Open Geospatial Consortium

开放地理空间联盟

Submission Date: 2018-06-04

提交日期:2018-06-04

Approval Date: TBD

批准日期:TBD

Publication Date: TBD

出版日期:TBD

External identifier of this OGC document: TBD

本 OGC 文件的外部标识符:TBD

Internal reference number of this OGC document: 18-053r2

本 OGC 文件的内部参考号:18-053r2

Version: 1.0r2

版本:1.0r2

Category: Candidate OGC Community Standard

类别:候选 OGC 社区标准

Editors: Patrick Cozzi, Sean Lilley, Gabby Getz

编辑:帕特里克·科齐、肖恩·莉莉、加贝·盖兹

3D Tiles Specification 1.0

3D 瓷砖规范 1.0

Copyright notice

版权通知

Warning

警告

This document is an OGC Member endorsed international Community standard.This Community standard was developed outside of the OGC and the originating party may continue to update their work;however, this document is fixed in content.This document is available on a royalty free, non-discriminatory basis.Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

本文件是 OGC 成员认可的国际社会标准。本共同体标准是在 OGC 以外制定的，发起方可以继续更新他们的工作；然而，这份文件的内容是固定的。该文件是免费的，非歧视性的。本文件的收件人被邀请提交他们知道的任何相关专利权的通知以及他们的意见，并提供支持性文件。

The companies listed above have granted the Open Geospatial Consortium (OGC) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version under a Attribution 4.0 International (CC BY 4.0) license (see below).

上述公司已授予开放地理空间联盟(OGC)非独家、免版税、已付费的全球许可，根据归属 4.0 国际(抄送 4.0)许可(见下文)复制和分发本文档以及修改本文档和分发修改版本的副本。

License Agreement

许可证协议

Copyright 2016-2018 Cesium and Open Geospatial Consortium

版权所有 2016-2018 铯和开放地理空间联盟

This Specification is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

本规范根据知识共享属性 4.0 国际许可证(抄送 4.0)获得许可。

Some parts of this Specification are purely informative and do not define requirements necessary for compliance and so are outside the Scope of this Specification.These parts of the Specification are marked as being non-normative, or identified as Implementation Notes.

本规范的某些部分纯粹是信息性的，没有定义合规性所需的要求，因此不在本规范的范围内。规范的这些部分被标记为非规范性的，或被识别为实施说明。

Contents

内容

Table of figures

图表

图 12:一个带有转换子瓦片的瓦片集

Figure 13: Tile JSON properties

图 13:平铺 JSON 属性

Figure 14: A tile bounding volume in red, and a content bounding volume in blue

图 14:红色的图块边界体积和蓝色的内容边界体积

Figure 15: A tileset JSON file with external tileset JSON files

图 15:一个带有外部 tileset JSON 文件的 tileset JSON 文件

Figure 16: A tileset with transforms referencing an external tileset with transforms

图 16:带有转换的颚化符集引用带有转换的外部颚化符集

Figure 17: A root tile and its four children tiles

图 17:根瓦片及其四个子瓦片

Figure 18: Two sibling tiles with overlapping bounding volumes

图 18:两个具有重叠边界体积的兄弟图块

Figure 19: A tileset with an overlapping grid spatial data structure

图 19:具有重叠网格空间数据结构的瓦片区

Figure 20: Feature Table layout

图 20:要素表布局

Figure 21: Feature Table binary body layout

图 21:特征表二进制体布局

Figure 22: Batch Table layout

图 22:批处理表布局

Figure 23: Batch Table binary body layout

图 23:批处理表二进制体布局

Figure 24: Batched 3D Model layout

图 24:批量 3D 模型布局

Figure 25: Instanced 3D Model layout

图 25:实例化三维模型布局

Figure 26: A box in the standard basis

图 26:标准基础中的一个框

Figure 27: A box transformed into a rotated basis

图 27:一个转换成旋转基础的盒子

Figure 28: A quantized volume

图 28:量化的体积

Figure 29: Point Cloud layout

图 29:点云布局

Figure 30: A quantized volume

图 30:量化的体积

Figure 31: Composite layout

图 31:复合布局

Source of the content for this OGC document

本 OGC 文件的内容来源

The majority of the content in this OGC document is a direct copy of the content contained at https://github.com/AnalyticalGraphicsInc/3d-tiles/releases/tag/1.0 (the 1.0 tag of the 3d-tiles repo).No normative changes have been made to the content.This OGC document does contain content not contained in the 1.0 tag of the 3d-tiles repo.

本 OGC 文件中的大部分内容是包含在以下网站的内容的直接副本:https://github . com/analytical graphics Inc/3d-tiles/releases/tag/1.0(3d-tiles repo 的 1.0 标记)。没有对内容进行规范性修改。这份 OGC 文件确实包含了 3d 瓷砖回购 1.0 标签中未包含的内容。

Note: Some elements (such as Vector Data) contained in https://github.com/AnalyticalGraphicsInc/ 3d-tiles (the 3d-tiles repo) have been removed from the OGC document because they are currently under development and not a part of this specification.These elements are identified as future work in this OGC document.

注意:包含在 https://github.com/AnalyticalGraphicsInc/3d-tiles(3d-tiles repo)中的一些元素(如矢量数据)已经从 OGC 文档中删除,因为它们目前正在开发中,不属于本规范的一部分。这些要素在本 OGC 文件中被确定为今后的工作。

Validity of content

内容的有效性

The Submission Team has reviewed and certified that the "snapshot" content in this Community Standard is true and accurate.

提交团队已经审核并证明了本社区标准中的"快照"内容是真实和准确的。

Preface

前言

Bringing techniques from graphics research, the movie industry, and the game industry to 3D geospatial, 3D Tiles defines a spatial data structure and a set of tile formats designed for 3D, and optimized for streaming and rendering.

3D 切片将图形研究、电影行业和游戏行业的技术引入 3D 地理空间，定义了一个空间数据结构和一组专为 3D 设计的切片格式，并针对流和渲染进行了优化。

NOTE: This draft policy does not address the issue of how the OGC migrates from WKT for CRS version 2 –CRS2 - (ISO 19162 and OGC 12-063r5 Geographic information - Well-known text representation of coordinate reference systems) to any subsequent future versions.

注:本政策草案不涉及 OGC 如何从 WKT 迁移至任何后续版本的 CRS 版本 2–CRS 2-(国际标准化组织 19162 和 OGC 12-063r5 地理信息-坐标参考系统的众所周知的文本表示)。

If approved, the contents of this document regarding coordinate reference systems will be updated as needed to ensure compatibility.

如果获得批准，本文件中关于坐标参考系统的内容将根据需要进行更新，以确保兼容性。

Future work

未来的工作

The 3D Tiles community anticipates that revisions to this Community Standard will be required to prescribe content appropriate to meet new use cases.These use cases may arise from either (or both) the external user and developer community or from OGC review and comments.Further, future revisions will be driven by any submitted change requests that document community uses cases and requirements.

3D 瓦片社区预计将需要对该社区标准进行修订，以规定适合新用例的内容。这些用例可能来自外部用户和开发人员社区中的一个(或者两者都有)，或者来自 OGC 的评审和评论。此外，未来的修订将由任何提交的变更请求驱动，这些变更请求记录了社区使用案例和需求。

Additions planned for future inclusion in the 3D Tiles Specification (future work) are described at https://github.com/AnalyticalGraphicsInc/3d-tiles/issues/247.

计划未来纳入 3D 切片规范(未来工作)的新增内容，请访问 http://git hub . com/AnalYSiS graphics Inc/ 3D 切片/issues/247。

Scope

范围

3D Tiles is designed for streaming and rendering massive 3D geospatial content such as Photogrammetry, 3D Buildings, BIM/CAD, Instanced Features, and Point Clouds.It defines a hierarchical data structure and a set of tile formats which deliver renderable content.3D Tiles does not define explicit rules for visualization of the content;a client may visualize 3D Tiles data however it sees fit.

3D 切片设计用于流式传输和渲染大量 3D 地理空间内容，例如摄影测量、3D 建筑物、BIM/计算机辅助设计、实例化要素和点云。它定义了分层数据结构和一组交付可呈现内容的平铺格式。3D 切片没有为内容的可视化定义明确的规则；客户端可以可视化 3D 切片数据，只要它认为合适。

A 3D Tiles data set, called a tileset, contains any combination of tile formats organized into a spatial data structure.

3D 图块数据集称为图块集，包含组织成空间数据结构的图块格式的任意组合。

3D Tiles are declarative, extendable, and applicable to various types of 3D data.The following tile formats have been specified as part of this document:

3D 切片是声明性的、可扩展的，适用于各种类型的 3D 数据。以下图块格式已被指定为本文档的一部分：

Batched 3D Model

批量 3D 模型

Instanced 3D Model

实例化三维模型

Point Cloud

点云

Composite

复合材料

This document also describes 3D Tile Styles, a declarative styling specification which may be applied to tilesets.

本文档还描述了 3D 平铺样式，这是一个可以应用于平铺的声明性样式规范。

The 3D Tiles specification for tilesets, associated tile formats, and the associated styling specification are open formats that are not dependent on any vendor-specific solution, technology, or products.

瓷砖的 3D 瓷砖规范、相关的瓷砖格式和相关的样式规范是开放格式，不依赖于任何特定于供应商的解决方案、技术或产品。

Conformance

顺应

Sections 7 through 11 of this document describe the Objects and Properties required to implement 3D Tiles.Conformance is relative to these elements and as partly expressed via the associated 3D Tiles JSON schema documents located at https://github.com/AnalyticalGraphicsInc/3d-tiles/releases/tag/1.0/specification/schema

本文档的第 7 至 11 节描述了实现 3D 切片所需的对象和属性。一致性与这些元素相关，部分通过相关的 3D 瓦片 JSON 模式文档来表达，这些文档位于

All figures, examples, notes, and background information are non-normative.

所有的数字、例子、注释和背景信息都是非规范性的。

References

参考

Normative

标准的

EPSG: 4979, 2007.http://spatialreference.org/ref/epsg/wgs-84-3/.

EPSG: 4979，2007。http://spatialreference.org/ref/epsg/wgs-84-3/.

IETF RFC2397: The "data" URL scheme, 1998.https://tools.ietf.org/html/rfc2397.

IETF RFC2397:《数据网址方案》，1998 年。https://tools.ietf.org/html/rfc2397.

IETF RFC3629: UTF-8, a transformation format of ISO 10646, 2003.https://tools.ietf.org/html/rfc3629.

IETF RFC3629: UTF-8，国际标准化组织 10646，2003 的转换格式。https://tools.ietf.org/html/rfc3629.

IETF RFC3986: IANA Registration for Enumservice 'XMPP', 2007.https://tools.ietf.org/html/rfc4979.

IETF RFC 3986:enum service ' XMPP '的 IANA 注册，2007 年。https://tools.ietf.org/html/rfc4979.

Khronos Group: glTF 2.0 - Runtime 3D Asset Delivery, 2017.https://github.com/KhronosGroup/glTF/blob/master/README.md.

Khronos 集团:glTF 2.0 -运行时 3D 资产交付，2017 年。https://github.com/KhronosGroup/glTF/blob/master/README.md.

OGC: [OGC 12-063r5] Geographic information - Well-known text representation of coordinate reference systems, 2015.http://docs.opengeospatial.org/is/12-063r5/12-063r5.html.

OGC:[·OGC 12-063 F5]地理信息——坐标参考系统的众所周知的文本表示，2015 年。http://docs.opengeospatial.org/is/12-063r5/12-063r5.html.

W3C: CSS3 Color, 2018.https://www.w3.org/TR/css-color-3/.

W3C: CSS3 颜色，2018。https://www.w3.org/TR/css-color-3/.

Terms and Definitions

术语和定义

Bounding Volume

边界体积

A closed volume completely containing the union of a set of geometric objects.[]

完全包含一组几何对象的并集的封闭体积。[]

Feature

特征

In 3D Tiles, an individual component of a tile, such as a 3D model in a Batched 3D Model or a point in a Point Cloud which contains position, appearance, and metadata properties.

在 3D 切片中，切片的单个组件，如批量 3D 模型中的 3D 模型或点云中包含位置、外观和元数据属性的点。

Geometric Error

几何误差

The difference, in meters, of a tile's simplified representation of its source geometry used to calculate the screen space error introduced if a tile's content is rendered and its children's are not.

图块的源几何图形的简化表示的差异，以米为单位，用于计算如果图块内容被渲染而其子元素没有被渲染时引入的屏幕空间误差。

glTF

glTF

An API-neutral runtime asset delivery format for 3D assets.

3D 资产的不依赖于应用编程接口的运行时资产交付格式。

Hierarchical Level of Detail (HLOD)

层次细节层次

Decreasing the complexity of a 3D representation according to metrics such as object importance or distance from the tile to the observation point or camera position.Generally, higher levels of detail provide greater visual fidelity.[]

根据诸如对象重要性或从图块到观察点或相机位置的距离等度量来降低 3D 表示的复杂性。一般来说，更高层次的细节提供了更高的视觉保真度。[]

Tile

瓷砖

In 3D Tiles, a subset of a tileset containing a reference to renderable content and the metadata, such as the content's bounding volume, which is used by a client to determine if the content is rendered.

在三维图块中，图块集的子集，包含对可渲染内容和元数据(如内容的边界体积)的引用，客户端使用该子集来确定内容是否已渲染。

Tile Content

平铺内容

A binary blob containing information necessary to render a tile which is an instance of a specific tile format (Batched 3D Model, Instanced 3D Model, Point Clouds, or Composite).

一个二进制 blob，包含渲染作为特定图块格式实例(成批 3D 模型、实例化 3D 模型、点云或合成)的图块所需的信息。

Tile Format

平铺格式

The structure or layout of tile content data, (Batched 3D Model, Instanced 3D Model, Point Clouds, or Composite).

图块内容数据的结构或布局(批量 3D 模型、实例化 3D 模型、点云或合成)。

Tileset

Tileset

In 3D Tiles, a collection of 3D Tiles tile instances organized into a spatial data structure and additional metadata, such that the aggregation of these tiles represent some 3D content at various levels of detail.

在 3D 图块中，组织成空间数据结构和附加元数据的 3D 图块实例的集合，使得这些图块的集合以不同的细节级别表示一些 3D 内容。

Screen-Space Error (SSE)

屏幕空间错误

The difference, in pixels, of a tile's simplified representation of its source geometry introduced if a tile's content is rendered and its children's are not.

如果图块的内容被渲染而其子对象没有被渲染，图块对其源几何图形的简化表示的差异(以像素为单位)。

Spatial Coherence

空间相干性

The union of all content of the child tiles is completely inside the parent tile's bounding volume

子图块的所有内容的合并完全在父图块的边界体积内

Style

风格

A set of expressions to be evaluated which modify how each feature in a tileset is displayed

要评估的一组表达式，用于修改颚化符集中每个特征的显示方式

Conventions

约定

No conventions are specified in this document.

本文档中未指定任何约定。

3D Tiles Specification

3D 瓷砖规范

Overview

概观

3D Tiles is designed for streaming and rendering massive 3D geospatial content such as Photogrammetry, 3D Buildings, BIM/CAD, Instanced Features, and Point Clouds.It defines a hierarchical data structure and a set of tile formats which deliver renderable content.3D Tiles does not define explicit rules for visualization of the content;a client may visualize 3D Tiles data however it sees fit.

3D 切片设计用于流式传输和渲染大量 3D 地理空间内容，例如摄影测量、3D 建筑物、BIM/计算机辅助设计、实例化要素和点云。它定义了分层数据结构和一组交付可呈现内容的平铺格式。3D 切片没有为内容的可视化定义明确的规则；客户端可以可视化 3D 切片数据，只要它认为合适。

In 3D Tiles, a tileset is a set of tiles organized in a spatial data structure, the tree.A tileset is described by at least one tileset JSON file containing tileset metadata and a tree of tile objects, each of which may reference renderable content of one of the following formats:

在 3D 图块中，图块集是以空间数据结构(树)组织的一组图块。一个 tileset 由至少一个 tileset JSON 文件描述，该文件包含 tileset 元数据和一个 tileset 对象树，每个 tileset 对象可以引用以下格式之一的可呈现内容:

Format	Uses

Batched 3D Model (b3dm)	Heterogeneous 3D models.E.g. textured terrain and surfaces, 3D building exteriors and interiors, massive models.
Instanced 3D Model (i3dm)	3D model instances.E.g. trees, windmills, bolts.
Point Cloud (pnts)	Massive number of points.
Composite (cmpt)	Concatenate tiles of different formats into one tile.

版式	使用
批量 3D 模型(b3dm)	异构 3D 模型。例如纹理化的地形和表面、3D 建筑外部和内部、大型模型。
实例化三维模型(i3dm)	三维模型实例。例如树木、风车、螺栓。
点云	大量点数。
复合材料(cmpt)	将不同格式的图块连接成一个图块。

A tile's content, an individual instance of a tile format, is a binary blob with format-specific components including a Feature Table and a Batch Table.

图块的内容(图块格式的单个实例)是一个二进制 blob，具有特定于格式的组件，包括要素表和批处理表。

The content references a set of features, such as 3D models representing buildings or trees, or points in a point cloud.Each feature has position and appearance properties stored in the tile's Feature Table, and additional application-specific properties stored in the Batch Table.A client may choose to select features at runtime and retrieve their properties for visualization or analysis.

内容引用一组要素，如表示建筑物或树的三维模型，或点云中的点。每个特征都有存储在图块特征表中的位置和外观属性，以及存储在批处理表中的附加应用程序特定属性。客户端可以选择在运行时选择特征并检索它们的属性用于可视化或分析。

The Batched 3D Model and Instanced 3D Model formats are built on glTF, an open specification designed for the efficient transmission of 3D content.The tile content of these formats embed a glTF asset, which contains model geometry and texture information, in the binary body.The Point Cloud format does not embed glTF.

批量 3D 模型和实例化 3D 模型格式建立在 glTF 之上，GLtf 是一个开放的规范，旨在高效传输 3D 内容。这些格式的图块内容在二进制体中嵌入了 glTF 资产，其中包含模型几何和纹理信息。点云格式不嵌入 glTF。

Tiles are organized in a tree which incorporates the concept of Hierarchical Level of Detail (HLOD) for optimal rendering of spatial data.Each tile has a bounding volume, an object defining a spatial extent completely enclosing its content.The tree has spatial coherence;the content for child tiles are completely inside the parent's bounding volume.

图块被组织在一棵树中，该树结合了层次细节级别(HLOD)的概念，用于空间数据的最佳渲染。每个图块都有一个边界体积，一个定义完全包围其内容的空间范围的对象。树具有空间一致性；子切片的内容完全在父切片的边界体积内。
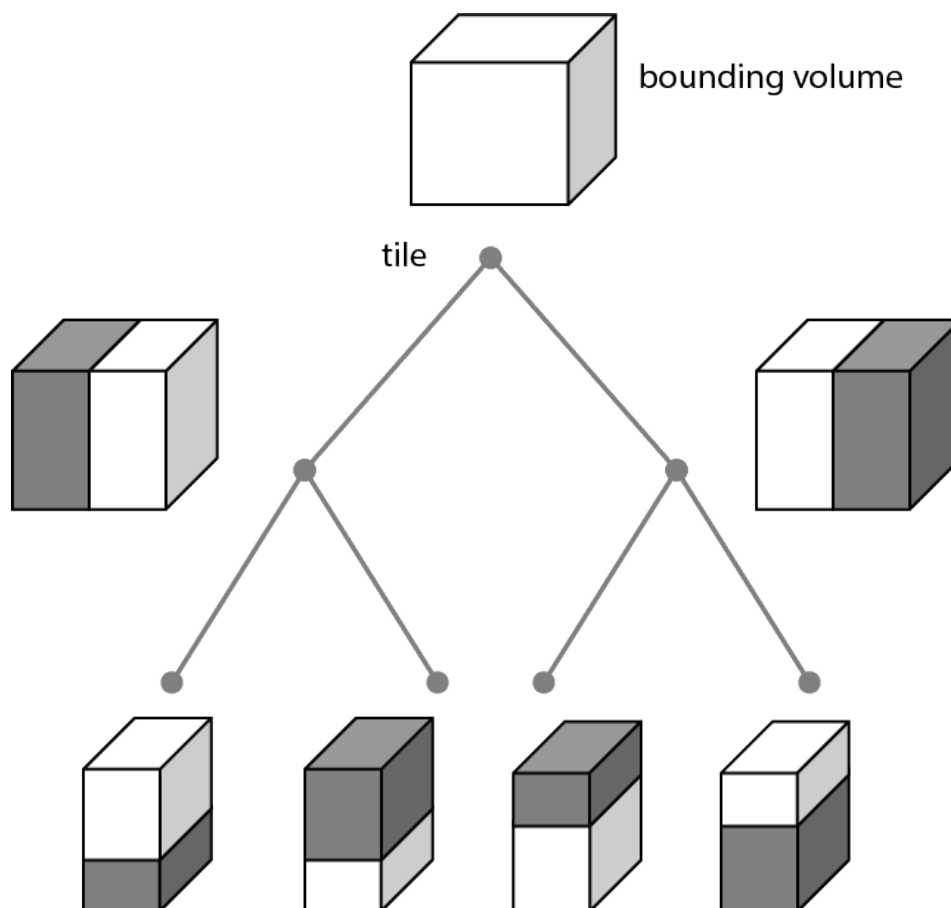
Figure 1: A sample 3D Tiles bounding volume hierarchy

图 1:一个示例 3D 切片包围体层次结构

A tileset may use a 2D spatial tiling scheme similar to raster and vector tiling schemes (like a Web Map Tile Service (WMTS) or XYZ scheme) that serve predefined tiles at several levels of detail (or zoom levels).However since the content of a tileset is often non-uniform or may not easily be organized in only two dimensions, the tree can be any spatial data structure with spatial coherence, including k-d trees, quadtrees, octrees, and grids.

瓦片区可以使用类似于光栅和矢量瓦片区方案(如网络地图瓦片服务(WMTS)或 XYZ 方案)的 2D 空间瓦片区方案，该方案在几个细节级别(或缩放级别)提供预定义的瓦片。然而，由于瓦片区的内容通常是不均匀的，或者可能不容易仅在二维中组织，树可以是具有空间一致性的任何空间数据结构，包括 k-d 树、四叉树、八叉树和网格。

Optionally a 3D Tiles Style, or style, may be applied to a tileset.A style defines expressions to be evaluated which modify how each feature is displayed.

可选地，3D 拼贴样式或样式可以应用于拼贴集。样式定义要评估的表达式，这些表达式修改每个特征的显示方式。

File extensions and MIME types

文件扩展名和 MIME 类型

3D Tiles uses the following file extensions and MIME types.

3D 切片使用以下文件扩展名和 MIME 类型。

Tileset files use the .json extension and the application/json MIME type.

Tileset 文件使用。json 扩展和应用程序/json MIME 类型。

Tile content files use the file type and MIME format specific to their tile format specification, see Tile format specifications.

平铺内容文件使用特定于其平铺格式规范的文件类型和 MIME 格式，请参见平铺格式规范。

Tileset style files use the .json extension and the application/json MIME type.

Tileset 样式文件使用。json 扩展和应用程序/json MIME 类型。

Explicit file extensions are optional.Valid implementations may ignore it and identify a content's format by the magic field in its header.

显式文件扩展名是可选的。有效的实现可以忽略它，并通过标题中的魔法字段来识别内容的格式。

JSON encoding

JSON 编码

3D Tiles has the following restrictions on JSON formatting and encoding.

3D 切片对 JSON 格式和编码有以下限制。

JSON must use UTF-8 encoding without BOM.

JSON 必须使用没有物料清单的 UTF-8 编码。

All strings defined in this spec (properties names, enums) use only ASCII charset and must be written as plain text.

本规范中定义的所有字符串(属性名、枚举)仅使用 ASCII 字符集，必须以纯文本形式书写。

Names (keys) within JSON objects must be unique, i.e., duplicate keys aren't allowed.

JSON 对象中的名称(键)必须是唯一的，即不允许重复的键。

URIs

URIs

3D Tiles uses URIs to reference tile content.These URIs may point to relative external references (RFC3986) or be data URIs that embed resources in the JSON.Embedded resources use the "data" URI scheme (RFC2397).

3D 图块使用 URIs 来参考图块内容。这些 URIs 可能指向相对外部引用(RFC3986)，或者是嵌入 JSON 资源的数据 URIs。嵌入式资源使用"数据"URI 方案(RFC2397)。

When the URI is relative, its base is always relative to the referring tileset JSON file.

当 URI 相对时，它的基础总是相对于引用的颚化符 JSON 文件。

Client implementations are required to support relative external references and embedded resources.Optionally, client implementations may support other schemes (such as http://).All URIs must be valid and resolvable.

客户端实现需要支持相对外部引用和嵌入式资源。可选地，客户端实现可以支持其他方案(例如 http://)。整个 URIs 都必须是有效和可解决的。

Units

单位

The unit for all linear distances is meters.

所有线性距离的单位是米。

All angles are in radians.

所有角度都以弧度为单位。

Coordinate reference system (CRS)

坐标参考系统

3D Tiles uses a right-handed Cartesian coordinate system;that is, the cross product of x and y yields z. 3D Tiles defines the z axis as up for local Cartesian coordinate systems.A tileset's global coordinate system will often be in a WGS 84 earth-centered, earth-fixed (ECEF) reference frame (EPSG 4979), but it doesn't have to be, e.g., a power plant may be defined fully in its local coordinate system for use with a modeling tool without a geospatial context.

3D 图块使用右手笛卡尔坐标系；也就是说，x 和 y 的叉积产生 z。3D 图块将 z 轴定义为局部笛卡尔坐标系的向上。tileset 的全球坐标系通常位于 WGS 84 地球中心、地球固定(ECEF)参考坐标系(EPSG 4979)中，但它并不一定是，例如，电厂可以在其本地坐标系中完全定义，以便与没有地理空间上下文的建模工具一起使用。

An additional tile transform may be applied to transform a tile's local coordinate system to the parent tile's coordinate system.

可以应用附加的图块变换来将图块的局部坐标系变换到父图块的坐标系。

The region bounding volume specifies bounds using a geographic coordinate system (latitude, longitude, height), specifically EPSG 4979.

区域边界体积使用地理坐标系(纬度、经度、高度)指定边界，特别是 EPSG 4979。

Tiles

瓷砖

Tiles consist of metadata used to determine if a tile is rendered, a reference to the renderable content, and an array of any children tiles.

切片由用于确定是否呈现切片的元数据、对可呈现内容的引用以及任何子切片的数组组成。

Geometric error

几何误差

Tiles are structured into a tree incorporating Hierarchical Level of Detail (HLOD) so that at runtime a client implementation will need to determine if a tile is sufficiently detailed for rendering and if the content of tiles should be successively refined by children tiles of higher resolution.An implementation will consider a maximum allowed Screen-Space Error (SSE), the error measured in pixels.

图块被结构化为包含层次细节级别(HLOD)的树，以便在运行时客户端实现将需要确定图块是否足够详细以用于渲染，以及图块的内容是否应该由更高分辨率的子图块连续细化。实现将考虑最大允许屏幕空间误差(SSE)，即以像素为单位测量的误差。

A tile's geometric error defines the selection metric for that tile.Its value is a nonnegative number that specifies the error, in meters, of the tile's simplified representation of its source geometry.The root tile, being the most simplified version of the source geometry, will have the greatest geometric error.Then each successive level of children will have a lower geometric error than its parent, with leaf tiles having a geometric error of or close to 0.

图块的几何误差定义了该图块的选择度量。它的值是非负数，以米为单位指定图块的源几何图形简化表示的误差。根瓦片是源几何的最简化版本，将具有最大的几何误差。那么每一个连续的子级的几何误差将比其父级的小，叶瓦片的几何误差为 0 或接近 0。

In a client implementation, geometric error is used with other screen space metrics—e.g., distance from the tile to the camera, screen size, and resolution— to calculate the SSE introduced if this tile is rendered and its children are not.If the introduced SSE exceeds the maximum allowed, then the tile is refined and its children are considered for rendering.

在客户端实现中，几何误差与其他屏幕空间度量(例如，图块到相机的距离、屏幕大小和分辨率)一起使用，以计算如果渲染了此图块而其子图块没有渲染时引入的 SSE。如果引入的 SSE 超过了允许的最大值，则会细化图块，并考虑渲染其子图块。

The geometric error is formulated based on a metric like point density, tile size in meters, or another factor specific to that tileset.In general, a higher geometric error means a tile will be refined more aggressively, and children tiles will be loaded and rendered sooner.

几何误差是基于像点密度、以米为单位的瓦片大小或特定于瓦片集合的另一个因素等度量来制定的。一般来说，较高的几何误差意味着图块将被更积极地细化，并且子图块将被更快地加载和渲染。

Refinement

精炼

Refinement determines the process by which a lower resolution parent tile renders when its higher resolution children are selected to be rendered.Permitted refinement types are replacement ("REPLACE") and additive ("ADD").If the tile has replacement refinement, the children tiles are rendered in place of the parent, that is, the parent tile is no longer rendered.If the tile has additive refinement, the children are rendered in addition to the parent tile.

细化决定了当选择较高分辨率的子切片进行渲染时，较低分辨率的父切片的渲染过程。允许的细化类型有替换（"替换"）和添加（"添加"）。如果图块具有替换细化，子图块将替代父图块呈现，即父图块不再呈现。如果图块具有附加细化，则除了父图块之外，还会呈现子图块。

A tileset can use replacement refinement exclusively, additive refinement exclusively, or any combination of additive and replacement refinement.

tileset 可以专门使用替换细化，专门使用加法细化，或者加法和替换细化的任意组合。

A refinement type is required for the root tile of a tileset;it is optional for all other tiles.When omitted, a tile inherits the refinement type of its parent.
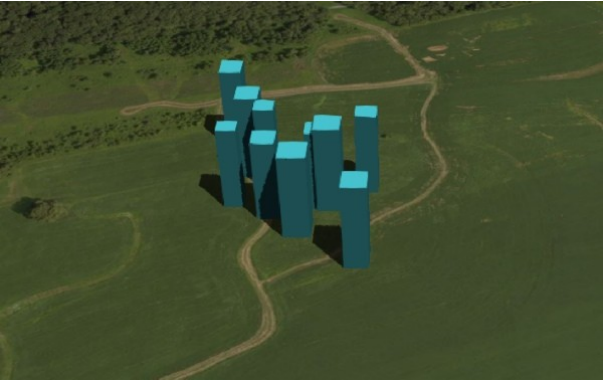
tileset 的根切片需要细化类型；它对于所有其他图块都是可选的。省略时，图块会继承其父图块的细化类型。

Replacement
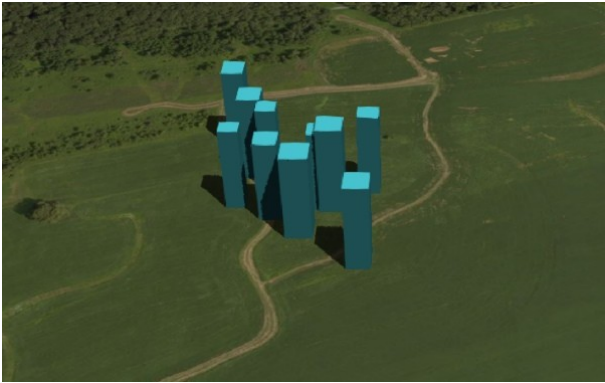
更换

If a tile uses replacement refinement, when refined it renders its children in place of itself.

如果一个图块使用替换细化，当细化时，它将呈现其子元素来代替自身。

Parent Tile	Refined
Figure 2: A parent tile with replacement refinement	Figure 3: A refined child tile of a tile with replacement refinement
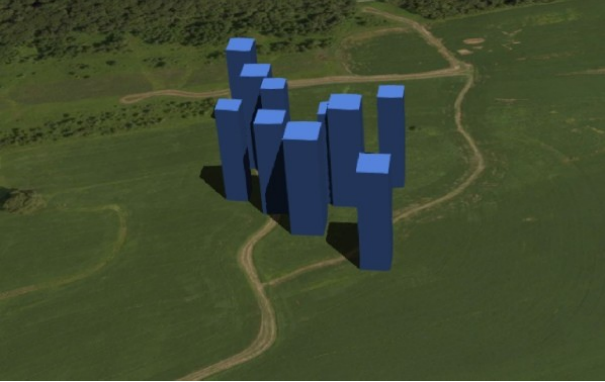
父平铺	精炼的
图 2:带有替换细化的父切片	图 3:带有替换细化的细化子瓦片

Additive

添加剂

If a tile uses additive refinement, when refined it renders itself and its children simultaneously.

<mark>如果图块使用附加细化，当细化时，它会同时渲染自身及其子对象。</mark>

Parent Tile	Refined
Figure 4: A parent tile with additive refinement	Figure 5: A refined child tile of a tile with additive refinement

父平铺	精炼的
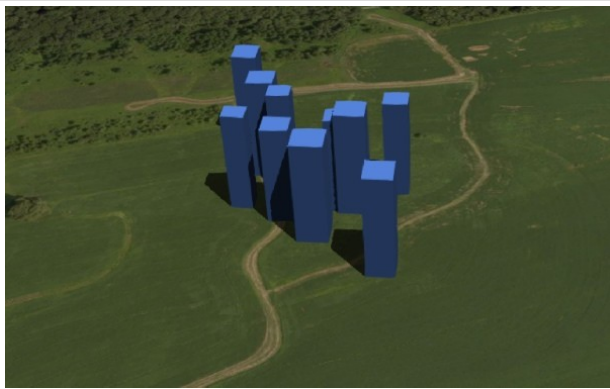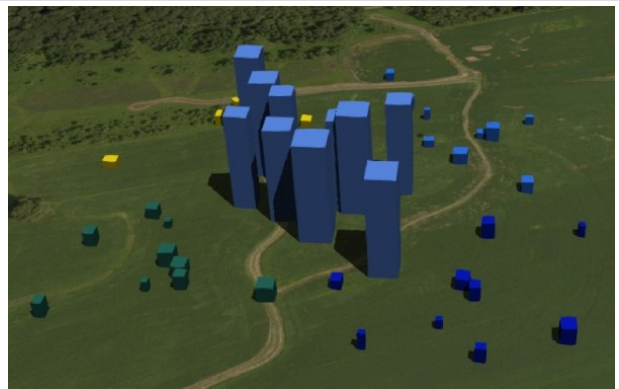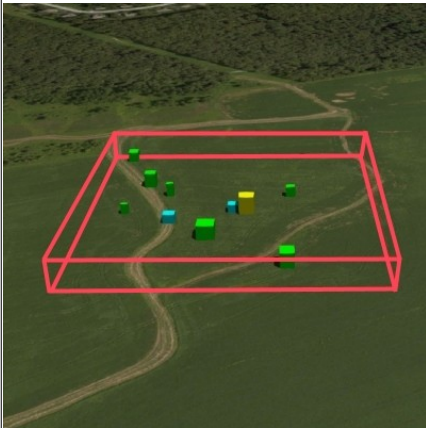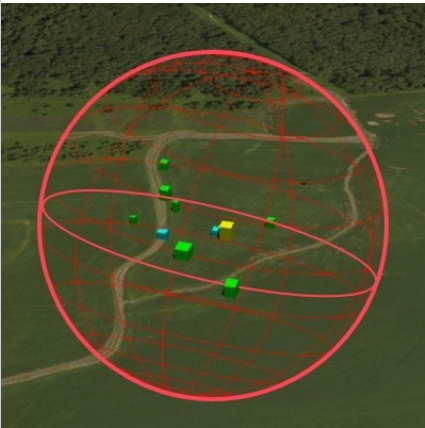
图 4:带有附加细化的父瓦片	图 5:带有附加细化的细化子瓦片

Bounding volumes

边界体积

A bounding volume defines the spatial extent enclosing a tile or a tile's content.To support tight fitting volumes for a variety of datasets such as regularly divided terrain, cities not aligned with a line of latitude or longitude, or arbitrary point clouds, the bounding volume types include an oriented bounding box, a bounding sphere, and a geographic region defined by minimum and maximum latitudes, longitudes, and heights.

