交通流シミュレーションにおけるクラス概念の導入について*

*Object Oriented Design on Traffic Flow Simulation**

田頭　直樹**・中辻　隆***

*By Naoki TAGASHIRA**, Takashi NAKATSUJI****

1. はじめに

一般にシミュレーション技法を用いて交通流をモデル化する手法は、

（1）既存アプリケーションの使用

（2）汎用シミュレーション言語を用いた開発

（3）独自開発

に大別できる。しかし、（1）は用いられているロジックを理解することができず、ユーザーは入力値と出力結果のみから判断するしかなく、仕様の変更・改良が不可能である。（2）は交通流の表現を専門としていないため、複雑な交通現象を表現するには問題がある。（3）は膨大な時間と労力を必要とし、他のプログラムとの共通性も低い。このようにそれぞれの手法には長所と短所があり、必要に応じ慎重に選択する必要がある。これらの問題点は、近年のシミュレーション解析の需要増大にも関わらず、交通流シミュレーションの共通のプラットホームといわれるものがないことに起因するところが大きく、そのアーキテクチャの整備が必要とされている。

そこで、本研究では、新しくオブジェクト指向言語のクラス概念を用いたモデル化手法を開発し、その延長線上にある「交通流シミュレーションの共通のプラットホーム」となる交通流専用シミュレーション言語のアーキテクチャを考察した。

2. オブジェクト指向

オブジェクト指向では、データ（属性）と手続き（メソッド）を一体化したオブジェクトを単位としてソフトウェアを開発する。またオブジェクト指向には、抽象化，カプセル化，継承，状態の４つの基本概念がある。それぞれ簡単に説明する。

（1）抽象化

現実世界のモノをオブジェクトとしてシステム上に投影する際、問題の重要な側面もしくは注目したい問題の側面のみを強調してモデル化する。このことを「抽象化」という。また、現実世界のモノをオブジェクトとして抽象化するという意味の他に、同じ性質を持つオブジェクト群をさらに「クラス」として抽象化する、そして、クラス群をもう一段上のさらに抽象的なクラスとして抽象化する、という意味もある。モノ中心に思考単位を変換することにより安定したモデルを構築することができ、モノ中心のフレームワークはシステムの変更に対して、局部的な変更だけで対応できる。またモノの構造に着目すれば、必要な機能や動作が推定しやすくなり、実世界の重要な側面だけをモデル化することは、注目すべき焦点を明確化できる。

（2）カプセル化

オブジェクトが保有するデータに外部からアクセスする場合は、手続きを通してのみアクセスできるようにする。これを「カプセル化」という。カプセル化により、オブジェクトの利用者はオブジェクトに定義された手続きの方法さえ知っていれば、オブジェクト内部のデータ構造あるいは手続きの実装を知らなくてもよい。逆に、オブジェクトの仕様と実装の分離は、オブジェクトの提供者がオブジェクト内部の変更、例えば内部データ構造や手続きの実装の変更を行ってもその影響が利用者に波及しないので、変更が容易になる。

（3）継承

上位クラスの概念を下位クラスが引き継ぐと共に下位クラスでは上位クラスに存在しない新たな性質が追加できるようなメカニズムを用いている。この

*キーワード:交通流シミュレーション、クラス概念、
共通のプラットホーム、アーキテクチャ

**正員、工修、㈱建設技術研究所

***正員、工博、アジア工科大学

（北海道北区北１３条西８丁目、
TEL011-706-6215、FAX011-706-6216）

ことを、「継承」という。継承における上位クラスのことをスーパークラス、下位クラスのことをサブクラスという。(図－1) 継承というメカニズムを取り入れることにより、クラスを体系化でき、新たなクラスの定義が容易に行える。

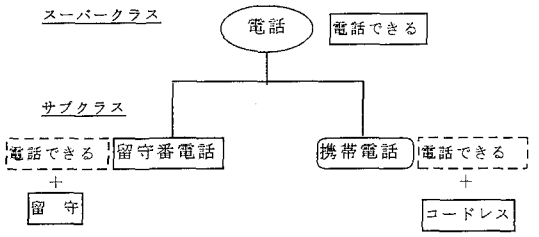

図-1 継承

（4）状態

オブジェクト指向では、オブジェクトに定義されている手続き群を利用できるかどうかは、特定の内部データにより判断される「状態」によって決まる。さらに、手続きの実行による状態の遷移を表現することで、提供者は利用者に対して確実に正しい利用手順を示すことができる。(図－2)

