3H−4

3次元CGモデルのOODBMSによる実装

金子邦彦, 黒木進, 近藤祐介, 牧之内顕文
(九州大学工学部)

1. はじめに

近年,3次元コンピュータグラフィックス(3次元CG)がさまざまな分野に普及しつつある.一方,3次元CGに十分な性能を持ったワークステーションが安価になっている. 3次元CGの作成には,かなりの労力がかかる上に計算機,プログラミングおよびCGの知識が要求される.

我々は,こうした3次元CG作成の際の労力を低減し,計算機の初心者にも比較的容易にCG作成が可能になることを目的として,新しいアニメーションシステムを設計,試作中である.本システムでは,3次元CGデータ管理の道具として,OODBMS を用いることにした. OODBMS を用いると,データとともに,データに伴う処理(メソッド)をデータベース内に置くことが可能になる.従って,本システムを用いることで,計算機やプログラミングの知識の少ない利用者でも簡単なデータベース操作を用いてメソッドを呼び出すことができ,容易にコンピュータアニメーションを作成することが可能となる.

我々は,3次元CGのデータや処理内容を,オブジェクト指向にもとづいてモデル化し,C++ で実装した.現在は,既存の OODBMS へ実装している.本稿では,3次元CGのOODBMS実装について報告する.

2. オブジェクト指向による3次元CGデータモデル

2.1 オブジェクト指向

我々は,3次元CGのデータや処理内容をデータベース化することが必要であると考えた.次に挙げる要求,すなわち,1)3次元CGの基本機能をすべて実現すること,2)データベース化の作業が容易であること. の2点を満足させるためには,データベース化において,オブジェクト指向の概念にもとづく3次元CGのデータや処理内容のモデル化が最適であるという結論を得た.その結果,図1のクラス階層を得た.以下で,オブジェクト指向が適しているという理由を簡単に説明する.

3次元CGの基本機能を計算機に実装するには,何らかのプログラミング言語を用いる必要がある.現在,さまざまなオブジェクト指向言語が存在するが,既存のOODBMSで高い性能を発揮するには,C++の使用が適当である. C++は基本的にCの上位互換である. CGでは,ベクトル演算・行列演算を多用するが,C++では,これらの演算の処理に,関数呼び出しの記法 (例,vecmul() など) の代わりに,*や+などの演算子記号を用いることができるので,簡潔に記述できた.また,物体表面の模様は,データ値として表すことが困難であり,シェーディング関数といわれる特別な関数で記述されるが,さまざまな模様をもつ物体を計算機内にモデル化するとき,C++の仮想関数(virtual function)を用いれば容易である.逆に,オブジェクト指向を用いない時は,関数へのポインタを使うことになるので,プログラムが難しくなる.

また,本システムのクラスの機能が不十分だと感じる利用者は,新しいクラスを定義することができる. C++では,あるクラスのオブジェクトへのポインタ変数は,そのクラスの派生クラスのオブジェクトを参照できる.このことを利用して,図1に示すクラスのいずれかを基底クラスとして新しいクラスを派生させた時,その新しいクラスのオブジェクトを引数として渡すことができるようになる.例えば,仮に新しく球形をあらわすクラスSphereを作る時,SphereをSolidの派生クラスとし,Sphereのコンストラクタ内で,Solidのコンストラクタを適切に呼び出すようにすれ.そうすると,Solidクラスの定義を全く変更しなくとも,Sphereのオブジェクトに Solid と全く同じ処理(例えば,幾何変換や画像生成)を行なわせることができる.

OODBMS へ組み込むために,大域変数や大域的な関数を使わないようにした.データの受渡しには大域変数ではなくて,メッセージを用いるようにした.

