グラフィクスとCAD 28－12
（1987・8・20・21）

# (言語＋対話) 型CGモデラ：FUSION/MODELER

村上　公一，　林　一司
㈱富士通研究所　システム研究部

デザイナにとって使い易いモデリング環境として，言語記述と対話操作を融合したモデラ，FUSION/MODELERを開発した。本稿では，従来のモデラの問題点を指摘し，我々の提案する方式の設計思想を述べる。また，幾つかの操作例を通してモデラの概要を説明する。

本モデラでは，オブジェクト指向のパラダイムを使ってモデラの汎用化，再利用化を実現した。また，FUSION/MMIの構造化ウインドウを用いて，CSG モデリングを絵言語で記述する。ウインドウ構造とモデル構造を対応付けることによって，直感的で容易なモデリングができると考えられる。

## A Integrated Modeler for Computer Graphics : *FUSION/MODELER*

*Koichi Murakami, Kazushi Hayashi*
Computer-based Systems Laboratory, Fujitsu Laboratories LTD. Kawasaki
1015, Kamikodanaka, Nakahara-ku, Kawasaki, 211, Japan

This paper describes *FUSION*, a modeling system for computer graphics that intergrates the two approaches, description by language and interactive operation. We point out the problem of the conventional modeling scheme, and present our design concept of this system. The examples demonstrate this system is useful especially for the designer who is not familier with the computer operations.

With object oriented paradigm, generic objects are defined and can be used for the various types of instances. Thus, the productivity of the modeling can increase. The CSG operations are specified using picture language that takes advantage of the structured multi-window environment *FUSION/MMI*. With window-based environment the designer can construct the model in a very natural manner.

## 1．はじめに

コンピュータグラフィックスにおけるモデリングの多くは，計算機に不慣れなデザイナによって行なわれる。彼等にとって効率の良いモデリング環境を構築することがＣＧの産業としての発展にとって重要である。本論文では，従来のモデリング方式の問題点を考察し，我々の提案する方式の思想を述べる。また，これに基づくモデリングシステム FUSION の概要を操作例を通して説明する。

## 2．ＣＧモデリングの問題点

最初に，ＣＧモデリングの問題点・要求事項を二つの観点から考察する。

### 2．1　真の統合システム

ＣＧには多様な処理方式が要求される。(1) CSGとB-rep，あるいは polygonと曲面等のデータ構造，(2)言語型と対話型のモデリング方式，(3)キイフレームとスクリプトの動画記述，等の異種の方式が種々な場面に用いられる。魅力的な映像を制作するためには，それらを複合的に使うことが必要である。例えば，人工物であるビルの背景に山を設定する場合には，CSG モデラで作った建築物とフラクタル生成した背景，あるいは写真を合成することが考えられる。この場合は，CSG モデラ，フラクタル生成，画像入力，画像合成の処理が必要となる。

これらの複数の処理を統合する一つの方法は，幾つかの処理方式を何々サブシステムとして構築し，ファイルを介在として結合することであろう。しかし，ユーザはシステムの切り換えの度にデザインに直接関係のないコマンド（例えば，ファイル管理）を操作しなければならない。これでは，デザインの思考の中断を起し，使い勝手の良いシステムとは云えない。望まれる統合化とは，デザインに関する全ての操作を一つの環境として実現することである。我々は複数の処理を統合する枠組みとしてFUSIONを考えた。

### 2．2　言語型と対話型のはざまで

モデラを記述方式の観点から見ると，言語と対話による二つの方式がある。どちらか一つの枠組みで実現していたのが従来の方式であった。しかし，それらの方式が共存して使われるのが望ましい場合がある。

CSG モデリングは一般に木構造として表現されるので，言語型の記述が適当である。しかし，CSGモデリングを中心的な枠組みとしている方式でさえ，対話的な操作が必要となる場合が多々ある。例えば，筆者が開発したアニメーション用言語 MEG［1］では，質感属性の記述も関数呼び出しで行っていた。ところが，質感の指定には十個以上のパラメータが必要であり，これを引数として記述することは難しい。これは対話的な指定が相応しい例である。この例から分るように，言語記述と対話操作をそれぞれの特徴が発揮される部分で共存して使う方式が有効であると考えられる。次に，それぞれの方式の相応しい適用部分を調べるために，我々のモデリングに対する要求事項とそれぞれの方式の特徴を考察する。

