飛しょう体シミュレーション・ソフトウェア開発環境におけるユーザインターフェイスの研究^{*1}A Visual Design of User Interface for Software Engineers to
Develop Missile Simulation Programs古 瀬 慶 博^{*2}
Nobuhiro FURUSE

Key Words: Design, Visual Design, Missiles, Simulation, User Interface

Abstract

Visual design of the user interfaces is performed with the design principles of the graphical user interfaces (GUI). The design process of the GUI is consist of determining the functions based on user's requests and interpreting the design principles to implement the GUI by using a specific graphic widget, that is the OSF/Motif. The users focused on this paper are software engineers to develop missile simulation systems. Most primitive functions to make programming and testing the simulation programs are classified into two groups: one is associated with editing and compiling the source codes, and the other is associated with visualizing the simulation results. The design principles deduced from usability inspection are applied to decide hierarchy and layout of the menus of functions. Details of the layout should be discussed with the layout guidelines of the OSF/Motif widget. The author built the system with the GUI to develop the missile simulation programs, and shows an example of utilizing the system to develop a missile trajectory simulation program.

1. は じ め に

飛しょう体の開発における国産化の傾向により、設計から製品評価までの至るところでシミュレーションが行われている。より高性能で多機能な製品が要求されるに伴い、数学的シミュレーション（以下、シミュレーションと呼ぶ）は性能の事前確認を行うために不可欠な道具といえる。

従来のシミュレーションは、パラメータを変更し計算機へ投入するバッチ処理が主流であった。しかし、ワークステーションが普及するにつれて、その処理形

態はグラフィカルユーザインターフェイス (GUI) を備えたインタラクティブな方式へと移りつつある。小規模なシミュレーションならば、ワークステーションの演算能力で十分対応できるし、大規模な高速演算を必要とする場合には、主計算はスーパーコンピュータで行い、ワークステーションは入出力インターフェイスに専念させるような、それぞれの得失を考慮した運用が可能となった。開発の段階では、仕様の変更に伴う柔軟な設計が要求されるため、モデル及びパラメータの変更は円滑に行えるようにしたい。この種の作業は、シミュレーションに携わるソフトウェア開発者に対しても、迅速な対応を迫られる場合が多い。そうした切迫した状況の下に置かれると、ユーザビリティ（使いやすさ）に優れた環境を持ち合わせている場合とそうでない場合とでは、ソフトウェアを開発する者が抱く心理的な圧迫感は大きく異なる。

ユーザビリティを重視した開発環境を構築するためには、GUIの導入は欠かせない。導入にあたり留意すべきことは、オブジェクト（ここでいうオブジェクトとは画面に表示する部品の総称で、表示を行うときの最小単位である。例えばアイコン・ダイアログメニュー・スクロールバーといったものがオブジェクトに相当する）の標準化と画面のデザインの二点である。幸いオブジェクトの標準化は進んでおり、Motif¹⁾、Open Look²⁾、Macintosh³⁾、Windows⁴⁾、SAA CUA⁵⁾といったスタイルが既に確立している。これらのスタイルはグラフィックキット（またはウィジェット）として提供されており、系列関係にある機種間での移植性の高さには定評がある。一方、オブジェクトをどのように配置したら使いやすいか、思考を妨げない配色とはどのようなものか、といったことは画面のデザインの問題に属する。この問題はガイドラインを設定することで解決しようという動きがあり、これ

^{*1} 平成5年5月25日原稿受理^{*2} 三菱スペース・ソフトウェア(株)鎌倉事業所

まで多くの成果が報告されている⁶⁻¹⁰⁾

ガイドラインは設計を進める上で役に立つが、製造の現場では必ずしも十分活用されていないのが実情である。その理由として、第一に、予算や工期といった外的な制約が導入を妨げているように見受けられる。ユーザインターフェイス設計の理想的な進め方は、ガイドラインに基づく設計、プロトタイプの実装、ユーザによる評価というプロセスを繰り返し実行することにある。しかし現実には工期や予算を短縮するために、このプロセスの繰り返しを省略したり、実装を行うかわりに紙上のデザインで済ませることが多い。またユーザが既存のシステムに慣れている場合には、ユーザは新しいデザインの導入に対して保守的になることがある。そうでなくてもデザインの変更は操作性の変更も伴うので、ユーザの積極的な協力と時間の余裕がなければ新しいGUIの導入は成功しない。つまり、既存の使い勝手がそのまま新しい標準(デファクト・スタンダード)として採用されてしまうのである。