边界体积定义包围图块或图块内容的空间范围。为了支持各种数据集(如规则划分的地形、不与纬度或经度线对齐的城市或任意点云)的紧密拟合体积，边界体积类型包括定向边界框、边界球体和由最小和最大纬度、经度和高度定义的地理区域。

Bounding box	Bounding sphere	Bounding region
Figure 6: A bounding box	Figure 7: A bounding sphere	Figure 8: A bounding region

边框	包围球	边界区域

| 图 6:一个边界框 | 图 7:一个边界球体 | 图 8:边界区域 |
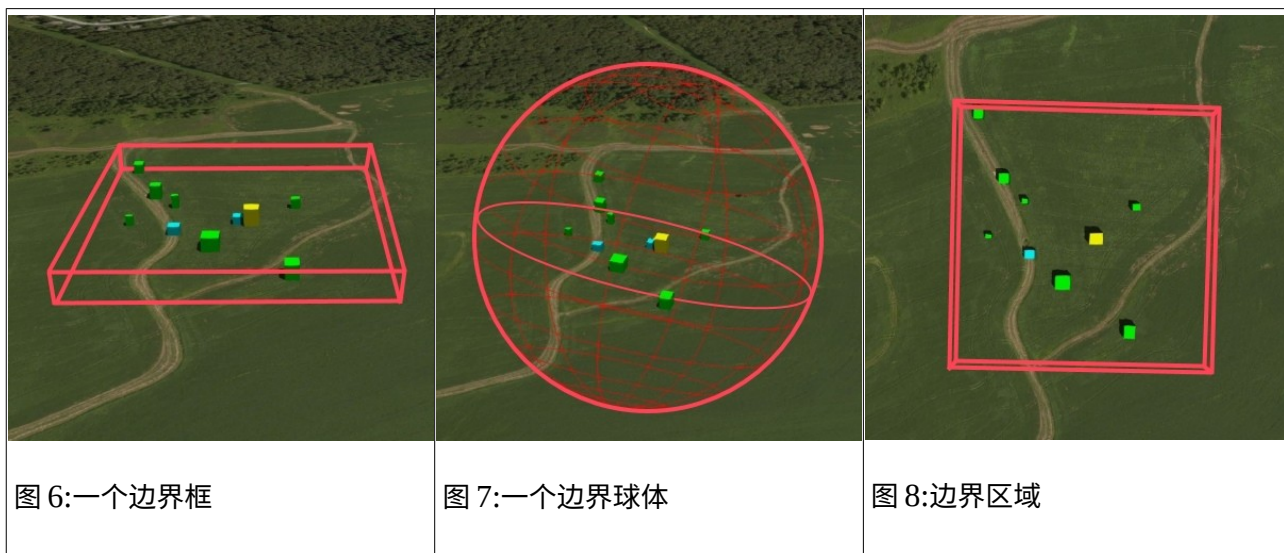
Region

地区

The boundingVolume.region property is an array of six numbers that define the bounding geographic region with latitude, longitude, and height coordinates with the order [west, south, east, north, minimum height, maximum height].Latitudes and longitudes are in the WGS 84 datum as defined in EPSG 4979 and are in radians.Heights are in meters above (or below) the WGS 84 ellipsoid.

boundingVolume.region 属性是一个由六个数字组成的数组，这些数字按照[西、南、东、北、最小高度、最大高度的顺序用纬度、经度和高度坐标来定义边界地理区域]。纬度和经度位于 EPSG 4979 中定义的 WGS 84 基准中，以弧度表示。高度以 WGS 84 椭球面以上(或以下)米为单位。

Figure 9: A bounding region

图 9:边界区域

"boundingVolume": {

" boundingVolume": {

"region": [

【地区】:[

-1.3197004795898053,

-1.3197004795898053，

0.6988582109,

0.6988582109，

-1.3196595204101946,

-1.3196595204101946，

0.6988897891,

0.6988897891，

0,

0，

20

20

]

]

}

}

Box

包厢

The boundingVolume.box property is an array of 12 numbers that define an oriented bounding box in a right-handed 3-axis (x, y, z) Cartesian coordinate system where the z-axis is up.The first three elements define the x, y, and z values for the center of the box.The next three elements (with indices 3, 4, and 5) define the x-axis direction and half-length.The next three elements (indices 6, 7, and 8)

define the y-axis direction and half-length.The last three elements (indices 9, 10, and 11) define the z-axis direction and half-length.

boundingVolume.box 属性是一个由 12 个数字组成的数组，它们在 z 轴向上的右手 3 轴(x，y，z)笛卡尔坐标系中定义了一个定向边界框。前三个元素定义了盒子中心的 x、y 和 z 值。接下来的三个元素(索引为 3、4 和 5)定义了 x 轴方向和半长。接下来的三个元素(索引 6、7 和 8)定义了 y 轴方向和半长。最后三个元素(索引 9、10 和 11)定义了 z 轴方向和半长度。
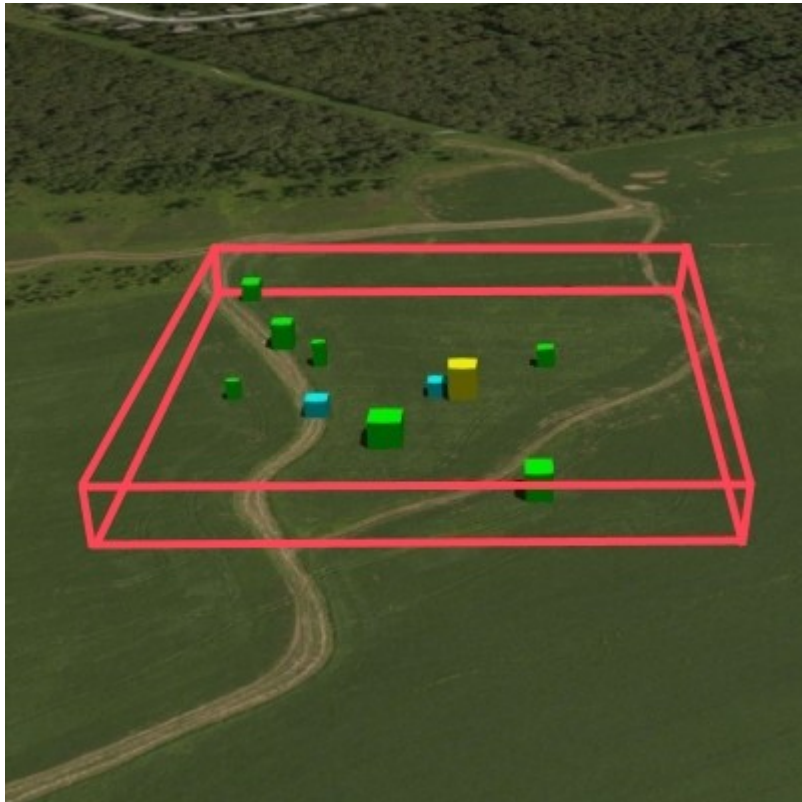
Figure 10: A bounding box

图 10:一个边界框

"boundingVolume": {

" boundingVolume": {

"box": [

【盒子】:[

0, 0, 10,

0，0，10，

100, 0, 0,

100，0，0，

0, 100, 0,

0，100，0，

0, 0, 10

0，0，10

]

]

}

}

Sphere

范围

The boundingVolume.sphere property is an array of four numbers that define a bounding sphere.The first three elements define the x, y, and z values for the center of the sphere in a right-handed 3-axis (x, y, z) Cartesian coordinate system where the z-axis is up.The last element (with index 3) defines the radius in meters.

boundingVolume.sphere 属性是定义边界球体的四个数字的数组。前三个元素定义了 z 轴向上的右手 3 轴(x，y，z)笛卡尔坐标系中球体中心的 x，y 和 z 值。最后一个元素(索引为 3)以米为单位定义半径。
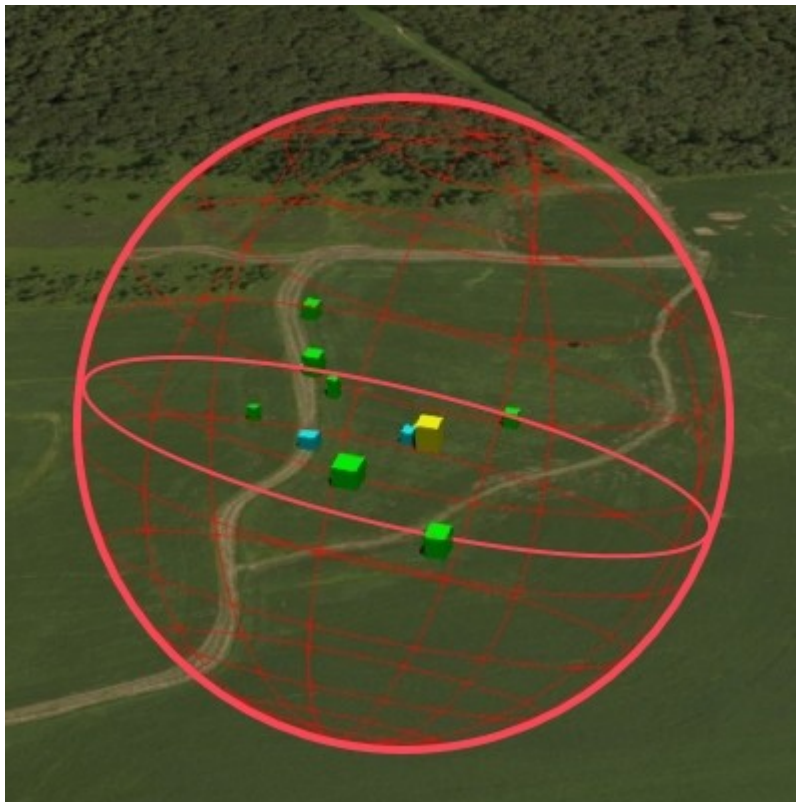
Figure 11: A bounding sphere

图 11:一个边界球体

"boundingVolume": {

" boundingVolume": {

"sphere": [

"球体":[

0,

0，

0,

0，

10,

10、

141.4214

141.4214

]

]

}

}

Viewer request volume

查看器请求卷

A tile's viewerRequestVolume can be used for combining heterogeneous datasets, and can be combined with External tilesets.

切片的 viewerRequestVolume 可用于组合异构数据集，并可与外部切片组合。

The following example has a building in a b3dm tile and a point cloud inside the building in a pnts tile.The point cloud tile's boundingVolume is a sphere with a radius of 1.25.It also has a larger sphere with a radius of 15 for the viewerRequestVolume.Since the geometricError is zero, the point cloud tile's content is always rendered (and initially requested) when the viewer is inside the large sphere defined by viewerRequestVolume.

以下示例在 b3dm 图块中有一个建筑，在 pnts 图块中有一个建筑内部的点云。点云图块的边界体积是半径为 1.25 的球体。对于 viewerRequestVolume，它还有一个半径为 15 的更大球体。由于几何误差为零，

当查看器位于由 viewerRequestVolume 定义的大球体内时，点云切片的内容总是被渲染(并且最初被请求)。

{

{

"children": [{

【儿童】:[

"transform": [

"转型":[

4.843178171884396, 1.2424271388626869, 0, 0,

4.843178171884396，1.2424271388626869，0，0，

-0.7993325488216595, 3.1159251367235608, 3.8278032889280675, 0,

-0.7993325488216595，3.1159251367235608，3.8278032889280675，0，

0.9511533376784163, -3.7077466670407433, 3.2168186118075526, 0,

0.951153376784163，-3.707746670407433，3.2168186118075526，0，

1215001.7612985559, -4736269.697480114, 4081650.708604793, 1

1215001.7612985559，-4736269.697480114，4081650.767678696

],

],

"boundingVolume": {

" boundingVolume": {

"box": [

【盒子】:[

0, 0, 6.701,

0，0，6.701，

3.738, 0, 0,

3.738，0，0，

0, 3.72, 0,

0，3.72，0，

0, 0, 13.402

0，0，13.402

]

]

},

},

"geometricError": 32,

"几何误差":32，

"content": {

"内容":{

"uri": "building.b3dm"

" uri": "building.b3dm "

}

}

}, {

}、{

"transform": [

"转型":[

0.968635634376879, 0.24848542777253732, 0, 0,

0.968635634376879，0.24848542777253732，0，0，

-0.15986650990768783, 0.6231850279035362, 0.7655606573007809, 0,

-0.15986650990768783，0.6231850279035362，0.7655606573007809，0，

0.19023066741520941, -0.7415493329385225, 0.6433637229384295, 0,

0.19023066741520941，-0.7415493329385225，0.64363729384295，0，

1215002.0371330238, -4736270.772726648, 4081651.6414821907, 1

1215002.0371330238，-4736270.772726648，4081651，1

],

],

"viewerRequestVolume": {

" viewerRequestVolume": {

"sphere": [0, 0, 0, 15]

【球体】:[0，0，0，15】

},

},

"boundingVolume": {

" boundingVolume": {

"sphere": [0, 0, 0, 1.25]

【球体】:[0，0，0，1.25】

},

},

"geometricError": 0,

"几何误差":0，

"content": {

"内容":{

"uri": "points.pnts"

" uri": "points.pnts "

}

}

}]

}]

}

}

For more on request volumes, see the sample tileset and demo video.

有关请求卷的更多信息，请参见示例 tileset 和演示视频。

Transforms

转换

Tile transforms

平铺变换

To support local coordinate systems—e.g., so a building tileset inside a city tileset can be defined in its own coordinate system, and a point cloud tileset inside the building could, again, be defined in its own coordinate system—each tile has an optional transform property.

为了支持局部坐标系(例如，城市瓦片区内的建筑瓦片区可以在其自己的坐标系中定义，并且建筑内的点云瓦片区也可以在其自己的坐标系中定义)，每个瓦片都具有可选的变换属性。

The transform property is a 4x4 affine transformation matrix, stored in column-major order, that transforms from the tile's local coordinate system to the parent tile's coordinate system—or the tileset's coordinate system in the case of the root tile.

transform 属性是一个 4x4 仿射变换矩阵，按列主顺序存储，它从图块的局部坐标系变换到父图块的坐标系，或者在根图块的情况下是图块的坐标系。

The transform property applies to:

转换属性适用于:

tile.content

平铺内容

Each feature's position.

每个特征的位置。

Each feature's normal should be transformed by the top-left 3x3 matrix of the inverse-transpose of transform to account for correct vector transforms when scale is used.

每个特征的法线应该由变换逆转置的左上角 3×3 矩阵进行变换，以便在使用比例时考虑正确的矢量变换。

content.boundingVolume, except when content.boundingVolume.region is defined, which is explicitly in EPSG:4979 coordinates.

content.boundingVolume，除非定义了 content.boundingVolume.region，该区域明确位于 EPSG:4979 坐标中。

tile.boundingVolume, except when tile.boundingVolume.region is defined, which is explicitly in EPSG:4979 coordinates.

除非定义了图块。边界体积。区域，该区域明确位于 EPSG:4979 坐标中。

tile.viewerRequestVolume, except when tile.viewerRequestVolume.region is defined, which is explicitly in EPSG:4979 coordinates.

除非在 EPSG:4979 坐标中明确定义了区域。

The transform property does not apply to geometricError—i.e., the scale defined by transform does not scale the geometric error—the geometric error is always defined in meters.

变换属性不适用于几何误差(geometricError)，即由变换定义的比例不会缩放几何误差，几何误差总是以米为单位定义。

When transform is not defined, it defaults to the identity matrix:

未定义转换时，它默认为身份矩阵:

[

[

1.0, 0.0, 0.0, 0.0,

1.0，0.0，0.0，0.0，

0.0, 1.0, 0.0, 0.0,

0.0，1.0，0.0，0.0，

0.0, 0.0, 1.0, 0.0,

0.0，0.0，1.0，0.0，

0.0, 0.0, 0.0, 1.0

0.0，0.0，0.0，1.0

]

]

The transformation from each tile's local coordinate system to the tileset's global coordinate system is computed by a top-down traversal of the tileset and by post-multiplying a child's transform with its parent's transform like a traditional scene graph or node hierarchy in computer graphics.

从每个图块的局部坐标系到图块的全局坐标系的转换是通过图块的自上而下遍历以及像传统场景图或计算机图形中的节点层次结构一样将子图块的转换与其父图块的转换后相乘来计算的。

glTF transforms

glTF 转换

Batched 3D Model and Instanced 3D Model tiles embed glTF, which defines its own node hierarchy and uses a y-up coordinate system.Any transforms specific to a tile format and the tile.transform property are applied after these transforms are resolved.

批量 3D 模型和实例化 3D 模型切片嵌入 glTF，GLtf 定义了自己的节点层次结构，并使用 y 向上坐标系。任何特定于平铺格式和平铺. transform 属性的转换都将在这些转换被解析后应用。

glTF node hierarchy

glTF 节点层次结构

First, glTF node hierarchy transforms are applied according to the glTF specification.

首先，根据 glTF 规范应用 glTF 节点层次结构转换。

y-up to z-up

y-up 至 z-up

Next, for consistency with the z-up coordinate system of 3D Tiles, glTFs must be transformed from y-up to z-up at runtime.This is done by rotating the model about the x-axis by π/2 radians.Equivalently, apply the following matrix transform (shown here as row-major):

接下来，为了与 3D 图块的 z 向上坐标系保持一致，glTFs 必须在运行时从 y 向上转换为 z 向上。这是通过将模型绕 x 轴旋转 π/2 弧度来实现的。等效地，应用以下矩阵变换(此处显示为行主变换):

[

[

1.0, 0.0, 0.0, 0.0,

1.0，0.0，0.0，0.0，

0.0, 0.0, -1.0, 0.0,

0.0，0.0，-1.0，0.0，

0.0, 1.0, 0.0, 0.0,

0.0，1.0，0.0，0.0，

0.0, 0.0, 0.0, 1.0

0.0，0.0，0.0，1.0

]

]

More broadly the order of transformations is:

更广泛地说，转换的顺序是:

glTF node hierarchy transformations

glTF 节点层次结构转换

glTF y-up to z-up transform

glTF y-up 至 z-up 转换

Any tile format specific transforms.

任何图块格式特定的转换。

Batched 3D Model Feature Table may define RTC_CENTER which is used to translate model vertices.

批量 3D 模型特征表可以定义 RTC _ CENTER，用于转换模型顶点。

Instanced 3D Model Feature Table defines per-instance position, normals, and scales.These are used to create per-instance 4x4 affine transform matrices that are applied to each instance.

实例化三维模型要素表定义了每个实例的位置、法线和比例。这些用于创建应用于每个实例的每个实例 4x4 仿射变换矩阵。

Tile transform

平铺变换

Implementation note: when working with source data that is inherently z-up, such as data in WGS 84 coordinates or in a local z-up coordinate system, a common workflow is:

实施说明:当使用固有向上的源数据时，例如 WGS 84 坐标或本地向上坐标系统中的数据，常见的工作流程是:

Mesh data, including positions and normals, are not modified - they remain z-up.

网格数据，包括位置和法线，不会被修改-它们保持 z 向上。

The root node matrix specifies a column-major z-up to y-up transform.This transforms the source data into a y-up coordinate system as required by glTF.

根节点矩阵指定一个列主 z 向上到 y 向上的变换。这将源数据转换为 glTF 要求的 y 向上坐标系。

At runtime the glTF is transformed back from y-up to z-up with the matrix above.Effectively the transforms cancel out.

运行时，glTF 通过上面的矩阵从 y 向上转换回 z 向上。这些转换实际上抵消了。

Example glTF root node:

glTF 根节点示例:

"nodes": [

"节点":[

{

{

"matrix": [1,0,0,0,0,0,-1,0,0,1,0,0,0,0,0,1],

"矩阵":[1，0，0，0，0，-1，0，0，1，0，0，0，0，0，1，

"mesh": 0,

【网格】:0、

"name": "rootNode"

"名称":" rootNode "

}

}

]

]

Example

例子

For an example of the computed transforms (transformToRoot in the code above) for a tileset, consider:

对于颚化符集的计算转换示例(上面代码中的 transformToRoot)，请考虑:
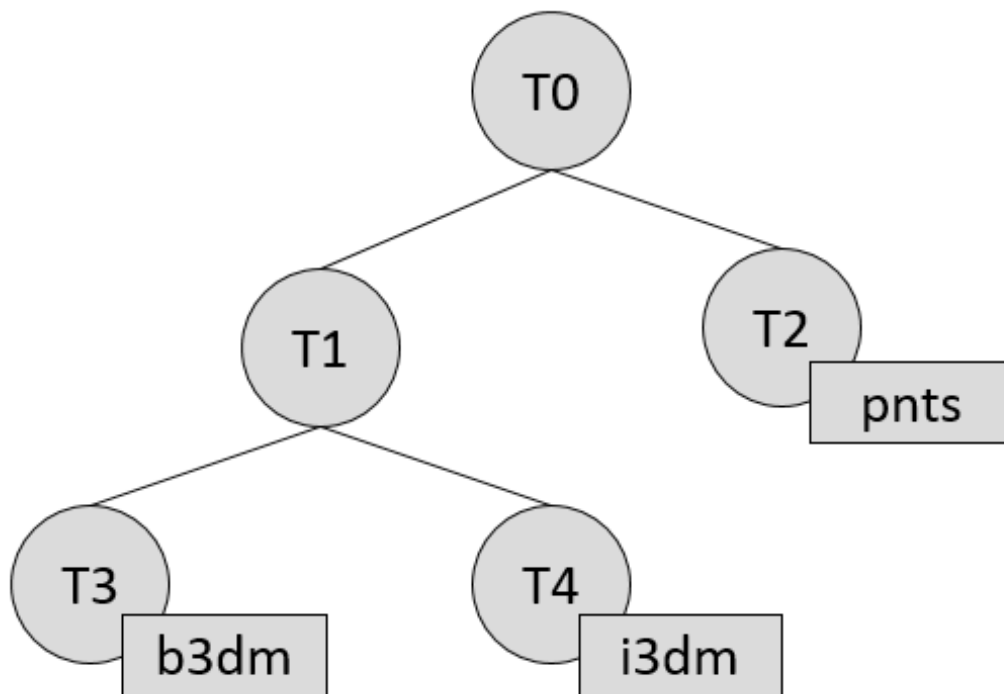
Figure 12: A tileset with transformed children tiles

图 12:一个带有转换子瓦片的瓦片集

The computed transform for each tile is:

每个图块的计算转换为:

TO: [T0]

致:[T0]

T1: [T0][T1]

T1: [T0][T1]

T2: [T0][T2]

T2:[T0][·T2]

T3: [T0][T1][T3]

T3: [T0][T1][T3]

T4: [T0][T1][T4]

T4: [T0][T1][T4]

The positions and normals in a tile's content may also have tile-specific transformations applied to them before the tile's transform (before indicates post-multiplying for affine transformations).Some examples are:

图块内容中的位置和法线也可以在图块变换之前应用图块特定的变换(之前表示仿射变换的后乘法)。一些例子是:

b3dm and i3dm tiles embed glTF, which defines its own node hierarchy and coordinate system.tile.transform is applied after these transforms are resolved.See glTF transforms.

b3dm 和 i3dm 图块嵌入 glTF，GLtf 定义了自己的节点层次结构和坐标系。在这些转换被解析后，应用 tile.transform。请参见 glTF 转换。

i3dm's Feature Table defines per-instance position, normals, and scales.These are used to create per-instance 4x4 affine transform matrices that are applied to each instance before tile.transform.

i3dm 的要素表定义了每个实例的位置、法线和比例。这些用于创建每个实例 4x4 仿射变换矩阵，这些矩阵在平铺变换之前应用于每个实例

Compressed attributes, such as POSITION_QUANTIZED in the Feature Tables for i3dm and pnts, and NORMAL_OCT16P in pnts should be decompressed before any other transforms.

压缩属性，如 i3dm 和 pnts 的特征表中的 POSITION_QUANTIZED 和 pnts 中的 NORMAL_OCT16P，应在任何其他转换之前解压缩。

Therefore, the full computed transforms for the above example are:

因此，以上示例的完整计算转换是:

TO: [T0]

致:[T0]

T1: [T0][T1]

T1: [T0][T1]

T2: [T0][T2][pnts-specific transform, including RTC_CENTER (if defined)]

T2: [T0][T2][pnts 特定转换，包括 RTC_CENTER(如果定义)]

T3: [T0][T1][T3][b3dm-specific transform, including RTC_CENTER (if defined), coordinate system transform, and glTF node hierarchy]

T3: [T0][T1][T3][b3dm 特定转换，包括 RTC_CENTER(如果定义)、坐标系转换和 glTF 节点层次结构]

T4: [T0][T1][T4][i3dm-specific transform, including per-instance transform, coordinate system transform, and glTF node hierarchy]

T4: [T0][T1][T4][i3dm 特定转换，包括每个实例转换、坐标系转换和 glTF 节点层次结构]

Implementation example

实现示例

This section is non-normative

本节是非规范性的

The following JavaScript code shows how to compute this using Cesium's Matrix4 and Matrix3 types.

下面的 JavaScript 代码展示了如何使用铯的矩阵 4 和矩阵 3 类型来计算它。

```
function computeTransforms(tileset) {
```

函数计算转换(tileset) {

```
var t = tileset.root;
```

```
var t = tileset.root
```

```
var transformToRoot = defined(t.transform) ?Matrix4.fromArray(t.transform) : Matrix4.IDENTITY;
```

var transformToRoot =已定义(t.transform)? 矩阵 4.fromArray(t.transform):矩阵 4。身份；

```
computeTransform(t, transformToRoot);
```

计算转换(t，transform ToRot)；

```
}
```

```
}
```

```
function computeTransform(tile, transformToRoot) {
```

函数计算转换(平铺，转换操作){

```
// Apply 4x4 transformToRoot to this tile's positions and bounding volumes
```

//将 4x4 转换对象应用于此图块的位置和边界体积

```
var inverseTransform = Matrix4.inverse(transformToRoot, new Matrix4());
```

var inverseTransform =矩阵 4 . inverse(transform torot，新矩阵 4())；

```
var normalTransform = Matrix4.getRotation(inverseTransform, new Matrix3());
```

var 法线变换=矩阵 4.getRotation(反转变换，新矩阵 3())；

```
normalTransform = Matrix3.transpose(normalTransform, normalTransform);
```

常态变换=矩阵 3 .转置(常态变换，常态变换)；

```
// Apply 3x3 normalTransform to this tile's normals
```

//将3×3法线变换应用于此图块的法线

var children = tile.children;

var children = tile.children

var length = children.length;

可变长度=儿童长度；

for (var i = 0;i < length;++i) {

(var I = 0；I <长度；++i) {

var child = children[i];

var child = children[I]；

var childToRoot = defined(child.transform) ?Matrix4.fromArray(child.transform) : Matrix4.clone(Matrix4.IDENTITY);

var childToRoot =已定义(子转换)？矩阵4 .来自数组(子变换) :矩阵4 .克隆(矩阵4)。身份);

childToRoot = Matrix4.multiplyTransformation(transformToRoot, childToRoot, childToRoot);

childToRoot =矩阵4 .多平台转换(transformToRoot、childToRoot、childToRoot);

computeTransform(child, childToRoot);

计算转换(子，子对象);

}

}

}

}

<mark>Tile JSON</mark>

平铺JSON

A tile JSON object consists of the following properties.

平铺JSON对象由以下属性组成。

tile

boundingVolume

box region sphere

geometricError

refine

content

– boundingVolume (box, region, or sphere)

– uri - - - - - - - - -▶ Separate file with tile contents, streamed on demand

children[ ]
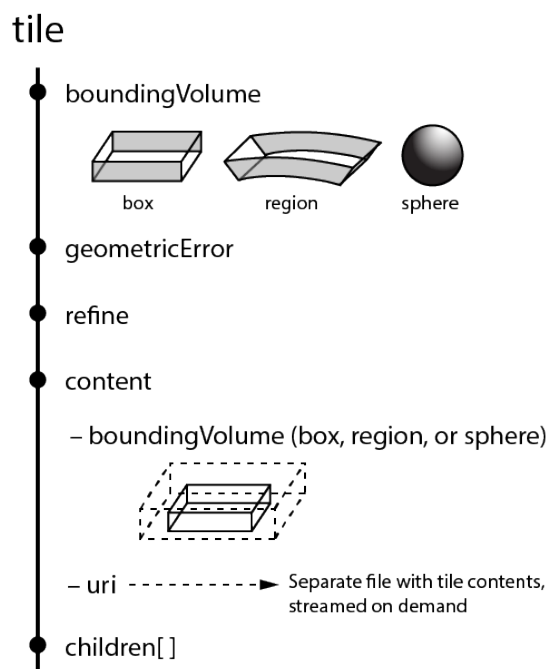
Figure 13: Tile JSON properties

图 13:平铺 JSON 属性

The following example shows one non-leaf tile.

以下示例显示了一个非叶图块。

{

{

"boundingVolume": {

" boundingVolume": {

"region": [

【地区】:[

-1.2419052957251926,

-1.2419052957251926，

0.7395016240301894,

0.7395016240301894，

-1.2415404171917719,

-1.2415404171917719，

0.7396563300150859,

0.7396563300150859，

0,

0，

20.4

20.4

]

]

},

},

"geometricError": 43.88464075650763,

"几何误差":43.88464075650763，

"refine" : "ADD",

"提炼":"添加"，

"content": {

"内容":{

"boundingVolume": {

" boundingVolume": {

"region": [

【地区】:[

-1.2418882438584018,

-1.2418882438584018，

0.7395016240301894,

0.7395016240301894，

-1.2415422846940714,

-1.2415422846940714，

0.7396461198389616,

0.7396461198389616，

0,

0，

19.4

19.4

]

]

},

},

"uri": "2/0/0.b3dm"

" uri": "2/0/0.b3dm "

},

},

"children": [...]

"儿童" :[...]

}

}

The boundingVolume defines a volume enclosing the tile, and is used to determine which tiles to render at runtime.The above example uses a region volume, but other bounding volumes, such as box or sphere, may be used.

边界体积(boundingVolume)定义了包围图块的体积，并用于确定在运行时渲染哪些图块。上面的例子使用了一个区域体积，但是也可以使用其他边界体积，例如盒子或球体。

The geometricError property is a nonnegative number that defines the error, in meters, introduced if this tile is rendered and its children are not.At runtime, the geometric error is used to compute Screen-Space Error (SSE), the error measured in pixels.The SSE determines if a tile is sufficiently detailed for the current view or if its children should be considered, see

geometricError 属性是一个非负数，它定义了如果渲染此图块而其子图块未渲染时引入的误差(以米为单位)。运行时，几何误差用于计算屏幕空间误差(SSE)，即以像素为单位测量的误差。上交所确定图块对于当前视图是否足够详细，或者是否应考虑其子对象，请参见

Tiles consist of metadata used to determine if a tile is rendered, a reference to the renderable content, and an array of any children tiles..

切片由用于确定是否呈现切片的元数据、对可呈现内容的引用以及任何子切片的数组组成..

The optional viewerRequestVolume property (not shown above) defines a volume, using the same schema as boundingVolume, which the viewer must be inside of before the tile's content will be requested and before the tile will be refined based on geometricError.See the Viewer request volume section.

可选的 viewerRequestVolume 属性(上面未显示)使用与 boundingVolume 相同的模式定义了一个卷，在请求切片的内容之前，以及在基于几何误差细化切片之前，查看器必须位于该卷内。请参见查看器请求卷部分。

The refine property is a string that is either "REPLACE" for replacement refinement or "ADD" for additive refinement, see

refine 属性是一个字符串，替换精化为"REPLACE"，添加精化为"ADD"，请参见

Tiles are structured into a tree incorporating Hierarchical Level of Detail (HLOD) so that at runtime a client implementation will need to determine if a tile is sufficiently detailed for rendering and if the content of tiles should be successively refined by children tiles of higher resolution.An implementation will consider a maximum allowed Screen-Space Error (SSE), the error measured in pixels..It is required for the root tile of a tileset;it is optional for all other tiles.A tileset can use any combination of additive and replacement refinement.When the refine property is omitted, it is inherited from the parent tile.

图块被结构化为包含层次细节级别(HLOD)的树，以便在运行时客户端实现将需要确定图块是否足够详细以用于渲染，以及图块的内容是否应该由更高分辨率的子图块连续细化。实现将考虑最大允许屏幕空间误差(SSE)，即以像素为单位测量的误差..它是 tileset 的根切片所必需的；它对于所有其他图块都是可选的。tileset 可以使用加法和替换细化的任意组合。省略 refine 属性时，它将从父切片继承。

The content property is an object that contains metadata about the tile's renderable content.content.uri is a uri that points to the tile's binary content (see Tile format specifications), or another tileset JSON to create a tileset of tileset (see External tilesets).

content 属性是一个对象，它包含关于切片的可呈现内容的元数据。content.uri 是指向平铺的二进制内容(请参见平铺格式规范)或另一个平铺 JSON 的 uri，以创建平铺集的平铺集(请参见外部平铺集)。

A file extension is not required for content.uri. A content's tile format (see Tile format specifications) can be identified by the magic field in its header, or else as an external tileset if the content is JSON.

content.uri 不需要文件扩展名。内容的平铺格式(请参见平铺格式规范)可以通过其标题中的魔法字段来标识，或者如果内容是 JSON，则可以作为外部平铺集来标识。

The content.boundingVolume property defines an optional bounding volume similar to the top-level boundingVolume property.But unlike the top-level boundingVolume property, content.boundingVolume is a tightly fit bounding volume enclosing just the tile's content.boundingVolume provides spatial coherence and content.boundingVolume enables tight view frustum culling, excluding from rendering any content not in the volume of what is potentially in view.When it is not defined, the tile's bounding volume is still used for culling (see Grids).

content.boundingVolume 属性定义了一个类似于顶级 boundingVolume 属性的可选边界卷。但是与顶级的边界卷属性不同，content.boundingVolume 是一个紧密匹配的边界卷，它只包含图块的内容。边界体积(boundingVolume)提供空间一致性和内容，边界体积(boundingVolume)支持紧密视图截头体剔除，排除不在潜在视图体积内的任何内容。未定义时，图块的边界体积仍用于剔除(请参见栅格)。

The screenshot below shows the bounding volumes for the root tile for Canary Wharf.boundingVolume, shown in red, encloses the entire area of the tileset;content.boundingVolume shown in blue, encloses just the four features (models) in the root tile.

下图显示了金丝雀码头根瓦片的边界体积。边界体积，以红色显示，包围了波浪线的整个区域；content.boundingVolume 以蓝色显示，仅包含根切片中的四个特征(模型)。
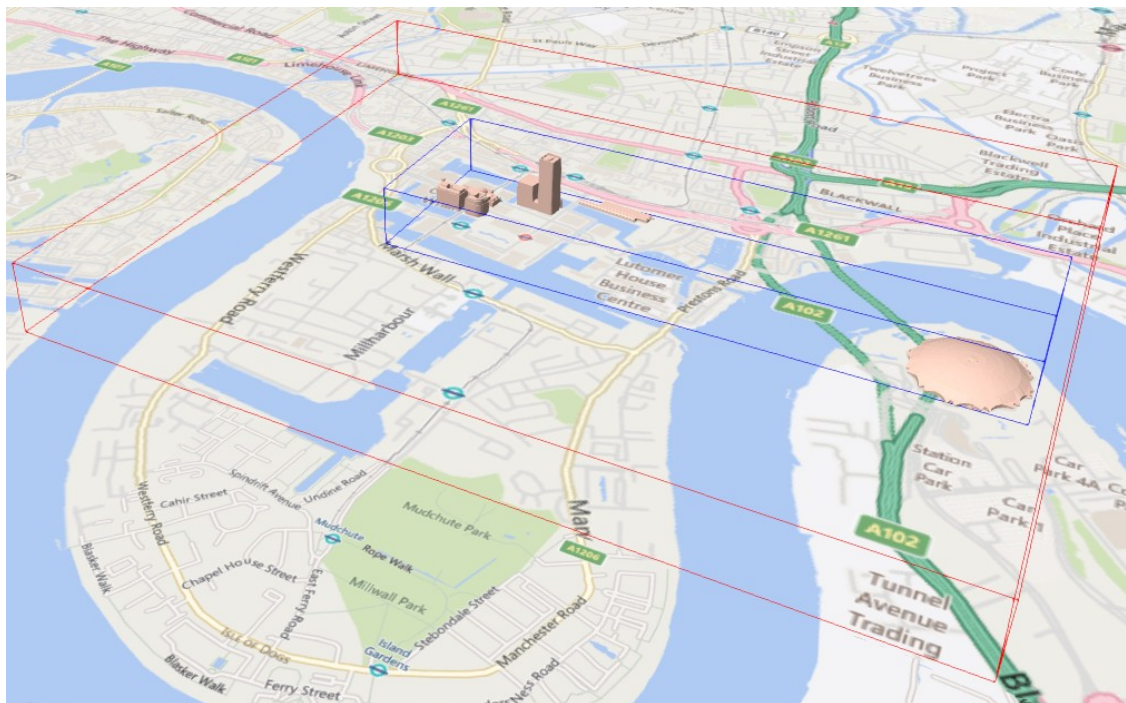
Figure 14: A tile bounding volume in red, and a content bounding volume in blue

图 14:红色的图块边界体积和蓝色的内容边界体积

The optional transform property (not shown above) defines a 4x4 affine transformation matrix that transforms the tile's content, boundingVolume, and viewerRequestVolume as described in the T section.

可选的转换属性(上面没有显示)定义了一个 4x4 仿射转换矩阵，该矩阵可以转换图块的内容、边界体积和视图请求体积，如测试部分所述。

The children property is an array of objects that define child tiles.Each child tile's content is fully enclosed by its parent tile's boundingVolume and, generally, a geometricError less than its parent tile's geometricError.For leaf tiles, the length of this array is zero, and children may not be defined.See the Tileset JSON section below.

children 属性是定义子切片的对象数组。每个子图块的内容都被其父图块的边界体积完全包围，并且通常几何误差小于其父图块的几何误差。对于叶图块，此数组的长度为零，并且可能没有定义子项。参见下面的 Tileset JSON 部分。

See Property reference for the tile JSON schema reference.

有关平铺 JSON 模式引用，请参见属性引用。

Tileset JSON

Tileset JSON

3D Tiles uses one main tileset JSON file as the entry point to define a tileset.Both entry and external tileset JSON files are not required to follow a specific naming convention.

3D 平铺使用一个主平铺 JSON 文件作为入口点来定义平铺。条目和外部 tileset JSON 文件都不需要遵循特定的命名约定。

Here is a subset of the tileset JSON used for Canary Wharf (also see the complete file, tileset.json):

这里是金丝雀码头使用的 tileset JSON 的子集(另请参见完整文件 tileset.json):

{

{

"asset" : {

"资产":{

"version": "1.0",

"版本":"1.0",

"tilesetVersion": "e575c6f1-a45b-420a-b172-6449fa6e0a59",

" Tilesetversion ":" e 575 c 6 f1-a45 b-420 a-b 172-6449 F6 e0a 59 ",

},

},

"properties": {

"属性":{

"Height": {

"高度":{

"minimum": 1,

【最小值】:1、

"maximum": 241.6

【最大】:241.6

}

}

},

},

"geometricError": 494.50961650991815,

"几何误差":49996 . 69998988996

"root": {

"根":{

"boundingVolume": {

" boundingVolume": {

"region": [

【地区】:[

-0.0005682966577418737,

-0.0005682966577418737，

0.8987233516605286,

0.8987233516605286，

0.00011646582098558159,

0.00011646582098558159，

0.8990603398325034,

0.8990603398325034，

0,

0，

241.6

241.6

]

]

},

},

"geometricError": 268.37878244706053,

"几何误差":26888 . 488888686056

"refine": "ADD",

"提炼":"添加",

"content": {

"内容":{

"uri": "0/0/0.b3dm",

" uri": "0/0/0.b3dm ",

"boundingVolume": {

" boundingVolume": {

"region": [

【地区】:[

-0.0004001690908972599,

-0.0004001690908972599，

0.8988700116775743,

0.8988700116775743，

0.00010096729722787196,

0.00010096729722787196，

0.8989625664878067,

0.8989625664878067，

0,

```
        0,

        241.6

        241.6

        ]

    ]

}

}

},

},

"children": [..]

"儿童":[..]

}

}

}

}
```

The tileset JSON has four top-level properties: asset, properties, geometricError, and root.

tileset JSON 有四个顶级属性:资产、属性、几何误差和根。

asset is an object containing metadata about the entire tileset.The asset.version property is a string that defines the 3D Tiles version, which specifies the JSON schema for the tileset and the base set of tile formats.The tilesetVersion property is an optional string that defines an application-specific version of a tileset, e.g., for when an existing tileset is updated.

asset 是一个对象，包含关于整个 tileset 的元数据。asset.version 属性是一个定义 3D 切片版本的字符串，它为切片集和切片格式的基本集指定 JSON 模式。tilesetVersion 属性是一个可选字符串，它定义了 tileset 的应用程序特定版本，例如，在更新现有 tileset 时。

properties is an object containing objects for each per-feature property in the tileset.This tileset JSON snippet is for 3D buildings, so each tile has building models, and each building model has a Height property (see Batch Table).The name of each object in properties matches the name of a per-feature property, and its value defines its minimum and maximum numeric values, which are useful, for example, for creating color ramps for styling.

properties 是一个对象，包含 tileset 中每个每个要素属性的对象。这个瓦片组 JSON 片段是用于 3D 建筑的，所以每个瓦片都有建筑模型，每个建筑模型都有一个高度属性(参见批处理表)。属性中每个对象的

名称都与每个要素属性的名称相匹配，其值定义了其最小值和最大值，这些值对于创建样式的颜色渐变非常有用。

geometricError is a nonnegative number that defines the error, in meters, when the tileset is not rendered.See

几何误差(geometricError)是一个非负数，它定义了不呈现颚化符时的误差，单位为米。看见

Tiles consist of metadata used to determine if a tile is rendered, a reference to the renderable content, and an array of any children tiles.for how this value is used to drive refinement.

切片由用于确定是否呈现切片的元数据、对可呈现内容的引用以及任何子切片的数组组成。了解如何使用该值来驱动细化。

root is an object that defines the root tile using the JSON described in the above section.root.geometricError is not the same as the tileset's top-level geometricError.The tileset's geometricError is the error when the entire tileset is not rendered;root.geometricError is the error when only the root tile is rendered.

root 是一个对象，它使用上一节中描述的 JSON 定义根切片。root.geometricError 与 tileset 的顶级 geometricError 不同。tileset 的 geometricError 是未呈现整个 tileset 时的错误；root.geometricError 是仅呈现根切片时的错误。

root.children is an array of objects that define child tiles.Each child tile's content is fully enclosed by its parent tile's boundingVolume and, generally, a geometricError less than its parent tile's geometricError.For leaf tiles, the length of this array is zero, and children may not be defined.

root.children 是定义子切片的对象数组。每个子图块的内容都被其父图块的边界体积完全包围，并且通常几何误差小于其父图块的几何误差。对于叶图块，此数组的长度为零，并且可能没有定义子项。

External tilesets

外部颚化符

To create a tree of trees, a tile's content.uri can point to an external tileset (the uri of another tileset JSON file).This enables, for example, storing each city in a tileset and then having a global tileset of tilesets.

