルファ／ディゾルブマップを
レンダリングして、第１メディアオブジェクトと
第２メディアオブジェクトおよび／
または第３メディアオブジェクトの間の
トランジションを生成する
716

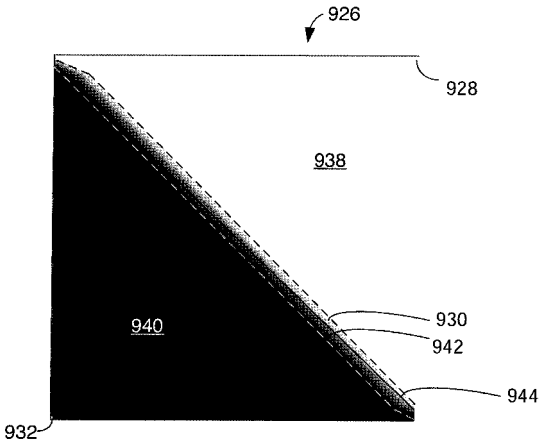
【図８】


800
ローカルエリア
ネットワーク
196
194
198
広域
ネットワーク
リモートアプリケーション
プログラム
192
130
ネットワーク
インターフェース
186
178
180
ビデオ
インターフェース
190
ユーザ入力
インターフェース
160
182
162
システムバス
184
プログラム
データ
176
132
処理ユニット
136
不揮発性メモリインターフェース
156
158
154
他のプログラム
モジュール
174
166
134
アプリケーション
プログラム
172
システムメモリ
138
(ROM)
BIOS
142
140
(RAM)
オペレーティング
システム
144
アプリケーション
プログラム
146
他のプログラム
モジュール
148
プログラム
データ
150
オペレーティング
システム
170

【図９】


0%
100%
900
902
950
910
952
C
922
924
904
914
912
908
A
906
918
916
920
B

【図１０】


926
928
938
940
930
942
944
932

フロントページの続き

(72)発明者　ウォーレン　エル．バーチ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

Ｆターム(参考) 5B050 BA08 BA09 BA18 EA19 EA24 EA28 FA02
5B080 AA14 BA04 CA01 FA01 FA03 FA08 GA02 GA22