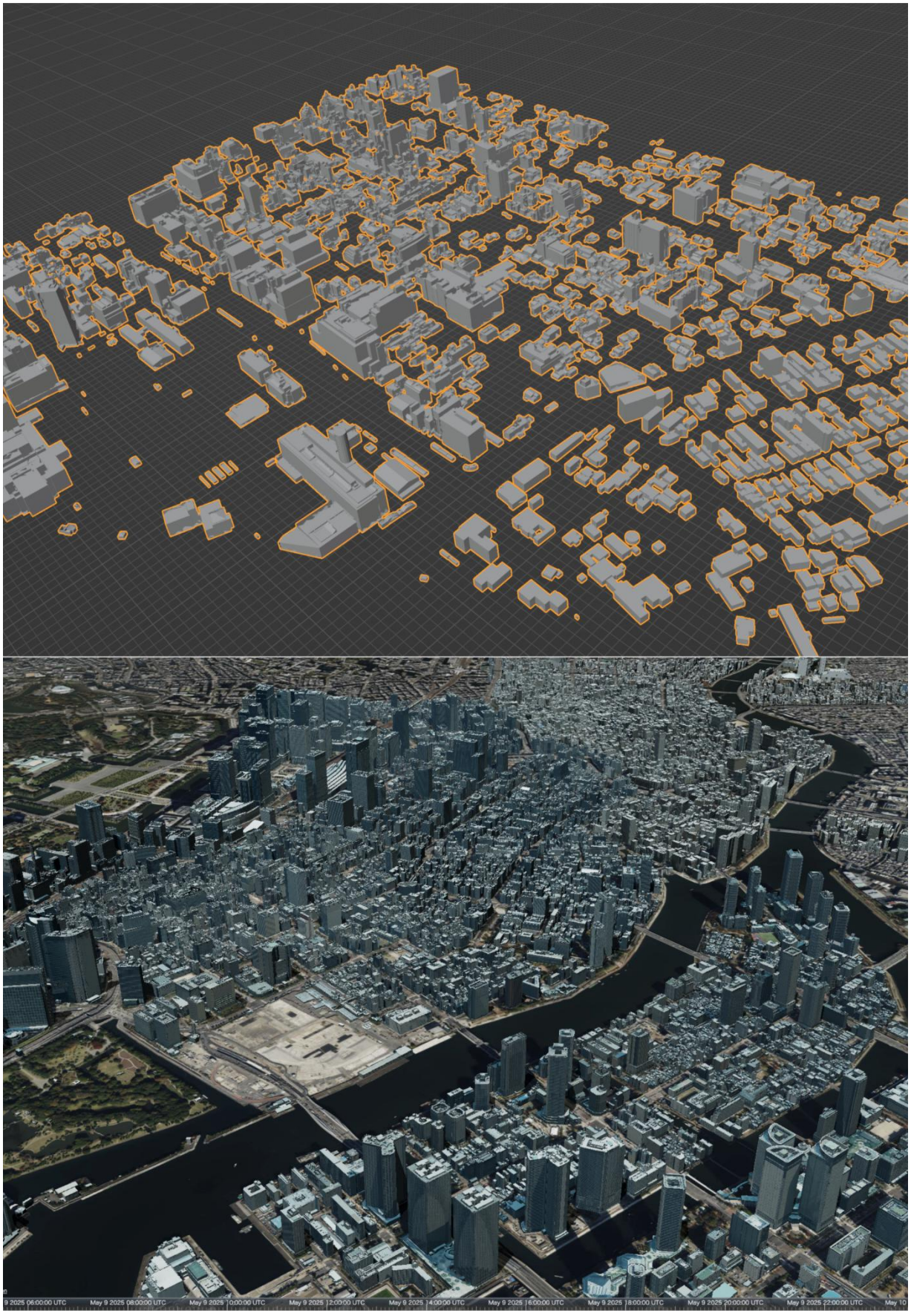
PLATEAU by MLIT

PLATEAU Technical Report

3D都市モデル活用のための技術資料



GIS コンバーターの開発　技術検証レポート

series No. 141

Technical Report on the Development of a GIS Converter

目次

1. GIS コンバーターの概要

1-1. 課題認識

1-1-1. 現状と課題

PLATEAU の標準仕様に準拠した CityGML2.0 形式の 3D 都市モデルは、測量や GIS の専門ツールを用いて大規模に生成することを基本としている。そのような状況の中、個人や専門分野の異なるベンダーなどにおいて簡易に 3D 都市モデルを作成し、研究や検証に用いるニーズが高まっている。しかしながら、現状、一般ユーザーが他の GIS 形式に変換できる簡易に利用可能な汎用的ツールは存在していない。

1-1-2. 過年度の手法とその認識

2023 年度業務では、3D 都市モデルを GeoJSON、Shapefile、GeoPackage、glTF、3DTiles、KML、CZML、Mapbox Vector Tiles に変換が可能なツール「PLATEAU GIS Converter」を開発した。
また、2024 年度ではテクスチャ付き 3D Tiles を変換した際に、Cesium での可視化時に描画パフォーマンスが落ちる課題を解決したほか、OBJ 形式への出力にも対応。さらに、出力可能な LOD を選択できる機能も実装し、3D 都市モデル変換ツールとして、一定の目処がついた。

とはいえ、新規ユーザーが 3D 都市モデルを快適に利用開始することができるかというとそうではなく、まだ一定以上のハードルが存在する。

ダウンロードした 3D 都市モデル群（zip ファイル）には大量の XML ファイル（CityGML）が格納されているが、ファイル名を一見しただけでは必要な箇所はどれなのかわからず、bldg などの地物型（建築物、道路などの地物の分類）などの省略記号でフォルダ分けされているため、必要なデータが分かりにくい。
このためユーザはデータ整備範囲を示す地図から必要なメッシュ番号を洗い出し、膨大なファイルの中から必要ファイルのみを抽出する必要があるためこれを解決する必要がある。

また、建築分野などでデータを利用する場合は OBJ 形式への変換などが利用されるが、現状の OBJ ファイル形式出力ではデータの重心位置を原点（0, 0）にシフトした ENU 座標系でのみ出力可能となっているため、複数のモデルを組み合わせる際に位置合わせがしづらいという課題もある。

加えて、昨年度アンケート結果より、GeoPackage など GIS データ形式に変換した場合、高 LOD では一部属性情報やポリゴンの欠落が見られる、といった意見があったため合わせて調査する必要がある。

1-1-3. 既存業務フロー

PLATEAU GIS Converter を利用した 3D 都市モデルの一般的な利用フローは以下の通りである。データをダウンロードしたのち、膨大なデータ整備情報から必要な範囲や地物型を選定し、その上でデータ変換を行う必要があった。特定の業者など、PLATEAU 利用に慣れていれば直感的に利用出来るツールではあるが、初めて

触るユーザーにとっては、依然ハードルが高いフローとなっている。

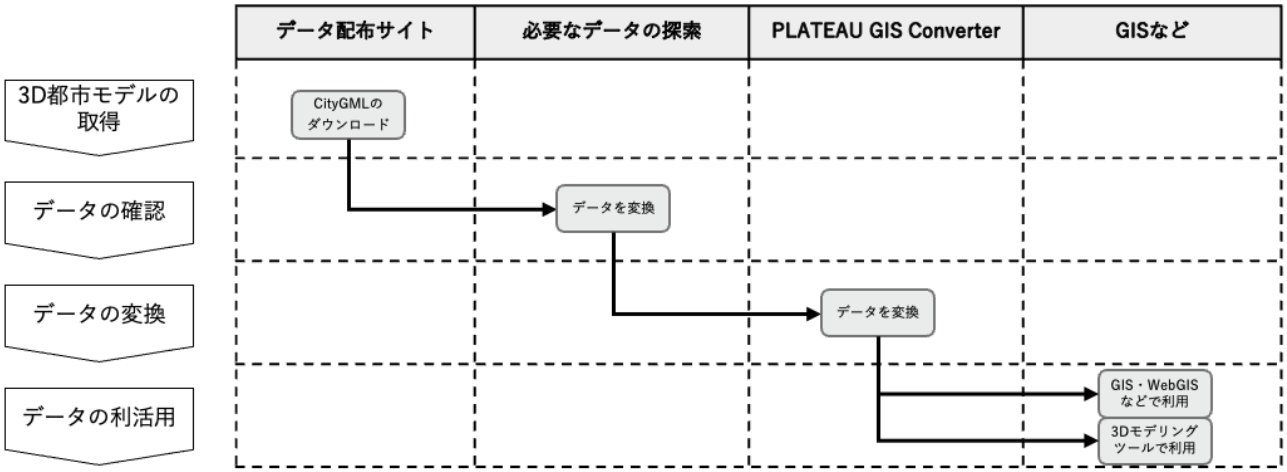


図 1-1　従来の業務フロー

1-2. 課題解決のアプローチ

ユーザーがデータをダウンロードしてから実際に利用するまでのステップ数を減らし敷居を下げる目的で、今年度開発では zip ファイルごとツールに取り込み、地図画面に 3D 都市モデルの範囲を表示する機能を導入する。
zip ファイルの中に格納されている地物型を抽出して表示するほか、ファイル名となっているメッシュ番号を取り出し、地図上にメッシュを表示する。
ユーザーは変換したいメッシュを地図上から直感的に選択し、地図画面からシームレスに必要なデータを出力できるようになる。

また、今年度では OBJ 形式など 3D データ形式の出力時に平面直角座標系を選択できるようになるため、BIM/CIM との親和性も高まることが考えられる。

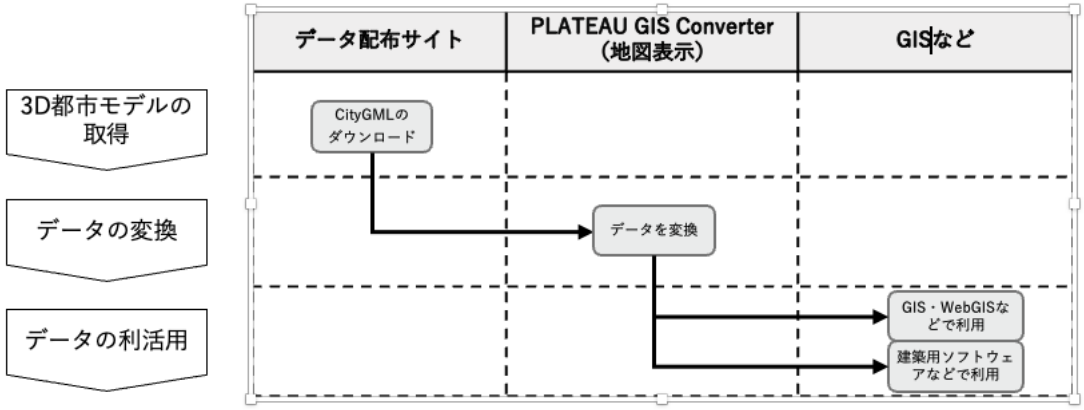


図 1-2　改善後の業務フロー

1-3. 創出価値

3D 都市モデルは、スマートシティを実現するための基盤となるデジタル工教材として幅広い分野での活用が期待されている。しかし、データ利用のハードルの高さにより、専門家以外の利用促進が制限されている。

本年度開発により実現する地図ベースでの直感的なデータ選択機能は、従来発生していたムダを省き、変換作業の負荷を削減する。
これにより、専門知識を持たない利用者でも即座に必要なデータを抽出・活用できるようになり、3D 都市モデルのアクセシビリティが飛躍的に向上する。
さらに、平面直角座標系への出力対応により、BIM/CIM 連携等が強化され、建築分野などとの連携も容易になり、異分野間でのデータ共有等が実現する。

1-4. 想定事業機会

表 1-1 想定事業機会

項目	内容
利用者	● 測量会社 ➢ GeoJSON・GeoPackage・Shapefile・ MVT・3D Tiles・glTF・OBJ・KML・CZML ● 建設コンサルタント ➢ GeoJSON・GeoPackage・Shapefile・ MVT・3D Tiles・glTF・OBJ・KML・CZML ● 3D モデリング・映像作成業者 ➢ glTF・OBJ
サービス仮説	● 測量会社 ➢ 既存の 2D 地図データと 3D 都市モデルを組み合わせた高精度な地理空間解析 ➢ 地下構造物や地質情報の 3D 可視化 ● 建設コンサルタント ➢ 都市計画などの 3D シミュレーション ➢ 建築物の日影分析や景観評価 ➢ 災害リスク評価や防災計画の立案 ● 3D モデリング・映像作成業者 ➢ リアルな都市空間を背景とした 3DCG コンテンツの制作 ➢ バーチャルツアーや AR/VR アプリケーションの開発 ➢ 都市景観のビジュアライゼーションや映像制作
提供価値	● 測量会社 ➢ より正確で詳細な地形情報の提供 ➢ 地下を含む立体的な空間情報の可視化 ➢ 効率的な測量作業と短納期の実現 ● 建設コンサルタント ➢ より精緻な都市計画や開発提案の実現 ➢ 視覚的にわかりやすい環境アセスメント結果の提示 ➢ データに基づいた合理的な防災・減災計画の策定 ● 3D モデリング・映像作成業者 ➢ 高品質で現実感のある 3D コンテンツの制作 ➢ インタラクティブな都市体験アプリケーションの開発 ➢ 都市の未来像を視覚的に表現する映像制作

2. 検証の概要

2-1. 検証仮説

昨年度までの開発により、PLATEAU GIS Converter は主要な GIS・3D データ形式への変換機能を実装し、特に 3D Tiles については大幅な軽量化とテクスチャ付き大規模変換を実現した。これにより、低スペック PC や低速インターネット環境でもデジタルツインの推進が可能となり、3D 都市モデルの活用基盤は整備されつつある。
しかしながら、実際の利用場面においては、データ準備段階での作業負荷が依然として高く、特に初心者や非専門家にとって大きな参入障壁となっている。具体的には、必要なメッシュ番号の特定から該当ファイルの抽出まで、複数の手順を経る必要があり、この煩雑さが 3D 都市モデルの普及を妨げる要因となっている。

本検証では、以下の 3 つの仮説を設定し、その有効性を検証する。

- 地図画面上でメッシュを視覚的に選択できる機能を実装することで、従来必要だった「地図確認→メッシュ番号特定→ファイル検索→変換実行」という 4 段階の作業を「地図上で選択→変換実行」の 2 段階に短縮できる。これにより、データ準備にかかる時間を半減でき、専門知識がなくても直感的な操作が可能になると想定する。
- OBJ 形式などの 3D データ出力時に平面直角座標系を選択可能にすることで、複数の地物型を統合する際の座標合わせ作業が不要となる。特に BIM/CIM 分野での利用において、従来必要だった座標変換処理が省略でき、建築・土木分野との連携がスムーズに行えるようになると想定する。
- GeoPackage や GeoJSON 等 GIS データ形式への変換時における属性情報やジオメトリの整合性を改めて検証・改善することで、GIS ソフトウェアでの読み込みエラーを解消する。これにより、昨年度アンケートで報告された「ポリゴンや属性情報の欠落」という問題を解決し、GIS 分野での PLATEAU データ活用が促進されると想定する。

2-2. 検証ポイント

- システム利用者によるユーザービリティ向上
  - システム利用者（航測会社・建設コンサルタントなど）に対して Google フォームでのヒアリング・アンケート調査を実施し、下記 4 点を KPI としてシステムのユーザビリティの向上を検証する。
    - UI の使いやすさ
    - 変換されたデータの使いやすさ
    - データの正しさ、業務への適用
    - データ利用までの手間・時間の短縮

2-3. 検証フロー

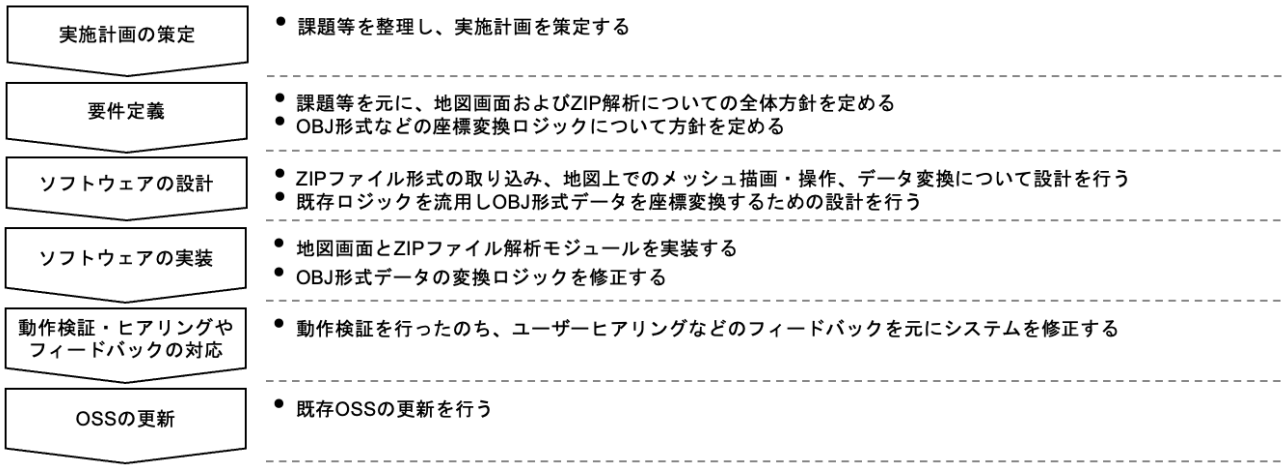


図 2-1 検証フロー

2-4. 実施体制

表 2-1 実施体制

役割	主体	詳細
全体管理	国交省 都市局	● プロジェクト全体ディレクション
	社会基盤情報流通推進協議会	● プロジェクトマネジメント
実施事業者	MIERUNE	● PLATEAU GIS Converter の改修

2-5. スケジュール

表 2-2 スケジュール

実施事項	2025 年							2026 年		
	6 月	7 月	8 月	9 月	10 月	11 月	12 月	1 月	2 月	3 月
1. 実施計画書の作成	◆――	――◆								
2. 要件定義書の作成		◆――	――◆							
3. 地図画面の作成・ZIP ファイル解析		◆――	――	――◆						
4. OBJ ファイルの座標系対応		◆――	――	――◆						
5. GIS ファイルの不具合調査・修正			◆――	――	――◆					
6. ユーザーヒアリング						◆――	――◆			
7. テスト・フィードバックの対応							◆――	――◆		
8. 技術検証レポート				◆――	――	――	――	――◆		
9. OSS のリリース							◆――	――	――◆	

