(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6981753号
(P6981753)
(45)発行日　令和3年12月17日(2021.12.17)　(24)登録日　令和3年11月22日(2021.11.22)

(51)Int.Cl.　F I
G06T 15/00 (2011.01)　G06T 15/00 501

請求項の数 13　外国語出願　(全 17 頁)

(21)出願番号　特願2017-116(P2017-116)
(22)出願日　平成29年1月4日(2017.1.4)
(65)公開番号　特開2017-123171(P2017-123171A)
(43)公開日　平成29年7月13日(2017.7.13)
審査請求日　令和1年12月17日(2019.12.17)
(31)優先権主張番号　15307166.7
(32)優先日　平成27年12月29日(2015.12.29)
(33)優先権主張国・地域又は機関
欧州特許庁(EP)

(73)特許権者　500102435
ダッソー　システムズ
ＤＡＳＳＡＵＬＴ　ＳＹＳＴＥＭＥＳ
フランス国　７８１４０　ベリジー　ビラクブレー　リュ　マルセル　ダッソー　１０
(74)代理人　110001243
特許業務法人　谷・阿部特許事務所
(72)発明者　ビクトール　バシェ
フランス　９２１００　ブーローニュ　ビヤンクール　アベニュー　デ　ミモザ　２５

最終頁に続く

(54)【発明の名称】複数のグラフィックカードの管理

(57)【特許請求の範囲】
【請求項１】
複数のグラフィックカードを管理するためのコンピュータ実施方法であって、グラフィックカードは、１または複数のグラフィックプロセッシングユニットを備え、方法は、
レンダエンジンにシーンをロードするステップであって、前記シーンは、前記シーンのビューをレンダリングするために使用されるべき少なくとも１つのグラフィックデータを備える、ステップと、
前記少なくとも１つのグラフィックデータのグラフィックリソースに関して抽象グラフィックリソースを作成するステップであって、前記抽象グラフィックリソースは、グラフィックカードのうちの少なくとも１つに関する前記グラフィックリソースの識別子を記憶する、ステップと、
前記少なくとも１つのグラフィックカード上で、前記少なくとも１つのグラフィックデータの前記グラフィックリソースをコピーするステップと、
前記グラフィックリソースを扱うための前記抽象グラフィックリソースに対するアクセスを前記レンダエンジンに提供するステップと
を備えることを特徴とするコンピュータ実施方法。
【請求項２】
前記レンダエンジンは、少なくとも２つの論理層、前記レンダエンジンに対するアクセスをアプリケーションに提供する上位層、およびグラフィックライブラリに対するアクセスを前記レンダエンジンに提供する下位層を備え、前記抽象グラフィックリソースを作成

する前記ステップは、前記上位層と前記下位層の間に備えられた抽象層によって実行されることを特徴とする請求項１に記載のコンピュータ実施方法。
【請求項３】
前記抽象グラフィックリソースを作成する前記ステップの前に、
前記下位層による、グラフィックライブラリ上で、グラフィックカードのうちの前記少なくとも１つに関する前記グラフィックリソースの前記識別子にアクセスするステップと、
前記アクセスされた識別子を前記抽象層に提供するステップと
をさらに備えることを特徴とする請求項２に記載のコンピュータ実施方法。
【請求項４】
前記作成された抽象グラフィックリソースは、グラフィックリソースの前記識別子、およびグラフィックカードのうちの前記少なくとも１つの識別子を記憶することを特徴とする請求項１乃至３のいずれか１つに記載のコンピュータ実施方法。
【請求項５】
前記抽象グラフィックリソースを作成する前記ステップは、前記抽象グラフィックリソースをテーブルに記憶することをさらに備え、アクセスを前記レンダエンジンに提供する前記ステップは、前記グラフィックリソースを扱うための前記抽象グラフィックリソースを記憶する前記テーブルに対するアクセスを前記レンダエンジンに提供するステップを備えることを特徴とする請求項１乃至４のいずれか１つに記載のコンピュータ実施方法。
【請求項６】
前記グラフィックリソース上で実行されるべきグラフィックライブラリアクションを受け取るステップであって、前記アクションは、前記レンダエンジンに対するアクセスを有するアプリケーションによって要求される、該ステップと、
前記グラフィックリソースに関して作成された前記抽象グラフィックリソースを識別するステップと、
グラフィックカードのうちの少なくとも１つに関する前記グラフィックリソースの前記識別子を取り出すステップと、
前記グラフィックリソースにアクセスし、かつ前記グラフィックリソース上で前記グラフィックライブラリアクションを実行するステップと
をさらに備えることを特徴とする請求項１乃至５のいずれか１つに記載のコンピュータ実施方法。
【請求項７】
前記グラフィックリソースを削除する命令を受け取るステップであって、前記アクションは、前記レンダエンジンに対するアクセスを有するアプリケーションによって要求される、ステップと、
前記グラフィックリソースに関して作成された前記抽象グラフィックリソースを識別するステップと、
グラフィックカードのうちの前記少なくとも１つに関する前記グラフィックリソースの前記識別子を取り出すステップと、
前記グラフィックリソースにアクセスし、かつグラフィックカードのうちの前記少なくとも１つの前記グラフィックリソースを削除するステップと
をさらに備えることを特徴とする請求項６に記載のコンピュータ実施方法。
【請求項８】
前記抽象グラフィックリソースを前記上位層に提供するステップをさらに備えることを特徴とする請求項２又は３に記載のコンピュータ実施方法。
【請求項９】
抽象リソースを作成するステップの前に、使用されるべき前記グラフィックリソースを扱うための前記少なくとも１つのグラフィックカードを選択するステップをさらに備えることを特徴とする請求項１乃至８のいずれか１つに記載のコンピュータ実施方法。
【請求項１０】

前記抽象グラフィックリソースは、前記複数のグラフィックカードのうちの各グラフィックカードに関してグラフィックリソースの識別子を記憶し、前記グラフィックリソースは、前記複数のグラフィックカードのうちの各グラフィックカード上にコピーされることを特徴とする請求項１乃至９のいずれか１つに記載のコンピュータ実施方法。

【請求項１１】

請求項１乃至１０のいずれか１つに記載の方法を実行するための指示を備えることを特徴とするレンダエンジンコンピュータプログラム。

【請求項１２】

請求項１１に記載のレンダエンジンコンピュータプログラムが記録されていることを特徴とするコンピュータ可読ストレージメディア。

【請求項１３】

メモリおよびグラフィカルユーザインターフェースに結合された処理回路を備えるシステムであって、

前記メモリは請求項１１に記載のレンダエンジンコンピュータプログラムが記録されていることを特徴とするシステム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、コンピュータプログラムおよびコンピュータシステムの分野に関し、より詳細には、複数のグラフィックカードを管理するための方法、システム、およびプログラムに関する。

【背景技術】

【０００２】

３次元（３Ｄ）シーンをレンダリングするためのコンピュータグラフィックス技法は、コンピュータ画面、テレビ、プロジェクタなどのディスプレイデバイス上で３Ｄシーンを描くことを目的とする。３Ｄシーンをレンダリングすることは、３Ｄレンダリングとも呼ばれる。