第二の理由として、ガイドラインはかなり整備が進んでいるものの、依然として実際の設計との間にはギャップが存在するという点があげられる。ガイドラインは、ユーザ中心のアプローチであるため、その内容が実装するグラフィックキットで容易に実現可能かどうかの判断は含まれていない。ガイドラインに基づいて操作性の優れたデザインを行っても、使用するグラフィックキットのスタイルから外れたものでは、実現に多大な労力を要する。例外的なスタイルを認めてしまえば、もはやグラフィックキットだけではGUIは実現できなくなるし、移植性に重大な影響を及ぼす逆に、スタイルガイド中心の設計だけでは、真に操作性に優れたGUIの構築は期待できない。こうした問題が起こる背景には、ガイドラインとスタイルガイドとの研究が独立に行われてきたため、明確なコンセンサスが両者でとられていないことに起因しているようであるが、本論ではこの問題にはこれ以上触れない。要するに、ガイドライン・スタイルガイドを利用する立場からいえるのは、ユーザの要求する機能、設計のガイドライン及び使用するキットのスタイルガイドの三つをバランスよく考慮することが重要なのである。

本研究はこうした現状を踏まえて、ユーザビリティを重視したGUI構築を試みる。その事例として、飛しょう体シミュレーションソフトウェア開発を行う開発者向けGUI構築を取り上げる。以下で述べるユーザとは、シミュレーションソフトウェアの開発経験をもつ、ソフトウェア技術者を指す。第2章では、開発システムに要求される機能の概略を示す。第3章では、要求機能を実現すべきGUIの設計を行う。現在

ガイドラインは比較的整備されているものの、使用するワークステーションやツールキットに基づいた検討が必要である。さらに状況によっては原則まで遡った解釈も必要である。第3章では、本研究で使用するグラフィックツールキットのスタイルを意識した原則に基づく設計を行う。第4章では、得られた開発環境でシミュレーションソフトウェアを開発している事例を示す。

2. システムの機能概略

2.1 シミュレーション・ソフトウェアの開発環境

飛しょう体の研究開発は、いくつかの段階に分けて進められる。一般的にそれは、運用要求に基づく機能分析、要求性能の最適化配分及びサブシステム分割、サブシステムの部分試作、システム合成並びに総合試験に分けられる。シミュレーションは、この流れのほぼ全域で実施される。特に本研究で対象とするシミュレーションとはこれらのうち、サブシステムへの分割及び部分試作段階で実施する数学的シミュレーションを指す。例えば、サブシステムの各特性を検討するための誘導制御系、空気力学、推進系、弾頭信管系の各シミュレーション、また機体の運動特性を検討するための飛しょう経路シミュレーション等がこれに相当する。

設計変数やモデルの実体に違いはあるものの、シミュレーション・ソフトウェアの開発工程には共通点が多く、ワークステーションを利用して一貫した作業を進めることができる。開発環境の一例を示すと第1図のようになる。シミュレーション・ソフトウェアの開発プロセスは、通常のソフトウェアのそれと変わりない。すなわち、機能設計、外部設計、内部設計、コーディング、単体試験、及びシステム試験の順に実施する。これらのプロセスは一台のワークステーションで行えるが、設計プロセスとコーディング以降のプロセスとは使用する開発支援ツールが異なる。前者ではCASE (Computer Aided Software Engineering) に基づく汎用支援ツールがいくつか知られており、活用されている。一方、後者にはコードデバッガのような特定の機能をもったツールはあるもののCASEツールのように体系化されたものはない。本論で構築する開発環境は、後者のプロセス(コーディングからシステム試験)を通して一貫して利用できるものを目指す。

2.2 ユーザインターフェイス機能 第1図に示す環境のもとで、ソフトウェア開発者(本論でのユーザ)がコーディングからシステム試験までを行うにあたり必要と判断したインターフェイス機能を第1表に示

第1図 ワークステーションを用いた開発環境・模式図
(Xウィンドウのクライアント・サーバー型)

第1表 ユーザインターフェース要求機能一覧

編集機能	シミュレーションプログラムのコーディングおよび実行を制御する機能である
ファイル操作 File	Open ファイルを開く Save ファイルを保存する Save As ファイルを別名称で保存する Print ファイルの内容を印刷する File Manager ファイルの一覧表を開く Exit 編集機能を終了する
編集操作 Edit	Cut 文字列を切り取る Paste 文字列を貼り付ける Copy 文字列を複写する Find 文字列を捜す
その他 Special	Show File Name 編集中のファイル名を表示する Command UNIXのコマンドを実行する Sensitive All ロック状態になった機能を解除する
作図機能	シミュレーション結果（モニタデータファイル）を読み込み、結果を作図する機能である
ファイル操作 File	Load モニタデータファイルを開く File Manager ファイルの一覧表を開く View Other Graph 他の作図結果を見る Exit 作図機能を終了する
作図 Plot	Layout 作図のレイアウトを変更する Graph1 図番1の作図内容とスケールを設定する Graph2 図番2の作図内容とスケールを設定する Graph3 図番3の作図内容とスケールを設定する Redraw 図の再描画する Hard Copy 作図のハードコピーをとる

