情 報 メ デ ィ ア 6－4
（1992．5．15）

インタラクティブアニメーション記述言語とその処理系「AV-Script」

阪本清美　浜田浩行　John Zurowski
松下電器産業株式会社　メディア研究所

従来のテレビやビデオで見られるような再生型のアニメーションではなく、ユーザの介在や、アニメーションの中の登場人物（キャスト）の性質や相互干渉によって、ストーリの展開やキャストの時空間上でのデータ属性を変化させることができる、インタラクティブなアニメーション創作のためのオブジェクト指向言語及びその処理系「AV-Script」について報告する。

本システムは、一つ一つのキャストをメディアに依存しない自律したオブジェクトとして定義できる。言語処理系は、ユーザとの対話機構、複数キャスト間の相互メッセージパッシング機構、キャストの速度制御機構、画像、テキスト、グラフィックス、ビデオや音などのメディアを管理するクラスライブラリで構成している。

Interactive Animation Scripting Language and its Environment 「AV-Script」

Kiyomi Sakamoto　Hiroyuki Hamada　John Zurowski
Media Research Laboratory,　Matsushita Electric Industrial CO., LTD.
1006, Kadoma, Kadoma-shi, Osaka　571, Japan

Traditionally television and video have only had straight-forward playback facilities. AV-Script introduces the concept of animated characters' that can facilitate communication between the user and devices. This interaction allows the user to change both the story line as well as individual cast member's characteristics. The following report will explain the object oriented management mechanisms of the AV-Script system.

AV-Script has the ability to define characters' behavioural characteristics independently from the media that they utilise . The language based animation system encloses the control mechanisms of user interaction, message processing, and velocity control within each cast member. These animated characters in turn are organised within a class hierarchy that can control graphics, text, video and audio media.

1　はじめに

アニメーションは、AVとコンピュータの両方の性質を兼ね備えており、テレビやVTRなどの再生型のメディアに勝る表現力の可能性もあることなど、非常に期待のできるメディアである。

また、近年のコンピュータ技術の発展に伴い、計算処理能力の向上やメモリ、２次記憶装置の低価格化が進み、ハードウェアによりリアルタイムにアニメーションを生成できる道が開けてきた。

本システムの目的は、アニメーション記述言語からのアプローチによるインタラクティブアニメーション創作システムを目指すことである。今回、インタラクティブにアニメーションを記述、実行できる言語及びその処理系「AV-Script」の開発を行ったので、その機能と実現方法について報告する。

2　アニメーション記述言語

アニメーション記述言語に関する研究は、１９８０年頃から「作り易さの追求」を中心に行われてきた。主な手法として、オブジェクト指向言語や、日本語記述などの自然言語からのアプローチが多い。

オブジェクト指向言語によるアプローチの代表的なものに、M.I.T.（米）で開発された「ASAS」[1]（１９８２年）やモントリオール大学(カナダ)で開発された「MIRA-3D」[2]（１９８３年）、カルフォルニア大学（米）で開発された「GRAMPS」[3]（１９８１年）などがある。

「ASAS」はActor/Scriptor Animation Systemの略称で、グラフィックスオブジェクトの振るまいと同様、アニメーションに時系列の流れも記述できる。アクターと呼ばれる構造を持ったオブジェクトは、並列かつ独立に動かすことが可能である。「ASAS」はLISP言語の拡張として開発されている。

「GRAMPS」はEvans and Sutherland社のMultiple-Picture-System上でFortran77をベースに開発された言語である。オブジェクト指向言語ではなく、コマンド形式とその実行環境で実装されている。

日本語記述などの自然言語によるアニメーション記述では、「ストーリ駆動型アニメーション」[4]や「EASY」[5]などがある。

しかし、これらのアニメーション記述言語を用いて作成されるアニメーションは最終的にはフィルムやＶＴＲに収録され、連続再生型のもので、実行時におけるインタラクティブ性を目指したものではない。

