(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4868535号
(P4868535)

(45)発行日　平成24年2月1日(2012.2.1)　(24)登録日　平成23年11月25日(2011.11.25)

(51)Int.Cl.		F I		
G11B 27/10	(2006.01)	G11B	27/10	A
G11B 20/10	(2006.01)	G11B	20/10	321Z
G06F 9/54	(2006.01)	G06F	9/06	640C

請求項の数 2　(全 12 頁)

(21)出願番号	特願2007-511908 (P2007-511908)
(86)(22)出願日	平成17年4月13日 (2005.4.13)
(65)公表番号	特表2007-536685 (P2007-536685A)
(43)公表日	平成19年12月13日 (2007.12.13)
(86)国際出願番号	PCT/EP2005/003867
(87)国際公開番号	WO2005/109188
(87)国際公開日	平成17年11月17日 (2005.11.17)
審査請求日	平成20年3月13日 (2008.3.13)
(31)優先権主張番号	04011063.7
(32)優先日	平成16年5月10日 (2004.5.10)
(33)優先権主張国	欧州特許庁 (EP)

(73)特許権者　501263810
トムソン　ライセンシング
Ｔｈｏｍｓｏｎ　Ｌｉｃｅｎｓｉｎｇ
フランス国，　９２１３０　イツシー　レ　ムーリノー，　ル　ジヤンヌ　ダルク，　１－５
１－５，　ｒｕｅ　Ｊｅａｎｎｅ　ｄ’Ａｒｃ，　９２１３０　ＩＳＳＹ　ＬＥＳ　ＭＯＵＬＩＮＥＡＵＸ，　Ｆｒａｎｃｅ
(74)代理人　100115864
弁理士　木越　力
(72)発明者　ホーレントル，ヨブスト
ドイツ国　３０１６３　ハノーファー　ガーベルスベルガーシユトラツセ　１８

最終頁に続く

(54)【発明の名称】ソフトウェア・アプリケーションを自動的に選択する方法

(57)【特許請求の範囲】

【請求項１】

１つまたは複数のタイトルと、タイトルの選択に使用することができる前記１つまたは複数のタイトルを列挙したインデックス・テーブルと、前記インデックス・テーブルに列挙されない１つまたは複数の更に別のタイトルとを含む読取り専用記憶媒体に記憶されたタイトルを自動的に選択する方法であって、

前記タイトルがマルチメディア・データの再生に関連する仮想マシンで実行されるプログラムであり、全てのタイトルが関連するタイトル識別子を有し、

前記方法は、

終了時に終了状態値を提供する第１のタイトルを実行するステップと、

前記終了状態値を、前記インデックス・テーブル中のエントリを表す、または前記記憶媒体に記憶されるタイトルのタイトル識別子を表す識別子に変換するステップであり、当該変換にマッピング・テーブルを使用するステップと、

前記識別子がインデックス・テーブルのエントリまたはタイトル識別子の何れを表すかを判定するステップと、

前記識別子がインデックス・テーブルのエントリを表す場合に、前記インデックス・テーブルを使用して前記識別子をタイトル識別子に変換するステップと、

前記識別子によって、または前記インデックス・テーブル・エントリに対応するタイトル識別子によって表されるタイトルを実行するステップと、

を含む、前記方法。

【請求項２】
１つまたは複数のタイトルと、タイトルの選択に使用することができる前記１つまたは複数のタイトルを列挙したインデックス・テーブルと、前記インデックス・テーブルに列挙されない１つまたは複数の更に別のタイトルとを含む読取り専用記憶媒体に記憶されたタイトルを自動的に選択する装置であり、前記タイトルがマルチメディア・データの再生に関連する仮想マシンで実行されるプログラムであり、全てのタイトルが関連するタイトル識別子を有する、タイトルを自動的に選択する装置であって、
終了する第１のタイトルから終了状態値を受信する手段と、
前記終了状態値を、前記インデックス・テーブル中のエントリを表す、または前記記憶媒体に記憶されるタイトルのタイトル識別子を表す識別子に変換する手段であり、マッピング・テーブルを使用する手段と、
前記識別子がインデックス・テーブルのエントリまたはタイトル識別子の何れを表すかを判定する手段と、
前記識別子がインデックス・テーブルのエントリを表す場合に、前記インデックス・テーブルを使用して前記識別子をタイトル識別子に変換する手段と、
前記識別子によって、または前記インデックス・テーブルのエントリに対応するタイトル識別子によって表されるタイトルを実行する手段と、
を含む、前記装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ソフトウェア・アプリケーションを自動的に選択する方法に関する。詳細には、本発明はＪａｖａ（登録商標）アプリケーションのライフサイクル管理に関する。
【背景技術】
【０００２】
例えば、ディジタル多用途ディスク（ＤＶＤ）などの事前記録光記憶媒体に記憶されたマルチメディア・データは、様々なビデオ・タイトルやゲームなど、様々なアプリケーションに属する可能性がある。これらは通常は、個々に開始することができるアプリケーションとして、媒体のプログラミング構造に組み込まれる。
【０００３】
しかし、事前記録光媒体の場合には、明確なコンテンツの外観と、例えば対話性などのユーザ・エクスペリエンスとを備え、且つ洗練されたオーサリングを行うことができる見込みがあることが望ましい。Ｊａｖａ技術は、その動作環境のハードウェアおよびオペレーティング・システム（ＯＳ）から独立しており、例えばグラフィックやネットワーク・インタフェースなども含む本格的なプログラミング環境を提供するので、これらの要件を満たすことができる。Ｊａｖａ技術では、Ｊａｖａ仮想マシン（ＪＶＭ）と呼ばれる実行ユニットを使用して、クラス・ファイルと呼ばれるファイルから取り出したいわゆるバイトコードを実行する。クラス・ファイルは、ライブラリまたはパッケージとして編成することができる。ＪＶＭは、抽象機械として規定され、その実行に処理ユニット・ハードウェアを必要とする別個のソフトウェア層として理解することができる。
【０００４】
家庭用電子機器内のＪａｖａ実行時環境には、様々な構成およびプロフィルがあり、各プロフィルは特定のタイプの機器に適している。様々なプロフィルの間の主な相違点は、それらがどのライブラリに対応しているかという点である。例えば「Ｐｅｒｓｏｎａｌ Ｂａｓｉｓ Ｐｒｏｆｉｌｅ（パーソナル・ベーシス・プロファイル）」や「Ｐｅｒｓｏｎａｌ Ｐｒｏｆｉｌｅ（パーソナル・プロファイル）」などのプロフィルは、家庭用電子機器により適している。
【０００５】
従来のアプリケーション・モデルでは、アプリケーションは活動状態（ａｃｔｉｖｅ）または非活動状態（ｉｎａｃｔｉｖｅ）の何れかの状態にある。この基本モデルを超えて

