(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2010-182408
(P2010-182408A)
(43)公開日　平成22年8月19日(2010.8.19)

(51)Int.Cl.		F I		テーマコード（参考）
G11B 27/10	(2006.01)	G11B 27/10	A	5D077

審査請求 有　請求項の数 9　ＯＬ　（全 32 頁）

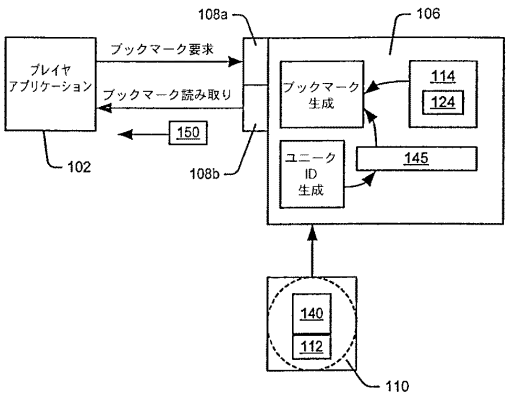
(21)出願番号　特願2010-56049 (P2010-56049)
(22)出願日　平成22年3月12日 (2010.3.12)
(62)分割の表示　特願2006-263196 (P2006-263196)の分割
原出願日　平成13年11月22日 (2001.11.22)
(31)優先権主張番号　09/721,266
(32)優先日　平成12年11月22日 (2000.11.22)
(33)優先権主張国　米国 (US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　グレン　エフ．エバンズ
アメリカ合衆国　９８０３４　ワシントン州　カークランド　ノースイースト　１３３　プレイス　７８３３

最終頁に続く

(54)【発明の名称】ＤＶＤプレイヤに対する改良プレイバック制御方法、システムおよびコンピュータ読取可能な媒体

(57)【要約】
【課題】ＤＶＤナビゲータプログラムを制御するためにプレイヤアプリケーションが使用でき、単純化された統一性のあるインターフェースを使用する方法を提供する。
【解決手段】マルチメディア・コンテンツ・プレイバック・プロセスに対するマルチメディア・ナビゲータ・プログラムのカレントステートを含むブックマークを、ＡＰＩに選択的に生成させ、ブックマークをマルチメディア・プレイヤ・アプリケーションに引渡すことによって、プレイヤアプリケーションがプレイバック期間中にロケーションにより正確に「ブックマーク」を付けて、選択したブックマーク付きのロケーションからプレイバックを、あとで再開できるようにする。
【選択図】図１３



【特許請求の範囲】
【請求項１】
少なくとも１つのアプリケーション・プログラミング・インターフェース（ＡＰＩ）を介して、マルチメディア・ナビゲータ・プログラムとのインターフェースとして動作するよう構成されているマルチメディア・プレイヤ・アプリケーションを含むシステムにおける方法であって、
マルチメディア・コンテンツ・プレイバック・プロセスに関係する前記マルチメディア・ナビゲータ・プログラムの現在のステートを含むブックマークを、選択的に前記ＡＰＩに生成させるステップであって、前記現在のステートは、プレイ、ストップ、ポーズ、リバース、早送り、スローモーションまたはアングルの少なくとも１つを含み、前記ブックマークは、ＤＶＤのＶＩＤＥＯ＿ＴＳディレクトリィ内のファイルヘッダおよび１または複数のファイルコンテンツに基づいて６４ビットＣＲＣを計算することにより生成されたマルチメディア・コンテンツに基づいた実質的に一意の識別子を含む、ステップと、
前記ブックマークを、前記マルチメディア・プレイヤ・アプリケーションに提供するステップと
を備えたことを特徴とする方法。
【請求項２】
前記ブックマークは、表示されるカレント・ビデオ・オブジェクトのアドレス、ループカウント、シャッフルヒストリ、カレントＤＶＤ再開情報、カレントＤＶＤ一般パラメータ（ＧＰＲＭ）情報、システムパラメータ（ＳＰＲＭ）情報、カレントドメイン情報、カレントフェーズ情報の少なくとも１つ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】
前記ブックマークは、該ブックマークのバージョン情報と保全情報とをさらに含むことを特徴とする請求項１に記載の方法。
【請求項４】
少なくとも１つのアプリケーション・プログラミング・インターフェース（ＡＰＩ）を介して、マルチメディア・ナビゲータ・プログラムとのインターフェースとして動作するよう構成されているマルチメディア・プレイヤ・アプリケーションを含むシステムにおけるコンピュータ読取可能な記録媒体であって、
マルチメディア・コンテンツ・プレイバック・プロセスに関係する前記マルチメディア・ナビゲータ・プログラムの現在のステートを含むブックマークを、選択的に前記ＡＰＩに生成させるステップであって、前記現在のステートは、プレイ、ストップ、ポーズ、リバース、早送り、スローモーションまたはアングルの少なくとも１つを含み、前記ブックマークは、ＤＶＤのＶＩＤＥＯ＿ＴＳディレクトリィ内のファイルヘッダおよび１または複数のファイルコンテンツに基づいて６４ビットＣＲＣを計算することにより生成されたマルチメディア・コンテンツに基づいた実質的に一意の識別子を含む、ステップと、
前記ブックマークを、前記マルチメディア・プレイヤ・アプリケーションに提供するステップと
を実行するためのコンピュータ実行可能命令を有することを特徴とするコンピュータ読取可能な記録媒体。
【請求項５】
前記ブックマークは、表示されるカレント・ビデオ・オブジェクトのアドレス、ループカウント、シャッフルヒストリ、カレントＤＶＤ再開情報、カレントＤＶＤ一般パラメータ（ＧＰＲＭ）情報、システムパラメータ（ＳＰＲＭ）情報、カレントドメイン情報、カレントフェーズ情報の少なくとも１つを含むことを特徴とする請求項４に記載のコンピュータ読取可能な記録媒体。
【請求項６】
前記ブックマークは、該ブックマークのバージョン情報と保全情報とをさらに含むことを特徴とする請求項４に記載のコンピュータ読取可能な記録媒体。
【請求項７】

メモリと、
該メモリに結合されたプロセッサと、
該プロセッサにより動作するマルチメディア・プレイヤ・アプリケーションと、
該マルチメディア・プレイヤ・アプリケーションとマルチメディア・ナビゲータ・プログラムとのインターフェースとして動作するように構成された少なくとも１つのアプリケーション・プログラミング・インターフェース（ＡＰＩ）であって、前記マルチメディア・プレイヤ・アプリケーションは、マルチメディア・コンテンツ・プレイバック・プロセスに関係する前記マルチメディア・ナビゲータ・プログラムの現在のステートを含むブックマークを、選択的に前記ＡＰＩに生成させ、前記現在のステートは、プレイ、ストップ、ポーズ、リバース、早送り、スローモーションまたはアングルの少なくとも１つを含み、前記ブックマークは、ＤＶＤのＶＩＤＥＯ＿ＴＳディレクトリィ内のファイルヘッダおよび１または複数のファイルコンテンツに基づいて６４ビットＣＲＣを計算することにより生成されたマルチメディア・コンテンツに基づいた実質的に一意の識別子を含む、ＡＰＩと
を備えたことを特徴とするシステム。
【請求項８】
前記ブックマークは、表示されるカレント・ビデオ・オブジェクトのアドレス、ループカウント、シャッフルヒストリ、カレントＤＶＤ再開情報、カレントＤＶＤ一般パラメータ（ＧＰＲＭ）情報、システムパラメータ（ＳＰＲＭ）情報、カレントドメイン情報、カレントフェーズ情報の少なくとも１つを含むことを特徴とする請求項７に記載のシステム。
【請求項９】
前記ブックマークは、該ブックマークのバージョン情報と保全情報とをさらに含むことを特徴とする請求項７に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般的には、コンピュータおよび類似装置に関し、より詳細には、あるアプリケーション・プログラミング・インターフェース（application programming interface：ＡＰＩ）によってマルチメディアプレイヤアプリケーションと汎用メディアコンテンツナビゲータプログラムとの間で使用される改良プレイバック制御方法および構成に関する。
【背景技術】
【０００２】
デジタル・バーサタイル・ディスク（digital versatile disc：ＤＶＤ）は、ＤＶＤ規格に定義されているように、３つの論理ユニット（logical unit）から構成されている。第１論理ユニットは、ユーザとのインターフェースとなって、ユーザのコマンドを第２論理ユニットに中継（リレー）するＤＶＤプレイヤアプリケーションである。第２論理ユニットは、ＤＶＤ上のデータを読み取って、解釈し、ビデオとオーディオのどのセグメントを処理するかを、ユーザのコマンドに基づいて制御するＤＶＤナビゲータである。第３論理ユニットは、ＤＶＤから読み取ったデータを伸張（圧縮復元）し、対応するオーディオ、ビデオおよびサブピクチャストリームのうち、該当するものを１つまたは２つ以上のレンダラ（renderer）に提示するＤＶＤプレゼンテーション層である。
【０００３】
これらの論理ユニットは、ハードウェアでも、ソフトウェアでも、あるいはその組み合わせでも実装（インプリメント）することが可能になっている。１つの例として、いくつかの実装では、ＤＶＤプレイヤは、ユーザに表示されるグラフィカル・ユーザ・インターフェース（graphical user interface：ＧＵＩ）によって移植され、ユーザはそのＧＵＩから、例えば、マウスなどのポインティング選択入力装置を使用して、ＤＶＤのプレイバックなどを選択的に制御できるようにしている。これは、システム開発者にとっては、通

常、かなり簡単な作業であり、カストマイズを行うことも容易化されている。
【０００４】
これに対して、ＤＶＤナビゲータを実装する作業は、もっと複雑化する傾向がある。このことが特に当てはまるケースとして、ＤＶＤ情報をプレゼンテーションなどに組み入れようとするアプリケーションがある。そこでは、各開発者エンティティは、そのＤＶＤを読み取って、解釈し、ＤＶＤプレゼンテーション層に置かれているデコーダ機構とのインターフェースとなる機構を用意する必要がある。ＤＶＤプレゼンテーション層内のデコーダ機構は、今後は、サードパーティのプロダクトとして提供されることが予想される。従って、ナビゲータは、多数の異なるデコーダ機構とのインターフェースとなる必要があるため、ＤＶＤナビゲータを作成する作業はますます複雑化することになる。
【発明の概要】
【発明が解決しようとする課題】
【０００５】
従って、ＤＶＤナビゲータプログラムを制御するためにプレイヤアプリケーションが使用でき、強力であるが単純化された、統一性のあるインターフェースが必要である。
【課題を解決するための手段】
【０００６】
アプリケーション開発者に課される潜在的作業負担を認識して、本願発明の特許出願人は、オペレーティングシステムおよびユーザの環境をさらに強化しようとして、汎用ナビゲータコンポーネント（generic navigator component）を開発した。この汎用ナビゲータコンポーネントによれば、標準規格に準拠するＤＶＤナビゲータがＷｉｎｄｏｗｓ（登録商標）の一部として用意され、アプリケーション開発者は、上記のような繰り返し起こり得る、困難な作業を行わないで済むようにしている。この汎用ナビゲータコンポーネントによれば、２つのアプリケーション・プログラミング・インターフェース（ＡＰＩ）が公開されており、これらが結合されて、強力で、しかも単純化され、統一されたインターフェースを提供し、プレイヤアプリケーションはそのインターフェースを使用して、ＤＶＤナビゲータを制御することを可能にしている。これらのＡＰＩは、基礎となるＤＶＤナビゲータのフレキシビリティと有用性に、さらに影響を及ぼすように設計されている。
【０００７】
本発明のいくつかの態様によれば、この汎用ナビゲータコンポーネントのパフォーマンスをさらに拡張するための強化機能が開発されている。ここで重要なことは、プレイバックを開始かつ停止するためのユーザおよびプレイヤアプリケーションの環境を改良する必要があることである。現行ナビゲータによって、ディスク上の前のロケーションにジャンプすることは煩わしく、信頼性に欠けていることがよくある。例えば、ユーザが特定のロケーションからムービ（動画）を見続けたいとき、ユーザはそのロケーションを覚えておき、手動操作でその個所に戻るようにナビゲートしなければならない。従って、プレイヤアプリケーションが、プレイバック期間中に、ロケーションにより正確に「ブックマーク（bookmark）」を付けておき、ブックマークを付けたロケーションを選択したプレイバックを、あとで再開できるようにする機構が用意されていると好都合である。
【０００８】
本発明の種々の方法および構成の理解を容易にするために、以下では、添付図面を参照して詳しく説明することにする。
【図面の簡単な説明】
【０００９】
【図１】例示ＤＶＤプレイヤ装置を示すブロック図である。
【図２】図１のＤＶＤプレイヤ装置で使用するのに適しているコンピュータ環境を示すブロック図である。
【図３】ＤＶＤプレイヤアプリケーションと汎用ナビゲータプログラムとの間の第１同期モードを示すブロック図である。
【図４】ＤＶＤプレイヤアプリケーションと汎用ナビゲータプログラムとの間の第２同期