3. 開発スコープ

3-1. 概要

本開発では、昨年度までに実装された PLATEAU GIS Converter の利用ハードルを大幅に下げることを目的とし、ユーザビリティの向上・改善に重点的に取り組む。
具体的には、3D 都市モデルの zip ファイルを直接ツールに取り込み、地図画面上にメッシュ範囲と地物型を可視化する機能を実装する。これにより、ユーザーは地図上でマウス操作により必要な範囲を直感的に選択でき、メッシュ番号の確認や手動でのファイル抽出作業が不要となる。加えて、選択したメッシュから直接変換処理を実行できるシームレスな操作フローを実現する。
また、建築・土木分野での利用拡大を図るため、OBJ 形式などの 3D データ出力時に平面直角座標系を選択可能とする機能を追加する。これにより、複数の地物型データを統合する際の座標合わせ作業が不要となり、BIM/CIM 環境での活用が促進される。
さらに、GIS ソフトウェアでの利用促進のため、GeoPackage および GeoJSON 形式への変換処理について不具合の検証を行い、属性情報とジオメトリの整合性を確保する改善を行う。これにより、変換後のデータの互換性を高め、幅広い GIS ツールでの活用を可能とする。

3-2. 開発内容

「4．実証システム」にて記載

4. 実証システム

4-1. アーキテクチャ

4-1-1. システムアーキテクチャ

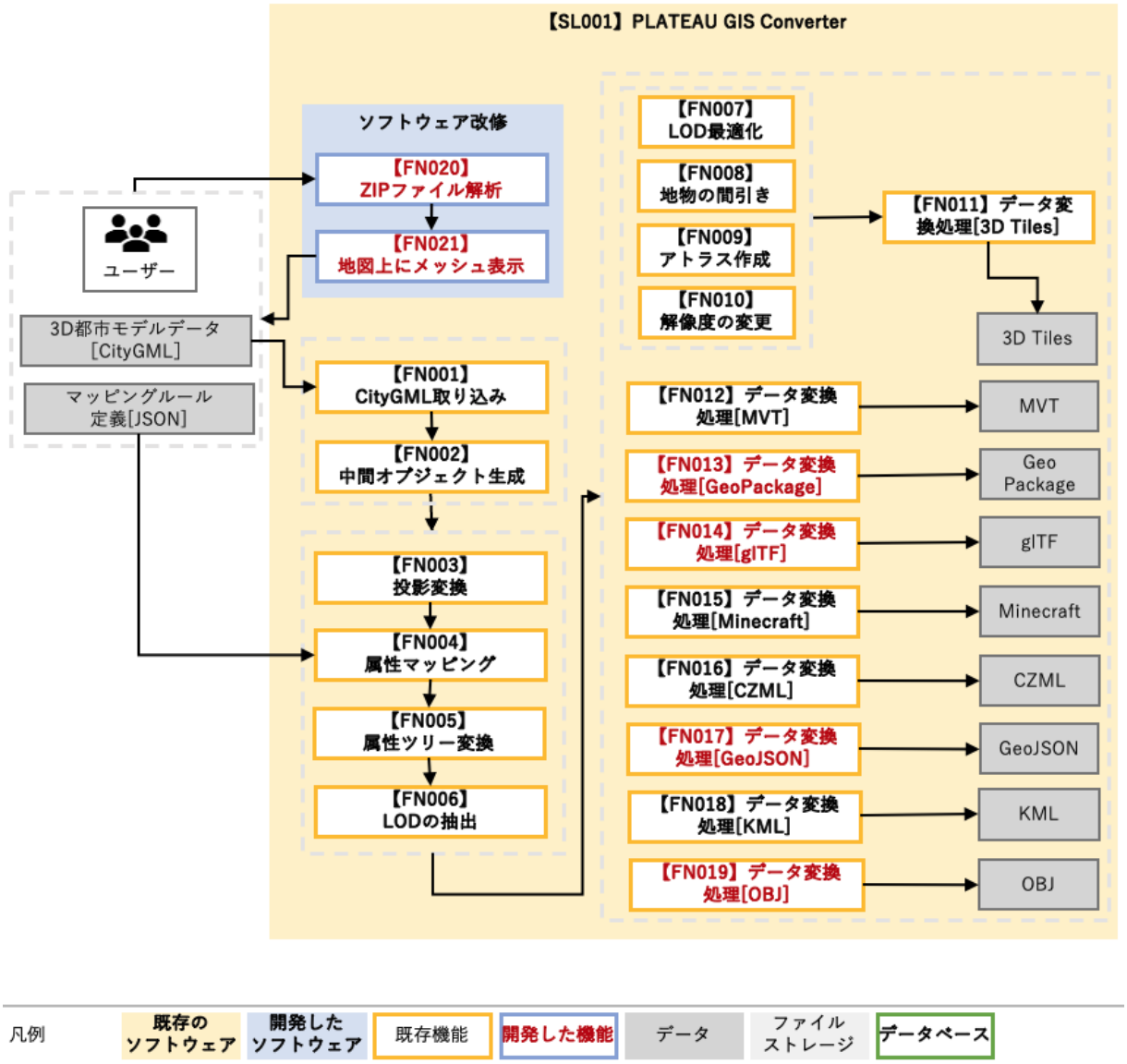


図 4-1 システムアーキテクチャ

4-1-2. データアーキテクチャ

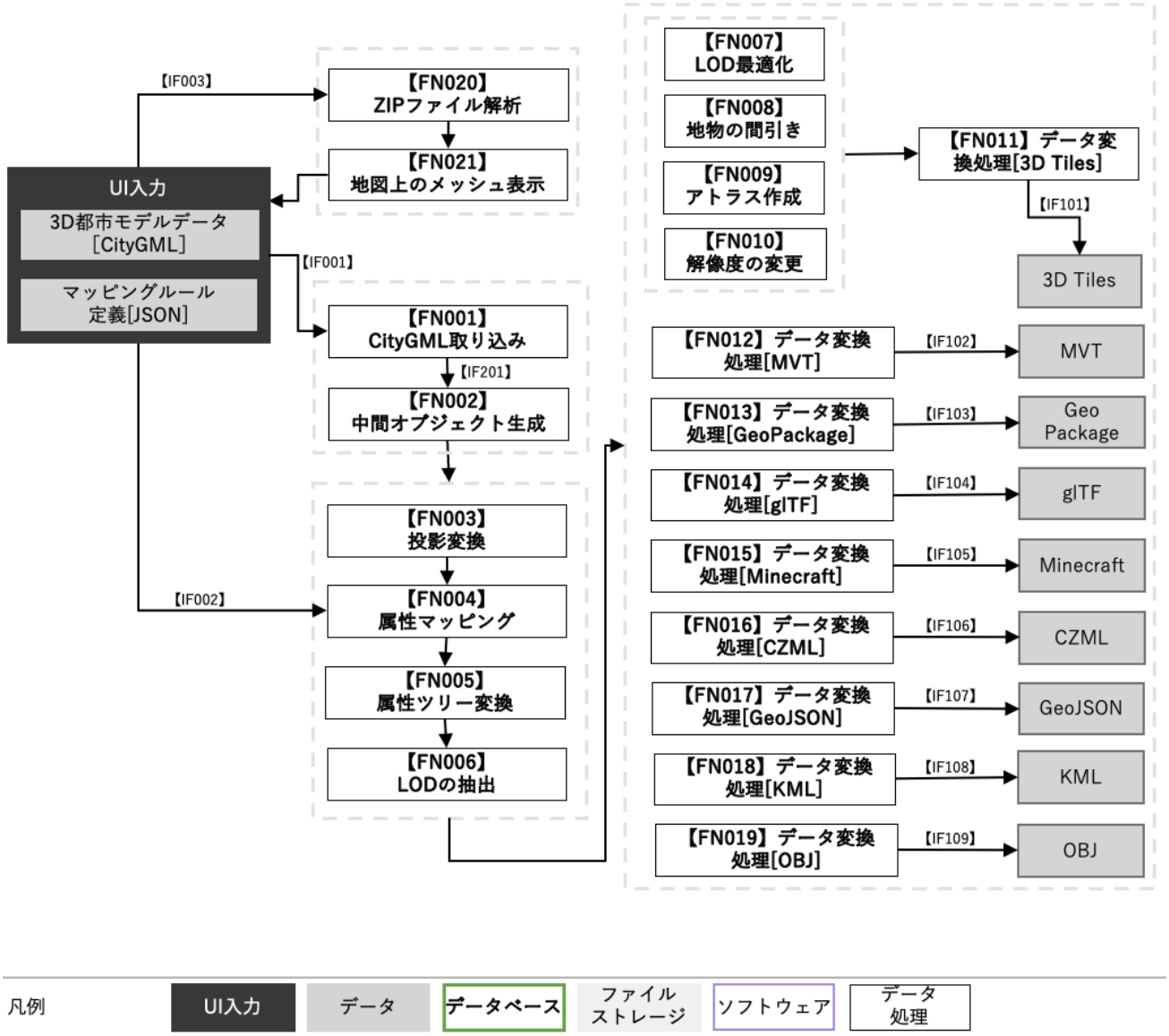


図 4-2 データアーキテクチャ

4-1-3. ハードウェアアーキテクチャ

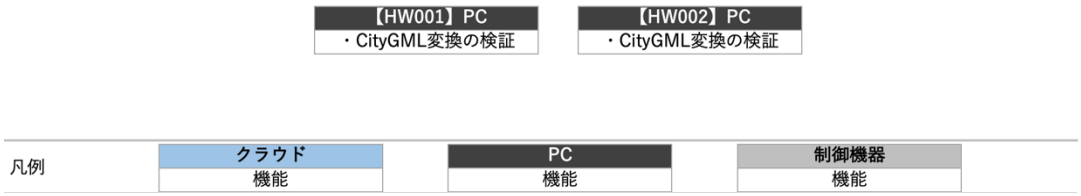


図 4-3 ハードウェアアーキテクチャ

表 4-1 利用するハードウェア一覧

ID	種別	品番	用途
HW001	PC	Lenovo ThinkBook 13s G3 ACN	● CityGML 変換の検証
HW002	PC	Apple Macbook Air 13inch	● CityGML 変換の検証

1) 【HW001】PC：Lenovo ThinkBook 13s G3 ACN

- 選定理由
  - PLATEAU GIS Converter を利用するための一般的なスペックであること
- 仕様・スペック
  - CPU：AMD Ryzen 7 5800U with Radeon Graphics 1.90GHz
  - メモリ：16GB
  - ストレージ：512GB
  - OS：Windows 11 Pro 22H2
- イメージ

図 4-4 Lenovo ThinkBook 13s G3 ACN[1]

[1] 公式 HP より抜粋：https://www.lenovo.com/jp/ja/p/laptops/thinkbook/thinkbook-series/thinkbook-13s-g3-acn/xxtbxtsa301

2)　【HW002】PC：Apple Macbook Air 13inch

- 選定理由
  - PLATEAU GIS Converter を利用するための一般的なスペックであること
- 仕様・スペック
  - CPU：Apple M1
  - メモリ：8GB
  - ストレージ：512GB
  - OS：macOS Sonoma
- イメージ

図 4-5 Apple Macbook Air 13inch[2]

[2] 公式 HP より抜粋：https://www.apple.com/jp/shop/buy-mac/macbook-air

4-2. システム機能

4-2-1. システム機能一覧

機能一覧を以下に示す。

表 4-2 機能一覧

※赤文字：2025 年度　新規開発・既存改修

大分類	小分類	ID	機能名	機能説明
空間データ変換機能	CityGML パーサー	FN001	CityGML 取り込み	● 指定された 3D 都市モデルを読み込み、プログラムへ取り込む
		FN002	中間オブジェクト生成	● 3D 都市モデル内の属性・空間属性を抽出し、中間オブジェクトに変換する
	データ変換	FN003	投影変換	● 中間オブジェクトに格納された座標情報を一括で変換する
		FN004	属性マッピング	● 中間オブジェクトをマッピングルール定義ファイルに記載の通り、属性名と型を変化させて、新たな中間オブジェクトを生成する
		FN005	属性ツリー変換	● 中間オブジェクトに格納された、入れ子になった主題属性を抽出し、JSON 化・配列化・別テーブル化などを行う
		FN006	LOD の抽出	● 中間オブジェクトに格納された、空間属性を個別に取り出すもしくはフィルタリングするなどの処理を行い、別の中間オブジェクトを出力する
	データ出力	FN007	LOD 最適化	● geometricError に応じ

			て、表示する LOD を切り替える
	FN008	地物の間引き	● geometricError に応じて、一定以下の高度を持つ地物を間引く
	FN009	アトラス作成	● CityGML に埋め込まれているテクスチャ画像をテクスチャアトラスにパッキングし、出力する
	FN010	解像度の変更	● geometricError に応じて、解像度を下げる
	FN011	データ変換処理[3D Tiles]	● ジオメトリのタイル分割・glb 出力・tileset.json 出力処理を行う
	FN012	データ変換処理[MVT]	● ジオメトリのタイル分割・pbf 出力処理を行う
	FN013	データ変換処理[GeoPackage]	● 入れ子属性や異なる地物型を別テーブルにする処理を行い、複数テーブル保有した GeoPackage を出力する ● 不具合の修正
	FN014	データ変換処理[glTF]	● 属性情報が付与された状態で glb 出力処理を行う ● 平面直角座標系への対応
	FN015	データ変換処理[Minecraft]	● Minecraft で利用可能な Anvil 形式のファイルを出力する
	FN016	データ変換処理[CZML]	● CZML 形式のファイルを出力する

		FN017	データ変換処理[GeoJSON]	● GeoJSON 形式のファイルを出力する ● 不具合の修正
		FN018	データ変換処理[KML]	● KML 形式のファイルを出力する
		FN019	データ変換処理[OBJ]	● OBJ ファイル仕様に沿ったファイル群を出力する ● 平面直角座標系への対応
	データ変換	FN020	ZIP ファイル解析	● ZIP ファイルを解凍せずフォルダ構造を読み取り、メッシュと地物型を一覧化する
		FN021	地図上にメッシュ表示	● ZIP ファイルから読み取ったメッシュ一覧から、地図上にメッシュを表示する

4-2-2. 利用したソフトウェア・ライブラリ

利用するライブラリ等を以下に示す。

表 4-3 利用したソフトウェア・ライブラリ

ID	項目	内容
SL001	Tauri	1. デスクトップアプリ開発のための GUI フレームワーク
SL002	Svelte	2. UI 構築のための JavaScript フレームワーク
SL003	rayon	3. 並列処理を行うためのライブラリ
SL004	tokio	4. 非同期処理を行うためのライブラリ
SL005	image	5. 画像処理・変換のためのライブラリ
SL006	quick-xml	6. XML のパースを行うためのライブラリ
SL007	serde	7. シリアライズ・デシリアライズを行うためのインターフェース
SL008	serde_json	8. serde を利用した Rust 用の JSON 変換用ライブラリ
SL009	clap	9. CLI ツールを簡単に作成するためのライブラリ
SL010	stretto	10. キャッシュのためのライブラリ
SL011	sqlx	11. データベース操作のためのライブラリ
SL012	Svelte MapLibre GL	12. Svelte で MapLibre GL JS を利用するためのライブラリ
SL013	zip	13. ZIP ファイル解析のためのライブラリ

4-2-3. 開発機能の詳細要件

開発機能の詳細要件を記す。なお、本業務において新規開発した要素（機能名）を示す。

1. 【FN001】CityGML 取り込み

14. 機能概要
    - 指定された 3D 都市モデルを読み込み、プログラムへ取り込む

15. フローチャート

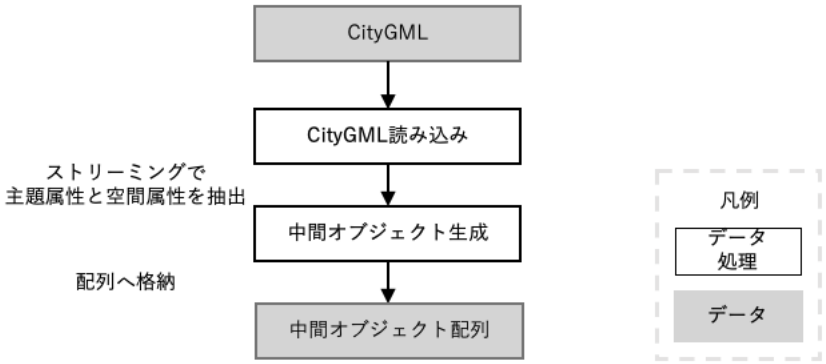


図 4-6 フローチャート

16. データ仕様
    - 入力
        - 3D 都市モデルを受け取る
            - 内容
                - 3D 都市モデル
            - 形式
                - CityGML
            - データ詳細
                - ファイル入力インタフェース【IF001】を参照
    - 出力
        - Entity
            - 内容
                - 対象の CityGML をストリーミングで受け取り、中間形式である Entity を出力する
            - 形式
                - プログラムの内部構造体
            - データ詳細
                - 内部連携インターフェース【IF201】を参照

17. 機能詳細
    - CityGML 取り込み

- ✧ 処理内容
  - 指定されたファイルパスから CityGML を読み込む
  - 読み込まれた CityGML はストリーミング処理され、地物ごとに「中間オブジェクト生成」アルゴリズムを利用し、主題属性・空間属性が整理される
  - 全ての地物に対して再帰的に処理を行い、全ての CityGML を読み取り次第、処理を完了する
- ✧ 利用するライブラリ
  - 【SL003】【SL006】【SL007】【SL008】を参照
- ✧ 利用するアルゴリズム
  - 【AL101】【AL102】を参照

2. 【FN002】中間オブジェクト生成

18. 機能概要

- 3D 都市モデル内の属性・空間属性を抽出し、中間オブジェクトに変換する

19. フローチャート

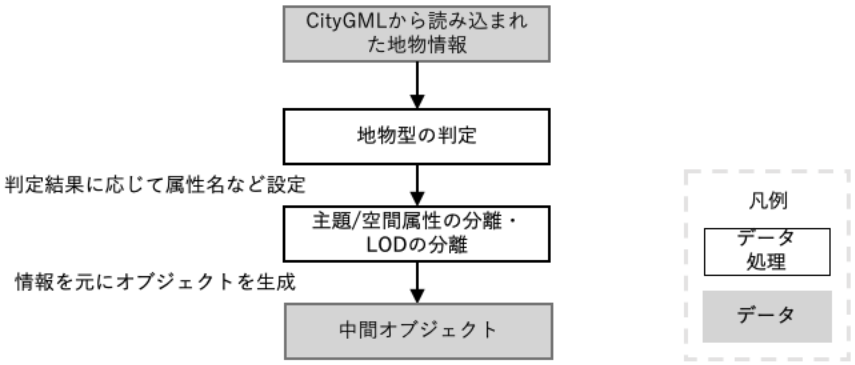


図 4-7 フローチャート

20. データ仕様

- 入力
  - 読み込まれた 3D 都市モデル（CityGML）の断片を受け取る
    - 内容
      - 3D 都市モデル
    - 形式
      - CityGML
    - データ詳細
      - ファイル入力インタフェース【IF001】を参照
- 出力
  - Entity
    - 内容
      - 対象の CityGML をストリーミングで受け取り、中間形式である Entity を出力する
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

21. 機能詳細

- 中間オブジェクト生成
  - 処理内容
    - 「PLATEAU3D 都市モデル読み込み」アルゴリズムをより渡された CityGML の情報を読み取る

- 整理された情報地物型ごとにあらかじめ定義された形式に沿って、メモリ上で中間オブジェクト（クラス）へ変換される
- 変換のたびに可変長配列にインサートする
- 全ての地物に対して再帰的に処理を行い、全ての CityGML を読み取り次第、処理を完了する

