(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4879480号
(P4879480)
(45)発行日　平成24年2月22日(2012.2.22)　(24)登録日　平成23年12月9日(2011.12.9)

(51)Int.Cl.		F I		
G11B 27/10	(2006.01)	G11B	27/10	A
G11B 20/10	(2006.01)	G11B	20/10	321Z
H04N 5/85	(2006.01)	H04N	5/85	Z

請求項の数 16　(全 65 頁)

(21)出願番号　特願2004-350193 (P2004-350193)
(22)出願日　平成16年12月2日 (2004.12.2)
(65)公開番号　特開2006-164328 (P2006-164328A)
(43)公開日　平成18年6月22日 (2006.6.22)
審査請求日　平成19年11月26日 (2007.11.26)

(73)特許権者　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100082762
弁理士　杉浦　正知
(74)代理人　100120640
弁理士　森　幸一
(73)特許権者　310021766
株式会社ソニー・コンピュータエンタテインメント
東京都港区港南１丁目７番１号
(72)発明者　浜田　俊也
東京都品川区北品川６丁目７番３５号　ソニー株式会社内

最終頁に続く

(54)【発明の名称】再生装置、再生方法および再生プログラム、記録媒体、ならびに、データ構造体

(57)【特許請求の範囲】
【請求項１】
記録媒体に記録されたコンテンツデータを再生する再生装置において、
少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、該コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体からデータを読み出す読み出し手段と、
上記再生制御プログラムに従い上記コンテンツデータを再生するプレーヤ手段と、
上記プレーヤ手段に対してユーザ操作に応じた制御コマンドを与える制御コマンド出力手段と
を備え、
上記プレーヤ手段は、上記コンテンツデータを再生しているか否かで分類される２状態と、上記制御コマンド出力手段からの制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき上記コンテンツデータの再生を制御し、
上記プレーヤ手段の上記４状態間の状態遷移は、上記再生制御プログラムによって発生される
再生装置。
【請求項２】
請求項１に記載の再生装置において、
上記プレーヤ手段の上記４状態間の状態遷移は、上記制御コマンドによって発生させな

いようにしたことを特徴とする再生装置。
【請求項３】
請求項１に記載の再生装置において、
上記プレーヤ手段の上記４状態間の状態遷移は、上記プレーヤ手段により自発的に発生しないようにしたことを特徴とする再生装置。
【請求項４】
請求項１に記載の再生装置において、
上記プレーヤ手段は、初期化直後には、上記コンテンツデータを再生していない状態で、且つ、上記制御コマンド出力手段からの上記制御コマンドを受け付けない状態になるようにしたことを特徴とする再生装置。
【請求項５】
請求項１に記載の再生装置において、
上記プレーヤ手段で再生中の上記コンテンツデータの状態を示す再生状態情報を記憶する第１の記憶手段と、
上記第１の記憶手段に記憶された上記再生状態情報がバックアップされる第２の記憶手段と
をさらに有し、
上記第１の記憶手段に記憶された上記再生状態情報の上記第２ の記憶手段へのバックアップおよび上記第２の記憶手段にバックアップされた上記再生状態情報の上記第１の記憶手段へのリストアは、上記プレーヤ手段の上記４ 状態間での状態遷移に伴い行われること
を特徴とする再生装置。
【請求項６】
請求項５に記載の再生装置において、
上記バックアップは、上記プレーヤ手段が上記制御コマンドを受け付ける状態且つ上記コンテンツデータを再生している状態から、他の状態へ遷移するときに行われることを特徴とする再生装置。
【請求項７】
請求項５に記載の再生装置において、
上記リストアは、上記プレーヤ手段の状態が上記制御コマンドを受け付ける状態且つ上記コンテンツデータを再生している状態に遷移するときに行われることを特徴とする再生装置。
【請求項８】
請求項５に記載の再生装置において、
上記リストアが、上記再生制御プログラムによる、上記第２の記憶手段にバックアップされた上記再生状態情報を用いて上記コンテンツデータの再生を復帰する命令に基づく上記状態遷移に伴い行われるときは、上記リストア後に上記第２の記憶手段にバックアップされている上記再生状態情報を破棄するようにしたことを特徴とする再生装置。
【請求項９】
請求項５に記載の再生装置において、
上記再生制御プログラムによる上記コンテンツデータの再生を行う命令に基づき上記プレーヤ手段の状態が、上記制御コマンドを受け付ける状態且つ上記コンテンツデータを再生している状態に遷移するときは、上記第２の記憶手段にバックアップされている上記再生状態情報を破棄するようにしたことを特徴とする再生装置。
【請求項１０】
請求項５に記載の再生装置において、
上記再生制御プログラムから上記プレーヤ手段に対して渡される、上記コンテンツの再生を停止させる命令に、上記第２の記憶手段にバックアップされている上記再生状態情報を破棄するか否かを示す情報を付加するようにしたことを特徴とする再生装置。
【請求項１１】

記録媒体に記録されたコンテンツデータを再生する再生方法において、
少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、該コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された上記再生制御プログラムに従いプレーヤ手段でなされる上記コンテンツデータの再生を、
上記プレーヤ手段の、上記コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御し、
上記プレーヤ手段の上記４状態間の状態遷移は、上記再生制御プログラムによって発生される
再生方法。
【請求項１２】
記録媒体に記録されたコンテンツデータを再生する再生方法をコンピュータ装置に実行させる再生プログラムにおいて、
上記再生方法は、
少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、該コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された上記再生制御プログラムに従いプレーヤ手段でなされる上記コンテンツデータの再生を、
上記プレーヤ手段の、上記コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御し、
上記プレーヤ手段の上記４状態間の状態遷移は、上記再生制御プログラムによって発生される
再生プログラム。
【請求項１３】
記録媒体に記録されたコンテンツデータを再生する再生方法をコンピュータ装置に実行させる再生プログラムが記録されたコンピュータ装置に読み取り可能な記録媒体において、
上記再生方法は、
少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、該コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された上記再生制御プログラムに従いプレーヤ手段でなされる上記コンテンツデータの再生を、
上記プレーヤ手段の、上記コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御し、
上記プレーヤ手段の上記４状態間の状態遷移は、上記再生制御プログラムによって発生される
コンピュータ装置に読み取り可能な記録媒体。
【請求項１４】
少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、該コンテンツデータの再生をプレーヤ手段に制御させる再生制御プログラムとが記録された、コンピュータ装置に読み取り可能な記録媒体であって、
上記再生制御プログラムは、
上記コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき上記コンテンツデータの再生を制御する上記プレーヤ手段に対して再生制御命令を与えて上記コンテンツの再生を制御させ、
さらに、上記プレーヤ手段に対して上記４状態間の状態遷移を発生させる

コンピュータ装置に読み取り可能な記録媒体。
【請求項１５】
請求項１４に記載の記録媒体において、
上記再生制御プログラムは、
再生中の上記コンテンツデータの状態を示す再生状態情報を記憶する第１の記憶手段に記憶された上記再生状態情報の、上記第１の記憶手段に記憶された上記再生状態情報がバックアップされる第２の記憶手段へのバックアップおよび上記第２の記憶手段にバックアップされた上記再生状態情報の上記第１の記憶手段へのリストアが上記プレーヤ手段の上記４状態間での状態遷移に伴い行われるようにされた上記プレーヤ手段に対して渡される、上記コンテンツの再生を停止させる命令に、上記第２の記憶手段にバックアップされている上記再生状態情報を破棄するか否かを示す情報を付加する
ようにしたことを特徴とするコンピュータ装置に読み取り可能な記録媒体。
【請求項１６】
少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、該コンテンツデータの再生をプレーヤ手段に制御させる再生制御プログラムと
を含むデータ構造体であって、
上記再生制御プログラムは、
上記コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき上記コンテンツデータの再生を制御する上記プレーヤ手段に対して再生制御命令を与えて上記コンテンツの再生を制御させ、
さらに、上記プレーヤ手段に対して上記４状態間の状態遷移を発生させる
データ構造体。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、大容量の記録媒体に記録されたプログラムの再生制御を容易に行うことができるようにした再生装置、再生方法および再生プログラム、記録媒体、ならびに、データ構造体に関する。
【背景技術】
【０００２】
ランダムアクセスおよび着脱が可能な記録媒体として、ＤＶＤ(Digital Versatile Disc)が出現して久しいが、近年では、このＤＶＤよりも大容量のディスク状記録媒体や、ＤＶＤより携帯に便利な小型のディスク状記録媒体の開発が進められている。
【０００３】
ＤＶＤに対してビデオコンテンツを記録するための規格として、ＤＶＤビデオ(DVD-Video)が規定されている。ＤＶＤビデオにおいては、ディスク再生中のＤＶＤプレーヤは、様々な状態を取り得る。ＤＶＤビデオでは、このＤＶＤプレーヤの取り得る状態をドメインと呼ばれる５種類の状態に分類し、ＤＶＤプレーヤは、様々な条件によってドメインの間を遷移するというモデルが適用されていた。すなわち、ＤＶＤドメインは、５個の可能な値を持つ状態変数の一種と考えることができる。ＤＶＤプレーヤは、現在ディスクから読み取り中のコンテンツの種類を把握するために、この状態変数を監視する。
【０００４】
ＤＶＤビデオでは、次の５種類のドメインが定義されている。
（１）ファーストプレイドメイン(FP_DOM)
（２）ビデオマネージャメニュードメイン(VMGM_DOM)
（３）ビデオタイトルセットメニュードメイン(VTSM_DOM)
（４）タイトルドメイン(TT_DOM)
（５）停止状態

なお、（５）の停止状態は、実際にはドメインではない。

【０００５】

（１）の、ファーストプレイドメインは、ディスク最初のセクションを意味し、ＤＶＤプレーヤにおいて再生の準備状態を示す。非再生コマンドが有効とされる。（２）の、ビデオマネージャメニュードメインは、ディスク全体またはディスク面のメインメニューを意味し、タイトルメニューの表示を示す。メニューに関係するコマンドが有効とされる。（３）の、ビデオタイトルセットメニュードメインは、タイトルまたはタイトルのグループのメニュー、あるいは、サブメニュー（サブピクチャ、言語、オーディオまたはアングル）を意味し、ルートメニューまたはサブメニューの何れかの表示を示す。（４）の、タイトルドメインは、タイトル内のビデオコンテンツを意味し、再生コマンドが有効とされる。また、（５）の停止状態は、ヘッドがディスク再生から離れて元の位置に戻っている状態を意味し、再生コマンドが有効とされる。

【０００６】

ＤＶＤプレーヤの動作を制御するナビゲーションコマンドは、現在の状態に適用されるドメインによって制限される。例えば、メニューから所定の項目を選択する「選択ボタン」コマンドは、メニューが表示されている場合に限り、意味がある。すなわち、「選択ボタン」コマンドは、上述の（２）のビデオマネージャメニュードメインまたは（３）のビデオタイトルセットメニュードメインの何れかの状態の場合に、意味をなすコマンドとされる。また例えば、通常の１倍速より高速にビデオ再生を指示する「早送り」コマンドは、プレーヤが停止していたり、静止画像で構成されるメニュー画面の表示中は、意味をなさない。すなわち、「早送り」コマンドは、タイトルドメインにおいて、意味をなすコマンドとされる。

【０００７】

このようなＤＶＤビデオにおけるドメインについては、例えば非特許文献１に記載されている。

【非特許文献１】Jim Taylor，ＤＶＤ解体新書，初版，ネクサスインターコム有限会社，２００３年６月７日，ｐ２７１

【発明の開示】

【発明が解決しようとする課題】

【０００８】

このような従来のＤＶＤビデオにおいては、上述したように、ドメインの種類が多く、ドメイン間の状態遷移の条件も細かいため、ＤＶＤプレーヤに対する実装が難しいという問題点があった。

【０００９】

また、ドメイン間遷移を十分に理解しないとディスク制作ができず、コンテンツ制作者側にとっても、ディスクが作りにくい要因の一つになっていたという問題点があった。すなわち、ドメインの種類が多く、また、ドメイン間遷移の条件が細かいため、ドメイン間遷移を完全に把握することが難しく、特に複雑な構成のディスクを制作する際に、制作者側に多大な負担がかかっていた。

【００１０】

さらに、ドメインの名称と実際の運用の方法とが必ずしも一致せず、ドメインの存在意義に必然性が乏しい面があった。これは、上述のドメインの種類が多いと共に、ドメイン間遷移の条件が細かいことと相まって、ディスク制作者側にディスク制作の負担を増加させる要因となるという問題点があった。

【００１１】

例えば、ＤＶＤビデオでは、上述のように、タイトルドメインと２種類のメニュードメイン（ビデオマネージャメニュードメインおよびビデオタイトルセットメニュードメイン）とが定義されている。本来は、タイトルドメイン中で、ＤＶＤに収録される主たるコンテンツ（例えば映画本編）が再生されるべきであるが、実際には、メニュードメイン中で映画本編を再生しても、問題は生じない。ドメイン間遷移を発生させるコマンドには違い

があるが、一旦これらのドメインに状態遷移してしまえば、メニュードメインおよびタイトルドメインに、プレーヤ動作としての差異が無い。これは、メニュードメインでのメニュー再生およびタイトルドメインでの映画本編の再生共、コンテンツデータとそれに関連する再生制御プログラムをひとまとまりとしたＰＧＣ（プログラムチェイン）と呼ばれる共通の論理構造で実現しているからである。

【００１２】

このことは、却って自由なコンテンツ制作に制約を加えることになることがあるという問題点があった。すなわち、あるコンテンツがメニューであるか、タイトルであるかは、ディスク制作者側の主観に依存するものである。例えば、本編の再生中に分岐がある場合に、分岐の何れかを選択するための選択ボタンが表示されるような、インタラクティブ性を取り入れたコンテンツを考える。この場合、従来のＤＶＤビデオの方法では、この選択ボタンが表示されるコンテンツをメニュードメインおよびタイトルドメインの何れで再生すべきなのかが曖昧である。そのため、例えば、このコンテンツの後にさらにインタラクティブ性を有するコンテンツを用意したい場合などに、状態遷移の予測が立て辛くなり、動作の検証などが難しくなるおそれがあった。

【００１３】

したがって、この発明の目的は、プレーヤ動作の状態遷移を明確に定義し、インタラクティブなコンテンツの制作を容易とするような再生装置、再生方法および再生プログラム、記録媒体、ならびに、データ構造体を提供することにある。

【課題を解決するための手段】

【００１４】

上述した課題を解決するために、請求項１に係る発明は、記録媒体に記録されたコンテンツデータを再生する再生装置において、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体からデータを読み出す読み出し手段と、再生制御プログラムに従いコンテンツデータを再生するプレーヤ手段と、プレーヤ手段に対してユーザ操作に応じた制御コマンドを与える制御コマンド出力手段とを備え、プレーヤ手段は、コンテンツデータを再生しているか否かで分類される２状態と、制御コマンド出力手段からの制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき上記コンテンツデータの再生を制御し、プレーヤ手段の４状態間の状態遷移は、再生制御プログラムによって発生される再生装置である。

【００１５】

また、請求項１１に記載の発明は、記録媒体に記録されたコンテンツデータを再生する再生方法において、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された再生制御プログラムに従いプレーヤ手段でなされる上記コンテンツデータの再生を、プレーヤ手段の、コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御し、プレーヤ手段の４状態間の状態遷移は、再生制御プログラムによって発生される再生方法である。

【００１６】

また、請求項１２に記載の発明は、記録媒体に記録されたコンテンツデータを再生する再生方法をコンピュータ装置に実行させる再生プログラムにおいて、再生方法は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された再生制御プログラムに従いプレーヤ手段でなされるコンテンツデータの再生を、プレーヤ手段の、コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御し、プレーヤ手段の４状態間の状態遷移は、再生制御プログラムによって発生される再生プログラムである。

【００１７】
また、請求項１３に記載の発明は、記録媒体に記録されたコンテンツデータを再生する再生方法をコンピュータ装置に実行させる再生プログラムが記録されたコンピュータ装置に読み取り可能な記録媒体において、再生方法は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された再生制御プログラムに従いプレーヤ手段でなされるコンテンツデータの再生を、プレーヤ手段の、コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御し、プレーヤ手段の４状態間の状態遷移は、再生制御プログラムによって発生されるコンピュータ装置に読み取り可能な記録媒体である。
【００１８】
また、請求項１４に記載の発明は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生をプレーヤ手段に制御させる再生制御プログラムとが記録された、コンピュータ装置に読み取り可能な記録媒体であって、再生制御プログラムは、コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づきコンテンツデータの再生を制御するプレーヤ手段に対して再生制御命令を与えてコンテンツの再生を制御させ、さらに、プレーヤ手段に対して４状態間の状態遷移を発生させるようにしたコンピュータ装置に読み取り可能な記録媒体である。
【００１９】
また、請求項１６に記載の発明は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生をプレーヤ手段に制御させる再生制御プログラムとを含むデータ構造体であって、再生制御プログラムは、コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づきコンテンツデータの再生を制御するプレーヤ手段に対して再生制御命令を与えてコンテンツの再生を制御させ、さらに、プレーヤ手段に対して４状態間の状態遷移を発生させるデータ構造体である。
【００２０】
上述したように、請求項１、１１、１２および１３に記載の発明は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生を制御する再生制御プログラムとが記録された記録媒体から読み出された再生制御プログラムに従いプレーヤ手段でなされるコンテンツデータの再生を、プレーヤ手段の、コンテンツデータを再生しているか否かで分類される２状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づき制御するようにしているため、プレーヤ手段の状態の数が少なく状態遷移の際の動作が理解し易くなり、プレーヤ手段の実装が容易になると共に、コンテンツ制作の負担が軽減される。
【００２１】
また、請求項１４に記載の発明は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生をプレーヤ手段に制御させる再生制御プログラムとが記録された、コンピュータ装置に読み取り可能な記録媒体であって、再生制御プログラムは、コンテンツデータを再生しているか否かで分類される２ 状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２状態との組み合わせにより定義される４状態に基づきコンテンツデータの再生を制御するプレーヤ手段に対して再生制御命令を与えてコンテンツの再生を制御させるようにしているため、複雑な制御を要するコンテンツでも容易に制作することができる。
【００２２】

また、請求項１６に記載の発明は、少なくともビデオストリームおよびオーディオストリームのうち何れか一方を含むコンテンツデータと、コンテンツデータの再生をプレーヤ手段に制御させる再生制御プログラムとを含むデータ構造体であって、再生制御プログラムは、コンテンツデータを再生しているか否かで分類される２ 状態と、ユーザ操作に応じた制御コマンドを受け付けるか否かで分類される２ 状態との組み合わせにより定義される４状態に基づきコンテンツデータの再生を制御するプレーヤ手段に対して再生制御命令を与えてコンテンツの再生を制御させるようにしているため、複雑な制御を要するコンテンツでも容易に制作することができ、データ構造として提供できる。

【発明の効果】

【００２３】

この発明は、プレイリストを再生するムービープレーヤの状態として、プレイリスト再生の観点からストップ状態およびプレイ状態の２状態を定義すると共に、スクリプトプログラムを介さない、ユーザ操作に応じた制御コマンドを受け付けるか否かで、ノーマルモードおよびメニューモードの２状態を定義し、これら４状態間の状態遷移でプレイリストの再生を制御するようにしている。そのため、ＡＶストリームの再生をスクリプトプログラムから制御できるようになるという効果がある。

【００２４】

また、ムービープレーヤの状態の数が少なく、状態の定義も明快なため、状態遷移が発生する条件や状態遷移が発生するときの動作が理解し易く、ＡＶストリームの再生を行うプレーヤの実装が容易になるという効果がある。

【００２５】

さらに、ムービープレーヤの状態の数が少なく、状態の定義が明快であることから、状態遷移が発生する条件や状態遷移が発生するときの動作が理解し易いため、コンテンツ制作者側にとっても、インタラクティブ性を有するコンテンツ制作をより容易に行うことができるようになるという効果がある。

【発明を実施するための最良の形態】

【００２６】

以下、この発明の実施の一形態について、下記の順序に従い説明する。

１．ＵＭＤビデオ規格について
２．ＵＭＤビデオ規格のプレーヤモデルについて
３．ムービープレーヤのイベントモデルについて
４．ムービープレーヤオブジェクトについて
５．スクリプトプログラムの例
６．ファイルの管理構造について
７．ディスク再生装置について
８．ムービープレーヤの状態遷移モデルについて
　８－１．ムービープレーヤの状態の定義について
　８－２．ムービープレーヤに状態遷移を発生させるメソッドについて
　８－３．プレイリスト再生中のムービープレーヤの動作について
　８－４．ムービープレーヤの再生復帰機能について
　８－５．各データのライフサイクルについて

【００２７】

１．ＵＭＤビデオ規格について

先ず、理解を容易とするために、この実施の一形態に適用可能なシステムについて概略的に説明する。この発明の実施の一形態では、ＥＣＭＡスクリプトと呼ばれるスクリプト言語を用いてプレーヤモデルを記述している。ＥＣＭＡスクリプトは、ＥＣＭＡ(European Computer Manufacturers Association)により定められた、ＪａｖａＳｃｒｉｐｔ（登録商標）に基づいたクロスプラットフォーム用のスクリプト言語である。ＥＣＭＡスクリプトは、ＨＴＭＬ文書との親和性が高いことと、独自のオブジェクトの定義が可能であるため、この発明によるプレーヤモデルに用いて好適である。

【００２８】
すなわち、従来からのＤＶＤビデオでは、インタラクティブな機能を実現するための制御プログラムを記述するために、ＤＶＤビデオ規格で定義された汎用的ではないコマンドを用いていた。制御プログラムは、複数のファイルやデータファイルの中の複数の箇所、さらには、ＡＶストリームファイル中に分散されて埋め込むようにされ、埋め込まれた制御プログラムが実行される条件や順序は、ＤＶＤ規格で決められていた。
【００２９】
このようなＤＶＤビデオのシステムでは、汎用的なコンテンツ制作システムを構築することが難しく、そのため、予め決められた脚本に当てはめてストーリーを作る、テンプレートを用いたコンテンツ制作を行っていた。そして、テンプレートでは対応できない、複雑なコンテンツを制作する場合には、先ず、コンテンツ制作システムそのものをカスタムメイドで作成していた。この発明の実施の一形態では、ＡＶコンテンツを制御するために、汎用的で拡張性の高いスクリプト言語であるＥＣＭＡスクリプトを用いることで、このような問題を解決している。
【００３０】
以下では、このＥＣＭＡスクリプトを元にしたスクリプト言語を用いた、この発明の実施の一形態に基づく規格を、ＵＭＤ(Universal Media Disc：登録商標)ビデオ規格と呼ぶ。また、ＵＭＤビデオ規格のうち、特にスクリプトに関する部分をＵＭＤビデオスクリプト規格と呼ぶ。
【００３１】
ＵＭＤビデオ規格について、概略的に説明する。図１は、ＵＭＤビデオ規格のレイヤ構成を示す。ＵＭＤビデオ規格では、スクリプトレイヤ、プレイリストレイヤおよびクリップレイヤの３層のレイヤ構造が定義され、この構造に基づきストリーム管理がなされる。
【００３２】
ＵＭＤビデオ規格においては、ディジタル符号化されたビデオ、オーディオおよび字幕を、ＭＰＥＧ２(Moving Pictures Experts Group 2)のパケッタイズドエレメンタリストリームとして多重化したＭＰＥＧ２ストリームとして扱う。このビデオ、オーディオおよび字幕のエレメンタリストリームが多重化されたＭＰＥＧ２ストリームを、クリップＡＶストリーム(Clip AV Stream)と呼ぶ。クリップＡＶストリームは、クリップＡＶストリームファイルに格納される。クリップＡＶストリームファイルの記録時に、当該クリップＡＶストリームファイルに１対１に対応して、クリップインフォメーションファイル(Clip Information File)が同時に作成される。これらクリップインフォメーションファイルと、対応するクリップＡＶストリームファイルとからなる組を、クリップ(Clip)と呼ぶ。
【００３３】
クリップは、ディスクへの記録の単位ともいうべきものであり、再生時にどのような順序でクリップを再生するかは、クリップの上位のレイヤであるプレイリストレイヤで管理する。プレイリストレイヤは、クリップの再生経路を指定するレイヤであり、１または複数のプレイリスト(PlayList)を含む。プレイリストは、プレイアイテムの(PlayItem)の集合からなる。プレイアイテムには、クリップの再生範囲を示した一組のイン(In)点およびアウト(Out)点が含まれており、プレイアイテムを連ねることによって、任意の順序でクリップを再生することができるようになる。プレイアイテムは、クリップを重複して指定することができる。クリップＡＶストリームファイルのイン点およびアウト点は、タイムスタンプ（クリップ内時刻）で指定され、タイムスタンプは、クリップインフォメーションファイルの情報によってクリップＡＶストリーム上のバイト位置に変換される。
【００３４】
プレイリストは、クリップの全部または一部を指すプレイアイテムを順序に従って再生していく構造だけを有しており、プレイリストのみを用いて再生順の分岐や、ユーザとの双方向性を実現することは、できない。この発明の実施の一形態では、複数のプレイリストが１つのファイル"PLAYLIST.DAT"にまとめられている。
【００３５】

スクリプトレイヤは、言語仕様のＥＣＭＡスクリプトを拡張した、ＵＭＤビデオスクリプトによって構築されるレイヤである。ＵＭＤビデオスクリプトは、ＥＣＭＡスクリプトを基本として、ＵＭＤビデオに特有な機能を実現するための拡張を加えたスクリプトである。

【００３６】

スクリプトレイヤは、プレイリストレイヤの上位のレイヤであり、プレイリストの再生指示や、プレーヤ設定を行うコマンド列から構成される。スクリプトレイヤのコマンドにより、複数の言語用に用意されたストリームのうち何れを選択する、ある条件に従って選択されるプレイリストに再生の流れが変化する、というような、条件分岐を伴うプレイリスト再生を実現することができる。このような条件分岐を伴うプレイリスト再生が用いられるアプリケーションの例としては、マルチストーリーが挙げられる。このスクリプトレイヤにより、ユーザとの双方向性機能（インタラクティブ機能）が導入されることになる。

【００３７】

なお、この発明の実施の一形態では、スクリプトレイヤは、リソースファイルと呼ばれるファイルにより構成される。リソースファイルは、実際のＥＣＭＡスクリプトに基づき記述されるスクリプトデータ（スクリプトプログラム）、ボタン操作の際の効果音などを出力するためのサウンドデータ、メニュー画面の背景画像などに用いる画像データからなるスクリーンデザイン、ならびに、ボタン画像などのＧＵＩ部品を表示させるための画像データ（ビットマップデータ）が含まれる。

【００３８】

リソースファイルは、複数、存在することができる。また、この実施の一形態においては、リソースファイルは、所定の命名規則に従いファイル名が与えられ、例えば、ファイル名の拡張子「RCO」により当該ファイルがリソースファイルであることが示される。

【００３９】

２．ＵＭＤビデオ規格のプレーヤモデルについて

次に、ＵＭＤビデオ規格に従ったデータを再生する再生装置（プレーヤ）のモデル、すなわち、プレーヤモデルについて説明する。プレーヤは、先ず、ディスクからリソースファイル、プレイリストフィルおよびクリップインフォメーションファイルを読み出し、次に、これらにより定められている再生順序に従って、クリップＡＶストリームファイルを読み出し、ビデオ、オーディオおよび字幕などを再生する。

【００４０】

スクリプトプログラムの言語仕様においては、プレイリストを再生する機能ブロックを、スクリプトプログラム内のオブジェクトとして実装する。このプレイリスト再生を行うオブジェクトを、ＵＭＤビデオ規格では、ムービープレーヤ(Movie Player)オブジェクトと呼ぶ。プレイリストの再生指示や、プレーヤ設定を行うコマンドは、このムービープレーヤオブジェクトが有するメソッドとなる。ムービープレーヤオブジェクトは、スクリプトレイヤからのメソッドによって制御される。このとき、ムービープレーヤオブジェクトからスクリプトレイヤに対して、状態の変化や再生位置などを通知する機能が必要となる。これは、ムービープレーヤオブジェクトがスクリプトプログラムに対してイベントを発することに対応し、そのイベントに対応した処理は、イベントハンドラとして記述される。

【００４１】

このように、ムービープレーヤオブジェクトからスクリプトプログラムへの情報伝達は、イベントにより行い、スクリプトプログラムからムービープレーヤオブジェクトに対する制御をメソッドにより行うモデルを構築することにより、クリップＡＶストリームの再生をスクリプトプログラムで制御できるようになる。

