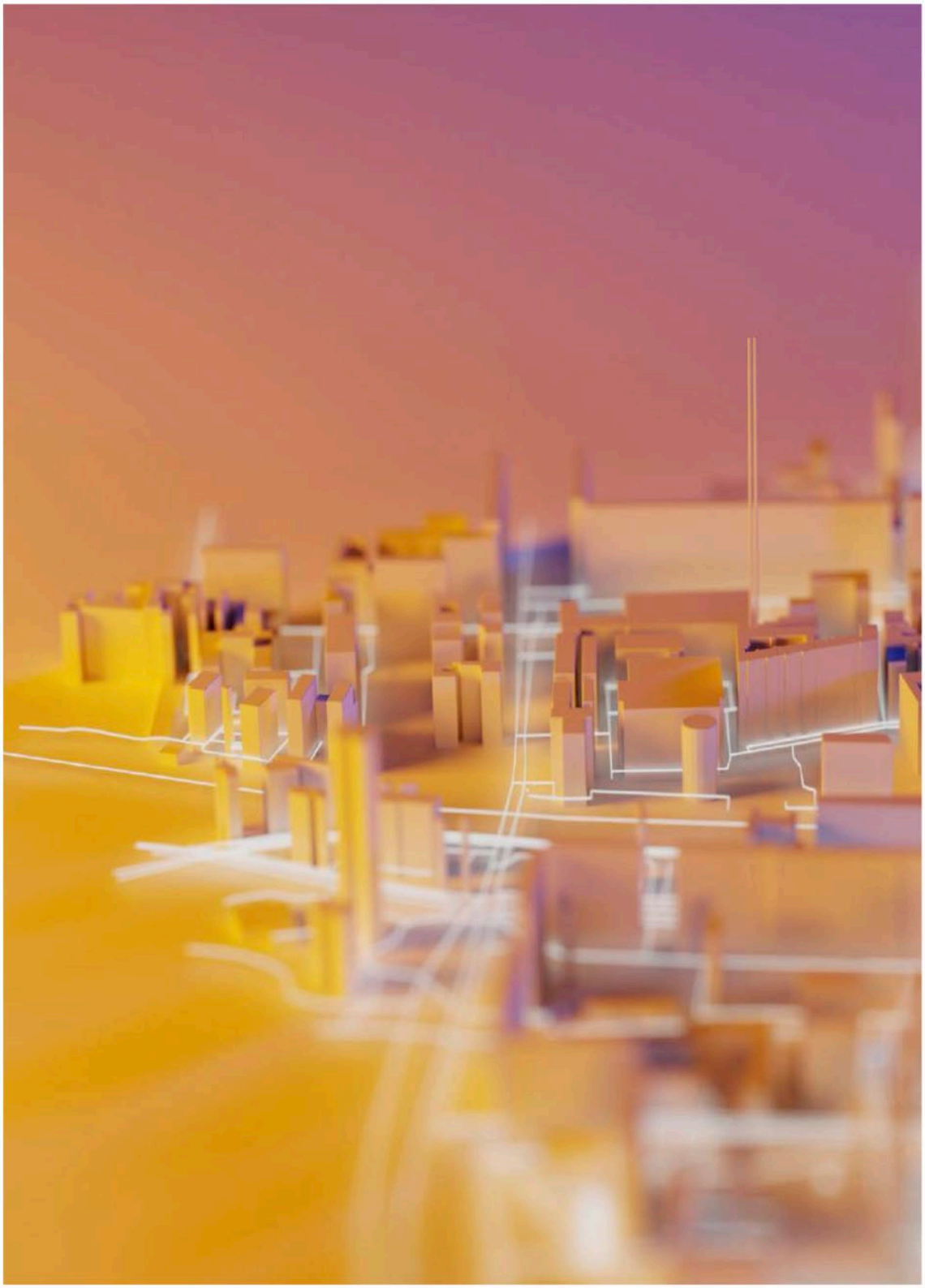
PLATEAU
by MLIT

Handbook of 3D City Models

3D都市モデル導入のためのガイドブック



実証環境構築マニュアル

series No. 09

3D City Model Demonstration Manual

はじめに

- Project PLATEAUでは、 2020年度に3D都市モデルの実証環境構築のための実証調査を実施した。本実証調査は、3D都市モデル及びこれを活用したユースケース開発のための可視化環境である「PLATEAU VIEW」を開発することで、3D都市モデルがもたらすソリューションの価値を検証することを目的としている。
- 実証環境の構築を検討するに当たっては、Project PLATEAUのオープンデータの思想に基づき、出来るだけオープンソースのフレームワークを利用することで、ベンダーロックを回避する形でシステム構成を行うことを心掛けた。また、オープンソースの利用には、地方公共団体等が低コストで類似のシステムを構築できることや、技術者コミュニティとの連携によるシステムの持続的な発展が期待されること等のメリットもある。
- 2021年度には、PLATEAU VIEWに機能追加を行った「PLATEAU VIEW ver1.1」を開発し、GitHUB上でソーススクリプトを公開した。また、「第2章　実証環境の構築手順」の内容を大幅に拡充し、PLATEAU VIEWを構築するためのチュートリアルを充実させるとともに、Project PLATEAU GitHUB上でTerria.js用カタログ生成アプリとそのチュートリアルを公開した。
- 本マニュアルは、PLATEAU VIEWの開発により得られた成果をもとに、その機能、システム環境、仕様、構築手法等を解説することで、地方公共団体や民間企業、技術者コミュニティ等に所属する多様なプレイヤーが3D都市モデルの可視化環境を構築する際に参照できる知見を提供し、3D都市モデルの整備及びこれを活用したユースケース開発への参画のすそ野を広げることを目的とするものである。
- 地方公共団体や民間企業、技術者等の多くの方に本マニュアルを参照していただき、Project PLATEAUの技術コミュニティがさらに発展することを期待する。

アップデートノート

2022.3.25　「実証環境構築マニュアル ver2.0」

2021年度の調査結果を踏まえ、以下の項目を改訂した。

- 「1.4　ソフトウェア構成」を新たにリリースしたPLATEAU VIEW(ver1.1)の構成にアップデート
- 「第2章　実証環境の構築手順」の内容を大幅に拡充
- 「第3章 ユーザーマニュアル」をPLATEAU VIEW(ver1.1)に合わせて改定

2021.3.26　「実証環境構築マニュアル ver1.0」

■目次

第１章　実証環境の構成

1.1本マニュアルの目的

3D都市モデルの実証環境とは、3D都市モデル及びこれを活用したユースケース開発のために可視化環境を提供するプログラム、サーバ、データ等の一連のシステムをいう。具体的な機能としては、3D都市モデルそれ自体を可視化することに加え、3D都市モデルと共に分析やシミュレーション等に用いられる各種データの可視化も行う。これにより、3D都市モデルの提供価値を検証することができる。

2020年度のProject PLATEAUでは、実証環境としてウェブ上で閲覧可能なビューア「PLATEAU VIEW」を開発し、ウェブサイト「PLATEAU」上で公開している。また、2021年度には、PLATEAU VIEWに機能追加を行った「PLATEAU VIEW ver1.1」を開発し、アップデートを行った。

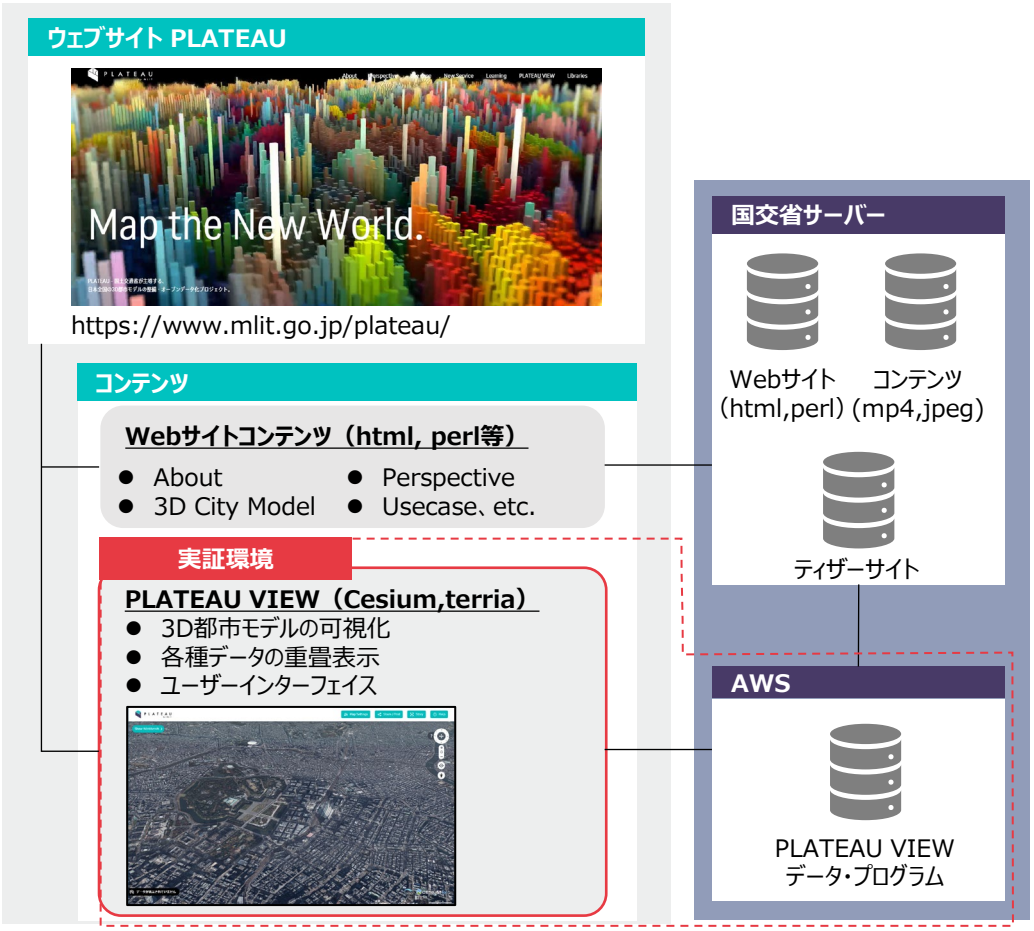
PLATEAU VIEWは、全国約50都市で整備された3D都市モデルを可視化すると共に、都市計画決定情報、人流データ、都市アセットデータ等の各種のデータを3D都市モデルに重畳して表示することが可能なUI（User Interface）を一般向け・行政向けに提供する。ブラウザ上の操作により、表示データの追加・削除、マップの拡大・縮小・視点移動、ベースマップの切り替え、任意の画角の保存・URL発行等が可能である。

本マニュアルでは、PLATEAU VIEWの機能、システム環境、仕様、構築手法等を解説することで、地方公共団体や民間企業等が3D都市モデルの可視化環境を構築する際に参照できる知見を提供し、3D都市モデルの整備及びこれを活用したユースケース開発を促進することを目的とするものである。

なお、2021年度に開発した「PLATEAU VIEW Ver1.1」のソースコードについては、Project PLATEAUのGitHUBにおいてオープンソースとして公開しているので、参考にして頂きたい。

Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW

図　Project PLATEAUウェブサイトの構成



1.2 サーバ環境

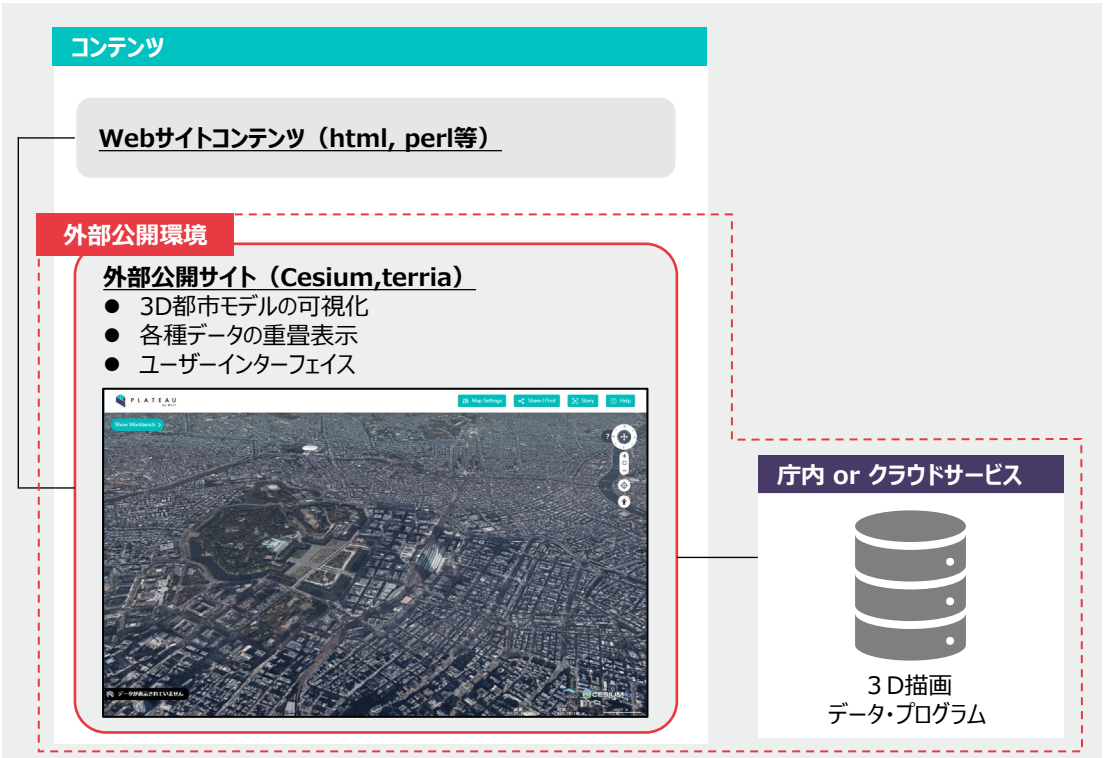
PLATEAU VIEWデータや関連プログラムは、レンダリングデータやオーバーレイデータと共にAWS（Amazon Web Services）上のサーバに格納されており、ユーザからの要求に応じてデータを呼び出す仕組みである。PLATEAU VIEWでは、全国約50都市分の巨大なデータを配信する必要があったことから、スケーラビリティに優れる外部クラウドサービスであるAWSを採用している。

他方、地方公共団体において実証環境を構築する場合は、必ずしも外部クラウドサービスを用いる必要はなく、庁内のサーバ（オンプレミス）を利用する方法も可能である。この二つの方法のメリット・デメリットを整理すると以下のようになる。

表　サービス提供方法の比較

	庁内サーバの利用（オンプレミス）	外部クラウドサービスの利用
利用方法	庁内の既存サーバ、新規調達する物理的なサーバを利用	民間企業が提供する外部クラウドサービスを利用
メリット	既存サーバを利用できれば、初期費用を抑えることが可能	搭載するデータ量の増加、利用者の増加への対応（スケールアップ）が容易に対応可能
デメリット	一度購入したサーバのスケールアップ（容量追加、台数追加）は調達上煩雑	インターネット回線の利用が前提であり、庁内からの接続可能な端末が限定される（自治体のセキュリティポリシーに依存）

図　地方公共団体独自サイトの構築例



1.3 サーバ構成

PLATEAU VIEWでは、AWSのサービスであるAmazon EC2（Amazon Elastic Compute Cloud）を利用して、 データやプログラムを格納するサーバを構築した。全国約50都市の大規模なデータをリアルタイムに処理する必要があることから、ELB（Elastic Load Balancing）により５台の仮想サーバでトラフィックを自動的に分散している。可視化やUIに関するデータ類の容量は約550GBであり、ストレージはAmazon S3（ストレージ容量が無制限）を利用している。下表に現在のPLATEAU VIEWを運用しているサーバ構成を示すが、利用者数やデータサイズ、求められるサービスレベルに応じて任意のサイズで構築されたい。

なお、AWSサーバの構築・運用では、サーバ、ストレージ、データ転送について料金が発生する。

表　AWS上のサーバ環境（全国版の事例）

項目	内容
サービス名	Amazon EC2（Amazon Elastic Compute Cloud）
OS	Linux（ubuntu）
プロセッサー	Intel Xeon Platinum 8000
インスタンスタイプ	t3.small
vCPU（仮想CPU）	2
メモリ	2.0 GiB
仮想化タイプ	ハードウェア仮想マシン（hvm）
ロードバランサー（負荷分散装置）	ELB（５台構成）
ストレージ	Amazon S3（Simple Storage Service ） （実際の使用容量は約550GB）
リージョン	ap-northeast-1（東京）

自治体向け留意点

各自治体が自らの都市を対象とした環境を構築する際は、PLATEAU VIEWに比べて扱うデータの規模が小さくなるため、比較的小規模なサーバ構成で足りるものと考えられる。庁内サーバ又は外部クラウドサービスのサーバ環境を構築する際のデータ規模の目安は以下のとおりであり、上述のAmazon EC2を利用する場合は、ストレージは1/50程度、ロードバランサーは1台(1/5)、CPU機能はXeon E5系プロセッサー程度のスペックで足りると見込まれる。

・建物データの目安：政令市・テクスチャ付き　2GB

基礎自治体・テクスチャなし　0.5～1GB

・地形データの目安：0.1～1.5GB（取得精度に依存）

・オルソデータの目安：1～50GB（面積に依存）

・災害リスクデータの目安：1～2GB（ハザード数に依存）

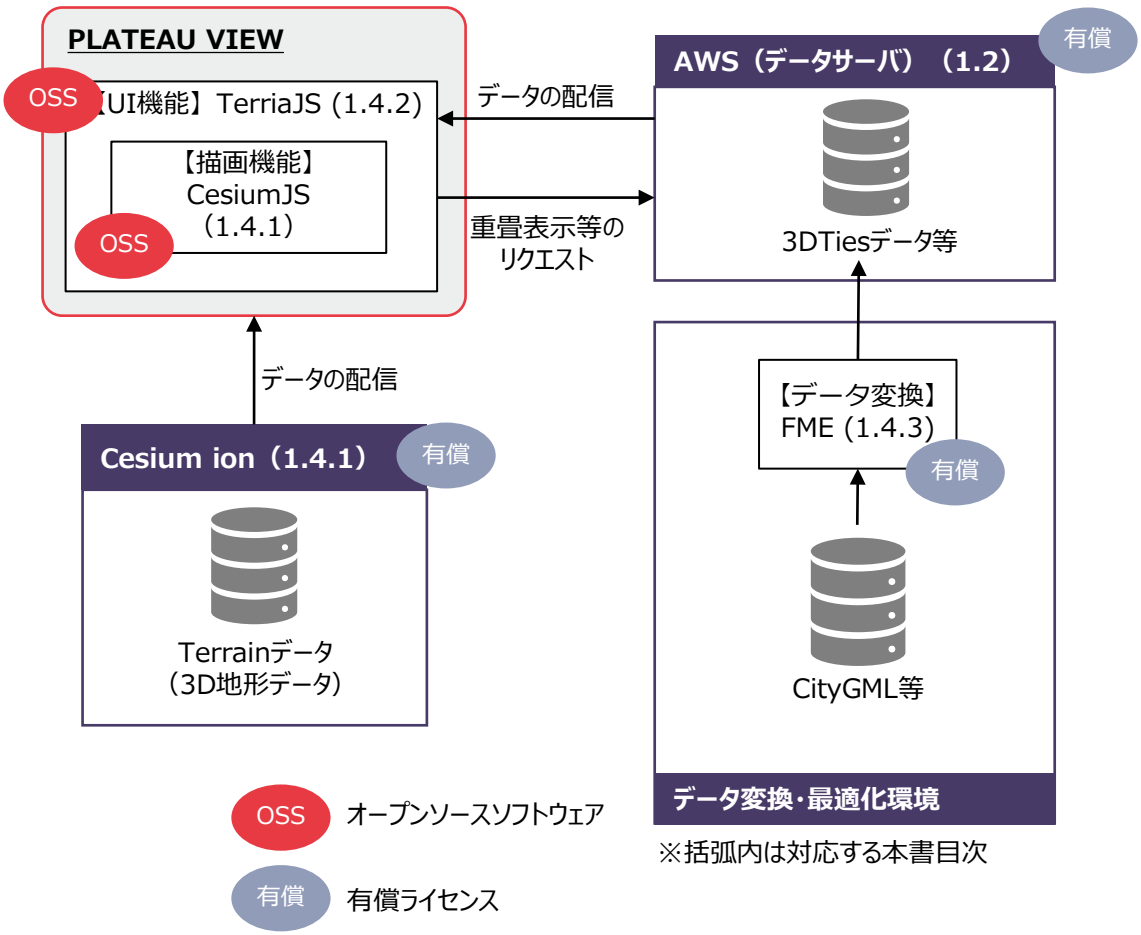
1.4 ソフトウェア構成

PLATEAU VIEWで使用している主要なソフトウェア類とその機能を解説する。ソフトウェアは、オープンソースソフトウェア（OSS）と有償ライセンスのソフトウェアの両方を組み合わせて利用している。「PLATEAU VIEW」ではこれらのソフトウェアが有する基本的な機能に加えて、独自に追加・拡張した機能も使用している。

【CesiumJSとTerriaJSの関係】

CesiumJSはデータの描画、TerriaJSはUI（ユーザーインターフェイス）の提供及びUIを介してCesiumJSの描画機能を制御する役割となる。実証環境が有する様々な機能は、TerriaJSを中心に構築されている。なお、各種3Dデータを配置する地表面を表現するデータ（Terrainデータ）については、Cesium社が提供しているクラウドサービスであるCesium ionを使用して全国の地形データの配信を行っている。

