OpenVolMesh: 삼차원 볼륨 기반의 메쉬 표현을 위한 범용적이고 효과적인 자료 구조

OpenVolMesh: Generic and Efficient Data Structure for 3D Volumetric Meshes

김준호, 서진석, 오세웅
동의대학교 게임공학과

Junho Kim(kim.junho@deu.ac.kr), Jinseok Seo(jsseo@deu.ac.kr),
Seiwoong Oh(osw@deu.ac.kr)

요약

메쉬는 삼차원 물체를 표현하기 위해 가장 널리 쓰이는 자료구조 중 하나이다. 지금까지 메쉬 자료구조로는 주어진 삼차원 물체의 표면 상의 정보만을 샘플링하는 표면 메쉬가 널리 쓰였다. 그러나, 컴퓨터의 처리 능력이 향상됨에 따라, 콘텐츠 개발시 물체의 내부 정보까지 계산할 수 있는 볼륨 메쉬의 필요성이 점점 증대되고 있는 추세이다. 본 논문에서는 반면 (half-face) 자료구조 기반의 새로운 삼차원 볼륨 메쉬 라이브러리인 OpenVolMesh를 제안하고, 그 디자인과 구현에 관하여 기술한다. 제안하는 OpenVolMesh는 세계적으로 널리 쓰이고 있는 표면 메쉬 라이브러리인 OpenMesh 위에 볼륨 메쉬를 다룰 수 있는 자료구조를 추가하는 방식으로 디자인하였다. OpenVolMesh는 제네릭 프로그래밍 (generic programming), 메쉬 구성 요소에 대한 동적속성 할당 (primitive property), 배열 기반의 자료구조 등을 지원하며, OpenMesh와 소스 레벨에서의 호환성을 제공한다. 볼륨 메쉬 스무딩 및 CW-셀 분할과 같은 적용사례를 통해 OpenVolMesh가 삼차원 볼륨 기반의 콘텐츠 개발에 효과적으로 쓰일 수 있음을 보인다.

■ 중심어 : | 메쉬 라이브러리 | 볼륨 메쉬 | 반면 자료구조 |

Abstract

Meshes are the most appropriate data structures for representing 3D geometries. Surface meshes have been frequently used for representing 3D geometries, which only samples data on the surfaces of the given 3D geometries. Thanks to the improvements of computing powers, it is required to develop more complicated contents which utilize the volumetric information of 3D geometries. In this paper, we introduce a novel volumetric mesh libraries based on the half-face data structure, called OpenVolMesh, and describe its designs and implementations. The OpenVolMesh extends the OpenMesh, which is one of the most famous mesh libraries, by supporting volumetric meshes. The OpenVolMesh provides the generic programming, dynamic allocations of primitive properties, efficient array-based data structures, and source-level compatibility with OpenMesh. We show the usefulness of the OpenVolMesh in the developments of 3D volumetric contents with prototypic implementations such as volumetric mesh smoothing and CW-cell decompositions.

■ keyword : | Mesh Library | Volumetric Meshes | Half-face Data Structure |

* 본 연구는 문화관광부 및 한국문화콘텐츠진흥원의 지역문화산업연구센터(CRC) 지원사업의 연구결과로 수행되었습니다.

접수번호 : #080624-004
접수일자 : 2008년 06월 24일
심사완료일 : 2008년 07월 08일
교신저자 : 김준호, e-mail : kim.junho@deu.ac.kr

I. 서 론

영화와 게임과 같은 3차원 디지털 콘텐츠 시장이 커짐에 따라, 3차원 디지털 형상(digital geometry)이 차세대 디지털 미디어로서 많은 주목을 받고 있다[8]. 디지털 형상이란 주어진 삼차원 물체의 형상에 관련된 데이터(예, 3차원 좌표, 색깔, 텍스쳐 등)를 샘플링하여 그 물체를 컴퓨터를 통해 디지털로 복원하는 것으로, 디지털 형상의 표현(representation) 및 변형(deformation), 시뮬레이션(simulation) 및 실시간 렌더링 (real-time rendering)과 같은 응용분야에서 디지털 형상을 활용하기 위한 많은 연구가 있었다[8].

디지털 형상 분야의 기존 연구들은 대부분 주어진 삼차원 물체가 가지고 있는 내부 정보는 배재한 체, 물체의 표면(surface) 상의 데이터를 어떻게 효과적으로 활용할 것인가에 많은 관심이 있었다. 이는 기존의 컴퓨팅 환경에서는 삼차원 물체의 내부까지 표현하기엔 계산량이 너무 많다는 인식과 더불어, 대부분의 영화 및 게임 콘텐츠에서는 눈에 보이는 물체의 표면에 관한 정보가 훨씬 더 중요하다고 생각했기 때문이었다. 그러나, 최근 CPU 및 GPU의 발전으로 인해 컴퓨팅 파워가 물체의 내부까지 효과적으로 처리하기에 부족함이 없고, 영화 및 게임에서 물체의 내부까지 다루는 복잡한 콘텐츠 개발의 필요성이 증대되고 있다. 이로 인해, 삼차원 물체의 내부를 효과적으로 처리하고 이를 응용하려는 연구가 태동단계에 있다[7].

