

# GL2PS: an OpenGL to PostScript printing library

Christophe Geuzaine

Version 0.9.2, 4 July 2003

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Usage</b>	<b>2</b>
2.1	gl2psBeginPage and gl2psEndPage . . . . .	2
2.2	gl2psText . . . . .	4
2.3	gl2psDrawPixels . . . . .	5
2.4	gl2psEnable and gl2psDisable . . . . .	6
2.5	gl2psPointSize and gl2psLineWidth . . . . .	6
2.6	gl2psBeginViewport and gl2psEndViewport . . . . .	7
<b>3</b>	<b>Example</b>	<b>8</b>
<b>4</b>	<b>Limitations</b>	<b>9</b>
<b>5</b>	<b>Contributors</b>	<b>9</b>
<b>6</b>	<b>Links</b>	<b>9</b>
<b>7</b>	<b>Versions</b>	<b>10</b>

## 1 Introduction

GL2PS is a library for creating high quality vector output (primarily PostScript) from any OpenGL application. The main difference between GL2PS and other similar libraries (see section 6) is the use of sorting algorithms capable of handling intersecting and stretched polygons, as well as non manifold objects.

The library, written in C, is released under the GNU Library General Public License (see <http://www.gnu.org/licenses/lgpl.html> for more details), and is available at <http://www.geuz.org/gl2ps/>. Any corrections, questions or suggestions should be e-mailed to [geuz@geuz.org](mailto:geuz@geuz.org).

The interface consists of ten functions, all beginning with the prefix `gl2ps`. All the data structures and the symbolic constants peculiar to GL2PS begin with GL2PS.

## 2 Usage

### 2.1 `gl2psBeginPage` and `gl2psEndPage`

#### 2.1.1 Specification

```
GLint gl2psBeginPage( const char *title, const char *producer,
                     GLint viewport[4],
                     GLint format, GLint sort, GLint options,
                     GLint colormode, GLint colorsize,
                     GL2PSrgba *colortable,
                     GLint nr, GLint ng, GLint nb,
                     GLint buffersize, FILE *stream,
                     const char *filename )
```

```
GLint gl2psEndPage( void )
```

#### 2.1.2 Description and arguments

`gl2psBeginPage` and `gl2psEndPage` delimit the OpenGL commands that will be caught in the feedback buffer (see section 4) and output to `stream`. The arguments given to `gl2psBeginPage` determine the way primitives are handled:

**title** Specifies the plot title. For PostScript output, this string is placed in the `%%Title` field.

**producer** Specifies the plot producer. For PostScript output, this string is placed in the `%%For` field.

**viewport** Specifies the plot viewport. The viewport can for example be obtained with a call to `glGetIntegerv(GL_VIEWPORT,viewport)`. This argument is ignored if the `GL2PS_USE_CURRENT_VIEWPORT` option is set.

**format** Specifies the output format, chosen among:

`GL2PS_PS` The output stream will be a PostScript file.

`GL2PS_EPS` The output stream will be an Encapsulated PostScript file.

`GL2PS_TEX` The output stream will be a  $\text{\LaTeX}$  file, containing only the text strings of the plot (cf. section 2.2), as well as an `\includegraphics` command permitting to include a graphic file having the same basename as `filename`.<sup>1</sup>

---

<sup>1</sup>The two steps to generate a  $\text{\LaTeX}$  plot with GL2PS are thus:

**sort** Specifies the sorting algorithm, chosen among:

**GL2PS\_NO\_SORT** The primitives are not sorted, and are output in **stream** in the order they appear in the feedback buffer.

**GL2PS\_SIMPLE\_SORT** The primitives are sorted according to their barycenter. This can be sufficient for simple scenes.

**GL2PS\_BSP\_SORT** The primitives are inserted in a Binary Space Partition (BSP) tree. The tree is then traversed back to front in a painter-like algorithm. This should be used for complex three-dimensional scenes, but keep in mind that the BSP tree algorithm is quite memory hungry...