モードを示すブロック図である。
【図５】ＤＶＤプレイヤアプリケーションと汎用ナビゲータプログラムとの間の第３同期モードを示すブロック図である。
【図６】ＤＶＤプレイヤアプリケーションと汎用ナビゲータプログラムとの間の第４同期モードを示すブロック図である。
【図７】ＤＶＤプレイヤアプリケーションと汎用ナビゲータプログラムとの間の非ブロッキング同期モードとブロッキング同期モードを示すブロック図である。
【図８】ＤＶＤプレイヤアプリケーションと汎用ナビゲータプログラムとの間の非ブロッキング同期モードとブロッキング同期モードを示すブロック図である。
【図９】プレイヤアプリケーションと、メディアコンテンツに関係するプログラムとの間の例示読み／書き通信機能を示すブロック図である。
【図１０】メディアコンテンツに対する制限付き／親制御に関連するデュアルブランチのプレイバック決定ポイントを示すライン図である。
【図１１】メディアコンテンツに対する制限付き／親制御に関連するマルチブランチのプレイバック決定ポイントを示すライン図である。
【図１２】プレイヤアプリケーションに用意されているコードの使用によってメディアデータへのアクセスを制御するための例示的方法を示すブロック図である。
【図１３】例示メディアコンテンツブックマーキング機能を示すブロック図である。
【図１４】メディアソースを表す実質的に唯一の識別子を生成するための例示的方法を示す図である。
【発明を実施するための形態】
【００１０】
以下に示す例示の方法および構成では、ＤＶＤプレイヤアプリケーションから見えるＡＰＩ（application programming interface：アプリケーション・プログラミング・インターフェース）をもつ汎用ナビゲータに関連する、いくつかの強化機能と特徴が説明されている。これらは、ＤＶＤナビゲータおよびＤＶＤ２　ＡＰＩと呼ばれている。なお、以下の説明の大部分は、Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムが稼動しているＰＣが中心になっているが、種々の方法および構成は、他のオペレーティングシステム、装置などにも適用可能であることはもちろんである。さらに、ここで使用されているＤＶＤという用語には、他のメディアフォーマットも含まれることはもちろんである。従って、ＤＶＤコンテンツ自体のソースとしては、ハードドライブ、コンパクトディスク、ネットワークなどが可能である。
【００１１】
以下で説明するように、ＤＶＤナビゲータおよび／またはＤＶＤ２　ＡＰＩによって、プレイヤアプリケーションに、ＤＶＤコンテンツのプレイバックを会話型で制御することを可能にする。ＤＶＤ　ＡＰＩは２つのインターフェースから構成されている。第１のインターフェースは“ＩＤｖｄｌｎｆｏ２”と名付けられている。第２のインターフェースは“ＩＤｖｄＣｏｎｔｒｏｌ２”と名付けられている。プレイヤアプリケーションは、ＩＤｖｄｌｎｆｏ２インターフェースを使用すると、ＤＶＤナビゲータのカレントステート（現在の状態）をクエリ（照会）することができ、ＩＤｖｄＣｏｎｔｒｏｌ２インターフェースを使用すると、プレイバックの制御を向上することおよび／またはＤＶＤナビゲータのステートを変更することができる。
【００１２】
ＤＶＤ２　ＡＰＩは、唯一のおよび斬新な複数の特徴を提供する。例えば、スレッドに基づく同期化メソッドをリアルタイムプレイバックに提供し、プレイバック制御機構を提供することによってやりとり（会話型）の程度を決定し、通信機構をプレイヤアプリケーションとディスクプログラムとの間に提供し、タイムレンジ（時間範囲）のプレイ実行を支援し、機構を親レベル（parental level）の要求を調整かつ処理することに提供し、最低親レベル（minimal parental level）を決定することによって制限されたコンテンツセグメントをプレイ実行することに機構を提供し、および唯一のディスク識別子のアルゴリ

ズムを提供しさらにＤＶＤコンテンツ内のあらゆるロケーションのブックマーク付けを支援する。

【００１３】

以上のことを念頭に置いて、図１を参照して説明すると、図１は、例示のＤＶＤプレイヤ１００を示す。プレイヤ１００は少なくとも１つのプレイヤアプリケーション１０２を含み、プレイヤアプリケーション１０２は、ユーザインターフェース（Ｕ／Ｉ）１０４をユーザに提示するように構成されている。Ｕ／Ｉ１０４によって、ユーザは、ＤＶＤコンテンツ１１０のプレイバックに関してプレイヤアプリケーション１０２に指示することが可能になっている。

【００１４】

図に示すように、プレイヤアプリケーション１０２はＤＶＤ２　ＡＰＩ１０８ａと１０８ｂとを備え、それぞれによって、ユーザの要求を伝達し、フィードバック情報を受け取るようになっている。ナビゲータ１０６内の関数へのアクセスは、ＤＶＤ２　ＡＰＩ１０８ａ－ｂによって行われる。ナビゲータ１０６はＤＶＤコンテンツ１１０とやりとりし、ＤＶＤコンテンツ１１０は、メディア情報のほかに、プログラム１１２を含む。プログラム１１２は、残りのコンテンツと関連付けられたメニュー、ジャンプなどを定義する。ナビゲータ１０６は、プレイバックプロセスに関連付けられたステート１１４を含む。ここで、ステート１１４では、例えば、カレントユーザオペレーション（例えば、ストップ（停止）、ポーズ（休止）、リバース（逆送り）、高速前送り、スローモーション、アングルなど）は、ＤＶＤコンテンツ（例えば、チャプタ、時間、フレーム）内のカレントロケーションおよび最近のジャンプ／ＵＯＰを記録しておくことができるよう他のある種のレジスタと一緒に格納される。

【００１５】

ナビゲータ１０６の出力は、符号化ビデオストリーム、符号化オーディオストリーム、サブピクチャストリームのうち、該当するものを含んでいる。これらの出力はデコーダ（復号器）１１６に入力され、このデコーダは、符号化データを復号化（デコード）（解読と伸張）し、対応するストリームをビデオレンダラ（video renderer）１１８またはオーディオレンダラ（audio renderer）１２０のうち、該当する方に出力するように構成されている。レンダラ１１８は、ビデオ情報が、例えば、ビデオモニタによってユーザに表示されるようにする。レンダラ１２０は、オーディオ情報が、例えば、１つまたは２つ以上のスピーカからリスナのために再生されるようにする。

【００１６】

次に、図２を参照すると、図２は、図１の構成で使用するのに適している例示コンピューティングシステム２００を示すブロック図である。

【００１７】

コンピューティングシステム２００は、この例では、パーソナルコンピュータ（personal computer：ＰＣ）の形体になっているが、他の例では、コンピューティングシステムは、専用サーバ、特殊目的装置、アプライアンス、ハンドヘルドコンピューティング装置、モバイルテレホン装置、ページャ装置などの形体にすることも可能である。

【００１８】

図示のように、コンピューティングシステム２００は、処理ユニット２２１、システムメモリ２２２、およびシステムメモリ２２３を含んでいる。システムバス２２３は、システムメモリ２２２と処理ユニット２２１を含む種々システムコンポーネントを１つにリンクしている。システムバス２２３は、数種類のバス構造のいずれかにすることが可能であり、その中には、種々のバスアーキテクチャのいずれかを採用しているメモリバスまたはメモリコントローラ、ペリフェラル（周辺）バス、およびローカルバスが含まれている。システムメモリ２２２の代表例としては、リードオンリメモリ（read only memory：ＲＯＭ）２２４とランダムアクセスメモリ（random access memory：ＲＡＭ）２２５がある。基本入出力システム（basic input/output system：ＢＩＯＳ）２２６は、スタートアップ時のときのように、コンピューティングシステム２００内のエレメント間で情報を転送

するのを支援する基本ルーチンで構成され、ＲＯＭ２２４に格納されている。コンピューティングシステム２００は、さらに、ハードディスク（図示せず）との間で読み書きを行うハードディスクドライブ２２７、取り外し可能磁気ディスク１６０との間で読み書きを行う磁気ディスクドライブ２２８、およびＣＤ－ＲＯＭや他の光メディアなどの、取り外し可能光ディスク２３１との間で読み書きを行う光ディスクドライブ２３０を装備している。ハードディスクドライブ２２７、磁気ディスクドライブ２２８、および光ディスクドライブ２３０は、それぞれ、ハードディスク・ドライブ・インターフェース２３２、磁気ディスク・ドライブ・インターフェース２３３、および光ドライブインターフェース２３４を介してシステムバス２２３に接続されている。これらのドライブおよびそれぞれに関連するコンピュータ読取可能な媒体は不揮発性ストレージ（nonvolatile storage）として、コンピューティングシステム２００のためのコンピュータ読取可能な命令、データ構造、コンピュータプログラムおよび他のデータを格納している。

【００１９】

いくつかのコンピュータプログラムは、ハードディスク、磁気ディスク２２９、光ディスク２３１、ＲＯＭ２２４またはＲＡＭ２２５に格納可能であり、それらプログラムの中には、オペレーティングシステム２３５、１つまたは２つ以上のアプリケーションプログラム２３６、他のプログラム２３７、およびプログラムデータ２３８が含まれている。

【００２０】

ユーザは、キーボード２４０やポインティングデバイス２４２（マウスなど）などの、入力装置によってコマンドと情報をコンピューティングシステム２００に入力することができる。カメラ／マイクロホン２５５またはリアルタイムデータ２５６を取り込み（キャプチャ）、あるいは出力できる機能を備えた他の類似メディアデバイスを、コンピューティングシステム２００への入力装置とすることも可能である。リアルタイムデータ２５６は、該当のインターフェース２５７からコンピューティングシステム２００に入力することができる。インターフェース２５７をシステムバス２２３に接続することによって、リアルタイムデータ２５６を、ＲＡＭ２２５または他のデータストレージ装置の１つに格納し、いずれにせよ処理することを可能にする。

【００２１】

図示のように、モニタ２４７や他の種類のディスプレイ装置も、ビデオアダプタ２４８のような、インターフェースを介してシステムバス２２３に接続されている。モニタのほかに、コンピューティングシステム２００は、スピーカやプリンタのような、他の周辺出力装置（図示せず）を装備することもできる。

