

GeOnDT for JS 데이터 규격 및 지원 정보 가이드



문서 기준일: 2026-07-29
적용 제품: GeOnDT for JS
기준 버전: 1.0.0.0.183.a

목차

GeOnDT for JS 데이터 규격 및 지원 정보 가이드	1
목차	2
공통 안내	8
1. 문서 개요	8
2. 지원 포맷 분류	8
3. 조건 등급과 용어	8
4. 공통 데이터 작성 원칙	9
5. 데이터 제출 전 점검표	9
6. 포맷별 비교표	10
3D Tiles	12
1. 포맷 정의와 공식 규격	12
2. GeOnDT for JS 지원 여부 및 지원 버전	12
3. 입력 방식	13
4. 코드 예제	14
5. 출력 예시	15
6. 권장 사항	18
glTF 및 GLB	19
1. 포맷 정의와 공식 규격	19
2. GeOnDT for JS 지원 여부 및 지원 버전	19
3. 입력 방식	19
4. 코드 예제	20
5. 출력 예시	21
6. 권장 사항	22
FBX	23
1. 포맷 정의와 공식 규격	23
2. GeOnDT for JS 지원 여부 및 지원 버전	23
3. 입력 방식	23

4. 코드 예제	24
5. 출력 예시	25
6. 권장 사항	26
OBJ	27
1. 포맷 정의와 공식 규격	27
2. GeOnDT for JS 지원 여부 및 지원 버전	27
3. 입력 방식	28
4. 코드 예제	29
5. 출력 예시	30
6. 권장 사항	32
3DS	33
1. 포맷 정의와 공식 규격	33
2. GeOnDT for JS 지원 여부 및 지원 버전	33
3. 입력 방식	33
4. 코드 예제	37
5. 출력 예시	40
6. 권장 사항	40
WFS	42
1. 포맷 정의와 공식 규격	42
2. GeOnDT for JS 지원 여부 및 지원 버전	42
3. 입력 방식	42
4. 코드 예제	44
5. 출력 예시	45
6. 권장 사항	47
WMS	48
1. 포맷 정의와 공식 규격	48
2. GeOnDT for JS 지원 여부 및 지원 버전	48
3. 입력 방식	49
4. 코드 예제	51

5. 출력 예시	52
6. 권장 사항	53
TMS	54
1. 포맷 정의와 공식 규격	54
2. GeOnDT for JS 지원 여부 및 지원 버전	54
3. 입력 방식	55
4. 코드 예제	56
5. 출력 예시	58
6. 권장 사항	58
XYZ	60
1. 포맷 정의와 공식 규격	60
2. GeOnDT for JS 지원 여부 및 지원 버전	60
3. 입력 방식	60
4. 코드 예제	61
5. 출력 예시	62
6. 권장 사항	63
SHP	64
1. 포맷 정의와 공식 규격	64
2. GeOnDT for JS 지원 여부 및 지원 버전	64
3. 입력 방식	65
4. 코드 예제	66
5. 출력 예시	68
6. 권장 사항	71
DXF	72
1. 포맷 정의와 공식 규격	72
2. GeOnDT for JS 지원 여부 및 지원 버전	72
3. 입력 방식	72
4. 코드 예제	73
5. 출력 예시	74

6. 권장 사항	76
KML	77
1. 포맷 정의와 공식 규격	77
2. GeOnDT for JS 지원 여부 및 지원 버전	77
3. 입력 방식	78
4. 코드 예제	79
5. 출력 예시	80
6. 권장 사항	80
GeoJSON	81
1. 포맷 정의와 공식 규격	81
2. GeOnDT for JS 지원 여부 및 지원 버전	81
3. 입력 방식	82
4. 코드 예제	82
5. 출력 예시	83
6. 권장 사항	83
TopoJSON	84
1. 포맷 정의와 공식 규격	84
2. GeOnDT for JS 지원 여부 및 지원 버전	84
3. 입력 방식	84
4. 코드 예제	85
5. 출력 예시	87
6. 권장 사항	88
WKT	89
1. 포맷 정의와 공식 규격	89
2. GeOnDT for JS 지원 여부 및 지원 버전	89
3. 입력 방식	90
4. 코드 예제	90
5. 출력 예시	91
6. 권장 사항	92

WKB	93
1. 포맷 정의와 공식 규격	93
2. GeOnDT for JS 지원 여부 및 지원 버전	93
3. 입력 방식	93
4. 코드 예제	94
5. 출력 예시	94
6. 권장 사항	94
U3F	95
1. 포맷 정의와 공식 규격	95
2. GeOnDT for JS 지원 여부 및 지원 버전	95
3. 입력 방식	96
4. 코드 예제	97
5. 출력 예시	98
6. 권장 사항	99
I3F	100
1. 포맷 정의와 공식 규격	100
2. GeOnDT for JS 지원 여부 및 지원 버전	100
3. 입력 방식	101
4. 코드 예제	102
5. 출력 예시	103
6. 권장 사항	103
UMF	105
1. 포맷 정의와 공식 규격	105
2. GeOnDT for JS 지원 여부 및 지원 버전	105
3. 입력 방식	105
4. 코드 예제	107
5. 출력 예시	108
6. 권장 사항	110
UMesh	111

1. 포맷 정의와 공식 규격	111
2. GeOnDT for JS 지원 여부 및 지원 버전	111
3. 입력 방식	111
4. 코드 예제	112
5. 출력 예시	112
6. 권장 사항	113

공통 안내

1. 문서 개요

1.1 목적

이 문서는 GeOnDT for JS용 데이터를 제작하거나 제출할 때 필요한 포맷별 지원 범위, 입력 구조, 필수 요소, 좌표·높이 기준, 리소스 구성과 최적화 방법을 안내합니다.

1.2 내용

1. 지원 범위는 실제 GeOnDT for JS 소스와 공개 예제를 기준으로 안내합니다.
2. 서버 제품, Java·Unity 제품과 별도 변환 도구의 기능은 JS 직접 지원 범위에 포함하지 않습니다.
3. 외부 표준은 포맷 자체를 설명하는 참고 자료로 사용합니다.

2. 지원 포맷 분류

그룹	대상
2D·벡터·도면	SHP, DXF, WKT, WKB, GeoJSON, TopoJSON, KML
3D 모델	FBX, 3DS, OBJ, glTF, GLB
대용량·자체 규격 3D/지형	3D Tiles, UMesh, U3F, UMF, I3F
웹 서비스	WFS, WMS, TMS, XYZ

3. 조건 등급과 용어

등급	의미
[필수 조건]	누락하거나 위반하면 파싱·로딩 실패, 객체 미표시, 좌표 오류, 심각한 형상 왜곡 또는 필수 참조 오류가 발생하는 조건입니다.
[권장 조건]	필수는 아니지만 안정적인 표시, 좌표 정밀도, 컬링, 피킹, 로딩 시간, 메모리 사용량 및 렌더링 성능 개선을 위해 적용을 권장하는 조건입니다.
[선택 조건]	텍스처, 재질, 애니메이션, 스타일, 추가 속성 또는 성능 개선 기능을 사용할 때 선택적으로 적용하는 조건입니다.

- **직접 지원:** 공개 레이어나 API에 데이터를 입력하여 바로 표시할 수 있습니다.
- **일부 지원:** 포맷 또는 서비스의 일부 타입과 기능만 사용할 수 있습니다. 포맷별 지원 범위를 먼저 확인합니다.
- **변환 후 지원:** 원본을 직접 입력할 수 없으므로 OBJ, GLB 또는 타일 등 지원 포맷으로 변환해야 합니다.
- **자체 규격 포맷:** GeOnDT에서 정의한 전용 포맷입니다. 외부 표준 포맷과 호환되는 것으로 보지 않으며, 포맷별 문서에 안내된 생성 도구, 지원 버전과 파일 구성을 따라야 합니다.

4. 공통 데이터 작성 원칙

1. **[필수 조건]** 데이터 CRS와 API에 전달하는 sourceCRS, crs, srsName 또는 타일 matrix CRS를 일치시킵니다.
2. **[필수 조건]** 모든 좌표·높이·인덱스는 유한한 숫자이고 배열 범위가 유효해야 합니다.
3. **[권장 조건]** GeOnDT 입력 경로에서 Z-up 기준을 사용하는 3D 모델은 상향축을 Z축으로 맞춥니다. 원본 포맷이 Y-up 등 다른 축 체계를 사용하면 로더 또는 모델 옵션의 축 변환 여부를 확인하고 변환 내용을 함께 기록합니다.
4. **[권장 조건]** 3D 모델 정점은 모델 원점 기준의 작은 로컬 좌표 범위로 구성하고 실제 지도 위치·회전·크기는 레이어 또는 모델 배치 옵션으로 분리합니다.
5. **[권장 조건]** 모델의 origin/pivot은 형상 중심 또는 바닥 중심 부근에 배치합니다. 특히 중앙 회전·크기 조절이 필요한 모델은 중심점이 형상의 중앙에 있는지 확인하고, 중심점이 멀리 떨어져 있으면 모델 데이터를 정리합니다.
6. **[필수 조건]** 외부 buffer-texture-MTL-metadata-tile content의 URL과 대소문자, 상대 경로를 배포 서버와 일치시킵니다.
7. **[권장 조건]** 사용자 속성은 필요한 필드만 남깁니다. GeOnDT가 고정 필드를 요구하지 않는 포맷에 임의의 필수 필드명을 만들지 않습니다.

5. 데이터 제출 전 점검표

1 공통

- [] GeOnDT for JS에서 실제 지원되는 입력 경로와 포맷인가
- [] 지원 버전 또는 확인된 파서 범위를 벗어나지 않는가
- [] 필수 지오메트리·좌표·index·참조가 포함되어 있는가
- [] CRS, 축 순서, Z 단위와 높이 기준을 기록했는가
- [] URL, 파일명 대소문자, 상대 경로와 CORS가 올바른가
- [] NaN·Infinity·잘못된 index·빈 geometry·되화 면을 제거했는가
- [] 실제 GeOnDT 예제/API에서 성공과 오류 경로를 확인했는가

2 3D 모델·타일

- [] 해당 포맷과 GeOnDT 로더의 상향축 기준을 확인했는가
- [] Z-up이 적용되는 입력 경로라면 모델 상향축이 Z축으로 구성되었는가
- [] Y-up 등 다른 축 체계의 포맷은 필요한 축 변환 또는 회전값을 확인했는가
- [] 모델 정점이 원점 기준의 작은 로컬 좌표이고 실제 지도 위치를 별도 옵션으로 지정했는가
- [] origin/pivot이 형상 중심 또는 바닥 중심 부근에 있는가
- [] 중앙 회전·크기 조절이 필요한 모델은 중심점이 형상의 중앙에 있는가

- [] bbox/accessor min-max/boundingBox이 실제 형상과 일치하는가
- [] 불필요한 node, mesh, primitive, material, animation을 제거했는가
- [] 텍스처 크기·개수와 draw call을 줄였는가
- [] 외부 texture/buffer/metadata/tile URI가 모두 접근 가능한가

3 벡터·서비스·래스터

- [] 폴리곤 외부 링·hole, arc/index, geometry type이 해당 레이어 지원 범위인가
- [] SHP의 도형·속성·좌표계 파일 세트가 같은 데이터인가
- [] WFS/WMS/WMTS의 버전·레이어명·CRS·응답 포맷이 서버와 일치하는가
- [] XYZ/TMS의 Y 방향, 레벨, tile size와 URL placeholder가 일치하는가

6. 포맷별 비교표

포맷	유형	지원 수준	권장 포맷	사전 변환	확인된 버전	Z 지원	로컬 좌표 권장	핵심 필수 요소	사용 API/파서
3D Tiles	3D 타일	일부 지원	tileset.json과 상대 URI로 연결된 지원 콘텐츠	경우에 따라 필요	1.0 기본 구조, 1.1 직접 GLB 일부 지원	콘텐츠별	타일 transform 사용	asset, geometricError, root, boundingVolume, content.uri	U3dModelTilesLayer
glTF 및 GLB	3D 모델	일부 지원	GLB	glTF는 변환 필요	glTF 2.0 계열, GLB만 지원	가능	권장	asset.version, POSITION, 유효한 buffer	U3dModelBasicLayer
FBX	3D 모델	직접 지원	FBX와 외부 텍스처	경우에 따라 필요	FBXVersion 6400 이상 권장	가능	권장	유효한 FBX 메시와 텍스처 경로	U3dModelBasicLayer
OBJ	3D 모델	직접 지원	동일 기본명의 OBJ/MTL과 외부 텍스처 묶음	불필요	별도 버전 없음, 주요 요소 부분 지원	가능	권장	v, f, 동일 기본명의 MTL, 텍스처 경로	U3dModelBasicLayer
3DS	3D 모델	직접 지원	3DS, 좌표정보와 필요 시 외부 텍스처	불필요	별도 포맷 버전 제한 없음, 주요 체크 부분 지원	가능	권장	유효한 메시, 좌표정보와 외부 리소스 경로	createTdsModelLayer
WFS	벡터 서비스	일부 지원	JSON 응답을 제공하는 WFS GetFeature 서비스	경우에 따라 필요	WFS 1.1.0 또는 1.0.0	fieldheight 속성으로 높이 표현	해당 없음, CRS 일치	서비스 URL, layername, JSON 응답, srsName	U2dVectorShaderLayer, U3dModelWFSLayer
WMS	이미지 서비스	직접 지원	EPSG:3857 GetMap 응답을 제공하는 WMS 서비스	불필요	WMS 1.3.0 또는 1.1.0 GetMap	해당 없음	해당 없음, EPSG:3857 사용	baseUrl, layerName, version, CRS/SRS, 이미지 형식	createWMSImageLayer, U3dImageWMSLayer
TMS	타일 서비스	직접 지원	TMS 타일셋 루트 URL과 GeoNT용 메타데이터	불필요	TMS 1.0.0 주소 체계 일부 지원	해당 없음	해당 없음, 타일 좌표계 일치	baseUrl, metadata, 타일 범위와 Y 방향	createTMSImageLayer
XYZ	타일 서비스	직접 지원	XYZ URL 템플릿	불필요	Web Mercator 기반 z/x/y URL 템플릿	해당 없음	해당 없음, Web Mercator 사용	URL의 (z), (x), (y)와 서버 타일 체계	U3dImageXYZLayer
SHP	파일/벡터	일부 지원	SHP + SHX + DBF (+ PRJ) 세트, UTF-8 DBF	경우에 따라 필요	명시적 버전 없음, ESRI 규격 (1998) 기준	Z/M 타입 미지원	해당 없음, CRS 일치	유효한 SHP 헤더와 지원 Shape Type	U3dShpModelLayer
DXF	파일/도면	직접 지원	그룹 코드와 값의 쌍으로 구성된 ASCII DXF	Binary DXF 또는 미지원 엔티티 포함 시 필요	특정 파일 버전 제한 없음	엔티티별 지원	해당 없음, CRS 일치	ASCII 그룹 코드, 지원 엔티티와 좌표	UDxtparser, U2dDxfLayer, U3dModelDxfLayer
KML	XML 벡터	일부 지원	UTF-8 KML	경우에 따라 필요	KML 2.2 또는 2.3	좌표 고도 입력 가능, 고도 모드 미지원	해당 없음, WGS84 사용	KML XML, 지원 지오메트리, coordinates	U3dKmlModelLayer
GeoJSON	파일·객체/벡터	일부 지원	UTF-8 GeoJSON 객체	불필요	RFC 7946 기준	세 번째 좌표 입력 가능	해당 없음, CRS 일치	지원 Geometry Type, sourceCRS	addGeometryAsGeojson
TopoJSON	파일·객체/위상 벡터	일부 지원	UTF-8 TopoJSON 객체	경우에 따라 필요	TopoJSON 객체 입력	복원된 좌표 기준	해당 없음, CRS 일치	Topology, objects, arcs, sourceCRS	addGeometryAsTopojson

포맷	유형	지원 수준	권장 포맷	사전 변환	확인된 버전	Z 지원	로컬 좌표 권장	핵심 필수 요소	사용 API/파서
WKT	텍스트/지오메트리	일부 지원	지원 WKT 타입	불필요	OGC Simple Features Access 기반, 지원 Geometry Type 제한	파싱 가능, 표시 경로별 확인	해당 없음, CRS 일치	지원 Geometry Type, sourceCRS	U2dVectorShaderLayer.addGeometryAsWKT
WKB	바이너리/지오메트리	공개 API 없음	GeoJSON 또는 WKT로 변환된 지오메트리	필요	OGC WKB, 직접 지원 범위 없음	변환 결과별	해당 없음, CRS 별도 확인	WKT 또는 GeoJSON 변환 결과	공개 API 없음
U3F	자체 규격 3D 타일	직접 지원	GeOnDT Builder가 생성한 U3F 2.2~2.5 타일셋 전체	원본 포맷은 Builder 변환 필요	콘텐츠 2.2, 2.3, 2.4, 2.5	가능	타일 좌표 사용	루트 U3G, 레벨 범위, U3M/패키지 정보, U3F 콘텐츠	create3DFModelLayer, U3dModelU3FLayer
I3F	자체 규격 3D 타일	직접 지원	GeOnDT Builder가 생성한 I3F 타일셋 전체	원본 포맷은 Builder 변환 필요	tilemap.json version 1.0.0.0	가능	EPSG:3857 타일 좌표 사용	tilemap.json, I3F/I3Z 콘텐츠와 텍스처	U3dModelI3FLayer
UMF	자체 규격 고도 타일	직접 지원	GeOnDT Builder가 생성한 UMF 타일셋 전체	IMG-GeoTIFF 원본은 Builder 변환 필요	TileMap 1.0.0, Int16 또는 Float32 고도 배열	고도 필수	EPSG:3857 타일 좌표 사용	tilemapsource.xml, 레벨 /X/Y UMF 타일	createHeightXYZLayer, U3dHeightXYZLayer
UMesh	자체 규격 3D 모델	직접 지원	UMesh와 외부 리소스	경우에 따라 필요	UMesh 1.0 또는 1.1	가능	권장	UMeshWriter magicKey, version, 메시 블록	U3dModelBasicLayer

공식 외부 사양

아래 링크는 각 포맷 자체의 외부 규격 또는 기술 참고 자료입니다. 외부 규격에서 제공하는 모든 기능을 GeOnDT for JS가 지원한다는 의미는 아니므로 실제 지원 범위는 각 포맷 문서를 확인합니다.

포맷	외부 공식 사양 또는 참고 자료
3D Tiles	OGC 3D Tiles
glTF 및 GLB	Khronos glTF Specification
FBX	Autodesk FBX SDK Documentation , FBX SDK 안내
OBJ	Library of Congress - Wavefront OBJ File Format
3DS	공개적으로 유지되는 단일 공식 파일 사양을 확인하기 어렵습니다.
WFS	OGC Web Feature Service
WMS	OGC Web Map Service
TMS	OSGeo Tile Map Service Specification — OGC 공식 표준이 아닌 참고 사양입니다.
XYZ	OpenStreetMap Slippy map tilenames — 단일 공식 표준이 아닌 기술 참고 자료입니다.
SHP	ESRI Shapefile Technical Description
DXF	AutoCAD DXF Reference
KML	OGC KML 표준 , KML 2.3 사양
GeoJSON	GeoJSON RFC 7946
TopoJSON	TopoJSON Specification
WKT	OGC Simple Feature Access
WKB	OGC Simple Feature Access
U3F	GeOnDT 자체 규격이며 외부 공개 사양은 없습니다.
I3F	GeOnDT 자체 규격이며 외부 공개 사양은 없습니다.
UMF	GeOnDT 자체 규격이며 외부 공개 사양은 없습니다.
UMesh	GeOnDT 자체 규격이며 외부 공개 사양은 없습니다.

3D Tiles

1. 포맷 정의와 공식 규격

3D Tiles는 대규모 3D 지리공간 데이터를 타일 계층으로 나누어 전송하고 표시하기 위한 OGC 표준입니다.

최상위 tileset.json에는 타일 계층, 공간 범위, 화면 오차와 콘텐츠 경로가 저장됩니다. 실제 형상은 B3DM-I3DM-PNTS-CMPT-GLB 등의 파일에 저장되며, 화면에 필요한 타일을 카메라 위치와 상세도에 따라 단계적으로 불러옵니다.

외부 공식 규격

항목	내용
규격명	3D Tiles Community Standard
발행 기관	Open Geospatial Consortium
조치일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 파일·서비스/3D 타일

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

OGC 3D Tiles에는 1.0과 1.1 규격이 있습니다. GeOnDT for JS는 1.0 방식의 기본 타일셋 구조와 주요 콘텐츠를 중심으로 지원하며, 파일의 asset.version 값만으로 지원 여부를 결정하지 않습니다.

항목	지원 여부	설명
기본 타일셋 구조	지원	asset, geometricError, root, children, transform, refine, content.uri를 사용합니다. 이전 표기인 content.url도 읽습니다.
공간 범위	일부 지원	boundingVolume.region, box, sphere를 읽습니다. sphere와 transform을 함께 사용하는 경우에는 데이터의 출력 확인이 필요합니다.
B3DM	일부 지원	B3DM에 포함된 glTF/GLB와 JSON 형식의 객체 위치·속성 정보(feature table-batch table)를 읽습니다. 같은 정보의 바이너리 형식은 지원하지 않습니다.
I3DM	일부 지원	기본 위치·방향·크기와 포함되거나 외부에서 참조된 GLB를 지원합니다. 압축된 위치 좌표(양자화 위치) 등 일부 인스턴스 속성은 지원하지 않습니다.
PNTS	일부 지원	위치·색상·압축된 위치 좌표(양자화 위치)와 Draco 압축 점을 지원합니다. 법선과 일부 객체별 분류 속성(batch 속성)은 지원하지 않습니다.
CMPT	일부 지원	CMPT 안의 B3DM-I3DM만 지원합니다. PNTS-GLB-중첩 CMPT는 지원하지 않습니다.
직접 GLB 콘텐츠	지원	타일의 content.uri가 가리키는 GLB를 읽습니다.
직접 glTF JSON 콘텐츠	미지원	.gltf 파일을 타일 콘텐츠로 직접 입력할 수 없습니다.
3D Tiles 1.1 추가 기능	미지원	암시적 타일링(implicit tiling), 다중 콘텐츠(contents), subtree, 구조 메타데이터(structural metadata)는 지원하지 않습니다.

항목	지원 여부	설명
3D Tiles 확장 기능 데이터	미지원	tileset, bounding volume, batch table의 extensions 객체를 지원하지 않습니다.
3D Tiles Style	미지원	3D Tiles 스타일 규칙을 직접 적용할 수 없습니다.

이 제한은 실제 extensions 객체에 적용됩니다. extensionsUsed-extensionsRequired 목록만 있는 경우와 PNTS의 Draco 압축처럼 위 표에서 별도로 지원한다고 명시한 기능은 구분합니다.

3. 입력 방식

입력 자료를 다운로드 받을 수 있는 URL을 관련 레이어에 입력합니다.

관련 레이어:

- GeOnDT.model.U3dModelTilesLayer

파일 구성과 외부 리소스

tileset.json에서 참조하는 하위 타일셋과 콘텐츠를 상대 경로 구조 그대로 배포합니다.

```

1  /data/3d-tiles/
2  |— tileset.json
3  |— tiles/
4  |   |— root.b3dm
5  |   |— instances.i3dm
6  |   |— points.pnts
7  |   |— child-tileset.json
8  |— models/
9     |— building.glb

```

content.uri에는 tiles/root.b3dm처럼 tileset.json 기준 상대 경로를 사용합니다. 경로 앞에 /를 붙이지 않고, 콘텐츠 확장자는 소문자로 작성합니다.

http:// 또는 https://로 시작하는 절대 URI와 /로 시작하는 URI는 사용하지 않습니다. GeOnDT가 이 URI를 tileset.json의 기준 경로와 다시 결합하므로 잘못된 URL이 될 수 있습니다.