、上述のプロフィルは、「Ｘｌｅｔ」アプリケーション・モデルと呼ばれる、機能強化したアプリケーション・モデルを規定する。このアプリケーション・モデルは、アプリケーションのライフサイクルに対する細粒度の高い（ｆｉｎｅ－ｇｒａｉｎｅｄ）制御を可能にする。このシステムは、特定のインタフェースを介して実行中の全てのアプリケーションのライフサイクルを制御するソフトウェア構成要素である「アプリケーション・マネージャ（ＡＭ：Ａｐｐｌｉｃａｔｉｏｎ　Ｍａｎａｇｅｒ）」を含んでいる。例えば、ＡＭは、アプリケーションをＪＶＭにロードし、このアプリケーションを開始し、終了することができる。

【０００６】

従って、アプリケーションは、任意の時点において、例えばロード状態（ｌｏａｄｅｄ）、開始状態（ｓｔａｒｔｅｄ）または停止状態（ｓｔｏｐｐｅｄ）などの幾つかのライフサイクル状態のうちの１つにある。アプリケーションは、それ自体でライフサイクル状態を変化させることができる。アプリケーション・マネージャ（ＡＭ）は、アプリケーションがその状態または状態変化を伝えるためのアプリケーション管理インタフェース（ＡＭＩ：Ａｐｐｌｉｃａｔｉｏｎ　Ｍａｎａｇｅｍｅｎｔ　Ｉｎｔｅｒｆａｃｅ）を提供する。従って、ＡＭは、常に全てのアプリケーションのライフサイクル状態に関する最新の情報を有する。

【発明の開示】

【０００７】

（発明の概要）

本発明が解決する課題は、既存のシステムにＪａｖａを組み込むことである。基本的には、アプリケーション・マネージャ（ＡＭ）とアプリケーションの間の相互作用のフレキシビリティを高めるために、いつどのようにアプリケーションを開始するか、またアプリケーションの終了または中止時に何が起きるのかを規定するライフサイクル・モデル、即ちシステムの挙動を規定するモデルが必要である。

【０００８】

本発明は、例えば事前記録記憶媒体などの記憶媒体に記憶された各アプリケーション毎にマッピング・テーブルを利用することを基本とする。マッピング・テーブルは、上記と同じ記憶媒体に記憶しても別の記憶媒体に記憶してもよく、各アプリケーションのアプリケーション終了状態情報（アプリケーション固有）を、例えば上記と同じ媒体などの媒体に記憶され、従ってディスク固有であり、次に実行すべきタイトルまたはＪａｖａアプリケーションを表す識別子に変換するために使用される。従って、Ｊａｖａアプリケーションが終了するときに、プレーヤは、終了するアプリケーションの終了状態情報を対応するマッピング・テーブルに当てはめ、マッピング・テーブルの戻り値を読み取り、該テーブルから得られた指定エンティティを引き続き実行する。

【０００９】

ソフトウェア・アプリケーションが状態情報を提供する一般的な方法は、終了時に、上位または後続のソフトウェア層によって解析することができる終了コードを出すというものである。

【００１０】

本発明では、記憶媒体に記憶された１つ、複数、または全てのアプリケーションに対して、特定のマッピング・テーブルを使用することが可能である。即ち、本発明の１実施形態では、記憶媒体は、それが保持するＪａｖａアプリケーションの数と同数のマッピング・テーブルを記憶するが、別の実施形態では、記憶媒体は、それより少ない数、ただし少なくとも１つのマッピング・テーブルを記憶し、これが、該記憶媒体に記憶されたアプリケーションの幾つかまたは全てによって共有される。また、記憶媒体は、それが保持するアプリケーションの一部に対してのみ本発明による１つまたは複数のマッピング・テーブルを記憶し、それが保持するアプリケーションの全てに対しては本発明による１つまたは複数のマッピング・テーブルを記憶していないこともある。

