In this Topic

Overview

This document provides information on the Resource Compiler (often referred as "RC"). For most users, the RC is a tool that runs in the background.

However, it is often important to understand the reasons for why the RC exists, as well as the essential role it plays in game development with CRYENGINE.

CRYENGINE has the concept of source and target resources. Source resources are stored in a lossless format (without lossy compression) and can have asset-specific metadata, for example, the compression quality that should be used. These source assets are not used directly by the engine, but must be compiled to a platform-optimized format. In this process, the source data gets pre-processed and often compressed to reduce the memory requirements and speed up the loading process in the game. The target format can be different on individual platforms. PC and consoles have a different endianness and support different texture compression schemes. The source to target transformation (compilation) is done by the Resource Compiler (RC), which you can find in the RC sub-folder of the Tools folder.

One of the primary purposes of the RC is to compress textures by using the *ResourceCompilerImage* module. All source textures are stored in the lossless TIF format and get converted to the DDS format. The Export Textures with CryTIF - Photoshop page can be used to export a TIF file from Photoshop and open a dialog box where the meta data, like the desired compression scheme, can be chosen.

The Resource Compiler is invoked from the following tools:

- Sandbox when using LiveCreate or when loading TIF textures for which the DDS is missing.
- The CryTIFPlugin (Photoshop 6 and higher plugin).
- The CryENGINE Renderer (the r_RC_AutoInvoke console variable is set to 0 in full screen).
- The PolybumpPlugin (3ds Max plugin).
- The CryMaxExporter (3ds Max plugin).
- The CryMayaExporter (Maya plugin).

Presets

The Resource Compiler supports different presets that specifies the options that should be applied when compiling an image asset. When starting the RC, it loads the main settings and presets from the main configuration file (rc.ini).

A preset file for image compilation (TIF to DDS) looks like this:

```
; diffuse colour textures without alpha channel
[Albedo]
pixelformat=BC1
pixelformat:es3=ETC2
rgbweights=ciexyz
powof2=1
mipmaps=1
colorspace=sRGB,auto
;discardalpha=1
filemasks=*_diff*

; diffuse colour textures with alpha channel for alphablend
[AlbedoWithOpacity]
pixelformat=BC7t
pixelformat:es3=ETC2A
rgbweights=ciexyz
powof2=1
mipmaps=1
colorspace=sRGB,auto
filemasks=*_diff*
```

Calling the RC from the Command Line

The following examples allows opening the user dialog box so that the user can modify the parameters stored in the TIF file.

Note that when using the dialog box, it is possible to generate the .dds file by using *Generate DDS* button.

```
Tools\rc\rc c:\temp\test.tif /userdialog
```

Multiple directories and file types can be processed recursively with the . or *. extension:

```
Tools\rc\rc c:\*.tif /refresh
```

Same command, but with skipping compilation of .tif files that already have .dds files:

```
Tools\rc\rc c:\*.tif
```