す。なお、表に示した機能は最小限の機能であり、使用頻度は低いが、あれば一層便利な機能は含まれていない。例えば、Undo（一度実行した機能を取り消し、実行前の状態に復元する）機能は含まれていない。ユーザから要求のあったインターフェイス機能は細かなレベルで見ると個人差があるものの、大別すればそれらは「編集機能」と「作図機能」に分けられる。「編集機能」とは、いわゆるフルスクリーンエディタ（ここではテキストのin-line入力が可能なエディタを意味する。UNIXのedはin-line入力ができないので該当しない）機能とシミュレーションのテストランを行うための機能からなる。一方、「作図機能」は主にシミュレーションの実行結果を実行後にオフライン表示する機能からなる。

これらの機能の実現にあたって、留意した点を以下に示す。一つめは、編集中のプログラム画面から直接UNIXのシェルコマンドを起動できるようにした点である。そのねらいはコーディングを中断することなしにプログラムのコンパイル、シミュレーションの実行を行うことにある。従来の方法では、プログラムのコンパイル・実行はシェル専用の画面（ktermやxterm）からシェルコマンドを起動する方法をとっていた。本システムでは、シェルコマンドを入力できるウィンドウを編集中の画面から生成できるようにして、編集画面を退避あるいは終了させることなくコンパイル・実行を実施できるようにした。こうすること

で、編集画面の退避や終了操作を行うときに感じる思考の中断は抑えられよう

二つめとして、汎用統計解析パッケージ¹⁴⁾のデータファイルの書式に倣い、シミュレーション実行時に出力するモニタデータのフォーマットを簡略化した点である。フォーマットは時系列順に記録したストリップデータでヘッダーやコメントは含まれない。ヘッダーやコメントがないとどの成分を何項目めに記録したかメモしておく不便が生じるが、モニタファイルを直接開いてデバッグすることは少ないので、この不便さはさほど感じない。そこで、本システムではデータそのものの見やすさより汎用パッケージ(可視化用ツール)とのデータ互換性を重視する立場をとった。

三つめは、モニタ表示はデバッグ中に頻繁に使用することを考慮して、モードレス環境(3.1.4(b)節参照)を導入した点である。飛しょう体シミュレーションのモニタデータの表示形式はタイムヒストリか航跡図である。タイムヒストリはペンチャートのイメージで、航跡表示はXYレコーダのイメージで行いたい。タイムヒストリは少なくとも3チャンネル(3成分)表示できないと不便である。一方、航跡表示はXY及びXZ表示ができれば十分である。モードレス環境であれば、作図ウィンドウを独立に複数起動することが可能となるから、あるウィンドウをタイムヒストリ表示に、別のウィンドウを航跡表示に割り当てることができる。

なお、本環境のOS(オペレーションシステム)はUNIX(パークレイ版4.2BSDサブセット+System V)である。ワークステーションはHewlett-Packard社のHP 9000モデル375を使用した。GUIにはオブジェクト指向型ウィジェットOSF Motif(ver 1.0)を使用し、C言語で記述した。

3. GUI構築の設計原則とその解釈

本章ではGUIの構築を行う。ユーザビリティを重視した設計ガイドラインか、その根拠となる原則に基づいた実現方法を示す。第2表に検討項目の一覧を示す。その内容はGUI構築にあたりMarcus¹²⁻¹⁴⁾が述べた原則(一部ガイドラインを含む)と、マウスの操作性に関するTelster¹⁵⁾、Shneiderman¹⁶⁾らの考え方を参考にまとめたものである。

なお、本論では大塚¹⁷⁾に従い、設計の「原則」と「ガイドライン」を次の意味で使用する。ガイドラインが対象とする機器のハードウェア・ソフトウェアの仕様に依存した具体的指針であるのに対し、原則はガイドラインがなぜ必要かについての抽象的(つまり機器に依存しない)理由付けとする。例えば、カーソルの形

第2表 設計にあたり留意した項目の分類表

大項目	中項目	小項目
操作性	レイアウト	構造化 一貫性 関連付け 慣例の踏襲
		読みやすさ
	文 字	経済性 コントラスト
	配 色	ナビゲーション モードレス環境
出 力	インタラクション	WYSIWYG

