—論文—

高岡短期大学紀要　　第6巻　　平成7年3月
Bull.Takaoka National College, Vol.6, March 1995

コンピュータ操作環境におけるスクリプト言語の意義

藤田徹也

（平成6年10月31日受理）

要　旨

本稿では，今日のパーソナルコンピュータの操作環境におけるスクリプト言語導入の意義，その実際，および将来的な可能性について述べる。

現在のＧＵＩ上のアプリケーションの操作を主とした操作環境に，スクリプト言語による総合的な制御の機構を導入することによって，利用者は，必要に応じて直接操作と論理的な操作を選択してデータを処理することが可能になる。また，ＯＳおよび各アプリケーションの機能単位を組み合わせた操作環境を構成し，利用することができる。

キーワード

スクリプト言語，ＧＵＩ，直接操作，アプリケーション機能単位，
アプリケーション・オブジェクト

1　はじめに

1970年代後半に誕生したパーソナルコンピュータは，ソフトウェアの機能向上と，ハードウェアの低価格化・高性能化を背景として急速な進化を遂げた。現在のパーソナルコンピュータでは，100MIPS相当のCPU，数百ピクセル平方のカラーCRT，数十MBのメモリ・数百MBのハードディスク，グラフィカル・ユーザ・インターフェース（以下，GUI）を採用したオペレーティング・システム（以下，ＯＳ），およびその上で動作するアプリケーション群等による構成が一般的なものになりつつあり，今なお性能の向上は続いている。

現在のパーソナルコンピュータの利用は，アプリケーションの操作が中心となっている。文書作成，表計算，データベースなどの用途に応じたアプリケーションが次々と誕生し，これらのアプリケーションは広く普及している。

しかし，各アプリケーションは，利用者のニーズに応える形でそれぞれ独自にバージョンアップを繰り返すこととなった。各アプリケーションは高機能化，多機能化したが，その一方で，アプリケーションの肥大化や複雑化などの弊害が指摘されるようになり，利用者にとって望ましい操作環境の実現に十分に対応できているとは必ずしもいえない状況が現出している。

近年，パーソナルコンピュータ用に，ＯＳおよびアプリケーションによる処理を総合的に制御する機構として，スクリプト言語が発表された。スクリプト言語の導入は，現在の操作環境が持っている問題点の解決への効果

開放センター

的なアプローチとして注目されている。さらに，利用者による操作環境のカスタマイズの有力な手段を提供することによって、スクリプト言語の導入は，パーソナルコンピュータの操作環境の変革の契機となり得る可能性をも持っている。

本稿では，まず，パーソナルコンピュータの操作環境の変遷を述べおよび現状での問題点を明らかにし（第2章），その上で，利用者の視点から見たスクリプト言語の導入の意義（第3章），その実際（第4章），さらには将来的な可能性（第5章）について述べる。

2．コンピュータ操作環境の変遷

パーソナルコンピュータの利用形態は様々ではあるが，ワードプロセッサや表計算アプリケーション，描画アプリケーション等がとりわけ利用されることが多い。これらのアプリケーションの多くは直接操作（Direct Manipulation）の要素を取り入れることによって，コンピュータそのものの動作に関する知識の習得の必要性を最小限に抑え，行おうとしている仕事そのものに集中できる環境を提供することを可能にした。直接操作性は，シュナイダーマンによれば，以下のように定義できる[1]。

・操作しようとする対象を視覚的に表現する
・高速，逐次的，可逆的な操作
・指示と選択による操作
・操作の結果が即座に視覚化される

現在幅広く利用されているこれらのアプリケーションは，当初から現在のような洗練されているものではなかった。アプリケーションを中心としたパーソナルコンピュータの操作環境は，先駆的な研究の成果を取り入れることによって，また，利用者の要望を開発に反映することによって成長を続けてきたといえる。

本章では，パーソナルコンピュータの操作環境の変遷，および現在，GUI環境を採用するに至ったパーソナルコンピュータの操作環境の持つ現在の問題点について，直接操作性との関連を中心として述べる。

2.1 直接操作の源流

直接操作環境を初めて具体化したシステムが，当時MITの大学院生だったサザーランドのSketchPadシステムであった。それまでのコンピュータにおいては，利用者が求めるデータはあくまでもアプリケーションの出力の結果として得られ，利用者はコマンド操作を中心としたアプリケーションの操作に習熟する必要があった。満足する結果が得られない場合は，再び入力からやり直さなければならなかった（図1参照）。

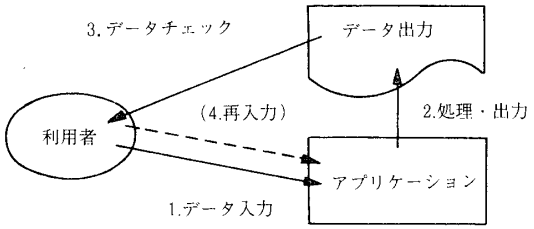


図1 従来のコンピュータの操作

SketchPadでは，ディスプレイ装置に図形を表示し，これらのディスプレイ上の図形をライトペンで直接指示することによって，変形・移動・複写などの操作を可能にした[2]。利用者は，SketchPadシステムを意識することなく，図形（データ）そのものを操作することが可能になり，その結果は即時に視覚化された（図2参照）。

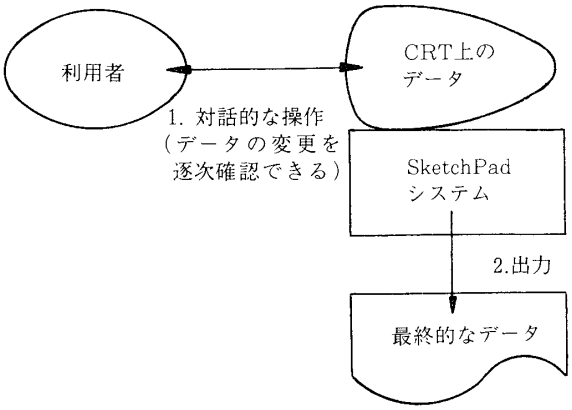


図2 SketchPadシステムの操作環境

SRI（スタンフォード研究所）のエンゲルバートが1968年に開発したNLS（oN Line System）は，複数の利用者（専門家集団）のネットワーク上での共同作業支援を目的としたシステムである。各自の端末インタフェース部分にはディスプレー装置がセットされ，各ディスプレー上のデータ（テキストと図形）はマウスによる指示によって操作することが可能であった。また，これらのデータとビデオ映像とを重ね合わせて，ディスプレイ内に同時に表示することができた。NLSも，Sketchpad同様に直接操作性を重視したシステムであり，1968年12月に行われたNLSのプレゼンテーションの成功は，Altoを初めとする後のパーソナルコンピュータの開発に大きな影響を与えることになった[3)4)]。