**options** Sets global plot options, chosen among (multiple options can be combined with the bitwise inclusive OR symbol |):

**GL2PS\_NONE** No option.

**GL2PS\_DRAW\_BACKGROUND** The background frame is drawn.

**GL2PS\_SIMPLE\_LINE\_OFFSET** A small offset is added in the z-buffer to all the lines in the plot. This is a simplified version of the **GL2PS\_POLYGON\_OFFSET\_FILL** functionality (cf. section 2.4), putting all the lines of the rendered image slightly in front of their actual position. This thus performs a simple anti-aliasing solution, e.g. for finite element like meshes.

**GL2PS\_SILENT** All the messages written by GL2PS on the error stream are suppressed.

**GL2PS\_BEST\_ROOT** The construction of the BSP tree is optimized by choosing the root primitives leading to the minimum number of splits.

**GL2PS\_NO\_TEXT** All the text strings are suppressed from output. This is useful to produce the image part of a L<sup>A</sup>T<sub>E</sub>X output.

**GL2PS\_NO\_PIXMAP** All the pixmaps are suppressed from output.

**GL2PS\_LANDSCAPE** Landscape orientation instead of portrait.

**GL2PS\_NO\_PS3\_SHADING** No use is made of the **shfill** PostScript level 3 operator (which can lead to problems when converting PostScript files to PDF files; see also options **nr**, **ng**, **nb** below).

**GL2PS\_OCCLUSION\_CULL** All the hidden polygons are removed from the output, thus substantially reducing the size of the output file.

- 
1. generate the PostScript file (e.g. **file.ps**) with no text strings, using the **GL2PS\_PS** or **GL2PS\_EPS** format combined with the **GL2PS\_NO\_TEXT** option;
  2. generate the L<sup>A</sup>T<sub>E</sub>X file **file.tex**, using the **GL2PS\_TEX** format and specifying **file.tex** as the **filename** argument to **gl2psBeginPage**.

You can of course combine the L<sup>A</sup>T<sub>E</sub>X output with other graphic formats than PostScript. For example, you may transform **file.ps** into **file.pdf** and use pdfL<sup>A</sup>T<sub>E</sub>X with the same **file.tex** as for PostScript. You may also use a bitmap image (e.g. JPEG or PNG) and still combine it with **file.tex**.

`GL2PS_USE_CURRENT_VIEWPORT` The current OpenGL viewport is used instead of `viewport`.

`colormode` Specifies the color mode: `GL_RGBA` or `GL_COLOR_INDEX`.

`colormap` Specifies the size of the colormap if `colormode` is `GL_COLOR_INDEX`.

`colortable` Contains the colormap if `colormode` is `GL_COLOR_INDEX`. This colormap must contain `colormap` elements of type `GL2PSrgba`.

`nr, ng, nb` Control the number flat-shaded (sub-)triangles used to approximate a smooth-shaded triangle when the `shfill` operator is not supported by the system, or when the `GL2PS_NO_PS3_SHADING` option is set. The arguments `nr`, `ng` and `nb` specify the number of values used for interpolating the full range of red, green and blue color components; that is, a triangle is recursively subdivided until the color difference between two of its vertices is smaller than  $1/nr$  for the red component,  $1/ng$  for the green component and  $1/nb$  for the blue component. If the arguments are set to zero, default values are used.

`bufferize` Specifies the size of the feedback buffer.

`stream` Specifies the stream to which data is printed.

`filename` Specifies a name for the stream to which data is printed.

### 2.1.3 Return value

`gl2psBeginPage` returns:

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

`gl2psEndPage` returns:

`GL2PS_NO_FEEDBACK` if the feedback buffer is empty;

`GL2PS_OVERFLOW` if the size of the feedback buffer given to `gl2psBeginPage` is not large enough;

`GL2PS_UNINITIALIZED` if `gl2psEndPage` is called when the library is not initialized (e.g. if `gl2psEndPage` is called before `gl2psBeginPage`);

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

## 2.2 gl2psText

### 2.2.1 Specification

```
GLint gl2psText( const char *string, const char *fontname,
                 GLint fontsize )
```

### 2.2.2 Description and arguments

`gl2psText` permits to include text in the PostScript or  $\text{\LaTeX}$  output in a very simple way. The text is inserted at the current raster position (set by one of the `glRasterPos` OpenGL commands). Beware that text will be sorted according to the position of the leftmost element of the string only. The arguments are:

**string** Specifies the text string to print.

**fontname** Specifies the name of a valid PostScript font (for example "`Times`" or "`HelveticaBoldItalic`"). This has no influence for  $\text{\LaTeX}$  output.

