(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4852674号
(P4852674)
(45)発行日　平成24年1月11日(2012.1.11)　(24)登録日　平成23年11月4日(2011.11.4)

(51)Int.Cl.　F I
G06F 13/00 (2006.01)　G06F 13/00　550L

請求項の数 50　(全 47 頁)

(21)出願番号	特願2004-523286 (P2004-523286)	(73)特許権者	511158959
(86)(22)出願日	平成15年7月21日 (2003.7.21)		シニヴァース　アイシーエックス　コーポレイション
(65)公表番号	特表2005-534108 (P2005-534108A)		アメリカ合衆国　フロリダ州　タンパ　ハイウッズ　パーム　ウェイ　８１２５
(43)公表日	平成17年11月10日 (2005.11.10)	(74)代理人	100092093
(86)国際出願番号	PCT/US2003/022888		弁理士　辻居　幸一
(87)国際公開番号	WO2004/010341	(74)代理人	100082005
(87)国際公開日	平成16年1月29日 (2004.1.29)		弁理士　熊倉　禎男
審査請求日	平成18年7月11日 (2006.7.11)	(74)代理人	100067013
審判番号	不服2009-11055 (P2009-11055/J1)		弁理士　大塚　文昭
審判請求日	平成21年6月15日 (2009.6.15)	(74)代理人	100086771
(31)優先権主張番号	60/398,211		弁理士　西島　孝喜
(32)優先日	平成14年7月23日 (2002.7.23)	(74)代理人	100109070
(33)優先権主張国	米国(US)		弁理士　須田　洋之
(31)優先権主張番号	10/273,670		
(32)優先日	平成14年10月18日 (2002.10.18)		
(33)優先権主張国	米国(US)		

最終頁に続く

(54)【発明の名称】特定のクライアント装置タイプに対して最適化された動的ビューポート階層化を提供する画像システム

(57)【特許請求の範囲】

【請求項１】

クライアント装置に伝送された画像の表示を動的に最適化する方法であって、

表示のためのターゲット画像を検索するための特定クライアント装置からのオンライン要求を受信する段階、

を含み、

前記要求は、前記クライアント装置に対する装置タイプの判断を助ける情報を含み、

前記ターゲット画像は、個々の層内に配置された画像成分を含み、

前記要求に基づいて、前記特定クライアント装置に対する装置タイプを判断する段階と、

前記判断された装置タイプに基づいて、前記特定クライアント装置に対するビューポート及び階層化情報を指定する情報を検索する段階と、

前記ビューポート及び階層化情報に基づいて、前記特定クライアント装置における表示のために最適化された前記ターゲット画像のバージョンを作成する段階であって、前記特定クライアント装置における表示のために最適化された前記ターゲット画像のバージョンを作成する段階が、前記クライアント装置における表示のために最適化された適当なターゲット画像が見つかるまで、前記ビューポート及び階層化情報に基づいて、前記ターゲット画像の異なるバージョンを反復して作成する段階を含んでおり、

前記ターゲット画像の前記作成されたバージョンを表示のために前記クライアント装置に伝送する段階と、

を更に含むことを特徴とする方法。
【請求項２】
前記装置タイプを判断する段階は、
前記クライアント装置に対する装置タイプを判断するために装置データベースを調べる段階、
を含むことを特徴とする請求項１に記載の方法。
【請求項３】
前記装置データベースを調べる段階は、
前記特定クライアント装置に対する画像の表示を最適化するのに利用可能な特定の構成ファイルを指示する情報を前記装置データベースから検索する段階、
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】
前記特定の構成ファイルは、前記特定クライアント装置に関する表示機能情報を指定することを特徴とする請求項３に記載の方法。
【請求項５】
前記特定の構成ファイルは、前記特定クライアント装置に対するビューポート及び階層化情報を指定するＸＭＬファイルを含むことを特徴とする請求項３に記載の方法。
【請求項６】
前記オンライン要求は、ウェブベースの要求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】
前記オンライン要求は、ＨＴＭＬベースの要求を含むことを特徴とする請求項１に記載の方法。
【請求項８】
前記オンライン要求は、特定ターゲット画像を識別する情報を含むことを特徴とする請求項１に記載の方法。
【請求項９】
前記オンライン要求は、前記特定クライアント装置で作動するブラウザから開始され、特定タイプのクライアント装置を識別する情報を含むことを特徴とする請求項１に記載の方法。
【請求項１０】
前記オンライン要求は、ＵＲＬを含むことを特徴とする請求項１に記載の方法。
【請求項１１】
前記特定クライアント装置の装置タイプの判断を助ける情報は、前記オンライン要求と共に伝送されたヘッダ情報を含むことを特徴とする請求項１に記載の方法。
【請求項１２】
前記オンライン要求と共に伝送された前記ヘッダ情報は、前記特定クライアント装置がどの特定タイプの装置であるかを判断するために装置タイプのデータベースに照らして比較されることを特徴とする請求項１１に記載の方法。
【請求項１３】
前記特定クライアント装置に対する装置タイプの判断を助ける前記情報は、ＨＴＴＰヘッダ情報を含むことを特徴とする請求項１に記載の方法。
【請求項１４】
前記特定クライアント装置に対するビューポート及び階層化情報を指定する前記情報は、ＸＭＬファイルに階層的に維持されることを特徴とする請求項１に記載の方法。
【請求項１５】
前記ＸＭＬファイルは、前記特定クライアント装置の装置限界に基づく制限値を含むことを特徴とする請求項１４に記載の方法。
【請求項１６】
前記制限値は、前記特定クライアント装置で表示することができる最大画像サイズを示

すことを特徴とする請求項１５に記載の方法。
【請求項１７】
前記クライアント装置における表示のために最適化されたターゲット画像のバージョンを作成する段階は、
前記特定クライアント装置に対する前記ビューポート及び階層化情報に適合する方法で個々の層の前記画像成分をレンダリングする段階、
を含むことを特徴とする請求項１に記載の方法。
【請求項１８】
前記クライアント装置における表示のために最適化されたターゲット画像のバージョンを作成する段階は、
前記クライアント装置における表示のために最適化された適切なターゲット画像が見つかるまで、前記ビューポート及び階層化情報に基づいて前記ターゲット画像の異なるバージョンを反復的に作成する段階、
を含むことを特徴とする請求項１に記載の方法。
【請求項１９】
前記ターゲット画像の前記作成されたバージョンを前記特定クライアント装置に伝送する前に、該作成されたバージョンを該クライアント装置に適切なファイルフォーマットに変換する段階、
を更に含むことを特徴とする請求項１に記載の方法。
【請求項２０】
前記ファイルフォーマットは、ＪＰＥＧ互換ファイルフォーマットを含むことを特徴とする請求項１９に記載の方法。
【請求項２１】
前記層の各々は、所定の種類の画像成分を維持することを特徴とする請求項１に記載の方法。
【請求項２２】
所定の層は、ビットマップ、アニメーション、テキスト、及びベクトルグラフィックの１つから選択された画像成分を維持することを特徴とする請求項２１に記載の方法。
【請求項２３】
請求項１に記載の方法を実行するためのプロセッサ実行可能命令を有するコンピュータ可読媒体。
【請求項２４】
請求項１に記載の方法を実行するための一組のダウンロード可能なプロセッサ実行可能命令。
【請求項２５】
特定の装置タイプに対してカスタマイズされた画像のオンデマンド作成のためのシステムであって、
各画像が個別の層内に配置された画像成分を含む、画像のためのリピジトリーの役目をするモジュールと、
前記リピジトリーから特定の画像を検索するための装置からの要求を処理するためのモジュールと、
を含み、
該モジュールは、前記要求に含まれた情報に部分的に基づいて前記装置に対する特定の装置タイプを判断し、
前記装置に対してカスタマイズされた前記特定画像のコピーを作成するためのモジュール、
を更に含み、
該モジュールは、前記特定画像の前記個別の層内の前記画像成分の少なくともいくつかが前記装置に対してカスタマイズされるように、前記判断された装置タイプに基づいて該特定画像の該個別の層内の画像成分を個々にレンダリングするようになっており、

コピーを作成するための前記モジュールは、前記装置において表示するのに最適化された適当なものが作成されるまで、前記特定画像の異なる候補を反復的に作成する、
ことを特徴とするシステム。
【請求項２６】
前記特定画像のコピーは、前記装置のビューポート制限に適合する方法で作成されることを特徴とする請求項２５に記載のシステム。
【請求項２７】
前記特定画像のコピーは、前記装置の画像サイズ制限に適合する方法で作成されることを特徴とする請求項２５に記載のシステム。
【請求項２８】
前記特定画像のコピーを前記装置に適合する画像フォーマットに変換するためのモジュール、
を更に含むことを特徴とする請求項２５に記載のシステム。
【請求項２９】
前記要求を処理するためのモジュールは、サーバコンピュータ上で作動することを特徴とする請求項２５に記載のシステム。
【請求項３０】
前記サーバコンピュータは、インターネット接続を含むことを特徴とする請求項２９に記載のシステム。
【請求項３１】
無線接続を通じて前記装置と通信することを特徴とする請求項２５に記載のシステム。
【請求項３２】
インターネットへの無線接続を通じて前記装置と通信することを特徴とする請求項２５に記載のシステム。
【請求項３３】
前記装置に対してカスタマイズされた前記特定の画像のコピーをキャッシュに入れるための画像キャッシュを更に含むことを特徴とする請求項２５に記載のシステム。
【請求項３４】
前記要求は、ＨＴＭＬ要求を含むことを特徴とする請求項２５に記載のシステム。
【請求項３５】
前記要求は、前記装置において作動するブラウザから開始されることを特徴とする請求項２５に記載のシステム。
【請求項３６】
前記要求は、装置タイプの判断を可能にするヘッダ情報を含むことを特徴とする請求項２５に記載のシステム。
【請求項３７】
前記ヘッダ情報に基づいて前記装置タイプの識別を助けるための装置データベース、
を更に含むことを特徴とする請求項３６に記載のシステム。
【請求項３８】
前記装置データベースは、前記装置に対してカスタマイズされた前記特定画像のコピーを作成するのに有用な利用可能な構成ファイルを指示することを特徴とする請求項３７に記載のシステム。
【請求項３９】
前記構成ファイルは、前記装置に対する階層化及びビューポート情報を指定するＸＭＬファイルを含むことを特徴とする請求項２５に記載のシステム。
【請求項４０】
前記コピーを反復的に作成するためのモジュールは、前記装置における表示に対して最適化された適切なものが作成されるまで、前記特定画像の異なる候補を作成することを特徴とする請求項２５に記載のシステム。
【請求項４１】

要求された画像の表示を最適化するための改良を含む画像検索方法であって、
前記改良は、
各層がある一定の種類の画像成分を有する異なる層に各画像を編成する段階と、
所定の画像が表示される特定タイプの装置に対して所定の層を最適化する方法を示す情報を記憶する段階と、
特定画像を検索するための要求を受信した時に、どのタイプの装置が該特定画像を要求しているかを特定する段階と、
前記特定画像を要求している前記装置のタイプに基づいて、該画像を要求している該装置に対する該画像の所定の層を最適化する方法を示す前記記憶した情報を検索する段階と、
前記検索した情報に基づいて、前記装置における表示に対して最適化されたレンダリングされた画像を動的に生成するために、前記画像の個々の層をレンダリングする段階と、および、
動的に生成された前記レンダリング画像が前記装置が要求する前記特定画像よりも大きすぎる画像サイズを有している場合には、前記装置における表示に対して適当なサイズのレンダリング画像が得られるまで、より小さい画像サイズを有するレンダリング画像を生成するために前記画像の個々の層を反復的に再度レンダリングする段階と、
を含むことを特徴とする方法。

【請求項４２】
前記レンダリングする段階を前記装置に適切な制限に適合する方法で実行することができるように、異なるタイプの装置に対するビューポート情報を維持する段階、
を更に含むことを特徴とする請求項４１に記載の改良。

【請求項４３】
前記動的に生成されたレンダリング画像が、前記特定画像を要求している前記装置に対して大きすぎる画像サイズを有する場合、前記レンダリングする段階は、より小さな画像サイズを有する画像を生成するために繰り返されることを特徴とする請求項４１に記載の改良。

【請求項４４】
所定の層は、ビットマップ、アニメーション、テキスト、及びベクトルグラフィックの１つから選択された画像成分を維持することを特徴とする請求項４１に記載の改良。

【請求項４５】
１つの層は、境界をレンダリングするための専用であることを特徴とする請求項４４に記載の改良。

【請求項４６】
前記特定タイプの装置に対して各層を最適化する方法を示す情報は、装置タイプ特定の構成ファイルに記憶されることを特徴とする請求項４１に記載の改良。

【請求項４７】
前記装置タイプ特定の構成ファイルは、特定の装置タイプに対する画像のレンダリングに関する情報を各ファイルが記憶するＸＭＬファイルを含むことを特徴とする請求項４６に記載の改良。

【請求項４８】
装置タイプ特定の構成ファイルの各々は、特定の装置タイプに対する階層化及びビューポート情報を含むことを特徴とする請求項４６に記載の改良。

【請求項４９】
前記要求は、インターネットに接続した装置から受信したブラウザ要求を含むことを特徴とする請求項４１に記載の改良。

【請求項５０】
前記どのタイプの装置が特定画像を要求しているかを特定する段階は、該装置のタイプを識別するのを助ける情報を得るために該要求を構文解析する段階を含むことを特徴とする請求項４１に記載の改良。

