1. 개요

1.1 프로젝트 개요

제조업의 제품 개발 과정은 일반적으로 설계, 해석, 실험 검증의 단계로 진행된다. 먼저 CAD 엔지니어가 제품의 형상과 구조를 설계하면, CAE 엔지니어는 이를 바탕으로 Abaqus, Ansys와 같은 기존 CAE(Computer-Aided Engineering) 소프트웨어를 활용하여 구조, 열, 유동 등 제품의 물리적 거동을 해석한다. 이후 실제 시제품 또는 시험 환경에서 실험을 수행하여 제품의 안전성과 성능을 최종적으로 검증한다. 이 프로세스는 제조업에서 제품 신뢰성을 확보하기 위한 핵심 절차이지만, 해석 단계에서 높은 계산 비용과 긴 대기 시간이 발생한다는 한계를 가진다.

본 프로젝트는 이러한 기존 CAE 해석 프로세스의 병목을 완화하기 위해, 전통적인 수치해석 기반 솔버를 AI 기반 솔버로 대체하거나 보완하는 Solver X의 AI Solver 서비스를 웹 기반 플랫폼으로 제공하기 위한 기반 기술을 개발하는 것을 목표로 한다. AI Solver는 기존 해석 데이터와 물리 기반 학습 모델을 활용하여 CAE 해석 결과를 빠르게 예측함으로써 반복적인 설계 검토 과정을 단축하고, 제조업 설계 프로세스의 생산성을 높이는 것을 지향한다.

본인이 담당한 영역은 AI 모델 자체의 연구 및 학습 알고리즘 개발이 아니라, AI Solver가 생성한 해석 결과를 실제 사용자가 웹 환경에서 확인하고 상호작용할 수 있도록 제공하는 애플리케이션 계층의 기술 개발이다. 이를 위해 본 프로젝트에서는 MIE라는 웹 기반 렌더링 엔진 및 프레임워크를 개발하였다. MIE는 CAD 설계 데이터에서 사용되는 B-rep(Boundary Representation) 기반 형상 데이터와 Tetrahedral, Hexahedral, Pyramid 등 다양한 요소로 구성된 Volume Mesh를 웹에서 처리하고 렌더링하기 위한 지오메트리 프로세싱 기능을 제공하며, CAE 해석 결과를 직관적으로 시각화하기 위한 과학적 시각화 기능과 사용자 인터페이스를 포함한다.

결과적으로 본 프로젝트는 AI Solver를 단순한 모델 추론 기능에 머무르게 하지 않고, 제조업 엔지니어가 실제 업무 환경에서 사용할 수 있는 웹 기반 해석 시각화 플랫폼으로 확장하기 위한 기반을 구축하는 데 목적이 있다. MIE는 AI 기반 CAE 서비스의 결과물을 효과적으로 전달하고, 사용자가 해석 데이터를 탐색하며, 설계 의사결정에 필요한 정보를 빠르게 파악할 수 있도록 지원하는 프론트엔드 및 렌더링 프레임워크로 개발되었다.

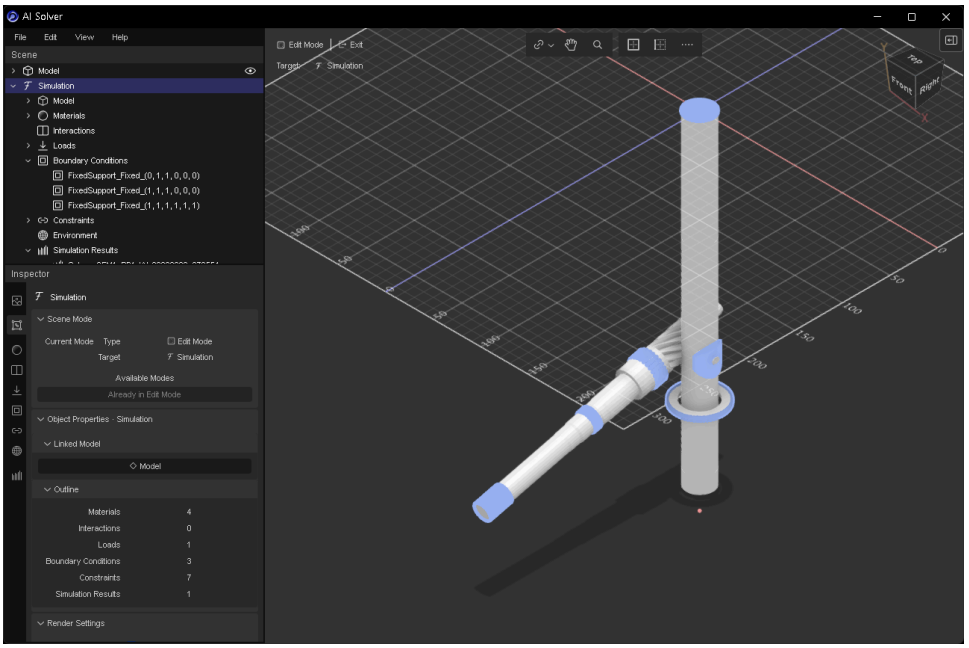

그림 1-1. MIE 웹 기반 CAE 플랫폼 대표 실행 화면

1.2 추진 배경 및 필요성

제조업의 설계 프로세스에서는 제품 형상을 조금만 변경하더라도 다시 메시를 생성하고, CAE 해석을 수행하며, 결과를 검토하는 반복 과정이 필요하다. 특히 자동차, 항공, 중공업, 배터리, 반도체와 같이 높은 안전성과 정밀도가 요구되는 산업에서는 해석 결과의 신뢰성이 매우 중요하기 때문에, 설계 변경마다 복잡한 물리 기반 시뮬레이션을 수행해야 한다. 그러나 기존 CAE 소프트웨어는 고성능 계산 자원과 전문 인력을 필요로 하며, 해석 조건에 따라 수 시간에서 수일 이상의 계산 시간이 소요될 수 있다. 이러한 병목은 빠른 설계 반복과 최적화가 중요한 현대 제조 환경에서 큰 제약으로 작용한다.

AI 기반 솔버는 이러한 문제를 해결하기 위한 새로운 접근 방식이다. 기존 시뮬레이션 데이터와 실험 데이터를 학습한 AI 모델을 활용하면, 복잡한 수치해석 과정을 매번 처음부터 수행하지 않고도 새로운 설계 조건에 대한 해석 결과를 빠르게 예측할 수 있다. 이를 통해 CAE 엔지니어는 초기 설계 검토 단계에서 더 많은 설계안을 빠르게 비교할 수 있으며, 실제 고정밀 해석과 실험 검증은 더 중요한 후보군에 집중할 수 있다. 즉, AI Solver는 기존 CAE를 완전히 대체하는 것뿐만 아니라, 설계 초기 단계의 반복 검토와 의사결정을 가속하는 도구로서 의미를 가진다.

하지만 AI Solver가 실제 산업 현장에서 활용되기 위해서는 모델의 예측 성능만으로는 충분하지 않다. CAE 엔지니어는 해석 결과를 단순한 수치 값이 아니라 3차원 메시, 물리량 분포, 단면 결과, 등고선, 변형 형상 등 다양한 시각적 형태로 확인해야 한다. 또한 대용량 Volume Mesh와 복잡한 해석 결과 데이터를 웹 환경에서 안정적으로 렌더링하고, 사용자가 카메라 조작, Contour Plot, Cutting Plane과 같은 기능을 통해 결과를 탐색할 수 있어야 한다. 따라서 AI Solver를 서비스 형태로 제공하기 위해서는 AI 추론 결과를 엔지니어링 관점에서 해석 가능한 형태로 표현하는 웹 기반 렌더링 엔진이 필수적이다.

MIE는 이러한 필요성에 대응하기 위해 개발되었다. MIE는 CAD 단계의 B-rep 형상 데이터와 CAE 단계의 Volume Mesh 데이터를 함께 다룰 수 있는 지오메트리 프로세싱 기반을 제공하며, CAE 메시 렌더링에 적합한 카메라 설정을 바탕으로 웹에서 과학적 시각화를 수행할 수 있는 렌더링 엔진 및 프레임워크를 제공한다. 또한 저수준 렌더링 기능에 그치지 않고, 사용자가 기능을 쉽게 활용할 수 있도록 UI와 고수준 애플리케이션 레이어를 함께 제공한다. 이를 통해 AI 연구 결과와 실제 사용자 경험 사이의 간극을 줄이고, AI Solver가 제조업 엔지니어의 업무 프로세스에 자연스럽게 통합될 수 있는 기반을 마련한다.

2. 개발 내용 및 결과물

2.1 목표

본 프로젝트의 목표는 Solver X의 AI Solver가 생성하는 CAE 해석 결과를 웹 환경에서 효과적으로 제공하기 위한 렌더링 엔진 및 애플리케이션 프레임워크를 개발하는 것이다. 기존 CAE 소프트웨어는 높은 계산 비용과 긴 해석 시간을 요구하기 때문에, AI 기반 솔버를 통해 해석 시간을 단축하더라도 그 결과를 엔지니어가 실제 업무에서 확인하고 분석할 수 있는 시각화 환경이 필요하다. 본 프로젝트에서는 이러한 요구를 충족하기 위해 웹 기반 렌더링 엔진인 MIE를 개발하였다.

MIE의 주요 목표는 CAD 단계의 B-rep 형상 데이터와 CAE 단계의 Volume Mesh 데이터를 웹에서 처리하고 렌더링할 수 있는 지오메트리 프로세싱 기반을 구축하는 것이다. 또한 Tetrahedral, Hexahedral, Pyramid 등 다양한 요소로 구성된 해석 메시를 안정적으로 표현하고, CAE 결과 데이터에 대해 Contour Plot, Cutting Plane 등 과학적 시각화 기능을 제공하는 것을 목표로 한다.

또한 MIE는 단순한 저수준 렌더러가 아니라, AI Solver 결과를 사용자에게 서비스하기 위한 프레임워크를 지향한다. 이를 위해 CAE 데이터 탐색에 적합한 카메라 조작, 렌더링 상태 관리, 시각화 옵션 제어, 사용자 인터페이스 및 고수준 애플리케이션 레이어를 함께 제공하는 것을 목표로 한다. 이를 통해 사용자는 별도의 데스크톱 CAE 소프트웨어에 의존하지 않고도 웹 브라우저에서 해석 결과를 확인하고 설계 의사결정에 필요한 정보를 탐색할 수 있다.

따라서 본 프로젝트의 개발 범위는 AI 모델 자체의 학습 알고리즘이나 예측 성능 개선이 아니라, AI Solver를 실제 서비스로 제공하기 위한 웹 기반 시각화 및 사용자 경험 계층의 구현에 있다. 최종적으로는 AI 기반 CAE 서비스가 제조업 엔지니어의 설계 검토 프로세스에 자연스럽게 통합될 수 있도록 하는 프론트엔드 기반 기술을 확보하는 것을 목표로 한다.

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용

본 프로젝트의 연구/개발 내용은 Solver X의 AI Solver를 웹 기반 서비스로 제공하기 위한 MIE 렌더링 엔진 및 애플리케이션 프레임워크 구현에 집중되어 있다. MIE는 AI 모델의 학습이나 추론 알고리즘을 직접 개발하는 영역이 아니라, CAD/CAE 데이터와 AI Solver의 해석 결과를 웹 브라우저에서 로딩, 처리, 렌더링, 탐색할 수 있도록 하는 클라이언트 기반 기술이다. 이를 위해 애플리케이션 구조, CAD 형상 처리, CAE Volume Mesh 처리, 해석 결과 시각화, 서비스 리소스 연동, 사용자 인터랙션 및 UI 계층을 함께 개발하였다.

2.2.1.1 웹 기반 CAE 렌더링 프레임워크 구조 설계

MIE는 크게 Core 계층과 UI 계층으로 나뉘는 구조로 설계되었다. Core 계층은 Babylon.js를 기반으로 한 웹 3D 렌더링 엔진, Rust 기반 WebAssembly로 작성된 지오메트리 프로세싱 커널, 그리고 ECS(Entity-Component-System) 기반의 scene 관리 시스템을 제공한다. 이를 통해 CAD/CAE 데이터를 단순히 화면에 표시하는 수준을 넘어, 대용량 메시 처리, 해석 결과 후처리, 객체 선택, 가시성 제어, 렌더링 상태 관리와 같은 기능을 하나의 엔지니어링 데이터 처리 계층 안에서 다룰 수 있도록 하였다.

