



PVRGeoPOD

User Manual

Copyright © Imagination Technologies Limited. All Rights Reserved.

This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Imagination Technologies and the Imagination Technologies logo are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.

Filename : PVRGeoPOD.User Manual
Version : PowerVR SDK REL_3.1@2284953a External Issue
Issue Date : 15 Feb 2013
Author : Imagination Technologies Limited

Contents

1. Introduction	4
1.1. Supported Features	4
1.2. Limitations.....	4
2. Compatibility	5
2.1. 3D Studio Max	5
2.2. Maya	5
2.3. Blender	5
3. Installation	6
3.1. From Installer	6
3.2. Manual Installation.....	6
3.2.1. Installing Qt	6
3.2.2. Installation for 3D Studio Max	6
3.2.3. Installation for Maya	7
3.2.4. Installation for Blender.....	7
4. Exporting/Converting	8
4.1. Available File Formats	8
4.1.1. POD File	8
4.1.2. Header/Source File	8
4.1.3. POD vs. Header/Source.....	8
4.2. Exporting a Scene from 3DS Max	9
4.3. Exporting a Scene from Maya	10
4.4. Exporting a Scene from Blender.....	10
4.5. Converting from a Collada File	11
4.5.1. Limitations	12
4.5.2. Command Line	12
5. PVRGeoPOD Options	13
5.1. Overview	13
5.2. Geometry Options.....	14
5.2.1. General Options	15
5.2.2. Vertex Data Options	16
5.3. Material Options.....	19
5.3.1. General Options	19
5.4. Transformation Options	20
5.4.1. General Options	21
5.5. User Options.....	22
5.5.1. General Options	23
5.6. Using Profiles.....	24
6. MaxScript/MEL User Data	25
6.1. Overview	25
6.2. Mel Script.....	25
6.3. MAX Script.....	26
7. Using POD Data	27
7.1. PVRShaman	27
7.2. PowerVR Insider SDK	27
7.2.1. Example: Loading a POD file	27
7.2.2. Example: Loading a Header/Source file.....	27
8. Related Material	28
9. Contact Details.....	29
Appendix A. Block Names.....	30
Appendix B. PVRGeoPOD Command Line Options	32

List of Figures

Figure 4-1 3DSMax Export Menu	9
Figure 4-2 PVRGeoPOD Standalone Main Window.....	11
Figure 5-1 PVRGeoPOD Main Window	13
Figure 5-2 Option Tooltip	13
Figure 5-3 Geometry Options.....	14
Figure 5-4 Vertex Data Options	16
Figure 5-5 Basic Vertex Data Options	17
Figure 5-6 Skinning Data Options	17
Figure 5-7 Mapping Channel Options	18
Figure 5-8 Tangent Space Options.....	18
Figure 5-9 Material Options.....	19
Figure 5-10 Transformation Options	20
Figure 5-11 User Options.....	22
Figure 5-12 Profile Window.....	24

1. Introduction

PVRGeoPOD is an exporter plugin for Maya, 3D Studio Max and Blender that exports 3D geometry/scene data to PowerVR's optimized deployment, POD file format; as well as a standalone application designed to convert from the Collada file format (DAE) to the POD file format. This standalone version comes in both a graphical and command line version and is provided for where a developer's 3D modeller does not have a compatible plugin.

1.1. Supported Features

Some of the notable features include:

- Skinned meshes
- Bone batching based on matrix palette size
- Data format choice (store data as float, byte, uint etc.)
- Interleaving of vertex data if desired
- Mesh instancing
- Multiple texture co-ordinate sets
- Parented nodes
- Polygon stripping
- Polygon/vertex sorting
- Tangent space generation
- Custom user data (Plugins Only)

1.2. Limitations

The following are not supported:

- UV animation
- Vertex animation
- Splines
- NURBs
- Per object culling information
- Physique Modifier (3D Studio Max only)

2. Compatibility

2.1. 3D Studio Max

Plug-in	3DS MAX release	"igame.dll" version
3dsmax\12\Windows_x86_32\PVRGeoPOD.dle	3DS MAX 2010	12.0.0.106
	3DS MAX 2011	13.0.0.94 13.0.0.104
	3DS MAX 2012	14.0.0.121
3dsmax\12\Windows_x86_64\PVRGeoPOD.dle	3DS MAX 2010 (x64)	12.0.0.106
	3DS MAX 2011 (x64)	13.0.0.94 13.0.0.104
	3DS MAX 2012 (x64)	14.0.0.121
3dsmax\15\Windows_x86_32\PVRGeoPOD.dle	3DS MAX 2013	15.0.0.347
3dsmax\15\Windows_x86_64\PVRGeoPOD.dle	3DS MAX 2013 (x64)	15.0.0.347

2.2. Maya

Maya version	Windows		MacOS	
	32-bit	64-bit	32-bit	64-bit
2010	✓	✓	✓	
2011	✓	✓	✓	✓
2012	✓	✓		✓
2013	✓	✓		✓

2.3. Blender

Supported versions:

- 2.59+

3. Installation

3.1. From Installer

Download the PowerVR Insider SDK and follow the on-screen instructions. Once the package has successfully installed, browse to:

```
<InstallDir>\PVRGeoPOD\
```

This folder will contain the plugin files. Once these files have been located they must be installed into the plugin system for the respective application. Installation instructions for each of the supported platforms are include below for cases when the user has decided not to auto-install the plugins using the PowerVR Insider SDK Installer; or for cases where the installer was unable to successfully install said plugins.

3.2. Manual Installation

3.2.1. Installing Qt

Some plugins are bundled with a series of Qt libraries, available in the same folder as the plugin. It is important that these Qt libraries be placed in the correct location so that the GUI for the plugin will function correctly. If no Qt libraries are present then they are no required and only the plugin should be copied.

For Windows the libraries must be placed local to the Max/Maya/Blender executable.

For Linux the libraries location must be set with LD_LIBRARY_PATH

For OSX the libraries must be placed in the same folder as the plugin.

3.2.2. Installation for 3D Studio Max

Copy: `PVRGeoPOD.dle

To: <3DSMAX_DIR>\plugins\

Copy: Qt Dlls

To: <3DSMAX_DIR>\

3.2.3. Installation for Maya

Windows

Copy: <VERSION>\Windows_x86_*\PVRGeoPOD_v<VERSION>.ml1, and any Qt files

To: <MAYA_DIR>\bin\plug-ins\

Copy: Qt Dlls

To: <MAYA_DIR>\

Linux

Copy: <VERSION>/Linux_x86_32/PVRGeoPOD_v<VERSION>.so

To: <MAYA_DIR>/bin/plug-ins/

Install: Qt 4.8.1

OSX

Copy <VERSION>/MacOS_x86*/PVRGeoPOD_v<VERSION>.bundle, and any Qt files

To: /Users/Shared/Autodesk/maya/<MAYA_VERSION>/plug-ins/

Once the plugin has been installed it must be activated in the Maya Plugin Manager.

3.2.4. Installation for Blender

Windows

Copy: 'PVRGeoPOD.dll', 'PVRGeoPODScript.py', and any Qt files

To: Blender Add-ons Folder (run bpy.utils.script_paths("addons") for location)

Linux

Copy: 'PVRGeoPOD.so', 'PVRGeoPODScript.py', and any Qt files

To: Blender Add-ons Folder (run bpy.utils.script_paths("addons") for location)

OSX

Copy: 'PVRGeoPOD.dylib', 'PVRGeoPODScript.py', and any Qt files

To: Blender Add-ons Folder (run bpy.utils.script_paths("addons") for location)

Once these files are copied the plugin must be activated in the user preferences, under the section 'Add-Ons'.

4. Exporting/Converting

4.1. Available File Formats

4.1.1. POD File

PVRGeoPOD exports data to POD files. The PowerVR SDK Tools library contains functions and classes to use POD files in applications. See Section 7 Using POD Data for some examples of their use.

4.1.2. Header/Source File

PVRGeoPOD can export the binary POD file directly to a C++ header file (H) or a C++ source file (CPP) as if it had been wrapped using the PowerVR SDK FileWrap utility (See the File Wrap User Reference Manual for further information). The PowerVR SDK contains tools to use POD file data from a header or source file. Further information can be found on in Section 7 Using POD Data.

4.1.3. POD vs. Header/Source

Memory Footprint

POD files have a smaller memory footprint. Once the POD file has been copied into memory the vertex data can be copied into hardware friendly buffers (e.g. Vertex Buffer Object/ Vertex Buffer). As the graphics API has its own copy of the data, the original data can be released, either the whole POD file, or just the vertex data if other information from the POD file is still required.