모든 JSON-타일 URL의 대소문자와 상대 경로를 실제 서버 구성과 일치시키고 CORS를 허용합니다. HTTP Range 요청은 필수 조건이 아닙니다.

오류 및 누락 시 결과

- tileset.json 요청이 실패하거나 JSON 문법이 올바르지 않으면 레이어 초기화가 실패할 수 있습니다.
- asset, root, geometricError, 유효한 boundingVolume이 없거나 콘텐츠 형식이 올바르지 않으면 빈 화면이 표시되거나 로딩이 끝나지 않을 수 있습니다.
- layer.ready의 오류 처리는 최상위 파일 요청 오류와 일부 초기화 오류를 확인하지만, 모든 구조 오류와 이후의 하위 콘텐츠 오류를 전달하지는 않습니다. 화면이 비어 있거나 일부 영역만 보이지 않으면 브라우저의 네트워크 요청과 오류 메시지도 확인합니다.
- 잘못된 transform, boundingVolume, geometricError는 모델을 다른 위치에 표시하거나, 타일을 너무 일찍 숨기거나, 불필요한 타일을 계속 요청하게 할 수 있습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerName = "3D_Tiles";
3
4 const layer = new GeOnDT.model.U3dModelTilesLayer({
5   name: layerName,
6
7   // 개별 B3DM 파일이 아니라 최상위 tileset.json URL을 입력합니다.
8   baseurl: "/data/3d-tiles/tileset.json",
9   useproxy: false,
10
11   // 기본값 20을 시작점으로 사용하고 실제 서비스 화면에서 조정합니다.
12   maximumScreenSpaceError: 20,
13   usebox: false
14 });
15
16 app.addLayer(layer);
17
18 layer.ready.then(
19   () => {
20     // ready는 최상위 타일셋의 위치와 범위가 준비된 시점입니다.
21     // 화면에 필요한 하위 타일은 카메라 위치에 따라 이후에도 계속 불러옵니다.
22     app.showLayer(layerName, true);
23
24     // 카메라 이동 오류는 타일셋 초기화 오류와 구분합니다.
25     app.fitLayerExtent(layerName)
26       .catch((error) => {
27         console.error("3D Tiles 카메라 이동 실패", error);
28       });
29   },
30   (error) => {
31     console.error("3D Tiles 초기화 실패", error);
32     app.removeLayer(layerName);
33   }
34 );
35
36 // 숨기기: app.showLayer(layerName, false);
37 // 제거하기: app.removeLayer(layerName);
```


5. 출력 예시

3D Tiles는 카메라에 가까운 영역을 더 세밀하게, 먼 영역을 낮은 상세도로 표시합니다. 카메라가 이동하면 필요한 하위 타일을 추가로 불러오거나 더 이상 필요하지 않은 타일을 정리합니다.



B3dm Tiles 출력



Pnts Tiles 출력



고정밀 Tiles 출력

6. 권장 사항

필수 조건

- 최상위 JSON에 asset, geometricError, root를 포함하고 각 타일의 boundingVolume과 geometricError를 유효한 값으로 작성합니다.
- boundingVolume은 region, box, sphere 중 하나로 작성하고 실제 좌표와 transform을 일치시킵니다.
- 콘텐츠가 있는 타일의 content.uri에는 지원되는 소문자 확장자와 tileset.json 기준 상대 경로를 사용합니다.
- 절대 URI와 /로 시작하는 URI는 사용하지 않습니다.
- 모든 JSON-타일 파일을 접근 가능한 URL로 배포하고 파일명 대소문자, 상대 경로와 CORS를 서버 구성과 일치시킵니다.
- tileset, bounding volume, batch table에는 extensions 객체를 포함하지 않습니다.

권장 조건

- boundingVolume이 실제 콘텐츠 전체를 포함하도록 작성합니다. 범위가 너무 작으면 객체가 일찍 사라지고, 너무 크면 불필요한 타일 요청이 늘어날 수 있습니다.
- 각 콘텐츠의 형상 중심점을 실제 형상의 중심 가까이에 둡니다. 중심점이 형상에서 멀리 떨어지면 큰 좌표값으로 인해 위치 정밀도가 낮아질 수 있습니다.
- 형상 정점은 형상 중심점을 기준으로 한 로컬 좌표로 작성하고, 지도상의 실제 위치는 타일의 transform 등 배치 정보로 분리합니다.
- 한 타일에 형상·텍스처·메타데이터가 과도하게 몰리지 않도록 나눕니다.
- 빈 타일, 사용하지 않는 메타데이터와 중복 콘텐츠를 제거하고 서버 캐시와 전송 압축을 적용합니다.
- 텍스처가 필요한 콘텐츠는 외부 파일 대신 텍스처를 포함한 GLB 형식을 권장합니다.
- maximumScreenSpaceError 기본값 20에서 시작하여 실제 장면의 표시 품질, 네트워크 요청량, 메모리 사용량을 함께 측정한 뒤 조정합니다.

선택 조건

- 같은 형상을 표시하는 3D Tiles라도 타일 트리 구성에 따라 요청 수, 메모리 사용량과 화면 갱신 성능이 크게 달라질 수 있습니다. 데이터 분포와 사용 화면에 맞는 트리 구조를 선택합니다.
- geometricError는 상위 타일에서 하위 타일로 갈수록 실제 상해도 차이에 맞게 규칙적으로 감소하도록 구성합니다. 단계별 값이 뒤바뀌거나 지나치게 급변하지 않도록 확인합니다.
- 한 부모에 자식 타일이 지나치게 많이 연결되는 넓은 트리와, 작은 영역까지 단계가 과도하게 이어지는 깊은 트리를 피합니다.
- 특정 상세 해상도가 필요한 영역은 공간을 적절히 분할하여 해당 영역의 타일만 세분화합니다. 고밀도 영역은 더 자세히 나누고 단순한 영역은 불필요한 단계를 줄여 트리의 너비와 깊이를 함께 조절합니다.

glTF 및 GLB

1. 포맷 정의와 공식 규격

glTF는 3D 장면, 메시, 재질, 텍스처, 스켈레톤과 애니메이션 정보를 웹 및 실시간 애플리케이션에서 효율적으로 전달하기 위한 3D 모델 형식입니다.

.gltf는 장면 구조를 JSON으로 저장하고 메시 바이너리와 이미지 파일을 외부 리소스로 참조할 수 있습니다.

.glb는 glTF JSON, 바이너리 버퍼와 일부 이미지 리소스를 하나의 바이너리 파일로 묶은 컨테이너 형식입니다.

외부 공식 규격

항목	내용
규격명	glTF Specification
발행 기관	Khronos Group
조회일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 파일/3D 모델

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

glTF 2.0 계열 중 asset.version의 주 버전이 2 이상인 GLB 파일만 지원합니다.

외부 버퍼와 이미지를 참조하는 .gltf JSON 파일은 직접 입력할 수 없으므로 GLB로 변환해야 합니다.

3. 입력 방식

URL 또는 Data URI로 읽어 온 .glb 데이터를 입력으로 받습니다.

GLB 단일 파일 형태를 권장합니다.

- baseUrl: GLB 파일이 있는 기본 URL입니다.
- fileName: 불러올 .glb 파일명입니다.
- needxml: 별도 XML 모델 목록을 사용하지 않을 때 false로 설정합니다.
- listmodel: GLB 경로 정보를 담은 모델 항목 배열입니다.

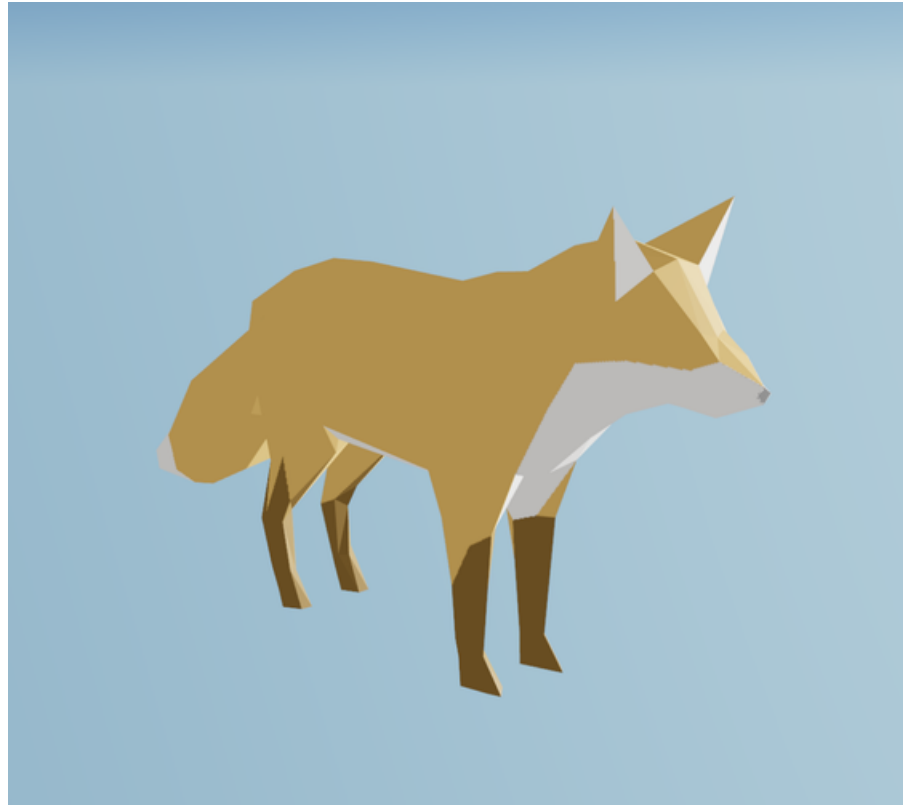
오류 및 누락 시 결과

- GLB URL이 잘못되었거나 파일을 불러오지 못하면 layer.ready가 실패하고 모델이 표시되지 않습니다.
- asset.version, 메시의 POSITION, bufferView 또는 accessor가 누락되거나 올바르지 않으면 파싱이 실패하거나 형상이 표시되지 않을 수 있습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const model = { // GLB 모델 데이터의 경로 정보입니다.
3   name: 'sample-glb',
4   baseUrl: '/data/models/',
5   fileName: 'sample.glb'
6 };
7
8 const layer = new GeOnDT.model.U3dModelBasicLayer({
9   name: 'GLB_Model',
10  needxml: false, // 별도 XML 모델 정보가 없는 경우 비활성화합니다. (기본값 true)
11  listmodel: [model]
12 });
13
14 app.addLayer(layer);
15
16 layer.ready.then() => {
17   app.showLayer(layer.getName(), true);
18   return app.fitLayerExtent(layer.getName());
19 }.catch((error) => {
20   console.error('GLB 로딩 실패', error);
21   app.removeLayer(layer.getName());
22 });
23
24 // 제거: app.removeLayer(layer.getName());
```

5. 출력 예시



GeOnDT for JS에서 출력된 GLB 결과 화면

6. 권장 사항

필수 조건

- 입력 파일은 .glb 형식이어야 하며 asset.version의 주 버전은 2 이상이어야 합니다.
- 메시의 POSITION, bufferView와 accessor 참조가 실제 GLB 데이터 범위와 일치해야 합니다.

권장 조건

- glTF 기본 Y-up, 오른손 좌표계, 선형 단위 미터. 로컬 정점을 권장하고 지도 배치는 layer location/rotation/scale로 분리합니다.
- 모델 원점을 형상 중심 또는 바닥 중심 부근에 배치하고 사용하지 않는 scene-node-mesh-material-animation을 제거합니다.
- primitive와 재질 수를 줄여 draw call을 낮추고 텍스처를 실제 화면 해상도에 맞게 최적화합니다.
- Draco, KTX2와 Meshopt는 GeOnDT 로더 설정 및 디코더 리소스가 준비된 경우에만 사용합니다.

FBX

1. 포맷 정의와 공식 규격

FBX(Filmbox)는 Autodesk에서 제공하는 3D 장면 및 자산 교환 포맷입니다.

메시, 재질, 텍스처 참조, 카메라, 조명, 스켈레톤, 애니메이션 등 3D 장면을 구성하는 데이터를 저장할 수 있으며, 바이너리 또는 ASCII 형식으로 작성될 수 있습니다.

외부 공식 규격

항목	내용
규격명	Autodesk FBX SDK Documentation
발행 기관	Autodesk
조회일	2026-07-31
공식 문서	공식 규격 보기

- 기술 설명: [What is Autodesk FBX Technology](#)
- 유형: 파일/3D 모델

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

버전은 FBXVersion 6400 이상을 권장합니다.

3. 입력 방식

URL 또는 Data URI로 읽어 온 .fbx 데이터를 입력으로 받습니다.

FBX 데이터는 바이너리 FBX 또는 ASCII 텍스트 FBX일 수 있습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const model = {
3   name: 'sample-fbx',
4   baseUrl: 'http://test-sever/data/models/',
5   fileName: 'sample.fbx',
6   ext: 'fbx'
7 };
8
9 const layer = new GeOnDT.model.U3dModelBasicLayer({
10   name: 'FBX_Model',
11   needxml: false,
12   listmodel: [model]
13 });
14
15 app.addLayer(layer);
16 app.showLayer(layer.getName(), true);
17
18 // 제거: app.removeLayer(layer.getName());
```


5. 출력 예시



GeOnDT for JS에서 출력된 FBX 결과 화면

6. 권장 사항

필수 조건

- 모델 정점 좌표는 EPSG:4326(WGS84 경위도) 또는 EPSG:3857(Web Mercator)과 같은 지리 좌표계가 아니라, 모델 원점 기준의 로컬 좌표(단위: 미터)를 사용해야 합니다.

권장 조건

- 형상의 중심점이 가운데에 위치되도록 권장합니다. 중심점이 형상에서 크게 벗어나 있거나, 바닥면 등으로 편향되면 회전, 스케일, 위치 보정, 카메라 포커싱, 객체 선택 시 기준점이 어긋나 모델이 의도와 다르게 움직이거나 배치될 수 있습니다.
- 형상 바운딩박스는 실제 형상 크기와 위치에 맞도록 합니다. 바운딩박스가 형상보다 과하게 크거나 작으면 화면 표시 범위 계산, 카메라 이동/줌, 객체 선택, 충돌 검사, 가시성 판단, LOD 처리에서 오차가 발생할 수 있습니다.
- 미사용 노드, 메시, 애니메이션, 재질, 참조되지 않는 텍스처, 중복 정점, 퇴화 면을 제거하고 material, object, draw call 수를 줄입니다.
- 웹 로딩 성능을 위해 텍스처 용량과 폴리곤 수는 서비스 환경에 맞게 조정합니다.

OBJ

1. 포맷 정의와 공식 규격

Wavefront OBJ는 3D 모델의 형상을 정점, 텍스처 좌표, 법선, 면으로 구분하여 저장하는 텍스트 형식입니다.

형상은 OBJ 파일에 저장하며 색상과 텍스처 같은 재질 정보는 MTL 파일에 저장합니다. MTL에서 참조하는 PNG-JPEG 등의 텍스처 파일도 함께 사용할 수 있습니다.

OBJ에는 좌표계, 길이 단위, 위쪽 축이 지정되어 있지 않습니다. 카메라, 조명, 애니메이션 정보도 지원 대상이 아닙니다.

외부 공식 규격

항목	내용
규격명	Wavefront OBJ File Format
문서 번호	fdd000507
발행 기관	Library of Congress
규격 상태	Full
조회일	2026-08-05
공식 문서	공식 규격 보기

- 기술 참고: [Three.js OBJLoader](#)
- 유형: 파일/3D 모델

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

OBJ는 별도의 포맷 버전 번호를 사용하지 않습니다.

항목	지원 여부	설명
정점·텍스처 좌표·법선	지원	v, vt, vn을 읽습니다. vn이 없으면 삼각형 면 법선을 계산합니다.
삼각형 면	지원	양수·음수 인덱스와 일반적인 OBJ 면 표기를 지원합니다.
삼각형 외 다각형 면	일부 지원	첫 정점을 기준으로 삼각형으로 나눕니다. 오목한 면은 제작 도구에서 미리 삼각분할합니다.
객체·그룹	지원	o, g로 구분된 형상을 읽습니다.
MTL 재질	일부 지원	기본색·반사색·발광색과 주요 텍스처 맵을 지원합니다.
자유곡선·NURBS 곡면	미지원	다각형 메시로 변환해야 합니다.
usemap	미지원	텍스처는 MTL 파일에서 지정해야 합니다.

Ka, Ni, illum은 재질에 반영되지 않습니다. 광택과 투명도는 레이어 설정에 따라 원본 MTL과 다르게 보일 수 있습니다.

3. 입력 방식

입력 자료를 다운로드 받을 수 있는 URL을 관련 레이어에 입력합니다.

관련 레이어:

- GeOnDT.model.U3dModelBasicLayer
- GeOnDT.model.U3dMultipleComponentLayer
- GeOnDT.model.U3dModelStaticLayer
- GeOnDT.model.U3dModelBIMObjLayer

파일 구성과 외부 리소스

현재 GeOnDT의 일반 OBJ 입력 방식에서는 building.obj을 불러올 때 동일한 기본명의 building.mtl이 필요합니다. OBJ 안에 다른 mtl lib 파일명이 있어도 요청 파일명은 바뀌지 않으므로 기본명을 맞춥니다.

```
1 /data/models/building/
2 |— building.obj
3 |— building.mtl
4 |— textures/
5 |   |— building_diffuse.jpg
6 |   |— building_normal.png
```

MTL의 텍스처 상대 경로와 파일명 대소문자를 유지하고, 모든 파일을 URL로 접근할 수 있게 배포합니다. ZIP 파일은 직접 읽지 않으므로 압축을 먼저 해제합니다.

오류 및 누락 시 결과

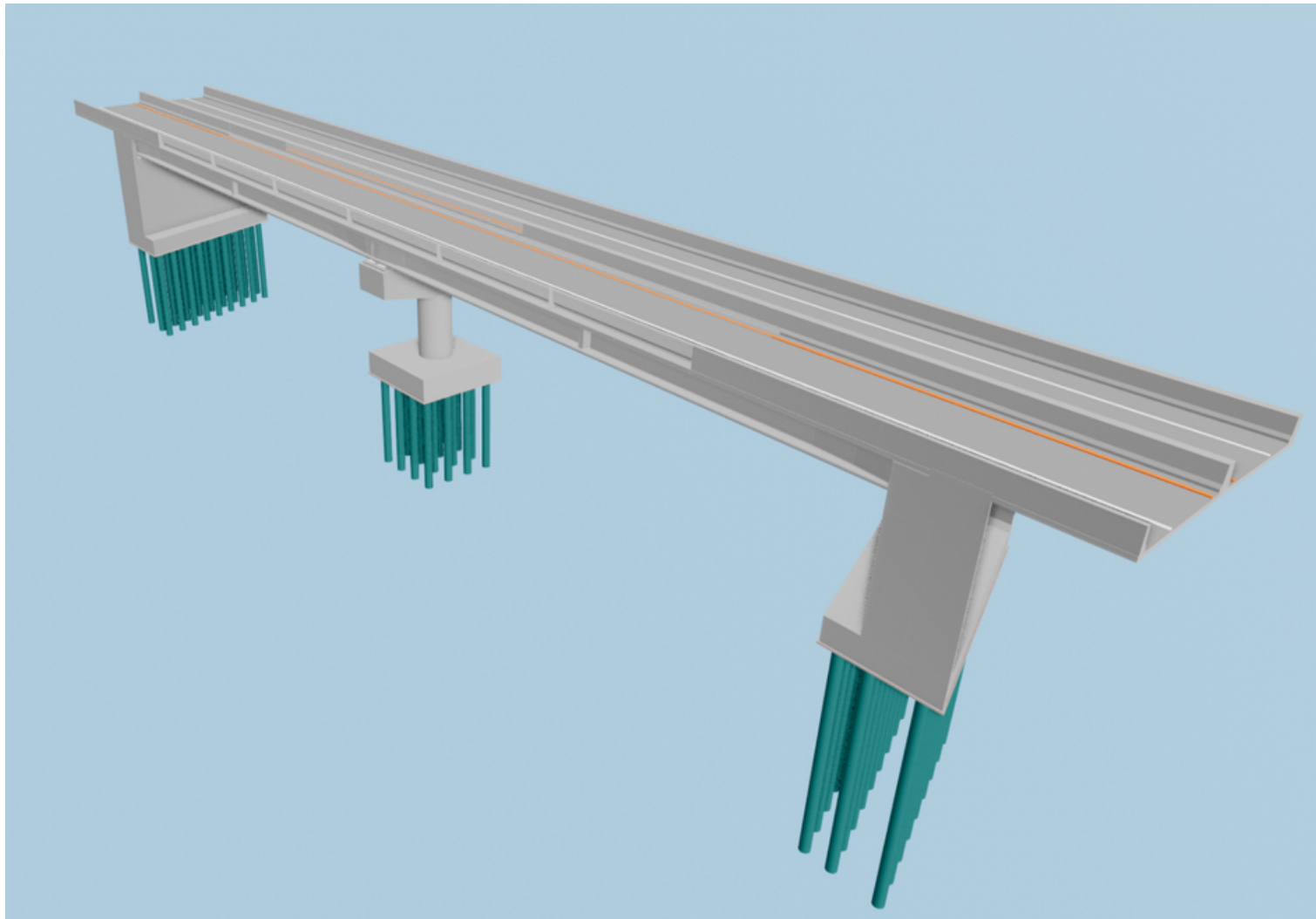
- 동일 기본명의 MTL이 없거나 OBJ-MTL URL이 잘못되면 모델이 표시되지 않습니다.
- 텍스처가 누락되면 형상은 표시되어도 색상, 표면 질감, 투명도가 의도와 다르게 보일 수 있습니다.
- 정점에 유한하지 않은 숫자가 있거나 면의 인덱스가 실제 데이터 범위를 벗어나면 형상이 깨지거나 표시되지 않을 수 있습니다.
- 서버의 파일명 대소문자, 상대 경로 또는 CORS 설정이 잘못되어도 파일을 불러올 수 없습니다.

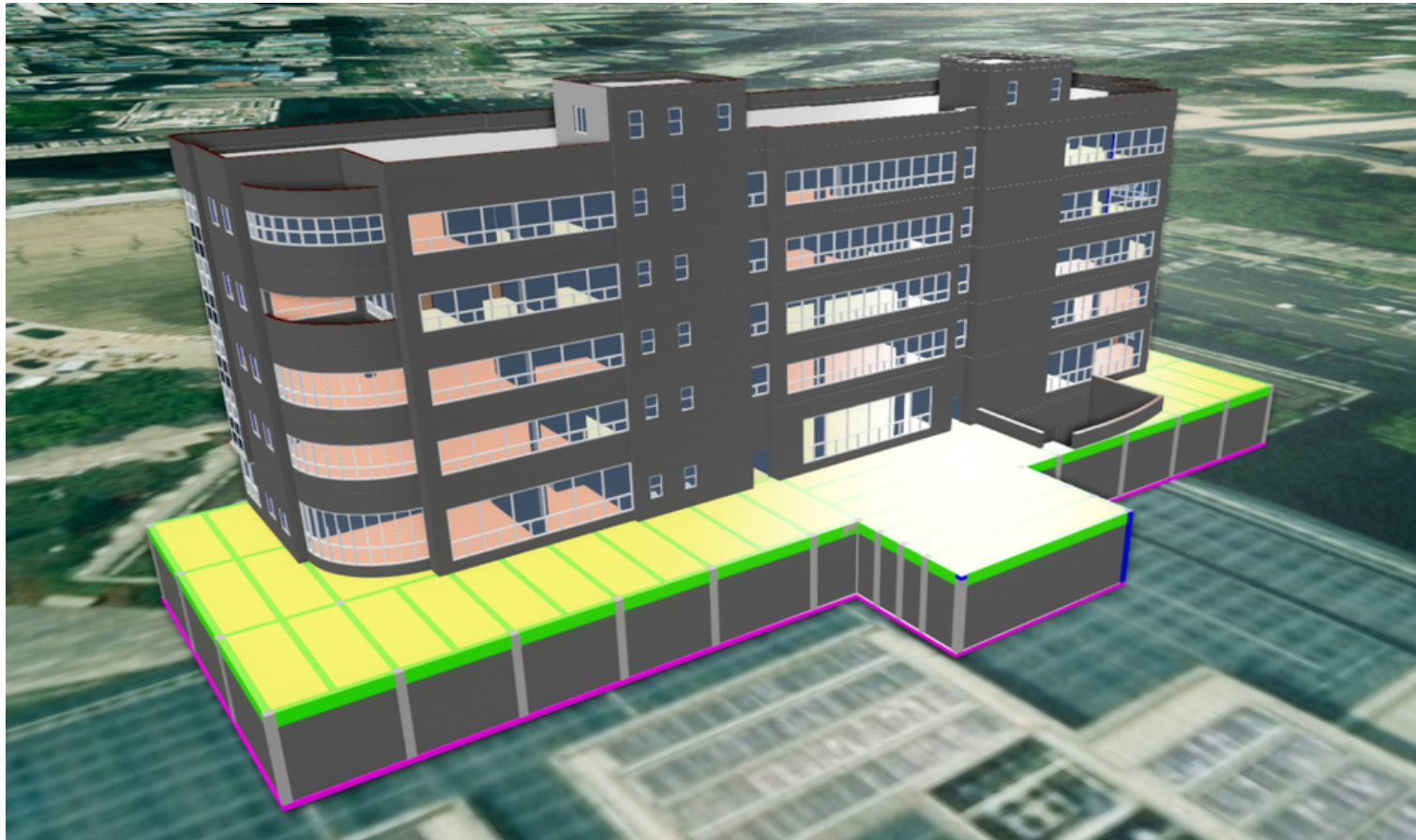
4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 // 아래 경로에는 sample.obj, sample.mtl, MTL이 참조하는 텍스처를 함께 배포합니다.
3 const model = {
4   name: "sample-obj",
5   baseUrl: "/data/models/sample/", // URL 결합을 위해 마지막 '/'를 유지합니다.
6   fileName: "sample.obj",
7   ext: ".obj"
8 };
9
10 const layer = new GeOnDT.model.U3dModelBasicLayer({
11   name: "OBJ_Model",
12   needXml: false,
13   listModel: [model],
14
15   // 로컬 좌표로 제작한 모델을 WGS84 경도·위도·높이 위치에 배치합니다.
16   geoLocation: { x: 127.0, y: 37.5, z: 0 },
17   rotation: { x: 0, y: 0, z: 0 },
18   scale: { x: 1, y: 1, z: 1 }
19 });
20
21 app.addLayer(layer);
22 app.showLayer(layer.getName(), true);
23
24 layer.ready
25   .then(() => {
26     app.fitLayerExtent(layer.getName());
27   })
28   .catch((error) => {
29     console.error("OBJ 로딩 실패", error);
30     app.removeLayer(layer.getName());
31   });
```

5. 출력 예시

OBJ의 형상은 3D 모델로 표시됩니다. MTL과 외부 텍스처가 정상적으로 연결되면 색상·재질·텍스처가 함께 적용됩니다.