【００２２】

コンピューティングシステム２００は、リモートコンピュータ２４９のような、１つまたは２つ以上のリモートコンピュータとの論理コネクションを使用するネットワーキング環境で動作することができる。リモートコンピュータ２４９は、別のパーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピアデバイスまたは他の共通ネットワークノードにすることが可能であり、図２にはメモリストレージ装置２５０だけが示されているが、コンピューティングシステム２００に関連して上述したエレメントの多くまたは全部を含んでいるのが代表的である。

【００２３】

図２に示す論理コネクションとしては、ローカル・エリア・ネットワーク（ＬＡＮ）２５１と広域ネットワーク（ＷＡＮ）２５２がある。このようなネットワーキング環境は、オフィス、企業内（enterprise-wide）コンピュータネットワーク、イントラネット（intranet）、およびインターネット（the Internet）で普及している。

【００２４】

ＬＡＮネットワーキング環境で使用されると、コンピューティングシステム２００は、ネットワークインターフェースまたはアダプタ２５３によってローカルネットワーク２５１に接続されている。ＷＡＮネットワーキング環境で使用されるときは、コンピューティングシステム２００は、インターネットなどの広域ネットワーク２５２上のコミュニケー

ションを確立するためのモデム２５４や他の手段を装備しているのが代表的である。モデム２５４は内蔵型と外付け型があり、どちらもシリアルポートインターフェース２４６を介してシステムバス２２３に接続されている。

【００２５】

ネットワーキング環境では、コンピューティングシステム２００に関連して上述したコンピュータプログラムまたはその一部は、リモートメモリストレージ装置に格納しておくことができる。以上から理解されるように、図示のネットワークコネクションは例示であり、コンピュータ間のコミュニケーションリンクを確立する他の手段を使用することも可能である。

【００２６】

ＤＶＤ２　ＡＰＩ１０８ａ－ｂによれば、アプリケーション作成が単純化され、機能が補強され、ＤＶＤプレイヤアプリケーション開発に共通している多数の、困難な同期化問題が解決される。基本的に、共通ＤＶＤ　ＡＰＩによると、スタンドアロンのＤＶＤプレイヤアプリケーションだけに役立つＤＶＤソリューション、すなわち、企業所有（proprietary）のシングルユースモノリシックＤＶＤソリューションに頼らないで済むことになる。また、このＡＰＩによると、種々のアプリケーション（プレゼンテーションプログラム、ＤＶＤプレイヤ、ゲーム、あるいは会話型学習プログラムなど）は、どのＤＶＤデコーダやＤＶＤハードウェアサポートがユーザのシステムに導入されているかを知らなくても、ＤＶＤサポートを追加することが可能になっている。歴史的に、カスタムＤＶＤソリューションは非常にハードウェアに依存する傾向があり、ユーザのためのアップグレードオプションが制限されている。

【００２７】

以下で詳しく説明するように、ＤＶＤ２　ＡＰＩ１０８ａ－ｂでは、フレキシブルな同期機構が追加されているため、ＤＶＤナビゲータ１０６に対して行われた要求の完了ステータスを、アプリケーションに知らせることが可能になっている。新規のコマンド完了通知によれば、アプリケーションは他のタスクの実行と同時に、以前の要求のステータスを知ることが可能になっている。従来のＤＶＤ　ＡＰＩでは、要求が完了するまでアプリケーションをブロックさせるか、あるいはどの通知もアプリケーションに送信しないことが前提になっていた。現在では、アプリケーションは、同期オブジェクトを受け取ったとき、そこで待ちを続けてもよいか、あるいは完了イベントの通知を受けるか、のどちらかが選択できるようになっている。

【００２８】

また、同期機構からは、要求が正常に実行されたかどうかを示す、要求のステータスが戻されるか、あるいは要求の実行が失敗した理由（エラーコード）が戻されるようにもなっている。従来のＤＶＤ　ＡＰＩでは、要求が正常に実行されるように見えても、ＤＶＤナビゲータ１０６が実際に要求の処理を開始したとき、ステートが変わったために、あとで失敗することがあった。その時点では、エラー通知をプレイヤアプリケーション１０２に戻すように伝播して行く方法がなかった。また、新規の機構では、どの要求も、ディスクのプログラム１１２によって、あるいは以後のユーザアクションによってキャンセルまたはオーバライドされたとき、そのことがプレイヤアプリケーション１０２に通知されるようになっている。

【００２９】

現行ＤＶＤ　ＡＰＩでは、予め定義されたビヘイビア（predefined behavior）が使用され、コマンドがカレントディスプレイとどのようにやりとりするかは、これらのビヘイビアによって決まるようになっている。あるプレイヤアプリケーションが新しい要求を出すと、そのプレイヤアプリケーションに優先権が与えられ、プレイ実行されているコンテンツ（ビデオまたはオーディオ）があれば、そのコンテンツはキャンセルされる。別の方法として、ＡＰＩのセマンティックスによれば、カレントプレゼンテーションが完了してから新しいコンテンツが表示されるようにし、ユーザを強制的に待たせてから別のアクションを要求できるようにしている。ＤＶＤプレイヤやゲームのような会話型アプリケーシ

ョンでは、一番目のビヘイビア（瞬時効果）が要求されることがあるが、スライドショウ（slideshow）などの、他のアプリケーションでは、２番目のビヘイビア（カレントプレゼンテーションの完了）が要求されることがある。これらの２オプションは相互に排他的であるため、予め定義されたＡＰＩのセマンティックスでは、両方を受け入れることが不可能になっている。ＤＶＤ２　ＡＰＩ１０８ａ－ｂによれば、プレイヤアプリケーション１０２は、望みのビヘイビアをフラグで指示できるだけでなく、同期機構とどのようにやりとりするかも指示できるようになっている。

【００３０】

ＤＶＤナビゲータ１０６は、実行ステート１１４（メモリレジスタ１２４の集まりの形態になっている（図９参照））を使用する仮想ＣＰＵをシミュレートする構成になっている。従来のＤＶＤ　ＡＰＩでは、アプリケーションはレジスタの内容を読み取ることが可能であった。ＤＶＤ２　ＡＰＩ１０８によれば、プレイヤアプリケーション１０２は、メモリレジスタの内容を変更することも可能になっている。この読み書き結合機能（combined read/write functionality）によると、プレイヤアプリケーション１０２は、図９に示すように、基本的にプログラム１１２と「やりとり」することが可能になる。

【００３１】

この読み書きメソッドは、同期化にも使用できるような働きをする。１つの例として、読み書き機能を使用すると、プレイヤアプリケーション１０２は、「制御によるアンロッキング（controlled unlocking）」、つまり、ＤＶＤ内容の全部または一部への限定アクセスを実現することができる。制御によるアンロッキングによると、プレイヤアプリケーション１０２が特定のメモリレジスタをセットするまで、ユーザがディスクの一部を見ることを制限することができる。プレイヤアプリケーション１０２は、コンテンツの作成者、ユーザ、別のプログラム、Ｗｅｂサイトなどの、だれからでもこの情報を受け取ることができる。例えば、図１２は、コードがプレイヤアプリケーション１０２によってレジスタ１２４にどのように書き込まれ、プログラム１１２によってどのように読み取られるかを示している。そのコードが正しければ、ＤＶＤコンテンツ１１０の一部１３０をプレイバックすることが可能になる。

【００３２】

ある種の実装では、ＤＶＤ２　ＡＰＩ１０８ａ－ｂは、ＤＶＤ規格付属書Ｊで提案されている、単純化されたネーミング構成（simplified naming scheme）を潜在的ユーザオペレーションのために取り入れている。ＤＶＤ２　ＡＰＩでは、使用されているＤＶＤ専門用語が少なく、ネーミング構成が直観的であることを特徴としている。ＤＶＤ規格に提案されているユーザオペレーション名は不明確であるため、アプリケーションによって誤用されるか、あるいは十分に活用されない可能性がある。現在では、抽象レベルではなく、その使い方が名前から分かるようにしている。また、時間コードは、扱いづらいＢＣＤコーディングによるのではなく、単純な整数フォーマットで戻されるようにしている。

【００３３】

従来のいくつかのＤＶＤ　ＡＰＩでは、ブランチがいつも失敗すると、ＤＶＤナビゲータがエラーイベントで知らせるようにしているため、最低親レベルのブランチングを正しく処理することができなかった（図１０参照）。そのため、プレイヤアプリケーションは親レベルを大きくしてから、初めからムービをリスタートする必要があった。ブランチが失敗すると、プレイヤアプリケーションは、プレイバックを中止してＳＴＯＰドメインに入ってから、親レベルを変更する必要があった。プレイヤアプリケーションが継続できるのは、ムービをリスタートすることによってのみである。

【００３４】

これに反して、ＤＶＤ２　ＡＰＩ１０８ａ－ｂでは、ナビゲータ１０６をポーズし、親レベル増大要求に応答する機会をプレイヤアプリケーション１０２に与えてからナビゲータ１０６が継続するようにするモードが用意されている。この増大要求が許可されると、ユーザがムービを初めからスタートしなくても、プレイバックが継続される。ＤＶＤ規格では、要求が成功したか、失敗したかをナビゲータが知るまで、ナビゲータをポーズさせ

ることだけが規定されている。この規格には、このタスクを実行する機構は記載されておらず、ナビゲータがプレイヤに組み込まれている「一時的親レベル変更機能をコールする」ことが提案されている（４．６．４．１　Ｖ１４－１９７）。

【００３５】

ＤＶＤ規格には、ユーザがマルチセグメント親レベルブランチをプレイ実行できるようにする機構も記載されていない（例えば、図１１参照）。そのために、従来のＤＶＤ　ＡＰＩでは、ブランチがカレントユーザレベルで許されないとき、ユーザがマルチセグメント（またはマルチブランチ）親レベルブランチをプレイ実行することを可能にする機構は用意されていなかった。従来、ナビゲータは、カレント親レベルではブランチが用意されていなかったために、プレイバックが中止したことをアプリケーションに通知するだけであった。

【００３６】

これに反して、ナビゲータ１０６とＤＶＤ２　ＡＰＩ１０８ａ－ｂでは、ブロックをプレイ実行するために必要な最低レベルが計算され、この値が「プレイバック中止」通知と一緒に戻されるようにしている。そのため、アプリケーションは、ＤＶＤコンテンツ１１０のプレイ実行を継続するために要求される、必要な親レベルをユーザに通知できるようになった。従って、ユーザは、試行錯誤で必要レベルを推量する必要がなく、試行のたびにムービをリスタートする必要がなくなった。

【００３７】

さらに、ＤＶＤ２　ＡＰＩ１０８ａ－ｂでは、ＤＶＤ規格付属書Ｊと従来のＤＶＤ　ＡＰＩの機能が拡張されている。ＤＶＤ規格付属書Ｊでは、実行されるアクションだけが規定されている。また、プレイヤアプリケーション１０２が、ディスクまたはＤＶＤナビゲータのステート１１４に関する情報をどのようにして探し出すかは、規定されていない。本発明では、新規のディスクおよびナビゲーション・ステート・クエリ（照会）機能が提供されている。

【００３８】

従来のＤＶＤ　ＡＰＩとは異なり、ＤＶＤ２　ＡＰＩ１０８ａ－ｂでは、アプリケーション作成者は、準備状態に置かれたＤＶＤ規格のコピーがまだなくても、ＤＶＤを使用できるようになっている（例えば、ＡＰＩから戻されるデータの記述が不完全であるため）。テキスト情報、タイトル属性、オーディオ属性およびサブピクチャ属性を得るためにメソッドにより戻されるデータは、アプリケーション開発者が新規のＡＰＩおよび関連ドキュメンテーションから必要情報を得られるようにドキュメント化されている。