これらのシステムの特長を受け継ぎ，ビットマップディスプレーとポインティングデバイスを持ったシステムとして開発されたのがPARC（パロ・アルト研究所）のケイのAltoワークステーションである。Altoはまた，利用者として専門家だけではなく，コンピュータに関する専門の知識を持たない一般の人を想定して開発された。ケイは，AltoとSmall Talk言語の組み合わせによる，さまざまなアプリケーションを例示している。これらのアプリケーションの多くは直接操作性に優れ，十代前半の子供が操作できるものもある。この点からAltoは最初のパーソナルコンピュータと呼ばれることになった[5)]。

これらの研究成果は，当時の最高の性能に近いコンピュータでのみ実現可能であったため，1970年代後半の初期のパーソナルコンピュータには直接影響を及ぼすには至らなかった。しかし，上記の各システムのコンセプトはその後のコンピュータの性能向上によって徐々に実現されていくことになる。

2.2 アプリケーションの普及

初期のパーソナルコンピュータは，グラフィックスに関しては，低解像度で数色程度の表示が可能な能力を持つにとどまり，描画・作図等の実際の利用に耐え得るものではなかった。一方，文字（キャラクタ）は数ビットのビット列として比較的高速に演算・表示することが可能であったため，これらのパーソナルコンピュータ上では，文字の操作による利用が中心となった。当初の利用環境はBASICなどのプログラミング言語によるものであり，アプリケーションはプログラミング言語を用いて利用者自らが作成する必要があったため，パーソナルコンピュータを使いこなすことのできる利用者は限られていた。しかし，直接操作性に優れたアプリケーションが登場し，これらのアプリケーションは利用者の支持を受け普及していった。これらのアプリケーションの代表例といえるのがワードプロセッサと表計算アプリケーションである。

ワードプロセッサは，プログラム編集用のスクリーン・エディタの機能にレイアウト機能を組み合わせることによって，ディスプレイ装置の画面上に実際の文書と同等のイメージを表示して，編集し，そのイメージを印刷することを可能にした。

表計算アプリケーションでは，利用者は，あたかも計算用紙に記入する要領でディスプレイ上に表示された「セル」と呼ばれる領域に数値を記入し，簡単な計算式を入力するだけで，求める結果を得ることができる。

この二つのアプリケーションに共通した主な操作上の特長は，

a）データのイメージを，そのままの形でディスプレー装置に表示し，印刷可能であること
b）カーソルキー（矢印キー）によるポインティングが可能であり，操作の対象となるデータを自然な形で指定できること
c）メニュー選択方式による簡単な操作が可能であること

であり，これらの特長は，直接操作性に沿っ

たものであることがわかる。

1981年に発表されたMS-DOS[*1](Microsoft Disk Operating System)は，事実上のパーソナルコンピュータの標準OSとなった。アプリケーションがMS-DOSに対応することによって，複数のアプリケーションを容易に切り替えて利用できるようになった。また，文字コードやファイル形式が統一されたことによって，一つの文字データに対して,異なる複数のアプリケーションの連携が可能となり，例えば，データベースから特定の語句を抽出したデータをテキストファイルとして保存し，ワードプロセッサで加工する等の処理ができるようになった（図3参照)。

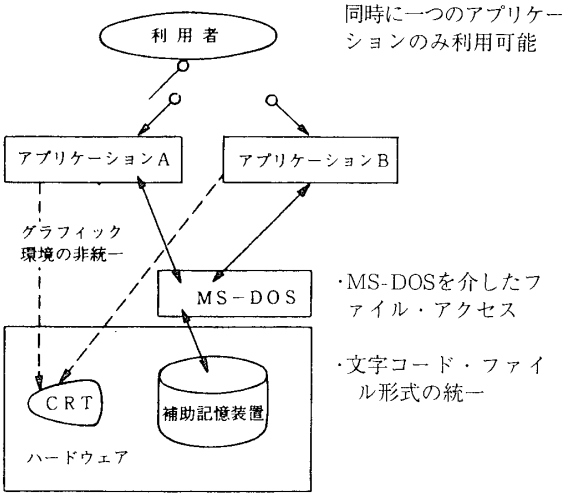


図3 MS－DOSの操作環境

2.3 GUIによる操作環境の改善

1980年代中期に入ると，CPU，ディスプレイ装置，補助記憶装置等のハードウェアの性能の向上および価格の低下によって，パーソナルコンピュータ上でも2．1節で示したような直接操作性に優れたシステムを実現できるようになった。このシステムの代表例として，MacintoshのMacOS[*2]や，MS-Windowsシステム等があり，これらはGUIに対応したOS（以下，GUIOS）と呼ばれる。

GUIOS上の操作は，基本的にはマウスによるメニュー選択式である。データはビットマップディスプレー上にマルチウィンドウ方式によって表示される。GUIOSでの操作環[*3]境の最大の特徴は、GUIOS上のアプリケーションが遵守すべきガイドラインが提示され，ウィンドウの様式やメニューの配置などのユーザー・インターフェースを全てのアプリケーションで統一したことである。これによって利用者は，複数のアプリケーションを類似した感覚で操作することが可能になり，新たなアプリケーションの操作の習得に必要な負荷を小さくすることができる。

また，データの様式は，文字データだけでなく図形データにおいても統一されているため，これらのデータは複数のアプリケーションで共通して利用できるようになった。GUIOS上で複数のアプリケーションが同時に動作している場合，アプリケーション間のデータ転送は，対象となるデータをマウスで指定して移動，複写を行う操作によって（「切り貼り」[Cut,Copy and Paste]と呼ばれる），ごく自然に行うことができるようになった。このことが，GUIOSを介した操作環境の大きな利点となっている（図4参照)。

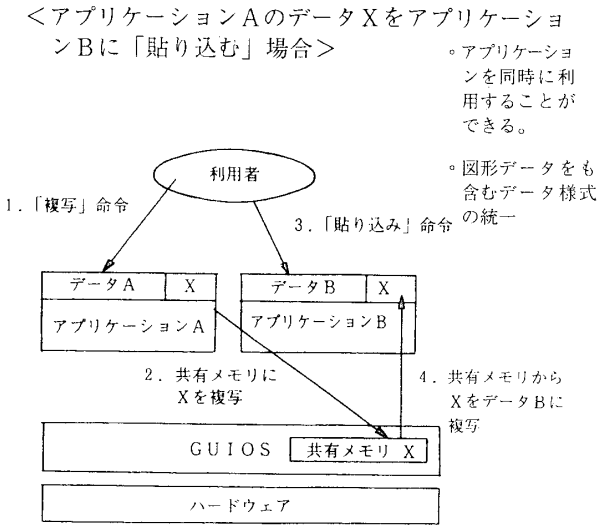


図4 GUIOSの操作環境

2.4 現在の課題

前節までに見てきたように，パーソナルコ

ンピュータの操作環境は，データに対する直接操作性を向上させることを目指して設計されてきたことがわかる。現在では，GUIOS上でのアプリケーションの利用（２．３節参照）が，パーソナルコンピュータの標準的な操作環境となっている。これは，直接操作性の重視が一般の利用者に広く受け入れられてきたことを示している。

現在では，仕事を行う際の有用な道具として，パーソナルコンピュータを利用することが日常化している。しかし，利用者が行いたい仕事すべてに対して，GUIOSとアプリケーションによる現在の操作環境が有効であるというわけではない。データの逐次的・直接的操作やアプリケーションによる操作が，利用者が求める仕事の方法に対応できない場合もある。