✧ 利用するライブラリ

- 【SL003】【SL006】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL102】を参照

3. 【FN003】投影変換

22. 機能概要

- 中間オブジェクトに格納された座標情報を一括で変換する

23. フローチャート

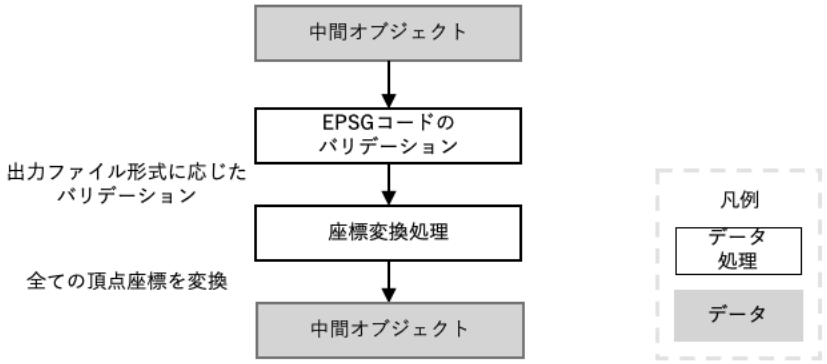


図 4-8 フローチャート

24. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - Entity
    - 内容
      - 中間形式である Entity を受けとり、加工済みの Entity を出力する
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

25. 機能詳細

- 投影変換
  - 処理内容
    - GUI 上からユーザーが変換先座標系の指定を行う
    - EPSG コードとして定義されている座標参照系の文字列と中間オブジェクトを受け取る
    - 以下のチェックを行う

- 存在しない EPSG コードが指定されていないか
- EPSG コードは出力ファイル形式に適用できる座標参照系かどうか
  - 例: MVT は EPSG:3857 にのみ対応・GeoJSON は仕様上 EPSG:4326 のみに対応、など

- バリデーション後、規定の数値計算を行い、全ての座標を変換し、変換後の中間オブジェクトを作成する

- 利用するライブラリ
  - 【SL003】を参照
- 利用するアルゴリズム
  - 【AL103】を参照

4. 【FN004】属性マッピング

26. 機能概要

- 中間オブジェクトをマッピングルール定義ファイルに記載の通り、属性名と型を変化させて、新たな中間オブジェクトを生成する

27. フローチャート

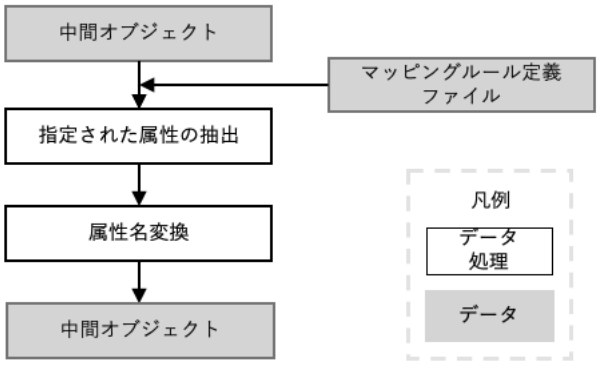


図 4-9 フローチャート

28. データ仕様

- 入力
  - マッピングルール定義ファイルを受け取る
    - 内容
      - 属性名マッピングのための定義ファイル
    - 形式
      - JSON
    - データ詳細
      - ファイル入力インタフェース IF002 を参照
- 出力
  - Entity
    - 内容
      - 中間形式である Entity を受けとり、加工済みの Entity を出力する
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

29. 機能詳細

- 属性マッピング
  - 処理内容
    - GUI から指定されたマッピングルール定義ファイルを読み込む
    - 中間オブジェクトから指定された属性名を抽出し、定義の通り変更する

- 全ての中間オブジェクトを走査し終えたら終了する

✧ 利用するライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL104】を参照

5. 【FN005】属性ツリー変換

30. 機能概要
    - 中間オブジェクトに格納された、入れ子になった主題属性を抽出し、JSON 化・配列化・別テーブル化などを行う

31. フローチャート

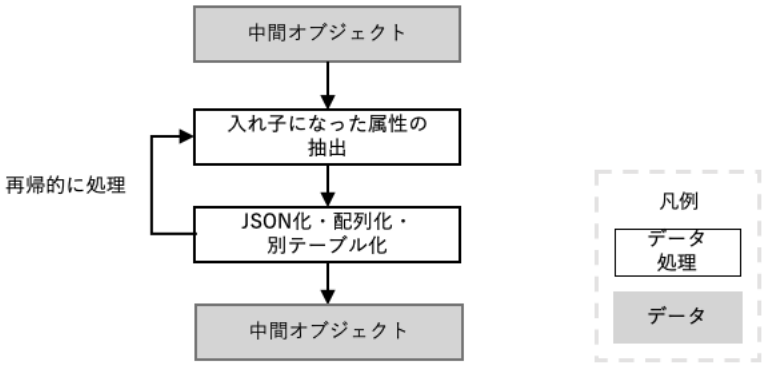


図 4-10 フローチャート

32. データ仕様
    - 入力
        - Entity
            - 内容
                - 中間形式である Entity
            - 形式
                - プログラムの内部構造体
            - データ詳細
                - 内部連携インターフェース【IF201】を参照
    - 出力
        - Entity
            - 内容
                - 中間形式である Entity を受けとり、加工済みの Entity を出力する
            - 形式
                - プログラムの内部構造体
            - データ詳細
                - 内部連携インターフェース【IF201】を参照

33. 機能詳細
    - 属性ツリー変換
        - 処理内容
            - 出力ファイル形式に応じて入れ子属性の抽出方法を定める
            - 出力ファイル形式に応じて、抽出方法が適切かどうかバリデーションする

- MVT や 3DTiles などは別テーブル化が不可、など
- 抽出方法に応じて、処理を行う
  - JSON 化・配列化
    - 子の属性名と値を取り出し、JSON 化・配列化する
    - 親の属性値に JSON・配列を格納する
    - 全ての子要素がなくなるまで、再帰的に処理を行う
  - 別テーブル化
    - 子の属性をルート要素まで持ち上げ、フラットな属性ツリーに変化させる
    - 属性に応じてテーブルやファイルなどを分割して出力する
    - 全ての子要素がなくなるまで再帰的に処理を行う

✧ 利用するライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL105】を参照

6. 【FN006】LOD の抽出

34. 機能概要

- 中間オブジェクトに格納された、空間属性を個別に取り出すもしくはフィルタリングするなどの処理を行い、別の中間オブジェクトを出力する

35. フローチャート

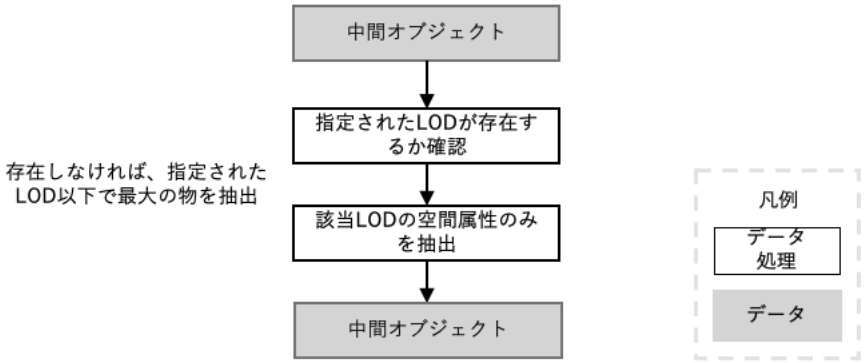


図 4-11 フローチャート

36. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - Entity
    - 内容
      - 中間形式である Entity を受けとり、加工済みの Entity を出力する
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

37. 機能詳細

- LOD の抽出
  - 処理内容
    - GUI からの指定により、抽出する空間属性を決定する
    - 地物ごとの空間属性を全て取り出す

- 指定されたパラメータに従い、空間属性のフィルタリングを行う
- フィルタリングされた空間属性のみを次の処理に渡し、終了する

✧ 利用するライブラリ

- 【SL003】を参照

✧ 利用するアルゴリズム

- 【AL106】を参照

7. 【FN007】LOD 最適化

38. 機能概要

- 広域表示では低 LOD の地物が表示され、狭域表示ではより高品質の LOD が表示される機能

39. フローチャート

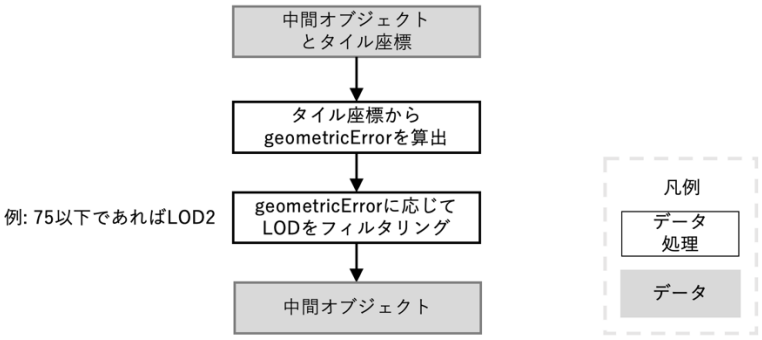


図 4-12 フローチャート

40. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity および`{z}/{x}/{y}`のようなタイル座標を示すテキスト
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - Entity
    - 内容
      - 中間形式である Entity を受けとり、加工済みの Entity を出力する
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

41. 機能詳細

- アトラス作成
  - 処理内容
    - 中間オブジェクトとタイル座標を受けとる
    - タイル座標から geometricError を計算する
    - geometricError が 150 以上であれば最低 LOD を抽出し、geometircError 半分になれば次

の LOD を抽出する、といったように geometricError に応じて LOD をフィルタリングする

- フィルタリングされた中間オブジェクトを返却する

✧ 利用するライブラリ

- 【SL003】を参照

✧ 利用するアルゴリズム

- 【AL120】【AL121】を参照

8. 【FN008】地物の間引き

42. 機能概要

- ズームレベルに応じて、事前に定めた高度以上の地物のみを抽出する

43. フローチャート

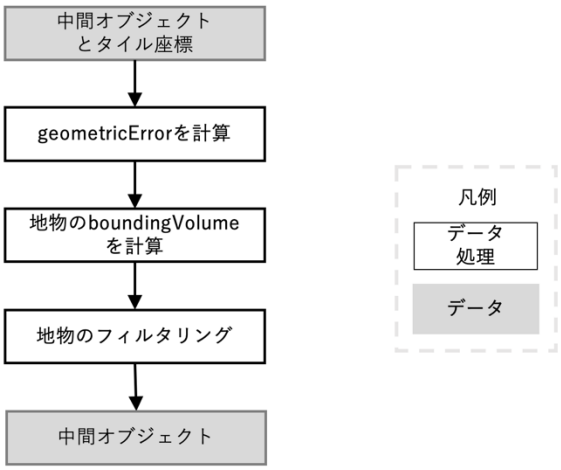


図 4-13 フローチャート

44. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity および`{z}/{x}/{y}`のようなタイル座標を示すテキスト
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - Entity
    - 内容
      - 中間形式である Entity を受けとり、加工済みの Entity を出力する
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

45. 機能詳細

- 地物の間引き
  - 処理内容
    - タイル座標から geometricError を算出する

- 地物の boundingVolume を算出する
- geometricError と事前に定めた係数を掛け合わせ、1 辺でも長さが上回れば抽出する

✧ 利用するライブラリ

- 【SL003】を参照

✧ 利用するアルゴリズム

- 【AL120】【AL108】を参照

9. 【FN009】アトラス作成

46. 機能概要

- CityGML に埋め込まれているテクスチャ画像をテクスチャアトラスにパッキングし、テクスチャアトラスの UV 座標を返す

47. フローチャート

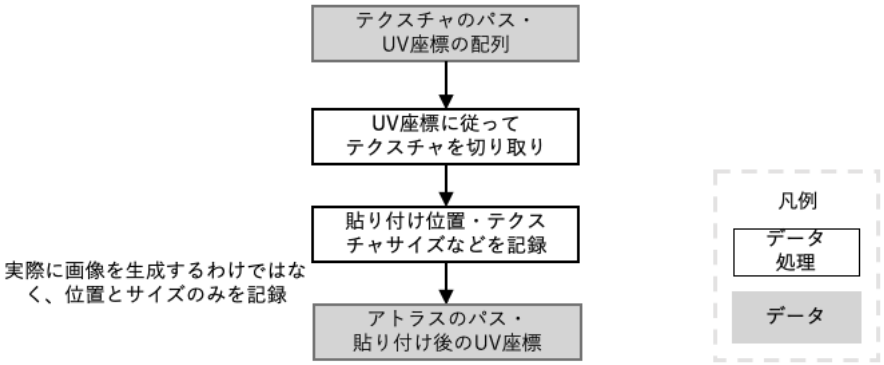


図 4-14 フローチャート

48. データ仕様

- 入力
  - テクスチャのファイルパスを示すテキストおよびテクスチャの UV 座標を示す配列
    - 内容
      - テクスチャのファイルパスを示すテキストと、中間形式である Entity 内部の AppearanceStore 属性に格納された UV 座標を示す配列
    - 形式
      - プログラムのテキストおよび配列
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - 加工後テクスチャのファイルパスを示すテキストおよびテクスチャの UV 座標を示す配列
    - 内容
      - 加工後のテクスチャのファイルパスを示すテキストと、中間形式である Entity 内部の AppearanceStore 属性に格納された UV 座標を示す配列
    - 形式
      - プログラムのテキストおよび配列
    - データ詳細
      - 内部連携インターフェース【IF201】を参照

49. 機能詳細

- アトラス作成
  - 処理内容

- 面ごとのテクスチャのパス・UV 座標を受け取る
- UV 座標に従って、テクスチャを切り抜く
- 大きなテクスチャ（アトラス）を模したオブジェクトに仮想的に貼り付けていく
- 全てのテクスチャを貼り付け終わったら処理を終了する

✧ 利用するライブラリ

- 【SL003】【SL005】【SL010】を参照

✧ 利用するアルゴリズム

- 【AL107】を参照

10. 【FN010】解像度の変更
50. 機能概要
    - ズームレベルに応じて、事前に定めた係数の通りに解像度を下げる
51. フローチャート

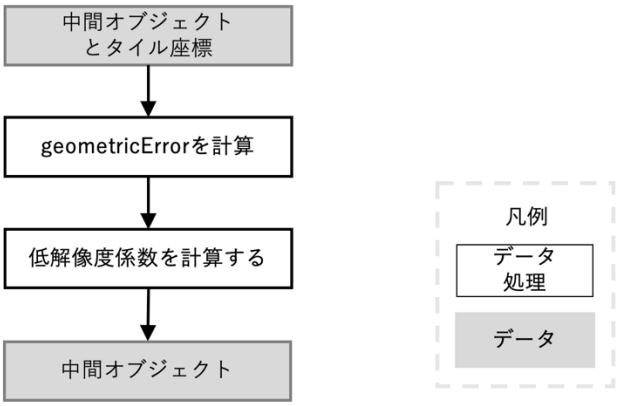


図 4-15 フローチャート

52. データ仕様
    - 入力
        - Entity
            - 内容
                - 中間形式である Entity および`{z}/{x}/{y}`のようなタイル座標を示すテキスト
            - 形式
                - プログラムの内部構造体
            - データ詳細
                - 内部連携インターフェース【IF201】を参照
    - 出力
        - Entity
            - 内容
                - 中間形式である Entity を受けとり、加工済みの Entity を出力する
            - 形式
                - プログラムの内部構造体
            - データ詳細
                - 内部連携インターフェース【IF201】を参照
53. 機能詳細
    - 解像度の変更
        - 処理内容
            - タイル座標から geometricError を計算する

- geometricError に応じて、低解像度化のための 0~1 の係数を算出する
- 中間オブジェクトに係数を保持する

✧ 利用するライブラリ

- 【SL003】【SL005】を参照

✧ 利用するアルゴリズム

- 【AL120】【AL109】を参照

11. 【FN011】データ変換処理[3D Tiles]

54. 機能概要

- FN001~FN010 までの処理が適切になされた中間オブジェクトの配列に対し、ジオメトリのタイル分割・glb 出力・tileset.json 出力処理を行う

55. フローチャート

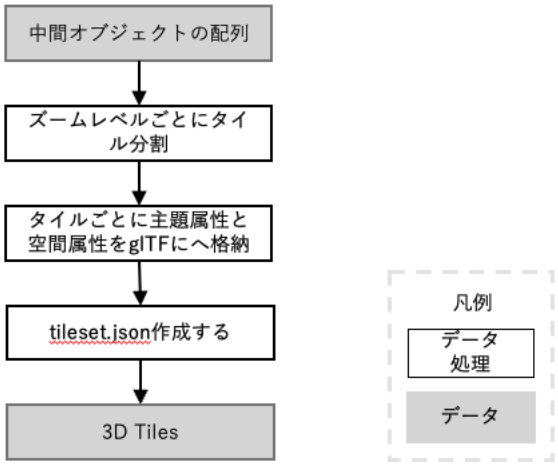


図 4-16 フローチャート

56. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - 3D Tiles
    - 内容
      - 3D Tiles が出力される
    - 形式
      - 3D Tiles
    - データ詳細
      - ファイル出力インターフェース【IF101】を参照

57. 機能詳細

- データ変換処理[3D Tiles]
  - 処理内容

- 変換対象地物が含まれる範囲をズームレベルごとにタイル分割する
- glTF 仕様に沿って、タイルごとにジオメトリと属性情報を格納していく
- tileset.json を出力する

✧ 利用するライブラリ

- 【SL003】【SL005】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL110】を参照

12. 【FN012】データ変換処理[MVT]

58. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、ジオメトリのタイル分割・pbf 出力処理を行う

59. フローチャート

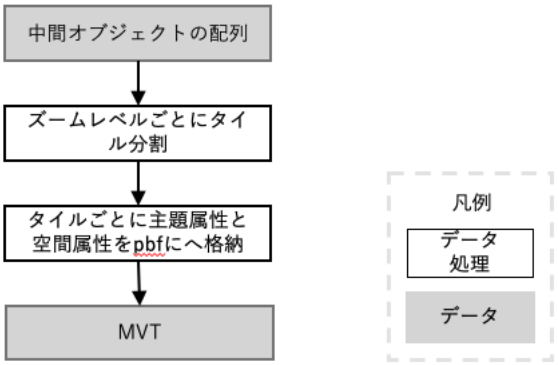


図 4-17 フローチャート

60. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - MVT
    - 内容
      - MVT が出力される
    - 形式
      - MVT
    - データ詳細
      - ファイル出力インターフェース【IF102】を参照

61. 機能詳細

- データ変換処理[MVT]
  - 処理内容
    - 変換対象地物が含まれる範囲をズームレベルごとにタイル分割する
    - Mapbox Vector Tile 仕様に沿って、タイルごとにジオメトリと属性情報を格納していく

- ズームレベルごとにフォルダを作成し、pbf ファイルを出力する

✧ 利用するライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL111】を参照

13. 【FN013】データ変換処理[GeoPackage]

62. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、入れ子属性や異なる地物型を別テーブルにする処理を行い、複数テーブル保有した GeoPackage を出力する
- ※2025 年度では、軽微なバグフィックスを行う

63. フローチャート

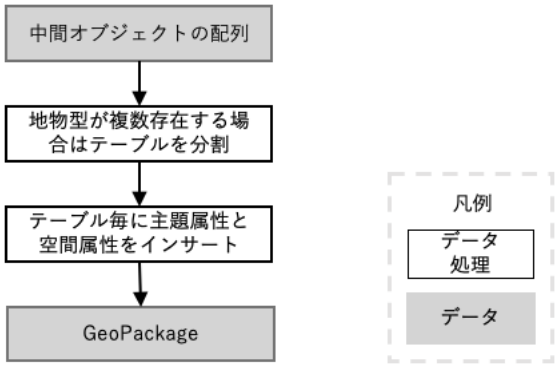


図 4-18 フローチャート

64. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - GeoPackage
    - 内容
      - GeoPackage が出力される
    - 形式
      - GeoPackage
    - データ詳細
      - ファイル出力インターフェース【IF103】を参照

65. 機能詳細

- データ変換処理[GeoPackage]
  - 処理内容
    - 入れ子属性や異なる地物型に応じてテーブルを作成する

- ジオメトリと属性情報をテーブルごとに格納する

✧ 利用するライブラリ

- 【SL003】【SL004】【SL007】【SL008】【SL011】を参照

✧ 利用するアルゴリズム

- 【AL112】を参照

14. 【FN014】データ変換処理[glTF]

66. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、属性情報が付与された状態で glTF 出力処理を行う

67. フローチャート

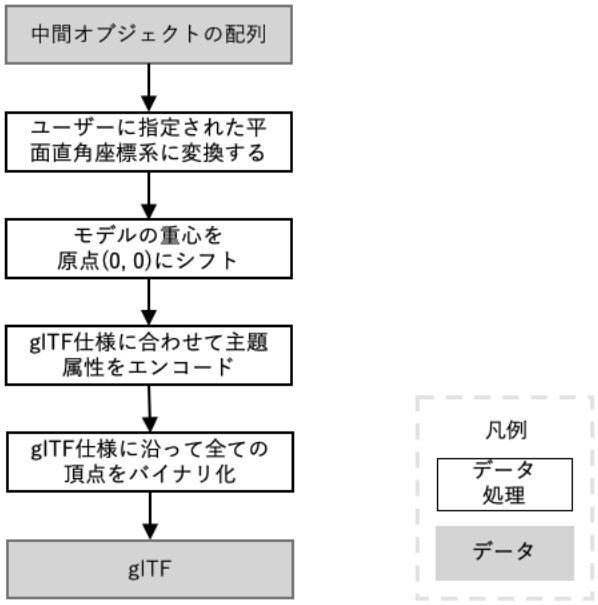


図 4-19 フローチャート

68. データ仕様

- 入力
  - ✧ Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - ✧ glTF
    - 内容
      - glTF が出力される
    - 形式
      - glTF
    - データ詳細
      - ファイル出力インターフェース【IF104】を参照

69. 機能詳細

- データ変換処理[glTF]
  - 処理内容
    - 全ての座標をユーザーに指定された平面直角座標系に変換する
    - モデルの重心を原点(0, 0)にシフト（オプション）
    - glTF 仕様に従って属性情報をエンコードする
    - glTF 仕様に従って頂点情報を格納する
    - ファイルの出力を行う
  - 利用するライブラリ
    - 【SL003】【SL005】【SL007】【SL008】を参照
  - 利用するアルゴリズム
    - 【AL103】【AL113】を参照

15. 【FN015】データ変換処理[Minecraft]

70. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、Minecraft で利用可能な Anvil 形式のファイルを出力する

71. フローチャート

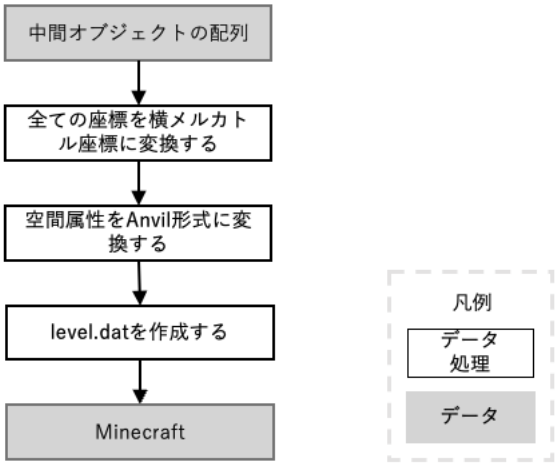


図 4-20 フローチャート

72. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - Anvil
    - 内容
      - Anvil 形式のファイル群が出力される
    - 形式
      - Anvil
    - データ詳細
      - ファイル出力インターフェース【IF105】を参照

73. 機能詳細

- データ変換処理[Minecraft]
  - 処理内容

- 全ての座標を、変換対象範囲の重心点を原点とした横メルカトル座標に変換する
- マインクラフトで利用される Anvil 形式でファイルを出力する
- メタデータとなる level.dat ファイルを出力する利用する

✧ ライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL114】を参照

16. 【FN016】データ変換処理[CZML]

74. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、CZML 形式の JSON ファイルを出力する

75. フローチャート

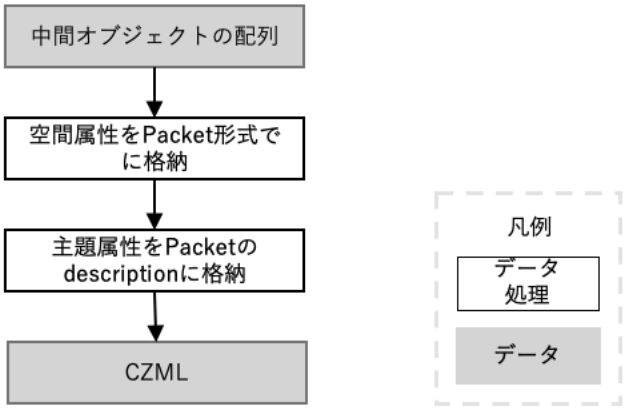


図 4-21 フローチャート

76. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - CZML
    - 内容
      - CZML 形式の JSON ファイルが出力される
    - 形式
      - JSON
    - データ詳細
      - ファイル出力インターフェース【IF106】を参照

77. 機能詳細

- データ変換処理[CZML]
  - 処理内容

- 中間オブジェクトからジオメトリを取り出し CZML 仕様である Packet 形式に格納する
- Packet の description 属性に、中間オブジェクトの主題属性を格納する
- 全ての中間オブジェクトを処理したら出力を行う

✧ ライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL115】を参照

17. 【FN017】データ変換処理[GeoJSON]
78. 機能概要
    - FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、GeoJSON 形式のファイルを出力する
    - ※2025 年度では、軽微なバグフィックスを行う
79. フローチャート

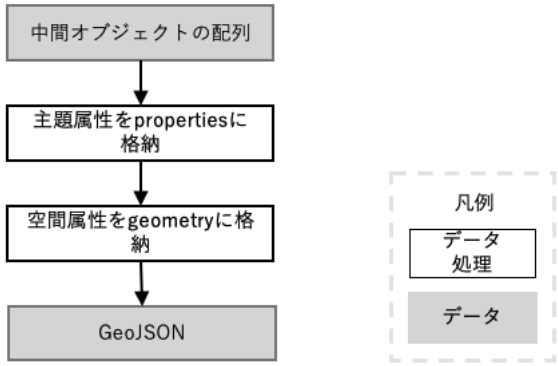


図 4-22 フローチャート

80. データ仕様
    - 入力
        - Entity
            - 内容
                - 中間形式である Entity の配列を受け取る
            - 形式
                - プログラムの内部構造体
            - データ詳細
                - 内部連携インターフェース【IF201】を参照
    - 出力
        - GeoJSON
            - 内容
                - GeoJSON 形式の JSON ファイルが出力される
            - 形式
                - JSON
            - データ詳細
                - ファイル出力インターフェース【IF107】を参照
81. 機能詳細
    - データ変換処理[GeoJSON]
        - 処理内容
            - 中間オブジェクトからジオメトリを取り出し GeoJSON 仕様のオブジェクトを作成する

- 主題属性と空間属性をそれぞれ properties・geometry に格納する
- 全ての中間オブジェクトを処理したら出力を行う

✧ ライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL116】を参照

18. 【FN018】データ変換処理[KML]

82. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、KML 形式のファイルを出力する

83. フローチャート

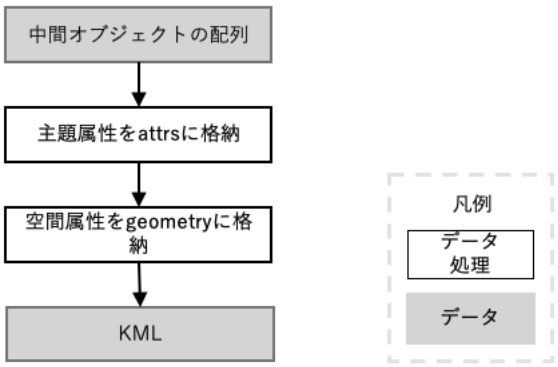


図 4-23 フローチャート

84. データ仕様

- 入力
  - ✧ Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - ✧ KML
    - 内容
      - KML 形式の XML ファイルが出力される
    - 形式
      - KML
    - データ詳細
      - ファイル出力インターフェース【IF108】を参照

85. 機能詳細

- データ変換処理[KML]
  - ✧ 処理内容
    - KML 仕様である Placemark の雛形を作成する

- 中間オブジェクトからジオメトリを取り出し主題属性と空間属性をそれぞれ Placemark の attrs や geometry に格納する
- 全ての中間オブジェクトを処理したら出力を行う

✧ ライブラリ

- 【SL003】【SL007】【SL008】を参照

✧ 利用するアルゴリズム

- 【AL117】を参照

19. 【FN019】データ変換処理[OBJ]

86. 機能概要

- FN001~FN007 までの処理が適切になされた中間オブジェクトの配列に対し、OBJ ファイル仕様に沿ったファイル群を出力する

87. フローチャート

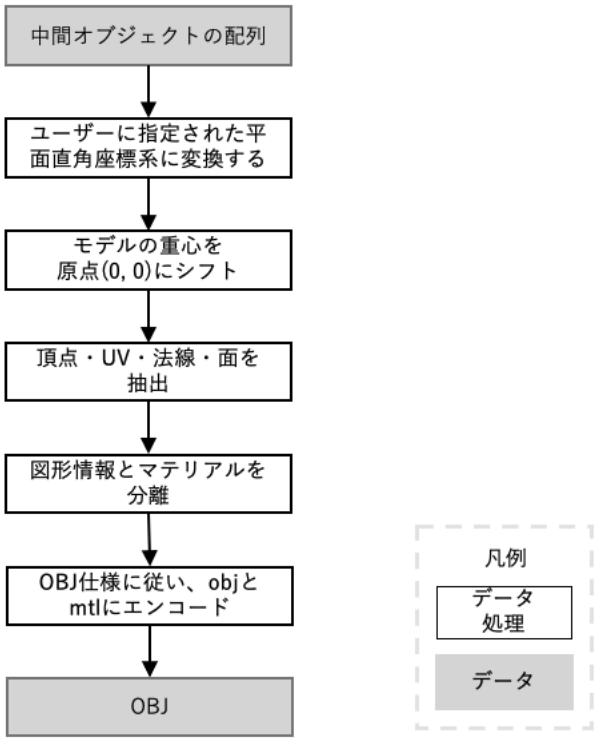


図 4-24 フローチャート

88. データ仕様

- 入力
  - Entity
    - 内容
      - 中間形式である Entity の配列を受け取る
    - 形式
      - プログラムの内部構造体
    - データ詳細
      - 内部連携インターフェース【IF201】を参照
- 出力
  - OBJ
    - 内容
      - OBJ 形式のテキストファイルが出力される
    - 形式

- OBJ
- データ詳細
  - ファイル出力インターフェース【IF109】を参照

89. 機能詳細

- データ変換処理[OBJ]
  - 処理内容
    - 全ての座標をユーザーに指定された平面直角座標系に変換する
    - モデルの重心を原点(0, 0)にシフト（オプション）
    - 中間オブジェクトからジオメトリを取り出し OBJ ファイル仕様に従い、頂点座標と UV・法線・面などを収集する
    - 図形情報とマテリアルの情報を分離する
    - 全ての中間オブジェクトを処理したら.obj ファイルと.mtl ファイル、利用するテクスチャ画像の出力を行う
  - ライブラリ
    - 【SL003】【SL007】【SL008】を参照
  - 利用するアルゴリズム
    - 【AL103】【AL118】を参照

20. 【FN020】ZIP ファイル解析

90. 機能概要

- 3D 都市モデル仕様に準拠した ZIP を解析し、対象となるファイルのパスを抽出する

91. フローチャート

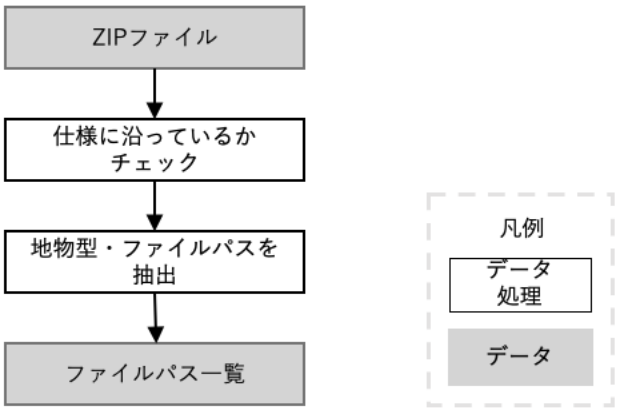


図 4-25 フローチャート

92. データ仕様

- 入力
  - 標準製品仕様書に乗っ取った構造の ZIP ファイルを受け取る
    - 内容
      - 標準製品仕様書に乗っ取った構造の ZIP ファイルを受け取る
    - 形式
      - ZIP ファイル
    - データ詳細
      - ファイル入力インターフェース【IF003】を参照
- 出力
  - 対象の CityGML のパスを示すテキストの一覧
    - 内容
      - 対象の CityGML のパスを示すテキストの一覧
    - 形式
      - 特になし
    - データ詳細
      - 特になし

93. 機能詳細

- ZIP ファイル解析
  - 処理内容
    - 3D 都市モデル仕様に沿っているか確認

- 指定された ZIP ファイルを解析し、含まれる地物型とファイルパスを識別する
- ファイルパスからメッシュコードを抽出する
- メッシュコードを一覧化し、地図 UI に渡す

✧ ライブラリ

- 【SL013】参照

✧ 利用するアルゴリズム

- 【AL122】を参照

21. 【FN021】地図上にメッシュ表示
94. 機能概要
    - 新規に実装される地図画面に、ZIP ファイルに格納されているファイルのメッシュのみを表示する機能
    - 【FN020】で生成されたファイルパス一覧を利用する
95. フローチャート

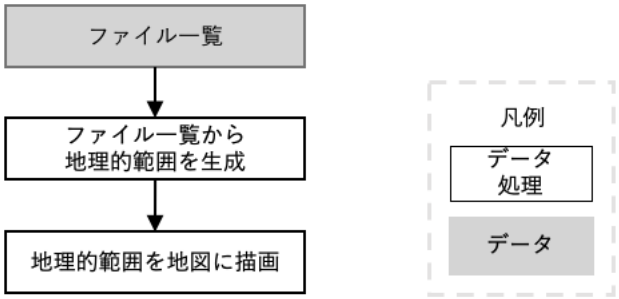


図 4-26 フローチャート

96. データ仕様
    - 入力
        - プログラム内部の、ファイル名を示すテキストの配列
            - 内容
                - プログラム内部の、ファイル名を示すテキストの配列
            - 形式
                - プログラムの内部変数
            - データ詳細
                - 特になし
    - 出力
        - 地理的範囲を地図上に描画
            - 内容
                - ファイル一覧から割り出された地理的範囲を地図上に描画
            - 形式
                - 特になし
            - データ詳細
                - 特になし
97. 機能詳細
    - 地図上にメッシュ表示
        - 処理内容
            - メッシュコード一覧を受け取る

- メッシュコードから地理的範囲を生成する
- 地図上にメッシュを表示
- メッシュがクリックされたら、変換可能な地物型を画面左のバーに表示
- 地物型と変換先データ形式等を指定し、変換開始

✧ ライブラリ

- 【SL002】【SL012】を参照

✧ 利用するアルゴリズム

- 【AL123】を参照

4-3. アルゴリズム

4-3-1. 利用したアルゴリズム

特殊な数値計算などが必要なコア部分に外部のアルゴリズムは利用していないため、割愛する。

4-3-2. 開発したアルゴリズム

開発したアルゴリズムの詳細について以下に記載する。

1) 【AL101】3D 都市モデル読み込み

- 本アルゴリズムを利用する機能
  - 【FN001】

- アルゴリズムの詳細

指定された 3D 都市モデルを読み込み、プログラムへ取り込むためのアルゴリズム。

1. 指定されたファイルパスから CityGML を読み込む
2. 読み込まれた CityGML はストリーミング処理され、地物ごとに「中間オブジェクト生成」アルゴリズムを利用し、主題属性・空間属性が整理される
3. 全ての地物に対して再帰的に処理を行い、全ての CityGML を読み取り次第、処理を完了する

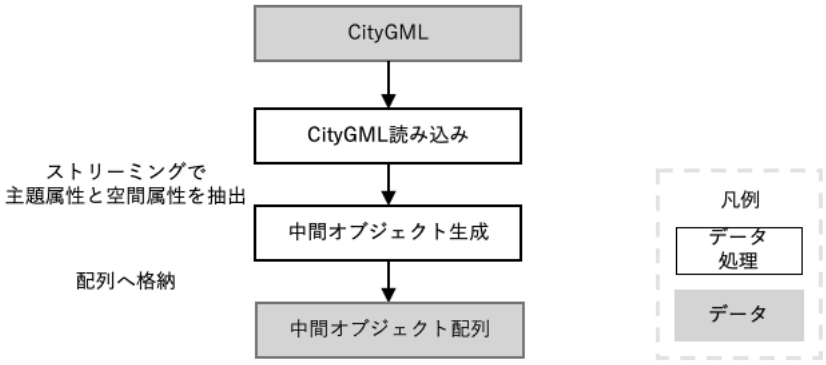


図 4-27 アルゴリズム詳細

2) 【AL102】中間オブジェクト生成

- 本アルゴリズムを利用する機能
  - 【FN002】

- アルゴリズムの詳細

3D 都市モデル内の属性・空間属性を抽出し、中間オブジェクトに変換するためのアルゴリズム

1. 「PLATEAU3D 都市モデル読み込み」アルゴリズムにより、渡された CityGML の情報を読み取る
2. 整理された情報地物型ごとにあらかじめ定義された形式に沿って、メモリ上で中間オブジェクト（クラス）へ変換される
3. 変換のたびに可変長配列にインサートする
4. 全ての地物に対して再帰的に処理を行い、全ての CityGML を読み取り次第、処理を完了する