要创建树树，切片的 content.uri 可以指向外部切片集(另一个切片集 JSON 文件的 uri)。例如，这使得能够将每个城市存储在一个颚化集合中，然后具有颚化集合的全局颚化集合。
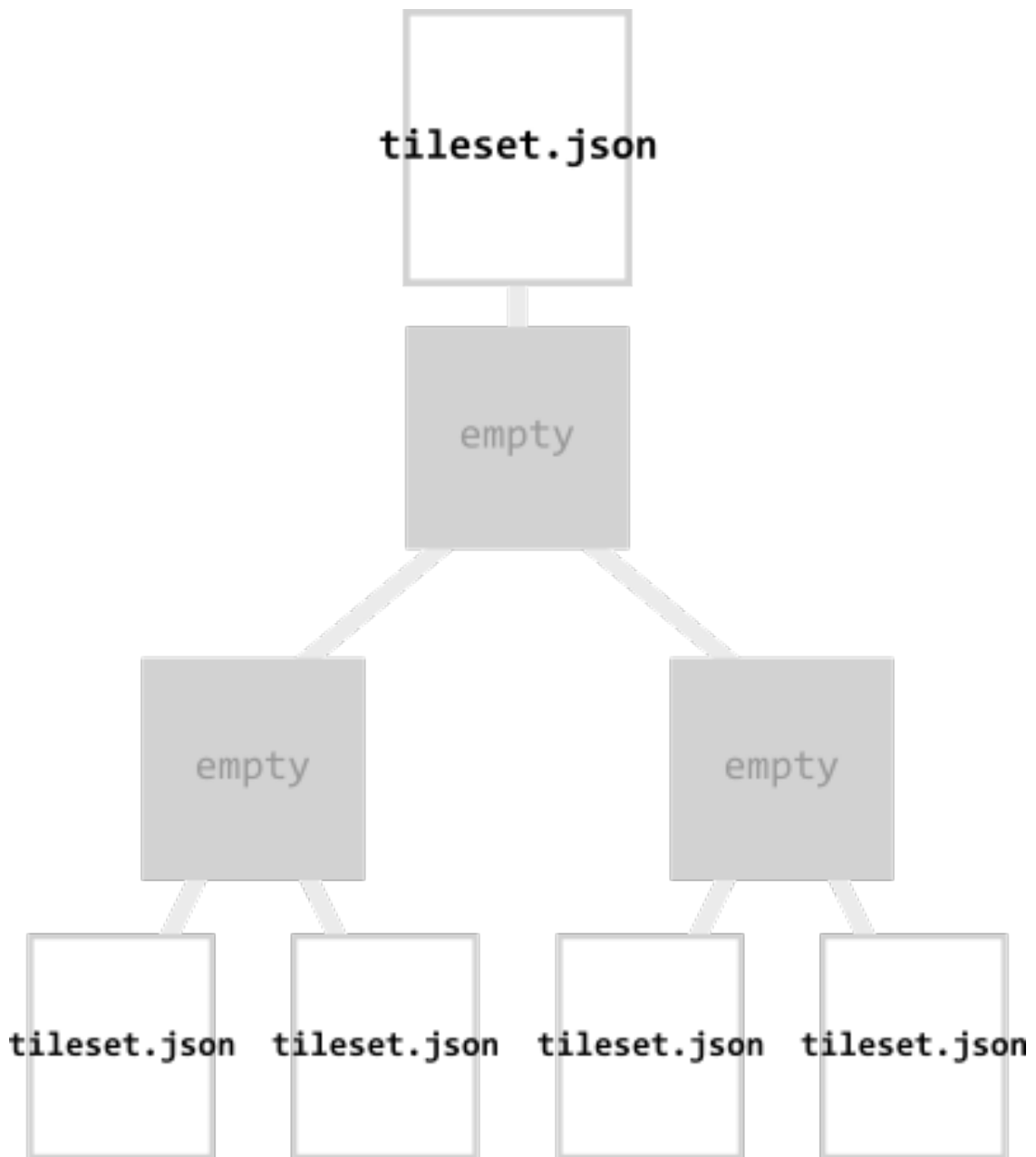
Figure 15: A tileset JSON file with external tileset JSON files

图 15:一个带有外部 tileset JSON 文件的 tileset JSON 文件

When a tile points to an external tileset, the tile:

当图块指向外部图块集时，图块:

Cannot have any children;tile.children must be undefined or an empty array.

不能有任何孩子；tile.children 必须未定义或为空数组。

Cannot be used to create cycles, for example, by pointing to the same tileset file containing the tile or by pointing to another tileset file that then points back to the initial file containing the tile.

不能用于创建循环，例如，通过指向包含平铺的同一个平铺文件或指向另一个平铺文件，然后指向包含平铺的初始文件。

Will be transformed by both the tile's transform and root tile's transform.For example, in the following tileset referencing an external tileset, the computed transform for T3 is [T0][T1][T2][T3].

将通过图块的变换和根图块的变换进行变换。例如，在下面引用外部颚化符集的颚化符集中，T3 的计算变换是[T0][T1][T2][T3]。
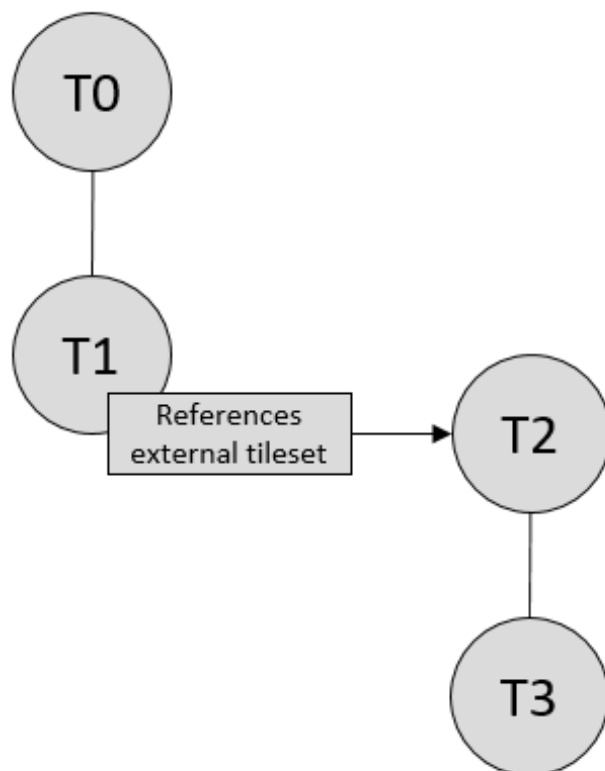
Figure 16: A tileset with transforms referencing an external tileset with transforms

图 16:带有转换的颚化符集引用带有转换的外部颚化符集

Bounding volume spatial coherence

包围体空间相干性

As described above, the tree has spatial coherence;each tile has a bounding volume completely enclosing its content, and the content for child tiles are completely inside the parent's bounding volume.This does not imply that a child's bounding volume is completely inside its parent's bounding volume.For example:

如上所述，树具有空间一致性；每个图块都有一个完全包围其内容的边界体积，子图块的内容完全位于父图块的边界体积内。这并不意味着子对象的边界体积完全位于其父对象的边界体积内。例如:

Spatial data structures

空间数据结构

3D Tiles incorporates the concept of Hierarchical Level of Detail (HLOD) for optimal rendering of spatial data.A tileset is composed of a tree, defined by root and, recursively, its children tiles, which can be organized by different types of spatial data structures.

3D 切片结合了层次细节层次(HLOD)的概念，用于空间数据的最佳渲染。tileset 由一棵树组成，树由根和递归地由其子切片定义，子切片可以由不同类型的空间数据结构组织。

A runtime engine is generic and will render any tree defined by a tileset.Any combination of tile formats and refinement approaches can be used, enabling flexibility in supporting heterogeneous datasets, see Refinement.

运行时引擎是通用的，它将呈现由 tileset 定义的任何树。可以使用切片格式和细化方法的任意组合，从而实现支持异构数据集的灵活性，请参见细化。

A tileset may use a 2D spatial tiling scheme similar to raster and vector tiling schemes (like a Web Map Tile Service (WMTS) or XYZ scheme) that serve predefined tiles at several levels of detail (or zoom levels).However since the content of a tileset is often non-uniform or may not easily be organized in only two dimensions, other spatial data structures may be more optimal.

瓦片区可以使用类似于光栅和矢量瓦片区方案(如网络地图瓦片服务(WMTS)或 XYZ 方案)的 2D 空间瓦片区方案，该方案在几个细节级别(或缩放级别)提供预定义的瓦片。然而，由于颚化符集的内容通常是不一致的，或者可能不容易仅在两个维度上组织，所以其他空间数据结构可能更为优化。

Included below is a brief description of how 3D Tiles can represent various spatial data structures.

以下是 3D 图块如何表示各种空间数据结构的简要描述。

Quadtrees

四叉树

A quadtree is created when each tile has four uniformly subdivided children (e.g., using the center latitude and longitude), similar to typical 2D geospatial tiling schemes.Empty child tiles can be omitted.

与典型的 2D 地理空间切片方案类似，当每个切片具有四个均匀细分的子切片(例如，使用中心纬度和经度)时，创建四叉树。可以省略空的子切片。

3D Tiles enable quadtree variations such as non-uniform subdivision and tight bounding volumes (as opposed to bounding, for example, the full 25% of the parent tile, which is wasteful for sparse datasets).

3D 切片支持四叉树变化，例如不均匀细分和紧密边界体积(与例如父切片的全部 25%的边界相反，这对于稀疏数据集是浪费的)。

For example, here is the root tile and its children for Canary Wharf.Note the bottom left, where the bounding volume does not include the water on the left where no buildings will appear:

例如，这里是金丝雀码头的根瓦片及其子瓦片。请注意左下角，其中边界体积不包括左边没有建筑物出现的水:
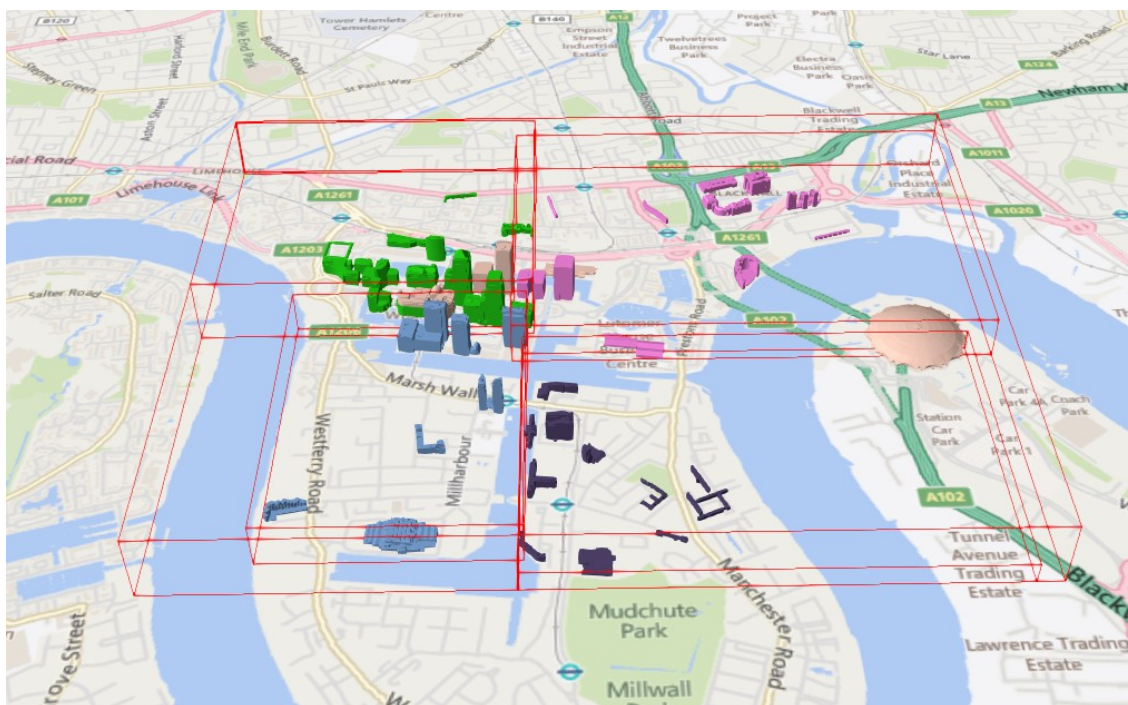
Figure 17: A root tile and its four children tiles

图 17:根瓦片及其四个子瓦片

3D Tiles also enable other quadtree variations such as loose quadtrees, where child tiles overlap but spatial coherence is still preserved, i.e., a parent tile completely encloses all of its children.This approach can be useful to avoid splitting features, such as 3D models, across tiles.

3D 拼贴还支持其他四叉树变体，如松散的四叉树，其中子拼贴重叠，但空间一致性仍然保留，即父拼贴完全包围其所有子拼贴。这种方法有助于避免将要素(如 3D 模型)分割到切片中。

Below, the green buildings are in the left child and the purple buildings are in the right child.Note that the tiles overlap so the two green and one purple building in the center are not split.

下面，绿色的建筑在左边的孩子，紫色的建筑在右边的孩子。请注意，瓷砖重叠，因此中间的两个绿色和一个紫色建筑不会分开。
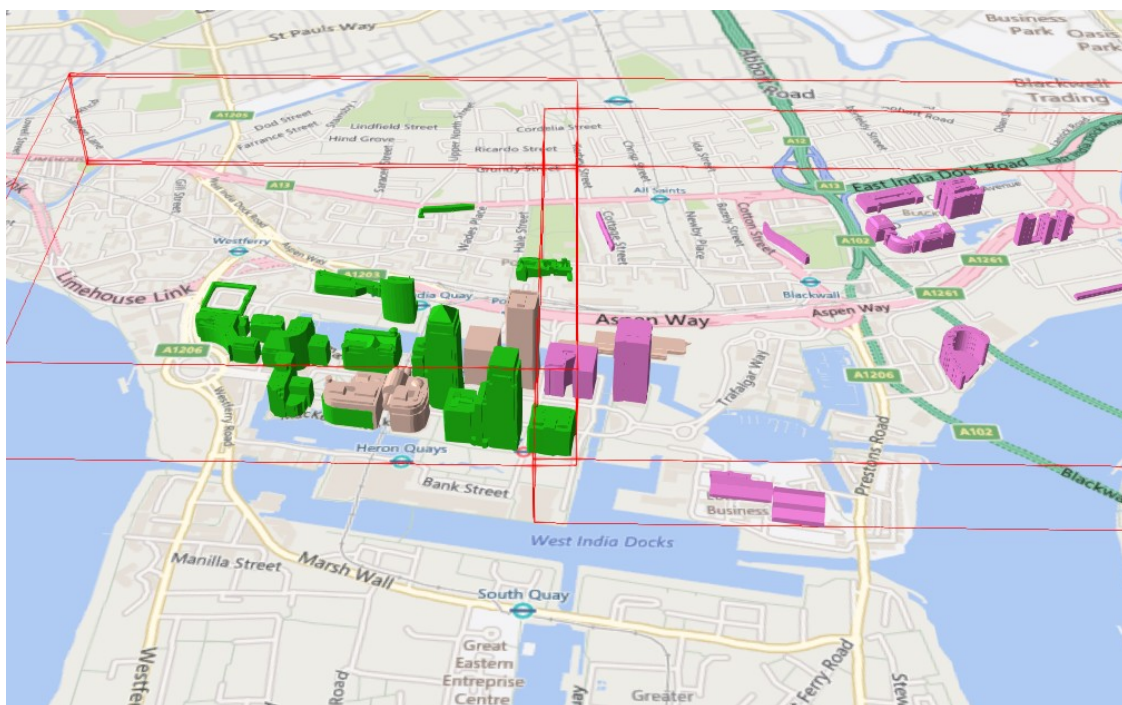
Figure 18: Two sibling tiles with overlapping bounding volumes

图 18:两个具有重叠边界体积的兄弟图块

K-d trees

K-d 树

A k-d tree is created when each tile has two children separated by a splitting plane parallel to the x, y, or z axis (or latitude, longitude, height).The split axis is often round-robin rotated as levels increase down the tree, and the splitting plane may be selected using the median split, surface area heuristics, or other approaches.

当每个图块有两个子图块被平行于 x、y 或 z 轴(或纬度、经度、高度)的分割平面分隔开时,将创建 k-d 树。分割轴通常随着水平沿树向下增加而循环旋转,并且可以使用中间分割、表面积试探法或其他方法来选择分割平面。

Note that a k-d tree does not have uniform subdivision like typical 2D geospatial tiling schemes and, therefore, can create a more balanced tree for sparse and non-uniformly distributed datasets.

请注意,k-d 树不像典型的 2D 地理空间切片方案那样具有统一的细分,因此可以为稀疏和非均匀分布的数据集创建更平衡的树。

3D Tiles enables variations on k-d trees such as multi-way k-d trees where, at each leaf of the tree, there are multiple splits along an axis.Instead of having two children per tile, there are n children.

3D 切片支持 k-d 树的变化,例如多向 k-d 树,其中在树的每片叶子上,沿着轴有多个分割。不是每块瓷砖有两个孩子,而是有 n 个孩子。

Octrees

八叉树

An octree extends a quadtree by using three orthogonal splitting planes to subdivide a tile into eight children.Like quadtrees, 3D Tiles allows variations to octrees such as non-uniform subdivision, tight bounding volumes, and overlapping children.

八叉树通过使用三个正交分割平面将一个图块细分为八个子图块来扩展四叉树。像四叉树一样，3D 切片允许八叉树的变化，例如不均匀细分、紧密边界体积和重叠子树。

Grids

网格

3D Tiles enables uniform, non-uniform, and overlapping grids by supporting an arbitrary number of child tiles.For example, here is a top-down view of a non-uniform overlapping grid of Cambridge:

3D 切片通过支持任意数量的子切片，实现了均匀、不均匀和重叠的网格。例如，以下是剑桥不均匀重叠网格的自上而下视图:
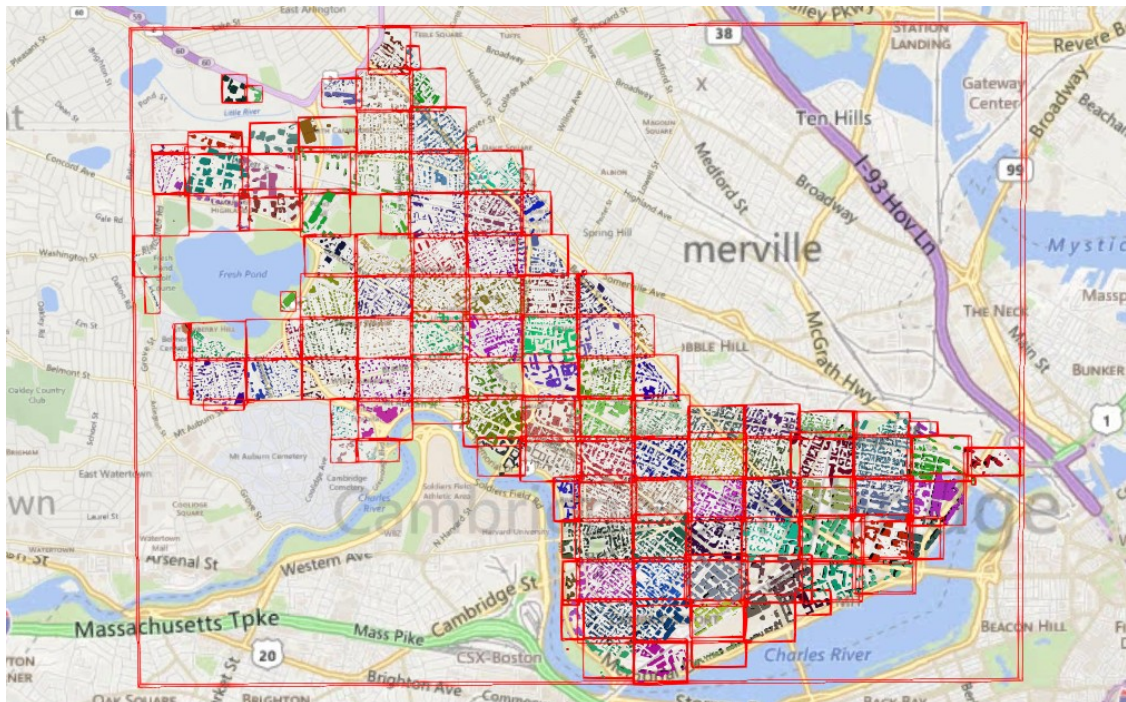
Figure 19: A tileset with an overlapping grid spatial data structure

图 19:具有重叠网格空间数据结构的瓦片区

3D Tiles takes advantage of empty tiles: those tiles that have a bounding volume, but no content.Since a tile's content property does not need to be defined, empty non-leaf tiles can be used to accelerate non-uniform grids with hierarchical culling.This essentially creates a quadtree or octree without hierarchical levels of detail (HLOD).

3D 拼贴利用了空拼贴:那些有边界体积但没有内容的拼贴。由于不需要定义切片的内容属性，空的非叶切片可用于通过分层剔除来加速非均匀网格。这实质上创建了一个没有层次细节层次的四叉树或八叉树。

Specifying extensions and application specific extras

指定扩展和特定于应用程序的附加功能

3D Tiles defines extensions to allow the base specification to have extensibility for new features, as well as extras to allow for application specific metadata.

3D 切片定义扩展，以允许基本规范具有新功能的可扩展性，以及允许应用程序特定元数据的附加功能。

Extensions

延长

Extensions allow the base specification to be extended with new features.The optional extensions dictionary property may be added to any 3D Tiles JSON object, which contains the name of the extensions and the extension specific objects.The following example shows a tile object with a hypothetical vendor extension which specifies a separate collision volume.

扩展允许用新特性扩展基本规范。可选的扩展字典属性可以添加到任何 3D 平铺 JSON 对象中，该对象包含扩展的名称和特定于扩展的对象。以下示例显示了一个带有假定供应商扩展的平铺对象，该扩展指定了一个单独的冲突卷。

{

{

"transform": [

"转型" :[

4.843178171884396, 1.2424271388626869, 0, 0,

4.843178171884396，1.2424271388626869，0，0，

-0.7993325488216595, 3.1159251367235608, 3.8278032889280675, 0,

-0.7993325488216595，3.1159251367235608，3.8278032889280675，0，

0.9511533376784163, -3.7077466670407433, 3.2168186118075526, 0,

0.951153376784163，-3.707746670407433，3.2168186118075526，0，

1215001.7612985559, -4736269.697480114, 4081650.708604793, 1

1215001.7612985559，-4736269.697480114，4081650.767678696

],

],

"boundingVolume": {

```
" boundingVolume": {

"box": [

【盒子】:[

0, 0, 6.701,

0，0，6.701，

3.738, 0, 0,

3.738，0，0，

0, 3.72, 0,

0，3.72，0，

0, 0, 13.402

0，0，13.402

]

]

},

},

"geometricError": 32,

"几何误差":32，

"content": {

"内容":{

"uri": "building.b3dm"

" uri": "building.b3dm "

},

},

"extensions": {

"扩展名":{

"VENDOR_collision_volume": {
```

" VENDOR _ conflict _ volume ":{

"box": [

【盒子】:[

0, 0, 6.8,

0，0，6.8，

3.8, 0, 0,

3.8，0，0，

0, 3.8, 0,

0，3.8，0，

0, 0, 13.5

0，0，13.5

]

]

}

}

}

}

}

}

All extensions used in a tileset or any descendant external tilesets must be listed in the entry tileset JSON in the top-level extensionsUsed array property, e.g.,

tileset 或任何后代外部 tileset 中使用的所有扩展必须在顶级扩展使用数组属性的条目 tileset JSON 中列出，例如，

{

{

"extensionsUsed": [

"扩展使用":[

"VENDOR_collision_volume"

"供应商_冲突_卷"

]

]

}

}

All extensions required to load and render a tileset or any descendant external tilesets must also be listed in the entry tileset JSON in the top-level extensionsRequired array property, such that extensionsRequired is a subset of extensionsUsed.All values in extensionsRequired must also exist in extensionsUsed.

加载和呈现颚化符集或任何后代外部颚化符集所需的所有扩展也必须在顶级 extensions required 数组属性的条目颚化符集 JSON 中列出，以便 Required extensions 是 sUsed 扩展的子集。扩展中需要的所有值也必须存在于扩展使用中。

Extras

额外费用

The extras property allows application specific metadata to be added to any 3D Tiles JSON object.The following example shows a tile object with an additional application specific name property.

extras 属性允许将特定于应用程序的元数据添加到任何 3D 切片 JSON 对象中。以下示例显示了一个带有附加应用程序特定名称属性的平铺对象。

{

{

"transform": [

"转型"：[

4.843178171884396, 1.2424271388626869, 0, 0,

4.843178171884396，1.2424271388626869，0，0，

-0.7993325488216595, 3.1159251367235608, 3.8278032889280675, 0,

-0.7993325488216595，3.1159251367235608，3.8278032889280675，0，

0.9511533376784163, -3.7077466670407433, 3.2168186118075526, 0,

0.951153376784163，-3.707746670407433，3.2168186118075526，0，

1215001.7612985559, -4736269.697480114, 4081650.708604793, 1

1215001.7612985559，-4736269.697480114，4081650.767678696

],

],

"boundingVolume": {

" boundingVolume": {

"box": [

【盒子】:[

0, 0, 6.701,

0，0，6.701，

3.738, 0, 0,

3.738，0，0，

0, 3.72, 0,

0，3.72，0，

0, 0, 13.402

0，0，13.402

]

]

},

},

"geometricError": 32,

"几何误差":32，

"content": {

"内容":{

"uri": "building.b3dm"

" uri": "building.b3dm "

```
            },

        },

    "extras": {
```

"附加功能"：

```
    "name": "Empire State Building"
```

"名称":"帝国大厦"

```
    }

    }

    }

}
```

Tile format specifications

平铺格式规格

Each tile's content.uri property may be the uri of binary blob that contains information for rendering the tile's 3D content.The content is an instance of one of the formats listed in the table below.

每个图块的 content.uri 属性可以是包含用于渲染图块 3D 内容的信息的二进制 blob 的 uri。内容是下表中所列格式之一的实例。

Format	Uses
Batched 3D Model (b3dm)	Heterogeneous 3D models.E.g. textured terrain and surfaces, 3D building exteriors and interiors, massive models.
Instanced 3D Model (i3dm)	3D model instances.E.g. trees, windmills, bolts.
Point Cloud (pnts)	Massive number of points.
Composite (cmpt)	Concatenate tiles of different formats into one tile.

版式	使用
批量 3D 模型(b3dm)	异构 3D 模型。例如纹理化的地形和表面、3D 建筑外部和内部、大型模型。

实例化三维模型(i3dm)	三维模型实例。例如树木、风车、螺栓。
点云	大量点数。
复合材料(cmpt)	将不同格式的图块连接成一个图块。

A tileset can contain any combination of tile formats.3D Tiles may also support different formats in the same tile using a Composite tile.

瓦片区可以包含瓦片格式的任意组合。3D 图块也可以使用复合图块支持同一图块中的不同格式。

Property reference

属性引用

Tileset

Tileset

A 3D Tiles tileset.

一个 3D 瓷砖镶嵌。

Properties

性能

	Type	Description	Required
asset	object	Metadata about the entire tileset.	Yes
properties	any	A dictionary object of metadata about per-feature properties.	No
geometricError	number	The error, in meters, introduced if this tileset is not rendered.At runtime, the geometric error is used to compute screen space error (SSE), i.e., the error measured in pixels.	Yes
root	object	A tile in a 3D Tiles tileset.	Yes
extensionsUsed	string [1-*]	Names of 3D Tiles	No

		extensions used somewhere in this tileset.	
extensionsRequired	string [1-*]	Names of 3D Tiles extensions required to properly load this tileset.	No
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
资产	目标	关于整个波浪集的元数据。	是
性能	任何的	关于每个要素属性的元数据的字典对象。	不
几何误差	数字	如果不呈现此颚化符集，则会引入以米为单位的错误。运行时，几何误差用于计算屏幕空间误差(SSE)，即以像素为单位测量的误差。	是
根	目标	3D 拼贴拼贴集中的拼贴。	是
扩展已使用	字符串[1-*]	此平铺集中某处使用的 3D 平铺扩展的名称。	不
需要扩展	字符串[1-*]	正确加载此瓦片集所需的 3D 瓦片扩展的名称。	不
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数	不

		据。	

Additional properties are not allowed.

不允许附加属性。

Tileset.asset

Tileset.asset

Metadata about the entire tileset.

关于整个波浪集的元数据。

Type: object

类型:对象

Required: Yes

必填:是

Tileset.properties

Tileset.properties

A dictionary object of metadata about per-feature properties.

关于每个要素属性的元数据的字典对象。

Type: any

类型:任何

Required: No

必填:否

Type of each property: object

每个属性的类型:对象

Tileset.geometricError

Tileset.geometricError

The error, in meters, introduced if this tileset is not rendered.At runtime, the geometric error is used to compute screen space error (SSE), i.e., the error measured in pixels.

如果不呈现此颚化符集，则会引入以米为单位的错误。运行时，几何误差用于计算屏幕空间误差(SSE)，即以像素为单位测量的误差。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

Tileset.root

Tileset.root

A tile in a 3D Tiles tileset.

3D 拼贴拼贴集中的拼贴。

Type: object

类型:对象

Required: Yes

必填:是

Tileset.extensionsUsed

Tileset.extensionsUsed 已使用

Names of 3D Tiles extensions used somewhere in this tileset.

此平铺集中某处使用的 3D 平铺扩展的名称。

Type: string [1-*]

类型:字符串[1-*]

Each element in the array must be unique.

数组中的每个元素必须是唯一的。

Required: No

必填:否

Tileset.extensionsRequired

需要 Tileset.extensionsRequired

Names of 3D Tiles extensions required to properly load this tileset.

正确加载此瓦片集所需的 3D 瓦片扩展的名称。

Type: string [1-*]

类型:字符串[1-*]

Each element in the array must be unique.

数组中的每个元素必须是唯一的。

Required: No

必填:否

Tileset.extensions

Tileset.extensions

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

Tileset.extras

Tileset.extras

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Asset

资产

Metadata about the entire tileset.

关于整个波浪集的元数据。

Properties

性能

	Type	Description	Required
version	string	The 3D Tiles version.The version defines the JSON schema for the tileset JSON and the base set of tile formats.	Yes
tilesetVersion	string	Application-specific version of this tileset, e.g., for when an existing tileset is updated.	No
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
版本	线	3D 图块版本。该版本定义了 tileset JSON 的 JSON 模式和平铺格式的基本集。	是
tilesetVersion	线	此颚化符集的特定于应用程序的版本，例如，用于更新现有颚化符集。	不
延长	目标	具有特定扩展对象的字	不

		典对象。	
额外费用	任何的	特定于应用程序的数据。	不

Additional properties are not allowed.

不允许附加属性。

Asset.version

资产版本

The 3D Tiles version.The version defines the JSON schema for the tileset JSON and the base set of tile formats.

3D 图块版本。该版本定义了 tileset JSON 的 JSON 模式和平铺格式的基本集。

Type: string

类型:字符串

Required: Yes

必填:是

Asset.tilesetVersion

资产. tilesetVersion

Application-specific version of this tileset, e.g., for when an existing tileset is updated.

此颚化符集的特定于应用程序的版本，例如，用于更新现有颚化符集。

Type: string

类型:字符串

Required: No

必填:否

Asset.extensions

资产.扩展

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

Asset.extras

资产。额外费用

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Bounding Volume

边界体积

A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.

包围图块或其内容的边界体积。只需要一个框、区域或球体属性。

Properties

性能

	Type	Description	Required
box	number [12]	An array of 12 numbers that define an oriented bounding box.The first three elements define the x, y, and z values for the center of the box.The next three elements (with indices	No

		3, 4, and 5) define the x axis direction and half-length.The next three elements (indices 6, 7, and 8) define the y axis direction and half-length.The last three elements (indices 9, 10, and 11) define the z axis direction and half-length.	
region	number [6]	An array of six numbers that define a bounding geographic region in EPSG:4979 coordinates with the order [west, south, east, north, minimum height, maximum height].Longitudes and latitudes are in radians, and heights are in meters above (or below) the WGS84 ellipsoid.	No
sphere	number [4]	An array of four numbers that define a bounding sphere.The first three elements define the x, y, and z values for the center of the sphere.The last element (with index 3) defines the radius in meters.	No
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
包厢	数字[12]	定义定向边界框的 12 个数字的数组。前三个元素定义了盒子中心的 x、y 和 z 值。接下来的	不

		三个元素(索引为 3、4 和 5)定义了 x 轴方向和半长。接下来的三个元素(索引 6、7 和 8)定义了 y 轴方向和半长。最后三个元素(索引 9、10 和 11)定义了 z 轴方向和半长。	
地区	数字[6]	定义 EPSG 边界地理区域的六个数字的数组:4979 坐标，顺序为[西、南、东、北、最小高度、最大高度]。经度和纬度以弧度为单位，高度以高于(或低于)WGS84 椭球面的米为单位。	不
范围	数字[4]	定义边界球体的四个数字的数组。前三个元素定义了球体中心的 x、y 和 z 值。最后一个元素(索引为 3)以米为单位定义半径。	不
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不

Additional properties are not allowed.

不允许附加属性。

BoundingVolume.box

边界体积.盒

An array of 12 numbers that define an oriented bounding box.The first three elements define the x, y, and z values for the center of the box.The next three elements (with indices 3, 4, and 5) define the x axis direction and half-length.The next three elements (indices 6, 7, and 8) define the y axis direction and half-length.The last three elements (indices 9, 10, and 11) define the z axis direction and half-length.

定义定向边界框的 12 个数字的数组。前三个元素定义了盒子中心的 x、y 和 z 值。接下来的三个元素(索引为 3、4 和 5)定义了 x 轴方向和半长。接下来的三个元素(索引 6、7 和 8)定义了 y 轴方向和半长。最后三个元素(索引 9、10 和 11)定义了 z 轴方向和半长。

Type: number [12]

类型:编号[12]

Required: No

必填:否

BoundingVolume.region

边界体积.区域

An array of six numbers that define a bounding geographic region in EPSG:4979 coordinates with the order [west, south, east, north, minimum height, maximum height].Longitudes and latitudes are in radians, and heights are in meters above (or below) the WGS84 ellipsoid.

定义 EPSG 边界地理区域的六个数字的数组:4979 坐标，顺序为[西、南、东、北、最小高度、最大高度]。经度和纬度以弧度为单位，高度以高于(或低于)WGS84 椭球面的米为单位。

Type: number [6]

类型:数字[6]

Required: No

必填:否

BoundingVolume.sphere

边界体积.球体

An array of four numbers that define a bounding sphere.The first three elements define the x, y, and z values for the center of the sphere.The last element (with index 3) defines the radius in meters.

定义边界球体的四个数字的数组。前三个元素定义了球体中心的 x、y 和 z 值。最后一个元素(索引为 3)以米为单位定义半径。

Type: number [4]

类型:数字[4]

Required: No

必填:否

BoundingVolume.extensions

边界卷.扩展

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

BoundingVolume.extras

边界卷。附加

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Extension

延长

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Additional properties are allowed.

允许附加属性。

Type of each property: object

每个属性的类型:对象

Extras

额外费用

Application-specific data.

特定于应用程序的数据。

Properties

性能

A dictionary object of metadata about per-feature properties.

关于每个要素属性的元数据的字典对象。

Properties

性能

	Type	Description	Required
maximum	number	The maximum value of this property of all the features in the tileset.	Yes
minimum	number	The minimum value of this property of all the features in the tileset.	Yes
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
最高的	数字	tileset 中所有要素的此属性的最大值。	是
最低限度	数字	tileset 中所有要素的此属性的最小值。	是
延长	目标	具有特定扩展对象的字典对象。	不

| 额外费用 | 任何的 | 特定于应用程序的数据。 | 不 |

Additional properties are not allowed.

不允许附加属性。

Properties.maximum

属性。最大值

The maximum value of this property of all the features in the tileset.

tileset 中所有要素的此属性的最大值。

Type: number

类型:数字

Required: Yes

必填:是

Properties.minimum

属性。最小值

The minimum value of this property of all the features in the tileset.

tileset 中所有要素的此属性的最小值。

Type: number

类型:数字

Required: Yes

必填:是

Properties.extensions

属性.扩展

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

Properties.extras

属性。附加功能

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Tile

瓷砖

A tile in a 3D Tiles tileset.

3D 拼贴拼贴集中的拼贴。

Properties

性能

	Type	Description	Required
boundingVolume	object	A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.	Yes
viewerRequestVolume	object	A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.	No
geometricError	number	The error, in meters,	Yes

		introduced if this tile is rendered and its children are not.At runtime, the geometric error is used to compute screen space error (SSE), i.e., the error measured in pixels.	
refine	string	Specifies if additive or replacement refinement is used when traversing the tileset for rendering.This property is required for the root tile of a tileset;it is optional for all other tiles.The default is to inherit from the parent tile.	No
transform	number [16]	A floating-point 4x4 affine transformation matrix, stored in column-major order, that transforms the tile's content - i.e., its features as well as content.boundingVolume, boundingVolume, and viewerRequestVolume - from the tile's local coordinate system to the parent tile's coordinate system, or, in the case of a root tile, from the tile's local coordinate system to the tileset's coordinate system.transform does not apply to geometricError, nor does it apply any volume property when the volume is a region, defined in EPSG:4979 coordinates.	No, default: [1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1]
content	object	Metadata about the tile's content and a link to the content.	No
children	array[]	An array of objects that	No

		define child tiles.Each child tile content is fully enclosed by its parent tile's bounding volume and, generally, has a geometricError less than its parent tile's geometricError.For leaf tiles, the length of this array is zero, and children may not be defined.	
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
边界体积	目标	包围图块或其内容的边界体积。只需要一个框、区域或球体属性。	是
viewerRequestVolume	目标	包围图块或其内容的边界体积。只需要一个框、区域或球体属性。	不
几何误差	数字	如果渲染此图块而其子图块未渲染，则会引入以米为单位的错误。运行时，几何误差用于计算屏幕空间误差(SSE)，即以像素为单位测量的误差。	是
改善	线	指定在遍历颚化符集进行渲染时是使用加法还是替换细化。tileset 的根切片需要此属性；它对于所有其他图块都是可选的。默认值是从父切片继承。	不

改变	数字[16]	一种浮点 4x4 仿射变换矩阵，以列主顺序存储，用于将图块的内容(即其特征以及内容、边界体积、边界体积和视图请求体积)从图块的本地坐标系变换到父图块的坐标系，或者，对于根图块，从图块的本地坐标系变换到图块的坐标系。变换不适用于几何误差，当体积是在 EPSG:4979 坐标中定义的区域时，它也不应用任何体积属性。	不，默认: [1，0，0，0，0，1，0，0，0，0，1，0，0，0，0，1]
内容	目标	关于图块内容的元数据和指向内容的链接。	不
孩子们	阵列[]	定义子切片的对象数组。每个子图块内容都被其父图块的边界体积完全包围，并且通常几何误差小于其父图块的几何误差。对于叶图块，此数组的长度为零，并且可能没有定义子项。	不
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不

Additional properties are not allowed.

不允许附加属性。

Tile.boundingVolume

平铺.边界卷

A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.

包围图块或其内容的边界体积。只需要一个框、区域或球体属性。

Type: object

类型:对象

Required: Yes

必填:是

Tile.viewerRequestVolume

平铺视图请求卷

A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.

包围图块或其内容的边界体积。只需要一个框、区域或球体属性。

Type: object

类型:对象

Required: No

必填:否

Tile.geometricError

瓷砖.几何误差

The error, in meters, introduced if this tile is rendered and its children are not.At runtime, the geometric error is used to compute screen space error (SSE), i.e., the error measured in pixels.

如果渲染此图块而其子图块未渲染，则会引入以米为单位的错误。运行时，几何误差用于计算屏幕空间误差(SSE)，即以像素为单位测量的误差。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

Tile.refine

瓷砖.精炼

Specifies if additive or replacement refinement is used when traversing the tileset for rendering.This property is required for the root tile of a tileset;it is optional for all other tiles.The default is to inherit from the parent tile.

