

05-R003

# ビジュアルインタフェースの 研究開発報告書

平成 6 年 3 月



財団法人 日本情報処理開発協会

この報告書は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて平成5年度に実施した「ビジュアルインタフェースの研究開発」の成果の一部をとりまとめたものがあります。





## はじめに

コンピュータシステムの利用者層の拡大とともに、人間とコンピュータとの間の相互作業（インタラクション）を扱うユーザインタフェースの改善が極めて重要になっております。

従来のユーザインタフェースは、コマンドなどの論理記号を用いるほか、グラフィカルユーザインタフェース（GUI）に見られるような2次元表示技術を用いた方法が主流でした。このため、人間がより理解しやすいインタフェースとして、グラフィックス、イメージ、アニメーション、3次元といった高度な技術を用い、人間とコンピュータとの複雑な相互作業を視覚化したビジュアルインタフェースが提案されております。例えば、データベース検索の場合は、従来、検索コマンドの論理的な組合せによって行われてきましたが、3次元グラフィックスとアニメーション技術を用いてデータベースの構造そのものを視覚化することにより、全体構造の表示、注目点へのズームインなどが可能となり、視覚思考でデータを検索できるようになります。

こうした背景のもとに、当協会では、平成5年度より2年計画で、「ビジュアルインタフェースの研究開発」事業を実施することといたしました。

実施に当たりましては、大学、企業などの研究者、専門家からなる「ビジュアルインタフェースに関する調査研究ワーキンググループ（主査 田中二郎、筑波大学電子・情報工学系助教授）」を設け、ワークスペースの視覚化、データの視覚化、視覚化プログラミングなどの技術動向、ビジュアルインタフェースの研究事例、新しいインタフェースの技術的な課題等について調査検討を行うほか、平成5年8月には国際会議に海外調査員を派遣し、最新技術情報の収集を図りました。

また、プロトタイプシステムを作成してビジュアルインタフェースの技術評価を行うこととし、コンセプトの検討、実現環境の整備などを行いました。

本報告書は、平成5年度における研究開発成果の一部を中間報告として取りまとめたものです。第1章はビジュアルインタフェースに対する問題提起、第2章は技術動向、第3章は内外の研究開発事例の構成となっております。

最後に、本研究開発に当たって、ご指導ご協力を頂いた、田中主査を始め各委員の方々及び関係各位に深甚なる謝意を表する次第です。

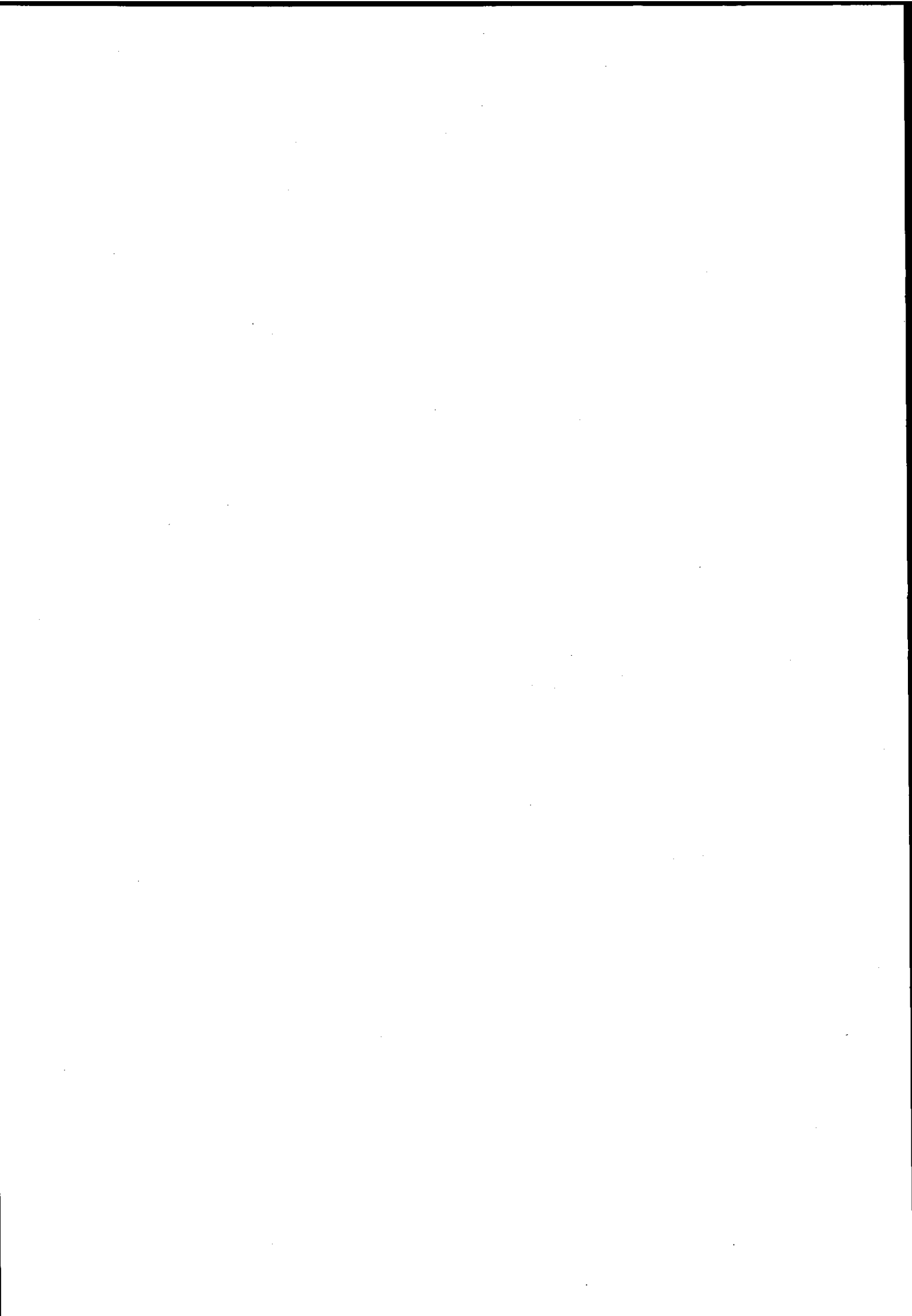
平成6年3月

財団法人日本情報処理開発協会



「ビジュアルインタフェースに関する調査研究ワーキンググループ」メンバー名簿（敬称略）

|        |                         |                                          |
|--------|-------------------------|------------------------------------------|
| 主 査    | 田中 二郎                   | 筑波大学 電子・情報工学系 助教授                        |
| 委 員    | 朝日 宣雄                   | 三菱電機（株） パーソナル情報機器開発研究所<br>開発第三課 第2グループ   |
| 委 員    | 岡田 幹夫                   | 東京電力（株） システム研究所 制御研究室 主管研究員              |
| 委 員    | 小池 英樹                   | 電気通信大学 電子情報学科 助手                         |
| 委 員    | 小島 啓二                   | （株）日立製作所 システム開発研究所<br>第5部503研究ユニット 主任研究員 |
| 委 員    | 斉藤 康己                   | 日本電信電話（株） NTT基礎研究所 主幹研究員                 |
| 委 員    | 竹内 彰一                   | （株）ソニー コンピュータサイエンス研究所 シニアリサーチャー          |
| 委 員    | 西川 博昭                   | 筑波大学 電子・情報工学系 助教授                        |
| 委 員    | 萩谷 昌巳                   | 東京大学 理学部 情報科学科 助教授                       |
| 委 員    | 服部 桂                    | 朝日新聞社 編集局 科学部                            |
| 委 員    | 平川 正人                   | 広島大学 工学部 第二類（電気系） 情報システム 助教授             |
| 委 員    | 増井 俊之                   | シャープ（株） 技術本部ソフトウェア研究所 係長                 |
| 委 員    | 松岡 聡                    | 東京大学 工学部 計数工学科 講師                        |
| 委 員    | 安村 通晃                   | 慶應義塾大学 環境情報学部 助教授                        |
| 委 員    | 暦本 純一                   | 日本電気（株） ソフトウェア生産技術開発研究所 主任               |
| 委 員    | 向山 博                    | （財）日本情報処理開発協会 開発研究室 主任研究員                |
| 事務局    | 妙泉 正隆                   | （財）日本情報処理開発協会 開発研究室 主任部員                 |
| オブザーバー | Jean-paul Smets-Solanes | （財）日本情報処理開発協会 開発研究室 客員研究員                |



# 目 次

## はじめに

|                                              |    |
|----------------------------------------------|----|
| 1. ビジュアルインタフェースについて .....                    | 1  |
| 1. 1 ポストGUIの時代 一次世代インタラクションスタイルのめざすものー ..... | 1  |
| 1.1.1 はじめに .....                             | 1  |
| 1.1.2 GUIシステムの限界 .....                       | 2  |
| 1.1.3 次世代インタラクションスタイル .....                  | 5  |
| 1.1.4 関連研究 .....                             | 7  |
| 1.1.5 おわりに .....                             | 13 |
| 1. 2 情報パースペクティブの変化とビジュアルインターフェース .....       | 15 |
| 1.2.1 最近のビジュアルインターフェースの傾向 .....              | 15 |
| 1.2.2 ビジュアルインターフェースの何が問題か? .....             | 15 |
| 1. 3 インタフェースとインタラクション .....                  | 18 |
| 1. 4 ビジュアルインタフェース研究の落とし穴 .....               | 23 |
| 1.4.1 VI研究のポイント .....                        | 23 |
| 1.4.2 聴覚化 .....                              | 27 |
| 1. 5 ビジュアルの次に来るもの .....                      | 30 |
| 1.5.1 はじめに .....                             | 30 |
| 1.5.2 ビジュアルの良さ .....                         | 30 |
| 1.5.3 ビジュアルインターフェイスの問題点 .....                | 31 |
| 1.5.4 マルチメディアは答になるか? .....                   | 32 |
| 1.5.5 人のコミュニケーションでは .....                    | 33 |
| 1.5.6 実世界との関わり .....                         | 35 |

|                                           |           |
|-------------------------------------------|-----------|
| 1.5.7 おわりに                                | 35        |
| <b>2. ビジュアルインタフェースの技術動向</b>               | <b>37</b> |
| 2. 1 実世界指向インタフェースの研究動向                    | 37        |
| 2.1.1 はじめに                                | 37        |
| 2.1.2 GUI関連研究の動向                          | 37        |
| 2.1.3 マルチメディア/マルチモーダル                     | 39        |
| 2.1.4 エージェント                              | 39        |
| 2.1.5 仮想現実感                               | 40        |
| 2.1.6 実世界指向インタフェース                        | 41        |
| 2.1.7 おわりに                                | 43        |
| 2. 2 3次元を用いたビジュアルインタフェースの研究動向             | 49        |
| 2.2.1 はじめに                                | 49        |
| 2.2.2 提示技術                                | 49        |
| 2.2.3 対話技法                                | 53        |
| 2.2.4 開発環境                                | 60        |
| 2.2.5 アプリケーション                            | 64        |
| 2.2.6 おわりに                                | 65        |
| 2. 3 3次元視覚化の文献紹介                          | 69        |
| 2.3.1 インフォメーションビジュアライザ                    | 70        |
| 2.3.2 ナレッジワーカのための3次元インタフェース               | 74        |
| 2.3.3 バーチャルスペースでの表現                       | 79        |
| 2.3.4 プログラム実行の3次元表現                       | 82        |
| 2.3.5 3次元グラフィックスを利用したインタラクティブなモジュール間コード解析 | 86        |
| 2.3.6 3次元コンピュテーションビジュアライゼーション             | 89        |
| 2.3.7 抽象データの3次元ビジュアライゼーション                | 93        |
| 2.3.8 3次元ダイレクトマニピュレーションの設計                | 98        |
| 2.3.9 容易な3次元ダイレクトマニピュレーション                | 103       |

|       |                                |     |
|-------|--------------------------------|-----|
| 2. 4  | インタフェースビルダの動向                  | 107 |
| 2.4.1 | インタフェースビルダとは                   | 107 |
| 2.4.2 | 実例                             | 107 |
| 2.4.3 | インタフェースビルダの利点                  | 111 |
| 2.4.4 | 問題点                            | 112 |
| 2.4.5 | 問題点の解決方法の摸索                    | 115 |
| 2.4.6 | 結論                             | 117 |
| 3.    | ビジュアルインタフェースの事例                | 119 |
| 3. 1  | ソフトウェア開発環境への3次元視覚化の応用          | 119 |
| 3.1.1 | はじめに                           | 119 |
| 3.1.2 | VOGUEにおける3次元視覚化の枠組             | 119 |
| 3.1.3 | VOGUEの特徴                       | 121 |
| 3.1.4 | 応用例                            | 123 |
| 3.1.5 | おわりに                           | 129 |
| 3. 2  | 電力におけるビジュアルインタフェースの応用          | 131 |
| 3.2.1 | ハードウェア環境                       | 132 |
| 3.2.2 | 実世界モデルへの応用                     | 132 |
| 3.2.3 | 実世界モデル応用での課題と今後の計画             | 134 |
| 3.2.4 | イメージ世界モデルへの応用                  | 134 |
| 3. 3  | ビデオ映像部品を利用したリアリティユーザインタフェースの試み | 143 |
| 3.3.1 | はじめに                           | 143 |
| 3.3.2 | 従来のユーザインタフェースにおけるビデオ映像         | 143 |
| 3.3.3 | インタラクション機能を持つビデオ映像             | 144 |
| 3.3.4 | 実装                             | 148 |
| 3.3.5 | おわりに                           | 150 |
| 3. 4  | 3次元GUI構築ツール                    | 152 |
| 3.4.1 | 仮想操作環境                         | 152 |

|       |                      |     |
|-------|----------------------|-----|
| 3.4.2 | 実現方式                 | 152 |
| 3.4.3 | 仮想操作環境の具体例           | 157 |
| 3.4.4 | まとめ                  | 160 |
| 3. 5  | アイコンックデータベース操作システム   | 163 |
| 3.5.1 | 研究背景                 | 163 |
| 3.5.2 | データベース操作の枠組み         | 163 |
| 3.5.3 | 基本演算                 | 166 |
| 3.5.4 | データベース操作例            | 169 |
| 3.5.5 | システム構成               | 174 |
| 3. 6  | ビジュアル・プログラミング・システムPP | 176 |
| 3.6.1 | 既存のビジュアルシステムの問題点     | 176 |
| 3.6.2 | 並列論理型言語の可読性と図形的表現    | 177 |
| 3.6.3 | 並列論理型言語のビジュアルシステムの機能 | 178 |
| 3.6.4 | 並列論理型言語のビジュアルシステムPP  | 179 |
| 3.6.5 | ビジュアルシステムPPの動作イメージ   | 179 |
| 3.6.6 | 関連研究の動向              | 186 |
| 3.6.7 | まとめ                  | 186 |
| 3. 7  | 視覚的仕様記述環境            | 188 |
| 3.7.1 | はじめに                 | 188 |
| 3.7.2 | 視覚的プログラミング環境の現状と課題   | 188 |
| 3.7.3 | よりユーザフレンドリな環境を目指して   | 189 |
| 3.7.4 | 仕様記述の再利用性を向上させるために   | 192 |
| 3.7.5 | おわりに                 | 194 |

## 1. ビジュアルインタフェースについて



## 1. ビジュアルインタフェースについて

### 1.1 ポストGUIの時代 一次世代インタラクションスタイルのめざすものー

コンピュータの小型化によって、携帯できるコンピュータの実現が可能になった。しかし、現状ではそのユーザインタフェースはデスクトップコンピュータのGUIを単に持ち込んだものにすぎず、携帯化によるコンピュータの利用形態の変化に対応していない。その原因の一つは、携帯性によってコンピュータを使う場面が飛躍的に拡大しているにもかかわらず、コンピュータがそういった状況から得られる情報をうまく活用できていないことによる。ここでは、利用者の状況を認識できる装着型コンピュータの構成方法を提案し、コンピュータとの新しいインタラクションスタイルについて考察する。

#### 1.1.1 はじめに

過去20年以上にわたって、バッチ式であったコンピュータはTSS、さらにはPC、ワークステーションと進化を遂げた。コンピュータを操作する手段も、キャラクタやコマンドを中心としたCUI (Character User Interface) から、現在ではデスクトップメタファ、ビットマップディスプレイ、ダイレクトマニピュレーション、WIMP (Windows, Icons, Menus, Pointing Device)、そしてWYSIWYG (What You See Is What You Get) などの概念で象徴される、いわゆるGUI (Graphical User Interface) がビジュアルインタフェースの主流になってきた。コンピュータの利用が一般化するに従って、ヒューマンコンピュータインタラクション (HCI) は今後ますます重要な分野になっていくだろう。今後のコンピュータの機能向上は、そのほとんどがHCIを充実させるために使われるであろう、という予測もある。

一方、コンピュータの利用形態はさらに拡大しつつある。そして、デスクトップコンピュータ向けに開発されていたGUIの技術では適応できない領域も出始めている。例えばGUI型のインタフェースは机上のコンピュータと対面して利用することを前提としているが、実際の人間のアクティビティは机上だけには限定されていない。現時点で製品化されている「ノート型」のコンピュータは、単にデスクトップコンピュータの小型版にすぎず、常に身につけて利用するコンピュータを実現しているとは言えない。例えば立ったままでノートコンピュータを利用することは困難である。

次世代のHCIでは、小型化・移動型といったコンピュータの利用形態の変化に基づくHCIの質の変化を考慮する必要がある。現在の携帯電話や手帳のように常に持ち歩いて利用するようになったコンピュータでは、必然的に現在の机上のコンピュータとは異なるインタフェースが要求される。ペンや音声などの新しい入出力手段を取り込んだインタフェースの実用化が、一部では既に始まっている。より長期的には、例えばウェアラブルコンピュータ (衣服のように身にまとえるコンピュータ) とい

ったコンセプトが提案されている。また、ユーザが発するあいまいな指令を、その場の状況や過去の履歴から判断するなど、より知的で融通の効くHCIの開発も望まれている。処理指向から通信指向へのシフト、操作指向からガイダンス指向へのシフトなど、HCIの前提となる目的自体が変化してきている（表1.1-1）。このシフトはGUIが誕生した約15年前の動きに匹敵する、HCIの大きな変革を引き起こすものであると考えられる。

表1.1-1 GUIからポストGUIへのシフト

|           | GUI       | ポストGUI      |
|-----------|-----------|-------------|
| 主目的       | 処理中心      | 情報中心        |
| コンピュータの役割 | コマンドの実行   | アシスタンス      |
| ユーザの注意    | コンピュータの操作 | 現実世界の作業     |
| 利用時間      | 短い        | 長い          |
| コントロール    | ユーザによる    | コンピュータによる   |
| 指令の構文     | オブジェクト・動作 | なし(あるいは1単語) |
| 対話手段      | ある瞬間では単一  | 同時に複数       |
| 対話の推移     | 相互に相手を待つ  | 常に並行している    |
| 対話の場所     | スクリーンに限定  | 現実世界に偏在     |

ここでは、このような、いわば「ポストGUIの時代」が到来しつつあるという現状認識に基づき、次世代のヒューマンインタフェースにおいて課題となる項目を議論する。関連する他の研究としてはモバイルコンピューティング、PDA、ペンベースコンピュータなどがあるが、ここでの視点は多少異なり、それらの要素技術を統合する上での中心となる考え方、これらの技術を前提としたときのコンピュータと人間の関係がどうなるか、について議論する。

### 1.1.2 GUIシステムの限界

ハードウェアの進歩によって、コンピュータの形態は大きく変化した。特にサイズの変化が著しい。部屋を占領するメインフレームから、デスクトップ型、ノート型、さらには手帳型と進化してきている。しかし、その一方でソフトウェアやヒューマンインタフェースが、小型化されたコンピュータに対応できていないのも事実である。携帯しての利用を前提とした手帳型コンピュータには、デスクトップコンピュータを想定して開発されたGUIはそぐわない。

小型化へのシフトは、コンピュータを利用する局面が従来に増して飛躍的に拡大することも意味している。今まではコンピュータの利用とは、「コンピュータのある所へ行く」ことを意味していた。したがって、コンピュータを利用するときのユーザの置かれている状況はいつも同じ（例えば机に向かっている）と仮定しても差し支えなかった。しかし、今後は日常生活のあらゆる局面に「コンピュ

ータが持ち込まれる」傾向が増大する。この主客逆転がHCIの設計方針に大きく影響を及ぼすだろう。つまり、机に向かって使うコンピュータではなく、会議中、通勤中、ビル内の移動中などに使うコンピュータが一般化するわけである。コンピュータの利用をオフィス内に限ったとしても、会議室でプレゼンテーションをしているときに必要な機能と、ファックスの前にいるときに必要な機能は同じではない。このように、ユーザの置かれている状況の変化に伴い、コンピュータに求められる機能は刻々と変化していく。しかし、現状のコンピュータが持っているインタラクションスタイルはその変化を考慮して設計されてはいない。以下で、現状のGUIシステムの限界について考察してみる。

#### (1) デスクトップの束縛

現状では、机上のコンピュータと対面している時にのみコンピュータの機能を利用できる。しかし、実際には在席のままでできる作業は現実の業務のうちのごく一部にすぎない。現状では、情報が必要な場所や作業が発生した場所で、すぐにコンピュータの能力を利用することが困難である。常に「仕事を自分の机に持ち帰る」ことを要求される。すなわち、従来のコンピュータは「コンピュータのあるところに行って利用する」という発想から抜け出せていない。

また、デスクトップコンピュータのインタフェースは、「ユーザが常にコンピュータに対面して、その注意をすべてコンピュータのスクリーンに注いでいる」という仮定の基に設計されている。コンピュータのハードウェアが小型化されて携帯可能になったとしても、そのインタフェースが従来型のインタラクションスタイルを継承している限り、真にその機能をユーザのために役立てることはできない。ワークステーションが机上のメインフレームコンピュータではなかったのと同様、携帯型コンピュータは手の平に載るデスクトップコンピュータではないのである。

#### (2) 現実世界とコンピュータ世界のギャップ

一般に、従来のコンピュータシステムでは現実世界のアクティビティとコンピュータ世界のアクティビティは一致していないし、その間で情報を自由に行き来させることが困難であった。これが多くのフラストレーションを生む原因になっている（例えば「データベース上は存在する貨物が倉庫に見あたらない」など）。

一方、すべてのアクティビティをコンピュータ内の世界で完結させようとする試みも不成功に終わっている。例えば、従来考えられていた「ペーパーレスオフィス」は結局実現しなかったし、今後も印刷物が消えることはないだろう。紙にはコンピュータで置き換えることのできない種々の利点がある。実際にはコンピュータの普及に伴い、使用される紙の量は増え続けているの現状である。

つまり、すべての処理を電子世界内に完結させよう、という発想は誤りであり、現実世界と電子世界という二つの異質な世界間の流通を良くする技術に注意を向ける必要がある。例えば、コンピュー

タからのプリントアウトは、もはやコンピュータが管理できるオブジェクトではない。したがって、ハードコピーとコンピュータ内のデータを二重管理するという問題が現実には起きている。

### (3) ユーザの状況の認識

映画の字幕は、ストーリーの進展や俳優の台詞に同期して表示されてこそ意味があり、例えば上映の前に一括して表示されても役には立たない。このように、同じ情報でも提示されるタイミングや状況によってその有効性は大きく変化する。従来のGUIでは、ユーザの置かれている状況をコンピュータが認識して、適切な対応をするという発想があまり見られなかった。デスクトップコンピュータは「机上で作業している」という限定された状況でしかユーザと対面しないので、そもそもこの問題が認識されていなかったともいえる。

しかし、コンピュータが携帯型になると、ユーザがどこにいて何をしようとしているのか、を認識することが重要になってくる。例えば自動車のナビゲーションシステムでは、「現在どこを走っているか」をシステムが知っているかどうか非常に重要なポイントとなる。会議に参加しているユーザは、その状況に即した情報、例えば「前回の会議の資料」が容易にアクセスできるようなコンピュータを望むであろう。どちらの例でも、ユーザが現在位置をコンピュータにインプットしたり、会議名をインプットして資料にアクセスすることは可能であるが、ユーザに不必要な操作を強いるため、システムの使い易さが阻害されてしまう。

### (4) 煩雑な操作

現状のGUIシステムでは、コンピュータの機能を利用するためには、コンピュータに対して何らかの指令を与えなければならない。指令がキーボードからのコマンドであるか、マウスによるものかを問わず、少なくともユーザは

- いまどんな情報、機能を必要としているか
- それにはどんな指令を入力する必要があるのか

を判断しながらコンピュータを利用しなければならない。古典的なコマンド指向インタフェースと比較して、GUIでは操作やその結果をビジュアルにすることでインタフェースの向上を図ってきた。しかし操作そのものが必要である点では、コマンド指向インタフェースから本質的には進歩していないとも言える。実際、GUIは操作の際の認知的負荷を軽減しているのであって、操作そのものを簡易化しているのではない点に注意すべきである。Nielsenはこういったインタラクションをexplicit dialogと総称し、バッチ指向からGUIに至るまでのコンピュータとユーザとの対話スタイルを支配してきた原理であるとしている[Nielsen93]。

デスクトップコンピュータと対面して、ユーザの注意をすべてコンピュータに注ぐような状況では、

操作の強要は大きな問題とはならなかった。しかし携帯型のコンピュータでは、移動中や会議で発言をしている最中など、「他の作業をしているときに」コンピュータの能力を利用する必要がある。その場合、もしユーザの置かれている状況をコンピュータが判断できるのなら、本質的でない操作は省略できるはずである。逆に、煩雑な操作をそのまま携帯型のコンピュータに持ち込むと、コンピュータの携帯が物理的に可能になっても、実際には利用する時間や場面が限られてしまう、という結果に終わるだろう。

例えば、ある人のオフィスを訪問して、その人が不在だったとしよう。そして、相手にメッセージを残すために、その場で（オフィスのドアの前で）携帯型コンピュータを使って電子メールを送信しようと思ったとする。もしコンピュータが

ー 今いる場所は△△氏のオフィスの前である

ー △△氏は現在不在である

といった状況を認識していれば、コンピュータに対して単に「伝言」という指令を与えるだけで、誰宛で何の理由での伝言なのかをコンピュータが推測して補足できるはずである。一方、状況認識を行わないコンピュータを使うと、相手の電子メールアドレスを指令として明示的に与えなければならず、操作が煩雑なものになってしまう。

### 1.1.3 次世代インタラクションスタイル

このように考えていくと、現状のデスクトップで有効な概念でも、携帯型のコンピュータでは必ずしも有効ではないことが分かる。例えばデスクトップメタファ、マルチウィンドウ、WYSIWYG、WIMP、ダイレクトマニピュレーション、といった現状のGUIスタイルの中心となる概念も、携帯型コンピュータではさして重要ではない。これは、一つには携帯型コンピュータの小さなスクリーンではデスクトップで有効な対話技法が通用しないということがある。しかし、より深い要因として、ユーザの置かれている状況を判断すること、ユーザの意図を推測し、余計な操作を不要にすること、などデスクトップではあまり注目されてこなかった対話手法への重要性がポストGUI時代には増していくことが挙げられる。

それでは次世代（ポストGUI）のHCIには何が求められるのだろうか。「現実の状況を認識し、それにコンピュータの情報を付加することでユーザの環境をより理解しやすくするようなコンピュータと人間の対話スタイル」であると考える。例えば、補聴器や眼鏡は現実の情報を処理し、（ハンディキャップを持った）人間により適した情報として提供している点に価値がある。同様に、次世代インタラクションに基づくシステムでは、オフィスや工場など、現実の環境を飛び交う情報を整理し、ユーザにとって理解しやすい方法で提供するであろう。こういったインタラクションスタイルは、以下のような概念で特徴づけられるだろう。

- 装着性： いつも身に付けて利用できる。極端な場合、利用していることを忘れてしまう。  
外見は、現在のページャ（ポケベル）、補聴器、携帯型ラジオ、ウォークマン、眼鏡といった機器に近いものになるだろう。
- 状況の認識： ユーザが現実世界のなかでどのような状況に置かれているかを認識する。状況としては、場所、時刻、ユーザの発話、操作の履歴、近くにある物品などが考えられる。
- 暗黙の対話： 状況認識によって、不必要なコマンドを与えることなく、ユーザが必要とする情報を提示できる。これは、別の言い方をすれば「状況を入力とする」一種のマルチモーダルインタフェースである。

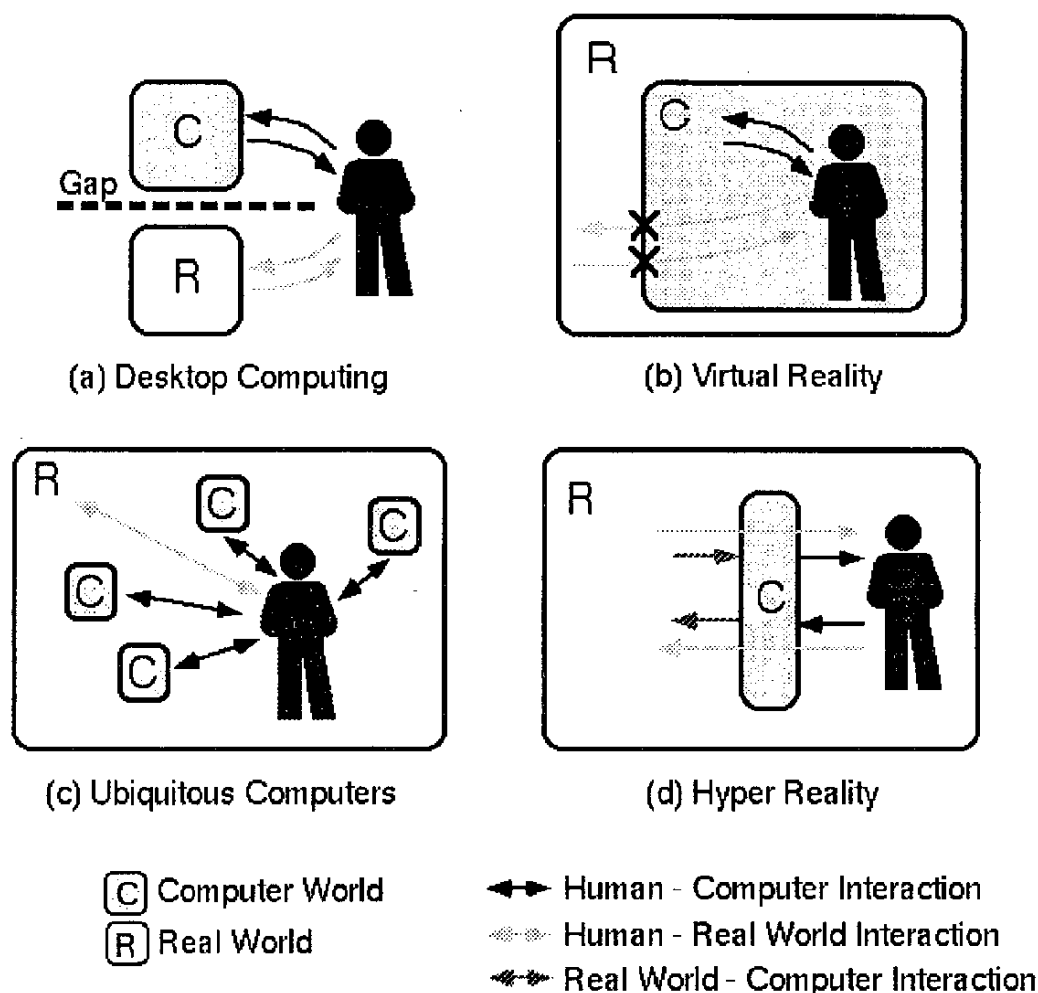


図1.1-1 インタラクションスタイルの比較

図1.1-1は、人間とコンピュータ間の交流における種々のスタイルを比較したものである。図の(d)が、ここで考えている次世代インタラクションに相当する。

- (a) 従来のデスクトップ型（GUIベース）コンピュータでは、ユーザはコンピュータディスプレイと対面し、キーボードやマウスを介して対話する。一方、ユーザは現実世界とも交流しているが、この二つのインタラクションは融合されておらず、いわばギャップが生じている。
- (b) 仮想現実感（Virtual Reality）に基づくインタラクションスタイルでは、ユーザの交流範囲は完全にコンピュータの作り出す世界に限定され、現実世界とのインタラクションは遮断されてしまう。
- (c) ユビキタスコンピューティング（Ubiquitous Computing）によるインタラクションでは、コンピュータは現実世界に埋め込まれており、コンピュータとの交流と現実世界との交流は混然一体のものになっている。
- (d) 仮にHyper Realityと名付けたインタラクションスタイルで、装着型コンピュータに基づいている。このスタイルでは、ユーザは現実世界と交流するが、そこにコンピュータが付加的な情報を付与し、より現実世界を理解しやすいものにする。またコンピュータはユーザと現実世界との交流、ユーザの置かれている状況を理解し、適切な対応をとる。このアプローチはユビキタスコンピューティングに似ているが、ユビキタスコンピューティングが人間とコンピュータを世界の対等な構成物と考えているのに対して、Hyper Realityではコンピュータを人間の能力を補強するための（眼鏡や補聴器といった器具に近い）概念として位置付けている点が異なる。

#### 1.1.4 関連研究

ここでは、ポストGUIに関連する研究開発の動向について説明する。

##### (1) 航行めがね（NaviGlasses）

マッキンゼリジェンズの大座畑による次世代コンピュータの提案[ozawa90]である。眼鏡状のコンピュータを利用し、眺めている物に関する情報が視野にスーパーインポーズされる。日本国内の雑誌、研究会等に何回か関連する記事、論文が発表されている。ただし技術的実現性はほとんど考慮されておらず、コンセプト提案の域を出ていない。

##### (2) NECのウェアラブルターミナル

日本電気デザインセンタ（現 NECデザイン）のデザイナを中心に、未来のコンピュータを考えるコンセプトワークがなされた。その結果提案されたコンセプトが「ウェアラブルターミナル」である。コンセプトワークの過程で数種類のモックアップが製作された（図1.1-2）。

ただし現状ではコンセプト提案以上の検討、技術的な裏付けなどは行われていない。またここで取り上げている「状況に依存した情報提示」や「現実とコンピュータ情報の融合」など、コンピュータと人間のインタラクションの進化についての検討は行われなかったようである。

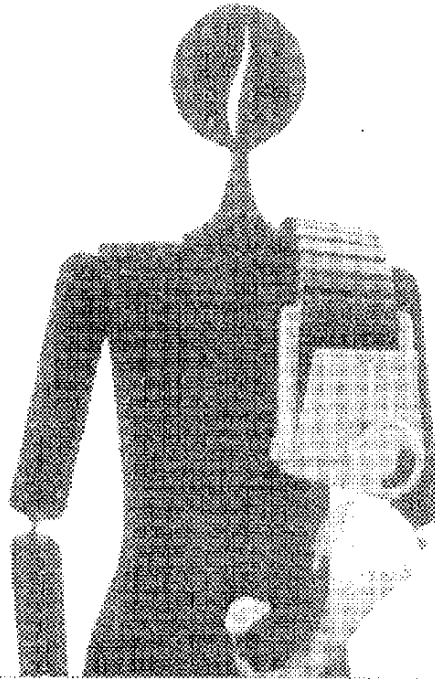


図1.1-2 ウェアラブルターミナル（模型）（出典：[日本電気90]）

### (3) Vu-Man, Navigator (CMU)

Vu-Manは、Intel386ベースの携帯型PCとヘッドアップディスプレイ（PrivateEye）からなる装着型のコンピュータである[Cogswell92, Nathan93]。キーボードは無く、ボタンで操作する。現在キャンパス内のナビゲーションなどのアプリケーションが試作されている（図1.1-3および図1.1-4）。NavigatorはVu-Manの後継版で、音声認識（CMUのSphinx）による操作が検討されている。

### (4) ユビキタスコンピューティング

東京大学の坂村（TRON プロジェクト）のHFDS（Highly Functionally Distributed System）[Sakamura87]や、Xerox PARCのUbiComp[Weiser91, Weiser93, Weiser94]（ユビキタスコンピューティング、偏在するコンピュータ）などの研究では、生活環境の中にコンピュータを偏在させていこうというアプローチをとっている。例えば、ユーザは自分のIDを赤外線や超音波で発生し続けるアクティブバッジを身に付けている。このIDによって、センサ付きのドアがユーザの存在を感知する（図1.1-5）。

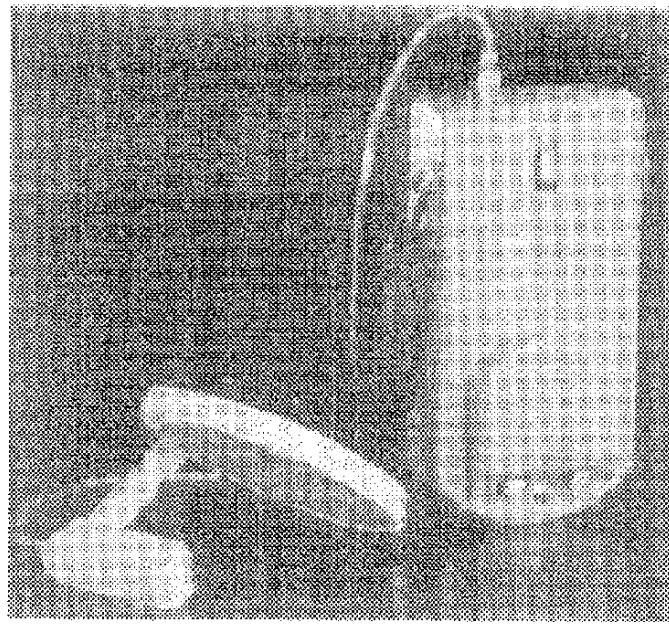


図1.1-3 Vu-Man外観（出典：[Cogswell92]）

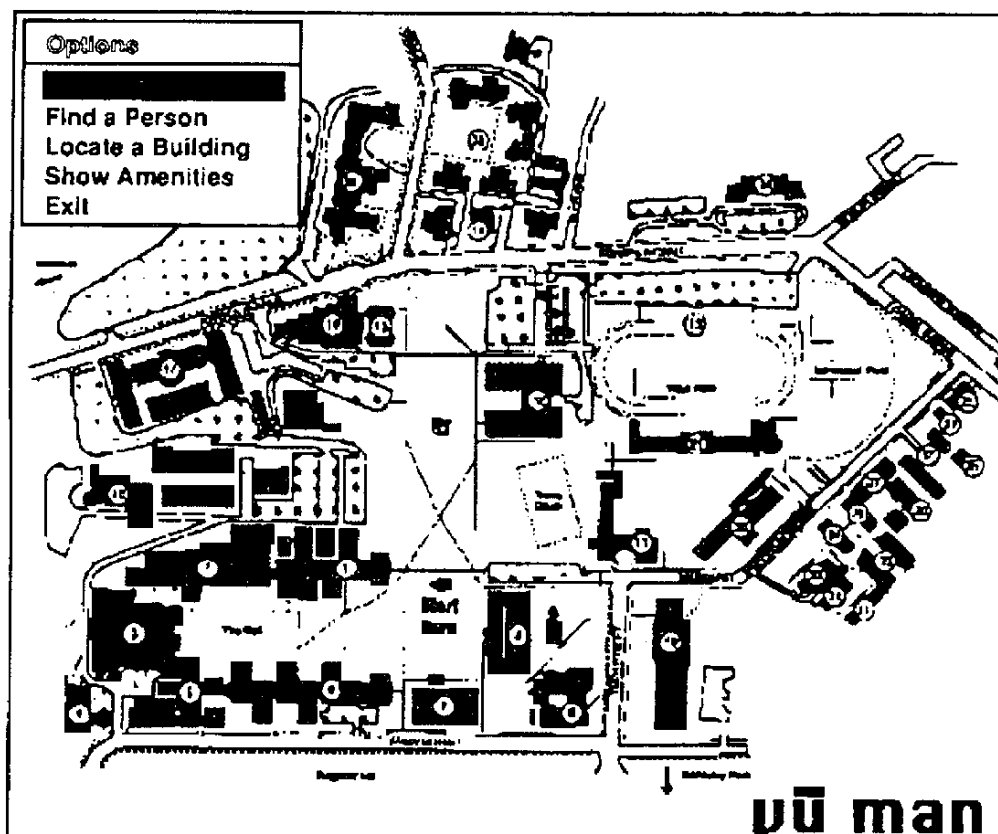


図1.1-4 Vu-Manキャンパスナビゲーションの画面例（出典：[Cogswell92]）

ユビキタスコンピューティングは研究室レベルでは成功しており、従来のデスクトップ的発想では考えられなかったような形態でコンピュータの機能を引き出すことができた。例えば、アクティブバッジ（図1.1-5）から発せられる赤外線形式のユーザID情報を会議室の壁に取り付けられたセンサが認識すると、「いつ、誰がこの会議室で会議をしたか」が自動的にコンピュータに記録される。

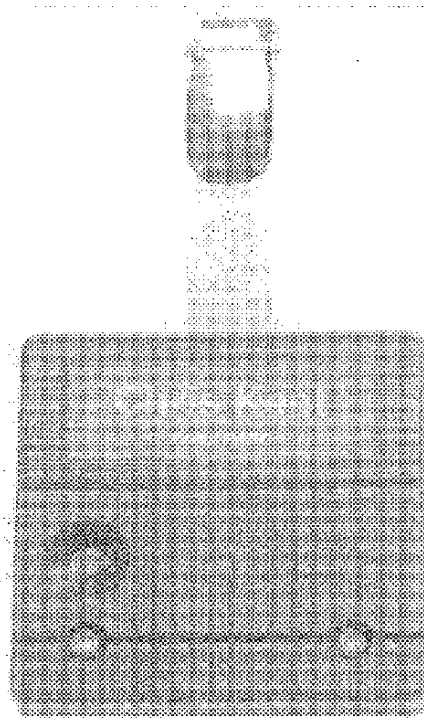


図1.1-5 アクティブバッジ（出典：[Weiser91]）

#### (5) CACMの Computer Augmented Environments特集

1993年6月に、Communications of the ACM（CACM）誌において「現実にコンピュータをaugmentする」というコンセプトに基づく研究の特集記事が掲載された[CACM93b]。またCACMの4月号（[CACM93a]）でも、次世代HCIの特集が組まれており、両者はポストGUIの流れを決定づけた特集として注目に値する。文献[CACM93b]に紹介されている事例のうち、関連するものをいくつか紹介する。

##### (a) Chameleon（Toronto大）

3次元トラックと液晶TVを組合せて、置かれている場所に即した情報が提示される一種のpalmtopコンピュータである。例えば、壁の世界地図にテレビを近づけると、近づけた地域に関連する情報が表示される（図1.1-6）。コンピュータの置かれている位置に応じて、提示される情報が変化するというところが従来型のpalmtopコンピュータと本質的に異なる点であろう。

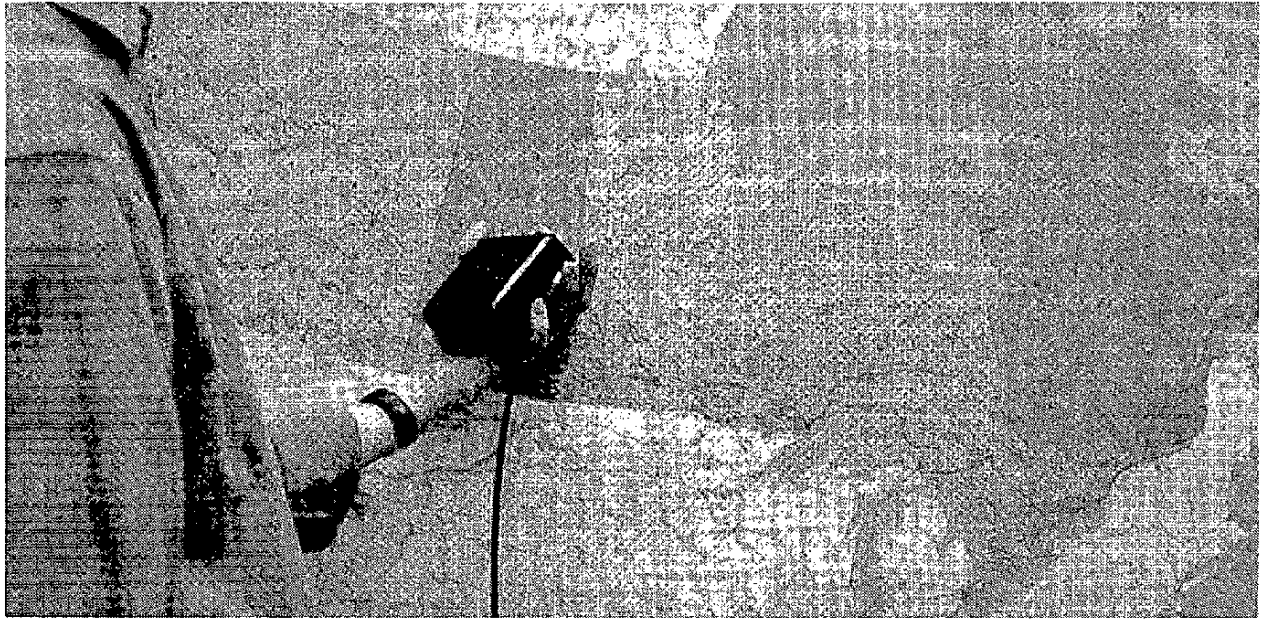


図1.1-6 Chameleon - 位置情報をセンスするPalmTopコンピュータ (出典: [CACM93b])

(b) KARMA (Columbia 大)

シースルーのHMD (head mounted display) によって、現実にはコンピュータの情報をスーパーインポーズする。レーザービームプリンタを利用する、というシナリオでのデモシステムが開発されている (図1.1-7)。例えば、プリンタの中にあるトレイがコンピュータ情報として視野に合成され、操作すべき方向が矢印として示される。

(c) DigitalDesk (Rank Xerox EuroPARC)

机上の紙の上に液晶プロジェクタによってコンピュータ情報を重ね表示する (図1.1-8)。またテレビカメラによって書類の内容や手の動きを取り込むことができる。例えば、次のような応用例が示されている。

- ・ 紙に通常のペンで描いた図形をジェスチャ操作でコピーして別の場所に表示する。
- ・ 書類中の数字を選択して電卓に送る。
- ・ 書類中の単語を指で選択すると、その意味が書類の横に表示される。

(6) 歌舞伎イヤフォンガイド

最後に、(コンピュータではないが) 状況に即した情報提示が非常に成功している例として、歌舞

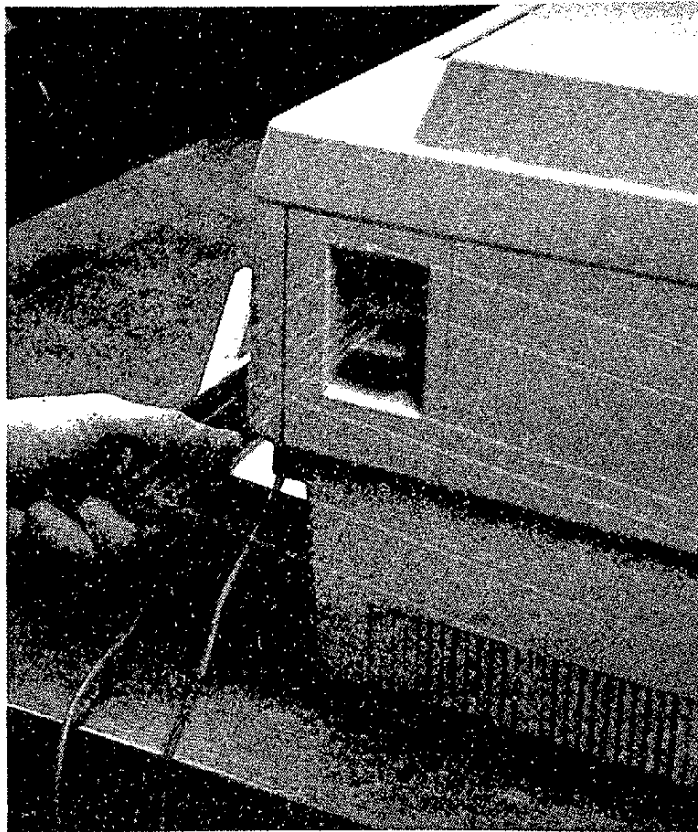


図1.1-7 KARMA — 現実世界に電子情報を合成（出典：[CACM93b]）

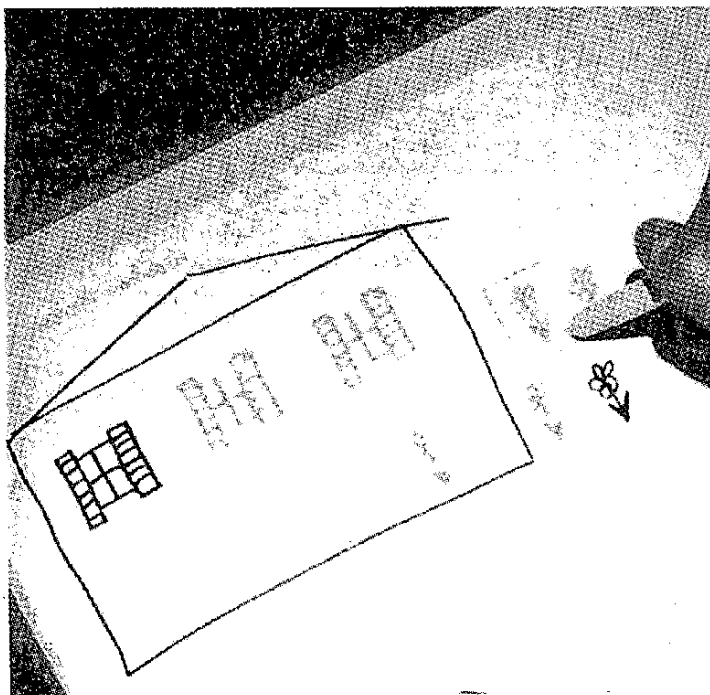


図1.1-8 DigitalDesk — 紙情報と電子情報の融合（出典：[CACM93b]）

伎イヤフォンガイドを紹介する。これは歌舞伎観劇中に利用する小型ラジオのような機器で、歌舞伎座内でレンタルして利用する。劇場中の電波によって、芝居の解説を聞くことができる。HCI的見地から興味深い特徴を列挙すると、以下のようになる。

- ・解説は芝居のペースに合わせて送出される。テープに録音された内容を専門の技師が操作しながら再生することで実現されている。例えば、同様の内容を芝居の前にすべて聞いてしまっても、ほとんど意味をなさないだろう。状況に即した解説をタイミングよく提示することで、全く同じ内容の情報でもその価値が変化する良い例である。
- ・同時通訳のレンジャーと違い、解説内容は単なる台詞の現代語訳ではない。むしろ台詞があるときには芝居に集中できるようにテープを止めている。解説は芝居の時代背景等をよりよく理解できるような補足情報に徹した内容になっている。
- ・その結果、このガイドを装着したユーザの思考の流れは、歌舞伎に対して深い知識を持つ人のそれに極めて近くなる。イヤフォンガイドから流れて来る情報は、歌舞伎鑑賞の熟練者が舞台を見ているとき、舞台上の事物に関連して想起される知識に近いと考えられるからである。このガイドを装着した人間は歌舞伎鑑賞の熟練者であるかのように振舞うことができる。

イヤフォンガイドはいわゆるハイテクな装置ではないし、コンピュータを利用しているわけでもない。しかし情報提示の本質をよく捉えており、参考にするべき点が大いにある。ここで述べている提案の目的は、このようなガイドを実世界でも利用できるようにすること、と言い換えることもできるのではないか。

#### 1.1.5 おわりに

ここでは、近未来の、コンピュータを常に持ち歩く生活が一般化している時代に要求されるユーザインタフェースについて考察した。重要なポイントは「現実とコンピュータをいかに融合するか」であり、そのために「ユーザがどんな状況に置かれているかを認識する」ことが必要であると主張した。注意しておきたいのは、ここで述べているコンピュータは現状のデスクトップ型（あるいは携帯型であっても従来型のユーザインタフェースを持つもの）を否定するものではない、ということである。ポストGUIシステム（群）は、むしろそれらのコンピュータと共存し、補完していく性質のものである。ポストGUIにとって、従来型のコンピュータも支援すべき現実の一部なのである。例えば、腕時計と置時計には、どちらがどちらを置き換えるといった競合関係はない。従来型のコンピュータとポストGUIの関係も同様であろう。

### 【参考文献】

- [CACM93a]Ed. Dave Mandelkern: Graphical user interfaces: the next generation, Communication of the ACM, 36(4) (April 1993).
- [CACM93b]Pierre Wellner, Wendy Mackay, and Rich Gold: Computer augmented environments: Back to the real world, Communication of the ACM, 36(7) (July 1993).
- [Cogswell92]Bryce Cogswell, Zary Segall, Dan Siewiorek, and Asim Smailagic: Wearable computers: Concepts and examples, Research Report CMUCDS-92-10, School of Computer Science, Carnegie Mellon University (December 1992).
- [Nathan93]Paco Xander Nathan: Escaping the desktrap: Wearables!, MONDO 2000, (9) (1993).
- [Nielsen93]Jakob Nielsen: Noncommand user interfaces, Communication of the ACM, 36(4) (April 1993).
- [日本電気90]日本電気デザインセンター アドバンスデザイン部: C&Cウェアラブルターミナル研究 (1990).
- [Ozawa90]大座畑重光: 航行めがね: 新しいコミュニケーションメディア, In Proceedings of the Japan Society of Software Science and Technology (1990).
- [Sakamura87]Ken Sakamura: The objectives of the TRON project. In TRON Project 1987: Open-Architecture Computer Systems, pp.3-16, Tokyo, Japan (1987).
- [Weiser91]Mark Weiser: 「21世紀のコンピュータ」(邦訳), 日経サイエンス, (11) (1991).
- [Weiser93]Mark Weiser: Some computer science issues in ubiquitous computing, communication of the ACM, 36(7) (August 1993).
- [Weiser94]Marc Weiser. The world is not a desktop, Interactions, 1(1): pp.7-8 (January 1994).