일반적으로 삼차원 물체를 컴퓨터에서 표현할 때 가장 널리 쓰이는 자료구조는 메쉬(mesh)이다. 메쉬 자료구조는 정점(vertex), 에지(edge), 면(face)등과 같은 구성요소(primitive)들을 통해 주어진 물체를 조각별 선형근사 (piece-wise linear approximation)하여 표현한다. 만일, 주어진 물체의 표면만을 컴퓨터에서 표현한다면 메쉬 자료 구조는 정점/에지/면 등으로 이루어진 표면 메쉬(surface mesh)로 충분하다. 현재, 표면 메쉬를 효과적으로 지원하는 라이브러리는 연구 목적 및 일반 공개용으로 널리 쓰이고 있다 (예, OpenMesh [1], CGAL [2], VCG [3]). 그러나 기존의 대부분의 자료구조들은 물체의 내부정보까지 컴퓨터에서 표현할 수 있는 볼륨 메쉬(volume mesh)를 지원하지 않거나, 사면체 기반의 볼륨 메쉬만을 지원하기 때문에, 물체의 내부까지 다루는 복잡한 콘텐츠 개발에는 한계가 있다.

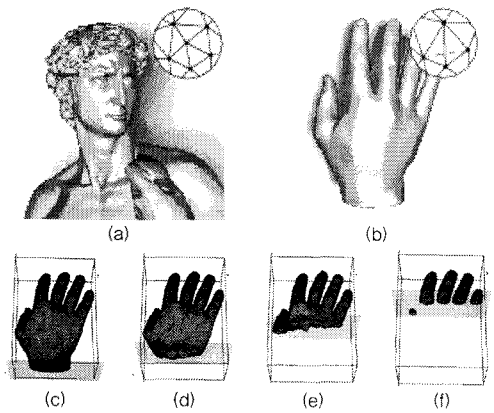
본 논문에서는 OpenVolMesh라 불리우는 삼차원 볼륨 기반의 메쉬 표현을 위한 범용적이고 효과적인 자료구조를 제안하고 그 활용사례를 보여준다. 제안하는 OpenVolMesh는 현재 학계에서 연구용으로 가장 널리 쓰이는 표면 메쉬 자료구조인 OpenMesh를 볼륨 메쉬를 다룰 수 있도록 확장, OpenMesh에 익숙한 기존 사용자들이 쉽게 빠르게 볼륨 메쉬 기반의 응용 콘텐츠를 개발할 수 있다는 장점이 있다. 또한, OpenMesh가 가지는 장점을 모두 수용, 제네릭 프로그래밍 (generic programming), 메쉬 구성요소에 대한 동적속성 할당 (primitive property), 배열 기반의 자료구조 등을 통한 효과적인 콘텐츠 개발이 가능하다. 뿐만 아니라, OpenMesh와의 소스 레벨 호환성을 통해, 기존 OpenMesh 기반의 원시코드들을 모두 재사용해 쓸 수 있어, OpenMesh로 개발된 표면 메쉬 기반의 콘텐츠들을 쉽게 볼륨 메쉬 기반의 콘텐츠들과 섞어 쓸 수 있다는 장점이 있다.

II. 배경 지식 및 이전 연구

메쉬란 정점/에지/면 등과 같은 구성요소들을 통해 주어진 삼차원 물체의 형상을 조각별 선형근사로 표현하는 자료구조이다. 특히, 복잡한 연산을 효과적으로 처리하기 위해서는 인접한 구성요소들 간의 복잡한 연결구조(adjacency)를 효과적으로 관리하는 것이 중요한데, 이러한 연결구조가 메쉬 자료구조의 핵심기능 중 하나이다. 이러한 연결구조를 통해 삼차원 형상 변형, 편집, 몰핑, 텍스쳐 생성, 시뮬레이션과 같은 복잡한 디지털 형상 처리를 빠르게 처리할 수 있게 된다.

삼차원 물체를 메쉬를 통해 표현할 때는 크게 표면 메쉬를 이용하는 경우와 볼륨 메쉬를 이용하는 경우, 두 가지를 생각할 수 있다. 표면 메쉬를 사용할 때는 삼차원 물체의 표면을 다각형의 집합으로 근사하여 물체를 표현하게 되고 [그림 1a], 볼륨 메쉬를 이용할 경우

에는 물체의 내부 및 표면을 다면체를 이용하여 표현하게 된다[그림 1b]~[그림 1f].