6. 권장 사항

필수 조건

- OBJ와 MTL의 기본명을 동일하게 작성하고 두 파일을 모두 접근 가능한 URL로 배포합니다.
- baseurl은 /로 끝내고, MTL의 텍스처 상대 경로와 실제 파일 경로를 일치시킵니다.
- 모든 정점은 유효한 숫자로 작성하고 정점·텍스처 좌표·법선 인덱스를 실제 배열 범위에 맞춥니다.

권장 조건

- 일반 OBJ는 모델 원점 기준의 로컬 좌표로 작성하고 길이 단위와 위쪽 축을 기록합니다.
- 오목한 다각형 면은 미리 삼각분할하고 중복 정점, 퇴화 면, 보이지 않는 내부 면을 제거합니다.
- 사용하지 않는 객체·그룹·재질·텍스처를 제거하고 텍스처 해상도와 파일 용량을 서비스 환경에 맞게 조정합니다.
- 큰 파일과 많은 면·그룹·재질은 초기 표시 시간과 메모리 사용량을 늘릴 수 있으므로 실제 서비스 환경에서 확인합니다.

선택 조건

- 정점이 투영좌표이고 Y축이 높이인 OBJ는 yUp: true와 실제 srs를 함께 사용할 수 있습니다. 일반 로컬 모델의 축 보정 용도로 사용하지 않습니다.
- 대형 모델은 GLB 또는 3D Tiles로 변환한 결과와 로딩 성능을 비교할 수 있습니다.

3DS

1. 포맷 정의와 공식 규격

3DS(3D Studio)는 3D 모델의 메시, 재질, 텍스처 참조와 객체 변환 정보를 저장하는 바이너리 형식입니다.

3DS 파일의 데이터는 용도별 청크로 구분하여 저장됩니다. 다만 좌표값이 어떤 지리 좌표계를 사용하는지 나타내는 EPSG 식별 정보는 포함하지 않습니다. 따라서 모델 형상, 지도상의 실제 위치, 수평 좌표계와 높이 기준을 구분하여 관리해야 합니다.

- 공식 공개 규격: 공개적으로 유지되는 단일 공식 규격 문서 확인 필요
- 유형: 파일/3D 모델

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

3DS는 별도의 포맷 버전 제한 없이 주요 청크를 부분 지원합니다.

GeOnDT for JS 1.0.0.7 이상에서는 POS_TRACK_TAG(0xB020)에 기록된 3DS 내부 Position을 지도상의 실제 위치로 적용할 수 있습니다. 이전 버전에서는 레이어 position 또는 XML 메타정보로 위치를 입력합니다.

실제 지원 여부는 파일 버전보다 모델에 포함된 청크 종류, 재질 구성, 텍스처 경로와 메시 데이터의 유효성에 따라 달라질 수 있습니다.

항목	지원 여부	설명
메시 정점-삼각형 면-UV	지원	형상과 텍스처 좌표를 읽고 법선을 계산합니다.
기본 재질-외부 텍스처	일부 지원	기본 색상, 투명도와 주요 텍스처를 처리합니다. 스무딩과 일부 재질 표현은 제작 화면과 다를 수 있습니다.
레이어 위치-회전-크기	지원	3DS 파일과 별도로 모델 전체의 배치값을 적용할 수 있습니다.
XML 메타정보	지원	모델 목록, 수평 범위, 기준 위치, 회전과 크기를 입력할 수 있습니다.
3DS 내부 Position	지원	GeOnDT for JS 1.0.0.7 이상에서 POS_TRACK_TAG의 위치값을 적용합니다.
3DS 내부 회전-크기 트랙과 객체 계층	확인 필요	여러 변환을 함께 사용한 결과는 실제 파일에서 확인합니다.
MESH_MATRIX 변환	미지원	현재 문서의 기준 버전에서는 이 청크에만 저장된 변환이 화면에 반영되지 않습니다.

3. 입력 방식

입력 자료를 다운로드 받을 수 있는 URL을 관련 레이어 또는 API에 입력합니다.

- baseUrl: 3DS 파일과 외부 리소스의 기본 URL
- fileName: 로드할 .3ds 파일명

관련 레이어:

- GeOnDT.model.U3dModelBasicLayer
- GeOnDT.model.U3dModelTdsLayer

관련 API:

- app.createTdsModelLayer()

모델 정점은 원점 주변의 로컬 좌표로 작성하고, 지도상의 위치는 레이어 배치값 또는 XML 메타정보로 분리하여 입력합니다. 각 위치 입력의 좌표 기준은 다음과 같습니다.

입력 위치	수평 좌표 기준	설명
3DS 메시 정점	모델 원점 기준 로컬 좌표	형상 중심 또는 바닥 중심 주변의 작은 좌표로 작성합니다.
레이어 position	EPSG:3857 월드 좌표	GeOnDT for JS 좌표계 변환 API로 경도·위도를 변환하여 입력할 수 있습니다.
3DS 내부 Position.X, Position.Y	EPSG:3857	GeOnDT for JS 1.0.0.7 이상에서 지도상의 실제 위치로 사용합니다.
XML BoundingBox	EPSG:3857	모델 전체의 수평 범위를 입력합니다.
XML Location.x, Location.y	EPSG:4326 경도·위도	모델의 기준 위치를 입력합니다.
레이어·XML의 Z값	데이터별 높이 기준	단위와 지표면·평균 해수면·프로젝트 기준 중 어떤 기준인지 별도로 기록합니다.

레이어의 position과 3DS 파일 내부의 Position은 서로 다른 값입니다. 소문자 position은 레이어 배치 옵션, 대문자 Position은 3DS 객체의 위치 트랙을 의미합니다.

파일 구성과 외부 리소스

일반적인 입력은 3DS 파일, 파일에서 참조하는 외부 텍스처와 좌표정보 문서입니다. 위치와 모델 목록을 파일과 분리하여 관리할 때는 XML 메타정보를 선택적으로 함께 배포합니다.

```

1 /data/models/sample/
2 |— sample.3ds
3 |— coordinate-info.txt
4 |— sample.xml      # XML 방식을 사용하는 경우
5 |— textures/
6 |   |— wall.jpg
7 |   |— roof.png
  
```

위 구조는 예시입니다. 3DS 재질이 참조하는 텍스처의 상대 경로와 파일명 대소문자를 실제 배포 구조와 일치시킵니다.

3DS에는 EPSG 식별 정보가 없으므로 coordinate-info.txt와 같은 문서에 다음 내용을 기록합니다.

```

1 수평 좌표계: EPSG:3857
2 수평 좌표 단위: 미터
3 모델 정점: 바닥 중심을 원점으로 하는 로컬 좌표
4 지도 위치: 3DS Position·레이어 position은 EPSG:3857, XML Location은 EPSG:4326
5 높이 단위: 미터
6 높이 기준: 지표면 기준 상대 높이
  
```

XML을 사용하는 경우에는 다음과 같이 작성할 수 있습니다. 값은 구조를 설명하기 위한 예시이며, 실제 모델에서 계산한 범위와 위치로 교체합니다.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <Root>
3    <Title>sample</Title>
4
5
6
7    <SRS>EPSG:3857</SRS>
8
9
10
11   <BoundingBox
12     minx="14133608.682"
13     miny="4512420.361"
14     maxx="14133708.682"
15     maxy="4512520.361"/>
16
17
18
19   <Location
20     x="126.96481614997346"
21     y="37.524504862572274"
22     z="0"/>
23
24   <Scale x="1" y="1" z="1"/>
25   <Rotation x="0" y="0" z="0"/>
26
27
28
29   <files>
30     <file>sample.3ds</file>
31   </files>
32 </Root>
```

- SRS는 사람이 좌표 기준을 확인하기 위한 정보이며 GeOnDT가 해석하거나 검증하는 값은 아닙니다. 값을 변경해도 BoundingBox가 자동 변환되지 않으므로 BoundingBox 값 자체를 EPSG:3857로 작성합니다.
- Location.x에는 경도, Location.y에는 위도를 입력합니다. EPSG:3857 X-Y 값을 Location에 입력하지 않습니다.

- BoundingBox는 회전과 크기를 적용한 최종 모델의 전체 수평 범위를 포함하도록 계산합니다.
- Rotation은 도 단위이며 Scale은 축별 배열입니다.
- <file>에는 실제로 불러올 3DS 파일만 입력합니다. 텍스처 전용 JPG-PNG 파일은 넣지 않습니다.
- 실제 모델 경로는 <file>과 레이어의 기본 경로로 결정합니다. ModelUrl이나 TextureUrl만 추가하여 경로가 바뀐다고 가정하지 않습니다.

오류 및 누락 시 결과

- 파일을 불러오지 못하거나 메시 청크의 정점·면 정보가 올바르지 않으면 모델이 표시되지 않을 수 있습니다.
- 텍스처가 누락된 경우 메시가 기본 재질이나 텍스처가 없는 상태로 표시될 수 있습니다.
- 레이어 position에 경도·위도를 직접 입력하거나 XML Location에 EPSG:3857 좌표를 입력하면 모델이 잘못된 위치에 표시될 수 있습니다.
- 정점에 큰 EPSG:3857 절대좌표를 직접 저장하면 형상 세부가 흔들리거나 위치 정밀도가 낮아질 수 있습니다.
- 정점, 레이어 position, XML 또는 3DS 내부 Position에 같은 이동량을 중복 적용하면 모델이 두 번 이동할 수 있습니다.
- XML에서 Title, BoundingBox, Location, Scale, Rotation 또는 <file>이 누락되거나 값이 올바르지 않으면 모델이 표시되지 않거나 위치·크기·회전이 잘못 적용될 수 있습니다.

4. 코드 예제

3DS 파일 URL과 레이어 배치값을 직접 입력하는 예시입니다.

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const modelBaseUrl = "/data/models/sample/";
3
4 // 원본 경도·위도·높이를 GeOnDT의 EPSG:3857 월드 좌표로 변환합니다.
5 const modelPosition = app.getGeographicToWorld(
6     126.96481614997346,
7     37.524504862572274,
8     0
9 );
10
11 const layer = app.createTdsModelLayer({
12     name: "3DS_Model",
13     baseUrl: modelBaseUrl,
14     needXml: false,
15     position: modelPosition,
16     rotation: { x: 0, y: 0, z: 0 },
17     scale: { x: 1, y: 1, z: 1 },
18     listModel: [
19         {
20             name: "sample-3ds",
21             fileName: "sample.3ds",
22             ext: ".3ds"
23         }
24     ]
25 });
26
```

```

27 if (!layer) {
28   console.error("3DS 레이어를 생성하지 못했습니다.");
29 } else {
30   // createTdsModelLayer()가 레이어를 등록하므로 app.addLayer()는 호출하지 않습니다.
31   layer.ready
32     .then(() => {
33       app.showLayer(layer.getName(), true);
34     })
35     .catch((error) => {
36       console.error("3DS 로딩 실패", error);
37       app.removeLayer(layer.getName());
38     });
39 }

```

XML 메타정보를 사용하는 경우에는 XML의 Location과 동일한 위치를 생성 시점의 기준 위치로 입력합니다. 생성 시 전달한 position은 초기화에 사용되고, XML을 읽은 뒤의 최종 위치는 XML Location이 결정합니다.

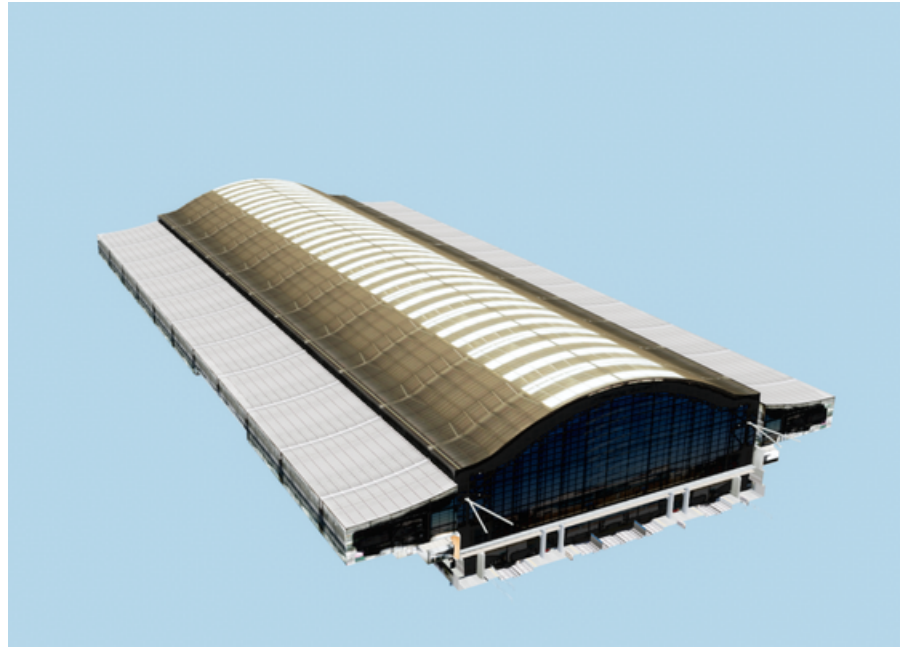
```

1  const modelBaseUrl = "/data/models/sample/";
2  const xmlLocation = app.getGeographicToWorld(
3    126.96481614997346,
4    37.524504862572274,
5    0
6  );
7
8  const layer = app.createTdsModelLayer({
9    name: "3DS_XML_Model",
10   baseUrl: modelBaseUrl,
11   path: modelBaseUrl,
12   needXml: true,
13   xmlUrl: `${modelBaseUrl}sample.xml`,
14   position: xmlLocation
15 });
16
17 if (layer) {
18   layer.ready
19     .then(() => app.showLayer(layer.getName(), true))
20     .catch((error) => {
21       console.error("3DS XML 로딩 실패", error);
22       app.removeLayer(layer.getName());
23     });

```


5. 출력 예시

위치, 회전, 크기와 텍스처가 올바르게 연결되면 3DS 형상이 지정한 지도 위치에 표시됩니다.



GeOnDT for JS에서 출력된 3DS 결과 화면

6. 권장 사항

필수 조건

- 모델 정점은 형상 중심 또는 바닥 중심을 원점으로 하는 유한한 로컬 좌표로 작성하고, 큰 EPSG:3857 절대좌표를 정점마다 직접 저장하지 않습니다.
- 지도 위치는 3DS 내부 Position, 레이어 position 또는 XML 중 하나를 기준으로 관리합니다. 같은 이동량을 여러 위치에 중복 입력하지 않습니다.
- 3DS 내부 Position, 레이어 position과 XML BoundingBox는 EPSG:3857, XML Location.x-Location.y는 EPSG:4326 경도-위도로 작성합니다.
- EPSG:3857은 수평 좌표계이므로 Z값의 단위와 높이 기준을 데이터와 함께 제공하는 좌표정보 문서에 별도로 기록합니다.
- XML을 사용할 때는 Title, BoundingBox, Location, Scale, Rotation과 하나 이상의 <file>을 올바르게 작성합니다.
- 3DS가 외부 텍스처를 참조하는 경우 텍스처를 접근 가능한 URL에 배포하고 상대 경로와 파일명 대소문자를 일치시킵니다.

권장 조건

- 모델 단위는 미터로 통일하고 X축은 동쪽, Y축은 북쪽, Z축은 위쪽을 향하도록 작성합니다.
- 지면에 배치하는 모델은 원점을 바닥 중심에 두고, 중앙 회전과 크기 조절이 중요한 모델은 원점을 형상 중심에 둡니다.
- 여러 객체가 하나의 3DS를 구성하면 공통 루트 객체에 실제 지도 위치를 입력하고 자식 객체는 루트 기준의 로컬 위치를 유지합니다.
- 정적인 모델은 프레임 0에 하나의 Position 키를 기록하고, 시간에 따라 변하지 않는 불필요한 위치 키를 만들지 않습니다.
- 내보낸 3DS를 다시 불러와 정점이 로컬 좌표로 유지되는지와 위치·높이·회전·크기·텍스처가 의도대로 보이는지 확인합니다.
- 내보낸 파일에서 Position이 POS_TRACK_TAG에 유지되고 정점 좌표나 MESH_MATRIX에 같은 이동량이 중복 반영되지 않았는지 확인합니다.
- 불필요한 객체, 재질, 면과 텍스처를 제거하고 폴리곤 수와 텍스처 크기를 서비스 환경에 맞게 조정합니다.

선택 조건

- 여러 3DS 파일의 위치와 범위를 함께 관리하거나 파일 내부 변환을 유지하기 어려운 경우 XML 메타정보를 사용합니다.
- GeOnDT for JS 1.0.0.7 이상에서 3DS 내부 Position을 사용하는 경우 X·Y는 EPSG:3857, Z는 데이터별 높이 기준으로 작성하고 좌표계·단위·높이 기준을 외부 좌표정보 문서에도 기록합니다.
- 서버미터 수준의 위치 정확도가 필요하면 큰 좌표값의 반올림 여부를 확인하고, 레이어 배치값 또는 XML 방식과 결과를 비교합니다.

확인 필요

- 3DS 내부 Position, 회전·크기 트랙, 객체 계층과 MESH_MATRIX를 함께 사용하는 데이터는 내보내기 도구에 따라 변환값의 저장 위치가 달라질 수 있으므로 최종 배치 결과를 확인합니다.

WFS

1. 포맷 정의와 공식 규격

WFS는 HTTP를 통해 점, 선, 면과 같은 지리 피처와 속성 정보를 조회하는 OGC 웹 서비스입니다.

GeOnDT for JS에서는 WFS 서버에 GetFeature 요청을 전송하고, 반환된 JSON 형식의 피처를 3D 모델 또는 벡터 레이어로 표현합니다.

외부 공식 규격

항목	내용
규격명	Web Feature Service Standard
발행 기관	Open Geospatial Consortium
조회일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 서비스/벡터 피처

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

버전은 1.0.0과 1.1.0을 지원합니다.

실제 지원 여부는 WFS 버전뿐 아니라 서버 응답 형식, 피처 지오메트리, 좌표계, 필수 속성 필드와 CORS 설정에 따라 달라질 수 있습니다.

3. 입력 방식

WFS 서비스의 URL과 조회할 레이어명을 레이어 생성 옵션으로 입력합니다.

- baseurl: WFS 서비스의 기본 URL입니다.
- layername: 조회할 WFS 피처 타입 또는 레이어명입니다.
- version: 서버에서 지원하는 1.0.0 또는 1.1.0입니다.
- ext: 서버가 반환할 JSON 응답 형식입니다.
- crs: 요청과 응답에 사용할 좌표계입니다.
- minlevel: 레이어를 표시할 최소 화면 레벨입니다.
- fieldheight: 3D 높이값으로 사용할 숫자 속성명입니다.
- fieldpk: 각 피처를 구분할 고유 속성명입니다.

파일 구성과 외부 리소스

```
1  
2 https://example.com/geoserver/wfs  
3   ├── service=WFS  
4   ├── request=GetFeature  
5   ├── version=1.1.0  
6   ├── typeName=workspace:building  
7   ├── outputFormat=application/json  
8   └── srsName=EPSG:3857
```

WFS는 파일을 직접 업로드하는 방식이 아니라 서버에서 피쳐 데이터를 요청하는 서비스 방식입니다.

WFS 서버는 브라우저에서 직접 접근할 수 있어야 하며, 다른 도메인의 서비스를 사용하는 경우 CORS 허용 또는 프록시 구성이 필요할 수 있습니다.

