Autodesk®
Scaleform®

GFxExport2 Reference Guide

This document describes the GFxExport2 utility tool which preprocesses SWF files into an optimized GFx format for use with Scaleform.

Author: Dmitry Polenur
Version: 1.0
Last Edited: April 28, 2014

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

How to Contact Autodesk Scaleform:

Document	GFxExport2 Reference Guide
Address	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
Website	www.scaleform.com
Email	info@scaleform.com
Direct	(301) 446-3200
Fax	(301) 446-3199

Table of Contents

1 Introduction

This document describes the GFxExport2, a utility tool in Autodesk® Scaleform®. GFxExport2 is introduced in Scaleform 4.4 as an alternative to the original GFxExport and serves the same purpose.

GFxExport2 is a command line utility that processes SWF files into a format that is optimized for streamlined loading. During preprocessing, resources are stripped out and extracted into separate files. Textures can be compressed into a more efficient format. Once stripped and converted, the Flash files will have the .gfx extension and are only compatible with Scaleform. The GFx files will then rely on the external data files, which can be managed by the application's resource system.

The GFxExport2 utility tool has various options for preprocessing the SWF files into different formats which is described in detail in Section 2. When used with these different options, GFxExport2 can export font textures, compress images, load shared copy of images and so on.

The main purpose of creating GFxExport2 for 4.4 was to improve the convenience and flexibility of using external texture conversion tools and to allow users easily modify and extend GFxExport2. The original GFxExport uses command line for configuration. GFxExport2, however mostly relies on JSON configuration file. Currently the new GFxExport2 does not have all the features of the legacy GFxExport. However GFxExport is still available in 4.4 in case you decide to use the legacy version instead of GFxExport2. We plan to add some of the missing GFxExport features to GFxExport2 in the subsequent 4.4 releases. Support of custom .dll plugins for image export will be added too in future releases. As always we appreciate your feedback on the new GFxExport2.

2 Usage

The GFxExport2 tool is prebuilt for Win32 and located in the bin directory in your SDK. A typical path would be:

C:\Program Files\Scaleform\GFx SDK 4.4 \Bin\GFxExport2.exe

GFxExport2 is used in the following way:

```
GFxExport2 [swf file(s)...] [options]
```

For the filenames, wildcards are supported, such as *.swf.
To see the full list of command line options and usage instructions, execute GFxExport2.exe without any additional arguments and help output will be printed.

The example of typical usage:

```
GFxExport2 myfile.swf –i tga –cfg myconfig.cfg
```

GFxExport2 configuration is stored in the JSON configuration file.

-cfg <filename>
Specifies the configuration file. By default (without -cfg option) the GFxExport.cfg is loaded. General structure of the configuration file is following:

```
{
     "Common":
     {
            "option1": value1,
            "option2": value2,
              ...
     },
     "Configs":
     {
            "config1":
            {
                  …
            },
            "config2":
            {
                 …
            },
            …
     }
 }
```

There are two main sections – "Common" and "Configs". The common section includes general GFxExport2 settings and the "Configs" section declares image export configurations and includes configuration specific settings. In this documentation we use the following format for the option:
"option" (Type) : default_value

2.1 "Common" Section

2.1.1 File and Directory related options

"directory"(String) : ""
Set the destination directory for exported data files. If not specified, files are stored in the directory containing the SWF.

"create_subdir" (Boolean) : false
Create subdirectories for each SWF file using the SWF filename. Extracted files are placed in the corresponding subdirectories.

"compress_gfx"(Boolean) : false
Write compressed stripped .gfx file(s). In practice many developers compress multiple files generated by GFxExport2 within their own package system, using custom GFx::FileOpener to load them as necessary.

"check_date"(Boolean) : false
Do not run if SWF file is older that existing gfx.

"export_sounds"(Boolean): false
Export sounds as .wav files.

2.1.2 Packed texture options

GFxExport2 can create packed texture (atlases). Using image atlases may improve runtime performance. In order to exclude the texture from packing you can add ".nopack" to the image Linkage ID/Class Name. Also, textures that have the "wrappable" property will be excluded from packing.

"pack_images" (Boolean) : false
Create packed textures.

"max_pack_texture_resolution"(Integer):1024

Maximum atlas texture resolution

"pack_texture_scale"(String) :"p2"

"p2" - create power of 2 atlas textures
"mult4" - create multiple of 4 atlas textures
"mult128"- create multiple of 128 atas textures
"no" - no scale restrictions

2.1.3 Image scaling options

"rescale" (String) : "none"
Rescale exported images. Can be:

"none" - do not rescale,
"high" – rescale to the power of two resolution higher than the original image
"low" – rescale to the power of 2 resolution lower than the original image
"nextlow" – rescale to the power of 2 resolution one step lower than "low"option
"nearest" – rescale to the nearest power of 2 resolution
"mult4" – rescale to the nearest multiple of 4 resolution.

In most cases, it is not necessary to do rescaling for packed textures (use "pack_texture_scale" instead). Also it is not necessary use the "rescale" option if the export converter (see section 2.2) does its own rescaling.

"square_images"(Boolean) : false
Force square images with "rescale" or "pack_images" options.

2.1.4 Font options

GFxExport2 can be used to compress font vector data more efficiently and to reduce memory usage. It also can be used to pre-generate font textures (static font cache), however we generally do not recommend it except for some special cases. Please refer to our Font Configuration Overview document for more information about the Scaleform font system.