(a) 표면 메쉬, (b) 볼륨 메쉬, (c)-(f) 다양한 높이의 단면을 (b)의 볼륨 메쉬에 적용함으로써 각 부분에서의 내부 사면체들을 확인하는 모습들.

그림 1. 삼차원 메쉬

표면 메쉬는 정점/에지/면 등으로 구성되며, 정점(vertex)은 물체 표면 상의 삼차원 위치 정보 및 텍스쳐 정보 등을 가지고 있으며, 에지(edge)는 인접한 정점들 간의 연결관계를 정의하며, 면(face)은 세 개 이상의 정점 및 에지들로 이루어진 다각형을 통해 물체 표면의 일부분을 표현한다. 표면 메쉬에서 연결구조를 효과적으로 처리하기 위한 대표적인 방법으로는 윙드에지(winged-edge) 자료구조와 반에지(half-edge) 자료구조가 있으며[4][5], CGAL, OpenMesh, VCG 등을 비롯한 최근의 메쉬 라이브러리에서는 반에지 기반의 자료구조가 일반적으로 많이 쓰이고 있다.

반에지 자료구조는 정점/에지/면으로 이루어진 메쉬 구성요소들 간의 유기적인 연결구조를 위해 각각의 에지가 자신이 둘로 쪼개진 형태인 반에지를 가지고 있다고 가정한다. 각각의 반에지는 자신에게 인접한 정점, 에지, 면에 대한 정보를 가지고 있으며, 같은 면에 있는 반에지 중 자신의 앞과 뒤에 있는 반에지에 접근할 수 있는 정보와 더불어 자신의 다른 반쪽인 반대편 반에지에 접근할 수 있는 정보를 가지고 있다. 또한, 정점/에지/면은 각각 인접한 반에지 중 하나에 접근할 수 있는 정보를 가지고 있다. 따라서, 반에지 자료구조에서는 반에지가 제공하는 인접관계를 통해 임의의 메쉬 구성요소로부터 다른 메쉬의 구성요소까지의 효과적인 접근이 가능하게 된다. 반에지 구조에 대한 보다 자세한 사항은 [5]를 참조하도록 한다.

기존에 널리 쓰이고 있는 메쉬 라이브러리인 CGAL, OpenMesh, VCG 등의 장단점을 정리하면 다음과 같다. CGAL의 경우 표면 메쉬 및 볼륨 메쉬를 통해 물체를 표현하는 방법을 모두 지원하는 장점이 있다. 그러나, CGAL의 볼륨 메쉬는 다양한 응용프로그램을 개발할 때 편리하게 쓸 수 있는 메쉬 구성요소에 대한 동적속성 할당 기능이 없다는 한계를 지니고 있다. 뿐만 아니라 볼륨 메쉬를 다룰 경우, 삼차원 물체를 사면체(tetrahedron)들로만 표현해야 하는 한계가 있으며, 에지 및 면 등에 사용자 데이터를 효과적으로 표현할 수 없다는 한계가 있다. OpenMesh의 경우, 메쉬 구성요소에 대한 동적속성 할당 기능을 통해 메쉬 처리에 필요한 데이터를 상황에 따라 동적으로 할당·해제할 수 있다는 장점이 있지만, 물체의 내부를 표현할 수 있는 볼륨 메쉬를 제공하지 않는다는 한계가 있다. VCG 라이브러리의 경우, 메쉬 구성요소에 대한 동적속성 할당 기능과 더불어 볼륨 메쉬를 지원하지만, 사면체 기반의 볼륨 메쉬만을 지원한다는 한계가 있다.

본 논문에는 OpenMesh 기반 위에 볼륨 메쉬를 다룰 수 있는 새로운 메쉬 라이브러리인 OpenVolMesh를 제안하고, 그 설계에 대해서 살펴본다. OpenVolMesh는 반면(half-face) 자료구조를 통해 사면체로 이루어진 볼륨 메쉬 뿐만 아니라 임의의 셀로 표현된 삼차원 볼륨 메쉬 형태를 지원한다. 또한, 기존 OpenMesh 소스코드를 수정없이 OpenVolMesh와 같이 쓸 수 있는 소스코드 레벨에서 호환성을 보장하며, 제네릭 프로그래밍 (generic programming), 메쉬 구성요소에 대한 동적 속성 할당 (primitive property) 및 배열 기반의 효과적인 자료구조를 지원한다.