【発明の詳細な説明】
【技術分野】
【０００１】
著作権に関する注意
本特許の開示の一部分は、著作権保護を受ける材料を含む。本著作権所有者は、特許商標局の特許ファイル又は記録にあるような特許文書又は特許開示の何人のファクシミリ複製にも異議を唱えないが、それ以外は、全ての著作権を保有する。
本発明は、一般的にデジタル画像処理、及び、より具体的には、異なる装置上のデジタル画像をレンダリングするための改良型技術に関する。
【背景技術】
【０００２】
今日、特にデジタルカメラという形式のデジタル画像は、これまでのフィルムではなく半導体画像センサを使用して写真を捕捉する新しい方法を提供する広く行き渡った現実である。デジタルカメラは、何らかの種類の感知機構で入射光を記録し、その後その情報を処理して（基本的に、アナログデジタル変換を通じて）ターゲット像のメモリ画像を作成することにより機能する。デジタルカメラの最大の利点は、デジタルで画像を作成することにより、あらゆる種類の装置及びアプリケーション間で画像を転送することを容易にする点である。例えば、簡単にデジタル画像をワープロ文書内に挿入し、電子メールで友達に送り、又は世界の誰かが見ることができるウェブサイト上に送ることができる。更に、写真編集ソフトウエアを使用してデジタル画像を操作し、それらを改良するか又は変更することができる。例えば、それらに切り取りを行って赤い目を取り除き、色又はコントラクタを変更し、要素を追加又は削除することさえできる。また、デジタルカメラでは、自分の画像に即座にアクセスすることができ、その結果、フィルム処理の煩わしさや遅延が回避される。総じて、デジタル画像は、画像を使用するか又は配信したい時にユーザに柔軟性を与えるために、益々一般的になっている。
【０００３】
デジタル画像は、その出所に関係なくユーザによって操作されることが多い。例えば、ユーザは、「Ａｄｏｐｅ　Ｐｈｏｔｏｓｈｏｐ」をデスクトップコンピュータ上で使用し、お互いの上に異なるオブジェクトを階層化することによって画像をマニュアルで作成することができる。例えば、画像の１つの層は、アートワークを含むことができ、別の層は、テキストを含むことができ、更に別の層は、ビットマップボーダを含むことができ、以下同様である。その後、その別々の層を有する画像は、「Ｐｈｏｔｏｓｈｏｐ」（固有）ファイルフォーマットで保存するか又は様々な異なるファイルフォーマットの１つで保存することができる。
【０００４】
「Ｐｈｏｔｏｓｈｏｐ」を使用して、世界の可能な（ディスプレイ使用可能）装置の各々に対して画像が正しくレンダリングされるように、所定画像の異なるバージョンを恐らく予め生成する（すなわち、画像の異なる層を予めレンダリングする）ことができるであろう。しかし、この手法は、実際に現実的なものではない。様々な装置には、ファイルサイズに関する制限（例えば、５Ｋバイト未満）、ビット深さの制限（例えば、ピクセル当たり８ビットを超えない）、及び画像サイズの制限（例えば、画像は、１００ｘ１００ピクセルを超えることができない）がある。従って、何千もの装置に対して満足できる画像バージョンを作成する作業は非現実的である。
【０００５】
例えば、ＪＰＥＧを表示することができるターゲット装置上に表示するために、アートワーク（例えば、ビットマップ背景）の上にキャラクター（例えば、ディズニーキャラクター）を階層化する作業を考えてみる。この場合、アートワークは、ターゲット装置の画面サイズにサイズ変更する必要があるであろう。その後、キャラクターは、サイズ変更したアートワークの上に重ねる（階層化する）必要があり、最後に、画像は、正しいＪＰＥＧ品質によって保存する必要であるであろう。仮に生成した画像ファイルがターゲット装

置に大きすぎる場合は、背景アートワークのサイズ変更及びアートワークの上へのキャラクターの再階層化を含む処理を繰り返す必要があるであろう。現在利用可能なツールを使用すると、この作業は、良くても面倒くさいか又は労働集約的である。更に、上述のマニュアル（すなわち、予備レンダリング）手法は、静的画像を処理している時に限り可能である。ユーザが既存の画像の上に直ちにオブジェクトを階層化したい場合、このマニュアル手法は、可能なソリューションを提供しない。

【０００６】

オブジェクトを階層化する既存の手法は、ブラウザベースのオンライン技術に依存する。しかし、これらの手法は、基本的に上述のデスクトップ手法のオンラインバージョン（すなわち、「Ａｄｏｐｅ　Ｐｈｏｔｏｓｈｏｐ」手法）である。特に、上述の手法は、手持ち式装置のような所定のターゲット装置によって課される場合がある様々な制限を考慮しない。その代わりに、これらの手法は、固定された装置制限の組（すなわち、固定されたビューポート）を伴う環境に依存する。画像がターゲット装置に転送された場合、画像は、サイズ変更する必要がある場合がある。画像は動的に作成変更されないので、ベクトルグラフィックを利用することはできず、従って、画像のいくつかの特徴が失われることになる。例えば、６４０ｘ４８０の解像度でデスクトップブラウザ上で表示された時に良く見えるテキストは、１００ｘ１００の画面解像度を有するモバイル装置上の表示に対してサイズ変更された時にはひどいものに見えることになる。その代わりに、ターゲット装置の最終画面解像度、並びに任意の他の適用可能なターゲット装置制限に基づいてテキスト（並びに、任意の他のグラフィック）をレンダリングすることが望ましいであろう。現在の手法の上記及び他の制限事項を考慮して、より良いソリューションが求められている。

必要なものは、デジタル画像のレンダリングが異なるターゲット装置に対して動的に最適化又はカスタマイズされるように、論理ビューポートの動的な再形成を可能にし、ファイルサイズの制限を含む符号化パラメータの動的な調節を可能にする方法を提供するシステムである。本発明は、上記及び他の必要性を満足させるものである。

【０００７】

語彙解説

以下で説明する内容の理解を助けるために、制限的ではなく例示的に以下の定義を与える。

カラースペース補正：カラースペース補正は、ターゲットディスプレイの赤色、緑色、及び青色値の色度に合わせるために画像におけるＲ、Ｇ、Ｂ値を調節する処理である。例えば、Ｐｏｙｎｔｏｎ、Ｃ．Ａ．著「デジタルビデオの技術的入門」、第７章、ジョン・ワイリー、ニューヨーク、１９９６年を参照。

ガンマ補正：これは、ディスプレイの非線形性の逆をソース画像に適用することによってディスプレイの非線形性を補正する処理である。例えば、Ｐｏｙｎｔｏｎ、Ｃ．Ａ．著「デジタルビデオの技術的入門」、第６章、ジョン・ワイリー、ニューヨーク、１９９６年を参照。

【０００８】

ＨＴＭＬ：ワールド・ワイド・ウェブ上で文書を作成するために使用される公知のオーサリング言語である「ＨｙｐｅｒＴｅｘｔ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ」の略語。ＨＴＭＬは、ＳＧＭＬと類似のものであるが、厳密なサブセットではない。ＨＴＭＬは、様々なタグ及び属性を使用することによってウェブ文書の構造及びレイアウトを定める。例えば、ＲＦＣ１８６６：「Ｈｙｐｅｒｔｅｘｔ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ－２．０」を参照。

【０００９】

ＨＴＴＰ：「ＨｙｐｅｒＴｅｘｔ　Ｔｒａｎｓｆｅｒ　Ｐｒｏｔｏｃｏｌ」の略語であり、これは、ワールド・ワイド・ウェブによって使用される基本的プロトコルである。ＨＴＴＰは、メッセージがどのようにフォーマット設定されて伝送されるか、及び、様々な指令に応答してウェブサーバ及びブラウザがどのようなアクションを取るべきかを定める

。例えば、ユーザが自分のブラウザでＵＲＬに入った時、実際には、これは、ＨＴＴＰ指令をウェブサーバに送り、要求したウェブ頁をフェッチして送信するようにそれに指図する。ＨＴＴＰの更なる説明は、ＲＦＣ２６１６：「Ｈｙｐｅｒｔｅｘｔ　Ｔｒａｎｓｆｅｒ　Ｐｒｏｔｏｃｏｌ－ＨＴＴＰ／１．１で入手可能である。ＲＦＣ２６１６は、ワールド・ワイド・ウェブ・コンソーシアム（Ｗ３）から入手可能であり、現在は、「ｗｗｗ．ｗ３．ｏｒｇ／Ｐｒｏｔｏｃｏｌｓ／」でインターネットを通じて利用可能である。

【００１０】

レッドアイ補正：「レッドアイ」効果は、人間の目の網膜から反射するカメラのフラッシュによって引き起こされる。その赤色をより暗い色に「彩度を減じる」コンピュータアルゴリズムは、「赤み」を低減することができる。例えば、Ｗａｎｇ他に付与された「デジタル画像のレッドアイを自動的に検出及び低減するための装置及び方法」という名称の米国特許第６，２７８，４９１号を参照。

鮮明化する：これは、外見の改善又はディスプレイのぼけを補正するために画像内のグレースケールエッジを「明瞭化する」処理である。これは、一般的に、「アンシャープマスキング」を通じて達成される。例えば、画像の低域通過濾過バージョンを画像から差し引くことができる方法を説明する、Ｊａｉｎ、Ａ．Ｋ．著「画像処理の基礎」、プレンティス・ホール、エンゲルウッド・クリフス、ニュージャージー、１９８９年を参照。

【００１１】

ＵＲＬ：ワールド・ワイド・ウェブ上の文書及び他のリソースの全世界的アドレスである「Ｕｎｉｆｏｒｍ　Ｒｅｓｏｕｒｃｅ　Ｌｏｃａｔｏｒ」の略語。アドレスの第１の部分は、どのプロトコルを使用すべきかを示し、第２の部分は、リソースが位置するＩＰアドレス又はドメイン名を指定する。

ビューポート：ビューポートは、ユーザが最終画像を閲覧することになるターゲットディスプレイを意味する。例えば、モバイル手持ち式装置の場合、ビューポートは装置の画面である。しかし、個々のターゲット装置によっては、ビューポートは、必ずしも画面の物理的サイズに制限されるとは限らない。例えば、装置がスクロール機能を含む場合、ビューポートの（論理的）サイズは、画面の物理的サイズを超える場合がある。

【００１２】

白色点補正：白色点は、所定の環境における「基準の白」の色座標である。人間の目は、白色点に対する「色順応」ができる。白色点補正は、ターゲットディスプレイの白色点に対する人間の目の調節に対処するためにＲ、Ｇ、Ｂ色座標を調節する処理である。例えば、Ｇｉｏｒｇｉａｎｎｉ、Ｅ．Ｊ．他著「デジタルカラー制御」、アジソン・ウェスリー、リーディング、マサチューセッツ、１９９８年を参照。

【００１３】

ＸＭＬ：ＸＭＬは、Ｗ３Ｃによって開発された仕様である「Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ」の意味である。ＸＭＬは、特にウェブ文書用に考案されたＳＧＭＬの切り詰めバージョンである。それによって設計者は、独自のカスタマイズされたタグを作成することができ、アプリケーション間及び組織間のデータの定義、伝送、検証、及び解釈が可能になる。ＸＭＬの更なる説明については、例えば、Ｗ３Ｃからの推奨仕様である「Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ（ＸＭＬ）１．０」、（第２版、２０００年１０月６日）を参照。この仕様のコピーは、「ｗｗｗ．ｗ３．ｏｒｇ／ＴＲ／２０００／ＲＥＣ－ｘｍｌ－２０００１００６」でインターネット上で現在利用可能である。

【００１４】

【特許文献１】米国特許第６，２７８，４９１号

【特許文献２】米国特許出願出願番号第０９／５８８，８７５号

【特許文献３】米国特許出願出願番号第１０／０１０，６１６号

【非特許文献１】Ｐｏｙｎｔｏｎ、Ｃ．Ａ．著「デジタルビデオの技術的入門」、第６～７章、ジョン・ワイリー、ニューヨーク、１９９６年

【非特許文献２】ＲＦＣ１８６６：「Ｈｙｐｅｒｔｅｘｔ　Ｍａｒｋｕｐ　Ｌａｎｇｕａ

ｇｅ－２．０」
【非特許文献３】ＲＦＣ２６１６：「Ｈｙｐｅｒｔｅｘｔ　Ｔｒａｎｓｆｅｒ　Ｐｒｏｔｏｃｏｌ－ＨＴＴＰ／１．１
【非特許文献４】Ｊａｉｎ、Ａ．Ｋ．著「画像処理の基礎」、プレンティス・ホール、エンゲルウッド・クリフス、ニュージャージー、１９８９年
【非特許文献５】Ｇｉｏｒｇｉａｎｎｉ、Ｅ．Ｊ．他著「デジタルカラー制御」、アジソン・ウェスリー、リーディング、マサチューセッツ、１９９８年
【非特許文献６】「Ｅｘｔｅｎｓｉｂｌｅ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ（ＸＭＬ）１．０」、第２版、２０００年１０月６日
【発明の開示】
【００１５】
特定の装置タイプに対してカスタマイズされた画像のオンデマンド作成のためのシステムを説明する。一実施形態では、システムは、各画像が個別の層内に配置された画像成分を含む、画像のためのリピジトリーの役目をするモジュールと、リピジトリーから特定の画像を検索するための装置からの要求を処理するためのモジュールとを含み、このモジュールは、要求に含まれた情報に部分的に基づいて装置に対する特定の装置タイプを判断し、システムは更に、装置に対してカスタマイズされた特定画像のコピーを作成するためのモジュールを含み、このモジュールは、特定画像の個別の層内の画像成分の少なくともいくつかが装置に対してカスタマイズされるように、判断された装置タイプに基づいて特定画像の個別の層内の画像成分を個々にレンダリングする。
【００１６】
また、クライアント装置に伝送された画像の表示を動的に最適化する方法も説明する。一実施形態では、本方法は、表示用のターゲット画像を検索するための特定クライアント装置からのオンライン要求を受信する段階を含み、この要求は、クライアント装置の装置タイプの判断を助ける情報を含み、ターゲット画像は、個々の層内に配置された画像成分を含み、要求に基づいて、特定クライアント装置の装置タイプを判断する段階と、判断された装置タイプに基づいて、特定クライアント装置に対するビューポート及び階層化情報を指定する情報を検索する段階と、ビューポート及び階層化情報に基づいて、特定クライアント装置での表示のために最適化されたターゲット画像のバージョンを作成する段階と、ターゲット画像の作成されたバージョンを表示のためにクライアント装置に伝送する段階とを更に含む。
【発明を実施するための最良の形態】
【００１７】
以下の説明は、デジタル画像環境で実施される本発明の現時点の好ましい実施形態に焦点を当てる。しかし、本発明は、いかなる特定の一用途又はいかなる特定の環境にも限定されるものではない。その代わりに、当業者は、本発明のシステム及び方法は、様々な異なる装置に有利に使用することができることを見出すであろう。従って、以下の例示的実施形態の説明は、例証を目的とし、限定的ではない。
【００１８】
Ｉ．デジタルカメラベースの実施
Ａ．デジタルカメラの基本的構成要素
本発明は、デジタルカメラのような媒体捕捉及び記録システム上で実施することができる。図１は、本発明の実施に適切なデジタルカメラ１００の非常に一般的なブロック図である。図示のように、デジタルカメラ１００は、画像装置１２０、システムバス１３０、及びプロセッサ又はコンピュータ１４０（例えば、マイクロプロセッサベースのユニット）を含む。また、その画像がデジタルカメラ１００によって捕捉される対象又はオブジェクト１５０が示されている。オブジェクト１５０の画像を捕捉する際のデジタルカメラ１００のこれらの構成要素の一般的な作動をここで以下に説明する。
【００１９】
図示のように、画像装置１２０は、装置がオブジェクトの視像を捕捉することができる