【００４２】

図２は、上述した、この発明の実施の一形態による一例のプレーヤモデルを模式的に示す。ムービープレーヤ３００は、ＵＭＤビデオ規格においてビデオ、オーディオおよび字

幕の再生を司るモジュールである。上述したムービープレーヤオブジェクトは、ムービーオブジェクトをスクリプトプログラムから操作するためにスクリプトプログラム内のオブジェクトとしたものである。換言すれば、ムービープレーヤオブジェクトは、ムービープレーヤの機能を実現する実装モジュールを抽象化してスクリプトプログラム上で扱える形にしたものである。

【００４３】

なお、ムービープレーヤ３００とムービープレーヤオブジェクトは、実質的に同一の対象を表すと考えられるので、以下、これらを同一の符号を付して説明する。

【００４４】

図２において、ムービプレーヤ３００は、ユーザ入力３１０などにより引き起こされる下位レイヤ（図２の例ではネイティブ実装プラットフォーム３０１）や、上位レイヤであるスクリプトレイヤ３０２からのメソッドに従って、プレイリストおよびクリップインフォメーションのデータベースに基づき、クリップＡＶストリームファイルの読み出し、読み出されたクリップＡＶストリームのデコードおよび表示を行う。

【００４５】

ムービープレーヤオブジェクト３００の内部は、ＵＭＤビデオを再生するＵＭＤビデオプレーヤの実装に依存するものであって、スクリプトレイヤ３０２からは、ブラックボックス化されたオブジェクトとして、メソッドやプロパティといったＡＰＩ(Application Programming Interface)が提供される。ここで、ＵＭＤビデオプレーヤ(UMD Video Player)は、ムービープレーヤを実装した実際の機器を指す。全てのＵＭＤビデオプレーヤは、ＵＭＤビデオ規格の制約を守ってムービープレーヤを実装しており、再生互換を有する。

【００４６】

図２に示されるように、ムービープレーヤオブジェクト３００は、ネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付けるパス、スクリプトレイヤ３０２に対してイベント３１２を通知するパス、スクリプトレイヤ３０２からのメソッド３１３を受け付けるパスの、３本の入出力パスを有する。

【００４７】

制御コマンド３１１は、ネイティブ実装のプラットフォーム３０１からの、ムービープレーヤオブジェクト３００の動作を制御するコマンドである。ネイティブ実装プラットフォーム３０１は、例えば、実際の機器としてのＵＭＤビデオプレーヤにおける、機器に固有の部分とムービープレーヤ３００とのインターフェイスである。イベント３１２は、ムービープレーヤ３００からスクリプトレイヤ３０２に対するスクリプトイベントである。メソッド３１３は、スクリプトレイヤ３０２のスクリプトプログラムからムービープレーヤ３００に指示されるメソッドである。

【００４８】

ムービープレーヤオブジェクト３００は、内部に、ＵＭＤビデオ規格のプレイリストおよびクリップインフォメーションのデータベース３２０を有する。ムービープレーヤオブジェクト３００は、このデータベース３２０を用いて、ユーザ入力３１０に対する無効化(mask)や、時刻で指定された再生位置をクリップＡＶストリーム内のバイト位置に変換する処理などを行う。

【００４９】

ムービープレーヤオブジェクト３００内のプレイバックモジュール３２１は、ビデオ、オーディオおよび字幕が多重されたＭＰＥＧ２　ＰＳ(Program Stream)であるクリップＡＶストリームのデコードを行う。プレイバックモジュール３２１は、プレイおよびストップの２状態を持ち、制御命令やメソッドによって、この２状態の間を遷移する（図３参照）。なお、クリップＡＶストリームは、ＭＰＥＧ２　ＰＳに限られない。例えば、クリップＡＶストリームとしてＭＰＥＧ２　ＴＳ(Transport Stream)を用いても、モデル上は同様に扱うことができる。

【００５０】

スクリプトレイヤ３０２は、ＵＭＤビデオスクリプト規格に基づくスクリプトプログラ

ムを実行し、ムービープレーヤオブジェクト３００の制御や、画面表示を行うレイヤである。このスクリプトレイヤ３０２は、コンテンツ制作者側の意図したシナリオを実現する役割を果たす。スクリプトレイヤ３０２は、ムービープレーヤオブジェクト３００に対してメソッド３１３を発行し、ムービープレーヤオブジェクト３００からは、イベント３１２を受け取る。スクリプトレイヤ３０２は、ネイティブ実装プラットフォーム３０１との間で、ユーザ入力３１０に応じたキーイベント３１４や、画面描画などをネイティブ実装プラットフォーム３０１に対して指示するメソッド３１５などのやりとりを行う。

【００５１】

なお、ネイティブ実装プラットフォーム３０１は、ＵＭＤビデオ規格で規定した以外の様々な機能を有する。この発明の実施の一形態では、、スクリプトレイヤ３０２からネイティブ実装プラットフォーム３０１に働きかけるメソッド３１５が存在するため、ネイティブ実装プラットフォーム３０１にも、機能を抽象化したオブジェクトを定義し、メソッド３１５は、スクリプトプログラム上では、このオブジェクトに属するものと見なす。これは、メソッドは、オブジェクトに属するからである。そこで、ネイティブ実装プラットフォーム３０１内にコントロールオブジェクト３３０を定義し、メソッド３１５は、コントロールオブジェクト３３０のメソッドであると定める。

【００５２】

例えば、メニュー画面上に配置されるボタンは、スクリプトレイヤ３０２のスクリプトプログラムからネイティブ実装プラットフォーム３０１に渡されるメソッド３１５に基づき、ネイティブ実装プラットフォーム３０１により描画される。ユーザがそのボタンに対して選択や決定などの操作を行ったときには、ユーザ入力３１０に応じたキーイベント３１４がネイティブ実装プラットフォーム３０１からスクリプトレイヤ３０２に通知され、スクリプトレイヤ３０２内のスクリプトプログラムは、キーイベント３１４に基づきキー入力３１０に応じた処理を行う。

【００５３】

このように、ビデオ、オーディオおよび字幕のデコードや表示制御はムービープレーヤ３００が司り、ボタンなどのＧＵＩ(Graphical User Interface)を構成するための部品画像（以下、ＧＵＩ部品と呼ぶ）の配置や表示、ならびに、ＧＵＩ部品に対して選択や決定などの操作がなされたときの処理は、スクリプトレイヤ３０２が行うというように、役割分担がなされている。

【００５４】

ネイティブ実装プラットフォーム３０１は、ムービープレーヤオブジェクト３００やスクリプトプログラムが動作するための基盤となるプラットフォームであって、例えば、実際のＵＭＤビデオプレーヤがハードウェアである場合、ハードウェアとプレーヤモデルとの間の処理を仲介する役割を果たすように、ハードウェアに固有に実装される。

【００５５】

例えば、ネイティブ実装プラットフォーム３０１は、ユーザからのユーザ入力３１０を受け付け、受け付けたユーザ入力３１０がムービープレーヤ３００に対する命令なのか、スクリプトレイヤ３０２で描画および表示しているボタンに対する命令なのかを判定する。ネイティブ実装プラットフォーム３０１は、ユーザ入力３１０がムービープレーヤ３００に対する命令であると判定されれば、ユーザ入力３１０をムービープレーヤ３００に対する内部制御命令である制御コマンド３１１に変換し、ムービープレーヤ３００に対して制御命令を発する。

【００５６】

一方、ネイティブ実装プラットフォーム３０１は、ユーザ入力３１０がスクリプトレイヤ３０２で描画および表示しているＧＵＩ部品に対する命令であると判定されれば、ユーザ入力３１０に応じたキーイベント３１４をスクリプトレイヤ３０２に通知する。そして、このキーイベント３１４に応じてスクリプトレイヤ３０２から指示されたメソッド３１５に基づき、例えば画面上にボタン画像を表示させることができる。すなわち、ネイティブ実装プラットフォーム３０１とスクリプトレイヤ３０２とは、ムービープレーヤ３００

を介さずに、直接的にイベントおよびメソッドの受け渡しを行うことができる。

【００５７】

また例えば、ネイティブ実装プラットフォーム３０１は、後述するような、ムービープレーヤ３００のプロパティにアクセスし、ムービープレーヤ３００のステータスを見ることができるようにされている。

【００５８】

次に、ムービープレーヤ３００についてより詳細に説明する。図３は、ムービープレーヤ３００の一例の内部構成を示す。上述したように、ムービープレーヤ３００は、データベース３２０およびプレイバックモジュール３２１とから構成される。データベース３２０は、ディスクから読み取ったプレイリストの情報と、クリップの情報すなわちクリップインフォメーションとを格納する領域である。

【００５９】

プレイバックモジュール３２１は、デコーダエンジン３２２と、プレイバックモジュール３２１の状態を表す値であるプロパティ３２３とからなる。プロパティ３２３は、例えば言語コードのように、ムービープレーヤ３００の初期設定で値が決まるプロパティ３２３Ａ（リードオンリーパラメータ）と、プレイバックモジュール３２１の状態によって値が変化するプロパティ３２３Ｂ（プレーヤステータス）の２種類がある。

【００６０】

初期設定で値が決まるプロパティ３２３Ａは、ネイティブなシステム、例えば実際の機器によって値がセットされ、プレイリストやクリップインフォメーション、スクリプトプログラムから値を変更されることがない。プロパティ３２３Ａは、スクリプトプログラムから値を読み出すことのみが可能とされる。一方、プレイバックモジュール３２１の状態を表すプロパティ３２３Ｂは、スクリプトプログラムから値を読み出すことができると共に、一部のスクリプトプログラムから値を書き込むことが可能とされる。

【００６１】

なお、この動作モデルにおいては、プレイリストおよびクリップインフォメーションは、クリップＡＶストリームの再生前にディスクからプリロードされていることを想定している。これに限らず、他の実装であっても、ムービープレーヤモデルで定めた動作を実現できていればよい。

【００６２】

ムービープレーヤオブジェクト３００は、スクリプトレイヤ３０２またはネイティブ実装プラットフォーム３０１からの指示に従い、指定されたプレイリストを再生する。例えば、ムービープレーヤ３００は、データベース３２０を参照し、指定されたプレイリストに対応するクリップＡＶストリームの再生位置をファイル中のバイト位置で得る。プレイバックモジュール３２１において、デコーダエンジン３２２は、この再生位置情報に基づき、クリップＡＶストリームのデコードを制御する。

【００６３】

ムービープレーヤ３００は、図４に示されるように、プレイリストの再生状況に応じてプレイ(play)およびストップ(stop)の２状態を持つ。プレイ状態は、プレイリストが指定され、プレイリストの再生を行っている状態を指す。通常再生の他、２倍速、１／２倍速といった変速再生や、順方向早送りおよび逆方向早送り、一時停止（ポーズ）といった各種の状態(status)がプレイ状態に含まれる。再生をフレーム単位で進めたり戻したりする所謂コマ送り再生は、ポーズ状態とプレイ状態とを繰り返している状態である。ストップ状態は、プレイリストを再生していない状態である。ストップ状態においては、プレイリストが選択されておらず、「現在再生中のプレイリスト番号」を表すプレーヤステータスの値は、不定である。

【００６４】

例えば、ムービープレーヤ３００の状態は、ムービープレーヤ３００内のデコーダエンジン３２２におけるプレイおよびストップの状態遷移に伴うもので、デコーダエンジン３２２の状態遷移に応じてプロパティ３２３Ｂの値が更新される。

【００６５】
リジュームインフォメーション３２４は、ストップ状態直前の状態を記憶する。例えば、ムービープレーヤ３００があるプレイリストをデコードしプレイ状態となっているときに、状態がストップ状態に遷移すると、ストップ状態の直前の状態を記憶する。また、リジュームインフォメーション３２４は、ハードウェアとしてのプレーヤが持つ不揮発性メモリに、ディスクのタイトル毎に識別可能なように複数を記憶させることができる。例えば、ディスクは、ディスクのタイトル毎にユニークな識別情報（タイトルＩＤと呼ぶ）を有し、リジュームインフォメーション３２４をこの識別情報と関連付けて記憶する。こうすることで、リジュームインフォメーション３２４の情報に基づき、識別情報が対応するタイトルのディスクが次にストップ状態からプレイ状態に遷移したときに、当該ディスクの再生を、以前ストップ状態になった直前の位置から開始させることができる。
【００６６】
３．ムービープレーヤのイベントモデルについて
ムービープレーヤ３００のイベントモデルについて説明する。ムービープレーヤ３００は、プレイリストを再生するプレイ状態で、様々なイベントを発生する。このイベントは、イベントハンドラと呼ばれる、スクリプトで記述された処理プログラムの実行を引き起こす。イベントハンドラは、イベント発生によって呼び出されるメソッドである。このイベント発生によって処理プログラムの実行を開始するプログラム実行モデルを、イベントドリブンモデルと呼ぶ。イベントドリブンモデルでは、不定期なイベントが発生し、イベント発生をきっかけに用意しておいたプログラムが実行される。この実施の一形態においては、スクリプトプログラムは、イベントハンドラ群によってムービープレーヤオブジェクト３００の動作を制御する。
【００６７】
図５は、この発明の実施の一形態によるムービープレーヤ３００のイベントモデルを模式的に示す。図５において、イベントハンドラonEventA()、onEventB()およびonEventC()は、インターフェイスであって、それぞれのイベントハンドラの内容は、スクリプトで記述されている。イベントハンドラの内容は、例えばコンテンツ制作者側で作成され実装される。ＵＭＤビデオスクリプト規格においては、ムービープレーヤオブジェクト３００からスクリプトプログラムに通知されるイベント毎に、イベントハンドラが用意される。図５の例では、イベントＡが発生したときに実行される処理プログラムは、イベントハンドラonEventA()に決められている。イベントＢおよびイベントＣについても同様に、イベントＢの発生時には対応するイベントハンドラonEventB()が実行され、イベントＣの発生時には対応するイベントハンドラonEventC()が実行される。
【００６８】
イベント発生に応じて呼び出されるイベントハンドラは、システム側で選択されるため、コンテンツ制作者側では、どのイベントが発生したかを判断する処理を、スクリプトプログラム内に記述しておく必要が無い。
【００６９】
図６は、プレイリストの再生中に発生する一例のイベントを示す。プレイリストPlayListの先頭には、チャプタマークChapterMarkが設定されているので、プレイリストの先頭からの再生開始時には、先ず、チャプタマークに対応したイベントChapterが発生する。さらに、チャプタが変わる度にイベントChapterがスクリプトレイヤ３０２に通知され、対応するイベントハンドラonChapterが実行される。また、イベントマークEventMarkが設定されている時刻に再生が到達すると、対応するマークイベントが発生する。そして、プレイリストの最後まで再生が到達すると、プレイリストの最後で再生が一時停止し、イベントPlayListEndがムービープレーヤ３００からスクリプトレイヤ３０２に通知される。スクリプトレイヤ３０２側では、対応するイベントハンドラonPlayListEnd()内で、別のプレイリストの再生開始が指示される。このようにして、コンテンツ制作者が意図した順序で、一連のプレイリスト再生が継続されていく。
【００７０】

このように、プレーヤの動作中にはさまざまなイベントが発生するものとし、イベント発生を上位プログラムに伝えることで、上位プログラムはプレーヤの状態を把握できるようになる。上位プログラムの方では、各イベント発生通知時に実行されるプログラム（イベントハンドラ）を用意しておくことで、各種イベント発生に対処する。イベントおよびイベントハンドラの詳細については、後述する。

【００７１】

イベントハンドラがコンテンツ制作者によって記述されていない場合には、規格で規定されているプレーヤ組み込みの動作（デフォルトのイベントハンドラ）を実行するか、あるいは、そのイベントが無視され何も実行されない。何も処理を行う必要がないときには、イベントに対応したイベントハンドラを記述しないようにすることで、積極的にイベントを無視することができる。

【００７２】

イベントモデルとしては、上述の他に、あるイベントに対応するリスナをオブジェクトがプレーヤオブジェクトに登録し、プレーヤオブジェクト内で発生したイベントが登録されたイベントであれば、プレーヤオブジェクトから当該イベントを登録したオブジェクトにイベントを送信し、当該オブジェクトで対応するメソッドを実行するようにしたイベントリスナのモデルや、どのようなイベントが発生しても一つのメソッドを呼び出すようにした単一メソッドのモデルなどが考えられる。

【００７３】

この実施の一形態によるイベントモデルは、イベント登録、イベント登録削除といった処理が必要なイベントリスナのモデルよりも簡単である。また、単一メソッドのモデルは、どのイベントが発生したかを知り、イベント毎に用意してある処理ルーチンを切り替えるという前処理を、そのメソッドの中に記述しておく必要がある。メソッドは、コンテンツ制作者側が実装するものであるから、モデルとしては簡単でも、コンテンツ制作者側の負担が大きくなる。さらに、大きな一つの処理プログラム（メソッド）がイベントの発生毎に呼ばれることになり、メモリの領域を多く占有し、実行速度も遅くなると考えられる。この発明の実施の一形態による、イベント毎に処理プログラム（イベントハンドラ）を用意するモデルでは、このような点について有利であるといえる。

【００７４】

４．ムービープレーヤオブジェクトについて

次に、ムービープレーヤオブジェクト３００の外部的な仕様について説明する。一般に、ＥＣＭＡスクリプト言語仕様に従う言語により定義されたオブジェクトは、プロパティとメソッドとを持つ。この実施の一形態によるムービープレーヤオブジェクト３００も、図２および図３を用いて既に説明したように、同様にしてプロパティとメソッドとを有する。プロパティは、外部のオブジェクトから、対象となるオブジェクト名とプロパティ名とを指定することで、直接的に読み書きすることが可能である。これに限らず、プロパティ値の設定を行うメソッドsetXXX()（「XXX」は、対象のプロパティ名）や、プロパティ値の読み出しを行うメソッドgetXXX()を定義することで、他のオブジェクトのプロパティの読み書きを、メソッドで行うことが可能となる。

【００７５】

図７は、ムービープレーヤオブジェクト３００が有する一例のプロパティを一覧して示す。これは、図３におけるプロパティ３２３に対応する。図７Ａは、図３におけるリードオンリーパラメータ３２３Ａに属する一例のプロパティを示す。プロパティscriptVersionは、ＵＭＤビデオスクリプト(UMD Video Script)のバージョンを示す。プロパティaudioChannelCapabilityは、ＵＭＤビデオプレーヤが再生可能なオーディオチャンネル数を示す。プロパティlanguageCodeは、ＵＭＤビデオプレーヤに設定された、メニュー表示言語の言語コードを示す。プロパティaudioLanguageCodeは、ＵＭＤビデオプレーヤに設定された、オーディオ言語の言語コードを示す。プロパティsubtitleLanguageCodeは、ＵＭＤビデオプレーヤに設定された、字幕（サブタイトル）言語の言語コードを示す。

【００７６】

ディスクが装填された際には、このリードオンリーパラメータ３２３Ａに設定されたプロパティlanguageCodeに示される言語コードに基づき、ディスクから読み出すスクリプトファイルが決められる。装填されたディスクに、当該言語に対応するスクリプトファイルがない場合は、デフォルトのスクリプトファイルが読み出される。例えば、複数のスクリプトファイルのうち、ディスク上で最も先頭側に配置されるファイルがデフォルトのスクリプトファイルとして読み出される。

【００７７】

図７Ｂは、図３におけるプレーヤステータス３２３Ｂに属する一例のプロパティを示す。プロパティplayListNumberは、現在再生中のプレイリストの番号を示す。プロパティchapterNumberは、現在再生中のチャプタの番号を示す。プロパティvideoNumberは、現在再生中のビデオストリームの番号を示す。プロパティaudioNumberは、現在再生中のオーディオストリームの番号を示す。プロパティsubtitleNumberは、現在再生中の字幕ストリームの番号を示す。プロパティplayListTimeは、プレイリスト先頭を０としたときの時刻を示す。プロパティaudioFlagは、オーディオ再生のＯＮ／ＯＦＦおよびデュアルモノＬＲの指定を示す。プロパティsubtitleFlagは、字幕表示のＯＮ／ＯＦＦを示す。

【００７８】

なお、デュアルモノ(dual mono)は、ステレオオーディオの左右（Ｌ、Ｒ）チャンネルを、互いに独立したモノラルオーディオチャンネルとして用いるモードである。

【００７９】

このプレーヤステータス３２３Ｂに属する各プロパティは、ムービープレーヤ３００が再生または一時停止状態のときに、これらの情報が存在する。停止状態に遷移した場合、その時点でプレーヤステータス３２３Ｂに属する各プロパティは、リジュームインフォメーション(resumeInformation)３２４としてバックアップされる。このとき、プレーヤステータス３２３Ｂの内容をクリアしてもよい。

【００８０】

図８は、ムービープレーヤオブジェクト３００が有する一例のメソッドを一覧して示す。これは、図２におけるメソッド３１３に対応する。メソッドplay()は、ビデオを再生する。メソッドplayChapter()は、チャプタを指定してビデオを再生する。メソッドresume()は、リジュームインフォメーション３２４を用いた再生開始を行う。メソッドstop()は、ビデオの再生を停止する。メソッドpause()は、ビデオの再生を一時停止する。メソッドplayStep()は、ビデオをコマ送り再生する。メソッドchangeStream()は、ビデオストリーム、オーディオストリームおよび／または字幕ストリームを変更する。メソッドgetPlayerStatus()は、ムービープレーヤ３００における再生、停止、一時停止などの状態を取得する。メソッドchangeResumeInfo()は、リジュームインフォメーション３２４の内容を変更する。メソッドreset()は、ビデオの再生を停止し、リジュームインフォメーション３２４の内容をクリアする。

【００８１】

ＵＭＤビデオ規格では、表示画面上の一部分にビデオを表示することができるようになっている。以下の４つのメソッドは、この場合のビデオ表示に関するメソッドである。メソッドsetPos()は、ビデオの表示位置を設定する。メソッドgetPos()は、ビデオの表示位置を取得する。メソッドsetSize()は、ビデオの表示サイズを設定する。メソッドgetSize()は、ビデオの表示サイズを取得する。

【００８２】

なお、実際には、ムービープレーヤ３００とネイティブ実装プラットフォーム３０１とは、一体的に構成される。すなわち、実際にディスクが装填され、これを再生するハードウェアとしてのＵＭＤプレーヤと、ＵＭＤプレーヤを制御するソフトウェアとの関係に対応付けられ、どの部分をハードウェアで行い、どの部分をソフトウェアで行うかは、実装時の構成に依存する。例えば、ＵＭＤプレーヤをパーソナルコンピュータなどで構成する場合は、ディスクドライブ以外は、ソフトウェア的に構成することができる。また、単体のＵＭＤプレーヤとして構成する場合は、ディスクドライブ以外に、例えばビデオデコー

ダやオーディオデコーダなどをハードウェア的に構成することができる。したがって、ムービープレーヤ３００とネイティブ実装プラットフォーム３０１との間でなされるメソッドやコマンド、イベントは、図２に一例が示されるような明示的なやりとりに限られない。

【００８３】

一方、ユーザからのキー入力については、図２を用いて既に説明したように、ユーザ入力３１０をネイティブ実装プラットフォーム３０１が先ず、受け取る。つまり、ネイティブ実装プラットフォーム３０１は、ユーザからのキー入力をユーザ入力３１０として受け取り、ユーザ入力３１０がムービープレーヤ３００に対するコマンドなのか、スクリプトレイヤ３０２のスクリプトプログラムに対するイベントなのかを判定し、判定結果に応じて、制御コマンド３１１またはキーイベント３１４を発生し、対応する上位レイヤ（ムービープレーヤ３００またはスクリプトレイヤ３０２）に通知する。

【００８４】

図９および図１０は、ユーザ入力３１０による一例のキー入力を示す。なお、図９および図１０に示される「VK」で始まる各キーは、仮想的なキー(Virtual Key)であることを意味する。

【００８５】

図９は、ムービープレーヤ３００の操作に関する一例のキー入力を示す。キーVK_PLAYは、再生を指示する再生キーに対応する機能を提供する。キーVK_STOPは、再生の停止を指示する停止キーに対応する機能を提供する。キーVK_PAUSEは、再生の一時停止を指示する一時停止キーに対応する機能を提供する。キーVK_FAST_FORWARDは、早送り再生を指示する早送りキーに対応する機能を提供する。キーVK_FAST_REVERSEは、早戻し再生を指示する早戻しキーに対応する機能を提供する。キーVK_SLOW_FORWARDは、順方向のスロー再生を指示するスロー（順方向）キーに対応する機能を提供する。キーVK_SLOW_REVERSEは、逆方向のスロー再生を指示するスロー（逆方向）キーに対応する機能を提供する。キーVK_STEP_FORWARDは、順方向のコマ送り再生を指示するコマ送り（順方向）キーに対応する機能を提供する。キーVK_STEP_REVERSEは、逆方向のコマ送り再生を指示するコマ送り（逆方向）キーに対応する機能を提供する。

【００８６】

キーVK_NEXTは、「次」を意味する値を入力する次指定キーに対応する機能を提供する。キーVK_PREVIOUSは、「前」を意味する値を入力する前指定キーに対応する機能を提供する。例えば、キーVK_NEXTおよびキーVK_PREVIOUSを用いて、前後のチャプタへの移動を指示することができる。

【００８７】

キーVK_ANGLEは、マルチアングルのビデオに対するアングル切り替えを指示するアングル切り替えキーに対応する機能を提供する。キーVK_SUBTITLEは、英語字幕、日本語字幕、字幕表示／非表示などを切り替える字幕切り替えキーに対応する機能を提供する。キーVK_AUDIOは、サラウンドやバイリンガルなどオーディオ設定を切り替えるオーディオ切り替えに対応する機能を提供する。キーVK_VIDEO_ASPECTは、ビデオのアスペクト比切り替えを指示するアスペクト切り替えキーに対応する機能を提供する。

【００８８】

図１０は、メニュー操作に関する一例のキー入力を示す。キーVK_UPは、「上」を意味する値を入力する上方向指定キーに対応する機能を提供する。キーVK_DOWNは、「下」を意味する値を入力する下方向指定キーに対応する機能を提供する。キーVK_RIGHTは、「右」を意味する値を入力する右方向指定キーに対応する機能を提供する。キーVK_LEFTは、「左」を意味する値を入力する左方向指定キーに対応する機能を提供する。キーVK_UP_RIGHTは、「右斜め上」を意味する値を入力する右上方向指定キーに対応する機能を提供する。キーVK_UP_LEFTは、「左斜め上」を意味する値を入力する左上方向指定キーに対応する機能を提供する。キーVK_DOWN_RIGHTは、「右斜め下」を意味する値を入力する右下方向指定キーに対応する機能を提供する。キーVK_DOWN_LEFTは、「左斜め下」を意味する値

を入力する左下方向指定キーに対応する機能を提供する。これらの方向キーを用いることで、例えば画面上のカーソル表示の移動を指示することができる。

【００８９】

キーVK_MENUは、メニューを表示させるメニューキーに対応する機能を提供する。キーVK_ENTERは、「決定」を指示する決定キーに対応する機能を提供する。キーVK_RETURNは、処理のステップを一つ戻す戻るキーに対応する機能を提供する。

【００９０】

キーVK_COLORED_KEY_1は、色つきファンクションキー１、キーVK_COLORED_KEY_2は、色つきファンクションキー２、キーVK_COLORED_KEY_3は、色つきファンクションキー３、キーVK_COLORED_KEY_4は、色つきファンクションキー４、キーVK_COLORED_KEY_5は、色つきファンクションキー５、キーVK_COLORED_KEY_6は、色つきファンクションキー６にそれぞれ対応する機能を提供する。

【００９１】

上述の図９に示したキー入力と図１０に示したキー入力とでは役割が異なるため、通知先をネイティブ実装プラットフォーム３０１で振り分ける必要がある。上述したように、図９に示されるキー入力により、ビデオ、オーディオおよび字幕の再生に関する指示がなされる。ネイティブ実装プラットフォーム３０１は、ユーザ入力３１０として図９に示されるキー入力を受け取ると、受け取ったキー入力を、図１１に示すコマンドに変換してムービープレーヤ３００に通知する。