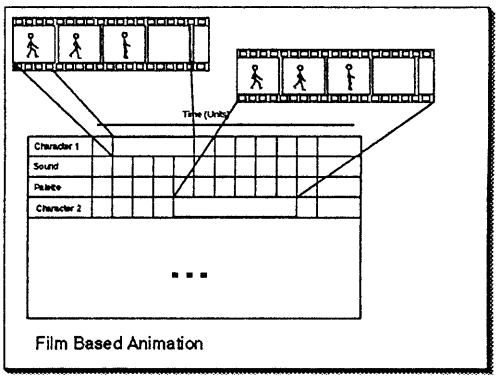
3　アニメーションモデル

アニメーションの記述性の良さを追求していくと、何をモデルに選ぶのかという点が非常に重要な問題となる。また、そのモデルの骨格によりインタラクティブ性も限定されてくる。ここでは、よく使用されているアニメーションモデルとそのモデルのインタラクティブへの適応性を検討する。

（１）フィルムモデル

殆どのアニメーションツールでは、Director[6]に代表されるように、セルアニメーションをモデル化したフィルムモデルが採用されている。

フィルムモデルの概念図を第１図に示す。



第１図

フィルムモデルでは、キャストを１つの連続したフィルムで表現する。アニメーションシナリオの作成は、複数のキャストをタイムライン上に配

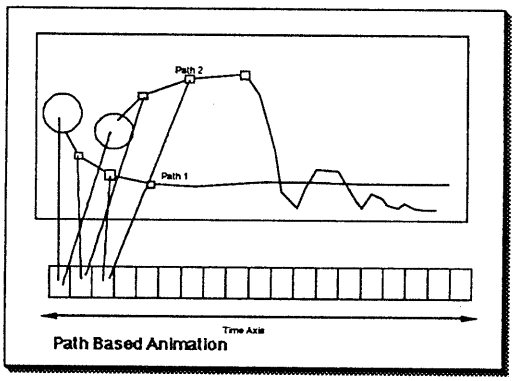
置することにより行われる。このモデルは非常に簡単で、セルアニメーション作成者にとっても分かりやすい。しかし、インタラクティブアニメーションのモデルとしては、次の問題点が考えられる。

1）インタラクティブな処理は、時間軸上の単純ブランチが基本で、それ以上の対話性を持たせることは非常に難しい。
2）一度作成したキャストの大きさ、移動速度、移動軌跡等を変更する場合、もう一度キャストに対応したフィルムを作成し直さなければならない。
3）シナリオの作成は、タイムラインにキャストを配置することにより行うことが基本なので、キャスト間の相互干渉を表現できない。
4）アニメーションの再生はフィルムの再生と同様であるから、時間制御は絶対時間のみで行われる。

（2）パスモデル

フィルムモデルの問題点を解決するために、最近ではAnimationWorks[7]にみられるように経路をベースにしたパスモデルを使用した例もある。

パスモデルの概念図を第2図に示す。



第2図

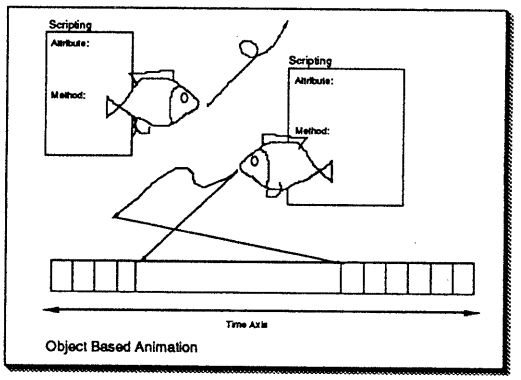
パスモデルでは、軌跡を基本に作成し、その軌跡の各制御点にキャストを写像することによりアニメーションを作成する。軌跡の各制御点は、時間軸の単位時間に対応しており、制御点間の距離で速度を表現する。キャストの軌跡や移動速度の変更が柔軟である点で、フィルムモデルより優れている。しかし、インタラクティブアニメーションのモデルとして考える場合、各制御点に対応したキャストを一つのセルフィルムとして考えている点で、フィルムモデルの延長であるため、次の問題点が考えられる。

1）インタラクティブな処理は、フィルムモデルに比べて一部分だけキャストを入れ換える、時間軸上の単純ブランチに加えてキャストの速度変更や軌跡変更が簡単であるが、それ以上の対話性を持たせることは非常に難しい。
2）フィルムモデルに比べてキャストの空間的配置は改善されたが、同一時間での複数のキャスト間の位置関係が分かりにくい。また、キャスト間の相互干渉を表現できない。
3）アニメーションの再生はフィルムの再生と同様であるから、時間制御は絶対時間軸のみで行われる。