という意味においてオブジェクト１５０に光学的に結合される。光学的結合としては、例えば、レンズアセンブリ（図示せず）のような画像装置１２０上にオブジェクト１５０を結像するための光学要素の使用を含むことができる。画像装置１２０は、例えば、システムバス１３０を通じてコンピュータ１４０と通信する。コンピュータ１４０は、画像装置１２０に対して全体的な制御を行う。作動面では、コンピュータ１４０は、実際には作動内容及び時期を告げることによって画像装置１２０を制御する。例えば、コンピュータ１４０は、デジタルカメラ１００の他の電機周辺装置（例えば、フラッシュ付属装置）で画像装置１２０の制御を調節することを可能にする全般的な入力／出力（Ｉ／Ｏ）制御を行う。

【００２０】

撮影者つまりカメラユーザがオブジェクト１５０に画像装置１２０の狙いを定め（ユーザによるピント合わせの有無を問わず）、捕捉ボタン又は何らかの他の手段を用いてカメラ１００にオブジェクト１５０の像を捕捉するように指示すると、コンピュータ１４０は、システムバス１３０を通じて画像装置１２０にオブジェクト１５０を表す画像を捕捉するように命令する。画像装置１２０は、本質的には、オブジェクト１５０から反射した光を捕捉してその光を画像データに変換することによって作動する。捕捉された画像データは、システムバス１３０でコンピュータ１４０に転送され、コンピュータ１４０は、様々な画像処理を画像データに行った後に内部メモリに記憶する。また、システムバス１３０は、様々な状態及び制御信号を画像装置１２０とコンピュータ１４０との間で伝送する。画像装置１２０及びコンピュータ１４０の構成要素及び作動について、ここでより詳細に以下に説明する。

【００２１】

Ｂ．画像装置上の画像捕捉

図２Ａは、従来のデジタル画像装置１２０のブロック図である。図示のように、画像装置１２０は、絞りを有するレンズ２１０、１つ又はそれ以上のフィルタ２１５、画像センサ２３０（例えば、ＣＭＯＳ、ＣＣＤなど）、焦点機構（例えば、モータ）２４１、タイミング回路２４２、信号プロセッサ２５１（例えば、アナログ信号プロセッサ）、アナログ／デジタル（Ａ／Ｄ）変換器２５３、及びインタフェース２５５を含む。これらの構成要素の作動について、ここで以下に説明する。

【００２２】

作動面では、画像装置１２０は、光路２２０に沿って画像センサ２３０に当たる反射光を通じてオブジェクト１５０の画像を捕捉する。レンズ２１０は、光路２２０に沿ってオブジェクト１５０からの光を画像センサ２３０上に集光するための光学要素を含む。焦点機構２４１は、レンズ２１０を調節するために使用することができる。フィルタ２１５は、オブジェクト１５０によって反射された光の異なる色成分を分離するために画像センサ２３０上に配置された１つ又はそれ以上のカラーフィルタを含むことが好ましい。例えば、画像センサ２３０は、赤色、緑色、及び青色フィルタで覆うことができ、このようなカラーフィルタは、より鮮明な画像及びより本来の色をもたらすように設計されたパターン（モザイク）で画像センサに亘って混ざって配置される。

【００２３】

従来のカメラは、画像を捕捉するためにフィルムを露光させるが、デジタルカメラは、画像センサ（例えば、画像センサ２３０）、すなわち、半導体電子素子上に光を集める。画像センサ２３０は、電荷結合素子（ＣＣＤ）又は相補型金属酸化膜半導体（ＣＭＯＳ）センサとして実施することができる。ＣＭＯＳ及びＣＣＤのいずれの画像センサも、表面にあるフォトサイト（又はフォトダイオード）として公知である小さなセルから成る格子上に光を捕捉することによって作動する。画像センサの表面は、一般的にフォトサイト上で輝く光を電荷に変換する何十万ものフォトサイトから成る。所定の画像に基づいて、異なる量の光が各フォトサイトに当たり、その結果、異なる量の電荷がフォトサイト上に発生する。その後、これらの電荷を測定してデジタル情報に変換することができる。デジタルカメラ内での介在に適切なＣＣＤセンサは、米国ニューヨーク州ロチェスター所在のイ

ーストマン・コダック、オランダのフィリップス、及び日本のソニーを含むいくつかの販売業者から市販されている。適切なＣＭＯＳセンサも、様々な販売業者から市販されている。代表的な販売業者としては、オランダの「ＳＴＭｉｃｒｏｅｌｅｃｔｒｏｎｉｃｓ」（元のＶＳＬＩ・ビジョン・リミテッド）、米国イリノイ州ショームバーグ所在のモトローラ、及び米国カリフォルニア州サンタクララ所在のインテルがある。

【００２４】

画像センサ２３０は、オブジェクト１５０の画像を捕捉するように命令されると、それに応答して、捕捉されたオブジェクト１５０を表す一組の生画像データを生成する（例えば、ＣＣＤによる実施例の場合は、ＣＣＤフォーマットで）。ＣＣＤセンサを使用する実施形態では、例えば、画像センサ２３０上で捕捉される生画像データは、信号プロセッサ２５１、アナログ／デジタル（Ａ／Ｄ）変換器２５３、及びインタフェース２５５を通る。インタフェース２５５は、信号プロセッサ２５１、焦点機構２４１、及びタイミング回路２４２を制御するための出力を有する。画像データは、インタフェース２５５から、図１で上述したようにシステムバス１３０でコンピュータ１４０に伝えられる。この画像データを処理する際のコンピュータ１４０の作動について、ここで以下に説明する。

【００２５】

Ｃ．画像処理

従来の搭載プロセッサ又はコンピュータ１４０は、カメラ１００の作動を指図し、画像装置１２０上で捕捉された画像データを処理するために設けられる。図２Ｂは、プロセッサ又はコンピュータ１４０のブロック図である。図示のように、システムバス１３０は、画像装置１２０、（任意的な）パワーマネージメント２６２、プロセッサ（ＣＰＵ）２６４、ランダム・アクセス・メモリ（ＲＡＭ）２６６、入力／出力（Ｉ／Ｏ）コントローラ２８０、不揮発性メモリ２８２、取外し可能メモリインタフェース２８３、及び液晶ディスプレイ（ＬＣＤ）コントローラ２９０間の接続経路を形成する。取外し可能メモリ２８４は、取外し可能インタフェース２８３を通じてシステムバス１３０に接続される。代替的に、カメラ１００（及び、従って搭載コンピュータ１４０）は、取外し可能メモリ２８４又は取外し可能メモリインタフェース２８３がなくても実施することができる。パワーマネージメント２６２は、電源２７２と通信する。また、図２Ｂには、ＬＣＤコントローラ２９０及び入力／出力コントローラ２８０と電気接続されたカメラユーザインタフェース２９５が示されている。これらの構成要素の各々について、ここでより詳細に以下に説明する。

【００２６】

プロセッサ（ＣＰＵ）２６４は、一般的に、カメラ１００の作動を制御するための従来のプロセッサ装置（例えば、マイクロプロセッサ）を含む。プロセッサ２６４の実施は、様々な異なる方法で達成することができる。例えば、プロセッサ２６４は、ＤＳＰ（デジタル信号処理）論理ブロック、メモリ制御論理ブロック、ビデオ制御論理ブロック、及びインタフェース論理ブロックを有するマイクロプロセッサ（例えば、ＭＰＣ８２３マイクロプロセッサ、米国イリノイ州ショームバーグ所在のモトローラから販売）として実施することができる。代替的に、プロセッサ２６４は、例えば、「Ｒａｐｔｏｒ　ＩＩ」チップセット（米国カリフォルニア州ニューポートビーチ所在のコネクスタント・システムズ・インコーポレーテッドから販売）、「Ｓｏｕｎｄ　Ｖｉｓｉｏｎ　Ｃｌａｒｉｔｙ」の２、３、又は４チップセット（米国マサチューセッツ州ウェイランド所在のサウンド・ビジョン・インコーポレーテッドから販売）、又は処理コアを画像処理周辺装置と一体化する類似のチップセットを使用する「カメラ・オン・チップ（セット）」として実施することができる。プロセッサ２６４は、一般的に、マルチスレッド環境内でカメラ１００の様々な処理を制御するために、複数のソフトウエアルーチンを同時に実行することができる。

【００２７】

デジタルカメラ１００は、いくつかのメモリ構成要素を含む。メモリ（ＲＡＭ）２６６は、選択的に様々な記憶機能に割当てることができる動的メモリの連続的ブロックである

。動的ランダム・アクセス・メモリは、例えば、日本の東芝、米国アイダホ州ボイズ所在のミクロン・テクノロジー、日本の日立、韓国のサムソン・エレクトロニクスを含む様々な販売業者から販売されている。一般的に従来の読取専用メモリ又はフラッシュメモリを含むことができる不揮発性メモリ２８２は、カメラ１００の作動を制御するための一組のコンピュータ可読プログラム命令を記憶する。取外し可能メモリ２８４は、更に別の画像データ記憶域の役目をし、取外し可能メモリインタフェース２８３を通じてカメラ１００ユーザによって容易に取外し可能及び交換可能である不揮発性素子を含むこともできる。従って、いくつかの取外し可能メモリ２８４を有するユーザは、カメラ１００撮影機能を拡張するために一杯の取外し可能メモリ２８４を空き状態の取外し可能メモリ２８４と有効に交換することができる。取外し可能メモリ２８４は、一般的に、フラッシュディスクを使用して実施される。フラッシュメモリの利用可能な販売業者としては、例えば、米国カリフォルニア州サニーベール所在の「ＳａｎＤｉｓｋ」コーポレーション、及び日本のソニーがある。当業者は、デジタルカメラ１００は、本発明の画像捕捉及び処理方法に容易に対応する他のメモリ構成及び設計を組み込むことができることを理解するであろう。

【００２８】

また、デジタルカメラ１００は、一般的に、カメラユーザ又は他のシステム及び装置との通信のためのいくつかのインタフェースを含む。例えば、Ｉ／Ｏコントローラ２８０は、コンピュータ１４０へ及びそれからの通信を可能にするインタフェース装置である。Ｉ／Ｏコントローラ２８０は、外部ホストコンピュータ（図示せず）がコンピュータ１４０と接続して通信することを可能にするものである。また、図示のように、Ｉ／Ｏコントローラ２８０は、複数のボタン及び／又はダイヤル２９８、及び任意的な状態ＬＣＤ２９９と接続し、これらは、ＬＣＤ画面２９６に加えて、装置のユーザインタフェース２９５のハードウエア要素である。デジタルカメラ１００は、例えば、カメラユーザにフィードバックを行ってカメラユーザから入力を受信するためのユーザインタフェース２９５を含むことができる。代替的に、これらの要素は、ホスト装置に対するクライアントとして実施される媒体捕捉装置のためのホスト装置（例えば、携帯情報端末）を通じてもたらすことができる。モニタカメラのようなユーザと対話する必要がない実施形態については、上述のユーザインタフェース構成要素は必要でないと考えられる。ＬＣＤコントローラ２９０は、メモリ（ＲＡＭ）２６６にアクセスし、処理された画像データを表示のためにＬＣＤ画面２９６に転送する。ユーザインタフェース２９５は、ＬＣＤ画面２９６を含むが、カメラユーザにフィードバックを行うために、ＬＣＤ画面に加えて又はＬＣＤ画面２９６の代わりに、光学的ファインダ又は直視ディスプレイを使用することができる。ユーザインタフェース２９５の構成要素は、様々な販売業者から販売されている。その例としては、日本のシャープ、東芝、及びシチズン・エレクトロニクス、韓国のサムソン・エレクトロニクス、及び米国カリフォルニア州パロ・アルト所在のヒューレット・パッカードがある。

【００２９】

パワーマネージメント２６２は、電源２７２と通信し、カメラ１００の電力制御作動を調節する。電源２７２は、作動電力をカメラ１００の様々な構成要素に供給する。一般的な構成においては、電源２７２は、作動電力を主電力バス２７８及び二次電力バス２７９に供給する。主電力バス２７８は、画像装置１２０、Ｉ／Ｏコントローラ２８０、不揮発性メモリ２８２、取外し可能メモリ２８４に電力を供給する。二次電力バス２７９は、パワーマネージメント２６２、プロセッサ２６４、及びメモリ（ＲＡＭ）２６６に供給する。電源２７２は、バッテリ２７５及び補助バッテリ２７６に接続される。また、カメラユーザは、必要に応じて電源２７２を外部電源に接続することができる。電源２７２の通常の作動中に、主バッテリ２７５は、作動電力を電源２７２に供給し、この電源は、次に、主電力バス２７８及び二次電力バス２７９の両方を通じて作動電力をカメラ１００に供給する。主バッテリ２７５が故障した電源異常モード中は（例えば、出力電圧が最小作動電圧レベルを下回った時）、補助バッテリ２７６が作動電力を電源２７２に供給する。一般的な構成においては、電源２７２は、補助バッテリ２７６からの電力をカメラ１００の二