Core 계층은 렌더링, 데이터 처리, scene 상태 관리를 서로 분리하면서도 동일한 데이터 모델 위에서 동작하도록 구성하였다. Babylon.js는 브라우저에서 안정적으로 3D 렌더링을 수행하기 위한 기반으로 사용되며, Rust/WebAssembly 커널은 JavaScript만으로 처리하기 부담스러운 Volume Mesh 표면 추출, 절단면 생성, 보간과 같은 계산 집약적인 지오메트리 작업을 담당한다. ECS 기반 scene 관리 시스템은 CAD 모델, CAE 해석 모델, 시뮬레이션 조건, 해석 결과와 같은 서로 다른 성격의 객체들을 일관된 방식으로 관리하기 위한 구조이다.

UI 계층은 React를 사용하되, 일반적인 웹 프론트엔드 구현보다 브라우저의 네이티브 기능을 적극적으로 활용하는 방향으로 구현하였다. 최신 브라우저가 제공하는 CSS layout, CSS variable, transition/animation 기능을 활용하여 가능한 많은 레이아웃 계산과 시각적 상태 변화를 JavaScript가 아니라 브라우저의 스타일 엔진에 맡기도록 구성하였다. 이를 통해 복잡한 3D 렌더링과 사용자 인터랙션이 동시에 발생하는 환경에서도 UI 업데이트 비용을 줄이고, Core 계층의 렌더링 및 지오메트리 처리와 UI 계층이 불필요하게 결합되지 않도록 하였다.

아래 그림은 MIE의 기본 실행 화면이다. 좌측에는 scene hierarchy와 inspector가 배치되고, 중앙에는 3D viewport, grid, navigation cube, viewport toolbar가 배치되어 CAD/CAE 모델 탐색과 렌더링 상태 조작을 한 화면에서 수행할 수 있도록 구성하였다.

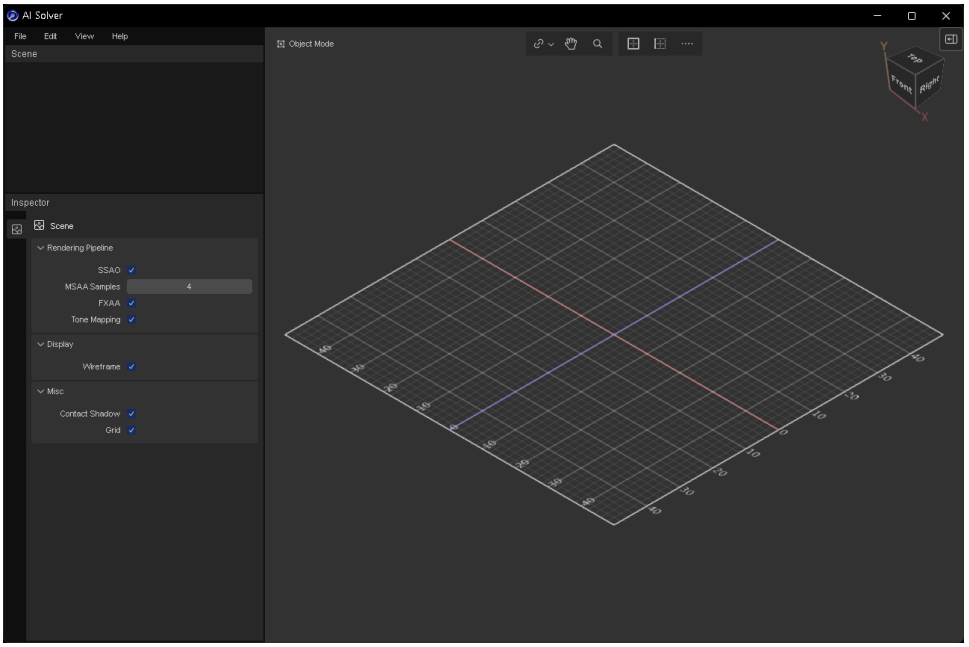

그림 2-1. MIE 기본 viewport 및 UI 구성

2.2.1.2 CAD(B-rep) 형상 데이터 처리

제조업 설계 프로세스에서 출발점이 되는 CAD 데이터 처리를 위해, MIE는 STEP 파일을 웹 환경에서 불러오고 렌더링 가능한 데이터로 변환하는 파이프라인을 구현하였다. STEP 파일은 B-rep 기반 CAD 형상 정보를 포함하므로, 이를 브라우저에서 처리하기 위해 OpenCascade를 WebAssembly로 빌드한 별도 바인딩을 사용하였다. 사용자가 STEP 파일을 로딩하면 TypeScript 기반 STEP 로더가 파일을 바이트 배열로 준비하고, OpenCascade WASM 모듈이 이를 XCAF 문서로 변환한다.

이후 OpenCascade 쪽에서 CAD 형상 트리를 순회하며 solid, shell 등의 형상에 대해 triangulation을 수행한다. B-rep 형상을 웹에서 직접 렌더링하는 것이 아니라, OpenCascade를 통해 face는 triangle mesh로, edge는 polyline으로, vertex는 point 데이터로 변환하고, 이를 다시 Babylon.js의 `Geometry`, `Mesh`, `TransformNode`, `Material`과 같은 3D 엔진 객체로 재구성하는 방식이다. 이 과정에서 CAD 모델의 계층 구조, transform, 색상 정보, 형상 단위의 대응 관계를 최대한 유지하도록 구성하였다.

다만 이 기능은 STEP의 모든 CAD 속성을 완전하게 복원하는 기능이 아니라, 웹 기반 시각화와 선택/탐색에 필요한 형상 계층, 삼각화된 지오메트리, 기본 색상 정보를 중심으로 처리하는 기능이다. 따라서 MIE의 CAD 처리 파이프라인은 브라우저에서 CAD 모델을 엔지니어링 관점으로 검토하고, 이후 CAE 데이터 및 해석 결과와 연결하기 위한 기반 기능으로 볼 수 있다.

아래 그림은 STEP 기반 CAD 모델을 MIE에서 로딩한 결과이다. CAD 파일의 계층 구조는 scene hierarchy에 반영되고, B-rep 형상은 OpenCascade를 통해 triangulation된 뒤 WebGL viewport에서 렌더링 가능한 mesh로 표시된다.

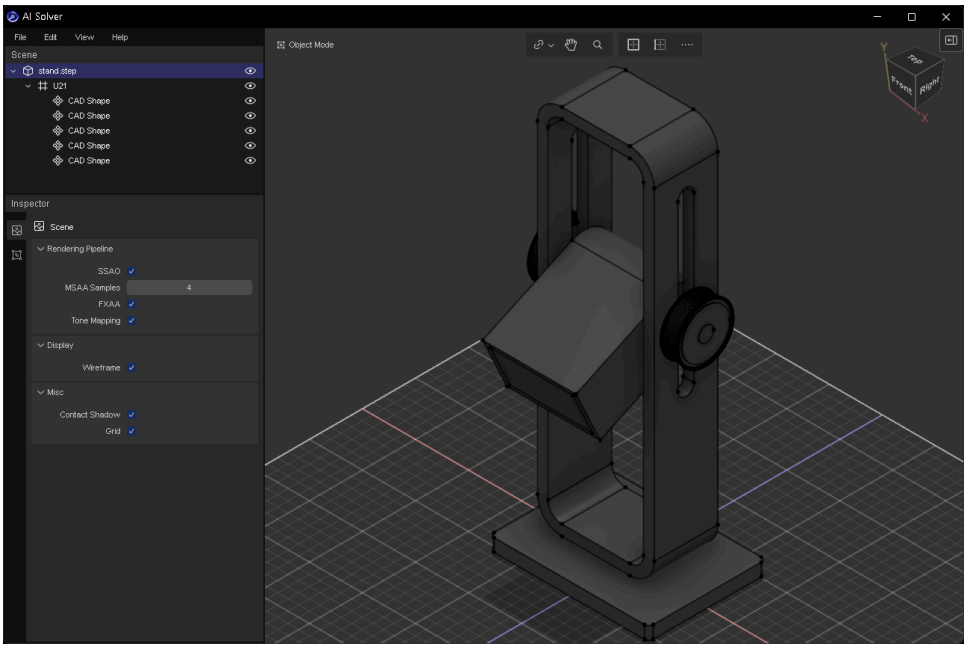

그림 2-2. STEP 기반 CAD 형상 로딩 및 렌더링 결과

2.2.1.3 CAE Volume Mesh 처리 및 Surface Mesh 생성

AI Solver와 CAE 후처리에서 핵심이 되는 해석 메시 처리를 위해, 본 프로젝트에서는 CATF라는 공통 CAE 데이터 포맷을 설계하고 이를 처리하는 파이프라인을 구현하였다. 기존 CAE 환경에서는 Ansys, Abaqus와 같은 소프트웨어가 각자 다른 독자적 입력 파일 및 결과 파일 포맷을 사용하기 때문에, 웹 기반 플랫폼에서 여러 해석 데이터를 일관되게 다루기 어렵다. CATF는 이러한 문제를 해결하기 위해 직접 설계한 공통 포맷으로, node, element, section, material, load, boundary condition, contact, constraint, result data 등을 하나의 일관된 구조로 표현하는 것을 목표로 한다.

MIE는 CATF 모델 데이터를 해석 데이터 보존용 구조와 렌더링용 표면 지오메트리로 나누어 처리한다. 내부적으로는 Bar, Triangle, Quad, Tetrahedron, Pyramid, Penta, Hexahedron 계열의 1차 및 2차 요소를 분류하고, 각 element의 connectivity와 section 정보를 바탕으로 해석 메시 구조를 구성한다. 이 구조는 특정 CAE 소프트웨어의 파일 형식에 직접 종속되지 않고, AI Solver와 웹 렌더링 엔진 사이에서 공통 중간 표현으로 동작한다.

Volume Mesh는 그 자체를 그대로 렌더링하기보다 외부 표면을 추출하여 사용자가 모델의 형상과 해석 결과를 확인할 수 있게 해야 한다. 이를 위해 MIE는 solid 요소의 각 면을 삼각형 인덱스로 전개하고, 동일한 노드 조합이 반대 방향으로 중복되는 삼각형을 내부 공유면으로 판단하여 제거한다. Wireframe 생성 과정에서도 표면에 속하지 않는 내부 edge와 중복 edge를 제거하여 외곽 표면 중심의 시각화가 이루어지도록 하였다. Shell 요소와 line 요소는 각각 표면 삼각형 또는 line primitive로 직접 반영된다.

생성된 표면 지오메트리는 vertex position, node id buffer, index buffer, section별 렌더링 단위 정보로 구성된다. 이때 node id buffer를 별도로 유지하는 이유는 표면 렌더링만 수행하더라도 원래 해석 mesh의 node-wise 결과 데이터와 연결되어야 하기 때문이다. 즉, MIE의 Volume Mesh 처리는 단순한 모델 표시가 아니라, 이후 contour plot, displacement, cutting plane과 같은 해석 결과 시각화를 가능하게 하는 데이터 연결 구조를 포함한다.

2.2.1.4 AI Solver 해석 결과 시각화 파이프라인

AI Solver가 생성한 해석 결과는 사용자가 바로 이해할 수 있는 형태로 표현되어야 한다. MIE에서는 해석 결과를 node-wise property로 관리하고, 사용자가 선택한 물리량, 모드, 시간 단계에 따라 렌더링 데이터를 갱신하는 파이프라인을 구현하였다. 선택된 결과 데이터는 surface mesh의 node id와 연결되며, 이를 통해 원본 Volume Mesh의 물리량을 웹 렌더링 가능한 표면 데이터와 일관되게 매핑할 수 있다.

해석 결과 시각화의 핵심에는 `VertexTensorManager` 기반의 Vertex Tensor 구조가 있다. Vertex Tensor는 CPU 배열 형태의 결과 데이터를 GPU texture로 변환하여 shader에서 정점별 결과 값을 직접 sampling할 수 있도록 하는 구조이다. 이때 시간 단계는 texture depth 방향으로 구성되고, 물리량의 성분 수에 따라 stride와 padding을 고려하여 데이터를 배치한다. 이를 통해 변위, 응력, 변형률, 온도 등 등록된 node property를 GPU 기반 렌더링 파이프라인에서 색상 또는 위치 변화로 사용할 수 있다.