Flexibility

A POD file can be changed post-compilation where as a header/source file must be recompiled into the final binary.

Storage

In situations where the POD file is compiled into the final executable there will be little size difference between methods. However, a POD file is smaller on disk than a header/source file containing the same information.

4.2. Exporting a Scene from 3DS Max

Once PVRGeoPOD is installed, open the .max file you wish to convert in 3DSMax and click on 'Main Menu->Export'.

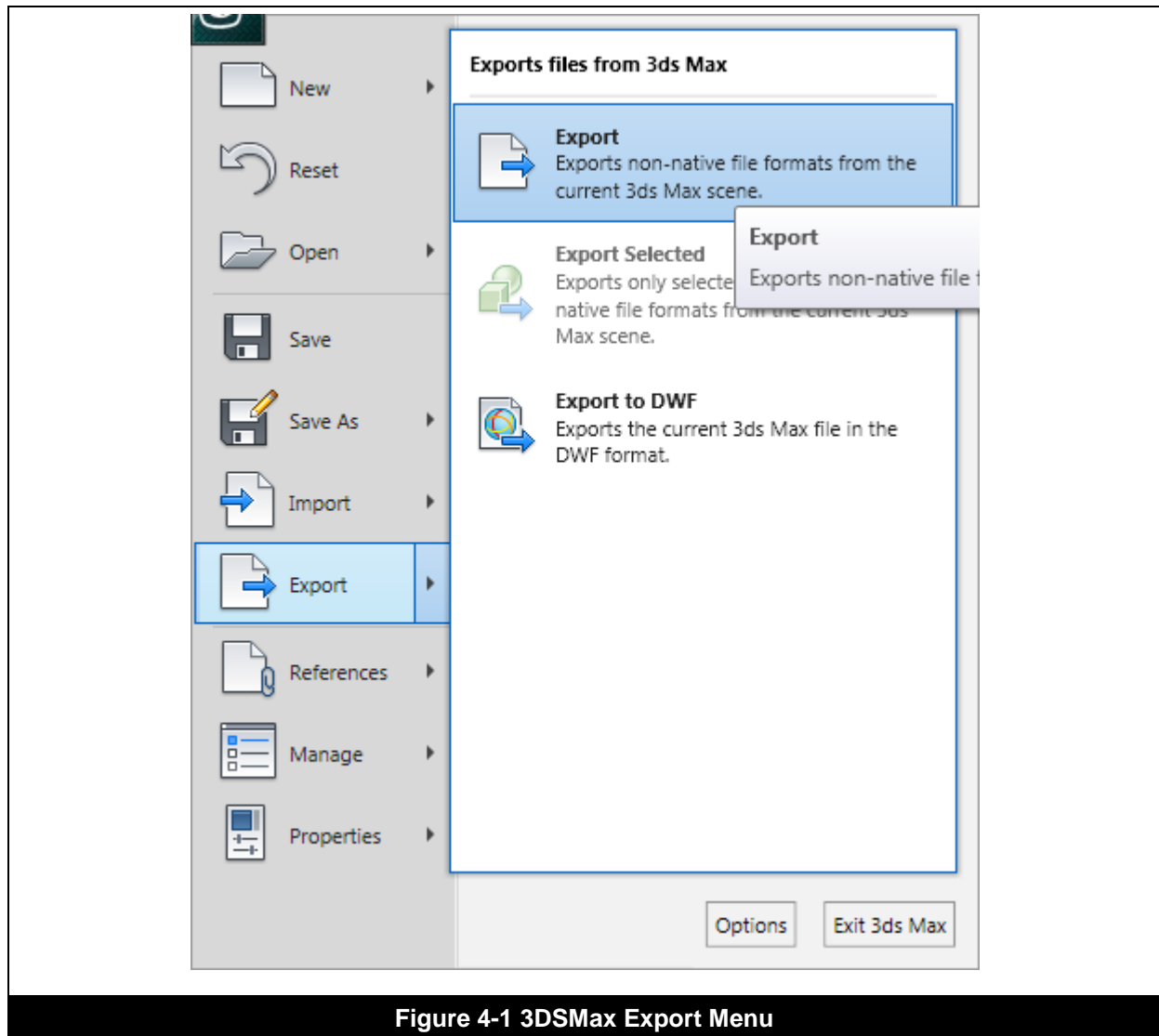


Figure 4-1 3DSMax Export Menu

4.3. Exporting a Scene from Maya

Once PVRGeoPOD is installed and has been activated in the Maya Plug-in manager you can export to POD by clicking on 'File -> Export All' to export an entire scene or 'File -> Export Selection' to export part of it.

It is also possible to export using a MEL script. The MEL commands that can be used are as follows:

```
PVRGeoPOD_Export("filename.pod")
```

This will export the scene to 'filename.pod' using the previously defined export options.

```
PVRGeoPOD_Export("filename.pod", "optionsfilename.txt")
```

This will export the scene to 'filename.pod' using the export options defined in 'optionsfilename.txt'.

```
PVRGeoPOD_Export("filename.pod" "optionsfilename.txt", "selected")
```

This will export the currently selected meshes to 'filename.pod' using the export options defined in 'optionsfilename.txt'.

4.4. Exporting a Scene from Blender

Once PVRGeoPOD is installed and has been activated in the Plug-in manager open the .blend file you wish to convert in Blender, click on

'File -> Export -> PVRGeoPOD (.pod/.h/.cpp)'.

4.5. Converting from a Collada File

In order to convert a DAE file to a POD file click 'File->Open...' navigate to the Collada file that is to be converted and open it, set the export options as desired then click 'export'.

For a full breakdown of the available options see Section 5 PVRGeoPOD Options.

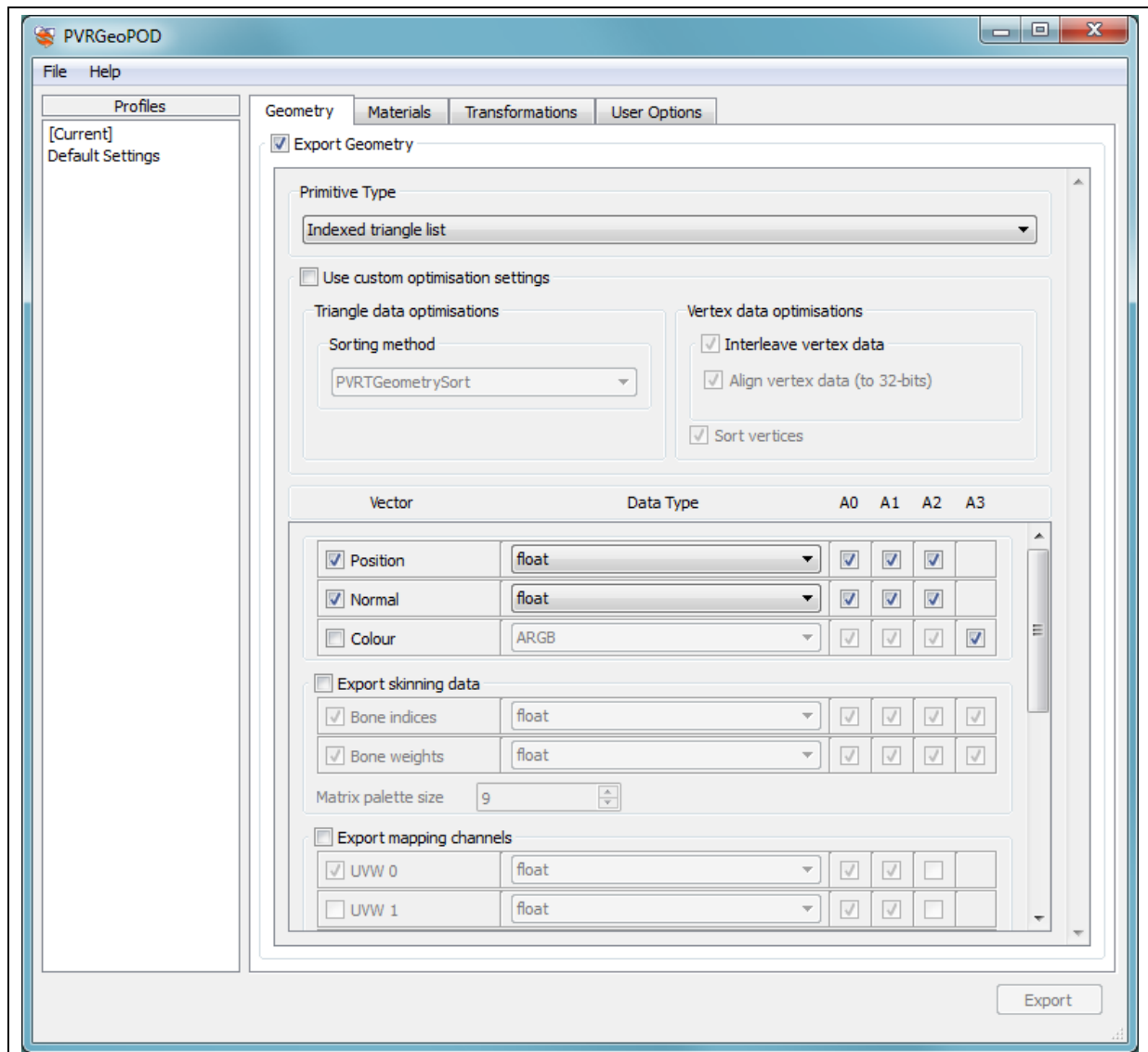


Figure 4-2 PVRGeoPOD Standalone Main Window

4.5.1. Limitations

- Only meshes constructed from triangles, convex polygons, and polylists are supported.
- Animations represented by curves are not supported.
- Skinned animations may not always export correctly; this varies by exporter but has been confirmed to work with OpenCOLLADA for 3D Studio Max.