図　全国版・実証環境を構成する主要なソフトウェア類の構成



1.4.1 CesiumJS

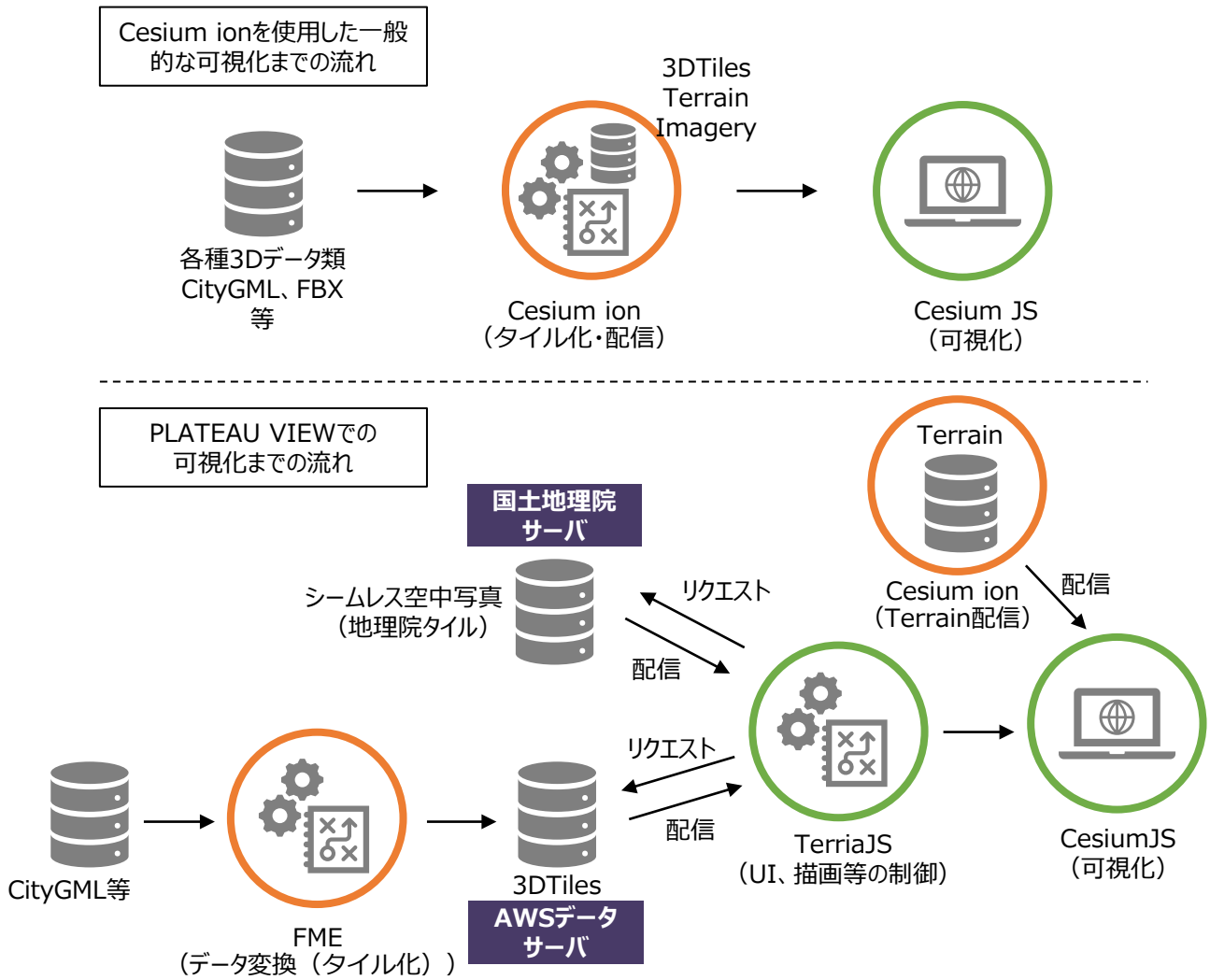
CesiumJSは、3次元位置情報を活用するために設計されたソフトウェアの基盤となるオープンプラットフォームである。 WebGLを利用しており、デスクトップ用あるいはスマートフォン用ブラウザで動作する。ブラウザ上で3Dマップを表示するためのJava Scriptライブラリ（CesiumJS）と、様々なファイル形式のデータをストリーミング用に最適化（タイル化）し、配信するサービスとしての機能（Cesium ion）で構成されている。なお、CesiumJSは、オープンソースコードとしてGithubにて公開されている。https://github.com/CesiumGS/cesium

CesiumJSでは、多様な形式かつ膨大なサイズのデータをブラウザ上でストレスなくストリーミングおよびレンダリングすることを可能とするために、3DTiles、Terrain（地形）、Imagery（イメージ）という独自の描画機構により軽量化・最適化を図ったファイル形式を採用している。Cesium ionは、様々なファイル形式に対応しており、 Cesium社が提供するプラットフォームを介して、ユーザーによるデータのアップロード、管理、タイリング、配信等を行うことができる。

PLATEAU VIEWでは、 FME（1.4.3参照）を用いてCityGMLから3DTiles形式への変換を行っており、描画機能としてCesiumJSを使用している（Cesium ionは使用していない）。

なお、Cesium社はCesium ionというクラウドサービスも展開しており、Cesiumがサポートする3Dデータ（p.12）をアップロードすると3D地図として表現するのに適した形式である3DTilesに変換したうえで、ホスティングするサービスを提供している。このサービスを利用すれば、独自にデータの変換（2.4参照）をせずに、参照するURLを取得し、3Dモデルを表示することができる。本事業では、3Dデータの配信には使用していないが、3Dデータを配置する際の基盤となる地形データ（Terrain）の配信に使用している。

図　CesiumJSを使用した可視化までの流れ



【地理院タイル等の配信】

PLATEAU VIEWでは、背景図として選択可能な「空中写真」は、国土地理院サーバーから配信される地理院タイルを取得してCesium上で描画している。地理院タイルとは、国土地理院が配信するタイル状の地図データである。国土地理院では、ユーザーからリクエストされた地図表示範囲に対して、予めタイル状に分割したデータを国土地理院サーバーから配信するサービスを提供している。

http://maps.gsi.go.jp/development/siyou.html

また、同じく背景図として選択可能な「空中写真（Bing）」についてはBingMAP、「地理院地図」についてはOSM（OpenStreetMap）から配信されるデータを描画している。

https://openstreetmap.jp/

https://www.microsoft.com/maps/ja-JP/choose-your-bing-maps-API.aspx#1

表　変換後にCesiumJSで利用できるファイル形式

描画機構	対応ファイル形式	拡張子	備考
3D Tiles	CityGML	.citygml,xml,gml	3次元地理空間データ
	KML/KMZ	.kml,.kmz	KML/KMZ
	LASer	.las,.laz	点群データ(LASファイル)
	COLLADA	.dae	3D-CG間の交換フォーマット
	Wavefront OBJ	.obj	3Dジオメトリーのみの表現形式
Terrain（地形）	Floating Point Raster	.flt	
	Arc/Info ASCII Grid	.asc	
	Source Map	.src	
	Geo TIFF	.tiff,.tif	
	Erdas Imageine	.img	
	USGS ASCII DEM /CDED	.dem	
	Cesium Terrain Database	.terraindb	
Imagery（イメージ）	Floating Point Raster	.flt	
	Arc/Info ASCII Grid	.asc	
	Source Map	.src	
	Geo TIFF	.tiff,.tif	
	Erdas Imageine	.img	
	USGS ASCII DEM /CDED	.dem	
	Cesium Terrain Database	.terraindb	
	JPEG	.jpg,.jpeg	
	PNG	.png	

表　TerriaJSがサポートするサービス

描画機構	区分	内容
グラフィックオブジェクト	ポイント	2D/3D空間上に一時的なグラフィックスを描画
	ライン	
	ポリゴン	
	ラベル	
	マーカー	
外部サービス（高解像度画像）	WMS	背景地図を表示
	TMS	
	WMTS	
	ArcGIS	
	Bing Maps	
	Google Earth	
	Mapbox	
	Open Steet Map	
	Open Steet Map Buildings	
外部サービス（地形データ）	Cesium World Terrain	海、湖、川の地球規模での高解像度の地形を表示
	Google Earth Enterprise	
	VT MAK VR- TheWorldサーバ	
ベクターデータ	GeoJSON	直接参照可能なファイルフォーマット
	TopoJSON	
	CZML	
	KML,KMZ	

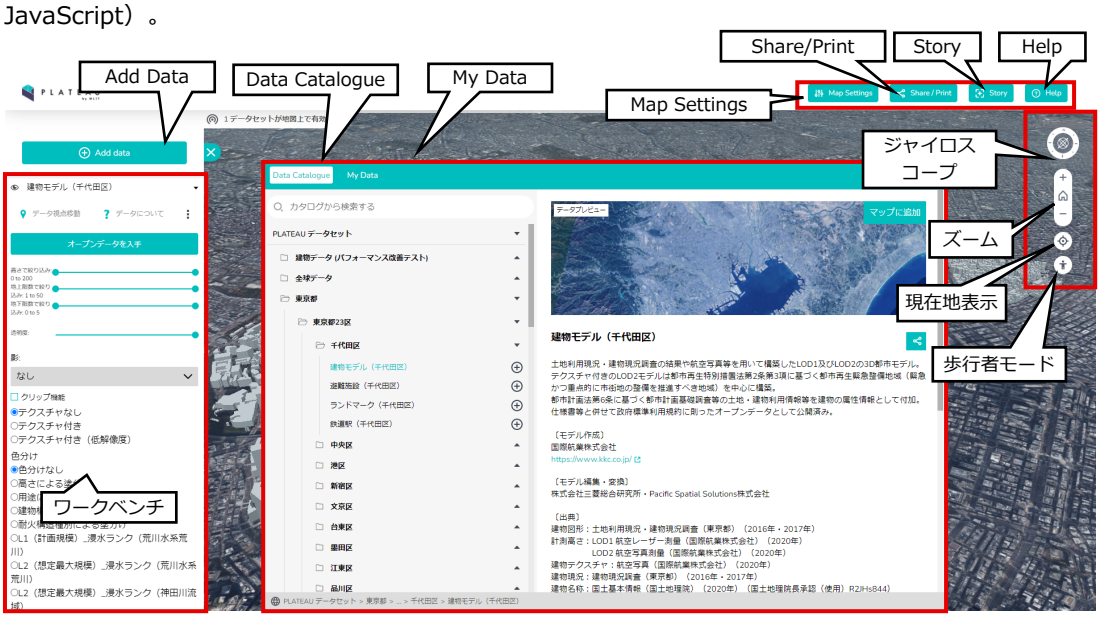
1.4.2 TerriaJS

TerriaJSは、CesiumJS向けにUIを付加したり、様々な形式のデータを読み込めるようにするオープンソースのフレームワークである。https://terria.io/

CesiumJSの描画機能をコントロールする役割を持っており、PLATEAU VIEWでは、TerriaJSのUI機能を使用して、表示データの選択・追加・削除（Add Data、データカタログ、ワークベンチ）、背景図の選択（Map Settings）、URL生成／印刷（Share/Print）、ストーリーの設定（Story）、Help、自分のデータを追加等を提供している。

【TerriaJSの拡張】

TerriaJSは機能を追加することができる。PLATEAU VIEWでは、CityGMLで作成した建物モデルに付与された属性情報を検索する機能や歩行者ビュー機能等を追加で開発した（開発言語：JavaScript）。



UI機能	内容
Add Data	Data Catalogue画面の起動
Data Catalogue	データの選択
My Data	ローカル及びWeb上のデータをアップロード
ワークベンチ	Data Catalogueで選択したデータの表示
Map Settings	背景地図の選択・地図機能の設定
Share / Print	URL生成と印刷
Story	ストーリーエディタの起動（複数のポイントでシーン設定をして保存、ポイントを移動しながら再生ができる）
Help	ヘルプ（マウス操作）の起動
ジャイロスコープ	ビューの回転・傾け、初期位置の表示
ズーム	ズームイン／ズームアウト
現在地表示	自位置に地図を寄せる
歩行者モード	歩行者モードの開始

