マルチメディア通信と分散処理 76－11
（1996．5．16）

計算モデルに基づくマルチメディアデータの記述

只野 俊介† 布川 博士‡ 宮崎 正俊†

†東北大学大学院情報科学研究科
‡宮城教育大学理科教育研究施設

マルチメディアシステムを設計する際には、そのデータモデルとしてオブジェクト指向モデルが使われることが多い。しかし現行のオブジェクト指向モデルでは、大規模なマルチメディアシステムを記述するには不都合な点が幾つかある。本稿ではそれらの点について議論し、マルチメディアシステムを記述するのに適したメディアオブジェクトについて考察する。

本稿でモデル化するメディアとは、音、画像、テキスト等が、ある時間軸をもったシナリオに沿って並べられたもので、それぞれをオブジェクトとしてとらえ、そのメディア（＝オブジェクト）間の関係を効率よく記述していくための枠組について述べていく。

Description of Multimedia Data by Computation Model

Shunsuke TADANO† Hiroshi NUNOKAWA‡ Masatoshi MIYAZAKI†

†Graduate School of Information Science, Tohoku University
‡Research Institute for Science Education, Miyagi University of Education

When the multi media system is designed, the object oriented computation model is often used as the data model. However, there are some inconvenient points in the description of a large-scale multi media system in a present object oriented model. Then, we think about suitable media object for describing the multi media system in this paper.

Media modeled by this paper are the one that the sound, the image, and the text, etc. were arranged according to the time axis. Each of media caught as an object and we propose the frame to describe the relation for the media (= object) efficiently.

1 はじめに

現在、マルチメディアに対する関心が非常に高まってきている。これは、高速なプロセッサ、大容量かつ安価な一次、二次記憶デバイスの出現などのハードウェア分野のめざましい進歩に依るところが大きい。人に情報を伝える場合、単一の器官に訴えるよりも、複数の器官に訴えるほうがより適切に伝わる。よって情報サービスがマルチメディアの方向に動くのは当然である。

しかし、音声、動画などの、それぞれのメディアの単独での扱いは進んできているものの、更にもう一歩進んだ、異なるメディア間での統一的な記述、操作はまだまだ実現されているとはいえない。我々はこの問題をアプリケーションレベルではなく、アプリケーションが利用する計算モデルのレベルで解決するべきであると考えた。統一的なメディアの操作、表現のためには、まず、様々なメディアを表現することのできる包括的な計算モデルが必要である。

本稿では、メディアとして動画や音声といった時間的な性質をもつものを想定した。これらを同じ枠組で再生、停止、早送り等できるようにし、さらにそれらを組み合わせたマルチメディアデータも同様に操作できる環境を構築することを目的としている。この目的をアプリケーション側で実現しようとすると、個々のアプリケーション特有のものになりがちである。そこで、もっと下のレイヤーの計算モデルでこの目的を達成するために、Media Object 計算モデルを構築した。

本論文は 6 章からなる。2章ではまずモデル化するメディアを明確に示す。3章でそれを計算機上で実現するための計算モデル Media Object を定義したのちに、4章で Media Object の構造について述べる。5章では Media Object の効果的な使用例について述べ、6章でまとめとする。

2 モデル化するメディア

ここでモデル化するメディアとは、音、動画、静止画などが、ある時間軸に沿って並べられたものである。ここで、図 1の様にメディアの再生タイミングを記したものをシナリオと呼ぶものとする。再生のタイミングとしては、時間やユーザーがボタンを押すなどのインタラクションが用いられる。

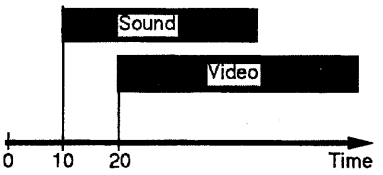

図 1: シナリオ

まずメディアを、単体で構成されているものと、複数で構成されているものの 2 種類に分ける。音声のみといった、単一の要素で構成されるメディアを *monoMedia* とする。図 1では Sound、Video が、それぞれ monoMedia である。また、メディアが時間的関連などの関係をもって集まったものを *compMedia* とする。これは、図 1のシナリオ全体を示すものである。さらに compMedia は他のメディアと組み合さって、新たな compMedia を構成することができるものとする。

現存するメディアで例をあげると、スピーカ等から流れてくる音楽は monoMedia である。映画は音声、BGM、映像といった monoMedia を組み合わせて作成した compMedia の例であるといえる。

3 Media Object

マルチメディアデータをモデル化する際には、オブジェクト指向計算モデルがよく使われる。これはそのモジュール性の高さから、将来扱うメディアが増えても、システムを設計する際のコストや期間を短縮することができるからである。

