(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4726096号
(P4726096)
(45)発行日　平成23年7月20日(2011.7.20)　(24)登録日　平成23年4月22日(2011.4.22)

(51)Int.Cl.　F I
H04N 7/26 (2006.01)　H04N 7/13 Z

請求項の数 20　(全 12 頁)

(21)出願番号　特願平10-543002
(86)(22)出願日　平成10年4月7日(1998.4.7)
(65)公表番号　特表2000-513178(P2000-513178A)
(43)公表日　平成12年10月3日(2000.10.3)
(86)国際出願番号　PCT/US1998/006790
(87)国際公開番号　WO1998/046024
(87)国際公開日　平成10年10月15日(1998.10.15)
審査請求日　平成17年3月31日(2005.3.31)
審判番号　不服2008-14401(P2008-14401/J1)
審判請求日　平成20年6月9日(2008.6.9)
(31)優先権主張番号　60/042,801
(32)優先日　平成9年4月7日(1997.4.7)
(33)優先権主張国　米国(US)

(73)特許権者　390035493
エイ・ティ・アンド・ティ・コーポレーション
ＡＴ&Ｔ　ＣＯＲＰ.
アメリカ合衆国　１００１３－２４１２
ニューヨーク　ニューヨーク　アヴェニュー　オブ　ジ　アメリカズ　３２

最終頁に続く

(54)【発明の名称】ＭＰＥＧコード化オーディオ・ビジュアル対象物を表すビット・ストリームの発生およびインターフェースで連結するためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
ＭＰＥＧ－４規格に従ってコード化された、ストリーム状のオーディオ・ビジュアル・オブジェクトを処理するためのシステムであって、
オーディオ・ビジュアル・オブジェクトを処理するための、処理資源又はユーザ入力に適応され得るインターフェースを提供するストリーム形成制御機能の組を含むストリーム形成インターフェース・ライブラリ、該組のストリーム形成制御機能の各々は所定の機能呼出を有しており、及び
前記ストリーム形成インターフェース・ライブラリにアクセスし、前記機能呼出に従ってストリーム状のオーディオ・ビジュアル・オブジェクトを復号するように構成されたプロセッサとからなる処理システム。
【請求項２】
請求項１に記載のシステムにおいて、前記プロセッサが、機能呼出を呼び出すクライアント・アプリケーションを実行する処理システム。
【請求項３】
請求項１に記載のシステムにおいて、プロセッサと通信して選択した機能呼出を呼び出すユーザ・入力ユニットを更に備える処理システム。
【請求項４】
請求項１に記載のシステムにおいて、前記インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームの、ビジュアル・オブジェクトを復号するためのビジ

ュアル復号インターフェースを備える処理システム。
【請求項５】
請求項１に記載のシステムにおいて、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームを発生および出力するための発生機能を備える処理システム。
【請求項６】
請求項１に記載のシステムにおいて、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームを編集する編集機能を備える処理システム。
【請求項７】
請求項１に記載のシステムにおいて、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームを解釈するための解釈機能を備える処理システム。
【請求項８】
請求項１に記載のシステムにおいて、前記プロセッサが、システム資源の変更に従って、前記ビット・ストリーム・インターフェース・ライブラリの実行を適合させるシステム。
【請求項９】
請求項１に記載のシステムにおいて、協力するクライアント・アプリケーションが提供する追加ビット・ストリーム機能を呼び出す、クライアント・アプリケーション・インターフェースを更に備える処理システム。
【請求項１０】
請求項１に記載のシステムにおいて、ユニット閲覧のためにビット・ストリーム・インターフェース・ライブラリを使用するマルチメディア・ブラウザ・モジュールを更に備える処理システム。
【請求項１１】
ＭＰＥＧ－４規格に従って、コード化されたストリーム形成オーディオ・ビジュアル・オブジェクトを処理するための方法であって、
処理資源又はユーザ入力に適応され得るインターフェースを提供する、各制御機能が予め定めた機能呼出を持つオーディオ・ビジュアル・オブジェクトを処理するストリーム形成制御機能の組を含むストリーム形成インターフェース・ライブラリをプロセッサにインストールするステップ、該組のストリーム形成制御機能の各々は所定の機能呼出を有しており、そして
ストリーム形成インターフェース・ライブラリへの呼出を処理し、前記機能呼出に従ってストリーム状のオーディオ・ビジュアル・オブジェクトを復号するステップとを含む方法。
【請求項１２】
請求項１１に記載の方法において、前記処理・復号ステップが、機能呼出を呼び出すクライアント・アプリケーションを実行するステップを含む方法。
【請求項１３】
請求項１１に記載の方法において、選択した機能呼出を呼び出すユーザ・入力を受信するステップを更に含む方法。
【請求項１４】
請求項１１に記載の方法において、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームに含まれる、ビジュアル・オブジェクトを復号するためのビジュアル復号インターフェースを備えている方法。
【請求項１５】
請求項１１に記載の方法において、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・データ・ストリームを発生および出力するための発生機能を備える方法。
【請求項１６】