4.5.2. Command Line

The PVRGeoPOD Standalone command line utility can be called directly as follows:

```
PVRGeoPODCL -i=<input file> -o=<output file> -cs=<ogl, d3d>
```

All of the options available in the GUI utility are available in the command line version via a series of flags. A full list of these flags can be found in Appendix B PVRGeoPOD Command Line Options; or from the command line as follows:

```
PVRGeoPODCL /?
```

5. PVRGeoPOD Options

5.1. Overview

Each option in Figure 5-1 will be documented in this chapter however, for ease of use, tooltips are also implemented as can be seen in Figure 5-2.

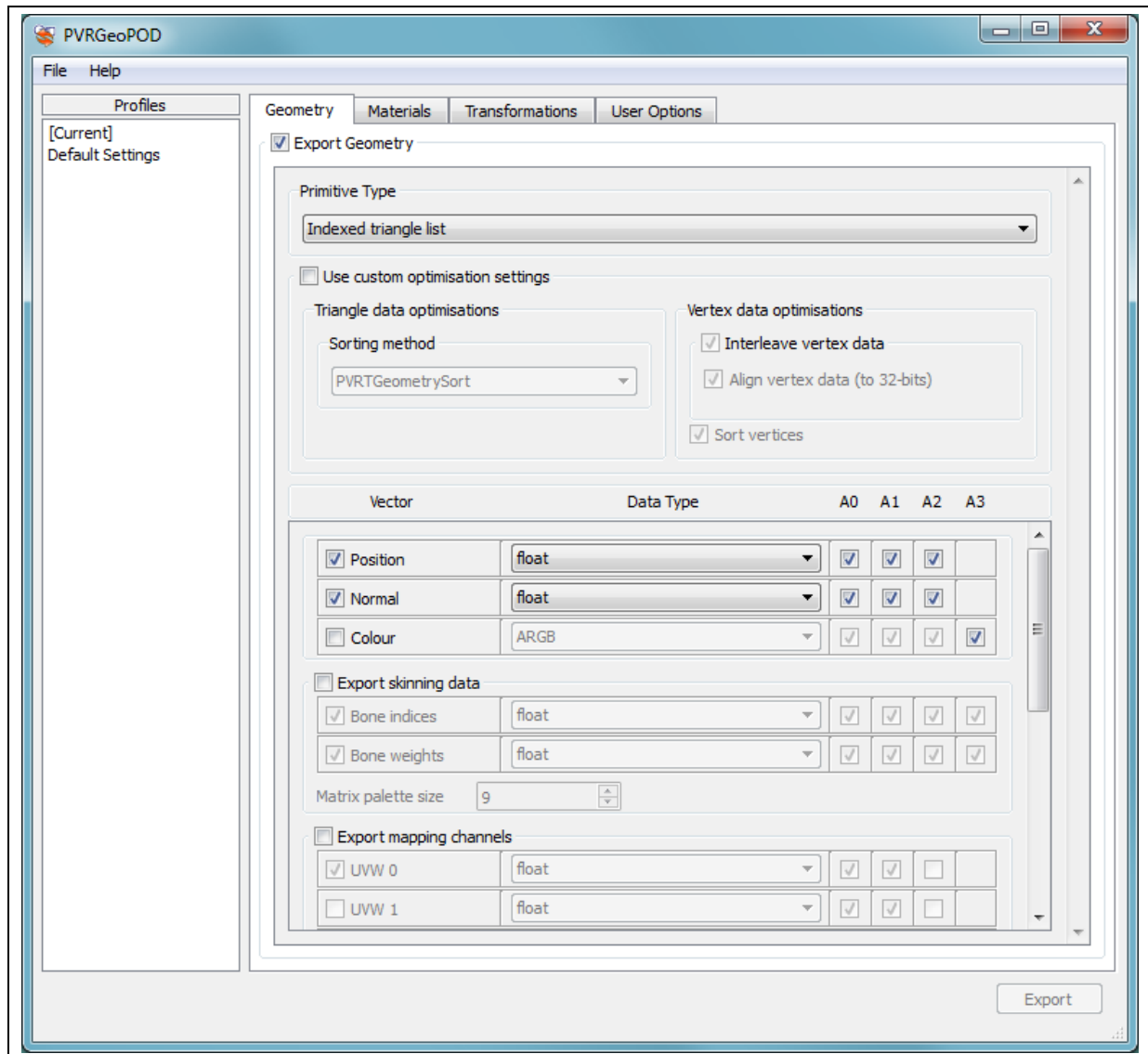


Figure 5-1 PVRGeoPOD Main Window

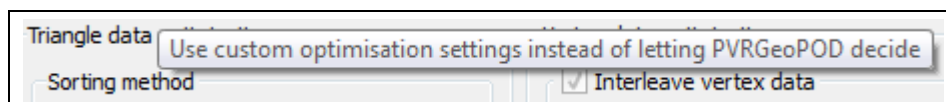


Figure 5-2 Option Tooltip

5.2. Geometry Options

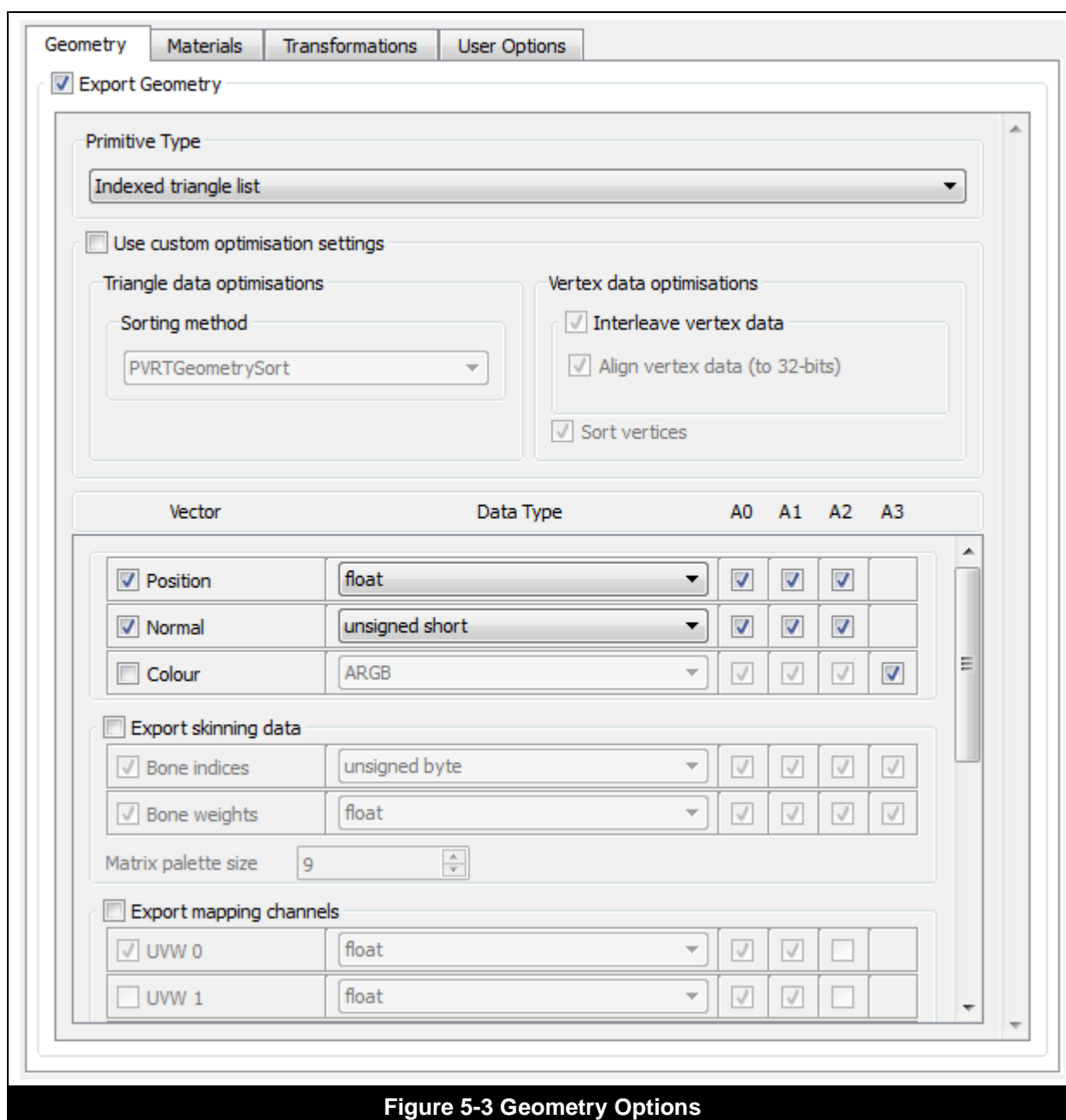


Figure 5-3 Geometry Options

5.2.1. General Options

‘Export Geometry’

This button must be ticked for geometry information to be exported

‘Primitive Type’

When targeting PowerVR hardware a performance improvement can be gained by ensuring the primitives are in the form of a `‘Triangle List’` that is `‘Indexed’`. The other option, `‘Triangle Strips’`, produces additional draw calls and thus can introduce unwanted overhead.

‘Use Custom Optimisation Settings’

PVRGeoPOD has a number of default export settings that are recommended for most use cases. If any of these settings require changing this option must be ticked.

Triangle Data Optimisations

‘Sorting Method’

The correct `‘Sorting Method’` can provide a performance improvement; the specific sorting algorithm to use varies with platform, however, the following are suggested:

- If PowerVR MBX hardware is being targeted use `‘PVRTTriStrip’`.
- If PowerVR SGX (Series 5) or Rogue (Series 6) hardware is being targeted use `‘PVRTGeometrySort’`.
- If the DirectX API is being targeted use either `‘PVRTGeometrySort’` or `‘D3DX’`.

Vertex Data Optimisations

‘Interleave Vertex Data’

Interleaving vertex data is an optimization that takes the vectors generated for the position, normals, UV co-ordinates etc. and places them into a single large array.