状を画面ごとに変えずアンダーバーに統一するというのがガイドラインに相当し(ただしアンダーバー表示をターゲットとなるソフトウェアで容易に実現できるという前提である)、知覚上の一貫性を保つというのがその原則にあたる。

3.1 操作性

3.1.1 レイアウト

(a) 構造化: 構造化は最も基本的な原則の一つである。それは、画面に要素(部品)を無秩序に配置するのではなく、意味あるいは操作の同じものを集めて、できる限り格子上に整然と配置することを意味する。この原則は、一画面に表示する項目が多く、階層構造が存在するときにきわめて有効である。第2図に一例を示す

構造化を行うにあたり注意すべきことは、配列する項目数をMiller数¹⁸⁾以内に収めることである。我々が短時間に一度に認識できる情報には、主観的なまとまりが存在し(chunk)、その大きさには限界がある。その限界数(7±2)をMiller数と呼んでいる。この考え方に従い、水平・垂直いずれの方向にも配置する区分け及び項目の限界数は9とした。

(b) 一貫性: レイアウトの一貫性を保つために、同じ種類の要素は同じ外観とし、異なる種類の動作を伴うものはそれぞれ独自の外観を持たせる。同一の動作を伴うものは、画面の中で相対的に同じ場所に配置する。例えば、ダイアログを閉じるための操作のOKボタンとCancelボタンの位置をダイアログ毎に変えたりしない。ここではダイアログの最下列に左からOK, Cancelの順で配置した。他のボタンを定義する場合は、それらの右側に置くようにして、一貫性を保った。

(c) 関連付け: 図形を認識する時、我々はその要素の形・配置に強く影響を受ける(群化の法則)。オブジェクトは同一でも要素間にまとまりがあることを伝

第2図 レイアウトの構造化の例

右図：ラベル、入力エリアの配置例

左図：ラベル、入力エリアの配置に使用した方眼

第3図 セパレータを用いたプルダウンメニューの関連付け

(a) セパレータなし、(b) セパレータあり

第5図 編集ウィンドウと作図ウィンドウのタイトルバーメニュー

第4図 代表的なウィンドウシステムのメニューバー上のメニューの配置

えるには、関連する要素をつなぎ合わせ、要素間の関係を誤解なくかつ容易に判断できるよう工夫しなければならない。要素の形や配置が単調で、視覚的な強い関連付けが容易でない場合は、補助線や色の濃淡を用いて工夫する。第3図は、表示した項目の機能的まとまりを視覚的に理解させるため、メニューに補助線（セパレータ）を引いた例である。

(d) 慣例の踏襲：これまで述べてきた構造化・一貫性・関連付けは、新規性の強いシステムの設計に役立つ。しかし、ガイドラインに忠実すぎるあまり、ユー

ザが馴染んできた手順から離れると逆に使いづらくなる。状況に応じて慣例の操作方法を優先せよというのが、慣例の踏襲である。一例を第4図に示す。図は代表的なウィンドウシステムのタイトルバーに並べられたメニューの配置を比較したものである¹⁹⁻²³⁾。図から明らかなように、メニューの並べ方に共通性を見いだすことができる。第一に、最左列はFileに関する操作が割り当てられている。第二に、一例を除いて左から2つめの位置はEdit（編集）に関する操作が割り当てられている（NeXTのOptionメニューにはEdit機能が含まれるため）。本システムのメニューの並べ方もこのルールに従い、「編集機能」では左からFile, Edit, Specialとし、「作図機能」では、File, Plotとした（第5図）。

3.1.2 文字の読みやすさ（readability）文字が読みやすいということは、視覚的に判読しやすく、表記された言葉の解釈に誤解が生じにくいということである。読みやすさに関する原則¹²⁾と本システムでの実現方法を第3表に示す。以下、表に示した順に説明する。

まず、コマンドやラベルを表す文字（英数字）列の表記には、大文字と小文字の両方を使う。テキストを大文字だけで表記すると、読み取り速度は12%ほど

第3表 文字の読みやすさの原則とその解釈

No	原 則	解釈 (実現方法)
1	コマンドは大文字だけ、小文字だけで表記しない	コマンド (ワード) は先頭1文字のみ大文字を使用する。File, Cancel, Edit 等
2	書体・文字の大きさは3種類以内に抑える	コマンドには Helvetica 体を用いる テキストには Courier 体を用いる
3	コマンド・テキストのアライメントを統一する	コマンド・テキストの配置は左揃えに統一する
4	可変幅フォントのワード間スペースは小文字の r または i の大きさに合わせる	ワード間のスペースは Helvetica 体の小文字 r の大きさとする
5	コマンドの構文を統一する	コマンドは「動詞+名詞」型構文で統一する。ただし、Cut, Paste, Copy は「名詞+動詞」型構文にする
6	意味が類似する語彙を混用しない	○ Find (× Search), ○ View (× Display) ○ Exit = 全系の終了 (○ Quit = ダイアログの消去) とした ○: 採用, ×: 不採用