【０００３】

３Ｄレンダリングのためのコンピュータグラフィックス技法は、互いに対話するハードウェア構成要素およびソフトウェア構成要素に依拠し、これらの構成要素は、３Ｄレンダリングに専用のアーキテクチャを形成する。このアーキテクチャのメインハードウェア構成要素は、いくつかのタイプの計算をより速くするように設計されたアクセラレータであるグラフィックカード（ＧＣ）である。ＧＣは、三角形をピクセルに変換することなどのグラフィックス計算に専用である。ＧＣは、１または複数のグラフィックプロセッシングユニット（ＧＰＵ）を備え、ＧＰＵは、ＧＣのグラフィックス計算を実行するチップである。ＧＣは、３Ｄレンダリングの結果を表示する１またはいくつかのディスプレイデバイスに接続され得る。ＧＣおよびＧＰＵに対する指示は、モニタにコンピュータグラフィックスをレンダリングすることを支援するように設計されたコンピュータプログラムライブラリであるグラフィックライブラリ（ＧＬ）を介して送られる。ＧＬは、ＧＣをホストするコンピュータの中央処理装置（ＣＰＵ）によって実行される（executed）（または実行される（run））。２つの最も有名なＧＬは、ＤｉｒｅｃｔＸ（Ｃ）およびＯｐｅｎＧＬ（Ｃ）である。実際には、ＧＬは、ハードウェア製造業者によってではなく、ハードウェア製造業者によって提供されるハードウェア仕様によりＧＬを開発するサードパーティによって書かれる。レンダエンジン（ＲＥ）は、３Ｄシーンを入力として取り、１または複数のＧＣを使用して（マルチグラフィックカードレンダリングの場合に）画面にそれを描くフレームワークである。３Ｄシーンは、ＲＥフレームワークを使用するアプリケーションによって作成される。ＲＥは、アプリケーションによって提供される情報をグラフィックリソース（ＧＲ）に変換する。ＧＲは、バッファ、テクスチャなどの操作が実行され得るＧＬによって与えられるオブジェクトである。グラフィックドライバ（ＧＤ）は、オペレーティングシステム（ＯＳ）とＧＣの間の通信を可能にするインターフェースを提供す

るコンピュータプログラムである。実際には、ＧＤは、ＧＣ製造業者によって提供される。

【０００４】

マルチＧＣレンダリングは、アプリケーションが、同一のマザーボードに差し込まれたいくつかのＧＣを使用してシーンをレンダリングする能力である。マルチＧＣ原理は最新であるにもかかわらず、それには、ＲＥフレームワークにおける多くの複雑さがかかわるため、少数のアプリケーションしかそれを使用しない。

【０００５】

マルチグラフィックカードレンダリングを実現する２つの一般的な技法が存在する。両方が、グラフィックドライバ作業に依拠し、レンダエンジンフレームワークには依拠しない。第１の技法は、ＡＭＤ（商標）によって開発されたｎＶＩＤＩＡ（商標）またはＣｒｏｓｓＦｉｒｅ（商標）によって開発されたＳＬＩ（商標）を使用する。基本的に、アプリケーションのＲＥが１つのフレームをレンダリングしている場合、ＲＥは、コンピュータ内に１つだけのＧＣが存在するかのようにＧＬを使用する。ＲＥは、いくつかのＧＣが存在することを認識していない。ＧＤは、命令を受け取り、それらをＧＣに拡散する。これは、２つのＧＣを有するコンピュータの例を示す図１上に示され、１つのＧＣが、画面の上側部分をレンダリングし、第２のＧＣが下側部分をレンダリングする。ＲＥは、アプリケーションから、ＧＬによって受け取られるＧＲに変換される命令（例えば、バッファ、テクスチャなど）を受け取り、ＧＬが、１つのＧＣに関する命令を送る。２つのカード（ＧＣ１、ＧＣ２）を管理するＧＤが、ＳＬＩ（商標）／ＣｒｏｓｓＦｉｒｅ（商標）技法を使用して２つのＧＣに関する命令を作成して、各ＧＣの各ＧＰＵ（ＧＣ１のＧＰＵ１、ＧＣ２のＧＰＵ２）がどのような情報を処理すべきかを知るようにする。

【０００６】

第２の技法は、第１の技法に基づき、モザイクモードと呼ばれる。モザイクモードは、ディスプレイドライバを使用して結果を複数の画面に拡散させて、より高い解像度でレンダリングする能力を表す。前述した第１の技法とモザイクモードを組み合わせることが、各ＧＣが別個のディスプレイを扱う能力を提供する。

【０００７】

これらの第１の技法および第２の技法は、主として、ゲームアプリケーションにおいて使用されてパフォーマンスを向上させ、または飛行シミュレータにおいて使用されて、適度なパフォーマンスで複数のディスプレイに出力する。

【０００８】

これらの２つの技法に加えて、別の技法が、ＧＬによってレンダリングフィーチャを開示することによって特定のマルチＧＣを提供する。しかし、この技術は、実験的であり、ハードウェア製造業者およびソフトウェア製造業者によって活用されていない。

【０００９】

マルチＧＣレンダリングのこれらの技法は、いくつかの欠点を抱える。第１の欠点は、それらが特定のシナリオに、マルチ画面レンダリングに関してモザイクモードに、ビデオゲームに関してＳＬＩ（商標）／ＣｒｏｓｓＦｉｒｅ（商標）に限定されることである。事実、これらの技法は、ＧＣに命令をディスパッチするのにＧＤ能力に依拠するが、ＧＤは、各時点で、ディスパッチを実行するために要求されるすべての情報を知っているわけではなく、したがって、これらのソリューションは、アプリケーション開発者によって保持された、いくつかのシナリオにしか当てはまらない。例えば、保持されないシナリオにおいて、１つのＧＣだけが使用され、アプリケーションは、その他のＧＣの計算リソースの恩恵を受けることができない。

【００１０】

別の限界は、これらの技法が、レンダリングされる３Ｄシーン上の１つの視点に限定されることである。このため、視点の変化の際に３Ｄシーンの表示速度を向上させる、いくつかの視点を並行に計算するためにＧＰＵの計算リソースを活用することが可能でない。

【００１１】

さらなる限界は、特定の命令を特定のＧＣにアドレス指定することが可能でなく、この特定の命令をＲＥに開示することが可能でないことである。前段で説明されるとおり、ＲＥは、いくつかのＧＣが存在することを認識していない。

【００１２】

このコンテキスト内で、複数のＧＣの向上された管理の必要性が依然として存在する。特に、ＧＣは、いくつかのディスプレイデバイス上に複数の視点で３Ｄシーンをレンダリングすることを可能にする。