次電力バス２７９のみに供給する。
上述のシステム１００は、本発明を実施するために使用することができる媒体捕捉及び記録システム（例えば、デジタルカメラ）の基礎となる基本的ハードウエアを例示するために示されたものである。しかし、本発明は、デジタルカメラ装置のみに限定されず、以下に詳細に示す本発明の方法をサポートし、及び／又はその恩典を得ることができる様々な装置に有利に適用することができる。
【００３０】
Ｄ．システム環境
図３は、本発明が好ましく具体化される例示的な無線接続環境３００を示す。図示のように、環境３００は、動的信号処理（ＤＳＰ）ユニット３２５を含む中央演算処理装置（ＣＰＵ）３２０と、ランダム・アクセス・メモリ（ＲＡＭ）３３０（例えば、ＤＲＡＭ、ＳＲＡＭなど）と、１つ又はそれ以上の圧縮画像を記憶するためのフラッシュメモリ３４０とを含む画像装置３１０（例えば、デジタルカメラ１００のようなデジタルカメラ）を含む。画像装置３１０の基本的な作動は、以下の通りである。画像装置３１０を操作するユーザは、１つ又はそれ以上のデジタル画像（写真）を撮り、画像ファイルを画像装置３１０上のフラッシュメモリ３４０内に記憶することができる。画像のカメラ側の処理（例えば、圧縮）は、作業メモリ（すなわち、ＲＡＭ３３０）と協働するＤＳＰユニットによって処理される。処理後、画像は、無線ネットワーク３６０を通じてサーバコンピュータ３７０に送ることができる（例えば、インターネット上で）。サーバ３７０では、画像装置３１０から受信した画像データは、付加的な処理（例えば、グラフィックの重ね合わせ）を行うために、メモリ（ＲＡＭ）３９０（例えば、ＤＲＡＭ、ＳＲＡＭなど）内に取り戻すことができる。その後、処理された画像は、必要に応じて、サーバ３７０上に記憶するか、又は元の装置（例えば、カメラ１００）に転送して戻すか、又は他の装置に転送することができる。
【００３１】
ＩＩ．動的ビューポート階層化
Ａ．序文
コンテント作成者は、ユーザの写真に追加する面白いコンテントを作成したいものである。例えば、コンテント作成者は、ユーザの写真を面白いテキスト又はアニメーションで階層化したいと考える場合がある。これは、オンザフライでコンテントを作成することを伴う。しかし、コンテント作成者がオンザフライでコンテントを作成する時、作成者は、更に、異なる表示特性を有する様々な装置でコンテントを正しく表示するか又はレンダリングするという問題に直面する。本発明の手法は、最終的な表示において起るべきことの説明を可能にするソリューションを作成することである。例えば、例示的な説明は、アニメーションが画像の上に載り、テキスト「お誕生日おめでとう」がその上に表示されるフレームを用いて画像を表示すべきであることを指示するであろう。このようにして、このソリューションは、画像を異なる表示特性を有する様々な装置上で正しく表示することを可能にする。
【００３２】
より具体的には、本発明は、二面的手法を適用するものである。第一に、本発明の手法は、階層化をどのように行うかを指定することを可能にする記述言語を提供することである。現在の好ましい実施形態では、この記述言語は、ＸＭＬフォーマットに従い、所定の画像を形成する層の階層的記述をもたらす。異なる層としては、画像（例えば、ビットマップ）、アニメーション、テキスト、ベクトルグラフィックなどがある。記述言語は、異なる層をまとめて構成する方法及びビューポートで上述の層を表示する方法を説明することができるシンタックスを含む。記述言語は、正確なレイアウトを指定するものではないが、様々なターゲット装置の制限に適合する。特定の画像の所定の記述はサーバ上にあり、それはターゲット装置には送られない。その代わりに、ターゲット装置は、最終的な符号化されたフォーマット（画像）を受信する。従って、記述言語は、特定のターゲット装置によって課せられた制限の符号化に適合する。

【００３３】
本発明の二面的手法の第２は、画像がターゲット装置で正しくレンダリングされるように、ビューポートを動的に再形成又は再構成することである。所定のターゲット装置の一組の装置上の制限を考える。制限により、ピクセル当たりの許容最大ビット（例えば、８ビット／ピクセル）、最大画面サイズ（例えば、１００ピクセルｘ１００ピクセル）のような特定の限界値が指定されることになる。本発明によれば、ビューポートは、その時点で現在のターゲット装置の制限に適合するように動的に再構成される。更に、複数の制限を通常は満たさなければならない。例えば、ターゲット装置は、最大画像サイズ（例えば、５Ｋ）を指定することができる。その制限に適合するために、ビット深さ（すなわち、ビット／ピクセル）を小さくすることが必要であると考えられる。本発明の手法では、例えば、画像ビット深さを５Ｋファイルサイズ制限に適合するために４ビット／ピクセルに変更することができるように、装置の制限を相互に満たす段階を伴う。しかし、ビット深さは、８ビット／ピクセル（すなわち、ターゲット装置によってサポートされる最大ビット深さ）を超えることができない。総じて、論理ビューポート（及び、従って画像）をターゲット装置に動的に適合させるように潜在的に調節することができる様々な制限すなわちパラメータが存在する。
【００３４】
Ｂ．基本的方法
本発明は、良好な画像品質を維持しながらターゲット装置の制限を満たすために使用される反復的最適化（カスタマイズ）の方法を提供する。図４の４０１に示すように、階層化手法は、各層が最初に「拡張及びビューポート」処理という２つの基本的ブロックを通る場合に使用される。前者は、レッドアイ低減、コントラスト調節などの拡張を表す。後者は、ビューポートカラー及び外見の制限がカラー補正、ガンマ、鮮明化などを使用して補正される論理を表す。
【００３５】
上述の処理が終わると、層（例えば、層０及び層１）は、４０３で示すように、ビューポートにマップされるように待機状態である。ビューポート仕様コンポーネント４１７と通信するファイルサイズ制御ブロック４０５は、このマッピングのためにビューポートサイズ４０７を指定する。ビューポートサイズは、（例えば、スクロール機能のために）ターゲットディスプレイよりも大きなものとすることができる。層は、４０９に示すように、マッピング後に合併される。この処理の次の段階は、４１１でビューポートをクリップ経路にクリップすることである。クリップ経路は、ビューポート単位矩形（０．０、０．０、１．０、１．０）に適合するものであるが、レンダリング済みの層の１つであるように指定することもできる。その後、クリップされた矩形は、色の深み、符号化方法、システムパレットのような装置側の制限に従って符号化される。マッピング４１３は、この作動を表す。得られるファイルサイズがファイルサイズ制限を満たす場合（４１５で試験される）、画像は、ターゲット（例えば、モバイル）ディスプレイに戻される。他の場合は、ファイルサイズ制御ブロックは、ファイルサイズ制御ブロック４０５に戻るループによって示すように、ビューポートをサイズ変更してビューポートマッピングや合併などを再開する。
【００３６】
Ｃ．画像変換ＡＰＩ
以下は、画像変換を指定するためのインタフェースを説明するものである。インタフェースを有効に利用するために、階層化という枠組みに基づく本発明によって使用される画像処理モデルを理解することは有利である。層としては、例えば、画像層、テキスト層、ベクトルグラフィック層などを含むことができる。層は、空間的及び時間的属性を有する。
１）空間的階層化：層は、互いに対してどのように積み重ねられるかを指定する「順位」空間的属性を有する。更に、「Ｖｉｅｗｐｏｒｔ＿ｍａｐ　ｃｈｉｌｄ－ｅｌｅｍｅｎｔ」は、層がマップされるビューポートの副領域を指定する。

２）時間的階層化：層は、時間的にどのように配置されるかを説明する「ｓｔａｒｔ＿ｔｉｍｅ」、「ｔｉｍｅ」、「ｄｕｒａｔｉｏｎ」のような時間的属性を有する。

【００３７】

１．空間的階層化

画像変換ＡＰＩは、特殊効果を作り出すために様々な層（画像、テキスト、アニメーションなど）を組み合わせる方法を説明する階層化ＡＰＩである。図５Ａは、階層化パイプラインを示す（ここでは、時間的階層化を無視）。

１）最初に、層がレンダリングされる。

２）その後、層は、マップされてビューポート上に積み重ねられる。ビューポートは、寸法がターゲットディスプレイの寸法及び層のマッピング方法で決まる仮想の矩形である。

３）層スタックは、ビューポートにおいて合併される。

４）合併されたビューポート画像は、要求しているクライアントのディスプレイ制限に適合するようにフォーマット設定される（ビット深さ、パレット、ファイルフォーマットなど）。

５）その後、フォーマット設定された画像は、クライアントに返却される。

６）クライアントは、フォーマット設定された画像をディスプレイ上に表示する。

【００３８】

ビューポート座標系は、「正規化」座標形（図５Ｂ）であり、以下のようになる。すなわち、

原点は、ビューポートの左上隅にあり、

Ｘ軸は、右に進み、

Ｙ軸は、下に進み、

Ｘ座標は、ビューポート幅に対して正規化され、

Ｙ座標は、ビューポート高さに対して正規化される。

「ビューポート単位矩形」５５１は、座標（０．０、０．０）、（１．０、１．０）に亘る矩形であると定義される。各層は、「Ｖｉｅｗｐｏｒｔ＿ｍａｐ」に従ってビューポートの副領域にマップされる。ビューポートマッピング又はウィンドウの一例を図５Ｂの５５３に示す。

【００３９】

２．時間的階層化

空間的「順位」属性に加えて、層はまた、以下のような時間的属性（全てミリ秒で表記）を有する。

１）「ｓｔａｒｔ＿ｔｉｍｅ」：これは、層が表示される開始時間を指定する。デフォルトは、０ｍｓである。

２）「ｄｕｒａｔｉｏｎ」：層が表示される持続時間。デフォルト値は、無限大（ＩＮＦ）である。また、０という値は、無限大持続時間と解釈される。

３）「ｒｅｐｅａｔ＿ｐｅｒｉｏｄ」：表示が繰り返される周期的速度。デフォルト値は、無限大（ＩＮＦ）である。また、０という値は、無限大と解釈される。両方の値では、アニメーションが決して繰り返されないということになる。

【００４０】

３．ＸＭＬ手法

階層化は、「ＸＭＬ　ＡＰＩ」を使用して達成される。本方法においては、（ａｒｇ、ｖａｌ）対「ｅｎｈ＝＜ＸＭＬ＿ＵＲＬ＞」は、使用する「ＸＭＬ　ＵＲＬ」を指定する。

例：

ｈｔｔｐ：／／ｅｓｗｉｔｃｈ．ｆｏｏ．ｃｏｍ／ｅｓ？ｓｒｃ＝ｈｔｔｐ：／／ｓｏｕｒｃｅ．ｆｏｏ．ｃｏｍ／ｉｍａｇｅｓ／ｉｍｇｌ．ｊｐｇ＆ｅｎｈ＝ｈｔｔｐ：／／ｓｏｕｒｃｅ．ｆｏｏ．ｃｏｍ／ｔｅｍｐｌａｔｅｓ／ｅｎｈａｎｃｅ．ｘｍｌ．

１）「ｓｒｃ」画像（ｈｔｔｐ：／／ｓｏｕｒｃｅ．ｆｏｏ．ｃｏｍ／ｉｍａｇｅｓ／

ｉｍｇｌ．ｊｐｇ）は、任意の背景層（層番号０）とＸＭＬ拡張ファイルで指定された他の層との間に挿入されるソース層になる。

２）ＸＭＬ（構成）ファイルは、他の層を説明する。更に、ビューポート制限を説明する。

３）ＸＭＬ拡張方法は、ＵＲＬライン（ａｒｇ、ｖａｌ）対と共に使用することができない（すなわち、この２つの方法は相互に排他的）。

【００４１】

４．ＸＭＬ階層

「ＸＭＬ　ＡＰＩ」で使用されるオブジェクトの階層を図５Ｃに示す。灰色がかった線は属性を指す。濃い線は要素を示す。この階層において、属性は、簡単な形式を表し、要素は、複雑な形式を表す。その後の節は、より詳細に階層内の要素及び属性を説明する。階層内のいくつかの要素及び属性は、上級ユーザ用のものであり、灰色（非強調）テキストで示されている。

【００４２】

５．画像変換

画像変換は、画像階層化作動の詳細を包み込むための要素タグから成る。

【００４３】

（表：画像変換）

属性	有効値	説明
ｘｍｌｎｓ	「ｈｔｔｐ：／／ｗｗｗ．ｌｉｇｈｔｓｕｒｆ．ｃｏｍ／ｅｓｗｉｔｃｈ２／ｉｍａｇｅ＿ｔｒａｎｓｆｏｒｍ／１．０」	画像変換マークアップのネームスペース及び改訂
子要素	説明	
ｉｍａｇｅ＿ｌａｙｅｒ	画像層	
ｔｅｘｔ＿ｌａｙｅｒ	テキスト層	
ｂｅｚｉｅｒ　Ｌａｙｅｒ	ベジエ曲線で形状を定めるための層	
Ｖｉｅｗｐｏｒｔ	出力にマップする方法を判断するビューポート制限及び機能	

【００４４】

６．層の共通の特性

層は、空間的及び時間的挙動を説明する共通の特性を有する。

ａ）空間的特性

層の空間的特性は、「順位」属性及び「ｖｉｅｗｐｏｒｔ＿ｍａｐ」子要素で判断される。

【００４５】

（表：層の空間的属性）

属性	有効値	説明
ｏｒｄｅｒ	１からｎ	これは、ビューポート上の表示の空間的順位を示す相対的番号であり、大きな順位を有する層は、小さい順位を有する層の上に積み重ねられる。
子要素	説明	
Ｖｉｅｗｐｏｒｔ＿ｍａｐ	これは、層をビューポートにマップする方法を説明する。	

【００４６】

「Ｖｉｅｗｐｏｒｔ＿ｍａｐ」は、全ての層の共通の要素である。これによって層がビ

ューポートにマップされる方法が決まる。マッピングは、以下に基づいている。
Ｗｉｎｄｏｗ：これは、層がマップされるべきビューポート内の領域である。デフォルトにより、ウィンドウは、ビューポートに亘っている。
Ｍｏｄｅ：これは、層をウィンドウに嵌め込む方法を説明する。デフォルトは、「適合」である。

【００４７】
以下の（上級）要素は、マッピング後に画像を再配置する上で有用である。
Ａｌｉｇｎ：これは、層をウィンドウ内に整列させる方法を説明する。指定がない場合、中央アラインメントが仮定される。
Ｏｆｆｓｅｔ：これは、ウィンドウにマッピングした後に、層にオフセットを適用すべきかを説明する。指定がない場合、（０．０、０．０）というオフセットが仮定される。

【００４８】
（表１：Ｖｉｅｗｐｏｒｔ＿ｍａｐ）

属性	有効値	説明
ｍｏｄｅ	以下のうちの１つ： ・Ｆｉｔ（デフォルト） ・Ｆｉｌｌ ・Ｆｏｒｃｅ ・Ａｓ－ｉｓ	層をウィンドウにマップする方法。本方法は、ウィンドウに対する層の最初のマッピングが起こるべき方法を定める。ウィンドウの外にある層の領域は、ウィンドウにクリップされる。 ・Ｆｉｔ：ウィンドウ内に収まるように層がスケーリングされることを意味する。層のアスペクト比は保持される。画像は、１つの方向に沿ってのみウィンドウを満たすことになる。 ・Ｆｉｌｌ：ウィンドウを満たすために画像をスケーリングする。画像の一部分がクロッピングされてもよい。 ・Ｆｏｒｃｅ：ウィンドウを満たすように層のアスペクト比をスケーリングして変える。 ・Ａｓ－ｉｓ：マッピング中にスケーリングを行わない。 ・Ｆｉｔ－ｔｏ－ｗｉｄｔｈ：層の幅がビューポート幅に対してサイズ変更されることを意味する。層のアスペクト比は保持される。層は、高さに沿ってビューポートから溢れてもよい（その結果、クロッピングされる）。

