(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2010-20755
(P2010-20755A)
(43)公開日　平成22年1月28日(2010.1.28)

(51)Int.Cl.		F I			テーマコード（参考）
G06T 15/00	(2006.01)	G06T	15/00	100A	5B050
G06F 3/153	(2006.01)	G06F	3/153	310A	5B057
G06F 9/44	(2006.01)	G06F	9/44	530A	5B069
G06T 1/00	(2006.01)	G06F	9/06	620K	5B080
G06T 1/20	(2006.01)	G06T	1/00	B	5B376

審査請求 未請求　請求項の数 52　ＯＬ　外国語出願　（全 29 頁）　最終頁に続く

(21)出願番号　特願2009-93554 (P2009-93554)
(22)出願日　平成21年4月8日 (2009.4.8)
(31)優先権主張番号　61/123,463
(32)優先日　平成20年4月8日 (2008.4.8)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　61/123,549
(32)優先日　平成20年4月8日 (2008.4.8)
(33)優先権主張国　米国 (US)

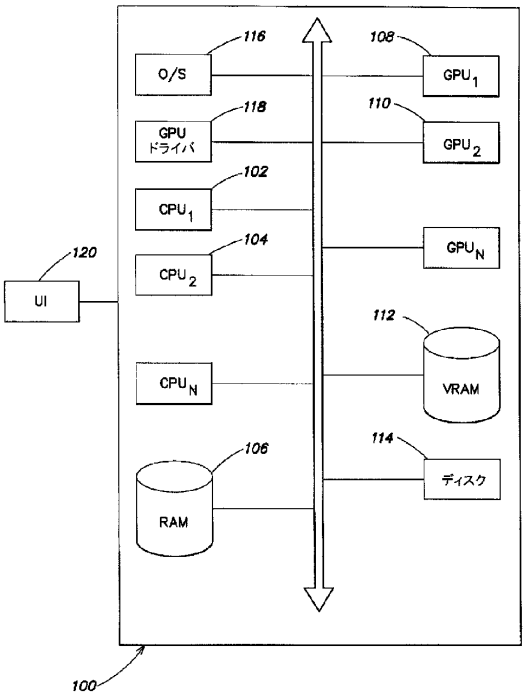
（特許庁注：以下のものは登録商標）
１．Ｌｉｎｕｘ

(71)出願人　500035823
アビッド　テクノロジー　インコーポレイテッド
アメリカ合衆国　０１８７６　マサチューセッツ州　チュークスベリー　ワン　パーク　ウェスト　アビッド　テクノロジー　パーク　（番地なし）
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100089705
弁理士　社本　一夫
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100080137
弁理士　千葉　昭男

最終頁に続く

(54)【発明の名称】複数のハードウェア・ドメイン、データ・タイプ、およびフォーマットの処理を統合し抽象化するフレームワーク

(57)【要約】　（修正有）
【課題】メディア・オブジェクトを処理するためのポータブルな開発および実行フレームワークの提供。
【解決手段】フレームワークは、メディア処理機能を実行する命令を受け入れ、メディア処理機能と関連付けるメディア・オブジェクトを受け入れ、メディア・オブジェクトのタイプおよびフォーマットを指定する属性と、メディア・オブジェクトと関連付けられたハードウェア・ドメインでメディア・オブジェクトをラップし、メディア・オブジェクト上においてメディア処理機能を実行ドメインに遂行させることを含む。メディア処理機能を実行する命令は、メディア・オブジェクトと関連付けられたハードウェア・ドメインには依存しない形態で表現され、メディア・オブジェクトのタイプおよびフォーマットにも依存しないこともできる。メディア・オブジェクトは、画像とすることができ、メディア処理機能は、ＧＰＵ上で実行する画像処理機能を含むことができる。
【選択図】図１



【特許請求の範囲】
【請求項１】
メディア処理システムであって、
複数の実行ドメインと、
前記複数の実行ドメインのうち第１実行ドメインと関連付けられたメモリとを備えており、該メモリは前記複数の実行ドメインのうち第１実行ドメインによって読み取り可能な命令を備えており、前記複数の実行ドメインのうち前記第１実行ドメイン上において前記命令が実行されると、前記複数の実行ドメインのうち前記第１実行ドメインに、
メディア処理機能を実行するための命令を受け入れさせ、
前記メディア処理機能と関連付けられるメディア・オブジェクトを受け入れさせ、前記メディア・オブジェクトは、当該メディア・オブジェクトのタイプ、当該メディア・オブジェクトのフォーマット、および当該メディア・オブジェクトと関連付けられるハードウェア・ドメインを指定する属性によってラップし、
前記複数の実行ドメインの少なくとも１つに、前記メディア処理機能を前記メディア・オブジェクト上で実行させ、前記メディア処理機能を実行する前記命令は、前記メディア・オブジェクトと関連付けられるハードウェア・ドメインには依存しない形態で表現される、
メディア処理システム。
【請求項２】
請求項１記載のメディア処理システムにおいて、前記メディア処理機能を実行する前記命令は、前記メディア・オブジェクト・タイプには依存しない形態で表現される、メディア処理システム。
【請求項３】
請求項１記載のメディア処理システムにおいて、前記メディア処理機能を実行する前記命令は、前記メディア・オブジェクト・フォーマットには依存しない形態で表現される、、メディア処理システム。
【請求項４】
請求項１記載のメディア処理システムにおいて、前記複数の実行ドメインは、中央処理ユニット（ＣＰＵ）と、グラフィックス処理ユニット（ＧＰＵ）とを含み、前記複数の実行ドメインのうち前記第１実行ドメインはＣＰＵである、メディア処理システム。
【請求項５】
請求項１記載のメディア処理システムにおいて、前記メディア処理機能は、画像効果であり、前記メディア・オブジェクトの前記タイプはラスタ画像である、メディア処理システム。
【請求項６】
請求項６記載のメディア処理システムにおいて、前記画像効果は、ディゾルブ、色補正、テキスト挿入、および動き効果のうち１つを含む、メディア処理システム。
【請求項７】
請求項１記載のメディア処理システムにおいて、前記メディア処理機能は画像効果であり、前記メディア・オブジェクト・タイプはグラフィックス・オブジェクトである、メディア処理システム。
【請求項８】
請求項１記載のメディア処理システムにおいて、前記複数の実行ドメインの各々は、下位命令ライブラリと関連付けられており、前記実行ドメインの１つと関連付けられた下位ライブラリの少なくとも部分集合が、前記複数の実行ドメインの別の１つと関連付けられた下位ライブラリの対応する部分集合と互換性がない、メディア処理システム。
【請求項９】
請求項１記載のメディア処理システムであって、更に、前記複数の実行ドメインのうち前記第１実行ドメイン上で実行されると、前記第１実行ドメインに、
前記メディア処理機能と前記メディア・オブジェクトと関連付けられた前記メディア・

オブジェクト・タイプ、メディア・オブジェクト・フォーマット、および実行ドメインのうちの少なくとも１つとの間における不一致を識別させ、
前記メディア・オブジェクトの前記タイプを別のタイプに変換すること、または前記メディア・オブジェクトの前記フォーマットを別のフォーマットに変換すること、または別のハードウェア・ドメインを前記メディア・オブジェクトと関連付けることのいずれかによって、前記識別した不一致を解消させる、
命令を含む、メディア処理システム。
【請求項１０】
請求項１記載のメディア処理システムにおいて、前記属性は、容認可能な属性集合の中の１つであり、前記容認可能な属性集合は、新たなメディア・オブジェクト・タイプ、新たなメディア・オブジェクト・フォーマット、および新たな関連するハードウェア・ドメインのうち少なくとも１つを有する新たな属性を含むように増強することができ、前記メディア処理機能を実行する前記命令は、当該命令を書き直し又はコンパイルし直す必要なく、前記新しい属性にラップされたメディア・オブジェクト上で前記メディア処理機能を実行させることができる、メディア処理システム。
【請求項１１】
請求項１記載のメディア処理システムであって、更に、前記複数の実行ドメインのうち前記第１実行ドメイン上で実行されると、当該第１実行ドメインに、前記メディア・オブジェクトを複数の部分に分割させ、前記第１実行ドメインを第２実行ドメインに接続するデータ・バスを通じて前記部分を順次送出させ、一度に前記部分の１つにおいて前記メディア処理機能を実行させる命令を含む、メディア処理システム。
【請求項１２】
請求項１記載のメディア処理システムにおいて、前記命令は複数の処理ユニットを備えており、前記メディア処理機能は、前記複数の処理ユニットのうち少なくとも１つの第１処理ユニットを実行することによって遂行され、前記複数の処理ユニットのうち前記少なくとも１つの第１処理ユニットは、前記複数の実行ドメインのうち前記第１実行ドメイン上で実行されると、前記複数の処理ユニットのうち第２処理ユニットをコールする、メディア処理システム。
【請求項１３】
請求項１記載のメディア処理システムにおいて、前記命令は複数の処理ユニットを備えており、前記メディア処理機能の遂行は、前記複数の処理ユニットの１つをコールし、コールされた処理ユニットからスレッドを生成することを含み、前記コールされた処理ユニットが前記メディア・オブジェクトに対して前記メディア処理機能を実行し続けている間、前記スレッドが非同期で実行される、メディア処理システム。
【請求項１４】
メディア・オブジェクト処理方法であって、
メディア処理機能を実行するための命令を受け入れるステップと、
前記メディア処理機能と関連付けられるメディア・オブジェクトを受け入れるステップであって、前記メディア・オブジェクトは、当該メディア・オブジェクトのタイプ、当該メディア・オブジェクトのフォーマット、および当該メディア・オブジェクトと関連付けられるハードウェア・ドメインを指定する属性によってラップされる、ステップと、
前記複数の実行ドメインの少なくとも１つに、前記メディア処理機能を前記メディア・オブジェクト上で実行させるステップであって、前記メディア処理機能を実行する前記命令は、前記メディア・オブジェクトと関連付けられるハードウェア・ドメインには依存しない形態で表現される、ステップと、
を備えた、メディア処理方法。
【請求項１５】
請求項１４記載の方法において、前記メディア処理機能を実行する前記命令は、前記メディア・オブジェクト・タイプには依存しない形態で表現される、方法。
【請求項１６】

請求項１４記載の方法において、前記メディア処理機能を実行する前記命令は、前記メディア・オブジェクト・フォーマットには依存しない形態で表現される、方法。
【請求項１７】
請求項１４記載の方法において、前記複数の実行ドメインは、中央処理ユニット（ＣＰＵ）と、グラフィックス処理ユニット（ＧＰＵ）とを含み、前記複数の実行ドメインのうち前記第１実行ドメインはＣＰＵである、方法。
【請求項１８】
請求項１４記載の方法において、前記メディア処理機能は、画像効果であり、前記メディア・オブジェクトの前記タイプはラスタ画像である、方法。
【請求項１９】
請求項１８記載の方法において、前記画像効果は、ディゾルブ、色補正、テキスト挿入、および動き効果のうちの１つを含む、方法。
【請求項２０】
請求項１４記載の方法において、前記メディア処理機能は画像効果であり、前記メディア・オブジェクト・タイプはグラフィックス・オブジェクトである、方法。
【請求項２１】
請求項１４記載の方法において、前記複数の実行ドメインの各々は、下位命令ライブラリと関連付けられており、前記実行ドメインの１つと関連付けられた下位ライブラリの少なくとも部分集合が、前記複数の実行ドメインの別の１つと関連付けられた下位ライブラリの対応する部分集合と互換性がない、方法。
【請求項２２】
請求項１４記載の方法であって、更に、
前記メディア処理機能と、前記メディア・オブジェクトと関連付けられた前記メディア・オブジェクト・タイプ、メディア・オブジェクト・フォーマット、および実行ドメインのうちの少なくとも１つとの間における不一致を識別するステップと、
前記メディア・オブジェクトの前記タイプを別のタイプに変換すること、または前記メディア・オブジェクトの前記フォーマットを別のフォーマットに変換すること、または別のハードウェア・ドメインを前記メディア・オブジェクトと関連付けることのいずれかによって、前記識別した不一致を解消するステップと、
を含む、方法。
【請求項２３】
請求項１４記載の方法において、前記属性は、容認可能な属性集合の中の１つであり、前記容認可能な属性集合は、前記命令を書き直し又はコンパイルし直す必要なく、新たなメディア・オブジェクト・タイプ、新たなメディア・オブジェクト・フォーマット、および新たな関連するハードウェア・ドメインのうち少なくとも１つを含むように増強することができる、方法。
【請求項２４】
請求項１４記載の方法であって、更に、
前記メディア・オブジェクトを複数の部分に分割するステップと、
前記第１実行ドメインを第２実行ドメインに接続するデータ・バスを通じて前記部分を順次送出するステップと、
一度に前記部分の１つにおいて前記メディア処理機能を実行するステップと、
を備えた、方法。
【請求項２５】
請求項１４記載の方法において、前記命令は複数の処理ユニットのうち第１処理ユニットに対するコールを含み、前記複数の処理ユニットの各々は、対応するメディア処理機能を実行することと関連付けられた命令の集合であり、前記複数の処理ユニットのうち前記第１処理ユニットは、前記複数の処理ユニットのうち第２処理ユニットをコールする、方法。
【請求項２６】