‘Align Vertex Data’

`‘Align Vertex Data’` pads the interleaved array so that each attribute for each vertex falls on a 32bit boundary; on some hardware this will provide a small performance improvement due to the overhead involved in accessing memory that does not align correctly with said boundary.

‘Sort Vertices’

`‘Sort Vertices’` is used for optimizing the vertex list to improve vertex-read memory cache. This option is only available in Windows on machines where DirectX9 is present.

5.2.2. Vertex Data Options

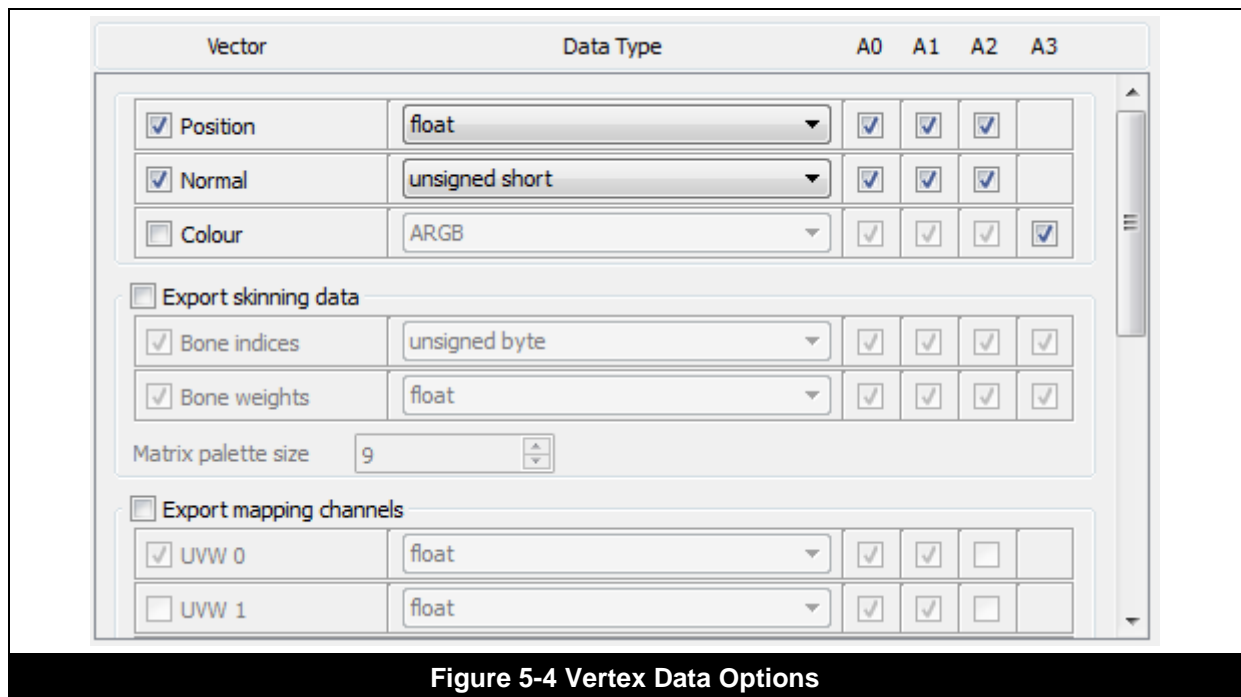


Figure 5-4 Vertex Data Options

Figure 5-4 displays the vertex data section of the PVRGeoPOD GUI; this section displays the vectors to be exported, their format, how many dimensions the given vector uses, and any other additional options related to a given set of vertex data; it also allows fine-grained control of these variables, such as which individual dimensions should be exported, and the specific data types to be used.

Data that is greyed out will not be exported, and any areas that are not greyed out are only exported if information exists to fill them.

Controlling the data types can be done from the 'Data Type' drop down menu. The following data types are available:

- ARGB
- RGBA
- D3DCOLOR
- DEC3N
- fixed 16.16
- float
- int
- short
- short, normalised
- UBYTE4
- unsigned byte
- unsigned byte, normalised
- unsigned int
- unsigned short
- unsigned short, normalised

Finally, if non-float data (e.g. ushort) is used for position 'Add Position Unpack Matrix' (available under Transformation Options) can be used as to improve the precision. This is done by using the entire range of the data type to store the position information and providing a matrix to unpack the data back to the correct values.

Basic Options

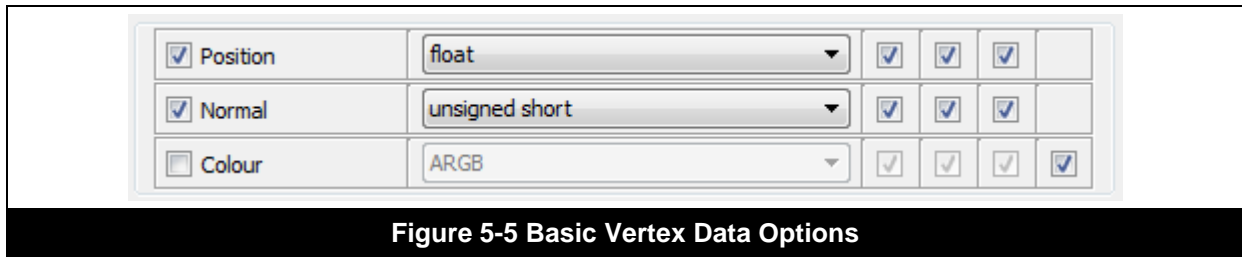


Figure 5-5 Basic Vertex Data Options

The basic vertex data options consist of the settings for vertex positions, normals and colours; their associated data types, and which dimensions are exported.

