Specification

OMATS
OFML compatible Materials*

Version 2.1

Status: Release

Thomas Gerth, EasternGraphics GmbH (Editor)

June 27, 2023

Contents

References

[jfif] JPEG File Interchange Format, Version 1.02
World Wide Web Consortium (W3C)
(www.w3.org/Graphics/JPEG/jfif3.pdf)

[odb] ODB – OFML Database (OFML Part I), Version 2.4.
Industrieverband Büro und Arbeitswelt e. V. (IBA)

[ofml] OFML – Standardized Data Description Format of the Office Furniture Industry, Version 2.0, 3rd revised edition
Industrieverband Büro und Arbeitswelt e. V. (IBA)

[png] Portable Network Graphics (PNG) Specification, Version 1.2
PNG Development Group
(www.libpng.org/pub/png/spec/1.2/png-1.2-pdg.html)

1 Introduction

This specification defines two material models which are used in OFML based applications for the representation of object surfaces (materials), both in the real-time and in the photorealistic domain.

Furthermore, this specification describes the supported texture mapping methods as well as the mapping of abstract model parameters into OFML material definition files.

The two material models are referenced as OMATS1 and OMATS2. OMATS1 is the older model. In newer OFML applications it is replaced by the new model OMATS2, which uses the concept of *physically based rendering* (PBR). With a more compact material description, this model allows a more realistic and appealing representation in real-time mode (see appendix A for more information). In addition, material editors based on this model offer a better usability, as fewer parameters need to be set and there are fewer dependencies between the parameters.

To ensure downwards compatible processing of materials, the following terms apply to OFML applications which use the new model:

- Materials created on the basis of OMATS1 automatically are converted to the new model (see also appendix B).
- Contrary to the previous point, when exporting an OFML material definition file, missing parameters for the processing according to OMATS1 are derived from the parameters for OMATS2 (and also exported).

2 The material models

2.1 Overview

Each model defines a set of parameters describing specific characteristics of a material.

The parameter *Material_Type* (s. 2.3) plays a special role: depending on the selected material type, only certain material parameters are used during rendering.

The following table provides an overview of the defined material parameters (in alphabetical order of their identifiers) and their correlation with the material types and the two models.

All parameters are optional, i.e. need not be specified in a material description[1].
If a parameter is not specified, a predefined value is used for parameters for model OMATS1. This value is specified below in the description of each parameter.
For parameters defined only for model OMATS2, no value is predefined. Rather, if the parameter specification is missing, the value is derived from OMATS1 parameters (or from their default values)[2].

[1] Theoretically, this allows empty material descriptions.
[2] The procedure for this derivation is undefined and may vary from application to application.

Parameter	Material type			Model	
	Common	Glass	Illuminant	OMATS1	OMATS2
Base_Color	X	X		X	X
Base_Color_Map	X	X		X	X
Emissive_Color			X	X	X
Luminance			X	X	X
Metallness	X				X
Metallness_Map	X				X
Normal_Map	X	X		X	X
Refractive_Index		X		X	X
Roughness	X	X			X
Roughness_Map	X	X			X
Shininess	X			X	
Sound_Absorption	X	X	X	X	X
Specular_Color	X			X	
Specular_Factor	X			X	
Transparency	X	X	X	X	X

2.2 Used data types

The following data types are used in the description of the parameters:

`PI` positive integer

`FP` floating-point number

`RGB` Vector of three color values representing the base colors red, green and blue

Each color value C must be in the range $0.0 \leq C \leq 1.0$.

`RGB-IMAGE` image file

This data type describes two-dimensional image files consisting of RGB color values.

The following formats are allowed: PNG, JPEG

The dimensions of the image files should be powers of two.

The maximum size allowed is 4.096 x 4.096, but in general textures should only be resolved as high as necessary.