指定在遍历颚化符集进行渲染时是使用加法还是替换细化。tileset 的根切片需要此属性；它对于所有其他图块都是可选的。默认值是从父切片继承。

Type: string

类型:字符串

Required: No

必填:否

Allowed values:

允许值:

"ADD"

"添加"

"REPLACE"

" REPLACE "

Tile.transform

平铺.变换

A floating-point 4x4 affine transformation matrix, stored in column-major order, that transforms the tile's content - i.e., its features as well as content.boundingVolume, boundingVolume, and viewerRequestVolume - from the tile's local coordinate system to the parent tile's coordinate system, or, in the case of a root tile, from the tile's local coordinate system to the tileset's coordinate system.transform does not apply to geometricError, nor does it apply any volume property when the volume is a region, defined in EPSG:4979 coordinates.

一种浮点 4x4 仿射变换矩阵，以列主顺序存储，用于将图块的内容(即其特征以及内容、边界体积、边界体积和视图请求体积)从图块的本地坐标系变换到父图块的坐标系，或者，对于根图块，从图块的本地坐标系变换到图块的坐标系。变换不适用于几何误差，当体积是在 EPSG:4979 坐标中定义的区域时，它也不应用任何体积属性。

Type: number [16]

类型:编号[16]

Required: No, default: [1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1]

必需:否，默认:[1，0，0，0，0，1，0，0，0，0，1，0，0，0，0，1]

Tile.content

平铺内容

Metadata about the tile's content and a link to the content.

关于图块内容的元数据和指向内容的链接。

Type: object

类型:对象

Required: No

必填:否

Tile.children

瓷砖，孩子们

An array of objects that define child tiles.Each child tile content is fully enclosed by its parent tile's bounding volume and, generally, has a geometricError less than its parent tile's geometricError.For leaf tiles, the length of this array is zero, and children may not be defined.

定义子切片的对象数组。每个子图块内容都被其父图块的边界体积完全包围，并且通常几何误差小于其父图块的几何误差。对于叶图块，此数组的长度为零，并且可能没有定义子项。

Type: array[]

类型:阵列[]

Each element in the array must be unique.

数组中的每个元素必须是唯一的。

Required: No

必填:否

Tile.extensions

平铺.扩展

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

Tile.extras

瓷砖。附加

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Tile Content

平铺内容

Metadata about the tile's content and a link to the content.

关于图块内容的元数据和指向内容的链接。

Properties

性能

	Type	Description	Required
boundingVolume	object	A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.	No
uri	string	A uri that points to the tile's content.When the uri is relative, it is relative to the referring tileset JSON file.	Yes
extensions	object	Dictionary object with extension-specific objects.	No

extras	any	Application-specific data.	No

	类型	描述	需要
边界体积	目标	包围图块或其内容的边界体积。只需要一个框、区域或球体属性。	不
上呼吸道感染	线	指向切片内容的 uri。当 uri 是相对的时，它是相对于引用的 tileset JSON 文件的。	是
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不

Additional properties are not allowed.

不允许附加属性。

TileContent.boundingVolume

TileContent.boundingVolume

A bounding volume that encloses a tile or its content.Exactly one box, region, or sphere property is required.

包围图块或其内容的边界体积。只需要一个框、区域或球体属性。

Type: object

类型:对象

Required: No

必填:否

TileContent.uri

TileContent.uri

A uri that points to the tile's content.When the uri is relative, it is relative to the referring tileset JSON file.

指向切片内容的 uri。当 uri 是相对的时，它是相对于引用的 tileset JSON 文件的。

Type: string

类型:字符串

Required: Yes

必填:是

TileContent.extensions

TileContent.extensions

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

TileContent.extras

TileContent.extras

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Feature Table

功能表

Overview

概观

A Feature Table is a component of a tile's binary body and describes position and appearance properties required to render each feature in a tile.The Batch Table, on the other hand, contains per-feature application-specific properties not necessarily used for rendering.

要素表是图块二元体的一个组成部分，描述了渲染图块中每个要素所需的位置和外观属性。另一方面，批处理表包含不一定用于渲染的每个功能的特定于应用程序的属性。

A Feature Table is used by tile formats like Batched 3D Model (b3dm) where each model is a feature, and Point Cloud (pnts) where each point is a feature.

要素表由切片格式使用，如批量 3D 模型(b3dm)，其中每个模型是一个要素，点云(pnts)，其中每个点是一个要素。

Per-feature properties are defined using tile format-specific semantics defined in each tile format's specification.For example, for Instanced 3D Model, SCALE_NON_UNIFORM defines the non-uniform scale applied to each 3D model instance.

每个特征的属性是使用每个图块格式规范中定义的图块格式特定语义来定义的。例如，对于实例化三维模型，"比例_不均匀"定义了应用于每个三维模型实例的不均匀比例。

Layout

布局

A Feature Table is composed of two parts: a JSON header and an optional binary body in little endian.The JSON property names are tile format-specific semantics, and their values can either be defined directly in the JSON, or refer to sections in the binary body.It is more efficient to store long numeric arrays in the binary body.The following figure shows the Feature Table layout:

特征表由两部分组成:JSON 头和可选的小端二进制体。JSON 属性名是特定于平铺格式的语义，它们的值可以直接在 JSON 中定义，也可以引用二进制体中的部分。在二进制体中存储长数值数组更有效。下图显示了要素表布局:
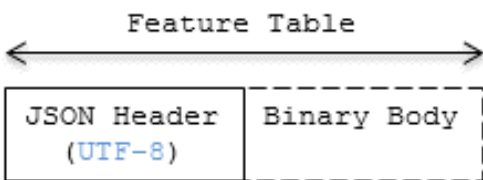
Figure 20: Feature Table layout

图 20:要素表布局

When a tile format includes a Feature Table, the Feature Table immediately follows the tile's header.The header will also contain featureTableJSONByteLength and

featureTableBinaryByteLength uint32 fields, which can be used to extract each respective part of the Feature Table.

当图块格式包含要素表时，要素表会紧跟图块的标头。标题还将包含 featureTableJSONByteLength 和 FeatureTableBinaryBytelength uint 32 字段，这些字段可用于提取要素表的各个部分。

Padding

填料

The JSON header must end on an 8-byte boundary within the containing tile binary.The JSON header must be padded with trailing Space characters (0x20) to satisfy this requirement.

JSON 头必须在包含平铺二进制文件的 8 字节边界上结束。JSON 头必须用尾随空格字符(0x20)填充以满足此要求。

The binary body must start and end on an 8-byte boundary within the containing tile binary.The binary body must be padded with additional bytes, of any value, to satisfy this requirement.

二进制主体必须在包含平铺二进制文件的 8 字节边界上开始和结束。为了满足这个要求，二进制体必须用任何值的附加字节填充。

Binary properties must start at a byte offset that is a multiple of the size in bytes of the property's implicit component type.For example, a property with the implicit component type FLOAT has 4 bytes per element, and therefore must start at an offset that is a multiple of 4.Preceding binary properties must be padded with additional bytes, of any value, to satisfy this requirement.

二进制属性必须以字节偏移量开始，该偏移量是属性隐式组件类型的字节数的倍数。例如，隐式组件类型 FLOAT 的属性每个元素有 4 个字节，因此必须从 4 的倍数的偏移量开始。为了满足这个要求，前面的二进制属性必须用任何值的附加字节填充。

JSON header

JSON 标题

Feature Table values can be represented in the JSON header in three different ways:

要素表值可以用三种不同的方式表示在 JSON 头中:

A single value or object, e.g., "INSTANCES_LENGTH" : 4.

单个值或对象，例如，"INSTANCES _ LENGTH":4。

This is used for global semantics like "INSTANCES_LENGTH", which defines the number of model instances in an Instanced 3D Model tile.

这用于全局语义，如"INSTANCES_LENGTH"，它定义了实例化三维模型切片中模型实例的数量。

An array of values, e.g., "POSITION" : [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0].

值的数组，例如，"位置":[1.0、0.0、0.0、0.0、1.0、0.0、0.0、0.0、0.0、0.0、1.0]。

This is used for per-feature semantics like "POSITION" in Instanced 3D Model.Above, each POSITION refers to a float32[3] data type so there are three features: Feature 0's position=(1.0, 0.0, 0.0), Feature 1's position=(0.0, 1.0, 0.0), Feature 2's position=(0.0, 0.0, 1.0).

这用于每个特征语义，如实例化三维模型中的"位置"。上面，每个位置都是指浮动 32[3]数据类型，因此有三个特征:特征 0 的位置=(1.0，0.0，0.0)，特征 1 的位置=(0.0，1.0，0.0)，特征 2 的位置=(0.0，0.0，1.0)。

A reference to data in the binary body, denoted by an object with a byteOffset property, e.g., "SCALE" : { "byteOffset" : 24}.

对二进制体中数据的引用，由一个具有字节数组属性的对象表示，例如，"比例":{"字节数组":24}。

byteOffset specifies a zero-based offset relative to the start of the binary body.The value of byteOffset must be a multiple of the size in bytes of the property's implicit component type, e.g., the "POSITION" property has the component type FLOAT (4 bytes), so the value of byteOffset must be of a multiple of 4.

byteOffset 指定相对于二进制体开始的从零开始的偏移量。byteOffset 的值必须是属性隐式组件类型的字节大小的倍数，例如，"位置"属性的组件类型为 FLOAT (4 字节)，因此 byteOffset 的值必须是 4 的倍数。

The semantic defines the allowed data type, e.g., when "POSITION" in Instanced 3D Model refers to the binary body, the component type is FLOAT and the number of components is 3.

语义定义了允许的数据类型，例如，当实例化三维模型中的"位置"指二进制体时，组件类型为 FLOAT，组件数量为 3。

Some semantics allow for overriding the implicit component type.These cases are specified in each tile format, e.g., "BATCH_ID" : { "byteOffset" : 24, "componentType" : "UNSIGNED_BYTE"}.

一些语义允许覆盖隐式组件类型。这些情况在每个图块格式中都有指定，例如，"BATCH _ ID":{"BYTEofSet":24，"componentType":"UNSIGNED _ BYTE"}。

The only valid properties in the JSON header are the defined semantics by the tile format and optional extras and extensions properties.Application-specific data should be stored in the Batch Table.

JSON 头中唯一有效的属性是由平铺格式定义的语义以及可选的附加和扩展属性。<mark>特定于应用程序的数据应该存储在批处理表中。</mark>

See Property reference for the full JSON header schema reference.The full JSON schema can be found in featureTable.schema.json.

有关完整的 JSON 头架构引用，请参见属性引用。完整的 JSON 模式可以在 featureTable.schema.json 中找到。

Binary body

二元体

When the JSON header includes a reference to the binary, the provided byteOffset is used to index into the data.The following figure shows indexing into the Feature Table binary body:

当 JSON 头包含对二进制文件的引用时，所提供的 byteOffset 用于索引数据。下图显示了对要素表二进制体的索引:
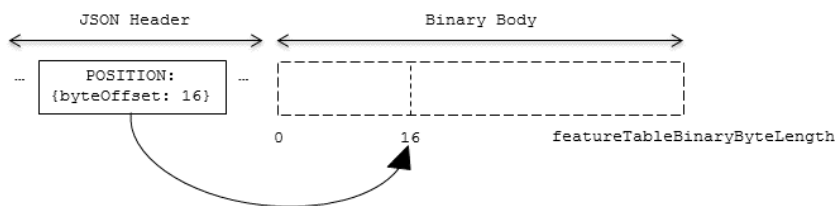
Figure 21: Feature Table binary body layout

图 21:特征表二进制体布局

Values can be retrieved using the number of features, featuresLength;the desired feature id, featureId;and the data type (component type and number of components) for the feature semantic.

可以使用要素数量、要素长度来检索值；所需的特征标识，特征标识；和特征语义的数据类型(组件类型和组件数量)。

Implementation example

实现示例

This section is non-normative

本节是非规范性的

The following example accesses the position property using the POSITION semantic, which has a float32[3] data type:

以下示例使用 position 语义访问 POSITION 属性，该语义具有 float32[3]数据类型:

var byteOffset = featureTableJSON.POSITION.byteOffset;

变量 byteOffset = featureTableJSON。位置。字节数组；

var positionArray = new Float32Array(featureTableBinary.buffer, byteOffset, featuresLength * 3);// There are three components for each POSITION feature.

可变位置数组=新浮动数组(featureTableBinary.buffer，byteOffset，FeatureSlength * 3)；//每个位置特征有三个组件。

var position = positionArray.subarray(featureId * 3, featureId * 3 + 3);// Using subarray creates a view into the array, and not a new array.

可变位置=位置数组.子数组(特征号* 3，特征号* 3+3)；//使用子数组创建数组中的视图，而不是新数组。

Code for reading the Feature Table can be found in Cesium3DTileFeatureTable.js in the Cesium implementation of 3D Tiles.

<mark>读取特征表的代码可以在铯三维瓦片的铯实现中的铯三维瓦片引用表中找到。</mark>

Property reference

属性引用

Feature Table

功能表

A set of semantics containing per-tile and per-feature values defining the position and appearance properties for features in a tile.

一组语义，包含定义图块中要素的位置和外观属性的每个图块和每个要素的值。

Properties

性能

	Type	Description	Required
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不

Additional properties are allowed.

允许附加属性。

Type of each property: Property

每个属性的类型:属性

FeatureTable.extensions

功能表.扩展

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

FeatureTable.extras

特色表。附加功能

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

BinaryBodyReference

BinaryBodyReference

An object defining the reference to a section of the binary body of the features table where the property values are stored if not defined directly in the JSON.

一个对象，定义对要素表的二进制主体的一部分的引用，如果没有直接在 JSON 中定义，属性值将存储在该部分中。

Properties

性能

	Type	Description	Required

byteOffset	number	The offset into the buffer in bytes.	Yes

	类型	描述	需要
字节数组	数字	缓冲区的偏移量，以字节为单位。	是

Additional properties are allowed.

允许附加属性。

BinaryBodyReference.byteOffset

binary body reference . byTeofSet

The offset into the buffer in bytes.

缓冲区的偏移量，以字节为单位。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

Property

财产

A user-defined property which specifies per-feature application-specific metadata in a tile.Values either can be defined directly in the JSON as an array, or can refer to sections in the binary body with a BinaryBodyReference object.

一种用户定义的属性，用于指定图块中每个要素的特定于应用程序的元数据。值可以直接在 JSON 中定义为数组，也可以用 BinaryBodyReference 对象引用二进制体中的部分。

Batch Table

批处理表

Overview

概观

A Batch Table is a component of a tile's binary body and contains per-feature application-specific properties in a tile.These properties are queried at runtime for declarative styling and any application-specific use cases such as populating a UI or issuing a REST API request.Some example Batch Table properties are building heights, geographic coordinates, and database primary keys.

批处理表是图块二进制体的一个组件，包含图块中每个特征的应用程序特定属性。这些属性在运行时被查询以获得声明性样式和任何特定于应用程序的用例，例如填充用户界面或发出 REST API 请求。批处理表属性的一些示例包括建筑高度、地理坐标和数据库主键。

A Batch Table is used by the following tile formats:

批处理表由以下平铺格式使用:

Batched 3D Model (b3dm)

批量 3D 模型(b3dm)

Instanced 3D Model (i3dm)

实例化三维模型(i3dm)

Point Cloud (pnts)

点云

Layout

布局

A Batch Table is composed of two parts: a JSON header and an optional binary body in little endian.The JSON describes the properties, whose values either can be defined directly in the JSON as an array, or can refer to sections in the binary body.It is more efficient to store long numeric arrays in the binary body.The following figure shows the Batch Table layout:

批处理表由两部分组成:JSON 头和可选的小端二进制体。JSON 描述属性，其值可以直接在 JSON 中定义为数组，也可以引用二进制体中的部分。在二进制体中存储长数值数组更有效。下图显示了批处理表布局:
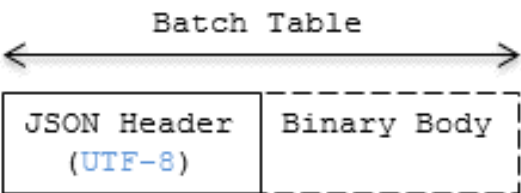
Figure 22: Batch Table layout

图 22:批处理表布局

When a tile format includes a Batch Table, the Batch Table immediately follows the tile's Feature Table.

<mark>当图块格式包含批次表时，批次表会紧跟图块的特征表。</mark>

The header will also contain batchTableJSONByteLength and batchTableBinaryByteLength uint32 fields, which can be used to extract each respective part of the Batch Table.

标题还将包含 batchTableJSONByteLength 和 BatchTableBinaryBytelength uint 32 字段，这些字段可用于提取批处理表的各个部分。

Padding

填料

The JSON header must end on an 8-byte boundary within the containing tile binary.The JSON header must be padded with trailing Space characters (0x20) to satisfy this requirement.

JSON 头必须在包含平铺二进制文件的 8 字节边界上结束。JSON 头必须用尾随空格字符(0x20)填充以满足此要求。

The binary body must start and end on an 8-byte boundary within the containing tile binary.The binary body must be padded with additional bytes, of any value, to satisfy this requirement.

二进制主体必须在包含平铺二进制文件的 8 字节边界上开始和结束。为了满足这个要求，二进制体必须用任何值的附加字节填充。

Binary properties must start at a byte offset that is a multiple of the size in bytes of the property's componentType.For example, a property with the component type FLOAT has 4 bytes per element, and therefore must start at an offset that is a multiple of 4.Preceding binary properties must be padded with additional bytes, of any value, to satisfy this requirement.

二进制属性必须以字节偏移量开始，该偏移量是属性组件类型的字节数的倍数。例如，组件类型为 FLOAT 的属性每个元素有 4 个字节，因此必须从 4 的倍数的偏移量开始。为了满足这个要求，前面的二进制属性必须用任何值的附加字节填充。

JSON header

JSON 标题

Batch Table values can be represented in the JSON header in two different ways:

批处理表值可以用两种不同的方式表示在 JSON 头中:

An array of values, e.g., "name": ['name1', 'name2', 'name3'] or "height" : [10.0, 20.0, 15.0].

值的数组，例如，"名称":["名称 1"、"名称 2"、"名称 3"或"高度":[10.0、20.0、15.0]。

Array elements can be any valid JSON data type, including objects and arrays.Elements may be null.

数组元素可以是任何有效的 JSON 数据类型，包括对象和数组。元素可能为空。

The length of each array is equal to batchLength, which is specified in each tile format.This is the number of features in the tile.For example, batchLength may be the number of models in a b3dm tile, the number of instances in a i3dm tile, or the number of points (or number of objects) in a pnts tile.

每个数组的长度等于 batchLength，这是在每个平铺格式中指定的。这是图块中的特征数量。例如，batchLength 可以是 b3dm 切片中的模型数、i3dm 切片中的实例数或 pnts 切片中的点数(或对象数)。

A reference to data in the binary body, denoted by an object with byteOffset, componentType, and type properties, e.g., "height" : { "byteOffset" : 24, "componentType" : "FLOAT", "type" : "SCALAR"}.

对二进制体中数据的引用，由具有字节数组、组件类型和类型属性的对象表示，例如，"高度":{"字节数组":24，"组件类型":"浮点"、"类型":"标量"。

byteOffset specifies a zero-based offset relative to the start of the binary body.The value of byteOffset must be a multiple of the size in bytes of the property's componentType, e.g., a property with the component type FLOAT must have a byteOffset value that is a multiple of 4.

byteOffset 指定相对于二进制体开始的从零开始的偏移量。byteOffset 的值必须是属性组件类型的字节大小的倍数，例如，组件类型为 FLOAT 的属性必须具有 4 的倍数的 byteOffset 值。

componentType is the datatype of components in the attribute.Allowed values are "BYTE", "UNSIGNED_BYTE", "SHORT", "UNSIGNED_SHORT", "INT", "UNSIGNED_INT", "FLOAT", and "DOUBLE".

组件类型是属性中组件的数据类型。允许的值是"字节"、"无符号_字节"、"短"、"无符号_短"、"整数"、"无符号_整数"、"浮点"和"双精度"。

type specifies if the property is a scalar or vector.Allowed values are "SCALAR", "VEC2", "VEC3", and "VEC4".

type 指定属性是标量还是向量。允许的值是"标量"、"矢量 2"、"矢量 3"和"矢量 4"。

The Batch Table JSON is a UTF-8 string containing JSON.

批处理表 JSON 是一个包含 JSON 的 UTF-8 字符串。

Implementation Note: In JavaScript, the Batch Table JSON can be extracted from an ArrayBuffer using the TextDecoder JavaScript API and transformed to a JavaScript object with JSON.parse.

实现注意:在 JavaScript 中，批处理表 JSON 可以使用文本解码器 JavaScript 应用程序接口从数组缓冲区中提取出来，并通过 JSON.parse 转换为一个 JavaScript 对象。

A batchId is used to access elements in each array and extract the corresponding properties.For example, the following Batch Table has properties for a batch of two features:

batchId 用于访问每个数组中的元素并提取相应的属性。例如，以下批处理表包含两个功能的批处理属性：

{

{

"id" : ["unique id", "another unique id"],

"id":["唯一 id"，"另一个唯一 id"]，

"displayName" : ["Building name", "Another building name"],

"显示名称":["建筑名称"、"另一建筑名称"]，

"yearBuilt" : [1999, 2015],

"年建成":[1999 年，2015 年，

"address" : [{"street" : "Main Street", "houseNumber" : "1"}, {"street" : "Main Street", "houseNumber" : "2"}]

"地址":[{"街":"主街"、"门牌号":"1"}、{"街":"主街"、"门牌号":"2"}]

}

}

The properties for the feature with batchId = 0 are

batchId = 0 的特征的属性是

id[0] = 'unique id';

id[0] ="唯一 id"；

displayName[0] = 'Building name';

显示名称[0] ="建筑名称"；

yearBuilt[0] = 1999;

年建成[0] = 1999 年；

address[0] = {street : 'Main Street', houseNumber : '1'};

地址[0] = {street : 'Main Street '，house number:' 1 ' }；

The properties for batchId = 1 are

batchId = 1 的属性是

id[1] = 'another unique id';

id[1] ="另一个唯一的 id";

displayName[1] = 'Another building name';

显示名称[1] ="另一个建筑名称";

yearBuilt[1] = 2015;

年建成[1] = 2015 年;

address[1] = {street : 'Main Street', houseNumber : '2'};

地址[1]= {街道:"主要街道",门牌号:"2"};

See Property reference for the full JSON header schema reference.The full JSON schema can be found in batchTable.schema.json.

有关完整的 JSON 头架构引用，请参见属性引用。完整的 JSON 模式可以在 batchTable.schema.json 中找到。

Binary body

二元体

When the JSON header includes a reference to the binary section, the provided byteOffset is used to index into the data, as shown in the following figure:

当 JSON 头包含对二进制部分的引用时，提供的 byteOffset 用于索引数据，如下图所示:
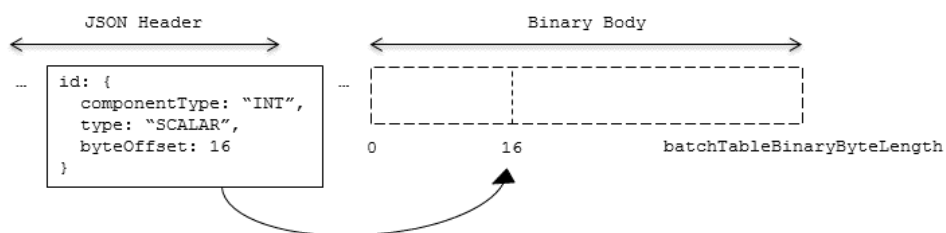
Figure 23: Batch Table binary body layout

图 23:批处理表二进制体布局

Values can be retrieved using the number of features, batchLength;the desired batch id, batchId;and the componentType and type defined in the JSON header.

可以使用要素数量 batchLength 检索值；所需的批次 id，batchId 以及 JSON 头中定义的组件类型和类型。

The following tables can be used to compute the byte size of a property.

下表可用于计算属性的字节大小。

componentType	Size in bytes
"BYTE"	1
"UNSIGNED_BYTE"	1
"SHORT"	2
"UNSIGNED_SHORT"	2
"INT"	4
"UNSIGNED_INT"	4
"FLOAT"	4
"DOUBLE"	8
type	Number of components
"SCALAR"	1
"VEC2"	2
"VEC3"	3
"VEC4"	4

组件类型	字节大小
" BYTE "	1
" UNSIGNED_BYTE "	1
"SHORT"	2
" UNSIGNED_SHORT "	2
" INT "	4
" UNSIGNED_INT "	4
" FLOAT "	4

"双倍"	8
类型	组件数量
" SCALAR "	1
"VEC2"	2
"VEC3"	3
"VEC4"	4

Implementation example

实现示例

This section is non-normative

本节是非规范性的

The following examples access the "height" and "geographic" values respectively given the following Batch Table JSON with batchLength of 10:

以下示例分别访问"高度"和"地理"值，给出了批处理表 JSON，批处理长度为 10:

{

{

"height" : {

"身高":{

"byteOffset" : 0,

"字节数组":0，

"componentType" : "FLOAT",

"组件类型":"浮动"，

"type" : "SCALAR"

"类型":" SCALAR "

},

}，

```
"geographic" : {

"地理":{

"byteOffset" : 40,

"字节数组":40，

"componentType" : "DOUBLE",

"组件类型":"双"，

"type" : "VEC3"

"类型":" VEC3 "

}

}

}

}
```

To get the "height" values:

要获得"高度"值:

var height = batchTableJSON.height;

var height = BatchTablejson . height；

var byteOffset = height.byteOffset;

变量 byteOffset =高度。byteOffset

var componentType = height.componentType;

变量组件类型=高度。组件类型；

var type = height.type;

变量类型=高度类型；

var heightArrayByteLength = batchLength * sizeInBytes(componentType) * numberOfComponents(type);// 10 * 4 * 1

可变高度阵列长度=批处理长度*大小字节(组件类型)*组件数量(类型)；// 10 * 4 * 1

var heightArray = new Float32Array(batchTableBinary.buffer, byteOffset, heightArrayByteLength);

可变高度数组=新浮动32数组(batchTableBinary.buffer，byteOffset，HeightArraybytelength)；

var heightOfFeature = heightArray[batchId];

不同高度特征=高度阵列[巴特奇];

To get the "geographic" values:

要获得"地理"值:

var geographic = batchTableJSON.geographic;

var geographic = BatchTablejson . geographic;

var byteOffset = geographic.byteOffset;

变量 byteOffset =地理。byteOffset

var componentType = geographic.componentType;

变量组件类型=地理组件类型;

var type = geographic.type;

var 类型=地理类型;

var componentSizeInBytes = sizeInBytes(componentType)

可变组件大小字节=大小字节(组件类型)

var numberOfComponents = numberOfComponents(type);

可变组件数=组件数(类型);

var geographicArrayByteLength = batchLength * componentSizeInBytes * numberOfComponents // 10 * 8 * 3

可变地理区域长度=批处理长度*组件大小字节*组件数量// 10 * 8 * 3

var geographicArray = new Float64Array(batchTableBinary.buffer, byteOffset, geographicArrayByteLength);

var geographicArray =新浮点 64 数组
(batchTableBinary.buffer，byteOffset，GeographicRayBytelength);

// Using subarray creates a view into the array, and not a new array.

//使用子数组创建数组中的视图，而不是新数组。

var geographicOfFeature = positionArray.subarray(batchId * numberOfComponents, batchId * numberOfComponents + numberOfComponents);

变化的地理属性=位置数组.子数组(组数*组件数，组数*组件数+组件数);

Code for reading the Batch Table can be found in Cesium3DTileBatchTable.js in the Cesium implementation of 3D Tiles.

读取批处理表的代码可以在铯三维瓦片的铯实现中找到。

Property reference

属性引用

Batch Table

批处理表

A set of properties defining application-specific metadata for features in a tile.

为图块中的要素定义特定于应用程序的元数据的一组属性。

Properties

性能

	Type	Description	Required
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No

	类型	描述	需要
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不

Additional properties are allowed.

允许附加属性。

Type of each property: Property

每个属性的类型:属性

BatchTable.extensions

BatchTable.extensions

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

BatchTable.extras

BatchTable.extras

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

BinaryBodyReference

BinaryBodyReference

An object defining the reference to a section of the binary body of the batch table where the property values are stored if not defined directly in the JSON.

一个对象，定义对批处理表二进制体中存储属性值的部分的引用，如果没有直接在 JSON 中定义属性值的话。

Properties

性能

	Type	Description	Required
byteOffset	number	The offset into the	Yes

		buffer in bytes.	
componentType	string	The datatype of components in the property.	Yes
type	string	Specifies if the property is a scalar or vector.	Yes

	类型	描述	需要
字节数组	数字	缓冲区的偏移量，以字节为单位。	是
组件类型	线	属性中组件的数据类型。	是
类型	线	指定属性是标量还是向量。	是

Additional properties are allowed.

允许附加属性。

BinaryBodyReference.byteOffset

binary body reference . byTeofSet

The offset into the buffer in bytes.

缓冲区的偏移量，以字节为单位。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

BinaryBodyReference.componentType

binary body reference . componentType

The datatype of components in the property.

属性中组件的数据类型。

Type: string

类型:字符串

Required: Yes

必填:是

Allowed values:

允许值:

"BYTE"

" BYTE "

"UNSIGNED_BYTE"

" UNSIGNED_BYTE "

"SHORT"

"SHORT"

"UNSIGNED_SHORT"

" UNSIGNED_SHORT "

"INT"

" INT "

"UNSIGNED_INT"

" UNSIGNED_INT "

"FLOAT"

" FLOAT "

"DOUBLE"

"双倍"

BinaryBodyReference.type

BinaryBodyReference.type

Specifies if the property is a scalar or vector.

指定属性是标量还是向量。

Type: string

类型:字符串

Required: Yes

必填:是

Allowed values:

允许值:

"SCALAR"

" SCALAR "

"VEC2"

"VEC2"

"VEC3"

"VEC3"

"VEC4"

"VEC4"

Property

财产

A user-defined property which specifies per-feature application-specific metadata in a tile.Values either can be defined directly in the JSON as an array, or can refer to sections in the binary body with a BinaryBodyReference object.

一种用户定义的属性，用于指定图块中每个要素的特定于应用程序的元数据。值可以直接在 JSON 中定义为数组，也可以用 BinaryBodyReference 对象引用二进制体中的部分。

Tile format specifications

平铺格式规格

Each tile's content.uri property points to a tile that is one of the formats listed in the table below.

每个图块的 content.uri 属性指向下表中列出的格式之一的图块。

Format	Uses

Batched 3D Model (b3dm)	Heterogeneous 3D models.E.g. textured terrain and surfaces, 3D building exteriors and interiors, massive models.
Instanced 3D Model (i3dm)	3D model instances.E.g. trees, windmills, bolts.
Point Cloud (pnts)	Massive number of points.
Composite (cmpt)	Concatenate tiles of different formats into one tile.

版式	使用
批量 3D 模型(b3dm)	异构 3D 模型。例如纹理化的地形和表面、3D 建筑外部和内部、大型模型。
实例化三维模型(i3dm)	三维模型实例。例如树木、风车、螺栓。
点云	大量点数。
复合材料(cmpt)	将不同格式的图块连接成一个图块。

A tileset can contain any combination of tile formats.3D Tiles may also support different formats in the same tile using a Composite tile.

瓦片区可以包含瓦片格式的任意组合。3D 图块也可以使用复合图块支持同一图块中的不同格式。

Batched 3D Model

批量 3D 模型

Overview

概观

Batched 3D Model allows offline batching of heterogeneous 3D models, such as different buildings in a city, for efficient streaming to a web client for rendering and interaction.Efficiency comes from transferring multiple models in a single request and rendering them in the least number of WebGL draw calls necessary.Using the core 3D Tiles spec language, each model is a feature.

批量 3D 模型允许离线批量处理异构 3D 模型，例如城市中的不同建筑，以便高效地流式传输到网络客户端进行渲染和交互。效率来自于在一个请求中传输多个模型，并在最少的必要 WebGL 绘图调用中呈现它们。使用核心 3D 切片规范语言，每个模型都是一个特征。

Per-model properties, such as IDs, enable individual models to be identified and updated at runtime, e.g., show/hide, highlight color, etc.Properties may be used, for example, to query a web service to access metadata, such as passing a building's ID to get its address.Or a property might be referenced

on the fly for changing a model's appearance, e.g., changing highlight color based on a property value.

每个模型的属性，例如 IDs，能够在运行时识别和更新单个模型，例如显示/隐藏、高亮颜色等。例如，属性可用于查询访问元数据的网络服务，例如传递建筑物的标识以获取其地址。或者可以动态引用属性来改变模型的外观，例如，基于属性值改变高亮颜色。

A Batched 3D Model tile is a binary blob in little endian.

批量 3D 模型切片是以小端序排列的二进制斑点。

Layout

布局

A tile is composed of two sections: a header immediately followed by a body.The following figure shows the Batched 3D Model layout (dashes indicate optional fields):

瓷砖由两部分组成:一个头部紧跟着一个主体。下图显示了批量 3D 模型布局(虚线表示可选字段):
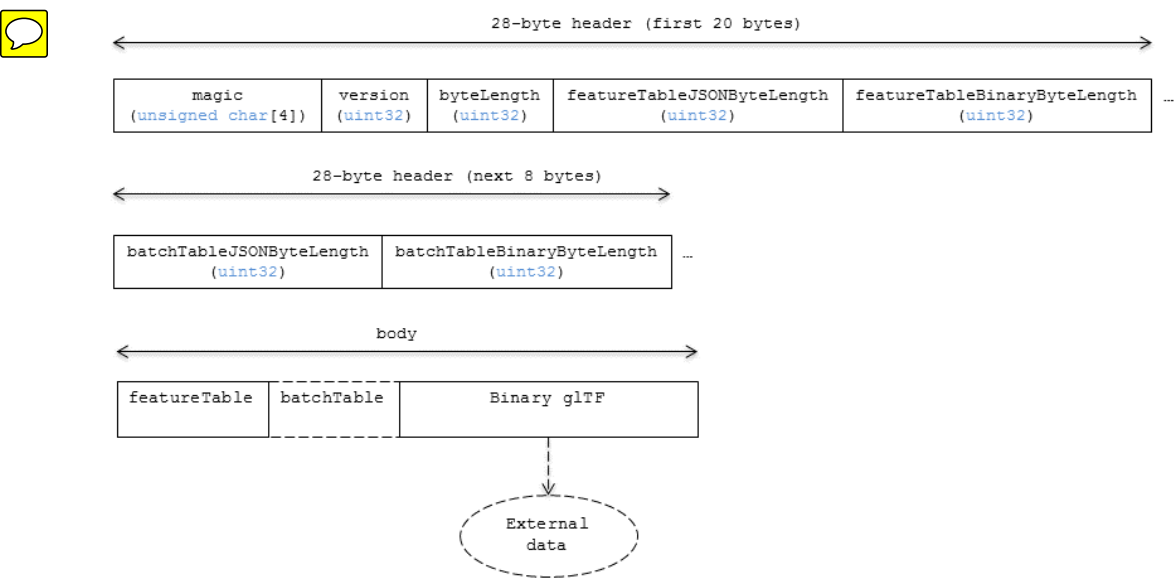
Figure 24: Batched 3D Model layout

图 24:批量 3D 模型布局

Padding

填料

A tile's byteLength must be aligned to an 8-byte boundary.The contained Feature Table and Batch Table must conform to their respective padding requirement.

图块的字节长度必须与 8 字节边界对齐。包含的特征表和批处理表必须符合它们各自的填充要求。

The binary glTF must start and end on an 8-byte boundary so that glTF's byte-alignment guarantees are met.This can be done by padding the Feature Table or Batch Table if they are present.

二进制 glTF 必须在 8 字节的边界上开始和结束，以满足 glTF 的字节对齐保证。这可以通过填充特征表或批处理表(如果存在)来完成。

Header

页眉

The 28-byte header contains the following fields:

28 字节报头包含以下字段:

Field name	Data type	Description
magic	4-byte ANSI string	"b3dm".This can be used to identify the content as a Batched 3D Model tile.
version	uint32	The version of the Batched 3D Model format.It is currently 1.
byteLength	uint32	The length of the entire tile, including the header, in bytes.
featureTableJSONByteLength	uint32	The length of the Feature Table JSON section in bytes.
featureTableBinaryByteLength	uint32	The length of the Feature Table binary section in bytes.
batchTableJSONByteLength	uint32	The length of the Batch Table JSON section in bytes.Zero indicates there is no Batch Table.
batchTableBinaryByteLength	uint32	The length of the Batch Table binary section in bytes.If batchTableJSONByteLength is zero, this will also be zero.

字段名	数据类型	描述
魔法	4 字节 ANSI 字符串	"b3dm"。这可用于将内容标识为批量 3D 模型切片。

版本	uint32	批量 3D 模型格式的版本。目前是 1。
byteLength	uint32	整个图块的长度，包括标头，以字节为单位。
featureTableJSONByteLength	uint32	特征表 JSON 部分的长度，以字节为单位。
功能表二进制字节长度	uint32	功能表二进制部分的长度，以字节为单位。
batchTableJSONByteLength	uint32	批处理表 JSON 部分的长度，以字节为单位。零表示没有批处理表。
batchTableBinaryByteLength	uint32	批处理表二进制部分的长度，以字节为单位。如果 batchTableJSONByteLength 为零，这也将为零。

The body section immediately follows the header section, and is composed of three fields: Feature Table, Batch Table, and Binary glTF.

正文部分紧跟着标题部分，由三个字段组成:特征表、批处理表和二进制 glTF。

Feature Table

功能表

Contains values for b3dm semantics.

包含 b3dm 语义的值。

More information is available in the Feature Table specification.

特性表规范中提供了更多信息。

Semantics

语义学

Feature semantics

特征语义

There are currently no per-feature semantics.

目前没有每个功能的语义。

Global semantics

全局语义学

These semantics define global properties for all features.

这些语义定义了所有特性的全局属性。

Semantic	Data Type	Description	Required
BATCH_LENGTH	uint32	The number of distinguishable models, also called features, in the batch.If the Binary glTF does not have a batchId attribute, this field must be 0.	Yes.
RTC_CENTER	float32[3]	A 3-component array of numbers defining the center position when positions are defined relative-to-center, (see Coordinate system).	No.

语义的	数据类型	描述	需要
BATCH_LENGTH	uint32	批次中可区分模型的数量，也称为特征。如果二进制 glTF 没有 batchId 属性，则该字段必须为 0。	是的。
RTC_CENTER	浮动 32[3]	当相对于中心定义位置时，定义中心位置的三分量数字数组(参见坐标系)。	没有。

Batch Table

批处理表

The Batch Table contains per-model application-specific metadata, indexable by batchId, that can be used for declarative styling and application-specific use cases such as populating a UI or issuing a REST API request.In the binary glTF section, each vertex has an numeric batchId attribute in the

integer range [0, number of models in the batch - 1].The batchId indicates the model to which the vertex belongs.This allows models to be batched together and still be identifiable.

批处理表包含每个模型的特定于应用程序的元数据，可由批处理索引，可用于声明性样式和特定于应用程序的用例，如填充用户界面或发出 REST 应用程序接口请求。在二进制 glTF 部分中，每个顶点都有一个整数范围内的数值 batchId 属性[0，批次中的模型数- 1]。batchId 表示顶点所属的模型。这使得模型可以一起批量处理，并且仍然是可识别的。

See the Batch Table reference for more information.

有关更多信息，请参见批处理表参考。

Binary glTF

二元 glTF

Batched 3D Model embeds glTF 2.0 containing model geometry and texture information.

批量 3D 模型嵌入包含模型几何和纹理信息的 glTF 2.0。

The binary glTF immediately follows the Feature Table and Batch Table.It may embed all of its geometry, texture, and animations, or it may refer to external sources for some or all of these data.

