PHIGSワークステーションのオブジェクト指向による実現

6K-3

白崎 昌俊　山守 主紀　塚本 隆啓

(株)沖テクノシステムズラボラトリ

1 はじめに

PHIGS は、ISO で標準化がすすめられているコンピュータグラフィックシステムの規格である。PHIGS では、物理的なグラフィック装置を抽象化して扱い、これをワークステーション(以下 WS と略)と呼んでいる。我々は、多種多様なグラフィック装置を WS として抽象化するため、各装置の機能を階層化し、それらをオブジェクト指向型プログラミングによって実現した。オブジェクト指向型プログラミング言語としては、LISP マシン ELIS の核言語である TAO を用いた。

2 PHIGS における WS の役割

PHIGS は、アプリケーションプログラムから各図形データを図形プリミティブ単位(Polyline, Fill area 等)で受取り、これをエレメントとしてストラクチャと呼ばれる図形単位を構成する。そして、ストラクチャは CSS (Centralized Structure Store) と呼ばれる図形データベースの中へ登録される。ストラクチャを表示するために、PHIGS では WS に対してストラクチャの提示(Post)という操作を行なう。

WS は提示されたストラクチャ中のエレメントを順次解釈しながら、物理的なグラフィック装置に対する描画コマンドを生成する。そして、この解釈過程のことを PHIGS ではトラバース(Traverse)と呼んでいる。カテゴリ OUTPUT, OUTIN に属する WS では、このトラバース機能によって図形を表示する機能を持つ必要がある。また、カテゴリ INPUT, OUTIN に属する WS では、三次元データの入力機能を持つ必要がある。

PHIGS における WS の役割は、このような機能を実現するためのインタフェースを多種多様なグラフィック装置に対して提供することである。

3 オブジェクト指向による WS の実現

多種多様なグラフィック装置を効率よくサポートするためには、以下の点について考慮することが必要である。

(1) WS カテゴリによる機能差
(2) 多種多様なグラフィック装置の能力差
(3) 新規グラフィック装置の追加等に対する拡張性
(4) 実行効率

これらの点に関し、オブジェクト指向の枠組みをどのように適用し、実現を行なったかについて報告する。

3.1 WS カテゴリによる機能の階層化

PHIGS では、各 WS を WS が持つ機能に従って OUTPUT, INPUT, OUTIN 等のカテゴリに分類している。そしてカテゴリによって異なる WS 状態リストと WS 記述テーブルとが、各 WS に対して定義されるため、これらを効率よく実現する必要がある。そこで、WS 状態リストや WS 記述テーブルの項目を三つのグループに分類し、それぞれの項目をインスタンス変数に持つ三つのＷＳ機能抽象クラスを定義した。

(1) クラス ws
　クラス ws は、すべての WS に共通な項目を持つクラスであり、すべての WS に共通するメソッドが定義される。
(2) クラス out
　クラス out は、カテゴリ OUTPUT に属するすべての WS に共通な項目を持つクラスであり、カテゴリ OUTPUT に属するすべての WS に共通するメソッドが定義される。
(3) クラス in
　クラス in は、カテゴリ INPUT に属するすべての WS に共通な項目を持つクラスであり、カテゴリ INPUT に属するすべての WS に共通するメソッドが定義される。

なお、カテゴリ OUTIN に属する WS は、クラス out とクラス in とを多重継承することによって実現することができる。

3.2 グラフィック装置の能力に応じた階層化

物理的なグラフィック装置には、三次元図形を陰線陰面消去して表示する能力を持つ高性能な三次元装置から、二次元のウインドウクリッピング機能すら持たない二次元装置まで多種多様なものがある。そこで、これら能力の異なる装置を統一的に扱うために、物理装置クラスに加えて、グラフィック装置の能力に応じた抽象クラスを定義した。

(1) 三次元装置抽象クラス out-3d, in-3d
　三次元装置抽象クラスは、三次元図形を直接表示できるような三次元装置に共通したファセット分割等のメソッドやデータを定義するためのクラスである。
(2) 三次元シミュレートクラス out-2d, in-2d
　三次元シミュレートクラスは、三次元装置の機能

An Implementation of PHIGS Workstations in Object Oriented Style

Masatoshi SHIRASAKI, Kazunori YAMAMORI, Takahiro TSUKAMOTO

OKI Techno Systems Laboratory, Inc.