Depending on the nature of the material, the following recommendations apply:

- 1.024 x 1.024 – very fine, highly structured materials
- 512 x 512 – "ordinary" materials
- 256 x 256 – simple, low-structured materials

The image files normally should be created in such a way that a repetition in both dimensions is visually appealing.

The naming of the image files is arbitrary.

`RGBA-IMAGE` image file with transparency

This data type represents an extension of type `RGB-IMAGE` and contains an additional transparency value. This either can be a scalar value or an explicit color value which digitally controls the transparency, i.e. texels[3] with this color value have a transparent representation.

The following formats are allowed: PNG

`GRAYSCALE-IMAGE`
grayscale image file

Unlike `RGB-IMAGE`, this type includes only one value per pixel[4].

The following formats are allowed: PNG, JPEG

`SYMBOL` symbolic identifier

Note regarding data types `RGB`, `RGB-IMAGE` and `RGBA-IMAGE`:
The sRGB color space is assumed for RGB color values.

Notes regarding image file formats PNG and JPEG:

Images in PNG format have to comply with the "PNG (Portable Network Graphics) Specification" [png]:

- have to be sequentially structured (non-interlaced/progressive)
- have to use the RGB color model in case of `RGB-IMAGE`
- have to use 8 bit for a (color) channel
- may not be animated

Images in JPEG format have to comply with the specification of the "JPEG File Interchange Format" [jfif]:

- have to be sequentially structured (non-interlaced/progressive)
- have to use *Huffman* coding (non-arithmetic coding)
- have to use the YCbCr color model[5]
- have to use 8 bit for a color channel

2.3 Material types

The material type (parameter *Material_Type* of type `SYMBOL`) is used to select an appropriate shader[6]. Furthermore, based on the material type, the available parameters may be restricted in a material editor.

The following types are defined:

- *Common*

  This is the recommended default type if a material can not or should not be correlated with any of the specific material types mentioned below.

- *Glass*

  This type should be assigned to all glass materials. If, instead, type *Common* is used, it may be the case that the material looks like a transparent plastic.

- *Illuminant*

  Type for self-luminous objects. Useful in conjunction with parameter *Luminance*.

[3] a pixel of a texture in 3D computer graphics
[4] If there is an image file with 3 color channels (RGB), the gray value of the color is used.
[5] When importing, color values are converted to the RGB color model
[6] Shaders are programs for calculating rendering effects, e.g. for the spatial perception of 3D models.

If this parameter is not specified, the application autonomously determines a matching shader using heuristics based on the (other) specified parameters (which then possibly might not deliver the desired results)[7].

2.4 The material parameters

Preliminary remarks:
The parameters are listed in alphabetical order. The identifier of the parameter is followed by the data type in square brackets as well as the corresponding material types and models in curly brackets.

- *Base_Color* `[RGB]` {*Common*, *Glass*, OMATS1, OMATS2}

  The base color is used to simulate the diffuse reflection characteristics of the object's surface. In the model OMATS2, for metals the parameter also serves to determine the color and intensity of the specular reflection.

  The predefined value is `1.0,1.0,1.0` (white).

- *Base_Color_Map* `[RGB-IMAGE, RGBA-IMAGE, GRAYSCALE-IMAGE]` {*Common*, *Glass*, OMATS1, OMATS2}

  The referenced image file is used as a compensating description for parameter *Base_Color*.

  Additionally, in the case of an `RGBA-IMAGE`, the transparency values resulting from the alpha channel are used as a compensating description of parameter *Transparency*.

  In the case of a `GRAYSCALE-IMAGE`, the (single) value is used for all 3 color channels.

  There is no predefined image file for this purpose.

  For details regarding the texture mapping methods see section 3.

- *Emissive_Color* `[RGB]` {*Illuminant*, OMATS1, OMATS2}

  Defines the color of the emitted light of a geometry based light source.

  The predefined value is `0.0,0.0,0.0` (black).

- *Luminance* `[FP]` {*Illuminant*, OMATS1, OMATS2}

  Specifies the luminance of a geometry based light source in cd/m^2.
  (*Candela* — *cd* — is the SI unit for the basic parameter *light intensity*.)