오류 및 누락 시 결과

- 서비스 URL, 레이어명 또는 버전이 서버 설정과 다르면 GetFeature 요청이 실패하거나 빈 결과가 반환될 수 있습니다.
- 응답이 JSON 형식이 아니거나 지원하지 않는 지오메트리를 포함하면 일부 또는 전체 피처가 표시되지 않을 수 있습니다.
- CORS가 허용되지 않으면 브라우저에서 WFS 응답을 불러올 수 없습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerOpt = {
3   name: 'WFS_Building', // 2D와 3D 레이어에서 사용할 이름입니다.
4   baseurl: 'https://example.com/geoserver/wfs',
5   layername: 'workspace:building',
6   minlevel: 17
7 };
8 let wfs2dLayer, wfs3dLayer;
9
10 try { // 2D WFS 피쳐 레이어를 표시합니다.
11   wfs2dLayer = new GeOnDT.vector.U2dVectorShaderLayer(layerOpt);
12   app.addLayer(wfs2dLayer);
13   app.showLayer(wfs2dLayer.getName(), true);
14 } catch (error) {
15   console.error('WFS 레이어 생성 실패', error);
16   if (wfs2dLayer) {
17     app.removeLayer(wfs2dLayer.getName());
18   }
19 }
20
21 try { // 3D WFS 피쳐 레이어를 표시합니다.
22   wfs3dLayer = new GeOnDT.model.U3dModelWFSLayer(layerOpt);
23   app.addLayer(wfs3dLayer);
24   app.showLayer(wfs3dLayer.getName(), true);
25 } catch (error) {
26   console.error('WFS 레이어 생성 실패', error);
27   if (wfs3dLayer) {
28     app.removeLayer(wfs3dLayer.getName());
29   }
30 }
31
32 // 제거: app.removeLayer(wfs2dLayer.getName());
33 // 제거: app.removeLayer(wfs3dLayer.getName());
```


5. 출력 예시



GeOnDT for JS에서 출력된 WFS 2D 결과 화면



GeOnDT for JS에서 출력된 WFS 3D 결과 화면

6. 권장 사항

필수 조건

- WFS 서버는 GetFeature 요청을 지원해야 하며, GeOnDT에서 처리할 수 있는 JSON 피쳐 응답을 반환해야 합니다.
- 요청에 사용하는 layername, version, ext, crs 값이 실제 WFS 서비스 설정과 일치해야 합니다.
- 3D 높이 표현을 사용하는 경우 fieldheight에 지정한 속성은 숫자로 변환 가능한 값이어야 합니다.
- 피쳐 구분에 사용하는 fieldpk는 각 피쳐를 고유하게 식별할 수 있어야 합니다.

권장 조건

- 요청 srsName, 서버 응답 좌표계와 GeOnDT 레이어의 crs 값을 일치시키는 것을 권장합니다.

선택 조건

- WFS 응답에 화면 표시와 관계없는 속성이 많은 경우 필요한 속성만 조회하여 네트워크 전송량과 메모리 사용량을 줄일 수 있습니다.
- 폴리곤 정점이 지나치게 많은 경우 서버 또는 사전 처리 단계에서 지오메트리를 단순화할 수 있습니다.

WMS

1. 포맷 정의와 공식 규격

WMS(Web Map Service)는 서버가 제작한 지리참조 지도 이미지를 요청하는 웹 서비스입니다. 클라이언트가 지도 범위(BBOX), 좌표계, 이미지 크기, 레이어와 이미지 형식을 전달하면 서버가 해당 범위를 PNG 또는 JPEG 등의 이미지로 반환합니다.

GeOnDT for JS는 현재 화면을 타일 영역으로 나누고 각 영역에 대한 GetMap 요청을 생성합니다. WMS 응답은 화면 표시용 이미지이므로 개별 지형지물의 형상이나 속성을 직접 제공하지 않습니다.

외부 공식 규격

항목	내용
규격명	Web Map Service Standard
발행 기관	Open Geospatial Consortium
조회일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 웹 서비스/지도 이미지

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

WMS 1.3.0을 기본으로 사용하며 WMS 1.1.0 GetMap 요청도 지원합니다. 1.3.0 요청에는 CRS, 1.1.0 요청에는 SRS 파라미터를 사용합니다. 화면 표시에 지원되는 좌표계는 EPSG:3857입니다.

항목	지원 여부	설명
WMS 1.3.0 GetMap	지원	기본 버전이며 CRS=EPSG:3857로 요청합니다.
WMS 1.1.0 GetMap	지원	SRS=EPSG:3857로 요청합니다.
EPSG:3857	지원	화면 타일의 Web Mercator 범위를 BBOX로 사용합니다.
PNG/JPEG 지도 이미지	지원	브라우저에서 읽을 수 있고 서버가 제공하는 이미지 형식을 사용할 수 있습니다.
GetCapabilities 자동 분석	미지원	서비스 주소, 레이어명과 요청 조건을 사용자가 입력합니다.
GetFeatureInfo-POST 요청	미지원	현재 WMS 레이어는 GET 방식의 GetMap 이미지 표시에 사용합니다.

layer.ready 완료는 레이어가 앱에서 사용할 준비가 되었다는 뜻입니다. 카메라 이동에 따라 이후 요청되는 모든 WMS 이미지의 수신 성공까지 의미하지는 않습니다.

3. 입력 방식

지도 이미지를 요청할 수 있는 WMS 서비스 URL을 관련 레이어 또는 생성 API에 입력합니다.

관련 레이어 및 API:

- `GeOnDT.image.U3dImageWMSLayer`
- `app.createWMSImageLayer()`
- `app.createWMSXYZImageLayerEx()`

파일 구성과 외부 리소스

WMS는 로컬 파일이 아니라 HTTP(S) 서비스 주소와 서버가 반환하는 이미지로 구성됩니다. 서비스는 GeOnDT가 실행되는 브라우저에서 접근할 수 있어야 하며, 다른 출처의 서비스를 직접 호출하려면 서버가 CORS를 허용해야 합니다. 직접 접근이 허용되지 않으면 프로젝트의 프록시를 구성합니다.

스타일과 필터는 WMS 서버에 미리 등록하거나 서버가 해석할 수 있는 값이어야 합니다. 쿼리에 공백이나 특수 문자가 포함되면 서버가 요구하는 URL 인코딩 방식도 함께 적용합니다.

오류 및 누락 시 결과

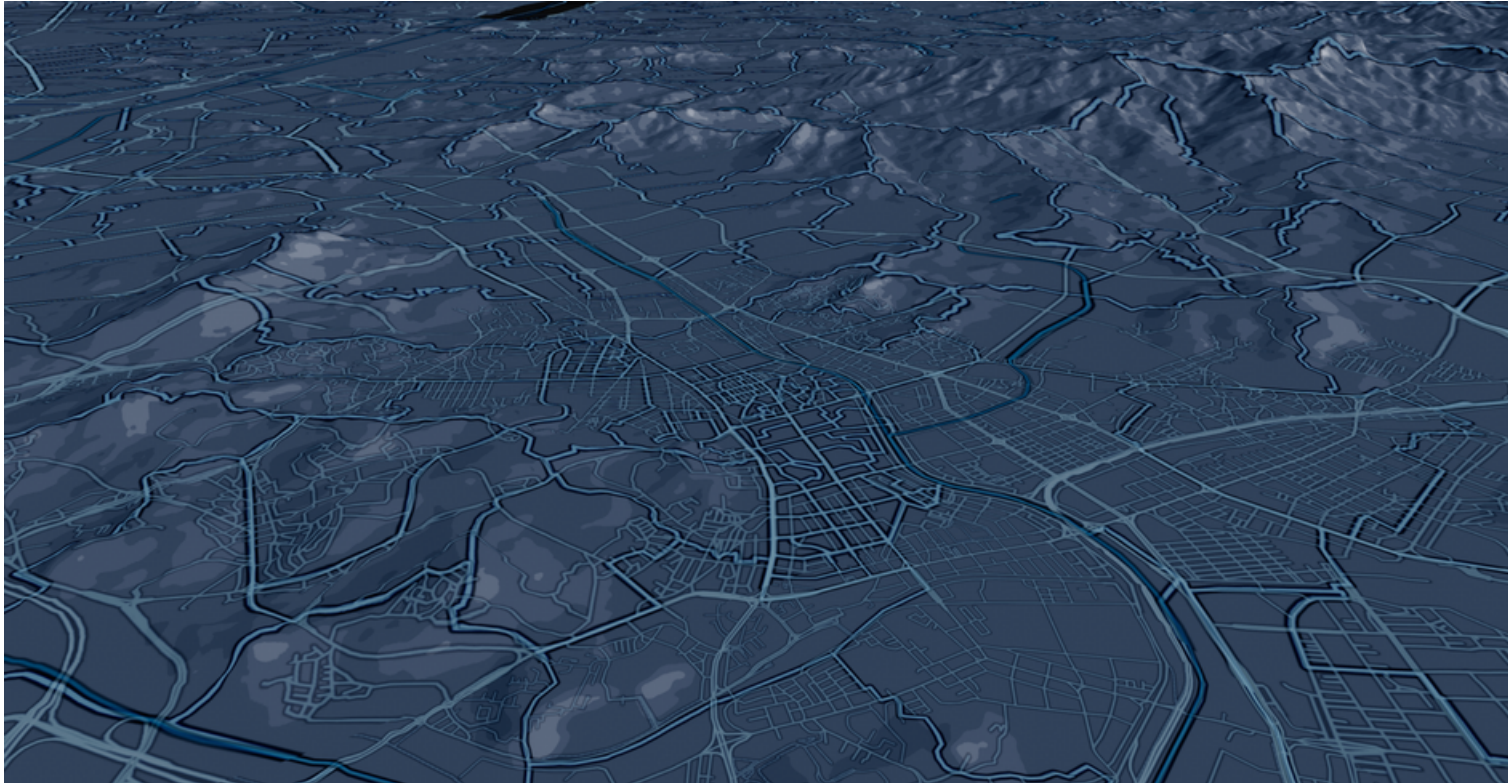
- 서비스 주소나 레이어명이 올바르지 않으면 유효한 GetMap 이미지를 받을 수 없어 지도가 표시되지 않습니다.
- crs가 정확히 EPSG:3857이 아니면 이미지 요청을 진행하지 않습니다.
- 버전, 레이어명, 스타일 또는 이미지 형식이 서버 설정과 다르면 서버 오류 문서, 빈 이미지 또는 읽을 수 없는 응답이 반환될 수 있습니다.
- HTTP 오류, CORS 차단 또는 이미지 처리 오류가 발생하면 해당 영역이 상위 레벨 이미지로 임시 표시되거나 비어 보일 수 있습니다.
- minLevel과 maxLevel 범위 밖에서는 WMS 이미지를 요청하지 않습니다.
- 레이어가 준비된 뒤에도 개별 이미지 요청 오류는 발생할 수 있습니다. 일부 영역만 비어 보이면 브라우저 네트워크 도구에서 요청 URL, HTTP 상태와 응답 본문을 확인합니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerName = "WMS_Map";
3
4 const layer = new GeOnDT.image.U3dImageWMSLayer({
5   name: layerName,
6
7   // GetMap 파라미터를 제외한 WMS 서비스 주소를 입력합니다.
8   baseUrl: "https://maps.example.com/geoserver/workspace/wms",
9   layerName: "workspace:layer",
10  version: "1.3.0",
11  crs: "EPSG:3857",
12  ext: "image/png",
13
14  minLevel: 6,
15  maxLevel: 19,
16  transparent: true,
17  useProxy: false
18 });
19
20 // 직접 생성한 WMS 레이어는 앱에 별도로 등록합니다.
21 app.addLayer(layer);
22
23 layer.ready.then(
24   () => {
25     // 이후 필요한 GetMap 이미지는 카메라 위치에 따라 계속 요청됩니다.
26     app.showLayer(layerName, true);
27   },
28   (error) => {
29     console.error("WMS 레이어 준비 실패", error);
30     app.removeLayer(layerName);
31   }
32 );
33
34 // 숨기기: app.showLayer(layerName, false);
35 // 제거하기: app.removeLayer(layerName);
```

5. 출력 예시

WMS 레이어는 현재 화면 범위에 해당하는 서버 지도 이미지를 지면 위에 이어서 표시합니다. 투명 PNG를 사용하면 도로, 행정 경계나 주제도 등을 기존 배경지도 위에 중첩할 수 있습니다.



6. 권장 사항

필수 조건

- WMS 서비스가 EPSG:3857의 GetMap 요청과 선택한 버전, 레이어 및 이미지 형식을 제공해야 합니다.
- baseUrl에는 서비스 주소를, layerName에는 서버에 등록된 정확한 레이어 이름을 입력합니다.
- crs를 지정하는 경우에는 대소문자를 포함하여 정확히 EPSG:3857로 입력합니다. 생략하면 이 값이 기본으로 적용됩니다.
- 서비스와 모든 이미지 URL은 브라우저에서 접근할 수 있어야 합니다. 교차 출처에서는 CORS를 허용하거나 프록시를 사용합니다.
- WMS 결과는 화면 표시용 이미지입니다. 개별 지형지물의 속성·형상·높이를 조회하거나 편집해야 하면 WFS 등의 벡터 서비스를 별도로 사용합니다.

권장 조건

- 실제 서비스에 필요한 minLevel과 maxLevel만 사용하여 지나치게 넓은 범위의 요청을 줄입니다.
- 투명한 중첩 지도에는 PNG를 사용하고, 투명도가 필요 없는 항공 영상이나 배경지도는 서버 품질을 확인한 뒤 JPEG를 검토합니다.
- 기본 256×256 요청에서 시작하고, 이미지 크기를 변경할 때는 서버 처리량과 화면 품질을 함께 확인합니다.
- 최소·최대 레벨과 데이터 경계에서 실제 응답이 올바른 위치에 표시되는지 확인합니다.
- 반복 요청이 많은 서비스는 서버의 타일 캐시를 사용할 수 있도록 구성합니다. TILED=true 같은 추가 쿼리는 서버가 지원할 때만 사용하며, 이 값만으로 캐시 사용이 보장되지는 않습니다.
- 운영 환경에서는 HTTPS를 사용하고, 서버 오류 문서가 이미지로 오인되지 않도록 HTTP 상태와 응답 형식을 함께 점검합니다.

선택 조건

- 서버가 제공하는 주제 표현이 필요하면 styles를 지정합니다.
- GeoServer 등 CQL 필터를 지원하는 서버에서는 cqlFilter로 표시 대상을 제한할 수 있습니다.
- key와 styles를 동시에 입력하면 STYLES가 요청에서 제외됩니다. 현재 제품 버전에서는 두 옵션을 함께 사용하지 않습니다.
- 서버별 추가 요청값은 appendQuery에 입력하고, 값에 필요한 URL 인코딩을 적용합니다.
- 여러 영상 레이어를 겹쳐 표현할 때는 투명도와 blendingType을 화면 목적에 맞게 조정합니다.

TMS

1. 포맷 정의와 공식 규격

TMS(Tile Map Service)는 미리 제작한 지도 이미지를 줌 레벨(z), 타일 열(x), 타일 행(y) 경로로 제공하는 웹 서비스 방식입니다. 각 타일셋은 좌표계, 공간 범위, 원점, 이미지 형식, 타일 크기와 제공 레벨 정보를 함께 사용합니다.

일반적인 TMS는 좌하단을 원점으로 사용하여 Y 번호가 남쪽에서 북쪽으로 증가합니다. 좌상단 원점에서 Y 번호가 북쪽에서 남쪽으로 증가하는 일반적인 XYZ 방식과 행 번호가 반대이므로, 같은 타일 이미지를 사용하더라도 서버의 Y 방향을 정확히 맞춰야 합니다.

TMS 1.0.0은 OSGeo 커뮤니티에서 제안한 참고 사양입니다.

- 공식 규격/참고: [OSGeo Tile Map Service Specification](#)
- 유형: 웹 서비스/지도 이미지 타일

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

GeOnDT for JS는 TMS 1.0.0의 z/x/y 타일 주소 체계를 기반으로 이미지 타일을 직접 표시합니다. 전체 TMS 서비스를 자동으로 탐색하거나 버전을 판별하지는 않으며, 타일셋 루트 URL과 GeOnDT에서 사용하는 메타데이터 객체를 함께 입력합니다.

항목	지원 여부	설명
TMS 이미지 타일	지원	타일셋 루트 아래의 레벨·열·행 경로에서 이미지를 요청합니다.
mercator 프로파일	지원	Web Mercator 타일을 표시하며 범위는 경도·위도로 입력합니다.
raster 프로파일	지원	타일셋의 좌표계, 범위, 원점과 해상도 정보를 사용합니다.
열 우선·행 우선 폴더 구조	지원	기본은 z/x/y, RowFirst는 z/y/x 구조입니다.
TMS-XYZ Y 방향 선택	지원	metadata.tileFormat.reverseY 값으로 타일 행 방향을 맞춥니다.
PNG/JPEG 이미지 타일	지원	브라우저에서 읽을 수 있는 이미지 타일을 사용합니다.
TileMapService.TileMap 문서 자동 분석	미지원	서비스 문서 URL 대신 준비된 metadata 객체를 입력합니다.
지형지물 속성·벡터 형상	미지원	TMS 레이어는 화면 표시용 이미지 타일입니다.

3. 입력 방식

지도 이미지 타일을 다운로드 받을 수 있는 TMS 타일셋 URL을 관련 생성 API에 입력합니다.

관련 API:

- `app.createTMSImageLayer()`
- `app.createTMSImageLayerAsXML()`

파일 구성과 외부 리소스

기본 폴더 구조는 `z/x/y.ext`입니다. `metadata.majorFolder`를 "RowFirst"로 지정한 타일셋은 `z/y/x.ext` 구조를 사용합니다.

```
1 /data/tms/sample/  
2 |— 7/  
3 | |— 109/  
4 | | |— 77.png  
5 | | |— 78.png  
6 | |— 110/  
7 | |— 77.png  
8 | |— 78.png  
9 |— 8/  
10 |— ...
```

위 숫자는 폴더 구조를 설명하기 위한 예시입니다. 실제 레벨과 타일 번호는 타일셋에 따라 달라집니다. `metadata`는 이 폴더에서 자동으로 찾는 파일이 아니라 `createTMSImageLayer()`에 전달하는 JavaScript 객체입니다. 서버가 별도의 메타데이터 API를 제공한다면 그 응답을 같은 구조로 전달할 수 있습니다.

오류 및 누락 시 결과

- `name`, `baseurl` 또는 필수 `metadata` 항목이 없으면 레이어를 생성할 수 없습니다.
- 같은 이름의 레이어가 이미 등록되어 있으면 새 레이어를 반환하지 않습니다.
- 좌표계, 공간 범위, 원점 또는 레벨별 해상도가 실제 타일셋과 다르면 타일이 다른 위치에 표시되거나 화면이 비어 보일 수 있습니다.
- 열·행 폴더 순서나 Y 방향이 다르면 404 오류가 발생하거나 타일이 상·하·좌·우로 뒤바뀝니다.
- 이미지 확장자, 레벨 범위 또는 URL이 잘못되면 해당 영역이 상위 레벨 이미지로 임시 표시되거나 비어 보일 수 있습니다.
- CORS 차단, HTTP 오류와 손상된 이미지 타일은 레이어가 준비된 뒤에도 개별 영역의 표시 실패로 나타날 수 있습니다.
- `layer.ready` 완료는 레이어의 기본 설정이 준비되었다는 뜻이며, 카메라 이동에 따라 이후 요청되는 모든 타일의 수신 완료를 의미하지 않습니다.

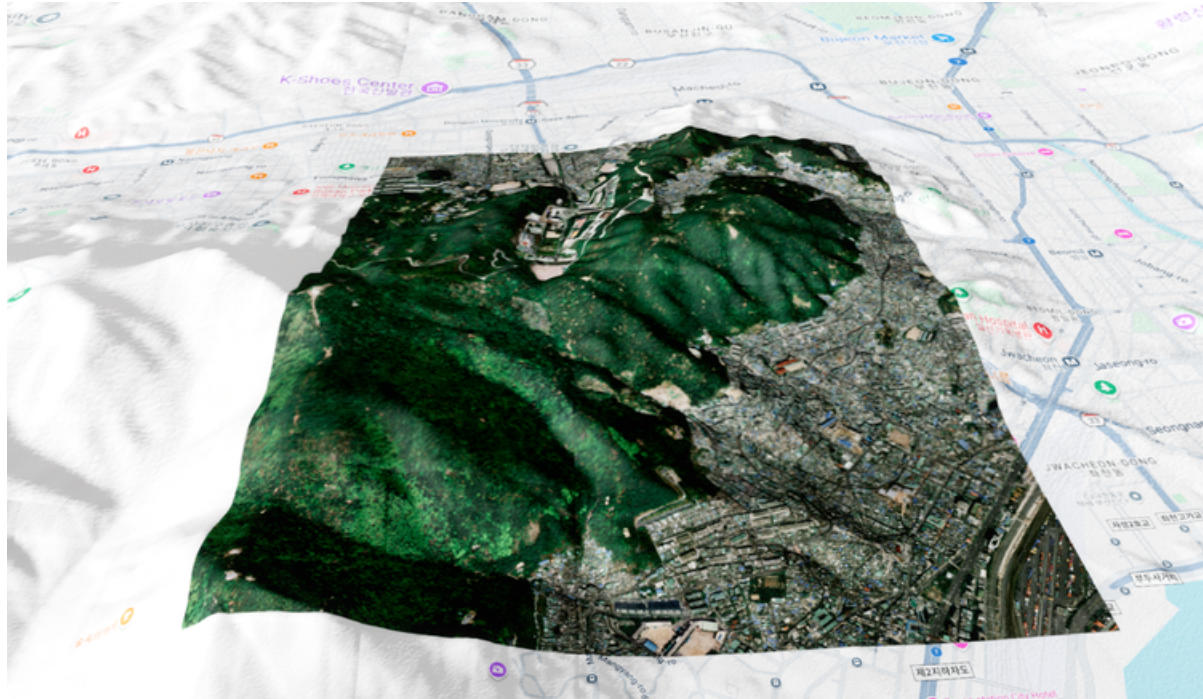
4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerName = "Seoul_TMS";
3
4 try {
5   const layer = app.createTMSImageLayer({
6     name: layerName,
7
8     // {z}/{x}/{y}를 포함하지 않은 타일셋 루트 URL입니다.
9     baseUrl: "/data/tms/seoul",
10    metadata: {
11      srs: "EPSG:3857",
12      profile: "mercator",
13
14      // mercator 프로파일에서는 경도-위도 범위를 입력합니다.
15      boundingBox: {
16        minx: 126.8,
17        miny: 37.4,
18        maxx: 127.2,
19        maxy: 37.7
20      },
21      tileFormat: {
22        ext: "png",
23        width: 256,
24
25        // false는 일반적인 좌하단 원점 TMS의 Y 번호를 사용합니다.
26        reverseY: false
27      },
28      raster: {
29        maxlevel: 19
30      }
31    },
32    minlevel: 7,
33    maxlevel: 19,
34    useproxy: false
35  });
36
37  if (!layer) {
38    throw new Error("같은 이름의 레이어가 이미 존재합니다.");
39  }
```

```
40
41 // createTMSImageLayer()가 레이어를 앱에 자동으로 등록합니다.
42 layer.ready.then(
43   () => {
44     // 이후 필요한 타일은 카메라 위치와 화면 레벨에 따라 계속 요청됩니다.
45     app.showLayer(layerName, true);
46   },
47   (error) => {
48     console.error("TMS 레이어 준비 실패", error);
49     app.removeLayer(layerName);
50   }
51 );
52 } catch (error) {
53   console.error("TMS 레이어 생성 실패", error);
54 }
55
56 // 숨기기: app.showLayer(layerName, false);
57 // 제거하기: app.removeLayer(layerName);
```

5. 출력 예시

TMS 레이어는 현재 카메라 위치와 화면 레벨에 맞는 지도 이미지 타일을 이어서 지면 위에 표시합니다. 타일 좌표계, 공간 범위, 폴더 순서와 Y 방향이 모두 맞으면 타일 경계에서도 지도가 연속적으로 연결됩니다.



6. 권장 사항

필수 조건

- 타일셋 루트 URL과 srs, profile, boundingBox, tileFormat, raster 메타데이터를 함께 제공합니다.
- 타일의 좌표계, 공간 범위, 원점, 이미지 확장자, X/Y 폴더 순서, Y 방향과 레벨을 실제 배포된 타일셋과 일치시킵니다.
- mercator 프로파일의 boundingBox는 srs가 EPSG:3857이어도 경도·위도 값으로 입력합니다.
- mercator 프로파일은 256×256 정사각형 타일을 사용합니다. raster 프로파일은 실제 정사각형 타일의 너비를 tileFormat.width에 입력합니다.
- 타일 URL은 브라우저에서 접근할 수 있어야 합니다. 교차 출처에서는 CORS를 허용하거나 프록시를 사용합니다.
- TMS 결과는 화면 표시용 이미지입니다. 개별 지형지물의 속성·형상·높이를 조회하거나 편집하는 데이터로 사용하지 않습니다.

권장 조건

- minlevel과 maxlevel을 실제 제공 범위로 명시하고, 최소:최대 레벨의 네 모서리와 데이터 경계에서 요청 URL과 표시 위치를 확인합니다.
- 먼저 대표 지역을 표시하여 타일이 상하·좌우로 뒤집히지 않는지, 인접 타일이 끊김 없이 연결되는지 확인합니다.
- 실제 서비스 범위와 화면 용도에 필요한 레벨만 제공하여 빈 타일, 404 응답과 불필요한 고해상도 요청을 줄입니다.
- 투명한 주제도에는 PNG를 사용하고, 투명도가 필요 없는 배경 영상은 품질을 확인한 뒤 JPEG를 검토합니다.
- 변경 주기가 긴 정적 타일은 적절한 HTTP 캐시 헤더를 사용하고 운영 환경에서는 HTTPS를 사용합니다.

선택 조건

- 타일 서버의 Y 방향에 따라 metadata.tileFormat.reverseY를 지정합니다.
- 타일 폴더가 행 우선 구조이면 metadata.majorFolder: "RowFirst"를 사용합니다.
- 별도 좌표계의 타일셋은 profile: "raster"와 origin, maxresolution, 필요한 경우 proj.proj를 함께 사용합니다.
- 단순 URL 템플릿만 사용할 때는 XYZ 레이어에서 {y} 또는 {-y}를 명시하는 방식을 선택할 수 있습니다.
- 직접 접근할 수 없는 서비스는 프로젝트의 프록시를 사용할 수 있습니다.

XYZ

1. 포맷 정의와 공식 규격

XYZ는 지도 타일 이미지를 줌 레벨(z), 타일 열(x), 타일 행(y) 인덱스로 요청하는 URL 템플릿 방식입니다.

일반적으로 Web Mercator(EPSG:3857) 기반 Slippy Map 타일 체계를 사용하며, 서버 URL에 {z}, {x}, {y} 값을 치환해 256px 단위의 PNG 또는 JPEG 타일 이미지를 받아옵니다.

- 공식 규격/참고: [Slippy map tilenames](#)
- 유형: 서비스/지도 타일

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

GeOnDT for JS에서는 U3dImageXYZLayer를 통해 XYZ URL 템플릿 기반 이미지 타일을 직접 지원합니다.

URL/타일 요소	지원 여부	비고
{z}	지원	줌 레벨
{x}	지원	타일 열 번호
{y}	지원	타일 행 번호
{-x}	지원	$2^z - x - 1$ 값으로 치환
{-y}	지원	$2^z - y - 1$ 값으로 치환
{a-c} 또는 {1-4} 형태의 URL 분산	지원	URL 목록으로 확장
PNG/JPEG 이미지 타일	지원	브라우저와 UTextureLoader가 디코딩 가능한 이미지
비 Web Mercator 타일 체계	미지원	별도 좌표계 타일은 서버 scheme과 샘플 검증 필요

3. 입력 방식

지도 타일 이미지를 요청할 URL을 {z}(줌 레벨), {x}, {y}(타일 좌표) 형식의 템플릿으로 입력받습니다.

예시 URL 템플릿은 다음과 같습니다.