【００３９】

また、ＤＶＤ　ＡＰＩ１０８ａ－ｂによれば、アプリケーションは、カレント・タイトル・インデックスだけでなく、任意のタイトルインデックスの属性をクエリすることも可能になっている。また、ＤＶＤ　ＡＰＩ１０８ａ－ｂからは、オーディオストリームのカラオケ情報も戻されるので、インテリジェントなカラオケアプリケーションを実現することが可能になっている。さらに、ＤＶＤ　ＡＰＩ１０８ａ－ｂからは、デコーダ１１６の機能も戻されるので、アプリケーションは、構成オプションをユーザに提示すること（両方向のフレームステッピング、スムーズな巻き戻しや高速前送り、などのような）、あるいはユーザインターフェースをインテリジェントに変更することが可能になっている。新規な制御機能も用意されている。例えば、ＤＶＤ　ＡＰＩ１０８ａ－ｂによれば、プレイヤアプリケーション１０２は、チャプタの範囲または時間の範囲をプレイ実行し、（相対的ボタンだけでなく）特定のメニューボタンを選択することができ、ユーザはマウスロケーションを使用してボタンを選択することができる。さらに、ブックマークオブジェクトを取得／設定すること、および計算されたカレント・ユニーク・ディスクＩＤをクエリできることもサポートされている。

【００４０】

ＤＶＤ　ＡＰＩ１０８ａ－ｂの同期機構および関連ナビゲータ１０６を、アプリケーションと共に理解しやすくするために、以下のセクションでは、種々の例示動作モードが検

討され、利点と欠点のいくつかが指摘されている。基本的には、４つの動作モードと、これらに変形を加えた種々のモードがある。最初の４モードは図３乃至図６に示されている。４モードの各々は、本発明による種々の方法および構成で支援することが可能になっている。

【００４１】

「ドントケア（don't care）」モードまたはモデルは図３に示されており、そこでは、プレイヤアプリケーション１０２は、結果があるとき、どのような結果であるかおよび／または要求がいつ完了するかを気にすることなく、要求をナビゲータ１０６に送信するようになっている。例としては、ロケーションへのジャンプ要求、メニュー表示要求などがある。ここでは、プレイヤアプリケーションは、基本的に、要求したオペレーションが完了していることを想定している。

【００４２】

図４には、イベントモードまたはモデルが示されている。ここでは、プレイヤアプリケーション１０２は、ナビゲータから送られる汎用イベント（generic event）が起こったとき（要求が完了したとき）通知されるようになっている。このモデルの１つの欠点は、プレイヤアプリケーション１０２が２つ以上の要求を行っている可能性があるため、イベントを別々に通知できないことである。

【００４３】

上記を改良したものが図５に示されている。そこでは、イベントが起こったとき、プレイヤアプリケーション１０２に通知を行うのではなく、ナビゲータ１０６はオブジェクトを生成し、プレイヤアプリケーション１０２はそのオブジェクトを使用して、要求のステータスをトラッキング（追跡）できるようにしている。このようにすると、プレイヤアプリケーション１０２は、インスタンストラッキング（instance tracking）を行うことが可能になる。

【００４４】

さらに別の改良では、図６に示すように、ナビゲータ１０６は、トラッキングのために使用できるオブジェクトのほかに、以後のイベントも生成することが可能になっている。このようにすると、プレイヤアプリケーション１０２は、オブジェクトを使用してイベントを別々に通知することが可能になる。従って、このモデルでは、複数のインスタンストラッキングがサポートされている。

【００４５】

これらの種々モードとＤＶＤ　ＡＰＩ１０８ａ－ｂの詳細を説明する前に、ブロッキングのみ（blocking-only）ＡＰＩまたは非ブロッキングのみ（non-blocking-only）ＡＰＩの欠点について説明することにする。１つの改良形態は図７に示されている。そこでは、プレイヤアプリケーション１０２は、要求をナビゲータ１０６に送信している（当然に、ＤＶＤ　ＡＰＩ１０８ａを経由して）。プレイヤアプリケーション１０２は、ナビゲータ１０６から結果メッセージが送られてくるのを待たされている。このモデルの１つの欠点は、プレイヤアプリケーション１０２が待っている間に、Ｕ／Ｉ１０４が「フリーズ（freeze）」されるおそれがあることである。

【００４６】

Ｕ／Ｉ１０４のフリーズ問題を解決する１つの方法は、図８に示すようなワーカプログラム（worker program）を用意することである。そこでは、ワーカプログラムは要求を受け取ると、その要求をナビゲータ１０６に転送した後、結果メッセージが送られるのを待たされている。ワーカプログラムが結果メッセージを受け取ると、結果メッセージはプレイヤアプリケーション１０２に転送される。これによりＵ／Ｉ１０４はフリーズから解放されるが、同時に動作している複数のワーカを管理することが困難になっている。

【００４７】

これに対して、非ブロッキングＡＰＩは、「ドントケア」モードと同じ働きをしている。オペレーションのステータスまたは結果に関する直接的フィードバックは得られない。プレイバックが変更（時間変更、メニュー変更など）されたことによるステータスは、ア

プリケーションが推量しなければならない。しかし、ディスクコンテンツと構造は変化するので、このアプローチは非常に信頼性に乏しく、エラーが起こりやすくなっている。このことを念頭に置いて、以下のセクションでは、ＤＶＤ　ＡＰＩ１０８ａ－ｂの使い方に至るまで詳細に説明されている。

【００４８】

従来のＤＶＤ　ＡＰＩにおけるＩＤＶＤＣｏｎｔｒｏｌメソッドは、すべてがアプリケーションと非同期に実行されている（非ブロッキングのみモード）。従って、アプリケーション１０２がメソッドをコールすると、ナビゲータ１０６は予備的検証を行い、その直後に結果を戻している。しかし、その間に、ＤＶＤナビゲータのステートが変わっている可能性があるため、ＤＶＤナビゲータが実際にコマンドの実行を始めたとき要求が失敗するおそれがある。

【００４９】

１つのソリューションは、すべての要求が完了するまでメソッドがリターンしないことを保証するようにＤＶＤ　ＡＰＩのセマンティックスを変更することである。しかし、非同期のビヘイビアを残しておくためには、アプリケーションはＤＶＤ　ＡＰＩコールを管理するために別々の実行パス（例えば、ヘルパスレッド（helper threads）を作成しなければならない（これは、ブロッキングのみモデルで上述した通りである）。マルチスレッドプログラミングモデルは、常にアプリケーション開発、特に単純なスクリプト可能インターフェースを複雑化している。

【００５０】

従って、この問題を解決するために、ＤＶＤ２　ＡＰＩ１０８ａ－ｂでは、同期コマンドオブジェクトが作成されている。このコマンドオブジェクトによると、アプリケーションを同期させ、コマンドのステータスをアプリケーションに知らせることができる。各ＡＰＩメソッドは、２つのエキストラ引数をもつように拡張されている。ＤＶＤ２　ＡＰＩコマンドの一般形式は次のようになっている。

【００５１】

【数１】

$$\mathrm{HRESULT\ IDVDControl2::Command(\ arguments,\ dwFlags,\ IDvdCmd**\ ppObj)}$$

【００５２】

ここで、ｐｐＯｂｊｅｃｔは、同期ＣＯＭ（Component Object Model：コンポーネントオブジェクトモデル）をアプリケーション１０２に戻すために使用される引数であり、ｄｗＦｌａｇｓは、同期オブジェクトのビヘイビアと使用状況を判断するためにメソッドに渡されるフラグのセットである。これらは、利用できる予め定義されたフラグをビットワイズ（ビット単位）に結合したものである。

【００５３】

同期オブジェクトのインターフェースは次のようになっている。

【００５４】

【数２】

```
interface IDvdCmd : IUnknown
{
HRESULT WaitForStart( );
HRESULT WaitForEnd( );
}
```

【００５５】
戻されたオブジェクトは、アプリケーションによって解放されなければならない。プリインクリメントされたＣＯＭオブジェクトを戻すと、オブジェクトの寿命を正しく保つことができる。このインターフェースの変形では、スタートとエンドが現れた個所でアプリケーションを待たせる２つのメソッドが、システムの他の変更と共に追加されるようにオリジナルインターフェースが拡張されている。
【００５６】
【数３】

```
HANDLE GetStartHandle( );
HANDLE GetEndHandle( );
```

【００５７】
フラグは次のような値をとる。
・ＤＶＤ＿ＣＯＭ＿ＦＬＡＧ＿ＳｅｎｄＥｖｅｎｔｓ　－要求のステータスに関するイベントが送信される。
・ＤＶＤ＿ＣＯＭ＿ＦＬＡＧ＿Ｂｌｏｃｋ　－コマンドが完了するまで継続されない。
・ＤＶＤ＿ＣＯＭ＿ＦＬＡＧ＿Ｎｏｎｅ　－フラグがないことを示すプレースホルダ。
【００５８】
特殊な戻りコードであるＶＦＷ＿Ｅ＿ＤＶＤ＿ＣＭＤ＿ＣＡＮＣＥＬＬＥＤは、初期ＤＶＤ　ＡＰＩメソッドによって、ＩＤｖｄＣｍｄ：：ＷａｉｔＦｏｒＳｔａｒｔまたはＩＤｖｄＣｍｄ：：ＷａｉｔＦｏｒＥｎｄによって、あるいはコマンドが優先使用され、有効でなくなったとのイベント通知と共に戻される。
【００５９】
Ｃ＋＋でのコマンドオブジェクトの使い方のサンプル例を示すと、次の通りである。
【００６０】
【数４】

```
IDvdCmd* pObj ;
HRESULT hres = IDvdControl2->PlayTitle (15, DVD_CMD_FLAG_None ,&p0bj );
// 待機または通知しない
pObj ->Release ( ) ;
```

【００６１】
上述したように、プレイヤアプリケーション１０２は、次のいずれかによってコマンド

の開始と完了を判断することができる。すなわち、コマンドオブジェクトを直接に使用する方法、コマンドオブジェクトを使用しない方法、コマンドに関連するイベントが起こるのを待つ（listen to）方法、イベントとオブジェクトの組み合わせを使用してコマンドの複数のインスタンスをトラッキングしやすくする方法、などである。

【００６２】

オブジェクトを使用する場合

コマンドを指すＩＤｖｄＣｍｄを渡すことによって、ナビゲータは、新しいＩＤｖｄＣｍｄオブジェクトを割り振って、戻すようにしている。インターフェースメソッドＩＤｖｄＶｍｄ：：ＷａｉｔＦｏｒＳｔａｒｔ（）をコールすることは、コマンドが開始するまでブロックされ、ＩＤｖｄＶｍｄ：：ＷａｉｔＦｏｒＥｎｄ（）は、コマンドが完了するまで待たされる。コマンドがキャンセルされたときは、ナビゲータからは、ＶＦＷ＿Ｅ＿ＣＯＭＭＡＮＤ＿ＣＡＮＣＥＬＬＥＤが戻される。アプリケーションはオブジェクトを完了した後、Ｒｅｌｅａｓｅ（）をコールしてＣＯＭオブジェクトを解放しなければならない。ＤＶＤ　ＡＰＩに渡されるＮＵＬＬポインタは、コマンドオブジェクトがアプリケーションに戻されないこと、コマンド実行が標準同期モードで継続されることを示している。

【００６３】

他の２つのメソッドＧｅｔＳｔａｒｔＨａｎｄｌｅ（）およびＧｅｔＥｎｄＨａｎｄｌｅ（）からは、アプリケーションが、スタートまたはエンドイベントが起こるのを待っている間、他の要求（ディスクＩ／Ｏ、ユーザインターフェース変更、セマフォア変更、スレッドのアンブロッキング、他のプロセスとの通信など）が処理されるのをアプリケーションに待たせるシステム固有同期オブジェクトが戻される。そのあと、アプリケーションは、ＷａｉｔＦｏｒＳｔａｒｔ（）メソッドまたはＷａｉｔＦｏｒＥｎｄ（）メソッドをコールして、結果を取り出す。Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ（登録商標）　ＡＰＩでの例は、次のようになっている。