【発明の概要】

【課題を解決するための手段】

【００１３】

したがって、複数のグラフィックカードを管理するためのコンピュータ実施方法が提供される。グラフィックカードは、１または複数のグラフィックプロセッシングユニットを備える。方法は、レンダエンジンにシーンをロードするステップであって、シーンは、シーンのビューをレンダリングするために使用されるべき少なくとも１つのグラフィックデータを備える、ステップ、少なくとも１つのグラフィックデータのグラフィックリソースに関して抽象グラフィックリソースを作成するステップであって、抽象グラフィックリソースは、グラフィックカードのうちの少なくとも１つに関するグラフィックリソースの識別子を記憶する、ステップ、前記少なくとも１つのグラフィックカード上で、少なくとも１つのグラフィックデータの前記グラフィックリソースをコピーするステップ、前記グラフィックリソースを扱うための抽象グラフィックリソースに対するアクセスをレンダエンジンに提供するステップを備える。

【００１４】

方法は、以下のうちの１または複数を備えることが可能である：

－　レンダエンジンが、少なくとも２つの論理層、レンダエンジンに対するアクセスをアプリケーションに提供する上位層、およびグラフィックライブラリに対するアクセスをレンダエンジンに提供する下位層を備え、抽象グラフィックリソースを作成することは、上位層と下位層の間に備えられた抽象層によって実行される、

－　抽象グラフィックリソースを作成するステップの前に、下位層による、グラフィックライブラリ上で、グラフィックカードのうちの少なくとも１つに関するグラフィックリソースの識別子にアクセスするステップ、およびアクセスされた識別子を抽象層に提供するステップを備える、

－　作成された抽象グラフィックリソースが、グラフィックリソースの識別子、およびグラフィックカードのうちの少なくとも１つの識別子を記憶する、

－　抽象グラフィックリソースを作成するステップは、抽象グラフィックリソースをテーブルに記憶することをさらに備え、アクセスをレンダエンジンに提供するステップは、前記グラフィックリソースを扱うための抽象グラフィックリソースを記憶するテーブルに対するアクセスをレンダエンジンに提供することを備える、

－　前記グラフィックリソース上で実行されるべきグラフィックライブラリアクションを受け取るステップであって、アクションは、レンダエンジンに対するアクセスを有するアプリケーションによって要求される、ステップ、前記グラフィックリソースに関して作成された抽象グラフィックリソースを識別するステップ、グラフィックカードのうちの少なくとも１つに関するグラフィックリソースの識別子を取り出すステップ、および、グラフィックリソースにアクセスし、かつグラフィックリソース上でグラフィックライブラリアクションを実行するステップを備える、

－　前記グラフィックリソースを削除する命令を受け取るステップであって、アクションは、レンダエンジンに対するアクセスを有するアプリケーションによって要求される、ステップ、前記グラフィックリソースに関して作成された抽象グラフィックリソースを識別するステップ、グラフィックカードのうちの少なくとも１つに関するグラフィックリソースの識別子を取り出すステップ、およびグラフィックリソースにアクセスし、かつグラフィックカードのうちの少なくとも１つのグラフィックリソースを削除するステップを備

える、
－　抽象グラフィックリソースを上位層に提供するステップを備える、
－　抽象リソースを作成することの前に、使用されるべき少なくとも１つのグラフィックリソースを扱うための少なくとも１つのグラフィックカードを選択するステップを備える、
－　抽象グラフィックリソースが、複数のグラフィックカードのうちの各グラフィックカードに関してグラフィックリソースの識別子を記憶し、グラフィックリソースは、複数のグラフィックカードのうちの各グラフィックカード上にコピーされる。
【００１５】
方法を実行するための指示を備えるレンダエンジンコンピュータプログラムがさらに提供される。
【００１６】
レンダエンジンコンピュータプログラムが記録されているコンピュータ可読ストレージメディアがさらに提供される。
【００１７】
メモリおよびグラフィカルユーザインターフェースに結合された処理回路を備えるシステムであって、メモリには、レンダエンジンコンピュータプログラムが記録されているシステムがさらに提供される。
【図面の簡単な説明】
【００１８】
次に、本発明の実施形態が、非限定的な例として、添付の図面を参照して説明される。
【図１】マルチＧＣ表示を実行するための従来技術の方法を示す流れ図である。
【図２】本発明の例を示す流れ図である。
【図３】抽象グラフィックリソースの作成の例を示す図である。
【図４】抽象グラフィックリソースの管理のためのテーブルの例を示す図である。
【図５】グラフィックデータに対して実行される操作の例を示す図である。
【図６】コンピュータシステムの例を示す図である。
【発明を実施するための形態】
【００１９】
図２の流れ図を参照すると、シーンをレンダリングするために使用される複数のグラフィックカード（ＧＣ）を管理するためのコンピュータ実施方法が提案される。ＧＣは、１または複数のグラフィックプロセッシングユニット（ＧＰＵ）を備え得る。方法は、レンダエンジンにシーンをロードすることを備える。シーンは、３次元（３Ｄ）シーンであることが可能である。シーンは、シーンのビューをレンダリングするために使用される１または複数のグラフィックデータを備える。方法は、その少なくとも１つのグラフィックデータに関して抽象グラフィックリソースを作成することをさらに備える。抽象グラフィックリソースは、グラフィックカードのうちの少なくとも１つに関するグラフィックリソースの識別子を記憶する。方法は、少なくとも１つのグラフィックカード上に、例えば、グラフィックカードのメモリ上にグラフィックデータのうちの少なくとも１つのグラフィックリソースをコピーすることも備える。次に、方法は、グラフィックリソースを扱うための抽象グラフィックリソースに対するアクセスをレンダエンジン（ＲＥ）に提供することを備える。
【００２０】
そのような方法は、現在のグラフィックライブラリ（ＧＬ）を使用しながら、複数のグラフィックカード（ＧＣ）の管理を向上させる。特に、本発明は、グラフィックリソース（ＧＲ）が、所与のグラフィックカードに関してグラフィックライブラリによって作成される場合、グラフィックリソースが、リソース名と呼ばれ得るグラフィックカード特有の名前を得ることである、大きな問題を回避する抽象グラフィックリソースの使用のお陰で、特定のＧＣに視点を結び付けることを可能にする。例えば、アプリケーションが、２つのグラフィックカード上で２つの異なる視点から木を描く場合、その木のグラフィックリ

ソースは、両方のグラフィックカード上にある必要がある。第１のグラフィックカード上で、バッファが、リソース名Ｎ１を有し、第２のものにおいて、リソース名Ｎ２を有する。このことは、レンダエンジンの各部分が、複雑さを扱い、両方のグラフィックカードに対処するのにすべてのレンダリング命令を複製しなければならないため、多くの複雑さをもたらす。次に、８つのグラフィックカードを有する同一の例を取ると、単一のオブジェクト「木」に関する８つのリソース名を管理することが非常に難しくなる。通常のシーンにおいて、数千のグラフィックオブジェクトが存在し、このため、数千倍、扱うべきグラフィックカードリソース名の数が存在する。本発明は、各グラフィックリソースに関して抽象グラフィックリソースを使用して、レンダエンジンが、リソース名を操作する代わりに、グラフィックデータに関する抽象グラフィックリソースを操作するだけであるようにする。このため、抽象グラフィックリソースに対するアクセスがレンダエンジンに与えられることを所与として、各内部命令（例えば、特定の表示をレンダエンジンに要求するアプリケーションの）は、グラフィックカード特有のリソース名をそれにマッチする抽象ハンドルによって置き換えることによって変更される。レンダエンジンは、前述の複雑さにもはや対処せず、それが受け取るすべてのレンダリング命令をもはや複製する必要がない。さらなる利点が、後段において説明される。

