PVRGeoPOD & Collada2POD

User Manual

Copyright © Imagination Technologies Ltd. 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 & Collada2POD.User Manual.1.15f.External.doc
Version	:	1.15f External Issue (Package: POWERVR SDK 2.09.29.0585)
Issue Date	:	14 Sep 2011
Author	:	Imagination Technologies Ltd

Contents

1. Introduction 4

1.1. Supported Features 4
1.2. Limitations 4

2. Compatibility 5

2.1. 3D Studio Max 5
2.2. Blender 5
2.3. Maya 6

3. Installation 7

3.1. Collada2POD 7
3.1.1. From Installer 7
3.1.2. From GZIP 7
3.2. PVRGeoPOD 7
3.2.1. From Installer 7
3.2.2. From GZIP 7
3.2.3. 3D Studio Max 8
3.2.4. Maya 8
3.2.5. Blender 8

4. Exporting/Converting 9

4.1. Available File Formats 9
4.1.1. POD File 9
4.1.2. Header/Source File 9
4.1.3. POD vs. Header/Source 9
4.2. Exporting a Scene from 3DS Max 10
4.3. Exporting a Scene from Maya 11
4.4. Exporting a Scene from Blender 11
4.5. Converting from a Collada File 12
4.5.1. Limitations 13
4.5.2. Command Line 13

5. PVRGeoPOD/Collada2POD Options 14

5.1. Overview 14
5.2. Options Breakdown 15
5.2.1. Export Options – General 15
5.2.2. Export Options – Materials 15
5.2.3. Export Options – Animation 15
5.2.4. Export Options – Coordinate System 16
5.2.5. Geometry Options – General 17
5.2.6. Geometry Options – Skinning Options 18
5.2.7. Geometry Options – Primitive Type 18
5.2.8. Geometry Options – Triangle Sort 18
5.2.9. Vertex Vector Formats 19
5.2.10. Extra Options 20

6. Using POD Data 22

6.1. PVRShaman 22
6.2. PODPlayer 22
6.3. PowerVR Insider SDK 22
6.3.1. Example: Loading a POD file 22
6.3.2. Example: Loading a Header/Source file 22
6.4. POD File Format 23
6.4.1. Binary File Format 23
6.4.2. Basic File Reading Algorithm 23

7. Related Material 24

8. Contact Details 25

Appendix A. Block Names 26

Appendix B. Collada2POD Command Line Options 28

Appendix C. Example PVRGeoPOD User Data Scripts 31

C.1. MAX Script 31
C.2. Mel Script 32

List of Figures

Figure 4-1 3DSMax Export Menu 10
Figure 4-2 Collada2POD Main Window 12
Figure 5-1 PVRGeoPOD Main Window 14
Figure 5-2 Option Tooltip 14
Figure 5-3 Export Options 15
Figure 5-4 Geometry Options 17
Figure 5-5 Further Geometry Options 18
Figure 5-6 Vertex Vector Formats 19
Figure 5-7 Extra Options 20

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. Collada2POD is a standalone application designed to convert from the Collada file format (.dae) to the .pod file format. It comes in both a graphical and command line version and is provided for where a developer's 3D modeller does not have a compatible PVRGeoPOD 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

1.2. Limitations

The following are not supported:

- UV animation
- Vertex animation
- Splines
- NURBs
- Multiple materials per mesh
- Per object culling information

2. Compatibility

2.1. 3D Studio Max

Plug-in	3DS MAX release	"igame.dll" version
3dsmax6\Windows_x86_32\PVRGeoPOD.dle	3DS MAX 6	6.0.0.56
	3DS MAX 6 + SP1	6.0.1.62
	3DS MAX 7	7.0.0.65
	3DS MAX 7 + SP1	7.0.1.76
	3DS MAX 7 + 3DXI 2.0	7.0.1.78
	3DS MAX 8 + 3DXI 2.0	8.0.0.40
	3DS MAX 8	8.0.0.92
	3DS MAX 8 + SP1	8.0.1.11
	3DS MAX 8 + SP2	8.0.1.18
	3DS MAX 8 + SP3	8.0.1.24
3dsmax9\Windows_x86_32\PVRGeoPOD.dle	3DS MAX 9	9.0.0.100
	3DS MAX 2008	10.0.0.86
	3DS MAX 2009	11.0.0.57
3dsmax12\Windows_x86_32\PVRGeoPOD.dle	3DS MAX 2010	12.0.0.106
	3DS MAX 2011	13.0.0.94
	3DS MAX 2012	14.0.0.121
3dsmax12\Windows_x86_64\PVRGeoPOD.dle	3DS MAX 2010 (x64)	12.0.0.106
	3DS MAX 2011 (x64)	13.0.0.94
	3DS MAX 2012 (x64)	14.0.0.121

