ISBN 978-89-6211-284-9

COVISE 분석
(Analysis on COVISE)

허 영 주 (popea@kisti.re.kr)

한 국 과 학 기 술 정 보 연 구 원
Korea Institute of Science & Technology Information

목차

그림 차례

표 차례

1. 개요

COVISE(COllaborative VIsualization and Simulation Environment)는 독일의 HLRS에서 개발한 소프트웨어로, 시뮬레이션 후처리(postprocessing) 기능과 가시화 기능을 연계함으로써 계산 제어가 가능하다. COVISE는 처음에는 네트워크상에서 공학자와 과학자들간의 협업이 가능하게 하려는 목적으로 개발됐으며 슈퍼컴퓨터를 기반으로 하는 시뮬레이션, 후처리, 그리고 가시화 기능을 통합 환경에서 제공한다. 이런 목적에 부합되도록 COVISE의 애플리케이션은 여러개의 모듈로 구성돼 있으며, 각 모듈은 서로 다른 형태의 컴퓨터 환경에서 실행할 수 있다.

COVISE의 렌더링 모듈은 파워월(powerwall), 곡면형 스크린, 혹은 케이브(CAVE)를 포함하는 가상환경 시스템을 지원한다.

즉, COVISE는 가상 환경과 협업 네트워크 기능을 지원하는 모듈화된 가시화 소프트웨어 프로그램으로 정의할 수 있다.

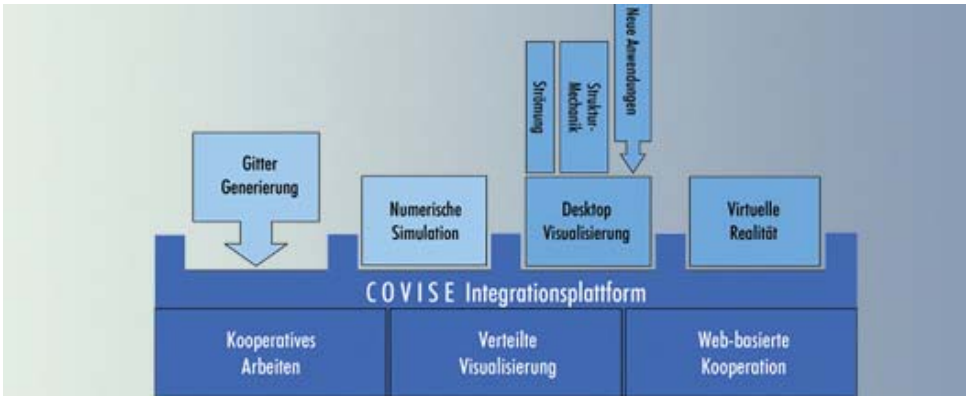
현재 COVISE는 VISENSO라는 회사에서 상용으로 판매, 지원하고 있다.

2. 구조 및 동작

가. 구조

COVISE의 역할은 [그림 2-1]에서 볼 수 있다. COVISE 통합 환경에서는 데이터 시뮬레이션, 가시화 및 몰입형 가상 환경을 포함한 가상 현실 환경과의 연동이 가능하며, 분산 작업 및 협업 기능도 제공한다.

COVISE는 주로 맵에디터(MapEditor)라 불리는 사용자 인터페이스(UI), 컨트롤러(Controller), CRB(COVISE Request Broker) 및 애플리케이션 모듈로 구성된다. 컨트롤러는 COVISE 구조에서 중심적인 역할을 담당하며, 전체 애플리케이션을 조절하는 역할을 수행한다. 컨트롤러는 세션에 포함된 컴퓨터에 분산된 모듈과 실행 애플리케이션의 관리상태를 감시한다. 따라서 애플리케이션 모듈은 실제적으로 컨트롤



[그림 2-1] COVISE의 구조

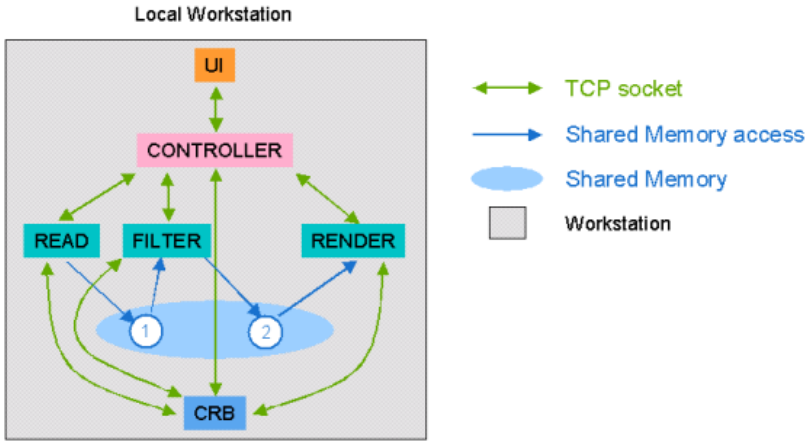
러 및 CRB와만 연결되면 된다. 컨트롤러는 애플리케이션 모듈에게 전체 애플리케이션의 적절한 실행을 보장하는 데 필요한 정보를 제공한다. 애플리케이션 모듈간에 교환되는 데이터는 CRB의 제어하에 저장되기 때문에 애플리케이션 모듈의 구조가 간단해진다.

COVISE 시스템 구조는 C++로 구현돼 있으며, 기본적인 통신 기능은 라이브러리로 제공된다.