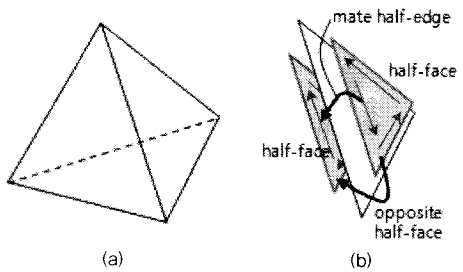
III. OpenVolMesh

본 장에서는 OpenVolMesh의 자료구조, 내부설계 및

그 구현, 프로그래밍 인터페이스 등에 대해서 살펴본다.

1. 자료구조

삼차원 볼륨은 임의의 지점 p와 그 인접 부위인 Ne(p)가 3차원 공간과 같은 위상구조를 가지는 3-다양체(3-manifold)이다. 따라서, 삼차원 볼륨 메쉬의 요소들은 정점/에지/면과 더불어 삼차원 공간을 표현할 수 있는 기본 단위인 셀(cell)로 구성된다. 또한, 이들 구성요소들 간의 유기적인 연결관계를 통해, 3-다양체의 인접성(adjacency)을 표현해야 하는데, OpenVolMesh에서는 이를 위해 반면(half-face) 자료구조를 사용한다[그림 2].



(a) 셀, (b) 반면 자료 구조

그림 2. 반면 자료구조 기반의 삼차원 볼륨 메쉬

반면 자료구조는 2-다양체 성격을 지니는 표면 메쉬를 다루는 데 쓰이는 반에지(half-face) 자료구조를 3-다양체 성격을 지니는 볼륨 메쉬에 맞도록 확장한 개념이다. 반면 자료구조에서는 각각의 셀이 면으로 둘러싸여 구성되며 [그림 2a], 각각의 면은 자신이 둘로 쪼개진 형태인 반면으로 구성되어 있다고 가정한다[그림 2b].

각각의 반면은 자신에게 인접한 반에지, 면, 셀에 대한 정보를 가지고 있으며, 이와 더불어 자신의 다른 반대쪽에 있는 반면에 대한 정보를 가지고 있다. 삼차원 볼륨 메쉬의 반에지는 표면 메쉬의 반에지와 유사하게 자신에게 인접한 정점, 에지, 반면, 그리고 자신에게 인접한 반에지들에 관한 정보를 가지고 있다. 따라서, 반면 자료구조에서는 반면 및 반에지를 통해 정점/에지/면/셀들 간의 인접관계 유지가 가능하며, 이를 통해 임의의 메쉬 구성요소로부터 다른 임의의 메쉬 구성요소로의 빠른 접근이 가능하게 된다.

또한, 반면 자료구조를 통해 정사면체의 셀 뿐만 아니라 다양한 형태의 셀로 이루어진 볼륨 메쉬를 표현할 수 있다. 만일, 각각의 셀이 정육면체로 표현되는 경우, 각 셀은 각 면이 사각형인 6개의 면을 가지며, 각 반면은 4개의 반에지를 가지게 된다. 이 경우도, 서로 간의 인접관계는 반면과 반에지들이 가지는 연결정보로부터 도출할 수 있다.

2. 설계 및 구현

OpenVolMesh는 C++ 언어기반으로 구현되어 있으며, 삼차원 볼륨 메쉬 애플리케이션의 효과적인 개발을 위해 제네릭 프로그래밍, 배열 기반의 자료구조 및 메쉬 구성요소에 대한 동적 속성 할당 기능을 지원한다.

2.1 제네릭 프로그래밍

일반적으로 자료구조를 어떻게 정의하느냐에 따라, 메모리 사용량과 수행시간 사이에 상관관계(trade-off)가 생긴다. 본 논문에서 제안하는 반면 자료구조 기반의 삼차원 볼륨 메쉬 자료구조에도 다음과 같은 메모리-수행시간 사이의 상관관계가 형성된다.

임의의 주어진 면에 대해 그 면을 구성하는 에지 및 정점을 순차적으로 접근하는 경우를 생각해 보자. 한 면에 있는 에지 및 정점을 반시계 방향(counter-clockwise)으로 순차적으로 접근하는 경우, 단순하게 각 반에지 상에 저장되어 있는 다음 반에지에 대한 정보를 이용해 접근하면 된다. 그러나, 주어진 면에 대해 시계방향으로 접근하는 경우는 두 가지 방법을 생각해 볼 수 있다. 하나는 반시계 방향으로 접근할 때와 마찬가지 방법으로, 각 반에지에 이전 반에지 정보를 저장해 놓고 이를 이용하는 방법이다. 이는 메모리는 많이 드는 단점이 있지만 처리속도가 빠르다는 장점이 있다. 다른 하나는 각 반에지의 이전 반에지를 찾기 위해 각 반에지에 저장된 다음 반에지의 정보를 이용, 주어진 면을 반시계방향으로 돌아 이전 반에지를 찾는 방법이다. 이 방법은 메모리는 적게 드는 장점은 있지만, 처리속도가 느린 단점이 있다.