【００６４】

【数５】

```
handleStart = GetStartHandle()

Signaled = WaitForMultipleObjects( handleDiscIO, handleUserInter, …, handleStart )

If signaled = handleStart

        Result = DvdCmd->WaitForStart()
```

【００６５】

オブジェクトを使用しない場合

オブジェクトを管理する代わりに、アプリケーションは、ＮＵＬＬオブジェクトポインタを使用してＤＶＤ＿ＣＭＤ＿ＦＬＡＧ＿Ｂｌｏｃｋフラグを指定するだけで済ますことができる。コマンドは、完了するか、あるいはキャンセルされるまでリターンしない。ＡＰＩは、同期ビヘイビアをエミュレートしている。例えば、

【００６６】

【数６】

```
HRESULT hres = IDvdControl2->PlayTitle(uTitle, DVD_CMD_FLAG_Block,0) ;
```

【００６７】

は、意味的には次のものと同じである。

【００６８】

【数７】

```
IDvdCmd* pObj ;
HRESULT hres = IDvdControl2->PlayTitle( uTitle,
DVD_CMD_FLAG_Block, &p0bj);
If( succeeded ( hres) ) {
        Hres = pObj->WaitToEnd();
        pObj->Release();
}
```

【００６９】

イベントを使用する場合

ＤＶＤ＿ＣＭＤ＿ＦＬＡＧ＿ＳｅｎｄＥｖｅｎｔｓフラグを指定すると、ナビゲータは次のようなイベントを引き起こす。

【００７０】

【数８】

{EC_DVD_CMD_START, lParam1, HRESULT}

{EC_DVD_CMD_END, lParam1, HRESULT}

【００７１】

アプリケーションがあるコマンドを同期するだけでよいときは（またはコマンドのインスタンス間を区別しないときは）、同期オブジェクトは必要でなく、イベントだけが必要になる。ＮＵＬＬオブジェクトポインタはＤＶＤ　ＡＰＩメソッドに渡され、イベントと一緒に送られるｌＰａｒａｍ１は常に０にセットされる。

【００７２】

イベントおよびオブジェクトを使用する場合

オブジェクトとＤＶＤ＿ＣＭＤ＿ＦＬＡＧ＿ＳｅｎｄＥｖｅｎｔｓフラグの両方を指定すると、アプリケーションは異なるコマンドをトラッキングすることができる。ＤＶＤ２　ＡＰＩをコールすると、後で参照するためにアプリケーションが使用できるオブジェクトが戻される。イベント通知が送信されるとき、ＤＶＤ２　ＡＰＩは各イベントに対してユニーク識別子（または「クッキー（cookie）」）ｌＰａｒａｍ１を生成するので、アプリケーションは各イベントをＩＤｖｄＣｍｄオブジェクトに逆マッピングすることができる。このクッキーの方法によると、アプリケーションはイベントを見落としたとき、メモリがリークするのを防止され、ＤＶＤナビゲータにオブジェクトの有効性を検証させることができる。

【００７３】

ＤＶＤ２　ＡＰＩメソッドであるＩＤｖｄｌｎｆｏ２：：ＧｅｔＣｍｄＦｒｏｍＥｖｅｎｔ（ｌＰａｒａｍ１）は、クッキーをコマンドオブジェクトポインタにマッピングする。アプリケーションは、これらイベントの各々の処理を終えた後、戻されたポインタ上でＣＯＭ“Ｒｅｌｅａｓｅ”メソッドをコールしなければならない。アプリケーションがメッセージの処理を完全に終えたとき（通常は、ＥＮＤイベントを受け取った後）、セーブしていたグローバル・コマンド・ポインタ上で”Ｒｅｌｅａｓｅ”をコールしなければならない。

【００７４】

ブロッキング／非ブロッキングの例

以下に説明する例では、ＩＤｖｄＣｏｎｔｒｏｌ２インターフェースを使用すると、同期がどのように行われるかを示している。

【００７５】

理解しやすくするために、例のいくつかは、ＥＣ＿ＤＶＤ＿ＣＭＤイベントからのｌＰａｒａｍ１値をＩＤｖｄＣｍｄオブジェクトにマッピングするために使用される、以下の

ユーティリティ関数のことを指している。

【００７６】

【数９】

```
IDvdCmd* GetDvdCmd( LONG_PTR lParam )
{
    IDvdCmd* pCmd;
    plDvdInfo2->GetCmdFromEvent (iParam, &pCmd) ;
    return pCmd;
}
```

【００７７】

同期なし（非同期モデル）

アプリケーションは、次のメソッドをコールしてアクションを要求する。

【００７８】

【数１０】

```
HRESULT hres = IDvdControl2->PlayTitle( uTitle, 0, NULL);
```

【００７９】

イベントなしの同期

次の例は、イベントを使用しないでコマンドが終了するまで待たせる正しいやり方を示している。

【００８０】

【数１１】

```
IDvdCmd* pObj ;

HRESULT hres = IDvdControl2->PlayTitle( uTitle, 0, &p0bj);

If( SUCCEEDED) hres)) {

        pObj ->WaitToEnd ( ) ;

        pObj ->Release ( ) ;

}
```

【００８１】

イベントを使用した部分的同期

次の例は、ＩＤｖｄＣｍｄオブジェクトを管理しないでシングルイベントを同期するやり方を示している。

【００８２】

【数１２】

```
HRESULT hres = IDvdControl2->PlayTitle( uTitle,
        DVD_CMD_FLAG_SendEvents, NULL);
Function ProcesEvent( type, lParam1, lParam2 )
{
        switch( type )
        {
        case EC_DVD_CMD_END:
                HRESULT hres = lParam2;  // 結果コードは 1Param2 に入っている
        break;
        }
}
```

【００８３】

イベントを使用した完全同期

次の例は、イベントを使用してコマンドを待つ正しいやり方を示している。

【００８４】

【数１３】

```
// グローバルコード内
IDvdCmd* pGlobalObj = 0 ;
// 注：pGlobalObj はイベントが出される前にナビゲータによって割り当てられ
// る。他の場合は、イベントは pGlobalObj が初期化される前に
// (*1) の個所で起こることがある。
HRESULT hres = IDvdControl2->PlayTitle( uTitle,
DVD_CMD_FLAG_SendEvents, &pGlobalObj );
// (*1)
If( FAILED ( hres)) {
        pGlobalObj = NULL;
}
...
```

イベント処理関数の場合：

```
Function ProcessEvent( type, lParam1, lParam2 )
switch (type)
{
case EC_DVD_CMD_END:
        IDvdCmd* pObj = GetDvdCmd( lParam1 ) ;
        HRESULT hres = lParam2;
        If( NULL != pObj ) {
                // イベントから戻されたオブジェクトが PlayTitle から戻された
                // グローバルポインタと一致していれば、それを処理する。
                If (pGlobalObj == obj ) {
                        ProcessCmdEnd....
                        pGlobalObj ->Release ( ) ;
                        pGlobalObj = NULL;
        }
        pObj ->Release ( );
}
break ;
```

【００８５】

イベントおよび別のイベントループスレッドを使用した完全同期

次の例は、イベントを使用してコマンドを待つ正しいやり方を示している。

【００８６】

【数１４】

// グローバルコード内

```
IDvdCmd* pGlobalObj=0;
{
    LockCriticalSection
    HRESULT hres = IDvdControl2->PlayTitle( uTitle,
        DVD_CMD_FLAG_SendEvents, &pGlobalObj );
    If( FAILED ( hres)) {
        pGlobalObj = NULL;
    }
    UnlockCriticalSection
}

Function ProcessEvent( type, lParam1, lParam2 )

switch (type)
{
    case EC_DVD_CMD_COMPLETE:
    case EC_DVD_CMD_CANCEL:
    {
        CautoLock(globalCritSect );
        IDvdCmd* pObj = GetDvdCmd( lParam1 ) ;
        HRESULT hres = lParam2
        If( NULL = pObj ) {
            If (pGlobalObj == obj ) {
                pGlobalObj ->Release ( ) ;
                pGlobalObj = NULL;
            }
            pObj ->Release ( ) ;
        }
        break ;
    }
```

【００８７】

プレイバックやりとり制御機構の例

従来のＤＶＤ　ＡＰＩコマンドでは、コンテンツになんらかの変更が行われると、プレイヤアプリケーション１０２はカレント・コンテンツ・プレゼンテーションを切り捨てることを望み、新しいコンテンツにスイッチすることが想定されていた。改良されたＤＶＤ２　ＡＰＩコマンドでは、次のようなフラグでコマンドオブジェクト機構が拡張されている。

【００８８】

【数１５】

DVD_CMD_FLAG_Flush

DVD_CMD_FLAG_StartWhenRendered

DVD_CMD_FLAG_EndAfterRendered

【００８９】

ここで、．．＿Ｆｌｕｓｈフラグは、カレントコンテンツのプレゼンテーションが即時に切り捨てられて、新しいコンテンツが表示されることを開始できることを示している（

従来と同じ）。このフラグがないときは、カレント・コンテンツ・プレゼンテーションを最初に終了させることを示している。…＿．．Ｒｅｎｄｅｒｅｄフラグは、各コマンドの開始と終了のセマンティックスを変更している。デフォルトでは、コマンドは開始し、その処理を終えると、終了するようになっている。この新しいフラグは、コンテンツの変更結果が、それぞれ処理され、表示されるとき、開始と終了が起こることを示している。

【００９０】

ディスク通信機構の例

ＤＶＤ２　ＡＰＩ１０８ａ－ｂによると、プレイヤアプリケーションは、ＤＶＤナビゲータの汎用レジスタ（ＧＰＲＭ）を読み取ることができるだけでなく、以下のものを使用して、プレイヤアプリケーションにＧＰＲＭをセットさせることも可能になっている。

【００９１】

【数１６】

IDvdInfo2::GetAllGPRMs(WORD pwRegisterArray[16])

IDvdControl2::SetGPRM(ULONG ulindex, WORD wValue, DWORD dwFlags, IDvdCmd** ppCmd)

【００９２】

読み書き結合機能によると、ＤＶＤアプリケーションはディスク上のプログラムと「やりとり」することが可能になり、「制御によるアンロッキング」、つまり、制限されたコンテンツへのアクセスを実現することが可能になる。アプリケーションはＧｅｔＡｌＩＧＰＲＭを使用してカレントステートを読み取り、ＳｅｔＧＰＲＭを使用して特定のレジスタをセットすることができる。

【００９３】

ＳｅｔＧＰＲＭメソッドは、アプリケーションとＤＶＤナビゲータの仮想ＣＰＵを同期するためにも使用することができる。ＳｅｔＧＰＲＭメソッドは、ＤＶＤナビゲータがユーザコマンドを処理することが許されている期間にだけ実行される（プレゼンテーションフェーズと静止フェーズ、３．３．６．１　Ｖ１３－２８）。ナビゲーションコマンドの実行は、アトミック（atomic）であるものとみなされている。従って、ＧＰＲＭをセットすることは、これらのフェーズが現れるまで延期される。アプリケーションはコマンドオブジェクトおよびイベントの機構を使用して、調整を図ることができる。コマンドオブジェクトのイベント機構は、イベント通知（ドメイン変更やシステムレジスタに対する変更など）とシリアル化されている。アプリケーションはＳｅｔＧＰＲＭをコールし、コマンド完了イベントが受信されるまで待った後で、ＤＶＤナビゲータのステートの変更（ドメイン変更のこともあり得る）を示すイベントを待つことができる。