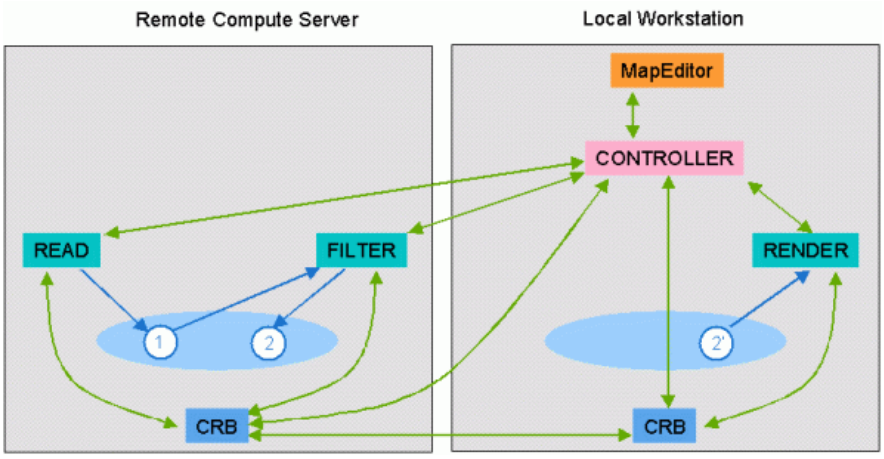
[그림 2-2]는 COVISE 동작시의 구조를 나타낸다. 그림에서 로컬 워크스테이션은 일반적으로 COVISE를 처음 시작한 워크스테이션을 나타낸다. 이 로컬 워크스테이션에 사용자 인터페이스가 나타나고 컨트롤러가 시작되며, 모든 다른 프로세스도 여기에서 생성되는 것이다. 그림에서는 로컬 프로세스만 나타냈는데, 로컬 프로세스는 exec 호출을 통해 생성되고 분산 작업이나 협업 작업을 할 경우, 원격지 컴퓨터의 프로세스는 rexec/rlogin/rsh/ssh 명령으로 생성된다. 사용자는 원격지 모듈 실행을 위해 세션에 새로운 호스트를 추가할 수 있는데, 각 컴퓨터에는 공유 메모리로 구성된 공유 데이터 공간이 존재하게 된다. CRB는 이 공유 데이터 공간을 데이터베이스 관리 방식으로 관리한다.

나. 분산 작업

COVISE에서는 원격지 컴퓨터와의 분산 작업을 지원한다. 분산 작업은 네트워크를 통해 원격지 컴퓨터의 리소스를 사용, 모듈을 분산함으로써 이뤄질 수 있으며, 이때의 제어권은 로컬 워크스테이션의 맵에디



[그림 2-2] COVISE의 동작



[그림 2-3] COVISE에서의 분산 작업

터(MapEditor)에서 가지게 된다.

[그림 2-3]은 COVISE에서 일어나는 분산 작업을 나타내고 있다. 그림에서 볼 수 있는 애플리케이션 모듈은 데이터를 읽는 모듈(READ), 특정 데이터를 걸러내는 모듈(FILTER), 그리고 추출된 데이터를 렌더링해서 보여주는 모듈(RENDER), 이렇게 모두 3가지다. 일반적으로 FILTER 모듈은 CPU 시간과 메모리를 다른 모듈에 비해 많이 소모하는 편이므로 원격지 컴퓨터 서버에서 실행돼야 하며, 이로 인해 FILTER가 가공할 데이터를 읽어들이는 READ 모듈 역시 원격지 컴퓨터에서 실행돼야 한다. 이 작업을 위해 COVISE에서 가장 먼저 실행하는 작업은 컨트롤러(Controller)를 구동하는 것이며, 이어서 사용자 인터페이스 프로세스인 MapEditor와 데이터 관리 프로세스인 CRB가 시작된다. CRB는 세션에 다른 컴퓨터가 추가될 때마다 추가된 컴

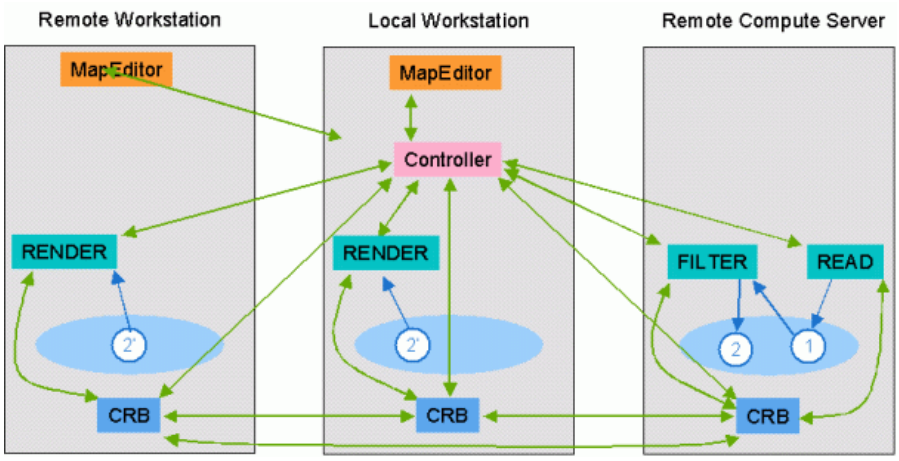
퓨터에서 실행된다. READ 모듈과 FILTER 모듈은 원격지 컴퓨터에서, 그리고 RENDER 모듈은 로컬 워크스테이션에서 실행된다. 그림에서 컨트롤러, 맵에디터, CRB 및 모듈간에 나타나는 초록색 화살표는 TCP 소켓을 나타내고 파란색 화살표는 공유 메모리 접근을 나타낸다.

맵에디터 모듈이 실행되면 컨트롤러는 원격지의 READ 모듈에 start 메시지를 전송한다. READ 모듈은 데이터를 읽어서 공유 메모리에 COVISE 데이터 오브젝트 (1)을 생성해 넣은 뒤, 컨트롤러에게 작업이 끝났음을 알린다. 컨트롤러는 원격지 컴퓨터의 FILTER 모듈에게 데이터 오브젝트 (1)에 대한 정보를 알려준다. 그러면 FILTER 모듈은 데이터 관리 프로세스(CRB)에게 데이터 오브젝트 (1)을 요청하고, 이 데이터로 원하는 추출 작업을 수행해서 공유 메모리에 데이터 오브젝트 (2)를 생성해 넣는다. 그런 다음, 컨트롤러에게 작업이 끝났음을 알리는 메시지를 보낸다. 그러면 컨트롤러는 RENDER 모듈에게 작업을 시작하라는 요청을 보낸다. 그러면 RENDER 모듈은 CRB에 오브젝트 (2)를 요청한다 이 때, 오브젝트 (2)는 로컬 워크스테이션에 존재하지 않으므로 CRB는 오브젝트 (2)를 원격지 컴퓨터로부터 전송해 와서 로컬 워크스테이션의 공유 메모리에 복사해 넣는다(데이터 오브젝트 (2)'). 이렇게 RENDER 모듈은 READ 모듈이 읽고 FILTER 모듈이 가공한 데이터를 화면에 보여줄 수 있다.