画像処理システムであって、
中央処理ユニット（ＣＰＵ）と、
グラフィックス処理ユニット（ＧＰＵ）と、
前記ＣＰＵと関連付けられたメモリと、
を備えており、該メモリは前記ＣＰＵによって読み取り可能な命令を備えており、該命令は、前記ＣＰＵによって実行されると、該ＣＰＵに、
画像処理機能を実行する命令を受け入れさせ、
前記画像処理機能と関連付けられる画像を受け入れさせ、前記画像は、該画像のフォーマットおよび該画像と関連付けられた前記ハードウェア・ドメインを指定する属性でラップされ、
前記ＧＰＵに、前記画像上で前記画像処理機能を実行させ、前記画像処理機能を実行する命令は、前記画像と関連付けられたハードウェア・ドメインには依存しない形態で表現される、画像処理システム。

【請求項２７】
請求項２６記載の画像処理システムにおいて、前記ＧＰＵは、関連するシェーダ言語を有し、前記画像処理機能を実行する命令は、前記シェーダ言語には依存しない形態で表現される、画像処理システム。

【請求項２８】
請求項２６記載の画像処理システムにおいて、前記命令の実行は、前記ＣＰＵ上で走るオペレーティング・システムによって制御され、前記画像処理機能を実行する命令は、前記オペレーティング・システムには依存しない形態で表現される、画像処理システム。

【請求項２９】
請求項２６記載の画像処理システムにおいて、前記ＧＰＵは、画像レンダリング・データ・バッファを含み、該画像レンダリング・データ・バッファは、テクスチャ、フレーム・バッファ・オブジェクト、マルチ・サンプル・レンダ・バッファ、リード専用画素バッファ・オブジェクト、ライト専用画素バッファ・オブジェクト、およびリード－ライト画素バッファ・オブジェクトのうち少なくとも１つを備えており、前記画像は、前記画像レンダリング・バッファには依存しない形態で表現される、画像処理システム。

【請求項３０】
請求項２６記載の画像処理システムにおいて、前記ＧＰＵは、画像レンダリング・テクスチャ・パラメータを含み、該画像レンダリング・テクスチャ・パラメータは、色空間、画素深さおよび画素範囲のうち少なくとも１つを備えており、前記画像は、前記画像レンダリング・テクスチャ・パラメータには依存しない形態で表現される、画像処理システム。

【請求項３１】
請求項２６記載の画像処理システムにおいて、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、多重パス実行と、ＣＰＵ上に適時にコンパイルしたマルチパス画素プログラムをキャッシュすること、前記画素プログラムを部分的にコンパイルすること、ならびに前記部分的にコンパイルされた画素プログラムをキャッシュし取り出すことを含む、画像処理システム。

【請求項３２】
請求項２６記載の画像処理システムにおいて、前記ＣＰＵは、前記メモリの一部を、画像データを格納するために割り当て、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記メモリの新たな部分を前記画像を格納するために割り当てずに、前記メモリの割り当てた部分をリサイクルすることを含む、画像処理システム。

【請求項３３】
請求項２６記載の画像処理システムにおいて、前記画像は、８ビットＲＧＢ色空間画像として表され、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記画像をＢＧＲＡテクスチャにパックすることを含む、画像処理システム。

【請求項３４】

請求項２６記載の画像処理システムにおいて、前記画像は、８ビットＹＣＣ色空間画像として表され、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記画像をＢＧＲＡテクスチャにパックすることを含む、画像処理システム。

【請求項３５】

請求項２６記載の画像処理システムにおいて、前記画像は、別個のアルファ・チャネルを有する８ビットＹＣＣ色空間画像（ＹＣＣＡ）として表され、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記画像をＢＧＲＡテクスチャにパックすることを含む、画像処理システム。

【請求項３６】

請求項２６記載の画像処理システムにおいて、前記画像処理機能を実行する前記命令は、前記画像を表すために用いられる色空間には依存しない形態で表現される、画像処理システム。

【請求項３７】

請求項２６記載の画像処理システムにおいて、前記画像処理機能を実行する前記命令は、前記画像を表現するために用いられる画素深さには依存しない形態で表現される、画像処理システム。

【請求項３８】

請求項２６記載の画像処理システムにおいて、前記画像処理機能を実行する前記命令は、前記画像を表現するために用いられる画素範囲には依存しない形態で表現される、画像処理システム。

【請求項３９】

請求項２６記載の画像処理システムにおいて、前記画像処理機能を実行する命令は、前記画像を格納するために用いられるメモリ・レイアウトおよびパッキングには依存しない形態で表現される、画像処理システム。

【請求項４０】

画像処理方法であって、

ＣＰＵ上で走るクライアント・アプリケーションから、画像処理機能を実行する命令を受け入れるステップと、

前記クライアント・アプリケーションから、前記画像処理機能と関連付けられる画像の指示を受け入れるステップと、

前記画像を、該画像のフォーマットおよび該画像と関連付けられるハードウェア・ドメインを指定する属性でラップするステップと、

ＧＰＵに、前記画像上で前記画像処理機能を実行させるステップであって、前記画像処理機能を実行する命令は、前記画像と関連付けられるハードウェア・ドメインには依存しない形態で表現される、ステップと、

を備えた、画像処理方法。

【請求項４１】

請求項４０記載の画像処理方法において、前記ＧＰＵは、関連するシェーダ言語を有し、前記画像処理機能を実行する前記命令は、前記シェーダ言語には依存しない形態で表現される、画像処理方法。

【請求項４２】

請求項４０記載の画像処理方法において、前記命令の実行は、前記ＣＰＵ上で走るオペレーティング・システムによって制御され、前記画像処理機能を実行する前記命令は、前記オペレーティング・システムには依存しない形態で表現される、画像処理方法。

【請求項４３】

請求項４０記載の画像処理方法において、前記ＧＰＵは、画像レンダリング・データ・バッファを含み、該画像レンダリング・データ・バッファのタイプは、テクスチャ、フレーム・バッファ・オブジェクト、マルチ・サンプル・レンダ・バッファ、リード専用画素バッファ・オブジェクト、ライト専用画素バッファ・オブジェクト、およびリードーライト画素バッファ・オブジェクトのうちの１つであり、前記画像は、前記画像レンダリング

・バッファの前記タイプには依存しない形態で表現される、画像処理方法。
【請求項４４】
請求項４０記載の画像処理方法において、前記ＧＰＵは、画像レンダリング・テクスチャ・パラメータを含み、該画像レンダリング・テクスチャ・パラメータは、色空間、画素深さおよび画素範囲のうち少なくとも１つを備えており、前記画像は、前記画像レンダリング・テクスチャ・パラメータには依存しない形態で表現される、画像処理方法。
【請求項４５】
請求項４０記載の画像処理方法において、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、多重パス実行と、ＣＰＵ上に適時にコンパイルしたマルチパス画素プログラムをキャッシュすること、前記画素プログラムを部分的にコンパイルすること、ならびに前記部分的にコンパイルされた画素プログラムをキャッシュし取り出すことを含む、画像処理方法。
【請求項４６】
請求項４０記載の画像処理方法において、前記ＣＰＵは、メモリと関連付けられており、前記ＣＰＵは、前記メモリの一部を、画像データを格納するために割り当て、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記メモリの新たな部分を前記画像を格納するために割り当てずに、前記メモリの割り当てた部分をリサイクルすることを含む、画像処理方法。
【請求項４７】
請求項４０記載の画像処理方法において、前記画像は、８ビットＲＧＢ色空間画像、８ビットＹＣＣ色空間画像、および別個のアルファ・チャネルを有する８ビットＹＣＣ色空間画像のうちの１つとして表され、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記画像をＢＧＲＡテクスチャにパックすることを含む、画像処理方法。
【請求項４８】
請求項４０記載の画像処理方法において、前記画像処理機能を実行する前記命令は、前記画像を表すために用いられる色空間には依存しない形態で表現される、画像処理方法。
【請求項４９】
請求項４０記載の画像処理方法において、前記画像処理機能を実行する前記命令は、前記画像を表現するために用いられる画素深さには依存しない形態で表現される、画像処理方法。
【請求項５０】
請求項４０記載の画像処理方法において、前記画像処理機能を実行する前記命令は、前記画像を表現するために用いられる画素範囲には依存しない形態で表現される、画像処理方法。
【請求項５１】
請求項４０記載の画像処理方法において、前記画像処理機能を実行する前記命令は、前記画像を格納するために用いられるメモリ・レイアウトおよびパッキングには依存しない形態で表現される、画像処理方法。
【請求項５２】
請求項４０記載の画像処理方法において、前記ＧＰＵに前記画像上において前記画像処理機能を実行させることは、前記ＧＰＵ上における処理スレッドの非同期の実行を含む、画像処理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
（関連出願に対する相互引用）
本願は、２００８年４月８日に出願した仮出願第６１／１２３，４６３号および６１／１２３，５４９号の、３５Ｕ．Ｓ．Ｃ．§１９９（ｅ）に基づく優先権およびその恩恵を主張する。

【従来技術】
【０００２】
ポスト制作ソフトウェア・アプリケーションおよび画像処理ソフトウェア・アプリケーションは、そのビデオ効果の処理を加速するために、カスタム・ハードウェアおよび包括的ハードウェアを増々利用することができるようになっている。新たに導入された技術によって提供される処理の高速化を利用することは、シネマおよびテレビジョン・コンテンツが向上し続けるに連れて、一層重要性を増しつつある。ブルー・レイ・ディスク・プレーヤを備えた高品位家庭用シアター・システムが日用品となっている一方で、２Ｋまたは４Ｋラインもの解像度を有するディジタル・シネマ投影が普及しつつある。これは、モーション・ピクチャおよび放送制作パイプライン全体における必要なデータ処理増大という代償によってなされている。画像およびビデオ処理システムは、このようなメディアを扱うために、その性能を拡大(scale up)していく必要がある。
【０００３】
メディア処理システムに利用可能な包括的ハードウェアの中には、Ｉｎｔｅｌ系プラットフォーム用のＳＳＥ２、およびＡｐｐｌｅ　Ｍａｃｉｎｔｏｓｈ・プラットフォーム用のＡｌｔｉｖｅｃ、市販のワークステーションに慣例的に実装されている複数のグラフィックス処理ユニット（ＧＰＵ）、およびＩｎｔｅｌ　Ｃｏｒｐｏｒａｔｉｏｎ（インテル社）からのＬａｒｒａｂｅｅのような、その他の特殊ハードウェアのように、種々のホストＣＰＵ技術がある。コンピュータ・ゲーム市場は、ＧＰＵを日用品にするのを促進した。これらのＧＰＵは、同じ価格帯の中央処理ユニット（ＣＰＵ）よりも算術処理能力が遥かに高い。ＧＰＵは、固有の画像レンダリング並列性を利用して、特にゲーム・アプリケーションにおいて制御指向ＣＰＵを凌駕する。
【０００４】
ゲームおよび画像処理アプリケーション用の画像レンダリングは、同様のプロセスを伴う。ＧＰＵ上の汎用計算（ＧＰＧＰＵ）と呼ばれる、発展途上の研究分野では、画像処理を含む種々の問題にＧＰＵを用いる技法を探求する。しかしながら、既存のＧＰＵ加速画像処理システムは、ＣまたはＣ＋＋のような汎用プログラミング言語を用いて、開発者が容易にプログラミングすることができない。現在のＧＰＵ加速画像処理システムは、ＧＰＵコンポーネント、レンダリング・パイプライン、およびレンダリングＡＰＩの込み入った知識を必要とする。
【０００５】
また、市販のプラットフォーム、オペレーティング・システム、グラフィックス・カード、およびシェーダ言語(shader language)に対するビデオ処理の必要性に向けて直接的に目標を定めたＧＰＵプログラミング・サービスが一般に欠如している。ビデオ処理要件は、主に、ホストとＧＰＵメモリとの間における高帯域幅転送要件、およびビデオ系視覚効果の処理に必要な特定の処理フォーマットおよびタイプを扱うその他のサービスによって特徴付けられる。
【０００６】
新たな技術が頻繁に導入されることによって、ソフトウェア・コーディングの複雑さが増大する。新たな技術が提供することができる性能加速を利用するためには、ソフトウェアが、用いられる特定のハードウェア・タイプおよびモデル、オペレーティング・システム、ならびにプラットフォームに密接に繋がれている必要がある。これが意味するのは、同じアプリケーションを異なるハードウェアおよびシステム構成上で最適に走らせるためには、複数のバージョンのコードが必要となるということである。
【０００７】
新たに導入されたハードウェアを用いる際、従前からの手法では、ソフトウェアを走らせようとする各新技術に特定的な個々の低レベル・ライブラリを開発していた。これらのライブラリは、特定のハードウェアおよびオペレーティング・システムに専用であり、高度に最適化されている。これらは、それら自体のプロトコルおよび特殊性(particularities)を有し、目標のハードウェアによって大きく影響されることが多いプログラミング・