本節では，以下に，現在のパーソナルコンピュータの操作環境の持つ課題，および現状での対応について述べる。

(a) アプリケーションの問題点

2.2節で述べたとおり，パーソナルコンピュータ上のアプリケーションはGUIによる統合的な操作環境の普及以前に発達したものである。このため，個々のアプリケーションは，それぞれが独立した操作環境となることを目指し，利用者の要望に対応するためにバージョンアップを重ね，機能を追加していった。このような各アプリケーションの操作を中心とした操作環境は，以下の問題を持つことになった。

その一つは，データのアプリケーション依存性が過度に強くなることである。アプリケーションの多機能化に伴って，データはそれに対応した構造を取る。このようなデータは他のアプリケーションで使うことが難しくなる。その結果，いわば利用者がアプリケーションに束縛される状態となり，パーソナルコンピュータの利用は特定のアプリケーションの操作のみに限定されることになる。

もう一つは，各アプリケーションが独自にバージョンアップを繰り返した結果，アプリケーションが肥大化し，各アプリケーションの操作が複雑で難解なものになっている点である。さらに，各アプリケーションが，それぞれ違う操作方法で同じ機能をサポートしているため，利用者は，複数のアプリケーションを切り替えて操作する際に混乱を避けれらない。

(b) 直接操作の問題点

もともと，コンピュータは「論理的な記号操作を高速に実行する機械」として，その特質を発揮する機械であるのだが，パーソナルコンピュータはあえてその能力を全面には出さず，アプリケーションやGUIによって，直接操作による使いやすさを第一義的に考えた設計を目指してきた。しかし，実際の操作の場面で，同一操作の反復（例えば，データを数十回複写する）や操作の自動化（例えば，ある数値データを同じ手順で読み込んで，毎日の報告書を作成する）が必要な場面では，直接操作が煩雑なものとなり，コンピュータを利用しているにもかかわらず，コンピュータの特質を生かせないことになる。

(c) 現状での対応とその限界

(a)のアプリケーションに関する問題点については，各アプリケーションがGUIOS上で動作することによって，データの互換性や，操作の複雑さに関する課題を解決することが可能であるが、実際は、肥大化したアプリケーションが，そのままの形で（あるいは，むしろ拡張されて）GUIOS上で動作しており，「アプリケーションに束縛されている」という状況は改善されていないのが現状である。

(b)の直接操作に関する問題点については，一部のアプリケーションにおいて「マクロ」と呼ばれる簡易言語を組み込み，アプリケーションの実行を制御することによって操作の反復や自動化を可能にした。この簡易言語は表計算アプリケーション等を中心として利用

されたが，簡易言語による制御はアプリケーション内部のみに限られ，アプリケーションごとに簡易言語も異なっていた。

また，GUIOSにおいては，「発行と引用(Publish and Subscribe)」「オブジェクトリンクと埋め込み（Object Linking and Enbedding)」と呼ばれる機構をサポートし，他のアプリケーションやあるいはネットワーク上の他のデータの動的な共有を簡単なメニュー操作によって可能にした。（ここで，データの動的な共有とは，あるアプリケーションのデータのコピーを，他のアプリケーションのデータに埋め込むことが可能であり，データを変更すると、埋め込まれたデータのコピーも自動的に更新されることをいう)。データの動的な共有は，アプリケーション間でのデータの共有を半自動化できる有用な機構であるが，現状ではこの動的なデータ共有に対応しているアプリケーションは少ないこともあり，十分活用されていないのが実態である。

3．スクリプト言語の意義

1993年に，AppleScript[*4 6) 7)] や Visual Basic for Applications[*5 8)] などのスクリプト言語が，各GUIOSに対応して発表された。これらのスクプリト言語は、GUIOS上での操作環境を利用者がカスタマイズするプログラミング言語として設計されている。

本章では，スクリプト言語について，その機能，および制御の原理について述べ，スクリプト言語が，パーソナルコンピュータの操作環境にどのような役割を果たし得るのか，その意義について考察する。

3.1 スクリプト言語とは

3.1.1 スクリプト言語

スクリプト言語は，OS，およびアプリケーションによる処理を総合的に制御するプログラミング言語である。ここでの制御とは，OS，およびアプリケーションにおいて，例えば，「ファイルの印刷」や「データの移動」等の，メニュー選択で利用者が実行可能な命令（以下，機能単位と呼ぶ）のレベルでの制御のことである。OSおよびアプリケーションの機能単位を制御する点からは，各アプリケーションに対応した簡易言語（マクロ)，あるいはコマンド型OSのジョブ制御言語と同等の機能を持つものであると考えることもできるが，スクリプト言語は，OSおよび各アプリケーションの機能単位を必要に応じて自由に組み合わせることができるという特色を持っている。

スクリプト言語は，単独では基本的な制御構造を持つプログラミング言語に過ぎない。スクリプト言語がその能力を発揮するためには，以下の前提が必要となる。

(a) OSおよび各アプリケーションがスクリプト言語に対応し，スクリプト言語内から機能単位を利用できること

(b)(a) の機能単位を組み合わせて利用できるよう，複数のアプリケーションが同時に動作可能な環境（マルチタスク環境）であること

3.1.2 スクリプト言語によるアプリケーションの制御

GUIOSでは，マウスのボタンの押し下げなどの利用者の行動や，画面更新要求命令などOS自身からのメッセージはイベントと呼ばれ，イベントは発生順にイベントキューと呼ばれるFIFO(First-In First-Out)型の記憶領域に蓄積される。蓄積されたイベントは順次取り出され，その内容に応じて，アプリケーションやOSが処理を実行する。これは，イベント駆動型の動作と呼ばれている（図5参照)。

スクリプト言語は，独自のアプリケーション制御用のイベントを送信することができる。このイベントは，イベントキューから読み出された後，対応するアプリケーションに転送される。アプリケーション側では，このイベ

ントに対応したイベントハンドラ（特定のイベントが発生した場合に機能単位を実行するプログラム）が動作する。結果として，スクリプト言語から，アプリケーションの機能単位を実行できることになる（図6参照)。

アプリケーションは，利用者の行動，およびスクリプト言語の両方のイベントに対応したイベントハンドラを持つことによって，直接操作のインターフェースとしての，またスクリプト言語の「サブルーチン」としての役割を同時に果たすことができる(図7参照)[9]。