```
1 https://tiles.example.com/{z}/{x}/{y}.png
2 https://tiles.example.com/{z}/{x}/{-y}.png
3 https://{a-c}.tiles.example.com/{z}/{x}/{y}.jpg
```

예를 들어 {z}=14, {x}=13949, {y}=6346이면 `https://tiles.example.com/14/13949/6346.png` 형태의 이미지 타일 URL로 요청됩니다.

URL 템플릿의 placeholder가 누락되었거나 서버의 Y 방향, 줌 레벨, 타일 크기, 좌표계가 GeOnDT 계산 방식과 다르면 타일이 비어 보이거나 위치가 어긋날 수 있습니다.

외부 서버를 사용하는 경우 브라우저 CORS 정책 때문에 이미지 요청이 실패할 수 있습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layer = new GeOnDT.image.U3dImageXYZLayer({
3   name: "xyz-layer",
4   baseUrl: "https://tiles.example.com/{z}/{x}/{y}.png",
5   reverseY: false,
6   reverseX: false,
7   needXml: false
8 });
9
10 app.addLayer(layer);
11 app.showLayer(layer.getName(), true);
12
13 // 제거: app.removeLayer(layer.getName());
```

5. 출력 예시

✕	헤더	미리보기	응답	시작점	타이밍
▼ 일반					
요청 URL	https://xdworld.vworld.kr/2d/Base/service/18/223502/101561.png				
요청 메서드	GET				
상태 코드	● 200 OK				
원격 주소	223.130.174.74:443				
리퍼리 정책	strict-origin-when-cross-origin				

xyz 데이터 URL 예시



GeOnDT for JS에서 출력된 xyz 데이터 결과 화면

6. 권장 사항

필수 조건

- 서버의 타일 스키마(scheme), 좌표계, 타일 크기, 줌 레벨 원점이 GeOnDT가 사용하는 Google/Slippy 방식의 Web Mercator 타일 체계와 일치해야 합니다.
- URL 템플릿에는 실제 타일 요청에 필요한 {z}, {x}, {y} placeholder가 포함되어야 합니다. 서버가 TMS 방식처럼 Y축이 반전된 좌표를 사용하는 경우 {-y} 등 서버가 요구하는 형식의 placeholder를 사용합니다.
- XYZ 레이어의 결과물은 렌더링용 이미지 타일입니다. 개별 피처의 속성, 지오메트리, Z값을 조회하거나 편집하는 데이터로는 사용할 수 없습니다.
- 외부 이미지 타일 서버는 브라우저의 CORS 정책 영향을 받습니다. GeOnDT가 이미지를 fetch하여 텍스처로 사용할 수 있도록, 서버에 CORS 헤더를 설정하거나 프록시를 구성해야 합니다.

권장 조건

- 최소·최대 레벨의 모서리 타일을 확인하여 좌우·상하 배치, Y 방향, 레벨 원점이 올바른지 검증합니다.
- 실제 서비스 범위에 필요한 레벨만 요청하고 빈 타일, 과도한 해상도, 불필요한 서버 요청을 줄입니다.
- 배경 지도처럼 투명도가 필요 없는 경우 JPEG를, 투명 배경이 필요한 경우 PNG를 우선 검토합니다.

SHP

1. 포맷 정의와 공식 규격

.shp는 ESRI Shapefile 포맷에서 점, 선, 폴리곤 등 지오메트리(도형) 정보를 담고 있는 바이너리 파일입니다.

각 레코드는 Shape Type과 좌표 배열로 구성되며, 파일 전체는 고정 길이의 헤더(100바이트)와 가변 길이의 레코드들로 이루어집니다.

좌표는 X(경도 또는 평면좌표 X), Y(위도 또는 평면좌표 Y) 순서로 저장되며, 좌표계는 .prj 파일에 정의된 값을 따릅니다. .prj가 없는 경우에는 서비스에서 사용할 좌표계를 별도로 확인해야 합니다.

외부 공식 규격

항목	내용
규격명	ESRI Shapefile Technical Description
발행 기관	Esri
조치일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 파일/바이너리 벡터 (다중 파일 세트)

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

.shp에 정의된 Shape Type 전체가 아니라 일부 타입만 지원합니다.

Shape Type	지원 여부	비고
Point	지원	
PolyLine	지원	
Polygon	지원	
MultiPoint	지원	
PointZ / PolyLineZ / PolygonZ / MultiPointZ	미지원	Z값 포함 도형
PointM / PolyLineM / PolygonM / MultiPointM	미지원	M값 포함 도형
MultiPatch	미지원	3D 표면/슬리드 표현
Null Shape	미지원	빈 지오메트리 레코드

3. 입력 방식

URL 또는 Data URI로 읽어 온 .shp 바이너리 데이터를 입력으로 받습니다.

2D 데이터로 출력 할 때는 `GeOnDT.image.U2dShpLayer`를 사용합니다.

3D 데이터로 출력 할 때는 `GeOnDT.model.U3dModelShapeLayer`를 사용합니다.

파일 구성과 외부 리소스

.shp 파일 외에 .shx, .dbf, .prj 파일 세트로 구성할 수 있습니다.

.shp는 도형 정보를 저장하고, .dbf는 속성 정보, .prj는 좌표계 정보를 저장합니다. 속성이나 좌표계가 필요한 경우 같은 데이터 세트의 보조 파일을 함께 준비해야 합니다.

압축된 형태(.zip)는 직접 입력 대상으로 보지 않습니다. 압축을 해제하여 .shp 파일과 보조 파일이 접근 가능한 경로에 존재하도록 정리한 뒤 입력합니다.

4. 코드 예제

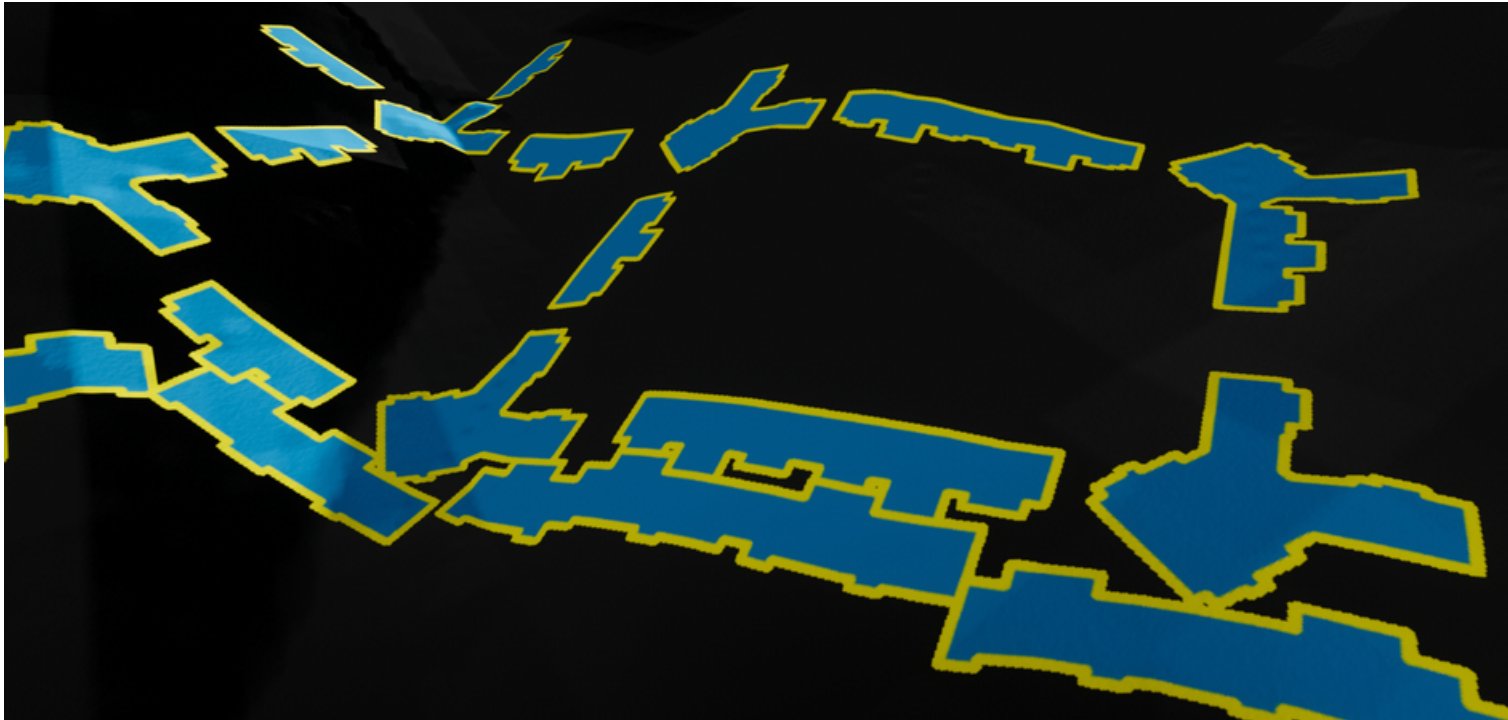
U2dShpLayer를 통해 .shp 데이터를 2D 데이터로 출력하는 예시 코드입니다.

```
1 // 파일 입력 또는 URL 요청으로 읽어들인 shp/dbf/prj 원본 데이터
2 const result = getShpFiles([file1, file2, file3]);
3
4 // shp 데이터 파싱 (.prj / .dbf 가 있으면 같이 파싱)
5 const parser = new GeOnDT.parser.UShpParser();
6 const sourceCRS = result.prj || 'EPSG:5179';
7 const shp = parser.parseShp(result.shp, sourceCRS);
8 const dbf = parser.parseDbf(result.dbf);
9 // 각 feature가 도형 좌표와 속성값을 함께 가진 데이터 묶음 (별도 dbf 파일이 없으면 생략 가능)
10 const featureCollection = parser.combine([shp, dbf]);
11
12 // shp 데이터를 레이어로 GeOnDT for js에 출력
13 const layer = new GeOnDT.image.U2dShpLayer({
14   name: result.name,
15   minlevel: 17,
16   transparent: true,
17   style: {
18     color: $('#stroke').val(),
19     opacity: 1,
20     transparent: false,
21     useStroke: true,
22     stroke: "#fff000"
23   }
24 });
25
26 app.addLayer(layer);
27 app.showLayer(layer.getName(), true);
28
29 // shp Feature Collection 등록
30 layer.addFeatureCollection(featureCollection);
```

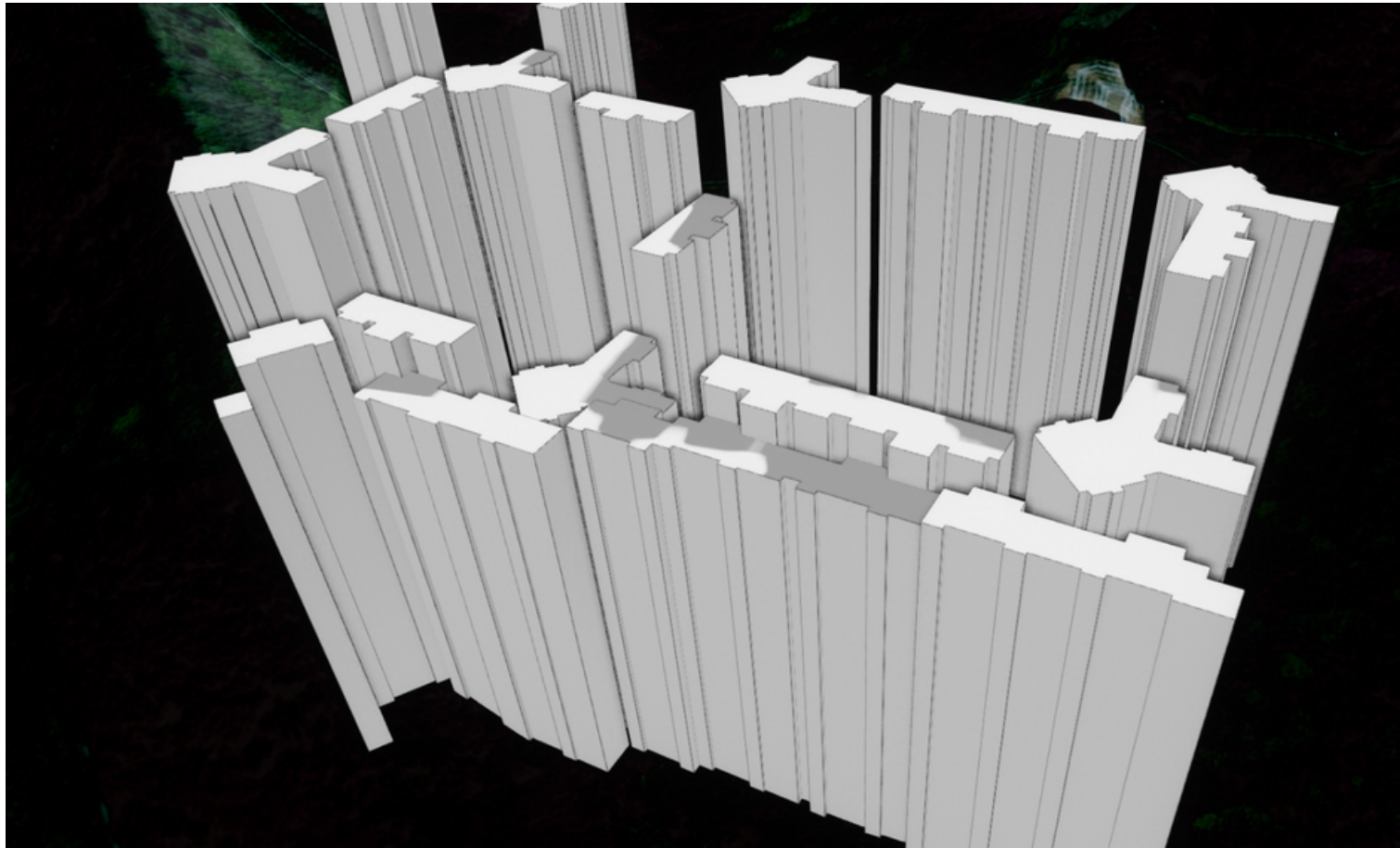
U3dModelShapeLayer 통해 .shp 데이터를 3D 데이터로 출력하는 예시 코드입니다.

```
1 // 파일 입력 또는 URL 요청으로 읽어들인 shp/dbf/prj 원본 데이터
2 const result = getShpFiles([file1, file2, file3]);
3
4 // shp 데이터 파싱 (.prj / .dbf 가 있으면 같이 파싱)
5 const parser = new GeOnDT.parser.UShpParser();
6 const sourceCRS = result.prj || 'EPSG:5179';
7 const shp = parser.parseShp(result.shp, sourceCRS);
8 const dbf = parser.parseDbf(result.dbf);
9 // 각 feature가 도형 좌표와 속성값을 함께 가진 데이터 묶음 (별도 dbf 파일이 없으면 생략 가능)
10 const featureCollection = parser.combine([shp, dbf]);
11
12 // shp 데이터를 레이어로 GeOnDT for js에 출력
13 const layer = new GeOnDT.model.U3dModelShapeLayer({
14   name: result.name,
15   minlevel: 17,
16   transparent: true,
17 });
18
19 app.addLayer(layer);
20 app.showLayer(layer.getName(), true);
21
22 // shp Feature Collection 등록
23 layer.addFeatureCollection(featureCollection);
```

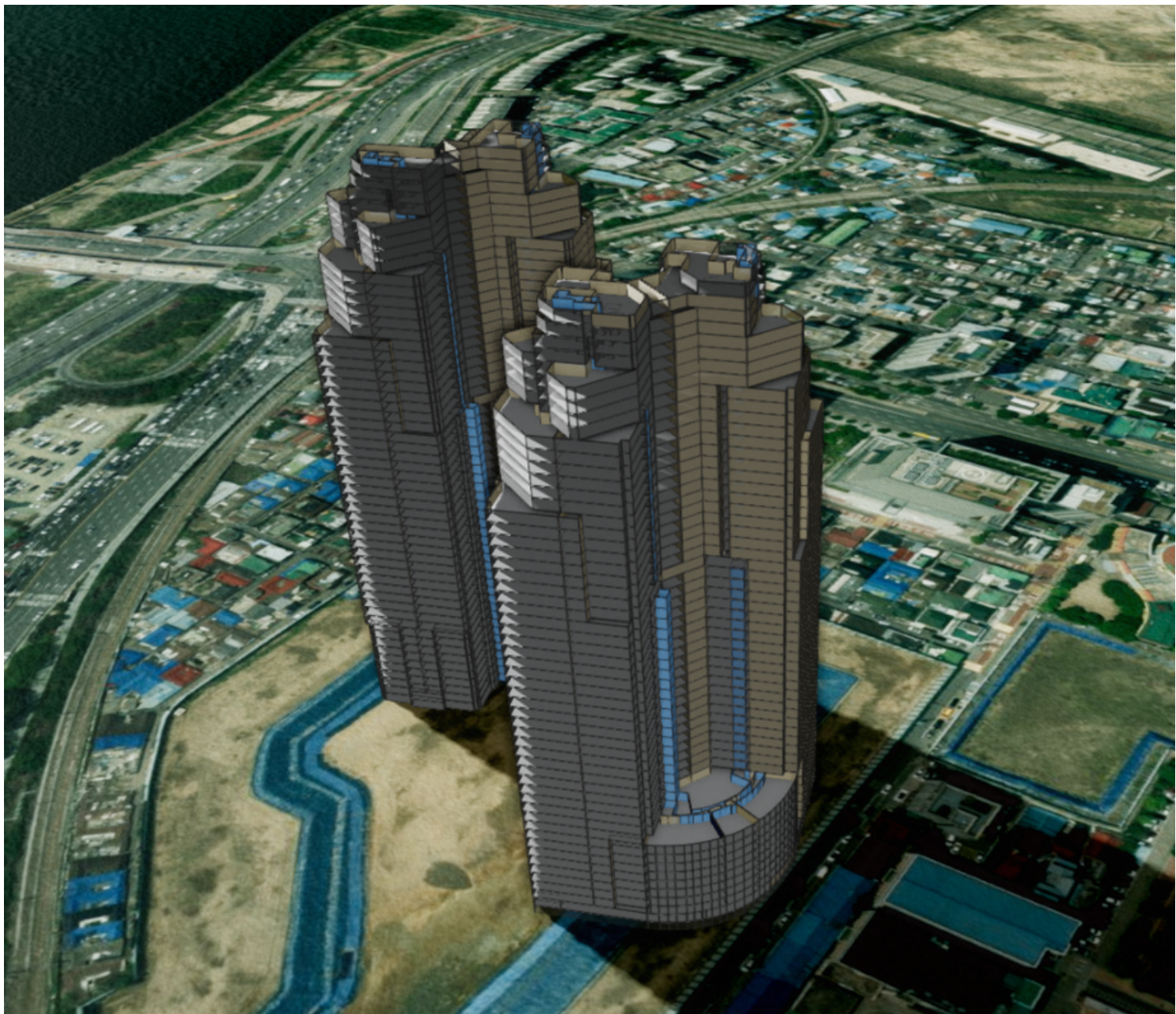
5. 출력 예시



GeOnDT for JS에서 2D로 출력된 .shp 파일 화면



GeOnDT for JS에서 3D 모델로 출력된 .shp 파일 화면



층 정보가 좀 더 잘보이는 예시 화면

6. 권장 사항

필수 조건

- .shp 자체는 좌표계 정보를 포함하지 않으므로, 서비스에서 기대하는 좌표계와 실제 데이터의 좌표계가 일치하는지 .prj 또는 별도 메타데이터로 확인해야 합니다.
- 모든 좌표는 유한한 숫자여야 하며, 폴리곤 링은 외부 링과 내부 링/hole 방향 규칙을 일관되게 유지해야 합니다.
- 파일 헤더의 도형 타입(Shape Type)과 실제 레코드의 도형 타입이 일치해야 하며, 레코드 길이 필드가 실제 데이터 길이와 일치해야 합니다.

권장 조건

- 과도한 좌표 정밀도와 중복 정점을 제거하고, 대용량 .shp는 서비스 축척에 맞게 지오메트리를 단순화합니다.
- 단순화 후 폴리곤 외부 링·hole 관계와 전체 extent를 다시 검증합니다.

DXF

1. 포맷 정의와 공식 규격

DXF는 AutoCAD 도면 교환 형식입니다. 이 프로젝트는 텍스트 DXF를 파싱해 2D 또는 3D 레이어로 변환합니다.

도면 데이터를 GIS 시스템과 교환하기 위한 그룹 코드 기반 도면 형식입니다.

외부 공식 규격

항목	내용
규격명	AutoCAD DXF Reference
발행 기관	Autodesk
조치일	2026-07-31
공식 문서	공식 규격 보기

- 유형: 파일/도면

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

ASCII DXF는 특정 파일 버전 제한 없이 입력할 수 있습니다.