【００１１】

本発明は、例えばマルチメディア・ディスクの制作者（ａｕｔｈｏｒ）が、当該ディスクの再生中、特に再生装置上のＪａｖａアプリケーションのライフサイクルが終了した場合に、再生装置のシステム挙動を制御できるようにするフレキシブル且つロバストな（ｒｏｂｕｓｔ：信頼性高い）方法を構成するものである。従って、ディスクの制作者は、ディスク上の独立して取り出すことができる様々なアプリケーションのフレキシブルなシーケンスを予め規定することができ、それによって例えばユーザ入力やプレーヤ（例えば、タイマ・ユニット）からの入力などの実行時の入力に対する反応を改善することもできる。

【００１２】

この方法を利用する装置は、請求項７に開示されている。

【００１３】

本発明の有利な実施形態は、従属請求項、以下の説明、および図面に開示されている。

【発明を実施するための最良の形態】

【００１４】

添付の図面を参照しながら、本発明の例示的な実施形態について述べる。

【００１５】

図１は、例えばＤＶＤやブルーレイ・ディスクなどの事前記憶媒体に記憶されたマルチメディア・データおよびメニュー・データの構造を示す図である。記憶媒体をプレーヤ即ち再生装置に挿入すると、プレーヤはインデックス・テーブルＩＴを検索し、検出する。更に、プレーヤは、インデックス・テーブルの第１のエントリＩＮ＿ＰＢがＪａｖａアプリケーションＪＦであることを検知する。これを検知することができるのは、例えば、第１のエントリＩＮ＿ＰＢがＪａｖａクラス・ファイルを示す拡張部を有するファイル名であるときである。次いで、プレーヤはＪａｖａアプリケーション・マネージャ（ＡＭ）およびＪａｖａ仮想マシン（ＪＶＭ）をセットアップして、指定されたファイルのプログラムを実行する。例えば、指定されたアプリケーションＪＦにより、イントロ・シーケンスなど、特定の音声映像（ＡＶ）シーケンスまたは動画オブジェクトの再生が開始される。

【００１６】

このＡＶシーケンスが終了すると、ＪａｖａアプリケーションＪＦは終了し、プレーヤは引き続きインデックス・テーブルＩＴを順次検索する。次のテーブル・エントリＴ＿ｍｅｎｕは、ＪＶＭ内で実行され、トップ・レベル・メニューおよびバックグラウンド・ビデオ・シーケンスの表示をもたらす別のＪａｖａアプリケーションＪＴへのリンクを含んでいる。これは、ユーザが何らかのアイテムを選択するまで表示することができる。

【００１７】

ユーザからの入力が行われない期間を測定して、一定時間後に、例えばインデックス・テーブルのエントリの１つや終了時にトップ・レベル・メニューに復帰する追加アニメーションなどのアプリケーションを自動的に実行すると、有用であると考えられる場合もある。

【００１８】

アイテムが選択されると、実行中のアプリケーションＪＴは自動的に終了され、終了値を戻す。プレーヤは、この終了値を用いて、次に開始するアプリケーション、例えば特定のタイトルＴｉｔｌｅ＿１を識別することができる。本発明によれば、マッピング・テーブルを使用して、次に開始するアプリケーションの識別を行う。マッピング・テーブルは、あるアプリケーションの戻り値と次に開始するアプリケーションの識別子の間の関連付けを行う。

【００１９】

このようなマッピング・テーブルを使用することの利点は、各アプリケーション毎に、１つまたは複数の個別の後続アプリケーションを提供することができることである。従って、コンテンツの制作者は、固定されたシーケンスを規定したり、アプリケーションの終了後にメイン・メニューに戻ったりせずに、幾つかのタイトルを保持するディスク上でフレキシブルなタイトル・シーケンスを規定することができる。例えば、選択された特定の

タイトルＴｉｔｌｅ＿１を再生しているアプリケーションＪ１が終了されたときに、このアプリケーションは、別のＪａｖａアプリケーションＪ３を開始する別のタイトルＴｉｔｌｅ＿３、またはトップ・レベル・メニューから選択不可能である場合もある第３のＪａｖａアプリケーションＪ５にマッピングすることができる終了値を戻す。この場合、インデックス・テーブルＩＴからの新たなタイトルの選択は行われないので、それ以前に活動化されたタイトルは活動状態のままである。以前に選択されたタイトル、即ちインデックス・テーブル・エントリは活動状態である。このインデックス・テーブル・エントリは、Ｊａｖａアプリケーションへのリンクとすることができるが、別のプログラムとすることもできる。