다. 협업 기능

멀티유저(multiuser) 세션에서는 참여자가 각자 맵에디터와 렌더러를 실행할 수 있는 협업 기능을 제공한다. 이 때, 다른 참여자를 세션에 추가해준 사용자는 마스터의 역할을 맡게 되며, 맵에디터와 렌더러에 대한 제어권을 갖게 된다. 만약 마스터 사용자가 렌더러에서 카메라의 위치를 바꾸면 다른 참여자의 카메라도 모두 마스터 사용자의 카메라 위치에 동조된다. 마스터 사용자의 권한은 참여자들간에 서로 교환할 수 있다.

협업 세션에서는 맵에디터와 렌더러를 원격지 컴퓨터에서 실행할 수 있다. 한 세션 내의 모든 컴퓨터에서 실행할 수 있는 모듈은 렌더러



[그림 2-4] 멀티유저 세션

모듈밖에 없다.

3. 데이터 타입

COVISE에서 내부적으로 사용되는 데이터 오브젝트 타입은 다음과 같다.

[표 3-1] COVISE의 데이터 타입

데이터 타입	설명
coDistributeObject	모든 데이터 오브젝트 타입에 대한 기본 클래스
coDoAbstractStructuredGrid	Structured Grid에 대한 추상화된 기본 클래스
coDoUniformGrid coDoRectilinearGrid coDoStructuredGrid	Structured Grid 타입
coDoUnstructuredGrid	Unstructured Grid 타입
coDoFloat	스칼라 데이터
coDoVec2 coDoVec3	벡터 데이터
coDoTensor	텐서 데이터
coDoRGBA	컬러값
coDoSet	데이터 오브젝트 그룹을 담는 컨테이너 역할

coDoGeometry	Geometry 데이터를 담는 컨테이너 역할
coDoCoordinates	Geometry 데이터에 대한 추상화된 기본 클래스
coDoPoints coDoLines coDoTriangleStrips coDoPolygons coDoSpheres	Geometry 데이터 타입

COVISE 데이터 모델은 다음의 5가지로 나눠볼 수 있다.

- Structured Grid: 내부적으로 연결 정보를 포함하는 데이터 구조로, x, y, z의 세 방향에 대한 인덱스로 데이터를 나타낸다. 가장 간단한 Structured Grid 데이터 구조로는 큐브(cube) 형태의 데이터 구조를 들 수 있다. 물론 큐브 외의 다른 형태의 Structured Grid 데이터 타입도 제공한다.
- Unstructured Grid: 각각의 구성요소에 대해 vertex의 위치가 명시된 데이터 오브젝트로 구성된다.
- Data 오브젝트: Structured Grid나 Unstructured Grid상의 데이터 요소를 포함하는 오브젝트다.
- Geometry 오브젝트: 다양한 형식으로 정의된 geometry 요소를 포함하며, 렌더러를 통해 가시화할 수 있다. 이 오브젝트에는 Collect 모듈을 사용함으로써 컬러값, normal 값 및 텍스쳐 정보를 추가할 수 있다.
- Containers: 일반적으로는 같은 형식의 오브젝트를 여러개 모아놓은 형태의 데이터 오브젝트다. 따라서 여러 부분으로 구성된 오브젝트나 시간에 따라 변하는 데이터 오브젝트를 포함할 수 있다. Container는 재귀적으로 사용할 수 있다.

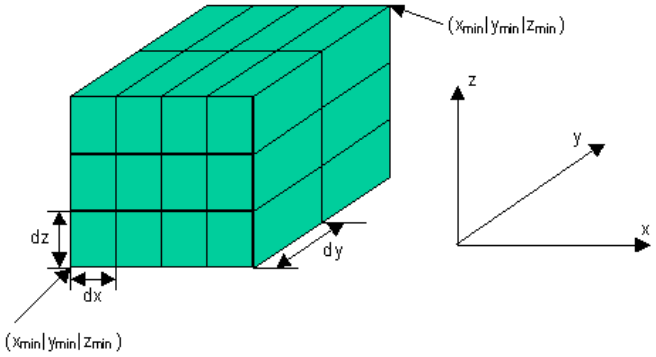
가. Structured Grid

Structured grid는 데이터 사이의 간격이 일정한 정규 직교 그리드(coDoUniformGrid)거나 데이터 사이의 간격에 차이가 있는 정규 직교

그리드(coDoRectilinerGrid), 혹은 curvilinear 그리드(coDoStructured Grid)로 구성된다.

1) Uniform Grid

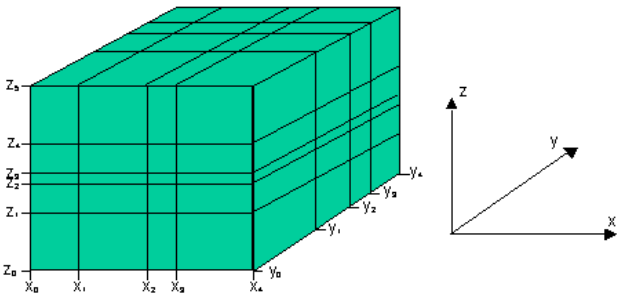
Uniform Grid는 [그림 3-1]과 같이 데이터 사이의 간격이 일정한 정규 직교 그리드를 나타낸다.



[그림 3-1] Uniform Grid

2) Rectilinear Grid

Rectilinear Grid는 Uniform Grid와 유사한 형태의 그리드인데, 데이터 샘플링 간격이 일정치 않다는 특징을 가진다. Rectilinear Grid를 생성할 때는 Grid의 전체 크기뿐만 아니라 각 좌표가 가지는 값도 명시해야 한다.