【００２１】

方法は、コンピュータ実施される。このことは、方法のステップ（または実質的にすべてのステップ）が、少なくとも１つのコンピュータ、または任意のシステムによって同様に実行されることを意味する。このため、方法のステップは、コンピュータによって、場合により、完全に自動的に、または半自動的に実行される。例において、方法のステップの少なくともいくつかをトリガすることは、ユーザ－コンピュータ対話を介して実行されることが可能である。要求されるユーザ－コンピュータ対話のレベルは、予測される自動性のレベルに依存し、ユーザの所望を実施する必要性とバランスがとられることが可能である。例において、このレベルは、ユーザ定義されること、および／または事前定義されることが可能である。

【００２２】

方法のコンピュータ実施の通常の例は、この目的のために適応されたシステムで方法を実行することである。システムは、メモリに結合されたプロセッサと、複数のグラフィックカードとを備え得る。レンダエンジン、およびシーンを表すデータが、メモリ上に記憶されることが可能であり、レンダエンジンは、ＣＰＵ上で動作することが可能である。より一般的に、メモリには、方法を実行するための指示を備えるコンピュータプログラムが記録されていることが可能である。メモリは、データベースを記憶することも可能である。メモリは、場合により、いくつかの物理的な別々の部分（例えば、プログラムに関して１つ、場合により、データベースに関して１つ）を備える、そのようなストレージのために適応された任意のハードウェアである。「データベース」によって意味されるのは、探索および取出しのために編成されたデータ（すなわち、情報）の任意の集まり（例えば、所定の構造化言語、例えば、ＳＱＬに基づく、例えば、リレーショナルデータベース）である。メモリ上に記憶される場合、データベースは、コンピュータによる迅速な探索および取出しを可能にする。データベースは、事実、様々なデータ処理操作に関連してデータの記憶、取出し、変更、および削除を容易にするように構造化される。データベースは、その各々が１または複数のフィールドから成るレコードに細分され得るファイル、またはファイルのセットから成ることが可能である。フィールドは、データストレージの基本単位である。ユーザは、主にクエリを介してデータを取り出すことが可能である。キーワードおよびソートコマンドを使用して、ユーザは、使用されているデータベース管理システムの規則によりデータの特定の集合を取り出すこと、またはそのような集合に関するレポートを作成することを行うように多くのレコードにおけるフィールドを探索すること、再配置すること、グループ化すること、および選択することを迅速に行うことができる。

【００２３】

方法は、グラフィックデータを操作し、例えば、グラフィックデータは、モデル化され

たオブジェクトであることが可能である。モデル化されたオブジェクトは、異なる種類のデータ、例えば、ＣＡＤオブジェクト、ＰＬＭオブジェクト、ＰＤＭオブジェクト、ＣＡＥオブジェクト、ＣＡＭオブジェクト、ＣＡＤデータ、ＰＬＭデータ、ＰＤＭデータ、ＣＡＭデータ、ＣＡＥデータによって定義されることが可能である。グラフィックデータは、モデル化されたオブジェクトを表すための任意のデータである。

【００２４】

例えば、ＣＡＤシステムを使用することによって獲得される３Ｄシーンのコンテキストにおいて、モデル化されたオブジェクトは、通常、例えば、部分などの製品、もしくは部分のアセンブリ、または、場合により、製品のアセンブリを表す３Ｄモデル化されたオブジェクトであることが可能である。「３Ｄモデル化されたオブジェクト」によって意味されるのは、その３Ｄ表現を可能にするデータによってモデル化される任意のオブジェクトである。３Ｄ表現は、すべての角度から部分を見ることを可能にする。例えば、３Ｄモデル化されたオブジェクトは、３Ｄ表現された場合、その軸のいずれかを中心に、またはその表現が表示される画面における任意の軸を中心に扱われ、回転させられることが可能である。このことは、特に、３Ｄモデル化されていない、２Ｄアイコンを排除する。３Ｄ表現の表示は、設計を容易にする（すなわち、設計者が彼らのタスクを統計的に実現する速度を増加させる）。製品の設計は、製造プロセスの一部であるので、このことは、産業における製造プロセスをスピードアップする。３Ｄモデル化されたオブジェクトは、例えば、ＣＡＤソフトウェアソリューションまたはＣＡＤシステムでその仮想設計を完了したことに続いて、現実世界において製造されるべき、（機械）部分、もしくは部分のアセンブリ、またはより一般的に任意の剛体アセンブリ（例えば、モバイル機構）などの製品の形状を表すことが可能である。ＣＡＤソフトウェアソリューションは、航空宇宙、建築、建設、消費財、ハイテクデバイス、産業機器、輸送、海洋、および／または海上石油／ガス生産もしくは輸送を含む、様々な、限定されない産業分野における製品の設計を可能にする。このため、方法によって設計される３Ｄモデル化されたオブジェクトは、地上の乗り物の部分（例えば、自動車機器および軽トラック機器、レース用自動車、オートバイ、トラックおよびモータ機器、トラックおよびバス、列車を含む）、空の乗り物の部分（例えば、機体機器、航空宇宙機器、推進機器、防衛用製品、航空会社機器、宇宙機器を含む）、海軍の乗り物の部分（例えば、海軍機器、商船、海上機器、ヨットおよび作業船、船舶機器を含む）、一般的な機械部分（例えば、産業製造機械、重量のあるモバイル機械もしくは機器、インストールされた機器、産業機器製品、製造された金属製品、タイヤ製造製品を含む）、電子機械部分もしくは電子部分（例えば、家庭用電子機器、セキュリティ製品および／または制御製品および／または計器製品、計算機器および通信機器、半導体、医療デバイスおよび医療機器を含む）、消費財（例えば、家具、住宅製品および庭製品、レジャー製品、ファッション製品、耐久消費財小売業者の製品、織物製品小売業者の製品を含む）、パッケージング（例えば、食品パッケージングおよび飲料パッケージングおよびタバコパッケージング、美容パッケージングおよびパーソナルケアパッケージング、世帯製品パッケージングを含む）などの任意の機械部分であることが可能な産業製品を表すことが可能である。

【００２５】

ＰＬＭによってさらに意味されるのは、物理的な製造される製品（または製造されるべき製品）を表現するモデル化されたオブジェクトの管理のために適応された任意のシステムである。このため、ＰＬＭシステムにおいて、モデル化されたオブジェクトは、物理的なオブジェクトの製造に適したデータによって定義される。これらは、通常、寸法値および／または公差値であることが可能である。オブジェクトの正しい製造のために、事実、そのような値を有する方が良い。

【００２６】

ＣＡＭによってさらに意味されるのは、製品の製造データを管理するために適応された、ソフトウェアまたはハードウェアである、任意のソリューションである。製造データは、一般に、製造すべき製品、製造プロセス、および要求されるリソースと関係するデータ