子要素	使用法	説明
ｗｉｎｄｏｗ	<window x =“<LT_X>” y =“<LT_Y>” width=“<WIDTH>” height=“<HEIGHT>” />	層をマップするビューポートの副領域。（ｘ、ｙ）属性は、左上隅を定め、幅及び高さ属性は、サイズを定める。 ・＜ＬＴ＿Ｘ＞：左上ｘ座標。デフォルトは０．０。 ・＜ＬＴ＿Ｙ＞：左上ｙ座標。デフォルトは０．０。 ・＜ＷＩＤＴＨ＞：ウィンドウの幅。デフォルトは１．０。 ・＜ＨＥＩＧＨＴ＞：ウィンドウの高さ。デフォルトは１．０。
Ａｌｉｇｎ	<align xalign=“<ALIGNX> yalign=“<ALIGNY> />	この子要素は、マッピング中に層がＸ及びＹ軸においてウィンドウ内で整列すべき方法を説明する。 ・＜ＡＬＩＧＮＸ＞：「左」、「右」、又は「中央」の１つとすることができる。デフォルトは「中央」。 ・＜ＡＬＩＧＮＹ＞：「上」、「下」、又は「中央」の１つとすることができる。デフォルトは「中央」。
Ｏｆｆｓｅｔ	<offset x=“<OFFSET_X>” y=“<OFFSET_Y>” />	マッピング及びアラインメント後に層をオフセットする量。 ＜ＯＦＦＳＥＴ＿Ｘ＞：Ｘ方向にオフセットする量。デフォルトは０．０。 ＜ＯＦＦＳＥＴ＿Ｙ＞：Ｙ方向にオフセットする量。デフォルトは０．０。 注：層は、マッピング、アラインメント、及びオフセット後に、ビューポートマップウィンドウに対してクリップされる。すなわち、ウィンドウの外側にある層に一部は見えなくなる。

【００４９】
ｂ）時間的特性
時間的属性：「ｓｔａｒｔ＿ｔｉｍｅ」、「ｄｕｒａｔｉｏｎ」、及び「ｒｅｐｅａｔ＿ｐｅｒｉｏｄ」は、全ての層によってサポートされる。
【００５０】

（表：層の時間的特性）

属性	有効値	デフォルト	説明
ｓｔａｒｔ＿ｔｉｍｅ	＞＝０ｍｓ	０ｍｓ	層の表示の開始時間
ｄｕｒａｔｉｏｎ	＞０ｍｓ	ＩＮＦＩＮＩＴＹ	表示の持続時間
ｒｅｐｅａｔ＿ｐｅｒｉｏｄ		０ｍｓ	層は、以下の制限を満たさなければならない。 ・Ｓｔａｒｔ＿ｔｉｍｅ＋ｄｕｒａｔｉｏｎ＜＝ｒｅｐｅａｔ＿ｐｅｒｉｏｄ

【００５１】

７．画像層

画像層の属性及び子要素は、作成方法とビューポート内でウィンドウにマップする方法とを決める。

【００５２】

（表：画像層の属性及び要素）

属性	有効値	デフォルト	説明
ｏｒｄｅｒ	６節を参照。		
ｓｔａｒｔ＿ｔｉｍｅ			
ｄｕｒａｔｉｏｎ			
ｒｅｐｅａｔ＿ｐｅｒｉｏｄ			
ｓｒｃ	Ａ　ＵＲＬ		ソース画像
子要素	説明		
Ｖｉｅｗｐｏｒｔ＿ｍａｐ	これは、層をビューポートにマップする方法を説明する。		

【００５３】

ａ）ソース画像層

「ｓｒｃ＝＜ＩＭＡＧＥ＿ＵＲＬ＞」（ａｒｇ、ｖａｌ）対によって指定された画像は、「ソース」層になる。この層は、任意の背景（層順位０）と残りの層の間に挿入される。この層は、「Ｖｉｅｗｐｏｒｔ＿ｍａｐ」に対するデフォルト属性値及び子要素値を有する。

【００５４】

８．テキスト層

この層は、テキストレンディションをサポートする。

【００５５】

（表：テキスト層の属性及び要素）

属性	有効値	デフォルト	説明
order	上記を参照。		
start_time			
duration			
repeat_period			
text	ＵＴＦ－８ユニコード・ストリング	なし	テキストストリングは、ＵＴＦ－８ストリングと定められる。このフォーマットは、ユニコード規格によって定められる一切の文字をサポートすることができる。指定されたフォントファイルでユニコード値の文字が得られる限り、文字はサポートされる。
centerx	イエス、ノー	イエス	Ｘ方向へのセンターリング。 ・「ノー」という値は、テキストを左境界に合わせる。
centery	イエス、ノー	イエス	Ｙ方向へのセンターリング。 ・「ノー」という値は、テキストを下の境界に合わせる。
font_file	フォントディレクトリ内の「Ｔｒｕｅ　Ｔｙｐｅ」ファイル名	なし	フォントファイルは、「Ｔｒｕｅ　Ｔｙｐｅ」ファイルでなければならない。このファイルは、単一のフェースファイル（*．ｔｔｆ）又は複数のフェース「Ｔｒｕｅ　Ｔｙｐｅ」収集（*．ｔｔｃ）ファイルとしてもよい。
font_color	色	０ｘ００００００（黒）	色は、ｈｅｘフォーマットで０ｘＲＲＧＧＢＢと指定される（ＲＲ＝赤色、ＧＧ＝緑色、ＢＢ＝青色）。
font_mode	・自動 ・固定	自動	・自動： ○フォントサイズは、ウィンドウ内の指定されたテキストに合わせるために自動で判断される。 ○「ｆｏｎｔ＿ｓｉｚｅ＿ｍｉｎ」属性が実施される。 ・固定： ○「ｆｏｎｔ＿ｓｉｚｅ」は、「点」（１点＝１／６４インチ）で指定される。 ○「ｆｏｎｔ＿ｓｉｚｅ＿ｍｉｎ」属性は無視される。
font_size	４－１２８	１２	固定モードフォントに使用するフォントのサイズ。点で指定。
font_size_min	４＋	６	このパラメータは、「自動」モードで有用であり、フォントサイズがこのレベルを下回らないことを保証するのに使用することができ、小さなディスプレイを有する装置の場合でも「明瞭な」テキストをもたらす。
子要素	説明		
Viewport_map	「ｍａｐ」、「ａｌｉｇｎ」、「ｏｆｆｓｅｔ」は無視される（すなわち、ウィンドウ要素のみが使用される）。		

【００５６】

９．ベジエ層

ベジエ層は、ベクトルグラフィックを重ね合わせるために使用される。この層の意図は、動的テキスト挿入機能でベクトルグラフィックをサポートすることである。

【００５７】

（表：ベジエ層の属性及び要素）

属性	有効値	デフォルト	説明
ｏｒｄｅｒ	上記を参照。		
ｓｔａｒｔ＿ｔｉｍｅ			
ｄｕｒａｔｉｏｎ			
ｒｅｐｅａｔ＿ｐｅｒｉｏｄ			
ｓｒｃ	Ａ　ＵＲＬ	指定すべきである。	「Ａｄｏｂｅ　Ｉｌｌｕｓｔｒａｔｏｒ　Ａ１８　ＥＰＳ」ファイルフォーマット内でベジエ曲線を指定するファイルに対するパスネーム。このパスネームは、「.ｅｓｐ」拡張子を有するべきである。
ｏｒｄｅｒ	１からｎ	指定すべきである。	順位は、最終出力が生成された時に層の積み重ねを定める。数字が大きいものが、小さい数字のものの上にレンダリングされる。
Ｏｐａｃｉｔｙ	０～１００	１００	グラフィックの全体的な不透明度。
子要素	説明		
Ｔｅｘｔ＿ｂｏｘ	これは、ベジエ層に挿入すべきテキストを説明する。		
Ｖｉｅｗｐｏｒｔ＿ｍａｐ	画像層と同じ。		

【００５８】
（表：ベジエ層の「Ｔｅｘｔ＿ｂｏｘ」要素）

属性	有効値	デフォルト	説明
ｔｅｘｔ	テキスト層の対応する属性と同じ。		
ｃｅｎｔｅｒｘ			
ｃｅｎｔｅｒｙ			
ｆｏｎｔ＿ｆｉｌｅ			
ｆｏｎｔ＿ｃｏｌｏｒ			
ｆｏｎｔ＿ｍｏｄｅ			
ｆｏｎｔ＿ｓｉｚｅ			
ｆｏｎｔ＿ｓｉｚｅ＿ｍｉｎ			
子要素	説明		
ｂｏｕｎｄｉｎｇ＿ｂｏｘ	これは、「Ａｄｏｂｅ　Ｉｌｌｕｓｔｒａｔｏｒ」ファイルの点座標空間で指定されたテキストの有界ボックスである。 ・使用法：＜ｂｏｕｎｄｉｎｇ＿ｂｏｘ　ｘ＝“＜ｌｌｘ＞”　ｙ＝“＜ｕｒｙ＞”ｗｉｄｔｈ＝“＜ｗｉｄｔｈ＞”ｈｅｉｇｈｔ＝“＜ｈｅｉｇｈｔ＞” 　○＜ｌｌｘ＞：点における左下Ｘ座標 　○＜ｕｒｙ＞：点における右上Ｙ座標 　○＜ｗｉｄｔｈ＞：点における有界ボックスの幅 　○＜ｈｅｉｇｈｔ＞：点における有界ボックスの高さ テキスト有界ボックスを判断する手順： ・「Ａｄｏｂｅ　Ｉｌｌｕｓｔｒａｔｏｒ」内で関連のグラフィックを開く。 ・「Ｆｉｌｅ－＞Ｄｏｃｕｍｅｎｔ　Ｓｅｔｕｐ－＞Ｕｎｉｔｓ－＞Ｐｏｉｎｔｓ」を選択する。 ・矩形ツールでテキスト有界ボックス領域を描く。 ・選択ツールで矩形を選択する。 　○これは、矩形を強調表示し、「“ｉｎｆｏ：」パレット内の有界ボックス情報を示す。これは、ＸＭＬ層仕様内に入力されるべき有界ボックス情報である。「ｉｎｆｏ」パレット内の（Ｘ、Ｙ、Ｗ、Ｈ）は、「ｌｌｘ」、「ｌｌｙ」、「ｗｉｄｔｈ」、「ｈｅｉｇｈｔ」に対応する。 ・矩形を削除する－それはもはや必要ではない（テキスト有界ボックスを判断するためにのみ有用であった）。		

【００５９】

１０．ビューポート

層がビューポート上にマップされて合併された状態で、得られる画像は、ビューポート要素で指定された制限に従って、クライアントの好ましい画像フォーマットにマップされる。

【００６０】

（表：ビューポート要素）

属性	有効値	デフォルト	説明
ａｓｐｅｃｔ＿ｌａｙｅｒ	画像層順位番号又は－１	最低画像層	アスペクト（又は「アンカー」）層は、全ての他の層を位置決めする時にアンカーとして使用される層を決める。アスペクト層は、ビューポートのアスペクト比を決める（上記を参照）。
ｆｏｒｃｅ＿ｌａｙｅｒ	Ａ　ＵＲＬ	色は強制されない。	この要素は、強制される色を定める。強制される色の組は、以下のフォーマットの１つで指定される（上記を参照）： ・ＡＣＴ（．ａｃｔ）：「Ａｄｏｂｅ Ａｃｔｉｖｅ Ｔａｂｌｅ Ｆｏｒｍａｔ」（．ａｃｔ） ・ＧＩＦ（．ｇｉｆ） ・ＰＮＧ（．ｐｎｇ

【００６１】

ａ）アスペクト／アンカー層

本発明では、ビューポート幅は、ターゲット装置の幅に設定される。しかし、ビューポート高さは、「ａｓｐｅｃｔ＿ｌａｙｅｒ」によって定められたアスペクト比に基づいて決定される。

・「ａｓｐｅｃｔ＿ｌａｙｅｒ」＝－１：これは、最も単純な場合である。この場合、アスペクト比は、ターゲット装置のディスプレイと同じである。

例：ターゲットモバイル装置は、１００ｘ１２０である。従って、本発明では、１００ｘ１２０であるビューポートが作成される。

・「ａｓｐｅｃｔ＿ｌａｙｅｒ」＝何らかの画像層の順位番号：画像層のアスペクト比でビューポートの高さが決まる。

例：画像は、６４０ｘ４８０である。モバイル装置は、１００ｘ１００である。従って、本発明では、１００ｘ７５であるビューポートが作成される。座標系はビューポートに対して正規化されるので、全ての階層化は、この画像層に対するものとなる。

・「ａｓｐｅｃｔ＿ｌａｙｅｒ」未指定（デフォルト）：アスペクト層が未指定の場合、「最低」（「順位」において）画像層がアスペクト層として使用される。画像層がない場合、「ａｓｐｅｃｔ＿ｌａｙｅｒ」は、－１に設定される。

ビューポートの寸法は、最初は上述の方法に従って決定されるが、この寸法は、ファイルサイズに関する制限を満たすように調節することができる。アスペクト比は、ビューポートがサイズ変更された時に保持される。

【００６２】

ｂ）Ｆｏｒｃｅ＿ｃｏｌｏｒｓ

強制される色の組は、以下のフォーマットの１つで指定される。

１）ＡＣＴ（．ａｃｔ）：「Ａｄｏｂｅ Ａｃｔｉｖｅ Ｔａｂｌｅ Ｆｏｒｍａｔ」（．ａｃｔ）：これは、色表を定める。色表内の色の組が使用される。

２）ＧＩＦ（．ｇｉｆ）：色の組は、ＧＩＦ画像内にある第１の色パレットである。

３）ＰＮＧ（．ｐｎｇ）：色の組は、ＰＮＧ画像内にある第１の色パレットである。

【００６３】

モバイル装置は、一般的に以下の色モードの１つを有する。

７）真の色：このモードにおいては、システムは、任意の色を表示することができる。「Ｆｏｒｃｅ＿ｃｏｌｏｒｓ」は、この場合は効果がない。

８）インデックス付きの色：このモードにおいては、システムは、限定された数の色を表示することができる。インデックス付き色モード内には、２つのサブモードがある：

ａ．固定パレット：固定パレットを有する装置は、柔軟性がないために「ｆｏｒｃｅ＿

ｃｏｌｏｒｓ」に適合することができない。「ｆｏｒｃｅ＿ｃｏｌｏｒｓ」の指令は、これらの装置の場合は無視される。

ｂ．適応パレット：大きな部類の装置は、小さな組の色（例えば、２５６）に適合することができるが、色は、任意の色とすることができる。「ｆｏｒｃｅ＿ｃｏｌｏｒｓ」は、この場合に最も有用である。