【００９４】

ディスクからアプリケーションへの通信を実現する１つのやり方は、次の擬似コードに示されている。

（１）ディスクはデータを送信し、応答を待つ：

ａ）ディスクはＧＰＲＭ値を変更する（オンディスク・ナビゲーション・コマンドを使用して）

ｂ）ディスクはそのステートを変更する（例えば、そのドメインを変更する）

ｃ）ループに入ってＧＰＲＭ変更を待つ（アプリケーションによって引き起こされる）

（２）アプリケーションはＧＰＲＭデータを受信し、応答する：

ｄ）ステート変更（例えば、ディスクのドメイン変更）を待つ

ｅ）ＧＰＲＭ値を読み取る

ｆ）ＳｅｔＧＰＲＭを使用してＧＰＲＭ値をセットする。

【００９５】

アプリケーションからディスクへの通信を実現する１つのやり方は、次の擬似コードに示されている。

（１）アプリケーションはデータを送信し、受信確認通知を待つ：

ａ）アプリケーションはＳｅｔＧＰＲＭを使用してデータをセットする

ｂ）アプリケーションはドメイン変更を待ってから継続する

（２）ディスクはデータを受信し、受信確認通知を返送する：

ｃ）ディスクはＧＰＲＭを読み取る

ｄ）ディスクはそのステートを変更する（例えば、そのドメインを変更する）。

【００９６】

クエリ（情報）インターフェースの例

ＤＶＤ規格では、どのデータ取り出しメソッドも提案されていなくても、ＤＶＤ２　ＡＰＩにはこの機能が提供されている。以下は、提供されているメソッドのリストである。

【００９７】

【数１７】
GetAllGPRMs
GetAllSPRMs
GetAudioLanguage
GetCurrentAngle
GetCurrentAudio
GetCurrentButton
GetCurrentDomain
GetCurrentLocation
GetCurrentSubpicture
GetNumberOfChapters
GetPlayerParentalLevel
GetSubpictureLanguage
GetTotalTitleTime
GetTitleParentalLevels
GetCurrentUOPS
GetCurrentVolumeInfo (IDVD1::GetDVDVolumeInfo)
GetDVDDirectory (IDVD1::GetRoot)
GetAudioAttributes([in] ULONG ulStream, [out] DVD_AudioAttributes *pATR);
GetCurrentVideoAttributes([out] DVD_VideoAttributes * pATR):
GetVMGAttributes([out] DVD_MenuAttributes * pATR);
GetTitleAttributes(ULONG ulTitle, [out] DVD_MenuAttributes * pMenu, [out] DVD_TitleAttributes* pTitle);
GetSubpictureAttributes([in] ULONG ulStream, [out] DVD_SubpictureAttributes *pATR);
GetButtonAtPosition(POINT point, [out] ULONG *puButtonIndex);
GetButtonRect(ULONG ulButton, RECT *pRect):
GetDefaultAudioLanguage(LCID* pLanguage, DVD_AUDIO_LANG_EXT* pAudioExt):
GetDefaultMenuLanguage(LCID* pLanguage):
GetDefaultSubpictureLanguage(LCID* pLanguage, DVD_SUBPICTURE_LANG_EXT*pSubpictureExtension);
GetDVDTextLanguageInfo(ULONG ulLangIndex, ULONG* pulNumOfStrings, LCID*pwLangCode, DVD_TextCharSet * pbCharacterSet);
GetDVDTextNumberOfLanguages(ULONG * pulNumOfLangs);
GetDVDTextStringAsNative(ULONG ulLangIndex, ULONG ulStringIndex, BYTE* pbBuffer,ULONG ulMaxBufferSize, ULONG* pulActualSize, enum DVD_TextStringType* pTyp);
GetDVDTextStringAsUnicode(ULONG ulLangIndex, ULONG ulStringIndex, WCHAR*pchBuffer, ULONG ulMaxBufferSize, ULONG* pActualSize, DVD_TextStringType* pType);
GetCmdFromEvent(LONG_PTR dwID, IDvdCmd** ppCmd);
GetDecoderCaps(DVD_DECODER_CAPS *pCaps);
GetDiscID(LPCWSTR pszwPath, ULONGLONG* pullUniqueID):
GetKaraokeAttributes([in] ULONG ulStream, DVD_KaraokeAttributes *pATR);
GetMenuLanguages(LCID *pLang, ULONG uMaxLang, ULONG *puActualLang);
IsAudioStreamEnabled(ULONG ulStreamNum, BOOL *pbEnabled);
IsSubpictureStreamEnabled(ULONG ulStreamNum, BOOL *pbEnabled);

【００９８】
制御インターフェースの例
１）期間プレイバックインターフェース（Period Playback Interface）
チャプタの範囲をプレイ実行するほかに、ＤＶＤ２　ＡＰＩでは、次のものを使用して時間期間をプレイ実行することを可能にしている。
【００９９】

【数１８】

PlayPeriodInTitleAutoStop(ULONG ulTitle, DVD_HMSF_TIMECODE* pStartTime,

DVD_HMSF_TIMECODE* pEndTime, DWORD dwFlags, IDvdCmd** ppCmd)

【０１００】

このメソッドを使用すると、アプリケーション（ビデオ編集プログラムやゲームなど）は、コンテンツの任意部分を正確にプレイバックすることができる。コマンドオブジェクト機構と結合すると、スライドショウプレゼンテーションやビデオゲームの合間、キオスクのようなアプリケーションは、シングルＤＶＤ２　ＡＰＩコマンドを使用して実現することができる。

【０１０１】

２）デフォルト言語インターフェース（Default Language Interface）

【０１０２】

【数１９】

SelectDefaultAudioLanguage(LCIDLanguage,DVD_AUDIO_LANG_EXT

audioExtension)

SelectDefaultSubpictureLanguage(LCIDLanguage, DVD_SUBPICTURE_LANG_EXT

subpictureExtension)

【０１０３】

これらのメソッドを使用すると、アプリケーション（ユーザからの）はＤＶＤプレイバックに対してデフォルト言語選択項目をセットすることができる。

【０１０４】

３）ボタンインデックス選択（Button Index Selection）

アプリケーションは、次のメソッドを使用すると、メニューナビゲーションを自動化することが可能になっている。

・SelectButton(ULONG ulButton)。

【０１０５】

４）ブックマーキングＡＰＩ（Bookmarking API）

アプリケーションはＤＶＤ全体のステートをセーブし、リストアすることができる（ブックマーク特許を参照）。

【０１０６】

【数２０】

GetState(IDvdState **pStateData)

SetState(IDvdState* pState, DWORD dwFlags, [out] IDvdCmd* ppCmd)

【０１０７】

５）その他

ＡｃｃｅｐｔＰａｒｅｎｔａｌＬｅｖｅｌＣｈａｎｇｅ（ＢＯＯＬ　ｂＡｃｃｅｐｔ）－下述する「最低親レベルブランチング」セクションを参照のこと。

・SetGPRM(ULONG ulindex, WORD wValue, DWORD dwFlags, [out] IDvdCmd** pp Cmd)

・SetOption(DVD_OPTION_FLAG flag, BOOL bEnable)－　拡張可能オプションセッティング機構。

【０１０８】

最低親レベルブランチングを調整する機構

ＤＶＤ規格（セクション４．６．４．１　ｐＶ１４－１９７）によれば、ＤＶＤナビゲータが’ＳｅｔＴｍｐＰＭＬ’（一時親管理レベルをセットする）コマンドに出会ったとき、アプリケーションに許可を要求して（「プレイヤに組み込まれている一時親レベル変更機能をコールする」）、カレントレベルを一時的に上げる必要があった。親レベル変更が許されていれば、ナビゲータは親レベルを上げて、限定されたコンテンツ部分にブランチしている。許されていなければ、ナビゲータは次のコマンドの実行を続ける。

【０１０９】

従来のＤＶＤ　ＡＰＩのセマンティックスによると、ＤＶＤナビゲータがＳｅｔＴｍｐＰＭＬ命令を実行するとき、ＰＡＲＥＮＴＡＬ＿ＬＥＶＥＬ＿ＴＯＯ＿ＬＯＷイベントをアプリケーションに送信するだけである。ナビゲータは、親レベル変更が失敗したものとして直ちに次のコマンドの実行を続けている。アプリケーションはイベントを受信し、プレイバックを中止し、親レベルを変更するためのユーザインターフェースを表示し、その後、初めからムービをリスタートしている。ＤＶＤ規格によれば、ナビゲータが親レベルを変更することが許されるのは、ナビゲータがＳＴＯＰドメインにあるときだけである。その結果、ナビゲータは変更個所でポーズしないので、プレイバックを中止しなければならない。

【０１１０】

ＤＶＤ２　ＡＰＩ１０８ａ－ｂによると、例えば、以下に説明するシーケンスで行うことができる。アプリケーションは、親レベル変更機能が利用可能であるかどうかを、次のメソッドを使用してＡＰＩに通知する。

【０１１１】

【数２１】

IDVDControl2::SetOption(DVD_NotifyParentalLevelChange, TRUE)

【０１１２】

ＤＶＤナビゲータがＳｅｔＴｍｐＰＭＬ命令に出会ったとき、ナビゲータは、ＰＡＲＥＮＴＡＬ＿ＬＥＶＥＬ＿ＴＯＯ＿ＬＯＷイベントをアプリケーションに送信する。アプリケーションは、ユーザが親レベルを大きくできるようにする、なんらかのユーザインターフェースを表示することが期待されている。ＤＶＤナビゲータは、アプリケーションがその応答として、ＴＲＵＥまたはＦＡＬＳＥを指定したＩＤＶＤＣＣｏｎｔｒｏｌ２：：ＡｃｃｅｐｔＰａｒｅｎｔａｌＬｅｖｅｌＣｈａｎｇｅ（）をコールするまでブロックし、その後、プレイバックを中止しなくても、その指定に応じて続行する。

【０１１３】

マルチセグメント親レベルブランチのプレイバックを支援する機構

ＤＶＤ規格（セクション４．１．４．１　Ｖ１４－２２）には、異なるプログラムチェイン（通常は、コンテンツに起こり得る、異なるセグメント）を、カレント親レベルに基づいて選択する構成が記載されている。例えば、ビデオのある個所で、異なるバージョンのシーンが利用可能になっていることがあり、これらは親レベルに基づいてナビゲータによって自動的に選択されている（例えば、ＰＧ、Ｐレーティングまたは子供を目的としたセグメント）。

【０１１４】

各々のタイトルごとに、ＰＴＬ＿ＭＡＩテーブルは、カレント親レベルを１６ビットマスクにマッピングしている。プレイバックの期間に、ＤＶＤナビゲータは、カレント親ビットマスクをＰＴＬ＿ＭＡＩテーブルから取得している。親ビットマスクは、ナビゲータが親ブロック（各プログラムが排他的親ビットマスクをもっているプログラムチェインの集まり）に出会ったとき使用される。ナビゲータは、ＶＴＳ＿ＰＧＣＩ＿ＳＲＰ（セクション４．２．３　Ｖ１４－６２）内の各ＰＴＬＩＤ＿ＦＬＤをサーチし、カレント親ビットマスクと共通ビットを共有しているビットマスクをもつプログラムチェインを探し出している。