を含む。ＣＡＭソリューションは、製品の製造プロセス全体を計画し、最適化するのに使用される。例えば、それは、ＣＡＭユーザに実現可能性、製造プロセスの持続時間、または接続プロセスの特定のステップにおいて使用されることが可能な特定のロボットなどのリソースの数に関する情報を提供することができ、このため、管理または要求される投資に関する決定を可能にする。ＣＡＭは、ＣＡＤプロセスおよび潜在的なＣＡＥプロセスの後の後続のプロセスである。そのようなＣＡＭソリューションは、商標、ＤＥＬＭＩＡ（登録商標）の下でダッソーシステムズ社によって提供される。

【００２７】

ＣＡＥによってさらに意味されるのは、モデル化されたオブジェクトの物理的な振舞いの解析のために適応された、ソフトウェアまたはハードウェアである、任意のソリューションである。よく知られ、広く使用されるＣＡＥ技法が、モデル化されたオブジェクトを、物理的な振舞いが数式を介して計算され、かつシミュレートされ得る要素に分割することが通常、かかわる有限要素法（ＦＥＭ）である。そのようなＣＡＥソリューションは、商標、ＳＩＭＵＬＩＡ（登録商標）の下でダッソーシステムズ社によって提供される。別の成長しているＣＡＥ技法には、ＣＡＤ形状データなしに物理の様々な分野からの複数の構成要素から構成される複雑なシステムのモデル化および解析がかかわる。ＣＡＥソリューションは、製造すべき製品のシミュレーションを可能にし、このため、製造すべき製品の最適化、向上、および検証を可能にする。そのようなＣＡＥソリューションは、商標、ＤＹＭＯＬＡ（登録商標）の下でダッソーシステムズ社によって提供される。

【００２８】

ＰＤＭは、製品データ管理を表す。ＰＤＭソリューションによって意味されるのは、特定の製品と関係するすべてのタイプのデータを管理するために適応された、ソフトウェアまたはハードウェアである、任意のソリューションである。ＰＤＭソリューションは、製品のライフサイクルにかかわるすべての当事者、すなわち、主にエンジニアであるが、管理者、財務の人々、販売の人々、および購入者も含む当事者によって使用されることが可能である。ＰＤＭソリューションは、一般に、製品指向のデータベースに基づく。それは、当事者が、彼らの製品に関する一貫性のあるデータを共有することを可能にし、したがって、当事者が相違するデータを使用するのを防止する。そのようなＰＤＭソリューションは、商標、ＥＮＯＶＩＡ（登録商標）の下でダッソーシステムズ社によって提供される。

【００２９】

図３は、システムの例を示し、システムは、コンピュータシステム、例えば、ユーザのワークステーションである。

【００３０】

例のシステムは、内部通信バス１０００に接続された中央処理装置（ＣＰＵ）１０１０と、バスにやはり接続されたランダムアクセスメモリ（ＲＡＭ）１０７０とを備える。システムには、バスに接続されたビデオランダムアクセスメモリ１１００に関連付けられるグラフィカルプロセッシングユニット（ＧＰＵ）１１１０がさらに提供される。ビデオＲＡＭ１１００は、当技術分野においてフレームバッファとしても知られる。大容量ストレージデバイスコントローラ１０２０が、ハードドライブ１０３０などの大容量メモリデバイスに対するアクセスを管理する。コンピュータプログラム指示およびデータを有形で実現するのに適した大容量メモリデバイスは、例として、ＥＰＲＯＭ、ＥＥＰＲＯＭ、およびフラッシュメモリデバイスなどの半導体メモリデバイス、内部ハードディスクおよびリムーバブルディスクなどの磁気ディスク、光磁気ディスク、ならびにＣＤ－ＲＯＭディスク１０４０を含む、すべての形態の不揮発性メモリを含む。以上のいずれも、特別に設計されたＡＳＩＣ（特定用途向け集積回路）によって補足されること、またはそのようなＡＳＩＣに組み込まれることが可能である。ネットワークアダプタ１０５０が、ネットワーク１０６０に対するアクセスを管理する。システムは、カーソル制御デバイス、キーボード、または類似したものなどの触覚デバイス１０９０も含むことが可能である。カーソル制御デバイスが、システムにおいて、ユーザが、ディスプレイ１０８０上の任意の所望さ

れる位置にカーソルを選択的に位置付けることを許すために使用される。さらに、カーソル制御デバイスは、ユーザが様々なコマンドを選択すること、および制御信号を入力することを可能にする。カーソル制御デバイスは、システムに制御信号を入力するためのいくつかの信号生成デバイスを含む。通常、カーソル制御デバイスは、マウスであることが可能であり、マウスのボタンが信号を生成するのに使用される。あるいは、またはさらに、システムは、センシティブパッドおよび／またはセンシティブ画面を備えることが可能である。

【００３１】

コンピュータプログラムは、コンピュータ、例えば、図３のシステムによって実行可能な指示を備えることが可能である。コンピュータプログラムの指示は、前述のシステムに方法を実行させる。プログラムは、システムのメモリを含む、任意のデータストレージメディア上に記録可能であることが可能である。プログラムは、例えば、デジタル電子回路において、もしくはコンピュータハードウェア、コンピュータファームウェア、コンピュータソフトウェアにおいて、またはそれらの組合せにおいて実施されることが可能である。プログラムは、装置、例えば、プログラマブルプロセッサによって実行されるようにマシン可読ストレージデバイスにおいて有形で実現された製品として実施されることが可能である。方法ステップは、入力データを操作すること、および出力を生成することによって方法の機能を実行するように指示のプログラムを実行するプログラマブルプロセッサによって実行されることが可能である。このため、プロセッサは、プログラマブルであって、データストレージシステム、少なくとも１つの入力デバイス、および少なくとも１つの出力デバイスからデータおよび指示を受け取るように、かつそれらにデータおよび指示を伝送するように結合されることが可能である。アプリケーションプログラムは、高レベルの手続きプログラミング言語もしくはオブジェクト指向プログラミング言語において、または所望される場合、アセンブリ言語もしくはマシン言語において実施されることが可能である。いずれにしても、言語は、コンパイラ型言語またはインタプリタ型言語であることが可能である。プログラムは、完全インストールプログラムまたは更新プログラムであることが可能である。システムにプログラムを適用することは、いずれにしても、方法を実行するための指示をもたらす。

【００３２】

図２の流れ図を再び参照すると、本発明による複数のグラフィックカードを管理するためのコンピュータ実施方法の例が説明される。

【００３３】

ステップＳ１００において、３次元シーンが、レンダエンジン（ＲＥ）にロードされる。レンダエンジン（レンダリングエンジンとも呼ばれる）は、アプリケーションの要求があると表示されるべき画像を生成するフレームワークである。例えば、ＣＡＤシステムのＣＡＤアプリケーションが、入力として３Ｄモデル化されたオブジェクトの３Ｄシーンをレンダエンジンに提供し、レンダエンジンが、ＣＡＤシステムの１または複数のグラフィックカードを使用して画面に３Ｄシーンを描く。フレームワークは、アプリケーションが表示されることを要求するデータを入力として取り、かつ表示され得る画像を出力するソフトウェア構成要素として実施される。