  The predefined value is `0.0`.

- *Metallness* `[FP]` {*Common*, OMATS2}

  In the real world, materials can be divided into metals and non-metals. Therefore, for most materials, this value should be `0.0` or `1.0`. Intermediate values are used to represent semi-metals or contaminated metals.

- *Metallness_Map* `[GRAYSCALE-IMAGE]` {*Common*, OMATS2}

  The referenced image file is used as a compensating description for parameter *Metallness*:
  Bright image areas receive metal characteristics, dark ones are interpreted as non-metal.

  There is no predefined image file for this purpose.

  For details regarding the texture mapping methods see section 3.

[7] For example, if the luminance value of the material is greater than `0` and the material is not textured, type *Illuminant* is assumed.

- *Normal_Map* `[RGB-IMAGE]` {*Common*, *Glass*, OMATS1, OMATS2}

  A normal map modifies the normal vectors of the surface in order to simulate the illumination of unevenness that is not present in the object geometry.

  The values in the referenced image file are interpreted as normalized normal vectors.

  There is no predefined image file for this purpose.

  For details regarding the texture mapping methods see section 3.

- *Refractive_Index* `[FP]` {*Glass*, OMATS1, OMATS2}

  The scalar value specifies the refraction of the light in the case of transparent materials. The refractive index refers to the ratio of the phase velocity of the light in vacuum to that in the respective material.

  Selected values are:

  – water: `1.33`
  – glass: `1.5` ... `1.9`

  The predefined value is `1.0` and corresponds to the refractive index of vacuum.

- *Roughness* `[FP]` {*Common*, *Glass*, OMATS2}

  The degree of roughness determines how smooth or rough a surface is. Depending on the degree, the reflected light is scattered more or less at the surface.

  The values are in the range of `0.0` to `1.0`.

- *Roughness_Map* `[GRAYSCALE-IMAGE]` {*Common*, *Glass*, OMATS2}

  The referenced image file is used as a compensating description for parameter *Roughness*:
  Bright areas of the image appear dull, dark areas appear glossy.

  There is no predefined image file for this purpose.

  For details regarding the texture mapping methods see section 3.

- *Shininess* `[FP]` {*Common*, OMATS1}

  The scalar value indicates the gloss for shiny surfaces. This is the integer exponent of the *cos term* according to the lighting model of *Phong*.

  As a rule of thumb: The larger this value, the smaller the gloss effect simulating the reflection of the light source.

  The predefined value is `30`.

- *Sound_Absorption* `[PI {PI FP}*]` {*Common*, *Glass*, *Illuminant*, OMATS1, OMATS2}

  The parameter is a set of value pairs specifying the sound absorption coefficient (2nd value) for different frequencies (1st value). In front of the value pairs the number of the pairs is specified.

  Usually, the sound absorption coefficient is given for the following frequencies:
  125Hz, 250Hz, 500Hz, 1000Hz, 2000Hz, 4000Hz.

  The sound absorption coefficient is a non-negative floating point number. Usually the value is in the range of `0.0` (no absorption) to `1.0` (complete absorption). But there can also be values slightly above `1.0`. This is possible if the actual effective surface of a sound-absorbing object is greater than the geometric surface that is used for calculating the acoustics.

Example: `6 125 0.1 250 0.3 500 0.2 1000 0.1 2000 0.5 4000 0.4`

If the parameter is missing in the material description, the corresponding object is not considered during acoustics calculation.

- *Specular_Color* `[RGB]` {*Common*, OMATS1}

  The specular color is used to simulate the specular reflection characteristics of the object's surface and also determines the color for highlights (Phong model).

  The predefined value is `0.0,0.0,0.0` (black).

- *Specular_Factor* `[FP]` {*Common*, OMATS1}

  The weight of the specular color is used to control the intensity of the specular reflection of the object's surface. Highlights (Phong model) are not affected by this parameter.