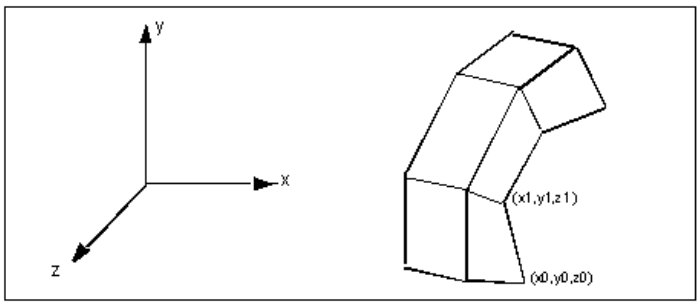


[그림 3-2] Rectilinear Grid

3) Structured Grid

Structured Grid는 6면체로 구성된 그리드로, i x j x k 육면체 구조

를 기본데이터 구조로 한다.



[그림 3-3] Structured Grid

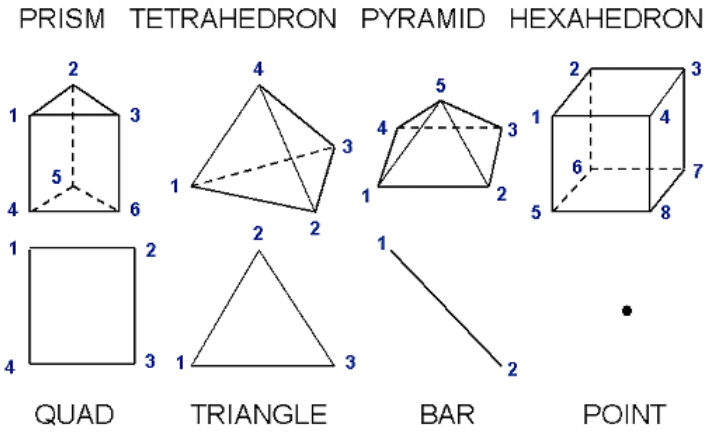
나. Unstructured Grid

Unstructured Grid란 지정된 기본 도형으로 구성된 그리드를 나타낸다. 이런 형태의 그리드는 CFD(Computational Fluid Dynamics)와 FEM(Finite Elements Methods) 분석에 많이 사용된다. COVISE에서 사용하는 Unstructured Grid의 기본 도형은 [그림 3-4]와 같다.

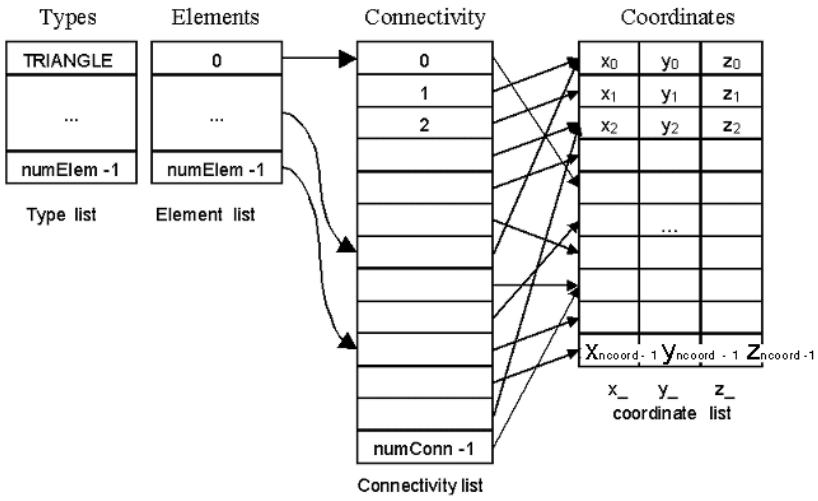
이런 그리드 데이터 구조를 표현하려면 리스트를 여러 개 사용해야 한다.

1) Type 리스트

특정 구성요소의 type을 나타낸다. 반드시 구성요소 각각에 대해 명시해야 한다. 4면체와 4각형은 vertex의 개수는 같지만 그 형태는 다르기 때문이다.



[그림 3-4] Unstructured Grid의 기본 구성 요소



[그림 3-5] Unstructured Grid 포맷

2) Element 리스트

connectivity 리스트에 대한 인덱스를 나타내는데, 특정 구성요소가 connectivity 리스트에서 표현되는 시작 위치를 나타낸다.

3) Connectivity 리스트

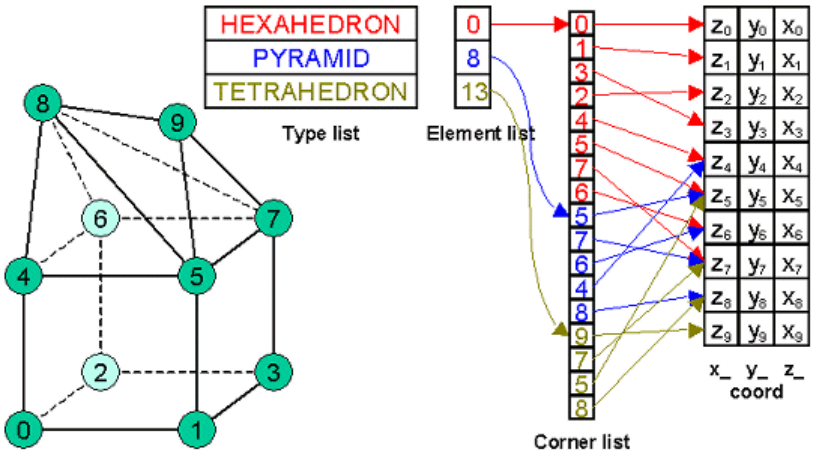
각 구성요소에 소속된 vertex에 대한 인덱스를 나타낸다.

4) X|Y|Z Coordinate 리스트

vertex 좌표를 나타낸다.

이 리스트간의 상호관계는 [그림 3-5]와 같다.

[그림 3-6]은 [그림 3-5]에서 설명한 포맷으로 나타낸 Unstructure



[그림 3-6] Unstructured Grid의 예

d Grid에 대한 사례다.

