ヒューマンインタフェース 52－5
（1994．1．13）

イベントマクロによるWindow Systemの高度化とその応用

奥村 晃弘　田川忠道　宮崎敏彦
沖電気工業(株)　関西総合研究所

我々は、エンドユーザがより快適にコンピュータの操作ができるようにするための、マルチモーダルインタフェースの研究を行なっている。その第一歩として、アプリケーションを変更せずにマルチモーダル化するツール「影武者」を試作した。「影武者」を使うことにより、Window System 上の一般のアプリケーションに、入力のマルチモーダル化、操作のマクロ化、ユーザインタフェースのカスタマイズなどの機能を付加できる。本稿ではこの「影武者」と、「影武者」を応用したプレゼンテーションシステム「もみじ」について述べる。

Augmenting a Window System with Event Macro

Akihiro OKUMURA　Tadamichi TAGAWA　Toshihiko MIYAZAKI
Oki Electric Industry Co.,Ltd.
Crystal Tower 2-27 Shiromi 1-chome,Chuo-ku,Osaka 540,JAPAN

We have been studying a multimodal interface for naive users to operate computers comfortably. We developed "KAGEMUSHA" which can add customization mechanisms of user interfaces to Window application programs. "KAGEMUSHA" provides the following facilities :(1)adding multimodal interface (2)creating a macro by example (3) customizing user interface by user original menu system. This paper describes "KAGEMUSHA" and it's application "MOMIJI".

1 はじめに

マンマシンインタフェースの研究は、計算機が持つ複雑な機能を誰もが簡単に使いこなせることが、全てのアプリケーションの重要なテーマであることもあって、かなり広い分野にわたって様々な研究がなされている。

計算機の簡単な利用を阻害している要因をマンマシンインタフェースの観点から考えた場合以下のようなものがあげられる。

1. 操作の不統一性
2. 操作手順の繁雑さとカスタマイズの難しさ
3. 操作チャネルの制限

1のためにはGUIの統一などが言われているが、比較的表面上の統一であり細かな操作の設計は開発者に任されていると言う点で十分ではない。

1と3を同時に解決する手段として、人間が連想しやすい言葉を使って音声で操作するという研究も盛んに行なわれている [1][2][3]。

2に対する研究の1つの方向としては、繰り返し操作のマクロ化 [4] があげられる。

3に関しては、いわゆるマルチモーダルインタフェースの研究として活発化しており、先の音声認識技術の利用や画像理解技術との融合 [5], データグローブ等の利用 [6] などがある。

ところでマンマシンインタフェースの研究では、改良のアイディアだけでなく、多くのユーザの経験を基にした実践的な検討と改良が重要である。従って、開発したシステムが一般のユーザに受け入れられる必要があろう。インタフェースを改良したシステムを試作する際に、そのインタフェースの特性を最大限生かすためにアプリケーションを新規に開発する場合があるが、多くの場合開発者だけの試用に終り一般に受け入れられないことが多い。

そこで我々は研究の取り掛かりとして、既存のアプリケーションのインタフェースを変更するとともに、上記のような機能を付加できるようなツール(我々はこのツールを「影武者」と呼んでいる)を開発することとした。

アプリケーション側のプログラムを変更せずにインタフェースを向上させようとする場合、ツール側ではアプリケーションの全ての内部状態を知ることができず、状態に応じた細かな制御ができないという欠点がある反面、先に述べたようにユーザが現在利用しているツールをそのまま使えるという連続性の点で優れている。

以下、第2章で人間の作業効率を高めるためのツールについて、我々の考えについて述べ、第3章で試作したツール「影武者」について、第4章で「影武者」の動作記述言語について、第5章でプレゼンテーションシステムへの応用について述べる。

2 人間の作業効率を高めるためのツール

人間の作業効率を高めるためには、人間ができる限り知的な作業に専念できるようにする必要がある。今回これに関して考察した結果、以下の3点をユーザインタフェース変更ツールの基本機能とすることにした。

入力のマルチモーダル化 入力メディアを新しいものに変更することにより、作業効率を高めることが可能であるのなら、誰もが利用したいと思うのは当然である。ところが、従来はアプリケーションがその入力メディアに対応していないと使うことができなかった。従って、アプリケーションに依存することなく新しい入力メディアをそのまま接続できる機能が必要である。