モデルを採用する。このため、ハードウェアの発展に合わせてアプリケーション・ソフトウェアを維持することが困難になり、新たなビデオ効果のような、新たなソフトウェア・アプリケーションの迅速な開発が阻害される。また、クライアント・アプリケーションにおいて大幅な変更を必要とせずに新たなハードウェア実行ドメインを採用することが妨げられる。

【０００８】

一旦ハードウェア特定低レベル・ライブラリを開発し、デバッグし、そして最適化すると、アプリケーション開発者およびユーザは、これらの円熟したライブラリの強さ(strength)を組み合わせようとする。これは、通常、種々のライブラリのデータ構造およびコードの一部を統一することによって遂行する。しかしながら、このために、既にデバッグしてあるソフトウェアが不安定になる可能性がある。

【発明の概要】

【発明が解決しようとする課題】

【０００９】

概して言えば、本発明は、ライブラリ自体に対する修正や、新たなハードウェア上で走る低レベル・ライブラリに基づく機構(feature)を提供するクライアント・アプリケーションに対する修正を必要とせずに、標準的な共通インターフェースを通じてアクセスする低レベル・ライブラリ能力およびハードウェア実行ドメインのホスティング、統合、および拡張を容易に行うことを可能にする、効率的にポータブルな実行ドメイン不可知フレームワーク(agnostic framework)を特徴とする。

【課題を解決するための手段】

【００１０】

一形態では、フレームワークは、ポータブルな、プラットフォームおよびオペレーティング・システム不可知な、コンポーネントを基本とするアーキテクチャ（例えば、コンポーネント処理ライブラリ）を含み、プラグイン・フォーマットでありプラグイン・ハードウェア・ドメインに位置する画像ラスタまたはグラフィックス・オブジェクトのような、プラグイン・メディア・オブジェクトを処理するツールボックスの上に、調和するインターフェースの集合体を設ける。ハードウェア・ドメインは、コンピュータＲＡＭ、ビデオＲＡＭ（ＶＲＡＭ）、オンボードＧＰＵメモリ、およびカスタム・ハードウェアと関連付けられるメモリを含む。処理フレームワークをＣＰＬ（コンポーネント処理ライブラリ）と呼び、アプリケーション・ソフトウェアの開発および実行双方のためのフレームワークとしての役割を果たす。

【００１１】

別の形態では、本発明は、異なるプラットフォーム、オペレーティング・システム、グラフィックス・カード、およびシェーダ言語の範囲に対する透明なホスティング、ならびにＧＰＵに基づく効果の高速実行を可能にする、画像およびビデオ処理プログラミング・サービスを特徴とする。このサービスは、加速化画像処理を遂行するために、現行のコンピュータ・システムのコンポーネントとして、カスタムに供給されるＧＰＵに組み入れる。本発明によるフレームワークを用いて、空間的、時間的および機能的画像処理の加速化を達成する。また、本フレームワークは、画素毎に画素シェーディング機能を指定する能力も特徴とし、コンピュータ・システム上にある必要な数だけの画素シェーダを同時に利用し、可能であれば基礎になるプログラムのアルゴリズムに基づくベクトル・マス加速(vector math acceleration)も活用する。

【００１２】

概して言えば、別の形態においては、本発明は、複数の実行ドメインと、当該複数の実行ドメインの１つに関連付けられたメモリとを備えているメディア処理システムを特徴とし、メモリは、複数の実行ドメインの１つによって読み取り可能な命令を備えており、これらの命令を複数の実行ドメインの１つ上で実行すると、メディア処理機能を遂行する命令を受け入れさせ、メディア処理機能と関連付けるメディア・オブジェクトを受け入れさせ、メディア・オブジェクトは、当該メディア・オブジェクトのタイプ、メディア・オブ

ジェクトのフォーマット、およびメディア・オブジェクトと関連付けられているハードウェア・ドメインを指定する属性でラップされている。更に、実行ドメインの少なくとも１つに、メディア・オブジェクト上でメディア処理機能を遂行させ、このメディア処理機能を遂行するための命令は、メディア・オブジェクトと関連付けられているハードウェア・ドメインには依存しない形態で表現されている。本発明の実施形態には、以下の特徴の１つ以上が含まれる。

【００１３】

メディア処理機能を遂行するための命令は、メディア・オブジェクト・タイプおよび／またはメディア・オブジェクト・フォーマットには依存しない形態で表現されている。複数の実行ドメインはＣＰＵおよびＧＰＵを含み、福数の実行ドメインの１つはＣＰＵである。メディア処理機能とは画像効果のことであり、メディア・オブジェクトのタイプはラスタ画像である。画像効果は、ディゾルブ、色補正、テキストの挿入、および動き効果のうち１つを含む。メディア処理機能は、画像効果であり、メディア・オブジェクト・タイプはグラフィックス・オブジェクトである。複数の実行ドメインの各々は、下位命令ライブラリと関連付けられており、実行ドメインの１つと関連付けられている下位ライブラリの少なくとも部分集合は、実行ドメインの別の１つと関連付けられている下位ライブラリの対応する部分集合とは互換性がない。本システムは、メディア処理機能とメディア・オブジェクトと関連付けられているメディア・オブジェクト・タイプ、メディア・オブジェクト・フォーマット、および実行ドメインの少なくとも１つとの間における不一致を識別し、メディア・オブジェクトのタイプを別のタイプに変換すること、またはメディア・オブジェクトのフォーマットを別のフォーマットに変換すること、または別のハードウェア・ドメインをメディア・オブジェクトと関連付けることのいずれかによって、識別した不一致を解消する。

【００１４】

メディア・オブジェクトの属性は、容認可能な属性集合の中の１つであり、容認可能な属性集合は、新たなメディア・オブジェクト・タイプ、新たなメディア・オブジェクト・フォーマット、および新たな関連するハードウェア・ドメインのうち少なくとも１つを有する新たな属性を含むように増強することができ、命令を書き直すまたはコンパイルし直す必要はない。メディア・オブジェクトを複数の部分に分割し、第１実行ドメインを第２実行ドメインに接続するデータ・バスを通じてこれらの部分を順次送出し、一度にこれらの部分の１つ上においてメディア処理機能を実行する。命令は複数の処理ユニットを備えており、メディア処理機能は、複数の処理ユニットのうち少なくとも１つの第１処理ユニットを実行することによって遂行され、複数の処理ユニットのうち第２処理ユニットをコールする。命令は、コール先の処理ユニットからスレッドを生成し(spawning)、処理ユニットがメディア・オブジェクトに対してメディア処理機能を実行し続ける間に、このスレッドを非同期で実行することを伴う。

【００１５】

概して言えば、更に別の形態では、本発明は、メディア処理方法を特徴とし、メディア処理機能を実行するための命令を受け入れるステップと、メディア処理機能と関連付けられるメディア・オブジェクトを受け入れるステップであって、メディア・オブジェクトは、当該メディア・オブジェクトのタイプ、メディア・オブジェクトのフォーマット、およびメディア・オブジェクトと関連付けられているハードウェア・ドメインを指定する属性によってラップする、ステップと、複数の実行ドメインの少なくとも１つに、メディア処理機能をメディア・オブジェクト上で実行させるステップであって、メディア処理機能を実行する命令は、メディア・オブジェクトと関連付けられているハードウェア・ドメインには依存しない形式で表現されている、ステップとを含む。

【００１６】

概して言えば、更に別の形態において、本発明は画像処理システムを特徴とし、中央処理ユニット（ＣＰＵ）と、グラフィックス処理ユニット（ＧＰＵ）と、ＣＰＵと関連付けられているメモリとを備えており、メモリはＣＰＵによって読み取り可能な命令を備えて

おり、これらの命令をＣＰＵによって実行すると、当該ＣＰＵに、画像処理機能を実行する命令を受け入れさせ、画像処理機能と関連付ける画像を受け入れさせ、画像を、当該画像のフォーマットおよび画像と関連付けられているハードウェア・ドメインを指定する属性でラップし、ＧＰＵに、画像上で画像処理機能を実行させ、画像処理機能を実行する命令は、画像と関連付けられているハードウェア・ドメインには依存しない形態で表現されている。

【００１７】

概して言えば、別の形態において、本発明は画像処理方法を特徴とし、ＣＰＵ上で走るクライアント・アプリケーションから、画像処理機能を実行する命令を受け入れるステップと、クライアント・アプリケーションから、画像処理機能と関連付ける画像の指示を受け入れるステップと、画像を、当該画像のフォーマットおよび画像と関連付けられているハードウェア・ドメインを指定する属性でラップするステップと、ＧＰＵに、画像上で画像処理機能を実行させるステップであって、画像処理機能を実行する命令を、画像と関連付けられているハードウェア・ドメインには依存しない形態で表現する、ステップとを備えている。

【００１８】

また、前述の方法は、以下の特徴のうち１つ以上も含む。ＧＰＵは、関連するシェーダ言語を有し、画像処理機能を実行する命令は、シェーダ言語には依存しない形態で表現されている。命令の実行は、ＣＰＵ上で走るオペレーティング・システムによって制御し、画像処理機能を実行する命令は、オペレーティング・システムには依存しない形態で表現されている。ＧＰＵは、画像レンダリング・データ・バッファを含み、画像レンダリング・バッファのタイプは、テクスチャ、フレーム・バッファ・オブジェクト、マルチ・サンプル・レンダ・バッファ、リード専用画素バッファ・オブジェクト、ライト専用画素バッファ・オブジェクト、およびリード－ライト画素バッファ・オブジェクトのうちの少なくとも１つであり、画像は、画像レンダリング・バッファのタイプには依存しない形態で表現されている。ＧＰＵは、画像レンダリング・テクスチャ・パラメータを含み、画像レンダリング・テクスチャ・パラメータは、色空間、画素深さおよび画素範囲のうち少なくとも１つを備えており、画像は、画像レンダリング・テクスチャ・パラメータには依存しない形態で表現されている。ＧＰＵに画像上において画像処理機能を実行させることは、多重パス実行を含み、ＣＰＵ上に適時にコンパイルしたマルチパス画素プログラムをキャッシュし、画素プログラムを部分的にコンパイルし、ならびに部分的にコンパイルした画素プログラムをキャッシュし読み出す。ＣＰＵは、メモリと関連付けられており、ＣＰＵは、メモリの一部を、画像データを格納するために割り当て、ＧＰＵに画像上において画像処理機能を実行させることは、メモリの新たな部分を画像を格納するために割り当てずに、メモリの割り当てた部分をリサイクルすることを含む。画像は、８ビットＲＧＢ色空間画像、８ビットＹＣＣ色空間画像、または８ビットＹＣＣＡ画像として表され、ＧＰＵに画像上において画像処理機能を実行させる際、画像をＢＧＲＡテクスチャにパックすることを含む。画像処理機能を実行する命令は、画像を表すために用いられる色空間および／または画素深さ、および／または画素範囲には依存しない形態で表現されており、および／または画像を格納するために用いられるメモリ・レイアウトおよびパッキングには依存しない形態で表現されている。ＧＰＵに画像上において画像処理機能を実行させる際、ＧＰＵ上における処理スレッドの非同期の実行を伴う。

