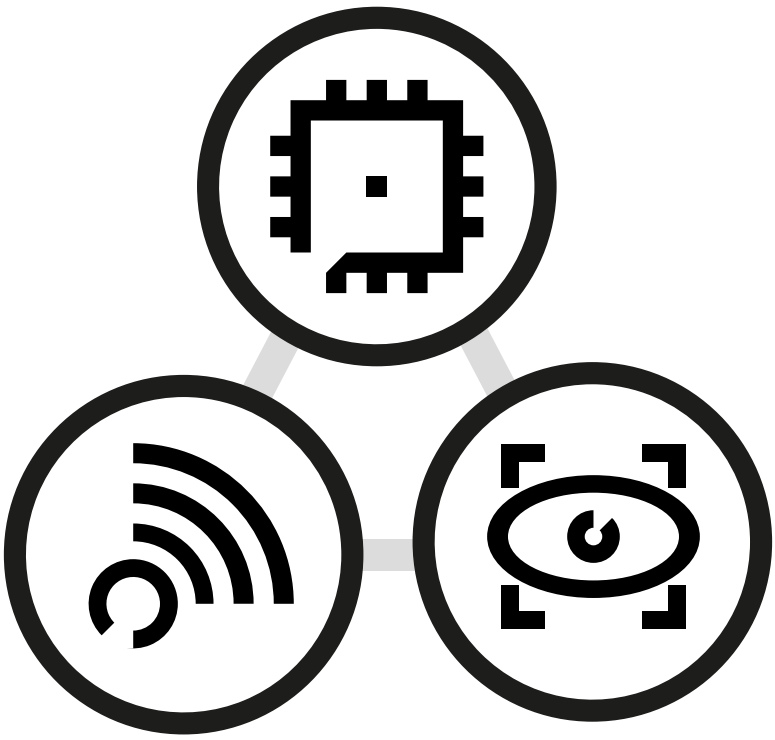
Imagination

POD File Format Specification

Revision: 1.0
11/02/2020
Public



Published: 11/02/2020-10:08

Contents

1. Introduction

An overview of this document and the structure of the POD file format

Document Overview

The purpose of this document is to act as a reference and specification for the POD file format (POD specification version 2.1), specifically all of the various types of data blocks which can make up a POD file.

What is a POD file?

The POD file format is deployment format which is used to contain scene and object data, including meshes, materials, lights, cameras, and so on.

Each POD file is made up of many different kinds of blocks which hold information about the scene. The overall block structure of a POD file is shown below.

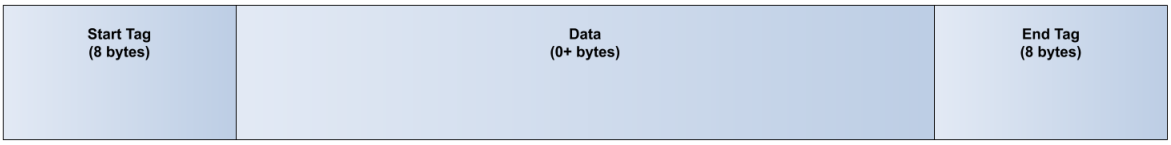
Each POD file contains a set of blocks containing header information including a POD version block, an export options block, and a history block, as well as a number of blocks which contain the actual scene data. It is important to note that *Data* may contain blocks, which may, in turn, contain further blocks and so on.

This document will cover the structure of these data blocks, as well as the various types of data blocks which can make up a POD file.

2. POD Block Structure

A basic block in a POD file is composed of a data sandwiched between a start and an end tag

Every block in the file is made up of data sandwiched between a start tag and an end tag, both of which are 8 bytes long. These tags are used to specify the type of data between them and its length.



Start and end tags

The start and end tags define the size of the data in a block, as well as what type of block it is

The *Start Tag* and *End Tag* share the same structure. They are split into two DWORDs.

The first DWORD specifies whether the tag is a *Start Tag* or an *End Tag* and also specifies the identifier of the block. Each type of block has a unique identifier number which can be used to specify the block type. For example, a block which contains data on the number of cameras in a scene has the unique identifier 2002, while a block which specifies the number of faces in a mesh has an identifier 6001.

The second DWORD specifies the length in bytes of the data block between the two tags.

DWORD	Bit	Symbol	Description
0	31	Start/End	Bit Value = 0 – This tag is the beginning of a block Bit Value = 1 – This tag is the end of a block
	0 – 30	Identifier	Block Type Identifier
1	0 – 31	Length	The length of ‘Data’ in bytes.

Data may contain blocks, which may, in turn, contain further blocks and so on. It should also be noted that a block which contains only further nested blocks between its *Start Tag* and *End Tag* will have a *Length* of zero.

Reading POD Files

The simple algorithm for reading a POD file is shown in pseudocode below.

```
While not at end-of-file
 Read 8 bytes from file
  If 'Identifier' is a valid 'Start Tag'
   Read 'Length' number of bytes of 'Data'.
    Handle ‘Data’
   Go down a level in nested structure
  Else if 'Identifier' is valid 'End Tag':
```

```
    Read 'Length' number of bytes of 'Data'.
     Handle 'Data'
    Go up a level in nested structure
```

3. Important Notes

This section contains a few brief notes about different aspects of the POD specification

Before looking at all of the different data block types which make up the POD file format, there are a few brief notes about different aspects of the POD specification.

Block Type Identifiers

Each block type identifier is an unsigned 32-bit integer. However, as the most significant bit of the integer is reserved for determining if a tag is a *Start Tag* or an *End Tag* it must be masked.

The *Start Tag* and *End Tag* masks are as follows:

- 0x00000000 – *Start Tag* mask.
- 0x80000000 – *End Tag* mask.

The *Identifier* section within each entry of the block list gives the value prior to masking.

Indicies

Several blocks within the POD format reference an index. This index refers to the position (counting from zero) of an element within a list or similar data structure. This indexing means that the ordering of objects within the file must be maintained, or translated, post-loading for these indices to have any meaning.

It should also be noted that indices can be set to '-1', in this instance the index does not refer to any element. For example, a camera that is not following an object may have its *Target Object Index* set to '-1'.

Float/Fixed

Some elements of a POD file may use either floating point or fixed point data types. These are referenced in the *POD Block Structure* as *Float/Fixed*. *Float* should be used by default, unless overwritten by the *Scene Flags* block.

Existence of Blocks

Only the existence of the *Version* block is guaranteed.

Nesting of blocks must be maintained as described in the POD File Structure. The existence of child blocks within a parent block is guaranteed only if the child block is required for the parent block to function.

Node Ordering

Nodes will appear in the following order:

1. Meshes.
2. Lights.

3. Cameras.
4. Everything else.

This is important to remember as the only way to be certain that the index of a node references, for example, a camera, is to know that all meshes and lights have already passed.

4. POD File Structure

This section outlines the various block which make up the structure of a POD file

As previously mentioned a POD file is divided into a set of header blocks and a set of blocks containing the scene data.

This section will go into more detail to describe the purpose of each of these blocks, as well as any sub-blocks located inside them.

Header Blocks

The header blocks give general information about the POD file.

Name	Identifier	Description
Version	1000	A null terminated character string containing the following: "AB.POD.2.0"
Export Options	1002	A null terminated character array containing the options used to export the POD file. The contents of this string are implementation specific from exporter to exporter and are primarily used to allow an exporter to re-read the options used in an existing POD file.
History	1003	A null terminated character array containing the history of the POD file. The exact contents of this string are implementation-specific from exporter to exporter. Its use is informational only.

Scene Data Blocks

The scene data block contains all of the data which describes a scene. This block is made up of various other blocks which describe the number of cameras, lights, meshes, materials, and so on.

A full list of the various different kinds of blocks within the scene data block is shown below.

