XR Flood – Configuration schema

[Configuration file 2](#_Toc192105728)

[Markers 3](#_Toc192105729)

[Landmark types 4](#_Toc192105730)

[Assets 6](#_Toc192105731)

[3D models – Level-of-detail (LOD) 8](#_Toc192105732)

[LODs on VisionOS 8](#_Toc192105733)

[LODs on iOS, Android, Windows, MacOS 9](#_Toc192105734)

[Landmarks 10](#_Toc192105735)

[Landmarks from an external CSV 10](#_Toc192105736)

[Panoramas – VR points 12](#_Toc192105737)

[VR panorama in conjunction with GeoData 13](#_Toc192105738)

[GeoImage – Image projection 14](#_Toc192105739)

[GeoData – Flood visualization 16](#_Toc192105740)

[CSV file format for Flood data 17](#_Toc192105741)

# Configuration file

Both the flood simulation client and operator apps use a configuration file (in JSON format) to determine their content. Below is an empty configuration that includes all properties.

configuration.json

{

"ipAddress": "192.168.0.100",

"display\_size": {"x":0,"y":0,"z":0},

"markers": [],

"assets": [],

"landmarkTypes": []

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*ipAddress | string | The IP address on the local network where the operator app / server can be found. |
| \*display\_size | vector3 | The size of an imaginary cube positioned at the tracking marker origin. Any content outside of this cube is not rendered in the client app. |
| \*markers | array | A list of SerializableTrackingmarker type that defines the tracking marker(s) to be used. |
| \*assets | array | A list of SerializableAsset type that defines the 3D models that are available, and their model-specific attributes. |
| landmarkTypes | array | A list of SerializableLandmarkType that landmarks can refer to for their icon or grouping. |

## Markers

The Flood Simulation app can support multiple tracking markers in the same session. This is useful for showing different content on different locations denoted by a physical image, without interference of an operator. Usually, a single marker is used, as in the following example:

SerializableTrackingmarker (json)

{

"name": "IMD lab QR",

"defaultToLoad": "Kizugawa",

"link": "https://some.domain/path/to/marker.png",

"printwidth": 0.175

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*name | string | A unique name for the marker to identify it in the apps. |
| \*defaultToLoad | string | A reference to the ID of an asset that will be loaded by default when this marker is found. |
| \*link | string | A URL to the image that is to be used as the tracking marker. The image can be of type PNG, JPG, GIF and the URL needs to be a public download-able link. |
| \*printwidth | double | The size of the marker image in meters. The marker image should be square (width=height) and of high contrast. |

A tracking marker can be any image, including QR codes. It is recommended to use a high contrast image to provide lots of features, increasing detection speed and quality.

Example tracking marker image; QR with a link to the configuration file.



## Landmark types

The “Landmarks” section discusses the landmarks feature that makes use of the type array described here. Landmark types are a way of grouping landmarks and to provide icon images. Landmark types are optional, in which case the landmarks will not show an icon or be grouped.

SerializableLandmarkType (json)

"landmarkTypes": [

{

"id":1,

"name":"指定緊急避難場所",

"iconUrl": "https://some.domain/icon\_1.png"

},

{

"id":2,

"name":"指定避難場所",

"iconUrl": "https://some.domain/icon\_2.png"

},

...

]

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*id | int | A unique number representing the type of landmark. |
| \*name | string | A label to describe the landmark type. |
| \*iconUrl | string | A URL to the image that is to be used as icon when referring to this landmark type. The image can be of type PNG, JPG, GIF and the URL needs to be a public download-able link. |

Landmark types will automatically be enumerated in the operator app, to provide the ability to control landmark visibility in the client app. When a landmark type is unchecked, all landmarks with that type will be hidden.

Landmark types with their icons in the operator app.

