RenderWare Graphics

Artist Guide

Technical

Contact Us

Criterion Software Ltd.

For general information about RenderWare Graphics e-mail info@csl.com.

Developer Relations

For information regarding Support please email devrels@csl.com.

Sales

For sales information contact: rw-sales@csl.com

Contributors

RenderWare Graphics development and documentation teams.

Table of Contents

1. Introduction

In the RenderWare Graphics Exporters, there is a clear separation between *what* gets exported (the actual artwork), and *how* it gets exported (the RenderWare Graphics specific data). To achieve this goal, all of the RenderWare Graphics specific export functionality was moved to export templates. These templates are customizable XML files, containing all of the different export options. Using this mechanism, the artist can concentrate on the artwork, leaving the RenderWare Graphics export details to the technical user.

The documents 3dsmaxReferenceGuide.pdf and MayaReferenceGuide.pdf contain the information required by artists to create and export artwork.

This document is aimed at advanced/technical users of the RenderWare Graphics Exporters.

1.1 Other Documentation

- 3dsmaxTutorials.pdf and MayaTutorials.pdf documents are organized around a series of tutorials that take you through the basics of RenderWare Graphics as well as some of the issues you need to think about as you create 3D worlds. If you're new to RenderWare Graphics this is where you should start.

- OptimizeStaticGeom.pdf - This document is a case study of how to optimize static geometry using knowledge of the PlayStation 2 architecture.

- Three viewers can be used to view artwork exported using the RenderWare Graphics exporters. The viewers are: RenderWare Visualizer; Clump View and World View. There are two viewer documents describing the controls and setup of these viewers RenderWare Visualizer and Clump View and World View Viewers .

- RenderWare Graphics has a range of documentation material aimed mainly at developers but still useful for the artist. After installation, take a look at additional documents in the docs directory. The User Guide in particular should be useful to you as it covers a lot of material relevant to the artist.

- RenderWare's Fully Managed Support Service (FMSS) contains RenderWare Graphics Art Examples, which are available for download. In the FMSS https://support.renderware.com/, click *Downloads* on the left of the screen.

PDF format: Most RenderWare Graphics documents are in PDF format, which is a self-contained document format from Adobe. You'll need to install the (free) Acrobat Reader to view and print these. In some cases the quality is better in the printed form than on-screen.

The RenderWare Graphics PDF documents have been designed to be printed double-sided.

2. Exporter File Formats

The exporters, by default, export to `.rws` files. This is the RenderWare Graphics file format for exporting any type of RenderWare Graphics data. It can contain the contents of multiple `.dff`, `.bsp` files etc. and embeds textures. RenderWare Visualizer can be used to view `.rws` files.

The file formats `.anm`, `.dff`, `.bsp`, `.spl` are now considered legacy. The binary format of these files will continue to be supported, and RenderWare Graphics will continue to read them. There is no need to re-export existing artwork in `.rws` file format. However, it is recommended that you export to `.rws` files as these file formats *may* be removed in future releases.

File Formats

- `.rws` files contain RenderWare Graphics streaming information and can include animated hierarchy, static world, animation, spline, texture dictionary and effect dictionary information.
- `.rf3` files contains platform-independent intermediary export data in XML format.
- `.dff` files contain Animated Hierarchy (`RpClump`) information.
- `.bsp` files contain Static World (`RpWorld`) information.
- `.anm` files contain Animation (`RtAnimAnimation`) information.
- `.spl` files contain Spline (`RpSpline`) information.

2.1 The RF3 File Format

From RenderWare Graphics 3.5 a new type of file format, the `.rf3` file is supported. The `.rf3` file format is an XML based editable file, which contains all the raw exported data. Unlike the `.rws` files, which are RenderWare Graphics optimized binary files, `.rf3` files contain no rendering optimizations, and are simply a snapshot of the raw 3D data. Using the `rf3cc` compiler tool, users can compile these `.rf3` file into optimized platform-specific RenderWare Graphics binary files, (`.rws`, `.rp2`, `.rg1`, `.rx1` etc.).

By using `.rf3` files, you can fully customize and control your art tool path. This can be achieved by exporting all your artwork as `.rf3` files and recompiling them each time your export templates are modified. You can use Makefiles for checking for dependencies between template files and `.rf3` files. This also eliminates the task of going back to the modeler package and re-exporting the data each time an export option requires modification.

`.rf3` files can be manually edited and modified, viewed by any XML viewing tool, and also rendered by the RenderWare Visualizer.

As for any XML file format, `.rf3` files have well defined structure, content and semantics. We ship both DTD (Document Type Definition) and XML schema files named `rf3.dtd` and `rf3.xsd` respectively. They are installed to the Rw\Graphics\export\bin folder. If you intend to `extend.rf3` files, or just want to have a better understanding of how they are structured, you should consult these two files. Both provide the same information and it's your preference which one you use.

As well as providing a reference, both DTD and XML schemas are widely used for validating XML files. There are many freeware tools available on the Internet that can be used for validating `.rf3` files using `rf3.dtd` or `rf3.xsd`. One of them is `xmllint.exe` which is available from http://xmlsoft.org/ To use it type:

```
xmllint.exe --noout --dtdvalid rf3.dtd skinning.rf3
```

For more information about exporting `.rf3` files, see the Output Options section.

For more information about the `rf3cc` tool, see the rf3cc.pdf document.

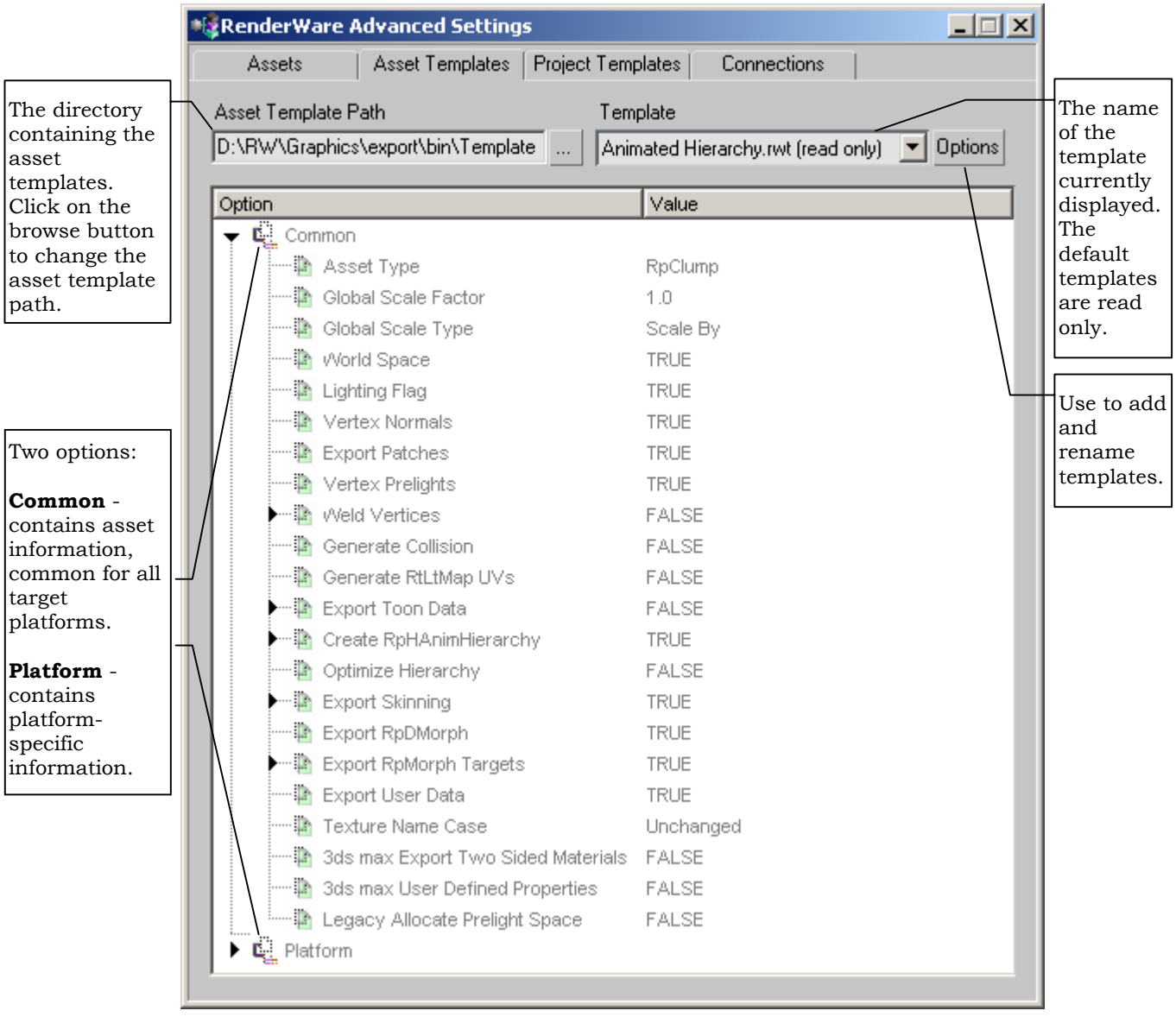
Modification of `rf3cc`

The source code for the `rf3cc` tool is provided for reference only; any modifications to `rf3cc` must be done through the Open Export plugins.

3. Export Templates

3.1 Asset Templates

Asset templates contain the options settings for exporting an asset in a particular format. All templates are stored as `.rwt` files and can be edited using the template editor in the advanced settings window of 3ds max or Maya or using a standard XML or text editor.



Each Asset template consists of two main sections:

- Common - containing all export options, common to all platforms. These options are picked up by the exporter, when exporting to any target platforms.
- Platform - containing all the platform-specific export options. Whenever exporting to a specific platform, the exporter will pick up the platform-specific options relevant to that target platform only.

There are four default asset templates which you can use, or you can create your own templates. The default templates are read-only and we recommend that you setup your own templates based on the default templates. The default templates are:

- Animated Hierarchy (page 11) - used to export a single hierarchy within a scene as a RenderWare Graphics container object, called `RpClump`.
- Animation (page 22) - used to export `RtAnimAnimation` animation data.
- Spline (page 23) - used to export `RpSpline` spline data .
- Static World (page 24) - used to export objects as a RenderWare Graphics World, called `RpWorld`. Worlds do not contain any hierarchy or animation information and are typically used for static level geometry in a game.

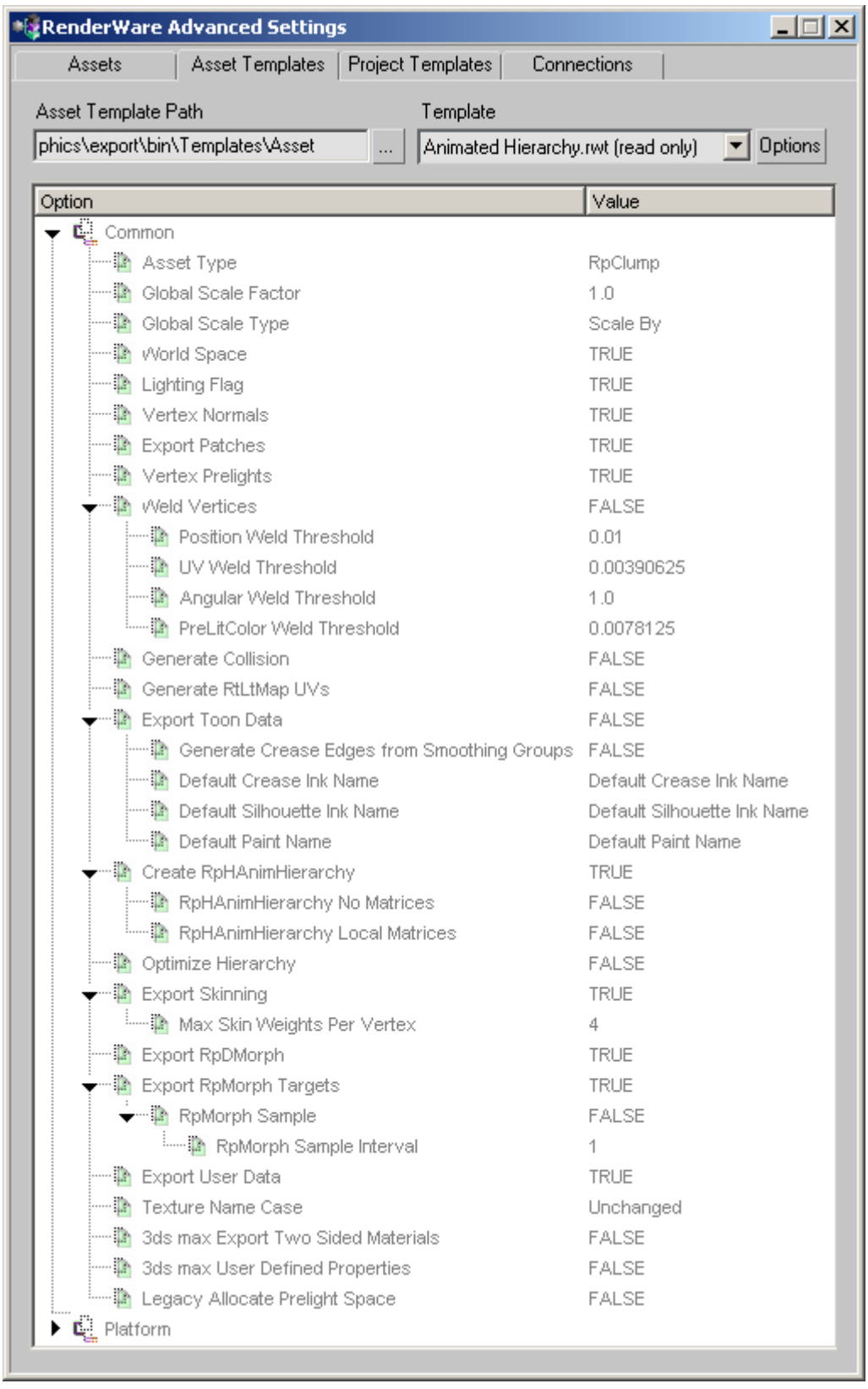
Asset Templates were referred to as batch node properties in RenderWare Graphics releases 3.3 and 3.4.