【００９２】

一方、図１０に示されるキー入力は、ＧＵＩに対するユーザ入力３１０であるので、このユーザ入力は、画面構成やボタンを配置するスクリプトレイヤ３０２に通知されて処理される必要がある。ネイティブ実装プラットフォーム３０１は、ユーザ入力３１０として図１０に示されるキー入力を受け取ると、図２におけるイベント３１４に変換してスクリプトレイヤ３０２に通知する。図１２は、このキー入力に対応する一例のイベント３１４を示す。

【００９３】

なお、上述した図９および図１０には、キーVK_ANGLE、キーVK_SUBTITLE、キーVK_AUDIOという、ストリーム切り替えに関するキー入力も含まれている。これらは、先ずユーザ入力３１０がムービープレーヤ３００に伝えられ、ムービープレーヤ３００からスクリプトにストリーム切り換え要求があったことを示すイベントが通知される。そして、スクリプトプログラムからムービープレーヤ３００に対するストリーム切り換えのメソッドで、オーディオや字幕が切り替わるようにしている。そのため、ネイティブ実装プラットフォーム３０１からムービープレーヤ３００に対して伝えられるべきキー入力である。

【００９４】

上述した図１１のコマンドについて、より詳細に説明する。コマンドuo_timeSearch(playListTime)は、再生中のプレイリストの指定時刻からの再生を指示する。引数playListTimeは、プレイリストの先頭を０としたときの時刻を表す。このコマンドでは、プレイリスト番号の指定はできないため、引数playListTimeで表される時刻は、現在再生中のプレイリストの範囲内での指定時刻となる。コマンドuo_play()は、１倍速での再生開始を指示する。開始位置は、リジュームインフォメーション３２４に基づき決められる。リジュームインフォメーション３２４に対応する情報が無い場合は、このユーザ操作は無効とされる。このコマンドは、プレイリスト番号の指定の無いメソッドplay()を実行したときに対応する。また、このコマンドにおいて、ユーザ操作ではプレイリスト番号を指定できない。

【００９５】

コマンドuo_playChapter(chapterNumber)は、再生中のプレイリストの、引数chapterNumberで指定されたチャプタからの再生開始を指示する。チャプタの指定がない場合には、現在再生中のチャプタの先頭からの再生開始を指示する。これは、チャプタ番号の指定の無いメソッドplayChapter()に対応する。コマンドuo_playPrevChapter()は、現在よりも

一つ前のチャプタからの再生開始を指示する。コマンドuo_playNextChapter()は、現在の次のチャプタからの再生開始を指示する。

【００９６】

コマンドuo_jumpToEnd()は、プレイリストの最後へのジャンプを指示する。このコマンドは、ムービープレーヤ３００に対して、現在の再生を中止してイベントplayListEndを発生させるように指示するユーザ操作に対応する。このコマンドに対応して、スクリプトレイヤ３０２では、イベントハンドラonPlayListEndが実行される。コマンドuo_forwardScan(speed)は、引数speedで指定された再生速度での順方向再生を指示する。コマンドuo_backwardScan(speed)は、引数speedで指定された再生速度での逆方向再生を指示する。これらコマンドuo_forwardScan(speed)およびコマンドuo_backwardScan(speed)における引数speedは、ＵＭＤビデオプレーヤの実装に依存する。

【００９７】

コマンドuo_playStep(forward)は、順方向のコマ送り再生を指示する。コマンドuo_playStep(backward)は、逆方向のコマ送り再生を指示する。コマンドuo_pauseOn()は、ユーザ操作に基づき再生の一時停止を指示する。コマンドuo_pauseOff()は、ユーザ操作に基づき再生の一時停止状態を解除する。

【００９８】

コマンドuo_setAudioEnabled(boolean)は、オーディオストリームのＯＮ／ＯＦＦを指定する。このコマンドの実行時に、フラグaudioFlagの値も対応した内容に変更する。コマンドuo_setSubtitleEnabled(boolean)は、字幕ストリームのＯＮ／ＯＦＦを指定する。このコマンドの実行時に、フラグsubtitleFlagの値も対応した内容に変更する。コマンドuo_angleChange()は、表示アングルの変更を指示する。このコマンドによるユーザ操作がムービープレーヤ３００に伝えられると、ムービープレーヤ３００は、スクリプトレイヤ３０２に対してイベントangleChangeを通知する。コマンドuo_audioChange(audioStreamNumber)は、再生するオーディオストリームの変更を指示する。コマンドuo_changeAudioChannel(value)は、オーディオのチャンネル切り換えまたはデュアルモノ再生時の片チャンネル切り替えを指示する。このコマンドの実行時に、フラグaudioFlagの値も対応した内容に変更する。コマンドuo_subtitleChange(subtitleStreamNumber)は、再生する字幕ストリームの変更を指示する。

【００９９】

上述した図１２に示すイベントおよびイベントのムービープレーヤ３００のメソッドとの関係について、より詳細に説明する。イベントmenuは、メニューにジャンプする。このイベントは、ムービープレーヤ３００に対してではなく、ネイティブ実装プラットフォーム３０１からスクリプトレイヤ３０２に通知される。このイベントmenuがスクリプトレイヤ３０２に受け取られると、スクリプトレイヤ３０２は、イベントハンドラonMenuを実行する。イベントexitは、ネイティブ実装プラットフォーム３０１がＵＭＤビデオアプリケーションを終了させる際に、ネイティブ実装プラットフォーム３０１から発せられるイベントである。このイベントexitがスクリプトレイヤ３０２に受け取られると、スクリプトレイヤ３０２は、イベントハンドラonExitを実行する。

【０１００】

イベントresourceChangedは、リソースファイルの切り換えが発生したときに、ネイティブ実装プラットフォーム３０１から発せられるイベントである。このイベントresouceChangedがスクリプトレイヤ３０２に受け取られると、スクリプトレイヤ３０２は、イベントハンドラonResourceChangedを実行する。

【０１０１】

イベントup、イベントdown、イベントleft、イベントright、イベントfocusIn、イベントfocusOut、イベントpushおよびイベントcancelは、画面に表示されているＧＵＩ部品であるボタン画像にフォーカスが当たっている場合に発生するイベントである。このイベントは、ムービープレーヤ３００に対してではなく、ネイティブ実装プラットフォーム３０１からスクリプトレイヤ３０２に通知される。なお、ボタン画像にフォーカスが当たった

場合とは、例えば、画面上の位置を指示するためのカーソルがボタン画像の表示座標を示し、当該ボタン画像が選択可能となっているような状態である。イベントup、イベントdown、イベントleftおよびイベントrightは、ボタン画像に対するフォーカスが、それぞれ上、下、左および右のボタン画像に移動した場合に発生する。イベントfocusInは、あるボタン画像にフォーカスが当たった場合に発生し、イベントfocusOutは、フォーカスの当たっていたボタン画像からフォーカスが外れた場合に発生する。また、イベントpushは、フォーカスの当たっているボタン画像に対して押下操作が行われた際に発生する。イベントcancelは、ボタン画像の押下操作に対してキャンセル操作が行われた際に発生する。

【０１０２】

イベントautoPlayおよびイベントcontinuePlayは、スクリプトレイヤ３０２におけるスクリプトの実行開始を指示するイベントである。イベントautoPlayは、ディスクの装填時に自動的にスクリプトの実行を開始するように指示するイベントである。イベントcontinuePlayは、ディスク装填時に、例えばリジュームインフォメーション３２４に基づき、以前中止された時点からのスクリプトの実行再開を指示する。

【０１０３】

図１２で示したイベントに対しては、イベントが発生したときに実行されるプログラムが存在する。このイベントに対応したプログラムをイベントハンドラと称する。イベントとイベントハンドラとは、例えば名前で対応関係をつけることができる。一例として、イベント名の先頭に「on」を付加したものがイベントハンドラ名となる。図１３および図１４は、一例のイベントハンドラを示す。イベントハンドラの内容をコンテンツ制作者が記述することにより、ＵＭＤビデオプレーヤにコンテンツ制作者が意図する様々な動作を実行させることが可能になる。

【０１０４】

図１３は、ムービープレーヤオブジェクト３００が持つ一例のイベントの一部と、対応するイベントハンドラとを示す。この図１３のイベントは、上述した図２のイベント３１２に対応し、ムービープレーヤ３００からスクリプトレイヤ３０２に通知される。イベントハンドラは、一種のインターフェイスであって、その内容は、例えばコンテンツ制作者がスクリプト言語を用いて実装する。イベントハンドラをこのように構成することで、イベント発生時に、コンテンツ制作者の意図する動作を実現することができる。

【０１０５】

イベントmarkおよびイベントハンドラonMark()は、イベントマーク(Event-mark)が検出された際に実行される。イベントマークは、例えば、プレイリスト中に埋め込まれ、プレイリストの再生中にムービープレーヤ３００により検出される。ムービープレーヤ３００によりイベントマークが検出されると、ムービープレーヤ３００からスクリプトレイヤ３０２に対してイベントmarkが通知される。スクリプトレイヤ３０２は、このイベントmarkに対応するイベントハンドラonMark()を実行する。同様にして、イベントpalyListEndおよびイベントハンドラonPlayListEnd()は、プレイリストが終了した際に実行される。イベントchapterおよびイベントハンドラonChapter()は、チャプタマーク(Chapter-mark)検出時に実行される。チャプタマークは、例えば、プレイリスト中に埋め込まれ、プレイリストの再生中にムービープレーヤ３００により検出される。

【０１０６】

イベントangleChangeおよびイベントハンドラonAngleChange()は、ユーザ操作によりアングル変更が指示されたときに実行される。例えば、ユーザ操作に応じてキー入力VK_ANGLEがユーザ入力３１０としてネイティブ実装プラットフォーム３０１に入力されると、ネイティブ実装プラットフォーム３０１は、当該ユーザ入力３１０をコマンドuo_angleChange()に変換してムービープレーヤ３００に渡す。ムービープレーヤ３００は、このコマンドuo_angleChange()に応じてイベントangleChangeを発生させ、スクリプトレイヤ３０２に渡す。スクリプトレイヤ３０２は、このイベントangleChangeに対応したイベントハンドラonAngleChange()を実行する。同様にして、イベントaudioChangeおよびイベントハンドラonAudioChange()は、ユーザ操作によりオーディオの変更が指示されたときに実行さ

れる。イベントsubtitleChangeおよびイベントハンドラonSubtitleChange()は、ユーザ操作により字幕変更が指示されたときに実行される。

【０１０７】

図１４は、コントローラオブジェクト３３０が有する一例のイベントハンドラの一部を示す。この図１４に示されるイベントハンドラは、ネイティブ実装プラットフォーム３０１のコントローラオブジェクト３３０に属するイベントハンドラであり、ネイティブ実装プラットフォーム３０１からスクリプトレイヤ３０２に通知されることにより実行される。

【０１０８】

イベントmenuおよびイベントハンドラonMenu()は、メニューにジャンプする。イベントmenuは、例えば、ユーザ操作などでメニューキーが押下されたときに、ネイティブ実装プラットフォーム３０１からスクリプトレイヤ３０２に通知されるイベントである。スクリプトレイヤ３０２は、このイベントを受けて、対応するイベントハンドラonMenu()を実行し、イベントハンドラonMenu()内でメニュー画面を構成するＧＵＩ部品の配置や表示などを行う。イベントexitおよびイベントハンドラonExit()は、ネイティブ実装プラットフォーム３０１がＵＭＤビデオアプリケーションを終了させる際に、ネイティブ実装プラットフォーム３０１から発せられるイベントおよび対応するイベントハンドラである。

【０１０９】

イベントexitは、例えば、ユーザ操作などによりＵＭＤビデオプレーヤの動作の終了が指示された際に、ネイティブ実装プラットフォーム３０１からスクリプトレイヤ３０２に通知される。スクリプトレイヤ３０２のスクリプトは、通知されたイベントexitを受けて、イベントハンドラonExit()内で終了処理を行うことができる

【０１１０】

イベントresourceChangedおよびイベントハンドラonResourceChangedは、ネイティブ実装プラットフォーム３０１がリソースファイルを切り換えた後に、ネイティブ実装プラットフォーム３０１から発せられるイベントおよび対応するイベントハンドラである。

【０１１１】

イベントautoPlayおよびイベントハンドラonAutoPlay()、ならびに、イベントcontinuePlayおよびイベントハンドラonContinuePlay()は、それぞれスクリプトの実行を開始する。

【０１１２】

なお、コントローラオブジェクト３３０のイベントハンドラ以外に、ボタンに関するイベントハンドラがある。このボタンに関するイベントハンドラは、この発明と関連性が低いので、説明を省略する。

【０１１３】

図１５のフローチャートを用いて、ユーザ入力イベントをきっかけとして、用意されたプログラムが実行される一例の処理について、概略的に説明する。図１５は、ＵＭＤビデオプレーヤにおいてディスクを通常再生中に、ユーザにより、次のチャプタを再生することを指示するためのキー（例えば"next"キー）が押されたときに、このキー入力に対応して、次のチャプタにジャンプして再生を開始すると共に、用意されたメッセージを画面上に表示する例である。

【０１１４】

例えば、ＵＭＤビデオプレーヤによりディスクを通常再生中に、ユーザがＵＭＤビデオプレーヤのリモートコントロールコマンダを用いてキー"next"を押下すると（ステップＳ１０）、ネイティブ実装プラットフォーム３０１に対するユーザ入力３１０として、キーVK_NEXTが渡される。ネイティブ実装プラットフォーム３０１では、このユーザ入力３１０に対応してユーザコマンドuo_playNextChapter()が発生する（ステップＳ１１）。このユーザコマンドuo_playNextChapter()は、ムービープレーヤ３００に通知される。

【０１１５】

このコマンドuo_playNextChapter()を受け取ったムービープレーヤ３００は、データベ

ース３２０を検索し、プレイリスト情報から現在再生している位置を基準として、次のチャプタマークの位置を取得する（ステップＳ１２）。ステップＳ１３で、次のチャプタマークが存在するか否かが判断され、若し、存在しないと判断された場合、チャプタジャンプを行わず、現在の再生が継続される。

【０１１６】

一方、ステップＳ１３で、次のチャプタマークが存在すると判断されれば、処理はステップＳ１４に移行する。ステップＳ１４では、ムービープレーヤ３００は、現在の再生を中止し、次のチャプタマークが指し示す、クリップＡＶストリームファイル内でのバイト位置を、データベース３２０のクリップインフォメーションファイルの特徴点情報から取得する。そして、ステップＳ１５で、取得されたファイル内バイト位置にアクセスし、その位置からストリームの読み込みを開始して再生を開始する。

【０１１７】

ステップＳ１６以下は、チャプタが切り替わったことを知らせるメッセージを画面上に表示するための一連の手順である。チャプタが切り替わりチャプタの先頭からの再生が開始されると、チャプタイベントが発生する（ステップＳ１６）。例えば、チャプタの先頭に設けられたチャプタマークがムービープレーヤ３００に検出され、イベントchapterが発生される。このチャプタイベントは、ムービープレーヤ３００からスクリプトレイヤ３０２に通知される。ムービープレーヤ３００は、このイベントの通知時に、ジャンプするチャプタのチャプタ番号も共に、スクリプトレイヤ３０２に対して通知する。スクリプトレイヤ３０２は、通知されたイベントに対応するイベントハンドラ、例えばイベントハンドラonChapter()の実行を開始する（ステップＳ１７）。

【０１１８】

この例では、イベントハンドラ内には、チャプタが切り替わった際に画面上にその旨を知らせるメッセージを表示する動作が記述されているものとする。スクリプトレイヤ３０２のスクリプトは、このイベントハンドラを実行し、イベント発生時にムービープレーヤ３００から通知されたジャンプ先のチャプタ番号を取得し（ステップＳ１８）、ネイティブ実装プラットフォーム３０１に対して、例えば取得したチャプタ番号のチャプタの先頭であるなど、所定のメッセージを画面上に表示する指示を出す。ネイティブ実装プラットフォーム３０１は、この指示に応じて、画面上にメッセージを表示し（ステップＳ１９）、イベントハンドラによる処理が終了される（ステップＳ２０）。

【０１１９】

上述のような処理により、ユーザが次のチャプタの再生開始を指示するキー"next"を操作することによりチャプタジャンプが行われ、ジャンプ先である次のチャプタの再生開始時にチャプタの先頭であることを示すメッセージが画面上に表示されることになる。

【０１２０】

このように、ユーザ入力イベントは、ムービープレーヤ３００の状態を変化させ、また、新たなイベントを発生させる契機ともなり、新たに発生したイベントを利用して様々な処理を行わせることができる。

【０１２１】

以上のようなプレーヤモデルにより、ビデオ、オーディオおよび字幕の再生が可能となる。コンテンツ制作者が予め設定しておいた、再生中のある時刻にあるイベントを発生させて、予め用意しておいたイベントハンドラが実行されることで、コンテンツ制作者が意図する動作を実現できる。また、プレイリストの再生中にプレーヤに対するユーザ操作があった場合には、ユーザ操作によるユーザ入力３１０に応じてネイティブ実装プラットフォーム３０１からムービープレーヤ３００に対して制御コマンドが与えられ、ユーザの意図する通りにプレーヤの状態を変化させることができる。さらに、プレーヤに対するユーザ操作によるユーザ入力３１０を受けたネイティブ実装プラットフォーム３０１がスクリプトレイヤ３０２のスクリプトにイベントを通知することにより、ユーザ操作に応じ、コンテンツ制作者が用意した動作を実行させることも可能となる。

【０１２２】

このようにプレーヤモデルを構築することで、ビデオ、オーディオおよび字幕の再生と、インタラクティブな操作とをユーザに提供することが可能となる。

【０１２３】

５．スクリプトプログラムの例

次に、スクリプトレイヤ３０２のスクリプトプログラムの例について説明する。先ず、図１６に示されるようなコンテンツ再生の流れが、コンテンツ制作者により作られているものとする。図１６に示されるコンテンツは、表示される要素としては、プレイリスト４００および４０１、トップメニュー４０２、ならびに、メッセージ４０３から構成される。プレイリスト４００は、ディスクが装填されると自動的に表示される警告文画面を表示するためのものである。プレイリスト４０１は、例えばこのコンテンツの主眼である映画の本編である。トップメニュー画面４０２は、プレイリスト４０１の再生を指示できるように、ボタンなどのＧＵＩ部品が配置される。また、メッセージ４０３は、プレイリスト４０１の再生中の任意の時刻に表示される。

【０１２４】

さらに、この図１６の構成では、幾つかのイベントハンドラが用意されている。イベントハンドラonAutoPlay()は、ディスクがＵＭＤプレーヤに装填されると、プレイリスト４００を自動的に再生し、警告文を表示させる。イベントハンドラonPlayListEnd()は、プレイリストの再生が終了すると呼び出されるイベントハンドラで、この図１６の例では、プレイリスト４００やプレイリスト４０１の終了で呼び出される。すなわち、イベントハンドラonPlayListEnd()は、どのプレイリストが終了したかを判定し、プレイリスト４００の再生が終了した場合には、プレイリスト４０１の再生開始を指示する。また、プレイリスト４０１の再生が終了した場合には、トップメニュー画面４０２を呼び出す。

【０１２５】

イベントハンドラonMenu()は、ユーザがメニューキーを操作したときに呼び出され、トップメニュー４０２を呼び出して画面に表示する。イベントハンドラonMark()は、再生中にマークMarkが指し示す時刻に到達したときに実行される。この図１６の例では、プレイリスト４０１に対してマークMarkが設定されており、プレイリスト４０１の再生がマークMarkの指し示す時刻に到達すると、画面上にメッセージ４０３が表示されるようになっている。

【０１２６】

すなわち、図１６の例では、ＵＭＤビデオプレーヤにディスクが装填されると、イベントハンドラonAutoPlayが呼び出されてプレイリスト４００が再生され、警告画面が表示される。プレイリスト４００の再生時間が経過し、プレイリスト４００の最後に到達すると、イベントハンドラonPlayListEndが呼び出され、プレイリスト４００が最後まで再生されたことが判定され、次のプレイリスト４０１が再生される。ここで、プレイリスト４０１の再生中に、ユーザによりメニューキーが操作されると、イベントハンドラonMenuが呼び出され、トップメニュー画面４０２が表示される。また、イベントハンドラonMenuにより、トップメニュー画面４０２に対する所定の操作に応じて、プレイリスト４０１の先頭から再生が開始される。さらに、プレイリスト４０１の再生時刻がマークMarkが指し示す時刻に到達したら、イベントハンドラonMarkが呼び出され、メッセージ４０３が画面上に表示される。プレイリスト４０１が最後まで再生されると、イベントハンドラonPlayListEndが呼び出され、プレイリスト４０１が最後まで再生されたことが判定され、トップメニュー画面４０２が表示される。

【０１２７】

図１７は、この図１６に示すような動作を実現するための一例のスクリプトプログラムを示す。上述したように、スクリプトプログラムは、イベントハンドラが並べられ、イベントの発生に応じて対応するイベントハンドラが実行されるようになっている。スクリプトプログラムは、拡張子が「RCO」であるリソースファイルに格納される。

【０１２８】

ムービープレーヤ３００に対してプレイリストの再生を指示するメソッドは、「moviep

layer.play()」である。括弧内には、引数として、再生するプレイリストの番号を記述する。プレイリストの再生が終了すると、イベントplayListEndが発生する。このイベントplayListEndが発生すると、スクリプトからイベントハンドラmovieplayer.onPlayListEnd()が呼び出される。このとき、スクリプトには、イベントplayListEndと共に、オブジェクトevent_infoが渡される。オブジェクトevent_infoには、どのプレイリストが終了したかを表すプレイリスト番号などが格納される。スクリプトでは、このオブジェクトevent_infoの内容により、次の動作を変えることができる。

【０１２９】

６．ファイルの管理構造について

次に、ＵＭＤビデオ規格に適用されるファイルの管理構造について、図１８を用いて説明する。ファイルは、ディレクトリ構造により階層的に管理されて、ディスク上に記録される。ディスクのファイルシステムは、ＩＳＯ(International Organization for Standarization)－９６６０あるいはＵＤＦ(Universal Disk Format)などで規定されたファイルシステムを適用することができる。

【０１３０】

ルートディレクトリの下に、ファイル"TITLEID.DAT"およびディレクトリ"VIDEO"が置かれる。ディレクトリ"VIDEO"の下には、さらに、ディレクトリ"RESOURCE"、ディレクトリ"CLIP"およびディレクトリ"STREAM"、ならびに、ファイル"PLAYLIST.DAT"が置かれる。

【０１３１】

ファイル"TITLEID.DAT"は、タイトル（コンテンツの種類）毎に異なるタイトル識別子が格納されるファイルである。１つのディスクに対し、１つのファイル"TITLEID.DAT"を有する。

【０１３２】

ディレクトリ"RESOURCE"の下には、リソースファイル("JA000000.RCO")が置かれる。リソースファイルには、上述したように、スクリプトレイヤ３０２を構成するスクリプトプログラムが格納されると共に、メニュー画面を構成するために用いられるデータ、例えば画像データやサウンドデータなどの部品データが格納される。ディレクトリ"RESOURCE"の下には、通常、１個のリソースファイルが置かれる。これに限らず、ディレクトリ"RESOURCE"の下に、複数のリソースファイルを置くこともできる。複数のリソースファイルは、例えば、表示言語の異なる複数のメニューなどを用意する際に、言語毎に作成される。この場合でも、同時に用いられるリソースファイルは、１個とされる。

【０１３３】

なお、リソースファイルは、そのファイル名において、デリミタであるピリオドの後ろの拡張子を「RCO」に固定的として、当該ファイルがリソースファイルであることを示すようにしている。また、ピリオドの前の文字列により、当該リソースファイルの内容を概略的に示すようにしている。例えば、リソースファイルのファイル名全体を「CCdannnn.RCO」の形式として、先頭の２文字「CC」が当該リソースファイルに対応する言語コード、次の１文字「d」が当該言語コードがデフォルト言語であるか否かを表すフラグ、次の「a」が表示画面のアスペクト比、次の４文字「nnnn」が識別番号を表す。識別番号は、複数のリソースファイルにおいて、ファイル名が互いに重複しないように決められる。

【０１３４】

リソースファイルのファイル名の命名規則をこのようにすることで、リソースファイルのファイル名によって、リソースデータの言語属性と表示画面のアスペクト比とが分かるようになっている。リソースファイルの初期選択時には、適切なリソースファイルをファイル名に基づき判別する。

【０１３５】

ディレクトリ"CLIP"の下には、１以上のクリップインフォメーションファイルが置かれる。クリップインフォメーションファイルは、ファイル名を、デリミタであるピリオドの前が「00001」などの５文字乃至数文字からなる文字列（この例では数字）、ピリオドの後ろの拡張子が「CLP」とされる。拡張子「CLP」により、当該ファイルがクリップインフ

ォメーションファイルであることを識別できる。

【０１３６】

ディレクトリ"STREAM"の下には、１以上のクリップＡＶストリームファイルが置かれる。クリップＡＶストリームファイルは、ファイル名を、デリミタであるピリオドの前が「00001」などの５文字乃至数文字からなる文字列（この例では数字）、ピリオドの後ろの拡張子が「PS」とされる。拡張子「PS」により、当該ファイルがクリップＡＶストリームファイルであることを識別できる。この実施の一形態では、クリップＡＶストリームファイルは、ビデオストリーム、オーディオストリームおよびサブタイトル（字幕）ストリームが多重化され、ＭＰＥＧ２(Moving Pictures Experts Group 2)のプログラムストリームとして、上述の拡張子「PS」で識別されるファイルに格納される。

【０１３７】

上述したように、クリップＡＶストリームファイルは、ビデオデータおよびオーディオデータを圧縮符号化および時分割多重して得られるファイルであって、このファイルを読み込み、デコード処理を行うことで、ビデオデータおよびオーディオデータが得られる。また、クリップインフォメーションファイルは、このクリップＡＶストリームファイルの性質などが記述されるファイルであって、クリップＡＶストリームファイルと対応する。この実施の一形態では、クリップインフォメーションファイルと対応するクリップＡＶストリームファイルとで、ファイル名における、拡張子の前の、５文字乃至数文字からなる文字列を一致させておくことで、両者の対応関係を容易に把握できる。

【０１３８】

リソースファイルは、上述したように、スクリプトプログラムが記述されたスクリプトファイルを含み、この実施の一形態が適用されるディスクの再生形態をインタラクティブなものとするために用いるプログラムが格納されている。リソースファイルは、ディスクに格納される他のファイルに先立って読み出される。

【０１３９】

ファイル"PLAYLIST.DAT"は、クリップＡＶストリームの再生順を指定するプレイリストが記述されたプレイリストファイルである。図１９～図２１を用いて、ファイル"PLAYLIST.DAT"の内部構造について説明する。図１９は、ファイル"PLAYLIST.DAT"の全体構造を表す一例のシンタクスを示す。ここでは、シンタクスをコンピュータ装置などのプログラムの記述言語として用いられるＣ言語の記述法に基づき示す。これは、他のシンタクスを表す図において、同様である。

【０１４０】

フィールドname_lengthは、８ビットのデータ長を有し、このプレイリストファイルに付された名称の長さを示す。フィールドname_stringは、２５５バイトのデータ長を有し、このプレイリストファイルに付された名称を示す。フィールドname_stringは、その先頭から、フィールドname_lengthが表すバイト長までが、有効な名称として使用される。例えば、フィールドname_lengthが値"１０"を持つ場合には、フィールドname_stringの先頭から１０バイト分が有効な名称として解釈される。

【０１４１】

フィールドnumber_of_PlayListsは、１６ビットのデータ長を有し、続けて記述されるブロックPlayList()の個数を示す。次行のforループによりフィールドnumber_of_PlayListsに示される回数分だけ、当該個数のブロックPlayList()が記述される。ブロックPlayList()は、プレイリストそのものである。

【０１４２】

ブロックPlayList()の一例の内部構造について説明する。ブロックPlayList()の先頭には、フィールドPlayList_data_lengthが配される。フィールドPlayList_data_lengthは、３２ビットのデータ長を有し、当該フィールドPlayList_data_lengthを含むブロックPlayList()のデータ長を示す。続いて、１５ビットのデータ長を有するフィールドreserved_for_word_alignmentと、１ビットのデータ長を有するフラグcapture_enable_flag_PlayListとが配される。フィールドreserved_for_word_alignmentは、データ長が１ビットのフラ

グcapture_enable_flag_PlayListと組み合わせて、ブロックPlayList()内での配置を１６ビットの位置に揃えるために用いられる。

【０１４３】

フラグcapture_enable_flag_PlayListは、当該capture_enable_flag_PlayListを含むブロックPlayList()に属する動画像の二次利用を許可するか否かを示すフラグである。例えば、このフラグcapture_enable_flag_PlayListの値が"１"であれば、当該PlayList()に属する動画像の、再生機内での２次利用を許可することを示す。

