IDM-3D.Geometry

Ekkehard Beier (Editor)

Version

Specification: IDM-3D.Geometry 1.2
Document Version: Final Version

Specification

IDM-3D.Geometry

IDM-3D.Geometry is a component part of the IDM-3D specification and describes the exchange of geometric data based on 3D meshes[1]. Here, the focus is on the master-data level. The dynamic data level (exchange or order data) may differ from this and focus on efficient formats, for example from IDM-3D.Geometry or generate other formats (and possibly with a different granularity), for example glTF or USD[z].

Preferably, an IDM-3D.Geometry data record is a complete compilation of the geometries of a project, whereby the assignment of a project to a manufacturer, a product series, a part-product series, several product series [at a commercial level] is not defined. This is because it may, for example, be expedient to combine geometries from different manufacturers in one project.

IDM-3D.Geometry allows the following application scenarios, among others:

- Provision of the geometries by the service provider for another service provider or user as a starting point for the commercially configured assembly of the products
- Provision of the geometries by the user to a service provider or sales partner for further use (e.g. CGI generation)

At the material level, the equivalent of IDM-3D.Geometry is IDM-3D.Material. IDM-3D.Geometry and IDM-3D.Material will be the building blocks for further IDM-3D specifications.

[1] Currently, the focus is exclusively on mesh geometries. At a later time, this may be extended towards parametric solids (CAD/BIM) or with a different focus.

Geometry

In principle, a geometry is a basic entity from the point of view of a structured, parametric 3D product representation, the existence of which is determined by the following criteria:

- The geometry can receive a direct material assignment.
- The geometry can be controlled via the commercial configuration (visibility, position, scaling where necessary, etc.).
- The geometry is addressed in the context of a functional representation (a.k.a. animation).

The [parametric] structure definition and the runtime animation control may be the subject of other IDM-3D standards that are based on IDM-3D.Geometry. This also applies to the description of deformations, i.e. the possibility of manipulating geometries on the vertex level. The deformation-relevant formats will then be defined in a later version of IDM-3D.Geometry.

Because the geometry is a basic entity by definition, it can only be assigned with precisely one material.

A real product is not mapped canonically onto a set of geometries. In principle, however, the above criteria should be observed. In general, a high degree of structural agreement with real components should be aimed for.

A geometry is therefore understood to be a basic building block for visual product representations that can be realized by alternative base or native representations. In principle, this involves two dimensions:

- Native representations in different formats [of the same quality] can be provided for a geometry.
- Native representations in different qualities [of the same format] can be provided for a geometry.

The term ‘base geometry’ describes a native representation that is part of an abstract geometry entity as defined above. A geometry can exist without base geometries - e.g. one that only defines parameters; however, a base geometry cannot exist without geometry in this context.

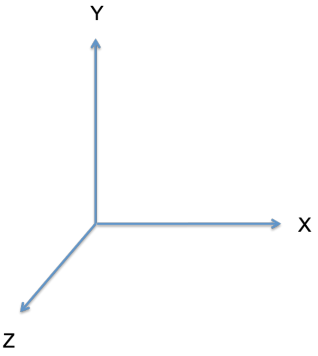
In terms of formats, a distinction is made between:

- Mesh-based geometries
- Parametric geometries
- Other formats (maps, deformations, etc.)

As a minimum, a base geometry defined via a mesh format contains vertex coordinates and triangles/polygons. Further information may include texture coordinates and vertex normals. A geometry is configured via an IDM-

3D.Material material, and therefore does not contain material properties. In this respect, material files that may exist in the context of the native geometry data defined below should be ignored. Likewise, non-geometric information that the native format may allow - such as light sources, camera definitions, animations, etc. - should be ignored.

The standard unit of base geometries is 1 meter. This means that 0.5 is 0.5 meters. This applies both to vertex coordinates and to non-object-related texture coordinates. The underlying coordinate system is defined as follows:



- X axis: to the right [increasing values]
- Y axis: upward
- Z axis: to the front

The coordinates of a base geometry must be retained. Automatic alignments are therefore not permitted. Moreover, base geometries of various formats/qualities must be aligned identically.

A geometry has an identifier that must remain unique within the context of a project. The full identifier of a geometry is made up of a 3-level name, whereby the first level contains a global identifier and the second level clearly describes the project within the context of the first level. For the geometry identifier, i.e. the third level, the following applies:

- The characters A - z, 0 - 9, _ and - are permitted.
- A project may not include two geometries that are identical in terms of a case-tolerant comparison.