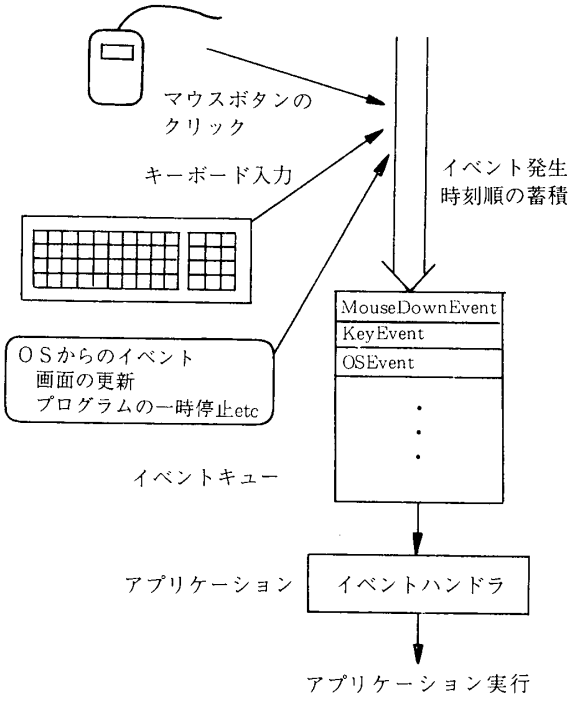


図5　イベント駆動型の動作

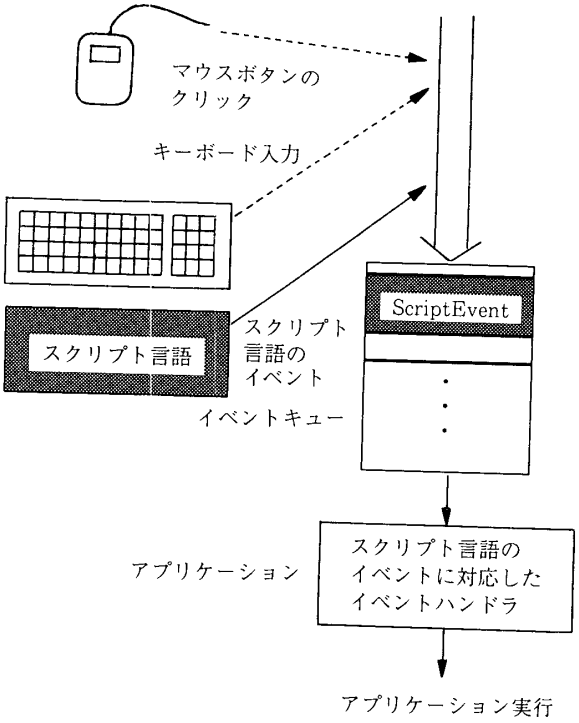


図6　スクリプト言語からのアプリケーションの利用

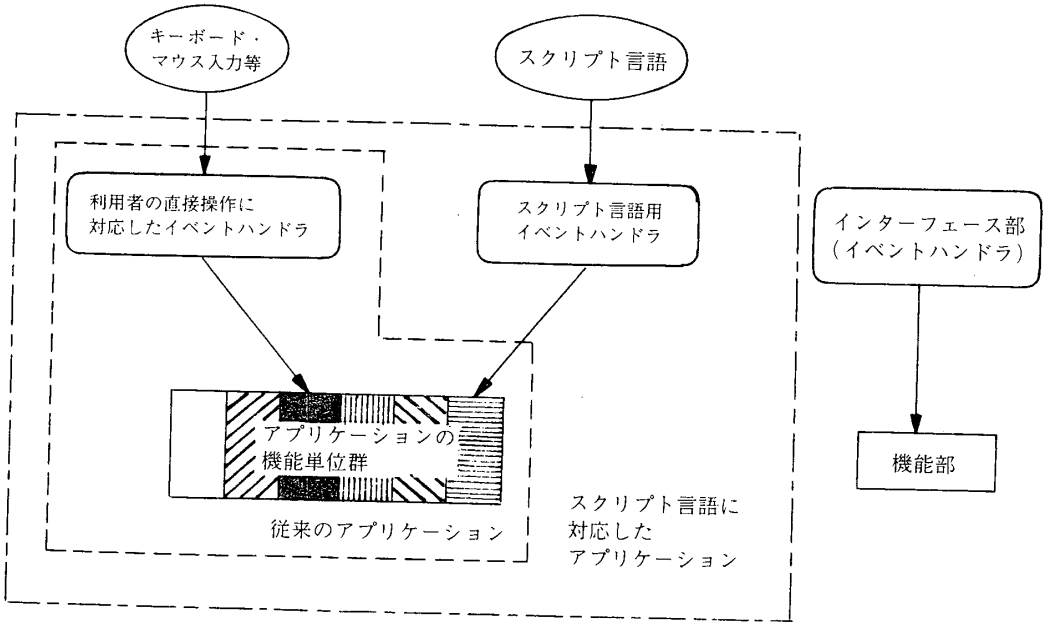


図7　スクリプト言語に対応したアプリケーション

3.1.3 アプリケーションのオブジェクトモデル

前節では，スクリプト言語からのアプリケーションの機能単位の利用を，サブルーチン呼び出しに例えたが，スクリプト言語によるアプリケーションの制御は，実際にはオブジェクト指向に基づいたものとなっている。すなわち，アプリケーションは，データと機能単位とを持つオブジェクトと見なされ，スクリプト言語からアプリケーションに対してメッセージ（イベント）を伝達することによって，アプリケーションの機能単位が実行される。

アプリケーション内のデータは，アプリケーションオブジェクトと呼ばれ，何らかのクラスに分類される。一般に，アプリケーションオブジェクト間には階層的な包含関係が成り立つ。文書の例でいえば，文書のクラスには，文書，段落，単語，文字などのクラスがあり，文書は段落をいくつか含み，段落は単語をいくつか含む（図8参照）。

これらのクラスのデータの参照は、アプリケーションクラスからの階層を指定することによって可能になる。文書の例では「word 2 of paragraph 5」「last paragraph of document "AAA"」（AppleScriptの場合）という表現によって，それぞれ，「5段落目の2番目の単語」，「文書"AAA"の最後の段落」のデータの参照が可能になる[10]。

3.2 スクリプト言語の意義

前節で述べたように，スクリプト言語からGUIOSとアプリケーションの機能単位を利用することが可能になると,以下のようなパーソナルコンピュータの操作環境を構築することが可能になる。

(a) 直接操作と論理的な操作を，必要に応じて組み合わせる。

例えば，DTPアプリケーションで大量の印刷物を作成する場合，文書や図表の作成は，ワードプロセッサや描画アプリケーションなどの直接操作性に優れたアプリケーションによって行い，それらのデータの割り付けはスクリプト言語によって自動化する。割り付けられたデータは，DTPアプリケーション上で，再び細部の修正を行うことができる。

(b) OS・アプリケーションの機能単位を，利用者が組み合わせる。

例えば，データベースアプリケーションにサーバのデータを参照させ，このデータをワードプロセッサで整形し出力するという操作をスクリプト言語によって自動化できる。また，