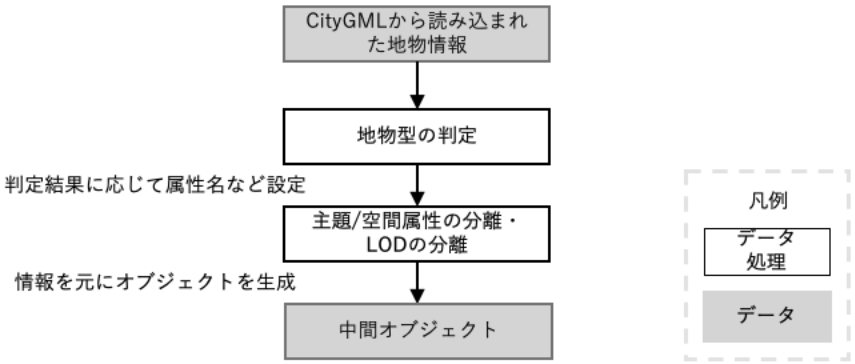


図 4-28 アルゴリズム詳細

3) 【AL103】投影変換

- 本アルゴリズムを利用する機能
  - 【FN003】

- アルゴリズムの詳細

中間オブジェクトに格納された座標情報を一括で変換するアルゴリズム

1. GUI 上からユーザーが変換先座標系の指定を行う
2. EPSG コードとして定義されている座標参照系の文字列と中間オブジェクトを受け取る
3. 以下のチェックを行う
   - 存在しない EPSG コードが指定されていないか
   - EPSG コードは出力ファイル形式に適用できる座標参照系かどうか
     - 例: MVT は EPSG:3857 にのみ対応・GeoJSON は仕様上 EPSG:4326 のみに対応、など
4. バリデーション後、規定の数値計算を行い、全ての座標を変換し、変換後の中間オブジェクトを作成する

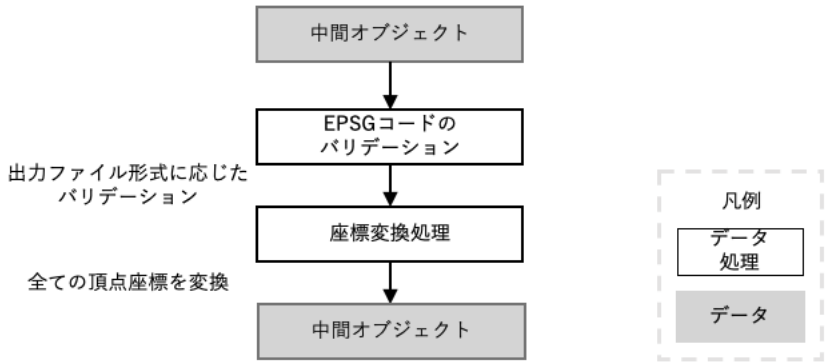


図 4-29 アルゴリズム詳細

4) 【AL104】属性マッピング

- 本アルゴリズムを利用する機能
  - 【FN004】

- アルゴリズムの詳細

中間オブジェクトをマッピングルール定義ファイルに記載の通り、属性名と型を変化させて、新たな中間オブジェクトを生成するアルゴリズム

1. GUI から指定されたマッピングルール定義ファイルを読み込む
2. 中間オブジェクトから指定された属性名を抽出し、定義の通り変更する
3. 全ての中間オブジェクトを走査し終えたら終了する

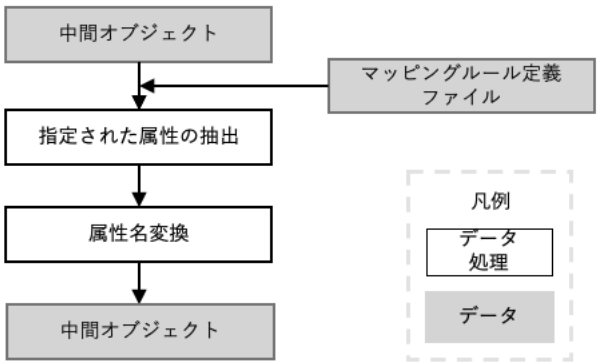


図 4-30 アルゴリズム詳細

5) 【AL105】属性ツリー変換

- 本アルゴリズムを利用する機能
  - 【FN005】

- アルゴリズムの詳細

中間オブジェクトに格納された、入れ子になった主題属性を抽出し、JSON 化・配列化・別テーブル化などを行うアルゴリズム

1. 出力ファイル形式に応じて入れ子属性の抽出方法を定める
2. 出力ファイル形式に応じて、抽出方法が適切かどうかバリデーションを行う
   - MVT や 3DTiles などは別テーブル化が不可、など
3. 抽出方法に応じて、処理を行う
   - JSON 化・配列化
     - 子の属性名と値を取り出し、JSON 化・配列化する
     - 親の属性値に JSON・配列を格納する
     - 全ての子要素がなくなるまで、再帰的に処理を行う
   - 別テーブル化
     - 子の属性をルート要素まで持ち上げ、フラットな属性ツリーに変化させる
     - 属性に応じてテーブルやファイルなどを分割して出力する
     - 全ての子要素がなくなるまで再帰的に処理を行う

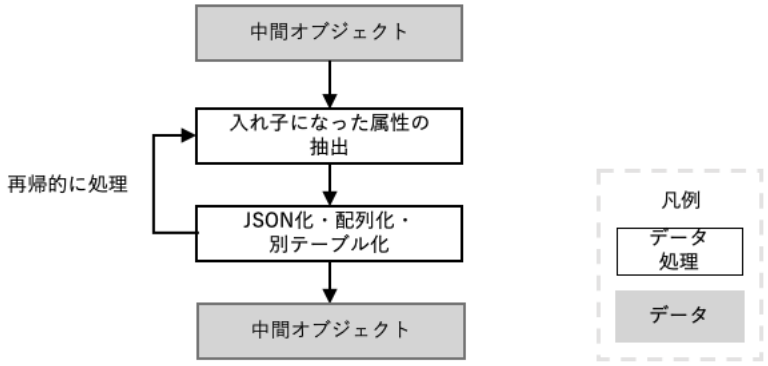


図 4-31 アルゴリズム詳細

6) 【AL106】LOD の抽出

- 本アルゴリズムを利用する機能
  - 【FN006】

- アルゴリズムの詳細

中間オブジェクトに格納された、空間属性を個別に取り出すもしくはフィルタリングするなどの処理を行い、別の中間オブジェクトを出力するアルゴリズム

1. GUI からの指定により、抽出する空間属性を決定する
2. 地物ごとの空間属性を全て取り出す
3. 指定されたパラメータに従い、空間属性のフィルタリングを行う
4. フィルタリングされた空間属性のみを次の処理に渡し、終了する

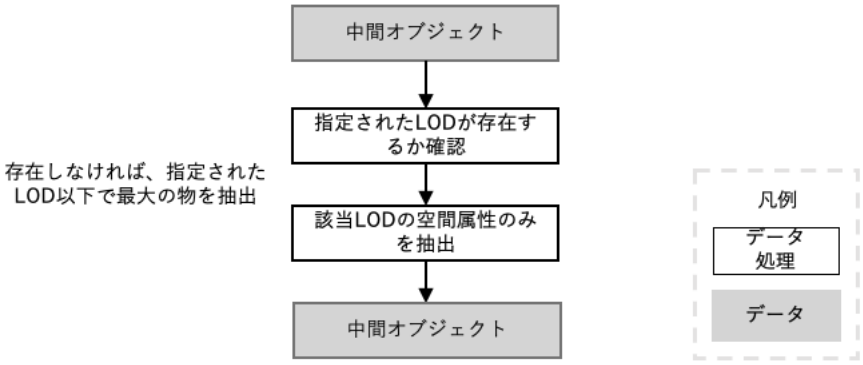


図 4-32 アルゴリズム詳細

7） 【AL107】アトラス作成

- 本アルゴリズムを利用する機能
  - 【FN007】

- アルゴリズムの詳細

CityGML に埋め込まれているテクスチャ画像をテクスチャアトラスにパッキングし、テクスチャアトラスの UV 座標を返すアルゴリズム

1. 面ごとのテクスチャのパス・UV 座標を受け取る
2. UV 座標に従って、テクスチャを切り抜く
3. 大きなテクスチャ（アトラス）を模したオブジェクトに仮想的に貼り付けていく
4. 全てのテクスチャを貼り付け終わったら処理を終了する

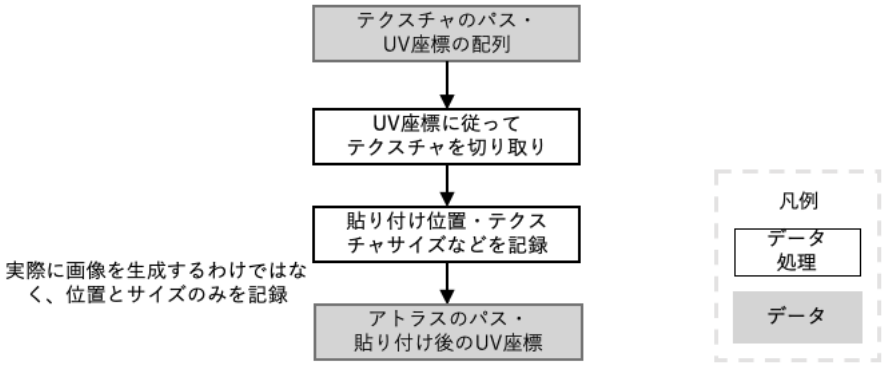


図 4-33 アルゴリズム詳細

8) 【AL108】地物の間引き

- 本アルゴリズムを利用する機能
  - 【FN008】

- アルゴリズムの詳細

geometricError × 係数（出力結果の見た目に応じて微調整）以下の大きさの地物をフィルタリングするアルゴリズム。地物ごとに boundingVolume を計算し 1 辺でも規定以上の大きさであれば抽出する

1. タイル座標から geometricError を算出する
2. 地物の boundingVolume を算出する
3. geometricError と事前に定めた係数を掛け合わせ、1 辺でも長さが上回れば抽出する

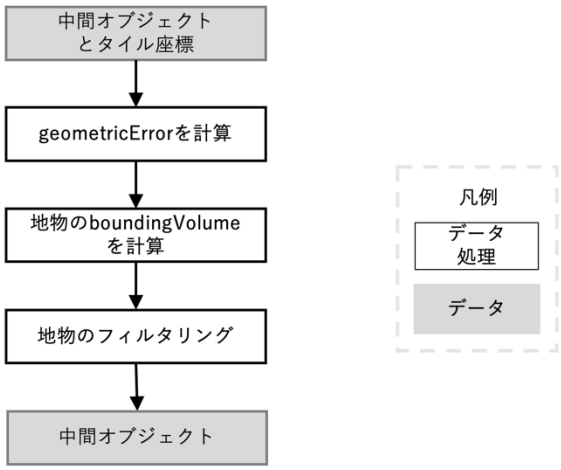


図 4-34 アルゴリズム詳細

9) 【AL109】解像度の変更

- 本アルゴリズムを利用する機能
  - 【FN009】

- アルゴリズムの詳細

geometricError に応じて、アトラス化時に解像度を下げるアルゴリズム。geometricError が上がれば上がるほど画面上の地物の大きさは小さくなるため、解像度を下げる。

1. タイル座標から geometricError を計算する
2. geometricError に応じて、低解像度化のための 0~1 の係数を算出する
3. 中間オブジェクトに係数を保持する

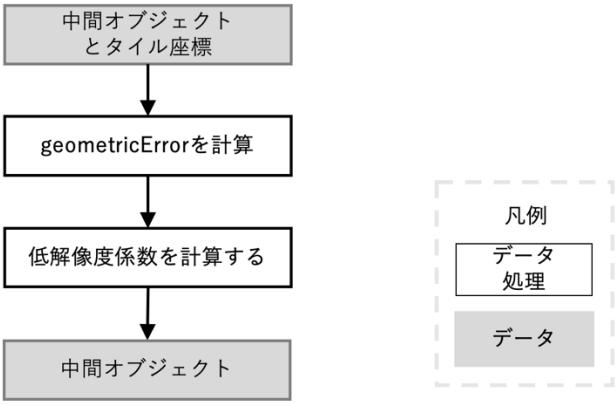


図 4-35 アルゴリズム詳細

10）【AL110】データ変換処理[3D Tiles]

- 本アルゴリズムを利用する機能
  - 【FN010】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、ジオメトリのタイル分割・glb 出力・tileset.json 出力処理を行うためのアルゴリズム

1. 変換対象地物が含まれる範囲をズームレベルごとにタイル分割する
2. glTF 仕様に沿って、タイルごとにジオメトリと属性情報を格納していく
3. tileset.json を出力する

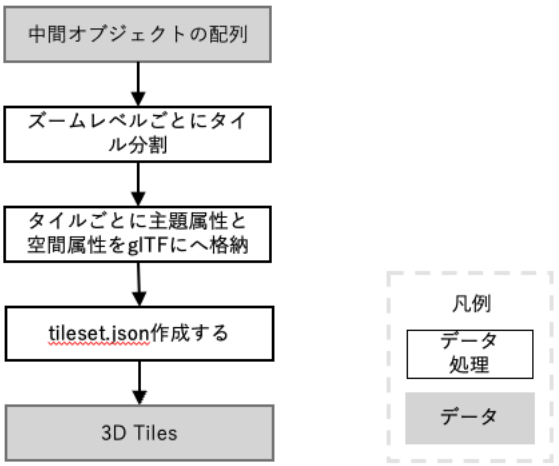


図 4-36 アルゴリズム詳細

11）【AL111】データ変換処理[MVT]

- 本アルゴリズムを利用する機能
  - 【FN011】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、ジオメトリのタイル分割・pbf 出力処理を行うためのアルゴリズム

1. 変換対象地物が含まれる範囲をズームレベルごとにタイル分割する
2. Mapbox Vector Tile 仕様に沿って、タイルごとにジオメトリと属性情報を格納していく
3. ズームレベルごとにフォルダを作成し、pbf ファイルを出力する

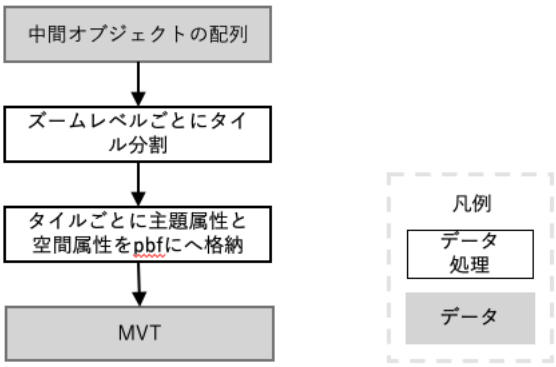


図 4-37 アルゴリズム詳細

12）【AL112】データ変換処理[GeoPackage]

- 本アルゴリズムを利用する機能
  - 【FN012】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、入れ子属性や異なる地物型を別テーブルにする処理を行い、複数テーブル保有した GeoPackage を出力するためのアルゴリズム

1. 入れ子属性や異なる地物型に応じてテーブルを作成する
2. ジオメトリと属性情報をテーブルごとに格納する

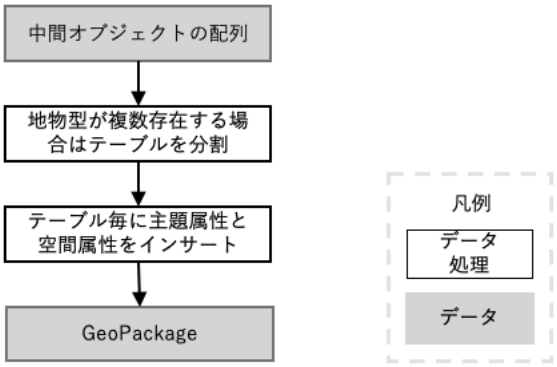


図 4-38 アルゴリズム詳細

13）【AL113】データ変換処理[glTF]

- 本アルゴリズムを利用する機能
  - 【FN013】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、属性情報が付与された状態で glb 出力処理を行うためのアルゴリズム

1. 全ての座標を ENU 座標系に変換する
2. glTF 仕様に従って属性情報をエンコードする
3. glTF 仕様に従って頂点情報を格納する
4. ファイルの出力を行う

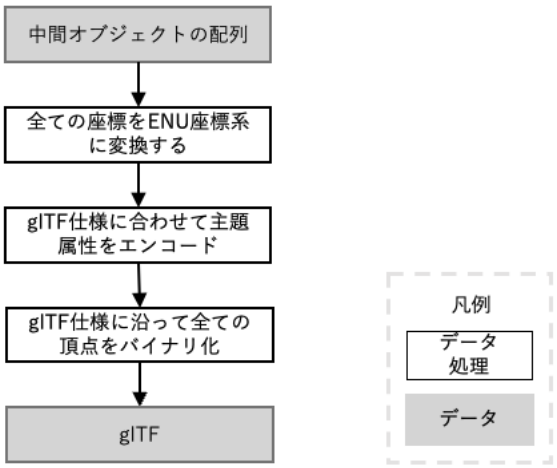


図 4-39 アルゴリズム詳細

14）【AL114】データ変換処理[Minecraft]

- 本アルゴリズムを利用する機能
  - 【FN014】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、Minecraft で利用可能な Anvil 形式のファイルを出力するためのアルゴリズム

1. 全ての座標を、変換対象範囲の重心点を原点とした横メルカトル座標に変換する
2. マインクラフトで利用される Anvil 形式でファイルを出力する
3. メタデータとなる level.dat ファイルを出力する

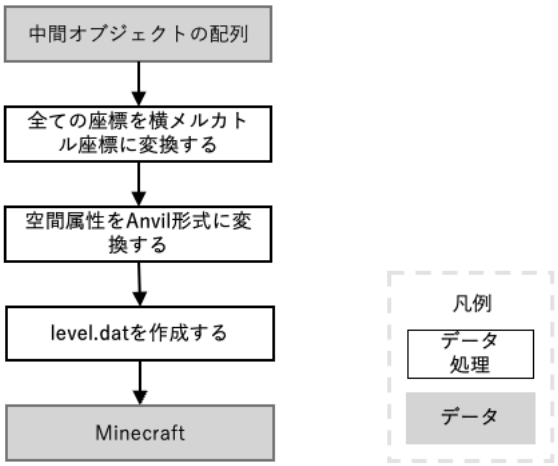


図 4-40 アルゴリズム詳細

15）【AL115】データ変換処理[CZML]

- 本アルゴリズムを利用する機能
  - 【FN015】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、CZML 形式のファイルを出力するアルゴリズム

1. 中間オブジェクトからジオメトリを取り出し CZML 仕様である Packet 形式に格納する
2. Packet の description 属性に、中間オブジェクトの主題属性を格納する
3. 全ての中間オブジェクトを処理したら出力を行う