Name	Description
Clear Colour	Clear colour of the scene (see *Clear Colour*).
Ambient Colour	Ambient colour of the scene (see *Ambient Colour*).
Num. Cameras	Number of cameras in the scene (see *Num. Cameras*)
Num. Lights	Number of lights in the scene (see *Num. Lights*).
Num. Meshes	Number of meshes in the mesh array (see *Num. Meshes*).
Num. Nodes	Number of nodes in the scene (see *Num. Nodes*).
Num. Mesh Nodes	The total number of meshes in the scene (this may be larger than *Num. Meshes* as this number will include instanced meshes) (see *Num. Mesh Nodes*).

Name	Description
Num. Textures	Number of textures in the scene (see *Num. Textures*).
Num. Frames	Number of frames of animation in the scene (see *Num. Frames*).
Num. Materials	Number of materials in the scene (see *Num. Materials*).
Camera	Specifies all the information relating to a single camera within the scene. This block may appear multiple times (see *Camera*).
Light	Specifies all the information relating to a single light within the scene. This block may appear multiple times (see *Light*).
Mesh	Specifies all the information relating to a single mesh within the scene. This block may appear multiple times (see *Mesh*).
Node	Specifies all the information relating to a single node within the scene. This block may appear multiple times (see *Node*).
Texture	Specifies all the information relating to a single texture within the scene. This block may appear multiple times (see *Texture*).
Material	Specifies all the information relating to a single material within the scene. This block may appear multiple times (see *Material*).
Scene Flags	Specifies whether a number of flags are set within the POD file (see *Scene Flags*).
FPS	Specifies the animation speed of the scene, in frames per second (see *FPS*).
Scene User Data	Custom data added by the exporter (see *Scene User Data*).

The rest of this section will provide a reference for each of these blocks including their unique identifier numbers and a description of the types of data they contain.

Clear Colour

The *Clear Colour* block contains the channel values of the clear colour of the scene

The *Clear Colour* block contains the channel values of the clear colour of the scene. These channel values are stored in the order RGB.

This block has the *Identifier*: **2000**

Name	Data Type	Description
Red Channel	Float/Fixed	A 4 byte float/fixed describing the value of the red channel in the range of [0 - 1].
Green Channel	Float/Fixed	A 4 byte float/fixed describing the value of the green channel in the range of [0 - 1].
Blue Channel	Float/Fixed	A 4 byte float/fixed describing the value of the blue channel in the range of [0 - 1].

Ambient Colour

The *Ambient Colour* block contains the channel values of the ambient colour of the scene

The *Ambient Colour* block contains the channel values of the ambient colour of the scene in the order RGB.

This block has the *Identifier*: **2001**

Table 1: Ambient Colour Data

Name	Data Type	Description
Red Channel	Float/Fixed	A 4 byte float/fixed describing the value of the red channel in the range of [0 - 1].
Green Channel	Float/Fixed	A 4 byte float/fixed describing the value of the green channel in the range of [0 - 1].
Blue Channel	Float/Fixed	A 4 byte float/fixed describing the value of the blue channel in the range of [0 - 1].

Num. Cameras

The *Num. Cameras* block specifies the number of cameras in the scene

The *Num. Cameras* block specifies the number of cameras in the scene, stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Cameras	2002	Unsigned 32-bit integer

Num. Lights

The *Num. Lights* block specifies the number of lights in the scene

The *Num. Lights* block specifies the number of lights in the scene. The data in this block is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Lights	2003	Unsigned 32-bit integer

Num. Meshes

The *Num. Meshes* block specifies the number of meshes in the scene

The *Num. Meshes* block specifies the number of meshes in the scene. This data is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Meshes	2004	Unsigned 32-bit integer

Num. Nodes

The *Num. Nodes* block specifies the number of nodes in the scene

The *Num. Nodes* block specifies the number of nodes in the scene. This data is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Nodes	2005	Unsigned 32-bit integer

Num. Mesh Nodes

The *Num. Mesh Nodes* block specifies the total number of meshes in the scene

The *Num. Mesh Nodes* block specifies the total number of meshes in the scene. This may be larger than *Num. Meshes* as this number will include instanced meshes. This data is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Mesh Nodes	2006	Unsigned 32-bit integer

Num. Textures

The *Num. Textures* block specifies the number of texture in the scene

The *Num. Textures* block specifies the number of texture in the scene. This data is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Textures	2007	Unsigned 32-bit integer

Num. Materials

The *Num. Materials* block specifies the number of materials used in the scene

The *Num. Materials* block specifies the number of materials used in the scene. This data is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Materials	2008	Unsigned 32-bit integer

Num. Frames

The *Num. Frames* block specifies the number of frames of animation for the scene

The *Num. Frames* block specifies the number of frames of animation for the scene. This data is stored as a 32-bit unsigned integer.

Name	Identifier	Data Type
Num. Frames	2009	Unsigned 32-bit integer

Camera

The *Camera* block contain all of the information relating to a single camera within the scene

The *Camera* block contain all of the information relating to a single camera within the scene.

The *Identifier* for this block is: **2010**

This block contains a number of sub-blocks which hold various properties of the camera. These blocks are listed below.

Name	Description
Field of View	The FOV of the camera (see *Field of View*).
Far Plane	The location of the far plane for the camera (see *Far Plane*).
Near Plane	The location of the near plane for the camera (see *Near Plane*).
FOV Animation	The FOV for each frame of animation, for use with FOV animation (see *FOV Animation*).
Target Object Index	The index into the node array of the object the camera should target (see *Target Object Index*).

Field of View

The *Field of View* block simply contains the field of view value of the camera

The *Field of View* block simply contains the field of view value of the camera.

Name	Identifier	Data Type
Field of View	8001	Float/Fixed

Far Plane

The *Far Plane* block contains the position of the far plane in relation to the camera.

The *Far Plane* block contains the position of the far plane in relation to the camera.

Name	Identifier	Data Type
Far Plane	8002	Float/Fixed

Near Plane

The *Near Plane* block contains the position of the near plane in relation to the camera

The *Near Plane* block contains the position of the near plane in relation to the camera.

Name	Identifer	Data Type
Near Plane	8003	Float/Fixed

FOV Animation

The *FOV Animation* block contains an array of Float/Fixed values, each of which represent the FOV of the camera during each frame of animation

The *FOV Animation* block contains an array of Float/Fixed values, each of which represent the FOV of the camera during each frame of animation.

Name	Identifier	Data Type
FOV Animation	8004	Float/Fixed array

Target Object Index

The *Target Object Index* block contains an index into the node list of the object whose position the camera should use as its target

The *Target Object Index* block contains an index into the node list of the object whose position the camera should use as its target.

Name	Identifier	Data Type
Target Object Index	8000	Unsigned 32-bit integer

Light

The *Light* block contains all of the information relating to a single light within the scene

The *Light* block contains all of the information relating to a single light within the scene.

The *Identifier* for this block is: **2011**

This block contains a number of sub-blocks which store various properties of the light object. This blocks are listed below.

Name	Description
Light Colour	The colour of the light (see *Light Colour*).
Light Type	The type of the light, such as a point, directional, spot, and so on (see *Light Type*).
Constant Attenuation	The constant attenuation of the light (see *Constant Attenuation*).
Linear Attenuation	The linear attenuation of the light (see *Linear Attenuation*).
Quadratic Attenuation	The quadratic attenuation of the light (see *Quadratic Attenuation*).
Falloff Angle	The falloff angle of the light (in radians) (see *Falloff Angle*).
Falloff Exponent	The falloff exponent of the light (see *Falloff Exponent*).
Target Object Index	The index into the node array of the object the light should target (see *Target Object Index*).