**fontsize** Specifies the size of the font. This has no influence for  $\text{\LaTeX}$  output.

### 2.2.3 Return value

`gl2psText` returns:

`GL2PS_UNINITIALIZED` if **string** is NULL or if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` if no error occurred.

## 2.3 gl2psDrawPixels

### 2.3.1 Specification

```
GLint gl2psDrawPixels( GLsizei width, GLsizei height,
                      GLint xorig, GLint yorig,
                      GLenum format, GLenum type,
                      const void *pixels )
```

### 2.3.2 Description and arguments

`gl2psDrawPixels` emulates the `glDrawPixels` function, i.e. permits to include bitmap images in the PostScript output. The image is inserted at the current raster position (set by one of the `glRasterPos` OpenGL commands). Beware that the image will be sorted according to the position of the current raster position only. The arguments are:

**width** Specifies the width of the image.

**height** Specifies the height of the image.

**xorig, yorig** Specify the location of the origin in the image. The origin is measured from the lower left corner of the image, with right and up being the positive axes.

**format** Specifies the format of the pixel data. The symbolic constant `GL_RGB` is the only one accepted at the moment.

**type** Specifies the data type for pixels. The symbolic constant `GL_FLOAT` is the only one accepted at the moment.

**pixels** Specifies a pointer to the pixel data.

### 2.3.3 Return value

`gl2psDrawPixels` returns:

`GL2PS_UNINITIALIZED` if `pixels` is `NULL` or if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

## 2.4 `gl2psEnable` and `gl2psDisable`

### 2.4.1 Specification

```
GLint gl2psEnable( GLint mode )
```

```
GLint gl2psDisable( GLint mode )
```

### 2.4.2 Description and arguments

`gl2psEnable` and `gl2psDisable` delimit OpenGL commands to which a local mode is applied. These modes are:

`GL2PS_POLYGON_OFFSET_FILL` Tries to emulate the `GL_POLYGON_OFFSET_FILL` functionality. The value of the offset is taken as the current value of the corresponding OpenGL offset (set with `glPolygonOffset`). Not fully functional yet.

`GL2PS_POLYGON_BOUNDARY` Not implemented yet.

`GL2PS_LINE_STIPPLE` Tries to emulate the `GL_LINE_STIPPLE` functionality.

### 2.4.3 Return value

`gl2psEnable` and `gl2psDisable` return:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

## 2.5 `gl2psPointSize` and `gl2psLineWidth`

### 2.5.1 Specification

```
GLint gl2psPointSize( GLfloat value )
```

```
GLint gl2psLineWidth( GLfloat value )
```

### 2.5.2 Description and arguments

`gl2psPointSize` and `gl2psLineSize` emulate the standard `glPointSize` and the `glLineWidth` functions. They are necessary since the point sizes and line widths are not saved in the OpenGL feedback buffer.

### 2.5.3 Return value

`gl2psPointSize` and `gl2psLineWidth` return:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

## 2.6 `gl2psBeginViewport` and `gl2psEndViewport`

### 2.6.1 Specification

`GLint gl2psBeginViewport ( GLint viewport[4] )`

`GLint gl2psEndViewport ( void )`

### 2.6.2 Description and arguments

`gl2psBeginViewport` and `gl2psEndViewport` permit to output different viewports<sup>2</sup> in the output stream. Each viewport is sorted separately and has its own background frame. The argument given to `gl2psBeginViewport` specifies the viewport (obtained for example with a call to `glGetIntegerv(GL_VIEWPORT, viewport)`).

### 2.6.3 Return value

`gl2psBeginViewport` returns:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

`gl2psEndViewport` returns:

`GL2PS_NO_FEEDBACK` if the feedback buffer is empty;

`GL2PS_OVERFLOW` if the size of the feedback buffer given to `gl2psBeginPage` is not large enough;

---

<sup>2</sup>See the description of `glViewport` and `glScissor` in the OpenGL documentation.