・モデルの汎用性・再利用性

モデリングの生産性を高めるためには，モデルの汎用性，再利用性が要求される。この要求事項は，オブジェクト指向の枠組みに PartOf の関係を導入した概念を用いることによって満足されると考えられる。オブジェクト指向のパラダイムではデザイナは物体（object）のクラスを定義する。クラスは抽象的な汎用部品と考えられる。クラス対してパラメータを与えて，種々な具現物（インスタンス）を生成することができる。またクラスの定義で別のクラスを部品として使うこともできる。更に，色や質感の属性値を継承の概念を使って記述することができる。

・操作性

しかし，クラスの定義は抽象的な記述であるので必然的に言語的な記述となる。これは操作性と矛盾する。デザイナにエディタを使わせて字面を入力させるのは酷と云うものである。

一方，対話型のモデリングではインスタンス自身を作ると考えられるので，実際に物（映像）を見ながらモデリング操作をしていることになる。しかし，モデルの修正や拡張が最大の問題となる。従来の対話型モデラでは，モデルの最終結果しか残らないのでモデルの部分的な更新はできなく，その場限りのモデリングと云うべき物であった。これはモデリングの生産性を著しく阻害する原因である。

・融合

我々は，言語記述と対話操作の特徴を取り入れた二つの方式の融合を試みる。また，狭義のモデラに留まらずにＣＧで行う複数の処理の融合を考える［2］。ここで融合と云う言葉を使ったのは，２．１で述べたように単なる統合では済む問題ではないという主張が入っているためである。次章では，我々の考える融合化のアプローチについて議論する。

## 3．FUSIONの設計思想

基本的なアプローチとして，モデリング処理を分解して言語記述と対話操作のそれぞれの方式の特徴が発揮される場面にそれを使う。

### 3．1　二つのレベルの記述

まず，モデルの記述を概念的に二つのレベルに分ける。大局的レベルではモデルのマクロな構造関係を与える。物体間の関係はオブジェクト指向で，物体内の部品の構造は CSG表現で記述する。部品の結合に関しては，最初から最後まで対話操作でモデリングする方式に比べて CSGによる構造記述の方が優れている。この記述は必然的に言語（的）な記述となり，モデリング過程が残っているために修正や更新ができる。

しかし，限定された種類のプリミティブを使って複雑な形状を表現することは難しい。これを改善するためにユーザにプリミティブを定義させ，これを CSGモデリングの枠組みで使う。ポリゴンを使って形状を定義する場合には，明らかにグラフィク操作が適している。この様な一つのプリミティブの定義の変更はモデリング全体に対して影響は少ないので，局所的なモデリングと考えることができる。対話操作をこのレベルでの記述に適用する。

説明のために二つのレベルを強調したが，これらは明確に分離されているのではなく FUSION/MMI ［3］を用いて自然な形で融合される。CSG 表現のなかで対話的に定義したプリミティブを使うことによって複雑な形状を容易にモデリングすることが可能になると考えられる。

局所的な場面は対話的なグラフィク操作だけで使われる分けではない。フラグ・選択子・値をキイ入力する代りに，Macに見られるような操作盤をエミュレートしたパネルを使ってマンマシンインタフェイスを向上させる。マウスを使って，ボタンを押したりスライダを動かして殆どの操作を行うことができる。

### 3．2　フォームを用いた構造記述

局所的な対話操作以外にも，物体の構造を記述する CSG表現を対話的に行うことを考える。我々はモデルの構造記述に絵言語を使うことを考えた。FUSION/MMIの構造化ウインドウを用いると CSG記述の木構造を対話的に入力することができ，モデルの構造をウインドウの構造に対応させてモデリングすることが可能となる。

プリミティブ，集合演算子，変換演算子等の言語の基本要素に個々のウインドウを対応させる。このウインドウをフォームと呼ぶ。フォームは，(1)パネル，(2)字面を入力する記述欄，(3)フォームの構造を記述するための下

位フォームのスロット，(4)必要に応じて存在するグラフィク操作用のサブウインドウで構成される。Class フォームを図1に示す。パネルとしてload,save のボタン，記述欄としてSizeParameter,InnerDefinition ，スロットして5種類のアイコンがある。また，フォームの種類を表1に示す。

デザイナは下位フォームを上位フォームのスロットに次々と結合してフォーム間の構造を与える。この操作がモデリングと等価になる。また，全てのフォームにはリンクされるべきフォームが文法として決っているので，誤ったリンク等の文法エラーはモデリング時に検出される。