If you do wish to modify the default templates, make sure you modify their read only access attribute.

The options for the default settings are listed below, each one with its type (the first argument in the square brackets), and default value (the second argument).

Animated Hierarchy

The Animated Hierarchy template enables you to export objects within a hierarchy. This format is used to export a single hierarchy within the scene as a RenderWare Graphics container object, called `RpClump`.



The Animated Hierarchy template was referred to as the DFF Batch Node settings in previous RenderWare Graphics releases.

For more details on `RpClump`, refer to the *Fundamental Types* and *Dynamic Models* chapters of the User Guide.

Asset Type **[enum, RpClump]**

The type of asset to be exported. For an animated hierarchy, the asset type is `RpClump`.

Global Scale Factor **[float, 1.0]**

Multiplies all the geometry by a scaling factor. This controls the relative scale of the exported geometry. For example, a value of 2 will be twice the size of the original, 0.1 is one-tenth etc.

Global Scale Type **[enum, ScaleBy]**

Use *Scale By* to scale the object by the scale value entered. Use *Scale To*, to scale the object so that the scale value entered is the largest dimension of the clump's bounding box.

The value entered for the *Scale To* value determines the minimum size of the bounding box around the object. The geometry will be rescaled to ensure that this value is the largest dimension.

World Space **[bool, true]**

By default, the exporter removes all translation and rotation data from the root of an object hierarchy. In other words, if an object is upside down in the modeling package, it will appear the right way up when exported. This function overrides this behavior so that the same object will have the same orientation when exported.

Lighting Flag **[bool, true]**

This sets the lighting flag in the exported data. If the flag is set to FALSE, the exported geometry is not lit by real time lights.

Vertex Normals **[bool, true]**

Setting this option to FALSE will ignore all vertex normal data.

Export Patches **[bool, true]**

This option exports Bézier Patches. If not enabled, patches will be tessellated and exported as standard geometry and, in that case, any skinning won't be exported. If enabled, patches will be exported as `RpPatchMesh` objects.

Vertex Prelights [bool, true]

This option enables the export of prelight information.

Weld Vertices [bool, false]

Check this to enable welding on vertex position and UV value. This function snaps vertex properties within a given threshold. The actual number of vertices remains the same.

Position Weld Threshold [float, 0.0]

The threshold value for the vertex position.

UV Weld Threshold [float, 0.0]

The weld vertex UVs option.

Angular Weld Threshold [float, 1.0]

The angular threshold value for vertex welding. If the normals of the vertices differ by more than this value, they are not welded.

PreLitColor Weld Threshold [float, 0.01]

The pre-lit color threshold value for vertex welding. If the pre-lit colors of the vertices, in red, green or blue differ by more than this value, they are not welded.

Generate Collision [bool, false]

This option tells the exporter to generate RenderWare Graphics collision data with the exported geometry. This increases the execution time of the export and increases the size of the exported file. It should only be used if you intend to use RenderWare Graphics collision testing in your application.

Process Lightmaps [enun, No Lightmaps]

This option selects which type of lightmapping will be supported during export for the asset. Three choices are supported:

No Lightmaps: No lightmapping information will be generated with the asset.

Generate RtLtMap Uvs: Generates lightmapping information for use by the RtLtMap toolkit for generating lightmaps externally.

Export Native Lightmaps: Converts the native lightmaps directly from the modeling package into RenderWare optimized lightmaps.

Default lightmap size **[enum, 512x512]**

This option selects the default size of the lightmap texture that will be generated when the native lightmaps are converted. The size can be set to 16x16, 32x32, 64x64, 128x128, 256x256, 512x512, 1024x1024 and 2048x2048.

Sampling Factor **[float, 2]**

Lightmaps can be converted at a higher resolution and down sampled to a lower resolution for display. This can produce better quality lightmaps due to the higher sampling resolution without needing similar higher resolution lightmaps for display.

This option specifies the amount that converted lightmaps are scaled up by during the conversion process before being finally sampled back down to the correct size.

For example, if we are exporting 256x256 lightmaps from our modeling package, and select a default lightmap size of 64x64, we might set the sampling factor to 4. This way, the conversion is performed at the same resolution before being sampled down to the correct size.

Resample Imported **[bool, false]**

If this option is set to true, the native lightmaps being exported from the modeling package will be re-sampled to the same size as the conversion lightmap size before being converted. This is good for exporting very large lightmaps from the modeling package.

Use Greedy Rasterization **[bool, false]**

This option controls whether the edges of the lightmaps will be utilized when the native lightmaps are converted. Setting this to true can cause lightmapped regions to bleed into each other, however this uses less space in the lightmap and can be more optimal.

Rejection Factor **[float, 0]**

This option determines the minimum accepted area for a lightmapped region; any smaller area will be rejected during conversion. The larger this value can be without ruining the effect of the lightmaps, the more optimal the lightmap will be.

Density Factor [float, 1]

The density factor controls how densely the lightmapped surfaces are packed together in the exported light-maps. The lower this value is the more densely the lightmaps are packed and the lower the resolution of the lightmapped surfaces. The exporter assigns a default value that attempts to ensure optimal usage.

Dump Lightmaps [bool, false]

This option controls whether a `.png` version of the generated lightmap(s) should be exported in the same directory as the export file.

This option should be used only for debug purposes. If possible, you should use the lightmap textures generated in the texture dictionary. Note that PS2 Darkmaps will not be exported to `.png` files.

Lightmap Prefix String [string, "ltmap"]

This option controls the prefix used to generate the name of the lightmap textures created. Maximum string length is 4. Longer strings are truncated.

Lightmap Mipmapping [bool, Generic, GCN, XBOX:true, PS2:false]

This option control whether any lightmaps generated during export (when *Process Lightmaps* is set to *Export Native Lightmaps*) will be mipmapped. It's default value varies by platform and is therefore in the platform specific sections of the template.

Lightmap Color Depth [enum, Generic, GCN, XBOX:888, PS2:PAL8]

This option control whether the color depth of any lightmaps generated during export (when *Process Lightmaps* is set to *Export Native Lightmaps*). It's default value varies by platform and is therefore in the platform specific sections of the template.

Export Toon Data [bool, false]

Toon is a plugin in RenderWare Graphics called `RpToon`. It's a chargeable optional extra in the FX Pack. It's recommended that you read the Toon documentation, RpToonArtistGuide.pdf, to understand toon in more detail.

You will need to check this box to generate toon information for the object and allow the toon exporter to do its work. This does a fairly large amount of pre-processing of the object which is saved within the exported file so that it does not have to be recomputed at game load time.

Generate Crease Edges from Smoothing Groups [bool, false]

If this option is set to FALSE and you export a cube, only the silhouette edges are rendered. If this option is set to TRUE, each edge that lies between polygons with different smoothing groups is drawn as a crease edge (vertex normals for the object must also be exported for this option to work).

Default Crease Ink Name [string, "Default Crease Ink Name"]

This option names the ink used to draw the crease edges generated from all edge information. The name can be anything you like. This name is stored in the exported file and can later be matched against an ink dictionary to define the ink's properties.

Default Silhouette Ink Name [string, "Default Sillouette Ink Name"]

By default, all the silhouette edges in an object use the same ink when they are drawn. The ink is named with this box.

Default Paint Name [string, "Default Paint Name"]

By default, all materials in an object use the same paint, which is named here. The paint name is matched against a paint dictionary to define its properties.

Create RpHAnimHierarchy [bool, true]

This option creates the `RpHAnimHierarchy`, needed for animating the clump.

RpHAnimHierarchy No Matrices [bool, false]

The `RpHAnim` plugin can either use a large matrix array, which uses a lot of memory, or it can use a list of frames, which is less efficient but uses a lot less memory. Checking this option causes `RpHAnim` to use a frame list which reduces the memory requirements of the animation. This option creates an `RpHAnimHierarchy` with the `rpHANIMHIERARCHYNOMATRICES` flag.

RpHAnimHierarchy Local Matrices [bool, false]

All matrices stored are relative to the animation rather than being in world coordinates. This option creates an `RpHAnimHierarchy` with the `rpHANIMHIERARCHYLOCALSPACEMATRICES` flag.

Traverse Order **[enum, By Name]**

This option controls the order in which transforms are processed when traversing the hierarchy (and child animations) being exported. This is important as the order of the transforms in two hierarchies must be identical in order to share animations (for instance, you may have duplicated a hierarchy in order to export multiple animations for it).

Unsorted means that transforms will be traversed in whatever order the art package returns them. This ordering is not guaranteed to be consistent across duplicated hierarchies or even multiple exports.

By Name means that the child transforms at each node will be traversed in alphabetical name order. If identically named transforms are encountered as children of the same transform, a warning will be generated and the traverse order will be dependent on the order they are returned by the art package.

By Node Tag means that the child transforms at each node will be traversed by increasing node tag. If identically tagged or untagged transforms are encountered as children of the same transform, a warning will be generated and the traverse order will be dependent on the order they are returned by the art package.

Optimize Hierarchy **[bool, false]**

This option will tell the exporter to try and remove unused nodes from the exported hierarchy. This is useful if you have modeled a hierarchy with non-rendering nodes such as locators and groups that you do not need in the final clump. Animation on any optimized nodes will be passed onto the nodes' children. There are a number of restrictions that can prevent nodes being optimized out:

- Nodes that are bones, bounded to a skinned geometry, are only removed if they are not animated.
- Nodes with user custom attributes will not be removed.
- Nodes that have shapes attached to them are never removed.
- Nodes at the top of the hierarchy with multiple children are not removed. Doing so would split the hierarchy into two.
- Nodes with rotation animation that have children with translation are not removed. Doing so would cause the circular motion of the children to interpolate linearly.
- Nodes that have user data attached will not be removed.

The output window will print information on any nodes that have been optimized out of the hierarchy.

Caution should be exercised when using the optimize hierarchy option if you intend to share multiple animation files with the same `.dff`. For such sharing to work, the number of nodes in each animation must match.

Export Skinning **[bool, true]**

This option enables the usage of skinned geometries. When this option is enabled the exporter will create `RpSkin` objects from all skin modifiers that are affected by the selected bone hierarchy in the scene graph. When using this option you must select a node in the hierarchy containing the skeleton joints. The bound skin meshes don't need to be in the same hierarchy but will still be included in the exporter clump.

Some hardware platforms have restrictions on the number of bones that can be used in a single skeleton hierarchy. In particular, RenderWare Graphics on PlayStation 2 is currently limited to 64 bones in one hierarchy. The exporter generates `RpSkin` bones for all nodes in the selected skeleton hierarchy. You can use the *Enable Skin Splitting* option (in Project Templates) to overcome this problem, or alternatively the *Optimize Hierarchy* may be of use if you have non-joint nodes in your skeleton hierarchy. The exporter prints a message in the Output Window with the number of bones (nodes) it has created on a given export.

`RpSkin` can store up to four joint influences per vertex. If your scene contains more than four influences on any vertices then the highest four will be exported.

Max Skin Weights Per Vertex **[int, 4]**

We export up to four bone weights per vertex. You can limit the number of weights to less then four using this option. Generally, fewer weights mean faster processing, although this lowers the quality in some cases.

Export RpDMorph **[bool, true]**

This option will enable support to convert 3ds max or Maya morph data into RenderWare Graphics `RpDMorph` data.

Export RpMorph Targets **[bool, true]**

Enables or disables morphing, using `RpMorph` targets. Morph targets will NOT be created if the geometry has a blend shape/Morpher modifier, and *its Export RpDMoph* option is on, or has a skin modifier attached, and the *Export Skinning* option is turned on.

RpMorph Sample [bool, false]

When set to true, this uses the *RpMorph Sample Interval* to sample the animation and create a number of morph targets which are linearly interpolated to produce an animation. For example, if there are 50 frames in the animation, and the sample interval is 10 frames, 5 morph targets will be created.

When set to false, this option tells the exporter to query the object vertices from keyframed vertex animation and generate RenderWare Graphics morph targets in the atomic at the keyframe times.