低下することが指摘されている¹²⁾。ここではコマンド、ラベルは、各ワードの先頭1文字だけ大文字で表し、残りは小文字で表記するようにした。

次に、書体 (フォントスタイル) は目的に応じて使い分けた。書体は3種類以下、文字の大きさも3種類以内が望ましい。本システムの表記は英数字のみ、つまり日本語表示をしないことを前提に、コマンド、ラベルの表示には、見出し文字としてよく使われる Helvetica 体を用いた。Helvetica 体は見やすいが、文字幅が可変のため桁が不揃いになりエディタには不向きといえる。よってソースコードの表示のみ、等幅文字の Courier 体を使用した。

さらに、アライメントを統一した。アライメントとは、文字列とそれを囲む枠との配置関係をいう。ラベルの表示、テキスト (文字列と数値) の入力左揃えとした。一般に数値表示は右揃えが推奨されている。しかし、入力モードのデフォルトは挿入モードであることを考慮して左揃えに統一した。挿入モードを右揃えにすると、文字入力に伴いカーソルは不動で入力文字が左側にシフトしてしまう。この動作では入力文字とカーソルが右に移動していく見なれた動きと異なるため、大変操作しづらい印象を受ける。

可変幅の書体を用いた時には、ワード間の空白 (スペース) を定義する必要がある。ここでは、小文字の

“r”に相当する大きさに設定した。

以上は判読しやすくするために行った処置である。次に、画面上に表示したコマンド (例えば Open, Close) の機能を誤解されないためのコマンド名称の選び方について述べる。

Carroll²⁴⁾ はコマンド名称の統語分析と実験を行い、意味上の誤解を最小限に抑える方法を考察した。その結果によると、ユーザの誤解を最小限に抑えるためには、操作順序に階層概念 (Hierarchicalness) を導入する必要がある。さらに、操作手順は「Operator (+Operand (+Specifier)) ; 動詞+名詞」型構文に統一するのが望ましいとのことである。

例えば、「ファイルを開く」操作と、編集時のテキストを「別のファイル名で保存する」という操作を考えてみる。操作順序の統一とは、構文を統一することに他ならないから、ファイルを開く操作は、「Open メニューを選択したあとでファイル名の入力」となる。またテキストを別名称で保存する操作は、「Save as を選択したあとでファイル名の入力」となる。

Carroll の主張は、キーボードとキャラクタ端末が主流であった頃の議論であるため、現在のマウス操作が中心のシステムでは検討を要する。キーボードとマウスというデバイスの違いはあるものの、計算機にメッセージを伝える順序は本質的に変わっていない。つまり、本システムで採用するプルダウンメニューによる操作にも、Carroll の主張は成り立つといえる。ただし、一部の操作 (Cut, Copy, Paste) については、ユーザはすでに直接操作 (direct manipulation)²⁵⁾ に慣れているので、それらの操作だけは Operand+Operator 型とした。

最後に、意味が類似したコマンドの混用は避け、一つの機能には一つの言葉を割り当てた。例えば Erase と Delete, Quit と Exit, Display と View, Find と Search との混用は避けるべきである¹⁰⁾。本システムもこの考え方に従う。ただし、Quit と Exit との概念的な違いをユーザが理解していたのでは混乱はないと判断し、両者を使用した。本システムでは、システム全体の終了に Exit を使用し、各イベントの終了 (例えばダイアログの消去) には Quit を使用した。

3.1.3 配色

(a) 経済性: レイアウトの構造化のところで述べた Miller 数は、配色についても成り立つ。すなわち使用する色の種類は7色±2色以内に収めるべきである。本システムでは、black, slight grey, spring green, ivory の4色を用いてオブジェクトを表示するようにした (ただし、これはフォーカス状態にあるオブジェクトの場合である。アンフォーカス状態のとき

の配色は(b)に述べる)。

(b) コントラスト：マウスポインタは、画面のどこにあっても発見しやすいような配慮が必要である。マウスポインタを他のオブジェクトに比べて常に前面に浮き出するような配色が望まれる。ここでは、マウスポインタのみ赤色を使用して、他のオブジェクトとのコントラストが現れるようにした。

