GL2PS: an OpenGL to PostScript printing library

Christophe Geuzaine

Version 1.3.7, September 2 2012

Contents

1 Introduction **1**

2 Usage **2**
2.1 `gl2psBeginPage` and `gl2psEndPage` 2
2.2 `gl2psText` and `gl2psTextOpt` 5
2.3 `gl2psDrawPixels` 7
2.4 `gl2psSpecial` 7
2.5 `gl2psEnable` and `gl2psDisable` 8
2.6 `gl2psPointSize` and `gl2psLineWidth` 9
2.7 `gl2psBlendFunc` 9
2.8 `gl2psBeginViewport` and `gl2psEndViewport` 10
2.9 `gl2psSetOptions` and `gl2psGetOptions` 10

3 Example **11**

4 Tips and tricks **12**

5 Limitations **13**

6 Contributors **13**

7 Links **14**

8 Versions **14**

1 Introduction

GL2PS is a C library providing high quality vector output for any OpenGL application. The main difference between GL2PS and other similar libraries (see section 7) is the use of sorting algorithms capable of handling intersecting

and stretched polygons, as well as non manifold objects. GL2PS provides advanced smooth shading and text rendering, culling of invisible primitives, mixed vector/bitmap output, and much more...

GL2PS can currently create PostScript (PS), Encapsulated PostScript (EPS), Portable Document Format (PDF) and Scalable Vector Graphics (SVG) files, as well as LaTeX files for the text fragments. GL2PS also provides limited, experimental support for Portable LaTeX Graphics (PGF). Adding new vector output formats should be relatively easy; you can also use the excellent pstoedit program to transform the PostScript files generated by GL2PS into many other vector formats such as xfig, cgm, wmf, etc.

GL2PS is available at `http://www.geuz.org/gl2ps/` and is released under the GNU Library General Public License (see COPYING.LGPL). GL2PS can also be used under an alternative license that allows (amongst other things, and under certain conditions) for static linking with closed-source software (see COPYING.GL2PS). Any corrections, questions or suggestions should be e-mailed to the GL2PS mailing list gl2ps@geuz.org.

The interface consists of fifteen 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 5) 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 in PostScript format.
- `GL2PS_EPS` The output stream will be in Encapsulated PostScript format.
- `GL2PS_PDF` The output stream will be in Portable Document Format.
- `GL2PS_TEX` The output will be a LaTeX file containing only the text strings of the plot (cf. section 2.2), as well as an `\includegraphics` command including a graphic file having the same basename as `filename`.[1]
- `GL2PS_SVG` The output stream will be in Scalable Vector Graphics format.
- `GL2PS_PGF` (Experimental) The output stream will be in Portable LaTeX Graphics format.

`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. This is sufficient for two-dimensional scenes.
- `GL2PS_SIMPLE_SORT` The primitives are sorted according to their barycenter. This can be sufficient for simple three-dimensional scenes and/or when correctness is not crucial.
- `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 whenever an accurate rendering of a three-dimensional scene is sought. Beware that this algorithm requires a lot more computational time (and memory) than the simple barycentric sort.

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

[1] The two steps to generate a LaTeX plot with GL2PS are thus:

1. generate the PostScript or PDF file (e.g. `file.ps` or `file.pdf`) with no text strings, using the `GL2PS_PS`, `GL2PS_EPS` or `GL2PS_PDF` format combined with the `GL2PS_NO_TEXT` option;
2. generate the LaTeX file `file.tex`, using the `GL2PS_TEX` format and specifying `file.tex` as the `filename` argument to `gl2psBeginPage`.

You can of course combine the LaTeX output with other graphic formats than PostScript or PDF. For example, you could export an image in JPEG or PNG format and use pdfLaTeX with the same `file.tex`.

`GL2PS_NONE` No option.

`GL2PS_DRAW_BACKGROUND` The background frame is drawn in the plot.

`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 2.5), 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 the output stream. This is useful to produce the image part of a LaTeX plot.

`GL2PS_NO_PIXMAP` All the pixmaps are suppressed from the output stream.

`GL2PS_LANDSCAPE` The plot is output in landscape orientation instead of portrait.