RpMorph Sample Interval [int, 1]

This option is linked to the *RpMorph Sample* option and controls how many morph targets are created for a morph target animation.

Export UV Animation [bool, true]

This option enables or disables texture UV animation. If a texture has UV animation on it, and the option is set to true, the exporter will create a UV animation dictionary in the asset list, and store the various UV animations inside it.

Export User Data [bool, true]

In 3ds max, this option allows text strings defined in an object's User Defined Properties dialog to be exported for use in an application which uses the `RpUserData` plugin.

In Maya, this option exports blind data on vertices, faces and objects. Object blind data is supported only on Animated Hierarchy assets and is turned on by default. Use the *Window→General Editors→Blind Data Editor* to apply this data type.

In both applications, the flag enables or disables the export of custom attributes.

More details about blind data can be found section the Blind Data section of this document.

For more details about attaching custom data to objects, see the 5_MAXScript section.

Export Object Name [bool, false]

When this option is set to true, the exporter will store the object name as a user data entry.

User Data Entry Name `[string, "name"]`

Determines the name of the user data entry, to use with the Export Object Name option.

Export RpGeometry Name `[bool, true]`

Enables or disables object naming for RpGeometrys.

Export RwFrame Name `[bool, true]`

Enables or disables object naming for RwFrames.

Export RwCamera Name `[bool, true]`

Enables or disables object naming for RwCameras.

Export RpLight Name `[bool, true]`

Enables or disables object naming for RpLights.

Export RpMaterial Name `[bool, true]`

Enables or disables object naming for RpMaterials.

Export RwTexture Name `[bool, true]`

Enables or disables object naming for RwTextures.

Texture Name Case `[enum, Unchanged]`

This options allow you to preserve the existing case of textures or convert the names used for textures (and hence bitmap filenames etc.) into all upper or lower case (in case your platform has restrictions).

3ds max Export Two Sided Materials `[bool, false]`

Automatically creates backfacing triangles for geometries that have the 2-Sided Material option defined. Duplicate polygons to maintain 2-sided rendering.

3ds max User Defined Properties `[bool, false]`

This option is 3ds max specific and exports object user properties values as frame `RpUserData`.

Legacy Allocate Prelight Space **[bool, false]**

This option does *not* prelight the geometry. Instead, it allocates 'space' within the exported geometry for lighting values. The prelight values are initialized to white. This option is for legacy purposes only. It will be removed it a future release.

ADC Processing **[enum, Off]**

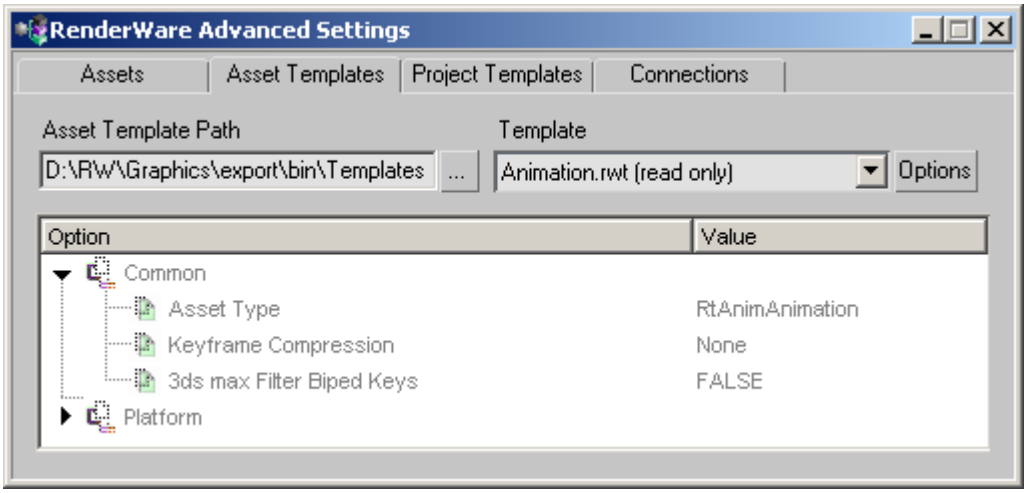
This option controls whether the RpGeometry objects exported are processed to support ADC flags. This should be set when a pipe override has been used that enables an ADC render pipeline. Whether or not the ADC processing should ignore winding order should be set to match the tristripper in use.

Convert Axis System **[bool, true]**

Some modelling packages use a different axis system to RenderWare and this needs to be fixed up in the exported data. When enabled this option does this fixup by burning the transform into the geometry and transforms of the exported data. With this option disabled (and always prior to version 3.7) this axis conversion was not applied at all to local space exports and was applied to world space exports by applying a transform to the root frame of the exported data.

Animation

The Animation template enables animation data to be exported. Animation assets only contain animation data and require an animated hierarchy asset to contain the geometry.



The Animation template was referred to as the ANM Exporter in previous RenderWare Graphics releases.

Asset Type **[enum, RtAnimAnimation]**

The type of asset to be exported. For an animation the asset type is `RtAnimAnimation`.

Keyframe Compression **[enum, None]**

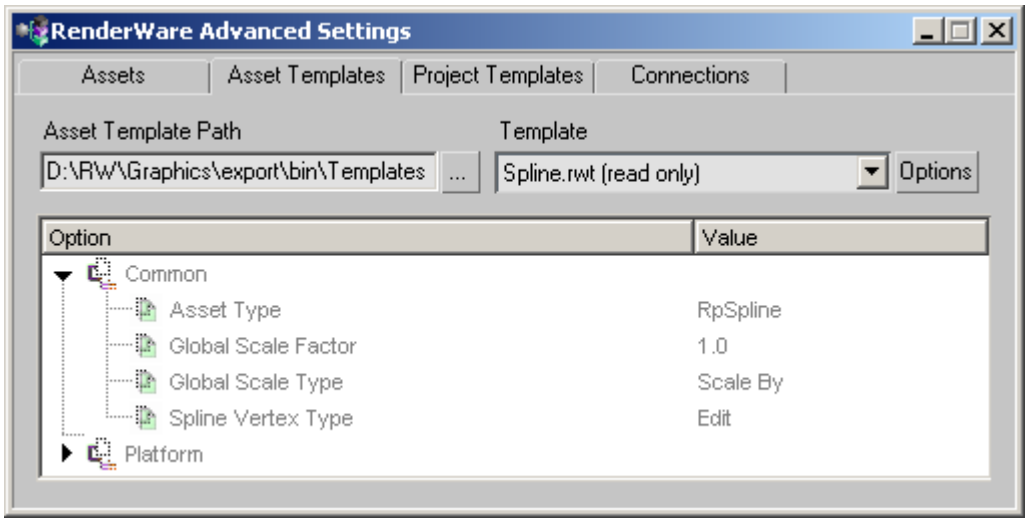
Enabling this option produces an animation using the `RtCmpKey` keyframe scheme and, hence, compressed animation.

3ds max Filter Biped Keys **[bool, false]**

This option is used if you have Triangle Pelvis options for the biped turned on and you have rotation on the spine link. In general, you should avoid Triangle Pelvis and not use this option because it will add extra keyframes and generate a much larger exported file.

Spline

A Spline asset is used to store spline data and requires an animated hierarchy asset to contain the world.



The Spline template was referred to as the SPL Exporter in previous RenderWare Graphics releases.

Asset Type

[enum, RpSpline]

The type of asset to be exported. For a spline the asset type is `RpSpline`.

Global Scale Factor

[float, 1.0]

Scale value for the control points of the exported spline. A value of 2 will be twice the size of the original, 0.1 is one-tenth, etc.

Global Scale Type

[enum, ScaleBy]

Use *Scale By* to scale the object by the scale value entered. Use *Scale To*, to scale the object so that the scale value entered is the largest dimension of the clump's bounding box.

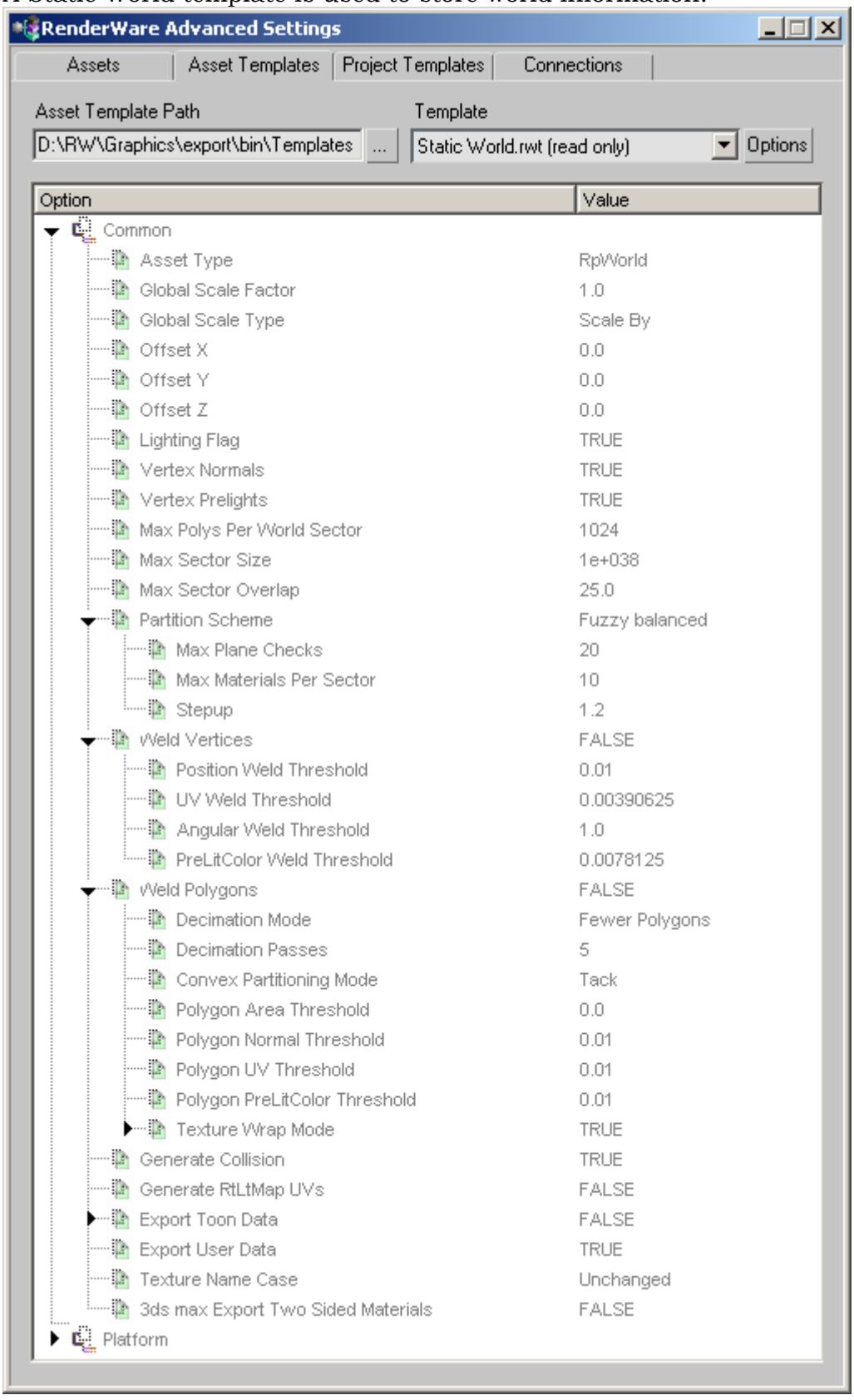
Spline Vertex Type

[enum, Edit]

This option controls whether the exported RenderWare Graphics spline will pass through the *Edit Points* or the *Control Verts* of a curve. Edit Points will normally give a better match.

Static World

A Static World template is used to store world information.



Worlds do not contain any hierarchy or animation information and are typically used for static level geometry in a game. They are automatically split into sections (PVS world sectors) that RenderWare Graphics uses to speed up the rendering process.

The Static World template was referred to as the BSP Exporter in previous RenderWare Graphics releases.

Asset Type **[enum, RpWorld]**

The type of asset to be exported. For a static world the asset type is `RpWorld`.

Global Scale Factor **[float, 1.0]**

Multiplies all the geometry by a scaling factor. This controls the relative scale of the exported geometry. For example, a value of 2 will be twice the size of the original, 0.1 is one-tenth etc.

Global Scale Type **[enum, ScaleBy]**

Use *Scale By* to scale the object by the scale value entered. Use *Scale To*, to scale the object so that the scale value entered is the largest dimension of the worlds bounding box.

Offset X **[float, 0.0]**

X World offset value.

Offset Y **[float, 0.0]**

Y World offset value.

Offset Z **[float, 0.0]**

Z World offset value.

Lighting Flag **[bool, true]**

This sets the lighting flag in the exported data. If the flag is set to FALSE, the exported geometry is not lit by real time lights.

Vertex Normals **[bool, true]**

This option enables/disables the export of normal information.

Vertex Prelights [bool, true]

This option enables/disables the export of prelight information.