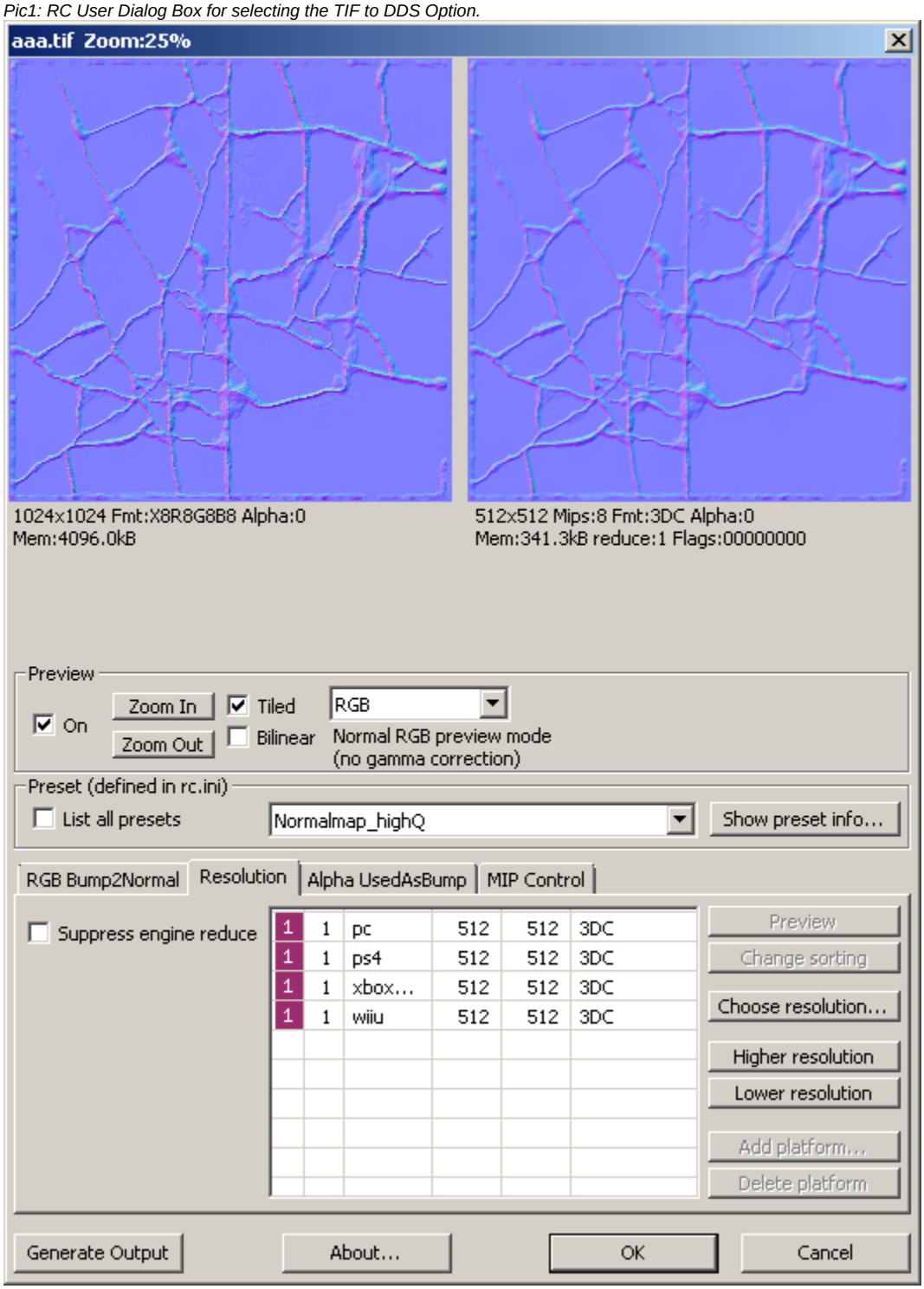
Use quotes to compile assets with special characters (for example, spaces) in the path:

```
Tools\rc\rc "c:\data folder\*.tif"
```

The Resource Compiler Dialog

The following image shows the user interface that can be invoked with /userdialog or by using the Export Textures with CryTIF - Photoshop for Photoshop. The Export Textures with CryTIF - Photoshop documentation has more information about using a source texture (.tif) in CRYENGINE.

Pic1: RC User Dialog Box for selecting the TIF to DDS Option.



Note

The RC Dialog can only be invoked when processing SRF and TIF files.

Processing a Collada File from the Command Line

This is optional, you may use it if you want to use Collada as intermediate format. Otherwise collada files are automatically converted during export process.

To process your Collada file with RC from command line, you have to perform the following steps:

1. Process the Collada file directly with RC from the command line to output an uncompressed **.caf* file.

```
rc.exe "fullpath"\walk.dae
```

2. Process the **.i_caf* file that you got from Step 1 again with RC from the command line specifying the sub-folder name that contains *SkeletonList.xml* and DBATable.json to output a compressed **.caf* file.

From engine version 5.6 onwards, the skeleton list is not used anymore, which means that you can ignore any references to it on this page.

```
cd SDKFolder
rc.exe path\to\walk.i_caf /animConfigFolder=Animations /sourceroot=GameSDK_Source /targetroot=GameSDK /SkipDBA
/refresh
```

To double check if the files were updated, refresh the Windows Explorer so that you can see if the file is a different size (re-processed), or check the time-stamp.

Also be sure to specify the following arguments:

- /skipdba - skip build dba.
- /refresh - force recompilation of resources with up to date timestamp.

Creating a Dump Log from Compiled Assets

The Resource Compiler can be used to generate a dump of the CGF. The syntax to perform this operation is mentioned below:

```
rc.exe /refresh /debugdump filename.cgf
```

This will create a new text file with the **.dump* extension with information on the data contained in the CGF file.