`GL2PS_NO_PS3_SHADING` (for PostScript output only) No use is made of the `shfill` PostScript level 3 operator. Using `shfill` enhances the plotting of smooth shaded primitives but can lead to problems when converting PostScript files into PDF files. See also options `nr`, `ng`, `nb` below.

`GL2PS_NO_BLENDING` Blending (transparency) is disabled alltogether (regardless of the current `GL_BLEND` or `GL2PS_BLEND` status).

`GL2PS_OCCLUSION_CULL` All the hidden polygons are removed from the output, thus substantially reducing the size of the output file.

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

`GL2PS_TIGHT_BOUNDING_BOX` The viewport is ignored and the the plot is generated with a tight bounding box, i.e., a bounding box enclosing as tightly as possible all the OpenGL entities in the scene.

`GL2PS_COMPRESS` The output stream is compressed. For this option to take effect you need to compile GL2PS with `HAVE_ZLIB`, `HAVE_LIBZ` or `GL2PS_HAVE_ZLIB` defined, and link the executable with the zlib library (http://www.gzip.org/zlib/).

PostScript or SVG files generated with this option turned on are simply compressed "as a whole", i.e., they are identical to regular PostScript or SVG files compressed with the gzip program. For PDF files the compression is done "locally" for each group of primitives, in accordance with the official PDF specification.

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

`colorsize` 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 `colorsize` elements of type `GL2PSrgba`.

`nr, ng, nb` (for PostScript and SVG output only) Controls the number of flat-shaded (sub-)triangles used to approximate a smooth-shaded triangle. (For PostScript output, this is only used 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 that 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.

`buffersize` Specifies the size of the feedback buffer.

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

`filename` (for LaTeX output only) 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` and `gl2psTextOpt`

2.2.1 Specification

```
GLint gl2psText( const char *string, const char *fontname,
                 GLint fontsize )
GLint gl2psTextOpt( const char *string, const char *fontname,
                    GLint fontsize, GLint align, GLfloat angle )
```

2.2.2 Description and arguments

`gl2psText` and `gl2psTextOpt` permit to include a text string in the output stream. The string is inserted at the current raster position (set by one of the `glRasterPos` OpenGL commands). Beware that text will be sorted according to the current raster position only. The arguments are:

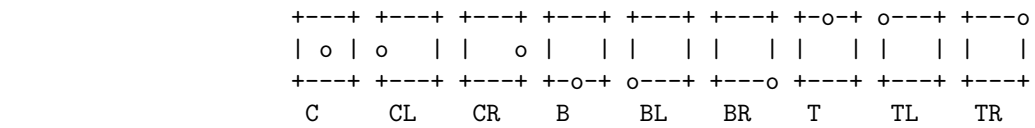
`string` Specifies the text string to print.

`fontname` Specifies the PostScript name of a valid Type 1 font[2]. This has no effect for LaTeX and PGF output.

`fontsize` Specifies the size of the font.

The additional arguments for `gl2psTextOpt` are:

`align` (for PostScript, LaTeX and PGF output only) Specifies the text string alignment with respect to the current raster position. Valid choices are `GL2PS_TEXT_C` (center-center), `GL2PS_TEXT_CL` (center-left), `GL2PS_TEXT_CR` (center-right), `GL2PS_TEXT_B` (bottom-center), `GL2PS_TEXT_BL` (bottom-left) `GL2PS_TEXT_BR` (bottom-right), `GL2PS_TEXT_T` (top-center), `GL2PS_TEXT_TL` (top-left) and `GL2PS_TEXT_TR` (top-right). The default alignment used by `gl2psText` is `GL2PS_TEXT_BL`.



`angle` (for PostScript, LaTeX and PGF output only) Specifies a rotation angle for the text string (counter-clockwise, in degrees).