2.2. Blender

Supported versions:

- 2.48
- 2.48a
- 2.49b

Requirement:

- Python 2.5+

2.3. Maya

Maya version	Windows		Linux		MacOS	
	32-bit	64-bit	32-bit	64-bit	32-bit	64-bit
7						
8						
8.5						
2008						
2009						
2010						
2011						
2012						

3. Installation

3.1. Collada2POD

Collada2POD is a pair of standalone executables that are available as part of the PowerVR Insider SDK or as a separate individual package, both of which can be downloaded from the PowerVR Insider website.

3.1.1. From Installer

Download either the PowerVR Insider SDK or the individual Collada2POD package and follow the on screen instructions. Once the package has successfully installed, the application will be available in the Windows `'Start Menu'`.

3.1.2. From GZIP

Download either the PowerVR Insider SDK or the individual Collada2POD package. Unzip the .tar.gz file, and then untar the .tar file. From the ensuing folder, browse to:

```
<SDK_ROOT>\Utilities\Collada2POD\<PLATFORM>\
```

This folder will contain both of the Collada2POD executables.

3.2. PVRGeoPOD

PVRGeoPOD is a series of plugins for various 3D modelling applications that are available as part of the PowerVR Insider SDK or as a separate individual package, both of which can be downloaded from the PowerVR Insider website.

3.2.1. From Installer

Download either the PowerVR Insider SDK or the individual PVRGeoPOD package and follow the on screen instructions. Once the package has successfully installed, the libraries will be available in:

```
<SDK_ROOT>\Utilities\PVRGeoPOD\<PLATFORM>\
```

3.2.2. From GZIP

Download either the PowerVR Insider SDK or the individual PVRGeoPOD package. Unzip the .tar.gz file, and then untar the .tar file. From the ensuing folder, browse to:

```
<SDK_ROOT>\Utilities\PVRGeoPOD\<PLATFORM>\
```

In both cases this folder will contain the PVRGeoPOD plugin files. Once these files have been located they must be installed into the plugin system for the respective application. Instructions for each of the supported platforms is include below.

3.2.3. 3D Studio Max

Copy: ‘PVRGeoPOD.dle’

To: <3DSMAX_DIR>\plugins\

3.2.4. Maya

Windows

Copy: <MAYA_VERSION>\Windows_x86_*\PVRGeoPOD_v<MAYA_VERSION>.mll

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

Linux

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

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

MacOS

Copy <MAYA_VERSION>/MacOS_x86*/PVRGeoPOD_v<MAYA_VERSION>.bundle

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

3.2.5. Blender

Linux

Copy: ‘libPVRGeoPOD.so.<VERSION>’ and ‘blenderPodExporter.py’

To: ~/.blender/scripts

Windows XP

Copy: ‘libPVRGeoPOD-<VERSION>.dll’ and ‘blenderPodExporter.py’

To: C:\Documents and Settings\<USERNAME>\Application Data\Blender Foundation\Blender\.blender\scripts\

Or: C:\Program Files\Blender Foundation\Blender\.blender\scripts\

Windows Vista/Seven

Copy: ‘libPVRGeoPOD-<VERSION>.dll’ and ‘blenderPodExporter.py’

To: C:\Users\<USERNAME>\AppData\Roaming\Blender Foundation\Blender\.blender\scripts\

MacOS (v10.5+)

Copy: ‘libPVRGeoPOD.dylib’ and ‘blenderPodExporter.py’

To: Blender scripts directory e.g. “…/blender.app/Contents/MacOS/.blender/scripts”

4. Exporting/Converting

4.1. Available File Formats

4.1.1. POD File

PVRGeoPOD exports data to POD files (.pod). The PowerVR SDK Tools library contains functions and classes to use .pod files in applications. See Section 6 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 6 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'`.