주어진 에지를 중심으로 그에 인접한 모든 면 또는 셀에 접근하는 경우에도 이와 유사한 문제가 발생한다. 각각의 반에지는 같은 셀 내부에서 마주보고 있는 반에지에 관한 정보를 저장하고 있으며, 이는 한 에지를 중심으로 인접한 면 및 셀을 접근하는데 활용된다. 이 정보만을 가지고 있을 경우 메모리는 적게 드는 장점이 있지만, 다른 셀에 속해 있으면서 같은 에지 상에 있는 반에지를 찾는 경우 처리시간이 많이 걸리는 단점이 있다. 만일, 각각의 반에지가 인접한 셀에 있으면서 마주보고 있는 반에지에 관한 정보까지도 저장하고 있으면, 메모리는 많이 들지만 처리 속도는 향상시킬 수 있다.

이와 같이 메모리의 중요성과 수행시간 중요성은 응용프로그램의 목적에 따라 프로그래머가 선택할 수 있도록 하는 것이 가장 바람직할 것이다. OpenVolMesh는 이를 C++의 템플릿 기능을 통해 컴파일 시에 선택할 수 있도록 설계되어 있다. [표 1]은 템플릿의 trait을 통해 메모리 사용량을 늘이되, 수행시간을 줄이는 방법으로 볼륨 메쉬를 정의하는 방법을 보여주고 있다.

표 1. 메모리는 많이 쓰지만, 수행시간을 줄일 수 있는 방법으로 볼륨 메쉬를 만드는 예

```
struct MyTriats: public DefaultTraints
{
  EdgeAttributes(Attributes::PrevHalfedge |
                 Attributes::MateHalfedge);
};
typedef OpenVolMesh::ArrayKernelT<MyTraits>  Mesh;
```

2.2 배열기반의 자료구조

일반적으로 메쉬를 활용하는 응용프로그램에서는 런타임(run-time) 시 정점/에지/면/셀 등과 같은 메쉬 구성요소를 랜덤 엑세스(random access)하는 경우가 대부분이다. 또, 메쉬 단순화와 같이 특수한 경우를 제외하고는, 메쉬 구성요소가 응용프로그램의 런타임 시 임의의 순서로 삭제되는 경우는 매우 드물며, 메쉬 구성요소들이 새로이 생성되는 경우가 일반적이다.

즉, 정점/에지/면/셀 등과 같은 메쉬 구성요소는 생성, 랜덤 엑세스가 효과적인 자료구조가 필요한데, 이는 배열이라는 자료구조가 가지는 특성이며, OpenVolMesh에서는 각각의 메쉬 구성요소들을 배열을 통해 표현한다. OpenVolMesh에서는 배열의 효과적인 처리를 위해, C++ STL의 vector 자료구조를 통해 메쉬 구성요소를 처리한다. 또한, 메쉬 구성요소에 대한 자료구조의 접근을 핸들(handle)을 통해서만 할 수 있도록 하여, OpenVolMesh 사용자가 OpenVolMesh의 내부 자료구조를 모르는 상태에서도 쉽게 프로그래밍이 가능하도록 설계하였다.

메쉬 단순화와 같은 응용 프로그램은 메쉬 구성요소가 임의의 순서로 삭제되게 된다. 이와 같은 메쉬 구성요소의 임의 삭제에 효과적으로 대처하기 위해, 삭제된 메쉬 구성요소에 대해 그것이 삭제되었다는 태그만을 체크해 놓은 다음, 프로그래머가 원하는 시점에 태그가 체크된 메쉬 구성요소들을 모두 삭제하는 가비지 컬렉션(garbage collection) 방식으로 메쉬 구성요소를 삭제하도록 설계하였다. [표 2]는 OpenVolMesh에서 메쉬에 정점을 추가하고 랜덤 엑세스하며, 가비지 컬렉션을 통해 정점을 삭제하는 과정을 보여주고 있다.

표 2. 메쉬 구성요소의 생성, 랜덤 엑세스 및 가비지 컬렉션을 통한 삭제의 예

```
Mesh                        myMesh;
Mesh::VertexHandle          vh0, vh1;
Mesh::Point   p;
vh0 = myMesh.add_vertex(Point(0,0,0));
vh1 = myMesh.add_vertex(Point(0,1,0));
p = myMesh.point(vh0);
myMesh.set_deleted(vh0);
myMesh.garbage_collection();
```

2.3 메쉬 구성요소에 대한 동적속성 할당

메쉬 응용 프로그램의 개발에 있어 메쉬 구성요소들에 대해 사용자 정의 자료구조가 필요할 경우가 많다. 뿐만 아니라, 이러한 사용자 정의 자료구조는 특정 시점에서만 필요하고 그 이후 필요하지 않을 경우가 대부분이다.