Skinning Data Options

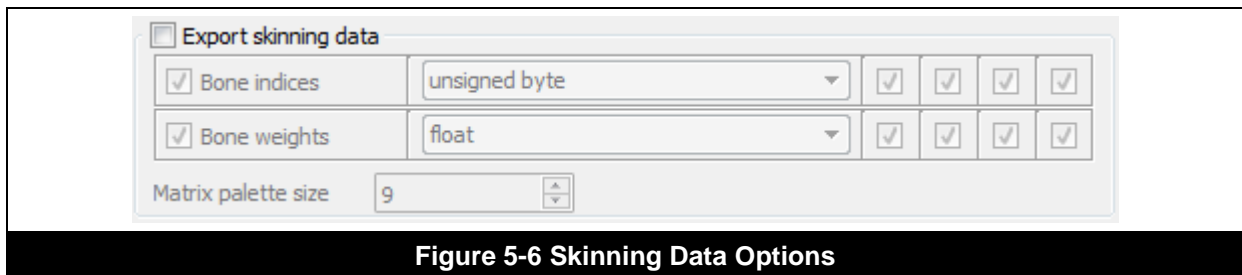


Figure 5-6 Skinning Data Options

The skinning data options consist of the settings for bone indices and bone weights, their associated data types, and dimensions.

'Export Skinning Data'

This option must be ticked if skinning is required.

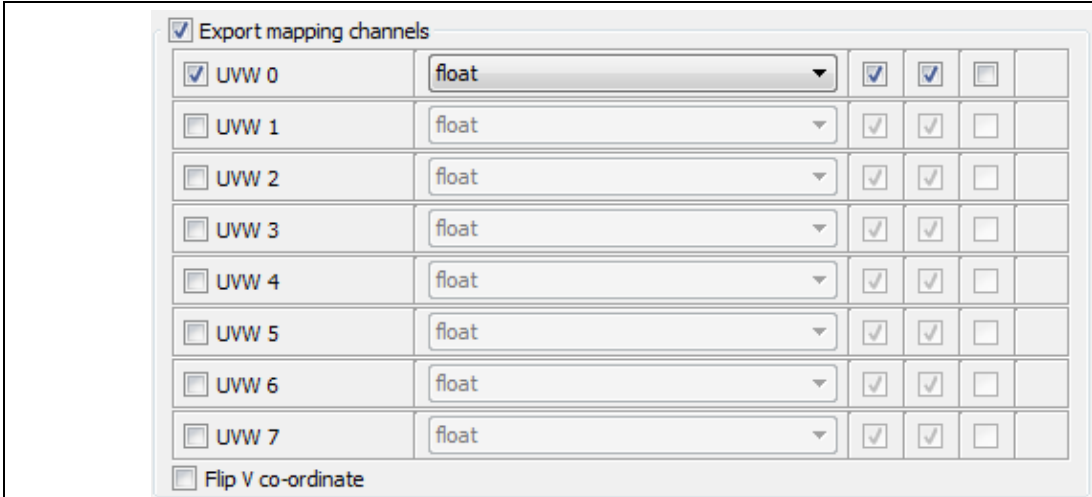
'Matrix Palette Size'

'Matrix Palette Size' represents the number of matrices that can affect a specific mesh. By default this value is nine in order to support APIs which limit the matrix palette size (e.g. OpenGL ES 1.1 for example).

If the number of matrices affecting a mesh is greater than the 'Matrix Palette Size' the mesh is split into batches which use fewer matrices per batch. The side effect is that more meshes produce more draw calls and thus a greater overhead.

Note: In some case when performing skinning or animation the default way of representing transformations in the POD format isn't sufficient for skinning. This can cause rendering issues. If this occurs with your scene ticking 'Export Transformations as Matrices' (available under Transformation Options) can rectify the situation.

Export Mapping Channels



Channel	Export	Data Type	Dim 1	Dim 2	Dim 3
<input checked="" type="checkbox"/> UVW 0	<input checked="" type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 1	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 2	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 3	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 4	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 5	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 6	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> UVW 7	<input type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

☐ Flip V co-ordinate

Figure 5-7 Mapping Channel Options

The mapping channel options consist of the settings for UV co-ordinate channels, their associated data types, and dimensions.

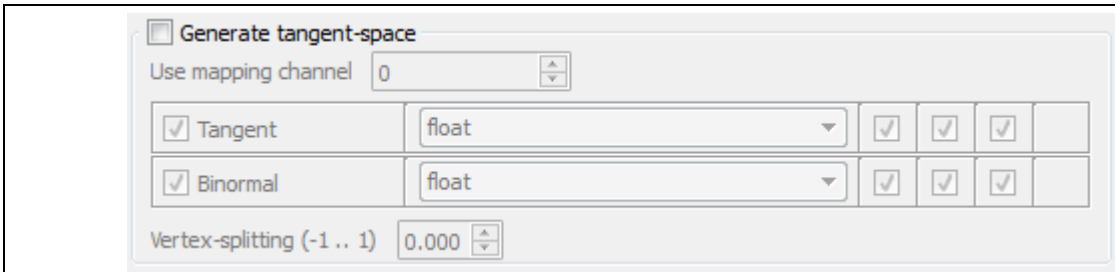
‘Export Mapping Channels’

This option must be ticked if UV co-ordinates are to be exported.

‘Flip V Coordinate’

This option flips the V Coordinate in the UV texture. This is used when you wish textures to behave in the same way as a ‘render to texture’ target in OpenGL. If both Direct3D and OpenGL are being targeted this option is unlikely to be desirable.

Generate Tangent Space



Channel	Export	Data Type	Dim 1	Dim 2	Dim 3
<input checked="" type="checkbox"/> Tangent	<input checked="" type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Binormal	<input checked="" type="checkbox"/>	float	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Vertex-splitting (-1 .. 1)

Figure 5-8 Tangent Space Options

The tangent space options consist of the settings for both of the bi-normal and tangent channels, their associated data types, and dimensions.

‘Generate Tangent Space’

This option must be ticked if tangents and bi-normals are to be exported.

‘Use Mapping Channel’

The value of this option determines which of the possible eight UV channels are used to create the tangent space data.

‘Vertex Splitting’

The ‘Vertex Splitting’ option is set between -1 and 1, representing a threshold value; if multiple faces share a vertex and the difference between the direction of those faces is greater than this threshold a new vertex is created for each face with separate tangent space data.

5.3. Material Options

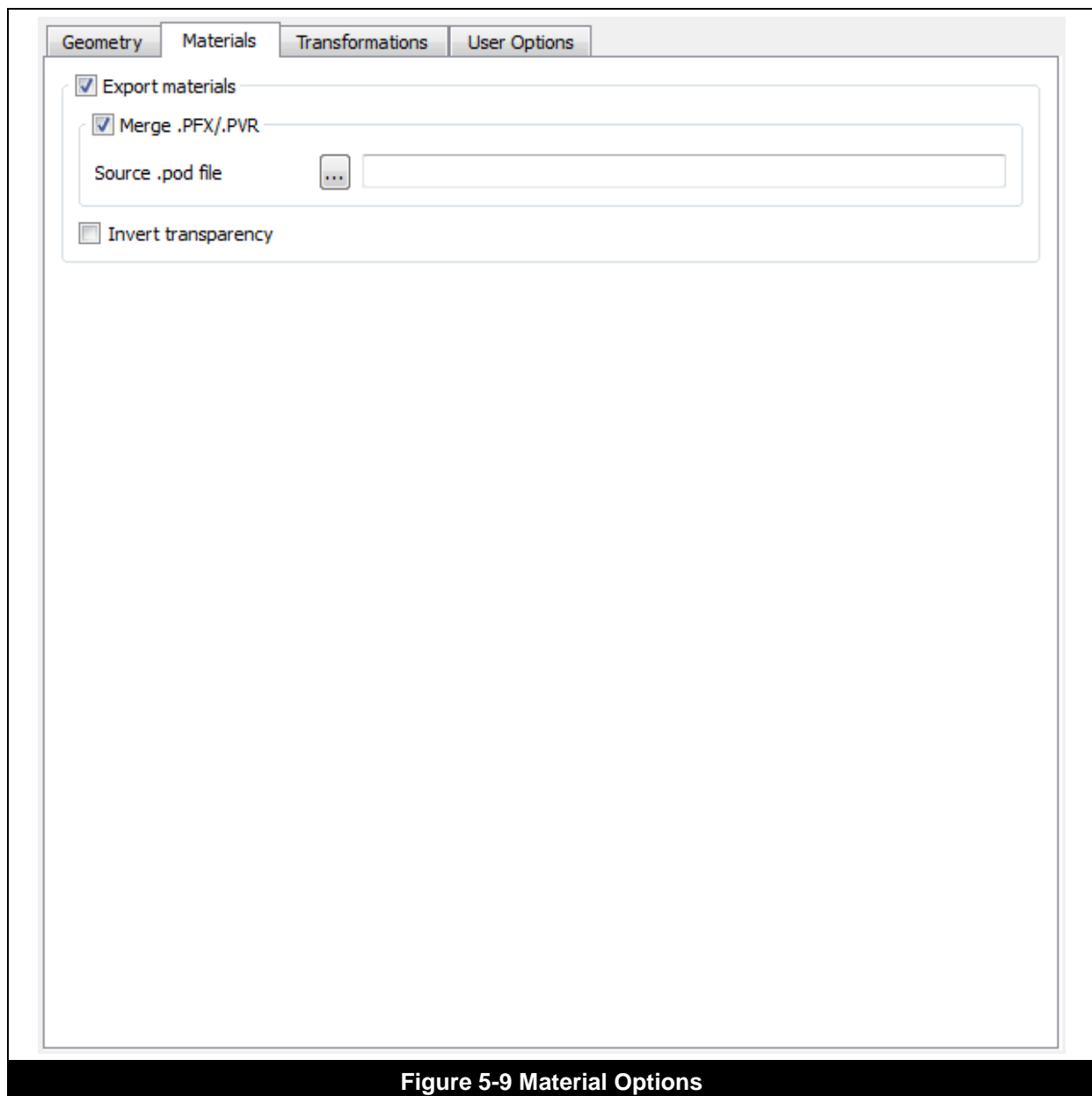


Figure 5-9 Material Options

5.3.1. General Options

'Export Materials'

This option must be ticked if materials are to be exported.

