4Q−3

WELL−PPP用ウインド間インターフェース機能

丹羽　直人　鴨志田　稔　榎本　肇
芝浦工業大学

1．　はじめに

カラー画像処理のできるパーソナルコンピュータやワークステイションが普及してきている今日、そのグラフィック機能とコンピュータの処理能力を十分発揮し、ユーザーフレンドリーな環境を提供しているソフトはあまり存在しない。そして、ソフトウエア開発段階においても同じようなことがいえる。そのためカラー画像処理用言語 WELL-PPP(Window based ELaboration Language for Picture Processing and Painting [2])を用いて制作されるCPPP(Color Picture Painting and Processing) system用のウインド間インターフェースを制作する。この CPPP systemは対象（オブジェクト）指向を前提に制作されている。

このインターフェース・モジュールは CPPP system上の輪郭ウインド、色度ウインド、輝度ウインド、合成ウインドの各ウインド間におけるデータの受け渡し、ウインド相互間で使用されるデータをオペレージョンと関連させ、ウインド間インターフェースをとり、コマンドの実行をスムーズに行うことを目的としている。そのため、インターフェース・モジュールは基本的にはこれらウインドからは完全独立した形で実行されている（図１参照）。しかしウインド間インターフェースはWELL_PPPのライブラリーに似た形をとっている。

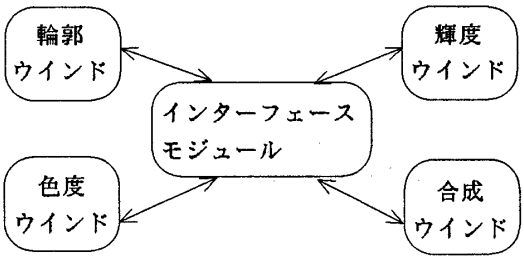

図　1　CPPP System構成図

2．インターフェースの機能

2.1インターフェースの主な処理内容

このインターフェースはモジュールとして存在している各プログラム間のメッセージ、データ交換を主な機能としている。この機能は各ウインドの要請によりデータを送ることができる。この要請とは相互にデータを扱うことができることを意味する。つまり、データを生成したウインド（以後オーナーウインドと呼ぶ）はインターフェースにこのデータを他のウインド（以後ユーザーウインドと呼ぶ）に送るように要求することもでき、他のウインドにあるデータの転送要求もできる。つまり、データの送受信要求が自由にでき、それを実行することが可能となる（図２、図３参照）。

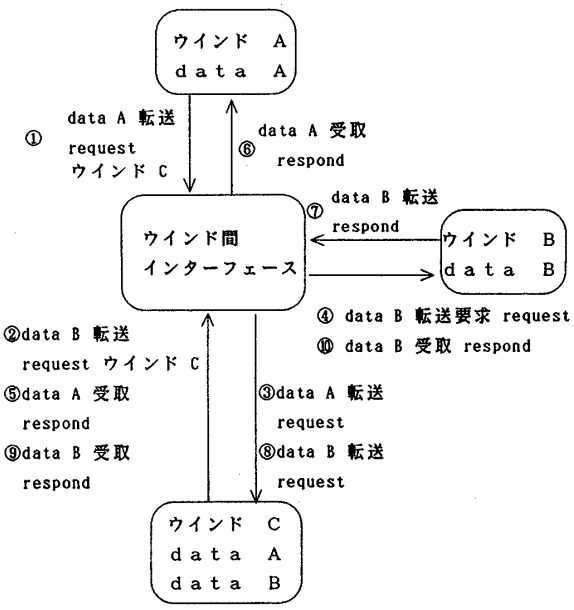

図　2　データ交換関係

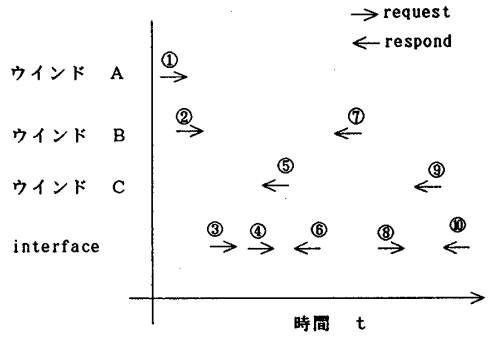

図　3　データ交換の順序

Function of window interface for WELL-PPP
Naoto NIWA,Minoru KAMOSHIDA,Hajime ENOMOTO
Shibaura Institute of Technology