操作のマクロ化 単純な繰り返し作業は機械に行なわせるほうが、作業効率を高めることができる。アプリケーションによっては、操作のマクロ化機能を備えているものもあるが、使用方法がアプリケーションによって異なっているという問題がある。そこで、すべてのアプリケーションの操作を同じ方法によりマクロ化可能とするほか、複数のアプリケーションに跨るマクロ化や、操作の一部のパラメータ化を可能とする機能が必要である。

ユーザインタフェースのカスタマイズ 作業効率を高めるためには、使いやすいユーザインタフェースが必要である。ところが、どのようなユーザインタフェースが使いやすいかは、各ユーザによって、また、同じユーザであっても経験や慣れによって異なってくるものである。従って、各ユーザが各自のコンピュータ操作の上達や、アプリケーションの使い方の習熟に合わせて、

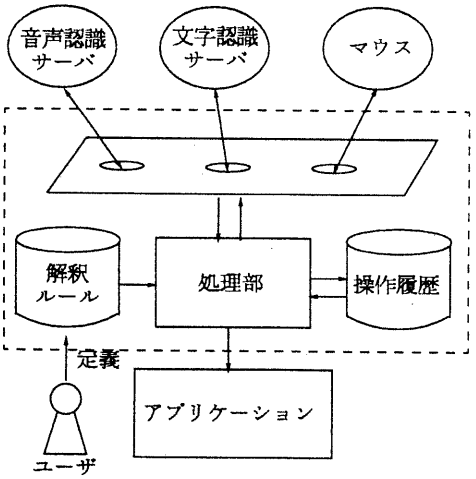

図 1: ユーザインターフェース変更ツールの概念図

簡単に自分の使いやすいユーザインタフェースを構築できる機能が必要である。

例えば、自分が使う機能だけを集めたユーザ独自のメニューを定義できるようにし、さらに、新しい機能を覚えた場合には、すぐその場でそのメニューに新しい項目を加えることができるようにすれば使いやすい。

これらの機能を実現するために、図1に示す枠組を考案した。

図の中で、破線で囲まれた部分がツールとして開発すべき部分である。このユーザインタフェース変更ツールは、各種入力メディア(図では音声認識、文字認識、マウス)とアプリケーションの仲介を行なう。つまり、各種入力メディアからの入力情報を受け取り、解釈ルールに従って必要な入力メディアの情報を参照して処理し、アプリケーションへ処理結果を渡すようにする。この機構を用いることにより、使いたい入力メディアを単に接続することができるようになるだけでなく、解釈ルールの記述によって、複数の入力メディアの協調および統合を行なうことができるようになる。

今回、この枠組を実現するための第1歩として、「影武者」を試作した。今回の試作では、マウスとキーボード以外の入力メディアとして音声認識装置を扱うことにした。

3 ユーザインタフェース変更ツール「影武者」

「影武者」は、X-Window System †上に構築した、ユーザインタフェース変更ツールである。イベント駆動形式で記述されている X-Window System 用アプリケーションに対して、イベントの横取りと送出を行なうことにより、そのユーザインタフェースを変更することができる。本章では、図2を用いて、「影武者」の構成と各機能の実現方法について解説する。

3.1 「影武者」の構成

「影武者」は、透明ウインドウと影武者本体とで構成される（図2）。

X-Window System では、複数のウインドウがツリー構造を成している。ここで、あるウインドウの下位に位置するウインドウをそのウインドウの子ウインドウと呼ぶことにする。

「影武者」は、対象とするウインドウとその全ての子ウインドウに対して、同一サイズの透明ウインドウを作成し、対応するウインドウの真上に重ねるように配置する（図3）。また、透明ウインドウに対して、もとのウインドウをアプリケーションウインドウと呼ぶことにする。透明ウインドウが常に対応するアプリケーションウインドウの真上に存在するようにすることにより、ユーザのアプリケーションに対する入力イベントを横取りする。

影武者本体は、定義ファイルの記述に従って、横取りしたキーボード、および、マウスのイベントや、新しい入力メディアからの入力を処理する。処理の詳細については、後述する。

3.2 各機能の実現方法

入力のマルチモーダル化 新しい入力メディアから入力を受け取った影武者本体が、その入力内容に対応するイベント列をアプリケーションウインドウに送ることにより実現する。新しい入力メディアからの入力内容とイベント列の対応づけは、定義ファイルを用いてユーザが行なうことができる。