Light Colour

The *Light Colour* block contains a list of the values of the colour channels of the associated light

The *Light Colour* block contains a list of the values of the colour channels of the associated light. This list is in the order RGB.

Name	Identifier	Data Type
Light Colour	7001	Float/Fixed array

Light Type

The *Light Type* block specifies which variety of light object the associated light is

The *Light Type* block specifies which variety of light object the associated light is.

The following values correspond to different types of light:

- 0 – Point Light
- 1 – Directional Light
- 2 – Spot Light

Name	Identifier	Data Type
Light Type	7002	Unsigned 32-bit integer

Constant Attenuation

The *Constant Attenuation* block specifies the constant attenuation of the associated light object

The *Constant Attenuation* block specifies the constant attenuation factor of the associated light object. This factor in combination with *Linear Attenuation* and *Quadratic Attenuation* determine how the intensity of the light falls off with distance.

Note:

This is only valid if used for a spot light.

Name	Identifier	Data Type
Constant Attenuation	7003	Signed 32-bit float

Linear Attenuation

The *Linear Attenuation* block specifies the linear attenuation factor of the associated light object

The *Linear Attenuation* block specifies the linear attenuation factor of the associated light object. This factor in combination with *Constant Attenuation* and *Quadratic Attenuation* determine how the intensity of the light falls off with distance.

Note: This is only valid if used for a spot light.

Name	Identifier	Data Type
Linear Attenuation	7004	Signed 32-bit float

Quadratic Attenuation

The *Quadratic Attenuation* block specifies the constant attenuation of the associated light object

The *Quadratic Attenuation* block specifies the quadratic attenuation factor of the associated light object. This factor in combination with *Constant Attenuation* and *Linear Attenuation* determine how the intensity of the light falls off with distance.

Note:

This is only valid if used for a spot light.

Name	Identifier	Data Type
Quadratic Attenuation	7005	Signed 32-bit float

Falloff Angle

The *Falloff Angle* block specifies the falloff angle of the associated light. The falloff angle determines how the intensity of a spot light fades as you move out radially from the centre line of the light. A larger falloff angle leads to a more spread out spot light which fades more gently as you move outwards.

Note: This block is only valid if the light is a spot light.

Name	Identifier	Data Type
Falloff Angle	7006	Signed 32-bit float

Falloff Exponent

The *Falloff Exponent* block specifies the falloff exponent of the associated light

The *Falloff Exponent* block specifies the falloff exponent of the associated light. The falloff exponent determines the rate of radial falloff of the intensity of a spot light. The higher the falloff, the faster intensity decreases as you move outwards from the centre of a spot light.

Note: This block is only valid if the light is a spot light.

Name	Identifier	Data Type
Falloff Exponent	7007	Signed 32-bit float

Target Object Index

The *Target Object Index* block contains the index in the node list of the object whose position the light should use as its target

The *Target Object Index* block contains the index in the node list of the object whose position the light should use as its target.

Name	Identifier	Data Type
Target Object Index	7000	Unsigned 32-bit integer

Mesh

The *Mesh* block contains all of the information relating to a single with the scene

The *Mesh* block contains all of the information relating to a single with the scene.

The identifier for this block is: **2012**

This block contains various sub-blocks which define the properties of the mesh object. These blocks are listed below.

Table 2: Sub-blocks

Name	Description
Num. Faces	The number of triangles in the mesh (see *Num. Faces*).
Num. UVW Channels	The number of texture coordinate channels in the mesh (see *Num. UVW Channels*).

Name	Description
Vertex Index List	The list of vertex indices for the faces in an indexed mesh (see *Vertex Index List*).
Strip Length	A list, one entry per strip, of the number of triangles within each strip (see *Strip Length*).
Num. Strips	The total number of strips (see *Num. Strips*).
Vertex List	The list of vertices, when data is interleaved this will contain and offset into the *Interleaved Data List* and a stride for moving from element to element (see *Vertex List*).
Normal List	The list of normals, when data is interleaved this will contain and offset into the *Interleaved Data List* and a stride for moving from element to element (see *Normal List*).
Tangent List	The list of tangents, when data is interleaved this will contain and offset into the *Interleaved Data List* and a stride for moving from element to element (see *Tangent List*).
Binormal List	The list of binormals, when data is interleaved this will contain and offset into the *Interleaved Data List* and a stride for moving from element to element (see *Binormal List*).
UVW List	The list of texture coordinates, when data is interleaved this will contain and offset into the *Interleaved Data List* and a stride for moving from element to element. This will appear a number of times equal to 'Num. UVW Channels' (see *UVW List*).
Vertex Colour List	A list of colours per vertex (see *Vertex Colour List*).
Bone Index List	A list of indices into the Bone Batch Index List detailing which matrices should affect which vertex, with a number of indices per vertex equal to the number of bones (see *Bone Index List*).
Bone Weights	The weight for each bone reference in the *Bone Index List* (see *Bone Weights*).
Bone Batch Index List	A list of indices into the *Node* list, each *Node* representing the transformations associated with a single bone. (Read via '*Bone Index List*) (see *Bone Batch Index List*).
Num. Bone Indices per Batch	A number of integers equal to *Num. Bone Batches* that state how many bones exist within each batch (see *Num. Bone Indices per Batch*).
Bone Offset per Batch	A number of integers equal to *Num. Bone Batches*' that state the offset into the *Vertex Index List*' for each sub-mesh that uses the given bone batch (see *Bone Offset per Batch*).
Max. Num. Bones per Batch	The maximum number bones any given bone batch can contain (see *Max. Num. Bones per Batch*).
Num. Bone Batches	The total number of bone batches used in the mesh (see *Num. Bone Batches*).
Unpack Matrix	A matrix used for unpacking scaled vertex data (see *Unpack Matrix*).
Interleaved Data List	The list of all vertex data interleaved, read using the offsets and strides mentioned above (see *Interleaved Data List*).

Name	Description
Num. Vertices	The number of vertices in the mesh (see *Num. Vertices*).

Mesh Type

The *Mesh Type* block specifies which type of geometric primitives make up the mesh

The *Mesh Type* block specifies which type of geometric primitives make up the mesh.

The mesh types supported by the POD file format are:

Primitive	Corresponding Block Value
Triangles	0
Points	1
Lines	2
Patches	3
Nurbs	4

Name	Identifier	Data Type
Mesh Type	6021	Unsigned 32-bit integer

Num. Faces

The *Num. Faces* block specifies the number of faces in the mesh, more specifically, the number of triangles in the mesh

The *Num. Faces* block specifies the number of faces in the mesh, more specifically, the number of triangles in the mesh.

Name	Identifier	Data Type
Num. Faces	6001	Unsigned 32-bit integer

Num. Vertices

The *Num. Vertices* block simply specifies the number of vertices in the mesh

The *Num. Vertices* block simply specifies the number of vertices in the mesh.

Name	Identifier	Data Type
Num. Vertices	6000	Unsigned 32-bit integer

Num. UVW Channels

The *Num. UVW Channels* block specifies the number of texture coordinate channels in the mesh

The *Num. UVW Channels* block specifies the number of texture coordinate channels in the mesh.

Name	Identifier	Data Type
Num. UVW Channels	6002	Unsigned 32-bit integer

Num. Strips

The *Num. Strips* block simply specifies the total number of strips in the mesh

The *Num. Strips* block simply specifies the total number of strips in the mesh.

Name	Identifier	Data Type
Num. Strip	6005	Unsigned 32-bit integer

Strip Length

The *Strip Length* block contains a list of the number of triangles within each strip

The *Strip Length* block contains a list of the number of triangles within each strip. Each entry of this list specifies the number of triangles in an individual strip.

Name	Identifier	Data Type
Strip Length	6004	Unsigned 32-bit integer array

Blocks using the POD Data block

This section contains descriptions of all the block which use the POD Data block

This section contains descriptions of all the blocks which use the POD Data block.