【０１４４】

なお、上述では、フラグcapture_enable_flag_PlayListを１ビットのフラグとしたが、これはこの例に限定されない。例えば、フラグcapture_enable_flag_PlayListを複数ビット構成として、２次利用の段階的な許可を記述するようにしてもよい。一例として、フラグcapture_enable_flag_PlayListを２ビット構成とし、値が"０"の場合には２次利用を完全禁止とし、値が"１"の場合には例えば６４画素×６４ラインなど、所定の解像度以下に圧縮符号化した場合のみ２次利用を可能とする。また、値が"２"であれば、制限無く２次利用を許可するといった利用が考えられる。これに限らず、２ビット構成のうちビット０が値"１"の場合にはコンテンツ再生アプリケーションでの２次使用を許可し、ビット１が値"１"の場合には同一筐体内の他のアプリケーション（例えば壁紙画像やスクリーンセーバ）での２次使用を許可する。この場合には、ビット０およびビット１の値を組み合わせて用いることができる。

【０１４５】

フィールドPlayList_name_lengthは、８ビットのデータ長を有し、このブロックPlayList()に付された名称の長さを示す。フィールドPlayList_name_stringは、２５５ビットのデータ長を有し、このブロックPlayList()に付された名称を示す。フィールドPlayList_name_stringは、その先頭から、フィールドPlayList_name_stringが表すバイト長までが、有効な名称として使用される。

【０１４６】

フィールドnumber_of_PlayItemsは、１６ビットのデータ長を有し、続けて記述されるブロックPlayItem()の個数を示す。次行のforループによりフィールドnumber_of_PlayItem2に示される回数分だけ、当該個数のブロックPlayItem()が記述される。ブロックPlayItem()は、プレイアイテムそのものである。

【０１４７】

ブロックPlayList()内の各ブロックPlayItem()には、識別情報（ＩＤ）が付与される。例えば、ブロックPlayList()内の最初に記述されるブロックPlayItem()は、０番とされ、以降、ブロックPlayItem()の出現順に、１番、２番、・・・と通し番号が付される。この通し番号が各ブロックPlayItem()の識別情報として用いられる。ブロックPlauItem()の個数だけ繰り返されるforループの引数iを、対応するブロックPlayItem()の識別情報として用いることができる。ブロックPlayItem()の次に、ブロックPlayListMark()が配置される。

【０１４８】

図２０を用いて、ブロックPlayItem()の一例の内部構造について説明する。ブロックPlayItem()の先頭には、フィールドlengthが配される。フィールドlengthは、１６ビットのデータ長を有し、当該ブロックPlayItem()の長さを示す。続いて、フィールドClip_Information_file_name_lengthが配される。フィールドClip_Information_file_name_lengthは、１６ビットのデータ長を有し、このブロックPlayItem()に対応するクリップインフォメーションファイルの名称の長さを示す。フィールドClip_Information_file_nameは、バイト単位で可変長のデータ長を有し、このブロックPlayItem()に対応するクリップインフォメーションファイルの名称を示す。フィールドClip_Information_file_nameは、その先頭から、フィールドClip_Information_file_name_lengthが表すバイト長までが、有効な名称として使用される。フィールドClip_Information_file_nameでクリップインフォメーションファイルが指定されると、上述したファイル名の対応関係により、当該クリップイン

フォメーションファイルに対応するクリップＡＶストリームファイルが特定できる。

【０１４９】

フィールドIN_timeおよびフィールドOUT_timeは、それぞれ３３ビットのデータ長を有し、ブロックPlayItem()内においてフィールドClip_Information_file_nameで指定したクリップインフォメーションファイルに対応するクリップＡＶストリームファイルの再生開始位置および再生終了位置を指定する時刻情報である。これらフィールドIN_timeおよびフィールドOUT_timeの情報を用いることで、クリップＡＶストリームファイルの先頭以外の部分からの再生開始を指定することができる。同様に、クリップＡＶストリームファイルの後端以外の再生終了を指定することができる。フィールドreserved_for_word_aliignmentは、データ構造のデータ長を１６ビットの整数倍にするための調整用のフィールドであって、１５ビットのデータ長を有する。

【０１５０】

図２１を用いて、ブロックPlayListMark()の一例の内部構造について説明する。ブロックPlayListMark()の先頭には、フィールドlengthが配される。フィールドlengthは、３２ビットのデータ長を有し、当該ブロックPlayListMark()の長さを示す。続いて、フィールドnumber_of_PlayList_marksが配される。フィールドフィールドnumber_of_PlayList_marksは、１６ビットのデータ長を有し、続くブロックMark()の個数を示す。次行のforループによりフィールドnumber_of_PlayList_marksに示される回数分だけ、当該個数のブロックMark()が記述される。

【０１５１】

ブロックMark()の一例の内部構造について説明する。ブロックMark()は、先頭にフィールドmark_typeが配される。フィールドmark_typeは、８ビットのデータ長を有し、当該フィールドmark_typeを含むブロックMark()の種類を示す。この実施の一形態では、図２２に一例が示されるように、チャプタマークおよびイベントマークの２種類のマークが規定されている。チャプタは、プレイリスト（ブロックPlayList()）を分割する頭出し単位であり、チャプタマークは、チャプタ位置を時刻情報で示す。イベントマークは、マークイベントを発生させるマークである。

【０１５２】

フィールドmark_name_lengthは、８ビットのデータ長を有し、このブロックMark()に付された名称の長さを示す。ブロックMark()の最下行に配されるフィールドmark_name_stringは、このブロックMark()に付された名称を示す。フィールドmark_name_stringは、その先頭から、フィールドmark_name_lengthが表すバイト長までが、有効な名称として使用される。

【０１５３】

フィールドref_to_PlayItem_id、フィールドmark_time_stamp、フィールドentry_ES_stream_idおよびフィールドentry_ES_private_stream_idの４要素は、ブロックPlayList()上で定義されるブロックMark()を、クリップＡＶストリームファイルと対応付ける。すなわち、フィールドref_to_PlayItem_idは、１６ビットのデータ長を有し、ブロックPlayItem()の識別情報を示す。これにより、クリップインフォメーションファイルと、クリップＡＶストリームファイルとが特定される。

【０１５４】

フィールドmark_time_stampは、３３ビットのデータ長を有し、クリップＡＶストリームファイル内でのマークの時刻を指定するために用いられる。図２３を用いて、概略的に説明する。図２３において、プレイリストは、番号０、１および２がそれぞれ指定された３つのプレイアイテム（PlayItem#0、PlayItem#1およびPlayItem#2）からなり、プレイリスト上の時刻t_0は、番号１のプレイアイテム（PlayItem#1）に含まれるものとする。また、番号０、１および２の各プレイアイテムは、それぞれ対応するクリップインフォメーションファイルを介してクリップＡＶストリームファイルのプログラムストリーム(Program Sream)Ａ、ＢおよびＣにそれぞれ対応しているものとする。

【０１５５】

このような場合において、プレイリスト上の時刻t_0にマークを指定する場合、フィールドref_to_PlayItem_idの値を、時刻t_0を含むプレイアイテムを示す"１"とし、さらに、対応するクリップＡＶストリームファイルＢ上で時刻t_0に相当する時刻を、フィールドmark_time_stampに記述する。

【０１５６】

図２１の説明に戻り、フィールドmark_time_stampに続けてフィールドentry_ES_stream_idおよびフィールドentry_ES_private_stream_idが配される。フィールドentry_ES_stream_idおよびフィールドentry_ES_private_stream_idは、それぞれ８ビットのデータ長を有し、当該ブロックMark()が特定のエレメンタリストリームに関連付けられている場合に、そのエレメンタリストリームを特定するために用いられる。フィールドentry_ES_stream_idおよびフィールドentry_ES_private_stream_idは、それぞれ該当するエレメンタリストリームが多重化されているパケット(packet())のストリームＩＤ(stream_id)と、プライベートパケットヘッダ(private_packet_header())のプライベートストリームＩＤ(private_stream_id)を示す。

【０１５７】

なお、これらパケット(packet())のストリームＩＤ(stream_id)、プライベートパケットヘッダ(private_packet_header())のプライベートストリームＩＤ(private_stream_id)は、例えばＭＰＥＧ２システムのプログラムストリームの規定に基づく。

【０１５８】

これらフィールドentry_ES_stream_idおよびフィールドentry_ES_private_stream_idは、例えば、クリップＡＶストリーム＃０とクリップＡＶストリーム＃１とで異なるチャプタ構成である場合などに用いられる。該当するブロックMark()が特定のエレメンタリストリームに関連付けられていない場合には、これら２つのフィールドの値がそれぞれ"０"とされる。

【０１５９】

次に、図２４～図２８を用いて、クリップインフォメーションファイルの内部構造について説明する。クリップインフォメーションファイル"XXXXX.CLP"は、上述したように、ディレクトリ"STREAM"の下に置かれた、対応するクリップＡＶストリームファイル"XXXXX.PS"の性質などを記述する。

【０１６０】

図２４は、クリップＡＶストリームファイル"XXXXX.CLP"の全体構造を表す一例のシンタクスを示す。クリップＡＶストリームファイル"XXXXX.CLP"は、先頭に、フィールドpresentation_start_timeおよびフィールドpresentation_end_timeがそれぞれ配される。フィールドpresentation_start_timeおよびフィールドpresentation_end_timeは、それぞれ３３ビットのデータ長を有し、対応するクリップＡＶストリームファイルの先頭と後端の時刻を示す。時刻情報は、ＭＰＥＧ２システムにおけるＰＴＳ(Presentation Time Stamp)を用いることができる。ＰＴＳは、９０ｋＨｚの精度を有する。

【０１６１】

次に、７ビットのデータ長を有するフィールドreserved_for_word_alignmentと、１ビットのデータ長を有するフラグcapture_enable_flag_Clipとが配される。フィールドreserved_for_word_alignmentは、データ長が１ビットのフラグcapture_enable_flag_Clipと組み合わせて、ファイル"XXXXX.CLP"内での配置を１６ビットの位置に揃えるために用いられる。フラグcapture_enable_flag_Clipは、当該ファイル"XXXXX.CLP"に対応するクリップＡＶストリームファイルに含まれる動画像の２次利用を許可するか否かを示すフラグである。例えば、このフラグcapture_enable_flag_Clipの値が"１"であれば、当該ファイル"XXXXX.CLP"に対応するクリップＡＶストリームファイルの動画像の、再生機内での２次利用を許可することを示す。

【０１６２】

フィールドnumber_of_streamsは、８ビットのデータ長を有し、続くブロックStreamInfo()構造の個数を示す。フィールドnumber_of_streamsの次から、forループによりフィー

ルドnumber_of_streamsで示される回数分だけ、ブロックStreamInfo()が記述される。forループの後には、ブロックEP_map()が配される。

【０１６３】

ブロックStreamInfo()の一例の内部構造について説明する。ブロックStreamInfo()の先頭には、フィールドlengthが配される。フィールドlengthは、１６ビットのデータ長を有し、当該ブロックStreamInfo()の長さを示す。続いて、それぞれ８ビットのデータ長を有するフィールドstream_idおよびフィールドprivate_stream_idが配され、図２５に一例が示されるように、当該ブロックStreamInfo()をエレメンタリストリームに関連付けている。この図２５の例では、当該ブロックStreamInfo()は、フィールドstream_idが値"０ｘＥ０"～値"０ｘＥＦ"でビデオストリームに関連付けられ、値"０ｘＢＤ"でＡＴＲＡＣ(Adaptive Transform Acoustic Coding)オーディオストリーム、ＬＰＣＭ(Linear Pulse Code Modulation)オーディオストリームまたは字幕ストリームと関連付けられる。また、当該ブロックStreamInfo()は、フィールドprivate_stream_idが値"０ｘ００"～値"０ｘ０Ｆ"、値"０ｘ１０"～値"０ｘ１Ｆ"および値"０ｘ８０"～値"０ｘ９Ｆ"で、ＡＴＲＡＣオーディオストリーム、ＬＰＣＭオーディオストリームおよび字幕ストリームにそれぞれ関連付けられる。

【０１６４】

なお、図２５での値の表記において、「０ｘ」は、後続する数値が１６進表記であることを示す。これは、以下の同様な表現において、共通である。

【０１６５】

ここで、ブロックStreamInfo()は、大別して、ストリーム中で変化しない情報とストリーム中で変化する情報との２種類の情報が記述されている。ストリーム中で変化しない情報は、ブロックStaticInfo()に記述される。一方、ストリーム中で変化する情報は、変化点を時刻情報で指定して、ブロックDynamicInfo()に記述される。

【０１６６】

ブロックStreamInfo()において、ブロックStaticInfo()の後ろにバイト位置を揃えるための、８ビットのデータ長を有するフィールドreserved_for_word_alignmentが配され、その次に、フィールドnumber_of_DynamicInfoが配される。フィールドnumber_of_DynaminInfoは、８ビットのデータ長を有し、ブロックStreamInfo()内にその後に記述されるブロックDynamicInfo()の個数が示される。forループにより、フィールドnumber_of_DynamicInfoで示される回数分だけ、フィールドpts_change_pointおよびブロックDynamicInfo()が記述される。

【０１６７】

フィールドpts_change_pointは、３３ビットのデータ長を有し、対応するブロックDynamicInfo()の情報が有効になる時刻をＰＴＳにより示す。ストリーム毎に先頭となる時刻も、フィールドpts_change_pointで示され、これは、ファイル"XXXXX.CLP"内で定義される、上述したフィールドpresentation_start_timeと等しくなる。

【０１６８】

図２６を用いて、ブロックStaticInfo()の一例の内部構造について説明する。ブロックStaticInfo()は、対応するエレメンタリストリームの種類により内容が異なる。対応するエレメンタリストリームの種類は、図２５を用いて説明した、フィールドstream_idおよびフィールドprivate_stream_idの値に基づき判断できる。図２６では、ブロックStaticInfo()が対応するエレメンタリストリームの種類がビデオストリーム、オーディオストリームおよびサブタイトル（字幕）ストリームの何れであるかを、if構文を用いてそれぞれ記述している。以下、ブロックStaticInfo()について、エレメンタリストリーム毎に説明する。

【０１６９】

エレメンタリストリームがビデオストリームであった場合、ブロックStaticInfo()は、それぞれ４ビットのデータ長を有するフィールドpicture_sizeおよびフィールドframe_rate、１ビットのデータ長を有するフラグcc_flagからなる。フィールドpicture_sizeおよ

びフィールドframe_rateは、当該ビデオストリームの画像のサイズおよびフレーム周波数をそれぞれ示す。フラグcc_flagは、当該ビデオストリームがクローズドキャプションを含むか否かを示す。例えば、フラグcc_flagの値が"１"で、当該ビデオストリームがクローズドキャプションを含む。フィールドreserved_for_word_alignmentは、データ配置を１６ビットに揃えるために用いられる。

【０１７０】

エレメンタリストリームがオーディオストリームであった場合、ブロックStaticInfo()は、１６ビットのデータ長を有するフィールドaudio_language_code、８ビットのデータ長を有するフィールドchannel_configuration、１ビットのデータ長を有するフラグlfe_exsistanceおよび４ビットのデータ長を有するフィールドsampling_frequencyからなる。フィールドaudio_language_codeは、当該オーディオストリームに含まれている言語を表すコードを示す。フィールドchannel_configurationは、モノラル、ステレオ、マルチチャンネルなど、オーディオデータのチャンネル属性を示す。フィールドlfe_existanceは、低域強調チャンネルが含まれているか否かを示し、例えば値が"１"で、含まれていることを示す。フィールドsampling_frequencyは、オーディオデータのサンプリング周波数を示す。フィールドreserved_for_word_alignmentは、データ配置を１６ビットに揃えるために用いられる。

【０１７１】

エレメンタリストリームがサブタイトル（字幕）ストリームであった場合、ブロックStaticInfo()は、１６ビットのデータ長を有するフィールドsubtitle_language_codeおよび１ビットのデータ長を有するフラグconfigurable_flagからなる。フィールドsubtitle_language_codeは、当該字幕ストリームに含まれている言語を表すコードを示す。フラグconfigurable_flagは、当該字幕ストリームを表示する際に、文字の大きさや位置の変更を許可するか否かを示し、例えば値が"１"で、許可することを示す。フィールドreserved_for_word_alignmentは、データ配置を１６ビットに揃えるために用いられる。

【０１７２】

図２７を用いて、ブロックDynamicInfo()の一例の内部構造について説明する。ブロックDynamicInfo()は、先頭に、８ビットのデータ長を有するフィールドreserved_for_word_alignmentが配される。続く内容は、対応するエレメンタリストリームの種類により異なる。対応するエレメンタリストリームの種類は、図２５を用いて説明した、フィールドstream_idおよびフィールドprivate_stream_idの値に基づき判断できる。図２７では、ブロックDynamicInfo()が対応するエレメンタリストリームの種類がビデオストリーム、オーディオストリームおよびサブタイトル（字幕）ストリームの何れであるかを、if構文を用いてそれぞれ記述している。以下、ブロックDynamicInfo()について、エレメンタリストリーム毎に説明する。

【０１７３】

エレメンタリストリームがビデオストリームであった場合、ブロックDynamicInfo()は、４ビットのデータ長を有するフィールドdisplay_aspect_ratioからなる。フィールドdisplay_aspect_ratioは、ビデオの表示出力アスペクト比が１６：９か４：３かを示す。フィールドreserved_for_word_alignmentは、データ配置を１６ビットに揃えるために用いられる。

【０１７４】

エレメンタリストリームがオーディオストリームであった場合、ブロックDynamicInfo()は、４ビットのデータ長を有するフィールドchannel_assignmentからなる。フィールドchannel_assignmentは、当該オーディオストリームが２チャンネルで構成されている場合に、出力がステレオかデュアルモノかを示す。デュアルモノは、例えば２ヶ国語の音声を再生可能とする際に用いられる。フィールドreserved_for_word_alignmentは、データ配置を１６ビットに揃えるために用いられる。

【０１７５】

エレメンタリストリームが字幕ストリームであった場合、ブロックDynamicInfo()は、

データ配置を１６ビットに揃えるために用いられる、フィールドreserved_for_word_alignmentで構成される。すなわち、字幕ストリームに関しては、動的に変化する属性が定義されていない。

【０１７６】

図２８を用いて、ブロックEP_map()の一例の内部構造について説明する。ブロックEP_map()は、エレメンタリストリーム毎に、ビットストリーム内のデコード開始可能位置（エントリポイント、あるいはランダムアクセスポイント(RAP)ともいう）を、時刻情報と位置情報とを用いて表したものである。位置情報は、例えばエレメンタリストリームが記録される記録媒体における、アクセスの最小単位を用いることができる。各エレメンタリストリームは、ブロックEP_map()で示された位置からのデコード処理が可能であるものとする。

【０１７７】

固定レートのストリームでは、デコード開始可能位置を計算で求めることができるので、このブロックEP_map()のような情報は、不要である。一方、可変レートのストリームや、ＭＰＥＧ系のビデオの圧縮符号化方式のようにアクセスユニット毎にデータのサイズが変わるようなストリームの場合には、ランダムアクセスを行うために重要な情報となる。

【０１７８】

ブロックEP_map()は、先頭に、配置を１６ビットに揃えるために、８ビットのデータ長を有するフィールドreserve_for_word_alignmentが配される。続いて、フィールドnumber_of_stream_id_entriesが配される。フィールドnumber_of_stream_id_entriesは、８ビットのデータ長を有し、このブロックEP_map()に記述されているエレメンタリストリームの数を示す。第１のforループにより、フィールドstream_id、フィールドprivate_stream_idおよびフィールドnumber_of_EP_entriesが、フィールドnumber_of_stream_id_entriesで示される回数分だけ、記述される。さらに、第１のforループの１回の記述毎に、第２のforループにより、フィールドnumber_of_EP_entriesで示される回数分だけ、フィールドPTS_EP_startおよびフィールドRPN_EP_startが配される。

【０１７９】

第１のforループ内において、最初に、それぞれ８ビットのデータ長を有するフィールドstream_idおよびフィールドprivate_stream_idが配され、図２５に一例が示されるようにして、エレメンタリストリームを特定している。次に、配されるフィールドnumber_of_EP_entriesは、３２ビットのデータ長を有し、当該エレメンタリストリームに対して記述されているエントリポイントの数を示す。その後、第２のforループにて、フィールドnumber_of_EP_entriesが示す数だけ、フィールドPTS_EP_startおよびフィールドRPN_EP_startがそれぞれ配される。

【０１８０】

フィールドPTS_EP_startおよびフィールドRPN_EP_startは、それぞれ３３ビットのデータ長を有し、エントリポイント自体を表す。フィールドPTS_EP_startは、エントリポイントのクリップＡＶストリームファイル内での時刻をＰＴＳで示す。一方、フィールドRPN_EP_startは、エントリポイントのクリップＡＶストリームファイル内での位置を例えば２０４８バイト単位で示す。

【０１８１】

この実施の一形態においては、ディスク状のアクセス単位である１セクタが２０４８バイトとされる。そのため、エントリポイントのクリップＡＶストリームファイル内での位置は、フィールドRPN_EP_startにより、セクタ単位で示されることになる。

【０１８２】

ここで、ビデオストリームの再生開始可能位置の直前には、必ず、パケットprivate_stream_2が配される。このパケットprivate_stream_2は、ビデオストリームをデコードするために利用可能な情報が格納されるパケットである。そのため、ビデオストリームのエントリポイントの位置は、当該パケットprivate_stream_2が格納されるパックpack()の位置とされる。

【０１８３】
ブロックEP_map()は、上述のようにして、クリップＡＶストリーム上の時刻と、クリップＡＶストリームファイル内での位置とを対応付けている。これにより、クリップＡＶストリームへのアクセスポイントの時刻情報（タイムスタンプ）が与えられたときに、クリップＡＶストリームファイルの中でデータの読み出しを開始すべきデータアドレスを検索することが容易となり、ディスクのランダムアクセスをスムースに行うことができる。
【０１８４】
なお、この実施の一形態では、ブロックEP_map()において、エレメンタリストリーム毎の時刻情報と位置情報との組（第２のforループ内のフィールドPTS_EP_startとフィールドRPN_EP_startとの組）は、フィールドPTS_EP_startおよびRPN_EP_startの両方に対して昇順（または降順）に予め並べて登録するようにしている。換言すれば、時刻情報と位置情報とは、予め所定の方向に並べ替えられている。このため、このままのデータに対して二分木検索を実行することが可能である。
【０１８５】
なお、この発明の実施の一形態では、ビデオのエレメンタリストリームは、ＭＰＥＧ２－Ｖｉｄｅｏの規格に基づくエレメンタリストリームであるとして説明したが、これはこの例に限定されない。例えば、ビデオのエレメンタリストリームは、ＭＰＥＧ４－ＶｉｓｕａｌやＭＰＥＧ４－ＡＶＣによるものでもよい。また、オーディオのエレメンタリストリームは、ＡＴＲＡＣオーディオのエレメンタリストリームであるとして説明したが、これもこの例に限らず、例えばＭＰＥＧ１／２／４オーディオにも適用可能である。
【０１８６】
７．ディスク再生装置について
次に、この発明の実施の一形態を適用可能なディスク再生装置について説明する。図２９は、この発明を適用可能なディスク再生装置１００の一例の構成を概略的に示す。バス１１１に対して、ＣＰＵ(Central Processing Unit)１１２、メモリ１１３、ドライブインターフェイス１１４、入力インターフェイス１１５、ビデオデコーダ１１６、オーディオデコーダ１１７、ビデオ入出力インターフェイス１１８およびオーディオ入出力インターフェイス１１９がそれぞれ接続される。このディスク再生装置１００の各部は、バス１１１を介してビデオストリーム、オーディオストリーム、各種コマンドやデータなどを互いにやりとりできるようになっている。
【０１８７】
ドライブインターフェイス１１４には、さらに、ディスクドライブ１０２が接続される。ディスクドライブ１０２は、ドライブインターフェイス１１４を介してバス１１１とデータやコマンドのやりとりを行う。
【０１８８】
ＣＰＵ１１２は、ＲＯＭ(Read Only Memory)およびＲＡＭ(Random Access Memory)を有し（図示しない）、ＲＯＭに予め記憶されたプログラムやデータに従い、バス１１１を介してこのディスク再生装置１００の各部とデータやコマンドのやりとりを行い、このディスク装置１００の全体を制御する。ＲＡＭは、ＣＰＵ１１２のワークメモリとして用いられる。
【０１８９】
なお、図２９では省略されているが、ディスク再生装置１００は、データの書き換えが可能で、且つ、ディスク再生装置１００の電源をＯＦＦとした後も記憶内容が保持される、フラッシュメモリなどの不揮発性メモリを持つことができる。不揮発性メモリは、例えばバス１１１に接続され、ＣＰＵ１１２による不揮発性メモリに対するデータの書き込みや、この不揮発性メモリからのデータの読み出しが可能なようにされている。
【０１９０】
入力インターフェイス１１５は、ユーザにより実際に入力操作が行われる入力装置からの入力信号が供給される。入力装置は、例えば、赤外線信号などで遠隔的にディスク再生装置１００を操作するリモートコントロールコマンダや、このディスク再生装置１００に

直接的に設けられたキーなどである。入力インターフェイス１１５は、これらの入力装置から供給された入力信号を、ＣＰＵ１１２に対する制御信号に変換して出力する。

【０１９１】

ディスク１０１は、図１８以降で説明したようなフォーマットで以て、プレイリスト、スクリプトプログラム、クリップインフォメーションファイル、クリップＡＶストリームファイルなどが記録されている。ディスク１０１がディスクドライブ１０２に装填されると、自動再生またはユーザの入力操作に従いディスク１０１が再生される。ディスク１０１から読み出されたスクリプトファイルやプレイリストファイル、クリップインフォメーションファイルは、ＣＰＵ１１２に供給され、例えばＣＰＵ１１２が有するＲＡＭに記憶される。ＣＰＵ１１２は、ＲＡＭに記憶されたこれらのデータやスクリプトプログラムに基づき、ディスク１０１からクリップＡＶストリームファイルを読み出す。

【０１９２】

ディスク１０１から読み出されたクリップＡＶストリームファイルは、メモリ１１３に一旦格納される。ビデオデコーダ１１６は、ＣＰＵ１１２の命令に基づき、メモリ１１３に格納されたクリップＡＶストリームファイルのビデオストリームや字幕ストリームをデコードする。デコードされたビデオデータや字幕データは、例えばＣＰＵ１１２によりそれぞれ拡大、縮小処理などの画像処理を施されると共に、合成、加算処理を施され、１本のビデオデータとされる。これらの画像処理は、これに限らず、ビデオデコーダ１１６やビデオ出力インターフェイス１１８において行うこともできる。このビデオデータは、メモリ１１３にバッファリングされ、ビデオ出力インターフェイス１１８に供給される。ビデオ出力インターフェイス１１８は、例えば、供給されたビデオデータをアナログビデオ信号に変換して、ビデオ出力端子１２０に導出する。

【０１９３】

同様に、オーディオデコーダ１１７は、ＣＰＵ１１２の命令に基づき、メモリ１１３に格納されたクリップＡＶストリームファイルのオーディオストリームをデコードする。デコードされたオーディオデータは、メモリ１１３にバッファリングされ、オーディオ出力インターフェイス１１９に供給される。オーディオ出力インターフェイス１１９は、供給されたオーディオデータを、例えばアナログオーディオ信号に変換してオーディオ出力端子１２１に導出する。

【０１９４】

なお、ここでは、図２９に示される各部がそれぞれ独立したハードウェアで構成されているように説明したが、これはこの例に限定されない。例えば、ビデオデコーダ１１６および／またはオーディオデコーダ１１７は、ＣＰＵ１１２上で動作するソフトウェアにより構成することができる。

【０１９５】

また、上述したディスク再生装置１００は、ＣＰＵ１１２およびメモリを備え、プログラムにより動作するので、一種のコンピュータ装置として考えることができる。

【０１９６】

図３０は、図２９に示したディスク再生装置１００における動作をより詳細に説明するための機能ブロック図である。ディスク再生装置１００は、概略的には、オペレーションシステム２０１と、ビデオコンテンツ再生部２１０とからなる。ビデオコンテンツ再生部２１０は、実質的には、オペレーションシステム２０１上で動作するソフトウェアプログラムである。これに限らず、ビデオコンテンツ再生部２１０は、ソフトウェアとハードウェアが統合的に動作するものとしてもよい。以下では、ビデオコンテンツ再生部２１０がソフトウェアであるものとして説明する。なお、図３０では、ディスクドライブ１０２は、省略されている。

