7N-2

ＬＩＳＰ上のオブジェクト指向言語ＫＰＬにおけるウィンドウ環境の実現

藤村　茂　鈴木　明　富田　昭司
横河電機株式会社

１．はじめに

現在，種々の分野で，ウィンドウ環境を用いたインタラクティブな操作環境を実現したシステムの開発が行なわれている．プログラム言語であるSmalltalk-80[1] では，ウィンドウ環境を言語仕様の一部として取り込み，ビットマップ・ディスプレイ上での高度な対話的インタフェースが実現されている．また，Lispの処理系においても，ウィンドウインタフェースについての研究が多々行なわれている．

一方，我々は，知識情報処理のための基本言語として，ＫＣＬ（Kyoto Common Lisp ）処理系上にオブジェクト指向言語ＫＰＬ（Knowledge Information Processing Language ）[2] を構築してきた．そして，この言語処理系において，Smalltalk-80流の環境を実現し，種々の部品を自由に組み合わせて視覚的ツールを構築できるようなウィンドウ環境を実現した．

そこで本稿では，まず，本ウィンドウ環境の構成，およびその実現方式について述べ，そしてこのウィンドウ環境を開発環境へ適用した例を示すことにする．なお，このウィンドウ環境は，現在，X Window System (X11R3)を用いて実装され，Sun-3/260上で稼動している．

２．構成

本ウィンドウ環境では，Smalltalk-80と同様ウィンドウマネージャも自分自身（ＫＰＬ処理系）で記述できるものとし拡張性を重視した．実現法としては，利用するウィンドウシステムに依存しないよう考慮し，図１のように階層的に分割し最下層ではウィンドウシステムの差異を吸収するようにしている．各層について以下で述べる．

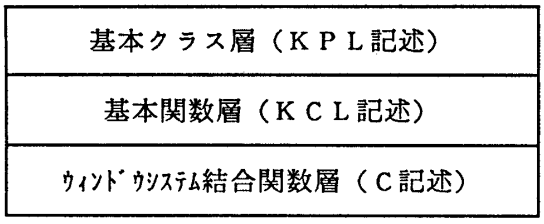

図１．ウィンドウインタフェースの構成

２．１．ウィンドウシステム結合関数層

ウィンドウシステムとLisp処理系との結合を行う部分で，ウィンドウシステムで用意されている基本的な入出力処理の関数を用いて実現している．現在の実装では，Ｃ言語，Xlibを用いて基本的な関数約６０個を記述し，組み込み関数としてＫＣＬ処理系に直接組み込んでいる．

２．２．基本関数層

高速な処理の必要性があり，拡張性あるいはメッセージパッシングの有効性の少ない部分をLispの関数として実現している．例えば，入力状態問い合わせの関数として，”ボタンが放されるまで待つ”などを上記の関数を組合せて実現している．さらに，例えば，点・矩形などの基本データの操作関数を抽象データ型でこの層に記述している．

２．３．基本クラス層

このクラスライブラリの層は，基本的にSmalltalk-80 のクラス階層に従っており，ウィンドウマネージャやアプリケーションプログラムを作成しやすいようにＭＶＣの概念[3] にもとづいて構成されている．各ツールは，トップビューと呼ばれるウィンドウの中に種々の機能を持つサブビュー（サブウィンドウ）を組み込むことで構成できるようになっている．それらの部品を約５０個のクラスとして実現した．

３．実現方式

３．１．ウィンドウシステムの機能の利用

本ウィンドウ環境では，ウィンドウシステムのウィンドウ管理機構を用いることができるため，描画のための座標変換，クリッピング領域の計算，保存型ウィンドウによる再描画はウィンドウシステムに任せ，処理の高速化を行っている．

３．２．トップレベル制御ループの実現

Smalltalk-80では，入力イベント処理と，アプリケーション処理・ビュー管理処理が処理系内で定義された別プロセスで処理されている．しかし，ＫＰＬではプロセスの概念がないため，１プロセスとして全ての処理を行う必要がある．そこでＫＰＬでは，図２のような制御ループを構成している．トップレベル制御ループでは，画面全体で受け付ける入力に応じてアクティブなコントローラ（ビューの入力処理を行なうオブジェクト）をビ

Realization of Window Environment for Lisp-based Object-oriented Language KPL
Shigeru Fujimura, Akira Suzuki, Shoji Tomita　Yokogawa Electric Corporation

ューの階層に従って探し，そのコントローラがアクティブな間はそこで定義されたアプリケーション処理が繰り返し実行される．この部分で，エディタの文字入力のループ，ビュー内での式の評価やポップアップメニューの表示および実行などが行われる．このループはアプリケーションの種類に応じて，同期式／非同期式の入力待ちを指定できるようになっている．