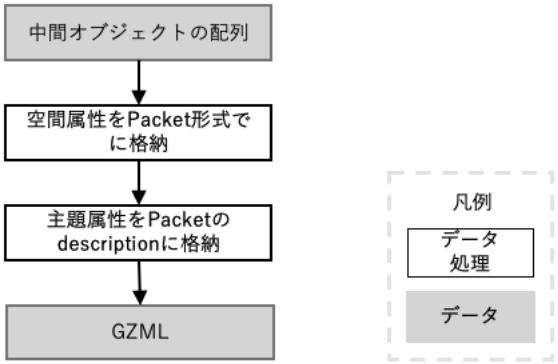


図 4-41 アルゴリズム詳細

16）【AL116】データ変換処理[GeoJSON]

- 本アルゴリズムを利用する機能
  - 【FN016】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、GeoJSON 形式のファイルを出力するアルゴリズム

1. 中間オブジェクトからジオメトリを取り出し GeoJSON 仕様のオブジェクトを作成する
2. 主題属性と空間属性をそれぞれ properties・geometry に格納する
3. 全ての中間オブジェクトを処理したら出力を行う

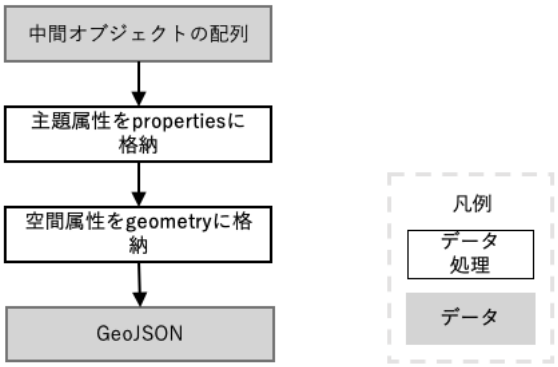


図 4-42 アルゴリズム詳細

17）【AL117】データ変換処理[KML]

- 本アルゴリズムを利用する機能
  - 【FN017】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、KML 形式のファイルを出力するアルゴリズム

1. KML 仕様である Placemark の雛形を作成する
2. 中間オブジェクトからジオメトリを取り出し主題属性と空間属性をそれぞれ Placemark の attrs や geometry に格納する
3. 全ての中間オブジェクトを処理したら出力を行う

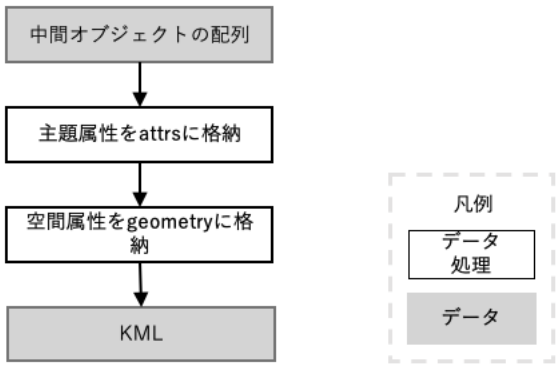


図 4-43 アルゴリズム詳細

18）【AL118】データ変換処理[OBJ]

- 本アルゴリズムを利用する機能
  - 【FN018】

- アルゴリズムの詳細

FN001~FN009 までの処理が適切になされた中間オブジェクトの配列に対し、OBJ ファイル仕様に沿ったファイル群を出力するためのアルゴリズム

1. 中間オブジェクトからジオメトリを取り出し OBJ ファイル仕様に従い、頂点座標と UV・法線・面などを収集する
2. 図形情報とマテリアルの情報を分離する
3. 全ての中間オブジェクトを処理したら.obj ファイルと.mtl ファイル、利用するテクスチャ画像の出力を行う

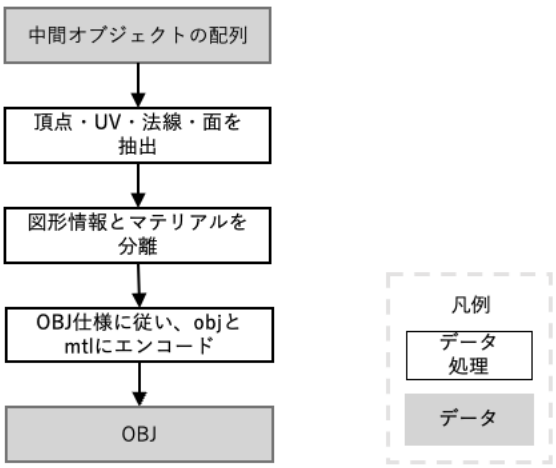


図 4-44 アルゴリズム詳細

19）【AL119】タイル分割

- 本アルゴリズムを利用する機能
  - 【FN011】

- アルゴリズムの詳細

2 次元地図で一般的に利用されるラスタータイルやベクタータイルの領域と同様の範囲でタイル分割を行う。すべての地物を取り出し、外部から与えられる最小・最大ズームレベルに応じて分割されたタイル領域に格納する。タイル境界にある地物は鉛直方向に切り、分割する。

1. 地物と最小・最大ズームレベルを受け取る
2. 個別に地物の中心座標とおおよそのサイズを計算する
3. 指定された最小・最大ズームレベルごとに、必要に応じて地物を分割し、タイル座標ごとに地物を収集する

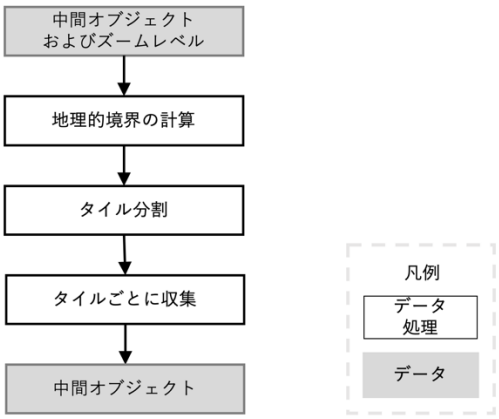


図 4-45 アルゴリズム詳細

20）【AL120】geometricError の算出

- 本アルゴリズムを利用する機能
  - 【FN007】【FN008】【FN010】

- アルゴリズムの詳細

一般に、ズームレベルを上げると、geometricError は減少する。ズームレベル 22 程度で geometricError が 0 程度となるように基準となる定数を設定し、ズームレベルごとに geometricError の計算を行う。また緯度が高くなり極に近づくとタイルの物理的なサイズが小さくなるため、合わせて geometricError を減少させる。

1. タイル座標を受け取る
2. タイル座標から経緯度を算出する
3. ズームレベルに基づく geometricError の計算
4. 緯度に基づく geometricError の計算

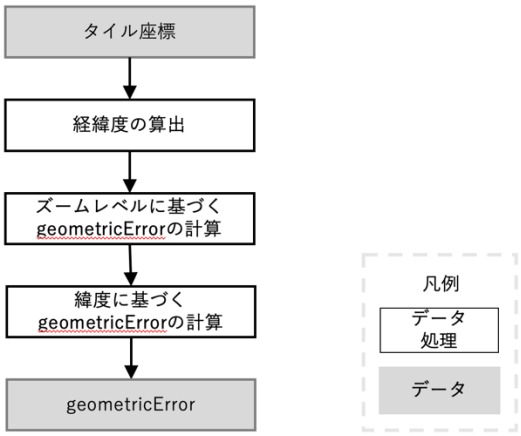


図 4-46 アルゴリズム詳細

21）【AL121】LOD 最適化

- 本アルゴリズムを利用する機能
  - 【FN007】

- アルゴリズムの詳細

geometricError に応じて、3D Tiles の各タイルに格納する地物詳細度を変更する。

1. 中間オブジェクトとタイル座標を受けとる
2. タイル座標から geometricError を計算する
3. geometricError が 150 以上であれば最低 LOD を抽出し、geometircError 半分になれば次の LOD を抽出する、といったように geometricError に応じて LOD をフィルタリングする
4. フィルタリングされた中間オブジェクトを返却する

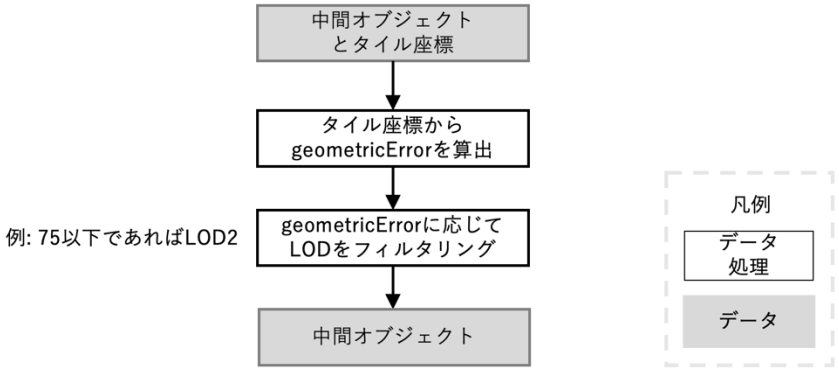


図 4-47 アルゴリズム詳細

22）【AL122】ZIP ファイル解析

- 本アルゴリズムを利用する機能
  - 【FN020】

- アルゴリズムの詳細

指定された ZIP ファイルを解析し、3D 都市モデル仕様に準拠しているか判別する。問題なければファイル構造を解析し、メッシュコード・地物型・ファイルパスを抽出する。

1. 3D 都市モデル仕様に沿っているか確認
2. 指定された ZIP ファイルを解析し、含まれる地物型とファイルパスを識別する
3. ファイルパスからメッシュコードを抽出する
4. メッシュコードを一覧化する

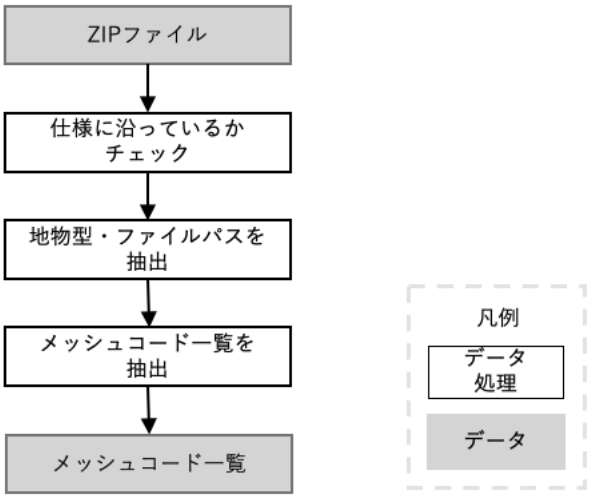


図 4-48 アルゴリズム詳細

23）【AL123】メッシュコードから矩形を作成

- 本アルゴリズムを利用する機能
  - 【FN021】

- アルゴリズムの詳細

ZIP ファイルに格納されているファイルのメッシュのみを表示し、対象選択・データ変換などを可能にする機能

1. メッシュコード一覧を受け取る
2. メッシュコードを 1 つ取り出す
3. コードから南西端の経緯度を計算
4. 分割に応じて矩形ポリゴンを生成
5. 全ての範囲の矩形を生成したら、GeoJSON を生成

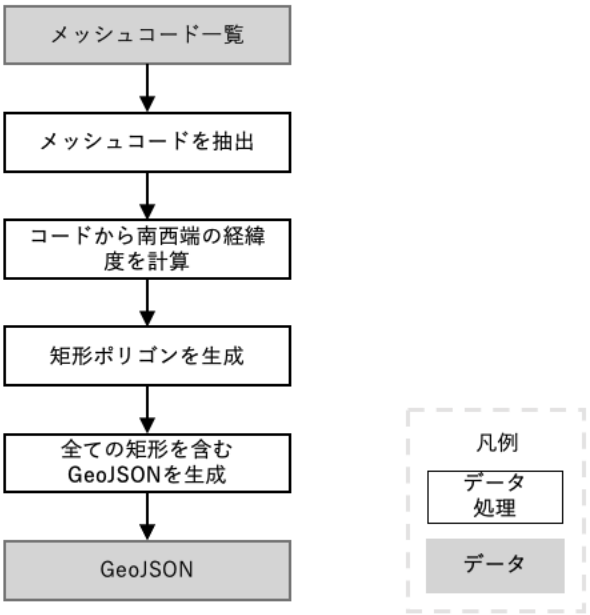


図 4-49 アルゴリズム詳細

4-4. データインタフェース

4-4-1. ファイル入力インタフェース

1) 【IF001】3D 都市モデル

- 本インタフェースを利用する機能：【FN001】

「3D 都市モデル標準製品仕様書 第 4.0 版」に示された形式の CityGML

2) 【IF002】マッピングルール定義

- 本インタフェースを利用する機能：【FN004】

```
{
  "rename": {
    "uro:buildingIDAttribute": "建物 ID",
    "bldg:address": "住所",
    "uro:buildingDataQualityAttribute": "データ品質",
    "uro:buildingDetailAttribute": "建物詳細",
    "gen:genericAttribute": "ジェネリック",
    "bldg:measuredHeight": "高さ",
    "uro:buildingDisasterRiskAttribute": "災害リスク",
    "gml:name": "名前"
  }
}
```

4-4-2. ファイル出力インタフェース

1) 【IF101】3D Tiles

v1.1 で尚且つ属性情報が付与されたテクスチャ付きの 3D Tiles を出力する

- 本インターフェースを利用した機能
  - 【FN011】

2) 【IF102】MVT

.pbf 形式の MVT を出力する

- 本インターフェースを利用した機能
  - 【FN012】

3) 【IF103】GeoPackage

必要に応じてテーブルが分割された GeoPackage を出力する

- 本インターフェースを利用した機能
  - 【FN013】

4) 【IF104】glTF

属性情報が付与された glTF を出力する

- 本インターフェースを利用した機能
  - 【FN014】

5) 【IF105】Minecraft

Anvil と呼ばれる形式のワールドデータおよび level.dat ファイルを出力する

- 本インターフェースを利用した機能
  - 【FN015】

6) 【IF106】CZML

CZML 仕様に沿った JSON ファイルを出力する

- 本インターフェースを利用した機能
  - 【FN016】

7) 【IF107】GeoJSON

GeoJSON 仕様に沿った JSON ファイルを出力する

- 本インターフェースを利用した機能
  - 【FN017】

8) 【IF108】KML

KML 仕様に沿った XML ファイルを出力する

- 本インターフェースを利用した機能
  - 【FN018】

9) 【IF109】OBJ

OBJ 形式の仕様に沿ったテキストファイルを出力する

- 本インターフェースを利用した機能
  - 【FN019】

4-4-3. 内部連携インタフェース

CityGML を内部で扱うため、ファイル出力を伴わずメモリ上で処理できる「Entity」オブジェクトを定義し、内部連携インタフェースとして利用する。

10) 【IF201】Entity

- 説明文
- 表形式で例を以下に示す（実態は Rust の構造体）

カラム名	root	geometry_store	appearance_store
説明	属性ツリーを示す Object オブジェクト	ジオメトリを格納するための GeometryStore オブジェクト	テクスチャ情報を格納するための AppearanceStore オブジェクト

1) GeometryStore

- 説明文
- 表形式で例を以下に示す（実態は Rust の構造体）

カラム名	epsg	vertices	indices	textures	uvs
説明	EPSG コード	頂点座標の配列	頂点インデックスの配列	AppearanceStore に格納されている textures のインデックスの配列	UV 座標の配列

2) AppearanceStore

- 説明文
- 表形式で例を以下に示す（実態は Rust の構造体）

カラム名	textures	materials
説明	テクスチャパスの配列	面のカラーなどのマテリアル情報

3) Object

- 説明文
- 表形式で例を以下に示す（実態は Rust の構造体）

カラム名	Typename	stereotype	attributes
説明	地物型の名称	データの種類（空間属性を持つ Feature か、空間属性を持たない Object）	属性名をキーとし、属性情報を値としてもつ Map

4-4-4. 外部連携インタフェース

外部のサービスは利用せず、全てシステム内部で完結するため割愛する。

4-5. 検証に用いたデータ

4-5-1. 利用したデーター覧

本システムでは、原則全ての 3D 都市モデルが利用可能なため、割愛する。

4-5-2. 生成・変換するデータ

本システムでは、外部で生成・変換したデータなどは利用しないため、割愛する。

4-6. ユーザーインタフェース

4-6-1. 画面一覧

1） メイン画面

表 4-4 PC 画面一覧

ID	連携（ID）	画面名	画面説明	画面を表示した機能（ID）
SC001	-	メイン画面	● データの読み込み・設定変更・出力ファイル指定やデータ変換などを行うための画面	FN001 ・ FN004 ・ FN010~FN020
SC002	-	地図画面	● データの読み込み・設定変更・出力ファイル指定やデータ変換などを行うための地図画面	FN001 ・ FN004 ・ FN010~FN021