フォーム記述欄には，プリミティブに関してはその大きさ，変換演算子に関しては変換量等，そのフォームが表す詳細な情報を記述する。

3．3　物としての性質

プリミティブは物と考えられる。この考え方を発展させると，物体にある演算を施した結果も物と考えるのが自然である。例えば，物体を変換した結果，あるいは複数の物体を結合した結果も物と考える。このように考えると，下位フォームの表す物を変換するフォームも物と考えられ，フォーム自身を物と見なすことができる。

フォームに物としての性質を持たせることによって，そのフォームに名前を付けることができる。また，物であるのでそれを見ることができなければならない。フォームのメニュー選択によって，それ自身の上にそのフォームの下位構造の物が線画表示される。この概念によって，物とフォームの関係が直感的に把握することができる。図2に複数の物体を結合しているCSG フォームの持つ映像を見る手続を示す。

3．4　仮のインスタンス

FUSIONでのマクロ的なモデリングではクラスを定義する。しかし，この概念は矛盾を含んでいる。抽象的な定義を行っている最中にはインスタンスは存在せず，物（映像）を見ることができない。対話的操作を実現するためにはインスタンスを生成しなければならない。FUSION/MODELERでは仮のインスタンスと云う概念を導入する。クラス変数にデフォルト値を与えてインスタンスを生成する。この機能によって定義の最中でも映像を見ることができるようになる。

3．5　シンボリックな位置指定

モデリングにおいて変換は重要な操作である。人工物のモデリングにおいては，結合される部品の間に抽象的な点の関係があると考えられる。例えば，机は天板と脚で構成されるが，部品の間に「脚の上の点と天板の下の点が一致する」という関係がある。FUSIONではこの記述を用いて変換操作を行う。これによって物体の大きさに依存しない抽象的な記述ができる。例えば，図3の例では，「物体A の"UPPERRIGHT"の点とプリミティブB の " LEFT + BOTTOM" で指定された点が一致する」と記述することで，自動的な変換操作を行うことができる。

## 4．FUSION/MODELERの操作

FUSION/MODELERではオブジェクト指向の枠組みを取り入れるように拡張されたCSG モデルをフォームの階層構造を使って記述する。ここでは，操作例を通してモデリング方式の概要を紹介する。

4．1　クラス定義

クラス定義では一般的に以下の記述を行う。

(1)そのクラスで使う部品をプリミティブ，部品フォームを使っての生成。

(2)それらを Link フォーム，rotateフォームを使っての変換。

(3)それらを CSGフォームを使っての結合。

(4)任意のレベルでの属性の記述。

典型的なクラスの記述を図4に示す。

クラス変数はサイズやその物体の特性を表すためのパラメータとして使われる。この変数，あるいは変数を使った式を，そのクラスのスコープ内のフォーム記述で参照できる。クラス変数を部品に対するパラメータとして使うことによって，パラメトリックな汎用部品を記述することができる。今，クラス Body のクラス変数として Width を定義する。ここで使う部品のパラメータとして "Width * 2"のような記述ができる。また，この変数は，変換量の指定にも使ことができる。

図1-A では二つのフォームが縦に並んでいる。下のDatabaseフォームでこのクラスのデータベース上の位置を指定する。Databaseフォームをクラスフォームに結合 (図1-B)することによって，指定された場所にこの定義が保存される。Databaseスロットがリバースされていることから，Databaseフォームが結合されたことが分る。可変個のクラス変数，位置シンボルを定義するために，SizeParameter 欄のadd とdeleteのボタンをクリックすることによって，記述欄が増減する (図1-C)。クラスフォームの下位フォームには，定義する物体フォームが接続される。また，そのクラスに対するデフォルトとしての色フォームと属性フォームを結合できる。

4．2　フォームの操作

必要なフォームは，画面上のCommonMenue をマウスでクリックすることによって生成される。図5はフォーム間のリンク操作を示している。図5-A の Primitiveフォームがその上の Link フォームに結合されて，空であったスロットにプリミティブのアイコンが入る (図5-B)。同様に Link フォームがRotateフォームに結合された様子を図5-C に示す。ここで，Rotateフォームでは，回転角，回転中心の座標系等パネルを使って指定する。

図6はプリミティブを記述するフォームである。デザイナはName, Width等の太字の後に，対応する値を書く。球，板，立方体，楕円球，円柱，楕円柱，円錐，双曲体，放物体の9種のプリミティブは，メニューで選択される。図6-A はプリミティブとしてBOX が選ばれていることを表している。ポップアップメニューを使って，別の種類のプリミティブに変更する。( 図6-B)。選択したプリミティブの種類に合せて，サイズパラメータの欄が自動的に変化する (図6-C)。