예를 들어, 메쉬 스무딩의 경우 각 정점은 스무딩된 삼차원 위치에 대한 계산 결과를 저장할 장소가 필요하다. 이를 위한 한가지 방법으로는 각 정점에 대해 컴파일 시 정점의 자료구조에 계산 결과를 저장할 공간을 지정하는 방법이 있을 것이다. 그러나, 이 방법은 메쉬 스무딩이 모두 끝나 해당 자료구조가 더 이상 필요 없

을 때, 이를 동적으로 해제할 수는 있는 방법이 없다. 다른 방법으로는 각 정점에 포인터를 두어 동적 메모리 할당 (dynamic memory allocation) 기법을 쓰는 것이 있을 것이다. 그러나, 이 방법은 각 정점마다 포인터를 위한 4바이트의 추가 공간이 필요하다는 단점이 있다. 일반적으로 메쉬의 구성요소 개수는 수천에서 수백만 개에 이르기 때문에, 포인터를 통한 동적 할당은 메모리 사용량 측면에서 좋은 아이디어가 아니다.

OpenVolMesh에서는 메쉬 구성요소에 대한 동적속성 할당 (primitive property) 기능을 통해 메모리 오버헤드를 최소하며 사용자 정의 자료구조를 효과적으로 다룬다. 또한, 사용자는 자신이 원하는 시점에 사용자 정의 자료구조를 해당 메쉬 구성요소들에 대해 동적으로 할당시키고 이후 자신이 원할 때 소멸시키는 것이 가능하다. 이는 OpenVolMesh에서 메쉬 구성요소들이 각각 배열의 형태로 관리되기 때문에 다음과 같은 방법으로 쉽게 구현 가능하다. 즉, 임의의 메쉬 구성요소에 대해 동적속성을 할당하고자 할 경우, 그 메쉬 구성요소의 크기와 같은 크기를 가지는 사용자 정의 자료구조의 배열을 만들고, 배열의 시작 주소를 해당 볼륨 메쉬가 관리하도록 한다. 각 메쉬 구성요소에 대한 해당 속성과의 연관성은 배열의 인덱스를 통해 자동으로 처리되며, OpenVolMesh의 사용자는 메쉬 구성요소의 핸들(handle)을 통해 해당속성에 접근할 수 있다.

[표 3]은 볼륨 메쉬 상의 각 셀에 대해 사용자 정의 자료구조인 Point를 할당하고 이후 해제하는 과정의 예를 보여주고 있다.

표 3. 메쉬 구성요소에 대한 동적속성 할당

```
Mesh                              myMesh;
Mesh::FaceHandle                  fh(0);
Mesh::Point    cnt;
OpenVolMesh::CPropHandle(Mesh::Point)   cprop_float;

myMesh.add_property(cprop_float);
myMesh.property(cprop_float, fh) = 10.0f;
myMesh.remove_property(cprop_float);
```

3. 프로그래밍 인터페이스

메쉬 응용프로그램에서는 각 구성요소를 순차적으로 접근하거나 인접성을 통해 접근하는 두 가지 경우가 주로 많이 활용된다. OpenVolMesh에서는 메쉬 구성요소들을 순차적으로 접근하기 위해 iterator를 제공하며, 인접성을 통해 메쉬 구성요소들에 접근하기 위해 circulator를 지원한다. [표 4]는 주어진 볼륨 메쉬에 존재하는 모든 셀들을 iterator를 통해 접근하는 경우와 한 에지에 인접해 있는 모든 셀들을 circulator를 통해 접근하는 예를 보여주고 있다.

표 4. Iterator를 이용한 순차적인 접근과 circulator를 통한 인접 메쉬 구성요소 접근의 예

```
Mesh                          myMesh;
Mesh::EdgeHandle              eh;
Mesh::CellIter c_it;
Mesh::EdgeCellIter            ec_it;

for (c_it=myMesh.cells_begin();
      c_it!=myMesh.cells_end(); ++c_it)
{
  // do something for each cell
}
eh = mesh.edge_handle(10);
for (ec_it=mesh.ec_iter(eh); ec_it; ++ec_it)
{
  // do something for each cell addjacent to edge (10)
}
```

IV. 적용 사례

본 장에서는 삼차원 메쉬 스무딩 및 CW-셀 분할과 같은 응응프로그램을 통해 OpenVolMesh의 활용성을 입증한다. 특히, 각각의 응용프로그램은 OpenVolMesh 라이브러리가 제공하는 제네릭 프로그래밍, 동적속성 할당, 배열 기반의 자료구조 등을 통해 보다 효과적으로 구현되었다.