A screenshot of a computer

Description automatically generated

## Assets

The assets list describes all 3D models that are going to be available in the client and operator apps, with all their asset-specific parameters and annotations. At least one asset is required per configuration. Below is an empty asset configuration:

{

"id": "Kizugawa",

"initialScale": 0.0003,

"conversionValue": 1.0,

"lodMultiplier": 1.0,

"models": [],

"originLatLonAlt": {"x": 0, "y": 0, "z": 0},

"minLatLonAlt": {"x": 0, "y": 0, "z": 0},

"maxLatLonAlt": {"x": 0, "y": 0, "z": 0},

"assetBundles": [],

"landmarks": [],

"landmarkListUrl": "https://some.domain/landmarks.csv",

"panoramas": [],

"geoImages": [],

"geoData": {}

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*id | string | A unique name to identify the asset. |
| \*initialScale | double | The default scaling applied when this asset is loaded into the operator or client app. The scaling is applied uniformly. |
| conversionValue | double | Determines the conversion from units in the asset space to units in the flood app. By default this value is *1*. In the flood app, 1 unit = 1 meter. |
| lodMultiplier | double | Optional multiplier to uniformly scale the LOD space. LODs are only enabled on VisionOS, and only when provided with multiple 3D models. By default this value is *1*. |
| \*models | array | A list of ModelLOD type that contain the link(s) to the 3D model. Every element in this list represents one level-of-detail (LOD) of the 3D model. By default, only VisionOS makes use of multiple LODs, whereas any other platform simply loads the highest detail level. |
| \*originLatLonAlt | vector3 | A vector describing the geodesic coordinates of the origin of the 3D model. x = latitude, y = longitude, z = altitude. |
| \*minLatLonAlt | vector3 | A vector describing the geodesic minimum coordinates of the rectangle that makes up the 3D model. x = latitude, y = longitude, z = altitude. The x and y components of this vector should always be smaller than the x and y components of *“maxLatLonAlt”*. |
| \*maxLatLonAlt | vector3 | A vector describing the geodesic maximum coordinates of the rectangle that makes up the 3D model. x = latitude, y = longitude, z = altitude. The x and y components of this vector should always be larger than the x and y components of *“minLatLonAlt”*. |
| assetbundles | array | DEPRECATED  A list of type AssetPlatformLink. Each element describes a Unity AssetBundle per supported platform. Will not be used when *“models”* is not empty. |
| landmarks | array | A list of type SerializableLandmark. Each element describes a landmark; an annotation in 3D space with a text label and/or an icon. |
| landmarkListUrl | string | A URL to an optional CSV file that contains landmarks. The CSV should be comma delimited and the URL needs to be a public download-able link. |
| panoramas | array | A list of type SerializablePanorama. Each element describes a panorama; A point in 3D space that the client app can ‘jump-into’ to transition to virtual reality (VR) mode. |
| geoImages | array | A list of type SerializableGeoImage. Each element describes a geoImage; An image that is projected onto its associated 3D model. |
| geoData | object | An object of type GeoData that describes optional geometry to be generated, separated by (time) steps. For example, interactable flooding is generated using geoData. |

## 3D models – Level-of-detail (LOD)

At the heart of the Flood simulation app is the 3D model that is loaded onto a marker in Augmented Reality (AR) mode. This model is of type ModelLOD and has the following configuration:

{

"url": "https://some.domain/path/model.glb",

"originLatLonAlt": {"x": 0, "y": 0, "z": 0},

"lodLevel": 1,

"triangles": 11671,

"screenVisibleFraction": 0.0

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*url | string | A URL to the 3D model. The model can be of type GLTF or GLB and the URL needs to be a public download-able link. |
| originLatLonAlt | vector3 | Optional geodesic coordinates of the model’s origin in case it does not coincide with the origin of the parent asset. |
| \*lodLevel | int | A number describing the detail level of the 3D model unique to the *“models”* array. Higher numbers denote more detail and a higher triangle count. In case of a single LOD, this number will be ignored. |
| \*triangles | int | The number of triangles that make up the 3D model. This value is used on VisionOS platform for performance enhancement. In case of a single LOD, this number will be ignored. |
| screenVisibleFraction | double | A number between 0 and 1 that describes the fraction of screen space that should be occluded by the 3D model to enable this LOD level. Higher levels of detail should have higher fractions. This value is optional, and when not specified, calculated as:  *1 - (1 / numLODs \* (LODindex + 1))* |

### LODs on VisionOS

Due to hardware performance issues and to prevent subsequent crashes, we limit the LODs for devices on VisionOS (i.e. Apple Vision Pro) to < 1.000.000 triangles. This is a rough estimate to the amount of allocatable RAM memory on Vision Pro. While loading 3D models, the LODs are ordered by their detail level (lowest detail first) and if the cumulative triangle count of the LOD is lower than 1 million, it is added to the loading list. Otherwise, that LOD is skipped.

### LODs on iOS, Android, Windows, MacOS

On any other platform than VisionOS, only the 3D model with the highest level of detail is loaded. While loading 3D models, the LODs are ordered by their detail level (highest detail first) and the first is added to the loading list, while the others are ignored.