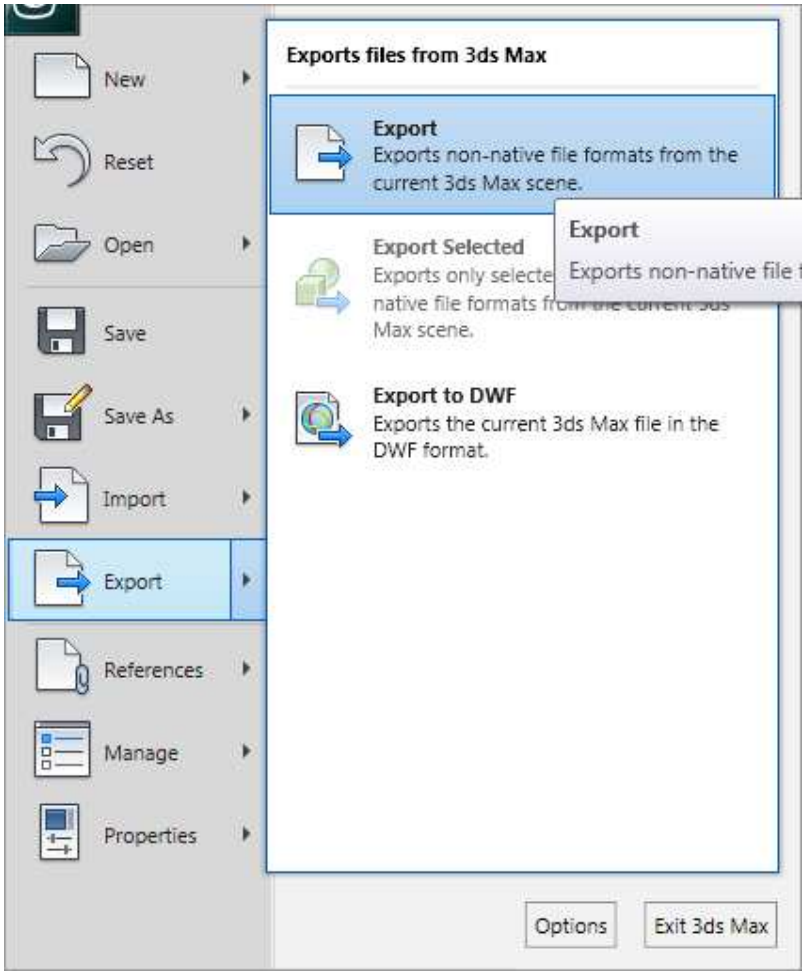

Figure 4-1 3DSMax Export Menu

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 open the .blend file you wish to convert in Blender, click on `'File -> Export -> POD'`.

4.5. Converting from a Collada File

In order to convert a .dae file to a .pod file use the ellipsis (…) button at the bottom of the Collada2POD window to select an input file, and an output destination; then click on `'Convert'`.

For a full breakdown of the available options see Section 5 PVRGeoPOD/Collada2POD Options.