【００２０】

本発明によるマッピング・テーブルは、コンテキスト固有且つアプリケーション固有である。即ち、各アプリケーションは、現在のコンテキスト、例えばエディションに対する独自のマッピング・テーブルを有することができる。しかし、多くの場合には、異なるアプリケーションが同一のマッピング・テーブルを共有することができる。このテーブルによって与えられる識別子は、例えばアプリケーションと関連付けられ、且つそれを通じて当該アプリケーションを呼び出すことができるタイトルまたはアプリケーション識別子（ＡＰＰＬ＿ＩＤ）などの数値または英数字値とすることができる。また、識別子は、例えばファイル名など、指定されたタイプまたはクラスのパラメータであってもよい。各Ｊａｖａアプリケーションに関連付けられたアプリケーション識別子（ＡＰＰＬ＿ＩＤ）を用いることによって、１つまたは複数のＪａｖａアプリケーションを１つのタイトルに割り当てることができる。

【００２１】

図２では、マッピング・テーブルＭＴを用いたアプリケーション・マネージャは、終了するＪａｖａアプリケーションＪ１の終了状態コードＥＳから、ＪＶＭで実行される別のＪａｖａアプリケーションを指定するインデックス・テーブルＩＴのエントリＴｉｔｌｅ＿４を指す識別子を決定している。マッピング・テーブルＭＴは、それが有効であるアプリケーションＪ１の可能な終了状態コードＥＳのそれぞれに対して、次に実行するアプリケーションの有効識別子Ａｐｐ＿ＩＤを提供する。これが可能であるのは、アプリケーションのデータ・タイプが分かっており、従ってアプリケーションの戻り値のビット幅が分かっているからである。図２のマッピング・テーブルＭＴは、最小値ＭＩＮに対するエントリと、最小値と０の間の値に対する１つのエントリと、終了状態コード０、１および２に対してそれぞれ１つずつのエントリと、２より大きく最大値より小さい終了状態コードに対するエントリと、最後に最大値に対するエントリとを含んでいる。終了コード値は、例えば４ビット幅とすることができ、従って、前述の例と同様に、符号付きの整数値－８、…、７と解釈することができる。戻り値は、符号なしの整数値のタイプであってもよいし、その他の任意のタイプであってもよい。

【００２２】

一般に、Ｊａｖａアプリケーションの終了方法は２通りある。アプリケーション・マネージャによって終了を開始する方法と、アプリケーション自体によって終了を開始する方法である。記憶媒体から取り出されるアプリケーションの場合、前者は、これはユーザが例えばタイトル検索などを進める方法を規定することを意味するが、後者は、媒体の制作者が次に起きることを規定することを意味する。従って、本発明によれば、フレキシブルな手段によって、コンテキストの制作者がアプリケーションの終了時に次にどこに進むかという情報を提供することが可能になる。

【００２３】

マッピング・テーブルは、大まかに言えば、入力値を出力値と関連付ける、即ちディスク上の１つのアプリケーションの戻り値を当該ディスク上の別のアプリケーションの識別子と関連付けるデータ構造である。マッピング・テーブルは、Ｊａｖａアプリケーションが終了し、アプリケーション・マネージャ（ＡＭ）によって処理される場合にシステムの挙動を指定する目的に適う。概念的なマッピング・テーブルの例を、表１に示す。

【００２４】
【表１】

終了状態	識別子	要求再開フラグ
Integer.MIN_VALUE	Title 1	0
…	…	…
0	Title 1	1
1	Application 98765	0
…	…	…
Integer.MAX_VALUE	Title 2	0

表１　例示的な概念マッピング・テーブル

【００２５】
この表の第１欄は、アプリケーションから渡される可能性のある終了状態値を規定している。本発明の好ましい実施形態では、これは整数である。この表は、最小値から最大値まで、アプリケーションが戻す可能性がある全ての値をカバーしている。別の実施形態では、最小値および最大値の代わりにデフォルト値が与えられる。このデフォルト値は、例えば終了するアプリケーションが値を戻さない場合、または戻された値が表１中の他のエントリと一致しない場合に使用することもできる。
【００２６】
表１の第２欄は、タイトルまたはＪａｖａアプリケーションを表す識別子を示している。現況技術の事前記録光ディスクは、いわゆるタイトルに論理的に分割され、各タイトルは関連するタイトル番号を有する。本発明のマッピング・テーブルでは、「識別子」欄でタイトル番号を示すことができる。これにより、別のタイトルへの移行を予め規定する、またはオーサリングすることが可能になる。例えばプレコマンド（ｐｒｅ－ｃｏｍｍａｎｄ）、ＡＶ再生コマンドおよびポストコマンド（ｐｏｓｔ－ｃｏｍｍａｎｄ）を規定するスクリプトなどＪａｖａ以外の既知の手段によってタイトルがオーサリングされる場合には、この移行に、プレーヤの「モード変更」が含まれることもある。また、この表により、識別子欄において、タイトルの代わりにＪａｖａアプリケーションを指定することも可能になる。この場合には、プレーヤは、タイトルまたはモードの移行を行うことなく、現在のタイトル内で指定されたアプリケーションを開始することができる。
【００２７】
更に、別のタイトルへの移行が起こった場合に、プレーヤは、当該タイトルの実行時に再開情報を保持することができる。制作者が当該タイトルを再開したいと望み、記憶した再開情報を評価するようプレーヤに要求する場合には、表中の関連するフィールドに「要求再開フラグ」を立てることによって、これを表すことができる。これを使用して、例えば、複数回呼び出されたアプリケーションの異なる挙動または異なる戻り値を制御することができる。
【００２８】
図３は、終了状態コードＥＳを「２」としてアプリケーションＪ１が終了する様子を示している。アプリケーション・マネージャ（ＡＭ）は、マッピング・テーブル中でこの値を検索し、関連するエントリ中に関連するアプリケーション識別子Ａｐｐ＿ＩＤ「９９」を発見する。ＡＭは、この識別子をＪａｖａアプリケーションＪ９９の識別子であると認識する。ＡＭは、指定されたＪａｖａアプリケーションＪ９９をＪａｖａ仮想マシン（ＪＶＭ）にロードし、ＪＶＭはこのアプリケーションを実行する。この場合も、指定されたＪａｖａアプリケーションＪ９９をインデックス・テーブルＩＴから直接開始することもできる。アプリケーションＪ９９が終了すると、ＡＭは戻り値ＥＳ９９を受信し、ＡＭは