（3）オブジェクトモデル

オブジェクトモデルは、一つ一つのキャストをオブジェクトとして定義し、そのキャストの時空間上での振舞い方を定義することによりアニメーションを作成する。

オブジェクトモデルの概念図を第3図に示す。



第3図

オブジェクトモデルは、キャストやシナリオの記述性が良いため、前述したように古くから再生型アニメーションの記述モデルとして採用されている。

オブジェクト指向の特徴であるカプセル化とメッセージパッシング機構は、インタラクティブ処理の記述性という点で他のモデルよりも適しているが、次のような問題点がある。

1）再生型アニメーションの言語処理系は、時間制御機構も含めてインタラクティブアニメーションにそのままでは適用できない。
2）インタラクティブな処理は、オブジェクトが受け取るメッセージに対する処理内容メソッドとして記述できるため他のモデルよりも簡単に定義できるが、オブジェクト間の相互干渉を記述する時に問題が生じる。
3）絵と音をインタラクティブな処理に合わせて合成する必要があるが、複数のメディアを考慮した枠組を持っていない。

4　AV-Scriptシステムの説明

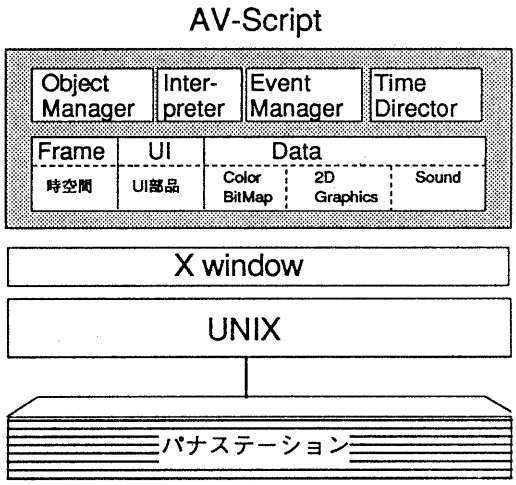
AV-Scriptは、オブジェクトモデルを基本にして構築している。オブジェクトモデルは、再生型アニメーションの記述言語では、古くから使用されている。本システムでは、テキスト、画像、音などの複合メディアを用いてアニメーションが簡単に記述できることを目的に、次のような構成にしている。

4.1　システムの構成

本システムのシステム構成図を第4図に示す。
ハードウェア：パナステーション
ソフトウェア：UNIX, X window, Motif上
開発言語　　：C++

AV-Scriptの言語処理系は、オブジェクトマネージャ、インタプリタ、イベントマネージャ及びタイムダイレクタからなる制御ソフトウェア層と、画像、グラフィックス及びユーザインタフェース部品のクラスライブラリからなるオブジェクトライブラリ層から構成されている。

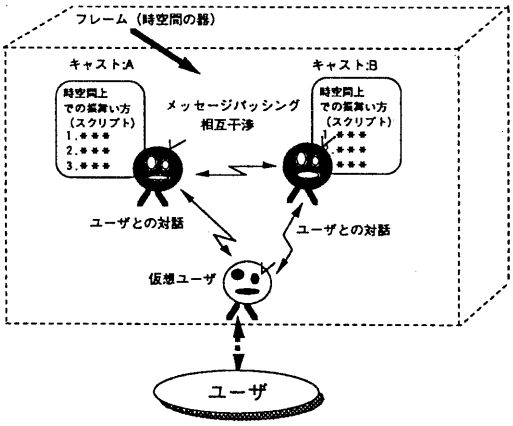
なお、言語処理系の詳細については、4.3章で説明する。



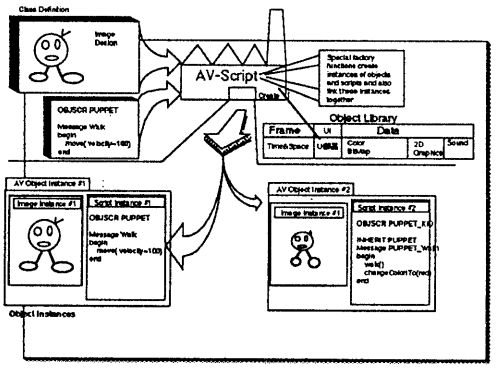
第4図

4.2　アニメーションモデル

本システムのアニメーションモデルの特徴は、一つ一つのキャストをメディアに依存しない自律したオブジェクトとして定義し、定義された複数のキャストをフレームという時空間の器に入れることにより、複数キャスト間のメッセージパッシングや相互干渉及びユーザとの対話を繰り返しながらアニメーションを展開してゆく、ロールプレーイング性のあるアニメーションが簡単に記述できることである。AV-Scriptにおけるアニメーションモデルを第5図に、アニメーションモデルが作成されるようすを第6図に示すとともに、AV-Scriptのメディア独立性とメディア記述性の特徴について説明する。