請求項１１に記載の方法において、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームを編集する編集機能を備える方法。
【請求項１７】
請求項１１に記載の方法において、前記ストリーム形成インターフェース・ライブラリが、オーディオ・ビジュアル・ビット・ストリームを解釈するための解釈機能を備える方法。
【請求項１８】
請求項１１に記載の方法において、前記処理ステップが、システム資源の変更に従って、前記ビット・ストリーム・インターフェース・ライブラリの実行を適合させるためのステップを含む方法。
【請求項１９】
請求項１１に記載の方法において、協力するクライアント・アプリケーションが提供する追加ビット・ストリーム機能を呼び出す、クライアント・アプリケーション・インターフェースを更にプロセッサにインストールするステップを含む方法。
【請求項２０】
請求項１１に記載の方法において、ユーザ閲覧のためにビット・ストリーム・インターフェース・ライブラリを使用するマルチメディア・ブラウザ・モジュールを更にプロセッサにインストールするステップを含む方法。
【発明の詳細な説明】
関連出願への相互参照
本出願は、優先権を主張する米国特許仮出願６０／０４２，８０１に関連する。
発明の背景
発明の分野
本発明は、コード化マルチメディアおよびその記憶、ユーザへの送信および配布に関し、特にマルチメディア対象物を表すビット・ストリームを発生、編集および解釈するための柔軟な手段が必要な場合のコード化に関する。
関連技術の説明
デジタル・マルチメディアは、操作面、複数のものを発生できるという面、処理面およびエラーに影響されないという利点およびその他の利点を持っているが、必要な記憶容量または送信帯域幅により制限を受ける。それ故、マルチメディアの内容は、多くの場合、圧縮またはコード化する必要がある。さらに、インターネットおよび他のネットワーク上のデジタル・マルチメディアに対する需要が急速に増大しているため、効率的な記憶装置、ネットワーク・アクセスおよびサーチおよび検索に対する必要性も増大しているために、現在までに多数のコード化スキーム、記憶フォーマット、検索技術および送信プロトコルが開発されてきた。例えば、画像およびオーディオ・ファイル用には、ＧＩＦ、ＴＩＦおよび他のフォーマットが使用されている。同様に、オーディオ・ファイルも、リアル・オーディオ、ＷＡＶ、ＭＩＤＩおよび他のフォーマットによりコード化され、記憶される。アニメーションおよびビデオ・ファイルは、多くの場合、ＣＧＩＦ８９ａ、Ｃｉｎｅｐａｋ、Ｉｎｄｅｏおよび他のフォーマットにより記憶される。非常に多数の現存のフォーマットを再生するために、多くの場合、デコーダおよびインタープリタが必要になるが、その場合、前記デコーダおよびインタープリタがハードウェアで実行されるのか、ソフトウェアで実行されるのかにより、特にソフトウェアの場合には、ホストコンピュータの能力により、速度、品質および性能がまちまちである。マルチメディアの内容が、コンピュータ（例えば、パソコン）を通してアクセスされた、ウェブ・ページに含まれている場合には、ウェブ・ブラウザは、予想されるすべての内容に対して正しく設定する必要があり、また前記内容を処理するために、各タイプの内容を認識し、内容ハンドラ（ソフトウェア・プラグインまたはハードウェア）の機構をサポートしなければならない。
相互動作が可能でなければならないこと、品質および性能が保証されていなければならないこと、チップ設計の際のスケールが小さくなければならないこと、また種々様々のフォーマットのための内容を発生する際にコストがかかること等の理由で、マルチメディア・

コード化、パケット化および確実な配布の分野で標準化が促進されてきた。特に、国際規格組織動画専門家グループ（ＩＳＯ　ＭＰＥＧ）は、ＭＰＥＧ－１およびＭＰＥＧ－２と呼ばれる二つの規格の形のコード化マルチメディアに対するビット・ストリーム・シンタックスおよび解読意味論の標準化を進めてきた。ＭＰＥＧ－１は、主として、コンパクト・ディスク（ＣＤ）のようなデジタル記憶媒体（ＤＳＭ）上で使用するためのものであり、一方、ＭＰＥＧ－２は、主として、放送環境（トランスポート・ストリーム）で使用するためものである。しかし、ＭＰＥＧ－２は、またＤＳＭ（プログラム・ストリーム）上で使用するためのＭＰＥＧ－１類似の機構もサポートする。ＭＰＥＧ－２は、また独立のまたはネットワークに接続している、ＭＰＥＧ－２の標準化再生のために必要な場合がある、基本的なユーザの双方向通信のためのＤＳＭ－制御およびコマンドのような追加機能を含む。安価なボードおよびＰＣＭＣＩＡカードが開発され、高速中央処理装置（ＣＰＵ）が入手することができるようになったために、ＭＰＥＧ－１規格が、パソコン上の映画およびゲームの再生に、共通に使用することができるようになった。一方、ＭＰＥＧ－２規格は、比較的高画質のアプリケーション用のものであるため、デジタル衛星ＴＶ、デジタル・ケーブルおよびデジタル汎用ディスク（ＤＶＤ）からの娯楽用アプリケーションに共通に使用されている。前記アプリケーション／プラットホームの他に、ＭＰＥＧ－１およびＭＰＥＧ－２は、ネットワークを通して送られるストリーム、ハードディスクまたはＣＤ上に記憶されているストリーム、およびネットワークによるアクセスおよびローカル・アクセスの組み合わせの種々の他のコンフィギュレーションでの使用が期待されている。

ＭＰＥＧ－１およびＭＰＥＧ－２の成功、インターネットおよび移動チャネルの帯域幅制限、ブラウザを使用するウェブをベースとするデータ・アクセス、および双方向個人通信の必要が増大しているために、マルチメディア使用および制御の新しいパラダイムが誕生した。それに応じて、ＩＳＯ－ＭＰＥＧは、ＭＰＥＧ－４と呼ばれる新しい規格を制定した。前記ＭＰＥＧ－４規格は、個々の対象物の形のオーディオ・ビジュアル情報のコード化、および前記対象物の合成および同期再生用のシステム向けのものである。前記固定システム用のＭＰＥＧ－４の開発が、引き続き行われる一方で、通信、ソフトウェアおよびＪａｖａ言語によるようなネットワーク運営用の新しいパラダイムにより、柔軟で、適応性を持ち、ユーザが双方向で通信できる新しい機会が生まれた。例えば、Ｊａｖａ言語により、ユーザがアクセスしたウェブ・ページで、ホスト役をするウェブ・サーバからの、クライアント・パソコン上でのダウンロードおよびアプレット（ジャバ・クラス）に重要な、ネットワーク運営およびプラットホームの独立が得られる。アプリットの設計により、サーバ側で記憶しているデータへの一回だけのアクセスが必要になる場合もあるし、すべての必要なデータを、クライアントのパソコン上に記憶することもできるし、または数回の部分的アクセス（記憶スペースを少なくし、スタートアップに必要な時間を短縮するために）が必要になる場合がある。後のシナリオは、ストリーム型再生と呼ばれる。