次に、オブジェクト（ウィンドウやダイアログ）をマウスポインタで選択した状態（フォーカス状態）と、そうでない状態（アンフォーカス状態）とを一目で判別できるようにするため、アンフォーカスな状態のオブジェクトはすべて同一色で表示するのが望ましい。本システムでは、すべてのオブジェクトの色彩だけを落として、アンフォーカス時のオブジェクトの色を slight grey にした。

さらに背景画面(backdrop)には、注視していても疲れにくく目障りでない色を割り当てた。網膜の色彩感覚器官は青に敏感なもの（S 錐体）が極端に少なく、これは視野の明るさの調整に関与しない²⁶⁾。背景色に青を使用して目の疲れを抑えるとともに、背景画面の視覚的位置を、アンフォーカス状態のオブジェクトよりさらに背後に置いたため dark blue に設定した。

最後に辺縁対比効果²⁷⁾を利用してオブジェクトに立体感を与えた。辺縁対比効果とは、明るい帯と暗い帯が接するオブジェクトを見たとき、帯の境界線の両側で暗い側はより暗く、明るい側はより明るく見えるという錯視効果である。ボタンエッジ部分の明るさを、ボタンと地の部分の明るさより、さらに明るいかに、より暗くとる。このエッジで囲まれた図形を注視すると、明暗の接解部分の組合せから、遠近感が現れる(第6図)。ボタンが押された時には、明暗のパターンを反転させればボタンが押されたように見える。

以上配色についての原則と、本システムでの実現方

法の概要を第4表に示す。

3.1.4 インタラクション

(a) ナビゲーション：一連の操作を繰り返す行うとき、マウスポインタが画面にあちこち飛び回ったのでは疲労感を招く。機能的には独立していても、一連の操作で行うものは近くに配置してマウスポインタの

第4表 配色についての原則とその解釈

No	小項目	原則	解釈（実現方法）
1	経済性	使用する色数について Miller 数 (7 色±2 色) 以内に収める	black, slight grey, spring green, ivory の 4 色とする
2		カーソルは他のすべての画面要素に対して近視効果を持つ色を選ぶ	マウスポインティングカーソルのみ赤色を割り当てる
3	コントラスト	彩度変化を与えて、フォーカスウィンドウの判別を容易にする	フォーカスしていないときのウィンドウの彩度を落とし、grey 表示する
4		背景色は注意度を下げため色彩感覚器官の感度特性に依存した色を用いる	背景には奥行きを感じさせるため、背景色は dark blue にする
5		辺縁対比効果により、表示要素に立体感を持たせる	ボタンが押されたときに、エッジの明暗を反転させて、押されたように見える

(a)

(b)

第6図 左図：ボタンの立体感の例 (a：通常、b：ボタンを押したとき)、右図：(a)のA-A'に沿ったボタンエッジの明るさの比較模式図、破線：真の明るさの変化、実線：視覚による明るさの変化

第7図 ナビゲーションの改善
(a) スクロールバースライダが右側の場合(ナビゲーション大)、(b) スクロールバースライダが左側の場合(ナビゲーション小)

移動量を最小に抑えたい。

第7図は、編集しながら文字列をCut & Pasteする場合のマウスポインタの動きを模式的に表している。図はCut操作とPaste操作の間に画面のスクロール操作を行う状況を示している。図の(a)と(b)とのレイアウト上の違いは、スクロールバースライダーの配置にある。(a)はMotifの標準的なスタイル(デフォルト)である。本システムではこれを(b)の配置に変更して、スクロールバースライダーとプルダウンメニューとの操作距離は短くした(Motifのオプションで実現可能)。

(b) モードレス環境：システムが複数のモードをもつとき、ある操作をユーザが行えるかどうかは状況ごとに違ってくる。ユーザは、操作が予期せぬ結果を招かないかどうか気に使うようになると、モードを意識するようになる。ユーザが例外的な操作を行うことによりシステムに重大な危険をもたらす場合、モードはシステムを保護する役割をもつ。反面、モードの導入はユーザにとって不要な概念を強要することになりかねないので、導入は慎重に行うべきである。

本システムでは、次の方針でモードの使用を回避し

た。一つは、デバック作業ではプログラムの修正画面を開いたままで実行した結果を見られると便利であることを考え、編集機能と作図機能を独立したプロセスにした。編集モード、作図モードの概念は不必要となり、それぞれの機能が相手の状態に拘束されることがもはやなくなった。

もう一つは、ダイアログのモードレス化である。ダイアログを複数開いた状態で、任意のダイアログから操作を行えるようにした。これにより、見えるものはいつでも操作できるという設計思想が守られた。