## 1.2 情報パースペクティブの変化とビジュアルインタフェース

近年のコンピュータのパワーアップは、ほとんどが画像表示に使われ、ビジュアルインタフェースの性能の向上に使われている。GUIの表示する画像はウィンドウの数を増やしたり動画を含むようになり、3D化やエンボス化、ドット数や色数の増大によるフォトグラフィックなリアルさの追求が行われている。

MITのアーキテクチャ・マシン・グループによる「データランド」などによる初期のGUIは、「もともと計算が得意なコンピュータに、わざわざ電卓の絵を描かせてボタンを押しながら計算させるとは何をする気だ」とひんしゆくを買ったが、こうしたインタフェースは、結果的に効率もさることながら生産性の向上に寄与した。しかしMacintoshやWindowsによるGUIの最新のバージョンは、基本的な機能はそのまま、デザイン上はもっぱら見栄えの良さの向上のみが先行しているようだ。

近年の仮想現実感 (Virtual Reality) やユビキタスコンピューティング (Ubiquitous Computing) などに見られるビジュアルインタフェースの拡張は、新規ではあるもののいま一つその機能向上の意味が不明確である。

これら最近のインタフェースに見られる問題点と意味を考察し、将来のビジュアルインタフェースへのヒントを探る。

### 1.2.1 最近のビジュアルインタフェースの傾向

仮想現実感においては、ビジュアルの3D化とユーザの視点からの情報表示の組み替えが行われ、主観的な映像の表示が強調された。当初のビジュアル環境としては、物理的にユーザを包み込む巨大な空間も考えられたが、身体に密着するHMDのようなインタフェースが主流になり、メディアのパーソナル化が進んだ。

しかし最近では、ユーザ全体を包み込むエマーシブなインタフェースが再び注目され、複数のユーザによる情報共有空間を演出することが指向されている。

ユビキタスコンピューティングは、仮想現実感による「実世界の情報空間への写像」とは逆に、情報空間から実空間への写像を行う逆のベクトルを持つ。そして、仮想空間での操作を直接的に実空間の操作に反映しようとする。これが一部では仮想現実感への批判とも受け取られている。

総じて、従来からのコンピュータと人間のインタフェースとしての界面の位置の変化と、マルチメディア化によるコミュニケーションチャンネルの幅の拡大が近年の特徴である。

### 1.2.2 ビジュアルインタフェースの何が問題か？

#### (1) パースペクティブとインタフェースの変化

ラインエディタ方式の文字ベースのインタフェースの大型コンピュータが主流の時代は、ビジュア

物的なものに対する要請はほとんどなかった。しかし、コンピュータがより小型になり、より個人に近くなるに従ってGUIなどによるビジュアル化が問題になってきた。専門家を対象にした大型コンピュータの時代は、情報の探索や多義性より、情報によるデジションの方に比重があった。つまり、コンピュータは左脳的な判断の道具であり、できるだけ解釈もしくはディメンジョンの幅を狭めることに力点が置かれてきた。大型コンピュータとユーザの距離は物理的・心理的に距離があったが、インタフェースは極めて個人の判断に近い、文字というベースにあった。

しかしミニコンやパソコンによるダウンサイジングの進行により、コンピュータは非定型業務を相手にするようになり、文字ベースではなく2次元的な画像を相手にするようになった。そして最終的な判断というより、判断以前の情報のブラウジングや組合せの変更による情報構造の再構築に重点が置かれ、ディメンジョンの拡大を指向するようになった。これを右脳的な変化と言ってもよいかもしれない。それに従って、タイプライタ的な距離にあったインタフェースの位置が、デスクトップメタファに代表されるより距離を置いたものに変化した。利用法の個人化と同時に、コンピュータのメタファはより離れた位置に界面を持つようになった。

これからさらに、コンピュータとユーザの距離が近くなるにつれ、論議されているのは、ビジュアルインタフェースの風景への変化だ。仮想現実感などの3Dインタフェースは、コンピュータとのインタフェースを奥行きのある風景へと変化させた。つまりコンピュータとユーザの距離が近くなるにつれ、コンピュータが見せる映像はより遠景へと変化する傾向がある。

それに従って、直接的なインタフェースから間接的なインタフェースへと操作法も変化し、自律的なエージェントや劇場的メタファなども研究されている。ネットワークの作るサイバースペース的なインタフェースが進化すると、ユーザの身体と直接インタラクションする情報空間は、巨大なものになる。

## (2) ダイナミックな界面を求めて

これからの情報環境は、大型のマシンから非常にパーソナルなものまで、幅広いレベルのマシンとのインタラクションが予想される。こうした場合のユーザの要請は、ありとあらゆるスコープで情報を操作できることだ。こうした奥行きのある情報環境においては、3D的なインタフェースもより意味を持ち、情報のブラウジングから絞り込み、さらにはデジションメイキングまでを一貫してシームレスに行えるビジュアルインタフェースが求められるのではないか。

## (3) もっと広い分野から学ぶ

今後のビジュアルインタフェースは、マルチメディア化やネットワーク化に対応しなくてはならない。そうした場合、より「見ること」のベーシックに帰った論議が必要だろう。映画や絵画における

映像表現、また動物行動学から視覚の持つ本質的な機能を探ったり、視覚の進化的な意味合いを研究することも必要だろう。どういう表現が理解しやすいか、という界面に近い部分でのオペレーショナルな論議では、今後のビジュアルインタフェースの発展はないように思われる。

### 1.3 インタフェースとインタラクション

ビジュアルインタフェースは、人間とコンピュータのインタラクションに関する技術の一つである。この大きな文脈の中でビジュアルインタフェースの位置は、多少狭い解釈ではあるが、グラフィックス技術を駆使してアプリケーションを使いよくするということである。本報告書には、ビジュアルインタフェースが成功しつつある様々な応用が報告されており、また、重要な要素技術も紹介されている。

しかしながら、今後の展開を考えるとビジュアルインタフェースをもう一度大きな文脈の中で検討し直す必要があると思われる。ここでは、ビジュアルインタフェースの基礎となっている考えの中に潜む問題点を取り上げ議論する。

論点は三つある。第一点はビジュアルということの意味である。コンピュータ技術の事情から言えば、近年のビジュアルに関する技術の進歩はめざましく、それによって多くの可能性が生まれてきたのは確かであるが、コンピュータをどう使うかの議論はそのような技術先行の議論だけでよいであろうかというのが、この一つ目の議論の背景にある。第二点はインタフェースという考え方である。ここでインタフェースという言葉はインタラクションという言葉と対比させて考えてみる。三つ目の論点は何をデザインするべきかである。

#### (1) ビジュアルの意味

近年のコンピュータ関連技術の中で RISCチップ技術の進歩はめざましい。この結果として、高品質なグラフィックスが低価格で可能になった。高品質の画像は、映画などの映像産業、出版、工業デザインなどの分野に多大な影響を与えた。技術の進歩は同時に安価で高速なグラフィックスの普及も促した。専門、非専門を問わず多くの人がグラフィックス機能を搭載したパーソナルコンピュータやワークステーションを使用して、グラフィックスの恩恵に大なり小なり浴している。ビジュアルインタフェースを追求するということには、この方向をもっとインテンシブに開拓し、受けうる恩恵をもっと拡大しようという発想もあると思われる。これはシーズ先行のアプローチと呼ぶことができる。

しかしながら、コンピュータ技術がグラフィックス技術周辺で大きく進んだからといって、人間とコンピュータのインタラクションにおいて視覚的なものが果たす役割が急に重要になるわけではない。人間とコンピュータのインタラクションの研究は、半分以上人間の行動の研究であり、コンピュータ科学だけにとどまらず、心理学、社会学、生物学など広い分野にまたがって学際的に研究されているところである。ここ数年の技術の発達を背景に、ビジュアルこそコンピュータを使いよくする技術であるとするのは錯覚であり、楽天的な技術拡張主義といえよう。

なるほど、コンピュータの視覚的能力は増したかのようである。しかし、進歩したのは CG のような画像の合成能力であって、それと双対な認識能力はあまり進んでいない。人間に提示するほうの品

質は向上したが、提示されたものを読み取る能力は低いままである。つまり、非対称的な成長である。インタラクションは本来対称性を前提とする。このため、インタラクションに歪みが生じている。それは例えば、ただ見るだけならば高品質な画像が提示されるが、実時間性の要求されるインタラクションを導入すると突然画質が落ちるということや、情報を3次元的に表示をすることはできても、その中を動き回ろうとしたたり3次元オブジェクトをつかんで操作しようとしたときのいらいらするような操作感の悪さに現われている。

ビジュアルの意義を否定するつもりはまったくない。実際、視覚は感覚の中でもよく発達しているところであり、これに対して上手に情報を提示できればその効果は大きい。しかし、ビジュアルはなるほど強力かもしれないが、人間とコンピュータのインタラクションにとっては、ビジュアルだけで解決できる問題は極くわずかであるということである。ビジュアルは強力である。しかし、十分ではない。考えなくてはならないのは何がゴールであり、そのためにどのようなインタラクションがあり得るかである。その上で初めてこのようなインタラクションは視覚的に行なうとよいとか、これは別の方法に訴える方がよいといった議論になるのである。この議論においては、ビジュアルは可能性の一つにすぎない。他の可能性には音を使うもの、触覚刺激を使うものなどがある。また、物理的な力をフォースフィードバックという形で導入することも考えられる。

必要なインタラクションを実現するために、こうした可能性をすべて使おうという立場はマルチモーダルインタラクションという立場であり、インタラクションの具現化として妥当な立場だと思われる。そして、マルチモーダルなインタラクションを考える際重要になるのは複数のモダリティ間の関係である。特に同期が大きな意味を持つ。人間の話言葉、表情、身振りなどの間の関係を観察すると、伝えたいメッセージを修飾したり、意味を変えたり、ニュアンスを付け加えたりするのに、それぞれが実にいろんな働きをしているのが分かる。そして、特に同期が意味の解釈において、とても実に大きな役割を果たしている。画像と音の同期がどれくらい影響力があるかは映画の演出を見ても明らかであろう。

ビジュアルインタフェースの研究は、こうしたことを踏まえてマルチモーダルな方向へと展開して行く必要がある。

## (2) 脱インタフェース

最初にも述べたが、ビジュアルインタフェース研究の本当のテーマは、人間とコンピュータのインタラクションである。インタラクションとインタフェースは、ほぼ同義として使われることがあるが、実はその根本の思想が大きく異なる。言葉としてはインタフェースの方が古い。マンマシンインタフェース、アプリケーションのインタフェース、インタフェース設計などという使い方をする。インタフェースは人間とコンピュータの接する面であり、アプリケーションの一番外側であり、アプリケー

ションの機能が煮詰まった後で一番最後に設計されるものである。つまり、インタフェースという言葉の背後には、人間とコンピュータを対立するものとして捉えるという思想があり、また、偉いのは中身であって、インタフェースは単に外側のおまけの化粧にすぎないという偏見がある。

インタラクションという用語は最近頻繁に使われるようになった。コンピュータヒューマンインタラクション、インタラクティブゲーム、インタラクティブビデオ、インタラクティブCDなどなど。インタフェースとの大きな違いは、インタラクションには形容詞があることである。名詞を形容できるということは、つまり何らかの存在形態を含意しているということである。それは、相互に影響を及ぼし合う状態、ということである。例えばビデオの場合、ビデオから映像や音楽が流れ人間に影響を与えるのは普通である。とすれば、インタラクティブビデオとは、逆に人間がビデオに作用を及ぼせる、そして、その結果ビデオの続きを変えられる、そういうビデオということである。インタラクションは、インタフェースのように面でコンピュータと人間を分けようとはしない。インタラクションは、両者を含んだ全体での営みである。インタラクションは、インタフェースのようにアプリケーションから独立して存在するものではない。アプリケーションの中身の性質そのものである。

インタフェースという概念を支持する思想は、実はコンピュータは入力を与えられたときに正しい出力が出せればよいとだけ考える機械至上主義である。出力が正しいかどうかは自明で、そのために苦労することがあるなんて想像できないし、ましてや入力の入れ方で悩むことがあることなど理解もしない。こうした発想は複雑度の高いアプリケーションの設計者にあった。彼らは難しい問題を解くシステムを設計したところで満足し、それを使うときのことは深く考えなかった。そういう設計者にとって、インタフェースという概念は手頃な処方箋だったのである。こうした考えはもちろん時代錯誤であり、最近のソフトウェア設計法の普及により、こうした考えはほとんど見られなくなった。しかし、インタフェースという用語の周辺には未だに残っている。今でもときどき何に使うのか分からない何でもできるインタフェースというのが見受けられる。こうしたものが相変わらずある最大の理由はインタフェースという概念のせいである。実際、インタフェースという言葉は死語にすると同時にこのようなインタフェースは存在理由を失ってしまうのである。

インタフェースという言葉の歴史的使命は終わった。これからはインタラクションという言葉を使い、人間とコンピュータの両者を含む系におけるインタラクションのあり方そのものを主題にすべきである。そしてありうべきインタラクションを求めてシステムの設計をしていかななくてはならない。

### (3) アプローチ

ここまで、人間とコンピュータのインタラクションに関する技術という大きな文脈でビジュアルインタフェースを考えてきた。最初に指摘したのは、ビジュアルというのは強力ではあっても十分なものであることである。進むべき方向はマルチモーダルであり、そのときに重要な課題は異なるモダ

---

#### 1. ビジュアルインタフェースについて

リティ間の同期の問題であることも述べた。次に提起したのは、インタフェースという言葉の背景にある時代錯誤的ソフトウェア観である。インタフェースという言葉捨て、インタラクションという概念に立脚しない限りこれからの時代に必要なソフトウェアの設計はできないことを述べた。

ではマルチモーダルインタラクションを指向するとして、どんなアプリケーションを設計するのか？一つの方法は様々な分野を調べ、それぞれのニーズを整理することである。しかしこれだけではない。むしろ力を注ぐべきなのは、この新しい技術を使って新しいニーズを創出することではないだろうか。コンピュータが TSS で使われる時代から一人一人がパーソナルコンピュータを持つような時代になったとき、その潮流を盛り上げたのは、コンピュータを個人の能力を増幅する思考の道具とするパラダイムであった。もう少し細いレベルでは以下のような例がある。

プログラミングに関連する活動の支援

問題解決能力の増幅

記憶の補助

理解の補助

デザインの支援

この思考の道具というパラダイムは、TSS時代に様々な応用分野を調査して出てきたニーズではない。技術的環境と社会的環境とが合わさったところで生まれてきた新しいパラダイムであった。

現在の状況もまたこのような新しいパラダイムの誕生のときかもしれない。そのような新しいパラダイム誕生のヒントになるかもしれないものを二つ挙げる。

#### (a) コミュニケーションの道具

コンピュータが仲介する人間どうしのコミュニケーションが盛んになっている。コンピュータが仲介すると、情報の蓄積・検索・加工機能が利用できる。また、通信路のバンド幅が広くなれば音声や画像なども送れるようになり、従来の通信を超えるものになる可能性もある。今現在でも以下のような形態のコンピュータが仲介するコミュニケーションがある。

例：CSCW／グループウェア、通信機能付きPDA、電子メール、電子BBS

#### (b) 社会（インタラクションの場合）の形成

上と近いものであるが、もう少し目的が明確である。この場合は、コンピュータを利用して社会的絆を強めようという方向である。コミュニケーションの仲介としてだけではなく、コミュニケーション

ンを促進するような場を創造することが課題である。

例：XEROX EuroPARCの Awareness server、Virtual Community

#### (4) まとめ

ここでは、人間とコンピュータのインタラクションという観点からビジュアルインタフェースに潜む問題点について検討した。ビジュアルというのは強力ではあっても十分なものではなく、進むべき方向はマルチモーダルであること、インタフェースの代わりにインタラクションという概念に立脚してソフトウェアの設計をしなくてはならないことを述べ、最後に、コンピュータの新しいパラダイムへの期待を述べた。

## 1.4 ビジュアルインタフェース研究の落とし穴

ここでは、ビジュアルインタフェース (VI) 研究のポイント、特にVI研究者が見落としがちなくつかの項目に関して具体例を挙げながら述べる。

また、プログラムの動きを視覚情報ではなく、聴覚情報に変換して「聞かせる」というアイデア (auralizationまたはsonificationという) についても簡単に述べる。

### 1.4.1 VI研究のポイント

#### (1) 何をみせないか？

VIということで、研究者は「何をどう見せるか」ということに心を砕く。VIにとって「見栄え」は確かに重要なファクタではある。しかし、一番大切なことは「何をみせるか」ではなくて、「何をみせないか」である。

「見る」というのは、絵画鑑賞の例を引くまでもなく、かなり能動的な活動である。人は狭いディスプレイの中とはいえ、そこに表示された様々な情報の中から自分に必要な情報をピックアップして意味を汲み取ろうと努力する。その時に余計な情報はかえって邪魔である。マシンの側から伝えたいメッセージを強調するためには、「省略すること」が大切なのである。

例えば、十数年も前にLispプログラムのvisualizerを試作したことがあるが[斉藤81]、その際にキーとなったのも「省略表示」であった。このvisualizerは、プログラムのソースコードを画面に表示し、そのソースコード上の現在実行されている部分を四角い枠で囲んで示すという原始的なものであった。ただ、ソースコードをそのまま表示したのでは非常に分かりにくく、Lispのpretty printingの機能の一つである省略表示 (あるレベル以下のS式の表示を省略する) を利用した画面の上を四角い箱が移動するように設定して、初めて理解を助けるものとなったことを覚えている。

また昔、テレビ電話の方式をいろいろと検討していた時代に、多くのビット数を使って伝送する画像の質を向上していく方向ではなく、顔や目口鼻のごく簡単な輪郭情報だけの線画図形を送るという安上がりな方式が提案された。この方式は予想以上に円滑なコミュニケーションを可能とし、手話などの伝送もこの線画方式で十分であったなどという話も残っている。これも、省略表示の御利益を示す一例である。

ユーザにとっては、すべてを狭い画面に表示してもらっても何もありがたくない。必要な情報 (それが何であるかは基本的な問題であるが) だけを選択的に表示することによって、初めて有効なコミュニケーションが成立するわけである。

#### (2) 目的に応じた見せ方

上記のことに関連して、同じものを見せるのでも「目的に応じた見せ方」を工夫することが必要で

ある。これはあまりにも当たり前すぎるので、忘れられてしまうことが多いのだが、人はその目的に応じて何を見たいかが変わるのである。

例えば、あるプログラムのだいたいの動きを理解したいと思っている人にとっては、そのソースコードの詳細をだらだらと見せられても何の役にも立たない。逆にデバッグの最中には、ソースコードのそのバグに関係した部分が一番見たいものとなる。つまり情報は常に、ユーザの目的に最も適したモードおよび詳細度で表示/提示すべきなのである。

これに関連して「マンホールの蓋の健康診断の話」は示唆的である。マンホールは毎日毎日自動車のタイヤにいじめられて傷んでくる。そのうち「ひび」が入るようになり、小さなものでも「ひび」が入ると危険なので交換しなければならなくなる。交換が必要かどうかのチェックは、マンホールを大量に所有している会社にとっては、時間と手間のかかる作業である。従来は、いちいちマンホールを外して大きな検査装置（人間用のレントゲンのようなもの）のある所まで運んで来て、精密な検査をしなければ「ひび」の有無は分からなかった。ところが、ある時ある人がうまい方法を思い付いた。マンホールを外して台の上に載せ、ハンマーのようなもので軽く叩くのである。健康なマンホールはきれいな澄んだ音を発するが、少しでも「ひび」の入ったマンホールは鈍く濁った音を発するので、その場ですぐに交換すべきマンホールが判定できるというわけである。

この画期的な方法のみそは、「ひび」の有無を知りたいという目的にとっては、マンホールの全体をレントゲンのような装置で「見る」ことは全く不要で、ただ「ひび」の有無を簡単に手に入る音情報に変換してさえやればよかったということである。大げさできらびやかな絵を見せることだけがVIではないはずだという教訓でもある。

### (3) 「本」はVIの基本

VI研究者の中には、できるだけテキスト情報を使わずに、すべてを図形的に表示しようと血眼になっている人がいるが、これも考えものである。なぜならばテキストは基本的にはvisualなインタフェースであり、中でも「本」は優れたVIであるからである。

「本」を読む/見るというのは、それ自体かなり複雑なインタラクションである。例えば、図1.4-1はあるお菓子作りの本[弓田92]の中の一部であるが、このように図や写真などを多用した本を人が眺めるときには、読み始める場所も千差万別だし、目はあちこちを動き回って必ずしも行儀よく順番に読み進むわけではない。また実際にお菓子を作る際には要点だけをピックアップして見たり、実際の作業の進み具合をメモするなどという役目も果たす。そこには必要十分な情報が盛り込まれているが、けっして煩雑ではなく、人の目を楽しませてくれるものがある。

このように洗練されたビジュアルな本の造り、デザインはVIの原点であり、まだまだ学ぶべきものがたくさんあるような気がする。適度な余白、色を過剰ではなく控えめに使った効果的な強調法などである。

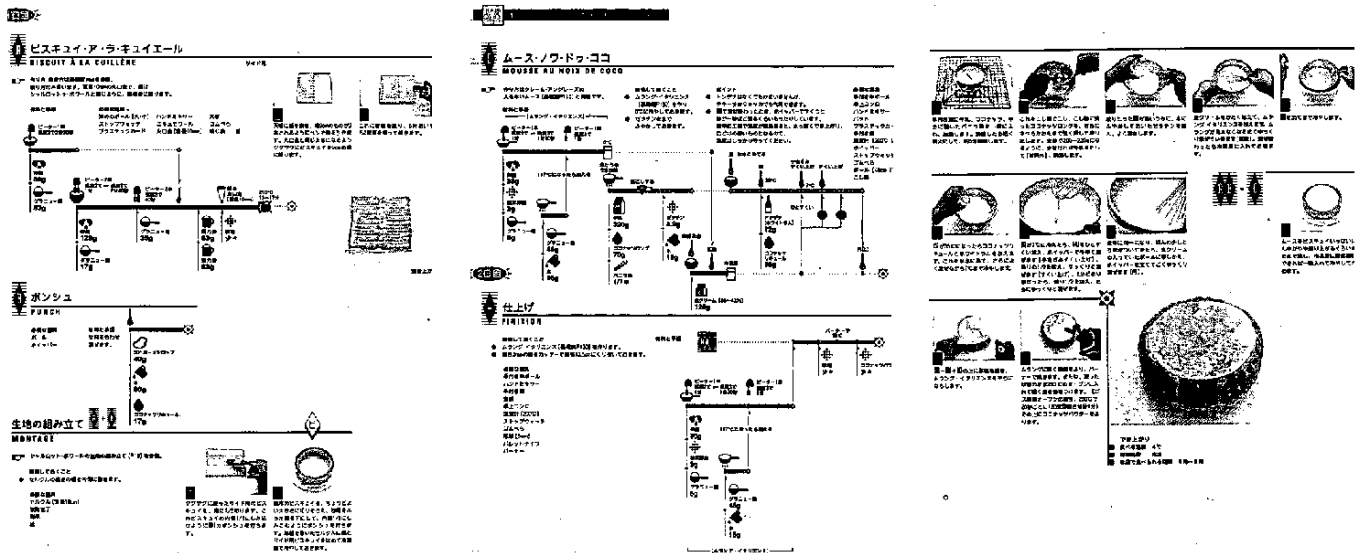


図1.4.1 ビジュアルなお菓子作りの本の例（出典：[弓田92]）

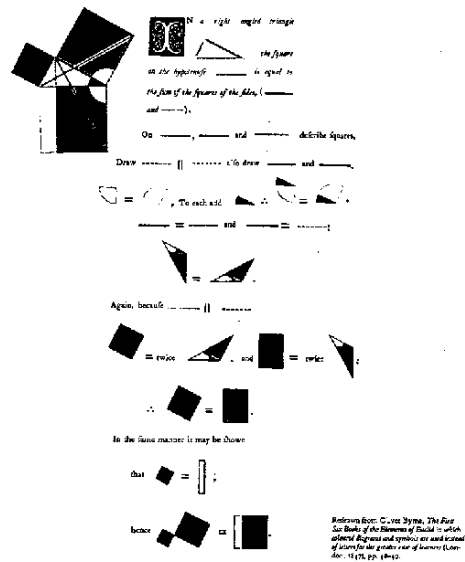


図1.4.2 ピタゴラスの定理の証明を図を使って表現した例（出典：[Tafte90]）

## 1. ビジュアルインタフェースについて

また、このようなビジュアルな本であっても、情報を伝える基本はテキストであるという事実も重要である。けっしてテキストの排除が優れたVIを作っているのではなく、テキストと適度な図、写真の混在、そして適切な役割分担と協調が、優れたVIを作り出しているのである。

どんなに頑張っても図形の可能性をぎりぎりまで追求しても、やはりテキストの必要性はなくならないという事実を図1.4-2に掲げるピタゴラスの定理の証明図[Tafte90]は示していると思う。

#### (4) 人は怠け者

上で、目的に応じた見せ方が必要であるということを述べたが、この「見え方」のコントロールをユーザに任せてしまったのでは実も蓋もない。

最近よく見かけるVIに、表示を3次元にして「ユーザがどの方向からでも眺めることができるから便利」と銘打ったものがあるが、3次元のバーチャルな空間の中をただ動き回らせるだけが能ではないだろう。人は基本的に怠け者であり、「何かを見たい」という欲求はあっても、それを実現するために、使いにくいジョイスティックで不慣れな空間の中を動き回らなければならないとしたら、最初はともかく、早晩疲れてしまうだろう。

つまり能動的なインタラクションは疲れるのである。それよりも、人が何を見たいのかをきちんと分析し、その見たいものが一番よく見える最高の視点を始めから用意してあげるのがVIデザイナーの役目だろう。

#### (5) グラフィカルデザインの視点

Edward R. Tufteという人が書いた「The Visual Display of Quantitative Information」という本がある[Tufte83]。この本は、科学分野の本や雑誌の中の図や表をデザインするデザイナーたちのバイブル的な本で、グラフィカルデザインの要点を豊富な例を使って分かりやすく解説したものである。

ここでは紙面の都合でそのすべてを紹介することはできないが、本や雑誌のグラフィックスに関して書かれた本であるにもかかわらず、計算機のディスプレイ上に物事を表現するVIのデザイナーにも役に立つことがたくさん書かれている。

例えば、一つの要素に複数の役割を担わせることとか、データの密度をどの程度にするのが適切かとか、単純な要素を繰り返し使って全体を構成する方法の功罪とか、細かな留意点から始まって、図全体の統一観の達成に至るまで、実に多くのヒントが隠されている。

全体の主張をあえて一言でまとめれば、「注意深くデザインすることの大切さ」を教えているとも言えようか。彼の主張する「洗練されたグラフィックス」(Graphical Excellence)の考え方に従えば、

Graphical excellence is that which gives to the viewer the greatest number of ideas  
in the shortest time with the least ink in the smallest space.

ということになる。要するに、トータルなインタラクションをデザインすることがグラフィカルデザインの基本であるとでもいえようか。

前にも述べたように、本というのは優れたVIであると考えられるので、そのグラフィックスの構成法に関して優れた解説書があるのならば、それを計算機のVIのデザインに応用しない手はない。

にもかかわらず、VIの研究者たちはこのような知恵の蓄積を無視して、自分勝手なインタフェースを次から次へと作り出しているような気がする。最近やっと、アニメーションの手法を利用するというような動きが出てきたようだが、さらにこのような本を勉強して、グラフィックスの出来映えを向上させる必要があるだろう。

同じ著者が最近書いた「Envisioning Information」[Tufte90]も同様に参考になる。図1.4-2はこの本から取ったものである。

## 1.4.2 聴覚化

### (1) アイデア

ここでは、一般に考えられているVI研究の枠から若干はみ出す試みである「音の利用に関する研究」のアイデアを紹介する。

基本的なアイデアはいたって簡単で、以下のように述べることができる：

「プログラムの走行中に、そのプログラムの処理の様子、挙動を彷彿とさせるような音（音楽）を出して、デバッグやソフトウェアの理解を助けようというアイデア。」

現在のマンマシンインタフェースは視覚に頼りすぎているが、もう一つのモードである聴覚も利用しない手はない。もちろん、視覚化（visualization）と組合せることも考えられる。既に音の付随した視覚化は多い。

### (2) 何が問題か？

一言で言うと、プログラムの実行に関するあらゆる情報の中の「何を」、あまたある音のうちの「どのような音」にマッピングして聞かせるかというのが最大の問題である。これをマッピングの問題と呼ぶことにしよう。

例えば、メモリ消費量をその量に応じた高さの音にマッピングするということが考えられる。果してこのマッピングによって、聞いている人にメモリの消費量が増えたあるいは減ったという印象を与えることがうまくできるかどうかの問題となる。

また、実行に関する「何」を音に対応させるかに関しては、様々な選択がありえる。例えば、プロ

グラムが再帰呼び出しを行なっているという事実を、なんらかの音に対応させるということを考えることもできるが、さて「再帰」をうまく表わすような音は何だろうか？無限音階はどうかというアイデアもあるが、すべては試してみなければ分からない。

もう一つ大きな問題として、実行速度と可聴音の周波数のミスマッチの問題がある。最近の計算機はその処理速度が圧倒的に速いので、そのスピードをそのまま音に変換したのでは、そもそも音として聞こえる範囲の音にならないという問題である。この問題をスケーリングの問題と呼ぶことにしよう。

この問題の解決法としては、実際の処理速度を遅らせて人に聞こえるようにするという単純な方法もあるが、必要な情報を適当なサンプリングレートで取り出して、実行に関する情報のうちのごく一部を音にするという方法が考えられるし、ログファイルを作って、後でそのログから音への変換を行なうというパッチ的な方法もありえる。

これらの問題が聴覚化 (auralization) を考えるときの基本的な問題である。

### (3) 聴覚化への本質的な批判

上記のような問題を考察して、聴覚化の実験的なシステムを作り、いくつかの例で試してみるというのが聴覚化研究の筋道である。ただ、このようなまったくのアイデアの段階でも、聴覚化に対してかなり本質的な批判があることも確かである。その一つを紹介する。

いろいろなデータには、そのデータごとに自然な表現ドメイン (時間、空間、 $n$ 次元、絵、音) が存在する。この自然な表現ドメインを別ドメインへ変換することによって、見えなかったものが見えてくる (聞こえなかったものが聞こえてくる) メリットを追求するのは面白い。ところが、いままで考えてきた聴覚化にはそのような面白味がないのではないかという批判である。

プログラムの実行というのは、基本的に時間軸上の出来事である。それを、同じ時間軸上のシーケンスである音にマッピングしても、見えなかったものが見えるようになったりはしないのではないかというわけである。

この批判に答えるためにも、先ずはこの単純な時間軸から時間軸へのマッピングを試してみるべきだろう。確かに、プログラムの論理構造のようなものが音に変換できたらさらに面白いのかも知れない。

### 【参考文献】

[斉藤81] 斉藤康己: グラフィック・ディスプレイを多用したLispプログラミング環境, 情報処理学会, 人工知能と対話技法研究会, 21-1 (1981.6.19).

[Tafte83]Edward R. Tufte: The Visual Display of Quantitative Information, Graphics Press (1983).

[Tafte90]Edward R. Tufte: Envisioning Information, Graphics Press (1990).

[弓田92]弓田享: 小さなレストラン、喫茶店、家庭で作る少量でおいしいフランス菓子のためのル  
セットウ, 第二巻, 実践編1, (株) イル・ブルー・シュル・ラ・セーヌ企画 (1992).

## 1.5 ビジュアルの次に来るもの

### 1.5.1 はじめに

1970年代の初めに、Alan Kay等がXerox PARCで初めて本格的に作ったグラフィカルユーザインタフェース (GUI) は、1980年代になって、Macintosh、Windowsなどの製品やX-winodwなどのソフトウェアとして世の中に出され、1990年代になってその利用が一層広まってきた。

ここでは、このようなビジュアルインタフェースの持つ意義をまず理解しておく。次に、ビジュアルインタフェースの問題点を捉え、その解決案を探ることにする。その最初の候補として、「マルチメディアインタフェース」が挙げられるが、これ単独では不十分であることを明らかにする。そこで、人間どうしのコミュニケーションを手がかりに、ビジュアルインタフェースに代わる新しいインタフェースをさらに模索する。

### 1.5.2 ビジュアルの良さ

ビジュアルインタフェース以前のコンピュータのインタフェースは、文字と数字だけのコマンドラインインタフェースが中心であった。現在では、ビジュアルインタフェースが全盛である。ここでは、まずビジュアルインタフェースの持つ利点と特徴がどこにあるかを最初に確認しておきたい。

#### (1) 人の認知

従来のコンピュータは、論理と言語的処理が中心であった。人とコンピュータとのインタフェースも言語的なコマンドラインインタフェースが中心であった。

しかし、人間の情報処理は、言語的なもの以外に、図形的なものが有効であることが昔から知られている。例えば、数学の定理の証明や経済統計の説明などでも図形を用いた視覚的なものにより、分かりやすさが急に増すことが知られている。さらに、20年前頃、脳科学の研究（具体的には、てんかん患者に対する分離脳の研究など）から、人間の脳は左半球で言語的処理を行い、右半球で図形的・空間的処理を行うことが確認されている。以上のように、人間の情報処理は、言語的・論理的処理がすべてではなく、図形的・空間的処理も重要であり、さらに、この両者が相補いあってこそ、有効に作用することが分かる。

#### (2) 視覚と空間的認知能力

ビジュアルインタフェースで中心となる感覚器は人間の視覚である。視覚は、ヒトを含む動物にはあるが、植物にはない。動物は行動し、ものと自分との位置関係を把握するために視覚を利用する。さらに、視覚により空間認知能力が発生するが、このことは先天盲に関する研究でも裏付けられている。記憶法の一つに空間認知と関連付ける方法がある。この場合、視覚が重要な役割を果たすのはいうまでもない。

### (3) ピアジェの段階的発達説

スイスの発達心理学者であるピアジェは、子供の認知的発達の段階を、

- ・ 身体的段階
- ・ 視覚的段階
- ・ 記号的段階

の三つに分類している。確かに、生まれてすぐの子供から幼児にかけては、穴が空いていると指をつっこんだり、出っ張っているものがあれば、押したり引いたりして、身体でいろいろ試しているようであり、さらに小学校低学年から高学年にかけては、視覚的に見たもので判断することが多くなる。さらに中学生ぐらいになると、やっと論理的思考や記号的操作などをこなせるようになってくる。成人では、これら三つの要素を組合せて、人間としての行動を行っている。

従来のコンピュータでは、このうち、記号的段階に相当するものしかなかった。ビジュアルインタフェースが作られるようになって初めて、視覚的なものが入り入れられるようになった。この三段階説に従えば、次に必要になってくるのが、身体的な認知をも考慮にいれたインタフェースということになる。

#### 1.5.3 ビジュアルインタフェースの問題点

では、ビジュアルインタフェースを手放しで評価できるだろうか？

以下では、現在のビジュアルインタフェースの持つ問題点を調べることにする。

##### (1) ビジュアルと疲労

ビジュアルなものは情報量は多いが、ずっと見続けると目が疲れる。また、同じ画面を見続けると眠くなることもある。人間は、目をずっと使っているが、視線は動き回り、けっして一箇所を見続けるようなことはしていない。さらに、見るだけでなく、聞くことや身体を動かすことなども連動させて行動している。

##### (2) ビジュアルと想像力

見る、あるいは、見せる、ということは具体的なインスタンスを提示することである。音で聞くだけとか、あるいは、文字で読むだけの場合、それがどのようなものであるか、想像する楽しみがある。ビジュアルなものは、想像力を奪ってしまう問題がある。

映画を作る人々は、とっくにこの問題に気がついていた。文学作品を映画化する場合に、いかに原文の持つ味わいを殺さないようにするかが、一つの腕の見せどころである。

ビジュアルインタフェースにおいても、今まではいかに多くの情報を見せるかに主眼があったが、

これからは、いかに隠すか、あるいは、いかに必要な情報だけに絞り込むかの技術が、むしろ重要になってくる。

### (3) ビジュアル単独の問題

人間の感覚は、視覚以外にも、聴覚、触覚、味覚、嗅覚、体感などいろいろな感覚が複合して回りの環境を感じとっている。視覚だけ、というビジュアルインタフェースは、この点では問題がある。

また、人間は、外界の情報を感覚器を通して受け入れるだけでなく、その結果を運動として、外界への働きかけも同時に行っている。この意味では、見るだけでは十分ではなく、インタラクティブに働きかける要素も欠かせない。

したがって、複数の感覚器と運動機能をできるだけ活かしたような、インタフェースが望まれる。

### 1.5.4 マルチメディアは答になるか？

以上述べたような、ビジュアルインタフェースの問題点を解決するものとして、マルチメディアインタフェースはその候補となるのだろうか。次にこの可能性を検討してみよう。

#### (1) 素材としてのマルチメディア

マルチメディアに関する期待は、現在、非常に高まっているが、そもそもマルチメディアとは何か、については必ずしもはっきりしない。

コンピュータが扱う素材が、従来の文字やテキストとしてのメディアのみから、画像、動画、音声などのメディアをも含むものへと変わったものをマルチメディアと呼ぶのが基本的な定義と考えてよいであろう。このような定義においては、マルチメディアとは、コンピュータが扱う多様な素材であるという点と、その素材がデジタル化されている点とに特徴がある。コンピュータが扱う素材が拡大すること、および、従来、アナログ的に扱ってきたものがデジタル的に扱える、というのはこれからの共通的な方向性であり、このことのみでは、ビジュアルインタフェースに代わるものとして、「マルチメディアインタフェース」を捉えるわけにはいかない。

#### (2) 通信・蓄積技術の課題

そこで、マルチメディアの第二の意義を考える。これは、第一の定義から導かれることである。イメージ、音声、動画などをデジタル的にコンピュータで扱うとなると、技術的に問題となるのは、その情報量の多さであり、その結果、これらのマルチメディア（の素材）をいかに効率よく蓄積し伝達するか、というマルチメディアの通信・蓄積の技術がクローズアップされてくる。具体的には、静止画・動画・音声などの圧縮・解凍技術と通信におけるプロトコルが技術課題となる。

---

### 1. ビジュアルインタフェースについて

これも、マルチメディアの技術的側面であって、ユーザインタフェースの直接的な課題とはなり得ない。

### (3) 応用中心からユーザ中心へ

では、技術的な課題ではなく、インタフェースの観点からは、マルチメディアには、どのような期待が持たれるのであろうか？

マルチメディアの言葉の中に含まれ、インタフェースと強い関係があるのが、ハイパーテキスト性とインタラクティビティとであろう。このうち、ハイパーテキスト性は、マルチメディアに限らず、モノメディア（テキストのみ）の場合にも有効な概念であるが、マルチメディアにおいて、さらにその重要性が増してくる。また、人とコンピュータとのインタラクションもマルチメディアだから新たに生じたものというよりも、従来のアナログ的なものをデジタル的に扱うようになったときに必然的に要請されるものである。

別の言葉でいえば、技術中心で無く、ということを強調すると、では、応用（アプリケーション）中心へ、と短絡しがちであるが、一方的に応用を作りそれをユーザに提供するだけでは十分ではない。ユーザ中心という、ヒューマンインタフェースの基本を考えると、インタラクション（ユーザとコンピュータとのやりとり）こそ重要である。

結局、ヒューマンインタフェースの観点からは、マルチメディアの素材としての側面や技術的側面よりも、ユーザ中心のインタラクションを重視することが大切である。しかし、一般的なマルチメディアの捉え方では、素材や技術のみに言及する場合が多く、人とのインタラクションが軽視されがちである。

## 1.5.5 人のコミュニケーションでは

コンピュータと人とのコミュニケーションをいったん離れて、実世界における人と人とのコミュニケーションを考えてみる。

### (1) 書き言葉と話し言葉

現在、本などの印刷物が広く普及し、人と人とのコミュニケーションは本などの印刷物（書き言葉）が基本である、との錯覚が起こりそうである。しかし、人と人とのコミュニケーションの基本は、むしろ話し言葉にある。

これは、日常生活のみならず、仕事上の打ち合わせに留まらず、学会活動などにおいてもいえる。聴覚障害者の場合にも、話すことが難しいなら、文字を書いてコミュニケーションをすればよいのでは、と一般に考えがちであるが、実際に聴覚障害者が主に使うのは、（書かれた）文字ではなく手話

である。手話は、聴覚障害者にとっての話し言葉であり、手話を用いることによって自然な会話が可能になっている。

## (2) 論理と感情

人のコミュニケーションの中心が書き言葉ではなく、話し言葉である、ということは、コミュニケーションの内容を考えるとさらにはっきりしてくる。

書き言葉で主に伝えられるのは、一義的な情報や論理であって、話し言葉が本来持っている感情とか欲求などは、書き言葉にしたとたんに失われてしまうことが多い。

このことから、人と人とのコミュニケーションにおいては、論理だけでなく、その背後にある感情とか欲求が重要な役割を果たしていることが分かる。この点は、最近の脳科学の研究からも裏付けられている。ヒトの脳は、知的な処理をする大脳新皮質と、大脳新皮質と脳幹との間にあって情動反応を受け持つ大脳辺縁系の両方の働きで、コミュニケーションを行っている。

## (3) 複数のモダリティ間の協調

話し言葉には、音声を使うので主として聴覚を使うが、これがすべてではない。話しているとき、相手の顔や口に動きを見ているか見ていないかで、理解の程度が異なることは、よく経験するところである。また、身振りや目の動きも重要な要素になっている。

すなわち、人と人とのコミュニケーションでは、複数の伝達様式（モダリティ）を用いてコミュニケーションを行っているのである。これをマルチモーダルコミュニケーションという。

## (4) コンピュータとの間では

では、コンピュータの場合はどうだろうか？現状のビジュアルインタフェースでは、視覚単独のモノモーダルであり、書き言葉を読む、論理に偏ったものになっている。

人とのコミュニケーションを参考にすれば、コンピュータと人とのインタフェースにおいても、マルチモーダルが必要になってくるであろう。伝えるべき内容も、一義的な情報である論理や知識のみではなく、その裏にある感情やニュアンスなども含めた豊かなコミュニケーションが望まれる。

マルチモーダルを活かしたインタフェースを、マルチモーダルインタフェースと呼ぶ。マルチモーダルインタフェースは、ユーザ中心のインタフェースであり、視覚のみならず、聴覚、触覚などのできるだけ多くの感覚器を有効に使うと同時に、コンピュータの側にも簡単な画像認識を行わせて、人の身振りや目の動きも入力の一部とするものである。

### 1.5.6 実世界との関わり

マルチメディアと並んで、仮想現実感 (Virtual Reality) は、人間の感覚に訴えるものが多いため、ビジュアルインタフェースに代わるもの、という期待もある。この妥当性を検討するために、コンピュータの世界 (仮想世界) と現実の世界 (実世界) との関係を次に考えてみたい。

#### (1) 窓の拘束を逃れられるか?

現状のビジュアルインタフェースの基本は、ウィンドウ (窓) である。窓を通して、現実世界のモデルである、人工世界のオブジェクトを眺め、処理をしている。人は通常、一つの窓からしか外を眺められない、ということはない。例外は、刑務所に入っている囚人とか、満員の通勤電車に乗っている乗客くらいである。もちろん、現在のワークステーションでは、マルチウィンドウになっているので、複数の窓を切り替えることは可能である。しかし、このような窓そのものの制約を外すことができないだろうか。

#### (2) 仮想世界=閉じた環境

仮想現実感の技術を使えば、窓には囚われずに、いろいろな世界を (仮想的にはあるが) 眺めることができそうである。しかし、現状の仮想現実感では Head Mounted Display と呼ばれる、身体を拘束した装置を身につける必要がある。また、実世界と切り離された仮想世界に閉じてしまっている例が多い。ゲームなどの娯楽とか、未来やあるいは (宇宙や原子炉の中などの) 極限状態を疑似体験する場合にはこれでよいが、やはり、「現実」との関わりを持ったシステムが望ましい。

#### (3) 実世界との相互交流

最近、このような「現実」と切り離された仮想現実感ではなく、実世界との相互交流を持ったヒューマンインタフェースの提案も徐々にではあるが増える兆しがみられる ([Wellner93])。

例えば、障害者にとっても、単に空想の世界で遊べるだけでなく、日常生活における行動をサポートしてくれるようなインタフェースが望ましい。

実世界から情報を受け取り、処理し、その結果を再び実世界にフィードバックし、実世界との意味のある交流を行うような、ヒューマンインタフェースシステムの研究開発を今後も行っていく必要がある。

### 1.5.7 おわりに

ここでは、ビジュアルインタフェースの次に来るものとして、マルチモーダルインタフェースを一つの可能性として提示した。マルチモーダルがすべてを解決してくれるとは思われないが、基本的な

方向性として押さえておくべき重要なアプローチの一つであることは間違いない、と思われる。

今まで、コンピュータサイエンスの牽引車とされていた、AI（人工知能）も転換期にきている。単に、推論のメカニズムを明らかにするだけとか、知識の量を増やすだけでは不十分であるとの認識から、状況とか環境における複数のエージェントの、部分的な情報に基づく知的振る舞いの方に関心が集まっている。ビジュアルインタフェースが全盛の現在のヒューマンインタフェースも、次の新しいインタフェースを模索するに適した環境が熟成しつつある。

ビジュアルインタフェースが初めて作られたのは、今までのコンピュータとは違った新しいコンピュータをイメージすることから始まった。ビジュアルインタフェースの次のインタフェースを考える場合も、今までのコンピュータとは違った新しいコンピュータの形と意味を考えることが「初めの一歩」となるであろう。

### 【参考文献】

- [Blattner92]Blattner, M. M., and Dannenberg: Multimedia Interface Design, ACM Press (1992).
- [Kay92]Kay, A., 鶴岡雄二訳, 浜野保樹監修: アラン・ケイ, アスキー出版 (1992).
- [Krueger91]Krueger, M. W.: Artificial Reality II, Addison-Wesley (1991) (下野隆生訳, 人工現実 — インタラクティブ・メディアの展開 —, トッパン (1991)).
- [Laurel90]Laurel, B., Ed: The Art of Human-Computer Interface Design, Addison-Wesley (1990).
- [佐々木87]佐々木正人: からだ: 認識の原点, 東京大学出版会 (1987).
- [下條93]下條真司: マルチメディアの理想と現実 — 研究と商品開発 —, 情報処理, Vol.34, No.4, (1993).
- [Shneiderman92]Shneiderman, B.: Designing the User Interface, 2nd ed., Addison-Wesley (1992).
- [Shu88]Shu, N. C.: Visual Programming, Van Nostrand Reinhold (1988) (西川博昭訳: ビジュアル・プログラミング, 日経BP社 (1991)).
- [館92]館・廣瀬監修: バーチャル・テック・ラボ, 工学調査会 (1992).
- [安村93a]安村通晃編, アクセス研究会: 障害者向けコンピュータ・インターフェイスへの序奏, 慶應義塾大学環境情報研究所 KEIO-IEI-RM-93-004 (1993).
- [安村93b]安村通晃, 伊賀聡一郎, 福井寿文: 音声対話システムの試作と評価, 慶應義塾大学環境情報研究所 KEIO-IEI-RM-93-009 (1993).
- [安村93c]安村通晃, 伊賀聡一郎: マルチメディアからマルチモーダルへ, Proc. of WISS'93 (1993).
- [安村94]安村通晃, 岡部 学, 田中公二他: マルチメディア言語学習環境の構築, 慶應義塾大学環境情報研究所 KEIO-IEI-RM-94-004 (1994).
- [Wellner93]Wellner, P., et.al. Eds: Computer-Augmented Environments: Back to the Real World, Comm. ACM, Vol.36, No.7 (1993).

## 2. ビジュアルインタフェースの技術動向



## 2. ビジュアルインタフェースの技術動向

### 2.1 実世界指向インタフェースの研究動向

#### 2.1.1 はじめに

ここ数年のCHIの予稿集を中心としてヒューマンインタフェースの最新動向を述べる。具体的には、以下のテーマに大別してポストGUIの展望を述べる。

- ① GUI
- ② マルチメディア/マルチモーダル
- ③ エージェント
- ④ 仮想現実感 (Virtual Reality)
- ⑤ 実世界指向インタフェース

#### 2.1.2 GUI関連研究の動向

現在のGUIの課題、言い換えると次世代のGUIが備えるべき要件として議論されているのは次の三点に絞られる。

- ① 優れたGUIを設計するための方法論の確立
- ② GUI開発工数の削減
- ③ 新しい有効なメタファ

##### (1) インタフェース設計方法論

良い設計手法と優れた評価手法は表裏一体の関係にある。これまでに提案されている評価手法としては、専門的評価者によるHeuristic Evaluation[Nielsen90]、実使用環境での可用性テスト (Usability Testing)、インタフェースガイドライン[Apple Computer89,Brown88]との照合、認知ウォークスルー (Cognitive Walkthrough) [Lewis90]などが挙げられる。またこれらを前提とした設計手法、例えば初期設計時点からのユーザ参加を重視するユーザ参加設計 (Participatory Design) [Johnson90]をはじめ、おなじみのラピッドプロトタイピング (Rapid Prototyping) や繰返し設計 (Iterative Design) などが実際に効果があることが多く報告されている。

しかしながら現実のシステム設計では、これらの方法の効果を知りつつも適用する時間的余裕がないことがほとんどである[Mulligan92]。そこでタイトなスケジュールを嘆く前にありうる解の一つは、要するに手早く簡単にシステムができてしまう方法を思い付くことであろう。というわけで次のテーマが重みを増す。

## (2) GUI開発効率の向上法

GUI開発の工数を削減する方法として従来から提案され実用になっているのは、部品であるウィジェット (widget) を組合せてGUI構築を容易化しようとするユーザインタフェースツールキット (User interface toolkit) [Nye92]あるいはUIMS(User Interface Management System)[Green87]のアプローチである。またこれらをベースにしたいわゆるユーザインタフェースビルダ (User Interface Builder) も優秀な製品が出回るようになってきた。インタフェースビルダの問題点として指摘されているのは、ビルダで作成したGUIがアプリケーションのセマンティクスと独立になっている点である [Johnson92]。多くの設計者が気づいているように、アプリケーションの持つ基本データ構造とGUI経由で入出力される情報の間にはしばしば直接的な関係がある [de Baar92]。そこで、アプリケーションの基本設計から適切なGUIを自動的に生成してしまおうという試みが始まっている [Janssen93]。

一方、コーディングレベルのカスタマイズやプログラミング作業をビジュアル化して少しでも簡単にしようという研究も数多く行われてきた。代表的なものは例示プログラミング (Programming by Example) あるいはデモによるプログラミング (Programming by Demonstration) と呼ばれる、実際の操作の履歴からプログラムを生成する手法である [Myers88]。これも例えばスプレッドシートでのマクロ作成等で製品レベルで応用されている。この方法の問題点の一つは、条件分岐や繰り返しといった制御が作りにくいことである。条件分岐については複数の例示から合成する方法 [Kojima89] が、また繰り返しについてはユーザの操作パターンから反復分を検知して生成する方法 [EAGER91] がそれぞれ提案されている。もう一つの問題点は、例示操作から何が作られたのかがエンドユーザにとって透明でないことが挙げられる。生成されたスクリプトをテキストとして表示し適宜編集を許すのが常道ではあるが、一般に生成されたものは読みづらく、ましてそのスクリプト言語を知らない人にとっては意味がない。これに対処するアプローチとしてユニークなのは、EAGER[EAGER91]の考え方である。EAGERはユーザの操作に繰り返しパターンを発見したと確信するとユーザの次の操作を予測して先回りして表示する。それが正しいか否かで、ユーザはシステムがうまく繰り返しを生成しているかをスクリプトを見なくても判断できるというわけである。

## (3) GUIにおける新しいメタファ

現在のPC/WS上では、デスクトップメタファが完全に主流を占めている。その自然な拡張として考えられるのが、複数のワークスペースを使うルームメタファ [Henderson86]である。ルームメタファは、今後グループウェアへの応用が増えるにしたがって必然性が高まると考えられる。また他の今後重要となるメタファとしては本メタファ [Arai92]が挙げられる。

一方、急速に改善されているグラフィックス性能を利用して、デスクトップ上の情報の可視化レベルを向上させる試みも行われている。代表的なのはInformation Visualizer[Card91]である。Information Visualizerでは、大きなファイル階層やスプレッドシートの表示を3次元的に行って効果をあげている。Information Visualizerの考え方やウィンドウシステムにおけるウィンドウの表現自体もより豊かにしようとの試み[Staples93]は、数年のうちに現在のデスクトップのLook&Feelを大きく変えると予想される。

### 2.1.3 マルチメディア/マルチモーダル

CG、ビデオ等の映像メディアや音声、音楽等の音メディアなどマルチメディアを活用して自然で豊かなヒューマンインタフェースを実現しようとするアプローチも、ハードウェア環境が整うのに伴って俄然現実味を帯びてきている。ビデオを使うものとしてはVideoDraw[Tang90]に始まる共有空間の実現を狙ったものが興味深く、盛んに研究が行われている。グループウェアに関するものは、臨場感の実現のために相手の画像をTV電話風に表示するものが多い。これに対して、共有すべきはお互いの作業対象であり、例えば隣に座っている人とも使いたくなるようなシステムでないと意義は小さい[Nardi93]というのは鋭い指摘ではなかろうか。またビデオの重要性が高まるのに伴い、その検索や編集といった技術も着実に進歩している[Ueda93]。

また複数の入出力のモダリティを随時切り替えたり多重に用いることによって、柔軟性と頑健性を向上させようとするマルチモーダルインタフェースの考え方も定着しつつある。具体的には、音声入力とジェスチャや視線追跡の組合せなどが用いられている[Blattner90]。音声による入力は否定的議論も多いテーマである[Brennan91]。しかしながらふと思いついたことを入れておきたいといったような、必然性のある音声の利用形態を提案しているVoiceNotes[Stifelman93]のような研究が新局面を開くように思われる。出力側のモダリティとしては表情を利用するものも面白い[Takeuchi93]。

### 2.1.4 エージェント

インタフェースにおけるエージェントとは、「何かを依頼すると代行して処理してくれる存在」であり、この意味ではソフトウェアサイエンスの中で用いられるエージェント指向と違いはない。しかしながらヒューマンインタフェース研究の中では、ともすれば古くから論争のある「擬人化」の問題と密接に絡み合って議論されることが多い点が異なる[Don92]。実際ユーザインタフェース設計の教科書の多くは、ダイアログボックスのメッセージとして「あなたのファイルを私は壊してしまいました。申し訳ありません」などと決して表示してはならない、「ファイルが壊れています」とクールに出しなさいと教えている。CG技術の進歩でいかにも人間のような顔のCGが出せるようになって、擬

人的インタフェースも見栄えは格段に進歩しているが、論争の本質には変化がないように感じる。反対派（であると同時に主流でもあるように思われる）の意見は、「ユーザをあざむくものであり、非生産的であり、かつ道徳的にも疑問」というものである[Don92]。にもかかわらず、SFをはじめApple社の「Knowledge Navigator」、Hewlett-Packard社の「1995」などのプロモーションビデオには、擬人的エージェントインタフェースが満載されているところに何か人の本質的欲求のようなものを感じるのである。

エージェントを使って成功が期待できる領域として、アミューズメントの他に教育分野が期待できる。具体例として、アメリカの歴史のマルチメディアDBのブラウザであるGUIDES[Don93]が挙げられる。GUIDESでは、ある地方のある時期をそれに合わせた容姿の擬人的エージェントがユーザと対話しながら説明する形式を採用している。GUIDESのように対象タスクを選んで認知心理的に慎重な設計をされたシステムが増えてくれば、擬人的エージェントの技術も今後適切な展開を見い出せるものと期待される。

スケジュール管理、メール分類、書類の回覧、情報検索といった日常的な秘書業務をエージェントに代行させようとするシステムも、Object Lens[Lai88]をはじめとして製品レベルのものを含めて種々開発されている。これらの普及に伴ってエージェント機能を持つシステムのヒューマンインタフェースに関する議論も深まっていくものと思われる。

エージェントをどう賢くするかも大きな課題である。これには例示プログラミング、すなわちオンザジョブトレーニングを利用するのが一つの案である[Maulsby93]。エージェントに限らずユーザが教えて賢くできるインタフェースはインストラクティブインタフェース (instructive interface) と総称されている。

## 2.1.5 仮想現実感

仮想現実感[Fisher86]もエージェント指向同様、混乱を招きやすい用語である。仮想現実感というとヘッドマウントディスプレイを着け、データグローブをはめた姿を連想させる各種マスメディアによる記事が多いのが誤解を与える一因であろう。そこで概念の抽象性に重きをおいて、Virtual EnvironmentやVirtual WorldあるいはVirtual Workspaceといった用語を好む研究者も増えているようである。この観点からいえば、現在のデスクトップメタファも立派なVirtual Worldでの直接操作インタフェースといえるであろうし、先に挙げたGUIDESもVirtual Worldの直接操作型でないアクセスインタフェースの例となっている。

さて仮想的なオブジェクトの直接操作技術についてみると、2次元についてはマウス等の入力デバイスについての評価[Card90]を除いて新しい動きは少ない。3次元については立体視を用いない普通

のマウススペースのもの[Houde92]も新味は少ないかも知れないがコスト面からは重要である。同様の考え方で、さらに普通のマウスの両端にローラーを付けたものによる3次元オブジェクトの直接操作法も提案されている[Venolia93]。

出力側についてみると、現在のヘッドマウントディスプレイの問題点は価格と解像度、目に対するインパクト等である。これらを解決する現実的アイデアとして、普通のグラフィックワークステーションのモニタの画面をヘッドポジションの検出機構と連動させて動かす方式[Ware93]は面白い。また仮想現実感応用プログラムを簡単に作成するためのツール[Shaw92]や、Virtual Worldの持つべきソフトウェアアーキテクチャ[Lewis91]についても今後研究が盛んになると予想される。

### 2.1.6 実世界指向インタフェース

ここまで見てきたGUI、エージェント、仮想現実感を一言でまとめるとすると、それは電子的仮想世界指向インタフェースであると考えられる。その意味でこれらは同一の進化の線上にあるといえよう。ひるがえって人間の日々の活動を見てみると、それらの多くはこのような仮想世界と非電子的現実世界にまたがって行われるようになっている。文書のハンドリングなどはその典型で、多くの資料はワープロ等で作成されてハードディスクに収まっている一方で、同一文書が紙に印刷されて実世界で流通している。言い換えると、概念的には同じものが実世界のオブジェクトとしても存在するし、仮想世界のオブジェクトとしても存在しているわけである。以下ここでは、実世界と仮想世界の両方から成り立っているという意味で現代社会を「複素世界」と呼び、先にあげた文書のように両世界にまたがる存在を複素オブジェクトと呼ぶことにする。

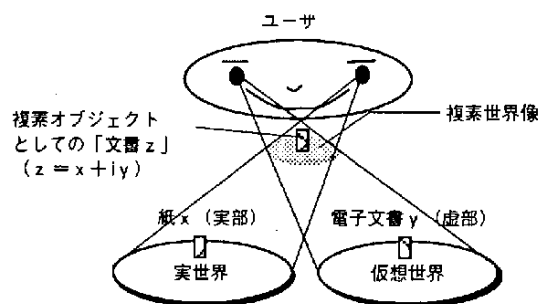


図2.1-1 現代社会の複素世界化

現代社会において必要なインタラクションスタイルは実世界を中心として考えるべきであるとする研究コンセプトの代表例は、ユビキタスコンピューティング (Ubiquitous Computing) [Weiser91]である。ユビキタスコンピューティングの流れの中での具体的な研究例は、複素オブジェクトの実現とそ

れに対するアクセス法の観点から次のように分類できる。

### (1) Ubiquitous CPU

ポピュラーな実世界オブジェクトを「CPU付き」にして仮想世界の機能を持たせて複素オブジェクト化をはかるアプローチである。いわゆる「電子〇×」（電子手帳、電子ブック、電子黒板、etc.）がこれに当たる。ユビキタスコンピューティングのなかで開発されているTab、Pad、Liveboard[Weiser93]等はこの好例といえる。低消費電力MPU、液晶、電池等の技術進歩により、大きな市場を開くと期待されているPDA (persona digital assistant)もこの流れの中で捉えることができよう。

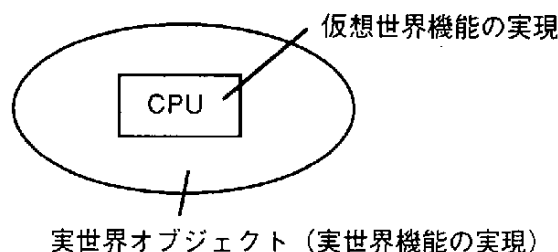


図2.1-2 Ubiquitous CPUによる複素オブジェクトの実現

### (2) ID-aware

実世界オブジェクトに、仮想世界から識別容易なID情報をタグとして付けることにより、実世界と仮想世界の連携、すなわち複素化を実現しようというものである。実世界の物にバーコードを付けて物流等をコンピュータから管理することは既に広く行われている。本アプローチは、これを非接触読み取りや無線通信等の技術を使って高度化を図るとともに、オフィスや家庭に広げようとの狙いを持つと考えられる。Active Budge[Weiser91]もその典型といえよう。

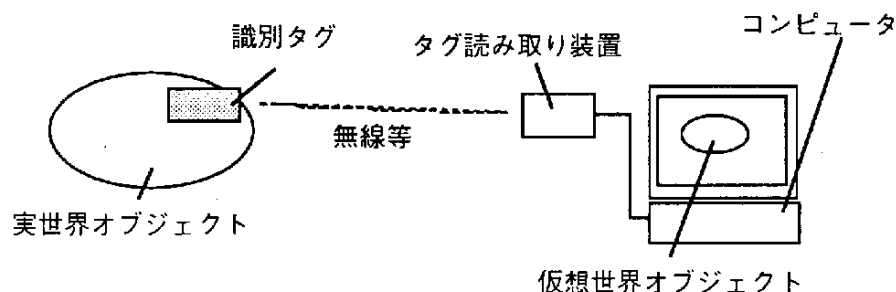


図2.1-3 ID-awareによる複素オブジェクトの実現

### (3) Location-aware

実世界オブジェクト（実部）と仮想世界オブジェクト（虚部）の対応を空間の位置座標でとろうとするアプローチである。近年の磁歪による3次元位置検出やGPS、ジャイロ、マップマッチングといったナビゲーション技術の進歩により有望と期待される方法である。Spatially-aware Palmtop[Fitzmaurice93]がこの研究の代表といえる。

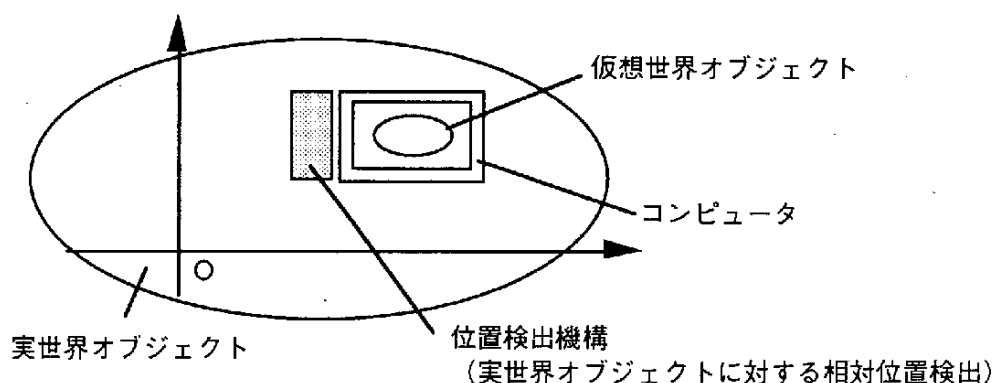


図2.14 Location-awareによる複素オブジェクトの実現

### (4) Overlay

この方法は、実世界のオブジェクトと複素世界のオブジェクトを画像的に重畳して複素オブジェクトとして合成しようというものである。ビジュアルインタフェースの観点からも興味深い。実世界オブジェクトとの対応は画像認識を使うものが基本的であるが、タブレットあるいは前述のLocation-awareと同様の方法をとることも考えられる。この流れの研究としては、オブジェクト指向ビデオ[Tani92]やDigital Desk[Wellner93]が挙げられる。

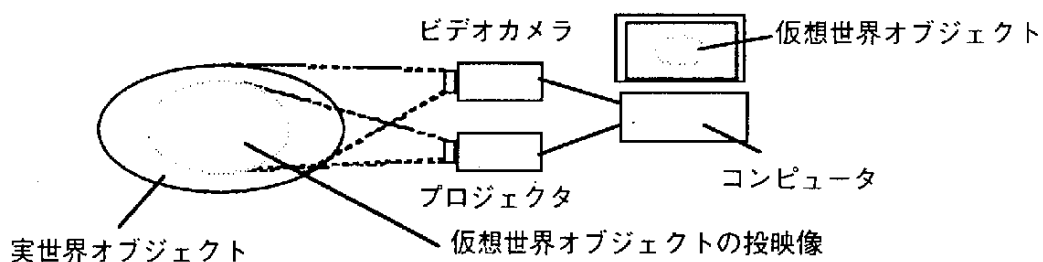


図2.15 Overlayアプローチによる複素オブジェクトの実現

### 2.1.7 おわりに

ヒューマンインタフェースの進化の度合いは、インタフェース開発がどこに焦点をおいて行われて

いるかによって見るができることとする考え方がある[Grudin90]。[Grudin90]はインタフェース開発の焦点は次の5段階をたどって移っていくとしている。

- ① The Interface at the Hardware
- ② The Interface at the Programming Task
- ③ The Interface at the Terminal
- ④ The Interface at the Interaction Dialogue
- ⑤ The Interface at the Work Setting

これから見ると現状は③～④といったフェーズにあり、実世界指向のインタフェースが⑤における試みといえそうである。特に⑤は期待される応用や市場が広範なこともあり、急速な進展が予想される。これまで仮想世界の視き窓といった色彩が強かったビジュアルインタフェース研究も、今後はOverlayアプローチにみられるような複素オブジェクトへのインタラクションという観点の研究加速化が期待される。

### 【参考文献】

- [Apple Computer89]Apple Computer, Inc.: Human interface guidelines: The Apple desktop interface, Cupertino, California (1989).
- [Arai92]Kyoichi Arai, Teruo Yokoyama & Yutaka Matsushita: A window system with leafing through mode :BookWindow, Proceedings of CHI'92, (Monterey,California, May 3-7, 1992), ACM, New York, pp.291-292 (1992).
- [Blattner90]Blattner M.M & Dannenberg R.G.: CHI'90 Workshop on multimedia and multimodal interface design, SIGCHI Bulletin 22,2, pp.54-58 (October 1990).
- [Brennan91]Paul Brennan, Gerhard Deffner, Debbie Lawrence, Monica Marics, Eileen Schwab, & Marita Franzke: Should We or Shouldn't We Use Spoken Commands in Voice Interfaces?, Proceedings of CHI'91, (New Orleans,Louisiana, April 27-May 2, 1991), ACM, New York, pp.369-372 (1991).
- [Brown88]Brown, C. M.: Human-computer interface design guidelines, Norwood, N.J.: Ablex Publishing Corporation (1988).
- [Card90]Stuart K.Card, Jock D.Mackinlay & George F. Robertson: The Design Space of Input Devices: Proceedings of CHI'90, (Seattle,Washington, April 1-5, 1990), ACM, New York, pp.117-124(1990).