1. 삼차원 볼륨 메쉬 스무딩

메쉬 스무딩은 입력으로 들어온 메쉬 상의 정점의 위치를 바꾸어, 메쉬를 보다 부드럽게 변형시키는 것을 말한다. 메쉬 스무딩을 위해서는 정점 데이터들이 가지고 있는 고주파(high frequency) 영역의 정보를 없애야 하는데, 이를 위해 아래와 같은 Laplace 변환을 이용한다.

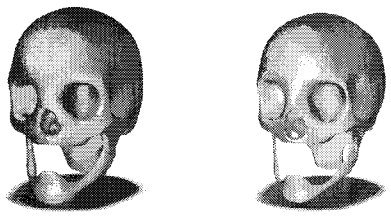
$$L(v_i) = v_i - \frac{1}{d}\sum_{j \in N(i)}^{d} v_j \qquad (1)$$

직관적으로 설명하자면, $L(v_i)$는 i-번째 정점의 위치 주파수에 대한 스펙트럼에 해당하며 [6], 다음과 같은 방식을 통해 고주파 영역을 제거하여 i-번째 정점의 위치를 부드럽게 만들 수 있다.

$$v_i \leftarrow v_i - tL(v_i) \quad (\text{단, } t \in [0,1])$$

만일, t=0이면 고주파 영역이 전혀 제거되지 않은 채 원래 위치가 그대로 유지되며, t=1이면 고주파 영역이 제거되어 i-번째 정점의 위치가 자신의 1-고리 이웃들의 중심 위치로 변형된다.

수식 (1)로부터 알 수 있듯이, 한 정점의 위치는 1-고리 이웃이 스무딩될 지점을 결정하는데 영향을 주기 때문에, 각 정점은 스무딩 연산을 수행한 직후 바로 정점의 위치를 수정하지 않고 그 계산 결과를 따로 저장하고 있어야 한다. 메쉬 스무딩은 모든 정점들의 스무딩 연산이 끝난 후, 각각의 정점에 대해 스무딩 연산의 결과로 저장된 정보를 정점의 위치에 적용함으로써 구현할 수 있다.



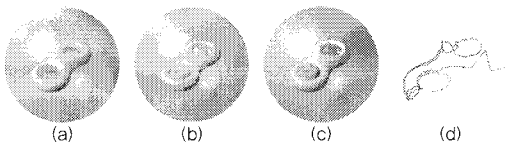
(a) 입력으로 들어온 삼차원 볼륨 메쉬
(b) Laplace 변환을 통해 스무딩된 삼차원 볼륨 메쉬

그림 3. 삼차원 볼륨 메쉬 스무딩

[그림 3]은 정점의 개수가 37813개인 볼륨 메쉬에 대한 메쉬 스무딩을 OpenVolMesh의 동적속성 할당 기능을 통해 수행한 결과이다. 이 경우, 스무딩된 결과를 저장하는 장소 이외의 여타 메모리(extra memory) 사용량은 동적속성의 배열 시작 주소를 저장할 4바이트이다. 그러나, 만일 이를 각 정점에 포인터를 두어 동적 메모리 할당을 통해 구현한다고 가정하면, 각 정점당 포인터를 저장할 4바이트 공간이 필요하기 때문에 주어진 볼륨 메쉬에 대해 총 151,252바이트의 여타 메모리가 필요하다는 점을 주목하기 바란다.

2. 삼차원 볼륨의 CW-셀 분할

임의의 주어진 삼차원 볼륨 메쉬에 대해, 그 위상 구조(topological structure)를 분석하는 것은 볼륨 메쉬 파라메터화(volumetric mesh parameterization), 볼륨 재메쉬화(volumetric remeshing) 볼륨 텍스쳐 매핑(volumetric texture mapping)과 같은 볼륨 메쉬 기반의 응용 분야에서 매우 중요한 작업이다. 이를 위해, 주어진 볼륨 메쉬에 대한 CW-셀 분할(CW-cell decomposition)이 선행되어야 한다. CW-cell은 약한 위상(weak topology) 구조를 표현하기 위한 구조로, 단체(simplex)를 일반화한 개념이다[9]. CW-cell 분할은 주어진 삼차원 물체가 어떤 위상 단위들로 이루어져 있는지를 분석하는 것으로, 각각의 차원에 대한 CW-cell들의 집합으로 주어진 물체의 위상을 표현하는 것이다. (CW-cell 및 CW-cell 분할에 대한 자세한 사항은 [9]를 참조).