They contain the bulk of the actual data which describes a mesh, including a list of vertices, normals, tangents, vertex colours, and so on.

Note: Within the *Mesh* block the structure is flat. Some of the data list blocks use the values of other blocks, but these blocks do not need to be located inside the data list blocks.

POD Data Block Structure

A POD data bock is a special type of block with a structure that is designed to store large amounts of data, such as a vertex list.

The data block is divided into four main blocks. The first three help to describe the type and format of the data while the final block contains the data itself.

Data Type

The *Data Type* block holds an unsigned 32-bit integer which represents the data type of the elements in the *Data* block.

Name	Identifier	Data Type	Valid Values
Data Type	9000	Unsigned 32-bit integer	• 0 – none • 1 – signed 32-bit float • 2 – unsigned 32-bit integer • 3 – unsigned short • 4 – four, single byte integer values representing colour channels in the order RGBA • 5 – four, single byte integer values representing colour channels in the order ARGB • 6 – a 4-byte value representing a D3DCOLOR (see msdn.microsoft.com) • 7 – a 4-byte value representing UBYTE4 • 8 – a 4-byte value representing a DEC3N • 9 – a 4-byte value representing a fixed point value in the format 16.16 • 10 – unsigned byte • 11 – short • 12 – normalised short • 13 – byte • 14 – normalised byte • 15 – unsigned normalised byte • 16 – unsigned normalised short • 17 – unsigned integer

Num. Components

The *Num. Components* block holds a 32-bit integer representing the number of components per item held in the *Data* block.

For example, if *Data* contained a list of vertex positions consisting of three floats, the value held in *Num. Components* would be '3', four floats would give a value of '4', and so on.

Name	Identifier	Data Type	Valid Values
Num. Components	9001	Unsigned 32-bit integer	Any valid unsigned 32-bit integer

Stride

The distance, in bytes, from one array member within *Data* to the next.

Name	Identifier	Data Type	Valid Values
Stride	9002	Unsigned 32-bit integer	Any valid unsigned 32-bit integer

Data

Data contains a list of elements. The type of these elements is determined by the value set in *Data Type*

If the data for a given block is interleaved, *Data* will instead contain a byte representing the offset into the *Interleaved Data List*' of the first element of the block in question, as an unsigned 8-bit integer value.

For example, if *Data* would represent normal data for a vertex, but that normal data is interleaved, *Data* will contain the offset into the *Interleaved Data List* of the first vertex's normal data, from that point onwards, the normal data for each vertex can be read by moving forward by the value of *Stride*.

Name	Identifier	Data Type	Valid Values
Data	9003	Variable array/byte	Any valid unsigned 32-bit integer

Vertex List

The *Vertex List* block contains a list of all of the individual vertices which make up an object

The *Vertex List* block contains a list of all of the individual vertices which make up an object. This is one of the most important blocks in a *Mesh* because it describes the actual shape of the mesh.

Name	Identifier	Data Type	Related Blocks
Vertex List	6006	*POD Data block*	*Unpack Matrix*

Unpack Matrix

The *Unpack Matrix* is used for unpacking the data found in the *Vertex List*. If this matrix is not the identity matrix and the *Vertex List* contains data in a non-float data type, then that data has been scaled to make better use of the precision of the given data type. In cases where this is true, vertices must be 'unpacked' using the *Unpack Matrix* before any other transformations are applied.

Note: Using *Unpack Matrix* with the *Fixed Point* data type will not function correctly.

Table 3: Unpack Matrix data

Name	Identifier	Data Type
Unpack Matrix	6020	Signed 32-bit float

Vertex Index List

The *Vertex Index List* contains a list of vertex indices for the faces in an indexed mesh. These indices are used to order the vertices in the *Vertex List* into the faces which form the mesh. Vertices are grouped into groups of three for triangles, groups of two for lines, and so on, depending on the mesh type.

Name	Identifier	Data Type
Vertex Index List	6003	*POD Data block*

Normal List

The *Normal List* block contains a list of surface normals for each vertex within the mesh

The *Normal List* block contains a list of all of the normals within the mesh. Normals are oriented outwards, perpendicularly away from the surface of the mesh. Normals are often used in lighting calculations.

Name	Identifier	Data Type
Normal List	6007	*POD Data block*

Tangent List

The *Tangent List* block contains a list of tangents for each vertex within the mesh

The *Tangent List* block contains a list of tangents for each vertex within the mesh. Tangent vectors are parallel to the surface of the mesh.

Name	Identifier	Data Type
Tangent List	6008	*POD Data block*

Binormal List

The *Binormal List* block contains a list of binormals, with one binormal vector per vertex

The *Binormal List* block contains a list of binormals, with one binormal vector per vertex. Binormals are perpendicular to both the normal and tangent vectors.

Name	Identifier	Data Type
Binormal List	6009	*POD Data block*

UVW List

The UVW List block contains a list of UVWs within the mesh, with one value for each vertex in the vertex list

The UVW List block contains a list of UVWs within the mesh, with one value for each vertex in the vertex list. The UVW values in this list are texture coordinates used during texture mapping. This block may appear multiple times, once per set of UVW mappings.

Name	Identifier	Data Type
UVW List	6010	*POD Data block*

Vertex Colour List

The *Vertex Colour List* block contains a list of vertex colour, with one colour value for each vertex in the *Vertex List*

The *Vertex Colour List* block contains a list of vertex colour, with one colour value for each vertex in the *Vertex List*. Vertex colours are colour information that is associated them each vertex. Shaders sometimes use this data to determine the polygon colours by interpolating between the colours of each vertex.

Name	Identifier	Data Type
Vertex Colour List	6011	*POD Data block*

Bone Index List

The *Bone Index List* block contains a list of indices of items in the *Bone Batch Index List* detailing which bones should affect which vertex.

The *Bone Index List* block contains a list of indices of items in the *Bone Batch Index List* detailing which bones should affect which vertex.

The total number of indices is equal to the highest number of bones affecting any vertex within the mesh (*Max. Num. Bones per Batch*), multiplied by the number of vertices:

$$num.indices = num.bones_{max} * num.vertices$$

Each vertex has an equal number of indices. The indices that are not relevant to a given vertex have the weight that matches the index in question set to zero.

Name	Identifier	Data Type	Associated Blocks
Bone Index List	6012	*POD Data block*	*Bone Weights* *Max Num. Bones per Batch* *Bone Batch Index List*

Bone Weights

The *Bone Weights* block contains the weight for each bone reference in the *Bone Index List*. The total number of weights is equal to the total number of indices and is in the same order.

Name	Identifier	Data Type
Bone Weights	6013	*POD Data block*

Max. Num. Bones per Batch

The *Max. Num. Bones per Batch* block contains an unsigned 32-bit integer which represents the maximum number of bones per bone batch.

Name	Identifier	Data Type
Max. Num. Bones per Batch	6018	Unsigned 32-bit integer

Bone Batch Index List

The *Bone Batch Index List* block contains a list of indices for *Nodes* in the *Node* list

The *Bone Batch Index List* block contains a list of indices for *Nodes* in the *Node* list. Each indexed *Node* represents the transformation associated with a single bone.

Each batch within the bone batch index list will be a certain number of elements long. The number of elements is set by the value of *Max. Num. Bones per Batch*.

For example, if one bone batch contains eight elements (the maximum number of bones per batch), and another three, the three element array will be padded with zero to eight elements, giving a list of indices 16 elements long. A number of elements from each batch should be read equal to the value in *Num. Bone Indices per Batch* for that batch.

In the above example, the *Num. Bone Indices per Batch* would contain [8, 3]. Eight indices would be read from the first batch within the list, and three from the second.

