Half-Edge Data Structure를 이용한 객체지향 Solid Modeling

최 봉 구, 박 윤 선
명지대학교 산업공학과

Abstract

본 연구는 CAD/CAM에서 3차원 물체를 표현하는 기법중 하나인 Solid Modeling을 구성하는 데이터 구조를 객체지향화 함으로써 보다 효율적이고 유연한 설계개발환경을 제공하고자 한다. 이를 위해서 가장 일반적으로 Solid Modeling을 표현하는 데이터 구조인 기존의 Half-Edge Data Structure를 기본모델로 하여 점(Vertex), 선(Edge), 면(Face), 입체(Solid)를 객체 Class로 하고 Euler Operator를 Method로 하는 객체지향 반모서리 데이터 구조(Object Oriented Half-Edge Data Structure)를 개발하였다.

1. 서론

객체지향 개념은 CAD/CAM분야에 많은 발전을 가져왔다. VLSI CAD분야의 개발자들은 방대한 양의 CAD/CAM 데이터를 관리하기 위해서는 기존의 관계형 데이터베이스(relational Database)로는 불가능하다는 것을 인식하고 분류화(classification), 집합화(aggregation), 일반화(generalization) 개념을 포함하고 있는 객체지향 개념을 도입하였다. 또한 최근에는 객체지향 CAD/CAM 사용자 인터페이스[2], 객체지향 랜더링 기법 등에서도 연구가 이루어지고 있다[1]. 그러나 이러한 객체지향의 CAD/CAM에 대한 응용은 geometric 구현 원리나, modeling 방법 측면에서의 연구는 많이 이루어지지 않은 실정이다. 특히 기존의 구조적 프로그래밍 기법으로 3차원 입체를 나타내는 Solid Modeling을 구현하려면 메모리의 효율적 관리, 재사용성을 강조한 모듈화, 코딩의 편이성 등을 기대하기 어렵다.

이에 본 연구에서는 CAD/CAM의 Solid Modeling의 typical한 데이터 구조인 Half-Edge Data Structure를 객체지향화 하여 구조적 기법의 문제점을 해결함으로써 효율적이고 유연한 설계개발환경을 제공하고자 하였다.

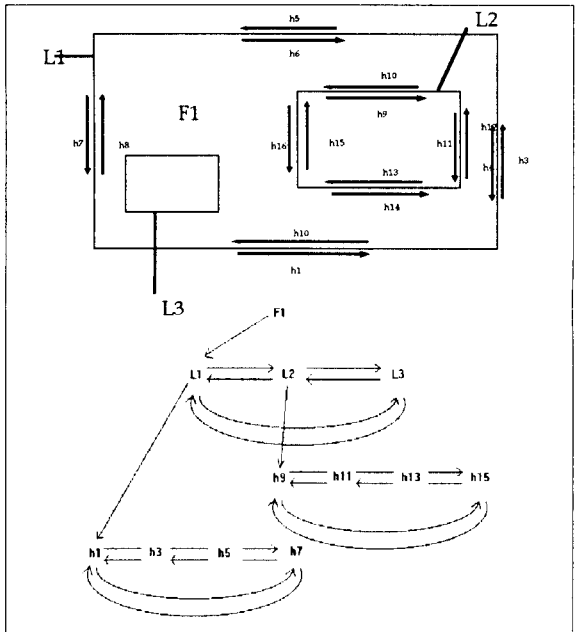
2. 기본모델

2.1 Half-Edge Data Structure

본 연구의 기본모델인 Half-Edge Data Structure는 B-REP방식으로 Solid Modeling을 나타내는 전형적인 데이터 구조로서 입체를 구성하는 각각의 모서리(edge)를 서로 방향이 반대인 두 개의 모서리의 이중연결리스트(doubly linked list)형태로 저장하는 것을 기본개념으로 하고 있다[3]. 또한 Half-Edge Data Structure에는 hole을 갖는 면(face)을 나타내기 위해서 loop의 개념을 도입하였다. loop란 면을 경계짓는 경계선으로서 모든 면의 외부 경계선에 해당하는 하나의 loop를 갖게 된다. 따라서 각각의 면(face)은 직접 반모서리의 이중연결리스트를 갖지 않고, 일단 loop를 보유하고 각각의 loop가 반 모서리의 이중연결리스트를 갖게 된다. Solid Modeling에서 면(face), 루프(loop), 반모서리(half-edge)의 관계를 나타낸 것이 [그림 1]이다.