2.2 data_nameの登録、転送

各ウインドは名前管理方式を採用しているため、生成したデータには data_nameが付いている。インターフェース・モジュールはこの data_nameとオーナーウインド名、ユーザーウインド名を対応させ記録していく。登録方法としては、同じ data_nameで別のデータを登録する事はできない。データを登録できるのはオーナーに限られていて、それ以外は登録できない。それ以外はユーザーとして登録されていく。ユーザーはユーザーウインドに取り込んだデータは加工できるが、同じ data_nameでほかのウインドへは転送できない。これにともない、ユーザーはデータの要求はできるが転送はできない。また、インターフェース・モジュールはデータ転送記録、メッセージ交換記録を記録、蓄積している。

2.3データの要求とデータ形式

インターフェース・モジュールではデータの要求と送付を request-respond機能を使い、各データは情報隠蔽の考え方を使って行っている。ここで requestするデータはどこのオーナーウインドのデータであってもかまわない。インターフェース・モジュールのこれらに対する respondは対応しているデータではなく専用のコードを結果として送り返し、データは別に送る。また、要求するデータの呼出方は data_nameと送付時の書式を送ることによりインターフェース内の data_nameバッファーを参照することによりどのオーナーのデータであるかが分かり、そのウインドに対し対応データと、データ書式の送付を要求する。送られたデータはインターフェースで要求のあったウインドに送る。そのため、データを requestしたウインドは求めているデータがどのオーナーウインドにあるか、どのような書式にて格納されているかは知らなくても良い。また、書式指定ができるため格納方法は気にしなくても良い。また、オーナーとユーザーウインドが記録されている事より、’分配’と呼ばれる転送を行う事ができる。オーナーウインドは対象データのユーザーウインドを知らなくても修正後の新しいデータを送る事ができる。これらの機能によりユーザーウインドはどこに必要なデータがあるのか、オーナーウインドはどのユーザーにどのデータを送ったか、などのデータを蓄積しなくても良い。

2.4各ウインドのインターフェースについて

インターフェース・モジュールは request-respond方式を使っているが、各ウインドにあるインターフェース・モジュールはイベント＆データドリブン方式を採用している。これは各ウインドがイベント＆データドリブン方式により処理を行っているためで、各ウインドごとのイベントやデータに関連した駆動機能が発生し、その駆動した事実がイベントとして処理をされ、データの受け渡しとしての最終駆動がインターフェース・モジュールへの requestとして機能する。また同じように、インターフェース・モジュールへでーた要求をしたウインドへの respondはイベントとなり、処理が行われこのシステムの信頼性を高める。

2.5メッセージパッシングについて

インターフェース・モジュールデータの転送だけでなく、メッセージパッシングをも行っている。インターフェース・モジュールではメッセージは別に特別な命令として扱ってはいない。そのためどのようなメッセージであろうと送る事はできる。また、このメッセージは特別なパイプを使って送られるのではなく、データ転送と似た形で送られる。これもインターフェース・モジュールの重要な処理です。

2.6処理時間、モジュールの大きさについて

インターフェース・モジュールはただ単にデータの移行をメインとしているため処理時間はほとんど必要としないし、インターフェース・モジュールでは処理時間をほとんど必要としてはいけない。また、プログラムソースも大きくてはいけない。しかし、インターフェース・モジュールが行っている処理内容は各ウインドのオブジェクト化に対し、前で述べたように大きな役割をはたしている。

3．評価

このインターフェース・モジュールは CPPP system用を前提に考えているため汎用性という点において弱い。このため今回制作した CPPP system用にしかならないが、各ウインドのカスタム化という点において十分であると思われる。

4．まとめ

プログラムのモジュール化の徹底化を行うことが、プログラムの制作時間の短縮、変更の容易さなどの利点を生ずる。しかしながら、各モジュールが作るデータだけで処理が行われるわけではない。そのため、各モジュール間の状態データの交換などのインタラクション、データ転送は重要になってきている。この様にプログラムのモジュール化の徹底化を行うにつれて各モジュール間の通信を行うインターフェースプログラムの存在、機能の充実が重要視され、必要となってくる。

<文献>

[1] 榎本　鴨志田　宮村：カラー画像処理・描画システムの開発
４２回情報処理大会論文集

[2] 鴨志田　榎本：カラー画像処理・描画用ウインド型言語
「ＷＥＬＬ−ＰＰＰ」
４２回情報処理大会論文集

[3] Enomoto H.,Yonezaki N.,Watanabe Y. and Saeki M. :
Towards Evolutional Structure for Database of Image
and object Body
1st Australasisn C. on CG.,1983

[4] Enomoto H.,and Miyamura I. :
Vector Representation Scheme of High Quality Picutre,
Signal Processing of HDTV Ⅱ
1990, Elsevier