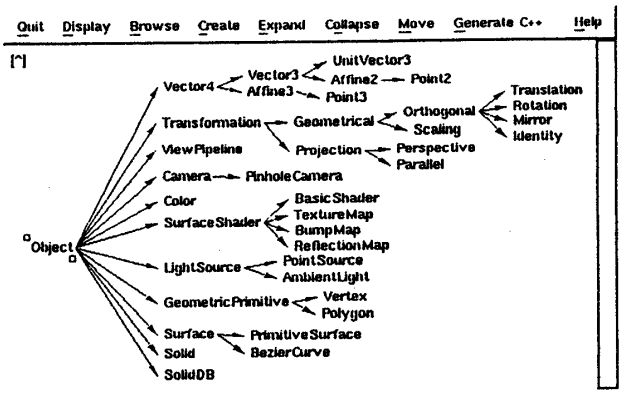

図1. 3次元CGデータモデルのクラス階層

2.2 記述例

我々は,3次元CG画像の生成に必要な全てのデータをモデル化して,計算機内に蓄えることができるように設計を行なった.クラスの記述には C++ (AT&T C++ ver2 準拠) を用いて,およそ 8000 行の C++ プログラムとなった.実際に,Sun Sparc2 ワークステーション上に実装し,画像が正しく生成できることを確認した.

CGは,CGの登場物,視点,光源などのさまざまな要素から成り立っている.CGの登場物は,形状と,登場物固有の座標系から成り立っている.図2に,立体(ティーポット)の記述例を示す.

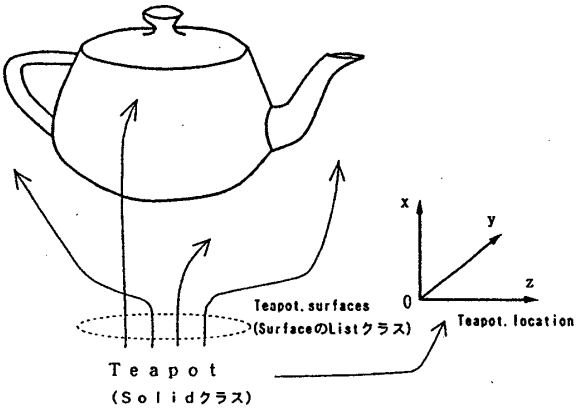

図2. 3次元CCデータモデルによる立体記述

An Implementation of Three Dimensional CG Model using OODBMS
Kunihiko KANEKO, Susumu KUROKI, Yusuke KONDO, Akifumi MAKINOUCHI
Kyushu University

3. OODBMSによる3次元CG

3.1 OODBMSの利点

OODBMSを用いると,データとともに,データに伴う処理(メソッド)もデータベースによって管理される.CGデータがOODBMSによって管理されると,CG作成が容易になるのは次の理由による.

データ値を変化させると,物体の形や属性,視点,光源を変えることができる.従来のレンダリングシステムではプログラミング言語や専用の対話型ツールを用いてデータアクセスを行なうが,OODBMSを用いると,データベースのアクセスメソッドでアクセスを行なうことになる.プログラミング言語を用いるものは,プログラミングの知識が必要であるし,簡単な修正の時は,プログラムの修正,コンパイルが面倒である.専用のツールを用いるものは,内部のデータ構造が決まっているため,システムにクラスを追加するなどで機能を拡張することが困難である.

また,メソッドがデータベース化されると,データ操作と同様の操作でメソッドを呼び出すことができ,計算機やプログラミングの知識の少ない利用者でも,容易にCGを作成することが可能となる.

3.2 実装

今回,我々は,既存のOODBMSの中からONTOSを利用することにした.ただし,トランザクション制御やメモリ管理の命令文などのうちONTOSに固有と思われる機能の利用は避け,特定のOODBMSには依存させない方針で実装することとした.