2.2.3 Return value

`gl2psText` and `gl2psTextOpt` return:

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

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

[2]The names of the 14 standard Type 1 fonts are as follows: `Times-Roman`, `Times-Bold`, `Times-Italic`, `Times-BoldItalic`, `Helvetica`, `Helvetica-Bold`, `Helvetica-Oblique`, `Helvetica-BoldOblique`, `Courier`, `Courier-Bold`, `Courier-Oblique`, `Courier-BoldOblique`, `Symbol` and `ZapfDingbats`. These fonts, or their font metrics and suitable substitution fonts, are guaranteed to be available to the viewer application. Using any other font will result in a non-portable PostScript or PDF file, as GL2PS does not include any font description in its output stream.

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 embed bitmap images in the PostScript, PDF or SVG output. To embed bitmaps in SVG output, GL2PS needs to be compiled with `HAVE_LIBPNG` or `GL2PS_HAVE_LIBPNG` defined, and the executable must be linked with the PNG library (http://www.libpng.org).

The bitmap 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. `GL_RGB` and `GL_RGBA` are the only values accepted at the moment.

`type` Specifies the data type for pixels. `GL_FLOAT` is the only value 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 gl2psSpecial

2.4.1 Specification

```
GLint gl2psSpecial( GLint format, const char *str )
```

2.4.2 Description and arguments

`gl2psSpecial` permits to output an arbitrary command string in an output stream of a given format. The arguments are:

`format` Specifies the output format for which the special string will be printed. If the stream format (specified with `gl2psBeginPage`, see Section 2.1) does not match `format`, the command has no effect.

`str` Specifies the string to print.

2.4.3 Return value

`gl2psSpecial` returns:

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

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

2.5 `gl2psEnable` and `gl2psDisable`

2.5.1 Specification

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

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

2.5.2 Description and arguments

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

`GL2PS_LINE_STIPPLE` Emulates the `GL_LINE_STIPPLE` functionality. The stippling pattern and repetition factor are taken as the current values of the corresponding OpenGL stippling options (set with `glLineStipple`). You thus need to call `gl2psEnable(GL2PS_LINE_STIPPLE)` *after* calling `glLineStipple(factor,pattern)`.

`GL2PS_POLYGON_OFFSET_FILL` Emulates 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`).

`GL2PS_BLEND` Emulates the `GL_BLEND` functionality. (Warning: this might change in future releases.)

`GL2PS_POLYGON_BOUNDARY` Not implemented yet.

2.5.3 Return value

`gl2psEnable` and `gl2psDisable` return:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

2.6 `gl2psPointSize` and `gl2psLineWidth`

2.6.1 Specification

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

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

2.6.2 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.

2.6.3 Return value

`gl2psPointSize` and `gl2psLineWidth` return:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

2.7 `gl2psBlendFunc`

2.7.1 Specification

```
GLint gl2psBlendFunc( GLenum sfactor, GLenum dfactor )
```

2.7.2 Description and arguments

`gl2psBlendFunc` emulates the `glBlendFunc` function.

2.7.3 Return value

`gl2psBlendFunc` returns:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`GL2PS_WARNING` if the blending mode is not (yet) supported;

`GL2PS_SUCCESS` otherwise.

2.8 gl2psBeginViewport and gl2psEndViewport

2.8.1 Specification

```
GLint gl2psBeginViewport ( GLint viewport[4] )

GLint gl2psEndViewport ( void )
```

2.8.2 Description and arguments

`gl2psBeginViewport` and `gl2psEndViewport` permit to output different viewports[3] 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)`).

2.8.3 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;

`GL2PS_UNINITIALIZED` if `gl2psEndViewport` is called when the library is not initialized;

`GL2PS_ERROR` if an error occurred;

`GL2PS_SUCCESS` otherwise.

2.9 gl2psSetOptions and gl2psGetOptions

2.9.1 Specification

```
GLint gl2psSetOptions ( GLint options )
GLint gl2psGetOptions ( GLint *options )
```

[3] See the description of `glViewport` and `glScissor` in the OpenGL documentation.

2.9.2 Description and arguments

`gl2psSetOptions` permits to change the global options initially set using the `options` argument to `gl2psBeginPage` (see section 2.1). `gl2psGetOptions` permits to retrieve the current options.

`gl2psSetOptions` can for example be used to force GL2PS to print the background for selected viewports, by setting/unsetting `GL2PS_DRAW_BACKGROUND` before calling `gl2psBeginViewport`.