Finally, there are a number of batches in the *Bone Batch Index List* equal to the value of *Num. Bone Batches*.

Name	Identifier	Data Type	Associated Blocks
Bone Batch Index List	6015	Unsigned 32-bit integer array	*Num. Bone Batches* *Bone Offset per Batch* *Num. Bone Indices per Batch*

Num. Bone Batches

The *Num. Bone Batches* block contains an unsigned 32-bit integer representing the number of bone batches in the *Bone Batch Index List*

The *Num. Bone Batches* block contains an unsigned 32-bit integer representing the number of bone batches in the *Bone Batch Index List*.

Name	Identifier	Data Type
Num. Bone Batches	6019	Unsigned 32-bit integer

Bone Offset per Batch

The *Bone Offset per Batch* block contains a list of integers which each represent the offset into the *Vertex List*, or *Vertex Index List* of the indexed data at which the batch starts

The *Bone Offset per Batch* block contains a list of integers which each represent the offset into the *Vertex List*, or *Vertex Index List* of the indexed data at which the batch starts.

For example, if the list contained [0, 799] the first bone batch would influence vertices 0-798. The second bone batch would influence vertices 799 onwards.

Name	Identifier	Data Type
Bone Offset per Batch	6017	Unsigned 32-bit integer array

Num. Bone Indices per Batch

The *Num. Bone Indices per Batch* block contains a list of integers which each represent the number of indices in each bone batch in the *Bone Batch Index List*

The *Num. Bone Indices per Batch* block contains a list of integers which each represent the number of indices in each bone batch in the *Bone Batch Index List*.

Name	Identifier	Data Type
Num. Bone Indices per Batch	6016	Unsigned 32-bit integer array

Adjacency Index List

The *Adjacency Index List* block contains the indices of the vertices of adjacent triangles

The *Adjacency Index List* block contains the indices of the vertices of adjacent triangles. It contains 6 indices per face. Indexes 0, 2 and 4 are the indices for the triangle vertices. Indexes 1, 3 and 5 point to the vertices in adjacent triangles.

This list is used, for example, in geometry shaders to find silhouettes to be able to extrude shadow volumes.

Name	Identifier	Data Type
Adjacency Index List	6022	POD Data block

Interleaved Data List

The *Interleaved Data List* block contains a list of all vertex data, interleaved on a per-vertex basis.

Meshes within POD files may contain interleaved vertex data. In this situation, the arrays of vertex positions, UVW channels, normal data, and so on, are repurposed.

The POD data blocks that normally contain the vertex positions, UVW Channels, normal data, and so on, will instead contain the position of the first element of the appropriate type within the interleaved data array and a stride. It is possible to read a particular data type for a given element from the interleaved data array by calculating the offset as follows:

$$position_n = position_{initial} + n(stride)$$

It is possible to check for interleaving by checking the size and contents of the *Interleaved Data List*. If the block has a size and contents then the mesh in question is interleaved. If a mesh is interleaved, the following data will be interleaved if present:

- Vertex Data.
- Normal Data.
- Tangent Data.
- Binormal Data.
- UVW Data.
- Vertex Colours.
- Bone Indices.
- Bone Weights.

Name	Identifier	Data Type
Interleaved Data List	6014	Byte array

Node

The *Node* block contains all of the information relating to a single node within the scene

The *Node* block contains all of the information relating to a single node within the scene. Nodes are the basic building blocks of a scene and represent objects in a scene. They can be used as meshes, cameras, or lights and can have associated materials.

They can also have parents so can form part of the scene hierarchy.

The *Identifier* for this block is: **2013**

The *Node* block contains various sub-blocks which are used to specify different properties of the node, including the related material, the node's parent, and the mesh, light, or cameras that is node represents.

This block also contains several sub-block which are listed below.

Name	Description
Node Name	The name of the object (see *Node Name*).
Material Index	The index of the material used with this node, if the node is a mesh (see *Material Index*).
Parent Index	The index of this objects parent in the node array (see *Parent Index*).
Animation Flags	A flag variable that is used to determine the forms of animation the node contains, if any (see *Animation Flags*).
Animation Position Index	A list of indices into the *Animation Position* block, one per frame, used for indexing animation (see *Animation Position Index*).
Animation Position	A list of position animations, in the form of three floats (x, y, z order), per frame when not indexed or applied in the order given by *Animation Position Index* when indexed (see *Animation Position*).
Animation Rotation Index	A list of indices into *Animation Rotation*, one per frame, used for indexing animation (see *Animation Rotation Index*).
Animation Rotation	A list of rotation animations, in the form of a quaternion, per frame when not indexed or applied in the order given by *Animation Rotation Index* when indexed (see *Animation Rotation*).
Animation Scale Index	A list of indices into *Animation Scale*, one per frame, used for indexing animation (see *Animation Scale Index*).
Animation Scale	A list of scaling animations, in the form of seven floats (x, y, z, x-axis, y-axis, z-axis, stretch rotation), per frame when not indexed or applied in the order given by *Animation Scale Index* when indexed. X-Axis, Y-Axis, Z-Axis and Stretch Rotation are used to convert the object into the axes the scaling is performed in (see *Animation Scale*).
Animation Matrix Index	A list of indices into *Animation Matrix*, one per frame, used for indexing animation (see *Animation Matrix Index*).
Animation Matrix	A list of matrix animations, in the form of sixteen floats (4x4), per frame when not indexed or applied in the order given by *Animation Matrix Index* when indexed. Matrices are stored *Row Major* in memory and use *Column Major* mathematically (see *Animation Matrix*).
Node User Data	Custom data added by the exporter (see *Node User Data*).
Node Index	The index of the node into the mesh, light, or camera array, as appropriate (see *Node Index*).

Node Name

The *Node Name* block simply contains the name of the object which the node represents

The *Node Name* block simply contains the name of the object which the node represents.

Name	Identifier	Data Type
Node Name	5001	Null terminated character array

Material Index

The *Material Index* block contains the index of the material that the node is going to use. This is only applicable if the node is a mesh

The *Material Index* block contains the index of the material that the node is going to use. This is only applicable if the node is a mesh.

Name	Identifier	Data Type
Material Index	5002	Signed 32-bit integer

Parent Index

The *Parent Index* block contains the index of the parent of this node in the node array

The *Parent Index* block contains the index of the parent of this node in the node array.

Name	Identifier	Data Type
Parent Index	5003	Signed 32-bit integer

Animation Flags

The *Animation Flags* block contains a series of flags which determine the forms of animation that are present in the node

The *Animation Flags* block contains a series of flags which determine the forms of animation that are present in the node.

The valid flags are:

- 0x01 – Position Animation
- 0x02 – Rotation Animation
- 0x04 – Scale Animation
- 0x08 – Matrix Animation

Name	Identifier	Data Type
Animation Flags	5012	Unsigned 32-bit integer

Animation Position

The *Animation Position* block contains a list of position animations

The *Animation Position* block contains a list of position animations. These position animations are in the form of three floats (XYZ order). The *Animation Position Index* block determines the order in which these positions will be applied, with a maximum number of entries equal to the maximum value within the index. If the *Animation Position Index* is not present then one position is applied per frame.

Name	Identifier	Data Type
Animation Position	5007	Float/Fixed array

Animation Position Index

The *Animation Position Index* block contains a list of indices in the *Animation Position* block which determine the ordering of the position animations

The *Animation Position Index* block contains a list of indices in the *Animation Position* block. These indices determine the order in which the position animations in the *Animation Position* block will be applied.

Name	Identifier	Data Type
Animation Position Index	5013	Signed 32-bit integer array

Animation Rotation

The *Animation Rotation* block contains a list of rotation animation in the form a quaternion

The *Animation Rotation* block contains a list of rotation animation in the form a quaternion. These animations are applied in the order given by the *Animation Rotation Index* block, with a maximum number of entries equal to the maximum values within the index.