【図面の簡単な説明】

【００１９】

【図１】図１は、メディア処理システムを実装するための計算構成例のブロック図である。

【図２】図２は、ポータブルな開発および実行フレームワークを組み込んだメディア処理システムのソフトウェア・レイヤを示す図である。

【図３】図３は、ポータブルなフレームワーク内における処理ユニットの図である。

【図４】図４は、ポータブルな開発および実行フレームワークを組み込んだ画像処理シス

テムのブロック図である。
【図５】図５は、ポータブルな開発および実行フレームワークを用いたビデオ処理アプリケーションの一例の流れ図である。
【発明を実施するための形態】
【００２０】
本発明を実施することができる計算構成例１００を図１に示す。計算構成１００は、汎用コンピュータおよび／またはワークステーションにおいて見られる複数の電子コンポーネントを備えている。例えば、計算構成１００は、１つ以上の中央処理ユニット（ＣＰＵ）１０２、１０４、ＣＰＵ１０２、１０４がアクセスすることができ、ホスト・メモリまたは単にＲＡＭと呼ばれる、ランダム・アクセス・メモリ１０６、１つ以上のグラフィカル処理ユニット（ＧＰＵ）１０８、１１０、ＧＰＵがアクセスすることができ、ＧＰＵメモリまたはＶＲＡＭ１１２と呼ばれる、カスタムの企業固有ハードウェアおよびランダム・アクセス・メモリを備えることができる。また、この構成は、ディスク１１４（磁気ディスク、ソリッド・ステート・ディスク、即ち、ＲＡＭディスク）や、追加の記憶および処理エレメントも含むことができる。
【００２１】
計算構成１００は、マルチタスキング・オペレーティング・システム、Ｏ／Ｓ１１６、およびＧＰＵドライバ１１８を含む。Ｏ／Ｓ１１６は、ＰＣ上で走るMicrosoft Windows(登録商標）、ＭａｃＰＰＣ／ＭａｃＩｎｔｅｌ上で走るApple社のOS/X、およびLinuxを含む、一般的なオペレーティング・システムの１つを含む。Ｏ／Ｓ１１６は、ドライバ、上位ＡＰＩ　OpenGL（ＭａｃおよびLinux用）のようなＡＰＩ、およびDirectX（Microsoft）のような、クライアント・プログラム、ならびにメディア処理ソフトウェアのようなアプリケーション、例えば、Avid Media Composerをホストする。
【００２２】
上位ＡＰＩは、ＧＰＵ上のソフトウェア・レイヤとしての役割を果たし、プログラムが特定の処理およびレンダリング・ジョブをそれに送出させる。「３Ｄ　ＡＰＩ」という用語は、この文書では、「上位ＡＰＩ」と総合交換可能に用いられる。上位ＡＰＩは、ジェオメトリ(geometries)、テクスチャ、およびシェーダ・プログラム(shader program)のハードウェア上への「押し出し」(pushing)を許容し、更にユーザが彼らの上位シェーダ言語プログラムを、下位のハードウェアが認識できるハードウェア特定命令にコンパイル／拡張することを可能にする。
【００２３】
図２を参照すると、記載している実施形態は、ポータブルな開発および実行フレームワーク２０２を特徴とし、このフレームワーク２０２は、メディア・オブジェクトのハードウェア実行ドメイン、データ構造タイプ、およびデータ構造フォーマットの抽象化を含む。設けられる抽象化は、低レベル・ライブラリ２０４ａ～２０４ｆに属する、本来互換性がないアルゴリズムの集合上にあり、ライブラリ２０４ａ～２０４ｆの各々は特定のハードウェア・ドメインに特定的である。フレームワーク２０２は、種々のデータ・ドメイン、ハードウェア実行ドメイン、データ・タイプ、およびフォーマット間における、処理ユニット、変換器、およびＣＰＬレイヤ２１６内部に実装されているユーティリティを通じた相互作用を可能にする。ＣＰＬレイヤ２１６は、画像処理コンポーネント２０６、グラフィックス処理コンポーネント２０８、変換器２１０、およびユーティリティ２１２を含む。これらの処理コンポーネントの各々は、共通のデータ・オブジェクト構造２１４を用いて作動する。
【００２４】
開発および実行フレームワーク２０２をコンポーネント化したため、その固有性により、ホストしたメディア・オブジェクトの処理可能範囲を広げることができる。これによって、新たなメディア・タイプ、フォーマット、およびハードウェア・ドメインに合わせたプラグイン・アーキテクチャが可能になる。また、メディア・オブジェクトをそのネーティブなタイプ、フォーマットまたは計算ドメインにおいて処理するように最適化して、ホ

ストしたアルゴリズム・インプリメンテーション(implementations)の集合も拡張する。これらのアルゴリズムは、外部低レベル・ライブラリに収容してもよい。

【００２５】

ポータブル開発および実行フレームワーク２０２は、ライブラリ自体の修正を必要とせずに、既存の低レベル・ライブラリ２０４ａ～２０４ｆのアルゴリズムのホスティングを可能にする。低レベル・ライブラリ２０４ａ～２０４ｆは、それらのデータおよび処理アルゴリズムを同じパイプラインの中で用いることができるように、統一することもできる。低レベル・ライブラリ資源のマルチスレッド型利用は、ＣＰＬレイヤ２１６として反映される、ステートレス・クラス実行レイヤを通じて行われる。

【００２６】

ポータブル開発および実行フレームワーク２０２は、ライブラリ・インプリメンテーションには関係なくアルゴリズムを設定し、制御し、データに対して実行するために、標準化したインターフェース集合を設ける。更に、その全てのパラメータおよびプロパティを扱い、スクリプティング・システムを通じて使い易くするための標準的な構造も設ける。

【００２７】

ポータブル開発および実行フレームワーク２０２は、クライアント・アプリケーション・レイヤ２１８によって呼び出す。記載している実施形態では、クライアント・アプリケーションは効果２２０、２２２、および２２４をメディア・オブジェクトに適用することを含むメディア処理アプリケーションを含む。クライアント・アプリケーションを実行するとき、特に効果が要求されるとき、クライアント・アプリケーションはフレームワーク２０２とインターフェースして、利用可能なハードウェア・ドメインの資源を呼び出す。フレームワーク２０２は、特定のメディア・オブジェクト上において要求されたアルゴリズムの実行には、どのドメインが適しているか判断することができる。しかしながら、クライアント・アプリケーションは、ユーザが選択したハードウェア・ドメイン上で実行を強行することを許容する。加えて、クライアント・アプリケーションは、複雑なパイプラインの完全なカプセル化を達成するために、他の処理ユニット内における、それ自体の処理ユニットの再利用を許容する。

【００２８】

フレームワーク２０２が設けるハードウェア・ドメイン抽象化は、当該ドメインの処理機能を、それと関連のあるストレージおよび企業固有フォーマット(proprietary format)と共に束に纏める。例えば、ハードウェア・ドメインは、メディア・オブジェクトに割り当てられるディスク型、ＲＡＭ型、またはＧＰＵメモリ型データ・バッファ、およびバッファがどこに割り当てられているか認識しこれらのドメインに位置するデータに対して動作するように最適化されている実行コードを参照することができる。

【００２９】

これより、先に言及したポータブル性および低レベル・ライブラリ独立性の利点を達成するために、開発および実行フレームワーク２０２に実装する抽象化について説明する。記載する実施形態では、フレームワーク２０２をコンポーネント処理レイヤ（ＣＰＬ）と呼ぶ。

【００３０】

データの抽象化

【００３１】

フレームワーク２０２内では、データの抽象化を容易にする、コンポーネント・データ（ＣＤａｔａ）と呼ばれる、ラッパ(wrapper)をメディア・オブジェクトに供給する。記載している実施形態では、ＣＤａｔａラッパ（以下では、単にＣＤａｔａと呼ぶ）は、三部属性、即ち、データ・タイプ、データ・フォーマット、およびデータ・ドメインを通じて、ラッパ内部における特定的な各データ構造の記述を可能にする。データ・タイプは、ＣＤａｔａによってホストされるデータ構造の種類を記述する。データ・タイプの例には、ラスタ画像、曲線、メッシュ、他のタイプのパラメータ・メディア・オブジェクト、オーディオ、またはテキストが含まれる。データ・フォーマットは、ＣＤａｔａによってホ

ストされるデータ構造のフォーマットを記述する。例えば、データ・タイプがラスタ画像である場合、データ・フォーマットは、空間解像度、アスペクト比、色空間、および時間的フレーム・レートを指定するフォーマットを含む。データ・タイプが曲線である場合、データ・フォーマットは、その曲線が線形、または四分円、または立体曲線のどれか指定するフォーマットを含む。データ・タイプがオーディオである場合、フォーマットはＭＰ３、ＷＡＶなどを含み、データ・タイプがテキストである場合、フォーマットは、ＨＴＭＬ、ワード(Word)、ＸＭＬ等を含む。データ・ドメインは、メディア・オブジェクト・データの主要ハードウェア・ドメインを含み、それに割り当てられたバッファ、および／またはメディア・オブジェクト上で処理機能の実行を引き受けるハードウェアであってもよい。データ・ドメインの例には、ＣＰＵ、ＧＰＵ、セル・プロセッサ、Intel CorporationからのＬａｒｒａｂｅｅ　ＧＰＵが含まれる。

【００３２】

各ＣＤａｔａには、名称（ストリング）で参照して、プロパティを当てはめることができる。各プロパティは、スカラー、ストリング、データ・ブロック、または別のＣＤａｔａというような、特定のタイプを有する。ＣＤａｔａコンポーネントを通じて、ユーザはもっと高いレベルでメディア・オブジェクト（即ち、低レベル）データ構造を操作することができる。何故なら、そのインプリメンテーションの詳細は、標準的なＣＤａｔａインターフェースの背後に隠されているからである。

【００３３】

処理の抽象化

【００３４】

低ライブラリ動作およびアルゴリズムを、フレームワーク２０２の標準的実行パラダイムにホストする。この標準的実行パラダイムは、処理ユニット（ＰＵ）と呼ばれるスレッド安全構造(thread-safe construct)を特徴とする。実行パスに必要とされるパラメータは、ＣＣｏｎｔｅｘｔオブジェクトと呼ばれるオブジェクトを通じて取り扱われる。ＣＣｏｎｔｅｘｔは、入力／出力パラメータを含むＰＵ状態情報、ならびに所望の実行ドメイン、データ・タイプ、およびデータ・フォーマットを保持するオブジェクトである。記載している実施形態では、クライアント２１８がＣＣｏｎｔｅｘｔを作成し初期化して、ここではＰＵ　ＦＸと呼ぶインターフェースを通じて、この状態情報をＰＵに受け渡す。何故なら、記載している実施形態では、クライアント・アプリケーションはビデオ効果（ＦＸ）を実施するためにフレームワーク２０２を用いるからである。しかしながら、インターフェースはビデオ効果に限定されるのではなく、ＰＵ　ＦＸインターフェースを用いてクライアント・アプリケーションから別の機能をコールすることもできる。

【００３５】

処理の抽象化は、外部ＰＵおよび内部ＰＵを含む。外部ＰＵは、種々の低レベル・ライブラリの中にあるハードウェア特定アルゴリズムを実施する１つ以上の内部ＰＵを互いに結束するために用いられる。各内部ＰＵは、１つ以上のＣＤａｔａ属性に対して関連する動作を実施する低レベルのライブラリ特定コードを収容する。例えば、ぼけ処理(blur operation)は、２つの別個のライブラリの中にある２つのハードウェア特定インプリメンテーションを有する場合があり、一方がＣＰＵのため、他方がＧＰＵのためにある。外部ＰＵがぼけ処理を取り扱う場合、２つの内部ＰＵ、即ち、ＣＰＵぼけを取り扱うＰＵとＧＰＵぼけを取り扱うＰＵとを論理的に結束する。別の実施形態では、ＣＤａｔａラッパの概念を用いる代わりに、ポータブルなフレームワークがメディア・オブジェクトをその属性と関連付けることを可能にする別の仕方で、メディア・タイプ、フォーマット、およびドメイン情報をメディア・オブジェクトに添付することによって、メディア・オブジェクト・データの抽象化を実行する。

【００３６】

外部ＰＵは、所与の動作を伝えるために必要なＣＣｏｎｔｅｘｔに関するパラメータの標準的な集合を定める。各内部ＰＵは、ＣＣｏｎｔｅｘｔを通じて送信される標準的なパラメータ集合を、目標とする下位ライブラリに適した形態で受け渡すことを責務とする。

内部ＰＵは、ＣＰＬレイヤ１２６内部に実装され、要求されたドメイン、データ・タイプ、およびデータ・フォーマット（即ち、三部ＣＤａｔａ属性）に応じてタスクを実行するために必要な下位ライブラリ・コールを実行する。外部ＰＵは、非同期実行や、ホストした処理に関する情報等を受け渡すためのコンパイラ・インターフェースのような、動作実行の種々の形態を制御するために用いられる共通の標準的インターフェースの集合を有する。以下に、図３と関連付けて、ＰＵおよびメディア処理システム１００のその他のコンポーネントとの相互作用について説明する。

【００３７】

ＣＰＬフレームワーク２０２は、個々のメディア・オブジェクトを置くために、１つの統一座標（または基準）システムを定める。ＣＰＬメディア・オブジェクトは、位置およびサイズ・プロパティ、または位置、サイズ、および距離プロパティを有し、この一意の座標システムに関して位置付けられる。内部ＰＵは、ＣＰＬ座標システムからの位置、サイズ、および距離情報を、特定の下位ライブラリ（例えば、ＩＬ、Ｇｋ、．．．）座標システムに変換する。

【００３８】

フル・メディア処理システム・パイプラインの一例は、種々の動作を実施する数個の外部ＰＵと、外部ＰＵに対する入力および／または出力としての役割を果たす数個のＣＤａｔａと、種々の実行のパラメータを格納するために用いられる数個のＣＣｏｎｔｅｘｔとを含む。

【００３９】

場合によっては、ＰＵが複数(multiple)の個々のＰＵで構成され、既存のフレームワークからより複雑な動作を構築できるようにすることもある。ＰＵ内における個々の動作のグラフを抽出するためにインターフェースを設け、こうしてＣＰＬクライアントに内部ＰＵのグラフを露出する。例えば、キーヤ効果(keyer effect)は複数の効果段階、プレブラー(pre-blur)、キーイング、ポストブラー(post-blur)、拡大縮小(grow-shrink)、形状、および組成で構成される。ＣＰＬフレームワーク２０２は、クライアント２１８によって用いられる１つのキーヤＰＵを定め、このＰＵが粒度の高いＰＵのグラフを含む。これは、形状ＰＵを供給する拡大縮小ＰＵを供給するブラーＰＵを供給するキーＰＵを供給するブラーＰＵで構成され、最終的に組成ＰＵを供給する。キーヤＰＵに対する１回のクライアント・コールによって、この実行パイプライン全体が行われる。これの方が、アプリケーション・プログラマにとって使い易い。何故なら、必要な効果コール回数が減少し、抽象化のレベルが高められるからである。クライアントに利用可能な内部で使用するＰＵのグラフを作ることによって、クライアント・アプリケーション、例えば、メディア・プレーヤは、グラフ・エレメントＰＵレベルで必要なハードウェア資源を取り決める(negotiate)ことが可能になる。