  The values usually are in the range of `0.0` to `1.0`.

  The predefined value is `1.0`.

- *Transparency* `[FP]` {*Common*, *Glass*, *Illuminant*, OMATS1, OMATS2}

  The transparency is used to simulate transparent characteristics of the material. It is a frequency-independent scalar value.

  The values are in the range of `0.0` to `1.0`.

  The predefined value is `0.0`, which means that there is no transparency.

3 Texture mapping methods

OMATS supports the texture mapping methods described in this section. These relate to the material parameters *Base_Color_Map*, *Metallness_Map*, *Normal_Map* and *Roughness_Map*.

All methods start with the data types `RGB-IMAGE`, `RGBA-IMAGE` or `GRAYSCALE-IMAGE` as defined in 2.2. As shown in figure 1, these images are projected onto the normalized UV coordinate space which is the basis for all other statements in this section.

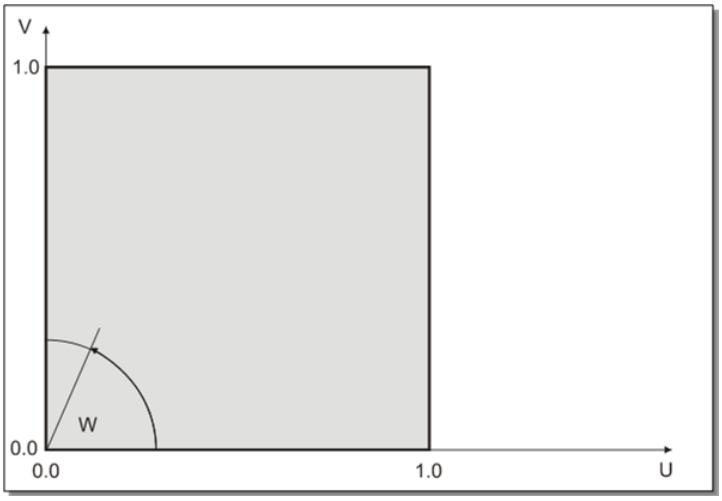


Figure 1: UV coordinate space

The following texture transformations are supported (in the specified order):

1. Rotation by the angle W
2. Translation by an U-V offset
3. Scaling in the UV space

For normal maps separate transformation parameters can be specified (related to the other maps). If no specific transformation parameter is specified for the normal map, the corresponding parameter for the other maps is used (if available).

3.1 Plane mapping

This is a planar mapping onto a given projection plane. It defines the location of the UV space and can be selected as follows:

- YZ plane
- XZ plane
- XY plane
- Definition by a normalized normal vector

In addition, the projection plane freely can be rotated around all coordinate axes.

Translation and scaling in the UV space is supported, too.

3.2 Block mapping

This is a automatic mapping of the model coordinates onto the boundary surfaces of a paraxial oriented block. The block defines its own UV space along each coordinate axis. If basic vectors U and V each correspond to a canonical basic vector, the 8 base variants can be specified, as shown in figure 2.

Inversions of the block at the coordinate planes cause corresponding inversions in the UV space, and thus lead to further variants. Translation, scaling and rotation in UV space are supported, too. The assignment of a vertex to a side surface of the block is based on the coordinate of the normal vector with the greatest amount.