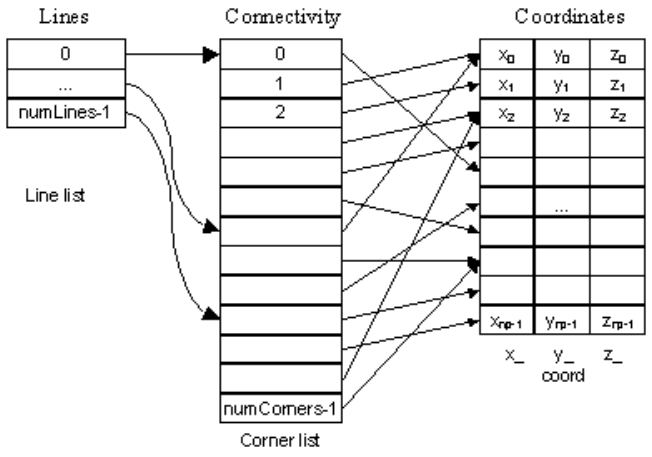
다. 데이터 타입

COVISE에서 사용하는 데이터 타입에는 스칼라 데이터, 2D 및 3D 벡터 데이터, 텐서(tensor) 데이터, 그리고 RGBA 데이터(packed RGBA data)가 있다. COVISE는 이런 데이터 유형을 사용, 데이터값과 기하정보를 저장한다.

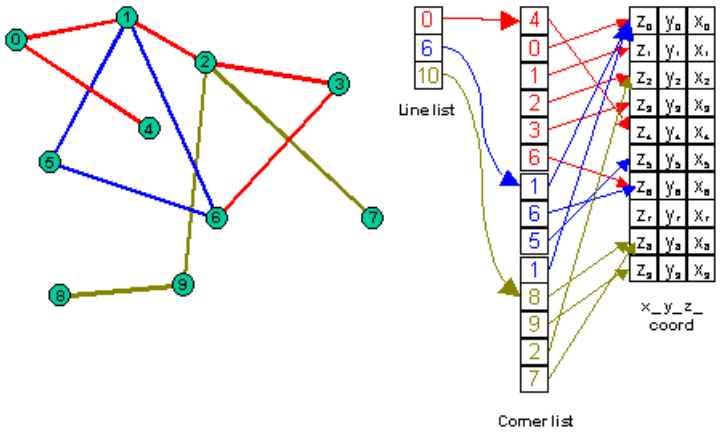
라. Geometry 데이터 타입

Geometry 데이터는 렌더러에서 렌더링할 수 있는 데이터 유형을 가리킨다. Geometry 데이터는 점과 연결 정보로 구성된 geometry 오브젝트를 나타내는데 사용된다.

- Point: point 오브젝트는 3차원 공간에서의 한 점의 위치를 나타내는데 사용되는 수치 리스트며, 렌더링 결과도 한 점으로 나타난다.
- Line: line은 coDoLine 오브젝트로 나타낼 수 있는데, 이 오브젝트의 포맷은 coDoUnstructuredGrid와 매우 유사하다. Line을 나타내는 연결정보는 2개의 리스트로 구성된다. 첫 리스트는 line의 일부에 속한 포인트의 위치를 나타내고, 2번째 리스트는 첫 리스트에 대한 인덱스를 나타낸다. 라인 구조를 나타내는 데이터 구조는 [그림 3-7]과 [그림 3-8]에서 볼 수 있다.

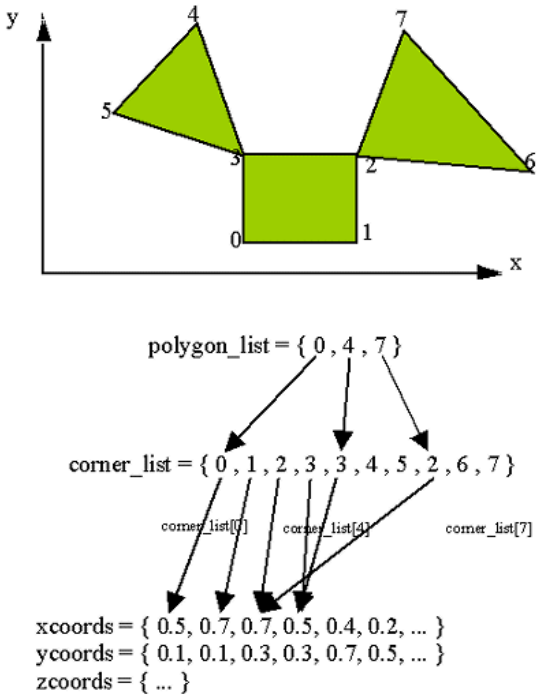


[그림 3-7] Line 구조를 정의하는 리스트

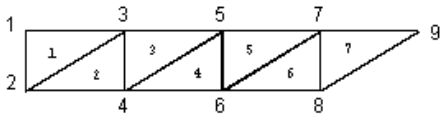


[그림 3-8] Line 구조에 대한 예

- Polygon: Polygon 오브젝트는 라인과 매우 유사한 구조를 보인다. 단지 다른 점은 마지막 point가 자동으로 첫 point와 연결된다는 것이다. Polygon 오브젝트에 대한 예는 [그림 3-9]에서 볼 수 있다.
- Triangle Strip: Triangle strip은 특별한 형태의 polygon이라 볼 수 있으며, 하드웨어 가속 효과를 이용, 매우 효율적으로 렌더링할 수 있다. 하지만 PER_VERTEX 컬러링은 현재 지원되지 않고 있



[그림 3-9] Polygon 구조



[그림 3-10] Triangle strip

다. Triangle strip 역시 line이나 polygon과 매우 유사한 형태로 저장된다.

마. Pixel Image

이미지 형태의 데이터는 Pixel Image 데이터 오브젝트에 저장할 수 있다. 그러나 pixel 버퍼에 이 유형으로 저장된 데이터는 기계에 종속적인 바이너리 데이터로 변환되기 때문에 다른 기계에서 사용하면 데이터가 제대로 저장되지 않을 수도 있다.

Pixel image 형태의 데이터 오브젝트에는 2D Texture가 있다.

바. Text

Text 오브젝트는 모듈간 바이너리 데이터를 운송하는 수단으로 사용할 수 있다. 게다가 OpenInventor 파일 포맷, 또는 VRML 1.0 포맷의 경우에는 Inventor 렌더러가 직접 렌더링할 수도 있다.