本稿でもこのオブジェクト指向の考え方を基に、2章で述べたメディアを計算機上で構築するために Media Object という計算モデルを提案する。

Media Object には次の 2 種類が存在する。

1. monoMObj: monoMedia をモデル化したもの。primitive な構成要素。
2. compMObj: compMedia をモデル化したもの。Media Object の集団。

MediaObject の主な特徴は次の通りである。

- 時間的同期を記述しやすくするために各 Media Object はタイマを持ちそれに同期して行動する。

- compMObj は Media Object の集団であると共にオブジェクトとしての性質を持つ。よって、Media Object 間のリンク情報であるシナリオを属性値として持つことができる。
- 編集性、再利用性を高める為に、時刻指定の際、ローカルタイムを用いることができる。

以下でこのモデルの詳細について述べる。

3.1 Media Object 間の協調の仕組み

Media Object 間の通信はメッセージを拡張した event を用いる。これは

(< 送信先 >, < コマンド >, < 実行条件 >)

からなるものである。実行条件では、Media Object の属性値が使われる。event を受け取った Media Object は、一旦その event をイベントプールに蓄える。そして自分でその実行条件をチェックし、条件が整った event から実行していく。

例として`MObjA` の属性値`flag` が`on` の時`play()` コマンドを実行したい場合を考える。この場合は以下の event を`MObjA` に送信すればよい。

```
(MObjA, play(), MObjA.flag = on)
```

もし`flag` の値が`on` でなかった場合はその event は実行されず、イベントプールがリセットされるまで蓄えられている。`MObjA` の属性値`flag` の値が`on` であったならば、`MObj` は`play()` コマンドを実行し、その event は消滅する。

また、通常のオブジェクト指向計算モデルではメッセージパッシングは一対一を想定しており、オブジェクトの集団を扱う概念がない。しかし、メディアをオブジェクトとしてモデル化する場合、マルチメディア = 複数のメディアであるので、オブジェクトの集団を扱う概念が必要である。

Media Object では一対一通信であるユニキャストの他に、一対多通信であるマルチキャストの機能を備えている。送信先に`M_` のタグがついている event を compMObj に送信すると、その event は compMObj に属している不特定多数の Media Object にマルチキャストされる (図 2)。

さらに compMObj は Media Object の集団であると同時に、オブジェクトとしても振舞うので、

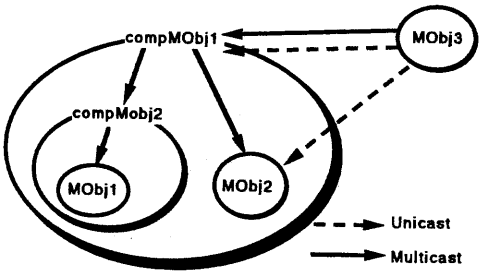

図 2: Media Object の通信形態

図 2のように、他の Media Object と新たな集団を階層的に形成することができる。よって、オブジェクトを意味的に分類し、集団を階層的に構成できるので、オブジェクト数の多い大規模なデータでも、自然にモデル化を行うことができる。

3.2 時間の扱い

従来のオブジェクト指向計算モデルはメッセージを受けてから行動を起こすという受け身の物であった。よって、音声や動画等といった時間依存のデータの再生などは、シナリオ通りの時間にメッセージを出す制御オブジェクトを用いて同期をとっていた。しかし、制御オブジェクトはその性質上その制御下のオブジェクトの状態値などを知る必要があり、これはオブジェクト指向の特徴であるモジュール性を損なうことになる。

そこで、Media Object ではそれぞれがタイマをもち、そのタイマに同期して自律的に行動できるものとする。タイマはデータの再生の開始とともにカウントをスタートし、再生の終了とともにカウントを停止し、その値をリセットする。タイマの役割をまとめると以下の通りである。

- Media Object の再生速度を決定する。
- Media Object の再生位置を決定する。
- Media Object の開始時間を決定する。
- Media Object の終了時間を決定する。

このタイマによって、例えば Picture Object に対して「10 秒間絵を表示しろ」といった命令を与えることができる。従来だと再生開始時とその 10 秒後の再生終了時に 2 度メッセージを送信しなけれ

ばならなかったが、Media Object では再生開始時にまとめて送信することができる。後は Media Object 自身が再生終了 event の実行条件をチェックし、自動的に再生を停止する。

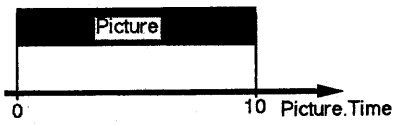

図 3: Picture Object のシナリオ

次に時間を扱う event の記述法について述べる。例として図 3のシナリオ通りに Picture Object を再生する場合を考える。このシナリオは、以下の event の集合で表現することができる。

```
(Picture, play())
(Picture, stop(), Picture.Time = 10)
```