【０１９７】

オペレーションシステム２０１は、ディスク再生装置１００に電源が投入されるとＣＰＵ１１２において最初に起動し、各部の初期設定など必要な処理を行い、アプリケーションプログラム（ここではビデオコンテンツ再生部２１０）をＲＯＭから呼び出す。オペレ

ーションシステム２０１は、ビデオコンテンツ再生部２１０の動作中に、ビデオコンテンツ再生部２１０に対して、ディスク１０１からのファイルの読み出しやファイルシステムの解釈といった、基本的なサービスを提供する。例えば、オペレーションシステム２０１は、ビデオコンテンツ再生部２１０から渡されたファイル読み出しリクエストに応じて、ドライブインターフェイス１１４を介してディスクドライブ１０２を制御し、ディスク１０１に記録されているデータを読み出す。読み出されたデータは、オペレーションシステム２０１の制御により、ビデオコンテンツ再生部２１０に渡される。

【０１９８】

また、オペレーションシステム２０１は、マルチタスク処理機能を備え、複数のソフトウェアモジュールを、例えば時分割制御により見かけ上並列的に制御することができる。すなわち、図３０に一例が示される、ビデオコンテンツ再生部２１０を構成する各モジュールは、オペレーションシステム２０１のマルチタスク処理機能により、全て、並列的な動作が可能である。

【０１９９】

以下、ビデオコンテンツ再生部２１０の動作について、より具体的に説明する。ビデオコンテンツ再生部２１０は、内部にさらに幾つかのモジュールを有しており、下記の機能を実現する。

（１）装填されたディスク１０１がＵＭＤビデオの規格に準じたディスク（以下、ＵＭＤビデオディスクと呼ぶ）であるか否かを判断する。

（２）装填されたディスク１０１がＵＭＤビデオディスクであると判断した場合、ディスク１０１からリソースファイルを読み出して、スクリプト制御モジュール２１１に渡す。

（３）装填されたディスク１０１がＵＭＤビデオディスクであると判断した場合、さらに、データベースを構成するファイル（プレイリストファイル、クリップインフォメーションファイルなど）を読み出して、プレーヤ制御モジュール２１２に渡す。

【０２００】

以下、ビデオコンテンツ再生部２１０の各モジュールの動作について説明する。

【０２０１】

スクリプト制御モジュール２１１は、受け取ったリソースファイルを、例えばＣＰＵ１１２の図示されないＲＡＭの所定領域に記憶する。ＣＰＵ１１２（スクリプト制御モジュール２１１）は、このＲＡＭに記憶されたスクリプトプログラムを読み込み、解釈して実行する。リソースファイルをメモリ１１３の所定領域に記憶させ、必要に応じてＣＰＵ１１２の図示されないＲＡＭに読み込むようにしてもよい。

【０２０２】

プレーヤモデルの説明で既に述べたように、メニュー画面などの画像の作成および出力や、ユーザ入力に応じたカーソル移動、メニュー画面の変更といったＧＵＩは、スクリプトプログラムによりグラフィクス処理モジュール２１９を制御することで実現する。このとき、メモリ１１３上のリソースファイルに格納された画像データやサウンドデータが用いられ、メニュー画面などの作成が行われる。また、スクリプト制御モジュール２１１は、スクリプトプログラムの実行により、プレーヤ制御モジュール２１２の制御などが可能である。

【０２０３】

プレーヤ制御モジュール２１２は、ディスク１０１から読み出された、プレイリストファイル"PLAYLIST.DAT"や、クリップインフォメーションファイル"XXXXX.CLP"といったファイルに格納されたデータベース情報を参照して、ディスク１０１に記録されているビデオコンテンツの再生に関わる、以下のような制御を行う。

（１）プレイリストやクリップインフォメーションといったデータベース情報を解析する。

（２）コンテンツデータ供給モジュール２１３、デコード制御モジュール２１４およびバッファ制御モジュール２１５を制御する。

（３）スクリプト制御モジュール２１１または入力インターフェイス１１５からの指示に

従い、再生、再生停止、再生一時停止といったプレーヤの状態遷移制御や、ストリーム切り替えなどの再生制御処理を行う。
（４）デコード制御モジュール２１４から、再生中のビデオストリームについて、時刻情報を取得し、時刻表示やマークイベントの生成などを行う。
【０２０４】
コンテンツデータ供給モジュール２１３は、プレーヤ制御モジュール２１２の指示に従い、ディスク１０１からクリップＡＶストリームファイルといったコンテンツデータを読み出し、バッファ制御モジュール２１５に渡す。バッファ制御モジュール２１５は、渡されたコンテンツデータをバッファの実体２１５Ａとしてのメモリ１１３に溜め込む。コンテンツデータ供給モジュール２１３は、バッファ制御モジュール２１５を制御し、ビデオデコーダ制御モジュール２１６、オーディオデコーダ制御モジュール２１７および字幕デコーダ制御モジュール２１８からの要求に従い、メモリ１１３に溜め込まれたコンテンツデータを、これらのモジュール２１６、２１７および２１８に所定に供給する。また、コンテンツデータ供給モジュール２１３は、バッファ制御モジュール２１５により溜め込まれるコンテンツデータの量を所定に制御するように、ディスク１０１からコンテンツデータの読み込みを行う。
【０２０５】
デコード制御モジュール２１４は、プレーヤ制御モジュール２１２の指示に従い、ビデオデコーダ制御モジュール２１６、オーディオデコーダ制御モジュール２１７および字幕デコーダ制御モジュール２１８の動作を制御する。また、デコード制御モジュール２１４は、内部に時計機能を有し、ビデオデータとオーディオデータとが同期的に出力されるように、各デコーダ制御モジュール２１６、２１７および２１８の動作を制御する。
【０２０６】
バッファ制御モジュール２１５は、バッファの実体２１５Ａとして、メモリ１１３の一部を排他的に用いる。また、バッファ制御モジュール２１５は、データ先頭ポインタおよびデータ書き込みポインタを記憶する。バッファ制御モジュール２１５は、さらに、内部モジュールとしてビデオ読み出し機能、オーディオ読み出し機能および字幕読み出し機能を有する。ビデオ読み出し機能の内部には、ビデオ読み出しポインタを有する。また、ビデオ読み出し機能の内部には、アクセスユニット情報である情報au_information()を蓄積するためのレジスタを備える。オーディオ読み出し機能の内部には、オーディオ読み出しポインタを有する。字幕読み出し機能の内部には、字幕読み出しポインタと字幕読み出し機能フラグとを有する。字幕読み出し機能フラグは、書き込む値に応じて字幕読み出し機能の有効／無効を制御する。例えば、字幕読み出し機能フラグに"１"を書き込むと、字幕読み出し機能が有効とされ、"０"を書き込むと、字幕読み出し機能が無効とされる。
【０２０７】
バッファ制御モジュール２１５の内部モジュールであるビデオ読み出し機能、オーディオ読み出し機能および字幕読み出し機能は、さらに、ビデオストリーム、オーディオストリームおよび字幕ストリームが多重化されたクリップＡＶストリームから、それぞれのストリームを分離するデマルチプレクサ機能を有する。この発明の実施の一形態では、ＭＰＥＧ２システムのプログラムストリームの形式で複数のエレメンタリストリームが時分割多重されて、クリップＡＶストリームが形成されている。したがって、ビデオ読み出し機能、オーディオ読み出し機能および字幕読み出し機能は、ＭＰＥＧ２システムのプログラムストリームに対するデマルチプレクサ機能を有する。
【０２０８】
このため、ビデオ読み出し機能は、ストリーム内に所定に配置されるフィールドstream_id（図２５参照）の値を読み取り、保持する。同様に、オーディオ読み出し機能および字幕読み出し機能は、フィールドstream_idおよびフィールドprivate_stream_id（図２５参照）の値を読み取り、保持する。これらフィールドstream_idやフィールドprivate_stream_idの値は、供給されたビットストリームを解析する際に用いる。
【０２０９】

ビデオデコーダ制御モジュール２１６は、メモリ１１３からビデオストリームの単一のビデオアクセスユニットを読み出してビデオデコーダ１１６に供給するように、バッファ制御モジュール２１５内のビデオ読み出し機能に対して指示を出す。そして、ビデオデコーダ制御モジュール２１６は、ビデオデコーダ１１６を制御して、ビデオデコーダ１１６に供給されたビデオストリームをアクセスユニット単位でデコードする。ビデオストリームをデコードして生成されたビデオデータは、グラフィクス処理モジュール２１９に供給される。

【０２１０】

同様に、オーディオデコーダ制御モジュール２１７は、メモリ１１３からオーディオストリームの単一のオーディオアクセスユニットを読み出してオーディオデコーダ１１７に供給するように、バッファ制御モジュール２１５内のオーディオ読み出し機能に対して指示を出す。なお、この実施の一形態では、オーディオストリームを構成するアクセスユニット（オーディオフレーム）は、既知の固定長とする。そして、オーディオデコーダ制御モジュール２１７は、オーディオデコーダ１１７を制御して、オーディオデコーダ１１７に供給されたオーディオストリームをアクセスユニット単位でデコードする。オーディオストリームをデコードして生成されたオーディオデータは、オーディオ出力モジュール２４２に供給される。

【０２１１】

さらに、字幕デコーダ制御モジュール２１８は、メモリ１１３から字幕ストリームの単一の字幕アクセスユニットを読み出して字幕デコーダ制御モジュール２１８に供給するように、バッファ制御モジュール２１５内の字幕読み出し機能に対して指示を出す。なお、この実施の一形態では、字幕ストリームを構成する字幕アクセスユニットは、ユニットの先頭に当該ユニットの長さ情報が格納されている。字幕デコーダ制御モジュール２１８は、字幕デコード機能を有し、供給された字幕ストリームをデコードすることができる。字幕デコーダ制御モジュール２１８の字幕デコード機能により字幕ストリームがデコードされた字幕の画像データは、グラフィクス処理モジュール２１９に供給される。

【０２１２】

グラフィクス処理モジュール２１９は、上述したように、ビデオデコーダ制御モジュール２１６の制御に基づきビデオデコーダ１１６でデコードされたビデオデータと、字幕デコーダ制御モジュール２１８によりデコードされた字幕の画像データとが供給される。グラフィクス処理モジュール２１９は、供給されたこれらビデオデータに対して字幕の画像データを所定に加算し、出力するためのビデオ信号を生成する。グラフィクス処理モジュール２１９では、さらに、スクリプト制御モジュール２１１やプレーヤ制御モジュール２１２の指示に従い、メニュー画像やメッセージ画像を生成し、出力ビデオ信号に対して合成（オーバーレイ）する。

【０２１３】

例えば、グラフィクス処理モジュール２１９は、供給された字幕の画像データに対して、スクリプト制御モジュール２１１からの指示に従い拡大処理や縮小処理を行い、ビデオデータに対して所定に加算する。

【０２１４】

また、グラフィクス処理モジュール２１９は、予め指定された出力ビデオデバイスのアスペクトレシオと、ディスク１０１から再生されたコンテンツ内で指定された出力アスペクトレシオとに基づき、出力信号のアスペクト変換を行う。例えば、出力ビデオデバイスのアスペクトレシオが１６：９である場合、出力アスペクトレシオが１６：９であれば、ビデオデータをそのまま出力し、出力アスペクトレシオが４：３であれば、出力されるビデオデータを、画像の高さが出力ビデオデバイスの画面高さに一致するようにスクイーズ（縮小）処理し、画像の左右に黒画像を挿入して出力する。出力ビデオデバイスが４：３である場合には、出力アスペクトレシオが４：３であれば、ビデオデータをそのまま出力し、出力アスペクトレシオが１６：９であれば、出力されるビデオデータを、画像の幅が出力ビデオデバイスの画面幅に一致するようにスクイーズ処理し、画像の上下に黒画像を

挿入して出力する。

【０２１５】

グラフィクス処理モジュール２１９は、さらに、プレーヤ制御モジュール２１２からの要求に応じて、現在処理中のビデオ信号をキャプチャし、プレーヤ制御モジュール２１２に返すような処理も行う。

【０２１６】

ビデオ出力モジュール２４１は、メモリ１１３の一部を排他的に占有してＦＩＦＯ(First In First Out)のバッファとして用い、グラフィクス処理モジュール２１９により処理されたビデオデータをこのバッファに一時的に溜め込み、所定のタイミングで読み出す制御を行う。バッファから読み出されたビデオデータは、ビデオ出力インターフェイス１１８から出力される。

【０２１７】

オーディオ出力モジュール２４２は、メモリ１１３の一部を排他的に占有してＦＩＦＯのバッファとして用い、オーディオデコーダ１１９から出力されたオーディオデータをこのバッファに溜め込み、所定のタイミングで読み出す制御を行う。バッファから読み出されたオーディオデータは、オーディオ出力インターフェイス１１９から出力される。

【０２１８】

また、オーディオ出力モジュール２４２は、コンテンツのオーディオモードがデュアルモノ（例えば２ヶ国語）であった場合、予め指定された音声出力モードに従いオーディオデータを出力する。音声出力モードが「主音声」に指定されている場合、例えばメモリ１１３において左チャンネルのオーディオデータを右チャンネルにもコピーして、２チャンネルの出力を両方とも左チャンネルのオーディオデータとして出力する。音声出力モードが「副音声」であった場合には、例えばメモリ１１３において右チャンネルのオーディオデータを左チャンネルにもコピーして、２チャンネルの出力を両方とも右チャンネルのオーディオデータとして出力する。音声出力モードが「主・副音声」である場合や、コンテンツがステレオである場合には、オーディオデータをそのまま出力する。

【０２１９】

このような音声出力モードの設定は、ビデオコンテンツ再生部２１０が生成するメニュー画面などにより、ユーザが対話的に行うことができるようになっている。

【０２２０】

不揮発性メモリ制御モジュール２５０は、プレーヤ制御モジュール２１２からの指示により、ビデオコンテンツ再生部２１０が終了しても消去されない領域へのデータの書き込みや、当該領域からのデータの読み出しを行う。タイトル識別ＩＤ(Title_ID)をキーとして、データSaved_Player_StatusおよびデータSaved_User_Dataの組を複数件、当該領域に記憶する機能を有する。データSaved_Player_Statusとして、プレーヤ制御モジュール２１２の持つデータBackup_Player_Statusが記憶される。このデータBackup_Player_Statusは、例えば上述したプレーヤステータス３２３Ｂの、プレーヤ制御モジュール２１２が終了する直前のデータに対応し、データSaved_Player_Statusは、リジュームインフォメーション３２４に対応する。また、データSaved_User_Dataとして、プレーヤ制御モジュール２１２が持つデータUser_Dataが記憶される。データUser_Dataは、ユーザによりプレーヤ制御モジュール２１２に対して設定された所定のデータである。

【０２２１】

不揮発性メモリ制御モジュール２５０は、ディスク装置１００が有する不揮発性メモリの所定領域にこれらデータSaved_Player_StatusおよびデータSaved_User_Dataの組を、ディスク１０１のタイトルＩＤと関連付けて記憶する。不揮発性メモリ制御モジュール２５０がデータを記憶する記憶媒体は、不揮発性メモリに限らず、例えばハードディスクなどでもよい。

【０２２２】

８．ムービープレーヤの状態遷移モデルについて

８－１．ムービープレーヤの状態の定義について

次に、この発明の実施の一形態によるムービープレーヤ３００の状態遷移モデルについて、より詳細に説明する。この発明では、ムービープレーヤ３００の状態を、ムービープレーヤ３００の内部状態においてのみ、定義する。すなわち、この発明では、ムービープレーヤ３００の状態を、ムービープレーヤ３００それ自身の動作および機能に基づきそれぞれ定義する。

【０２２３】

より具体的には、動作面において、プレイリスト再生の観点から、ムービープレーヤ３００がプレイ状態にあるかストップ状態にあるかの２状態を定義する。また、機能面において、ムービープレーヤ３００がネイティブ実装プラットフォーム３０１からの制御コマンドを受け付けるか否かの２状態を定義する。

【０２２４】

図３１は、この発明によるムービープレーヤ３００の状態の定義を概念的に示す。ムービープレーヤ３００の動作面から見た状態について説明する。図３も参照し、ムービープレーヤ３００は、プレイリスト再生の観点から、プレイ状態およびストップ状態の何れかの状態にある。プレイ状態は、ムービープレーヤ３００がプレイリストを選択し、選択されたプレイリストの再生を行っている状態である。一方、ストップ状態は、ムービープレーヤ３００がプレイリストを再生していない状態である。ストップ状態では、プレイリストは、選択されていない。換言すれば、ムービープレーヤ３００のプレイバックモジュール３２１でクリップＡＶストリームがデコードされている状態がプレイ状態であって、デコードされていない状態がストップ状態であるといえる。

【０２２５】

なお、プレイ状態は、さらに幾つかの状態に細分化される。例えば、順方向１倍速の通常再生、順方向および逆方向への１倍速以外の再生速度で再生する変速再生、一時停止といった、各種の再生状態をプレイ状態に含む。コマ送りおよびコマ戻し再生は、通常再生と一時停止とを交互に行うことで実現される。このような、プレイリストの再生中は、ムービープレーヤ３００は、プレイ状態にあることと同義である。

【０２２６】

ムービープレーヤ３００の機能面から見た状態について説明する。ムービープレーヤ３００は、機能の観点から、ネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付けるモード（ノーマルモードと呼ぶ）と、制御コマンド３１１を無視するモード（メニューモードと呼ぶ）との２つの動作モードを有する。これらムービープレーヤ３００の２つの動作モードを、それぞれムービープレーヤ３００の状態として定義する。

【０２２７】

ノーマルモードは、ムービープレーヤ３００の動作を、スクリプトレイヤ３０２のスクリプトプログラムを介さずに、ユーザ入力３１０によって制御することができる。

【０２２８】

一方、メニューモードは、ムービープレーヤ３００が制御コマンド３１１を受け付けない。ムービープレーヤ３００は、スクリプトレイヤ３０２からのメソッド３１３のみを受け付ける。そのため、ムービープレーヤ３００の動作を、スクリプトレイヤ３０２のスクリプトプログラムで全て、制御できる。例えば、ユーザ入力３１０は、ネイティブ実装プラットフォーム３０１からのイベント３１４として、スクリプトレイヤ３０２に渡される。スクリプトレイヤ３０２のスクリプトプログラムは、このイベント３１４に応じたメソッド３１３を用いて、ムービープレーヤ３００の動作を制御する。

【０２２９】

すなわち、メニューモードを用いることで、ムービープレーヤ３００の動作をコンテンツ制作者側が制御することができる。また、メニューモードを利用することで、少ない種類のキーで多彩な制御を実現することができる。

【０２３０】

このように、ムービープレーヤ３００は、動作面ではプレイ状態とストップ状態の２状態を有すると共に、機能面ではノーマルモードとメニューモードの２つの動作モードを有

する。したがって、ムービープレーヤ３００としては、これら動作面の２状態と機能面の２動作モードとをそれぞれ組み合わせた、４状態が定義される。すなわち、ムービープレーヤ３００は、生成されてから消滅するまでの間、この４状態のうち何れかの状態に属する。ムービープレーヤ３００の生成および消滅については、後述する。

【０２３１】

なお、ムービープレーヤ３００に対して現在の状態と異なる状態に遷移することを指示するメソッド３１３を発行すると、モデル上では、ムービープレーヤ３００は、発行されたメソッド３１３に応じて即座に状態を遷移させる。実際の機器においては、ムービープレーヤ３００に対してあるメソッド３１３を発行してから、ムービープレーヤ３００が当該メソッド３１３に応じた状態遷移を完了するまでの時間は、当該機器の実装に依存することになる。

【０２３２】

また、ある状態にあるムービープレーヤ３００に対して、当該状態と同一の状態に状態遷移させることを指示するメソッド３１３を発行しても、ムービープレーヤ３００の状態は、変化しない。例えば、既にノーマルモードで、且つ、ストップ状態にあるムービープレーヤ３００に対して、ムービープレーヤ３００の状態をノーマルモードで且つストップ状態に遷移させるようなメソッド３１３を発行しても、ムービープレーヤ３００の状態は、変化しない。

【０２３３】

さらに、一時停止（ポーズ）状態は、プレイ状態の一種である。ストップ状態から一時停止状態に遷移させるためには、一時停止を指定する値pauseModeを引数としたメソッドplay()を用いる。

【０２３４】

次に、ムービープレーヤ３００の２状態と２動作モードとを組み合わせた４状態それぞれの状態と、４状態間の状態遷移について説明する。以下では、ムービープレーヤ３００の状態を機能面で分類した状態のうち、ノーマルモードを「normal」とし、メニューモードを「menu」とする。また、ムービープレーヤ３００の状態を動作面で分類した状態のうち、プレイ状態を「play」とし、ストップ状態を「stop」とする。そして、ムービープレーヤ３００のモード(mode)および状態(status)の組み合わせを、便宜的に、状態{mode,status}と表す。また、以下では、ムービープレーヤ３００における状態の変化とモードの切り換えとを、共に状態遷移と呼ぶ。

【０２３５】

図３１からも分かるように、ムービープレーヤ３００の状態遷移は、自らの状態への遷移も含めると、４×４＝１６通りが存在する。これらの状態遷移は、スクリプトレイヤ３０２からムービープレーヤ３００に渡されるメソッド３１３によって引き起こされる。すなわち、ムービープレーヤ３００における状態遷移は、ムービープレーヤ３００の外部から引き起こされ、スクリプトレイヤ３０２からのメソッドによる指示無しに、ムービープレーヤ３００の内部で自動的に状態遷移が発生することが無い。また、ネイティブ実装プラットフォーム３０１からの制御コマンドでも、ムービープレーヤ３００において状態遷移が発生することが無い。

【０２３６】

なお、この実施の一形態では、メソッド３１３の引数の組み合わせに制限があるため、ムービープレーヤ３００において可能な１６通りの状態遷移の全てをメソッドで発生させることは、できない。

【０２３７】

以下、ムービープレーヤ３００が取り得る４状態（状態{Menu,Stop}、状態{Normal,Stop}、状態{Menu,Play}および状態{Normal,Play}）について、それぞれ説明する。

【０２３８】

（１）状態{Menu,Stop}

ムービープレーヤ３００がプレイリストの再生を行っていないストップ状態で、且つ、

ネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付けない状態である。この状態は、例えば背景に動画が再生されていないメニュー画面などで用いられる。

【０２３９】

ムービープレーヤ３００の生成直後にスクリプトプログラムからの制御を確実なものとするためには、その時点でネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付けないようにすることが有効である。そのため、ムービープレーヤ３００の生成直後は、この状態{Menu,Stop}になるように定める。

【０２４０】

（２）状態{Normal,Stop}

ムービープレーヤ３００がプレイリストの再生を行っていないストップ状態で、且つ、ネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付ける状態である。この状態は、例えば動画を再生していない状態で用いられる。この状態は、制御コマンド３１１を受け付けるので、ムービープレーヤ３００の生成直後には用いないのが好ましい。

【０２４１】

（３）状態{Menu,Play}

ムービープレーヤ３００がプレイリストの再生を行っているプレイ状態で、且つ、ネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付けない状態である。この状態は、例えば背景に動画が再生されているメニュー画面などで用いられる。

【０２４２】

（４）状態{Normal,Play}

ムービープレーヤ３００がプレイリストの再生を行っているプレイ状態で、且つ、ネイティブ実装プラットフォーム３０１からの制御コマンド３１１を受け付ける状態である。この状態は、例えば本編の再生中などに用いられる。

【０２４３】

上述した、ムービープレーヤ３００が生成される際のモデルについて、概略的に説明する。ムービープレーヤ３００の生成は、例えば上述したように、ディスク再生装置１００に電源が投入され、ＣＰＵ１１２においてオペレーションシステム２０１が起動されると、各部の初期設定など必要な処理を行うと共に、ビデオコンテンツ再生部２１０をＲＯＭから呼び出す。このビデオコンテンツ再生部２１０がＣＰＵ１１２により実行されることでなされる。ディスク再生装置１００において電源がＯＦＦ状態とされると、ムービープレーヤ３００が消滅される。

【０２４４】

ここで、ムービープレーヤ３００は、暗黙の(implicit)オブジェクトが生成されているものと見做され、スクリプトプログラムにおいて明示的にムービープレーヤ３００の生成を行う必要はない。

【０２４５】

上述したように、ムービープレーヤ３００生成直後の状態は、メニューモードのストップ状態（状態{Menu,Stop}）とされる。ムービープレーヤ３００の生成直後では、例えばムービープレーヤ３００が持つ下記のプロパティが不定値となる。

プロパティaudioFlag
プロパティaudioNumber
プロパティchapterNumber
プロパティplayListNumber
プロパティplaySpeed
プロパティsubtitleFlag
プロパティsubtitleNumber
プロパティvideoNumber

【０２４６】

なお、前回再生を停止された位置から再生を開始する「続き再生機能」を備えるＵＭＤビデオプレーヤでは、ムービープレーヤ３００の初期化において、例えば上述の各プロパティについて、各プロパティのデフォルト値ではなく不揮発性メモリに保存された値を用いて値を設定することができる。例えば、リジュームインフォメーション３２４を利用できる。

【０２４７】

８－２．ムービープレーヤに状態遷移を発生させるメソッドについて

次に、ムービープレーヤ３００に状態遷移を発生させるメソッド３１３について説明する。図３２は、現在の状態{Mode,Status}と、メソッド３１３によって状態遷移した後の状態{Mode,Status}を、ムービープレーヤ３００の４状態のそれぞれについて組み合わせて示す。図３２から分かるように、ムービープレーヤ３００に対して状態遷移を発生させるメソッド３１３として、メソッドstop()、メソッドplay()およびメソッドresume()が用意される。なお、メソッドresume()は、リジュームインフォメーション３２４が存在するか否かで動作が変化する。

【０２４８】

メソッドstop()について説明する。メソッドstop()は、ムービープレーヤ３００の状態の如何に関わらず、ムービープレーヤ３００をストップ状態に遷移させる。メソッドstop()は、引数にモードを指定することができ、ストップ状態への遷移と同時に、ムービープレーヤ３００のモードを変更することができる。後述するように、ある条件を満たしてメソッドstop()が実行されると、プレーヤステータス３２３Ｂがバックアップされ、リジュームインフォメーション３２４として保持される。

【０２４９】

メソッドplay()について説明する。メソッドplay()は、ムービープレーヤ３００をプレイ状態に遷移させる。メソッドplay()は、引数にモードを指定することができ、プレイ状態への遷移と同時に、ムービープレーヤ３００のモードを変更することができる。後述するように、ある条件を満たしてメソッドplay()が実行されると、プレーヤステータス３２３Ｂがバックアップされると共に、リジュームインフォメーション３２４が保持される。

【０２５０】

メソッドresume()について説明する。メソッドresume()は、リジュームインフォメーション３２４をプレーヤステータス３２３Ｂにリストアして再生復帰させるメソッドである。すなわち、メソッドresume()は、ムービープレーヤ３００に対して、リジュームインフォメーション３２４で示された位置からの再生を開始させる。リジュームインフォメーション３２４が存在しない状態でメソッドresume()が実行されても、ムービープレーヤ３００の状態は変化しない。

【０２５１】

メソッドresume()によってリジュームインフォメーション３２４がリストアされる条件は、次の通りである。メソッドresume()が実行された際に、リジュームインフォメーション３２４が存在し、且つ、現在の状態が状態{Normal,Play}でなければ、ムービープレーヤ３００は、リジュームインフォメーション３２４をリストアする。換言すれば、メソッドresume()が実行された際に、リジュームインフォメーション３２４が存在し、且つ、現在の状態が状態{Menu,Stop}、状態{Normal,Stop}および状態{Menu,Play}のうち何れかであれば、ムービープレーヤ３００は、状態{Normal,Play}に遷移すると同時に、リジュームインフォメーション３２４をリストアする。

【０２５２】

メソッドplay()は、複数の引数を持つ。ここでは、説明のため、メソッドplay()は、引数pauseMode、引数menuModeおよび引数playListNumberの３種類の引数を持つものとする。実際には、さらに多くの引数が定義される。