- [Card91]Stuart K. Card., George G. Robertson. & Jock D. Mackinlay: The information visualizer, an information workspace, Proceedings of CHI'91, (New Orleans,Louisiana, April 27-May 2, 1991), ACM, New York, pp.181-188(1991).
- [de Baar92]Dennis J.M.J. de Baar, James D.Foley & Kevin E.Mullet: Coupling application design and user interface design, Proceedings of CHI'92, (Monterey,California, May 3-7, 1992), ACM, New York, pp.259-266 (1992).
- [Don92]Abbe Don, Susan Brennan, Brenda Laurel & Ben Shneiderman: Anthropomorphism: From Eliza to Terminator 2, Proceedings of CHI'92, (Monterey,California, May 3-7, 1992), ACM, New York, pp.67-70 (1992).
- [Don93]Abbe Don, Tim Lauren, & Brenda Laurel: GUIDES 3.0., Proceedings of CHI'93, (Amsterdam,The Netherlands, April 24-29, 1993), ACM, New York, pp.447-448 (1993).
- [EAGER91]Allen Cypher: EAGER: Programming repetitive tasks by example, Proceedings of CHI'91, (New Orleans,Louisiana, April 27-May 2, 1991), ACM, New York, pp.33-39 (1991).
- [Fisher86]Fisher, S. S., McGreevy, M., Humphries, J. & W. Robinett: Virtual Environment Display System, Proceedings of the 1986 Workshop on Interactive 3-D Graphics (Chapel Hill, North Carolina) (1986).
- [Fitzmaurice93]George W. Fitzmaurice: Situated Information Spaces and Spatially Aware Palmtop Computers, Communication of the ACM, Vol.36, No.7, pp.39-49 (July 1993).
- [Green87]Green, M.: Directions for User Interface Management Systems Research, Computer Graphics 21,2, pp.113-116 (April 1987).
- [Grudin90]Jonathan Grudin: The Computer Reaches out : The Historical Continuity of Interface Design, Proceedings of CHI'90, (Seattle,Washington, April 1-5, 1990), ACM, New York, pp.261-268 (1990).
- [Henderson86]Henderson, D.A., Jr. & Card, S.K.: Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface, ACM Transactions on Graphics 5(3,July 1986), pp.211-243(1986).
- [Houde92]Stephanie Houde: Iterative Design of an Interface for Easy 3-D Direct Manipulation, Proceedings of CHI'92, (Monterey,California, May 3-7, 1992), ACM, New York, pp.135-142 (1992).
- [Janssen93]Christian Janssen, Anette Weisbecker, Jurgen Ziegler: Generating User Interface from Data Models and Dialogue Net Specifications, Proceedings of CHI'93, (Amsterdam,The Netherlands, April 24-29, 1993), ACM, New York, pp.418-423 (1993).
- [Johnson90]Jeff Johnson, Pelle Ehn, Jonathan Grudin, Bonnie Nardi, Kari Thoresen, Lucy Suchman: Participatory Design of Computer Systems, Proceedings of CHI'90, (Seattle,Washington, April 1-5,

- 1990), ACM, New York, pp.141-144 (1990).
- [Johnson92]Jeff Johnson, Selectors: Going Beyond User-Interface Widgets, Proceedings of CHI'92, (Monterey, California, May 3-7, 1992), ACM, New York, pp.273-279 (1992).
- [Kojima89]Kojima, K., Matsuda, Y. & Futatsugi, S.: LIVE - Integrating Visual and Textual Programming Paradigms -, Proceedings of 1989 IEEE Workshop on Visual Languages, (Rome, Italy, October 4-6, 1989), IEEE, Washington, pp.80-85 (1989).
- [Lai88]Lai, K.Y., Malone, T., Yu, K.C.: Object Lens: A spreadsheet for cooperative work, ACM Transactions on Office Information Systems (October 1988).
- [Lewis90]Lewis, C., Polson, P., Wharton, C. & Rieman, J.: Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces, Proceedings of CHI'90, (Seattle, Washington, April 1-5, 1990), ACM, New York, pp.235-242 (1990).
- [Lewis91]J.Brayan Lewis, Lawrence Koved & Daniel T. Ling: Dialogue Structures of Virtual Worlds, Proceedings of CHI'91, (New Orleans, Louisiana, April 27-May 2, 1991), ACM, New York, pp.131-136 (1991).
- [Maulsby93]David Maulsby, Sau Greenberg & Richard Mander: Prototyping an Intelligent Agent through Wizard of Oz, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp. 277-284 (1993).
- [Mulligan91]Robert M. Mulligan, Mark W. Altom, & David K. Simkin: User interface design in the trenches: some tips on shooting from the hip, Proceedings of CHI'91, (New Orleans, Louisiana, April 27-May 2, 1991), ACM, New York, pp.232-236 (1991).
- [Mulligan92]Robert M. Mulligan, Mary Dieli, Jakob Nielsen, Steven Poltrock, Daniel Rosenberg, & Susan Ehrlich Rudman: Designing usable systems under real-world constraints: a practitioners forum, Proceedings of CHI'92, (Monterey, California, May 3-7, 1992), ACM, New York, pp.149-152 (1992).
- [Myers88]Myers, B.: Creating User Interface by Demonstration. Academic Press, San Diego, Calif., (1988).
- [Nardi93]Bonnie A. Nardi, Heinrich Schwartz, Allan Kuchinsky, Robert Leichner, Steve Whittaker, & Robert Scabassi: Turning Away from Talking Heads: The Use of Video-as-Data in Neurosurgery, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.327-334 (1993).
- [Nielsen90]Nielsen, J., & Molich R.: Heuristic evaluation of user interfaces, Proceedings of CHI'90, (Seattle, Washington, April 1-5, 1990), ACM, New York, pp.149-256 (1990).
- [Nye92]Adrian Nye & Tim O'Reilly: X Toolkit Intrinsics Programming Manual OSF/Motif 1.2

- Edition, O'Reilly & Associates, Inc. (1992).
- [Shaw92] Chris Shaw, Jiandong Liang, Mark Green & Yunqi Sun: The Decoupled Simulation Model of Virtual Reality Systems, Proceedings of CHI'92, (Monterey, California, May 3-7, 1992), ACM, New York, pp.321-328 (1992).
- [Staples93] Loretta Staples: Representation in Virtual Space: Visual Convention in the Graphical User Interface, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.348-354 (1993).
- [Stifelman93] Lisa J. Stifelman, Barry Arons, Chris Schmandt & Eric A. Hulteen: VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.179-186 (1993).
- [Sukaviriya93] Piyawadee "Noi" Sukaviriya, James D. Foley, Todd Griffith: A Second Generation User Interface Design Environment: The Model and The Runtime Architecture, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.375-382 (1993).
- [Takeuchi93] Akikazu Takeuchi & Katashi Nagao: Communicative Facial Displays as a New Conversational Modality, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.187-193 (1993).
- [Tang90] John C. Tang & Scott L. Minneman: VideoDraw: A video interface for collaborative drawing, Proceedings of CHI'90, (Seattle, Washington, April 1-5, 1990), ACM, New York, pp.313-320 (1990).
- [Tani92] Tani, M., Yamaashi, K., Tanikoshi, K. & Futakawa, M.: Object-Oriented video: Interaction with real-world objects through live video, Proceedings of CHI'92, (Monterey, California, May 3-7, 1992), ACM, New York, pp.593-598 (1992).
- [Ueda93] Hirotada Ueda, Takafumi Miyatake, Shigeo Sumino & Akio Nagasaka: Automatic Structure Visualization for Video Editing, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.137-141 (1993).
- [Venolia93] Dan Venolia: Facile 3D Direct Manipulation, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.31-36 (1993).
- [Ware93] Colin Ware & Kevin Arthur and Kellogg S. Booth: Fish Tank Virtual Reality, Proceedings of CHI'93, (Amsterdam, The Netherlands, April 24-29, 1993), ACM, New York, pp.37-42 (1993).
- [Weiser91] Mark Weiser: The Computer for the 21st Century, Scientific American. September, pp.66-75 (1991).
- [Weiser93] Mark Weiser: Some Computer Science Issues in Ubiquitous Computing, Communication of the ACM, Vol.36, No.7, pp.75-84 (July 1993).

[Wellner93]Pierre Wellner: Interacing with Paper on the DigitalDesk, Communication of the ACM,  
Vol.36, No.7, pp.87-96 (July 1993).

## 2.2 3次元を用いたビジュアルインタフェースの研究動向

### 2.2.1 はじめに

ハードウェアの進歩に伴い、3次元物体を実時間で対話的に操作する環境が比較的安価に手に入るようになってきた。つい数年前までは数千万円という価格が付けられていたグラフィックスワークステーションの世界にも、続々と普及機種が投入されている。その一方で、ハイエンド機の世界では、並列アーキテクチャを取り入れたものなど、スーパーコンピュータに迫る性能を誇るものが登場している。

また、各社から発表が相次いでいる「次世代ゲームマシン」では、いずれも3次元処理の高速化に焦点を当てている。数年後には、現在の高機能グラフィックスワークステーションと本質的には同等の機能が家庭で利用できるようになるだろう。

こういったハードウェアの進歩・低価格化と呼応して、今までは非常に限られた分野でのみ利用されていた対話型3次元グラフィックスを、一般的なヒューマンコンピュータインタラクションの素材として利用しようという機運が高まってきている。人間は3次元空間で生活しており、人類の長い進化の過程によって3次元空間の構造を直観的に把握する極めてすぐれた能力を体得している。この能力を人間・コンピュータ間の対話の場に引き出すことができれば、従来とは比較にならない程効率のよい情報交換が可能になるだろう。

ここでは、このように急速に発展しつつある3次元ビジュアルインタフェースの技術動向を、提示技術、対話技術、開発環境、そしてアプリケーションの4項目に分けて解説する。

### 2.2.2 提示技術

人間は3次元空間を様々な要因から得られる情報を統合して感じとっている。例えば、物体の位置関係 (occlusion)、陰影、明るさ、物体の見かけ上のサイズ、透視図法、動的視差 (motion parallax)、静的視差 (ステレオ視)、焦点の調整などの要因がある。コンピュータを利用して3次元情報を提示する場合も、これらの要素を組合せて利用するのが常である。ここでは、静的視差を得るためのステレオ画像と、動的視差を得るためのヘッドトラッキングについて解説する。なお、立体表示に関するより網羅的なサーベイとしては、例えば[McKenna92]や[Aukstakalnis92]があるので参照されたい。

#### (1) ステレオ画像

左右の目にそれぞれの視点からの画像を表示して立体感を得るこの方法は、コンピュータの登場以前から広く利用されていた。コンピュータの画像出力をステレオ映像として提示するには、

- 左右で別々のディスプレイを使う (Head Mounted Displayなど)
- 左右の画像を別の色 (赤と青) で表示し、それを色付きガラス等で分離する

- 左右の画像を併置して、仕切板や鏡を使って分離する
- 左右の画像を時分割して表示し、液晶シャッタによって左右の目に振り分ける
- その他の光学系（レンティキュラプリズムなど）を用いる

などの方式が現在利用されている。最も一般的に用いられているのは液晶シャッタによるもので、ディスプレイに左右の画像を高速に切り替えながら表示し、それと同期した液晶シャッタ（ディスプレイと眼鏡）によってそれぞれの画像を左右の目に分離して送り込む方式であり、CrystalEyes[CrystalEyes93]などの製品例がある（図2.2-1）。最近のグラフィックスワークステーション（例えばIRISIndyなど）には、液晶シャッタを制御するためのポートが標準で装備されている場合がある。

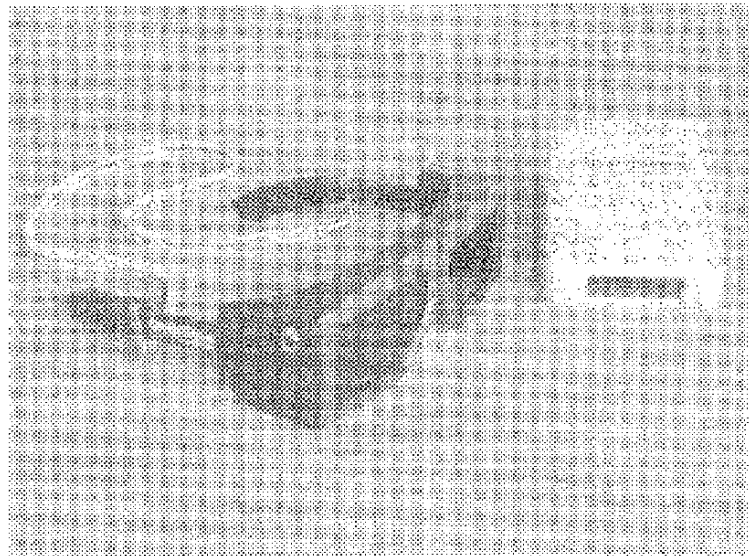


図2.2-1 液晶シャッタによる立体視用グラス（CrystalEyes）（出典：[CrystalEyes93]）

レンティキュラを用いる方式は、観測者が何も装着する必要がない点で優れているが、観測する位置が固定されてしまう、ディスプレイに表示される画像と光学系との位置合わせが厳密でなければならないなどの問題により、液晶シャッタ方式ほどは普及していない。

## (2) ヘッドトラッキング

一方、観測者の頭の位置と方向を計測し、それに基づいて画像を更新していくと、motion parallaxによる強い立体感を得ることができる。この方式は、視点位置による画像の高速な更新が必要とされるため、ごく最近まではほとんど利用されることがなかった。

画像を表示する方法は、HMD（Head Mounted Display）によるもの、通常のディスプレイによる

ものが考えられる。どちらもステレオ画像との併用が可能である。後者は、ディスプレイを水槽の壁面に見立てると、ちょうど水槽のガラス越しに中の物体を覗き込むような感覚が得られる。そのため、この方式のことを特にFish Tank型と呼ぶことがある。

頭の位置・方向をセンスする方法としては、メカニカル方式、超音波、磁気、光学方式が主に用いられている（より詳しいサーベイとしては、例えば[Meyer92,Applewhite92]がある）。

メカニカル方式は、頭に多関節からなるロッドを取り付けて、各関節の角度から頭部の位置を求める方式である。ステレオディスプレイと組合せたFakespace[Fakespace93]や、ポジショントラッカのみからなるADL-1[ADL93]などの製品例がある（図2.2-2）。

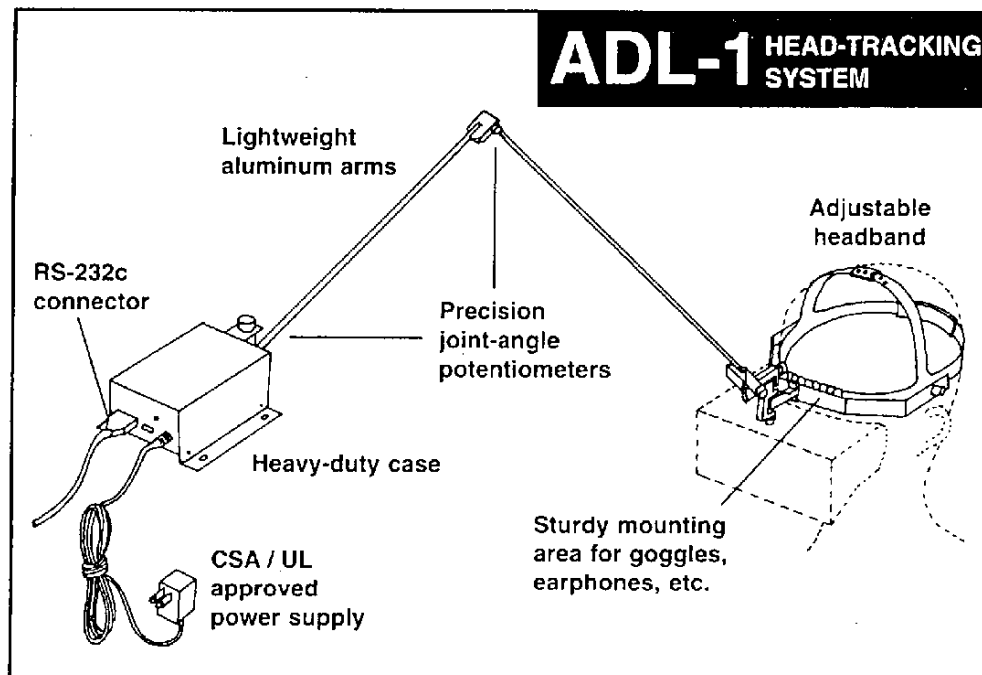


図2.2-2 メカニカル方式によるヘッドトラッキングの例（ADL-1）（出典：[ADL93]）

超音波方式では、超音波の到達時間によって位置を判定する。頭部に超音波の発信器を取り付け、ディスプレイの上などに受信器を設置する。精度はあまりよくないが、比較的安価であることから、液晶シャッターによる両眼視と組合せて使われる場合が多い。

磁気センサ方式は、装置の作り出す磁界のなかにコイルを置き、その誘導電流によって位置および方向を判定する方式である。Polhemus社のISOTRAK[ISOTRAK87]やAscension社のBirdが代表的で、多くの仮想現実感（VR）システムで利用されている。長所はトラッキングの精度が高いことで、その反面、トラックできるレンジが狭い、コンピュータが発生する磁気などの影響を受けやすいといった問題がある。

光学方式は、光源を可動部側に持つか固定部側に持つかでさらに分類できる。前者は、頭部に赤外線LED（複数）を取り付け、固定されたカメラでその画像を認識する。複数のカメラによる情報から、三角測量の原理で光源の位置を判定することができる。光源を複数個にすることで、頭部の方向も判定可能になる。後者の方式の例としては、ノースカロライナ大の光学式ポジショントラッカがある[Wang90]。これは頭に搭載したカメラによって天井に取り付けられた発光体（赤外線LED）を画像認識し、頭部の位置・方向を計測するシステムである（図2.2-3）。SIGGRAPH91でデモされた緒元によると、位置精度2mm、方向精度0.2度、トラッキングレート20～100Hz、遅延時間20～60msecの性能を達成している。

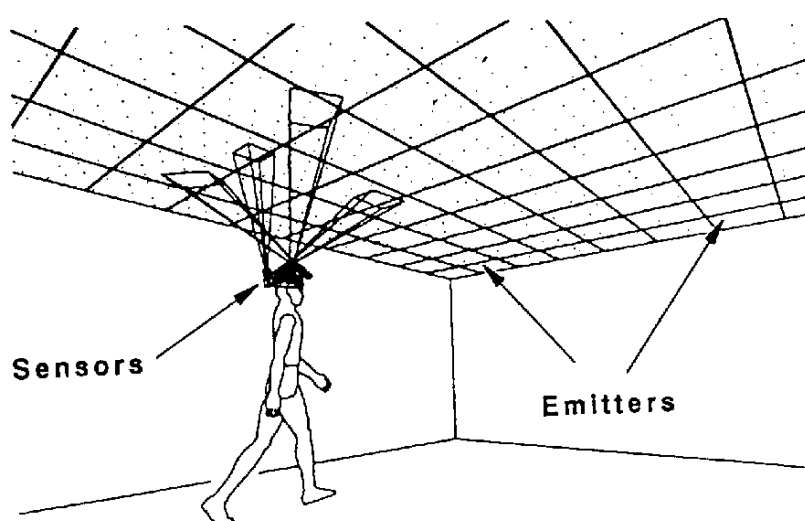


図2.2-3 光学式ヘッドトラッキング（ノースカロライナ大）（出典：[Wang90]）

### (3) ステレオ画像とヘッドトラッキングの比較

カナダのブリティッシュコロンビア大学で、Arthurらによるステレオ画像とヘッドトラッキングの優劣に関する興味深い実験が行われている[Arthur93]。この実験では、以下の5通りの3D表示方式を7名の被験者によって比較している。

1. Picture — （非ステレオ）画像
2. Stereo only — ステレオ画像
3. HC mono — ヘッドトラッキングによる画像を単眼で観測
4. HC binocular — ヘッドトラッキングによる画像を両眼で観測
5. HC & stereo — ヘッドトラッキング付きステレオ画像

実験内容は①被験者による主観的評価と、②タスクを被験者が実行する際の達成時間と正確さの評価の二つであった。結果は表2.2-1および表2.2-2のようである。

表2.2-1 各3D表示方式の主観評価（出典：[Arthur93]）

|                | 1    | 2    | 3   | 4   | 5   |
|----------------|------|------|-----|-----|-----|
| 1 Picture      |      | 43%  | 4%  | 0%  | 7%  |
| 2 Stereo only  | 57%  |      | 7%  | 11% | 0%  |
| 3 HC mono      | 96%  | 93%  |     | 29% | 61% |
| 4 HC binocular | 100% | 89%  | 71% |     | 68% |
| 5 HC & stereo  | 93%  | 100% | 39% | 32% |     |

表2.2-1は主観評価の結果で、それぞれの方式が、他方式と比較して好ましいと答えた被験者の比率を示している。これによると、ヘッドトラッキングに対する評価が高いことが分かる。単純なステレオは非ステレオ画像と比較しても顕著な差はなく、非ステレオヘッドトラッキング画像（表の1と2）と比較すると、圧倒的にヘッドトラッキングを支持する被験者が多かった（91%対9%）。

表2.2-2 3D表示方式とタスク実行時間・エラー率（出典：[Arthur93]）

|                | 時間(秒) | エラー率(%) |
|----------------|-------|---------|
| 1 Picture      | 7.50  | 21.8    |
| 2 Stereo only  | 8.09  | 14.7    |
| 3 HC mono      | 8.66  | 3.7     |
| 4 HC binocular | 9.12  | 2.7     |
| 5 HC & stereo  | 6.83  | 1.3     |

表2.2-2は3次元の木の枝をたどるというタスクを被験者に与え、達成時間とエラー率を計測した結果である。これによると、タスク達成時間に大きな差はない（それでもステレオとヘッドトラッキングを併用した場合が最も短い）が、エラー率に顕著な差が見られる。

両実験の示唆するところは興味深く、立体表示には従来の常識のようにステレオ画像は必ずしも必須ではなく、むしろ視点による画面の更新が効果的であるということがいえる。特に3次元情報を使ったビジュアルインタフェースを実現する立場では、ヘッドトラッキング方式は、通常のディスプレイを使ったFishTank型のシステム構成で、しかも特別な眼鏡を装着せずに利用できるなど、有利な点が多い。

### 2.2.3 対話技法

ここでは、3次元空間中のオブジェクトを操作するための基本的な技法について述べる。

#### (1) オブジェクトの位置・方向の指定

3次元空間中で、オブジェクトの位置や方向を指定することは、3次元ビジュアルインタフェースの

最も基本的なインタラクションである。マウスを使った指示など、2次元の延長として捉えられる部分もあるが、3次元の対話技法には、2次元インタフェースと比較して、以下の二つの問題がある。

まず、3次元物体の位置や方向を完全に指定するためには、位置に関して3自由度 ( $x, y, z$ )、方向に関して3自由度 ( $x$ 軸、 $y$ 軸、 $z$ 軸に関する回転) の計6自由度が必要である。一方、マウスやタブレットは水平位置に関する2自由度の情報しか取り出せない。したがって何らかの工夫を施さない限り、そのままでは3次元物体を操作することはできない。

また、単に3次元シーンを表示した画面では、深さ方向の位置を視覚的に正確に認識することは困難である。したがって、前述したようなヘッドトラッキングやステレオ画像によって立体感を増すか、別の技法によって曖昧性を解消する工夫が必要である。

そこで、現状では、以下のような技法、あるいはその組合せを用いている場合が多い。

- それぞれのスカラー量を制御する対話手段を提供する
- 6自由度が採取できる入力デバイスを使う
- 3自由度程度の装置を利用する
- 3次元物体操作用の対話技法を考案する

#### (a) それぞれのスカラー量を制御する

例えばスクロールバーやスライダーのような2次元対話技法によって、位置や方向の各要素を独立して制御する方法である。現状のシステムでは、依然としてこの方式に頼っているものが多い。この方式は、各要素の値を精密に設定できる長所はあるが、操作の結果が直観的に把握しにくい、操作を各要素の値に分解してやらなければならない、といった問題があるのは明らかだろう。特に、オブジェクトの回転を三つの回転角に分解する作業は自明ではなく、ユーザに試行錯誤を強いる結果となる。

#### (b) 6自由度デバイスによる操作

これはヘッドトラッキングのところで述べたような3次元計測技術を入力手段に用いるものである。ヘッドトラッキングと同様、メカニカル・超音波・磁気・光学などの原理で動作するデバイスが利用可能である。例えばアルバート大のLiangらは、機械系のCADシステムにおいてISOTRAKを用いたユーザインタフェースを開発しており、良好な結果を得ている[Liang93b]。

#### (c) 3自由度デバイスによる操作

3自由度デバイスでは、位置の設定を直接行うことはできるが、位置と方向を同時に制御できないので、何らかの対話技法を工夫する必要がある。一方、ハードウェア的にはマウス+ $\alpha$ 程度のデバイスで済み、コストや従来のマウスからの上位互換性といった点で有利である。

---

## 2. ビジュアルインタフェースの技術動向

例えばApple社のVenoliaは、ローラー付きマウスによって3次元物体の操作を試みている [Venolia93]。このマウスは、ボタンの左右にローラーが配置されていて、前後に回転させることができる。マウス本来の機能はそのままなので、3自由度（マウスの2自由度とローラーの1自由度）の入力装置である（図 2.2.4）。

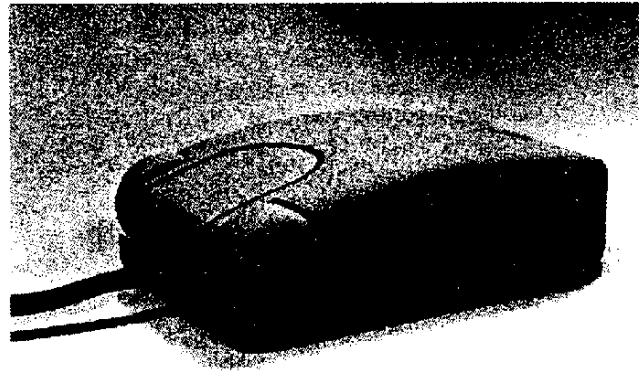


図 2.2.4 3自由度マウス（出典：[Venolia93]）

3次元空間中の位置指定にはこの3自由度をそのまま用いる。ローラーによって深さ方向（画面の手前から奥へ方向）を制御する。カーソル形状が特徴的で、単に3次元空間中の位置を示すだけではなく、方向も表現するためにコーン型の形状を持っている。

また、方向と移動を同時に制御する方法として、Venooliaはtail-draggingという技法を用いている。これは、3次元空間中に浮いている線分の先頭をマウスで曳きずることで、線分全体の位置と方向を制御していく方式である（図2.2-5）。

この方式は、位置と方向の値（この場合は線分に関する回転成分は制御していないので5自由度になる）を3自由度の入力装置で制御していることになる。ある瞬間の値をそのまま用いるのではなく、時間の経過に伴って変化する値の系列から、より多くの情報を取り出している点に特徴があるといえよう。

#### (d) 2自由度デバイスによる操作

通常のマウスによって3次元物体を操作する方法には多くの研究例がある（例えば [Houde92, Snibbe92]）が、GUIにおける図形のダイレクトマニピュレーションのように標準的な手法が確立しているとはいえない。ここではInteractiveShadow と Snap Dragging について紹介する。

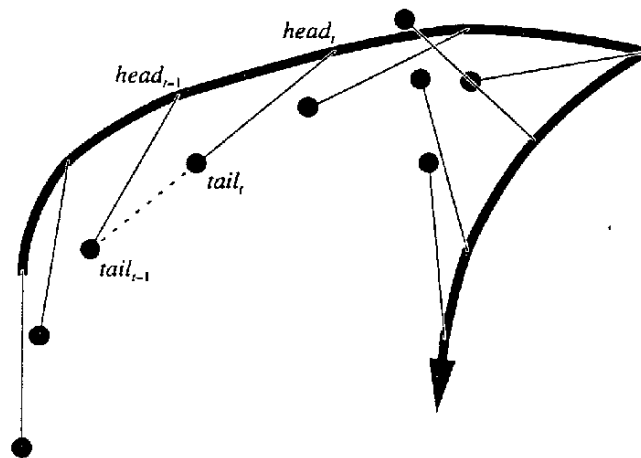


図 2.2-5 Tail-Dragging (出典：[Venolia93])

### (i) Interactive Shadow

Interactive Shadow[Herndon92]は、3次元物体の「影」によってオブジェクトを操作するインタフェースである。通常のディスプレイで3次元情報を表示すると、物体の位置関係を正確に認識することが困難である。例えば空中に浮いた二つのオブジェクトの前後関係は、両者が重なって表示されていない限り簡単に判定することはできない。立体表示の他に、いわゆる3面図を併用することもできるが、複数の表示を概念的に統合するのもユーザの負担になる。

Interactive Shadowでは、3次元図形の「影」を、操作可能なオブジェクトとすることで、これらの問題を解決する方式である(図2.2-6)。影は3次元オブジェクトの複数ビューを統合するための自然なメタファである。例えば影をマウスでドラッグすることができ、これはオブジェクトをある平面に拘束して移動させる操作に相当する。影は2次元平面上のオブジェクトなので、通常のマウス操作がそのまま適用できる。影の変種で「鏡像」を使うことも考えられる。Interactive Shadowでは、オブジェクトの影を操作することで意図的に操作の自由度を下げている。これが(低自由度のデバイスである)マウスでの操作性を向上させているといえよう。

### (ii) Snap-Dragging

XeroxのBierによるSnap-Draggingは、2次元図形エディタでよく見られる「アラインメント」の概念を拡張したものである[Bier90]。3次元空間中にSkitterと呼ばれるガイドをマウスで制御していくが、Skitterの位置は、近くにあるオブジェクトの端点やエッジ、面などからの「重力」によって引き付けられ、拘束を受ける。この処理をスナップ(snapping)と呼ぶ。例えば面にスナップしたSkitterは、その平面内に拘束されて動くようになる。このようなSkitterを用いて、オブジェクトの移動、回転、変形などの処理を行うことができる。Snap-Draggingでは、場面ごとに、操作しているオブジェクト

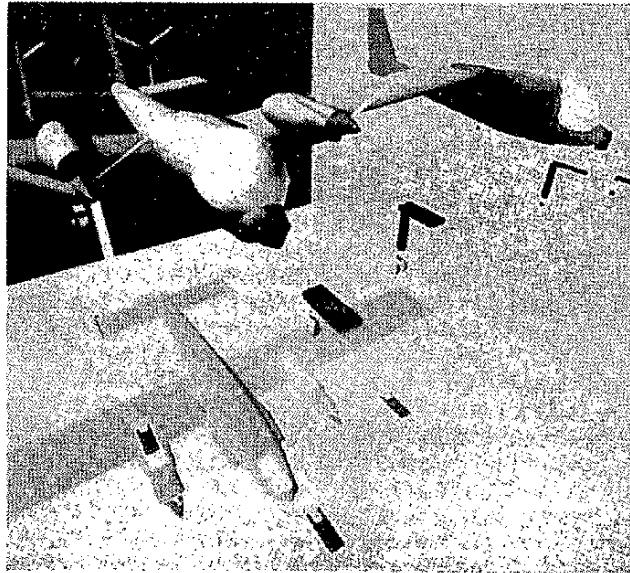


図2.2-6 InteractiveShadow (出典：[Herndon92])

の形状に応じてSkitterの動きを制限していることになる。この制限によって、逆に自由度の低いマウスでも操作できるインタフェースを生み出していると考えられる。

## (2) ナビゲーション

3次元空間中で視点を変更する技法も、3次元ユーザインタフェースの基本となるインタラクションである。視点を完全に設定するには、オブジェクトの移動・回転と同様、6自由度の情報が必要である。オブジェクトの操作と異なり、「視点」は通常画面上での操作対象物としては現われない。ユーザが視野に写し出された情報から、視点の変化を間接的に知ることになる。この違いによって、オブジェクト操作とはまた別種の対話技法が要求される。

### (a) 6自由度デバイス

ISOTRAKのように6自由度の情報が取り出せる入力装置によって、原理的には任意の視点移動・方向づけが可能なのはであるが、実際には以下の理由によってこの方法は必ずしも成功していない。

- 一 現状の入力装置で指定できる位置が限られている。例えばISOTRAKでは装置が作り出す磁界の範囲内でしか機能しない。したがって、扱う空間のサイズがこれよりも大きい場合は、センサの位置と視点の位置を単純に結合することはできない。

ー 6自由度デバイスで指定した位置や方向が、必ずしも画面上の視野に収まらない。自分の背後に移動する場合などは、まず背後が視野に収まるように視線方向を変えなければならない。

後者の問題は、VRのようにHMDの位置と方向をそのまま視点制御に用いれば発生しないが、前者の問題は依然として残る。したがってVRアプリケーションでも、「手で示した方角に移動する」といった別の移動手段を併用するのが通常である。

カナダのニューブラウンズウィック大のWareらは、6自由度デバイスを用いた以下の3種類の移動方法についての比較実験を行っている。

Scene-in-hand： 入力デバイスが世界全体の位置・方向を制御する。

(世界全体を手中に収めたかのようなメタファ)

Eyeball-in-hand： 手の中の入力デバイスが視点の位置・方向に直結している。

(目玉を直接コントロールするようなメタファ)

Flight-vehicle： 入力デバイスを操縦菅に見立て、制御する。

(基準の位置・方向とデバイスの位置・方向の差が、移動や回転の速度を決定する)

Wareらは、7人の被験者に対して、以下の三つの環境を用意し、それぞれの技法を用いた主観評価を行った。

標識： 平面上に置かれた三つの標識。それぞれの片面にはマークが貼ってある。被験者には各標識の間を渡り歩くタスクが与えられる。

立方体： 立方体の3面にマークが貼り付けられている。被験者には各マークを見るタスクが与えられる。

迷路： T字型の迷路。迷路の奥（行き止まり）にはマークが取り付けられている。被験者には迷路の中を歩くタスクが与えられる。

各被験者は-1（悪い）から1（良い）の範囲で各方式を採点する。実験の結果、それぞれの環境に対する被験者の嗜好は表2.2.3のようになった（7人の合計）。

表 2.2.3 ナビゲーション方式の比較

|                 | 標識   | 迷路 | 立方体 |
|-----------------|------|----|-----|
| Scene-in-hand   | +2.0 | -6 | +6  |
| Eyeball-in-hand | -0.5 | 0  | -2  |
| Flight-vehicle  | -1.5 | +6 | -4  |

この結果から、すべての環境で有効な方式は無く、それぞれの得意不得意があることが分かる。例えばFlight-vehicleは迷路の中を航行するには向いているが、立方体の周りを移動するのは不得意であ

る。逆に、Scene-in-handは立方体を回転させるのに適しているが、迷路全体を動かすことを被験者は嫌っている。

実験中の観察された面白い現象として、次のようなエピソードがあった。飛行機の操縦経験がある被験者が、Flight-vehicleの操作に手こずってしまった。この被験者は、回転しながら移動する際に（飛行機のように）全体が傾かないことに違和感を感じていたようである。また、その被験者の身体が、画面の傾斜とは無関係に移動中に傾いてしまうことが観察されたそうである。

#### (b) Point of Interest

Point of Interest 方式（以下POI）は、3次元空間中の視点の位置と方向を設定するための方法で、マウスによって操作する[Mackinlay90]。まず画面上の目的地（これをPOIと呼ぶ）をマウスで指定し、次に移動キー（前・後の2種類がある）を押してPOIに接近（あるいは離脱）していく。画面上の点をマウスで指示すると、（ちょうどレイトレーシングの逆のアルゴリズムで）視点とマウスの点を結ぶ線を延長していった最初に交差したオブジェクト上の点がPOIとなる。移動後の視線の方向は、POIが置かれたオブジェクト表面の法線ベクトルと同じになる（図2.2-7）。

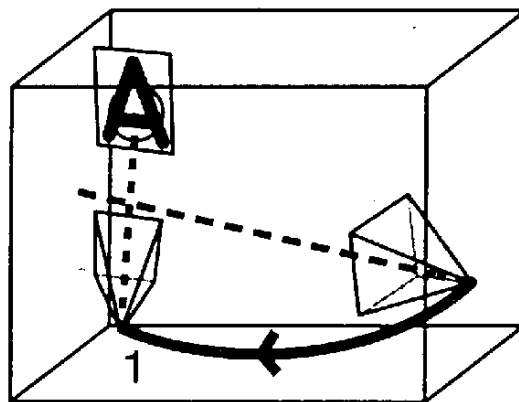


図2.2-7 Point of Interest 法（出典：[Mackinlay90]）

POIで特徴的なのは移動に際しての速度制御である。移動は現在位置から目的地に瞬間的に飛び移るのではなく、計算された速度で航行していくようになっている。画面上では航行中の光景がアニメーションとして表示され、移動前後の位置関係を把握するのが容易になっている。航行速度は、POIと現在位置との距離に関して対数的な関数によって求められる。目的地から離れているときは高速に移動し、目的地に接近するに従って次第にスピードダウンしていく。したがって、長距離の移動が短時間で行え、かつ近距離での精密な移動が可能である。

### (c) 迷子問題

3次元空間内の移動で問題になるのが、自分がどこにいるかが分からなくなってしまうことである。現実世界では、移動速度や重力、壁など種々の制約があり、自分の居場所が唐突に変化することはない。一方、人工的な3次元環境では、移動方法によっては容易にそういった状況が起こり得る。これに対処するために、多くの移動方式では現在位置から目的位置までの移動をスムーズなアニメーションでつなぐようにしている。また、空間全体の中で位置関係を知るための手掛かりになる物体（ランドマーク）を置くことも有効な方法である。情報視覚化など、純粋に抽象的な3次元オブジェクトを操作する場合でも、その周囲に床や壁などのランドマークとして機能するオブジェクトを表示することで位置関係の把握が容易になる。

ジョージワシントン大学のDarkenらは、ランドマークとして3次元音響を用いた試みを行っている[Darken93]。3次元空間中のある1点から音が出るように設定しておき、ユーザはその音をたよりに現在位置や方向を認識する。

### 2.2.4 開発環境

3次元を用いたビジュアルインタフェースの開発は、従来のGUIと比較してもさらに困難なものとなる。例えば、

- 各種のレンダリング技術
- 各種座標系の管理
- 各種の入力装置の制御
- 入力装置を用いた3次元対話技法の実現
- 進歩し続けるハードウェアに呼応した描画性能

などの要素を考慮してアプリケーションを構築することは多くの開発者の手に余る作業である。従来の3次元アプリケーション開発環境は、単なるレンダリングパッケージの水準にとどまり、ユーザとのインタラクションを支援するものではなかった。が、最近になって各種のアプリケーションプラットフォームが開発され、こういった状況は急速に変わりつつある。

3D環境の開発環境の分類についての共通認識はないようであるが、ここでは便宜的に、低レベル・中レベル・高レベルの3層に分けて解説する。

#### (1) 低レベルプラットフォーム

3次元図形の透視変換や、隠面消去、シェーディング、テクスチャマッピングといった基本的なレンダリングをサポートする層で、PHIGSやGLなどがこれに相当する。

PHIGS (Programmer's Hierarchical Interactive Graphics System) はANSIによる3Dグラフィックス

の規格である[PHIGS93]。PHIGSをX-Window経由で利用するためのXの拡張機能がPEXである。

IRIS GL[SGI91]はSilicon Graphics社が自社のグラフィックスワークステーションのために開発したライブラリで、最近になってプラットフォームに対してオープンな拡張規格であるOpenGLが発表されている[OpenGL93]。OpenGLは、IRIS以外でも例えばSunワークステーションやNEC EWSなどで利用できる共通規格である。またMicrosoftのWindows-NTでのサポートが表明されている。

一方、トロント大のStampeらによって開発されたIBM-PC用の3次元グラフィックス用ライブラリとしてREND386がある。これはIBM-PCに特化したインプリメンテーションであるため、PCクラスのマシンでもかなりの速度を得ている。REND386はソースコードを含め無償で公開されているフリーソフトウェアでもある。

## (2) 中レベルプラットフォーム

PHIGSやGL等のグラフィックスライブラリは、3次元オブジェクトを表示するのに、視点や光源の設定等、数多くの属性を設定する必要がある。また、3次元オブジェクトを対話的に操作する処理は、すべてアプリケーションの責任で記述しなければならなかった。複雑なオブジェクトをマウスでピックアップして移動させたりするインタラクションをアプリケーションプログラムで記述するのは非常に困難な作業である。そこで、これらの処理を部品化してより簡易な記述で3次元インタフェースを構築するための層（GUIにおけるツールキットに相当する）がここである中レベルプラットフォームである。

### (a) IRIS Inventor

SGIのIRIS Inventor[Strauss92,Inventor92]は、GLライブラリの上位に位置するオブジェクト指向3Dツールキットである。表示対象、光源、各種表示属性、座標変換、操作手段（Manipulator）などをオブジェクトとして扱うことができる。例えば、一つの3次元空間を想定すると、それは複数の光源や複数の3Dオブジェクトから構成される。3Dオブジェクトはさらに下位の3Dオブジェクトをグループ化して構成することもできる。

Inventorで特徴的なのはManipulatorオブジェクトで、3次元オブジェクトをマウスで操作するための種々の技法もオブジェクトとして部品化されている。Manipulatorには、例えばcylinder（シリンダ型回転オブジェクト）、directional light（向きを変更できる光源）、handle box（移動・スケーリングのための長方形オブジェクト）、drag 1D/2D（線上または平面上に沿ってオブジェクトを移動する）、あるいはperspective camera（編集可能なカメラ）といったものが含まれる（図2.2-8）。

### (b) 3D-Widget

ブラウン大のZelevnikは、3D-Widgetをプリミティブな部品から組み上げていく手法を提案してい

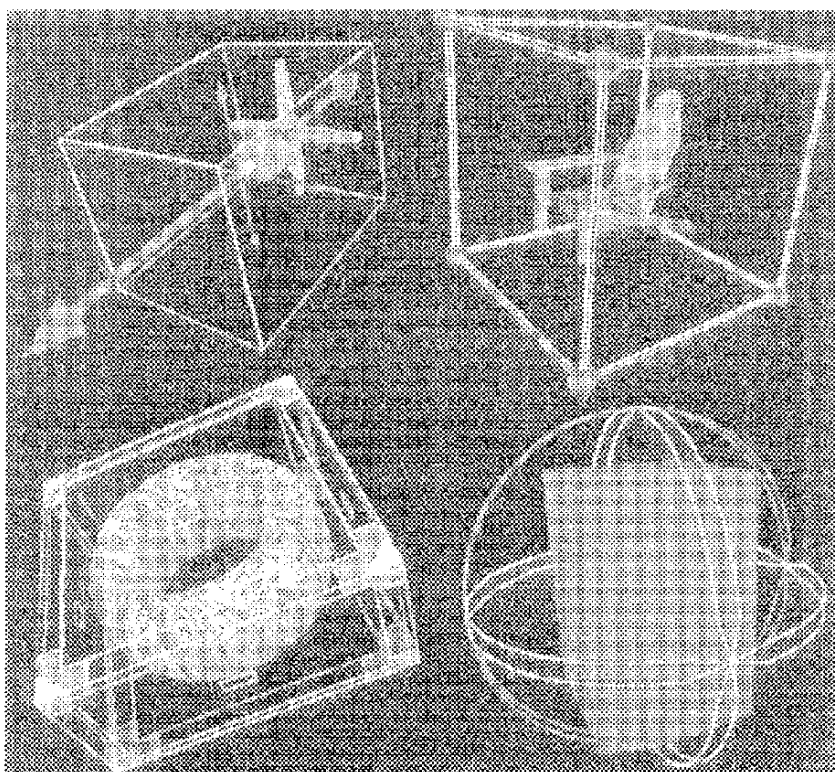


図2.2-8 Inventorのマニピュレータ (出典：[Inventor92])

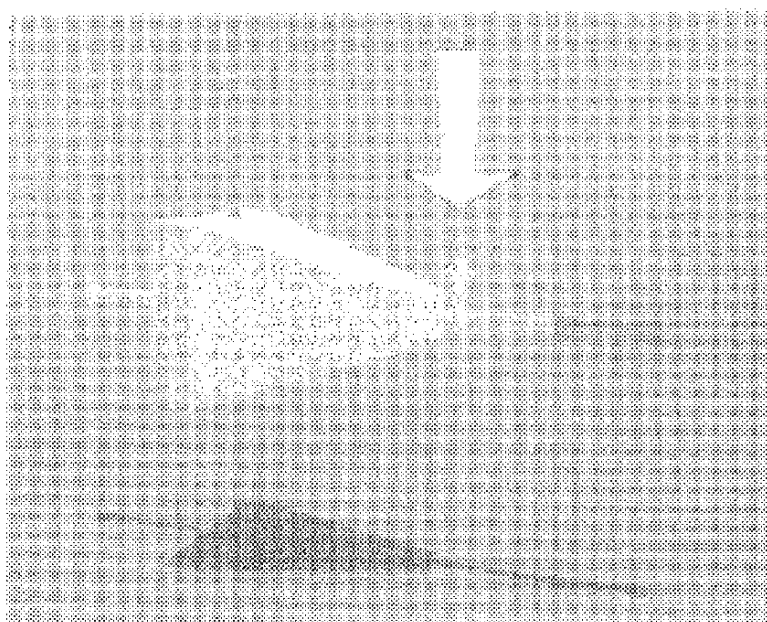


図2.2-9 ブラウン大の3D-Widget (出典：[Zelevnik93])

る[Zelevnik93]。ここでいう3D-Widgetとはマウスで操作できる3次元オブジェクトで、特定の平面に沿って動く「点」のような単純なものから、別のオブジェクトを変形させるためのハンドル、さらには先に述べたInteractive Shadowのように複雑なものまで含まれる。

Zelevnikの手法は、プリミティブ部品（Point, Ray, Planeなど）間に制約を与えて組合せていくことで、複雑なWidgetを構成していく。例えばPointをPlaneに接続すると、ある平面に拘束されて移動可能な点となる。また、特殊な（幾何的でない）制約条件を与えるためにBlackBoxという部品も用意されている。このような手法の繰り返しで、例えば図2.2-9のようなWidgetを構成することができる。

### (3) 高レベルプラットフォーム

#### (a) MR Toolkit

アルバタ大のGreenらによって開発されているMR Toolkit[Green92]は仮想現実感（VR）を用いたアプリケーションを構築するためのツールキットである。データグローブやHMDなど各種のVRデバイスをコントロールするためのサーバ群、レンダリングを担当するプロセス（両眼用に二つ）、そしてアプリケーションの計算を担当するプロセスからなる。これらのプロセスは、互いにプロセス間通信によって情報を交換する。

MR Toolkitで特徴的なのはレンダリングプロセスと計算プロセスの分離である[Shaw93]。例えば流体シミュレーションを考えると、流れそのものをシミュレーションとして計算する処理は、その結果をポリゴンの集合として視覚化する処理よりも通常は重い計算である。この両者を一つの対話ループとして構成してしまうと計算処理の負荷によって対話性が損なわれてしまう。MRではレンダリングと計算処理を別プロセスとして構成し、両者を共有メモリで接続している。したがって例えばシミュレーションの更新が毎秒1回程度しか行えない場合でも、レンダリングプロセスの処理負荷のみに応じた速度でユーザと対話することができる。

MR Toolkitのさらに高位に位置する層としてOML（Object Modeling Language）がある。これは3次元空間中に自律的に存在するオブジェクトを定義するためのプログラミング言語で、オブジェクトのジオメトリ構成と外界からの刺激に反応する行動ルールを記述することができる。例えば「昆虫」というオブジェクトは、体と足6本からなり（ジオメトリ）、通常は床の上をランダムに歩き回っている（行動ルール）、壁や他のオブジェクトに激突した場合には、ひっくり返って一瞬気を失ったのち、また通常の行動パターンに復帰する（刺激に対する反応）、といった記述によって定義可能である。

#### (b) WorldToolkit

Sense8社のWorldToolkit[Sense8]もVRアプリケーション構築用のプラットフォームで、IBM-PCや各種のUnixワークステーション上で動作する。仮想世界の各種エンティティ（3Dオブジェクト、セ

ンサ、光源、視点など）をオブジェクトとして扱うことができる。MRのような分散プロセス構成は取っておらず、すべての機能がライブラリとして提供されている。

### (c) その他

電通大の小池らによるVOGUE[Koike93]は、3次元情報視覚化アプリケーション向けのプラットフォームで、各種3次元レイアウトアルゴリズムを部品化している。三菱の朝日らによるMECOT[Asahi94]は、3次元オブジェクトを使ったUIMSである。

## 2.2.5 アプリケーション

### (1) CAD

3次元物体を設計・加工するインタフェースとして3次元を用いることの有効性は論を待たないだろう。最近の動向としては、透視図と3面図を「影」のメタファで有機的に結合したもの[Herndon92]、6自由度入力デバイスによる設計の容易さを狙ったもの[Liang93a]などが注目できる。

### (2) シミュレーション

シミュレーションでは現実世界の一部をコンピュータの中で再現することが本質である。現実が3次元の世界であることから、シミュレーションの実現に3次元ビジュアルインタフェースを採用するのは極めて自然であろう。

フライトシミュレータはもっとも古くから利用されている3次元シミュレーションであるが、最近のハードウェアの進化、低価格化に伴って、シミュレーションの応用分野も広がってきている。例えば、風洞実験のシミュレーション[Bryson92]、建造物のウォークスルー[Brooks86]、地形データから再構成された仮想火星表面上の飛行などの事例がある。

### (3) 情報視覚化 (Information Visualization)

3次元情報視覚化は、3次元の表現を使って複雑あるいは巨大な情報を、人間が直観的に把握できる表現形態としてユーザに提示することを目的としている。対話型のコンピュータグラフィックスが一般のシステムでも利用できるようになってきたのと呼応して、近年盛んに研究されている分野である。

情報視覚化と、従来からあるサイエンティフィックビジュアライゼーションとの違いを列挙すると、以下になるだろう。

- サイエンティフィックビジュアライゼーションが（連続的な）多次元の数値データの視覚化を目指しているのと比較して、情報視覚化では離散的な情報構造の視覚化も範疇としている。
- 対話性を重視している。例えば構造の変化をスムーズなアニメーションで示したり、対話的

に視点を移動させたりする機能によって、情報の直感的理解を促進している。したがって厳密には4次元視覚化と呼ぶべきかもしれない。

—大量の情報に対して、遠近法による見かけ上の情報圧縮などの効果を積極的に取り入れている。

#### (4) 協調作業環境

共有された3次元環境は、複数の人間が協調して作業するための仮想的な空間となる。例えばIBMのCodellaらによるシステムは、複数の作業者が3次元空間に入りこんでシミュレーションを行う環境を提供している[Codella92]。ATRの竹村らは、仮想空間を通じて遠隔共同作業を行うための環境を構築している[Takemura92]。

#### (5) エンタテインメント

3次元ビジュアルインタフェースの、おそらくは最も大きな応用分野となるのが（広義の）エンタテインメントであろう。

数年前まではスプライト（画面上に登場するパターン）を利用した「疑似」3次元によるゲームが多かったが、現在ではポリゴンからなる完全な3次元形状によるゲームが主流になってきた。またSEGAのVirtua Fighterに見られるように、人間の身体モデルや重力モデルに基づいたリアリティの高いオブジェクトの動きなど、コンピュータグラフィックスの最近の研究成果が反映されたものも出現している。家庭用のゲームマシンにおいても、次の世代では現在のグラフィックスワークステーションと本質的には同じ3D処理が可能になり、本格的な3次元化が進むであろう。

#### 2.2.6 おわりに

ここでは3次元を用いたビジュアルインタフェースに関する技術動向について概観した。3次元グラフィックスの機能を実時間で対話的に使えるようになってからまだ日が浅い。2次元ビジュアルインタフェース（GUI）と比較しても圧倒的に経験不足で、未成熟の分野である。例えばGUIにおける基本的な対話手段（例えばメニューやドラッグ処理、ウィンドウ、アイコン）に相当する3次元の概念は、現時点では確立しているとはいえない。

また、本文中にも述べたように、現状では「完全な」3次元表示技術はなく、ステレオ画像やヘッドトラッキングなどによって疑似的に情報を表示しているにすぎない。したがって、例えば単に3次元オブジェクトを表示するだけのアプリケーションを考えてみても、インタラクションの有無によって、その分かりやすさが大きく左右してしまう。この意味で、3次元環境では、2次元環境にも増して良質のインタフェース技術が要求されるといえるだろう。

より大きな課題として、3次元ビジュアルインタフェースが本当に有効な適用分野は何なのかを見

定めていく必要があるだろう。フライトシミュレータ、CAD、エンタテインメントなどの分野では3次元の効用は広く理解されているといえる。しかし、例えばオフィスシステムや情報視覚化などの分野における3次元の利用には技術先行の傾向が強く、今後はその有効性を客観的に評価していく必要があるだろう。

### 【参考文献】

- [ADL93]Shooting Star Technology: ADL-1 6-D Tracking System, SIGGRAPH'93 brochure (1993).
- [Applewhite92]HughL. Applewhite: Position tracking in Virtual Reality, CyberEdge, (8) (1992).
- [Arthur93]KevinW. Arthur, KelloggS. Booth, and Colin Ware: Evaluating 3d task performance for fish tank virtual worlds, ACM Transactions on Information Systems, 11(3) (1993).
- [Asahi94]Nobuo Asahi: An environment for developing metaphor worlds - toward a user-friendly virtual workspace based on metaware, FRIEND21 International Symposium on Next Generation Human Interface (1994).
- [Aukstakalnis92]Steve Aukstakalnis and David Blatner: Silicon Mirage, Peachpit Press (1992).
- [Bier90]EricA. Bier: Snap-dragging in three dimensions, Computer Graphics (Special issue on 1990 Symposium on Interactive 3D Graphics), 24(2): pp.193-204 (1990).
- [Brooks86]F.P. Brooks, Jr.: Walkthrough - a dynamic graphics system for simulating virtual buildings, Proceedings of the 1986 Workshop on Interactive 3D Graphics, pp.9-21, Chapel Hill, NC (October 1986).
- [Bryson92]S.Bryson and M.Gerald-Yamasaki: The distributed windtunnel, Proceedings of Supercomputing '92 (1992).
- [Codella92]Christopher Codella, Reza Jalili, Lawrence Koved, et al.: Interactive simulation in a multi-person virtual world, CHI '92 (1992).
- [CrystalEyes93]StereoGraphics Corporation: CrystalEyes Eyewear, SIGGRAPH'93 brochure (1993).
- [Darken93]RudyP. Darken and JohnL. Sibert: A toolset for navigation in virtual environments, UIST '93, pp.157-165 (1993).
- [Fakespace93] Fakespace: Putting Virtual Reality to Work, SIGGRAPH'93 brochure (1993).
- [Green92]Mark Green and Dani Beaubien: Minimal Reality Toolkit Version 1.2 Programmer's Manual, Department of Computing Science, University of Alberta, Edmonton, Alberta (1992).
- [Herndon92]KennethP. Herndon, RobertC. Zeleznik, DanielC. Robbins, D.Brookshier Conner, Scott S. Snibbe, and Andries van Dam: Interactive shadows, Proc. of the ACM Symposium on User

- Interface Software and Technology (UIST '92), pp.1-6 (1992).
- [Houde92]Stephanie Houde: Iterative design of an interface for easy 3-D direct manipulation, CHI '92, pp.533-540 (1992).
- [Inventor92]Iris Inventor technical report: Technical report, Silicon Graphics (1992).
- [ISOTRAK87]Polhemus, Inc., Colchester, Vermont: 3SPACE ISOTRAK User's Manual (1987).
- [Koike93]Hideki Koike: The role of another spatial dimension in software visualization, ACM Transactions on Information Systems, 11(3): pp.266-286 (July 1993).
- [Liang93a]Jiandong Liang: JDCAD: A highly interactive 3d modeling system, Proceedings of the Fifth Annual Western Computer Graphics Symposium (1993).
- [Liang93b]Jiandong Liang and Mark Green: Geometric modeling using six degrees of freedom input devices, Proc. of the 3rd International Conference on CAD & Computer Graphics (CAD/Graphics '93), Beijing, China (1993).
- [Mackinlay90]JockD. Mackinlay, StuartK. Card, and GeorgeG. Robertson: Rapid controlled movement through a virtual 3d workspace, SIGGRAPH '90 Conference Proceedings, pp.171-176, Dallas Texas (1990).
- [McKenna92]Michael McKenna and David Zeltzer: Three dimensional visual display systems for virtual environments, Presence, 1(4):pp.421-458 (1992).
- [Meyer92]Kenneth Meyer and HughL. Applewhite: A survey of position trackers, Presence, 1(2):pp.173-200 (1992).
- [OpenGL93]Neider, Davis, and Woo: The OpenGL Programming Guide, Addison-Wesley (1993).
- [PHIGS93]Linda Kosko: PHIGS Reference Manual, O'Reilly & Associates (1993).
- [Sense8]Sense8 Corporation: WorldToolkit version 2.0 - The only choice in Virtual Reality software, SIGGRAPH'93 brochure (1993).
- [Shaw93]Chris Shaw, Mark Green, Jiandong Liang, and Yunqi Sun: Decoupled simulation in virtual reality with the MR Toolkit, ACM Transactions on Information Systems, 11(3) (1993).
- [SGI91]Silicon Graphics, Inc., Mountain View, California: Graphic Library Programming Guide (1991).
- [Snibbe92]ScottS. Snibbe, KennethP. Herndon, DanielC. Robbins, D.Brookshire conner, and Andries van Dam: Using deformations to explore 3d widget design, Computer Graphics, 26(2): pp.351-352 (July 1992).
- [Strauss92]PaulS. Strauss and Rikk Carey: An object-oriented 3D graphics toolkit, SIGGRAPH '92 Proceedings, pp.341-349 (1992).

- [Takemura92]Haruo Takemura and Fumio Kishino: Cooperative work environment using virtual workspace, ACM 1992 Conference on Computer-Supported Cooperative Work, pp.226-232, Tronto, Canada (November 1992).
- [Venolia93]Dan Venolia: Facile 3D direct manipulation, INTERCHI '93, pp. 31-36 (1993).
- [Wang90]Jih fang Wang, Vernon Chi, and Henry Fuchs: A real-time optical 3D tracker for head-mounted display systems, Computer Graphics (Special issue on 1990 Symposium on Interactive 3D Graphics), 24(2):205-215 (March 1990).
- [Zelevnik93]RobertC. Zelevnik, KennethP. Herndon, DanielC. Robbins, Nate Huang, Tom Meyer, Noah Parker, and JohnF. Hughes: An interactive 3D toolkit for constructing 3D widgets, SIGGRAPH'93 Proceedings, pp.81-84 (1993).

## 2.3 3次元視覚化の文献紹介

3次元コンピュータグラフィックス技術を用いた視覚化に関する論文を紹介する。

一般に、3次元視覚化は対象によって科学技術計算結果の視覚化 (Scientific Visualization、Volume Visualization)、情報の視覚化 (Information Visualization)、プログラム構造/実行過程の視覚化 (Program Visualization)、仮想現実感 (Virtual Reality) などに分けられる。

ここでは、情報の視覚化、プログラムの視覚化並びにマウスを用いた3次元物体の操作に関する論文を紹介する。取り上げた論文は、以下の9論文である。

### (1) 情報の視覚化

- ・ Card, Stuart et al.: Information Visualizer, An information Workspace, CHI'91 Conference Proceedings pp.181-188(1991).
- ・ Poltrock, Steven et al.: Virtual Metaphors : 3D Interfaces for Knowledge Workers, Proceedings of Hawaii International Conference on System Science-1992, Vol. 2 pp.678-686(1992).
- ・ Staples, Loretta: Representation in Virtual Space : Virtual Convension in the Graphical User Interface, INTERCHI'93 Conference Proceedings pp.348-354(1993).

### (2) プログラムの視覚化

- ・ Lieberman, Henry: A Three-Dimentional Representation For Program Execution, IEEE Worshop on Visual Languages pp.111-116(1989).
- ・ Calliss, Frank et al.: Inter-Module Code Analysis Using 3-Dimensional Interactive Graphics, Proceedings of the Hawaii International Conference on System Sciences-1992, Vol. 2 pp.670-677(1992).
- ・ Stasko, John et al.: Three Dimensional Computation Visualization, VL'93 pp.100-107(1993).
- ・ Reiss, Steven: A Framework for Abstract 3D Visualization, VL'93 pp.108-115(1993).

### (3) マウスを用いた3次元操作

- ・ Houde, Stephanie: Iterative Design of an Interface for Easy 3-D Direct Manipulation, Human Factors in Computing Systems : CHI'92 Conference Proceedings pp.135-142(1992).
- ・ Venolia, Dan: Facile 3D direct Manipulation, INTERCHI'93 Conference Proceedings pp.31-36(1993).