'Merge .PFX/.PVR'

In cases where a POD file has been edited and .pvr/.pfx files have been added it is possible to re-export the geometry from the original file back into the edited POD file. To do this tick 'Merge .PFX/.PVR', click the ellipsis (...) button, browse to the POD file to be updated and click 'Open'; once all export options are set and the export has been completed the POD file will contain the new geometry.

'Invert Transparency' – PVRGeoPOD Standalone Only

This option inverts the transparency value of the materials being exported (e.g. if the transparency is in the range of $0 < n < 1$ all values become $1 - n$ post export).

5.4. Transformation Options

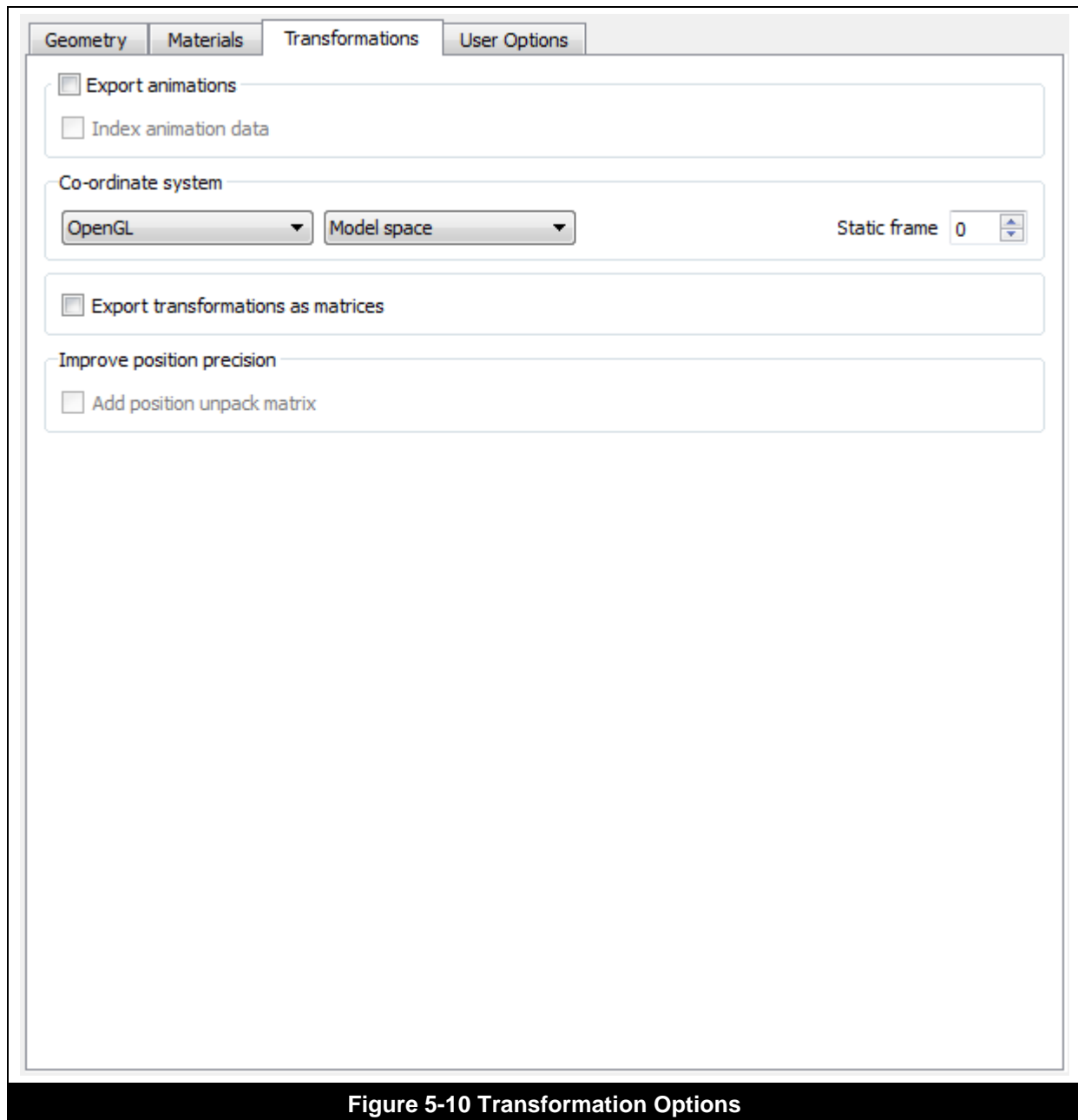


Figure 5-10 Transformation Options

5.4.1. General Options

‘Export Animations’

This option must be ticked if animations are to be exported.

‘Indexed Animation Data’

Instead of storing transformation data per keyframe ticking this option forces PVRGeoPOD to create a list of the transformations instead; transformations are then referred to by an index into that list within each keyframe. In instances with a small number of animations being performed repeatedly this can produce a smaller POD file.

Note: If issues occur when performing skinning or animation `‘Export Transformations as Matrices’` should be ticked as this solves many of the problems associated with these effects.

Co-ordinate System

The first drop-down menu contains two options:

- OpenGL
- DirectX

This must be set to match the API being targeted.

The second drop down menu also contains two options:

- World Space
- Model Space

These represent the ‘spaces’ the file will be exported into. This option only becomes available when no animation is present; when animation is present the export will always be in model space.

The final option is `‘Static Frame’`. If no animation is to be exported, and World Space has been selected a single frame can be selected for export. This allows for certain transformations to be ‘baked in’.

‘Export Transformations as Matrices’