【００３４】

通常のレンダエンジンは、当技術分野において知られているとおり、レンダエンジンの実施詳細を隠すためにいくつかの論理層のコードを包含する。各層が、他の層、システムのハードウェア構成要素、またはシステムによって実行されるアプリケーションと通信するための機能およびインターフェースのセットを提供する。論理層の数は、様々であることが可能である。上位層ｎは、層ｎ－１に依拠し、以下同様である。層ｎ－１は、層ｎと比べてハードウェアにより近い。

【００３５】

本発明の例において、レンダエンジンは、少なくとも３つの論理層、すなわち、上位層、下位層、および抽象層を備える。上位層は、レンダエンジンに対するアクセスをアプリ

ケーション（例えば、ＣＡＤアプリケーション）に提供することを目的とし、例えば、上位層は、ＡＰＩなどのインターフェースを備える。上位層は、レンダエンジンに３Ｄシーンを提供するアプリケーション自体によってアクセスされ得る。下位層は、当技術分野において知られているとおり、ディスプレイにコンピュータグラフィックスをレンダリングするのを支援するように設計されたコンピュータプログラムライブラリであるグラフィックライブラリに対するアクセスをレンダエンジンに提供する。抽象層は、上位層と下位層の間にあり、ステップＳ１１０において作成された抽象グラフィックリソースを管理することを目的とする。最下位の層だけが、グラフィックリソースのリソース名に対するアクセスを有し、上位層だけが、抽象グラフィックリソースに対処する。

【００３６】

レンダエンジンは、通常、レンダエンジンコンピュータプログラムと呼ばれるコンピュータプログラムである。したがって、レンダエンジンは、方法のステップが実行される場合に実行される。

【００３７】

レンダエンジンに３Ｄシーンをロードすることは、３Ｄシーンを表すデータ（例えば、ファイル）が、レンダエンジンに提供されること、すなわち、レンダエンジンが、そのデータにアクセスし、それに対して計算を実行することができることを意味する。３Ｄシーンという表現は、少なくとも１つの３Ｄモデルが配置された３Ｄ空間を意味する。システムの見地から、シーンは、３Ｄシーンのビューをレンダリングするために使用されるべき少なくとも１つのグラフィックデータを備えるファイルである。

【００３８】

次に、ステップＳ１１０において、ステップＳ１００においてロードされた３Ｄシーンの少なくとも１つのグラフィックデータに関して抽象グラフィックリソースが作成される。作成は、前段で説明される抽象層によって実行されることが可能である。抽象グラフィックリソースは、グラフィックカードのうちの少なくとも１つに関してグラフィックリソースの識別子を記憶する。グラフィックリソースは、グラフィックカードによって表示可能なデータである、すなわち、それは、前にロードされたグラフィックデータからグラフィックライブラリによって計算される。グラフィックデータをグラフィックリソースに変換することは、当技術分野において知られているとおり、実行される。グラフィックデータは、通常、表示されるべき形状を記憶するバイナリアセットであり、それは、テクスチャ、バッファ、照光、シェーディング情報、視点などの、この形状を表すためのオプションのパラメータをさらに備えることが可能である。グラフィックライブラリの見地から、グラフィックリソースは、形状、バッファ、テクスチャ、フレームバッファ、サンプラ、シェーダ、視点などであることが可能であるが、以上には限定されない。シーンは、シーンのビューをレンダリングするために使用されるべき少なくとも１つのグラフィックデータを備え、通常のシーンは、一般に、数千のグラフィックデータを備えるものと理解される。

【００３９】

このため、ステップＳ１１０は、システムの各グラフィックカードに関して同一のグラフィックリソースの（ロードされたシーンのグラフィックデータの）識別を実行することを備え、識別されるグラフィックリソースのセットはそれ自体、グラフィックリソースの一意の識別子の役割をする抽象グラフィックリソースで識別される。

【００４０】

ステップＳ１１０において、作成される抽象グラフィックリソースは、システムの各グラフィックカード上のグラフィックリソースの一意の識別子を記憶する。実際には、抽象グラフィックリソースは、システムの各グラフィックカードに関する識別子をさらに記憶することが可能である。各グラフィックカードに関する識別子は、一意であることが可能である。このため、抽象グラフィックリソースは、グラフィックリソースの識別子を、グラフィックリソースがロードされるグラフィックカードの識別子と一緒に扱う。識別子は、数字、英数字などであることが可能であるが、以上には限定されない。例えば、各グラ

フィックカードが、整数として実施されるキーに結び付けられることが可能であり、その数値は、グラフィックカード１に関して１であり、グラフィックカード２に関して２であり、グラフィックカードｎに関してｎであり、以下同様である。

【００４１】

グラフィックリソースの識別子は、グラフィックライブラリに対するレンダエンジンのクエリがあると、獲得される。例において、レンダエンジンの下位層が、グラフィックリソースを作成したグラフィックライブラリにアクセスし、グラフィックライブラリが、システムのグラフィックカードのうちの１つの上のグラフィックリソースの識別子を提供する。下位層が識別子を獲得すると、それが抽象層に提供される。好ましくは、抽象グラフィックリソースの作成速度を向上させるために、所与のグラフィックリソースのすべての識別子に一度にクエリが行われる。

【００４２】

実際には、グラフィックカード上のグラフィックリソースの識別子は、リソース名とも呼ばれる。リソース名は、グラフィックデータがグラフィックリソースに変換される時点でグラフィックライブラリによって作成される。

【００４３】

次に、ステップＳ１２０において、抽象グラフィックリソースがテーブルに記憶され、例えば、そのテーブルが、システムのメモリ上に記憶される。テーブルは、既に存在することが可能であり、この場合、テーブルは、新たな抽象グラフィックリソースで完成させられる。テーブルがまったく存在しない場合、それが最初に作成され（すなわち、テーブルは、メモリにおいて利用可能である）、次に、新たな抽象グラフィックリソースで完成させられる。実際には、レンダエンジンは、実行される場合、メモリにロードされるソフトウェアであり、実行されるレンダエンジンには、抽象グラフィックリソースを記憶するためのテーブルがついてくる。このため、レンダエンジンには、３Ｄシーンをロードすることの結果、作成される抽象グラフィックリソースを記憶するテーブルに対するアクセスが提供される（Ｓ１４０）。このようにして、レンダエンジンは、３Ｄシーンのグラフィックデータを扱うこと（または管理すること）ができる。

【００４４】

次に、ステップＳ１３０において、グラフィックリソース（抽象グラフィックリソースを介してシステムの各グラフィックカード上で今や識別されている）が、システムの各グラフィックカード上にコピーされる。グラフィックカード上にコピーされる情報は、グラフィックカードによって活用されることが可能であり、すなわち、グラフィックリソースは、グラフィックカードの少なくとも１つのグラフィカルプロセッシングユニットによって操作が実行され得るグラフィックライブラリによって与えられたオブジェクトである。グラフィックカードにグラフィックリソースをコピーすることは、当技術分野において知られているとおり、実行され、例えば、グラフィックリソースが、グラフィックカードのメモリ上に記憶されることを理解されたい。かつステップＳ１４０において、レンダエンジンは、３Ｄシーンをロードすることの結果、作成された抽象グラフィックリソースを記憶するテーブルにアクセスすることができる。