4-6-2. 画面遷移図

本システムはメイン画面と地図画面を行き来するシンプルな遷移のみを行う。

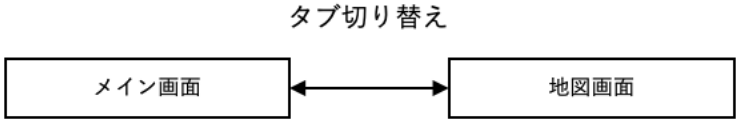


図 4-50 画面遷移図

4-6-3. 各画面仕様詳細

1. 【SC001】メイン画面

- 画面の目的・概要
  - ➢ データの読み込み・設定変更・出力ファイル指定やデータ変換などを行うための画面。

- 画面イメージ

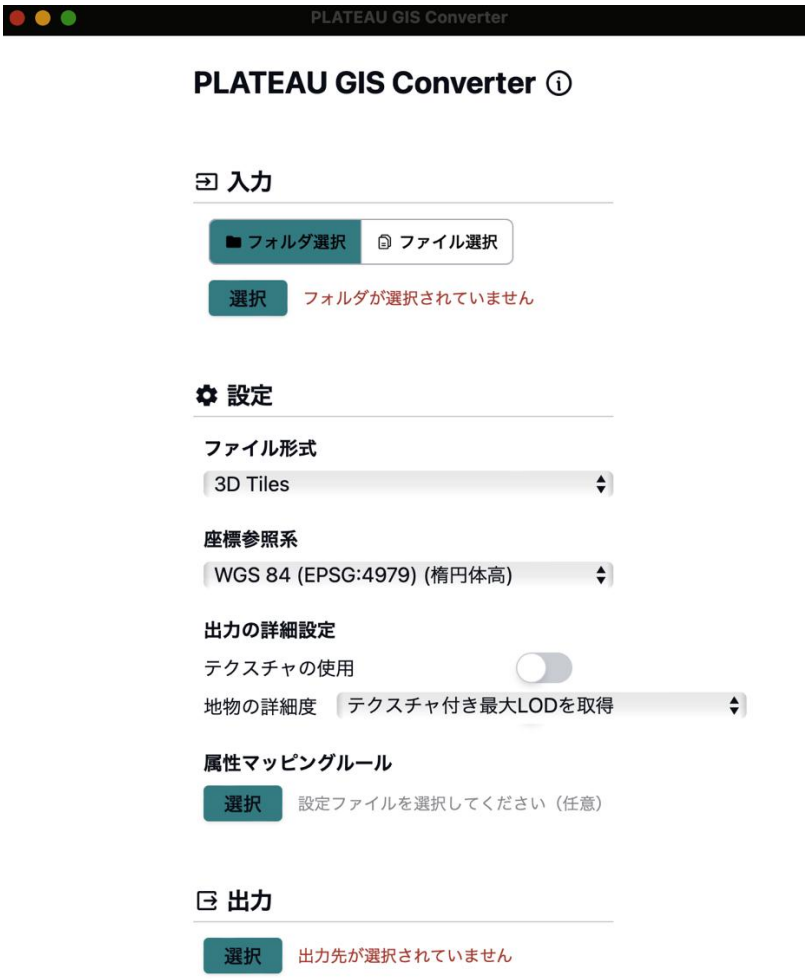


図 4-51 メイン画面のイメージ

2. 【SC002】地図画面

- 画面の目的・概要
  - ZIP ファイルの読み込み・地図表示・メッシュ選択・設定変更・出力ファイル指定やデータ変換などを行うための画面。

- 画面イメージ

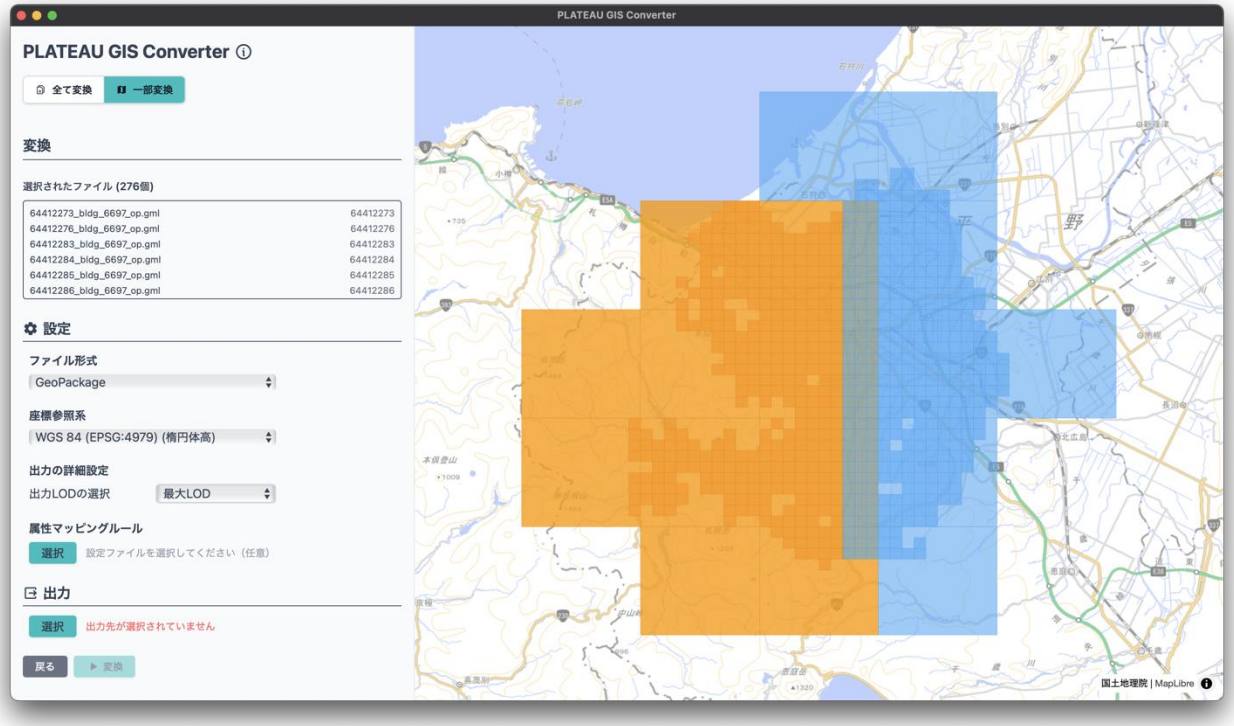


図 4-52 メイン画面のイメージ

4-7. システムの利用手順

4-7-1. システムの利用フロー

本システムは変換対象の CityGML を選択し、簡単に設定の変更を行なったのちにファイルを出力できるだけの非常にシンプルな機能を有している。

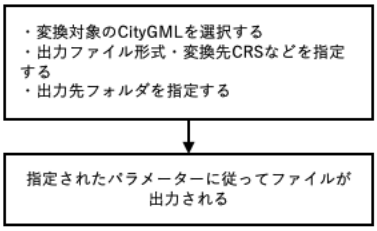


図 4-53 システムの利用フロー

4-7-2. 各画面操作方法

操作方法は下記の通り、ユーザーマニュアルを整備しているため割愛する。

https://mierune.github.io/plateau-gis-converter/

5. システムの非機能要件

5-1. 社会実装に向けた非機能要件

本システムは 3D 都市モデル利用促進のためのツールであることから、特にユーザーの使いやすさに配慮した設計が必要である。このため「使いやすさ」や「業務への適用性」について、PLATEAU の Slack やコンソーシアム参加メンバーから、3D 都市モデルを熟知した航測会社スタッフや地方公共団体向けの業務を行っている事業者など、想定されるユーザーからヒアリングやアンケートを行い、その有用性を評価する必要がある。また、テクスチャ最適化に伴いパフォーマンスが向上したかどうかを定量的に計測する必要がある。

表 5-1：非機能要件一覧

カテゴリ	ID	項目	詳細
ユーザビリティ	NR001	UI の使いやすさ	● 改修された UI が直感的に理解できるか、不明な操作がないか、などをヒアリング・アンケートで評価
	NR002	変換されたデータの使いやすさ	● 変換されたデータ形式の利用しやすさについてヒアリング・アンケートで評価
	NR003	データの正しさ、業務への適用	● データが分析などの実務で利用できるようになっているかヒアリング・アンケートで評価
	NR004	データ利用までの手間・時間の短縮	● 1 度の変換に何割程度、作業時間が短縮されたかをヒアリング・アンケートで評価

1) 【NR001】UI の使いやすさ

- 本非機能要件を適用するシステム
  - 全て
- 目標値
  - アンケートにより、高い評価を得る
- 設定理由
  - 3D 都市モデル利用促進ツールという側面があるため、ユーザーの「使いやすさ」が重要である
  - 特に、新設された地図画面の使いやすさは今年度業務の評価に直結する
- 評価方法
  - 5 段階評価でのアンケートを実施する

2) 【NR002】変換されたデータの使いやすさ

- 本非機能要件を適用するシステム
  - 全て
- 目標値
  - アンケートにより、高い評価を得る
- 設定理由

- データ変換ツールである以上、データの使いやすさが重要である

- 評価方法
  - 5 段階評価でのアンケートを実施する

3) 【NR003】データの正しさ、業務への適用

- 本非機能要件を適用するシステム
  - 全て
- 目標値
  - アンケートにより、高い評価を得る
- 設定理由
  - 昨年度アンケートにより、一部データ形式で不備があるのではないかという指摘を受けたため、それが改善されたかどうかは重要である
- 評価方法
  - 5 段階評価でのアンケートを実施する

4) 【NR004】データ利用までの手間・時間の短縮

- 本非機能要件を適用するシステム
  - 全て
- 目標値
  - アンケートにより、高い評価を得る
- 設定理由
  - 今年度業務の大きなテーマとしてデータ利活用促進のために利用ハードルを下げるのが目標であるため、手間や時間が短縮されたかどうかは本年度業務の評価に直結する
- 評価方法
  - 5 段階評価でのアンケートを実施する

6. 品質

6-1. 機能要件の品質担保

表 6-1：機能要件の品質担保方針

対象プロセス/サブシステム	品質評価項目	目標値	期間の単位	アクティビティ
地図機能・ZIP ファイル解析	入出力対応	● 入出力の一致	2025 年 12 月～2026 年 1 月	● 統合テストによる検証
OBJ ファイルの座標系対応	入出力対応	● 入出力の一致	2025 年 12 月～2026 年 1 月	● 統合テストによる検証
GIS ファイルの修正対応	入出力対応	● 入出力の一致	2025 年 12 月～2026 年 1 月	● 単体テストによる検証

6-1-1. 地図機能・ZIP ファイル解析の検証

利便性向上のため、ZIP ファイルを読み込み地図上で変換対象メッシュを指定可能となる機能を実装した。ユーザーが指定したメッシュ・地物型のデータを適切に取り込むことができるかどうか、またそのデータが、ZIP ではなく直接ファイルを指定する既存機能と比較して同一のデータが生成されるかが検証のポイントとなる。

検証対象のデータとして、千代田区の CityGML（メッシュ番号: 53394621）のうち、建築物のデータを対象とした。

（13101_chiyoda-ku_pref_2023_citygml_2_op/udx/bldg/53394621_bldg_6697_op.gml）

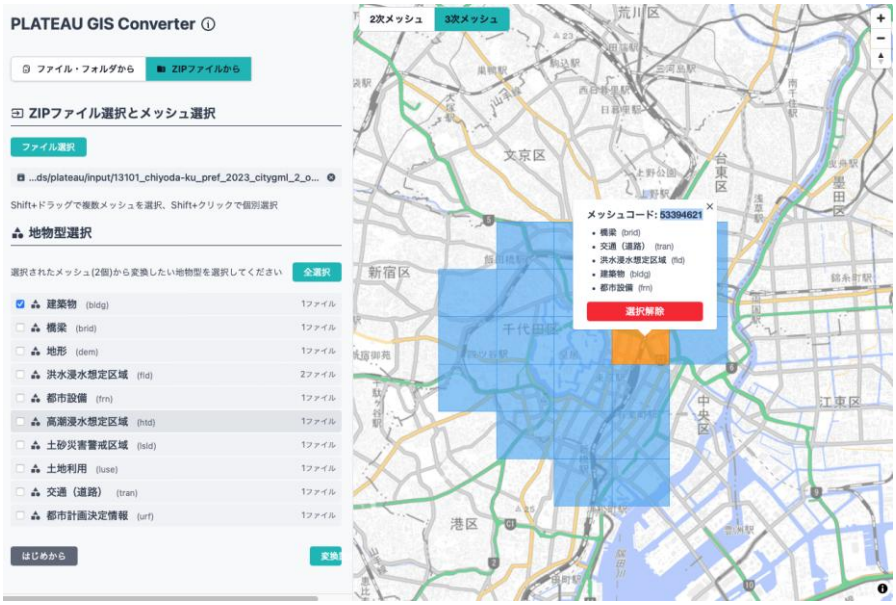


図 6-1 ZIP 取り込み・地図表示機能

6-1-1-a. 実装の概要

ZIP ファイルと通常ファイルを統一的に読み取るインターフェースを新たに実装し、ファイルパスに ZIP アーカイブが含まれるかどうかを自動判定する仕組みとした。ZIP の場合はアーカイブ内のファイルをメモリ上に展開して読み取るため、呼び出し側はファイルソースの違いを意識する必要がない。

これに伴い、CityGML の読み込み処理、コードリストの解決処理、テクスチャパスの変換処理をそれぞれ ZIP パスに対応させた。

6-1-1-b. テスト結果

ZIP から CityGML データを読み込み、通常ファイルからの読み込みと同一のオブジェクト数・属性値が得られることを確認するテスト 2 件を実施した。

また、3D Tiles, GeoJSON, MVT, GeoPackage, Shapefile, OBJ 等の全 13 種の出力フォーマットについて統合テストを実施し、ZIP から読み込んだデータが全フォーマットで正常に変換されることを確認した。

6-1-1-c. 変換ファイルの実データ検証

上記テストに加え、実際の千代田区建物データを用いて、通常ファイル経由の変換結果と ZIP ファイル経由の変換結果の GeoPackage を比較した。

両ファイルに含まれるテーブル構成（10 テーブル）は完全に一致した。地物数は合計 18,352 レコードで全テーブルにおいて一致し、属性フィールドの名称・型および全レコードの属性値も完全に一致したことを確認した。

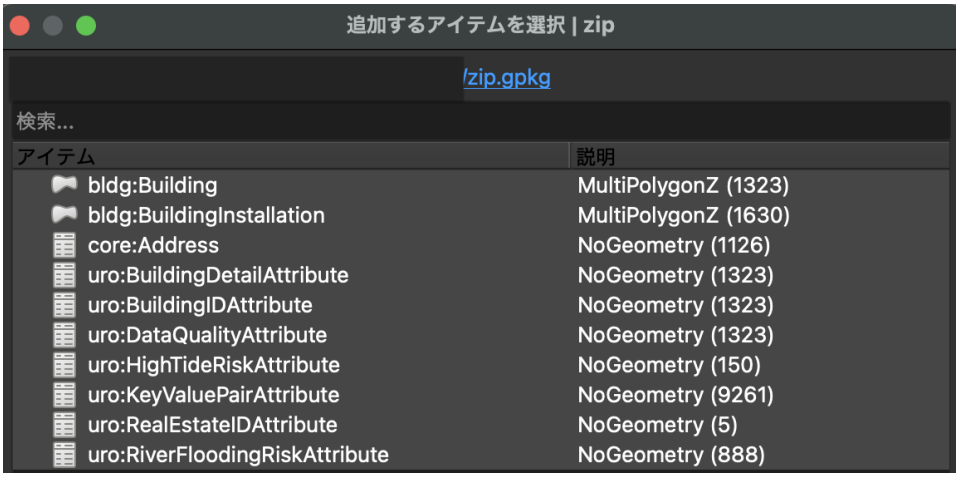


図 6-2 ZIP 形式での読み取り

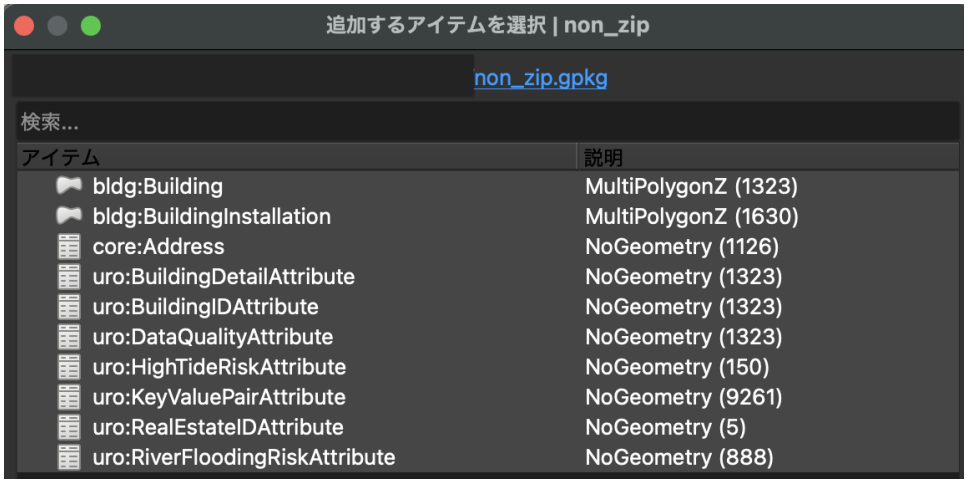


図 6-3 CityGML の直接読み取り（既存機能）

6-1-2. OBJ ファイルの座標系対応の検証

6-1-2-a. 概要

建築・土木分野での利用を想定し、OBJ 出力時に平面直角座標系で座標値を出力する機能を新たに実装した。従来はモデルの重心を原点に移動して出力する方式のみであったが、平面直角座標系の座標値をそのまま保持して出力できるようになった。

CityGML の地理座標系（JGD2011、緯度経度）から平面直角座標系（メートル単位）への投影変換が正しく行われ、出力座標値が地理的に妥当であるかが検証のポイントとなる。あわせて、既存の原点移動モードが引き続き正しく動作することも確認する。

検証対象のデータとして、千代田区の CityGML（メッシュ番号: 53394621）のうち、建築物のデータを対象とした。

（13101_chiyoda-ku_pref_2023_citygml_2_op/udx/bldg/53394621_bldg_6697_op.gml）

6-1-2-b. 実装の概要

OBJ 出力に平面直角座標系（EPSG:6669-6687, EPSG:10162-10174）への投影変換機能を追加した。

「モデルの重心を原点に設定」オプション（デフォルト有効）を無効にした場合、平面直角座標系の座標値が出力される。

6-1-2-c. テスト結果

平面直角座標系（EPSG: 10170）を指定し、いくつかの CityGML ファイルを OBJ 形式に変換する統合テストを実施した。各地物タイプについて OBJ ファイルが正常に生成されること、および平面直角座標系のサポート範囲外の座標系が指定された場合にエラーとなることを確認した。

6-1-2-d. 出力座標値の実データ検証

千代田区の建物データを用いて、原点移動モードと平面直角座標系モードの 2 つで OBJ 出力の座標値を検証した。いずれのモードでも頂点数は 1,199,829、面数は 399,943 で一致しており、モードの違いがメッシュ構

造に影響しないことを確認した。

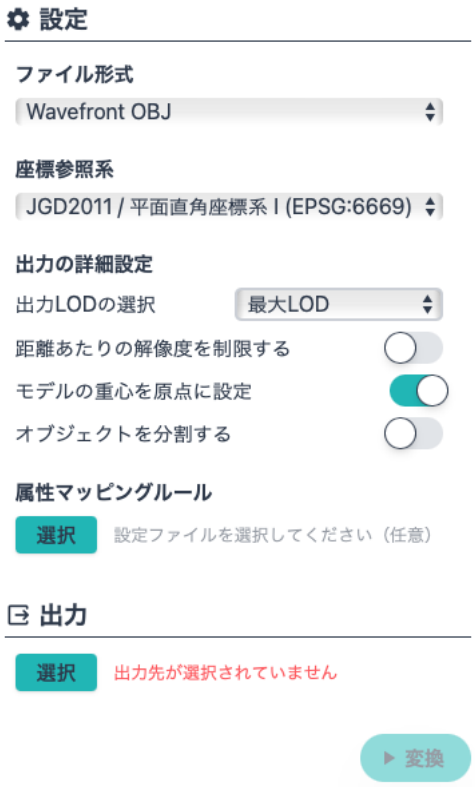


図 6-4 OBJ 形式への変換

1. 原点移動モード

バウンディングボックスの中心は(0.000, 0.000, 0.000)であり、原点に完全に一致した。座標レンジは X 方向が約 1,230m、Z 方向が約 1,075m（いずれも水平面）、Y 方向が約 219m（高さ方向、建物高さ最大約 109m）であり、既存の原点移動機能が正しく動作していることを確認した。