をシミュレートし、二次元装置で表示する場合に共通した座標変換等のメソッドやデータを定義するためのクラスである。

(3) 物理装置クラス ws-3d-term, ws-2d-term, etc.

物理装置クラスは、物理的なグラフィック装置に固有のメソッドやデータを定義するためのクラスである。

このように、多種多様なグラフィック装置を、その能力に応じたクラスに階層化することによって、統一的に抽象化することができる。

3.3 装置の抽象化による拡張性の向上

オブジェクト指向型プログラミングでは、あるクラスを定義する場合、上位クラスとの差分だけ定義すればよいので、クラスの定義は下位へゆくほど容易となる。そこで、この特性を利用して物理装置クラスの定義をより容易にするためには、よく似た機能をまとめた抽象クラスを定義すればよい。例えば、二次元の物理装置クラスの上位抽象クラスとして、CGI に準拠した抽象クラス out-CGI, in-CGI や、X-Window をサポートする装置に対応するための抽象クラス out-X, in-X 等が考えられる。これらの抽象クラスを定義することによって、新規グラフィック装置の追加等が容易に行なえるため、拡張性をより向上することができる。

3.4 実行効率の向上

サポートするグラフィック装置が増えてくるに従ってクラス階層が厚くなってきても、実行効率を落とさないように以下の点について考慮した。

(1) メソッド呼び出しのオーバーヘッド

メソッドの定義におけるスーパークラスの呼び出しはメソッドコンビネーションをなるべく用いずに、直接スーパークラスを指定することによって行なう。従って、階層が厚くなるに従って増加するメソッド呼び出しのオーバーヘッドは最小限に食い止められる。

(2) オンデマンドローディング機構

数多くのクラスやメソッドの定義をすべてメモリに常駐しておくことは、メモリ効率の点からみて望ましくない。そこで、オンデマンドローディング機構を用いることによって、クラス階層に関する情報だけをメモリ上に常駐しておき、必要なクラスだけをファイルから読み込んでメモリ上へ置くようにした。従って、不要なクラスやメソッドの定義がメモリに常駐しないため、メモリ効率が下がることはない。

3.5 クラス階層の一覧

以上述べてきたクラスの階層を図1に、クラス名の一覧を表1に示す。

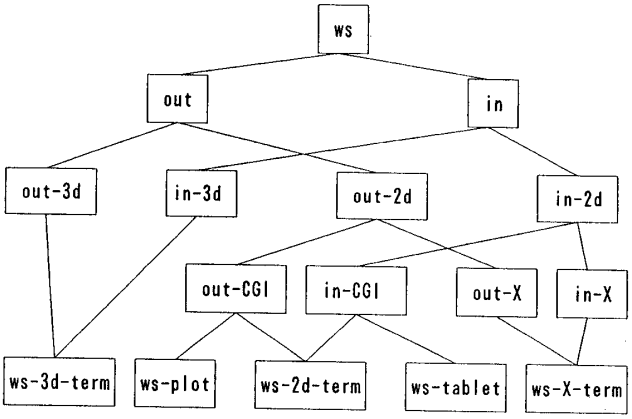

図1 クラス階層

表1 クラス名一覧

クラス名	説明
ws, out, in	WS機能抽象クラス
out-3d, in-3d	三次元装置抽象クラス
out-2d, in-2d	三次元シミュレートクラス
out-CGI, in-CGI, out-X, in-X, etc.	二次元装置抽象クラス
ws-3d-term, ws-2d-term, etc.	物理装置クラス

4 おわりに

オブジェクト指向型プログラミングによって、PHIGS における WS は、能力や機能が異なっていても、それぞれを抽象化したクラスを定義することによって、統一的に扱えることを示した。

今後は、以下の課題について対応してゆきたいと考えている。

(1) サポートできるグラフィック装置の増加
(2) 三次元シミュレートクラスの能力強化
(3) メタファイル機能の実現

謝辞：日頃から御指導頂く当社研究部の西垣部長、並びに沖電気工業総合システム研究所ＡＩワークステーション第二研究室の長坂室長に感謝致します。

5 参考文献

(1) ISO/DIS 9592 (PHIGS)
(2) ISO/DP 9636 (CGI)