すでに説明したように、コード化マルチメディアが、例えば、パソコンのようなコンピュータ上で、インターネットおよびローカルのネットワーク接続アプリケーションのために使用された場合には、多くの状況が発生する。第一に、マルチメディアのネットワークによるアクセス用の帯域幅は、制限されるか、時間と共に変化するようになり、その場合には、最も重要な情報だけを送信し、その後で、もっと広い帯域幅を使用できるようになったとき、他の情報を送信しなければならない。第二に、使用できる帯域幅がどのような幅のものであっても、解読を行わなければならないクライアント側の、パソコンのＣＰＵおよび／またはメモリ資源は制限を受け、さらに、前記資源も時間により変化する。第三に、マルチメディア・ユーザ（消費者）は、高度に双方向の非直線ブラウジングおよび再生を必要とする場合がある。これは特別のことではない。何故なら、ウェブ・ページ上の多くのテキスト内容は、ハイパー・リンクされた機能を使用することにより、ブラウズすることができ、コード化オーディオ・ビジュアル対象物を使用する表示用に、同じパラダイムの使用が期待されるからである。強化された能力を持たないＭＰＥＧ－４システムは、非常に制限された方法でしか上記の状況を処理できないだろう。

ソフトウェア業界では、長い間、アプリケーション・プログラミング・インターフェース（ＡＰＩ）が、多数の異なるタイプのコンピュータ・プラットホーム上での標準化動作および機能を達成するための手段として認識されてきた。通常、ＡＰＩを定義することにより、種々の動作を標準化することができるが、前記動作の性能は、特定のプラットホームに関心を持つ特定のベンダーが、そのプラットホーム用に最適化された実行を供給する場合があるので、種々のプラットホーム上で異なる場合がある。グラフィックの分野では、バーチャル・リアリティ・モデリング言語（ＶＲＭＬ）は、シーン・グラフ・アプローチを使用することにより、対象物とシーンの説明との間の、空間的および時間的関係を指定する手段を使用することができる。ＭＰＥＧ－４は、リアルタイムのオーディオ／ビジュアル・データおよび顔または体の動作のような効果を処理するために、多くの方法で、ＶＲＭＬおよび拡張ＶＲＭＬに対して、中心の構造物の２進フォーマット・スクリーン表示（ＢＩＦＳ）を使用してきた。ＭＰＥＧ－４規格は、種々のタイプの媒体およびシーン・グラフ表示のコード化のために多くのツールを提供し、さらに、各媒体のコード化が、個々の対象物のコード化を含んでいるので、ビット・ストリーム発生、編集および解釈用の組織化されているが柔軟な機構の開発が待望されている。

発明の概要

本発明は、ＭＰＥＧ－４オーサリング、ビット・ストリームの操作、編集および解釈用の標準化インターフェースに関する。本発明は、前記動作をかなり容易にするためのツールおよびインターフェースを提供する。その結果、ＭＰＥＧ－４規格に適合する一方で、コード化ビット・ストリームをもっと容易に試験、チェックおよびデバッグすることができる。特定のインターフェースは、十分な処理資源が存在しない場合に、ビット・ストリームの編集を可能にすることにより、容易にうまく適合させることができる。前記特定のインターフェースを使用することにより、オーディオ・ビジュアル・アプリケーションに、直接または間接に埋設したユーザの要求、および近い将来に重要になると思われるサービスに応じて、解読可能なビット・ストリームを生成することができる。それ故、本発明は、ビット・ストリームのコード化および解読の従来のシステムの欠点を解決するばかりでなく、うまくデグラデーションおよびユーザの双方向通信をサポートするような、もっと適応性に富んだシステムに内蔵することができるツールを提供する。

より詳しく説明すると、本発明は、ＭＰＥＧ－４規格によりコード化された、オーディオ・ビジュアル対象物を表すビット・ストリームの、柔軟な発生、編集および解釈を容易にするシステムおよびインターフェースによる結合方法を提供する。本発明は、コード化がＭＰＥＧ－４規格により行われるとき、特に、オーディオ・ビジュアル対象物の、ビット・ストリームのコード化および解読を容易にするために、Ｊａｖａ言語によりビット・ストリームの入力／出力パッケージを指定する。これは、提案のパッケージが、固定長のコード化および可変長のコード化を分離し、リアルタイムまたはほぼリアルタイムの動作を助けるために必要な、最適化された実行の可能性を持つ柔軟な文法的関係の分析を行うことができるからである。

本発明が行われた一つの理由は、ＭＰＥＧ－４の開発、ビット・ストリームの操作、編集および解釈用の標準化インターフェースが、必要であったためである。本発明の一つの目的は、前記動作をかなり容易にするためのツールおよびインターフェースを提供し、それにより、ＭＰＥＧ－４規格に適合しながら、コード化ビット・ストリームをもっと容易に試験、チェックおよびデバッグすることができるツールを提供することである。特定のインターフェースは、十分な処理資源が存在しない場合に、ビット・ストリームの編集を可能にすることにより、うまい適合を容易に行うことができる。前記特定のインターフェースを使用することにより、オーディオ・ビジュアル・アプリケーションに直接または間接に埋設したユーザの要求、および近い将来に重要になると思われるサービスに応じて、解読可能なビット・ストリームを生成することができる。