3.2 出力

3.2.1 等価性(WYSIWYG) これまで画面の見栄えを中心に述べてきたが、印刷したときの見栄えも重要である。WYSIWYG (What You See Is What You Get)の考え方は、画面に視覚的に表現されたものだけでなく、紙面に出力した結果にも適用されるべきである。出力結果は、画面に表示した結果と見栄えの点で何等変わりのないことが望ましい。これを行うには、通常二通りの方法がある。第一の方法は、画面をビットマップフォーマットで出力する方法であり、第二の方法は、図形記述言語(例えばポストスクリプ

第8図 本システムのGUIメニュー構成図

ト言語）を用いて出力する方法である。本システムはXウィンドウシステムであり、その標準である前者の方式を採用した。

4. GUI構築結果

本システムのメニュー表示のツリー図を第8図に示す。メニュー、ダイアログは漸進的開示(progressive disclosure)となっていて、図の左の項目から順に右側の項目を選択する。モードレス・ダイアログで示された内容は、それが表示されていれば、いつでも実行が許されることを意味する。

最後に、本システムである飛しょう体シミュレーション・ソフトウェアを開発した使用例を示す。第9図は、ある飛しょう体の飛しょう経路の計算ブロックの概念図である。図は、誘導制御を行わない弾道飛しょうの例である。第10図は、コーディング中のプログラムからプログラムのコンパイル及び実行を起動している例である。第11図は、シミュレーション・モニタデータの表示例である。図のグラフは上から飛しょう距離(x成分)、機体速度(x成分)、推力の時間による変化を表している。作図のレイアウトを変更して同じ結果の飛しょう距離-機体速度表示を行ったものを第12図に示す。作図のレイアウト、表示内容の変更は、

モードレス・ダイアログで行えるため、表示結果を見ながら変更の指示が容易に行えるのが特徴である。

なお、第11図はマニュアルでスケールを指定した例であり、第12図は縦軸をオートスケールで描いた例である。見栄えをよくするため、オートスケールは変域(最大値と最小値との差)と作画域との比が常に一定となるように描いている。

5. ま と め

本研究では、飛しょう体シミュレーション・ソフトウェア開発環境の構築にあたり、GUI設計のガイド

第9図 飛しょう経路シミュレーションの機能ブロック概略の一例²⁰⁾

第10図 コーディング中のシミュレーション・プログラムからコンパイルを起動する一例

第11図 モニタ表示の一例

上図：飛しょう距離 x 成分；中図：機体速度 x 成分；下図：推力パターン，右下に表示のウィンドウは第8図(a)のモードレス・ダイアログの一部

ライン及び設計原則に基づいた設計を行った。また事例として構築した開発環境のもとで、ある飛しょう体の飛しょう経路シミュレーション・ソフトウェアの開発時の様子を示した。

シミュレーション環境へのワークステーションの導入が進むにつれ、GUIを構築する機会は一層増える予想される。GUIの構築は、設計・プロトタイプの実装・ユーザによる評価というプロセスの繰り返しである。繰り返し作業を早期に収束させるための方策として、ガイドラインの積極的な導入が望まれる。しかし第1章で述べたように、ガイドラインを直接適用するにはいくつか解決しなければならない問題もある。ガイドラインはスタイルガイドと独立に示されている場合が多く、スタイルガイドにはガイドラインは含まれていないというのが現状である。したがって、期間や予算が限られている現場の設計者の進め方としては、ガイドラインにあまりかわらず、むしろGUIの設計原則と使用するスタイルガイドとを照らし合わせながら設計を進めるのが得策であろう。

本論をまとめるにあたり、三菱スペース・ソフトウェア(株)鎌倉事業所第一技術部渡辺克昭課長、清水裕

司グループリーダーには、開発経験に基づく数多くの助言を頂きました。また同課、青山智彦君には事例で紹介した飛しょう経路シミュレーションプログラムの設計及び製作をして頂きました。合わせて感謝します。