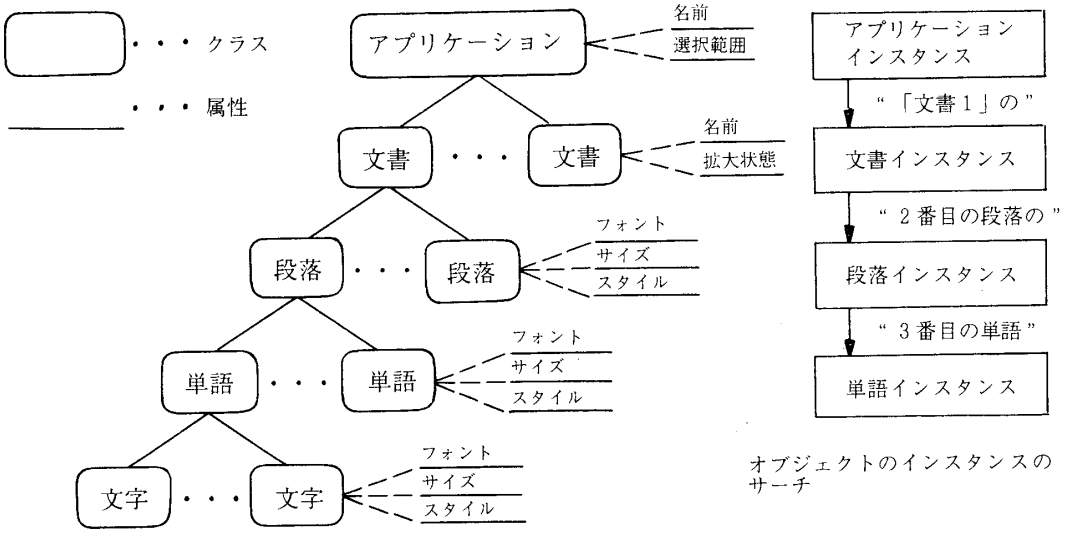


図8 アプリケーション・オブジェクトの階層構造（文献[10]による）

このような一連の操作を記述したスクリプト言語のプログラム（以下，スクリプト・プログラム）自身も，一つの機能単位として，他のスクリプト・プログラムから参照することができる。

スクリプト言語による上記(a)(b)の操作環境の導入は，特定のアプリケーションに対する依存性や，操作の反復・自動化に対する対応などの，現在の操作環境の持つ問題点（2．4節参照）を解決する有力な手段となり得ることがわかる。

これに対して，現在提供されている，スクリプト言語とは異なるいくつかの代表的なアプローチでは，上記(a)(b)の操作環境の実現は難しい。例えば，ウィンドウやボタンのようなGUIの構成要素を部品として提供し，その動作をプログラムとして記述する方法(Interface Builder[*6]など）では，スクリプト言語と同様に，オブジェクトの合成によって，汎用性のあるアプリケーションの生成が可能であるが，オブジェクトの組み合わせの際に高度なプログラミング能力が要求される。また，オブジェクト指向型開発環境を利用する方法（HyperTalk[*7]など）では，その言語が動作する範囲内で提供されるオブジェクトの制御は容易であるが，範囲外での制御は著しく限定されており，他のアプリケーションやシステムとの連携が難しい。

汎用的で，かつ容易なオブジェクトの合成によるプログラミングが可能になること。また，直接操作と論理的な操作の自由な組み合わせによる利用が可能になること。以上の2点がスクリプト言語導入の最も重要な意義であると考えられる[*8]。

4．スクリプトの記述の実際

本章では，ネットワーク上にあるアプリケーション資源をスクリプト言語を用いて記述し，ユーザ独自の新たな操作環境を作り上げる「スクリプト」の実際例を示す。具体的には，筆者が作成した「プレゼンテーション資料作成」のスクリプトの記述例を示す。そしてこのスクリプトの概要，機能，および操作環境の特徴について詳しく説明する。ここで示すスクリプトの例は説明のために機能を限定してある。実際のスクリプトはこれ以外のコンピュータ資源を縦横に使うことで，単一のアプリケーションでは不可能と思えるようなユーザ利用環境を構築することができる。

4.1 スクリプトの概要

本スクリプトは，素材となるデータを作成・編集し，プレゼンテーション資料を作成する過程を，スクリプト言語によって統合したものである。図9に，このシステムの構成を示す。本スクリプトは，コンピュータ1の制御部，アプリケーション，データ，およびコンピュータ2のイベントハンドラ，アプリケーションで構成されている。

制御部は作業用の領域を持ち，マウスのクリックでスクリプト・プログラムの実行を開始するボタンや，変数の値等を表示するテキスト・フィールドを配置することができる。スクリプト・プログラムは、随時編集することが可能である。

アプリケーションは，スクリプト言語からのイベントを受け，図形編集，プレゼンテーション資料作成等の機能単位を実行することができる[*9]。

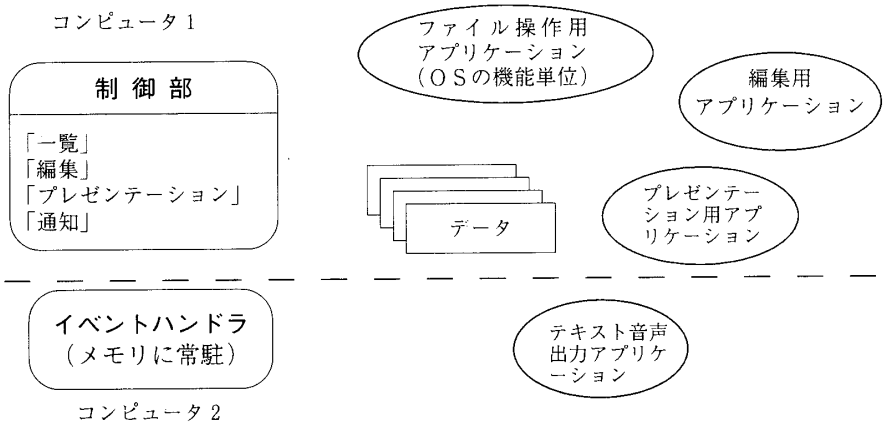
コンピュータ2のイベントハンドラは，アプリケーションとしてメモリに常駐し，制御部のスクリプト言語からイベントを受け取ると，コンピュータ2内のアプリケーションを制御する。

4.2 機能実行時の制御の流れ

今回のスクリプトでは，4種類の機能を設定した。以下に，各機能の実行時の制御の流れについて説明する。

(a) データ一覧の表示（「一覧」ボタン）

制御部のテキスト・フィールドに，編集の対象となるデータファイルの一覧を表示する。このとき，制御部は，ファイル操作用アプリケーション（OSの機能単位）にデータファイルの一覧を要求するイベントを送信し，戻り値としてデータファイルの一覧を得る（図10参照）。



コンピュータ1：Macintosh Quadra 700
OS：System 7.1J
メモリ：20MB HDD：180MB
制御部：HyperCard 2.2（Apple Computer）
AppleScript J1-1.1（Apple Computer）
ファイル操作用アプリケーション：
Finder Liaison 1.1（Gregory H. Dow作）
編集用アプリケーション：
Canvas 3.5.1J（Deneva Software）
プレゼンテーション用アプリケーション：
HyperCard 2.2(Apple Computer)
ネットワーク：Ethernet（10BaseT）

コンピュータ2：Macintosh Centris 660AV
OS：System 7.1J
メモリ：20MB HDD：250MB
イベントハンドラ：
AppleScript J1-1.1（Apple Computer）
テキスト音声出力アプリケーション：
Speech Lab, Speech Manager
（KURT Software, Apple Computer）