【０２５３】

引数pauseModeは、プレイ状態における再生の状況を指定し、値「x1」、値「pause」および値「-1」の何れかの値を取り得る。値「x1」は、通常速度の順方向再生を指定する。

値「pause」は、一時停止を指定する。値「-1」は、現状の再生速度を維持するよう指定する。このように、引数pauseModeは、メソッドplay()実行後のムービープレーヤ３００のプレイ状態の詳細を設定する。なお、一時停止を指定した場合、引数で指定される位置のピクチャ、または引数による指定がない場合は、別途定められた選択ルールで指定される位置のピクチャが表示されて一時停止した状態になる。

【０２５４】

引数menuModeは、ムービープレーヤ３００のモード（ノーマルモードおよびメニューモード）を指定し、値「Normal」、値「Menu」および値「-1」の何れかの値を取り得る。値「Normal」は、ノーマルモードへの切り換えを指定する。値「Menu」は、メニューモードへの切り換えを指定する。値「-1」は、現在のモードを維持するよう指定する。

【０２５５】

引数playListNumberは、再生するプレイリストの番号を指定する。引数playListNumberは、省略することができ、その場合には、現在選択しているプレイリストのまま変更がないことを表す。

【０２５６】

図３３を用いて、メソッドplay()を実行した際のムービープレーヤ３００の状態遷移の例について説明する。なお、図３３Ａ～図３３Ｅの各図において、左側は、ムービープレーヤ３００の現在の状態３４０Ａを示し、右側は、現在の状態３４０Ａのムービープレーヤ３００に対してスクリプトプログラムがメソッド３１３を発行することで遷移された遷移後の状態３４０Ｂを示す。また、状態３４０Ａおよび３４０Ｂの下に、その状態において指定されているプレイリスト番号（ＰＬ１、ＰＬ２）が示される。

【０２５７】

図３３Ａは、状態{Normal,Stop}の状態にあるムービープレーヤ３００に、メソッドplay(x1,Normal,PL2)を発行した場合の例である。メソッドplay(x1,Normal,PL2)に応じて、ムービープレーヤ３００の状態がプレイリスト番号「ＰＬ２」のプレイリストをノーマルモード且つ通常速度で再生する状態になることを表している。ムービープレーヤ３００は、状態{Normal,Stop}から状態{Normal,Play}に状態遷移している。

【０２５８】

図３３Ｂは、プレイリスト番号「ＰＬ１」のプレイリストを再生中に一時停止している、状態{Normal,Play}の状態にあるムービープレーヤ３００に、メソッドplay(x1,Normal,PL2)を発行した場合の例である。メソッドplay(x1,Normal,PL2)に応じて、ムービープレーヤ３００の状態がプレイリスト番号「ＰＬ２」のプレイリストをノーマルモード且つ通常速度で再生を開始する状態になることを表している。この場合、ムービープレーヤ３００の再生動作が一時停止から順方向の通常速度再生に変化するが、状態は、メソッドplay(x1,Normal,PL2)の発行の前後で状態{Normal,Play}のままであり、状態遷移は発生していない。

【０２５９】

図３３Ｃは、プレイリスト番号「ＰＬ１」のプレイリストを順方向の通常速度で再生中の、状態{Normal,Play}の状態にあるムービープレーヤ３００に、メソッドplay(-1,-1,PL2)を発行した場合の例である。メソッドplay(-1,-1,PL2)に応じて、ムービープレーヤ３００の状態がプレイリスト番号「ＰＬ２」のプレイリストをノーマルモード且つ通常速度で再生する状態になることを表している。この場合も、ムービープレーヤ３００で再生されるプレイリストが変更されるが、状態は、状態{Normal,Play}のままであり、状態遷移は発生していない。

【０２６０】

図３３Ｄは、プレイリスト番号「ＰＬ１」のプレイリストを順方向の通常速度で再生中の、状態{Normal,Play}の状態にあるムービープレーヤ３００に、メソッドplay(pause,-1,PL2)を発行した場合の例である。メソッドplay(pause,-1,PL2)に応じて、ムービープレーヤ３００は、プレイリスト番号「ＰＬ２」のプレイリストを選択すると共に、ノーマルモード且つプレイリスト番号「ＰＬ２」のプレイリストの先頭で一時停止している状態に

なることを表している。この場合も、ムービープレーヤ３００の再生動作が順方向の通常速度再生から一時停止に変化するが、状態は、状態{Normal,Play}のままであり、状態遷移は発生していない。

【０２６１】

図３３Ｅは、プレイリスト番号「ＰＬ１」のプレイリストを再生中に一時停止している、状態{Normal,Play}の状態にあるムービープレーヤ３００に、メソッドplay(-1,Menu)を発行した場合の例である。メソッドplay()において、引数playListNumberが省略されている。メソッドplay(-1,Menu)に応じて、ムービープレーヤ３００は、プレイリスト番号「ＰＬ１」のプレイリストを選択すると共に、メニューモード且つプレイリスト番号「ＰＬ１」のプレイリストの先頭で一時停止している状態になることを表している。ムービープレーヤ３００は、状態{Normal,Play}から状態{Menu,Stop}に状態遷移している。

【０２６２】

このように、ムービープレーヤ３００は、スクリプトプログラムからのメソッドplay()を受けて様々な動作を行い、その際、条件によっては状態遷移を生じる。コンテンツ制作者は、引数を変えたメソッドplay()をスクリプトプログラムに記述することで、ムービープレーヤ３００における様々な動作を実現することができる。

【０２６３】

なお、ムービープレーヤ３００は、スクリプトプログラムからメソッドplay()を実行することによってのみ、プレイリスト番号を選択したプレイリストの再生を開始する。プレイリストの再生開始とは、ストップ状態からプレイリストの再生を開始すること、あるいは、あるプレイリストの再生を中断して新たなプレイリストを選択し、選択されたプレイリストの再生を開始することを指す。

【０２６４】

スクリプトプログラムがムービープレーヤ３００に対し、引数付きのメソッドplay()を発行した際には、引数の値がプレーヤステータス３２３Ｂにセットされる。省略された引数については、各パラメータ毎に定められたルールに従い、デフォルト値や自動選択によって当該引数の値がセットされる。

【０２６５】

また、コンテンツ制作者の意図しない順序でプレイリストを再生できてしまっては問題があるため、ユーザ操作に起因する制御コマンド３１１では、プレイリスト番号を指定してのプレイリスト再生開始ができないようにされる。これは、この発明の実施の一形態によるムービープレーヤ３００の動作モデルにおける特徴の一つである。

【０２６６】

さらに、メソッドplay()の引数の値として、無効なプレイリストや、存在しない時刻が指定された場合は、そのメソッドplay()の実行は、失敗する。これは、スクリプトプログラムに誤りがあることを意味し、当該スクリプトプログラムが規格違反を含んでいることになる。このときのエラーハンドリングは、ムービープレーヤ３００の実装依存になる。

【０２６７】

ここで、プレイアイテム間の再生について説明する。ムービープレーヤ３００は、一旦プレイリストの再生を開始すると、そのプレイリストの最後まで再生を継続する。１つのプレイリストの先頭から最後までの再生は、ユーザ操作や、スクリプトプログラムからの制御を要しない。ムービープレーヤ３００は、図３４に概略的に示されるように、プレイリストを構成するプレイアイテムを、プレイリストファイル"PLAYLIST.DAT"（図１９参照）で指定された通りに順次、再生していく。プレイリストを構成するプレイアイテムは、イベントハンドラによる制御無しで、連続的に再生される。

【０２６８】

プレイアイテムを再生し終えてから次のプレイアイテムを再生するまでの間の動作は、ムービープレーヤ３００の実装依存であり、フォーマットでは規定しない。例えば、プレイアイテムの最後のピクチャを表示し続けるか、直ちに黒画面にしてしまうか、といったプレイアイテム間の動作は、実装依存となる。ただし、プレイアイテムのＩＮ点をランダ

ムアクセスポイント（エントリポイント：図２８参照）に設定するなどのオーサリング上の工夫を行うことにより、プレイアイテム間でのギャップ時間ができるだけ短くなるような配慮をすることは、可能である。

【０２６９】

８－３．プレイリスト再生中のムービープレーヤの動作について

次に、プレイリスト再生中のムービープレーヤ３００の動作について説明する。例えば２倍速再生、３倍速再生といった高速再生や、１／２倍速再生のような低速再生、また、逆方向再生などの、ユーザによる変速再生指示は、ユーザ入力３１０としてネイティブ実装プラットフォーム３０１に対して入力され、このユーザ入力３１０に応じて、実装に依存した制御コマンド３１１の形でネイティブ実装プラットフォーム３０１からムービープレーヤ３００に伝えられる。

【０２７０】

変速再生時の速度は、ムービープレーヤ３００の実装に依存する。例えば、速度指定が可能なネイティブ実装プラットフォーム３０１の命令としてムービープレーヤ３００に伝える、「より速く」か「より遅く」を引数としてムービープレーヤ３００に渡し、ムービープレーヤ３００が実現可能な速度に変換する、などの変速再生の実現方法が考えられ、どの方法を用いるかは、ムービープレーヤ３００の実装に依存する。スクリプトプログラムは、メソッドgetPlayerStatus()により、ムービープレーヤ３００が決めた速度を知ることができる。

【０２７１】

一方、スクリプトプログラムからムービープレーヤ３００に対するメソッドplay()では、引数で速度を指定することが出来ない。メソッドplay()は、一時停止（引数「pause」）および通常速度再生（引数「x1」）の２通りのみが指定できる。

【０２７２】

ムービープレーヤ３００は、プレイリストを順方向で変速再生しているときにプレイアイテムの終わりに到達した場合、次のプレイアイテムの再生を行う。このとき、ムービープレーヤ３００は、当該次のプレイアイテムを前のプレイアイテムと同じ向き、同じ速度で再生し、変速再生を継続する。

【０２７３】

図３５は、プレイリストの再生中にプレイリストの始点、終点に到達したときのムービープレーヤ３００の一例の動作を示す。ムービープレーヤ３００は、順方向で再生中にプレイリストの最後に到達した場合、最後のピクチャを表示したまま一時停止となる。最後のピクチャを消去するには、イベントハンドラonPlayListEndなどの中で、明示的にムービープレーヤ３００に対して停止（メソッドstop()発行）を指示する必要がある。

【０２７４】

なお、通常速度よりも高速で再生を行う高速再生時では、プレイリストの終点に到達した際に、プレイリストの最後のピクチャが飛び込み点に該当していなくても、プレイリストの最後のピクチャを表示する。

【０２７５】

一方、ムービープレーヤ３００は、プレイリストを逆方向で再生中にプレイアイテムの先頭に到達した場合、前のプレイアイテム、すなわち、順方向に再生した場合に時間的に前に再生されるようになっているプレイアイテムの再生を行う。この、前のプレイアイテムの再生も、最後から先頭に向かっての逆方向再生を継続する。再生速度も維持される。また、逆方向再生中にプレイリストの先頭に到達した時には、変速再生は解除され、ムービープレーヤ３００は、プレイリストの先頭で一時停止となる。

【０２７６】

さらに、ムービープレーヤ３００は、一時停止を指示する制御コマンド３１１によっても、状態が一時停止に変化する。制御コマンド３１１で一時停止を解除したときのプレイリストの再生方向、再生速度は、ＵＭＤビデオプレーヤの実装依存である。

【０２７７】

次に、プレイリストの再生中に発生するイベントについて説明する。プレイリスト再生中に発生するイベントには、図１３を用いて既に説明したように、ユーザ操作に応じたイベントangleChange、イベントaudioChangeおよびイベントsubtitleChange、ならびに、プレイリスト中に埋め込まれたマークに応じたイベントchapterおよびイベントeventマークがある。また、イベント発生時の動作の詳細な例は、図１５を用いて既に説明されている。

【０２７８】

ここでは、プレイリストの最後での処理について説明する。既に説明したように、ムービープレーヤ３００は、メソッドplay()によって指定された番号のプレイリストを再生する。一旦プレイリストの再生が開始されると、スクリプトプログラムや制御コマンド３１１による制御無しに、当該プレイリストの最後まで、再生が継続される。再生がプレイリストの最後に到達すると、ムービープレーヤ３００は、イベントplayListEndをスクリプトプログラムに通知する。プレイリストの最後に到達するまでの方法については、問わない。すなわち、プレイリストの再生が通常再生、早送り再生、他プレイリストからの飛び込みによる再生などの如何によらず、プレイリストの最後に到達した時点で、ムービープレーヤ３００は、イベントplayListEndを発生する。

【０２７９】

再生がプレイリストの最後に到達し、イベントplayListEndが発生されると、ムービープレーヤ３００の状態が一時停止になり、ムービープレーヤ３００が内部的に有するプレイリストの再生時刻は、プレイリストの最後の時刻に一致している。なお、プレイリストの最後の時刻とは、プレイリストの最後のピクチャの再生終了時刻で、再生時間軸上で最後のプレイアイテムのＯＵＴ点と一致する。

【０２８０】

イベントplayListEndは、プレイリストを一列に再生させる場合や、マルチストーリの分岐点でメニューを表示させるためなどに利用できる。

【０２８１】

例えば、スクリプトプログラムは、イベントplayListEndが発生したときに実行するプログラムとして、イベントハンドラonPlayListEndを有していれば、そのイベントハンドラonPlayListEndを実行する。このイベントハンドラonPlayListEndに、他のプレイリストの再生を開始するメソッドplay()が記述されていれば、ムービープレーヤ３００は、他のプレイリストの再生を開始することになる。このようにして、プレイリストの再生が継続される。

【０２８２】

図３６を用いてより具体的に説明すると、プレイリスト番号「ＰＬ１」のプレイリストの再生が終了して、イベントplayListEndが発生する。このイベントplayListEndの発生を受けて、スクリプトプログラムが有するイベントハンドラonPlayListEndが実行される。このイベントハンドラonPlayListEndには、プレイリスト番号「ＰＬ２」のプレイリストの再生が指定されている。ムービープレーヤ３００は、このイベントハンドラonPlayListEndを受けて、指定されたプレイリスト番号「ＰＬ２」のプレイリストを再生する。

【０２８３】

再生パスは、プレイリスト番号「ＰＬ１」のプレイリストの最後から一旦イベントハンドラonPlayListEndに移り、さらにプレイリスト番号「ＰＬ２」のプレイリストの先頭に移ることになる。

【０２８４】

また例えば、マルチストーリの分岐点でメニューを表示させるような場合には、プレイリストの最後を分岐点とし、イベントplayListEndに対応するイベントハンドラonPlayListEndにメニュー画面を表示させるプレイリストを再生する指示を記述することが考えられる。

【０２８５】

図３７は、プレイリストの最後におけるスクリプトレイヤ３０２での処理の流れと、ム

ービープレーヤ３００の動作の一例をより詳細に示す。図３７において、ステップＳ３０～ステップＳ３３がスクリプトレイヤ３０２側の処理を示し、ステップＳ４０～ステップＳ４４がムービープレーヤ３００側の処理を示す。

【０２８６】

あるプレイリストを最後まで再生した後、次のプレイリストが再生されるためには、スクリプトプログラムによる明示的な再生指示が必要である。プレイリストの再生順序は、スクリプトプログラムで決められているため、ムービープレーヤ３００側では次に再生すべきプレイリストを自発的に決めることはできない。

【０２８７】

再生がプレイリストの最後に到達すると（ステップＳ４０）、ムービープレーヤ３００は、イベントplayListEndをスクリプトレイヤ３０２に通知する（ステップＳ４１）。ムービープレーヤ３００は、ステップＳ４０で最後まで再生されたプレイリストの最後のピクチャを表示し続けたまま、一時停止に状態遷移する（ステップＳ４２）。

【０２８８】

スクリプトレイヤ３０２は、イベントplayListEndを受けて、イベントハンドラonPlayListEndが実行される（ステップＳ３０）。ムービープレーヤ３００の次の動作は、このイベントハンドラonPlayListEnd内でのスクリプトの記述によって決まる。

【０２８９】

なお、ムービープレーヤ３００は、ステップＳ４０の後、プレイリストの最後で一時停止しているときに一時停止解除あるいは順方向再生を開始するメソッドや制御コマンド３１１を受けても、無視する。順方向再生を開始するメソッドは、引数で順方向再生を指示したメソッドplay()およびメソッドplayStep()である。また、順方向再生を開始する制御コマンド３１１としては、コマンドuo_play()、コマンドuo_playNextChapter()、コマンドuo_forwardScan()、コマンドuo_playStep()、コマンドuo_pauseOn()およびコマンドuo_pauseOff()があり、これらのコマンドは、プレイリストの最後で一時停止しているときには無視される。

【０２９０】

プレイリストの最後で一時停止しているときでも、メソッドstop()やメソッドresume()は有効である。また、モード切替も、プレイリストの最後の一時停止状態において有効である。

【０２９１】

イベントplayListEndが発生した後でも、ノーマルモードのムービープレーヤ３００は、順方向再生を開始する制御コマンド３１１以外の制御コマンド３１１を受け付ける。その場合でも、スクリプトプログラムからムービープレーヤ３００に対してメソッド３１３が実行されると、そのメソッド３１３が指示する動作に従う。

【０２９２】

図３７の例では、イベントハンドラonPlayListEndでメソッドstop()が指示される（ステップＳ３１）。ムービープレーヤ３００は、メソッドstop()の実行により、制御コマンド３１１によって引き起こされた動作は中断され、ストップ状態に遷移する（ステップＳ４３）。ストップ状態では、例えば直前まで再生していたプレイリストの最後のピクチャが消去され、黒画面となる。

【０２９３】

さらに、スクリプトレイヤ３０２では、イベントハンドラonPlayListEndにおいて、次のプレイリストを再生するためのメソッド３１３が指示される（ステップＳ３２）。例えば、メソッドplay()において、引数pauseModeとして値「x1」、引数menuModeとして値「Menu」および引数playListNumberとして次に再生するプレイリスト番号がそれぞれ指定され、ムービープレーヤ３００に対して、メニューモードにモードを切り換えると共に、引数playListNumberで指定された番号のプレイリストを通常速度再生で再生することが指示される。そして、スクリプトレイヤ３０２において、イベントハンドラonPlayListEndが終了される（ステップＳ３３）。ムービープレーヤ３００側では、ステップＳ３２で指示

されたメソッドplay()に応じてモードが切り換えられると共に、指定されたプレイリストが指定された速度で再生開始される（ステップＳ４４）。

【０２９４】

なお、コンテンツ制作者は、ユーザの操作性を高めるため、プレイリストの再生が終わった後は、ムービープレーヤ３００に対して次の動作を指定せずに、そのままにしておくべきではない。イベントハンドラonPlayListEnd内に次の動作を記述しておき、ムービープレーヤ３００の状態をストップ状態に遷移させるか、メソッドplay()で次のプレイリストの再生を指示するか、メニュー画面に戻るようにオーサリングすべきである。

【０２９５】

８－４．ムービープレーヤの再生復帰機能について

次に、ムービープレーヤ３００の状態遷移と再生復帰機能について説明する。先ず、図３８を用いて、ＵＭＤビデオプレーヤが備える３種類のメモリ領域について説明する。ＵＭＤビデオプレーヤのモデルでは、必須の３種類のメモリ領域である、プレーヤステータス領域４０１、リジュームインフォメーション領域４０２およびユーザデータ領域４０３が定義される。なお、これら３種類のメモリ領域４０１、４０２および４０３は、例えばメモリ１１３上に形成される。ＣＰＵ１１２のワークメモリとしてのＲＡＭ上に形成してもよい。

【０２９６】

プレーヤステータス領域４０１は、ムービープレーヤ３００の再生状態を表す情報を保持するメモリ領域である。すなわち、プレーヤステータス領域４０１には、図３におけるプレーヤステータス３２３Ｂが保持される。プレーヤステータス領域４０１の内容は、スクリプトプログラム４００から、メソッドgetPlayerStatus()で読み出すことができる。

【０２９７】

リジュームインフォメーション領域４０２は、プレーヤステータス領域４０１に保持される情報の一部を一時的に退避（バックアップ）させるためのメモリ領域である。すなわち、プレーヤステータス領域４０１の一部の情報は、図３におけるリジュームインフォメーション３２４として、リジュームインフォメーション領域４０２に保持される。リジュームインフォメーション領域４０２に退避されれたプレーヤステータス領域４０１の一部の情報は、必要に応じてプレーヤステータス領域４０１に復帰される（リストア）。これらバックアップおよびリストアは、ネイティブ実装プラットフォーム３０１により行われる。リジュームインフォメーション領域４０２に保持された情報は、以前の再生停止箇所から再生を開始する、リジューム(resume)再生機能で使用される。

【０２９８】

リジュームインフォメーション領域４０２の内容は、スクリプトプログラム４００からメソッドgetResumeInfo()で読み出すことができる。リジュームインフォメーション領域４０２に保持されるリジュームインフォメーション３２４のうち、ストリームに関係するパラメータは、スクリプトプログラム４００からメソッドchangeResumeInfo()で値を変更することが可能である。

【０２９９】

リジュームインフォメーション領域４０２に保持された情報は、ネイティブ実装プラットフォーム３０１により必要に応じて不揮発性メモリ４１０に書き込まれる(save)。同様に、リジュームインフォメーション領域４０２から不揮発性メモリ４１０に書き込まれた情報は、ネイティブ実装プラットフォーム３０１により必要に応じて不揮発性メモリ４１０から読み出されて(load)、リジュームインフォメーション領域４０２に記憶される。

【０３００】

なお、プレーヤステータス領域４０１からリジュームインフォメーション領域４０２へのバックアップと、リジュームインフォメーション領域４０２からプレーヤステータス領域４０１へのリストアは、メソッド実行によってムービープレーヤ３００が特定の状態遷移を行うことに伴って発生する処理であり、ムービープレーヤ３００が自動的に行う。

【０３０１】

ユーザデータ領域４０３は、コンテンツ依存の情報を保持する領域であって、コンテンツ制作者が任意に使用できる。ムービープレーヤ３００によるプレイリストの再生経路の履歴、クイズの正解／不正解など、コンテンツによって使い方は任意である。

【０３０２】

ユーザデータ領域４０３に対するデータの書き込みは、スクリプトプログラム４００からメソッドsetUserData()により行うことができる。ユーザデータ領域４０３の内容は、スクリプトプログラム４００からメソッドgetUserData()で読み出すことができる。ユーザデータ領域４０３に保持された情報は、ネイティブ実装プラットフォーム３０１により必要に応じて不揮発性メモリ４１０に書き込まれる(save)。同様に、ユーザデータ領域４０３から不揮発性メモリ４１０に書き込まれた情報は、ネイティブ実装プラットフォーム３０１により必要に応じて不揮発性メモリ４１０から読み出されて(load)、ユーザデータ領域４０３に記憶される。

【０３０３】

再生復帰機能を実現するために、この発明の実施の一形態において構築したＵＭＤビデオプレーヤのモデルについて説明する。

【０３０４】

先ず、リジューム動作について、概略的に説明する。リジュームインフォメーション領域４０２にバックアップされた情報を用いて再生状態を復帰させる動作を、リジューム(resume)と呼ぶ。リジュームは、メソッドresume()によって行われる。

【０３０５】

より具体的には、プレーヤステータス領域４０１内のプレーヤステータス３２３Ｂをリジュームインフォメーション領域４０２にバックアップしてリジュームインフォメーション３２４とし、メソッドresume()に応じて、リジュームインフォメーション領域４０２にバックアップされたリジュームインフォメーション３２４を用いて、再生状態を復帰させる。プレーヤステータス３２３Ｂは、ムービープレーヤ３００の状態、すなわちムービープレーヤ３００が現在再生しているプレイリストの番号、チャプタ番号、選択されたストリーム番号などからなる。

【０３０６】

ムービープレーヤ３００に対し、メソッドresume()を発行した時のムービープレーヤ３００の動作は、リジュームインフォメーション領域４０２内にリジュームインフォメーション３２４が存在するか否かにより異なる。リジュームインフォメーション領域４０２内にリジュームインフォメーション３２４が存在するときは、当該リジュームインフォメーション３２４が、プレーヤステータス領域４０１にプレーヤステータス３２３Ｂとしてリストアされる。このとき、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４は、破棄される。

【０３０７】

コンテンツ再生中に呼び出したメニューにおいて再生ストリームを変更する場合は、メソッドchangeResumeInfo()を用いる。メソッドchangeResumeInfo()でリジュームインフォメーション領域４０２に保持されているリジュームインフォメーション３２４を所定に変更した後に、メソッドresume()でリジュームを行えば、再生ストリームを変更して再生を始めることができる。

【０３０８】

メソッドresume()を実行することで、ムービープレーヤ３００にリジュームを行わせることができるが、メソッドgetResumeInfo()でリジュームインフォメーション３２４を取得した上で、引数を指定したメソッドplay()を実行することによっても、リジュームを実現することが可能である。

【０３０９】

プレーヤステータス３２３Ｂのリジュームインフォメーション領域４０２へのバックアップについて、図３９および図４０を用いて説明する。図３９は、ムービープレーヤ３００に定義される４状態間の状態遷移のうち、プレーヤステータス領域４０１に保持されて

いるプレーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされる状態遷移を示す。図４０は、プレーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされるときの条件を示す。

【０３１０】

プレイリストを再生している、ノーマルモードでプレイ状態（状態{Normal,play}）のムービープレーヤ３００がストップ状態に遷移すると、プレーヤステータス領域４０１に保持されているプレーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされ、リジュームインフォメーション３２４として保持される。なお、ストップ状態では、プレーヤステータス３２３Ｂの一部の値は、不定になる。

【０３１１】

また、ムービープレーヤ３００が状態{Normal,play}から状態{Menu,play}に遷移するときにも、プレーヤステータス領域４０１に保持されているプレーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされる。

【０３１２】

一方、メニューモードでプレイリストを再生しているムービープレーヤ３００が状態遷移をしても、プレーヤステータス領域４０１に保持されているプレーヤステータス３２３Ｂは、リジュームインフォメーション領域４０１にバックアップされない。

【０３１３】

すなわち、プレーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされリジュームインフォメーション３２４とされるのは、次の場合である。

（１）ムービープレーヤ３００の現在の状態が状態{Normal,Play}であり、メソッドplay()の実行により状態{Menu,Play}に直接的に状態遷移する場合。

（２）ムービープレーヤ３００の状態が状態{Normal,Play}であり、メソッドstop()の実行により、状態{Normal,Stop}あるいは状態{Menu,Stop}に状態遷移する場合。このとき、メソッドstop()の引数resumeInfoClearFlagは、値「false」である。

【０３１４】

プレーヤステータス３２３Ｂのリジュームインフォメーション領域４０２への保存（バックアップ）は、本編中の戻り位置を保存しておくために使用されることを想定している。例えば、本編再生中に動画メニューに飛び、再び本編に戻って再生といった一連の動作を実現する際に、リジュームインフォメーション領域４０２にプレーヤステータス３２３Ｂがバックアップされたデータであるリジュームインフォメーション３２４が用いられることが想定される。

【０３１５】

このため本編再生中、すなわちムービープレーヤ３００が状態{Normal, Play}の状態では、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４は、破棄されている。ムービープレーヤ３００が状態{Normal, Play}の状態からその他の状態に遷移したときに、プレーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされ、リジュームインフォメーション３２４とされる。

【０３１６】

このように、リジューム再生を実現するため、ムービープレーヤ３００の状態遷移に応じて、プレーヤステータス３２３Ｂのリジュームインフォメーション領域４０２へのバックアップと、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４の破棄が適宜、行われる。また、スクリプトレイヤ３０２でメソッドresume()が指示されたときに、リジュームインフォメーション領域４０２内にリジュームインフォメーション３２４が存在すれば、リジュームインフォメーション３２４は、プレーヤステータス領域４０１にリストアされ、プレーヤステータス３２３Ｂとされる。

【０３１７】

スクリプトレイヤ３０２から、メソッドgetResumeInfo()を用いて、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４を読み出すことができる。リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４

における、ストリームに関係するパラメータは、メソッドchangeResumeInfo()によって変更することができる。さらに、メソッドstop()の引数で指定することにより、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４を破棄することができる。

【０３１８】

リジュームインフォメーション領域４０２に保持されたリジュームインフォメーション３２４のプレーヤステータス領域４０１へのリストアと、破棄について、図４１～図４４を用いて説明する。本編中の戻り位置として保存したリジュームインフォメーション３２４は、ムービープレーヤ３００が本編再生状態、すなわち状態{Normal,Play}の状態に戻った後に破棄される。このとき、リジュームインフォメーション３２４は、プレーヤステータス領域４０１にプレーヤステータス３２３Ｂとしてリストアされた後に破棄される場合と、リストアされずに破棄される場合の２つの場合がある。