表　TerriaJSによって実装された主な機能一覧（1／2）

機能名称	操作	内容	機能区分		
			基本	追加	
				ver1	Ver1.1
3次元視点移動	マウス移動	任意の位置に視点を移動	○		
	エリア選択移動	選択した都市への視点移動	○		
	UC選択移動	選択したUC（ユースケース）への視点移動	○		
属性情報の表示	地物選択	【選択時に表示される属性情報の例】 建物：高さ、階数、住所、都市計画区域、用途地域、建物用途など 鉄道駅：路線名、運営会社、駅名など 避難施設：施設名、住所、避難施設の種類など	○		
属性情報の色分け表示	属性選択（排他制御）	特定のコードを持つ地物に色を割り当てて表示		○	
属性情報のピックアップ表示	属性選択（範囲選択）	特定の属性項目に属するコードにしきい値を設定し、しきい値以上のコードを持つ地物のみを表示など、属性値に基づいた絞り込み		○	
検索機能	キーワード選択・入力	特定の属性項目に属するコードを検索して表示		○	
オーバーレイデータの表示	地図選択[Map Settings]	3D地形データ（DEM)の表示	○		
		2D表示	○		
		空中写真（国土地理院）の表示		○	
		空中写真（Bing）の表示	○		
		地理院地図（淡色）の表示	○		
		Dark Matter地図の表示	○		
	静的情報表示	建物関係、都市インフラ、都市計画、災害系情報、国土数値情報（駅、避難施設など）の表示	○		
	動的情報表示	人流、バス走行位置、コミュニティサイクル走行状況（基本） リアルタイムデータ（GTFS-RT、CSVなど）の表示（ver2）	○		○
日影表示	日影表示	受光、投影、受光＋投影の3パターンから選択して日影を表示、あるいは非表示の選択	○		

基本機能：TerriaJSに元々備わっている基本機能を利用してPLATEAU VIEWに実装した機能

追加機能：TerriaJSの機能を拡張してPLATEAU VIEWに実装した機能（実装後TerriaJSに統合された機能も含む）

※表中の[]は、P.13に示したUI機能を指す

表　TerriaJSによって実装された主な機能一覧（2／2）

機能名称	操作	内容	機能区分		
			基本	追加	
				ver1	Ver1.1
基本操作機能	開始地点に戻る	Homeボタンを配置し任意の1点に戻る	○		
	[歩行者モード]	視点と注目点を決めて、アイレベルから見たシーンを表示（ver1） 地表面だけでなく、3Dオブジェクト（建物モデルなど）の表面に沿って移動する機能（ver2）		○	○
	メニュー表示・非表示（折りたたみ）	画面上に表示される各種情報を折りたたむことができる機能	○		
	印刷機能 [Share/Print]	印刷、PDFファイル出力	○		
	クリップ機能	3Dモデルを任意の直方体で切り取り表示する機能			○
現在地表示	[現在地表示]	GPSで取得した現在地を表示	○		
スマホ対応		スマートフォン上に表示	○		
データカタログ作成UI	[自分のデータを追加]	ユーザーがデータをアップロードして、データカタログに新たなレイヤとして登録	○		
データカタログからの選択	[データカタログ]	データカタログに登録してあるデータレイヤを選択して、重ね合わせる	○		
ストーリー作成と保存	[Story]	ルートを設定し動画を作成	○		
パブリッシュ設定	[Share/Print]	外部向けの公開サイトにURL等を発行	○		
タイムライン表示	[Map Settings]	タイムラインの表示・非表示切り替え			○

基本機能：TerriaJSに元々備わっている基本機能を利用してPLATEAU VIEWに実装した機能

追加機能：TerriaJSの機能を拡張してPLATEAU VIEWに実装した機能（実装後TerriaJSに統合された機能も含む）

※表中の[]は、P.13に示したUI機能を指す

1.4.3 FME

FMEは、位置情報データ変換の世界標準ソフトであり、 スマートシティ向けのシステム構築でも利用されている。CityGML、IFC等400種類以上のフォー マットに対応し、プログラミング不要の画面操作だけで、データ、サーバー、外部APIデータに接続し、データ変換、各種ビューアへの取り込みができる。トランスフォーマーと呼ばれるデータ変換ロジックを有したパーツを組み込んだワークスペースを作成することにより、多種多様な形式のデータ構造・内容を自由に変換することができる。また、作成したワークスペースをFME Serverに登録することにより、データの変換作業を自動化することも可能である。

FMEは、多様なデータ形式に対応したトランスフォーマー（データ変換ロジック）をライブラリとして用意しており、 PLATEAU VIEWでは、CityGML形式等から3DTiles形式への変換を行っている。

FMEは以下の製品・サービスから構成され、有料ライセンスソフトウェアである。

FME Desktop: ワークスペース（データ変換のフロー）を作成、実行するためのツールセット
FME Server: ワークスペースに基づくデータ変換サービスを提供するためのサーバーソフトウェア
FME Cloud: クラウド上でFME Server の機能を提供するサービス(時間課金または年課金)

https://pacificspatial.com/fme/

なお、PLATEAU VIEWで用いたFMEの変換テンプレートは以下のURLで公開している。

https://github.com/Project-PLATEAU/FMEscript-CityGML-to-3DTiles

図　FMEによるデータ変換処理のイメージ

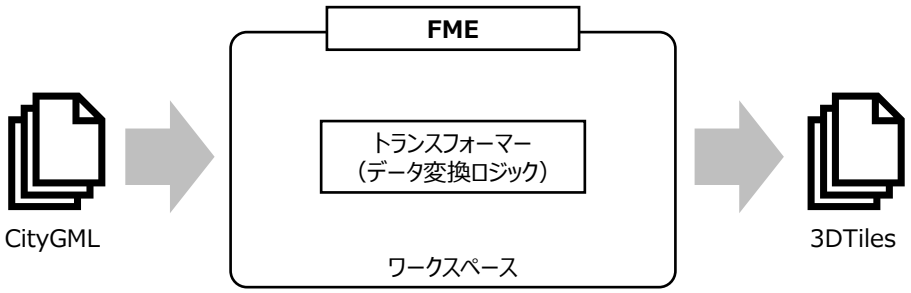
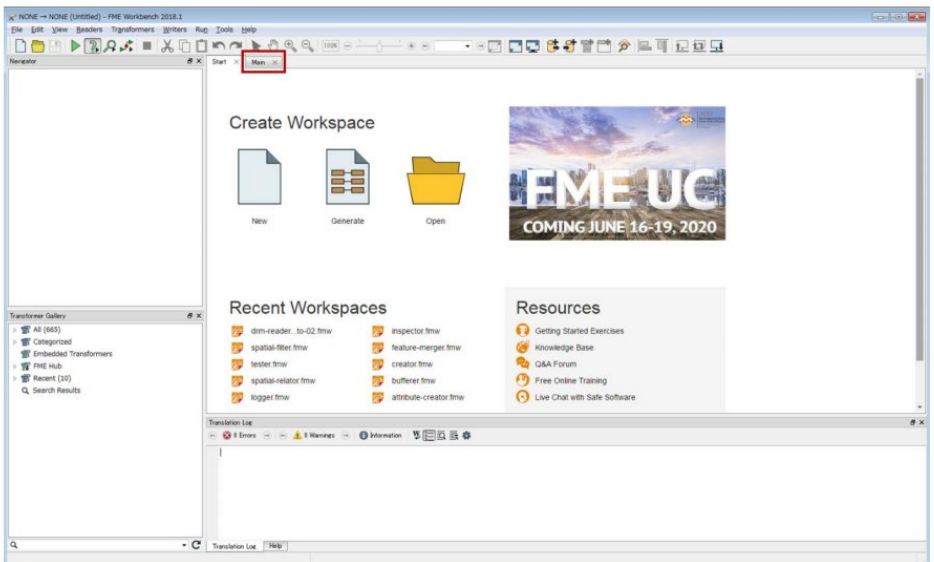


図　FMEの画面（Workbench）



1.5 実証環境のデータ構成

1.5.1 データの種類

「PLATEAU VIEW Ver1.0」で表示可能なデータセットの種類の例と出典を示す。

表　東京都の主なデータセット（1／2）

データ区分/データ名			出典
建物関連	建物モデル ※LOD1及びLOD2の3D都市モデル		建物図形：土地利用現況・建物現況調査（東京都）、航空写真測量（国際航業） 計測高さ：LOD1 航空レーザー測量（国際航業）、LOD2 航空写真測量（国際航業） 建物テクスチャ：航空写真（国際航業、八王子市） 建物現況：建物現況調査（東京都） 都市計画情報：東京都の都市計画地理情報システムデータ 土砂災害警戒区域：国土数値情報 津波浸水想定：東京都津波浸水想定 洪水浸水想定区域：浸水想定区域図（国土交通省）、浸水予想区域図（東京都市型水害対策連絡会）
	避難施設		国土数値情報（国土交通省）
	ランドマーク		建物の属性（名称）：国土地理院発行の数値地図（国土基本情報）（測量法に基づく国土地理院長承認（使用）R 2JHs 844」を加工して作成。
	鉄道駅		国土数値情報（国土交通省）
都市アセット	道路		土地利用現況（土地利用現況・建物現況調査の結果や航空写真等）
	公園		土地利用現況（土地利用現況・建物現況調査の結果や航空写真等）
	橋梁モデル ※橋梁、道路高架部、鉄道高架部、歩道橋、ペディストリアンデッキ		航空写真（国際航業、八王子市）
交通	鉄道		国土数値情報（国土交通省）
	緊急輸送道路 ※第1～3次緊急輸送道路		国土数値情報（国土交通省）
都市計画	都市計画道路		都市計画道路（東京都）2都市基街都第247号
	都市高速鉄道		都市高速鉄道（東京都）2都市基交都第56号
	地区計画		都市計画地理情報システムデータ（東京都）
	地域地区	用途地域	都市計画決定情報GISデータ
		高度地区	
		防火地域・準防火地域	
	用途地域		都市計画決定情報GISデータ
	土地利用 ※LOD1及びLOD2の3D都市モデル		土地利用現況・建物現況調査（東京都）
	行政界		国土数値情報（国土交通省）

表　東京都の主なデータセット（2／2）

データ区分/データ名			出典
都市活動	高速道路、一般道 ※車の移動速度から求めた渋滞状況		スマートフォンアプリを通じて許諾を取得したアプリユーザーの位置情報を、個人特定を回避する形で加工したデータ（フリックテック株式会社）
	地価バリューマップ		国土交通省国土政策局　国土数値情報　「地価公示」、「都道府県地価調査」をもとに株式会社日建設計総合研究所が編集・加工
	東日本大震災時の帰宅困難者 ※東京駅周辺、池袋駅周辺		国土交通省都市局都市安全課、株式会社日建設計総合研究所
	（比較用）平常時の滞留者 ※東京駅周辺		国土交通省都市局都市安全課、株式会社日建設計総合研究所
	経済センサス　事業所数 平成28年度		総務省統計局　「平成28年度経済センサス・基礎調査」
	経済センサス　従業者数 平成28年度		総務省統計局　「平成28年度経済センサス・基礎調査」
洪水浸水想定区域図	洪水浸水想定区域図 ※荒川、江戸川、利根川、多摩川、城南地区、江東内部河川、石神井川および白子川、神田川、野川		河川事務所、東京都が作成した浸水想定区域図
ユースケース例	大丸有	丸の内ストリートパーク人流データ	大丸有エリアマネジメント協会、大手町・丸の内・有楽町地区まちづくり協議会、三菱地所株式会社
		丸の内シャトル：バス走行位置、バス停、走向ルート	日の丸自動車興業株式会社
		千代田区コミュニティサイクル：利用状況、サイクルポート	株式会社ドコモ・バイクシェア
	渋谷区	渋谷人流データ	許諾に基づきスマートフォンアプリを通じてユーザーの位置情報を取得し、個人特定を回避する形に処理したうえで株式会社ブログウォッチャーが提供するデータを編集・加工
	羽田	羽田イノベーションシティ BIMデータ	鹿島建設株式会社
	竹芝	東京ポートシティ竹芝 BIMデータ	鹿島建設株式会社
	リアルタイム	東京都交通局　バスロケーション情報	東京公共交通オープンデータチャレンジ
		西武バス　バス関連リアルタイム情報	東京公共交通オープンデータチャレンジ
		関東地方整備局　河川ライブカメラ	関東地方地方整備局京浜河川事務所、荒川下流河川事務所