Contour Plot과 Displacement 시각화는 모두 Vertex Tensor를 기반으로 GPU 가속되도록 구현하였다. Contour Plot은 해석 결과의 scalar 값을 LUT 기반 색상으로 변환하며, 사용자가 선택한 출력 채널과 물리량 범위에 따라 색상 매핑을 수행한다. Displacement는 변위 데이터를 GPU vertex shader에서 정점 위치 변화로 반영하여, 사용자가 변형 전후의 형상 차이를 직관적으로 확인할 수 있게 한다. 두 기능 모두 해석 결과 데이터를 매번 CPU에서 새로운 mesh로 재생성하지 않고 GPU texture와 shader를 활용하므로, 웹 환경에서도 높은 응답성을 제공할 수 있다.

반면 Cutting Plane은 실제 절단 단면 지오메트리를 생성해야 하므로 CPU-bound 지오메트리 프로세싱으로 처리하였다. 절단면 생성에는 Rust로 작성된 WebAssembly 지오메트리 프로세싱 커널을 사용하며, Volume Mesh와 평면의 교차 관계를 계산하고 교차점의 물리량을 보간한다. 이 과정은 단순 shader 효과가 아니라 새로운 단면 geometry를 생성하는 작업이기 때문에, Contour Plot이나 Displacement와는 다른 계산 구조가 필요하다.

지오메트리 프로세싱 커널은 SharedArrayBuffer와 JS Worker 기반 멀티스레딩을 활용할 수 있도록 설계하였다. 이를 통해 대용량 Volume Mesh에서 절단면을 계산하거나 해석 결과 데이터를 보간하는 작업을 메인 UI 스레드와 분리하여 처리할 수 있다. 따라서 MIE의 결과 시각화 파이프라인은 GPU가 유리한 색상/변위 렌더링은 Vertex Tensor로 처리하고, CPU 계산이 필요한 절단면 생성은 Rust/WASM 커널과 멀티스레딩으로 처리하는 하이브리드 구조를 가진다. 다만 이 기능은 브라우저에서 CAE solver를 수행하는 것이 아니라, 이미 생성된 해석 결과를 웹에서 후처리하고 시각화하는 파이프라인이다.

아래 그림은 AI Solver 결과를 Contour Plot으로 시각화한 화면이다. 사용자는 inspector에서 물리량, 시간 단계, 색상 범위, wireframe 표시 여부 등을 조작할 수 있으며, viewport에서는 선택된 결과 값이 모델 표면에 색상 분포로 표시된다.

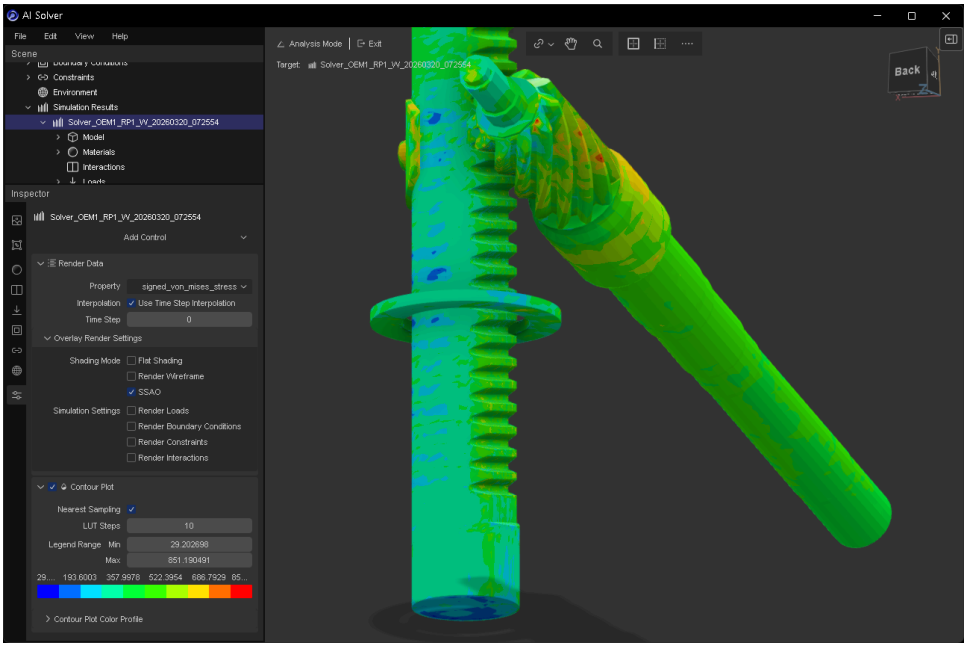

그림 2-3. AI Solver 결과의 Contour Plot 시각화

아래 그림은 Cutting Plane을 적용하여 모델 내부 단면의 결과 분포를 확인하는 화면이다. 절단면의 축 방향과 깊이를 조절하면, Rust/WebAssembly 지오메트리 커널이 Volume Mesh와 평면의 교차 지오메트리를 계산하고, 해당 단면 위에 보간된 해석 결과를 Contour Plot으로 표시한다.

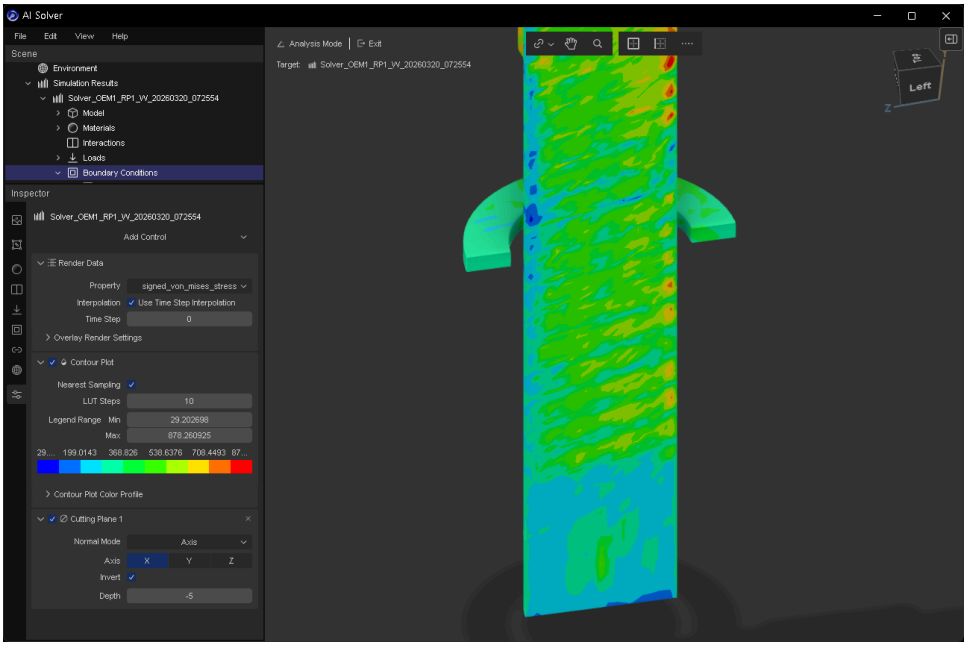

그림 2-4. Cutting Plane 기반 내부 결과 분포 시각화

2.2.1.5 Solver X 서비스 및 리소스 로딩 연동

MIE의 리소스 로딩 계층은 서버 의존성을 최소화하고, 동일한 애플리케이션이 독립형 CAE 데이터 뷰어로도 동작할 수 있도록 설계하였다. 즉, MIE는 Solver X 서버와 연결된 웹 애플리케이션으로 사용할 수 있지만, 서버가 없는 환경에서도 로컬 파일 기반으로 CAD/CAE 데이터를 로딩하고 시각화할 수 있다. 이를 위해 파일 로딩 과정에서 데이터의 출처를 직접 다루는 대신, 스토리지와 리소스 접근 방식을 인터페이스로 추상화하였다.

사용자는 원격 스토리지에 저장된 리소스를 서버 API를 통해 로딩할 수도 있고, 로컬 파일 시스템에서 직접 데이터를 불러올 수도 있다. MIE는 이 두 경우를 서로 다른 코드 경로로 분리하지 않고, 동일한 리소스 로딩 파이프라인에서 처리하도록 구성하였다. 원격 리소스는 download URL, content hash, encoding 정보를 통해 접근하고, 로컬 리소스는 사용자가 제공한 파일 객체를 통해 접근하지만, 이후 단계에서는 동일한 모델 로더와 속성 로더를 거쳐 scene에 반영된다.

이러한 추상화는 STEP, STL, OBJ, CATF와 같은 모델 포맷과 CATF 기반 해석 속성 및 결과 데이터를 일관되게 처리하기 위한 기반이 된다. STEP 파일은 CAD 형상 로더를 통해 렌더링 가능한 CAD 모델로 변환되고, CATF 모델은 CAE 해석 mesh와 analysis model로 변환된다. CATF 속성 및 결과 데이터는 해석 조건과 결과 시각화 기능의 입력으로 연결된다. 서

비스 연동 환경에서는 원격 solver 목록 조회, solver job 생성, 결과 polling, 결과 리소스 resolve가 같은 리소스 로딩 추상화 위에서 동작하므로, MIE는 standalone 뷰어와 Solver X 플랫폼 클라이언트의 역할을 모두 수행할 수 있다.

아래 그림은 MIE에서 원격 Solver X 서버의 AI Solver job을 실행하는 화면이다. 사용자는 구성된 시뮬레이션 입력 조건을 바탕으로 사용할 AI Solver 모델과 결과 이름을 선택하고, 웹 클라이언트에서 solver job을 생성할 수 있다. 실행 중인 job의 상태는 UI에 표시되며, 완료된 결과 리소스는 동일한 로딩 파이프라인을 통해 scene에 연결되어 후처리 시각화에 사용된다. 이를 통해 MIE가 단순한 로컬 viewer가 아니라, 원격 서버에서 실행되는 AI 기반 solver를 실제로 서빙하는 클라이언트 역할을 수행함을 확인할 수 있다.

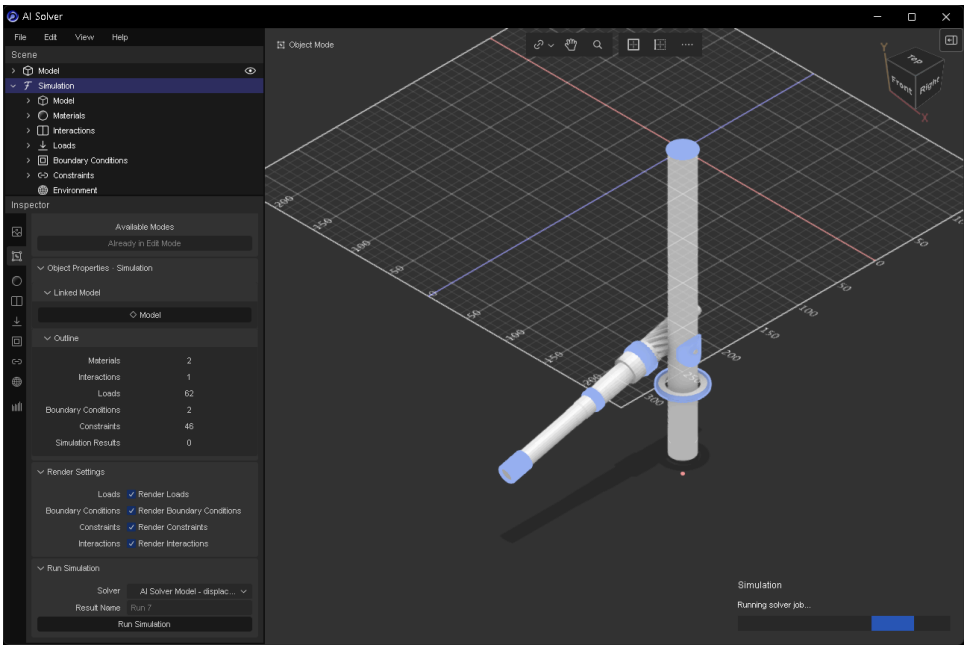

그림 2-5. Solver X 원격 AI Solver 실행 화면

2.2.1.6 CAE 전용 사용자 인터랙션 및 UI 구현

CAE 데이터는 단순한 3D 모델보다 복잡한 계층과 많은 primitive를 가지므로, 모델 탐색과 선택을 위한 전용 사용자 인터랙션이 필요하다. MIE의 UX는 범용 3D 저작 도구인 Blender, Unreal Engine 5, Unity의 scene hierarchy, inspector, viewport 조작 방식을 참고하고, CAE 소프트웨어인 Abaqus와 웹 기반 CAE 플랫폼인 SimScale의 해석 모델 탐색 및 결과 후처리 흐름을 함께 참고하여 설계하였다.