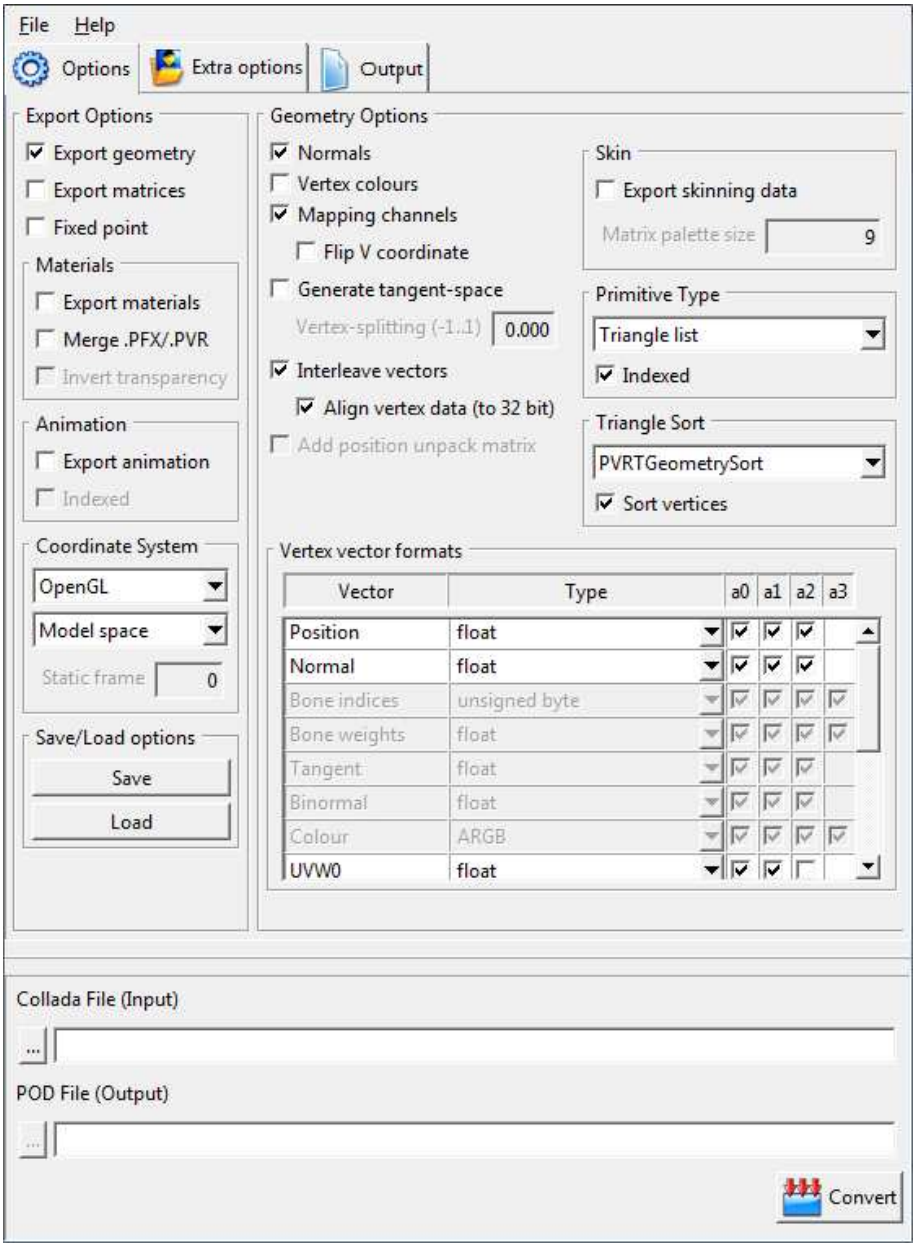

Figure 4-2 Collada2POD Main Window

4.5.1. Limitations

- Only meshes constructed from triangles 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.

4.5.2. Command Line

The Collada2POD command line utility can be called directly as follows:

```
Collada2POD –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 Collada2POD Command Line Options or from the command line as follows:

```
Collada2POD /?
```

5. PVRGeoPOD/Collada2POD 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.

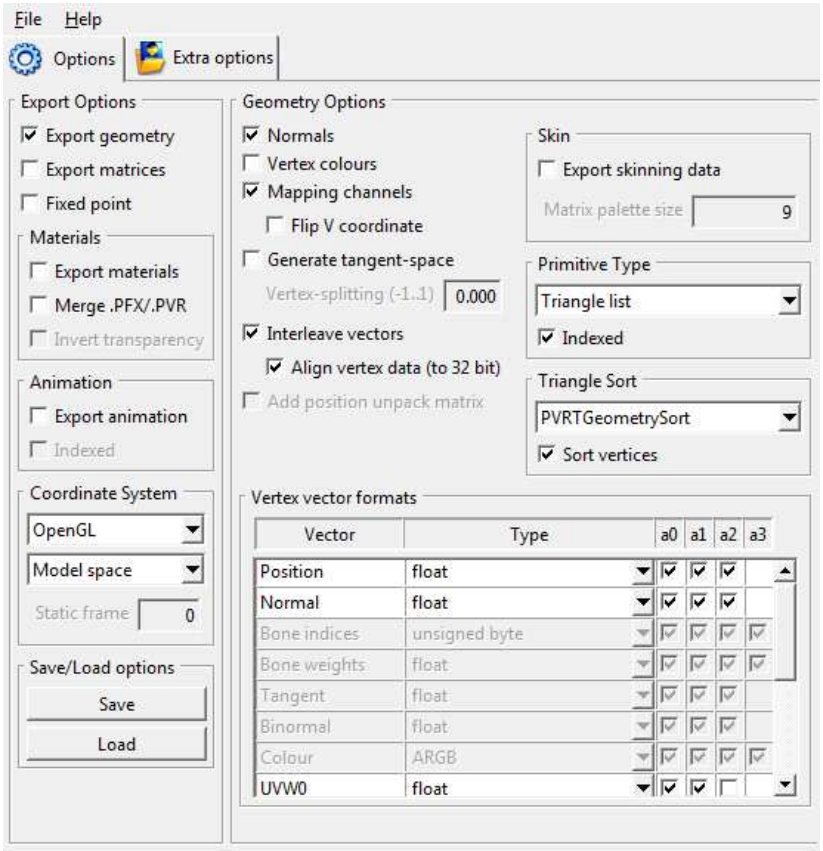

Figure 5-1 PVRGeoPOD Main Window

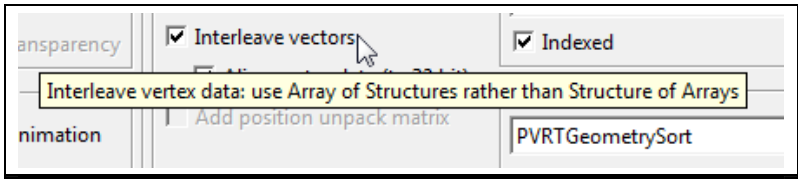

Figure 5-2 Option Tooltip

5.2. Options Breakdown

5.2.1. Export Options – General

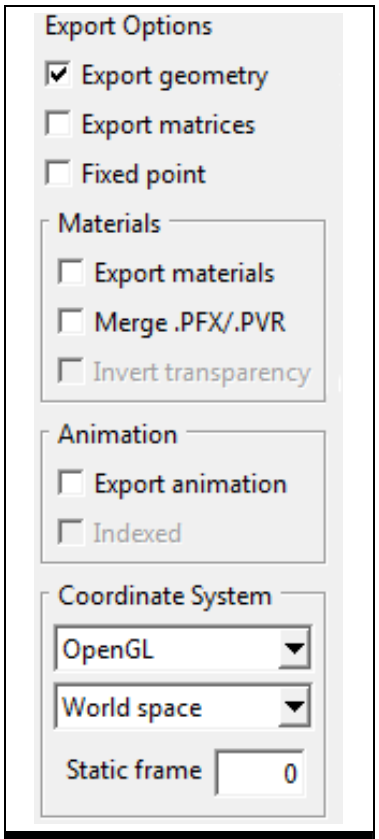

Figure 5-3 Export Options

'Export Geometry'

This option ensures that vertex position information is exported.

'Export Matrices'

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

'Fixed Point'

If this option is ticked then all transformations and floating point based material properties will be exported using fixed point numbers in the format 16.16.

5.2.2. Export Options – Materials

'Export Materials'

If material information has been set in the editor from which you are exporting and you wish that material information to be include in the .pod file, tick this option.

'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'` and when asked where to save the .pod file overwrite the file you wish to re-export to. It is important to note that material names must match between versions.

'Invert Transparency' – Collada2POD 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).

5.2.3. Export Options – Animation

'Export Animation'

PVRGeoPOD will export all animation data to the output .pod file.

'Indexed'

Instead of storing transformation data per keyframe, `'Indexed'` forces PVRGeoPOD to create a list of all of the transformations instead and refer to an index into that list in each keyframe. In instances with a small number of animations being performed repeatedly this can produce a smaller .pod file.

5.2.4. Export Options – Coordinate System

The first drop-down menu contains two options:

- OpenGL
- DirectX

This must be set to match the API being targeted to ensure that vertex information is stored in the correct coordinate system for the target API.

The second drop down menu contains two options also:

- World Space
- Model Space

These represent the 'spaces' the file will be exported into. This option only becomes available when no animation is present as 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'.

5.2.5. Geometry Options – General

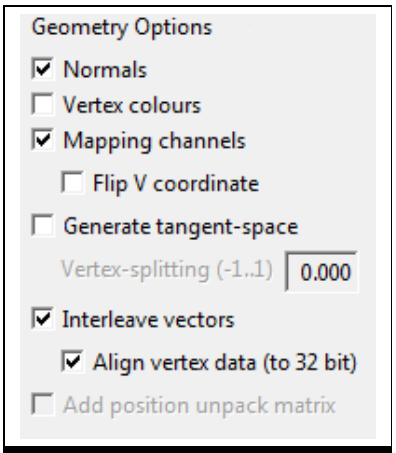

Figure 5-4 Geometry Options

'Normals'

With this option ticked the normal vectors for each vertex will be exported. This data is formatted as a vector of data types as set in Vertex Vector Formats.

'Vertex Colours'

This option outputs vertex colouration. This might be used if, for instance, there are to be no textures applied to a model. This data is formatted as a vector of data types as set in Vertex Vector Formats, though by default it is a four dimensional vector in form ARGB.