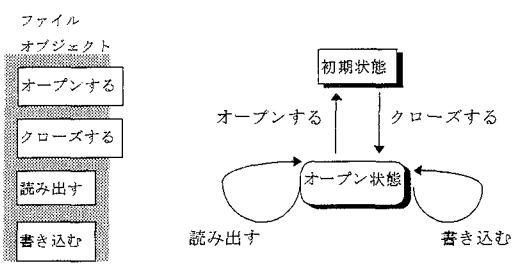

図-2 状態

状態という概念を導入し、状態遷移図をオブジェクトの提供者と利用者が、共通の取り決めとして明文化することにより、互いの責任と権利を明確にでき、双方に次のようなメリットが生まれる。

3. クラス概念の導入

（1）C++とクラス概念

オブジェクト指向で最も重要な要素であるオブジェクトは、クラスによって定義される。逆に言えば、オブジェクトを定義するものをクラスと呼ぶ。またC++では、クラスも整数型，倍精度型と同じようなデータ型の一種として考える。つまり、クラスはユーザー定義のデータ型として考えることもできる。

（2）データ型の遷移

最も原始的なデータ型として、整数型，文字列型，倍精度型などと言われる基本型がある。これらは、どのコンピュータ言語でも使用できる。次に登場したのが、それら複数の基本型データを集合体として定義できるいわゆる構造体と呼ばれるものである。そしてさらに進化して、複数のデータとそれらデータを操作する動作も一体化して定義できるデータ型が考え出された。これがクラスである。このようなデータと動作を一体化することにより、より実世界のモノをコンピュータ上でリアルに表現することができるようになった。

図－３では「person」をそれぞれのデータ型で定義した場合の具体例である。ただし、基本型では変数１つしか定義できないので、int型を宣言している。

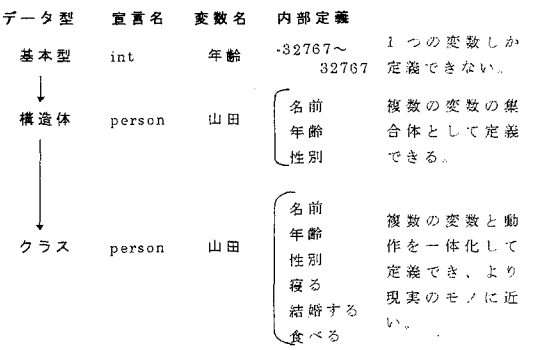

図-3 データ型の遷移

（3）クラスの構成

C＋＋では、クラスは予約語 class を用いて定義される。クラスの内部で宣言されたデータ、それを操作する動作をそれぞれメンバ変数、メンバ関数と言う。また総じてそのクラスのメンバと呼ぶ。private: の後

```
class CVehicle{
private:
    //メンバ変数の宣言          } 非公開部分
    int x, y;
    double rad;
public :
    //メンバ関数の宣言          } 公開部分
    void Move();
};
void CVehicle::Move(){
    x += s * cos(rad);          } メンバ関数
    y += s * sin(rad);          } の外部定義
}
```

図-4 クラスの定義

に宣言されたメンバは外部からアクセスできず、public:の後に宣言されたメンバはどこからもアクセスを許可される。一般にデータを表すメンバ変数は非公開にし、データを操作するメンバ関数のみ公開するようにする。これにより、メンバ変数はメンバ関数を介してのみ参照できるようになり、外部からの勝手な変更などを禁止することができる。図－４は、クラス名 CVehicle で定義している。また、メンバ関数は::演算子を用いて外部定義されている。