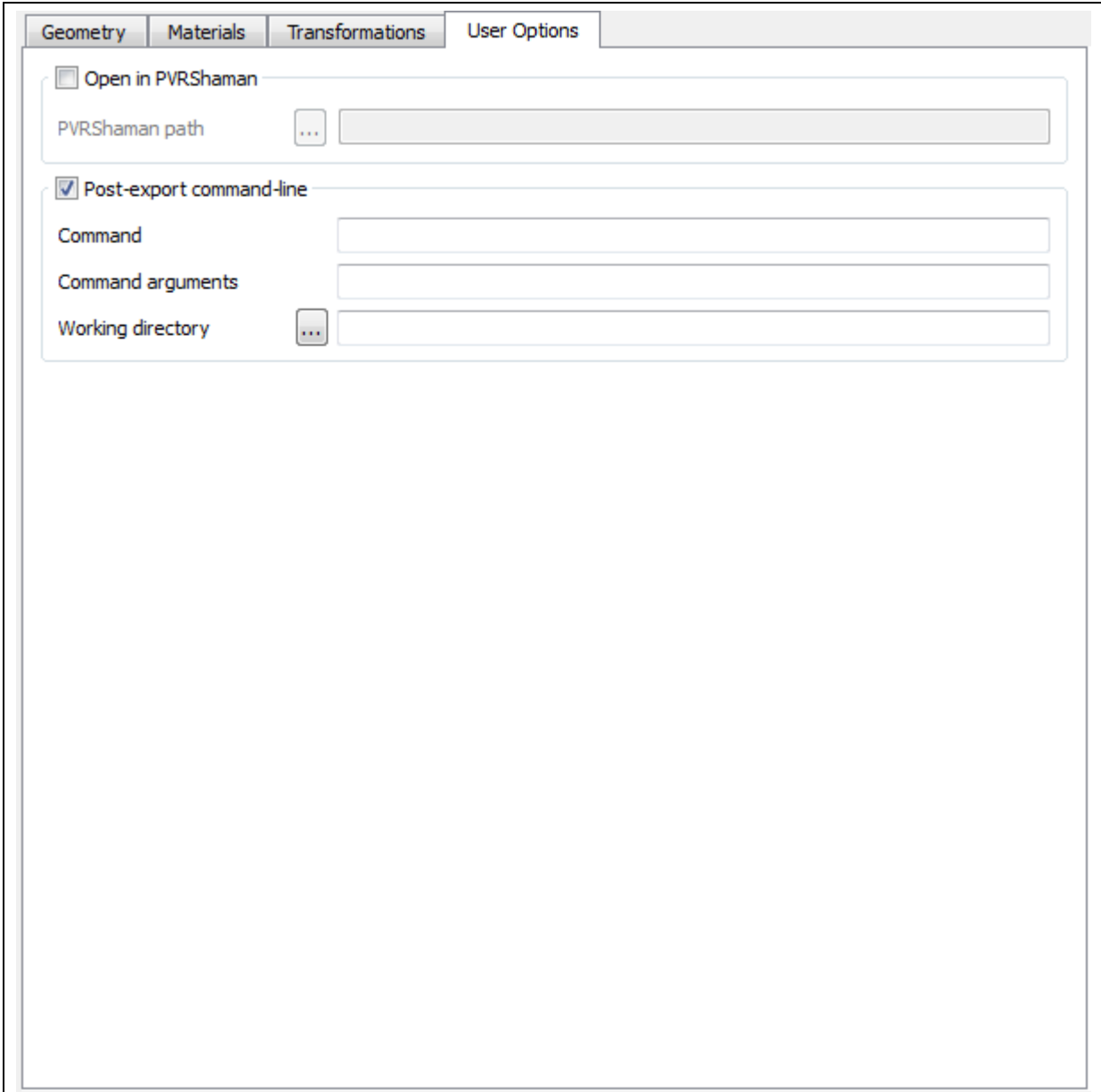
All transformations will be exported as matrices rather than as positions/quaternions/scale etc. if this option is selected.

Note: In some case when performing skinning or animation the default way of representing transformations in the POD format isn’t sufficient for skinning. This can cause rendering issues. If this occurs with your scene ticking `‘Export Transformations as Matrices’` (available under Transformation Options) can rectify the situation.

Improve Position Precision

If a non-float data type (e.g. ushort) is set for position (see Section 5.2.2 Vertex Data Options) `‘Add Position Unpack Matrix’` can be ticked to improve the precision of the position data. This is done by using the entire range of the data type to store the information; a matrix is provided to unpack the data back to the correct values.

5.5. User Options



The screenshot shows the 'User Options' dialog box with the following elements:

- Geometry** | **Materials** | **Transformations** | **User Options**
- ☐ Open in PVRShaman
 - PVRShaman path:
- ☒ Post-export command-line
 - Command:
 - Command arguments:
 - Working directory:

Figure 5-11 User Options

5.5.1. General Options

‘Open in PVRShaman’

If a path to PVRShaman is set in `‘PVRShaman Path’`, once export is complete, the resulting POD file will be opened in PVRShaman.

‘Post-export Command Line’

This series of options allows you to specify a separate program to be run upon successful completion of an export.

`‘Command’` represents the path to the executable, or the command line call for the command to run.

`‘Command Arguments’` represents any arguments that are to be passed to the program specified in `‘Command’`. If the phrase `‘%POD’` is found in the argument list it will be replaced with the absolute file path of the exported POD file.

`‘Working Directory’` represents the directory in which the program specified by `‘Command’` is to be run.

‘Export User Data’ (Plugins Only)

This option allows you to add user data to POD files by using a Max/MEL script. The user defined script should implement the functions:

- `DefineNodeUserData`
- `DefineMaterialUserData`
- `DefineSceneUserData`

These functions will be called by the exporter and their return values will be stored in the POD file. Examples can be seen in Section 6 MaxScript/MEL User Data.

5.6. Using Profiles

PVRGeoPOD includes support for profiles, groups of settings that can be saved and used again.

Once a series of export options has been set, right clicking on '[Current]' and clicking on 'Save As...' will pop up a box asking for a name for the profile. Once the profile is named and saved it will appear in the profile window on the left.

Double clicking an existing profile allows the name of the profile to be edited.

Right clicking on a profile and clicking 'Set as [Current]' will set the selected profile as active.

It is also possible to export and import profiles. Exporting a profile can be done by right clicking on a profile and clicking 'Export to file', this file can be saved as either a TXT file or a CFG file. Profiles can be imported by right clicking on the empty space and clicking 'Import from file'; profiles can be imported from either TXT or CFG files exported by PVRGeoPOD or from the settings stored in a POD file.

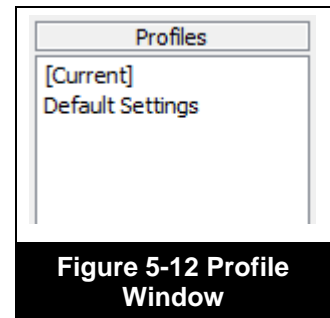


Figure 5-12 Profile Window

6. MaxScript/MEL User Data

6.1. Overview

During the export process, if 'Export User Data' is ticked, a script can be used to export any data the user desires. The location of this script must be entered in the 'Mel File' box, or the script must be found using the ellipsis (...) button, for this feature to function.

The user defined script should implement the following functions:

- DefineNodeUserData
- DefineMaterialUserData
- DefineSceneUserData

These functions will be called during the exporting process: DefineNodeUserData when exporting an object to a POD node; DefineMaterialUserData when exporting a Material; DefineSceneUserData when exporting a full scene. DefineNodeUserData will be passed a node id; DefineMaterialUserData will be passed a material id; DefineSceneUserData will be passed nothing; the return value of each script will be written into the pUserData field of the SPODNode, SPODMaterial, or SPODScene as appropriate. Once the export process is complete it is the user's responsibility to read the pUserData field and parse the exported data correctly. An example MAX and Mel script are provided below.

6.2. Mel Script

```
// Note: You can change the return type of the procedures if you wish

global proc string DefineNodeUserData(string $nodeid)
{
    // Return the node name and class type as a string to be included in the POD node for
    nodeid
    string $type = `nodeType $nodeid`;
    return $nodeid + " is of type " + $type;
}

global proc string DefineMaterialUserData(string $matid)
{
    /*
    We're going to return the material colour but we're going to do it
    as a comma separated string. The first value will be a unique tag to
    identify that we're exporting the colour value. The second value will
    be the material's red colour value, the third green and the final value
    blue.
    */
    string $s = "";
    $s += "3002,";
    float $c[] = `getAttr ($matid + ".color")`;
    $s += $c[0] + "," + $c[1] + "," + $c[2];
    return $s;
};

global proc string DefineSceneUserData()
{
    /*
    Anything returned from this function will be included in the SPODScene's user data.
    This is for information that doesn't belong with a node or a material.
    */
    return "Extra global data";
}
```

6.3. MAX Script

```
function DefineNodeUserData nodeid =
(
    /* Return the node name and class type as a string to
    be included in the POD node for nodeid */
    node = maxOps.getNodeByHandle nodeid
    str = stringstream ""
    format "% is of type " node.name to:str
    print (superClassOf node) to:str
    str as string
)

function DefineMaterialUserData matid subid =
(
    mat = sceneMaterials[matid]
    superClassOf mat

    /* Is our material a subobject material? */
    if isKindOf mat multimaterial then
    (
        /* If it is a subobject material return its name and parent material name */
        str = stringstream ""
        submat = mat.materialList[subid]
        format ":%%" mat.name submat.name to:str
        return str as string
    )

    /*
    Our material is just a material. We're going to return the diffuse colour
    but we're going to do it in a structured way using a MAXScript array. The
    first value will be a unique tag to identify that we're exporting the
    diffuse value. The second value will be the data size and the third vale
    is the material's diffuse value.
    */

    matdata = #()
    append matdata 3004                                /* The POD tag for diffuse */
    append matdata (4 * 3)                             /* The size of the diffuse data (3 floats) */
    append matdata mat.diffuse                         /* The diffuse data */

    /* return our data */
    matdata
)

function DefineSceneUserData =
(
    /*
    Anything returned from this function will be included
    in the SPODScene's user data.
    This is for information that doesn't belong with a node or a material.
    */
    "Extra global data"
)
```

7. Using POD Data

7.1. PVRShaman

POD files can be loaded into PVRShaman, the PowerVR Insider SDK shader composer. For more information on PVRShaman please see the PVRShaman User Manual.

7.2. PowerVR Insider SDK

The following are two examples of how to load POD file data. Example suggestions for utilising the SDK can be found in Section 8 Related Material.