Max Polys Per World Sector [int, 1024]

This controls the maximum number of polygons in a world sector. Fewer polygons mean smaller sectors which cull more accurately but take longer to process. The default value of 1024 should be fine for most worlds, but this number is very useful when optimizing performance.

Max Sector Size [float, 1e38]

This option controls the maximum size of sectors in the world. This option should only be changed if your application has a specific need to reduce the size of the generated world sectors.

Max Sector Overlap [int, 25.0]

This option controls the percentage of the size of the world sector that polygons are allowed to overlap when the world sector is split.

Partition Scheme [enum, Fuzzy Balanced]

This selects one of a number of 'partitioning schemes' (and activates a set of associated parameters) that can build a scene in a variety of ways.

Fuzzy Balanced: This is the default scheme. It is designed to lead to a fairly balanced BSP that minimizes splitting of polygons and textures. The additional parameter is "Max Plane Checks" (see the end of this section for a description).

Quick (preview): This scheme attempts to generate the shallowest possible BSP tree by choosing partitions that geometrically divide the scene into equal halves. It is a fast scheme, useful for previewing scenes. This does *not* honor partition hints or shield hints.

Closed (Indoor) Occluder: This is designed to work with PVS and works best with global parameter overlaps set to zero (note, this can sometimes lead to excessive splitting of polygons, so caution should be taken). This option should be selected if the environment comprises closed, indoor geometry, and is a largely orthogonal scene. The selected partitions are positioned so that they align with large surfaces, such as floors and walls, which are naturally good occluders. This may also implicitly identify some portals.

Open (Outdoor) Occluder: This is designed to work with PVS. This option should be selected if the environment comprises open, outdoor geometry, and is most efficient in a largely orthogonal scene. The selected partitions are positioned so that they align with large surfaces, such as floors and walls of buildings, which are naturally good occluders. This may also implicitly identify some thoroughfares, which can act like portals. The additional parameter is "Stepup".

Note, while the "Max Polys Per World Sector" parameter is honored, many sectors could give rise to significantly fewer polygons per sector. If this is the case, then the scene is not suitable for the open occluder scheme. The "general occluder" scheme is probably the most suitable in the case!

General Occluder: This is designed to work with PVS and works best with global parameters' overlaps set to zero (note, this can sometimes lead to excessive splitting of polygons, so caution should be taken). This option should be selected if the environment comprises a combination of open, outdoor geometry and closed indoor geometry, and is most efficient in a largely orthogonal scene. The partitions selected are positioned so that they align with large surfaces, such as floors and walls of buildings or rooms, which are naturally good occluders. This may also implicitly identify portals. The additional parameter is "Stepup".

Maximum Extent: This is designed to lead to a spatially balanced scene, and simply subdivides it by choosing partitioners in the middle of the largest axis of the sector. This does *not* honor partition hints or shield hints.

Material Boundary Occluder: This is principally designed to produce sectors with few different materials, and thus be efficient on platforms where texture swapping is expensive. It attempts to achieve this while maintaining good partitioning for PVS, and in many cases uses the same heuristics as the indoor occluder scheme.

Material Count: This scheme is similar to the Material Boundary Occluder, but it uses a slightly different approach. It is also significantly quicker to preprocess. The additional parameter is "Max Materials Per Sector".

Partition Hints: NB: This should only be used with a strong understanding of the way in which RenderWare Graphics World Sectors work! This scheme uses only partition hints to create the subdivision. The order in which the partition hints are chosen is largely based on their powers (see the 3dsmaxReferenceGuide.pdf or the MayaReferenceGuide.pdf for more details), but there is also a balancing heuristic at work. Because of this, some partition hints that are very close to each other's plane might occasionally be omitted to avoid unbalanced or very thin sectors from being generated. Note: Because this scheme can only use the supplied set of hints, it is possible that some sectors exceed the global parameter limits (such as maximum polygons per world sector). If this is not desirable, any one of the other schemes, which honors partition hints, should be chosen; these are guaranteed not to exceed the limits.

Max Plane Checks `[int, 20]`

This option controls the number of planes that are tested when trying to determine the best plane to divide a sector whilst generating data. A higher figure usually generates fewer polygons or less splits. However, very higher values don't necessary give better results, so experiment to find the optimum value.

Max Materials Per Sector `[int, 10]`

No sector will have more than this number of materials in it.

Stepup [float, 1.2]

Roughly governs the number of sectors generated in the scene. Since it is typical in such a scene that visibility increases with height, it is beneficial to create more sectors the lower one is located. Thus, the world is to create more sectors lower in the scene. Thus, the world is partitioned differently at different heights. Loosely speaking, "Stepup" is the increase in number of sectors as one descends. The value is only a guide to the build process, and some experimentation may be necessary.

Weld Vertices [bool, false]

Check this to enable welding on vertex position and UV value. This function snaps vertex properties within a given threshold. The actual number of vertices remains the same. To remove identical vertices polygon welding is required as well. Vertex welding act as a preparation for polygon welding and combination of both should be used for best results.

Position Weld Threshold [float, 0.01]

The threshold value for vertex position.

UV Weld Threshold [float, 0.00390625]

The threshold value for UV values.

Angular Weld Threshold [float, 1.0]

The angular threshold value for vertex welding. If the normals of the vertices differ by more than this value, they are not welded.

PreLitColor Weld Threshold [float, 0.0078125]

The pre-lit color threshold value for vertex welding. If the pre-lit colors of the vertices, in red, green or blue, differ by more than this value, they are not welded.

Weld Polygons [bool, false]

This runs a global conditioning algorithm over the world geometry. It will potentially optimize the geometry (i.e. reduce the number of polygons and vertices) but may not maintain the level of tessellation setup by the artist, hence lighting may not be as required. Should be used in combination with polygon welding.

Decimation Mode [enum, Fewer Polygons]

The approach geometry conditioning takes regarding edge decimation – i.e. the algorithm adopted for polygon welding.

Fewer Polygons: This minimizes the number of polygons, but some may be long and thin.

Smaller Polygons: Where possible, this avoids long and thin polygons, at the cost of less of a reduction in number.

Decimation Passes [int, 5]

This controls how thoroughly polygon welding searches for, and welds, polygons. Since the algorithm has a multi-pass approach, this sets the number of passes.

Convex Partitioning Mode [enum, Tack]

The approach geometry conditioning takes regarding re-triangulation after polygon welding.

Fan: This re-triangulates in preparation for tri-fanning.

Tack: This re-triangulates in preparation for tri-stripping. Note, this is the default and highly recommended setting, it would be a rare circumstance that would warrant the use of another method!

Ear: This tries to maximize the size of the triangles most central to the primitive. This might be appropriate where hierarchical culling is used, but usually tacking is still most beneficial.

Polygon Area Threshold [float, 0.0]

The area below which a polygon is considered to have zero area and is therefore culled.

Polygon Normal Threshold [float, 0.01]

The normal threshold value for polygon welding. If the total of the weighted normal area of one polygon differs by more than this fraction to the other, the polygons are not welded.

Polygon UV Threshold [float, 0.01]

The UV threshold value for polygon welding. If the total of the weighted UV area of one polygon differs by more than this fraction to the other, they are not welded. (This even accounts for mismatching two neighboring polygons which are the mirror of each other.)

Polygon PreLitColor Threshold [float, 0.01]

The pre-lit color threshold value for polygon welding. If the total of the weighted pre-lit color area of one polygon differs by more than this fraction to the other, the polygons are not welded.

Texture Wrap Mode [bool, true]

If FALSE, the texture wrap modes 0..7 (see below) are overridden and all set to FALSE). TRUE otherwise.

Texture Wrap Mode [0..7] [bool, true]

If TRUE, UV coordinates are aligned prior to welding. For example, a stretch of three polygons with U value's 0..1, 0..1, 0..1 get realigned to 0..1, 1..2, 2..3. Subsequently, the three polygons become one with a U-range of 0..3. Without this, the three polygons are mismatched under 'polygon UV threshold' and do not get welded. Note, each UV set, 0..7, is individually controllable. Note also, any UV set used for lightmapping is always overridden and set to FALSE.

Generate Collision [bool, true]

This option tells the exporter to generate RenderWare Graphics collision data with the exported world. This increases the execution time of the export and increases the size of the exported file. It should only be used if you intend to use RenderWare Graphics collision testing in your application.

Process Lightmaps [enun, No Lightmaps]

This option selects which type of lightmapping will be supported during export for the asset. Three choices are supported:

No Lightmaps: No lightmapping information will be generated with the asset.

Generate RtLtMap Uvs: Generates lightmapping information for use by the RtLtMap toolkit for generating lightmaps externally.

Export Native Lightmaps: Converts the native lightmaps directly from the modeling package into RenderWare optimized lightmaps.

Default lightmap size [enum, 512x512]

This option selects the default size of lightmap texture that will be generated when the native lightmaps are converted. The size can be set to 16x16, 32x32, 64x64, 128x128, 256x256 and 512x512.

Sampling Factor [float, 2]

Lightmaps can be converted at a higher resolution and down sampled to a lower resolution for display. This can produce better quality lightmaps due to the higher sampling resolution without needing similar higher resolution lightmaps for display.

This option specifies the amount the converted lightmaps are scaled up by during the conversion process before finally being sampled back down to the correct size.

For example, if we are exporting 256x256 lightmaps from our modeling package, and select a default lightmap size of 64x64, we might set the Sampling Factor to 4 so that the conversion is performed at the same resolution before being sampled down to the correct size.

Resample Imported [bool, false]

If this option is set to true, the native lightmaps being exported from the modeling package will be re-sampled to the same size as the conversion lightmap size before being converted. This is good for exporting very large lightmaps from the modeling package.

Use Greedy Rasterization [bool, false]

This option controls whether the edges of the lightmaps will be utilized when the native lightmaps are converted. Setting this to true can cause lightmapped regions to bleed into each other, however this uses less space in the lightmap and can be more optimal.

Rejection Factor [float, 0]

This option determines the minimum accepted area for a lightmapped region; any smaller area will be rejected during conversion. The larger this value can be without ruining the effect of the lightmaps, the more optimal the lightmap will be.

Density Factor [float, 1]

The density factor controls how densely the lightmapped surfaces are packed together in the exported light-maps. The lower this value is the more densely the lightmaps are packed and the lower the resolution of the lightmapped surfaces. The exporter assigns a default value that attempts to ensure optimal usage.

Dump Lightmaps [bool, false]

This option controls whether a .png version of the generated lightmap(s) should be exported in the same directory as the export file.

This option should be used only for debug purposes. If possible, you should use the lightmap textures generated in the texture dictionary. Note that PS2 Darkmaps will not be exported to .png files.

Lightmap Prefix String [string, "ltmap"]

This option controls the prefix used to generate the name of the lightmap textures created. Maximum string length is 4. Longer strings are truncated.

Lightmap Mipmapping [bool, Generic, GCN, XBOX:true, PS2:false]

This option controls whether any lightmaps generated during export (when *Process Lightmaps* is set to *Export Native Lightmaps*) will be mipmapped. It's default value varies by platform and is therefore in the platform specific sections of the template.

Lightmap Color Depth [enum, Generic, GCN, XBOX:888, PS2:PAL8]

This option controls the color depth of any lightmaps generated during export (when *Process Lightmaps* is set to *Export Native Lightmaps*). It's default value varies by platform and is therefore in the platform specific sections of the template.

Export Toon Data [bool, false]

Toon is a plugin in RenderWare Graphics called RpToon. It's a chargeable optional extra in the FX Pack. It's recommended that you read the Toon documentation to understand toon in more detail.

You will need to check this box to generate toon information for the object and allow the toon exporter to do its work. This does a fairly large amount of pre-processing of the object which is saved with the BSP so it doesn't have to be recomputed at game load time.

Generate Crease Edges from Smoothing Groups [string, ""]

If this option is set to FALSE, and you export a cube, only the silhouette edges are rendered. If this option is set to TRUE, each edge that lies between polygons with different smoothing groups is drawn as a crease edge (vertex normals for the object must also be exported for this option to work).

Default Crease Ink Name `[string, "Deafult Create Ink Name"]`

This option names the ink used to draw the crease edges generated from all edge information. The name can be anything you like. This name is stored in the exported file and can later be matched against an ink dictionary to define the ink's properties.

Default Silhouette Ink Name `[string, "Default Silhouette Ink Name"]`

By default, all the silhouette edges in an object use the same ink when they are drawn. The ink is named with this box.

Default Paint Name `[string, "Default Paint Name"]`

By default, all materials in an object use the same paint, which is named here. The paint name is matched against a paint dictionary to define its properties.

Export UV Animation `[bool, true]`

Enables or disables texture UV animation. If a texture has UV animation on it, and the option is set to true, the exporter will create a UV animation dictionary in the asset list, and store the various UV animations inside it.

Export User Data `[bool, true]`

Export user data option. When this is checked, RenderWare Graphics exports any Custom Attributes associated with your scene.

Export Object Name `[bool, false]`

When this option is set to true, the exporter will store the object name as user data.

User Data Entry Name `[string, "name"]`

Determines the name of the user data entry, to use with the Export Object Name option.