(a) 속이 꽉 찬 솔리드(solid) 구에서 2-토러스에 해당하는 부분을 걷어내어 만들어진 삼차원 볼륨 메쉬
(b) CW 3-cell
(c) 각각 다른 색으로 표시된 CW 2-cell들
(d) 각각 다른 색으로 표시된 CW 1-cell들

그림 4. CW-cell 분할의 적용 사례

CW-cell 분할을 통해 주어진 물체에 대한 CW-cell들을 표현하기 위해서는 정점/반에지/에지/반면/셀 등의 각 메쉬 구성요소들이 동적속성 할당을 필요로 하며, 각 구성요소들 간의 인접관계를 효과적으로 접근할 수 있어야 한다. [그림 4]는 OpenVolMesh를 통해 주어진 삼차원 볼륨 메쉬에 대해 CW-cell 분할을 수행한 결과이다.

V. 결 론

본 논문에서는 제네릭 프로그래밍, 메쉬 구성요소들에 대한 동적속성 할당, 배열 기반의 효과적인 자료구

조, OpenMesh와 소스코드 레벨에서의 호환성을 가지는 반면 자료구조 기반의 삼차원 볼륨 메쉬 라이브러리인 OpenVolMesh를 제안하였다. 메쉬 스무딩과 CW-셀 분할과 같은 적용 사례를 통해 OpenVolMesh가 삼차원 볼륨 기반의 콘텐츠 개발에 유용하게 쓰일 수 있음을 보였다.

OpenVolMesh의 현재 버전은 삼차원 볼륨 메쉬 파일의 로딩 및 정점의 위치 변환 연산을 제공한다. 향후, 면 스왑(face swap), 면 붕괴(face collpase) 등과 같은 삼차원 볼륨 메쉬에 대한 다양한 위상 연산자를 지원할 계획이며, 삼차원 볼륨 기반의 콘텐츠 개발 적용된 다양한 사례 수집을 지속할 예정이다. 현재, 정량적이고 수치적인 분석을 통해 OpenVolMesh의 성능을 분석하고, 이를 향상시키기 연구가 진행 중이다.

참 고 문 헌

[1] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "OpenMesh - a generic and efficient polygon mesh data structure," OpenSG Symp. 2002.

[2] http://www.cgal.org

[3] http://vcg.sourceforge.net

[4] K. Weiler, "Edge-based Data Structures for Solid Modeling in Curved-Surface Environments," IEEE CG&A, Vol.5, No.1, pp.21-40, 1985.

[5] K. Lutz, "Using Generic Programming for Designing a Data Structure for Polyhedral Surfaces," In Proc. 14th Annual ACM Symp. On Computational Geometry, 1998.

[6] G. Taubin, "A Signal Processing Approach to Fair Surface Design," SIGGRAPH 95 Proceedings, pp.351-358, 1995.

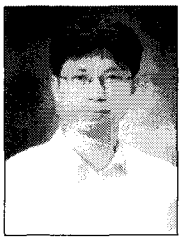
[7] P. Alliez, D. Cohen-Steiner, M. Yvinec M. Desbrun, "Variational Tetrahedral Meshing," SIGGRAPH 2005, pp.617-625, 2005.

[8] W. Sweldens and P. Schroder, "Digital Geometry Processing," SIGGRAPH 2001 course notes.

[9] A. Hatcher, "Algebraic Topology," Cambridge University Press, 2001.

저 자 소 개

김 준 호(Junho Kim) 정회원



- 2000년 2월 : 포항공과대학교 컴퓨터공학과(공학석사)
- 2005년 2월 : 포항공과대학교 컴퓨터공학과(공학박사)
- 2005년 3월 ~ 2005년 10월 : 포항공과대학교 박사후 연구원
- 2005년 11월 ~ 2008년 2월 : 뉴욕주립대(스토니브룩 소재) 포스트닥 연구원
- 2008년 3월 ~ 현재 : 동의대학교 게임공학과 전임강사

<관심분야> : 컴퓨터 그래픽스, 인터렉티브 게임

서 진 석(Jinseok Seo) 정회원

- 2000년 2월 : 포항공과대학교 컴퓨터공학과(공학석사)
- 2005년 2월 : 포항공과대학교 컴퓨터공학과(공학박사)
- 2005년 9월 ~ 현재 : 동의대학교 게임공학과 조교수

<관심분야> : 게임 엔진, 저작 도구, 기능성 게임

오 세 웅(Seiwoong Oh) 종신회원

- 1987년 2월 : 한양대학교 전자공학과(공학석사)
- 1998년 2월 : 오사카대학교 정보공학과(공학박사)
- 2004년 9월 ~ 현재 : 동의대학교 게임공학과 부교수

<관심분야> : 네트워크 가상현실, 온라인게임, 유비쿼터스 컴퓨팅