第5図



第6図

（1）メディア独立性

AV-Scriptでは、メディア独立性を保つために、次のような構成をとり、メディアに依存したデータオブジェクトを隠している。

キャスト＝データオブジェクト＋スクリプト

データオブジェクトの定義

素材データファイルを実体データとして持つデータクラスのインスタンスオブジェクト。

スクリプトの定義

スクリプトファイルを実体データとして持つスクリプトクラスのインスタンスオブジェクト。

キャストの定義

データオブジェクトとスクリプトを統合化した自律的なオブジェクト。

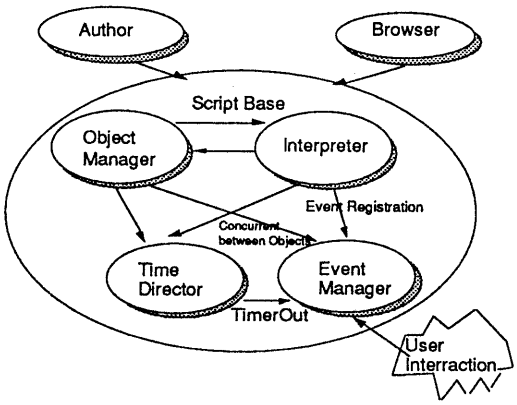
（2）メディアの記述性

画像、グラフィックス、テキスト等の素材データに関するクラスライブラリと、ボタン、メニュー、ダイアログボックス等のユーザインタフェースに関するクラスライブラリがシステムで用意されているので、第6図に示すように、メディアに依存した属性やメソッドはユーザが新たにスクリプトファイルの中で定義する必要がないため、アニメーションの記述が簡単になる。

4.3 言語処理系

インタラクティブアニメーション記述言語で、再生型アニメーションと最も異なる点は、出来上がったアニメーションを時系列に沿って実行するだけかどうかという点である。インタラクティブアニメーションでは、状況に応じてキャストの属性やシナリオ展開を変化させる必要があるため、その言語処理系は複雑なメカニズムが要求される。

そこで、本システムでは、言語処理系の処理を機能単位に分割することで単純化し、リアルタイム性を考慮に入れた機能拡張やハードウェア化、及びメディアの追加に迅速に対応できるシステムを構築することを目的としている。言語処理系の構成図を第7図に示すとともに、言語処理系の解釈実行できる機能及び特徴を以下に示す。



第7図

（1）言語処理系の解釈実行できる機能

1）インタラクティブ性

○ユーザとの対話性

○オブジェクト同士の相互干渉

○実行中に新しいキャストを登場させたり、刻々と変化する状況に応じて物語りの展開を柔軟に変化させることができる。

2）時間制御

○絶対／相対時間の制御

○異なるオブジェクトの同期制御

○オブジェクト別の速度制御

3）並列性

○オブジェクト個別に時空間上でのふるまいを定義し、複数のオブジェクトの並列実行制御。

（2）リアルタイム処理への機能拡張性

言語処理系が機能単位に分割されているため、リアルタイム性を考慮に入れた機能拡張やハードウェア化が行い易い。

（3）メディアの追加、拡張性

メディアに依存した部分はオブジェクトライブラリの中にカプセル化して開発されているため新しいメディアの追加、拡張性に優れている。

次に、言語処理系における各部品の機能の説明とその特徴を説明する。

4.3.1 オブジェクトマネージャ

通常のシステムにおけるオブジェクトマネージャの主な機能は、システム内に存在するすべてのオブジェクトを管理することである。

それに加えて、本システムでは、オブジェクト間の相互干渉を、オブジェクトマネージャの機能を拡張することにより実現している。以下にそのメカニズムを説明する。

1）検出したいオブジェクト間の相互干渉の種類（例えば「衝突」）と、それに関連したオブジェクトの名前をオブジェクトマネージャに登録する。

2）オブジェクトの状態変化が生じた時、そのオブジェクトはオブジェクトマネージャに知らせる。

3）オブジェクトマネージャは、状態変化を起こしたオブジェクトの名前と状態変化の種類が1）で登録されているか否か調べる。

4）登録されている場合、その相互干渉が成立するかを調べ、成立すれば、オブジェクトマネージャはイベントマネージャに対して相互干渉が成立したことを知らせるイベントを発行する。