二进制 glTF 紧跟在特征表和批处理表之后。它可以嵌入它的所有几何图形、纹理和动画，或者它可以引用外部源来获取部分或全部这些数据。

As described above, each vertex has a batchId attribute indicating the model to which it belongs.For example, vertices for a batch with three models may look like this:

如上所述，每个顶点都有一个 batchId 属性，指示它所属的模型。例如，具有三个模型的批次的顶点可能如下所示:

batchId: [0, 0, 0, ..., 1, 1, 1, ..., 2, 2, 2, ...]

巴特奇:[0，0，0，...，1，1，1，...，2，2，2，...]

position: [xyz, xyz, xyz, ..., xyz, xyz, xyz, ..., xyz, xyz, xyz, ...]

位置:[xyz，xyz，xyz，...，xyz，xyz，xyz，...，xyz，xyz，xyz，...]

normal: [xyz, xyz, xyz, ..., xyz, xyz, xyz, ..., xyz, xyz, xyz, ...]

正常:[xyz，xyz，xyz，...，xyz，xyz，xyz，...，xyz，xyz，xyz，...]

Vertices do not need to be ordered by batchId, so the following is also OK:

顶点不需要由 batchId 排序，因此以下也可以:

batchId: [0, 1, 2, ..., 2, 1, 0, ..., 1, 2, 0, ...]

巴特奇:[0，1，2，...，2，1，0，...，1，2，0，...]

position: [xyz, xyz, xyz, ..., xyz, xyz, xyz, ..., xyz, xyz, xyz, ...]

位置:[xyz，xyz，xyz，...，xyz，xyz，xyz，...，xyz，xyz，xyz，...]

normal: [xyz, xyz, xyz, ..., xyz, xyz, xyz, ..., xyz, xyz, xyz, ...]

正常:[xyz，xyz，xyz，...，xyz，xyz，xyz，...，xyz，xyz，xyz，...]

Note that a vertex can't belong to more than one model;in that case, the vertex needs to be duplicated so the batchIds can be assigned.

请注意，一个顶点不能属于多个模型；在这种情况下，需要复制顶点，以便可以分配 batchIds。

The batchId parameter is specified in a glTF mesh primitive by providing the _BATCHID attribute semantic, along with the index of the batchId accessor.For example,

batchId 参数是通过提供_BATCHID 属性语义以及 BATCHID 访问器的索引在 glTF 网格原语中指定的。例如，

"primitives": [

"原始人" :[

{

{

"attributes": {

"属性":{

"_BATCHID": 0

" _BATCHID": 0

}

}

}

}

]

]

{

{

"accessors": [

"访问者":[

{

{

"bufferView": 1,

《水牛城评论》:1、

"byteOffset": 0,

"字节数组":0,

"componentType": 5125,

【组件类型】:5125,

"count": 4860,

【计数】:4860、

"max": [2],

"最大":[2 号,

"min": [0],

"最小值":[0",

"type": "SCALAR"

"类型":" SCALAR "

}

}

]

]

}

}

The accessor.type must be a value of "SCALAR".All other properties must conform to the glTF schema, but have no additional requirements.

访问器类型必须是"SCALAR"值。所有其他属性必须符合 glTF 模式，但没有附加要求。

When a Batch Table is present or the BATCH_LENGTH property is greater than 0, the _BATCHID attribute is required;otherwise, it is not.

当批处理表存在或 BATCH_LENGTH 属性大于 0 时，需要_BATCHID 属性；否则，就不是了。

Coordinate system

坐标系

By default embedded glTFs use a right handed coordinate system where the y-axis is up.For consistency with the z-up coordinate system of 3D Tiles, glTFs must be transformed at runtime.See glTF transforms for more details.

默认情况下，嵌入式 glTFs 使用 y 轴朝上的右手坐标系。为了与 3D 切片的 z 向上坐标系保持一致，glTFs 必须在运行时进行转换。有关更多详细信息，请参见 glTF 转换。

Vertex positions may be defined relative-to-center for high-precision rendering, see Precisions, Precisions.If defined, RTC_CENTER specifies the center position that all vertex positions are relative to after the coordinate system transform and glTF node hierarchy transforms have been applied.

对于高精度渲染，顶点位置可以相对于中心进行定义，请参见精度、精度。如果已定义，RTC_CENTER 指定应用坐标系变换和 glTF 节点层次结构变换后所有顶点位置相对的中心位置。

File extension and MIME type

文件扩展名和 MIME 类型

Batched 3D Model tiles use the .b3dm extension and application/octet-stream MIME type.

批量 3D 模型切片使用. b3dm 扩展和应用程序/八位字节流 MIME 类型。

An explicit file extension is optional.Valid implementations may ignore it and identify a content's format by the magic field in its header.

显式文件扩展名是可选的。有效的实现可以忽略它，并通过标题中的魔法字段来识别内容的格式。

Implementation example

实现示例

This section is non-normative

本节是非规范性的

Code for reading the header can be found in

读取标题的代码可以在中找到

Batched3DModelTileContent.js

批处理 d3DModelTileContent.js

in the Cesium implementation of 3D Tiles.

在三维瓦片的铯实施中。

Property reference

属性引用

Batched 3D Model Feature Table

批量 3D 模型特征表

A set of Batched 3D Model semantics that contain additional information about features in a tile.

一组批量 3D 模型语义，包含关于图块中特征的附加信息。

Properties

性能

	Type	Description	Required
extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No
BATCH_LENGTH	object, number [1], number	A GlobalPropertyScalar object defining a numeric property for all features.See the corresponding property semantic in Semantics.	Yes
RTC_CENTER	object, number [3]	A GlobalPropertyCartesian3 object defining a 3-component numeric property for all features.See the corresponding property semantic in Semantics.	No

	类型	描述	需要
延长	目标	具有特定扩展对象的字	不

		典对象。	
额外费用	任何的	特定于应用程序的数据。	不
BATCH_LENGTH	对象，编号[1]，编号	一个全局属性标量对象，为所有要素定义一个数值属性。参见语义中相应的属性语义。	是
RTC_CENTER	对象，编号[3]	一个 GlobalPropertyCartesian3 对象，为所有要素定义一个三分量数值属性。参见语义中相应的属性语义。	不

Additional properties are allowed.

允许附加属性。

Type of each property: Property

每个属性的类型:属性

Batched3DModelFeatureTable.extensions

批处理 D3d modelFeatureTable . extensions

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

Batched3DModelFeatureTable.extras

批处理 d3DModelFeatureTable.extras

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Batched3DModelFeatureTable.BATCH_LENGTH

批处理 d3DModelFeatureTable。BATCH_LENGTH

A GlobalPropertyScalar object defining a numeric property for all features.See the corresponding property semantic in Semantics.

一个全局属性标量对象，为所有要素定义一个数值属性。参见语义中相应的属性语义。

Type: object, number [1], number

类型:对象，数字[1]，数字

Required: Yes

必填:是

Batched3DModelFeatureTable.RTC_CENTER

批处理 d3DModelFeatureTable。RTC_CENTER

A GlobalPropertyCartesian3 object defining a 3-component numeric property for all features.See the corresponding property semantic in Semantics.

一个 GlobalPropertyCartesian3 对象，为所有要素定义一个三分量数值属性。参见语义中相应的属性语义。

Type: object, number [3]

类型:对象，编号[3]

Required: No

必填:否

BinaryBodyReference

BinaryBodyReference

An object defining the reference to a section of the binary body of the features table where the property values are stored if not defined directly in the JSON.

一个对象，定义对要素表的二进制主体的一部分的引用，如果没有直接在 JSON 中定义，属性值将存储在该部分中。

Properties

性能

	Type	Description	Required
byteOffset	number	The offset into the buffer in bytes.	Yes

	类型	描述	需要
字节数组	数字	缓冲区的偏移量，以字节为单位。	是

Additional properties are allowed.

允许附加属性。

BinaryBodyReference.byteOffset

binary body reference . byTeofSet

The offset into the buffer in bytes.

缓冲区的偏移量，以字节为单位。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

GlobalPropertyCartesian3

全球财产卡特尔 3

An object defining a global 3-component numeric property values for all features.

为所有要素定义全局三分量数值属性值的对象。

GlobalPropertyScalar

全局属性标量

An object defining a global numeric property values for all features.

为所有要素定义全局数值属性值的对象。

Property

财产

A user-defined property which specifies per-feature application-specific metadata in a tile.Values either can be defined directly in the JSON as an array, or can refer to sections in the binary body with a BinaryBodyReference object.

一种用户定义的属性，用于指定图块中每个要素的特定于应用程序的元数据。值可以直接在 JSON 中定义为数组，也可以用 BinaryBodyReference 对象引用二进制体中的部分。

Instanced 3D Model

实例化三维模型

Overview

概观

Instanced 3D Model is a tile format for efficient streaming and rendering of a large number of models, called instances, with slight variations.In the simplest case, the same tree model, for example, may be located—or instanced—in several places.Each instance references the same model and has per-instance properties, such as position.Using the core 3D Tiles spec language, each instance is a feature.

实例化 3D 模型是一种平铺格式，用于大量模型的高效流式传输和渲染，称为实例，略有变化。例如，在最简单的情况下，相同的树模型可能位于——或实例化于——几个地方。每个实例引用相同的模型，并具有每个实例的属性，例如位置。使用核心 3D 切片规范语言，每个实例都是一个特征。

In addition to trees, Instanced 3D Model is useful for exterior features such as fire hydrants, sewer caps, lamps, and traffic lights, and for interior CAD features such as bolts, valves, and electrical outlets.

除了树之外，实例化三维模型对于外部特征(如消防栓、下水道盖、灯和交通灯)以及内部计算机辅助设计特征(如螺栓、阀门和电源插座)也很有用。

An Instanced 3D Model tile is a binary blob in little endian.

实例化三维模型切片是以小端序排列的二进制斑点。

Implementation Note: A Composite tile can be used to create tiles with different types of instanced models, e.g., trees and traffic lights by combing two Instanced 3D Model tiles.

实施注意:通过组合两个实例化三维模型切片，复合切片可用于创建具有不同实例化模型类型的切片，例如树和交通灯。

Implementation Note: Instanced 3D Model maps well to the ANGLE_instanced_arrays extension for efficient rendering with WebGL.

实现注意:实例化三维模型很好地映射到 ANGLE _实例化_数组扩展，以便使用 WebGL 进行高效渲染。

Layout

布局

A tile is composed of a header section immediately followed by a binary body.The following figure shows the Instanced 3D Model layout (dashes indicate optional fields):

一个图块由一个标题部分和一个二进制主体组成。下图显示了实例化三维模型布局(虚线表示可选字段):
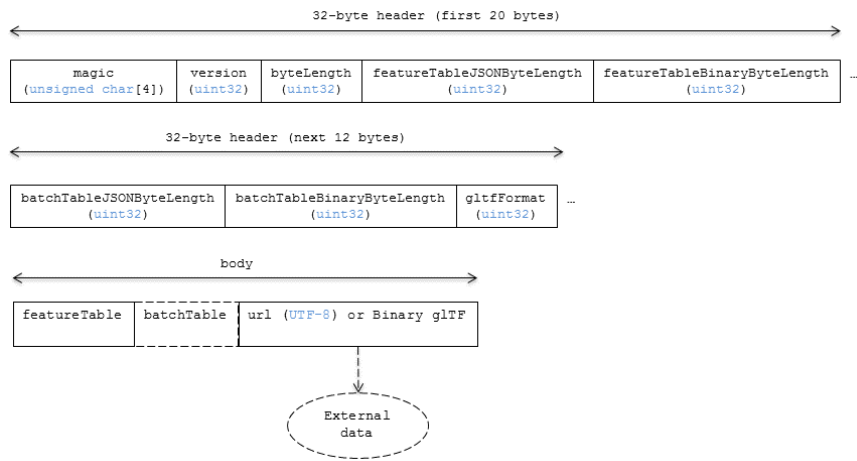
Figure 25: Instanced 3D Model layout

图 25:实例化三维模型布局

Padding

填料

A tile's byteLength must be aligned to an 8-byte boundary.The contained Feature Table and Batch Table must conform to their respective padding requirement.

图块的字节长度必须与 8 字节边界对齐。包含的特征表和批处理表必须符合它们各自的填充要求。

The binary glTF (if present) must start and end on an 8-byte boundary so that glTF's byte-alignment guarantees are met.This can be done by padding the Feature Table or Batch Table if they are present.

二进制 glTF(如果存在)必须在 8 字节的边界上开始和结束，以满足 glTF 的字节对齐保证。这可以通过填充特征表或批处理表(如果存在)来完成。

Otherwise, if the glTF field is a UTF-8 string, it must be padded with trailing Space characters (0x20) to satisfy alignment requirements of the tile, which must be removed at runtime before requesting the glTF asset.

否则，如果 glTF 字段是 UTF-8 字符串，则必须用尾随空格字符(0x20)填充它，以满足切片的对齐要求，在请求 glTF 资产之前，必须在运行时删除这些字符。

Header

页眉

The 32-byte header contains the following fields:

32 字节报头包含以下字段:

Field name	Data type	Description
magic	4-byte ANSI string	"i3dm".This can be used to identify the content as an Instanced 3D Model tile.
version	uint32	The version of the Instanced 3D Model format.It is currently 1.
byteLength	uint32	The length of the entire tile, including the header, in bytes.
featureTableJSONByteLength	uint32	The length of the Feature Table JSON section in bytes.
featureTableBinaryByteLength	uint32	The length of the Feature Table binary section in bytes.
batchTableJSONByteLength	uint32	The length of the Batch Table JSON section in bytes.Zero indicates that there is no Batch Table.
batchTableBinaryByteLength	uint32	The length of the Batch Table binary section in bytes.If batchTableJSONByteLength is zero, this will also be zero.
gltfFormat	uint32	Indicates the format of the glTF field of the body.0 indicates it is a uri, 1 indicates it is embedded binary glTF.See the glTF section below.

字段名	数据类型	描述
魔法	4 字节 ANSI 字符串	"i3dm"。这可用于将内容标识为实例化三维模型切片。
版本	uint32	实例化三维模型格式的版本。目前是 1。
byteLength	uint32	整个图块的长度，包括标头，以字节为单位。
featureTableJSONByteLength	uint32	特征表 JSON 部分的长度，以字节为单位。
功能表二进制字节长度	uint32	功能表二进制部分的长度，以字节为单位。
batchTableJSONByteLength	uint32	批处理表 JSON 部分的长度，以字节为单位。零表示没有批处理表。
batchTableBinaryByteLength	uint32	批处理表二进制部分的长度，以字节为单位。如果 batchTableJSONByteLength 为零，这也将为零。
gltfFormat	uint32	指示正文的 glTF 字段的格式。0 表示它是 uri，1 表示它是嵌入的二进制 glTF。参见下面的 glTF 部分。

The body section immediately follows the header section and is composed of three fields: Feature Table, Batch Table, and glTF.

正文部分紧跟着标题部分，由三个字段组成:特征表、批处理表和 glTF。

Feature Table

功能表

The Feature Table contains values for i3dm semantics used to create instanced models.

要素表包含用于创建实例模型的 i3dm 语义值。

More information is available in the Feature Table specification.

特性表规范中提供了更多信息。

Semantics

语义学

Instance semantics

实例语义

These semantics map to an array of feature values that are used to create instances.The length of these arrays must be the same for all semantics and is equal to the number of instances.

这些语义映射到用于创建实例的特征值数组。对于所有语义，这些数组的长度必须相同，并且等于实例的数量。

The value for each instance semantic must be a reference to the Feature Table binary body;they cannot be embedded in the Feature Table JSON header.

每个实例语义的值必须是对要素表二进制体的引用；它们不能嵌入到要素表 JSON 标题中。

If a semantic has a dependency on another semantic, that semantic must be defined.

如果一个语义依赖于另一个语义，则必须定义该语义。

If both SCALE and SCALE_NON_UNIFORM are defined for an instance, both scaling operations will be applied.

如果为实例定义了"缩放"和"缩放_非均匀"，则两种缩放操作都将被应用。

If both POSITION and POSITION_QUANTIZED are defined for an instance, the higher precision POSITION will be used.

如果为一个实例定义了位置和位置量化，将使用更高精度的位置。

If NORMAL_UP, NORMAL_RIGHT, NORMAL_UP_OCT32P, and NORMAL_RIGHT_OCT32P are defined for an instance, the higher precision NORMAL_UP and NORMAL_RIGHT will be used.

如果为一个实例定义了正常向上、正常向右、正常向上_OCT32P 和正常向右_OCT32P，则将使用更高精度的正常向上和正常向右。

Semantic	Data Type	Description	Required
POSITION	float32[3]	A 3-component array of numbers containing x, y, and z Cartesian coordinates for the position of the instance.	Yes, unless POSITION_QUANTIZED is defined.

POSITION_QUANTIZED	uint16[3]	A 3-component array of numbers containing x, y, and z in quantized Cartesian coordinates for the position of the instance.	Yes, unless POSITION is defined.
NORMAL_UP	float32[3]	A unit vector defining the up direction for the orientation of the instance.	No, unless NORMAL_RIGHT is defined.
NORMAL_RIGHT	float32[3]	A unit vector defining the right direction for the orientation of the instance.Must be orthogonal to up.	No, unless NORMAL_UP is defined.
NORMAL_UP_OCT32P	uint16[2]	An oct-encoded unit vector with 32-bits of precision defining the up direction for the orientation of the instance.	No, unless NORMAL_RIGHT_OCT32P is defined.
NORMAL_RIGHT_OCT32P	uint16[2]	An oct-encoded unit vector with 32-bits of precision defining the right direction for the orientation of the instance.Must be orthogonal to up.	No, unless NORMAL_UP_OCT32P is defined.
SCALE	float32	A number defining a scale to apply to all axes of the instance.	No.
SCALE_NON_UNIFORM	float32[3]	A 3-component array of numbers defining the scale to apply to the x, y, and z axes of the instance.	No.
BATCH_ID	uint8, uint16 (default), or uint32	The batchId of the instance that can be used to retrieve metadata from the Batch Table.	No.

语义的	数据类型	描述	需要
位置	浮动 32[3]	包含实例位置的 x、y 和 z 笛卡尔坐标的三分量数字数组。	是的，除非定义了位置量化。
位置_量化	uint16[3]	实例位置的量化笛卡尔坐标中包含 x、y 和 z 的三分量数字数组。	是的，除非定义了位置。
正常_向上	浮动 32[3]	定义实例方向向上方向的单位向量。	不，除非定义了正常_右。
正常_右侧	浮动 32[3]	定义实例方向正确方向的单位向量。必须垂直向上。	不，除非定义了正常启动。
正常_向上_OCT32P	uint16[2]	具有 32 位精度的 oct 编码单位向量，定义实例方向的向上方向。	不，除非定义了正常_右_OCT32P。
正常_右_八月 32P	uint16[2]	具有 32 位精度的 oct 编码单位向量，定义实例方向的正确方向。必须垂直向上。	不，除非定义了正常_向上_OCT32P。
规模	float32	定义应用于实例所有轴的比例的数字。	没有。
比例_不一致	浮动 32[3]	定义应用于实例的 x、y 和 z 轴的比例的三分量数字数组。	没有。
批处理标识	uint8、uint16(默认)或 uint32	实例的批处理数据，可用于从批处理表中检索元数据。	没有。

Global semantics

全局语义学

These semantics define global properties for all instances.

这些语义定义了所有实例的全局属性。

Semantic	Data Type	Description	Required
INSTANCES_LENGTH	uint32	The number of instances to generate.The length of each array value for an instance semantic should be equal to this.	Yes.
RTC_CENTER	float32[3]	A 3-component array of numbers defining the center position when instance positions are defined relative-to-center.	No.
QUANTIZED_VOLUME_OFFSET	float32[3]	A 3-component array of numbers defining the offset for the quantized volume.	No, unless POSITION_QUANTIZED is defined.
QUANTIZED_VOLUME_SCALE	float32[3]	A 3-component array of numbers defining the scale for the quantized volume.	No, unless POSITION_QUANTIZED is defined.
EAST_NORTH_UP	boolean	When true and per-instance orientation is not defined, each instance will default to the east/north/up reference frame's orientation on the WGS84 ellipsoid.	No.

语义的	数据类型	描述	需要
实例_长度	uint32	要生成的实例数。实例语义的每个数组值的长度应该等于这个长度。	是的。
RTC_CENTER	浮动 32[3]	当实例位置相对于中心定义时，定义中心位置的三分量数字数组。	没有。
量化_音量_偏移量	浮动 32[3]	定义量化体积偏移量的三分量数字数组。	不，除非定义了位置量化。

量化体积标度	浮动 32[3]	定义量化体积标度的三分量数字数组。	不，除非定义了位置量化。
东_北_上	布尔型	如果未定义真方向和每个实例的方向，每个实例将默认为 WGS84 椭球上的东/北/上参考系的方向。	没有。

Examples using these semantics can be found in the examples section.

使用这些语义的示例可以在示例部分找到。

Instance orientation

实例方向

An instance's orientation is defined by an orthonormal basis created by an up and right vector.The orientation will be transformed by the tile transform.

实例的方向由上下向量创建的正交基定义。方向将通过平铺变换进行变换。

The x vector in the standard basis maps to the right vector in the transformed basis, and the y vector maps to the up vector.

标准基中的 x 向量映射到变换基中的右向量，y 向量映射到上向量。

The z vector would map to a forward vector, but it is omitted because it will always be the cross product of right and up.

z 向量将映射到正向向量，但它被省略了，因为它总是右向上的叉积。
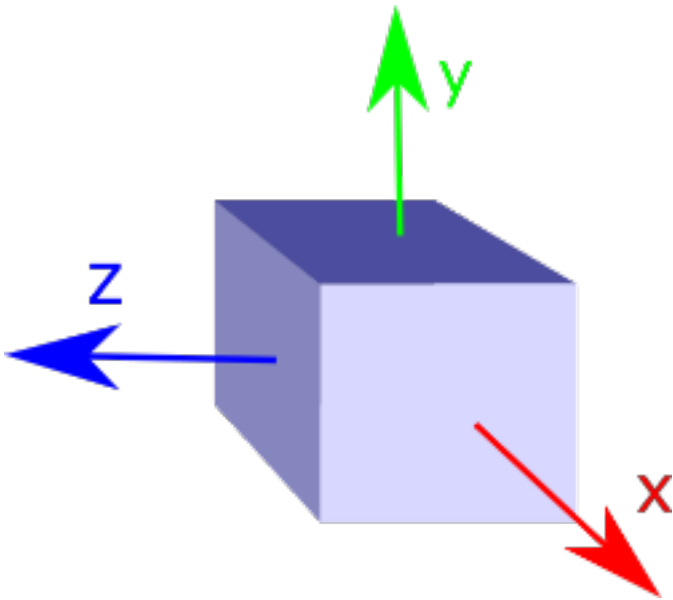
Figure 26: A box in the standard basis

图 26:标准基础中的一个框
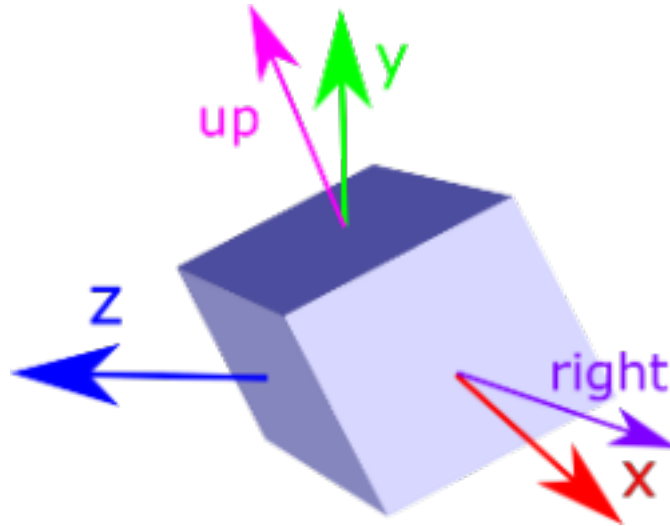
Figure 27: A box transformed into a rotated basis

图 27:一个转换成旋转基础的盒子

Oct-encoded normal vectors

Oct 编码的法向量

If NORMAL_UP and NORMAL_RIGHT are not defined for an instance, its orientation may be stored as oct-encoded normals in NORMAL_UP_OCT32P and NORMAL_RIGHT_OCT32P.

如果没有为实例定义法线向上和法线向右，则其方向可以存储为法线向上 OCT32P 和法线向右 OCT32P 中的 oct 编码法线。

These define up and right using the oct-encoding described in A Survey of Efficient Representations of Independent Unit Vectors.Oct-encoded values are stored in unsigned, unnormalized range ([0, 65535]) and then mapped to a signed normalized range ([-1.0, 1.0]) at runtime.

它们使用独立单位向量有效表示综述中描述的十进制编码进行向上和向右定义。Oct 编码的值存储在无符号、非标准化的范围内([0，65535)，然后在运行时映射到有符号标准化的范围内([-1.0，1.0)。

An implementation for encoding and decoding these unit vectors can be found in Cesium's AttributeCompression

在铯的属性压缩中可以找到编码和解码这些单位向量的实现

module.

模块。

Default orientation

默认方向

If NORMAL_UP and NORMAL_RIGHT or NORMAL_UP_OCT32P and NORMAL_RIGHT_OCT32P are not present, the instance will not have a custom orientation.If EAST_NORTH_UP is true, the instance is assumed to be on the WGS84 ellipsoid and its orientation will default to the east/north/up reference frame at its cartographic position.

如果不存在正常向上和正常向右或正常向上 OCT32P 和正常向右 OCT32P，实例将不会有自定义方向。如果 EAST_NORTH_UP 为真，则假设该实例位于 WGS84 椭球体上，其方向默认为制图位置处的东/北/上参考系。

This is suitable for instanced models such as trees whose orientation is always facing up from their position on the ellipsoid's surface.

这适用于实例模型，如方向总是从椭球面上的位置朝上的树。

Instance position

实例位置

POSITION defines the location for an instance before any tile transforms are applied.

位置在应用任何图块变换之前定义实例的位置。

RTC_CENTER

RTC_CENTER

Positions may be defined relative-to-center for high-precision rendering, see Precisions, Precisions.If defined, RTC_CENTER specifies the center position and all instance positions are treated as relative to this value.

对于高精度渲染，可以相对于中心定义位置，请参见精度、精度。如果定义，RTC_CENTER 指定中心位置，所有实例位置都被视为相对于该值。

Quantized positions

量化位置

If POSITION is not defined for an instance, its position may be stored in POSITION_QUANTIZED, which defines the instance position relative to the quantized volume.

如果没有为实例定义位置，则其位置可以存储在 POSITION_QUANTIZED 中，该位置定义了实例相对于量化体积的位置。

If neither POSITION or POSITION_QUANTIZED are defined, the instance will not be created.

如果没有定义“位置”或“位置_量化”，则不会创建实例。

A quantized volume is defined by offset and scale to map quantized positions into local space, as shown in the following figure:

量化体积通过偏移和缩放来定义，以将量化位置映射到局部空间，如下图所示:
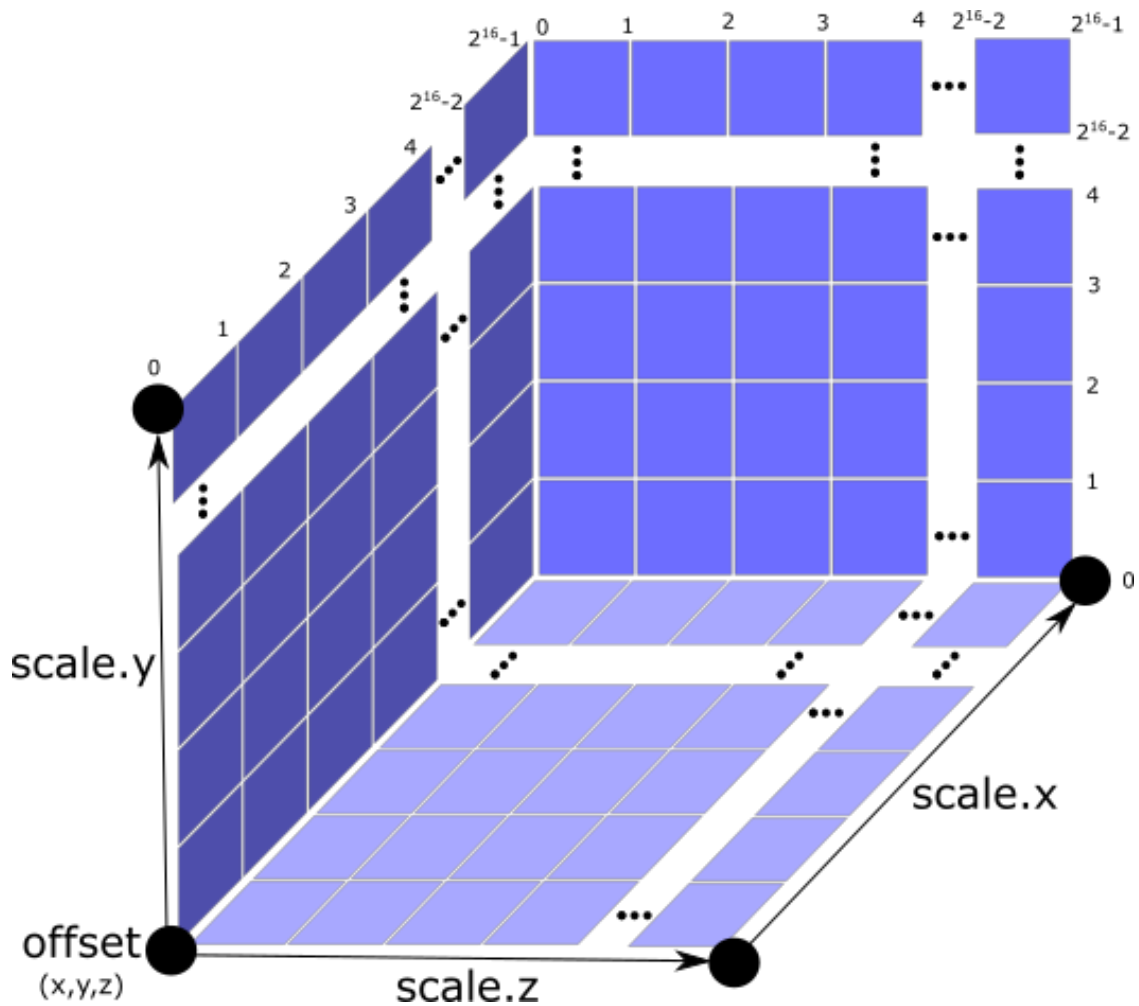
Figure 28: A quantized volume

图 28:量化的体积

offset is stored in the global semantic QUANTIZED_VOLUME_OFFSET, and scale is stored in the global semantic QUANTIZED_VOLUME_SCALE.

偏移量存储在全局语义 QUANTIZED_VOLUME_OFFSET 中，比例存储在全局语义 QUANTIZED _ VOLUME _ SCALE 中。

If those global semantics are not defined, POSITION_QUANTIZED cannot be used.

如果没有定义这些全局语义，就不能使用 POSITION_QUANTIZED。

Quantized positions can be mapped to local space using the following formula:

量化的位置可以使用以下公式映射到局部空间:

POSITION = POSITION_QUANTIZED * QUANTIZED_VOLUME_SCALE / 65535.0 + QUANTIZED_VOLUME_OFFSET

位置=位置_量化*量化_体积_标度/ 65535.0 +量化_体积_偏移

Instance scaling

实例缩放

Scaling can be applied to instances using the SCALE and SCALE_NON_UNIFORM semantics.

缩放可以使用 SCRALE 和 SCRALE _ NON-UNIFIC 语义应用于实例。

SCALE applies a uniform scale along all axes, and SCALE_NON_UNIFORM applies scaling to the x, y, and z axes independently.

"缩放"沿所有轴应用统一的缩放，而"缩放_非统一"独立地将缩放应用于 x、y 和 z 轴。

Examples

例子

These examples show how to generate JSON and binary buffers for the Feature Table.

这些示例显示了如何为特征表生成 JSON 和二进制缓冲区。

Positions only

仅职位

In this minimal example, we place four instances on the corners of a unit length square with the default orientation:

在这个最小的例子中，我们将四个实例放置在具有默认方向的单位长度正方形的角上:

var featureTableJSON = {

var featureTableJSON = {

INSTANCES_LENGTH : 4,

实例_长度:4，

POSITION : {

位置:{

byteOffset : 0

字节数组:0

}

}

};

};

var featureTableBinary = new Buffer(new Float32Array([

可变特征表二进制=新缓冲区(新浮动 32 阵列([

0.0, 0.0, 0.0,

0.0，0.0，0.0，

1.0, 0.0, 0.0,

1.0，0.0，0.0，

0.0, 0.0, 1.0,

0.0，0.0，1.0，

1.0, 0.0, 1.0

1.0，0.0，1.0

]).buffer);

])。缓冲区);

Quantized positions and oct-encoded normals

量化位置和 oct 编码法线

In this example, the four instances will be placed with an orientation up of [0.0, 1.0, 0.0] and right of [1.0, 0.0, 0.0] in oct-encoded format

在本例中，这四个实例将以 oct 编码格式放置，方向为[0.0、1.0、0.0]，[1.0、0.0、0.0]

and they will be placed on the corners of a quantized volume that spans from -250.0 to 250.0 units in the x and z directions:

并且它们将被放置在在 x 和 z 方向上从-250.0 到 250.0 单位的量化体积的角上:

var featureTableJSON = {

var featureTableJSON = {

INSTANCES_LENGTH : 4,

实例_长度:4，

QUANTIZED_VOLUME_OFFSET : [-250.0, 0.0, -250.0],

量化体积偏移量:[-250.0，0.0，-250.0]，

QUANTIZED_VOLUME_SCALE : [500.0, 0.0, 500.0],

量化体积比例:[500.0，0.0，500.0]，

POSITION_QUANTIZED : {

位置_量化:{

byteOffset : 0

字节数组:0

}，

}，

NORMAL_UP_OCT32P : {

正常_向上_OCT32P : {

byteOffset : 24

字节数组:24

}，

}，

NORMAL_RIGHT_OCT32P : {

正常_右_八月 32P : {

byteOffset : 40

字节数组:40

}

}

};

}；

var positionQuantizedBinary = new Buffer(new Uint16Array([

可变位置量化二进制=新缓冲器(新单元 16 阵列([

0, 0, 0,

0，0，0，

```
65535, 0, 0,

65535，0，0，

0, 0, 65535,

0，0，65535，

65535, 0, 65535

65535，0，65535

]).buffer);
```

]).缓冲区);

```
var normalUpOct32PBinary = new Buffer(new Uint16Array([
```

var normalUpOct32PBinary =新缓冲区(新 Uint16Array([

```
32768, 65535,

32768，65535，

32768, 65535,

32768，65535，

32768, 65535,

32768，65535，

32768, 65535

32768，65535

]).buffer);
```

]).缓冲区);

```
var normalRightOct32PBinary = new Buffer(new Uint16Array([
```

var normalRightOct32PBinary =新缓冲区(新 Uint16Array([

```
65535, 32768,

65535，32768，

65535, 32768,

65535，32768，
```

65535, 32768,

65535，32768，

65535, 32768

65535，32768

]).buffer);

])。缓冲区);

var featureTableBinary = Buffer.concat([positionQuantizedBinary, normalUpOct32PBinary, normalRightOct32PBinary]);

var featureTableBinary = buffer . concat([位置量化二进制，正态八进制 32 二进制，正态八进制 32 二进制]);

Batch Table

批处理表

Contains metadata organized by batchId that can be used for declarative styling.See the Batch Table reference for more information.

包含由 batchId 组织的元数据，可用于声明性样式。有关更多信息，请参见批处理表参考。

glTF

glTF

Instanced 3D Model embeds glTF 2.0 containing model geometry and texture information.

实例化三维模型嵌入包含模型几何和纹理信息的 glTF 2.0。

The glTF asset to be instanced is stored after the Feature Table and Batch Table.It may embed all of its geometry, texture, and animations, or it may refer to external sources for some or all of these data.

要实例化的 glTF 资产存储在特征表和批处理表之后。它可以嵌入它的所有几何图形、纹理和动画，或者它可以引用外部源来获取部分或全部这些数据。

header.gltfFormat determines the format of the glTF field

header.gltfFormat 决定 glTF 字段的格式

When the value of header.gltfFormat is 0, the glTF field is a UTF-8 string, which contains a uri of the glTF or binary glTF model content.

当 header.gltfFormat 的值为 0 时，glTF 字段是 UTF-8 字符串，其中包含 glTF 或二进制 glTF 模型内容的 uri。

When the value of header.gltfFormat is 1, the glTF field is a binary blob containing binary glTF.

当 header.gltfFormat 的值为 1 时，glTF 字段是包含二进制 glTF 的二进制 blob。

In either case, header.gltfByteLength contains the length of the glTF field in bytes.

在任一种情况下，header.gltfByteLength 都包含 glTF 字段的长度(以字节为单位)。

Coordinate system

坐标系

By default glTFs use a right handed coordinate system where the y-axis is up.For consistency with the z-up coordinate system of 3D Tiles, glTFs must be transformed at runtime.See glTF transforms for more details.

默认情况下，glTFs 使用 y 轴朝上的右手坐标系。为了与 3D 切片的 z 向上坐标系保持一致，glTFs 必须在运行时进行转换。有关更多详细信息，请参见 glTF 转换。

File extension and MIME type

文件扩展名和 MIME 类型

Instanced 3D models tiles use the .i3dm extension and application/octet-stream MIME type.

实例化的 3D 模型切片使用. i3dm 扩展和应用程序/八位字节流 MIME 类型。

An explicit file extension is optional.Valid implementations may ignore it and identify a content's format by the magic field in its header.

显式文件扩展名是可选的。有效的实现可以忽略它，并通过标题中的魔法字段来识别内容的格式。

Property reference

属性引用

Instanced 3D Model Feature Table

实例化三维模型特征表

A set of Instanced 3D Model semantics that contains values defining the position and appearance properties for instanced models in a tile.

一组实例化三维模型语义，包含定义图块中实例化模型的位置和外观属性的值。

Properties

性能

	Type	Description	Required

extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No
POSITION	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
POSITION_QUANTIZED	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
NORMAL_UP	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
NORMAL_RIGHT	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
NORMAL_UP_OCT32P	object	A BinaryBodyReference object defining the	No

		reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	
NORMAL_RIGHT_OCT32P	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
SCALE	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
SCALE_NON_UNIFORM	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
BATCH_ID	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
INSTANCES_LENGTH	object, number [1], number	A GlobalPropertyScalar	Yes

		object defining a numeric property for all features.See the corresponding property semantic in Semantics.	
QUANTIZED_VOLUME_OFFSET	object, number [3]	A GlobalPropertyCartesian3 object defining a 3-component numeric property for all features.See the corresponding property semantic in Semantics.	No
QUANTIZED_VOLUME_SCALE	object, number [3]	A GlobalPropertyCartesian3 object defining a 3-component numeric property for all features.See the corresponding property semantic in Semantics.	No

	类型	描述	需要
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不
位置	目标	BinaryBodyReference对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
位置_量化	目标	BinaryBodyReference对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
正常_向上	目标	BinaryBodyReference对象，定义对存储属性值的二进制体部分的引	不

		用。参见语义中相应的属性语义。	
正常_右侧	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
正常_向上_OCT32P	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
正常_右_八月 32P	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
规模	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
比例_不一致	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
批处理标识	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
实例_长度	对象，编号[1]，编号	一个全局属性标量对象，为所有要素定义一个数值属性。参见语义中相应的属性语义。	是
量化_音量_偏移量	对象，编号[3]	一个 GlobalPropertyCartesia	不

		n3 对象，为所有要素定义一个三分量数值属性。参见语义中相应的属性语义。	
量化体积标度	对象，编号[3]	一个 GlobalPropertyCartesian3 对象，为所有要素定义一个三分量数值属性。参见语义中相应的属性语义。	不

Additional properties are allowed.

