2J－4

事例プログラムの開発における オブジェクト指向と手続き指向との比較

中村智法　藤井　諭

松下通信工業（株）

1．はじめに

近年、オブジェクト指向を導入する動きが広がっている。しかし、オブジェクト指向を利用するのは、従来の手続き指向で業務を遂行している技術者である。したがって、オブジェクト指向への移行に際し、オブジェクト指向と手続き指向との差異を明確にすることは、方法論の教育を徹底することと並んで重要なことである。

本報告では、オブジェクト指向と手続き指向の双方で三次元迷路プログラムを試作した結果から、手続き指向とオブジェクト指向の比較を、設計上での考え方、C言語とC++言語での実装サイズという観点で論じる。

2．三次元迷路プログラムの概要

三次元迷路プログラムは、プレイヤーが、疑似的な三次元像で表現される地形（図2－1）および平面地図を参照しながら、迷路的な構造を持つ地図内を、ゴールを目指して進むプログラムである。

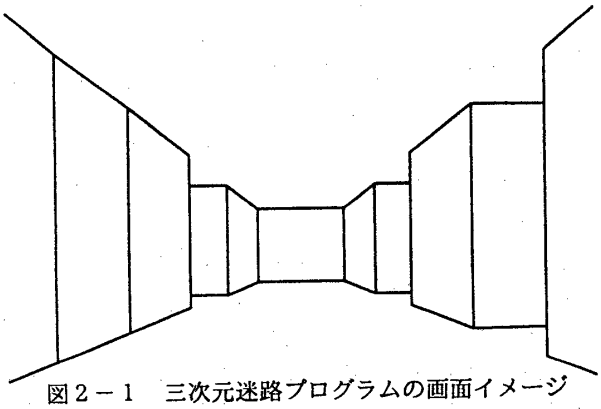

図2－1　三次元迷路プログラムの画面イメージ

三次元迷路プログラムは、三次元像処理部、地図処理部などから成る。各処理部について、以下に解説する。

2－1．三次元像処理部の概要

三次元像は、奥行き方向（プレイヤーの進行方向）に分割した階層で構成する。階層単位の処理は、表示座標が異なる以外、同一の処理になる。また各階層は複数の線分から構成し、各線分の表示または消去で描画する。

三次元像処理部では、各線分の座標設定、および地形をもとにした三次元像表示を行う。

2－2．地図処理部の概要

地図は、平面地図（縦方向×横方向）を高さ方向に重ねて構成する。地図の構成イメージを図2－2に示す。

地図処理部では、地図の作成と要素抽出を行う。

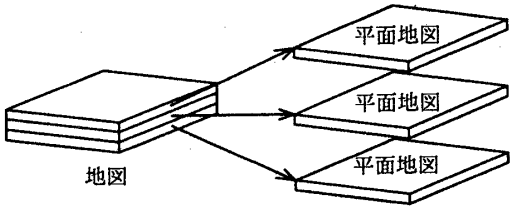

図2－2　地図の構成イメージ

2－3．その他

上記以外の処理部としては、プレイヤー制御処理部、地図と三次元像を結合する三次元迷路処理部がある。プレイヤー制御処理部では、プレイヤーの現在位置の変更（移動）、プレイヤーの向きの変更、平面地図の画面表示を行う。三次元迷路処理部では、プレイヤーの現在位置と向きを参照して、地図から取得した地図要素を組み合わせて地形を作成し、三次元像の描画を行う。

3．設計に関する比較

以下に、三次元迷路プログラムの設計について、手続き指向とオブジェクト指向との比較結果を述べる。ただし、オブジェクト指向に関するモデル図および用語は、OMT法[1]に基づく。

3－1．三次元像処理部

三次元像は、階層的に分割した線分の集合である。したがって、オブジェクト指向で設計すれば、階層的な集約（組立）の関係で記述することができる。三次元像処理部のオブジェクトモデル図を図3－1に示す。

Comparision Object−Oriented and Procedure−Oriented by Development of an exsample program

Tomonori NAKAMURA, Satoru FUJII

Matsushita Communication Ind. Co., Ltd.