'Mapping Channels'

`'Mapping Channels'` exports the UV co-ordinates for each vertex. This is required in most cases, specifically when any form of texturing, normal mapping, height mapping etc. is to be used.

'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'

This option generates tangent and bi-normal information for each vertex being exported. As with all other information of this sort the data types used can be set in Vertex Vector Formats. It should be noted that this information is calculated as part of the export, and that any pre-existing tangent space information will not be used.

'Vertex Splitting'

The `'Vertex Splitting'` option is set to a value between -1 and 1; it represents a threshold; 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.

'Interleave Vectors'

`'Interleave Vectors'` 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 the information 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.

'Add Position Unpack Matrix'

If non-float data (e.g. ushort) has been set in Vertex Vector Formats for position `'Add Position Unpack Matrix'` can be used as a workaround 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.

5.2.6. Geometry Options – Skinning Options

'Export Skinning Data'

This option must be ticked if skinning is used; it forces PVRGeoPOD to export the skinning data into the .pod file.

'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).

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.

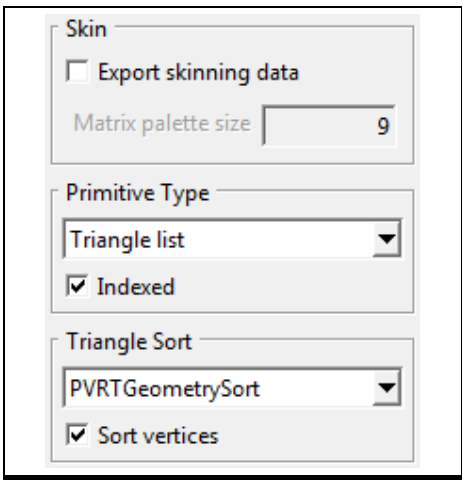

Figure 5-5 Further Geometry Options

5.2.7. Geometry Options – 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.

5.2.8. Geometry Options – Triangle Sort

'Triangle Sort' is a performance optimization; 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 hardware is being targeted use `'PVRTGeometrySort'`.
- If the DirectX API is being targeted use either `'PVRTGeometrySort'` or `'D3DX'`.

`'Sort Vertices'` is not used to enable the triangle sorting, but is a separate option for optimizing the vertex list to improve vertex-read cache memory. This option is only available in Windows.

5.2.9. Vertex Vector Formats

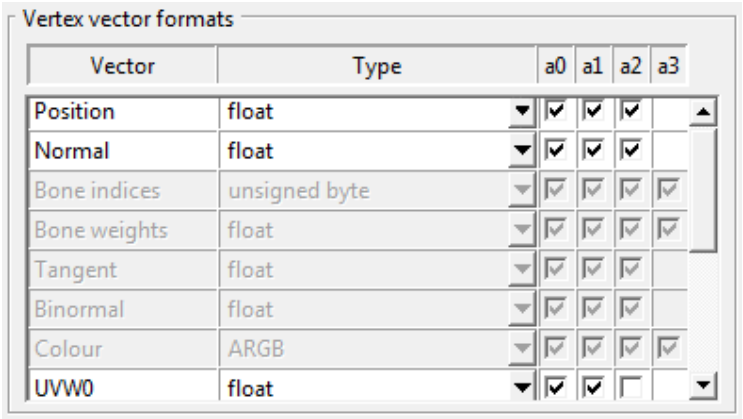

Figure 5-6 Vertex Vector Formats

Figure 5-6 displays the Vertex Vector Formats section of the PVRGeoPOD GUI; this section displays the vectors to be exported, their format, and how many dimensions the given vector uses; it also allows fine-grained control of these variables.

Data that will not be exported is greyed out, and any areas that are not greyed out are only exported if information exists to fill them. For example, even though eight UV channels are available only the channels that are used will be exported.

Controlling the data types can be done from the 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