允许附加属性。

Type of each property: Property

每个属性的类型:属性

Instanced3DModelFeatureTable.extensions

实例 3DModelFeatureTable.extensions

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

Instanced3DModelFeatureTable.extras

实例 3DModelFeatureTable.extras

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

Instanced3DModelFeatureTable.POSITION

实例三维模型特征表。位置

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.POSITION_QUANTIZED

实例三维模型特征表。位置_量化

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.NORMAL_UP

实例三维模型特征表。正常_向上

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.NORMAL_RIGHT

实例三维模型特征表。正常_右侧

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.NORMAL_UP_OCT32P

实例三维模型特征表。正常_向上_OCT32P

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.NORMAL_RIGHT_OCT32P

实例三维模型特征表。正常_右_八月 32P

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.SCALE

实例三维模型特征表。规模

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.SCALE_NON_UNIFORM

实例三维模型特征表。比例_不一致

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.BATCH_ID

实例三维模型特征表。批处理标识

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

Instanced3DModelFeatureTable.INSTANCES_LENGTH

实例三维模型特征表。实例_长度

A GlobalPropertyScalar object defining a numeric property for all features.See the corresponding property semantic in Semantics.

一个全局属性标量对象，为所有要素定义一个数值属性。参见语义中相应的属性语义。

Type: object, number [1], number

类型:对象，数字[1]，数字

Required: Yes

必填:是

Instanced3DModelFeatureTable.QUANTIZED_VOLUME_OFFSET

实例三维模型特征表。量化_音量_偏移量

A GlobalPropertyCartesian3 object defining a 3-component numeric property for all features.See the corresponding property semantic in Semantics.

一个 GlobalPropertyCartesian3 对象，为所有要素定义一个三分量数值属性。参见语义中相应的属性语义。

Type: object, number [3]

类型:对象，编号[3]

Required: No

必填:否

Instanced3DModelFeatureTable.QUANTIZED_VOLUME_SCALE

实例三维模型特征表。量化体积标度

A GlobalPropertyCartesian3 object defining a 3-component numeric property for all features.See the corresponding property semantic in Semantics.

一个 GlobalPropertyCartesian3 对象，为所有要素定义一个三分量数值属性。参见语义中相应的属性语义。

Type: object, number [3]

类型:对象，编号[3]

Required: No

必填:否

BinaryBodyReference

BinaryBodyReference

An object defining the reference to a section of the binary body of the features table where the property values are stored if not defined directly in the JSON.

一个对象，定义对要素表的二进制主体的一部分的引用，如果没有直接在 JSON 中定义，属性值将存储在该部分中。

Properties

性能

	Type	Description	Required
byteOffset	number	The offset into the buffer in bytes.	Yes

	类型	描述	需要
字节数组	数字	缓冲区的偏移量，以字节为单位。	是

Additional properties are allowed.

允许附加属性。

BinaryBodyReference.byteOffset

binary body reference . byTeofSet

The offset into the buffer in bytes.

缓冲区的偏移量，以字节为单位。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

GlobalPropertyCartesian3

全球财产卡特尔 3

An object defining a global 3-component numeric property values for all features.

为所有要素定义全局三分量数值属性值的对象。

GlobalPropertyScalar

全局属性标量

An object defining a global numeric property values for all features.

为所有要素定义全局数值属性值的对象。

Property

财产

A user-defined property which specifies per-feature application-specific metadata in a tile.Values either can be defined directly in the JSON as an array, or can refer to sections in the binary body with a BinaryBodyReference object.

一种用户定义的属性，用于指定图块中每个要素的特定于应用程序的元数据。值可以直接在 JSON 中定义为数组，也可以用 BinaryBodyReference 对象引用二进制体中的部分。

Point Cloud

点云

Overview

概观

The Point Cloud tile format enables efficient streaming of massive point clouds for 3D visualization.Each point is defined by a position and by optional properties used to define its appearance, such as color and normal, as well as optional properties that define application-specific metadata.

点云平铺格式支持高效的海量点云流，以实现三维可视化。每个点都由一个位置和用于定义其外观的可选属性(如颜色和法线)以及定义应用程序特定元数据的可选属性来定义。

Using 3D Tiles terminology, each point is a feature.

使用 3D 切片术语，每个点都是一个特征。

A Point Cloud tile is a binary blob in little endian.

点云图块是以小端序排列的二进制斑点。

Layout

布局

A tile is composed of a header section immediately followed by a body section.The following figure shows the Point Cloud layout (dashes indicate optional fields):

瓦片由紧接着主体部分的头部部分组成。下图显示了点云布局(虚线表示可选字段):
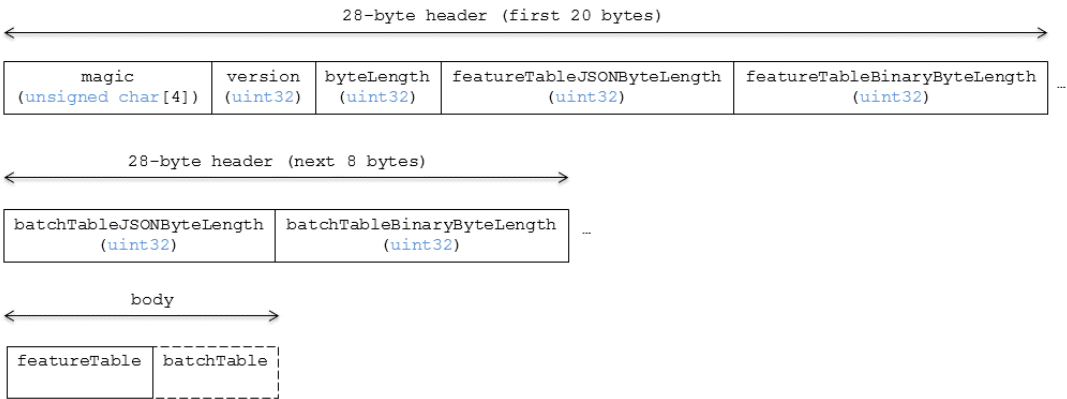
Figure 29: Point Cloud layout

图 29:点云布局

Padding

填料

A tile's byteLength must be aligned to an 8-byte boundary.The contained Feature Table and Batch Table must conform to their respective padding requirement.

图块的字节长度必须与 8 字节边界对齐。包含的特征表和批处理表必须符合它们各自的填充要求。

Header

页眉

The 28-byte header contains the following fields:

28 字节报头包含以下字段:

Field name	Data type	Description
magic	4-byte ANSI string	"pnts".This can be used to identify the content as a Point Cloud tile.
version	uint32	The version of the Point Cloud format.It is currently 1.
byteLength	uint32	The length of the entire tile, including the header, in bytes.
featureTableJSONByteLength	uint32	The length of the Feature Table JSON section in bytes.
featureTableBinaryByteLength	uint32	The length of the Feature Table binary section in bytes.
batchTableJSONByteLength	uint32	The length of the Batch Table JSON section in bytes.Zero indicates that there is no Batch Table.
batchTableBinaryByteLength	uint32	The length of the Batch Table binary section in bytes.If batchTableJSONByteLength is zero, this will also be zero.

字段名	数据类型	描述
魔法	4 字节 ANSI 字符串	"pnts"。这可用于将内容标识为点云图块。
版本	uint32	点云格式的版本。目前是 1。
byteLength	uint32	整个图块的长度，包括标头，以字节为单位。
featureTableJSONByteLength	uint32	特征表 JSON 部分的长度，以字节为单位。
功能表二进制字节长度	uint32	功能表二进制部分的长度，以字节为单位。
batchTableJSONByteLength	uint32	批处理表 JSON 部分的长度，以

		字节为单位。零表示没有批处理表。
batchTableBinaryByteLength	uint32	批处理表二进制部分的长度，以字节为单位。如果 batchTableJSONByteLength 为零，这也将为零。

The body section immediately follows the header section, and is composed of a Feature Table and Batch Table.

正文部分紧跟着标题部分，由特征表和批处理表组成。

Feature Table

功能表

Contains per-tile and per-point values that define where and how to render points.

包含定义渲染点的位置和方式的每图块和每点值。

More information is available in the Feature Table specification.

特性表规范中提供了更多信息。

Semantics

语义学

Point semantics

点语义学

These semantics map to an array of feature values that define each point.The length of these arrays must be the same for all semantics and is equal to the number of points.

这些语义映射到定义每个点的特征值数组。对于所有语义，这些数组的长度必须相同，并且等于点数。

The value for each point semantic must be a reference to the Feature Table binary body;they cannot be embedded in the Feature Table JSON header.

每个点语义的值必须是对要素表二进制体的引用；它们不能嵌入到要素表 JSON 标题中。

If a semantic has a dependency on another semantic, that semantic must be defined.

如果一个语义依赖于另一个语义，则必须定义该语义。

If both POSITION and POSITION_QUANTIZED are defined for a point, the higher precision POSITION will be used.

如果为一个点定义了位置和位置量化，将使用更高精度的位置。

If both NORMAL and NORMAL_OCT16P are defined for a point, the higher precision NORMAL will be used.

如果为一个点定义了法线和法线_OCT16P，将使用更高精度的法线。

Semantic	Data Type	Description	Required
POSITION	float32[3]	A 3-component array of numbers containing x, y, and z Cartesian coordinates for the position of the point.	Yes, unless POSITION_QUANTIZED is defined.
POSITION_QUANTIZED	uint16[3]	A 3-component array of numbers containing x, y, and z in quantized Cartesian coordinates for the position of the point.	Yes, unless POSITION is defined.
RGBA	uint8[4]	A 4-component array of values containing the RGBA color of the point.	No.
RGB	uint8[3]	A 3-component array of values containing the RGB color of the point.	No.
RGB565	uint16	A lossy compressed color format that packs the RGB color into 16 bits, providing 5 bits for red, 6 bits for green, and 5 bits for blue.	No.
NORMAL	float32[3]	A unit vector defining the normal of the point.	No.
NORMAL_OCT16P	uint8[2]	An oct-encoded unit vector with 16 bits of precision defining the normal of the point.	No.
BATCH_ID	uint8, uint16 (default), or uint32	The batchId of the point that can be used to retrieve metadata from the Batch Table.	No.

语义的	数据类型	描述	需要
位置	浮动 32[3]	包含 x、y 和 z 笛卡尔坐标的三分量数字数组。	是的，除非定义了位置量化。
位置_量化	uint16[3]	包含 x、y 和 z 的三分量数字数组，以量化的笛卡尔坐标表示点的位置。	是的，除非定义了位置。
RGBA	uint8[4]	包含点的 RGBA 颜色的四分量值数组。	没有。
RGB	[3]	包含点的 RGB 颜色的三分量值数组。	没有。
RGB565	uint16	一种有损压缩颜色格式，将 RGB 颜色打包成 16 位，提供 5 位红色、6 位绿色和 5 位蓝色。	没有。
标准	浮动 32[3]	定义点法线的单位向量。	没有。
正常_OCT16P	uint8[2]	16 位精度的 oct 编码单位向量，定义点的法线。	没有。
批处理标识	uint8、uint16(默认)或 uint32	可用于从批处理表中检索元数据的点的批处理数据。	没有。

Global semantics

全局语义学

These semantics define global properties for all points.

这些语义定义了所有点的全局属性。

Semantic	Data Type	Description	Required
POINTS_LENGTH	uint32	The number of points to render.The length of	Yes.

		each array value for a point semantic should be equal to this.	
RTC_CENTER	float32[3]	A 3-component array of numbers defining the center position when point positions are defined relative-to-center.	No.
QUANTIZED_VOLU ME_OFFSET	float32[3]	A 3-component array of numbers defining the offset for the quantized volume.	No, unless POSITION_QUANTIZ ED is defined.
QUANTIZED_VOLU ME_SCALE	float32[3]	A 3-component array of numbers defining the scale for the quantized volume.	No, unless POSITION_QUANTIZ ED is defined.
CONSTANT_RGBA	uint8[4]	A 4-component array of values defining a constant RGBA color for all points in the tile.	No.
BATCH_LENGTH	uint32	The number of unique BATCH_ID values.	No, unless BATCH_ID is defined.

语义的	数据类型	描述	需要
点数_长度	uint32	要渲染的点数。点语义的每个数组值的长度应该等于这个长度。	是的。
RTC_CENTER	浮动 32[3]	当相对于中心定义点位置时，定义中心位置的三分量数字数组。	没有。
量化_音量_偏移量	浮动 32[3]	定义量化体积偏移量的三分量数字数组。	不，除非定义了位置量化。
量化体积标度	浮动 32[3]	定义量化体积标度的三分量数字数组。	不，除非定义了位置量化。
常数_RGBA	uint8[4]	为图块中的所有点定义恒定 RGBA 颜色的四分	没有。

		量值数组。	
BATCH_LENGTH	uint32	唯一 BATCH_ID 值的数量。	否，除非定义了 BATCH_ID。

Examples using these semantics can be found in the examples section below.

使用这些语义的例子可以在下面的例子部分找到。

Point positions

点位置

POSITION defines the position for a point before any tileset transforms are applied.

"位置"定义应用任何波浪线变换之前点的位置。

Coordinate reference system (CRS)

坐标参考系统

3D Tiles local coordinate systems use a right-handed 3-axis (x, y, z) Cartesian coordinate system;that is, the cross product of x and y yields z. 3D Tiles defines the z axis as up for local Cartesian coordinate systems (also see coordinate reference system).

3D 图块局部坐标系使用右手 3 轴(x，y，z)笛卡尔坐标系；也就是说，x 和 y 的叉积产生 z。3D 切片将 z 轴定义为局部笛卡尔坐标系(也参见坐标系)。

RTC_CENTER

RTC_CENTER

Positions may be defined relative-to-center for high-precision rendering, see Precisions, Precisions.If defined, RTC_CENTER specifies the center position and all point positions are treated as relative to this value.

对于高精度渲染，可以相对于中心定义位置，请参见精度、精度。如果定义，RTC_CENTER 指定中心位置，所有点位置都被视为相对于该值。

Quantized positions

量化位置

If POSITION is not defined, positions may be stored in POSITION_QUANTIZED, which defines point positions relative to the quantized volume.

如果没有定义位置，位置可以存储在位置量化中，它定义了相对于量化体积的点位置。

If neither POSITION nor POSITION_QUANTIZED is defined, the tile does not need to be rendered.

如果既没有定义位置也没有定义位置量化，则不需要渲染图块。

A quantized volume is defined by offset and scale to map quantized positions to a position in local space.The following figure shows a quantized volume based on offset and scale:

量化体积通过偏移和缩放来定义，以将量化位置映射到局部空间中的位置。下图显示了基于偏移和比例的量化音量:
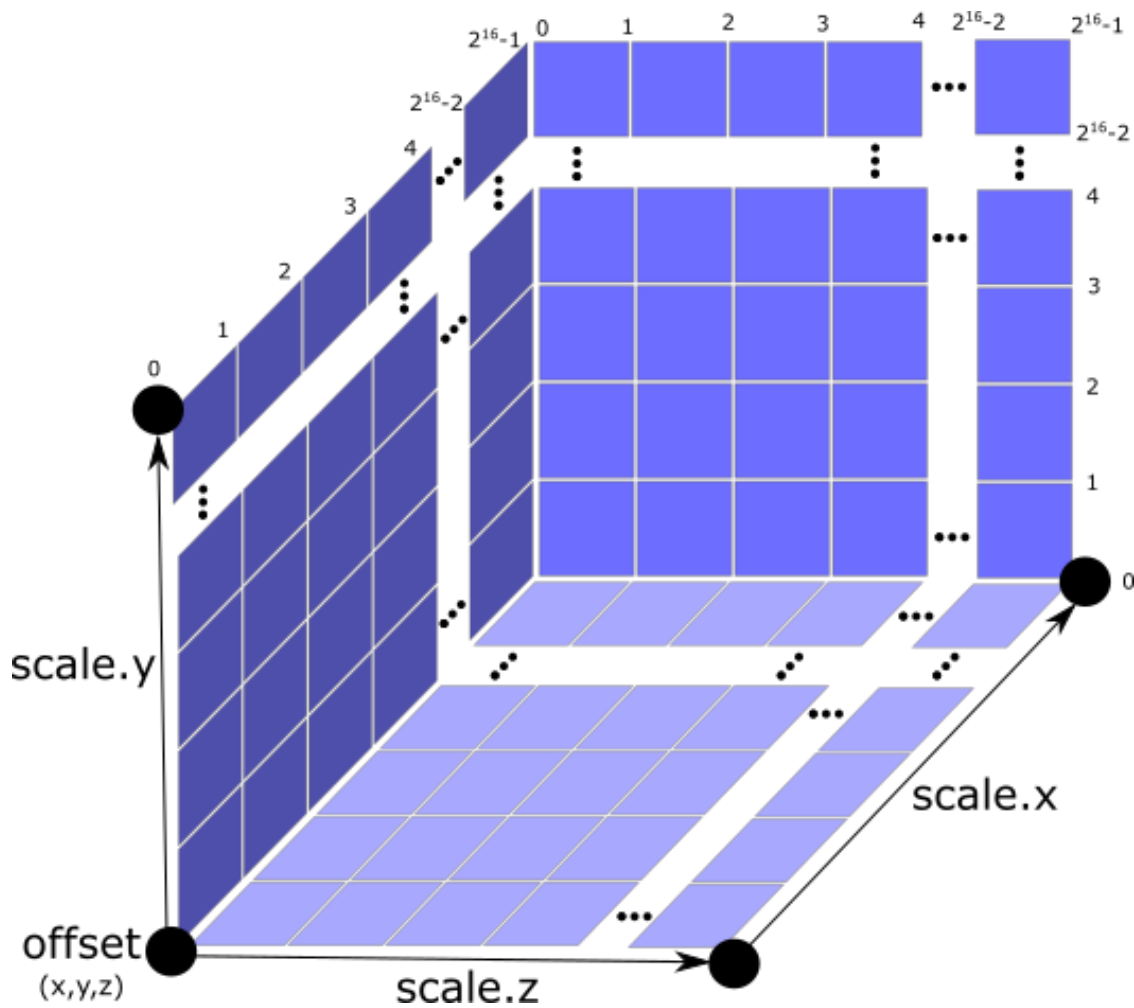
Figure 30: A quantized volume

图 30:量化的体积

offset is stored in the global semantic QUANTIZED_VOLUME_OFFSET, and scale is stored in the global semantic QUANTIZED_VOLUME_SCALE.

偏移量存储在全局语义 QUANTIZED_VOLUME_OFFSET 中，比例存储在全局语义 QUANTIZED _ VOLUME _ SCALE 中。

If those global semantics are not defined, POSITION_QUANTIZED cannot be used.

如果没有定义这些全局语义，就不能使用 POSITION_QUANTIZED。

Quantized positions can be mapped to local space using the following formula:

量化的位置可以使用以下公式映射到局部空间:

POSITION = POSITION_QUANTIZED * QUANTIZED_VOLUME_SCALE / 65535.0 + QUANTIZED_VOLUME_OFFSET

位置=位置_量化*量化_体积_标度/ 65535.0 +量化_体积_偏移

Point colors

点颜色

If more than one color semantic is defined, the precedence order is RGBA, RGB, RGB565, then CONSTANT_RGBA.For example, if a tile's Feature Table contains both RGBA and CONSTANT_RGBA properties, the runtime would render with per-point colors using RGBA.

如果定义了多个颜色语义，优先顺序是 RGBA、RGB、RGB565，然后是常量_RGBA。例如，如果图块的要素表包含 RGBA 和 CONSTANT_RGBA 属性，运行时将使用 RGBA 以每点颜色渲染。

If no color semantics are defined, the runtime is free to color points using an application-specific default color.

如果没有定义颜色语义，运行时可以使用特定于应用程序的默认颜色给点着色。

In any case, 3D Tiles Styling may be used to change the final rendered color and other visual properties at runtime.

在任何情况下，3D 图块样式可用于在运行时更改最终渲染颜色和其他视觉属性。

Point normals

点法线

Per-point normals are an optional property that can help improve the visual quality of points by enabling lighting, hidden surface removal, and other rendering techniques.

每点法线是一个可选属性，可以通过启用照明、隐藏表面移除和其他渲染技术来帮助提高点的视觉质量。

The normals will be transformed using the inverse transpose of the tileset transform.

法线将使用 tileset 变换的逆转置变换。

Oct-encoded normal vectors

Oct 编码的法向量

Oct-encoding is described in A Survey of Efficient Representations of Independent Unit Vectors.Oct-encoded values are stored in unsigned, unnormalized range ([0, 255]) and then mapped to a signed normalized range ([-1.0, 1.0]) at runtime.

独立单位向量的有效表示综述中描述了十进制编码。Oct 编码的值存储在无符号、非标准化的范围([0，255])中，然后在运行时映射到有符号标准化的范围([-1.0，1.0])。

An implementation for encoding and decoding these unit vectors can be found in Cesium's AttributeCompression

在铯的属性压缩中可以找到编码和解码这些单位向量的实现

module.

模块。

Batched points

分批点数

Points that make up distinct features of the Point Cloud may be batched together using the BATCH_ID semantic.For example, the points that make up a door in a house would all be assigned the same BATCH_ID, whereas points that make up a window would be assigned a different BATCH_ID.

构成点云不同特征的点可以使用 BATCH_ID 语义一起批处理。例如，构成房屋门的点将被分配相同的批次标识，而构成窗户的点将被分配不同的批次标识。

This is useful for per-object picking and storing application-specific metadata for declarative styling and application-specific use cases such as populating a UI or issuing a REST API request on a per-object instead of per-point basis.

这对于为声明性样式和特定于应用程序的用例选择和存储特定于应用程序的元数据非常有用，例如填充用户界面或基于每个对象而不是基于每个点发出 REST 应用程序接口请求。

The BATCH_ID semantic may have a componentType of UNSIGNED_BYTE, UNSIGNED_SHORT, or UNSIGNED_INT.When componentType is not present, UNSIGNED_SHORT is used.

BATCH_ID 语义的组件类型可以是 UNSIGNED_BYTE、UNSIGNED_SHORT 或 UNSIGNED_INT。当组件类型不存在时，使用 UNSIGNED_SHORT。

The global semantic BATCH_LENGTH defines the number of unique batchId values, similar to the batchLength field in the Batched 3D Model header.

全局语义 BATCH_LENGTH 定义了唯一的批处理数据值的数量，类似于批处理三维模型头中的批处理长度字段。

Examples

例子

This section is non-normative

本节是非规范性的

These examples show how to generate JSON and binary buffers for the Feature Table.

这些示例显示了如何为特征表生成 JSON 和二进制缓冲区。

Positions only

仅职位

This minimal example has four points on the corners of a unit length square:

这个最小的例子在一个单位长度正方形的角上有四个点:

var featureTableJSON = {

var featureTableJSON = {

POINTS_LENGTH : 4,

点数_长度:4,

POSITION : {

位置:{

byteOffset : 0

字节数组:0

}

}

};

}；

var featureTableBinary = new Buffer(new Float32Array([

可变特征表二进制=新缓冲区(新浮动 32 阵列([

0.0, 0.0, 0.0,

0.0，0.0，0.0,

1.0, 0.0, 0.0,

1.0，0.0，0.0,

0.0, 0.0, 1.0,

0.0，0.0，1.0,

1.0, 0.0, 1.0

1.0，0.0，1.0

]).buffer);

])。缓冲区);

Positions and colors

位置和颜色

The following example has four points (red, green, blue, and yellow) above the globe.Their positions are defined relative to center:

以下示例在地球上方有四个点(红色、绿色、蓝色和黄色)。它们的位置是相对于中心定义的:

var featureTableJSON = {

var featureTableJSON = {

POINTS_LENGTH : 4,

点数_长度:4,

RTC_CENTER : [1215013.8, -4736316.7, 4081608.4],

[1215013.8，-4736316.7，4081608.4],

POSITION : {

位置:{

byteOffset : 0

字节数组:0

},

},

RGB : {

RGB : {

byteOffset : 48

字节数组:48

}

}

};

}；

```
var positionBinary = new Buffer(new Float32Array([
```

可变位置二进制=新缓冲器(新浮动 32 阵列([

```
0.0, 0.0, 0.0,
```

0.0，0.0，0.0，

```
1.0, 0.0, 0.0,
```

1.0，0.0，0.0，

```
0.0, 0.0, 1.0,
```

0.0，0.0，1.0，

```
1.0, 0.0, 1.0
```

1.0，0.0，1.0

```
]).buffer);
```

])。缓冲区);

```
var colorBinary = new Buffer(new Uint8Array([
```

可变颜色二进制=新缓冲区(新 Uint8 阵列([

```
255, 0, 0,
```

255，0，0，

```
0, 255, 0,
```

0，255，0，

```
0, 0, 255,
```

0，0，255，

```
255, 255, 0,
```

255，255，0，

```
]).buffer);
```

])。缓冲区);

```
var featureTableBinary = Buffer.concat([positionBinary, colorBinary]);
```

var FeatureTableBinary = buffer . concat([位置二进制，颜色二进制]);

Quantized positions and oct-encoded normals

量化位置和 oct 编码法线

In this example, the four points will have normals pointing up [0.0, 1.0, 0.0] in oct-encoded format, and they will be placed on the corners of a quantized volume that spans from -250.0 to 250.0 units in the x and z directions:

在本例中，四个点的法线指向 oct 编码格式的[0.0、1.0、0.0]，它们将被放置在 x 和 z 方向上从-250.0 到 250.0 单位的量化体积的角上:

var featureTableJSON = {

var featureTableJSON = {

POINTS_LENGTH : 4,

点数_长度:4，

QUANTIZED_VOLUME_OFFSET : [-250.0, 0.0, -250.0],

量化体积偏移量:[-250.0，0.0，-250.0]，

QUANTIZED_VOLUME_SCALE : [500.0, 0.0, 500.0],

量化体积比例:[500.0，0.0，500.0]，

POSITION_QUANTIZED : {

位置_量化:{

byteOffset : 0

字节数组:0

},

},

NORMAL_OCT16P : {

正常_OCT16P : {

byteOffset : 24

字节数组:24

}

}

};

```
};

var positionQuantizedBinary = new Buffer(new Uint16Array([
```

可变位置量化二进制=新缓冲器(新单元 16 阵列([

```
0, 0, 0,
```

0，0，0，

```
65535, 0, 0,
```

65535，0，0，

```
0, 0, 65535,
```

0，0，65535，

```
65535, 0, 65535
```

65535，0，65535

```
]).buffer);
```

])。缓冲区);

```
var normalOct16PBinary = new Buffer(new Uint8Array([
```

var normalOct16PBinary 二进制=新缓冲区(新 uint 8 阵列([

```
128, 255,
```

128，255，

```
128, 255,
```

128，255，

```
128, 255,
```

128，255，

```
128, 255
```

128，255

```
]).buffer);
```

])。缓冲区);

```
var featureTableBinary = Buffer.concat([positionQuantizedBinary, normalOct16PBinary]);
```

var featureTableBinary = buffer . concat([位置量化二进制，常态八进制));

Batched points

分批点数

In this example, the first two points have a batchId of 0, and the next two points have a batchId of 1.Note that the Batch Table only has two names:

在本例中，前两点的 batchId 为 0，后两点的 batchId 为 1。请注意，批处理表只有两个名称:

var featureTableJSON = {

var featureTableJSON = {

POINTS_LENGTH : 4,

点数_长度:4，

BATCH_LENGTH : 2,

批量长度:2，

POSITION : {

位置:{

byteOffset : 0

字节数组:0

},

},

BATCH_ID : {

批处理_标识:{

byteOffset : 48,

字节数组:48，

componentType : "UNSIGNED_BYTE"

组件类型:"UNSIGNED_BYTE"

}

}

};

```
};
var positionBinary = new Buffer(new Float32Array([
```

可变位置二进制=新缓冲器(新浮动 32 阵列([

```
0.0, 0.0, 0.0,
```

0.0，0.0，0.0，

```
1.0, 0.0, 0.0,
```

1.0，0.0，0.0，

```
0.0, 0.0, 1.0,
```

0.0，0.0，1.0，

```
1.0, 0.0, 1.0
```

1.0，0.0，1.0

```
]).buffer);
```

])。缓冲区);

```
var batchIdBinary = new Buffer(new Uint8Array([
```

var batchIdBinary =新缓冲区(新 Uint8 阵列([

```
0,
```

```
0,
```

```
0,
```

```
0,
```

```
1,
```

1、

```
  1
```

```
1
```

```
]).buffer);
```

])。缓冲区);

```
var featureTableBinary = Buffer.concat([positionBinary, batchIdBinary]);
```

var FeatureTableBinary = buffer . concat([位置二进制，batchidBinary])；

var batchTableJSON = {

var batchTableJSON = {

names : ['object1', 'object2']

名称:['object1 '，' object2']

};

}；

Per-point properties

每点属性

In this example, each of the 4 points will have metadata stored in the Batch Table JSON and binary.

在本例中，4 个点中的每一个都将元数据存储在批处理表 JSON 和二进制文件中。

var featureTableJSON = {

var featureTableJSON = {

POINTS_LENGTH : 4,

点数_长度:4，

POSITION : {

位置:{

byteOffset : 0

字节数组:0

}

}

};

}；

var featureTableBinary = new Buffer(new Float32Array([

可变特征表二进制=新缓冲区(新浮动 32 阵列([

0.0, 0.0, 0.0,

0.0，0.0，0.0，

1.0, 0.0, 0.0,

1.0，0.0，0.0，

0.0, 0.0, 1.0,

0.0，0.0，1.0，

1.0, 0.0, 1.0

1.0，0.0，1.0

]).buffer);

])。缓冲区);

var batchTableJSON = {

names : ['point1', 'point2', 'point3', 'point4']

名称:["点 1"、"点 2"、"点 3"、"点 4"]

};

}；

Batch Table

批处理表

The Batch Table contains application-specific metadata, indexable by batchId, that can be used for declarative styling and application-specific use cases such as populating a UI or issuing a REST API request.

批处理表包含特定于应用程序的元数据，可由批处理索引，可用于声明性样式和特定于应用程序的用例，如填充用户界面或发出 REST 应用程序接口请求。

If the BATCH_ID semantic is defined, the Batch Table stores metadata for each batchId, and the length of the Batch Table arrays will equal BATCH_LENGTH.

如果定义了 BATCH_ID 语义，批处理表存储每个批处理的元数据，批处理表数组的长度将等于 BATCH_LENGTH。

If the BATCH_ID semantic is not defined, then the Batch Table stores per-point metadata, and the length of the Batch Table arrays will equal POINTS_LENGTH.

如果没有定义 BATCH_ID 语义，那么批处理表存储每个点的元数据，并且批处理表数组的长度将等于 POINTS_LENGTH。

See the Batch Table reference for more information.

有关更多信息，请参见批处理表参考。

File extension and MIME type

文件扩展名和 MIME 类型

Point cloud tiles use the .pnts extension and application/octet-stream MIME type.

点云图块使用。pnts 扩展和应用程序/八位字节流 MIME 类型。

An explicit file extension is optional.Valid implementations may ignore it and identify a content's format by the magic field in its header.

显式文件扩展名是可选的。有效的实现可以忽略它，并通过标题中的魔法字段来识别内容的格式。

Implementation example

实现示例

This section is non-normative

本节是非规范性的

Code for reading the header can be found in PointCloud3DModelTileContent.js in the Cesium implementation of 3D Tiles.

读取头的代码可以在三维瓦片铯实现中的点云三维模型中找到。

Property reference

属性引用

Point Cloud Feature Table

点云要素表

A set of Point Cloud semantics that contains values defining the position and appearance properties for points in a tile.

一组点云语义，包含定义图块中点的位置和外观属性的值。

Properties

性能

	Type	Description	Required

extensions	object	Dictionary object with extension-specific objects.	No
extras	any	Application-specific data.	No
POSITION	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
POSITION_QUANTIZED	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
RGBA	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
RGB	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
RGB565	object	A	No

		BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	
NORMAL	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
NORMAL_OCT16P	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
BATCH_ID	object	A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.	No
POINTS_LENGTH	object, number [1], number	A GlobalPropertyScalar object defining a numeric property for all points.See the corresponding property semantic in Semantics.	Yes
RTC_CENTER	object, number [3]	A GlobalPropertyCartesian3 object defining a 3-	No

		component numeric property for all points.See the corresponding property semantic in Semantics.	
QUANTIZED_VOLUME_OFFSET	object, number [3]	A GlobalPropertyCartesian3 object defining a 3-component numeric property for all points.See the corresponding property semantic in Semantics.	No
QUANTIZED_VOLUME_SCALE	object, number [3]	A GlobalPropertyCartesian3 object defining a 3-component numeric property for all points.See the corresponding property semantic in Semantics.	No
CONSTANT_RGBA	object, number [4]	A GlobalPropertyCartesian4 object defining a 4-component numeric property for all points.See the corresponding property semantic in Semantics.	No
BATCH_LENGTH	object, number [1], number	A GlobalPropertyScalar object defining a numeric property for all points.See the corresponding property semantic in Semantics.	No

	类型	描述	需要
延长	目标	具有特定扩展对象的字典对象。	不
额外费用	任何的	特定于应用程序的数据。	不

位置	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
位置_量化	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
RGBA	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
RGB	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
RGB565	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
标准	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
正常_OCT16P	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。	不
批处理标识	目标	BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的	不

		属性语义。	
点数_长度	对象，编号[1]，编号	一个全局属性标量对象，为所有点定义一个数值属性。参见语义中相应的属性语义。	是
RTC_CENTER	对象，编号[3]	一个全局属性卡特尔 n3 对象，为所有点定义一个三分量数值属性。参见语义中相应的属性语义。	不
量化_音量_偏移量	对象，编号[3]	一个全局属性卡特尔 n3 对象，为所有点定义一个三分量数值属性。参见语义中相应的属性语义。	不
量化体积标度	对象，编号[3]	一个全局属性卡特尔 n3 对象，为所有点定义一个三分量数值属性。参见语义中相应的属性语义。	不
常数_RGBA	对象，编号[4]	一个全局属性卡特尔 4 对象，为所有点定义一个 4 分量数值属性。参见语义中相应的属性语义。	不
BATCH_LENGTH	对象，编号[1]，编号	一个全局属性标量对象，为所有点定义一个数值属性。参见语义中相应的属性语义。	不

Additional properties are allowed.

允许附加属性。

Type of each property: Property

每个属性的类型:属性

PointCloudFeatureTable.extensions

pointCloudFeaturetable . extensions

Dictionary object with extension-specific objects.

具有特定扩展对象的字典对象。

Type: object

类型:对象

Required: No

必填:否

Type of each property: Extension

每个属性的类型:扩展

PointCloudFeatureTable.extras

PointCloudFeatureTable.extras

Application-specific data.

特定于应用程序的数据。

Type: any

类型:任何

Required: No

必填:否

PointCloudFeatureTable.POSITION

PointCloudFeatureTable。位置

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.POSITION_QUANTIZED

PointCloudFeatureTable。位置_量化

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.RGBA

PointCloudFeatureTable。RGBA

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.RGB

PointCloudFeatureTable。RGB

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.RGB565

PointCloudFeatureTable。RGB565

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.NORMAL

PointCloudFeatureTable。标准

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.NORMAL_OCT16P

PointCloudFeatureTable。正常_OCT16P

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.BATCH_ID

PointCloudFeatureTable。批处理标识

A BinaryBodyReference object defining the reference to a section of the binary body where the property values are stored.See the corresponding property semantic in Semantics.

BinaryBodyReference 对象，定义对存储属性值的二进制体部分的引用。参见语义中相应的属性语义。

Type: object

类型:对象

Required: No

必填:否

PointCloudFeatureTable.POINTS_LENGTH

PointCloudFeatureTable。点数_长度

A GlobalPropertyScalar object defining a numeric property for all points.See the corresponding property semantic in Semantics.

一个全局属性标量对象，为所有点定义一个数值属性。参见语义中相应的属性语义。

Type: object, number [1], number

类型:对象，数字[1]，数字

Required: Yes

必填:是

PointCloudFeatureTable.RTC_CENTER

PointCloudFeatureTable。RTC_CENTER

A GlobalPropertyCartesian3 object defining a 3-component numeric property for all points.See the corresponding property semantic in Semantics.

一个全局属性卡特尔 n3 对象，为所有点定义一个三分量数值属性。参见语义中相应的属性语义。

Type: object, number [3]

类型:对象，编号[3]

Required: No

必填:否

PointCloudFeatureTable.QUANTIZED_VOLUME_OFFSET

PointCloudFeatureTable。量化_音量_偏移量

A GlobalPropertyCartesian3 object defining a 3-component numeric property for all points.See the corresponding property semantic in Semantics.

一个全局属性卡特尔 n3 对象，为所有点定义一个三分量数值属性。参见语义中相应的属性语义。

Type: object, number [3]

类型:对象，编号[3]

Required: No

必填:否

PointCloudFeatureTable.QUANTIZED_VOLUME_SCALE

PointCloudFeatureTable。量化体积标度

A GlobalPropertyCartesian3 object defining a 3-component numeric property for all points.See the corresponding property semantic in Semantics.

一个全局属性卡特尔 n3 对象，为所有点定义一个三分量数值属性。参见语义中相应的属性语义。

Type: object, number [3]

类型:对象，编号[3]

Required: No

必填:否

PointCloudFeatureTable.CONSTANT_RGBA

PointCloudFeatureTable。常数_RGBA

A GlobalPropertyCartesian4 object defining a 4-component numeric property for all points.See the corresponding property semantic in Semantics.

一个全局属性卡特尔 4 对象，为所有点定义一个 4 分量数值属性。参见语义中相应的属性语义。

Type: object, number [4]

类型:对象，编号[4]

Required: No

必填:否

PointCloudFeatureTable.BATCH_LENGTH

PointCloudFeatureTable。BATCH_LENGTH

A GlobalPropertyScalar object defining a numeric property for all points.See the corresponding property semantic in Semantics.

一个全局属性标量对象，为所有点定义一个数值属性。参见语义中相应的属性语义。

Type: object, number [1], number

类型:对象，数字[1]，数字

Required: No

必填:否

BinaryBodyReference

BinaryBodyReference

An object defining the reference to a section of the binary body of the features table where the property values are stored if not defined directly in the JSON.

一个对象，定义对要素表的二进制主体的一部分的引用，如果没有直接在 JSON 中定义，属性值将存储在该部分中。

Properties

性能

	Type	Description	Required
byteOffset	number	The offset into the buffer in bytes.	Yes

	类型	描述	需要
字节数组	数字	缓冲区的偏移量，以字节为单位。	是

Additional properties are allowed.

允许附加属性。

BinaryBodyReference.byteOffset

binary body reference . byTeofSet

The offset into the buffer in bytes.

缓冲区的偏移量，以字节为单位。

Type: number

类型:数字

Required: Yes

必填:是

Minimum: >= 0

最小值:> = 0

GlobalPropertyCartesian3

全球财产卡特尔 3

An object defining a global 3-component numeric property values for all features.

为所有要素定义全局三分量数值属性值的对象。

GlobalPropertyCartesian4

全球财产卡特尔 4

An object defining a global 4-component numeric property values for all features.

为所有要素定义全局四分量数值属性值的对象。

GlobalPropertyScalar

全局属性标量

An object defining a global numeric property values for all features.

为所有要素定义全局数值属性值的对象。

Property

财产

A user-defined property which specifies per-feature application-specific metadata in a tile.Values either can be defined directly in the JSON as an array, or can refer to sections in the binary body with a BinaryBodyReference object.

一种用户定义的属性，用于指定图块中每个要素的特定于应用程序的元数据。值可以直接在 JSON 中定义为数组，也可以用 BinaryBodyReference 对象引用二进制体中的部分。

Composite

复合材料

Overview

概观

The Composite tile format enables concatenating tiles of different formats into one tile.