### 2.3.1 インフォメーションビジュアライザ

原文タイトル： THE INFORMATION VISUALIZER, AN INFORMATION WORKSPACE

著 者： Stuart K.Card  
Geroge G.Robertson  
Jack D.MackinLay  
(XEROX Palo Alto Research Center)

出 典： CHI'91 Conference Proceedings, pp.181-188, 1991

#### (1) 概要

本論文は、インフォメーションワークスペースと呼ばれる情報探索システムのためのユーザインタフェースについて述べている。

このシステムは、以下の基本方針に従っている。

- ・即時ストレージの容量を大きくするために3D/Roomsを用いる
- ・ユーザと情報システムとを結び付けるために、認知コプロセッサというスケジュールベースのユーザインタフェース・アーキテクチャを用いる
- ・情報構造を操作するために情報のビジュアライゼーションを行う

これまでの情報検索システムの研究は、主に情報をいかに蓄え、選択し、並べるかに重点が置かれており、ユーザインタフェースにはほとんど目が向けられていなかった。そこで、Information Visualizerと名付けた実験システムを作成し、その上で新しい方向を探ってみた。

#### (2) 情報のコスト構造

一般に、オフィスのような人間社会だけでなく、自然界においても、情報処理システムは情報処理のコストを小さくしようとする傾向がある。様々な観察から、情報検索システムのユーザインタフェースの設計の基本となる考え方を追及し、次の六つの結果を得た。

##### ① 階層化

情報を階層化することにより、効率よいアクセスが可能である。情報を階層化し効果を上げているものは、自然界やエンジニアリングシステムなどに見られる。

##### ② コスト比

情報を得るためのコストは、情報が目の前に在るか無いかで、大変な差がある。例えば、仮

想メモリシステムでのストレージごとのアクセスタイムの時間比を示すと次の表のようになる。

表2.3.1-1 コンピュータの記憶のコスト比率

| ストレージタイプ |         | アクセスタイム | 比率                    |
|----------|---------|---------|-----------------------|
| 即時ストレージ  | RAM     | 80 ns   |                       |
| 2次ストレージ  | Disk    | 20 ms   | $2.5 \times 10^{**5}$ |
| 3次ストレージ  | Optical | 5 s     | $2.5 \times 10^{**2}$ |

### ③ 参照の局所性

情報処理において、参照される情報には局所性がある。ある短い期間をとってみれば、参照は集積された情報に対して均一に行われるのではなく、ワーキングセットと呼ばれるサブセットに集中する傾向がある。

### ④ 参照のクラスタリング

ある作業を繰り返し行くと、その中で使用される情報は特定のクラスタを形成する。情報処理作業を進めていくと、あるクラスタを局所的に参照する傾向がでてくる。その後、突然別のクラスタに移っていく。

### ⑤ 情報コスト比の最大化

情報システムは、限られたコストの中で、処理する情報の量をできるだけ多く（希に少なく）しようとする。

### ⑥ アブストラクション

低レベルの情報システムは、情報を集約し、より上位のセンタースystemに渡す。低レベルのシステムは、上位のシステムが処理できるパターン化された構造にまで抽象化を進め、上位のシステムはさらに抽象化を行うことにより情報量を減らす。

## (3) インフォメーションワークスペース

歴史上、最初のワークスペースとしてのスクリーンは、スクロール機能を持つコマンドベースのテレタイプである。これは現在でもDOSやUNIXの環境の中で使われている。次に現われたのが、NLSシステム（EnglebartのNLS system）で代表される、マウスを用いたポイントクリックでドキュメントの操作を可能にしたスクリーンである。その次の段階では、SmallTalkとStarシステムの一部として開発されたデスクトップメタファが現われた。メニューとウィンドウとアイコンを備え、マウスを用いたダイレクトインタラクションを可能にした。このシステムは、複数のドキュメントの表示が可能なワークスペースを持っている。デスクトップメタファは、即時ストレージ上で以前より大量の情報を低コストで扱えるようにした。そうして、Datalandへ発展する。Datalandの拡張デスクトップメタ

ファは、一つの大きなデスクトップである。しかし、ワークスペースの関連部を検索するためのコストが、ワークスペース内の要素の数の増加に比例して急激に増大した。最後にRoomsに到達する。Roomsはマルチデスクトップであるが、異なるワークスペースで同一の情報を共有する機能を付加してマルチデスクトップの欠点を補った。また、オーバビュー機能やナビゲーションヘルプ機能なども付加した。

情報検索のユーザインタフェースのために、Roomsを改善した情報ワークスペースを提案する。離れた場所にある情報を単純にアクセスするという従来の情報検索の考え方とは異なり、以下の点を考慮した。

- ・情報のコスト構造は、検索対象（即時ストレージだけでなく、遠くにある2次、3次ストレージの情報も含む）を完全に統合したものとする。
- ・情報アクセスをより大きな仕事の一部と考える。検索効率の向上といった狭い範囲に固執しないで、ユーザの作業の効率を向上させることを狙いとする。

Information Visualizerは、このような考え方を実験的に実現したものであり、主要な三つのコンポーネントからなる。

- ① 3D/Rooms・・・Roomsシステムを3次元化したもの
- ② 認識コプロセッサ・・・アニメーション指向のユーザインタフェース・アーキテクチャであり、ユーザとシステム間のインタラクション速度を大きくする
- ③ 情報の視覚化・・・様々な構造を表示するブラウザであり、種々のレベルの抽象化を提供する

#### (4) 即時ストレージの増大：3D/Rooms

Roomsはマルチデスクトップを用いることにより、即時ストレージを大きくしている。そして、アクティブなRoomのみ表示することで情報の溢れを防いでいる。

3/D Roomsはこの考えを3次元ワークスペースに取り入れた。今までのデスクトップメタファやオリジナルのRoomsシステムでは、Roomの表示は固定であったが、3/D Roomsではユーザは自由に位置や方向を指定することができる。ユーザはRoomの中を動き回ったり、注目したいオブジェクトをズームしたり、ドアを開けて別のRoomへ行ったりすることができる。3/D Roomsは、デスクトップメタファとは異なる新しい構成要素を持っている。ユーザの探索方法を支援するウォーキングメタファ（Walking Metaphor）アルゴリズムと、素速く正確に注目点にたどり着くことを支援する注目点指数移動（a point of interest logarithmic movement）アルゴリズムである。また、マウスのみを用いて3次元オブジェクトを素速く移動させるアルゴリズムも組み入れた。3/D Roomsは、アニメーションをベースとしたユーザインタフェース（認知コプロセッサ）の上に構築されている。Roomsシステムと同様、3/D Roomsは同時にすべての3次元ワークスペースを見渡すことのできるオーバビューを持つ

ている。3/D RoomsはオーバビューからRoomsに実際に入ったり、オブジェクトを変更したりできる点がRoomsシステムと異なる。

#### (5) 情報の操作性の増大：認知コプロセッサ

人間は3種類の応答時間の制約を持っており、システムはそれに合わせて動作することが望ましい。認知コプロセッサは応答時間に関して次のように機能する。

- ・知覚的処理に対する応答時間の制約

アニメーションがスムーズな動きをするためには、0.1秒以内で描画を繰り返す必要がある。制限時間を超えるような場合には、レンダリングの質を落として、応答時間を速めようとする。

- ・即時応答を求められている場合の時間制約

人間は、なんらかの刺激に対し、あらかじめ用意していなくてもなんらかの応答を1秒以内にすることができ、これによってインタラクションが継続中であることを知らせる。これと同様、1秒を超えるときは、なんらかの疑似応答をし、ユーザの注意力を引き付けておくようにする。

- ・定型的仕事の指示に対しての応答時間の制約

定型的仕事の指示に対する応答時間は10秒以内とする。複雑な仕事には時間が必要であるが、ユーザは常にアクティブでいられる。

#### (6) 情報の抽象（要約）化：情報の視覚化

多くの情報の集まりに共通な構造がある。1番目の例がUNIXのディレクトリに見られる階層構造である。Cone TreeやCam Treeと呼ばれる構造化ブラウザは、階層構造を3次元アニメーションで可視化するものである。2番目の例が線形構造である。このための構造化ブラウザがPerspective Wallである。3番目の例が科学データと類似した2次元の連続したデータの外観を扱ったものである。

4番目の例がビルのような空間構造である。例えば、オフィス内の組織を選択すると、そこに所属する人の名前と写真が表示されるというようなものである。

このような視覚化には、情報構造の動的な変化を調べるために、対話型のコンピュータグラフィックスアニメーションを利用する。さらに視覚化が進んだものがRoomsのオーバビューである。Information Visualizerは、人間の知覚にとってパターン認識が容易なように、データを抽象化している。

#### (7) まとめ

Information Visualizerは、情報検索のための新しいユーザインタフェースを開発するための実験システムである。これは、情報検索の問題を情報ワークスペースのコスト構造の問題として捉え直すと

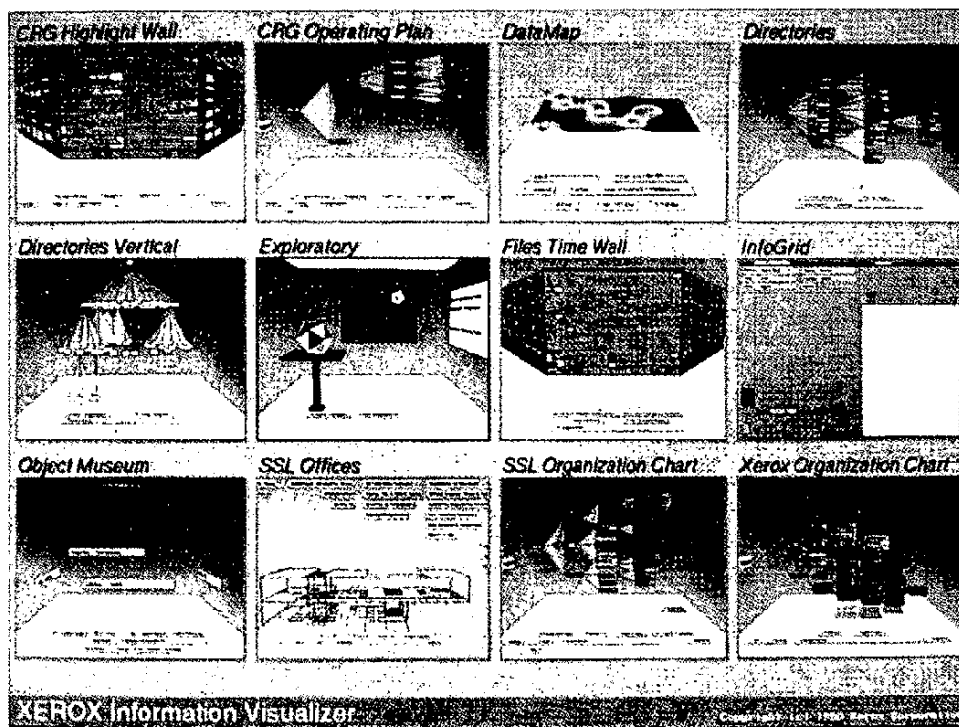


図2.3.1-1 オーバビュー

いう考察を基に作成された。そのため、コンピュータのデスクトップメタファを以下のように改善した。

- ・情報ストレージのコスト階層をコントロールするために3D/Roomsを用いる
- ・ユーザが複数のエージェントと繰り返しスムーズなインタラクションをできるよう認知プロセスを用いる
- ・ユーザに抽象度の高いデータを提供するために、情報の視覚化を用いる

## 2.3.2 ナレッジワーカのための3次元インタフェース

原文タイトル： Visual Metaphors : 3D Interfaces for Knowledge Workers

著 者 : Steven Poltrock  
John Lovgren

Mark A. Tarlton

P.Nong Tarlton

(Microelectronics and Computer Technology Corporation)

出典 : Proceeding of Hawaii International Conference on System Science, Vol.2,  
pp.678-686, 1992

#### (1) 概要

本論文は、Metaphor Builderと呼ばれる3次元ユーザインタフェースの開発環境と、Metaphor Builderを評価するために作成したビジュアルオフィスという実験システムについて述べている。3次元環境の中で親しみやすいオブジェクトを作成することが、日常業務を行っている人には理解されやすいと考え、2次元アイコンベースのインタフェースよりも、もっと柔軟な環境を構築することを目的としてMetaphor Builderを開発した。

#### (2) システム開発

##### (a) 最初の開発システム

デスクトップ環境の拡張として、3次元ワイヤフレームのグラフィックエディタを用いて、ビジュアルオフィスを作成した。

これらを使用して分かったことは、第一に、ワイヤフレームは効果的な手法ではないということである。なぜならば、オブジェクトが重なっていた場合に、どちらが手前にあるのかが分からなくなるからである。第二に、モノクロのSunワークステーションとSunCoreのグラフィックパッケージでは、ダイナミックな3次元の表現力に欠けることである。オブジェクトの移動は遅く、ジャギーを生じる。第三に、開発した3次元グラフィックエディタは決して使いやすいといえる代物ではなかった。なぜならば、2次元エディタに3次元の機能を盛り込んで作成していたからである。

##### (b) 開発システムの変更とモデルベースへのアプローチ

インタラクティブかつリアルな3次元インタフェースの開発には、オブジェクト指向的な開発環境と、インタラクティブな3次元グラフィックをサポートするハードウェアの必要性を感じた。そこで、Lambda上でIRISをコントロールするプログラムをLispを用いて開発した。

##### (c) リアリズムの実現

ユーザインタフェースは、インタラクティブに操作することができるリアルな3次元環境がよいと

思っていたが、ハードウェアの制約を受けることは免れない。3次元オブジェクトの位置決め精度と速さをマウスとインデックスフィンガー付のセンサについてテストしたところ、驚くべきことにマウスが一番正確かつ速く制御できるということが判明した。

IRISの場合、ワイヤフレームの移動はスムーズなのだが、1イメージの陰面消去に30秒を要する。仕方なく、オブジェクトは単に表面を塗るだけとした。

### (3) Metaphor Builder

Metaphor Builderは、任意のアプリケーションのリアルな3次元インタフェースをインタラクティブに設計、インプリメント、テストするインタフェースの専門家によって使われることを狙ったツールである。最初にインプリメントしたツールは、GrafBagと呼ぶグラフィック表現スキーマと、Visageと呼ぶ3次元エディタ、それにGrabGrafと呼ぶモデルエディタの三つである。同時に、これらのツールを検証するため、ビジュアルオフィスを作成した。

#### (a) ビジュアルオフィスの機能

ビジュアルオフィスは、研究活動を支援するものであり、研究者のソフトウェア開発やレポート作成などの仕事を順序よく行えるように支援し、その結果を保存、管理する。

設計の目標と制約を認識するために業務分析を行った。その結果、以下の五つのアクティビティ（機能要素）を見い出した。

- ・プログラムの作成とデバッグ
- ・レポート作成、手書きのメモ
- ・メールの送受信
- ・会議のスケジューリングおよび参加
- ・様々な情報の検索とそのファイリング

これらのアクティビティはすべてLambda上のツールでサポートされ、各個人の好みのウィンドウとツールを使用できるようにした。

ビジュアルオフィスを構成するための基本機能は次のとおりである。

- ・アクティビティの作成
- ・アクティビティの中断
- ・アクティビティの再開
- ・アクティビティの終了

最初に新しいアクティビティを作成する場合、そのアクティビティで一般的に使用されるツール群のテンプレートを使用することができる。作成されたアクティビティとそれに関係のあるファイル群

---

## 2. ビジュアルインタフェースの技術動向

をコンテキストと呼ぶ。ビジュアルオフィスは作業中、中断中のすべてのコンテキストを表示する。研究者は、コンテキストを切り替えて、ウィンドウやツールの配置を変えたり、コンテキスト中のツールを選択して、作業中のウィンドウを切り替えたりすることができる。

#### (b) ビジュアルオフィスの外観

ビジュアルオフィスのオブジェクトはVisageを用いて作成され、変更される。ビジュアルオフィスの外観はAaron Marcus and Associatesの助けを借りて設計した。

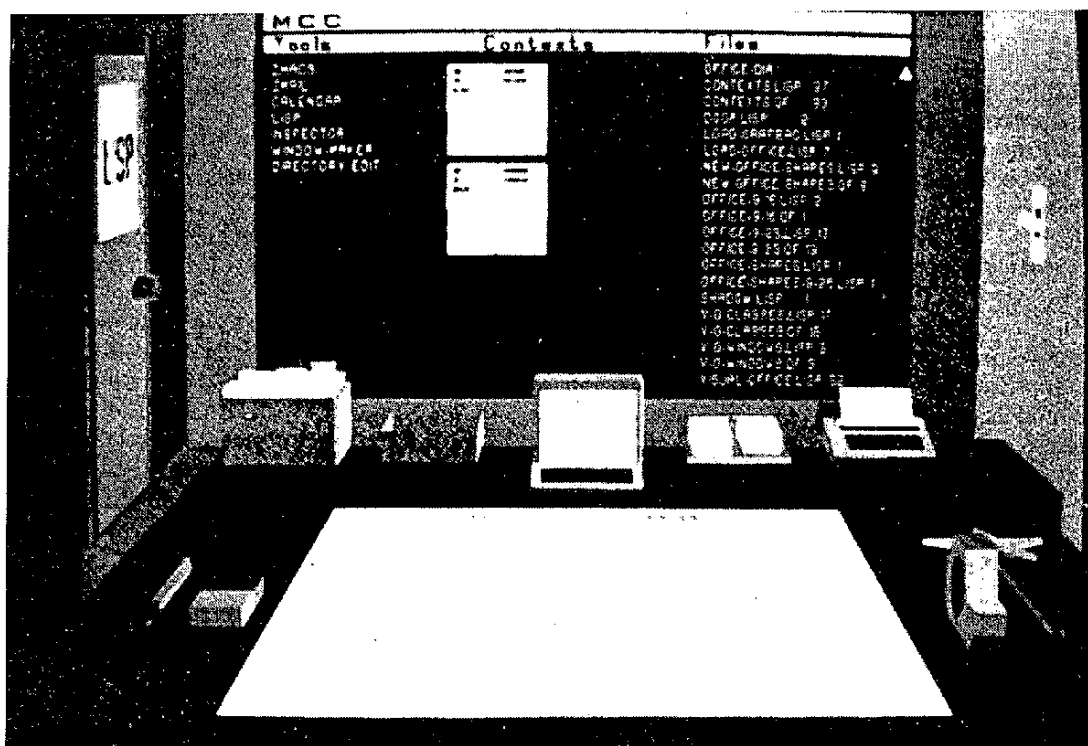


図2.3.2-1 IRISターミナルに表示されたビジュアルオフィス

図2.3.2-1で、テープホルダが机の上から浮いているのは、テープホルダが選択されたことを表わしている。また、壁には中断状態のコンテキストが二つある。

現在作業中のコンテキストは、デスクパッド上にツールやファイルのリストとして表現され、中断状態にあるコンテキストは壁に格納されている。壁にはさらに、コンテキストに追加可能なツールとファイルのリストがある。壁にコンテキストを格納するというのは、ホワイトボードにアイデアを書いたりそれを格納したりという、実世界のメタファである。

ファイリングやソフト開発、書類作成といったアクティビティはオブジェクトとして表現され、机の後方に配置されている。これらのファンクションはメニュー形式を用いずに、鉛筆や消しゴム、電

灯スイッチ、ドアなどで表現されている。

#### (c) ビジュアルオフィスのインプリメンテーションモデル

ビジュアルオフィスは以下の構成要素からなる。

- ・ビュー
- ・メタファ
- ・アプリケーション
- ・ユーザアクション

最初、ビューとメタファは等価であると考え、まず三つだけの構成要素からスタートした。しかし、スクリーン上のオブジェクトの外観（ビュー）の変更は、必ずしもメタファの変更を意味しない。例えば、インタフェース設計者がファイルディレクトリを引き出しとして作成しようとする。これはメタファである。次に、設計者は3次元の箱を作成し、それにラベルと把手をつけて引き出しとする。これはビューである。

メタファモデルはインタフェースにどんな概念があるかに注目したものであり、ビューモデルはメタファオブジェクトをどのように外観付けするか注目したものである。ビューモデルはインタフェース上のグラフィックオブジェクトを定義する。

ビジュアルオフィスの作成に使用したMetaphor Builderを構成する三つのツールは次のとおりである。

##### (i) Visage

Visageはインタラクティブな3次元オブジェクトエディタであり、これを使ってビューモデルを作成した。ビューモデルには、ビジュアルオフィス上のすべてのオブジェクトが含まれている。オブジェクトは、パレットから選んだ基本オブジェクトを組合せて作成した。

##### (ii) GrafBag

GrafBagは、クラスライブラリで構成されるオブジェクト指向のグラフィック表示システムである。GrafBagは、コンポーネントとインヘリタンスの2種類のオブジェクトの階層化を支援する。ビジュアルオフィスではカメラ視点が一つであるが、GrafBagそのものは複数のカメラ視点での表示が可能である。それによつて、多様な表示を行うことができる。

##### (iii) GrabGraf

GrabGrafは、インタフェースモデルのグラフィック表示を構築、表示、操作するエディタである。例えば、GrafBagモデルのインヘリタンスと階層関係を表示する。

#### (4) 結論

---

## 2. ビジュアルインタフェースの技術動向

ビジュアルオフィスとMetaphor Builderなどのツールの開発を通じて、ナレッジワーカのためのアニメーションを使ったインタラクティブな3次元インタフェースは実現可能であり、なおかつ優れたビジュアルメタファをサポートすることができることが分かった。これらのインタフェースは、オブジェクト指向の環境の中で、ナレッジベースモデルを構築することで、素早いデザイン、インプリメント、見直しが可能である。しかしながら、グラフィックスに必要なハードウェアは非常に高価なので、このインタフェースは現実的でない。

ビジュアルオフィスはバーチャルワールドとして位置付けられる。またSemNetはサイバースペースとして位置付けられる。これらのインタフェースは視覚的に興味深く、3次元環境でのインタラクションの研究の契機となるものであった。しかし、3次元インタフェースが他の手法より優れていることを明らかにしたわけではない。効果的なユーザインタフェースの設計には、しっかりした業務分析モデルが必要である。このようなモデルの構築に、3次元グラフィックを使用することはできるが、インタフェース設計者は使いこなすまでには何度も失敗を重ねることになる。3次元インタフェースをどういう場合に使うべきか、あるいは使うべきでないか、こういったものに対して使えばよいのか明らかになったわけではない。

### 2.3.3 バーチャルスペースでの表現

原文タイトル： Representation in Virtual Space:  
Visual Convention in the Graphical User Interface

著 者： Loretta Staples  
(Loretta Staples Design)

出 典： INTERCHI'93 CONFERENCE PROCEEDINGS, pp.348-354, 1993

#### (1) 概要

本論文は、絵画での表現技法、特に遠近法と光の効果を取り入れた新たな手法を提示している。背景には、既存のGUIデスクトップ環境は、通常、平面的なワークスペースの上にマルチウィンドウを配置する方法をとっているが、ワークスペースの中で3次元表示を行うための操作は、ますます複雑になってきていることがある。

## (2) 遠近法による表現

既に、MotifとNew Wave iconsで疑似遠近法が取り入れられている。しかし、それはただ描画方式に目新らしさを加えたにすぎず、表現は依然平面的である。

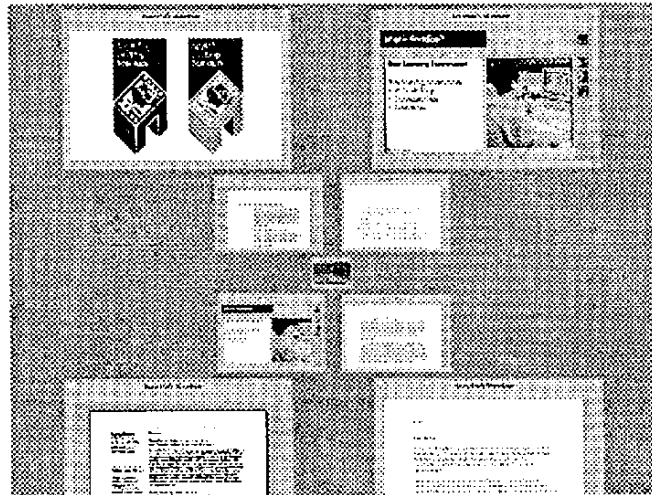


図2.3.3-1 複数のウィンドウに対する一点遠近法の例

この例では一点遠近法を用い、一様にウィンドウのサイズを変えて、様々なオブジェクトを表現している。ウィンドウのサイズを変えることによる遠近法の効果により、空間の奥行きを表現しようとしている。しかし、オブジェクトそのものの表現（ここでは各ウィンドウ）は未だに平面的である。

これをさらに改善し、重ね合わせ技法を加えたのが図2.3.2-2である。改善は見られるが、まだ平面的である。

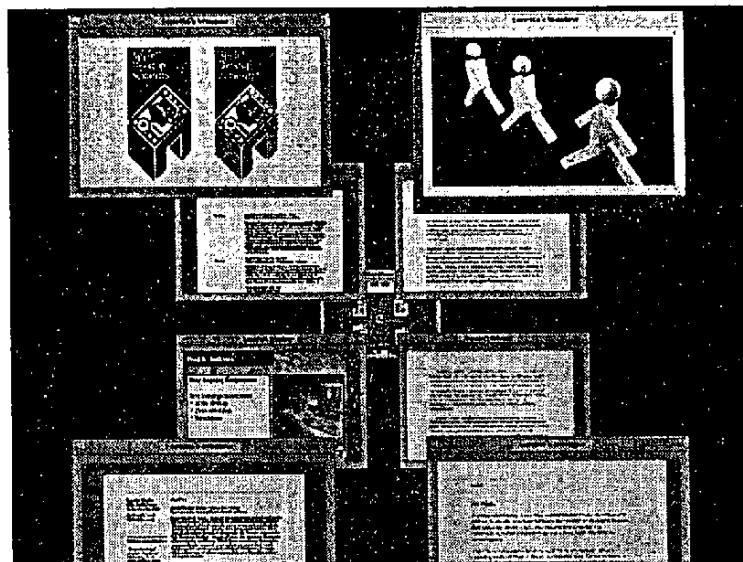


図2.3.3-2 奥行きを出すために重ね合わせ手法を加えた例

### (3) 光線と陰影付け

光線の効果を付加することにより、さらに現実の世界に近づく。例えば、影を付けることは遠近感を増し、あるオブジェクトが他のオブジェクトに影響を及ぼすことを表現する場合にも使用することができる。

また、注目するオブジェクトに光を当てる手法は、古い絵画の手法にも見られるが、特定のオブジェクトに注目するために、スポットライトを当てるという方法がある。

### (4) 透過・不透過

実際の世界で透明なものは限られている。したがって、透過表現を用いることにより超現実的表現を行うことができる。透過を用いると、そのものを見にくくする。透過は表現される情報量を増加させる。あまり多用するのは好ましくないが、オブジェクトの属性として適切に使用することで効果的となる。オブジェクトの背後にあるものを見たい場合に、再配置無しに表現できるのも利点である。

### (5) 複合効果

これまでに説明した手法を一つの場面に表現している例を図2.3.3-3に見る。

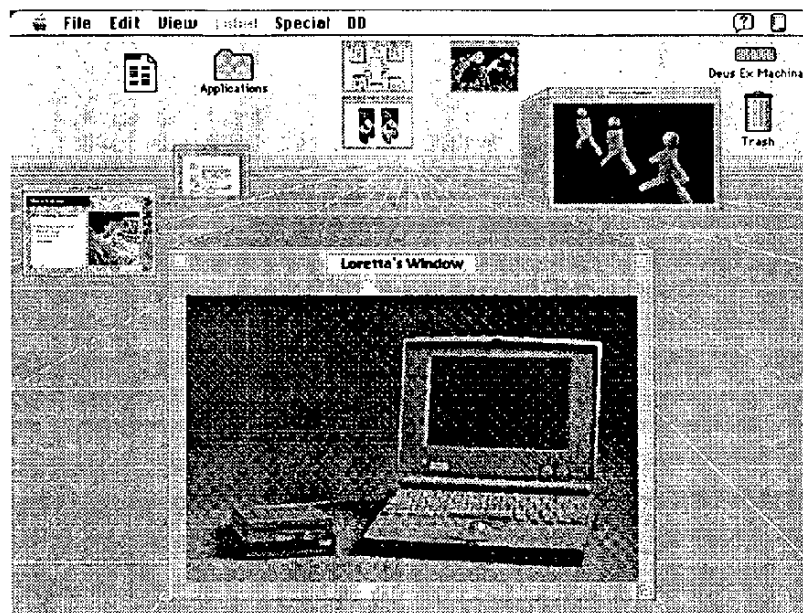


図2.3.3-3 アクティブ状態の前景部分と非アクティブ状態の背景部分からなるデスクトップの風景画法

デスクトップは、水平線で二つのエリアに分けられている。手前のエリアにあるウィンドウは、現在アクティブ状態である。また、ウィンドウが箱型のものは、そのオブジェクトが次元を持ったもの、

あるいは配列構造のものであることを表す。奥のエリアには、現在非アクティブ状態であり、実行待ちのオブジェクトが配置されている。

しかしこのように、様々な手法を単純に組合せただけの環境は、疑いようもなくとても複雑である。この複雑さに適応する新しいインタフェースが求められる。

#### (6) 総論

まだ未開発のGUI手法は多く存在する。その中の一つとして、多様なウィンドウ表現は多様な意味表現を可能にするであろう。ここでの実験から、絵画での表現手法をコンピュータインタフェースの表現手法に応用できる可能性が多くあることを示していると考ええる。さらに、絵画の歴史をひもとくことにより、コンピュータインタフェースの新しい解決方法を探ることができるのではなかろうか。

#### 2.3.4 プログラム実行の3次元表現

原文タイトル : A Three-Dimensional Representation  
For Program Execution

著 者 : Henry Liberman  
(Media Laboratory Visual Language Workshop Massachusetts Institute of  
Technology)

出 典 : IEEE Workshop on Visual Language, 1989, pp.111-116

#### (1) 概要

本論文はコンピュータプログラムの抽象的3次元表示について述べている。プログラムは動的なプロセスの記述であるので、プログラム実行のアニメーション中の図式表示の動的振る舞いに焦点を当てる。目標は、プログラムの動きの動的な側面を視覚化することより、デバッグ処理の助けとすることである。

統合化されたプログラム実行のビューを提供するために、デバッガで用いられた図式化技術は以下のものである。

- ・コンテキストの変化を表示しながらビューポイントを固定する再ルート化処理

---

## 2. ビジュアルインタフェースの技術動向

- ・グラフィックオブジェクトの代用としての数値表示
- ・逆戻り可能な制御構造
- ・連続的時間列ビューを同時に表示するための3次元回転の使用

## (2) 木構造としてのプログラム表示

以前、関数や引数の呼び出し木をノードとアークで表現するというグラフィック表現を実験した。2次元上に木構造を表示する場合、これはうまい表現方法であった。関数と引数の関係を空間的に表示するための別の可能性として、図式形態に内包する（閉じ込め）という方法がある。この例は、様々なグラフィカルプログラミングのプロジェクトに見ることができる。大きいアイコンが関数を図式的に表現し、その中に含まれる小さいアイコンが引数を表現している。この手法を3次元に用いた例が図2.3.4-1である。

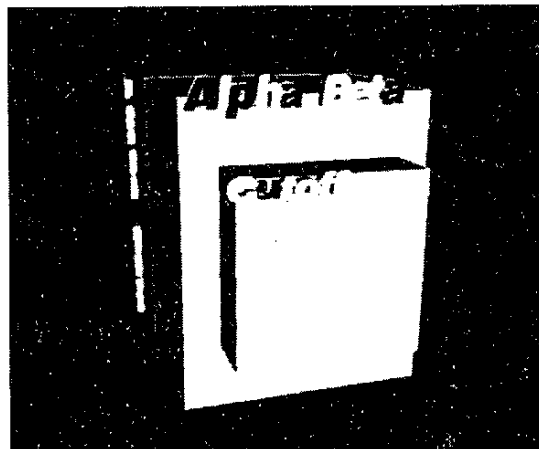


図2.3.4-1 3次元を用いた関数の表示

図2.3.4-1 は、Cutoffを引数として持つ関数Alpha-Betaの関数呼び出しを表している。プログラム要素のもっと複雑な抽象的3次元表現が可能ではあるが、ここではラベルでのみ認識される簡単な3次元形状の箱で表現している。

## (3) 注目点の移動か、内容の移動

動的なプログラム表現のための視覚化表示には、現在注目しているプログラムの特定の部分と、それ以外の部分との関係をいかに表示するかというデザイン上の問題がある。

関心のあるビューポイントや現在の注目点の表示には二つの方法がある。一つはプログラムの表示自身は固定して、強調表示、点滅、カーソルの動き、あるいはその他の注意を促す手段等を用いて、ビューポイントを移動する方法である。ソーステキストを表示し、注目点を強調表示し、実行に連れ

てカーソルを移動する方法がこれにあたる。もう一つの方法は、プログラミング環境ではあまり一般的ではないが、ビューポイントを固定し、現在の注目点を反映させるために、回りのコンテキストの内容を変化させる方法である。

以前は、コンテキストの固定表示を用いるか、または、注目点の動的表示を用いるかの選択は、表示の動的な更新を最小限にするということに左右されていた。これは一般に、注目点は周囲のコンテキストの表示よりもはるかに小さいことが理由である。しかし、変更時のスクリーン全体の再表示を禁止する必要がなくなるほど、コンピュータの処理スピードが向上したことが、別の（内容の移動）方法を試みる自由を与えた。この試みは、連続フレーム上の視覚的連続性を一部犠牲にしているが、大きいプログラムの詳細レベルを簡単に制御することができるという利点がある。内容の動的表示と注目点の固定表示を用いた方法を調べることにより、それらに慣れるまでの間の本当のトレードオフとは何かを見つけ出したいと思っている。

#### (4) 再ルート化（木構造でのコンテキストの移動）

再ルート化とは、木構造上で注目点を移動する方法である。これは、木のルートノードからターゲットノードへルートの分岐の一つに沿って移動した場合、以前と同じ連結関係を保ちつつ、ターゲットノードをルートとする新しい木構造を表示するというものである。図2.3.4-2の再ルート化はビューポイントをルートノードに固定する。

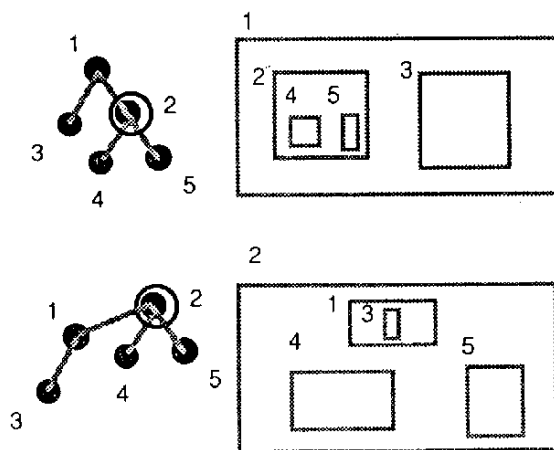


図2.3.4-2 再ルート化

#### (5) 評価の表示（回転と代用）

動的に表示する必要のあるイベントの種類としては、プログラム式の評価と戻り値の表示がある。評価の基本モデルとしては、式自身を表示する代わりに、その式で計算された値を表示する。

プログラム式と戻り値とを区別するために、なんらかの視覚化手段が必要であった。3次元の表現では、評価は、観察者に向かっている軸に対する回転としてグラフィカルに表現される。観察者に対しては同じラベルの付いた面を向けているが、式と値はそれぞれ箱や菱形で表現されるので、明確に区別することができる。値をラベル付けしたテキストは、式をラベル付けしたテキストと置き換わる。

このバージョンのシステムでは、置き換えの操作は一瞬にして行われるが、表示されたオブジェクトの作成と削除を、フェードイン/フェードアウトの効果を用いて、連続的に表示するインタフェースの調査も行っている。これは連続表示機能の向上に役立つ。また、オブジェクトに半透明表示を用いる方法の調査も行っている。この方法により、あまり重要ではない情報（以前の状態の履歴等）は、より重要な情報を隠すことなく、控えめに表示しておくことが可能となった。

ユーザは二つの異なったビューを持つ。一つは実行中のプログラムを、時間軸に沿ったイベントの列としてみる「ヘッドオン」ビューである。これは既存のシングルステップデバッガと等価である。もう一つは、プログラム環境では古くから「スタック」と呼ばれているもので、ある時点でプログラムが解決しようとしている目標の時分割ビューである。

旧来のデバッガでも、通常、イベントの連続的なビューを見ることができ、プログラムを止めてその時のスタックの様子を見ることが出来る。しかし、ほとんどのシステムでは、個々のスタックフレームの表示は、ユーザが個別に要求を行わなければならない。以前開発したシステムでは、プログラム実行時に動的に更新された結果を、自動的に別のウィンドウ上にスタックの連続表示を行うことによって、この状況を改善した。これによって、特別な要求を行うことなくスタックの概観を見ることが可能であり、スタックの変化とプログラムの変化の関連をアニメ化することにより、プログラムが何を行っているのかをユーザに分かりやすくした。

しかし、これだけではまだ完全ではない。なぜならば、スタックとプログラムが別々のウィンドウに表示されているからである。プログラムとそのスタックフレームの関連は、表示されていない。

一方、3次元上のプログラム表現では、各々のプログラム式は箱で表現され、その正面にラベルが付いている。また同時に、側面にもラベルが付いている。現在評価されている式から離れるに従って、小さなプログラム要素は観察者に近づいてくる。すなわち、Z軸はスタックの深さとして用いられている。

3次元プログラム表現において、スタックの物理的なメタファを直接視覚化することが可能となった。ヘッドオンビューが通常事象の実行を連続的に表示している間、それと同時に3次元上の視点を回転することにより、スタックビューを簡単に見ることができる。各々の連続するスタックフレームはそれに含まれる式を表示する。また、視点を回転することにより、サイドビューはすべてのスタックフレームを同時に見ることができる。ハードウェアが行う回転、パン、ズームにより、ユーザの注目するスタックの一部を強調することができる。スタックを見るということは、単純にプログラム表

示のもう一つのビューポイントであり、各々のイベントとそれに関連するスタックフレームとの対応は視覚的にはっきりと認識することができる。

### 2.3.5 3次元グラフィックスを利用したインタラクティブなモジュール間コード解析

原文タイトル： Inter-Module code analysis Using 3-Dimensional Interactive Graphics

著 者： Frank W. Calliss

Thomas A. Forey

(Department of Computer Science and Engineering Arizona State University)

Mustafa Ismail

(IDEA Courier)

出 典： Proceedings of the Hawaii International Conference on System Sciences, Vol. 2,  
pp.670-677, 1992

#### (1) 概要

本論文は、3次元カラーグラフィックスを使ったインタラクティブなモジュール間コード解析システムについて述べている。

コード解析には二つの重要な面がある。その一つはプログラムコンポーネント間の依存関係を知ることであり、もう一つはプログラムコンポーネントの使い方を解析することである。

インターコネクショングラフを表示することにより、モジュール間コード解析をサポートすることを狙っている。3次元空間を使うことにより、2次元ディスプレイ技術では難しかった表示上の問題のいくつかを解決している。

#### (2) モジュール間コード解析

モジュールは、プログラムコンポーネントの集まりをグループ化したクラスタリング構造をしている。モジュール構造を含む言語で書かれたプログラムは、通常、相互接続されたモジュールの集まりからなる。モジュールを接続する方法には以下の三つがある。

##### ① インポート／エクスポート

あるモジュールが他のモジュールで宣言されたコンポーネントを使う場合

## ② インヘリタンス

あるモジュールが他のモジュールの特化したもの、あるいは拡張したものとして宣言される場合

## ③ 宣言

あるモジュールが他のモジュールの中で宣言される場合

### (3) モジュール間コードのビジュアル化

このシステムの主な目標は、以下のとおりである。

- ・大きなプログラムをより短時間に理解できる
- ・ユーザがインタラクティブに表示を編集できる
- ・ユーザにプログラム構造の明確なイメージを与える
- ・ユーザが表示のビューを変えられる
- ・ユーザが表示される情報量を制御できる

インターコネクショングラフを描くデータベースを創るため、ユーティリティを使ってソースコードから演繹データベースをジェネレートする。階層グラフ構造は、エンティティ間の依存関係を表すエッジを定義するのに使う。グラフのノードは、六つのプログラムエンティティからなる。それらはモジュール、プロシジャ、関数、変数、定数、型である。モジュールは3次元の箱で、プロシジャと関数は3次元の四面体、変数は2次元の長方形、定数は2次元の円、型は2次元の六角形で表示する。モジュール、プロシジャ、関数が3次元オブジェクトで表示される理由は、これらが階層構造中に他のエンティティを含んでいることがありえるからである。変数、定数、型は他のエンティティを含まないので、2次元図形で表示する。プロシジャと関数は、同じようなプログラムエンティティなので、ともに四面体で表し異なる色を使うことにより区別する。

エンティティ間には、次の依存関係が存在しうる。

#### ① モジュールとモジュール

local-to、uses、instantiates-to、inherits-from

#### ② エンティティとモジュール

injected、imported、exported、inherited

#### ③ エンティティとエンティティ

delimited-by、of-type、parameter-of-type、used-within

これら12の依存関係のそれぞれが色付けされる。

システムは、階層グラフ構造のデータベースを読み込んで、最上位レベルのモジュール、関数、プロシジャ、変数、定数、型をデフォルト位置に表示する。デフォルトが不適当であれば、これらの位

置を後で変えることができる。デフォルトの位置は、位置の変更が簡単に行え、ビジュアル化が簡単に行え、しかもどのエンティティも隠れず、不明瞭にならないように決定される。標準的な右手3次元座標系を使って、すべてのモジュール、プロシジャ、関数を正のyz平面内に、すべての変数、定数を正のxz平面内に、すべての型をxy平面内に位置付ける。デフォルトの視点は、xyz座標系の原点と点(1,1,1)を結ぶ直線の延長線上に置く。

画面は、三つの領域に分けられる。メニュー領域が右側に、凡例を表す情報域が下部に、大きな表示領域が中央に位置する。メインメニューとビューメニューという2種類のメニューがある。メインメニューは、インターコネクショングラフの図形表示を操作するためのものである。ビューメニューは、視点の位置、ズームファクタのような3次元ビューイングパラメータを変更することができる。すべてのインタラクションはマウスを使って行うことができる。

3次元ビューイングモデルは、ズームレンズ付カメラの考え方を基礎としている。カメラは、あるベクタによって決まる方向に置かれ、遠近射影面はこの視線ベクタに垂直に置かれる。ビューイングの角度は、どの程度の角度のビューを画面上の表示域にマッピングするかを決める。小さなビューイングの角度は強力な望遠レンズに相当し、大きなビューイングの角度は遠近が誇張される広角レンズに相当する。ビューイングアップベクタを変化させることによって、カメラをビューイングの方向ベクタの回りに回転させることができる。ユーザのインタラクションを容易にするため、正のz軸が常に上方を向くようにビューイングアップベクタを固定している。オブジェクトを大きく見せたり、小さく見せたりするために、カメラの方向ベクタに沿って視点を近づけたり、遠ざけたりすることができる。ビューイングの角度を変えることにより、ズームイン、ズームアウトしてもオブジェクトを大きく見せたり、小さく見せたりすることができる。図2.3.5-1では、プロシジャを要素として持つモジュールがオープンされている。

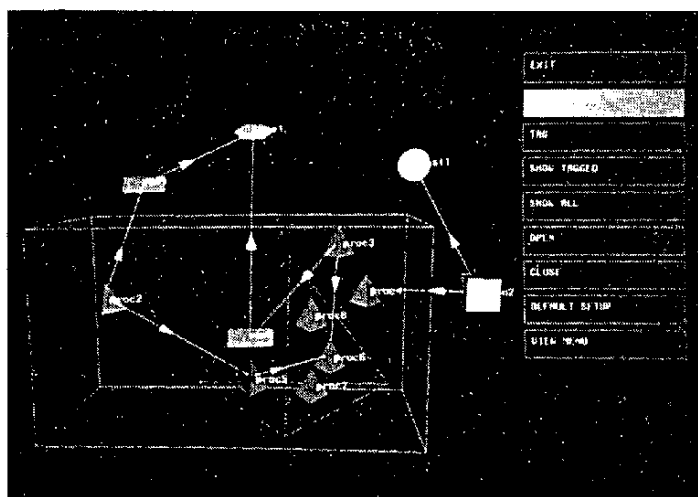


図2.3.5-1 表示例

#### (4) 結論

このツールを使うと、大量のプログラム情報をユーザにすばやく伝えることができる。ユーザは、表示される情報の量を制御したり、大きなデータベースを使って分類することが容易にできる。さらに、ソフトウェアシステムの内部の働きについてとらえ難い点を、ソースコードのリストを読むより、はるかに速く見つけることができる。3次元表示は、2次元表示よりもより多くの情報をより分かりやすく表示することができる。遠近の状態の変更、スケーリング、位置変更によって、ユーザはプログラムの内部の働きを詳しく知ることができる。

### 2.3.6 3次元コンピュテーションビジュアライゼーション

原文タイトル： Three-Dimensional Computation Visualizaion

著 者： Ohn T. Stasko

Joseph F. Wehrli

(Graphics, Visualiation and Usabiliy Center College of Computiong Georgia

Institute of Technology)

出 典： IEEE Symposium on Visual Language, pp.100-107, 1993

#### (1) 概要

本論文は、計算処理やプログラムの視覚化のために、アプリケーション固有の情報を3次元で表示する利点とその重要性について述べている。また、特別なクラス（分野）の計算処理に対する3次元表示がなぜ必要なのか、厳密には3次元表示は必要としないビューに対して、どのようにして有益な情報を表示するのかについて述べている。さらに、3次元アニメーション方法論やPolka-3Dと呼ばれるツールキットを紹介し、これを使用して開発したアニメーションの例について述べている。このシステムはソフトウェアの視覚化を行うのに適しており、システムを利用するにあたって、3次元表示の知識を必要としない。

#### (2) 3次元計算処理の視覚化のビュー

##### (a) カテゴリ

3次元計算処理の視覚化には三つの広義のカテゴリがある。

① 2次元視覚化を修飾したもの

これは、本来二つの空間次元のみで表示可能ではあるが、見ためや表現上の問題のためだけに、三つめの空間次元を追加することである。例えば、2次元棒グラフに一定の奥行きを与えて、3次元の立方体に変更する場合である。

② 本質的に3次元なものの視覚化

このカテゴリは、本質的に3次元エンティティを含む処理である。これには、非平面グラフアルゴリズム、分子構造シミュレーション、立方（3次）の並列アーキテクチャの視覚化などがある。

③ 2次元視覚化に補助的な次元を加えたもの

①と同様、このカテゴリも最低二つの空間次元を必要とする。三つめの空間次元は処理やデータの補助的な特性や属性を表現するために用いる。つまり、アルゴリズムについての情報を2次元の場合よりも多く提供するために、3番目の空間次元を使用する。例として、分散プロット形式のソーティングアルゴリズムの表現について考えてみる。アルゴリズムを表現するには二つの表示次元で十分である。要素の水平位置は配列へのインデックス、垂直位置は各配列要素の相対値である。したがって、ソート後は左下から右上方向の対角線上に点が並ぶ。さらに、履歴を表現するために奥行き（Z次元）を用いることを考える。二つの要素がソート時に入れ替えられるごとに、対角線に沿って向かい合わせとなる二つの点を使って平面を描く。各平面は一つ前の平面の少し後ろに置かれ（Z次元）、最初の平面は2次元上の分散点のすぐ後ろに置かれる。

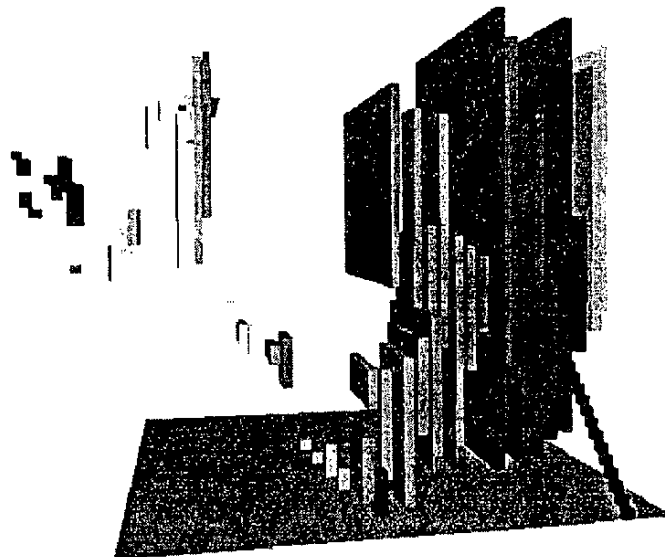


図2.3.6-1 クイックソートの3次元分散プロットのビュー

図2.3.6-1では、右方向の点は2次元ビューでの配列値を示し、左方向の平面は入れ替えの履歴を示す。

三つのカテゴリは、3次元計算処理の視覚化を分類する有効な方法として役に立つ。カテゴリの①と②のアニメーションの開発には、一応、標準的技法が存在している。一方③については、これから多くの新しい視覚表現が開発され、3次元表示の価値をさらに高めていくと考えられる。

#### (b) 次元の使用方法

2次元での計算処理の視覚化では、設計者は、計算処理の情報の表示のために、二つの空間次元（垂直と水平）と一つの「仮の次元」を用いる。「仮の次元」には、色とか形がある。

3次元を用いた計算処理の視覚化では、次元は以下の項目の表現として用いられる。

##### ・ 値 (Value)

要素の値を表すために用いられる。例えば、プロセッサ数、配列のインデックス、変数の値などを表す。

##### ・ 履歴 (History)

計算処理の履歴や時間による状態変化などの表現に用いられる。例えば、プログラム実行の各ステップで送られたメッセージを、時系列的なグラフィック表現を用いて表示する。

##### ・ コネクション (Connection)

平行に並べられた一連の平面に表された、様々な計算処理の属性や操作の集合を表示する。例えば、Z次元上に並んだ一連の2次元平面は、プロセッサ利用の最適化、最近のメッセージ交換量の平均値、注目している変数の値といった並列計算処理の様々な特徴を表現するのに用いることができる。

##### ・ 見栄え (Aesthetics)

特別な目的は持たないが、アニメーションの見栄えを改善したり、洗練したりするために用いられる。

次元による表現方法は、計算処理の視覚化を設計する人が、3次元イメージをいかに利用するかを確認するのを助ける。例えば、2次元マトリックス上の値を操作するプログラムについて考えてみる。二つの空間次元を位置情報を表現するために用い、3番目の次元は2次元マトリックス上の要素の値を表し、時間はプログラムの状態（履歴）を反映している。要素のリスト（1次元）の操作を行うプログラムが、値と最新の変更時刻といった二つの関連する属性を持ち、それらを表示することもある。

#### (c) 欠点

計算処理の視覚化に3次元グラフィックスを用いることに、欠点がないわけではない。ほとんどの

計算処理の視覚化は、プログラムの実行と並行して表示を行う。したがって、いかなるアニメーションも、レンダリングによる大幅な遅れのない、疑似リアルタイムでなければならない。

3次元計算処理の視覚化の動作で、解決しなければならない問題の一つとして、ナビゲーション制御の問題があることが分かった。これは、その情報に適した最適な角度や投影法を明らかにするために、表示を調整する手段を観察者に提供することである。アニメーションが動作するに従って、様々な外観を確認するためにビューを切り替える。ナビゲーション制御を行うために、ダイヤルとスクロールバーを設けたが、それだけでは不十分であった。別の解決方法として、計算処理の表示に効果的なfly-byと呼ぶ、特別のビューイングシーケンスのプログラムを事前に作ることがある。しかし、これはインタラクティブではない。

さらに、2次元アニメーションの基本的な問題は、3次元アニメーションではより複雑な問題となる。それは、情報を効果的に表現するビューをいかに設計するかである。新しい計算処理のビューを開発することは、依然、設計者の次元を表現する技術（手法）や多大な創造力、表現能力や視覚化の技術に負うところが大い。

### (3) 視覚化システム

独自の計算結果の視覚化ツールキットを開発した。開発には、Polkaと呼ばれる新しい2次元アニメーション手法を3次元環境に適応する方法をとった。2次元PolkaシステムはXウィンドウシステム上でC++を用いてインプリメントされており、そのアニメーションの手法は、2次元アルゴリズムアニメーションのための簡単なパスベースの手法と、より洗練された高度グラフィックスシステムのフレームベースのアニメーションクロック手法との複合構成となっている。

既存のアルゴリズムアニメーションシステムでは、プログラムでなんらかの事象が発生した場合、その事象に従ってアニメーションが行われる。例えば、ソーティングアルゴリズム上での二つの値の論理的な交換は、その値をモデル化している四角形の入れ替えとして表現される。このモデルは、連続動作をする持続性を持ったアニメーションとしてうまく動作する。

しかし、ユーザが並列計算処理の視覚化を行おうとする場合には、このモデル化では適応できない。並列計算処理では、様々な並列アクティビティの組合せが、実行時にたくさん発生すると考えられる。例えば、並列ソーティングアルゴリズムでは、多くの交換が同時に発生する。事前に個別のアクティビティ（ソーティングプログラムでの個別の交換）のプログラムを作成し、その後、実行時のパラメータに応じて、これらのアニメーション単位を結合するためのスケジューリング手法を利用する。したがって、最終的に計算処理の視覚化は、最新のアニメーションシステムのスケジュールベースの技術に適応しなければならないことを意味する。Polkaは計算処理の視覚化を行うために設計された最小構成の操作環境の中で、このようなアニメーション技術を表現した最初のシステムであると思う。

---

## 2. ビジュアルインタフェースの技術動向

上記の3次元ツールキットを作成するために、Polkaのデータ形式と操作に3次元の拡張機能を付け加えた。例えば、Polkaの線、四角、円、文字等のアニメーションオブジェクトに立体、球、円錐、円柱が追加された。

Polka-3Dは、ビューポイントの位置と方向を制御する特別なEyeオブジェクトを持つ。この機能により、受動的に分断されていたプログラム実行結果のビューを繋げて動くアニメーションを作成することができるようになる。例えば、単にグラフ検索を見るというよりは、グラフの中で視点を移動しながら検索を実体験することができる。

#### (4) 仮想環境

Polka-3Dの基本的なグラフィックスサポート環境によって、アニメーションを仮想環境へ適応することは比較的容易である。ヘッドマウントディスプレイ、トラック、データグローブを用い、アニメーションの中へ入って動き回ることが可能である。現在のシステムではイメージの応答操作と動作操作は支援していない。既に開発した有益なビューとして、パラレルキューブアーキテクチャの例がある。観察者は、キューブ内に足を踏み入れて様々なプロセスのパフォーマンス測定を行うことが可能である。

#### (5) 結論

計算処理のビューを表現する3次元グラフィックスの価値を明らかにし、3番目の空間次元をいかに利用すべきか、その特性を述べてきた。また、3次元を用いた計算処理の視覚化の例をいくつか示した。これからの研究は、利用者が計算処理を理解する際、3次元視覚化がそれをいかにうまく手助けするかを調査することである。例えば、プログラム方法論を理解する上で、3次元棒グラフのソーティングビューのような、2次元ビューを3次元化したものを、2次元ビューと比較して優れているかどうかを知ることは興味深い。また、我々は、様々なソフトウェアエンジニアリングツールから自動的に作成できるような、標準的なプログラムビューを作成することも計画している。また、ツールキットからコーディングによって開発するのではなく、3次元アニメーションビューの作成を直接操作で行うことができるツールを開発していきたいと思っている。

### 2.3.7 抽象データの3次元ビジュアライゼーション

原文タイトル： A Framework for Abstract 3D Visuarization

著 者 : Steven P.Reiss

(Department of Computer Science, Brown University)

出 典 : IEEE Symposium on Visual Language, pp.108-115, 1993

#### (1) 概要

本論文は抽象化データ、特にプログラマに関係ある情報を3次元で表示するために開発した、PLUMと呼ばれるパッケージについて述べている。目標は、どの情報をどのように表示するかを、プログラマが簡単に指定することができるシステムを提供することである。これを実現するために、宣言型ビジュアル言語と宣言型テキスト言語の両方を持つ枠組みを設計した。

#### (2) 抽象グラフィックオブジェクト

抽象グラフィックオブジェクトは、抽象オブジェクトからのマッピングの目標であり、具体的な3次元表現を生み出すための出発点である。そのためには、次の基準を満たす必要がある。

- ・完全性
- ・拡張性
- ・階層化
- ・パラメータ化
- ・抽象化
- ・具体化

抽象グラフィックオブジェクトは、PLUMの中で属性、制約、コンポーネントを持ったエンティティとして表わされる。属性は、オブジェクトをどのように描くかを制御するパラメータを表わす。制約は、オブジェクトをお互いに関連付ける。コンポーネントは、このオブジェクトに含まれる他のオブジェクトを識別する。

汎用的な属性の集合は、すべてのオブジェクトで共有される。これらには形、サイズ、優先順位の設定が含まれる。優先順位は、あるオブジェクトは重要で強調表示の必要があることを指示する。形属性は色、文字、線、塗りつぶしを含む。サイズ属性は、3次元空間に依存しないようにオブジェクトのサイズを拡大する。

基本の抽象グラフィックオブジェクトには次のようなものがある。

##### ① データオブジェクト

3次元空間上で、図形とオプションの文字列を持つ箱として表現される。図形は2次元または3

次元である。データオブジェクトは制約とコンポーネントオブジェクトを持たない。

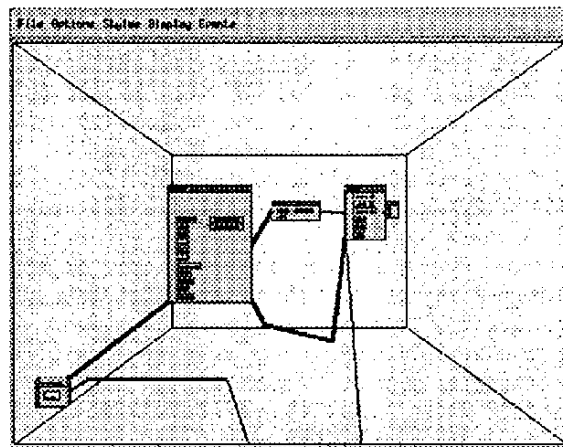


図2.3.7-1 呼び出しグラフの表示例

図2.3.7-1は、PLUMでの3次元の呼び出しグラフ表示の例である。四角い箱はファイルを表わし、小さな箱はルーチンを表わしている。アークは呼び出し関係を表わしている。

② アークオブジェクト

二つのオブジェクト間の連結を表わす。アークオブジェクトは常に、親と呼ばれる他のオブジェクトのコンポーネントである。図2.3.7-1の中で、アークの線の幅は連結の数を示している。

③ タイルオブジェクト

連結によりグループ化されたオブジェクトを表わす。タイル張りは、直線で囲まれた領域とタイルで構成される3次元の箱からなる。図2.3.7-1は、ファイルボックス部分に単純なタイルオブジェクトを用いている例である。ファイルボックスは、ファイル名を持つデータオブジェクトと、そのファイルで定義されているルーチンを含むレイアウトオブジェクトの、二つのコンポーネントを含むタイルである。

タイルオブジェクトは、制約関係を解析して、コンポーネントのレイアウトを決める。制約は、コンポーネントの大きさや座標空間の配分などにより決まる。

④ レイアウトオブジェクト

直線で囲まれた領域で、ノードとアークの二つのコンポーネントを含む。アークによる連結情報をもとに、オブジェクトは領域内に収まるようにレイアウトされる。レイアウトオブジェクトの属性により、レイアウトは制御される。各コンポーネントの相対位置を決定する場合、属性として設定される経験則を用いて、レイアウトを決める。図2.3.7-1では、ファイルボックスの主要なコンポーネントは、2次元レイアウトオブジェクトである。2次元レイアウトオブジェ

クトは、ファイル内のルーチンに関するデータオブジェクトとアークを要素として持っている。

コンポーネントはファイルを表わすタイルとアークの集まりである。

これまで述べたオブジェクトだけでは実現できない視覚化表現を扱うために、抽象データオブジェクトの2次集合を作成した。

#### ① タグ付けされたオブジェクト

ちょうつがい (hinge) 的メカニズムを用いて、別のオブジェクトや内容に結び付けられたタグオブジェクトである。タグ付けされたオブジェクトは、タグに関連する情報が一時的に隠れているときでも、3次元表示の中で利用者がタグを見ることができるようにしている。

#### ② 分散プロット

3次元の座標値に基づくコンポーネントを持つ直線で囲まれた領域からなる。分散プロットのコンポーネントは任意のオブジェクトである。

#### ③ マーカ

単純なマーカである。分散プロットで用いるために、データオブジェクトの形を単純にした。

#### ④ 時間列

3次元の使用法の一つは、Z軸上で実行の履歴を表示することである。時間列オブジェクトは、これを実現する方法の一つである。時間列オブジェクトは、オブジェクトのX、Y座標を識別する基本コンポーネントと要素の集合からなる。

#### ⑤ ファイルオブジェクト

ファイルオブジェクトは、ファイルの抽象ビューである。ファイルは、ファイルの1行が領域の1行に対応している平面領域として表現される。

前述の汎用的なオブジェクトに加え、その他の既存の3次元表現を反映するオブジェクトのインプリメントを計画している。これには以下のものがある。

#### ① コーンツリー

ツリーの3次元表現としてXeroxで開発された表現手法である。

#### ② 透視オブジェクト

透視壁と呼ばれるXeroxで開発されたアイデアを取り入れたものである。

#### ③ 汎用配列

このオブジェクトは、配列のような構造を3次元表示する。汎用配列では、各コンポーネントが、それぞれの次元で独自の座標を持つ。例えば、あるブロックが発生したメモリ内のページをY軸にとり、X軸にページ内の位置、Z軸にそのブロックの寿命を対応させ、それを3次元表示するなどの利用方法が考えられる。

### (3) 抽象グラフィックオブジェクトのための支援

PLUMは、オブジェクト指向のパッケージである。抽象グラフィックオブジェクトを表現するために、次に三つの基本クラスがある。

- ・ PlumObject . . . . . 視覚化オブジェクトを表現する
- ・ PlumComponent . . . . . コンポーネントオブジェクトを表現する
- ・ PlumConstraint . . . . . 制約を表現する

PLUMとアプリケーションのインタフェースは、これら三つのクラスを基に作成される。アプリケーションは、適切な抽象グラフィックオブジェクトを表現するために、新しいPlumObjectを生成する。その後、PlumObjectオブジェクトのメソッドは、PlumComponentオブジェクトとPlumConstraintオブジェクトを作成することによって、PlumObjectオブジェクトを相互に関係付ける。

抽象グラフィックオブジェクトが表示されるまでには、三つの段階がある。最初の段階では、各オブジェクトの望ましい大きさを決定する。次の段階ではレイアウトを処理する。変換処理の最終段階は、オブジェクトを描くことである。

2次元表示空間上での3次元表示のような錯覚は、動きによって誇張される。PLUMの構文的インタラクションで、この機能が一部実現されている。アニメーション機能により、情報の別の面を見ることができる。表示を更新するために、アプリケーションはオブジェクトの属性を変えたり、コンポーネントや制約の加除や、新しいグラフィックオブジェクトを作ったりする。

### (4) レイアウトメソッド

2次元表示のグラフィックレイアウトには、様々な手法が試みられてきた。問題はノードとアークの配置にあり、いかに美しい満足のいくグラフを作り出すかである。様々な試みの結果、2次元のプログラムデータに適したアルゴリズムは、レベルグラフ法の一つであった。

グラフのレイアウトアルゴリズムを2次元から3次元へ拡張するのは容易でない。最初の問題は、3次元空間では、どの表示方法が分かりやすいかということである。

2番目の問題は、3次元空間は2次元空間に比べて自由度が大きいことである。2次元空間のレベルグラフは、レベルを行または列のいずれかで表現するように配置すればよく、グラフの向きを除けば、行と列は同一と見なすことができる。一方、3次元空間では、レベルを表現するのに三つの方法がある。さらにレベル処理が終わった時点では、それぞれのレベルは潜在的に平面で表わされる。

3番目の問題は、情報伝達のために3次元空間を使用したいのであって、単にレイアウトによって大きな空間を与えるためではないということである。つまり、基本となるオブジェクトの属性を反映できるレイアウトメソッドを見つけなくてはならないということである。例えば、実行時間やどんなファイルが入っているか、ユーザが注目する選択ノードからはどれくらい離れているかなどに応じて、

Z座標をノードに割り当てなければならない。

PLUMでは、3次元レイアウトに様々なアルゴリズムを用いて、その善し悪しを確認することができるよう、柔軟な実現方法を採用した。

#### (5) PLUMでの経験

FIELDプログラミング環境によって提供されるプログラム情報とプロトタイプブラウザを用いて、3次元呼び出しグラフの視覚化の基礎システムとして、PLUMを使用してきた。システムは、FIELDクロスリファレンスデータベースより得た静的呼び出しグラフに関する情報を集める。さらに、データベースのグローバル変数の情報、デバッガの動的呼び出し情報、そして、プロファイリングインタフェースツールからのプロファイル情報をこれに加える。

プロトタイプブラウザは二つの基本機能を提供する。1番目は、生成された図に、ある意味論的インタラクションを与えることである。これによってユーザは、階層構造でノードの拡大/縮小を行ったり、表示からノードを除いたり、特に関心のあるノードを選んだりすることができる。これは、PLUMの高次元のインタラクション支援機能の簡単なテストとなる。2番目に、ブラウザは異なる視覚化表現を試すためのプロトタイプ機能を提供する。PLUMは、数千のオブジェクトからなる比較的大きな表示を扱うように設計された。現在までのPLUMの実験では、まだ数百ノードの視覚化を行っただけにすぎないので、今後、規模の拡大を行ったときの問題を推測することはできない。

我々の実験は、PLUMが3次元でプログラムの視覚化を行う能力を持った柔軟なパッケージであることを実証した。それは異なった視覚化の実験にも確かな基礎を提供した。例えば、PLUMの機能を利用することで、タグレイアウト処理は600行以下のコードで実現できたし、分散プロットは400行以下で実現できた。PLUMは、自動アニメーションがプログラム視覚化に有効であることを証明した。

#### 2.3.8 3次元ダイレクトマニピュレーションの設計

原文タイトル： Iterative Design of an Interface for Easy 3-D Direct Manipulation

著 者： Stephanie Houde

(Human Interface Group, Advanced Technology Apple Computer, Inc.)