【図面の簡単な説明】

添付の図面を参照しながら、本発明を説明する。図面中、類似の素子には類似の番号が使用されている。

図１Ａは、本発明の一実施形態を詳細に示すコード化システムのブロック図である。
図１Ｂは、本発明の前記実施形態を詳細に示す解読システムのブロック図である。
図２は、本発明のビット・ストリーム発生インターフェースである。
図３は、本発明のビット・ストリーム編集および解釈インターフェースである。
図４は、本発明で使用されるバッファ更新プロセスのフローチャートである。
好適な実施形態の詳細な説明
本発明は、ストリーム状のオーディオ・ビジュアル情報を処理するための、内蔵型インターフェース装置を提供するが、本発明については、ＭＰＥＧ－４環境で説明する。本発明のインターフェース装置は、ＭＰＥＧ－４規格によりコード化された、オーディオ・ビジュアル対象物を表すビット・ストリームの、柔軟な発生、編集および解釈を行うためのビット・ストリーム入力／出力ライブラリを含む。ある角度から見た場合、本発明は、Ｊａｖａ言語で、ビット・ストリームの入力／出力パッケージを定義する。このパッケージ、ｍｐｇｊ．ｂｉｔｓｉｏは、当業者には周知の標準Ｊａｖａライブラリに追加することができ、ＭＰＥＧ－４シンタックス解読に共通な、固定長コード化および可変長コード化を含む、ビット・ストリームの入力および出力動作を簡単にする。本発明は、種々の利点を持つが、最も重要な利点は、前記パッケージが、最大速度に対する最適化を容易に行うことができるように組織されていることである。可変長コード化文法的解析モジュールは、さらに、リアルタイムまたはほぼリアルタイムの動作を助けるために構成可能な多段並列索引を使用することができる。

表１　本発明のビット・ストリーム入力／出力ライブラリ

	クラス	説明
1.	入力ストリーム	このクラスは、ビット・ストリーム入力能力を提供する。
2.	マップ	このクラスは、入力ストリーム・クラス、および出力ストリーム・ストリーム・クラスにより使用される。
3.	出力ストリーム	このクラスは、ビット・ストリーム出力能力を提供する。

このライブラリは、当業者には周知の、ＭＳＤＬ－Ｓ（ＭＰＥＧ－４構文記述語）の、ビット入力／出力部分のＪａｖａ等価部分である。実際、本発明は、またＭＳＤＬ－ＳからＪａｖａのヘトランスレータ（ｆｌｖｏｒｊ）により内部で使用することができる。
図１Ａは、本発明の一実施形態を詳細に示す、コード化システムのブロック図である。コード化される自然をソースとするビデオ・シーンは、リンク１００を通して、ビデオ・セグメンタ１０１に入力され、前記ビデオ・セグメンタは、前記シーンを多数の意味論上の対象物に分割し、ライン１０３、１０４、１０５上に出力する。前記シーンの外側の他のビデオ対象物も、同様に、ライン１０２上で混合することができる。次に、ビデオ対象物１０２、１０３、１０４、１０５は、逐次スイッチ１１２を通過し、媒体１エンコーダ１１８への入力である、ライン１１３上で次々と入手することができる。平行して、ライン１３８上でコード化される、自然をソースとするオーディオ・シーンは、オーディオ・セグメンタ１０６により、個々の他の１０８、１０９、１１０および外部対象物１０７に分割される。次に、オーディオ対象物１０７、１０８、１０９、１１０は、逐次スイッチ１１４を通過し、媒体２エンコーダへの入力である、ライン１１５上で次々と入手すること

ができる。自然のオーディオおよびビデオ対象物の他に、オーディオまたはビデオ合成対象物が、媒体３エンコーダ１２０の入力である、ライン１１１に入力される。開発者が入力した内容１１６に基づいて、シーンの説明が、シーン・グラフ１１７でオプションとして発生するが、これはライン１２４上の出力である。シーン・グラフ１１７は、また任意の制御信号を発生し、この制御信号は、例えば、ライン１２１を通して媒体１エンコーダ１１８に、ライン１２２を通して媒体２エンコーダ１１９へ、またライン１２７を通して媒体３エンコーダ１２０へというように、各媒体エンコーダに送られる。

図には三台の媒体エンコーダを示したが、本発明により使用することができる媒体エンコーダの数には制限はない。さらに、媒体エンコーダそれ自身をサブ・エンコーダから構成することもできる。ライン１２５、１２６および１２７上の媒体エンコーダの出力は、各媒体ビット（ストリーム）発生装置、すなわち、媒体１ビット発生装置１２９、媒体２ビット発生装置１３０、および媒体３ビット発生装置１３１への入力になる。ライン１２４上のシーン・グラフ１１７の出力は、ＢＩＦＳビット発生装置１２８への入力を形成する。ＢＩＦＳビット発生装置１２８および媒体ビット発生装置１２９、１３０および１３１は、図２に詳細に示す本発明のインターフェースを、使用しているものと仮定する。ライン１３２、１３３、１３４および１３５上の種々のビット発生装置の出力は、システム・マルチプレクサ、Ｍｕｘ１３６へ送られる。多重化されたビット・ストリームは、記憶または送信するために、チャネル１３７上で入手することができる。