复合图块格式可将不同格式的图块连接成一个图块。

3D Tiles and the Composite tile allow flexibility for streaming heterogeneous datasets.For example, buildings and trees could be stored either in two separate Batched 3D Model and Instanced 3D Model tiles or, using a Composite tile, the tiles can be combined.

3D 切片和复合切片允许灵活地流式传输异构数据集。例如，建筑物和树可以存储在两个单独的成批 3D 模型和实例化 3D 模型切片中，或者使用复合切片，可以组合切片。

Supporting heterogeneous datasets with both inter-tile (separate tiles of different formats that are in the same tileset) and intra-tile (different tile formats that are in the same Composite tile) options allows conversion tools to make trade-offs between number of requests, optimal type-specific subdivision, and how visible/hidden layers are streamed.

支持具有块间(同一瓦片集中不同格式的单独瓦片)和块内(同一合成瓦片中不同瓦片格式)选项的异构数据集允许转换工具在请求数量、最佳类型特定细分以及可见/隐藏层的流式传输方式之间进行权衡。

A Composite tile is a binary blob in little endian.

复合图块是以小端序排列的二进制斑点。

Layout

布局

Composite layout (dashes indicate optional fields):

复合布局(虚线表示可选字段):
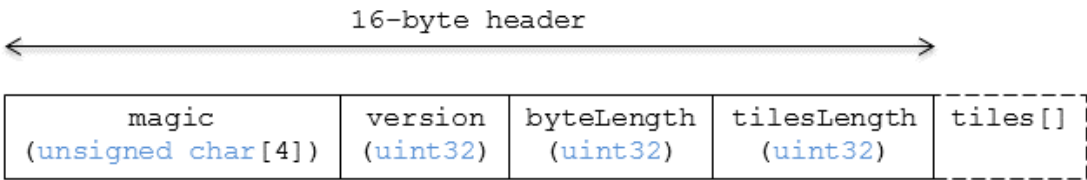
Figure 31: Composite layout

图 31:复合布局

Padding

填料

A tile's byteLength must be aligned to an 8-byte boundary.All tiles contained in a composite tile must also be aligned to an 8-byte boundary.

图块的字节长度必须与 8 字节边界对齐。复合图块中包含的所有图块也必须与 8 字节边界对齐。

Header

页眉

The 16-byte header section contains the following fields:

16 字节标题部分包含以下字段:

Field name	Data type	Description
magic	4-byte ANSI string	"cmpt".This can be used to identify the content as a Composite tile.
version	uint32	The version of the Composite format.It is currently 1.
byteLength	uint32	The length of the entire Composite tile, including this header and each inner tile, in bytes.
tilesLength	uint32	The number of tiles in the Composite.

字段名	数据类型	描述
魔法	4 字节 ANSI 字符串	"cmpt"。这可用于将内容标识为复合图块。
版本	uint32	复合格式的版本。目前是 1。
byteLength	uint32	整个复合图块的长度，包括此标头和每个内部图块，以字节为单位。
tilesLength	uint32	复合中的图块数量。

Inner tiles

内部瓷砖

Inner tile fields are stored tightly packed immediately following the header section.The following information describes general characteristics of all tile formats that a Composite tile reader might exploit to find the boundaries of the inner tiles:

内部磁贴字段紧挨着标题部分存储。以下信息描述了复合图块读取器可以用来查找内部图块边界的所有图块格式的一般特征：

Each tile starts with a 4-byte ANSI string, magic, that can be used to determine the tile format for further parsing.See tile format specifications for a list of possible formats.Composite tiles can contain Composite tiles.

每个图块都以一个 4 字节的 ANSI 字符串 magic 开始，该字符串可用于确定图块格式以供进一步解析。有关可能的格式列表，请参见平铺格式规范。复合瓷砖可以包含复合瓷砖。

Each tile's header contains a uint32 byteLength, which defines the length of the inner tile, including its header, in bytes.This can be used to traverse the inner tiles.

每个图块的标头包含一个 uint32 byteLength，它以字节为单位定义内部图块的长度，包括其标头。这可以用来遍历内部瓦片。

For any tile format's version 1, the first 12 bytes of all tiles is the following fields:

对于任何图块格式的版本 1，所有图块的前 12 个字节是以下字段:

Field name	Data type	Description
magic	4-byte ANSI string	Indicates the tile format
version	uint32	1
byteLength	uint32	Length, in bytes, of the entire tile.

字段名	数据类型	描述
魔法	4 字节 ANSI 字符串	指示平铺格式
版本	uint32	1
byteLength	uint32	整个图块的长度，以字节为单位。

Refer to the spec for each tile format for more details.

有关更多详细信息，请参考每个图块格式的规范。

File extension and MIME type

文件扩展名和 MIME 类型

Composite tiles use the .cmpt extension and application/octet-stream MIME type.

复合瓷砖使用。cmpt 扩展和应用程序/八位字节流 MIME 类型。

An explicit file extension is optional.Valid implementations may ignore it and identify a content's format by the magic field in its header.

显式文件扩展名是可选的。有效的实现可以忽略它，并通过标题中的魔法字段来识别内容的格式。

Implementation examples

实现示例

This section is non-normative

本节是非规范性的

Python packcmpt tool in gltf2glb toolset contains code for combining one or more Batched 3D Model or Instanced 3D Model tiles into a single Composite tile file.

gltf2glb 工具集中的 Python packcmpt 工具包含用于将一个或多个批处理 3D 模型或实例化 3D 模型切片组合到单个复合切片文件中的代码。

Code for reading the header can be found in

读取标题的代码可以在中找到

Composite3DTileContent.js

Composite3DTileContent.js

in the Cesium implementation of 3D Tiles.

在三维瓦片的铯实施中。

Declarative styling specification

声明式样式规范

Overview

概观

3D Tiles styles provide concise declarative styling of tileset features.A style defines expressions to evaluate the display of a feature, for example color (RGB and translucency) and show properties, often based on the feature's properties stored in the tile's Batch Table.

3D 平铺样式提供平铺特征的简洁声明性样式。样式定义表达式来评估要素的显示，例如颜色(RGB 和半透明)和显示属性，通常基于存储在图块批处理表中的要素属性。

A style may be applied to a tile that doesn't contain features, in which case the tile is treated as an implicit single feature without properties.

样式可以应用于不包含要素的图块，在这种情况下，图块被视为没有属性的隐式单个要素。

While a style may be created for and reference properties of a tileset, a style is independent of a tileset, such that any style can be applied to any tileset.

虽然可以为波浪线创建样式并引用波浪线的属性，但是样式独立于波浪线，因此任何样式都可以应用于任何波浪线。

Styles are defined with JSON and expressions written in a small subset of JavaScript augmented for styling.Additionally, the styling language provides a set of built-in functions to support common math operations.

样式是用 JSON 定义的，表达式是用 JavaScript 的一个小子集编写的，为样式而扩展。此外，样式语言提供了一组内置函数来支持常见的数学运算。

The following example assigns a color based on building height.

以下示例基于建筑高度指定颜色。

{

{

"show" : "${Area} > 0",

"显示":$ {面积} > 0 "，

"color" : {

"颜色":{

"conditions" : [

"条件":[

["${Height} < 60", "color('#13293D')"],

["$ { Height } < 60"，"color(# 13293D")"]，

["${Height} < 120", "color('#1B98E0')"],

["$ { Height } < 120"，"color(# 1B 98E 0")"]，

["true", "color('#E8F1F2', 0.5)"]

[“真”、“颜色(# E8f1 F2’，0.5)”]

]

]

}

}

}

}

Concepts

概念

Styling features

造型特征

Visual properties available for styling features are the show property, the assigned expression of which will evaluate to a boolean that determines if the feature is visible, and the color property, the assigned expression of which will evaluate to a Color object (RGB and translucency) which determines the displayed color of a feature.

可用于造型特征的视觉属性是 show 属性，其指定表达式将计算为布尔值，确定特征是否可见；颜色属性，其指定表达式将计算为颜色对象(RGB 和半透明)，确定特征的显示颜色。

The following style assigns the default show and color properties to each feature:

以下样式为每个特征指定默认的显示和颜色属性:

{

{

"show" : "true",

“显示”:“真实”，

"color" : "color('#ffffff')"

"颜色":"颜色(#ffffff)" "

}

}

Instead of showing all features, show can be an expression dependent on a feature's properties, for example, the following expression will show only features in the 19341 zip code:

show 可以是一个依赖于要素属性的表达式，而不是显示所有要素，例如，以下表达式将只显示 19341 邮政编码中的要素:

{

{

"show" : "${ZipCode} === '19341'"

" show ":$ { ZipCOde } = = = ' 19341 ' "

}

}

show can also be used for more complex queries;for example, here a compound condition and regular expression are used to show only features whose county starts with 'Chest' and whose year built is greater than or equal to 1970:

show 还可以用于更复杂的查询；例如，这里复合条件和正则表达式仅用于显示其县以"胸部"开头且其年份大于或等于 1970 年的特征:

{

{

"show" : "(regExp('^Chest').test(${County})) && (${YearBuilt} >= 1970)"

《秀》:(regExp('^Chest').测试($ {县})& &($ {年建成} >= 1970)"

}

}

Colors can also be defined by expressions dependent on a feature's properties.For example, the following expression colors features with a temperature above 90 as red and the others as white:

颜色也可以由依赖于特征属性的表达式定义。例如，以下表达式以红色表示温度高于 90 度的特征，以白色表示其他特征:

{

{

"color" : "(${Temperature} > 90) ?color('red') : color('white')"

"颜色":($ {温度} > 90)? 颜色("红色"):颜色("白色")

}

}

The color's alpha component defines the feature's opacity.For example, the following sets the feature's RGB color components from the feature's properties and makes features with volume greater than 100 transparent:

颜色的 alpha 分量定义了特征的不透明度。例如，以下内容根据要素的属性设置要素的 RGB 颜色分量，并使体积大于 100 的要素透明:

{

{

"color" : "rgba(${red}, ${green}, ${blue}, (${volume} > 100 ?0.5 : 1.0))"

"颜色":" rgba($ {红色}、$ {绿色}、$ {蓝色}、($ {音量} > 100？0.5 : 1.0))

}

}

Conditions

情况

In addition to a string containing an expression, color and show can be an array defining a series of conditions (similar to if...else statements).Conditions can, for example, be used to make color maps and color ramps with any type of inclusive/exclusive intervals.

除了包含表达式的字符串之外，颜色和显示可以是定义一系列条件的数组(类似于 if...其他声明)。例如，条件可以用于制作具有任何类型的包含/排除间隔的彩色地图和彩色斜坡。

For example, the following expression maps an ID property to colors.Conditions are evaluated in order, so if ${id} is not '1' or '2', the "true" condition returns white.If no conditions are met, the color of the feature will be undefined:

例如，以下表达式将标识属性映射到颜色。条件按顺序计算，因此如果${id}不是"1"或"2"，则"true"条件返回白色。如果不满足任何条件，特征的颜色将是未定义的:

{

{

"color" : {

"颜色":{

"conditions" : [

"条件":[

["${id} === '1'", "color('#FF0000')"],

["${id} === '1 ' "，"颜色(#FF0000')"]，

```
["${id} === '2'", "color('#00FF00')"],
```

["${id} === '2 ' "，"颜色(#00FF00')"],

```
["true", "color('#FFFFFF')"]
```

["真"、"颜色(# FFFFFF")"]

```
]
```

```
]
```

```
}
```

```
}
```

```
}
```

```
}
```

The next example shows how to use conditions to create a color ramp using intervals with an inclusive lower bound and exclusive upper bound:

下一个示例显示了如何使用条件创建一个使用区间的色带，区间具有包含下限和排除上限:

```
"color" : {
```

"颜色":{

```
"conditions" : [
```

"条件":[

```
["(${Height} >= 1.0) && (${Height} < 10.0)", "color('#FF00FF')"],
```

["($ { Height } > = 1.0)&($ { Height } < 10.0)"，"颜色(# FF00FF ')"],

```
["(${Height} >= 10.0) && (${Height} < 30.0)", "color('#FF0000')"],
```

["($ { Height } > = 10.0)&($ { Height } < 30.0)"，"颜色(# ff 0000 ')"],

```
["(${Height} >= 30.0) && (${Height} < 50.0)", "color('#FFFF00')"],
```

["($ { Height } > = 30.0)&($ { Height } < 50.0)"，"颜色(# FFFF00 ')"],

```
["(${Height} >= 50.0) && (${Height} < 70.0)", "color('#00FF00')"],
```

["($ { Height } > = 50.0)&($ { Height } < 70.0)"，"颜色(# 00ff 00 ')"],

```
["(${Height} >= 70.0) && (${Height} < 100.0)", "color('#00FFFF')"],
```

["($ { Height } > = 70.0)&($ { Height } < 100.0)"，"颜色(# 00 ffff ')"],

["(${Height} >= 100.0)", "color('#0000FF')"]

["($ { Height } > = 100.0)"，"颜色(#0000FF')"]

]

]

}

}

Since conditions are evaluated in order, the above can be written more concisely as the following:

由于条件是按顺序评估的，上面的内容可以更简洁地写成如下：

"color" : {

"颜色":{

"conditions" : [

"条件":[

["(${Height} >= 100.0)", "color('#0000FF')"],

["($ { Height } > = 100.0)"、"颜色(# 0000 ff")"],

["(${Height} >= 70.0)", "color('#00FFFF')"],

["($ { Height } > = 70.0)"、"颜色(# 00 ffff")"],

["(${Height} >= 50.0)", "color('#00FF00')"],

["($ { Height } > = 50.0)"，"颜色(#00FF00')"],

["(${Height} >= 30.0)", "color('#FFFF00')"],

["($ { Height } > = 30.0)"、"颜色(# ffff 00")"],

["(${Height} >= 10.0)", "color('#FF0000')"],

["($ { Height } > = 10.0)"，"颜色(#FF0000')"],

["(${Height} >= 1.0)", "color('#FF00FF')"]

["($ { Height } > = 1.0)"，"颜色(#FF00FF')"]

]

]

}

}

Defining variables

定义变量

Commonly used expressions may be stored in a defines object with a variable name as a key.If a variable references the name of a defined expression, it is replaced with the result of the referenced evaluated expression:

常用表达式可以存储在以变量名为关键字的定义对象中。如果变量引用已定义表达式的名称，它将被引用的计算表达式的结果替换:

{

{

"defines" : {

"定义":{

"NewHeight" : "clamp((${Height} - 0.5) / 2.0, 1.0, 255.0)",

"新高度":"夹钳((${Height} - 0.5) / 2.0、1.0、255.0)"。

"HeightColor" : "rgb(${Height}, ${Height}, ${Height})"

"高度颜色":"rgb(${Height}、${Height}、${Height})"

},

},

"color" : {

"颜色":{

"conditions" : [

"条件":[

["(${NewHeight} >= 100.0)", "color('#0000FF') * ${HeightColor}"],

["($ {新高度} >= 100.0)"、 "颜色(# 0000 ff) * $ {高度颜色}"],

["(${NewHeight} >= 50.0)", "color('#00FF00') * ${HeightColor}"],

["($ {新高度} >= 50.0)"、 "颜色(# 00ff 00) * $ {高度颜色}"],

["(${NewHeight} >= 1.0)", "color('#FF0000') * ${HeightColor}"]

["($ {新高度} >= 1.0)"、 "颜色(# ff 0000) * $ {高度颜色}"]

]

]

},

},

"show" : "${NewHeight} < 200.0"

" show ":$ { NewHeight } < 200.0 "

}

}

A define expression may not reference other defines;however, it may reference feature properties with the same name.In the style below a feature of height 150 gets the color red:

定义表达式不能引用其他定义；但是，它可能引用同名的特征属性。在下面的样式中，高度为 150 的特征得到红色:

{

{

"defines" : {

"定义":{

"Height" : "${Height}/2.0}",

"高度":${Height}/2.0} ",

},

},

"color" : {

"颜色":{

"conditions" : [

"条件":[

["(${Height} >= 100.0)", "color('#0000FF')"],

["(\$ { Height } > = 100.0)"、 "颜色(# 0000 ff")"],

["(${Height} >= 1.0)", "color('#FF0000')"]

["($ { Height } > = 1.0)"，"颜色(#FF0000')"]

]

]

}

}

}

}

Meta property

元属性

Non-visual properties of a feature can be defined using the meta property.For example, the following sets a description meta property to a string containing the feature name:

可以使用元属性定义特征的非视觉属性。例如，以下内容将描述元属性设置为包含要素名称的字符串:

{

{

"meta" : {

" meta" : {

"description" : '"Hello, ${featureName}."'

"描述":"您好，$ {功能名称}"。"

}

}

}

}

A meta property expression can evaluate to any type.For example:

元属性表达式可以计算为任何类型。例如:

{

{

"meta" : {

" meta" : {

"featureColor" : "rgb(${red}, ${green}, ${blue})",

" FeatureColor ":" RGB($ {红色}、$ {绿色}、$ {蓝色})",

"featureVolume" : "${height} * ${width} * ${depth}"

功能卷:"$ {高度} * $ {宽度} * $ {深度}"

}

}

}

}

Expressions

公式

The language for expressions is a small subset of JavaScript (EMCAScript 5), plus native vector and regular expression types and access to tileset feature properties in the form of readonly variables.

表达式语言是 JavaScript (EMCAScript 5)的一个小子集，加上本机向量和正则表达式类型，以及对只读变量形式的 tileset 特性属性的访问。

Implementation Note: Cesium uses the jsep JavaScript expression parser library to parse style expressions into an abstract syntax tree (AST).

实现注意:铯使用 jsep JavaScript 表达式解析器库将样式表达式解析成抽象语法树(AST)。

Semantics

语义学

Dot notation is used to access properties by name, e.g., building.name.

点符号用于按名称访问属性，例如建筑名称

Bracket notation ([]) is also used to access properties, e.g., building['name'], or arrays, e.g., temperatures[1].

括号符号([)也用于访问属性，如建筑[的"名称")，或数组，如温度[1]。

Functions are called with parenthesis (()) and comma-separated arguments, e.g., (isNaN(0.0), color('cyan', 0.5)).

函数用括号(())和逗号分隔的参数调用，例如，(isNaN(0.0)，color('青色'，0.5))。

Operators

经营者

The following operators are supported with the same semantics and precedence as JavaScript.

支持下列运算符的语义和优先级与 JavaScript 相同。

Unary: +, -, !

一元:+，-，！

Not supported: ~

不支持:~

Binary: ||, &&, ===, !==, <, >, <=, >=, +, -, *, /, %, =~, !~

二进制:||，&&，===，！==，<，>，< =，> =，+，-，*，/，%，=~，！~

Not supported: |, ^, &, <<, >>, and >>>

不支持:|、^、&、<<、>>和>>>

Ternary: ?:

三元:以下内容:

(and) are also supported for grouping expressions for clarity and precedence.

(和)也支持对表达式进行分组，以提高清晰度和优先级。

Logical || and && implement short-circuiting;true || expression does not evaluate the right expression, and false && expression does not evaluate the right expression.

逻辑||和&&实现短路；true ||表达式不计算正确的表达式，false &&表达式不计算正确的表达式。

Similarly, true ?leftExpression : rightExpression only executes the left expression, and false ? leftExpression : rightExpression only executes the right expression.

同样，是真的吗？左表达式:右表达式只执行左表达式，是否为假？左表达式:右表达式只执行右表达式。

Types

类型

The following types are supported:

支持以下类型:

Boolean

布尔型

Null

空

Undefined

不明确的

Number

数字

String

线

Array

排列

vec2

vec2

vec3

vec3

vec4

vec4

RegExp

RegExp

All of the types except vec2, vec3, vec4, and RegExp have the same syntax and runtime behavior as JavaScript.vec2, vec3, and vec4 are derived from GLSL vectors and behave similarly to JavaScript Object (see the Vector section).Colors derive from CSS3 Colors and are implemented as vec4.RegExp is derived from JavaScript and described in the RegExp section.

除了 vec2、vec3、vec4 和 RegExp 之外，所有类型都具有与 JavaScript 相同的语法和运行时行为。向量 2、向量 3 和向量 4 源自 GLSL 向量，其行为类似于 JavaScript 对象(参见向量部分)。颜色源自 CSS3 颜色，并作为矢量 4 实现。RegEx 源自 JavaScript，并在 RegEx 部分进行了描述。

Example expressions for different types include the following:

不同类型的示例表达式包括:

true, false

对，错

null

空

undefined

不明确的

1.0, NaN, Infinity

1.0，NaN，Infinity

'Cesium', "Cesium"

铯'，"铯"

[0, 1, 2]

[0，1，2]

vec2(1.0, 2.0)

vec2(1.0，2.0)

vec3(1.0, 2.0, 3.0)

vec3(1.0，2.0，3.0)

vec4(1.0, 2.0, 3.0, 4.0)

vec4(1.0，2.0，3.0，4.0)

color('#00FFFF')

颜色(#00FFFF)

regExp('^Chest'))

regExp('^Chest'))

Number

数字

As in JavaScript, numbers can be NaN or Infinity.The following test functions are supported:

和 JavaScript 一样，数字可以是 NaN 或 Infinity。支持以下测试功能：

isNaN(testValue : Number) : Boolean

isNaN(测试值:数字) :布尔值

isFinite(testValue : Number) : Boolean

isFinite(测试值:数字) :布尔值

Vector

矢量

The styling language includes 2, 3, and 4 component floating-point vector types: vec2, vec3, and vec4.Vector constructors share the same rules as GLSL:

样式语言包括 2、3 和 4 个组成浮点向量类型:向量 2、向量 3 和向量 4。向量构造函数与 GLSL 有相同的规则:

vec2

vec2

vec2(xy : Number) - initialize each component with the number

向量 2(xy:数字)-用数字初始化每个组件

vec2(x : Number, y : Number) - initialize with two numbers

向量 2(x:数字，y:数字)-用两个数字初始化

vec2(xy : vec2) - initialize with another vec2

vec2(xy : vec2) -用另一个 vec2 初始化

vec2(xyz : vec3) - drops the third component of a vec3

vec2(xyz : vec3) -删除 vec3 的第三个成分

vec2(xyzw : vec4) - drops the third and fourth component of a vec4

vec2(xyzw : vec4) -删除 vec4 的第三和第四个组件

vec3

vec3

vec3(xyz : Number) - initialize each component with the number

向量 3(xyz : Number) -用数字初始化每个组件

vec3(x : Number, y : Number, z : Number) - initialize with three numbers

向量 3(x:数字，y:数字，z:数字)-用三个数字初始化

vec3(xyz : vec3) - initialize with another vec3

vec3(xyz : vec3) -用另一个 vec3 初始化

vec3(xyzw : vec4) - drops the fourth component of a vec4

vec3(xyzw : vec4) -去除 vec4 的第四个成分

vec3(xy : vec2, z : Number) - initialize with a vec2 and number

向量 3(xy:向量 2，z:数字)-用向量 2 和数字初始化

vec3(x : Number, yz : vec2) - initialize with a vec2 and number

向量 3(x:数字，yz:向量 2) -用向量 2 和数字初始化

vec4

vec4

vec4(xyzw : Number) - initialize each component with the number

vec4(xyzw : Number) -用数字初始化每个组件

vec4(x : Number, y : Number, z : Number, w : Number) - initialize with four numbers

向量 4(x:数字，y:数字，z:数字，w:数字)-用四个数字初始化

vec4(xyzw : vec4) - initialize with another vec4

vec4(xyzw : vec4) -用另一个 vec4 初始化

vec4(xy : vec2, z : Number, w : Number) - initialize with a vec2 and two numbers

向量 4(xy:向量 2，z:数字，w:数字)-用向量 2 和两个数字初始化

vec4(x : Number, yz : vec2, w : Number) - initialize with a vec2 and two numbers

向量 4(x:数字，yz:向量 2，w:数字)-用向量 2 和两个数字初始化

vec4(x : Number, y : Number, zw : vec2) - initialize with a vec2 and two numbers

向量 4(x:数字，y:数字，zw:向量 2) -用向量 2 和两个数字初始化

vec4(xyz : vec3, w : Number) - initialize with a vec3 and number

vec4(xyz : vec3，w : Number) -用 vec3 和 Number 初始化

vec4(x : Number, yzw : vec3) - initialize with a vec3 and number

vec4(x : Number，yzw : vec3) -用 vec3 和 Number 初始化

Vector usage

向量用法

vec2 components may be accessed with

vec2 组件可通过以下方式访问

.x, .y

。x、。y

.r, .g

。r、。g

[0], [1]

[0]，[1]

vec3 components may be accessed with

vec3 组件可通过以下方式访问

.x, .y, .z

。x、。y、。z

.r, .g, .b

。r、。g、。b

[0], [1], [2]

[0]，[1]，[2]

vec4 components may be accessed with

vec4 组件可通过以下方式访问

.x, .y, .z, .w

。x、。y、。z、。w

.r, .g, .b, .a

。r、。g、。b、。a

[0], [1], [2], [3]

[0]，[1]，[2]，[3]

Unlike GLSL, the styling language does not support swizzling.For example, vec3(1.0).xy is not supported.

与 GLSL 不同，造型语言不支持痛饮。例如，vec3(1.0)。不支持 xy。

Vectors support the following unary operators: -, +.

向量支持下列一元运算符:，+。

Vectors support the following binary operators by performing component-wise operations: ===, !==, +, -, *, /, and %.For example vec4(1.0) === vec4(1.0) is true since the x, y, z, and w components are equal.Operators are essentially overloaded for vec2, vec3, and vec4.

向量通过执行组件操作支持以下二进制运算符:===，！==、+、-、*、/、和%。例如，vec 4(1.0)= vec 4(1.0)为真，因为 x、y、z 和 w 分量相等。vec2、vec3 和 vec4 的运算符基本上超载。

vec2, vec3, and vec4 have a toString function for explicit (and implicit) conversion to strings in the format '(x, y)', '(x, y, z)', and '(x, y, z, w)'.

vec2、vec3 和 vec4 具有 toString 函数，用于显式(和隐式)转换为格式
为'(x，y)'、'(x，y，z)'和'(x，y，z，w)'的字符串。

toString() : String

toString():字符串

vec2, vec3, and vec4 do not expose any other functions or a prototype object.

vec2、vec3 和 vec4 不公开任何其他功能或原型对象。

Color

颜色

Colors are implemented as vec4 and are created with one of the following functions:

颜色实现为 vec4，并通过以下功能之一创建:

color()

颜色()

color(keyword : String, [alpha : Number])

颜色(关键词:字符串，[字母:数字)

color(6-digit-hex : String, [alpha : Number])

颜色(6 位十六进制:字符串，[字母:数字)

color(3-digit-hex : String, [alpha : Number])

颜色(三位数十六进制:字符串，[字母:数字))

rgb(red : Number, green : Number, blue : Number)

rgb(红色:数字，绿色:数字，蓝色:数字)

rgba(red : Number, green : Number, blue : Number, alpha : Number)

rgba(红色:数字，绿色:数字，蓝色:数字，字母:数字)

hsl(hue : Number, saturation : Number, lightness : Number)

色调:数字，饱和度:数字，亮度:数字

hsla(hue : Number, saturation : Number, lightness : Number, alpha : Number)

hsla(色调:数字，饱和度:数字，亮度:数字，阿尔法:数字)

Calling color() with no arguments is the same as calling color('#FFFFFF').

不带参数调用颜色()与调用颜色(#FFFFFF)相同。

Colors defined by a case-insensitive keyword (e.g., 'cyan') or hex rgb are passed as strings to the color function.For example:

由不区分大小写的关键字(如"青色")或十六进制 rgb 定义的颜色作为字符串传递给颜色函数。例如:

color('cyan')

颜色("青色")

color('#00FFFF')

颜色(#00FFFF)

color('#0FF')

颜色(#0FF)

These color functions have an optional second argument that is an alpha component to define opacity, where 0.0 is fully transparent and 1.0 is fully opaque.For example:

这些颜色函数有一个可选的第二个参数，它是定义不透明度的 alpha 组件，其中 0.0 是完全透明的，1.0 是完全不透明的。例如:

color('cyan', 0.5)

颜色("青色"，0.5)

Colors defined with decimal RGB or HSL are created with rgb and hsl functions, respectively, just as in CSS (but with percentage ranges from 0.0 to 1.0 for 0% to 100%, respectively).For example:

用十进制 rgb 或 hsl 定义的颜色分别用 RGB 和 HSL 函数创建，就像在 CSS 中一样(但是百分比范围分别为 0.0 到 1.0，分别为 0%到 100%)。例如:

rgb(100, 255, 190)

rgb(100，255，190)

hsl(1.0, 0.6, 0.7)

hsl(1.0，0.6，0.7)

The range for rgb components is 0 to 255, inclusive.For hsl, the range for hue, saturation, and lightness is 0.0 to 1.0, inclusive.

rgb 分量的范围是 0 到 255，包括 0 和 255。对于 hsl，色调、饱和度和亮度的范围为 0.0 至 1.0(含 0.0 和 1.0)。

Colors defined with rgba or hsla have a fourth argument that is an alpha component to define opacity, where 0.0 is fully transparent and 1.0 is fully opaque.For example:

用 rgba 或 hsla 定义的颜色有第四个参数，它是定义不透明度的 alpha 组件，其中 0.0 是完全透明的，1.0 是完全不透明的。例如:

rgba(100, 255, 190, 0.25)

rgba(100，255，190，0.25)

hsla(1.0, 0.6, 0.7, 0.75)

hsla(1.0，0.6，0.7，0.75)

Colors are equivalent to the vec4 type and share the same functions, operators, and component accessors.Color components are stored in the range 0.0 to 1.0.

颜色等同于 vec4 类型，并共享相同的函数、运算符和组件访问器。颜色分量存储在 0.0 到 1.0 的范围内。

For example:

例如:

color('red').x, color('red').r, and color('red')[0] all evaluate to 1.0.

颜色("红色")。x，颜色("红色")。r 和颜色("红色")[0 都评估为 1.0。

color('red').toString() evaluates to (1.0, 0.0, 0.0, 1.0)

颜色("红色")。toString()计算结果为(1.0，0.0，0.0，1.0)

color('red') * vec4(0.5) is equivalent to vec4(0.5, 0.0, 0.0, 0.5)

颜色('红色')* vec4(0.5)相当于 vec4(0.5，0.0，0.0，0.5)

RegExp

RegExp

Regular expressions are created with the following functions, which behave like the JavaScript RegExp constructor:

正则表达式由以下函数创建，这些函数的行为类似于 JavaScript RegExp 构造函数:

regExp()

正则表达式()

regExp(pattern : String, [flags : String])

正则表达式(模式:字符串，[标志:字符串)

Calling regExp() with no arguments is the same as calling regExp('(?:)').

不带参数调用 regExp()与调用 regExp((？:)')。

If specified, flags can have any combination of the following values:

如果指定，标志可以具有下列值的任意组合:

g - global match

g -全球匹配

i - ignore case

i -忽略案例

m - multiline

m -多线

u - unicode

u - unicode

y- sticky

y-粘性

Regular expressions support these functions:

正则表达式支持这些功能:

test(string : String) : Boolean - Tests the specified string for a match.

测试(字符串:字符串) :布尔型-测试指定字符串是否匹配。

exec(string : String) : String - Executes a search for a match in the specified string.If the search succeeds, it returns the first instance of a captured String.If the search fails, it returns null.

exec(字符串:字符串) :字符串-在指定字符串中搜索匹配项。如果搜索成功，它将返回捕获字符串的第一个实例。如果搜索失败，它将返回 null。

For example:

例如:

{

{

"Name" : "Building 1"

"名称":"1 号楼"

}

}

regExp('a').test('abc') === true

正则表达式(' a ')。测试(' abc') ===真

regExp('a(.)', 'i').exec('Abc') === 'b'

正则表达式(' a(')。)'，' I ')。exec('Abc') === 'b '

regExp('Building\s(\d)').exec(${Name}) === '1'

正则表达式(' Building\s(\d)')。exec(${Name}) === '1 '

Regular expressions have a toString function for explicit (and implicit) conversion to strings in the format 'pattern':

正则表达式具有 toString 函数，用于显式(和隐式)转换为格式为"模式"的字符串:

toString() : String

toString():字符串

Regular expressions do not expose any other functions or a prototype object.

正则表达式不公开任何其他函数或原型对象。

The operators =~ and !~ are overloaded for regular expressions.The =~ operator matches the behavior of the test function, and tests the specified string for a match.It returns true if one is found, and false if not found.The !~ operator is the inverse of the =~ operator.It returns true if no matches are found, and false if a match is found.Both operators are commutative.

运算符=~和！~对于正则表达式是重载的。=~运算符匹配测试函数的行为，并测试指定字符串的匹配情况。如果找到了，则返回 true 如果找不到，则返回 false。那个。~运算符是=~运算符的反义词。如果没有找到匹配项，则返回 true 如果找到匹配项，则返回 false。两个算子都是可交换的。

For example, the following expressions all evaluate to true:

例如，以下表达式都计算为真:

regExp('a') =~ 'abc'

正则表达式(' a') =~ 'abc '

'abc' =~ regExp('a')

ABC ' = ~ RegEx(' a ')

regExp('a') !~ 'bcd'

正则表达式(' a ')! ~ 'bcd '

'bcd' !~ regExp('a')

bcd! ~ RegEx(' a ')

Operator rules

操作员规则

Unary operators + and - operate only on number and vector expressions.

一元运算符+和-只对数字和向量表达式进行运算。

Unary operator !operates only on boolean expressions.

一元运算符！仅在布尔表达式上操作。

Binary operators <, <=, >, and >= operate only on number expressions.

二进制运算符<、< =、>和> =只对数字表达式进行运算。

Binary operators || and && operate only on boolean expressions.

二元运算符||和&&只对布尔表达式进行运算。

Binary operator + operates on the following expressions:

二元运算符+对以下表达式进行运算:

Number expressions

数字表达式

Vector expressions of the same type

相同类型的向量表达式

If at least one expressions is a string, the other expression is converted to a string following String Conversions, and the operation returns a concatenated string, e.g. "name" + 10 evaluates to "name10"

如果至少有一个表达式是字符串，则在字符串转换后，另一个表达式被转换为字符串，并且该操作返回串联字符串，例如，"name10 计算为"name10"

Binary operator - operates on the following expressions:

二元运算符-对以下表达式进行运算:

Number expressions

数字表达式

Vector expressions of the same type

相同类型的向量表达式

Binary operator * operates on the following expressions:

二元运算符*对以下表达式进行运算:

Number expressions

数字表达式

Vector expressions of the same type

相同类型的向量表达式

Mix of number expression and vector expression, e.g. 3 * vec3(1.0) and vec2(1.0) * 3

数字表达和载体表达的混合，例如 3 * vec3(1.0)和 vec2(1.0) * 3

Binary operator / operates on the following expressions:

二元运算符/对以下表达式进行运算:

Number expressions

数字表达式

Vector expressions of the same type

相同类型的向量表达式

Vector expression followed by number expression, e.g.vec3(1.0) / 3

向量表达式后跟数字表达式，例如 vec3(1.0) / 3

Binary operator % operates on the following expressions:

二元运算符%对以下表达式进行运算:

Number expressions

数字表达式

Vector expressions of the same type

相同类型的向量表达式

Binary equality operators === and !== operate on any expressions.The operation returns false if the expression types do not match.

二元等式运算符= = = =和！==对任何表达式进行运算。如果表达式类型不匹配，该操作返回 false。

Binary regExp operators =~ and !~ require one argument to be a string expression and the other to be a RegExp expression.

二元 regExp 运算符=~和！~要求一个参数是字符串表达式，另一个参数是 RegExp 表达式。

Ternary operator ?: conditional argument must be a boolean expression.

三元运算符？:条件参数必须是布尔表达式。

Type conversions

类型转换

Explicit conversions between primitive types are handled with Boolean, Number, and String functions.

原语类型之间的显式转换由布尔函数、数字函数和字符串函数处理。

Boolean(value : Any) : Boolean

布尔型(值:任意) :布尔型

Number(value : Any) : Number

数字(值:任意) :数字

String(value : Any) : String

字符串(值:任意) :字符串

For example:

例如:

Boolean(1) === true

布尔(1) ==真

Number('1') === 1

数字(' 1') === 1

String(1) === '1'

字符串(1) === '1 '

Boolean and Number follow JavaScript conventions.String follows String Conversions.

布尔和数字遵循 JavaScript 约定。字符串跟随字符串转换。

These are essentially casts, not constructor functions.

这些本质上是强制转换，而不是构造函数。

The styling language does not allow for implicit type conversions, unless stated above.Expressions like vec3(1.0) === vec4(1.0) and "5" < 6 are not valid.

样式语言不允许隐式类型转换，除非上面有说明。像 vec 3(1.0)= = vec 4(1.0)和" 5" < 6 "这样的表达式无效。

String conversions

字符串转换

vec2, vec3, vec4, and RegExp expressions are converted to strings using their toString methods.All other types follow JavaScript conventions.

vec2、vec3、vec4 和 RegExp 表达式使用它们的 toString 方法转换为字符串。所有其他类型都遵循 JavaScript 约定。

true - "true"

真-"真"

false - "false"

假-"假"

null - "null"

空-"空"

undefined - "undefined"

未定义-"未定义"

5.0 - "5"

5.0-"5"

NaN - "NaN"

NAn-"NAn"

Infinity - "Infinity"

无限──"无限"

"name" - "name"

"名称"-"名称"

[0, 1, 2] - "[0, 1, 2]"

[0，1，2] - "[0，1，2]"

vec2(1, 2) - "(1, 2)"

向量 2(1，2)-(1，2)

vec3(1, 2, 3) - "(1, 2, 3)"

向量 3(1，2，3)-(1，2，3)

vec4(1, 2, 3, 4) - "(1, 2, 3, 4)"

vec4(1，2，3，4)-(1，2，3，4)

RegExp('a') - "/a/"

正则表达式(' a') - "/a/"

Constants

常数

The following constants are supported by the styling language:

样式语言支持以下常量:

Math.PI

数学。PI

Math.E

数学。E

PI

PI

The mathematical constant PI, which represents a circle's circumference divided by its diameter, approximately 3.14159.

数学常数 π，代表一个圆的周长除以其直径，约为 3.14159。

{

{

"show" : "cos(${Angle} + Math.PI) < 0"

"显示":"cos(${Angle} +数学"。PI) < 0 "

}

}

E

E

Euler's constant and the base of the natural logarithm, approximately 2.71828.

欧拉常数和自然对数的底数，约为 2.71828。

{

{

"color" : "color() * pow(Math.E / 2.0, ${Temperature})"

"颜色":"颜色()*力量(数学)。E / 2.0，$ {温度})"

}

}

Variables

变量

Variables are used to retrieve the property values of individual features in a tileset.Variables are identified using the ES 6 (ECMAScript 2015) template literal syntax, i.e., ${feature.identifier} or ${feature['identifier']}, where the identifier is the case-sensitive property name.feature is implicit and can be omitted in most cases.

变量用于检索波浪集中各个要素的属性值。变量使用 ES 6 (ECMAScript 2015)模板字面语法来标识，即$ {功能.标识符}或$ {功能['标识符']}，其中标识符是区分大小写的属性名。特性是隐式的，在大多数情况下可以省略。

Variables can be used anywhere a valid expression is accepted, except inside other variable identifiers.For example, the following is not allowed:

变量可以在任何接受有效表达式的地方使用，除了在其他变量标识符内部。例如，不允许以下情况:

${foo[${bar}]}

${foo[${bar}]}

If a feature does not have a property with the specified name, the variable evaluates to undefined.Note that the property may also be null if null was explicitly stored for a property.