7.2.1. Example: Loading a POD file

```
// Create the model object
CPVRTModelPOD m_model;

// Load the model & fill the object
if(m_model.ReadFromFile("modelPOD") != PVR_SUCCESS)
    return false;

// Do stuff

// Free the memory
m_model.Destroy();
```

7.2.2. Example: Loading a Header/Source file

```
// Include the scene data
#include "modelH"

// Create the model object
CPVRTModelPOD m_model;

// Load the model * fill the object
if(m_model.ReadFromMemory(model_pod, model_pod_size) != PVR_SUCCESS)
    return false;

// Do stuff

// Free the memory
m_model.Destroy();
```

8. Related Material

Software

- PVRShaman

Documentation

- PVRShaman User Manual
- POD File Format Specification
- PowerVR SDK User Guide

9. Contact Details

For further support contact:

devtech@imgtec.com

PowerVR Developer Technology
Imagination Technologies Limited
Home Park Estate
Kings Langley
Herts, WD4 8LZ
United Kingdom

Tel: +44 (0) 1923 260511

Fax: +44 (0) 1923 277463

Alternatively, you can use the PowerVR Insider forums:

www.imgtec.com/forum

For more information about PowerVR or Imagination Technologies Limited visit our web pages at:

www.imgtec.com

Appendix A. Block Names

The following is taken from the 'EPODFileName' enumerator in 'PVRTModelPOD.cpp':

- ePODFileCamAnimFOV
- ePODFileCamera
- ePODFileCamFar
- ePODFileCamFOV
- ePODFileCamIdxTgt
- ePODFileCamNear
- ePODFileColourAmbient
- ePODFileColourBackground
- ePODFileData
- ePODFileDataType
- ePODFileEndiannessMismatch
- ePODFileExpOpt
- ePODFileFlags
- ePODFileFPS
- ePODFileHistory
- ePODFileLight
- ePODFileLightColour
- ePODFileLightConstantAttenuation
- ePODFileLightFalloffAngle
- ePODFileLightFalloffExponent
- ePODFileLightIdxTgt
- ePODFileLightLinearAttenuation
- ePODFileLightQuadraticAttenuation
- ePODFileLightType
- ePODFileMatAmbient
- ePODFileMatBlendColour
- ePODFileMatBlendDstA
- ePODFileMatBlendDstRGB
- ePODFileMatBlendFactor
- ePODFileMatBlendOpA
- ePODFileMatBlendOpRGB
- ePODFileMatBlendSrcA
- ePODFileMatBlendSrcRGB
- ePODFileMatDiffuse
- ePODFileMatEffectFile
- ePODFileMatEffectName
- ePODFileMaterial
- ePODFileMatFlags
- ePODFileMatIdxTexAmbient
- ePODFileMatIdxTexBump
- ePODFileMatIdxTexDiffuse
- ePODFileMatIdxTexEmissive
- ePODFileMatIdxTexGlossiness
- ePODFileMatIdxTexOpacity
- ePODFileMatIdxTexReflection
- ePODFileMatIdxTexRefraction
- ePODFileMatIdxTexSpecularColour
- ePODFileMatIdxTexSpecularLevel
- ePODFileMatName
- ePODFileMatOpacity
- ePODFileMatShininess
- ePODFileMatSpecular
- ePODFileMatUserData
- ePODFileMesh
- ePODFileMeshBin

- ePODFileMeshBoneBatchBoneCnts
- ePODFileMeshBoneBatchBoneMax
- ePODFileMeshBoneBatchCnt
- ePODFileMeshBoneBatches
- ePODFileMeshBoneBatchOffsets
- ePODFileMeshBoneIdx
- ePODFileMeshBoneWeight
- ePODFileMeshFaces
- ePODFileMeshInterleaved
- ePODFileMeshNor
- ePODFileMeshNumFaces
- ePODFileMeshNumStrips
- ePODFileMeshNumUVW
- ePODFileMeshNumVtx
- ePODFileMeshStripLength
- ePODFileMeshTan
- ePODFileMeshUnpackMatrix
- ePODFileMeshUVW
- ePODFileMeshVtx
- ePODFileMeshVtxCol
- ePODFileN
- ePODFileNode
- ePODFileNodeAnimFlags
- ePODFileNodeAnimMatrix
- ePODFileNodeAnimMatrixIdx
- ePODFileNodeAnimPos
- ePODFileNodeAnimPosIdx
- ePODFileNodeAnimRot
- ePODFileNodeAnimRotIdx
- ePODFileNodeAnimScale
- ePODFileNodeAnimScaleIdx
- ePODFileNodeIdx
- ePODFileNodeIdxMat
- ePODFileNodeIdxParent
- ePODFileNodeMatrix
- ePODFileNodeName
- ePODFileNodePos
- ePODFileNodeRot
- ePODFileNodeScale
- ePODFileNodeUserData
- ePODFileNumCamera
- ePODFileNumFrame
- ePODFileNumLight
- ePODFileNumMaterial
- ePODFileNumMesh
- ePODFileNumMeshNode
- ePODFileNumNode
- ePODFileNumTexture
- ePODFileScene
- ePODFileStride
- ePODFileTexName
- ePODFileTexture
- ePODFileUserData
- ePODFileVersion

Appendix B. PVRGeoPOD Command Line Options

-LoadOptions

e.g. -LoadOptions=c:\options.cfg

-SaveOptions

e.g. -SaveOptions=c:\options.cfg

-FixedPoint

e.g. -FixedPoint=1 to enable

-FlipTextureV

e.g. -FlipTextureV=1 to enable

-Indexed

e.g. -Indexed=1 to enable

-Interleaved

e.g. -Interleaved=1 to enable

-SortVertices

e.g. -SortVertices=1 to enable

-TangentSpace

e.g. -TangentSpace=1 to enable

-ExportControllers

e.g. -ExportControllers=1 to enable

-IndexAnimation

e.g. -IndexAnimation=1 to enable

-ExportGeometry

e.g. -ExportGeometry=1 to enable

-ExportMatrices

e.g. -ExportMatrices=1 to enable

-ExportMappingChannels

e.g. -ExportMappingChannels=1 to enable

-ExportMaterials

e.g. -ExportMaterials=1 to enable

-ExportNormals

e.g. -ExportNormals=1 to enable

-ExportSkin

e.g. -ExportSkin=1 to enable

-ExportVertexColor, -ExportVertexColour

e.g. -ExportVertexColor=1 to enable

-InvertTransparency

e.g. -InvertTransparency=1 to enable

-ExportModelSpace

e.g. -ExportModelSpace=1 to enable

-TangentSpaceVtxSplit

e.g. -TangentSpaceVtxSplit=0.000

-BoneLimit

e.g. -BoneLimit=0

-cs

By default PVRGeoPOD converts the coordinate system of the scene to OpenGL.

The -cs allows for the overriding of this default.

e.g. -cs=ogl

Valid values:

ogl, d3d

-PrimitiveType

e.g. -PrimitiveType=TriList

Valid values:

TriList, TriStrips

-TriSort

e.g. -TriSort=None

Valid values:

None, PVRTGeometrySort, D3DX, PVRTTriStrip

-PosType

The data type for vertex positions.

e.g. -PosType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-NorType

The data type for vertex normals.

e.g. -NorType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-ColorType, -ColourType

The data type for vertex colours.

e.g. -ColorType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-UVWnType

The data type for vertex UVW co-ordinates (set n).

e.g. -UVW0Type=float -UVW5=ARGB

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-BinormalType

The data type for binormals.

e.g. -BinormalType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-TangentType

The data type for tangents.

e.g. -TangentType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-BoneIdxType

The data type for bone indices.

e.g. -BoneIdxType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-BoneWtType

The data type for bone weights.

e.g. -BoneWtType=float

Valid values:

float, int, unsigned int, ushort, ushortN, RGBA, ARGB, D3DCOLOR, UBYTE4, DEC3N, fixed, ubyte, ubyteN, short, shortN, byte, byteN

-PadDataTo

Pad the vertex data to PadDataTo no. of bytes. The GUI option 'Align Vertex Data' is functionally equivalent to '-PadDataTo=4'

e.g. -PadDataTo=1

Valid values:

1, 2, 4

-Merge

Merge the output POD file with an existing POD file, functionally equivalent to the GUI option 'Merge .PFX/.PVR'.

e.g. -Merge=c:/otherPOD

Imagination Technologies, the Imagination Technologies logo, AMA, Codescape, Enigma, IMGworks, I2P, PowerVR, PURE, PURE Digital, MeOS, Meta, MBX, MTX, PDP, SGX, UCC, USSE, VXD and VXE are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.