2.2 Euler Operator

Half-Edge Data Structure를 사용하여 실질적으로 3차원 물체가 형성되어 가는 개개의



[그림 1] Face, Loop, Half-Edge 의 관계

과정을 데이터 구조로 직접 나타낼 수 있는 도구가 Euler Operator이다[3]. Euler Operator는 Modeling한 3차원 입체가 유효하기 위해서는 Euler 공식을 만족해야 한다고 해서 붙여진 이름이다. Euler 공식은 다음 식과 같다.

$$F + V - E - R = 2(S - H)$$

F : 면(Face)의 수
V : 꼭지점(Vertex)의 수
E : 모서리(Edge)의 수
R : 고리(Ring)의 수
S : Shell의 수
H : 구멍(Hole)의 수

이러한 Euler 공식의 유효성을 바탕으로 entity를 새로 생성하거나 소멸시킬 수 있는 Euler Operator를 사용한다. 하나의 3차원 입체를 완성하기 위해서는 여러 가지의 Euler Operator가 유효성이 고려되면서 사용되어지는데 이것들을 구분하기 위해 다음과 같은 알파벳을 이용하여 기호화한다.

m : make	v : vertex	h : hole
k : kill	e : edge	r : ring
f : face	s : solid	l : low-level

예를 들어 mev라는 Euler Operator는 edge 하나와 vertex하나를 생성하는 연산을 수행한다.

3. 객체지향 Half-Edge Data Structure

3.1 Class의 구성

Half-Edge Data Structure를 객체지향화하기 위해서는 Solid를 구성하는 모든 요소들, 즉 점(vertex), 선(edge), 면(face), 입체(solid) 뿐만 아니라, 반모서리(half-edge), 루프(loop) 등의 class를 효과적으로 구성하는 것이 가장 중요하다. class를 어떻게 구성하느냐에 따라서 객체지향 모델링의 성공여부가 결정되어지기 때문이다. class가 독립적인 하나의 객체로서 존재하기 위해서는

(1) 각 객체의 구성 특성(attribute)을 저장할 수 있는 data structure와
(2) 그 객체가 생성되어 활동할 수 있도록 하는 method,
(3) 그리고 이러한 객체에 전달되어 실제적인 작동을 가능하게 message

등이 반드시 필요하다.[4]

본 연구에서는 이러한 객체지향 개념을 바탕으로 기존의 Half-Edge Data Structure를 attribute으로 하고 Euler Operator를 method로 하는 새로운 class들을 구현하였다. 또한 이 class들에는 하나의 객체가 생성 혹은 소멸시 메모리의 할당 및 반환을 가능하게 하는 메모리 관리 method를 포함시키고, 다른 class가 접근하여 정보를 변화시키는 일이 없도록 정보를 은닉함으로써 완전한 독립성을 갖도록 하였다. 기존의 구조적 프로그래밍 기법으로 Solid Modeling을 구현할 경우는 이러한 메모리의 관리나 Euler 연산을 모두 하나의 main함수가 관장하므로 복잡한 3차원 입체를 나타낼 때 어려움이 많았다.

Half-Edge Data Structure를 이용한 Solid Modeling의 가장 기본이 되는 객체는 점(vertex)이므로 Vertex Class를 어떻게 구현하는가는 시스템 전체에 영향을 줄 수 있다. 본 연구에서는 Vertex Class를 [그림 2]와 같이 구현하였다. 그 외의 다른 class들, 즉 Edge, Half-Edge, Face, Loop, Solid와 같은 Class들은 [그림 3]과 같다.

VertexClass

private protected public

Attributes

변수명	설명
VertexNo	vertex 번호
X, Y, Z	vertex의 좌표값
VerToHalf	이 점을 시작으로하는 Half-Edge
PrevVertex	이 점의 바로 전 점 포인터
NextVertex	이 점의 바로 다음 점 포인터

Methods

Memory와 List 관리 Method

Euler Operator (mvfs)

[그림2] Vertex Class 의 구성

HalfEdgeClass

Attribute
- edge를 가리키는 포인터
- 시작점 vertex 포인터
- loop 포인터
- 다음 HalfEdge 포인터
- 이전 HalfEdge 포인터

Method
- Memory와 List 관리
- Euler Operator

EdgeClass

Attribute
- 방향이 같은 Halfedge
- 방향이 다른 Halfedge
- 다음 Edge 포인터
- 이전 Edge 포인터

Method
- Memory와 List 관리
- Euler Operator(mev)

LoopClass

Attribute
- HalfEdge cycle link의 처음
- face 포인터
- loop포인터
- 다음 loop 포인터
- 이전 loop 포인터

Method
- Memory와 List 관리
- Euler Operator (kemr, mekr)

FaceClass

Attribute
- Face 번호
- Solid 포인터
- 외곽 loop 포인터
- 내부 loop 포인터
- 평면방정식
- 이전 face 포인터
- 다음 face 포인터

Method
- Memory와 List관리
- Euler Operator (mef, kef)

SolidClass

Attribute
- solid번호
- face 포인터
- edge 포인터
- vertex포인터
- 다음 Solid 포인터
- 이전 Solid 포인터

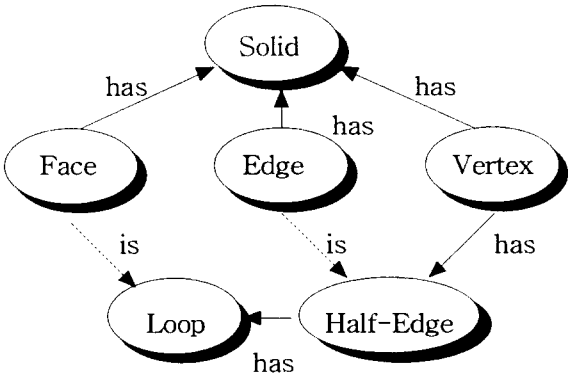
Method
- Memory와 List관리
- Euler Operator