4．3　位置シンボル

前述したように，FUSION/MODELERでは部品の抽象的な位置をシンボリックに指定して大きさに依存しない変換を行う。プリミティブの位置シンボルとして TOP, BOTTOM, RIGHT, LEFT, FRONT, REAR, CENTERの7個が決っている。デザイナの定義するクラスのそれは，クラスフォームで指定する。クラスの位置シンボルを，使用したプリミティブや部品の位置シンボル，定数やクラス変数を使ったベクトル値，及びそれらを使ったシンボル式を用いて定義する。

Linkフォームを用いて接合点の結合ができる。このフォームに記述された移動物体の接続点と接続すべき相手の接続点から自動的な変換を行う。接続点の指定にはシンボル式を使う。

4．4　CSG 演算

物体間の集合演算は CSGフォームで記述する。このフォームは直感的な演算を行うためにアイコン自身をオペランドとして扱っている。図7-A では結合される二つのフォームと CSGフォームがある。 CSGフォームの中段は境界の無いスロットである。ここに下位フォームを結合する。図7-B では，二つのプリミティブフォームを CSGフォームに結合した状態を示している。ここで，LeftPartはフォームを残したまま結合して，サイズ記述欄の更新を行なえる状態にしている。また，CSG フォームのアイコンにもフォームで記述した名前が表示されている。

CSG フォームではアイコンの間にポップアップメニューを出して，＋，－，&，（，）の演算子を指定する。図7-C では，前の二つのフォームの和集合を指定している。

## 5．むすび

複数の処理の協調処理を実現する枠組みFUSION/MMIの上で実現した，言語記述と対話操作を融合したＣＧモデラ FUSION/MODELER の思想や概要を述べた。

今後の課題として，条件や繰り返し等の言語の制御機構，視覚化フォーム上での対話的な操作の実現がある。

［謝辞］本研究に議論等で御援助して下さいました石井システム研究部長，白石室長に感謝します。

［参考文献］

［1］村上：”３次元アニメーション用モデリング言語”，日経ＣＧ，10月号，PP.146-158，1986
［2］村上：”コンピュータグラフィックスのための統合環境”，情報処理第34回全国大会，PP1881，1987
［3］林：”構造化マルチウインドウを使ったCG統合環境：FUSION/MMI”，集中研究会資料，1987

表1．フォーム一覧

| フォーム | 機能 | 下位フォームスロット |
|---|---|---|
| クラス | クラスの定義 | ＤＢ、光源、色、属性、物体 |
| 部品 | 部品の宣言 | ＤＢ、色、属性 |
| プリミティブ | プリミティブの宣言 | 色、属性 |
| ＣＳＧ | 物体間の集合演算 | 物体、色、属性 |
| リンク | 物体の位置を指定 | 物体 |
| 回転 | 物体の回転を指定 | 物体 |
| 色 | 色の指定 | |
| 属性 | 属性の指定 | |
| ＤＢ | データベースファイル | |
| 光源 | 光の色と位置指定 | |
| カメラ | カメラの位置指定 | |
| 映像生成 | 映像の生成 | トップレベルのインスタンス |