The following is an example configuration for 3 LODs:

"models": [

{

"url": "https://some.domain/Model-LOD-1.glb",

"lodLevel": 1,

"triangles": 11671,

},

{

"url": "https://some.domain/Model-LOD-3.glb",

"lodLevel": 3,

"triangles": 425451,

},

{

"url": "https://some.domain/Model-LOD-4.glb",

"lodLevel": 4,

"triangles": 2789650,

},

]

## Landmarks

Landmarks are annotations in 3D space during AR mode. These annotations have a text label and can have multiple icons as defined by the landmark type ID.

{

"label": "木津川市立木津中学校",

"icons": [1, 3, 4],

"position": {"x": 0, "y": 0, "z": 0},

"level": 0,

"referenceAssetId": "Kizugawa"

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*label | string | A text label that will be visualized in 3D space. If empty, the landmark will be skipped. |
| icons | array | A list of integers that refer to *“landmarkTypes”* ID property. For each matched landmark type, a small image will be displayed below the text label. |
| \*position | vector3 | A vector describing the geodesic position of the landmark. x = latitude, y = longitude, z = altitude. |
| level | int | Optional level that determines visibility and font size. Default = 0.  Level 1: <1.5 meters, 20pt font size.  Level 2: <0.9 meters, 18pt font size.  Level 3: <0.5 meters, 16pt font size.  Any other level: Always visible. 20pt font size. |
| referenceAssetId | string | The name of an asset in the configuration. When this optional property is not empty, a region indicator will be visualized at the reference asset’s origin, with size determined by *“minLatLonAlt”* and *“maxLatLonAlt”.* The landmark text label will appear at the origin, icons will be ignored, and a red border will show. When interacting with the region indicator on the client app, the reference asset will be loaded. |

### Landmarks from an external CSV

When “landmarkListUrl” in the asset configuration contains a valid download link to a CSV file, landmarks will be loaded according to each line in the file. The layout of the CSV file is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **label** | **latitude** | **longitude** | **type** |
| 萩の谷樋門 | 34.78348632 | 135.8064978 | 7 |
| 棚倉小学校 | 34.78226542 | 135.8115007 | 1;2;4 |
|  | 34.78688364 | 135.8152759 | 8 |

Each line in the CSV needs to end in a newline character and have *exactly 4 comma-delimited values*. Otherwise, it is skipped. Numbers in the “type” column denote landmark type IDs and should be *separated by a semicolon*. Both “label” and “type” can be empty, but not both.

## Panoramas ­­– VR points

A panorama or VR point is a location in 3D space that links to an VR experience when interacted with. It is called ‘panorama’ because this feature requires equirectangular panoramic, very wide, images as input.

Equirectangular panorama from Google Street view; (left) color (right) depth

A street with buildings and a bike lane

Description automatically generated A low angle view of a city

Description automatically generated

The flood client app takes these images and constructs a VR experience with them. The client is able to look around as if they are standing at the origin of the panoramic image. Movement in this VR mode is disabled, and when no depth image is provided it is automatically approximated by an AI[[1]](#footnote-1) neural network[[2]](#footnote-2). The depth image is used to construct a physical mesh of the environment, which can then be interacted with. It is required for the simulation of flood water in VR mode. The quality of the VR experience is dependent on the quality of the input images. It is strongly recommended to provide depth images as the AI estimation based on the input color panorama will often be of lower quality.

A panorama is defined in the configuration file as follows:

{

"position": {"x": 0, "y": 0, "z": 0},

"offsetInVR": {"x": 0, "y": 0, "z": 0},

"spherical360ImageUrl": "https://some.domain/color.png",

"depthImageUrl": "https://some.domain/depth.png",

"rotationFromNorthDegrees": 0.0,

"invertDepth": **false**

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*position | vector3 | A vector describing the geodesic position of the VR point. x = latitude, y = longitude, z = altitude. |
| offsetInVR | vector3 | A vector containing an optional translation from the panoramic origin in VR mode. This is Unity-space, e.g. moving the VR viewpoint one meter down:  *“offsetInVR”: {“x”:0, “y”:-1, “z”: 0}* |
| \*spherical360ImageUrl | string | A URL to the panoramic color image. The image can be of type PNG, JPG, GIF and the URL needs to be a public download-able link. |
| depthImageUrl | string | A URL to the panoramic depth (single channel) image. The image can be of type PNG, JPG, GIF and the URL needs to be a public download-able link. |
| rotationFromNorthDegrees | double | The degrees rotation around the world-up axis that needs to be applied in order to face north in the panoramic image. This optional value is only used to align equirectangular images, such as Google Street View, with the 3D map model. Defaults to *0* degrees. |
| invertDepth | bool | Only applies when “depthImageUrl” is provided. A boolean value denoting whether to invert the logic for converting depth image pixel values to distance. Default to *false*.  *false* = 0 is closest, 255 is furthest  *true* = 0 is furthest, 255 is closest |