[그림 3] 그 외의 class들

3.2 Class의 상속(Inheritance)

Half-Edge Data Structure는 B-REP방식의 한 종류이기 때문에 root가 되는 입체(solid)로부터 하위 element, 즉 면(face), 선(edge), 점(vertex)에 이르기까지 계층적 tree 구조를 하고있다[5]. 그러나 이러한 계층적 tree구조는 하나의 모서리(edge)가 방향이 서로 다른 두 개의 반모서리(half-edge)로 분리된 것이나, 하나의 면(face)이 여러 개의 루프(loop)로 구성되는 등의 Half-Edge Data Structure의 특징을 효과적으로 표현하지 못하고 있다. 따라서 본 연구에서는 객체지향의 상속(inheritance)의 개념을 Half-Edge Data Structure에 적용하여 구조적 프로그래밍 기법이 갖고있는 문제점을 해결하려 하였다. 또한 상속의 개념을 사용하면 코딩의 반복을 피할 수 있을 뿐만이 아니라, 각 객체간의 유기적인 관계를 효율적으로 나타낼 수 있는 장점이 있다.

본 연구에서 객체지향화한 Half-Edge Data Structure를 구성하는 각 Class들의 상속관계는 [그림 4]와 같다.



[그림 4] 각 Class간의 상속관계

(1) 하나의 반모서리(half-edge)는 방향성을 갖기 때문에 어느 점(vertex)에서 시작했느냐에 따라 틀리다. 따라서 반모서리와 점은 HAS-A관계를 갖는다.

(2) 하나의 모서리(edge)는 서로 방향이 반대인 두 개의 반모서리(half-edge)로 이루어져 있다. 따라서 모서리와 반모서리는 class와 subclass의 관계로 나타낼 수 있으므로 IS-A 관계를 갖는다.

(3) 하나의 루프(loop)는 하나이상의 반모서리들로 이루어지기 때문에 HAS-A관계를

갖는다.

(4) 하나의 면(face)은 하나이상의 루프로 이루어 진다. 루프는 면의 특성을 모두 갖고 있기 때문에 class와 subclass의 관계로 나타낼 수 있다. 따라서 IS-A관계를 갖는다.

(5) 하나의 입체(solid)는 다수의 면(face), 선(edge), 점(vertex)들로 이루어진다. 따라서 이러한 객체들간에는 HAS-A관계를 가진다.

4. Solid Modeling 구현 방법

본 연구에서는 Visual C++를 이용하여 객체지향 Half-Edge Data Structure를 구현하였다. Solid, Face, Loop, Edge, Half-Edge, Vertex 등의 Class외에도 이러한 데이터들을 화면에 입체적으로 표현하기 위한 3D Class, 기본적인 메모리관리를 위한 Memory Class 등을 각 객체에 상속시켰다.

5. 결론

본 연구에서는 Half-Edge Data Structure의 측면에서 각각의 entity, 즉 면(face), 선(edge), 점(vertex)을 객체로 인식하고 이러한 객체들 상호 연관에 의하여 입체(solid)라는 하나의 객체를 형성하도록 하였다. 이렇게 객체지향화한 entity들은 상속(inheritance), 정보은닉(information hiding), 다형성(polymorphism) 등의 여러 가지 진보된 특성들을 가지고 있기 때문에 기존의 구조적 프로그래밍기법을 바탕으로한 Solid Modeling에 비해 메모리의 관리나 코딩의 편이성, 모듈의 재사용성 등에서 매우 유용하다. 또한 객체 entity들은 서로 독립성을 유지하기 때문에 새로운 객체의 생성, 메모리의 할당 및 환원, 객체 변형 등의 작업을 자체적으로 수행할 수 있다. 이러한 기능은 CAD프로그램 내에 새로운 기능의 추가나 불필요한 기능의 삭제 등을 용이하게 하므로 매우 유연한 디자인 환경을 제공할 수 있을 뿐만 아니라 도면 정보의 표준화에도 많은 도움을 줄 수 있을 것으로 기대 된다.

6. 참고문헌

[1] Deborah Silver, "Object-Oriented Visualization", *IEEE Computer Graphics and Application*, May, pp.54-62, 1995.

[2] Merilyn Ch'ng, "An Object-Oriented Application Framework for 3D Graphics", *OOPSLA '94 Poster*, 1994.

[3] M. Mantyla, "An Introduction to Solid Modeling", *Computer Science Press*, 1988.

[4] 김형주, "알기쉬운 객체지향시스템", *동아출판사*, pp.157-161, 1993.

[5] 이건우, "컴퓨터그래픽과 CAD", *영지문화사*, pp.120-129, 1994.