図３－１　三次元像のクラス構成

このことから、構成が集約の関係になっている場合、手続き指向による設計とオブジェクト指向による設計を比較すると、オブジェクト指向は、全体の構成をクラスへの分割に反映した後、各クラスの中だけで必要な処理を設計できることから、手続き指向よりも優位であると考えられる。

３－２．地図処理部

地図は平面地図の集合であり、三次元像と同様、階層的な集約の関係で記述できる。地図処理部のオブジェクトモデル図を図３－２に示す。

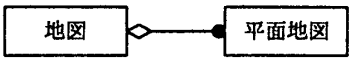

図３－２　地図のクラス構成

地図は複雑な迷路としたいため、地図の作成には特殊なアルゴリズムを用いた。そのアルゴリズムは、オブジェクト指向での設計、手続き指向での設計共に、複雑な手続きで実現する以外になかった。

このことから、処理アルゴリズム（データ構造を持たないアルゴリズム）の実現について、手続き指向とオブジェクト指向を比較すると、処理アルゴリズムは単一の手続きの中に隠蔽されるため、手続き指向とオブジェクト指向の間には差異がないと考えられる。

３－３．プレイヤー座標と地図座標の対応付け

プレイヤーの現在位置を変更するには、前方（の地図要素）との差分が必要である。また地形データを作成するには、前方／右方／左方（の地図要素）との差分が必要である。しかし前方／右方／左方はプレイヤー中心、現在位置の変更／地図要素の取得は地図中心の座標系である。

手続き指向による設計では、モジュール間での不整合を回避するためには、複数の差分を単一のモジュールから取得するしかなく、不要な差分を取得しなければならなかった。一方、オブジェクト指向による設計では、座標系の対応を単一のクラスにすることで、不整合を回避しつつ、必要な差分のみを取得することができた。

このことから、必要な情報のみを取得する処理の設計について、手続き指向とオブジェクト指向を比較すると、オブジェクト指向は、クラスの中に全情報を隠蔽し、必要な情報のみを取得できることから、手続き指向よりも優位であると考えられる。

４．実装サイズに関する比較

表４－１に三次元迷路プログラムのソースコード（ステップ数）および実行形式のサイズについてまとめる。

表４－１　三次元迷路プログラムの実装サイズ

使用言語	ソースコード（単位：Step）	実行形式（単位：Byte）
C	８８９	４８１７３
C++	１２１０	９０５７０

※C/C++はBorland社のBORLAND C/C++を使用

三次元迷路プログラムをＣ言語とＣ＋＋言語で実装した結果を比較すると、Ｃ＋＋言語の場合は、Ｃ言語の場合に比べて、ソースコードで約１．５倍、実行形式で約２倍になった。実行形式のサイズに制限がある場合、現状ではＣ＋＋の導入は非常に難しい。

５．まとめ

オブジェクト指向と手続き指向を比較した結果、処理アルゴリズムの実現に差異はないが、クラスに分割した後、クラスの中だけで必要な処理を設計できること、クラスの中に全情報を隠蔽して必要な情報のみを取得できることから、オブジェクト指向の方が優位であると考えられる。ただし実行形式のサイズに制限がある場合、Ｃ＋＋言語で実装すると実行形式が大きくなるため、Ｃ＋＋言語の導入は非常に難しい。

結論として、設計時にクラスの概念を導入するのであれば、実装には影響しないため、言語をＣ＋＋に限定する必要はない。したがって、実行形式のサイズやコンパイラなどに制限がある場合、設計にオブジェクト指向におけるクラスの考え方を導入し、実装は現用の言語で行うのが有効であると考えられる。その後、クロスコンパイラなどの環境が整った時点で、Ｃ＋＋などのオブジェクト指向言語まで含めて、オブジェクト指向を導入すべきであると考えられる。

参考文献

１）「オブジェクト指向方法論ＯＭＴ」、Ｊ．ランボー／Ｍ．ブラハ／Ｆ．エディ／Ｗ．ローレンセン著、羽生田栄一監訳、プレンティスホール／トッパン、１９９２