【００４０】

ＣＰＬフレームワークは、これら３つの属性に対する処理コードが入手可能であることを条件に、特定のフォーマットおよびメディア・タイプに対して特定のドメイン上における特定のＰＵの実行を強制するために、ユーザがＣＣｏｎｔｅｘｔを通じて実行ドメインを指定することを可能にする。

【００４１】

ＧＰＵに対するような、大きなアップロードおよびダウンロードの不利(penalty)を有するハードウェア・ドメイン上における競合を抑えるために、そしていずれの特定のＣＤａｔａ属性について可能であればいつでも、ＰＵの実行を順次行う。この順次実行は、通例、ハードウェア・ドメインに特定的であり、目標ハードウェアの並列性およびパイプライン処理特性を考慮に入れている。データは、空間的および時間的に並べる(tile)ことができる。入力および出力は、自動的にＣＰＬフレームワーク・コアによって、更に小さいチャンク(chunk)に分割され、次いで同時または順次ハードウェアにアップロードされ、ＰＵに供給され、入力に挿入するためにダウンロードされる。これには、ＰＵがデータ・サイズに対するハードウェア特定制限を回避することができるという、追加の便益がある

。

【００４２】

競合制限およびデータ・サイズに対するハードウェア特定制限の回避の実現は、次のように進められる。広義の用語を用いると、ＣＰＬ実行のパスは、入力ドメインから実行ドメインへのデータのアップロードから開始し、この後に実行ドメインに対するデータ処理段階が続き、この後に実行ドメインから目標ドメインへのダウンロードが続く。入力／出力ドメインが実行ドメインと一致する場合、アップロードおよびダウンロード・ステップは設けられない。アップローディング／ダウンローディングが必要なときには、ドメイン毎に変化するデータ転送時間の負担を負うことになる。処理の性質によっては、このレイテンシ期間が、当該ドメインに位置するデータを処理するために必要な時間よりも遥かに大きくなる可能性がある。競合を制限しないと、ＰＵは、（１）入力データ集合全体のアップロードを開始し、（２）転送が完了するのを待ち、（３）処理を開始し、（４）処理が完了するのを待ち、（５）出力データ集合全体のダウンロードを開始し、（６）転送が終了するのを待つことになる。対照的に、競合を制限すると、ＣＰＬは、ドメイン支援同時転送および実行を利用することにより、入力および出力データをより小さなチャンクに分割し、それらのサイズの方が小さいことから、データ集合全体よりもアップロードおよびダウンロードが速く行われ、このプロセスを加速する。実行シーケンスは、（１）チャンク＃１を開始し、この動作の完了を待ち、（２）チャンク＃１の処理を開始し、チャンク＃２のアップロードを開始し、これら２つの動作の完了を待ち、（３）チャンク＃１のダウンロードを開始し、チャンク＃２の処理を開始し、チャンク＃３のアップロードを開始し、これら３つの動作の完了を待ち、（４）チャンク＃２のダウンロードを開始し、チャンク＃３の処理を開始し、チャンク＃４のダウンロードを開始し、これら３つの動作の完了を待ち、（５）処理するチャンクがなくなるまで以上のことを続ける。しかるべきチャンク・サイズを用いれば、目標ドメインのハードウェアは連続的にデータを処理しており、その間にその次のデータ集合をアップロードし、以前の結果をダウンロードする。

【００４３】

このように、ＣＰＬにおいて順次処理することにより、２つの便益が得られる。最初に、同時転送および実行を支援するドメインに対するアップロード／ダウンロード・レイテンシの影響を抑えることにより、システム性能向上が得られる。最初のチャンクをアップロードするとき（そしてまだ何も処理するものがないとき）、および最後のチャンクをダウンロードするとき（そして他の処理するものがないとき）にのみ、残留アイドル時間が発生する。第２に、入力データ集合が大き過ぎて実行ドメインのメモリに収まらない場合、またはその現行の仕様を超過する場合でも、目標ドメイン上で使用される資源はＣＤａｔａパラメータ当たり３チャンクを超えることは決してないので、これらの入力データ集合を処理することができる。

【００４４】

変換の抽象化

変換は、特殊な外部ＰＵの集合として実施する。これらは、１つのＣＤａｔａ属性から他の何らかのＣＤａｔａ属性に変換するために用いられる内部ＰＵ集合を結束する。変換は、変換する対象を用いて外部処理ユニットをコールすることによって、明示的に呼び出すことができ、または予期しなかった属性のＣＤａｔａが外部ＰＵに入力として与えられたときには暗示的に呼び出すことができる。不一致があったときはいつでも、ＣＰＬフレームワーク・コアによる要求に応じて、自動変換を行い、与えられたパラメータを常にＰＵが理解できることを保証する（要求駆動型実行）。

【００４５】

ＣＰＬ開発者によって新しい属性または新しいＰＵがＣＰＬフレームワークに追加されたとき、既存の属性インスタンスのための入力および出力変換器は新たな処理ユニットおよび新たな属性を、現存する集合に統合する。これによって、新たな属性および処理ユニット間における相互運用性を可能にする。例えば、ＩＬは主（ＲＡＭ）メモリに配置しなければならないＩＤＳＩｍａｇｅ型(kind)のデータに対して動作する。ＩＬ－ＧＰＵは、

ＧＰＵと関連のあるメモリであるＶＲＡＭに配置する必要があるＩＤＳＧＰＵＩｍａｇｅ型にフォーマットされたデータに対して動作する。変換器は、自動的に、データを一方のドメイン（この例では、ＩＬ／ＣＰＵ　ＲＡＭ／ＩＤＳＩｍａｇｅ）から他方（例えば、ＩＬ　ＧＰＵ／ＧＰＵ　ＶＲＡＭ／ＩＤＳＧＰＵＩｍａｇｅ）に変換する。

【００４６】

種々のオブジェクトにおいて見られるパラメータおよびプロパティには、文字列の名称が付けられている(string-named)ことによって、スクリプティング・エンジンによる容易なインターフェース処理が可能となる。

【００４７】

ＣＰＬフレームワーク２０２と関連のあるオブジェクトは、外部ライブラリ連結を必要としない個々のコンポーネントとして開発される。ＣＰＬプロトコルは、フレームワーク２０２内部にあるＰＵと、種々の利用可能な実行および記憶ハードウェア・ドメインに特定的な低レベル・ライブラリとの間にあるインターフェースを通じて実施される。ＣＰＬ開発者は、新たな低レベルライブラリを、集合、新たなデータ・タイプ、新たなＰＵに追加することができ、更にＣＰＬフレームワークを用いるクライアント・アプリケーションを変更することなく新たなインプリメンテーションを追加することによって、既に存在するＰＵを拡張することができる。

【００４８】

再度図２を参照すると、ＣＰＬレイヤ２１６は、クライアント・アプリケーション２１８と、ドメイン特定のメディア処理に用いられる低レベル・ライブラリ２０４ａ～２０４ｆとの間にある中間レイヤを表す。記載している実施形態では、ＣＰＬレイヤ２１６は、ＣＰＬデータ・オブジェクト２１４、即ち、開発および実行フレームワーク２０２全体を通じて用いられるデータ・オブジェクトと、データ・オブジェクト上で有意な動作を実行するオブジェクト集合を含み、正しく定められているインターフェースによって制御される関数コールと同等の機能を有するＣＰＬ　ＰＵとで構成されている。ＣＰＬ　ＰＵは、全体的にＣＰＬ：：ＩＰ２０６で示す画像処理ユニットと、全体的にＣＰＬ：：ＧＰ２０８で示すグラフィックス処理ユニットと、ドメイン、タイプ、およびフォーマット変換を実施するＣＰＬ：：変換器２１０で示す変換器と、ディスプレイ・ドライバのような、ＣＰＬ：：Ｕｔｉｌｓ２１２で示すユーティリティとを含む。

【００４９】

クライアント・アプリケーション２１８は、下位ライブラリ２０４ａ～２０４ｆのいずれに対しても直接的な連結を必要としない。ＣＰＬフレームワーク２０２の主要な特徴の１つは、その中にホストされているオブジェクトはプラットフォームにいは依存しないことである。オブジェクトは均一なコンポーネント・フォーマット（ＣＦ）に準拠しており、各コンポーネントはＣＦプラグインとして実施される。新たなオブジェクトまたは新たなドメイン・インプリメンテーションは、新たなプラグイン・ファイルを計算システム上にあるしかるべきフォルダ内に単に追加するだけで、クライアントに利用可能にすることができる。

【００５０】

ＣＰ：：データ・オブジェクト２１４は、データの関連するハードウェア・ドメイン、タイプ（例えば、ラスタ、パラメトリック形状）、およびフォーマット（オブジェクト・タイプに結束され、呈示されるデータの品質に関する情報を含む）を抽象化して、データを一意に表すために用いられる。ＣＰＬ：：データ・オブジェクト２１４は、ドメイン特定下位ライブラリ１０２ａ～１０２ｆにおいて定められるデータ・オブジェクトのいずれにも、それを取り巻くラッパとしての機能を果たす。ＣＰＬ：：ＩＰ２０６およびＣＰＬ：：ＧＰ２０８ＰＵの各々は、しかるべきＣＰＬ：：データ・オブジェクト２１４であればいずれでも受け入れる。フレームワーク２０２は、代理設計パターンを用いる要求駆動型実行を考慮している。例えば、ラスタは、画素が要求されるまで作成されず、連接動作によって発生する。

【００５１】

図３を参照すると、個々のＰＵ３００は、外部ＰＵ３０２および数個の内部ＰＵ３０４とで構成されている。ＰＵ３００は、ＣＰＬ処理ユニットからの画像ＰＵ（ＣＰＬ：：ＩＰ２０６）またはグラフィックスＰＵ（ＣＰＬ：：ＧＰ２０８）の集合の一部材であり、ＣＰＬ：：データ・オブジェクト１２４に対してグラフィックおよび画像処理を行う処理オブジェクトを表す。ＰＵ３００は、任意の数の入力ＣＰＬ：：データ・オブジェクト（ファイル・リーダＰＵのように、ソースＰＵについては０を含む）を取り込み、任意の数の出力ＣＰＬ：：データ・オブジェクトを生成する（出力表示を発生するＰＵのような、シンクＰＵ(sink PU)については０を含む）。ＰＵ３００は、ステートレス・オブジェクトであり、クライアントがそれに受け渡した実行コンテキストの中に、その内部またはコンテキスト状態を格納する。記載している実施形態では、ＰＵ３００はＦＸインターフェース３０６およびコンパイラ・インターフェース３０８を露出する。ＦＸインターフェース３０６は、クライアント・アプリケーション２１８とインターフェースする。クライアント・アプリケーション２１８は、例えば、メディア・プレーヤである。ＦＸインターフェース３０６を用いて、クライアント・アプリケーション２１８は、特定のタイプの目標メディア・オブジェクトや、目標メディア・オブジェクトを表す／格納する際のフォーマットや、効果を実行しようとするハードウェアには依存しない命令を用いて、ビデオ効果（ＦＸ）のようなメディア処理機能を実行するために必要な１つ以上のＰＵを呼び出す。

【００５２】

記載している実施形態では、ＦＸインターフェース３０６は、（ｉ）入力および出力ＣＰＬ：：データ・オブジェクトの指定、（ｉｉ）入力パラメータの指定、（ｉｉｉ）特定の実行ドメインを強制するか否か、（ｉｖ）特定の実行タイプおよびフォーマットを強制するか否かを含み、必要であれば、利用可能な変換器を利用する。これらの代わりにまたはこれらに加えて、他のパラメータを指定することもできる。

【００５３】

コンパイラ・インターフェース３０８は、ＰＵ３００が、支援する実行ドメイン、データ・オブジェクト・タイプ、およびフォーマットを含む、それに利用可能な能力を問い合わせることを可能にする。また、そのＰＵに対して好ましい実行ドメイン、タイプ、およびフォーマットも問い合わせる。コンパイラ・インターフェース３０８を通じて入手した情報によって、ＰＵ３００はホストＣＰＵ、ＧＰＵ、あるいはセル・プロセッサまたはＬａｒｒａｂｅｅ　ＧＰＵのような、カスタム・グラフィックス処理デバイスのようなその他のハードウェアのような、それに利用可能なハードウェア資源の使用を最適化することができる。また、コンパイラ・インターフェースを通じて供給される情報によって、ＰＵ３００は、データ・オブジェクトがクライアント・アプリケーション２１８の順次的機能ユニットを伝わっていく方法を適正に取り決めることが可能となる。例えば、メディア・プレーヤ・アプリケーションの場合、ＰＵ３００は、コデック、効果、変換器、およびディスプレイのような、種々のプレーヤ・ノードにオブジェクトを通過させることを、確実に可能にする。記載している実施形態では、コンパイラはＣＰＬレイヤ２１６内にある。