4.3.2 インタプリタ

インタプリタは、オブジェクトに付与されたスクリプトを解釈し、並列に実行する。

現在、インタプリタの処理速度向上のため、スクリプトは中間言語に翻訳されてから実行される。

インタプリタの機能を以下に示す。

1）スクリプトの字句解析及び構文解析を行い、並列実行用のランタイムスタック及び中間言語を作成する。

2）四則演算、ループ制御及び変数値の代入等の実行を行う。

3）中間言語を解釈し、該当するオブジェクトに該当するメッセージを送信する。例えば、CreateTimerというメッセージが解釈されると、インタプリタはタイムダイレクタに、そのメッセージを送信する。この機能は、新しいメディアやメソッドの追加に対してもインタプリタの機能拡張を最小限にとどめている。

4.3.3 イベントマネージャ

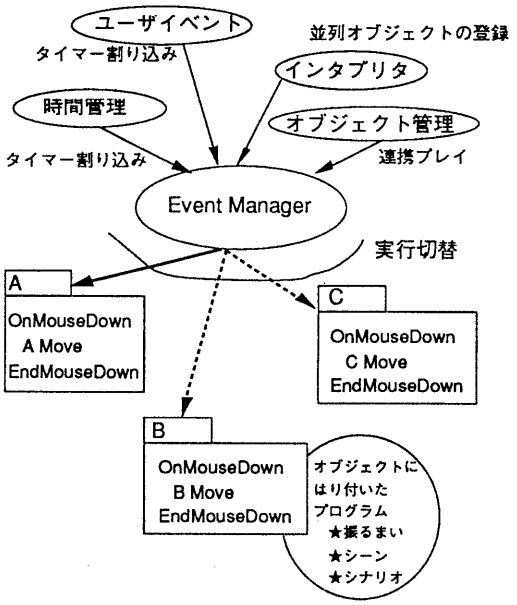
本システムが目指すような柔軟なインタラクティブ性を追求しようとすると、イベントマネージャの機能は非常に重要となる。

本システムにおけるイベントマネージャの機能を以下に示し、イベントマネージャの処理説明図を第8図に示す。

1）ユーザからの対話処理イベント、オブジェクトマネージャからのオブジェクト間の状態変化検出イベント、タイムダイレクタからのタイマー割り込みイベント等のすべてのイベントを一元的に管理する。

2）次に実行すべきイベントを選択する。

3）並列オブジェクトの実行切替やブラウジング／オーサリングモードの切替をする。



第8図

4.3.4 タイムダイレクタ

タイムダイレクタの機能は、システム全体の時間管理を行うことである。本システムでは、絶対時間だけでなく相対時間も扱えるように、次のような手法で時間管理を行っている。

1）相対時間軸の作成

時間軸に関するクラスオブジェクトであるタイムオブジェクトに、DefineTimeObjectというメッセージを送り、タイムオブジェクトのインスタンスを作成する。タイムオブジェクトの時間軸は、単位時間（タイムクウォンタム）に対する比率の値とオフセット値で表現されている。

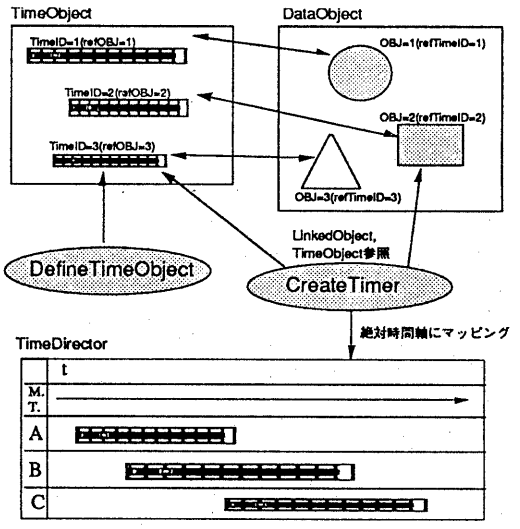
2）タイマーの作成

オブジェクトのスクリプトの中でCreate-Timerメッセージが発行されると、インタプリタは、そのメッセージをタイムダイレクタに送る。タイムダイレクタは、オブジェクトとタイムオブジェクトのリンクを作成するとともに、絶対時間軸にマッピングを行い、タイムダイレクタの管理下にある時間管理テーブルに登録する。