（５）交通流クラス群の定義

本研究では一般的な街路を対象とした交通流を表現するために、汎用的な数種類のクラスで構成される交通流クラス群を定義した。（図－５）

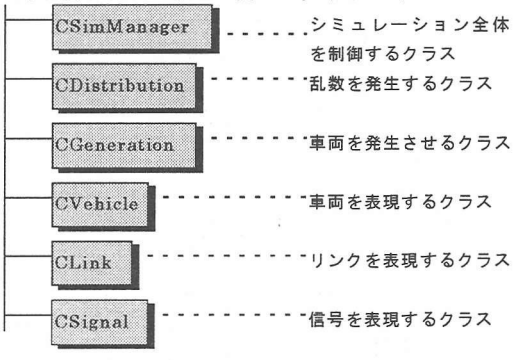

図-5 交通流クラス群

（６）サンプルプログラム

次に（５）の交通流クラス群を用いて作成した１信号交差点におけるミクロ交通流シミュレーションプログラムを説明する。

従来手法では、実現象を機能的な側面に注目してデータ主体に考えなくてはならなかった。しかし、本研究で作成した交通流クラス群を用いて交通流をモデル化すると、メンバ変数をすべてユーザーからアクセス禁止にしているので、実現象を動作主体で考えることができる。CVehicle クラスのメンバ関数では、リンク内を走行する場合は Move( )が、交差点内を走行する場合は左折，直進，右折それぞれに TurnLeft( )，GoThrough( )，TurnRight( )が提供されている為、今回のモデル化では車両の加速度や前車との距離を求めるルーチンなどを考えず、図－６のようにオブジェクトの状態により動作を決定するという考え方で設計することができた。従来のフローより簡便でより実現象に近い設計方法だと言える。

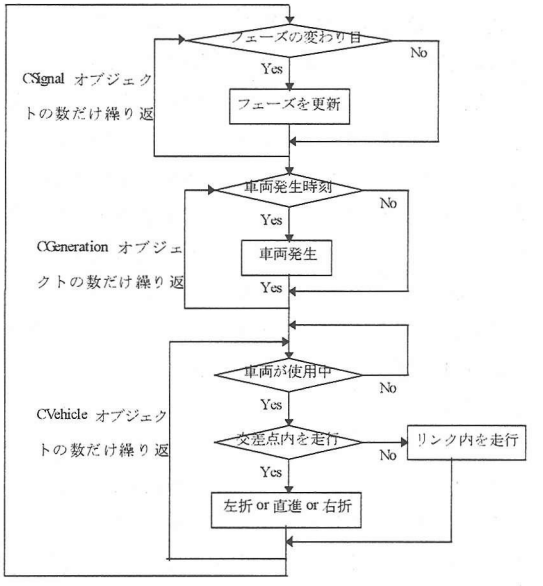

図-6 プログラムフロー

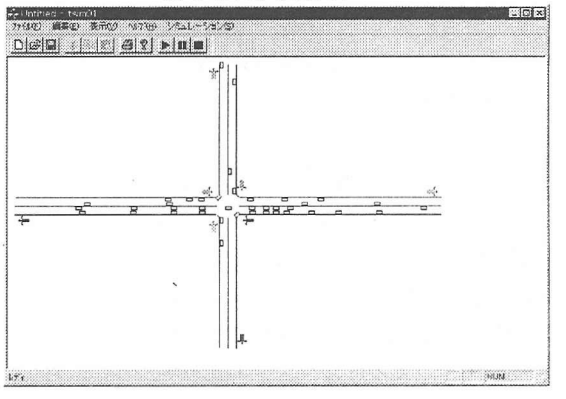

図-7 実行画面

4. 交通流専用シミュレーション言語

本研究で作成した交通流クラス群は、最終的に「交通流シミュレーションの共通のプラットホーム」となり得る交通流専用シミュレーション言語の開発を目的として作成した。そこで、ここでは交通流専用シミュレーション言語としての交通流クラス群について考察してみる。交通流専用シミュレーション言語としてのクラス群を TFFC（Traffic Flow Foundation Class）と呼ぶことにする。

（１）TFFC の内部構造

動作を表すメンバ関数を細分化しすぎるとクラス利用者のプログラミング作業は当然大きくなるが、一つの交通流から多様なモデルを開発することができる。逆にあまりに大別しすぎると、クラス利用者

の負担は軽減するが汎用性がなくなり言語としての価値が下がる。どの程度クラス利用者に負担させるかが難しい問題であり、実際に本研究で作成した交通流クラス群を定義するにあたって一番この点を苦慮した。ここで、オブジェクト指向の本来の設計を考えるとメンバ関数はそのクラスから生成されるオブジェクトの動作を表現するものであるので、CVehicle クラスのメンバ関数であれば、車両の動作を適切に表現していなければならない。メンバ関数の設計は重要でなおかつ非常に難しい作業なので、追従式なども含めて今後議論しなければならない。