【００５４】

ＰＵ３００は、ドメイン特定ライブラリをコールする内部処理ユニット３１２ａ～ｄに、インターフェース３１０ａ～ｄを通じた低レベル・ライブラリ２０４ａ～ｆへのコールを供給する。例えば、ＧＰＵインターフェース３１０ａは、外部ＰＵ３０２をＧＰＵ１０８（図１参照）をコールする内部ＰＵ３１２ａとインターフェースさせ、ＧＰＵ画像ライブラリＩＬ－ＧＰＵ２０４ｂ（図２参照）およびＧＰＵグラフィックス・ライブラリＧｋ－ＧＰＵ２０４ｅの双方を呼び出すことができる。同様に、インターフェース３１０ｂは、外部ＰＵ３０２を、ＣＰＵ画像ライブラリＩＬ２０４ａおよびＣＰＵグラフィックス・ライブラリＧｋ２０４ｄをコールする内部ＰＵ３１２ｂにインターフェースする。他の下位インターフェースも、どの追加ドメインがメディア処理システム１００に利用可能かに応じて、ＰＵ３００に利用可能となる。例えば、外部ＰＵ３０２および内部ＰＵ３１２ｃ間のインターフェース３１０ｃは、Avid Technology社のＮｉｔｒｉｓシステムがある場合に利用可能となり、ＰＵ３００はＮｉｔｒｉｓ画像ライブラリＩＬ－ＤＬＥ２０４ｃと

インターフェースすることが可能になる。同様に、インターフェース３１０ｄは、セル・プロセッサがあれば利用可能となり、外部ＰＵ３０２は、セル・プロセッサをコールすることができる内部ＰＵ３１２をコールすることが可能になる。加えて、他のインターフェースも利用可能になれば、ＰＵ３００は、ディスク・ドライブのようなその他のドメインと、パーザ・ライブラリ２０４ｆを通じて相互作用することが可能になる。

【００５５】

ＣＰＬ：：ＩＰ　ＰＵ２０６は、画像処理動作を実行し、主にラスタ画像であるメディア・オブジェクトを取り扱う。ＣＰＬ：ＩＰ　ＰＵの例には、色補正、ぼけ、および形状に基づく光沢(matte)調節が含まれる。ＣＰＬ：：ＧＰ　ＰＵ２０８は、幾何学およびその他のグラフィックス動作を行い、主にパラメータ表現による曲線、表面、および立体であるメディア・オブジェクトを取り扱う。ＣＰＬ：：ＧＰ　ＰＵの例には、絵文字発生器や形状変形が含まれる。

【００５６】

ＰＵの実行は、ＣＤａｔａ属性、即ち、指定されたデータ・ドメイン、タイプ、およびフォーマットに基づく実行属性に応じて進む。実行属性は、クライアント２１８によって強制されるか、またはＰＵ自体が自動的に決定する。特定の実行属性を強制するために、クライアント２１８は、クライアント２１８およびＣＰＬレイヤ２１６間のインターフェースを用いて、コンテキストから所望のドメイン、タイプ、およびフォーマットを返す。例えば、クライアント２１８またはＰＵは、ＰＵの同期実行を強制することができ、その場合、メソッドはブロッキング(blocking)であり、あるいは非同期実行を強制することができ、その場合メソッドは非ブロッキング(non-blocking)である。実行を非同期にするためには、クライアント２１８は「コールバック助言」をコンテキスト・オブジェクトＣＣｏｎｔｅｘｔを通じて戻す。

【００５７】

ある種のドメインでは、入力メディア・オブジェクトをより小さなチャンクに分圧することが必要な場合や、有利な場合もあり得る。例えば、ＣＰＵがラスタ・メディア・オブジェクト全体を一度で処理できる広大なメモリを有していても、ＧＰＵの方がメモリ制約が大きく、ラスタ全体を取り扱うことができない場合がある。この問題を克服するために、タイル分割(tiled)実行モデルが必要となる。開発フレームワーク２０２は、クライアントからは隠されている自動タイリング・メカニズムを提供する。自動タイリング・メカニズムは、入力オブジェクトをタイル状にして、ＰＵに供給し、次いで意図する出力を出力タイルから構成し直す。このメカニズムは、現在用いられているハードウェア構成およびドメインに応じて個別に作成する。

【００５８】

タイル型実行モデルは、マルチ・スレッディングを必要としない。しかしながら、特定のドメインにおいて、バッファ転送および実行を並列に実行することが許される場合、フレームワーク２０２はこの特徴を利用する。処理を更に小さな実行ブロックに分割し並列化することによるマルチ・スレッディングは、ＣＰＬフレームワーク・レベルではなく、マルチ・スレッダ・ブロック２２６によって図２に示すように、クライアント・レベルにおいて管理する。マルチ・スレッド様式でＰＵにコールするのは、クライアントの責務である。場合によっては、この目的のために、コールされるＰＵの対象領域およびフィールド・マスキング機能を呼び出すことができる。これを行うためには、ＣＰＬを意識したマルチ・スレッダ・サービスへのアクセスを、クライアント・アプリケーション２１８に供給する。これらのサービスは、実行する動作、用いられるドメインおよびフォーマット、ならびに許可される並列タスクの数、スレッドの一群の指定、スレッド親和性(affinity)および優先度のような、クライアントが供給する追加の指令に基づいて、ＰＵコールをいかにしてマルチ・スレッド処理またはパイプライン処理するかを決定する。

【００５９】

ＣＰＬ：：変換器２１０は、特殊化したＰＵであり、三部属性（タイプ、フォーマット、ドメイン）を有するＣＰＬ：：データ（Ｃｄａｔａオブジェクト）を入力として取り込

み、これを異なる属性を有する指定の出力ＣＤａｔａオブジェクトに変換する。変換器２１０は、メディア・オブジェクトを１つのタイプから別のタイプへ、１つのフォーマットから別のフォーマットへ、そして１つのドメインから別のドメインに変換する。場合によっては、変換器ＰＵが順次行うのではなく、１回のステップでメディア・オブジェクト属性の３つのコンポーネントの内１つよりも多いコンポーネントを変換する場合もある。タイプ変換器は、曲線、表面、およびボリューム(volume)のような、パラメータで定めたグラフィックス・オブジェクトを、例えば、シーン・レンダラ(scene renderer)が用いるような、ラスタ画像に変換するラスタライザを含む。逆に、「シンセサイザ」型変換器は、ラスタ画像を、例えば、マジック・ワンド(magic-wand)におけるような、パラメータで定めたオブジェクトに変換する。フォーマット変換器は、画像フォーマット変換器を含み、例えば、ＹＣＣ画像をＲＧＢ画像に変換する。ドメイン変換器の一例では、ＧＰＵと関連のあるメディア・オブジェクトを、ＣＰＵホストと関連のあるメディア・オブジェクトに変換する。変換器２１０は、分散したユーティリティ・ルーチン集合として実装するのではなく、ＣＰＬフレームワーク２０２の内部に統合する。

【００６０】

ＣＰＬユーティリティＰＵ２１２は、ライン・ツールＴｏ（）や矩形ツールＲｅｃ（）のような原始的描画ツール、ならびにファイルに対する単純なリーダおよびライタを含む。これらのＰＵは、主に検査の目的で利用される。

【００６１】

下位ライブラリ１０２ａ～ｆは、下位ツールボックスを備えており、これらは種々のドメインおよびプラットフォーム特定画像およびグラフィック処理を実施する。記載している実施形態では、利用可能な下位ライブラリには、ＣＰＵ１０２上のラスタ／画像処理機能（ＩＬ２０４ａ）、ＣＰＵ１０２上のグラフィック処理（Ｇｋ２０４ｄ）、ＧＰＵ１０８上のラスタ／画像処理（ＩＬ－ＧＰＵ２０４ｂ）、ＧＰＵ１０８上のグラフィック処理（Ｇｋ－ＧＰＵ２０４ｅ）、およびディスク１１８上に位置するまたはこれに出力するメディア・オブジェクトの処理（パーザ２０４ｆ）を実施するライブラリが含まれる。

【００６２】

各下位ライブラリは、総じて互いに依存しない。これらが収容するオブジェクト・タイプ、これらが実施する抽象化のレベル、およびそのシンタックス間には互換性がないと考えればよい。

【００６３】

これより、ＧＰＵ１０８および１１０のような、グラフィックス処理ハードウェア・ドメインのために主に開発されたハードウェアを用いることによる加速化画像処理の推移および実行を容易にするフレームワーク２０２の用途について更に詳しく説明する。画像処理は、メディア・プレーヤ、エディタ、または画像処理システムのような、画像処理機能を含むサービス・クライアント・アプリケーション２１８を必要とする。先に展開した概念を用いて、記載している実施形態はこのような加速化画像処理を、ここではＧＰＧＰＵ処理と呼ぶ、１つ以上のＧＰＵ上で、ラスタ画像またはグラフィックス・オブジェクトに対応するＣＤａｔａタイプとＧＰＵに対応する実行ドメインとを有するメディア・オブジェクトのために実施する。ＣＰＬレイヤ２１６は、DirectXおよびOpenGLのような上位画像処理ＡＰＩ、あるいはPinnacle 3D-ServerまたはApple Core Imageライブラリのような現在利用可能ないずれの技術に対しても、ＧＰＧＰＵ抽象化を可能にする。

【００６４】

図４を参照すると、ＣＰＬ抽象化レイヤ２１６は、ＧＰＧＰＵサブレイヤ４０４および３Ｄレンダリング・レイヤ４０６を通じて、ＧＰＵ上でクライアント画像処理システム４０２を実装する。ＧＰＧＰＵサブレイヤ４０４は、画素、ベクトル、およびデータ処理のためにＧＰＵを使い易くする。３Ｄレンダリング・サブレイヤ４０６は、ＧＰＵ上で３Ｄシーンをレンダリングするために、３Ｄ　ＡＰＩを抽象化する。レイヤ４０４および４０６は、OpenGL４１０、DirectX４１２、コア画像４１４、および３Ｄ－サーバ４１６、ならびにその他の企業固有のＡＰＩまたは公開ＡＰＩの部分集合で構成することもできる。

図示の実施形態では、下位ライブラリは、Microsoft Corp.からのＷｉｎｄｏｗｓ(登録商標）オペレーティング・システムを走らせるＰＣ（４１８、４２２）、およびＭａｃ　ＯＳ　Ｘのような、Apple Inc.,が提供するオペレーティング・システム（４２０、４２４）を含む、市販のパーソナル・コンピュータ・プラットフォームによってホストされるＧＰＵ上に実装されている。

【００６５】

先に論じたように、フレームワーク２０２は、ビデオ効果２２０、２２２、および２２４（図２参照）が、ＣＰＵ１０２、オペレーティング・システム１１６、およびグラフィックス・ハードウェア１０８、１１０を含むメディア処理システムの処理システム１００の選択には依存しないことを可能にする。ＧＰＧＰＵ処理を実行するために、フレームワーク２０２は主にＧＰＵ１０８を画素処理エンジンとして用いて、ＧＰＵ上のラスタ下位ライブラリおよびグラフィックス下位ライブラリの部分集合を、それぞれ、ＩＬ－ＧＰＵ２０４ｂおよびＧｋ－ＧＰＵ２０４ｅとして実装する。ＧＰＵのＧｋ　３Ｄレンダリング能力も利用することができ、その結果光と影を伴う高品質のレンダリングが得られる。ＧＰＧＰＵレイヤ４０４および３Ｄレンダリング・レイヤ４０６。

【００６６】

また、フレームワーク２０２は、ＧＰＵ加速画像処理システムも提供する。このシステムは、テクスチャ、フレーム・バッファ・オブジェクト、マルチ・サンプル・レンダ・バッファ、およびリード専用／ライト専用／リード－ライト画素バッファ・オブジェクト（即ち、多くの異なる画像レンダリング・データ・バッファ）を１つの画像インターフェースに抽象化する。加えて、フレームワーク２０２は、画像レンダリング・テクスチャ・パラメータを、色空間、画素深さ、画素範囲を含む１つの画像インターフェースに抽象化する。