図１Ｂは、本発明の解読動作のより詳細なブロック図である。いくつかの例外はあるが、本発明のこの機能の動作は、図１Ａのコード化の動作の反対の動作である。多重化されたビット・ストリーム（記憶装置または送信から）は、チャネル１３７上で入手することができ、デマルチプレクサ、Ｄｅｍｕｘ１５１へ入力される。前記デマルチプレクサは、前記ストリームを、ビデオ（自然および合成ビデオ対象物）、オーディオ（自然および合成オーディオ対象物）、ＢＩＦＳシーン記述等のような個々のビット・ストリームに分離する。ＢＩＦＳシーン・ビット・ストリームは、ライン１５２上で入手することができ、ライン１５３上のビデオ対象物ビット・ストリーム、ライン１５４上のオーディオ対象物ビット・ストリーム、ライン１５５上の合成（ビジュアルまたはオーディオ）対象物ビット・ストリームは、ビット（トリーム）エディタ１５６への入力を形成する。前記ビット（トリーム）エディタ５６は、ユーザの要求の滑らかなデグラデーション、または複数の機能を必要とするいくつかの条件に応じる。前記ビット・エディタ１５６は、ビット・ストリームのリアルタイムおよび非リアルタイム編集用に使用することができ、図３に詳細に示す本発明のインターフェースを使用する。

修正したビット・ストリーム、ライン１５７上のＢＩＦＳビット、ライン１５８上の媒体１ビット、ライン１５９上の媒体２ビット、ライン１６０上の媒体３ビットは、各ビット・ストリーム・インタープリタ、すなわち、ＢＩＦＳビット・インタープリタ１６１、媒体１ビット・インタープリタ１６２、媒体２ビット・インタープリタ１６３、媒体３ビット・インタープリタ１６４へ送られ、前記インタープリタは、それぞれの記号の各ストリームをライン１６５、１６６、１６７および１６８に出力する。前記ビット・インタープリタは、本発明の図３のインターフェースを使用する。ライン１６５上のＢＩＦＳ記号は、ライン１７９上にシーン・グラフを形成するために解読される。ライン１６６、１６７、１６８上の種々のタイプの媒体記号ストリームは、それぞれ、媒体デコーダ、すなわち、媒体１デコーダ１７０、媒体２デコーダ１７１、および媒体３デコーダ１７２により解読され、解読された媒体ストリーム（ビデオ対象物、オーディオ対象物、合成対象物等）は、ライン１７６、１７７および１７８上に出力される。図に示すように、シーン・グラフ１６９は、種々の媒体デコーダの一例であり、前記デコーダに対する制御装置である。媒体１デコーダはライン１７３を通して制御され、媒体２デコーダはライン１７４を通して制御され、媒体３デコーダはライン１７５を通して制御される。コンポジタ１８２は、ライン１８０上のシーン・グラフおよびライン１７６、１７７および１７８上の三つの媒体デコーダの出力を入力として取り、ビューア／ユーザに表示するシーンを合成するが、最初に、ライン１８３上のコンポジタの出力が、ライン１８１上のシーン・グラフにより

同様に制御されるレンダーに送られ、合成シーンを出力する。
図２は、本発明のビット・ストリーム発生インターフェースである。ＢＩＦＳビット発生装置１２８は、ライン１２４上のＢＩＦＳ記号を入力として入手し、ライン１３２上に、ビット・ストリームの形の、対応するコード化表現を出力する。同様に、媒体１、２、３ビット発生装置１２９、１３０、１３１は、ライン１２５、１２６および１２７上の媒体記号を、入力として入手し、それぞれ、ライン１３３、１３４および１３５上に、ビット・ストリームの形の、対応するコード化表現を出力する。ＢＩＦＳビット発生装置１２８および媒体１、２、３ビット発生装置１２９、１３０、１３１は、本発明のビット発生装置インターフェース２００を使用する。ビット発生装置インターフェース２００は、ＯｕｔｐｕｔＳｔｒｅａｍ２０１およびＭａｐ２０２のような、複数のＪａｖａクラスからなる。前記クラスのインターフェースの動作を以下に説明する。
Ｃｌａｓｓ　ｍｐｇｊ．ｂｉｔｓｉｏ．ＯｕｔｐｕｔＳｔｒｅａｍ
ｊａｖａ．ｌａｎｇ．Ｏｂｊｅｃｔ
｜
＋－－－－ｍｐｇｊ．ｂｉｔｓｉｏ．ＯｕｔｐｕｔＳｔｒｅａｍ
ｐｕｂｌｉｃ　ｃｌａｓｓ　ＯｕｔｐｕｔＳｔｒｅａｍは、Ｏｂｊｅｃｔを広げる。
ＯｕｔｐｕｔＳｔｒｅａｍは、出力ストリームに対する基本的なインターフェースである。
コンストラクタ
ｐｕｂｌｉｃ　ＯｕｔｐｕｔＳｔｒｅａｍ（ＦｉｌｅＯｕｔｐｕｔＳｔｒｅａｍファイル）は、
ファイル内に新しいＯｕｔｐｕｔＳｔｒｅａｍを作成する。
ｐｕｂｌｉｃ　ＯｕｔｐｕｔＳｔｒｅａｍ（ｓｔｒｉｎｇ　ｂｉｔｓ）は、
ストリング内に、新しいＯｕｔｐｕｔＳｔｒｅａｍを作る。
方法
ｐｕｂｌｉｃ　ｖｏｉｄ　ａｌｉｇｎ（ｉｎｔ　ｎｕｍｂｉｔｓ）は、
ｎｕｍｂｉｔｓの倍数である次のビット境界と整合する。現在のポインタと整合境界との間の複数のビットは、ゼロで書き表される。
ｐｕｂｌｉｃ　ｖｏｉｄ　ａｌｉｇｎ（ｓｔｒｉｎｇ　ｓｔｕｆｆｉｎｇ［］，ｉｎｔ　ｎｕｍｂｉｔｓ）は、
ｎｕｍｂｉｔｓの倍数である次のビット境界と整合する。現在のポインタと整合境界との間の複数のビットは、充填ストリングにより充填される。
ｐｕｂｌｉｃ　ｂｏｏｌｅａｎ　ｅｏｓ（）は、
ストリームの終わりまたはエラーのとき、真に戻る。そうでない場合には、偽に戻る。
ｐｕｂｌｉｃ　ｂｏｏｌｅａｎ　ｅｒｒｏｒ（）は、
エラーの場合、真に戻る。そうでない場合は、偽に戻る。
ｐｕｌｉｃ　ｖｏｉｄ　ｐｕｔｂｉｔｓ（ｉｎｔ　ｎｕｍｂｉｔｓ，ｉｎｔ　ｖａｌｕｅ）は、
指定の数のビットを使用して、符号のない整数を置く。ストリームを書き込むことができない場合、または数値が負である場合には、エラー・フラッグをセットする。
ｐｕｌｉｃ　ｖｏｉｄ　ｐｕｔｓｂｉｔｓ（ｉｎｔ　ｎｕｍｂｉｔｓ，ｉｎｔ　ｖａｌｕｅ）は、
指定の数のビットを使用して、符号付きの整数を置く。最後のビットは、記号ビットになるようにうセットされる。ストリームを書き込むことができない場合には、エラー・フラッグをセットする。
ｐｕｂｌｉｃ　ｖｏｉｄ　ｐｕｔｖｌｃ（Ｍａｐ　ｍａｐ，ｉｎｔ　ｖａｌｕｅ）は、
指定のｍａｐを使用して、指定の数値を置く。ストリームを書き込むことができない場合には、エラー・フラッグをセットする。
Ｃｌａｓｓ　ｍｐｇｊ．ｂｉｔｓｉｏ．Ｍａｐ
ｊａｖａ．ｌａｎｇ．Ｏｂｊｅｃｔ