If the *Animation Rotation Index* block is not present then these animations are applied one per frame.

Name	Identifier	Data Type
Animation Rotation	5008	Float/Fixed array

Animation Rotation Index

The *Animation Rotation Index* block contains a list of indices which determine what order the rotation animations in the *Animation Rotation* block are applied

The *Animation Rotation Index* block contains a list of indices which determine what order the rotation animations in the *Animation Rotation* block are applied. One of these indices correspond to a single frame.

Name	Identifier	Data Type
Animation Rotation Index	5014	Signed 32-bit integer array

Animation Scale

The *Animation Scale* block contains a list of rotation animations in the form of 7 floats (x, y, z, x-axis, y-axis, z-axis, and stretch rotation)

The *Animation Scale* block contains a list of rotation animations in the form of 7 floats (x, y, z, x-axis, y-axis, z-axis, and stretch rotation). The *Animation Scale Index* block determines the order in which these animations will be applied with a

maximum number of entries equal to the maximum value within the index. If the *Animation Scale Index* block is not present these animations are applied one each per frame.

x-axis, y-axis, z-axis, and stretch rotation are used to convert the object to the axes the scaling is performed in.

Name	Identifier	Data Type
Animation Scale	5019	Float/Fixed array

Animation Scale Index

The *Animation Scale Index* block contains a list of indices in the *Animation Scale* block. These indices are used to order the animation and correspond to one per frame

The *Animation Scale Index* block contains a list of indices in the *Animation Scale* block. These indices are used to order the animation and correspond to one per frame.

Name	Identifier	Data Type
Animation Scale Index	5015	Signed 32-bit integer array

Animation Matrix

The *Animation Matrix* block contains a list of matrix animation in the form of 16 floats (4x4)

The *Animation Matrix* block contains a list of matrix animation in the form of 16 floats (4x4). The *Animation Matrix Index* block determines the order in which these animations are applied, with a maximum number of entries equal to the maximum value within the index. If *Animation Matrix Index* is not present then the animations are simply applied one per frame.

Matrices are stored *Row Major* in memory and used *Column Major* mathematically.

Name	Identifier	Data Type
Animation Matrix	5010	Float/Fixed array

Animation Matrix Index

The *Animation Matrix Index* block contains a list of indices in the *Animation Matrix* block

The *Animation Matrix Index* block contains a list of indices in the *Animation Matrix* block. These indices determine the ordering the animations in *Animation Matrix*. One index corresponds to one frame.

Name	Identifier	Data Type
Animation Matrix Index	5016	Signed 32-bit integer array

Node User Data

The *Node User Data* block contains custom data which can be added by the exporter.

The *Node User Data* block contains custom data which can be added by the exporter.

The format of this data is undefined.

Name	Identifier	Data Type
User Data	5017	Variable

Node Index

The *Node Index* block contains the index of the mesh, light, or camera in their associated arrays

The *Node Index* block contains the index of the mesh, light, or camera in their associated arrays. The mesh, light, or camera which corresponds to this index will be object associated with this node.

Name	Identifier	Data Type
Node Index	5000	Signed 32-bit integer

Texture

The *Texture* block contains all of the information relating to a single texture used within the scene.

The *Texture* block contains all of the information relating to a single texture used within the scene.

The *Identifier* for this block is: **2014**

This block contains a sub-block which is used to hold filename of the required texture.

Name	Description
Texture Name	The filename of the texture (see Section *Texture Name*).

Texture Name

The *Texture Name* block within the *Texture* block is used to hold the name of the texture file which is used by the texture

The *Texture Name* block within the *Texture* block is used to hold the name of the texture file which is used by the texture.

This block does not contain the full file path.

Name	Identifier	Data Type
Texture Name	4000	Null terminated character array

Material

The *Material* block contains all of the information relating to a single material within the scene

The *Material* block contains all of the information relating to a single material within the scene.

The *Identifier* for this block is: **2015**

This block contains many different sub-blocks which contains information about the material. These blocks include the indices of the scene's textures in the texture list and information about blending.

These sub-blocks are listed below:

Name	Description
Diffuse Texture Index	The index of the diffuse texture into the scenes texture list (see *Diffuse Texture Index*).
Ambient Texture Index	The index of the ambient texture into the scenes texture list (see *Ambient Texture Index*).
Specular Colour Texture Index	The index of the specular colour texture into the scenes texture list (see *Specular Colour Texture Index*).
Specular Level Texture Index	The index of the specular level texture into the scenes texture list (see *Specular Level Texture Index*).
Bump Map Texture Index	The index of the bump map texture into the scenes texture list (see *Bump Map Texture Index*).
Emissive Texture Index	The index of the emissive texture into the scenes texture list. (see *Emissive Texture Index*).
Glossiness Texture Index	The index of the glossiness texture into the scenes texture list (see *Glossiness Texture Index*).
Opacity Texture Index	The index of the opacity texture into the scenes texture list. (see *Opacity Texture Index*).
Reflection Texture Index	The index of the reflection texture into the scenes texture list. (see *Reflection Texture Index*).
Refraction Texture Index	The index of the refraction texture into the scenes texture list (see *Refraction Texture Index*).
Material Opacity	The opacity of the material (see *Material Opacity*).
Ambient Colour	The ambient colour of the material (see *Ambient Colour*).
Diffuse Colour	The diffuse colour of the material (see *Diffuse Colour*).
Specular Colour	The specular colour of the material (see *Specular Colour*).
Shininess	The shininess of the material (see *Shininess*).
Effect File Name	The name of the effect file used by the material (see *Effect File Name*).
Effect Name	The name of the effect within the file *Effect File Name*. (see *Effect Name*).
Blending RGB Source Value	The first RGB data source, with an optional pre-blend operation (see *Blending RGB Source Value*).
Blending Alpha Source Value	The first alpha data source, with an optional pre-blend operation (see *Blending Alpha Source Value*).
Blending RGB Destination Value	The second RGB data source, with an optional pre-blend operation (see *Blending RGB Destination Value*).
Blending Alpha Destination Value	The second alpha data source, with an optional pre-blend operation (see *Blending Alpha Destination Value*).
Blending RGB Operation	The blending operation defining how the materials RGB data sources should be combined (see *Blending RGB Operation*).

Name	Description
Blending Alpha Operation	The blending operation defining how the materials alpha data sources should be combined (see *Blending Alpha Operation*).
Blending RGBA Colour	An RGBA colour used with some blend operations (see *Blending RGBA Colour*).
Blending Factor Array	A factor value for used with some blend operations (see *Blending Factor Array*).
Material Flags	Specifies whether a number of flags are set within the POD file (see *Material Flags*).
Material User Data	Custom data added by the exporter (see *Material User Data*).
Material Name	The name of the material (see *Material Name*).

Extended PBR Material Data

The table below contains a list of additional blocks which contain material properties which are useful when implementing physically-based rendering (PBR).

Name	Description
Metallicity	Metallicity should mix between two shading models. When metallicity = 0.0, a non-metal material should be rendered utilizing the base colour as a diffuse component, adding reflections on top of this with sub-surface scattering and transparency. When metallicity = 1.0, a metallic material should be rendered that is only reflective, reflecting the base colour for facing angles and the reflection colour (generally white) on edges (see *Metallicity*).
Roughness	This property measures the roughness of the associated material. A higher roughness yields a blurrier material, a lower roughness yields a more mirror-like material (see *Roughness*).
IOR	The Index of Refraction level, defining both how much rays bend when entering the medium, but also the angular dependency of reflectivity (see *IOR*).
Reflectivity	The relative measurement of reflectivity (see *Reflectivity*).
SubSurface Scattering	The relative measurement of sub-surface scattering (see *SubSurface Scattering*).
SubSurface Scattering Depth	The amount of spread of light in the material (see *SubSurface Scattering Depth*).
SubSurface Scattering Colour	The colour of the sub-surface scattering, generally the same as the base colour (see *SubSurface Scattering Colour*).

