グラフィクスとCAD 21－2
（１９８６. 5. ９）

オブジェクト指向によるグラフィックス・ソフトウェア開発用トゥールの試作

荒牧利充
ソニー・テクトロニクス株式会社 情報機器部

オブジェクト指向の基本的な概念であるクラス、インスタンス、インヘリタンス等の性質を持たせた環境をＦｏｒｔｒａｎ語で実現する。この環境は、特にグラフィックス・プログラムにおいてはディスプレイ上のオブジェクトとプログラム上のオブジェクトが対応するため、プロトタイプの作成に有効である。他にオブジェクト指向の環境では開発時に必要なデバッガ、インタプリタ、データ・メモリ・エリアの管理、エラー・トラップ等の手続きを簡潔な形で実現できる。

"Graphics software development tools by objective method" (in Japanese)

by Toshimitsu ARAMAKI
(SONY/TEKTRONIX Corp., 5－9－31 Kitashinagawa, Shina－gawa－ku, Tokyo, 141, Japan)

Environment of graphics software development by objective method of the concepts of Class, Instance, Inheritance, is described in FORTRAN Language. Objective－Fortran as an environment is especially effective to development of graphic software product prototyping, by the reason of the instance object in program is just as a visual object on display screen. Also in this environment, debugger, interpreter, data－memory－area－supervisor, error－type, etc. can be deve－loped in briefly.

1．　まえがき

Ｏｂｊｅｃｔ指向による言語として代表的なものはＳｍａｌｌｔａｌｋ－８０（＊ＸＥＲＯＸ　ｃｏｒｐ．）であり、本グラフィックス・ソフトウェア開発用トゥール（Ｏｂｊｅｃｔｉｖｅ－Ｇｒａｐｈｉｃｓ－Ｌｉｂ－ｒａｒｙ　以下ＯＧＬと略す）はこのＳｍａｌｌｔａｌｋ－８０のＯｂｊｅｃｔ指向言語としての基本的な概念をＦｏｒｔｒａｎ言語で記述したＯｂｊｅｃｔｉｖｅ－Ｆｏｒｔｒａｎの環境に、グラフィックスの機能を持たせたグラフィックス・ソフトウェア・プロダクトの開発環境である。

ＯＧＬのプログラム開発環境はＯｂｊｅｃｔ指向の開発環境と従来のＦｏｒｔｒａｎ言語のプログラムを混在させることが出来るため、現在のソフトウェア資産を新しい言語に書き換えることなしに、将来かなり有望と考えられているＯｂｊｅｃｔ指向の概念をもった開発環境に移行することが出来る。

特にグラフィックスにおいて、ディスプレイ上に視覚できるＯｂｊｅｃｔはプログラム上のＯｂｊｅｃｔをそのまま表示したものであり、プログラムの開発を効率良く行なうことが出来る。

2．　Ｏｂｊｅｃｔｉｖｅ－Ｆｏｒｔｒａｎ

Ｏｂｊｅｃｔｉｖｅ－ＦｏｒｔｒａｎはＳｍａｌｌｔａｌｋ－８０の概念を参考にしたものであり、Ｓｍａｌｌｔａｌｋ風の記述と類似した環境を限定された部分で実現するものである。

2．1　Ｏｂｊｅｃｔ指向言語の概念

Ｏｂｊｅｃｔ指向言語において、基本的な概念は下記の３つである。

a）　クラス　　（Ｏｂｊｅｃｔの抽象部分）

b）　インスタンス　　（Ｏｂｊｅｃｔの具象部分）

c）　インヘリタンス　（Ｏｂｊｅｃｔの継承概念）

Ｏｂｊｅｃｔ指向言語においては全ての表現はＯｂｊｅｃｔという概念に集約される。このＯｂｊｅｃｔはクラスという抽象部分と、インスタンスという具象部分に分離される。あるＯｂｊｅｃｔは定められた機能（性質）を持つが、この機能を動作させるためにはＯｂｊｅｃｔに対してメッセージを送ることによって行なうとする。

クラスは抽象部分であるために再利用することが出来、クラスに特定のメッセージを送ることによって生み出されるインスタンスは各々Ｏｂｊｅｃｔの限定された範囲内での種々な個性を表現する。クラスは階層構造になっており、本幹を基にして、徐々に詳細の性質を表現するクラスに細分化していく。この時末端のクラスはもと（親）の性質を受け継ぐものとする。具体的には末端のクラスに対してメッセージを送った場合、自分でそのメッセージが理解出来なければ自分の親のクラスに自動的に同じメッセージを送る。この概念をインヘリタンスという。