システムがサポートすることができる色が「ｆｏｒｃｅ＿ｃｏｌｏｒｓ」よりも多い場合、「ｆｏｒｃｅ＿ｃｏｌｏｒｓ」内の色の全てが使用される。システムがサポートすることができる色が「ｆｏｒｃｅ＿ｃｏｌｏｒｓ」よりも少ない場合、「ｆｏｒｃｅ＿ｃｏｌｏｒｓ」のサブセットが使用される。

【００６４】

１１．クラスの定義

「ＩｍａｇｅＴｒａｎｓｆｏｒｍ」クラス、「ＩｍａｇｅＬａｙｅｒ」クラス、及び「Ｖｉｅｗｐｏｒｔ」クラスのＣ＋＋クラスの定義をここで示す。

ａ）ＩｍａｇｅＴｒａｎｓｆｏｒｍ

【００６５】

a) ImageTransform

```
/**
  * class ImageTransform
  **/
class ImageTransform
{
    friend class Layer;
    friend class Viewport;
public:
     /// Constructor
     ImageTransform();
     /// Destructor
     ~ImageTransform();
   /// Get the viewport object
     Viewport* GetViewport();
     /// Set the Output File Name
     ITERR SetOutputFileName(const std::string & outFileName);
     /// Creating a layer
```

【００６６】

```
    ImageLayer*       CreateImageLayer  (int32_t StackOrder);
    TextLayer*        CreateTextLayer         (int32_t StackOrder);
    BezierLayer*      CreateBezierLayer (int32_t StackOrder);
    /// Get the aspect/anchor layer. This is the layer that determines
   /// "anchor" when displaying all other layers.
    Layer *GetAspectLayer();

    /// --------Encoding---------------------------
  /// Enable (or disable) encoding MIME type image/gif images
  /// compressed with the LZW algorithm
  void EnableLzwGifEncoding(bool enable = true);
  /// Enable (or disable) decoding MIME type image/gif images
  /// compressed with the LZW algorithm
  void EnableLzwGifDecoding(bool enable = true);
    /// --------Rendering----------------------------
    /// Render the image transform
    ITERR     Render();
  /// Getting rendered parameters
  int32_t       GetRenderedWidth();
  int32_t       GetRenderedHeight();
  int32_t       GetRenderedContentLength();
  std::string GetRenderedMimeType();
  /// Typedef for a UrlAccess call-back which is plugged into the
  /// image transform object to access media by URL - It returns the
  /// HTTP status code from the access.
  typedef int32_t (UrlAccessFunction)(std::string url,
                                      std::ostream * fromUrlStream,
                                      void * ref,
                                      std::string * resStr = NULL);
   /// Set the Url Accessor funciton which is called to accessing
  /// media by URL
  void SetUrlAccessFunction(UrlAccessFunction * fxn, void * ref =
NULL);

  // Anchor to Display Mapping Mode. This mode decides how an anchor
  // layer is mapped to the display:
  // CLAMP_TO_WINDOW: Clamp to fit withing display window
  // CLAMP_TO_WIDTH:  Allow height to exceed display height,
  //                  but clamp to Width
  typedef enum
  {
```

【００６７】

```
        CLAMP_TO_WINDOW,
        CLAMP_TO_WIDTH
    } AnchorToDisplayMapMode;
    ITERR SetAnchorToDisplayMapMode(AnchorToDisplayMapMode Mode);
    AnchorToDisplayMapMode GetAnchorToDisplayMapMode() const;

private:
      // Fetch a "media" or other object and return a temp file name
    std::string FetchUrlObject(const std::string& url);

    // Private rendering functions:

    // Load the layers
    ITERR   LoadLayers();
    // Just size the layers
    ITERR   SizeLayers();

    // Compute Viewport size - previous to enforcing file size
constraint
    ITERR   ComputeViewportSize(int32_t *pWidth, int32_t *pHeight);

    // Do the actual rendering to output
    ITERR   RenderOutput();

    // Internal rendering to memory
    ITERR   RenderToMemory(IMG_IOHANDLER *pIO);

    // Render with no output: Useful to compute Rendered parameters
    ITERR   RenderParameters();

    // Setting rendered parameter values
    ITERR       SetRenderedWidth(int32_t Width);
    ITERR       SetRenderedHeight(int32_t Height);
    ITERR       SetRenderedContentLength(int32_t ContentLength);
    ITERR       SetRenderedMimeType(IMG_type MimeType);

    /// Animation
    void SetAnimatedFlag(bool AnimatedFlag);
    bool GetAnimatedFlag() const;

     /// The layers to be stacked
```

【0068】

```
    typedef std::map<int32_t,Layer *> LayerMap;
    LayerMap                          mLayerMap;

      /// Viewport
      Viewport                        mViewport;

      /// Output filename
      std::string                     mOutFileName;

    /// Parameters that are set after rendering
    int32_t                           mRenderedWidth;
    int32_t                           mRenderedHeight;
    int32_t                           mRenderedContentLength;
    IMG_type                          mRenderedMimeType;

    /// temporary file streams for input media
    std::vector<LSCC::FileStream>     mFileStreams;

    UrlAccessFunction *               mUrlAccessFxn;
    void *                            mUrlAccessRef;

    // The enable which allows MIME types of image/gif to be decoded
    // using LZW decompression
    bool                              mEnableLzwGifDecode;

    // animation
    bool                              mAnimatedFlag;

    // Anchor to display mapping mode
    AnchorToDisplayMapMode            mAnchorToDisplayMapMode;

};
```

【００６９】

ｂ）層クラス

層クラスは、全ての層（画像、テキスト、その他）が導出される基本クラスである。

【００７０】

```
/**
  * class Layer
  **/
class Layer
{
```

【００７１】

```
public:
    /// Layer Type
    typedef enum
    {
        LAYER_TYPE_IMAGE,
        LAYER_TYPE_TEXT,
        LAYER_TYPE_BEZIER,
        LAYER_TYPE_ANIMATION,
        LAYER_TYPE_UNKNOWN
    } LayerType;

    /// Constructor
    Layer(class ImageTransform * imgXfm);

    /// Destructor
    virtual ~Layer();

    /// Get the type of layer
    virtual LayerType GetLayerType() const;

    /// Set the layer order - layers with a larger order number will
    /// be more visible when the layers are stacked (i.e. stacked
    /// later)
    void SetLayerNumber(int16_t number);

    /// Get the layer order number.
    int32_t GetLayerOrder() const;

    /// Set opacity
    ITERR   SetOpacity(double OpacityPercent);

    /// Get Opacity
    double  GetOpacity() const;

    /// Get aspect ratio
    virtual ITERR GetAspectRatio(double *pAspectRatio) const;

    /// Get the layers size (width and height)
    virtual ITERR GetSize(int32_t *pWidth, int32_t *pHeight) const;

    /// Decode a layer
```

【００７２】

```
virtual ITERR Load(const Viewport & viewport);

/// Size a layer
virtual ITERR Size(const Viewport & viewport);

/// Enhance
virtual ITERR Enhance();

/// EnhanceSize
virtual ITERR EnhanceSize();

/// Apply PreProcessing to accomodate viewport constraints
virtual ITERR PreProcess(const Viewport & viewport);

/// Render all the frames in a Layer
virtual ITERR Render(const Viewport & viewport);

/// Get the count of the number if frames this layer has
virtual uint32_t GetFrameCount() const;

/// Get a pointer to a particular frame
virtual const ImageFrame * GetFrame(uint32_t index) const;

/// Get the viewport Map
ViewportMap * GetViewportMap();

/// Set the identifier for this layer
void SetId(const std::string & id);

/// Get the identifier for this layer
std::string GetId() const;

/// Set the time to start displaying this frame (aka Time of
/// arrival [TOA]) - time is in ms
void SetStartTime(int32_t time);

/// Get the time to set for starting to displaying the frame
int32_t GetStartTime() const;

/// Set the duration this frame will be displayed for - time is in
/// ms
```

【0073】

```
    void SetDuration(int32_t time);

    /// Get the duration this frame will be displayed for.
    int32_t GetDuration() const;

    /// Set the display count for how many times to display this frame
    void SetDisplayCount(int32_t count);

    /// Get the display count for this frame.
    int32_t GetDisplayCount() const;

    /// Set the repeat period which is the duration between starting to
    /// reshow this frame
    void SetRepeatPeriod(int32_t time);

    /// Get the repeat period for this frame.
    int32_t GetRepeatPeriod() const;

    /// Is the layer "animated"
    bool IsAnimated() const;

protected:
    // Is it okay to Load a LZW GIF file
    bool IsLzwGifDecodeOK();

    // Fetch a "media" or other object and return a temp file name
    std::string FetchUrlObject(const std::string& url);

    /// Opacity of a layer
    double          mOpacity;

    /// Viewport mapping parameters
    ViewportMap     mViewportMap;

private:
    ImageTransform* mParentTransformObj;
    std::string     mLayerId;
    int16_t         mLayerNumber;

    uint32_t        mStartTime;     /// display start (presentatin) time
    uint32_t        mDuration;      /// display duration (in ms)
```

【００７４】

```
    uint32_t        mRepeatPeriod;  /// repeat period (in ms)
    uint32_t        mDisplayCount;  /// display count
};
```

【００７５】
ｃ）画像層クラス
「ＩｍａｇｅＬａｙｅｒ」は、層クラスから導出される。

【００７６】

```
/**
  * class ImageLayer
  **/
class ImageLayer : public Layer
{
public:
    /// Constructor
    ImageLayer(class ImageTransform * imgXfm);
    /// Destructor
    ~ImageLayer();

    /// return the layer type (i.e. LAYER_TYPE_IMAGE)
    LayerType GetLayerType() const;

    /// ------- Setting of parameters -------------
    /// Set the source file name
    ITERR SetSrc(const std::string & srcFileName);
    /// Set enhancement string
    ITERR SetEnhance(const std::string & enhanceString);

    /// ------- Getting of parameters -------------
    /// Get aspect ratio. Call only after image
    /// has been loaded.
    ITERR GetAspectRatio(double *pAspectRatio) const;
    ITERR GetSize(int32_t *pWidth, int32_t *pHeight) const;

    /// -------  Processing  -------------
    /// Set the Load Clamp Rectangle, i.e. the image that is loaded
    /// will be pre-clamped to ClampWidth, ClampHeight. This function
    /// is typically used to minimize processing overhead, as fewer
    /// pixels need be processed during subsequent processing.
    ITERR SetLoadClamp(int32_t ClampWidth, int32_t ClampHeight=0);
    /// Load a source image
    ITERR Load(const Viewport & viewport);
```

【0077】

```
        /// Size a layer
        ITERR Size(const Viewport & viewport);

        /// Apply enhancements
        ITERR Enhance();
        /// Compute the size effects of enhancements
        ITERR EnhanceSize();

        /// Apply PreProcessing to accomodate viewport "appearance"
        /// constraints, like color etc.
        ITERR PreProcess(const Viewport & viewport);
        /// Render a ImageLayer
        ITERR Render(const Viewport & viewport);

        /// Get the count of the number if frames this layer has
        uint32_t GetFrameCount() const;

        /// Get a pointer to a particular frame
        const ImageFrame * GetFrame(uint32_t index) const;

    private:

        /// Is this an LZW TIF Image?
        bool    IsLzwTIF(const std::string &filenam);
        /// Verify if this is a valid "allowed" image (for e.g. LZW
        /// may be disallowed and the image could be LZW GIF
        /// Also Compute the "preclamp" dimensions
        ITERR VerifyImageAndComputePreclamp(const std::string &pFileName,
                                int32_t          DisplayWidth,
                                int32_t          DisplayHeight,
                                int32_t          *pClampWidth,
                                int32_t          *pClampHeight);

        std::string mSrcFileName;
        int32_t     mLoadClampWidth;
        int32_t     mLoadClampHeight;
        std::string mEnhanceString;
        IMG_image   mImg;
        ImageFrame  mRenderedImage;
    };
```

【００７８】

ｄ）「Ｖｉｅｗｐｏｒｔ」クラス

【００７９】

```
/**
  * Class Viewport
  **/
class Viewport
{
public:

    /// Constructor
    Viewport(class ImageTransform * parent);
    /// Destructor
    ~Viewport();

    /// -------------Viewport initialization------------------
    /// Initialization
    ITERR       Init(){return ReInit();};
    /// Reinitialization
    ITERR       ReInit();

    ///-------------adaptive vs. custom palette
    bool        UseAdaptivePalette();

    /// -------------Viewport external params ------
    /// preprocessing parameter - sharpen
    ITERR       SetSharpen(double Sharpen);
    double      GetSharpen() const;

    /// adaptation: Variable params
    /// Only set the width
    ITERR       SetDisplaySize(int32_t Width);
    /// Set the width and height
    ITERR       SetDisplaySize(int32_t Width, int32_t Height);
    /// **WARNING*: This returns the raw device display size without
    /// considering any scaling.
    void        GetDisplaySize(int32_t *pWidth, int32_t *pHeight) const;
    /// **WARNING*: This returns the effective display size after
    /// considering any scaling.
    void        GetEffectiveDisplaySize(int32_t *pWidth, int32_t
*pHeight) const;

    /// scaling of display
```

【００８０】

```
    ITERR       SetDisplaySizeScale(double ScaleX, double ScaleY);
    void        GetDisplaySizeScale(double *pScaleX, double *pScaleY)
const;

    /// bits per pixel
    ITERR       SetBitsPerPixel(int32_t BitsPerPixel);
    int32_t     GetBitsPerPixel() const;

    /// Amount of error diffusion
    ITERR       SetDiffuseLevel(int32_t DiffuseLevel);
    int32_t     GetDiffuseLevel() const;

    /// quality level for JPEG output
    ITERR       SetJPEGQuality(int32_t JPEGQuality);
    int32_t     GetJPEGQuality() const;

    /// Maximum file size allowed
    ITERR       SetFileSize(int32_t FileSize);
    /// **WARNING*: This returns the raw device file size without
    /// considering any scaling.
    int32_t       GetFileSize() const;
    /// **WARNING*: This returns the effective file size after
    /// considering any scaling.
    ITERR       GetEffectiveFileSize(int32_t *pEffFileSize) const;
    ITERR       SetFileSizeScale(double FileSizeScale);
    double      GetFileSizeScale() const;

    /// Mime type for static (un-animated) output
    ITERR       SetMimeType(const std::string & mimeType);
    IMG_type    GetMimeType() const;

    /// Dots per inch of device
    ITERR       SetDPI(double DotsPerInch);
    double      GetDPI() const;

    /// Color capability of device
    ITERR       SetColorFlag(bool ColorFlag);
    bool        GetColorFlag() const;

    /// System Palette
    ITERR       SetSystemPalette(const std::string & sysPalFileName);
```