これらの event は、アプリケーションや他の Media Object によって Picture Object に送信される。最初の event は実行条件がないので、これを受け取った Picture Object はすぐに`play()` コマンドを実行し、再生を開始する。それと同時に Picture Object のタイマはカウントを始める。次に 2 番目の event を受け取った Picture Object は、その event の実行条件として`Picture.Time = 10` とあるので、自分のタイマの監視を始め、その値が 10 になったときに`stop()` コマンドを実行して絵の表示を終了する。この様に、Media Object ではシナリオを記述してある event をいったん受け取ると、他のオブジェクトに指示されなくても、自分でその通りに行動することができる。

3.3 複数の Media Object を同期させる

複数の Media Object を同期させたい場合は、まずそれらの Media Object を含む compMObj を定義する。compMObj の属性値の一部は、その下の階層にいる Media Object が協調して行動する材料として公開される。つまり compMObj は協調型計算モデルでよく例題とされる黒板モデルの、黒板に相当するのである。

公開される属性値としては、その階層下の Media Object にとって共通の時間軸となる`Time`、ボタン等が押されたことを感知するためのフラグとなる変数`Button` 等がある。これらの値を基にシナリオを表す event の集合が作成され、その event の集合は compMedia の属性値 Scenario に格納される。

ここで Media Object の`play()` コマンドの働きについて整理する。まず monoMObj は、単一の要素で構成されるメディアをモデル化したもので、絵や音声等といったデータを属性値として持っている。この場合の`play()` コマンドの働きはそのオブジェクトが持っているデータを再生することである。Picture Object ならば絵を、Sound Object ならば音を、Video Object ならば動画を再生し`play()` コマンドの働きは終了する。

compMObj の場合はメディアがあるタイミングで並べられているシナリオをモデル化したもので、シナリオをその属性値として持っている。シナリオは event の集合であるので、`play()` コマンドを受け取った compMObj はその event をそれぞれの Media Object に送信するのである。その結果 compMObj に含まれている Media Object はシナリオ通りに再生されていく。

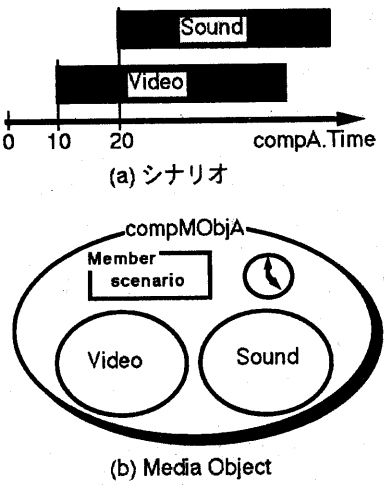

図 4: 複数の Media Object を同期させる

例として、図 4(a) のシナリオをモデル化する。この場合、まず Sound Object と Video Object が含まれる compMObjA を用意する (図 4(b))。そこで、Sound Object と Video Object の共通の時間軸として、CompMObjA のタイマが使用され

るのである。その値を利用して図4(a)のシナリオは以下のイベントの集合で表現される。

```
(Video, play(), compMObjA.Time = 10)
(Sound, play(), compMObjA.Time = 20)
```

このeventの集合がcompMObjAの属性値であるScenarioに格納されるのである。

compMObjAはplay()コマンドを受け取ると、これらのeventをそれぞれSound Object、Video Objectに送信し、その結果、2つのMedia Objectは図4(a)のシナリオの通りに再生されていく。

ここで重要なのはシナリオの情報をSound Objectが持っているのではなく、その協調の場であるcompMObjAが持っているということである。Sound Objectはこのシナリオに関する情報をいっさい持っておらず、再生開始時にcompMObjAからeventとしてシナリオの情報を受け取る。その為、このSound Objectを別のシナリオで再利用するといったことも容易に行えるようになる。

4 Media Objectの構造

Media Objectの一般的構造を図5に示す。

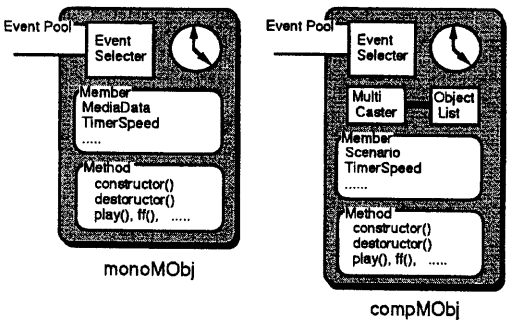

図5: Media Objectの構造

eventを受け取ったMedia Objectは、いったんそのeventをEvent Poolに蓄える。条件が整ったeventはEvent Selecterで撰択され、Event Poolから取り除かれた後に、実行される。

monoMObjは、再生すべきデータをその属性値MediaDataに格納している。compMObjは、その属性値Scenarioに再生すべきシナリオを格納している。