As mentioned in Section 5.2.5 Geometry Options – General; if non-float data (e.g. ushort) has been set in Vertex Vector Formats for position `'Add Position Unpack Matrix'` can be used as a workaround 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.

Finally the individual dimensions in each vector can be activated or deactivated by adding or removing the tick from the associated box a0 through a3.

5.2.10. Extra Options

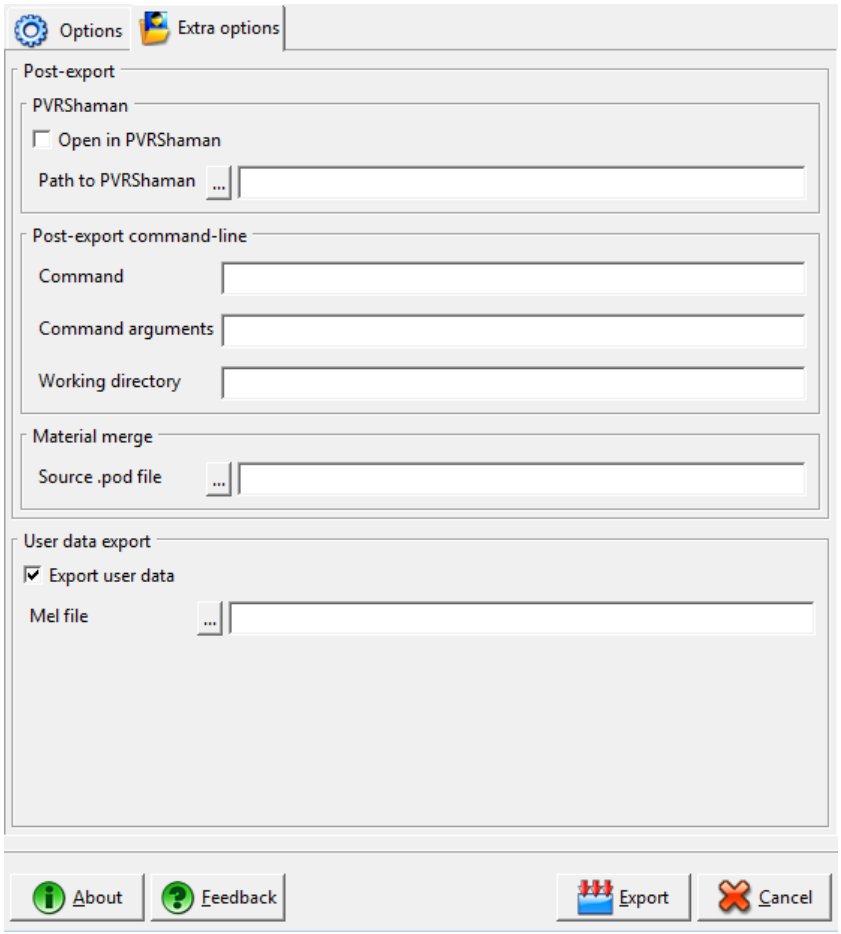

Figure 5-7 Extra Options

'Open in PVRShaman'

With this option ticked, if a path to PVRShaman is set in `'Path to PVRShaman'` then the .pod file resulting from the export will be opened in PVRShaman upon completion of export.

'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 exported .pod file is to be passed as an argument the name of the .pod file must be in this section.

`'Working Directory'` represents the directory in which the program specified by `'Command'` is to be run. If an absolute path is not set in `'Command Arguments'` for the .pod file to be opened, then the `'Working Directory'` must be set so that the relative path to the .pod file is accurate.

'Material Merge'

Using `'Material Merge'` will merge the output of PVRGeoPOD's export with the selected .pod file, retaining any materials/.pfx settings from the selected .pod file. Unlike the `'Merge .PFX/.PVR'` option under Export Options this does not require an overwrite of the file to be merged; though it does require `'Merge .PFX/.PVR'` to be ticked.

'Export User Data'

This option allows you to add user data to .pod files by using a MAX or 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 Appendix C Example PVRGeoPOD User Data Scripts.

6. Using POD Data

6.1. PVRShaman

.pod files can be loaded into PVRShaman, the PowerVR Insider SDK shader composer; a fully integrated IDE for creating, testing and optimizing shaders. For more information on PVRShaman please see the PVRShaman User Manual.

6.2. PODPlayer

PODPlayer is an application for displaying .pod files and their animations which includes an inbuilt menu, and basic options for playback control; it is currently only packaged for the Open GL ES 2.0 versions of the SDK. For more information on PODPlayer please see the PODPlayer User Manual.

6.3. PowerVR Insider SDK

The following are two examples of how to load .pod file data. Training course and demo suggestions for utilising the SDK can be found in Section 7 Related Material.

6.3.1. Example: Loading a POD file

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

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

// Do stuff

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

6.3.2. Example: Loading a Header/Source file