"compact_fonts"(Boolean) : false
Enable font compactor (see our Font documentation for more details).

"font_normalized_size"(Integer) : 256
Set the normalized size of compact fonts.

"merge_font_edges"(Boolean) : false
Merge edges for compact fonts.

"export_fonts" (Boolean) : false
Generate font textures (static cache). This option is not compatible with *"compact_fonts"* and generally is not recommended in most cases.

"font_texture_size" (integer): 1024
Set font texture size for *"export_fonts".*

"strip_font_shapes" (Boolean) : false
Strip font vector data from the gfx file.

2.2 "Configs" Section

This section defines image export configurations.
The command line option
-i <config>
selects the image configuration.

Image export configurations have following structure:

```
"config_name" :
{
      "converter" : "imageconverter",
      "converter-params" :
      {
            "param1" : value1,
            "param2" : value2,
      }
}
```

"converter" is a mandatory parameter, the "converter-params" section is optional.

Currently converter can be one of the following:
scaleform:tga - Truevision (Targa or TGA)
scaleform:dds - DirectDraw Surface (DDS).
scaleform:pvr - PowerVR compression.
scaleform:tool - Use external tool to compress /convert images

Support for custom converters using .dll plugins will be added in future releases.
Currently the recommended way of using custom image conversion is to use the scaleform:tool converter and your own executable.

The "Config" section may have several configurations using the same converter. For example, we may have two dds configurations DXT5 and uncompressed:

```
"dds" :
{
	"converter" : "scaleform:dds",
	"converter-params" :
	{
		"format" : 5,
		"quality" : "normal",
		"DXT1apha" : false
	}
},
"dds_uncompressed" :
{
	"converter" : "scaleform:dds",
	"converter-params" :
	{
		"format" : 0
	}
},
```

2.2.1 scaleform:tga

Saves image in TGA format. This converter currently does not have any options.

2.2.2 scaleform:tool

Uses an external executable to convert images. It can be used to convert images to platform-specific formats (for example, using orbis-image2gnf.exe for PlayStation 4, or psp2gxt.exe for PS Vita etc.).

"input_config"(String) : "tga"
Name of the configuration to be used to produce input file for the tool. This configuration must be defined in the same configuration file. For example:

```
"tga" :
{
	"converter" : "scaleform:tga"
},
```

"delete_input"(Boolean) : true
Delete input file after conversion. It must be set to false if output file for the tool is the same as input file.

"tool"(String) : "texconv.exe"
The conversion tool executable. If the executable is not in current working directory this option should include the full or relative path.

"pre_args" : "f.tga -f BC7_UNORM"
Command line arguments for the tool. There is a special symbol "f" that will be replaced with image path (without extension). This symbol may be used several times in the command line.

"ext" : ".dds"
File extension produced by the tool (including "."). This is necessary to save the correct image reference to the .gfx file.

Here is the example of a configuration for BC7 compression using Microsoft texconv.exe:

```
            "dx11":
            {
                  "converter" : "scaleform:tool",
                  "converter-params" :
                  {
                        "input_config" : "tga",
                        "delete_input" : true,
                        "tool" :
"C:/Scaleform/GFx/4.4/3rdParty/Texconv/texconv.exe",
                        "args" : "$f$.tga -ft dds -f BC7_UNORM -m 1 -pow2 -nogpu",
                        "ext" : ".dds"
                  }
            }
```

And another example that uses Sony orbis-image2gnf.exe tool to produce .gnf files:

```
            "ps4":
            {
                  "converter" : "scaleform:tool",
                  "converter-params" :
                  {
                        "input_config" : "tga",
                        "delete_input" : true,
                        "tool" : "C:/Program Files (x86)/SCE/ORBIS
SDKs/0.930/host_tools/bin/orbis-image2gnf.exe",
                        "args" : "-i $f$.tga -o $f$.gnf -f bc7unorm",
                        "ext" : ".gnf"
                  }
            }
```

2.2.3 scaleform:dds

This converter uses nvtt library to produce dds files. "converter-params" section may have following options:

"format"(Integer) : 5
DXT compression, can be 0, 1, 3, 5 for uncompressed, DXT1, DXT3 or DXT5 respectively.

"quality"(String) : "normal"
Compression quality – can be “quick”, “normal”, “production” and “highest”.

2.2.4 scaleform:pvr

This converter uses PowerVR PVRTexLib library to produce .pvr files

"format"(Integer) : 4
Can be 2 or 4 for PVR2 or PVR4 respectively.

2.3 Command line options

-help
Print command line options. In the output you may see some options not described in this documentation. These options are for internal use/debugging.

-i <config_name>
Select image export configuration (see previous section for details).

-cfg <config_file_name>
Load configuration file. GFxExport.cfg will be loaded by default.
Additional options that produce output text files:

-list
Export font list and textfield/font map (.fnt-file).

-fntlst
Export font list and textfield/font map (.fnt-file).

-det
Export list of unique default values of dynamic/input textfields to the text file with .det extensions. Text file will be saved in UTF-8 encoding. Default text field values may be useful for localization.

-ddt
The same as -det option except it does not save values of input textfields. The output files will have .ddt extension.

-fstree
Export list of fscommands as a tree (.fst-file)

-fslist
Export list of fsommands as a sorted list (.fsl-file)

-fsparams
Save parameters of fscommands. This option works with -fstree and -fslist options and will save parameters of fscommands to .fst or .fsl file. However it will work only for hardcoded parameters.