GL2PS\_UNINITIALIZED if `gl2psEndViewport` is called when the library is not initialized;

GL2PS\_ERROR if an error occurred;

GL2PS\_SUCCESS otherwise.

### 3 Example

Here is a typical calling sequence to produce BSP sorted PostScript output in the file "MyFile", with all the lines slightly shifted front in the z-buffer and all invisible primitives removed to reduce the size of the output file. The `draw()` function contains all the OpenGL commands.

```
FILE *fp = fopen("MyFile", "w");
GLint bufsize = 0, state = GL2PS_OVERFLOW;
GLint viewport[4];

glGetIntegerv(GL_VIEWPORT, viewport);

while( state == GL2PS_OVERFLOW ){
    bufsize += 1024*1024;
    gl2psBeginPage ( "MyTitle", "MySoftware", viewport,
                    GL2PS_EPS, GL2PS_BSP_SORT,
                    GL2PS_SIMPLE_LINE_OFFSET | GL2PS_SILENT |
                    GL2PS_OCCLUSION_CULL | GL2PS_BEST_ROOT,
                    GL_RGBA, 0, NULL, 0, 0, 0, 0, bufsize,
                    fp, NULL );

    draw();
    state = gl2psEndPage();
}

fclose(fp);
```

To output the text "MyText" at the current raster position, the `draw()` function should contain something like:

```
gl2psText("MyText", "Courier", 12);
```

If you plan to convert the PostScript file into a PDF file you may need to disable the use of the Level 3 PostScript `shfill` operator, i.e. add `GL2PS_NO_PS3_SHADING` to the list of options. Note that you can also edit the output file a posteriori (just set `/tryPS3shading` to `false` in the PostScript file header).

A complete example (`gl2psTest.c`) is included in the distribution.



## 4 Limitations

GL2PS works by capturing the contents of the OpenGL feedback buffer<sup>3</sup>. As such, all the OpenGL operations applied in the pipeline after the creation of the feedback buffer will be ignored or have to be duplicated by GL2PS (e.g. font/image rendering, polygon offset or line stippling—see sections 2.2, 2.3, 2.4 and 2.5).

Other limitations include:

- Rendering large and/or complicated scenes is slow and/or can lead to large output files. This is normal: vector-based images are not destined to replace bitmap images. They just offer an alternative when high quality (especially for 2D and small 3D plots) and ease of manipulation (how do you change the scale, the labels or the colors in a bitmap picture long after the picture was produced, and without altering its quality?) are important.
- Transparency is not supported yet. See the `TODO` file for an explanation why this may or may not actually be possible.
- GL2PS does not support textures, fog effects, etc.

## 5 Contributors

Michael Sweet for the original implementation of the feedback buffer parser; Bruce Naylor for BSP tree and occlusion culling hints; Marc Umé for the original list code; Jean-François Remacle for plane equation fixes; Bart Kaptein for memory leak fixes; Quy Nguyen-Dai for output file size optimization; Sam Buss for the `shfill`-based smooth shaded triangle code; Shane Hill for the landscape option implementation; Romain Boman for the Windows dll generation; Diego Santa Cruz for the new optimized shaded triangle code and the `shfill` management; Shahzad Muzaffar and Lassi Tuura for the new occlusion culling code and the improvement of `GL2PS_BEST_ROOT`; Guy Barrand for his work on `gl2psDrawPixels` and the new viewport management; Rouben Rostamian and Prabhu Ramachandran for various bug reports and fixes.

## 6 Links

Projects similar to GL2PS include: Mark J. Kilgard's `rendereps` (<http://www.opengl.org/developers/code/mjktips/Feedback.html>); Michael Sweet's GLP library (<http://www.easysw.com/~mike/opengl/index.html>); the GLpr library from CEI international (<http://www.ceintl.com/>; this product does not seem to be available anymore).

---

<sup>3</sup>See the description of `glFeedbackBuffer` and `glRenderMode(GL_FEEDBACK)` in the OpenGL documentation.

Toby White maintains a Python wrapper for GL2PS, available at <http://tow.freeshell.org/software.html>.