従って、ある端末のクラスに特定のメッセージを送ってインスタンスを生成するとそのインスタンスはクラスと同様に親のインスタンスを継承する。

次の図はクラスとインスタンスの概念、およびインヘリタンスの概念を表わしたものである。

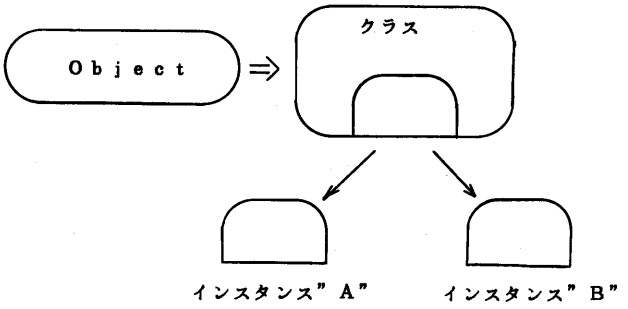

図1　クラスとインスタンス

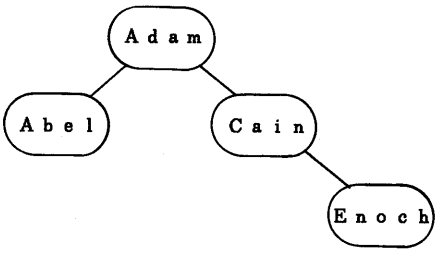

図2　インヘリタンス

2．2　Objective－Fortran

Objective－Fortranにおいて、クラスはある特定の機能を持つサブルーチンであり、インスタンスはそのサブルーチン内で定義されたデータ・エリアと同じ配列の大きさのデータエリアである。クラスはさらにインスタンス生成他に関する記述の部分と、作成されたインスタンスに対して送られるメッセージに応答する記述の部分に分けられる。クラスにメッセージを送るという概念は、サブルーチンにメッセージをパラメータで渡してcallすることで実現する。クラスに関する基本的な規則は次のように定める。

a）　クラスにメッセージを送った場合、所定の動作を行なった後もどされるのはインスタンスのアドレス、あるいはフラッグとする。

b）　クラスは再利用を可能とするため、具体的な値を持たないよう記述する。

c）　クラスは次の３つの基本的な機能を持つ。

①　インスタンスに値を入れる。

②　インスタンスから値をひろう。

③　クラスに所定の動作をさせる。（計算の実行など）

２．２．１　インスタンス・コール

次の図はクラスＰｅｎのインスタンスであるｐｅｎ１に対して、メッセージｃｏｌｏｒを送る場合の概念を表わしたものである。

（プログラム記述はＳｍａｌｌｔａｌｋ－８０ではメソッドという）

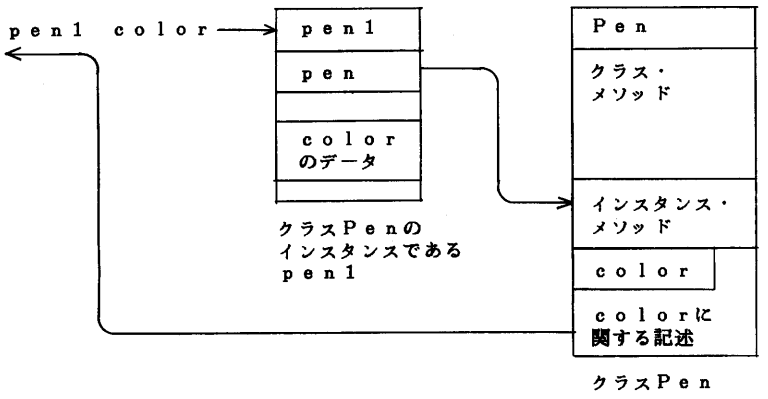

図３　インスタンス・コール

インスタンスｐｅｎ１にｃｏｌｏｒというメッセージを送ると、受けとったｐｅｎ１は自分の持つデータが処理される親がクラスＰｅｎであることを知っており、クラスＰｅｎに対して受けとったメッセージに自分自身のデータを付けて送る。クラスＰｅｎはインスタンスｐｅｎ１から受けとったメッセージを解釈し、受けとったデータを用いて所定の処理を行なったあと必要な値をもどす。