Note: The specifications for the first and second identifier level are defined elsewhere and are, in principle, related. However, the first character may not be a number or a hyphen.

Geometry identifiers should be readable as far as possible (e.g. through the use of camel case or underlining) and be selected so as to relate to the problem domain.

Mesh formats

The following alternative mesh formats, which differ significantly in terms of functional power, scope and usability (reverse engineering) will be defined. This enables a decision to be made individually as to which format(s) should be distributed in a specific case.

Wavefront OBJ

OBJ [OBJ] is a polygonal[2] mesh format with an optional UV set.

Here, the vertex and polygon information, i.e. vertex coordinates, normal vectors, texture coordinates and the definition of the polygons, have to be defined. Moreover, smoothing groups are to be supported. The number of points per polygon is not limited to 3. Furthermore, the polygons have to be convex and planar.

Further information is not supported. This applies in particular to:

- Material libraries and names
- Spline curves and surfaces, smoothing groups
- Trim curves
- Level-of-detail information
- Ray-tracing and shadow-casting information

If the UV set is available, this is to be used for assigning the material.

If a geometry contains mesh formats, the OBJ format must be available as a minimum.

The file extension is *obj*.

OpenCTM

OpenCTM [OPENCTM] is a loss-free compressing triangle format that can contain any type of vertex information. In the context of this specification, this information is to be defined as follows:

- The optional 'Material' (alternative: 'Diffuse') texture coordinate set describes an endlessly repeating grid for assigning the material.
- The optional 'Object' texture coordinate set describes the mapping of the triangles onto a non-repeating, two-dimensional normalized value rant that is assigned to either one geometry or to several geometries.

The file extension is *ctm*. Only version 1.03 is to be used.

[2] At least, this is the standard form and the one used here.

FBX

FBX [FBX] is a complex format that, in this context, may only be used for describing meshes. FBX can contain multiple texture coordinate sets. In this regard, the following should apply:

- The first texture coordinate set (if available) describes an endlessly repeating grid for assigning the material.
- The second texture coordinate set (if available) describes the mapping of the triangles onto a non-repeating, two-dimensional normalized value range that is assigned to either one geometry or to several geometries.

In the context of IDM-3D.Geometry, the following determinations and limitations apply to FBX:

- The binary version of Version 7 (or later) is be used.
- If included, materials and animations will be ignored.

The file extension is *fbx*.

Deformationen

Deformations are data entities that may applied to Geometries at run-time, and which are especially useful to describe deformations in an efficient, dynamic and parametric way. Depending on the specific type of the deformation, they can be used also in Animations.

The currently supported approaches for Deformations are:

- Simplified replacement geometries, such as wrapping blocks (in the following: lattice), with explicitly described deformations that will be applied indirectly to the actual geometry.
- Explicit deformation versions of the geometry

The explicit approach is resolution-dependent, i.e., both geometry (mesh) and deformations must be provided for specific resolutions, if needed. In the context of the IDM-3D.Geometry specification, alternative deformation approaches are provided to be able to adapt to the technical progress.

In case, the run-time environment does not (or just partially) support Deformations, the corresponding geometries of the IDM-3D Geometries should be displayed in general. Consequently, IDM-3D Data should prioritize the un-deformed state. That means, they should not assume deformation support to get into a correct initial state. This may be disadvantageous from the modeling perspective.

Primarily, the Deformation will be described in a JSON file that must provide the following information:

Algorithm: "IG1" | "Blender1" | "Keys1"

The parameter defines the algorithm and possibly further files.

- "IG1" – A lattice deformation based on Mean Value Coordinates [FFD], will be applied.
- "Blender1" – A lattice deformation based on the algorithm of the modeling software blender, will be applied.
- "Keys1" – The deformation is based on explicit variants, which are described in an additional file.

Animation: "First" | "All" = "First"

In case of the "Keys1" algorithm, this parameter defines if only the first or—if given—all contained animations should be interpreted.

Type: "XYZ" | "XY" | "YZ" | "YZ" = "XYZ"

The parameter defines for the algorithms "IG1" and "Blender1", which coordinates will be provided for the description of the lattice. If the value is not "XYZ", the missing dimension must be set automatically at run-time, in a way that it entirely enclosed the in-deformed geometry including an offset to avoid rounding artefacts.

Dimensions: {"X": Integer, "Y": Integer, "Z": Integer}

The parameter defines for the algorithms "IG1" and "Blender1" for each dimension the number of the points. If the Type is not equal to "XYZ", the corresponding dimension value will be ignored and thus can be omitted.