사. 컨테이너(Container) 클래스

coDoSet 클래스는 같은 타입의 오브젝트를 여러개 포함하는 컨테이너의 역할을 담당한다. 컨테이너 클래스가 포함하는 오브젝트를 역시 같은 컨테이너 클래스로 지정하는 것도 가능하며, 따라서 데이터의 계층적 구조를 표현하는 것이 가능하다. 그러나 모든 데이터 오브젝트에 접근하려면 CRB를 통해 작업간 커뮤니케이션을 수행해야 하므로 주의해야 한다. 계층구조의 남용은 성능 저하를 불러일으키기 때문이다.

컨테이너 클래스는 다음 2가지 경우에 가장 흔히 사용된다.

- 실제 데이터가 계층 구조로 이뤄져 있는 경우: 일반적으로 CAD 모델에서 추출한 FEM 데이터는 이런 계층 구조로 구성돼 있다.
- 데이터가 시간에 따라 변하는 경우: 이 경우, 시간 단위마다 데이터 오브젝트가 하나씩 존재하게 된다. 이런 데이터는 데이터 오브젝트가 시간에 따라 변하는 전체 데이터의 일부라는 것을 선언하

면서 컨테이너 클래스에 추가하면 된다.

n개의 오브젝트를 가지는 coDoSet 클래스는 다음과 같이 생성한다.

1. 오브젝트 포인터에 대한 배열 생성: coDistributedObject **objects = new coDistributedObjet* [n+1]
2. n개의 오브젝트를 생성해서 object[i]에 할당한다. coDistributedObject는 모든 오브젝트 클래스의 기본 클래스이므로 캐스팅 없이 할당이 가능하다. 그러나 모든 오브젝트의 이름은 서로 달라야 한다.
3. objects[n] = NULL은 리스트의 끝을 의미한다.
4. objects를 파라미터로 set 오브젝트를 생성한다.
5. delete objects[i]로 모든 오브젝트를 삭제한다.
6. delete [] objects로 모든 배열을 삭제한다.

4. COVISE에서의 볼륨 렌더링

COVISE는 텍스쳐 하드웨어를 기반으로 하는 볼륨 렌더링 기능을 지원한다. 이 기법은 전체 볼륨 데이터셋을 디스플레이한다. COVISE의 볼륨 렌더링 기능은 원래 HLRS에서 수행하던 VIRVO(Virtual Reality Volume Rendering) 프로젝트에서 개발된 것이다.

볼륨 데이터를 렌더링하려면 호환되는 데이터가 있어야 한다. COVISE에서 렌더링할 수 있는 볼륨 데이터는 반드시 카테시안 그리드상에 위치해야 하는데, 만약 소스 데이터가 카테시안 그리드상의 데이터가 아닌 경우에는 반드시 적절한 COVISE 모듈을 사용해서 샘플링을 다시 수행해야 한다.

렌더링에 필요한 전체 텍스쳐 메모리의 크기는 복셀의 개수에 복셀당 필요로 하는 바이트 크기를 곱하면 된다. 예를 들어 256 x 256 x 256개의 복셀이 있는 볼륨 데이터가 복셀당 16비트의 메모리를 필요로 한다고 하면, 렌더링에 필요한 전체 텍스쳐 메모리의 크기는 256 x 256 x 256 x 2 바이트 = 32 메가바이트가 된다. 만약 텍스쳐 메모리에 전체 데이터가 로딩되지 못하는 경우에는 스왑 인(swap in)/스왑 아웃(swap out)을 통해 렌더링을 수행하거나 데이터가 아예 로딩되지

않는다. 전자의 경우에는 렌더링에 너무 많은 시간이 소모되고, 후자의 경우에는 볼륨 데이터가 하얗게 표시된다.

COVISE에서 사용되는 볼륨 데이터는 내부적으로 다음 데이터 유형 중 하나로 표현된다.

- 8bit/voxel 스칼라 데이터
- 16bit/voxel 스칼라 데이터 (일반적으로 그래픽스 하드웨어에 의해 가장 중요한 12비트만 디스플레이됨)
- 24bit/voxel RGB 데이터: 각각의 색깔 구성요소에 대해 8비트씩 저장됨
- 32bit/voxel RGB + 스칼라 데이터: 여기에서 스칼라 값은 opacity transfer function과 렌더링에 대한 참조값으로 사용되고, 컬러 요소는 ReadVolume 모듈의 결과값으로 곱해진다.

COVISE에서 볼륨 데이터는 런타임에 계산될 수도 있고 ReadVolume 모듈로 읽어들일 수도 있다. 이 모듈은 2D 슬라이스 이미지뿐만 아니라 표준화된 VIRVO 볼륨 파일도 읽어들인다. 볼륨 파일은 WriteVolume으로 생성할 수 있다.

가. ReadVolume 모듈

ReadVolume 모듈은 여러 유형의 볼륨 데이터를 받아들이는 모듈로 일련의 2D 이미지를 받아들여서 볼륨 데이터셋으로 합칠 수도 있다. ReadVolume 모듈이 지원하는 파일 유형은 다음과 같다.

- rvf: Raw Volume File
- xvf: Extended Volume File
- avf: ASCII Volume File
- tif, tiff: 3D TIF File (2D TIFF는 지원되지 않음)
- dat: Raw volume data (헤더 없음)
- rgb: RGB 이미지 파일 (SGI 8비트 그레이스케일만 지원)
- pgm: Portable Graymap 파일 (P5 바이너리만 지원)
- ppm: Portable Pixmap 파일 (P6 바이너리만 지원)

ReadVolume은 RGB, PGM, 혹은 PPM 파일과 같은 2D 슬라이스 이미지로부터 볼륨 데이터를 생성할 수 있다. 이 기능이 가능하려면 슬라이스 이미지가 오름 순서로 정렬돼 있어야 한다. (ex. IMAGE001.RGB, IMAGE002.RGB, IMAGE003.RGB, etc.) ReadVolume 창에 첫 파일명만 입력하면 나머지 슬라이스 이미지는 자동으로 로딩돼서 볼륨 데이터셋을 생성한다.