２．２．２．　インヘリタンス

インヘリタンスとは従来のサブルーチン・コールを階層的に行なう場合に類似した概念である。Ｏｂｊｅｃｔ指向言語の場合、前述のようにクラスは階層構造をもち、子のクラスにおいては親のクラスに付け加える必要のある部分の記述のみとする。次の図はクラスＰｅｎのインヘリタンスの概念図である。

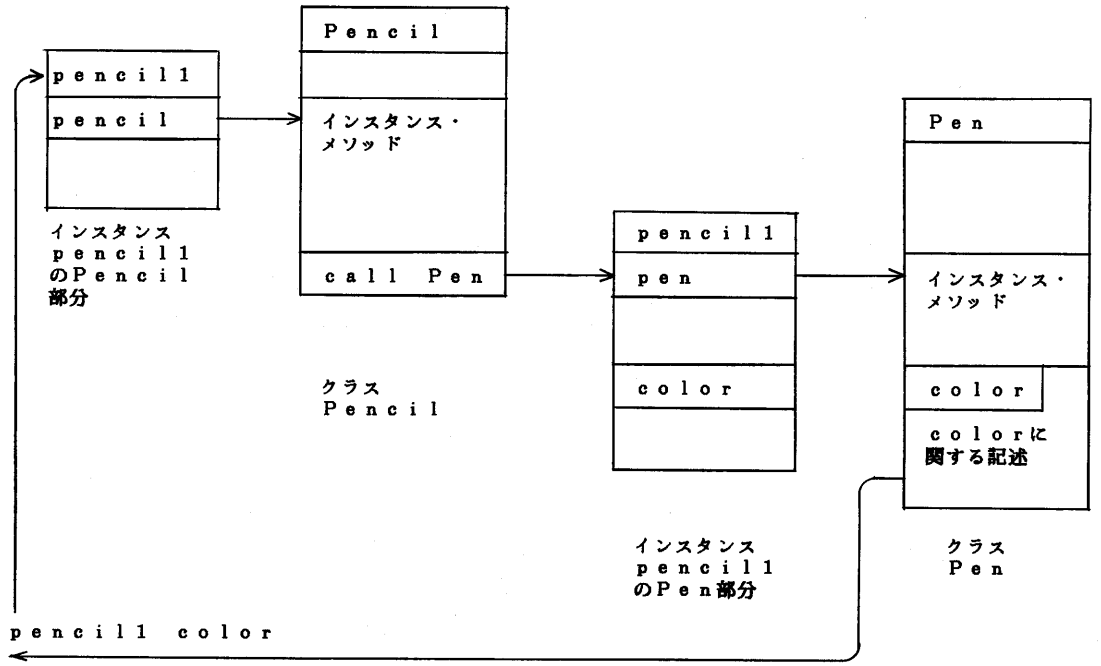

図４　インヘリタンス

インスタンスｐｅｎｃｉｌ１にｃｏｌｏｒというメッセージを送る場合、クラスＰｅｎｃｉｌは送られたｃｏｌｏｒというメッセージに関する記述を自分自身が持たないため、自分自身の親であるクラスＰｅｎに対してｃｏｌｏｒというメッセージを送る。クラスＰｅｎにはｃｏｌｏｒというメッセージに関する記述があるため、所定の処理を行なったあと、必要な値をもどす。

２．２．３　カーネル構造

Ｏｂｊｅｃｔｉｖｅ－Ｆｏｒｔｒａｎのカーネル構造の内、基本的な部分は図のようにクラスあるいはインスタンスのコール部分、およびインヘリタンスを行なう場合のインスタンスのアドレス計算部分に分けられる。ここでインスタンス（データ・エリア）のアドレス参照において、各クラスの中での参照は全てそのクラス内での相対アドレスであり、相対アドレスと全体のデータ・エリアでの絶対アドレスとの参照は全てカーネル部分で行なう。従ってクラス内の記述をする場合、全体での絶対アドレスを考慮する必要はない。