（2）TFFC の外部構造

（1）では、各クラスの内部の構造であったがここでは各クラスの外観について考察する。交通流シミュレーションに求められるものの１つとして汎用性がある。多くの場合、特殊な交通条件でシミュレーション解析を行わなければならず、既存アプリケーションなどでは表現できないので、独自開発に頼らざる得ない。本研究で考える交通流専用シミュレーション言語は、まさにこのような場合利用されることを対象としている。よって、TFFC を最初から継承を前提として設計する。具体的にリンクを表現するクラスの構造を考えてみる。リンクを１つに規定してしまうことは不可能であるし、仮に規定したとしても存在するすべてのリンクを表現することはできない。そこで、図－８のようにリンクをクラス階層を用いて規定することを提案する。CLink クラスはどのようなリンクでも持っている必要最低限な性質しか存在しない。すべてのリンク系のクラスはこのクラスから派生する。これを一次クラスとする。次にこの一次クラスに１つだけ特長を付け加えたクラスを二次クラスとする。ここで、付け加えられる特長は、原則として１つである。その代わり、二次クラスは複数用意しておく。ここまで、TFFC として提供する。クラス利用者は提供されているリンクでは表現できない場合、多重継承により必要なクラスを定義する。また、直接 CLink クラスから派生することも可能である。

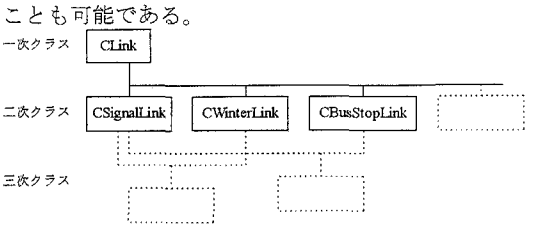

図-8 CLink のクラス階層

（3）クラス階層

実際に TFFC を設計した場合のクラス階層は、図－９ようになるであろう。ここで、すべてのクラスを MFC（Microsoft Foundation Class）の CObject から派生させているのは、オブジェクト配列の操作など便利なメンバ関数が提供されているからである。

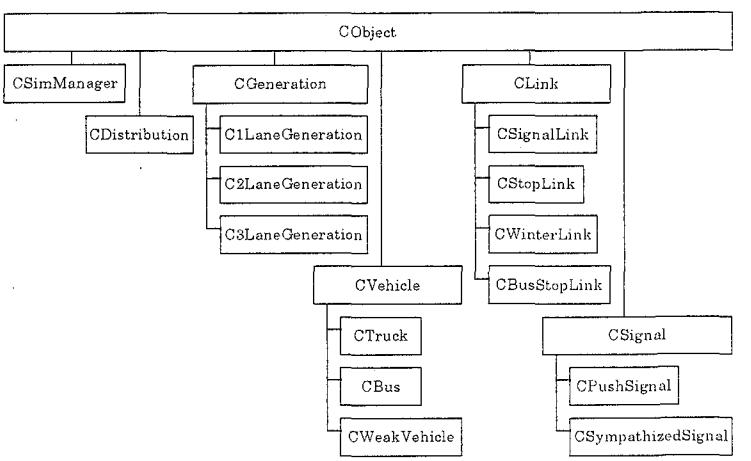

図-9 TFFC のクラス階層

5. まとめ

交通流が非線形であるということを考慮すると、現状の再現性から将来予測の妥当性を評価するのは、指標として不十分であると言わざるを得ない。やはり、どのようにモデルを構築したかが重要であると考える。そこで本研究では、「交通流シミュレーションの共通のプラットホーム」となる TFFC が必要であると認識した。この標準的な規格を利用して交通流をモデル化すれば、モデル同士の比較やモデルの改良も明確になる。しかし TFFC の規定には多数の団体による十分な議論を要し、また TFFC を標準規格として利用されるかどうかはどのように TFFC を公開できるかが大きな課題となるであろう。

[参考文献]「オブジェクト指向システム開発」 本意田 真一他（日経 BP 出版センター）「ラーニング C++」 N. グラハム（海文堂出版）