### VR panorama in conjunction with GeoData

When GeoData is provided, that data will be interpreted as flood data by the system (see GeoData – Flood visualization). In VR mode, flood water will be simulated using the input equirectangular color and depth images based on the current height of the flood water at the input latitude and longitude. When there is flood water at the time of activating the VR point, a panel with information about the flood water will be displayed.

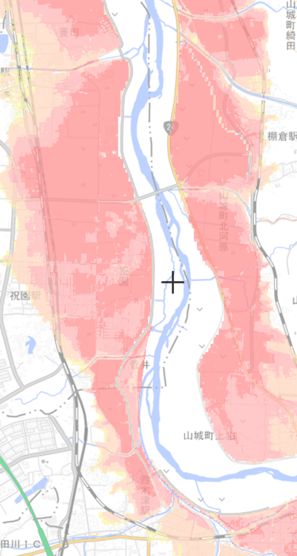
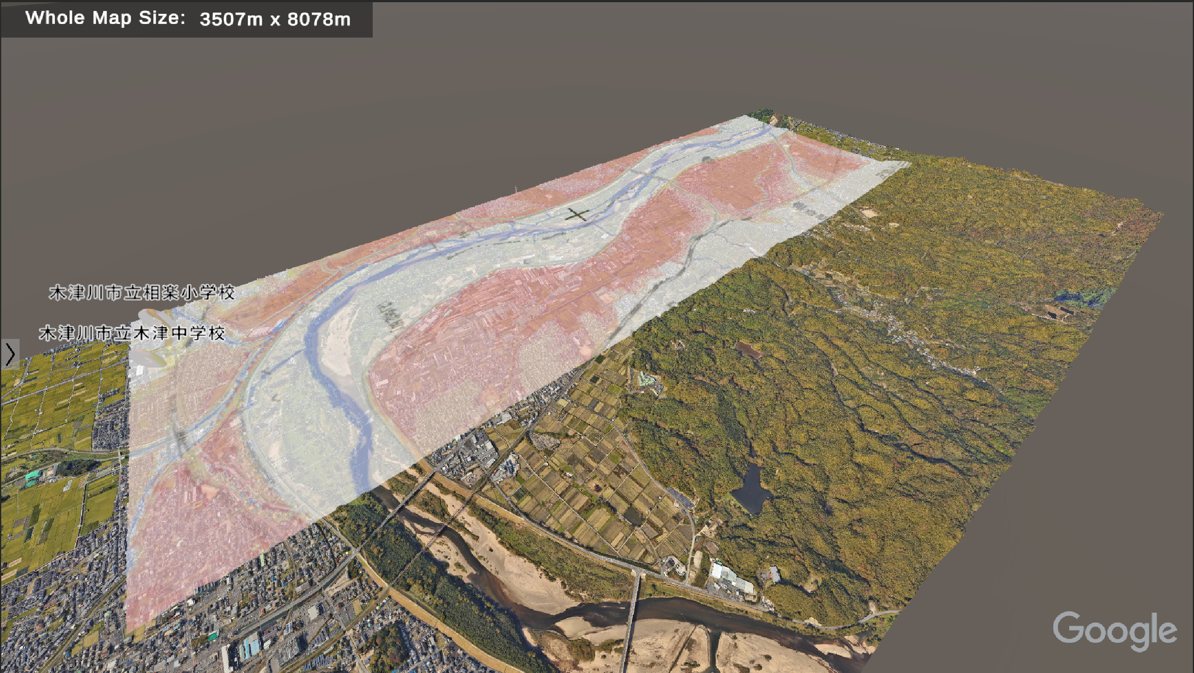
Flood result in VR mode based on the example input from Google Street View.