Vertices: {<t>: {"X": Number, "Y": Number, "Z": Number}[]}

For the algorithms "IG1" and "Blender1", the parameter specifies the lattice vertices. Here, <t> is used for a time value in the range from 0.0 to 1.0, which however must be specified as string.

The number of the array elements is given by the multiplication of the single values of Dimensions, with unspecified dimensions assumed as 1.

The vertices will be interpreted in the following order: Z (from back to front) – Y (from top to bottom) – X (from left to right).

The deformation description file is named *deformation.json*. The alternative protobuf variant has the name *deformation.ffd.*

In the context of the explicit geometry deformations (algorithm: "Blender1") an additional geometry file with format glTF resp. GLB [GLTF] is expected. This file contains the geometry (with up to 2 texture-coordinate sets) as well as one ore more animations, to be mapped to the range 0 – 1. Here, all kinds of glTF animations needs to be supported: key frames, bones and displacements. All

other glTF content (such as materials, light sources, cameras) needs to be ignored.

Other formats

Normal maps

A normal map describes a manipulation of the normal vector resulting from the geometry at a specific point (determined via the respective texture coordinates). This is performed via an image file; the dimensions of which are powers of two. As with the mesh function, there may be a number of different permutations (e.g. different qualities).

The normal maps are specified as tangent normal maps (light blue is vertical).

The file extension is *jpeg/jpg* (JPEG) or *png* (PNG).

Geometry qualities/modes

The following qualities/modes are defined.

Standard

Standard quality is designed for mobile real-time use. Here, fast loading times and low storage requirements are prioritized over display quality. This can be achieved through manual or automatic mesh reduction and/or the use of normal maps.

The file name[3] is *standard* for mesh files and *normals_standard* for normal maps.

Guideline values[4] for mesh data with the standard quality are defined as follows: (reference: uncompressed OBJ, including texture coordinates):

- Larger objects (e.g. entire bed frames, entire bed headboards): 500 KB or 20,000 triangles
- Medium parts (e.g. armrests, seat cushions): 200 KB or 10,000 triangles
- Small parts (e.g. handles, bed feet): 50 KB or 200 triangles

For normal maps, the pixel size is 1024 or 2048.

High Resolution

High Resolution quality describes the minimum fully comprehensive geometric representation. The primary purpose is high-quality image generation using ray tracing/radiosity methods.

---

[3] in the following, always to be interpreted in the same way as file extension
[4] Hard limits should be avoided. However, statistical compliance with the specification is important for uniform processing criteria (memory requirements, download/loading times).

In general, geometry-replacing data structures such as normal maps are not to be used here. However, data of this quality should be minimal, i.e. not unreasonably extensive. This applies, for example, to data that is generated from a 3D scanner.

The guide value for the size is factor 20 applied to the size of the corresponding file with Resolution Standard, and an absolute limit of 20 MB for each file.

It is legitimate to create a normal map for this quality. However, in contrast to the mesh, there is no fallback to a normal map with a lower resolution if it is not available.

The file name is *highres*.

Low Resolution

Low Resolution quality is an optional, heavily reduced representation of the Standard resolution. The primary purpose is to provide a quick preview. Normally, the Low Resolution quality is then replaced by the Standard quality. In this respect, clearly visible deviations/simplifications are legitimate. Texture coordinates can be left out.

As a guideline value, the Low Resolution quality is specified as being an 80% reduction of the Standard quality.

The file name is *lowres*.

Collider

Collider quality is an optional, extremely reduced representation of the geometry. The primary purpose is internal, for rapidly determining collisions. With regard to the simplification, it should be noted that concave geometries are reproduced sufficiently to prevent artifacts being generated due to the incorrect detection of collisions.

The file name is *collider*.

Source

The Source mode identifies a source file and can be used to enable consistent storage of source files with the derived formats where necessary.

The file name is *source*. A description of the source formats permitted is not given.

Geometry parameters

As an option, a geometry can define parameters that control its interpretation in the processing context.

Complexity: int = 100

The parameter specifies the complexity of a building block, in order to implement quantitative tests. Such tests should be applied to evaluate data to consider their useability in mobile scenarios.

The default value corresponds to an intermediate building block, as defined above. The smaller (higher) the spatial size of the building block, the smaller (higher) it's complexity. The simpler (more sophisticated) the building block, the smaller (higher) it's complexity.