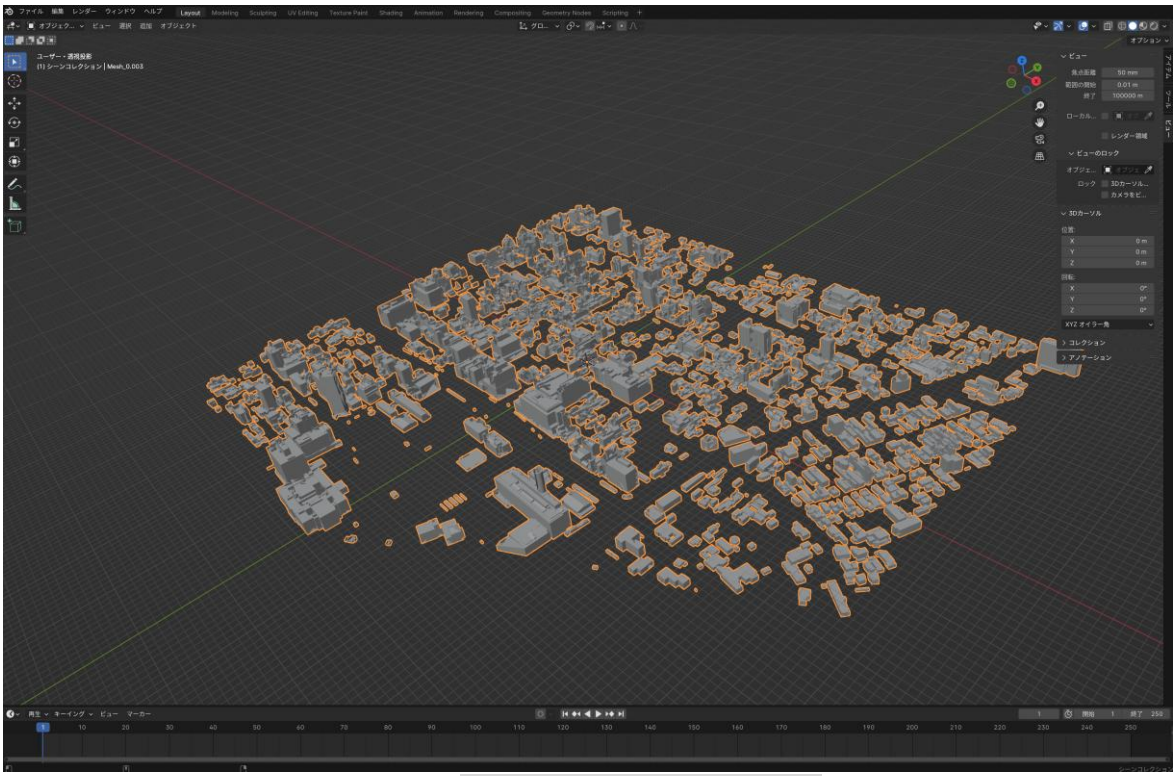


図 6-5 原点を中心にした変換

2. 平面直角座標系モード（EPSG:10170）

平面直角座標系 IX 系の原点（北緯 36 度、東経 139 度 50 分）に対して、出力座標値は南北方向が原点から南約 34～35km、東西方向が原点から西約 5～6km、高さ方向が-3m～216m であった。これは千代田区の地理的位置および建物の高さとして妥当な値であり、投影変換が正しく行われていることを確認した。

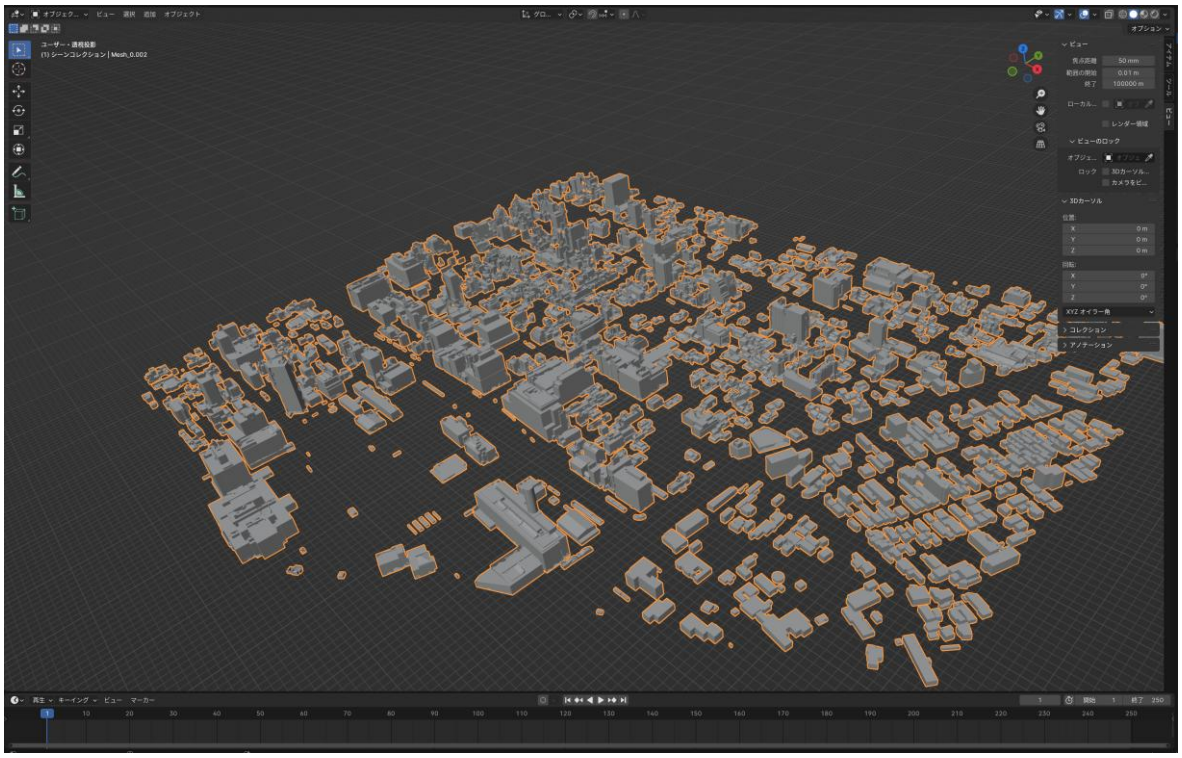


図 6-6 平面直角座標系での変換

6-1-3. GIS ファイルの修正対応の検証

6-1-3-a. 概要

GIS ファイル出力に関して、Shapefile の属性情報欠落、GeoPackage の 3 次元 CRS 定義不足、配列属性値の出力形式という 3 件の不具合・改善に対応した。

各修正が意図通りに動作し、既存の変換処理に影響を与えていないことが検証のポイントとなる。

6-1-3-b. Shapefile 出力時の属性情報欠落の修正

Shapefile 出力において、shp/shx ファイルのヘッダに記録するファイルサイズとレコードサイズの計算に誤りがあり、GIS ソフトウェアで属性情報（dbf）が正しく読み込めない不具合が発生していた。ヘッダ・レコードサイズの定数を Shapefile 仕様に準拠するよう修正した。

Shapefile 出力の単体テスト（ジオメトリ付き地物の出力確認）と統合テスト（CityGML から Shapefile への変換全体の確認）を実施した。

6-1-4. GeoPackage 出力時の 3 次元 CRS 定義の追加

GeoPackage 出力において、高さ情報を含む 3 次元の複合 CRS 定義が含まれておらず、3 次元データの変換時に適切な CRS が割り当てられない不具合があった。JGD2011 系の複合 CRS 定義（EPSG:6697、EPSG:10162～10174、EPSG:6669～6687）を追加した。

GeoPackage のジオメトリ処理テストと変換パイプライン統合テストを実施した。

6-1-4-a. 配列属性値のカンマ区切り文字列出力オプションの追加

CityGML の配列型属性値が JSON 文字列としてそのまま出力されており、GIS ソフトウェアでの利便性が低いという問題があった。配列型の属性値をカンマ区切り文字列に変換するオプションを追加した。スカラー型の要素のみ変換対象とし、ネストした配列やオブジェクトは JSON 文字列にフォールバックする仕様とした。

単一/複数文字列の変換、空配列、型混在の変換、非スカラー型の除外、フォールバック動作、スキーマ更新の各パターンについて単体テストを実施した。

6-2. 非機能要件の品質担保

表 6-2：機能要件の品質担保方針

対象項目	品質評価項目	目標値	期間の単位	アクティビティ
システム全般	UI の使いやすさ	● 平均 4 以上の評価を目標とする	2025 年 12 月～2026 年 1 月	● アンケートによる検証
	変換されたデータの使いやすさ	● 平均 4 以上の評価を目標とする	2025 年 12 月～2026 年 1 月	● アンケートによる検証
	データの正しさ、業務への適用	● 平均 4 以上の評価を目標とする	2025 年 12 月～2026 年 1 月	● アンケートによる検証
	データ利用までの手間・時間の短縮	● 平均 4 以上の評価を目標とする	2025 年 12 月～2026 年 1 月	● アンケートによる検証

今回、2026 年 1 月 19 日から 31 日にかけて、業務で GIS や 3D データを扱う企業所属の対象にアンケートを実施し、7 名からの回答を得た。以下の 4 つの評価基準を設け、いずれも 5 段階評価で平均 4 以上を目標とした。

1. UI の使いやすさ
2. 変換されたデータの使いやすさ
3. データの正しさ、業務への適用
4. データ利用までの手間・時間の短縮

6-2-1. 評価結果

6-2-1-a. UI の使いやすさ

UI 全般の直感的なわかりやすさ（Q4）と、地図 UI・ZIP 読み取り機能の使いやすさ（Q5）で測定した。Q4 が平均 4.00、Q5 が平均 3.67 で、総合では 3.83 となり目標の 4.0 を下回った。

基本操作は「シンプルで良い」と評価されている。しかし、変換時の設定値が何を意味するのかわからないという声が繰り返し上がった。回答者の一人は UI、マニュアル、機能要望、総合意見の 4 つの設問にわたってこの点を指摘しており、設定値の不透明さが UI 評価全体を押し下げている構造が見える。

また、LOD 選択が最小 LOD、最大 LOD、テクスチャ付きの 3 択に限られており、LOD1 だけ出力するといった使い方ができないという指摘が 2 名から独立して挙がった。

マニュアルのわかりやすさも全設問中もっとも低い評価だった。設定値の説明不足がその主因である。

6-2-1-b. 変換されたデータの使いやすさ

動作のスムーズさ（Q2）で測定した。平均は 3.86 で、目標の 4.0 をわずかに下回った。

過半数の回答者は問題なく動作したと評価している一方、前橋市全域のような大規模データでエラーが出る、

複数データの同時変換でクラッシュするといった不具合が報告された。評価は二極化しており、基本的な変換は動くが、負荷の高い使い方で安定性に課題がある。
変換後のデータについても、GeoJSON 出力の場合ファイルが 1 つにまとまって 500MB 規模になり Web 利用には重すぎる、fbx など 3D 業界標準の形式がほしいといった声があった。

6-2-1-c. データの正しさ、業務への適用

継続利用意向（Q10）で測定した。平均は 4.14 で、目標の 4.0 を上回った。
動作に不満のあった回答者でさえ継続利用意向は高く、ツールの方向性そのものは広く支持されている。地方公共団体への紹介や打合せでの 3D データ共有、GIS 業務への活用、社内システム連携など、利用シーンも多岐にわたる。
ただし、3D Tiles への変換でタイル境界にあたる建物が途中で切断される問題や、変換結果の品質に対する不満も指摘されている。コンセプトへの支持が高いだけに、品質を上げれば評価はさらに伸びる余地がある。

6-2-1-d. データ利用までの手間・時間の短縮

コスト削減割合（Q11）と削減額（Q12）で測定した。5 段階スコアではないため目標基準をそのまま適用できないが、業務利用者の 75%がコスト削減効果を認めており、部分的達成と判定した。削減見込みは最大で 40%・300 万円という回答もあった。
一方で、一括変換ができず地物型ごとに個別実行が必要なこと、変換中のクラッシュ、設定値の意味がわからず試行錯誤に時間がかかることが、効率化を妨げる要因として挙がっている。

6-2-2. 改善の方向性

- 短期的に取り組めるもの

設定パラメータの説明を UI やマニュアルに盛り込むこと。今回もっとも根深く繰り返し指摘された課題であり、これだけで UI 体験は大きく変わる。LOD 選択肢の拡充も、2 名が独立して挙げた合意度の高い要望である。マニュアルを変換フォーマットごとに整理し直すことや、UI へのバージョン表記追加も手をつけやすい。

- 中期的に取り組むもの

複数データの同時変換やクラッシュの安定性向上。3D Tiles のタイル境界での建物切断問題の修正。ファイル分割出力への対応。複数地物型の一括変換対応。いずれも業務利用の信頼性と実用性に直結する。

- 長期的に見据えるもの

fbx など GIS 以外の業界で使われる形式への対応。GIS の知識がなくても使える UI の検討。回答者からは「現在の GIS Converter は GIS を日常的に使っている層向けに寄りすぎている」という指摘があり、建設・不動産・建築設計といった分野のユーザーを取り込むにはここが課題になる。

6-2-3. 総括

4 基準のうち目標を達成したのは「データの正しさ、業務への適用」の 1 基準。「UI の使いやすさ」と「変換

されたデータの使いやすさ」は目標に近いが届かず、「手間・時間の短縮」は部分的な達成にとどまった。
ツールの方向性への支持は強く、コンセプトそのものに否定的な声はなかった。課題の多くは設定値の説明不足、安定性、出力の柔軟性といった対処可能な領域にある。設定パラメータの説明充実と LOD 選択肢の拡充を手始めに改善を進めることで、未達の基準も達成に近づけられる見通しである。
なお、回答者は 7 名とサンプルサイズが小さく、特定企業への偏りもある。利用者の広がりに合わせた追加評価が望ましい。

7. 成果と課題

7-1. 本実証で得られた成果

PLATEAU GIS Converter の開発により、CityGML 形式の 3D 都市モデルを 3D Tiles、Mapbox Vector Tiles、GeoPackage など多様な形式へ変換できるようになった。さらに、テクスチャ最適化等の機能を実装したことで、3D Tiles への変換結果は高品質かつ軽量なデータとして提供可能となった。

一方で、ダウンロードした 3D 都市モデル群（zip ファイル）には大量の CityGML（XML）ファイルが含まれており、ファイル名から対象範囲を特定することが困難であった。ファイルは bldg（建築物）などの地物型ごとにフォルダ分けされているため、利用者は地図上の整備範囲を確認し、該当するメッシュ番号を特定した上で、膨大なファイル群の中から必要なデータを手作業で抽出する必要があった。この作業負担が、利活用促進の大きな障壁となっていた。

今年度の開発では、この課題を解決するため、zip ファイルをツールに直接取り込み、地図上に 3D 都市モデルの範囲を可視化する機能を実装した。ツールは zip 内の地物型を自動抽出し、ファイル名に含まれるメッシュ番号を解析して地図上にメッシュを描画する。これにより、利用者は地図画面上から変換対象メッシュを直感的に選択し、必要なデータをシームレスに抽出・変換できるようになった。

この地図ベースのデータ選択機能により、従来の手作業によるファイル探索の手間が大幅に削減され、変換作業の効率が向上した。専門知識を持たない利用者でも容易に必要データを扱えるようになり、3D 都市モデルのアクセシビリティと実用性が大きく向上した。

7-1-1. 3D 都市モデルの技術面での優位性

本ツールの構築を通じて、以下のような 3D 都市モデルの技術面での優位性が示された。

表 7-1 3D 都市モデルの技術面での優位性

大項目	小項目	3D 都市モデルの技術面での優位性
データ変換	利活用の幅の向上	● PLATEAU GIS Converter により軽量かつ高速にデータ変換できるようになり、多様な形式で出力可能となった。これにより各種 GIS ツールや Web アプリケーションで 3D 都市モデルを利用できるようになり、利活用の幅が大きく広がった。 ● テクスチャ最適化等の実装により高品質かつ軽量な 3D Tiles 生成が実現し、低スペック PC や低速回線環境化でも円滑に動作するデジタルツインを構築可能となった。

7-1-2. 3D 都市モデルのビジネス面での優位性

表 7-2 3D 都市モデルのビジネス面での優位性

大項目	小項目	3D 都市モデルのビジネス面での優位性
地図機能の提供	利用ハードルの低減	● 従来、高価な商用ツールを必要としていた 3D 都市モデルの変換が、PLATEAU GIS Converter により、誰でも容易に実施することが可能となった。これにより、特定のツールに依存しないオープンな利用環境が整備され、利活用コストが大幅に低減した。 ● また、地図機能の実装により利用箇所の特定が容易になり、データ変換に要する作業負担が削減され、民間事業者による活用の裾野が広がった。

7-2. 今後の展望

3D 都市モデルは、建築・土木分野における DX 推進の中核を担う基盤データであり、BIM/CIM との連携が強く求められている。しかし現状では、PLATEAU が採用する CityGML 形式と主要な BIM/CIM ソフトウェアとの間に形式的な非互換性が存在し、活用範囲が限定されている。また、標準製品仕様書 5.0 版から新たに追加された点群データ（LAS/LAZ 形式）は、既存の PLATEAU GIS Converter との相互運用性が確立されておらず、これが建築・土木分野への利活用拡大を阻む要因となっている。これらの課題を解消しつつ、3D 都市モデルと他データとの互換性を高めるため、本ツールを開発し、変換機能の妥当性、業務での活用可能性を示すことができた。

他方、アンケート等で判明した課題を解消するため、今後は以下の 3 点を重点的に取り組む方針である。
第一に、BIM/CIM ソフトウェアで広く採用されている FBX 形式への変換機能を開発し、設計・施工プロセスにおける 3D 都市モデルの直接利用を可能とする。

第二に、標準製品仕様書に準拠した LAS/LAZ 形式の点群データの取り込み機能を実装し、PLATEAU データの新仕様に対応する。これにより、地形・構造物・地表面の統合的な可視化・解析が可能となる。

第三に、変換後データの内容を即時に確認できるビューワー機能を提供し、変換結果の品質確認や活用シーンの拡大を支援する。

これらの取り組みにより、異なるデータ形式間の相互運用性が大幅に向上し、3D 都市モデルを中心としたデータが共通で使われる環境（エコシステム）が形成される。特に、従来は複数のソフトウェアを組み合わせて行っていた変換・閲覧・確認作業を一体化することで、業務効率の向上と利用者層の拡大が期待される。開発成果は MIT ライセンスのもとで GitHub に公開し、ドキュメント整備やコミュニティ連携を通じて、産学官民に広く活用されるオープンな基盤として発展させていく予定である。

8. 用語集

A)　アルファベット順

表 8-1 用語集（アルファベット順）

No.	用語	説明
1	3D Tiles	空間的にインデックスを付けた 3D データのセットをストリーミングするためのフォーマットで、主に CesiumJS などで利用される
2	CZML	時間と共に変化する空間データを表現する JSON ベースのフォーマット
3	GDAL	地理データ形式を扱うためのライブラリ及びツールセット
4	GeoJSON	地理データを JSON 形式で符号化する地理情報データ形式
5	GeoPackage	SQLite ベースの地理情報データ形式
6	glTF	3D モデルデータを JSON 形式とバイナリ形式の 2 つで格納するファイルフォーマット
7	glb	3D モデルデータをバイナリ形式で格納するファイルフォーマット
8	KML	地理データを表現・配布するための XML ベースのフォーマット
9	Rust	安全性と並行性を重視したシステムプログラミング言語
10	Shapefile	GIS で広く使用される地理情報データ形式
11	Svelte	宣言的に記述することができるフロントエンドフレームワーク

以上

GIS コンバーターの開発
技術検証レポート

2026 年 3 月　発行

委託者：国土交通省　都市局

受託者：株式会社 MIERUNE