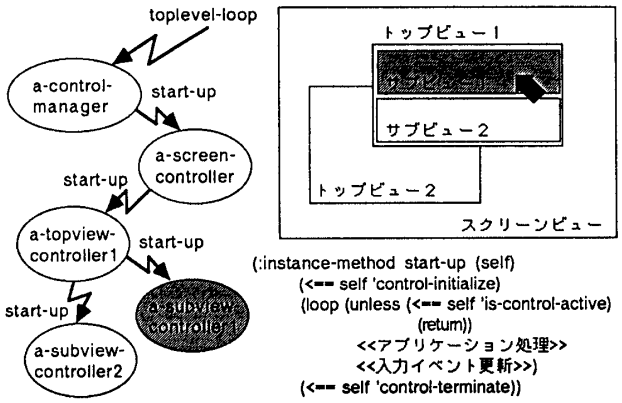

図２．トップレベル制御ループ

３．３．エラー処理ループの実現

上記の制御ループに従って処理を行なっている場合，アプリケーションの処理は通常Smalltalk-80の処理と同じように行なわれる．しかし，その処理においてエラーが生じた時，ＫＣＬのエラー処理ルーチンが呼ばれ，デバッグ作業をウィンドウ環境上で行なうことができない．そこで，ＫＣＬ処理系のエラーハンドラ，ブレークレベルの書換えを行ない図３のような構造に変更した．実際には，ウィンドウ環境内でエラーが生じた場合，ブレークレベル上でのトップレベル制御ループに処理が移行し，エラー箇所からの復帰は，catch-throw の機構を利用してもとのトップレベル制御ループに戻るように処理系を変更した．

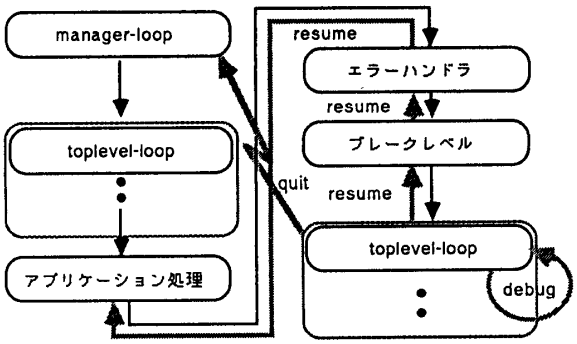

図３．エラー処理制御ループ

この機構を用いることで，エラーが生じた場合，エラー情報をユーザに知らせるためのビューが表示され，マウス操作により，その処理の続行／終了や，デバッガの呼び出しが指定でき，インタラクティブに作業が行える環境が用意できる．

４．開発環境への適用

このウィンドウ環境を用いて，ＫＰＬのプログラム開発環境（ブラウザ，インスペクタ，デバッガなど）（図４）や，我々がＫＰＬ上で開発している知的システム構築用シェルＡＵＫ[4] の開発環境を構築した．これらは，インタラクティブな操作によってプログラムや知識ベースの開発を進めることができる環境として実現できた．

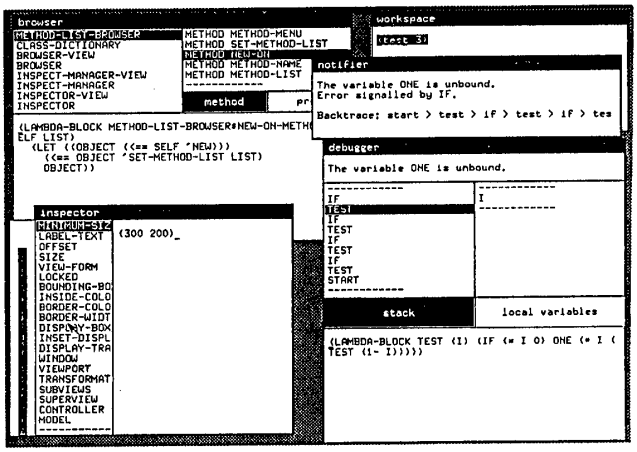

図４．ＫＰＬプログラム開発環境

５．まとめ

このように，Lisp上のオブジェクト指向言語ＫＰＬにおいて，クラスライブラリを用いることにより視覚的ツールを容易に作成できるウィンドウ環境を，汎用ウィンドウシステム上に実現した．今後は各種ツールの充実を行っていく．

参考文献

[1] A.Goldberg, D.Rabson :Smalltalk-80 :The Language and its Implementation－ Addison-Wesley, (1983)
[2] 鈴木他 :オブジェクト指向知識表現処理言語の試作とその評価－日本ソフトウェア科学会第２回大会論文集, p.109-112 (1985)
[3] 上谷 :統合化プログラミング環境 Smalltalk-80とInterlisp-D－丸善 (1987)
[4] 藤村他 :オブジェクト指向知識表現を用いた知的システム構築用シェル－情報処理学会第37回全国大会論文集(Ⅱ), p.1228-1229 (1988)