またcompMObjのMulti Casterはマルチキャストを実現するものである。そのcompMObjに属しているMedia ObjectをObject Listに登録して、M_のタグがついているeventを受け取った場合には、そのリストに基づいてマルチキャストを行う。

5 Media Objectの効果的な使用

3章のはじめでMedia Objectは時間指定の際、ローカルタイムを用いることができると述べた。これは、compMObjが階層的に構成されることによる。Media Objectのそれぞれがタイマを持っており、タイマは再生を開始するとカウントをスタートさせる。よって、上の階層から見ると下の階層のタイマは、そのMedia Objectが再生されてからの相対時間になるのである。

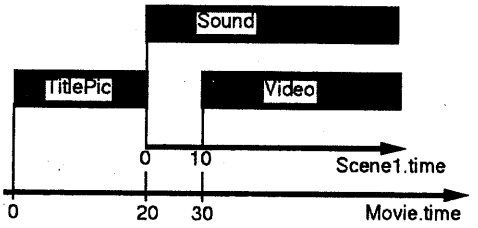

図6: Movieのシナリオ

ローカルタイムを利用した比較的長いシナリオの作成例として、映画をモデル化する場合を考える。このシナリオは、まず小さなシーンを作成しておき、それらを組み合わせることによって作られていく。ここでは図6のようなシナリオを想定する。これは音声と動画によるシーンが1つあり、タイトルを20秒表示してからそのシーンを再生するというものである。

これをMedia Objectで表現したのが図7である。まずSound ObjectとVideo Objectを収納するcompMObjであるScene1 Objectを作成する。次にこのScene1 ObjectとTitlePic Objectを含むMovie Objectを作成する。図の中で四角で囲まれた部分は各compMObjが持っているシナリオである。ここで、Scene1の中のVideoの再生タイミングを指定する際に、Movie.Timeの値ではなく、Scene1.Timeの値を使用できるのである。

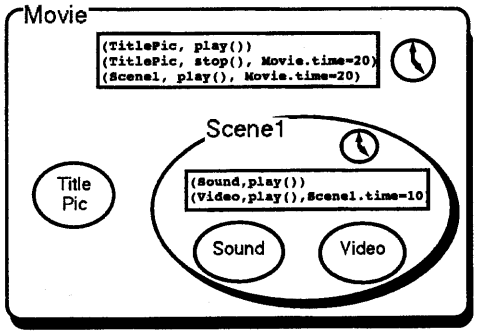

図 7: Movie Object

この為に、シーンごとの編集がより容易にできると考えられる。

6 おわりに

時間的な性質を持つマルチメディアデータを記述することのできる計算モデル Media Object を提案した。

この計算モデルは、event の実行条件を自分で判定することによって、自律的にタイマやユーザのインタラクションに同期して行動することができる。

また、協調して行動するために、その協調の場として compMObj を用意した。compMObj は階層的に構成でき、それによって event の実行条件において、ローカルタイムを使用できる様になった。その為、Media Object を、編集性、再利用性の高いものにすることができた。

現在、C++ による実装を進め、将来的にはオブジェクト指向データベースに Media Object を格納できるようにする予定である。

参考文献

[1] 只野 俊介, 布川 博士, 宮崎 正俊, マルチメディアシステムを記述するためのメディアオブジェクトの提案, 電気関係学会東北支部連合大会講演論文集, pp.203, 1995

[2] 藤川 和利, 下條 真司, 松浦 敏雄, 西尾 章治郎, 宮原 秀夫, オブジェクト指向に基づくハイパーメディアシステム Harmony の構築, 電子情報通信学会論文誌 D-1 Vol.J75-D-I No.11 pp.1015-1024, 1992

[3] 西尾 郁彦, 渡辺 豊英, 杉江 昇, オブジェクトと場に基づいた協調的プログラム言語, 情報処理学会論文誌, Vol.34, No.12, pp.2499-2508, 1993

[4] 芳賀 博英, 小嶋 弘行, 絹川 博之, マルチメディアシステムの記述のための自律オブジェクトの提案, Progress in Human Interfase, Vol.3, pp.45-52, 1994

[5] 武宮 博, 布川 博士, 野口 正一, 協調型計算モデルにおける field への自律性の導入, 日本ソフトウェア科学会第 8 回大会論文集, pp.61-64, 1991

[6] 大市 津義, 布川 博士, 宮崎 正俊, コミュニケーションのためのメディア記述言語の提案, 情報処理学会第 50 回全国大会予稿集, pp.6-165-166, 1995

[7] 亀山 渉, MHEG の最新動向, アドバンスド・データベースシステム・シンポジウム'93 講習会資料, pp.31-49, 1993