【００４５】

図３から図５を次に参照して、レンダエンジンがアプリケーションの命令を受け取った場合のステップＳ１００からＳ１３０の例が説明される。レンダエンジン命令は、レンダエンジンが実行しなければならないコマンドである。命令は、例えば、特定の色で形状を描くこと、または画面をクリアすることであり得る。命令は、レンダエンジンがどのように構造化されるかに依存して、レンダエンジンを使用するユーザから（例えば、表示要求を実行するアプリケーションを介して）、またはレンダエンジンから直接に来ることが可能である。一般的なレンダエンジンにおいて、命令は、通常、アクション（描く、クリアするなど）と、グラフィックリソース（例えば、描くべき形状グラフィックリソース）を識別する１または複数のリソース名とから成る。本発明において、レンダエンジンの抽象層が、実行されるべき命令において、グラフィックリソースのリソース名を、グラフィッ

クリソースに関連付けられた抽象グラフィックリソースで置き換える。命令は、システムのグラフィックカードの数により実行される。例えば、システムが、８のグラフィックカードを備える場合、命令は、８回、１つのグラフィックカードに関して１つの実行で、実行される。

【００４６】

図３において、抽象グラフィックリソースとリソース名とグラフィックカードとの間の関係が表される。グラフィックデータが、アプリケーションの要求があって、レンダエンジン上にロードされている（３Ｄシーンをロードする場合）。抽象グラフィックリソースが、グラフィックデータをロードすることの結果、作成されている。抽象グラフィックリソースは、コンピュータシステムの２つのグラフィックカード、すなわち、グラフィックカード１およびグラフィックカード２の上のグラフィックリソース（ロードされたグラフィックデータからグラフィックライブラリによって計算された）を表す。各グラフィックリソースは、リソース名とも呼ばれる一意の識別子を有し、ここで、グラフィックリソースは、グラフィックカード１上のリソース名１と、グラフィックカード２上のリソース名２とを有する。グラフィックカード上にグラフィックリソースをコピーすることは、抽象グラフィックリソースの作成より前に実行されることも可能であり、すなわち、ステップＳ１３０が、ステップＳ１１０より前に実行されることを理解されたい。ステップＳ１３０より前にステップＳ１１０を実行することは、グラフィックカード上のグラフィックリソースの関連付けを向上させ、抽象グラフィックリソースの作成および管理も向上させる。

【００４７】

図５は、グラフィックデータがレンダエンジンにロードされた場合、実行されるステップを示す。作成された抽象グラフィックリソースは、グラフィックリソースのｎのリソース名を表し、各リソース名がｎのグラフィックカードのうちの１つに関連付けられる。グラフィックリソースは、グラフィックカード上にコピーされる。コピーすることが終了されると（ステップＳ１３０）、前に作成された（ステップＳ１２０）抽象グラフィックリソースがアプリケーションに戻される。リソース名の代わりに抽象グラフィックリソースが使用されることは、抽象グラフィックリソースが複数のグラフィックリソースを表すことを認識していないアプリケーションに透過的である。このため、アプリケーションの変更はまったく要求されず、抽象グラフィックリソースの作成／管理は、レンダエンジンの抽象層によって実行される。抽象層は、作成された抽象グラフィックリソースを上位層に提供するように構成されることが可能である。このようにして、アプリケーションは、アプリケーションがアクセスすることができる上位層経由で新たな抽象グラフィックリソースの作成について直接に通知され得る。このことは、システムに１つだけのグラフィックカードしか存在しないかのように、グラフィックカード管理および作業の複雑さを隠すことを可能にする。

【００４８】

次に図４を参照すると、作成された抽象グラフィックリソースを記憶するため、および管理するために使用されるアーキテクチャの例が次に説明される。この例のアーキテクチャは、テーブルの良好な行を探索するのに使用される抽象グラフィックリソースを有するダブルエントリタビュラ（double entry tabular）と同様であり、グラフィックカード識別子（グラフィックカードに関するキーとも呼ばれる）がテーブルの良好な列を探索するのに使用される。例えば、抽象ハンドル４０が、いくつかの列４２ａ～４２ｃを備えるテーブルの行４２を識別することを可能にし、各列は、１つのグラフィックカード識別子を記憶する。次に、列から、グラフィックリソースのすべてのリソース名（すなわち、各グラフィックカード上のリソース名）を獲得することが可能である。実際には、リソース名は、グラフィックカードの識別子と一緒にテーブル上に、例えば、行４２ａ～４２ｃ上に記憶され得る。

【００４９】

次に、２つの異なる視点で３Ｄシーンを表示するための抽象グラフィックリソース作成

および使用の例が説明される。この例において、システムは、２つのグラフィックカードを備え、各グラフィックカードは、１つの視点で３Ｄシーンを表示することを担う。アプリケーションが、２つの異なる視点で３Ｄシーンを表示するようレンダエンジンに要求した後、グラフィックライブラリ命令（グラフィックライブラリアクションとも呼ばれる）がレンダエンジンによって受け取られる。前段で見られるとおり、シーン上の視点は、グラフィックライブラリによって与えられ、かつ操作が実行され得るオブジェクトであるグラフィックリソースである。かつこの視点グラフィックリソースは、レンダエンジンに命令を送ることによってアプリケーションによって要求され得る。説明される例において、アプリケーションは、同一のシーン上の２つの視点を要求する命令を送る。レンダエンジンの上位層は、通常、アプリケーションの命令を受け取ることを担い、言い換えると、アプリケーションは、上位層経由でレンダエンジンに対するアクセスを有する。命令が受け取られると、レンダエンジンは、この命令によって関与させられる抽象グラフィックリソースを識別する、例えば、テーブルにおいて抽象グラフィックリソースが探索される。実際には、このことは、レンダエンジンがアプリケーションに、ロードされたグラフィックデータに関連付けられた抽象グラフィックリソースを提供しているために可能にされる。抽象グラフィックリソースのテーブル行が、すべてのグラフィックカード上のリソース名を獲得するために走査され、グラフィックリソースは、知られており、命令は、命令において抽象グラフィックリソースを対応するリソース名によって置き換えることによって実行され得る。命令は、各グラフィックカードに関して、各グラフィックカード上に記憶されたグラフィックリソースのリソース名で実行されることを理解されたい。