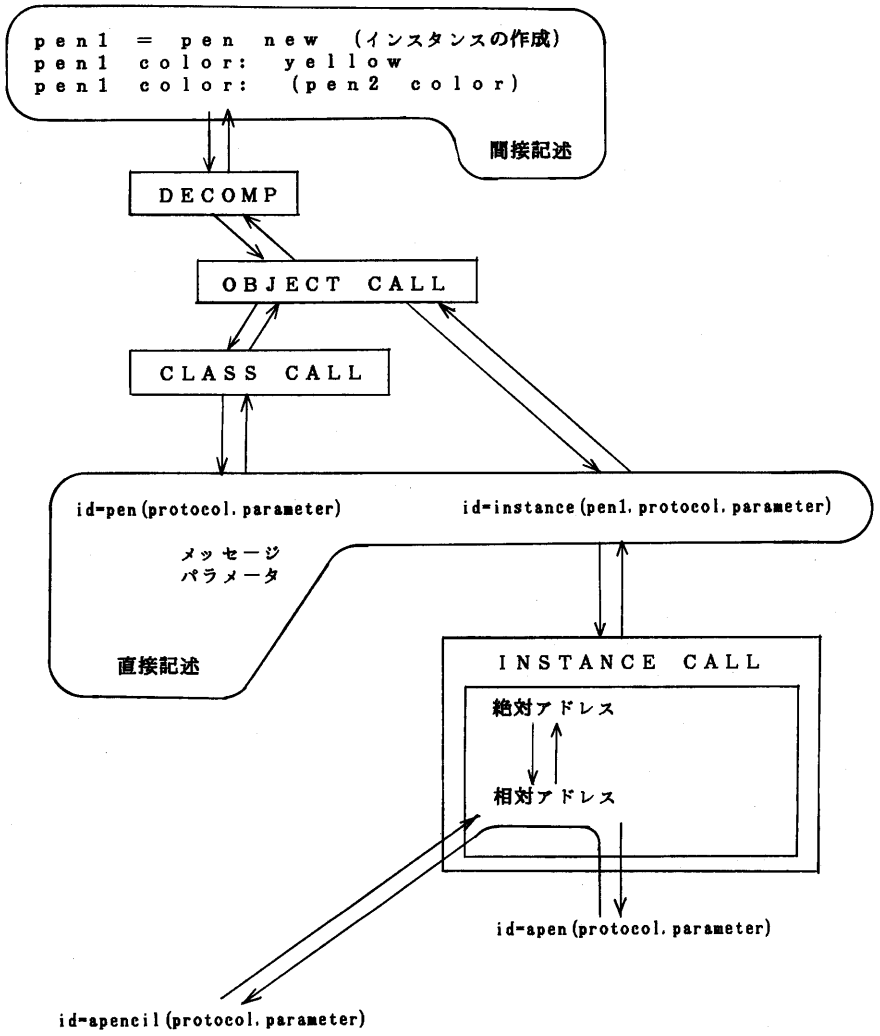

図５　カーネル構造

2．3　Ｏｂｊｅｃｔｉｖｅ－Ｇｒａｐｈｉｃｓ－Ｌｉｂｒａｒｙ

Ｏｂｊｅｃｔｉｖｅ－Ｆｏｒｔｒａｎのカーネルを基にして構築されるグラフィックス部分を含む全体のクラスの階層構造は次の図のようになっている。

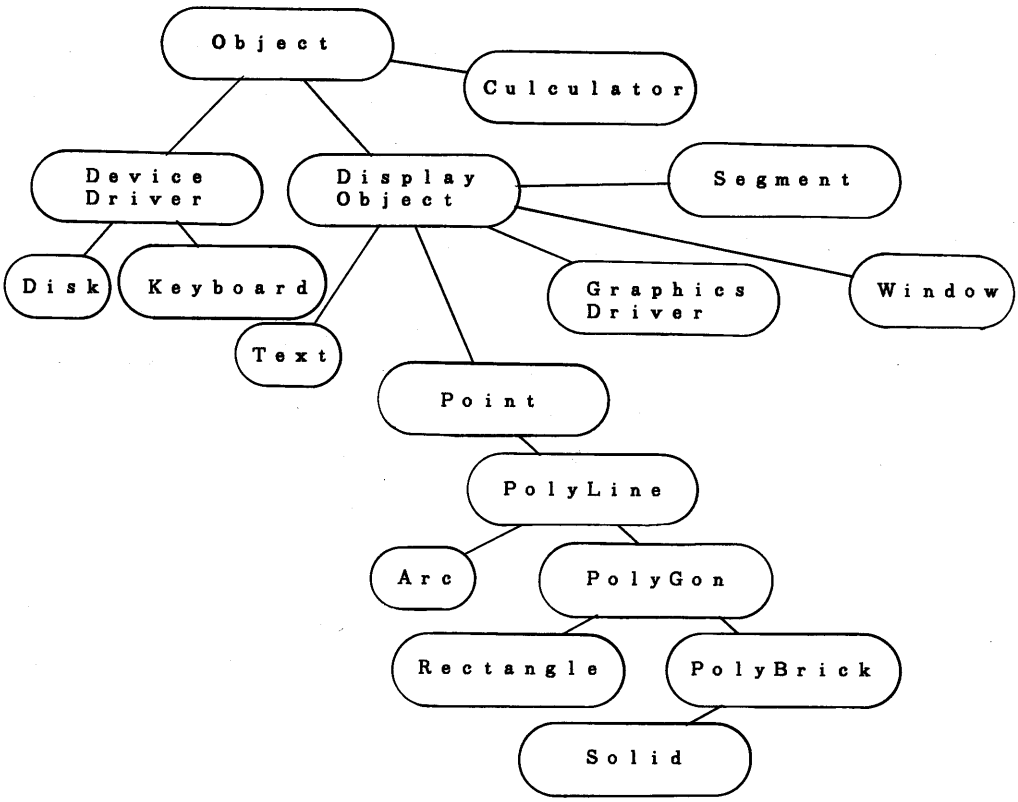

図６　ＯＧＬの階層構造

ＯＧＬの全体の本幹はクラスＯｂｊｅｃｔであり、その子としてクラスＤｉｓｐｌａｙ－Ｏｂｊｅｃｔがある。グラフィックスの機能は全てこのクラスの子として定義され、大きくデータ作成部分（点座標、点列、面、立体など）と作成されたインスタンスへのＩ／Ｏ部分とに分かれる。ＯＧＬで画を表示する場合の概念は以下のようになる。クラスＤｅｖｉｃｅ－Ｄｒｉｖｅｒ、またはクラスＧｒａｐｈｉｃｓ－Ｄｒｉｖｅｒのインスタンスとして作成される点座標、点列などのデータの集合はクラスＰｏｉｎｔ、クラスＰｏｌｙＬｉｎｅなどの子として登録される。これらのインスタンスは、その他必要な属性を与えられ、属するクラスＷｉｎｄｏｗのインスタンスに加えられる。このインスタンスに対しディスプレイ表示のメッセージを送ると、このインスタンスはクラスＧｒａｐｈｉｃｓ－Ｄｒｉｖｅｒによって端末依存のコマンド群に変換されディスプレイ端末に送られる。

3．　あとがき