【００８１】

```
    char        *GetSystemPalette() const;

    /// Force color palette
    ITERR       SetForceColorPalette(const std::string & fCPalFileName);
    char        *GetForceColorPalette() const;

    /// Animation parameter: Mime type for animated output
    ITERR       SetAnimationMimeType(const std::string & mimeType);
    IMG_type    GetAnimationMimeType() const;

    /// Animation parameter: Animation capable?
    void        SetAnimationCapable(bool AnimationCapable);
    bool        GetAnimationCapable() const;

    /// Animation parameter: Animation Max Frames
    ITERR       SetAnimationMaxFrames(const std::string & MaxFrames);
    int32_t     GetAnimationMaxFrames() const;

    /// Animation parameter: Animation Max Repeat Count
    ITERR       SetAnimationMaxRepeatCount(const std::string &
MaxRepeatCount);
    int32_t     GetAnimationMaxRepeatCount() const;

    /// -------------Viewport: internal params ------
    ITERR       SetViewportSize(int32_t Width, int32_t Height = 0);
    void        GetViewportSize(int32_t *pWidth, int32_t *pHeight)
const;
    ITERR       SetIntBitsPerPixel(int32_t BitsPerPixel);
    int32_t     GetIntBitsPerPixel() const;
    ITERR       SetIntDiffuseLevel(int32_t DiffuseLevel);
    int32_t     GetIntDiffuseLevel() const;
    ITERR       SetIntJPEGQuality(int32_t JPEGQuality);
    int32_t     GetIntJPEGQuality() const;

    /// Aspect Layer
    ITERR       SetAspectLayerNumber(int32_t LayerNumber);
    int32_t     GetAspectLayerNumber() const;

    /// Mime type for output
    void        SetOutputMimeType(IMG_type mimeType);
    IMG_type    GetOutputMimeType() const;
```

【００８２】

```
    /// -------------Viewport save to memory-----------------
    ITERR       Save(IMG_IOHANDLER *pIO = NULL);

    /// Enable (or disable) encoding MIME type image/gif images
    /// compressed with the LZW algorithm
    void EnableLzwGifEncoding(bool enable = true);
    /// Is it okay to do LzwGifEncoding Okay?
    bool IsLzwGifEncodeOK() const;

    /// Add the frame to the image frame held by the viewport
    void AddFrame(const ImageFrame * frame);

private:

    ///----------- Viewport params: External--------
    ///         Preprocessing
    double              mSharpen;
    ///         adaptation: variable
    int32_t             mDisplayWidth;
    int32_t             mDisplayHeight;
    double              mDisplayScaleX;
    double              mDisplayScaleY;
    int32_t             mReqBitsPerPixel;
    int32_t             mReqDiffuseLevel;
    int32_t             mReqJPEGQuality;
    ///         adaptation: fixed
    bool                mColorFlag;
    int32_t             mFileSize;
    double              mFileSizeScale;
    IMG_type            mMimeType;
    double              mDPI;
    std::string         mFCPalFileName; ///force color palette
    std::string         mSysPalFileName;
    IMG_colorPalette    mPalette;
    bool                mJPEGThumbSave;
    int32_t             mJPEGThumbClamp;
    int32_t             mJPEGThumbQuality;

    ///   Animation parameters
    bool                mAnimationCapable;
```

【００８３】

```
    uint32_t            mAnimationMaxFrames;
    uint32_t            mAnimationMaxRepeatCount;
    IMG_type            mAnimationMimeType;

    /// Output Mime type: Output mime type is set to one of the
    /// mMimeType or mAnimationMimeType based on:
    /// If the image seq. to be rendered has more than one frame
    ///        and the device is animation capable:
    /// then set to mAnimationMimeType
    /// else use mMimeType.
    IMG_type            mOutputMimeType;

    ///----------- Viewport parameters: Internal-------
    ///         adaptation: variable
    int32_t             mViewportWidth;
    int32_t             mViewportHeight;
    int32_t             mBitsPerPixel;
    int32_t             mDiffuseLevel;
    int32_t             mJPEGQuality;

    ///  The layer that determines the aspect ratio of the viewport.
    ///  The significance of this is that the viewport coordinates
    ///  are now effectively normalized relative to this layer.
    int32_t             mAspectLayerNumber;

    /// Substitution for transparency for devices that do not support
transp.
    uint8               mTrans_R;
    uint8               mTrans_G;
    uint8               mTrans_B;

    /// Drawing Canvas
    double              mCanvasX;
    double              mCanvasY;
    double              mCanvasW;
    double              mCanvasH;

    FrameMap            mFrameMap;

    // The enable which allows MIME types of image/gif to be encoded
    // using LZW compression
```

【００８４】

```
    bool                mEnableLzwGifEncode;

    class ImageTransform * mParent;
};
```

【００８５】

１２．階層化の例

以下の副節は、ＸＭＬベースの階層化ＡＰＩの使用例を示す。

ａ）グラフィックの重ね合わせ

この例は、以下の制約の下でグラフィックをソース画像上に重ね合わせる方法を示す。

○画像は、ビューポートに「合わせ」られる。

○グラフィックは、ビューポートの右下隅に現状のままペーストされる。

要求しているＵＲＬは、以下のようになるであろう。

ｈｔｔｐ：／／ｅｓｗｉｔｃｈ．ｆｏｏ．ｃｏｍ／ｅｓ？ｓｒｃ＝ｈｔｔｐ；／／ｓｏｕｒｃｅ．ｆｏｏ．ｃｏｍ／ｂｏｙ．ｊｐｇ＆ｅｎｈ＝ｈｔｔｐ：／／ｓｏｕｒｃｅ．ｆｏｏ．ｃｏｍ／ｅｎｈａｎｃｅ．ｘｍｌ

拡張ＸＭＬは、以下のようになるであろう。

【００８６】

```
<image_transform xmlns="http://www.lightsurf.com/image_transform/1.0">
    <!-- Graphics layer-->
    <image_layer src=http://www.image.com/flower.png order="2">
        <Viewport_map mode="as-is">
            <align xalign="right" yalign="bottom" />
        </Viewport_map>
    </image_layer>
</image_transform>
```

【００８７】

ｂ）フレーミング

この節は、フレームを画像の上に重ねる例である。

要求しているＵＲＬは、以下のようになるであろう。

ｈｔｔｐ：／／ｅｓｗｉｔｃｈ．ｆｏｏ．ｃｏｍ／ｅｓ？ｅｎｈ＝ｈｔｔｐ：／／ｓｏｕｒｃｅ．ｆｏｏ．ｃｏｍ／ｅｎｈａｎｃｅ．ｘｍｌ

拡張ＸＭＬを以下に示す。

ビューポートの「ａｓｐｅｃｔ＿ｌａｙｅｒ」属性は、２に設定される。これによってビューポートは、強制的に画像層２、すなわち「ｉｍａｇｅ　ｌａｙｅｒ　２」と同じアスペクト比を有する。

「Ｉｍａｇｅ＿ｌａｙｅｒ　２」は、完全なビューポートにマップされる。

画像層１は、「花」に透明に整列するサブウィンドウにマップされる。

【００８８】

```
image_transform xmlns="http://www.lightsurf.com/image_transform/1.0">
   <!-- Image layer--->

  <image_layer src=http://www.image.com/boy.jpg order="1">
     <Viewport_map mode="fit">
       <window x="0.45" y="0.16" width="0.37" height="0.29"/>
     </Viewport_map>
  </image_layer>

  <!-- Graphics layer--->
  <image_layer src=http://www.image.com/frame.gif order="2">
  </image_layer>

  <!-- Force the anchor/aspect layer to be the "frame"--->
  <Viewport aspect_layer="2" />
</image_transform>
```

【００８９】

ｃ）テキストの重ね合わせ

この例は、テキストをビューポートの下部２０％の上に重ねる。

【００９０】

```
<image_transform xmlns="http://www.lightsurf.com/image_transform/1.0">
<!-- The text layer --->
   <text_layer order="2" text="hello world" fontfile="arial.ttf"
font_color="0x000000" font_size="12" font_size_min="6">
      <Viewport_map>
          <window x="0.0" y="0.8" width="1.0" height="0.2"/>
      </Viewport_map>
   </text_layer>
</image_transform>
```

【００９１】

Ｄ．内部作動の要約

１．全体的作動

図６Ａ～Ｂは、動的ビューポート階層化をサポートするために本発明によって使用される全体的な方法６００を示す流れ図を構成する。最初に、ストックＨＴＴＰサーバ（例えば、「Ａｐａｃｈｅ」サーバ）は、段階６０１に示すように、ターゲット画像を検索する（例えば、画像リピジトリーから）ための（ブラウザ）クライアントからのＵＲＬのようなオンライン要求（例えば、ＭＴＭＬ要求）によって呼び出される。クライアントからのこのＨＴＴＰ呼び出し（オンライン要求）は、ＵＲＬ及びヘッダ（クライアントブラウザ形式を識別するヘッダを含む）を備えた「ＨＴＴＰ　ＧＥＴ」指令を含む。ＵＲＬ自体は、例えば、位置及び付随する名前／値の対を特定する一般的なウェブベースのＵＲＬを含むことができる。クライアントがＨＰＰＴサーバを直接に呼び出した時、ＨＴＴＰサーバをシステムのフロントエンドと考えることができる。段階６０２に示すように、着信要求

をフォークするのにプラグインモジュール（「ｅＳｗｔｉｃｈ（登録商標）」ハンドラ）が使用される。ここで、「ｅＳｗｔｉｃｈ（登録商標）」ハンドラは、段階６０３に示すように、ブラウザクライアントを特定するために「ＨＴＴＰ　ＧＥＴ」ヘッダを検査し、この識別結果から、ハンドラは、クライアント装置の形式又はアイデンティティ（すなわち、装置タイプ）を推論することができる。この段階の作動中に、ハンドラは、例えば、先に参照した本出願人所有の２０００年６月６日出願の米国特許出願出願番号第０９／５８８，８７５号及び２００１年１１月８日出願の米国特許出願出願番号第１０／０１０，６１６号で説明するように、ヘッダを適切な装置と適合させるために装置データベースを調べる。

【００９２】

装置の識別後に、ハンドラは、続いて段階６０４においてＸＭＬ（構成）ファイルをフェッチする。クライアントによって提出された（段階６０１）ＵＲＬは、画像変換ツリーの値（これは、ビューポート及び層の両方を説明する）を階層的に記憶する特定のＸＭＬファイルを名前／値の対の１つとして指定する。段階６０５では、ストックＸＭＬパーサー（例えば、「ｌｉｂＸＭＬ２」）を使用し、フェッチされたＸＭＬファイルをここで構文解析することができる。その後、名前／値の属性は、画像変換ツリーのメモリ内コピーを作成するために使用される。

【００９３】

次の段階は、段階６０６に示すように、クライアントデータベースから導出されたビューポート情報を画像変換ツリー内の属性及びその値（例えば、階層化情報）の全てと合併させることである。段階６０７では、画像変換モジュールを呼び出すと、本方法は、先に進んで画像をレンダリングする（例えば、クライアントに対して最適化又はカスタマイズされたバージョンを動的に作成する）。特に、このような画像は、画像変換ツリー内の階層化及びビューポート情報に従って、特定されたクライアント装置のビューポートに対してレンダリングされる。画像を要求されたフォーマットに変換することにより、クライアントの任意の画像フォーマット考慮事項（例えば、ＪＰＥＧフォーマット要件）を適用することができる。先の処理は、反復的に行われる場合がある。例えば、動的に作成されたバージョンがクライアント装置に大きすぎるか又はクライアントの機能を超えるビット深さを有すると見なされた場合、適合するバージョンを作成するためにこの段階が繰り返される。所定の反復中に、クライアント装置に対して最適化される画像を要求があり次第生成するために、符号化／レンダリングパラメータ（例えば、画像寸法）を動的に調節することができる。最後に、段階６０８で示すように、本方法は、完全にレンダリングされた画像（制限に従って）を放出し、これは、次に、適切なフォーマットでクライアント装置に返信される（例えば、無線接続を通じて、インターネット接続を通じて、又は無線インターネット接続を通じてなど）。画像は、必要に応じて、今後の検索（例えば、同じ装置タイプによる）に対してキャッシュに入れることができる。

【００９４】

２．画像変換オブジェクト

ＸＭＬ記述を近密に反映する「画像変換オブジェクト」クラス定義（ｃｌａｓｓ　ＩｍａｇｅＴｒａｎｓｆｏｒｍ）は、様々な画像層の作成／サポートを担うデータメンバーを含む。各層自体は、その独自のオブジェクトを有する。「画像変換オブジェクト」のインスタンスが作成されると、組み込まれた全てのオブジェクトのインスタンスが同様に作成される。

「画像変換オブジェクト」は、「Ｒｅｎｄｅｒ」方法、すなわち、「Ｒｅｎｄｅｒ（）」を含む。基本的な作動において、「Ｒｅｎｄｅｒ」方法では、各層が正しくレンダリングされるように、対応するレンダリング方法が各組込オブジェクト上に呼び出される。レンダリングは、ビューポートのメモリ内のバージョン（例えば、正規フォーマット、ビットマップなど）、すなわち、ビューポートオブジェクトに対して行われる。最終的には、各組込オブジェクトは、「候補」レンダリング済み画像を生成するためにビューポートオブジェクトに対してレンダリングされる。次に、候補画像は、候補変換画像を生成するた

めにクライアントに適切なフォーマットに符号化される（例えば、ＪＰＥＧ符号化される）。候補画像が変換された状態で、得られる画像は、図４で上述したように適切な制限（例えば、ファイルサイズ）に順守しているかを検査される。例えば、完全にレンダリングされた画像がＪＰＥＧに変換された場合、得られるＪＰＥＧファイルは、ファイルが最大指定ファイルサイズを超える場合は、最終出力として許容可能なものではない。従って、ターゲットクライアントに適用可能な制限に従う最終画像ファイル生成するために、ビューポートを「再マッピング」して画像を再レンダリングする（必要であれば）段階を含んで、処理を反復することができる。内部的には、「ファイルサイズ制御」ブロックは、新しいファイルサイズを取得するために、異なる組の（制御）パラメータを推定する（例えば、ビューポートサイズ、ビット深さ、ＪＰＥＧ品質などの低減）。例えば、変換された候補画像のサイズが大きすぎる場合、本方法では、より小さいファイルサイズを有する変換された候補画像を生成するためのより小さな画面を有するビューポートをリセットすることができる。