、次に実行するアプリケーションを決定するために、この戻り値をマッピング・テーブルＭＴ中で検索する。

【００２９】

図４は、ユーザ入力の処理方法を示す図である。トップ・レベル・メニューを実行しているＪａｖａアプリケーションＪＭは、ユーザがメニュー・アイテムを選択すると、終了状態値を戻す。アプリケーションＪＭの終了状態値は、例えば遠隔制御装置（リモコン）などの入力装置を介してユーザがどのメニュー・アイテムを選択したかに応答して異なる。この終了状態値は、上述の本発明のマッピング・テーブルを用いて、インデックス・テーブルＩＴのその他のエントリの１つにマッピングされる。選択されたインデックス・テーブル・エントリの１つＴｉｔｌｅ＿２は、ＡＭの制御の下で、ただしＪａｖａ仮想マシン（ＪＶＭ）は使用せずに、例えば動画の再生を開始することができるが、別のインデックス・テーブル・エントリＴｉｔｌｅ＿３は、ＪＶＭ上で別のＪａｖａアプリケーションＪＴを開始し、このアプリケーションが次いで別のビデオ・タイトルを再生する。

【００３０】

アプリケーションが戻す終了状態値は、プレーヤ、或いはユーザ入力以外のその他の要因、例えばタイマ・ユニット、擬似乱数発生器、またはパーソナル・コンピュータのような別の接続された装置の影響を受けることもある。

【００３１】

図５は、４つのタイトル、Ｔｉｔｌｅ＿１からＴｉｔｌｅ＿４を備えた、記憶媒体の例示的なショート・メニューを示す図である。プレーヤが記憶媒体の読取りを行うときに、プレーヤはインデックス・テーブルＩＴを評価し、ＡＭはそのエントリを順次検索する。第１のエントリＦｉｒｓｔＰｌａｙｂａｃｋは、ＪａｖａアプリケーションＪＦの識別子を含んでいる。このアプリケーションは、ＪＶＭにロードされ、実行される。このアプリケーションは、終了時に終了コード値を戻さず、マッピング・テーブルはこれと関連付けられない。従って、ＡＭは引き続きインデックス・テーブルＩＴを順次検索し、次のエントリとして、別のＪａｖａアプリケーションＪＴの識別子Ｔ＿ｍｅｎｕを発見する。このアプリケーションも、ＪＶＭにロードされて実行され、ユーザに対してメニューの表示を行うことになる。しかし、このアプリケーションＪＴは、ユーザ入力があったときに終了するか、またはユーザ入力が行われずに既定の時間が経過した後で自動的に終了するかの何れかである。これは、タイマ・ユニットによって制御することができる。例えばユーザがメニューからＴｉｔｌｅ＿３を選択した場合には、実行中のアプリケーションＪＴは、終了コード値「３」をＡＭに戻す。ＡＭは、この終了するアプリケーションに関連するマッピング・テーブルＭＴ＿ＪＴを使用して、終了コード値「３」を、例えば対応するインデックス・テーブル・エントリＴｉｔｌｅ＿３を指すアプリケーションまたはタイトルの識別子「１０３」にマッピングする。ユーザが別のメニュー・アイテムを選択した場合には、メニュー・アプリケーションＪＴの戻り値は、別のインデックス・テーブル・エントリを指すか、或いはアプリケーションＪ４１を直接指すことができる。選択されたインデックス・テーブル・エントリＴｉｔｌｅ＿３は、別のＪａｖａアプリケーションＪ３を表し、このアプリケーションが次いでＪＶＭにロードされて実行され、例えば動画またはゲームが再生される。

【００３２】

このアプリケーションは、終了時に終了状態値を、例えば２つの異なるアプリケーションＪ３およびＪ４に対して有効なマッピング・テーブルＭＴ＿Ｊ４を使用するＡＭに戻す。ＪａｖａアプリケーションＪ３は、実際には別の記憶媒体や同じ媒体の別のバージョンなどに合わせてプログラムされている可能性もあり、２から７の間の終了コード値を戻す可能性がある。しかし、制作者は、この媒体のこのエディションについてはアプリケーションＪ３からの終了コード値２、６および７を無視し、別のエディションについてはこれらのコードが別の機能を有するように決めておくことができる。従って、別のアプリケーションＪ４と同じマッピング・テーブルＭＴ＿Ｊ４を使用することができる。一般に、２つ以上のアプリケーションの関連する戻り値が同じである場合には、それらのアプリケー