## GeoImage – Image projection

It is possible to overlay color images onto the loaded 3D model in AR mode. This creates a kind of projection of the image on top of the geometry of the map, for the purpose of annotating regions on the map. This is referred to as a GeoImage.

A flood area image (left) is projected onto a 3D map (right) in AR mode.

A GeoImage like the above is created with the following example configuration:

{

"id": "洪水浸水想定区域",

"minLatLon": {"x": 34.735018, "y": 135.786381, "z": 0},

"maxLatLon": {"x": 34.787338, "y": 135.820026, "z": 0},

"imageUrl": "https://some.domain/projection-1.png",

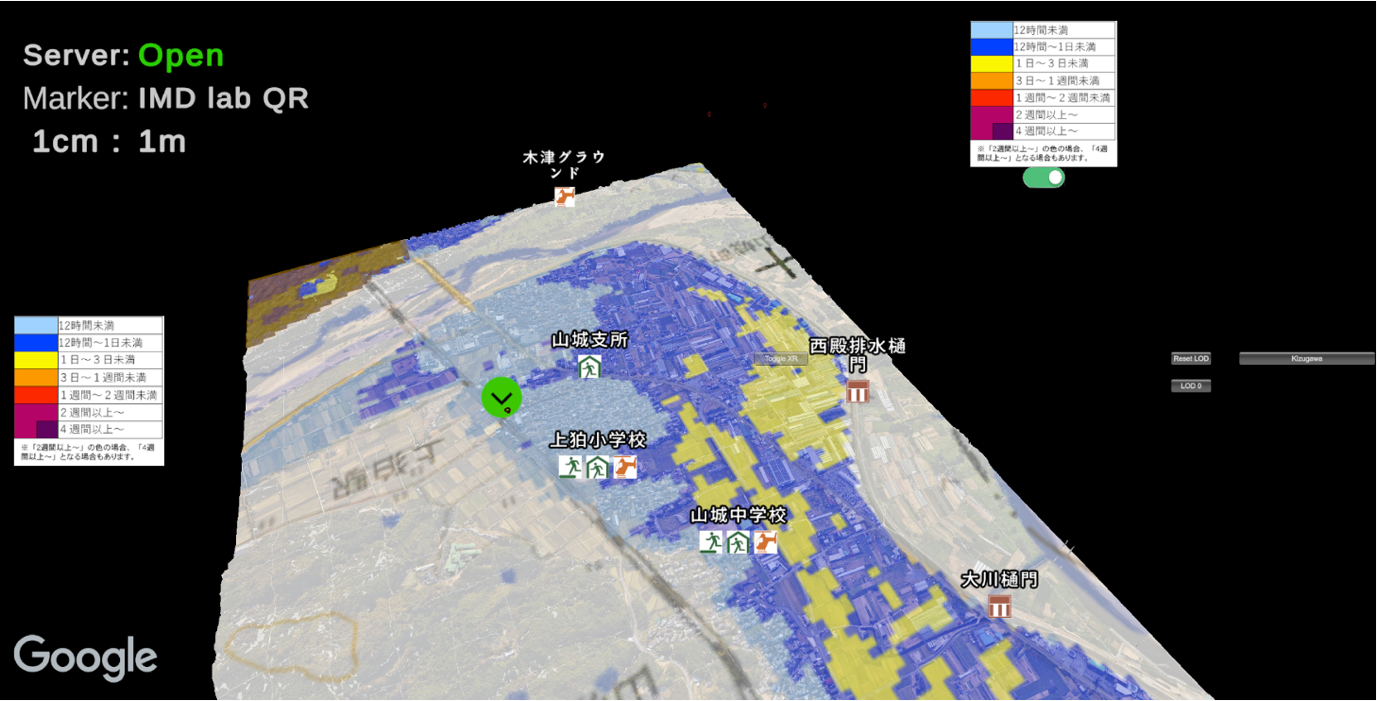
"legendImageUrl": "https://some.domain/legend-1.png",