2.9.3 Return value

`gl2psSetOptions` and `gl2psGetOptions` return:

`GL2PS_UNINITIALIZED` if the library is not initialized;

`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", "wb");
GLint buffsize = 0, state = GL2PS_OVERFLOW;
GLint viewport[4];

glGetIntegerv(GL_VIEWPORT, viewport);

while( state == GL2PS_OVERFLOW ){
  buffsize += 1024*1024;
  gl2psBeginPage ( "MyTitle", "MySoftware", viewport,
                   GL2PS_EPS, GL2PS_BSP_SORT, GL2PS_SILENT |
                   GL2PS_SIMPLE_LINE_OFFSET | GL2PS_NO_BLENDING |
                   GL2PS_OCCLUSION_CULL | GL2PS_BEST_ROOT,
                   GL_RGBA, 0, NULL, 0, 0, 0, buffsize,
                   fp, "MyFile" );
  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);
```

Complete example programs (`gl2psTestSimple.c` and `gl2psTest.c`) are included in the distribution.

4 Tips and tricks

Here are, in no particular order, some useful tips and solutions to common problems:

- For PDF (both compressed and non-compressed) and for compressed PostScript and SVG output, files should always be opened in binary mode, i.e., with `fopen(...,"wb")`, instead of `fopen(...,"w")`.

- Blending is not yet very well supported by many viewers/printers. To disable blending entirely, add `GL2PS_NO_BLENDING` to the list of options passed to `gl2psBeginPage`.

- Make sure that localization is turned off when using GL2PS, via:

  ```
  unsigned char *oldlocale = setlocale(LC_NUMERIC, "C");