†X-Window System は MIT の商標です

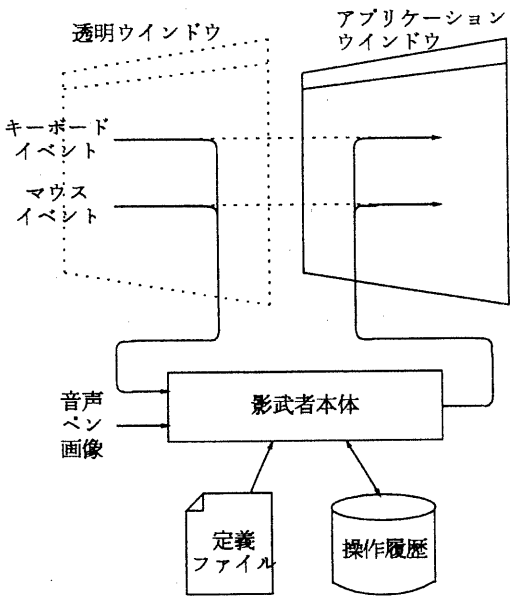

図 2: 「影武者」の構成

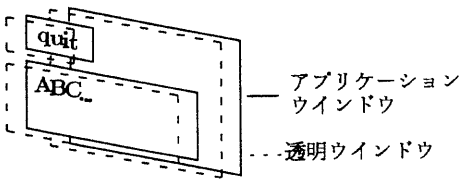

図 3: 透明ウインドウの配置

操作のマクロ化 操作のマクロ化は、透明ウインドウによって、横取りしたキーボードやマウスのイベントの内容を、アプリケーションウインドウに送ると同時に操作履歴に記録することによって行なう。

横取りしたイベントの種類が、ButtonPress, ButtonRelease, KeyPress, KeyRelease, MotionNotify のいずれかであれば、次のデータを取り出す。

- イベントの種類
- イベントの発生時刻
- イベントが発生したウインドウ
- イベント発生時のマウスカーソルの位置
- イベント発生時の修飾キーの状態
- 各イベントに固有なデータ

そしてこの内、イベントの発生時刻とイベント発生ウインドウは、それぞれイベントの発生間隔とウインドウ ID に変換して記録する。イベントの発生間隔は、現在のイベントと一つ前のイベントとの発生時刻の間隔である。また、ウインドウ ID は、アプリケーション内のウインドウのツリー構造に従って割り振るコードであり、透明ウインドウを張り付ける際に、透明ウインドウ、および、アプリケーションウインドウとウインドウ ID の対応を決定する。このウインドウ ID を使うことにより、同じ種類のアプリケーションにおいて、マクロを共有することができる。

また、マクロの実行は、操作履歴のイベントのデータから、イベントを再構築しアプリケーションウインドウに送ることによって実現している。この際に、ウインドウ ID より対応するアプリケーションウインドウを決定し、イベントの発生間隔でイベントの送出間隔を調節する。

また、MotionNotify を記録イベントから外して記録データを減らすこともできる。この場合は、イベント再構築時にイベントの発生間隔とマウスカーソルの位置を使って、適当なイベントを幾つか作成して補間することにより、マウスカーソルが滑らかに移動するようにしている。

複数のアプリケーションに跨るマクロは、後述する joint ステートメントを使って行なう。

また、パラメータ化に関しては始点のパラメータ化のみを扱った。マクロごとの設定で、始点のパラメータ化を選択すると、マクロの実行時に始点を指定することができる。始点を指定することにより、始点のあったウインドウに対するイベントに限り、イベント再構築時のマウスカーソルの座標を相対的に移動させるようにした。これにより、マクロをある程度一般化することができる。

```
application:kterm
application:console
application:InterViews drawing editor

start_state:root
 コンソール[こんそーる]:center(console);
 ターミナル[たーみなる]:raise(kterm);
 マクロ開始[まくろかいし]:rec_start();
 マクロ終了[まくろしゅうりょう]:rec_end();
end_state:

start_state:KTerm
include_state:root
 クリア[くりあ]:macro(clear);
 リスト[りすと]:macro(clear);macro(ls);
end_state:

start_state:idraw
include_state:root
 クリア[くりあ]:macro(all_delete);
end_state:
```