Binary DXF와 파서가 처리하지 않는 엔티티는 지원 포맷으로 변환해야 합니다.

3. 입력 방식

데이터 경로의 URL 또는 로컬 텍스트를 입력합니다.

일반적인 입력은 단일 .dxf 파일입니다.

DXF 형상 자체는 별도의 텍스트나 바이너리 버퍼를 필수로 요구하지 않습니다.

- text: UTF-8 등 브라우저에서 문자열로 읽을 수 있는 ASCII DXF 내용입니다.
- sourceCRS: DXF 좌표에 실제로 적용된 좌표계입니다.

오류 및 누락 시 결과

- Binary DXF 또는 손상된 그룹 코드 구조를 입력하면 파싱이 실패할 수 있습니다.
- 지원하지 않는 엔티티는 표시되지 않을 수 있으며, 잘못된 좌표계는 도형의 위치와 범위를 왜곡합니다.

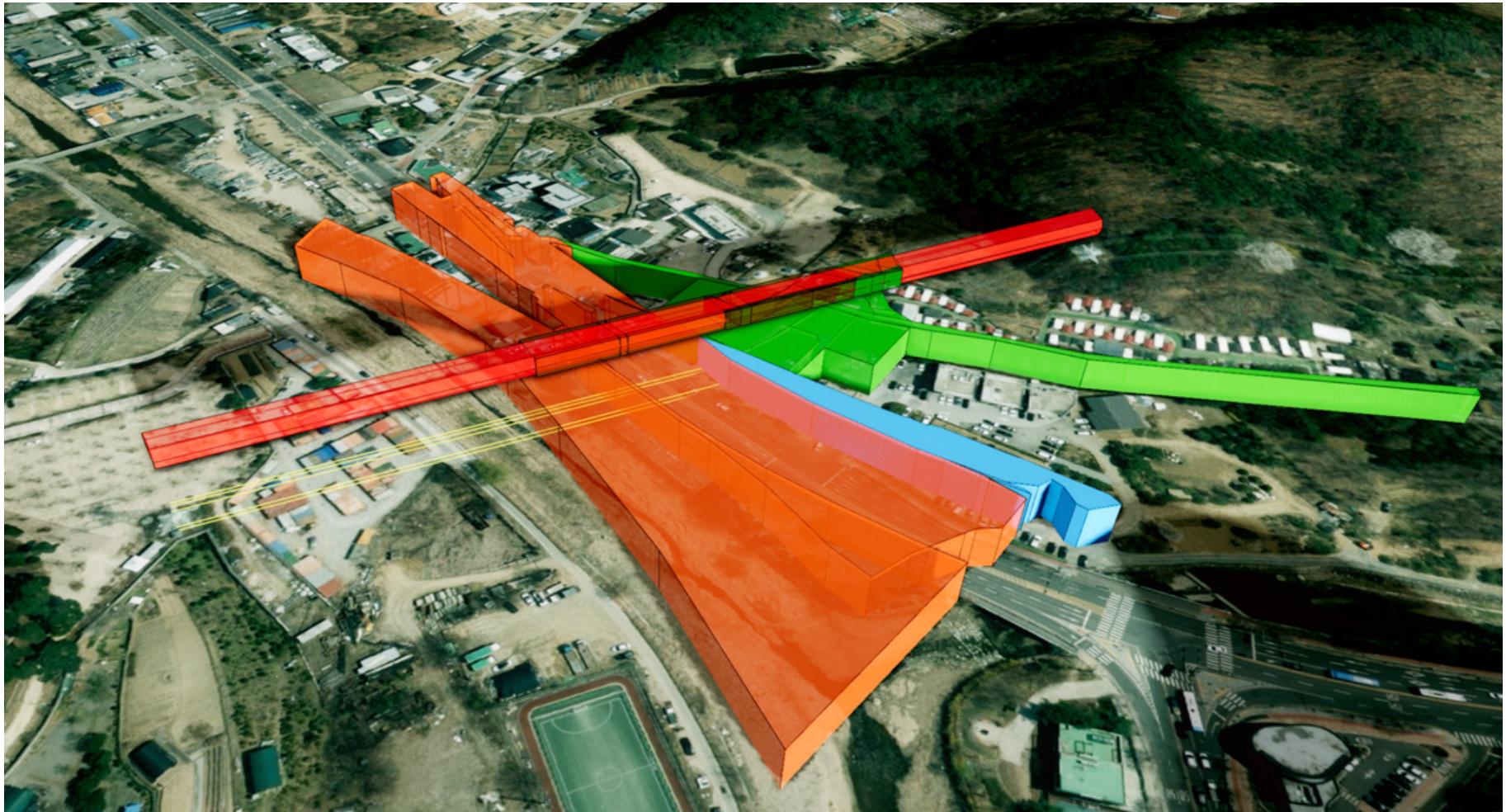
4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 // id가 dxfile인 <input type="file" accept=".dxf"> 요소를 사용합니다.
3 document.querySelector("#dxfile").addEventListener('change', async (event) => {
4     const file = event.target.files[0];
5     if (!file) return;
6
7     let dxf2dLayer;
8     let dxf3dLayer;
9
10    try {
11        const text = await file.text();
12        const dxflinfo = new GeOnDT.parser.UdxParser().parse(text);
13
14        // DXF 2D 레이어를 생성합니다.
15        dxf2dLayer = GeOnDT.image.U2dDxfLayer({
16            name: 'DXF_2D',
17            sourceCRS: 'EPSG:5186'
18        });
19        dxf2dLayer.addDxfInfo(dxflinfo);
20        app.addLayer(dxf2dLayer);
21        app.showLayer(dxf2dLayer.getName(), true);
22
23        // DXF 3D 레이어를 생성합니다.
24        dxf3dLayer = new GeOnDT.model.U3dModelDxfLayer({
25            name: 'DXF_3D',
26            sourceCRS: 'EPSG:5186'
27        });
28        dxf3dLayer.addDxfInfo(dxflinfo);
29        app.addLayer(dxf3dLayer);
30        app.showLayer(dxf3dLayer.getName(), true);
31    } catch (error) {
32        console.error("DXF 로딩 실패", error);
33        if (dxf2dLayer) app.removeLayer(dxf2dLayer.getName());
34        if (dxf3dLayer) app.removeLayer(dxf3dLayer.getName());
35    }
36 });
```

5. 출력 예시



GeOnDT for JS에서 출력된 DXF 2D 결과 화면



GeOnDT for JS에서 출력된 DXF 3D 결과 화면

6. 권장 사항

필수 조건

- 파일 좌표와 레이어 sourceCRS를 일치시킵니다.
- \$INSUNITS는 모델 단위 정보이며 지도 CRS를 결정하지 않습니다.
- X/Y/Z는 유한값이어야 하며 실제 CRS·단위와 sourceCRS를 일치시키고 극단 좌표가 전체 표시 범위를 확대하지 않도록 합니다.

권장 조건

- 불필요한 레이어·블록·세그먼트 제거와 중복 정점을 제거한 ASCII DXF를 사용합니다.
- 곡선 분할 수는 화면 품질과 정점 수의 균형을 맞추고 변환 후 범위를 확인합니다.

KML

1. 포맷 정의와 공식 규격

KML(Keyhole Markup Language)은 지구 브라우저와 지도 서비스에서 지리 피쳐, 스타일, 설명 정보, 뷰포인트, 오버레이 등을 XML로 표현하는 OGC 표준 포맷입니다.

점, 선, 폴리곤, 다중 지오메트리와 스타일 정보를 포함할 수 있으며, 좌표는 기본적으로 경도, 위도, 고도 순서로 작성합니다.

좌표계는 WGS84(EPGS:4326)를 기준으로 하며, 고도 값의 단위는 미터(m)입니다.

외부 공식 규격

항목	내용
규격명	OGC KML Standard
발행 기관	Open Geospatial Consortium
조회일	2026-07-30
공식 문서	공식 규격 보기

- 참고 규격: [OGC KML 2.2](#)
- 유형: 파일/XML 벡터

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

KML 2.2 또는 2.3 버전 사용 가능합니다.

KML 표준의 모든 데이터 타입을 지원하지 않고 일부 타입만 지원합니다.

Geometry/Element Type	지원 여부	비고
Point	지원	단일 점
LineString	지원	선
Polygon	지원	면
MultiPoint	지원	다중 점
MultiGeometry	미지원	여러 지오메트리 혼합
Model	미지원	3D 모델
GroundOverlay / ScreenOverlay / PhotoOverlay	미지원	오버레이
NetworkLink	미지원	외부 KML 연결

KML 스타일은 point/line/polygon 색상·선 두께·아이콘 일부를 지원합니다.

clampToGround, relativeToGround, absolute 등 별도의 고도 모드는 지원하지 않습니다.

3. 입력 방식

KML URL 또는 KML 문자열을 입력합니다.

- baseUrl: KML 파일 URL에서 데이터를 읽어 파싱합니다.
- kmlText: 이미 읽어 둔 KML XML 문자열을 직접 파싱합니다.

KMZ 압축 파일은 현재 직접 입력을 지원하지 않습니다.

파일 구성과 외부 리소스

KMZ 데이터는 압축을 해제해 내부 KML과 외부 리소스(아이콘 이미지 등)를 접근 가능한 경로로 정리한 뒤 입력합니다. 이 경우가 프론트매터의 preConversionRequired(경우에 따라 필요)에 해당하는 대표적인 사례입니다.

또한 원본 KML이 UTF-8이 아닌 인코딩(EUC-KR 등)으로 작성된 경우 UTF-8로 재인코딩한 뒤 입력하는 것을 권장합니다(recommendedInput: UTF-8 KML).

외부 리소스 접근 시 CORS 정책에 따라 로드가 차단될 수 있으므로, 리소스 서버에서 적절한 CORS 헤더를 제공해야 합니다.

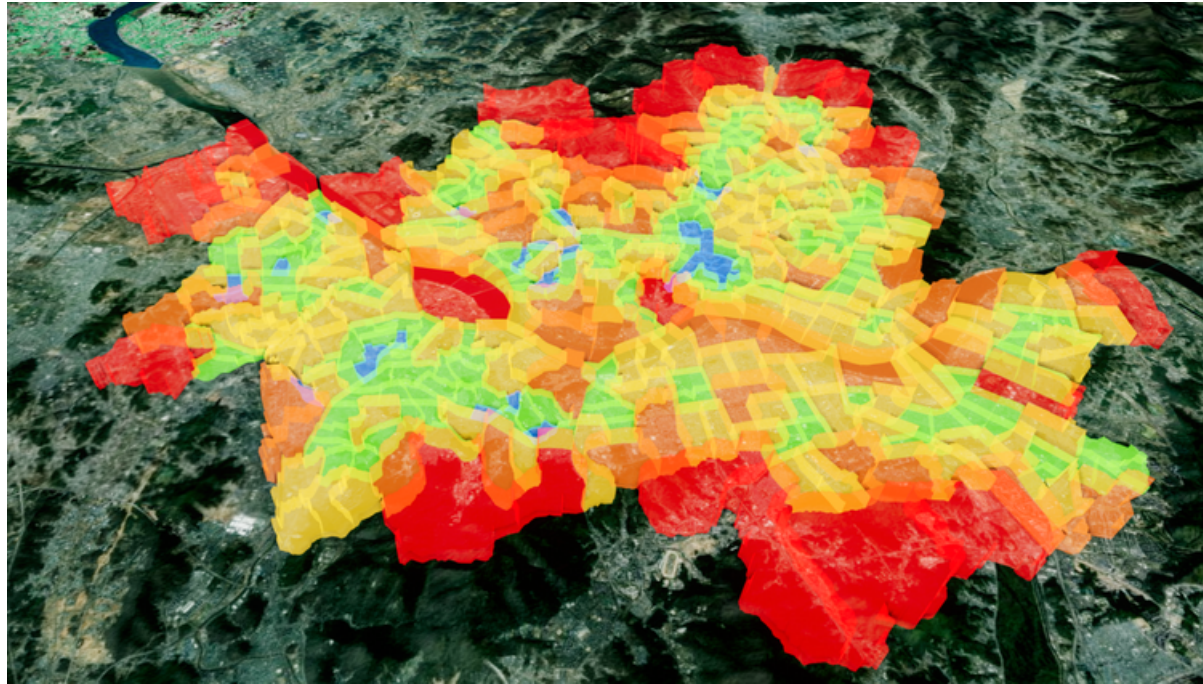
4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layer = new GeOnDT.vector.U3dKmlModelLayer({
3   name: "kml-layer",
4   baseUrl: "http://test-server/data/sample.kml",
5 });
6
7 app.addLayer(layer);
8 app.showLayer(layer.getName(), true);
9
10 // 제거: app.removeLayer(layer.getName());
```

문자열 KML을 직접 전달하는 경우:

```
1 const text = await fetch("/data/sample.kml").then((response) => response.text());
2
3 const layer = new GeOnDT.vector.U3dKmlModelLayer({
4   name: "kml-text-layer",
5   kmlText: text
6 });
7
8 app.addLayer(layer);
9 app.showLayer(layer.getName(), true);
```

5. 출력 예시



GeOnDT for JS에서 출력된 KML 벡터 데이터 결과 화면

6. 권장 사항

필수 조건

- KML 좌표는 경도,위도[,고도] 순서입니다. 위도·경도 순서의 CSV나 GeoJSON 좌표를 그대로 붙여 넣으면 위치가 잘못 표시됩니다.

권장 조건

- 불필요한 Placemark, 중복 Style, 과도한 좌표 정밀도, 중복 정점, 표시하지 않는 속성을 제거합니다.
- 폴리곤과 라인은 서비스 축척에 맞게 단순화하고, 단순화 후 외부 링·hole 관계와 전체 extent를 다시 검증합니다. 아이콘 이미지는 접근 가능한 URL과 적절한 크기로 정리합니다.

GeoJSON

1. 포맷 정의와 공식 규격

GeoJSON은 JSON 객체로 지리 피처, 지오메트리, 속성 정보를 표현하는 IETF 표준 포맷입니다.

일반적으로 FeatureCollection, Feature, Geometry 객체를 사용하며, 좌표는 기본적으로 경도, 위도, 높이 순서로 작성합니다.

좌표계는 WGS84(EPGS:4326)를 기준으로 하며, 고도 값의 단위는 미터(m)입니다.

외부 공식 규격

항목	내용
규격명	The GeoJSON Format
문서 번호	RFC 7946
발행 기관	IETF
조치일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 파일.객체/벡터

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

GeoJSON 공식 표준(RFC 7946)을 기준으로 작성한 데이터를 권장합니다.

GeoJSON 표준의 모든 타입이 아닌 일부 타입만 지원합니다.

Geometry Type	지원 여부	비고
Point	지원	단일 점
LineString	지원	선
Polygon	지원	면
MultiPolygon	지원	다중 면
MultiPoint	미지원	다중 점
MultiLineString	미지원	다중 선
GeometryCollection	미지원	여러 지오메트리 혼합

3. 입력 방식

URL 또는 GeoJSON 파일을 입력합니다.

기본 입력은 UTF-8 .geojson 또는 .json 파일을 권장합니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const text = await fetch("/data/sample.geojson").then((response) => response.text());
3 const geojson = JSON.parse(text);
4
5 const layer = app.getLayerByName("subVectorLayer");
6 await layer.addGeometryAsGeojson(geojson, "EPSG:4326", (geometry, index) => {
7   geometry.feature.set("sourceIndex", index);
8 });
9 app.showLayer(layer.getName(), true);
10
11 // 제거: app.removeLayer(layer.getName());
```

객체를 직접 전달하는 경우:

```
1 const geojson = {
2   type: "FeatureCollection",
3   features: [
4     {
5       type: "Feature",
6       properties: { name: "sample-point" },
7       geometry: {
8         type: "Point",
9         coordinates: [126.9782, 37.5666]
10      }
11     }
12   ]
13 };
14
15 await layer.addGeometryAsGeojson(geojson, "EPSG:4326");
```

5. 출력 예시



GeOnDT for JS에서 출력된 GeoJSON 벡터 데이터 결과 화면

6. 권장 사항

필수 조건

- RFC 7946 GeoJSON 좌표는 경도,위도,[높이] 순서와 WGS84 기준을 사용합니다. 프로젝트 API는 sourceCRS를 별도로 받으므로 실제 데이터 CRS와 반드시 일치시킵니다.
- 모든 좌표는 유한한 숫자여야 합니다. NaN, Infinity, 문자열 숫자, 빈 좌표 배열은 제출 전에 제거하거나 수정합니다.
- FeatureCollection.features, Feature.geometry, geometry.type, coordinates 구조가 GeoJSON 형식에 맞아야 합니다.

권장 조건

- 필요한 properties만 남기고 좌표 정밀도, 정점 수, 피쳐 수를 서비스 목적에 맞게 줄입니다.
- 폴리곤은 링 폐합, hole 관계, 자기 교차 여부를 확인합니다. 단순화 후 전체 extent와 표시 위치를 다시 검증합니다.

TopoJSON

1. 포맷 정의와 공식 규격

TopoJSON은 GeoJSON 지오메트리의 경계를 공유 arc로 표현해 중복 좌표를 줄이는 JSON 기반 위상 벡터 포맷입니다.

일반적으로 Topology, objects, arcs, transform 구조를 사용하며, transform.scale과 transform.translate가 있으면 delta-encoded arc 좌표를 실제 좌표로 복원합니다.

외부 공식 규격

항목	내용
규격명	TopoJSON Specification
문서 번호	RFC 2119
발행 기관	TopoJSON
조치일	2026-07-30
공식 문서	공식 규격 보기

- 유형: 파일·객체/위상 벡터

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

TopoJSON 표준의 전체가 아닌, 일부 타입만 지원합니다.

Geometry Type	지원 여부	비고
Point	지원	단일 점
LineString	지원	선
Polygon	지원	면
MultiLineString	지원	다중 선
MultiPolygon	지원	다중 면
MultiPoint	미지원	다중 점
GeometryCollection	미지원	여러 지오메트리 혼합

3. 입력 방식

URL 또는 TopoJSON 파일을 입력합니다.

기본 입력은 UTF-8 .topojson 또는 .json 파일을 권장합니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const text = await fetch("/data/sample.topojson").then((response) => response.text());
3 const topojson = JSON.parse(text);
4
5 const layer = app.getLayerByName("subVectorLayer");
6 await layer.addGeometryAsTopojson(topojson, "EPSG:3857", (geometry, index) => {
7   geometry.feature.set("sourceIndex", index);
8 });
9 app.showLayer(layer.getName(), true);
10
11 // 제거: app.removeLayer(layer.getName());
```

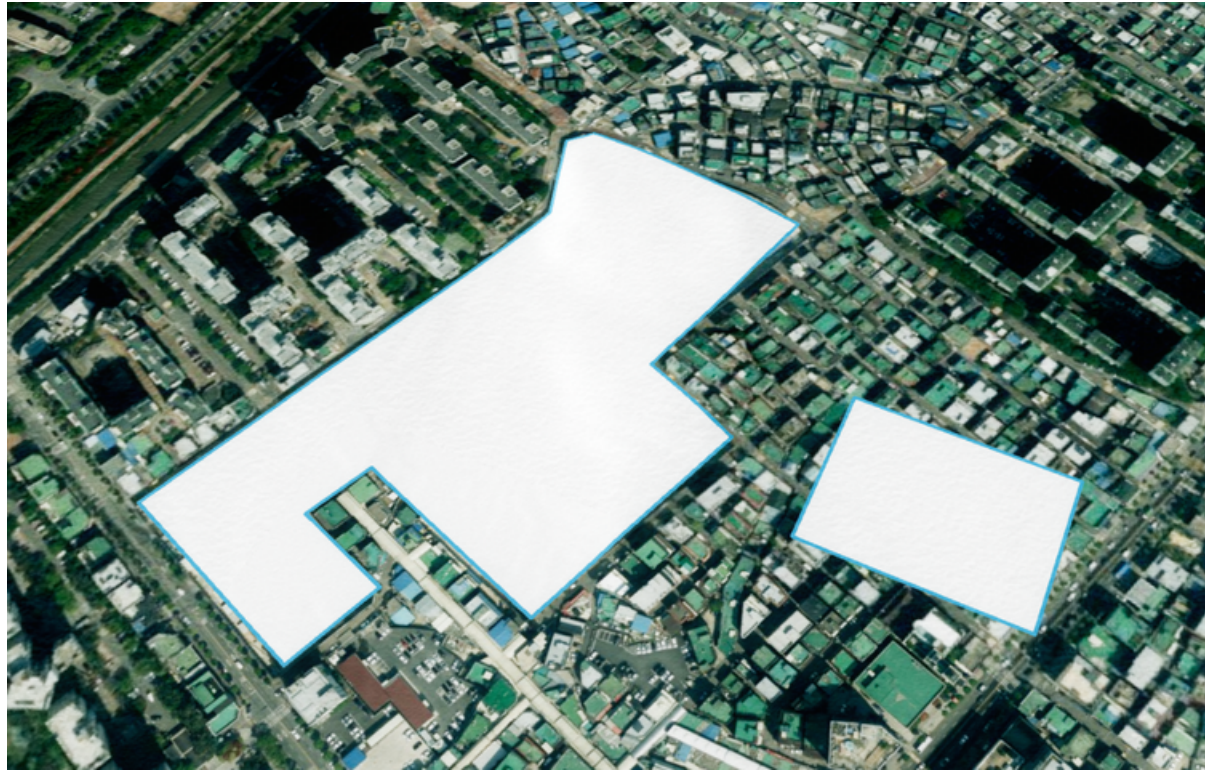
객체를 직접 전달하는 경우:

```
1
2 let topojson = {
3   type: "Topology",
4   transform: {
5     scale: [0.0001, 0.0001],
6     translate: [126.97, 37.56]
7   },
8   objects: {
9     sample: {
10      type: "LineString",
11      arcs: [0],
12      properties: { name: "sample-line" }
13    }
14  },
15   arcs: [
16     [[0, 0], [10, 0], [0, 10]]
17   ]
18 };
19 // 또는 URL로 데이터 Load
20 $.ajax({
21   url: "http://test-server/data/topojsonData.json",
22   method: "GET",
23   dataType: "json",
24   beforeSend: function (xhr) {
```



```
25     xhr.setRequestHeader('User-Authorization', API_KEY);
26   }
27   }).then(function (response) {
28     topojson = response.data ?? response;
29   });
30
31   // topojson 출력 레이어 생성
32   const subVectorLayer = new GeOnDT.vector.U2dVectorShaderLayer({
33     name: 'subVectorLayer',
34     minLevel: 15,
35   });
36   app.addLayer(subVectorLayer);
37   app.showLayer(layer.getName(), true);
38
39   // topojson 결과 출력
40   layer.addGeometryAsTopojson(topojson, "EPSG:4326");
```

5. 출력 예시



GeOnDT for JS에서 출력된 TopoJSON 벡터 데이터 결과 화면

6. 권장 사항

필수 조건

- TopoJSON 좌표는 GeoJSON과 동일하게 경도, 위도 순서를 따릅니다. 위도·경도 순서의 CSV나 다른 포맷의 좌표를 그대로 붙여 넣으면 위치가 잘못 표시됩니다.
- 지도용 TopoJSON은 WGS84 경위도 좌표계(EPSG:4326)를 권장하며, 실제 데이터 좌표계 값이 레이어 생성 시 지정한 sourceCRS 옵션 값과 일치해야 합니다. sourceCRS는 TopoJSON 표준 필드가 아닌 GeoJSON 입력 옵션이므로, 데이터 좌표계와 다르게 지정하면 위치 왜곡이 발생합니다.
- 모든 복원 좌표는 유한한 숫자여야 합니다. 잘못된 scale, translate, delta 좌표는 위치 왜곡이나 표시 실패로 이어질 수 있습니다.