ションは１つのマッピング・テーブルを共有することができる。従って、マッピング・テーブルは媒体固有であるが、アプリケーションの戻り値はアプリケーション固有である。

【００３３】

本発明によるアプリケーションをマッピング・テーブルに関連付ける方法は多様である。有効な終了コード・マッピング・テーブルの決定はいつでも行うことができるが、アプリケーションの実行中に行うことが好ましい。アプリケーションは実行可能なプログラムであればどのようなものであってもよく、マッピング・テーブルは、入力値と出力値を関連付けるものであればどのようなデータ構造であってもよい。アプリケーションとマッピング・テーブルの間の関連付けは、例えばファイル名を介して、ディレクトリ構造を介して、または別個のマッピング・テーブルを介して行われる。

第１の場合には、アプリケーション・プログラムと、マッピング・テーブルを保持するファイルとが、例えば同じファイル名を有し、ファイル名の拡張部が異なっている。

第２の場合には、アプリケーション・プログラムと、マッピング・テーブルを保持するファイルとを、媒体上の同じディレクトリまたはフォルダに記憶しておけばよい。

第３の場合には、実行中のアプリケーションのアプリケーション識別子を対応する終了コード・マッピング・テーブルの識別子にマッピングする別個のマッピング・テーブルを、媒体に記憶しておけばよい。

【００３４】

本発明によれば、アプリケーションがそのライフサイクルの終了を開始するときには、次のステップが実行される。第１に、終了状態値を与える。第２に、マッピング・テーブルを用いて、この終了状態値を識別子に変換する。第３に、プレーヤが、この識別子が表すエンティティを引き続き実行する。以下、これらのステップについて更に詳細に説明する。

【００３５】

「従来の」アプリケーション・モデルでは、終了状態値は、Ｓｙｓｔｅｍ．ｅｘｉｔ（ｉｎｔ　ｓｔａｔｕｓ）メソッドで与えられる、即ち終了するアプリケーションが、その状態を表す整数値を次の制御層、即ちＡＭに戻す。Ｘｌｅｔアプリケーション・モデルの場合には、アプリケーションは、そのライフサイクルの終了のみをＡＭＩを通じてＡＭに伝える。更に、本発明によれば、終了するアプリケーションは、終了状態をＡＭに提供する。これは、様々な方法で行うことができる。好ましい実施形態では、アプリケーションが終了状態値を直接ＡＭに渡すようにＡＭＩを拡張する。本発明の別の実施形態では、ライフサイクルの終了を伝える前に、アプリケーションおよびＡＭの両方にとって既知であり且つ利用可能である指定位置に終了状態値を記憶する。この目的のために、任意の種類の揮発性または不揮発性メモリを使用することができる。

【００３６】

アプリケーションがＡＭに対して終了状態情報を送出しない場合、例えば現況技術のＡＭＩが使用されている場合には、テーブルは、このような場合に対処する特殊なエントリを提供することもできるし、或いはデフォルト値を使用することもできる。

【００３７】

第２のステップで、ＡＭは、マッピング・テーブル中の情報を使用して、実際の終了状態を識別子に変換する、即ちテーブル索引を行う。

【００３８】

第３のステップで、プレーヤは、この識別子が表す構成要素を引き続き実行する。上述のように、タイトル番号が指定された場合には、別のタイトルへの移行である可能性があり、この移行には、「モード変更」が含まれることもある。この識別子がＪａｖａアプリケーションを表す場合には、プレーヤは現在のタイトル内で指定されたアプリケーションを開始する。即ち、タイトルまたはモードの移行は不要である。