|
+－－－－mpgj．bitsio．Map
public class Mapは、Objectを広げる。
固定長および可変長コード化用のマップ表
コンストラクタ
public Map（FileInputStream file、int step）は、索引用の指定のステップを使用して、ファイルから読み出す新しいマップを作る。
public Map（FileInputStream file）は、
ファイルから読み出す新しいマップを作る。１回索引を行う度に、１ビットだけ進む。
public Map（string［］bitstring，int［］value，int array－size，int step）は、
アレイ・サイズおよび指定のステップを知り、ストリング・アレイおよび整数アレイから新しいマップを作る。
public Map（string［］bitstring，int［］value，int array－size）は、
アレイ・サイズを知り、ストリング・アレイおよび整数アレイから新しいマップを作る。１回索引を行う度に、１ビットだけ進む。
方法
ユーザ・レベルの方法は供給されない。
図３は、本発明のビット・ストリーム編集および解釈インターフェースである。ＢＩＦＳビット・インタープリタ１６１は、ライン１５７上のＢＩＦＳビット・ストリームを入力として入手し、ライン１６５上に対応する解読記号を出力する。同様に、媒体１、２、３ビット・インタープリタ１６２、１６３、１６４は、ライン１５８、１５９および１６０上の、媒体ビット・ストリームを入力として入手し、それぞれ、ライン１６６、１６７および１６８上に対応する解読記号を出力する。ＢＩＦＳビット・インタープリタ１６１および媒体１、２、３ビット・インタープリタ１６２、１６３、１６４は、本発明のビット・エディタ／インタープリタ・インターフェース３００を使用する。ビット・エディタ／インタープリタ・インターフェース３００は、ＩｎｐｕｔＳｔｒｅａｍ３０３、Ｍａｐ３０２およびＯｕｔｐｕｔＳｔｒｅａｍ３０１のような、複数のＪａｖａクラスからなる。ビット・ストリーム解釈の他に、この図のインターフェースは、またビット・ストリーム編集をサポートする。
ビット・ストリーム編集動作は、ビット・ストリーム発生プロセスに大体類似している。例えば、ビット・エディタ１５６は、ライン１５２、１５３、１５４、４５５上のデマルチプレクスした、ＢＩＦＳおよび媒体ビット・ストリームを入力として入手し、各ライン１５７、１５８、１５９、１６０上に対応する修正した（編集したビット・ストリーム）を出力する。前記編集動作は、過負荷のシステム資源による対象物を捨てる必要に応じて行うことができ、またＢＩＦＳエディタ／インタープリタ・インターフェース３００により動作可能になる。すでに説明したように、このインターフェースは、ＩｎｐｕｔＳｔｒｅａｍおよびＭａｐクラスをサポートするが、前記インターフェースは、またＯｕｔｐｕｔＳｔｒｅａｍクラスをサポートする。後者は、（Ｍａｐと一緒に）ビット・ストリーム編集動作のために必要である。前記ＯｕｔｐｕｔＳｔｒｅａｍ、およびＭａｐクラスのインターフェース動作については、（図２のところで）すでに説明した。ＩｎｐｕｔＳｔｒｅａｍクラスの、前記インターフェース動作は下記のとおりである。
Ｃｌａｓｓ mpgj．bitsio．ＩｎｐｕｔＳｔｒｅａｍ
java．lang．Object
|
+－－－－mpgj．bitsio．ＩｎｐｕｔＳｔｒｅａｍ
public classＩｎｐｕｔＳｔｒｅａｍは、Ｏｂｊｅｃｔを広げる。
このクラスは、ＩｎｐｕｔＳｔｒｅａｍに対する、基本的なインターフェースである。
コンストラクタ