MIE에서는 CAD/CAE 작업에 적합한 orbit camera를 구현하여 중심점 기준 회전, pan, zoom, orthographic projection, view transition을 지원하였다. 또한 navigation cube를 통해 정면, 측면, 상면, 등각 방향 등 엔지니어링 모델 검토에서 자주 사용하는 시점으로 빠르게 전환할 수 있도록 하였다. 이러한 조작 방식은 기존 3D 툴 사용자에게 익숙한 viewport 경험을 제공하면서도, CAE 모델 검토에 필요한 정확한 시점 전환과 모델 탐색을 지원하기 위한 것이다.

선택 기능은 객체 단위 선택과 primitive 단위 선택으로 나누어 구현하였다. 객체 단위 선택은 scene hierarchy와 inspector의 대상 객체를 결정하며, primitive 단위 선택은 face, edge, vertex 단위의 선택을 지원한다. 사용자는 단일 클릭뿐 아니라 viewport에서 box selection 영역을 지정하여 해당 영역에 포함되는 primitive를 선택할 수 있다.

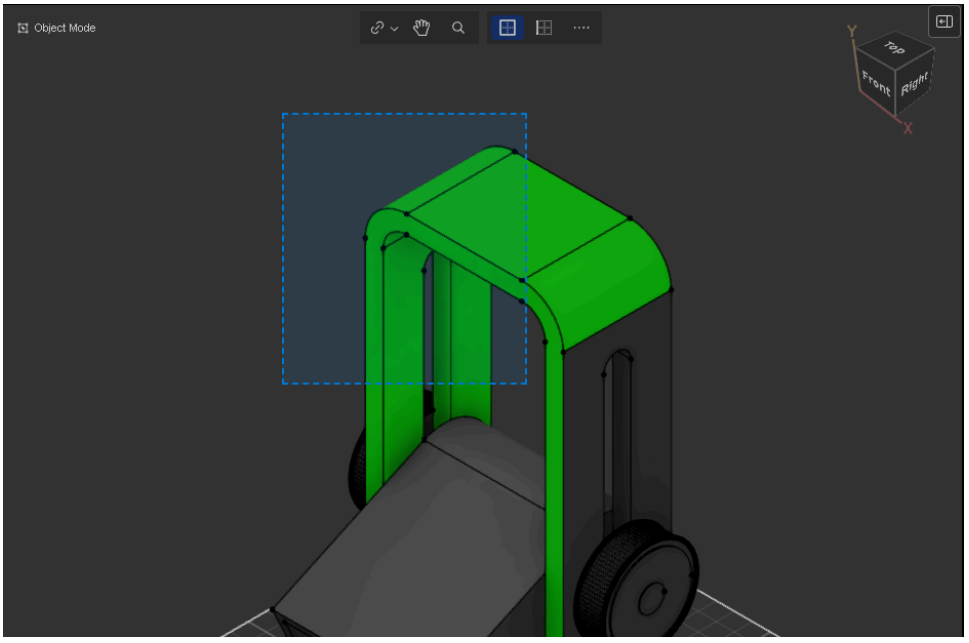

그림 2-6. GPU Picking 기반 Box Selection 화면

이를 위해 MIE는 GPU picking을 사용하여 별도의 render target에 mesh, instance, primitive id를 색상으로 인코딩하고, 사용자의 클릭 또는 박스 선택 영역에 해당하는 픽셀을 읽어 선택 대상을 계산하는 구조를 구현하였다. CPU 기반 picking은 일반적으로 BVH와 같은 가속 구조를 구축한 뒤 ray 또는 영역 쿼리를 수행해야 하므로 메시 크기와 선택 방식에

따라 비용이 달라진다. 반면 GPU picking은 선택 대상 정보를 한 번 렌더링한 뒤 필요한 픽셀을 읽는 방식이므로, 단일 클릭, 다중 포인트 선택, 박스 선택 등 선택 방식이 달라져도 비교적 일정한 비용으로 처리할 수 있다는 성능상의 장점이 있다.

UI는 사용자가 해석 모델과 결과를 탐색하는 워크플로우에 맞춰 구성하였다. Scene hierarchy는 CAD 모델, CAE 해석 모델, 시뮬레이션 조건, 해석 결과의 계층을 표시하고, inspector는 선택된 객체의 속성, CAE property, solver 실행, 결과 표시 옵션을 제공한다. Result control에서는 표시할 물리량, 모드, 시간 단계, contour plot, displacement, cutting plane 등을 조작할 수 있다. 이를 통해 MIE는 AI Solver 결과를 단순히 보여주는 화면을 넘어, 제조업 엔지니어가 웹에서 해석 데이터를 탐색하고 설계 의사결정에 활용할 수 있는 사용자 경험 계층을 제공한다.

아래 그림은 boundary condition 객체를 선택한 화면이다. 선택된 경계 조건에 할당된 node 정보와 Mechanical X/Y/Z 값을 inspector에서 확인할 수 있으며, 이를 통해 시뮬레이션 입력 조건을 시각적으로 검토하고 수치 입력을 다루기 위한 pre-processing 기반을 제공한다.

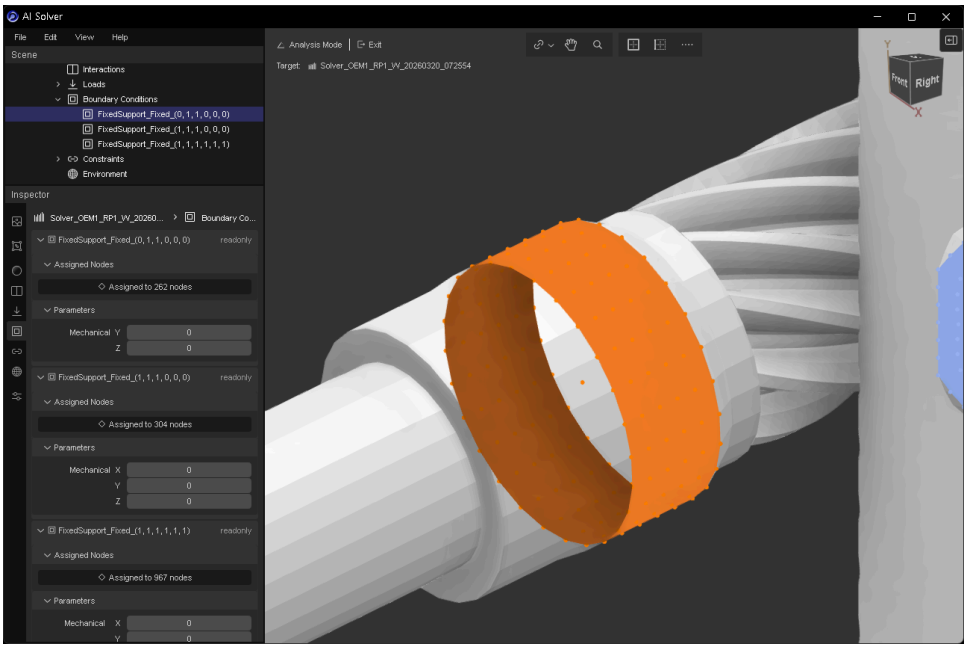

그림 2-7. Boundary Condition 선택 및 수치 확인 화면

2.2.2 시스템 기능 요구사항

본 시스템의 기능 요구사항은 AI Solver의 해석 결과를 웹 환경에서 실제로 탐색하고 분석하기 위해 필요한 기능을 기준으로 정의하였다. 요구사항은 CAD/CAE 데이터 로딩, 메시 처리, 결과 시각화, 사용자 인터랙션, 서비스 연동으로 구분할 수 있다.

ID	기능 요구사항	설명
FR-01	CAD 형상 데이터 로딩	STEP 기반 CAD 형상 데이터를 웹 브라우저에서 불러오고, B-rep 형상을 렌더링 가능한 mesh 데이터로 변환할 수 있어야 한다.
FR-02	CAE 공통 포맷 처리	Ansys, Abaqus 등 서로 다른 CAE 소프트웨어의 독자 포맷 문제를 완화하기 위해 설계한 CATF 포맷을 로딩하고 처리할 수 있어야 한다.
FR-03	Volume Mesh 처리	Tetrahedron, Hexahedron, Pyramid 등 다양한 요소로 구성된 Volume Mesh를 해석 데이터 구조로 보존하고, 렌더링 가능한 표면 지오메트리로 변환할 수 있어야 한다.
FR-04	해석 결과 데이터 연결	해석 mesh의 node-wise 결과 데이터를 표면 mesh와 연결하여, 렌더링 단계에서 물리량을 정확히 매핑할 수 있어야 한다.
FR-05	Contour Plot	응력, 변위, 온도 등 해석 결과의 scalar 또는 vector 성분을 LUT 기반 색상 분포로 시각화할 수 있어야 한다.
FR-06	Displacement 시각화	변위 결과를 모델 형상에 반영하여 변형 형상을 표시하고, 사용자가 scale을 조절할 수 있어야 한다.
FR-07	Cutting Plane	사용자가 지정한 절단면을 기준으로 Volume Mesh의 단면을 생성하고, 절단면 위의 결과 값을 보간하여 표시할 수 있어야 한다.
FR-08	모델 탐색	Orbit camera, pan, zoom, orthographic projection, navigation cube를 통해 대형 CAD/CAE 모델을 효율적으로 탐색할 수 있어야 한다.
FR-09	객체 및 primitive 선택	객체 단위 선택과 face, edge, vertex 단위 primitive 선택을 지원해야 한다.
FR-10	보조 렌더링	Grid, 축 표시, Contact Shadow와 같은 보조 렌더링을 통해 모델의 위치, 방향, 스케일, 공간감을 사용자가 쉽게 파악할 수 있어야 한다.
FR-11	리소스 로딩 추상화	로컬 파일 시스템과 원격 스토리지의 리소스를 동일한 로딩 파이프라인으로 처리할 수 있어야 한다.
FR-12	Solver X 서비스 연동	원격 AI Solver job의 실행 결과를 가져오고, 생성된 해석 결과 리소스를 웹에서 시각화할 수 있어야 한다.

ID	기능 요구사항	설명
FR-13	독립형 뷰어 동작	서버가 없는 환경에서도 로컬 CAE 데이터 뷰어로 동작할 수 있어야 한다.
FR-14	해석 조건 수치 조작	Material property와 boundary condition의 수치 값을 표시하고 수정하여, AI Solver 실행에 필요한 시뮬레이션 입력 조건을 구성할 수 있어야 한다.

2.2.3 시스템 비기능(품질) 요구사항

비기능 요구사항은 웹 환경에서 대용량 3D CAE 데이터를 다루기 위해 필요한 성능, 확장성, 유지보수성, 사용성 관점에서 정의하였다.

ID	품질 요구사항	설명
NFR-01	렌더링 성능	저사양 랩탑 환경에서도 약 50만 triangle 규모의 CAD/CAE 모델을 16ms 이하의 프레임타임으로 렌더링할 수 있어야 한다. 16ms는 60fps 유지를 위한 프레임타임 상한선에 해당하므로, 이를 기준으로 웹 환경에서 탐색 가능한 응답성을 확보해야 한다.
NFR-02	GPU 활용	Contour Plot, Displacement, primitive picking 등 GPU에 적합한 작업은 shader, texture, render target을 활용하여 처리해야 한다.
NFR-03	지오메트리 처리 성능	Cutting Plane, Volume Mesh 표면 추출, 보간 등 CPU-bound 작업은 Rust/WebAssembly와 Worker 기반 병렬 처리를 활용할 수 있어야 한다.
NFR-04	선택 연산 성능	CPU 기반 BVH 구축 및 쿼리에 과도하게 의존하지 않고, GPU Picking을 통해 클릭/다중 선택/박스 선택을 비교적 일정한 비용으로 처리할 수 있어야 한다.
NFR-05	확장성	새로운 CAE 물리량, result type, 파일 포맷, 렌더링 기능을 기존 구조에 큰 변경 없이 추가할 수 있어야 한다.
NFR-06	포맷 독립성	특정 CAE 소프트웨어의 입력/결과 파일 형식에 직접 종속되지 않고, CATF를 중심으로 공통 데이터 모델을 유지할 수 있어야 한다.
NFR-07	독립 실행성	Solver X 서버와 연결된 서비스 환경뿐 아니라, 로컬 파일만 사용하는 standalone 환경에서도 동작할 수 있어야 한다.
NFR-08	브라우저 네이티브 최적화	UI 레이아웃과 시각적 상태 변화는 가능한 한 CSS layout, CSS variable, transition/animation 등 브라우저 네이티브 기능을 활용해야 한다.
NFR-09	사용성	Blender, Unreal Engine, Unity, Abaqus, SimScale 등 기존 3D/CAE 도구 사용자에게 익숙한 viewport, hierarchy, inspector 기반 UX를 제공해야 한다.
NFR-10	유지보수성	Core, UI, 지오메트리 커널, 리소스 로딩, 서비스 연동 계층을 분리하여 각 계층을 독립적으로 개선할 수 있어야 한다.