【００５０】

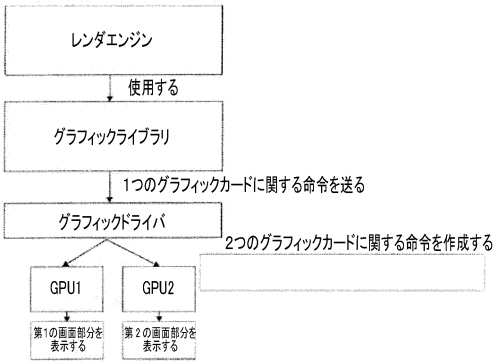
抽象グラフィックリソースの削除は、類似した様態で実行される。アプリケーションが、レンダエンジンにグラフィックデータを削除する命令を送る。その命令（またはアクション）が、アプリケーションがレンダエンジンに、例えば、レンダエンジンの上位層を介してアクセスすることができるので、レンダエンジンによって受け取られる。命令は、アプリケーションがグラフィックデータ（グラフィックカード上のグラフィックリソースとしてコピーされた）を抽象グラフィックリソースで識別するので、抽象グラフィックリソースにアドレス指定される。次に、レンダエンジンが、抽象グラフィックリソーステーブルにクエリを行い、システムのグラフィックカード上のグラフィックリソースの識別子を識別する。次に、命令は、抽象グラフィックリソースを、テーブルにおいて見出されるグラフィックリソースによって置き換えることによって変更されることが可能であり、新たな命令が、抽象グラフィックリソースに関連付けられた各グラフィックリソースに関して作成される。その結果、削除命令が、実行されることが可能であり、グラフィックリソースが、グラフィックカードメモリから消去されることが可能である。

【００５１】

前段で提示される本発明の例は、グラフィックデータの表示にすべてのグラフィックカードがかかわるシナリオだけを企図する。興味深いことに、本発明は、アプリケーションが、表示プロセスにかかわるシステムのグラフィックカードのなかから１または複数のグラフィックカードを選択することをさらに可能にする。グラフィックカードの数のアプリケーションによる選択は、１または複数のパラメータ、例えば、グラフィックデータを表示するために要求される計算リソースに（例えば、ＧＰＵに）依存することが可能である。選択は、ユーザアクションがあると、構成されることも可能であり、例えば、ユーザが、グラフィックデータをレンダリングするために使用されるグラフィックカードの数が所定の数を超えないようにアプリケーションを構成する。システムのグラフィックカードのなかから１または複数のグラフィックカードが使用されるシナリオに関して、レンダエンジンに、アプリケーションによって、レンダエンジンを構成する特定の命令を使用することによって通知が行われる。レンダエンジンは、すべてのグラフィックデータに関して、または１または複数の特定のグラフィックデータに関して限定された数のグラフィックカードを使用するように構成されることが可能である。少なくとも１つのグラフィックデータを扱うための少なくとも１つのグラフィックカードのこの選択は、レンダリングにかか

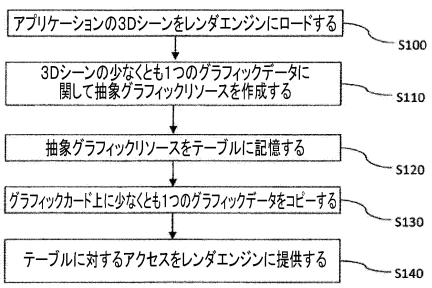
わるグラフィックカードだけが、表示すべきグラフィックリソースに関連付けられた抽象グラフィックリソースの行において識別されるように、抽象リソースが作成される前に実行される。このため、本発明は、マルチグラフィックカードシステムの管理を大幅に向上させることを可能にし、このため、シーンのレンダリングを向上させることを可能にし、特に、３Ｄシーン上の視点を同時にレンダリングすることが、アプリケーションもグラフィックライブラリも変更することなしに可能である。

【図１】

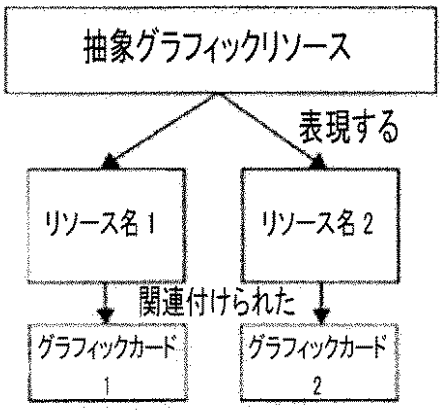


（従来技術）

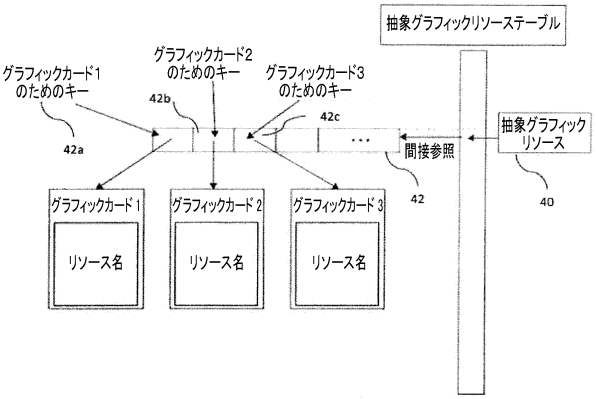
【図２】



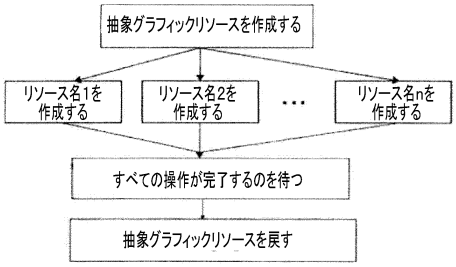
【図３】



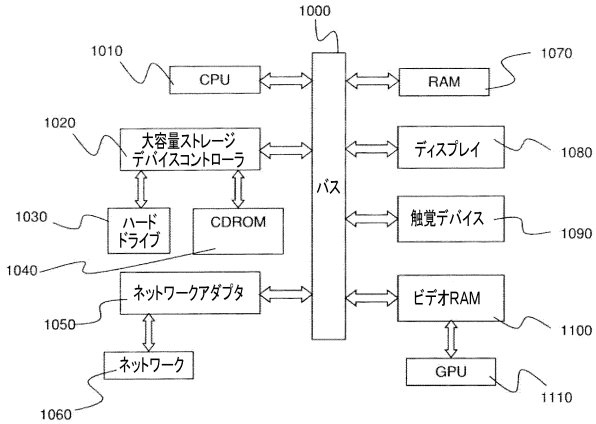
【図４】



【図５】


抽象グラフィックリソースを作成する
リソース名1を作成する
リソース名2を作成する
...
リソース名nを作成する
すべての操作が完了するのを待つ
抽象グラフィックリソースを戻す

【図６】


1000
1010
CPU
1070
RAM
1020
大容量ストレージデバイスコントローラ
ディスプレイ
1080
バス
1030
ハードドライブ
CDROM
触覚デバイス
1090
1040
ネットワークアダプタ
ビデオRAM
1050
1100
ネットワーク
GPU
1060
1110

フロントページの続き

(72)発明者　ニコラ　ジャン
フランス　７８１４０　ベリジー－ビラクブレー　アベニュー　ルイ　ブレゲー　３７　シー
(72)発明者　ニコラ　コロンブ
フランス　９１３００　マシー　プレイス　ビクトール　シェルシェル　１１

審査官　村松　貴士

(56)参考文献　米国特許出願公開第２０１２／０００１９２５（ＵＳ，Ａ１）
特表２０１５－５１１３４６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｔ　１５／００　－　１５／８７