表　東京都以外の主なデータセット（例：横浜市）

データ区分/データ名		出典
建物関連	建物モデル ※LOD1あるいはLOD1及びLOD2の3D都市モデル	建物図形：都市計画基本図、都市計画基礎調査（各自治体） 計測高さ：航空写真測量（各自治体） 建物現況：建物現況調査（各自治体） 土地利用現況：土地利用現況調査（各自治体） 都市計画情報：都市計画区域（各自治体） 土砂災害警戒区域：国土数値情報（国土交通省） 洪水浸水想定区域：浸水想定区域図（国土交通省） 津波浸水想定区域：津波浸水想定図（各自治体） 建物の属性：名称は、数値地図（国土基本情報）「国土地理院」を加工。「測量法に基づく国土地理院長承認（使用）R 2JHs 809」 建物の属性：名称（駅名）は、「国土数値情報（鉄道データ）」「国土交通省」を加工。
	避難施設	国土数値情報（国土交通省）
	鉄道駅	国土数値情報（国土交通省）
都市アセット	道路	都市計画基本図、基盤地図情報
	公園	国土数値情報（国土交通省）
交通	鉄道	国土数値情報（国土交通省）
	緊急輸送道路 ※第１～3次緊急輸送道路	国土数値情報（国土交通省）
都市計画	区域区分	都市計画情報（各自治体）
	都市計画区域	都市計画情報（各自治体）
	用途地域	都市計画情報、土地利用現況図（各自治体）
	土地利用	土地利用現況図（各自治体）、国土数値情報（国土交通省）
	行政界	国土数値情報（国土交通省）
洪水浸水想定区域		河川事務所、各自治体が作成した浸水想定区域図
土砂災害想定区域		各地方自治体が作成した土砂災害想定区域
津波浸水想定区域図		各地方自治体が作成した津波浸水想定区域図
ユースケース例	人流データ（赤外線センサ）（横浜市）	赤外線センサーを用いて人数を計測（株式会社日建設計総合研究所）
	みなとみらい　クイーンズスクエア横浜（地下鉄改札～地上階・共用部分） ※LOD4の建物モデル	3DshapeデータをCityGML形式に変換（株式会社日建設計総合研究所）
	リアルタイムデータ	横浜市交通局　バス関連リアルタイム情報（東京公共交通オープンデータチャレンジ）

1.5.2 データの形式

実証環境で扱った主なデータ形式及びデータ変換パターンの例を示す。

表　データ形式の一覧

データ形式	概要
CityGML（City Geography Markup Language）※1、2	• OGC (Open Geospatial Consortium)が策定した3D都市モデルのためのオープンデータモデル及びデータ形式の国際標準を指す。 • 建築物や道路、橋梁などの様々な地物（オブジェクト）について定義し、これに名称や用途、建築年、行政計画といった都市活動に関する情報（セマンティクス）を付与することで、都市空間の意味や地物間の関係性を再現したジオメトリ（幾何形状）とセマンティクスを統合したモデルである。 • 地物や地物属性を追加できる拡張機能やADE（Application Domain Extension）と呼ばれる、CityGMLの仕様自体を拡張し、地物や属性の応用スキーマを新たに定義する機能も有している。 • 建物等の地物の表現に関して、LOD（Level of Details）と呼ばれる概念を採用することで、同じ地物に関する詳細度の異なる様々な情報を統合的に管理・蓄積・利用することを可能としている。
3D Tiles※3	• 3D建物、BIM/CAD、点群データ（ポイントクラウド）、地形データ、写真測量などの3次元地理空間コンテンツをストリーミング及びレンダリングするための空間データの構造とタイルフォーマットを定義した形式である。OGC標準としても採用されている。 • 軽量・最適化されたデータ形式のため、3Dデータでもスムーズなweb配信が可能であり、Cesiumで採用されている。
CZML※4	• 主にCesiumで用いられるJSON形式を基本としたデータ形式である。 • 3Dデータ及び時間データの表現に適しており、時間の経過と共に値を変更するプロパティを正確に記述することができる（例えば、ある時間帯は赤色の線、ある時間帯は青色の線を表示する等）
Moving Feature Json※5、6	• 3次元形状の物体の移動データを簡潔に記述する形式として開発され、OGCが移動体データ形式の国際標準として採択している。 • 移動体（人や自動車など）の動的な空間情報を一体的に記録する。
GeoJSON※7、8	• JSONを使用して様々な地理データ構造をエンコードするためのフォーマットである。 • GeoJSONオブジェクトは、GeometryとPropertyを含むFeature、または、複数のFeatureのリストから構成される。Geometryには、Point、LineString、Polygon、等が含まれ、プロパティ（属性）として、オブジェクトのタイトルや説明、ポイントの大きさ・色、線の太さ・色などの情報を持つことができる。
Shapefile※9	• トポロジー構造をもたない空間データの位置と形に関する情報と、その属性情報を格納するためのデータ形式である。位置と形に関する情報は、（座標値のベクトルで構成される）シェープとして格納される。 • 地理空間上の要素をベクター形式であるポイント、ライン、ポリゴンで表現し、各要素に任意の属性を付与できる。
FBX※10	• モデル ジオメトリ、マテリアル テクスチャ、照明、およびアニメーション シーケンスに関連するデータが格納されている。 • 動く3Dオブジェクトの記述に向いている。
MVT (Mapbox Vector Tile)※11	• 画像データをブラウザ上で高速に表示するために開発されたマップタイル形式の考え方をベクターデータ（点、線、面）について適用し、これらのデータの高速配信、クライアントでの高速描画を実現したデータ形式で、個々のオブジェクトへの属性の付与やアプリケーション側での表示スタイル（色）の設定が可能である。 • Mapbox Inc.等により開発されている。

※1 標準製品仕様書
※2 3D都市モデルの導入ガイダンス
※3 https://github.com/CesiumGS/3d-tiles/blob/master/README.md
※4 https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Guide
※5 https://www.aist.go.jp/aist_j/press_release/pr2020/pr20200602_2/pr20200602_2.html
※6 http://docs.opengeospatial.org/is/19-045r3/19-045r3.html
※7 https://geojson.org/geojson-spec.html
※8 https://docs.geolonia.com/geojson/
※9 https://www.esrij.com/getting-started/learn-more/shapefile/
※10 https://knowledge.autodesk.com/ja/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2019/JPN/3DSMax-Data-Exchange/files/GUID-26E80277-1645-4C4E-A6B2-44399376490F-htm.html
※11https://github.com/madefor/vector-tile-spec/blob/master/2.1/README.md

表　データ変換パターンの例

データ区分		元データ形式	変換後データ形式	データ変換の留意点
建物関連	建物	CityGML	3DTiles	・元データ（CityGML）の品質が担保されていれば、正常の3DTilesへの変換は可能 ・建物属性のコード（数値）は、文字列に変換する処理を追加
	都市計画	CityGML	GeoJSON、MVT	・ポリゴン内に中空の部分がないように分割処理が必要 ・分割し過ぎるとデータ量が肥大化し、動作性能が低下
	道路	CityGML	GeoJSON、MVT	・同上
	洪水浸水想定区域	CityGML	3DTiles	・元データ（CityGML）を3DTilesに変換すると描画性能が低下する場合がある ・浸水域が広い場合（例：東京都の荒川）には描画性能を考慮しデータを分割するか、CityGMLを作成中に発生する3D Shapeファイルを用いて3DTilesに変換する方法もある
	土砂災害警戒区域	CityGML	GeoJSON、MVT	・土砂災害特別警戒区域は土砂災害警戒区域の内側に設定されているため、両者のデータを任意に選択する場合、重なり部分の土砂災害警戒区域をくり抜いて作成する必要がある
	津波浸水想定区域	CityGML	CZML	・洪水浸水想定区域と同様
	内水・高潮想定区域	CityGML	CZML	同上
BIM	建物	CityGML	3DTiles	・元データの構成部材を確認し、変換後の着色方法を事前に規定することが必要 ・建築物に絶対座標が付与されていない場合には、１か所基準点を定め、絶対座標を付与したのち、建物の１面の方位角を規定、PLATEAU VIEW上に位置合わせを行う
	建物、地下空間	FBX	3DTiles	同上
動的データ	人流	JSON,CSV	CZML	・データ量が非常に多く、描画方法を事前検討した上で、必要なデータを絞り込み変換
	GPSデータ	JSON,CSV	CZML	・高層ビルが多いエリアでは、位置精度が低下し、異常な動き（例：バスが後退する）をすることから、データクリーニング処理が必要

第2章　実証環境の構築手順

2.1 実証環境構築の前提

本章では、実証環境「PLATEAU VIEW」のクローンを構築するための手順を説明する。本手順は、公開されているPLATEAU VIEWと同様にAmazon Web Service（以下、「AWS」という。）の利用を前提とした。重畳データの変更やユーザーインターフェースの改変、機能の追加などを行う場合は、Project PLATEAUのGitHub（https://github.com/Project-PLATEAU）のドキュメントを参照のこと。

2.1.1 作業環境の準備

作業するマシンに以下の環境を準備する。本手順ではWindows10におけるプロセスを説明する。

- Node.js
- npm
- yarn
- awscli
- Git

（１）Node.js

JavaScriptは通常ブラウザで実行するが、これをサーバ側で実行するためにNode.jsを使用する。Node.jsのインストールは下記の手順で行う。Node.jsに関する説明はNode.jsのウェブサイト（https://nodejs.org/ja/about/）を参照のこと。

Node.jsのウェブサイト（https://nodejs.org/ja/download/）からお使いの使用環境に対応したインストーラ（ v10.0 以降）を入手し、実行する。

インストールが完了したら、コマンドプロンプトを起動して、以下のコマンドを実行し、動作を確認する。

```
node --version
```

```
C:¥Users¥info>node --version
v14.17.6
```

インストールしたバージョンと同じ数字であることを確認。

（２）npm

npmは（１）でインストールしたNode.jsのモジュールを管理するツールで、Node.jsと同時にインストールされるので、ここではインストールされたnpmのバージョンを表示して動作を確認する。本手順の作業を行うためには、v6.0 以降が必要である。Node.jsと同様にコマンドプロンプトを使ってバージョンを確認する。

```
npm --version
```

v6.0 以降であることを確認。

（３）yarn

近年のJavaScriptアプリケーションは複数のJavaScriptファイルで構成されており、その中には再利用可能な部品（node modules）がある。yarnや前出のnpmはこれらの部品のインストールを管理するもので、インストールすべき部品が記述されたpackages.jsonに従って、yarn installやnpm installといったコマンドでインストールを実行することができる。また、yarnやnpmを使えば、このpackages.jsonの内容を編集することもできる。

yarnは以下のコマンドでインストールし、バージョンを表示して動作を確認する。

```
npm install –g yarn
```

v1.19.0 以降であることを確認。

（４）awscli

awscli (AWS Command Line Interface)は、AWSをコマンド入力で操作できるようにするインターフェースである。AwscliについてはAWSのドキュメント（https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-welcome.html）を参照のこと。