아래 그림은 Babylon.js Inspector를 통해 렌더링 상태를 확인한 화면이다. 해당 예시에서는 약 12만 faces, 37만 indices 규모의 CAE 모델을 렌더링하면서 60fps를 유지하는 것을 확인할 수 있다. 이는 대용량 CAE 모델을 웹 viewport에서 탐색 가능한 수준의 응답성으로 표시하기 위한 성능 기준을 검증하는 참고 자료로 활용하였다.

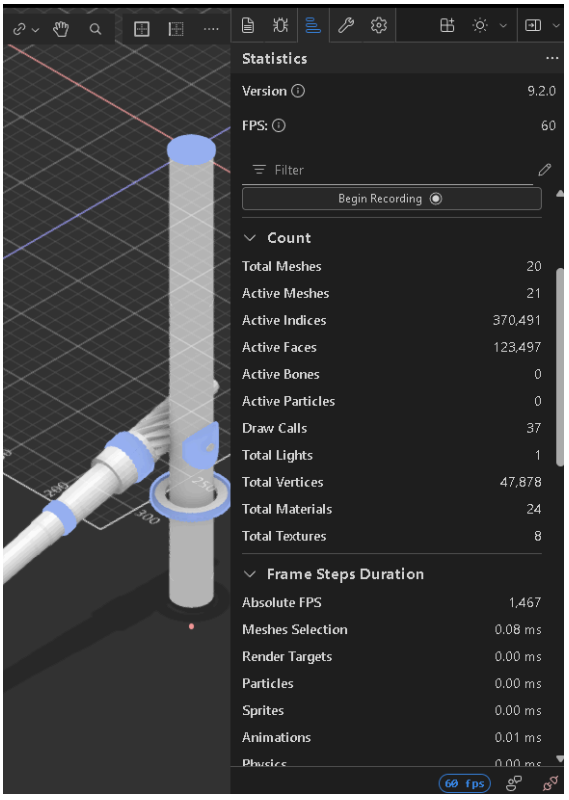

그림 2-8. Babylon.js Inspector 기반 렌더링 성능 확인 화면

2.2.4 시스템 구조 및 설계도

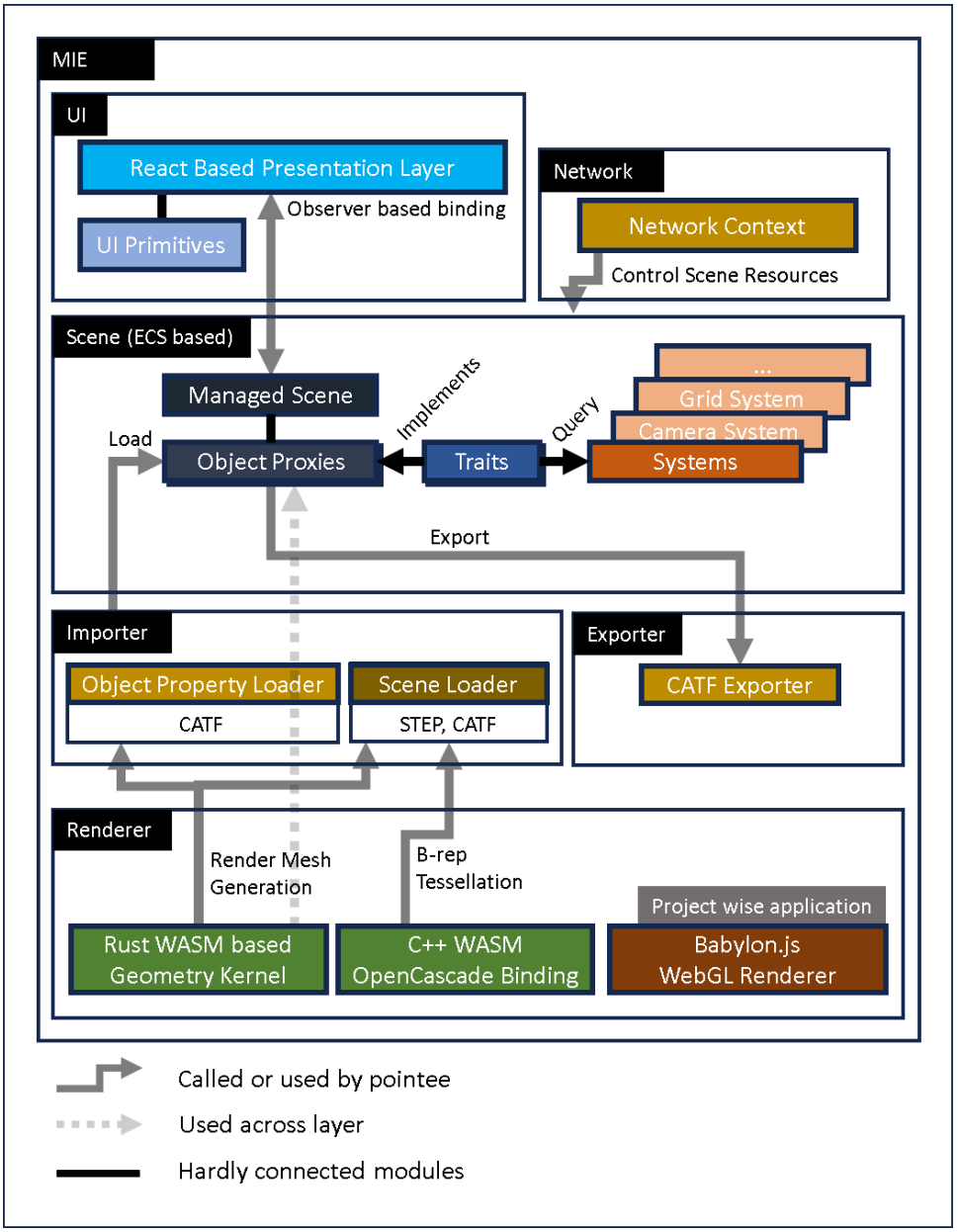

그림 2-9. MIE 전체 아키텍처

MIE의 전체 구조는 UI, Network, Scene(ECS based), Importer/Exporter, Renderer 계층으로 구성된다. 이 중 중심이 되는 계층은 Scene이며, UI와 Network는 사용자의 조작과 서버 상태를 Scene에 반영하고, Importer/Exporter는 외부 데이터와 Scene 사이의 변환을 담당하며, Renderer는 Scene의 데이터를 실제 WebGL 렌더링 및 지오메트리 연산으로 연결한다.

UI 계층은 React 기반 Presentation Layer와 공통 UI Primitives로 구성된다. 이 계층은 도메인 데이터를 직접 소유하기보다 Scene 상태를 표시하고, 사용자의 선택, 카메라 조작, 객체 표시/숨김, 시각화 옵션 변경 등의 입력을 Scene으로 전달하는 역할을 한다. Network 계층은 Solver X 서버 및 원격 스토리지 접근을 Network Context로 추상화하여, 서버 기반 실행 환경에서도 UI와 Scene이 네트워크 구현 세부사항에 직접 의존하지 않도록 한다.

Scene 계층은 ECS 기반 scene 관리 구조로 설계되었다. Scene 관리자는 CAD 모델, CAE 해석 모델, 경계 조건, 하중 조건, 접촉 조건, 해석 결과와 같은 도메인 객체를 하나의 scene state 안에서 관리한다. 객체별 데이터 표현은 Object Proxy 계층에서 담당하고, 선택 가능 여부, 가시성, 속성 표시, 결과 표시와 같은 공통 동작은 Traits로 조합된다. Camera System, Grid System, selection system, visualization system 등 scene 전체에 걸친 기능은 Systems로 분리되어, 개별 객체의 데이터 구조와 전역 동작을 독립적으로 확장할 수 있도록 하였다.

Importer 계층은 외부 CAD/CAE 데이터를 Scene으로 변환하는 진입점이다. Scene Loader는 CAD 형상 또는 CAE 모델 파일을 불러와 scene object로 구성하고, Object Property Loader는 material, load, boundary condition, contact, result data와 같은 해석 속성을 기존 scene object에 연결한다. 반대로 Exporter 계층은 Scene에 구성된 모델과 해석 정보를 CATF Exporter를 통해 공통 CAE 포맷으로 직렬화한다. 이를 통해 MIE는 단순 viewer를 넘어, 서로 다른 데이터 소스에서 들어온 해석 모델을 통합하고 다시 표준화된 형태로 내보낼 수 있는 구조를 가진다.

Renderer 계층은 실제 렌더링과 계산 집약적 지오메트리 처리를 담당한다. Babylon.js WebGL Renderer는 viewport 렌더링의 기반이 되며, 프로젝트별 애플리케이션 계층은 이를 Solver X 서비스 화면과 연결한다. Rust WASM 기반 Geometry Kernel은 Volume Mesh 표면 추출, Cutting Plane, 결과 보간 등 CAE 시각화에 필요한 계산을 수행하고, C++ WASM OpenCascade Binding은 STEP/B-rep CAD 데이터의 import와 triangulation을 담당한다. 따라서 MIE는 웹 UI, ECS 기반 Scene 관리, 데이터 import/export, WASM 기반 지오메트리 커널, WebGL 렌더러가 Scene을 중심으로 결합되는 구조로 설계되었다.

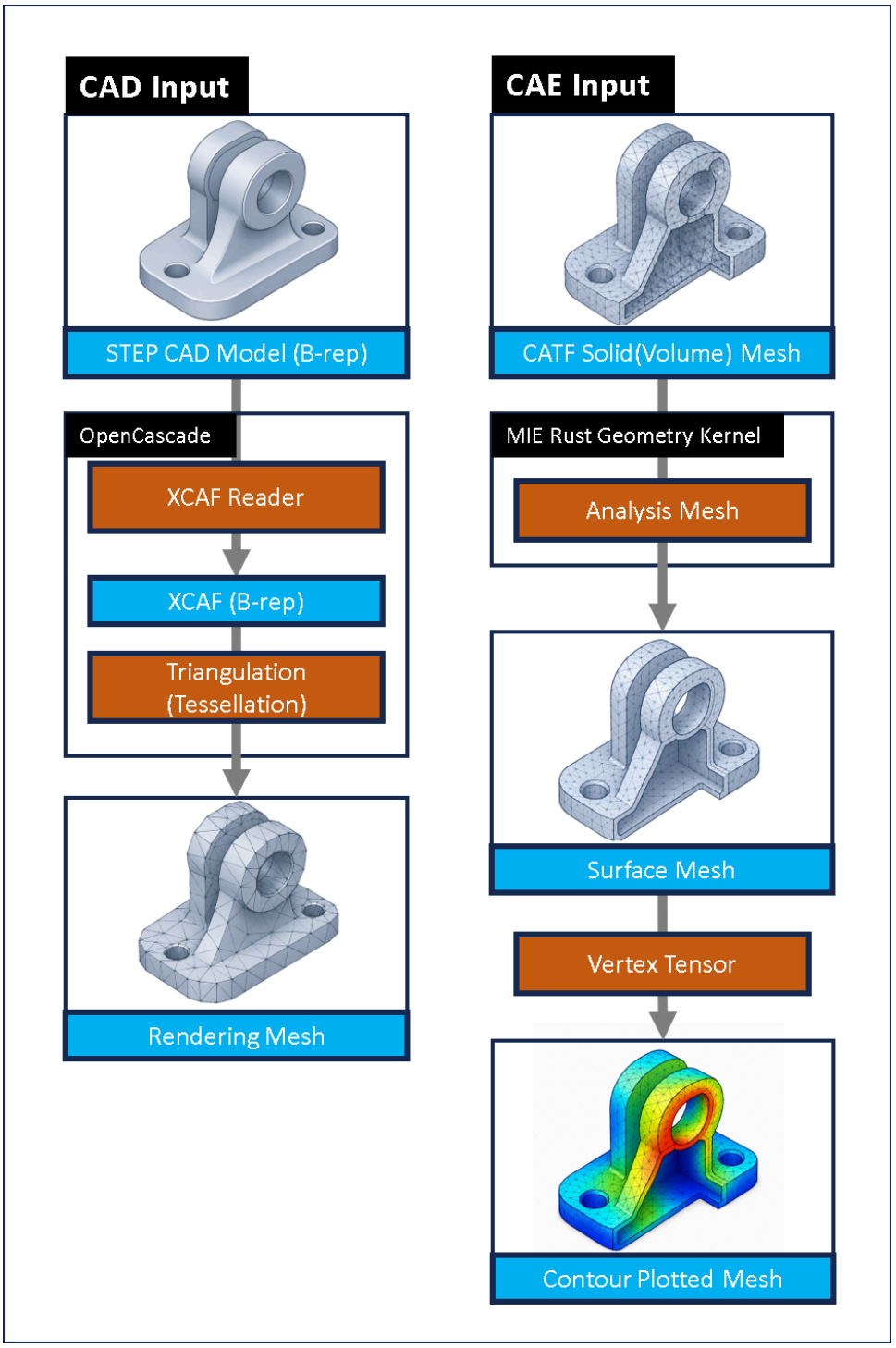

그림 2-10. MIE 데이터 처리 및 시각화 파이프라인

MIE의 데이터 처리 파이프라인은 CAD 입력과 CAE 입력을 분리하여 처리한다. CAD 입력은 제품의 형상 정보를 표현하는 STEP CAD Model(B-rep)을 대상으로 하며, CAE 입력은 해석에 사용되는 CATF Solid(Volume) Mesh를 대상으로 한다. 두 입력은 모두 최종적으로 웹 viewport에서 시각화되지만, 원본 데이터의 의미와 필요한 처리 방식이 다르기 때문에 서로 다른 변환 경로를 거친다.