권장 조건

- 공유 arc 구조를 유지하면서 좌표 정밀도와 정점 수를 서비스 목적에 맞게 줄입니다.
- 단순화 후 공유 경계, 폴리곤 링, hole 관계, 전체 extent를 다시 검증합니다.

WKT

1. 포맷 정의와 공식 규격

WKT는 점, 선, 면 등의 지오메트리를 텍스트 문자열로 표현하는 공간 데이터 형식입니다.

이 프로젝트는 WKT 문자열을 파싱하여 2D 벡터 레이어의 지오메트리로 변환합니다.

외부 공식 규격

항목	내용
규격명	Simple Feature Access – Part 1: Common Architecture
발행 기관	Open Geospatial Consortium
조치일	2026-07-31
공식 문서	공식 규격 보기

- 유형: 텍스트/지오메트리

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 일부 지원

OGC Simple Features Access 계열의 일반 WKT 지오메트리를 지원합니다.

Geometry Type	지원 여부	비고
Point	지원	단일 점
LineString	지원	선
Polygon	지원	면
MultiPolygon	지원	다중 면
MultiPoint	미지원	다중 점
MultiLineString	미지원	다중 선
GeometryCollection	미지원	여러 지오메트리 혼합

3. 입력 방식

WKT 문자열과 해당 문자열에 사용된 좌표계를 입력합니다.

WKT는 별도의 파일이 아닌 문자열 형태로 GeOnDT 레이어의 WKT 입력 메서드에 전달합니다.

WKT 지오메트리 자체는 별도의 텍스처, 바이너리 버퍼 또는 외부 리소스를 필수로 요구하지 않습니다.

입력 좌표계는 WKT 문자열에 포함되지 않으므로 sourceCRS 인자로 별도로 전달해야 합니다.

- 첫 번째 인자: 지원되는 Geometry Type의 WKT 문자열입니다.
- 두 번째 인자: WKT 좌표에 실제로 적용된 sourceCRS입니다.
- 세 번째 인자: 생성된 지오메트리에 속성을 설정할 때 사용하는 선택 콜백입니다.

오류 및 누락 시 결과

- WKT 문법이 잘못되었거나 지원하지 않는 Geometry Type을 입력하면 지오메트리가 생성되지 않습니다.
- sourceCRS가 실제 좌표계와 다르면 도형이 잘못된 위치에 표시되거나 화면 범위를 벗어날 수 있습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const vectorLayer = new GeOnDT.vector.U2dVectorShaderLayer({
3   name: 'vectorLayer', // 레이어 이름
4   minLevel: 15 //최소 가시화 타일 레벨
5 });
6 app.addLayer(vectorLayer);
7 app.showLayer(vectorLayer.getName(), true);
8 vectorLayer.addGeometryAsWKT( // WKT 생성
9   "POLYGON ((14177209.001973484 4392175.000611392,14177159.001973478 4391833.000611345," +
10   "14177465.001973521 4391764.000611337,14177395.00197351 4391952.000611366," +
11   "14177425.001973512 4392047.000611377,14177572.001973534 4392089.000611383," +
12   "14177391.001973508 4392213.000611402,14177209.001973484 4392175.000611392))", // WKT 문자열
13   'EPSG:3857', (geometry) => geometry.setMeta({ source: 'meta 데이터 정보' })
14 ).then(() => {
15   app.fitLayerExtent(vectorLayer.getName());
16 }).catch((error) => {
17   console.error('WKT 로딩 실패', error);
18   app.removeLayer(vectorLayer.getName());
19 });
20 // 제거: app.removeLayer(vectorLayer.getName());
```

5. 출력 예시



GeOnDT for JS에서 출력된 WKT 결과 화면

6. 권장 사항

필수 조건

- WKT 좌표와 입력 인자로 전달하는 좌표계를 일치시킵니다.
- WKT 문자열 자체에는 일반적으로 좌표계 정보가 포함되지 않으므로 실제 좌표의 기준과 sourceCRS를 일치시켜야 합니다.

권장 조건

- 불필요한 소수 자릿수와 중복 정점을 제거한 WKT 문자열을 사용합니다.
- 선과 폴리곤은 화면 품질을 유지하는 범위에서 정점을 단순화하고 변환 후 표시 범위를 확인합니다.

WKB

1. 포맷 정의와 공식 규격

WKB(Well Known Binary)는 OGC Simple Features 지오메트리의 바이너리 표현입니다.

점, 선, 면 같은 공간(지리) 데이터를 컴퓨터가 읽기 좋은 바이너리(이진) 스트림으로 표현하는 표준 형식입니다

외부 공식 규격

항목	내용
규격명	Simple Feature Access – Part 1: Common Architecture
발행 기관	Open Geospatial Consortium
조치일	2026-07-31
공식 문서	공식 규격 보기

- 유형: 바이너리/지오메트리

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 공개 API 없음

WKB는 GeOnDT for JS에서 직접 사용할 수 없으며, WKT 또는 GeoJSON과 같이 공개 입력 경로가 확인된 형식으로 변환해야 합니다.

직접 지원하는 WKB 버전 또는 규격 범위가 없습니다.

3. 입력 방식

WKB를 GeOnDT API에 직접 입력할 수 없습니다.

외부 라이브러리, 서버 또는 데이터베이스에서 WKB를 WKT나 GeoJSON으로 변환한 후 해당 결과를 GeOnDT 레이어에 입력합니다.

WKB는 일반적으로 독립 파일보다 다음 형태로 전달됩니다.

- ArrayBuffer / Uint8Array
- 데이터베이스의 바이너리 컬럼
- 16진수 Hex 문자열
- Base64 문자열
- API 응답에 포함된 바이너리 데이터

GeOnDT for JS에서는 위 입력을 직접 처리하지 않으므로 별도의 외부 변환기가 필요합니다.

4. 코드 예제

1	해당 사항 없음
---	----------

5. 출력 예시

해당 사항 없음

6. 권장 사항

권장 조건

- WKB 데이터만으로 실제 CRS를 결정할 수 없으며, 데이터 제공처의 메타데이터 또는 별도 설정을 통해 좌표계를 확인해야 합니다.
- GeOnDT가 직접 지원하는 GeoJSON/WKT 등으로 변환 후 검증합니다.

U3F

1. 포맷 정의와 공식 규격

U3F는 넓은 지역의 대규모 3D 형상 그룹을 단계적으로 표시하기 위한 GeOnDT 자체 바이너리 타일 포맷입니다.

SHP-OBJ-3DS 등의 원본 데이터는 GeOnDT Builder에서 U3F용 구조로 정규화되고, EPSG:3857 격자 체계에 따라 타일로 분할됩니다. 각 타일에는 해당 영역의 형상과 표시 정보가 저장되며, GeOnDT for JS는 현재 화면에 필요한 타일만 불러옵니다.

U3F는 개별 .u3f 파일 하나가 아니라 루트 메타데이터, 타일 정보, 다수의 U3F 콘텐츠와 텍스처로 구성된 폴더 전체를 사용합니다.

- 공식 규격/참고: GeOnDT 자체 규격이며 외부 공개 규격은 없습니다. 제작 세부 사항은 GeOnDT Builder 매뉴얼을 확인합니다.
- 유형: 파일 묶음/대규모 3D 형상 타일

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

GeOnDT for JS는 U3F 콘텐츠 버전 2.2, 2.3, 2.4, 2.5를 직접 지원합니다.

항목	지원 여부	설명
U3F 콘텐츠 2.2~2.5	지원	각 타일의 형상, 객체 식별 정보와 텍스처 참조를 읽습니다.
그 외 U3F 콘텐츠 버전	미지원	해당 버전의 타일은 표시되지 않습니다.
EPSG:3857 타일 계층	지원	레벨/X/Y 격자 경로에 따라 필요한 타일을 요청합니다.
gzip 압축 U3F	지원	Builder 결과가 .u3f.gz일 때 compressmodel: true로 입력합니다.
U3F 패키지	지원	Builder 결과가 .u3fpackage일 때 makeU3FPackage: 1로 입력합니다.
Builder가 생성한 텍스처	지원	외부 텍스처와 U3F에 포함된 미리보기 텍스처를 사용할 수 있습니다.

루트 U3G와 타일 정보 U3M의 버전 값은 U3F 콘텐츠 버전과 별개입니다. Builder가 생성한 파일 구성을 그대로 사용합니다.

3. 입력 방식

입력 자료를 다운로드 받을 수 있는 URL을 관련 레이어 또는 API에 입력합니다.

관련 레이어 및 API:

- GeOnDT.model.U3dModelU3FLayer
- app.create3DFModelLayer()

파일 구성과 외부 리소스

baseurl: "/data/u3f/seoul-buildings", basename: "seoul-buildings"를 사용하는 타일셋은 다음과 같이 구성됩니다.

```
1 /data/u3f/seoul-buildings/
2 |— seoul-buildings.u3g
3 |— {level}/{x}/{y}/
4 |   |— info.u3m          # 패키지 JSON을 사용하지 않는 경우
5 |   |— 0.u3f
6 |   |— 1.u3f
7 |   |— 0.jpg, 0_1.jpg, ... # 외부 텍스처가 있는 경우
```

위 구조는 일반 결과의 예입니다. 패키지 인덱스용 JSON을 사용하는 결과에서는 info.u3m 대신 JSON에서 타일 정보를 읽을 수 있습니다. gzip 결과는 .u3f.gz, 패키지 결과는 .u3fpackage 또는 .u3fpackage.gz를 사용합니다. 서로 임의로 섞거나 파일명을 바꾸지 말고 Builder가 생성한 폴더 구조 전체를 그대로 배포합니다.

모든 메타데이터, 타일과 텍스처 URL의 대소문자와 상대 경로를 서버 구성과 일치시킵니다. 교차 출처로 배포할 때는 CORS를 허용하거나 올바른 프록시 설정을 사용합니다.

오류 및 누락 시 결과

- baseurl 또는 basename이 없거나 루트 .u3g 요청이 실패하면 레이어를 초기화할 수 없습니다.
- minlevel, maxlevel, 압축 또는 패키지 옵션이 실제 Builder 결과와 다르면 존재하지 않는 타일을 요청하여 화면이 비어 보일 수 있습니다.
- 사용 중인 구성에 필요한 info.u3m 또는 패키지 인덱스 JSON, 개별 U3F, 패키지, 텍스처가 누락되거나 손상되면 해당 영역의 형상이 표시되지 않거나 텍스처가 의도와 다르게 보일 수 있습니다.
- 지원하지 않는 U3F 콘텐츠 버전은 해당 타일에 형상을 표시하지 않습니다.
- layer.ready 이후의 개별 타일 오류는 초기화 오류 처리로 전달되지 않을 수 있습니다. 일부 영역이 비어 보이면 브라우저의 네트워크 요청과 오류 메시지도 확인합니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerName = "Seoul_Buildings";
3
4 // U3dModelU3FLayer를 생성합니다.
5 const layer = app.create3DFModelLayer({
6   name: layerName,
7   baseUrl: "/data/u3f/seoul-buildings",
8   basename: "seoul-buildings",
9   ext: ".u3f",
10
11   // Builder가 발행한 타일 레벨과 같은 값을 입력합니다.
12   minlevel: 17,
13   maxlevel: 19,
14   useproxy: false
15 });
16
17 if (layer === false) {
18   console.error("U3F 레이어를 생성하지 못했습니다. 레이어 이름과 입력값을 확인하세요.");
19 } else {
20   layer.ready.then(
21     () => {
22       app.showLayer(layerName, true);
23     },
24     (error) => {
25       console.error("U3F 레이어 초기화 실패", error);
26       app.removeLayer(layerName);
27     }
28   );
29 }
30
31 // 숨기기: app.showLayer(layerName, false);
32 // 제거하기: app.removeLayer(layerName);
```

5. 출력 예시

U3F 타일셋은 넓은 지역의 여러 3D 형상을 한 레이어로 표시합니다. 카메라가 이동하거나 확대되면 현재 위치와 타일 레벨에 맞는 형상과 텍스처가 단계적으로 표시됩니다.



6. 권장 사항

필수 조건

- 개별 .u3f 파일이 아니라 GeOnDT Builder가 생성한 .u3g, 타일 메타데이터, U3F 콘텐츠와 텍스처를 포함한 폴더 전체를 사용합니다.
- U3F 콘텐츠 버전은 2.2~2.5로 생성합니다.
- EPSG:3857 레벨/X/Y 타일 폴더, 좌표.범위 메타데이터와 파일명을 임의로 수정하지 않습니다.
- baseUrl, basename, minlevel, maxlevel을 실제 Builder 발행 결과와 일치시킵니다. 압축 또는 패키지 결과라면 compressmodel, makeU3FPackage도 일치시킵니다.
- 모든 메타데이터.타일.텍스처를 접근 가능한 URL로 배포하고 파일명 대소문자와 상대 경로를 서버 구성과 일치시킵니다. 교차 출처에서는 CORS 또는 프록시를 설정합니다.

권장 조건

- SHP-OBJ-3DS 등 원본 데이터의 좌표계와 단위를 확인하고, 3D 모델은 축과 텍스처 참조도 확인한 뒤 GeOnDT Builder로 변환합니다.
- 원본의 중복 형상, 퇴화 면, 사용하지 않는 객체.속성.재질.텍스처를 정리합니다.
- 타일 범위와 실제 형상이 일치하고 상세도별 데이터량이 한 타일에 과도하게 몰리지 않도록 Builder 설정을 조정합니다.
- 전체 데이터를 발행하기 전에 대표 영역을 먼저 변환하여 위치, 높이, 형상, 텍스처와 타일 전환을 확인합니다.
- 입력 포맷별 지원 버전, 좌표 설정, 정규화, 타일 레벨, 분할.병합, 텍스처.압축.패키지 옵션의 상세값은 GeOnDT Builder 매뉴얼을 따릅니다.

선택 조건

- gzip 결과를 사용할 때만 compressmodel: true를 설정합니다.
- 패키지 결과를 사용할 때만 makeU3FPackage: 1을 설정합니다. .u3fpackage는 ZIP 파일이 아니라 U3F 전용 패키지입니다.
- 텍스처 품질을 제한할 때 textureLevel: "low" 또는 "high"를 사용할 수 있습니다.
- Builder 결과가 Y 타일 주소를 반대로 사용한 경우에만 reverseY: true를 설정합니다. 이 옵션은 형상의 좌표축을 보정하지 않습니다.

I3F

1. 포맷 정의와 공식 규격

I3F는 배관·터널처럼 넓은 영역에 길게 분포한 3D 형상 그룹을 단계적으로 표시하기 위한 GeOnDT 자체 바이너리 타일 포맷입니다. OGC 3D Tiles나 I3S와는 다른 형식입니다.

SHP-OBJ-3DS 등의 원본 데이터는 GeOnDT Builder에서 I3F용 구조로 정규화되고, EPSG:3857 격자 체계에 따라 타일로 분할됩니다. tilemap.json에는 전체 공간 범위, 타일 트리과 콘텐츠 경로가 저장되며, 실제 형상은 여러 .i3f 또는 압축된 .i3z 파일에 저장됩니다.

I3F는 격자에 저장되지만 현재 지도 레벨의 쿼드 타일만으로 출력 대상을 결정하지 않습니다. 카메라 주변의 거리와 화면에 보이는 범위를 기준으로 트리에 등록된 공간 타일을 불러오고, 범위를 벗어난 타일은 정리합니다. 이 동작 개념은 3D Tiles와 유사하지만, geometricError에 따라 상세도를 전환하는 3D Tiles 방식과 동일하지는 않습니다.

예를 들어 여러 격자를 길게 통과하는 배관이나 터널도 하나의 공간 트리에서 관리할 수 있으므로, 일반적인 격자 단위 출력만으로 다루기 어려운 선형 시설물을 표시하는 데 적합합니다.

- 공식 규격/참고: GeOnDT 자체 규격이며 외부 공개 규격은 없습니다. 제작과 발행의 세부 설정은 GeOnDT Builder 매뉴얼을 확인합니다.
- 유형: 파일 묶음/대규모 3D 형상 타일

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

I3F 파일에는 major/minor 버전 값이 포함되지만, 사용자가 선택할 수 있는 공개 버전별 호환표는 제공되지 않습니다. 따라서 지원 버전은 tilemap.json의 version 속성으로 구분하며, 사용 중인 GeOnDT for JS와 함께 제공되는 GeOnDT Builder에서 생성한 결과를 권장합니다.

항목	지원 여부	설명
Builder I3F 타일셋	지원	tilemap.json의 공간 트리와 현재 화면에 필요한 콘텐츠를 읽습니다.
I3F 바이너리 콘텐츠	지원	정점, 면 인덱스, UV, 법선, 기본 색상, 외부 텍스처 참조와 문자열 속성을 읽습니다.
I3Z 압축 콘텐츠	지원	소문자 .i3z 확장자의 압축 콘텐츠를 해제하여 I3F와 같은 방식으로 표시합니다.
EPSG:3857 타일 체계	지원	Builder가 생성한 EPSG:3857 범위와 공간 트리를 기준으로 타일을 배치합니다.
객체별 회전·크기	일부 지원	파일에 값이 있더라도 회전과 크기는 화면에 자동으로 적용되지 않습니다. 원하는 상태를 정점에 반영한 뒤 Builder로 변환합니다.
재질 상세 속성	일부 지원	기본 색상과 외부 텍스처를 중심으로 표시하며, 파일의 모든 재질 계수를 그대로 재현하지는 않습니다.

3. 입력 방식

입력 자료를 다운로드 받을 수 있는 URL을 관련 레이어에 입력합니다. I3F는 폴더나 개별 .i3f 파일이 아니라 tilemap.json의 전체 URL을 사용합니다.

관련 레이어:

- GeOnDT.model.U3dModelI3FLayer

파일 구성과 외부 리소스

I3F 타일셋은 tilemap.json과 이 파일이 참조하는 I3F/I3Z 콘텐츠 및 텍스처로 구성됩니다.

```
1 /data/i3f/underground/  
2 |— tilemap.json  
3 |— tiles/  
4 |   |— tile-a.i3f  
5 |   |— tile-b.i3z  
6 |— textures/  
7   |— material.png
```

위 폴더명과 파일명은 설명을 위한 예시입니다. I3F는 고정된 XYZ 폴더 구조를 요구하지 않으며, 실제 콘텐츠와 재질 파일은 tilemap.json에 기록된 상대 경로를 따릅니다.

tilemap.json이 위치해 있는 폴더를 기준으로 상대 경로가 입력됩니다. Builder가 생성한 폴더 구조와 파일명 대소문자를 그대로 유지하고, 타일셋을 이동할 때는 참조된 콘텐츠와 텍스처도 함께 이동합니다.