Name	Description
Emission	The relative measurement of self-illumination (see *Emission*).
Emission Luminance	The surface's luminosity, in cd/m^2 (also known as "nits") (see *Emission Luminance*).
Metallicity Texture Index	The index of the metallicity texture into the scenes texture list (see *Metallicity Texture Index*).
Roughness Texture Index	The index of the roughness texture into the scenes texture list (see *Roughness Texture Index*).

Diffuse Texture Index

The *Diffuse Texture Index* block contains a 32-bit signed integer representing the index of the diffuse texture in the texture list of the scene

The *Diffuse Texture Index* block contains a 32-bit signed integer representing the index of the diffuse texture in the texture list of the scene.

Name	Identifier	Data Type
Diffuse Texture Index	3001	Signed 32-bit integer

Ambient Texture Index

The *Ambient Texture Index* block contains a 32-bit signed integer representing the index of the ambient texture in the texture list of the scene

The *Ambient Texture Index* block contains a 32-bit signed integer representing the index of the ambient texture in the texture list of the scene.

Name	Identifier	Data Type
Ambient Texture Index	3009	Signed 32-bit integer

Specular Colour Texture Index

The *Specular Colour Texture Index* block contains a 32-bit signed integer representing the index of the specular colour texture in the texture list of the scene.

The *Specular Colour Texture Index* block contains a 32-bit signed integer representing the index of the specular colour texture in the texture list of the scene.

Name	Identifier	Data Type
Specular Colour Texture Index	3010	Signed 32-bit integer

Specular Level Texture Index

The *Specular Level Texture Index* block contains a 32-bit signed integer representing the index of the specular level texture in the texture list of the scene

The *Specular Level Texture Index* block contains a 32-bit signed integer representing the index of the specular level texture in the texture list of the scene.

Name	Identifier	Data Type
Specular Level Texture Index	3011	Signed 32-bit integer

Bump Map Texture Index

The *Bump Map Texture Index* block contains a 32-bit signed integer representing the index of the bump map texture in the texture list of the scene

The *Bump Map Texture Index* block contains a 32-bit signed integer representing the index of the bump map texture in the texture list of the scene.

Name	Identifier	Data Type
Bump Map Texture Index	3012	Signed 32-bit integer

Emissive Texture Index

The *Emissive Texture Index* block contains a 32-bit signed integer representing the index of the emissive texture in the texture list of the scene

The *Emissive Texture Index* block contains a 32-bit signed integer representing the index of the emissive texture in the texture list of the scene.

Name	Identifier	Data Type
Emissive Texture Index	3013	Signed 32-bit integer

Glossiness Texture Index

The *Glossiness Texture Index* block contains a 32-bit signed integer representing the index of the glossiness texture in the texture list of the scene

The *Glossiness Texture Index* block contains a 32-bit signed integer representing the index of the glossiness texture in the texture list of the scene.

Name	Identifier	Data Type
Glossiness Texture Index	3014	Signed 32-bit integer

Opacity Texture Index

The *Opacity Texture Index* block contains a 32-bit signed integer representing the index of the opacity texture in the texture list of the scene.

The *Opacity Texture Index* block contains a 32-bit signed integer representing the index of the opacity texture in the texture list of the scene.

Name	Identifier	Data Type
Opacity Texture Index	3015	Signed 32-bit integer

Reflection Texture Index

The *Reflection Texture Index* block contains a 32-bit signed integer representing the index of the reflection texture in the texture list of the scene

The *Reflection Texture Index* block contains a 32-bit signed integer representing the index of the reflection texture in the texture list of the scene

Name	Identifier	Data Type
Reflection Texture Index	3016	Signed 32-bit integer

Refraction Texture Index

The *Refraction Texture Index* contains a 32-bit signed integer representing the index of refraction texture in the texture list of the scene

The *Refraction Texture Index* contains a 32-bit signed integer representing the index of refraction texture in the texture list of the scene.

Name	Identifier	Data Type
Refraction Texture Index	3017	Signed 32-bit integer

Material Opacity

The *Material Opacity* block contains the opacity of the associated material

The *Material Opacity* block contains the opacity of the associated material in the form of a float.

Table 4: Material Opacity Data

Name	Identifier	Data Type	Description
Material Opacity	3002	Float/Fixed	The opacity of the material.

Ambient Colour

The *Ambient Colour* block contains the ambient colour of the associated material

The *Ambient Colour* block contains the ambient colour of the associated material in the form of three channels in the order RGB.

Name	Identifier	Data Type
Ambient Colour	3003	Float/Fixed

Diffuse Colour

The *Diffuse Colour* block contains the diffuse colour of the associated material

The *Diffuse Colour* block contains the diffuse colour of the associated material in the form of three channels in the order RGB.

Name	Identifier	Data Type
Diffuse Colour	3004	Float/Fixed

Specular Colour

The *Specular Colour* block contains the specular colour of the associated material

The *Specular Colour* block contains the specular colour of the associated material. This data is in the form of three channels in the order RGB.

Name	Identifier	Data Type
Specular Colour	3005	Float/Fixed

Shininess

The *Shininess* block the shininess of the associated material

The *Shininess* block the shininess of the associated material

Name	Identifier	Data Type
Shininess	3006	Float/Fixed

Effect File Name

The *Effect File Name* contains the name of the effect file used by the associated material

The *Effect File Name* contains the name of the effect file used by the associated material.

Name	Identifier	Data Type
Effect File Name	3007	Null terminated character array

Effect Name

The *Effect Name* contains the name of the effect in the effect file used by the material

The *Effect Name* contains the name of the effect in the effect file used by the material.

Name	Identifier	Data Type
Effect Name	3008	Null terminated character array

Blending RGB Operation

The *Blending RGB Operation* block defines how the RGB data sources of the material should be combined

The *Blending RGB Operation* block defines how the RGB data sources of the material should be combined. The RGB data sources for a material are contained within the *Blending RGB Source Value* and the *Blending RGB Destination Value* blocks.

The valid values for the data in this block are:

- 0 – ZERO
- 1 – ONE
- 2 – BLEND_FACTOR
- 3 – ONE_MINUS_BLEND_FACTOR
- 0x8001 – CONSTANT_COLOUR
- 0x8002 – ONE_MINUS_CONSTANT_COLOUR
- 0x8006 – ADD
- 0x8007 – MIN
- 0x8008 – MAX
- 0x800a – SUBTRACT
- 0x800b – REVERSE_SUBTRACT

Table 5: Blending RGB Operation Data

Name	Identifier	Data Type
Blending RGB Operation	3022	Unsigned 32-bit integer

Blending RGB Source Value

The *Blending RGB Source Value* contains the first RGB data for blending operations. It also has an optional pre-blend operation.

The following values are valid operations:

- 0x0300 - SRC_COLOR
- 0x0301 - ONE_MINUS_SRC_COLOR

Name	Identifier	Data Type
Blending RGB Source Value	3018	Unsigned 32-bit integer

Blending RGB Destination Value

The *Blending RGB Destination Value* block contains the second RGB data source for blending operations

The Blending RGB Destination Value block contains the second RGB data source for blending operations. It also has an optional pre-blend operation.

The following values are valid operations:

- 0x0306 - DST_COLOR
- 0x0307 - ONE_MINUS_DST_COLOR

Name	Identifier	Data Type
Blending RGB Destination Value	3020	Unsigned 32-bit integer

Blending Alpha Operation

The *Blending Alpha Operation* block contains information specifying the blending operation. This is how the alpha data source of the material should be combined. The alpha data sources for a material are contained within the *Blending Alpha Source Value* and the *Blending Alpha Destination Value* blocks.