CAD 입력 경로에서는 STEP 기반 B-rep 모델을 OpenCascade로 로딩한다. OpenCascade의 XCAF Reader는 CAD 파일에 포함된 형상, assembly, 색상, 계층 정보를 읽어 XCAF 기반 B-rep 데이터 구조로 변환한다. 이후 각 face와 edge는 triangulation(tessellation) 과정을 통해 WebGL에서 렌더링 가능한 Rendering Mesh로 변환된다. 이 경로의 목적은 설계 형상을 정확하게 불러오고, 브라우저 환경에서 확인 가능한 triangle 기반 렌더링 메시를 생성하는 것이다.

CAE 입력 경로에서는 CATF에 저장된 Solid(Volume) Mesh를 MIE Rust Geometry Kernel로 전달한다. Geometry Kernel은 tetrahedral, hexahedral, pyramid element 등으로 구성된 해석 메시를 Analysis Mesh로 구성한 뒤, volume 내부에 존재하는 face를 제거하고 외부 경계면만 추출하여 Surface Mesh를 생성한다. 이 Surface Mesh는 단순한 표시용 geometry가 아니라 원래 해석 메시의 node/element 대응 관계를 유지하도록 구성되며, 이후 결과 데이터와 연결되는 기반이 된다.

해석 결과 시각화 단계에서는 node-wise 또는 element-wise 결과 값을 Vertex Tensor 형태로 변환한다. Vertex Tensor는 결과 값을 GPU에서 직접 참조할 수 있는 데이터 표현으로, Contour Plot과 Displacement 렌더링을 shader 기반으로 처리할 수 있게 한다. 따라서 CAE 입력 경로는 Volume Mesh를 Surface Mesh로 변환하는 지오메트리 처리와, 해석 결과를 Vertex Tensor로 변환하는 GPU 기반 시각화 처리가 결합된 구조로 설계되었다.

2.2.5 활용/개발된 기술

본 프로젝트에서는 기존 웹/3D 기술을 활용하는 동시에, CAE 데이터 처리와 과학적 시각화에 필요한 MIE 전용 기술을 개발하였다.

구분	기술	활용 내용
활용 기술	TypeScript	Core 계층, 리소스 로딩, 렌더링 제어, UI 애플리케이션 구현에 사용하였다.
활용 기술	React	Scene hierarchy, inspector, viewport overlay, result control 등 UI 계층 구현에 사용하였다.
활용 기술	UI/빌드 기반 기술	styled-components, Radix UI, Webpack을 활용하여 UI 컴포넌트 스타일링, 접근성 기반 primitive 구성, 번들링 및 개발 환경 구성을 수행하였다.
활용 기술	CSS Layout / CSS Variable / Animation	JavaScript 기반 레이아웃 계산을 줄이고, 브라우저 네이티브 스타일 엔진을 활용하여 UI 상태와 레이아웃을 처리하였다.
활용 기술	Babylon.js	WebGL 기반 3D 렌더링, `Geometry`, `Mesh`, `Material`, camera, shader, render target 등을 관리하는 기반 엔진으로 사용하였다.
활용 기술	OpenCascade	STEP/B-rep CAD 데이터를 브라우저에서 import하고, 렌더링 가능한 triangle/line/point geometry로 변환하는 데 사용하였다.
활용 기술	Rust + WebAssembly	Volume Mesh 표면 추출, Cutting Plane 생성, 결과 보간 등 계산 집약적인 지오메트리 프로세싱 커널 구현에 사용하였다.
활용 기술	Web Worker / SharedArrayBuffer	대용량 지오메트리 연산을 메인 UI 스레드와 분리하고, 멀티스레딩 기반 처리를 수행하기 위해 사용하였다.
개발 기술	CATF	Ansys, Abaqus 등 서로 다른 CAE 소프트웨어의 독자 포맷 문제를 완화하기 위해 설계한 공통 CAE 데이터 포맷이다.
개발 기술	Vertex Tensor	node-wise 해석 결과를 GPU texture로 변환하여 Contour Plot과 Displacement를 shader에서 처리하기 위한 데이터 표현 및 렌더링 구조이다.
개발 기술	GPU Picker	mesh, instance, primitive id를 render target에 색상으로 인코딩한 뒤 픽셀을 읽어 선택 결과를 계산하는 picking 시스템이다. BVH 기반 CPU picking과 달리 선택 방식이 달라져도 비교적 일정한 비용으로 처리할 수 있다.
개발 기술	Contact Shadow 렌더러	모델의 scene bound를 기준으로 shadow용 카메라와 render target을 구성하고, depth shader와 blur postprocess를 통해 모델 하단에 접지감을 주는 그림자를 생성하는 보조 렌더러이다.
개발 기술	Grid 렌더링 시스템	모델의 bound, camera rotation, zoom factor에 따라 grid scale과 위치를 동적으로 조정하고, 축/수치 텍스트를 함께 표시하여 사용자가 모델의 방향과 스케일을 파악할 수 있도록 하는 시스템이다.
개발 기술	Shader Material Extension	Babylon.js 표준 재질에 primitive highlight, Vertex Tensor color output, displacement output 등을 결합하기 위한 shader 확장 구조이다.
개발 기술	Resource Loading Abstraction	로컬 파일 시스템과 원격 스토리지 리소스를 동일한 로딩 파이프라인으로 처리하기 위한 리소스/스토리지 추상화 계층이다.
개발 기술	CAE Viewport UX	Blender, Unreal Engine 5, Unity, Abaqus, SimScale의 UX를 참고하여 CAD/CAE 모델 탐색, 선택, 속성 편집, 결과 후처리에 적합한 viewport 경험을 구성하였다.

2.2.6 현실적 제한 요소 및 그 해결 방안

본 프로젝트는 기존 데스크톱 CAE 소프트웨어가 수행하던 데이터 로딩, 메시 처리, 결과 후처리, 3D 탐색 기능을 웹 브라우저 환경에서 제공하는 것을 목표로 하였기 때문에 여러 현실적 제약이 존재하였다. 주요 제한 요소와 해결 방안은 다음과 같다.

제한 요소	문제점	해결 방안
브라우저 환경의 성능 한계	CAE 모델은 수십만 개 이상의 triangle과 대용량 node-wise result data를 포함하므로, 일반적인 웹 UI 방식으로는 60fps 수준의 응답성을 유지하기 어렵다.	Babylon.js 기반 GPU 렌더링을 사용하고, Contour Plot과 Displacement는 Vertex Tensor를 통해 GPU texture/shader 기반으로 처리하였다.
CPU 기반 지오메트리 처리 비용	Cutting Plane, Volume Mesh 표면 추출, 결과 보간은 단순 렌더링이 아니라 실제 geometry를 계산해야 하므로 JavaScript 단일 스레드에서 처리하기 어렵다.	Rust/WebAssembly 기반 지오메트리 프로세싱 커널을 구현하고, SharedArrayBuffer 및 JS Worker 기반 멀티스레딩을 활용하였다.
브라우저 멀티스레딩 보안 제약	SharedArrayBuffer를 사용한 고성능 멀티스레딩은 일반 웹 페이지에서 항상 사용할 수 있는 기능이 아니며, 브라우저의 cross-origin isolation 조건을 만족해야 한다.	서비스 배포 환경에서 COOP/COEP 등 필요한 보안 헤더를 설정하여 SharedArrayBuffer와 Worker 기반 병렬 처리가 정상적으로 동작하도록 구성하였다.
대용량 mesh 선택 비용	CPU ray picking은 BVH 구축 및 쿼리 비용이 필요하고, 클릭/박스 선택 등 선택 방식에 따라 성능이 달라진다.	GPU Picker를 구현하여 mesh, instance, primitive id를 render target에 인코딩하고 픽셀을 읽는 방식으로 선택 비용을 비교적 일정하게 유지하였다.
CAE 포맷 파편화	Ansys, Abaqus 등 CAE 소프트웨어마다 입력/결과 포맷이 달라 웹 플랫폼에서 일관되게 처리하기 어렵다.	CATF라는 공통 CAE 데이터 포맷을 설계하여 node, element, section, material, boundary condition, result data를 통일된 구조로 표현하였다.
CAD B-rep 처리의 복잡성	STEP 파일은 B-rep 기반 CAD 형상을 포함하므로 JavaScript만으로 안정적으로 import 및 triangulation을 수행하기 어렵다.	OpenCascade를 WebAssembly로 빌드하여 브라우저에서 STEP import와 B-rep triangulation을 수행하도록 하였다.
OpenCascade 라이선스 이슈	OpenCascade는 LGPL 기반 라이선스를 사용하므로, CAD import 기능을 메인 애플리케이션 코드와 강하게 결합하면 배포 및 라이선스 관리가 복잡해질 수 있다.	OpenCascade에 의존하는 C++ embind 모듈을 메인 MIE 레포지토리와 분리하고, 동적으로 로드되는 별도 오픈소스 WebAssembly 모듈로 구현하였다. 이를 통해 메인 애플리케이션과 CAD 커널 바인딩의 배포 경계를 명확히 하였다.

제한 요소	문제점	해결 방안
서버 의존성	Solver X 서비스와 연결되는 구조만 고려하면, 서버가 없는 환경에서 로컬 CAE 뷰어로 활용하기 어렵다.	리소스 로딩과 스토리지를 추상화하여 로컬 파일 시스템과 원격 스토리지를 동일한 파이프라인으로 처리하도록 설계하였다.
UI 렌더링 비용	3D 렌더링과 React UI 업데이트가 동시에 발생하면 브라우저 메인 스레드 부하가 커질 수 있다.	레이아웃 계산과 상태 표현을 CSS layout, CSS variable, transition/animation 등 브라우저 네이티브 기능에 최대한 위임하였다.

2.2.7 결과물 목록

본 프로젝트를 통해 개발된 주요 결과물은 다음과 같다.