【００３９】

上述の表１は、概念的な表である。この表は、例えば以下の表２に示すようにデフォルト値を用いることにより、上記で説明したよりも効率的に記憶することができる。

【００４０】
【表２】

int status	Title# / App_ID
0	3781
1	8273
2	3922
デフォルト	1

表２　より効率的に符号化したマッピング・テーブル

【００４１】
好ましい実施形態では、このテーブルは、対応するアプリケーションの可能な終了状態値の全範囲をカバーする。別の実施形態では、このテーブルは、全範囲をカバーする必要はなく、幾つかの識別子フィールドを空欄のまま残す。この場合には、戻り値をアプリケーションまたはタイトルにマッピングすることができない場合には、いかなる移行も行われない。
【００４２】
アプリケーションまたはタイトルを特定する表２の右欄の値に対しては、例えばタイトル識別子に対する第１の範囲と、アプリケーション識別子に対する第２の範囲など、複数の値範囲を確保することができる。
【００４３】
本発明は、Ｊａｖａアプリケーションのライフサイクルの終了時のシステムの挙動を指定する非常にロバストな手段を提供するので有利である。特に、マッピング・テーブルは、その一貫性をディスクが一般公開される前に容易に検証することができるので、アプリケーション実行時に、任意の終了状態値に対して有効なタイトルまたはアプリケーションが起動されることを保証する。
【００４４】
本発明のもう１つの利点は、Ｊａｖａアプリケーションを容易に再使用できる点である。これは、上述のように１つのアプリケーションを異なる構成で同一のディスクと共に再使用すること、並びに特定の１つのアプリケーションを使用して、異なるディスクのオーサリングを行うことを含んでいる。システムの移行動作はマッピング・テーブルによって規定され、従ってアプリケーション自体とは別個に規定されるので、同じアプリケーションを異なるコンテキストで容易に再使用することができる。例えば、同じアプリケーションを、異なるディスクに容易に組み込むことができる。マッピング・テーブルの適応のみを行えばよく、アプリケーション・コードを変更するためのプログラムミングは不要である。
【００４５】
本発明のもう１つの利点は、アプリケーションの終了時、即ちアプリケーションのライフサイクルの終了時に、別のＪａｖａアプリケーションを起動することができることである。これにより、幾つかのアプリケーションを１つのタイトルに一体化する簡単な手段が得られる。現況技術の解決策では、異なるアプリケーションを単一のアプリケーションに一体化するためにはプログラミング作業が必要となる。従って、本発明は、フレキシブルであり且つ細粒度の高いタイトルの構成をサポートする。このモジュール性の改善により、更なるフレキシビリティがコンテキストの制作者に与えられる。
【００４６】
更に、本発明は、Ｊａｖａアプリケーションを現況技術のシステムに一体化する構成ブロックとして機能することができる。
【００４７】

本発明の実施形態では、終了するソフトウェア・アプリケーションおよび次に実行されるソフトウェア・アプリケーションのプログラム・データは、読取り専用記憶媒体から取り出され、マッピング・テーブルを表すデータは、例えばハード・ディスク・ドライブ（ＨＤＤ）などの書換え可能な記憶媒体から取り出される。このことには、マッピング・テーブルを後に変更することができるという利点がある。２つのマッピング・テーブルが利用可能であり、一方が終了するアプリケーションを記憶した媒体に記憶され、もう一方がＨＤＤなどのローカル記憶媒体に記憶される場合には、例えば「ＨＤＤのマッピング・テーブルの方が優先される」などの規則を定めておくことができる。

【００４８】

本発明の一般概念は、アプリケーションの終了状態情報（アプリケーション固有）を、タイトルまたはＪａｖａアプリケーション（ディスク固有）を表す識別子に変換するために使用されるマッピング・テーブルの概念である。Ｊａｖａアプリケーションが終了すると、プレーヤはマッピング・テーブルを評価し、引き続きこのテーブルから得られた指定エンティティを実行する。

【００４９】

本発明の主な利点は、他のタイトルまたはＪａｖａアプリケーションへの移行がフレキシブルになり、ロバストネス（信頼性）の改善によって検証がより容易になり、アプリケーションの再使用が容易になり、フレキシビリティが改善したことである。特に、オーサリング後の媒体の検証は、媒体全体を様々なコンテキストで一度に検証する必要がなくなり、モジュール単位で段階的に行うことができるので、より容易に行うことができる。

【００５０】

本発明は、ＪＶＭで実行されるＪａｖａアプリケーションなど、仮想マシン上で実行されるアプリケーションのデータを含むデータ構造、特に記憶媒体に記憶されたデータ構造に使用することができる。本発明は、Ｊａｖａプログラミング言語で書かれたアプリケーション・プログラムと共に使用することもできるし、或いはＪＶＭまたは一般的な仮想マシン上で実行されるその他の言語で書かれたプログラムと共に使用することもできる。

【図面の簡単な説明】

【００５１】

【図１】マルチメディア・ディスクに記憶された再生データの構造、および関連するＪａｖａアプリケーションを示す図である。

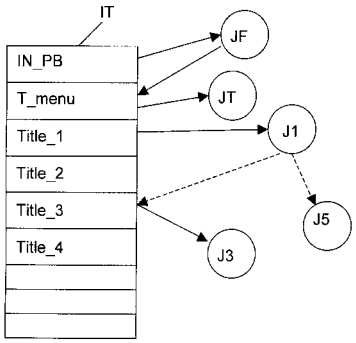
【図２】マッピング・テーブルを使用して、終了状態コードを次のタイトルの識別子にマッピングする様子を示す図である。