NormalMap: string

As an alternative to a normal map as an included part of a geometry, a normal map can exist autonomously and be used by several geometries. This can be realized via the NormalMap parameter. Here, the normal map can be specified as being fully or non-qualified. The non-qualified version is to be preferred, because the data set is then more flexible in use. The normal map is specified without a specific file name, because the actual resolution is only specified at runtime.

If both local normal map files and a map defined via geometry parameters exist, then the latter takes precedence.

NormalMapStrength: float = 1

This parameter controls the strength of a normal map. It is considered as a direct parameter for the calculation.

NormalMapTiling: bool = false

This parameter controls the interpretation of the texture coordinates to be used for the normal map, outside the range [0, 1] as follows:

- false – Values greater than 1 (smaller then 0) are mapped to 1 (0) (‚clamp').
- true – Values outside [0, 1], are mapped into the range [0, 1] (‚repeat').

PatchSize: float

The parameter specifies the size of the texture coordinates that is used for the material assignment, in Meter. The parameter can be used for an automatic adaptation of materials, which themselves provide such information.

Information file

The information file contains additional information on the geometry.

The respective optional entries are:

- *Name* – The name of the geometry can be specified as an option to avoid inadvertent changes (e.g. case deviations). If a name is specified, this has priority over the name that is derived from the directory name. If the name is not specified, a name will be derived from the directory name. (see below)
- *Description* – This entry contains an informative, single line description of the geometry.
- *Parameters* – This entry contains the geometry parameters. The permitted JSON dictionary keys and the respective values are generated via the above defined geometry parameters. Every parameter of the type int, float or bool can also be provided as string. In case of float, the string must be provided in scientific notification. In case of bool, the possible values are True/true and False/false.

The file name of the information file is *info.json*. The coding of the file must be identical in ASCII and UTF-8 (without BOM). Consequently, the description should not contain any special characters.

Index file

The optional index file is not contained within the geometry directories.

The primary purpose of the index file is to clearly describe the geometries required for a project so that a service provider has an appropriate starting point. If both an index file and geometry information files exist in a project, the latter have priority.

The file contains a JSON dictionary whose keys are the geometry names. The structure is specified for each value in the same way as specified for the information file. The following entry can also be specified as an option:

- *Alignment* – This entry is an informative description of the alignment of the geometry. If an entry is not specified, then "left-down-back" applies with respect to the local origin of the geometry relative to the axis-orthogonal boundary volume of all vertices. In almost all cases, this alignment is practical for the later parametric assembly of products. However, for axially or rotationally symmetrical components it may be expedient to select a different alignment, e.g. 'centre-down', and to specify this accordingly via Alignment.

The file name of the index file is *index.json*. The coding of the file must be identical in ASCII and UTF-8 (without BOM). Consequently, the description should not contain any special characters.

Physical representation

Index

The optional index file is located parallel to the geometry directories on the top level of a distribution (e.g. ZIP file).

Geometry files

One directory exists per geometry. The name of the directory is derived from the name of the geometry to which the extension *geo* is appended. If, however, a split normal map is involved, then the extension is *tex*.

The files defined above are located in the directory, whereby their names are derived from the specified prefix and the format-specific extension.

As an option, a unique content hash can be assigned per file which can be used for optimization purposes. The available hashing methods are:

- MD5 (file extension: md5)

The name of the hash file is derived from the name of the respective file and the extension[5] *md5*. The file content is the hash value without line break.

Note: The hash does not necessarily have to correspond to the file content. For example, the hash of reduced-resolution files can refer to the original resolution.

References

[FBX] https://docs.fileformat.com/3d/fbx/
[FFD] https://www.mn.uio.no/math/english/people/aca/michaelf/papers/mv3d.pdf
[GLTF] https://github.com/KhronosGroup/glTF
[OBJ] https://docs.fileformat.com/3d/obj/,
https://www.fileformat.info/format/wavefrontobj/egff.htm
[OPENCTM] http://openctm.sourceforge.net

History

IDM-3D.Geometry 1.2 – 2023-03-21

- Specification of size for High-Resolution Geometries
- Specification of Deformations

IDM-3D.Geometry 1.1 – 2023-01-10

[5] The extension will thus be appended - e.g. *standard.obj.md5*

- Specification the of geometry-parameter types float, int and bool
- New geometry parameters: Complexity, PatchSize, NormalMapStrength and NormalMapTiling

IDM-3D.Geometry 1.0 – 2022-02-17