図 4: 動作記述言語の記述例

ユーザインタフェースのカスタマイズ 影武者本体が、必要に応じて、メニューやメッセージを表示することによって実現する。今回のインプリメントでは、操作のマクロ化の終了時にコマンド名と発音データを入力することにより、自動的にメニューと音声認識辞書への登録を行なうようにした。また、ユーザからの入力イベントを、アプリケーションウインドウへ送らないようにすることによって、ユーザからの入力を制限することなどもできる。

4 「影武者」の動作記述言語

前章でも述べたように、「影武者」の動作は、定義ファイルで定義する。この章では、この定義ファイルを記述する動作記述言語を通して、「影武者」の各種動作について解説する。

動作記述言語の記述例を図4に示す。

application: で始まる行で「影武者」が対象とするアプリケーション名を定義する。「影武者」は実行開始時に、指定されたアプリケーションのウインドウに透明ウインドウを張り付ける。また、手裏剣ボタンを表示して、そのアプリケーションが「影武者」の影響下にあることを示す。この手裏剣ボタンをクリックすると、そのアプリケーション用の影武者操作メニューを表示する。

start_state と end_state で挟まれたブロックはステートの定義である。ステートはアプリケーションの状態を示し、各ステートは「影武者」が受け付けるコマンドの集合を保有する。「影武者」の影響下にあるアプリケーションは、それぞれが、ある一つのステートを値として持ち、マウスカーソルが入っているアプリケーションのステートの値が「影武者」の現在のステートの値となる。但し、マウスカーソルが「影武者」の影響下にあるアプリケーション上に無い場合は、ステートは root という特殊な値になる。また、各アプリケーションはステートの初期値として、そのアプリケーションのクラス名を取る。但し、そのステートが定義されていなかった場合は、root を値として取る。ステートを変更する方法としては、上述したマウスカーソルを移動させる他に以下の二つがある。

- ステート変更のステートメントを実行する
- メニューでステート変更を選択する

どちらも、対象アプリケーションのステートを変更する。

このクラスに基づくステートの導入により、以下の効果がある。

- 同種のアプリケーションに対してコマンドを共通して使うことができる。
- 違う種類のアプリケーション間でコマンド名が衝突しない。

コマンド定義は、

コマンド名 [発音データ]: ステートメント列

という形で記述する。これにより、コマンド名と音声認識の結果、および、「影武者」の動作の対応付けを行なう。include_state: は、定義中のステートは指定ステートの全てのコマンドを含むという意味である。

なお、今回利用した音声認識装置は、認識対象の単語あるいは文をあらかじめ定義しておく方式であ

表 1: 動作記述言語の主なステートメント

ステートメント	引数	機能
center	ウインドウ名	ウインドウを画面の中央へ移動
bigcenter	ウインドウ名	ウインドウを拡大して画面の中央へ移動
unmap	ウインドウ名	ウインウインドウを非表示にする
raise	ウインドウ名	ウインドウを最前面に移動
pointer	ウインドウ名	ウインドウの中央にポインタを移動
default	なし	全てのウインドウを初期状態に戻す
macro	マクロ名	マクロを実行する
joint	ウインドウ名 コマンド名	コマンドを実行する
rec_start	なし	マクロの定義を開始する
rec_end	なし	マクロの定義を終了する
cstate	ステート名	ステートを変更する
push_cstate	ステート名	現在のステートをスタックに積んでからステートを変更する
pop_cstate	なし	スタックからステートを取り出してそのステートに変更する
keypress	キーシンボル	キー入力のイベントを送る
menu	メニュー	ユーザ定義メニューの表示
confirm	なし	コマンド実行の確認を行なう

るため、ステートに合わせて認識辞書を切替えて認識対象語を絞り込むことによって、誤認識を減らすことができる。

このステートメント列に記述可能なステートメントを表 1に示す。

joint は、複数のアプリケーションに跨るマクロを実行するためのステートメントである。joint ステートメントを記述する場合には、引数にウインドウ名とコマンド名を指定する。joint ステートメントが実行されると、これらを用いて指定ウインドウにマウスカーソルを移動した後に、指定コマンドを実行する。この際に、通常行なわれるマウスカーソルを移動したときのステートの変更も行なわれる。現在のシステムでは、複数のアプリケーションに跨る操作を一度に記録する仕組みがないので、このように joint ステートメントでコマンドを連結することにより、この機能を実現している。