【図３】マッピング・テーブルを使用して、終了状態コードを次のアプリケーションの識別子にマッピングする様子を示す図である。

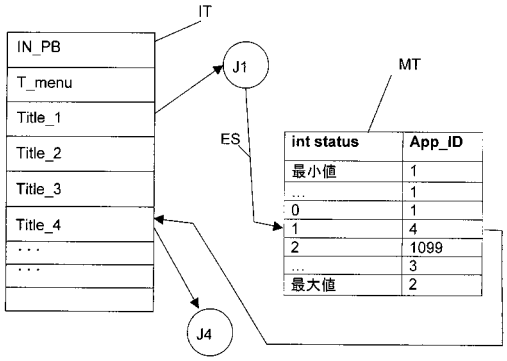
【図４】Ｊａｖａタイトル・メニューの構造を示す図である。

【図５】異なるアプリケーションが１つのマッピング・テーブルを共有する、例示的な複数のマッピング・テーブルを示す図である。

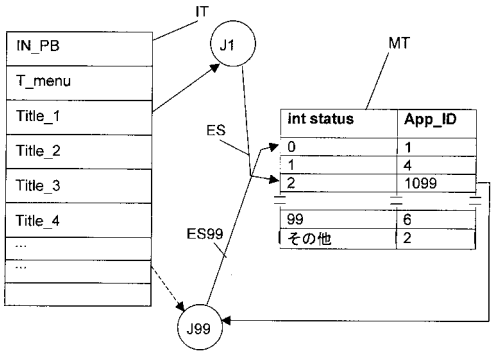
【図１】


IT
IN_PB
T_menu
Title_1
Title_2
Title_3
Title_4
JF
JT
J1
J5
J3

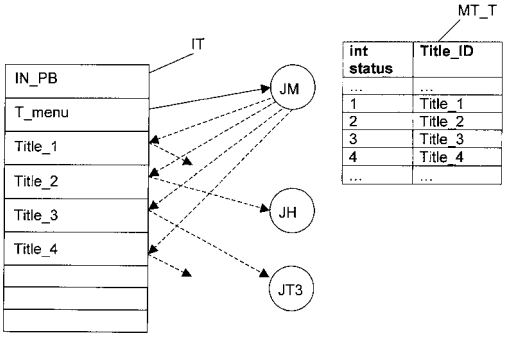
【図２】


IT
IN_PB
T_menu
Title_1
Title_2
Title_3
Title_4
J1
ES
MT
int status
App_ID
最小値
1
0
1
2
1099
最大値
2
J4

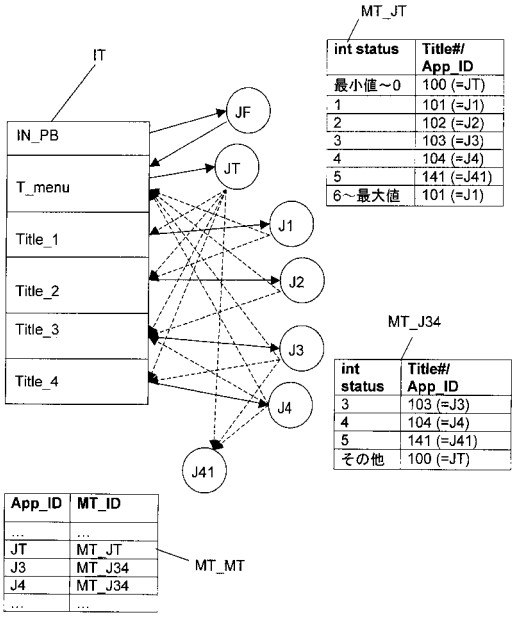
【図３】


IT
IN_PB
T_menu
Title_1
Title_2
Title_3
Title_4
J1
MT
ES
ES99
int status
App_ID
0
1
1
4
2
1099
99
6
その他
2
J99

【図４】


IT
IN_PB
T_menu
Title_1
Title_2
Title_3
Title_4
JM
JH
JT3
MT_T
int status
Title_ID
1
Title_1
2
Title_2
3
Title_3
4
Title_4

【図５】


IT
IN_PB
T_menu
Title_1
Title_2
Title_3
Title_4
JF
JT
J1
J2
J3
J4
J41
MT_JT
int status
Title#/ App_ID
最小値～0
100 (=JT)
1
101 (=J1)
2
102 (=J2)
3
103 (=J3)
4
104 (=J4)
5
141 (=J41)
6～最大値
101 (=J1)
MT_J34
int status
Title#/ App_ID
3
103 (=J3)
4
104 (=J4)
5
141 (=J41)
その他
100 (=JT)
App_ID
MT_ID
JT
MT_JT
J3
MT_J34
J4
MT_J34
MT_MT

フロントページの続き

(72)発明者　アドルフ，デイルク
ドイツ国　３０９５２　ロネンベルク　バールブリンク　２
(72)発明者　オスターマン，ラルフ
ドイツ国　３０６５７　ハノーフアー　ニデナー・ベーグ　７
(72)発明者　ヘルペル，カーステン
ドイツ国　３０９７４　ベニングセン　シユワルツエードルン－シユトラツセ　４
(72)発明者　ヤンセン，ウベ
ドイツ国　３０９２６　ゼールツエ　ニーダーザクセンシユトラツセ　５３
(72)発明者　ペータース，ハルトムート
ドイツ国　３０８９０　バールジンクハウゼン　オーベーグ　３４
(72)発明者　シエウツオウ，アンドレ
ドイツ国　３０１６３　ハノーフアー　タラベラシユトラツセ　１４
(72)発明者　ビンター，マルコ
ドイツ国　３０１７３　ハノーフアー　ベーマーシユトラツセ　１７

審査官　早川　卓哉

(56)参考文献　特表２００３－５０４７５３（ＪＰ，Ａ）
特表２００１－５０３５４７（ＪＰ，Ａ）
特開２００３－１７７９３３（ＪＰ，Ａ）
特開２００３－１４１０７０（ＪＰ，Ａ）
特開平０６－２７４３２８（ＪＰ，Ａ）
特開平０５－０１２０３７（ＪＰ，Ａ）
米国特許第０５６０７３５６（ＵＳ，Ａ）
特表２００７－５０８６１２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G11B20/10-20/16
G11B27/00-27/34
G06F9/06
G06F9/46