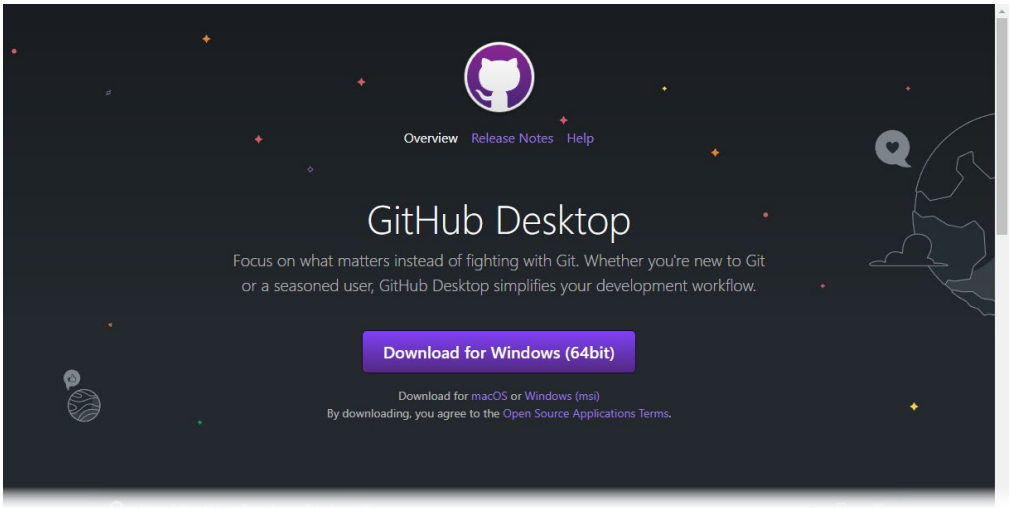
Windowsには、インストーラ（https://awscli.amazonaws.com/AWSCLIV2.msi）が用意されているので、これを利用してインストールを実施する。上記のリンクからダウンロードしたMSIファイルをダブルクリックしてインストールを実行し、コマンドプロンプトからバージョンを表示して動作を確認する。

```
aws --version
```

aws-cli/で始まるメッセージが表示されればインストールが成功。

（５）Git

Gitは今回のインストールに必要なパッケージをPLATEAU VIEWのGitHubから取得する際に使用する。GitのインストールはGitHub Desktopのホームページ（https://desktop.github.com/）からインストーラを入手して実施する。



これまでと同様、コマンドラインでバージョンを表示してインストールを確認する。

```
git --version
```

2.1.2 AWSの準備

この手順ではAWSアカウントの作成については割愛するが、参考となるAWSのドキュメントのURLを示す。（https://aws.amazon.com/jp/register-flow/）

（１）AWSプロファイルの用意

まず、上記で作成したAWSアカウントの情報を使って、AWSマネジメントコンソール（https://aws.amazon.com/jp/console/）にログインし、サービス一覧からIAMを選択する。さらに、IAMの画面から「ユーザー」を選択し、　一覧に表示されるユーザーの中からAdministratorAccess権限を持っているユーザーを選択する（この作業のためにアカウントを作られた場合は、表示されているユーザーがAdministratorAccess権限を持つユーザです）。さらに「認証情報」タブを選択すると「アクセスキーの作成」ボタンが表示されるので、これをクリックする。ここで表示されるアクセスキーIDとシークレットアクセスキーを使用して以下の設定を行う。この情報はここでしか確認できないので、手元に控えるか、「.csvファイルのダウンロード」ボタンを使ってcsvファイルとして保管する。ここで取得したアクセスキーIDとシークレットアクセスキーのセットをAWSプロファイルと呼ぶ。

次に、コマンドプロンプトを開いて、以下のコマンドを実行し、プロファイルを設定する。ここでは「plateau」というプロファイル名で作成する。

```
aws configure –profile plateau
```

AWS Access Key ID、AWS Secret Access Key、Default region name、Default output formatの入力を順次求められるので、AWS Access Key IDとAWS Secret Access Keyについては先ほど作成したものを入力し、Default region nameでは「ap-northeast-1」、Default output formatは「json」を入力する。さらに、以降の作業でこのプロファイルを自動的に読み込むため、環境変数に登録する。

```
setx AWS_PROFILE plateau
```

（２）AWS Route53

AWSコンソールを使用して、AWS Route53のパブリックホストゾーンを作成する。作成の方法は下記のAWSドキュメントを参照のこと。AWS Route53は、サーバやロードバランサをドメインに割り当てるために使用する、AWS製のDNSのようなものである。ホストゾーンとはAWS Route53の用語で、例えばplateauview.mlit.go.jpや hogehoge.mlit.go.jpというような、同じルートドメインのDNSレコードを管理するスペースのことをいう。また、ホストゾーンにはパブリックとプライベートがあり、パブリックはインターネット上に公開されるもので、プライベートは閉じられたネットワーク（VPC, Virtual Private Cloud）のDNS管理スペースとなる。

https://docs.aws.amazon.com/ja_jp/Route53/latest/DeveloperGuide/AboutHZWorkingWith.html

（３）EC2（Amazon Elastic Compute Cloud）のキーペア

EC2は今回PLATEAU VIEWをインストールするサーバの名称で、ここにSSHでアクセスするためには公開鍵と秘密鍵のキーペアを取得する必要がある（今回のインストールではSSHでのアクセスはしないが、EC2を立ち上げる際の必須のパラメータなので取得する）。キーペアの作成方法に関するドキュメントは下記の通りだが、次のコマンドを実行して取得する。

https://docs.aws.amazon.com/ja_jp/ja_jp/cli/latest/userguide/cli-services-ec2-keypairs.html

```
aws ec2 create-key-pair --key-name MyKeyPair --query ‘KeyMaterial’ --output text >
MyKeyPair.pem
```

（４）AWS SSL Certificate

AWS Certificate Manager（https://aws.amazon.com/jp/certificate-manager/）を使用して画面に従って証明書をリクエストし、SSL/TLS証明書（AWS SSL Certificate）を作成する。

SSL証明書はウェブサイトをhttpsで公開するために必要で、そのサイトの公開元が実在することを証明するものである。

（５）S3バケット

AWSマネジメントコンソール（https://aws.amazon.com/jp/console/）からS3を選択し、「バケットを作成」ボタンをクリックし、画面に従ってAWS S3バケットを作成する。

AWS S3バケットはAWSのファイルストレージサービスで、PLATEAU VIEWのインストールの際に使用するPLATEAU VIEWのアプリケーションパッケージを圧縮したものを保管するために使用する。先ほど設定したEC2インスタンス（サーバ）がこの圧縮されたパッケージをダウンロードして展開し、PLATEAU VIEWのアプリケーションを起動する。

2.2 インフラ構築とデプロイ

2.2.1 TerriaMapのクローン

以下のコマンド（git clone）を使ってProject-PLATEAUのGitHubからTerriaMapを作業マシンにクローンする。2行目のcdは、クローンしたディレクトリ（TerriaMap）に移動するためのコマンド。

```
git clone -b plateauview git@github.com:Project-PLATEAU/PLATEAU-VIEW.git
cd TerriaMap
```

2.2.2 TerriaJSのクローン

以下のコマンド（mkdir）を使って、2.2.1でクローンした”TerriaMap”ディレクトリ内に”packages” ディレクトリを作成し、そこにProject-PLATEAUのGitHubからTerriaJSを作業マシンにクローンする。

```
mkdir packages && cd packages
git clone -b plateauview git@github.com:Project-PLATEAU/terriajs.git
cd ../../
```

2.2.3 設定ファイルの編集

（1）TerriaMap/package.jsonの編集

2.2.1, 2.2.2で作業マシンにクローンしたファイルの内、”TerriaMap”直下にあるpackage.jsonを編集する。package.jsonは2.1.2で準備したAWSの環境に関する情報を記述するファイルで、変更する各項目の内容は「表　package.jsonの項目について」に示す通り。

表　package.jsonの項目について

項目名	内容
packageName	アプリケーションのパッケージ名（任意で設定。次ページの例ではplateauview）
awsS3PackagesPath	アプリケーションのパッケージを保管するS3のパス（2.1.2（5）で作成したバケットのURL：東京リージョンで作成していればhttp://s3-ap-northeast-1.amazonaws.com/バケット名）
awsRegion	AWS のリージョンID（東京リージョンで問題なければ次ページの例と同じ：ap-northeast-1）
awsEc2InstanceType	EC2のインスタンスタイプ（この例ではt3.smallを使用（p.8参照））
awsEc2ImageID	EC2のAMIのID （Ubuntu 18.04） ※現在のPLATEAU VIEWは上記で動作確認済み
awsKeyName	EC2のキーペア（2.1.2（３）で取得したもの。例にある“ami-00bc9b7f0e98dc134”と“my-great-key-pair”を置き換える。）

（変更例）

```
"config": {
  "packageName": “plateauview",
  "awsProfile": "terria",
  "awsS3PackagesPath": "s3://my-great-bucket/viewer",
  "awsRegion": "ap-northeast-1",
  "awsEc2InstanceType": "t3.small",
  "awsEc2ImageId": "ami-00bc9b7f0e98dc134",
  "awsKeyName": "my-great-key-pair",
```

（2）TerriaMap/deploy/aws/stack.json の編集

”TerriaMap/deploy/aws” ディレクトリにあるstack.jsonを編集する。stack.jsonは2.1.2で準備したRoute53のパブリックホストゾーンおよびSSL/TLS証明書（AWS SSL Certificate） に関する情報を記述するファイルで、変更する各項目の内容は「表　stack.jsonの項目について」に示す通り。

表　stack.jsonの項目について

項目名	内容
Parameters.HostedZoneName.Default	Rout53のパブリックホストゾーン
SSLCertificateId	AWS SSL CertificateのARN

2.2.4 nodejs依存モジュールのインストール

下記のコマンドを実行し、nodejs依存モジュールをインストールする。このコマンド（yarn）を実行することで、packages.jsonのdependenciesやdevDependenciesに記述された部品がインストールされる。

```
yarn
```

2.2.5 インフラ構築とデプロイ

下記のコマンドを実行し、インフラ構築（サーバ、ロードバランサの整備とDNSレコードの構築）とデプロイを完了する。

```
export NODE_OPTIONS=--max_old_space_size=4096
yarn deploy-without-reinstall
```

これでPLATEAU VIEWのコピーが完成する。表示データを変更する場合は、2.5カタログの編集を参照。

2.3 Dockerを用いた環境構築

AWSを使わずに実証環境を構築する方法として、Dockerを使用した方法を示す。Dockerファイル及び使用するソースコードはProject-PLATEAUのGitHub（https://github.com/Project-PLATEAU/plateau-view-docker-example）から入手する。

2.3.1 前提条件

この手順による実証環境の構築では、構築する環境にDockerがインストールされているものとする。

2.3.2 環境構築の手順

（1）PLATEAU VIEWのソースコードのクローン

下記のコマンドを実行して、PLATEAU VIEWのソースコードをクローンする。

```
cd terria/src
git clone https://github.com/Project-PLATEAU/PLATEAU-VIEW.git TerriaMap
cd TerriaMap
mkdir packages && cd packages
git clone https://github.com/Project-PLATEAU/terriajs.git terriajs
cd ../../../
```

（2）Dockerイメージのビルド

本プロジェクトのルートディレクトリ（docker-compose.ymlがあるディレクトリ）で以下を実行する。

```
docker compose build
```

（3）PLATEAU VIEWの起動

下記を実行し、PLATEAU VIEWを起動すると、ポート80でPLATEAU VIEWが見れる。

```
docker compose up –d
docker compose exec terria /app/node_modules/.bin/pm2 start /app/ecosystem-production.config.js --update-env --env production
```