クラス定義は,そのクラスに属するメンバ変数の名前,型や,メンバ関数の名前,型,引数,処理内容の記述などからなる.何らかのC++のクラスを,データベースで扱えるようにするためには,クラス定義の内容を変更して,それぞれのクラスについて,表1に示す機能を定義することが必要となる.読み込み,書き込みの機能はデータベース化するために必要な機能である.生成,消去,割り当て,解放については,データベース化の有無で,処理内容が変わることがある.オブジェクトの長さや構造は,クラスごとに異なっているので,これらの機能の処理内容も,クラスごとに異なるものとなる.ONTOSでは,ONTOSのライブラリ関数を利用して,C++ソースに変更を加える必要がある.例えば,図3にRGBクラスのヘッダファイルの一部分を示したが,これをONTOSで扱えるようにするためには,図4に示すように変更する必要がある.

3.3 マクロ

3次元CGに必要なクラスは数十になるので,全てをONTOSに合うように正しく書き換えるのは面倒であるし,誤りもおきやすい.また,他のOODBMSへ移植するような場合も面倒である.そこで,C++プリプロセッサのマクロの機能を用いて,Class, ClassImplementation, Constructor, Destructorの4種類のマクロを定義し,このマクロを用いてクラス定義を行なう(図5)ことで,記述が簡単になり,ONTOSの利用が容易になった.

このような言語処理にはマクロの他にもさまざまな方法があるが,マクロを使うことによる特徴は次の通りである. 1) マクロ自体がC++の機能なので,新しくデバッガなどの開発環境を整備しなくとも,従来のC++のものがそのまま利用できる.また,ONTOSにそのまま読ませることができるので簡単である. 2) マクロを用いて,C++にONTOSの機能を追加する形を取っているので,従来のC++の完全な上位互換である. 3) もし,他のOODBMSを使う時にも,マクロ定義の部分だけを変えればよく,プログラムを修正する必要が無い.また,マクロ定義の部分はC++の#define文で書かれているので,理解が簡単である.

機能	説明
生成	オブジェクト生成と初期化
消去	オブジェクトのメモリ上からの消去
割り当て	オブジェクト生成時のメモリ領域の割当
解放	オブジェクト消去時のメモリ領域の解放
読み込み	オブジェクトをDBから読み込む
書き込み	オブジェクトをDBに保存する
破壊	オブジェクトをDBから削除する

表1. データベース化されたオブジェクトの管理に必要な機能

```
class RGB {
 public:
   double   red,green,blue;
   (以下続く)
```

図3. サンプルクラス(RGBクラス)

```
class RGB : public Object {
 public:
  Color( APL* theAPL );
  virtual void putObject(OC_Boolean dealloc=FALSE);
  virtual void deleteObject(OC_Boolean dealloc=FALSE);
  virtual void Destroy OC_Boolean aborted=FALSE);
  ~Color() { Destroy( FALSE ); };
  virtual Type *getDirectType(); private:

  double   red,green,blue;
    (以下続く)
```

図4. サンプルクラスのONTOSによるDB化(C++のみ)

```
Class(RGB,Persistent,0,)
  double   red,green,blue;
  (以下続く)
```

図5. サンプルクラスのONTOSによるDB化(C++とマクロ)

4. おわりに

OODBMSが3次元CGに有効であることを示した.

ONTOSには,オブジェクトの検索とデータ操作を行なうための,グラフィカルなツールが付属している.データ操作のためにプログラムの必要はなく,計算機やプログラムの知識のない利用者でも,簡単にデータを操作することができる.しかし,メッセージパッシングによるオブジェクトのもつメソッドの起動や,オブジェクトのデータ隠蔽の機能は無い.

原理的には,データベース内にメソッドの情報が入っているので,メソッドを検索して実行することができることから,実際に,そのためのプログラムを試作し,実際のCG作成の場面から考察を加える予定である.

参考文献

[1] 黒木進,金子邦彦,牧之内顕文:“アニメーション用データベースにおける時間的知識の表現”,情報処理学会第43回全国大会

[2] 金子邦彦,黒木進,牧之内顕文:“3次元アニメーションに適したデータモデルの検討”,情報処理学会第43回全国大会