결과물	설명
MIE Core 렌더링 프레임워크	Babylon.js 기반 웹 3D 렌더링 엔진, ECS 기반 scene 관리 시스템, 리소스 로딩 추상화 계층을 포함한 Core 프레임워크
CAD(B-rep) 처리 파이프라인	OpenCascade 기반 STEP import 및 B-rep triangulation 파이프라인
OpenCascade WASM 바인딩 모듈	LGPL 라이선스 이슈를 고려하여 메인 레포지토리와 분리하고 동적으로 로드되도록 구성한 C++ embind 기반 CAD 커널 바인딩
CATF 공통 CAE 데이터 포맷	CAE 해석 모델, 해석 조건, 해석 결과 데이터를 일관되게 표현하기 위해 설계한 공통 포맷
Volume Mesh 처리 모듈	Tetrahedron, Hexahedron, Pyramid 등 Volume Mesh를 처리하고 Surface Mesh를 생성하는 지오메트리 처리 기능
Rust/WASM 지오메트리 프로세싱 커널	Cutting Plane, 결과 보간, 대용량 geometry 처리 등 CPU-bound 작업을 위한 WebAssembly 기반 커널
Vertex Tensor 시각화 시스템	node-wise 해석 결과를 GPU texture로 변환하여 Contour Plot과 Displacement를 처리하는 GPU 기반 시각화 시스템
GPU Picker	render target 기반 mesh, instance, primitive 선택 시스템
Contact Shadow 렌더러	모델의 접지감과 공간감을 개선하기 위한 보조 그림자 렌더러
Grid 렌더링 시스템	모델의 크기, 방향, 위치 파악을 돕는 동적 grid 및 축/수치 표시 시스템
Shader Material Extension	Babylon.js 표준 재질에 primitive highlight, Vertex Tensor color output, displacement output 등을 결합하기 위한 shader 확장 구조
CAE Viewport UI	Scene hierarchy, inspector, result control, navigation cube를 포함한 웹 기반 CAE 사용자 인터페이스
Solver X 연동 기능	원격 AI Solver job 결과와 리소스를 웹 클라이언트에서 불러와 시각화하기 위한 연동 기능
Standalone CAE Viewer 기능	서버 없이 로컬 CAD/CAE 파일을 불러와 확인할 수 있는 독립형 뷰어 기능

2.3 기대효과 및 활용방안

본 프로젝트의 기대효과는 AI 기반 CAE Solver를 실제 제조업 설계/해석 프로세스에서 사용할 수 있는 웹 기반 엔지니어링 플랫폼으로 확장할 수 있다는 점에 있다. 기존 CAE 프로세스에서는 CAD 설계 데이터, 해석용 mesh, 재료 물성, 경계 조건, 하중 조건, 접촉 조건, 해석 결과가 서로 다른 소프트웨어와 파일 포맷에 분산되어 있으며, 이를 확인하고 수정하기 위해서는 Abaqus, Ansys와 같은 복잡한 데스크톱 기반 CAE 소프트웨어가 필요하다. MIE는 이러한 데이터를 하나의 scene 구조에서 로딩, 관리, 시각화할 수 있는 기반을 제공함으로써 AI Solver를 실제 사용자가 접근 가능한 서비스 형태로 제공할 수 있게 한다.

첫 번째 기대효과는 AI Solver 결과와 CAE 데이터의 접근성 향상이다. 사용자는 별도의 고가 CAE 소프트웨어를 설치하지 않고도 웹 브라우저에서 CAD 형상, 해석 mesh, 시뮬레이션 조건, contour plot, displacement, cutting plane 등의 데이터를 확인할 수 있다. 이는 AI Solver가 단순한 연구 모델에 머무르지 않고, 제조업 실무자가 사용할 수 있는 제품형 서비스로 제공되기 위한 핵심 기반이 된다.

두 번째 기대효과는 기존 CAE 도구의 사용 흐름을 현대적으로 재해석했다는 점이다. 기존 CAE 소프트웨어는 매우 강력하지만, 범용성과 레거시 워크플로우를 모두 포함하기 때문에 사용 흐름이 무겁고 복잡하다. MIE는 AI Solver 서비스에 필요한 CAD/CAE 데이터 관리, scene interaction, 해석 조건 표시 및 편집, 결과 시각화 기능을 중심으로 인터페이스와 내부 구조를 재구성하였다. 이를 통해 사용자는 설계 모델 확인, 해석 조건 검토, 결과 분석과 같은 핵심 작업을 더 직관적이고 연속적인 흐름 안에서 수행할 수 있다.

세 번째 기대효과는 데이터 포맷 통합과 플랫폼 확장성이다. CATF를 통해 서로 다른 CAE 소프트웨어의 독자적인 입력/결과 포맷을 공통 구조로 표현할 수 있으며, MIE는 이 데이터를 scene 관리 구조와 렌더링 파이프라인에 직접 연결한다. 따라서 향후 다양한 AI Solver, CAD 데이터, 해석 조건, 결과 데이터가 추가되더라도 동일한 프레임워크 위에서 확장할 수 있다.

활용방안으로는 먼저 Solver X의 AI Solver를 위한 pre-processing 및 post-processing 플랫폼으로 사용할 수 있다. Pre-processing 측면에서는 CAD/CAE 모델을 불러오고, material, section, boundary condition, load, contact와 같은 시뮬레이션 입력 조건을 확인하거나 편집하는 용도로 활용할 수 있다. 이 기능은 아직 고도화된 전용 CAE pre-processor 수준은 아니지만, AI Solver 실행에 필요한 입력 데이터를 구성하고 검토하는 기반 기능으로 활용 가능하다.

Post-processing 측면에서는 AI Solver가 생성한 해석 결과를 브라우저에서 바로 확인하는 결과 분석 도구로 활용할 수 있다. 사용자는 contour plot, displacement, cutting plane 등을 통해 응력, 변형, 온도 등 주요 물리량의 분포를 시각적으로 검토할 수 있다. 특히 Volume Mesh 기반 결과를 Surface Mesh 및 Vertex Tensor 기반 시각화 파이프라인으로 연결함으로써, 단순한 파일 표시를 넘어 CAE 후처리 도구로 사용할 수 있다.

또한 MIE는 standalone CAD/CAE 데이터 viewer로 활용될 수 있다. 서버와 연결되지 않은 환경에서도 로컬 파일 시스템의 CAD 또는 CATF 데이터를 불러와 형상, mesh, 해석 조건, 결과 데이터를 확인할 수 있으므로, 데이터 검증, 포맷 변환 확인, 데모, 교육용 도구로 사용할 수 있다.

장기적으로는 MIE를 AI 기반 CAE 플랫폼의 공통 frontend framework로 확장할 수 있다. 현재는 CAD/CAE 데이터 로딩, 시뮬레이션 조건 표시 및 일부 편집, geometry processing, result visualization을 중심으로 구현되었지만, 향후에는 해석 조건 편집 기능, AI Solver 실행 요청, 결과 비교, 설계안 관리, 협업 리뷰 기능을 강화하여 제조업의 설계-해석 프로세스 전반을 연결하는 플랫폼으로 발전시킬 수 있다.

3. 자기평가

본 프로젝트에서 본인은 Solver X의 AI Solver를 실제 사용자가 접근 가능한 웹 기반 CAE 플랫폼으로 제공하기 위한 MIE의 핵심 구조와 렌더링 엔진 개발을 담당하였다. AI 모델 자체의 연구보다는, AI Solver가 생성하거나 사용하는 CAD/CAE 데이터를 브라우저 환경에서 로딩, 관리, 시각화하고, 사용자가 해석 조건과 결과를 직관적으로 확인할 수 있는 애플리

케이션 기반을 구축하는 역할을 수행하였다.

프로젝트 수행 과정에서 가장 의미 있었던 성과는 웹 브라우저를 단순한 UI 실행 환경이 아니라, CAD/CAE 데이터를 처리하고 과학적 시각화를 수행할 수 있는 엔지니어링 플랫폼으로 확장했다는 점이다. Babylon.js 기반 WebGL 렌더링, Rust WebAssembly 기반 지오메트리 처리, OpenCascade 기반 CAD B-rep 처리, ECS 기반 scene 관리 구조를 결합하여 CAD 형상, CAE mesh, 해석 조건, 해석 결과를 하나의 프레임워크 안에서 다룰 수 있도록 구현하였다.

또한 기존 CAE 소프트웨어의 데이터 구조와 사용 흐름을 분석하면서, AI Solver 서비스에 필요한 기능을 중심으로 새로운 구조를 설계하였다. 특히 CATF 공통 포맷, Volume Mesh의 Surface Mesh 변환, Vertex Tensor 기반 결과 시각화, Contour Plot 및 Displacement 렌더링, Cutting Plane 처리와 같은 기능은 단순한 3D 모델 뷰어가 아니라 CAE 후처리와 일부 전처리 기능을 지원하는 도구로 발전시키기 위한 기반이 되었다.

기술적으로는 대용량 CAE 데이터를 브라우저에서 다루기 위해 성능과 구조적 확장성을 지속적으로 고려하였다. CPU에서 처리하기 어려운 결과 시각화는 GPU shader와 texture 기반 구조로 옮기고, 계산량이 큰 지오메트리 처리는 Rust WebAssembly와 Worker 기반 처리로 분리하였다. 이를 통해 웹 환경에서도 수십만 triangle 규모의 모델을 실시간으로 다룰 수 있는 렌더링 성능을 확보할 수 있었다.

다만 프로젝트의 한계도 존재한다. 현재 MIE는 CAD/CAE 데이터 로딩, 시각화, 일부 시뮬레이션 조건 표시 및 편집 기능을 제공하지만, Abaqus나 Ansys와 같은 전문 CAE pre-processor 수준의 완성된 해석 조건 편집 기능에는 아직 도달하지 못하였다. 또한 현재 구현 범위는 구조 해석(Structural Analysis)을 중심으로 구성되어 있으며, 향후에는 Modal Analysis, Fluid Analysis 등 다양한 역학 분야로 지원 범위를 확장할 필요가 있다. 이를 위해서는 각 해석 분야에서 요구하는 데이터 구조뿐만 아니라, Vector Field, Streamline 렌더링, mode shape visualization 등 해당 물리 현상을 이해하기 위한 과학적 시각화 방식도 추가적으로 구현되어야 한다.

또한 장기적으로는 단순히 주어진 시뮬레이션 입력에 대한 해석 결과를 보여주는 것을 넘어, 형상의 최적 설계를 제안할 수 있는 Optimizer 기능도 필요하다. 제조업 설계 프로세스에서 CAE는 단순 검증 도구가 아니라 설계 의사결정을 지원하는 도구로 활용되기 때문에, 사용자가 입력한 설계 조건과 해석 결과를 바탕으로 더 나은 형상 또는 설계 방향을 제안할 수 있어야 한다. 이를 위해서는 형상 파라미터 관리, 설계안 비교, 최적화 결과 시각화, AI Solver와 Optimizer의 연동 구조가 추가적으로 필요하다.

이번 프로젝트를 통해 AI 기술이 실제 산업 도구로 활용되기 위해서는 모델의 정확도뿐만 아니라, 데이터를 입력하고 결과를 해석할 수 있는 소프트웨어 인프라가 매우 중요하다는 점을 체감하였다. 특히 제조업 도메인에서는 CAD, CAE, mesh, material, boundary condition, result field와 같은 데이터가 복잡하게 연결되어 있기 때문에, 이를 안정적으로 표현하고 조작할 수 있는 애플리케이션 구조가 AI Solver의 활용 가능성을 크게 좌우한다는 것을 배울 수 있었다.

향후에는 MIE의 pre-processing 기능을 고도화하고, CATF 포맷과 Solver X AI Solver 연동을 더 안정화하며, 기존 CAE 소프트웨어가 제공하는 폭넓은 기능 범위를 점진적으로 커버하는 AI 기반 CAE 플랫폼으로 발전시키고자 한다. 또한 Modal Analysis, Fluid Analysis와 같은 새로운 해석 영역을 지원하기 위해 Vector Field, Streamline, mode shape visualization 등 과학적 시각화 기능을 확장하고, Optimizer 기반 설계안 제안, 결과 비교, 설계안 관리, 협업 리뷰 기능을 추가하는 것이 목표이다. 렌더링 측면에서는 향후 WebGPU 기반 고성능 렌더링으로 확장할 가능성을 고려하여, 현재 WebGL 실행을 위한 GLSL shader와 WebGPU 대응을 위한 WGSL shader를 동일한 shader logic에 대해 병행 작성하고 있다. 이는 구현과 유지보수 비용이 늘어나는 부담이 있지만, 장기적으로 WebGL과 WebGPU를 모두 고려한 렌더링 구조를 확보하기 위한 설계 선택이다.

