4L－3

空間パターン情報処理のための オブジェクト指向型言語とその応用

渡辺弥寿夫、小松孝司、中村公亮
金沢工業大学

1 はじめに

現実世界における立体や画像などの空間パターン情報を扱う場合、情報そのものの表現、すなわちモデル化をうまく行う必要がある。しかしパターン情報を扱うためのソフトウエア環境は、必ずしも十分ではない。

オブジェクト指向概念のプログラミングにより、対象とする問題の物理的実体や概念を自然にモデル化することができる。これは、ユーザに解くべき問題の記述のみに専念させ、問題解決はシステムに任せることができるという利点があるが、純粋のオブジェクト指向型言語にはオブジェクト間の関係や動的振舞いを表現する方法が用意されていない。このため、制約指向を取り入れるアプローチがなされている [1][2]。ところが、その制約は、グラフの同型問題のような制約充足問題 [3] として知られているものであって、離散的な構成要素間の計算領域における関係であり、連続的なシステム、すなわち、微分方程式や積分方程式で与えられるようなシステムを扱うことはできない。

そこで、離散的のみならず連続的な空間パターン情報の表現方法を持ち、現実における力学、電磁気学、光学などの法則を表す物理的制約と、空間一意性などの幾何的制約を記述する方法を用意した制約オブジェクト指向型言語 [4] のトランスレータの作成を行い、立体や空間を扱う VR や CAD などのアプリケーション開発を容易にすることを試みる。

本稿ではその言語から C++へのトランスレートを行う際の内部での処理方法について述べる。

An Object-oriented Language for Spacial Pattern Processing and Its Application

Yasuo Watanabe,Takashi Komatsu,Kimiaki Nakamura
Kanazawa Institute of Technology

2 言語仕様

我々が提唱する言語は以下のように既存の言語 C++に、新たに立体や空間、制約を表現するための枠組を用意したものであり、ConstraintC と呼ぶ。

OBJECT:立体定義
　立体の形状や物理的実体を記述
SPACE:空間定義
　制約、立体名を記述
WORLD:仮想世界定義
　空間、視点を記述
CONSTRAINT:制約定義
　空間や立体に対する制約を記述

実際に制約付きの空間を定義し、その空間内に立体を配置した場合の記述例を以下に示す。

```
OBJECT Dumbbell{
  BALL ballA(6[cm]),ballB(6[cm]);//半径
  CYLINDER body(3[cm],20[cm]);//半径, 高さ
INITIALIZE:
  ballA.Position(0,0,10[cm]);//座標
  ballB.Position(0,0,-10[cm]);
  Dumbbell=ballA|body|ballB;//集合演算
};
WORLD {
  Dumbbell a;
CONSTRAINT:
  Gravitation((0,-1,0),4.0[m/s^2]);//重力
  SpatialUniqueness;//空間一意性
  Newton;//ニュートンの法則
INITIALIZE:
  a.Position(0,3[m],0);
};
```

ユーザが任意に制約を記述した場合の例を示す。

```
CONSTRAINT Fall{
  OBJECT;
RULE:
  positionY=velocity*sin(angle)*TIME
            -1/2*gravitation*(TIME^2);
};
```

この言語の特徴として、

(1) アプリケーションユーザは既に言語側で与えられる制約により現実世界に反するモデル化を行う事無

く、プログラムの作成を行うことが可能である。これは、現実のモデルに矛盾する制約や違反する操作を定義した場合、誤りを適切に指摘し、エラーを防ぐ機構である。
(2) 仮想世界での空間パターンを、構造的性質も含め、幾何的情報、物理的情報とそれらに対する操作をオブジェクトとしてミクロなレベルからマクロなレベルまで階層的に表現できる。
(3) 実世界での単位を直接プログラム中に記述出来ることにより、ユーザが単位変換の作業を行うことがなくなり、負担を減らす。

3 立体と空間の管理

この言語ではユーザが立体を表現するには、基本立体の集合演算によって立体を表現するCSG表現によって行う。しかし、動的な場合に対して対応するため、内部で境界表現に変換を行い、CSG表現と境界表現の二つを持ち管理、演算を行う。また、言語側でメモリ管理の機能を持つことにより、効率よくメモリを使用することができる。

立体の処理: 立体が定義されると、形状としては、(1)CSG表現による管理、(2)境界表現のデータ作成、パラメータとしては、(1)単位変換とその型チェック、(2)幾何的制約としての空間一意性の検証、これらを内部で行う。また、それらデータは構造体として格納し空間内に配置する際の空間オブジェクトに渡される。

空間の処理: 空間が定義されると、空間内の立体の配置、立体のワールド座標変換、その空間内に存在する全ての立体に対する制約の宣言が行われる。

4 制約の管理

実世界に存在する物体には、様々な幾何的制約、物理的制約が課せられている。これらをユーザが全てを把握することは難しいため、言語側でライブラリとして用意する。

制約は、時間、イベントを軸として動作し、全てのオブジェクトの動作を受け持つ。記述では、その制約を行なう際の条件式、または動作する時間の間隔、メッセージを送受するオブジェクト、実際の制約として関数式を必要とする。また不特定多数のオブジェクトに対しメッセージの送受を限定するために、オブジェクトのグループ化を用意する。

ユーザのプログラムは制約の実行を中心としたプログラムへとトランスレートされる。

5 単位の変換

単位は[] で囲み記述を行う。積は単位を繋げて記述を行うが、曖昧な場合にはピリオドで積の記号とする。例えば、[mg] はミリグラム、[m.g] メートルとグラムの積を表す。商は"/" で記述する。また累乗は"X^n" の形式で記述する。

実際に内部での単位の処理について述べていく。トランスレータ側では実際に記述された変数、数値の他に単位を次元化して格納する。単位の次元は内部では次のように表現されている。

(長さ, 質量, 時間, 電流, 光度, 角度)

単位を読みとり、数値をMKS単位系に変換し、トランスレート後のプログラムには実際の数値だけを渡す。また、パラメータ代入の際にも各パラメータに単位が設定されており、それが一致しない場合にはエラーとして検出を行う。

6 おわりに

制約を付加したConstraintCを設計し、C++にトランスレートする際に行なわれる内部処理について述べた。今後、本言語を画像パターンへ適用するとともに、改良を行い、記述が容易で信頼性の高い処理系を作成していきたい。

[参考文献]

[1]Bing Liu,Yuen-Wah Ku:ConstraintLisp:An Object-Oriented Constraint Programing Language , ACM SIGPLAN Notices,Volume.27,No.11 , November , pp.17-26(1992).
[2] 中島震:制約伝播機構を内蔵するオブジェクト指向言語:COOL, 情報処理学会論文誌,Vol.30,No.1,pp.101-107(1989).
[3] 甘利俊一監修, 大原育夫著:認知情報処理, オーム社(1991).
[4] 宮下益幸, 渡辺弥寿夫:制約によるパターン情報処理のためのオブジェクト指向型言語の一考察–立体モデルを例に– , 電気関係学会北陸支部連合大会,pp.365-366(1992).