Export RpMaterial Name `[bool, true]`

Enables or disables object naming for RpMaterials.

Export RwTexture Name `[bool, true]`

Enables or disables object naming for RwTextures.

Texture Name Case [enum, Unchanged]

These options allow you to preserve the existing case of textures or convert the names used for textures (and hence bitmap filenames etc.) into all upper or lower case (in case your platform has restrictions).

3ds max Export Two Sided Materials [bool, false]

Automatically creates backfacing triangles for geometries that have the 2-Sided Material option defined. Duplicate polygons to maintain 2-sided rendering.

ADC Processing [enum, Off]

This option controls whether the RpAtomics exported are processed to support ADC flags. This should be set when a pipe override has been used that enables an ADC render pipeline. Whether or not the ADC processing should ignore winding order should be set to match the tristripper in use.

Graphic Pipeline Customization

Users can choose which RenderWare Graphics' pipeline to use by changing the pipeline specific options, under the platform section in each asset template. To change a specific rendering pipeline, simply change the assigned pipeline name.

You can add additional pipelines overrides, for pipelines that are not in the pipeline option list. Simply add a new string option under the platform section, and name it as the pipeline name (you may need to remove the pre-defined option if overriding one of the standard pipes). Assign it with the customized pipeline name or a string representing the pipeline id, e.g. “rwPDS_G3x_APLSkin_MatPipeID” or “44”.

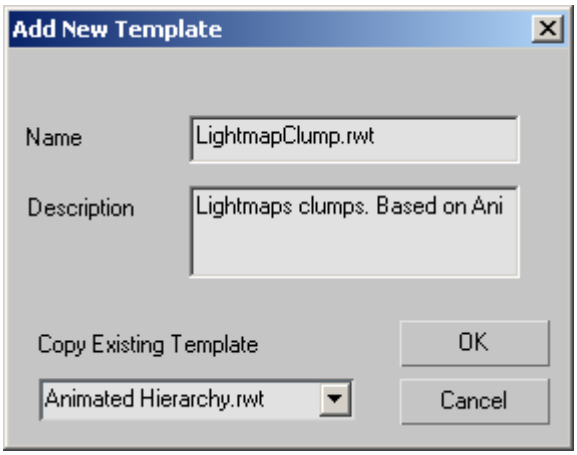
Creating Asset Templates

It is recommended that you setup your own asset templates, as the default templates may not contain the specific options you require. For example, you may want to setup templates to:

- export lightmaps
- export pre-lit geometry
- export toon objects
- export specific animation
- export partitioning

To create templates:

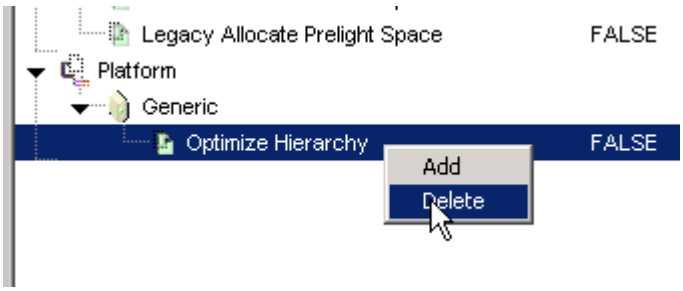
1. *RenderWare→Advanced Settings→Asset Templates* tab
2. Click on the *Options* button and choose *Add*



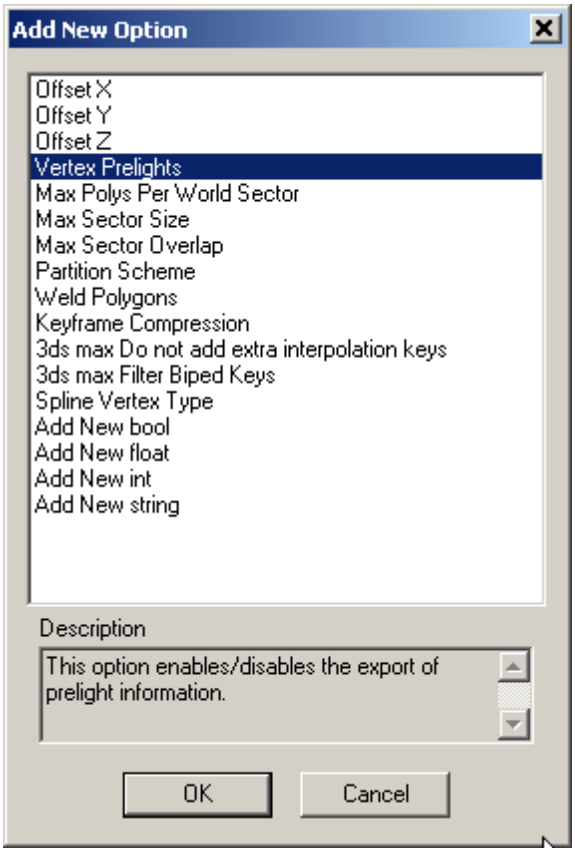
3. Type a template name with the extension `.rwt`
4. Type a description. This description will be displayed when this template is chosen in the Assets tab.
5. Select an existing template to copy. This option is recommended as the options can then be optimized.

Adding or Deleting options

To add or delete options, right click within a section and choose the relevant menu option.



Adding a new option will bring up the Add New Option Dialog.

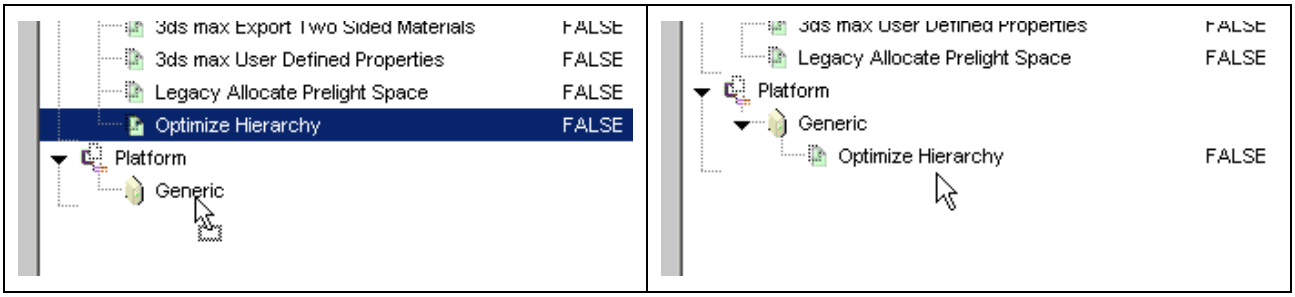


Using this dialog you can either add a custom option by selecting one of the "Add New *X*" entries or you can select one of the standard types. Only the standard types that are not already in, within the currently selected section, are displayed.

Moving options

Options can be moved to any location within the template by dragging the option to the required location.

Left click on the option you want to move and then move the mouse cursor over the place you want to move the option to, and then release the mouse button.

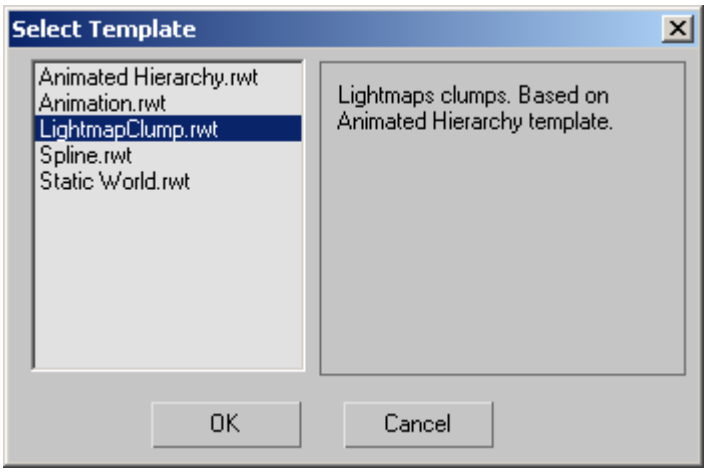


In order to add options from other template files, you will need to manually add custom options with the correct name or edit the template files in an XML or text editor.

To use customized templates:

1. *RenderWare→Advanced Settings→Assets* tab

2. Select an asset, in the template column, select the browse button to choose a template.

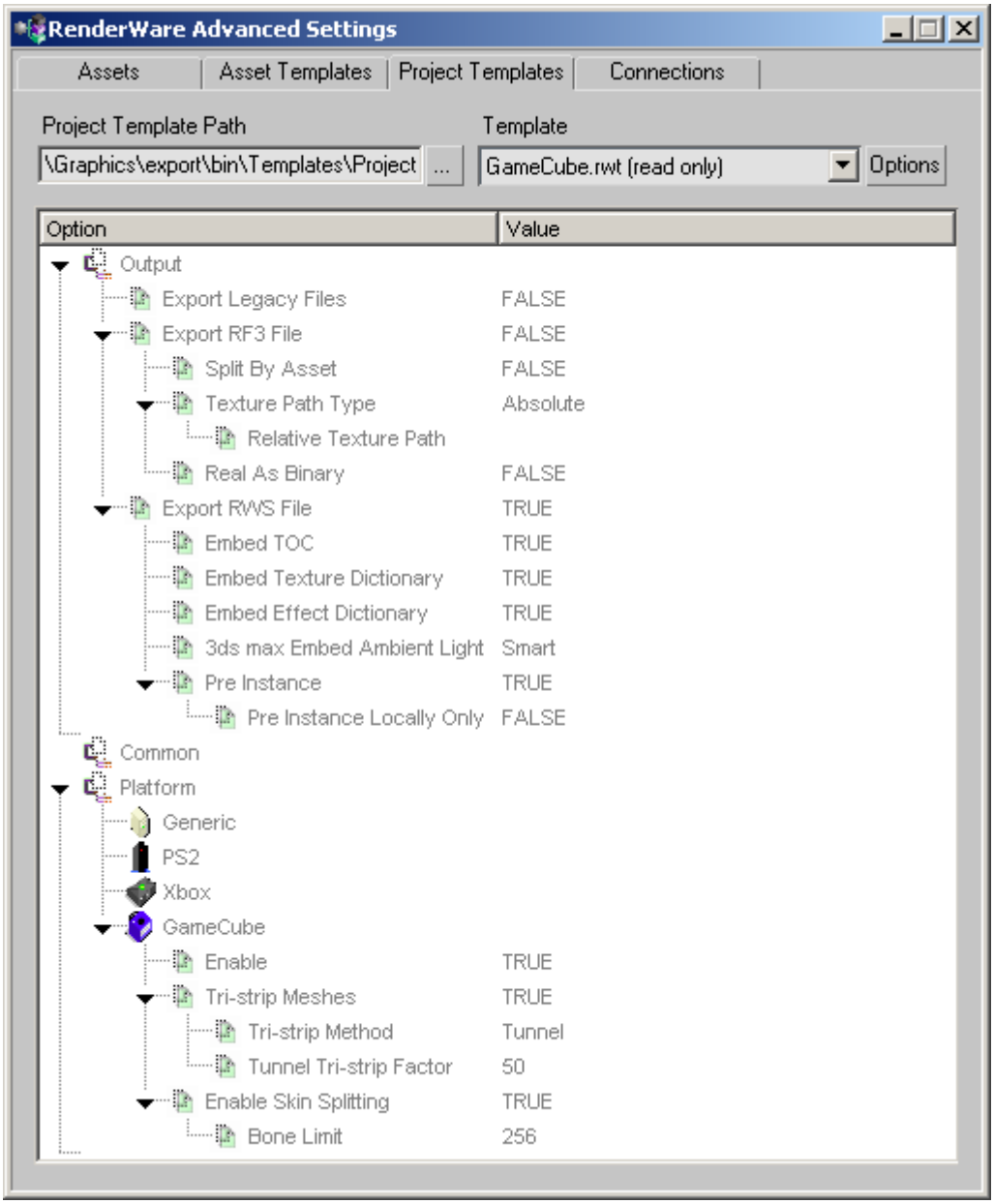


Deleting Asset Templates

The asset templates can be found in the *Asset Template Path* specified in the Asset Templates tab. To delete the templates, use Windows Explorer as they can not be deleted directly from the exporter.

3.2 Project Templates

The project templates contain all the export settings specified for an entire project. These project settings are shared between all assets belonging to that project. Once these settings are modified, they affect all the assets, linked to that project template. Just like asset templates, project templates are stored as `.rwt` files and can be edited using the template editor in the advanced settings window of 3ds max or Maya or using a standard XML or text editor.



The project templates consist of three main sections:

- Output Options – specifying the output file format and other output related features.
- Common Options – options that are shared between all target platforms.
- Platform Options – options that are specific to a certain platform.

Project Template Path: The directory containing the project templates.

We've included four default project templates, each one fine tuned for each specific platform. Please note that these are only suggested settings that can always be customized for a specific project. Each default project template is named after the target platform that it is tuned for, and only has its platform option enabled in the platform section. You should also note that, although these default project templates only enable one platform, it is possible to enable and specify settings for all the platforms within a project template, which will result in a multi-platform export.