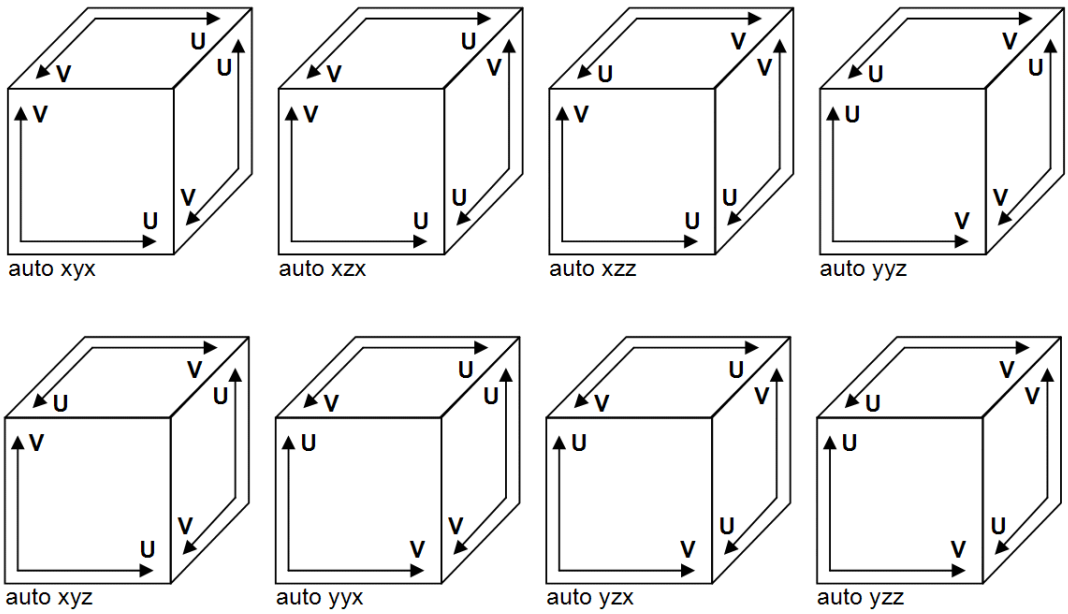


Figure 2: Block mapping

In the triple after the keyword `auto`, the direction of the U vector is encoded for each side of the block in the order: front, right, top.
(Analog the directions for back, left and bottom sides.)

3.3 Texture coordinates

Not always the desired result can be described by means of general mapping methods. Sometimes an explicit specification of the UV coordinates is required. Then, these coordinates are not stored with the material, but with the geometry itself. This eliminates the need to calculate the projection from the model space into the UV space. Scaling, offset and rotation still are applied to the UV coordinates.

This method applies to all types of maps.

How the texture coordinates are stored depends on the geometry format, i.e. the specification of texture coordinates must be provided there. For this purpose formats 3DS and OBJ can be used.

4 OFML data format for materials

Preliminary note:
The statements in this section replace and update the statements in appendix D.2 "Materials" from [ofml]!

The definition of an OFML material consists of a set of *parameters*. A parameter is comprised of a key that defines the meaning of the parameter, followed by space-separated arguments[8]. The tables below define the currently supported keys and corresponding arguments.

A material definition can be represented in two formats, which differ essentially in the form of separation of the parameters:

- **material definition file**

  The parameters are separated by the end of the line.

  The name of a material definition file (extension `.mat`) results from the last component of the fully qualified name of the material which is used to reference it in the OFML data (e.g. ODB, OFML part I [odb]), where the file name is spelled with lower case[9].

- **inline declaration**

  The parameters are separated by a semicolon (';').

  Inline declarations can be used in OFML programming according to part III of the OFML specification [ofml] or in ODB data [odb].

  Inline declarations can be specified in two forms:

  - *Pure inline declarations* start with the dollar character ('$') and contain a complete material definition.
  - *Material modifiers* start with a fully qualified material name that refers to a material in the OFML database (basic material). This is followed by individual parameters, separated by a semicolon, which overwrite the corresponding parameter of the basic material.

The syntactic and lexical elements used in the description of the arguments in the following tables are described in the legend at the end of this section.

[8] There are also parameters without arguments.

[9] The name of a material – without the prepended package name space – should follow the rules for OFML identifiers, i.e. should contain only alphanumeric characters (including the underscore) and not begin with a digit.