오류 및 누락 시 결과

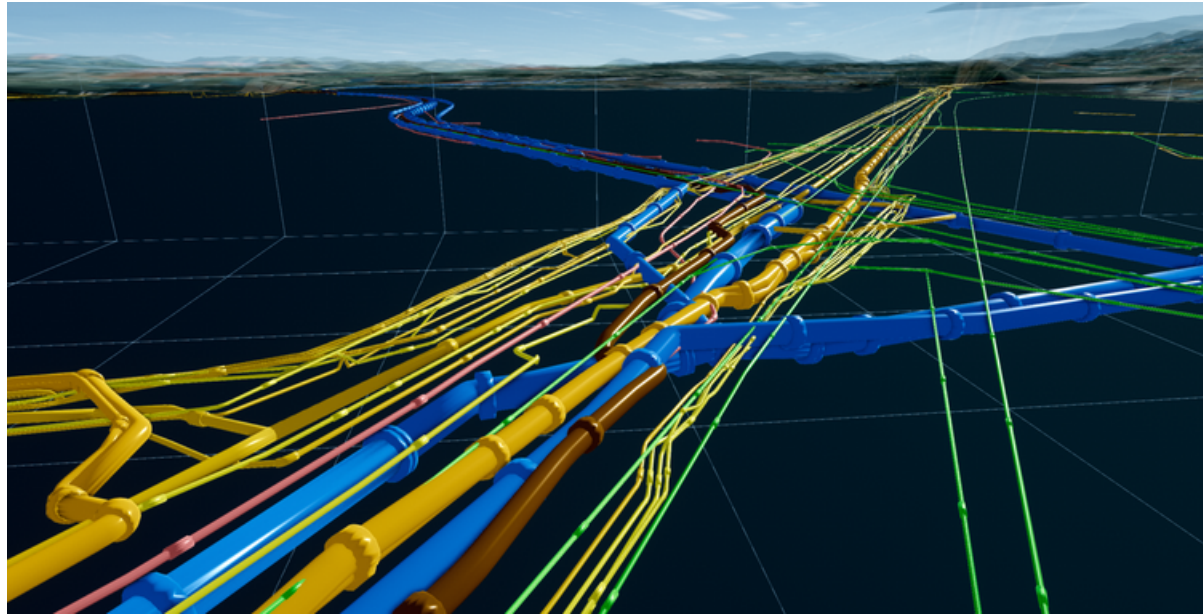
- baseUrl이 없거나 tilemap.json을 불러오지 못하면 레이어를 초기화할 수 없습니다.
- tilemap.json의 전체 범위, 좌표 변환 범위, 트리 또는 공통 재질 목록이 누락되거나 올바르지 않으면 초기화가 끝나지 않거나 화면이 비어 보일 수 있습니다.
- 개별 I3F/I3Z 파일이 없거나 손상되면 해당 공간의 형상이 표시되지 않을 수 있습니다.
- 타일 범위가 실제 형상과 다르면 형상이 다른 위치에 표시되거나, 카메라가 가까이 있어도 일찍 사라질 수 있습니다.
- 텍스처의 상대 경로, 파일명 대소문자 또는 CORS 설정이 올바르지 않으면 재질이 다르게 보이거나 해당 형상이 표시되지 않을 수 있습니다.
- 개별 콘텐츠와 텍스처 오류는 layer.ready의 초기화 오류로 전달되지 않을 수 있습니다. 일부 영역이 비어 보이면 브라우저의 네트워크 요청에서 tilemap.json, I3F/I3Z와 텍스처의 404 및 CORS 오류를 함께 확인합니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerName = "Underground_Pipes";
3
4 // I3F 전용 레이어를 생성합니다.
5 const layer = new GeOnDT.model.U3dModelI3FLayer({
6   name: layerName,
7
8   // 폴더가 아니라 Builder가 생성한 tilemap.json의 전체 URL을 입력합니다.
9   baseurl: "/data/i3f/underground/tilemap.json",
10  useproxy: false,
11
12  // 220은 공식 예제의 시작값입니다. 자료 범위와 서비스 성능에 맞게 조정합니다.
13  limitdistance: 220,
14  modeldetail: "high"
15 });
16
17 // I3F 레이어는 직접 생성하므로 앱에 별도로 등록합니다.
18 app.addLayer(layer);
19
20 layer.ready.then(
21   () => {
22     // 이 시점에는 타일 트리가 준비되며, 개별 콘텐츠는 카메라 위치에 따라 불러옵니다.
23     app.showLayer(layerName, true);
24   },
25   (error) => {
26     console.error("I3F 초기화 실패", error);
27     app.removeLayer(layerName);
28   }
29 );
30
31 // 숨기기: app.showLayer(layerName, false);
32 // 제거하기: app.removeLayer(layerName);
```

5. 출력 예시

I3F 타일셋은 배관·터널처럼 여러 격자에 걸쳐 이어지는 3D 형상을 한 레이어로 표시합니다. 카메라가 이동하면 설정된 거리와 현재 화면 범위에 들어온 타일을 추가로 불러오고, 범위를 벗어난 타일은 정리합니다.



6. 권장 사항

필수 조건

- 개별 .i3f 파일이 아니라 GeOnDT Builder가 생성한 tilemap.json, I3F/I3Z 콘텐츠와 텍스처를 포함한 타일셋 전체를 사용합니다.
- baseUrl에는 tilemap.json의 전체 URL을 입력합니다.
- 원본 데이터의 실제 좌표계와 단위를 Builder에 정확히 입력합니다. 출력 I3F의 격자가 EPSG:3857이라는 이유로 원본 좌표계까지 EPSG:3857로 지정하지 않습니다.
- Builder가 생성한 전체 범위, 좌표 변환 범위, 타일 트리, 공통 재질 목록과 콘텐츠 경로를 임의로 수정하지 않습니다.
- 모든 메타데이터·콘텐츠·텍스처를 접근 가능한 URL로 배포하고 파일명 대소문자와 상대 경로를 서버 구성과 일치시킵니다. 교차 출처에서는 CORS를 허용합니다.

권장 조건

- 원본의 위치, 높이 단위와 상향축을 확인하고, 원하는 회전과 크기를 정점에 반영한 뒤 GeOnDT Builder로 변환합니다.

- 형상 중심점과 공간 범위가 실제 형상 가까이에 있도록 구성하고, 정점 좌표와 공간 범위를 서로 일치시킵니다. 표시할 때 공간 범위를 기준으로 형상 중심이 계산되므로, 형상과 멀리 떨어진 범위는 피합니다.
- 타일의 공간 범위가 실제 콘텐츠 전체를 포함하도록 합니다. 범위가 너무 크면 불필요한 타일까지 불러오고, 너무 작으면 형상이 일찍 사라질 수 있습니다.
- 배관·터널의 연결이 타일 경계에서 끊기지 않는지 확인하고, 긴 형상이 너무 많은 작은 타일로 잘리거나 하나의 타일에 과도하게 몰리지 않도록 트리와 분할 결과를 조정합니다.
- 올바른 조명 표현이 필요하다면 각 정점의 법선을 포함합니다. 법선이 없으면 모든 정점에 위쪽 방향의 기본 법선이 적용되어 곡면이 평평하거나 부자연스럽게 보일 수 있습니다.
- 중복 형상, 퇴화 면, 사용하지 않는 객체·속성·재질·텍스처를 정리하고 텍스처 크기를 서비스 환경에 맞게 조정합니다.
- 전체 데이터를 발행하기 전에 대표 구간을 먼저 확인하고, 카메라 이동 시 타일이 자연스럽게 나타나고 사라지는지와 네트워크 요청량·메모리 사용량을 함께 측정합니다.
- 입력 포맷별 지원 범위, 정규화, 좌표 변환, 타일 레벨과 트리 구성의 상세 설정은 GeOnDT Builder 매뉴얼을 따릅니다.

선택 조건

- limitdistance는 공식 예제값 220에서 시작하여 실제 자료 분포와 화면 용도에 맞게 조정할 수 있습니다. 값을 크게 하면 넓은 범위를 표시하지만 요청량과 메모리 사용량도 늘어날 수 있습니다.
- 전송량을 줄이기 위해 Builder가 생성한 .i3z 압축 콘텐츠를 사용할 수 있습니다. 압축 생성과 발행 방법은 사용하는 Builder 버전의 매뉴얼을 확인합니다.

UMF

1. 포맷 정의와 공식 규격

UMF는 넓은 지역의 지면 높이를 단계적으로 표시하기 위한 GeOnDT 자체 바이너리 고도 타일 포맷입니다.

IMG-GeoTIFF 등에 저장된 고도정보는 GeOnDT Builder에서 EPSG:3857 기반의 레벨/X/Y 구조로 타일링되어 UMF 타일셋으로 생성됩니다. GeOnDT for JS는 카메라 위치와 확대 수준에 필요한 UMF 타일의 고도값을 지면에 적용합니다.

UMF는 색상을 표시하는 이미지 타일이나 독립된 3D 모델이 아닙니다. 개별 .umf 파일 하나가 아니라 타일셋 메타데이터와 여러 레벨의 고도 타일로 구성된 폴더 전체를 사용합니다.

- 공식 규격/참고: GeOnDT 자체 규격이며 외부 공개 규격은 없습니다. 제작 세부 사항은 GeOnDT Builder 매뉴얼을 확인합니다.
- 유형: 파일 묶음/지형 고도 타일

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

UMF는 별도의 공개 버전 번호를 사용하지 않는 GeOnDT 내부 규격입니다. GeOnDT for JS는 **2바이트 Int16 또는 4바이트 Float32 고도 배열로 구성된 내부 규격을 부분 지원합니다.**

항목	지원 여부	설명
Builder UMF 타일셋	지원	타일의 고도값을 읽어 지면 높이에 적용합니다.
EPSG:3857 레벨/X/Y 타일 계층	지원	현재 화면에 필요한 위치와 레벨의 타일을 요청합니다.
tilemapresource.xml 메타데이터	지원	타일 크기, 고도 샘플 크기, 스케일-오프셋, 압축 여부와 공간 범위를 읽습니다.
4바이트 Float32-2바이트 Int16 고도 배열	지원	메타데이터에 기록된 고도 샘플 크기와 실제 UMF 데이터가 일치해야 합니다.
gzip 압축 UMF	지원	Builder 메타데이터에 압축 사용이 기록된 타일셋을 읽습니다.
IMG-GeoTIFF 원본 직접 입력	미지원	원본 고도정보는 GeOnDT Builder에서 UMF 타일셋으로 변환한 뒤 입력합니다.

3. 입력 방식

입력 자료를 다운로드 받을 수 있는 URL을 관련 레이어 또는 API에 입력합니다. UMF는 개별 .umf 파일이 아니라 타일셋의 루트 폴더 URL을 사용합니다.

관련 레이어 및 API:

- GeOnDT.terrain.U3dHeightXYZLayer
- app.createHeightXYZLayer()

파일 구성과 외부 리소스

baseUrl: "/data/umf/korea-terrain"을 사용하는 기본 타일셋은 다음과 같이 구성됩니다.

```
1 /data/umf/korea-terrain/  
2 |— tilemapresource.xml  
3 |— 9/  
4 |   |— {x}/  
5 |       |— {y}.umf  
6 |— 10/  
7     |— {x}/  
8         |— {y}.umf
```

각 고도 타일은 {baseUrl}/{level}/{x}/{y}.umf 경로로 요청합니다. 위 구조는 예시이며 실제 레벨과 X/Y 폴더는 Builder 결과에 따라 달라집니다. 압축 결과는 .umf.gz를 사용하며, tilemapresource.xml의 압축 정보도 함께 유지해야 합니다. 파일명이나 폴더 구조를 임의로 바꾸지 말고 Builder가 생성한 타일셋 전체를 그대로 배포합니다.

모든 메타데이터와 타일 URL의 대소문자 및 상대 경로를 서버 구성과 일치시킵니다. 교차 출처로 배포할 때는 CORS를 허용하거나 올바른 프록시 설정을 사용합니다.

오류 및 누락 시 결과

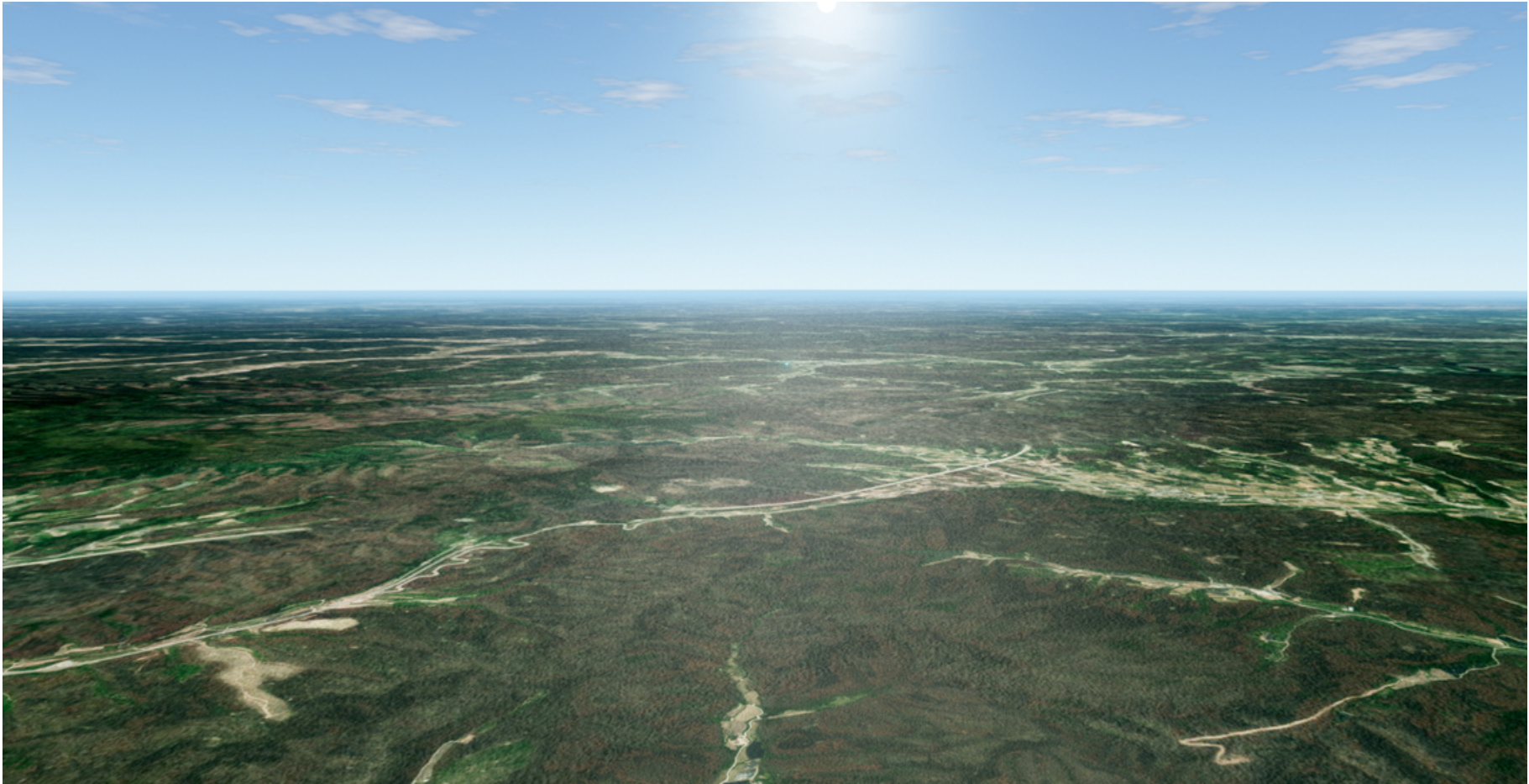
- 기본 구성에서 tilemapresource.xml이 없거나 요청에 실패하면 레이어 초기화가 완료되지 않은 상태로 기다릴 수 있습니다. 파일 내용이 올바르지 않으면 지형이 표시되지 않거나 높이가 잘못 적용될 수 있습니다.
- minlevel, maxlevel, reverseY 또는 레벨/X/Y 경로가 실제 Builder 결과와 다르면 존재하지 않거나 다른 위치의 타일을 요청할 수 있습니다.
- 개별 UMF 타일을 불러오지 못하면 가능한 경우 상위 레벨의 고도를 대신 사용합니다. 상위 데이터도 없으면 해당 영역에 고도가 적용되지 않을 수 있습니다.
- 타일 크기, 고도 샘플 크기, 스케일오프셋 또는 공간 범위가 실제 UMF 데이터와 다르면 지형이 평평하거나 과도하게 높아지고, 표면이 불규칙하게 깨지거나 타일 경계에 단차가 생길 수 있습니다.
- 메타데이터 및 개별 타일 오류가 초기화 오류 처리로 모두 전달되는 것은 아닙니다. 초기화가 끝나지 않거나 일부 영역의 높이가 잘못 보이면 브라우저의 네트워크 요청과 오류 메시지도 확인합니다.

4. 코드 예제

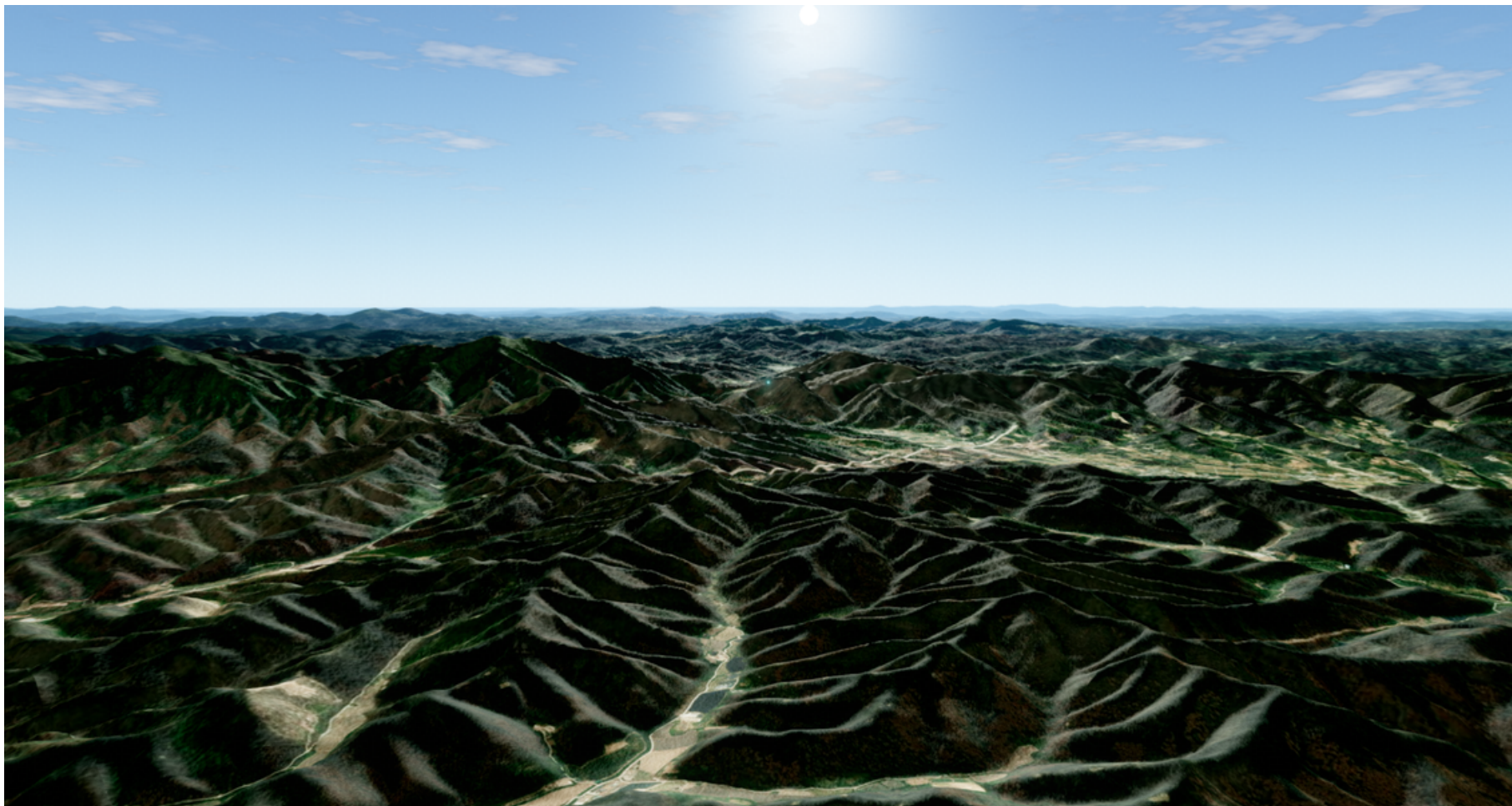
```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const layerName = "Korea_Terrain";
3
4 // U3dHeightXYZLayer를 생성하고 앱에 자동으로 등록합니다.
5 const layer = app.createHeightXYZLayer({
6   name: layerName,
7   baseurl: "/data/umf/korea-terrain",
8   ext: ".umf",
9
10  // Builder가 발행한 타일 주소 방향과 레벨을 그대로 입력합니다.
11  reverseY: true,
12  minlevel: 9,
13  maxlevel: 17,
14  useproxy: false
15 });
16
17 if (!layer) {
18   console.error("UMF 고도 레이어를 생성하지 못했습니다. 레이어 이름과 입력값을 확인하세요.");
19 } else {
20   layer.ready.then(
21     () => {
22       app.showLayer(layerName, true)
23         .catch((error) => {
24           console.error("UMF 고도 레이어 표시 실패", error);
25         });
26     },
27     (error) => {
28       console.error("UMF 고도 레이어 초기화 실패", error);
29       app.removeLayer(layerName);
30     }
31   );
32 }
33
34 // 숨기기: app.showLayer(layerName, false);
35 // 제거하기: app.removeLayer(layerName);
```

5. 출력 예시

UMF의 고도값은 지면의 높이에 적용됩니다. 카메라가 이동하거나 확대되면 해당 위치와 레벨의 고도 타일이 추가로 적용되어 지형의 높낮이가 표시됩니다.



umf 고도 자료 미적용



umf 고도 자료 적용

6. 권장 사항

필수 조건

- 개별 .umf 파일이 아니라 GeOnDT Builder가 생성한 메타데이터와 레벨/X/Y 고도 타일을 포함한 폴더 전체를 사용합니다.
- 기본 needXml: true 구성에서는 루트에 tilemapresource.xml을 포함합니다.
- EPSG:3857 레벨/X/Y 타일 구조와 파일명을 임의로 수정하지 않습니다.
- baseurl, ext, minlevel, maxlevel, reverseY를 실제 Builder 발행 결과와 일치시킵니다.
- 메타데이터의 타일 크기, 고도 샘플 크기, 스케일·오프셋, 압축 여부와 공간 범위를 실제 UMF 타일과 일치시킵니다. 공간 범위는 실제 경도·위도 범위와 일치해야 합니다.
- 모든 메타데이터와 타일을 접근 가능한 URL로 배포하고 파일명 대소문자와 상대 경로를 서버 구성과 일치시킵니다. 교차 출처에서는 CORS 또는 프록시를 설정합니다.

권장 조건

- IMG-GeoTIFF 원본의 좌표계, 고도 단위, 높이 기준과 데이터 없음 값을 확인한 뒤 GeOnDT Builder로 변환합니다.
- 서비스 범위와 화면 축척에 필요한 타일 레벨과 해상도만 생성하고, 값이 없는 타일과 불필요하게 중복된 타일을 줄입니다.
- 전체 데이터를 발행하기 전에 대표 지점의 원본 고도와 UMF 출력 고도를 비교합니다.
- 인접 타일의 경계 고도가 자연스럽게 이어지는지 확인하여 지형의 틈이나 단차를 줄입니다.
- 입력 포맷별 지원 범위, 좌표 변환, 보간법, 타일 크기, 고도 저장 형식, 데이터 없음 값과 압축 옵션의 상세 설정은 GeOnDT Builder 매뉴얼을 따릅니다.

선택 조건

- Builder 결과가 반전된 Y 타일 주소를 사용할 때만 reverseY: true를 설정합니다. 이 옵션은 고도값의 부호를 바꾸지 않습니다.
- Builder 발행 정보에서 메타데이터를 별도로 받은 경우에만 needXml: false로 설정하고 width, height, unitheight, boundingbox, 레벨과 공간 범위를 실제 결과에 맞게 입력합니다. 일반적인 폴더 배포에서는 tilemapresource.xml을 사용하는 기본 구성을 권장합니다.
- 프록시는 교차 출처 또는 서버 접근 정책상 필요한 경우에만 사용합니다.

UMesh

1. 포맷 정의와 공식 규격

UMesh는 GeOnDT for JS에서 사용하는 자체 바이너리 메시 포맷입니다. 3D 메시의 정점, 노멀, UV, 인덱스, 재질, 텍스처 리소스, 인스턴스 모델, 사용자 속성 등을 저장하여 GeOnDT for JS에서 빠르게 로드하고 출력하기 위한 데이터 포맷입니다.

- 공식 규격/참고: GeOnDT for JS 내부 UMesh 파서와 작성 규격을 기준으로 합니다.
- 유형: 파일/자체 바이너리 3D 메시

2. GeOnDT for JS 지원 여부 및 지원 버전

지원 상태: 직접 지원

버전은 1.0 또는 1.1 형식을 사용합니다. 버전 1.1에서는 srs 값이 포함되며, model 또는 world 값으로 모델 좌표 기준 저장 여부를 구분합니다.

3. 입력 방식

URL 또는 Data URI로 읽어 온 .umesh 바이너리 데이터를 입력으로 받습니다.

UMesh 파일은 헤더, 좌표계 및 영역 정보, Resource 목록, InstanceModel 섹션, MeshModel 섹션 순서의 바이너리 데이터로 구성됩니다.

파일 구성과 외부 리소스

텍스처를 사용하는 UMesh는 파일 내부 Resource 목록의 리소스 이름과 실제 참조 가능한 외부 리소스 경로가 일치해야 합니다.

오류 및 누락 시 결과

- magicKey, InstanceModel, MeshModel 식별자가 맞지 않으면 파서가 UMesh 구조를 올바르게 인식하지 못합니다.
- 필수 정점 정보가 누락되면 해당 메시 객체가 출력되지 않거나, 출력 형상에 오차가 생길 수 있습니다.
- 텍스처 리소스 참조가 누락되면 재질이 기본값으로 보이거나 텍스처가 표시되지 않을 수 있습니다.

4. 코드 예제

```
1 // app은 초기화가 끝난 GeOnDT 애플리케이션 인스턴스입니다.
2 const model = {
3   name: 'sample-umesh',
4   baseUrl: 'http://test-sever/data/models/',
5   fileName: 'GeonDT_file.umesh',
6   ext: 'umesh'
7 };
8
9 const layer = new GeOnDT.model.U3dModelBasicLayer({
10   name: 'UMesh_Model',
11   needxml: false,
12   listmodel: [model]
13 });
14
15 app.addLayer(layer);
16 app.showLayer(layer.getName(), true);
17
18 // 제거: app.removeLayer(layer.getName());
```

5. 출력 예시

UMesh 출력 예시 이미지는 현재 제공되지 않습니다.

6. 권장 사항

필수 조건

- 메시의 형상 정점 정보는 모델 원점 기준의 로컬 좌표로 이루어져 있어야 합니다.
- 버전 1.1의 srs가 model이면 모델 좌표 기준으로 저장하고, world이면 월드 좌표 기준으로 저장합니다.
- UMesh 파일의 magicKey는 UMeshWriter여야 합니다.
- 인스턴스 섹션의 식별자는 InstanceModel이어야 하고, 메시 섹션의 식별자는 MeshModel이어야 합니다.
- mesh, component, line, linesegments 타입의 객체는 정점 정보를 가져야 합니다.
- 텍스처를 사용하는 경우 리소스 목록의 name과 재질의 리소스 참조명이 일치해야 합니다.

권장 조건

- normal, uv, index는 형상 타입에 따라 비어 있을 수 있으나, 먼 메시에서는 정상 렌더링과 텍스처 매핑을 위해 정점 수와 대응되도록 구성하는 것을 권장합니다.
- 형상의 중심점이 가운데에 위치되도록 권장합니다. 중심점이 형상에서 크게 벗어나 있으면 회전, 스케일, 위치 보정, 카메라 포커싱, 객체 선택 시 기준점이 어긋나 모델이 의도와 다르게 움직이거나 배치될 수 있습니다.
- 형상 바운딩박스는 실제 형상 크기와 위치에 맞도록 합니다. 바운딩박스가 형상보다 과하게 크거나 작으면 화면 표시 범위 계산, 카메라 이동/줌, 객체 선택, 충돌 검사, 가시성 판단, LOD 처리에서 오차가 발생할 수 있습니다.
- 사용하지 않는 정점, 노멀, UV, 재질, 텍스처 리소스는 제거하는 것을 권장합니다.
- 같은 텍스처를 여러 메시가 사용하는 경우 리소스를 중복 저장하지 않는 것을 권장합니다.
- 웹 로딩 성능을 위해 정점 수, 인덱스 수, 텍스처 용량을 서비스 환경에 맞게 조정합니다.