- `Generic.rwt` used for exporting files to the PC (e.g. D3D8 or OpenGL), exports files with the extension `.rws`.
- `GameCube.rwt` used for exporting files to GameCube exports files with the extension `.rg1`.
- `PS2.rwt` used for exporting files to PlayStation 2 exports files with the extension `.rp2`.
- `Xbox.rwt` used for exporting files to Xbox exports files with the extension `.rx1`.

The project templates contains the setup information. The actual project template used is specified in the assets template.

Output Options

Export Legacy Files **[bool, false]**

Controls whether to export to the legacy file formats: `.anm`, `.bsp`, `.dff`, `.dma` or `.spl` files. The name of the exported files is based on the current scene's export path plus the asset name plus the extension for the asset type.

This file formats contain the following information:

ANM

A `.anm` file stores only animation data and requires a `.dff` file of the same name to contain the geometry. To setup a `.anm` export format, select an object or joint within the hierarchy you wish to export before running the exporter.

The `.anm` file format refers to assets using the *Animation* template.

BSP

A `.bsp` file contains worlds that do not contain any hierarchy or animation information and are typically used for static level geometry in a game. They are automatically split into sections (BSP world sectors) that RenderWare Graphics uses to speed up the rendering process. A RenderWare Graphics binary stream, by convention, stores a single world.

The performance of the model as .bsp or .dff greatly depends on what's in view. The moment the entire model is not in view, the .bsp will be faster. The concept of a .bsp is that only what needs to be rendered, is rendered. The .bsp exporter divides the model up into sectors and only the sectors in view are rendered.

Use .bsp for worlds and .dff for dynamic objects. If the object is never going to move or do anything, it should be part of the .bsp, not a .dff.

The .bsp file format refers to assets using the *Static World* template.

DFF

A .dff file contains a single hierarchy as a RenderWare Graphics container object, called RpClump (see the *Fundamental Types* and *Dynamic Models* chapters of the User Guide for more details on RpClump). Depending on the options you select, the saved RpClump file will contain the object hierarchy, hierarchical animation, skinned animation, and morph target animation.

The .dff file format refers to assets using the *Animated Hierarchy* template.

DMA

A .dma file stores only delta morph animation data and requires a .dff file containing one or more atomics setup for delta morph rendering. .dma files can only be exported at the same time as a .dff export.

SPL

A .spl file stores only spline data.

The .spl file format refers to assets using the *Spline* template.

Export RF3 File **[bool, false]**

The Export to RF3 option controls whether to export to the platform-independent XML file format. The name and location of the exported file is based on the current scene's export path plus the project name plus the .rf3 extension.

Split By Asset **[bool, false]**

The RF3 Split By Asset controls whether to export a single RF3 file or an RF3 file for each asset.

Texture Path Type **[enum, Absolute]**

The RF3 Texture Path Method determines how to stream out the texture path. This option can be set to *Relative*, *Absolute* and *Relative To Export Path*. Set this option to *Absolute* to export the full path of the textures. When this option is set to *Relative*, all paths are streamed out relative to the given texture path specified by the *Relative Texture Path* option. Finally, when this option is set to *Relative To Export Path*, all texture paths are streamed out relative to the file export path.

Relative Texture Path **[string, ""]**

The relative texture path, which is only valid if the *RF3 Texture Path Type* is set to *Relative*.

Real as Binary **[bool, false]**

Determines whether to export all real numbers in the file using a binary representation, or a text representation. (Use binary for greater accuracy).

Export RWS File **[bool, true]**

By default, this option is set to TRUE. This option exports all data from all/selected RenderWare Graphics Assets into one `.rws` file. The name and location of the exported file are based on the current scene's export path plus the project name plus the `.rws` extension. If the *Export RWS File* option is enabled, all binary data/chunks that were previously stored in `.anm`, `.bsp`, `.dff`, `.dma` and `.spl` files, will be contained in one `.rws` file. This means that one file can store worlds, animation, splines and clumps at the same time, so you're able to view your entire scene in one instance. Texture dictionaries can also be embedded in `.rws` files.

If you select assets and right click and choose *Export*, all exported assets will be exported to an `.rws` file in the order the assets were selected. If you choose *RenderWare→Export*, all assets will be exported to an `.rws` file and the chunks will be ordered alphabetically.

This file format as set to TRUE by default and is the recommended file format.

Embed TOC **[bool, true]**

The Embed Table of Contents option embeds separate chunks at the beginning of the `.rws` file and creates a table of contents listing all chunks. This option uses the RenderWare Graphics `RtTOC` toolkit.

Embed Texture Dictionary [bool, true]

The Embed Texture Dictionary option controls whether we should export all used scene textures into a platform-independent texture dictionary. The texture dictionary will be embedded at the start of the `.rws` file.

Embed Effect Dictionary [bool, true]

The Embed Effect Dictionary option controls whether to generate an `RpMatFX` effect dictionary for all RenderWare materials produced during export.

3dsMax Embed Ambient Light [bool, false]

The Embed Ambient Light option creates and embeds an extra `RpLight` for the ambient light.

Embed Background Color [bool, false]

Embeds a background color in the RWS file. In 3ds max this color is taken from the environment background color. In Maya the color is taken from the background color of the first visible camera found in the scene. If no cameras are visible it is taken from the camera for the currently selected viewport. If no viewport is selected black is exported.

Pre Instance (except Generic) [bool, true]

The Pre Instance option controls whether to generate platform-specific versions of the exported `.rws` file. When this option is enabled, the exporters will connect to remote console targets as a part of the export process and save extra versions of the `.rws` file specific to those platforms. Platform-specific files are placed in the same directory as the `.rws` file and given a different extension.

PLATFORM	EXTENSION
PlayStation 2	`.rp2`
GameCube	`.rg1`
Xbox	`.rx1`

The list of remote console targets that are used in the pre-instancing process is set via the Remote Connections settings. Each target has a 'Pre Instance' option and only those targets with this setting enabled will be connected to. If there are multiple targets enabled for the same platform, the first target of that platform that can be connected to will be used.

The documentation for the *RenderWare Graphics Instancing Tool*, Instance.pdf, provides more information on the instancing process and the motivations for using it.

Pre Instance Locally Only (except Generic) `[bool, false]`

Determines whether pre-instancing should be performed locally only (on the local host machine) or also on remote console targets. When this option is set to TRUE, only texture dictionaries and rendering pipelines are pre-instanced, but not the geometry data. In order to assure a full pre-instance operation, set your remote connections and turn this option off.

Common Options

Using a customized template, you can add common platform-independent options to this section.

Platform Options

All platforms have the following default options:

Enable `[bool, *]`

This enables the exporter to export files to this platform.

Tri-strip Meshes `[bool, true]`

This option post-processes exported geometry into a tri-strip format which is more efficient at run-time on most hardware platforms. This may not suit all applications and is therefore an export option. The exporter provides multiple ways of generating tri-strips. These have varying trade-offs between export time and tri-strip efficiency. It is a good idea to try them out and discover which one works best for you.

Tri-strip Method `[enum, Generic:Tunnel, PS2:Tunnel, GCN:Tunnel, XBOX:CacheAware]`

The Tri Strip Meshes' options are:

TRI STRIP MESHES	DESCRIPTION
PreProcess	This tri-stripper builds three possible tri-strips starting from the lowest adjacency triangle and taking the longest of these strips. This is fast and produces reasonable results, but Tunnel should be used when the best quality results are required (i.e. final in-game artwork).
Tunnel	This tri-stripper will produce the best results for most artwork. It should be used with a high quality setting to generate final artwork. It can be used with a low quality setting to preview artwork.
IgnoreWindingOrder	These versions of the tri-strippers ignore winding order restrictions when creating tri-strips. This improves efficiency but means the face direction of the generated triangles may be incorrect and should only be used if backface culling is not used at run-time.
CacheAware	Produces tri-strips tuned for the vertex cache present on NVIDIA hardware (Xbox, GeForce).

Tri-strip Factor `[int, 15]`

The quality setting used by the tunnel tri-strip method. The quality improves the higher the setting.

Limit UVs (PlayStation 2 only) `[enum, Texel Limit]`

If either Texel Limit or UV limit are selected, UV values will be hard limited to the *Min UV Value* and *Max UV Value* entered below. This option is used to ensure that UV values do not go outside the range permitted by the target platform. Texel limiting will use the corresponding texture to calculate the UV limit. Limiting will also add extra vertices on the boundaries, which can affect tri-stripping.

Min UV Value (PlayStation 2 only) `[float, -2048]`

The minimum UV value for the target platform into this box which can be negative if the platform supports negative values.

Max UV Value (PlayStation 2 only) [float, 2047]

The maximum UV value for the target platform into this box.

Generate Darkmaps (PlayStation 2 only) [bool, true]

Lightmaps on the PlayStation 2 are implemented differently than on other platforms and are need to be inverted to Darkmaps. This option should always be set to true for PlayStation 2 exports.

Process Base DarkMap texture (PlayStation 2 only) [bool, true]

Darkmaps for the PlayStation 2 require the 'luminance' value to be stored inside the alpha channel of the base textures. This option should always be set to true for PlayStation 2 exports.

Enable Skin Splitting [bool, true]

This option enables skin splitting. Some platforms impose a limit on the number of bones that are permissible in a single object. An object that requires more bones, than the bone limit, can now be supported using this option. The option splits the geometry such that each mesh will use, at most, the *Bone Limit* number of bones matrices. The source geometric data is not modified. Instead, the underlying meshes are rearranged such that each mesh requires, at most, the specified bone limit.

Bone Limit [int, Generic:1024, PS2:64, GCN:256, XBOX:60]

The bone limit varies depending on your target platform.

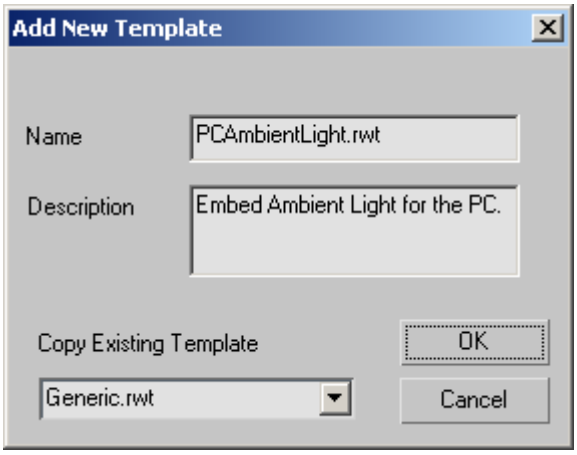
Creating Project Templates

It is recommended that you setup your own asset templates. The default templates may not contain the specific options you require. For example, you may want to setup templates to:

- export legacy file formats
- export to RF3 file format
- embed ambient light in an RWS export

To create templates:

1. *RenderWare→Advanced Settings→Project Templates* tab
2. Click on the *Options* button and choose *Add*



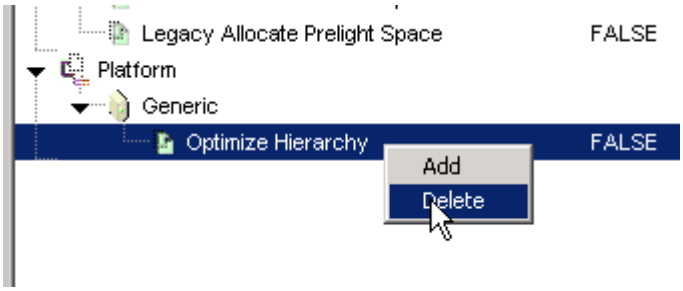
3. Type a template name with the extension `.rwt`

4. Type a description. This description will be displayed when this template is chosen in the Assets tab.

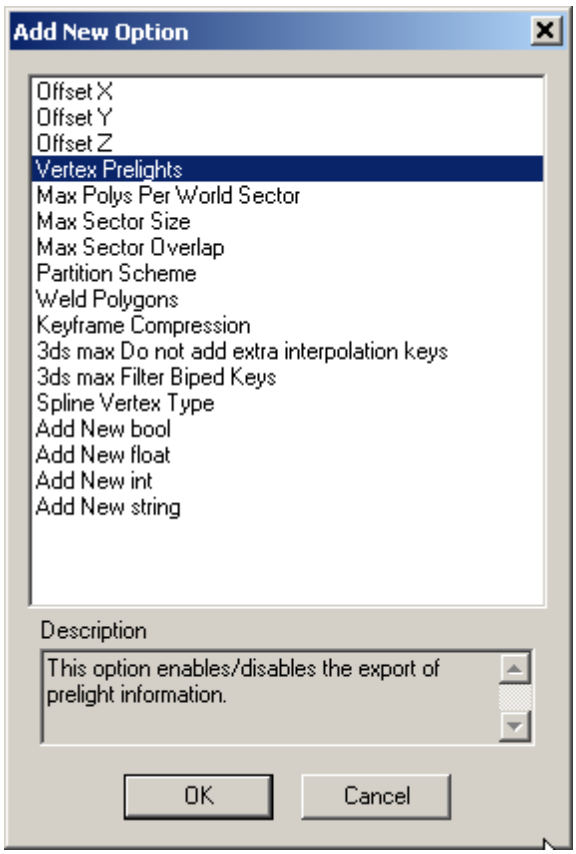
5. Select an existing template to copy. This option is recommended as the options can then be optimized.