（4）PLATEAU VIEWの停止

下記を実行し、PLATEAU VIEWを停止する。

```
docker compose down
```

2.4 データ変換の方法

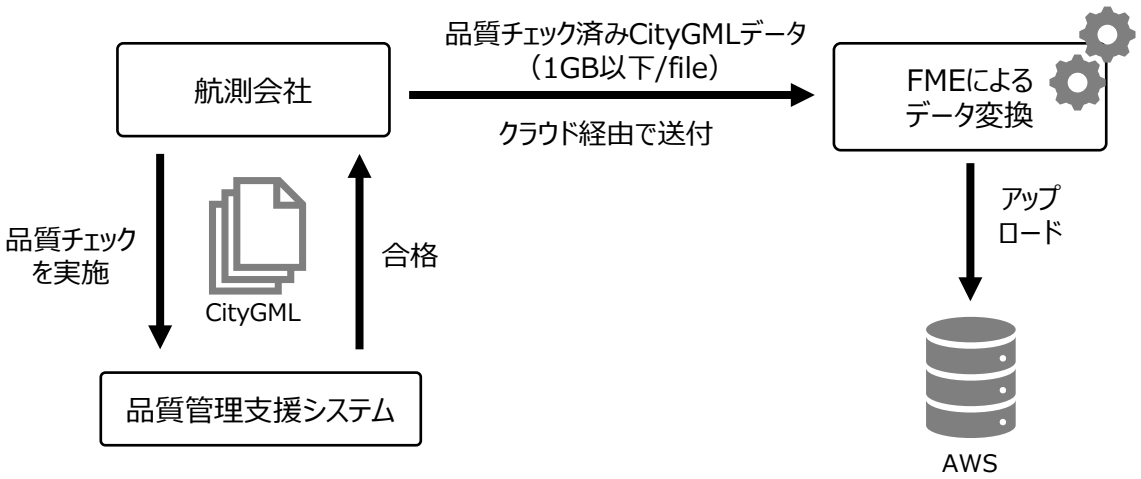
2.4.1 データ変換のワークフロー

PLATEAU VIEWにおけるデータ変換のワークフローを説明する。

前述のとおり、PLATEAU VIEWでは、CityGML形式の建物等の都市データをFMEを使用して3DTilesに変換し、Cesiumで表示している。データ変換に使用するFMEの変換ロジックは、CityGMLデータが「標準製品仕様書」に定められた要求品質を満足していることを前提として構築した。

データ変換の確実性を高めるため、Project PLATEAUでは、データ整備担当者が作成したCityGMLデータの品質評価を品質管理支援システムを用いて行い、合格したものをFMEにより3DTilesに変換するワークフローを構築した。

図　本事業でのデータ変換のワークフロー



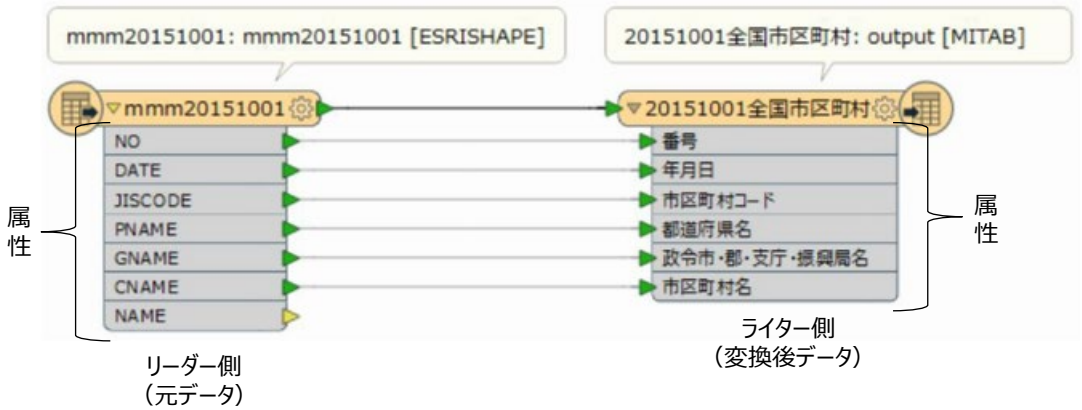
<u>地方自治体向け留意点</u>

同じデータ形式でも内部構造が異なるデータが混在すると、FME変換における変換ロジックを一貫して適用できず、作業工程上の負担となる。このため、CityGMLを3DTilesに変換する場合等は、統一的な仕様（標準製品仕様）による品質チェックの過程をワークフローに取り入れる必要がある。

2.4.2 FMEによるデータ変換

FMEによるデータフォーマットの変換は、リーダーフィーチャータイプ（元データ）とライターフィーチャータイプ（変換後データ）の間、及びそれらの個々の属性の間の対応づけを定義し（マッピング）、ワークスペースを実行することで実現する。FMEは、CityGML、3DTiles等のデータ形式の変換ロジックのためのライブラリも有している。ユーザーは、ライブラリの中から、元データの形式と変換後データの形式を選択し、ワークスペースを実行する。

図　FMEによるデータ変換（マッピング）

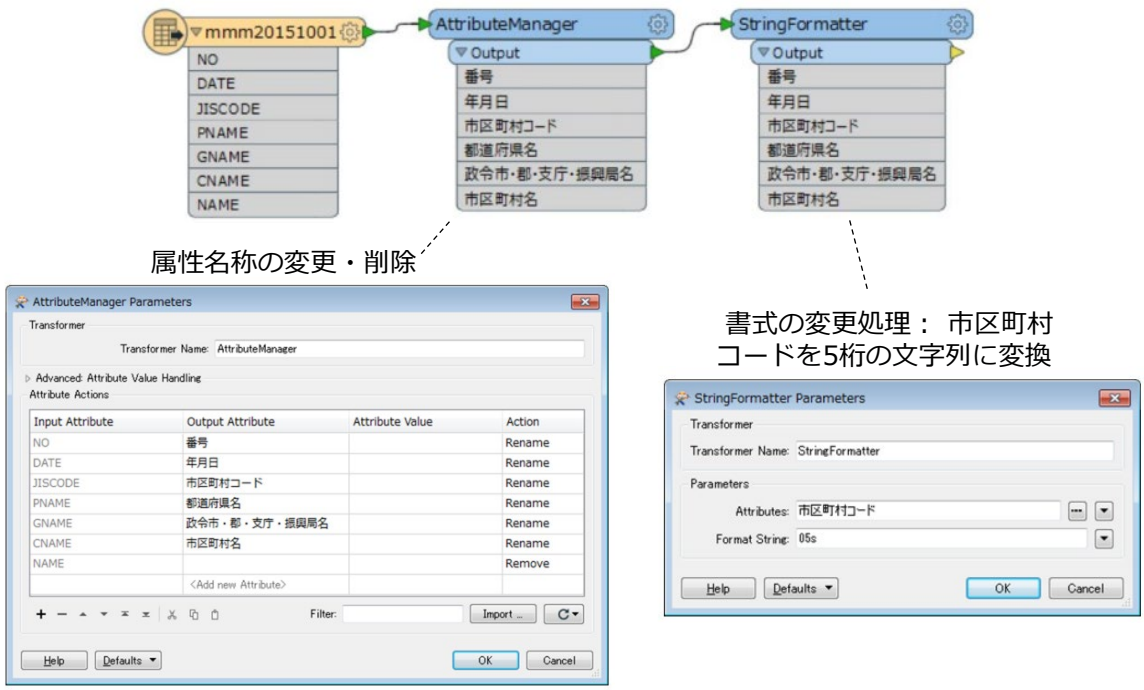


リーダー側とライター側の属性を直接マッピング可能な変換ではなく、データの読み込みから書き出しまでの間に何段階かのデータ変換を行う場合、トランスフォーマーを使用する。トランスフォーマーは、各属性のデータに対して、名称や書式等の変更処理をユーザーが記述できるものである。

なお、PLATEAU VIEWで用いたFMEの変換ロジックは以下のURLで公開している。

https://www.geospatial.jp/ckan/dataset/plateau-tokyo23ku

図　FMEによるデータ変換の例



2.5 カタログの編集

PLATEAU VIEWで表示するデータセットは、JSONで記述されたカタログファイルで定義する。本マニュアルに沿って構築されたPLATEAU VIEWには、公開版と同じ構成のカタログファイルが読み込まれているが（wwwroot/init/plateauview.json）、表示するデータセットの追加など、変更が必要な場合は、このカタログファイルの編集が必要となる。カタログファイルで設定できる内容は、データセットのタイトル（下図①）、説明文（下図②）、属性表、凡例の表示の他、一部のデータタイプについては、属性値による色の指定の他、地物の絞込の機能の指定なども行うことができる。

PLATEAU VIEWで使用しているカタログファイルを簡単に編集できるようにするためのツール、カタログファイルジェネレーターをProject-PLATEAUのGitHub（https://github.com/Project-PLATEAU/plateau-catalog-generator）で公開している。このツールを使えば、表示したいデータセットのタイトルや参照先を記入したCSVファイルから、PLATEAU VIEWで読み込み可能なJSONファイルを生成することができる。ツールの使用方法など詳細は、上記GitHubのチュートリアルを参考にする。

カタログファイルで設定できる項目は多岐にわたるため、カタログファイルジェネレーターのチュートリアルは、PLATEAU VIEWで使用しているファイルタイプと機能に絞った説明となっているが、網羅的な説明はTerriaJSのドキュメントが詳しい（https://docs.terria.io/guide/customizing/initialization-files/）。

第３章　ユーザーマニュアル

3.1 PLATEAU VIEWのUI機能の利用方法

基本的な利用方法として、地図選択、データの選択・表示、重畳に関する一連の基本操作について説明する。

図　PLATEAU VIEWの基本的なUI機能

UI機能	内容
Add Data	Data Catalogue画面の起動
Data Catalogue	データの選択
My Data	ローカル及びWeb上のデータをアップロード※
ワークベンチ	Data Catalogueで選択したデータの表示
Map Settings	背景地図の選択・地図機能の設定
Share / Print	URL生成と印刷
Story	ストーリーエディタの起動（複数のポイントでシーン設定をして保存、ポイントを移動しながら再生ができる）
Help	ヘルプ（マウス操作・日影機能・クリップ機能）の起動
ジャイロスコープ	ビューの回転・傾け、初期位置の表示
ズーム	ズームイン／ズームアウト
現在地表示	自位置に地図を寄せる
歩行者モード	歩行者モードの開始
ご意見・ご要望	コメントまたは質問の送信

※追加可能なデータフォーマットについては次ページ参照。

PLATEAU VIEWでは対応するフォーマットのデータについては、My Dataボタンを選択すると表示されるデータ登録画面にドラッグ＆ドロップすることでカタログに追加し、表示することができる。なお、このドラッグ＆ドロップによるデータの追加はPLATEAU VIEWのメインの画面上でも行える。

図　My Data登録画面（左：初期状態、右：データが登録された状態）

ドラッグ＆ドロップでの登録に対応したフォーマットとその概要は下表の通り。

表　対応フォーマット

対応フォーマット	説明
GeoJSON	GeoJSONはJSONで空間データを扱うためのフォーマットでポイント、ライン、ポリゴンを扱うことができる。 GeoJSON公式サイト：https://geojson.org/
KML/KMZ	KML/KMZは空間データをXMLで記述するフォーマットで、ポイント（マーカー）、ライン、ポリゴンを扱うことができる。 KML入門用ドキュメント： https://developers.google.com/kml/documentation/?hl=ja
CSV	CSVはテキストファイルのフォーマットの一つで、PLATEAU VIEWでは緯度経度情報を持ったポイントデータとして取り込むことができる。読み込むためには少なくとも緯度、経度を表す列が必要で、それぞれの列名が緯度（latitude, lat）、経度（longitude, lon, lng)となっていれば自動的にポイントデータに変換される。
CZML	CZMLはJSON形式を基本とした、PLATEAU VIEWの基盤となっているCesiumで空間データを表現するのに適したフォーマットである。（https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Structure）
GPX	GPXはGPSのデータを扱うためのフォーマットで、軌跡（ライン）とウェイポイント（ポイント）を扱うことができる。 https://www.topografix.com/gpx.asp
Shape	ShapeファイルはGISデータ（2次元）の業界標準フォーマットで、shp、shx、dbfのファイルから構成される。ドラッグ＆ドロップで使用するためにはこれらのファイルをZIP圧縮する必要がある。 ESRIジャパンの解説ページ：https://www.esrij.com/gis-guide/esri-dataformat/shapefile/