  /* gl2ps drawing stuff */

  setlocale(LC_NUMERIC, oldlocale);
  ```

  French or German localizations would for example lead to corrupted output files, as they represent the decimal point by a comma.

- If you plan to convert PostScript files into PDF files, 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 passed to `gl2psBeginPage`. (Note that you can also edit the output file *a posteriori*—just set `/tryPS3shading` to `false` in the PostScript file header.) The best way to generate PDF files is of course to set the `format` argument to `GL2PS_PDF` in the `gl2psBeginPage` call...

- By default, GL2PS checks if blending is globally enabled in `gl2psBeginPage()`. To enable blending for selected primitives only, you should use `gl2psEnable(GL2PS_BLEND)` and `gl2psDisable(GL2PS_BLEND)` pairs around the OpenGL calls that need blending. (Warning: this might change in future releases.)

- `gl2psEnable(GL2PS_LINE_STIPPLE)` uses the current values of the OpenGL stippling options to compute the stippling pattern and repetition factor. You thus need to call `gl2psEnable(GL2PS_LINE_STIPPLE)` *after* calling `glLineStipple(factor,pattern)`.

5 Limitations

GL2PS works by capturing the contents of the OpenGL feedback buffer[4]. 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, 2.5 and 2.6).

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 only supported for PDF and SVG output.
- GL2PS does not support textures, fog effects, etc.

6 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, the improvement of `GL2PS_BEST_ROOT` and the imagemap support; Guy Barrand for his work on `gl2psDrawPixels` and the new viewport management; Rouben Rostamian and Prabhu Ramachandran for various bug reports and fixes; Micha Bieber for the PDF code; Olivier Couet for the initial SVG code; Fabian Wenzel for the PGF code and the backend reorganization; Shai Ayal for rotated text support in PostScript; Ian D. Gay for 64 bit arch patch; Cosmin Truta and Baiju Devani for various bug fixes and the new `gl2psSpecial` code; Alexander Danilov for a polygon offset bug fix; Ben Fletcher for a stippling pattern parser bug report; Jason Anderssen for memory leak fix in pdf code; Sylvestre Ledru for SVG patches; Calixte Denizet for 64 bit patch; Ion Vasilief and Paul Griffiths for rotated text in PDF output; Ben Abbott for text alignment in SVG.

[4] See the description of `glFeedbackBuffer` and `glRenderMode(GL_FEEDBACK)` in the OpenGL documentation.

7 Links

GL2PS was inspired by Mark Kilgard's original "rendereps" tutorial (http://www.opengl.org/resources/code/samples/mjktips/Feedback.html) and Michael Sweet's GLP library (http://www.easysw.com/~mike/opengl/). The (commercial) GLpr library from CEI (http://www.ceintl.com/) used to provide functionality similar to GL2PS but does not seem to be available anymore.

8 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 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 (vector+raster) PostScript 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.

1.0.0 (Sep 24, 2003) Native PDF support contributed by Micha Bieber.

1.1.0 (Nov 4, 2003) New `GL2PS_COMPRESS` option to create compressed PostScript and PDF files; fixed small bug in the PDF output that prevented the PDF files to be correctly included in LaTeX documents; new alternative license (see `COPYING.GL2PS`).

1.1.1 (Nov 9, 2003) Small memory optimization; documentation update (binary files, fonts).

1.1.2 (Nov 16, 2003) Fixed various compiler warnings (mostly for Windows Visual C++).

1.2.0 (May 13, 2004) New (experimental...) transparency support for PDF output; fixed bug for empty feedback buffer but non-empty primitive list; fixed more compiler warnings and cleaned up the code (mostly to reduce the global namespace pollution).

1.2.1 (Jul 13, 2004) New imagemap support for PostScript output; new text alignment support for PostScript and LaTeX output; new support for rotated text for LaTeX output; fixed NULL check on input strings in `gl2psBeginPage`.

1.2.2 (Sep 21, 2004) Fixed a couple of small bugs in the example code.

1.2.3 (Dec 23, 2004) Fixed small bugs in (unused) PostScript pixmap code; better scaling of the z-buffer (improves `GL2PS_SIMPLE_LINE_OFFSET` and occlusion culling); added support for general stippling patterns.

1.2.4 (Apr 27, 2005) Fixed feedback buffer test for `GL2PS_TEX` output; fixed missing brace in LaTeX output for text aligned using `GL2PS_TEXT_C`; fixed clipping in multi-viewport PostScript output when `GL2PS_DRAW_BACKGROUND` is not set; new `gl2psSetOptions` interface to change the current options on the fly.

1.2.5 (Jun 18, 2005) Fixed a couple of uninitialized variables in PDF code; new `GL2PS_TIGHT_BOUNDING_BOX` option; added rotated text support for PostScript output.

1.2.6 (Jun 22, 2005) Fixed crash when creating PDF file with overflowing feedback buffer (bug introduced in 1.2.5); added additional example program `gl2psTestSimple.c` to the distribution.

1.2.7 (Feb 15, 2006) Fixed bug that could cause sorting to be inverted in BSP mode (bug introduced in 1.2.3); added limited support for SVG and PGF formats; made backend code more generic.

1.3.0 (Aug 8, 2006) Full SVG support; improved line stippling (whenever possible lines are now rendered in a single path); better sorting of text and bitmap elements; new function `gl2psSpecial` to print device-specific strings in the output stream.

1.3.1 (Aug 11, 2006) Fixed a small bug for multi-viewport PostScript output, where a viewport could be drawn with an unwanted colored outline.

1.3.2 (Nov 5, 2006) Fixed bug in polygon offset computation; fixed landscape mode in SVG; fixed potential out-of-bounds array access in stippling pattern parser.

1.3.3 (Feb 28, 2009) Fixed memory leak in PDF code; added `gl2psGetOptions`; nicer SVG rendering (crispEdges, font attributes); fixed possible divisions by zero.

1.3.4 (Sep 30, 2009) Added support for rotated text in SVG output; fixed MSVC warnings.

1.3.5 (Oct 16, 2009) Added support for rotated text in PDF output; fixed PDF image output when compiled in 64 bit mode; added cmake configuration.

1.3.6 (Aug 14, 2011) Added support for PixelZoom (set using glPixelZoom) in postscript images; fixed text rotation in TeX; added support for text alignment in SVG; fixed other minor bugs.

1.3.7 (Sep 2, 2012) Minor documentation and build system fixes.