Adding or Deleting options

To add or delete options, right click within a section and choose the relevant menu option.



Adding a new option will bring up the Add New Option Dialog.



Using this dialog, you can either add a custom option by selecting one of the "Add New *X*" entries or you can select one of the standard types. Only the standard types that are not already in, within the currently selected section, are displayed. It is not possible to delete the standard Export options from the Output section.

Moving options

Options can be moved to any location within the template by dragging the option to the required location, except the Output section.

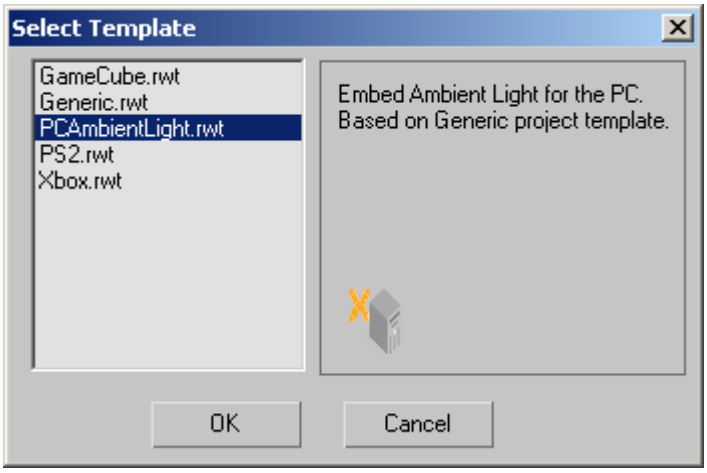
Left click on the option you want to move and then move the mouse cursor over the place you want to move the option to, and then release the mouse button.

In order to add options from other template files, you need to manually add custom options with the correct name or edit the template files in an XML or text editor.

To use customized templates:

1. *RenderWare→Advanced Settings→Assets* tab

2. To the right of the Project Template box, click the browse button to choose a template.



Deleting Project Templates

Project Templates can be found in the *Project Template Path* specified in the Project Templates tab. To delete the templates, use Windows Explorer as they can not be deleted directly from the exporter.

3.3 Option Precedence

The same option can live in multiple templates, or even in different sections of the same template. This is useful in various situations, an example being having a common tristripper set in your project template but overriding it in particular asset templates. For this reason, the option precedence is important to consider. The order of precedence, from lowest to highest, is as follows:

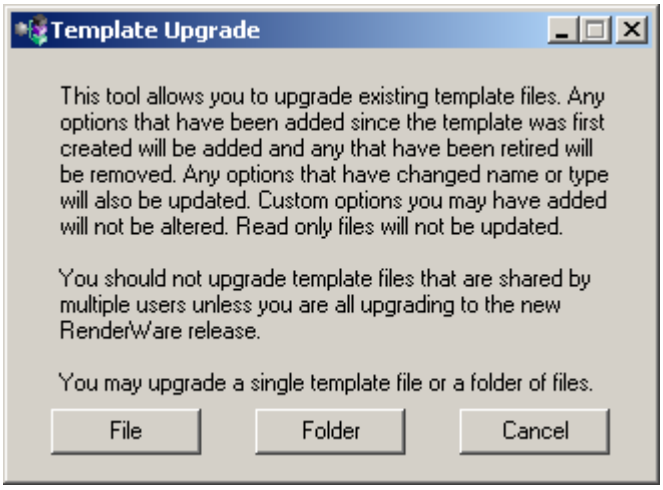
1. The common section of the project template.
2. The current platform section of the project template.
3. The common section of the asset template.
4. The current platform section of the asset template.
5. Per-asset artist options

3.4 Updating Template Files

Template files from a previous version of the RenderWare Graphics exporters can be updated to include new and updated options as well as removing older obsolete options. Updating template files can be performed by using the *Update Templates* tool from the *RenderWare->Tools* menu. This tool checks for any options that have been added, removed or renamed since the template was first created and adds, removes and renames them accordingly. Any custom options added in your template files will not be altered. Read only templates will not be updated.

It is recommended that template files are not updated, unless all parties using them upgrade to the same RenderWare Graphics version.

Running the tool brings up the following dialog box, giving 2 options. You can choose to upgrade either a single template file or an entire folder of template files.



Template files can also be updated using a script. See the MEL Script and MAXScript sections.

4. MEL Script

4.1 Provided MEL Commands

RenderWareNodeExport

`RenderWareNodeExport` takes the following arguments:

- **s** exports selected assets.
- **a** exports all assets.

RenderWareViewExport

`RenderWareViewExport` takes the following arguments:

- **s** views selected assets.
- **a** views all assets.

RenderWareBatchExport

`RenderWareBatchExport` loads all Maya scenes from a specified folder and exports all assets from each scene. It takes two arguments:

- boolean value specifying recursion
- path of the directory containing the scenes. If it is an empty string, a dialog will be displayed for browsing directories.

For example, to export scenes from a command line, you can export one of the following commands:

```
mayabatch –command "RenderWareBatchExport 1 \"C:/MayaScenes\""
```

The output from the above script is displayed in the command prompt window.

```
maya –batch -command "RenderWareBatchExport 1 \"C:/MayaScenes\""
```

The output from the above script is displayed in Maya 's output window.

RenderWareAutoCreateAsset

Analyzes the current selection in the scene, creates an asset for the selected objects, and sets up the appropriate options. This command returns the name

of the RenderWareAsset node created. Please note that, in the case of animated hierarchies, this node may have children.

RenderWareAutoCreateAssets

Automatically creates assets for all the nodes in the scene. Returns the names of all asset nodes created.

RenderWarePluginInfo

Pops up the plugin information dialog.

RenderWareAdvancedSettings

Pops up the advanced settings dialog.

RenderWareTriangulate

Triangulates the selected nodes.

4.2 Creating Assets

If you need to create RenderWareAsset nodes from your own scripts, you can either use the provided RenderWareAutoCreateAsset command or you can create a node from scratch using the standard MEL functionality. When doing the latter, you will need to ensure the attributes of the node created are set to appropriate values. Badly formed assets will certainly not export correctly and may cause problems when displayed in the user interface. For example:

```
createNode RenderWareAsset;
// Result: RenderWareAsset2 //
setAttr RenderWareAsset2.Scope 0;
setAttr RenderWareAsset2.Enable 1;
setAttr -type "string" RenderWareAsset2.Name "My Asset";
setAttr -type "string" RenderWareAsset2.TemplateName "Animated
Hierarchy.rwt";
setAttr -type "string" RenderWareAsset2.Selection "|pSphere1";
```

The following table lists the attributes that can be set on a RenderWareAsset node.

NAME	TYPE	DESCRIPTION
Scope	short	Indicates whether this node is a standard asset or the special scene node used for storing global settings. 0 = asset, 1 = scene
Name	string	The name of this asset. This is the name displayed in the asset manager interface and is used to construct the export file name when exporting to legacy files.

Enable	short	Controls whether this asset is enabled. Disabled assets will be greyed-out in the asset manager interface and will not be exported.
TemplateName	string	The file name of the template used to export this asset. This should not include a path.
ExportPath	string	For asset nodes, this controls the export path used when exporting to legacy files. For the scene node, this controls the global export path.
Selection	string	The selection this asset refers to. The Maya long path convention should be used, e.g. "|pSphere1|pSphereShape1;|pTorus1|pTorusShape1"
Parent	string	If this is a child asset, the name of the parent asset, empty string otherwise. e.g. "RenderWareAsset2"
Children	string	The names of any children of this asset, e.g. "RenderWareAsset3;RenderWareAsset4"
Options	string	Any per-asset options for this asset, e.g. "BOOL:Visualize Partitions=1"
GUID	string	The unique identifier for this asset, e.g. "a299f913-42f9-43bd-bd5c-cd9fe78c4762"
CustomExportFile	string	The custom export file for this asset, e.g. "RenderWareAsset2.CustomExportFile "c:\\exportpath\\filename" ". See the Export Location section of the Maya Reference Guide for further information.

4.3 Updating Template Files

It is possible to script the update of template files taken from a previous version of the RenderWare exporter and update these files to contain any updated/new options and remove old options. The main function for this is:

```
RwUpdateTemplateFile (file)
```

where `file` is the template file(s) to process. An example script is:

```
--update some template files
RwUpdateTemplateFile ("d:\\rwt\\~PS2.rwt");
```

This script would update the `~ps2.rwt` template file in the `rwt` folder.

4.4 Blind Data

Blind data is typically used to tag areas of scenes with user-defined attributes. Export of this data enables your application to query this data and use it appropriately (e.g. if the character steps on a certain polygon area, have the floor fall away).

The Maya exporter uses the RenderWare Graphics plug-in, RpUserData, to store blind data. This plug-in allows arrays of user-defined data to be stored with various RenderWare Graphics objects (world sectors, geometries, frames, lights, cameras, materials and textures).

The option *Export User Data* in the Animated Hierarchy template exports user data for any vertices, faces or objects which have a blind data type set. The generated RenderWare Graphics object (geometry or frame) will have a corresponding RpUserData array.

Similarly, for *Static World* exports, if any vertices or faces in the scene have a blind data type set on them, all generated RenderWare Graphics world sectors will have a corresponding RpUserData array set on them. Those faces and vertices that do not have the blind data type set will have a default value in the RpUserData array.

The size of the array will correspond to the number of faces or vertices in the geometry or world sector. In case of object blind data, the size will be one, since you can only apply a single element of each blind data attribute to an object.

If the blind data type contains multiple attributes, a separate RpUserData array will be generated for each. The name of the RpUserData array will be generated from the blind data type name and the Long Name of the attribute as follows: *Type name.Long Name.*

The Maya exporter now exports blind data, on lights, cameras, shaders and textures, to RpUserData. Since the Maya blind data editor does not support all these object types, the data for some must be added through MEL script.

The format of the RpUserData array will depend on the data type of the blind data attribute. The table below illustrates the conversion.

MAYA BLIND DATA ATTR TYPE	RENDERWARE GRAPHICS USER DATA TYPE	DEFAULT VALUE
Double	rpREALUSERDATA	0.0
Int	rpINTUSERDATA	0
Hex	rpINTUSERDATA	0
Boolean	rpINTUSERDATA	0
String	rpSTRINGUSERDATA	NULL
Binary	rpSTRINGUSERDATA	NULL

Conversion from Maya blind data types to RenderWare Graphics UserData types

The *Blind Data Editor* adds blind data to meshes, patches, material and texture nodes. The *Blind Data Editor* uses the provided blind data MEL script command, polyBlindData, to add blind data to nodes. When the *Blind Data Editor* adds blind data to transformation nodes, it hands polyBlindData the extra argument *–shape*. This means that the blind data is added to the shape used by the transformation node, not the transformation node. Blind data can be applied to transformation nodes using the MEL script command polyBlindData.

Adding Blind Data

Here is an example that uses polyBlindData to add blind data to a selected node:

```
polyBlindData –id 6 -associationType "object" -longDataName
"AttributeName" -stringData "my string";
```

In the example above, -associationType is set to "object" and the -shape argument is not used. Blind data can be added to any type of node selected using these arguments.

The above only works when the blind data template 6 has a string attribute called "AttributeName".

Creating Blind Data Templates from MEL Script

You can create blind data templates from MEL script as well as from the *Blind Data Editor*. This is done with the MEL script command blindDataType.

Here is an example that uses blindDataType to create the blind data template used by the above polyBlindData example:

```
blindDataType -id 6 -dataType "string" –longDataName "AttributeName"
-shortDataName "AttrName";
```

Processing Blind Data Attributes

One other feature that has been implemented is a better way to store binary blind data. Maya stores binary data as strings. We've added code to the exporter to process blind data attributes of type Int32Array. To do this, you must do what polyBlindData would normally do.

The blind data MEL script command, polyBlindData, adds a "compound" attribute, with the name of the blind data template being used, to the selected node. This attribute will be the parent of the blind data attributes.

In terms of the MEL script above this means:

Create the parent template attribute:

```
addAttr -ln BlindData n -nc c -at compound;
```

Where **n** is the ID number of an existing blind data template and **c** is the number of attributes in template **n**. In this case **c** will be 1.

Add an attribute child:

```
addAttr -ln x -dt Int32Array -p BlindDatan;
```

Where **x** is the name of an attribute in template **n**, and **t** is the type of the attribute. The type of the attribute added must match the type of the attribute in the blind data template.

Set the value of the child:

```
setAttr s.x -typ Int32Array z xxx yyy ...;
```

where **s** is the selected object, **z** is the size of the array and **"xxx yyy ..."** are the values to set. This would be exported as an `RpUserData int` array.

5. MAXScript

5.1 Creating Assets

If you need to create RenderWare Asset nodes from your own script, or change per asset and per scene settings, this example will try to give you all the necessary information you need.