4. 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	사용자매뉴얼	Abaqus/CAE User's Guide	https://docs.software.vt.edu/abaqusv2024/English/SIMACAECAERefMap/simacae-c-ov.htm	2024	Dassault Systèmes SIMULIA	공개 접근한 Abaq 2024 문
2	키워드레퍼런스	Abaqus Keywords Guide	https://docs.software.vt.edu/abaqusv2024/English/SIMACAEKEYRefMap/simakey-c-ov.htm	2024	Dassault Systèmes SIMULIA	공개 접근한 Abaq 2024 문
3	사용자매뉴얼	Fluent User's Guide	https://ansyshelp.ansys.com/public/Views/Secured/corp/v251/en/flu_ug/flu_ug.html	2025	Ansys, Inc.	Ansys F 2025 R1 Docume
4	이론문서	Ansys Fluent Theory Guide	https://ansyshelp.ansys.com/public/Views/Secured/corp/v251/en/pdf/Ansys_Fluent_Theory_Guide.pdf	2025	Ansys, Inc.	Ansys F 2025 R1 Docume
5	공식문서	SimScale Documentation	https://www.simscale.com/docs/	N/A	SimScale GmbH	N/A
6	사용자매	SimScale Tutorials and User Guides	https://www.simscale.com/docs/	N/A	SimScale GmbH	N/A

번호	종류	제목	출처	발행년도	저자	기타
	뉴얼					
7	공식문서	Babylon.js Documentation	https://doc.babylonjs.com/	N/A	Babylon.js	N/A
8	소스코드	BabylonJS/Babylon.js Source Code	https://github.com/BabylonJS/Babylon.js	N/A	Babylon.js Contributors	GitHub Reposit
9	개발자문서	Blender Developer Documentation	https://developer.blender.org/docs/	N/A	Blender Foundation	N/A
10	공식문서	Open CASCADE Technology Documentation	https://dev.opencascade.org/doc/overview/html/index.html	N/A	Open Cascade SAS	N/A
11	소스코드	Open-Cascade-SAS/OCCT Source Code	https://github.com/Open-Cascade-SAS/OCCT	N/A	Open Cascade SAS	GitHub Reposit

5. 부록

5.1 테스트 케이스

번호	대상 형상	테스트 목적	입력 데이터 규모	수행 내용	결과	로드 시간	성능 지표	비고
TC-01	HL Mando Rack & Pinion	CATF 기반 CAE 형상 로딩, Surface Mesh 생성, WebGL 렌더링 성능 확인	Nodes: 14,731 / Elements: Tetra4 49,472, Penta6 5,176, Hex8 118,994	Rack & Pinion 형상 정보를 로드하고, Volume Mesh로부터 Surface Mesh를 생성한 뒤 MIE viewport에서 렌더링하였다.	형상 로딩, Surface Mesh 생성, 렌더링에 성공하였다.	1,951ms	Absolute FPS: 2,500 / Actual FPS: 60	N/A
TC-02	Hyundai WIA Gearbox	CATF 기반 대규모 CAE 형상 로딩, Surface Mesh 생성, WebGL 렌더링 성능 확인	Nodes: 130,898 / Elements: Tetra4 530,659	Gearbox 형상 정보를 로드하고, Tetra4 기반 Volume Mesh로부터 Surface Mesh를 생성한 뒤 MIE viewport에서 렌더링하였다.	형상 로딩, Surface Mesh 생성, 렌더링에 성공하였다.	1,351ms	Absolute FPS: 1,303 / Actual FPS: 60	N/A
TC-03	LED Stand CAD Model	CAD 형상 로딩, B-rep triangulation, WebGL 렌더링 성능 확인	CompSolid: 5 / Surface Mesh Triangle Count: 29,355	LED 스탠드 CAD 모델을 로드하고, B-rep 형상을 triangulation하여 Surface Mesh를 생성한 뒤 MIE viewport에서 렌더링하였다.	CAD 형상 로딩, Surface Mesh 생성, 렌더링에 성공하였다.	2,851ms	Absolute FPS: 2,091 / Actual FPS: 60	N/A

5.2 렌더러 구현에 대한 기술 문서

5.2.1 Vertex Tensor 기반 Contour Plot 및 Displacement 렌더링 기술 문서

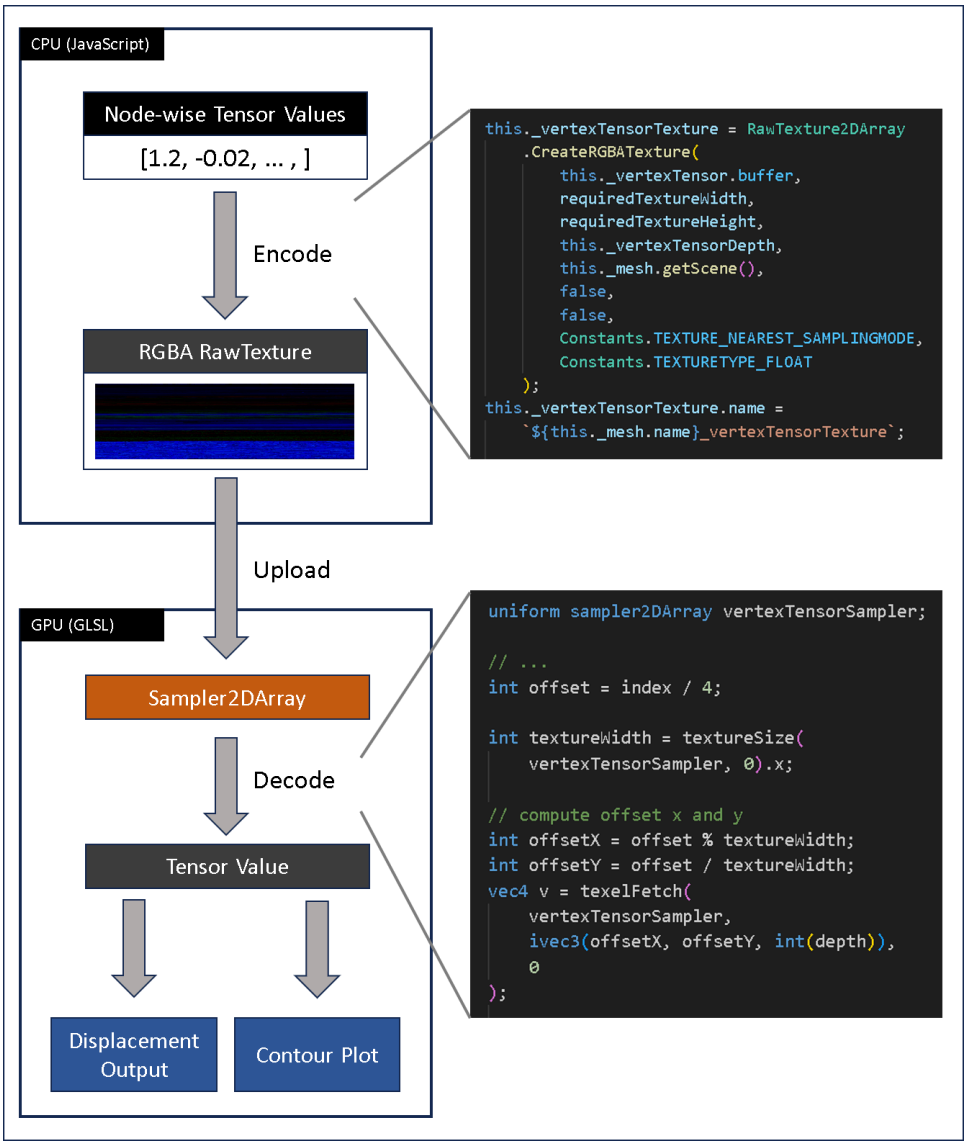

그림 5-1. Vertex Tensor 데이터 인코딩 및 GPU 시각화 구조

Vertex Tensor는 node-wise(vertex-wise) 해석 결과를 GPU에서 직접 읽을 수 있도록 설계한 MIE의 과학적 시각화용 데이터 표현이다. 구조 해석 결과는 각 node마다 스칼라, 3차원 벡터, 6성분 텐서 등 다양한 차원의 값을 가질 수 있으며, 시간 단계 또는 해석 모드에 따라 여러 depth를 가진다. 이러한 데이터를 매 프레임 CPU에서 mesh geometry로 다시 변환하면 비용이 크고, vertex attribute로 모두 전달하려고 하면 WebGL에서 사용할 수 있는 attribute 개수 제한에 쉽게 도달한다. 특히 응력, 변형률, 변위처럼 여러 성분을 가진 물리량을 동시에 다루는 CAE 시각화에서는 attribute 기반 전달 방식이 확장성 측면에서 불리하다.

이 문제를 해결하기 위해 Vertex Tensor는 Babylon.js의 Skeleton 및 MorphTarget 구현에서 사용되는 texture 기반 데이터 전달 방식을 참고하였다. Babylon.js는 bone matrix나 morph target deformation 정보를 uniform 배열 또는 vertex attribute만으로 전달하지 않고, 필요한 경우 texture에 인코딩한 뒤 shader에서 vertex id를 기준으로 texel을 읽어 복원한다. 이는 Compute Shader나 Storage Buffer를 사용할 수 없는 WebGL 환경에서도 대량의 per-vertex 데이터를 효율적으로 GPU에 전달할 수 있는 방식이다. MIE의 Vertex Tensor 역시 동일한 방향으로, JavaScript의 Float32Array에 저장된 해석 결과를 RGBA float texture로 인코딩하고, 이를 shader에서 sampler2DArray로 읽어 시각화에 사용한다.

구현상 Vertex Tensor의 중심에는 `VertexTensorManager`가 있다. 이 관리자는 mesh에 연결되어 해당 mesh의 결과 데이터를 보관하고, `RawTexture2DArray.CreateRGBATexture`를 이용해 `Float32Array` 기반 tensor buffer를 GPU texture로 업로드한다. texture의 width와 height는 엔진의 `maxTextureSize`를 고려하여 계산되며, 시간 단계 또는 mode는 2D array texture의 depth 방향으로 배치된다. 또한 texture가 RGBA 단위로 값을 저장하기 때문에, stride가 3 또는 6처럼 4의 배수가 아닌 경우 padding을 추가하여 shader에서 안정적으로 읽을 수 있도록 구성하였다. 이 구조는 1성분, 2성분, 3성분, 4성분, 6성분 결과 데이터를 모두 처리할 수 있도록 설계되었다.

해석 결과 데이터가 로딩되면 `SimulationResultRenderDataStateManager`는 원본 node data의 배치를 GPU sampling에 적합한 형태로 재정렬한다. 원본 데이터는 일반적으로 node index, time step, component 순서로 저장되지만, Vertex Tensor buffer에는 time step을 texture depth로 두고 각 depth 안에서 node별 component를 연속적으로 배치한다. 이후 사용자가 time step을 변경하면 texture를 다시 생성하지 않고 `depthIndex` uniform만 변경하여 다른 depth의 결과를 읽는다. 시간 단계 보간이 활성화된 경우 shader에서는 현재 depth와 다음 depth를 함께 읽고 `mix` 연산을 수행하여 중간 결과를 계산한다.

shader 단계에서는 각 vertex의 `gl_VertexID`를 기준으로 Vertex Tensor texture에서 대응되는 tensor 값을 읽는다. GLSL 구현에서는 `sampler2DArray`와 `texelFetch`를 사용하고, WebGL1 호환 경로에서는 texture coordinate를 계산하여 sampling하는 방식을 제공한다. WGSL 구현도 동일한 논리를 유지하며 `texture_2d_array`와 `textureLoad`를 사용한다. 이를 통해 현재 WebGL 기반 렌더링을 안정적으로 지원하면서도, 향후 WebGPU 렌더링 경로로 확장할 수 있도록 동일한 shader logic을 GLSL과 WGSL 양쪽에 병행 구현하였다.

Vertex Tensor에서 읽은 값은 두 가지 방식으로 사용된다. 첫 번째는 Contour Plot이다. `VertexTensorColorOutput`은 사용자가 선택한 output channel에 따라 개별 성분 또는 3성분 magnitude를 scalar 값으로 변환하고, LUT range를 계산한 뒤 LUT texture를 생성한다. Vertex shader는 선택된 tensor 값을 scalar로 변환하여 fragment shader로 전달하고, fragment shader는 이 값을 LUT texture에 매핑하여 최종 색상을 계산한다. 이 방식은 해석 결과의 색상 분포를 geometry 재생성 없이 shader에서 처리할 수 있게 한다.

두 번째는 Displacement 출력이다. `VertexTensorDisplacementOutput`은 displacement scale과 output channel을 shader에 전달하며, vertex shader는 Vertex Tensor에서 읽은 3성분 벡터 값을 현재 vertex position에 더한다. 6성분 tensor의 경우에도 첫 3성분 또는 마지막 3성분을 선택하여 변위 벡터로 사용할 수 있다. 따라서 사용자는 동일한 Vertex Tensor 데이터를 기반으로 해석 결과를 색상으로 확인하거나, 실제 변형 형상으로 확인할 수 있다.

결과적으로 Vertex Tensor는 WebGL 환경의 attribute 개수 제한과 CPU 기반 mesh 재생성 비용을 피하면서, 대용량 CAE 결과 데이터를 GPU 중심으로 처리하기 위한 구조이다. MIE에서는 이 구조를 통해 Contour Plot과 Displacement를 하나의 공통 데이터 표현 위에서 구현하였고, 시간 단계 변경, output channel 변경, LUT 범위 조절, displacement scale 조절과 같은 사용자 조작을 실시간 렌더링 파이프라인에 연결할 수 있었다.