3）タイマー割り込みイベントの発行

タイムダイレクタは、そのテーブルを単位時間づつ進ませながら、タイマー割り込みを発生させる必要のあるタイマーを見つけたら、イベントマネージャに対してタイマ割り込みイベントを発生させる。

なお、タイムダイレクタ、タイムオブジェクト、及びデータオブジェクトの関係を第9図に示す。



第9図

5　プログラム及び実行例

プログラム例を第１０図に、その実行結果を第１１図に示す。なお、アニメーションに登場してくるキャスト（登場人物）をOBJECTSCRIPTで定義し、そのキャストがシナリオを展開して行くフレーム（時空間環境）をAVENVで定義している。

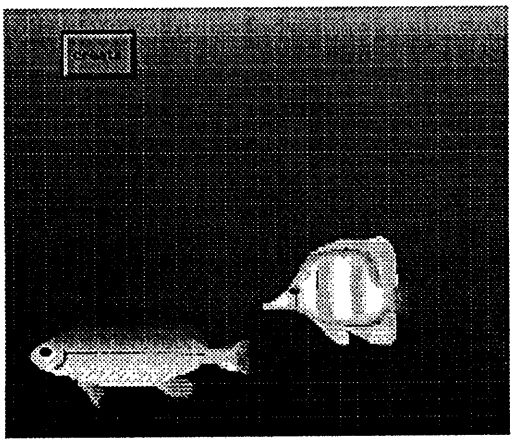
【キャスト用のスクリプト記述例】

```
OBJECTS
BEGIN
OBJECTSCRIPT        angel_scr
        MESSAGE init
        BEGIN
                setPosition( 330, 200 );
        END
        MESSAGE mouseDown
        BEGIN
                move( 330, 200, 0, 200, 50 );
        END
ENDSCRIPT
END
```

【シナリオ用のスクリプト記述例】

```
AVENV
BEGIN
 mainframe = getBackground;
 /* gets the animation background */
 umiName[0] = "umi1";
 aUmi = createAVObject( "umi", "S_ColorImage", mainframe, "umi_scr",
                umiName, 1 );
 createB = createAVObject( "create", "S_PButton", mainframe, "create_scr",
                0, 0 );
END
```

第１０図



第１１図

6　おわりに

本報告では、インタラクティブアニメーションに必要とされるアニメーションモデル、及びその言語処理系について述べてきた。

アニメーションモデルにおいては、キャストの記述が独立に定義できること、メディアに依存した部分を意識せずに使用できることなどを目標にして、オブジェクトモデルを拡張したモデルを提案した。

また、言語処理系においては、複雑な処理系を機能分散し、柔軟なインタラクティブ性、メディア拡張性、リアルタイム性やハードウェア化に迅速に対応できる原型開発を行った。

今後の課題として、オーサリング環境の開発やオブジェクトライブラリの機能の充実を図りたい。

【参考文献】

[1]C.W.Reynolds,"Computer Animation with Script and Actors," Computer Graphics(Proc.SIGGRAPH 82), July 1982,pp.289-296

[2]N.Magnenat-Thalmann and D.Thalmann,"The Use of 3D HighLevel Graphical Types in the MIRA Animation System," IEEE CG&A, Dec. 1983, pp.9-16

[3]T.J.O'Donnel and A.J.Olson, "GRAMPS-A Graphics Language Interpreter for Real-time,Interactive,Three-Dimensional Picture Editing and Animation," Computer Graphics (Proc. SIGGRAPH 81), Aug. 1981, pp. 133-142

[4]高島、島津、友納、"ストーリ駆動型アニメーションーシステム概要ー",情報処理学会第３３回、全国大会講演論文集、pp. 1303-1304, 1986, 10月

[5]花田、宮本、"イベント駆動型動画生成システム EASY",情報処理学会全国大会講演論集,Vol.41, No. 5, pp. 5.48-5.49, 1990

[6]MacroMind社の商標。

[7]Gold Disk社の商標。