【００６７】

記載している実施形態では、８ビット・テクスチャの転送を最適化する。当然そして遺物(legacy)の問題のために、ＧＰＵドライバは８ビットＢＧＲＡフォーマットを直接転送する。他の８ビット・フォーマット全ては、最初に、ＣＰＵ上でＢＧＲＡフォーマットへの変換段階で処理され、その後でネーティブに(native fashion)転送される。記載している実施形態では、ＧＰＵドライバを「騙して」、ＲＧＢ、ＹＣＣ、またはネーティブ、ＢＧＲＡテクスチャのような別のフォーマットのいずれであれ、他の８ビット・テクスチャ・フォーマットを受け入れさせる。これによって、低速ＣＰＵ変換段階の迂回が可能になる。このような、ＧＰＵ上を「そのまま」転送される画像データにアクセスするために、ＧＰＵ上に格納されているシェーダ・プログラムは、本当のソース・データ・フォーマットを知らされており種々のチャネルがアクセスされている際に実行中にこれらをその正しい位置にアンスクランブルする小さなソフトウェア・レイヤを用いる。これはＧＰＵ自体上で実行されるので、変換を取り扱うためにドライバを利用する標準的な内蔵変換方法よりもはるかに速く実行する。この転送最適化は、８ビット・テクスチャのみに適用する。何故なら、１６ビット浮動小数点のような、更に大きなフォーマットの方が遅れて開発されたのであり、遺物のインプリメンテーションや慣例による影響を受けないので、ネーティブにＲＧＢＡに転送されるからである。

【００６８】

ＧＰＵ加速化画像処理システムは、ＲＧＢＡ画像データを、ホスト－ＧＰＵメモリ間転送およびＧＰＵ処理に最適なフォーマットにパックすることができる。８ビット・データのホスト－ＧＰＵ間転送に最適な内部フォーマットはＢＧＲＡであり、一方他の全てのデータ・タイプのホスト－ＧＰＵ間転送に最適な内部フォーマットはＲＧＢＡである。一実施形態では、８ビットＲＧＢ色空間画像の全てをＢＧＲＡテクスチャにパックする。画素チャネル・レイアウト変換レイヤは、画素プログラムの全てをラップ(wrap)して、これらのプログラムが画素をＲＧＢＡとしてアクセスし書き込むことができるようにする。このレイヤは、ＧＰＵテクスチャにおける正しいチャネルにアクセスするために、リード／ライト動作を変換する。

【００６９】
一実施形態では、ＧＰＵ加速化画像処理システムは、ＹＣＣ画像を、ホスト－ＧＰＵメモリ間転送およびＧＰＵ処理に最適なフォーマットにパックする。この場合、８ビット・データ用内部フォーマットはＢＧＲＡであり、８ビットＹＣＣ色空間画像をＢＧＲＡテクスチャにパックし、その他の全てのデータ・タイプ用の内部フォーマットはＲＧＢＡである。他のＹＣＣ色空間画像は全て、ＲＧＢＡテクスチャにパックする。画素色空間変換レイヤは、画素プログラムをラップして、このプログラムが画素をＲＧＢＡデータとしてアクセスし書き込むことができるようにする。ＣＰＬレイヤ２１６は、実行中に色空間変換を行うために、リード／ライト動作を変換する。

【００７０】
別の実施形態では、ＧＰＵ加速化画像処理システムは、別個のアルファ・チャネルのＹＣＣ画像を（ＹＣＣＡ）、ホストからＧＰＵへのメモリ転送に最適なフォーマットでパックし、８ビット・データ用ＢＧＲＡを用いたＧＰＵ処理は、全ての８ビットＹＣＣＡ色空間画像をＢＧＲＡテクスチャにパックし、他の全てのデータ・タイプにはＲＧＢＡを用いる。他のＹＣＣＡ色空間画像はＲＧＢＡテクスチャにパックされ、一方アルファ・チャネルのパッキングは、矩形領域をテクスチャの右側に添付することによって行う。このようにして、ＹＣＣＡ画像をホストからＧＰＵへの転送に最適な１つのテクスチャに格納する。アルファ・チャネルを水平にパックすることには、テクスチャ・メモリ空間を浪費しない利点があり、アルファ・チャネルの空間解像度を、クロミナンス・チャネルの空間解像度と異ならせることができる。

【００７１】
本システムは、適時にコンパイルしたマルチ・パス画素プログラムを、インテリジェントな部分的プログラム・コンパイル、キャッシング、およびパス毎の読み出しによってキャッシュすることができる。

【００７２】
本システムは、ＧＰＵデータ・バッファ・タイプの各々を、メモリ・プールを用いてリサイクルする。プール内に収容されるバッファ・タイプは、いずれのフォーマットのテクスチャも、リード専用／ライト専用／リード・ライト画素バッファ・オブジェクト、フレーム・バッファ・オブジェクト、およびマルチ・サンプル・レンダ・バッファを含む。ＧＰＵバッファのリサイクルは、バッファを割り当てそして割り当てを解除するよりも遥かに速い。システムがビデオ処理を実行しているとき、バッファのリサイクルによって、ＶＲＡＭ１１２の断片化を回避するという便益が追加される。

【００７３】
別の実施形態では、ＧＰＵ加速化画像処理システムは、色空間（ＲＧＢ、ＹＣＣ６０１、ＹＣＣ７０９）、画素深さ（８ビット、１６ビット、３２ビット、整数／浮動小数点／符号なし）、画素範囲（ビデオ・レベル、グラフィック・レベル、正規化浮動レベル）、および／またはメモリ・レイアウトおよびパッキング（ＲＧＢ、ＲＧＢＡ、ＢＧＲＡ、４２２、４４４、分離アルファ(separate alpha)、上から下／下から上）の自動取扱を設ける。

【００７４】
また、ＧＰＵ加速化画像処理システムは、アルゴリズムに対する自動マスキング・サービスも提供することができ、別個のマスク画像を用いた処理画素マスキング動作、ライト・フィールド（奇数／偶数ライン）マスキング、処理チャネル・マスキング（赤－緑－青－および／またはアルファ）マスキング、および／または対象領域マスキングを含む。

【００７５】
また、本システムは、ホストからＧＰＵメモリへの転送およびＧＰＵからホスト・メモリへの転送の改善を、複数の表示構成、および使い易い画素プログラム－Ｃ＋＋機能オブジェクト結束メカニズムによって達成する。

【００７６】
種々の実施形態では、フレームワーク２０２は、以下のデータ構造および機能を、画像

処理アプリケーションの開発者に、計算構成１００に接続されているユーザ・インターフェース１２０を通じて提供する。開発者にメモリ割り当ておよび所有権（ホストまたはＧＰＵ上）を定義および／または管理させる画像機能、テクスチャ、リード専用画素バッファ・オブジェクト、ライト専用画素バッファ・オブジェクト、リード－ライト画素バッファ、オブジェクト、フレーム・バッファ・オブジェクト、および複数のサンプル・レンダ・バッファ・オブジェクトを含む多くの種類のＧＰＵバッファへのインターフェース、ＲＧＢ、ＹＣＣ６０１、およびＹＣＣ７０９のような色空間、８ビット、１６ビット、３２ビット、整数／符号付き浮動小数点／符号なしのような画素深さ、ビデオ・レベル、グラフィック・レベル、および正規化浮動レベルのような画素範囲、ならびにＲＧＢ、ＲＧＢＡ、ＢＧＲＡ、４２２、４４４、分離アルファ、上から下／下から上を含むメモリ・レイアウトおよびパッキング。また、このフレームワークは、画像プール機能も提供し、各バッファを割り当てそして割り当てを解除するよりも高速なバッファのリサイクル、および計算構成１００におけるＧＰＵバッファのいずれのリサイクルをも含む高速画像リサイクルを開発者に定義および／または管理させる。

【００７７】

フレームワーク２０２は、画素プログラムを書くことができるＣ型言語、画素プログラムを、複数の入力および出力を有するアルゴリズムをサポートし、更に入力を有さないソース・アルゴリズムをサポートする単純なＣ＋＋機能オブジェクトに結束するベーク・クラスを提供することができる。また、このフレームワークは、再利用のためにＣ＋＋関数オブジェクトを保持するためのライブラリ、ならびに画素プログラムの適時コンパイル、既にコンパイルしたプログラム、多重パス画素プログラムのハッシングおよび高速読み取りを容易にする画素プログラム・キャッシュも特徴とする。パスとは、画像の一部または全体に対する何らかの画素動作の実行であり、適時コンパイルおよびキャッシュは、キャッシュの粒土が単一パスであるパス・レベルにおいて実行する。

【００７８】

種々の実施形態において、フレームワーク２０２はいずれの処理動作に対する自動マスキング・サービスも提供することができ、別個のマスク画像を用いた処理画素マスキング動作、ライト・フィールド（奇数／偶数ライン）マスキング、処理チャネル・マスキング（例えば、赤－緑－青－および／またはアルファ）、および／または対象領域マスキングを含む。また、画像をホスト・メモリからＧＰＵメモリに、そしてＧＰＵメモリからホスト・メモリに転送する機能の最適化、およびフレームワーク画像のリアル・タイム表示のためのウィンドウ・インターフェースも提供する。

【００７９】

フレームワーク２０２内部では、ＧＰＵを用いる処理ユニットをアクティブ・オブジェクトとすることができる。したがって、開発者は、このオブジェクトをコールして非ブロッキング様式で作業を実行するプログラムを作成することができる。コール元プログラムは、他のタスクを進め、処理ユニット内部で実行が進む間に、将来値を保持することができる。最終同期点が発生し、結果を一緒に送り出さなければならないとき、ブロッキング様式で将来値にアクセスすることができ、データを併合することができる。処理ユニットは、ＧＰＵ１０８、１１０の１つまたはその他のドメインのような、特定の種類のハードウェアとのその親和性を公開する能力を有し、所与の作業単位を片づけつつ、どのハードウェアを用いるべきかに関する判断を行うのに役立つ。

【００８０】

フレームワーク２０２とインターフェースする開発者は、新たな画像処理アルゴリズムを書き、それを既存のアプリケーションに統合したり、または既存の画像処理ルーチンを再利用することができる。フレームワーク２０２は、目標ハードウェア・ドメイン、データ・タイプ、およびデータ・フォーマット毎に、新たなアルゴリズムを異なるバージョンで書く必要性を未然に防ぐ。これによって、新たな機能をクライアント２１８に追加するのに必要な時間を短縮し、コストを削減して、デバッグおよび互換性の問題を軽減する。

【００８１】

図５は、クライアント２１８と下位ライブラリＩＬ－ＧＰＵ２０４ｂとの間における、ＧＰＵフレームワーク・レイヤ５０２を通じたデータの流れおよび実行を示す。ＧＰＵフレームワーク・レイヤ５０２は、ＧＰＵドメインに要求が発行されたときに、ＣＰＵレイヤ２１６内部におけるＰＵからのコールを受け取る。更に具体的には、図５は、クライアント・アプリケーション２１８およびインターフェースとＩＬ－ＧＰＵ内に実装されているエレメントとの間における画素データの流れ、画像の割り当て、Ｃ＋＋関数コール、ならびに画素プログラム・コードを示す。太めの矢印は画素データの流れを示し、破線の矢印はメモリ割り当てを追跡し、太い破線の矢印は、ＩＬ－ＧＰＵとの全動作、即ち、ＧＰＵにデータをアップロードし、プログラムを実行し、出力をＣＰＵにダウンロードする動作を遂行するために必要となるＣ＋＋関数コールを示す。破線は、プログラムがどこに位置するか、そしてどこで実行が要求されるかを示す。

【００８２】

クライアント・アプリケーション２１８のために新たな画像処理アルゴリズムを実装するために、開発者はOpenGL GLSLおよびDirect3D HLSL間の相違を抽象化するコンパイラ・マクロを有するＣ型シェーダ言語を用いて、処理パス毎に別個の画素プログラム５０４を書く。新たなＣ＋＋関数オブジェクト・クラス５０６は、主要画素プロセッサのベース・クラスから導出され、処理パス毎に画素プログラム・コードを戻す１つの関数を実装する。

【００８３】

主要画素プロセッサのベース・クラスは、入力画像５０８、出力画像５１０、対象領域のパラメータ、およびマスキング・パラメータを含むクライアント・アプリケーション・パラメータを受け入れる。実行時に、主要画素プロセッサのベース・クラスは、新たなアルゴリズムを実施するＣ＋＋関数オブジェクトの実行時に、以下のアクションを適時に実行する。（１）処理パス毎に、画素プログラム・キャッシュ５１２をチェックして、当該画素プログラム・パスが既にコンパイルされているか否か調べ、コンパイルされていない場合、（ａ）コードを検索して、導出されたクラスから、新たなアルゴリズムを実装する各処理パス求め、（ｂ）入力および出力画像画素深さ、画素範囲、色空間およびメモリ・レイアウトに応じて、自動変換機能を画素プログラムに添付し、自動変換機能は、コンパイルが成功するために画素を読み取り書き込むための新たな画素プログラムにおいてコールされるフック(hook)であり、（ｃ）クライアント・アプリケーション・パラメータに応じて自動画素マスキング動作を画素プログラムに添付し、（ｄ）画素プログラムのコンパイルのためにＯｐｅｎＧＬドライバにコードを送り、（ｅ）コンパイルしたプログラムを画素プログラム・キャッシュ５１０に格納する。（２）入力テクスチャを、レンダリングのためのソース・テクスチャとして結束する。（３）また、出力テクスチャをOpenGLフレーム・バッファ（レンダ目標と言っても分かる）として結束する。（４）処理パス毎に、OpenGLによって１つまたは多くのテクスチャを施した矩形のレンダリングを開始する。レンダリングは、直前のステップにおいて画素プログラム・キャッシュから得た画素プログラムを用いて実行する。対象領域に対するクライアント・アプリケーション・パラメータが、矩形およびOpenGLビューポートのサイズを決定する。