```
// Include the scene data
#include "model.h"

// Create the model object
CPVRTModelPOD m_model;

// Load the model * fill the object
if(!m_model.ReadFromMemory(_model_pod, _model_pod_size))
        return false;

// Do stuff

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

6.4. POD File Format

The .pod file format is provided for information purposes only; it is highly recommended that the tools provided in the PowerVR Insider SDK be used for manipulating .pod files.

6.4.1. Binary File Format

.pod files are stored in a tagged, nested structure of blocks. Each block is 'book-ended' by a pair of markers; each marker consists of two 32 bit values in the following format:

DWORD	Bit	Symbol	Description
0	31	End	Start/End tag. 0 This marker is the beginning of a block 1 This marker is the end of a block
	0 – 30	Name	Block name.
1	0 – 31	Length	Number of bytes in the block.

The full list of valid block names can be found in Appendix C Block Names.

6.4.2. Basic File Reading Algorithm

```
Read initial marker.
While 'Name' is a recognised value.
        Read 'Length' data.
        Read next start marker.
```

7. Related Material

Software

- PVRShaman
- PODPlayer

Documentation

- PVRGeoPOD Getting Started
- PVRShaman Getting Started
- PVRShaman User Manual
- PODPlayer User Manual
- PowerVR SDK User Guide

8. Contact Details

For further support contact:

devtech@imgtec.com

PowerVR Developer Technology
Imagination Technologies Ltd.
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 Ltd. 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. Collada2POD Command Line Options

-LoadOptions

Equivalent in GUI: Export Options – General – Load
e.g.-LoadOptions=c:\options.cfg

-SaveOptions

Equivalent in GUI: Export Options – General – Save
e.g.-SaveOptions=c:\options.cfg

-FixedPoint

Equivalent in GUI: Export Options – General – 'Fixed Point'
e.g. -FixedPoint=1 to enable

-FlipTextureV

Equivalent in GUI: Geometry Options – General – 'Flip V Coordinate'
e.g. -FlipTextureV=1 to enable

-Indexed

Equivalent in GUI: Geometry Options – Primitive Type – 'Indexed'
e.g. -Indexed=1 to enable

-Interleaved

Equivalent in GUI: Geometry Options – General – 'Interleave Vectors'
e.g. -Interleaved=1 to enable

-SortVertices

Equivalent in GUI: Geometry Options – Triangle Sort – 'Sort Vetices'
e.g. -SortVertices=1 to enable

-TangentSpace

Equivalent in GUI: Geometry Options – General – 'Generate Tangent Space'
e.g. -TangentSpace=1 to enable

-ExportControllers

Equivalent in GUI: Export Options – Animation – 'Export Animation'
e.g. -ExportControllers=1 to enable

-IndexAnimation

Equivalent in GUI: Export Options – Animation – 'Indexed'
e.g. -IndexAnimation=1 to enable

-ExportGeometry

Equivalent in GUI: Export Options – General – 'Export Geometry'
e.g. -ExportGeometry=1 to enable

-ExportMatrices

Equivalent in GUI: Export Options – General – 'Export Matrices'
e.g. -ExportMatrices=1 to enable

-ExportMappingChannels

Equivalent in GUI: Geometry Options – General – 'Mapping Channels'
e.g. -ExportMappingChannels=1 to enable

-ExportMaterials

Equivalent in GUI: Export Options – Materials – 'Export Materials'
e.g. -ExportMaterials=1 to enable

-ExportNormals

Equivalent in GUI: Geometry Options – General - 'Normals'
e.g. -ExportNormals=1 to enable

-ExportSkin

Equivalent in GUI: Geometry Options – Skinning Options – 'Export Skinning Data'

e.g. -ExportSkin=1 to enable

-ExportVertexColor, -ExportVertexColour

Equivalent in GUI: Geometry Options – General – 'Vertex Colours'

e.g. -ExportVertexColor=1 to enable

-InvertTransparency

Equivalent in GUI: Export Options – Materials – 'Invert Transparency'

e.g. -InvertTransparency=1 to enable

-ExportModelSpace

Equivalent in GUI: Export Options – Coordinate System

e.g. -ExportModelSpace=1 to enable

-TangentSpaceVtxSplit

Equivalent in GUI: Geometry Options – General – 'Vertex Splitting'

e.g. -TangentSpaceVtxSplit=0.000

-BoneLimit

Equivalent in GUI: Geometry Options – Skinning Options – 'Matrix Palette Size'

e.g. -BoneLimit=0

-cs

By default Collada2POD 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

Equivalent in GUI: Geometry Options – Primitive Type

e.g. -PrimitiveType=TriList

Valid values:

TriList, TriStrips

-TriSort

Equivalent in GUI: Geometry Options – Triangle Sort

e.g. -TriSort=None

Valid values:

None, PVRTGeometrySort, D3DX, PVRTTriStrip

-PosType

The data type for vertex positions, see Section 5.2.9 Vertex Vector Formats for more information.

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, see Section 5.2.9 Vertex Vector Formats for more information.

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, see Section 5.2.9 Vertex Vector Formats for more information.

e.g. -ColorType=float

Valid values:

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

-UVWnype

The data type for vertex UVW co-ordinates (set n), see Section 5.2.9 Vertex Vector Formats for more information.

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, see Section 5.2.9 Vertex Vector Formats for more information.

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, see Section 5.2.9 Vertex Vector Formats for more information.

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, see Section 5.2.9 Vertex Vector Formats for more information.

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, see Section 5.2.9 Vertex Vector Formats for more information.

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’ in Geometry Options – General is functionally equivalent to ‘-PadDataTo=4’

e.g. -PadDataTo=1

Valid values:

1, 2, 4

Appendix C. Example PVRGeoPOD User Data Scripts

C.1. 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.
	*/

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

	/* return our data */
	h
)

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"
)
```

C.2. Mel Script

```
// Note: You can change the return type of the proceedures 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 seperated 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";
}
```

Imagination Technologies, the Imagination Technologies logo, AMA, Codescape, Ensigma, 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.