ｐｕｂｌｉｃ　ＩｎｐｕｔＳｔｒｅａｍ（ＦｉｌｅＩｎｐｕｔＳｔｒｅａｍ　ｆｉｌｅ）は、
ファイルから新しいＩｎｐｕｔＳｔｒｅａｍを作成する。
ｐｕｂｌｉｃ　ＩｎｐｕｔＳｔｒｅａｍ（ｓｔｒｉｎｇ　ｂｉｔｓ、ｉｎｔ　ｌｅｎｇｔｈ）は、
ある長さを持つストリングから新しいＩｎｐｕｔＳｔｒｅａｍを作る。
方法
ｐｕｂｌｉｃ　ｖｏｉｄ　ａｌｉｇｎ（ｉｎｔ　ｎｕｍｂｉｔｓ）は、
ｎｕｍｂｉｔｓの倍数である次のビット境界と整合する。現在のポインタと整合境界との間の複数のビットが、読み出され、捨てられる。
ｐｕｂｌｉｃ　ｂｏｏｌｅａｎ　ｅｏｓ（）は、
ストリームの終わりで、真に戻る。そうでない場合には、偽に戻る。
ｐｕｂｌｉｃ　ｂｏｏｌｅａｎ　ｅｒｒｏｒ（）は、
エラーの場合真に戻る。そうでない場合は、偽に戻る。
ｐｕｌｉｃ　ｉｎｔ　ｇｅｔｂｉｔｓ（ｉｎｔ　ｌｅｎｇｔｈ）は、
指定の数のビットから符号のない整数を入手する。ストリームの終わりでｅｏｓフラッグをセットする。ストリームを読むことができない場合は、エラー・フラッグをセットする。
ｐｕｌｉｃ　ｉｎｔ　ｇｅｔｓｂｉｔｓ（ｉｎｔ　ｌｅｎｇｔｈ）は、
指定の数のビットから符号付きの整数を入手する。最後ビットは、前記整数の符号を示す。ストリームの終わりでｅｏｓフラッグをセットする。ストリームを読むことができない場合は、エラー・フラッグをセットする。
ｐｕｂｌｉｃ　ｉｎｔ　ｎｅｘｔｂｉｔｓ（ｉｎｔ　ｌｅｎｇｔｈ）は、
次の指定の数のビットを調べる。数値を３２ビットの整数として戻す。現在のポインタを進めない。ストリームを読むことができない場合は、エラー・フラッグをセットする。
ｐｕｂｌｉｃ　ｖｏｉｄ　ｓｋｉｐｂｉｔｓ（ｉｎｔ　ｌｅｎｇｔｈ）は、
指定の数のビットをスキップする。ストリームの終わりでｅｏｓフラッグをセットする。
ｐｕｂｌｉｃ　ｉｎｔ　ｇｅｔｖｌｃ（Ｍａｐ　ｍａｐ）は、
指定の数値マップに従って可変長コードおよび固定長コードを入手する。数値を３２ビットの整数として戻す。ストリームの終わりでｅｏｓフラッグをセットする。ストリームを読むことができない場合は、エラー・フラッグをセットする。
ｐｕｂｌｉｃ　ｉｎｔ　ｎｅｘｔｖｌｃ（Ｍａｐ　ｍａｐ）は、
指定の数値により固定コード長を調べる。数値を３２ビットの整数として戻す。ストリームの終わりで、ｅｏｓフラッグをセットする。ストリームを読むことができない場合は、エラー・フラッグをセットする。
図４は、データ・バッファを使用する、ある長さのビット・ストリングの処理に、一般的に関連する前記ルーチンの一つの特徴を示す。この図においては、入力の長さをステップ４００で読み取り、その後でステップ４１０でデータ・バッファがチェックされる。必要な動作により、ビット長または他のパラメータに従って、前記バッファに書き込み（ステップ４１５）または読み出し（ステップ４２０）を行うことができる。その後、前記バッファが更新される（ステップ４３０）。
本発明を実行する場合には、柔軟なビット・ストリーム装置が導入され、すべて万能で一貫したストリーム状の方法で、埋設されているオーディオ・ビジュアル対象物を、より簡単でありながら、より複雑な制御を行うことができるように、コア・ルーチンが形成される。
本発明のシステムおよび方法の上記説明は、例示としてのものに過ぎず、当業者なら本発明の構造および実行を種々に変更することができるだろう。例えば、例示としての一組のストリーム形成機能を説明してきたが、ネットワーク、アプリケーションの変更および他の必要により、種々の機能を追加したり、削除することができる。本発明の範囲は、下記の請求の範囲によってのみ制限される。