【０１１５】
カレントビットマスクと部分的に一致しているプログラムチェインが見つからなければ、従来バージョンのＤＶＤナビゲータでは、プレイバックを中止し、ＤＶＤ＿ＥＲＲＯＲ＿ＬｏｗＰａｒｅｎｔａｌＬｅｖｅｌイベントをアプリケーションに送信している。
【０１１６】
ユーザを援助するために、ＤＶＤ２　ＡＰＩ１０８ａ－ｂのいくつかの実装例では、ユーザが継続できるようにする、最低限必要な親レベルを計算するために次のようなアルゴリズムが使用されている。
ＰＴＬ＿ＭＡＳＫを＝０に初期化する（取り得る許容親レベル）
ＶＴＳ＿ＰＧＣＩ＿ＳＲＰ内の各プログラムチェインインデックスｉについて
ＶＴＳ＿ＰＧＣＩ＿ＳＲＰ［ｉ］．ＢｌｏｃｋＴｙｐｅ＝１ならば（親ブロック内）
ＰＴＬ＿ＭＡＳＫ＝ＰＴＬ＿ＭＡＳＫ　ｕｎｉｏｎ　ＶＴＳ＿ＰＧＣＩ＿ＳＲＰ［ｉ］．ＰＴＬ＿ＩＤ＿ＦＬＤ
ＰＴＬ＿ＭＡＳＫ＝０ならば
親レベルは存在しないので、どのレベルでも有効である。
そうでなければ
ＰＴＬ＿ＭＡＩ内の各親レベルインデックスｉについて
ＰＴＬ＿ＬＶＬＩ＝ＰＴＬ＿ＭＡＩ［８－ｉ］と置く
ＰＴＬ＿ＬＶＬＩ［ｔｉｔｌｅ＿ｉｎｄｅｘ］＆ＰＴＬ＿ＭＡＩ［８－ｉ］＝０ならば
（注：ＶＭＧＭドメインではｔｉｔｌｅ＿ｉｎｄｅｘ＝０
ｉを戻す。
【０１１７】
インデックスｉは、ＤＶＤ＿ＥＲＲＯＲ＿ＬｏｗＰａｒｅｎｔａｌＬｅｖｅｌイベントと一緒に戻される。アプリケーション１０２は、このインデックスを使用して、取り得る親レベル設定値をユーザに提案することができる。
【０１１８】
ブックマーキング
ＤＶＤナビゲータ１０６は、プレイヤアプリケーション１０２がＤＶＤプレイバックのカレントステート１１４を符号化（エンコード）して、持続的データブロックを収めている抽象オブジェクト（ブックマーク１５０と呼ばれる）に格納しておくことを可能にする構成になっている。図１３は、例示のブックマーキング機能を示す図である。
【０１１９】
使い方をさらに抽象化し、単純化するために、ＤＶＤ２　ＡＰＩ１０８ａ－ｂは、ブックマークに収められているステート情報をセーブし、リストアし、クエリするように構成されている。プレイヤアプリケーション１０２は、ナビゲータ１０６を使用してブックマーク１５０内の情報をクエリし、将来の使用に備えてその情報をセーブしておくことができる。そのあと、プレイヤアプリケーション１０２は、ブックマークに収められたＤＶＤプレイバックステート１１４をリストアするようにＤＶＤナビゲータ１０６に指示することによって、プレイバックを再開することができる。ブックマークをリストアすると、プレイヤアプリケーションは、任意のどのロケーションからでも、いくつのロケーションからでも、ＤＶＤコンテンツ１１０をプレイ実行することを開始することができる。ブックマークは、短期（メモリ）ストレージにも、長期ストレージ（例えば、ハードドライブ）にも格納しておくことができ、プレイヤアプリケーション１０２および／またはＰＣがシャットダウンされ、リスタートされた後でも、リストアすることができる。ブックマークはＤＶＤナビゲータのステート（内部レジスタ値、プレイバックロケーション、プレイバックステートなど）だけでなく、プレイ実行されているカレント・ディスク・コンテンツとユーザの設定値に関する情報も収めている。プレイヤアプリケーションは、このエキストラ情報を使用すると、以前にセーブしておいたブックマークから該当のブックマークをインテリジェントに選択して、例えば、それを特定のディスク（通常は、プレイ実行され

ているディスク）に対してプレイ実行させることができる。ブックマークは、ユーザ間でも、種々のアプリケーション間でも、共有することができる。

【０１２０】

ブックマーキング抽象データ型は２つの側面からなっている。すなわち、１）実際のブックマーク１５０自体、および２）ブックマークに収められた情報をセーブし、リストアし、クエリするために使用されるＡＰＩコールである。いくつかの実装例によれば、ブックマーク１５０は、少なくとも次のような情報を収めている。すなわち、実質的に唯一のディスク識別子１４５、表示されているカレント・ビデオ・オブジェクト・ユニット（video object unit：ＶＯＢＵ）のアドレス（ＤＶＤ規格のセクション５．１．１）と、ループカウントおよびシャッフルヒストリ（ＤＶＤ規格のセクション３．３．３．２）と、カレントＤＶＤ再開情報（その概要はＤＶＤ規格のセクション３．３．３．２に記載されている）と、カレントＤＶＤ一般パラメータ（ＧＰＲＭ）およびシステムパラメータ（ＳＰＲＭ）の値（セクション４．６．１．１と４．６．１．２）と、カレントドメインおよびフェーズ（セクション３．３．３と３．３．６）とである。いくつかの別の実装では、ブックマークは、バージョン管理情報および保全性情報も収めている。ブックマーク１５０は抽象オブジェクトとしても、バイナリデータのブロックとしてもパッケージ化して、格納しておくことができる。

【０１２１】

上記のようなブックマーキング技法を提供するために、ＤＶＤ２　ＡＰＩ　１０８のいくつかの典型的な実装では、次のようなメソッドがサポートされている。

１．カレントロケーションからブックマークを作成する場合
　　Ｂｏｏｋｍａｒｋ＝ＧｅｔＢｏｏｋｍａｒｋ（）
２．ＤＶＤナビゲータにそのロケーションをブックマークに変換させる場合
　　ＳｅｔＢｏｏｋｍａｒｋ（ｂｏｏｋｍａｒｋ）
３．ブックマークを目的としているディスクを見付ける場合
　　ＤｉｓｃＩＤ＝ＧｅｔＤｉｓｃＩｄｅｎｔｉｆｉｅｒＦｒｏｍＢｏｏｋｍａｒｋ（ｂｏｏｋｍａｒｋ）
４．ブックマークをそのバイナリ表現におよびそのバイナリ表現から変換する場合
　　ＢｉｎａｒｙＤａｔａ（ｄａｔａ，ｓｉｚｅ）＝ＣｏｎｖｅｒｔＢｏｏｋｍａｒｋＴｏＢｉｎａｒｙ（ｂｏｏｋｍａｒｋ）
　　Ｂｏｏｋｍａｒｋ＝ＣｏｎｖｅｒｔＢｉｎａｒｙＴｏＢｏｏｋｍａｒｋ（ＢｉｎａｒｙＤａｔａ）。

【０１２２】

以下は、カレントロケーションを格納することを実現し、パワーセーブ（電力節約）機能を実現するように（つまり、コンピュータのステートをセーブし、復帰可能な低電力状態に入れるように）書かれたアプリケーションの擬似コードを示したものである。

【０１２３】

【数２２】

```
Bookmark = GetBookmark()

BinaryData(data,size) = ConvertBookmarkToBinary( bookmark )

Store BinaryData(data,size)

シャットダウンするか、パワーセーブに入る。
```

【０１２４】

以下は、パワーセーブから戻るとき、ＤＶＤのプレイバックを再開するためのコードを示したものである。

【０１２５】

【数２３】

```
Retrieve BinaryData(data,size)
Bookmark = ConvertBinaryToBookmark( BinaryData )
If GetDiscIdentifierFromBookmark( bookmark ) = current Disc Id
Then
    SetBootmark( bookmark )
```

【０１２６】

以下は、インテリジェントブックマークを実現するために書かれたアプリケーションの擬似コードの例を示したものである。

【０１２７】

【数２４】

```
For each stored bookmark “bookmark”
If GetDiscIdentifierFromBookmark( bookmark ) = current Disc Id
Then
    Add bookmark to the user selectable list
```

【０１２８】

ユニーク識別子の生成

現行ＤＶＤ規格では、組み込まれたユニーク識別子が各ディスクに付けられている（「ＤＶＤユニーク識別子」）。しかし、アプリケーションは、ディスクの作成者が識別子を正しく実装していることを想定していなければならない。残念ながら、常にそうであるとは限らない。

【０１２９】

多くのアプリケーションは、どのＤＶＤディスクであるかを特定するために唯一のタグを必要とし、例えば、ユーザがＤＶＤディスクをスワップしたとき、プレイバックシステムは、新しいディスクが導入されたかどうかを判断する必要がある。新しいディスクが導入されていれば、プレイバックシステムはプレイバックをリセットする必要があり、そうでなければ、ユーザが見ているのを中断することなく継続することが可能になっている。ディスクを区別する能力がなければ、常にリセットしなければならない。ユニーク識別子１４５（図１３参照）を使用すると、異なるディスク（ただし、同じディスクの異なるコピーではなく）を区別する能力が得られることになる。

【０１３０】

また、ユニーク識別子１４５を使用すると、アプリケーションは、格納された情報と特定のＤＶＤディスクとの間に矛盾がないかを確かめることもできる。アプリケーションは、キャッシュされた情報を正しくないディスクで使用すると、失敗することになる。例えば、ユーザがディスク上のセーブされたロケーションを、ブックマークを使用して再コールしようとしたとき、ＤＶＤナビゲータ１０８は、ブックマークに格納されたユニーク識別子をカレントディスクのユニーク識別子と比較することによって、データに矛盾がないかを確かめることができる。プレイバックは、これらの識別子が一致しているときだけ継続される。

【０１３１】

ユニーク識別子１４５を使用すると、アプリケーションは、ユニーク識別子をデータベ

ースへのインデックスとして使用することにより、追加情報をディスクと関連付けることができる。例えば、ＤＶＤ規格がディスク上のテキスト情報をサポートしていても、その情報が使用されるのは稀である。ディスクのタイトルとコンテンツのＷｅｂベースデータベースは、ディスク上の識別子を計算した後、アプリケーションによって格納し、取り出すことが可能になっている。

【０１３２】

ＤＶＤディスク上の現行組み込みユニーク識別子は不十分である。第一に、識別子はサイズが相対的に大きく（３２バイト）、識別子が実際に唯一であることを保証するには、ディスク作成者が頼りであり、中央のエンティティは、企業間でユニーク性が保たれることを保証するために、識別子の範囲をディスク作成者に割り当てなければならない。

【０１３３】

他の従来の「ユニーク」識別子アルゴリズムでは、ディスクが非常に多数であるときは、ユニーク識別子が得られない。このアルゴリズムによれば、２つのディスクに同じ識別子が割り当てられる確率は、ＤＶＤディスクの総数が増加するのに伴い指数的に増加していく。今後の動向として、ＤＶＤディスクは増加することが予想されるので、多くの「ユニーク」識別子ルーチンは不十分になってしまう。さらに、これらのアルゴリズムは、その特性が分かっていないことおよび／またはその特性が実証可能でないことが多い。特性が分かっていないと、得られた識別子の有効性または適切性を述べることは不可能である。

【０１３４】

本発明のいくつかの実装例によれば、ユニーク識別子１４５は、ＤＶＤのＶＩＤＥＯ＿ＴＳディレクトリに登録されている種々ファイルのファイルヘッダとファイル内容のバイナリ表現を連結または配列したものの６４ビットＣＲＣを計算することによって生成される。この機能の詳細は図１３と図１４に示されている。