【０３１９】

すなわち、このモデルでは、ムービープレーヤ３００が状態{Normal,Play}の状態に戻ると、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４が破棄され、その際に、ムービープレーヤ３００などが所定の条件を満たしている場合は、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４がプレーヤステータス領域４０１にリストアされた後に破棄されるという特徴を有する。リジュームインフォメーション３２４がプレーヤステータス領域４０１にリストアされる場合には、リジュームインフォメーション３２４で指定された箇所からの再生開始となり、これがリジューム再生に相当する。

【０３２０】

図４１は、ムービープレーヤ３００に定義される４状態間の状態遷移のうち、リジュームインフォメーション３２４がプレーヤステータス領域４０１にリストアされ、その後に破棄される状態遷移を示す。

【０３２１】

以下の（１）～（３）に記す３条件が揃った場合に、リジュームインフォメーション３２４がリストアされた後に破棄される。

（１）ムービープレーヤ３００の現在の状態が状態{Menu,Stop}、状態{Normal,Stop}または状態{Menu,Play}である。

（２）リジュームインフォメーション３２４がリジュームインフォメーション領域４０２内に存在する。

（３）メソッドresume()の実行により状態{Normal,Play}に遷移する。

【０３２２】

図４２は、これらの条件をまとめて示す。なお、ムービープレーヤ３００の状態が状態{Normal,Play}の場合は、リジュームインフォメーション３２４が存在しないため、図４２中では定義されていない。

【０３２３】

なお、リジュームインフォメーション領域４０２内にリジュームインフォメーション３２４が存在するときにメソッドresume()が実行されると、ムービープレーヤ３００の状態は、状態{Normal,Play}に遷移する。また、リジュームインフォメーション領域４０２内にリジュームインフォメーション３２４が存在しない場合のメソッドresume()は、ムービープレーヤ３００の状態を変更しない。このとき、メソッドresume()実行直前の状態{Mode,State}が維持され、プレーヤステータス３２３Ｂも変更されない。

【０３２４】

一方、以下の（４）～（６）に記す３条件が揃った場合に、リジュームインフォメーション３２４がリストアされずに破棄される。

（４）ムービープレーヤ３００の現在の状態が、状態{Menu,Stop}、状態{Normal,Stop}または状態{Menu,Play}である

（５）リジュームインフォメーション３２４がリジュームインフォメーション領域４０２

内に存在する。
（６）Play()メソッドの実行により、状態{Normal,Play}に遷移する。
【０３２５】
図４３は、これらの条件をまとめて示す。なお、ムービープレーヤ３００の状態が状態{Normal,Play}の場合は、リジュームインフォメーション３２４がリジュームインフォメーション領域４０２内に存在しないため、図４３中では定義されていない。
【０３２６】
なお、リジュームインフォメーション３２４が存在しないときに、メソッドplay()の実行によりムービープレーヤ３００の状態が状態{Normal,Play}に遷移した場合、結果として、リジュームインフォメーション３２４が存在しない状況が保持される。
【０３２７】
リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４の破棄は、メソッドstop()の引数の設定によっても発生させることができる。具体的には、この発明の実施の一形態では、メソッドstop()の引数として、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４の破棄を行うか否かを指定する引数resumeInfoClearFlagを定義した。図４４に示されるように、メソッドstop()の実行時に、引数resumeInfoClearFlagに対して値「True」が指定されたときに、リジュームインフォメーション３２４の破棄が行われる。
【０３２８】
例えば、映画本編を最後まで再生してムービープレーヤ３００を停止させた場合は、映画本編の最後の位置がリジュームインフォメーション３２４として記録されてしまう。その後にユーザが再生（続き再生）をすると、映画本編の最後にジャンプして一時停止した状態になってしまい、使い勝手が悪くなる。
【０３２９】
これを改善するには、モデルの定義上、自動的に記録されてしまうリジュームインフォメーション３２４を破棄する手段が必要になる。映画本編の最後がどこであるかは、コンテンツ制作者しか分からないため、スクリプトプログラム４００からムービープレーヤ３００に対するメソッドstop()の引数resumeInfoClearFlagによって、リジュームインフォメーション３２４の破棄を指定することが出来るようにした。
【０３３０】
図４５は、メソッドstop()の引数resumeInfoClearFlagを用いたＵＭＤビデオプレーヤの一例の動作を示す。図４５において、ステップＳ５０～ステップＳ５４がスクリプトレイヤ３０２側の処理を示し、ステップＳ６０～ステップＳ６４がムービープレーヤ３００側の処理を示す。
【０３３１】
再生がプレイリストの最後に到達すると（ステップＳ６０）、ムービープレーヤ３００は、イベントplayListEndをスクリプトレイヤ３０２に通知する（ステップＳ６１）。ムービープレーヤ３００は、ステップＳ６０で最後まで再生されたプレイリストの最後のピクチャを表示し続けたまま、一時停止に状態遷移する（ステップＳ６２）。
【０３３２】
スクリプトレイヤ３０２は、イベントplayListEndを受けて、イベントハンドラonPlayListEndが実行される（ステップＳ５０）。次のステップＳ５１では、イベントplayListEndが通知されたプレイリストがオーサシナリオの最後であるか否かが判断される。あるプレイリストがシナリオの最後のプレイリストであるか否かは、例えばスクリプトプログラム４００に基づき判断することができる。
【０３３３】
若し、最後ではないと判断されれば、処理はステップＳ５３に移行し、メソッドstop()の引数resumeInfoClearFlagを値「false」とし、リジュームインフォメーション３２４を破棄しないメソッドstop()をムービープレーヤ３００に対して発行する。ムービープレーヤ３００は、このメソッドstop()を受けて、状態がストップ状態に遷移されると共に、プ

レーヤステータス３２３Ｂがリジュームインフォメーション領域４０２にバックアップされる（ステップＳ６４）。

【０３３４】

一方、スクリプトレイヤ３０２において、ステップＳ５１でシナリオの最後であると判断されれば、処理はステップＳ５２に移行し、メソッドstop()の引数resumeInfoClearFlagを値「True」とし、リジュームインフォメーション３２４を破棄するメソッドstop()をムービープレーヤ３００に対して通知する。ムービープレーヤ３００は、このメソッドstop()を受けて、状態がストップ状態に遷移されると共に、リジュームインフォメーション領域４０２内のリジュームインフォメーション３２４が破棄（クリア）される（ステップＳ６３）。

【０３３５】

なお、スクリプトレイヤ３０２において、ステップＳ５２の後、スクリプトプログラム４００の記述によっては、メソッドend()が実行される（ステップＳ５４）。

【０３３６】

８－５．各データのライフサイクルについて

次に、プレーヤステータス３２３Ｂ、リジュームインフォメーション３２４およびユーザデータのライフサイクルについて説明する。

【０３３７】

図４６は、プレーヤステータス３２３Ｂの一例のライフサイクルを示す。ムービープレーヤ３００の生成時に、プレーヤステータス３２３Ｂも生成される。ムービープレーヤ３００が消滅すると、プレーヤステータス３２３Ｂも消滅する。プレーヤステータス３２３Ｂは、生成時に初期化される。生成時は、ムービープレーヤ３００の状態を表すプロパティは、停止状態を表し、それ以外のプロパティは、不定となる。プレーヤステータス３２３Ｂの値は、ムービープレーヤ３００における再生状態の変化に伴い変化する。また、プレーヤステータス３２３Ｂの値は、リジュームインフォメーション領域４０２の内容がリストアされたときに、変化する。また、プレーヤステータス３２３Ｂは、スクリプトレイヤ３０２からのメソッドgetPlayerStatus()により読み出すことができる。

【０３３８】

なお、プレーヤステータス３２３Ｂをどのような形で記憶するかは、ムービープレーヤ３００の実装に依存する。スクリプトからメソッドgetPlayerStatus()によって情報を得ることができるようになってさえいれば、どのような形で情報を保持していてもよい。

【０３３９】

図４７は、リジュームインフォメーション３２４の一例のライフサイクルを示す。ムービープレーヤ３００生成時にリジュームインフォメーションのメモリ領域４０２が確保され、リジュームインフォメーション３２４の生成と共に初期化が行われる。初期化されると、リジュームインフォメーション３２４の内容は、破棄される。不揮発性メモリを実装しているＵＭＤビデオプレーヤは、ムービープレーヤ３００を初期化する際に、リジュームインフォメーション３２４を不揮発性メモリからロード(load)する。このとき、ユーザデータのロードも同時に行われる。

【０３４０】

ムービープレーヤ３００の状態が状態{Normal,Play}からその他の状態に遷移したとき、プレーヤステータス３２３Ｂは、リジュームインフォメーション領域４０２にバックアップされる。

【０３４１】

リジュームインフォメーション３２４におけるストリームに関係するパラメータvideoNumber、audioNumber、audioFlag、subtitleNumberおよびsubtitleFlagは、スクリプトレイヤ３０２からのメソッドchangeResumeInfo()によって変更することができる。

【０３４２】

ムービープレーヤ３００において、ノーマルモードでのプレイリスト再生が開始されたとき、リジュームインフォメーション３２４の内容は、破棄される。このとき、破棄に先

立ってリジュームインフォメーション３２４のプレーヤステータス３２３Ｂへのリストアが行われる場合と行われない場合とがある。また、引数resumeInfoClearFlagの値が「True」とされたstop()メソッドが実行されたときに、リジュームインフォメーション３２４の内容は破棄される。

【０３４３】

リジュームインフォメーション３２４が存在するときに、メソッドresume()が実行されると、リジュームインフォメーション３２４のプレーヤステータス３２３Ｂへのリストアが行われる。

【０３４４】

スクリプトレイヤ３０２から、メソッドgetResumeInfo()により、リジュームインフォメーション３２４の値を読み出すことができる。破棄された状態のリジュームインフォメーション３２４を読み出すと、返値playStatusとして値「０」が戻るため、リジュームインフォメーション３２４の有無を区別することが出来る。

【０３４５】

ムービープレーヤ３００が終了（消滅）するときに、リジュームインフォメーション３２４も消滅する。不揮発性メモリを実装しているＵＭＤビデオプレーヤは、ムービープレーヤ３００が終了（消滅）する際に、リジュームインフォメーション３２４を不揮発性メモリにセーブする。そのとき、ユーザデータのセーブも同時に行われる。

【０３４６】

図４８は、ユーザデータの一例のライフサイクルを示す。ムービープレーヤ３００の生成時にメモリ領域が確保され、ユーザデータが生成される。生成と同時に初期化が行われる。初期化されると、ユーザデータの内容はクリアされる(メソッドgetUserData()で、長さ「０」の配列が返る)。不揮発性メモリを実装しているＵＭＤビデオプレーヤは、ムービープレーヤ３００の初期化の際に、ユーザデータを不揮発性メモリからロードする。このとき、リジュームインフォメーションのロードも同時に行われる。

【０３４７】

メソッドsetUserData()が実行されたときに、ユーザデータ領域４０３にユーザデータが書き込まれる。メソッドsetUserData()により、最大でデータ長が６４ビットのＩｎｔ型の配列がユーザデータ領域４０２に保持される。ユーザデータ領域４０３のデータは、スクリプトレイヤ３０２からのメソッドgetUserData()で読み出すことができる。ユーザデータがセットされていないときは、長さが０の配列が返る。

【０３４８】

スクリプトレイヤ３０２からユーザデータ領域４０３の内容をクリアするメソッドは、無い。ユーザデータ領域４０３に対する上書きにより、内容を書き換えることができる。

【０３４９】

ムービープレーヤ３００が終了（消滅）するときに、ユーザデータ領域４０３も消滅する。不揮発性メモリを実装しているＵＭＤビデオプレーヤは、ムービープレーヤ３００が終了（消滅）する際に、ユーザデータ領域４０３の保持されているデータを不揮発性メモリにセーブする。このとき、リジュームインフォメーション３２４のセーブも同時に行われる。

【０３５０】

なお、上述では、この発明がオーディオストリームおよびビデオストリームを共に処理するようなディスク再生装置１００に適用されるように説明したが、これはこの例に限定されない。例えば、オーディオストリームのみ、ビデオストリームのみを再生する場合にも、この発明を適用することができる。

【０３５１】

また、上述では、コンテンツデータが記録される記録媒体として、ディスク上記録媒体を用いるように説明したが、これはこの例に限定されない。例えば、コンテンツデータが記録される記録媒体として、半導体メモリを用いることができる。

【０３５２】

さらに、上述では、この発明が適用されるディスク再生装置１００が専用的なハードウェアで構成されるように説明したが、これはこの例に限定されない。すなわち、ディスク再生装置１００のディスクドライブ以外の構成は、コンピュータ装置上で動作するソフトウェアによっても実現することができる。この場合、ディスク再生装置１００を実現するソフトウェアは、ＣＤ－ＲＯＭ(Compact Disc-Read Only Memory)やＤＶＤ－ＲＯＭ(Digital Versatile Disc-ROM)といった記録媒体に記録されて供給することができる。ディスク再生装置１００を実現するためのソフトウェアが記録された記録媒体を、コンピュータ装置のディスクドライブに装填し、記録媒体に記録された当該ソフトウェアをコンピュータ装置にインストールする。コンピュータ装置に対して、ＵＭＤに対応したディスクドライブ装置を接続することで、この発明によるディスク再生装置１００と同等の構成が実現できる。ＵＭＤビデオのコンテンツを記録した記録媒体に、当該ソフトウェアを共に記録して提供することも考えられる。

【図面の簡単な説明】

【０３５３】

【図１】ＵＭＤビデオ規格のレイヤ構成を示す略線図である。

【図２】この発明の実施の一形態による一例のプレーヤモデルを模式的に示す略線図である。

【図３】ムービープレーヤの一例の内部構成を示す略線図である。

【図４】ムービープレーヤのプレイ状態およびストップ状態を説明するための図である。

【図５】この発明の実施の一形態によるムービープレーヤのイベントモデルを模式的に示す模式図である。

【図６】プレイリストの再生中に発生する一例のイベントを示す略線図である。

【図７】ムービープレーヤオブジェクトが有する一例のプロパティを一覧して示す略線図である。

【図８】ムービープレーヤオブジェクトが有する一例のメソッドを一覧して示す略線図である。

【図９】ユーザ入力による一例のキー入力を示す略線図である。

【図１０】ユーザ入力による一例のキー入力を示す略線図である。

【図１１】キー入力に応じた一例の制御コマンドを示す略線図である。

【図１２】キー入力に対応する一例のイベントを示す略線図である。

【図１３】一例のイベントハンドラを示す略線図である。

【図１４】一例のイベントハンドラを示す略線図である。

【図１５】ユーザ入力イベントをきっかけとして、用意されたプログラムが実行される一例の処理を示すフローチャートである。

【図１６】スクリプトプログラムの例について説明するための図である。

【図１７】一例のスクリプトプログラムを示す略線図である。

【図１８】ＵＭＤビデオ規格に適用されるファイルの一例の管理構造を示す略線図である。

【図１９】ファイル"PLAYLIST.DAT"の全体構造を表す一例のシンタクスを示す略線図である。

【図２０】ブロックPlayItem()の一例の内部構造を示す略線図である。

【図２１】ブロックPlayListMark()の一例の内部構造を示す略線図である。

【図２２】ブロックMark()内のフィールドmark_typeについて説明するための図である。

【図２３】クリップＡＶストリームファイル内でのマーク時刻の指定について説明するための図である。

【図２４】クリップＡＶストリームファイル"XXXXX.CLP"の全体構造を表す一例のシンタクスを示す略線図である。

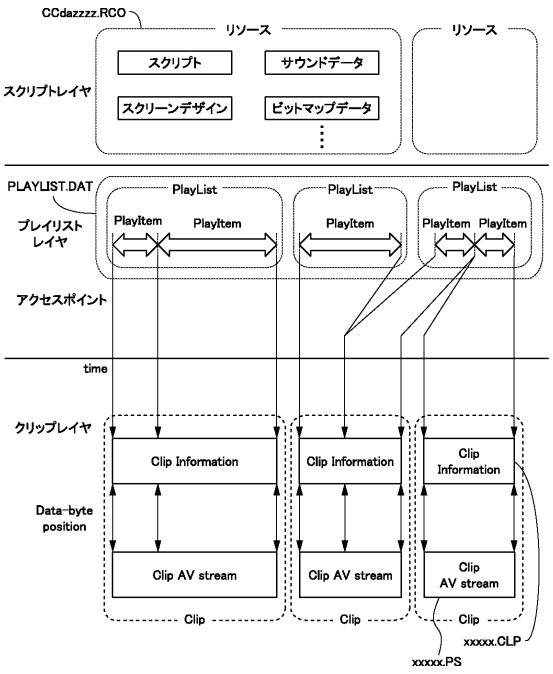
【図２５】ブロックStreamInfo()のエレメンタリストリームに対する関連付けを説明するための図である。

【図２６】ブロックStaticInfo()の一例の内部構造を示す略線図である。

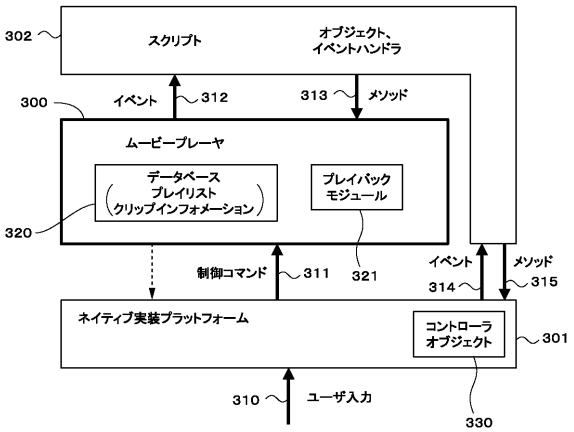
【図２７】ブロックDynamicInfo()の一例の内部構造を示す略線図である。
【図２８】ブロックEP_map()の一例の内部構造を示す略線図である。
【図２９】この発明を適用可能なディスク再生装置の一例の構成を概略的に示すブロック図である。
【図３０】ディスク再生装置における動作をより詳細に説明するための機能ブロック図である。
【図３１】この発明によるムービープレーヤの状態の定義を概念的に示す略線図である。
【図３２】現在の状態と、メソッドによって状態遷移した後の状態を、ムービープレーヤの４状態のそれぞれについて組み合わせて示す略線図である。
【図３３】メソッドplay()を実行した際のムービープレーヤの状態遷移の例について説明する略線図である。
【図３４】プレイアイテムの再生方法を説明するための略線図である。
【図３５】プレイリストの再生中にプレイリストの始点、終点に到達したときのムービープレーヤの一例の動作を示す略線図である。
【図３６】プレイリスト間の再生を説明するための略線図である。
【図３７】プレイリストの最後におけるスクリプトレイヤでの処理の流れと、ムービープレーヤの動作の一例をより詳細に示すフローチャートである。
【図３８】ＵＭＤビデオプレーヤが備える３種類のメモリ領域について説明するための略線図である。
【図３９】プレーヤステータスのバックアップについて説明するための略線図である。
【図４０】プレーヤステータスのバックアップについて説明するための略線図である。
【図４１】リジュームインフォメーションのリストアと、破棄について説明するための略線図である。
【図４２】リジュームインフォメーションのリストアと、破棄について説明するための略線図である。
【図４３】リジュームインフォメーションのリストアと、破棄について説明するための略線図である。
【図４４】リジュームインフォメーションのリストアと、破棄について説明するための略線図である。
【図４５】メソッドstop()の引数resumeInfoClearFlagを用いたＵＭＤビデオプレーヤの一例の動作を示す略線図である。
【図４６】プレーヤステータスの一例のライフサイクルを示す略線図である。
【図４７】リジュームインフォメーションの一例のライフサイクルを示す略線図である。
【図４８】ユーザデータの一例のライフサイクルを示す略線図である。
【符号の説明】
【０３５４】
１０１　ディスク
１１２　ＣＰＵ
１１３　メモリ
１１５　入力インターフェイス
１１６　ビデオデコーダ
１１７　オーディオデコーダ
１１８　ビデオ出力インターフェイス
１１９　オーディオ出力インターフェイス
２０１　オペレーションシステム
２１０　ビデオコンテンツ再生部
２１１　スクリプト制御モジュール
２１２　プレーヤ制御モジュール
２１４　デコード制御モジュール
２１５　バッファ制御モジュール

２１６　ビデオデコーダ制御モジュール
２１７　オーディオデコーダ制御モジュール
２１８　字幕デコーダ制御モジュール
２１９　グラフィクス制御モジュール
２４１　ビデオ出力モジュール
２４２　オーディオ出力モジュール
２５０　不揮発性メモリ制御モジュール
３００　ムービープレーヤ
３０１　ネイティブ実装プラットフォーム
３０２　スクリプトレイヤ
３１０　ユーザ入力
３１１　制御コマンド
３１２　イベント
３１３　メソッド
３２０　データベース
３２１　プレイバックモジュール
３２２　デコーダエンジン
３２３　プロパティ
３２３Ｂ　プレーヤステータス
３２４　リジュームインフォメーション
４００　スクリプトプログラム
４０１　プレーヤステータス領域
４０２　リジュームインフォメーション領域
４０３　ユーザデータ領域
４１０　不揮発性メモリ

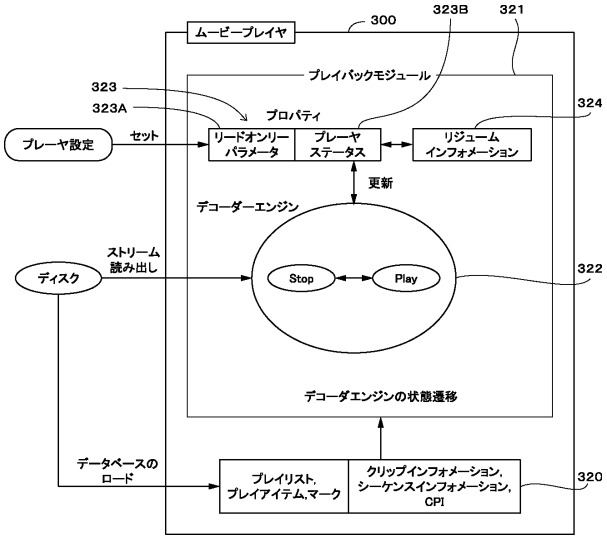
【図１】



【図２】



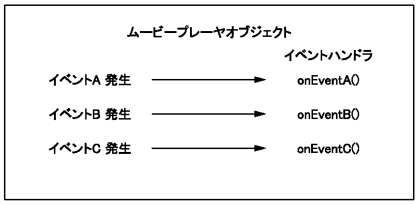
【図３】



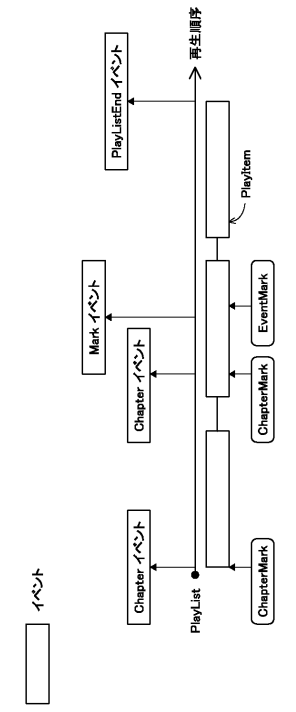
【図４】

状態	説明
play	PlayList を選択し、再生を行っている状態を指す。 Play 状態は status に細分化され、通常再生のほか、早送り・巻き戻し等の変速再生、一時停止といった各種の status を含む。
stop	PlayList を再生していない状態。 PlayList は選択されていない。

【図５】



【図６】



【図７】

A

Name	説明
scriptVersion	UMD Video Scriptのバージョン
audioChannelCapability	UMD Video Playerが再生可能なオーディオチャンネル数
languageCode	UMD Video Playerに設定されたメニュー表示言語コード
audioLanguageCode	UMD Video Playerに設定されたオーディオ言語コード
subtitleLanguageCode	UMD Video Playerに設定された字幕言語コード

リードオンリー パラメータ

B

Name	説明
playListNumber	現在再生中のプレイリスト番号
chapterNumber	現在再生中のチャプタ番号
videoNumber	現在再生中のビデオストリーム番号
audioNumber	現在再生中のオーディオストリーム番号
subtitleNumber	現在再生中の字幕ストリーム番号
playListTime	プレイリスト先頭を0とした時の時刻
audioFlag	オーディオ再生ON/OFF及びdual mono LRの指定
subtitleFlag	字幕表示ON/OFF

プレーヤ ステータス

【図８】

Name	説明
play()	再生
playChapter()	チャプタ指定再生
resume()	resumeInformation を使った再生開始
stop()	再生の停止
pause()	再生の一時停止
playStep()	コマ送り
changeStream()	ビデオ、オーディオ、字幕ストリーム変更
getPlayerStatus()	MoviePlayerの再生、停止、一時停止などの状態を取得
changeResumeInfo()	resumeInformation の内容を変更する
reset()	再生を停止し、resumeInformation をクリア
setPos()	ビデオの表示位置の設定
getPos()	ビデオの表示位置の取得
setSize()	ビデオの表示サイズの設定
getSize()	ビデオの表示サイズの取得

【図９】

キー名称	説明
VK_PLAY	再生
VK_STOP	停止
VK_PAUSE	一時停止
VK_FAST_FORWARD	早送り
VK_FAST_REVERSE	早戻し
VK_SLOW_FORWARD	スロー（順方向）
VK_SLOW_REVERSE	スロー（逆方向）
VK_STEP_FORWARD	コマ送り（順方向）
VK_STEP_REVERSE	コマ送り（逆方向）
VK_NEXT	次
VK_PREVIOUS	前
VK_ANGLE	アングル切り替え
VK_SUBTITLE	字幕切り替え
VK_AUDIO	オーディオ切り替え
VK_VIDEO_ASPECT	ビデオのアスペクト比切り替え

【図１０】

キー名称	説明
VK_UP	上
VK_DOWN	下
VK_RIGHT	右
VK_LEFT	左
VK_UP_RIGHT	右斜め上
VK_UP_LEFT	左斜め上
VK_DOWN_RIGHT	右斜め下
VK_DOWN_LEFT	左斜め下
VK_MENU	メニュー
VK_ENTER	決定
VK_RETURN	戻る
VK_COLORED_KEY_1	色つきファンクションキー1
VK_COLORED_KEY_2	色つきファンクションキー2
VK_COLORED_KEY_3	色つきファンクションキー3
VK_COLORED_KEY_4	色つきファンクションキー4
VK_COLORED_KEY_5	色つきファンクションキー5
VK_COLORED_KEY_6	色つきファンクションキー6

【図１１】

ユーザ操作に起因する制御命令	説明
uo_timeSearch(playListTime)	再生中のplayListの指定時刻から再生する。playListTimeはPlayList先頭を0としたときの時刻を表す。PlayList番号は指定できない。 そのため、現在再生中のPlayListの範囲内での時刻指定になる。
uo_play()	順方向1倍速で再生開始する。開始位置はresumeInformationによって決められる。resumeInformationが無い場合は、このユーザ操作は無効になる。playListNumberの指定のないplay()メソッドを実行したときに対応。ユーザ操作では、PlayList番号を指定できない。
uo_playChapter(chapterNumber)	再生中のplayListの指定のChapterから再生開始する。Chapterの指定がない場合には、現在再生中のChapterの先頭から再生開始する。chapterNumberの指定のないplayChapter()メソッドに対応。
uo_playPrevChapter()	前Chapterの先頭から再生開始する。
uo_playNextChapter()	次Chapterの先頭から再生開始する。
uo_jumpToEnd()	PlayListの最後にjumpする。MoviePlayerに対して、現在の再生を中止して、playListEndイベントを発生させるよう指示するユーザ操作。スクリプトでは、onPlayListEndイベントハンドラが実行される。
uo_forwardScan(speed)	speedで指定された速度で順方向再生。 speedはUMD Video Player実装依存。
uo_backwardScan(speed)	speedで指定された速度で逆方向再生。 speedはUMD Video Player実装依存。
uo_playStep(forward)	順方向コマ送り再生
uo_playStep(backward)	逆方向コマ送り再生
uo_pauseOn()	ユーザによる一時停止
uo_pauseOff()	一時停止を解除
uo_setAudioEnabled(boolean)	オーディオストリームのON,OFFを指定。 audioFlagの変更する。
uo_setSubtitleEnabled(boolean)	字幕ストリームのON,OFFを指定。 subtitleFlagの変更する。
uo_angleChange()	表示アングルを変更する。このユーザ操作がMoviePlayerに伝えられると、MoviePlayerはスクリプトにangleChangeイベントを通知する。
uo_audioChange(audioStreamNumber)	再生するオーディオを変更する。
uo_changeAudioChannel(value)	オーディオのチャンネル数切り替えおよびdual monoの時の片チャンネル切り換え。 audioFlagを変更する。
uo_subtitleChange(subtitleStreamNumber)	再生する字幕を変更する。