【図１Ａ】

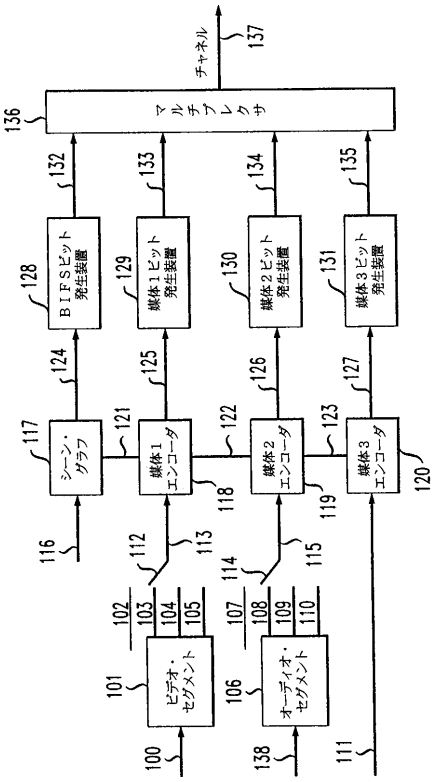

FIG. 1A

【図１Ｂ】

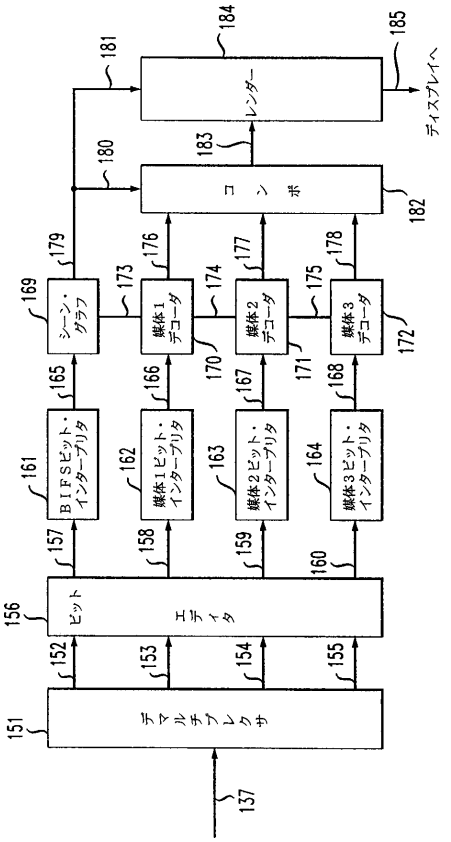

FIG. 1B

【図２】

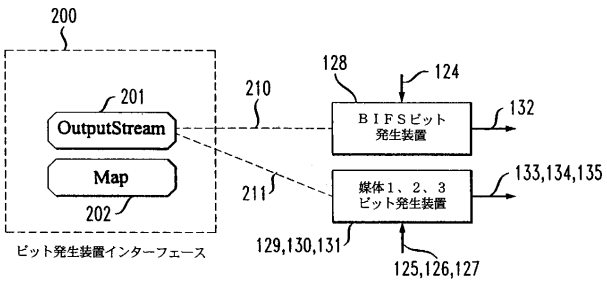

FIG. 2

【図３】

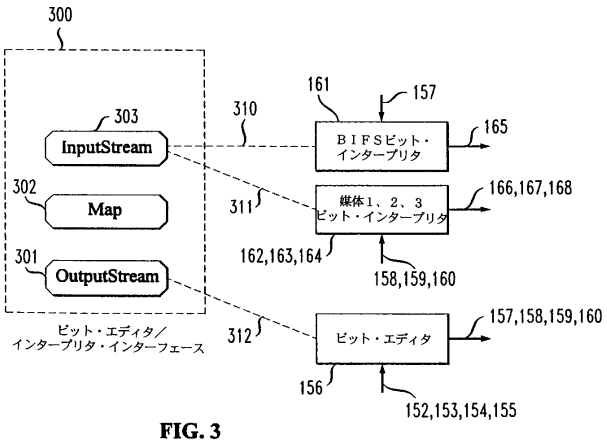

FIG. 3

【図４】

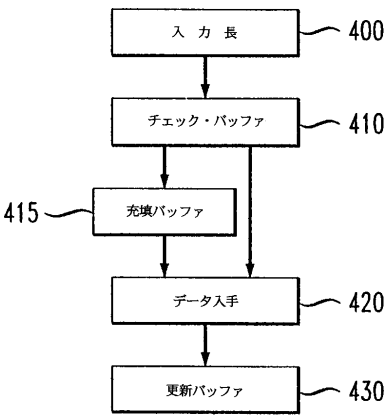

FIG. 4

フロントページの続き

(73)特許権者 507352341
ザ トラスティーズ オブ コロンビア ユニヴァーシティ イン ザ シティ オブ ニューヨーク
アメリカ合衆国 １００２７ ニューヨーク, ニューヨーク, ウエスト ワンハンドレッド シックスティーンス ストリート ５３５, ロウ メモリアル ライブラリー １１０
(74)代理人 100094112
弁理士 岡部 譲
(74)代理人 100064447
弁理士 岡部 正夫
(74)代理人 100085176
弁理士 加藤 伸晃
(74)代理人 100096943
弁理士 臼井 伸一
(74)代理人 100101498
弁理士 越智 隆夫
(74)代理人 100104352
弁理士 朝日 伸光
(72)発明者 エレフゼリアディス, アレキサンドロス
アメリカ合衆国 １００２７ ニューヨーク, ニューヨーク, アパートメント ４エル, リヴァーサイド ドライヴ ５６０
(72)発明者 ファング, イーハン
アメリカ合衆国 １００２５ ニューヨーク, ニューヨーク, アパートメント ２イー, ウエスト ワンハンドレッド ツェルヴス ストリート ５３９
(72)発明者 プリ, アチュル
アメリカ合衆国 １０４６３ ニューヨーク, リヴァーデール, ナンバー １エー, ワルド アヴェニュー ３６６０
(72)発明者 シュミット, ロバート エル.
アメリカ合衆国 ０７７３１ ニュージャーシィ, ホーウェル, オーク グレン ロード ３３３

合議体
審判長 乾 雅浩
審判官 梅本 達雄
審判官 渡邊 聡

(56)参考文献 Laier, J., et al, Content-Based Multimedia Data Access in Internet Video Communication, Proc. of 1st Workshop on Wireless Image/Video Communications, 1996年 9月, P. 126-133
Fang, Y., Eleftheriadis, A., Puri, A., and Schmidt, R., Adding Bitstream Input/Output Library in Java, ISO/IEC JTC1/SC29/WG11MPEG97/M2063, 1997年 4月 1日