"opacity": 0.5

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*id | string | The unique identifier for the geoImage. This *id* will also be used as the label for the geoImage. |
| \*minLatLon | vector3 | A vector describing the geodesic minimum coordinates of the rectangle that makes up the image. x = latitude, y = longitude, z = altitude. The x and y components of this vector should always be smaller than the x and y components of *“maxLatLon”*. |
| \*maxLatLon | vector3 | A vector describing the geodesic maximum coordinates of the rectangle that makes up the image. x = latitude, y = longitude, z = altitude. The x and y components of this vector should always be smaller than the x and y components of *“minLatLon”*. |
| \*imageUrl | string | A URL to the projection image. The image can be of type PNG, JPG, GIF and the URL needs to be a public download-able link. |
| legendImageUrl | string | A URL to an optional legend image that supports the projection image. The image can be of type PNG, JPG, GIF and the URL needs to be a public download-able link. The legend will appear on the left-side of the screen of the client app on handheld devices, or next to the longest side of the 3D map on HMD devices. |
| \*opacity | double | A number between 0 and 1 that denotes the opacity of the projection image, where 0 will be fully transparent and 1 will be fully opaque. Transparency in the input projection image is also supported, and will be multiplied by this value. |

An optional legend is shown on the left (handheld devices) or in 3D space (HMDs).



## GeoData – Flood visualization

The geoData property of an asset is used to construct and visualize geometry in conjunction with the 3D model. This geometry can be divided into multiple (time-) steps and can be interacted with by the client. The primary use-case of geoData is to visualize flood water and its properties (area at the time step, depth, speed, etc.). Below is the full configuration of geoData:

"geoData": {

"dataUrls": [

"https://some.domain/flood-time-1.csv",

"https://some.domain/flood-time-2.csv",

],

"fileStartIndex": 0,

"scale": 1.0,

"delimiter": ",",

"numColumns": 0,

"elevationColumnIndex": 0,

"dataValueColumnIndex": 0,

"latLngShapeStartColumnIndex": 0,

"xDegreesOffset": 0.0

}

|  |  |  |
| --- | --- | --- |
| **Property** | **Type** | **Description** |
| \*dataUrls | array | A list of strings, with each element being a URL to a CSV file that contains geometry data. See below for CSV format. |
| fileStartIndex | int | A number between 0 and the length of the *“dataUrls”* array (exclusive) at which the geometry visualization will start. By default this is 0. |
| scale | double | Optional scaling that will be applied to all geodesic values when constructing the geometry. Default to 1. |
| delimiter | string | The delimiter used by the CSV files in *“dataUrls”*. Normally, CSV files are comma-delimited, and “,” is the default value. Use this field if your input data is separated by a different character (e.g. “;”, “|” etc.). |
| numColumns | int | DEPRECATED  A positive number describing the number of columns in the CSV files. This value is ignored. |
| \*elevationColumnIndex | int | The (zero-based) index of the column in the CSV files which contains the altitude/elevation value of the geometry quad. |
| \*dataValueColumnIndex | int | The (zero-based) index of the column in the CSV files which contains the data value used for visualization. For flood it contains depth. |
| \*latLngShapeStartColumnIndex | int | The (zero-based) index of the starting column in the CSV files which contains the geometry quad geodesic points. The system expects 4 geodesic points pairs (latitude, longitude) therefore the CSV file should contain columns of this index + 8. |
| xDegreesOffset | double | An optional rotation around the X-axis in degrees that should be applied to the constructed geometry. |

### CSV file format for Flood data

The CSV files that are references by “dataUrls” should at least contain the following columns. The column titles do not matter and are skipped, but the order of the column is fixed. First, “dataValueColumnIndex”+1 is used for a secondary data value (flood speed). Secondly, the order of the quad geodesic positions (P1, P2, P3 and P4) are expected in ascending order, with each geodesic position first Latitude then Longitude:

GeoData CSV file format.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **標高** | **浸水深** | **流速** | **P1経度** | **P1緯度** | **P2経度** | **P2緯度** | **P3経度** | **P3緯度** | **P4経度** | **P4緯度** |
| 48.774 | 0.004 | 0.012 | 135.79 | 34.63625 | 135.7903 | 34.63625 | 135.79031 | 34.63645 | 135.79 | 34.63645 |
| 48.795 | 0.019 | 0.068 | 135.79 | 34.63625 | 135.7903 | 34.63625 | 135.79031 | 34.63645 | 135.79 | 34.63645 |

Column index:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

In the above example CSV file, the following would be the correct configuration:

"elevationColumnIndex": 0,

"dataValueColumnIndex": 1,

"latLngShapeStartColumnIndex": 3,

Example output of GeoData; visualizing flood inundation with discrete depth steps.



1. Barracuda <https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/index.html> [↑](#footnote-ref-1)
2. MiDas 2.1 <https://github.com/isl-org/MiDaS> [↑](#footnote-ref-2)