如果某个要素没有指定名称的属性，则该变量的计算结果为未定义。请注意，如果为属性显式存储 null，则属性也可能为 null。

Variables may be any of the supported native JavaScript types:

变量可以是任何支持的本机 JavaScript 类型:

Boolean

布尔型

Null

空

Undefined

不明确的

Number

数字

String

线

Array

排列

For example:

例如:

{

{

"enabled" : true,

"启用":正确,

"description" : null,

【描述】:空,

"order" : 1,

【订单】:1、

"name" : "Feature name"

"名称":"功能名称"

}

}

${enabled} === true

${enabled} ===真

${description} === null

${description} === null

${order} === 1

${order} === 1

${name} === 'Feature name'

${name} === '功能名称'

Additionally, variables originating from vector properties stored in the Batch Table binary are treated as vector types:

此外，源自存储在批处理表二进制文件中的向量属性的变量被视为向量类型:

componentType	variable type

"VEC2"	vec2
"VEC3"	vec3
"VEC4"	vec4

组件类型	可变类型
"VEC2"	vec2
"VEC3"	vec3
"VEC4"	vec4

Variables can be used to construct colors or vectors.For example:

变量可以用来构造颜色或向量。例如:

rgba(${red}, ${green}, ${blue}, ${alpha})

rgba($ {红色}、$ {绿色}、$ {蓝色}、${alpha})

vec4(${temperature})

vec 4($ {温度})

Dot or bracket notation is used to access feature subproperties.For example:

点或括号表示法用于访问特征子属性。例如:

{

{

"address" : {

"地址":{

"street" : "Example street",

"街道":"示例街道",

"city" : "Example city"

"城市":"示例城市"

}

```
}

}

}
```

${address.street} === `Example street`

${address.street} === `示例街道'

${address['street']} === `Example street`

$ {地址['街道']} = = = = `街道示例'

${address.city} === `Example city`

${address.city} === `示例城市'

${address['city']} === `Example city`

$ {地址['城市']} = = = = `示例城市'

Bracket notation supports only string literals.

括号符号只支持字符串。

Top-level properties can be accessed with bracket notation by explicitly using the feature keyword.For example:

通过显式使用 feature 关键字，可以使用括号符号访问顶级属性。例如:

```
{

{

"address.street" : "Maple Street",
```

"地址.街道":"枫树街"，

```
"address" : {
```

"地址":{

```
"street" : "Oak Street"
```

"街":"橡树街"

```
}

}

}
```

}

${address.street} === `Oak Street`

${address.street} === `Oak Street '

${feature.address.street} === `Oak Street`

$ { feature . address . street } = = = ` Oak street '

${feature['address'].street} === `Oak Street`

$ {以["地址"为特色]。街道} === `橡树街'

${feature['address.street']} === `Maple Street`

$ {特写['地址.街道']} = = = = `枫树街'

To access a feature named feature, use the variable ${feature}.This is equivalent to accessing ${feature.feature}

要访问名为 feature 的功能，请使用变量${feature}。这相当于访问${feature.feature}

{

{

"feature" : "building"

"功能":"建筑"

}

}

${feature} === `building`

$ {功能} === `建筑'

${feature.feature} === `building`

${feature.feature} === `building '

Variables can also be substituted inside strings defined with backticks, for example:

变量也可以在用反斜杠定义的字符串中替换，例如:

{

{

"order" : 1,

【订单】:1、

"name" : "Feature name"

"名称":"功能名称"

}

}

`Name is ${name}, order is ${order}`

`名称为${name}，订单为${order} ' '

Bracket notation is used to access feature subproperties or arrays.For example:

括号符号用于访问功能子属性或数组。例如:

{

{

"temperatures" : {

"温度":{

"scale" : "fahrenheit",

"标度":"华氏"，

"values" : [70, 80, 90]

【价值】:[70、80、90】

}

}

}

}

${temperatures['scale']} === 'fahrenheit'

$ {温度['标度']} = = = = '华氏'

${temperatures.values[0]} === 70

$ {温度值[0]} === 70

${temperatures['values'][0]} === 70 // Same as (temperatures[values])[0] and temperatures.values[0]

$ {温度['值'][0]} === 70 //与(温度[值])[0]和温度相同。值[0]

Built-in functions

内置功能

The following built-in functions are supported by the styling language.Many of the built-in functions take either scalars or vectors as arguments.For vector arguments the function is applied component-wise and the resulting vector is returned.

样式语言支持以下内置函数。许多内置函数以标量或向量作为参数。对于向量参数，函数按组件方式应用，并返回结果向量。

abs

丙烯腈-丁二烯-苯乙烯

abs(x : Number) : Number

abs(x:数字) :数字

abs(x : vec2) : vec2

abs(x : vec2) : vec2

abs(x : vec3) : vec3

abs(x : vec3) : vec3

abs(x : vec4) : vec4

abs(x : vec4) : vec4

Returns the absolute value of x.

返回 x 的绝对值。

{

{

"show" : "abs(${temperature}) > 20.0"

" show ":" ABS($ { temperature })> 20.0 "

}

}

sqrt

sqrt

sqrt(x : Number) : Number

sqrt(x:数字) :数字

sqrt(x : vec2) : vec2

sqrt(x : vec2) : vec2

sqrt(x : vec3) : vec3

sqrt(x : vec3) : vec3

sqrt(x : vec4) : vec4

sqrt(x : vec4) : vec4

Returns the square root of x when x >= 0.Returns NaN when x < 0.

当 x >= 0 时，返回 x 的平方根。x < 0 时返回 NaN。

{

{

"color" : {

"颜色":{

"conditions" : [

"条件":[

["${temperature} >= 0.5", "color('#00FFFF')"],

["$ {温度} >= 0.5", "颜色(# 00 ffff")"],

["${temperature} >= 0.0", "color('#FF00FF')"]

["$ {温度} >= 0.0", "颜色(# FF00FF")"]

]

]

}

}

}

}

cos

cos

cos(angle : Number) : Number

cos(角度:数字) :数字

cos(angle : vec2) : vec2

cos(角度:vec2) : vec2

cos(angle : vec3) : vec3

cos(角度:vec3) : vec3

cos(angle : vec4) : vec4

cos(角度:vec4) : vec4

Returns the cosine of angle in radians.

返回角度的余弦值，单位为弧度。

{

{

"show" : "cos(${Angle}) > 0.0"

"显示":" cos(${Angle}) > 0.0 "

}

}

sin

犯罪

sin(angle : Number) : Number

正弦(角度:数字) :数字

sin(angle : vec2) : vec2

sin(角度:vec2) : vec2

sin(angle : vec3) : vec3

sin(角度:vec3) : vec3

sin(angle : vec4) : vec4

sin(角度:vec4) : vec4

Returns the sine of angle in radians.

以弧度为单位返回角度的正弦值。

{

{

"show" : "sin(${Angle}) > 0.0"

" show" : "sin(${Angle}) > 0.0 "

}

}

tan

黝黑色

tan(angle : Number) : Number

tan(角度:数字) :数字

tan(angle : vec2) : vec2

tan(角度:vec2) : vec2

tan(angle : vec3) : vec3

tan(角度:vec3) : vec3

tan(angle : vec4) : vec4

tan(角度:vec4) : vec4

Returns the tangent of angle in radians.

返回角度的正切值，单位为弧度。

{

{

"show" : "tan(${Angle}) > 0.0"

"显示":" tan(${Angle}) > 0.0 "

}

}

acos

应用控制操作系统

acos(angle : Number) : Number

acos(角度:数字) :数字

acos(angle : vec2) : vec2

acos(角度:vec2) : vec2

acos(angle : vec3) : vec3

acos(角度:vec3) : vec3

acos(angle : vec4) : vec4

acos(角度:vec4) : vec4

Returns the arccosine of angle in radians.

以弧度为单位返回角度的反余弦值。

{

{

"show" : "acos(${Angle}) > 0.0"

" show" : "acos(${Angle}) > 0.0 "

}

}

asin

印度历的 7 月

asin(angle : Number) : Number

asin(角度:数字) :数字

asin(angle : vec2) : vec2

asin(角度:vec2) : vec2

asin(angle : vec3) : vec3

asin(角度:vec3) : vec3

asin(angle : vec4) : vec4

asin(角度:vec4) : vec4

Returns the arcsine of angle in radians.

以弧度为单位返回角度的反正弦。

{

{

"show" : "asin(${Angle}) > 0.0"

" show" : "asin(${Angle}) > 0.0 "

}

}

atan

阿坦

atan(angle : Number) : Number

atan(角度:数字) :数字

atan(angle : vec2) : vec2

atan(角度:vec2) : vec2

atan(angle : vec3) : vec3

atan(角度:vec3) : vec3

atan(angle : vec4) : vec4

atan(角度:vec4) : vec4

Returns the arctangent of angle in radians.

以弧度为单位返回角度的反正切。

{

{

"show" : "atan(${Angle}) > 0.0"

" show" : "atan(${Angle}) > 0.0 "

}

}

atan2

atan2

atan2(y : Number, x : Number) : Number

atan2(y:数字，x:数字) :数字

atan2(y : vec2, x : vec2) : vec2

atan2(y : vec2，x : vec2) : vec2

atan2(y : vec3, x : vec3) : vec3

atan2(y : vec3，x : vec3) : vec3

atan2(y : vec4, x : vec4) : vec4

atan2(y : vec4，x : vec4) : vec4

Returns the arctangent of the quotient of y and x.

返回 y 和 x 的商的反正切。

{

{

"show" : "atan2(${GridY}, ${GridX}) > 0.0"

" show" : "atan2(${GridY}，${GridX}) > 0.0 "

}

}

radians

弧度

radians(angle : Number) : Number

弧度(角度:数字) :数字

radians(angle : vec2) : vec2

弧度(角度:矢量 2):矢量 2

radians(angle : vec3) : vec3

弧度(角度:矢量 3):矢量 3

radians(angle : vec4) : vec4

弧度(角度:矢量 4):矢量 4

Converts angle from degrees to radians.

将角度从角度转换为弧度。

{

{

"show" : "radians(${Angle}) > 0.5"

"显示":"弧度($ {角度}) > 0.5 "

}

}

degrees

度

degrees(angle : Number) : Number

度数(角度:数字) :数字

degrees(angle : vec2) : vec2

度数(角度:vec2) : vec2

degrees(angle : vec3) : vec3

度数(角度:矢量 3):矢量 3

degrees(angle : vec4) : vec4

度数(角度:矢量 4):矢量 4

Converts angle from radians to degrees.

将角度从弧度转换为角度。

{

{

"show" : "degrees(${Angle}) > 45.0"

"显示":"度数($ {角度}) > 45.0 "

}

}

sign

符号

sign(x : Number) : Number

符号(x:数字) :数字

sign(x : vec2) : vec2

符号(x : vec2) : vec2

sign(x : vec3) : vec3

符号(x : vec3) : vec3

sign(x : vec4) : vec4

符号(x : vec4) : vec4

Returns 1.0 when x is positive, 0.0 when x is zero, and -1.0 when x is negative.

x 为正数时返回 1.0，x 为零时返回 0.0，x 为负数时返回-1.0。

{

{

"show" : "sign(${Temperature}) * sign(${Velocity}) === 1.0"

"显示":"符号($ {温度}) *符号($ {速度}) === 1.0 "

}

}

floor

地面

floor(x : Number) : Number

楼层(x:数字) :数字

floor(x : vec2) : vec2

楼层(x : vec2) : vec2

floor(x : vec3) : vec3

楼层(x : vec3) : vec3

floor(x : vec4) : vec4

楼层(x : vec4) : vec4

Returns the nearest integer less than or equal to x.

返回小于或等于 x 的最近整数。

```
{

{

"show" : "floor(${Position}) === 0"

"显示":"楼层($ {位置}) === 0 "

}

}
```

ceil

装天花板

ceil(x : Number) : Number

上限(x:数字) :数字

ceil(x : vec2) : vec2

ceil(x : vec2) : vec2

ceil(x : vec3) : vec3

ceil(x : vec3) : vec3

ceil(x : vec4) : vec4

天花板(x : vec4) : vec4

Returns the nearest integer greater than or equal to x.

返回最接近的大于或等于 x 的整数。

{

{

"show" : "ceil(${Position}) === 1"

"显示":"上限($ {位置}) === 1 "

}

}

round

圆形物

round(x : Number) : Number

回合(x:数字) :数字

round(x : vec2) : vec2

圆形(x : vec2) : vec2

round(x : vec3) : vec3

圆形(x : vec3) : vec3

round(x : vec4) : vec4

圆形(x : vec4) : vec4

Returns the nearest integer to x. A number with a fraction of 0.5 will round in an implementation-defined direction.

返回最接近 x 的整数。分数为 0.5 的数字将沿实现定义的方向舍入。

{

{

"show" : "round(${Position}) === 1"

"显示":"圆形($ {位置}) === 1 "

}

}

exp

exp

exp(x : Number) : Number

实验(x:数字) :数字

exp(x : vec2) : vec2

exp(x : vec2) : vec2

exp(x : vec3) : vec3

exp(x : vec3) : vec3

exp(x : vec4) : vec4

exp(x : vec4) : vec4

Returns e to the power of x, where e is Euler's constant, approximately 2.71828.

将 e 返回 x 的幂，其中 e 是欧拉常数，约为 2.71828。

{

{

"show" : "exp(${Density}) > 1.0"

"显示":" exp($ {密度}) > 1.0 "

}

}

log

原木

log(x : Number) : Number

日志(x:数字) :数字

log(x : vec2) : vec2

日志(x : vec2) : vec2

log(x : vec3) : vec3

日志(x : vec3) : vec3

log(x : vec4) : vec4

日志(x : vec4) : vec4

Returns the natural logarithm (base e) of x.

返回 x 的自然对数(以 e 为底)。

{

{

"show" : "log(${Density}) > 1.0"

"显示":"日志($ {密度}) > 1.0 "

}

}

exp2

exp2

exp2(x : Number) : Number

示例 2(x:数字) :数字

exp2(x : vec2) : vec2

exp2(x : vec2) : vec2

exp2(x : vec3) : vec3

exp2(x : vec3) : vec3

exp2(x : vec4) : vec4

exp2(x : vec4) : vec4

Returns 2 to the power of x.

返回 2 的 x 次方。

{

{

"show" : "exp2(${Density}) > 1.0"

"显示":" ex p2($ {密度}) > 1.0 "

}

}

log2

log2

log2(x : Number) : Number

log2(x:数字) :数字

log2(x : vec2) : vec2

log2(x : vec2) : vec2

log2(x : vec3) : vec3

log2(x : vec3) : vec3

log2(x : vec4) : vec4

log2(x : vec4) : vec4

Returns the base 2 logarithm of x.

返回 x 的以 2 为底的对数。

{

{

"show" : "log2(${Density}) > 1.0"

"显示":" log2($ {密度}) > 1.0 "

}

}

fract

部分

fract(x : Number) : Number

分形(x:数字) :数字

fract(x : vec2) : vec2

分形(x : vec2) : vec2

fract(x : vec3) : vec3

分形(x : vec3) : vec3

fract(x : vec4) : vec4

分形(x : vec4) : vec4

Returns the fractional part of x. Equivalent to x - floor(x).

返回 x 的小数部分，相当于 x - floor(x)。

{

{

"color" : "color() * fract(${Density})"

"颜色":"颜色()*分形($ {密度})"

}

}

pow

粉末

pow(base : Number, exponent : Number) : Number

幂(基数:数，指数:数) :数

pow(base : vec2, exponent : vec2) : vec2

功率(基数:vec2，指数:vec2) : vec2

pow(base : vec3, exponent : vec3) : vec3

功率(基数:vec3，指数:vec3) : vec3

pow(base : vec4, exponent : vec4) : vec4

功率(基数:vec4，指数:vec4) : vec4

Returns base raised to the power of exponent.

返回指数幂的基数。

{

{

"show" : "pow(${Density}, ${Temperature}) > 1.0"

"显示":"功率($ {密度}、$ {温度}) > 1.0"

}

}

min

部

min(x : Number, y : Number) : Number

最小(x:数字，y:数字) :数字

min(x : vec2, y : vec2) : vec2

最小(x : vec2，y : vec2) : vec2

min(x : vec3, y : vec3) : vec3

最小(x : vec3，y : vec3) : vec3

min(x : vec4, y : vec4) : vec4

最小(x : vec4，y : vec4) : vec4

min(x : Number, y : Number) : Number

最小(x:数字，y:数字) :数字

min(x : vec2, y : Number) : vec2

最小(x : vec2，y : Number) : vec2

min(x : vec3, y : Number) : vec3

最小(x : vec3，y : Number) : vec3

min(x : vec4, y : Number) : vec4

最小(x : vec4，y : Number) : vec4

Returns the smaller of x and y.

返回 x 和 y 中较小的一个。

{

{

"show" : "min(${Width}, ${Height}) > 10.0"

"显示":"最小值($ {宽度}、$ {高度}) > 10.0 "

}

}

max

max

max(x : Number, y : Number) : Number

最大(x:数字，y:数字) :数字

max(x : vec2, y : vec2) : vec2

最大值(x : vec2，y : vec2) : vec2

max(x : vec3, y : vec3) : vec3

最大值(x : vec3，y : vec3) : vec3

max(x : vec4, y : vec4) : vec4

最大值(x : vec4，y : vec4) : vec4

max(x : Number, y : Number) : Number

最大(x:数字，y:数字) :数字

max(x : vec2, y : Number) : vec2

最大值(x : vec2，y : Number) : vec2

max(x : vec3, y : Number) : vec3

最大值(x : vec3，y : Number) : vec3

max(x : vec4, y : Number) : vec4

最大值(x : vec4，y : Number) : vec4

Returns the larger of x and y.

返回 x 和 y 中较大的一个。

{

{

"show" : "max(${Width}, ${Height}) > 10.0"

"显示":"最大($ {宽度}、$ {高度}) > 10.0 "

}

}

clamp

夹子

clamp(x : Number, min : Number, max : Number) : Number

夹钳(x:数量，最小值:数量，最大值:数量) :数量

clamp(x : vec2, min : vec2, max : vec2) : vec2

箝位(x : vec2，最小值:vec2，最大值:vec2) : vec2

clamp(x : vec3, min : vec3, max : vec3) : vec3

箝位(x : vec3，最小值:vec3，最大值:vec3) : vec3

clamp(x : vec4, min : vec4, max : vec4) : vec4

箝位(x : vec4，最小值:vec4，最大值:vec4) : vec4

clamp(x : Number, min : Number, max : Number) : Number

夹钳(x:数量，最小值:数量，最大值:数量) :数量

clamp(x : vec2, min : Number, max : Number) : vec2

夹钳(x : vec2，最小值:数值，最大值:数值):vec2

clamp(x : vec3, min : Number, max : Number) : vec3

夹钳(x : vec3，最小值:数量，最大值:数量):vec3

clamp(x : vec4, min : Number, max : Number) : vec4

夹具(x : vec4，最小值:数值，最大值:数值):vec4

Constrains x to lie between min and max.

将 x 限制在最小值和最大值之间。

{

{

"color" : "color() * clamp(${temperature}, 0.1, 0.2)"

颜色":" color() *夹钳(${temperature}，0.1，0.2)"

}

}

mix

混合

mix(x : Number, y : Number, a : Number) : Number

混合(x:数字，y:数字，a:数字) :数字

mix(x : vec2, y : vec2, a : vec2) : vec2

混合(x : vec2，y : vec2，a : vec2) : vec2

mix(x : vec3, y : vec3, a : vec3) : vec3

混合(x : vec3，y : vec3，a : vec3) : vec3

mix(x : vec4, y : vec4, a : vec4) : vec4

混合(x : vec4，y : vec4，a : vec4) : vec4

mix(x : Number, y : Number, a : Number) : Number

混合(x:数字，y:数字，a:数字) :数字

mix(x : vec2, y : vec2, a : Number) : vec2

混合(x : vec2，y : vec2，a : Number) : vec2

mix(x : vec3, y : vec3, a : Number) : vec3

混合(x : vec3，y : vec3，a : Number) : vec3

mix(x : vec4, y : vec4, a : Number) : vec4

混合(x : vec4，y : vec4，a : Number) : vec4

Computes the linear interpolation of x and y.

计算 x 和 y 的线性插值。

{

{

"show" : "mix(20.0, ${Angle}, 0.5) > 25.0"

"显示":"混合(20.0，＄{角度}，0.5) > 25.0 "

}

}

length

长度

length(x : Number) : Number

长度(x:数字) :数字

length(x : vec2) : vec2

长度(x : vec2) : vec2

length(x : vec3) : vec3

长度(x : vec3) : vec3

length(x : vec4) : vec4

长度(x : vec4) : vec4

Computes the length of vector x, i.e., the square root of the sum of the squared components.If x is a number, length returns x.

计算向量 x 的长度，即平方和的平方根。如果 x 是一个数字，长度返回 x。

{

{

"show" : "length(${Dimensions}) > 10.0"

"显示":"长度($ {尺寸}) > 10.0 "

}

}

distance

距离

distance(x : Number, y : Number) : Number

距离(x:数字，y:数字) :数字

distance(x : vec2, y : vec2) : vec2

距离(x : vec2，y : vec2) : vec2

distance(x : vec3, y : vec3) : vec3

距离(x : vec3，y : vec3) : vec3

distance(x : vec4, y : vec4) : vec4

距离(x : vec4，y : vec4) : vec4

Computes the distance between two points x and y, i.e., length(x - y).

计算两点 x 和 y 之间的距离，即长度(x - y)。

{

{

"show" : "distance(${BottomRight}, ${UpperLeft}) > 50.0"

" show ":"距离($ {右下角}，$ {上方}) > 50.0 "

}

}

normalize

使标准化

normalize(x : Number) : Number

标准化(x:数字) :数字

normalize(x : vec2) : vec2

归一化(x : vec2) : vec2

normalize(x : vec3) : vec3

归一化(x : vec3) : vec3

normalize(x : vec4) : vec4

归一化(x : vec4) : vec4

Returns a vector with length 1.0 that is parallel to x. When x is a number, normalize returns 1.0.

返回与 x 平行的长度为 1.0 的向量。当 x 是一个数字时，规格化返回 1.0。

{

{

"show" : "normalize(${RightVector}, ${UpVector}) > 0.5"

"显示":"归一化($ {右向量}、$ {上向量}) > 0.5 "

}

}

dot

点

dot(x : Number, y : Number) : Number

圆点(x:数字，y:数字) :数字

dot(x : vec2, y : vec2) : vec2

点(x : vec2，y : vec2) : vec2

dot(x : vec3, y : vec3) : vec3

点(x : vec3，y : vec3) : vec3

dot(x : vec4, y : vec4) : vec4

点(x : vec4，y : vec4) : vec4

Computes the dot product of x and y.

计算 x 和 y 的点积。

{

{

"show" : "dot(${RightVector}, ${UpVector}) > 0.5"

"显示":"点($ {右向量}，$ {上向量}) > 0.5 "

}

}

cross

跨过

cross(x : vec3, y : vec3) : vec3

十字(x : vec3，y : vec3) : vec3

Computes the cross product of x and y. This function only accepts vec3 arguments.

计算 x 和 y 的叉积。该函数只接受 vec3 参数。

{

{

"color" : "vec4(cross(${RightVector}, ${UpVector}), 1.0)"

颜色":"矢量 4(十字($ {右矢量}、$ {上矢量})，1.0)"

}

}

Notes

笔记

Comments are not supported.

不支持注释。

Point Cloud

点云

A Point Cloud is a collection of points that may be styled like other features.In addition to evaluating a point's color and show properties, a Point Cloud style may evaluate pointSize, or the size of each point in pixels.The default pointSize is 1.0.

点云是可以像其他要素一样进行样式化的点的集合。除了评估点的颜色和显示属性之外，点云样式还可以评估点大小，或者以像素为单位的每个点的大小。默认的点数大小是 1.0。

{

{

"color" : "color('red')",

"颜色":"颜色('红色')",

"pointSize" : "${Temperature} * 0.5"

"磅值":$ {温度} * 0.5 "

}

}

Implementations may clamp the evaluated pointSize to the system's supported point size range.For example, WebGL renderers may query ALIASED_POINT_SIZE_RANGE to get the system limits when rendering with POINTS.A pointSize of 1.0 must be supported.

实现可以将评估的点大小限制在系统支持的点大小范围内。例如，当使用点渲染时，WebGL 渲染器可以查询别名点大小范围来获得系统限制。必须支持磅值 1.0。

Point Cloud styles may also reference semantics from the Feature Table including position, color, and normal to allow for more flexible styling of the source data.

点云样式还可以引用特征表中的语义，包括位置、颜色和法线，以允许源数据的更灵活的样式。

${POSITION} is a vec3 storing the xyz Cartesian coordinates of the point before the RTC_CENTER and tile transform are applied.When the positions are quantized, ${POSITION} refers to the position after the QUANTIZED_VOLUME_SCALE is applied, but before QUANTIZED_VOLUME_OFFSET is applied.

${POSITION}是一个矢量 3，存储应用 RTC _ CENTER 和图块变换之前点的 xyz 笛卡尔坐标。当位置被量化时，${POSITION}指的是在应用量化体积标度之后，但在应用量化体积偏移量之前的位置。

${POSITION_ABSOLUTE} is a vec3 storing the xyz Cartesian coordinates of the point after the RTC_CENTER and tile transform are applied.When the positions are quantized, ${POSITION_ABSOLUTE} refers to the position after the QUANTIZED_VOLUME_SCALE, QUANTIZED_VOLUME_OFFSET, and tile transform are applied.

$ { POSITION _绝对值}是一个矢量 3，存储应用 RTC _ CENTER 和图块变换后点的 xyz 笛卡尔坐标。当位置被量化时，$ { POSITION _绝对值}指的是应用量化_体积_比例、量化_体积_偏移和平铺变换后的位置。

${COLOR} evaluates to a Color storing the rgba color of the point.When the Feature Table's color semantic is RGB or RGB565, ${COLOR}.alpha is 1.0.If no color semantic is defined, ${COLOR} evaluates to the application-specific default color.

${COLOR}评估为存储点 rgba 颜色的颜色。当要素表的颜色语义为 RGB 或 RGB565 时，$ {COLOR}。阿尔法是 1.0。如果没有定义颜色语义，${COLOR}将评估为应用程序特定的默认颜色。

${NORMAL} is a vec3 storing the normal, in Cartesian coordinates, of the point before the tile transform is applied.When normals are oct-encoded, ${NORMAL} refers to the decoded normal.If no normal semantic is defined in the Feature Table, ${NORMAL} evaluates to undefined.

${NORMAL}是一个向量 3，以笛卡尔坐标存储应用图块变换之前的点的法线。当法线被 oct 编码时，$ {NORMAL}指的是解码后的法线。如果特征表中没有定义正常语义，则${NORMAL}计算结果为未定义。

For example:

例如:

{

{

"color" : "${COLOR} * color('red')'",

"颜色":${COLOR} *颜色("红色")",

"show" : "${POSITION}.x > 0.5",

" show ":$ { POSITION }。x > 0.5",

"pointSize" : "${NORMAL}.x > 0 ?2 : 1"

" PointSiZe ":$ { NORMAL }。x > 0? 2 : 1 "

}

}

Implementation Note: Point cloud styling engines may often use a shader (GLSL) implementation, however some features of the expression language are not possible in pure a GLSL implementation.Some of these features include:

实现注意:点云样式引擎可能经常使用着色器(GLSL)实现，但是表达式语言的一些特性在纯 GLSL 实现中是不可能的。其中一些功能包括:

Evaluation of isNan and isFinite (GLSL 2.0+ supports isnan and isinf for these functions respectively)

isnan 和 isFinite 的评估(GLSL 2.0+分别支持 isNan 和 isinf 的这些功能)

The types null and undefined

类型为空且未定义

Strings, including accessing object properties (color()['r']) and batch table values

字符串，包括访问对象属性(颜色()[)和批处理表值

Regular expressions

正则表达式

Arrays of lengths other than 2, 3, or 4

长度不是 2、3 或 4 的数组

Mismatched type comparisons (e.g. 1.0 === false)

不匹配的类型比较(例如 1.0 ===假)

Array index out of bounds

数组索引越界

File extension and MIME type

文件扩展名和 MIME 类型

Tileset styles use the .json extension and the application/json mime type.

Tileset 样式使用。json 扩展和应用程序/json mime 类型。

Property reference

属性引用

style

风格

A 3D Tiles style.

3D 平铺样式。

Properties

性能

	Type	Description	Required
defines	object	A dictionary object of expression strings mapped to a variable name key that may be referenced throughout the style.If an expression references a defined variable, it is replaced with the evaluated result of the corresponding expression.	No
show	boolean, string, object	A boolean expression or conditions property which determines if a feature should be shown.	No, default: true
color	string, object	A color expression or conditions property which determines the color blended with the feature's intrinsic color.	No, default: color('#FFFFFF')
meta	object	A meta object which	No

	类型	描述	需要
		determines the values of non-visual properties of the feature.	

	类型	描述	需要
定义	目标	表达式字符串的字典对象，映射到可在整个样式中引用的变量名键。如果表达式引用了定义的变量，它将被相应表达式的计算结果替换。	不
显示	布尔、字符串、对象	一种布尔表达式或条件属性，用于确定是否应显示特征。	否，默认值:真
颜色	字符串，对象	一种颜色表达式或条件属性，用于确定与特征固有颜色混合的颜色。	否，默认:颜色(#FFFFFF)
meta	目标	一个元对象，用于确定特征的非视觉属性值。	不

Additional properties are not allowed.

不允许附加属性。

style.defines

样式。定义

A dictionary object of expression strings mapped to a variable name key that may be referenced throughout the style.If an expression references a defined variable, it is replaced with the evaluated result of the corresponding expression.

表达式字符串的字典对象，映射到可在整个样式中引用的变量名键。如果表达式引用了定义的变量，它将被相应表达式的计算结果替换。

Type: object

类型:对象

Required: No

必填:否

Type of each property: string

每个属性的类型:字符串

style.show

A boolean expression or conditions property which determines if a feature should be shown.

一种布尔表达式或条件属性,用于确定是否应显示特征。

Type: boolean, string, object

类型:布尔型、字符串型、对象型

Required: No, default: true

必需:否,默认:真

style.color

风格.颜色

A color expression or conditions property which determines the color blended with the feature's intrinsic color.

一种颜色表达式或条件属性,用于确定与特征固有颜色混合的颜色。

Type: string, object

类型:字符串、对象

Required: No, default: color('#FFFFFF')

必需:否,默认:颜色(#FFFFFF)

style.meta

A meta object which determines the values of non-visual properties of the feature.

一个元对象,用于确定特征的非视觉属性值。

Type: object

类型:对象

Required: No

必填:否

Type of each property: string

每个属性的类型:字符串

boolean expression

布尔表达式

A boolean or string with a 3D Tiles style expression that evaluates to a boolean.See Expressions.

一个布尔值或字符串，具有计算结果为布尔值的 3D 切片样式表达式。请参见表达式。

color expression

颜色表达

3D Tiles style expression that evaluates to a Color.See Expressions.

评估为颜色的 3D 切片样式表达式。请参见表达式。

conditions

情况

A series of conditions evaluated in order, like a series of if...else statements that result in an expression being evaluated.

按顺序评估的一系列条件，如一系列 if...导致表达式被求值的其他语句。

Properties

性能

	Type	Description	Required
conditions	array []	A series of boolean conditions evaluated in order.For the first one that evaluates to true, its value, the 'result' (which is also an expression), is evaluated and returned.Result expressions must all be the same type.If no condition evaluates to true, the result is undefined.When conditions is undefined, null, or an empty object, the result	No

		is undefined.	

	类型	描述	需要
情况	阵列[]	按顺序计算的一系列布尔条件。对于第一个计算为真的值，计算并返回它的值"result"(也是一个表达式)。结果表达式必须都是相同的类型。如果没有条件评估为真，则结果是未定义的。当条件未定义、为空或为空对象时，结果是未定义的。	不

Additional properties are not allowed.

不允许附加属性。

conditions.conditions

条件

A series of boolean conditions evaluated in order.For the first one that evaluates to true, its value, the 'result' (which is also an expression), is evaluated and returned.Result expressions must all be the same type.If no condition evaluates to true, the result is undefined.When conditions is undefined, null, or an empty object, the result is undefined.

按顺序计算的一系列布尔条件。对于第一个计算为真的值，计算并返回它的值"result"(也是一个表达式)。结果表达式必须都是相同的类型。如果没有条件评估为真，则结果是未定义的。当条件未定义、为空或为空对象时，结果是未定义的。

Type: array []

类型:阵列[]

Required: No

必填:否

condition

情况

An expression evaluated as the result of a condition being true.An array of two expressions.If the first expression is evaluated and the result is true, then the second expression is evaluated and returned as the result of the condition.

作为条件为真的结果而评估的表达式。两个表达式的数组。如果第一个表达式被求值并且结果为真，则第二个表达式被求值并作为条件的结果返回。

expression

表示

A valid 3D Tiles style expression.See Expressions.

有效的 3D 切片样式表达式。请参见表达式。

meta

meta

A series of property names and the expression to evaluate for the value of that property.

一系列属性名称和表达式，以计算该属性的值。

Additional properties are allowed.

允许附加属性。

Type of each property: expression

每个属性的类型:表达式

number expression

数字表达式

3D Tiles style expression that evaluates to a number.See Expressions.

计算结果为数字的 3D 切片样式表达式。请参见表达式。

Point Cloud Style

点云样式

A 3D Tiles style with additional properties for Point Clouds.

带有点云附加属性的 3D 切片样式。

Properties

性能

	Type	Description	Required
defines	object	A dictionary object of expression strings	No

		mapped to a variable name key that may be referenced throughout the style.If an expression references a defined variable, it is replaced with the evaluated result of the corresponding expression.	
show	boolean, string, object	A boolean expression or conditions property which determines if a feature should be shown.	No, default: true
color	string, object	A color expression or conditions property which determines the color blended with the feature's intrinsic color.	No, default: color('#FFFFFF')
meta	object	A meta object which determines the values of non-visual properties of the feature.	No
pointSize	number, string, object	A number expression or conditions property which determines the size of the points in pixels.	No, default: 1

	类型	描述	需要
定义	目标	表达式字符串的字典对象，映射到可在整个样式中引用的变量名键。如果表达式引用了定义的变量，它将被相应表达式的计算结果替换。	不
显示	布尔、字符串、对象	一种布尔表达式或条件属性，用于确定是否应显示特征。	否，默认值:真
颜色	字符串，对象	一种颜色表达式或条件	否，默认:颜色

		属性，用于确定与特征固有颜色混合的颜色。	(#FFFFFF)
meta	目标	一个元对象，用于确定特征的非视觉属性值。	不
点数大小	数字、字符串、对象	决定像素点大小的数字表达式或条件属性。	否，默认值:1

Additional properties are not allowed.

不允许附加属性。

PointCloudStyle.defines

点云样式.定义

A dictionary object of expression strings mapped to a variable name key that may be referenced throughout the style.If an expression references a defined variable, it is replaced with the evaluated result of the corresponding expression.

表达式字符串的字典对象，映射到可在整个样式中引用的变量名键。如果表达式引用了定义的变量，它将被相应表达式的计算结果替换。

Type: object

类型:对象

Required: No

必填:否

Type of each property: string

每个属性的类型:字符串

PointCloudStyle.show

PointCloudStyle.show

A boolean expression or conditions property which determines if a feature should be shown.

一种布尔表达式或条件属性，用于确定是否应显示特征。

Type: boolean, string, object

类型:布尔型、字符串型、对象型

Required: No, default: true

必需:否，默认:真

PointCloudStyle.color

点云样式.颜色

A color expression or conditions property which determines the color blended with the feature's intrinsic color.

一种颜色表达式或条件属性，用于确定与特征固有颜色混合的颜色。

Type: string, object

类型:字符串、对象

Required: No, default: color('#FFFFFF')

必需:否，默认:颜色(#FFFFFF)

PointCloudStyle.meta

PointCloudStyle.meta

A meta object which determines the values of non-visual properties of the feature.

一个元对象，用于确定特征的非视觉属性值。

Type: object

类型:对象

Required: No

必填:否

Type of each property: string

每个属性的类型:字符串

PointCloudStyle.pointSize

点云样式。点大小

A number expression or conditions property which determines the size of the points in pixels.

决定像素点大小的数字表达式或条件属性。

Type: number, string, object

类型:数字、字符串、对象

Required: No, default: 1

必需:否，默认:1

Annex A: Conformance Class Abstract Test Suite (Normative)

附件一:一致性类抽象测试套件(规范性)

An Abstract Test Suite is not required for a Community Standard

社区标准不需要抽象测试套件

Annex B: Contributor Acknowledgements (Non-normative)

附件二:投稿人致谢(非规范性)

Matt Amato

马特·阿马托

Erik Andersson

埃里克·安德森

Dan Bagnell

丹·巴内尔

Ray Bentley

雷·本特利

Jannes Bolling

Jannes Bolling

Dylan Brown

迪伦·布朗

Sarah Chow

莎拉·周

Paul Connelly

保罗·康纳利

Volker Coors

Volker Coors

Tom Fili

汤姆·菲利

Leesa Fini

Leesa Fini

Ralf Gutbell

拉尔夫·古贝尔

Frederic Houbie

弗雷德里克·胡比

Christopher Mitchell, Ph.D.

克里斯托弗·米切尔博士

Claus Nagel

克劳斯·内格尔

Jean-Philippe Pons

让-菲利普·庞斯

Carl Reed

卡尔·里德

Kevin Ring

凯文·林

Scott Simmons

斯科特·西蒙斯

Stan Tillman

斯坦·蒂尔曼

Piero Toffanin

皮耶罗·托法宁

Pano Voudouris

Pano Voudouris

Dave Wesloh

Dave Wesloh

Annex C: Revision History

附件三:修订历史

Date	Release	Author	Description
2018-06-04	1.0	Gabby Getz	Put 3D Tiles specification document into OGC document template
2018-08-03	1.0r1	Gabby Getz	Respond to public comments
2018-11-16	1.0r2	Gabby Getz	Updated conformance section to include schema references Update preface to address updates to WKT for CRS (OGC12-053r5) Revise Introduction section to clarify conceptual concepts, including tile, tile content, tile formats, features, properties, styles, and the Batch Table and Features Table and how glTF is used in 3D Tiles Updated "tile" and "HLOD" definitions, added "tile content", "tile format", "geometric error", "screen space error", and "spatial coherence" definitions Add reference to EPSG in CRS section Remove glTF transforms from CRS section.Add only to relevant sections after being introduced.

			Remove references to "generation tool"
			Fix phrasing describing Tileset JSON
			Revise Tile Formats intro
			Revise embedded glTF wording.
			Fix "required" property errors in the properties reference as generated from the schema
			Remove 3d_tileset_time 3D Tiles Style variable
			Add clarification for styling tilesets without features
			Update normative references to include: EPSG 4979 IETF RFC3986 IETF RFC2397 UTF-8 W3C CSS3 Color Change glTF specification from non-normative to normative reference

日期	发布	作者	描述
2018-06-04	1.0	Gabby Getz	将 3D 瓷砖规范文件放入 OGC 文件模板
2018-08-03	1.0r1	Gabby Getz	回应公众意见
2018-11-16	1.0r2	Gabby Getz	更新了一致性部分，以包含模式引用 更新前言，以解决 CRS (OGC12-053r5)对 WKT 的更新 修订简介部分以阐明概念概念，包括图块、图块内容、图块格式、特征、属性、样式、批处理表和特征表以及如何在 3D 图块中使用 glTF 更新了"图块"和

			"HLOD"定义，添加了"图块内容"、"图块格式"、"几何误差"、"屏幕空间误差"和"空间一致性"定义
			在通用报告系统部分添加对 EPSG 的引用
			从 CRS 部分删除 glTF 转换。引入后仅添加到相关部分。
			删除对"生成工具"的引用
			修正描述颚化符 JSON 的短语
			修改平铺格式介绍
			修改嵌入的 glTF 措辞。
			修复从架构生成的属性引用中的"必需"属性错误
			删除三维平铺样式变量
			为没有功能的样式颚化符添加说明
			更新规范性参考文献，以包括: EPSG 4979 IETF RFC3986 IETF RFC2397 UTF-8 W3C CSS3 颜色
			将 glTF 规范从非规范改为规范参考