出 典： CHI'92 CONFERENCE PROCEEDINGS, pp.135-142, 1992

## (1) 概要

本論文は、マウスを使ってオブジェクトを操作する新しい方法を見つけるために繰り返し行った設計とテストの経過について述べている。この方法は実世界の物体の操作方法から直感的に類推できるものである。なお、3次元デザインアプリケーションは、既に多くのものが開発され、広く利用されるようになったが、3次元オブジェクトを操作するためには、専門知識を必要とし使いにくい。

## (2) 設計の基本方針

まず、6人の被験者に机や椅子等のサンプルモデルを並べ変えさせる実験を行い、彼等がとった行動パターンを分析することから始めた。これによって、以下の考察が得られた。

- ① サンプルモデルの扱い方は被験者により多様である。したがって、3次元オブジェクト操作には柔軟な機能が要求される。
- ② ほとんどの3次元オブジェクト操作は次の三つに分類することができる。
  - ・床に沿ってスライドさせる
  - ・オブジェクトを持ち上げる
  - ・回転させる
- ③ 被験者は、オブジェクトとして表わされた家具の本来の性質から、その動きを推測することができる。

この考察を基に、次のようなユーザインタフェースの目標を設定した。

- ① スライド/持ち上げ/回転の各モードのスムーズな切り替えが要求される。
- ② オブジェクトの操作に対する反応は、模倣した実際の家具の機能にできる限り近くなければならない。
- ③ オブジェクト操作の方法は、どのようなユーザにとっても分かりやすいインタフェースでなければならない。

## (3) 2次元入力の3次元へのマッピング

マウスのみを用いて3次元オブジェクトを操作する場合、スライド・持ち上げ・回転をどのように表現するか、またそれらの操作をどうやって切り替えるかが問題である。

一般に用いられる手法としては、X、Y、Z軸に対する回転・移動などのツールパレットを用意してオブジェクトを操作するという手法がある。

## (4) インタフェース設計の模索

### (a) ハンドカーソル

プロトタイプの開発に当たり、まず実際の行動を模倣するハンドカーソルを用意した。このプロトタイプを用いてテストを行ったところ、以下のような問題点があった。

- ・準備状態から、持ち上げ状態への移行は問題ないが、スライドや回転の操作がスムーズでない。
- ・被験者は、この2次元表示のハンドカーソルは、3次元空間で実際の手と比べて動きが不自然であると感じた。

#### (b) アクティブエリア

上記の問題を解決するため、目に見えないアクティブエリアを各オブジェクトに定義した。「準備状態」ハンドカーソルでオブジェクトをクリックすると、そのオブジェクトの形に合わせて、ハンドカーソルはスライド、持ち上げ、回転のいずれかのカーソルに変わる。この方法では、スライド、持ち上げ、回転の切り替えをスムーズに行うことができた。だが、オブジェクトに定義されたアクティブエリアの場所がユーザによく分からないという別の問題が発生した。このため、ユーザはアクティブエリアを探してオブジェクトのいたるところを盲滅法クリックすることになった。

#### (c) ナレーティブハンドル

そこで実際の操作に近づけるために、矢印カーソルでオブジェクトをポイントするとスライド/持ち上げ/回転を表わす「手」が現われるという、ナレーティブハンドルを設けた。この手法により回転と移動間のモードの移行が容易になった。

しかし、まだオブジェクトに対する直感的な位置付け操作の方法は解決されていない。また、椅子のように、「手」のナレーティブハンドルではうまく操作できないものもある。

#### (d) バウンディングボックスを用いたダイレクトマニピュレーション

複数のオブジェクトから特定のオブジェクトを選択するためには、バウンディングボックスを用いるのが一般的手法である。このバウンディングボックスにナレーティブハンドルを組合せることにより、オブジェクトの選択や操作を容易に行うことを可能にした。

#### (5) ユーザテスト

前述のようなユーザインタフェースに対し、ユーザがどのような行動をとったかを観察した結果をまとめると次のようになる。

##### (a) 認識テスト

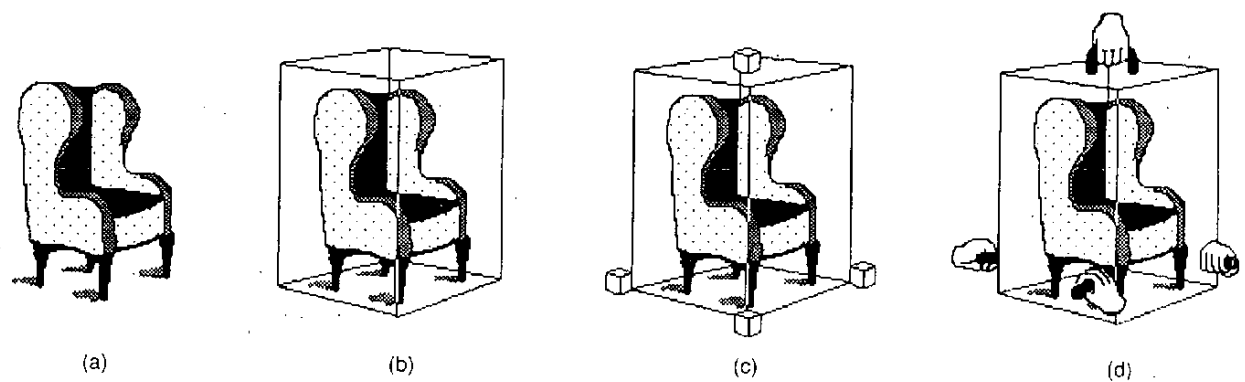


図2.3.8-1 椅子のみのイメージ(a)、椅子とバウンディングボックス(b)、キューブシェープハンドルの付いたバウンディングボックス(c)、ハンドシェープハンドルの付いたバウンディングボックス。ユーザの認識テストでは以上の四つを用いた。

#### ① 椅子のみ

椅子を回転させるために、ほとんどの被験者が椅子の角を選択し、水平方向にドラッグした。また、床に沿って移動させるためには椅子の中央を選択した。さらに、被験者からは「椅子は重力の法則に基づいており、操作はそれに従って行った。」という意見があった。

#### ② バウンディングボックス

ほとんどの被験者が、バウンディングボックスを選択の指示と認識し、椅子を操作するために使うものと仮定した。そして回転操作のためには、ほとんどの被験者がボックスの角あるいは縁をクリックした。持ち上げとスライド操作には、半数が面を、半数が中央をクリックした。1人を除き、操作を遂行することができた。

#### ③ キューブシェープハンドル

被験者は、キューブシェープが「移動」、「拡大」、「その両方」のいずれの操作を行うかを判断できなかった。持ち上げの操作は問題なかったが、下部のキューブシェープを使用した回転操作には混乱が生じた。ある被験者は椅子の中心の回りに回転すると考え、他の人は反対側の角を軸にして回転すると考えた。スライド操作については、多様な反応を示した。

#### ④ ハンドシェープハンドル

椅子を操作するために、ハンドシェープを選択する被験者と、ハンドシェープを選択せずに把手そのものを選択する被験者がいた。また、一部の被験者はハンドシェープをクリックすると、対応する操作が自動的に行われるものと考えた。回転操作に関しては、ほとんどの被験者が下部のハンドシェープを選択したが、その他の操作には、多様な反応が見られた。

#### (b) 単一オブジェクト操作テスト

ほとんどの被験者が、椅子を操作する場合には、椅子をクリックしてそのままドラッグした。椅子を選択するとバウンディングボックスが現われるが、ほとんどの人が以前行った認識テストによって、その意味を理解していた。すべての人が容易にスライド/持ち上げ/回転の各操作を行った。

テストの中で特に気がついたことを次に述べる。幾人かは椅子の回転軸が床に垂直な軸以外にもあることに不自然さを感じた。多くの人は、椅子は立っているものであるという常識から、床に垂直な軸以外には回転しないと思い込んでいた。ある人は「椅子の回転軸は一つに制限されるべきである。」と話していた。

#### (c) 複合オブジェクト操作テスト

被験者は、異なるオブジェクトを切り替えながら、簡単に移動操作を行った。スライド操作の時に、オブジェクトの位置につれてその大きさが変化するのを認識し、それを好んだ。

テストの中で特に気がついたことを次に述べる。何人かの被験者は椅子を横倒しにしようとしたが、それが横長であるので横に倒すことができなくても気にかけなかった。被験者はオブジェクトがそれらしく動いてほしいと感じた。

#### (6) 結論

開発とテストを通して、3次元操作インタフェースに重要であると思われる事項を以下にまとめる。

- ・バウンディングボックスを用いた位置のコントロールは、様々な形状のオブジェクトに対して効果的である。
- ・実世界に存在するものを操作する場合、人間が通常操作する位置にコントロールハンドルを置く  
と直感的で分かりやすくなる。
- ・回転と移動の複合モードは、繰り返し行われる位置調整操作をスムーズに行わせることができる。
- ・ナレーティブハンドルを用いることにより、オブジェクトに対してどのような操作が可能なのか  
をユーザに示唆することができる。
- ・3次元操作上の自由度は、できるだけ減らした方がユーザには使用しやすい。

### 2.3.9 容易な3次元ダイレクトマニピュレーション

原文タイトル： Facile 3D Direct Manipulation

著 者： Dan Venolia  
(Apple Computer, Inc.)

出 典： INTERCHI'93 CONFERENCE PROCEEDINGS, pp.31-36, 1993

#### (1) 概要

本論文は、高速レンダリングのためのハードウェア、3次元マウス、3次元インタラクション技術を含む実験的な3次元インタフェースについて述べたものである。本インタフェースでは、拡張マウスによって制御された3次元カーソルによって3次元オブジェクトの直接操作が可能になった。カーソルの先端をオブジェクトの内部に置くと、そのオブジェクトが選択される。"tail-dragging"という方法を使って、オブジェクトの移動と回転を同時にできる。"snap-to"という方法を使って、オブジェクトの位置を正確に調節することができる。

#### (2) 3次元マウス

従来のマウスによる3次元デザインツールのある種のものは、回転などの動作指示のために五つのパレットが用意され、動作モードを切り替えながら操作をしなければならなかった。

そこで、簡単に利用できる2次元マウスを基に、3次元ダイレクトマニピュレーションが可能なポインティングデバイスを新たに開発した。ローラーマウス (roller mouse) と呼ばれるこのデバイスは、通常の2次元マウスの正面に二つのローラーを取付けたものである。このローラーは一つの方向のみの指示が可能で、右利きでも左利きでも問題なく使用できるようにした (3次元マウスの写真は2.2節を参照)。

#### (3) 3次元カーソル

3次元カーソルは、3次元マウスの移動につれて上下左右に移動し、3次元マウスのローラーを動かすことにより近づいてきたり、遠ざかったりする。オブジェクトがカーソルを覆い隠した場合、オブジェクトは半透明で描画され、カーソルが見えるようになっている。3次元カーソルは円錐形をしており、カーソルの先端を基点にして、後述の"tail-dragging"法により、カーソルの指示方向を変化させながら移動する。

3次元マウスの操作を、3次元カーソルポジションの移動へとマッピングする方法としては、次の二つの手法を用意した。一つは、3次元マウスの動きをカメラ視点にマッピングする手法である。ローラーの動きをカメラ視点の前後の動きにマッピングし、マウスの垂直方向の移動をカメラアップ、水平方向の移動をカメラレフト/ライトの変更動作にマッピングする。もう一つはスクリーンセンタマッピングで、これはマウスの動きにつれ、円錐形のカーソルの頂点が普通のマウスのように移動し、ローラーによりスクリーンの中を前後に移動する。被験者は3次元マウスを2次元マウスの延長でとらえ、使いこなした。

#### (4) オブジェクトの選択

カーソルを移動し、カーソルがオブジェクトに触れると、カーソルの尖端からオブジェクトの輪郭に向かってクロスヘア（6方向）が表示される。この「カーソルがオブジェクトに触れる」という動作がオブジェクトの選択となる。

また、クロスヘアはオブジェクト内のカーソル位置を知る手掛かりにもなる。

#### (5) オブジェクトの移動

オブジェクトを移動する際、回転を同時に行うか、あるいは移動のみとするかを指定することができる。この判別は、マウスボタンを押した時点のカーソルの位置により決まる。そして、その結果はカーソルの形状によって示される。カーソルがオブジェクトの輪郭近辺にある場合は円錐形のカーソル、オブジェクトの中心近辺にある場合には6方向を示すジャッキ型のカーソルとなる。

ジャッキ型のカーソルが現われている時は、マウスボタンを押し続けながらマウスを操作することで、オブジェクトは3次元空間を移動していく。マウスとローラーを動かして、ジャッキ型カーソルをドラッグすることにより、3次元空間におけるオブジェクトの移動を一度に行うことができる。

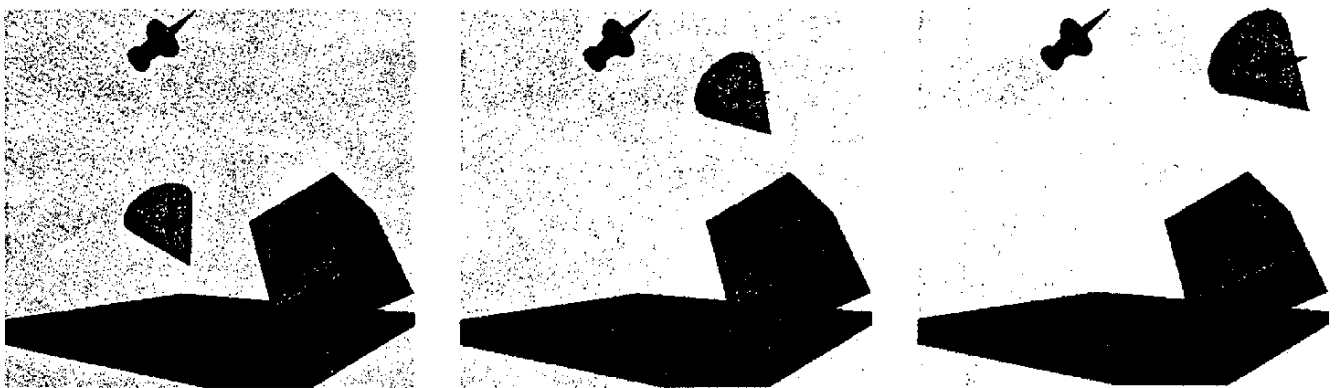


図2.3.9-1 ジャッキ型カーソルによるオブジェクトの移動

#### (6) "tail-dragging"による動作

円錐形のカーソルは、その頂点が3次元マウスに連動して動く。また、底面の中心は、頂点との関係が以下の式になるよう変化しながら、円錐の方向を変えて動く。

$$tail_t = head_t + (tail_0 - head_0) / \|tail_0 - head_0\| * \|tail_t - head_t\|$$

(0 = 基点、t = t時間後の位置、1 = (0 < t) 時間後の位置)

ユーザはこのような動作する円錐カーソルを用いて、オブジェクトを回転、移動させる。円錐カーソルが現われている時に、マウスボタンを押し続けながらマウスを操作すると、オブジェクトは回転しながら移動する。

この方式そのものはあまり使いやすいものではなく、また精度の高いものともいえない。改善は今後の課題である。

#### (7) "snap-to"

正確な位置決めを行うために、"snap-to"と呼ばれる手法を考えた。"snap-to"は、オブジェクトの位置と方向の両方を揃えるのを援助する機能で、磁石のように振る舞う。あるオブジェクトをドラッグして別のオブジェクトに近づけていくと、ある地点でオブジェクトはカーソルの制御から離れ、もう一方のオブジェクトに引きつけられていく。この引きつける力は、オブジェクトが近づけば近づくほど強くなる。そして、オブジェクトが十分近づいた時点で正確な位置に配置される。

"snap-to"はビジュアルなフィードバックを用いると、よりユーザに分かりやすいものとなる。例えば、二つのオブジェクトの間に赤いバネを表示し、両者が近づくとバネが縮んで行くようにする。

#### (8) サウンド

インタフェースの補助機能として、サウンドフィードバックを加えると、一層操作性が向上する。例えば、"snap-to"では二つのオブジェクトが近づくと、音量が距離に比例して大きくなる。また、モードが変更されると音が聞こえたり、カーソルがオブジェクトに触れたり、離れたりとすると音が聞こえるなどの使い方がある。

#### (9) 結論

このシステムは、簡単で自然なインタフェースを達成することを念頭に置いて設計されており、オブジェクトのリアリズムなどは犠牲になっている。また、直接操作を第一の目的にして、例外処理などはあまり考慮していない。

ユーザ操作並びにそのインタフェースの負担の軽減を目標にしたが、その成果は確認できた。複雑な位置揃えコマンドや、それに付随するオブジェクト選択コマンドを覚える代わりに、それらを簡便

化することもできたのではないかと考える。

インタフェースの複雑さを軽減することは、システムの使いやすさと設計の質の向上が望めるであろう。

"tail-dragging"や"snap-to"の技術はすぐにでも応用できると思うが、最終的な目標は、3次元オブジェクトのブラウジングやアレンジング、モデリングなどの機能を含む、完全な形での3次元インタフェースのツールや技法を開発することである。

## 2.4 インタフェースビルダの動向

### 2.4.1 インタフェースビルダとは

計算機のグラフィックインタフェースの作成を支援するいわゆる「インタフェースビルダ」は、近年非常に成功したアプリケーション分野の一つである。「インタフェースビルダ」という名前は、NeXT社のワークステーションに登載されている「Interface Builder」のようなシステムという意味で、ここでは一般名詞として使うことにする。「ウィンドウシステムのツールキット部品を直接操作によりウィンドウ画面に配置しながらアプリケーションの見栄えを設計し、かつツールを操作したときのアプリケーションの動作記述を支援するシステム」を想定している。[暦本93]では「ユーザインタフェースの視覚的・対話的設計システム」としてInterface Builderをはじめとする各種のシステムが紹介されているが、ここでいうインタフェースビルダはそのサブセットである。

一般にインタフェースビルダと呼ばれるものは以下のような特徴を持っているのが普通である。

- ・ 直接操作インタフェースによるグラフィック画面設計とテキスト編集によるプログラミングを同時進行させながらアプリケーションを作成する。
- ・ ボタンやスライダのようなインタフェース部品それぞれに対し、それらの部品が操作されたとき (e.g. ボタンが押されたとき/スライダが動かされたとき) 呼ばれるようなサブルーチンに対応付ける。
- ・ 配置した部品の情報からサブルーチンの雛型を生成することができる。
- ・ 部品の配置情報は特別な配置情報ファイルに格納して利用する。
- ・ サブルーチンをきちんと書く前にアプリケーションの見かけの動作をテストすることができる。
- ・ アプリケーション作成のプロジェクト全体を管理する機能がある。
- ・ インタフェース部品の初期化のためなどのコードは自動的に生成される。

### 2.4.2 実例

現在いろいろなインタフェースビルダが販売/配付されているが、ここでは実例としてNeXT社のInterface Builder、Microsoft社のVisual Basic、Silicon Graphics社のワークステーションで動作するForms Designerを紹介する。

#### (1) NeXTのInterface Builder

NeXT社は1988年より自社製のUNIXワークステーションのウィンドウシステム「NEXTSTEP」のためのGUI構築ツールとして「Interface Builder」をバンドルしており、これが現在の各種インタフェースビルダの元祖となっている。その後NeXTワークステーションのバージョンアップに対応して各種の改良が行われ、現在第3.0版がリリースされているが、基本的な機能は変わっていない。

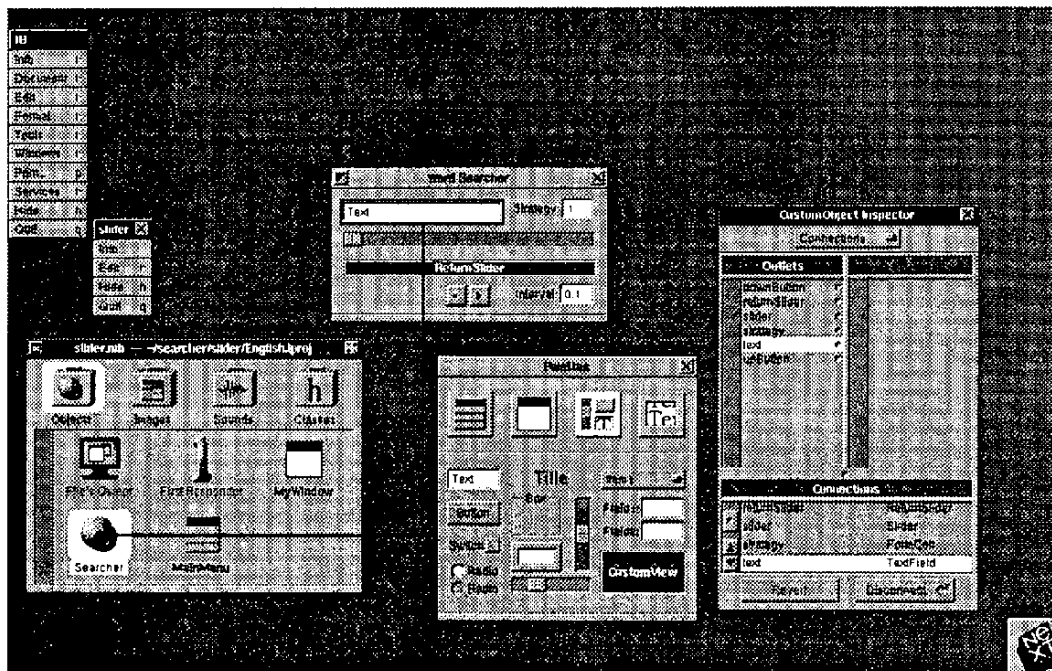


図2.4-1 NeXTのInterface Builder

NEXTSTEPのツールキットはAppkitと呼ばれ、あらゆる部品がObjective-Cのクラスで実現されている。例えばボタンを示すクラスはObject（クラス階層の根）→Responder（イベントに反応するもの）→View（描画可能な領域）→Control（一般のインタフェース部品）→Buttonというクラス階層で構成されている。Appkit自体はXのツールキットにも共通な一般的特徴を持っている。ツールキットのメインループは常にマウスなどのイベントを待っており、各種のイベントが来るたびに対応するクラスの対応するメソッドが呼び出される。例えばボタンをマウスでクリックしたときはButtonクラスのインスタンスにmouseDownというメッセージが送られる。部品が継承可能なクラスで実現されているので、既存の部品のサブクラスを作りメソッドを追加したりすることにより再利用することが可能になっている。このため、例えば既存のスライダを再利用して少しだけ動作の異なるスライダを作ったりすることが可能である。

Interface Builderは以下のような機能をサポートしている。

- ・ Drag & DropによるAppkit部品のウィンドウへの配置
- ・ Appkitのクラス階層の管理
- ・ メソッド、インスタンス変数の管理
- ・ オブジェクト間の接続関係をグラフィックに操作
- ・ 各種リソースの管理

インタフェース部品や一般のオブジェクトのインスタンスは画面上のアイコンやウィンドウなどで表現され、これらの間を線で結んで関係付けることにより、インスタンス変数を初期化したりメッセージの送り先を指定したりすることができる。例えば、オブジェクトAのインスタンス変数iの初期値をオブジェクトBとする、というような設定がiとBを線で結ぶだけで指定できる。

対話的に設計されたクラス定義は、「Unparse」機能によりObjective-Cのプログラムテキストの離型に変換される。ユーザはその後これを編集するが、その際にインスタンス変数の追加などにより前のクラス定義と整合がとれなくなることがあるので、テキストの修正をInterface Builderに知らせるための「Parse」機能も用意されている。最新のバージョンではProject BuilderというシステムによりMakefileなどの資源が管理され、デバッグやテストの機能もサポートする総合的プログラミング環境となっている。

Interface Builderはインタフェースビルダの元祖だけあって完成度は高い。Interface Builderでは視覚的でないオブジェクトである名前や関係もある程度ビジュアルに操作することができ、またProject Builderがプロジェクト管理機能も持っているため、Objective-Cのプログラミング環境として使うことも可能である。

## (2) Visual Basic

Visual BasicはMicrosoft社のWindows用のプログラミング言語/アプリケーション開発環境である。Basicと銘打ってはいるが、いわゆるパソコンBASICとは大きく異なるもので、単純な構造的インタプリタ言語である。

Visual Basicは特定のツールキットに対応して後から開発されたインタフェースビルダではなく、最初から統合的プログラミング環境として意図されたシステムである。ボタンなどの部品はウィンドウの上でClick & Dragによる直接操作により作成/配置し、属性をメニューで設定することができる。部品が操作されるときは、その部品名 (e.g. 「button1」) とその上の操作名 (e.g. 「click」) を組合せた名前 (e.g. 「button1\_click」) を持つサブルーチンが呼ばれるので、配置の後でこれらのサブルーチンを定義することによりアプリケーションが完成する。

Visual Basicはプログラミング言語としては洗練されたものではないが、インタプリタ言語であることからインタフェースの作成/テストのサイクルを迅速に行うことができるし、OLE、DDEのような機能を使ってExcelやWordのような既存のアプリケーションと組合せて使うこともできるし、統合的プログラミング環境としてよくできているため今後着実にユーザが増えると予想される。

Visual Basicはまた、描画のための命令を内蔵していたり、Drag & Dropのインタフェースを標準でサポートしているという特徴も持っている。



図2.4-2 Visual Basic

### (3) Forms Designer

Formsはユトレヒト大学のMark H. Overmarsにより開発されたシリコングラフィックス (SGI) 社のワークステーションで動作するツールキットであり、Forms Designerというインタフェースビルダを装備している。

Formsは比較的シンプルなツールキットである。Xlibなどと異なり、描画などの処理はSGIのGLライブラリを使用するので、Formsライブラリに含まれる関数のほとんどはツール部品を作成/表示したり属性を変更したりするためのものとなっている。

Formsの実行の中心は、`fl_do_forms()`という関数である。これが呼ばれるとライブラリはユーザの入力待ち状態となり、部品に対しなんらかの操作があったときその部品を返り値としてリターンする。Forms Designerは他のインタフェースビルダと同様に、操作ウィンドウ (Formと呼ぶ) の上への部品の配置および属性の設定を支援し、`fl_do_forms()`を含むメインプログラムの雛型を生成する。FormsおよびForms Designerはシンプルではあるが、パブリックドメインのソフトウェアでソースがすべて公開されており、全体のサイズも小さいため見通しがよいという特徴もある。

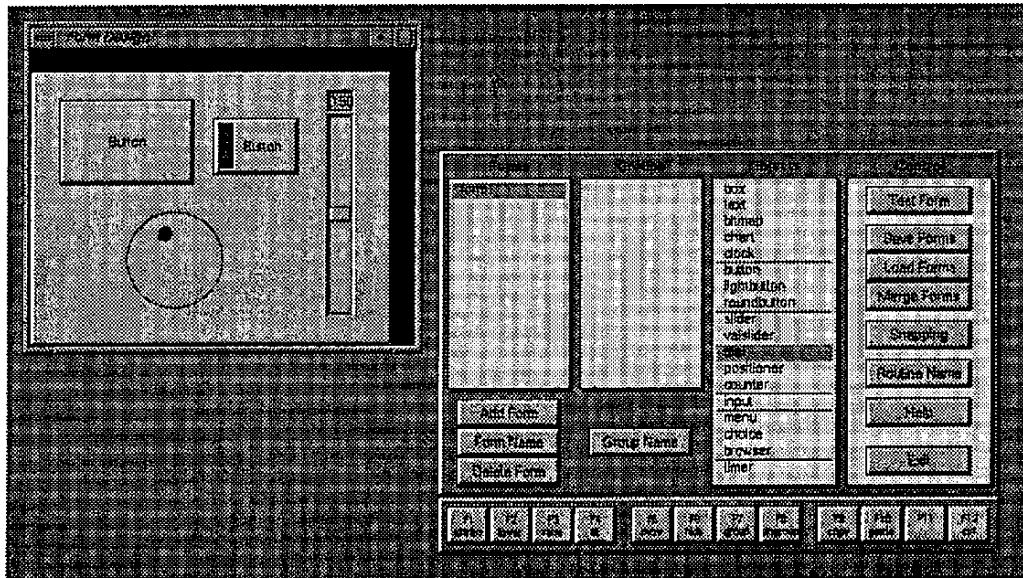


図2.4.3 Forms Designer

#### (4) その他

以上に紹介したものその他、Sunの「DevGuide」やXウィンドウ上の簡易インタプリタ言語「Tcl」とそのツールキット「Tk」で使用する「xf」というインタフェースビルダもよく使用されている。

### 2.4.3 インタフェースビルダの利点

インタフェースビルダが流行しているのは、以下のような利点を持っているためと考えられる。

- ・マウスでツール部品を画面に配置することはテキストで指定するのに比べ非常に楽である。
- ・画面の設計、見栄えのテストが簡単にできる（ラビッドプロトタイピング）。
- ・ビジュアルな部分とテキスト編集部分を相補的に使用している。

画面の配置などはビジュアルに行う方が速く、プログラム本体はテキストエディタを使って書く方が速いという仮定があって、このようになっているわけであろうが、従来のプログラマでもとっつきやすいだろうし、実際大抵のプログラマにとって上の仮定は当てはまると思われる。

- ・インタプリタ言語と組合せて使う場合、特に高速にプロトタイピングができる。
- ・自分自身の作成に自分を使っているものが多く、評価できる。
- ・プログラミング環境として統合化されている。
- ・結果的に素人でもウィンドウプログラミングができる。

このように、インタフェースビルダは実際に役に立つ（世間に広まった）ビジュアルプログラミング

グシステムの代表例とすることができるだろう。

#### 2.4.4 問題点

先に述べたように、インタフェースビルダは数多くの特長を持っているが問題点もたくさんある。そもそもインタフェースビルダがもてはやされた最大の理由は、それ無しにツールキットのプログラミングを行うことが非常にやりにくいことが原因であり、あまり積極的なものではないように思われる。問題と思われる点を以下に列挙してみることにする。

- ・用意されている仕様と異なるインタフェースを作るのが大変

インタフェースビルダに用意されている部品がそのまま利用できれば簡単だが、仕様が違うインタフェースを作ろうとすると非常に手間がかかることがある。このためインタフェースビルダでは、ユーザの様々な好みに応じるために各部品に対してたくさんのオプションを用意しているのが普通であるが、オプションの数が多くなって設定が複雑になるにもかかわらず、一般的でないインタフェースを作ろうとするとオプションでは対応しきれず苦労することになる。ずっと同じインタフェースビルダを使い続けていれば、段々ノウハウが蓄積されるので似たような次の仕事はなんとかできるようになるが、このようなノウハウが非常にたくさん必要なので、他のインタフェースビルダ/ツールキットに移行するのが困難になってしまう。

やり方を思いつけば（知っていれば）簡単に解けるが、それがわからないとどうしようもないという点において、インタフェースビルダやツールキットはパズルであるということができようであろう。同じことを実現するやり方は何通りもあるのであるであろうが、一番良いやりかたを見つけるのは実に難しい。

ほんの少し仕様が違うものを作るのすら大変なのだから、まったく違う部品を作るのは本当に骨が折れるかまったく不可能である。このため多くのツールやオプションが最初から用意されているインタフェースビルダやツールキットをどうしても使うことになってしまう。

- ・使用言語やツールキットなどが制約を受ける

先に紹介したインタフェースビルダは、いずれも特定の言語やツールキット、描画システムと密接に結び付いており、他のものを使うことができない。例えばNeXTのInterface Builderを採用するということは、すなわちUNIX、Objective-C、AppKit、Display PostScriptの採用も意味してしまう。ところが現実にはいろいろな言語を混ぜて使えた方が便利なのが多い。Interface BuilderをLispやPerlのようなインタプリタ言語で使えたら便利だろうし、PostScriptを使いたいと思うVisual Basicユーザも多いであろうが、現状ではこういうことは不可能である。このよう

に既存のインタフェースビルダでは、いろいろなツールや言語を組合せて使うための機能は一般に弱いため、システムを総合的に判断して有利なツールを選択し、うまくいかないところは目をつぶらざるを得ず、良いインタフェースビルダを使いたいために嫌でも指定された言語を使うというようなことになってしまっている。

- 一枚岩の統合的環境になってしまう

インタフェースビルダを中心としたプログラム開発環境は、一枚岩の統合的環境になっているものが多い。実際インタフェースビルダの多くはプロジェクト管理機能を持っており、プログラム編集、コンパイル、デバッグなどプログラム開発における各局面をサポートするようになっている。MESAやSmalltalkに代表されるように、昔はこのような統合的プログラミング環境というものが流行ったものである。近年はUNIXに代表されるような小さなツールの集合体で構成されるプログラミング環境が主流になっているようであるが、インタフェースビルダにより昔のような巨大システムが復活するのかもしれない。

- ビジュアル部とテキスト部の相互変換が面倒

大抵のインタフェースビルダは、ビジュアルに編集した部分をプログラムなどのテキストに落とす機能を持っている。これらの部分の役割の区別が明確ならばよいが、どちらを使っても同じことができることも多いので、しばしば混乱が起こる。またインタフェースビルダがプログラムを生成するときは通常、雛型の生成だけを行い、ユーザがこれを編集して最終的なプログラムを作成することになるが、この方式は以下のような不都合がある。

- 以前修正したテキストを新しい雛型で上書きしてしまう危険がある。
- テキストの変更がビジュアルな部分に反映されない。ビジュアル部とテキスト部の対応を常に保つように注意しなければならない。

Interface Builderでは修正したテキストを構文解析してビジュアル部に反映させる「Parse」機能があるが、テキストを編集するたびにこれを起動するのは面倒である。

また、例えばNEXTSTEPの場合、クラス名/メソッド名/インスタンス変数名などをInterface Builderで指定することができるが、いったん名前を決めてしまうとその影響が多方面にわたるため、スペリングミスなどあったとき後から変更するだけでもかなり大変である。例えばInterface Builderで間違ったクラス名を設定すると、間違った名前のファイルがいくつもできてしまうし、間違った名前がMakefileの中に書き込まれるし、他のファイルから間違った名前前で参照されるし、直すのに相当骨が折れる。またオブジェクト名、呼び出しメソッド名等いろいろな名前をあらかじめ決めなければならないことも面倒である（もちろんこれは普通

のプログラミングでも同様ではある。)。オブジェクト指向プログラミングでは、一般にクラス階層は後で変更しにくいので熟慮して決めなければならないが、このことに加えてビジュアルな指定とテキスト編集による指定が混在しているややこしさや、相互参照のややこしさが加わるためさらに注意が必要で、ラピッドプロトタイピングの敵となる。Visual Basicでは部品名とイベント名を並べたものを関数名として使用するなど、そのあたりを簡略化しているが、そのためプログラミング言語としては変わったものとなっている。

- ・グラフィック操作による作業がテキスト編集による作業より効率が劣ることがある

例えばウィンドウが大量に必要な巨大なアプリケーションを作るのにインタフェースビルダを使用するとき、同じような操作を何度も何度もマウスで実行しなければならないことがある。また既存のデータや別のプログラムからインタフェース画面を自動生成することもできないので、インタフェースビルダでは手工業的作業しかできない。

- ・WIMP (Window-Icon-Menu-Pointer) インタフェースしか考慮されていない

これもツールキットの責任であるが、インタフェースビルダで設計できるのはグラフィックディスプレイを持つ卓上計算機上のペンやマウスを使うインタフェースだけであり、音声を使うインタフェースや仮想現実感 (Virtual Reality) のようなインタフェース作成には役に立たない。

- ・動的な動作を記述できない

インタフェースビルダで設計できることは、実はインタフェースのほんの一部である。インタフェースビルダを使っても図形エディタを作れるわけではないし、アニメーションを記述できるわけでもない。結局インタフェースビルダは静的なツール部品を配置することしかできないわけで、どちらかというところ「レイアウトビルダ」と言ったほうが近いかもしれない。ユーザインタフェースの作成では、静的なものよりも動的な反応の作成が難しいが、インタフェースビルダはそういうものをサポートしてくれるわけではない。昔のUIMSでよく使用されていたような、インタフェースの状態遷移の設計に使うこともできない。

- ・制御構造が固定的

ここで紹介したインタフェースビルダで使用するツールキットは、基本的に複数プロセスを扱うことができない。一方、擬似的に複数プロセスを扱わなければならないことは非常に多い。例えば時計を表示するウィンドウのような簡単なアプリケーションでも、時刻を更新してはそれを通知する時計プロセスと、ユーザ操作を処理するプロセスの二つのプロセスが必要になる

が、ツールキットは複数プロセスを扱えないのでタイマによってイベントや割り込みを発生させるといった細工が必要になってしまう。またツールキットは基本的に常にユーザの入力待ち状態にいたる必要があるため、別の関数やプロセスから来たメッセージに応答できるようにするには制御構造をひと工夫する必要がある。

- ・巨大になりがち

一枚岩のインタフェースビルダは巨大になりがちなので、中で何が起きているのか分かりにくいことがある。内部動作を推測するしかないので、何かを変更する場合、とにかくやってみてから考えようという態度になってしまう。

- ・テスト機能が不十分

多くのインタフェースビルダにはテストモードが用意されているが、アプリケーションはまったく動かさずにテストするわけであるからほとんど意味がないし、アプリケーションからの情報を必要とするインタフェースにはまったく役に立たない。

#### 2.4.5 問題点の解決方法の探索

前述したような問題点を解決するのはかなり大変のようではあるが、それぞれについて解決法を考えてみる。

- ・グラフィック部とテキスト部の整合について

現在一般的なOSの普通のファイルシステムを使っている限り、グラフィック操作とテキスト編集の内容を常に整合させることはかなり難しいと思われる。構文エディタを使って統合システムとすれば少しはよくなるかもしれないが、Visual Basicのような統合的システムでもそれは実現されていない。テキストのみのプログラミングでも同様の問題は起こるのだから、最初から名前をきちんと設計するべきではあるが。

- ・制御構造について

タイマがしばしば必要になったり制御構造をうまく書けなかったりするの、インタフェースビルダのせいではなく、ツールキットの責任である。並行処理をサポートするツールキットが最低限必要である。

- ・統合的プログラミング環境について

All in oneのシステムは、小さなツールを自分で作って既存のツールと混ぜて使うことができないので制約を感じることが多い。統合的インタフェースビルダでは、「ウインドウはこう初期化しろ」「ボタンオブジェクトへはこうメッセージを送れ」といったことをシステムに言われるとおりに書かなければならない。

では、インタフェースビルダはどのように分解すればよいのであろうか。単純に考えて、

- ① ボタン、スライダなどの各部品
- ② 部品を配置するための図形エディタ
- ③ 使用する部品の名前、リンク管理
- ④ プロジェクト管理
- ⑤ グラフィック描画システム

のように分解することはできる。これらを別々のモジュールとしておいて、その上に全体を管理するシステムがあれば、既存のインタフェースビルダと同じように使うこともできるし、高級なプロジェクト管理機構を使うように拡張したり、異なる言語やグラフィックシステムを使うことができるようになるだろう。Interface BuilderやFormsは、グラフィックシステムは含んでおらずユーザが自力でプログラムすることになっているので、⑤は分離されているといえる。DevGuideでは、使用するグラフィックシステムをPostScript、Xlibなどから選択することができるようになっている。

独立した部品を組合せる方法としては、例えばいろいろな部品間で一つの共有空間を使用する方式が考えられる[Masui91]。現在、NEXTSTEPの部品をコルーチンによるスレッドで並行動作させ、Lindaサーバを介して他の部品やシステムと交信するという方式を実験している。このような実装ではObjective-Cによる統合的環境という性質は失われるが、他のシステムと一緒に動かしたりすることは圧倒的に易くなるので総合的には有利であろう。このようにすれば、例えば制御画面の設計はNeXTのInterface Builderで行い、グラフィック描画はSGIのワークステーションに任せるといった部品化が可能である。またインタフェースビルダがWIMPインタフェースにしか対応していないというのは問題でなくなるし、小さなプロセスの集まりとしてシステムを構成することにより見通しがよくなるという効果もある。並列動作が苦手というツールキットの問題ももちろん解決するだろう。

#### ・インタフェースビルダのブートストラップ

先に述べたように、通常インタフェースビルダに用意されているインタフェース方式は限られており、ユーザが簡単に拡張することはできない。しかしインタフェースビルダも一つのアプリケーションプログラムであり、インタフェースビルダの作成に自分自身を使っているものも

多い。ということはコンパイラ作成の場合と同じように、自分にどんどん新しい機能を追加して、次の版の開発には拡張版を使用するといったことが簡単にできていいはずである。これができるば、奇妙なインタフェース方式を開発して既存のインタフェースビルダに追加し、そのインタフェースビルダでは以後その奇妙なインタフェースを標準で使うといったことも可能になる。現存のインタフェースビルダは一枚岩システムなのでこういうことは難しいが、小さな部品の集合として構築すればなんとかなるかもしれない。最近流行の研究結果を取り込んで、「制約解決機構付きインタフェースビルダ」とか「予測実行インタフェースビルダ」といったものを簡単に作れるかもしれない。

また、UI部品の作成にインタフェースビルダを使うこともできるかもしれない。現存のインタフェースビルダはアプリケーションを作成するためのものであるが、部品そのものをさらに小さな部品の組合せとしてインタフェースビルダで設計できれば素晴らしいことである。一つのボタンもアプリケーションと同じように開発するわけである。

再利用はInterface Builderではサブクラス化で、ある程度実現されているが、十分とはいえない。例えば、スライダもダイアルも見かけが違っただけで動作は似たようなものと考えられるが、見かけと動作を分離して動作の部分を利用できるようにはなっていない。見栄えと動作も分離、再利用できるようになればいいであろう。

#### 2.4.6 結論

ここで述べたような様々の問題点は早急に解決するとは思えないが、なんといってもインタフェースビルダは急成長株のアプリケーションであり、Visual Languageの一つの方向として動向に注目しておくことが大事であろう。

#### 【参考文献】

[Masui91] Toshiyuki Masui: User interface construction based on parallel and sequential execution specification, IEICE Transactions, Vol. E 74, No.10, pp.3168-3179 (October 1991).

[暦本93] 暦本純一, 垂水浩幸: ユーザインタフェースとオブジェクト指向, 「オブジェクト指向コンピューティング」第6章, pp.213-244, 岩波書店 (1993).



### 3. ビジュアルインタフェースの事例



### 3. ビジュアルインタフェースの事例

#### 3.1 ソフトウェア開発環境への3次元視覚化の応用

##### 3.1.1 はじめに

膨大な量の情報が溢れる現代社会においては、情報の迅速な理解と取捨選択が非常に重要である。これに対し情報を単に文字の羅列としてでなく適当な図として表示することにより、人間の情報に対する理解を高めようとする研究は情報視覚化 (Information Visualization) と呼ばれ、ビットマップディスプレイを装備したワークステーションの普及により加速された。一方、CAD (Computer Aided Design) の分野では以前から3次元コンピュータグラフィックス (CG) が利用されていたが、近年の3次元CGワークステーションの低価格化により、情報視覚化への3次元の導入が新たに模索され始めている。

3次元CGを情報視覚化に利用した先駆的システムはMCCで開発されたSemNet[Farchild88]である。SemNetはPrologで記述された知識ベースをノード・リンクを基本要素とする3次元グラフとして表示した。そして2次元グラフにおいて一般に回避困難なリンク交差の問題が解決できることを示した。Xerox PARCで開発されたInformation Visualizer [Card91, Mackinlay91, Robertson91] は3次元空間を利用する意義をより明確に示したといえる。例えばCone Tree[Robertson91]は階層データを3次元の木として表示し、物理的制限のある計算機画面上に2次元に比べより大規模な階層構造を表示できることを示した。Perspective Wall[Mackinlay91]は3次元の壁上に表示された線形データを透視図法を利用して見ることにより、局所情報の詳細と大局情報の概要の統合表示ができることを示した。また両者ともCGの動画機能が利用者の認知的負荷を軽減することを示した。この他に、UNIXディレクトリブラウザ[Silicon92]、アルゴリズムアニメーションシステム[Brown93, Stasko93]への3次元の導入も見られる。

これに対しVOGUE (Visualiation Oriented Generic User-interface Environment) はソフトウェア開発で利用される情報に焦点を絞り、2次元と3次元の本質的相違点、つまり「次元の増加」が情報視覚化に与える影響を明らかにすることを目的として設計され、様々な例に応用され評価が行われている[小池90, 小池91b, Koike93a]。ここではVOGUEにおける3次元視覚化の枠組、システム構成、応用例などについて述べる。

##### 3.1.2 VOGUEにおける3次元視覚化の枠組

ソフトウェアには多種多様な”もの”とそれら間の関係が存在し、これらは到底一つの図で表現し得ない。悪戯に多くの情報が描画された図は、逆に人間の認知を妨げるからである。現在、視覚化システムの多くは、マルチウィンドウ環境内に実現され、ソフトウェア情報はユーザの視点ごとに、

各々別な図として提示されるのが一般的である。しかし、これには以下のような問題点が存在する。

#### ・2次元図の表現力の限界

描画される図形要素間に存在する単純な順序関係は、一般に1次元で表現可能である。したがって、直交座標を用いた2次元表現においては、二つの順序関係を記述することが可能である。例えば、現在一般的に利用されているプロセス実行状態の視覚表現[McDowell89]は、y軸に各プロセスを並べ、x軸を時間軸にとるが、プロセス間に何らかの平面的関係が存在する場合、これを1軸で表現することは不可能である。また、後述するオブジェクト指向言語のクラス・メソッド関係を表現する場合、クラスに所属するメソッドの並びは、2次元の表として表現できるが、クラス集合は階層構造をなし、これを自然に表現するためには、やはり自由度不足が生じる。

#### ・メンタルモデル再構成の負荷

視点ごとに異なる図を用いる場合、同一の実体を持つ「もの」が異なる図として表現されることがしばしば起こる。図は人間のメンタルモデル形成を支援するのみならず、強制的にメンタルモデルを与えるため、ユーザは各々の図から異なるメンタルモデルを獲得する。そして、ソフトウェア全体の理解に到達するためには、これらのメンタルモデルを一つのメンタルモデルに再構成しなければならない(図3.1-1)。

以上の問題点は、VOGUEの提案する3次元視覚化によって解決することが可能である。3次元図形は一つの視覚表現に対し複数の視点が存在する。図形要素を3次元空間に適当に配置することにより、見る方向を変えることで異なる関係を把握することが可能である(図3.1-2)。

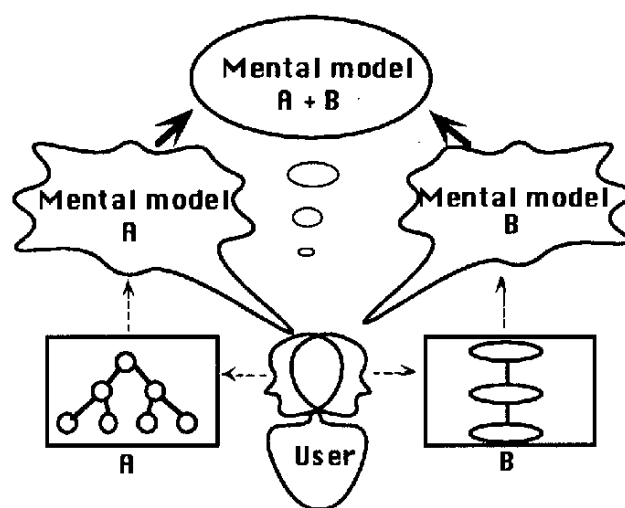


図3.1-1 メンタルモデル再構成

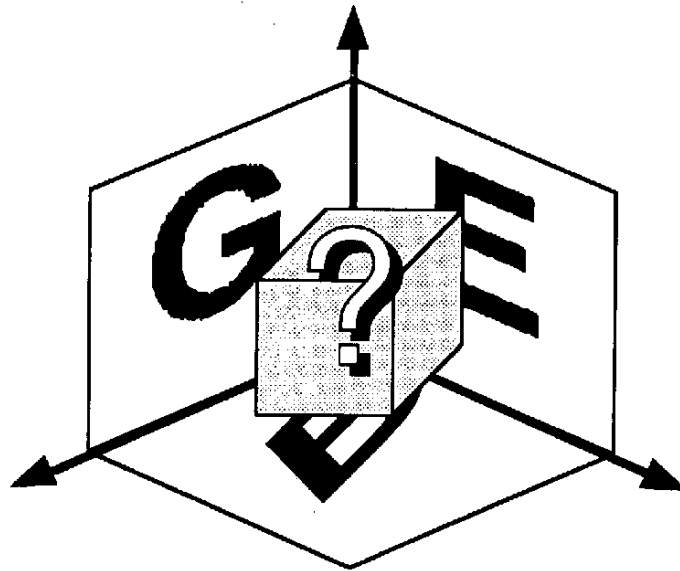


図3.1-2 3次元モデルから得られる複数のビュー

この枠組を用いると、複数情報をユーザに一度に提示でき、与える視覚表現はあくまでも一つであるから、メンタルモデルの再構成は必要ない。しかも、視点変更によって、ユーザは着目する関係に焦点を移動できる点が特に重要である。また、描画空間の自由度が一つ増加することにより、先述したプロセス間の平面的関係やクラス間の階層関係を、自然に記述することが可能となる。

### 3.1.3 VOGUEの特徴

VOGUEは、様々なソフトウェア情報を視覚化表示するためのプラットフォームとして開発された[小池90]。現在VOGUEはSilicon Graphics社製IRISワークステーション上でC++ (GNU C++) とグラフィックスライブラリGLを用いて実現されている。

VOGUEの特徴としては以下が挙げられる。

#### ・汎用性

VOGUEは表示方式にノード・リンクを基本要素とする3次元グラフを採用した。結果としてVOGUEは後述する様々なソフトウェア情報の視覚化に対応できる汎用性を持っている。3次元ノードには形、位置、姿勢、縮尺、色、透明度が定義・変更できる。またリンクは色の定義・変更が可能である。

#### ・オブジェクト指向

VOGUEはC++で記述されており、上述した3次元属性、マウスやキーボードによる対話機能等は

すべて上位クラスで定義されている。したがって、新しいアプリケーションを作成する場合、VOGUEが提供するプリミティブクラスのサブクラスを定義すれば、ノード・リンクの3次元属性や対話機能はすべて継承して利用できる。これはアプリケーション作成の手間を軽減するとともに、あるアプリケーションで作成された機能の再利用を容易にした。また、プログラマはグラフィックスライブラリに関する詳細な知識なしでアプリケーションを作成できる。

- ・ 自動レイアウト機能

- 2次元木構造

- 指定したリンク・クラスにより2次元木構造

- 3次元木構造

- Cone Treeのような3次元木構造

- 3次元フラクタル木構造[Koike93b]

- 枝の長さがフラクタル関数に従って短くなるような3次元木構造

- 3次元ボックス構造[小池90]

- 指定したリンク・クラスによって入れ子状の箱にレイアウトする

- 焼き鈍し法による自動配置[Fairchild88]

- 焼き鈍し法を用いた自動配置

- ・ 図形数削減機能

視覚化する情報量が多いと、得られた視覚表現は非常に煩雑になる。VOGUEでは以下の図形数削減機能を持っている。

- 指定したクラスだけの表示・非表示を行う

- 着目ノードからの距離を指定して表示・非表示を行う

- Generalized Fisheye Views[Furnas86]を利用した表示・非表示

- Fractal View[小池91a]を利用した表示・非表示

- ・ 高速かつスムーズな視点変更

VOGUEが主張する3次元視覚化の重要な点は複数の視点の統合と視点間の自由な移動である。

したがって、利用者は必要に応じてどちらかに着目した位置、あるいは双方を見渡せる位置に視点を移動できなければならない。しかもこの視点移動は利用者の認知的負荷を低減させるため連続的でなくてはならない。

- ・高い移植性

VOGUEはグラフィックスライブラリに依存する部分を独立させているため移植性が高い。実際、VOGUEはSilicon Graphics社製IRIS上のGLだけでなく、Hewlette Packard社製HP9000上のStarbase、クボタコンピュータ社製TITAN上のDoreでの動作実績がある。

- ・仮想現実感 (Virtual Reality) システムとの整合性

VOGUEは実際のソフトウェア開発等での利用を前提としているため、利用者インタフェースとしてはマウス、キーボードと普通の計算機ディスプレイであり、特殊な入出力デバイスはあえて利用していない。しかし、手袋型入力デバイス、頭部搭載型ディスプレイ、液晶シャッター眼鏡、大型プロジェクタといったいわゆる仮想現実感装置の使用も行われており、容易に統合できることが確かめられている[名井91]。

### 3.1.4 応用例

ここではVOGUEを利用して作成されたアプリケーションについて説明する。

#### (1) 並列プロセスのパフォーマンス視覚化システム

計算機システムの大規模分散並列化により、今後、並列プログラミングの必要性は増すばかりである。McDowellらは、視覚化が有効なデバッグ手法であると指摘している[McDowell89]。

並列プログラムの視覚化によく利用されるのは、プロセス間関連図(図3.1-3)と縦軸にプロセス並び、横軸に時間をとったガントチャートである(図3.1-4)。しかし今後の大規模並列システムを考慮に入れた場合、2次元チャートは以下のような問題を持つ。

- ・縦軸1次元は大規模プロセス数に耐えられない
- ・縦軸1次元はプロセス間関係を表現できない
- ・プロセス間通信線の増加が図の視認性を低下させる

これに対し図3.1-5は3次元視覚化による両図の統合の概念図である。プロセスをxy平面に関連プロセスが近くなるよう配置し、z軸を時間軸にとる。xy平面に垂直な方向から見るとプロセス間関係を把握でき、z軸に垂直な方向から見ると時間関係が把握できる。本枠組は電力制御システム、自律分散マニピュレータ、並列言語Lindaにそれぞれ適用されている[小池91b,Koike93a]。

#### (2) オブジェクト指向言語のクラスライブラリブラウザ

オブジェクト指向言語におけるメソッド継承は、コード再利用という点から非常に便利な機能である。しかし、あるインスタンスへ送られたメッセージは必ずしもそのインスタンスが属するクラスで

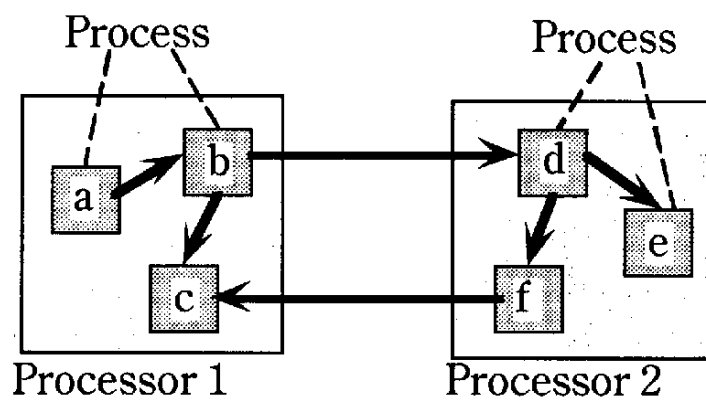


図3.1-3 プロセス関連図

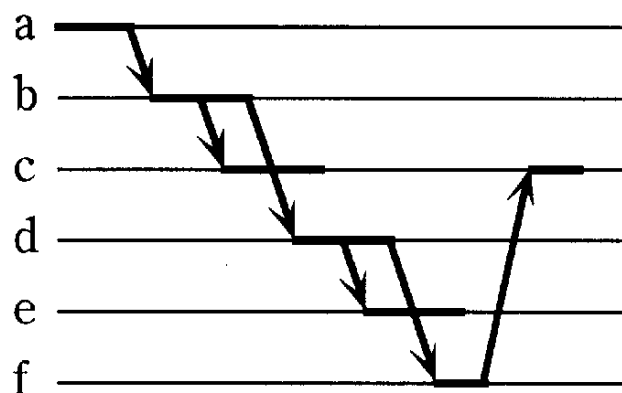


図3.1-4 時間関係図

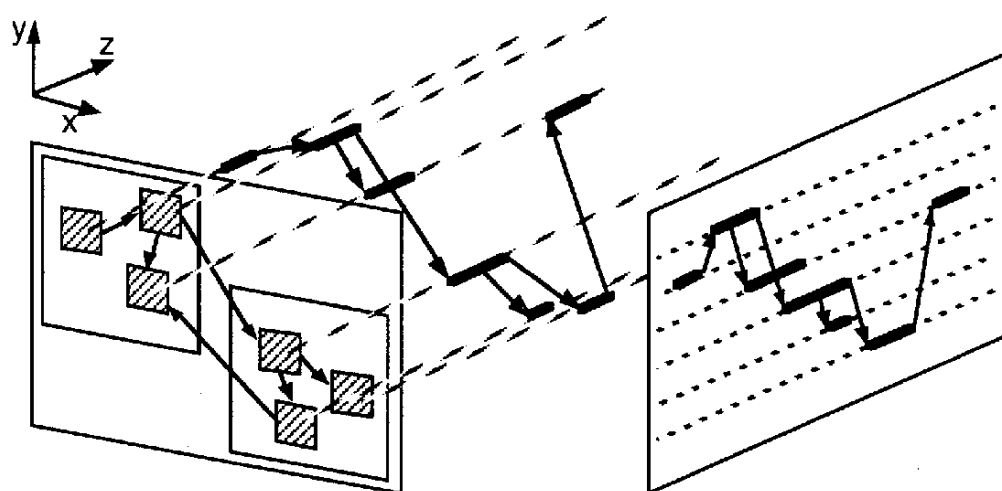


図3.1-5 3次元プロセスモニタの概念図

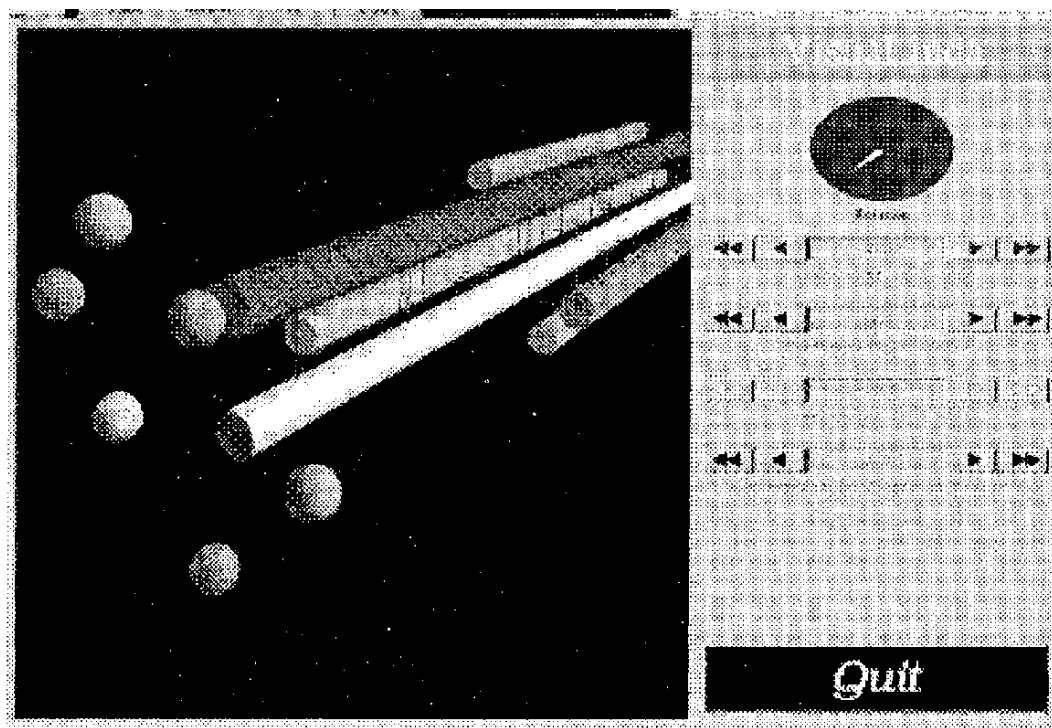


図3.1-6 VOGUEで実現した3次元プロセスモニタ

処理されるとは限らない。この場合、クラス階層を下からたどり最初に現れたメソッドが起動される。このメソッド継承のメカニズムは実行時にはシステムによって自動的に行われるため意識する必要はないが、デバッグ時にはプログラマが一つ一つ追跡する必要が生じる。

視覚的支援として一般に利用されるのはクラス階層図（図3.1-7）である。しかし、クラス階層図だけでは、あるクラスのインスタンスに対してメッセージを送った場合、実際に起動されるメソッドの位置は明らかでない。したがって、同一名を持つメソッドがどのクラスに存在するかという情報（図3.1-8）も必要である。

図3.1-9は3次元視覚化の概念図、図3.1-10はVOGUEで実現したクラスライブラリブラウザである。クラス階層図をxy平面に配置し、各メソッドはこの平面と垂直な方法、つまりz軸方向に、所属するクラスに対応するノードと同一x,y座標を持ち、かつ同じ名前を持つメソッドは同一のz座標を持つように配置する。xy平面に垂直な方向から見ればクラス階層が、z軸に垂直な方向から見れば継承関係が把握できる。被験者を使い、Smalltalk上のシステムブラウザ、Prographの2次元ブラウザとのメソッド探索実験を行った結果すべての課題に対して3Dブラウザが優位であった[小池91b]。

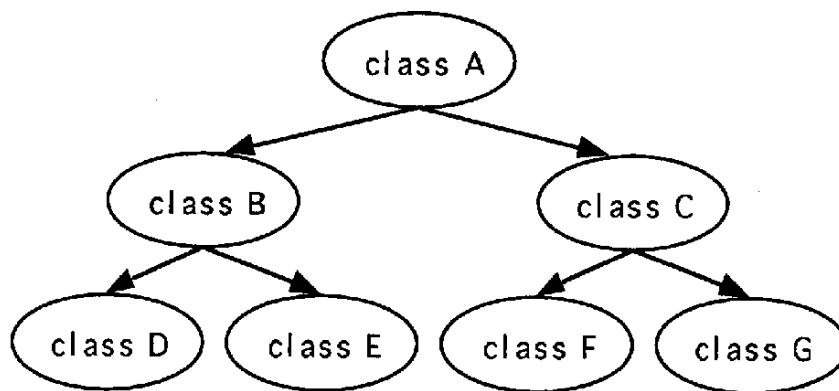


図3.1-7 クラス階層図

|         | Method 1 | Method 2 |
|---------|----------|----------|
| class A | ○        | ○        |
| class B | ○        |          |
| class C |          | ○        |
| class D |          | ○        |
| class E | ○        |          |
| class F | ○        |          |
| class G |          | ○        |

図3.1-8 メソッドリスト

### (3) 版管理・構成管理システム

実際のソフトウェア開発において版管理と構成管理は非常に重要である。UNIXではSCCS (Source Code Control System) やRCS (Revision Control System) が版管理システムとして、また Makeが構成管理システムとして一般的に利用されている。残念ながらこれらのツールは視覚的インタフェースを持たない。一方、CASE (Computer Aided Software Engineering) ツールはより強力な機能と同時に視覚的インタフェースをも持つ。版情報は2次元の木として表現され (図3.1-11)、構成情報は2次元の木 (図3.1-12) あるいは包含関係を表す領域図として表示される。しかしながら、版情報と構成情報は本来不可分である。ソフトウェア開発者にとって必要なのは、ソフトウェアシステムがどのモジュールから構成されるかだけでなく、各モジュールの版情報も同時に必要なのである。