The following table defines the corresponding keys and arguments for all currently supported model parameters (see section 2):

Parameter	Model	Key	Argument(s)
Material_Type	1, 2	`type`	`(common|glass|illuminant)`
Base_Color	1, 2	`dif`	`R[F1] G[F1] B[F1]`
Base_Color_Map	1, 2	`tex image`	`FT[FT] FN[FN]`
Emissive_Color	1, 2	`emission`	`R[F1] G[F1] B[F1]`
Luminance	1, 2	`luminance`	`S[F]`
Metallness	2	`metallic`	`S[F1]`
Metallness_Map	2	`metallic image`	`FT[FT] FN[FN]`
Normal_Map	1, 2	`bumps`	`FT[FT] FN[FN]`
Refractive_Index	1, 2	`ref`	`S[F]`
Roughness	2	`roughness`	`S[F1]`
Roughness_Map	2	`roughness image`	`FT[FT] FN[FN]`
Shininess	1	`shi`	`S[F]`
Sound_Absorption	1, 2	`sndabsorb`	`N[I] {F[I] C[F]}*`
Specular_Color	1	`spe`	`R[F1] G[F1] B[F1]`
Specular_Factor	1	`reflection`	`S[F]`
Transparency	1, 2	`tra`	`S[F1]`

The following table defines the keys and corresponding arguments required for the texture mapping methods (see section 3).

Parameter	Key	Argument(s)
Transformations[a]		
Rotation	`rotate`, `nrotate`	`0 0 A[F]`
Translation	`offset`, `noffset`	`U[F] V[F] 0`
Scaling	`scale`, `nscale`	`U[F] V[F] 0`
Plane mapping		
YZ plane	`prjx`	
XZ plane	`prjy`	
XY plane	`prjz`	
normal vector	`prj`	`X[F1] Y[F1] Z[F1]`
Block mapping	`auto`	`(xyx|xzx|xzz|yyz|xyz|yyx|yzx|yzz)`
Texture coordinates[b]	`import`	

[a] The parameters that begin with the letter 'n', affect only the normal map. If one of them is not specified, the corresponding parameter without letter 'n' at the beginning is used.

[b] Texture coordinates stored in the object geometry are used only if the `import` parameter is present. If the key is specified but there are no texture coordinates in the geometry, the behavior is undefined.

Legend:

- An argument is described either by an explicit list of the possible (alternative) values in the form `(value1|value2|...)`[10] or in the form `name[type]`, where the name denotes the semantics of the argument.
- A repeating set of arguments is represented in the form `{arg1 ...}*`.
- The following identifiers (abbreviations) are used for named arguments:
  - `S` – scalar value
  - `FT`, `FN` – file type, file name
  - `R`, `G`, `B` – red, green, blue
  - `U`, `V`, `A` – UV coordinates or scaling, angle
  - `X`, `Y`, `Z` – XYZ coordinates
  - `N`, `F`, `C` – number, frequency, absorption coefficient
- The following identifiers are used for the types of arguments:
  - `F` – floating-point number
  - `F1` – floating-point number in the range of `0.0` to `1.0`
  - `I` – integer
  - `FT` – file type: `(png|jpg|any)`[11]
  - `FN` – file name: (possibly fully qualified) OFML name which refers to an image file in the OFML database[12]

[10] If there is only a single possible value the enclosing parenthesis is omitted.

[11] In the case of `any` the type is determined from the file extension.

[12] Qualification is necessary if the image file is not in the data directory of the OFML series, where the material definition file is stored, or if it is not in the data directory of the series of the OFML instance to which an inline declaration is applied.

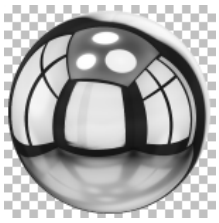
A Introduction into Physically Based Rendering (PBR)

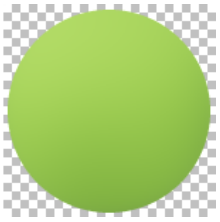
PBR simulates what happens when light hits the surface of an object. A physically correct described material interacts with light in different ways: Light is *reflected*, *refracted* or *absorbed*. In natural conditions, light is not completely absorbed, reflected or refracted – all materials are in the spectrum between these extremes.