図9　プレゼンテーション用システムの構成

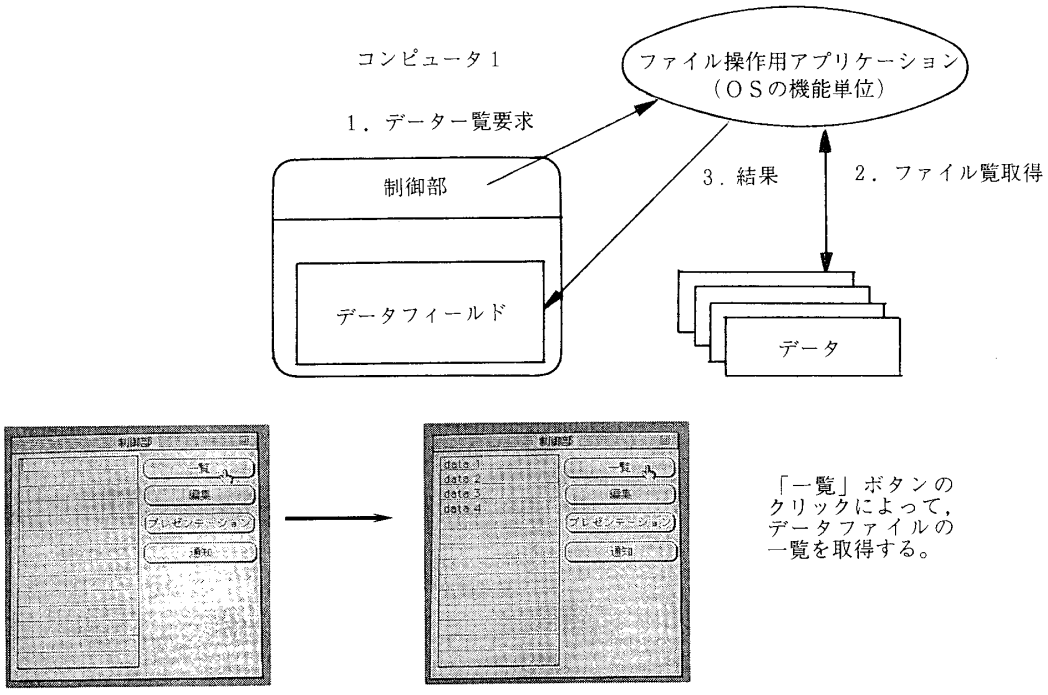
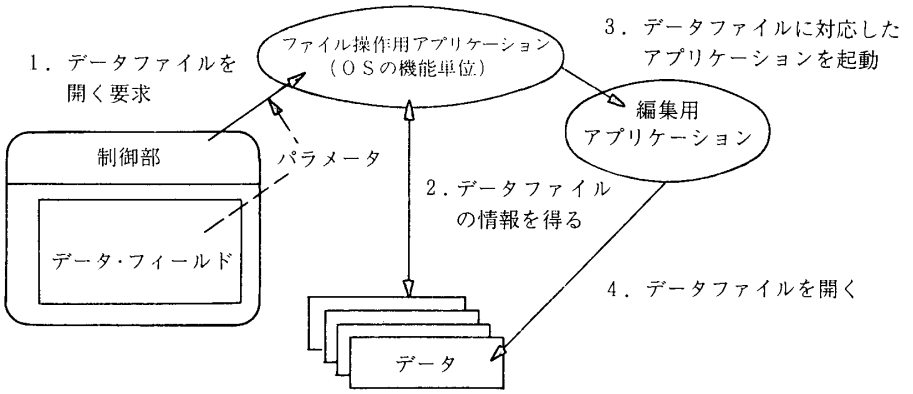


図10　データファイルの一覧

(b) データの編集（「編集」ボタン） (a)で得たファイルの一覧のうち，マウスで指定した特定のデータファイルを開き，編集可能な状態にする。このとき，制御部は，ファイル操作用アプリケーションに指定されたアプリケーションを開くイベントを送信し，OSを介してデータファイルに対応するアプリケーションが起動する（図11参照）。

(c) プレゼンテーション資料の作成（「プレゼンテーション」ボタン）

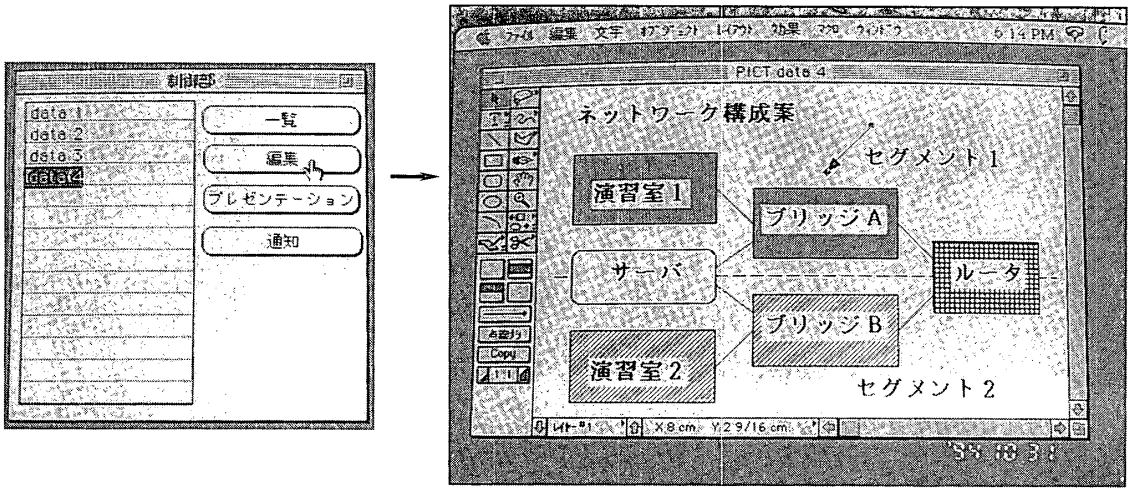
対象となるデータファイルをまとめて，プレゼンテーション資料を作成する。制御部は，プレゼンテーション用アプリケーションを起動するイベントを送信した後，テキスト・フィールドのデータファイルの一覧をパラメータとしてプレゼンテーション資料を作成するよ



＜制御部のスクリプト＞

```
on mouseUp
    set fileName to (contents of selection)
    tell, application "Finder Liaison 1.0"
        Open File fileName in Folder "example"
          in Folder "desktop Folder" in Disk "Cirrus 180"
    end tell
end mouseUp
```

・・・マウスボタンクリック時の動作
・・・ファイル名をデータフィールド内から選択
・・・ファイル操作用アプリケーションを対象とする
・・・指定したファイルを開くよう指示する



データファイルは，随時編集することができる。

図11　データファイルの編集

う指示する（図12参照）。

(d) ネットワーク上の他のコンピュータへの音声による通信（「通知」ボタン）

プレゼンテーションの準備ができたことを，コンピュータ２に音声によって通知する。このとき，制御部はコンピュータ２のイベントハンドラに，テキスト音声出力アプリケーションを起動するイベントを送信し，プレゼンテーションの準備ができたことを示す音声（"Presentation is now ready"）をコンピュータ２に出力するよう指示する（図13参照）。