これに対し、図3.1-13は3次元空間における版情報と構成情報の統合の概念図、図3.1-14はVOGUE で実現した3次元版・構成管理システムである。各ファイルの変更履歴と同時にシステムの構成情報が把握できる。

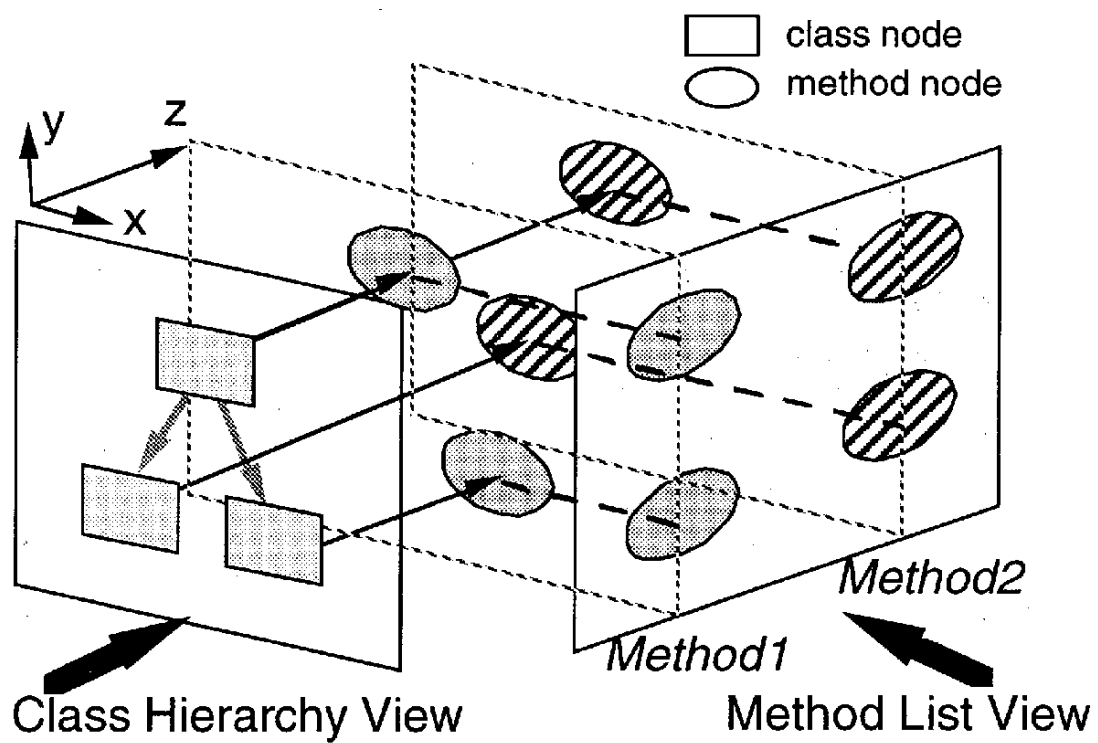


図3.1-9 クラスライブラリ3次元視覚化の概念図

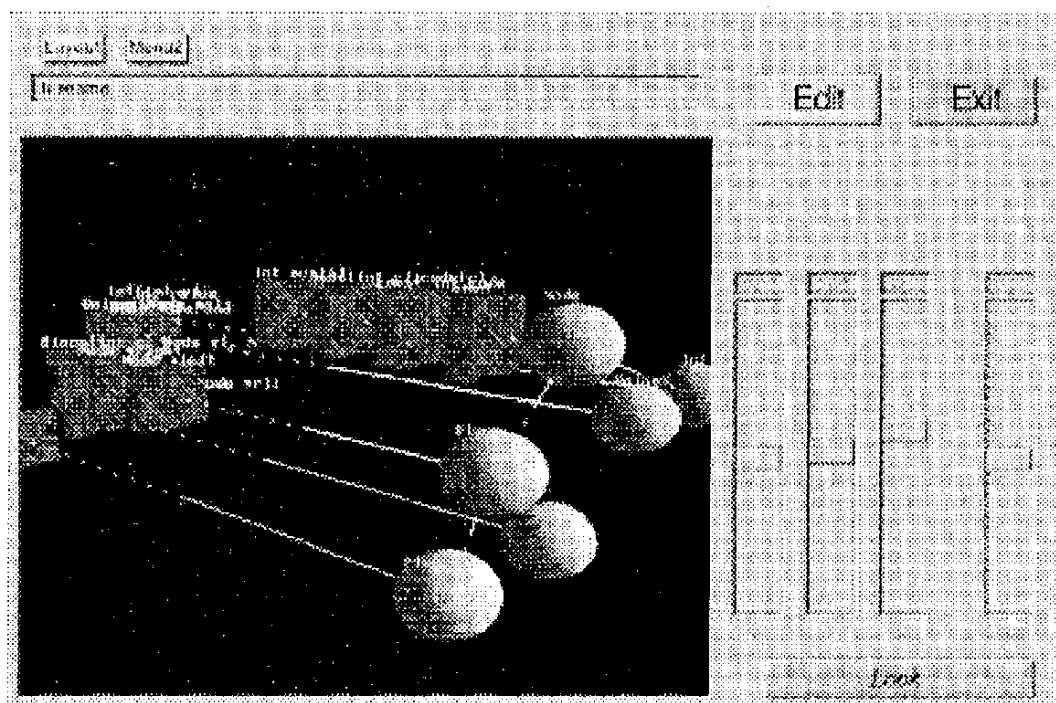


図3.1-10 VOGUEで実現した3次元クラスライブラリブラウザ

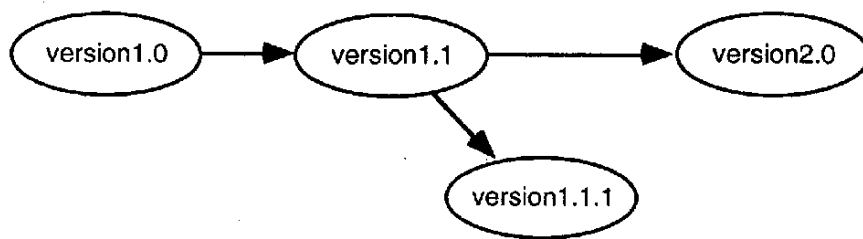


図3.1-11 版情報

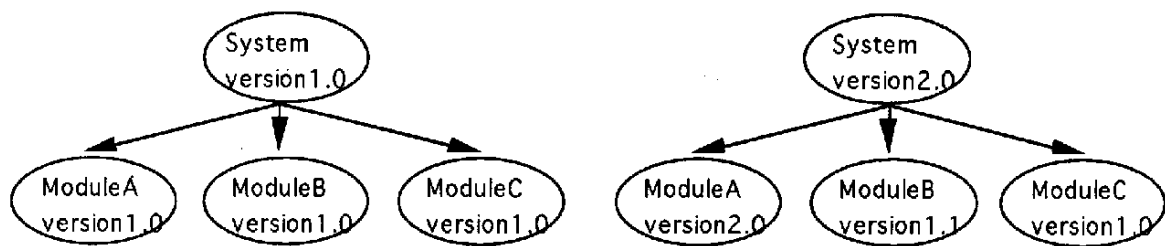


図3.1-12 構成情報

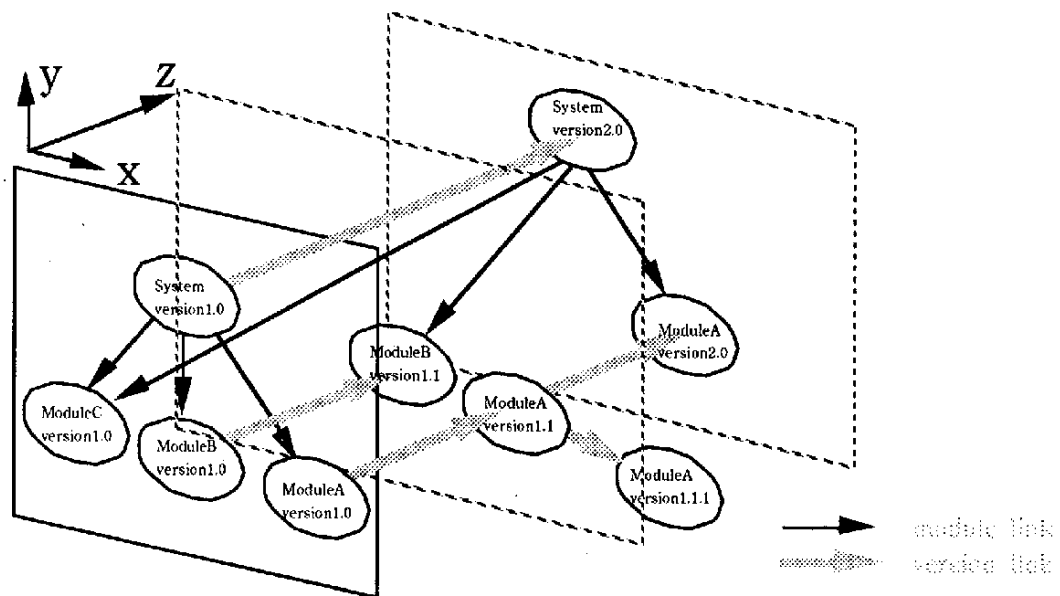


図3.1-13 版情報・構成情報の3次元視覚化の概念図

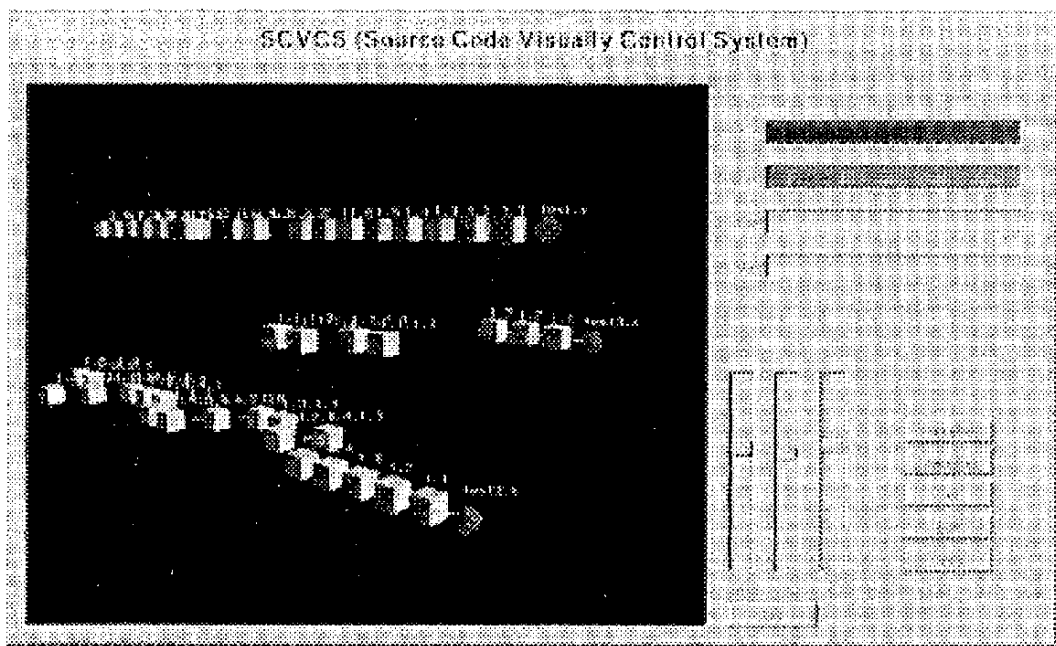


図3.1-14 VOGUEで実現した3次元版情報・構成情報視覚化システム

### 3.1.5 おわりに

ここではVOGUEを利用したソフトウェア開発への3次元視覚化の応用例を示した。従来の3次元視覚化システムにより明らかにされた、大規模データへの対応、リンク交差の回避、透視投影図法を用いた大局情報と局所情報の融合表示、アニメーション機能による利用者の認知的負荷の低減、といった3次元CG利用のメリットの他に、新たに次元の増加を利用した複数の2次元図の統合表示が可能であることを示した。

### 【参考文献】

- [Brown93] M.H. Brown and M. Najor: Algorithm animation using 3d interactive graphics, Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'93) (1993).
- [Card91] S.K. Card, G.G. Robertson, and J.D. Mackinlay: The Information Visualizer, an information workspace, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91), pp.181-188, ACM (1991).
- [Fairchild88] K.M. Fairchild, S.E. Poltrock, and G.W. Furnas: SemNet: Three-dimensional graphic representation of large knowledge bases, R. Guindon, editor, Cognitive Science And Its Applications For Human-Computer Interaction, pp.201-233, Lawrence Erlbaum Associates (1988).

- [Furnas86]G.W. Furnas: Generalized Fisheye Views, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'86), pp.16-23, ACM Press (1986).
- [小池90]小池, 広瀬, 石井: Software Visualization -3次元ソフトウェア視覚化ツールの開発, 計測自動制御学会第6回ヒューマンインタフェースシンポジウム論文集 (1990).
- [小池91a]小池, 石井: フラクタルの概念に基づく提示情報量制御手法, 情報処理学会論文誌, Vol.33, No.2 (1991).
- [小池91b]小池, 石井: 3次元ソフトウェア視覚化の枠組と実例による有効性の評価, 情報処理学会論文誌, Vol.33, No.6 (1991).
- [Koike93a]H.Koike: The Role of Another Spatial Dimension in Software Visualization, ACM Trans. on Information Systems, Vol.11, No.3 (1993).
- [Koike93b]H.Koike and H.Yoshihara: Fractal Approaches for Visualizing Huge Hierarchies, Proceedings of 1993 IEEE/CS Symposium on Visual Languages (VL'93) (1993).
- [名井91]名井, 甘利, 稲村, 小池, 葛岡, 広瀬, 石井, 林: 視覚表現を活用した大規模制御システムの一設計法, 電気学会論文誌, Vol.111-C, No.5 (1991).
- [McDowell89]C.E. McDowell and D.P. Helmbold: Debugging Concurrent Programs, ACM Computing Surveys, Vol.21, No.4 (1989).
- [Mackinlay91]J.D. Mackinlay, G.G. Robertson, and S.K. Card: The perspective wall: detail and context smoothly integrated, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91), pp.173-179, ACM Press (1991).
- [Robertson91]G.G. Robertson, J.D. Mackinlay, and S.K. Card: Cone Trees: Animated 3D visualizations of hierarchical information, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91), pp.189-194, ACM Press (1991).
- [Silicon92]Silicon Graphics, Inc: FSN: File System Navigator, ,online manual (1992).
- [Stasko93]J.T. Stasko and J.F.Wehrli: Three-dimensional computation visualization, Proceedings of 1993 IEEE/CS Symposium on Visual Languages (VL'93) (1993).

### 3.2 電力におけるビジュアルインタフェースの応用

仮想現実感（Virtual Reality）の電力への応用として、東京電力では次の二つの観点で研究を行っている。

- ① 実世界モデルへの応用（Real World）
- ② イメージ世界モデルへの応用（Image World）

仮想現実感とは、人間の五感と実世界やイメージ世界との新しいインタフェースを実現するものであり、特に視覚情報とのインタフェースという意味では以下のように捉えられる。

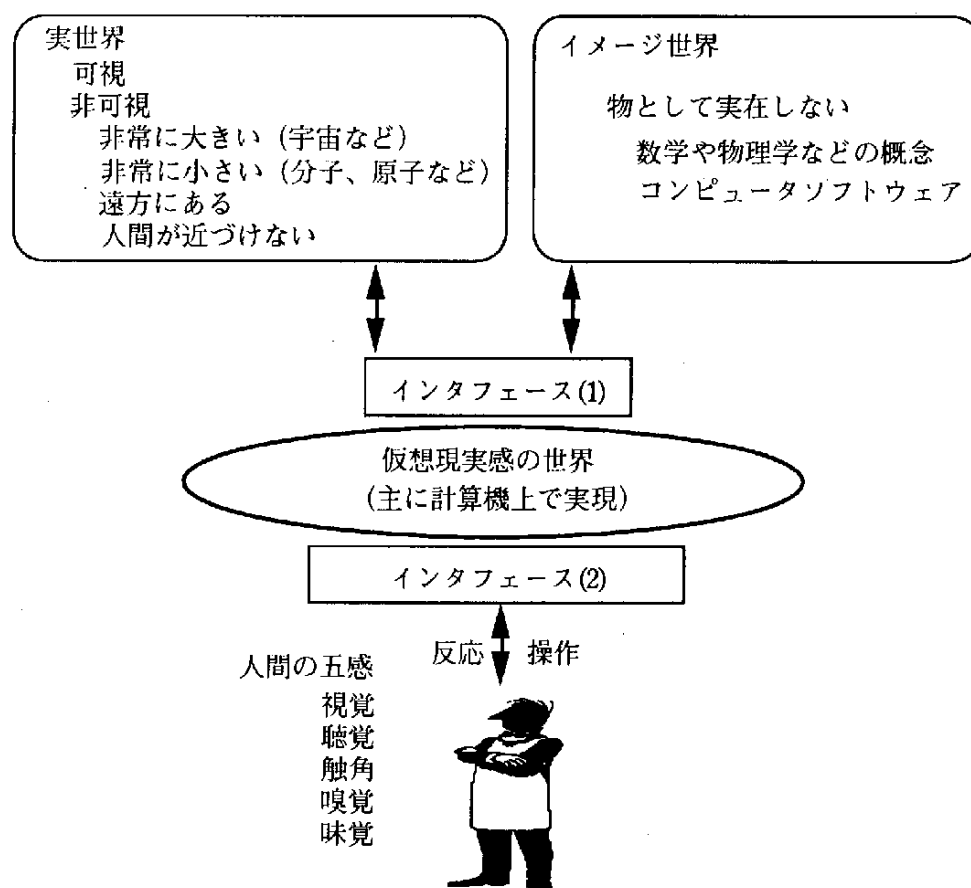


図3.2-1 視覚情報での仮想現実感の世界

図3.2-1において、インタフェース(1)はカメラやセンサなどを介して直接的に実世界やイメージ世界と計算機上の仮想世界とインタラクションを起こすものであり、インタフェース(2)はディスプレイなどの仮想世界と人間とのインタラクションに関わるものである。ここでは、インタフェース(2)について以下に述べる。

### 3.2.1 ハードウェア環境

現在使用している視覚情報のインタフェースは、人間への出力インタフェースとして、立体視対応の大画面プロジェクタ（70インチ）と液晶シャッター眼鏡を採用している。これは、実世界モデルへの応用には3次元立体視が必要不可欠であり、またイメージ世界モデルへの応用についても、大規模で複雑な情報を様々な次元要素で見るためであり、さらに同時に複数の人間で見るという意味でも、大画面立体視方式を採用している。

次に、人間から仮想世界への入力インタフェースとしては、Data Gloveと3次元位置センサであるPolhemus 3 Spaceを使用している。

また、実画像やコンピュータグラフィックス（以下CGと略す）処理用のワークステーションとして、Silicon Graphics社のONYX Reality Engine 2を使用しており、これらのシステム構成図を図3.2.2に示す。

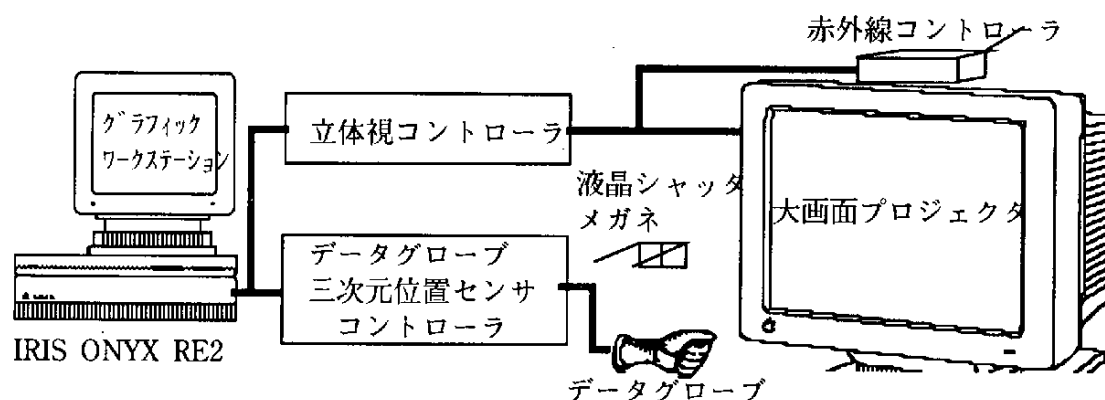


図3.2.2 システム構成図

### 3.2.2 実世界モデルへの応用

無人設備（無人変電所等）や人間が近づけない設備（原子炉内部等）に対する遠隔監視・制御あるいは、電力機器設備（制御室や燃料電池等）の設計への応用について述べる。

遠隔監視・制御のための現状のビジュアルインタフェースとしては、遠隔地にある機器をシンボル化して、開閉機器の入/切状態や充電/停電状態を色情報で表示している。このシンボルに対して、ライトペンやマウスカーソルを用いて遠隔操作を行っている。このシンボル表示の例を図3.2.3（注：本節末のカラー写真）に示す。

図3.2.3のようなシンボル表現とそれに対する操作では、実際の機器操作としての感覚が得られず、また遠隔地の機器が正常に動作したか否かは、ディスプレイ上のシンボルの色変化でしか捉えられな

いたため、運転員としては不安が残る場合がある。そこで、遠隔現地の実映像を取り入れたいという要求があり、制御室のどこにどのように取り込むかの検討が必要になる。現状では、実物大のモックアップを作成し検討するという手段を用いているが、柔軟性や費用の面で問題点があり、仮想現実感による早期の実現が期待されている。図3.24（注：本節末のカラー写真）に現地映像取り込みの例を示す。

また、原子力発電所の炉心などの人が近づけない設備の監視・制御についても、現状は水位や燃料棒の状態などをセンサ情報に基づいて数値化、シンボル化して表示している。これらの監視・制御についても同様に、実映像やCGを駆使した新しいビジュアルインタフェースにより、より高度で人に分かりやすいシステムが構築できるものと思われる。

これらのビジュアルインタフェースは、危険箇所の補修工事などの事前説明や事前の作業シミュレーションにおいても、現状の図面中心のインタフェースに代わって、3次元立体視やリアルタイムなインタラクション機能により、より安全で的確なものになる。

以下に、本システムの実世界モデルにおける機能について述べる。

#### ① 視点移動

Data Gloveのジェスチャにより、前進、後退、上下移動、左右回転の動作が行える。3次元立体視表示においては必須の機能である。

#### ② 物体移動

Data Gloveにより仮想空間内のすべてのCG物体をつかんで移動することができる。この時、物体をつかんだことを、Data Gloveのハンドカーソルの色変化と音で表現している。また、空間内で物体を放すと自由落下する自律（autonomy）の機能も持っている。

#### ③ 実映像のCGへの融合

CGの一部に実映像を融合して、実映像も立体視できる。また、この機能は、CGデータを削減してよりリアリティを高めるテクスチャマッピング機能としても使われている。

#### ④ 配色変更

物体ごとのR、G、Bの組合せによる配色機能である。

#### ⑤ 光源の設定

光源の位置や種類（点、並行）が設定できる。

①～③の機能については、すべてData Gloveによるインタフェースであるが、④⑤についてはマウスインタフェースであり、インタフェースの混在という意味で使いづらい面もあるが、これに代わるよいインタフェースが現状では難しい。

また、設計シミュレーションという観点では、物体の移動だけでは不十分で、CADシステムとの連係によるリアルタイムでの物体変更機能が必要であり、この時も、数値情報などは可能な限り削除し、

Data Gloveだけのインタフェースを指向している。

### 3.2.3 実世界モデル応用での課題と今後の計画

ビジュアルインタフェースという観点で現状のシステムを考察すると、70インチの大画面プロジェクタは、臨場感を得るという意味では視野角が狭く感じられる。また、左右の横面、上下面の情報がまったくないため、これらを補うなんらかの方法（例えば、ヘッドマウントディスプレイや多面プロジェクションなど）が必要である。

次に、仮想空間への入力インタフェースという観点では、Data Gloveや3 Spaceに代わって、Cyber GloveやFastrakなどの新しいデバイスが開発されており、精度や信頼度の点で少しずつ改善されており、実用化へ一歩近づいた感がある。現在は、両手を用いた自然なインタフェースについても研究を行っている。

ところで、実世界モデルへの適用において、3次元立体視や手操作によるアクションは物体移動では自然なインタフェースであるが、ジェスチャによる視点移動は、まだまだ改善の余地が残るインタフェースである。

最後に、グラフィックワークステーションや視覚表現技術についてであるが、現状のワークステーションで、約4000ポリゴンのCGデータと約4000枚のテクスチャデータを用いたアプリケーションにおいて、立体視時に10～12画面/秒の実行性能が得られているが、今後のより複雑なデータ表現に対する性能向上が望まれる。また、3次元CGと3次元実映像については、CGの中への映像の取り込みを行っているが、今後は、3次元の実映像に3次元CGを重ねて、実時間処理を行う機能が必要であるが、この点に関しては、まだまだ解決すべき問題が山積している。

実世界モデルへの応用は、最終的には遠隔地にロボットなどの情報収集と遠隔操作を同時に行える装置を設置し、テレイグジスタンス機能として発展させる予定であり、この面での研究も今後実施していく予定である。

### 3.2.4 イメージ世界モデルへの応用

#### (1) 概略

仮想現実感の応用分野として、建築物や電力機器等の実在する物の設計等への応用とともに、計算機内のソフトウェアの構造や実行状況などのイメージの世界における設計や解析への応用も研究している。

ソフトウェアビジュアライゼーションは、機能的に大別して解析と設計に分類できる。解析では、作成されたソフトウェアのソースコードや計算機上での実行トレースデータを元に、タスクやモジュールの実行状態やファイルアクセスの時間的遷移が大画面プロジェクタに3次元立体表示される。こ

---

## 3. ビジュアルインタフェースの事例

の立体表示の中をデータグローブにより自由に移動でき、タスク、モジュールやファイルをつかんで空間的に自由に移動できる。これにより、実行中の Live-Lockや Dead-Lock等の障害や、多量のファイルアクセス等による性能障害の検出が視覚的に行える。

設計では、基本的にはモジュール内部のセンテンスや制御構造、モジュール間やタスク間の制御構造やファイルアクセスを図形表現で作成し、ソースコードへの展開を行うことを前提としている。

設計におけるもう一つの重要な機能としてシミュレーション機能があり、タスクやモジュール間の呼び出し関係やファイルアクセスのタイミングを、視覚的にシミュレーションすることが可能である。電力系統制御システムのようなリアルタイム制御プログラムでは、このような時間に関するシミュレーション機能は非常に重要である。

設計において現在実現されている機能は、簡単な呼び出し関係のソースコードへの展開機能とシミュレーション機能である。

今後の計画としては、設計機能の完成と、現在行っている制御構造中心の視覚化にデータ構造の視覚化を取り入れる。これにより、ソフトウェア開発工程の機能設計からテスト工程への適用に加えて、要求仕様定義段階での今までのようなテキストベースの要求定義から、映像等の情報を用いた制御対象物との対応を陽に意識した要求定義の可能性についても考察を行う。

ソフトウェアビジュアライゼーションの対象として大別すると、サブシステム、タスクモジュール、フロー解析の単位である基本ブロック、実行文等の実行に関する要素、制御対象である機器定義等のデータに関する要素に分類できる。

実行に関する要素の表示においては、図3.2-5（注：本節末のカラー写真）に示すような平面や空間での広がりを利用した階層性の表示とCGの半透明表示を利用した階層性の表示がある。また、制御フローの時間的な遷移を表す場合などは、3次元空間の一つの軸に時間を割り当てる。

現在は、タスク、モジュールについてのみ表示を行っているが、将来的には、その他の階層についても表示する予定である。

データに関する表示は、現在ファイル単位に行っており、階層性などの構造に関する表示は行っていないが、将来的には構造表示やプログラムデータだけではなく、ターゲットである制御対象機器そのものの表示も行う予定である。

さらに、3次元空間内の図形データを扱うようなインタフェースによりソフトウェアの解析、設計を行うことを前提にしているため、3次元空間図形に対する追加・削除や COPY/CUT and Paste のような空間的な操作性についても重要と考えている。

## (2) 仮想現実感としての位置付け

仮想現実感の適用分野として、多くのものは建築物や機器などのいわゆる実在する物の世界（Real-

World) への応用である。これとは対象的な世界として、物理学や数学などのいわゆる概念またはイメージの世界 (Image-World) への応用も考えられる。

ところで、このような観点でソフトウェアビジュアライゼーションを考えると、ソフトウェアは、ソースコードやロードモジュールとしてはメモリ上に実在する世界であるが、タスクやモジュールとして時間の経過とともに実行されたものは、結果として様々な形で残るが、そのものは実在しない物の世界であると考えられる。

また、データに着目すると、プログラムデータも制御対象機器などのターゲットデータも実在する世界である。つまり、ソフトウェアビジュアライゼーションは、両者の特徴を有するものと考えられる。

以上の関係を、無人変電所の遠隔監視制御システムを例に図示すると、図3.2-6のようになる。



現実の変電所の例

### (3) ソフトウェアビジュアライゼーションの現状

ソフトウェアビジュアライゼーションの機能を大別すると、視覚化解析 (Visual Analysis) と視覚化設計 (Visual Design) に分類できる。

視覚化解析は、OSの実行管理機能からのトレースデータとソースコード解析により構造データを作成し表示する。

表示例を図3.2-7、図3.2-8、図3.2-9 (注：図3.2-7、8、9とも本節末のカラー写真) に示すが、図3.2-7

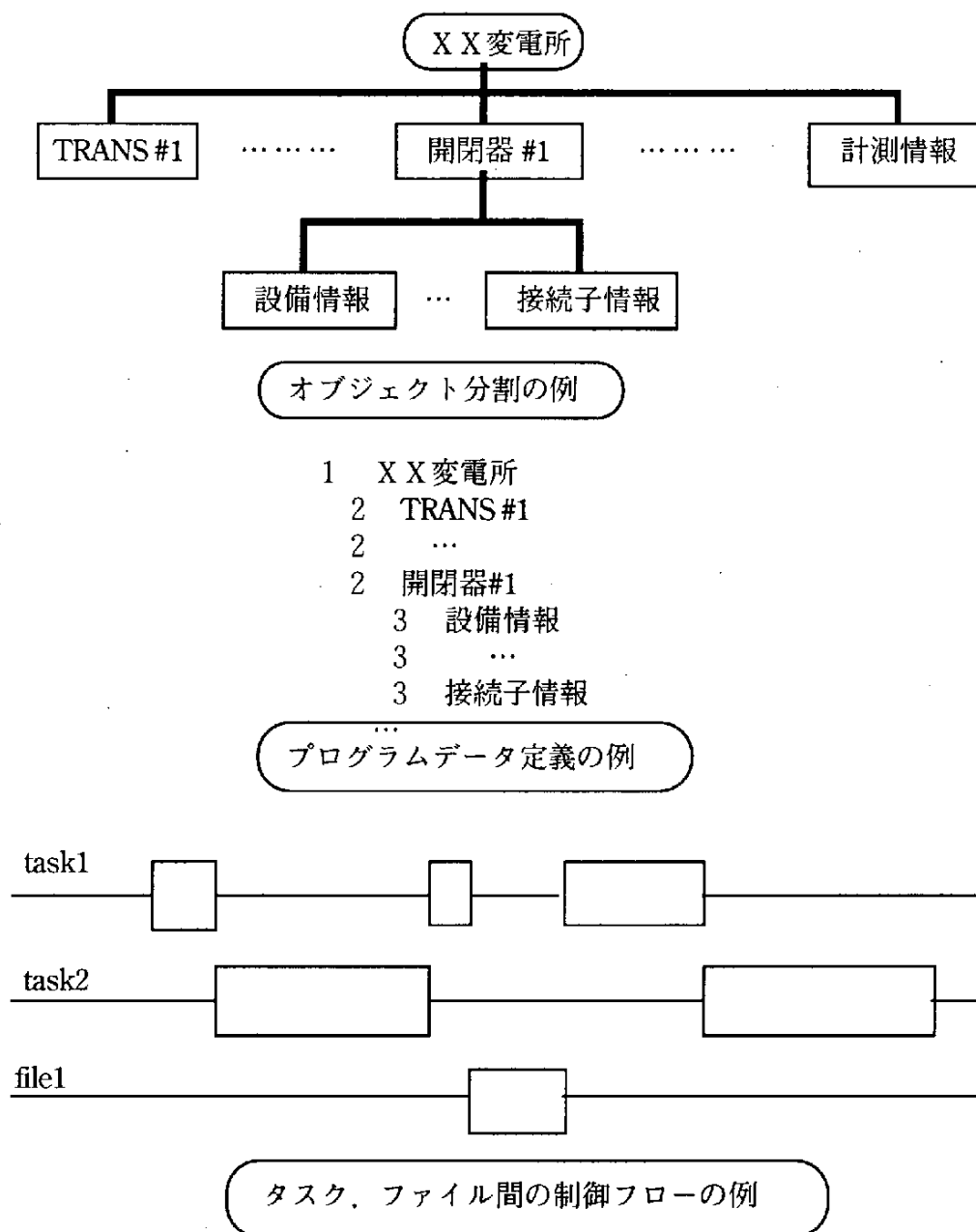


図3.2-6 REAL WORLD と IMAGE WORLD

は、タスク間とタスク・ファイル間の実行状態を構造を表す平面から見た透視図であり、図3.2-8は、時間軸に沿って表示したものである。また図3.2-9は、モジュールの階層関係や静的な呼び出し関係と共通モジュールとの静的な呼び出し関係を同時に表示したもので、さらにこれらの実行時間遷移も時間軸に沿って表示される。

現在は、視点移動や表示されたタスク、モジュール、ファイルの空間内の移動をデータグローブで

行い、人間が解釈し、処理の錯綜状況や Dead-Lock、Live-Lockなどの障害の検出を行っている。しかし、将来的にはこれらの解析を知識処理などを利用して知的に支援する計画である。

視覚化設計は、図形上の簡単な呼び出し関係図をデータグローブで作成して、それをCプログラムの関数呼び出し文にソースコード展開する機能、処理タイミングなどのシミュレーション機能が実現されている。図3.2-10、図3.2-11（注：図3.2-10、11とも本節末のカラー写真）に例を示すが、図3.2-10は、タイミングシミュレーションによるチューニングの例、図3.2-11は二つのタスクと二つのファイル間のアクセス競合による Dead-Lockのシミュレーションの例である。

ところで、視覚化解析機能と視覚化設計機能の使い方として、新規にプログラムを作成する場合と既存プログラムに対する機能追加・変更で図3.2-12に示すような作業の流れを想定している。

仮想現実感操作環境に対する考え方として、「人間に装着する装置をできる限り減らし、かつ一度に多くの人が仮想空間を共有できる」という前提で構築した。今後、新しい入出力デバイスに関してもこの前提は変えないが、距離的に離れた地点での共同作業は必要であるので、ネットワークを介した仮想現実感環境についても構築する予定である。

また、ソフトウェアビジュアライゼーションのための実行トレースの取得やソースコード解析については、グラフィックワークステーションとネットワーク結合された汎用ワークステーション上で行っている。

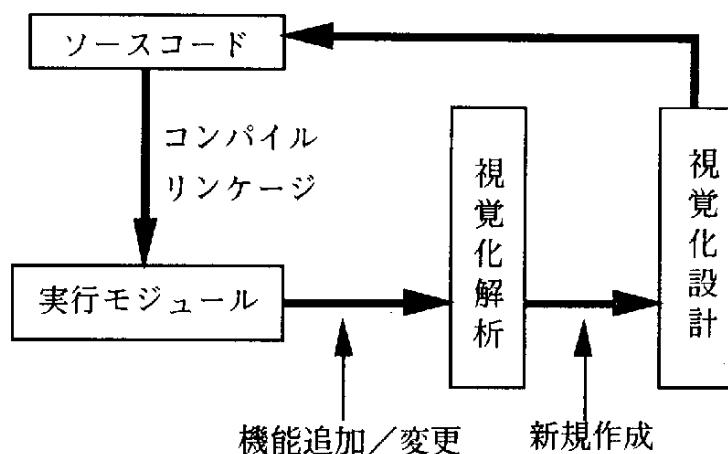


図3.2-12 作業の流れ

#### (4) 仮想現実感環境としての評価

##### ① 入出力デバイス

入力デバイスは現在データグローブを使用しているが、機器設備のレイアウトや設計においては、手でつかむ感覚として有効であるが、ソフトウェアビジュアライゼーションのような仮

仮想空間のオブジェクトを編集する場合は、位置精度の高い3次元ポインティングデバイスのような入力装置の方が操作しやすい。

また、操作空間では、視点移動や仮想オブジェクトの編集などの多くのジェスチャをデータグローブに割り当てているという理由もあるが、データグローブの耐久性にも難点がある。もう少し耐久性に優れた方式が望まれる。

次に、視覚系の出力デバイスとして70インチプロジェクタを使用しているが、これは多くの人が同時に仮想空間を共有するためにコストパフォーマンスのよい現状の方式として採用した。HMDの方が、より仮想空間への没入感が得られるが、現状のHMDの重量や体勢感覚との連動の不備、精細度、それにコストパフォーマンスを考えるとまだまだ改良の余地がある。

また、立体視方式として液晶シャッター眼鏡を用いているが、視差の個人差への対応や初心者への疲労感などは、方式自体改良の余地はあるが、簡易的な立体視方式としては現状では有効な手段である。

## ② その他のデバイス

3次元の位置センサとしてデータグローブ連動の POLHEMUS センサを使用しているが、微細な3次元空間の位置決めや有効範囲の狭さに問題点はあるが、耐久性などは特に問題はない。対象物が建築物や機器設備などであっても、最終的には微細な位置決めが必要となる場合が多いので、このような仕様を満たす3次元位置センサの開発が必要である。

## ③ グラフィックワークステーション

IRIS 4D/210 VGX から IRIS ONYX RE 2に最近変更したが、テクスチャを多用するアプリケーションでは性能は向上したが、ソフトウェアビジュアライゼーションなどのポリゴンデータのみのものでは並列化などの高速化のチューニングを考える必要がある。

## (5) 今後の計画

ソフトウェアビジュアライゼーションに関する今後の計画として、仮想現実感環境、視覚化解析、視覚化設計、要求仕様定義への適用について述べる。

まず、仮想現実感環境としては、3次元空間位置精度のよい位置センサと組合せた3次元入力デバイスの開発とネットワークを介した仮想空間の共有による共同作業環境の整備が必要である。

視覚化解析においては、現状の人間による機能障害箇所や性能障害箇所の判別に代わる計算機による支援を行う必要がある。障害箇所の検出方法、表現方法、あるいは性能障害の判定基準の設定等、視覚的に表現したソフトウェアにさらに情報を追加する必要がある。

また、現状の制御構造の視覚化に加えて、データ構造についても視覚化する必要がある。これにより、ソフトウェアのイメージ世界と機器設備等の実世界との自由な行き来が実現できる。

視覚化設計においても同様に、制御対象機器とのダイレクトなマッピングによるソフトウェアの設計が行える。

最後に、今まではソフトウェア開発工程で言えば、機能設計、詳細設計、プログラム作成、テストの工程を中心に考えていたが、実世界とのマッピングが実現すると、要求仕様定義段階への適用も考えられる。

ソフトウェアの発注者と作成者が同じ仮想環境を共有して仕様定義を行うわけであり、これを視覚化要求仕様定義 (Visual Specification Definition) と呼んでいる。今後、視覚化要求定義に必要な機能をまとめ、ソフトウェア開発におけるすべての工程を通した新しい開発環境を構築する。

### 【参考文献】

- [Amari93]Amari, H. et al.: A VIRTUAL REALITY APPLICATION FOR SOFTWAREVISUALIZATION, IEEE VRAIS '93 (1993).
- [甘利92]甘利治雄他: 仮想現実感を利用したソフトウェア設計方式の考察, 計測自動制御学会 8th Symposium on Human Interface (1992).
- [甘利93]甘利治雄他: 仮想現実感を利用したソフトウェア視覚化方式について, 信学技報, SS92-23, KBSE92-44 (1993-01).
- [南雲92]南雲俊喜他: 仮想現実感を利用した遠隔監視・制御方式の検討, 計測自動制御学会 8th Symposium on Human Interface (1992).
- [Shu88]Shu, N. C.: Visual Programming, Van Nostrand Reinhold Company, New York (1988).

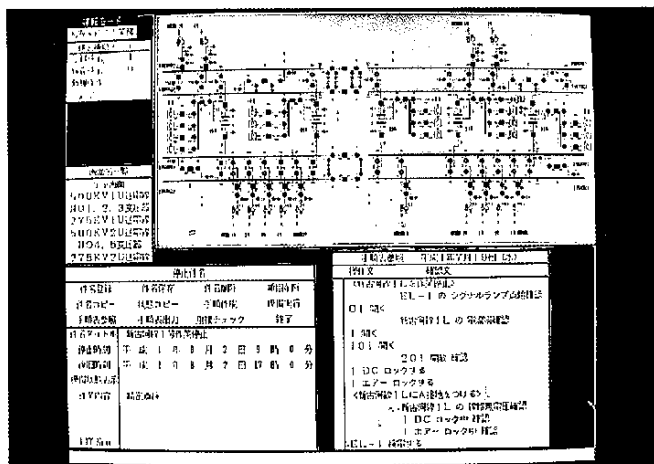


図3.2-3 シンボル表示の例



図3.2-4 現地映像表示のシミュレーション例

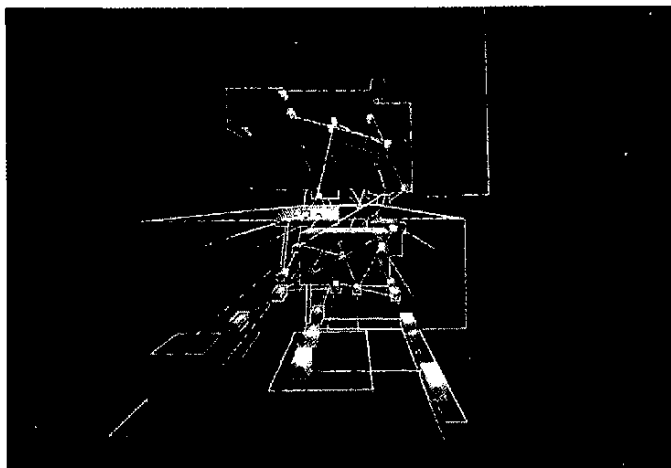


図3.2-5 階層表示の例

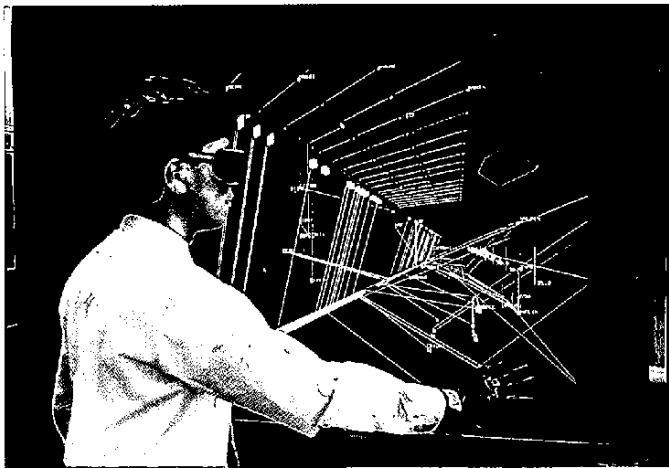


図3.2-7 タスク、ファイル実行の構造表示



図3.2-8 タスク、ファイル実行時間表示

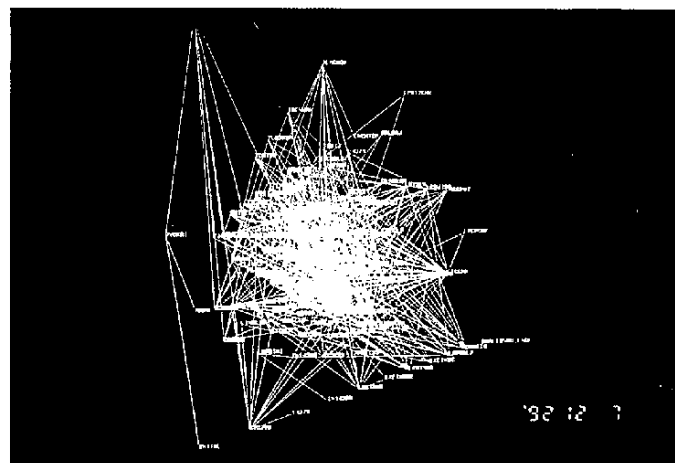


図3.2-9 モジュール静的構造表示

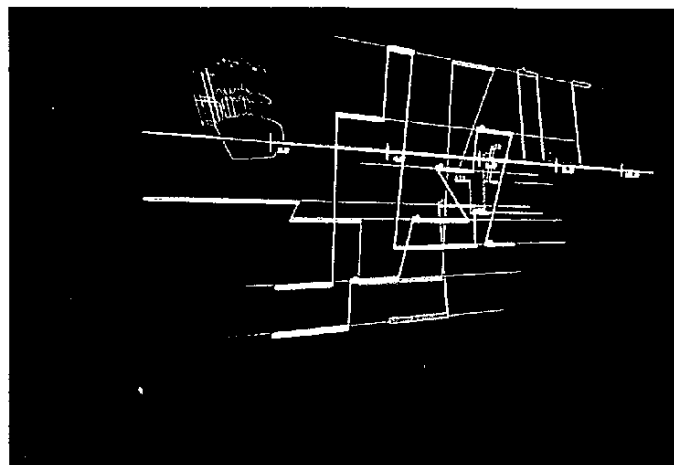


図3.2-10 タイミングシミュレーション

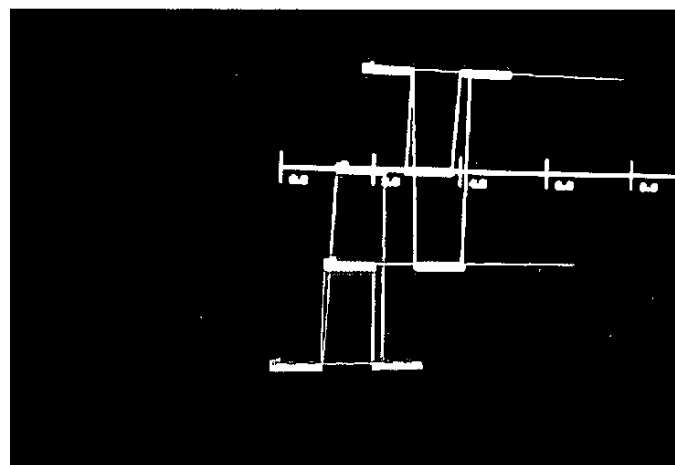


図3.2-11 Dead-Lock シミュレーション

### 3.3 ビデオ映像部品を利用したリアリティユーザインタフェースの試み

#### 3.3.1 はじめに

コンピュータによるメディア処理技術が次第に向上してきた。これにより、様々なメディアを利用した新しいユーザインタフェースが実現されている。例えば、メッセージを表現する際に以前はテキストだけで表示していたような場面で、現在はグラフィックスやアニメーションで表示したり、合成音声で説明したりできる。一般に、メディア処理技術の向上がすぐにユーザインタフェースの向上に波及するわけではなく、そこには何か新しいアイデアを必要とする。デスクトップメタファというアイデアに基づき、ウィンドウ、アイコンなどを直接操作するGUI (Graphical User Interface) が開発されてユーザインタフェース向上に大きく貢献したことはよく知られている。

さて、最近注目されているメディア処理技術は、ビデオ映像を対象とするものである。初期はビデオテープやレーザディスクに蓄積した映像をコンピュータ制御可能な再生装置から再生してディスプレイに表示するだけであったが、現在は映像をデジタル化、圧縮してハードディスクに蓄積し、高速かつランダムに検索・再生できるようになってきた。しかし、これをユーザインタフェースで利用しようとするとあまり柔軟性がなく、通常のVTRなどでの再生と同様の画面を表示するだけにとどまっている。他のメディアであるグラフィックスならば、はるかに柔軟な利用法が既に開発されている。例えばボタン、スクロールバー、メニューなどの部品に体系化され、それらを組み合わせて画面を構成できる。

以上の立場に基づき、ビデオ映像をユーザとコンピュータとのインタラクションのための部品として利用する試みを行ってきた。以下では、ビデオ映像を利用して作成した画面構成のための部品や、それらの組合せで作成したユーザインタフェース (リアリティユーザインタフェース) について述べる。

#### 3.3.2 従来のユーザインタフェースにおけるビデオ映像

従来、ユーザインタフェースでビデオ映像を利用していたのは以下のような場合である。

##### ・映像の検索・表示システム

ユーザが蓄積された複数の映像の中から自分で見たいものを選択して、その結果をディスプレイに表示する、というものである。映像は、映画の内容であったり、電子マニュアルのように何かの作業手順をビデオで示すものであった。この手法の対話性を向上したものとしてLippmanのMovie-Mapsがある[Lippman80]。これは、ある街の中のあらゆる道を通った時の景色をビデオカメラで撮影してレーザディスクに蓄積しておき、ユーザがコンピュータを利用して選択した道に応じて適切な映像を検索・表示するものである。ユーザはちょうど街の中を自由にドライブしている感覚で映像を見ることができる。

### ・ 在席会議システム

在席会議システムなどで、ある地点の人の様子を撮影しているビデオカメラの映像を、別の地点の人のディスプレイに表示する[渡部91]。これにより、会議参加者は他の人の様子を容易に把握できる。この場合はオンライン映像が利用される。

以上に共通するのは、ビデオ映像がユーザに情報を表示するためだけに利用されていることである。しかし、ビデオ映像を例えばGUIにおけるボタンなどと同様、インタラクションの部品として利用するためには、映像自身がユーザの入力を受理して応答する必要がある。その例として、山足等による「被写体の構造を持ったカメラ」がある[山足91]。山足等は、遠隔操作可能なビデオカメラで撮影している被写体の3次元モデルをあらかじめコンピュータ内に持つことで、ユーザが直接制御可能なビデオ映像を実現している。例えば機械装置をビデオカメラで撮影している際に、ユーザが画面に映ったビデオ映像の特定の部分を指示すると、指示した画面上の位置をコンピュータが内部の3次元モデルに対応づけ、カメラを機械的に制御してその部分にズームインする。この場合、ビデオ映像は情報表示を行なうと同時に、ユーザからの入力を受理する機能も持っている。山足等の手法は、オンラインでビデオ撮影している対象物を制御する場合には非常に適している。しかし、目的としているのは、汎用のユーザインタフェースを構築する部品としてビデオ映像を利用することである。画面構成の部品を絶えずオンライン撮影していることはできないので、蓄積型ビデオ映像を利用しなければならない。また、オンラインの場合には、ユーザの操作に対応して必要な映像を直ちに撮影することができるが、蓄積型ビデオ映像を利用する場合には、あらかじめ撮影しておいた映像しか利用できない。これらの問題に対処するために、ユーザインタフェース部品として利用するビデオ映像（これをビデオ映像部品と呼ぶ）に付加するインタラクション機能を分類し、それぞれの機能に対応してインタラクション方法を規定した。

### 3.3.3 インタラクション機能を持つビデオ映像

ここで述べるビデオ映像部品とは、ディスプレイ画面の構成要素として表示され、ユーザの入力を受理する蓄積型ビデオ映像である。映像はハードディスクなど十分高速にランダムアクセス可能な媒体に蓄積されていることを前提としている。GUI部品であるボタン、スクロールバー、ウィンドウなどと同様、ユーザインタフェース設計に利用することを目的としている。ビデオ映像はグラフィックスと異なり連続した多数の画面（1枚1枚の画面を以下ではフレームと呼ぶ）の集合からなるので、このような特徴を活かした部品を作成できる。一つのビデオ部品を表示するために、一つ以上のカット（ビデオカメラのスタートボタンを押してからストップするまでの一区切り）をあらかじめ用意する。例えば、ある一つの物体を被写体とし、その回転や変形の様子を撮影したカットを用意しておく。ユーザ入力に対応して最も適切なカットを選択して表示すれば、ユーザ操作に応じて物体が回転したり

変形したりするので、ユーザは被写体を直接制御しているように見える。このためには、表現の目的に応じてあらかじめ撮影したカットに関連付けを行い、ユーザ操作に対応するカットの制御方法を設定しておく必要がある。ユーザ操作とカットの制御方法との関係によって、次に示す3種類のビデオ映像部品を用意した。なおユーザからの入力装置としてはマウスを仮定している。

① 操作依存ビデオ映像部品

表示フレーム（デジタイズされたビデオの1画面）がユーザ操作に直接依存するビデオ

② 位置依存ビデオ映像部品

表示フレームが、部品の画面上での位置に直接依存するビデオ

③ 時間依存ビデオ映像部品

表示フレームが時間に直接依存するビデオ

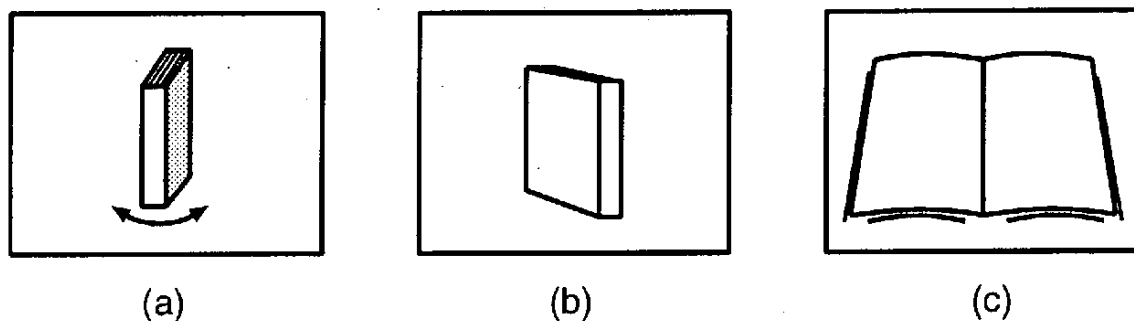


図3.3-1 操作依存ビデオ映像部品の例(1)

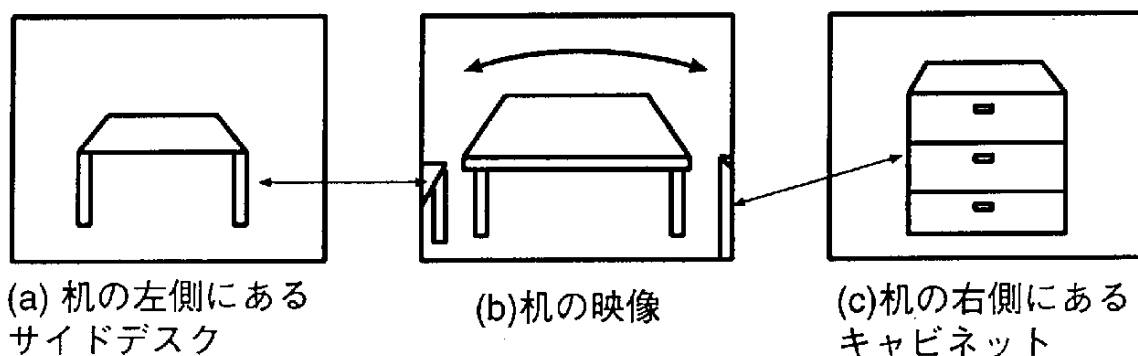


図3.3-2 操作依存ビデオ映像部品の例(2)

①の操作依存ビデオ映像部品は主として、ユーザの操作対象（例えばファイルを示す1冊の本）や、ユーザの作業環境である情景（例えばデスクトップのまわり）を表現する時に利用する。例えば、1冊の本を操作依存ビデオ映像部品で表現すると仮定する。利用する映像として、本を右に回転するシーンを撮影したカット、左に回転するシーンを撮影したカット、本を開くシーンを撮影したカットを用意しておく。そして、次のような表示制御を行う。

- ・図3.3-1(a)のような本の映像が表示されている時に、ユーザが画面の右半分でマウスボタンをプッシュすると、プッシュしている間、本が右へ回転する映像を再生して図3.3-1(b)のようになる。
- ・画面の左半分でマウスボタンをプッシュすると、プッシュしている間、本は上記と逆方向に回転する。
- ・本の中央付近でマウスボタンをクリックすると図3.3-1(c)のように本が開く。
- ・上記の本の回転速度は、マウスボタンをプッシュする位置と、画面の中心点との距離に比例する。つまり中心から離れた点をプッシュすれば本は速く回転し、近い点でプッシュすればゆっくり回転する。
- ・開いた本の中央付近をクリックすると、本が開く際の映像を逆回しして本が閉じる。

別の例として、図3.3-2(b)のような机のまわりの情景において、ユーザが画面の右半分でマウスボタンをプッシュするとシーンが回転してさらに右側の様子を見ることができ、画面の左半分でプッシュすると左側の風景を見ることができるようになる、というものがある。

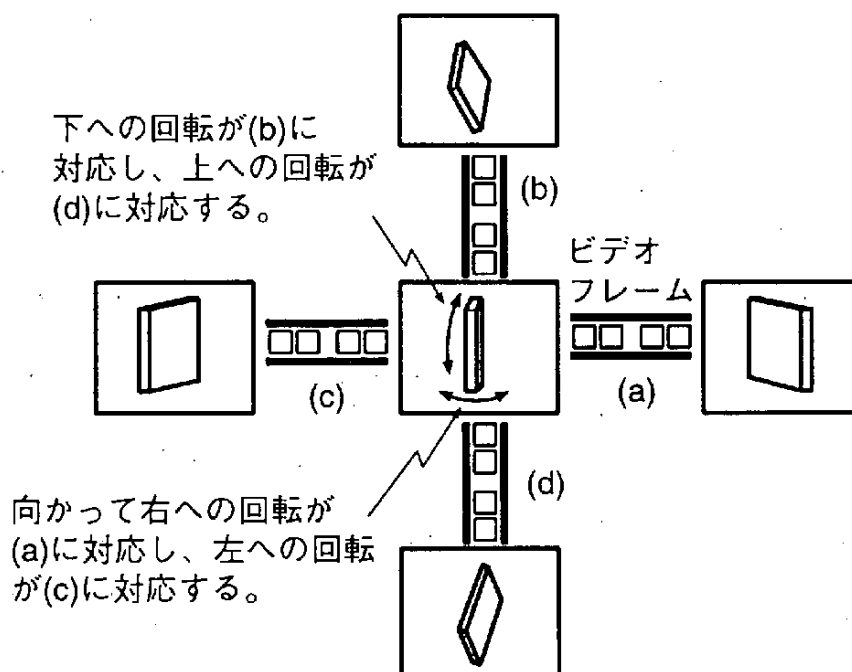
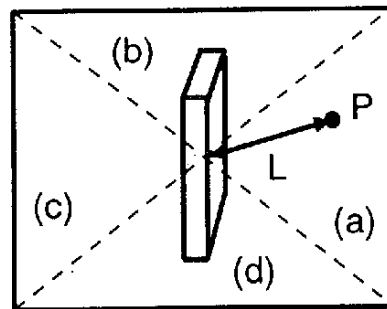


図3.3-3 ビデオ映像のネットワーク



P: ユーザの指示したポイント

L: 中心点からのPの距離

図3.3-4 表示するビデオフレームの決定法

上記のような操作依存ビデオ映像部品を次のようにモデル化した。

- ・図3.3-3に示すように複数のカットをネットワーク状に連結し、ユーザがマウスボタンをプッシュしている間はプッシュした点に応じて適切な経路とフレームを選択して表示する。
- ・マウスクリックをした際には、クリックしたマウスボタンの種別（右、左、真中など）に応じて上記とは別に用意したカットを表示する。
- ・表示の際のビデオ再生の速度はユーザがポインティングした位置とあらかじめ設定した基準点との間の距離によって決定する。

例えば、図3.3-3のようなネットワークを持ち、図3.3-4に示すように表示経路選択のために領域をa、b、c、dの4つに分けてあるビデオ部品が表示されている場合を考える。ユーザが図3.3-4の点Pでマウスボタンをプッシュすると、点Pの位置が領域aに入っていることに基づいてシステムが図3.3-3の経路aを選択し、点Pの中心点からの距離Lに基づいてビデオ再生速度を決定する。中心点からの距離が大きければ図3.3-4の本は高速で回転し、小さければ低速で回転する。

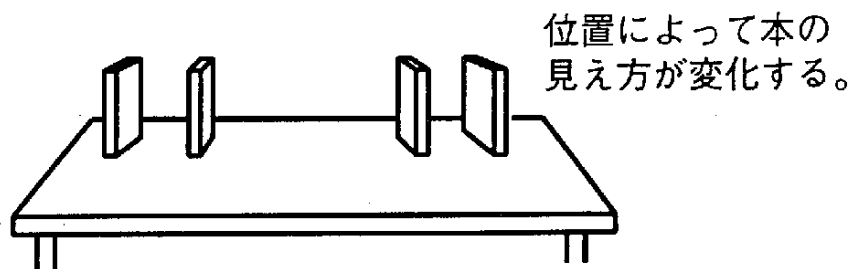


図3.3-5 位置依存ビデオ映像部品

②の位置依存ビデオ映像部品は、画面上で移動する物体を表現する。一般に物理的な環境で物体を

移動すると、一定の場所にいる人に見える物体の側面は変化する。したがって、例えば1冊の本をビデオ映像部品で表示する場合、物理的な環境を模擬するならば、本を向かって右の方に移動すれば本の左側の側面が見えるようになり、左の方に移動すれば右側の側面が見えるようになる必要がある(図3.3-5参照)。用意するカットは①の操作依存ビデオ映像部品の場合と同様、被写体あるいはカメラを回転させながら撮影した映像である。制御方法としては、ビデオ映像部品自身が画面上で占める位置を絶えず監視し、それによって表示フレームを決定する。

③の時間依存ビデオ映像部品は、通常のビデオ再生と同様に映像を一定速度で再生・表示するものであり、被写体の変化の様子を示す際に利用する。短い映像を繰り返し表示している場合や、一定の時刻になったら再生を始めるビデオなどもこれに相当する。状況や機能の説明に利用することを想定している。あらかじめ用意するカットは一つの部品に対して通常一つだけであり、時刻、あるいは表示開始からの時間によって表示フレームを制御する。

### 3.3.4 実装

#### (1) 実装環境

3種類のビデオ映像部品を、オブジェクト指向言語のクラスとして実現した。実現には、UNIXワークステーション(NEC製EWS4800/260)、X-Windowシステム、InterViews(GUIのC++クラスライブラリ[Linton88])、Xavier(InterViewsのAudio-Visual拡張)を利用した。Xavierでは、ビデオ映像をオブジェクトとして画面上に表示して、そのオブジェクトにplay、pause、stopなどのメッセージを送ることで表示を制御できるが[Hamakawa92]、ここではXavierをさらに拡張し、操作依存ビデオ映像部品、位置依存ビデオ映像部品、時間依存ビデオ映像部品に対応するクラスを追加した。

#### (2) リアリティユーザインタフェース構築への利用

既に、ビデオ映像を利用した画面背景にグラフィカルオブジェクトをオーバーレイ表示してそれらの同期制御を行うリアリティユーザインタフェース[神場91]を提案した。従来のウィンドウとアイコンを利用するシステムでは、初心者にとってアイコンの意味が分かりにくい場合などがある。写実的な画面でデスクトップ環境を明示的に実現することで、ユーザにとっての分かりやすさや覚えやすさが向上することが分かっている[神場93]。そこで、ビデオ映像の利用を画面全体に拡張し、ここで示したビデオ映像部品だけを合成することによりデスクトップ環境全体の作成を試みた。ビデオ映像は、次の3種類を利用した。

##### ① 環境

サイドデスク、机、キャビネットの間をビデオカメラをパン(横方向への回転)しながら撮影した映像。

## ② 本

回転台の上に載せた本を180°回転するシーンを撮影した映像。後でデスクトップ背景にオーバーレイ表示する際に本以外の映像部分の表示を抜きやすいように、背景映像は一定の色（この場合は青）で撮影した。これはテレビなどで映像を作成する際に用いられるクロマキー合成と同様の原理に基づいている。

## ③ 操作のシーン

「はさみで紙を切っているシーン（切り取り機能）」「紙を糊で貼っているシーン（貼り込み機能）」「紙をゴミ箱に捨てるシーン（削除機能）」などの映像。それぞれのシーンは1秒程度からなる。

次に、それぞれの映像にビデオ映像部品として適当なクラスを割り当てた。①は操作依存ビデオ映像部品であり、ユーザが映像の右半分をクリックすれば右方向にカメラをパンする映像を表示し、映像の左半分をクリックすれば左方向にパンする映像を表示する。本をクリックすると開いた本の映像が表示される。②は位置依存ビデオ映像部品であり、背景となるデスクトップの映像上で右方向に移動すれば本の左側の側面を表示するように回転し、左方向に移動すれば本の右側の側面を表示するように回転する。③は時間依存ビデオ部品であり、表示されている間は常にシーンを繰り返し再生する。

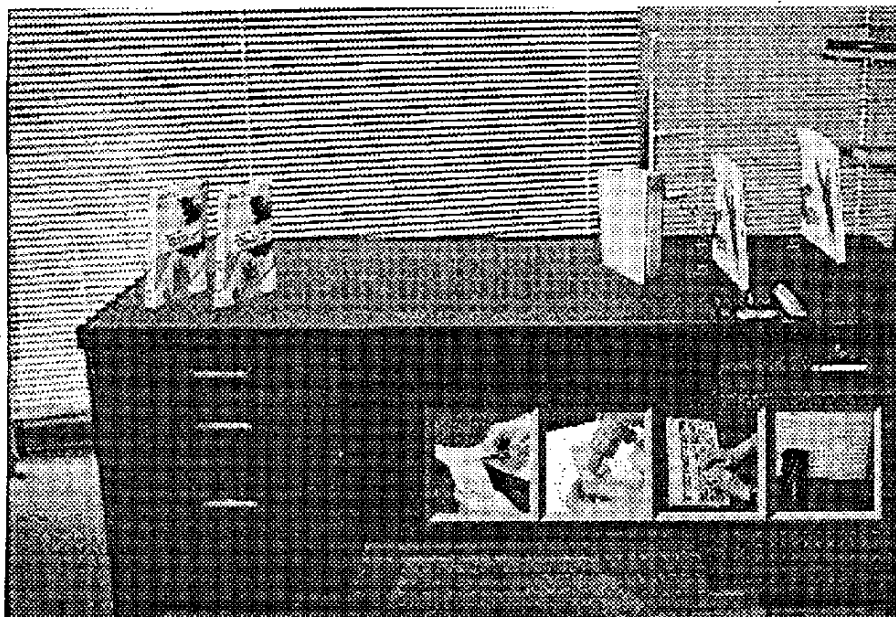


図3.3-6 デスクトップ環境の例

作成したデスクトップ環境の画面を図3.3-6に示す。画面は640×480ドットの領域に表示した。表示されている5冊の本は、すべて同じ位置依存ビデオ映像部品であり、表示位置によって外観が異なる

ことが分かる。中央付近にポップアップしている4つの小さな映像は、選択可能な操作メニューをビデオ映像で示すものである。本の映像を机の映像に合成表示する際には、本の背景を抜いて表示した。これを実際のデスクトップアプリケーションのエンドユーザインタフェースとして利用する場合には、ユーザが本をクリックした際に通常のアプリケーションが開くようにすればよい。従来、直観的に分かりやすいデスクトップ環境を実現することを目的として、3次元グラフィックスを利用してデスクトップ環境を作成した例などもあり[Lovgren87]、ここで示したビデオ映像部品は、そのような3次元グラフィックス環境と組合せて利用することも可能であると考えている。

### 3.3.5 おわりに

従来のマルチメディアアプリケーションにおいて、ビデオ映像は情報表示に利用されるだけであり、ユーザが入力を行うためインタラクション部品として利用されたものはほとんどなかった。ここで述べたのは、複数のカットに関連付けを行ない、ユーザ入力との関係を定義することで、ビデオ映像を画面構成部品とする手法である。ここではビデオ映像部品だけでデスクトップ環境を構成したが、実際にアプリケーションのユーザインタフェースを構築する際には、従来のGUI部品と組合せて利用することが現実的であろう。本研究は、現在メディア処理技術に研究の中心があるビデオ映像を、ユーザインタフェース向上に利用するための方向を示すものであると考えている。

## 【参考文献】

- [Hamakawa92] Hamakawa, R., Sakagami, H. and Rekimoto, J.: Audio and Video Extensions to Graphical User Interface Toolkits, Proceedings of the Third International Workshop on Network and Operating Systems for Digital Audio and Video, San Diego, California, U.S.A. (1992).
- [神場91] 神場, 久松: 実写を利用したマルチメディアユーザインタフェースの試作 - リアリティユーザインタフェース -, 情報処理学会第43回全国大会予稿集4F-5 (1991).
- [神場93] 神場, 橋本: リアリティユーザインタフェースの提案と試作 - マルチメディアユーザインタフェースの試み -, 情報学論, Vol.34, No.11 (1993).
- [Linton88] Linton, M.A., Calder, P.R., and Vlissides, J.M.: InterViews: A C++ Graphical Interface Toolkit, Technical Rep., No. CSL-TR-88-358, Dept. of Computer Science, Stanford Univ. (1988).
- [Lippman80] Lippman, A.: Movie-Maps: An Application of the Optical Videodisc to Computer Graphics, SIGGRAPH'80, Vol.14, No.3, pp.32-42 (1980).
- [Lovgren87] Lovgren, J.E. and Poltrock, S.E.: The Visual Office: An Experiment in 3-D User Interface Development, MCC Technical Rep. No. HI-321-86 (1987).