The valid values for this block are:

- 0 – ZERO
- 1 – ONE
- 2 – BLEND_FACTOR
- 3 – ONE_MINUS_BLEND_FACTOR
- 0x8003 – CONSTANT_ALPHA
- 0x8004 – ONE_MINUS_CONSTANT_ALPHA
- 0x8006 – ADD
- 0x8007 – MIN

- 0x8008 – MAX
- 0x800a – SUBTRACT
- 0x800b – REVERSE_SUBTRACT

Name	Identifier	Data Type
Blending Alpha Operation	3023	Unsigned 32-bit integer

Blending Alpha Source Value

The *Blending Alpha Source Value* contains the first alpha data source for a blending operation

The *Blending Alpha Source Value* contains the first alpha data source for a blending operation and provides an optional pre-blend operation.

The following values are valid operations:

- 0x0302 - SRC_ALPHA
- 0x0303 - ONE_MINUS_ALPHA

Name	Identifier	Data Type
Blending Alpha Source Value	3019	Unsigned 32-bit integer

Blending Alpha Destination Value

The *Blending Alpha Destination Value* block contains the second alpha data source with an optional pre-blend operation

The *Blending Alpha Destination Value* block contains the second alpha data source with an optional pre-blend operation.

The following values are valid operations:

- 0x0304 - DST_ALPHA
- 0x0305 - ONE_MINUS_DST_ALPHA

Name	Identifier	Data Type
Blending RGB Destination Value	3021	Unsigned 32-bit integer

Blending Factor Array

The *Blending Factor Array* block contains a list of blend factors for the associated material

The *Blending Factor Array* block contains a list of blend factors for the associated material. There will be one per colour in the *Blending RGBA Colour* block used for some blending operations.

Name	Identifier	Data Type
Blending Factor Array	3025	Float/Fixed

Blending RGBA Colour

The *Blending RGBA Colour* block contains an RGBA colour used with some blending operations

The *Blending RGBA Colour* block contains an RGBA colour used with some blending operations. This block contains four floats in the order RGBA.

Name	Identifier	Data Type
Blending RGBA Colour	3024	Float/Fixed

Material Flags

The *Material Flags* block specifies whether a number of flags are set within the POD file.

The *Material Flags* block specifies whether a number of flags are set within the POD file.

The allowed values of this flag are:

- 0x01 - Blending enabled
- 0x00 - Blending disabled

Table 6: Material Flags Data

Name	Identifier	Data Type
Material Flags	3026	Unsigned 32-bit integer

Material User Data

The *Material User Data* block contains custom data added by the exporter

The *Material User Data* block contains custom data added by the exporter. The format of this data is undefined.

Table 7: Material User Data

Name	Identifier	Data Type
User Data	3027	Variable

Material Name

The *Material Name* block simply contains the name of the material as a character array

The *Material Name* block simply contains the name of the material as a character array.

Name	Identifier	Data Type
Material Name	3000	Null terminated character array

Metallicity

The *Metallicity* block defines the metallicity of the associated material

The *Metallicity* block defines the metallicity of the associated material. The metallicity of a material determines how to two shading models should be mixed. Materials with metallicity = 0.0 should reflect light perfectly diffusely while materials with metallicity = 1.0 should reflect light specularly, like a metal. A material with a

metallicity between these two values should interpolate between these two models of shading.

Name	Identifier	Data Type
Metallicity	3028	Float

Roughness

The *Roughness* block defines the level of roughness of the associated material

The *Roughness* block defines the level of roughness of the associated material. The roughness property determines how blurry the material will be. The lower, the roughness the closer the material becomes to being mirror-like.

Name	Identifier	Data Type
Roughness	3029	Float

IOR

The IOR (*Index of Refraction*) block defines the index of refraction level which measures how much rays bend when entering the medium

The IOR (*Index of Refraction*) block defines the index of refraction level which measures how much rays bend when entering the medium, but also the angular dependency of the reflectivity

Name	Identifier	Data Type
IOR	3030	Float

Reflectivity

The *Reflectivity* block specifies the reflectivity of the associated material

The *Reflectivity* block specifies the reflectivity of the associated material.

Name	Identifier	Data Type
Reflectivity	3032	Float

SubSurface Scattering

The *SubSurface Scattering* block contains information about the amount of sub-surface scattering of the associated material.

Table 8: Material Data

Name	Identifier	Data Type
SubSurface Scattering	3033	Float

SubSurface Scattering Depth

The *SubSurface Scattering Depth* block contains information on the amount of spread of light in the associated material.

The *SubSurface Scattering Depth* block contains information on the amount of spread of light in the associated material.

Name	Identifier	Data Type
SubSurface Scattering Depth	3034	Float

SubSurface Scattering Colour

The *SubSurface Scattering Colour* block contains information about the sub-surface scattering colour of the material.

The *SubSurface Scattering Colour* block contains information about the sub-surface scattering colour of the material. This colour is composed of three channels in the order RGB.

Name	Identifier	Data Type
SubSurface Scattering Colour	3035	Float

Emission

The *Emission* block contains information about the amount of self-illumination of a material

The *Emission* block contains information about the amount of self-illumination of a material.

Table 9: Emission Data

Name	Identifier	Data Type
Emission	3036	Float

Emission Luminance

The *Emission Luminance* block contains information about the surface's luminosity in a material

The *Emission Luminance* block contains information about the surface's luminosity in a material. The luminosity is in cd/m^2.

Table 10: Emission Luminance Data

Name	Identifier	Data Type
Emission Luminance	3037	Float

Metallicity Texture Index

The *Metallicity Texture Index* block contains information on the index of the metallicity texture into the scenes texture list

The *Metallicity Texture Index* block contains information on the index of the metallicity texture into the scenes texture list.

Table 11: Metallicity Texture Data

Name	Identifier	Data Type
Metallicity Texture Index	3040	Signed 32-bit integer

Roughness Texture Index

The *Roughness Texture Index* block contains information about on the index of the roughness texture into the scenes texture list.

The *Roughness Texture Index* block contains information about on the index of the roughness texture into the scenes texture list.

Table 12: Roughness Texture Index Data

Name	Identifier	Data Type
Roughness Texture Index	3041	Signed 32-bit integer

Scene Flags

The *Scene Flags* block is used to specify whether a number of flags are set within the POD file.

The *Scene Flags* block is used to specify whether a number of flags are set within the POD file.

Name	Identifier	Data Type	Allowed Values
Scene Flags	2016	Unsigned 32-bit integer	The values are as follows: • 0x00000001 – The fixed 16.16 data type is used

FPS

The FPS block specifies the animation speed of the scene in frames per second.

The FPS block specifies the animation speed of the scene in frames per second.

Table 13: FPS Data

Name	Identifier	Data Type
FPS	2017	Unsigned 32-bit integer

Scene User Data

The *Scene User Data* block contains custom data added by the exporter.

The *Scene User Data* block contains custom data added by the exporter. The format of this data is undefined.

Name	Identifier	Data Type
User Data	2018	Variable

Units

The *Units* block is used to specify the number of metres that a single unit of length represents

The *Units* block is used to specify the number of metres that a single unit of length represents. For example, a vertex at the coordinates (0, 0, 0) moving to (0, 0, 1) has moved 1 unit along the Z-axis.

Name	Identifier	Data Type
Units Data	2019	Variable

5. Contact Details

For further support, visit our forum:

http://forum.imgtec.com

Or file a ticket in our support system:

https://pvrsupport.imgtec.com

For general enquiries, please visit our website:

http://imgtec.com/corporate/contactus.asp