その他のステートメントに関する説明は紙面の都合上省略する。

5 プレゼンテーションシステムへの応用

「影武者」に後述する幾つかの機能を追加することにより、マルチメディア添削システム「もみじ」を試作した。

「もみじ」は既存アプリケーションに文字や図形，音声を張り付けて、説明手順を記録することができ、さらに、記録したデータを電子メールを使って被添削者に送り、被添削者側であたかも添削者がその場で指導しているかのように、添削内容を再現することができる一種のプレゼンテーションシステムである。

このため、「もみじ」では「影武者」に次の機能を追加した。

コメントの編集機能 手書き曲線，図形，テキスト文字列などのコメントを既存アプリケーション上に張り付けることができる。また、それらの編集を行なうこともできる。

コメントの操作の記録 / 再生機能 アプリケーションの操作と同時に、コメントに対するアクションを記録／再生することができる。

サンプリング音声の録音 / 再生機能 アプリケーションの操作を記録すると同時に音声をサンプリングし、操作の再生に合わせて音声も再生することができる。

添削者の作業は、「コメント作成」と「解説収録」の二つのフェーズに分かれる。これは、通常の紙と鉛筆を用いた添削の場合の、原稿を赤ペンで訂正した後に、口頭で詳しい説明を本人の前で行なうことに対応する。

コメント作成フェーズでは、以下のようなのコメントの作成、および、編集を行なう。

手書き曲線 マウスなどのポインティングデバイスを使って線画を描く。手書き曲線のアイコンをクリックすると線画を再描画する。また、clear ステートメントを実行することにより、線画を消去できる。

図形 任意のビットマップを張り付ける。ビットマップをシェイプすることにより、アプリケーションの元の表示との重ね合わせを実現している。

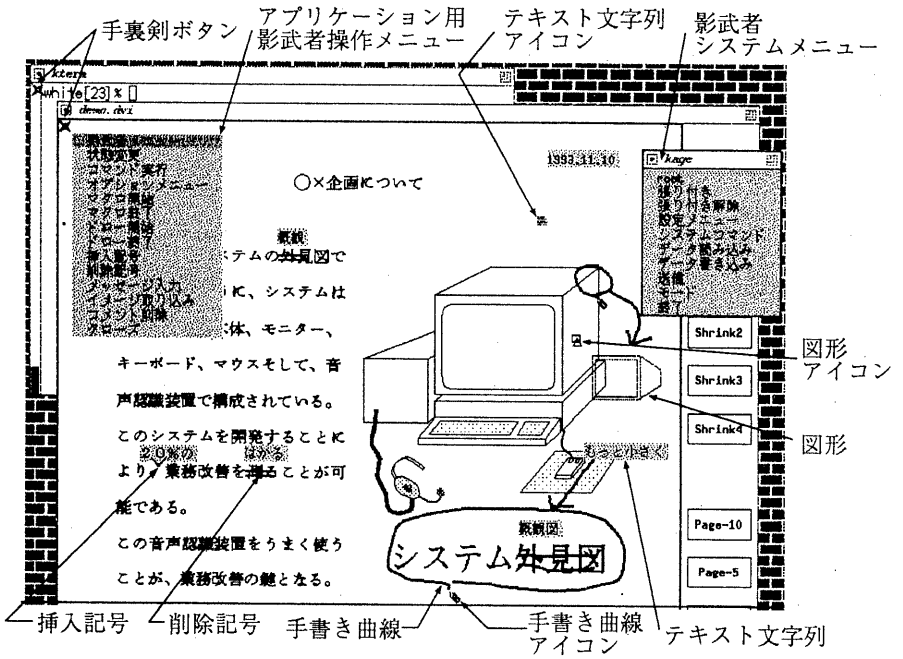

図 5: プレビューア (xdvi) で表示した文書を添削した例

画面上のイメージを切りとって使うことができるので、一般の絵を書くツールで作成した図形を使うこともできる。図形のアイコンをクリックすると表示／非表示が切り替わる。

テキスト文字列 テキストの入力と編集ができる。クリックでアイコン／非アイコンが切り替わる。

テキスト文字列のアイコン表示は、テキスト文字列の表示が無いときだけであるが、手書き曲線、および、テキスト文字列の場合は線画などクリックしにくい場合があるので、アイコンを常に表示している。各コメントのアイコンとアクションを図 6にまとめる。