[渡部91]渡部,阪田,前野,福岡,大森: マルチメディア分散在席会議システム MERMAID, 情処学論,  
Vol.32, No.9, pp.1200-1209 (1991).

[山足91]山足,谷,谷越,二川,谷藤,平沢: 被写体の構造を持ったカメラ, 計測自動制御学会第7回ヒュー  
マンインタフェースシンポジウム論文集, pp.461-468 (1991).

### 3.4 3次元GUI構築ツール

#### 3.4.1 仮想操作環境

近年、グラフィカルユーザインタフェース（以下、GUI）が多くのシステムで使われるようになり、ユーザが直接操作によって直感的に操作しやすいインタフェースが一般的になってきている。これが可能となったのは、メニュー、アイコン、ボタンなどのGUI部品を持つプラットフォームが整備され、多くのアプリケーションプログラマが、これらGUI部品を組合せることによって、ユーザインタフェースを構築できる環境が整ったからであるといえる。しかし、一方で、このようなGUI環境で提供されるGUI部品は、計算機能力の飛躍的な向上にもかかわらずXerox Starシステムの頃から大きな変化は見られない。この理由は、現在のGUIの形が、(1) 2次元表示に基づくユーザインタフェースの規格化、(2) アプリケーションからの独立性、および、(3) 設計者のための有限の部品セットの提供という制約の下で最適化したことによる限界であるといえる。

多くの場合、人間は具体的、空間的に表現されている対象の方が、抽象的、論理的な表現よりも記憶や理解が行いやすい[Caroll80]。この特性を利用したユーザインタフェースの試みとして、HI-VISUAL'89のオフィス情報処理[田原89]、Monsterの仮想オフィス[勝山92]、TVメタファ[FRIEND89]などがある。これらの共通の目的は、オフィスやテレビというメタファをできるだけ具体的なイメージでユーザに提示することによって、より分かりやすいユーザインタフェースを提供することにある。しかし、このような直感的なメタファを表示するためには現状のGUIでは表現力に限界がある。

これを可能とするために、仮想操作環境として次の性質を備えた新しいGUIの枠組みを提案する。

- ① 3次元モデルで構成された任意のオブジェクトが3次元空間内に配置される。
- ② 各オブジェクトはユーザの操作またはアプリケーションからのイベントに応じて形状および位置等が変化し、それがアニメーションで表示される。
- ③ 複数のアプリケーションがこれらの3次元オブジェクトの動作と連携して動作する。
- ④ ユーザは3次元空間内で視点を自由に変更することが可能であり、3次元空間内に配置された3次元オブジェクトを利用することでタスクを行える。

3次元GUI構築ツールMECOT (Metaphor Environment Construction Tools) は、このような仮想操作環境を構築/実行する環境を提供することにより、現状のGUIが持つ表現力の限界を解消する新しい枠組みを提供するシステムである。以下、MECOTの実現方式、および、MECOTによって作成した仮想操作環境の具体例について説明する。

#### 3.4.2 実現方式

##### (1) システム構成

MECOTは、図3.4-1に示すようにクラスオブジェクトエディタ、操作環境ビルダ、操作環境マネジ

の3つのツール群により構成される。

クラスオブジェクトエディタおよび操作環境ビルダは構築ツールであり、これらの2つのツールを使ってメタファの部品化を図る。クラスオブジェクトエディタでは、3次元CADによって作られた3次元モデルとそれらの組合せや動きを記述したクラスオブジェクト定義によってインタフェースメタファのクラスを作る。操作環境ビルダでは、インタフェースメタファのクラスからそれらのインスタンスを生成し、3次元空間内に配置することによって仮想操作環境を構成する。

操作環境マネージャは実行環境であり、構築ツールによって構成された操作環境定義を実際のアプリケーションと連動した形で利用できるようにする。

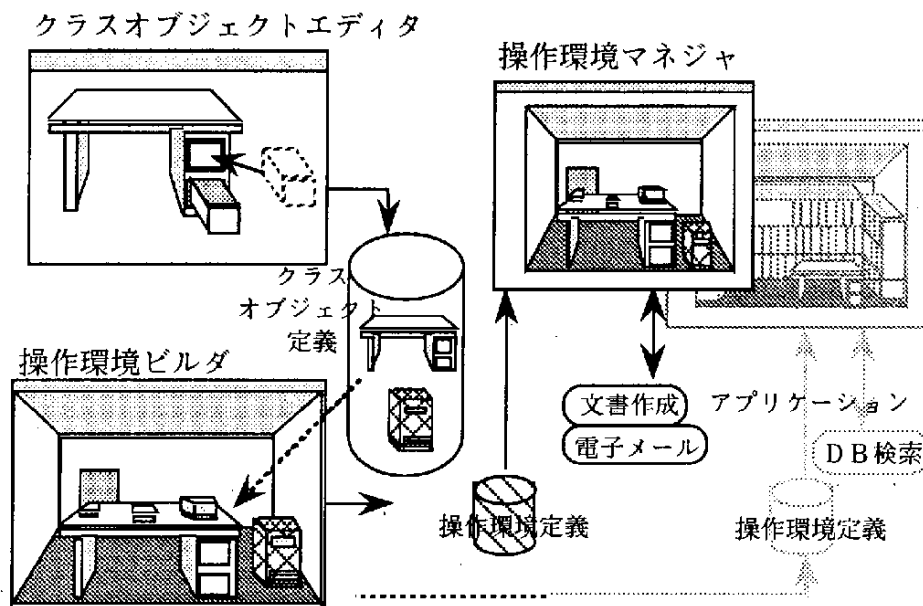


図3.4-1 MECOTのシステム構成

## (2) 階層的部品化モデル

MECOTでは、従来のGUIで採られているような、あらかじめ定義された少数の汎用的なクラスオブジェクトを組合せるという方法ではなく、新たなクラスオブジェクトもユーザインタフェース設計者が定義することを可能としている。このとき新たに定義するクラスオブジェクトは、形状だけではなく、その動きも同時に記述される必要があるが、その際に重要となるのは、再利用性の高いクラスオブジェクトの部品化の枠組みである。

後述する3次元オフィス環境を例にとると、本システムが目指す仮想操作環境では、机、書類、ゴミ箱、部屋などの対象が3次元的に表示され、これらの対象は、それぞれがユーザの操作に応じて動作すると同時に、互いに関連する動作も行う。例えば、机の引き出しが開閉する動作は机の動作であ

るが、ドキュメントを机の引き出しに収納する動作は机とドキュメントの2つの対象が関与する動作である。そのため、クラスオブジェクトを部品化するためには、1つのオブジェクトに閉じた動作定義と複数のオブジェクトが関与する動作定義を切り分けた上で、前者を部品化することを考えなくてはならない。

MECOTでは、これをコンポーネントレベル、オブジェクトレベル、環境レベルという3つのレベルからなる階層化を採用することによって解決している（図3.4-2）。

コンポーネントレベルでは、オブジェクトの構成要素を定義する。この構成要素は、オブジェクトの部品となる3次元形状（ポリゴンデータ）とメニュー、ボタンなどの従来型GUIのユーザインタフェースオブジェクト（ウィジェット）である。

オブジェクトレベルでは、コンポーネントを所定の位置に配置することにより1つのクラスオブジェクトを定義する。クラスオブジェクトは、いくつかの状態を持ち、状態遷移によってその動作を定義する。クラスオブジェクトの各状態には、それぞれ異なったコンポーネントの配置をとるグラフィカルな状態が対応しており、状態の遷移はグラフィカルな変化に対応する。

環境レベルでは、オブジェクトレベルで定義されたクラスオブジェクトからインスタンスを生成し、それを所定の位置に配置することによって定義する。環境レベルでは、複数のオブジェクトが関与するような動作および環境を考慮することによって可能となる動作が定義される。

このように、コンポーネントレベル、オブジェクトレベル、環境レベルという3つのレベルに定義を階層化することによって再利用性の高い部品化が可能となる。例えば、コンポーネントレベルで作られたコンポーネントは、様々なクラスオブジェクトの構成要素として利用可能であり、また、オブジェクトレベルで作られたクラスオブジェクトを利用して、様々な仮想操作環境を構成することができる。

### (3) インタフェース記述言語

以上の部品化方式を反映して、仮想操作環境全体を定義するための記述言語を開発した。本記述言語は後述する操作環境マネージャによって解釈され、仮想操作環境の構築および制御するためのデータとして扱われる。

本記述言語は、図3.4-3に示すような階層を持ち、それぞれ独立に記述する。個々の記述言語の説明を表3.4-1に示す。

以上のようにインタフェース全体の記述を階層的に分割してそれぞれ独立に記述することから、インタフェースの仕様変更に伴う変更が局所的に行え、またメタファとアプリケーションの組合せに自由度を持たせることができる。

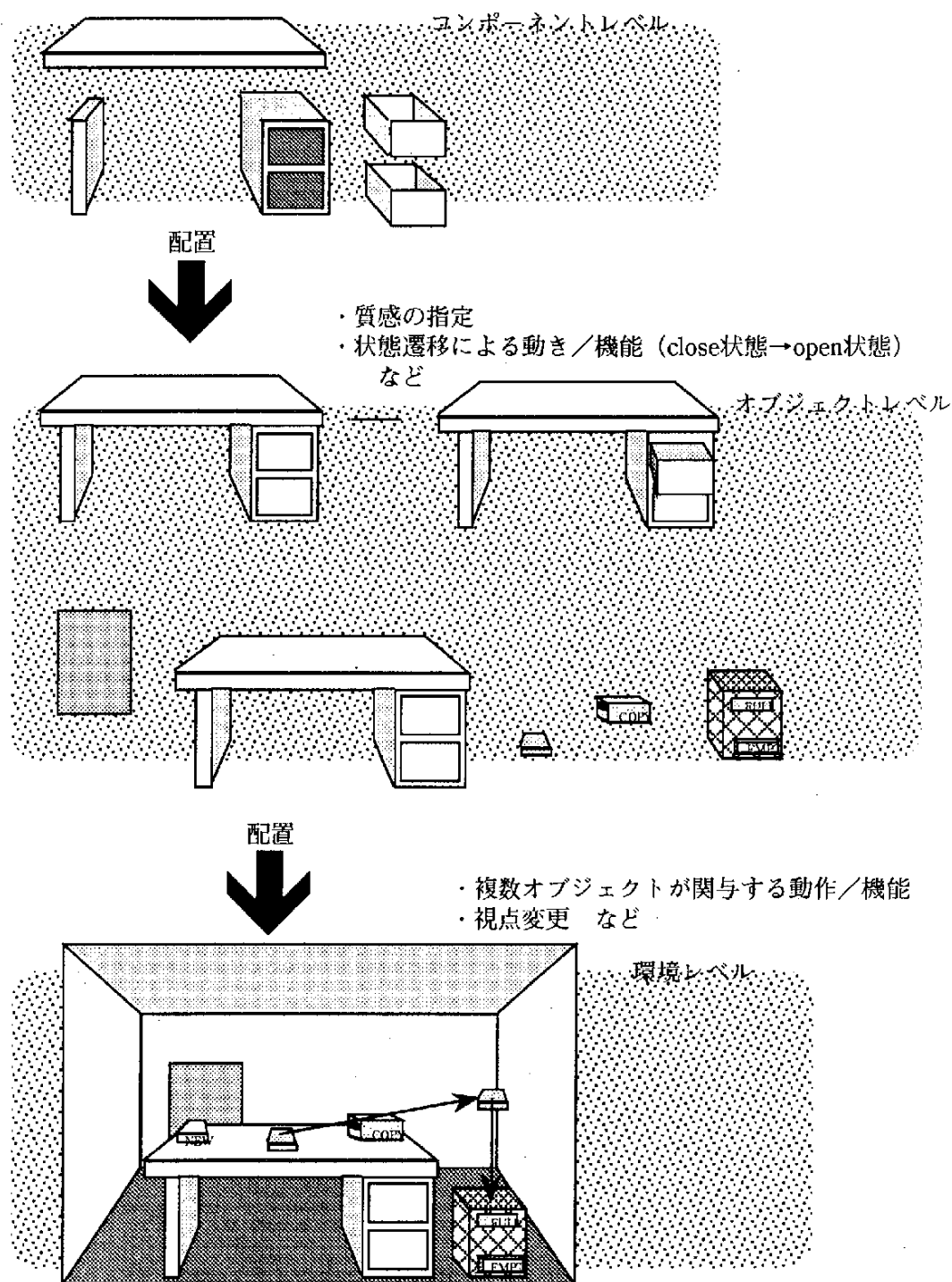


図3.4-2 コンポーネント/オブジェクト/環境

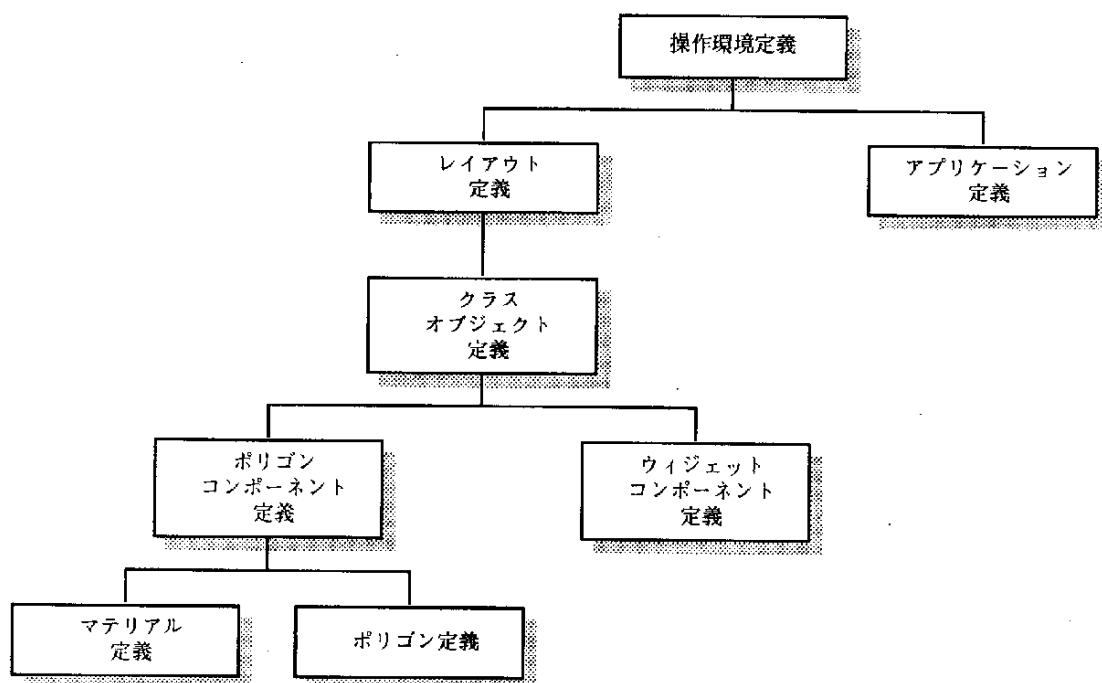


図3.4-3 記述言語の階層

表3.4-1 記述言語の説明

| 記述言語            | 説明                                                                                        |
|-----------------|-------------------------------------------------------------------------------------------|
| 操作環境定義          | 操作環境定義では、ユーザおよびアプリケーションからのイベントによる各オブジェクト間の相互作用動作など、構築するインタフェース全体のメカニズムを記述する。              |
| レイアウト定義         | レイアウト定義では、クラスオブジェクト定義で記述されたクラスオブジェクトのインスタンスを3次元空間内にどのように配置するかによって初期状態におけるインタフェースの外観を定義する。 |
| クラスオブジェクト定義     | クラスオブジェクト定義では、操作対象となるクラスオブジェクトを構成するコンポーネントの配置、クラスオブジェクトの属性および固有の動作などを記述する。                |
| アプリケーション定義      | アプリケーション定義では、インタフェースと独立して動作するアプリケーションとの通信プロトコルを記述する。                                      |
| ポリゴンコンポーネント定義   | 3次元CADで作成された3次元コンポーネントのポリゴン情報およびその色を定義する。                                                 |
| ウィジェットコンポーネント定義 | ウィジェットの配置、色などを定義する。                                                                       |
| ポリゴン定義          | 頂点座標、ポリゴンの法線ベクトルなどによってポリゴンを定義する。                                                          |
| マテリアル定義         | 環境光の反射、拡散反射、鏡面反射、透過率によってポリゴンコンポーネントの色を定義する。                                               |

### 3.4.3 仮想操作環境の具体例

#### (1) 3次元オフィス環境

MECOTによる仮想操作環境の事例として、電子メール、文書作成、文書検索を行う3次元オフィスルーム環境を作成した。

この仮想操作環境は、パーソナルな作業空間としての部屋（パーソナルルーム）、会議を行うパブリックな空間としての部屋（パブリックルーム）、たくさんの文書を格納しその検索をするための部屋（ライブラリルーム）という3つの部屋により異なった種類の仕事を空間的に配置するというコンセプトで作られている。（図3.44）

パーソナルルームは、個人的なドキュメント作成およびドキュメントの整理を行う空間である。机の上のドキュメントを選択すると、そのドキュメントそのものがエディタとなり編集可能となる。このドキュメントは電子メールとして送信することも可能である。電子メールとして送信したドキュメントは部屋の外に飛んでいくというアニメーションによって直感的にメール送信が理解できるようになっている。メールが来ると逆に部屋の外からドキュメントが飛んでくる。飛んできたメールはドキュメントと同様に編集/保存が可能である。机の上は作業中のドキュメントが置かれており、また、一次格納のためには引き出しという格納場所が用意されている。ゴミ箱は不要なドキュメントを捨てるためのメタファである。机が一次的なドキュメントの格納を表わしている一方、部屋の後ろには個人のファイルキャビネットが置かれており、ここで個人のファイルを検索し、編集のために取り出すことができる。

パーソナルルームでは個人ファイルの編集、検索を行うのに対して、ライブラリルームは、多くの人と共有しているさらに大きなデータベースからのファイルの検索を行う場所である。ここにはパーソナルルームで使われているファイルキャビネットメタファが再利用されて置かれている。

また、パブリックルームは、将来、ネットワーク上の他のユーザとの電子会議を行うという利用を想定した会議室メタファとなっているが、現在はアプリケーションとのリンクは行っていない。

#### (2) 動作の説明

MECOTの上で3次元オフィス環境がどのように動作するかについて説明する。

図3.45は、仮想操作環境の概念的な実行モデルを表わしたものである。クラスオブジェクトは、操作環境マネージャ上では、そのインスタンスオブジェクトが表示され、アプリケーション機能の表象としてユーザに提示される。これらのインスタンスオブジェクトの動作は、オブジェクトレベルの動作についてはクラスオブジェクト定義、また、環境レベルの動作については操作環境定義を解釈することにより実行される。これらの動作は、基本的にはイベントとそれに対する動作記述という形式で書かれている。アプリケーションからのイベントおよびアプリケーションへの処理要求は環境レベルの動作記述の中に書かれる。

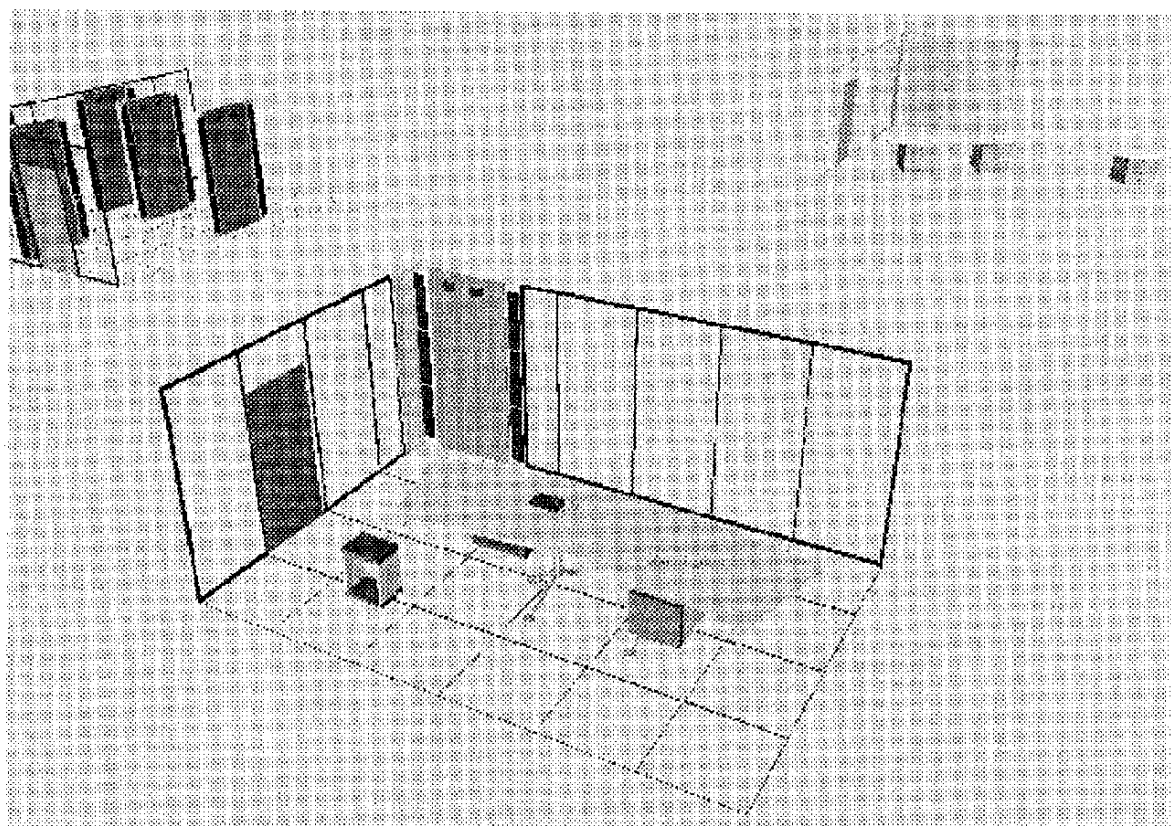


図3.44 3次元オフィス環境の全景

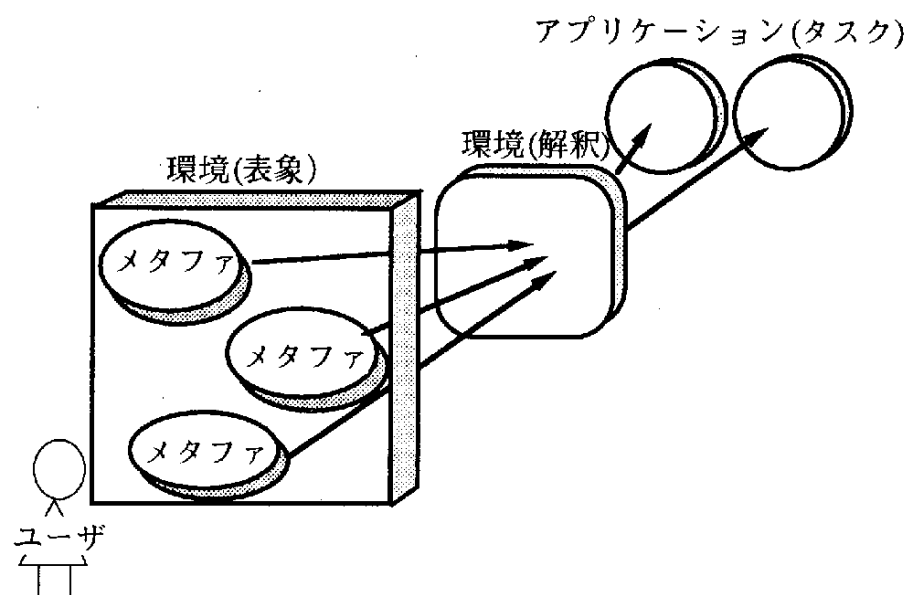


図3.4-5 仮想操作環境の概念モデル

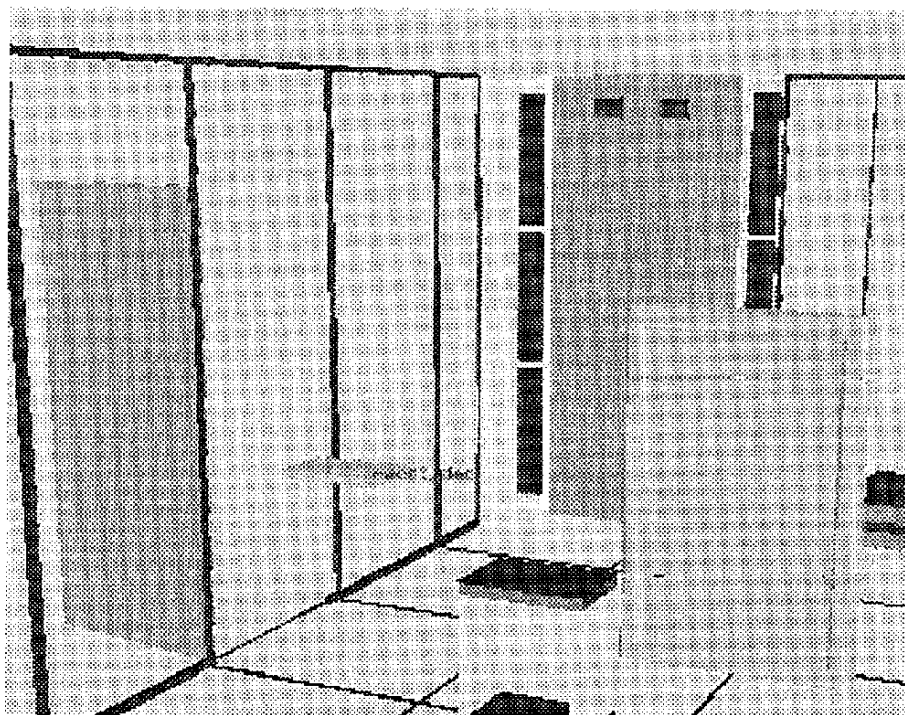


図3.4-6 メール送信画面例

図3.4-6はオープンしたドキュメントをメールとして送信する画面の例である。まず、ユーザのドキュメントインスタンスに対する操作は、ドキュメントクラスの以下の動作ルールを発火する（図3.4-7上段）。

```
mail_open -> mail_closed
trigger: { select(doc_base_component) }
effect: {
  send_mail();
  fly_mail(integer 10);
}
```

ここで発火された動作ルールには、電子メールアプリケーションに、ドキュメントインスタンスが表わしているファイルを送信する要求を出すアクション（send\_mail）とドキュメントインスタンスが部屋の外に飛んでいくと同時に部屋のドアが開閉するアニメーションを表示するアクション（fly\_mail）が記述されている。

send\_mailは、アプリケーション定義に書かれているメッセージプロトコルに従ってプロセス間通信によって処理要求を送信する（図3.4-7中段）。fly\_mailは、操作環境定義においてアクション関数と

して次のように定義されている。

```
fly_mail(integer step) {  
    event(room1, open_door);  
    fly(CALLER, behind_door, *step);  
    event(room1, close_door);  
    delete(CALLER);  
}
```

ここでevent(room1, open\_door)はroom1というインスタンスにopen\_doorというイベントを送ることであり、room1はこのイベントを受け、部屋クラス定義に従って、もしドアが閉じていたらドアを開くアクションを実行する(図3.4-7下段)。fly(CALLER, behind\_door, \*step)において、flyは操作環境マネージャが持つアニメーション表示のための組み込み関数で、飛ばす対象となるオブジェクトインスタンス、飛ばす行き先を表わす仮想エリア、アニメーションのステップ数を引数とする。CALLERはこの関数を呼び出したオブジェクトインスタンスを示す特別な変数である。最後にroom1にclose\_doorイベントを送信し、ドキュメントインスタンス自身はdelete関数により消滅する。

以上の流れを図3.4-5の概念モデルに基づいて図示すると図3.4-7のようになる。

#### 3.4.4 まとめ

MECOTは、形状、質感、動き、アプリケーションの記述を分離し、階層的な依存関係を持たせることにより、様々なレベルでの交換や再利用が可能となっている。このため、形状や質感はデザイナーにまかせる一方で簡単な形状を用いて動きの定義やアプリケーションを作成し、後でそれらを組合せることなどが可能となる。

ツールとしての有効性を確認するため、簡単な仮想操作環境のコーディング量をC言語とグラフィックライブラリで行った場合、すべて定義言語で記述した場合、構築ツールを用いて行った場合とで比較した。

この結果、定義言語では、C言語とグラフィックライブラリで開発するものと比較して約1/20のコーディング量で行えることが分かった。また、構築ツールを用いることにより、配置に関する定義が直接操作で行えるため、コーディング量はさらに半分となるという結果が得られた。

このように簡単に記述可能な定義言語とするために、その記述能力を制限したことも事実である。例えば、オブジェクトのアニメーションは、構成要素であるコンポーネントの変位に基づいて定義されるが、この変位が定数で与えられるため、アニメーションを実行時に変更することは現在不可能となっている。また、コンポーネントは変化しないものとして定義するため、コンポーネントの形状が変化するようなアニメーションも表現できない。一般に、記述能力と記述効率はトレードオフの関係

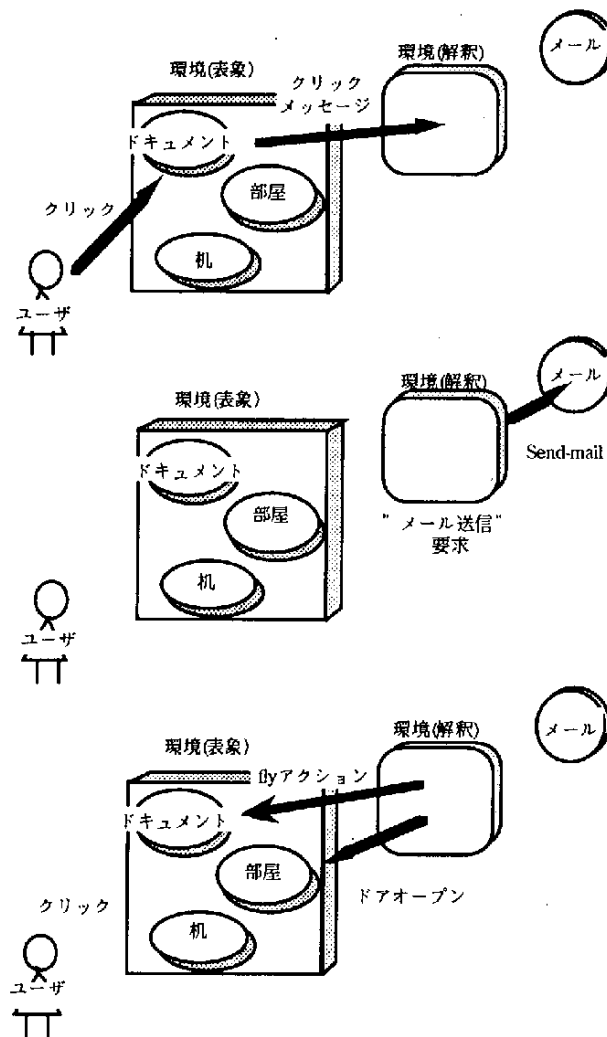


図3.4-7 実行の流れ

にあると考えらるので、設計者の能力や要求に応じて段階的に詳細なレベルの記述が可能となる枠組みを提供していくことが必要であろう。

### 【参考文献】

- [Caroll80]Caroll, J.M., Thomas, J.C., Malhotra, A.: Presentation and representation in design problem-solving, British Journal of Psychology, 71, pp.143-153 (1980).
- [FRIEND89]FRIEND21研究センタ: 昭和63年度情報化推進基盤整備未来型分散情報処理環境基盤技術開発研究報告書, pp.36-38 (1989).

[勝山92]勝山恒夫他: MONSTERのヒューマンインタフェース, 情報処理学会研究報告92-HI-40(3) (1992).

[田原89]田原, 吉見, 平川, 田中, 市川: 視覚的プログラミング空間に関する一考察, 情報処理学会研究報告98-SE-64(16), pp.121-128 (1989).

### 3.5 アイコニックデータベース操作システム

ここでは、データベース環境への視覚情報の活用事例として、広島大学の視覚的データベース操作システムIconicBrowserについて述べる[Tsuda90]。このシステムは当初はデータベース検索ツールとして開発されたが、最近ではデータベース定義/変更も行えるように機能拡張されている[Yoshitaka93]。なお文献[Yoshitaka93]にはデータ定義を支援するためにconfigurationなどといったアイデアも導入されているが、ここではユーザ操作環境における視覚化のアプローチに話題を限っている。

#### 3.5.1 研究背景

データベース操作に視覚情報を持ち込むことによってシステム操作性を改善しようとする試みは、かなり以前から行われている。IBM社のZloofiによって開発されたQBE (Query-By-Example) はその代表であり、最初の提案は1975年になされている[Zloof75]。その後、ERC [Czejdo89]、Tabletalk [Epstein91]、OOQBE [Staes91]、OQD [Kwak93]、VQL [Mohan93]などといった多くの視覚的データベース操作システムが提案・開発されてきているが、それらにおいて可視化の対象はいずれもデータベース構造(スキーマ)である。このようなアイデアはデータベース応用を意識した時には自然であり、データベースを恒常的に利用するユーザ(例えば図書館司書)にとっては非常に強力なツールとなる。ただし一般の不特定多数の人々がごく稀に使用するような状況においては、ユーザにデータベース構造を見せてもほとんど意味はなく、かえって誤解を生じる原因にもなる。

ここで述べるIconicBrowserシステムは、そのような従来のアプローチとは逆に、データベース構造をユーザから隠すことによってユーザ親和性を高めようとしている。視覚化されるのは概念や現実世界に存在するものである。なお、IconicBrowserはデータベース検索だけでなく、定義/変更機能もサポートしている。また、システムが基盤としているデータベースモデルはオブジェクトオリエンテッドモデル(特にMORE [Tsuda91])であるが、提案されている考え方は必ずしもそれに限定されるものではなく、関係モデルなどの異なったデータベースモデルへも適用可能である。

#### 3.5.2 データベース操作の枠組み

本システムにおけるデータベース問い合わせ手順は、大まかには図3.5-1に示すような状態遷移で説明できる。図3.5-2に与えるデータベース構造の下で、あるVTRを選び出す操作を例に、その手順を具体的に説明する。

まず、あるクラスを選択することから問い合わせが開始される。例えばElectrical\_Applianceクラスが選ばれたと仮定する(Class状態)。最終的に選び出したいのはVTRであるので、Visualクラスを介してVTRクラスに移動する。次に、VTRクラスに用意されているインスタンス変数に必要な条件付け

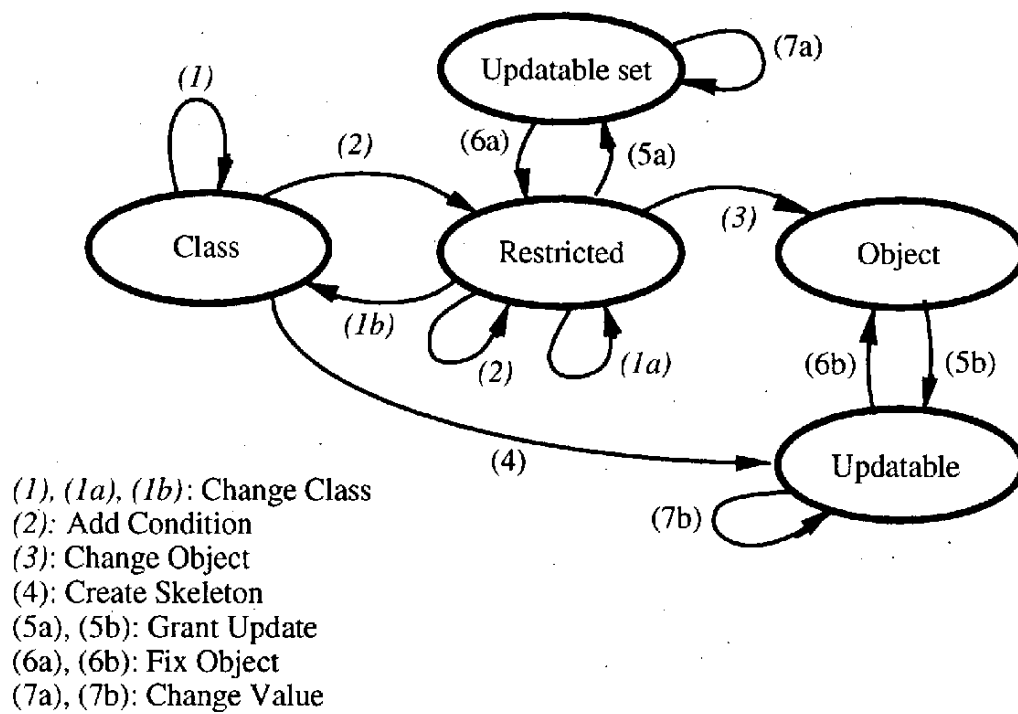


図3.5-1 状態遷移モデル

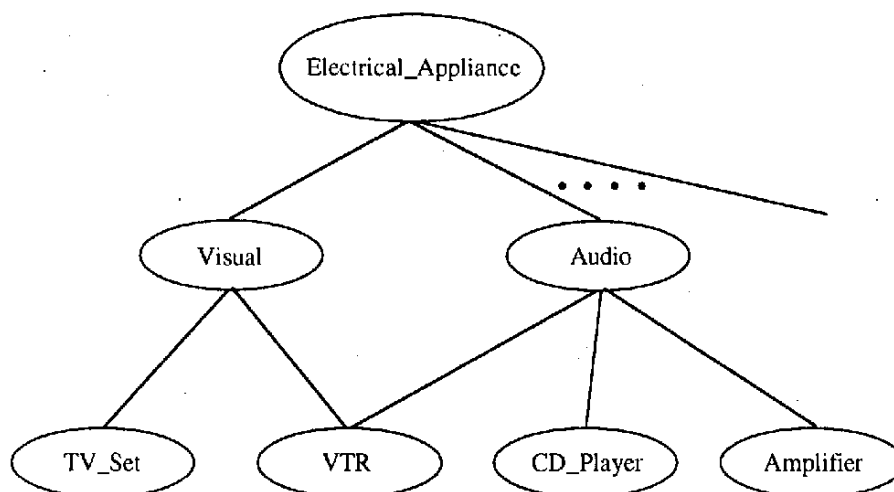


図3.5-2 例スキーマ

を行ない、制約を加えていく（Restricted状態）。そうして最終的に所望のオブジェクトあるいはオブジェクトの集まりを得る（Object状態）。

なお、オブジェクト（の集合）に対する変更操作は、システムがUpdatable（Updatable set）状態に移行後に適用可能である。Updatable状態へはClass状態またはObject状態から遷移できる。前者は、該当するクラスのインスタンスとしてのオブジェクトを生成する際のものである。既存のオブジェクトの内容を変更する場合は、後者のように、Object状態からUpdatable状態への遷移によって実行される。なおUpdatable set状態でのオブジェクト変更の際には、該当するすべてのオブジェクトの内容が変更される。逆に言うと、そのような変更が可能な場合に限って（例えば、該当するオブジェクトすべてが共通に持っているインスタンス変数の）変更操作が許されており、適用可能な操作はUpdatable状態でのそれと比べて限定されている。

状態遷移を引き起こす操作を状態間のリンク上に示す。それらの操作に関する詳しい説明は後で行う。

ところで、上記の状態遷移モデルはあくまでもシステム内部での処理内容を示しており、ユーザはそれを一切意識する必要はない。ユーザの目に見えるものは、それぞれの状態に対応したアイコンのみである。VTR検索の場合を例にとって具体的な視覚イメージを図3.5-3に示す。図に示すように、独自のアイコンが用意されるのはClassならびにObjectの両状態であり、その他の状態は複数のアイコンの組合せでもって表現される。例えば、VTRアイコンに値段アイコンとHi-Fiアイコンが組み合わされてRestricted状態が表現されている。またUpdatable状態は鉛筆アイコンとの組合せとして表現される。

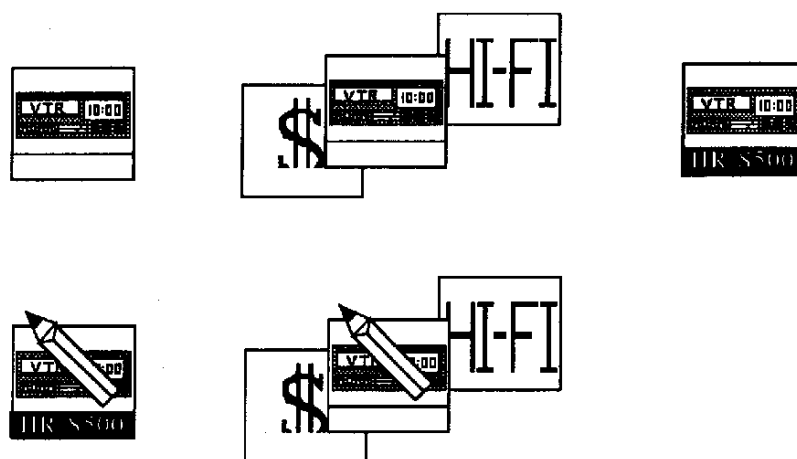


図3.5-3 アイコンによる状態の視覚化

この例に示されるように、システムではアイコンを重ね合わせることによって問い合わせを記述する。

### 3.5.3 基本演算

システムに提供されている内部的な基本操作を以下に説明する。なおそれらは状態間を遷移するための遷移操作群と内容（値）を操作するためのアイコン操作群の二つに大別される。

#### (1) 遷移操作群

遷移操作群には `Change_Class`、`Add_Condition`、`Change_Object`、`Create_Skeleton`、`Grant_Update`、`Fix_Object` の6つがある。前者3つは検索用であり、残りの3つが定義/変更用に用意されたものである。個々の操作について以下に説明する。

##### (a) `Change_Class`

あるクラスから、その親クラス、子クラス、あるいは兄弟クラスのいずれかに遷移するための操作であり、元のアイコンが `Class` または `Restricted` の状態にあるときに限って実行可能である。もしも `Restricted` 状態にあったときには、新たなクラス状態に遷移した後に、既に定義されている条件を受け継ぐかどうかユーザに確認される。なお画面上では、希望する遷移先のクラスを表わすアイコンを重ねることで本操作が起動される。

##### (b) `Add_Condition`

クラスの有する属性に制約を加える操作である。この操作は図3.5-4の画面左側に示すように、属性に対応したアイコンをクラスのアイコンに重ね合わせることで表現される。なお図中にも見られるように、属性のタイプによって、値段アイコンのように条件（例えば115,000円より高く168,000円より安い）を持つものと、Hi-Fiアイコンのようにそれ以上の具体的な条件設定が必要とされないものがある。なお、システムでは重ね合せた制約はすべてAND条件と見なされる。OR条件を設定したい場合はアイコン操作群の(c)に示すような別の操作を用いる。

##### (c) `Change_Object`

`Class` または `Restricted` の状態から `Object` 状態へと遷移するための操作である。ユーザが明示的にこの操作を指定することもできるが、制約付けしていった結果該当するオブジェクトの数が一つになったときには、システムによって自動的に本操作が起動される。

##### (d) `Create_Skeleton`

これは新しいオブジェクトを生成させるための操作である。生成を希望するオブジェクトのクラスに対応したアイコンに鉛筆アイコンを重ね合わせることで記述する。もちろん生成直後のオブジェクト（アイコン）においては、デフォルト以外はインスタンス変数に具体的な値は記述されていない。アイコン操作群(h)の `Add_Property` を用いて具体値を設定する。

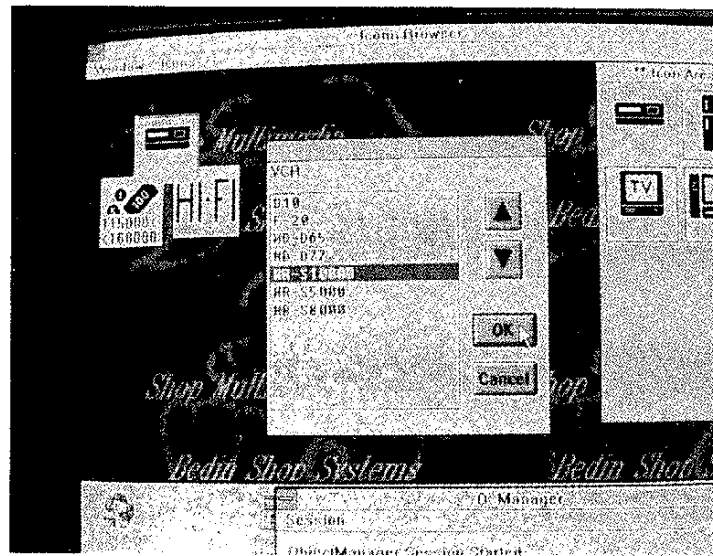


図3.5-4 制約条件の記述

(e) Grant\_Update

データベース更新操作を行なうに当たって起動される操作である。この操作の実行が完了してはじめて更新が可能となる。先のCreate\_Skeleton操作と同様に、鉛筆アイコンを重ね合わせることで指示する。重ね合わせられるアイコンはRestrictedあるいはObject状態のアイコンのいずれかである。

(f) Fix\_Object

Grant\_Update操作と対をなすもので、更新操作の完了を指示するものである。この操作が実行されると、それまでに行った変更操作の結果が実際にデータベースに書き込まれる。本操作の起動に当たっては、磁気ディスク装置の形状のデータベースアイコンを対象のアイコン上に重ね合わせる。

(2) アイコン操作群

アイコン操作群にはShow\_Object、Icon\_Set、Set\_UnionPlane、Send\_Message、Association、Show\_Property、Show\_Additional\_Property、Add\_Property、Delete\_Property、Change\_Valueの合計10種がある。前者5つは検索用であり、残りの5つが定義/変更用に用意されたものである。個々の操作について以下に説明する。

(a) Show\_Object

指定されたアイコンに連想付けられているオブジェクトの一覧を表示するための操作である。図3.5-4に示したように、表示は独立したウィンドウに出力される。制約が加えられている状態では、そ

の制約を満たすオブジェクトだけが表示対象となる。

(b) Icon\_Set

ある指定したアイコンに適用可能なアイコンの一覧を表示するための操作であり、Show\_Objectの場合と同様に、表示用に独立したウィンドウが現われる。VTRアイコンに対して本操作を起動したときの様子を図3.5-5に示す。いうまでもなく、表示される属性にはVTRクラスに定義されたものとその上位クラスで定義されたものの両者が含まれている。

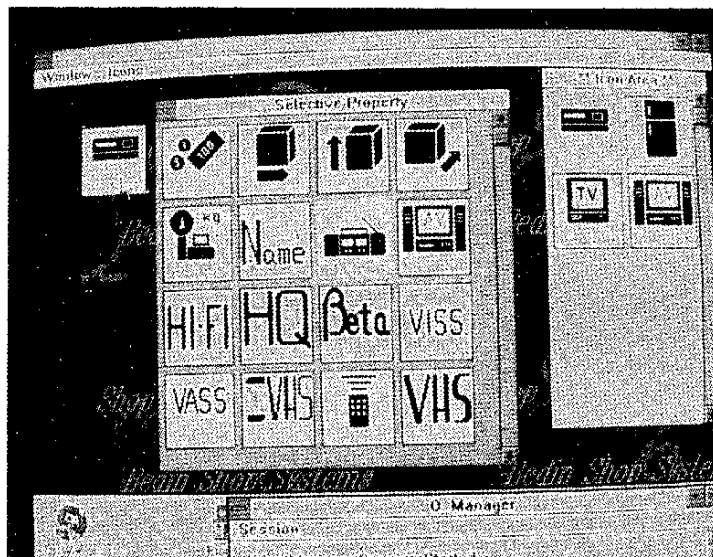


図3.5-5 適用可能アイコンの表示

(c) Set\_UnionPlane

この操作を起動させることにより、OR条件を記述するためのUnionPlaneと呼ばれるウィンドウが現われる。このウィンドウ上で指示された複数の条件項目の間でOR演算が評価される。

(d) Send\_Message

本操作はObject状態にあるアイコンにメッセージを送るためのものである。操作が起動されるとまず受理可能なメッセージ（すなわち起動可能なメソッド）の一覧がメニューの形で表示されるので、その中から希望するものを選ぶ。なお、本システムはMORE[Tsuda91]と呼ばれるオブジェクトオリエンテッドモデル上に実現されているが、MOREではメッセージ送付によってオブジェクト構造（アグリゲーション階層）を変更することができるようになっており、そのためのメッセージ（structural messageと呼ばれる）も表示される。

(e) Association

### 3. ビジュアルインタフェースの事例

Associationは関係データベースにおける結合 (join) 演算に相当する操作を実現する。具体的には、joinアイコンと呼ばれるアイコンを用いて記述する。

(f) Show\_Property

データベース変更に当たり、対象としているオブジェクトが現在保持している属性項目 (すなわちそのオブジェクトを構成している子オブジェクト) を表示させるための操作である。

(g) Show\_Additional\_Property

この操作はShow\_Propertyとは逆に、現時点ではオブジェクトの構成要素とはなっていないけれども、追加定義可能であるような属性項目を表示するものである。適用可能なものだけが表示されるので、構造的に許されない誤った定義をすることが避けられる。

(h) Add\_Property

オブジェクトに新しく項目を追加するための操作である。いうまでもなく追加されうる項目はShow\_Additional\_Propertyを用いて提示されるものに限られる。値を持ち得る項目については、ユーザは具体的な値を設定するようシステムから促される。

(i) Delete\_Property

Add\_Propertyとは逆に、オブジェクトに定義されている項目を削除するための操作である。当然のことながら、削除された項目はShow\_Additional\_Property操作によって表示されることになる。

(j) Change\_Value

属性項目の値を変更するに当たって起動される操作である。

### 3.5.4 データベース操作例

ここでは、具体例を用いながらシステムの操作法を例示的に説明する。ここでは、「価格が115,000円以上かつ168,000以下のHi-Fi方式VTRを探せ」という問い合わせを考える。

画面は図3.5-6に示すように、Icon Area (画面右端) とそれ以外の検索条件記述領域から構成されている。Icon Areaには問い合わせに用いられるクラス/オブジェクトのアイコンが表示される。図中では、VTR、冷蔵庫、TVそしてAV (Audio-Visual) に対応したアイコンが用意されている。

VTRを選び出すといった目標に従って、まずVTRアイコンを選んで検索条件記述領域に移動する。このアイコン上でマウスボタンを押すことにより、図3.5-5に示したような問い合わせ条件付け可能項目がSelective Propertyウィンドウに現われる。Selective Propertyウィンドウの最上段左端にある値段アイコンを選んでVTRアイコンの上に重ね合せ、値段アイコン上でマウスボタンを押すと、データベースに登録されているVTR製品の値段の一覧が表示される (図3.5-7)。この上で値段に関する条件 (115,000円以上かつ168,000以下) を指定する。

このように、アイコンは関連付けられているオブジェクトの値の参照を許すとともに、必要な制約

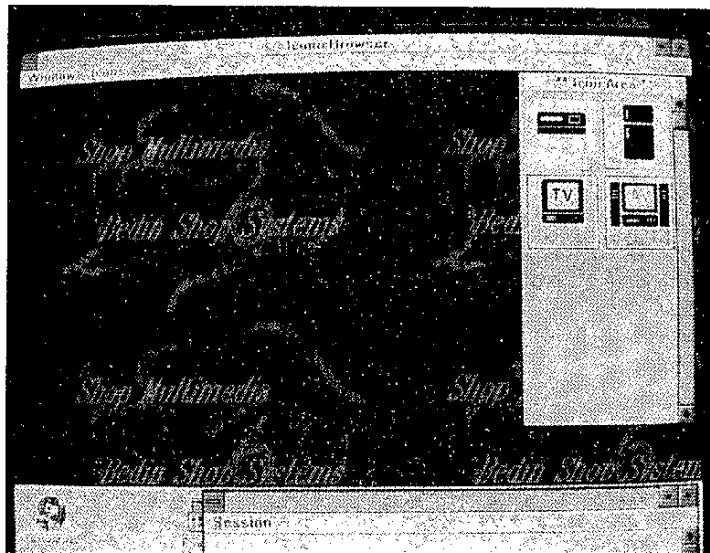


図3.5-6 初期画面構成

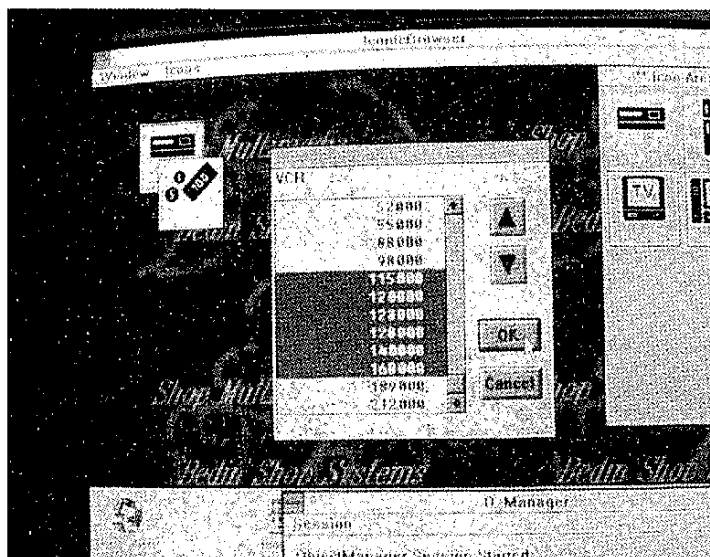


図3.5-7 値段の表示と条件付け

付けを行わせるインターフェースも兼ねている。さらに、条件付けは重ね合わせが行われているときだけ有効であり、検索対象アイコンから離された制約（アイコン）はキャンセルされる。例えば値段アイコンをVTRアイコンから引き離すと、その値段の制約は取り除かれる。再び重ね合せると制約が元どおり課せられる。

さて先程の値段アイコンを選んで重ね合せたのと同じ手順で、今度はHi-FiアイコンをVTRアイコン

ンに重ね合せる（図3.5-4）。それらの条件に合致するオブジェクトが7つあることが分かる。そのうちの一つHR-S10000を選ぶとアイコンは図3.5-8に示すように変化する。選んだVTRの製品名がアイコン下部に表示されている。システム内部では、アイコンの状態がクラスからオブジェクトに変化しているわけであるが、ユーザにはVTRアイコンが依然として見えており、システム内部変化を意識せずに問い合わせを続けることができる。