The RenderWare scene settings are stored in the $rwDefaultSceneNode node:

```
-- Set the export folder.
$rwDefaultSceneNode.export_path = "c:\\"


-- Set the output file name. Extension will depend on the template
-- options for example .rws, .dff etc. Btw here you have to use
-- .baseObject since INode has built in property called "name" so it
-- won't propagate it to the object like in the "export_path" case.
$rwDefaultSceneNode.baseObject.name = "test"
```

Since all the assets are automatically managed by the exporter, they will be deleted and recreated on each export. So, unless you turn Customized mode on, you'll lose the assets you've created through the script. This should be the first step you do before changing/creating any assets.

```
-- Turn the Customized mode on
RenderWare_Exporter.AssetMgr.SetCustomizedAssets(true)


-- Note: For other functions in the RenderWare_Exporter interface
-- you can call:
showInterfaces RenderWare_Exporter
```

After you select the object you want to create the assets for, you should call `CreateAsset`. This will trigger automatic code creation, which will analyze the selected node and set up the template for it.

```
select $Teapot01
myAsset = RenderWare_Exporter.AssetMgr.CreateAsset()
```

The type of the asset and all the options are now defined by an asset template file. If you want to change it, use this:

```
myAsset.template_name = "My Static World.rwt"
```

You can list all the properties of the assets:

```
showProperties myAsset
  .name (Spin) : string
  .scope (Spin) : integer
  .num_refs (Spin) : integer
  .export_path (Spin) : string
  .template_name (Spin) : string
  .options (Spin) : string array
  .enable (Spin) : boolean
  .guid (Spin) : string
  .custom_export_file (Spin) : string
```

NAME	DESCRIPTION
name	The name of this asset. This is the name displayed in the asset manager interface and is used to construct the export file name when exporting to legacy files.
scope	Indicates whether this node is a standard asset or the special scene node used for storing global settings. 0 = asset, 1 = scene.
num_refs	For internal use only.
export_path	For asset nodes, this controls the export path used when exporting to legacy files. For the scene node, this controls the global export path.
template_name	The file name of the template used to export this asset. This should not include a path.
options	Any per-asset options for this asset, e.g. "BOOL:Visualize Partitions=1".
enable	Controls whether this asset is enabled. Disabled assets will be greyed-out in the asset manager interface and will not be exported.
guid	The unique identifier for this asset, e.g. "a299f913-42f9-43bd-bd5c-cd9fe78c4762".
custom_export_file	The custom export file for this asset, e.g. "RenderWareAsset01.custom_export_file = “c:\\exportpath\\filename”. See the Export Location section of the 3ds max Reference Guide for further information.

The asset hierarchy information displayed in the Asset Manager control is representing in the 3ds max scene using the standard parenting mechanisms. You can be edit these relationships using the normal MAXScript parenting commands:

```
$myAsset.parent = $parentAsset
```

Note that CreateAsset returns an INode and this interface overrides name, to return the name of the node in the scene. To access the export name of a node created using CreateAsset, use the baseObject:

```
myAsset = RenderWare_Exporter.AssetMgr.CreateAsset()
myAsset.baseObject.name = "new asset"
```

5.2 Batch Exports

It is possible to script batch files using MAXScript. The main functions are:

```
RwBatchExport("AssetName")
RwBatchExportAll()
```

where AssetName is the name of the asset you wish to export. So, a typical script might be:

```
--export.ms

--export artwork in several formats
loadmaxfile "export1.max"
max views redraw
RwBatchExport "Asset01"  "Asset02" "Asset03"
loadmaxfile "export2.max"
RwBatchExportAll()

--exit 3ds max
quitMax #noPrompt
```

To make the script run as a batch file, create the following as a text file named "export.bat":

```
REM export.bat
set MAXEXE=c:\3dsmax4\3dsmax.exe
%MAXEXE% -U MAXScript export.ms
```

where c:\3dsmax4\3dsmax.exe is the path to your 3ds max application.

Run the batch file to start 3ds max and run the export.ms script.

RwBatchExportAll exports all assets and will iterate across all the assets, exporting each in turn.

Use RwLoadSelection("Asset01") and RwSaveSelection("Asset01") to save and load the selection within an asset.

The "Selection" scripts only take one argument.

5.3 Updating Template Files

It is possible to script the update of template files taken from a previous version of the RenderWare exporter, and update these files to contain any updated/new options and remove old options. The main function for this is:

```
RwUpdateTemplateFiles(filelist)
```

where `filelist` is the list of template files to process. An example script might be:

```
--update some template files
RwUpdateTemplateFiles "c:\\rwt\\ps2.rwt" "c:\\rwt\\gcn.rwt"
```

This script would update the `ps2.rwt` and `gcn.rwt` template files in the `rwt` folder.

5.4 Adding Custom Data

To add a custom attribute to something in MAXScript, first you must create a custom attribute to add. This custom attribute must be in a format that the exporter will recognize. For the exporter to recognize your attribute, you must have two strings at the beginning of your attribute's parameters. The first should be called TargetApp, and should be set as "RenderWare", and the second should be called DataType, and should be set to any of following:

"Object", "Transform", "Face", "Vertex", "Light", "Camera", "Material" or "Texture"

Which of these to use will become apparent later, though most are self apparent.

Here is an example of a custom attribute that will be used for putting user data on a geometry, frame, material, texture, light or camera.

```
myattrib = attributes mydata
(
   parameters main rollout:paramas
   (
       TargetApp type:#string ui:rwlab
       DataType type:#string ui:dattype

--use one or more of the following types
       myint type:#integer ui:myintslider
       myfloat type:#float ui:myfloatslider
       mystring type:#string ui:teststr
   )
   rollout paramas "UserData"
   (
```

```
        dropdownlist rwlab "TargetApp" items:#("RenderWare")
        dropdownlist dattype "Data Type"
items:#("Object","Transform","Light","Camera","Material","Texture")

        spinner myintslider "My Int" type:#integer
        spinner myfloatslider "My Float" type:#float

        dropdownlist teststr "Test String" items:#("Test1","Test2")
    )
)
```

Attaching Custom Data to an Object

This section shows you how to attach custom attributes to a geometry, frame, material, texture, light or camera.

Attaching Custom Data to a Geometry

```
-- export data for an object
custAttributes.add $Box01 myattrib
$Box01.mydata.TargetApp = "RenderWare"
$Box01.mydata.DataType = "Object"
$Box01.mydata.myint = 1
$Box01.mydata.myfloat = 2.5
$Box01.mydata.mystring = "Test"
```

Attaching Custom Data to a Frame

```
-- export data for a frame
custAttributes.add $Box02 myattrib
$Box02.mydata.TargetApp = "RenderWare"
$Box02.mydata.DataType = "Transform"
$Box02.mydata.myint = 3
$Box02.mydata.myfloat = 4
$Box02.mydata.mystring = "Test2"
```

Attaching Custom Data to a Material

```
-- export data for a material
mat = $Box02.material
custAttributes.add mat myattrib
```

```
mat.mydata.TargetApp = "RenderWare"
mat.mydata.DataType = "Material"
mat.mydata.myint = 3
mat.mydata.myfloat = 4
mat.mydata.mystring = "Test2"
```

Attaching Custom Data to a Texture

```
-- export data for a texture (assuring RwMaterial type material)
basetexture = mat.defmtl_texmap_texture
custAttributes.add basetexture myattrib
basetexture.mydata.TargetApp = "RenderWare"
basetexture.mydata.DataType = "Texture"
basetexture.mydata.myint = 3
basetexture.mydata.myfloat = 4
basetexture.mydata.mystring = "Test2"
```

Attaching Custom Data to a Light

```
-- export data for a light
custAttributes.add $Light01 myattrib
$Light01.mydata.TargetApp = "RenderWare"
$Light01.mydata.DataType = "Light"
$Light01.mydata.myint = 3
$Light01.mydata.myfloat = 4
$Light01.mydata.mystring = "Test2"
```

Attaching Custom Data to a Camera

```
-- export data for a camera
custAttributes.add $Camera01 myattrib
$Camera01.mydata.TargetApp = "RenderWare"
$Camera01.mydata.DataType = "Camera"
$Camera01.mydata.myint = 3
$Camera01.mydata.myfloat = 4
$Camera01.mydata.mystring = "Test2"
```

Attaching Custom Data per Vertex or per Polygon on a Geometry

Per vertex or per polygon custom data is different to other customer attributes. They work by having a map of values. The map must have the same number of entries as the subject being mapped (vertices or polygons).

The following is a structure for attaching per polygon or per vertex custom attributes:

```
myattrib2 = attributes mydata2
(
   parameters main rollout:paramas
   (
       TargetApp type:#string ui:rwlab
       DataType type:#string ui:dattype

-- more than one tab type can be used: floatTab, stringTab, intTab
       myint type:#intTab tabSizeVariable:true ui:intdata
   )
   rollout paramas "UserData"
    (
       dropdownlist rwlab "TargetApp" items:#("RenderWare")
       dropdownlist dattype "Data Type" items:#("Face","Vertex")
       dropdownlist intdata "My Int Map" items:#("Set in Script")
    )
)
```

Filling in Custom Data for a Geometry

This section shows you how to use custom attributes to create user data for selected polygons/vertices.

Filling in Custom Data per Polygon on a Geometry

```
-- Add per polygons custom attributes to selected polygons

custAttributes.add $testMesh myattrib2

$testMesh.mydata2.TargetApp = "RenderWare"
$testMesh.mydata2.DataType = "Face"

-- there should be an entry for each polygon
-- the code below adds 0 for all polygons
for i=1 to $testMesh.numfaces do
(
    $testMesh.mydata2.myint[i] = 0
)

-- assigns values to all selected polygons
for i=1 to $testMesh.selectedFaces.count do
```

```
(
    n = $testMesh.selectedFaces[i].index
    if n!=0 then
    (
        $testMesh.mydata2.myint[n] = 10
    )
)
```

Filling in Custom Data per Vertex on a Geometry

```
-- Add per vertex custom attributes to selected vertices
custAttributes.add $testMesh myattrib2

$testMesh.mydata2.TargetApp = "RenderWare"
$testMesh.mydata2.DataType = "Vertex"

-- there should be an entry for each vertex
-- adds 0 for all vertices
for i=1 to $testMesh.numverts do
(
    $testMesh.mydata2.myint[i] = 0
)
-- assigns values to all selected vertices
for i=1 to $testMesh.selectedVerts.count do
(
    n = $testMesh.selectedVerts[i].index
    if n!=0 then
    (
        $testMesh.mydata2.myint[n] = 10
    )
)
```

Filling in Custom Data per Polygon on a Geometry

```
-- Add per polygons custom attributes to selected polygons

custAttributes.add $testMesh myattrib2

$testMesh.mydata2.TargetApp = "RenderWare"
$testMesh.mydata2.DataType = "Face"
```

```
-- there should be an entry for each polygon
-- the code below adds 0 for all polygons
for i=1 to $testMesh.numfaces do
(
    $testMesh.mydata2.myint[i] = 0
)

-- assigns values to all selected polygons
for i=1 to $testMesh.selectedFaces.count do
(
    n = $testMesh.selectedFaces[i].index
    if n!=0 then
    (
        $testMesh.mydata2.myint[n] = 10
    )
)
```

Filling in Custom Data per Vertex on a Geometry

```
-- Add per vertex custom attributes to selected vertices
custAttributes.add $testMesh myattrib2

$testMesh.mydata2.TargetApp = "RenderWare"
$testMesh.mydata2.DataType = "Vertex"

-- there should be an entry for each vertex
-- adds 0 for all vertices
for i=1 to $testMesh.numverts do
(
    $testMesh.mydata2.myint[i] = 0
)
-- assigns values to all selected vertices
for i=1 to $testMesh.selectedVerts.count do
(
    n = $testMesh.selectedVerts[i].index
    if n!=0 then
    (
        $testMesh.mydata2.myint[n] = 10
    )
)
```

Practical Custom Data Example

As an example, the following script adds a custom rollout and attributes to a texture to control PS2 K and L values. These texture settings are supported by the RenderWare Graphics exporter but, due to a 3ds max limitation, are unavailable from the RenderWare Graphics Object Settings rollout.

```
myattrib = attributes mydata
(
   parameters main rollout:paramas
   (
       TargetApp type:#string
       DataType type:#string

       RwK type:#float default:1 ui:RwKspinner
       RwL type:#integer default:0 ui:RwLspinner
   )
   rollout paramas "PS2 K&L"
   (
       spinner RwKspinner "K" range:[-1000, 1000, 1] type:#float
       spinner RwLspinner "L" range:[0, 3, 0] type:#integer
   )
)

-- get texture
mat = $Box01.material
basetexture = mat.defmtl_texmap_texture

-- add custom attributes
custAttributes.add basetexture myattrib

-- set attribute values
basetexture.mydata.TargetApp = "RenderWare"
basetexture.mydata.DataType = "Texture"
basetexture.mydata.RwK = -10.0
basetexture.mydata.RwL = 1
```

More Custom Data Information

You can find out more about custom attributes in section "Scripted Custom Attributes" of the 3ds max 4.2 MAXScript Reference.

Index