なお、相山女学園大学短期大学部・谷口俊治助教授には、心理学用語の使い方について原稿の段階でいくつかのご指摘を頂きました。お礼申し上げます。

参考文献

- 1) Open Software Foundation: OSF/Motif Style Guide (Rev. 1.1), Prentice Hall, Inc., 1991.
- 2) 日本サン・マイクロシステムズ(株): OPEN LOOK™ スタイルガイド Graphical User Interface Application Style Guidelines, 星雲社, 1991.
- 3) アップルコンピュータジャパン(株): Apple Human Interface Guidelines, Apple Desktop Interface (日本語版), トッパン, 1989.
- 4) Norton, P. and Yao, P. L.: Peter Norton's Windows 3.0 Power Programming Techniques, Bantam Books, 1991.
- 5) Berry, R. E.: Common User Access—A Consistent and Usable Human-Computer Interface for the SAA Environments, IBM SYSTEMS JOURNAL, 27, No. 3 (1998).
- 6) Card, S. K., Morgan, T. P. and Newell, A.: The Psychology of Human-Computer Interaction, Lawrence Erlbaum Associates, 1983.
- 7) Norman, D. A.: Cognitive Engineering, D. A. Norman

第12図 モニタ表示の一例 (第11図のレイアウトを変更したもの)
縦軸: 機体速度 x 成分, 横軸: 飛しょう距離 x 成分, ロ印: タイムスタンプ (0.5 秒間隔)

- and S. W. Draper eds., User Centered System Design, Lawrence Erlbaum Associates, 1986.
- 8) Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley, 1987 (東 基衛, 井関 治訳: ユーザ・インターフェイスの設計, 日経 BP 社).
 - 9) Smith, S. L. and Mosier, J. N.: Guidelines for Designing User Interface Software, Technical Report ESD-TR-86-278, MITRE (1986) (田村 博訳: 利用インタフェース・ソフトウェア設計ガイドライン, フジテクノシステム).
 - 10) Mayhew, D. J.: Principles and Guidelines in Software User Interface Design, Prentice Hall, 1992.
 - 11) 市川伸一, 大橋靖雄: SAS によるデータ解析入門, 竹内啓監修, 東京大学出版会, 1987.
 - 12) Marcus, A. (日経バイト編集部訳): GUI を設計する (前編) グラフィック・デザインのノウハウと具体例, 日経バイト, 12 月号 (1990), pp. 287-297.
 - 13) Marcus, A. (日経バイト編集部訳): GUI を設計する (後編) グラフィック・デザインのノウハウと具体例, 日経バイト, 1 月号 (1991), pp. 257-264.
 - 14) Marcus, A. (黒須正明訳): 高水準の GUI デザインを利用したコミュニケーションを実現するための原則, 情報処理, 32 (1991), pp. 633-635.
 - 15) Telster, L.: The Smalltalk Environment, BYTE, 6 (1981), pp. 90-147.
 - 16) Shneiderman, B.: Direct Manipulation; A Step beyond Programming Language, IEEE COMPUTER, 16, No. 8 (1983), pp. 57-68.
 - 17) 大塚 巖: UI 設計原則の検討ー「ユーザー中心」視点からー, 第 8 回ヒューマンインタフェースシンポジウム講演予稿集, 計測制御学会, 1992, pp. 571-576.
 - 18) Miller, G. A.: The Magic Number Seven, Plus Minus Two: Some Limitation on Our Capacity for Processing Information, Psychol. Sci., 63 (1956), pp. 81-97.
 - 19) Kobara, S.: Visual Design with OSF/Motif, Hewlett-Packard Press, Addison-Wesley Pub. Co. Ltd., 1991.
 - 20) NeXT Computer Inc.: NeXT 日本語システムリファレンスマニュアルコンセプト, アスキー出版, 1991.
 - 21) Silicon Graphics: Programming the IRIS WorkSpace ver 1.0, 007-2006-010, 1989.
 - 22) ソニー(株): NEWS Desk ユーザーズガイド vol. 1, NWM-660, NEWS-OS-Release 4.0 マニュアル, 1990.
 - 23) SUN microsystems, Inc.: Open Windows Version 2 User's Guide, Rev. A, 800-4930-10, 1990.
 - 24) Carroll, J. M.: Learning, Using and Designing Filenames and Command Paradigms, Behavior and Information Technology, 1, No. 4 (1982), pp. 327-346.
 - 25) Hutchins, E., Hollan, J. and Norman, D. A.: Direct Manipulation Interfaces, User-Centered System Design, Erlbaum Associates, Hillsdale, New Jersey, 1986, pp. 87-124.
 - 26) 記録・記憶技術ハンドブック編集委員会編: 記録・記憶技術ハンドブック, 第 4 章 視覚の構造と機能, 丸善, 1992.
 - 27) 秋田宗平: 現代基礎心理学, 第 2 巻 知覚 第 1 章, 相場 寛編, 東京大学出版会, 1982.
 - 28) 渡辺昭昭, 清水裕司, 佐藤英樹, 谷口敦子, 鷲尾十郎, 青木義典: 飛しょう体のフィジカル・シミュレーション, MSS 技報, 1 (1989), pp. 32-41.