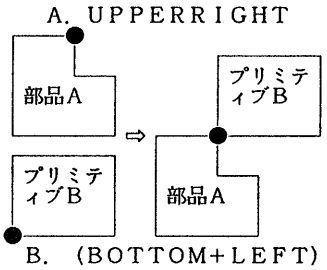


図3．シンボリックな位置指定

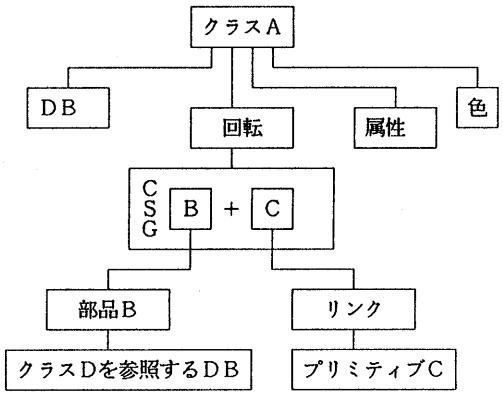


図4．クラス定義の例

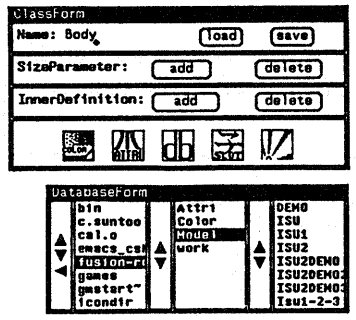


図1-A

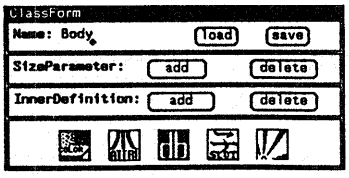


図1-B

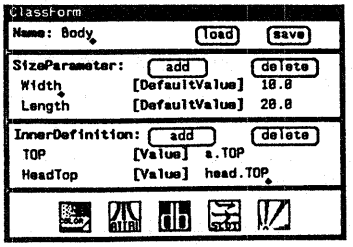


図1-C

図1　クラスフォームの例

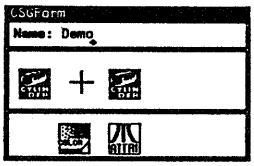


図2-A

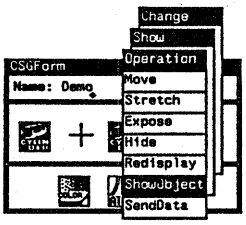


図2-B

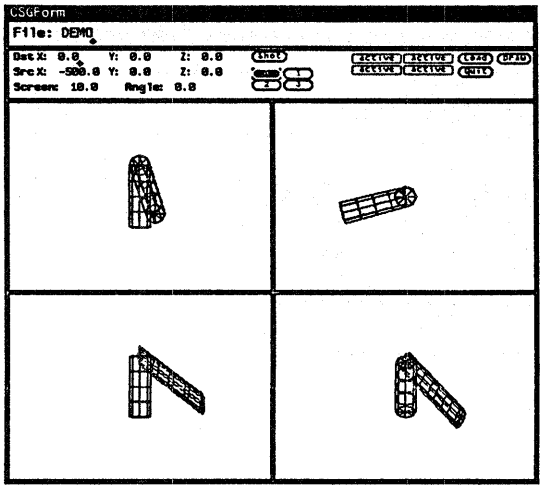


図2-C

図2　線画表示操作

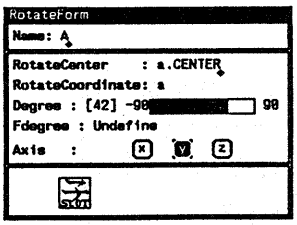


図5-A

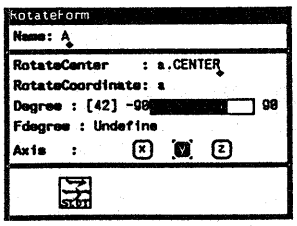


図5-B

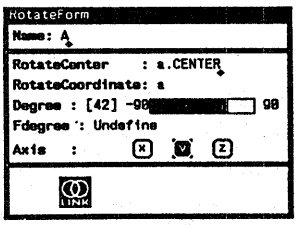


図5-C

図5　フォームの入れ子

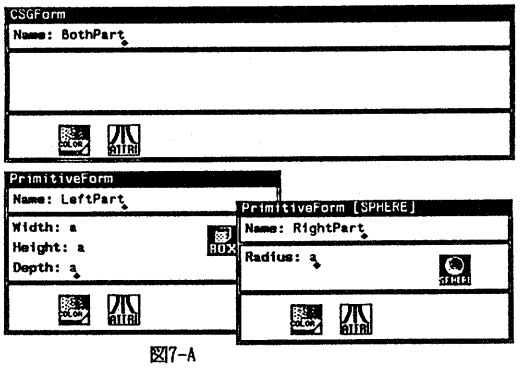


図7-A

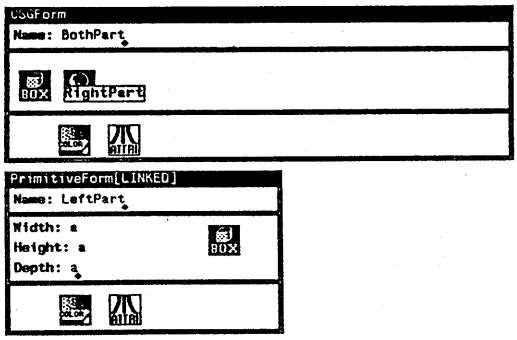


図7-B

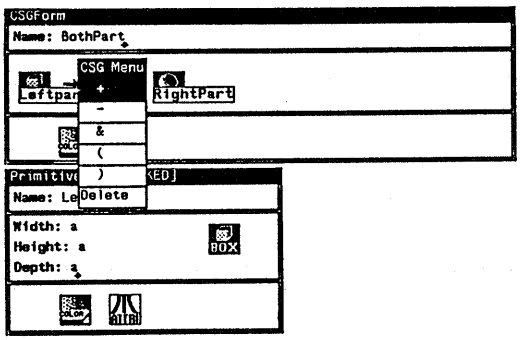


図7-C

図7　CSG フォームの例

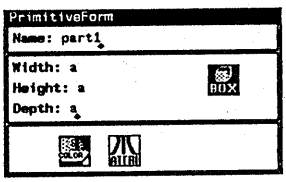


図6-A

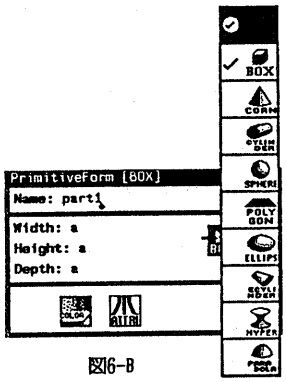


図6-B

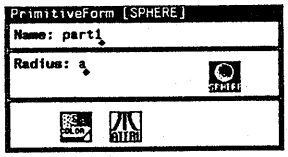


図6-C

図6　プリミティブフォームの例

## 討論

１１．構造化マルチウィンドウを使ったＣＧ統合環境　　林　（富士通）
１２．（言語＋対話）型ＣＧモデラ　　村上（富士通）

川合：親子関係のウィンドウの表示で、親の方には子どものアイコンが入りますが、子どものウィンドウには自分のアイコンは入らないのですか。
村上：子どもの方からはどの親につながっているかは分かりませんが、親からはすぐに分かる仕組みになっています。