【図１２】

イベント	説明
menu	メニューにジャンプする。
exit	native 実装 platform がUMD Videoアプリケーションを終了させる際に native 実装 platform から発せられるイベント。
resourceChanged	リソースファイルが切り替わったときにスクリプトレイヤに通知されるイベント
up,down,left,right focusIn, focusOut, push, cancel	画面に表示されているボタンwidgetにフォーカスが当たっている間に発生するイベント
autoPlay, continuePlay	スクリプトの実行開始を指示するイベント

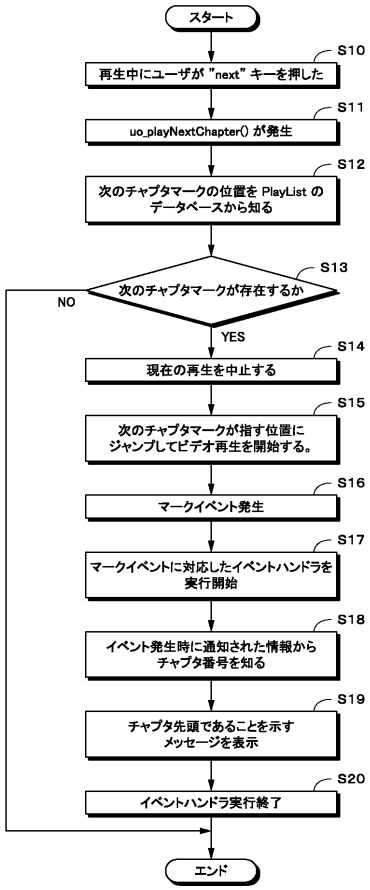
【図１３】

イベント名	対応するイベントハンドラ名	説明
mark	onMark()	Event-mark 検出時に実行される
playListEnd	onPlayListEnd()	プレイリストが終了した時に実行される
chapter	onChapter()	Chapter-mark 検出時に実行される
angleChange	onAngleChange()	ユーザ操作のアングル変更が指示された時に実行される
audioChange	onAudioChange()	ユーザ操作のオーディオ変更が指示された時に実行される
subtitleChange	onSubtitleChange()	ユーザ操作の字幕変更が指示された時に実行される

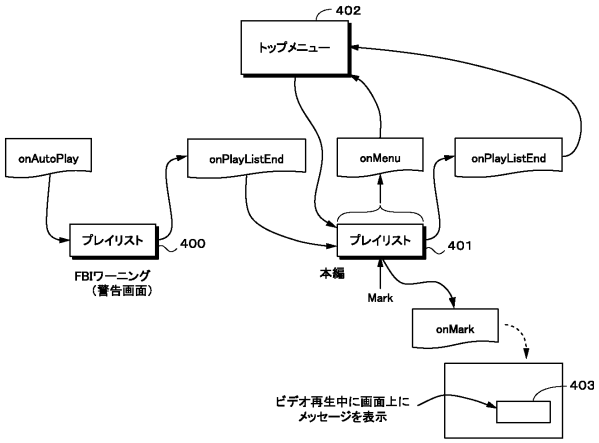
【図１４】

イベント名	対応するイベントハンドラ名	内容
menu	onMenu()	メニューにジャンプする。
exit	onExit()	ネイティブ実装プラットフォーム が UMD ビデオアプリケーション を終了させる際に ネイティブ実装プラットフォーム から発せられるイベント。
resourceChanged	onResourceChanged()	リソース切り替え後に行う処理が記述される。
autoPlay, continuePlay	onAutoPlay(), onContinuePlay()	スクリプトの実行を開始する。

【図１５】



【図１６】



【図１７】

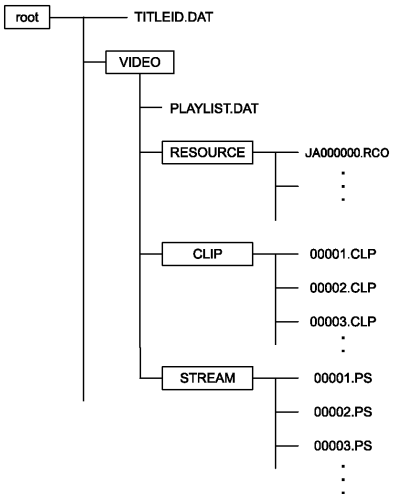
```
Controller.onAutoPlay = function(){
      //Play PlayList # 1 FBI warning.
     movieplayer.play(1);
}

movieplayer.onPlayListEnd = function(event_info){
     if(event_info.playListNumber == 1){
          // play feature film after FBI warning ends.
          movieplayer.play(2);
     }else{
          // transit to top menu after feature film ends.
          resource.pagetable["top_menu"].open();
     }
}

Controller.onMenu = function(){
       // transfer to top menu with display menu user
operation.
       resource.pagetable["top_menu"].open();
 }

movieplayer.onMark = function(event_info){
    //display dialog when event mark encountered.
    if(event_info.mark_data == 1){
         resource.pagetable["dialog_window_1"].open();
    }
     ...
 }
```

【図１８】



【図１９】

Syntax	No. of bits	Mnemonic
"PLAYLIST.DAT" {		
name_length	8	uimsbf
name_string	8*255	bslbf
number_of_PlayLists	16	uimsbf
for(i=0; i<number_of_PlayLists; i++){		
PlayList(){ // A PlayList()		
PlayList_data_length	32	uimsbf
// 属性情報		
reserved_for_word_alignment	15	bslbf
capture_enable_flag_PlayList	1	bslbf
PlayList_name_length	8	uimsbf
PlayList_name_string	8*255	bslbf
//		
number_of_PlayItems	16	uimsbf
for (i=0; i<number_of_PlayItems; i++) {		
PlayItem()		
}		
PlayListMark()		
}		
}		
}		

【図２０】

Syntax	No. of bits	Mnemonic
PlayItem() {		
length	16	uimsbf
Clip_Information_file_name_length	16	uimsbf
Clip_Information_file_name	8*Clip_Information_file_name_length	bslbf
reserved_for_word_alignment	15	bslbf
IN_time	33	uimsbf
reserved_for_word_alignment	15	bslbf
OUT_time	33	uimsbf
}		

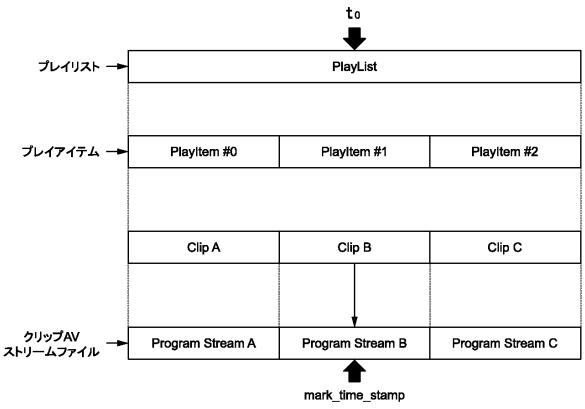
【図２１】

Syntax	No. of bits	Mnemonic
PlayListMark() {		
length	32	uimsbf
number_of_PlayList_marks	16	uimsbf
for(i=0; i < number_of_PlayList_marks; i++) {		
Mark(){		
mark_type	8	uimsbf
mark_name_length	8	uimsbf
ref_to_PlayItem_id	16	uimsbf
reserved_for_word_alignment	15	bslbf
mark_time_stamp	33	uimsbf
entry_ES_stream_id	8	uimsbf
entry_ES_private_stream_id	8	uimsbf
mark_data	32	bslbf
mark_name_string	8*24	bslbf
}		
}		
}		

【図２２】

mark_type	stream coding
0	reserved
1	Chapterマーク
2	Eventマーク
3-255	reserved

【図２３】



【図２４】

Syntax	No. of bits	Mnemonic
XXXXX.CLP{		
reserved_for_word_alignment	15	bslbf
presentation_start_time	33	uimsbf
reserved_for_word_alignment	15	bslbf
presentation_end_time	33	uimsbf
reserved_for_word_alignment	7	bslbf
capture_enable_flag_Clip	1	bslbf
number_of_streams	8	uimsbf
for (i = 0;i < number_of_streams;i++) {		
StreamInfo() {		
length	16	uimsbf
stream_id	8	uimsbf
private_stream_id	8	uimsbf
StaticInfo()		
reserved_for_word_alignment	8	bslbf
number_of_DynamicInfo	8	uimsbf
for (j = 0;j < number_of_DynamicInfo;j++) {		
reserved_for_word_alignment	15	bslbf
pts_change_point	33	uimsbf
DynamicInfo()		
}		
} //end of oneStreamInfo		
}		
EP_map()		
}		

【図２５】

エレメンタリストリームの種類	stream_id	private_stream_id
ビデオ	0xE0～0xEF	（なし）
ATRACオーディオ	0xBD	0x00～0x0F
LPCMオーディオ	0xBD	0x10～0x1F
字幕	0xBD	0x80～0x9F

【図２６】

Syntax	No. of bits	Mnemonic
StaticInfo() {		
if (stream == VIDEO) {		
reserved_for_word_alignment	16	bslbf
picture_size	4	uimsbf
frame_rate	4	uimsbf
reserved_for_word_alignment	7	bslbf
cc_flag	1	bslbf
} else if (stream == AUDIO) {		
audio_language_code	16	bslbf
channel_configuration	8	uimsbf
reserved_for_word_alignment	3	bslbf
lfe_existence	1	bslbf
sampling_frequency	4	uimsbf
} else if (stream == SUBTITLE) {		
subtitle_language_code	16	bslbf
reserved_for_word_alignment	15	bslbf
configurable_flag	1	uimsbf
}		
}		

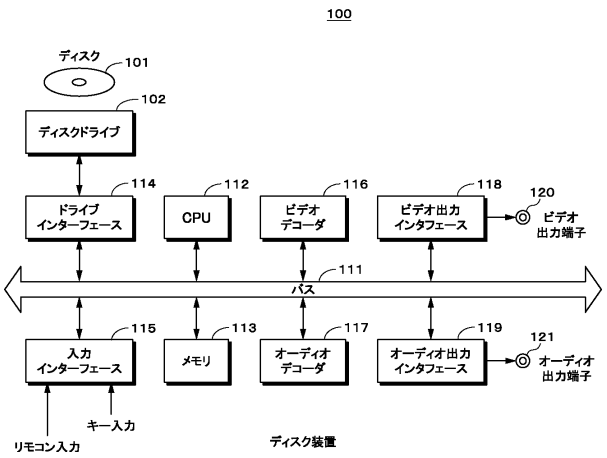
【図２７】

Syntax	No. of bits	Mnemonic
DynamicInfo(i,j) {		
reserved_for_word_alignment	8	bslbf
if (stream == VIDEO){		
reserved_for_word_alignment	4	bslbf
display_aspect_ratio	4	uimsbf
} else if (stream == AUDIO) {		
reserved_for_word_alignment	4	bslbf
channel_assignment	4	uimsbf
} else if (stream == SUBTITLE) {		
reserved_for_word_alignment	8	bslbf
}		
}		

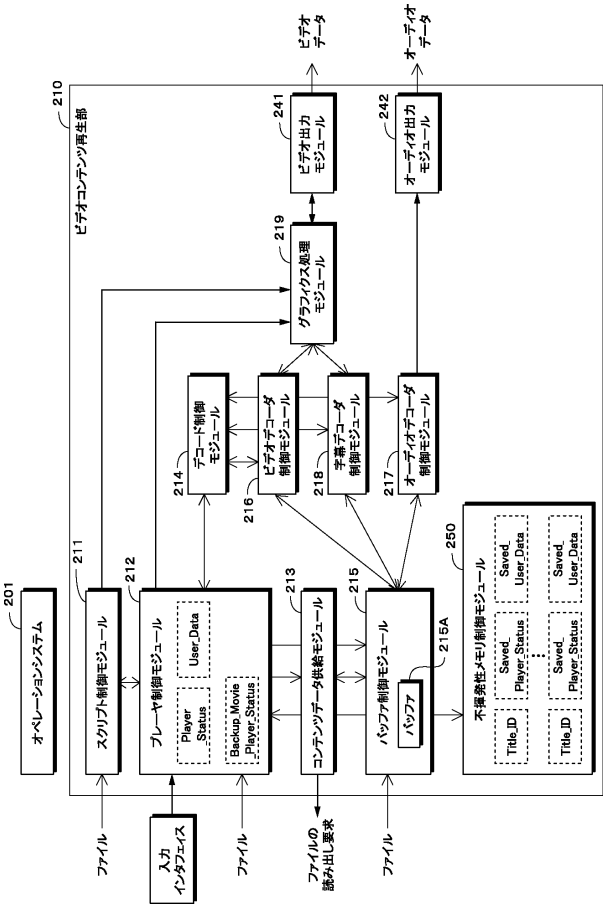
【図２８】

Syntax	No. of bits	Mnemonic
EP_map(){		
reserved_for_word_alignment	8	bslbf
number_of_stream_id_entries	8	uimsbf
for (k=0; k<number_of_stream_id_entries; k++) {		
stream_id	8	bslbf
private_stream_id	8	bslbf
number_of_EP_entries	32	uimsbf
for (i=0; i<number_of_EP_entries; i++) {		
reserved_for_word_align	15	bslbf
PTS_EP_start	33	uimsbf
RPN_EP_start	32	uimsbf
}		
}		
}		

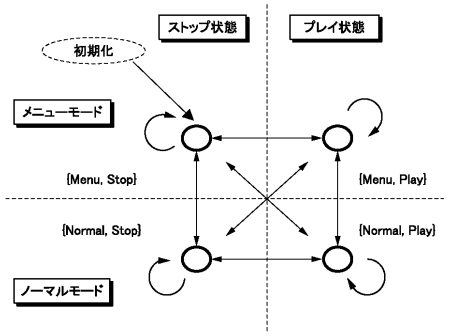
【図２９】



【図３０】



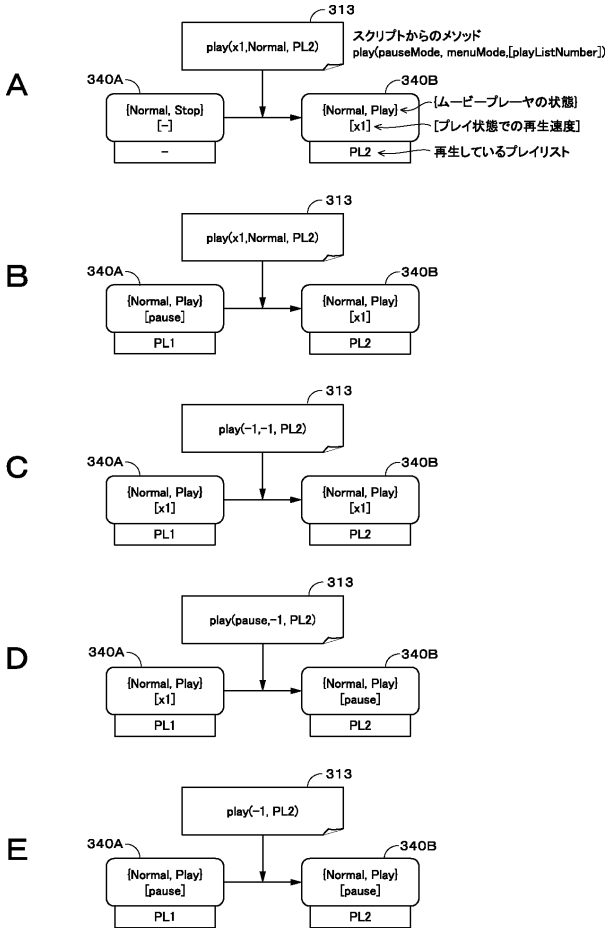
【図３１】



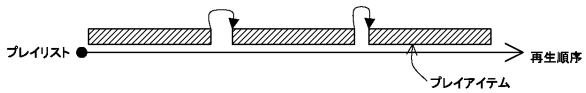
【図３２】

		メソッド実行後の状態			
		Menu	Normal	Menu	Normal
現在の状態		Stop	Stop	Play	Play
Menu	Stop	stop(), resume()	stop()	play()	play(), resume()
Normal	Stop	stop()	stop(), resume()	play()	play(), resume()
Menu	Play	stop()	stop()	play(), resume()	play(), resume()
Normal	Play	stop()	stop()	play()	play(), resume()

【図３３】



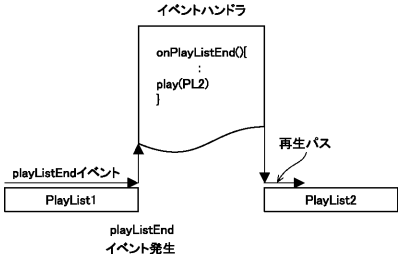
【図３４】



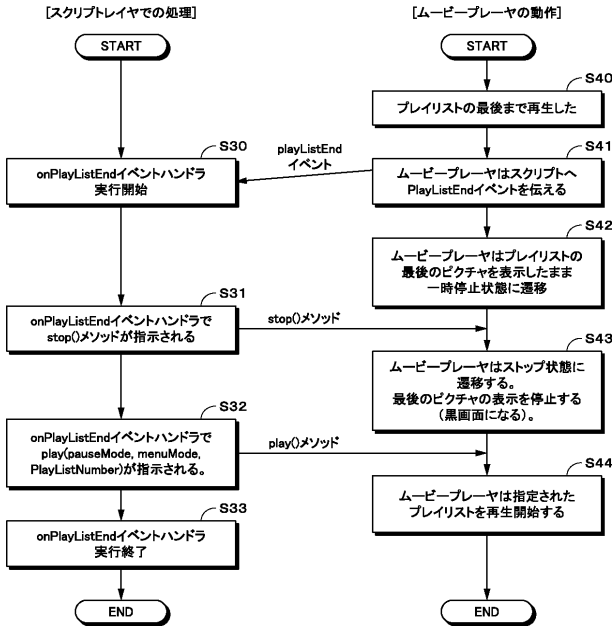
【図３５】

PlayListの再生方法	PlayListの始点、終点における動作
順方向	PlayListの最後に到達したらpauseして、PlayListの最後のピクチャの表示を継続します。 （高速再生時、PlayListの最後のピクチャが飛び込み点に該当していなくてもPlayListの最後のピクチャを表示します。）
逆方向	PlayListの先頭に到達したらPlayListの先頭でpauseします。

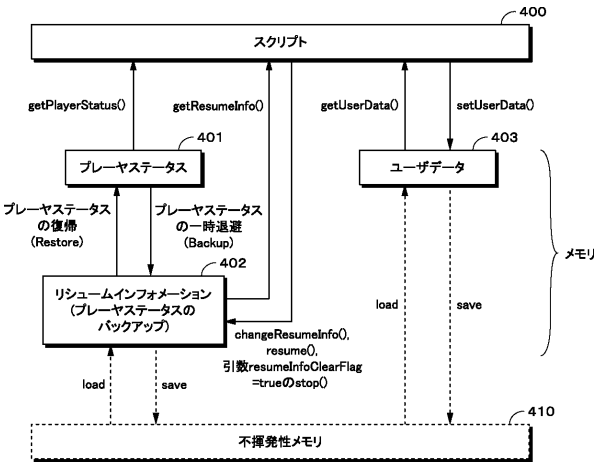
【図３６】



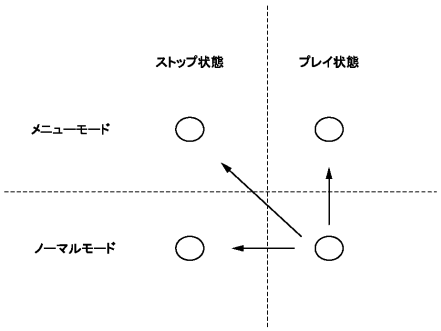
【図３７】



【図３８】



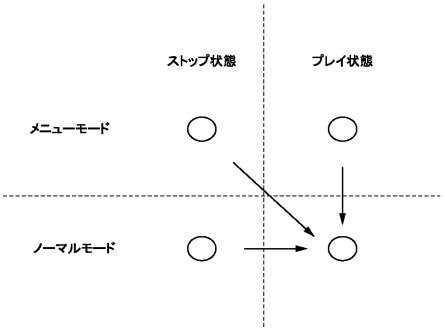
【図３９】



【図４０】

現在の状態		メソッド実行後の状態			
		Menu	Normal	Menu	Normal
		Stop	Stop	Play	Play
Menu	Stop	−	−	−	−
Normal	Stop	−	−	−	−
Menu	Play	−	−	−	−
Normal	Play	stop()による遷移、バックアップ	stop()による遷移、バックアップ	play()による遷移、バックアップ	−

【図４１】



【図４２】

現在の状態		メソッド実行後の状態			
		Menu	Normal	Menu	Normal
		Stop	Stop	Play	Play
Menu	Stop	−	−	−	resume()による遷移、リストア、破棄
Normal	Stop	−	−	−	resume()による遷移、リストア、破棄
Menu	Play	−	−	−	resume()による遷移、リストア、破棄
Normal	Play	−	−	−	(定義しない)

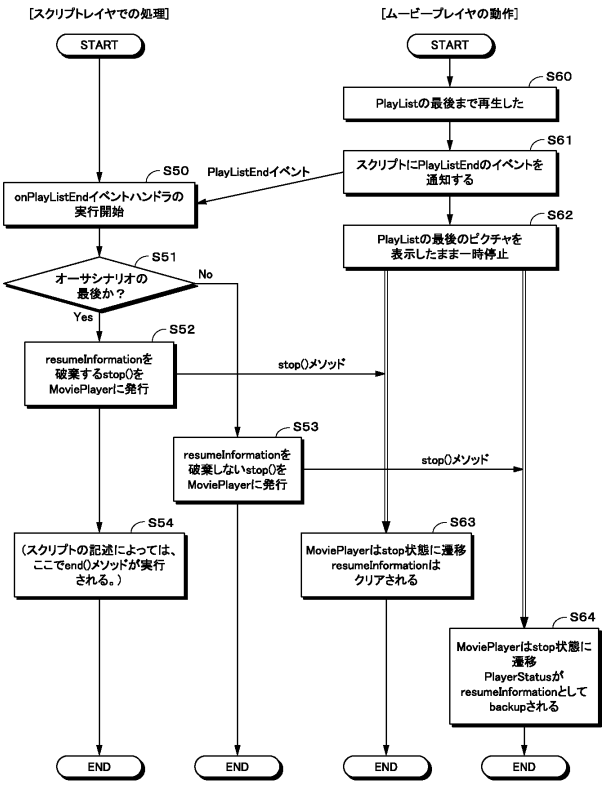
【図４３】

現在の状態		メソッド実行後の状態			
		Menu	Normal	Menu	Normal
		Stop	Stop	Play	Play
Menu	Stop	−	−	−	Playによる遷移、破棄
Normal	Stop	−	−	−	Playによる遷移、破棄
Menu	Play	−	−	−	Playによる遷移、破棄
Normal	Play	−	−	−	(定義しない)

【図４４】

現在の状態		メソッド実行後の状態			
		Menu	Normal	Menu	Normal
		Stop	Stop	Play	Play
Menu	Stop	resumeInfoClearFlag=Trueの場合に破棄	resumeInfoClearFlag=Trueの場合に破棄	−	−
Normal	Stop	resumeInfoClearFlag=Trueの場合に破棄	resumeInfoClearFlag=Trueの場合に破棄	−	−
Menu	Play	resumeInfoClearFlag=Trueの場合に破棄	resumeInfoClearFlag=Trueの場合に破棄	−	−
Normal	Play	resumeInfoClearFlag=Trueの場合に破棄	resumeInfoClearFlag=Trueの場合に破棄	−	−

【図４５】



【図４６】

PlayerStatusの状態	説明
生成	MoviePlayerの生成時に、PlayerStatusも生成される。 生成時には初期化され、MoviePlayerの状態を表すプロパティは停止状態を表し、それ以外のプロパティは不定になる。
値の変化	(1)再生状態の変化に伴いPlayerStatusの内容は変化していく。 (2) resumeInformationの内容がrestoreされたとき
メソッドによる値の読み出し	getPlayerStatus()メソッドで読み出すことができる。
消滅	MoviePlayerが終了(消滅)するとき。

【図４７】

resumeInformationの状態	説明
生成	MoviePlayerオブジェクト生成時にresumeInformationのメモリ領域が確保され、生成と同時に初期化が行われる。初期化されるとresumeInformationの内容は破棄される。 不揮発メモリを実装しているUMD Video Playerは、MoviePlayerを初期化する際に、resumeInformationを不揮発性メモリからloadする。 そのとき、userDataのloadも同時に行われる。
書き込み (PlayerStatusのbackup)	MoviePlayerが{Normal,Play}から、その他の状態に遷移したとき、PlayerStatusはresumeInformationにbackupされる。
変更	ストリームに関係するパラメータ、videoNumber,audioNumber,audioFlag,subtitleNumber,subtitleFlagは、changeResumeInfo()によって変更することが出来る。
破棄(その1)	Normal modeでのPlayList再生が開始されたとき、resumeInformationの内容は破棄される。このとき破棄に先立ってrestoreが行われる場合と行われない場合がある。
破棄(その2)	resumeInfoClearFlag=Trueのstop()メソッドが実行されたとき resumeInformationの内容は破棄される。
PlayerStatusのrestore	resumeInformationが存在するときに、resume()メソッドが実行されると、restoreが行われる。
メソッドによる値の読み出し	スクリプトからはgetResumeInfo()メソッドで値を読み出すことができる。 破棄された状態のresumeInformationを読み出すとplayStatus=0が戻るため、resumeInformationの有無を区別することが出来る。
消滅	MoviePlayerが終了(消滅)するときにresumeInformationも消滅する。 不揮発メモリを実装しているUMD Video Playerは、MoviePlayerが終了(消滅)する際に、resumeInformationを不揮発性メモリにsaveする。 そのとき、userDataのsaveも同時に行われる。

【図４８】

userDataの状態	説明
生成	MoviePlayer オブジェクト生成時にメモリ領域が確保される。生成と同時に初期化が行われる。初期化されるとuserDataの内容はクリアされる。(getUserData()で長さ0の配列が返る。) 不揮発メモリを実装しているUMD Video Playerは、MoviePlayerの初期化の際に、userDataを不揮発性メモリからloadする。 このとき、resumeInformationのloadも同時に行われる。
書き込み	setUserData()メソッドが実行されたときに書き込まれる。 setUserData()メソッドにより、最大64の長さのInt型の配列がuserDataに保持される。
読み出し	getUserData()メソッドで読み出すことができる。 UserDataがセットされていないときは、長さ0の配列が返る。
内容のクリア	スクリプトからuserDataの内容をクリアするメソッドはない。 上書きにより、内容を書き換えることが出来る。
消滅	MoviePlayerが終了(消滅)するときにuserDataも消滅する。 不揮発メモリを実装しているUMD Video Playerは、MoviePlayerが終了(消滅)する際に、userDataを不揮発性メモリにsaveする。 このとき、resumeInformationのsaveも同時に行われる。

フロントページの続き

(72)発明者　藤波　靖
東京都品川区北品川６丁目７番３５号　ソニー株式会社内
(72)発明者　各務　辰哉
東京都港区南青山２丁目６番地２１号　株式会社ソニー・コンピュータエンタテインメント内
(72)発明者　内海　秀介
東京都港区南青山２丁目６番地２１号　株式会社ソニー・コンピュータエンタテインメント内
(72)発明者　井原　宏二
東京都港区南青山２丁目６番地２１号　株式会社ソニー・コンピュータエンタテインメント内

審査官　▲吉▼澤　雅博

(56)参考文献　特開２００４－３２８４５０（ＪＰ，Ａ）
特開２００４－７５１８（ＪＰ，Ａ）
特開２００３－２４９０５７（ＪＰ，Ａ）
特開平１０－２７１４５４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ１１Ｂ　２７／１０
Ｇ１１Ｂ　２０／１０
Ｈ０４Ｎ　５／８５