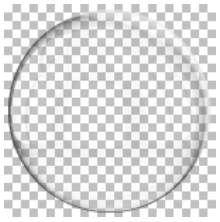
A material becomes visible to us because it reflects incident light. Furthermore we are able to see materials that emit light by themselves.

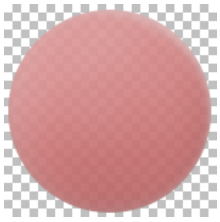
Amongst others, the mentioned three basic possibilities of interaction between light and material are affected by the **characteristics of the material**:

- The characteristics of the material determine the type of reflection:
  In *specular reflection*, the light is reflected directly at the surface.
  Diffuse reflection occurs due to scattering within the material (light rays penetrate into the material a little and are deflected in different directions.)
  Metal materials only have specular reflection, non-metals predominantly reflect diffuse.
- Depending on the characteristics of the material, the light rays enter deeper into the material. They are either passed through the material (*transparency*), or thrown back within the material (*translucency*) or swallowed by the material (*absorption*).


specular reflection


diffuse reflection


transparency


translucency


absorption

In addition to the characteristics of the material (see above) Physically Based Rendering also considers the physical **characteristics of the light**:

- According to the principle of conservation of energy, no more light is reflected than irradiated.
  A – non-luminous – material is displayed according to the lighting of the environment.
- The amount of reflected light rays depends on the angle of view. This so-called *Fresnel effect* causes surfaces to reflect more intensely at a flat viewing angle[13] than when looking perpendicular to the surface.

The characteristics of the light are simulated via the shader of the application and can not be manipulated directly by the user (material data creator).

The characteristics of materials, however, are the instruments for creating physically coherent materials. The PBR-oriented material model OMATS2 defines corresponding parameters, where the amount of the relevant material parameters depends on the specified *material type* (see section 2).

[13] the so-called grazing angle

B Conversion of older materials to the new model

Materials that were created based on the old model OMATS1 automatically are converted by the OFML application to the new model.

Normally, this conversion provides a satisfying representation. In rare cases, however, adjustments may be necessary:

- In some cases, materials shine more intensely.
  In this case, the roughness has to be adjusted by specifying the (new) parameter *Roughness*.
- Metals may not be recognized as such (this may be true, e.g., for chrome surfaces).
  In this case, new parameter *Metallness* has to be specified explicitly (with value `1.0`).

C History

The first versions of this specification were prepared by Ekkehard Beier (EasternGraphics GmbH) on behalf of the Working Group *Industrielle Aspekte der OFML-Normung*[14] (IAON) in cooperation with wegscheider office solution gmbh (Germany) and weber office solution gmbh (Schwitzerland). Starting with version 1.4, the specification is subject to standardization by the OFML standardization board of the IBA.

Version 2.1 (2023-06-27)

- For image files (data types **-IMAGE*) the maximum allowed size was set to 4.096 x 4.096.

Version 2.0 (2019-06-19)

- New material model OMATS2
- New data type *GRAYSCALE-IMAGE*
- Removed data type *FP3-IMAGE*, instead enhanced description of parameter *Normal_Map*
- Renamed parameter *Diffuse_Color* in *Base_Color* and *Diffuse_Map* in *Base_Color_Map* as well as *Refraction* in *Refractive_Index*
- Removed material types *Glass_Translucent* and *Metal_Polished* as well as parameter *Ambient_Color* due to low practical relevance
- Image file format TGA now is obsolet
- Removed references to AutoCAD
- Restructuring of the document

Version 1.5 (2015-02-27)

- New, explicit transformation parameters *nrotate*, *noffset* and *nscale* for normal maps.

Version 1.4, 1st revised version (2014-01-08)

- First english issue of this specification.

[14] Industrial aspects of OFML standardization