オブジェクト指向の手法によるソフトウェア開発の利点は大きく２つあげられる。１つは効率の良い機能のモジュール化、階層化であり、もう１つはデータ部分と手続き部分を完全に分離することによってＯｂｊｅｃｔに関するインタプリタ、デバッグ・トレースなどのソフトウェア開発時に必要な種々のトゥールを簡潔に作成できることである。

機能のモジュール化、階層化はまた、ソフトウェア・プロダクトの開発の過程で作業の分散化に有効である。オブジェクト指向においては、そのデータ部分であるインスタンスがそのデータの処理される手続き部分であるクラスを知っており、さらにその各々が自分の親を知っているために、並列処理、分散処理など今後のシステムの機能別分散化に適合しているといえる。

今回試作したＯｂｊｅｃｔｉｖｅ－Ｆｏｒｔｒａｎ－Ｋｅｒｎｅｌは、Ｓｍａｌｌｔａｌｋ－８０のようにＯ／Ｓ、コンパイラなどハードウェアの基本命令から上の部分を全てＯｂｊｅｃｔ化したＯｂｊｅｃｔ指向言語ではなく、インスタンスに関する部分をＯｂｊｅｃｔ化したものであり、Ｆｏｒｔｒａｎ言語の制約であるための再帰呼び出しが出来ないなど完全なものであるとは言えない。またこのＫｅｒｎｅｌ上に種々な機能を持つＯｂｊｅｃｔを階層的に構築していく場合のＯｂｊｅｃｔ化はいくつかのアプローチが考えられ、使い勝手の良さ、処理速度の向上などの問題を考慮すると、さらに種々の試みが行なわれる必要がある。

－　以上　－