나. WriteVolume 모듈

WriteVolume 모듈이 지원하는 파일 유형은 다음과 같다.

- rvf: Raw Volume File
- xvf: Extended Volume File
- dat: Raw volume data (헤더 없음)
- pgm, ppm: Density 혹은 RGB 이미지 (볼륨 데이터 유형에 따라 달라짐)

다. 볼륨 파일

- DAT: 볼륨 데이터 파일

Raw 형태의 볼륨 데이터를 헤더 정보 없이 그대로 저장한 파일이다. 데이터는 1, 2, 3, 혹은 4바이트 크기의 복셀 데이터를 저장할 수 있다. 이런 유형의 파일을 로딩할 때, 프로그램은 자동으로 볼륨의 크기를 파악하려 한다.

이 유형의 파일에서 복셀의 나열 순서는 왼쪽 맨 위 복셀부터 시작해서 오른쪽, 아래, 그리고 뒷 방향으로 진행된다. 각 복셀값에 대해 모든 데이터는 연속적으로 저장되며, 가장 중요한 비트(the most significant bit)가 맨 앞에 나오게 된다(big endian). DAT 파일은 단일 타임 스텝에 대한 데이터만 저장할 수 있다.

- RVF: Raw Volume File

이 유형의 파일은 복셀 데이터 배열에 적절한 헤더를 붙임으로써 생성할 수 있다. 이 헤더는 3 x 2 byte(big endian)로 볼륨의 가로, 세로, 및 깊

이를 나타낸 것이다. 예를 들어 256 x 128 x 127개의 복셀로 구성된 볼륨 데이터의 헤더는 16진수로 10 00 00 80 00 7F가 된다. RVF 파일에서 볼륨 데이터는 복셀당 8비트까지만 표현할 수 있으며, 한 타임 스텝만 저장할 수 있다. 데이터 저장 순서는 DAT 파일과 동일하다.

- XVF: Extended Volume File

DAT이나 RVF보다는 보다 많은 정보를 저장할 수 있지만, 이 파일 역시 수동으로 생성할 수 있다. XVF 파일은 여러개의 볼륨 데이터셋을 한 파일에 저장할 수 있으며, 마찬가지로 다수의 transfer function도 저장할 수 있다. 복셀 당 저장 가능한 데이터 크기는 8~32비트다.

- AVF: ASCII Volume File

AVF 파일은 볼륨 데이터를 ASCII 형태로 표현한 것이다. 이 파일은 헤더 부분과 데이터 부분으로 구성된다.

헤더에서는 데이터 포맷에 대한 정보를 나타낸다. 각 라인은 화이트스페이스로 구분된 형태로 인식자(identifier)와 값을 나타낸다. 각 라인은 인식자 하나와 값 하나만 포함할 수 있다. 이 파일 포맷에서는 transfer function을 저장할 수 없다. 데이터는 헤더 바로 뒤에 나온다. 복셀 데이터값이 나열되며 화이트스페이스 또는 end-of-line 마커로 구분된다. 복셀 나열 순서는 DAT, RVF 및 XVF 파일과 유사하며, 모든 복셀 요소는 연속적으로 저장된다.

5. COVISE의 모듈

COVISE의 모듈은 여러 카테고리로 분류할 수 있다. COVISE에서 지원하는 모듈은 다음과 같다.

- Converter: 이 카테고리에 속한 모듈은 한 오브젝트를 다른 형태의 데이터로 변환하는 역할을 한다. AssembleUsg, DataToGrid, GridToData, Scalar2Vector, StoU 모듈이 여기에 속한다.
- Filter: 전체 데이터로부터 사용자가 필요로 하는 특정 정보를 뽑아내는 모듈이 여기에 속한다. CutGeometry, CuttingLine, GetSubset 등과 같은 모듈이 여기에 속한다.
- Interpolator: 데이터를 여러 형태의 그리드로 바꾸던가 cell 형태

의 데이터를 vertex로 변환하는 모듈이 여기에 속하며, CellToVert, Interpolate, Sample같은 모듈이 있다.

- I/O: 다양한 포맷의 데이터를 읽거나 쓰는 모듈이 여기에 속한다. I/O 모듈은 VI장에서 따로 다루기로 한다.
- Mapper: 추상 데이터를 geometry 형태의 데이터로 변환하는 모듈로, IsoSurface, IsoLines, VectorField같은 모듈이 있다.
- Renderer: 이 모듈은 기하 데이터를 디스플레이하는 역할을 한다.
- Simulation: 온라인 시뮬레이션에 사용되는 모듈로 구성되며, Frieder와 Weather라는 모듈이 있다.
- Tools: 딱히 분류할 수 없는 모듈을 모아놓은 카테고리다. AttAttribute, MinMax, PipelineCollect, ShowGrid같은 모듈 외 다수의 모듈이 있다.
- Tracer: 이 모듈은 벡터 필드에서 파티클의 경로를 생성한다. Tracer, TracerComp 같은 모듈이 여기에 속한다.
- Examples: COVISE 프로그래머를 위한 예제 모듈이다. Cube, ReadObjSimple, PolygonSet 등의 모듈이 속한다.

6. I/O 모듈

여기에서는 COVISE에서 지원하는 I/O 모듈중, 몇가지 중요한 모듈을 나열하겠다.

- ReadABAQUS
- ReadANSYS: 기계학, FLOTRAN, 혹은 열 분석의 결과물인 .rst, .rfl, .rth 파일을 읽는 모듈이다. 기계 분야의 파일에서는 압력과 변형도를, 그리고 열 분석 분야의 파일에서는 열의 흐름을 읽어들여서 렌더링한다. 버전 5.2 이상에서 사용 가능하며 SGI 시스템(IRIX 6.5)에서 테스트했다.
- ReadASCII: ASCII 파일을 읽어서 포인트나 그리드를 생성하는 모듈이다. (이때 스칼라나 벡터 값은 있을수도, 혹은 없을 수도 있다.) 이 모듈은 모든 유형의 데이터를 다룰 수 있으므로 파라미터를 사용해서 모듈을 읽는 동작을 제어한다.