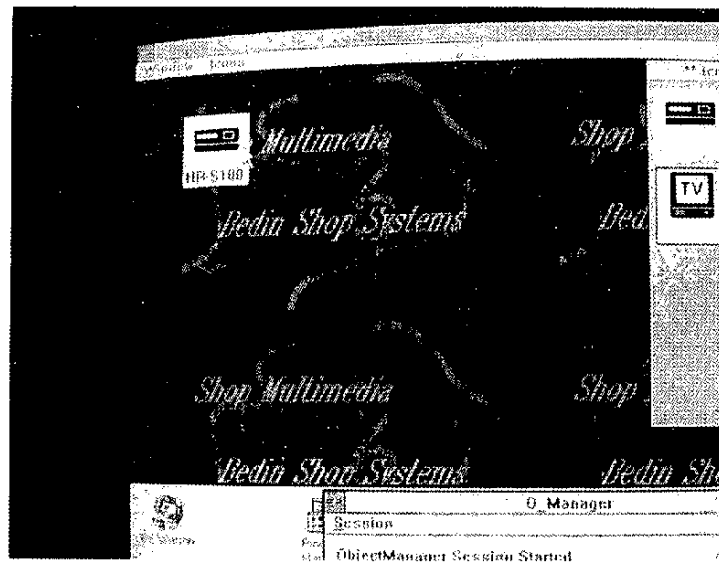


図3.5-8 オブジェクトアイコンの表示

選択された製品についての詳細仕様はSend\_Message操作によって得ることができる。図3.5-9に例を示す。システム内部では、選ばれたオブジェクトは図に示されるようなオブジェクトを構成要素として持っていることが認識されていることになる。同様に別のメッセージを送ると図3.5-10のような製品外観を表示させることもできる。

これらの機能に加えて、システムでは部品情報の管理もできるようになっている。具体的には図3.5-11に示すように、製品（例ではVTRではなく冷蔵庫）を構成している部品の表示が可能である。なお、個々の部品は独立した部品（オブジェクト）として管理されているため、ある部品を選択して画面上で移動させることはもちろん、部品についての問い合わせ（例えば部品番号や在庫状況）も可能である。

一方、データ定義/変更操作も同様なインタフェースを通して行える。図3.5-12、13に例を示す。これらは、あるVTRオブジェクトの変更操作の過程を示している。図中、画面の右上に現われているSelective propertyウィンドウには、選択されているオブジェクトに加えることができる項目が列挙さ

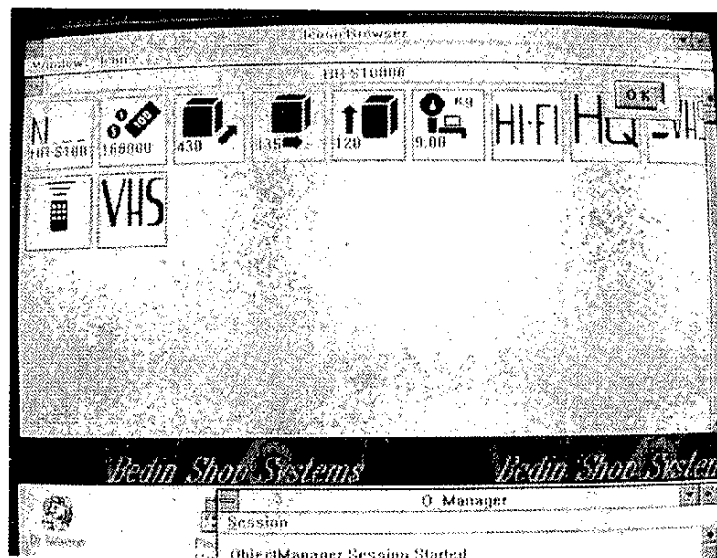


図3.5-9 製品仕様の表示

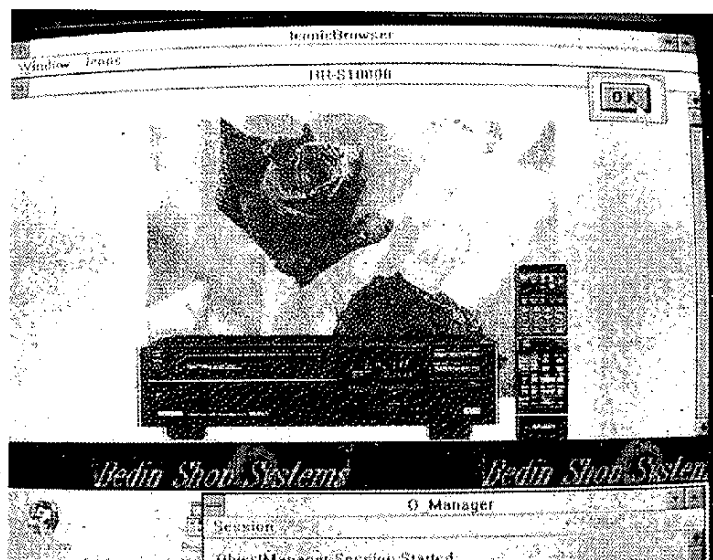


図3.5-10 製品外観の表示

れている。すなわち、Show\_Additional\_Property操作の実行結果が表示される。画面下方にある Registered property ウィンドウには、既に登録済みの項目が表示されるようになっている。システム 内部操作との対応でいえば、Show\_Propertyの実行結果が表示されるわけである。なお、定義操作の 開始/終了などを指示するためのアイコンが画面左側に用意されている。

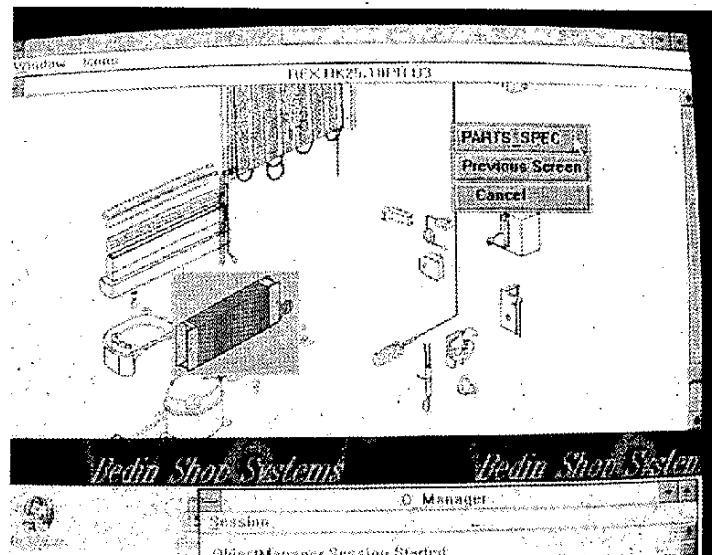


図3.5-11 部品情報の管理と表示

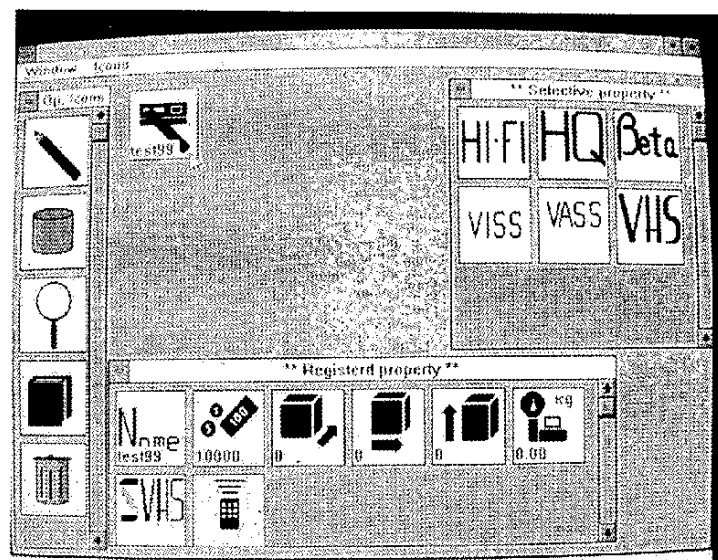


図3.5-12 データ定義/変更ツール

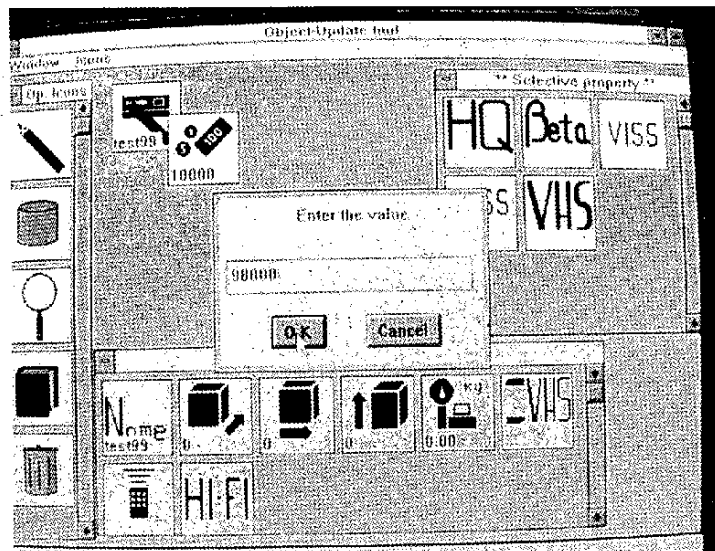


図3.5-13 値の変更

例えば図3.5-12においては、対象とするVTRは価格が100,000円でSVHS機能が備わっていることなどが定義されている。一方、製品の大きさは現在のところ具体的な値は設定されていない。データの定義/変更操作は、Registered propertyウィンドウとShow\_Additional\_Propertyウィンドウの間で、希望する属性アイコンの移動という形で視覚化される。属性値の定義/変更にあたっては、図3.5-13に示すようなウィンドウが表示される。図は値段を98,000円に変更しているところである。

ところで、ある属性項目を追加する際に必ず別の属性項目とペアで記述されたり、逆にある項目どうしは互いに排他的にのみ定義可能であったりということがある。例えばVTR製品に関していえば、SVHS機能が用意さえしていれば必ずHi-Fiであるし、VHS方式と $\beta$ 方式は同時には提供され得ない。このような制約をあらかじめシステムに記述し持たせておくことによって、ユーザのデータ定義/変更支援が図られている。

### 3.5.5 システム構成

システムはMODEと呼ばれるオブジェクトオリエンテッドデータベースシステム上に実現されている。このシステムは、db-VISTAと呼ばれるネットワーク型のデータベース管理システムを核として、その上にオブジェクトオリエンテッドデータベース管理機能が実現されたものである。IconicBrowserならびに定義システムはMODE上のアプリケーションとして構築されている。実現に係わる詳細については文献[Tsuda90]、[Yoshitaka93]を参照されたい。

なおシステムはIBM PC/AT互換機上でMS-WindowsならびにC言語を用いて開発されている。

### 【参考文献】

- [Czejdo89]Czejdo, B. et al.: A Visual Query Language for an ER Data Model, Proceedings of Workshop on Visual Languages, IEEE, pp.165-170 (1989).
- [Epstein91]Epstein, R. G.: The TableTalk Query Language, Journal of Visual Languages and Computing, 2, 2, pp.115-141 (1991).
- [Kwak93]Kwak, J. C. et al.: Object Query Diagram: An Extended Query Graph for Object-Oriented Databases, Proceedings of Symposium on Visual Languages, IEEE, pp.44-48 (1993).
- [Mohan93]Mohan, L. et al.: A Visual Query Language for Graphical Interaction With Schema-Intensive Databases, Transactions on Knowledge and Data Engineering, IEEE, 5, 5, pp.843-858 (1993).
- [Staes91]Staes, F. et al.: A Graphical Query Language for Object Oriented Databases, Proceedings of Workshop on Visual Languages, IEEE, pp.205-210 (1991).
- [Tsuda90]Tsuda, K. et al.: IconicBrowser: An Iconic Retrieval System for Object-Oriented Databases, Journal of Visual Languages and Computing, 1, 1, pp.59-76 (1990).
- [Tsuda91]Tsuda, K. et al.: MORE: An Object-Oriented Data Model with a Facility for Changing Object Structures, Transactions on Knowledge and Data Engineering, IEEE, 3, 4, pp.444-460 (1991).
- [Yoshitaka93]Yoshitaka, A. et al.: Iconic Object Definition System for Object-Oriented Databases, Proceedings of Symposium on Visual Languages, IEEE, pp.169-177 (1993).
- [Zloof75]Zloof, M. M.: Query-By-Example, Proceedings of National Computer Conference, IFIPS, pp.431-438 (1975).

### 3.6 ビジュアル・プログラミング・システムPP

並列論理型言語GHC[Ueda85]は、論理型言語Prologに、コミットオペレータを導入し、Prologに並列実行の機能を持たせた言語である。GHCの意味論や高速な処理系については各種の研究がなされ、言語の持つ並列性から、今後の並列プログラミング言語としての役割が期待されるが、論理型プログラミング言語としての性格からプログラムソースの可読性は必ずしも高くない。ここでは、並列論理型言語GHCをビジュアル化のターゲットとして行った、ビジュアル入力/実行環境の研究事例について述べる。

#### 3.6.1 既存のビジュアルシステムの問題点

通常、視覚化の試みは、既存のテキストベースのプログラムを視覚化する試みや、既存プログラムの実行を視覚化する試みから始まるが、やがて、既存の言語を視覚化するだけでは物足りなくなり、図形がテキストから独立し、図形やアイコンだけで直接実行やデバッグができるようにした新たなビジュアル言語の提案へと結び付いていくのが常である。

このようなビジュアル言語の試みは魅力的であるが、こうしたビジュアル言語は、実用化という観点からは問題点が多い。それは、以下のような理由からである。

- ・ 既存言語と比べ、機能が不十分、基本機能しかない

一般にビジュアル言語のシステムは試験的に作られることが多く、機能も基本機能しかサポートされない。ユーザの数も少なく、使えるマシンも限定されるので、やがてすたれてしまう。プログラミング言語の世界とは組み込み関数の数や、UNIX環境への親和性、ユーザの数、作られた既存ソフトウェアツールの数など、いわば総合力で勝敗の決まる世界である。したがって、ビジュアル言語のシステムをテキストベースの世界から切り離して作るより、既存のテキストベースの処理系に寄生した方が現実的である。

- ・ 図形はかさばり見づらい

図形で書くとどうしてもかさばって量が多くなる。これは、通常、「普通の書物より漫画の方が分かりやすいがページ数が多くなる」のと同じ原理である。

- ・ 入力の大変さ

図形やアイコンを組合せてプログラムを作るというのは手間がかかる。図形やアイコンには、その抽象度によって様々なレベルがあり、それらをうまく結合させてプログラムを作る必要がある。しかしながら、図形やアイコンの結合力の弱さから、プリミティブなアイコンを結合さ

せて高度のアイコンを作ることが難しく、その結果図形やアイコンの入力に手間がかかることになる。

- ・オモチャのシステムしかできない

これは、図形やアイコンの結合力の弱さに由来するものであるが、アイコンベースの既存のビジュアル言語のシステムを見ると、アイコンには最初からかなり高レベルのものを想定している場合が多い。これではデモプログラムは作れても、実用的なシステムは作れない。

これらの理由から、スタンドアローン型の「ビジュアル言語によるアプローチ」には問題が多いと思われる。そこで、並列論理型言語のビジュアルシステムの研究の手法としては、図形とテキストを「併用」しながらテキストのビジュアル化を模索する現実的な方向を選択した。

### 3.6.2 並列論理型言語の可読性と図形的表現

並列論理型言語のプログラムソースの可読性が低い原因を並列論理型言語の図形的表現という観点から捉えなおしてみると以下のようにまとめられる。

- ・ゴールの表示

各プログラム節のゴールは、ボディ部、ガード部ともにゴール列として1次元的に表示され、それらの間の相互関係や重要性を表現できない。そこで、これらのゴール列を2次元的に図形的表現で配置することができれば、ユーザは各プログラム節の意味を、ゴール列の2次元的な位置関係として、よりヒューマンフレンドリに把握することができる。

- ・引数、入出力モードの表示

それぞれのゴールは、ゴール間で共有される引数を通じて、情報のやりとりを行うが、ゴール間の引数の共有関係は、通常「変数名」を共有することにより示される。また、それぞれのゴールでは、引数は通常入出力のどちらかとして使われるが、並列論理型言語ではそれを陽には記述しない。そこでユーザは各プログラム節を理解する場合、引数の入出力のモードを推定することが必要となる。

したがって、ゴールの引数については、ゴールの引数を図形的に表現し、引数間の共有関係を、あるゴールと他のゴールを結ぶ直線として表現すれば、入出力関係や共有関係を極めて簡潔に表現することができる。

- ・統合的環境

並列論理型言語では、それぞれのプログラム節は独立に定義され、プログラムは構造を持たない。こうした言語の特徴は、ユーザがプログラムを見ながら、プログラムを理解することを困難にしている。この問題に対する一つの解決方法は、プログラムにモジュール構造を導入することであるが、並列論理型言語プログラムには、いまだよいモジュール化の方法が見出されていない。そこで、もう一つの方法として、統合的なプログラミング環境を用いる方法を考える。統合的な環境によって、例えば、ある述語についてすべてのプログラム節の定義を表示したり、さらにプログラム節に表れるサブゴールの定義をたどったりすることができれば便利である。

### 3.6.3 並列論理型言語のビジュアルシステムの機能

並列論理型言語の可読性に関する分析をふまえ、期待されるべき並列論理型言語のビジュアルシステムの機能を整理すると、以下のようになる。

- ・図形による入力

ユーザは図形の組合せの形で、並列論理型言語のプログラム節を入力できる。

- ・プログラムコードの自動作成

図形表現の形式でプログラム節を入力すると、それに応じて、テキスト表現が自動的に作成され表示される。

- ・図形表現の自動作成

逆にテキスト表現を入力し、図形表現を自動的に作成することもできる。この場合、ガード部やボディ部におけるゴールの配置などは、あるアルゴリズムに従って自動的に定められる。

- ・修正機能

テキスト表現や図形表現は、それを後で修正できる。この機能を用い、自動作成された図形表現を修正し、例えばゴールの位置を移動してユーザのイメージどおりの配置にすることができる。また既に定義したプログラム節の引数の数を変更したり、ガード部やボディ部のゴールを変更したりすることができる。

- ・保存・再生機能

テキスト表現と図形表現は、それをファイルに保存し必要ときに再生できる。図形表現につ

---

## 3. ビジュアルインタフェースの事例

いては、引数、ゴールの位置を記憶する。

- ・プログラムの実行機能

入力されたプログラムコードや図形表現は、それを図形的に実行することができる。

- ・統合的環境

本システムは、並列論理型言語プログラムの作成、修正、実行を支援するような統合的なプログラミング環境として構成する。一つのプログラム節だけでなく、いくつかのプログラム節を同時に扱うことができ、述語定義の途中に、他のサブゴールの定義や修正を行ったり、また、ゴールを実行したりすることもできる。

### 3.6.4 並列論理型言語のビジュアルシステムPP

並列論理型言語のビジュアルシステムとして、既にPP[田中92,Tanaka94]と呼ばれる並列論理型言語のビジュアルな入力/実行環境を実現している（PPは"Program-less Programming"の意味である）。本システムは、汎用のUNIXワークステーション上で通常のSICStus Prolog、またGUIとしてはSICStus PrologのグラフィックパッケージであるGM（Graphic Manager）を用いている。PPはX-Window上で稼働し、UNIX環境との親和性もよい。

PPでは、静的な図形表現と動的な図形表現のギャップがほとんどなく、双方とも同様な技法を用いて可視化を行っている。通常のプログラミング言語においては、プログラムの可視化のために、可視化のためのプリミティブやスクリプトなどをソーステキストに埋め込むなどの低レベルの操作が必要とされる場合が多い。しかしながら、PPではこれらの低レベルの操作をまったく行う必要がない。

### 3.6.5 ビジュアルシステムPPの動作イメージ

ビジュアルシステムPPを起動すると、ディスプレイ上にPPという名前を持ったメニューが現れる（図3.6-1参照）。このメニューにはExecute、Define、Load、Haltの四つのボタンが付いている。

Executeボタンは、プログラムの入力終了した後で、プログラムを実行、デバッグするのに使われる。Defineボタンは、最初にプログラム節を入力するのに使用される。Loadボタンは、あらかじめ作成して内部データベースに格納してあったプログラムを述語単位で取り出すために使われる。また、Haltボタンは、最後にシステムを終了させるために用いられる。

#### (1) 図形による入力

図形による入力の例として、appendプログラムを図形的に入力する場合を考える。まずメニュー

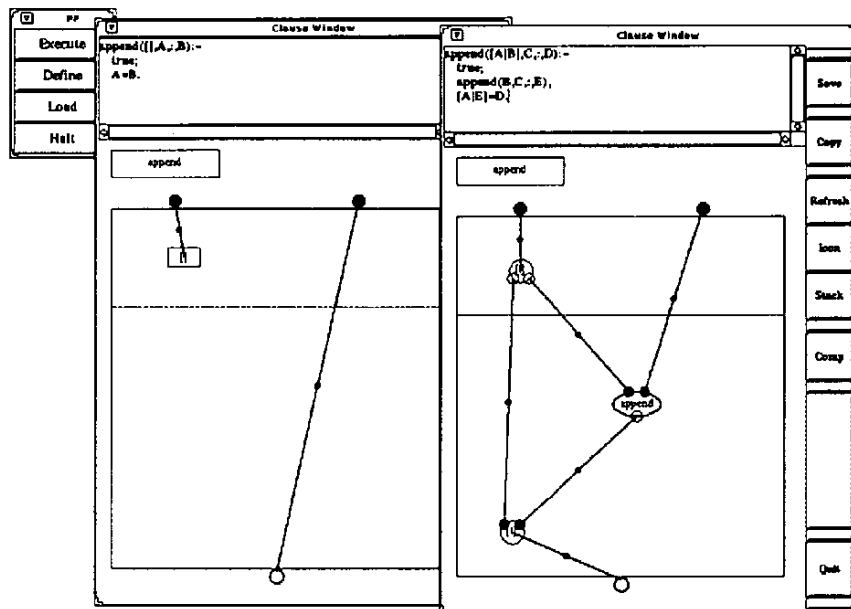


図3.6-1 図形によるプログラムの入力例

のDefineボタンを押すと、Clause Windowが一つ開く。一つのClause Windowは、二つのサブウィンドウからなっており、上方の1/5くらいを占めるEmacs Windowがプログラム節のテキスト表現を表示するためのプログラムビューウィンドウである。また残りの部分がプログラム節の図形表現を表示するためのグラフィックビューウィンドウである。

ユーザはこのグラフィックビューウィンドウから図形の形でプログラム節を入力できる。図形の入力では、マウスによりカーソルの位置さえ合わせれば、その場所に入力されるべき図形は決定される場合が多い。PPではこのような情報を使用することにより、入力の際のマウスのクリック数を最小化することを図っている。

append述語は、二つのプログラム節からなるので、図3.6-1では、Clause Windowが二つ開かれ、二つのプログラム節の定義が示されている。いずれも、グラフィックビューウィンドウから図形の形でプログラム節を入力するとプログラムビューウィンドウには、対応するテキスト表現が自動合成されて表示される。

いつ図形表現からコードへの変換を行うかというタイミングであるが、グラフィックビューウィンドウに変化があると、常に変換のトリガーがかかり、再計算が行われ、プログラムビューウィンドウに、常に図形表現に対応する最新のコードが示されるようになっている。

一般には、図形表現に対応するコードは、一意には決まらない。例えば、図3.6-1では、append述語の一つめの定義節のコードとして

```
append([A:B],C,_,D):-  
  true;  
  append(B,C,_,E),  
  [A:E]=D.
```

を生成しているが、

```
append(A,B,_,C):-  
  A=[D:E];  
  append(E,B,_,F),  
  [D:F]=C.
```

というコードを生成してもよい。

そこでPPでは、ガードのユニフィケーションがヘッドで表現可能な場合には、常にヘッドに代入を行うようにしている。また、ガード部やボディ部の述語が空のときにも、「true」をテキスト表現に生成するようにしている。

## (2) 入出力のモード

一般に、GHCプログラムでは、ゴールの引数の入出力のモードを完全に決定することはできない。通常、ユーザはゴールの引数について、ある意図をもってプログラムを書くのが普通であり、図形表示においても、ゴールの引数の入出力のモードがあったほうが、より理解が容易になると考えられる。そこで、本システムでは、ゴールの引数の入出力モードについては、「ユーザが自由に指定できる付加情報」として定義した。すなわち、述語やゴールの引数の入出力モードを、テキスト表現では、引数を必ず入力、出力の順に書くことにして、その間にはダミーの引数をはさみ、例えば、三引数のappend述語で最初の二引数が入力、第三引数が出力であれば、append([A:X],Y,\_,Z)などと表現するようにした。

ここでは、述語やゴールの引数の入出力モードはプログラムの図的表示のための付加情報であり、(プログラムの図的表示を伴わない) プログラム実行では無視される点に留意されたい。すなわち、ユーザは自分の思い入れどおりに入出力を指定でき、そのモードはプログラムの実行結果には、何ら影響を及ぼさない。

### (3) 変数名の扱い

一般に、プログラムビューでは、プログラム節の定義で変数名が使用されるが、グラフィックビューでは、変数名という概念は現れなかった。しかしながら、一般にはプログラムコードの変数名は、その変数についての思い入れを含め名称を付ける場合もある。例えば、リストとして使われる変数をListと命名したり、重リストとして使われる対の変数をHead、Tailと命名したりする場合である。したがって本システムでは、実は、二種類の変数、すなわち無名変数と名前付き変数の二つを用意した。

無名変数では、システムが自動的に変数名を管理し、変数名は保存されない。また名前付き変数では、システムが変数名を保存する。無名変数は、変数名が大文字だけ、もしくは大文字と数字だけからなる。変数名に小文字を含む変数は名前付き変数として扱われる。

ユーザがグラフィックビューから図形入力をした場合、変数名の指定を行わなければ、プログラムビューに表示されるプログラム節の変数にはA、B、C、...といった無名変数が自動的に割り当てられる。

また、グラフィックビューでは、同じ名前付き変数を重複して二箇所以上で使うことができ、その場合プログラムビューでは、その変数名に数字が自動的に付加されて区別される。名前付き変数を用いた述語定義の例を図3.6-2に示す。

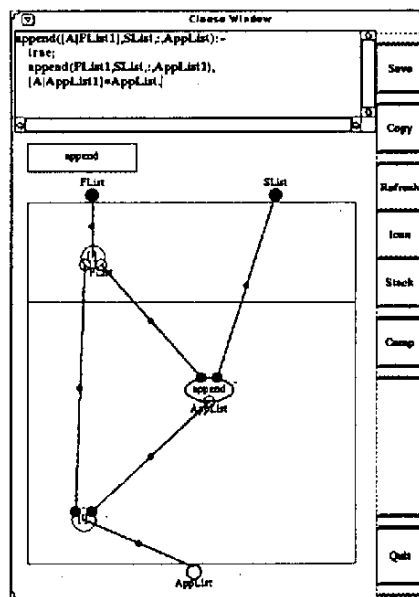


図3.6-2 名前付き変数を用いた述語定義の例

### (4) テキストによる入力

図3.6-3は、反対にテキストプログラムをプログラムビューウィンドウから入力したときの例を示し

ている。この場合、右側のClause Windowでは、demo\_appという定義節をテキストの形式で入力しており、グラフィックビューウィンドウには、テキスト表現から自動合成された図形表現が表示されている（PPでは、リストについてコンス構造で表示するのではなく、テキスト表現に対応するマクロな図形化を行っている。）。

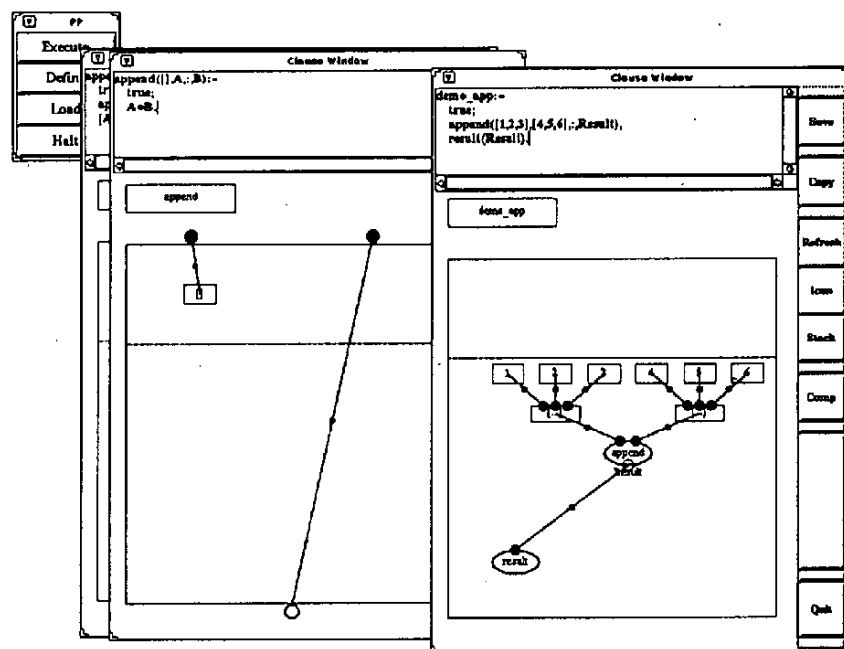


図3.6.3 プログラムビューウィンドウからの入力例

なお、いつコードから図形表現への変換を行うかというタイミングであるが、プログラムビューウィンドウで、「」,「,」,「:-」などのデリミタが入力されると、常に変換のトリガーがかかり、計算が行われる。その場合、入力されているデータがまだ不完全なものであっても補完が行われ、現に対応する最新の図形表現が示されるようになっている。append述語の一番目の定義節をプログラムビューウィンドウから入力していったときのグラフィックビューウィンドウの図形表現の変化を図3.6.4に示す。

PPでは、ゴールの図形表現をそのまま実行することができる。それにはメニューのExecuteボタンを押しExecution Windowを開き、そこに実行すべきゴールを図形的に入力してやればよい。このExecution Windowを用いることにより、ユーザは作成したゴールの動作を直接ビジュアルに確認することができる。また、Execution Windowは、ゴールを図形的に表示するだけでなく、実行の途中で表示された実行ゴールを図形的に編集し、実行ゴールの内容を動的に入れ換えて実行を継続することもできる。

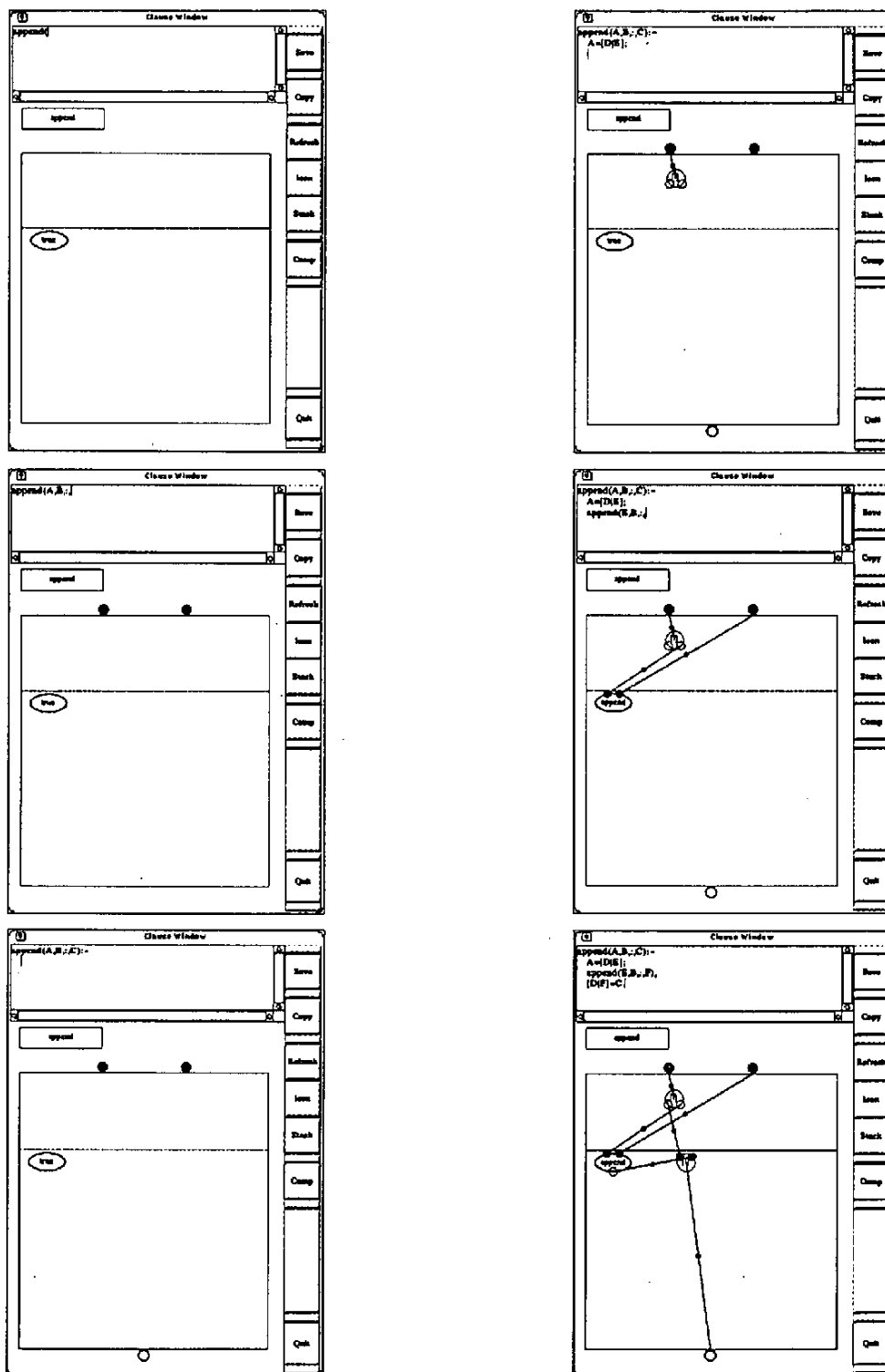


図3.64 テキスト入力による図形表現の変化

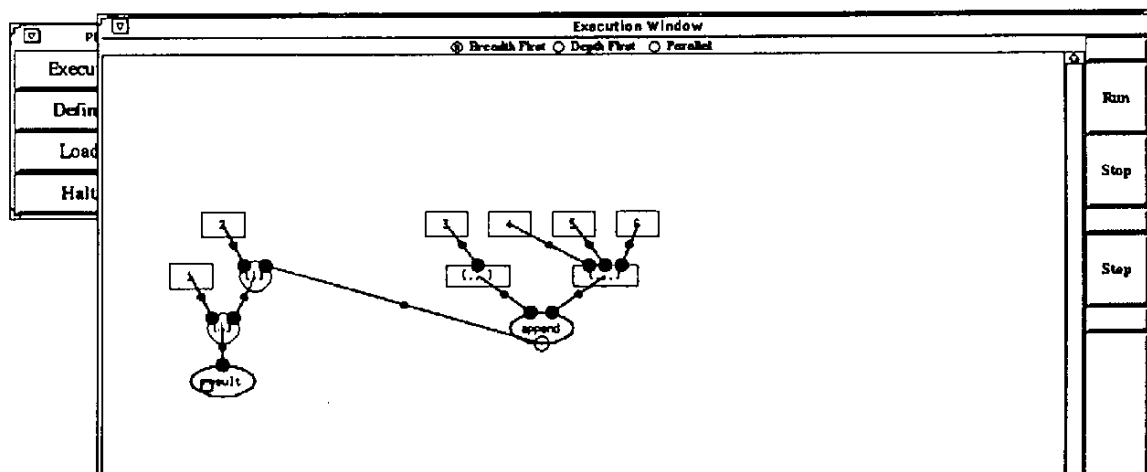


図3.6-5 プログラム実行のスナップショット(1)

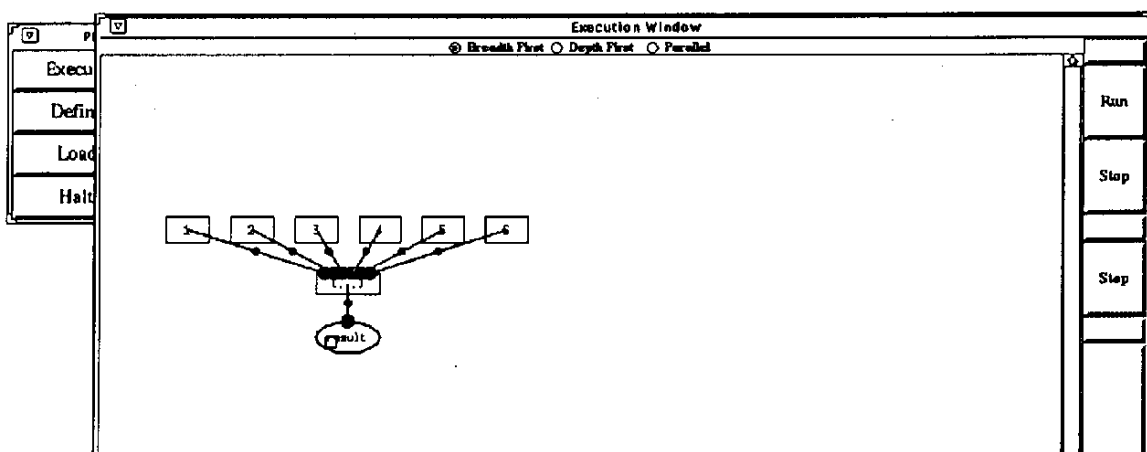


図3.6-6 プログラム実行のスナップショット(2)

図3.6-5および図3.6-6は、図3.6-3で定義したdemo\_appという述語を図形的に実行しているときの様子を示している。図3.6-5は、demo\_appがappendとresultゴールに展開され計算が進んでいく様子、図3.6-6はappendの計算が終了したときの様子を示している。ここで、result述語は「noop」に相当する述語で、リダクションされても変化しない述語である。

なお、PPでは、ゴールをステップ実行して図形的に表示するだけでなく、表示された実行ゴール

を図形的に編集して実行ゴールの内容を入れ変えて実行を継続することもできるようになっている。

### 3.6.6 関連研究の動向

Prologの図的表現については、プログラミング言語が提案された時代から、図的表現が既に使われていた[van Emden81]。Prologのビジュアル化した言語の提案にLograph[Cox85]があり、オブジェクト指向風の言語をターゲットにしたビジュアル言語にPrograph[Prograph89]がある。

また、並列論理型言語のような並列論理型言語を対象としたビジュアル化の研究には、Xerox PARCのKen KahnによってなされたPictorial Janus[Kahn92]がある。Pictorial Janusは、特にプログラム実行の可視化に研究の重点が置かれており、本システムで目指したようなプログラム入力の見視化や、図形とテキストの併用という観点からは研究を行っていないところに違いがある。

### 3.6.7 まとめ

視覚的インタフェースの実験的な対象として「並列論理型言語のビジュアルな入力/実行環境」を選び、並列論理型言語GHCを視覚化の対象としたビジュアル・プログラミング・システムについて述べた。

MVC (Model-Controller-View) パラダイムの観点から見た場合、PPは、モデルにGHCのプログラム節を格納し、それを二つのビュー、すなわち、プログラムビューとグラフィックビューの二つで表示するシステムであると考えることができる。それぞれのビューには、それぞれコントローラが組み込まれており、ユーザはこのプログラムビューとグラフィックビューのどちらからでも、GHCのプログラム節を入力することができる。

PPにおいては、プログラムビューとグラフィックビューは互いに因果結合されており、どちらのビューから入力しても、それらはすぐにモデルに反映され、それは直ちにもう一方のビューに反映される。PPが、ユニークに見える理由の一つは、このグラフィックビューとプログラムビューの因果関係を極限まで追求したところにある。すなわち、グラフィックビューに変化があると、プログラムビューの変換のトリガーがかかるが、その変換の粒度を図形入力の一動作に相当するように非常に細かく設定してある。図形入力の一動作ごとに変換のトリガーがかかり、再計算が行われる様子、また逆にプログラムビューからテキストを入力していくに従い、グラフィックビューで描画がされていく様子は非常にアトラクティブである。

今後、PPについてシステムの評価、改良を行っていく予定である。PPは、実行スピードなどでまだ問題があり、今後システムの最適化や描画の高速化などを計る必要がある。また、プログラムの実行に関しては、特に入力されたプログラムの実行機能やアニメーション機能を強化し、プログラムの実行がよりビジュアルに表示されることを目指していきたい。そのための方法として、実行中のゴー

---

## 3. ビジュアルインタフェースの事例

ル列の描画を、以前に実行されたゴールの履歴を反映して描画するような方式、ゴールの描画位置が変わるとき、それをアニメーション機能によってユーザに分かりやすく見せるための工夫について検討を行っていく。

#### 【参考文献】

- [Cox85]P.T. Cox and T. Pietrzykowski: LOGRAPH: A Graphical Logic Programming Language, Proc. of IEEE COMPINT 85, Montreal, pp. 145-151 (1985).
- [Kahn92]K.M. Kahn: Concurrent Constraint Programs to Parse and Animate Pictures of Concurrent Constraint Programs, Proc. FGCS'92, Tokyo, pp. 943-950 (1992).
- [Prograph89]The Gunakara Sun Systems: Prograph Tutorial, The Gunakara Sun Systems (1989).
- [Tanaka94]J. Tanaka: Visual Programming System for Parallel Logic Languages, The NSF/ICOT Workshop on Parallel Logic Programming and its Program Environments, the University of Oregon, March 4-6, pp.175-186 (1994).
- [田中92]田中二郎: ビジュアルプログラミングをめざして、ユーザーインタフェース大作戦(7), bit, Vol.24, No.7, pp.758-766 (1992).
- [Ueda85]K. Ueda: Guarded Horn Clauses, ICOT Technical Report TR-103, (1985).
- [van Emden81]M.H. van Emden: An Algorithm for Interpreting PROLOG Programs, Research Report CS-81-28, Department of Computer Science, University of Waterloo (1981).

### 3.7 視覚的仕様記述環境

#### 3.7.1 はじめに

情報システムとユーザとのインタフェースの了解性が重要であることは、改めて強調するまでもないことである。視覚化の試みに限っても、より良いインタフェースを求めて、数々の提案がなされ、また実際にシステム機能として提供されている。

しかしながら、現在の情報システム環境は、既に経済・社会活動の重要な基盤となっていることは疑いを容れない事実であるが、なお人間に親しみやすい存在であるとはいいがたい状況にある。さらに、広帯域サービス統合デジタル網などの通信処理技術を融合したこれからの情報システム環境を想定すれば、情報システムの新規開発の生産性およびこれに続く保守性の向上に加えて、インフラストラクチャとしてのサービスの連続性を維持すべく自己発展的な再利用を可能とすることも必要となる[Nishikawa90][西川94a]。

このような情報システム環境を構築・維持していくためには、人間と情報システム環境の最も高水準の直接的な接点である仕様記述水準での、了解性・再利用性の向上に関する再検討が非常に重要であることは言うまでもない。現在、筑波大学では、データ駆動パラダイム研究[Nishikawa87][寺田88][Nishikawa91][Terada92]の一環として、仕様記述から最下位の実行可能プログラム水準に至るまでのすべての水準をデータ駆動スキーマとして捉える統合的仕様記述環境の研究が行われている[西川93][西川94b]。

ここでは、このような環境を構想するに至った経緯と問題解決のための着想を紹介することを意図している。まず、これまで研究されてきた視覚的プログラミング環境[Shu88]をよりユーザフレンドリにするための課題について述べる。続いて、データ駆動スキーマを核とした仕様記述の再利用性向上のための一手法について述べる。

#### 3.7.2 視覚的プログラミング環境の現状と課題

よく指摘されるように、視覚的プログラミング環境とは何かという定義は、必ずしも確立しているとはいいがたい。しかしながら、「いわゆるソフトウェアのライフサイクルの過程で、意味のある図的表現を活用する環境」と捉えてもさしつかえないであろう。既に、図的な表現による視覚化の試みが、ソフトウェアライフサイクルのあらゆる側面に適用されているからである。

これらの試みは、視覚化の目標に基づいて、二つの大きな側面から捉えられるとされている。すなわち、次の二つである。

- ・ プログラム開発用のツールに視覚情報を導入し、その了解性の向上を目標とする視覚的支援
  - ・ 2次元の視覚的表記をソフトウェア設計やプログラム作成のために直接的に用いる視覚的言語
- 視覚化の対象に基づきより詳細にみれば、主として視覚的支援を目指すアプローチには、

---

#### 1. ビジュアルインタフェースについて

- ① データベースの定義・操作の支援を目指すデータの視覚化
- ② プログラムの構造や実行状況の了解性の向上のためのプログラム/実行の視覚化

さらには、

- ③ いわゆるソフトウェアライフサイクル上の要求分析・定義あるいは設計過程における構造化分析手法などを基礎としたソフトウェア設計の視覚化
- ④ 例示によるプログラミングを発展させた視覚的コーチング

などが含まれる。

また、視覚的言語はその視覚化に用いる図的表現形式の違いにより、

- ① データフロー図、制御フロー図、ブロック図などを用いたグラフ指向言語
- ② 表およびそれに準じた形式の表現方法を用いたフォーム指向言語
- ③ アイコニックインタフェースから発展したアイコン指向言語

に分類できる。

このように、歴史的には異なる動機から研究開発されてきた視覚化の試みは、次第に融合・統合化される方向に進んでいる。現在までに提案・開発されている具体的な各システムの詳細については、参考文献に譲る[西川89]。

### 3.7.3 よりユーザフレンドリな環境を目指して

先にも触れたように、視覚的プログラミング環境に関して共通の理解が得られているとはいいがたい現状にある。この分野が非常に新しいものであることも一つの原因と考えられるが、共通の理解は、この研究の進展のためだけでなく、潜在的なユーザに広く利用されるためにも重要となる。プログラミング環境の成熟には、いわゆる使い込みが大きな役割を果たすからである。

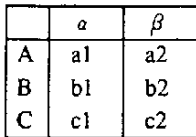
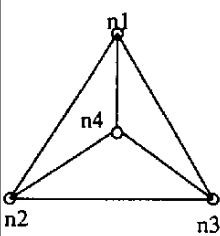
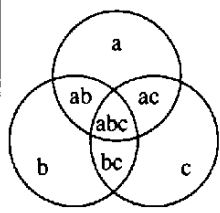
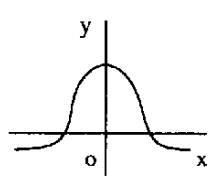
ここでは、視覚的プログラミング環境を、「いわゆるソフトウェアのライフサイクルの過程で、意味のある図的表現を活用する環境」と定義した。この延長線上に共通の理解を得るとしても、「意味のある図的表現」を明確にしなければならない。このためには、図的表現に関する基礎的研究とともに、単にシステム設計、プログラミングの分野にとどまらず、これまであまり関心が払われてこなかった他の分野での図的表現の活用法に目を向けるとともに、統合的に捉える積極的な姿勢が必要である。

#### (1) 意味のある図的表現とは

情報の（広い意味での）視覚的な表現形式は一般に、文、式、図、および、絵に大別される。文、式と図、絵の違いは明らかであるが、ともに2次元表現である図と絵についても、以下の知見が得られている[出原86]。

一般に言語と呼ばれるものは、狭義には日本語、英語などの自然言語のみを指すが、広義にはコンピュータ言語、数式、論理式などの人工言語も含められる。すなわち、有限個の要素とその配列規則（文法）によって無限個の文や式を生成可能で、かつこれらの数限りない文や式の意味が、要素の意味とその配列規則から導出できる場合、その表現体系は言語と呼ばれる。表3.7-1に示す表現形式を持つ図は、いわゆる絵と違って、これらの要件を満足する図的言語と捉えられている。

表3.7-1 図的言語の分類とその特徴

|      | 表現形式の一例                                                                             | 意味単位                           | 配列規則                               | 解 釈                                                    | 例                                                                |
|------|-------------------------------------------------------------------------------------|--------------------------------|------------------------------------|--------------------------------------------------------|------------------------------------------------------------------|
| 行列系  |    | 平面を横に分割する行線と縦に分割する列線。          | 行と列の交叉関係が意味を持つ。                    | 線が持つ平面分割機能を利用して、交叉する行項目と列項目との関係を表す。                    | 関係表、決定表、フォーム、時刻表、工程図、マトリックス図                                     |
| グラフ系 |   | アークと呼ぶ線分とノードと呼ぶ点(または図形や文字)。    | ノード間を連結するアークの有無が意味を持つ。             | 線が持つ連結機能を利用して、ノード間の関係を表す。                              | データフロー図、制御フロー図、ブロック図、組織図、工程図、地図、航路図、KJ図、連関図、系統図、過程決定計画図、アロダイアグラム |
| 領域系  |  | 任意の形状の閉曲線                      | 閉曲線相互の包含、交叉、分離、隣接関係が意味を持つ。         | 閉曲線が持つ平面分割機能を利用して、部分平面間の関係を表す。                         | ベン図、地図、三面図、KJ図、親和図                                               |
| 座標系  |  | 座標軸を示す軸線、位置を示す点、数量や方向や運動を表す線分。 | 座標軸が示す場(フィールド)における、点や線の大小関係が意味を持つ。 | 点を持つ位置決め機能と、線が持つ量表示、方向指示、運動軌跡表示などの機能を利用して、位置や量を数量的に表す。 | 棒グラフ、円グラフ、地図、天気図などの分布図、三面図、航路図、列車ダイヤ図                            |

一方、絵の場合は、それを構成する要素の種類は不定であって、個々の絵に用いられる要素は、無数といってもよいほど多様性があり、その解釈は極めて主観的となるため、形象系と呼ばれる。しかし、先に述べた図的言語に属さない図とともに、この形象系についても、たとえその解釈が主観的で一意的に定まらないとしても、形象としての意義を追究する姿勢もまた必要である。形式性をあまり

にも追究すれば、視覚的プログラミング環境といえども、再びユーザ指向でなくシステム指向の環境となってしまう危険性がある。エンドユーザプログラミングの次には、より自由な個性化が可能な記述環境が求められると予想されるからである。

## (2) 図的表現の積極的な活用のために

これまでに研究されてきた視覚化のための図的言語では、いわゆるグラフ系、行列系といった、コンピュータシステムの分野で慣れ親しまれてきた表記法への偏りがみられる。

視覚化の本来の目的の達成には、図的表現のさらに積極的な活用が必要であることは明らかである。すなわち、システムの性質を包括的に捉えられる領域系やシステムの定量的あるいは動的な挙動を陽に示す座標系などに属する表現形式も含めた統合的な環境の構築が必要である。要求定義から設計、プログラミング、保守・運用を統合的に一貫して支援できるシステムがまだ構築されていないのはその証左である。

システムの分野にとらわれず、広い視野に立てば、図的な表現法を積極的に活用した図的な思考法に関する試みが盛んに行われている。例えば、KJ図による発想法などはその先駆的研究である。また、統合的品質管理 (TQC) の分野においても、親和図、連関図、系統図などの多様な表現形式が思考の手段に広く用いられている[二見86]。これらへの積極的な取り組みが、視覚的プログラミング環境の将来には必要である。

## (3) よりユーザ指向な環境の実現のために

先にふれた統合的な視覚的プログラミング環境の実現には、先に述べた図的表現に関する基礎的な研究および積極的な活用に加えて、よりユーザ指向にシステムの設計思想を転換しなければならない。すなわち、これまでのプログラミング環境は、視覚化のいかんを問わず、対象とする処理機械の実行形式に依存した手続き性などの本質をそのまま反映した仕様を持つ言語ないしは表現形式をユーザに強いてきたきらいがある。ユーザ指向のシステムにとって、このような傾向は明らかに好ましいことではない。エンドユーザ、プログラマを問わず、ユーザは一般に、問題解決の当初から、当面する問題を処理機械に依存したある特定の表現形式を用いて、手続き的に記述できる段階に達しているわけではないからである。したがって、真にユーザフレンドリな環境の実現には、先に述べた一般的なブロック図的な表記法、さらにはユーザ自身の定義をも含めて、複数以上の表現形式を用いた多面的記述を支援する、いわゆるマルチリンガルな環境が実現されるべきである。

この結論に従えば、視覚的プログラミング環境の定義に関してよく議論の対象になるマルチウィンドウ環境についても新たな捉え方が必要である。これまでウィンドウの利用は、視覚的プログラミング環境に直接的には関係ないとされてきた[Shu88]。その理由として、

① ウィンドウは必ずしもプログラミングに関する図的表現を含んでいないこと  
および、

② プログラミングは必ずしもマルチウィンドウを用いることに依存していないこと  
などが指摘されてきた。しかしながら、マルチリンガルな記述方式を導入して、対象とする問題を多様な側面から自然に表現するためには、単に平面的なウィンドウ環境にとどまらず、より自然な3次元的な空間の広がりを持つウィンドウシステムも将来の視覚的プログラミング環境をイメージするために重要な鍵となると考えるのが自然である。

#### 3.7.4 仕様記述の再利用性を向上させるために

図的データ駆動言語 (Diagrammatic Data-Driven Language: D3L) [西川83]を核にして、

- ① 従来から種々の専門領域で広く使用され定義・解釈が明確に確立している信号流れ図、状態遷移図などから副作用が生じないD3Lで記述された高度並列処理プログラムを自動的に生成する方式が可能であることを実証するとともに、
- ② この方法の拡張の可能性を検討した結果、定義・解釈が必ずしも厳密ではないが、ユーザの直感的理解を助け問題を定式化する上で重要な役割を果たす一般的なブロック図や一般的なシーケンス図からもデータ依存図式が誘導可能であるという見通しを得て、真にユーザフレンドリーな仕様記述環境を構築する努力を続けてきた[西川86,西川90]。

これらのデータ駆動パラダイムに関する研究の一環として、現在、ソフトウェアの再利用を促進する仕様記述環境の枠組みを確立するために、改変時の予期せぬ副作用の検出手法を検討している。以下では、この研究の基礎としている副作用を明示するデータ駆動スキーマの考え方を示した後、仕様記述から実行可能プログラムに至るまで一貫したデータ駆動スキーマと捉える統合的な仕様記述環境の枠組みとその自由な階層構造化による副作用検出の一手法を紹介する。

##### (1) 副作用を明示するデータ駆動スキーマ

再利用性を向上させる一手法であるソフトウェアの部品化を促進するには、入出力データ構造と発火規則に厳密なデータフロー原理を用いた方法が有効であることは周知である。しかしながら、関数的処理のみを対象としたデータフロー原理では、履歴依存性を含む、一般的な処理のすべてを包含できるわけではない。この研究ではさらに、データ駆動スキーマにおけるデータ依存性と処理発火規則を活用して、ソフトウェアの自由な改変を許すより広い意味での保守性・再利用性を向上させることを意図している。

データ駆動パラダイム研究の当初から、データ駆動原理の本質は、処理の関数性ではなく、トークンの生成・消費に関する連続性にあると着想し、履歴依存性を許すデータ駆動図式と副作用が回避で

きる実行規則[西川83]を提案しさらに、この考え方を基礎として、超高位図的仕様記述環境(Advanced Environment for Software Production; AESOP)のプロトタイプを構築して、履歴依存性を許すデータ駆動スキーマの有効性を実証してきた[西川90]。

従来の制御フロー型プログラムでは、履歴依存性の有無を問わず、文の間のデータの受け渡しに、すべて変数への代入文が用いられるため、その副作用の検出は非常に困難となっている。一方、この研究に採用したデータ駆動スキーマでは、本質的なデータ依存性の表現とデータ駆動型発火原理を持つモジュールの集合としてプログラムが構成されるため、本質的な履歴依存性が明示的になるとともに、トークンの生成と消費に関する限り、厳密な検証性が与えられている。したがって、このデータ駆動スキーマを基礎にした記述環境を構築すれば、その改変の際に副作用を生じる可能性のある処理・データ構造を、データ駆動解釈の適用によって、陽に明示できる可能性があり、再利用性の向上が期待される。

## (2) データ駆動スキーマの階層構造化による副作用検出手法

先に述べたようなデータ駆動スキーマの優位性の主張に対して、あくまで、「programming in the small」での優位性であり、保守性・再利用性が真に要求されている「programming in the large」においては意味をなさないという反論がある。よく知られているように、ソフトウェア危機が叫ばれて以来、構造化プログラミングに始まり構造化分析手法[Ross77]、最近では、視覚的プログラミング環境など様々な試みが提唱されてきたが、大規模ソフトウェア開発についてはその困難さが指摘されるにとどまっている。

事実、構造化プログラミングの提唱者であるE.W.Dijkstraは、「自身の能力の限界を認め、より高い信頼性を与えるよう」階層構造化の必要性を説いた[Dahl72]。しかしながら、その流れを継承した構造化分析手法は、その提唱者自身によってオブジェクト指向手法にとって代われようとしている。また、Fred Brooksは、ソフトウェア工学に関する論説のなかで、「ソフトウェア本体は、非常に複雑な構造を持ち、空間的には捉えることは不可能である」と大規模ソフトウェアの視覚化の困難さを指摘した[Shu88]。

大規模ソフトウェアの階層構造化は、データ駆動スキーマを含めて、結局恣意的なものであり、その了解性向上のための決定的な手段とはなりえないとの考え方に深く依拠している。

しかしながら、この考え方は大規模ソフトウェアの新規開発と開発者自身による階層構造化を念頭においていることに注意を要する。観点を变えて、ソフトウェアを仕様記述から実行可能プログラムに至る一貫した階層的データ駆動スキーマとして組織化し、これを、分散協調型ないし改変を含めた再利用型に開発することに主眼に置けば、以下のような新たな捉え方も可能と考えられる。

すなわち、ソフトウェアの仕様記述環境の枠組みを、最上位に外部仕様としてのデータ駆動スキーマ

マ、最下位に内部仕様としての実行可能なデータ駆動プログラムを位置付けた任意の階層構造を持つ統合的なデータ駆動スキーマとして捉える。プログラムの正しさは、それ自体独立に議論できるものではなく、あくまである定められた仕様に対して検証されるべきものであるからである。また、階層構造が本来的に恣意的であるという事実は、最下位の実行可能なデータ駆動プログラムに影響を与えない限り、任意の階層構造化が可能であることを意味する。したがって、外部仕様にてらして、最下位のデータ駆動プログラムを解釈し、目的に応じた階層構造化を仕様記述環境の機能として提供すれば、副作用の明示を含めて、非常に高い了解性と再利用性を維持できる可能性がある。同時に、このような枠組みを採用すれば、外部仕様に対応した実行可能プログラムのプロトタイピングが可能となり、仕様記述水準での設計の誤りの検出にも効果的である。さらに、構造化分析・設計手法の効用も最大限に活用されるであろう。この手法の限界が、ソフトウェア開発の上流工程にのみ適用可能とされ、最終的には因習的なプログラミング言語への変換を余儀なくされていたことにあると考えるからである。

### 3.7.5 おわりに

空間的な広がりを持つ視覚情報の利用は、より自然なマンマシンインタフェース、さらにはマシンを意識する必要のない理想的なインタフェースの実現には、欠くことのできない要素技術となると予想される。

しかしながら、冒頭に述べたように、インフラストラクチャとしての情報システム環境を構築・維持していくためには、視覚的仕様記述環境だけで十分なわけでもない。

## 【参考文献】

- [Dahl72]Dahl,O.J.et al: Structured Programming, Academic press(1972) (野下浩平他訳:構造化プログラミング,サイエンス社 (1975)) .
- [二見86]二見良治: TQC に役立つ図形思考法, 日科技連出版社(1986).
- [出原86]出原,吉田,渥美: 図の体系-図的思考とその表現, 日科技連出版社(1986).
- [Nishikawa87]Nishikawa,H.et al.: Architecture of a one-chip data-driven processor: Q-p, ICCP'87,IEEE, pp.319-326 (1987).
- [Nishikawa90]Nishikawa,H.et al.: Advanced software technologies implementing intelligent and user-friendly services-Trends of recent research activities-, ICCS'90, IEEE, pp.11.2.1-11.2.5 (1990).
- [Nishikawa91]Nishikawa,H.et al.: Architecture of a VLSI-oriented Data-Driven processor: The Q-v1, Advanced topics in data-flow computing, Chapter9, pp.247-264 Printice-Hall, Inc (1991).

- [西川83]西川博昭他: 履歴依存性を許すデータ駆動図式, 電子通信学会論文誌(D), Vol.J66-D, No.10, pp.1169-1176 (1983).
- [西川86]西川博昭他: 超高位図的言語処理システムの設計思想, 第33回情報処理学会全国大会予稿集 4F-9 (1986).
- [西川89]西川博昭他: 視覚的プログラミング環境, 情報処理, 30, 4, pp.354-362 (1989).
- [西川90]西川博昭他: 超高位図的仕様記述環境 (AESOP) の構想, 情報処理学会コンピュータアーキテクチャ研究会報告, 90-ARC-83-2, pp.7-12 (1990).
- [西川93]西川博昭: 超並列処理記述のためのデータ駆動スキーマと副作用を明示する解釈に関する研究, 文部省重点領域研究「超並列原理による情報処理基本体系」 第3回シンポジウム予稿集, pp.80-81 (1993).
- [西川94a]西川博昭: 超分散システムに向けて, 電子情報通信学会誌, Vol.77, No.4 (1994).
- [西川94b]西川博昭: "データ駆動スキーマにおける副作用検出手法", 文部省重点領域研究「超並列原理による情報処理基本体系」 第4回シンポジウム予稿集 (1994).
- [Ross77]Ross, D.T.: Structured Analysis(SA): A Language for Communicating Ideas, IEEE Trans. on Software Engineering, Vol.SE-3, No.1, pp16-34 (1977).
- [Shu88]Shu, N.C.: Visual Programming, Van Nostrand Reinhold Company Inc.(1988) (西川博昭訳: ビジュアル・プログラミング, 日経BP社(1991)).
- [Terada92]Terada, H. et al.: VLSI design of a one-chip data-driven processor: Q-v1, Data Flow Computing: Theory and Practice, Chapter 17, p.423-440, Ablex Publishing Corporation (1992).
- [寺田88]寺田浩昭他: VLSI向きデータ駆動プロセッサ: Q-x, 電子情報通信学会論文誌(D), Vol.J71-D, No.8, pp.1383-1390 (1988).



————— 禁 無 断 転 載 —————

平 成 6 年 3 月 発 行

発行所 財団法人 日本情報処理開発協会  
東京都港区芝公園3丁目5番8号  
機 械 振 興 会 館 内  
TEL (03)3432-9384

印刷所 有限会社 盛光印刷所  
東京都千代田区飯田橋4丁目6番3号  
宝第3ビル  
TEL (03)3264-1851