Since the CHR and CGA are also using the file structure of the CGF, it's possible to make a dump on these file as well. You just need to use the overwritee xtension command-line parameter:

```
rc.exe /refresh /debugdump /overwriteextension=cgf filename.chr
```

Command Line Arguments

The following are some more examples of how you can run the Resource Compiler from the command line:

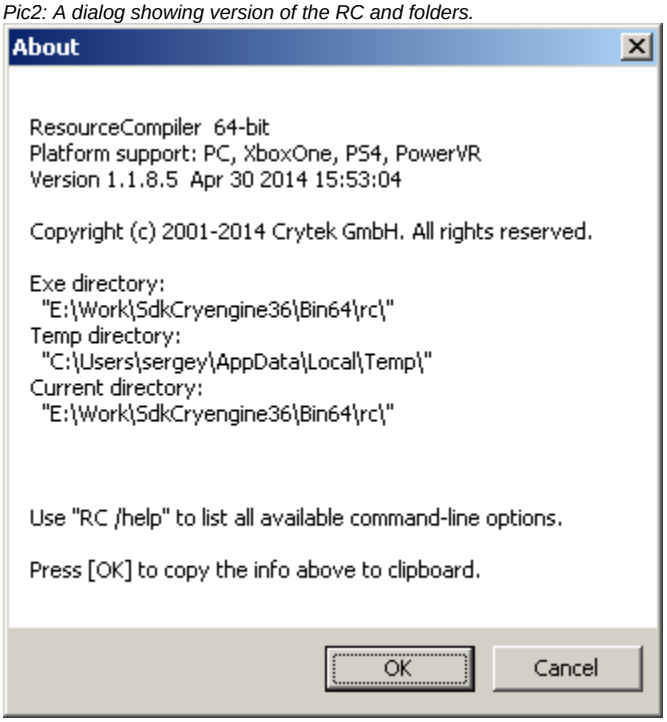
```
rc myassset.tif /userdialog

Tools/rc/rc.exe c:\temp\*.tif /refresh /wait
```

The following is a list of useful command line arguments, specified as key-value pairs. Further details and a complete list of keys can be retrieved by using rc.exe with the /help argument (refer to the sample file, RCkeys.txt).

- /recursive = 0/1.
- /refresh to deactivate the incremental compilation (ignores the file date).
- /wait to wait for a key pressed on finish.
- /wait=4 pop up a message box and wait for a button pressed on start and on finish.
- /statistics = 1 to generate statistic files: *rc_stats_filedependencies.log*, *rc_stats_materialdependencies.log*, *rc_stats_presetusage.log*, *rc_stats_files izes_xls.log* (in a format that can be easily read from Excel).
- /logfiles = 1 to suppress generation of log file *rc_log.log (rc_log_warnings.log and rc_log_errors.log still generated).*
- /logtime=0 to suppress logging time.
- /WX to treat warnings as errors.
- /quiet to suppress all the printouts.
- /targetroot = to define the destination folder. note: this folder and its subtrees will be excluded from the source files scanning process.
- /overwriteextension = extension. Choose convertor using the specified extension.
- /userdialog to show the user dialog box for the ResourceCompilerImage.
- /cachefolder=path to enable the RC to cache compiled files. This allows to use the cached files instead of recompiling them again every time.
- /forceVCloth: Enables creation of VCloth chunk. It overwrites the MAKE_VCLOTH export setting of the CGF.
  This allows the CGF file to contain the information regarding the pre-processing of the cloth-mesh by storing the metadata in the CGF file. And prevents the VCloth animation system to pre-process the cloth-mesh of a SKIN file at runtime which avoids performance drops when spawning a group of characters using the same cloth-mesh. For more information on VCloth animation system, please refer to Tutorial - VCloth 2.0 Setup.

If you run rc.exe without any argument, then RC pops up a dialog showing version of the RC and folders:

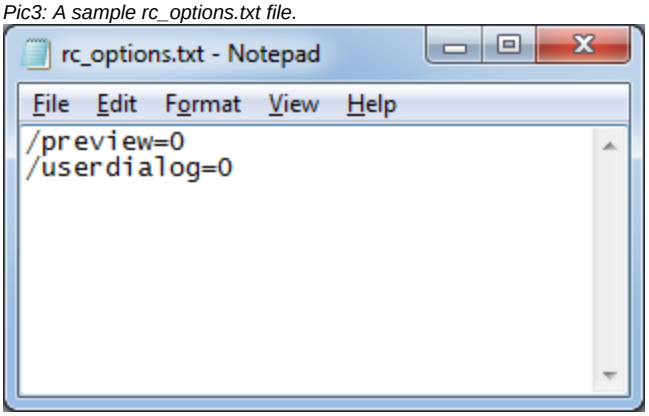
Pic2: A dialog showing version of the RC and folders.



RC_Options.txt

If you're looking for a more permanent/frequent-use solution, you can create a file called rc_options.txt inside the RC folder that is being used. This can be used to apply specific settings every time the RC is invoked. Below is an example:

Pic3: A sample rc_options.txt file.