【０１３５】

ＵｎｉｑｕｅＩＤ２アルゴリズムは、次の４ステップで識別子を生成する。

ステップ１。ＶＩＤＥＯ＿ＴＳディスレクトリのファイル名が収集され、アルファベット順にソートされる。

ステップ２。各ファイルからのファイルヘッダがＣＲＣで計算される。

ステップ３。ＶＭＧＩファイル（“ＶＩＤＥＯ＿ＴＳ¥ＶＩＤＥＯ¥ＴＳ．ＩＦＯ”）からのデータがＣＲＣで計算される。

ステップ４。最初のＶＴＳＩファイル（“ＶＩＤＥＯ＿ＴＳ¥ＶＴＳ＿ｘｘ．Ｏ．ＩＦＯ”）からのデータがＣＲＣで計算される。

【０１３６】

６４ビットＣＲＣは、フィールドＧＦ（２）内の既約多項式（irreducible polynomial）を使用して計算される。次は、多項式の例である。

【０１３７】

【数２５】

$$P_{64} = x^{64} + x^{61} + x^{58} + x^{56} + x^{55} + x^{51} + x^{50} + x^{47} + x^{42} + x^{39} + x^{38} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{26} + x^{25} + x^{22} + x^{17} + x^{14} + x^{13} + x^{9} + x^{8} + x^{6} + x^{3} + x^{0}$$

【０１３８】

この多項式は、ｘ”－１が次数６４の既約（素数）因数をもつような指数ｎを見つけることによって計算される。

【０１３９】

実際のＣＲＣ値は、いくつかの例では、バイナリデータのすべてを、シングルブロック（ビットb_0からb_nまで）になるように連結し、多項式の係数x^iに各ビットb_iを割り当て、多項式P_{64}で除した後の余りを計算することによって計算される。

【０１４０】

【数２６】

$$CRC_{64}=\left[\sum_{i=0}^{n}b_i x^i\right]\bmod p_{64}$$

【０１４１】

以下は、実装例を示したものである。

ステップ１

ＶＩＤＥＯ＿ＴＳディレクトリのファイル名が収集され、アルファベット順リストにソートされる。

ステップ２

リスト内の各ファイル名について、以下に示す構造が記述され、ＣＲＣに追加される（すべてのデータフィールドはＬＳＢが先頭になっている）。

符号なし６４ビット整数：ｄａｔｅＴｉｍｅ（Ｊａｎｕａｒｙ　１，１６０１からの１００ナノ秒インターバルの経過時間）

符号なし３２ビット整数：ｄｗＦｉｌｅＳｉｚｅ

ＢＹＴＥ　ｂＦｉｌｅｎａｍｅ［ｆｉｌｅｎａｍｅＬｅｎｇｔｈ］

ＢＹＴＥ　ｂＦｉｌｅｎａｍｅＴｅｒｍＮｕｌｌ＝０

ステップ３

ファイル“ＶＩＤＥＯ＿ＴＳ￥ＶＩＤＥＯ＿ＴＳ．ＩＦＯ”の最初の６５５３６バイトが存在していれば、読み取られ、ＣＲＣに追加される。ＩＦＯファイルが６５５３６未満であれば、ファイル全体が追加される。

ステップ４

ファイル“ＶＩＤＥＯ＿ＴＳ￥ＶＴＳ＿０１＿Ｏ．ＩＦＯ”の最初の６５５３６バイトが存在していれば、読み取られ、ＣＲＣに追加される。ＩＦＯファイルが６５５３６未満であれば、ファイル全体が追加される。

【０１４２】

本発明の種々の方法および構成のいくつかの好ましい実施形態を、添付図面を参照して詳しく説明してきたが、当然に理解されるように、本発明は上述した実施形態例に限定されるものではなく、請求項に記載されている本発明の精神から逸脱しない範囲で種々の再構成、変更および置換が可能である。さらに、本明細書の中で引用されている参考文献の各々は、その全内容が、引用によっておよびあらゆる目的のために、本明細書に含まれるものである。

【０１４３】

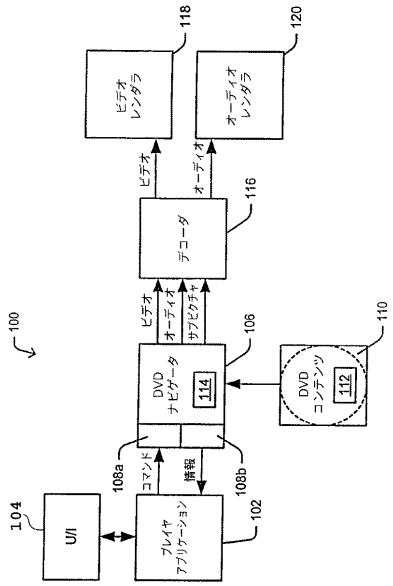
以上、説明したように、本発明によれば、マルチメディア・コンテンツ・プレイバック・プロセスに対するマルチメディア・ナビゲータ・プログラムのカレントステートを含むブックマークを、ＡＰＩに選択的に生成させ、ブックマークをマルチメディア・プレイヤ・アプリケーションに引渡すことによって、プレイヤアプリケーションがプレイバック期間中にロケーションにより正確に「ブックマーク」を付けて、選択したブックマーク付きのロケーションからプレイバックを、あとで再開することができる。

【符号の説明】

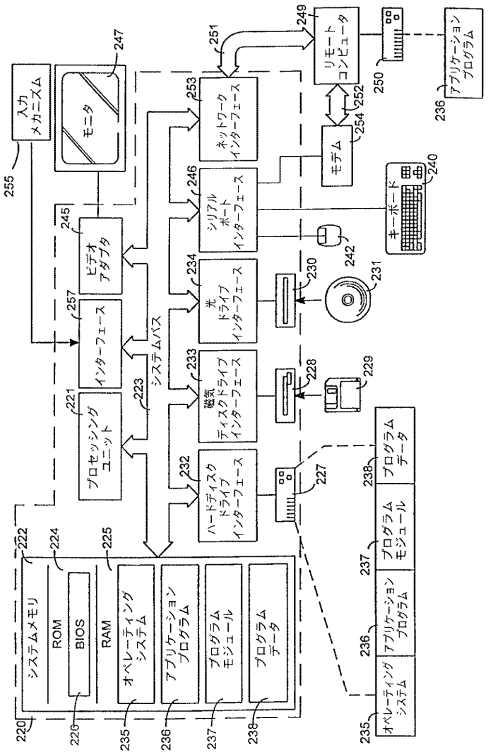
【０１４４】

１００　ＤＶＤプレイヤ
１０２　プレイヤアプリケーション
１０４　ユーザインターフェース（Ｕ／Ｉ）
１０６　ＤＶＤナビゲータ
１０８ａ－ｂ　ＤＶＤ２　ＡＰＩ
１１０　ＤＶＤコンテンツ
１１８　ビデオレンダラ
１２０　オーディオレンダラ

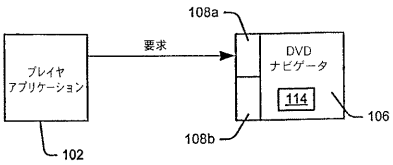
【図１】



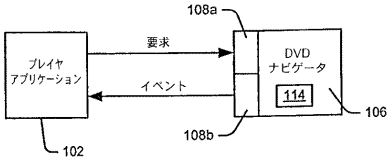
【図２】



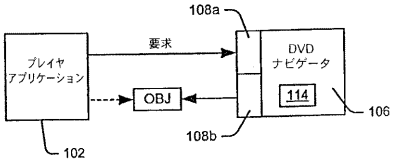
【図３】



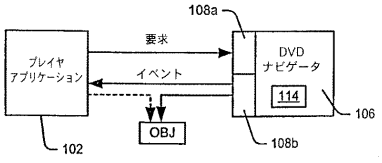
【図４】



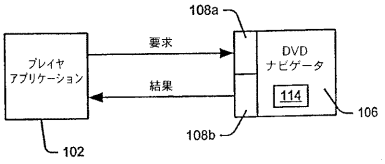
【図５】



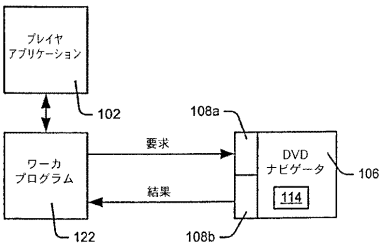
【図６】



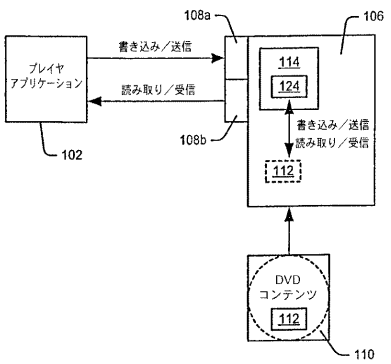
【図７】



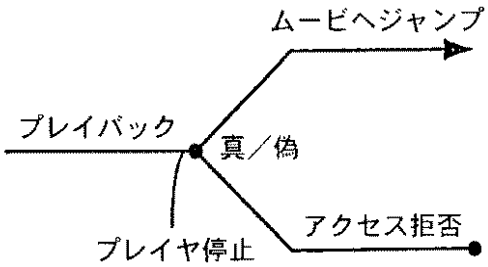
【図８】



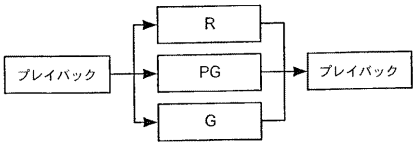
【図９】



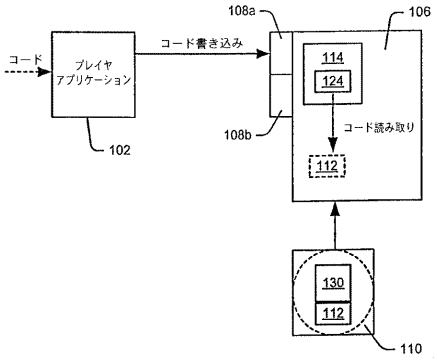
【図１０】



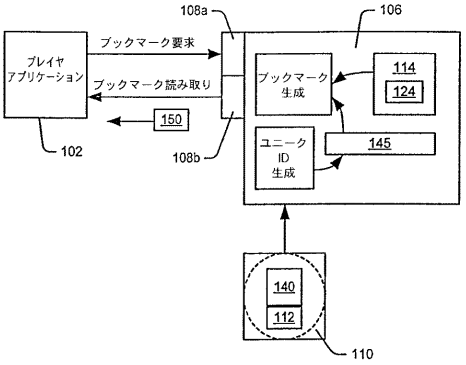
【図１１】



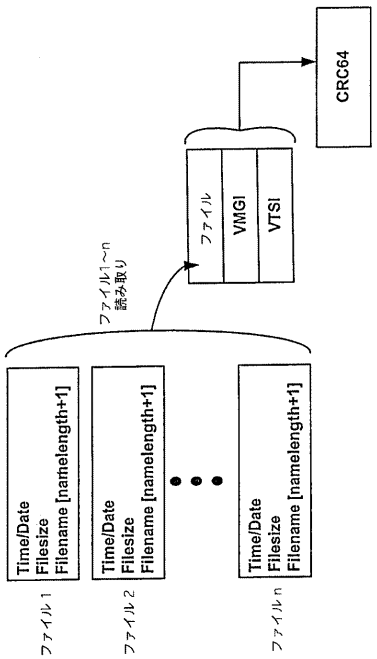
【図１２】



【図１３】



【図１４】



フロントページの続き

(72)発明者　アロック　チャクラバルティ
アメリカ合衆国　９８００６　ワシントン州　ベルビュー　１４１　プレイス　サウスイースト　５７２４

Ｆターム(参考) 5D077 AA23　EA31