(1) 背景地図の選択・地図機能の設定

- 画面右上の[Map Settings]から、背景地図の選択と地図機能の設定ができる。
- [Map Settings]を押下すると、マップビュー、ベースマップの選択及び地図機能に関するチェックボックスが表示され、背景地図の選択と地図機能の設定ができる。

図　地図選択・地図機能設定画面（[Map Settings]）

マップビューは以下を選択可能

- 地図を3D地形で表示
- 地図を3D地形なしで表示
- 地図を2Dで表示

地下を隠す

- 地下のオブジェクトを表示しない

地下に入る

- 視点の移動時に地面の下に移動できるようにする

ベースマップは以下を選択可能

- 空中写真
- 空中写真（Bing）
- 地理院地図
- Dark Matter

タイムライン

- 画面下部に表示した日の午前0時から午後12時までの24時間のタイムラインが表示される。日影表示機能と併用することで、日影を表現する時刻を任意に設定できる。なお、ワークベンチに時系列データが追加されている間は、この指定に関係なくタイムラインが表示される。

画像の最適化

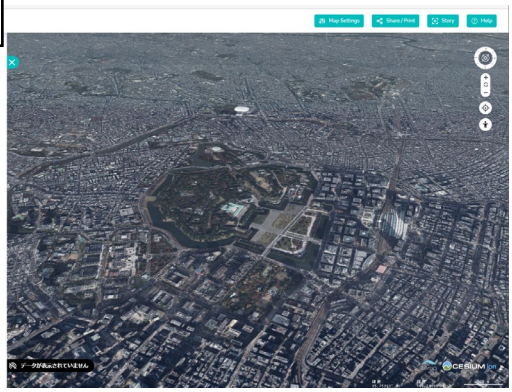
- オンにすると、描画する解像度がお使いのデバイスの解像度に合わせて表現される。お使いのモニタの解像度が低い場合は、オンにすると描画の改善が期待される。

ラスターデータの質

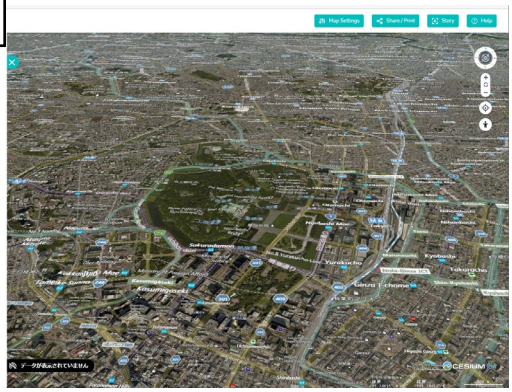
- 質の方にスライダを移動させると、ラスタデータ及びポイントクラウドデータで解像度が高くなるがマシンのスペックによっては描画速度が遅くなる。パフォーマンスの方にスライダを移動させると解像度が低くなるが描画速度は速くなる。

図　選択可能な地図

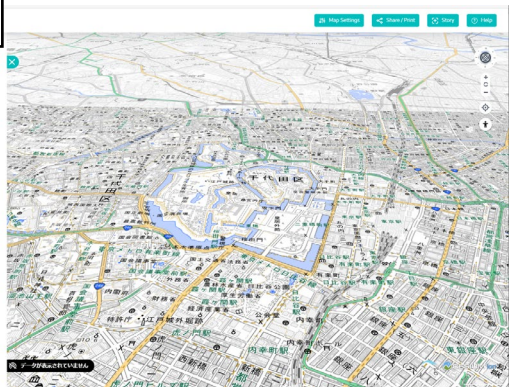
空中写真



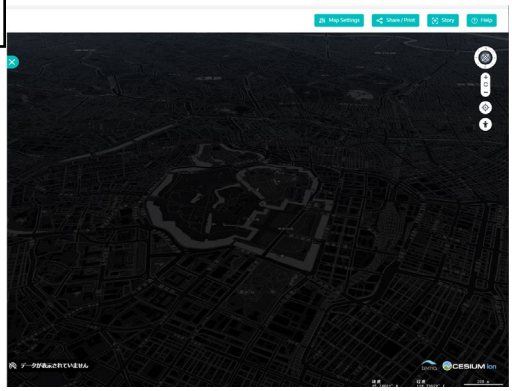
空中写真
（Bing）



地理院地図

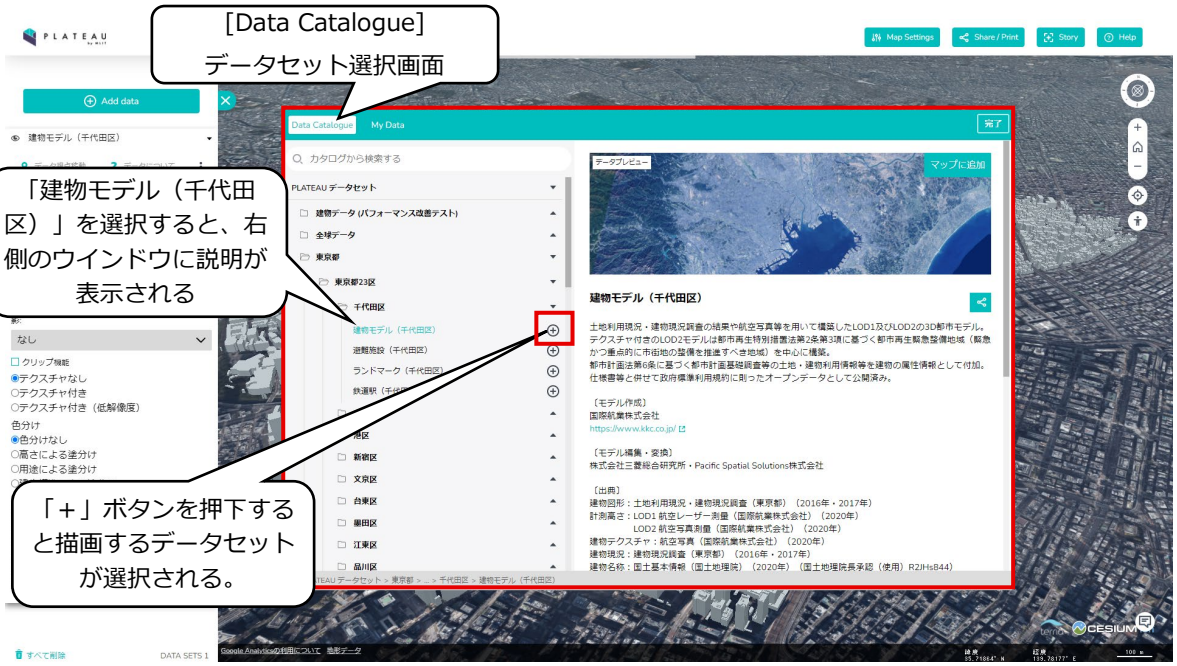


Dark Matter



(2) データの選択・表示

- [Add Data]を押下すると、データセットの選択画面（[データカタログ]）が表示される。
- [データカタログ]においてデータ選択すると、右側のウインドウに説明が表示される。[+]ボタンを押下することで、データを描画することができる。

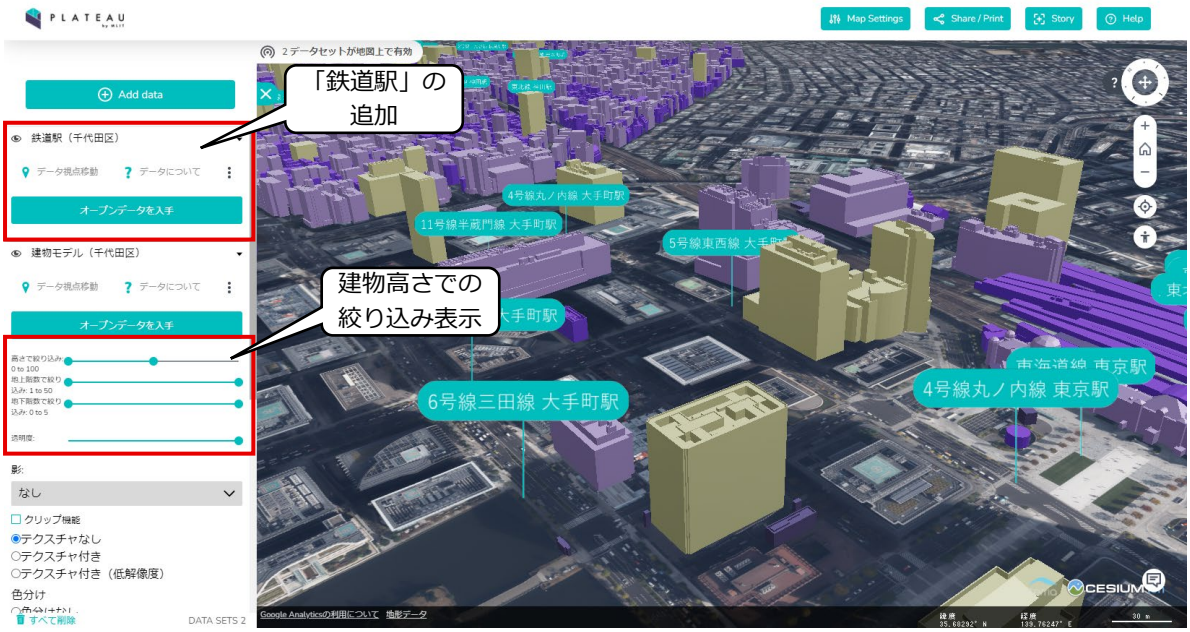


- 選択したデータセットが描画領域に表示され、ワークベンチにはデータセットの情報が表示される。

(3) データの重畳

- [ワークベンチ]から重畳表示したいデータを選択する（下図では「高さによる塗分け」を選択）。
- 描画領域の建物に対して、高さ別の色分けが重畳表示される。

- [Add Data]から[データカタログ]を呼び出し、追加で別のデータセットを表示することも可能である（下図では「鉄道駅（千代田区）」を追加）。
- また、属性の条件を指定することで、描画対象を絞り込む等の機能も有している。
- 下図は、建物高さ100m以下の建物のみを表示し、大手町駅付近を拡大した状態。

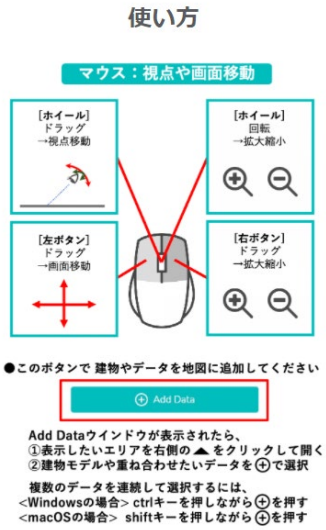


(4) 画面操作

- 描画領域の視点移動や拡大縮小等は、ボタン（画面右上）かマウスで操作することができる。なお、マウスの使い方は、画面右上の[Help]から呼び出すことができる。

図　操作ボタン（画面右上）

図　マウスの使い方（Help）



- また、地図上に描画されたオブジェクト（建物、駅など）をマウスでクリックすることで、属性情報が表示される。