解説収録のフェーズでは、マウスなどのポインティングデバイスと音声を使って修正理由や意図の説明の収録を行なう。ポインティングデバイスで重要箇所を指し示しながら、また、アイコン化してあったコメントを順番に表示していきながら解説することにより、解説を受ける人はどこに注目すべきがよく分かるようになる。この機能は、「影武者」の操作のマクロ化機能を拡張することにより実現した。

表 2: 「もみじ」で追加したステートメント

ステートメント	引数	機能
voice	ファイル名	サンプリング音声再生
display	コメント名	コメントの表示
undisplay	コメント名	コメントの非表示
flush	コメント名	コメントの点滅
move	コメント名	コメントの移動 (アプリケーション内)
free_move	コメント名	コメントの移動
draw	ファイル名	手書き曲線の描画
clear	なし	手書き曲線の消去
sync	なし	音声の再生が終了するまで待つ

解説収録の開始合図はメニューから選ぶか、もしくは、定義ファイルを図 4のように記述することにより、「マクロ開始」と言うことにより実行できる。また、解説収録中であることはマウスカーソルの形状が変化することで確認できる。各状態でのマ

コメントの種類	アイコン	アクション
手書き曲線		再描画
図形		表示／非表示
テキスト文字列		アイコン／非アイコン

図 6: 各コメントのアイコン

状態	マウスカーソル
手書き曲線の入力中	
解説収録中	
解説再生中	

図 7: 各状態のマウスカーソル

ウスカーソルの形状を図7にまとめる。

さらに、コメントの操作を記述できるように表2に示すステートメントを動作記述言語に追加した。これらの動作記述言語を使えば、実際には操作を行なわずに解説を組み立てることもできる。また、記録した幾つかのマクロを組み合わせて解説を作成することができる。

このようにして作成した解説は、定義ファイルと各種データファイルとで構成される。各種データファイルは音声データを含むこともあり、比較的サイズが大きい。このことから、今回の試作ではデータファイルは、ネットワーク共有ファイルシステムを使って共有することとし、定義ファイルのみを添削依頼者に電子メールで送ることとした。添削依頼者は、送られて来た定義ファイルを「もみじ」で読み込むことにより、解説を再生することができる。

「もみじ」を使うことは、以下の2点において有効である。

- 既存のアプリケーションをそのまま添削システムとして利用できる。
- 複数のアプリケーションを連動させた解説を行なうことができる。

最後に、「もみじ」による添削例を図5に示す。

6 おわりに

既存のアプリケーションのユーザインタフェースを変更するツールである「影武者」とその応用である「もみじ」について述べた。

「影武者」を使って従来から使用してきたアプリケーションに音声入力装置を接続することによって、その長所および短所を実感することができた。今後は、試用を通した客観的な評価、ならびに、パラメータ化を含めたマクロ化機能の強化、さらには、入力メディアの協調およびその動作を定義するための動作記述言語の拡張を行なっていく予定である。

参考文献

[1] 竹林洋一 ほか: “不特定ユーザを対象とした音声対話システムの試作”, 人工知能学会 第1回言語・音声理解と対話研究会資料 (SIG-SLUD-9201), pp.27-36(1992-01)

[2] C.Schmandt, M.S.Ackerman, and D Hindus: “Augmenting a Window System with Speech Input”, IEEE COMPUTER, Vol.23, No.8, pp.50-58(1990)

[3] 中谷吉久 ほか: “文書編集における音声制御の一方式”, 情報処理学会論文誌 Vol.33, No.2, pp.195-203(1992-02)

[4] B.A.Myers: “Demonstrational Interfaces: A Step Beyond Direct Manipulation”, IEEE COMPUTER, Vol.25, No.8, pp.61-73(1992-08)

[5] 秋藤俊介 ほか: “画像処理を用いた操作入力インタフェース”, 情報処理学会研究報告, Vol.93, No.100, pp.15-21(1993-11)

[6] 渡辺茂晃 ほか: “コンピュータ・デスクトップへの身振り対話の導入”, 計測自動制御学会 ヒューマン・インタフェース研究論文集 Vol.2, No.1 pp.65-72(1993-11)