- *preview=0* will disable the image output preview in the CryTIF dialog. This allows working with high resolution images much faster as the preview image doesn't need to be processed.
- *userdialog=0* will disable the CryTIF dialog completely. This can be useful if you're working on a particular set of files that you know have been set correctly (their presets, resolutions, etc) and you want to make frequent adjustments without being bothered by the CryTIF dialog pop-up every time.

Any parameter can be used in this file to automate the RC to suit your needs. To get a full list of available parameters, open the RC folder in the command prompt and type rc /help.

Automatic Adjustments

If there is an alpha channel, the chosen pixel format is adjusted so that the alpha channel can be used.

For example, X8R8G8B8 becomes A8R8G8B8. If there is no alpha channel the format is adjusted as well. The system tries to be intelligent by using a similar texture format quality.

That means you don't need a new preset to support textures with alpha channel. Below the right preview window, you can see the used texture format (CryTIFPlugin).

Output in the Extended DDS Format

The DDS file format includes additional meta data (a proprietary extension mechanism) as mentioned below:

```
void CImageObjectBase::AppendExtendedData( FILE *out )
{
  uint32 dwStartMagicWord = swizzle32('CExt');                 // Crytek Extended -start
  fwrite(&dwStartMagicWord,4,1,out);

  uint32 dwChunkName = swizzle32('AvgC');                      // Chunk AverageColor
  fwrite(&dwChunkName,4,1,out);
  uint32 dwChunkSize = 4;
  fwrite(&dwChunkSize,4,1,out);
  fwrite(&m_colAverageARGB,4,1,out);

  uint32 dwEndMagicWord = swizzle32('CEnd');                  // Crytek Extended - end
  fwrite(&dwEndMagicWord,4,1,out);
}
```

A sample of the new dds file (the build now does for all DDS files that are created from TIF) is provided below:

```
005555a0h: 21 22 22 23 25 25 25 27 5B 5C 5C 5F 5A 5A 5B 5E ; !""#%%%'[\\_ZZ[^
005555b0h: 58 58 58 5A 60 60 61 63 70 70 71 74 6D 6D 6E 70 ; XXXZ``acppqtmmnp
005555c0h: 2B 2C 2C 2E 2D 2D 2E 30 39 39 3A 3B 58 59 59 5C ; +,,.-.-.099:;XYY\
005555d0h: 33 34 34 36 43 45 78 74 41 76 67 43 04 00 00 00 ; 3446CExtAvgC....
005555e0h: 33 34 34 36 43 45 6E 64                         ; 3446CEnd
```

Common Animation Related Usage

Exported *.*i_caf* files contain animation in uncompressed format. To be loaded in the engine, they first need to be compressed into *.*caf* files. For local use during production, this is normally done by automatic invocation of RC in Sandbox.

To perform a full build on your local PC, you can use following command line:

```
cd C:\CryENGINE
Tools\rc\rc.exe *.i_caf /animConfigFolder=Animations /p=PC /sourceroot=GameSDK_Source /targetroot=C:
\Build\GameSDK /refresh
```

- /animConfigFolder is used to specify location of SkeletonList.xml and DbaTable.json within /sourceroot.
- The /sourceroot and /targetroot options should not be pointing to the same location.
- The /p options specifies the target platform. "PC", "X360" and "PS3" are possible values. If not specified, it defaults to the "PC".
- /refresh options is used to ensure that target files will be overwritten if they exist and modification dates match source files.
- You can find result of the log of the invocation in Tools\rc\rc_log*.log files.

Animation Compression Details

An animation source file is processed by the RC to output a compressed CAF file used by CRYENGINE. The compression involves the following steps:

Remove data in channels with no moves (same to the bind pose within the whole duration)

Remove some key-frames based on error thresholds

- Key-frames which can be interpolated with the given error threshold are removed.
- Each joint can have a custom error threshold.

Quantize quaternions

- A quaternion consists of 4 floats (128bit) normally. But for the normalized one, we can keep just 3 floats (96bit).
- There are three kinds of schemes for the quantization:
  - 128bit (no compression),
  - 48bit (15bit+15bit+15bit, and rest bits for indexing the omitted element),
  - 64bit (20bit+21bit+21bit, and rest bits for indexing the omitted element).
- In the default setting, the 48-bit scheme is used for most cases.
- RC determines the most effective scheme under the given error threshold (the same one given in step 2).
  - Actually, the RC applies key-frame removal and quantization for all 3 schemes within the given error threshold and chooses the one that results in the best compression(i.e. minimal size).
- You can find more details in the '*QuatQuantization.h*'.

Create animation databases (optional)

- During this multiple CAF files are combined into a single DBA file, all duplicate controllers get eliminated.