- ReadCFX: CFX-5 데이터 형식을 읽어들이는 모듈이다. 버전 5.2.3부터 사용 가능하며 리눅스와 SGI32에서만 사용 가능하다.
- ReadDx: IBM data explorer용 모듈이다. 버전 5.2.3부터 사용 가능하며 테스트용으로만 사용 가능하다.
- ReadDyna3D: Livermore Software Tecnology Corporation이 제작한 비선형 동역학 분석용 소프트웨어 패키지인 LS-DYNA3D로 생성한 바이너리 플롯 파일인 LS930.PTF 파일을 읽어들이는 모듈이다. 버전 4.5부터 제공되며 SGI 시스템(IRIX 6.2, 6.3, 6.4)에서 테스트된 상태다.
- ReadEnsight: 이 모듈은 Ensight로 작성된 데이터 파일을 읽는 모듈이다. 주로 Ensight6와 Ensight Gold 버전을 지원하는데, Ensight6 데이터는 기하 정보를 담는 *.geo 파일과 다른 필요 파일에 대한 포인터를 포함하는 *.case 파일, 그리고 variable 파일로 구성돼 있으며, 시간에 따라 변하는 데이터도 지원한다.
- ReadIv: 이 모듈은 Inventor 파일을 읽어들이는 모듈이다. 이 파일을 디스플레이하려면 Open Inventor Renderer를 사용해야 한다. 이 모듈은 버전 4.5 이상에서 사용 가능하며 SGI 시스템(IRIX 6.2, 6.3, 6.4, 6.5)에서 테스트가 완료됐다.
- ReadLat: IRIS-Explorer로 작성한 ASCII latice 파일을 읽어들인다.
- ReadMovieBYU: ASCII 포맷으로 작성된 MovieBYU 데이터를 읽어들이는 모듈이다. 그리드 데이터뿐만 아니라 스칼라, 벡터 및 시간에 따라 변하는 데이터를 모두 지원한다. 버전 4.5 이상부터 지원하며 SGI 시스템(IRIX 6.2, 6.3, 6.4)와 HP 시스템(HP-UX 10.20)에서 테스트됐다.
- ReadNasASC: TECPLOT 데이터를 읽어들이는 모듈이다.
- ReadNastran: MSC Nastran이 작성한 바이너리 아웃풋2 버전 70 포맷의 데이터를 읽어들인다. 버전 4.5 이상부터 지원하며 SGI와 HP 시스템에서 테스트된 상태다.
- ReadObj: Wavefront OBJ 포맷을 읽어들이는 모듈이다. 버전 5 이상에서 지원하며 SGI에서 사용 가능하다.

- ReadPAM: DSY와 THP 파일을 읽어들이는 모듈로, DSY 파일로부터는 그리드, 노드, 셀 데이터를 모두 읽을 수 있다. 버전 5.1부터 사용 가능하며 SGI 시스템(IRIX 6.5)에서 테스트됐다.
- ReadPatran: PATRAN 2.5 파일 포맷에 호환되는 데이터를 읽는 모듈이다.
- ReadStl: 이 모듈은 Sandia Stl 포맷으로 정의된 데이터의 surface 정보를 COVISE에서 받아들일 수 있는 포맷으로 변환하는 역할을 수행한다. 이 모듈은 ASCII 파일과 바이너리 파일을 모두 인식할 수 있으며, 대부분의 경우 파일 유형을 자동으로 감지한다. 버전 5.2.3부터 바이너리 입력을 지원한다.
- ReadTascflowTDI: CTX-TASCflow 버전 2.7-2.10으로 작성된 데이터를 인식한다. SGI와 HP에서만 사용 가능하며 LINUX는 지원하지 않는다.
- ReadVolume: 볼륨 데이터 파일을 읽고 uniform grid에 스칼라 값을 채워 넣는다. .dat, .rvf, .xvf, .tiff 파일을 지원하며, .rgb, .pgm, .ppm, .dcm, .dcom, .tif, .tiff같은 2D 이미지 파일로부터 볼륨 데이터를 생성하는 것도 가능하다.
- ReadVTF: ViewTech 파일 포맷의 surface 정보를 COVISE에서 받아들일 수 있는 포맷으로 변환하는 역할을 수행하며, 사용자는 스칼라, 또는 벡터 값중 원하는 값을 결과로 받을 수 있다.
- ReadVTK
- ReadXYZ: 여러개의 타임 스텝에 대해 원자의 위치와 요소 유형을 기록한 XYZ 파일을 읽어서 처리하는 모듈이다.
- ReadZPR: 여러개의 타임 스텝에 대해 원자의 위치와 요소 유형을 기록한 ZPR/CRD 파일을 읽어서 처리하는 모듈이다.
- RW_AVS_TriMesh: INDEX exchange 포맷에서 삼각형 메쉬 정보를 읽거나 해당 포맷으로 작성하는 모듈이다.
- RWCovise: COVISE에서 사용할 수 있는 데이터 오브젝트를 읽거나 이 포맷으로 데이터를 저장하는 모듈이다. 버전 4.5 이상부터 사용할 수 있으며 LINUX 6.4와 IRIX에서 테스트됐다.
- RWCoviseASCII: COVISE ASCII 파일을 읽거나 작성하는 모듈이

다. 버전 5.2.2 이상에서 사용 가능하다.

- RWCoviseGroup: 일단의 COVISE 파일을 읽거나 작성하는 모듈로, 버전 5.1 이상에서 지원한다.
- WriteASC: COVISE 오브젝트 정보를 포함하는 ASCII 파일을 작성하는 모듈이다.
- WritePatran: COVISE 포맷의 geometry 데이터를 PATRAN 포맷으로 변환해서 작성하는 모듈로, 변환된 데이터는 /usr/tmp/object.pat에 디폴트로 저장된다.
- WriteVolume: uniform grid상의 스칼라 데이터, 혹은 3차원으로 저장된 packed RGBA 데이터를 볼륨 데이터 파일로 저장하는 역할을 수행하는 모듈이다. 볼륨 데이터는 .rvf, .xvf, .dat, .ppm, .pgm 포맷으로 저장되며, 이렇게 저장된 데이터는 바로 ReadVolume 모듈로 읽을 수 있다.