【００９５】

本発明を１つの好ましい実施形態及びいくつかの代替形態を特に参照してある程度詳細に説明したが、本発明をその特定実施形態又はその特定代替形態に限定する意図はない。例えば、クライアント装置で画像を「表示すること」に着目する例が示された。当業者は、印刷のような他のクライアント側出力又はレンダリングが本発明の適用から恩典を得ることができることを認めるであろう。従って、当業者は、本発明の教示から逸脱することなく好ましい実施形態に修正を為し得ることを認めるものである。

【図面の簡単な説明】

【００９６】

【図１】本発明の実施に適切なデジタルカメラの非常に一般的なブロック図である。

【図２Ａ】従来のデジタル画像装置のブロック図である。

【図２Ｂ】デジタルカメラの作動を指図し、画像データを処理するために設けられた従来の搭載プロセッサ又はコンピュータのブロック図である。

【図３】本発明が好ましく組み込まれる例示的な無線接続環境を示すブロック図である。

【図４】良好な画像品質を維持しながらターゲット装置制限を満たすために使用される本発明の反復最適化／カスタマイズ化の方法を示す図である。

【図５Ａ】階層化ＡＰＩを示し、様々な層を組み合わせる方法を説明するために準備された図である。

【図５Ｂ】好ましく使用されるビューポート座標系を示す図である。

【図５Ｃ】本発明の「ＸＭＬ　ＡＰＩ」で使用されるオブジェクトの階層を示すグラフである。

【図６Ａ】動的ビューポート階層化をサポートする本発明によって使用される全体的方法を示す流れ図である。

【図６Ｂ】動的ビューポート階層化をサポートする本発明によって使用される全体的方法を示す流れ図である。

【符号の説明】

【００９７】

６００　本発明によって使用される全体的な方法
６０２　着信要求をフォークするのにプラグインモジュールを使用する段階
６０４　ＸＭＬ（構成）ファイルをフェッチする段階
６０７　画像をレンダリングする段階

【図１】

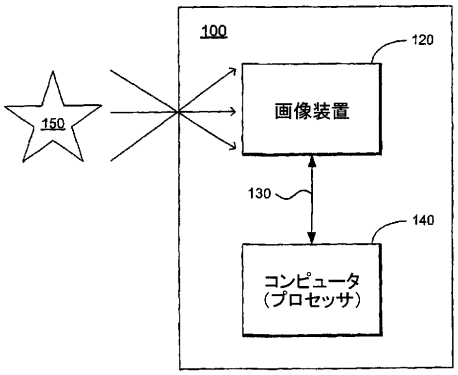

FIG. 1
（従来技術）

【図２Ａ】

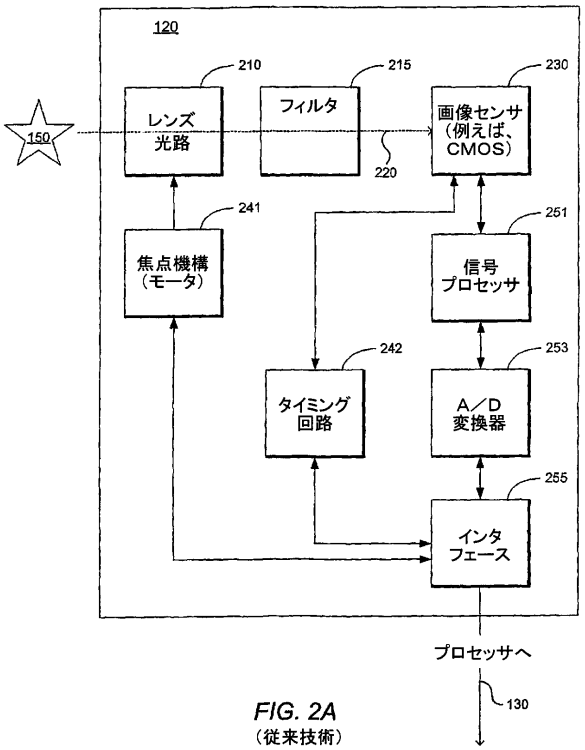

FIG. 2A
（従来技術）

【図２Ｂ】

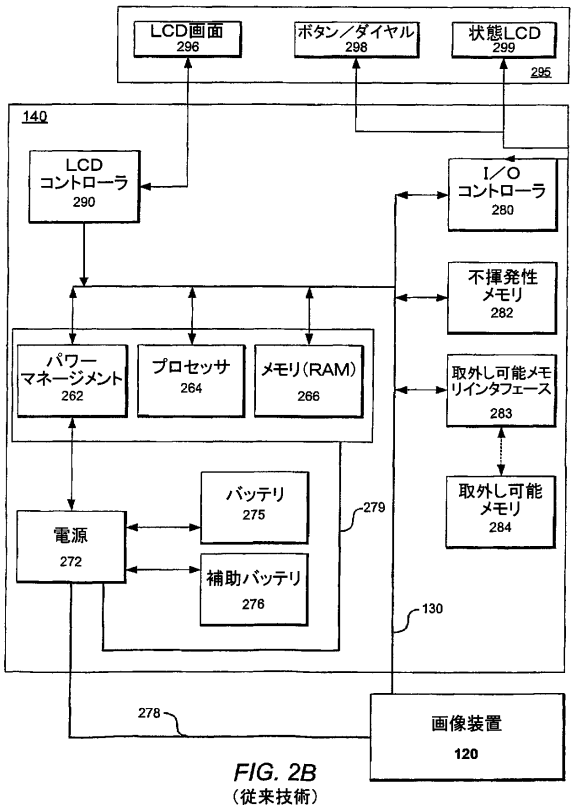

FIG. 2B
（従来技術）

【図３】

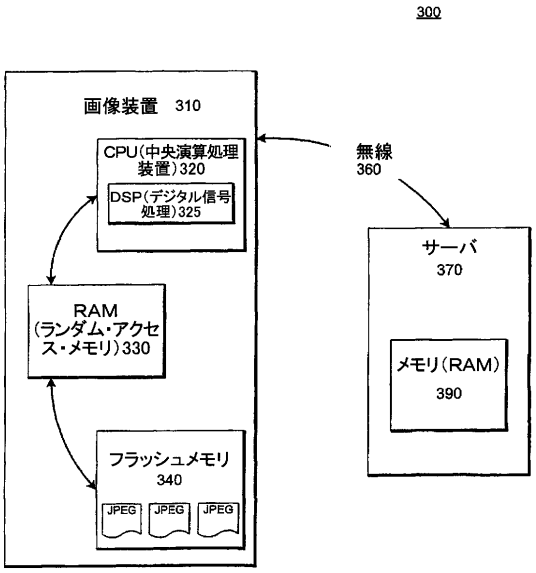

FIG. 3

【図４】

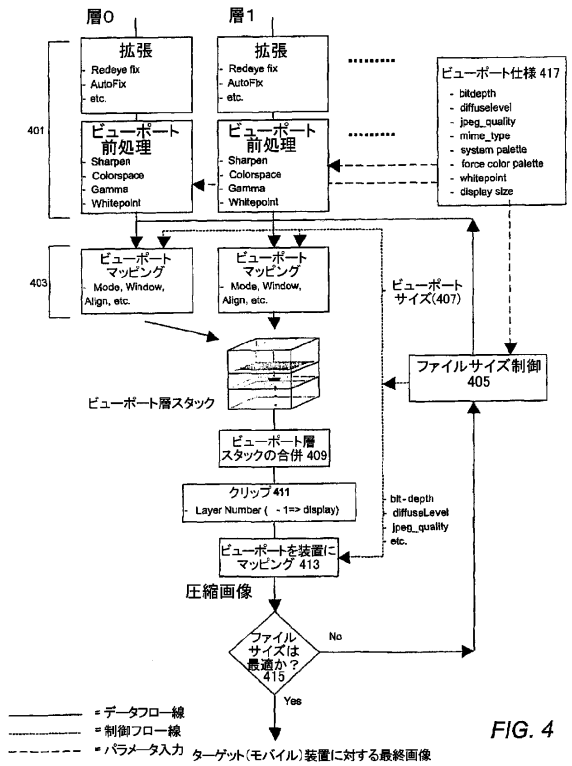

FIG. 4

【図５Ａ】

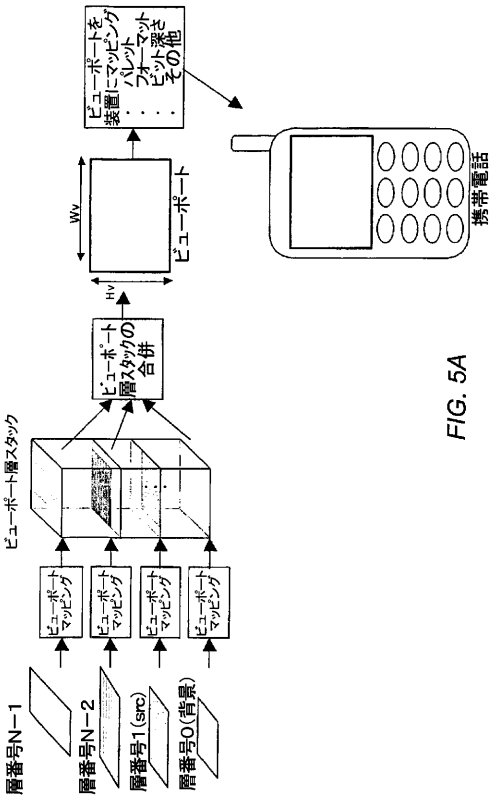

FIG. 5A

【図５Ｂ】

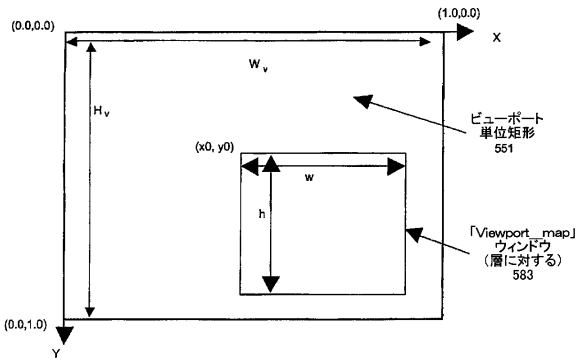

FIG. 5B

【図５Ｃ】

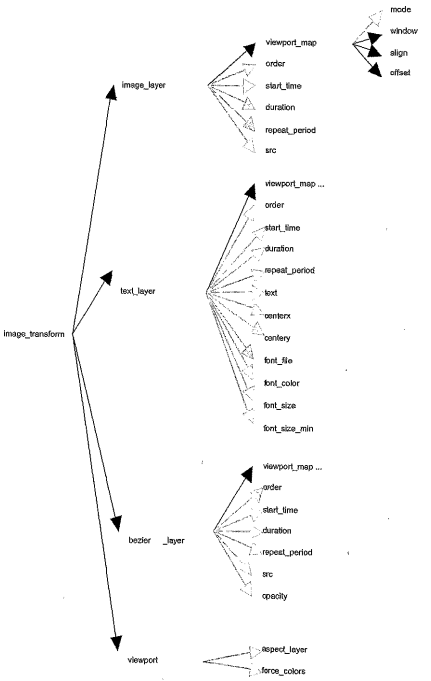

FIG. 5C

【図６Ａ】

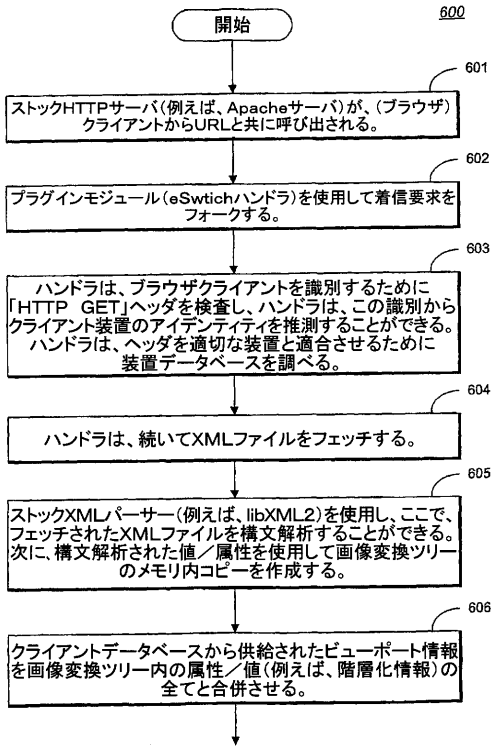

FIG. 6A

【図６Ｂ】

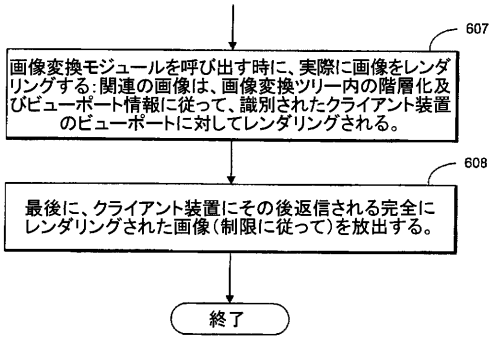

FIG. 6B

フロントページの続き

(74)代理人 100109335
弁理士 上杉 浩
(72)発明者 イースウォー ヴェンカット
アメリカ合衆国 カリフォルニア州 ９５０１４ クーパーティノ リンダ ヴィスタ ドライヴ １０７３６

合議体
審判長 大野 克人
審判官 清水 稔
審判官 鈴木 重幸

(56)参考文献 国際公開第０２／２７５４３（WO，Ａ２）
特開２００２－２０２９３５（ＪＰ，Ａ）
米国特許第６２２６６４２（ＵＳ，Ｂ１）
特開平１１－２３０７６９（ＪＰ，Ａ）
特開２０００－２５８１７２（ＪＰ，Ａ）
特開２００２－１９４６１３（ＪＰ，Ａ）
特開２００２－１３２１３１（ＪＰ，Ａ）
米国特許第６１６７４４１（ＵＳ，Ａ）
国際公開第０１／５７７１８（WO，Ａ２）
国際公開第０２／１５１２８（WO，Ａ１）
米国特許第６３９７２３０（ＵＳ，Ｂ１）
特開平５－６４００１（ＪＰ，Ａ）
特開２００１－３０６４４９（ＪＰ，Ａ）
特開２００２－７２７０（ＪＰ，Ａ）
特開２００１－３４５８９６（ＪＰ，Ａ）
アニメーションＧＩＦ対応の画像自動変換配信サーバー～ピクチャーＩＱの「ＴｒａｎｓＦｏｒｃｅＭｏｄｅｌ３００」，ｍｏｂｉｌｅ ｍｅｄｉａ ｍａｇａｚｉｎｅ，第７９巻，株式会社シーメディア，２００２年3月5日，ｐ．１４

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 13/00