## 7 Versions

- 0.1** (Feb 12, 2000) First distributed version.
- 0.2** (Feb 20, 2000) Added `GL2PS_POLYGON_BOUNDARY` and `GL2PS_BEST_ROOT`.  
API change: changed arguments of `gl2psBeginPage` and `gl2psText`. Corrected some memory allocation stuff. First version of this user's guide.
- 0.21** (Mar 16, 2000) Initialization fixes.
- 0.3** (Jul 29, 2000) Code cleanup. Added `GL2PS_LINE_STIPPLE`.
- 0.31** (Aug 14, 2000) Better handling of erroneous primitives.
- 0.32** (May 23, 2001) Fixed memory leaks.
- 0.4** (Jun 12, 2001) Added `gl2psPointSize` and `gl2psLineWidth`. Some code cleanup to allow easier generation of vector file formats other than postscript.
- 0.41** (Aug 6, 2001) Fixed string allocation (1 char too short). Set smaller default line width.
- 0.42** (Oct 8, 2001) Optimization of output file size. PostScript header cleanup. Better line width computation.
- 0.5** (Nov 19, 2001) API change: new `format` and `filename` arguments for `gl2psBeginPage`. Better PostScript handling of smooth shaded primitives. Fix handling of zero-length strings. New options for  $\text{\LaTeX}$  output. Changed (again) the line width computation.
- 0.51** (Jan 22, 2002) Fixed erroneous drawing of text primitives lying outside the viewport.
- 0.52** (Feb 14, 2002) New `GL2PS_LANDSCAPE` option.
- 0.53** (Mar 11, 2002) New `GL2PSDLL` compilation flag to allow the generation of a Windows dll.
- 0.6** (Jun 4, 2002) Fixed some incoherences in string allocation; fixed sorting of text objects; removed (non functional) occlusion culling code; fixed handling of color and line width attributes when `gl2ps` was called multiple times inside the same program.
- 0.61** (Jun 21, 2002) Fixed the fix for the sorting of text objects; introduced tolerance for floating point comparisons.

- 0.62** (Sep 6, 2002) New `GL2PS_EPS` option to produce Encapsulated PostScript files; optimized drawing of shaded primitives; new `GL2PS_NO_PS3_SHADING` option and `gl2psNumShadeColors` function to control the use of the PostScript level 3 `shfill` operator (usually not well handled when converting to PDF).
- 0.63** (Nov 12, 2002) Changed `GLvoid` to `void` to accommodate some SUN compilers; made subdivision parameters modifiable a posteriori in the output file; revised documentation.
- 0.7** (Dec 11, 2002) Occlusion culling (`GL2PS_OCCLUSION_CULL`) is (finally!) working thanks to the great work of Shahzad Muzaffar; enhanced `GL2PS_BEST_ROOT`.
- 0.71** (Dec 13, 2002) Removed C++ style comments inadvertently left in the code; added example program `gl2psTest.c` to the distribution.
- 0.72** (Jan 21, 2003) Fixed crash in occlusion culling code; enhanced documentation.
- 0.73** (Jan 30, 2003) Minor code cleanup.
- 0.8** (Mar 10, 2003) API change: `gl2psNumShadeColors` has been removed and the color subdivision parameters `nr`, `ng` and `nb` are now given as arguments to `gl2psBeginPage`; API change: `gl2psBeginPage` takes an additional argument (`viewport`) to specify the print viewport; new `gl2psDrawPixels` interface to produce mixed mode PostScript/bitmap output; new `gl2psBeginViewport` and `gl2psEndViewport` interface to handle multiple OpenGL viewports; fixed small bug in occlusion culling code; better error handling.
- 0.81** (Mar 22, 2003) Fixed small typos in comments and documentation.
- 0.9.0** (Jun 2, 2003) Fixed smooth shading detection for mixed smooth/flat shaded scenes; new library numbering scheme (“major.minor.patch”).
- 0.9.1** (Jun 12, 2003) Fixed two `GL2PS_TEX` output bugs (`glRenderMode` not reset to `GL_RENDER` + crash when printing empty scenes); changed default pixmap depth to 8 bits per color component; changed default line cap to “Butt cap” and default line join to “Miter join”.
- 0.9.2** (Jul 4, 2003) Improved occlusion culling; new `GL2PS_USE_CURRENT_VIEWPORT` option.