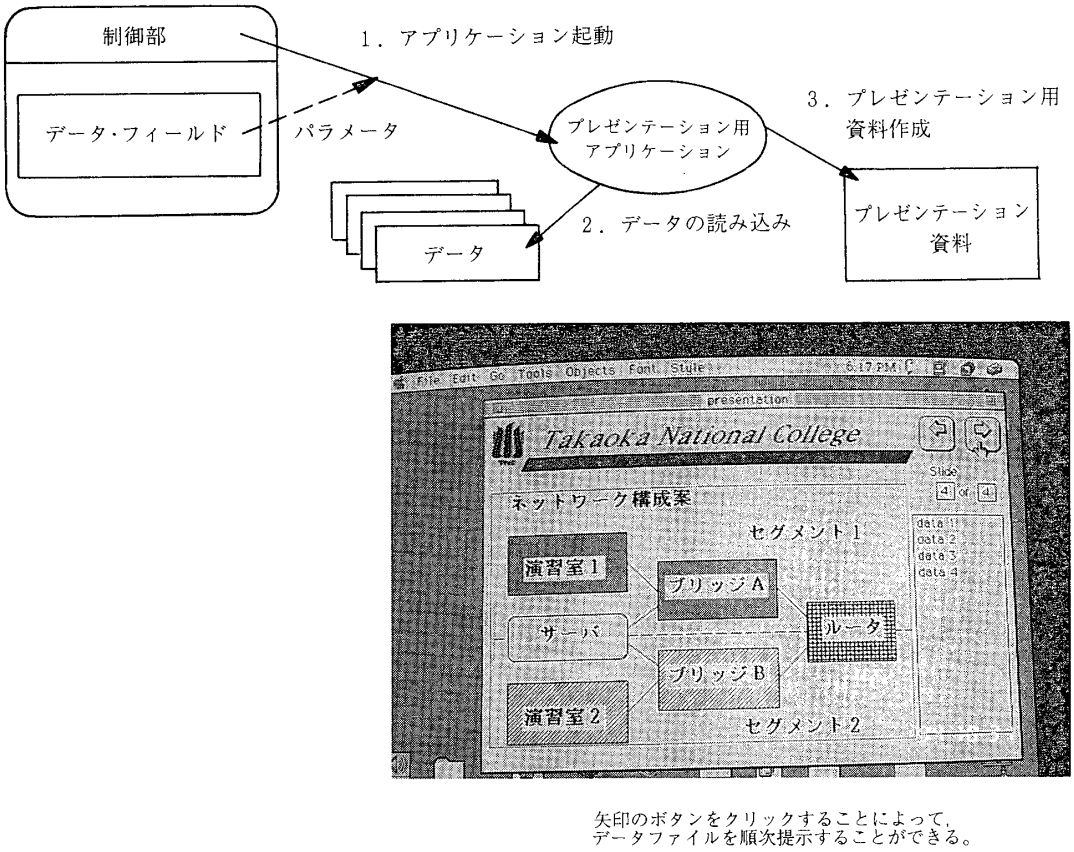


図12 プレゼンテーション用資料の作成

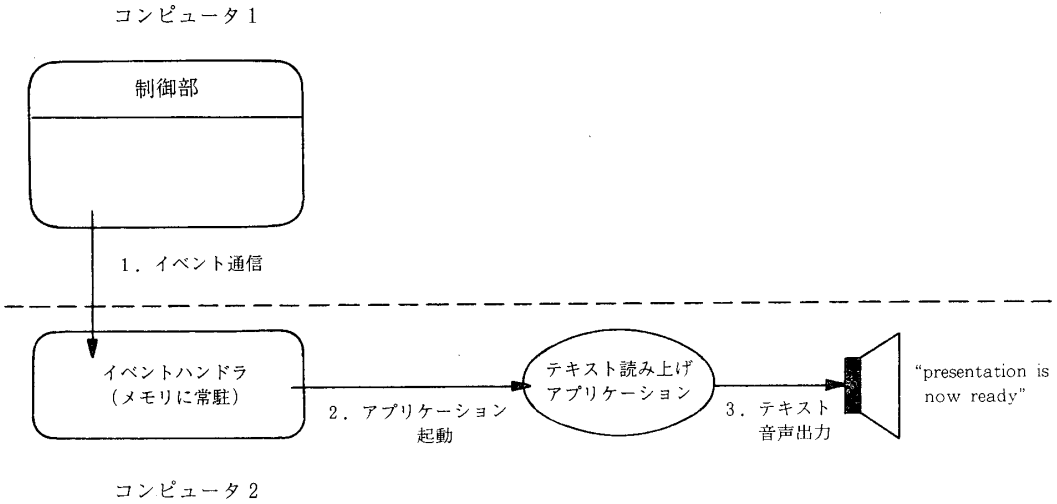


図13 他のコンピュータへの音声による通知

4.3 スクリプトによる操作環境

(a) このスクリプトでは，「データファイルの作成・編集」，「プレゼンテーション資料の作成」，「プレゼンテーション資料の利用」という一連の作業の流れのうち，「プレゼンテーション資料の作成」の過程は自動化され，他の過程は，直接操作性に優れたアプリケーションを，そのまま利用することができる。このため，利用者はデータの作成・編集や資料の提示などの本来必要な作業に集中することができる。このことは、3.2 (a) に述べた，利用者による直接操作と論理的操作の組み合わせが効果的であることを裏付けている。

(b) このスクリプトの各機能は，ＯＳの機能単位（「ファイル一覧の取得」「アプリケーションの起動」），および各アプリケーションの機能単位（「プレゼンテーション資料の作成」「テキスト音声出力」）の，組み合わせによって、利用者が実現することができる（3.2 (b) 参照）

(c) 前節 (d) の，制御部とイベントハンドラによる機構によって，利用者は，独自にネットワーク上の他のコンピュータへの通信チャンネルを設定できることになり，アプリケーション間通信によって，データの転送（イベントの送信）や，アプリケーションの遠隔制御（テキスト音声出力）などを容易に行うことができる。

5．スクリプト言語の可能性

第３章および第４章で見てきたように，スクリプト言語は，操作の自動化や複数のアプリケーションの連携が必要とされる場面で有効に機能することが明らかになった。今後，各アプリケーションおよびデータのオブジェクト化が進むことによって，これらのオブジェクトを総合的に操作できるスクリプト言語は，利用者の要求に応えることのできるツールとして大きな力を発揮できる可能性を持っている。本章では，スクリプト言語を介した操作環境の持つ可能性について述べる。

5.1 ドキュメント・ベースの操作環境

スクリプト言語は，他のアプリケーションの機能単位を利用することを前提としている。スクプリト・プログラムは，プログラム作成時点に存在しているアプリケーションを対象として記述されている。現状では，各アプリケーションが独自のデータ形式およびデータ参照形式を持っている。このため，スクリプト・プログラムは，アプリケーションへの依存性が強くなり，プログラムとしての汎用性が低くなる。