秋本：モデラのやり方ですが、１つの部品が複数の親にリンクされるようにはできないのですか。
村上：現在、インスタンスは１つと考えていますからそれはできません。コピーというオペレーションがありますから、それを使って複製します。
川越：たくさんのウィンドウがある場合に、アイコンを使って親子関係を表現したのでは、一目見てどういう構造か分からないのではないでしょうか。実際に線を引いても良いのではないでしょうか。
村上：必ずしもいつも階層関係が見えている必要はないと思います。オーバーラップ機能を重視して、有限のスクリーンにどれだけの情報を埋め込むかは、デザイナーの意志に任せたかったのでこのようにしました。
福井：画面全体が上下に平行移動して広く使えれば良いですね。
村上：そうすれば良いかもしれませんね。
原田：デザイナーが階層構造を作ったときに、リアルタイムにモデラの表示が変わるようにした方が使いやすいのではないでしょうか。
村上：確かにそうですが、裏でモデラーが動いているので実現は難しい。
原田：これだけウィンドウを開いていて、ＳｕｎＶｉｅｗでパフォーマンスはどうですか。
林：　１４０クラスですとほぼ問題なく使えます。スワップ領域が少ないと苦しいです。
坂下：今はX-window などを使えば解決する問題ですね。
村上：FUSION/MMIでは今後X-window 使って行くつもりです。
川合：シンボリックな位置指定で、オペレーションには何が使えるのですか。
村上：ベクトル値とベクトル値に対しては＋と－で、ベクトル値とスカラー値に対しては＊と／です。
川合：ｍａｘなどの非線形なオペレーションまでは考えていませんか。
いくつかの部品の一番高いものの上に板を置く場合などに有用ですが。
村上：そうですね。あるといいですね。
辻堂：TOP とかBOTTOMなどの位置シンボルは何種類ぐらいあるのですか。
村上：プリミティブに関しては７種類で、プリミティブを組み合わせて作ったクラスに関してはデザイナーが自由に定義できます。
守屋：モデル構造としては物、ウィンドウ、データ構造の３階層のようですが、もっと直接物を扱えないですか。また、元々３次元の物を２次元のウィンドウで扱うのは少し気になります。
村上：ウィンドウは物の１つの側面を表現しています。言語と対話の融合ということで、図の中での変更が言語記述に反映されるようにしたいと考えています。
川越：１つのウィンドウの構成は名前とパラメータとリンク関係だけですか。
村上：そうですね。リンク関係はウィンドウによって特有のものがあります。