【００８４】

新たなアルゴリズムを実装するＣ＋＋関数オブジェクトは、ＯｐｅｎＧＬテクスチャをラップする画像を受け付けるだけである。正規画像をシステムのホスト・メモリからビデオ・メモリにおけるOpenGLテクスチャに変換する手段のために、他の関数が用意されている。

【００８５】

ＧＰＵ実行時コンポーネント（ＩＬＧＰＵ２０４ｂ）は、前述のＣ＋＋関数オブジェクトをコールすることができるように、以下のサービスを提供する。（１）（ａ）Ｃ＋＋関数に対する入力および出力として用いられるテクスチャ、（ｂ）ＧＰＵからホストへの高速転送のためのリード専用画素バッファ・オブジェクト、（ｃ）ホストからＧＰＵへの高速転送のためのライト専用画素バッファ・オブジェクト、（ｄ）多種多様なホスト・バッ

ファのためのリード－ライト画素バッファ・オブジェクト、（ｅ）クライアント・アプリケーションに露出されない、出力用テクスチャと組み合わせてフレームワークが用いる、フレーム・バッファ・オブジェクト、および（ｆ）クライアント・アプリケーションに露出されない、エリアシング防止用にフレームワークが用いる、複数のサンプル・レンダ・バッファ・オブジェクトを含む、種々のタイプのバッファのためのメモリ割り当て機能。（２）（ａ）画素深さ、範囲、色空間、およびメモリ・レイアウト、（ｂ）ＢＧＲＡのような８ビット・データ配列、およびＲＧＢＡのようなその他のデータ・タイプの配列を含む最適テクスチャ・チャネル配列を用いた、書き込み専用画素バッファ・オブジェクト・ホスト・メモリ－ＧＰＵテクスチャ・メモリ間転送、（ｃ）ＧＰＵテクスチャ・メモリからリード専用画素バッファ・オブジェクト・ホスト・メモリ、および（ｄ）同時に発生する変換および転送によって、１つの関数によって設けられるメモリ割り当て機能（前述の点（１）において説明した）の種々の組み合わせの単一機能における画像転送および変換。ＧＰＵによってネーティブにサポートされる画素フォーマットには、レンダリング・パスは不要である。１回のレンダリング・パスは、変換機能によって自動的に実行され、ＧＰＵ上において実際の変換算術処理を遂行する。これは、ＣＰＵよりも遥かに速くこれらの動作を実行する。

【００８６】

フレームワーク５０２は、メモリ・プール５１４を、前述の割り当て機能が用いる背景サービスとして提供する。これは、バッファをそのサイズおよびタイプに応じて、使用済みバッファおよびリサイクル準備完了バッファの２つのリストにハッシュする(hash)ことによって動作する。割り当て機能は、リサイクル準備完了バッファが利用可能な場合、メモリ・プールからバッファを読み出す。割り当て解除機能は、バッファをリサイクル準備完了バッファ・リストにつけ加える。

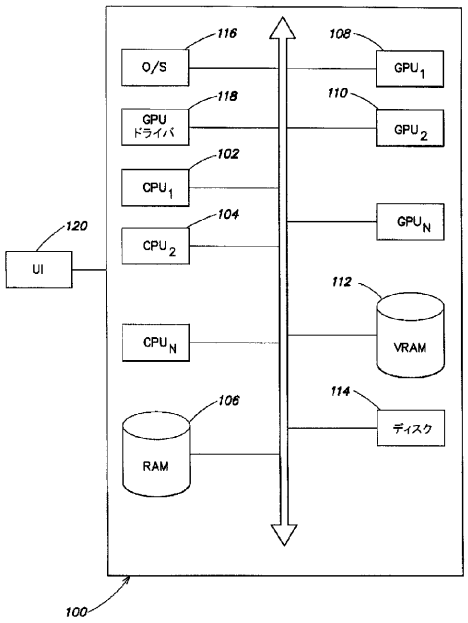
【００８７】

メモリ・プール５１４は、以下の技法によって、ビデオ処理用途における割り当て機能の性能を向上させる。しかるべきときはいつでも、ビデオ処理アプリケーション２１８は、同じサイズの画像を再利用し、フレームワーク５０２が画像バッファをリサイクルすることを可能にすることによって、OpenGL標準的割り当て機能と比較して、約２桁割り当てを高速化する。加えて、メモリ・プールはクライアント・アプリケーション２１８に、ビデオ・メモリを管理するための問い合わせ機能も提供する。更に、メモリ・プールはメモリ断片化の問題を軽減する。

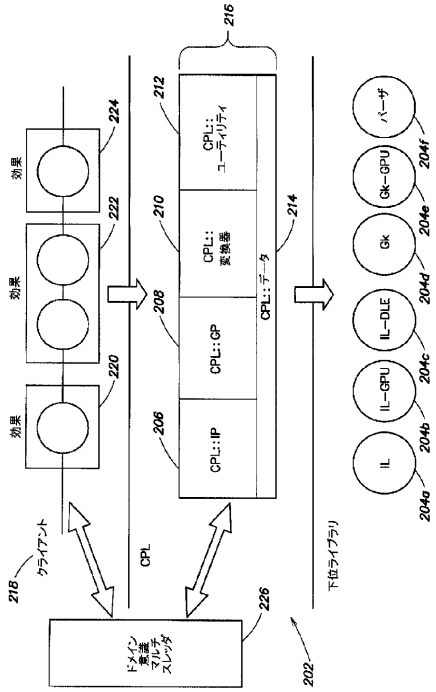
【００８８】

以上、実施形態例について説明したが、以上のことは例示に過ぎず限定ではなく、一例として呈示したに過ぎないことは、当業者には明白なはずである。複数の変更やその他の実施形態も、当業者の範囲内のことであり、本発明の範囲に該当するものとして想定している。

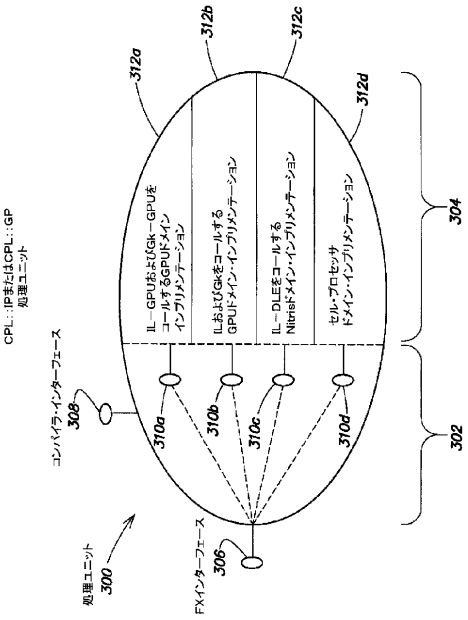
【図１】


O/S
GPU ドライバ
CPU1
CPU2
CPUN
UI
RAM
GPU1
GPU2
GPUN
VRAM
ディスク
116
118
102
104
120
106
108
110
112
114
100

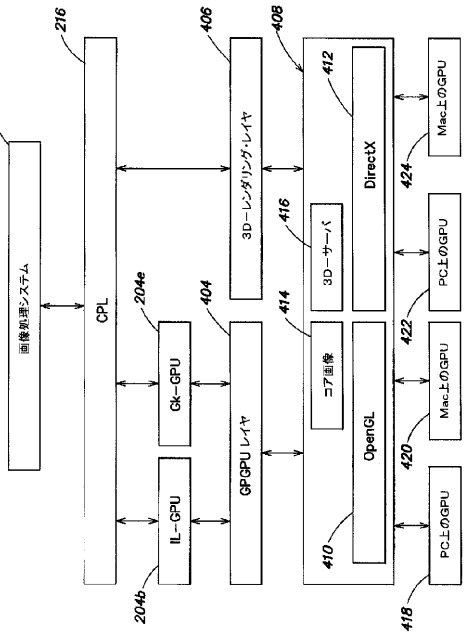
【図２】


効果
効果
効果
CPL
CPL::IP
CPL::GP
CPL:: 変換器
CPL:: ユーティリティ
CPL::データ
IL
IL-GPU
IL-DLE
Gk
Gk-GPU
バーザ
ドメイン 衰退 マルチ スレッダ
クライアント
下位ライブラリ
216
212
210
208
206
224
222
220
214
218
226
202
204a
204b
204c
204d
204e
204f

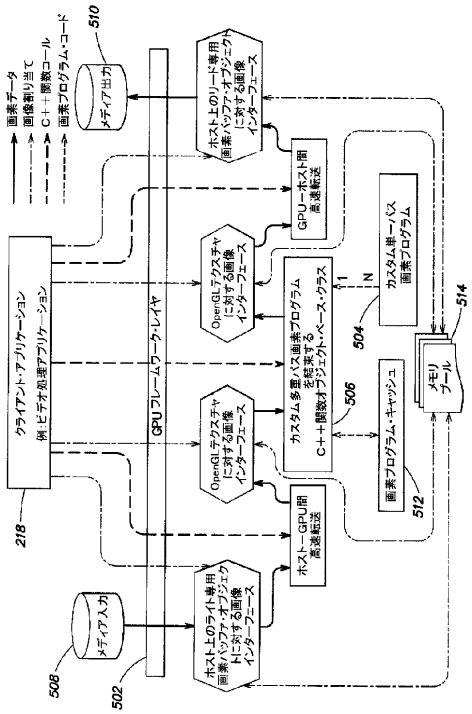
【図３】


CPL::IPまたはCPL::GP 処理ユニット
コンパイラ・インターフェース
処理ユニット
FXインターフェース
IL-GPUおよびGk-GPUを コールするGPUドメイン インプリメンテーション
ILおよびGkをコールする GPUドメイン・インプリメンテーション
IL-DLEをコールする Nitrisドメイン・インプリメンテーション
セル・プロセッサ ドメイン・インプリメンテーション
308
300
306
310a
310b
310c
310d
312a
312b
312c
312d
304
302

【図４】


画像処理システム
CPL
IL-GPU
Gk-GPU
GPGPU レイヤ
3D-レンダリング・レイヤ
3D-サーバ
コア画像
OpenGL
DirectX
Mac上のGPU
PC上のGPU
Mac上のGPU
PC上のGPU
402
216
204b
204e
404
406
414
416
408
412
410
424
422
420
418

【図５】


画素データ
画像割り当て
C++関数コール
画素プログラム・コード
510
メディア出力
ホスト上のリード専用
画素バッファ・オブジェクト
におする画像
インターフェース
GPU－ホスト間
高速転送
クライアント・アプリケーション
例：ビデオ処理アプリケーション
GPUフレームワーク・レイヤ
OpenGLテクスチャ
におする画像
インターフェース
カスタム多意バス画素プログラム
C++関数オブジェクト・ベース・クラス
504
1
N
カスタム単一パス
画素プログラム
514
メモリ
プール
506
画素プログラム・キャッシュ
512
218
メディア入力
ホスト－GPU間
高速転送
ホスト上のライト専用
画素バッファ・オブジェク
トにおする画像
インターフェース
508
502

フロントページの続き

(51)Int.Cl.	FI	テーマコード（参考）
	G06T 1/20 B	

(74)代理人 100096013
弁理士 富田 博行
(74)代理人 100120112
弁理士 中西 基晴
(72)発明者 マイケル・エイド
カナダ国ケベック エイチ３ピー ２エム１，ヴィル・モントリオール，ダンレイブン １１７４
(72)発明者 シャイレンドラ・マトゥール
カナダ国ケベック エイチ９ダブリュー ４エム７，ビーコンズフィールド，レッドファーン・プレイス ９
(72)発明者 ダニエル・ビュードリー
カナダ国ケベック エイチ１ティー ２エル２，モントリオール，シャトレーン ６７１０
(72)発明者 マシュー・ラマール
カナダ国ケベック エイチ４アール ２アール３，ヴィル・サンーローラン，デ・エベレスト １４３３
(72)発明者 レイ・タイス
アメリカ合衆国マサチューセッツ州０１８２１，ビレリカ，ブライアン・サークル ５

Ｆターム(参考) 5B050 BA08 BA09 CA02 EA19 EA27 EA30
5B057 CA01 CA08 CA12 CA13 CA16 CA17 CB01 CB08 CB12 CB13
CB16 CB17 CC04 CE06 CE08 CE17 CE18 CH02 CH14 CH16
5B069 AA16 BB06 KA00
5B080 CA01 CA03 FA02 GA00
5B376 BC16 BC25 DA18 DA24 FA25 GA01

【外国語明細書】
2010020755000001.pdf