この問題は，アプリケーション間でデータ形式およびデータの参照形式を統一し，また基本的な機能単位を統一した上で，各アプリケーションがパーツとして固有な機能単位を提供するという環境の実現によって改善される。このときの操作環境は，アプリケーションの利用を主としたものではなく，データそのものを主ととしたものとなるであろう。この場合，さまざまなデータは「ドキュメント」とよばれる汎用の領域上に置かれ，データに対して，必要に応じてシステム共通のパーツ，あるいはアプリケーションのパーツを選択して利用する形態となる。このとき，スクリプト言語は，利用者がこれらの機能単位の実行を制御する手段としての役割を果たすことになる。[*10 11)]

5.2 連続系データとスクリプト言語

スクリプト言語によるプログラミングでは，操作の対象となるアプリケーション内のデータを参照する必要がある。文書や表計算の表のように，順番に並んでいるデータや，形式に従って配置されているデータは，容易に参照できる（3.1.3節参照）が，描画アプリケーションにおけるビットマップ図形のように，配置の形式を持たないデータの細部は，スク

リプトから参照することは難しい。結果として、オブジェクトとして「全ての図形」を対象とした操作のみが可能になる。もともと、これらの図形の編集作業は、プログラムによる制御では難しいからこそ、直接操作性に優れたアプリケーションが利用されてきたのである。

テキストや表などの離散系データと同様に、ビットマップ図形・動画・音声などの連続系データに対して、利用者の感覚に近い形でオブジェクトを指定する（例えば、人物を記録した動画データで、「左側の人」「動いている人」といった指定をする）ことは容易ではない。しかし、オブジェクトの指定がより自然なものになるほど、スクリプト言語によるプログラミングに対する抵抗は減少し，より多くの利用者がコンピュータが本来持っている「論理的な記号操作を高速に行う機械」としての能力を利用できるようになると考えられる。

6．おわりに

本稿では，パーソナルコンピュータの操作環境の変遷について述べ，ＧＵＩおよびアプリケーションによる操作環境にスクリプト言語を導入することによって，汎用的で，かつ容易なオブジェクトの合成によるプログラミングが可能になること，および，直接操作と論理的な操作の自由な組み合わせによる利用が可能になることを，プレゼンテーション用資料作成の実際例等を通じて明らかにした。

スクリプト言語が発表されて間もないこともあり，現段階ではスクリプト言語に対応しているアプリケーションは少なく，スクプリト言語を本格的に利用するのは難しいのが現状である。とはいえ，本稿で取り上げた，直接操作と論理的な操作との関わり，アプリケーションのパーツ化、あるいはオブジェクトの指定方法などの課題は今後のパーソナルコンピュータの操作環境を考える上で重要なポイントである。これらの課題をも踏まえて、今後も，利用者にとってより望ましいパーソナルコンピュータの操作環境について考えていきたい。

謝　辞

大阪大学経済学部の小郷直言助教授には，貴重なご意見と示唆を頂きました。厚くお礼を申し上げます。

注　釈

＊1　MS-DOSは，米Microsoft社の登録商標である。

＊2　Macintoshは米Apple Computer社のパーソナルコンピュータであり，MacOSは，MacintoshコンピュータのＯＳである。

＊3　MS-Windowsは，MS-DOS上で動作するGUIOSとして，米Microsoft社が開発した。

＊4　AppleScriptは，MacOS用のスクリプト言語として，米Apple Computer社が開発した。

＊5　Visual Basic for Applicationsは，MS-Windows用のスクリプト言語として，米Microsoft社が開発した。現在のところ，同社の表計算アプリケーション MS-Excel ver.5.0 上でのみ動作する。

＊6　Interface Builderは，米Next Computer社の開発したNextStep OS上で動作する開発環境である。

＊7　HyperTalkは，多機能カード型データベースHyperCard上で動作するオブジェクト指向型簡易言語として，米Apple Computer社が開発した。

＊8 北海道大学工学部の田中譲教授らによるIntelligentPadシステムは，オブジェクトの合成および直接操作の利用の面に関して，スクリプト言語によるものと同様の操作環境を実現している。

＊9 これらのアプリケーションのうち，プレゼンテーション用のアプリケーションは，スクリプト言語に対応したものがなかったため，プレゼンテーション資料作成の機能単位は，筆者が多機能カード型データベース(HyperCard 2.2)を利用して制作した。また，ＯＳの機能単位の利用は，ＯＳ制御用アプリケーションFinder Liaisonを介して，間接的に利用可能である。

＊10 このようなドキュメントを主体にした操作環境の構想は始まっている。現在のところ，OLE2.0(米Microsoft社)や，OpenDoc(OpenDocコンソーシアム)などが有力な候補として挙げられている（文献11参照）。また，近年，文書データの多角的利用と異機種間の文書交換を目的として制定され，国際規格となったSGML(Standard Generalized Markup Language)および，そのマルチメディア版のHTML(Hyper Text Markup Language)が広く注目されるようになった。

引用文献

1) B. Shneiderman:"Designing the User Interface 2nd edition", Addison-Wesley(1992)

2) I.E.Sutherland:"SKETCHPAD,A MAN-MACHINE GRAPHICAL COMMNUNICATION SYSTEM", Proceedings of the Spring Joint Computer Conference, 329-346(1963)

3) D.C.Engelbert:"The Augmented Knowledge Workshop",A History of Personal Workstations, ed. by Adele Goldberg, ACM PRESS, 187-232(1988)

4) W.English et al:"Display-Selection Techniques for Text Manipulation", IEEE Transactions on human factors in electronics, March 1967, 5-8(1967)

5) A.Kay and A.Goldberg:"Personal Dynamic Media",IEEE Computer, March 1977,31-41(1977)

6) D.Goodman:"The Complete AppleScript Handbook", Random House(1993)

7) D.Schneider:"The Tao of AppleScript",Hyden Books(1993)

8) 西田雅昭："ＶＢＡハンドブック",技術評論社(1994)

9) Apple Computer Inc.:"The Event Manager, The AppleEvent Manager",Inside Macintosh Volume IV Part 1,Addison-Wesley(1991)

10) アップルコンピュータ(株)："AppleScriptによるユーザプログラミング",Apple SE Workshop No.1,アップルコンピュータ（株）(1994)

11) J.Udell:"Componentware", BYTE, May 1994(1994)

The Significance of Script language in the Operating Environments of Personal Computers

Tetsuya FUJITA

(Received October 31,1994)

ABSTRACT

This paper expounds upon the significance of the introduction of script language in the operating environments of today's personal computers. The practical use, and possible future importance of script language are also discussed.

By introducing of script language for greater user customization and overall control, recently introduced applications on graphical user interface can provide the PC user with a better operating environment. One possibility allows the user to choose either direct manipulation or logical operations depending upon his needs. A second alternative is possible by combining the functional element of system software and application programs, a customized working environment can be created.

KEY WORDS

Script Language, Graphical User Interface, Direct Manipulation, Application Functional Element, Application Object